aws-service-catalog-factory

Eamonn Faherty

Aug 09, 2023

CONTENTS:

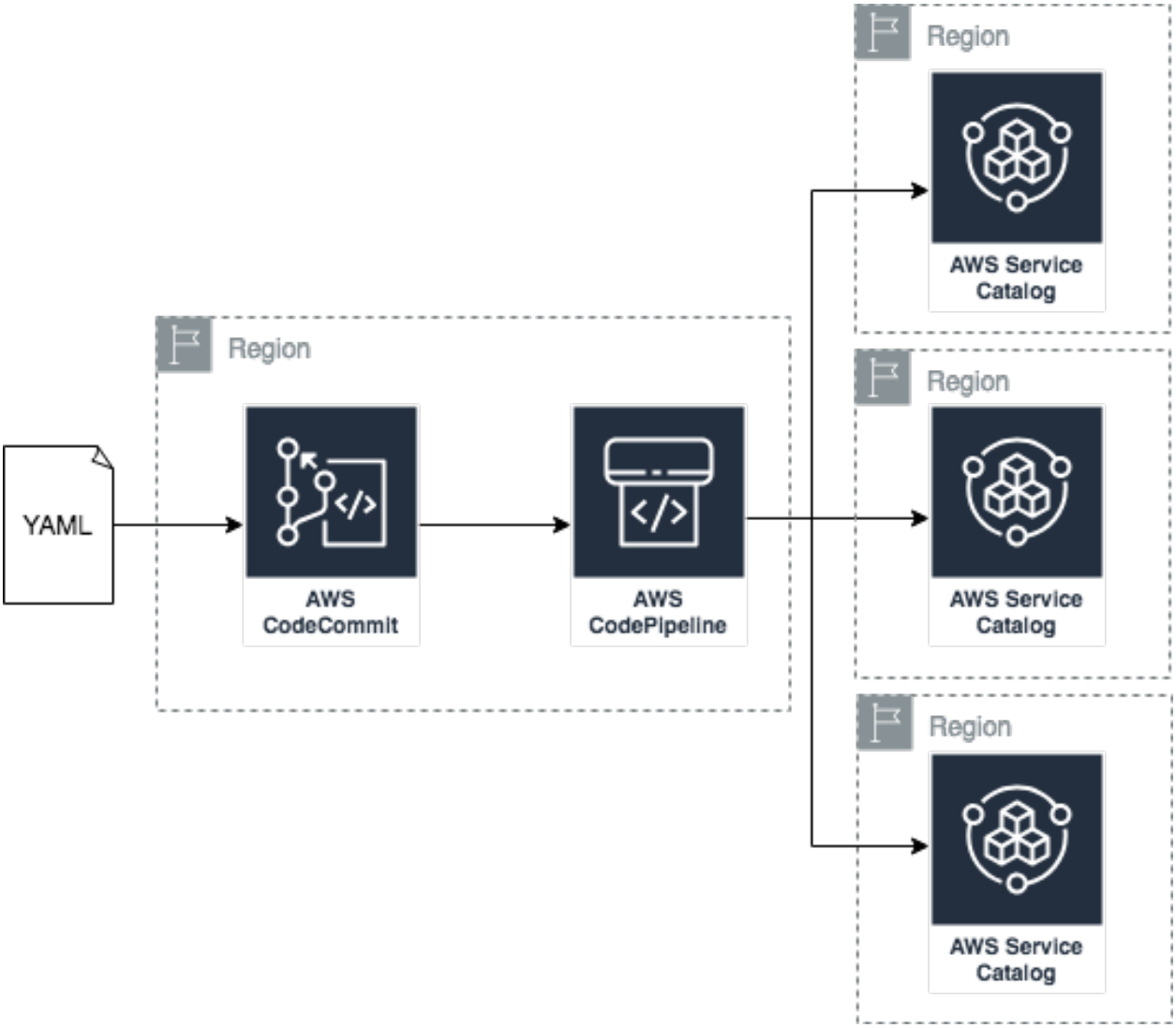
CHAPTER

ONE

WHAT IS THIS?

This is a framework where you define a Service Catalog portfolio, products and versions using YAML. For versions of your products you specify where the source code for them can be found and the framework publishes the portfolio, products and versions in every* AWS Region after validating, linting and testing them.

