ArtifactS3 Repository

v0.0.3

Table of Contents

Purpose

CloudFormation templates are a great way to automate AWS provisioning and updates. The AWS architecture team provides and updates many sample templates but release versioning is left to the consumer. Teams within organizations will commonly incorporate some of the AWS example material into local solutions which are then revised further over time introducing breaking changes.

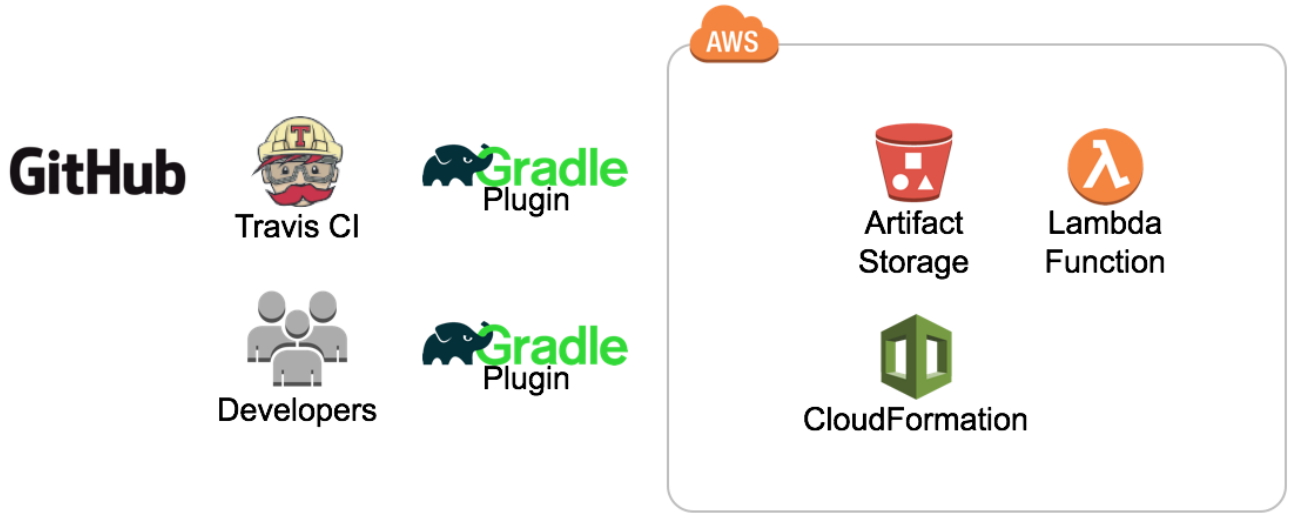
As the templates are just source code so one approach would be to use git submodules, but those aren't everyone's cup of tea. The cfn-stacks approach is to publish versioned packages which can be referenced from other Cloudformation templates directly or pulled locally using a dependency management tool. Rather than inventing a new toolchain the reference implementation uses Java tooling as it's well known but the concept could be implemented in many different ways.

Overview

A repository is used to store artifacts and a Gradle plugin to automate publishing. The plugin embeds version information into the output section of the template for identification post deployment and then packages the templates into a Java Archive or .jar file with and additional identifying suffix .cfn.jar. A publish task is provided to create the Maven repo artifact metadata and to automate the push to the repository.

The repository is an S3 bucket with an attached Lambda function that will unpack CloudFormation jars (*.cfn.jar) using key prefixes that preserve the organization and version information. The CloudFormation service requires that template files be hosted from S3 and by publishing to the repo we make versioned artifacts available for inclusion into other templates.

Diagram



Data Flows

1. **Publish templates via GitHub push:** Developers → GitHub → Travis CI → Gradle plugin in

project → ArtifactS3 Repo in S3. Template projects would include a .travis.yml file that would call the publish task.

    - Example: `git commit -m"Updated template"; git push`

2. **Publish templates directly:** Developers → Gradle plugin in project → ArtifactS3 Repo in S3
    - Example: `./gradlew publish`
3. **ArtifactS3 repo process:** New artifact → Unzipped by Lambda function → Artifact is now available as a Maven dependency or a Cloudformation template hosted in S3
    - Example `./gradlew createStack` using a template that includes another published template as a Nested Stack.

Build/Deploy

Prerequisite

- Java 8 JDK: Download from Oracle or use a packaged version for your OS

Command

`./gradlew build` will create the ArtifactS3 repo CloudFormation template in `build/cloudformation/artifats3-repo.yaml`

Deployment

There are possibly several prerequisite steps that have to be accomplished manually before CloudFormation can automatically deploy and then later update a stack.

1. Install the AWS CLI and configure a named profile
2. We'll go over all the options later in the docs but at a minimum a private repo can be created in the us-east-2 region using just a single command with a few parameters. The build.gradle file has a list of default options that can be overridden. An example of the region could have be overridden with an additional argument of -Pregion=us-west-1

```
./gradlew updateStack -Pprofile=YOUR_PROFILE_NAME -Pstack=STACK_NAME
-PDomainName=UNIQUE_S3_BUCKET_NAME
```

3. Once the stack has finished deploying note the name of the bucket that was created as that will be referenced in future publishing commands as your repo. By default the bucket is private and likely you are the only person who can access the published materials. CloudFormation runs with the permission of the user executing the commands so if others need access to the published artifacts use standard S3 permissions to authorize users, groups or instance roles.

The default deployment creates the all of the needed components for an artifact repo. The additional parameters offer the ability to deploy a website fronted by CloudFormation but requires a number of additional steps to create an SSL certificate and CloudFront user prior to deployment.

Updates

1. Updates to the stack are accomplished by running the `./gradlew updateStack` commmand again and changing either a command line parameter or one of the config files.

Configuration

Command line arguments can always be used and will override other settings but if the same parameters are used for the majority of deployments they can be specified in a file named local-config.groovy. If present in the root directory of the settings will get picked up and only if you want to override them do you need to use param arguments on the command line. The file is listed in .gitignore so you won't accidentally commit it and push it to a public repo.

```
project.params.with {
    profile = 'my-aws-cli-profile-name'
    stack = 'artifacts3-repo-test'
    DomainName = 'repo-test-38e894739r879'
}
```

Tool Links

- CloudFormation
- Git
- Gradle

Documentation Links

Web

Source

Version

This documentation was generated for version 0.0.3 from commit 7a095fd.