Paco

Release 9.3.28

March 04, 2022

Get Started with Paco

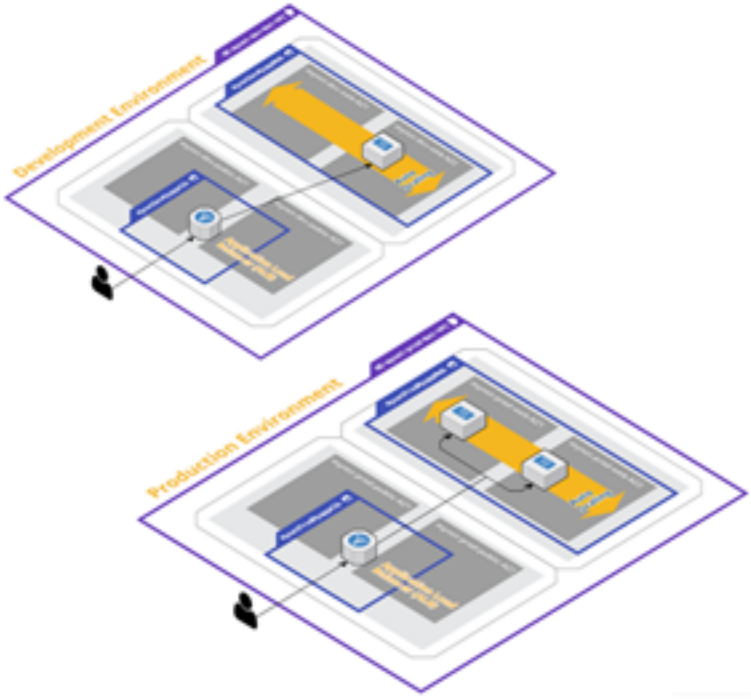
```
$ paco provision netenv.anet.dev
Loading Paco project at /Users/name/my-paco-project
MFA Token: master: 555555
Object selected to validate:
  Name: dev
  Type: Environment
  Title: Development Environment
  Reference: netenv.anet.dev
Provision   NetEnv      anet
Provision   Environment dev eu-central-1
Provision   dev         Net-VPC
```

100% automation of the creation and management of your cloud resources.

Define your cloud with declarative YAML.

Consistent semantic structure for all your cloud projects.

Automate your AWS cloud with an Infrastructure as Code project - without writing any code. Paco includes all the code you need built-in to the tool, so you only need to create declarative configuration to describe your cloud from at a high level.



CHAPTER 1

Turnkey AWS solutions, repeatable environments and DRY configuration

Complete turnkey solutions out-of-the box.

Easily clone complete environments.

Don't Repeat Yourself and override only the differences between environments.

Start with complete turnkey AWS solutions and customize to your needs. Begin with complete networks with reference applications that include monitoring, alerting, centralized logging already configured.

Reproduce your development, staging and production environments exactly. DRY configuration allows you to override just the differences between environments - easily see just what is different in each environment.

CHAPTER 2

Prescribed automation instead of repetitive coding

Monitoring, alerting and centralized logging all built-in.

Automatic secure mutli-account IAM user, policy and role management.

EC2 Launch Manager configures EC2 instances easily and quickly.

Paco is an opinionated cloud orchestration tool that believes that routine, boilerplate automation should be abstracted away by tooling. If you configure an S3 Bucket to notify a Lambda function, why should you have to create a Lambda Permission by hand? Mounting an EFS filesystem as easy as creating a Paco Reference for your AutoScalingGroup resource to an EFS resource. List in-host metrics and logs and your EC2 instances will have a CloudWatch agent automatically configured to collect them.

```
network:
  title: My Network
  enabled: true

applications:
  wordpress:
    title: "WordPress for example.com"
  saas:
    title: "SaaS for saas.example.com"

environments:
  dev:
    us-west-2:
      applications:
        wordpress:
        saas:
  prod:
    us-west-2:
      applications:
        wordpress:
        saas:
    eu-central-1:
        saas:
```

CHAPTER 3

Declarative semantic configuration for your cloud

Declarative configuration needed only.

Semantic high-level concepts such as applications, environments, networks and accounts.

Paco References allow you to describe your resources coupling without hard-coding.

Resources tagged 100% consistently based on your semantic configuration.

Declarative configuration gives you predictability and repeatability in the automation of your cloud resources. See configuration changes as a whole before you apply them to your cloud resources.

CHAPTER 4

Install Paco and get started today

Paco is free and open source with the source code on GitHub.

Get started by installing Paco and connecting it to your AWS account.

CHAPTER 5

Paco community

Join us on reddit at r/paco_cloud.

Ask us questions and chat with us on paco-cloud on gitter.

CHAPTER 6

Waterbear Cloud

PACO is developed by Waterbear Cloud. Contact us about our support and consulting professional services.

6.1 How to install Paco

6.1.1 Install Python

Paco is written in Python and requires Python 3.6 or greater.

Paco currently works with macos and Linux. **Windows support is not yet available.**

Get the latest version of Python from python.org or with your operating systems package manager. Some helpful links for specific operating systems:

- Python on macos
- Python on Ubuntu 16.04 LTS

Verify your Python version on your shell by typing `python` (or sometimes `python3`):

```
Python 3.x.y
[GCC 4.x] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

6.1.2 Install Paco

Paco can be installed with any Python package manager. Pip is the most popular and often comes with your Python installer. The Paco project is named `paco-cloud` on PyPI, to install it simply type:

```
$ pip install paco-cloud
```

You should now have the `paco` application installed:

```
$ paco --help
Usage: paco [OPTIONS] COMMAND [ARGS]...

  Paco: Prescribed Automation for Cloud Orchestration
...
```

6.2 Getting Started with Paco

Once you have the *paco* command-line installed, to get up and running you will need to:

1. Create a Paco project.
2. Create an IAM User and Role in your AWS account.
3. Connect your Paco project with your IAM User and Role.

6.2.1 Create a Paco project

The `paco init` command is there to help you get started with a new Paco project. It will let you create a new Paco project from a template and connect that project to your AWS account(s).

First you will use the `paco init project` command to create a new project. This command takes as a single arguement the name of directory to create with your new Paco project files. Run it with:

```
$ paco init project <my-paco-project>
```

You will be presented with a series of questions about your new project.

You will be asked to supply some `name` and `title` values. Paco makes an important distinction between a `name` field and a `title` field. The `name` fields are used to construct unique resource names in AWS, while `title` is for human-readable descriptions.

Note: Name guidelines in Paco

1. **AWS resources have different character set restrictions.** We recommend using only alphanumeric charcters and the hyphen character in names (a-zA-Z-).
2. **Try to limit names to only 3 to 5 characters.** Paco `name` fields are concatenated together to create unique names. Certain AWS resources names are limited to only 32 characters. If you use long names they may be too long for AWS.
3. **Names can not be changed after they provsion AWS resources.** Names identify resources in AWS. Once you use Paco to create resources in AWS, if you change `name` fields Paco will no longer know where those resources are. The only way to change a `name` field is to delete the resources, change the name, and create new ones.

An example set of answers for creating a Paco project:

```
project_title: My Paco Project
network_environment_name: ne
network_environment_title: My Paco Network
application_name: app
application_title: My Application
aws_default_region: us-west-2
master_account_id: 123456789012
master_root_email: you@example.com
```

After this you will have a new directory of files that comprises and Paco project.

The path to this Paco Project directory is called your PACO home. The rest of the commands you run will need this path supplied with the *–home* CLI option. For macos and linux users, there is also a file named *profile.sh* which will export an *PACO_HOME* environment variable to your shell. This environment variable can be used to make it easier by avoiding the need to type out the *–home* option for every command:

```
$ source my-paco-project/profile.sh
(My AWS Paco Project) laptop username$
```

6.2.2 Create a User and Role in your AWS account

When you run Paco it requires access to your AWS account.

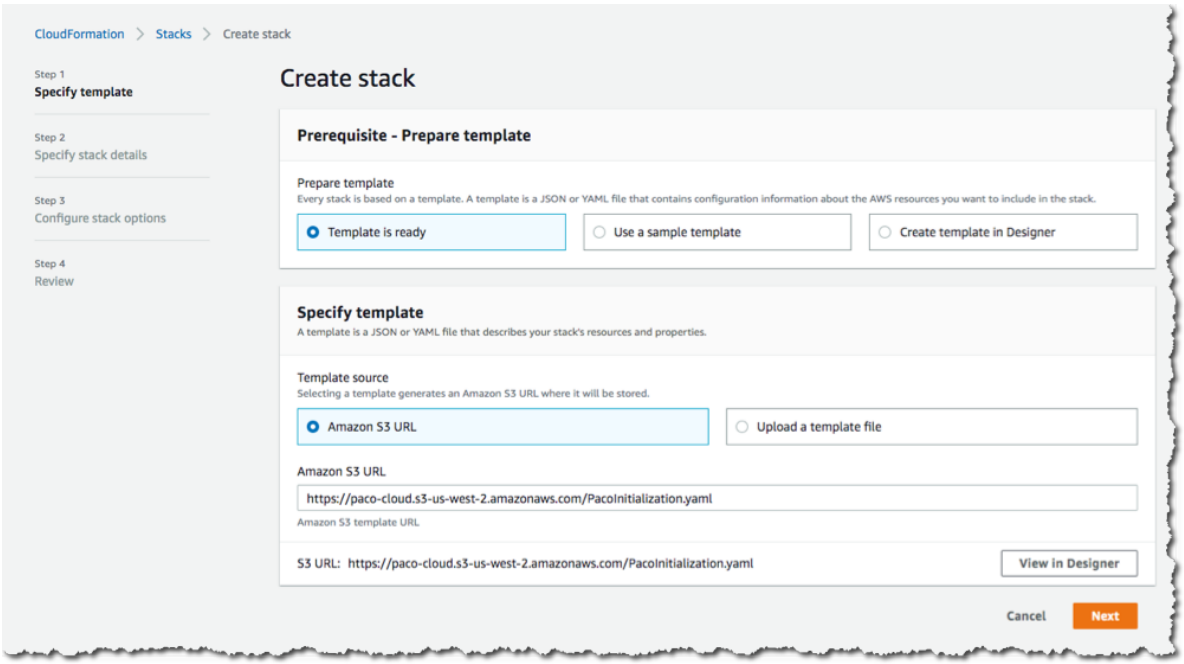
Paco requires access key credentials for an IAM User that has permissions to switch to an IAM Role that delegates full Administrator access.

Note: Why can't I just use any AWS Access Key with Administrator access with Paco?

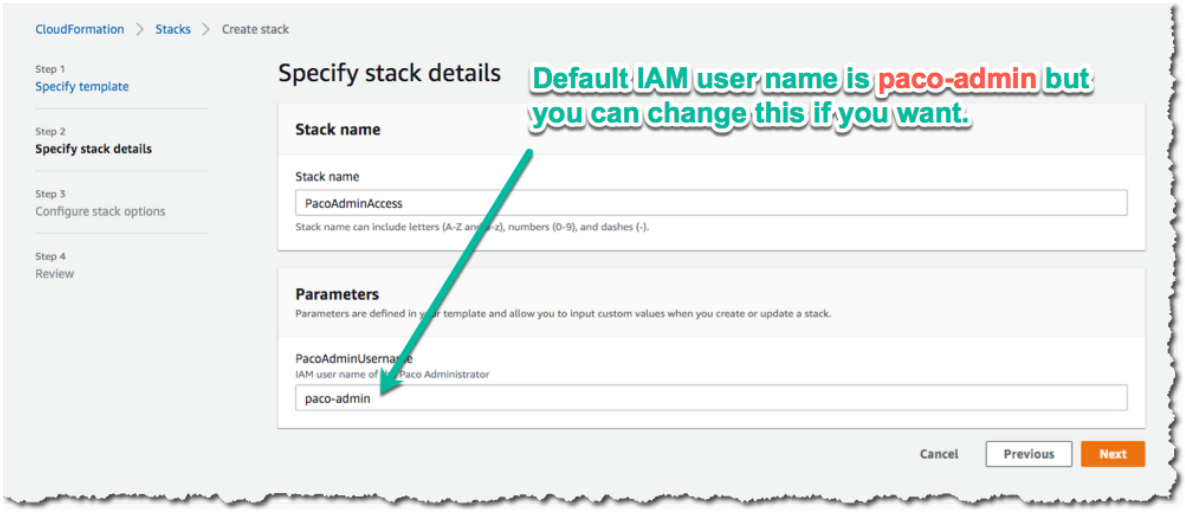
Paco requires an IAM User capable of switching to a Role that contains Administrator permissions. Paco does this for security reasons. Paco will ask you for your MFA token from the CLI. As you store an AWS Access Key and Secret in a Paco `.credentials` file, if this file is accidentaly leaked then unwanted users will not be able to use your key without also being able to access your MFA device.

To install a CloudFormation template that will create a User and Role to use with Paco.

1. Click on this URL to create a PacoAdminAccess CloudFormation stack in your AWS Account.



2. Click "Next" and take note that you will create a IAM User with the name `paco-admin`. If you like you can change this username here.

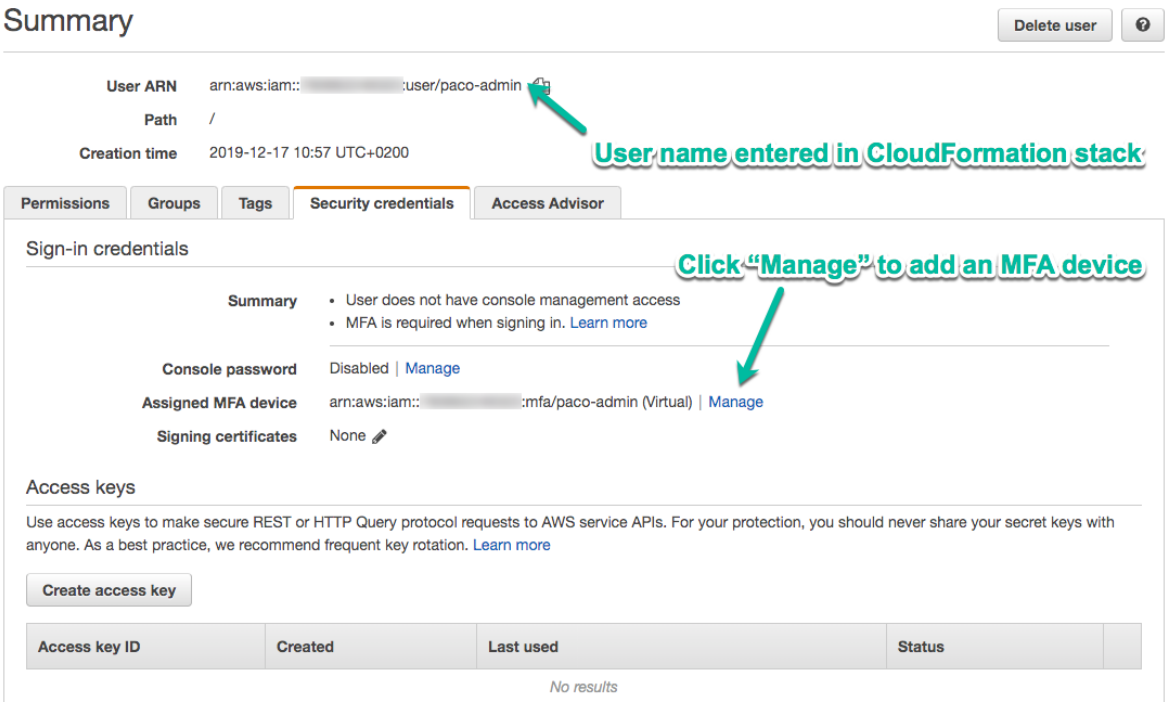


3. On the "Configure stack options" screen you can leave everything default and click "Next". On the "Review PacoInitialization" you can also leave all the defaults click "I acknowledge that AWS CloudFormation might create IAM resources with custom names." to confirm that this stack can create an IAM User. Finally click "Create stack".

6.2.3 Create an AWS Access Key and configure MFA

Next you will need to set-up the new User account with an API key:

1. In the AWS Console, go to the Identity and Access Management (IAM) Service, click on "Users" and click on the User name you supplied earlier. Then click on the "Security credentials" tab.



2. Set-up multi-factor authentication (MFA). Where it says, "Assigned MFA device" click on "Manage". Choose "Virtual MFA device" and use either Authy or Google Authenticator on your computer or phone as a virtual MFA device.

3. Create an AWS Access Key. While still on the "Security credentials" tab, click on "Create access key". You will be given an "Access key ID" and "Secret access key". Copy these and you will use them to configure your Paco credentials next.

Note: If you no longer want to use Paco, you can go to CloudFormation and delete the stack that you created. However, before you delete the stack, you will need to return to this user and manually delete the Assigned MFA Device and Access key. If you try and delete the stack without doing this first, you will get the error message "DELETE_FAILED: Cannot delete entity, must delete MFA device first.".

6.2.4 Connect your Paco project with your AWS account

Next use the `paco init credentials` command to initialize your credentials. Enter the name of your IAM User if you used the CloudFormation template your role name will be `Paco-Admin-Delegate-Role`.

```
$ paco init credentials --home=/path/to/your-paco-project

Paco project credentials initialization
---------------------------------------

Paco Admin Username: [paco-admin]:
AWS Access Key: KAKIA**********4MXP
AWS Secret Key: 56aU****************57cT
Paco credentials file created at:

/Users/bob/paco-project/.credentials.yaml
```

(continues on next page)

(continued from previous page)

```
It is NOT recommended to store this file in version control.
Paco starter project include a .gitignore file to prevent this.
You can store this file in a secrets mananger or re-create it again
by generating a new AWS Api Key for the Paco Admin User and re-running
this 'paco init credentials' command.
```

This will create a file named `.credentials` in your Paco project directory. Starting Paco projects also have a `.gitignore` file that will prevent you from committing this credentials file to a git repo. You can save this file somewhere secure, or if it is lost use the AWS Console to create a new acccess key for your IAM User and re-run `paco init credentials` to generate a new `.credentials` file.

Finally, use the `paco validate` command to verify your credentials allow you to connect to your AWS account. The `paco validate` command generates CloudFormation templates and validates them in your AWS account. Validate will never modify resources. It's a safe command to run to test the state of your Paco proejct.

```
$ paco validate netenv.ne.prod
```

6.3 Using PACO_HOME

With the exception of creating a new Paco project with `paco init project` all of the Paco commands operate on a Paco project. This is a directory of YAML files that conform to Paco project schemas.

These commands can all be run with a `--home` option to specify the path to this project. For example:

```
paco provision netenv.mynet.dev --home=/Users/username/projects/my-paco-project
```

However, it's tedious to need to type the full path to the Paco project for every command. You can change the current working directory to a Paco project and use `--home=.` but then you can't change directories.

The PACO_HOME environment variable can also be used to specify the Paco project home directory. You can export this environment variable on a BASH shell with the command:

```
export PACO_HOME=/Users/username/projects/my-paco-project
```

If you will only be working on a single Paco project, you could export this environment variable in your `~/.bash_profile`. However, if you are using more than one Paco project, we recommend putting a file named `profile.sh` in your Paco project's root directory that looks like this:

```
export PACO_HOME=/Users/username/projects/my-paco-project
export PS1="(my-paco-project) \h \W$ "
```

Then you can simply change directory to your Paco project and source the profile.sh file:

```
$ cd ~/projects/my-paco-project
$ source profile.sh
(my-paco-project) hostname my-paco-project$
```

Exporting the PS1 environment variable will remind you which Paco project is currently active in the PACO_HOME environment variable.

If you keep your Paco project in a git repo (which we highly recommend) and this is shared with other users, they will have different paths to their PACO_HOME. In this case, you can create a new `profile.sh` file after each time you clone a Paco project repo and put `profile.sh` in your `.gitignore` file to keep yourself from committing it to the repo.

Finally, if you have a project installation tool that is used to ensure that you are using the same fixed version of Paco and it's dependencies, it may also be able to use it dynamically create a `profile.sh` for your convenience.

6.4 Multi-account Setup

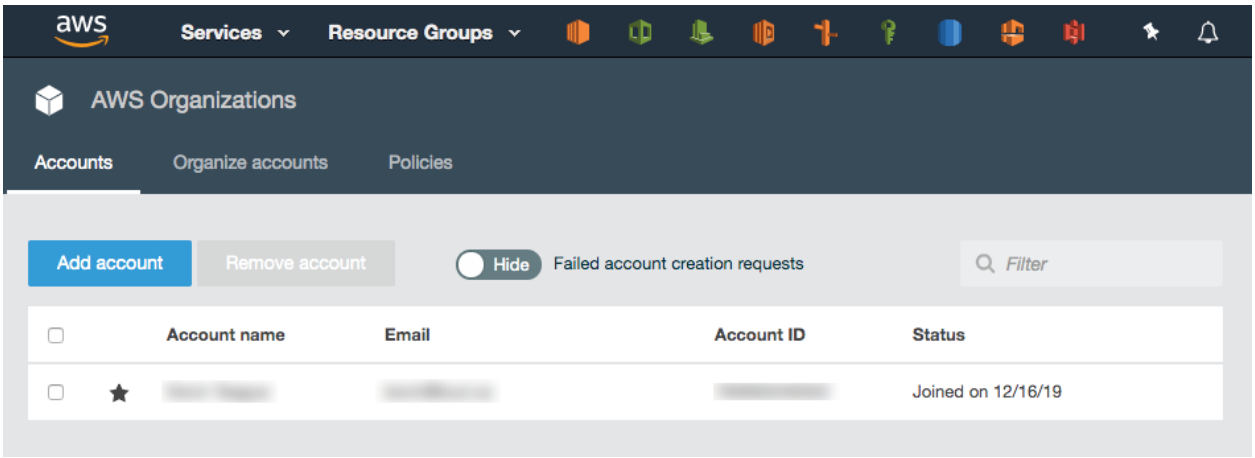
Paco can be run against a single AWS account or can be configured to control multiple accounts. Having a multi-account configuration is recommended for more serious cloud uses, for some of the benefits see Why use a multi-account architecture on AWS?.

6.4.1 Enable AWS Organizations

AWS has a service called AWS Organizations which lets you centrally govern your accounts. You will have a master account where you create an organization and from there child accounts can be created.

To get started, login to the AWS Console with your AWS account as a user with Administrator access. Assuming you only have one account - this will become your master account. This account will control the child accounts and the billing for all the child accounts will be charged to the master account.

Go to the AWS Organizations page and click "Create Organization" to convert your AWS account into a master account.



You will get an email from AWS to your root account email asking you to confirm your email address. Click on the link in the email and you should see a message saying **Your email address has been verified. You can now invite existing AWS accounts to join your organization.**

Do not create any of the child accounts from the console - we will let Paco create the accounts and it will store the configuration for those accounts in your Paco project.

6.4.2 Plan your accounts and AWS account limits

Take a minute to think about which AWS accounts you will want. A few common account suggestions and uses - although these are only suggestions, you are encouraged to think about your own requirements and determine your own multi-account structure:

- master: This is the only required account, and Paco always refers to this account by the name 'master'.
- prod: Having production resources in their own dedicated account is the most common reason to be multi-account.

- staging: Resources for a pre-prod, test or staging environment.
- dev: Resources for development environments.
- test: If you anticipate a lot of non-production environments, or simply have environments which can be easily created and destroyed, you may want to put your development and staging environments in one account.
- security: You may forward logging (especially CloudTrail) from all your accounts into a security account.
- sharedservices: Expensive global resources that support all environments, such as ActiveDirectory, may go in a dedicated shared services account.
- tools: For CI/CD and CodeCommit resources it can be useful to have a dedicated account that will have limited deploy capabilities to your dev/staging and prod accounts. Cross-account tooling

In your Paco project, the file `accounts/master.yaml` has a field named `organization_account_ids` which is for a list of all the child accounts. Open this file and create a list of the child accounts you want to create initially. If you later decide you want more accounts, you can simply add them here and follow the "Create child accounts with Paco" section again.

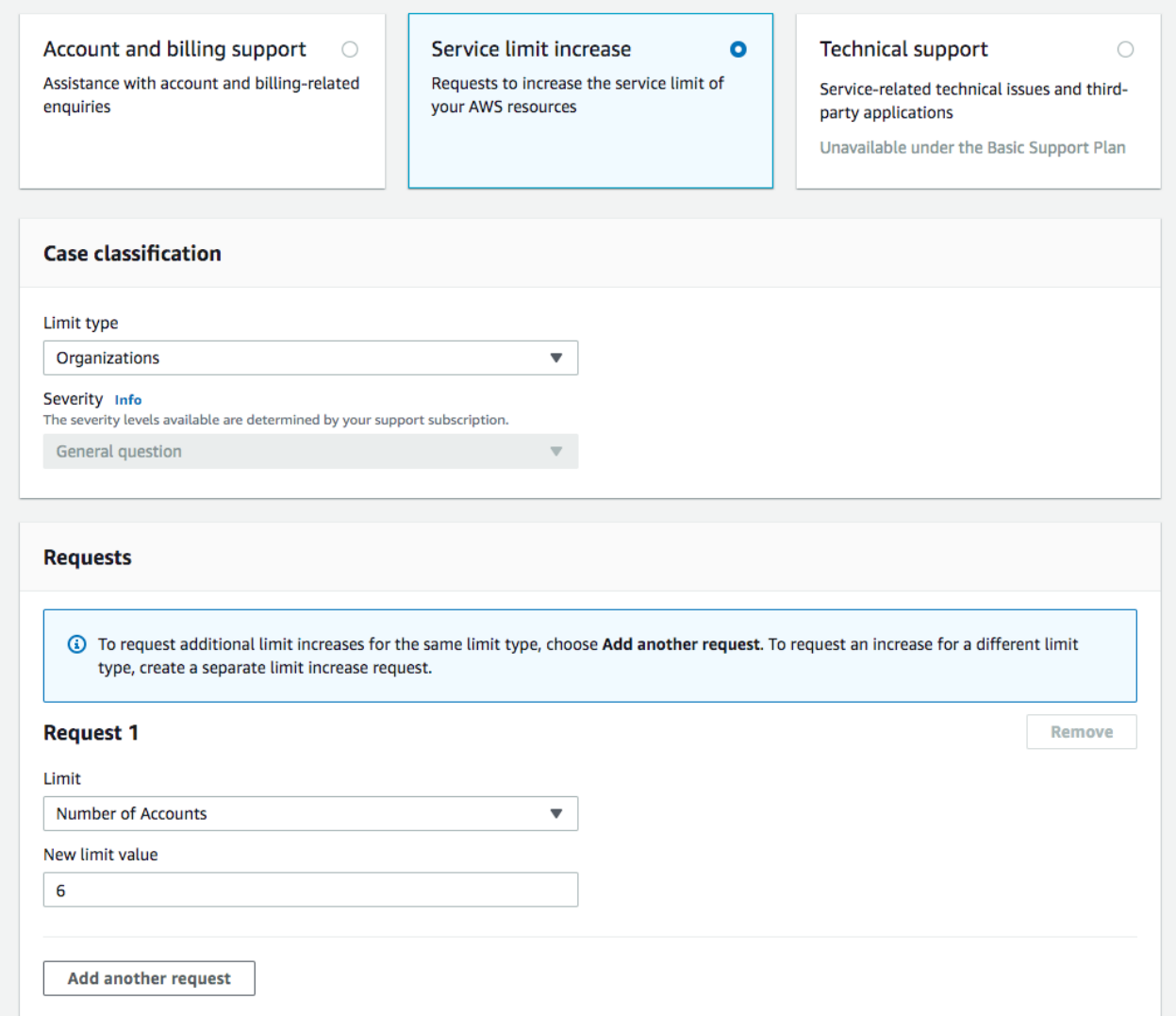
```
file: accounts/master.yaml

organization_account_ids:
- prod
- staging
- dev
- security
- tools
```

Next in the planning is to think about email addresses. AWS requires that every child account has a unique email address associated with it. These email addresses can be used to login as root to the child account, so ensure that whoever has access to the email addresses is trusted with access to that account.

Note: If you are setting up a multi-account for a personal or smaller organization and are using gmail, you can insert . characters in the left-hand portion of an email to create email aliases. For example, `examples@gmail.com` and `e.x.ample@gmail.com` will both go to the same inbox.

Out of the gate, AWS limits you to **only four accounts** - including the master account. If you plan on using more accounts than four accounts, you will need to contact AWS Support. In the upper-right corner of the console, choose Support and then Support Center. On the Support Center page, choose Create case.



After you submit your account limit increase request it can take a few hours or a full day or more before your account is reviewed and your limit is increased.

6.4.3 Create child accounts with Paco

To create child accounts is a two-step process. First you will run `paco init accounts` which will create a YAML file for every child account listed in `organization_account_ids` in the `accounts/master.yaml` file. The command is safe to run, if you've already created some child accounts and those child account YAML files already exist they will simply be skipped over.

```
$ cd ~/my-paco-project
$ paco init accounts

Loading Paco project: /Users/home/my-paco-project

AWS Account Initialization
--------------------------

AWS Organization account names have already been defined: prod,devstage,tools
```

(continues on next page)

(continued from previous page)

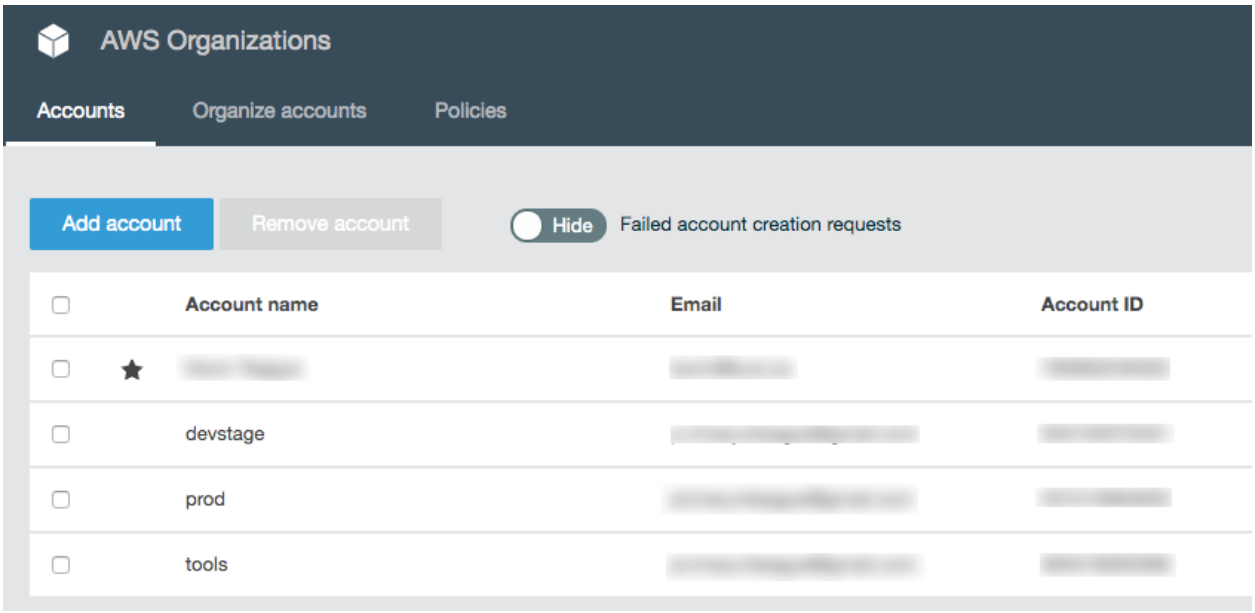
```
Initializing Account Configuration: prod

Title: [Production AWS Account]:
Region: [us-west-2]:
Root email address: you@example.com
```

Next you will actually create the child accounts. You will simply run `paco provision` with a scope of `accounts` to do this:

```
$ cd ~/my-paco-project
$ paco provision accounts
```

When this finishes you should be able to go to the AWS Organizations service in the console for your master account and see your child accounts:



6.4.4 Importing existing AWS accounts

If you have existing AWS accounts, you can manually invite them to join your master account by using the *Invitations* tab in the AWS Organizations service in the console for the master account. The existing accounts will need confirmation from the root email account associated with them, and then will join the master account.

Next you simply need to create an file in your Paco project's `accounts` directory where the filename is the name of account.

```
file: accounts/legacy.yaml

account_type: AWS
admin_delegate_role_name: Paco-Admin-Delegate-Role
region: us-west-2
title: Legacy AWS Account
root_email: you@example.com
account_id: '012345678912'
```

After you do this, run `paco provision accounts` in your Paco directory to update the IAM Role to allow it to delegate access into your newly imported account.

6.5 Managing IAM Users with Paco

The *Getting Started with Paco* showed you how to create an IAM User and Role that was able to allow a Paco project access to your AWS account. However, what if you have several people working in your AWS accounts and you want each one to have thier own dedicated account?

Paco can create IAM Users for you. It will also help you to configure permissions allowing a user cross-account access if you have a multi-account set-up. Each multi-account user can be granted access to all accounts, or restricted just to certain accounts. In addition, each user can have full admin access or have limited access.

For example, you could allow one user access to update a dev account but restrict them from accessing a production account. Or you could allow other users only access to CodeCommit and CodePipeline to only do application deployments.

6.5.1 IAM Users with resource/iam.yaml

A Paco project can have a `resource/iam.yaml` file that defines IAM Users.

```
users:
  yourusername:
    enabled: true
    account: paco.ref accounts.master
    username: yourusername
    description: 'Your Name - Paco Administrator'
    console_access_enabled: true
    programmatic_access:
      enabled: true
      access_key_1_version: 1
      access_key_2_version: 0
    account_whitelist: all
    permissions:
      administrator:
        type: Administrator
        accounts: all
```

Each user can be given access to all accounts or just certain ones. Use the `account_whitelist` with a comma-seperated list for this:

```
account_whitelist: dev,staging,tools # limit to only the dev, staging and tools␣
↪accounts

account_whitelist: all # special keyword for all accounts
```

Each user can be given full administrator access or limited to custom policies that only allow specific access. Use the `permissions` field for this:

```
permissions:
  # grants full access to all accounts that are defined in the account_whitelist field
  administrator:
    type: Administrator
    accounts: all
```

(continues on next page)

(continued from previous page)

```
  # grants custom access to only a test account
  custom:
    type: CustomPolicy
    accounts: test
    policies:
      - name: CloudWatchLogs
        statement:
          - effect: Allow
            action:
              - logs:Describe*
              - logs:Get*
              - logs:List*
            resource:
              - '*'
```

After you have added user(s) to `resource/iam.yaml` run:

```
paco provision resource.iam.users
```

This will generate a starting password for each user as well as an API key if `programmatic_access` was enabled for them.

6.5.2 Setting up a new User

A new user will first need to sign-in to the AWS Console with the AWS account id (with the master account id in a multi-account set-up), their username and starting password.

After signing in, they will be prompted to set a new password. After they are signed in, the only permission they will have is to set an MFA device for their User account. They will need to go to the IAM service, click on Users, then click on their User account. Then under the **Security Credentials** tab they need to click on the link **Manage** beside "Assign MFA Device". For more information, see AWS docs on Enabling MFA Devices.

6.5.3 Assuming a Role

Paco will only grants a User the ability to view and set their password and MFA device and the ability to **assume a role**. All permissions that a User will typically use must be gained by first assuming a Role that contains those permissions. This is done for security, as when a Role is assumed, it can enfore that the user has logged in with MFA.

Note that the first time a User logs in and sets MFA, they must then log out and log in again with their new MFA credentials. Only then will they be able to assume a Role.

In the AWS Console, assuming a Role is called switching roles, see the AWS docs on Switching to a Role. Each Role created by Paco will have a roleName in the format `IAM-User-Account-Delegate-Role-<username>`.

A user signed in to the console can switch roles by visiting a link in the format:

```
https://signin.aws.amazon.com/switchrole?account=123456789012&roleName=IAM-User-
↪Account-Delegate-Role-<username>
```

If you visit the CloudFormation service you can also see this in the `Resource-IAM-*` stacks on the Outputs tab with the Key `SigninUrl`.

6.5.4 AWS Extend Switch Roles

In a multi-account set-up, the AWS Console will only remember the five most recently used Roles. If you access more than five Roles, you will need to either manage Bookmarks with the SigninUrl for every Role or consider using the **AWS Extend Switch Roles** browser extension for Chrome or Firefox.

After you've installed this extension, you will see a green key in the top right of your browser. Click on **Configuration** and enter your configuration. You can use the example configuration below and replace `<username>` with your own username and refer to your Paco project `accounts` directory for the account id for your child accounts. Suggested colors are also provided ;P

```
[profile AwsOrgName Master]
aws_account_id = 123456789012
role_name = IAM-User-Account-Delegate-Role-<username>
color = 000000

[profile AwsOrgName Prod]
aws_account_id = 123456789012
role_name = IAM-User-Account-Delegate-Role-<username>
color = 800000

[profile AwsOrgName Stage]
aws_account_id = 123456789012
role_name = IAM-User-Account-Delegate-Role-<username>
color = 4f901a

[profile AwsOrgName Dev]
aws_account_id = 123456789012
role_name = IAM-User-Account-Delegate-Role-<username>
color = 008080

[profile AwsOrgName Tools]
aws_account_id = 123456789012
role_name = IAM-User-Account-Delegate-Role-<username>
color = 8000ff

[profile AwsOrgName Security]
aws_account_id = 123456789012
role_name = IAM-User-Account-Delegate-Role-<username>
color = e26453
```

6.6 The Paco CLI

The `paco` CLI is used to create, update and delete your cloud resources.

The CLI is divided into sub-commands. Run `paco --help` to see them:

```
$ paco --help
Usage: paco [OPTIONS] COMMAND [ARGS]...

