**Coverage Checker CI & CD**

Coverage Checker Application consists of the below components:

CCSL

CCUI

3gin

3gout

4gin

4gout

uk

Dev Team has provided only the below components:

CCSL

CCUI

3G

4G

UK

Hence the document explains only the CI and CD part of components that are available till date 22nd Feb, 2018.

CCSL component is built using Maven build tool and an image is generated using Docker Build. All the other components make use of only Docker Builds. All the generated images are deployed using Kubernetes.

Below is the flow from Code commit to Deployment in Coverage Checker:

git-commit -> build Docker file and generate image -> login azure cloud registry -> push docker image -> delete current deployment -> deploy the new image using kubernetes

**List of Environments**

DEV-A

Master rbdeachcovamw00 10.71.101.27

Node rbdeachcovaaw00 10.71.101.20

Node rbdeachcovaaw01 10.71.101.21

DEV-B

Master rbdecchcovamw00 10.71.101.38

Node rbdecchcovaaw00 10.71.101.29

Node rbdecchcovaaw01 10.71.101.34

DEV-C

Master rbdecchcovamw00 10.71.101.30

Node rbdecchcovaaw00 10.71.101.28

Node rbdecchcovaaw01 10.71.101.22

CIT-A

Master rbciachcovamw00 10.70.101.38

Node rbciachcovaaw00 10.70.101.39

Node rbciachcovaaw01 10.70.101.40

CIT-C

Master rbcicchcovamw00 10.70.101.41

Node rbcicchcovaaw00 10.70.101.42

Node rbcicchcovaaw01 10.70.101.43

SIT-A

Master rbsiachcovamw00 172.28.196.12

Node rbsiachcovaaw00 172.28.196.13

Node rbsiachcovaaw01 172.28.196.14

**List of Agents**

Coverage Checker uses Kubernetes cluster type of environment. Hence UrbanCode Deploy agents are installed only on master for deployments and master will distribute the components to nodes.

DEV-A Channels-CC.DEV-A.10.71.101.27

DEV-B Channels-CC.DEV-B.Master.10.71.101.38

DEV-C Channels-CC.DEV-C.Master.10.71.101.30

CIT-A Channels-CC.CIT-A.Master.10.70.101.38

CIT-C Channels-CC.CIT-C.Master.10.70.101.41

**Build Strategy:**

CCSL component is built using Maven. Use “clean package” goal in Jenkins to build CCSL. Below are the commands used in Jenkins for generating docker images including Git Commit ID in the tag name and for pushing the images to Azure Container Registry for all the components:

echo $GIT\_COMMIT

echo $POM\_VERSION

cd ${WORKSPACE}/CCSL

sudo docker build -t rbdevchacr01.azurecr.io/dkori/ccsi:$GIT\_COMMIT -f Dockerfile .

sudo docker login rbdevchacr01.azurecr.io -u ${acrUserName} -p ${acrPassword}

sudo docker push rbdevchacr01.azurecr.io/dkori/ccsi:$GIT\_COMMIT

cd ${WORKSPACE}/UI

sudo docker build -t rbdevchacr01.azurecr.io/dkori/ui:$GIT\_COMMIT -f Dockerfile .

sudo docker push rbdevchacr01.azurecr.io/dkori/ui:$GIT\_COMMIT

cd ${WORKSPACE}/tilserver/3G

sudo docker build -t rbdevchacr01.azurecr.io/dkori/3g:$GIT\_COMMIT -f Dockerfile .

sudo docker push rbdevchacr01.azurecr.io/dkori/3g:$GIT\_COMMIT

cd ${WORKSPACE}/tilserver/4G

sudo docker build -t rbdevchacr01.azurecr.io/dkori/4g:$GIT\_COMMIT -f Dockerfile .

sudo docker push rbdevchacr01.azurecr.io/dkori/4g:$GIT\_COMMIT

cd ${WORKSPACE}/tilserver/UK

sudo docker build -t rbdevchacr01.azurecr.io/dkori/uk:$GIT\_COMMIT -f Dockerfile .

sudo docker push rbdevchacr01.azurecr.io/dkori/uk:$GIT\_COMMIT

**Note:**

1.Credentials required to login to ACR(acrUserName and acrPassword) are set as variables in Jenkins using Bindings plugin in order to avoid using credentials in plain text.

2. Docker commands should be executed with “sudo” access.

**Deployment Strategy:**

UCD plugin is configured in Jenkins in order to push the files all\_azure.yml, azure\_redep.sh to UCD code station. Commit ID is passed to UCD as a version property in order to modify the yml file with Commit ID.