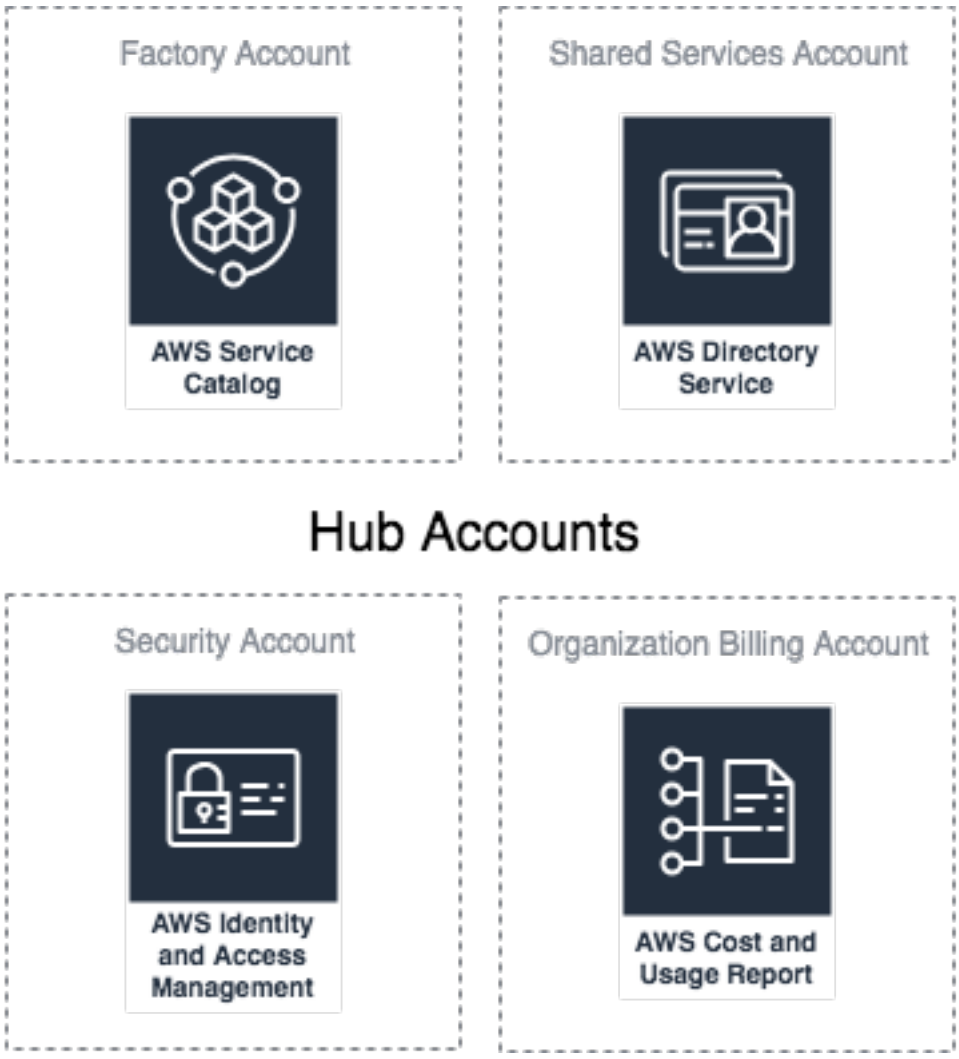
1.1 High level architecture diagram



You build products in a central hub account using AWS CodePipeline and AWS CodeBuild, you then deploy them into AWS Service Catalog in every enabled region of your hub account using AWS CodePipeline and AWS CloudFormation.

1.2 Providing your factory to multiple accounts - hub and spoke model

If you would like to share your Factory and its contents with other accounts, it is recommended you create a Factory Account within your organization. This would be considered a *hub* account. This framework can be installed into many accounts within the same AWS Organization - meaning you can have multiple factory accounts.



You can use your Factory to provide AWS Service Catalog products to your *spoke* accounts. You can even use the factory to build the content in other hub accounts (including the account hosting your factory).

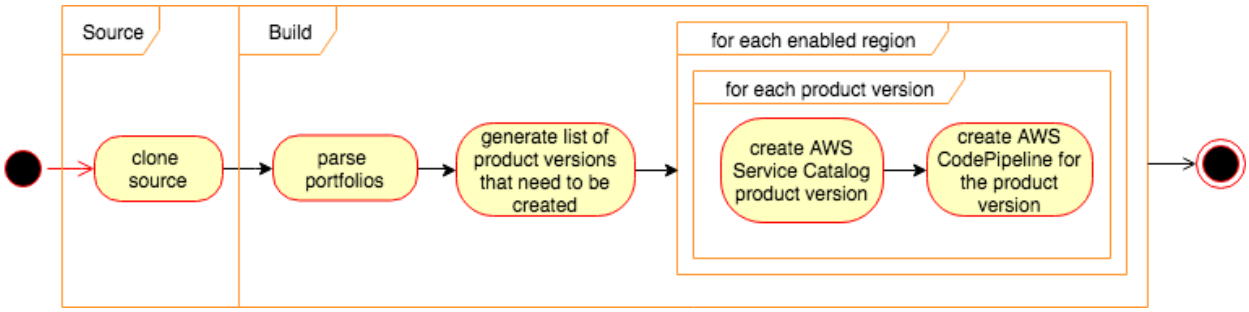
CHAPTER

TWO

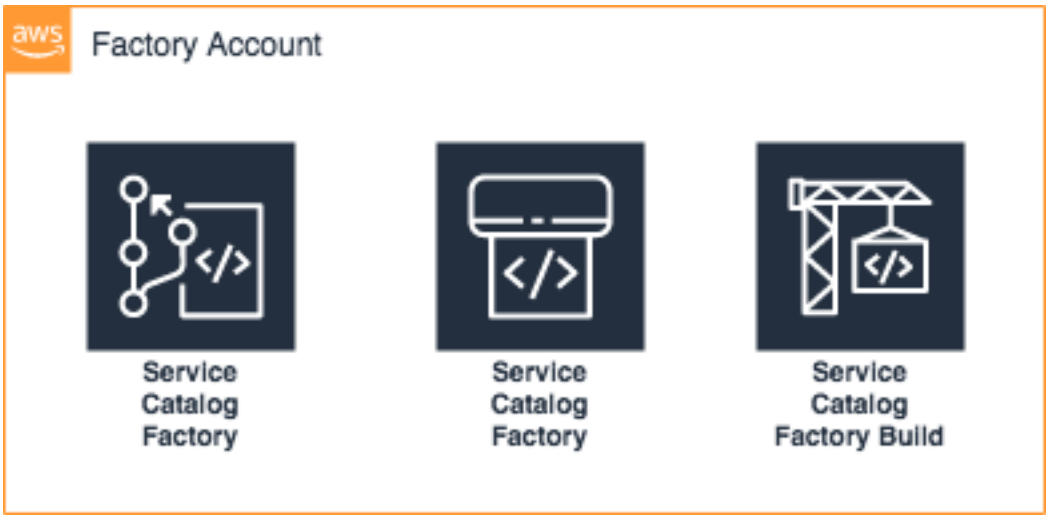
GETTING IT UP AND RUNNING

ServiceCatalog-Factory runs in your AWS Account. In order for you to install it into your account you can use the aws-service-catalog-factory cli. This is distributed via [PyPi](https://pypi.org/project/aws-service-catalog-factory/)

2.1 What am I going to install?



using the following services:



2.2 Before you install

You should consider which account will be the home for your factory. This account will contain the AWS CodePipelines and will need to be accessible to any accounts you would like to share with.

2.3 Installation

See https://service-catalog-tools-workshop.com/installation.html

CHAPTER

THREE

BUILDING YOUR PIPELINES

You write portfolio files that contain products and their versions.

Each Product version in your portfolio becomes a CodePipeline in your Factory account.

For each product version you must specify the following:

3.1 Source

The source for your pipeline can be a CodeCommit repo or a GitHub repo.

3.1.1 Using AWS CodeCommit

See https://service-catalog-tools-workshop.com/every-day-use/100-creating-a-product/300-add-the-source-code.html

3.1.2 Using Amazon S3

See https://service-catalog-tools-workshop.com/every-day-use/100-creating-a-product/310-creating-s3-pipelines.html

3.1.3 Using GitHub

See https://service-catalog-tools-workshop.com/every-day-use/100-creating-a-product/305-creating-codestar-pipelines.html

3.1.4 Using CodeStarSourceConnection (Bitbucket / Github.com / Github Enterprise)

See https://service-catalog-tools-workshop.com/every-day-use/100-creating-a-product/305-creating-codestar-pipelines.html

3.2 Preprocessing templates (using Jinja2)

Note: ShouldParseAsJinja2Template was added in version 0.20.0

When you write your product.template.yaml file you can use Jinja2 to optimise the way you write your CloudFormation template. To enable this you must specify the `ShouldParseAsJinja2Template` option and your product.template.yaml file must be renamed product.template.yaml.j2

Here is an example showing the option turned on:

```
Portfolios:
  -
    Products:
      - Name: account-iam
        Options:
          ShouldParseAsJinja2Template: True
        Versions:
          - Name: v1
            Description: IAM Policies needed
            Source:
              Provider: CodeCommit
              Configuration:
                RepositoryName: development-account-networking
                BranchName: v1
```

3.3 Using JSON

By default factory assumes you will be using YAML based CloudFormation templates. You can use JSON based products by changing the provisioning configuration for CloudFormation:

```
Products:
  - Name: json-product
    Portfolios:
      - mandatory
    Versions:
      - Name: v1
        Provisioner:
          Type: CloudFormation
          Format: json
        Source:
          Configuration:
            BranchName: master
            RepositoryName: aws-iam-administrator-access-assumable-role-account
          Provider: CodeCommit
        Tags:
          - Key: provider
            Value: central-it-team
```

Please note the example above is not complete, it is just illustrating how to set a provisioner.