Paco: Prescribed Automation for Cloud Orchestration

Options:
--version  Show the version and exit.
--help     Show this message and exit.
```

(continues on next page)

(continued from previous page)

```
Commands:
delete     Delete cloud resources
init       Create a new Paco project.
provision  Create and configure cloud resources.
validate   Validate cloud resources.
```

For most commands, you will need to tell `paco` where your Paco config directory is located. You can do this with the `--home` argument, or you can set an `PACO_HOME` environment variable.

6.6.1 Init

The `paco init` is divided into sub-commands:

- `paco init project <project-name>`: Creates a new directory with a boilerplate Paco project in it.
- `paco init credentials`: Initializes the .credentials file for a Paco project.
- `paco init accounts`: Initializes the accounts for a Paco project.

6.6.2 Cloud commands

There are three cloud commands to interact with the cloud:

- `paco validate <CONFIG_SCOPE>`: Generates CloudFormation and validates it with AWS.
- `paco provision <CONFIG_SCOPE>`: Creates or updates cloud resources.
- `paco delete <CONFIG_SCOPE>`: Deletes cloud resources.

The CONFIG_SCOPE argument is a reference to an object in the Paco project configuration.

6.6.3 Paco CLI config file

A `.pacoconfig` file can be used to set default Paco CLI options.

This file supports the `--warn` and `--verbose` options:

```
warn: true
verbose: true
```

6.6.4 Config Scope

A CONFIG_SCOPE is a valid Paco reference to a Paco object. Paco references start at the top of the Paco project tree and walk their way down. Consider the following Paco project:

```
paco-project/
  accounts/
    master.yaml
    prod.yaml
    dev.yaml
  monitor/
    Logging.yaml
    AlarmSets.yaml
```

(continues on next page)

(continued from previous page)

```
  project.yaml
  netenv/
    saas.yaml
    intra.yaml
  resource/
    CloudTrail.yaml
    CodeCommit.yaml
    EC2.yaml
    IAM.yaml
    SNSTopics.yaml
    Route53.yaml
    S3.yaml
  service/
    CustomAddOn.yaml
```

The top-level directory named `paco-project` is the start of the scope and project configuration is contained in the `project.yaml` file. You can not express this root scope with CONFIG_SCOPE.

CONFIG_SCOPE will start by expressing a directory in the Paco project. This can be one of four directories: accounts, netenv, resource or service. We will look at how scope works in each directory.

Note: Case insensitive filenames

The YAML filenames are case-insensitive. The scope `resource.cloudTRAIL` will match the filename `resource/CloudTrail.yaml`.

accounts scope

The accounts directory only has a single scope: `accounts`.

This will apply account actions on all accounts listed in the `organization_account_ids:` field in the `accounts/master.yaml` file. Typically you will create new accounts by giving them names in the `organization_account_ids:` list and then running `paco provision accounts` to create them.

There are no validate or delete commands for the accounts scope. If you need to delete an account, you should follow the AWS steps to close an account and then delete the appropriate `accounts/<account-name>.yaml` file.

netenv scope

The netenv scope is used to select environments, regions and applications for a single NetworkEnvironment.

At a minimum, you must specify a NetworkEnvironment and Environment with this scope:

```
netenv.saas.dev
```

The NetworkEnvironment is the name of a YAML file in the netenv directory, e.g. `netenv/saas.yaml`.

The Environment is the name of an environment in the `environment:` section of a netenv file. For example, consider this netenv file:

```
network:
  title: "My SaaS network"
  enabled: true
  availability_zones: 2
```

(continues on next page)

(continued from previous page)

```
  ...

applications:
  saas:
    title: "My SaaS application"
    enabled: false
    ...

environments:
  dev:
    title: "Development Environment"
    us-west-2:
      applications:
        saas:
          enabled: true
      network:
        aws_account: paco.ref accounts.dev
  prod:
    title: "Production Environment"
    default:
      applications:
        saas:
          enabled: true
      network:
        aws_account: paco.ref accounts.prod
    us-west-2:
      enabled: true
    eu-central-1:
      enabeld: true
```

The scopes available for this NetworkEnvironment are:

```
netenv.saas.dev
netenv.saas.dev.us-west-2
netenv.saas.prod
netenv.saas.prod.us-west-2
netenv.saas.prod.eu-central-1
```

After the NetworkEnvironment and Environment, the next component in the scope is the Region. If you do not specify a Region and you can have configured your Environments to belong to more than one region, Paco will apply the scope to all regions in that Environment.

You can drill down deeper than a Region. You may just want to update a single Application, which you can select with the `applications` name and the name of the application:

```
netenv.saas.prod.us-west-2.applications.saas
```

Within an Application you can scope even deeper and select only a ResourceGroup or a single Resource:

```
netenv.saas.prod.us-west-2.applications.saas.groups.cicd
netenv.saas.prod.us-west-2.applications.saas.groups.web.resources.server
```

Going this deep in the netenv scope is possible, but if you are trying to update some resources but not others, consider using the `change_protected: true` configuration. This field can be applied to any Resource and if set then Paco will never attempt to make any modifications to it:

```
saas:
  title: "My Saas App"
  enabled: false
  groups:
    web:
      type: Application
      enabled: true
      order: 10
      resources:
        servers:
          type: ASG
          # Tell Paco to never touch this resource
          change_protected: true
```

resource scope

The resource scope is used to select global resources.

You must specify a minimum of a global Resource type and you must have a YAML file for that type:

```
resource.codecommit
resource.ec2
```

These would scope to `resource/codecommit.yaml` and `resource/ec2.yaml` respectively. For most use cases, you will want to apply changes to all configuration in a global resource and you can not specify deeper scopes.

A few resources allow for deeper scoping - however, unless you have a very large Resource file, it's encouraged to simply scope the entire file:

CloudTrail resources in `resource/cloudtrail.yaml`:

```
resource.cloudtrail # applies to all CloudTrails
resource.cloudtrail.trails # also applies to all CloudTrails
resource.cloudtrail.trails.<trail-name> # select a single CloudTrail
```

EC2 resources in `resource/ec2.yaml`:

```
resource.ec2 # applies to all EC2 Keypairs
resource.ec2.keypairs # also applies to all EC2 Keypairs
resource.ec2.keypairs.<my-keypair> # select a single Keypair
```

IAM resources in `resource/iam.yaml`:

```
resource.iam # applies to all IAM Users
resource.iam.users # also applies to all IAM Users
resource.iam.users.<my-user> # select a single IAM User
```

service scope

The service scope is used to select Paco extension resources.

You must specify a minimum of a global Resource type and you must have a YAML file for that type:

```
service.patch
service.security
```

Typically you will only scope a complete add-on, but it is possible for an add-on to implement deeper scopes. Consult the add-on documentation directly.

6.7 Paco Workflows

Workflows describe the processes around how Paco is used to change cloud resources.

6.7.1 Enforce Git Branches for Environments

If you want to make changes to Paco configuration that is not yet ready to be applied to other environments then it is recommended to use a Git branch for each environment.

For example, if you have a `test` environment and a `prod` environment, you can override changes between the test and prod environments directly in the NetworkEnvironment file and provision both environments from the same Git branch. But what happens if you are making bigger changes between environments? What if you want to be less rigorous about changes to your `test` environment, but don't want a mistake to inadvertently be carried into your `prod` environment?

You can create a Git branch for each environment, then apply changes to one environment and test them before merging from one environment branch to the next and applying them there. The Paco default for naming your branches is:

```
ENV-<environment-name>
```

For example, with `dev`, `test` and `prod` environments you would create these Git branches:

```
master
ENV-dev
ENV-test
ENV-prod
```

Then you would only run `paco provision netenv.mynet.test` from within the `ENV-test` branch, after tests pass you would merge those changes into the `ENV-prod` branch and then from that branch run `paco provision netenv.mynet.prod`.

For provisioning global resources you can additionally designate those changes can happen from a designated branch. The suggested default for global resources is `prod`.

The Paco project's `project.yaml` configuration lets you enforce this Git branch workflow, and will prevent you from accidentally applying changes in an `ENV-test` branch to a prod environment.

The configuration to enable this beviour is in a Paco project's `project.yaml` file and is:

```
version_control:
  enforce_branch_environments: true
```

You can supply additional configuration if you don't want to use Paco's default conventions:

```
version_control:
  enforce_branch_environments: true
  environment_branch_prefix: "AWS_ENV_"
  git_branch_environment_mappings:
    - production:master
  global_environment_name: production
```

That additional configuration options allows you to configure different Git branch prefix names, map to branch names that don't have a prefix or follow a convention, and change the environment that can provision global resources.

6.8 WordPress Single-Tier

The **WordPress Single-Tier** starter project creates an budget-conscious single server WordPress site.

It is primarily intended to show you the basics of using Paco and for hands-on experience with Paco. It can be provisioned in a single AWS account with a low cost per hour.

The documentation for this starter project exists as a Waterbear Cloud blog post, "Turnkey AWS with Paco - Create and Manage a WordPress server".

6.9 Managed WebApp with CI/CD

The **Managed WebApp with CI/CD** starter project will provision a standard web application: ALB load balancer, AutoScalingGroup of web server(s) and an RDS MySQL database. This application has dev, staging and prod environments with a multi-account set-up. A CodePipeline deployment pipeline will build and deploy code to different environments based on your application's git repo branch names. This is a managed application, with a CloudWatch agent to gather host-level metrics and central logs, a suite of CloudWatch Alarms to alert on metrics and a CloudWatch Dashbaord to assess overall performance.

6.9.1 Create a "Managed WebApp with CI/CD" Project

Install Paco and then get started by running the `paco init project <your-project-name` command. Review the instructions on Getting Started with Paco to understand the importance of `name` fields in Paco and the difference between a name and title. Then follow the instructions on creating credentials for your project to connect it to your AWS Account.

Take a minute to set-up a PACO_HOME environment variable, this will save you lots of time typing.

This is a multi-account project template. The CI/CD will use cross-account permissions that are designed to be used in an account that is seperate from the accounts that they deploy into, so you will need at a minimum of two accounts. Review the Multi-account Setup instructions to understand how a multi-account set-up works with Paco.

After you've created a Paco project, connected it to your AWS master account and created child accounts, take a look at the Managing IAM Users with Paco docs. This template will start with only a single IAM User with Administrator access to all accounts. If you need to grant access to this Paco project to more than one person, or need to manage fine-grained access to different abilities across multiple accounts, then following this document is a must.

At this point you will have ran:

```
paco provision accounts
paco provision resource.iam.users
```

Finally return here to follow the instructions on customizing and provisioning the project!

6.9.2 Customize and Provision CloudTrail

This is an optional resource. CloudTrail is an AWS service which logs all changes to an AWS account. It is critical for securely managing accounts and can be extremely helpful in debugging why something broke when you have more than one person managing an account.

The CloudTrail file for this project is at `resource/cloudtrail.yaml`. It is configured to send CloudTrail for every account into an S3 Bucket in the tools account. If you're creating a more security conscious set-up, you will want to create a dedicated security account and change the `s3_bucket_account` field to direct CloudTrail there.

```
s3_bucket_account: paco.ref accounts.security
```

Also the CloudTrail will also be temporarily stored in a CloudWatch LogGroup for 14 days. You may want to disable that or make it longer. CloudWatch LogGroups are an easier way to search through your CloudTrail logs and you can also configure MetricFiters to alert you when events happen that can effect your AWS account security profile.

```
paco provision resource.cloudtrail
```

6.9.3 Customize and Provision SNS Topics

You will need to create SNS Topics if you plan on enabling and provisioning monitoring. These SNS Topics contain SNS Subscriptions. Review the `resource/snstopics.yaml` file and note that there is an **admin** group with one email subscription.

This group is configured to recieve any alarm notifications. You can add as many subscriptions to this group as you want. See the SNS Topics docs for examples of all protocols.

Also note that if you deployed in a region other than us-east-1 that your project will be configured to create a second SNS Topic in that region. This is because the Route 53 Health Check Service only works in that region. If you are not enabling HTTP health checks for your application, you can remove this region from your snstopics.yaml file.

6.9.4 Customize and Provision EC2 Key Pairs

You will need to create EC2 Key pairs in your dev, staging and prod accounts before you can launch EC2 instances in them. You can do this by running:

```
paco provision resource.ec2.keypairs
```

Make sure to save these keypairs somewhere safe, you will only see them once and need them for SSH access to your servers. If you prefer to use your own key pairs, you can create them in the AWS Console and simply edit the `resource/ec2.yaml` file and change the `keypair_name` field to match the name you gave your own keypair in AWS.

6.9.5 Customize and Provision Route 53

You were asked to supply a domain name when creating this project. This domain name is in the `resource/route53.yaml` file.

If you register a domain with the Route 53 service, it will create a Hosted Zone for you. When you provision the Route 53 file, it will create a new Hosted Zone:

```
paco provision resource.route53
```

After this runs, you will need to manually update the new Hosted Zone with the SOA (Start of Authority) and NS (nameservers) that are registered with your domain by AWS. Then you can remove the original Hosted Zone.

When you provision environments, the load balancers will add A Records to your HostedZone to automically enable your domain to be directed to the laod balancer.

You can also use a domain with another registrar. You will need to manually manage the A Records yourself in this case.

6.9.6 Customize and Provision CodeCommit

The CodeCommit docs describes your git repos and users in the `resource/codecommit.yaml` file.

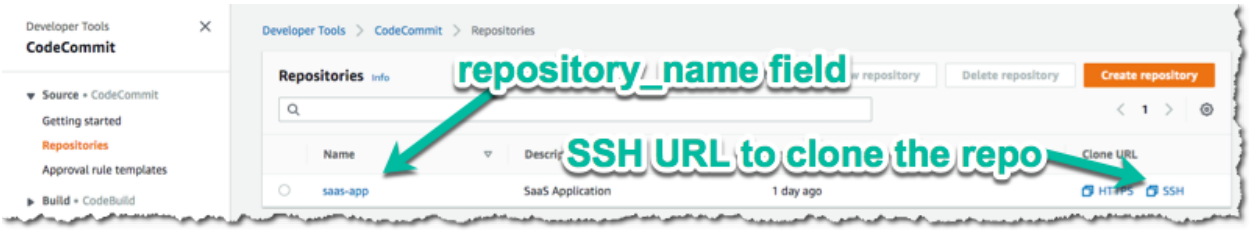
This file will start with a single git repo and a single user. Each user will be a new IAM User that only has permissions for that repo. It is possible to grant a normal Paco IAM User access to CodeCommit repo's but we recommend creating dedicated users through `resource/codecommit.yaml` as this limts the blast radius if these credentials are leaked.

If you've got more than one developer, add them to the `users:` section and then create the repo and users with:

```
$ paco provision resource.codecommit
Loading Paco project: /Users/username/projects/my-paco-project
 ...
Provision  tools       Create          Resource-CodeCommit-Git-Repositories
Run        tools       Hook            Resource-CodeCommit-Git-Repositories: :␣
↪CodeCommitSSHPublicKey: post: create
tools:   Upload:  SSHPublicKeyId: you@example.com: APKA2......FPV2EAI
```

Be sure to save the AWS SSH key ID for each user. You can also see these keys in IAM in the AWS Console if you lose them.

Next, you will need to use the AWS Console to switch to the tools account that the CodeCommit repo was provisioned in and go to the CodeCommit service. You should see something like:



Copy the SSH Url and clone the repo with *git clone <ssh-url>*.

To authenticate when cloning the repo, each user can either add the AWS SSH key Id to their *~/.ssh/config* file:

```
Host git-codecommit.*.amazonaws.com
  User APKAV........63ICK
  IdentityFile ~/.ssh/my_pubilc_key_rsa
```

Or if they are using their default public key, they can embed the AWS SSH key ID as the user in SSH Url:

```
git clone ssh://APKAV........63ICK@server/project.git
```

6.9.7 Create a Web Application with CodeBuild and CodeDeploy YAML files

This starting template is set-up to deploy a simple Python Pyramid web application although we will show you how to replace this with your own application.

Your application will need two files at in the top level directory:

- buildspec.yaml defines how the application is built using CodeBuild
- appspec.yaml defines how your application is deployed using CodeDeploy

If you want to see the example web application in action, after you provision an environment, you will need to follow the README.txt for that application to create a MySQL database named `saas` and run the database initialization scripts in `/var/www/saas-app/` to pre-populate the database.

6.9.8 Customize and Provision Environments

This project starts with three environments: dev, staging and prod. Each of these environments will be provisioned in a single region.

In the examples below, we will assume you named your NetworkEnvironment `mynet` and you chose `us-west-2` for your region.

You can provision an environment with:

```
paco provision netenv.mynet.dev
```

6.9.9 Customizing environments

Your `netenv/mynet.yaml` contains all the configuration for your environment, it's network, applications and other shared resources such as backups and secrets. Each top-level seciton will define the default configuration. This is configuration only and is not used to create actual cloud resources.

The `environments:` section will then name these default resources in specific environments and regions. This section controls what you want to actually provision in the cloud.

An environment has a `default:` section. This area allows you to override any base configuration.

Let's see what starting overrides have been applied to the dev environment:

```
dev:
  title: "Development Environment"
  us-west-2:
    enabled: true
  default:
    secrets_manager:
      ap:
        site:
          database:
            enabled: true
    applications:
      app:
        enabled: false
        groups:
          bastion:
            resources:
              instance:
                instance_key_pair: paco.ref resource.ec2.keypairs.app_dev
          app_deploy:
            resources:
              pipeline:
                source:
                  codecommit:
                    deployment_branch_name: "master"
                build:
                  codebuild:
                    deployment_environment: "master"
          site:
            resources:
              alb:
                dns:
                  - domain_name: dev.example.com
```

(continues on next page)

(continued from previous page)

```
            listeners:
              https:
                rules:
                  app_forward:
                    host: 'dev.example.com'
                  app_redirect:
                    enabled: false
          web:
            instance_key_pair: paco.ref resource.ec2.keypairs.app_dev
            monitoring:
              enabled: false
          database:
            multi_az: false
```

First, you will have different `instance_key_pair` values for your EC2 instances. If you wanted to share keypairs between your dev and staging environments, you could copy the values from your staging environment into your dev environment.

Next, you have an Application Load Balancer (ALB) which is configured to redirect `*.yourdomain.com` to `yourdomain.com` in your default prod configuration. In the dev environment this redirect is disabled and the listener to forward to the TargetGroup that has your web servers has the host `dev.yourdomain.com`.

This exposes your dev environment at `dev.yourdomain.com`. You may not want to do this, however. Instead you might want to rely on using the more obfuscated ALB DNS name directly. To change this, remove the `dns:` and `host:` overrides:

```
dev:
  default:
    applications:
      app:
        groups:
          site:
            resources:
              alb:
                # remove DNS entry
                # dns:
                #  - domain_name: dev.pacosaas.net
                listeners:
                  https:
                    rules:
                      # remove this section setting the host
                      #app_forward:
                      #  host: 'dev.pacosaas.net'
                      app_redirect:
                        enabled: false
```

Beyond the scope of this starting template, but to make your non-prod envs completely private, you could also run a VPN service on the bastion instance and run the load balancer in the private subnets.

Finally you may want to customize your CI/CD. The starting template uses AWS CodePipeline together with CodeCommit, CodeBuild and CodeDeploy. Each environment will watch a different branch of the git repo stored in the CodeCommit repo.

- prod env <– prod branch
- staging env <– staging branch
- dev env <– master branch

These branch names are arbitrary. You might want to designate master as production, or even not have master deploy to any environents. These can be customized to suit whatever branching system you want to use in your version control workflow.

6.9.10 SSH to a Web Server and connect to MySQL

In your `netenv/mynet.yaml` you will have Security Groups defined in your `network:` configuration. The SSH port for your bastion is open to all IP addresses. You may wish to restrict this to only specific IP addresses to improve your security.

You can change the `ingress:` to be only your IP address:

```
bastion:
  instance:
    enabled: true
    egress:
      - cidr_ip: 0.0.0.0/0
        name: ANY
        protocol: "-1"
    ingress:
      - from_port: 22
        name: SSH
        protocol: tcp
        cidr_ip: 128.255.255.128/32
        #cidr_ip: 0.0.0.0/0
        to_port: 22
```

Then update your security groups. If you have already provisioned all three environments, you will need to update them all:

```
paco provision netenv.mynet.dev
paco provision netenv.mynet.staging
paco provision netenv.mynet.prod
```

If you don't want to run your bastion host 24/7, you can disable it to save on your AWS costs. If you only want to disable it for certain environments, customize the `enabled:` field in the environment section:

```
environments:
  dev:
    default:
      applications:
        app:
          enabled: true
          groups:
            bastion:
              enabled: false # add this line below the bastion: line
```

And run `paco provision` after changing this.

Also notice that your bastion has an EIP resource and an `eip:` field. This will provision an Elastic IP and attach it to the bastion. If you start/stop the bastion, it will keep the same fixed IP address. If you don't want to use this feature, you can disable the EIP resource and remove the `eip:` field.

Once you are connected to your bastion, you can then connect to your web servers in your private subnets. You will need to go to the EC2 service in the AWS Console to see what the private IP address of a web server is. In order to avoid having to copy your SSH private key to the bastion server, you can use the SSH ProxyCommand to connect directly to the web server from your own computer. Edit your `~/.ssh/config` file and add:

```
Host myweb-dev
  Hostname 10.0.0.100 # <-- private IP of a web server
  User ec2-user
  IdentityFile ~/.ssh/myweb-dev-us-west-2.pem # <-- path to your private SSH key
  # replace the path to your private SSH key and your bastion public EIP (or dynamic␣
↪public IP)
  ProxyCommand ssh -i ~/.ssh/myweb-dev-us-west-2.pem ec2-user@128.255.255.128 -W %h:%p
```

Now you can simply run:

```
$ ssh myweb-dev
```

Note that the web servers are in an AutoScalingGroup. This means instances will be replaced if they become unhealthy, and new web servers will have different private IP addresses. You will need to change your Hostname IP after this happens.

Once you are on the web server, try connecting to your MySQL database. You will need the endpoint of the RDS database and the password from Secrets Manager. You can find these in the console, or if have the `get_rds_dsn.sh` script installed, you can run it too see this from the server:

```
$ ssh myweb-dev
$ sudo su
# /tmp/get_rds_dsn.sh
# mysql -h ne-wa-staging-app-ap-site-database-rds.c1aqauvngpug.us-west-2.rds.
↪amazonaws.com -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
...
MySQL [(none)]>
```

6.9.11 Customize your Web Server to support your web application

CloudFormation Init is is a method to configure an EC2 instance after it is launched. It's a much more complete and robust method to install configuration files and pakcages than with a UserData script.

If you look at your project's `netenv/mynet.yaml` file in the `applications:` section you will see a `web:` resource that defines your web server AutoScalingGroup. There is a `cfn_init:` field for defining your cfn-init configuration.

```
launch_options:
    cfn_init_config_sets:
    - "Install"
cfn_init:
  parameters:
    DatabasePasswordarn: paco.ref netenv.wa.secrets_manager.ap.site.database.arn
  config_sets:
    Install:
      - "Install"
  configurations:
    Install:
      packages:
        yum:
          jq: []
          httpd: []
          python3: []
          gcc: []
```

(continues on next page)

(continued from previous page)

```
          httpd-devel: []
          python3-devel: []
          ruby: []
          mariadb: []
      files:
        "/tmp/get_rds_dsn.sh":
          content_cfn_file: ./webapp/get_rds_dsn.sh
          mode: '000700'
          owner: root
          group: root
        "/etc/httpd/conf.d/saas_wsgi.conf":
          content_file: ./webapp/saas_wsgi.conf
          mode: '000600'
          owner: root
          group: root
        "/etc/httpd/conf.d/wsgi.conf":
          content: "LoadModule wsgi_module modules/mod_wsgi.so"
          mode: '000600'
          owner: root
          group: root
        "/tmp/install_codedeploy.sh":
          source: https://aws-codedeploy-us-west-2.s3.us-west-2.amazonaws.com/latest/
↪install
          mode: '000700'
          owner: root
          group: root

      commands:
        10_install_mod_wsgi:
          command: "/bin/pip3 install mod_wsgi > /var/log/cfn-init-mod_wsgi.log 2>&1"
        11_symlink_mod_wsgi:
          command: "/bin/ln -s /usr/local/lib64/python3.7/site-packages/mod_wsgi/
↪server/mod_wsgi-py37.cpython-37m-x86_64-linux-gnu.so /usr/lib64/httpd/modules/mod_
↪wsgi.so > /var/log/cfn-init-mod_wsgi_symlink.log 2>&1"
        20_install_codedeploy:
          command: "/tmp/install_codedeploy.sh auto > /var/log/cfn-init-codedeploy.
↪log 2>&1"

      services:
        sysvinit:
          httpd:
            enabled: true
            ensure_running: true
            commands:
              - 11_symlink_mod_wsgi
          codedeploy-agent:
            enabled: true
            ensure_running: true
```

There is a lot of configuration here. First, the `launch_options:` simply tells Paco to inject a script into your UserData that will ensure that cfn-init is installed and runs your cfn-init configuration.

Next, the `parameters:` section is the only section that doesn't map to cfn-init config. It's used to make configuration parameters available to be interpolated into cfn-init files. These can be static strings or references to values created by resources provisioned in AWS.

The `packages:` section is simply a list of rpm packages.

The `files:` section is a list of files.The content of this file can be defined either as a `content_cfn_file:` which will be interpolated with CloudFormation Sub and Join functions, or a static non-interpolated with the `content_file:` field, or simply in-lined with the `content:` field.

You can see that for the example Python Pyarmid application, there is custom WSGI configuration used with the Apache web server. There is also a script to install the CodeDeploy agent. You will need this CodeDeploy agent installed and running to work with the CI/CD regardless of what application you deploy.

The `get_rds_dsn.sh` file is an example of interpolating the ARN of the provisioned RDS MySQL database into a file on the filesystem. It also shows you the command to run to get the secret credentials to connect to your database. Note that there is an IAM Role created for this instance when it is connected to the secret by the `secrets:` field for the ASG that allows access to only the listed secrets.

The `commands:` section runs shell commands in alphanumeric order. You can customize the mod_wsgi commands, but again leave the command to install the CodeDeploy agent.

Finally the `services:` section is used to ensure that services are started and remain running on the server. Again, you might want to replace Apache (httpd) with another web server, but will want to leave CodeDeploy as-is.

6.9.12 Working with Regions

When you provision an environment, you can also specify the region:

```
paco provision netenv.mynet.dev.us-west-2
```

If you look at your `netenv/mynet.yaml` file you will see an `environments:` section at the bottom of the file:

```
environments:
  dev:
    title: "Development Environment"
    us-west-2:
      enabled: true
  default:
```

Let's say that you wanted to also have a development environment in eu-central-1 for your European developers. You can simply add a second region:

```
environments:
  dev:
    title: "Development Environment"
    us-west-2:
      enabled: true
    eu-central-1:
      enabled: true
  default:
```

The first time you make a new region available, you will want to add it to your `project.yaml` file:

```
name: my-paco-project
title: My Paco
active_regions:
  - eu-central-1
  - us-west-2
  - us-east-1
```

You will also need to provision any global support resources for that region, such as SNS Topics and EC2 Key pairs.

Then you can provision into that region:

```
paco provision netenv.mynet.dev.eu-central-1
```

Now when you run provision on the environment, it would apply changes to both regions:

```
paco provision netenv.mynet.dev # <-- applies to both us-west-2 and eu-central-1
```

6.9.13 Monitoring an environment

To start, monitoring is only enabled for the prod environment. You may wish to enable your monitoring for your other environments, but this will add a small amount to your AWS bill from CloudWatch. Monitoring is enabled/disabled with the `enabled:` field under the `monitoring:` configuration.

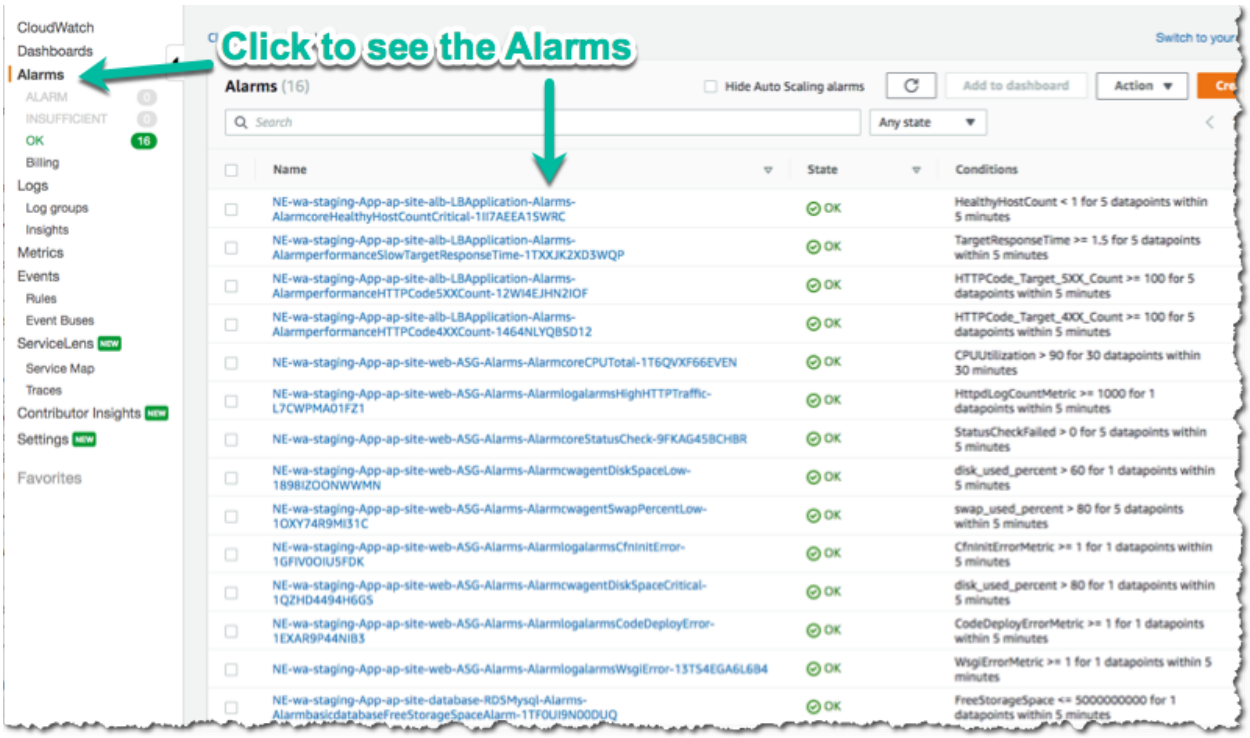
```
site:
  resources:
    alb:
      monitoring:
        enabled: true # changed from false
      dns:
        - domain_name: staging.example.com
      listeners:
        https:
          rules:
            app_forward:
              host: 'staging.example.com'
            app_redirect:
              enabled: false
    web:
      instance_key_pair: paco.ref resource.ec2.keypairs.app_staging
      monitoring:
        enabled: true # changed from false
    database:
      multi_az: false
      monitoring:
        enabled: true # changed from false
    dashboard:
      enabled: true # changed from false
```

With monitoring enabled you will have:

- CloudWatch Alarms for your Application Load Balancer, web server AutoScalingGroup and RDS MySQL.
- CloudWatch Agent which runs on your web servers to collect logs and in-host metrics.
- CloudWatch Log Groups to collect os, ci/cd and application logs.
- Cloudwatch Log Group metric filters to gather metrics on errors in logs.
- CloudWatch Alarms to alert you when your logs have errors.

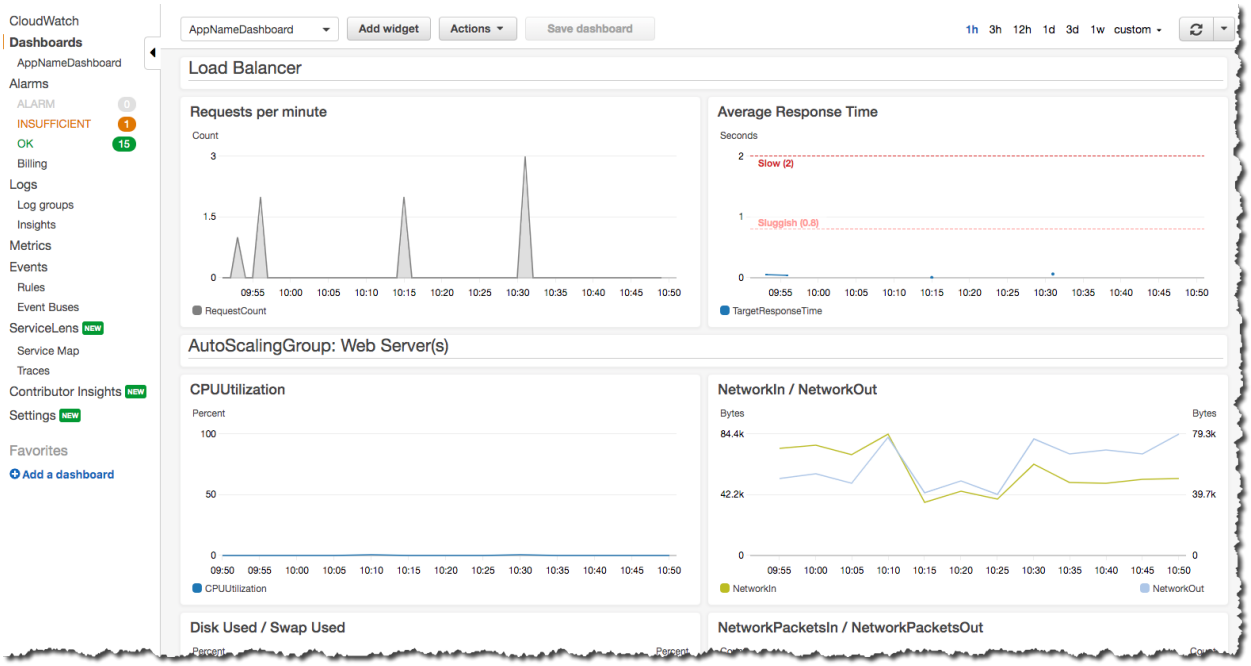
Note that when you enable/disable monitoring, this will change the CloudWatch agent installation configuration for your web servers. This will cause AWS to terminate your existing web servers and launch new instances.

From the AWS Console you can visit the CloudWatch service to see your Alarms:



The alarms are all contained in the `monitor/AlarmSets.yaml` file. You also may wish to remove certain alarms or add new ones - customizing alarms and the thresholds is very specific to the application you are running and it's traffic.

If you want to see how your application's resources are performing overall, take a look at the CloudWatch Dashboard that was provisioned when you enabled monitoring:



Here you can see graphs of some metrics for your Load Balancer, web server AutoScalingGroup and RDS MySQL database. Again, this is only a basic selection of some of the metrics available - it is common to customize this to be specific to your application.

You can change this Dashboard directly in the AWS Console, but these Dashboard settings are controlled by a configuration file at `netenv/dashboards/complete-dashboard.json` and can be restored to the original settings with subsequent paco provision commands. Instead, when viewing the Dashboard choose "Actions –> Save dashbaord as ..." and create a copy of this Dashboard, then make manual customizations.

It's also possible to choose "Actions –> View/edit source" and put the JSON configuration for a Dashboard into your project's configuration. Note that you will need to replace the hard-coded region and resource ids with placeholders to be dynamically interpolated when the Dashboard is created.

The configuration for your starting Dashboard looks like this:

```
dashboard:
  type: Dashboard
  enabled: true
  order: 200
  title: AppNameDashboard
  dashboard_file: ./dashboards/complete-dashboard.json
  variables:
    ApplicationTargetGroup: paco.ref netenv.mynet.applications.app.groups.site.
↪resources.alb.target_groups.app.fullname
    LoadBalancerName: paco.ref netenv.mynet.applications.app.groups.site.resources.
↪alb.fullname
    WebAsg: paco.ref netenv.mynet.applications.app.groups.site.resources.web.name
    DBInstance: paco.ref netenv.mynet.applications.app.groups.site.resources.database.
↪name
```

Feel free to change the `title:` field - but remember that CloudWatch Dashboards can only contain alphanumeric characters.

Finally, take a look at your `monitor/Logging.yaml` file. Here you will see the log files that are collected. You will most likely want to keep the `rpm_linux` and `cloud` logs as-is. Take a look at the Metric Filters for the `cloud` logs:

```
cloud:
  # cloud logs specific to configuration and operation in AWS
  log_groups:
    cfn_init:
      sources:
        cfn_init:
          path: /var/log/cfn-init.log
          log_stream_name: "{instance_id}"
      metric_filters:
        CfnInitErrors:
          filter_pattern: '"[ERROR]"'
          metric_transformations:
            - metric_name: 'CfnInitErrorMetric'
              metric_value: '1'
    codedeploy:
      sources:
        codedeploy:
          path: /var/log/aws/codedeploy-agent/codedeploy-agent.log
          log_stream_name: "{instance_id}"
      metric_filters:
        CodeDeployErrors:
          filter_pattern: '" ERROR "'
          metric_transformations:
            - metric_name: 'CodeDeployErrorMetric'
              metric_value: '1'
```

These Metric Filters apply a filter pattern to every log line ingested. If they match the pattern, they will send a metric value to CloudWatch. There are special LogAlarms in your AlarmSets.yaml file to watch for these metrics and notify on them:

```
# CloudWatch Log Alarms
log-alarms:
  CfnInitError:
    type: LogAlarm
    description: "CloudFormation Init Errors"
    classification: health
    severity: critical
    log_set_name: 'cloud'
    log_group_name: 'cfn_init'
    metric_name: "CfnInitErrorMetric"
    period: 300
    evaluation_periods: 1
    threshold: 1.0
    treat_missing_data: notBreaching
    comparison_operator: GreaterThanOrEqualToThreshold
    statistic: Sum
  CodeDeployError:
    type: LogAlarm
    description: "CodeDeploy Errors"
    classification: health
    severity: critical
    log_set_name: 'cloud'
    log_group_name: 'codedeploy'
    metric_name: "CodeDeployErrorMetric"
    period: 300
    evaluation_periods: 1
    threshold: 1.0
    treat_missing_data: notBreaching
    comparison_operator: GreaterThanOrEqualToThreshold
    statistic: Sum
```

These alarms will alert you if your instance has errors during the CloudFormation Init launch configuration, or if the CodeDeploy agent has errors during a new application deployment. These can be very helpful at letting you know your CI/CD set-up has gone off the rails.

There are similar alarms for the example Python Pyramid application. These are under the "# application specific logs" comment in Logging.yaml and in AlarmSets.yaml for the alarms named WsgiError and HighHTTPTraffic. You will want to customize these logs and alarms to whatever web server and application-specific logs you have in your web server set-up.

6.9.14 Backup and Restore

This project also has a BackupVault that will make daily database backups on the prod database.

You can already take advantage of RDS's built-in automatic backups to create snapshots. However, AWS also provides AWS Backup as a centralized location to do backups. The advantage of using this service to back-up your database is you can retain backups longer than 35 days, you can transition older back-ups to S3 Glacier and if you can have several backup schedules (e.g. daily, weekly and monthly) with different lifecycle policies.

The `netenv/mynet.yaml` has a `backup_vaults:` section that looks like:

```
backup_vaults:
  app:
```

(continues on next page)

(continued from previous page)

```
    enabled: false
    plans:
      database:
        title: RDS Backups
        enabled: true
        plan_rules:
          - title: Daily RDS backups
            schedule_expression: cron(0 7 ? * * *)
            lifecycle_delete_after_days: 30
        selections:
          - title: RDS Daily Backups Selection
            tags:
              - condition_type: STRINGEQUALS
                condition_key: Paco-Application-Name
                condition_value: {{cookiecutter.application_name}}
              - condition_type: STRINGEQUALS
                condition_key: Paco-Application-Group-Name
                condition_value: site
              - condition_type: STRINGEQUALS
                condition_key: Paco-Application-Resource-Name
                condition_value: database
              - condition_type: STRINGEQUALS
                condition_key: paco.env.name
                condition_value: prod
```

This will be overridden only in the prod environment to turn on `enabled: true`. The backup selection is configured to use tags to select resources to backup. This can be helpful if you want to have a whole group of things backed up without needing to remember to adjust your backup selections. For example, if you had multiple databases, you could put them all into the same Resource Group named `persistence` and select that group. If you added a new database, it would automatically be included in the backup selection.

Paco automatically applies a standard set of Tags to all resources it creates. Every Paco resource is located in a hierarchical tree in this order:

- NetworkEnvironment: A shared collection of environments
- Environment: A complete set of working resources, e.g. dev, staging and prod
- Application: An application within an environment, e.g. wordpress or saas-app
- Resource Group: A group of resources to support an environment. Helpful to seperate CI/CD resources from app resources, for example.
- Resource: A specific conceptual resource. Sometimes this can be more than one actual AWS Resource, such as a Lambda and a Lambda Permission.

Your prod RDS database will have these Tags:

```
paco.netenv.name: mynet
paco.env.name: prod
Paco-Application-Name: app
Paco-Application-Group-Name: site
Paco-Application-Resource-Name: database
```

Alternatively there is a `resource:` field for selections that can be used to specify a specific resource with a Paco reference. Using this field will ensure that you have correctly chosen a real resource - if there is a typo, Paco will warn you when it loads your Paco project configuration. Otherwise if you are using tag-based selections, you are recommended to review your BackupVault in the AWS Console and ensure it's working correctly.

