|  |  |  |
| --- | --- | --- |
| **CI/CD OF CAFE COMPONENTS** | September 7  2016 | |
|  | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date**  **(dd/mm/yyyy)** | **Prepared By** | **Initial Reviewed by** | **Final Reviewed by** |
| **1.0** | **12/09/2016** | Daksha | Subbu |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Contents

[Overview and CI Architecture 3](#_Toc461458993)

[**1. Build Objective** 3](#_Toc461458994)

[**2. Publish Job Objective** 4](#_Toc461458995)

[**3. Jobs Template in Cloudset+** 6](#_Toc461458996)

[Service Components 6](#_Toc461458997)

[**1. Create a new job in cloudset** 6](#_Toc461458998)

[**2. Configure the job** 7](#_Toc461458999)

[**3. Build the job** 27](#_Toc461459000)

[**4. Check the job in SonarQube** 28](#_Toc461459001)

[Widget Components 29](#_Toc461459002)

[**1. Create a new job in Cloudset** 29](#_Toc461459003)

[**2. Configuration of job in Cloudset** 29](#_Toc461459004)

[**3. Build the job in Cloudset** 34](#_Toc461459005)

[**4. Check the SonarQube analysis** 35](#_Toc461459006)

[Database Components 36](#_Toc461459007)

[**1. Create a new job in Cloudset** 36](#_Toc461459008)

[**2. Configuration of job in Cloudset** 36](#_Toc461459009)

[**3. Build the job in Cloudset** 39](#_Toc461459010)

[Self Service Job 39](#_Toc461459011)

[**1. Create a new job in Cloudset** 39](#_Toc461459012)

[**2. Configuration of job in Cloudset** 39](#_Toc461459013)

[Admin Job 43](#_Toc461459014)

[**1. Create a new job in Cloudset** 43](#_Toc461459015)

[**2. Configuration of job in Cloudset** 43](#_Toc461459016)

[Template Job 44](#_Toc461459017)

[**1. Template For Service components With Test Cases** 44](#_Toc461459018)

[ **Create the template Component** 44](#_Toc461459019)

[ **Configure the Template component** 44](#_Toc461459020)

[**2. Template For Service components Without Test Cases** 59](#_Toc461459021)

[ **Create the template Component** 59](#_Toc461459022)

[ **Configure the Template component** 59](#_Toc461459023)

[**3. Template For Database components** 65](#_Toc461459024)

[ **Create the template Component** 65](#_Toc461459025)

[ **Configure the Template component** 65](#_Toc461459026)

[**4. Template For Widgets componentes without test cases** 70](#_Toc461459027)

[ **Create the template Component** 70](#_Toc461459028)

[ **Configure the Template component** 70](#_Toc461459029)

[**5. Template For Widgets Components with Test cases** 72](#_Toc461459030)

[ **Create the template Component** 72](#_Toc461459031)

[ **Configure the Template component** 72](#_Toc461459032)

[Overview and CD Architecture 75](#_Toc461459033)

[Component 76](#_Toc461459034)

[**1. Create Components** 76](#_Toc461459035)

[**2. Add Properties** 77](#_Toc461459036)

[**3. Component Process** 78](#_Toc461459037)

[**4. DB\_Deployment Process** 88](#_Toc461459038)

[Application 89](#_Toc461459039)

[**1. Create Application** 89](#_Toc461459040)

[**2. Create Environment** 90](#_Toc461459041)

[**3. Add Properties** 90](#_Toc461459042)

[**4. Map Component** 91](#_Toc461459043)

[**5. Application Process** 92](#_Toc461459044)

[Resource 96](#_Toc461459045)

[**1. Create Resource** 96](#_Toc461459046)

[**2. Add Resource to application** 97](#_Toc461459047)

# Overview and CI Architecture

CAFE is a completely DOT NET application and the way versions are controlled & Releases are made in CAFE is different from other application in BSD. So based on the understanding of [CAFE CI CD](https://wiki.sys.comcast.net/display/CCCC/CAFE+CI+CD) as per our AS IS document, the CI CD approach is recommended.

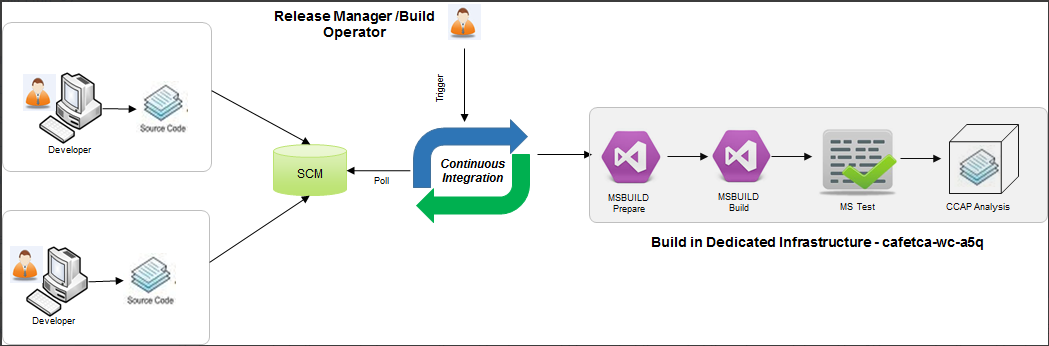
In CAFE the CI pipeline will be configured in CloudSet+ & at any given time the CI pipeline will be active for the Trunk & last 3 branches (T, T-1, T-2 & T-3), The typical CI scope would be from getting the latest code from TFVC, building, Unit Testing, Code quality analysis & packaging the application. All the Build Jobs for now will be executed from cafetca-wc-a5q (Agent).

