**AEM-CI & CD**

**AEM Application:**

AEM is a Java based application. Ui.apps and ui.content are the two zip artifacts generated using Maven build.

**Build Artifacts**

three-rebus.ui.apps-1.0.zip

three-rebus.ui.content-1.0.zip

AEM build artifacts are built and deployed continuously to the AEM server on different environments using Jenkins and UrbanCode Deploy.

**Environments and agents in AEM:**

All the environments are Redhat Linux 7.3 servers

STUBDEV : Channels-AEM.STUBDEV-Author.10.40.101.7

DEV-A : Channels-AEM.DEV-A-Author.10.71.101.10

DEV-C : Channels-AEM.DEV-C-Author.10.71.101.42

CIT-A : Channels-AEM.CIT-A-Author.10.70.101.23

CIT-B : Channels-AEM.CIT-B-Author.10.70.101.24

CIT-C : Channels-AEM.CIT-C-Author.10.70.101.31

SIT-A :

**Note:** Set the below properties on agents

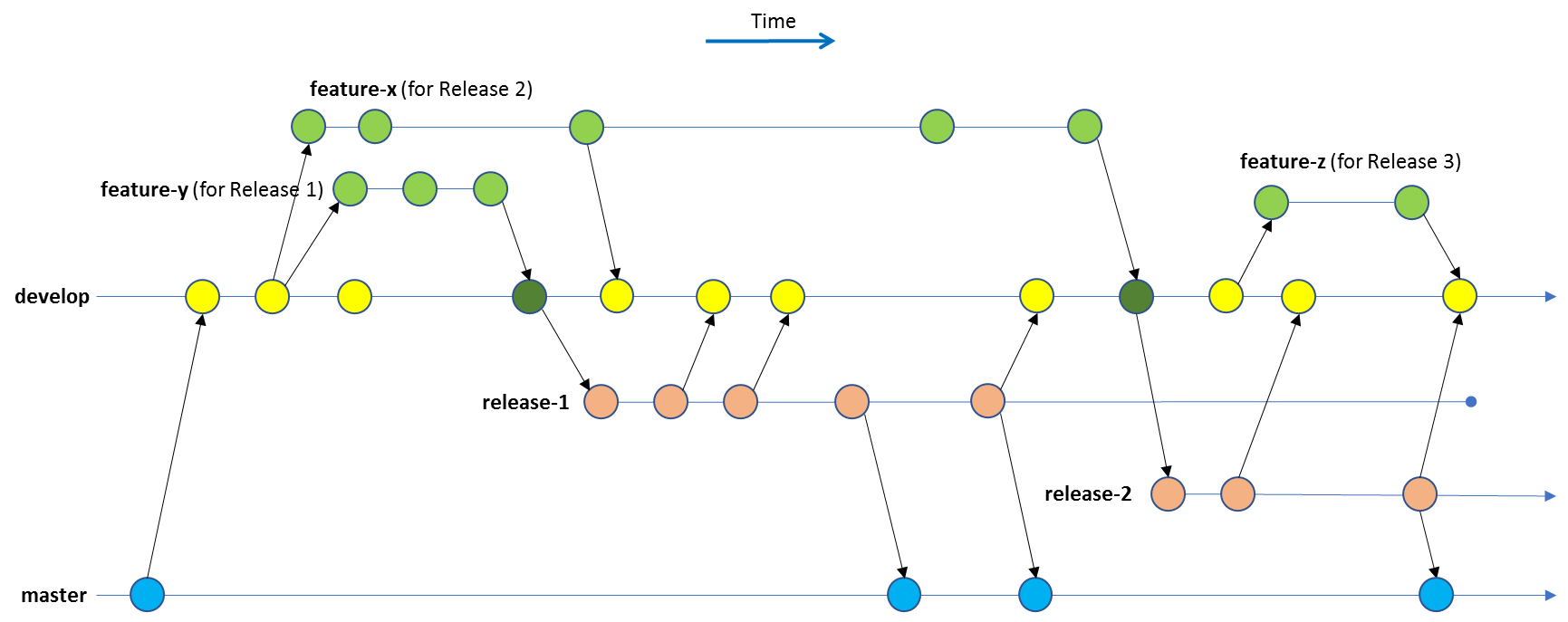
**Properties to be set**

aemPort

aemUserName

aemUserPassword

**Branching Strategy in BitBucket:**



Feature development takes place on the “develop” branch; and specific feature branches can be used to facilitate working in parallel.

develop branch used :

Functional-Drop

feature branches used :

feature/<release-name> feature/FD2, feature/FD3. . .