```
selections:
  - title: RDS Daily Backups Selection
    resource: paco.ref netenv.mynet.applications.app.groups.site.resources.database.
↪name
```

6.10 S3 Bucket Lambda replicator

The **S3 Bucket Lambda replicator** starter project creates an S3 Bucket which will invoke a Lambda function with every object addition and deletion. The Lambda will copy or delete the object from the source S3 Bucket to replica S3 Buckets in other regions.

S3 Buckets already have a Cross-Region replication (CRR) feature and we recommned you use this feature for robust data replication. However, CRR only allows you to replicate to only a single other region. It is also not possible to daisy-chain from the target S3 Bucket to another region. This solution was originally developed for deploying Lambda artifacts to multiple regions.

It serves as an example of using Paco to manage S3 Bucket and Lambda objects. There is no network or other complex resources in this starting project.

6.10.1 Create a "S3 Bucket Lambda replicator" Project

Install Paco and then get started by running the `paco init project <your-project-name>` command. Review the instructions on Getting Started with Paco to understand the importance of `name` fields in Paco and the difference between a name and title. Then follow the instructions on creating credentials for your project to connect it to your AWS Account.

You will be asked to provide prompts for a NetworkEnvironment name and title. While this project does not provision any network resources, Paco still uses the name `netenv` to refer to a set of environments that contain the same set(s) of applications and shared resources.

Take a minute to set-up a PACO_HOME environment variable, this will save you lots of time typing.

6.10.2 Customize and Provision SNS Topics

You will need to create SNS Topics if you want to provision the prod environment, which has CloudWatch Alarms to notify you if the Lambda function throws errors or is taking too long to complete.

These SNS Topics contain SNS Subscriptions. Review the `resource/snstopics.yaml` file and note that there is an **admin** group with one email subscription.

This group is configured to recieve any alarm notifications. You can add as many subscriptions to this group as you want. See the SNS Topics docs for examples of all protocols.

6.10.3 Customize and Provision Environments

There are two environments with this project: dev and prod. They are almost the same except the prod environment has a pair of CloudWatch Alarms to notify you if your Lambda function has invocation problems.

Before you provision these environments, if you are using this netenv in a multi-account set-up, review the `aws_account` field and change this to the correct account name you want to use:

```
prod:
  title: "Production Environment"
  default:
    network:
      aws_account: paco.ref accounts.prod # deploy prod env to prod account
```

Now provision an environment with:

```
paco provision netenv.mynet.dev
paco provision netenv.mynet.prod
```

The prod environment is also intended to be used with more than one region to replicate into. You will see this at the very bottom of your project's `netenv/mynet.yaml` file:

```
us-west-2:
  enabled: true
  applications:
    app:
      groups:
        replica:
          enabled: true
```

You can add as many regions here as you need:

```
us-west-2:
  enabled: true
  applications:
    app:
      groups:
        replica:
          enabled: true
us-east-1:
  enabled: true
  applications:
    app:
      groups:
        replica:
          enabled: true
ca-central-1:
  enabled: true
  applications:
    app:
      groups:
        replica:
          enabled: true
```

This will create the S3 Buckets to hold the replicated objects. You will also need to tell the Lambda which buckets to replicate into using an environment variable named `REGIONS`:

```
prod:
  ca-central-1:
    applications:
      app:
        groups:
          original:
            enabled: true
            resources:
```

(continues on next page)

(continued from previous page)

```
            replicator:
              environment:
                variables:
                  - key: 'ENV'
                    value: 'prod'
                  - key: 'REGIONS'
                    value: 'usw2;use1;cac1'
```

You will need to use the short region name for each AWS region. See the `aws_regions` section in the paco.models vocabulary file to look-up the short names for regions. There will also be an S3 Bucket created in the same region as the original bucket, if you need to replicate into that region with an S3 Bucket name that is consistent with the other regions.

Finally, update your Paco `project.yaml` file to have a list of all of your `active_regions`. This is a master lists of regions you should be active in. It can be used in certain places in your configuration to list `all` as a special keyword to refer to all your Paco project's useable regions:

```
name: myproj
title: MyProj
active_regions:
- eu-central-1
- us-west-2
- us-east-1
- ca-central-1
```

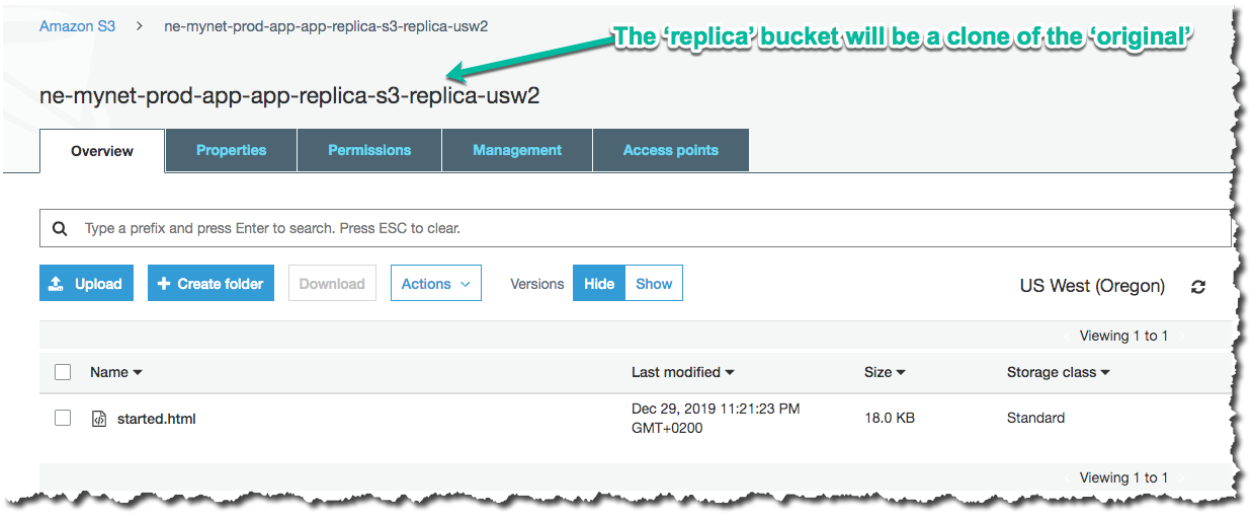
6.10.4 Test Your S3 Bucket Lambda

Log into the AWS Console and go to the S3 Bucket service. You will see buckets with names like this:

```
ne-mynet-prod-app-app-original-source-original-cac1
ne-mynet-prod-app-app-replica-s3-replica-usw2
```

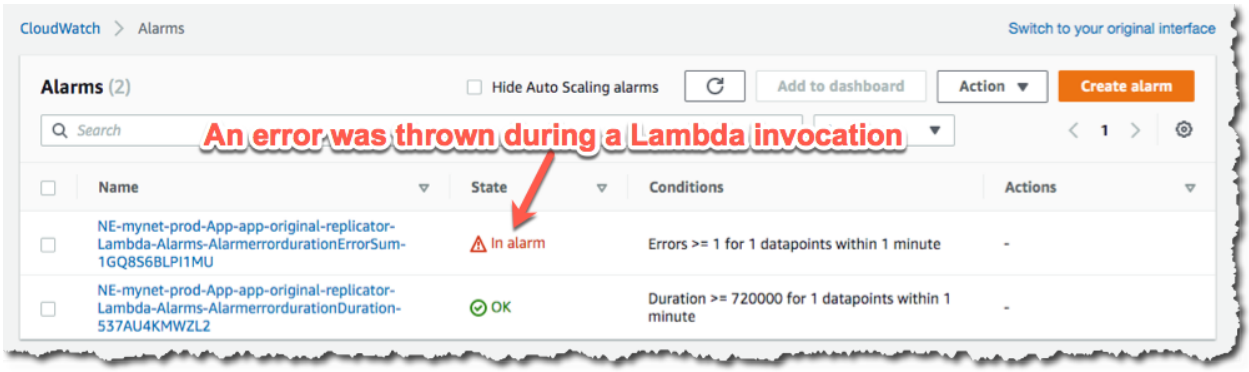
Go the "orginal" bucket and upload an object:

Then navigate to a "replica" bucket and you should see a copy of your object:



If you didn't and this is in the prod environment, a CloudWatch Alarm will fire after the Lambda invocation failed. This will happen if your environment variable names are incorrect. You can also go to your Lambda and generate a Test invocation with an empty event and this will cause the Lambda to safely throw an error.

In the CloudWatch service you will see your "Errors" Alarm in an alarm state:



There are two alarms, one for invocation errors and a second for duration. If the Lambda takes longer than 80% of the total allocated run time, this error will fire. With this simple Lambda it is unlikely that you will ever see this alarm be triggered, but such an alarm is generally useful for any Lambdas that you deploy. AWS will suddenly stop an Lambda which reaaches it's maximum duration, so it's good to be notified before this happens.

6.10.5 Apply an S3 Bucket Policy

If you were to use this for a real-world solution, you would also need to determine what kind of S3 Bucket Policy should protect your buckets. This project starts with a simple policy that allows only the root account access to s3:GetObject API calls on the replica buckets. Adjust this policy to suit your needs:

```
replica:
  type: Application
  title: "Replica S3 Bucket"
  order: 1
  enabled: false
  resources:
    s3:
      type: S3Bucket
```

(continues on next page)

(continued from previous page)

```
    enabled: true
    order: 1
    bucket_name: 'replica'
    deletion_policy: 'delete'
    policy:
      - aws:
          - 'arn:aws:iam::123456789012:root'
        effect: 'Allow'
        action:
          - 's3:GetObject'
        resource_suffix:
          - '/*'
          - ''
```

After updating the policy YAML, you can run:

```
paco provision -y netenv.mynet.dev
paco provision -y netenv.mynet.prod
```

And watch Paco update the S3 Bucket policy for ALL of your replica buckets. Enjoy!

6.11 Private PyPI Server

The **Private PyPI Server** creates a PyPI server for hosting your own Python Packages. The server can be password protected to host private Python packages.

The Python packages are stored on an EFS filesystem mount. The PyPI server is hosted in an AutoScalingGroup and will automatically relaunch and remount the EFS filesystem if the server is terminated. Configuration is included which can be enabled to do routine back-ups on the EFS filesystem and monitor the PyPI server and alert if the server is not responding.

The PyPI server can be run with two network configurations: "budget" and "professional". The first configuration runs a simple single public network with the server directly serving requests to the internet with an ElasticIP.

The second configuration has public and private subnets with an an Application Load Balancer (ALB) in the public subnet and a NAT Gateway to allow the web server(s) internet access. This increases the overall cost of the solution but allows for more robust uptime and improved security. The Application Load Balancer is run in a separate application named "shared". This configuration is designed to show you how a real-world AWS deployment might run a single ALB and direct requests to a variety of backend resources.

6.11.1 Create a "Private PyPI Server" Project

Install Paco and then get started by running the `paco init project <your-project-name>` command. Review the instructions on Getting Started with Paco to understand the importance of `name` fields in Paco and the difference between a name and title. Then follow the instructions on creating credentials for your project to connect it to your AWS Account.

When asked "Budget - Lower cost but less robust set-up?" if you choose "Y" your configuration will run a single server in a public subnet, if you choose "N" your configuration will have public/private subnets and an Application Load Balancer and NAT Gateway. The latter configuration increases your monthly costs by over $30 USD per month but allows you to run a fault tolerant cluster of servers.

Take a minute to set-up a PACO_HOME environment variable, this will save you lots of time typing.

6.11.2 Provision SNS Topics and EC2 Keypairs

If you want to configure monitoring and alerting to let you know when your PyPI server is having potential problems, you will need to provision SNS Topics. The "admin" group group is configured to recieve any alarm notifications. The project default is to use the same email as your root account, but you can edit `resource/snstopics.yaml` and change this to whatever you want. See the SNS Topics docs for examples.

```
paco provision resource.snstopics
```

Next, you will need to have an EC2 SSH Keypair in the same account and region as your PyPI deployment. You can create a new keypair by running:

```
paco provision resource.ec2.keypairs
```

Alternatively, you can edit `resource/ec2.yaml` and configure it to use an existing EC2 Keypair, if you already have one created. See the EC2 Key pairs reference documentation for more information.

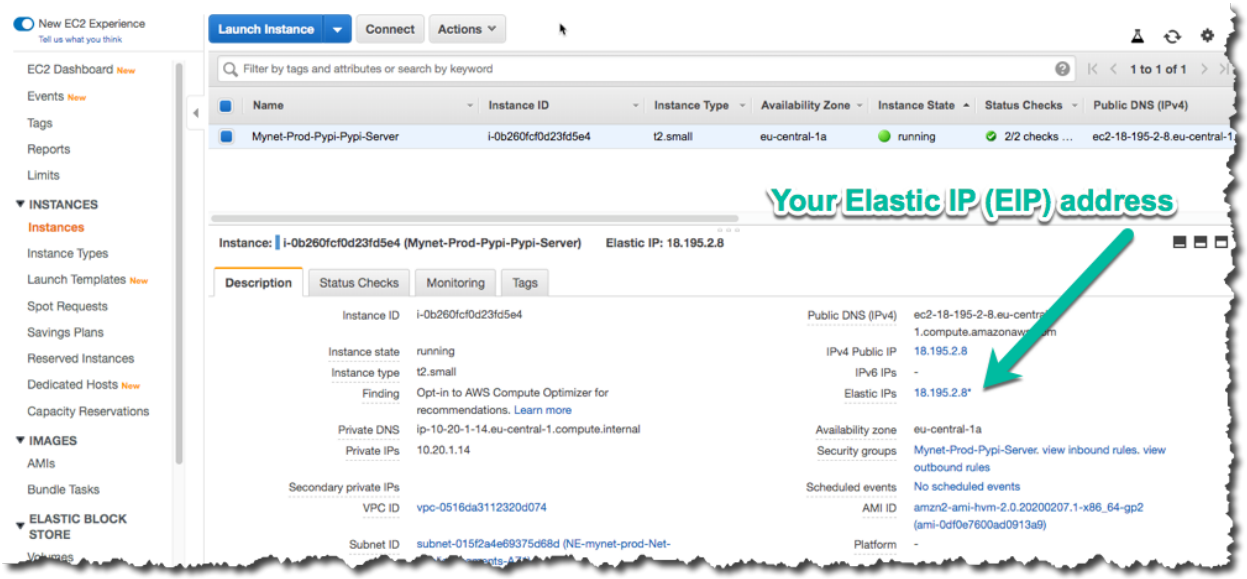
6.11.3 Provision the PyPI server resources

Now you are ready to provision the PyPI server. This will provision everything you need - VPC network, EFS filesystem, AutoScalingGroup server(s). After this step, you should have a fully working PyPI server!

```
paco provision netenv.mynet.prod
```

If you changed your NetworkEnvironment name, change the `mynet` string to that name. This starter project only contains a single "prod" environment. However, Paco supports easily cloning environments, so you could provision a completely duplicate "dev" or "test" environment if you did need a place to test new changes to your PyPI set-up without impacting your production environment.

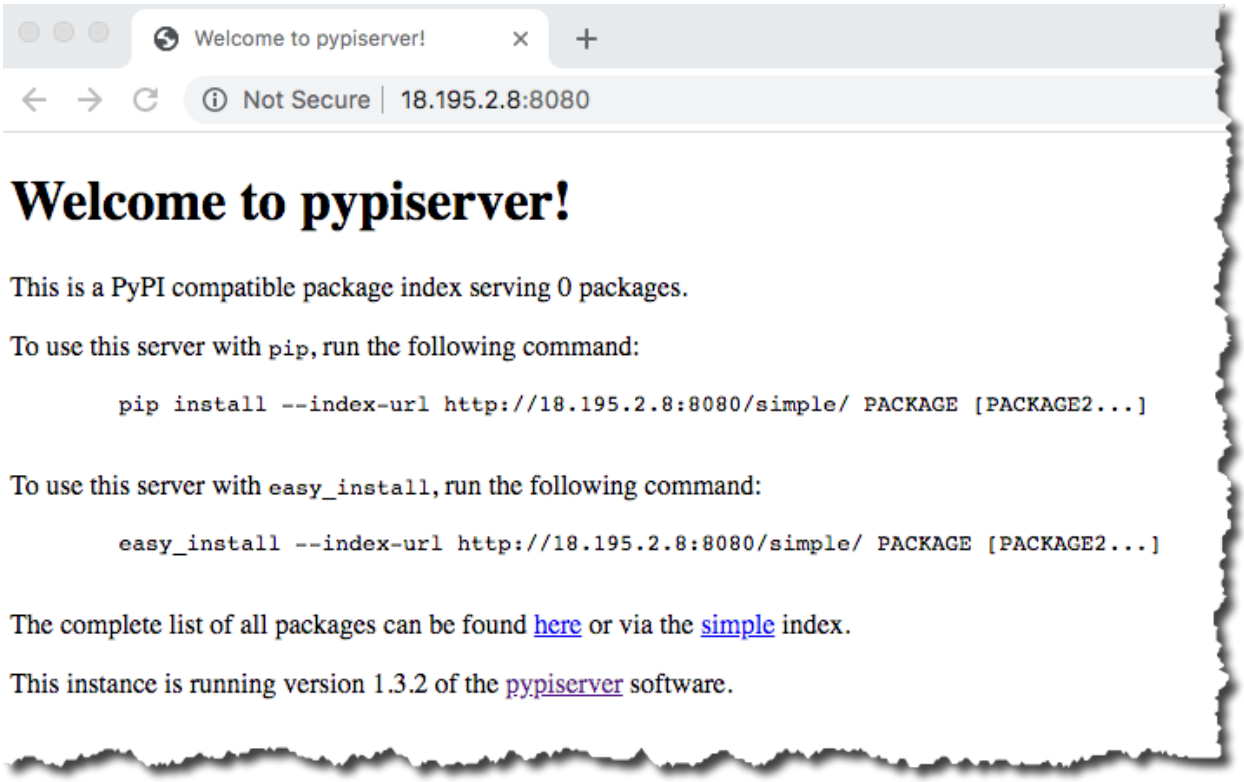
To visit your new PyPI server, if you are running the "budget" set-up, use the AWS Console to go to **EC2 –> Instances** and click on your server. Find the Elastic IPs field:



Copy that IP into your web browser and your server will be running on port 8080, for example: http://255.255.255.255:8080

The PyPI server runs as a `www-data` user for security reasons. This user doesn't have permissions to bind to port 80, so port 8080 is used instead.

If you are running the non-budget configuration, then you can go to **EC2 –> Load Balancers** in the AWS Console and find the **DNS name** for your ALB. Put this name into your web browser and you should see your PyPI server:



6.11.4 Using and managing your PyPI server

The PyPI server is running the pypiserver application. The default configuration runs the PyPI server as a private package server and requires authentication to download packages. It uses an `htpasswd.txt` file to contain the usernames and passwords for the server.

You can customize your PyPI server if you want to make it public or use a different authentication method. Visit the pypiserver GitHub page to see all of the option available for server configuration.

The PyPI server is started and configured automatically by SystemD when a new EC2 instance launches. The configuration management for the server is done with AWS CloudFormationInit. You will see a file in your Paco project at `netenv/pypi-config/pypiserver.service` which contains the PyPI server configuration. You can customize the configuration here, pay attention to the `ExecStart=` line which contains all of the configuration options supplied to the PyPI server.

Take note that your packages are served from the directory `/var/pypi/`. Paco has provided configuration to automatically mount this directory to an EFS filesystem mount. The default configuration will also have created a `/var/pypi/htpasswd.txt` file which contains a starting username and password of "uploader" and "Pass123...". This is a public example password, **you will want to change this file and replace it with private usernames and passwords!**

You will have a file at `netenv/pypi-config/htpasswd.txt` which contains an Apache-style htpasswd file. You can run the `htpasswd` CLI to add new usernames to this file:

```
htpasswd -b ./netenv/pypi-config/htpasswd.txt newuser NewPassword123!
```

The contents of this file is embedded into the CloudFormation template used to provision the PyPI server AutoScalingGroup. You can add and remove passwords locally and then run:

```
paco provision -y netenv.mynet.prod
```

This will apply changes to the file to the CloudFormation stack. The PyPI server is configured to run `cfn-hup` to automatically ping the CloudFormation service every 15 minutes and if new changes are detected, to automatically apply them to the PyPI server.

This is a simple and easy way to manage PyPI server passwords, but it does expose hashed passwords in both your Paco project (which you will likely want to keep in a git repo) and through your AWS Console. These passwords could then be compromised through brute force. If you want to run stricter security on your PyPI authentication, then change the authentication configuration and you can disable cfn-hup in your `netenv/mynet.yaml` file:

```
cfn-hup:
  enabled: false
  ensure_running: false
```

Finally,

6.11.5 Paco project configuration

Paco project's are designed to be easily configurable. Let's look at some configuration changes you could make to your project.

SSH Access

With the SSH keypair that you used to launch your PyPI server, you can SSH to your server. You may find this necessary if you want to remove packages from your PyPI server, or if you are trying to change the pypiserver

```
ssh -i ~/.ssh/my-keypair.pem ec2-user@<my-eip-address>
```

In the budget set-up, you can SSH directly to your server. You will see this configuration in your `netenv/mynet.yaml` file:

- port: 22 name: SSH protocol: tcp cidr_ip: 0.0.0.0/0 # change to your IP to improve security

Change the open to the world "cidr_ip: 0.0.0.0/0" string to your IP address (e.g. "cidr_ip: 120.90.255.120/32") and run `paco provision netenv.mynet.prod` to limit access to just your IP address.

With the ALB set-up, you will need to launch a bastion server to be able to SSH to your PyPI server. To enable the bastion you will need to enable it's configuration in the `environments.prod.applications` section:

```
environments:
  prod:
    title: "Production Environment"
    ...
    applications:
      pypi:
        enabled: true
      bastion:
        enabled: true
```

And then run `paco provision netenv.mynet.prod` and the instance will be launched.

Application Load Balancer (ALB)

The non-budget set-up runs behind an Application Load Balancer (ALB). This ALB can be configured to serve requests for other applications. In addition, you can conifgure it to connect to your Route 53 hosted zone for your domain and serve traffic over HTTPS.

The default configuration listens on port 80 only, but you will see commented out configuration to instead listen on port 443. In your `netenv/mynet.yaml` find this configuration under the "listeners:" field:

```
http:
  port: 80
  protocol: HTTP
  target_group: pypi
  rules:
    pypi_forward:
      enabled: true
      rule_type: forward
      host: 'pypi.example.com'
      target_group: pypi
      priority: 10
```

Remove this configuration and uncomment everything below it. Also uncomment the `example_com_cert:` section and change it's configuration to match the domain name you will use with your ALB. Finally you can uncommned the `dns:` section and Paco will take care of adding the ALB DNS name to your Route 53 Hosted Zone.

Later, if you add a second application behind your ALB, you might have configuration such as:

```
http:
  port: 80
  protocol: HTTP
  redirect:
    port: 443
    protocol: HTTPS
https:
  port: 443
  protocol: HTTPS
  ssl_certificates:
    - paco.ref netenv.mynet.applications.pypi.groups.shared.resources.example_com_cert
  target_group: pypi
  rules:
    pypi_forward:
      enabled: true
      rule_type: forward
      host: 'pypi.example.com'
      target_group: pypi
      priority: 10
    apptwo_forward:
      enabled: true
      rule_type: forward
      host: 'apptwo.example.com'
      target_group: apptwo
      priority: 20
```

Monitoring

The `netenv/mynet.yaml` file starts with monitoring disabled:

```
# enable/disable web server monitoring
groups:
  pypi:
    resources:
      server:
        monitoring:
          enabled: false # metrics are OFF
          alarm_sets:
            pypiserver:
# enable/disable HTTP checks
monitoring:
  enabled: false # health checks are OFF
  health_checks:
    external_ping:
      enabled: false # health checks are OFF
```

You can change the `monitoring` and `external_ping` sections to `enabled: true` and then run:

```
paco provision netenv.mynet.prod
```

This will install a CloudWatch Agent to gather in-host metrics from the PyPI servers and enable a Route 53 Health Check which will continually ping your server and alert you when it goes down.

There will also be CloudWatch Alarms created for StatusChecks, CPU, Swap and DiskSpace on the root volume.

Also be warned that enabling in-host metrics will change your AutoScalingGroup UserData which will cause new EC2 instance(s) to be launched. In the default configuration this will cause a few minutes downtime for your PyPI server.

Enabling in-host metrics, health checks and alarms will increase your AWS bill by a few dollars a month.

Backups

Your PyPI packages live on an AWS EFS filesystem. This is a very robust and reliable filesystem, however you may still like to maintain regular backups of this filesystem. In your `netenv/mynet.yaml` find this configuration:

```
backup_vaults:
  pypi:
    enabled: false # backups are off
```

And simply change it to `enabled: true` and run `paco provision netenv.mynet.prod`.

You can review your backup configuration in the `netenv/mynet.yaml` file:

```
backup_vaults:
  pypi:
    enabled: false
    plans:
      ebs:
        title: EFS Backups
        enabled: true
        plan_rules:
          - title: Daily EFS backups
            schedule_expression: cron(0 6 ? * * *)
            lifecycle_delete_after_days: 365
        selections:
          - title: EFS Daily Backups Selection
```

(continues on next page)

(continued from previous page)

```
            tags:
              - condition_type: STRINGEQUALS
                condition_key: Paco-Application-Name
                condition_value: pypi
```

This default configuration will do daily back-ups at 6 am every day, and keep backups for 365 days.

6.11.6 Questions?

This is only a sample of what you can do with Paco to configure and run an application such as a PyPI server on AWS. This documentation is far from exhaustive, so if you get stuck during installation or configuration, then you're welcome to jump on the paco-cloud Gitter community and ask us questions.

6.12 Configuration Basics

6.12.1 Paco configuration overview

Paco configuration is a complete declarative description of a cloud project. These files semantically describe cloud resources and logical groupings of those resources. The contents of these files describe accounts, networks, environments, applications, resources, services, and monitoring configuration.

The Paco configuration files are parsed into a Python object model by the library `paco.models`. This object model is used by Paco to provision AWS resources using CloudFormation. However, the object model is a standalone Python package and can be used to work with cloud infrastructure semantically with other tooling.

File format overview

Paco configuration is a directory of files and sub-directories that make up a Paco project. All of the files are in YAML format.

In the top-level directory are sub-directories that contain YAML files each with a different format. This directories are:

- `accounts/`: Each file in this directory is an AWS account.
- `netenv/`: Each file in this directory defines a complete set of networks, applications and environments. Environments are provisioned into your accounts.
- `monitor/`: These contain alarm and logging configuration.
- `resource/`: For global resources, such as S3 Buckets, IAM Users, EC2 Keypairs.
- `service/`: For extension plug-ins.

Also at the top level are `project.yaml` and `paco-project-version.txt` files.

The `paco-project-version.txt` is a simple one line file with the version of the Paco project file format, e.g. `2.1`. The Paco project file format version contains a major and a medium version. The major version indicates backwards incompatable changes, while the medium version indicates additions of new object types and fields.

The `project.yaml` contains gloabl information about the Paco project. It also contains an `paco_project_version` field that is loaded from `paco-project-version.txt`.

The YAML files are organized as nested key-value dictionaries. In each sub-directory, key names map to relevant Paco schemas. A Paco schema is a set of fields that describe the field name, type and constraints.

An example of how this hierarchy looks, in a NetworksEnvironent file, a key name `network:` must have attributes that match the Network schema. Within the Network schema there must be an attribute named `vpc:` which contains attributes for the VPC schema. That looks like this:

```
network:
    enabled: true
    region: us-west-2
    availability_zones: 2
    vpc:
        enable_dns_hostnames: true
        enable_dns_support: true
        enable_internet_gateway: true
```

Some key names map to Paco schemas that are containers. For containers, every key must contain a set of key/value pairs that map to the Paco schema that container is for. Every Paco schema in a container has a special `name` attribute, this attribute is derived from the key name used in the container.

For example, the NetworkEnvironments has a key name `environments:` that maps to an Environments container object. Environments containers contain Environment objects.

```
environments:
    dev:
        title: Development
    staging:
        title: Staging
    prod:
        title: Production
```

When this is parsed, there would be three Environment objects:

```
Environment:
    name: dev
    title: Development
Environment:
    name: staging
    title: Staging
Environment:
    name: prod
    title: Production
```

Attention: Key naming warning: As the key names you choose will be used in the names of resources provisioned in AWS, they should be as short and simple as possible. If you wanted rename keys, you need to first delete all of your AWS resources under their old key names, then recreate them with their new name. Try to give everything short, reasonable names.

Key names have the following restrictions:

- Can contain only letters, numbers, hyphens and underscores.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Certain AWS resources have additional naming limitations, namely S3 bucket names can not contain uppercase letters and certain resources have a name length of 64 characters.

The `title` field is available in almost all Paco schemas. This is intended to be a human readable name. This field can contain any character except newline. The `title` field can also be added as a Tag to resources, so any characters

beyond 255 characters would be truncated.

6.12.2 Enabled/Disabled

Many Paco schemas have an `enabled:` field. If an Environment, Application or Resource field have `enabled: True`, that indicates it should be provisioned. If `enabled: False` is set, then the resource won't be provisioned.

To determine if a resource should be provisioned or not, if **any** field higher in the tree is set to `enabled: False` the resource will not be provisioned.

In the following example, the network is enabled by default. The dev environment is enabled, and there are two applications, but only one of them is enabled. The production environment has two applications enabled, but they will not be provisioned as enabled is off for the entire environment.

```
network:
    enabled: true

environments:
    dev:
        enabled: true
        default:
            applications:
                my-paco-example:
                    enabled: false
                reporting-app:
                    enabled: true
    prod:
        enabled: false
        default:
            applications:
                my-paco-example:
                    enabled: true
                reporting-app:
                    enabled: true
```

Attention: Note that currently, this field is only applied during the `paco provision` command. If you want delete an environment or application, you need to do so explicitly with the `paco delete` command.

6.12.3 References and Substitutions

Some values can be special references. These will allow you to reference other values in your Paco Configuration.

- `paco.ref netenv`: NetworkEnvironment reference
- `paco.ref resource`: Resource reference
- `paco.ref accounts`: Account reference
- `paco.ref function`: Function reference
- `paco.ref service`: Service reference

References are in the format:

`type.ref name.seperated.by.dots`

In addition, the `paco.sub` string indicates a substitution.

paco.ref netenv

To refer to a value in a NetworkEnvironment use an `paco.ref netenv` reference. For example:

`paco.ref netenv.my-paco-example.network.vpc.security_groups.app.lb`

After `paco.ref netenv` should be a part which matches the filename of a file (without the .yaml or .yml extension) in the NetworkEnvironments directory.

The next part will start to walk down the YAML tree in the specified file. You can either refer to a part in the `applications` or `network` section.

Keep walking down the tree, until you reach the name of a field. This final part is sometimes a field name that you don't supply in your configuration, and is instead can be generated by the Paco Engine after it has provisioned the resource in AWS.

An example where a `paco.ref netenv` refers to the id of a SecurityGroup:

```
network:
    vpc:
        security_groups:
            app:
                lb:
                    egress
                webapp:
                    ingress:
                        - from_port: 80
                        name: HTTP
                        protocol: tcp
                        source_security_group: paco.ref netenv.my-paco-example.
↪network.vpc.security_groups.app.lb
```

You can refer to an S3 Bucket and it will return the ARN of the bucket:

```
artifacts_bucket: paco.ref netenv.my-paco-example.applications.app.groups.cicd.
↪resources.cpbd_s3
```

SSL Certificates can be added to a load balancer. If a reference needs to look-up the name or id of an AWS Resource, it needs to first be provisioned, the `order` field controls the order in which resources are created. In the example below, the ACM cert is first created, then an Applicatin Load Balancer is provisioned and configured with the ACM cert:

```
applications:
    app:
        groups:
            site:
                cert:
                    type: ACM
                    order: 1
                    domain_name: example.com
                    subject_alternative_names:
                    - '*.example.com'
                alb:
                    type: LBApplication
                    order: 2
                    listeners:
                        - port: 80
                          protocol: HTTP
                          redirect:
```

(continues on next page)

(continued from previous page)

```
                            port: 443
                            protocol: HTTPS
                        - port: 443
                            protocol: HTTPS
                            ssl_certificates:
                            - paco.ref netenv.my-paco-example.applications.app.groups.
↪site.resources.cert
```

paco.ref resource

To refer to a global resource created in the Resources directory, use an `paco.ref resource`. For example:

`paco.ref resource.route53.example`

After the `paco.ref resource` the next part should matche the filename of a file (without the .yaml or .yml extension) in the Resources directory. Subsequent parts will walk down the YAML in that file.

In the example below, the `hosted_zone` of a Route53 record is looked up.

```
# netenv/my-paco-example.yaml

applications:
    app:
        groups:
            site:
                alb:
                    dns:
                    - hosted_zone: paco.ref resource.route53.example

# resource/Route53.yaml

hosted_zones:
example:
    enabled: true
    domain_name: example.com
    account: paco.ref accounts.prod
```

paco.ref accounts

To refer to an AWS Account in the Accounts directory, use `paco.ref`. For example:

`paco.ref accounts.dev`

Account references should matches the filename of a file (without the .yaml or .yml extension) in the Accounts directory.

These are useful to override in the environments section in a NetworkEnvironment file to control which account an environment should be deployed to:

```
environments:
    dev:
        network:
            aws_account: paco.ref accounts.dev
```

paco.ref function

A reference to an imperatively generated value that is dynamically resolved at runtime. For example:

`paco.ref function.mypackage.mymodule.myfunction`

This must be an importable Python function that accepts three arguements: reference, project, account_ctx.

This function must return a value that is compatable with the fields data type (e.g. typically a string).

There is one built-in function:

`paco.ref function.aws.ec2.ami.latest.amazon-linux-2`

Currently can only look-up AMI IDs. Can be either `aws.ec2.ami.latest.amazon-linux-2` or `aws.ec2.ami.latest.amazon-linux`.

```
web:
    type: ASG
    instance_ami: paco.ref function.aws.ec2.ami.latest.amazon-linux-2
```

paco.ref service

To refer to a service created in the Services directory, use an `paco.ref service`. For example:

`paco.ref service.notification.<account>.<region>.applications.notification.groups.lambda.resources.snstopic`

Services are plug-ins that extend Paco with additional functionality. For example, custom notification, patching, back-ups and cost optimization services could be developed and installed into a Paco application to provide custom business functionality.

paco.sub

Can be used to look-up a value and substitute the results into a templated string.

6.13 YAML File Schemas

6.13.1 Base Schemas

Base Schemas are never configured by themselves, they are schemas that are inherited by other schemas.

Interface

A generic placeholder for any schema.

Named

A name given to a cloud resource. Names identify resources and changing them can break configuration.

Table 1: *Named*

Field name	Type	Purpose	Constraints	Default
name	String	Name		

Base Schemas Title

Title

A title is a human-readable name. It can be as long as you want, and can change without breaking any configuration.

Table 2: *Title*

Field name	Type	Purpose	Constraints	Default
title	String	Title		

Name

A name that can be changed or duplicated with other similar cloud resources without breaking anything.

Table 3: *Name*

Field name	Type	Purpose	Constraints	Default
name	String	Name		

Resource

Configuration for a cloud resource. Resources may represent a single physical resource in the cloud, or several closely related resources.

Table 4: *Resource*

Field name	Type	Purpose	Constraints	Default
change_protected	Boolean	Boolean indicating whether this resource can be modified or not.		False
order	Int	The order in which the resource will be deployed		0

Base Schemas DNSEnablable, Deployable, Named, Title, Type

Deployable

Indicates if this configuration tree should be enabled or not.

Table 5: *Deployable*

Field name	Type	Purpose	Constraints	Default
enabled	Boolean	Enabled	Could be deployed to AWS	False

Enablable

Indicate if this configuration should be enabled.

Table 6: *Enablable*

Field name	Type	Purpose	Constraints	Default
enabled	Boolean	Enabled		True

Type

Table 7: *Type*

Field name	Type	Purpose	Constraints	Default
type	String	Type of Resources	A valid AWS Resource type: ASG, LBApplication, etc.	

DNSEnablable

Provides a parent with an inheritable DNS enabled field

Table 8: *DNSEnablable*

Field name	Type	Purpose	Constraints	Default
dns_enabled	Boolean	Boolean indicating whether DNS record sets will be created.		True

Monitorable

A monitorable resource

Table 9: *Monitorable*

Field name	Type	Purpose	Constraints	Default
monitoring	Object<*MonitorConfig*>			

MonitorConfig

A set of metrics and a default collection interval

Table 10: *MonitorConfig*

Field name	Type	Purpose	Constraints	Default
alarm_sets	Container<AlarmSets>	Sets of Alarm Sets		
asg_metrics	List<String>	ASG Metrics	Must be one of: 'GroupMinSize', 'GroupMaxSize', 'GroupDesiredCapacity', 'GroupInServiceInstances', 'GroupPendingInstances', 'GroupStandbyInstances', 'GroupTerminatingInstances', 'GroupTotalInstances'	
collection_interval	Int	Collection interval		60
health_checks	Container<HealthChecks>	Set of Health Checks		
log_sets	Container<CloudWatchLogSets>	Sets of Log Sets		
metrics	List<Metric>	Metrics		

Base Schemas *Deployable*, *Named*, *Notifiable*, *Title*

RegionContainer

Container for objects which do not belong to a specific Environment.

Table 11: *RegionContainer*

Field name	Type	Purpose	Constraints	Default
alarm_sets	Container<AlarmSets>	Alarm Sets		

Base Schemas *Named*, *Title*

AccountRegions

An Account and one or more Regions

Table 12: *AccountRegions*

Field name	Type	Purpose	Constraints	Default
account	PacoReference	AWS Account	Paco Reference to Account.	
regions	List<String>	Regions		

Notifiable

A notifiable object

Table 13: *Notifiable*

Field name	Type	Purpose	Constraints	Default
notifications	Container<AlarmNotifications>	Alarm Notifications		

SecurityGroupRule

Table 14: *SecurityGroupRule*

Field name	Type	Purpose	Constraints	Default
cidr_ip	String	CIDR IP	A valid CIDR v4 block or an empty string	
cidr_ip_v6	String	CIDR IP v6	A valid CIDR v6 block or an empty string	
description	String	Description	Max 255 characters. Allowed characters are a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=;{}!$*.	
from_port	Int	From port	A value of -1 indicates all ICMP/ICMPv6 types. If you specify all ICMP/ICMPv6 types, you must specify all codes.	-1
port	Int	Port	A value of -1 indicates all ICMP/ICMPv6 types. If you specify all ICMP/ICMPv6 types, you must specify all codes.	-1
protocol	String	IP Protocol	The IP protocol name (tcp, udp, icmp, icmpv6) or number.	
to_port	Int	To port	A value of -1 indicates all ICMP/ICMPv6 types. If you specify all ICMP/ICMPv6 types, you must specify all codes.	-1

Base Schemas Name

ApplicationEngine

Application Engine : A template describing an application

Table 15: *ApplicationEngine*

Field name	Type	Purpose	Constraints	Default
groups	Container<ResourceGroups>			
order	Int	The order in which the application will be processed		0

Base Schemas *DNSEnablable*, *Deployable*, *Monitorable*, *Named*, *Notifiable*, *Title*

VPCConfiguration

Table 16: *VPCConfiguration*

Field name	Type	Purpose	Constraints	Default
security_groups	List<PacoReference>	List of VPC Security Group Ids	Paco Reference to SecurityGroup.	
segments	List<PacoReference>	VPC Segments to attach the function	Paco Reference to Segment.	

Base Schemas *Named*, *Title*

HostedZone

Base interface for IRoute53HostedZone and IPrivateHostedZone

- –
  –
  –
  –
  –

Function

A callable function that returns a value.

6.14 Accounts

AWS account information is kept in the `accounts/` directory. Each file in this directory defines one AWS account, the filename is the `name` of the account, with a .yml or .yaml extension.

Listing 1: Typical accounts directory

```
accounts/
  dev.yaml
  master.yaml
  prod.yaml
  tools.yaml
```

6.14.1 Account

Cloud accounts.

The specially named *master.yaml* file is for the AWS Master account. It is the only account which can have the field *organization_account_ids* which is used to define and create the child accounts.

Listing 2: Example accounts/master.yaml account file

```
name: Master
title: Master AWS Account
is_master: true
account_type: AWS
account_id: '123456789012'
region: us-west-2
organization_account_ids:
  - prod
  - tools
  - dev
root_email: master@example.com
```

Listing 3: Example accounts/dev.yaml account file

```
name: Development
title: Development AWS Account
account_type: AWS
account_id: '123456789012'
region: us-west-2
root_email: dev@example.com
```

Table 17: *Account*

Field name	Type	Purpose	Constraints	Default
account_id	String	Account ID	Can only contain digits.	
account_type	String	Account Type	Supported types: 'AWS'	AWS
admin_delegate_role_name	String	Administrator delegate IAM Role name for the account		Paco-Organization-Account-Delegate-Role
admin_iam_users	Container<*AdminIAMUsers*>	Admin IAM Users		
is_master	Boolean	Boolean indicating if this a Master account		False
organization_account_ids	List<String>	A list of account ids to add to the Master account's AWS Organization	Each string in the list must contain only digits.	
region	String	Region to install AWS Account specific resources	Must be a valid AWS Region name	no-region-set
root_email	String	The email address for the root user of this account	Must be a valid email address.	

Base Schemas Deployable, Named, Title

6.14.2 AdminIAMUsers

A container for AdminIAMUser objects

Table 18: *AdminIAMUsers* Container<AdminIAMUser>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

6.14.3 AdminIAMUser

An AWS Account Administerator IAM User

Table 19: *AdminIAMUser*

Field name	Type	Purpose	Constraints	Default
username	String	IAM Username		

Base Schemas Deployable, Named, Title

6.15 Global Resources

Global Resources are defined in the top-level `resource/` directory. They define cloud resources which do not belong to an environment or other logical grouping.

6.15.1 CloudTrail

The `resource/cloudtrail.yaml` file specifies CloudTrail resources.

AWS CloudTrail logs all AWS API activity. Monitor and react to changes in your AWS accounts with CloudTrail. A CloudTrail can be used to set-up a multi-account CloudTrail that sends logs from every account into a single S3 Bucket.