Note: ShouldParseAsJinja2Template was added in version 0.35.0

3.4 Build

Note: This was added in version 0.48.0

Each product pipeline can have a build stage. You can specify the BuildSpecImage and the BuildSpec to use by setting the Build properties in the Stages object:

```
Products:
  - Description: account-bootstrap-shared
    Distributor: CCOE
    Name: account-bootstrap-shared
    Owner: CCOE@Example.com
    Source:
      Configuration:
        RepositoryName: account-bootstrap-shared
      Provider: CodeCommit
    Stages:
      Build:
        BuildSpecImage: aws/codebuild/standard:4.0
        BuildSpec: |
          version: 0.2
          phases:
            install:
              runtime-versions:
                python: 3.x
            build:
              commands:
                - make build
          artifacts:
            files:
              - '*'
              - '**/*'

    SupportDescription: Find us on Slack or Wiki
    SupportEmail: ccoe-support@Example.com
    SupportUrl: https://example.com/intranet/teams/ccoe/products/account-factory
    Tags: []
    Portfolios:
      - account-vending
    Versions:
      - Description: Creates, codebuild project that can be run to bootstrap an␣
↪account
          and lambda function that can be used to back a custom resource so the␣
↪codebuild
          project can be started from CloudFormation
        Name: v1
```

In the example above we added a BuildSpecImage and a BuildSpec. You cal also override the stages object in the versions list:

```
Products:
  - Description: account-bootstrap-shared
    Distributor: CCOE
    Name: account-bootstrap-shared
    Owner: CCOE@Example.com
```

(continues on next page)

(continued from previous page)

```
    Source:
      Configuration:
        RepositoryName: account-bootstrap-shared
      Provider: CodeCommit

    SupportDescription: Find us on Slack or Wiki
    SupportEmail: ccoe-support@Example.com
    SupportUrl: https://example.com/intranet/teams/ccoe/products/account-factory
    Tags: []
    Portfolios:
      - account-vending
    Versions:
      - Description: Creates, codebuild project that can be run to bootstrap an␣
↪account
          and lambda function that can be used to back a custom resource so the␣
↪codebuild
          project can be started from CloudFormation
        Name: v1
        Stages:
          Build:
            BuildSpecImage: aws/codebuild/standard:4.0
            BuildSpec: |
              version: 0.2
              phases:
                install:
                  runtime-versions:
                    python: 3.x
                build:
                  commands:
                    - make build
              artifacts:
                files:
                  - '*'
                  - '**/*'
```

Please note you cannot set a build stage for a combined pipeline.

3.5 Tests

Each product pipeline will run aws cloudformation validate-template on your product.template.yaml.

You can optionally run CFNNag on your template. You can enable it using the Options configuration for your product or for your product version.

3.5.1 Specifying the options at a product level

You can add CFN for all versions of a product:

```
Portfolios:
  -
    Products:
      - Name: account-iam
        Options:
```

(continues on next page)

(continued from previous page)

```
          ShouldCFNNag: True
        Versions:
          - Name: v1
            Description: IAM Policies needed
            Source:
              Provider: CodeCommit
              Configuration:
                RepositoryName: development-account-networking
                BranchName: v1
```

3.5.2 Specifying the options at a version level

You can add CFN for a specific version of a product:

```
Portfolios:
  -
    Products:
      - Name: account-iam
        Versions:
          - Name: v1
            Description: IAM Policies needed
            Options:
              ShouldCFNNag: True
            Source:
              Provider: CodeCommit
              Configuration:
                RepositoryName: development-account-networking
                BranchName: v1
```

3.6 Package

By default, the BuildSpec for the AWS CodeBuild project used at the package stage will run the following for each region:

```
aws cloudformation package \
    --template $(pwd)/product.template.yaml \
    --s3-bucket sc-factory-artifacts-${ACCOUNT_ID}-{{ region }} \
    --s3-prefix ${STACK_NAME} \
    --output-template-file \
    product.template-{{ region }}.yaml
```

This allows you to use AWS CloudFormation transform statements within your products meaning you can use AWS::Serverless::Function and other AWS CloudFormation types.

You can override this behaviour be making a change to your product version, adding a BuildSpec string:

```
Versions:
  - Name: v1
    Description: MVP for iam development account.
    Source:
      Provider: CodeCommit
      Configuration:
```

(continues on next page)

(continued from previous page)

```
        RepositoryName: guardduty-master-enabler
        BranchName: v1
    Stages:
      Package:
        BuildSpec: |
          version: 0.2
          phases:
            install:
              runtime-versions:
                python: 3.8
            build:
              commands:
              {% for region in ALL_REGIONS %}
                - aws cloudformation package \
                    --template $(pwd)/product.template.yaml \
                    --s3-bucket sc-factory-artifacts-${ACCOUNT_ID}-{{ region }} \
                    --s3-prefix ${STACK_NAME} \
                    --output-template-file product.template-{{ region }}.yaml
              {% endfor %}
          artifacts:
            files:
              - '*'
              - '**/*'
```

Note: Since version 0.48.0 you should be setting your Package BuildSpec in a Stages object unless using a combined pipeline. Prior to this you set it in the versions or product object. The stages object can be set in the product object and then overridden in the version object.

Please note, you need to specify the runtime-versions you intend to use.

Please note, when using this your BuildSpec will be rendered as a Jinja2 template with the following variables available in the context: - product - version - ALL_REGIONS

If you do decide to override the default build spec please ensure you capture the artifacts needed for the deploy stage.