Developers push the code to the feature branch corresponding to the ongoing release. Jenkins pipeline for daily builds and deployments to stubdev is configured with feature branch to run twice a day.

When the development for the next release is complete (i.e. all features are code-complete, unit tested and merged to develop) the release branch for the next release is created off the develop branch.

Deployment to CIT-A (functional testing), CIT-B (NFT), CIT-C (end to end testing) and Dev-C (defects) can now take place from the release branch.

Release branches used :

release/<release-name> release/FD2, release/FD3

Any change on the release branch is merged to develop, to keep it up to date.

As tests are run on the CIT environments, defects will be found. These will be fixed on the release branch, or on a bugfix branch off the release branch.

A tag will be applied to the fix; and the tagged package will be deployed to Dev-C to run verification and regression tests.

Then the fixed code can be deployed to the CIT environments for further verification and regression testing.

When the CIT testing phase is done, the code on the release branch is deployed to the SIT testing environments.

Any bugs found in SIT are again fixed on the release branch; tagged and deployed to Dev-C for unit testing; deployed to CIT + SIT for verification and regression testing – and so on.

When all the testing is complete, the release branch is tagged, and merged to master for deployment to production.

Master branch used:

master

**Build Strategy:**

AEM build artifacts are generated using Maven build. Jenkins job is configured to execute Maven builds on the source code fetched from BitBucket repository. Auto version increment of the format x.x.x is implemented using Shell script in Jenkins.

file\_path="/appdata/jenkins-slave"

properties\_name="aem\_version.properties"

file\_name="Dev-A-Pipeline-version-update.txt"

full\_path=$file\_path/$file\_name

properties\_path=$file\_path/$properties\_name

if [ ! -f $full\_path ]; then

touch $full\_path

echo "1.0.0" > $full\_path

echo "c: $full\_path"

else

echo "File exist $full\_path"

fi

version\_number=`cat $full\_path`

echo "$version\_number"

last\_digit=`echo $version\_number | awk -F "." '{print $NF}'`

echo $last\_digit

last\_digit=$(($last\_digit+1))

remaining=`echo "${version\_number::-2}"`

middle\_digit=`echo $remaining | awk -F "." '{print $NF}'`

echo $middle\_digit

first\_digit=`echo "${version\_number::-4}"`

echo $first\_digit

if [[ $last\_digit -eq 10 ]];then

last\_digit=0

middle\_digit=$(($middle\_digit+1))

fi

if [[ $middle\_digit -eq 10 ]];then

middle\_digit=0

first\_digit=$(($first\_digit+1))

fi

if [[ $first\_digit -eq 10 ]];then

last\_digit=1

middle\_digit=0

first\_digit=0

fi

complete\_string=$first\_digit.$middle\_digit.$last\_digit

echo $complete\_string > $full\_path

chmod 755 $full\_path

version=$complete\_string

echo "Version: $version"

if [ ! -f $properties\_path ]; then

touch $properties\_path

echo "Version=$version" > $properties\_path

echo "c: $properties\_path"

else

echo "File exist $properties\_path"

rm -rf $properties\_path

touch $properties\_path

echo "Version=$version" > $properties\_path

echo "c: $properties\_path"

fi

cat $properties\_path

#mvn clean package -Dproject.version=$version

mvn build-helper:parse-version versions:set -DnewVersion=$version

**Deployment Strategy:**

Deployment of build artifacts to AEM server has been achieved through curl commands.

**Steps involved in the deployment:**

Check the type of version : check the type(feature, bugfix) of version

getOldVersion : read the existing version of artifacts from the artifacts locations and store in properties

Rebuild existing Content Package : rebuild existing version of Content package

Create a backup directory if it doesn't exist : Create a base backup directory under which backup directories with timestamp are created

getDate : get timestamp for creating backup directories with timestamp

Create Build Backup directory with timestamp : Create backup directory with timestamp for rebuilt content package

Create backup directories with timestamp : Create backup directory with timestamp for existing artifacts

BackUp of rebuilt content package : Download rebuild content package to the build backup directory

Backup of existing artifacts : Take backup of existing artifacts to the backup directory