```
paco provision resource.cloudtrail
```

Prescribed Automation

`enable_kms_encryption`: Encrypt the CloudTrai logs with a Customer Managed Key (CMK). Paco will create a CMK for the CloudTrail in the same account as the `s3_bucket_account`.

`kms_users`: A list of either IAM User names or paco references to `resource/iam.yaml` users. These users will have access to the CMK to decrypt and read the CloudTrail logs.

Listing 4: example resource/cloudtrail.yaml configuration

```
trails:
  mycloudtrail:
    enabled: true
    region: 'us-west-2'
    cloudwatchlogs_log_group:
      expire_events_after_days: '14'
      log_group_name: CloudTrail
    enable_log_file_validation: true
    include_global_service_events: true
    is_multi_region_trail: true
    enable_kms_encryption: true
    kms_users:
      - bob@example.com
      - paco.ref resource.iam.users.sallysmith
    s3_bucket_account: paco.ref accounts.security
    s3_key_prefix: cloudtrails
```

Table 20: *CloudTrail*

Field name	Type	Purpose	Constraints	Default
accounts	List<PacoReference>	Accounts to enable this CloudTrail in. Leave blank to assume all accounts.	Paco Reference to Account.	
cloudwatchlogs_loggroup	Object<CloudWatchLogGroup>	CloudWatch Logs LogGroup to deliver this trail to.		
enable_kms_encryption	Boolean	Enable KMS Key encryption		False
enable_log_file_validation	Boolean	Enable log file validation		True
include_global_service_events	Boolean	Include global service events		True
is_multi_region_trail	Boolean	Is multi-region trail?		True
kms_users	List<PacoReference>	IAM Users with access to CloudTrail bucket	Paco Reference to *IAMUser*. String Ok.	
region	String	Region to create the CloudTrail	Must be a valid AWS Region name or empty string	
s3_bucket_account	PacoReference	Account which will contain the S3 Bucket where the CloudTrail is stored.	Must be an paco.ref to an account Paco Reference to Account.	
s3_key_prefix	String	S3 Key Prefix specifies the Amazon S3 key prefix that comes after the name of the bucket.	Do not include a leading or trailing / in your prefix. They are provided already.	

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

6.15.2 CodeCommit

The `resource/codecommit.yaml` file manages CodeCommit repositories and users. The top-level of the file is CodeCommitRepositoryGroups, and each group contains a set of CodeCommit Repositories.

Listing 5: Example resource/codecommit.yaml file

```
# Application CodeCommitRepositoryGroup
application:
  # SaaS API CodeCommitRepository
  saas-api:
    enabled: true
    account: paco.ref accounts.tools
    region: us-west-2
    description: "SaaS API"
    repository_name: "saas-api"
    users:
      bobsnail:
        username: bobsnail@example.com
        public_ssh_key: 'ssh-rsa AAAAB3Nza.........6OzEFxCbJ'
```

(continues on next page)

(continued from previous page)

```
  # SaaS UI CodeCommitRepository
  saas-ui:
    enabled: true
    account: paco.ref accounts.tools
    region: us-west-2
    description: "Saas UI"
    repository_name: "saas-ui"
    users:
      bobsnail:
        username: bobsnail@example.com
        public_ssh_key: 'ssh-rsa AAAAB3Nza.........6OzEFxCbJ'
      external_dev_team:
        username: external_dev_team
        public_ssh_key: 'ssh-rsa AAZA5RNza.........6OzEGHb7'

# Docs CodeCommitRepositoryGroups
docs:
  saas-book:
    enabled: true
    account: paco.ref accounts.prod
    region: eu-central-1
    description: "The SaaS Book (PDF)"
    repository_name: "saas-book"
    users:
      bobsnail:
        username: bobsnail@example.com
        public_ssh_key: 'ssh-rsa AAAAB3Nza.........6OzEFxCbJ'
```

Provision CodeCommit repos and users with:

```
paco provision resource.codecommit
```

Be sure to save the AWS SSH key ID for each user after your provision their key. You can also see the SSH keys in the AWS Console in the IAM Users if you lose them.

Visit the CodeCommit service in the AWS Console to see the SSH Url for a Git repo.

To authenticate, if you are using your default public SSH key, you can embed the AWS SSH key ID as the user in SSH Url:

```
git clone ssh://APKAV........63ICK@server/project.git
```

Or add the AWS SSH key Id to your *~/.ssh/config* file. This is the easiest way, especially if you have to deal with multiple SSH keys on your workstation:

```
Host git-codecommit.*.amazonaws.com
  User APKAV........63ICK
  IdentityFile ~/.ssh/my_pubilc_key_rsa
```

CodeCommit

Container for *CodeCommitRepositoryGroup* objects.

Table 21: *CodeCommit* Container<CodeCommitRepositoryGroup>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CodeCommitRepositoryGroup

Container for *CodeCommitRepository* objects.

Table 22: *CodeCommitRepositoryGroup* Container<CodeCommitRepository>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CodeCommitRepository

CodeCommit Repository

Table 23: *CodeCommitRepository*

Field name	Type	Purpose	Constraints	Default
account	PacoReference	Account this repo belongs to.	Paco Reference to Account.	
description	String	Repository Description		
external_resource	Boolean	Boolean indicating whether the CodeCommit repository already exists or not		False
region	String	AWS Region		
repository_name	String	Repository Name		
users	Container<*CodeCommitUser*>	CodeCommit Users		

Base Schemas Deployable, Named, Title

CodeCommitUser

CodeCommit User

Table 24: *CodeCommitUser*

Field name	Type	Purpose	Constraints	Default
permissions	Choice	Permissions	Must be one of ReadWrite or ReadOnly	ReadWrite
public_ssh_key	String	CodeCommit User Public SSH Key		
username	String	CodeCommit Username		

Base Schemas Named, Title

6.15.3 EC2 Keypairs

The `resource/ec2.yaml` file manages AWS EC2 Keypairs.

```
paco provision resource.ec2.keypairs # all keypairs
paco provision resource.ec2.keypairs.devnet_usw2 # single keypair
```

Listing 6: Example resource/ec2.yaml file

```
keypairs:
  devnet_usw2:
    keypair_name: "dev-us-west-2"
    region: "us-west-2"
    account: paco.ref accounts.dev
  staging_cac1:
    keypair_name: "staging-us-west-2"
    region: "ca-central-1"
    account: paco.ref accounts.stage
  prod_usw2:
    keypair_name: "prod-us-west-2"
    region: "us-west-2"
    account: paco.ref accounts.prod
```

EC2KeyPair

EC2 SSH Key Pair

Table 25: *EC2KeyPair*

Field name	Type	Purpose	Constraints	Default
account	PacoReference	AWS Account the key pair belongs to	Paco Reference to Account.	
keypair_name	String	The name of the EC2 KeyPair		
region	String	AWS Region	Must be a valid AWS Region name	no-region-set

Base Schemas Named, Title

6.15.4 IAM

The `resource/iam.yaml` file contains IAM Users. Each user account can be given different levels of access a set of AWS accounts. For more information on how IAM Users can be managed, see Managing IAM Users with Paco.

```
paco provision resource.iam.users
```

IAMResource

IAM Resource contains IAM Users who can login and have different levels of access to the AWS Console and API.

Table 26: *IAMResource*

Field name	Type	Purpose	Constraints	Default
users	Container<*IAMUsers*>	IAM Users		

Base Schemas Named, Title

IAMUsers

Container for *IAMUser* objects.

Table 27: *IAMUsers* Container<IAMUser>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

IAMUser

IAM User represents a user that will exist in one account, but can also have delegate IAM Roles in other accounts that they are allowed to assume.

Listing 7: example IAM User

```
enabled: true
account: paco.ref accounts.master
username: yourusername
description: 'Your Name - Paco Administrator'
console_access_enabled: true
programmatic_access:
  enabled: true
  access_key_1_version: 1
  access_key_2_version: 0
account_whitelist: all
permissions:
  administrator:
    type: Administrator
    accounts: all
  custom:
    accounts: dev
    managed_policies:
       - 'AWSDirectConnectReadOnlyAccess'
       - 'AmazonGlacierReadOnlyAccess'
    policies:
      - name: "AWS Polly full access"
        statement:
          - effect: Allow
            action:
              - 'polly:*'
            resource:
              - '*'
            condition:
              StringEquals:
                aws:username:
                  "yourusername"
```

Table 28: *IAMUser*

Field name	Type	Purpose	Constraints	Default
account	PacoReference	Paco account reference to install this user	Paco Reference to Account.	
account_whitelist	CommaList	Comma separated list of Paco AWS account names this user has access to		
console_access_enabled	Boolean	Console Access Boolean		
description	String	IAM User Description		
permissions	Container<*IAMUserPermissions*>	Paco IAM User Permissions		
programmatic_access	Object<*IAMUserProgrammaticAccess*>	Programmatic Access		
username	String	IAM Username		

Base Schemas Deployable, Named, Title

IAMUserProgrammaticAccess

IAM User Programmatic Access Configuration

Table 29: *IAMUserProgrammaticAccess*

Field name	Type	Purpose	Constraints	Default
access_key_1_version	Int	Access key version id		0
access_key_2_version	Int	Access key version id		0

Base Schemas Enablable

IAMUserPermissions

Container for IAM User Permission objects.

Table 30: *IAMUserPermissions*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

BaseRole

Table 31: *BaseRole*

Field name	Type	Purpose	Constraints	Default
assume_role_policy	Object<*AssumeRolePolicy*>	Assume role policy		
global_role_name	Boolean	Role name is globally unique and will not be hashed		False
instance_profile	Boolean	Instance profile		False
managed_policy_arns	List<String>	Managed policy ARNs		
max_session_duration	Int	Maximum session duration	The maximum session duration (in seconds)	3600
path	String	Path		/
permissions_boundary	String	Permissions boundary ARN	Must be valid ARN	
policies	List<*Policy*>	Policies		
role_name	String	Role name		

Base Schemas Named, Title

Role

IAM Role that is disabled by default

Table 32: *Role*

Field name	Type	Purpose	Constraints	Default

Base Schemas *BaseRole*, Deployable, Named, Title

RoleDefaultEnabled

IAM Role that is enabled by default

Table 33: *RoleDefaultEnabled*

Field name	Type	Purpose	Constraints	Default

Base Schemas *BaseRole*, Enablable, Named, Title

AssumeRolePolicy

Table 34: *AssumeRolePolicy*

Field name	Type	Purpose	Constraints	Default
aws	List<String>	List of AWS Principals		
effect	Choice	Effect	Must be one of 'Allow' or 'Deny'	
service	List<String>	Service		

Policy

Table 35: *Policy*

Field name	Type	Purpose	Constraints	Default
name	String	Policy name		
statement	List<*Statement*>	Statements		

Statement

Table 36: *Statement*

Field name	Type	Purpose	Constraints	Default
action	List<String>	Action(s)		
condition	Dict	Condition	Each Key is the Condition name and the Value must be a dictionary of request filters. e.g. { "StringEquals" : { "aws:username" : "johndoe" }}	{}
effect	Choice	Effect	Must be one of 'Allow' or 'Deny'	
principal	Object<*Principal*>	Principal		
resource	List<String>	Resrource(s)		

Base Schemas Named, Title

Principal

Table 37: *Principal*

Field name	Type	Purpose	Constraints	Default
aws	List<String>	List of AWS Principals		
service	List<String>	List of AWS Service Principals		

Base Schemas Named, Title

6.15.5 Route 53

Route53Resource

The `resource/route53.yaml` file manages AWS Route 53 hosted zones.

Provision Route 53 with:

```
paco provision resource.route53
```

Listing 8: Example resource/route53.yaml file

```
hosted_zones:
  example:
    enabled: true
    domain_name: example.com
    account: aim.ref accounts.prod
```

Table 38: *Route53Resource*

Field name	Type	Purpose	Constraints	Default
hosted_zones	Container<*Route53HostedZone*>	Hosted Zones		

Base Schemas Named, Title

Route53HostedZone

Route53 Hosted Zone

Table 39: *Route53HostedZone*

Field name	Type	Purpose	Constraints	Default
account	PacoReference	Account this Hosted Zone belongs to	Paco Reference to Account.	
domain_name	String	Domain Name		
external_resource	Object<*Route53HostedZoneExternalResource*>	External HostedZone Id Configuration		
parent_zone	String	Parent Hozed Zone name		
private_hosted_zone	Boolean	Make this hosted zone private.		False
record_sets	List<*Route53RecordSet*>	List of Record Sets		
vpc_associations	PacoReference	The VPC the private hosted zone will be provisioned in.	Paco Reference to VPC.	

Base Schemas Deployable, Named, Title

Route53HostedZoneExternalResource

Existing Hosted Zone configuration

Table 40: *Route53HostedZoneExternalResource*

Field name	Type	Purpose	Constraints	Default
hosted_zone_id	String	ID of an existing Hosted Zone		
nameservers	List<String>	List of the Hosted Zones Name-servers		

Base Schemas Deployable, Named, Title

Route53RecordSet

Route53 Record Set

Table 41: *Route53RecordSet*

Field name	Type	Purpose	Constraints	Default
record_name	String	Record Set Full Name		
resource_records	List<String>	Record Set Values		
ttl	Int	Record TTL		300
type	String	Record Set Type		

6.15.6 SNS Topics

The `resource/snstopics.yaml` file manages AWS Simple Notification Service (SNS) resources. SNS has only two resources: SNS Topics and SNS Subscriptions.

```
paco provision resource.snstopics
```

Listing 9: Example resource/snstopics.yaml file

```
account: paco.ref accounts.prod
regions:
  - 'us-west-2'
  - 'us-east-1'
groups:
  admin:
    title: "Administrator Group"
    enabled: true
    cross_account_access: true
    subscriptions:
      - endpoint: http://example.com/yes
        protocol: http
      - endpoint: https://example.com/orno
        protocol: https
      - endpoint: bob@example.com
        protocol: email
      - endpoint: bob@example.com
        protocol: email-json
      - endpoint: '555-555-5555'
        protocol: sms
      - endpoint: arn:aws:sqs:us-east-2:444455556666:queue1
        protocol: sqs
      - endpoint: arn:aws:sqs:us-east-2:444455556666:queue1
        protocol: application
```

(continues on next page)

(continued from previous page)

```
      - endpoint: arn:aws:lambda:us-east-1:123456789012:function:my-function
        protocol: lambda
```

Prescribed Automation

`cross_account_access`: Creates an SNS Topic Policy which will grant all of the AWS Accounts in this Paco Project access to the `sns.Publish` permission for this SNS Topic.

You will need this if you want to send CloudWatch Alarms from multiple accounts to the same SNS Topic(s) in one account.

6.16 NetworkEnvironments

NetworkEnvironments are files in the top-level `netenv/` directory.

NetworkEnvironments are the core of any Paco project. Every .yaml file in the `netenv` directory contains information about networks, applications and environments. These files define how environments are provisioned and which networks and applications will be provisioned in each one.

NetworkEnvironment files are hierarchical. They are nested many levels deep. At each node in the hierarchy a different field schema is used. The top level has several key names: `network:`, `secrets_manager:`, `backup_vaults:`, `applications:` and `environments:`. The `network:` must contain a key/value pairs that matches a NetworkEnvironment schema. The `applications:` and `environments:` are containers that hold Application and Environment schemas.

```
network:
    availability_zones: 2
    enabled: true
    region: us-west-2
    # more network YAML here ...

applications:
    my-paco-app:
        # more application YAML here ...
    reporting-app:
        # more application YAML here ...

environments:
    dev:
        title: Development Environment
        # more environment YAML here ...
    prod:
        title: Production Environment
        # more environment YAML here ...
```

The network, applications, backup_vaults and secrets_manager configuration sections hold logical configuration - this configuration does not get direclty provisioned to the cloud - it doesn't reference any environments or regions. Think of it as default configuration.

Environments are where actual cloud resources are declared to be provisioned. Environments reference the default configuration from networks, applications, backups and secrets and declare which account(s) and region(s) to provision them in.

In environments, any field from the default configuration being referenced can be overridden. This could be used for running a smaller instance size in the dev environment, enabling monitoring only in a production environment, or specifying a different git branch name for a CI/CD for each environment.

6.16.1 Network

The network config type defines a complete logical network: VPCs, Subnets, Route Tables, Network Gateways. The applications defined later in this file will be deployed into networks that are built from this network template.

Networks have the following hierarchy:

```
network:
    # general config here ...
    vpc:
        # VPC config here ...
        nat_gateway:
            # NAT gateways container
        vpn_gateway:
            # VPN gateways container
        private_hosted_zone:
            # private hosted zone config here ...
        security_groups:
            # security groups here ...
```

SecurityGroups have two level nested names. These can be any names, but typically the first name is the name of an application and the second name is for a resource in that application. However, other name schemes are possible to support workloads sharing the same Security Groups.

Listing 10: Example security_groups configuration

```
  network:
    vpc:
      security_groups:
        myapp:
          lb:
            egress:
              - cidr_ip: 0.0.0.0/0
                name: ANY
                protocol: "-1"
            ingress:
              - cidr_ip: 128.128.255.255/32
                from_port: 443
                name: HTTPS
                protocol: tcp
                to_port: 443
              - cidr_ip: 128.128.255.255/32
                from_port: 80
                name: HTTP
                protocol: tcp
                to_port: 80
          web:
            egress:
              - cidr_ip: 0.0.0.0/0
                name: ANY
                protocol: "-1"
            ingress:
              - from_port: 80
```

(continues on next page)

(continued from previous page)

```
                name: HTTP
                protocol: tcp
                source_security_group: paco.ref netenv.my-paco-example.network.vpc.
↪security_groups.app.lb
                to_port: 80
```

NetworkEnvironment

NetworkEnvironment

Table 42: *NetworkEnvironment*

Field name	Type	Purpose	Constraints	Default

Base Schemas Deployable, Named, Title

Network

Table 43: *Network*

Field name	Type	Purpose	Constraints	Default
availability_zones	Int	Availability Zones		0
aws_account	PacoReference	Account this Network belongs to	Paco Reference to Account.	
vpc	Object<*VPC*>	VPC		

Base Schemas Deployable, Named, Title

VPC

VPC

Table 44: *VPC*

Field name	Type	Purpose	Constraints	Default
cidr	String	CIDR		
enable_dns_hostnames	Boolean	Enable DNS Hostnames		False
enable_dns_support	Boolean	Enable DNS Support		False
enable_internet_gateway	Boolean	Internet Gateway		False
nat_gateway	Container<*NATGateways*>	NAT Gateways		
peering	Container<*VPCPeerings*>	VPC Peering		
private_hosted_zone	Object<*PrivateHostedZone*>	Private hosted zone		
security_groups	Container<*SecurityGroupSets*>	Security Group Sets	Security Groups Sets are containers for SecurityGroups containers.	
segments	Container<*Segments*>	Segments		
vpn_gateway	Container<*VPNGateways*>	VPN Gateways		

Base Schemas Deployable, Named, Title

VPCPeerings

Container for *VPCPeering* objects.

Table 45: *VPCPeerings* Container<VPCPeering>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

VPCPeering

VPC Peering

Table 46: *VPCPeering*

Field name	Type	Purpose	Constraints	Default
network_environment	PacoReference	Network Environment Reference	Paco Reference to *NetworkEnvironment*.	
peer_account_id	String	Remote peer AWS account Id		
peer_region	String	Remote peer AWS region		
peer_role_name	String	Remote peer role name		
peer_vpcid	String	Remote peer VPC Id		
routing	List<*VPCPeeringRoute*>	Peering routes		

Base Schemas Deployable, Named, Title

VPCPeeringRoute

VPC Peering Route

Table 47: *VPCPeeringRoute*

Field name	Type	Purpose	Constraints	Default
cidr	String	CIDR IP	A valid CIDR v4 block or an empty string	
segment	PacoReference	Segment	Paco Reference to *Segment*.	

NATGateways

Container for *NATGateway* objects.

Table 48: *NATGateways* Container<NATGateway>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

NATGateway

NAT Gateway

Table 49: *NATGateway*

Field name	Type	Purpose	Constraints	Default
availability_zone	String	Availability Zones to launch instances in.	Can be 'all' or number of AZ: 1, 2, 3, 4 ...	all
default_route_segments	List<PacoReference>	Default Route Segments	Paco Reference to *Segment*.	
ec2_instance_type	String	EC2 Instance Type		t2.nano
ec2_key_pair	PacoReference	EC2 key pair	Paco Reference to EC2KeyPair.	
security_groups	List<PacoReference>	Security Groups	Paco Reference to *SecurityGroup*.	
segment	PacoReference	Segment	Paco Reference to *Segment*.	
type	String	NAT Gateway type		Managed

Base Schemas Deployable, Named, Title

VPNGateways

Container for *VPNGateway* objects.

Table 50: *VPNGateways* Container<VPNGateway>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

VPNGateway

VPN Gateway

Table 51: *VPNGateway*

Field name	Type	Purpose	Constraints	Default

Base Schemas Deployable, Named, Title

PrivateHostedZone

Private Hosted Zone

Table 52: *PrivateHostedZone*

Field name	Type	Purpose	Constraints	Default
name	String	Hosted zone name		
vpc_associations	List<String>	List of VPC Ids		

Base Schemas Deployable

Segments

Container for *Segment* objects.

Table 53: *Segments* Container

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

Segment

Segment

Table 54: *Segment*

Field name	Type	Purpose	Constraints	Default
az1_cidr	String	Availability Zone 1 CIDR		
az2_cidr	String	Availability Zone 2 CIDR		
az3_cidr	String	Availability Zone 3 CIDR		
az4_cidr	String	Availability Zone 4 CIDR		
az5_cidr	String	Availability Zone 5 CIDR		
az6_cidr	String	Availability Zone 6 CIDR		
internet_access	Boolean	Internet Access		False

Base Schemas Deployable, Named, Title

SecurityGroupSets

Container for *SecurityGroups* objects.

Table 55: *SecurityGroupSets* Container<SecurityGroups>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

SecurityGroups

Container for *SecurityGroup* objects.

Table 56: *SecurityGroups* Container<SecurityGroup>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

SecurityGroup

AWS Resource: Security Group

Table 57: *SecurityGroup*

Field name	Type	Purpose	Constraints	Default
egress	List<*EgressRule*>	Egress	Every list item must be an EgressRule	
group_description	String	Group description	Up to 255 characters in length	
group_name	String	Group name	Up to 255 characters in length. Cannot start with sg-.	
ingress	List<*IngressRule*>	Ingress	Every list item must be an IngressRule	

Base Schemas Deployable, Named, Title

EgressRule

Security group egress

Table 58: *EgressRule*

Field name	Type	Purpose	Constraints	Default
destination_security_group	PacoReference|String	Destination Security Group Reference	A Paco reference to a SecurityGroup Paco Reference to *SecurityGroup*. String Ok.	

Base Schemas SecurityGroupRule, Name

IngressRule

Security group ingress

Table 59: *IngressRule*

Field name	Type	Purpose	Constraints	Default
source_security_group	PacoReference|String	Source Security Group Reference	A Paco Reference to a SecurityGroup Paco Reference to *SecurityGroup*. String Ok.	

Base Schemas SecurityGroupRule, Name

6.16.2 Applications

Applications define a collection of AWS resources that work together to support a workload.

Applications specify the sets of AWS resources needed for an application workload. Applications contain a mandatory `groups:` field which is container of ResrouceGroup objects. Every AWS resource for an application must be contained in a ResourceGroup with a unique name, and every ResourceGroup has a Resources container where each Resource is given a unique name.

Attention: ResourceGroups and individual Resources both have an `order` field. When resources are created, they will be created based on the value of these `order` fields. First, the ResrouceGroup order is followed. The lowest order for a ResourceGroup will indicate that all those resources need to be created first, and then each Resource within a group will be created based on the order it is given.

In the example below, the `groups:` contain keys named `cicd`, `website` and `bastion`. In turn, each ResourceGroup contains `resources:` with names such as `cpbd`, `cert` and `alb`.

```
applications:
    my-paco-app:
        enabled: true
        groups:
            cicd:
                type: Deployment
                resources:
                    cpbd:
                        # CodePipeline CI/CD
                        type: DeploymentPipeline
                        # configuration goes here ...
            website:
                type: Application
                resources:
                    cert:
                        type: ACM
                        # configuration goes here ...
                    alb:
                        # Application Load Balancer (ALB)
                        type: LBApplication
                        # configuration goes here ...
                    webapp:
                        # AutoScalingGroup (ASG) of web server instances
```

(continues on next page)

(continued from previous page)

```
                    type: ASG
                    # configuration goes here ...
            bastion:
                type: Bastion
                resources:
                    instance:
                    # AutoScalingGroup (ASG) with only 1 instance (self-healing␣
↪ASG)
                    type: ASG
                    # configuration goes here ...
```

ApplicationEngines

A container for Application Engines

Table 60: *ApplicationEngines* Container<ApplicationEngine>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

Application

An Application is groups of cloud resources to support a workload.

Table 61: *Application*

Field name	Type	Purpose	Constraints	Default

Base Schemas DNSEnablable, Deployable, ApplicationEngine, Monitorable, Named, Notifiable, Title

ResourceGroups

A container of Application *ResourceGroup* objects.

Table 62: *ResourceGroups* Container<ResourceGroup>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ResourceGroup

A group of *Resources* to support an *Application*.

Table 63: *ResourceGroup*

Field name	Type	Purpose	Constraints	Default
dns_enabled	Boolean			
order	Int	The order in which the group will be deployed		
resources	Container<*Resources*>			
title	String	Title		
type	String	Type		

Base Schemas Deployable, Named

Resources

A container of Resources to support an *Application*.

Table 64: *Resources*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

6.16.3 Environments

Environments define where actual cloud resources are to be provisioned. As Environments copy all of the defaults from `network`, `applications`, `backups` and `secrets_manager` config in the same NetworkEnvironment file.

The top level `environments:` container is simply a name and a title. This defines logical names for each environment.

```
environments:

    dev:
        title: Development

    staging:
        title: Staging and QA

    prod:
        title: Production
```

Environments contain EnvironmentRegions. The name of an EnvironmentRegion must match a valid AWS region name. The special `default` name is also available, which can be used to override config for a whole environment, regardless of region.

The following example enables the applications named `marketing-app` and `sales-app` into all dev environments by default. In `us-west-2` this is overridden and only the `sales-app` would be deployed there.

```
environments:

    dev:
        title: Development
```

(continues on next page)

(continued from previous page)

```
        default:
            applications:
                marketing-app:
                    enabled: true
                sales-app:
                    enabled: true
        us-west-2:
            applications:
                marketing-app:
                    enabled: false
        ca-central-1:
            enabled: true
```

Environment

Environment

Table 65: *Environment*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

EnvironmentDefault

Default values for an Environment's configuration

Table 66: *EnvironmentDefault*

Field name	Type	Purpose	Constraints	Default
applications	Container<*ApplicationEngines*>	Application container		
network	Container<*Network*>	Network		
secrets_manager	Container<*SecretsManager*>	Secrets Manager		

Base Schemas RegionContainer, Named, Title

EnvironmentRegion

An actual provisioned Environment in a specific region. May contains overrides of the IEnvironmentDefault where needed.

Table 67: *EnvironmentRegion*

Field name	Type	Purpose	Constraints	Default

Base Schemas RegionContainer, *EnvironmentDefault*, Deployable, Named, Title

6.16.4 Secrets

SecretsManager

Secrets Manager contains *SecretManagerApplication* objects.

Table 68: *SecretsManager* Container<SecretsManagerApplication>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

SecretsManagerApplication

Container for *SecretsManagerGroup* objects.

Table 69: *SecretsManagerApplication* Container<SecretsManagerGroup>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

SecretsManagerGroup

Container for *SecretsManagerSecret* objects.

Table 70: *SecretsManagerGroup* Container<SecretsManagerSecret>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

SecretsManagerSecret

Secret for the Secrets Manager.

Table 71: *SecretsManagerSecret*

Field name	Type	Purpose	Constraints	Default
account	PacoReference	Account to provision the Secret in	Paco Reference to Account.	
generate_secret_string	Object<*GenerateSecretString*>	Generate SecretString object		

Base Schemas Deployable, Named, Title

GenerateSecretString

Table 72: *GenerateSecretString*

Field name	Type	Purpose	Constraints	Default
exclude_characters	String	A string that includes characters that should not be included in the generated password.		
exclude_lowercase	Boolean	The generated password should not include lowercase letters.		False
exclude_numbers	Boolean	The generated password should exclude digits.		False
exclude_punctuation	Boolean	The generated password should not include punctuation characters.		False
exclude_uppercase	Boolean	The generated password should not include uppercase letters.		False
generate_string_key	String	The JSON key name that's used to add the generated password to the JSON structure.		
include_space	Boolean	The generated password can include the space character.		
password_length	Int	The desired length of the generated password.		32
require_each_included_type	Boolean	The generated password must include at least one of every allowed character type.		True
secret_string_template	String	A properly structured JSON string that the generated password can be added to.		

Base Schemas Deployable

6.16.5 Backups

AWS Backup can be provisioned with the `backup_vaults:`. This is a container of BackupVaults. Each BackupVault can contain BackupPlans which are further composed of a BackupRules and BackupSelections.

```
backup_vaults:
  accounting:
    enabled: false
    plans:
      ebs_daily:
        title: EBS Daily Backups
        enabled: true
        plan_rules:
          - title: Backup EBS volumes once a day
            schedule_expression: cron(0 8 ? * * *)
            lifecycle_delete_after_days: 14
        selections:
          - title: EBS volumes tagged with "backup-accounting: daily"
            tags:
              - condition_type: STRINGEQUALS
```

(continues on next page)

(continued from previous page)

```
                condition_key: backup-accounting
                condition_value: daily
      database_weekly:
        title: Weekly MySQL Backups
        enabled: true
        plan_rules:
          - title: Rule for Weekly MySQL Backups
            schedule_expression: cron(0 10 ? * 1 *)
            lifecycle_delete_after_days: 150
        selections:
          - title: Database resource selection
            resources:
              - paco.ref netenv.mynet.applications.accounting.groups.app.resources.
↪database
```

BackupVaults must be explicity referenced in an environment for them to be provisioned.

```
environmnets:
  prod:
    ca-central-1:
      backup_vaults:
        accounting:
          enabled: true
```

BackupVaults

Container for *BackupVault* objects.

Table 73: *BackupVaults* Container<BackupVault>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

BackupVault

An AWS Backup Vault.

Table 74: *BackupVault*

Field name	Type	Purpose	Constraints	Default
notification_events	List<String>	Notification Events	Each notification event must be one of BACKUP_JOB_STARTED, BACKUP_JOB_COMPLETED, RESTORE_JOB_STARTED, RESTORE_JOB_COMPLETED, RECOVERY_POINT_MODIFIED	
notification_group	String	Notification Group		
plans	Container<*BackupPlans*>	Backup Plans		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

BackupPlans

Container for *BackupPlan* objects.

Table 75: *BackupPlans* Container<BackupPlan>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

BackupPlan

AWS Backup Plan

Table 76: *BackupPlan*

Field name	Type	Purpose	Constraints	Default
plan_rules	List<*BackupPlanRule*>	Backup Plan Rules		
selections	List<*BackupPlanSelection*>	Backup Plan Selections		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

BackupPlanRule

Table 77: *BackupPlanRule*

Field name	Type	Purpose	Constraints	Default
copy_actions	List<BackupPlanCopyActionResourceType_>	Copy actions		[]
lifecycle_delete_after_days	Int	Delete after days		
lifecycle_move_to_cold_storage_after_days	Int	Move to cold storage after days	If Delete after days value is set, this value must be smaller	
schedule_expression	String	Schedule Expression	Must be a valid Schedule Expression.	

Base Schemas Named, Title

BackupPlanSelection

Table 78: *BackupPlanSelection*

Field name	Type	Purpose	Constraints	Default
resources	List<PacoReference>	Backup Plan Resources	Paco Reference to Interface.	
tags	List<*BackupSelectionConditionResourceType*>	List of condition resource types		
title	String	Title		

BackupSelectionConditionResourceType

Table 79: *BackupSelectionConditionResourceType*

Field name	Type	Purpose	Constraints	Default
condition_key	String	Tag Key		
condition_type	String	Condition Type	String Condition operator must be one of: StringEquals, StringNotEquals, StringEqualsIgnoreCase, StringNotEqualsIgnoreCase, StringLike, StringNotLike.	
condition_value	String	Tag Value		

6.17 Application Resources

An Application is a collection of Resources. These are the Resources which can exist as part of an Application.

6.17.1 ApiGatewayRestApi

An ApiGateway Rest API resource.

Listing 11: API Gateway REST API example

```
type: ApiGatewayRestApi
order: 10
enabled: true
fail_on_warnings: true
description: "My REST API"
endpoint_configuration:
  - 'REGIONAL'
models:
  emptyjson:
    content_type: 'application/json'
cognito_authorizers:
  cognito:
    identity_source: 'Authorization'
    user_pools:
      - paco.ref netenv.mynet.applications.app.groups.cognito.resources.userpool
dns:
  - domain_name: api.example.com
    hosted_zone: paco.ref resource.route53.example_com
    ssl_certificate:  arn:aws:acm:us-east-1:*******:certificate/********
    base_path_mappings:
        - base_path: ''
          stage: 'prod'
methods:
  get:
    http_method: GET
    authorizer: cognito_authorizers.cognito
    integration:
      integration_type: AWS
      integration_lambda: paco.ref netenv.mynet.applications.app.groups.restapi.
↪resources.mylambda
      integration_responses:
        - status_code: '200'
          response_templates:
            'application/json': ''
      request_parameters:
        "integration.request.querystring.my_id": "method.request.querystring.my_id"
    authorization_type: NONE
    request_parameters:
      "method.request.querystring.my_id": false
      "method.request.querystring.token": false
    method_responses:
      - status_code: '200'
        response_models:
          - content_type: 'application/json'
            model_name: 'emptyjson'
  post:
```

(continues on next page)

(continued from previous page)

```
    http_method: POST
    integration:
      integration_type: AWS
      integration_lambda: paco.ref netenv.mynet.applications.app.groups.restapi.
↪resources.mylambda
      integration_responses:
        - status_code: '200'
          response_templates:
            'application/json': ''
    authorization_type: NONE
    method_responses:
      - status_code: '200'
        response_models:
          - content_type: 'application/json'
            model_name: 'emptyjson'
stages:
  prod:
    deployment_id: 'prod'
    description: 'Prod Stage'
    stage_name: 'prod'
```

Table 80: *ApiGatewayRestApi*

Field name	Type	Purpose	Constraints	Default
api_key_source_type	String	API Key Source Type	Must be one of 'HEADER' to read the API key from the X-API-Key header of a request or 'AUTHORIZER' to read the API key from the UsageIdentifierKey from a Lambda authorizer.	
binary_media_types	List<String>	Binary Media Types. The list of binary media types that are supported by the RestApi resource, such as image/png or application/octet-stream. By default, RestApi supports only UTF-8-encoded text payloads.	Duplicates are not allowed. Slashes must be escaped with ~1. For example, image/png would be image~1png in the BinaryMediaTypes list.	
body	String	Body. An OpenAPI specification that defines a set of RESTful APIs in JSON or YAML format. For YAML templates, you can also provide the specification in YAML format.	Must be valid JSON.	
body_file_location	StringFileReference	Path to a file containing the Body.	Must be valid path to a valid JSON document.	
body_s3_location	String	The Amazon Simple Storage Service (Amazon S3) location that points to an OpenAPI file, which defines a set of RESTful APIs in JSON or YAML format.	Valid S3Location string to a valid JSON or YAML document.	
clone_from	String	CloneFrom. The ID of the RestApi resource that you want to clone.		
cognito_authorizers	Container<*ApiGatewayCognitoAuthorizer*>	Authorizers		
description	String	Description of the RestApi resource.		
dns	List<*ApiGatewayDNS*>	DNS domains to create to resolve to the ApiGateway Endpoint		
endpoint_configuration	List<String>	Endpoint configuration. A list of the endpoint types of the API. Use this field when creating an API. When importing an existing API, specify the endpoint configuration types using the *parameters* field.	List of strings, each must be one of 'EDGE', 'REGIONAL', 'PRIVATE'	

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

ApiGatewayMethods

Container for *ApiGatewayMethod* objects.

Table 81: *ApiGatewayMethods* Container<ApiGatewayMethod>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ApiGatewayMethod

API Gateway Method

Table 82: *ApiGatewayMethod*

Field name	Type	Purpose	Constraints	Default
authorization_type	String	Authorization Type	Must be one of NONE, AWS_IAM, CUSTOM or COGNITO_USER_POOLS	NONE
authorizer	String	Authorizer	Must be tan authorizer type and authorizer name in this API Gateway, seperated by a . char. For example, 'cognito_authorizers.cognito'.	
http_method	String	HTTP Method	Must be one of ANY, DELETE, GET, HEAD, OPTIONS, PATCH, POST or PUT.	
integration	Object<*ApiGatewayMethodIntegration*>	Integration		
method_responses	List<*ApiGatewayMethodMethodResponse*>	Method Responses	List of ApiGatewayMethod MethodResponses	
request_parameters	Dict	Request Parameters	**Specify request parameters as key-value pairs** with a source as the key and a Boolean as the value. The Boolean specifies whether a parameter is required. A source must match the format method.request.location.name, where the location is querystring, path, or header, and	{}

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

ApiGatewayModels

Container for *ApiGatewayModel* objects.

Table 83: *ApiGatewayModels* Container<ApiGatewayModel>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ApiGatewayModel

Table 84: *ApiGatewayModel*

Field name	Type	Purpose	Constraints	Default
content_type	String	Content Type		
description	String	Description		
schema	Dict	Schema	JSON format. Will use null({}) if left empty.	{}

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

ApiGatewayResources

Container for *ApiGatewayResource* objects.

Table 85: *ApiGatewayResources* Container<ApiGatewayResource>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ApiGatewayResource

Table 86: *ApiGatewayResource* Container<**'unknown'**_>

Field name	Type	Purpose	Constraints	Default
child_resources	Container<*ApiGatewayResource*>	Child Api Gateway Resources		
enable_cors	Boolean	Enable CORS		False
path_part	String	Path Part		

Base Schemas Named, Title

ApiGatewayStages

Container for *ApiGatewayStage* objects

Table 87: *ApiGatewayStages* Container<ApiGatewayStages>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ApiGatewayStage

API Gateway Stage

Table 88: *ApiGatewayStage*

Field name	Type	Purpose	Constraints	Default
deployment_id	String	Deployment ID		
description	String	Description		
stage_name	String	Stage name		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

ApiGatewayMethodIntegration

Table 89: *ApiGatewayMethodIntegration*

Field name	Type	Purpose	Constraints	Default
integration_http_method	String	Integration HTTP Method	Must be one of ANY, DELETE, GET, HEAD, OPTIONS, PATCH, POST or PUT.	POST
integration_lambda	PacoReference	Integration Lambda	Paco Reference to *Lambda*.	
integration_responses	List<*ApiGatewayMethodIntegrationResponse*>	Integration Responses		
integration_type	String	Integration Type	Must be one of AWS, AWS_PROXY, HTTP, HTTP_PROXY or MOCK.	AWS
pass_through_behavior	Choice	Pass Through Behaviour		
request_parameters	Dict	The request parameters that API Gateway sends with the backend request.	Specify request parameters as key-value pairs (string-to-string mappings), with a destination as the key and a source as the value. Specify the destination by using the following pattern *integration.request.location.name*, where *location* is query string, path, or header, and *name* is a valid, unique parameter name. The source must be an existing method request parameter or a static value. You must enclose static values in single quotation marks and pre-encode these values based on their destination in the request.	{}
request_templates	Dict	Request Templates		{}

ApiGatewayMethodIntegrationResponse

Table 90: *ApiGatewayMethodIntegrationResponse*

Field name	Type	Purpose	Constraints	Default
content_handling	String	Specifies how to handle request payload content type conversions.	Valid values are: CONVERT_TO_BINARY: Converts a request payload from a base64-encoded string to a binary blob. CONVERT_TO_TEXT: Converts a request payload from a binary blob to a base64-encoded string. If this property isn't defined, the request payload is passed through from the method request to the integration request without modification.	
response_parameters	Dict	Response Parameters		{}
response_templates	Dict	Response Templates		{}
selection_pattern	String	A regular expression that specifies which error strings or status codes from the backend map to the integration response.		
status_code	String	The status code that API Gateway uses to map the integration response to a MethodResponse status code.	Must match a status code in the method_respones for this API Gateway REST API.	

ApiGatewayMethodMethodResponse

Table 91: *ApiGatewayMethodMethodResponse*

Field name	Type	Purpose	Constraints	Default
response_models	List<*ApiGatewayMethodMethodResponseModel*>	The resources used for the response's content type.	Specify response models as key-value pairs (string-to-string maps), with a content type as the key and a Model Paco name as the value.	
response_parameters	Dict	Response Parameters		{}
status_code	String	HTTP Status code		

ApiGatewayMethodMethodResponseModel

Table 92: *ApiGatewayMethodMethodResponseModel*

Field name	Type	Purpose	Constraints	Default
content_type	String	Content Type		
model_name	String	Model name		

ApiGatewayCognitoAuthorizers

Container for **'ApiGatewayAuthorizer'_** objects.

Table 93: *ApiGatewayCognitoAuthorizers* Container<**'ApiGatewayCognitoAuthorizer'_**>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ApiGatewayDNS

Table 94: *ApiGatewayDNS*

Field name	Type	Purpose	Constraints	Default
base_path_mappings	List<String>	Base Path Mappings		[]
ssl_certificate	PacoReference|String	SSL certificate Reference	Paco Reference to *ACM*. String Ok.	

Base Schemas DNS

6.17.2 ASG

An AutoScalingGroup (ASG) contains a collection of Amazon EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management.

The Paco ASG resource provisions an AutoScalingGroup as well as LaunchConfiguration and TargetGroups for that ASG.

Prescribed Automation

ASGs use Paco's **LaunchBundles**. A LaunchBundle is a zip file of code and configuration files that is automatically created and stored in an S3 Bucket that the ASG has read permissions to. Paco adds BASH code to the UserData script for the ASG's LaunchConfiguration that will iterate through all of the LaunchBundles and download and run them. For example, if you specify in-host metrics for an ASG, it will have a LaunchBundle created with the necessary CloudWatch agent configuration and a BASH script to install and configure the agent.

`launch_options`: Options to add actions to newly launched instances: `ssm_agent`, `update_packages` and `cfn_init_config_sets`. The `ssm_agent` field will install the SSM Agent and is true by default. Paco's **LaunchBundles** feature requires the SSM Agent installed and running. The `update_packages` field will perform a operating system package update (`yum update` or `apt-get update`). This happens immediately after the `user_data_pre_script` commands, but before the LaunchBundle commands and `user_data_script` commands. The `cfn_init_config_sets` field is a list of CfnInitConfigurationSets that will be run at launch.

`cfn_init`: Contains CloudFormationInit (cfn-init) configuration. Paco allows reading cfn-init files from the filesystem, and also does additional validation checks on the configuration to ensure it is correct. The `launch_options` has a `cfn_init_config_sets` field to specify which CfnInitConfigurationSets you want to automatically call during instance launch with a LaunchBundle.

`ebs_volume_mounts`: Adds an EBS LaunchBundle that mounts all EBS Volumes to the EC2 instance launched by the ASG. If the EBS Volume is unformatted, it will be formatted to the specified filesystem. **This feature only works with "self-healing" ASGs**. A "self-healing" ASG is an ASG with `max_instances` set to 1. Trying to launch a second instance in the ASG will fail to mount the EBS Volume as it can only be mounted to one instance at a time.

`eip`: Adds an EIP LaunchBundle which will attach an Elastic IP to a launched instance. **This feature only works with "self-healing" ASGs**. A "self-healing" ASG is an ASG with `max_instances` set to 1. Trying to launch a second instance in the ASG will fail to attach the EIP as it can only be mounted to one instance at a time.

`efs_mounts`: Adds an EFS LaunchBundle that mounts all EFS locations. A SecurityGroup must still be manually configured to allow the ASG instances to network access to the EFS filesystem.

`monitoring`: Any fields specified in the `metrics` or `log_sets` fields will add a CloudWatchAgent LaunchBundle that will install a CloudWatch Agent and configure it to collect all specified metrics and log sources.

`secrets`: Adds a policy to the Instance Role which allows instances to access the specified secrets.

`ssh_access`: Grants users and groups SSH access to the instances.

Listing 12: example ASG configuration