The default image used for the Package stage of the pipeline is `aws/codebuild/standard:4.0`. To choose the image, add a BuildSpecImage configuration to either the product or version.

```
Versions:
  - Name: v1
    Description: MVP for iam development account.
    BuildSpecImage: aws/codebuild/standard:4.0
    BuildSpec: |
      mybuildspec...
```

3.7 Deploy

The deploy stage will push your templates into AWS Service Catalog for each region you are opperating in. The deploy stage will look for files matching: `product.template-{{ region }}.yaml`

3.8 Setting versions to be active or not

From the portfolio you can set a version to be active or not using the following syntax:

```
Products:
  - Name: account-vending-machine
    Owner: central-it@customer.com
    Description: The iam roles needed for you to do your jobs
    Distributor: central-it-team
    SupportDescription: Contact us on Chime for help #central-it-team
    SupportEmail: central-it-team@customer.com
    SupportUrl: https://wiki.customer.com/central-it-team/self-service/account-iam
    Tags:
    - Key: product-type
      Value: iam
    Versions:
      - Name: v1
        Description: The iam roles needed for you to do your jobs
        Active: False
        Source:
          Provider: CodeCommit
          Configuration:
            RepositoryName: account-vending-machine
            BranchName: v1
```

You set Versions[].Active to False to stop users from provisioning your product version.

Please note the `servicecatalog-factory-pipeline` updates the active setting. If you find the value is not in sync run the pipeline.

3.9 Specifying versions of a component and/or products outside of the main portfolio file

You may find that your portfolio file increases in size fairly quickly. Having a large file to manage is often more complicated than having multiple, smaller files. If you find yourself in this situation you can provide the specification for component versions outside of your main portfolio file.

For example:

You have a portfolio file named `demo.yaml` under your `portfolios` directory.

In `demo.yaml` you define a portfolio named `central-it-team-portfolio` under the `Portfolios` section and a component/product named `account-vending-account-creation` under the `Products` section:

```
Schema: factory-2019-04-01
Portfolios:
  - DisplayName: central-it-team-portfolio
    Description: A place for self service products ready for your account
    ProviderName: central-it-team
    Associations:
      - arn:aws:iam::${AWS::AccountId}:role/Admin
    Products:
      - Name: account-vending-account-creation
        Owner: central-it@customer.com
        Description: template used to interact with custom resources in the shared␣
↪projects
```

(continues on next page)

(continued from previous page)

```
      Distributor: central-it-team
      SupportDescription: Contact us on Chime for help #central-it-team
```

3.9.1 For Products

As well as specifying your `Products` section for the product in the portfolio file, you can specify it as a product file within a directory structure which matches the flow of the manifest file using the following syntax:

- `/portfolios/<name_of_manifest_without.yaml>/Portfolios/<DisplayName_of_portfolio>/Products/<name_of_product>/<name of the product>.yaml`

For example: To specify a product called `product-a` under the Products section of the `central-it-team-portfolio` portfolio defined in the 'demo.yaml' file, you can create a directory named in the following way:

- `/portfolios/demo/Portfolios/central-it-team-portfolio/Products/product-a/`

And a product file like:

- `/portfolios/demo/Portfolios/central-it-team-portfolio/Products/product-a/product-a.yaml`

With the Content:

```
Owner: central-it@customer.com
Description: template used to interact with custom resources in the shared projects
Distributor: central-it-team
SupportDescription: Contact us on Chime for help #central-it-team
```

The Product name is taken from the filename of the product - this should also match the product folder name

---

Note: You can put products in files or in the portfolio file

3.9.2 For Versions

Rather than specifying your `Versions` section for the component/product, you can specify it in a specifications file within a directory structure which matches the flow of the manifest file using the following syntax:

- `/portfolios/<name_of_manifest_without.yaml>/Portfolios/<DisplayName_of_portfolio>/Products/<name_of_product>/Versions`

Note: You can alternatively use 'Components' instead of 'Products'

For example:

To specify the Versions section of the `account-vending-account-creation` defined in the 'demo.yaml' file, you can create a directory named in one of the following two ways:

- `/portfolios/demo/Portfolios/central-it-team-portfolio/Components/account-vending-account-creation/Versions/`
- `/portfolios/demo/Portfolios/central-it-team-portfolio/Products/account-vending-account-creation/Versions/`

Note:

- You create this structure within the root of your `ServiceCatalogFactory` repository.
- The `demo.yaml` file should already be under the `/portfolios` folder.

Under the `Versions` folder, you can now create a folder for each version of your component/product which you place a `specification.yaml` file which contains the relevant version information:

```
# tree .
.
├── v1
│   └── specification.yaml
└── v2
    └── specification.yaml

2 directories, 2 files
```

The files named `specification.yaml` need to contain the details for the version:

```
Description: template used to interact with custom resources in the shared projects.
Active: True
Source:
  Provider: CodeCommit
  Configuration:
    RepositoryName: account-vending-account-creation
    BranchName: master
```

Example of the full folder structure:

Folder Structure for above examples should look like this under `ServiceCatalogFactory`

```
# tree .

.
└── portfolios
    ├── demo
    │   └── Portfolios
    │       └── central-it-team-portfolio
    │           └── Products
    │               └── account-vending-account-creation
    │                   └── Versions
    │                       ├── v1
    │                       │   └── specification.yaml
    │                       └── v2
    │                           └── specification.yaml
    │               └── account-vending-account-creation.yaml
    └── demo.yaml

9 directories, 4 files
```

When your service-catalog-factory pipeline runs it will treat these versions as if they were defined within the portfolio file.

3.10 Reducing the number of pipelines

By default factory will create an AWS CodePipeline for each product version you specify. By specifying a different PipelineMode you can alter this behaviour:

```
Products:
  - Description: iam-assume-roles-spoke product
    Distributor: central-it-team
    Name: aws-iam-assume-roles-spoke
    Owner: central-it@customer.com
    SupportDescription: Contact us on Chime for help
    SupportEmail: central-it-team@customer.com
    SupportUrl: https://wiki.customer.com/central-it-team/self-service/account-iam
    PipelineMode: combined
    Portfolios:
      - combined
```

When you specify a combined pipeline mode only a single pipeline will be created. There will be a source for each version of your product. When the pipeline runs only the changed product version will be updated in AWS Service Catalog. You can only set PipelineMode for products that you define outside of portfolios and you can only specify a buildspec for the product and not the versions.

CHAPTER

FOUR

DESIGNING YOUR PRODUCTS

Your products are AWS Service Catalog Products and so you are confined to the conventions of Service Catalog.

4.1 Product Versions

In Service Catalog, each product has one or more versions. Versions represent changes to your products. When you make a change to a published product it is recommended to make a new version and make it available to your users.

Users can provision multiple versions of your product at the same time or they may choose to update a product from one version to another. It is good to communicate to your users whether you would like them to update a product or provision a new product when you issue a change.