For Trunk & Branches there will be a single Job per component / project in CloudSet+ to ensure the Build & publish objectives are met

In Trunk & branch the jobs will be created for all the components of CAFÉ.

## **1. Build Objective**

* **Trigger** - For now the CI jobs will be scheduled to run ever 15 / 30 min only if there are any code changes made with respect to the last build cycle.
* **SCM** - Each build job will be mapped to the TFVC folder of the respective project (component & associated folder) & on trigger the latest code from the respective TFVC will be downloaded in the agents folder.
* **Prepare** - MSBUILD plugin of CloudSet will be leveraged during the prepare step and the "prepare" target would be invoked, the assembly version required for the prepare step will be updated in the respective component's Assemblyinfo.cs file and the developers of CAFE will ensure the assembly version is rightly maintained for each component.
* **Build** - MSBUILD plugin of CloudSet will be leveraged during the build step to compile & build the application and the "rebuild" target would be invoked
* **UT** - MSTEST will be invoked using the CloudSet+ plugin / command line to trigger the MSTEST suite on the respective solution, there are some components where MSBUILD do not exist at component level. For those components MSTEST will not be applicable.
* **Coverage** - DOT COVER will be used for coverage instrumentation from CloudSet+
* **Code Analysis** - CCAP will be used for Code analysis, but the rules of code analysis will be as per Resharper profile. Along with C# other applicable languages like JS, XML & HTML etc. would also be analyzed for the component as applicable.
* **Post Build** - In the event of any failure in any step during the CI process notifications would be sent to the AD groups configured in CloudSet for the respective job. In the event of success the notification would be sent at the end of the job completion. The AD groups for the notification would be
* **RBAC** - By default all members of CAFE would be given read only access to all the CAFE jobs in Cloudset+, release managers would be given an additional option to Trigger jobs in CloudSet+ trigger permission would be available for component specific jobs, Common UT jobs, Central CAFE level CCAP analysis Jobs and Self Service Jobs



## **2. Publish Job Objective**

As mentioned in the CI Architecture section, the objective of the Publish Job is to handover the binaries to uDeploy’s code-station

* **Trigger** - This objective will also be configured a part of build job and will ensure binaries are published in UDeploy and the deployment for lowest environment is automatically triggered.

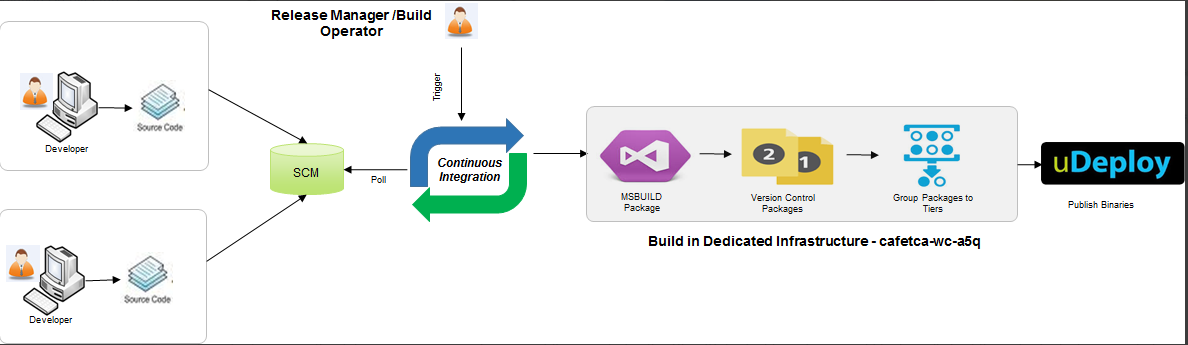
Typically every check in would result in an automated deployment in the Dev & CI environment.

* **Package** - MSBUILD plugin of CloudSet will be leveraged during the package step & package will be done at each component level. The current approach of packaging will be reused in CloudSet+ as well.
* **Version** **Control** – The current naming convention of CAFE will be respected [YY][MM].[R].{build} where The naming of YY.MM.R will be directly retrieved as given in the Assemblyinfo.cs, the build number will be appended based on the following logic <Assembly Version>.<Build\_Identifier> - Build Identifier will Start from 001 for trunk jobs<Assembly\_Version>.<200 + Build\_Identifier> - Build Identifier will Start from 201 for branch jobs Note: Build Identifier is not same as build number, it will be incremented for each successful package and the value will be reset to 001 in the trunk job, whenever a new branch is created.
* **Publish** - This is the final step in Publish job where using the CloudSet+ UDeploy plugin the final version of the binaries will be published to uDeploy’ s code station against the respective components. Along with publish certain properties like Sprint Version # , Component property path etc. will be updated for the respective versions

Note: During implementation since are going to use only LKG so the auto-deployment option will be mapped to LKG initially

* **CloudSet Properties** -

|  |  |  |
| --- | --- | --- |
|  | Property | Description |
| 1 | Build Identifier | This parameter will be available in all the CI jobs for CAFE. For all the jobs on the event of successful pass of quality gate this parameter value will be incremented by 1   * For Trunk Jobs the value of this parameter will start from 001 and will be reset to 001 on the event of branch creation self service   + For Branch jobs till parameters value will start from 201 (will be auto set as part of Self Service) |
| 2 | Property URL | For each component the URL for the property file will be constructed and will be published as a version property as part of UDeploy binary publish step |