```
type: ASG
order: 30
enabled: true
associate_public_ip_address: false
cooldown_secs: 200
ebs_optimized: false
health_check_grace_period_secs: 240
```

(continues on next page)

(continued from previous page)

```
health_check_type: EC2
availability_zone: 1
ebs_volume_mounts:
  - volume: paco.ref netenv.mynet.applications.app.groups.storage.resources.my_volume
    enabled: true
    folder: /var/www/html
    device: /dev/xvdf
    filesystem: ext4
efs_mounts:
  - enabled: true
    folder: /mnt/wp_efs
    target: paco.ref netenv.mynet.applications.app.groups.storage.resources.my_efs
instance_iam_role:
  enabled: true
  policies:
    - name: DNSRecordSet
      statement:
        - effect: Allow
          action:
            - route53:ChangeResourceRecordSets
          resource:
            - 'arn:aws:route53:::hostedzone/HHIHkjhdhu744'
instance_ami: paco.ref function.aws.ec2.ami.latest.amazon-linux-2
instance_ami_type: amazon
instance_key_pair: paco.ref resource.ec2.keypairs.my_keypair
instance_monitoring: true
instance_type: t2.medium
desired_capacity: 1
max_instances: 3
min_instances: 1
rolling_update_policy:
  max_batch_size: 1
  min_instances_in_service: 1
  pause_time: PT3M
  wait_on_resource_signals: false
target_groups:
  - paco.ref netenv.mynet.applications.app.groups.web.resources.alb.target_groups.
↪cloud
security_groups:
  - paco.ref netenv.mynet.network.vpc.security_groups.web.asg
segment: private
termination_policies:
  - Default
scaling_policy_cpu_average: 60
ssh_access:
  users:
    - bdobbs
  groups:
    - developers
launch_options:
    update_packages: true
    ssm_agent: true
    cfn_init_config_sets:
    - "InstallApp"
cfn_init:
  config_sets:
    InstallApp:
```

(continues on next page)

(continued from previous page)

```
      - "InstallApp"
  configurations:
    InstallApp:
      packages:
        yum:
          python3: []
      users:
        www-data:
          uid: 2000
          home_dir: /home/www-data
      files:
        "/etc/systemd/system/pypiserver.service":
          content_file: ./pypi-config/pypiserver.service
          mode: '000755'
          owner: root
          group: root
      commands:
        00_pypiserver:
          command: "/bin/pip3 install pypiserver"
        01_passlib_dependency:
          command: "/bin/pip3 install passlib"
        02_prep_mount:
           command: "chown www-data:www-data /var/pypi"
      services:
        sysvinit:
          pypiserver:
            enabled: true
            ensure_running: true
monitoring:
  enabled: true
  collection_interval: 60
  metrics:
    - name: swap
      measurements:
        - used_percent
    - name: disk
      measurements:
        - free
      resources:
        - '/'
        - '/var/www/html'
      collection_interval: 300
user_data_script: |
  echo "Hello World!"
```

AutoScalingGroup Rolling Update Policy

When changes are applied to an AutoScalingGroup that modify the configuration of newly launched instances, AWS can automatically launch instances with the new configuration and terminate old instances that have stale configuration. This can be configured so that there is no interruption of service as the new instances gradually replace old ones. This configuration is set with the `rolling_update_policy` field.

The rolling update policy must be able to work within the minimum/maximum number of instances in the ASG. Consider the following ASG configuration.

Listing 13: example ASG configuration

```
type: ASG
max_instances: 2
min_instances: 1
desired_capacity: 1
rolling_update_policy:
  max_batch_size: 1
  min_instances_in_service: 1
  pause_time: PT0S # default setting
  wait_on_resource_signals: false # default setting
```

This will normally run a single instance in the ASG. The ASG is never allowed to launch more than 2 instances at one time. When an update happens, a new batch of instances is launched - in this example just one instance. There wil be only 1 instance in service, but the capacity will be at 2 instances will the new instance is launched. After the instance is put into service by the ASG, it will immediately terminate the old instance.

The `wait_on_resource_signals` can be set to tell AWS CloudFormation to wait on making changes to the AutoScalingGroup configuration until a new instance is finished configuring and installing applications and is ready for service. If this field is enabled, then the `pause_time` default is PT05 (5 minutes). If CloudFormation does not get a SUCCESS signal within the `pause_time` then it will mark the new instance as failed and terminate it.

If you use `pause_time` with the default `wait_on_resource_signals: false` then AWS will simply wait for the full duration of the pause time and then consider the instance ready. `pause_time` is in format PT#H#M#S, where each # is the number of hours, minutes, and seconds, respectively. The maximum `pause_time` is one hour. For example:

```
pause_time: PT0S # 0 seconds
pause_time: PT5M # 5 minutes
pause_time: PT2M30S # 2 minutes and 30 seconds
```

ASGs will use default settings for a rolling update policy. If you do not want to use an update policies at all, then you must disable the `rolling_update_policy` explicitly:

```
type: ASG
rolling_update_policy:
  enabled: false
```

With no rolling update policy, when you make configuration changes, then existing instances with old configuration will continue to run and instances with the new configuration will not happen until the AutoScalingGroup needs to launch new instances. You must be careful with this approach as you can not know 100% that your new configuration launches instances proprely until some point in the future when new instances are requested by the ASG.

Prescribed Automation

Paco can help you send signals to CloudFormation when using `wait_on_resource_signals`. If you set `wait_on_resource_signals: true` then Paco will automatically grant the needed `cloudformation:SignalResource` and `cloudformation:DescribeStacks` to the IAM Role associated with the instance for you. Paco also provides an `ec2lm_signal_asg_resource` BASH function available in your `user_data_script` that you can run to signal the instance is ready: `ec2lm_signal_asg_resource SUCCESS` or `ec2lm_signal_asg_resource SUCCESS`.

If you want to wait until load balancer health checks are passing before an instance is considered healthy, then send the SUCCESS signal to CloudFormation, you will need to configure this yourself.

Listing 14: example ASG signalling using ELB health checks

```
'until [ "$state" == ""InService"" ]; do state=$(aws --region $
↪{AWS::Region} elb describe-instance-health
--load-balancer-name ${ElasticLoadBalancer}
--instances $(curl -s http://169.254.169.254/latest/meta-data/instance-id)
--query InstanceStates[0].State); sleep 10; done'
```

See the AWS documentation for more information on how AutoScalingRollingUpdate Policy configuration is used.

Table 95: *ASG*

Field name	Type	Purpose	Constraints	Default
associate_public_ip_address	Boolean	Associate Public IP Address		False
availability_zone	String	Availability Zones to launch instances in.		all
block_device_mappings	List<*BlockDeviceMapping*>	Block Device Mappings		
cfn_init	Object<*CloudFormationInit*>	CloudFormation Init		
cooldown_secs	Int	Cooldown seconds		300
desired_capacity	Int	Desired capacity		1
desired_capacity_ignore_changes	Boolean	Ignore changes to the desired_capacity after the ASG is created.		False
dns	List<*DNS*>	DNS domains to create to resolve to one of the ASGs EC2 Instances		
ebs_optimized	Boolean	EBS Optimized		False
ebs_volume_mounts	List<*EBSVolumeMount*>	Elastic Block Store Volume Mounts		
ecs	Object<*ECSASGConfiguration*>	ECS Configuration		
efs_mounts	List<*EFSMount*>	Elastic Filesystem Configuration		
eip	PacoReference|String	Elastic IP or AllocationId to attach to instance at launch	Paco Reference to *EIP*. String Ok.	
health_check_grace_period_secs	Int	Health check grace period in seconds		300
health_check_type	String	Health check type	Must be one of: 'EC2', 'ELB'	EC2
instance_ami	PacoReference|String	Instance AMI	Paco Reference to Function. String Ok.	
instance_ami_ignore_changes	Boolean	Do not update the instance_ami after creation.		False
instance_ami_type	String	The AMI Operating System family	Must be one of amazon, centos, suse, debian, ubuntu, microsoft or redhat.	amazon
instance_iam_role	Object<Role>			

Continued on next page

Table 95 – continued from previous page

Field name	Type	Purpose	Constraints	Default
instance_key_pair	PacoReference	Key pair to connect to launched instances	Paco Reference to EC2KeyPair.	
instance_monitoring	Boolean	Instance monitoring		False
instance_type	String	Instance type		
launch_options	Object<*EC2LaunchOptions*>	EC2 Launch Options		
lifecycle_hooks	Container<*ASGLifecycleHooks*>	Lifecycle Hooks		
load_balancers	List<PacoReference>	Target groups	Paco Reference to *TargetGroup*.	
max_instances	Int	Maximum instances		2
min_instances	Int	Minimum instances		1
rolling_update_policy	Object<*ASGRollingUpdatePolicy*>	Rolling Update Policy		
scaling_policies	Container<*ASGScalingPolicies*>	Scaling Policies		
scaling_policy_cpu_avg	Int	Average CPU Scaling Polciy		0
secrets	List<PacoReference>	List of Secrets Manager References	Paco Reference to SecretsManagerSecret.	
security_groups	List<PacoReference>	Security groups	Paco Reference to SecurityGroup.	
segment	String	Segment		
ssh_access	Object<*SSHAccess*>	SSH Access		
target_groups	List<PacoReference>	Target groups	Paco Reference to *TargetGroup*.	
termination_policies	List<String>	Terminiation policies		
user_data_pre_script	String	User data pre-script		
user_data_script	String	User data script		

Base Schemas Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

ASGLifecycleHooks

Container for *ASGLifecycleHook* objects.

Table 96: *ASGLifecycleHooks* Container<ASGLifecycleHook>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ASGLifecycleHook

ASG Lifecycle Hook

Table 97: *ASGLifecycleHook*

Field name	Type	Purpose	Constraints	Default
default_result	String	Default Result		
lifecycle_transition	String	ASG Lifecycle Transition		
notification_target_arn	String	Lifecycle Notification Target Arn		
role_arn	String	Licecycel Publish Role ARN		

Base Schemas Deployable, Named, Title

ASGScalingPolicies

Container for *ASGScalingPolicy* objects.

Table 98: *ASGScalingPolicies* Container<ASGScalingPolicy>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ASGScalingPolicy

Auto Scaling Group Scaling Policy

Table 99: *ASGScalingPolicy*

Field name	Type	Purpose	Constraints	Default
adjustment_type	String	Adjustment Type		ChangeInCapacity
alarms	List<SimpleCloudWatchAlarm>	Alarms		
cooldown	Int	Scaling Cooldown in Seconds		300
policy_type	String	Policy Type		SimpleScaling
scaling_adjustment	Int	Scaling Adjustment		

Base Schemas Deployable, Named, Title

ASGRollingUpdatePolicy

AutoScalingRollingUpdate Policy

Table 100: *ASGRollingUpdatePolicy*

Field name	Type	Purpose	Constraints	Default
enabled	Boolean	Enable an UpdatePolicy for the ASG		True
max_batch_size	Int	Maximum batch size		1
min_instances_in_service	Int	Minimum instances in service		0
pause_time	String	Minimum instances in service	Must be in the format PT#H#M#S	
wait_on_resource_signals	Boolean	Wait for resource signals		False

Base Schemas Named, Title

ECSASGConfiguration

Table 101: *ECSASGConfiguration*

Field name	Type	Purpose	Constraints	Default
capacity_provider	Object<*ECSCapacityProvider*>	Capacity Provider		
cluster	PacoReference	Cluster	Paco Reference to *ECSCluster*.	
log_level	Choice	Log Level		error

Base Schemas Named, Title

ECSCapacityProvider

Table 102: *ECSCapacityProvider*

Field name	Type	Purpose	Constraints	Default
maximum_scaling_step_size	Int	Maximum Scaling Step Size		10000
minimum_scaling_step_size	Int	Minimum Scaling Step Size		1
target_capacity	Int	Target Capacity		100

Base Schemas Deployable, Named, Title

SSHAccess

Table 103: *SSHAccess*

Field name	Type	Purpose	Constraints	Default
groups	List<String>	Groups	Must match a group declared in resource/ec2.yaml	[]
users	List<String>	User	Must match a user declared in resource/ec2.yaml	[]

Base Schemas Named, Title

BlockDeviceMapping

Table 104: *BlockDeviceMapping*

Field name	Type	Purpose	Constraints	Default
device_name	String	The device name exposed to the EC2 instance		
ebs	Object<*BlockDevice*>	Amazon Ebs volume		
virtual_name	String	The name of the virtual device.	The name must be in the form ephemeralX where X is a number starting from zero (0), for example, ephemeral0.	

BlockDevice

Table 105: *BlockDevice*

Field name	Type	Purpose	Constraints	Default
delete_on_termination	Boolean	Indicates whether to delete the volume when the instance is terminated.		True
encrypted	Boolean	Specifies whether the EBS volume is encrypted.		
iops	Int	The number of I/O operations per second (IOPS) to provision for the volume.	The maximum ratio of IOPS to volume size (in GiB) is 50:1, so for 5,000 provisioned IOPS, you need at least 100 GiB storage on the volume.	
size_gib	Int	The volume size, in Gibibytes (GiB).	This can be a number from 1-1,024 for standard, 4-16,384 for io1, 1-16,384 for gp2, and 500-16,384 for st1 and sc1.	
snapshot_id	String	The snapshot ID of the volume to use.		
volume_type	String	The volume type, which can be standard for Magnetic, io1 for Provisioned IOPS SSD, gp2 for General Purpose SSD, st1 for Throughput Optimized HDD, or sc1 for Cold HDD.	Must be one of standard, io1, gp2, st1 or sc1.	

EBSVolumeMount

EBS Volume Mount Configuration

Table 106: *EBSVolumeMount*

Field name	Type	Purpose	Constraints	Default
device	String	Device to mount the EBS Volume with.		
filesystem	String	Filesystem to mount the EBS Volume with.		
folder	String	Folder to mount the EBS Volume		
volume	PacoReference|String	EBS Volume Resource Reference	Paco Reference to *EBS*. String Ok.	

Base Schemas Deployable

EFSMount

EFS Mount Folder and Target Configuration

Table 107: *EFSMount*

Field name	Type	Purpose	Constraints	Default
folder	String	Folder to mount the EFS target		
target	PacoReference|String	EFS Target Resource Reference	Paco Reference to *EFS*. String Ok.	

Base Schemas Deployable

EC2LaunchOptions

EC2 Launch Options

Table 108: *EC2LaunchOptions*

Field name	Type	Purpose	Constraints	Default
cfn_init_config_sets	List<String>	List of cfn-init config sets		[]
codedeploy_agent	Boolean	Install CodeDeploy Agent		False
ssm_agent	Boolean	Install SSM Agent		True
ssm_expire_events_after_days	String	Retention period of SSM logs		30
update_packages	Boolean	Update Distribution Packages		False

Base Schemas Named, Title

CloudFormationInit

CloudFormation Init is a method to configure an EC2 instance after it is launched. CloudFormation Init is a much more complete and robust method to install configuration files and pakcages than using a UserData script.

It stores information about packages, files, commands and more in CloudFormation metadata. It is accompanied by a `cfn-init` script which will run on the instance to fetch this configuration metadata and apply it. The whole system is often referred to simply as cfn-init after this script.

The `cfn_init` field of for an ASG contains all of the cfn-init configuration. After an instance is launched, it needs to run a local cfn-init script to pull the configuration from the CloudFromation stack and apply it. After cfn-init has applied configuration, you will run cfn-signal to tell CloudFormation the configuration was successfully applied. Use the `launch_options` field for an ASG to let Paco take care of all this for you.

Prescribed Automation

`launch_options`: The `cfn_init_config_sets:` field is a list of cfn-init configurations to apply at launch. This list will be applied in order. On Amazon Linux the cfn-init script is pre-installed in /opt/aws/bin. If you enable a cfn-init launch option, Paco will install cfn-init in /opt/paco/bin for you.

Refer to the CloudFormation Init docs for a complete description of all the configuration options available.

Listing 15: cfn_init with launch_options

```
launch_options:
    cfn_init_config_sets:
    - "Install"
cfn_init:
  parameters:
    BasicKey: static-string
    DatabasePasswordarn: paco.ref netenv.mynet.secrets_manager.app.site.database.arn
  config_sets:
    Install:
      - "Install"
  configurations:
    Install:
      packages:
        rpm:
          epel: "http://download.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.
↪noarch.rpm"
        yum:
          jq: []
          python3: []
      files:
        "/tmp/get_rds_dsn.sh":
          content_cfn_file: ./webapp/get_rds_dsn.sh
          mode: '000700'
          owner: root
          group: root
        "/etc/httpd/conf.d/saas_wsgi.conf":
          content_file: ./webapp/saas_wsgi.conf
          mode: '000600'
          owner: root
          group: root
        "/etc/httpd/conf.d/wsgi.conf":
          content: "LoadModule wsgi_module modules/mod_wsgi.so"
          mode: '000600'
          owner: root
          group: root
        "/tmp/install_codedeploy.sh":
          source: https://aws-codedeploy-us-west-2.s3.us-west-2.amazonaws.com/latest/
↪install
          mode: '000700'
          owner: root
          group: root
```

(continues on next page)

(continued from previous page)

```
      commands:
        10_install_codedeploy:
          command: "/tmp/install_codedeploy.sh auto > /var/log/cfn-init-codedeploy.
↪log 2>&1"
      services:
        sysvinit:
          codedeploy-agent:
            enabled: true
            ensure_running: true
```

The `parameters` field is a set of Parameters that will be passed to the CloudFormation stack. This can be static strings or `paco.ref` that are looked up from already provisioned cloud resources.

CloudFormation Init can be organized into Configsets. With raw cfn-init using Configsets is optional, but is required with Paco.

In a Configset, the `files` field has four fields for specifying the file contents.

- `content_file`: A path to a file on the local filesystem. A convenient practice is to make a sub-directory in the `netenv` directory for keeping cfn-init files.
- `content_cfn_file`: A path to a file on the local filesystem. This file will have FnSub and FnJoin CloudFormation applied to it.
- `content`: For small files, the content can be in-lined directly in this field.
- `source`: Fetches the file from a URL.

If you are using `content_cfn_file` to interpolate Parameters, the file might look like:

```
!Sub |
    #!/bin/bash

    echo "Database ARN is " ${DatabasePasswordarn}
    echo "AWS Region is " ${AWS::Region}
```

If you want to include a raw `${SomeValue}` string in your file, use the `!` character to escape it like this: `${!SomeValue}`. cfn-init also supports interpolation with Mustache templates, but Paco support for this is not yet implemented.

Table 109: *CloudFormationInit*

Field name	Type	Purpose	Constraints	Default
config_sets	Container<*CloudFormationConfigSets*>	CloudFormation Init configSets		
configurations	Container<*CloudFormationConfigSets*>	CloudFormation Init configurations		
parameters	Dict	Parameters		{}

Base Schemas Named, Title

CloudFormationConfigSets

Table 110: *CloudFormationConfigSets*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CloudFormationConfigurations

Table 111: *CloudFormationConfigurations* Container<CloudFormationConfiguration>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CloudFormationConfiguration

Table 112: *CloudFormationConfiguration*

Field name	Type	Purpose	Constraints	Default
commands	Container<*CloudFormationInitCommands*>	Commands		
files	Container<*CloudFormationInitFiles*>	Files		
groups	Object<*CloudFormationInitGroups*>	Groups		
packages	Object<*CloudFormationInitPackages*>	Packages		
services	Object<*CloudFormationInitServices*>	Services		
sources	Container<*CloudFormationInitSources*>	Sources		
users	Object<*CloudFormationInitUsers*>	Users		

Base Schemas Named, Title

CloudFormationInitCommands

Table 113: *CloudFormationInitCommands*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CloudFormationInitCommand

Table 114: *CloudFormationInitCommand*

Field name	Type	Purpose	Constraints	Default
command	String	Command		
cwd	String	Cwd. The working directory		
env	Dict	Environment Variables. This property overwrites, rather than appends, the existing environment.		{}
ignore_errors	Boolean	Ingore errors - determines whether cfn-init continues to run if the command in contained in the command key fails (returns a non-zero value). Set to true if you want cfn-init to continue running even if the command fails.		False
test	String	A test command that determines whether cfn-init runs commands that are specified in the command key. If the test passes, cfn-init runs the commands.		

CloudFormationInitFiles

Table 115: *CloudFormationInitFiles*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CloudFormationInitFile

Table 116: *CloudFormationInitFile*

Field name	Type	Purpose	Constraints	Default
authentication	String	The name of an authentication method to use.		
content	Object<Interface>	Either a string or a properly formatted YAML object.		
content_cfn_file	YAMLFileReference	File path to a properly formatted CloudFormation Functions YAML object.		
content_file	StringFileReference	File path to a string.		
context	String	Specifies a context for files that are to be processed as Mustache templates.		
encoding	String	The encoding format.		
group	String	The name of the owning group for this file. Not supported for Windows systems.		
mode	String	A six-digit octal value representing the mode for this file.		
owner	String	The name of the owning user for this file. Not supported for Windows systems.		
source	String	A URL to load the file from.		

Base Schemas Named, Title

CloudFormationInitGroups

Container for CloudFormationInit Groups

- –
  - –
  - –
  - –
  - –

CloudFormationInitPackages

Table 117: *CloudFormationInitPackages*

Field name	Type	Purpose	Constraints	Default
apt	Container<*CloudFormationInitVersionedPackageSet*>	Apt packages		
msi	Container<*CloudFormationInitPackageSet*>	MSI packages		
python	Container<*CloudFormationInitVersionedPackageSet*>	Apt packages		
rpm	Container<*CloudFormationInitPackageSet*>	RPM packages		
rubygems	Container<*CloudFormationInitVersionedPackageSet*>	Rubygems packages		
yum	Container<*CloudFormationInitVersionedPackageSet*>	Yum packages		

Base Schemas Named, Title

CloudFormationInitVersionedPackageSet

- –
  - –
  - –
  - –
  - –

CloudFormationInitPathOrUrlPackageSet

- –
  - –
  - –
  - –
  - –

CloudFormationInitServiceCollection

Table 118: *CloudFormationInitServiceCollection*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CloudFormationInitServices

Table 119: *CloudFormationInitServices*

Field name	Type	Purpose	Constraints	Default
sysvinit	Container<*CloudFormationInitSysVInitServices*>	SysVInit Services for Linux OS		
windows	Container<*CloudFormationInitWindowsServices*>	Windows Services for Windows OS		

Base Schemas Named, Title

CloudFormationInitService

Table 120: *CloudFormationInitService*

Field name	Type	Purpose	Constraints	Default
commands	List<String>	A list of command names. If cfn-init runs the specified command, this service will be restarted.		
enabled	Boolean	Ensure that the service will be started or not started upon boot.		
ensure_running	Boolean	Ensure that the service is running or stopped after cfn-init finishes.		
files	List<String>	A list of files. If cfn-init changes one directly via the files block, this service will be restarted		
packages	Dict	A map of package manager to list of package names. If cfn-init installs or updates one of these packages, this service will be restarted.		{}
sources	List<String>	A list of directories. If cfn-init expands an archive into one of these directories, this service will be restarted.		

CloudFormationInitSources

Table 121: *CloudFormationInitSources*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CloudFormationInitUsers

Container for CloudFormationInit Users

- –
  –
  –
  –
  –

6.17.3 ACM

Table 122: *ACM*

Field name	Type	Purpose	Constraints	Default
domain_name	String	Domain Name		
external_resource	Boolean	Marks this resource as external to avoid creating and validating it.		False
private_ca	String	Private Certificate Authority ARN		
region	String	AWS Region	Must be a valid AWS Region name	
subject_alternative_names	List<String>	Subject alternative names		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

6.17.4 CloudFront

CloudFront CDN Configuration

Table 123: *CloudFront*

Field name	Type	Purpose	Constraints	Default
cache_behaviors	List<*CloudFrontCacheBehaviors*>	List of Cache Behaviors		
custom_error_responses	List<*CloudFrontCustomErrorResponses*>	List of Custom Error Responses		
default_cache_behavior	Object<*CloudFrontDefaultCacheBehavior*>	Default Cache Behavior		
default_root_object	String	The default path to load from the origin.		
domain_aliases	List<*DNS*>	List of DNS for the Distribution		
factory	Container<**CloudFrontFactories_**>	CloudFront Factory		
origins	Container<**CloudFrontOrigins_**>	Map of Origins		
price_class	String	Price Class		All
viewer_certificate	Object<*CloudFrontViewerCertificate*>	Viewer Certificate		
webacl_id	String	WAF WebACLId		

Base Schemas Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

CloudFrontDefaultCacheBehavior

Table 124: *CloudFrontDefaultCacheBehavior*

Field name	Type	Purpose	Constraints	Default
allowed_methods	List<String>	List of Allowed HTTP Methods		['DELETE', 'GET', 'HEAD', 'OPTIONS', 'PATCH', 'POST', 'PUT']
cached_methods	List<String>	List of HTTP Methods to cache		['GET', 'HEAD', 'OPTIONS']
compress	Boolean	Compress certain files automatically		False
default_ttl	Int	Default TTL		86400
forwarded_values	Object<*CloudFrontForwardedValues*>	Forwarded Values		
lambda_function_associations	List<*CloudFrontLambdaFunctionAssociation*>	Lambda Function Associations		
max_ttl	Int	Maximum TTL		31536000
min_ttl	Int	Minimum TTL		0
target_origin	PacoReference	Target Origin	Paco Reference to *CloudFrontOrigin*.	
viewer_protocol_policy	String	Viewer Protocol Policy		redirect-to-https

Base Schemas Named, Title

CloudFrontCacheBehavior

Table 125: *CloudFrontCacheBehavior*

Field name	Type	Purpose	Constraints	Default
path_pattern	String	Path Pattern		

Base Schemas *CloudFrontDefaultCacheBehavior*, Named, Title

CloudFrontFactory

CloudFront Factory

Table 126: *CloudFrontFactory*

Field name	Type	Purpose	Constraints	Default
domain_aliases	List<*DNS*>	List of DNS for the Distribution		
viewer_certificate	Object<*CloudFrontViewerCertificate*>	Viewer Certificate		

Base Schemas Named, Title

CloudFrontOrigin

CloudFront Origin Configuration

Table 127: *CloudFrontOrigin*

Field name	Type	Purpose	Constraints	Default
custom_origin_config	Object<*CloudFrontCustomOriginConfig*>	Custom Origin Configuration		
domain_name	PacoReference|String	Origin Resource Reference	Paco Reference to Route53HostedZone. String Ok.	
s3_bucket	PacoReference	Origin S3 Bucket Reference	Paco Reference to *S3Bucket*.	

Base Schemas Named, Title

CloudFrontCustomOriginConfig

Table 128: *CloudFrontCustomOriginConfig*

Field name	Type	Purpose	Constraints	Default
http_port	Int	HTTP Port		
https_port	Int	HTTPS Port		
keepalive_timeout	Int	HTTP Keepalive Timeout		5
protocol_policy	String	Protocol Policy		
read_timeout	Int	Read timeout		30
ssl_protocols	List<String>	List of SSL Protocols		

Base Schemas Named, Title

CloudFrontCustomErrorResponse

Table 129: *CloudFrontCustomErrorResponse*

Field name	Type	Purpose	Constraints	Default
error_caching_min_ttl	Int	Error Caching Min TTL		300
error_code	Int	HTTP Error Code		
response_code	Int	HTTP Response Code		
response_page_path	String	Response Page Path		

CloudFrontViewerCertificate

Table 130: *CloudFrontViewerCertificate*

Field name	Type	Purpose	Constraints	Default
certificate	PacoReference	Certificate Reference	Paco Reference to *ACM*.	
minimum_protocol_version	String	Minimum SSL Protocol Version		TLSv1.1_2016
ssl_supported_method	String	SSL Supported Method		sni-only

Base Schemas Named, Title

CloudFrontForwardedValues

Table 131: *CloudFrontForwardedValues*

Field name	Type	Purpose	Constraints	Default
cookies	Object<*CloudFrontCookies*>	Forward Cookies		
headers	List<String>	Forward Headers		['*']
query_string	Boolean	Forward Query Strings		True

Base Schemas Named, Title

CloudFrontCookies

Table 132: *CloudFrontCookies*

Field name	Type	Purpose	Constraints	Default
forward	String	Cookies Forward Action		all
whitelisted_names	List<String>	White Listed Names		

Base Schemas Named, Title

CloudFrontLambdaFunctionAssocation

Table 133: *CloudFrontLambdaFunctionAssocation*

Field name	Type	Purpose	Constraints	Default
event_type	Choice	Event Type	Must be one of 'origin-request', 'origin-response', 'viewer-request' or 'viewer-response'	
include_body	Boolean	Include Body		False
lambda_function	PacoReference	Lambda Function	Paco Reference to *Lambda*.	

Base Schemas Named, Title

CognitoLambdaTriggers

Table 134: *CognitoLambdaTriggers*

Field name	Type	Purpose	Constraints	Default
create_auth_challenge	PacoReference	CreateAuthChallenge Lambda trigger	Paco Reference to *Lambda*.	
custom_message	PacoReference	CustomMessage Lambda trigger	Paco Reference to *Lambda*.	
define_auth_challenge	PacoReference	DefineAuthChallenge Lambda trigger	Paco Reference to *Lambda*.	
post_authentication	PacoReference	PostAuthentication Lambda trigger	Paco Reference to *Lambda*.	
post_confirmation	PacoReference	PostConfirmation Lambda trigger	Paco Reference to *Lambda*.	
pre_authentication	PacoReference	PreAuthentication Lambda trigger	Paco Reference to *Lambda*.	
pre_sign_up	PacoReference	PreSignUp Lambda trigger	Paco Reference to *Lambda*.	
pre_token_generation	PacoReference	PreTokenGeneration Lambda trigger	Paco Reference to *Lambda*.	
user_migration	PacoReference	UserMigration Lambda trigger	Paco Reference to *Lambda*.	
verify_auth_challenge_response	PacoReference	VerifyAuthChallengeResponse Lambda trigger	Paco Reference to *Lambda*.	

6.17.5 CodeDeployApplication

CodeDeploy Application creates CodeDeploy Application and Deployment Groups for that application.

This resource can be used when you already have another process in-place to put deploy artifacts into an S3 Bucket. If you also need to build artifacts, use *DeploymentPipeline* instead.

Prescribed Automation

CodeDeploy Service Role: The AWS CodeDeploy service needs a Service Role that it is allowed to assume to allow the service to run in your AWS Account. Paco will automatically create such a service role for every CodeDeploy Application.

Listing 16: Example CodeDeployApplication resource YAML

```
type: CodeDeployApplication
order: 40
compute_platform: "Server"
deployment_groups:
  deployment:
    title: "My Deployment Group description"
    ignore_application_stop_failures: true
    revision_location_s3: paco.ref netenv.mynet.applications.app.groups.deploybucket
    autoscalinggroups:
      - paco.ref netenv.mynet.applications.app.groups.web
```

It can be convienent to install the CodeDeploy agent on your instances using CloudFormationInit.

Listing 17: Example ASG configuration for cfn_init to install CodeDeploy agent

```
launch_options:
  cfn_init_config_sets:
    - "InstallCodeDeploy"
cfn_init:
  config_sets:
    InstallCodeDeploy:
      - "InstallCodeDeploy"
  files:
    "/tmp/install_codedeploy.sh":
      source: https://aws-codedeploy-us-west-2.s3.us-west-2.amazonaws.com/latest/
↪install
      mode: '000700'
      owner: root
      group: root
  commands:
    01_install_codedeploy:
      command: "/tmp/install_codedeploy.sh auto > /var/log/cfn-init-codedeploy.log 2>&
↪1"
  services:
    sysvinit:
      codedeploy-agent:
        enabled: true
        ensure_running: true
```

Table 135: *CodeDeployApplication*

Field name	Type	Purpose	Constraints	Default
compute_platform	String	Compute Platform	Must be one of Lambda, Server or ECS	
deployment_groups	Container<*CodeDeployDeploymentGroups*>	CodeDeploy Deployment Groups		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

CodeDeployDeploymentGroups

Table 136: *CodeDeployDeploymentGroups*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CodeDeployDeploymentGroup

Table 137: *CodeDeployDeploymentGroup*

Field name	Type	Purpose	Constraints	Default
autoscalinggroups	List<PacoReference>	AutoScalingGroups that CodeDeploy automatically deploys revisions to when new instances are created	Paco Reference to *ASG*.	
ignore_application_stop_failures	Boolean	Ignore Application Stop Failures		
revision_location	Object<*DeploymentGroupS3Location*>	S3 Bucket revision location		
role_policies	List<Policy>	Policies to grant the deployment group role		

Base Schemas Deployable, Named, Title

6.17.6 CognitoUserPool

Amazon Cognito lets you add user sign-up, sign-in, and access control to your web and mobile apps.

The `CognitoUserPool` resource type is a user directory in Amazon Cognito. With a user pool, users can sign in to your web or mobile app through Amazon Cognito.

Prescribed Automation

`mfa`: If this is `on` or `optional` then an IAM Role will be created to allow sending SMS reset codes. If you are supporting SMS with Cognito, then you will also need to manually create an AWS Support ticket to raise the accounts limit of SMS spending beyond the default of $1/month.

Listing 18: Example CognituUserPool YAML

```
type: CognitoUserPool
order: 10
enabled: true
auto_verified_attributes: email
mfa: 'optional'
mfa_methods:
 - software_token
 - sms
account_recovery: verified_email
password:
  minimum_length: 12
  require_lowercase: true
  require_uppercase: true
  require_numbers: false
  require_symbols: false
email:
  reply_to_address: reply-to@example.com
user_creation:
  admin_only: true
  unused_account_validity_in_days: 7
  invite_message_templates:
```

(continues on next page)

(continued from previous page)

```
    email_subject: 'Invite to the App!'
    email_message: >
      <p>You've had an account created for you on the app.</p>
      <p><b>Username:</b> {username}</p>
      <p><b>Temporary password:</b> {####}</p>
      <p>Please login and set a secure password. This request will expire in 7 days.</
↪p>
lambda_triggers:
  pre_sign_up: paco.ref netenv.mynet.applications.app.groups.serverless.resources.
↪mylambda
schema:
  - attribute_name: email
    attribute_data_type: string
    mutable: false
    required: true
  - attribute_name: name
    attribute_data_type: string
    mutable: true
    required: true
  - attribute_name: phone_number
    attribute_data_type: string
    mutable: true
    required: false
ui_customizations:
  logo_file: './images/logo.png'
  css_file: './images/cognito.css'
app_clients:
  web:
    generate_secret: false
    callback_urls:
      - https://example.com
      - https://example.com/parseauth
      - https://example.com/refreshauth
    logout_urls:
      - https://example.com/signout
    allowed_oauth_flows:
        - code
    allowed_oauth_scopes:
        - email
        - openid
    domain_name: exampledomain
    identity_providers:
      - cognito
```

Table 138: *CognitoUserPool*

Field name	Type	Purpose	Constraints	Default
account_recovery	String	Account Recovery Options (in order of priority)	Can be either 'admin_only', 'verified_email', 'verified_phone_number', 'verified_phone_number,verified_email' or 'verified_email,verified_phone_number'	
app_clients	Container<*CognitoUserPoolClients*>	App Clients		
auto_verified_attributes	String	Auto Verified Attributes	Can be either 'email', 'phone_number' or 'email,phone_number'	
email	Object<*CognitoEmailConfiguration*>	Email Configuration		
lambda_triggers	Object<*CognitoLambdaTriggers*>	Lambda Triggers		
mfa	Choice	MFA Configuration	Must be one of 'off', 'on' or 'optional'	off
mfa_methods	Choice	Enabled MFA methods	List of 'sms' or 'software_token'	[]
password	Object<*CognitoUserPoolPasswordPolicy*>	Password Configuration		
schema	List<*CognitoUserPoolSchemaAttribute*>	Schema Attributes		[]
ui_customizations	Object<*CognitoUICustomizations*>	UI Customizations		
user_creation	Object<*CognitoUserCreation*>	User Creation		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

CognitoInviteMessageTemplates

Table 139: *CognitoInviteMessageTemplates*

Field name	Type	Purpose	Constraints	Default
email_message	String	Email Message		
email_subject	String	Email Subject		
sms_message	String	SMS Message		

Base Schemas Named, Title

CognitoUserPoolClients

A container of *CognitoUserPoolClient* objects.

Table 140: *CognitoUserPoolClients* Container<CognitoUserPoolClient>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CognitoUserPoolClient

Table 141: *CognitoUserPoolClient*

Field name	Type	Purpose	Constraints	Default
allowed_oauth_flows	Choice	Allowed OAuth Flows		[]
allowed_oauth_scopes	List<String>	Allow OAuth Scopes		[]
callback_urls	List<String>	Callback URLs		[]
domain_name	String	Domain Name or domain prefix		
generate_secret	Boolean	Generate Secret		False
identity_providers	Choice	Identity Providers		[]
logout_urls	List<String>	Logout URLs		[]

Base Schemas Named, Title

CognitoEmailConfiguration

Table 142: *CognitoEmailConfiguration*

Field name	Type	Purpose	Constraints	Default
from_address	String	From Email Address		
reply_to_address	String	Reply To Email Address		
verification_message	String	Verification Message		
verification_subject	String	Verification Subject		

Base Schemas Named, Title

CognitoUserPoolPasswordPolicy

Table 143: *CognitoUserPoolPasswordPolicy*

Field name	Type	Purpose	Constraints	Default
minimum_length	Int	Minimum Length		
require_lowercase	Boolean	Require Lowercase		True
require_numbers	Boolean	Require Numbers		True
require_symbols	Boolean	Require Symbols		True
require_uppercase	Boolean	Require Uppercase		True

Base Schemas Named, Title

CognitoUserPoolSchemaAttribute

Table 144: *CognitoUserPoolSchemaAttribute*

Field name	Type	Purpose	Constraints	Default
attribute_data_type	Choice	Attribute Data Type		
attribute_name	String	Name	From 1 to 20 characters	
mutable	Boolean	Mutable		
required	Boolean	Required		

CognitoUICustomizations

Table 145: *CognitoUICustomizations*

Field name	Type	Purpose	Constraints	Default
css_file	StringFileReference	File path to a CSS file.	Contents must be valid CSS that applies to the Cognito Hosted UI.	
logo_file	BinaryFileReference	File path to an image.	Must be a PNG or JPEG and max 100 Kb.	

Base Schemas Named, Title

CognitoUserCreation

Table 146: *CognitoUserCreation*

Field name	Type	Purpose	Constraints	Default
admin_only	Boolean	Allow only Admin to create users		False
invite_message_templates	Object: *CognitoInviteMessageTemplate*	Invite Message Templates		
unused_account_validity_in_days	Integer	Unused Account Validity in Days		7

Base Schemas Named, Title

6.17.7 CognitoIdentityPool

The `CognitoIdentityPool` resource type grants authorization of Cognito User Pool users to resources.

Listing 19: Example CognituIdentityPool YAML

```
type: CognitoIdentityPool
order: 20
enabled: true
allow_unauthenticated_identities: true
identity_providers:
```

(continues on next page)

(continued from previous page)

```
  - userpool_client: paco.ref netenv.mynet.applications.myapp.groups.cognito.resources.
↪cup.app_clients.web
    serverside_token_check: false
unauthenticated_role:
  enabled: true
  policies:
    - name: CognitoSyncAll
      statement:
        - effect: Allow
          action:
            - "cognito-sync:*"
          resource:
            - '*'
authenticated_role:
  enabled: true
  policies:
    - name: ViewDescribe
      statement:
        - effect: Allow
          action:
            - "cognito-sync:*"
            - "cognito-identity:*"
          resource:
            - '*'
        - effect: Allow
          action:
            - "lambda:InvokeFunction"
          resource:
            - '*'
```

Table 147: *CognitoIdentityPool*

Field name	Type	Purpose	Constraints	Default
allow_unauthenticated_identities	Boolean	Allow Unauthenticated Identities		False
authenticated_role	Object<RoleDefaultEnabled>			
identity_providers	List<*CognitoIdentityProvider*>	Identity Providers		[]
unauthenticated_role	Object<RoleDefaultEnabled>			

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

CognitoIdentityProvider

Table 148: *CognitoIdentityProvider*

Field name	Type	Purpose	Constraints	Default
serverside_token_check	Boolean	ServerSide Token Check		False
userpool_client	PacoReference	Identity Provider	Paco Reference to *CognitoUserPoolClient*.	

6.17.8 DeploymentPipeline

DeploymentPipeline creates AWS CodePipeline resources configured to act as CI/CDs to deploy code and assets to application resources. DeploymentPipelines allow you to express complex CI/CDs with minimal configuration.

A DeploymentPipeline has a number of Actions for three pre-defined Stages: source, build and deploy. The currently supported list of actions for each stage is:

Listing 20: Current Actions available by Stage