Change permission of backup folder : change permissions of copied the artifacts

Delete existing packages : Delete existing artifacts

Download Packages and Bundles : Download new artifacts

Change permissions : Change permissions of downloaded artifacts

getNewVersion : fetch the version of new artifacts

Install apps Package : Deploy apps package

Wait before retrying apps installation : If the deployment of apps is failed, wait for 5 mins before trying to deploy again

Install apps Package-Retry : Retry deploying apps package

Replicate Apps bugfix : If version is bugfix, then replicate apps soon after deploying apps package

Install content package : If version is feature type, the deploy content package

Replicate Apps : Replicate apps package

Wait before retrying the installation : If content deployment is failed, wait for 5 mins before trying to deploy it again

Install content package--retry 1 : Retry deploying the content package

Delete current version on failure : If content package deployment fails for the second time, delete current version of artifacts

restore old artifacts : restore old version of artifacts

Change permissions for restored artifacts : Change permissions for restored artifacts

Rollback apps : rollback apps to old version

Wait before retrying apps installation : If rollback of apps fails, wait for few mins before trying to rollback again

Rollback apps-retry : retry to deploy apps old version

**Check the type of version**

Check the type(feature, bugfix) of the version using switch, as the deployment process is determined based on the type of version.

${p:version/type}

**getOldVersion**

Get the old version of the artifacts

script:

cd /appdata01/AEM\_Artifacts/ui.apps/target

for i in \*.zip; do { echo "apps=$i"; x=$i; y=${x%.zip};echo "apps.name=${y##\*/}";}; done

cd /appdata01/AEM\_Artifacts/ui.content/target

for i in \*.zip; do { echo "content=$i"; x=$i; y=${x%.zip};echo "content.name=${y##\*/}";}; done

post processing script to store the old version:

var exit = properties.get('exitCode');

scanner.register("apps=", function(lineNumber, line) {

var value=line.replace("apps=","");

properties.put("appsOldVersion",value);

});

scanner.register("apps.name=", function(lineNumber, line) {

var value=line.replace("apps.name=","");

properties.put("appsOldVersionName",value);

});

scanner.register("content=", function(lineNumber, line) {

var value=line.replace("content=","");

properties.put("contentOldVersion",value);

});

scanner.register("content.name=", function(lineNumber, line) {

var value=line.replace("content.name=","");

properties.put("contentOldVersionName",value);

});

if (exit == 0) {

properties.put('Status', 'Success');

}

else {

properties.put('Status', 'Failure');

}

scanner.scan();

**Rebuild existing Content Package**

curl -u ${p:agent/aemUserName}:${p:agent/aemUserPassword} -X POST "https://${p:agent/ip}:${p:agent/aemPort}/crx/packmgr/service/.json/etc/packages/Three Rebus Content Package/${p:getOldVersion/contentOldVersion}?cmd=build" –k

**Create a backup directory if it doesn't exist**

Working Directory : /appdata01

[ -d aem\_build\_backup ] || mkdir aem\_build\_backup

chmod -Rf 777 aem\_build\_backup

[ -d aem\_backup ] || mkdir aem\_backup

chmod -Rf 777 aem\_backup

**getDate**

script:

echo DATE=$(date '+%Y-%m-%d\_%H:%M:%S')

post-processing script:

var exit = properties.get('exitCode');

if (exit == 0) {

properties.put('Status', 'Success');

scanner.register("DATE=", function(lineNumber, line) {

var value=line.replace("DATE=","");

properties.put("backup\_date",value);

});

scanner.scan();

}

else {

properties.put('Status', 'Failure');

}

**Create Build Backup directory with timestamp**

Create build backup directory with timestamp under base backup directory “aem\_build\_backup”

Working Directory : /appdata01/aem\_build\_backup

${p:getDate/backup\_date}

**Create backup directories with timestamp**

Create backup directory with timestamp under base backup directory “aem \_backup”

Working Directory : /appdata01/aem\_backup

Script:

${p:getDate/backup\_date}

**BackUp of rebuilt content package**

Download rebuild content package to the build backup directory

Script:

curl -u ${p:agent/aemUserName}:${p:agent/aemUserPassword} "https://${p:agent/ip}/etc/packages/Three%20Rebus%20Content%20Package/${p:getOldVersion/contentOldVersion}" -o "/appdata01/aem\_build\_backup/${p:getDate/backup\_date}/${p:getOldVersion/contentOldVersion}" –k