## **3. Jobs Template in Cloudset+**

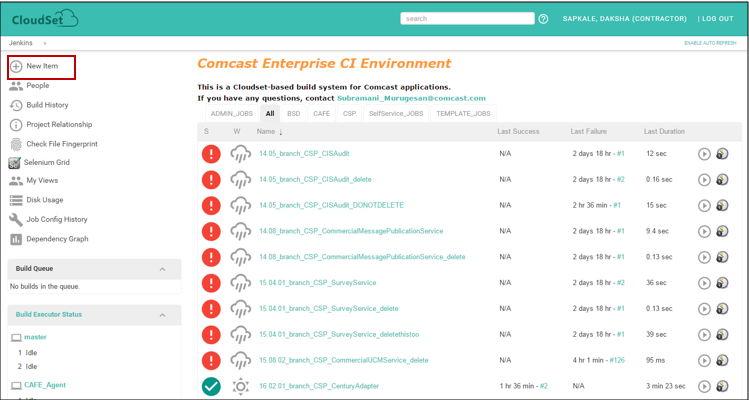
There will be two types of Job templates in CloudSet+ for CAFE,

* **Component Job with UT** - Checkout latest code from respective TFS roots, MS Build Prepare, MS Build Rebuild, MS Build Current way of packaging, MS Build native way of packaging, MS Test, DOT Cover, CCAP, CCAP Quality Gate with UT metrics, Publish to you UDeploy, Publish to current file server for binaries & Auto Deploy in first lower environment using UDeploy
* **Component Job without UT -**Checkout latest code from respective TFS roots, MS Build Prepare, MS Build Rebuild, MS Build Current way of packaging, MS Build native way of packaging, CCAP, CCAP Quality Gate without UT Metrics, Publish to you UDeploy, Publish to current file server for binaries & Auto Deploy in first lower environment using UDeploy

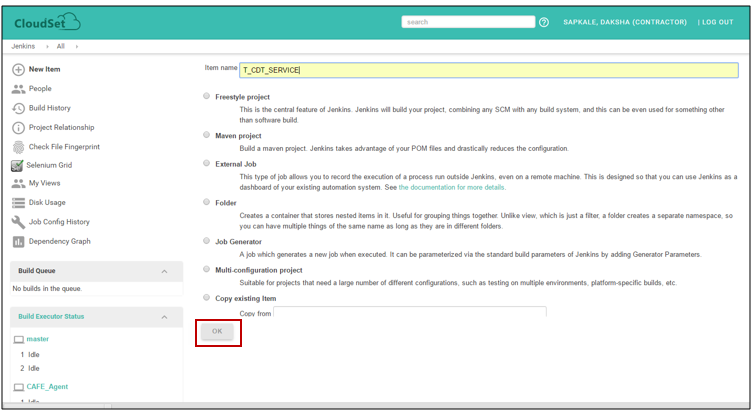
# Service Components

## **1. Create a new job in cloudset**

Step 1 :- Click on **New Item** to create new job.

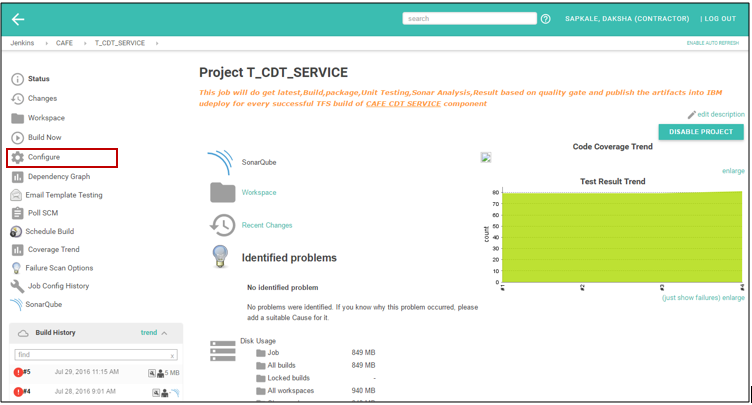


Step 2 :- Add Item name 🡪 click on **OK**



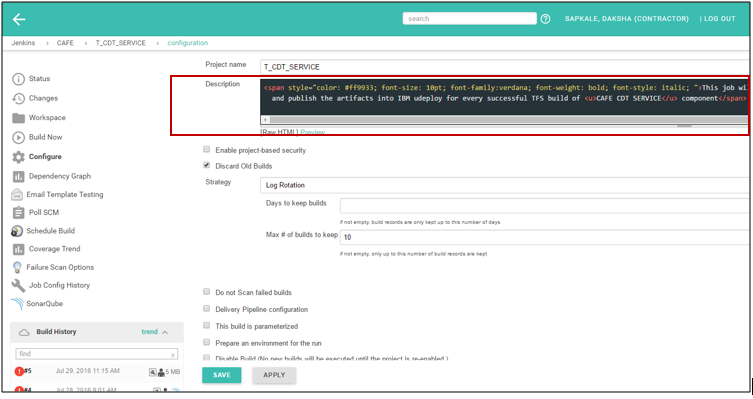
## **2. Configure the job**

To configure job 🡪 Click on **Configure**.