```
source:
  type: CodeCommit.Source
  type: ECR.Source
  type: GitHub.Source
build:
  type: CodeBuild.Build
deploy:
  type: CodeDeploy.Deploy
  type: ECS.Deploy
  type: ManualApproval
```

DeploymentPipelines can be configured to work cross-account and will automatically encrypt the artifacts S3 Bucket with a KMS-CMK key that can only be accessed by the pipeline. The `configuration` field lets you set the account that the DeploymentPipeline's CodePipeilne resource will be created in and also specify the S3 Bucket to use for artifacts.

Listing 21: Configure a DeploymentPipeline to run in the tools account

```
configuration:
  artifacts_bucket: paco.ref netenv.mynet.applications.myapp.groups.cicd.resources.
↪artifacts
  account: paco.ref accounts.tools
```

DeploymentPipeline caveats - there are a few things to consider when creating pipelines:

- You need to create an S3 Bucket that will be configured to for artifacts. Even pipelines which don't create artifacts will need this resource to hold ephemeral files created by CodePipeline.
- A pipeline that deploys artifacts to an AutoScalingGroup will need the `artifacts_bucket` to allow the IAM Instance Role to read from the bucket.
- A pipeline with an `ECR.Source` source must be in the same account as the ECR Repository.
- A pipeline with an `ECR.Source` source must have at least one image alreaay created in it before it can be created.
- A pipeline that is building Docker images needs to set `privileged_mode: true`.
- If you are using a manual approval step before deploying, pay attention to the `run_order` field. Normally you will want the approval action to happen before the deploy action.

Listing 22: Example S3 Bucket for a DeploymentPipeline that deploys to an AutoScalingGroup

```
type: S3Bucket
enabled: true
order: 10
bucket_name: "artifacts"
deletion_policy: "delete"
```

(continues on next page)

(continued from previous page)

```
account: paco.ref accounts.tools
versioning: true
policy:
  - aws:
      - paco.sub '${paco.ref netenv.mynet.applications.myapp.groups.container.
↪resources.asg.instance_iam_role.arn}'
    effect: 'Allow'
    action:
      - 's3:Get*'
      - 's3:List*'
    resource_suffix:
      - '/*'
      - ''
```

Listing 23: Example DeploymentPipeline to deploy to ECS when an ECR Repository is updated

```
type: DeploymentPipeline
order: 10
enabled: true
configuration:
  artifacts_bucket: paco.ref netenv.mynet.applications.myapp.groups.cicd.resources.
↪artifacts
  account: paco.ref accounts.tools
source:
  ecr:
    type: ECR.Source
    repository: paco.ref netenv.mynet.applications.myapp.groups.container.resources.
↪ecr_example
    image_tag: latest
deploy:
  ecs:
    type: ECS.Deploy
    cluster: paco.ref netenv.mynet.applications.myapp.groups.container.resources.ecs_
↪cluster
    service: paco.ref netenv.mynet.applications.myapp.groups.container.resources.ecs_
↪config.services.simple_app
```

Listing 24: Example DeploymentPipeline to pull from GitHub, build a Docker image and then deploy from an ECR Repo

```
type: DeploymentPipeline
order: 20
enabled: true
configuration:
  artifacts_bucket: paco.ref netenv.mynet.applications.myapp.groups.cicd.resources.
↪artifacts
  account: paco.ref accounts.tools
source:
  github:
    type: GitHub.Source
    deployment_branch_name: "prod"
    github_access_token: paco.ref netenv.mynet.secrets_manager.myapp.github.token
    github_owner: MyExample
    github_repository: MyExample-FrontEnd
    poll_for_source_changes: false
```

(continues on next page)

(continued from previous page)

```
build:
  codebuild:
    type: CodeBuild.Build
    deployment_environment: "prod"
    codebuild_image: 'aws/codebuild/standard:4.0'
    codebuild_compute_type: BUILD_GENERAL1_MEDIUM
    privileged_mode: true # To allow docker images to be built
    codecommit_repo_users:
      - paco.ref resource.codecommit.mygroup.myrepo.users.MyCodeCommitUser
    secrets:
      - paco.ref netenv.mynet.secrets_manager.myapp.github.ssh_private_key
    role_policies:
      - name: AmazonEC2ContainerRegistryPowerUser
        statement:
          - effect: Allow
            action:
              - ecr:GetAuthorizationToken
              - ecr:BatchCheckLayerAvailability
              - ecr:GetDownloadUrlForLayer
              - ecr:GetRepositoryPolicy
              - ecr:DescribeRepositories
              - ecr:ListImages
              - ecr:DescribeImages
              - ecr:BatchGetImage
              - ecr:GetLifecyclePolicy
              - ecr:GetLifecyclePolicyPreview
              - ecr:ListTagsForResource
              - ecr:DescribeImageScanFindings
              - ecr:InitiateLayerUpload
              - ecr:UploadLayerPart
              - ecr:CompleteLayerUpload
              - ecr:PutImage
            resource:
              - '*'
deploy:
  ecs:
    type: ECS.Deploy
    cluster: paco.ref netenv.mynet.applications.myapp.groups.container.resources.
↪cluster
    service: paco.ref netenv.mynet.applications.myapp.groups.container.resources.
↪services.services.frontend
```

Listing 25: Example DeploymentPipeline to pull from CodeCommit, build an app artifact and then deploy to an ASG using CodeDeploy

```
type: DeploymentPipeline
order: 30
enabled: true
configuration:
  artifacts_bucket: paco.ref netenv.mynet.applications.myapp.groups.cicd.resources.
↪artifacts
  account: paco.ref accounts.tools
source:
  codecommit:
    type: CodeCommit.Source
    codecommit_repository: paco.ref resource.codecommit.mygroup.myrepo
```

(continues on next page)

(continued from previous page)

```
    deployment_branch_name: "prod"
build:
  codebuild:
    type: CodeBuild.Build
    deployment_environment: "prod"
    codebuild_image: 'aws/codebuild/amazonlinux2-x86_64-standard:1.0'
    codebuild_compute_type: BUILD_GENERAL1_SMALL
deploy:
  approval:
    type: ManualApproval
    run_order: 1
    manual_approval_notification_email:
      - bob@example.com
      - sally@example.com
  codedeploy:
    type: CodeDeploy.Deploy
    run_order: 2
    alb_target_group: paco.ref netenv.mynet.applications.myapp.groups.backend.
↪resources.alb.target_groups.api
    auto_scaling_group: paco.ref netenv.mynet.applications.myapp.groups.backend.
↪resources.api
    auto_rollback_enabled: true
    minimum_healthy_hosts:
      type: HOST_COUNT
      value: 0
    deploy_style_option: WITHOUT_TRAFFIC_CONTROL
```

Table 149: *DeploymentPipeline*

Field name	Type	Purpose	Constraints	Default
build	Container<*DeploymentPipelineBuildStage*>	Deployment Pipeline Build Stage		
configuration	Object<*DeploymentPipelineConfiguration*>	Deployment Pipeline General Configuration		
deploy	Container<*DeploymentPipelineDeployStage*>	Deployment Pipeline Deploy Stage		
source	Container<*DeploymentPipelineSourceStage*>	Deployment Pipeline Source Stage		
stages	Container<*CodePipelineStages*>	Stages		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

CodePipelineStages

Container for *CodePipelineStage* objects.

Table 150: *CodePipelineStages* Container<CodePipelineStage>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CodePipelineStage

Container for different types of DeploymentPipelineStageAction objects.

Table 151: *CodePipelineStage*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

DeploymentPipelineSourceStage

A map of DeploymentPipeline source stage actions

Table 152: *DeploymentPipelineSourceStage*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

DeploymentPipelineDeployStage

A map of DeploymentPipeline deploy stage actions

Table 153: *DeploymentPipelineDeployStage*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

DeploymentPipelineBuildStage

A map of DeploymentPipeline build stage actions

Table 154: *DeploymentPipelineBuildStage*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

DeploymentPipelineDeployCodeDeploy

CodeDeploy DeploymentPipeline Deploy Stage

Table 155: *DeploymentPipelineDeployCodeDeploy*

Field name	Type	Purpose	Constraints	Default
alb_target_group	PacoReference	ALB Target Group Reference	Paco Reference to *TargetGroup*.	
auto_rollback_enabled	Boolean	Automatic rollback enabled		True
auto_scaling_group	PacoReference	ASG Reference	Paco Reference to *ASG*.	
deploy_instance_role	PacoReference	Deploy Instance Role Reference	Paco Reference to Role.	
deploy_style_option	String	Deploy Style Option		WITH_TRAFFIC_CONTRO
elb_name	String	ELB Name		
minimum_healthy_hosts	Object<*CodeDeployMinimumHealthyHosts*>	The minimum number of healthy instances that should be available at any time during the deployment.		

Base Schemas Enablable, Named, *DeploymentPipelineStageAction*, Title

DeploymentPipelineSourceECR

Amazon ECR DeploymentPipeline Source Stage

This Action is triggered whenever a new image is pushed to an Amazon ECR repository.

```
pipeline:
  type: DeploymentPipeline
  stages:
    source:
      ecr:
        type: ECR.Source
        enabled: true
        repository:  paco.ref netenv.mynet.applications.myapp.groups.ecr.resources.
↪myecr
        image_tag: "latest"
```

Table 156: *DeploymentPipelineSourceECR*

Field name	Type	Purpose	Constraints	Default
image_tag	String	The name of the tag used for the image.		latest
repository	PacoReference|String	An ECRRepository ref or the name of the an ECR repository.	Paco Reference to *ECRRepository*. String Ok.	

Base Schemas Enablable, Named, *DeploymentPipelineStageAction*, Title

CodeDeployMinimumHealthyHosts

CodeDeploy Minimum Healthy Hosts

Table 157: *CodeDeployMinimumHealthyHosts*

Field name	Type	Purpose	Constraints	Default
type	String	Deploy Config Type		HOST_COUNT
value	Int	Deploy Config Value		0

Base Schemas Named, Title

DeploymentPipelineManualApproval

ManualApproval DeploymentPipeline

Table 158: *DeploymentPipelineManualApproval*

Field name	Type	Purpose	Constraints	Default
manual_approval_notification_email	List<String>	Manual Approval Notification Email List		

Base Schemas Enablable, Named, *DeploymentPipelineStageAction*, Title

DeploymentPipelineDeployS3

Amazon S3 Deployment Provider

Table 159: *DeploymentPipelineDeployS3*

Field name	Type	Purpose	Constraints	Default
bucket	PacoReference	S3 Bucket Reference	Paco Reference to *S3Bucket*.	
extract	Boolean	Boolean indicating whether the deployment artifact will be unarchived.		True
input_artifacts	List<String>	Input Artifacts		
object_key	String	S3 object key to store the deployment artifact as.		

Base Schemas Enablable, Named, *DeploymentPipelineStageAction*, Title

DeploymentPipelineBuildCodeBuild

CodeBuild DeploymentPipeline Build Stage

Table 160: *DeploymentPipelineBuildCodeBuild*

Field name	Type	Purpose	Constraints	Default
buildspec	String	buildspec.yml filename		
codebuild_compute_type	String	CodeBuild Compute Type		
codebuild_image	String	CodeBuild Docker Image		
codecommit_repo_users	List<PacoReference>	CodeCommit Users	Paco Reference to CodeCommitUser.	
deployment_environment	String	Deployment Environment		
ecr_repositories	List<*ECRRepositoryPermission*>	ECR Respository Permissions		[]
privileged_mode	Boolean	Privileged Mode		False
role_policies	List<Policy>	Project IAM Role Policies		
secrets	List<PacoReference>	List of PacoReferences to Secrets Manager secrets	Paco Reference to SecretsManagerSecret.	
timeout_mins	Int	Timeout in Minutes		60

Base Schemas Enablable, Named, *DeploymentPipelineStageAction*, Title

DeploymentPipelineSourceCodeCommit

CodeCommit DeploymentPipeline Source Stage

Table 161: *DeploymentPipelineSourceCodeCommit*

Field name	Type	Purpose	Constraints	Default
codecommit_repository	PacoReference	CodeCommit Respository	Paco Reference to CodeCommitRepository.	
deployment_branch_name	String	Deployment Branch Name		

Base Schemas Enablable, Named, *DeploymentPipelineStageAction*, Title

DeploymentPipelineStageAction

Deployment Pipeline Source Stage

Table 162: *DeploymentPipelineStageAction*

Field name	Type	Purpose	Constraints	Default
run_order	Int	The order in which to run this stage		1
type	String	The type of DeploymentPipeline Source Stage		

Base Schemas Enablable, Named, Title

DeploymentPipelineConfiguration

Deployment Pipeline General Configuration

Table 163: *DeploymentPipelineConfiguration*

Field name	Type	Purpose	Constraints	Default
account	PacoReference	The account where Pipeline tools will be provisioned.	Paco Reference to Account.	
artifacts_bucket	PacoReference	Artifacts S3 Bucket Reference	Paco Reference to *S3Bucket*.	

Base Schemas Named, Title

DeploymentGroupS3Location

Table 164: *DeploymentGroupS3Location*

Field name	Type	Purpose	Constraints	Default
bucket	PacoReference	S3 Bucket revision location	Paco Reference to *S3Bucket*.	
bundle_type	String	Bundle Type	Must be one of JSON, tar, tgz, YAML or zip.	
key	String	The name of the Amazon S3 object that represents the bundled artifacts for the application revision.		

6.17.9 EBS

Elastic Block Store (EBS) Volume.

It is required to specify the `availability_zone` the EBS Volume will be created in. If the volume is going to be used by an ASG, it should launch an instance in the same `availability_zone` (and region).

Listing 26: Example EBS resource YAML

```
type: EBS
order: 5
enabled: true
size_gib: 4
volume_type: gp2
availability_zone: 1
```

Table 165: *EBS*

Field name	Type	Purpose	Constraints	Default
availability_zone	Int	Availability Zone to create Volume in.		
size_gib	Int	Volume Size in GiB		10
snapshot_id	String	Snapshot ID		
volume_type	String	Volume Type	Must be one of: gp2 | io1 | sc1 | st1 | standard	gp2

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

6.17.10 EC2

EC2 Instance

Table 166: *EC2*

Field name	Type	Purpose	Constraints	Default
associate_public_ip_address	Boolean	Associate Public IP Address		False
disable_api_termination	Boolean	Disable API Termination		False
instance_ami	String	Instance AMI		
instance_key_pair	PacoReference	key pair for connections to instance	Paco Reference to EC2KeyPair.	
instance_type	String	Instance type		
private_ip_address	String	Private IP Address		
root_volume_size_gb	Int	Root volume size GB		8
security_groups	List<PacoReference>	Security groups	Paco Reference to SecurityGroup.	
segment	String	Segment		
user_data_script	String	User data script		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

6.17.11 ECRRepository

Elastic Container Registry (ECR) Repository is a fully-managed Docker container registry.

Prescribed Automation

`cross_account_access`: Adds a Repository Policy that grants full access to the listed AWS Accounts.

Listing 27: Example ECRRepository

```
type: ECRRepository
enabled: true
order: 10
repository_name: 'ecr-example'
cross_account_access:
  - paco.ref accounts.dev
  - paco.ref accounts.tools
```

Table 167: *ECRRepository*

Field name	Type	Purpose	Constraints	Default
account	PacoReference	Account the ECR Repository belongs to	Paco Reference to Account.	
cross_account_access	List<PacoReference>	Accounts to grant access to this ECR.	Paco Reference to Account.	
lifecycle_policy_registry_id	String	Lifecycle Policy Registry Id		
lifecycle_policy_text	String	Lifecycle Policy		
repository_name	String	Repository Name		
repository_policy	Object<Policy>	Repository Policy		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

ECRRepositoryPermission

Table 168: *ECRRepositoryPermission*

Field name	Type	Purpose	Constraints	Default
permission	Choice	Permission	Must be one of 'Push', 'Pull' or 'PushAndPull'	
repository	PacoReference	ECR Repository	Paco Reference to *ECRRepository*.	

6.17.12 ECSCluster

The `ECSCluster` resource type creates an Amazon Elastic Container Service (Amazon ECS) cluster.

Listing 28: example ECSCluster configuration YAML

```
type: ECSCluster
title: My ECS Cluster
enabled: true
order: 10
```

Table 169: *ECSCluster*

Field name	Type	Purpose	Constraints	Default

Base Schemas Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

6.17.13 ECSServices

The `ECSServices` resource type creates one or more ECS Services and their TaskDefinitions that can run in an *ECSCluster*.

Listing 29: example ECSServices configuration YAML

```
type: ECSServices
title: "My ECS Services"
enabled: true
order: 40
cluster: paco.ref netenv.mynet.applications.myapp.groups.ecs.resources.cluster
service_discovery_namespace_name: 'private-name'
secrets_manager_access:
  - paco.ref netenv.mynet.secrets_manager.store.database.mydb
task_definitions:
  frontend:
    container_definitions:
      frontend:
```

(continues on next page)

(continued from previous page)

```
        cpu: 256
        essential: true
        image: paco.ref netenv.mynet.applications.myapp.groups.ecr.resources.frontend
        image_tag: latest
        memory: 150 # in MiB
        logging:
          driver: awslogs
          expire_events_after_days: 90
        port_mappings:
          - container_port: 80
            host_port: 0
            protocol: tcp
        secrets:
          - name: DATABASE_PASSWORD
            value_from: paco.ref netenv.mynet.secrets_manager.store.database.mydb
        environment:
          - name: POSTGRES_HOSTNAME
            value: paco.ref netenv.mynet.applications.myapp.groups.database.resources.
↪postgresql.endpoint.address
  demoservice:
    container_definitions:
      demoservice:
        cpu: 256
        essential: true
        image: paco.ref netenv.mynet.applications.myapp.groups.ecr.resources.
↪demoservice
        image_tag: latest
        memory: 100 # in MiB
        logging:
          driver: awslogs
          expire_events_after_days: 90
        port_mappings:
          - container_port: 80
            host_port: 0
            protocol: tcp

services:
  frontend:
    desired_count: 0
    task_definition: frontend
    deployment_controller: ecs
    hostname: frontend.myapp
    load_balancers:
      - container_name: frontend
        container_port: 80
        target_group: paco.ref netenv.mynet.applications.myapp.groups.lb.resources.
↪external.target_groups.frontend
  demoservice:
    desired_count: 0
    task_definition: demoservice
    deployment_controller: ecs
    load_balancers:
      - container_name: demoservice
        container_port: 80
        target_group: paco.ref netenv.mynet.applications.myapp.groups.lb.resources.
↪internal.target_groups.demoservice
```

Table 170: *ECSServices*

Field name	Type	Purpose	Constraints	Default
cluster	PacoReference	Cluster	Paco Reference to *ECSCluster*.	
secrets_manager_secrets	List<PacoReference>	List Secrets Manager secret Paco references	Paco Reference to SecretsManagerSecret.	
service_discovery_namespace	String	Service Discovery Namespace		
services	Container<*ECSServicesContainer*>	Service		
setting_groups	Container<**ECSSettingsGroups**>	Setting Groups		
task_definitions	Container<*ECSTaskDefinitions*>	Task Definitions		

Base Schemas Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

ECSServicesContainer

Container for *ECSService* objects.

Table 171: *ECSServicesContainer* Container<ECSService>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ECSService

ECS Service

Table 172: *ECSService*

Field name	Type	Purpose	Constraints	Default
deployment_controller	Choice	Deployment Controller	One of ecs, code_deploy or external	ecs
deployment_maximum_percent	Int	Deployment Maximum Percent		200
deployment_minimum_healthy_percent	Int	Deployment Minimum Healthy Percent		100
desired_count	Int	Desired Count		
health_check_grace_period_seconds	Int	Health Check Grace Period (seconds)		0
hostname	String	Container hostname		
launch_type	Choice	Launch Type	Must be one of EC2 or Fargate	EC2
load_balancers	List<*ECSLoadBalancer*>	Load Balancers		[]
maximum_tasks	Int	Maximum Tasks in service		0
minimum_tasks	Int	Minimum Tasks in service		0
suspend_scaling	Boolean	Suspend any Service Scaling activities		False
target_tracking_scaling_policies	Container<*ECSTargetTrackingScalingPolicies*>	Target Tracking Scaling Policies		
task_definition	String	Task Definition		
vpc_config	Object<*ServiceVPCConfiguration*>	VPC Configuration		

Base Schemas Monitorable, Named, Title

ECSTaskDefinitions

Container for *ECSTaskDefinition* objects.

Table 173: *ECSTaskDefinitions* Container<ECSTaskDefinition>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ECSTaskDefinition

ECS Task Definition

Table 174: *ECSTaskDefinition*

Field name	Type	Purpose	Constraints	Default
container_definitions	Container<*ECSContainerDefinition*>	Container Definitions		
cpu_units	Int	CPU in Units	Must be one of 256, 512, 1024, 2048 or 4096	256
fargate_compatible	Boolean	Require Fargate Compability		False
memory_in_mb	Int	Memory in Mb	Must be one of 512, 1024, 2048, 2048 or 4096 thru 30720	512
network_mode	Choice	Network Mode	Must be one of awsvpc, bridge, host or none	bridge
volumes	List<*ECSVolume*>	Volume definitions for the task		[]

Base Schemas Named, Title

ECSContainerDefinitions

Container for *ECSContainerDefinition* objects.

Table 175: *ECSContainerDefinitions* Container<ECSContainerDefinition>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ECSContainerDefinition

ECS Container Definition

Table 176: *ECSContainerDefinition*

Field name	Type	Purpose	Constraints	Default
command	List<String>	Command (Docker CMD)	List of strings	
cpu	Int	Cpu units		
depends_on	List<*ECSContainerDependency*>	Depends On	List of ECS Container Dependencies	[]
disable_networking	Boolean	Disable Networking		False
dns_search_domains	List<String>	List of DNS search domains. Maps to 'DnsSearch' in Docker.		[]
dns_servers	List<String>	List of DNS servers. Maps to 'Dns' in Docker.		[]
docker_labels	Container<*DockerLabels*>	A key/value map of labels. Maps to 'Labels' in Docker.		
docker_security_options	List<String>	List of custom labels for SELinux and AppArmor multi-level security systems.	Must be a list of no-new-privileges, apparmor:PROFILE, label:value, or credentialspec:CredentialSpecFilePath	[]
entry_point	List<String>	Entry Pont (Docker ENTRYPOINT)	List of strings	
environment	List<*NameValuePair*>	List of environment name value pairs.		
essential	Boolean	Essential		False
extra_hosts	List<*ECSHostEntry*>	List of hostnames and IP address mappings to append to the /etc/hosts file on the container.		[]
health_check	Object<*ECSHealthCheck*>	The container health check command and associated configuration parameters for the container. This parameter maps to 'HealthCheck' in Docker.		
hostname	String	Hostname to use for your container. This parameter maps to 'Hostname' in Docker.		
image	PacoReference|String	The image used to start a container. This string is passed directly to the Docker daemon.	If a paco.ref is used to ECR, then the image_tag field will provide that tag used. Paco Reference to *ECRRepository*. String Ok.	

Continued on next page

Table 176 – continued from previous page

Field name	Type	Purpose	Constraints	Default
image_tag	String	Tag used for the ECR Repository Image		latest
interactive	Boolean	When this parameter is true, this allows you to deploy containerized applications that require stdin or a tty to be allocated. This parameter maps to 'OpenStdin' in Docker.		
logging	Object<*ECSLogging*>	Logging Configuration		
memory	Int	The amount (in MiB) of memory to present to the container. If your container attempts to exceed the memory specified here, the container is killed.		
memory_reservation	Int	The soft limit (in MiB) of memory to reserve for the container. When system memory is under heavy contention, Docker attempts to keep the container memory to this soft limit.		
mount_points	List<*ECSMountPoint*>	The mount points for data volumes in your container.		
port_mappings	List<*PortMapping*>	Port Mappings		[]
privileged	Boolean	Give the container elevated privileges on the host container instance (similar to the root user).		False
pseudo_terminal	Boolean	Allocate a TTY. This parameter maps to 'Tty' in Docker.		
readonly_root_filesystem	Boolean	Read-only access to its root file system. This parameter maps to 'ReadonlyRootfs' in Docker.		
secrets	List<*ECSTaskDefinitionSecret*>	List of name, value_from pairs to secret manager Paco references.		
setting_groups	List<String>	List of names of setting_groups.		[]
start_timeout	Int	Time duration (in seconds) to wait before giving up on resolving dependencies for a container.		300
stop_timeout	Int	Time duration (in seconds) to wait before the container is forcefully killed if it doesn't exit normally on its own.		30
ulimits	List<*ECSUlimit*>	List of ulimits to set in the container. This parameter maps to 'Ulimits' in Docker		[]
user	String	The user name to use inside the container. This parameter maps to 'User' in Docker.		

Continued on next page

Table 176 – continued from previous page

Field name	Type	Purpose	Constraints	Default
volumes_from	List<*ECSVolumesFrom*>	Volumes to mount from another container (Docker VolumesFrom).		[]
working_directory	String	The working directory in which to run commands inside the container. This parameter maps to 'WorkingDir' in Docker.		

Base Schemas Named, Title

ECSLoadBalancer

ECS Load Balancer

Table 177: *ECSLoadBalancer*

Field name	Type	Purpose	Constraints	Default
container_name	String	Container Name		
container_port	Int	Container Port		
target_group	PacoReference	Target Group	Paco Reference to *TargetGroup*.	

Base Schemas Named, Title

ECSVolume

ECS Volume

Table 178: *ECSVolume*

Field name	Type	Purpose	Constraints	Default
name	String	Name		

ECSUlimit

ECS Ulimit

Table 179: *ECSUlimit*

Field name	Type	Purpose	Constraints	Default
hard_limit	Int	The hard limit for the ulimit type.		
name	Choice	The type of the ulimit		
soft_limit	Int	The soft limit for the ulimit type.		

ECSHealthCheck

ECS Health Check

Table 180: *ECSHealthCheck*

Field name	Type	Purpose	Constraints	Default
command	List<String>	A string array representing the command that the container runs to determine if it is healthy. The string array must start with CMD to execute the command arguments directly, or CMD-SHELL to run the command with the container's default shell.		
interval	Int	The time period in seconds between each health check execution.		30
retries	Int	Retries		3
start_period	Int	The optional grace period within which to provide containers time to bootstrap before failed health checks count towards the maximum number of retries.		
timeout	Int	The time period in seconds to wait for a health check to succeed before it is considered a failure.		5

Base Schemas Named, Title

ECSHostEntry

ECS Host Entry

Table 181: *ECSHostEntry*

Field name	Type	Purpose	Constraints	Default
hostname	String	Hostname		
ip_address	String	IP Address		

DockerLabels

Table 182: *DockerLabels*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ECSContainerDependency

ECS Container Dependency

Table 183: *ECSContainerDependency*

Field name	Type	Purpose	Constraints	Default
condition	Choice	Condition	Must be one of COMPLETE, HEALTHY, START or SUCCESS	
container_name	String	Container Name	Must be an existing container name.	

ECSTaskDefinitionSecret

A Name/ValueFrom pair of Paco references to Secrets Manager secrets

Table 184: *ECSTaskDefinitionSecret*

Field name	Type	Purpose	Constraints	Default
name	String	Name		
value_from	PacoReference	Paco reference to Secrets manager	Paco Reference to SecretsManagerSecret.	

ECSLogging

ECS Logging Configuration

Table 185: *ECSLogging*

Field name	Type	Purpose	Constraints	Default
driver	Choice	Log Driver	One of awsfirelens, awslogs, fluentd, gelf, journald, jsonfile, splunk, syslog	

Base Schemas CloudWatchLogRetention, Named, Title

ECSVolumesFrom

VoumesFrom

Table 186: *ECSVolumesFrom*

Field name	Type	Purpose	Constraints	Default
read_only	Boolean	Read Only		False
source_container	String	The name of another container within the same task definition from which to mount volumes.		

ECSTargetTrackingScalingPolicies

Container for *ECSTargetTrackingScalingPolicy* objects.

Table 187: *ECSTargetTrackingScalingPolicies* Container<ECSTargetTrackingScalingPolicy>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

ECSTargetTrackingScalingPolicy

Table 188: *ECSTargetTrackingScalingPolicy*

Field name	Type	Purpose	Constraints	Default
disable_scale_in	Boolean	Disable ScaleIn		False
predefined_metric	Choice	Predfined Metric to scale on	Must be one of ALBRequestCountPerTarget, ECSServiceAverageMemoryUtilization or ECSServiceAverageCPUUtilization	
scale_in_cooldown	Int	ScaleIn Cooldown		300
scale_out_cooldown	Int	ScaleIn Cooldown		300
target	Int	Target		
target_group	PacoReference	ALB TargetGroup	Paco Reference to *TargetGroup*.	

Base Schemas Enablable, Named, Title

ServiceVPCConfiguration

Table 189: *ServiceVPCConfiguration*

Field name	Type	Purpose	Constraints	Default
assign_public_ip	Boolean	Assign Public IP		False

Base Schemas Named, VPCConfiguration, Title

ECSMountPoint

ECS TaskDefinition Mount Point

Table 190: *ECSMountPoint*

Field name	Type	Purpose	Constraints	Default
container_path	String	The path on the container to mount the host volume at.		
read_only	Boolean	Read Only		False
source_volume	String	The name of the volume to mount.	Must be a volume name referenced in the name parameter of task definition volume.	

PortMapping

Port Mapping

Table 191: *PortMapping*

Field name	Type	Purpose	Constraints	Default
container_port	Int	Container Port		
host_port	Int	Host Port		
protocol	Choice	Protocol	Must be either 'tcp' or 'udp'	tcp

6.17.14 EIP

Elastic IP (EIP) resource.

Prescribed Automation

`dns`: Adds a DNS CNAME to resolve to this EIP's IP address to the Route 53 HostedZone.

Listing 30: Example EIP resource YAML

```
type: EIP
order: 5
enabled: true
dns:
  - domain_name: example.com
    hosted_zone: paco.ref resource.route53.examplecom
    ttl: 60
```

Table 192: *EIP*

Field name	Type	Purpose	Constraints	Default
dns	List<*DNS*>	List of DNS for the EIP		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

6.17.15 EFS

AWS Elastic File System (EFS) resource.

Listing 31: Example EFS resource

```
type: EFS
order: 20
enabled: true
encrypted: false
segment: private
security_groups:
  - paco.ref netenv.mynet.network.vpc.security_groups.cloud.content
```

Table 193: *EFS*

Field name	Type	Purpose	Constraints	Default
encrypted	Boolean	Encryption at Rest		False
security_groups	List<PacoReference>	Security groups	SecurityGroup the EFS belongs to Paco Reference to SecurityGroup.	
segment	String	Segment		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

6.17.16 ElastiCache

Base ElastiCache Interface

Table 194: *ElastiCache*

Field name	Type	Purpose	Constraints	Default
at_rest_encryption	Boolean	Enable encryption at rest		
auto_minor_version_upgrade	Boolean	Enable automatic minor version upgrades		
automatic_failover_enabled	Boolean	Specifies whether a read-only replica is automatically promoted to read/write primary if the existing primary fails		
az_mode	String	AZ mode		
cache_clusters	Int	Number of Cache Clusters		
cache_node_type	String	Cache Node Instance type		
description	String	Replication Description		
engine	String	ElastiCache Engine		
engine_version	String	ElastiCache Engine Version		
maintenance_preferred_window	String	Preferred maintenance window		
number_of_read_replicas	Int	Number of read replicas		
parameter_group	PacoReference|String	Parameter Group name	Paco Reference to Interface. String Ok.	
port	Int	Port		
security_groups	List<PacoReference>	List of Security Groups	Paco Reference to Security-Group.	
segment	PacoReference	Segment	Paco Reference to Segment.	

ElastiCacheRedis

Redis ElastiCache Interface

Table 195: *ElastiCacheRedis*

Field name	Type	Purpose	Constraints	Default
cache_parameter_group_family	String	Cache Parameter Group Family		
snapshot_retention_limit	Int	Snapshot Retention Limit in Days		
snapshot_window	String	The daily time range (in UTC) during which ElastiCache begins taking a daily snapshot of your node group (shard).		

Base Schemas *ElastiCache*, Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

6.17.17 ElasticsearchDomain

Amazon Elasticsearch Service (Amazon ES) is a managed service for Elasticsearch clusters. An Amazon ES domain is synonymous with an Elasticsearch cluster. Domains are clusters with the settings, instance types, instance counts, and storage resources that you specify.

Prescribed Automation

`segment`: Including the segment will place the Elasticsearch cluster within the Availability Zones for that segment. If an Elasticsearch ServiceLinkedRole is not already provisioned for that account and region, Paco will create it for you. This role is used by AWS to place the Elasticsearch cluster within the subnets that belong that segment and VPC.

If segment is not set, then you will have a public Elasticsearch cluster with an endpoint.

Listing 32: example Elasticsearch configuration

```
type: ElasticsearchDomain
order: 10
title: "Elasticsearch Domain"
enabled: true
access_policies_json: ./es-config/es-access.json
advanced_options:
  indices.fielddata.cache.size: ""
  rest.action.multi.allow_explicit_index: "true"
cluster:
  instance_count: 2
  zone_awareness_enabled: false
  instance_type: "t2.micro.elasticsearch"
  dedicated_master_enabled: true
  dedicated_master_type: "t2.micro.elasticsearch"
  dedicated_master_count: 2
ebs_volumes:
  enabled: true
  iops: 0
  volume_size_gb: 10
  volume_type: 'gp2'
segment: web
security_groups:
  - paco.ref netenv.mynet.network.vpc.security_groups.app.search
```

Table 196: *ElasticsearchDomain*

Field name	Type	Purpose	Constraints	Default
access_policies_json	StringFileReference	Policy document that specifies who can access the Amazon ES domain and their permissions.		
advanced_options	Container<*ESAdvancedOptions*>	Advanced Options		
cluster	Object<*ElasticsearchCluster*>	Elasticsearch Cluster configuration		
ebs_volumes	Object<*EBSOptions*>	EBS volumes that are attached to data nodes in the Amazon ES domain.		
elasticsearch_version	String	The version of Elasticsearch to use, such as 2.3.		1.5
node_to_node_encryption	Boolean	Enable node-to-node encryption		
security_groups	List<PacoReference>	List of Security Groups	Paco Reference to SecurityGroup.	
segment	String	Segment		
snapshot_start_hour	Int	The hour in UTC during which the service takes an automated daily snapshot of the indices in the Amazon ES domain.		

Base Schemas Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

ElasticsearchCluster

Table 197: *ElasticsearchCluster*

Field name	Type	Purpose	Constraints	Default
dedicated_master_count	Int	The number of instances to use for the master node.	If you specify this field, you must specify true for the dedicated_master_enabled field.	
dedicated_master_enabled	Boolean	Indicates whether to use a dedicated master node for the Amazon ES domain.		
dedicated_master_type	String	The hardware configuration of the computer that hosts the dedicated master node	Valid Elasticsearch instance type, such as m3.medium.elasticsearch. See https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/aes-supported-instance-types.html	
instance_count	Int	The number of data nodes (instances) to use in the Amazon ES domain.		
instance_type	String	The instance type for your data nodes.	Valid Elasticsearch instance type, such as m3.medium.elasticsearch. See https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/aes-supported-instance-types.html	
zone_awareness_availability_zone_count	Int	If you enabled multiple Availability Zones (AZs), the number of AZs that you want the domain to use.		2
zone_awareness_enabled	Boolean	Enable zone awareness for the Amazon ES domain.		

EBSOptions

Table 198: *EBSOptions*

Field name	Type	Purpose	Constraints	Default
enabled	Boolean	Specifies whether Amazon EBS volumes are attached to data nodes in the Amazon ES domain.		
iops	Int	The number of I/O operations per second (IOPS) that the volume supports.		
volume_size_gb	Int	The size (in GiB) of the EBS volume for each data node.	The minimum and maximum size of an EBS volume depends on the EBS volume type and the instance type to which it is attached.	
volume_type	String	The EBS volume type to use with the Amazon ES domain.	Must be one of: standard, gp2, io1, st1, or sc1	

ESAdvancedOptions

An unconstrainted set of key-value pairs used to set advanced options for Elasticsearch.

6.17.18 EventsRule

Events Rule resources match incoming or scheduled events and route them to target using Amazon EventBridge.

Prescribed Automation

`targets`: If the `target` is a Lambda, an IAM Role will be created that is granted permission to invoke it by this EventRule.

Listing 33: Lambda function resource YAML

```
type: EventsRule
enabled: true
order: 10
description: Invoke a Lambda every other minute
schedule_expression: "cron(*/2 * * * ? *)"
targets:
    - target: paco.ref netenv.mynet.applications.myapp.groups.mygroup.resources.
↪mylambda
```

Table 199: *EventsRule*

Field name	Type	Purpose	Constraints	Default
description	String	Description		
enabled_state	Boolean	Enabled State		True
schedule_expression	String	Schedule Expression		
targets	List<*EventTarget*>	The AWS Resources that are invoked when the Rule is triggered.		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

EventTarget

Table 200: *EventTarget*

Field name	Type	Purpose	Constraints	Default
input_json	String	Valid JSON passed as input to the target.		
target	PacoReference	Paco Reference to an AWS Resource to invoke	Paco Reference to Interface.	

Base Schemas Named, Title

6.17.19 Lambda

Lambda Functions allow you to run code without provisioning servers and only pay for the compute time when the code is running.

The code for the Lambda function can be specified in one of three ways in the `code:` field:

- S3 Bucket artifact: Supply an‘‘s3_bucket‘‘ and `s3_key` where you have an existing code artifact file.
- Local file: Supply the `zipfile` as a path to a local file on disk. This will be inlined into CloudFormation and has a size limitation of only 4 Kb.
- Local directory: Supply the `zipfile` as a path to a directory on disk. This directory will be packaged into a zip file and Paco will create an S3 Bucket where it will upload and manage Lambda deployment artifacts.

Listing 34: Lambda code from S3 Bucket or local disk

```
code:
    s3_bucket: my-bucket-name
    s3_key: 'myapp-1.0.zip'

code:
    zipfile: ./lambda-dir/my-lambda.py

code:
    zipfile: ~/code/my-app/lambda_target/
```

Prescribed Automation

`expire_events_after_days`: Sets the Retention for the Lambda execution Log Group.

`log_group_names`: Creates CloudWatch Log Group(s) prefixed with '<env>-<appname>-<groupname>-<lambdaname>-' (or for Environment-less applications like Services it will be '<appname>-<groupname>-<lambdaname>-') and grants permission for the Lambda role to interact with those Log Group(s). The `expire_events_after_days` field will set the Log Group retention period. Paco will also add a comma-seperated Environment Variable named PACO_LOG_GROUPS to the Lambda with the expanded names of the Log Groups.

`sdb_cache`: Create a SimpleDB Domain and IAM Policy that grants full access to that domain. Will also make the domain available to the Lambda function as an environment variable named `SDB_CACHE_DOMAIN`.

`sns_topics`: Subscribes the Lambda to SNS Topics. For each Paco reference to an SNS Topic, Paco will create an SNS Topic Subscription so that the Lambda function will recieve all messages sent to that SNS Topic. It will also create a Lambda Permission granting that SNS Topic the ability to publish to the Lambda.

Lambda Permissions Paco will check all resources in the Application for any: S3Bucket configured to notify this Lambda, EventsRule to invoke this Lambda, IoTAnalyticsPipeline activities to invoke this Lambda. These resources will automatically gain a Lambda Permission to be able to invoke the Lambda.

Listing 35: Lambda function resource YAML

```
type: Lambda
enabled: true
order: 1
title: 'My Lambda Application'
description: 'Checks the Widgets Service and applies updates to a Route 53 Record Set.
↪'
code:
    s3_bucket: my-bucket-name
    s3_key: 'myapp-1.0.zip'
environment:
    variables:
    - key: 'VAR_ONE'
      value: 'hey now!'
    - key: 'VAR_TWO'
      value: 'Hank Kingsley'
iam_role:
    enabled: true
    policies:
      - name: DNSRecordSet
        statement:
          - effect: Allow
            action:
              - route53:ChangeResourceRecordSets
            resource:
              - 'arn:aws:route53:::hostedzone/AJKDU9834DUY934'
handler: 'myapp.lambda_handler'
memory_size: 128
runtime: 'python3.7'
timeout: 900
expire_events_after_days: 90
log_group_names:
  - AppGroupOne
sns_topics:
```

(continues on next page)

(continued from previous page)