4.2 Product Right-Sizing

Right-sizing your product is difficult. Try to create products that achieve a business outcome and that can be deployed independently. If something is reusable try to make it independent using AWS SSM Parameters or exporting the values from the AWS CloudFormation templates.

4.3 Including your product in multiple portfolios

Note: This was added in version 0.2.0

You may want to include a product in more than one portfolio. This is possible with aws-service-catalog-factory. To do this you must specify the product in a Products attribute at the root of the portfolio:

```
Schema: factory-2019-04-01

Products:
  - Name: account-iam-standalone
    Owner: central-it@customer.com
    Description: The iam roles needed for you to do your jobs
    Distributor: central-it-team
    SupportDescription: Contact us on Chime for help #central-it-team
    SupportEmail: central-it-team@customer.com
    SupportUrl: https://wiki.customer.com/central-it-team/self-service/account-iam
    Portfolios:
```

(continues on next page)

(continued from previous page)

```
      - central-it-team-portfolio
      - my-other-portfolio
    Tags:
      - Key: product-type
        Value: iam
    Versions:
      - Name: v1
        Description: The iam roles needed for you to do your jobs
        Active: True
        Source:
          Provider: CodeCommit
          Configuration:
            RepositoryName: account-iam
            BranchName: v1
```

4.4 Adding associations to a portfolio

Within the products specification you can list the portfolios it should be associated with. The values for the portfolios should be the display name of the portfolio as specified in the portfolio file:

```
Schema: factory-2019-04-01

Products:
  - Name: account-iam-standalone
    ...
    Portfolios:
      - my-other-portfolio
    Versions:
      - Name: v1
        Description: The iam roles needed for you to do your jobs
        Active: True
        Source:
          Provider: CodeCommit
          Configuration:
            RepositoryName: account-iam
            BranchName: v1
Portfolios:
  - DisplayName: my-other-portfolio
    Description: A place for self service products ready for your account
    ProviderName: central-it-team
    ...
```

4.5 Adding launch role constraints to a product

Note: This was added in version 0.39.0

Within the product specification you can also specify a LocalRoleName for a LaunchRoleConstraint:

```
Schema: factory-2019-04-01
```

(continues on next page)

(continued from previous page)

```
Products:
  - Name: account-iam-standalone
    Owner: central-it@customer.com
    Distributor: central-it-team
    Constraints:
        Launch:
            LocalRoleName: ServiceCatalogLaunchRole
```

The rolename you specify must exist in the puppet account and should exist in any spoke account where you want the association to take affect. The role must be assumable by the Service Catalog principal and should have the permissions needed to provision your product.

4.6 Deleting a Product Version

Note: This was added in version 0.42.0

Add "Status: terminated" to your product version. The product version will be delete from AWS Service Catalog and the AWS CloudFormation stack will be deleted also - deleting the AWS CodePipeline it created.

```
Schema: factory-2019-04-01

Products:
  - Name: account-iam-standalone
    Versions:
    Versions:
      - Name: v1
        Description: The iam roles needed for you to do your jobs
        Status: terminated
        Active: True
        Source:
          Provider: CodeCommit
          Configuration:
            RepositoryName: account-iam
            BranchName: v1
```

4.7 Deleting a Product

Note: This was added in version 0.42.0

Add "Status: terminated" to your product. Any products matching this product Name will be disassociated from any portfolios, then deleted.

This will also delete all Product Version pipelines relating to this product and any portfolios it is associated with from AWS CloudFormation

```
Schema: factory-2019-04-01

Products:
```

(continues on next page)

(continued from previous page)

```
  - Name: account-iam-standalone
    Status: terminated
```

CHAPTER

FIVE

UPGRADING

Firstly, verify which version you have installed already:

```
servicecatalog-factory version
```

If this errors, check you have activated your virtualenv.

Then you are ready to install the version you want:

```
pip install aws-service-catalog-factory==<version>
```

If you want to upgrade to the latest you can run:

```
pip install --upgrade aws-service-catalog-factory
```

Once you have completed the upgrade you will have to bootstrap your install again:

```
servicecatalog-factory bootstrap
```

And finally, you can verify the upgrade has worked by running version again:

```
servicecatalog-factory version
```

CHAPTER

SIX

USING THE CLI

When you install `aws-service-catalog-factory` you install a command line tool `servicecatalog-factory`.

When you bootstrap the framework and upgrade it you use the cli tool to perform these actions.

There are other commands that you may find useful:

6.1 validate

You can use the `servicecatalog-factory` cli to validate your portfolio yaml files:

```
servicecatalog-factory validate portfolios/
```

6.2 add-secret

Note: This was added in version 0.17.0

You can use the `servicecatalog-factory` cli to add Secret Key and OAuth tokens that can be used to create products where the source code comes from a GitHub Repo.

```
servicecatalog-factory add-secret <secret-name> <oauth_token> <secret_token>
```

If you do not specify an secret token your oauth token will be used. You should do this if you are using personal access tokens.

6.3 add-product-to-portfolio

Note: This was added in version 0.9.0

You can use the `servicecatalog-factory` cli to add products into your portfolio. This will add your product to the remote version of the portfolio file you specify:

```
servicecatalog-factory add-product-to-portfolio portfolio_file_name portfolio_display_
↪name product_definition_file.yaml
```

product_definition_file.yaml should be a yaml file in the following format:

```
Description: The iam roles needed for you to do your jobs
Distributor: central-it-team
Name: account-iam-2
Owner: central-it@customer.com
SupportDescription: Contact us on Chime for help
SupportEmail: central-it-team@customer.com
SupportUrl: https://wiki.customer.com/central-it-team/self-service/account-iam
Tags:
- Key: product-type
  Value: iam
```

Note: If your `product_definition_file.yaml` file contains a source which is CodeCommit then the repo will be created for you.

6.4 remove-product-from-portfolio

Note: This was added in version 0.9.0

You can use the `servicecatalog-factory` cli to remove products from your portfolio. This will remove your product from the remote version of the portfolio file you specify:

```
servicecatalog-factory remove-product-from-portfolio portfolio_file_name portfolio_
↪display_name product_name
```

Note: This command will not delete git repos

6.5 add-version-to-product

Note: This was added in version 0.9.0

You can use the `servicecatalog-factory` cli to add versions to your products. This will add your version to the specified product in the remote version of the portfolio file you specify:

```
servicecatalog-factory add-version-to-product example-simple.yaml central-it-team-
↪portfolio account-iam-2 version_definition_file.yaml
```

version_definition_file.yaml should be a yaml file in the following format:

```
Name: v1
Description: The iam roles needed for you to do your jobs
Active: true
Source:
  Provider: CodeCommit
```

(continues on next page)

(continued from previous page)

```
  Configuration:
    BranchName: v1
    RepositoryName: account-iam
```

Note: If your `version_definition_file.yaml` file contains a source which is CodeCommit then the repo will be created for you.