**Configurations**

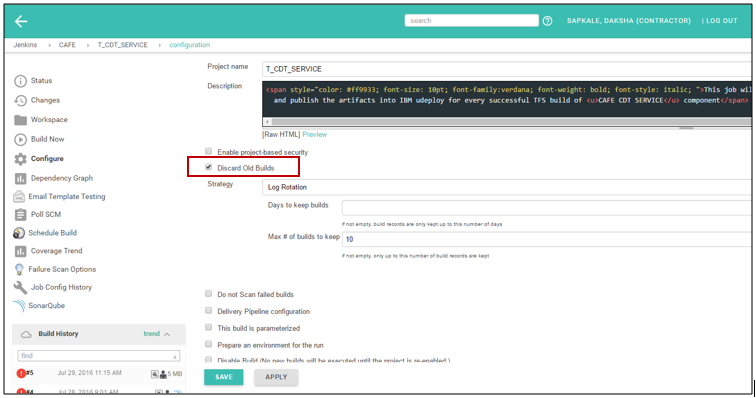
1. **Description**



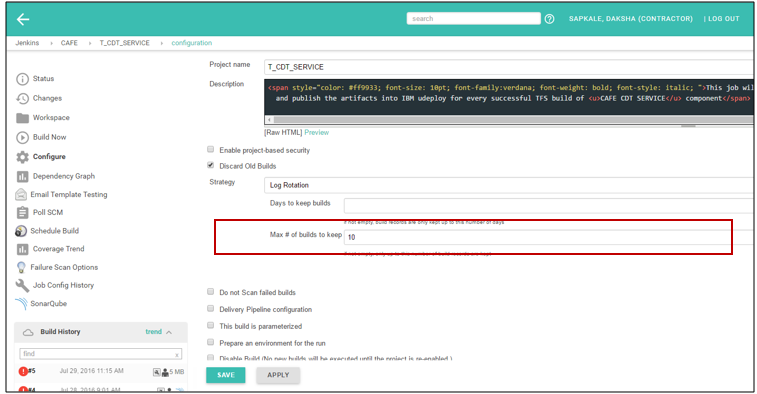
This is a HTML tag providing the job details to user. The detail Description can be as given below.

<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Result based on quality gate and publish the artifacts into IBM udeploy for every successful TFS build of <u>CAFE CDT SERVICE</u> component</span>

1. To discard old builds of this jobs select “**Discard old builds”** checkbox.



1. Set the value of **Max number of builds to keep**. Builds within 30 days are kept.



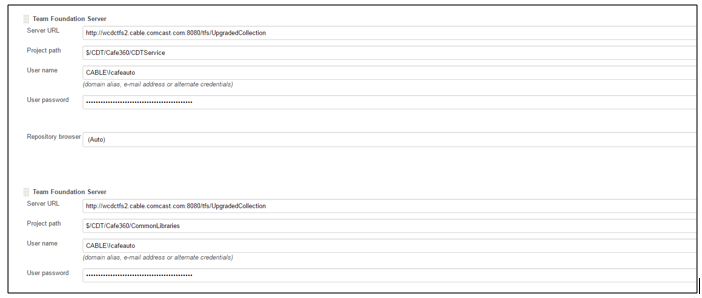
1. In JDK select the JDK you use in the project, here it is **JAVA\_CAFE\_Agent\_a5q**. In label expression, give **agent name ,** here it is **CAFE­**\_**agent**.



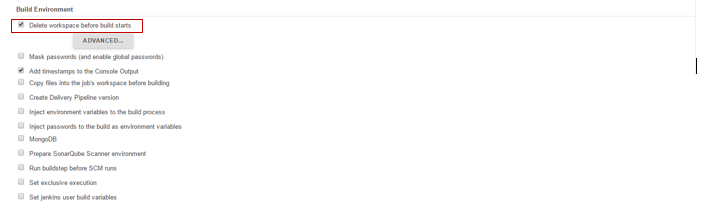
1. Select the **Multiple SCMs** In **Source Code Management**



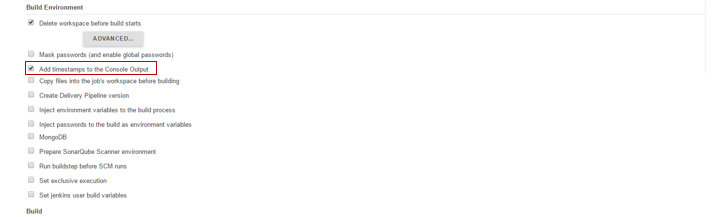
1. Select **Team Foundation Server**, and provide **Repository URL**, and credential of TFS.



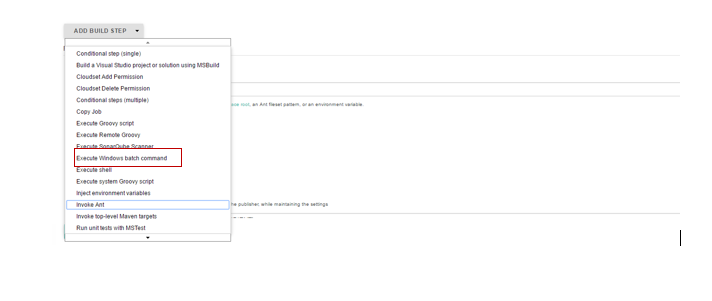
1. In **Build Environment,** select “**Delete workspace before build starts” check box.**



1. In **Build Environment**, tick **Add timestamps to Console Output**



1. Add the Build Step as **Execute Windows batch command** to add the batch command for execution.



To copy all library files to the workspace is the job of the following batch command.

echo D | xcopy "./CDTService\_1" "CDTService" /E

echo D | xcopy "./CommonLibraries\_1" "CommonLibraries" /E

RD /S /Q "./CDTService\_1"