A yaml file(all\_azure.yml) has been developed that will fetch the images from ACR with the given tag name and creates deployments & services accordingly. Below is the example:

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: ccsi

spec:

replicas: 1

template:

metadata:

labels:

run: ccsi

spec:

containers:

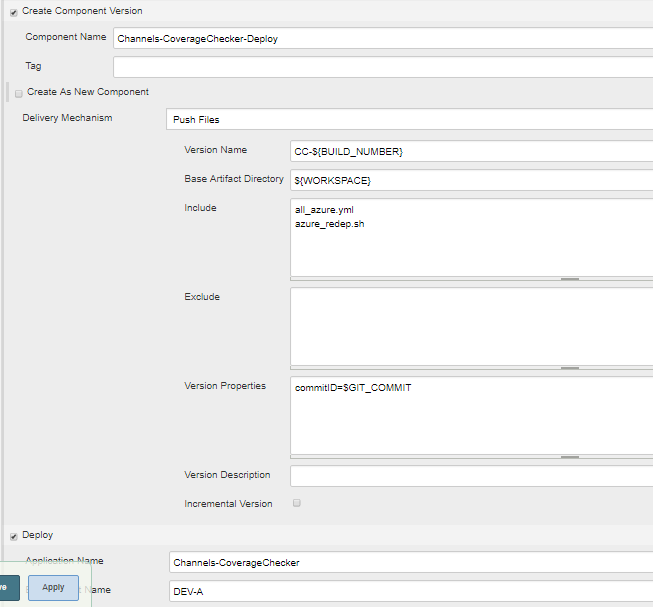
- name: ccsi

image: rbdevchacr01.azurecr.io/dkori/ccsi:@commitID@

ports:

- containerPort: 8080

Add the token “@commitID” in the yaml file, which will be replaced with the Git Commit ID value in UrbanCode Deploy. This facilitates the yml file to fetch the images with specified Commit ID.



**Steps involved in Deployment**

Clean working directory : Clean the working directory

Download Artifacts : Download the files on target server

Change permissions : Change the permissions of downloaded files

Delete existing deployments : Delete existing deployments and services

Modify yml file : Replace GIT\_COMMIT tokens with actual CommitID

Run deployment : Deploy components

**Details of Deployment steps:**

**Clean working directory**

Clean the working directory

Working Directory : /appdata01/cc\_ucd\_deployments

**Download Artifacts**

Download the files on target server

Working Directory : /appdata01/cc\_ucd\_deployments

**Change permissions**

Change the permissions of downloaded files

chmod -Rf 777 /appdata01/cc\_ucd\_deployments

**Delete existing deployments**

Delete existing deployments and services

Working Directory : /appdata01/cc\_ucd\_deployments

Script:

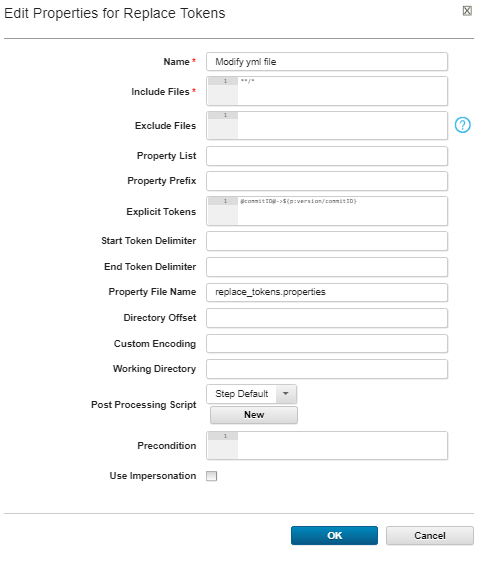
sh -x azure\_redep.sh

**Modify yml file**

Replace GIT\_COMMIT tokens with actual CommitID

Working Directory : /appdata01/cc\_ucd\_deployments

@commitID@->${p:version/commitID}



**Run deployment**

Deploy components

Working Directory : /appdata01/cc\_ucd\_deployments

Script:

kubectl create -f all\_azure.yml

**CI and CD Pipeline**

**List of Jobs**

Channels-CoverageChecker-Daily-Build : To be used for daily builds & deployments to DEV-A Environment and can be scheduled

Channels-CoverageChecker-DEV-A-Pipeline : To be used for promotions to higher env

Channels-CoverageChecker-CIT-A-Pipeline

Channels-CoverageChecker-CIT-B-Pipeline

Channels-CoverageChecker-CIT-C-Pipeline

Channels-CoverageChecker-SIT-A-Pipeline

Pending:

Waiting for new Windows VM with docker installed

Once we get the VM, need to install Jenkins slave and tie all the Coverage checker jobs to that slave

Dev team has to check in the latest code into BitBucket

Updated yml file is in devops-test branch---please use this