6.6 remove-version-from-product

Note: This was added in version 0.9.0

You can use the `servicecatalog-factory` cli to remove versions from products in your portfolio. This will remove your version from your product in the remote version of the portfolio file you specify:

```
servicecatalog-factory remove-version-from-product portfolio_file_name portfolio_
↪display_name product_name version_name
```

Note: This command will not delete git repos

6.7 import-product-set

Note: This was added in version 0.8.0

You can use the `servicecatalog-factory` cli to import products from the aws-service-catalog-products shared repo.

This will update your portfolio file, create your AWS CodeCommit repos, export the code from the AWS shared code repo and push the code into your AWS CodeCommit repo on the correct branch.

```
servicecatalog-factory import-product-set ServiceCatalogFactory/portfolios/example-
↪simple-github.yaml aws-iam central-it-team-portfolio
```

You must specify the path to the portfolio file you want to add the product set to, the name of the product set and the name of the portfolio you want to add it to.

6.8 list-resources

Note: This was added in version 0.7.0

You can use the `servicecatalog-factory` cli to list all the resources that will be created to bootstrap the framework

```
servicecatalog-factory list-resources
```

Will return the following markdown:

```
# Framework resources
## SSM Parameters used
- /servicecatalog-factory/config
## Resources for stack: servicecatalog-factory-regional
| Logical Name | Resource Type | Name |
|---|---|---|
| Param | AWS::SSM::Parameter | service-catalog-factory-regional-version |
| PipelineArtifactBucket | AWS::S3::Bucket | Fn::Sub: sc-factory-artifacts-${AWS::AccountId}-${AWS::Region} |
## Resources for stack: servicecatalog-factory
| Logical Name | Resource Type | Name |
|---|---|---|
| Param | AWS::SSM::Parameter | service-catalog-factory-version |
| SourceRole | AWS::IAM::Role | SourceRole |
| CodeRepo | AWS::CodeCommit::Repository | ServiceCatalogFactory |
| BuildRole | AWS::IAM::Role | CodeRole |
| BuildProject | AWS::CodeBuild::Project | servicecatalog-product-factory-build |
| CodePipelineTriggerRole | AWS::IAM::Role | CodePipelineTriggerRole |
| PipelineRole | AWS::IAM::Role | CodePipelineRole |
| FactoryPipelineArtifactBucket | AWS::S3::Bucket | Not Specified |
| CatalogBucket | AWS::S3::Bucket | Not Specified |
| Pipeline | AWS::CodePipeline::Pipeline | Fn::Sub: ${AWS::StackName}-pipeline |
| DeliverySourceRole | AWS::IAM::Role | DeliverySourceRole |
| DeliveryBuildRole | AWS::IAM::Role | DeliveryCodeRole |
| DeliveryPipelineRole | AWS::IAM::Role | DeliveryCodePipelineRole |
AWS::StackName evaluates to servicecatalog-factory
```

6.9 show-pipelines

Note: This was changed in version 0.70.0

You can use the `servicecatalog-factory` cli to list all the AWS CodePipelines in your factory along with their status

```
servicecatalog-factory show-pipelines ServiceCatalogFactory
```

Will return the following:

```
+------------+-------------------------------------------------+----------------------
↪---------------+-----------------------------------+-----------+----------------------
↪---------------------+-------------------------------------------------------------------+-
↪---------------+-----------------------------------------------------------+
| Type       | Name                                            | Execution Id
↪                | Start Time                        | Status    | Last Commit Id
↪                     | Last Commit Message                                              |
↪Duration       | Trend                                                        |
+------------+-------------------------------------------------+----------------------
↪---------------+-----------------------------------+-----------+----------------------
↪---------------------+-------------------------------------------------------------------+-
↪---------------+-----------------------------------------------------------+
| core       | servicecatalog-factory-pipeline                 | d435a6b7-cc21-442f-
↪ae47-e2947ae56ce3 | 2021-09-02 12:54:07.136000+01:00 | Failed    |
↪c765347e01a36514a1e2f4cce691fc51964005d1 | sdfsdf
↪                  | 0:01:06.964000 | Failed, Failed, Succeeded, Succeeded, Failed
↪       |
| apps       | app--ssm-parameter-v2-pipeline                  | 6f556352-3354-4641-
↪a78e-f95bfe262470 | 2021-08-03 14:38:44.429000+01:00 | Failed    |
↪95e272ad32858b2a2d263268dde8fc7ba0eb6cc1 | Added param.tf
↪                  | 0:01:10.692000 | Failed
↪       |
| portfolios | bug-demo-portfolio-cdk-support-iam-v2-pipeline | N/A
↪               | 2021-11-03 19:43:28.612484       | N/A       | N/A
↪                    | N/A                                                              |
↪0:00:00        |                                                             |
| portfolios | cdk-support-iam-v2-pipeline                     | 21cebf98-9dcb-4a10-
↪85bd-d7e273f9eaf1 | 2021-09-02 12:34:28.904000+01:00 | Succeeded |
↪09705a6242cd67c6e46364a7e70ae3857a2e1c65 | sdsdsd
↪                  | 0:01:48.474000 | Succeeded, Failed, Succeeded, Succeeded,
↪Succeeded    |
| portfolios | cdk-support-bootstrap-v4-pipeline               | 3740315e-f317-4d5b-
↪baa2-1a022f22f6f4 | 2021-04-09 19:51:09.178000+01:00 | Succeeded |
↪b38fe7fea05002e1e3d1f86f9454d8a5a64bbceb | Edited handler.py
↪                  | 0:02:47.400000 | Succeeded, Succeeded, Succeeded, Superseded,
↪Failed   |
| portfolios | cdk-ssm-parameter-single-stack-v1-pipeline      | c84c925a-ca04-4763-
↪b430-8fa8c370e995 | 2021-04-09 19:24:09.927000+01:00 | Succeeded |
↪aadb4ca8198c318f976c975df1c3d3ad62f1d84f | initial add
↪                  | 0:05:17.851000 | Succeeded, Succeeded, Failed, Failed, Failed
↪       |
|            |                                                 |
↪               |                                   |           |
↪                    |                                                                  |
↪               |                                                             |
```

(continues on next page)

