AWS CodeBuild & UrbanCode Deploy

AWS DevOps

November 2019

An initiative to establish a connection from AWS CodePipeline to enterprise tools, most notably UrbanCode Deploy (UCD).

Table of Contents

Version 1.0: jonathan_scharf@fanniemae.com ; christopher_rai@fanniemae.com

Overview

Fannie Mae (FNMA), like many other companies, is beginning its cloud journey utilizing Amazon Web Services (AWS). One challenge along this journey is the integration of AWS with the DevOps tool-chain. FNMA is currently using UCD to deploy applications while ensuring the proper guardrails and gating are met along the path to production. The existing tool-chain allows Jenkins to push the build to UCD's CodeStation via an IBM plugin, here we replicate this process with AWS CodeBuild push to UCD.

At first glance, a REST call should be sufficient to accomplish this goal, however, UCD's REST API lacks the necessary commands. Additionally, enterprise standards and guidelines suggest steps which may be unnecessary if you are developing locally.

This documentation focuses on how to import artifacts into CodeStation through the AWS CodeBuild tool. This strategy is explained in the context of UCD, the overall process can be applied to integrate various tools with the AWS suite. This approach doesn't require restructuring of predefined deployment process. We aim to ensure the separation of the build and the deploy, meaning, we do not want the build to fail if the deployment fails. If you already have an application and deployment processes setup, then most of the work is already done! This will guide you to setup an extensible integration that allows you call UCD and other DevOps tools via AWS.

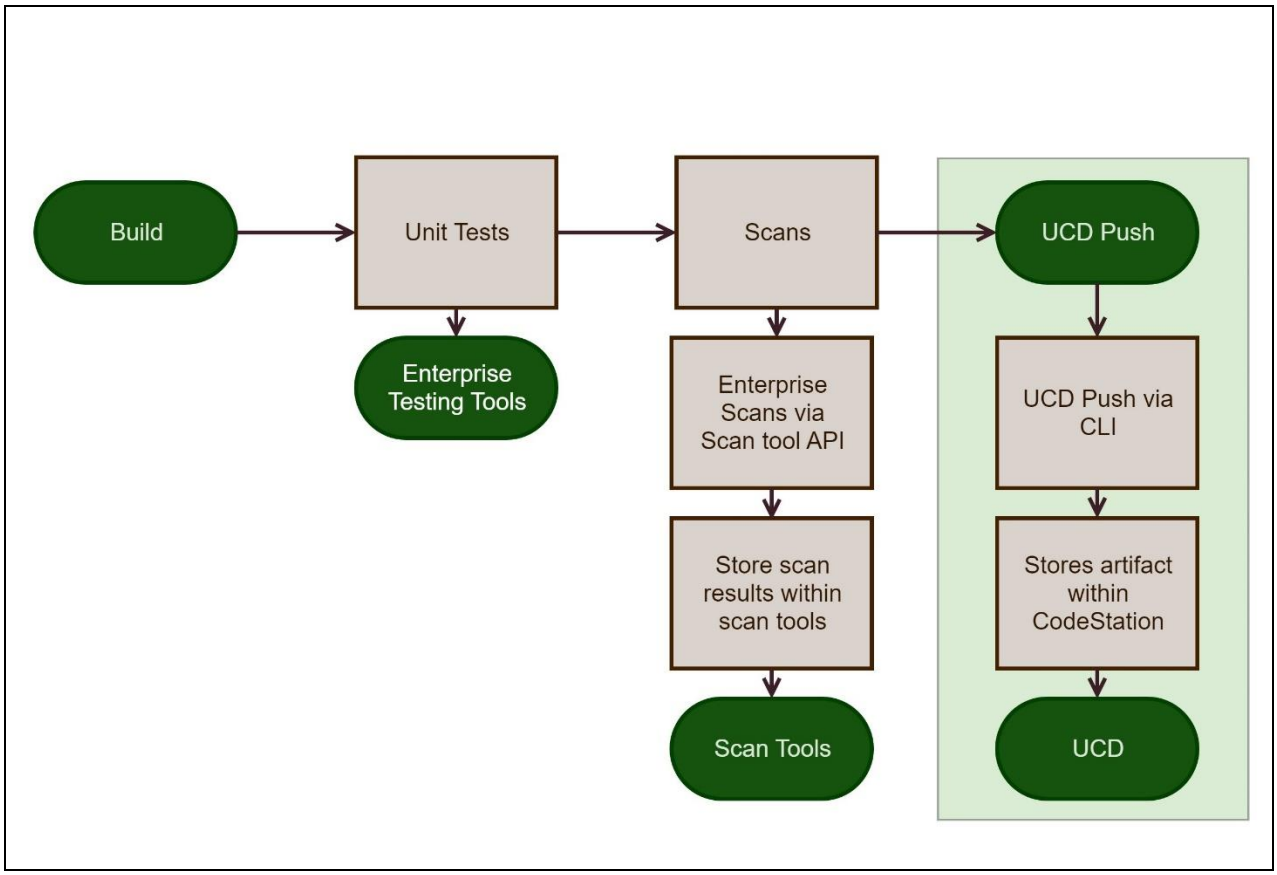
Problem

Integrate CI/CD tools, notably UCD with AWS CodeBuild.

Solution

Utilize UCD's API's to create the component version and upload the build artifact into that component version.

High-Level Design



Steps

1. **Access UCD API**: We utilize the udclient provided by IBM to access the UCD CLI. We have a separate pom file containing udclient to minimize impact on the application build process. Even though this file containing the udclient will be versioned, it will not be built with the application. The application's pom file will reference the udclient a dependency. This allows us to call the udclient within the post-build step just as you would on your own machine.
2. **Create UCD component version**: Here we utilize the *create version* command provided by IBM to create a version in UCD.
3. **Upload build** to the created component version: Here we utilize the *add version files* command provided by IBM to the newly created component version within UCD.
4. **Schedule deployment**: Here we utilize the *request application process* command to schedule a deployment just after the component is created with the build artifacts. We choose to pass a versioned json file as directed by IBM. This allows us to keep the separation between the build and the deployment. Once the deployment is scheduled the build will complete, regardless of the deployment's outcome. Within our application's deployment process we do use the *acquire* and *release lock* processes provided within UCD. We aim to avoid race conditions involving developers triggering build jobs simultaneously, resulting in multiple simultaneous scheduled deployments. This lock is based off the application process running on a specific environment.

Sample Execution

buildspec.yml:

```
version: 0.2
phases:
  install:
    commands:
      - cp ./settings.xml /root/.m2/settings.xml
      - set
  pre_build:
    commands:
      - TIMESTAMP_VERSION=$(TZ=":US/Eastern" date +%Y%m%d%H%M%S)
      - echo $TIMESTAMP_VERSION
      - GIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c1-8)
      - echo $GIT_HASH
  build:
    commands:
      - mvn -B package
  post_build:
    commands:
      - mvn -B dependency:resolve -f UCDpom.xml
      - echo "Creating component version $TIMESTAMP_VERSION-$GIT_HASH"
      - sed -i "s,\!VERSION_LABEL\!,$TIMESTAMP_VERSION-$GIT_HASH,g"
deploy.json
      - java -jar <PATH_TO>/udclient/7.0.3.1.1/udclient-7.0.3.1.1.jar -weburl
"<UCD_URL>" -authtoken "<TOKEN>" createVersion -component AWS-EB -name
$TIMESTAMP_VERSION-$GIT_HASH > /dev/null
      - java -jar <PATH_TO>/udclient/7.0.3.1.1/udclient-7.0.3.1.1.jar -weburl
"<UCD_URL>" -authtoken "<TOKEN>" createVersion -component AWS-EB -name
$TIMESTAMP_VERSION-$GIT_HASH > /dev/null
      - echo "Uploading artifact to component version $TIMESTAMP_VERSION-
$GIT_HASH"
      - java -jar <PATH_TO>/udclient/7.0.3.1.1/udclient-7.0.3.1.1.jar -weburl
"<UCD_URL>" -authtoken "<TOKEN>" addVersionFiles -component AWS-EB -version
$TIMESTAMP_VERSION-$GIT_HASH -base ./target -include acheck_api-package.zip
      - java -jar <PATH_TO>/udclient/7.0.3.1.1/udclient-7.0.3.1.1.jar -weburl
"<UCD_URL>" -authtoken "<TOKEN>" addVersionFiles -component FZM_Provisioning
-version $TIMESTAMP_VERSION-$GIT_HASH -base . -include awsManifest.json
      - java -jar <PATH_TO>/udclient/7.0.3.1.1/udclient-7.0.3.1.1.jar -weburl
"<UCD_URL>" -authtoken "<TOKEN>" requestApplicationProcess deploy.json
      - java -jar <PATH_TO>/udclient/7.0.3.1.1/udclient-7.0.3.1.1.jar -weburl
"<UCD_URL>" -authtoken "<TOKEN>" requestApplicationProcess provision.json
```

udclient pom:

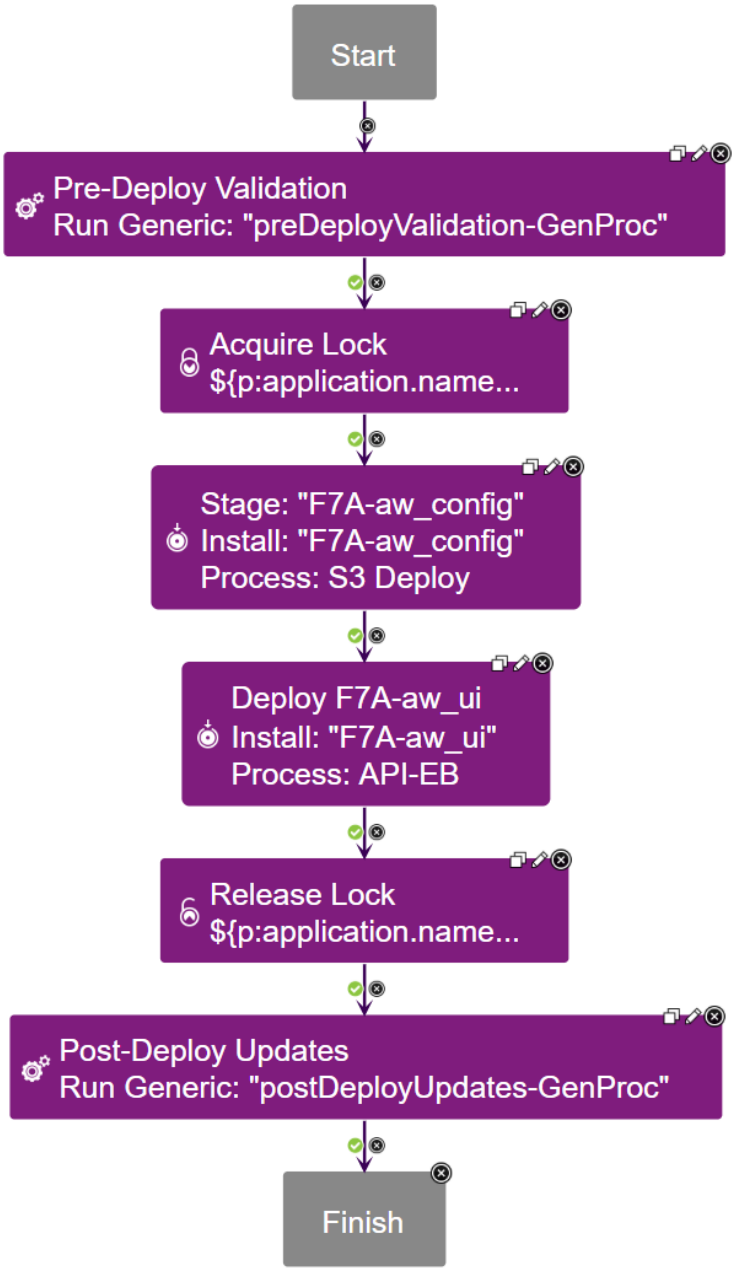
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.fanniemae.UCD</groupId>
    <artifactId>UCD</artifactId>
    <version>0.0.1</version>

    <dependencies>
        <dependency>
            <groupId>com.fanniemae.release-pipeline</groupId>
            <artifactId>udclient</artifactId>
            <version>7.0.3.1.1</version>
        </dependency>
        <dependency>
               <groupId>com.fanniemae.release-pipeline</groupId>
            <artifactId>nexus-iq-cli</artifactId>
            <version>1.63.0-01</version>
         </dependency>
    </dependencies>

</project>
```

deploy.json:

```
{
  "application": "AWS-FZM",
  "applicationProcess": "D10-App Deploy",
  "description": "DEVL Deploy triggered from AWS",
  "environment": "DEVL",
  "onlyChanged": "false",
  "properties": {
    "awsVersionLabel": "!VERSION_LABEL!"
  },
  "versions": [{
    "component": "AWS-EB",
    "version": "latest"
  }]
}
```

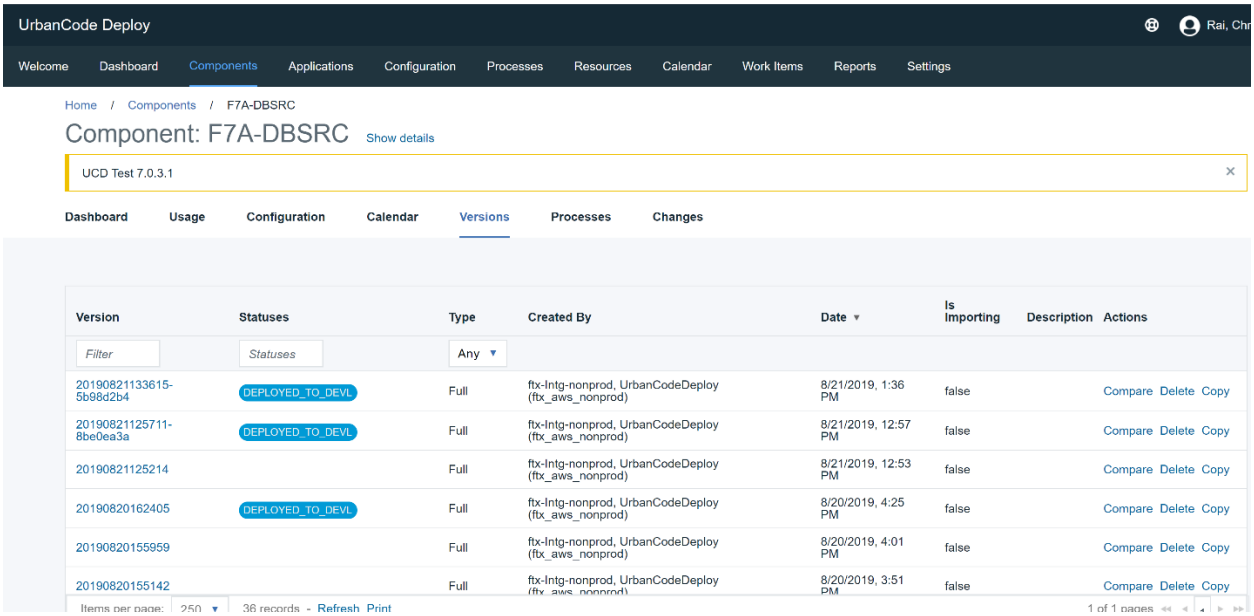
Deployment Process:

For triggering a deployment, we use the "latest" tag along with locks within the UCD process. Our AWS CodeBuild does not allow for parallel builds. With these two safeguards, we are confident that the component version we expect will always be deployed. Please note that the property within the lock is: ${p:application.name}-${p:applicationProcess.name}-${p:environment.name}.



Results

Within the example, please note the version name, we take advantage of the commit hash provided by git as well as the timestamp. We are using a sed command, as illustrated to return this result. The idea is to return a unique component version that will also provide tractability back to the baseline and source of the code change.



Notes

In this version of the integration, we did have to adjust our virtual private cloud (VPC), depending on your enterprise setup this may be necessary.

```
aws codebuild update-project --name <CODEBUILD_NAME> --vpc-config
file://vpc.json
{
        "vpcId": "vpc-xxxxxxxx",
                "subnets": [
                "subnet-xxxxxxxx"
                ],
        "securityGroupIds": [
                "sg-xxxxxxxx"
                ]
}
```

This is just one approach to integrate AWS' CodeBuild with IBM's UrbanCode Deploy. While utilizing a pom file may not be ideal, it allows for other tools to be integrated into AWS, such as Nexus IQ, SonarQube, Fortify, and Twistlock scans.

While this strategy is not necessary to automate the push from AWS to UCD, we did investigate other possible options:

- **Dependent POM**: In order to keep the number of committed files down, the udclient can be referenced within the application pom file as a excluded dependency.
- **Call via wget**: To avoid any extraneous pom files, one can perform the UCD commands via wget where applicable.
- **Buildscript**: To clean up the buildspec.yml, a buildscript can be used in its place.
- **UCD api**: Hopefully, the api will be extended to support the necessary cli commands to make the udclient unnecessary.

Links

https://www.ibm.com/support/knowledgecenter/SS4GSP_7.0.3/com.ibm.udeploy.doc/ucd_version_welcome.html

https://aws.amazon.com/codebuild/