**Backup of existing artifacts**

Take backup of existing artifacts to the backup directory

Script:

cd /appdata01/AEM\_Artifacts/ui.apps/target

if [ -e \*.zip ]

then

cp /appdata01/AEM\_Artifacts/ui.apps/target/${p:getOldVersion/appsOldVersion} /appdata01/aem\_backup/${p:getDate/backup\_date}

cp /appdata01/AEM\_Artifacts/ui.content/target/${p:getOldVersion/contentOldVersion} /appdata01/aem\_backup/${p:getDate/backup\_date}

else

echo "no existing artifacts"

fi

**change permission of backup folder**

change permissions of copied the artifacts

Script:

chmod -Rf 777 /appdata01/aem\_build\_backup

chmod -Rf 777 /appdata01/aem\_backup

**Delete existing packages**

Working Directory : /appdata01/AEM\_Artifacts

ui.apps/target/\*

ui.content/target/\*

**Download Packages and Bundles**

Download artifacts to the target directory

Working Directory : /appdata01/AEM\_Artifacts

**Change permissions**

Change permissions of downloaded artifacts

chmod -Rf 775 /appdata01/AEM\_Artifacts

Working Directory : /appdata01/AEM\_Artifacts

**getNewVersion**

cd /appdata01/AEM\_Artifacts/ui.apps/target

for i in \*.zip; do { echo "apps=$i"; x=$i; y=${x%.zip};echo "apps.name=${y##\*/}";}; done

cd /appdata01/AEM\_Artifacts/ui.content/target

for i in \*.zip; do { echo "content=$i"; x=$i; y=${x%.zip};echo "content.name=${y##\*/}";}; done

post processing script:

var exit = properties.get('exitCode');

scanner.register("apps=", function(lineNumber, line) {

var value=line.replace("apps=","");

properties.put("appsNewVersion",value);

});

scanner.register("apps.name=", function(lineNumber, line) {

var value=line.replace("apps.name=","");

properties.put("appsNewVersionName",value);

});

scanner.register("content=", function(lineNumber, line) {

var value=line.replace("content=","");

properties.put("contentNewVersion",value);

});

scanner.register("content.name=", function(lineNumber, line) {

var value=line.replace("content.name=","");

properties.put("contentNewVersionName",value);

});

if (exit == 0) {

properties.put('Status', 'Success');

}

else {

properties.put('Status', 'Failure');

}

scanner.scan();

**Install apps Package**

curl -u ${p:agent/aemUserName}:${p:agent/aemUserPassword} -F file=@"/appdata01/AEM\_Artifacts/ui.apps/target/${p:getNewVersion/appsNewVersion}" -F name="${p:getNewVersion/appsNewVersionName}" -F force=true -F install=true https://${p:agent/ip}:${p:agent/aemPort}/crx/packmgr/service.jsp -k

post processing script;

var exit = properties.get('exitCode');

properties.put('Status', 'Success');

if (exit == 0) {

properties.put('Status', 'Failure');

scanner.register('status code="200"', function(lineNumber, line) {

properties.put('Status', 'Success');

});

scanner.scan();

}

else {

properties.put('Status', 'Failure');

}

**Install apps Package-Retry**

curl -u ${p:agent/aemUserName}:${p:agent/aemUserPassword} -F file=@"/appdata01/AEM\_Artifacts/ui.apps/target/${p:getNewVersion/appsNewVersion}" -F name="${p:getNewVersion/appsNewVersionName}" -F force=true -F install=true https://${p:agent/ip}:${p:agent/aemPort}/crx/packmgr/service.jsp -k

post processing script;

var exit = properties.get('exitCode');

properties.put('Status', 'Success');

if (exit == 0) {

properties.put('Status', 'Failure');

scanner.register('status code="200"', function(lineNumber, line) {

properties.put('Status', 'Success');

});

scanner.scan();

}

else {

properties.put('Status', 'Failure');

}

**Install content package**

curl -u ${p:agent/aemUserName}:${p:agent/aemUserPassword} -F file=@"/appdata01/AEM\_Artifacts/ui.content/target/${p:getNewVersion/contentNewVersion}" -F name="${p:getNewVersion/contentNewVersionName}" -F force=true -F install=true https://${p:agent/ip}:${p:agent/aemPort}/crx/packmgr/service.jsp -k

post processing script:

var exit = properties.get('exitCode');

properties.put('Status', 'Success');

if (exit == 0) {

properties.put('Status', 'Failure');

scanner.register('status code="200"', function(lineNumber, line) {

properties.put('Status', 'Success');

});

scanner.scan();