```
  - paco.ref netenv.app.applications.app.groups.web.resources.snstopic
vpc_config:
    segments:
      - paco.ref netenv.app.network.vpc.segments.public
    security_groups:
      - paco.ref netenv.app.network.vpc.security_groups.app.function
```

Table 201: *Lambda*

Field name	Type	Purpose	Constraints	Default
code	Object<*LambdaFunctionCode*>	The function deployment package.		
description	String	A description of the function.		
edge	Object<*LambdaAtEdgeConfiguration*>	Lambda@Edge configuration		
environment	Object<*LambdaEnvironment*>	Lambda Function Environment		
handler	String	Function Handler		
iam_role	Object<Role>	The IAM Role this Lambda will execute as.		
layers	List<String>	Layers	Up to 5 Layer ARNs	
log_group_names	List<String>	Log Group names	List of Log Group names	[]
memory_size	Int	Function memory size (MB)		128
reserved_concurrent_executions	Int	Reserved Concurrent Executions		0
runtime	String	Runtime environment		python3.7
sdb_cache	Boolean	SDB Cache Domain		False
sns_topics	List<PacoReference>	List of SNS Topic Paco references or SNS Topic ARNs to subscribe the Lambda to.	Paco Reference to *SNSTopic*. String Ok.	
timeout	Int	The amount of time that Lambda allows a function to run before stopping it.		3
vpc_config	Object<*LambdaVpcConfig*>	Vpc Configuration		

Base Schemas Resource, DNSEnablable, Deployable, CloudWatchLogRetention, Monitorable, Named, Title, Type

LambdaFunctionCode

The deployment package for a Lambda function.

Table 202: *LambdaFunctionCode*

Field name	Type	Purpose	Constraints	Default
s3_bucket	PacoReference|String	An Amazon S3 bucket in the same AWS Region as your function	Paco Reference to *S3Bucket*. String Ok.	
s3_key	String	The Amazon S3 key of the deployment package.		
zipfile	LocalPath	The function code as a local file or directory.	Maximum of 4096 characters.	

LambdaEnvironment

Lambda Environment

Table 203: *LambdaEnvironment*

Field name	Type	Purpose	Constraints	Default
variables	List<*LambdaVariable*>	Lambda Function Variables		

LambdaVpcConfig

Lambda Environment

Table 204: *LambdaVpcConfig*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, VPCConfiguration, Title

LambdaVariable

Lambda Environment Variable

Table 205: *LambdaVariable*

Field name	Type	Purpose	Constraints	Default
key	String	Variable Name		
value	PacoReference|String	String Value or a Paco Reference to a resource output	Paco Reference to Interface. String Ok.	

LambdaAtEdgeConfiguration

Table 206: *LambdaAtEdgeConfiguration*

Field name	Type	Purpose	Constraints	Default
auto_publish_version	String	Automatically publish a Version. Update this name to publish a new Version.		

Base Schemas Enablable, Named, Title

6.17.20 LoadBalancer

Base class for Load Balancers

Table 207: *LoadBalancer*

Field name	Type	Purpose	Constraints	Default
access_logs_bucket	PacoReference	Bucket to store access logs in	Paco Reference to *S3Bucket*.	
access_logs_prefix	String	Access Logs S3 Bucket prefix		
dns	List<*DNS*>	List of DNS for the ALB		
enable_access_logs	Boolean	Write access logs to an S3 Bucket		
idle_timeout_secs	Int	Idle timeout in seconds	The idle timeout value, in seconds.	60
listeners	Container<*Listeners*>	Listeners		
scheme	Choice	Scheme		
security_groups	List<PacoReference>	Security Groups	Paco Reference to SecurityGroup.	
segment	String	Id of the segment stack		
target_groups	Container<*TargetGroups*>	Target Groups		

Base Schemas Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

6.17.21 ApplicationLoadBalancer

The `LBApplication` resource type creates an Application Load Balancer. Use load balancers to route traffic from the internet to your web servers.

Load balancers have `listeners` which will accept requrests on specified ports and protocols. If a listener uses the HTTPS protocol, it can have a Paco reference to an SSL Certificate. A listener can then either redirect the traffic to another port/protcol or send it one of it's named `target_groups`.

Each target group will specify it's health check configuration. To specify which resources will belong to a target group, use the `target_groups` field on an ASG resource.

Prescribed Automation

`dns`: Creates Route 53 Record Sets that will resolve DNS records to the domain name of the load balancer.

`enable_access_logs`: Set to True to turn on access logs for the load balancer, and will automatically create an S3 Bucket with permissions for AWS to write to that bucket.

`access_logs_bucket`: Name an existing S3 Bucket (in the same region) instead of automatically creating a new one. Remember that if you supply your own S3 Bucket, you are responsible for ensuring that the bucket policy for it grants AWS the *s3:PutObject* permission.

Listing 36: Example LBApplication load balancer resource YAML

```
type: LBApplication
enabled: true
enable_access_logs: true
target_groups:
    api:
        health_check_interval: 30
```

(continues on next page)

(continued from previous page)

```
        health_check_timeout: 10
        healthy_threshold: 2
        unhealthy_threshold: 2
        port: 3000
        protocol: HTTP
        health_check_http_code: 200
        health_check_path: /
        connection_drain_timeout: 30
listeners:
    http:
        port: 80
        protocol: HTTP
        redirect:
            port: 443
            protocol: HTTPS
    https:
        port: 443
        protocol: HTTPS
        ssl_certificates:
            - paco.ref netenv.app.applications.app.groups.certs.resources.root
        target_group: api
dns:
    - hosted_zone: paco.ref resource.route53.mynetenv
      domain_name: api.example.com
scheme: internet-facing
security_groups:
    - paco.ref netenv.app.network.vpc.security_groups.app.alb
segment: public
```

Table 208: *ApplicationLoadBalancer*

Field name	Type	Purpose	Constraints	Default

Base Schemas *LoadBalancer*, Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

6.17.22 NetworkLoadBalancer

The `LBNetwork` resource type creates a Network Load Balancer. Use load balancers to route traffic from the internet to your web servers.

Prescribed Automation

`dns`: Creates Route 53 Record Sets that will resolve DNS records to the domain name of the load balancer.

`enable_access_logs`: Set to True to turn on access logs for the load balancer, and will automatically create an S3 Bucket with permissions for AWS to write to that bucket.

`access_logs_bucket`: Name an existing S3 Bucket (in the same region) instead of automatically creating a new one. Remember that if you supply your own S3 Bucket, you are responsible for ensuring that the bucket policy for it grants AWS the *s3:PutObject* permission.

Listing 37: Example LBNetwork load balancer resource YAML

```
type: LBNetwork
enabled: true
enable_access_logs: true
target_groups:
    api:
        health_check_interval: 30
        health_check_timeout: 10
        healthy_threshold: 2
        unhealthy_threshold: 2
        port: 3000
        protocol: HTTP
        health_check_http_code: 200
        health_check_path: /
        connection_drain_timeout: 30
listeners:
    http:
        port: 80
        protocol: HTTP
        redirect:
            port: 443
            protocol: HTTPS
    https:
        port: 443
        protocol: HTTPS
        ssl_certificates:
            - paco.ref netenv.app.applications.app.groups.certs.resources.root
        target_group: api
dns:
    - hosted_zone: paco.ref resource.route53.mynetenv
      domain_name: api.example.com
scheme: internet-facing
segment: public
```

Table 209: *NetworkLoadBalancer*

Field name	Type	Purpose	Constraints	Default

Base Schemas *LoadBalancer*, Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

DNS

Table 210: *DNS*

Field name	Type	Purpose	Constraints	Default
domain_name	PacoReference|String	Domain name	Paco Reference to Route53HostedZone. String Ok.	
hosted_zone	PacoReference|String	Hosted Zone Id	Paco Reference to HostedZone. String Ok.	
ssl_certificate	PacoReference	SSL certificate Reference	Paco Reference to *ACM*.	
ttl	Int	TTL		300

Listeners

Container for *Listener* objects.

Table 211: *Listeners* Container<Listener>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

Listener

Table 212: *Listener*

Field name	Type	Purpose	Constraints	Default
redirect	Object<*PortProtocol*>	Redirect		
rules	Container<**ListenerRules_**>	Container of listener rules		
ssl_certificates	List<PacoReference>	List of SSL certificate References	Paco Reference to *ACM*.	
ssl_policy	Choice	SSL Policy		
target_group	String	Target group		

Base Schemas *PortProtocol*

ListenerRule

Table 213: *ListenerRule*

Field name	Type	Purpose	Constraints	Default
host	String	Host header value		
path_pattern	List<String>	List of paths to match		
priority	Int	Forward condition priority		1
redirect_host	String	The host to redirect to		
rule_type	String	Type of Rule		
target_group	String	Target group name		

Base Schemas Deployable, Named, Title

PortProtocol

Port and Protocol

Table 214: *PortProtocol*

Field name	Type	Purpose	Constraints	Default
port	Int	Port		
protocol	Choice	Protocol		

TargetGroups

Container for *TargetGroup* objects.

Table 215: *TargetGroups* Container<TargetGroup>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

TargetGroup

Target Group

Table 216: *TargetGroup*

Field name	Type	Purpose	Constraints	Default
connection_drain_timeout	Int	Connection drain timeout		
health_check_http_codes	String	Health check HTTP codes		
health_check_interval	Int	Health check interval		
health_check_path	String	Health check path		/
health_check_protocol	Choice	Protocol		HTTP
health_check_timeout	Int	Health check timeout		
healthy_threshold	Int	Healthy threshold		
target_type	Choice	Target Type	Must be one of 'instance', 'ip' or 'lambda'.	instance
unhealthy_threshold	Int	Unhealthy threshold		

Base Schemas Resource, DNSEnablable, Deployable, Named, *PortProtocol*, Title, Type

6.17.23 PinpointApplication

Amazon Pinpoint is a flexible and scalable outbound and inbound marketing communications service. You can connect with customers over channels like email, SMS, push, or voice.

A Pinpoint Application is a collection of related settings, customer information, segments, campaigns, and other types of Amazon Pinpoint resources.

Currently AWS Pinpoint only supports general configuration suitable for sending transactional messages.

Prescribed Automation

`email_channel`: Will build an ARN to a Simple Email Service Verified Email in the same account and region.

Listing 38: example Pinpoint Application configuration

```
type: PinpointApplication
enabled: true
order: 20
title: "My SaaS Transactional Message Service"
email_channel:
    enable_email: true
    from_address: "bob@example.com"
sms_channel:
    enable_sms: true
    sender_id: MyUniqueName
```

Table 217: *PinpointApplication*

Field name	Type	Purpose	Constraints	Default
email_channel	Object<*PinpointEmailChannel*>	Email Channel		
sms_channel	Object<*PinpointSMSChannel*>	SMS Channel		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

PinpointSMSChannel

Pinpoint SMS Channel

Table 218: *PinpointSMSChannel*

Field name	Type	Purpose	Constraints	Default
enable_sms	Boolean	Enable SMS		True
sender_id	String	The identity that you want to display on recipients' devices when they receive messages from the SMS channel.		
short_code	String	The registered short code that you want to use when you send messages through the SMS channel.		

PinpointEmailChannel

Pinpoint Email Channel

Table 219: *PinpointEmailChannel*

Field name	Type	Purpose	Constraints	Default
enable_email	Boolean	Enable Email		True
from_address	String	The verified email address that you want to send email from when you send email through the channel.		

6.17.24 IoTTopicRule

IoTTopicRule allows you to create a list of actions that will be triggered from a MQTT message coming in to IoT Core.

Prescribed Automation

IoTTopicRule Role Every IoTTopicRule will have a Role created that it can assume to perform any actions that it has. For example, it will be allowed to call a Lambda or an IoTAnalyticsPipeline.

Listing 39: example IoTTopicRule configuration

```
type: IoTTopicRule
title: Rule to take action for MQTT messages sent to 'sensor/example'
order: 20
enabled: true
actions:
  - awslambda:
      function: paco.ref netenv.mynet.applications.app.groups.app.resources.iotlambda
  - iotanalytics:
      pipeline: paco.ref netenv.mynet.applications.app.groups.app.resources.
↪analyticspipeline
aws_iot_sql_version: '2016-03-23'
rule_enabled: true
sql: "SELECT * FROM 'sensor/example'"
```

Table 220: *IoTTopicRule*

Field name	Type	Purpose	Constraints	Default
actions	List<*IoTTopicRuleAction*>	Actions	An IoTTopicRule must define at least one action.	[]
aws_iot_sql_version	String	AWS IoT SQL Version		2016-03-23
rule_enabled	Boolean	Rule is Enabled		True
sql	String	SQL statement used to query the topic	Must be a valid Sql statement	

Base Schemas Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

IoTTopicRuleAction

Table 221: *IoTTopicRuleAction*

Field name	Type	Purpose	Constraints	Default
awslambda	Object<*IoTTopicRuleLambdaAction*>	Lambda Action		
iotanalytics	Object<*IoTTopicRuleIoTAnalyticsAction*>	IoT Analytics Action		

IoTTopicRuleIoTAnalyticsAction

Table 222: *IoTTopicRuleIoTAnalyticsAction*

Field name	Type	Purpose	Constraints	Default
pipeline	PacoReference	IoT Analytics pipeline	Paco Reference to *IoTAnalyticsPipeline*.	

IoTTopicRuleLambdaAction

Table 223: *IoTTopicRuleLambdaAction*

Field name	Type	Purpose	Constraints	Default
function	PacoReference	Lambda function	Paco Reference to *Lambda*.	

6.17.25 IoTAnalyticsPipeline

An IoTAnalyticsPipeline composes four closely related resources: IoT Analytics Channel, IoT Analytics Pipeline, IoT Analytics Datastore and IoT Analytics Dataset.

An IoT Analytics Pipeline begins with a Channel. A Channel is an S3 Bucket of raw incoming messages. A Channel provides an ARN that an IoTTopicRule can send MQTT messages to. These messages can later be re-processed if the analysis pipeline changes. Use the `channel_storage` field to configure the Channel storage.

Next the Pipeline applies a series of `pipeline_activities` to the incoming Channel messages. After any message modifications have been made, they are stored in a Datastore.

A Datastore is S3 Bucket storage of messages that are ready to be analyzed. Use the `datastore_storage` field to configure the Datastore storage. The `datastore_name` is an optional field to give your Datastore a fixed name, this can be useful if you use Dataset SQL Query analysis which needs to use the Datastore name in a SELECT query. However, if you use `datastore_name` it doesn't vary by Environment - if you use name then it is recommended to use different Regions and Accounts for each IoTAnalytics environment.

Lastly the Datastore can be analyzed and have the resulting output saved as a Dataset. There may be multiple Datasets to create different analysis of the data. Datasets can be analyzed on a managed host running a Docker container or with an SQL Query to create subsets of a Datastore suitable for analysis with tools such as AWS QuickSight.

Prescribed Automation

IoTAnalyticsPipeline Role Every IoTAnalyticsPipeline has an IAM Role associated with it. This Role will have access to every S3 Bucket that is referenced by a Channel, Datastore or Dataset.

`pipeline_activities`: Every list of activities beings with an implicit Channel activity and ends with a Datastore activity.

Listing 40: example IoTAnalyticsPipeline configuration

```
type: IoTAnalyticsPipeline
title: My IoT Analytics Pipeline
order: 100
enabled: true
channel_storage:
  bucket: paco.ref netenv.mynet.applications.app.groups.iot.resources.iotbucket
  key_prefix: raw_input/
pipeline_activities:
  adddatetime:
    activity_type: lambda
    function: paco.ref netenv.mynet.applications.app.groups.iot.resources.iotfunc
    batch_size: 10
  filter:
    activity_type: filter
    filter: "temperature > 0"
datastore_name: example
datastore_storage:
  expire_events_after_days: 30
datasets:
  hightemp:
    query_action:
      sql_query: "SELECT * FROM example WHERE temperature > 20"
    content_delivery_rules:
      s3temperature:
        s3_destination:
          bucket: paco.ref netenv.mynet.applications.app.groups.iot.resources.
↪iotbucket
          key: "/HighTemp/!{iotanalytics:scheduleTime}/!{iotanalytics:versionId}.csv"
    expire_events_after_days: 3
    version_history: 5
```

Table 224: *IoTAnalyticsPipeline*

Field name	Type	Purpose	Constraints	Default
channel_storage	Object<*IotAnalyticsStorage*>	IoT Analytics Channel raw storage		
datasets	Container<*IoTDatasets*>	IoT Analytics Datasets		
datastore_name	String	Datastore name		
datastore_storage	Object<*IotAnalyticsStorage*>	IoT Analytics Datastore storage		
pipeline_activities	Container<*IoTPipelineActivities*>	IoT Analytics Pipeline Activies		

Base Schemas Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

IoTDatasets

Container for *IoTDataset* objects.

Table 225: *IoTDatasets* Container<IoTDataset>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

IoTDataset

Table 226: *IoTDataset*

Field name	Type	Purpose	Constraints	Default
container_action	Object<*DatasetContainerAction*>	Dataset Container action		
content_delivery_rules	Container<*DatasetContentDeliveryRules*>	Content Delivery Rules		
query_action	Object<*DatasetQueryAction*>	SQL Query action		
triggers	List<*DatasetTrigger*>	Triggers		[]
version_history	Int	How many versions of dataset contents are kept. 0 indicates Unlimited. If not specified or set to null, only the latest version plus the latest succeeded version (if they are different) are kept for the time period specified by expire_events_after_days field.		

Base Schemas *StorageRetention*, Named, Title

DatasetTrigger

Table 227: *DatasetTrigger*

Field name	Type	Purpose	Constraints	Default
schedule_expression	String	Schedule Expression		
triggering_dataset	String	Triggering Dataset		

DatasetContentDeliveryRules

Container for *DatasetContentDeliveryRule* objects.

Table 228: *DatasetContentDeliveryRules* Container<DatasetContentDeliveryRule>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

DatasetContentDeliveryRule

Table 229: *DatasetContentDeliveryRule*

Field name	Type	Purpose	Constraints	Default
s3_destination	Object<*DatasetS3Destination*>	S3 Destination		

Base Schemas Named, Title

DatasetS3Destination

Table 230: *DatasetS3Destination*

Field name	Type	Purpose	Constraints	Default
bucket	PacoReference	S3 Bucket	Paco Reference to *S3Bucket*.	
key	String	Key		

DatasetQueryAction

Table 231: *DatasetQueryAction*

Field name	Type	Purpose	Constraints	Default
filters	List<String>	Filters		[]
sql_query	String	Sql Query Dataset Action object		

Base Schemas Named, Title

DatasetContainerAction

Table 232: *DatasetContainerAction*

Field name	Type	Purpose	Constraints	Default
image_arn	String	Image ARN		
resource_compute_type	Choice	Resource Compute Type	Either ACU_1 (vCPU=4, memory=16 GiB) or ACU_2 (vCPU=8, memory=32 GiB)	
resource_volume_size_gb	Int	Resource Volume Size in GB		
variables	Container<*DatasetVariables*>	Variables		

Base Schemas Named, Title

DatasetVariables

Container for *DatasetVariables* objects.

Table 233: *DatasetVariables* Container<DatasetVariables>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

DatasetVariable

Table 234: *DatasetVariable*

Field name	Type	Purpose	Constraints	Default
double_value	Float	Double Value		
output_file_uri_value	String	Output file URI value	The URI of the location where dataset contents are stored, usually the URI of a file in an S3 bucket.	
string_value	String	String Value		

Base Schemas Named, Title

IoTPipelineActivities

Container for *IoTPipelineActivity* objects.

Table 235: *IoTPipelineActivities* Container<IoTPipelineActivity>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

IoTPipelineActivity

Each activity must have an `activity_type` and supply fields specific for that type. There is an implicit Channel activity before all other activities and an an implicit Datastore activity after all other activities.

Listing 41: All example types for IoTAnalyticsPipeline pipeline_activities

```
activity_type: lambda
batch_size: 1
function: paco.ref netenv.mynet[...]mylambda

```

(continues on next page)

(continued from previous page)

```
activity_type: add_attributes
attributes:
  key1: hello
  key2: world

activity_type: remove_attributes
attribute_list:
  - key1
  - key2

activity_type: select_attributes
attribute_list:
  - key1
  - key2

activity_type: filter
filter: "attribute1 > 40 AND attribute2 < 20"

activity_type: math
attribute: "attribute1"
math: "attribute1 - 10"

activity_type: device_registry_enrich
attribute: "attribute1"
thing_name: "mything"

activity_type: device_shadow_enrich
attribute: "attribute1"
thing_name: "mything"
```

Table 236: *IoTPipelineActivity*

Field name	Type	Purpose	Constraints	Default
activity_type	String	Activity Type		
attribute	String	Attribute		
attribute_list	List<String>	Attribute List		
attributes	Container<*Attributes*>	Attributes		
batch_size	Int	Batch Size		
filter	String	Filter		
function	PacoReference	Lambda function	Paco Reference to *Lambda*.	
math	String	Math		
thing_name	String	Thing Name		

Base Schemas Named, Title

Attributes

Dictionary of Attributes

Table 237: *Attributes*

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

IotAnalyticsStorage

Table 238: *IotAnalyticsStorage*

Field name	Type	Purpose	Constraints	Default
bucket	PacoReference	S3 Bucket	Paco Reference to *S3Bucket*.	
key_prefix	String	Key Prefix for S3 Bucket		

Base Schemas *StorageRetention*, Named, Title

StorageRetention

Table 239: *StorageRetention*

Field name	Type	Purpose	Constraints	Default
expire_events_after_days	Int	Expire Events After Days	Must be 1 or greater. If set to an explicit 0 then it is considered unlimited.	0

6.17.26 ManagedPolicy

IAM Managed Policy

Table 240: *ManagedPolicy*

Field name	Type	Purpose	Constraints	Default
path	String	Path		/
policy_name	String	Policy Name used in AWS. This will be prefixed with an 8 character hash.		
roles	List<String>	List of Role Names		
statement	List<Statement>	Statements		
users	List<String>	List of IAM Users		

Base Schemas Deployable, Named, Title

6.17.27 RDS

Relational Database Service (RDS) allows you to set up, operate, and scale a relational database in AWS.

You can create a single DB Instance or an Aurora DB Cluster.

DB Instance

Currently Paco supports `RDSMysql` and `RDSPostgresql` for single database instances.

Prescribed Automation

Using Secrets Manager with RDS

You can set the initial password with `master_user_password`, however this requires storing a password in plain-text on disk. This is fine if you have a process for changing the password after creating a database, however, the Paco Secrets Manager support allows you to use a `secrets_password` instead of the `master_user_password` field:

```
type: RDSMysql
secrets_password: paco.ref netenv.mynet.secrets_manager.app.grp.mysql
```

Then in your NetworkEnvironments `secrets_manager` configuration you would write:

```
secrets_manager:
  app: # application name
    grp: # group name
        mysql: # secret name
          enabled: true
          generate_secret_string:
            enabled: true
            # secret_string_template and generate_string_key must
            # have the following values for RDS secrets
            secret_string_template: '{"username": "admin"}'
            generate_string_key: "password"
```

This would generate a new, random password in the AWS Secrets Manager service when the database is provisioned and connect that password with RDS.

Listing 42: RDSMysql resource example

```
type: RDSMysql
order: 1
title: "Joe's MySQL Database server"
enabled: true
engine_version: 5.7.26
db_instance_type: db.t3.micro
port: 3306
storage_type: gp2
storage_size_gb: 20
storage_encrypted: true
multi_az: true
allow_major_version_upgrade: false
auto_minor_version_upgrade: true
publically_accessible: false
master_username: root
master_user_password: "change-me"
backup_preferred_window: 08:00-08:30
backup_retention_period: 7
maintenance_preferred_window: 'sat:10:00-sat:10:30'
license_model: "general-public-license"
cloudwatch_logs_exports:
  - error
```

(continues on next page)

(continued from previous page)

```
  - slowquery
security_groups:
  - paco.ref netenv.mynet.network.vpc.security_groups.app.database
segment: paco.ref netenv.mynet.network.vpc.segments.private
primary_domain_name: database.example.internal
primary_hosted_zone: paco.ref netenv.mynet.network.vpc.private_hosted_zone
parameter_group: paco.ref netenv.mynet.applications.app.groups.web.resources.dbparams_
↪performance
```

Aurora DB Cluster

AWS Aurora is relational databases built for the cloud. Aurora features a distributed, fault-tolerant, self-healing storage system and can easily scale from a single database instance to a cluster of multiple database instances.

When creating an Aurora RDS resource, you must specify your `db_instances`. If you specify more than one database instance, then Aurora will automatically designate one instance as a Writer and all other instances will be Readers.

Each `db_instance` can specify it's own complete set of configuration or you can use the `default_instance` field to shared default configuration between instances. If a `db_instance` doesn't specify a value but it is specified by `default_instance` it will fall back to using that value.

A simple Aurora with only a single database instance could be:

Listing 43: Simple Aurora single instance

```
type: RDSMysqlAurora
default_instance:
  db_instance_type: db.t3.medium
db_instances:
  single:
```

A more complex Aurora with a cluster of three database instances could be:

Listing 44: Three instance Aurora cluster

```
type: RDSMysqlAurora
default_instance:
  db_instance_type: db.t3.medium
  enhanced_monitoring_interval_in_seconds: 30
db_instances:
  first:
    availability_zone: 1
    db_instance_type: db.t3.large
    enhanced_monitoring_interval_in_seconds: 5
  second:
    availability_zone: 2
  third:
    availability_zone: 3
```

Prescribed Automation

`secrets_password`: Uses a Secrets Manager secret for the database master password.

`enable_kms_encryption`: Encrypts the database storage. Paco will creates a KMS-CMK dedicated to the DB Cluster. This key can only be accessed by the AWS RDS service.

`enhanced_monitoring_interval_in_seconds`: Paco will create an IAM Role to allow the RDS monitoring service access to perform enhanced monitoring.

`cluster_event_notifications` and `event_notifications` must reference a group specified in `resource/sns.yaml`. This group (SNS Topic) must already be provisioned in the same account and region as the database.

`monitoring` applies to `db_instances` and will apply CloudWatch Alarms that are specific to each database instance in the Aurora cluster.

Listing 45: RDSPostgresqlAurora db cluster example

```
type: RDSPostgresqlAurora
order: 10
enabled: true
availability_zones: all
engine_version: '11.7'
port: 5432
master_username: master
secrets_password: paco.ref netenv.anet.secrets_manager.anet.app.database
backup_preferred_window: 04:00-05:00
backup_retention_period: 7
maintenance_preferred_window: 'Sat:07:00-Sat:08:00'
cluster_parameter_group: paco.ref netenv.mynet.applications.app.groups.web.resources.
↪clusterparams
cloudwatch_logs_exports:
  - error
security_groups:
  - paco.ref netenv.mynet.network.vpc.security_groups.app.database
segment: paco.ref netenv.anet.network.vpc.segments.private
dns:
  - domain_name: database.test.internal
    hosted_zone: paco.ref netenv.mynet.network.vpc.private_hosted_zone
enable_kms_encryption: true
cluster_event_notifications:
  groups:
    - wb_low
  event_categories:
    - failover
    - failure
    - notification
default_instance:
  parameter_group: paco.ref netenv.mynet.applications.app.groups.web.resources.
↪dbparams_performance
  enable_performance_insights: true
  publicly_accessible: false
  db_instance_type: db.t3.medium
  allow_major_version_upgrade: true
  auto_minor_version_upgrade: true
  event_notifications:
    groups:
      - admin
    event_categories:
      - availability
      - configuration change
```

(continues on next page)

(continued from previous page)

```
      - deletion
      - failover
      - failure
      - maintenance
      - notification
      - recovery
  monitoring:
    enabled: true
    alarm_sets:
      basic_dbinstance:
db_instances:
  first:
    db_instance_type: db.t3.medium
    enhanced_monitoring_interval_in_seconds: 30
    availability_zone: 1
    monitoring:
      enabled: true
      alarm_sets:
        complex_dbinstance:
  second:
    enable_performance_insights: false
    event_notifications:
      groups:
        - admin
      event_categories:
        - maintenance
```

RDSMysql

RDS for MySQL

Table 241: *RDSMysql*

Field name	Type	Purpose	Constraints	Default

Base Schemas *RDSInstance*, *RDS*, Resource, DNSEnablable, Deployable, Monitorable, *RDSMultiAZ*, Named, Title, Type

RDSPostgresql

RDS for Postgresql

Table 242: *RDSPostgresql*

Field name	Type	Purpose	Constraints	Default

Base Schemas *RDSInstance*, *RDS*, Resource, DNSEnablable, Deployable, Monitorable, *RDSMultiAZ*, Named, Title, Type

RDSPostgresqlAurora

RDS PostgreSQL Aurora Cluster

Table 243: *RDSPostgresqlAurora*

Field name	Type	Purpose	Constraints	Default
database_name	String	Database Name to create in the cluster	Must be a valid database name for the DB Engine. Must contain 1 to 63 letters, numbers or underscores. Must begin with a letter or an underscore. Can't be PostgreSQL reserved word.	

Base Schemas *RDSAurora*, *RDS*, Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

RDSMysqlAurora

RDS MySQL Aurora Cluster

Table 244: *RDSMysqlAurora*

Field name	Type	Purpose	Constraints	Default
database_name	String	Database Name to create in the cluster	Must be a valid database name for the DB Engine. Must contain 1 to 64 letters or numbers. Can't be MySQL reserved word.	

Base Schemas *RDSAurora*, *RDS*, Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

RDSOptionConfiguration

Option groups enable and configure features that are specific to a particular DB engine.

Table 245: *RDSOptionConfiguration*

Field name	Type	Purpose	Constraints	Default
option_name	String	Option Name		
option_settings	List<*NameValuePair*>	List of option name value pairs.		
option_version	String	Option Version		
port	String	Port		

NameValuePair

A Name/Value pair to use for RDS Option Group configuration

Table 246: *NameValuePair*

Field name	Type	Purpose	Constraints	Default
name	String	Name		
value	PacoReference|String	Value	Paco Reference to Interface. String Ok.	

RDSMultiAZ

RDS with MultiAZ capabilities. When you provision a Multi-AZ DB Instance, Amazon RDS automatically creates a primary DB Instance and synchronously replicates the data to a standby instance in a different Availability Zone (AZ).

Table 247: *RDSMultiAZ*

Field name	Type	Purpose	Constraints	Default
multi_az	Boolean	Multiple Availability Zone deployment		False

Base Schemas *RDSInstance*, *RDS*, Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

RDSInstance

RDS DB Instance

Table 248: *RDSInstance*

Field name	Type	Purpose	Constraints	Default
allow_major_version_upgrade	Boolean	Allow major version upgrades		
auto_minor_version_upgrade	Boolean	Automatic minor version upgrades		
db_instance_type	String	RDS Instance Type		
license_model	String	License Model		
option_configurations	List<*RDSOptionConfiguration*>	Option Configurations		
parameter_group	PacoReference	RDS Parameter Group	Paco Reference to *DBParameterGroup*.	
publically_accessible	Boolean	Assign a Public IP address		
storage_encrypted	Boolean	Enable Storage Encryption		
storage_size_gb	Int	DB Storage Size in Gigabytes		
storage_type	String	DB Storage Type		

Base Schemas *RDS*, Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

RDSAurora

RDS Aurora DB Cluster

Table 249: *RDSAurora*

Field name	Type	Purpose	Constraints	Default
availability_zones	String	Availability Zones to launch instances in.	Must be one of all, 1, 2, 3 . . .	all
backtrack_window	Int in_seconds	Backtrack Window in seconds. Disabled when set to 0.	Maximum is 72 hours (259,200 seconds).	0
cluster_event_notifications	Object<*RDSDBClusterEventNotifications*>	Cluster Event Notifications		
cluster_parameter_group	Paco Reference	DB Cluster Parameter Group	Paco Reference to *DBClusterParameterGroup*.	
db_instances	Container<*RDSClusterInstances*>	DB Instances		
default_instance	Object<*RDSClusterDefaultInstance*>	Default DB Instance configuration		
enable_http_endpoint	Boolean	Enable an HTTP endpoint to provide a connectionless web service API for running SQL queries		False
enable_kms_encryption	Boolean	Enable KMS Key encryption. Will create one KMS-CMK key dedicated to each DBCluster.		False
engine_mode	Choice	Engine Mode	Must be one of provisioned, serverless, parallelquery, global, or multimaster.	
read_dns	List<*DNS*>	DNS domains to create to resolve to the connection Read Endpoint		
restore_type	Choice	Restore Type	Must be one of full-copy or copy-on-write	full-copy
use_latest_restorable_time	Boolean	Restore the DB cluster to the latest restorable backup time		False

Base Schemas *RDS*, Resource, DNSEnablable, Deployable, Monitorable, Named, Title, Type

RDSDBInstanceEventNotifications

DB Instance Event Notifications

Table 250: *RDSDBInstanceEventNotifications*

Field name	Type	Purpose	Constraints	Default
event_categories	Choice	Event Categories		
groups	List<String>	Groups		

Base Schemas Named, Title

RDSClusterDefaultInstance

Default configuration for a DB Instance that belongs to a DB Cluster.

Table 251: *RDSClusterDefaultInstance*

Field name	Type	Purpose	Constraints	Default
allow_major_version_upgrade	Boolean	Allow major version upgrades		
auto_minor_version_upgrade	Boolean	Automatic minor version upgrades		
availability_zone	Int	Availability Zone where the instance will be provisioned.	Must be one of 1, 2, 3 ...	
db_instance_type	String	DB Instance Type		
enable_performance_insights	Boolean	Enable Performance Insights		False
enhanced_monitoring_interval_in_seconds	Int	Enhanced Monitoring interval in seconds. This will enable enhanced monitoring unless set to 0.	Must be one of 0, 1, 5, 10, 15, 30, 60.	0
event_notifications	Object<*RDSDBInstanceEventNotification*>	DB Instance Event Notifications		
parameter_group	PacoReference	DB Parameter Group	Paco Reference to *DBParameterGroup*.	
publicly_accessible	Boolean	Assign a Public IP address		False

Base Schemas Monitorable, Named, Title

RDSClusterInstance

DB Instance that belongs to a DB Cluster.

Table 252: *RDSClusterInstance*

Field name	Type	Purpose	Constraints	Default
allow_major_version_upgrade	Boolean	Allow major version upgrades		
auto_minor_version_upgrade	Boolean	Automatic minor version upgrades		
availability_zone	Int	Availability Zone where the instance will be provisioned.	Must be one of 1, 2, 3 ...	
db_instance_type	String	DB Instance Type		
enable_performance_insights	Boolean	Enable Performance Insights		
enhanced_monitoring_interval_in_seconds	Int	Enhanced Monitoring interval in seconds. This will enable enhanced monitoring unless set to 0.	Must be one of 0, 1, 5, 10, 15, 30, 60.	
event_notifications	Object<*RDSDBInstanceEventNotification*>	DB Instance Event Notifications		
parameter_group	PacoReference	DB Parameter Group	Paco Reference to *DBParameterGroup*.	
publicly_accessible	Boolean	Assign a Public IP address		

Base Schemas Monitorable, Named, Title

RDSClusterInstances

Container for *RDSClusterInstance* objects.

Table 253: *RDSClusterInstances* Container<RDSClusterInstances>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

RDSDBClusterEventNotifications

Event Notifications for a DB Cluster

Table 254: *RDSDBClusterEventNotifications*

Field name	Type	Purpose	Constraints	Default
event_categories	Choice	Event Categories		
groups	List<String>	Groups		

Base Schemas Named, Title

DBParameters

If you want to use DB Parameter Groups with your RDS, then use the `parameter_group` field to reference a *DBParameterGroup* resource. Keeping DB Parameter Groups as separate resources allows having multiple Paramater Groups provisioned at the same time. For example, you might have both resources for `dbparams_performance` and `dbparams_debug`, allowing you to use the AWS Console to switch between performance and debug configuration quickl in an emergency.

DBParameterGroup

DBParameterGroup

Table 255: *DBParameterGroup*

Field name	Type	Purpose	Constraints	Default
description	String	Description		
family	String	Database Family		
parameters	Container<*DBParameters*>	Database Parameter set		

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

DBClusterParameterGroup

DBCluster Parameter Group

Table 256: *DBClusterParameterGroup*

Field name	Type	Purpose	Constraints	Default

Base Schemas Resource, *DBParameterGroup*, DNSEnablable, Deployable, Named, Title, Type

6.17.28 Route53HealthCheck

Route53 Health Check

Table 257: *Route53HealthCheck*

Field name	Type	Purpose	Constraints	Default
domain_name	String	Fully Qualified Domain Name	Either this or the load_balancer field can be set but not both.	
enable_sni	Boolean	Enable SNI		False
failure_threshold	Int	Number of consecutive health checks that an endpoint must pass or fail for Amazon Route 53 to change the current status of the endpoint from unhealthy to healthy or vice versa.		3
health_check_type	String	Health Check Type	Must be one of HTTP, HTTPS or TCP	
health_checker_regions	List<String>	Health checker regions	List of AWS Region names (e.g. us-west-2) from which to make health checks.	
ip_address	PacoReference|String	IP Address	Paco Reference to *EIP*. String Ok.	
latency_graphs	Boolean	Measure latency and display CloudWatch graph in the AWS Console		False
load_balancer	PacoReference|String	Load Balancer Endpoint	Paco Reference to *LoadBalancer*. String Ok.	
match_string	String	String to match in the first 5120 bytes of the response		
port	Int	Port		80
request_interval_fast	Boolean	Fast request interval will only wait 10 seconds between each health check response instead of the standard 30		False
resource_path	String	Resource Path	String such as '/health.html'. Path should return a 2xx or 3xx. Query string parameters are allowed: '/search?query=health'	/

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

6.17.29 S3Bucket

S3Bucket is an object storage resource in the Amazon S3 service.

S3Buckets may be declared either in the global `resource/s3.yaml` file or in a network environment in as an application resource.

S3Buckets in an application context will use the same `account` and `region` as the application, although it is still possible to override this to use other accouns and regions if desired.

Prescribed Automation

`deletion_policy`: The `deletion_policy:` field supports a `delete` or `keep` values. The `delete` choice will delete all objects from the S3 Bucket if a Paco delete command is applied. Otherwise AWS will not allow you to delete an S3 Bucket that is not empty until all objects are deleted.

`resource_suffix`: The `policy` field allows you to declare S3 Bucket policies. These policies need to be restricted to the S3 Bucket resource itself. The `resource_suffix` will be prefixed with the S3 Bucket ARN and you only need to declare keys within the bucket.

Listing 46: example S3Bucket resource

```
type: S3Bucket
title: My S3 Bucket
enabled: true
order: 10
account: paco.ref accounts.data
region: us-west-2
deletion_policy: "delete"
notifications:
    lambdas:
     - paco.ref netenv.mynet.applications.app.groups.serverless.resources.mylambda
cloudfront_origin: false
external_resource: false
versioning: false
add_paco_suffix: true
policy:
  - principal:
      Service: iotanalytics.amazonaws.com
    effect: 'Allow'
    action:
      - s3:Get*
      - s3:ListBucket
      - s3:ListBucketMultipartUploads
      - s3:ListMultipartUploadParts
    resource_suffix:
      - '/*'
      - ''
    condition:
      StringEquals:
        s3:x-amz-acl:
          "public-read"
      IpAddress:
        "aws:SourceIp": "192.0.2.0/24"
      NotIpAddress:
        "aws:SourceIp": "192.0.2.188/32"
```

(continues on next page)

(continued from previous page)

```
  - aws:
      - paco.sub '${paco.ref netenv.mynet.applications.app.groups.site.resources.demo.
↪instance_iam_role.arn}'
    effect: 'Allow'
    action:
      - 's3:Get*'
      - 's3:List*'
    resource_suffix:
      - '/*'
      - ''
```

Table 258: *S3Bucket*

Field name	Type	Purpose	Constraints	Default
account	PacoReference	Account that S3 Bucket belongs to.	Paco Reference to Account.	
add_paco_suffix	Boolean	Add the Paco s3bucket_hash suffix to the bucket name		False
bucket_name	String	Bucket Name	A short unique name to assign the bucket.	bucket
cloudfront_origin	Boolean	Creates and listens for a CloudFront Access Origin Identity		False
deletion_policy	String	Bucket Deletion Policy		delete
external_resource	Boolean	Boolean indicating whether the S3 Bucket already exists or not		False
notifications	Object<*S3NotificationConfiguration*>	Notification configuration		
policy	List<*S3BucketPolicy*>	List of S3 Bucket Policies		
region	String	Bucket region		
static_website_hosting	Object<*S3StaticWebsiteHosting*>	Static website hosting configuration.		
versioning	Boolean	Enable Versioning on the bucket.		False

Base Schemas Resource, DNSEnablable, Deployable, Named, Title, Type

S3BucketPolicy

S3 Bucket Policy

Table 259: *S3BucketPolicy*

Field name	Type	Purpose	Constraints	Default
action	List<String>	List of Actions		
aws	List<String>	List of AWS Principals.	Either this field or the principal field must be set.	
condition	Dict	Condition	Each Key is the Condition name and the Value must be a dictionary of request filters. e.g. { "StringEquals" : { "aws:username" : "johndoe" }}	{}
effect	Choice	Effect	Must be one of 'Allow' or 'Deny'	
principal	Dict	Prinicpals	Either this field or the aws field must be set. Key should be one of: AWS, Federated, Service or CanonicalUser. Value can be either a String or a List.	{}
resource_suffix	List<String>	List of AWS Resources Suffixes		
sid	String	Statement Id		

S3LambdaConfiguration

Table 260: *S3LambdaConfiguration*

Field name	Type	Purpose	Constraints	Default
event	String	S3 bucket event for which to invoke the AWS Lambda function	Must be a supported event type: https://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html	
function	PacoReference	Lambda function to notify	Paco Reference to *Lambda*.	

S3NotificationConfiguration

Table 261: *S3NotificationConfiguration*

Field name	Type	Purpose	Constraints	Default
lambdas	List<*S3LambdaConfiguration*>	Lambda configurations		

S3StaticWebsiteHosting

Table 262: *S3StaticWebsiteHosting*

Field name	Type	Purpose	Constraints	Default
redirect_requests	Object<*S3StaticWebsiteHostingRedirectRequests*>	Redirect requests configuration.		

Base Schemas Deployable

S3StaticWebsiteHostingRedirectRequests

Table 263: *S3StaticWebsiteHostingRedirectRequests*

Field name	Type	Purpose	Constraints	Default
protocol	String	Protocol		
target	PacoReference|String	Target S3 Bucket or domain.	Paco Reference to *S3Bucket*. String Ok.	

6.17.30 SNSTopic

Simple Notification Service (SNS) Topic resource.

Prescribed Automation

`cross_account_access`: Creates an SNS Topic Policy which will grant all of the AWS Accounts in this Paco Project access to the `sns.Publish` permission for this SNS Topic.

Listing 47: Example SNSTopic resource YAML

```
type: SNSTopic
order: 1
enabled: true
display_name: "Waterbear Cloud AWS"
cross_account_access: true
subscriptions:
  - endpoint: http://example.com/yes
    protocol: http
  - endpoint: https://example.com/orno
    protocol: https
  - endpoint: bob@example.com
    protocol: email
  - endpoint: bob@example.com
```

(continues on next page)

(continued from previous page)

```
    protocol: email-json
    filter_policy: '{"State": [ { "anything-but": "COMPLETED" } ] }'
  - endpoint: '555-555-5555'
    protocol: sms
  - endpoint: arn:aws:sqs:us-east-2:444455556666:queue1
    protocol: sqs
  - endpoint: arn:aws:sqs:us-east-2:444455556666:queue1
    protocol: application
  - endpoint: arn:aws:lambda:us-east-1:123456789012:function:my-function
    protocol: lambda
```

Table 264: *SNSTopic*

Field name	Type	Purpose	Constraints	Default
cross_account_access	Boolean	Cross-account access from all other accounts in this project.		False
display_name	String	Display name for SMS Messages		
locations	List<AccountRegions>	Locations	Only applies to a global SNS Topic	[]
subscriptions	List<*SNSTopicSubscription*>	List of SNS Topic Subscriptions		

Base Schemas Resource, DNSEnablable, Enablable, Named, Title, Type

SNSTopicSubscription

Table 265: *SNSTopicSubscription*

Field name	Type	Purpose	Constraints	Default
endpoint	PacoReference|String	SNS Topic ARN or Paco Reference	Paco Reference to *SNSTopic*. String Ok.	
filter_policy	String	Filter Policy	Must be valid JSON	
protocol	String	Notification protocol	Must be a valid SNS Topic subscription protocol: 'http', 'https', 'email', 'email-json', 'sms', 'sqs', 'application', 'lambda'.	email

6.18 Monitoring

The `monitor` directory can contain two files: `monitor/alarmsets.yaml` and `monitor/logging.yaml`. These files contain CloudWatch Alarm and CloudWatch Agent Log Source configuration. These alarms and log sources are grouped into named sets, and sets of alarms and logs can be applied to resources.

Currently only CloudWatch is supported, but it is intended in the future to support other monitoring and logging services in the future.

6.18.1 AlarmSets

Alarm Sets are defined in the file `monitor/alarmsets.yaml`.

AlarmSets are named to match a Paco Resource type, then a unique AlarmSet name.

Listing 48: Structure of an alarmets.yaml file

```
# AutoScalingGroup alarms
ASG:
    launch-health:
        GroupPendingInstances-Low:
            # alarm config here ...
        GroupPendingInstances-Critical:
            # alarm config here ...