(continued from previous page)

```
| portfolios | cdk-ssm-parameter-single-stack-v2-pipeline      | a9a289fe-55e5-44e1-
↪8b5c-e73f003c0467 | 2021-05-07 16:34:45.587000+01:00 | Succeeded |␣
↪5509f682d2207e0439c16b7dc63deccdded86c44 | Edited cdk-ssm-parameter-single-stack-v1-
↪pipeline-stack.ts | 0:05:17.644000 | Succeeded, Failed, Failed, Failed, Failed      ␣
↪       |
| portfolios | cdk-ssm-parameter-two-stacks-v1-pipeline       | ddd83a86-c10f-4031-
↪b4bd-4c17e266561f | 2021-03-25 18:34:53.223000+00:00 | Failed    | N/A              ␣
↪                        | N/A                                                        ␣
↪ | 0:00:01.293000 | Failed, Failed, Failed, Failed, Failed                 |
| portfolios | simpleproduct-v1-pipeline                      | 043bf8e6-154f-436f-
↪9ef2-c0d19d2de57e | 2021-06-07 11:28:48.976000+01:00 | Failed    | N/A              ␣
↪                        | N/A                                                        ␣
↪ | 0:00:01.734000 | Failed, Failed, Succeeded, Succeeded, Succeeded       |
| portfolios | simpleproduct-suffixed-v1-pipeline             | 6cdcb202-dcc7-4c08-
↪b8a7-a7b42d0cf2da | 2021-03-03 22:46:08.472000+00:00 | Failed    | N/A              ␣
↪                        | N/A                                                        ␣
↪ | 0:00:01.247000 | Failed                                                 |
| stacks     | stack--ssm-parameter-v2-pipeline               | 7badb987-137d-4c3b-
↪b773-3e0cc66b5782 | 2021-09-02 11:02:35.745000+01:00 | Succeeded |␣
↪467f87832e5330dbdac346ce823e5e0671b27435 | Added stack.template.yaml                 ␣
↪                   | 0:02:22.555000 | Succeeded, Failed, Failed, Succeeded          ␣
↪       |
| stacks     | stack--aac-type-b-network-v1-pipeline          | N/A                       ␣
↪               | 2021-11-03 19:43:28.612484       | N/A       | N/A                  ␣
↪                     | N/A                                                          |␣
↪0:00:00        |                                                          |
| workspaces | workspace--ssm-parameter-v2-pipeline           | e95b2731-5c8f-4a1c-
↪a35e-430770e10783 | 2021-08-03 15:13:42.181000+01:00 | Succeeded |␣
↪64d866c7205f33266d85d7c99eb11f38f4ff99d2 | Edited param.tf                           ␣
↪                   | 0:01:46.176000 | Succeeded, Succeeded, Succeeded, Succeeded,␣
↪Succeeded |
+------------+-------------------------------------------------+-----------------------
↪---------------+-------------------------------------+----------+---------------------
↪---------------------+-------------------------------------------------------------+-
↪---------------+-----------------------------------------------------------+
```

Note: This was added in version 0.11.0

You can specify the output format for show-pipelines. Valid options are `table`, `json` and `html`

```
servicecatalog-factory show-pipelines ServiceCatalogFactory/ --format json
```

6.10 nuke-product-version

You can use the `servicecatalog-factory` cli to remove a product from AWS Service Catalog and to remove the AWS CodePipeline that was generated by this library. To use it, you will need your portofolio name, product name and product version - all of which is available to view from your AWS Service Catalog console.

Once you have the details you can run the following command:

```
servicecatalog-factory nuke-product-version example-simple-central-it-team-portfolio␣
↪account-iam v1
```

6.11 delete-stack-from-all-regions

You can delete a stack from every region using the following command:

```
servicecatalog-factory delete-stack-from-all-regions stack-name
```

Please note, this will only delete the stack from the regions you have specifed in your config.

6.12 fix-issues

Whilst developing your products you may find AWS CloudFormation stacks in states you cannot work with. If this happens the fix-issues command will try to resolve it for you. It will prompt you to confirm anything it does within your account before it does it so give it a try when you get stuck.

```
servicecatalog-factory fix-issues ServiceCatalogFactory/portfolios
```

CHAPTER

SEVEN

USING THE SDK

Service Catalog Factory includes a published SDK. You can make use of the python functions available:

```
from servicecatalog_factory import sdk
```

The functions available are:

7.1 Functions

servicecatalog_factory.sdk.**add_product_to_portfolio**(*portfolio_file_name*, *portfolio_display_name*, *product*)

This function allows to you to add a product to a portfolio that exists already. If the product contains a Source and it is defined as an AWS CodeCommit type then the function will ensure the repository and branch exist.

Parameters

- **portfolio_file_name** – The name of the file (including .yaml) of your portfolio file. For example ServiceCatalogFactory/portfolios/central_it.yaml should be central_it.yaml
- **portfolio_display_name** – The value of the portfolio DisplayName the product should be added to.
- **product** – a dict of the product that you want to add to the portfolio

servicecatalog_factory.sdk.**add_version_to_product**(*portfolio_file_name*, *portfolio_display_name*, *product_name*, *version*)

This function allows to you to add a version to product within a portfolio that exists already. If the version contains a Source and it is defined as an AWS CodeCommit type then the function will ensure the repository and branch exist.

Parameters

- **portfolio_file_name** – The name of the file (including .yaml) of your portfolio file. For example ServiceCatalogFactory/portfolios/central_it.yaml should be central_it.yaml
- **portfolio_display_name** – The value of the portfolio DisplayName the product version should be added to.
- **product_name** – The value of the product Name the version should be added to.
- **version** – a dict of the version that you want to add to the portfolio

servicecatalog_factory.sdk.**bootstrap**()

Bootstrap the factory. This will create the AWS CodeCommit repo containing the config and it will also create the AWS CodePipeline that will run the solution.

servicecatalog_factory.sdk.**remove_product_from_portfolio**(*portfolio_file_name*, *portfolio_display_name*, *product_name*)

This function allows to you to remove a product from a portfolio that exists already.

Parameters

- **portfolio_file_name** – The name of the file (including .yaml) of your portfolio file. For example ServiceCatalogFactory/portfolios/central_it.yaml should be central_it.yaml
- **portfolio_display_name** – The value of the portfolio DisplayName where the product exists.
- **product_name** – The name of the product you want to remove

servicecatalog_factory.sdk.**remove_version_from_product**(*portfolio_file_name*, *portfolio_display_name*, *product_name*, *version_name*)

This function allows to you to remove a version of a product within a portfolio that exists already.

Parameters

- **portfolio_file_name** – The name of the file (including .yaml) of your portfolio file. For example ServiceCatalogFactory/portfolios/central_it.yaml should be central_it.yaml
- **portfolio_display_name** – The value of the portfolio DisplayName the product version should be removed from.
- **product_name** – The value of the product Name the version should be removed from.
- **version_name** – The name of the version you want to remove

servicecatalog_factory.sdk.**set_regions**(*regions*)

Set the regions for the factory.