}

else {

properties.put('Status', 'Failure');

}

**Install content package--retry 1**

curl -u ${p:agent/aemUserName}:${p:agent/aemUserPassword} -F file=@"/appdata01/AEM\_Artifacts/ui.content/target/${p:getNewVersion/contentNewVersion}" -F name="${p:getNewVersion/contentNewVersionName}" -F force=true -F install=true https://${p:agent/ip}:${p:agent/aemPort}/crx/packmgr/service.jsp -k

post processing script:

var exit = properties.get('exitCode');

properties.put('Status', 'Success');

if (exit == 0) {

properties.put('Status', 'Failure');

scanner.register('status code="200"', function(lineNumber, line) {

properties.put('Status', 'Success');

});

scanner.scan();

}

else {

properties.put('Status', 'Failure');

}

**Delete current version on failure**

If content package deployment fails for the second time, delete current version of artifacts

Working Directory : /appdata01/AEM\_Artifacts

ui.apps/target/\*

ui.content/target/\*

**restore old artifacts**

cp /appdata01/aem\_backup/${p:getDate/backup\_date}/${p:getOldVersion/appsOldVersion} /appdata01/AEM\_Artifacts/ui.apps/target

cp /appdata01/aem\_backup/${p:getDate/backup\_date}/${p:getOldVersion/contentOldVersion} /appdata01/AEM\_Artifacts/ui.content/target

**Change permissions for restored artifacts**

Working Directory : /appdata01/AEM\_Artifacts

chmod -Rf 777 /appdata01/AEM\_Artifacts

**Rollback apps**

curl -u ${p:agent/aemUserName}:${p:agent/aemUserPassword} -F file=@"/appdata01/AEM\_Artifacts/ui.apps/target/${p:getOldVersion/appsOldVersion}" -F name="${p:getOldVersion/appsOldVersionName}" -F force=true -F install=true https://${p:agent/ip}:${p:agent/aemPort}/crx/packmgr/service.jsp -k

**CI and CD pipeline in Jenkins**

**Build Promotion to various Environments:**

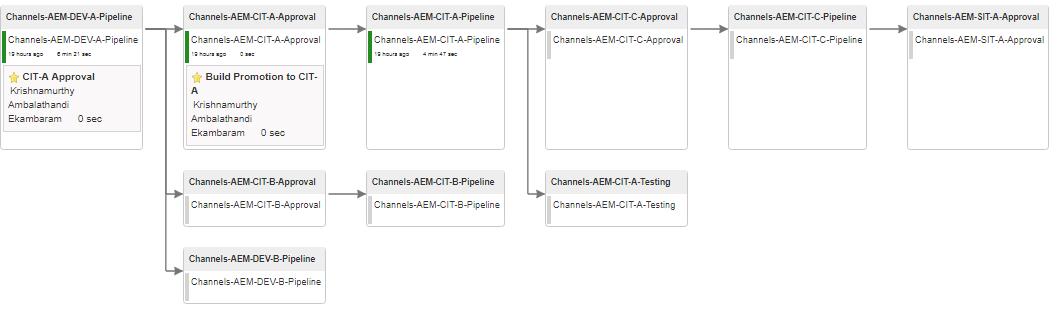
Build promotion to different environments based on approvals from different teams is configured in Jenkins.

Code that is fetched from release branches in BitBucket is built once, deployed in DEV-A and the same package is promoted to the higher environments.

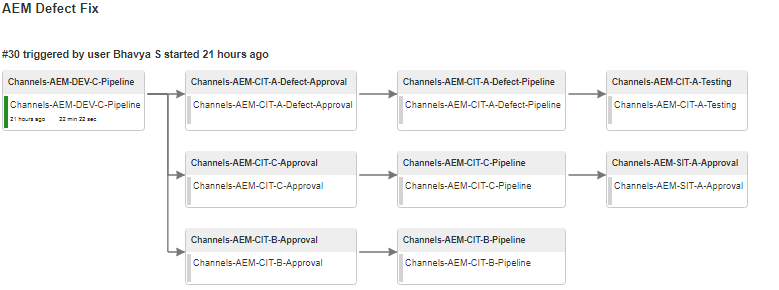
DEV-A 🡪 DEV-B,

CIT-A, 🡪 CIT-C 🡪 SIT-A

CIT-B



**Defect fix build promotion:**



DEV-C 🡪 CIT-A, CIT-B, CIT-C 🡪 SIT-A