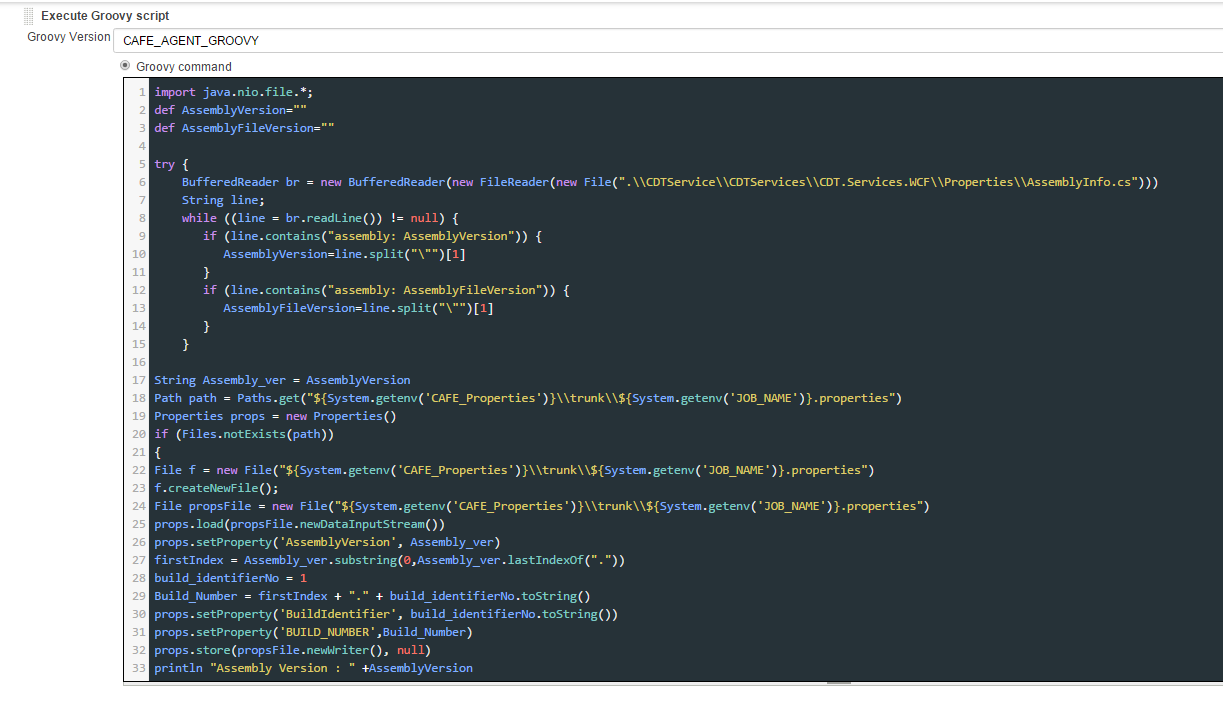
RD /S /Q "./CommonLibraries\_1"

1. **Execute Groovy Script**

To update the value of Assembly Version in Assemblyinfo.cs file this groovy script is used.

Go to 🡪 **Add Build Step** 🡪 **Select Execute groovy script**

**Groovy version** – Provide name of groovy version.



Steps for the script –

1. To get assembly version from Assemblyinfo.cs file.
2. If the component is trunk job then have to initialize the build identifier with 1.
3. Update the property file with build identifier, build number and Assembly version values.

**The groovy Script -:**

import java.nio.file.\*;

def AssemblyVersion=""

def AssemblyFileVersion=""

try {

//Fetching AssemblyVersion and AssemblyFileVersion from AssemblyInfo.cs

BufferedReader br = new BufferedReader(new FileReader(new File(".\\CDTService\\CDTServices\\CDT.Services.WCF\\Properties\\AssemblyInfo.cs")))

String line;

while ((line = br.readLine()) != null) {

if (line.contains("assembly: AssemblyVersion")) {

AssemblyVersion=line.split("\"")[1]

}

if (line.contains("assembly: AssemblyFileVersion")) {

AssemblyFileVersion=line.split("\"")[1]

}

println line

}

String Assembly\_ver = AssemblyVersion

Path path = Paths.get("${System.getenv('CAFE\_Properties')}\\trunk\\${System.getenv('JOB\_NAME')}.properties")

println path

Properties props = new Properties()

//This if statement will only be executed for the build when there is no properties file

if (Files.notExists(path))

{

//If properties file does not exists, creating properties file

File f = new File("${System.getenv('CAFE\_Properties')}\\trunk\\${System.getenv('JOB\_NAME')}.properties")

f.createNewFile();

File propsFile = new File("${System.getenv('CAFE\_Properties')}\\trunk\\${System.getenv('JOB\_NAME')}.properties")

props.load(propsFile.newDataInputStream())

//Set property AssemblyVersion in properties file from AssemblyVersion in AssemblyInfo.cs file

props.setProperty('AssemblyVersion', Assembly\_ver)

//firstIndex is fetch from AssemblyVersion(YY.MM.R)

firstIndex = Assembly\_ver.substring(0,Assembly\_ver.lastIndexOf("."))

//Initialize build\_identifierNo to 1(trunk job)

build\_identifierNo = 1

//Build number is constructed from AssemblyVersion and build\_identifierNo

Build\_Number = firstIndex + "." + build\_identifierNo.toString()

props.setProperty('BuildIdentifier', build\_identifierNo.toString())

props.setProperty('BUILD\_NUMBER',Build\_Number)

props.store(propsFile.newWriter(), null)

println "Assembly Version : " +AssemblyVersion

println "Build Number : " +Build\_Number

}

//If properties file exists, setting property AssemblyVersion in properties file

else

{

File propsFile = new File("${System.getenv('CAFE\_Properties')}\\trunk\\${System.getenv('JOB\_NAME')}.properties")

props.load(propsFile.newDataInputStream())

props.setProperty('AssemblyVersion', Assembly\_ver)

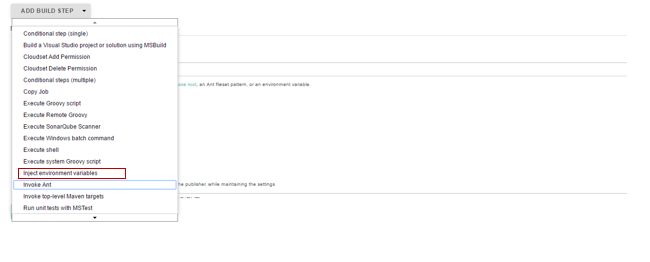
props.store(propsFile.newWriter(), null)

println "Assembly Version : " +Assembly\_ver

}

} catch (Exception e) {e.printStackTrace()}

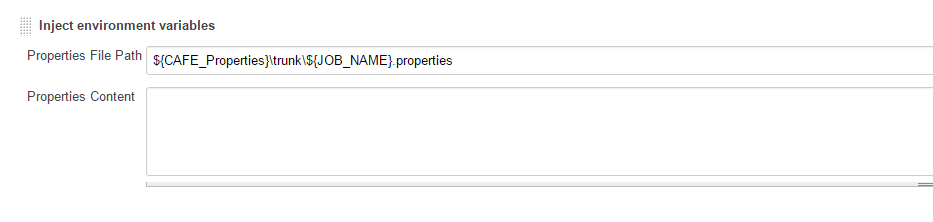
1. Go to, **Add Build Step** 🡪 **Inject environment variables.**



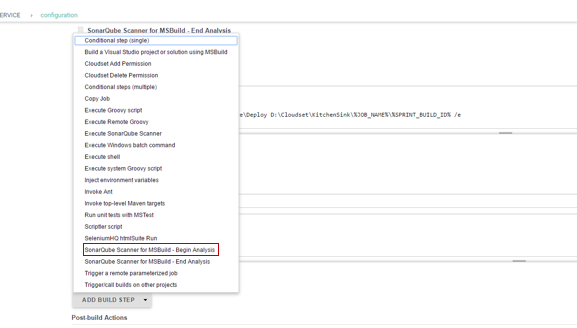
1. Provide properties File path, here the file path is constructed as below –

**${CAFÉ\_Properties}\trunk\${JOB\_NAME}.properties**

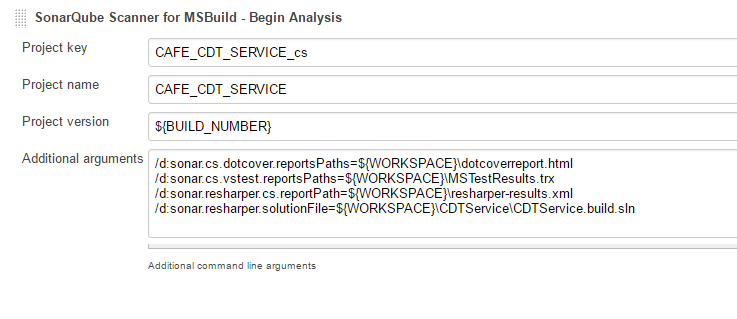
1. In workspace there is folder structure, café\_properties 🡪 trunk 🡪 {JOB\_NAME}.properties



1. **Add Build Step 🡪 SonarQube Scanner for MSBuild-Begin Analysis** to invoke sonarqube.



1. Provide **Project key, Project name, Project version** – SonarQube properties for project.



1. Provide the **Additional arguments –**
2. Dotcover report path
3. Vstest report path
4. Resharper report path
5. Resharper solution file

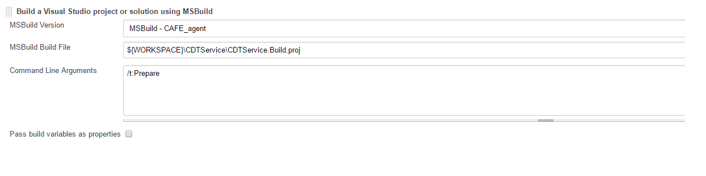
d:sonar.cs.dotcover.reportsPaths=${WORKSPACE}\dotcoverreport.html

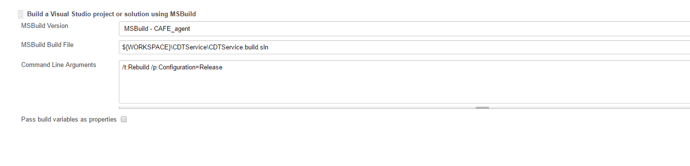
/d:sonar.cs.vstest.reportsPaths=${WORKSPACE}\MSTestResults.trx

/d:sonar.resharper.cs.reportPath=${WORKSPACE}\resharper-results.xml

/d:sonar.resharper.solutionFile=${WORKSPACE}\CDTService\CDTService.build.sln

1. **Add Build step** 🡪 **Build a Visual Studio project or solution using MSBuild.**
2. Provide the build file of job in **MSBuild Build File.**
3. Provide target in command line argument field.





1. **Execute Batch Command** –
2. Add the Build Step as **Execute Windows batch command** to add the batch command for MSTest. In this batch command Provide the test files. Here for this component two test assemblies are there so provide two test container.

"C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\MSTest.exe" /testcontainer:%WORKSPACE%\CDTService\CDTWCFService-CSTest\bin\Release\CDTWCFService-CSTest.dll /testcontainer:%WORKSPACE%\CDTService\CommercialSvc\CommercialSvc.DataAccess-CSTest\bin\Release\CommercialSvc.DataAccess-CSTest.dll /resultsfile: MSTestResults.trx

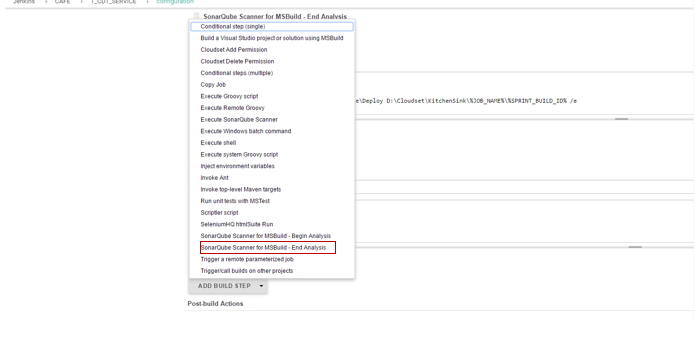
1. Add the Build Step as **Execute Windows batch command** to add the batch command for dotcover. In this batch command provide the configuration for Dotcover.