# Application LoadBalancer alarms
LBApplication:
    instance-health:
        HealthyHostCount-Critical:
            # alarm config here ...
    response-latency:
        TargetResponseTimeP95-Low:
            # alarm config here ...
        HTTPCode_Target_4XX_Count-Low:
            # alarm config here ...
```

The base *Alarm* schema contains fields to add additional metadata to alarms. For CloudWatchAlarms, this metadata set in the AlarmDescription field as JSON:

Alarms can have different contexts, which increases the number of metadata that is populated in the AlarmDescription field:

- Global context. Only has base context. e.g. a CloudTrail log alarm.
- NetworkEnvironmnet context. Base and NetworkEnvironment context. e.g. a VPC flow log alarm.
- Application context alarm. Base, NetworkEnvironment and Application contexts. e,g, an external HTTP health check alarm
- Resource context alarm. Base, NetworkEnvironment, Application and Resource contexts. e.g. an AutoScalingGroup CPU alarm

```
Base context for all alarms
---------------------------

"project_name": Project name
"project_title": Project title
"account_name": Account name
"alarm_name": Alarm name
"classification": Classification
"severity": Severity
"topic_arns": SNS Topic ARN subscriptions
"description": Description (only if supplied)
"runbook_url": Runbook URL (only if supplied)
```

(continues on next page)

(continued from previous page)

```
NetworkEnvironment context alarms
---------------------------------

"netenv_name": NetworkEnvironment name
"netenv_title": NetworkEnvironment title
"env_name": Environment name
"env_title": Environment title
"envreg_name": EnvironmentRegion name
"envreg_title": EnvironmentRegion title

Application context alarms
--------------------------

"app_name": Application name
"app_title": Application title

 Resource context alarms
 -----------------------

"resource_group_name": Resource Group name
"resource_group_title": Resource Group title
"resource_name": Resource name
"resource_title": Resource title
```

Alarms can be set in the `monitoring:` field for Application and Resource objects. The name of each *AlarmSet* should be listed in the `alarm_sets:` field. It is possible to override the individual fields of an Alarm in a netenv file.

Listing 49: Examples of adding AlarmSets to Environmnets

```
environments:
  prod:
    title: "Production"
    default:
      enabled: true
      applications:
        app:
          monitoring:
            enabled: true
            alarm_sets:
              special-app-alarms:
          groups:
            site:
              resources:
                alb:
                  monitoring:
                    enabled: true
                    alarm_sets:
                      core:
                      performance:
                        # Override the SlowTargetResponseTime Alarm threshold field
                        SlowTargetResponseTime:
                          threshold: 2.0
```

Stylistically, `monitoring` and `alarm_sets` can be specified in the base `applications:` section in a netenv file, and set to `enabled: false`. Then only the production environment can override the enabled field to true.

This makes it easy to enable a dev or test environment if you want to test alarms before using in a production environment.

Alternatively, you may wish to only specify the monitoring in the `environments:` section of your netenv file only for production, and keep the base `applications:` configuration shorter.

Alarm notifications tell alarms which SNS Topics to notify. Alarm notifications are set with the `notifications:` field at the Application, Resource, *AlarmSet* and *Alarm* level.

Listing 50: Examples of Alarm notifications

```
applications:
  app:
    enabled: true
    # Application level notifications
    notifications:
      ops_team:
        groups:
        - cloud_ops
    groups:
      site:
        resources:
          web:
            monitoring:
              # Resource level notifications
              notifications:
                web_team:
                  groups:
                  - web
              alarm_sets:
                instance-health-cwagent:
                  notifications:
                    # AlarmSet notifications
                    alarmsetnotif:
                      groups:
                      - misterteam
                  SwapPercent-Low:
                    # Alarm level notifications
                    notifications:
                      singlealarm:
                        groups:
                        - oneguygetsthis
```

Notifications can be filtered for specific `severity` and `classification` levels. This allows you to direct critical severity to one group and low severity to another, or to send only performance classification alarms to one group and security classification alarms to another.

Listing 51: Examples of severity and classification filters

```
notifications:
  severe_security:
    groups:
    - security_group
    severity: 'critical'
    classification: 'security'
```

Note that although you can configure multiple SNS Topics to subscribe to a single alarm, CloudWatch has a maximum limit of five SNS Topics that a given alarm may be subscribed to.

It is also possible to write a Paco add-on that overrides the default CloudWatch notifications and instead notifies a single SNS Topic. This is intended to allow you to write an add-on that directs all alarms through a single Lambda (regardless or account or region) which is then responsible for delivering or taking action on alarms.

Currently Global and NetworkEnvironment alarms are only supported through Paco add-ons.

Listing 52: Example alarmsets.yaml for Application, ALB, ASG, RDSMySQL and LogAlarms

```
App:
  special-app-alarms:
    CustomMetric:
      description: "Custom metric has been triggered."
      classification: health
      severity: low
      metric_name: "custom_metric"
      period: 86400 # 1 day
      evaluation_periods: 1
      threshold: 1
      comparison_operator: LessThanThreshold
      statistic: Average
      treat_missing_data: breaching
      namespace: 'CustomMetric'

LBApplication:
  core:
    HealthyHostCount-Critical:
      classification: health
      severity: critical
      description: "Alert if fewer than X number of backend hosts are passing health␣
↪checks"
      metric_name: "HealthyHostCount"
      dimensions:
        - name: LoadBalancer
          value: paco.ref netenv.wa.applications.ap.groups.site.resources.alb.fullname
        - name: TargetGroup
          value: paco.ref netenv.wa.applications.ap.groups.site.resources.alb.target_
↪groups.ap.fullname
      period: 60
      evaluation_periods: 5
      statistic: Minimum
      threshold: 1
      comparison_operator: LessThanThreshold
      treat_missing_data: breaching
  performance:
    SlowTargetResponseTime:
      severity: low
      classification: performance
      description: "Average HTTP response time is unusually slow"
      metric_name: "TargetResponseTime"
      period: 60
      evaluation_periods: 5
      statistic: Average
      threshold: 1.5
      comparison_operator: GreaterThanOrEqualToThreshold
      treat_missing_data: missing
      dimensions:
        - name: LoadBalancer
```

(continues on next page)

(continued from previous page)

```
          value: paco.ref netenv.wa.applications.ap.groups.site.resources.alb.fullname
        - name: TargetGroup
          value: paco.ref netenv.wa.applications.ap.groups.site.resources.alb.target_
↪groups.ap.fullname
    HTTPCode4XXCount:
      classification: performance
      severity: low
      description: "Large number of 4xx HTTP error codes"
      metric_name: "HTTPCode_Target_4XX_Count"
      period: 60
      evaluation_periods: 5
      statistic: Sum
      threshold: 100
      comparison_operator: GreaterThanOrEqualToThreshold
      treat_missing_data: notBreaching
    HTTPCode5XXCount:
      classification: performance
      severity: low
      description: "Large number of 5xx HTTP error codes"
      metric_name: "HTTPCode_Target_5XX_Count"
      period: 60
      evaluation_periods: 5
      statistic: Sum
      threshold: 100
      comparison_operator: GreaterThanOrEqualToThreshold
      treat_missing_data: notBreaching

ASG:
  core:
    StatusCheck:
      classification: health
      severity: critical
      metric_name: "StatusCheckFailed"
      namespace: AWS/EC2
      period: 60
      evaluation_periods: 5
      statistic: Maximum
      threshold: 0
      comparison_operator: GreaterThanThreshold
      treat_missing_data: breaching
    CPUTotal:
      classification: performance
      severity: critical
      metric_name: "CPUUtilization"
      namespace: AWS/EC2
      period: 60
      evaluation_periods: 30
      threshold: 90
      statistic: Average
      treat_missing_data: breaching
      comparison_operator: GreaterThanThreshold
  cwagent:
    SwapPercentLow:
      classification: performance
      severity: low
      metric_name: "swap_used_percent"
      namespace: "CWAgent"
```

(continues on next page)

(continued from previous page)

```
      period: 60
      evaluation_periods: 5
      statistic: Maximum
      threshold: 80
      comparison_operator: GreaterThanThreshold
      treat_missing_data: breaching
    DiskSpaceLow:
      classification: health
      severity: low
      metric_name: "disk_used_percent"
      namespace: "CWAgent"
      period: 300
      evaluation_periods: 1
      statistic: Minimum
      threshold: 60
      comparison_operator: GreaterThanThreshold
      treat_missing_data: breaching
    DiskSpaceCritical:
      classification: health
      severity: low
      metric_name: "disk_used_percent"
      namespace: "CWAgent"
      period: 300
      evaluation_periods: 1
      statistic: Minimum
      threshold: 80
      comparison_operator: GreaterThanThreshold
      treat_missing_data: breaching

  # CloudWatch Log Alarms
  log-alarms:
    CfnInitError:
      type: LogAlarm
      description: "CloudFormation Init Errors"
      classification: health
      severity: critical
      log_set_name: 'cloud'
      log_group_name: 'cfn_init'
      metric_name: "CfnInitErrorMetric"
      period: 300
      evaluation_periods: 1
      threshold: 1.0
      treat_missing_data: notBreaching
      comparison_operator: GreaterThanOrEqualToThreshold
      statistic: Sum
    CodeDeployError:
      type: LogAlarm
      description: "CodeDeploy Errors"
      classification: health
      severity: critical
      log_set_name: 'cloud'
      log_group_name: 'codedeploy'
      metric_name: "CodeDeployErrorMetric"
      period: 300
      evaluation_periods: 1
      threshold: 1.0
      treat_missing_data: notBreaching
```

(continues on next page)

(continued from previous page)

```
      comparison_operator: GreaterThanOrEqualToThreshold
      statistic: Sum
    WsgiError:
      type: LogAlarm
      description: "HTTP WSGI Errors"
      classification: health
      severity: critical
      log_set_name: 'ap'
      log_group_name: 'httpd_error'
      metric_name: "WsgiErrorMetric"
      period: 300
      evaluation_periods: 1
      threshold: 1.0
      treat_missing_data: notBreaching
      comparison_operator: GreaterThanOrEqualToThreshold
      statistic: Sum
    HighHTTPTraffic:
      type: LogAlarm
      description: "High number of http access logs"
      classification: performance
      severity: low
      log_set_name: 'ap'
      log_group_name: 'httpd_access'
      metric_name: "HttpdLogCountMetric"
      period: 300
      evaluation_periods: 1
      threshold: 1000
      treat_missing_data: ignore
      comparison_operator: GreaterThanOrEqualToThreshold
      statistic: Sum

RDSMysql:
  basic-database:
    CPUTotal-Low:
      classification: performance
      severity: low
      metric_name: "CPUUtilization"
      namespace: AWS/RDS
      period: 300
      evaluation_periods: 6
      threshold: 90
      comparison_operator: GreaterThanOrEqualToThreshold
      statistic: Average
      treat_missing_data: breaching

    FreeableMemoryAlarm:
      classification: performance
      severity: low
      metric_name: "FreeableMemory"
      namespace: AWS/RDS
      period: 300
      evaluation_periods: 1
      threshold: 100000000
      comparison_operator: LessThanOrEqualToThreshold
      statistic: Minimum
      treat_missing_data: breaching
```

(continues on next page)

(continued from previous page)

```
    FreeStorageSpaceAlarm:
      classification: performance
      severity: low
      metric_name: "FreeStorageSpace"
      namespace: AWS/RDS
      period: 300
      evaluation_periods: 1
      threshold: 5000000000
      comparison_operator: LessThanOrEqualToThreshold
      statistic: Minimum
      treat_missing_data: breaching
```

Table 266: *AlarmSets* Container<AlarmSet>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

AlarmSet

A container of Alarm objects.

Table 267: *AlarmSet*

Field name	Type	Purpose	Constraints	Default
resource_type	String	Resource type	Must be a valid AWS resource type	

Base Schemas Named, Notifiable, Title

Alarm

A Paco Alarm.

This is a base schema which defines metadata useful to categorize an alarm.

Table 268: *Alarm*

Field name	Type	Purpose	Constraints	Default
classification	String	Classification	Must be one of: 'performance', 'security' or 'health'	unset
description	String	Description		
notification_groups	List<String>	List of notification groups the alarm is subscribed to.		
runbook_url	String	Runbook URL		
severity	String	Severity	Must be one of: 'low', 'critical'	low

Base Schemas Deployable, Named, Notifiable, Title

Dimension

A dimension of a metric

Table 269: *Dimension*

Field name	Type	Purpose	Constraints	Default
name	String	Dimension name		
value	PacoReference|String	String or a Paco Reference to resource output.	Paco Reference to Interface. String Ok.	

AlarmNotifications

Container for *AlarmNotification* objects.

Table 270: *AlarmNotifications* Container<AlarmNotification>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

AlarmNotification

Alarm Notification

Table 271: *AlarmNotification*

Field name	Type	Purpose	Constraints	Default
classification	String	Classification filter	Must be one of: 'performance', 'security', 'health' or ''.	
groups	List<String>	List of groups		
severity	String	Severity filter	Must be one of: 'low', 'critical'	

Base Schemas Named, Title

SimpleCloudWatchAlarm

A Simple CloudWatch Alarm

Table 272: *SimpleCloudWatchAlarm*

Field name	Type	Purpose	Constraints	Default
actions_enabled	Boolean	Actions Enabled		
alarm_description	String	Alarm Description	Valid JSON document with Paco fields.	
comparison_operator	String	Comparison operator	Must be one of: 'GreaterThanThreshold','GreaterThanOrEqualToThreshold', 'LessThanThreshold', 'LessThanOrEqualToThreshold'	
dimensions	List<*Dimension*>	Dimensions		
evaluation_periods	Int	Evaluation periods		
metric_name	String	Metric name		
namespace	String	Namespace		
period	Int	Period in seconds		
statistic	String	Statistic		
threshold	Float	Threshold		

MetricFilters

Container for *Metric'Filter* objects.

Table 273: *MetricFilters* Container<MetricFilter>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

MetricFilter

Metric filter

Table 274: *MetricFilter*

Field name	Type	Purpose	Constraints	Default
filter_pattern	String	Filter pattern		
metric_transformations	List<*MetricTransformation*>	Metric transformations		

Base Schemas Named, Title

MetricTransformation

Metric Transformation

Table 275: *MetricTransformation*

Field name	Type	Purpose	Constraints	Default
default_value	Float	The value to emit when a filter pattern does not match a log event.		
metric_name	String	The name of the CloudWatch Metric.		
metric_namespace	String	The namespace of the CloudWatch metric. If not set, the namespace used will be 'AIM/{log-group-name}'.		
metric_value	String	The value that is published to the CloudWatch metric.		

Metric

A set of metrics to collect and an optional collection interval:

- **name: disk** measurements: - free collection_interval: 900

Table 276: *Metric*

Field name	Type	Purpose	Constraints	Default
collection_interval	Int	Collection interval		
drop_device	Boolean	Drops the device name from disk metrics		True
measurements	List<String>	Measurements		
name	String	Metric(s) group name		
resources	List<String>	List of resources for this metric		

6.18.2 CloudWatchLogging

CloudWatch Logging configuration

Table 277: *CloudWatchLogging*

Field name	Type	Purpose	Constraints	Default
log_sets	Container<*CloudWatchLogSets*>	A CloudWatchLogSets container		

Base Schemas *CloudWatchLogRetention*, Named, Title

CloudWatchLogRetention

Table 278: *CloudWatchLogRetention*

Field name	Type	Purpose	Constraints	Default
expire_events_after_days	String	Expire Events After. Retention period of logs in this group		

CloudWatchLogSets

Container for *CloudWatchLogSet* objects.

Table 279: *CloudWatchLogSets* Container<CloudWatchLogSet>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CloudWatchLogSet

A set of Log Group objects

Table 280: *CloudWatchLogSet*

Field name	Type	Purpose	Constraints	Default
log_groups	Container<*CloudWatchLogGroups*>	A CloudWatchLogGroups container		

Base Schemas *CloudWatchLogRetention*, Named, Title

CloudWatchLogGroups

Container for *CloudWatchLogGroup* objects.

Table 281: *CloudWatchLogGroups* Container<CloudWatchLogGroup>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CloudWatchLogGroup

A CloudWatchLogGroup is responsible for retention, access control and metric filters

Table 282: *CloudWatchLogGroup*

Field name	Type	Purpose	Constraints	Default
log_group_name	String	Log group name. Can override the LogGroup name used from the name field.		
metric_filters	Container<*MetricFilters*>	Metric Filters		
sources	Container<*CloudWatchLogSources*>	A CloudWatchLogSources container		

Base Schemas *CloudWatchLogRetention*, Named, Title

CloudWatchLogSources

A container of *CloudWatchLogSource* objects.

Table 283: *CloudWatchLogSources* Container<CloudWatchLogSource>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

CloudWatchLogSource

Log source for a CloudWatch agent.

Table 284: *CloudWatchLogSource*

Field name	Type	Purpose	Constraints	Default
encoding	String	Encoding		utf-8
log_stream_name	String	Log stream name	CloudWatch Log Stream name	
multi_line_start_pattern	String	Multi-line start pattern		
path	String	Path	Must be a valid filesystem path expression. Wildcard * is allowed.	
timestamp_format	String	Timestamp format		
timezone	String	Timezone	Must be one of: 'Local', 'UTC'	Local

Base Schemas *CloudWatchLogRetention*, Named, Title

6.18.3 HealthChecks

Container for Route53HealthCheck objects.

Table 285: *HealthChecks* Container<Route53HealthCheck>

Field name	Type	Purpose	Constraints	Default

Base Schemas Named, Title

6.19 Extending Paco with Services

Paco has an add-on feature called **Services**.

A **Paco Service** is a Python module that is loaded during Paco initialization and is capable of extending or changing Paco in any way.

Services commonly provision cloud resources. For example, if you wanted to send CloudWatch Alarm notifications to a Slack Channel, you would need to send your Alarm messages to a custom Lambda. A Slack Service could provision this custom Lambda and customize your AlarmActions to send to messages this Lambda.

Services that provision resources have the PACO_SCOPE `service.<servicename>`:

```
$ paco validate service.slack
$ paco provision service.slack
$ paco delete service.slack
```

6.19.1 Creating a minimal Paco Service

The minimal requirments to create a Paco Service is to create a Python project and a `service/<my-name>.yaml` file in a Paco Project that will use that service.

Let's take a look at what's involved in a Paco Service that prints "Hello World" during Paco initialization.

First, create a `mypacoaddon` directory for your Paco Service and make it a Python project by creating a `setup.py` file. This file describes the layout of your Python project.

```
from setuptools import setup

setup(
    name='helloworld-service',
    version='0.1.dev',
    install_requires=['paco-cloud'],
    entry_points = {
        'paco.services': [
            'helloworld = mypacoaddon.helloworld',
        ],
    },
    packages=['mypacoaddon',],
    package_dir={'': 'src'},
)
```

The `setup.py` is described in standard Python packaging. The important parts to note are that the Paco Service should declare it depends on the `paco-cloud` Python project in the `install_requires` field.

The `entry_points` field will register `paco.services` entry points. You can register more than one Paco Service here. Each Paco Service declared is in the format `<service-name> = <python-dotted-name-of-module>`.

The Paco Service service name must be unique within the Services your Paco has installed.

A Python module that is a Paco Service **must** provide two functions:

```
def instantiate_model(config, project, monitor_config, read_file_path):
    "Return a Python object with configuration for this Service"
    pass

def instantiate_class(paco_ctx, config):
    "Return a Controller for this Service"
    pass
```

The `load_service_model` function is called during model loading. It could return any empty Python object, it could use `paco.mdoel` laoding APIs to read and validate custom YAML configuration or do any other kind of custom configuration initialization and loading you need.

The `get_service_controller` function is called during Controller initialization and it needs to return a Paco Controller specific to your Paco Service.

In your `mypacoaddon` project, create the following directory structure:

```
mypacoaddon/
  setup.py
  src/
    mypacoaddon/
      __init__.py
      helloworld.py
```

Then put this code into `helloworld.py`:

```
"""
Hello World Paco Service
"""

# Hook into the Paco Service loading

def load_service_model(config, project, monitor_config, read_file_path):
    return HelloWorldModel()

def get_service_controller(paco_ctx, config):
    "Return a HelloWorld controller for the HelloWorld Service"
    return HelloWorldController(config)

# Model and Controller

class HelloWorldModel:
    speak = "Hello World!"

class HelloWorldController:

    def __init__(self, config):
        self.config = config

    def init(self, command=None, model_obj=None):
        print(self.config.speak)
```

Next you can install your Python project from the `mypacoaddon` directory with the command `pip install -e .`. This will register your Paco Service entry point in your for your Python environment.

By default, if you run Paco commands on a Paco Project, if there is no file for your Paco Service in the `services/` directory, then Paco will not load that Paco Service. This is by design to allow you to install a Paco Service but only use it in Paco Projects that you explicitly declare.

In a Paco Project, create a file `services/helloworld.yaml`. This can be an empty file or valid YAML that will be read into a Python data structure and passed as the argument `config` to your `load_service_model` function.

Now run any Paco command and you should see "Hello World!" printed on your terminal.

```
$ paco validate netenv.mynet.staging
Loading Paco project: /Users/example/my-paco-project
Hello World!
...
```

6.19.2 Service module specification

Every Paco Service Python module **must** have `load_service_model` and `get_service_controller` functions. These will be called when the Service is initialized. In addition, the module may optionally provide a `SERVICE_INITIALIZATION_ORDER` attribute.

```
"""
Example barebones Paco Service module
"""

# Every Paco Service *must*  provide these two functions
def load_service_model(config, project, monitor_config, read_file_path):
    pass

def get_service_controller(paco_ctx, config):
    pass

# Optional attribute
SERVICE_INITIALIZATION_ORDER = 1000
```

load_service_model

This required function loads the configuration YAML into model objects for your Service. However, it isn't required for a Service to have any model and this method can simply return None.

If a Paco Project doesn't have a `service/<servicename>.yaml` file, then that service is not considered active in that Paco Project and will **NOT** be enabled. The configuration file for a Service must be valid YAML or an empty file.

The `load_service_model` must accept four arguments:

- `config`: A Python dict of the Services `service/<servicename>.yaml` file.
- `project`: The root Paco Project model object.
- `monitor_config`: A Python dict of the YAML loaded from config in the‘‘monitor/‘‘ directory.
- `read_file_path`: The location of the file path of the Service's YAML file.

```
class Notification:
    pass

def load_service_model(config, project, monitor_config, read_file_path):
    "Loads services/notification.yaml and returns a Notification model object"
    return Notification()
```

get_service_controller

This required function must return a Controller object for your Service.

The `get_service_controller` must accept two arguments:

- `paco_ctx`: The PacoContext object contains the CLI arguments used to call Paco as well as other global information.
- `service_model`: The model object returned from this Service's `load_service_model` function.

A Controller **must** provide an `init(self, command=None, model_obj=None)` method. If the Service can be provisioned, it must also implement `validate(self)`, `provision(self)` and `delete(self)` methods.

```
class NotificationServiceController:
    def init(self, command=None, model_obj=None):
        pass

def get_service_controller(paco_ctx, service_model):
    "Return a Paco controller"
    return NotificationServiceController()
```

SERVICE_INITIALIZATION_ORDER

The `SERVICE_INITIALIZATION_ORDER` attribute determines the initialization order of Services. This is useful for Services that need to do special initialization before other Services are initialized.

If this order is not declared the initialization order will be randomly assigned an order starting from 1000.

Overview of Paco Initialization

Every time Paco loads a Paco Project, it starts by determing which Services are installed and actived. Configuration for Services is in the `service/` directory of a Paco Project. If a file exists at `service/<service-name>.yaml` than that Service will be active in that Paco Project. If a Service is installed with Paco but there is no service file, it is ignored.

All of the active Services are imported and given the chance to apply configuration that extends Paco.

Next, Paco reads all of the YAML files in the Paco Project and creates a Python object model from that YAML configuration.

Then Paco will initialize the Controllers that it needs. Controllers are high level APIs that implement Paco commands. Controllers can govern the creating, updating and deletion of cloud resources, typically by acting on the contents of the Paco model.

Controller initialization happens in a specific order:

1. Controllers for Global Resources declared in the `resource/` directory are initialized first. This allows other Controllers to depend upon global resources being already initialized and available.
2. Service Controllers declared in the `service/` are initialized second. They are initialized in an `initialization_order` that each Service add-on may declare. Controllers with a low *initialization_order* have a chance to make changes that effect the initialization of later Controllers.
3. The Controller specific to the current PACO_SCOPE is initialized last. For example, if the command *paco provision netenv.mynet.staging* was run, the scope is a NetworkEnvironment and a NetworkEnvironment Controller will be initialized.

6.19.3 Paco Extend API

The `paco.extend` module contains convenience APIs to make it easier to extend Paco. These APIs will be typically called from your custom Paco Service Controllers.

`paco.extend.`**`add_cw_alarm_hook`**`(`*`hook`*`)`

Customize CloudWatchAlarm with a hook that is called before the Alarms are initialized into CloudFormation templates.

This is useful to add extra metadata to the CloudWatch Alarm's AlarmDescription field. This can be done in the hook by calling the `add_to_alarm_description` method of the cw_alarm object with a dict of extra metadata.

```
import paco.extend

def my_service_alarm_description_function(cw_alarm):
    slack_metadata = {'SlackChannel': 'http://my-slack-webhook.url'}
    cw_alarm.add_to_alarm_description(slack_metadata)

paco.extend.add_cw_alarm_hook(my_service_alarm_description_function)
```

paco.extend.**add_extend_model_hook**(*extend_hook*)

Add a hook can extend the core Paco schemas and models. This hook is called first during model loading before any loading happens.

```
from paco.models import schemas
from paco.models.metrics import AlarmNotification
from zope.interface import Interface, classImplements
from zope.schema.fieldproperty import FieldProperty
from zope import schema

class ISlackChannelNotification(Interface):
    slack_channels = schema.List(
        title="Slack Channels",
        value_type=schema.TextLine(
            title="Slack Channel",
            required=False,
        ),
        required=False,
    )

def add_slack_model_hook():
    "Add an ISlackChannelNotification schema to AlarmNotification"
    classImplements(AlarmNotification, ISlackChannelNotification)
    AlarmNotification.slack_channels = FieldProperty(ISlackChannelNotification[
↪"slack_channels"])

paco.extend.add_extend_model_hook(add_slack_model_hook)
```

paco.extend.**add_security_groups_hook**(*hook*)

Customize SecurityGroup lists with a hook that is called before security groups are set on: LoadBalancers. TODO: EC2, etc

```
import paco.extend

def my_service_security_groups_function():
    security_group_list = []
    ...
    return security_group_list

paco.extend.add_security_groups_hook(my_service_security_groups_function)
```

paco.extend.**load_app_in_account_region**(*parent*, *account*, *region*, *app_name*, *app_config*, *project=None*, *monitor_config=None*, *read_file_path='not set'*)

Load an Application from config into an AccountContainer and RegionContainer. Account can be a paco.ref but then the Paco Project must be supplied too.

paco.extend.**load_package_yaml**(*package*, *filename*, *replacements={}*)

Read a YAML file from the same directory as a Python package and parse the YAML into Python data structures.

paco.extend.**override_codestar_notification_rule**(*hook*)
Add a hook to change CodeStar Notification Rule's to your own custom list of SNS Topics. This can be used to send notifications to notify your own custom Lambda function instead of sending directly to the SNS Topics that Alarms are subscribed too.

```
def override_codestar_notification_rule(snstopics, alarm):
    "Override normal alarm actions with the SNS Topic ARN for the custom␣
↪Notification Lambda"
    return ["paco.ref service.notify...snstopic.arn"]

paco.extend.add_codestart_notification_rule_hook(override_codestar_notification_
↪rule)
```

paco.extend.**override_cw_alarm_actions**(*hook*)
Add a hook to change CloudWatch Alarm AlarmAction's to your own custom list of SNS Topics. This can be used to send AlarmActions to notify your own custom Lambda function instead of sending Alarm messages directly to the SNS Topics that Alarms are subscribed too.

The hook is a function that accepts an `alarm` arguments and must return a List of paco.refs to SNS Topic ARNs.

```
def override_alarm_actions_hook(snstopics, alarm):
    "Override normal alarm actions with the SNS Topic ARN for the custom␣
↪Notification Lambda"
    return ["paco.ref service.notify...snstopic.arn"]

paco.extend.override_cw_alarm_actions(override_alarm_actions_hook)
```

paco.extend.**override_cw_alarm_description**(*hook*)
Add a hook to modify the CloudWatch Alarm Description.

The description is sent as a dictionary of fields that will be stored in the AlarmDescription as json.

The hook is a function that accepts an `alarm` and `description` arguments and must return the description when done.

```
def override_cw_alarm_description(alarm, description):
    "Override alarm description"
    return description

paco.extend.override_cw_alarm_description(override_alarm_description_hook)
```

paco.extend.**register_model_loader**(*obj*, *fields_dict*, *force=False*)
Register a new object to the model loader registry.

The `obj` is the object to register loadable fields for.

The `fields_dict` is a dictionary in the form:

```
{ '<fieldname>': ('loader_type', loader_args) }
```

If an object is already registered, an error will be raised unless `force=True` is passed. Then the new registry will override any existing one.

For example, to register a new `Notification` object with `slack_channels` and `admins` fields:

```
paco.extend.register_model_loader(
    Notification, {
        'slack_channels': ('container', (SlackChannels, SlackChannel))
```

(continues on next page)

(continued from previous page)

```
        'admins': ('obj_list', Administrator)
    }
)
```

6.20 Paco Internals

Discussions on what Paco does under the hood.

6.20.1 Paco Architecture

What happens when you run "paco provision" to create cloud resources?

Paco will go through the following steps:

1. PacoContext: Command line arguments are read and parsed. An object of the class `paco.config.paco_context.PacoContext` is created which holds the command-line arguments. Most notably, this object will have a .home attribute, which is the path to the Paco project and a .project attribute which will contain that project loaded as a model.
2. Load the project config: After the home directory is set on PacoContext, then `paco_ctx.load_project()` will call `paco.models.load_project_from_yaml` with that directory. The paco.models loader will read all of the YAML files and construct Python objects. The Paco model is a tree of objects, the root node is a `Project` object. Every object in the tree has a `name` and `__parent__` attribute. This allows any object to know it's paco.ref by walking up the parents to the root node and concatenating the names.
3. AccountContext: Next and object of the class `paco.config.paco_context.AccountContext` is created for the master account. This will ask the user for their MFA credentials. AccountContext objects manage connections to the AWS accounts.
4. Global Initialization: Paco has Controllers which are objects which initialize and orchestrate the CloudFormation templates and stacks. They are also responsible for connecting the model to the stacks they create, so that resources can find the outputs that they create. Global controllers that are widely depended upon are initialized (Route53, S3 and SNS Topics). Finally once everything is almost ready, Service controllers are loaded - these are Paco Add-Ons. These are last in the process to give them a chance to react/modify the final set-up without limit.
5. Scope Initialization: Depending on the scope that is being provisioned (e.g. netenv.mynet.dev or resource.s3) a controller of the appropriate type will be looked up and initialized.
6. Perform Cloud Action: The cloud action (validate, provision, delete) is called on the controller for the scope. It is up to the controller to determine how it goes about doing that action, but most controllers follow the common pattern of iterating through their StackGroups and calling the cloud action on each StackGroup.

6.20.2 Stacks and Templates

AWS CloudFormation

Paco uses the AWS CloudFormation service to provision AWS resources and maintain resource state. CloudFormation has two core concepts: a template and a stack. A template is a CloudFormation document the declares resources. A stack is when a template is uploaded to AWS to create those resources. A stack will always belong to a specific account and region.

Paco has several Classes which it uses to model stacks and templates and control how they interact with the AWS CloudFormation service.

Controller

Controller objects initialize and set-up other objects. They create StackGroups and add Stacks to them. The can also interact with commands from the CLI.

Controllers also inject a `resolve_ref_obj` into model objects, to allow model objects to use Paco References to refer to Stack outputs.

PacoContext

The `paco.config.paco_context.PacoContext` class contains the arguments and options parsed from the CLI. PacoContext also makes a call to load a Paco project into a model and make the project root node available as a `.project` attribute.

The `.get_controller(<controller-name>)` method on PacoContext is used to fetch a controller. This ensures that controllers are initialized once and only once.

StackGroup

The `paco.stack.stack_group.StackGroup` class implements a StackGroup. A StackGroup is a logical collection of stacks that support a single concept. StackGroups apply the same operation against all Stacks that it contains and ensure that they are executed in the correct order and if necessary wait for stacks to be created if one stack depends upon the output of another stack.

StackGroups are often subclassed and the subclass adds logic to related to that subclasses purpose. For example, a BackupVault needs an IAM Role to assume. If you have a BackupVault Stack, you also need an IAM Role Stack with a Role. The BackupVaultsStackGroup adds the ability to create a Stack for that IAM Role.

Stack

The `paco.stack.Stack` class defines a Paco Stack. A Stack is connected to an account and region, and can fetch the state of the Stack as well as create, update and delete a stack in AWS.

Every Stack expects a StackTemplate object to be set on the `.template` attribute. This happens by calling `add_new_stack()` on a StackGroup. This method ensures that the Stack is created first, then the StackTemplate is created with the stack object passed in the constructor, after the new StackTemplate object is set on the `.template` attribute any commands that need to happen after are applied and the stack is given orders to the StackGroup.

Every Stack is created with a `.resource` attribute. This is the model object which contains the configuration for that Stacks template. The `IResource` interface in the models provides an `is_enabled()` method, and a `.order` and `.change_protected` attributes. This helps inform the stack if it should be modified, and in which order, or if it shouldn't be touched at all.

Every Stack as a `stack_ref` property. This is normally the paco.ref for the `.resource` but it can also be extended with a `support_resource_ref_ext` when the Stack is created. For example, an ASG resource needs a LogGroup stack where it will log to. This is a supporting resource that isn't explicitly declared in the configuration. The same happens for Alarms, which add a '.alarms' extension to the ref.

StackTemplate

The `paco.cftemplates.StackTemplate` class defines a Paco Template. A StackTemplate has a .body attribute which is a string of CloudFormation in YAML format.

A StackTemplate requires a Stack object to be passed to the constructor. In Paco, a StackTemplate can provision a CloudFormation template in several different locations and potentially look different in each of those locations. The StackTemplate has access to the Stack. The StackTemplate typically sets Parameters on the Stack. It can also change the behaviour of Stack updates, for example, certain Parameters can be set to use the previously existing value of the Stack.

A `troposphere.Template` class defines a StackTemplate's .template attribute. Troposphere is an external Python dependency of Paco. It's a great library with a complete and updated representation of CloudFormation objects. However a StackTemplatecan provide any kind of return string, so simple Python strings can also be constructed and set as the template body.

When Paco uses a StackTemplate it never instantiates it directly. It's a base class that resource specific templates inherit from. These subclasses are responsible for creating the template.

StackHooks

StackHooks are programatic actions that happen before or after a create, update or delete stack operation.

Paco uses them to upload files to an S3 Bucket after it's created in the EC2LaunchManager, to delete all files in an S3 Bucket before it's deleted, and to create and manage access keys for IAM Users.

The `paco.stack.stack.StackHooks` class should be created and have one or more hooks added to it, then passed to `StackGroup.add_new_stack` to have the hooks added to a stack, or `Stack.add_hooks` can be called after creation to have hooks after stack creation. The `Stack.add_hooks` will merge new hooks with existing ones, so several places can contribute StackHooks.

To create a hook, call `StackHooks.add()` method with:

- `name`: This will be displayed on the command-line interface.
- `stack_action`: Must be one of `create`, `update` or `delete`. The `update` action is called every time an existing stack is in scope, if the hook's `cache_method` returns a different cache id or the cache does not exist. `update` hooks should be designed to be idempotent and able to be re-run multiple times.
- `stack_timing`: Must be one of `pre` or `post`.
- `hook_method`: A method that will perform the work of the hook. It is called with two arguments: the `hook` iteslf and the `hook_arg` value.
- `cache_method`: Optional. A method that will return a cache id. If this value does not change between provisions, then the hook will be skipped. This only applies to hooks on the `update` stack action.
- `hook_arg`: Optional. A value which is supplied as an argument to the `hook_method` with it is invoked.

Listing 53: example usage of StackHooks

```
stack_hooks = StackHooks()
stack_hooks.add(
   name='UploadZipFile',
   stack_action='create',
   stack_timing='post',
   hook_method=self.upload_bundle_stack_hook,
   cache_method=self.stack_hook_cache_id,
   hook_arg=bundle
)
```

6.20.3 The .paco-work directory

Paco creates a directory in every Paco project named `.paco-work`. This directory contains several sub-directories that Paco will read/write to while it's working.

`.paco-work/applied` Paco starts a provision command by showing you a diff of the configuration from the last provision. It does this by keeping a cache of YAML configuration files after it applies them here.

Paco will also show you changes between previously CloudFormation stacks and Parameters and the new ones it wants to apply. Paco creates a cache of stacks here when after they have been applied.

If this directory gets out-of-sync then Paco can skip updates to Resrouces believing that they haven't changed. You can remedy this by using the `-n --nocahce` flag with the Paco CLI.

Alternatively, you could run `rm -rf .paco-work/applied/cloudformation` to remove this cache and Paco will simply run slower on it's next run as it fetches state from CloudFormation.

`.paco-work/build` This is a scratch space that Paco can use. For example, the EC2LaunchManager creates a zip file bundles of files used to configure EC2 instances. These zip files are created in here.

`.paco-work/outputs` Stack outputs are cached here. These outputs are organized according to the structure of the Paco model as opposed to the structure of the CloudFormation stacks.

`.paco-work/describe` When the `paco describe` command is run the generated Paco web site is output here.

6.21 Copyright and Attributions

Paco

by Kevin Lindsay and Kevin Teague

6.21.1 Attributions

Contributors: Kevin Lindsay, Kevin Teague and comments from Luda and Karen.

CHAPTER 7

Indices and tables

- *Glossary*
- genindex

7.1 Glossary

Container A key/value mapping where each value is a key/value mapping that corresponds to the Paco schema that the container is intended to hold.

Infrastructure as Code Any software project which is intended to automatically provision and manage cloud resources.

Paco project Structured directory of files and sub-directories that declaratively describe an IaC project.

Paco Schema A set of fields that describe the fields name, type and constraints.

Python Module Index

p

Index