ArtifactS3 Plugin

v0.3.4

Table of Contents

Overview

A Gradle plugin that provides workflow capabilities for CloudFormation script authoring and publishing. The capabilities enabled by this plugin are created through the integration of a number of existing plugins to support the process shown below.

CloudFormation Stacks Process

Create ➔ Document ➔ Publish ➔ Version ➔ Deploy ➔ Update 🔄

- *Create:* The Gradle Base Plugin and an internal copy and filter task provide a basic template value replacement which is used to add the artifact id and version pulled from the build.gradle file and the packaging of templates into jar files. These values are exposes as stack outputs in all of the example templates for easy inspection of version information on a running stack.
- *Document:* The AsciiDoctor Gradle Plugin provides the ability to turn AsciiDoc markup into HTML and PDF documents. The LiveReload Gradle Plugin along with a browser extension will continuously rebuild the docs as you type.
- *Publish:* The Gradle Release Plugin automates the SCM tagging process to create a final release before publishing The Maven Publishing Plugin provides the ability to generate the metadata files needed for a Maven repository and the abilty to publish to an ArtifactS3 repo.
- *Version:* During development a SNAPSHOT version can be published but prior to using the stack in another project you'll publish a versioned release. This allows users of the template to use a stable version while new versions are being developed.
- *Deploy:* The Gradle AWS Plugin automates the interaction with CloudFormation so they can be easily run from the command line.
- *Update:* CloudFormation offers both on-demand updates and the ability to perform a "dry-run" though the use of a change set which describes the actions that will be taken if executed to avoid any update surprises.

Workflow

To support the process there is a recommended workflow.

Create Project

1. Create a copy of the example project
2. Update the project name in settings.gradle and the description in build.gradle
3. Create a local-config.groovy file to hold local settings
4. Remove the .git directory and then run `git init` again to reset the repo with your project content

Create Templates

1. Decompose your solution into components and either reuse existing functionality or write new components with the intention of creating reusable content were possible. The CloudFormation documentation has some best practices documented.
2. If your solution contains resources that are deployed within a VPC the AWS architects established a pattern of creating an app template that accepts VPC or subnet parameters to deploy within an existing VPC and then a second template, usually called *-main.yaml that creates a VPC and deploys the application as a substack.

Document

1. The project has already been configured with a doc template in src/docs/asciidoc/index.adoc which should be used to document as you develop by running the command `./gradlew docs` after adding content.
2. To use the live reload feature there's a couple of steps:
    - Install a browser extension
    - In one console window run gradle with the -t continuous build option `./gradlew -t docs`
    - In another console window run the live reload server process `./gradlew liveReload`
    - Open the build/asciidoc/html5/index.html file in a web browser with the livereload plugin activated and shortly after you make a change and save the source files the browser will update.

Publish

1. Before publishing you'll need to Install the AWS CLI and configure a named profile with access to your ArtifactS3 repo bucket. This profile name will be supplied either in your local-config.groovy file (example below) or as a command line argument. `./gradlew updateStack -Pprofile=my-aws-cli-profile-name -Pstack=test-repo`
2. If there is only one template in the main/cloudformation directory you can test during development by delploying the snapshot directly. The plugin will handle uploading a single script to a temp bucket before running in CloudFormation but if several scripts are present that reference each other such as the app.yaml and app-main.yaml pattern you must publish first before deploying. To publish

local-config.groovy example

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

Version

1. While in development you can skip this step and deploy/update until you're satified with the state of your component. Prior to using in another project create a release version using the command `./gradlew release`. This will ask two questions (release version and next snapshot version) which you'll probably just accept the defaults and then it will tag, publish and advance to the next snapshot version for future development.

Deploy

To deploy a stack use the create or update stack commands. `./gradlew updateStack`.

Update

To update a stack you can either run the updateStack command to immediately execute the changes or you can run the createChangeSet task first at which point you can inspect the changes that will occur in a future update but without actually applying the update in the first step. If you're happy with the change set you can then execute to apply the update.

Project Structure

The cfn-stacks example project explains all of the project files in more detail but for purposes of illustration of plugin functionality the project file structure is shown below.

```
├──── LICENSE
├──── build
│    └──── cloudformation
│        ├──── app-main.yaml
│        └──── app.yaml
├──── build.gradle
├──── gradle/
├──── gradlew
├──── gradlew.bat
├──── local-config.groovy
├──── settings.gradle
└──── src/
    ├──── docs/
    │    └──── asciidoc
    │        ├──── index.adoc
    │        └──── stylesheets
    │            └──── style.css
    └──── main/
        └──── cloudformation
            ├──── app-main.yaml
            └──── app.yaml
```

Plugin Tasks

build

The build task will compare the files in the build/ and src/ directories and rebuild if the /src files have been updated since the last run. For CloudFormation template projects a build consists of running the copy and filter task which replaces tokens found in the templates in the src/ directory with values from the project.

clean

The clean task will remove the build/ directory and all its contents. As Gradle is good about detecting if a build needs to occur based on file contents you don't need to run the clean command in between builds.

docs

Generate the HTML and PDF documentation to build/asciidoc/[html5|pdf] from Asciidoctor markup in src/docs/asciidoc

projectParams

Prints the parameter names and values that will be passed into the stack template. This task can be useful to check the state of overrides when setting up build.gradle, local-config.groovy and CLI properties.

projectVersion

Prints the version of the project as found in the gradle.properties file. Useful for scripted tasks like the publishing of documentation when the version captured as part of the URL.

publish

Runs the build task and then zips up the resulting templates into a *.cfn.jar file which is then pushed to an ArtifactS3 repo (S3 bucket).

release

Wraps the Gradle Release Plugin to perform the following steps:

- Checks for any un-committed files (Added, modified, removed, or un-versioned).
- Checks for any incoming or outgoing changes.
- Removes the SNAPSHOT flag on your project's version (If used)
- Prompts you for the release version.

- Checks if your project is using any SNAPSHOT dependencies
- Builds the project.
- Commits the project if SNAPSHOT was being used.
- Creates a release tag with the current version.
- Prompts you for the next version.
- Commits the project with the new version.

By default the plugin will prompt for version information but the information can be sent in via parameters for automation/CI usage. The release task will automatically run the publish task.

createStack

An alias for updateStack. There is also a createStackAndWWait task that will not return until the operation has completed.

updateStack

Collect the set of parameters specified in the available config files and command line params and either launches a new CloudFormation stack or updates an existing stack. There is also a updateStackAndWWait task that will not return until the operation has completed.

deleteStack

Deletes an existing CloudFormation stack. There is also a deleteStackAndWWait task that will not return until the operation has completed.

createChangeSet

From the CloudFormation docs on Change Sets:

When you need to update a stack, understanding how your changes will affect running resources before you implement them can help you update stacks with confidence. Change sets allow you to preview how proposed changes to a stack might impact your running resources, for example, whether your changes will delete or replace any critical resources, AWS CloudFormation makes the changes to your stack only when you decide to execute the change set, allowing you to decide whether to proceed with your proposed changes or explore other changes by creating another change set.

executeChangeSet

To make the changes described in a change set to your stack, execute the change set

Building

Prerequisite

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

Build

The project uses the Gradle build tool and is configured to use the Gradle Wrapper utility. This means that Java is the only dependency of the project. To build and install the plugin locally run the command `./gradlew install` or `gradlew.bat install` if running from a Windows machine.

Documentation Links

Web

Source

Version

This documentation was generated for version 0.3.4 from commit 674316a.