"C:\BuildAgent4\tools\dotCover\dotCover.exe" analyse /ReportType=HTML /Output="%WORKSPACE%\dotcoverreport.html" "/TargetExecutable=C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\MSTest.exe" /TargetWorkingDir=. "/TargetArguments=/testcontainer:%WORKSPACE%\CDTService\CDTWCFService-CSTest\bin\Release\CDTWCFService-CSTest.dll /testcontainer:%WORKSPACE%\CDTService\CommercialSvc\CommercialSvc.DataAccess-CSTest\bin\Release\CommercialSvc.DataAccess-CSTest.dll"

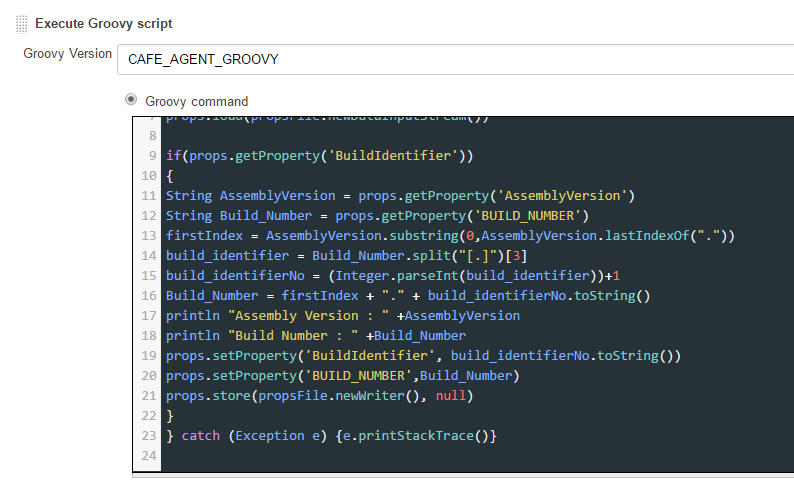
1. Add the Build Step as **Execute Windows batch command** to add the batch command for Resharper configuration. In this batch command Provide the Resharper file path.

"D:\cs\tools\JetBrains.ReSharper.CommandLineTools\inspectcode.exe" /output="resharper-results.xml" "%WORKSPACE%\CDTService\CDTService.build.sln"

1. **Add Build Step 🡪 SonarQube Scanner for MSBuild-End Analysis**



1. **Groovy Script**



Increment Build Number based on build success or failure -

1. The groovy script will only be executed when all above steps have been build successfully.
2. In this step, we are fetching the Assembly Version and Build Number from properties file.
3. The Build Identifier number keeps track of last successful build. It is assign with the last Index value we get after splitting the Build  Number.
4. This value is incremented for each successful build. The Build Number is updated accordingly.
5. The updated values for Build Identifier No and Build Number are written back to property file.

**Groovy script**

import java.nio.file.\*;

Properties props = new Properties()

try

{

//Loading properties file

File propsFile = new File("${System.getenv('CAFE\_Properties')}\\trunk\\${System.getenv('JOB\_NAME')}.properties")

props.load(propsFile.newDataInputStream())

if(props.getProperty('BuildIdentifier'))

{

String AssemblyVersion = props.getProperty('AssemblyVersion')

String Build\_Number = props.getProperty('BUILD\_NUMBER')

firstIndex = AssemblyVersion.substring(0,AssemblyVersion.lastIndexOf("."))

//Fetching the build\_identifier from Build\_Number(<AssemblyVersion>.<Build\_Identifier>)

build\_identifier = Build\_Number.split("[.]")[3]

//build\_identifierNo will be incremented for each successful package

build\_identifierNo = (Integer.parseInt(build\_identifier))+1

//Updating Build\_Number based on AssemblyVersion and build\_identifierNo

Build\_Number = firstIndex + "." + build\_identifierNo.toString()

println "Assembly Version : " +AssemblyVersion

println "Build Number : " +Build\_Number

//Setting updated values of BuildIdentifier and BUILD\_NUMBER to properties file

props.setProperty('BuildIdentifier', build\_identifierNo.toString())

props.setProperty('BUILD\_NUMBER',Build\_Number)

props.store(propsFile.newWriter(), null)

}

} catch (Exception e) {e.printStackTrace()}

1. Execute batch command –
2. Add the Build Step as **Execute Windows batch command**.

Go to 🡪 **Add build Step** 🡪 Select **Execute Windows batch command**

This batch command works to store the files to one folder.

cd D:\Cloudset\KitchenSink\

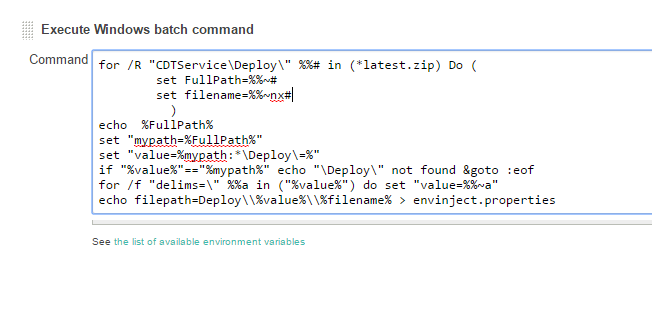
mkdir %JOB\_NAME%\%SPRINT\_BUILD\_ID%

echo D | xcopy %WORKSPACE%\CDTService\Deploy D:\Cloudset\KitchenSink\%JOB\_NAME%\%SPRINT\_BUILD\_ID% /e

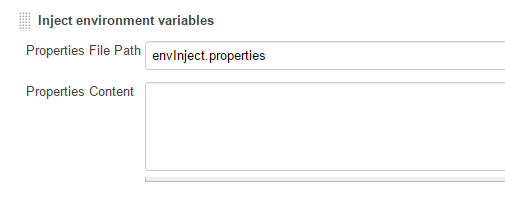
1. Add the Build Step as **Execute Windows batch command**.