Parameters **regions** – The list of AWS regions

servicecatalog_factory.sdk.**upload_config**(*config*)

This function allows you to upload your configuration for factory. At the moment this should be a dict with an attribute named regions:

regions: ['eu-west-3', 'sa-east-1',

]

Parameters **config** – The dict containing the configuration used for factory

CHAPTER

EIGHT

TERRAFORM SUPPORT

Service Catalog Factory includes support for HashiCorp's Terraform.

8.1 What is working

- Provision your Terraform based products using ServiceCatalog Puppet
- Sharing Terraform based products using ServiceCatalog Puppet portfolio sharing
- Using Parameters in ServiceCatalog Puppet
- Using SSM Parameters in ServiceCatalog Puppet

8.2 Getting it working

- you need to add the following product set to your portfolio: https://github.com/awslabs/aws-service-catalog-products/tree/master/aws-servicecatalog-factory-provisioners
- you need to provision the products in aws-servicecatalog-factory-provisioners into your spoke accounts where you will be provisioning Terraform based products
- you can then create products. Your products need to have their terraform code in a directory named tf.

Note: This was added in version 0.13.0 and the syntax changed in 0.14.0

To create a Terraform based product you must set the product versions provisioner to Terraform:

```
Versions:
    - Active: true
      Description: The iam roles needed for you to do your jobs
      Name: v1
      Provisioner:
        Type: Terraform
        Version: 0.11.14
        TFVars:
          - Foo
          - Bar
      Source:
        Configuration:
          BranchName: v1
```

(continues on next page)

(continued from previous page)

```
      RepositoryName: account-iam-terraform
    Provider: CodeCommit
```

You can choose which version of Terraform is used to provision your product. The valid options are listed here

The TFVars you specify are exposed as parameters when using AWS Service Catalog (meaning you can set them using parameters in AWS Service Catalog Puppet) .

8.3 How does it work

When you specify a product uses a Terraform provisioner the framework will generate an AWS CloudFormation template with the following resources: - an AWS S3 bucket that will be used to store the state - an AWS CodePipeline containing AWS CodeBuild steps that download and run a Terraform plan and apply - when you provision a Terraform based product the bucket and pipeline are provisioned into the account and the resources defined in the Terraform code are provisioned

8.4 What is still to come

- Using Depends On in ServiceCatalog Puppet where you can depend on a Terraform based product
- Using Outputs in ServiceCatalog Puppet for a Terraform based product

CHAPTER

NINE

AWS CDK SUPPORT

Service Catalog Factory includes support for preparing AWS CDK projects for provisioning or sharing using Service Catalog Puppet.

This doc only addresses the use case of building AWS Service Catalog products comprising of AWS CDK source code that will be shared using Service Catalog Puppet.

9.1 What is working

- Create a product using AWS CDK
- Provide parameters for your AWS CDK project using AWS CloudFormation parameters
- Capture outpouts for your AWS CDK Project using AWS CloudFormation stack outputs

9.2 Getting it working

9.2.1 Prerequisites

If you are building a service catalog of products comprising of AWS CDK projects you will need to run a CDK bootstrap in each account you are sharing portolios with. There is a helper product set that makes this easier for you:

https://github.com/awslabs/aws-service-catalog-products/tree/main/unsorted/cdk-support

You will need to provision the IAM product to each account only once and you can then provision the bootstrap product to specific regions where you would like to run cdk bootstrap. There are parameters on those products allowing you to customise the cdk bootstrap arguements.

You can look at the example manifest file for an example of how you can provision the prerequisites.

9.2.2 Creating a CDK product pipeline

To create a CDK product you will need a specialised pipeline that will create the wrapper AWS CloudFormation template that will be added to AWS Service Catalog and copy the source code for later deploying. In order to do this you can use the following syntax within your Portfolios section:

```
Versions:
  - Active: true
    Description: The iam roles needed for you to do your jobs
    Name: v1
```

(continues on next page)

(continued from previous page)

```
      Template:
        Name: CDK
        Version: 1.0.0
      Source:
        Configuration:
          BranchName: v1
          RepositoryName: account-iam-cdk
        Provider: CodeCommit
```

Adding the Template attribute tells the solution the product being described will be an AWS CDK product and so the specialised pipeline will be created.

If you need additional runtime-versions and commands to be run before the CDK deploy command is run you can specify these like so:

```
Versions:
  - Name: "v1"
    Owner: "data-governance@example.com"
    Description: "Simple product"
    Distributor: "cloud-engineering"
    SupportDescription: "Speak to data-governance@example.com about exceptions and␣
↪speak to cloud-engineering@example.com about implementation issues"
    SupportEmail: "cloud-engineering@example.com"
    SupportUrl: "https://wiki.example.com/cloud-engineering/governance/data-controls"
    Template:
      Name: CDK
      Version: 1.0.0
      Configuration:
        runtime-versions:
          python: 3.8
        install:
          commands:
            - pip install -r requirements.txt
```

Please note you can add multiple runtime-versions (it is a dictionary) and you can add multiple install commands (it is a list).

The runtime-versions are added and the commands are executed in the hub account when the product template is being created and in the spoke account when the CDK deploy is being executed.

9.2.3 Using the CDK based products

Once you have generated the product you can use them in a launch - the parameters and outputs will work as if the project was a plain old CloudFormation based product. You can share the portfolio containing the CDK based product using the normal spoke-local-portfolio method.

CHAPTER

TEN

PROJECT ASSURANCE

10.1 Assurance

This project has been through an assurance process to ensure the project is:

- valuable to AWS customers
- properly licenced

The same process ensures that there are mechanisms to ensure maintainers are:

- likely able to acceptably support it with regards to being responsive to github issues and pull requests

And finally, at the time of publishing:

- any 3rd party components actually contained in the repo are checked to ensure they are correctly licensed and that we are correctly complying with the open source licenses that apply to those 3rd party components.

10.2 Project Management

10.2.1 Quality Assurance

CICD Process

Unit tests are run on every commit. If unit tests fail a release of the project cannot occur.

The project dependencies are scanned on each commit for known vulnerabilities. If an issue is discovered a release of the project cannot occur.

Review Process

There are regular reviews of the source code where static analysis results and unit test coverage are assessed.

10.2.2 Raising a feature request

Product feature requests drive the majority of changes to this project. If you would like to raise a feature request please raise a Github issue.

10.2.3 Backwards compatibility

All changes to date have been fully backwards compatible. Effort will be made to ensure this where possible.

10.2.4 Design consultation

When there is a significant addition or change to the internal implementation we consult a limited number of users. Users are asked to access the potential impact so that we can understand the impact and the potential value of the change. If you would like to register as such a user please raise a Github issue.

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX