Pipeline Scripts

- A Pipeline is a user-defined model of a CD pipeline. A Pipeline's code defines your entire build process, which typically includes stages for building an application, testing it and then delivering it
- Jenkins with its rich set of plugins provides many ways to create a CI/CD pipeline, pipeline-as-code is just one of the many; and the most prominent approach to that.
- The pipeline code can either be written in a Jenkins job (using the Groovy Plugin) or as a Jenkinsfile and committing it to the source-control.
- Pipeline script skeleton differs for **Declarative Pipeline syntax** and **Scripted Pipeline syntax.**

Declarative Pipeline

```
pipeline {                          # a pipeline block is a key part of Declarative Pipeline syntax

    agent any                       #Execute this Pipeline or any of its stages, on any available agent.

    environment {                   #The environment directive specifies a sequence of key-value pairs which will be defined as environment variables for the all steps, or stage-specific steps, depending on where the environment directive is located within the Pipeline.
        CC = 'clang'
    }

    tools {                         # A section defining tools to auto-install and put on the PATH.
        maven 'apache-maven-3.0.1'
    }

    stages {
        stage('Build') {            #A stage block defines a conceptually distinct subset of tasks performed through the entire Pipeline (e.g. "Build", "Test" and "Deploy" )

            steps {                 # a step tells Jenkins what to do at a particular
```

point in time. For example, to execute the shell command make use the sh step: sh 'make'

```
            //
        }
    }
    stage('Test') {
        steps {
            //
        }
    }
    stage('Deploy') {
        steps {
            //
post {
        always {
                echo 'I will always say Hello again!'
                }
        }
    }
  }
}
```

- The amount of sophistication that can be brought in the pipelines is very much dependent on the use of sessions , the list goes long, **refer Pipeline Syntax in jenkins.io.** (https://jenkins.io/doc/book/pipeline/syntax/)

- ***Scripted Pipeline***

```
node {                                  # In Scripted Pipeline syntax, one or more node blocks do the core work throughout the entire Pipeline.
    stage('Build') {
    //
}
    stage('Test') {
    //
}
    stage('Deploy') {
    //
}
}
```

- The concept of stages and sessions remains unchanged in both scripted and declarative pipelines.

• A working DevOps Pipeline (Canon CI/CD)

```
node {
   cleanWs()
   stage '1. Git checkout ContentAPI'
      git branch: 'develop', credentialsId: 'continuous-delivery-bitbucket', url:
'https://Continuous.Delivery@canon-europe-bitbucket.valiantys.net/scm/cmp/content-
api.git'
      dir('PROPERTIES') {                    # in below git checkout will be performed in the
                                               PROPERTIES directory
         git credentialsId: 'continuous-delivery-bitbucket', url:
'https://Continuous.Delivery@canon-europe-
bitbucket.valiantys.net/scm/cmp/properties.git'
      }

   stage '2. Prepare Environment'
      tool name: 'Maven 3.3.9', type: 'maven'
      env.JAVA_HOME="${tool 'OpenJDK 1.8'}"
      env.PATH="${env.JAVA_HOME}/bin:${env.PATH}"

   stage '3. Add <distributionManagement>'
      dir('contentapiapp') {
         sh """                              # sh here indicates a bash script will ensue, bat is used for
                                               windows
         echo '<distributionManagement>
            <snapshotRepository>
              <id>snapshots</id>
              <name>Snapshots</name>
<url>http://nexus:8081/nexus/content/repositories/snapshots/ContentApiStaging</url>
            </snapshotRepository>
          </distributionManagement>
         </project>' > append
```

```
sed -i 's|</project>||g' pom.xml          #since while adding <distributionManagement>
    cat append >> pom.xml                  the </project>is added, the sed(stream Editor):
                                           a cmd used to edit a file w/o opening it is used
                                           to replace existing </project> with a white space
                                          's(used for substitution)| search pattern| replacement| g- global
                                           replacement(replaces all occurence in the give pattern'

      tail -n 50 pom.xml                  #prints the last 50 lines of pom.xml
      """
      sh "cat pom.xml"
    }

  stage '4. Execute Unit Tests'
    withMaven(globalMavenSettingsConfig: 'a3c77df0-d448-414b-8c6c-
4879598a99e1', jdk: 'OpenJDK 1.8', maven: 'Maven 3.3.9') {
      dir('contentapiapp') {
        sh "mvn clean install -X"
      }
    }

  stage '5. SonarQube analysis'
    dir('contentapiapp') {
      def ACCESS_TOKEN = 'a3ed1f9fb8fd4639b97fa4d8a4e9af1452bbe146';
      def SONAR_URL = 'http://172.31.82.116:9000'
      def scannerHome = tool 'SonarQube scanner';
      def pom = readMavenPom file: 'pom.xml';
      sh """
        ${scannerHome}/bin/sonar-scanner \
          -Dsonar.host.url=${SONAR_URL} \
          -Dsonar.login=${ACCESS_TOKEN} \
          -Dsonar.projectKey=${pom.artifactId} \
          -Dsonar.projectName=${pom.artifactId} \
          -Dsonar.projectVersion=${pom.version} \
          -Dsonar.sources=. \
          -Dsonar.java.binaries=. \
          -Dsonar.java.source=1.8 \
          -Dsonar.language=java \
          -X
      """
    }

  stage '6. Make project comply with DEV_STAGING mode'
    dir('contentapiapp') {
      sh """
```

```
            sed -i '/<artifactId>/ s/contentapi/contentapi-Staging/' pom.xml
            cp ../PROPERTIES/cms/dev/staging/cd_client_conf.xml content-
api/src/main/resources/cd_client_conf.xml
            ls -la content-api/src/main/resources/
          """
       }

   stage '7. Build and deploy contenapi-STAGING to Nexus Snapshots repository'
       withMaven(globalMavenSettingsConfig: 'a3c77df0-d448-414b-8c6c-
4879598a99e1', jdk: 'OpenJDK 1.8', maven: 'Maven 3.3.9') {
          dir('contentapiapp') {
             sh "mvn deploy"
          }
       }

   stage '8. Deploy contentapi-STAGING in DEV_API machine'
       dir('contentapiapp') {
          def pom = readMavenPom file: 'pom.xml';
          build job: 'CONTENT_API_DEV_STAGING_DEPLOYMENT', parameters:
[string(name: 'DEPLOYMENT_VERSION', value: pom.version)]
       }
}
```