Go to 🡪 **Add build Step**🡪 Select **Execute Windows batch command**

This batch script is written to get the file path which is passed to Udeploy as a property “**Package path**”.

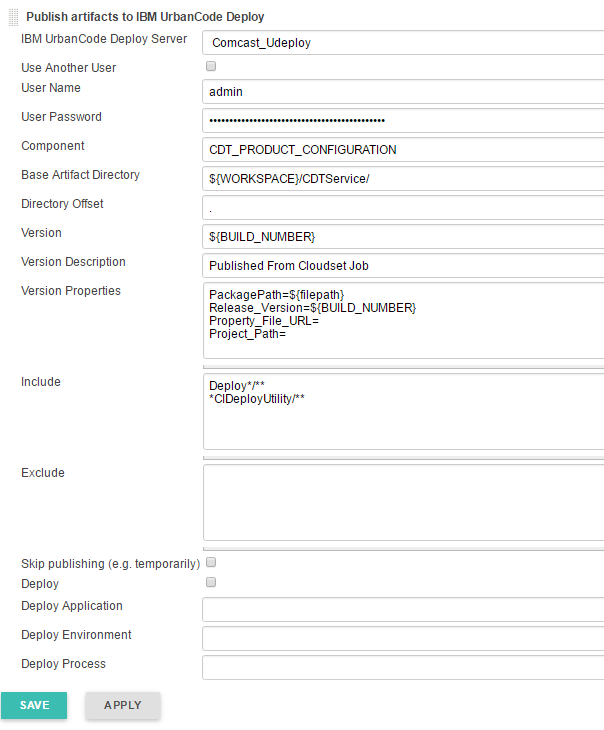


1. Inject the file path variable to envinject.properties file.



1. Publish artifact to Udeploy.

Go to 🡪 **Add post build Action** 🡪**Publish artifact to IBM UrbanCode Deploy**



**Configuration** -

|  |  |
| --- | --- |
|  | 1. **Component** – Provide Job Name to Component. 2. **Version** – Provide the Build\_Number 3. **Package path** – The Package path is file path which is constructed with batch script and injected to environment properties. 4. **Realease version** –Provide Build number 5. **Include** – Need to include all files under deploy older as well as deploy utility folder. |
|  |  |

1. Publish report –

Go to 🡪 **Add Post Build action** 🡪 **Publish MSTest test result report**

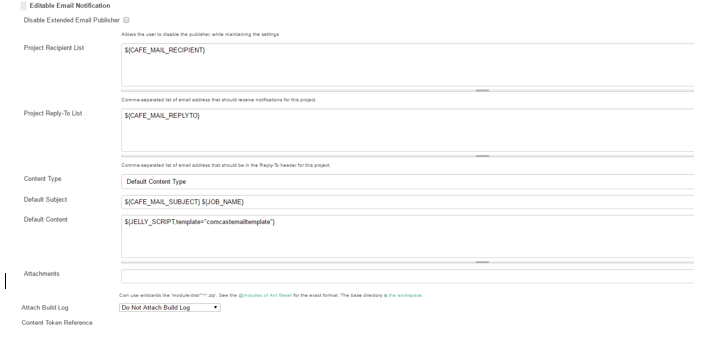
Provide the test file name.



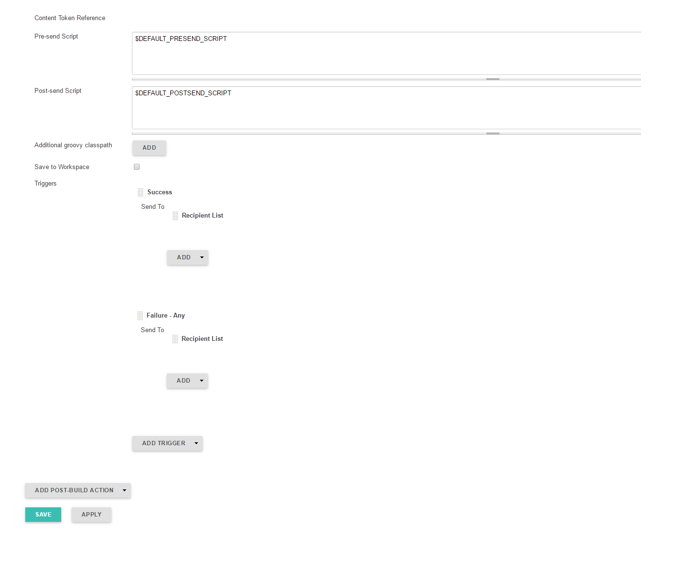
1. Email Notification –

Go to 🡪 **Add Post Build action** 🡪 **Editable Email notification**

Provide necessary details.



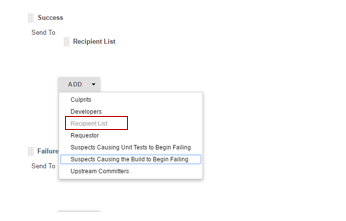
1. Click on **Advanced setting** tab.



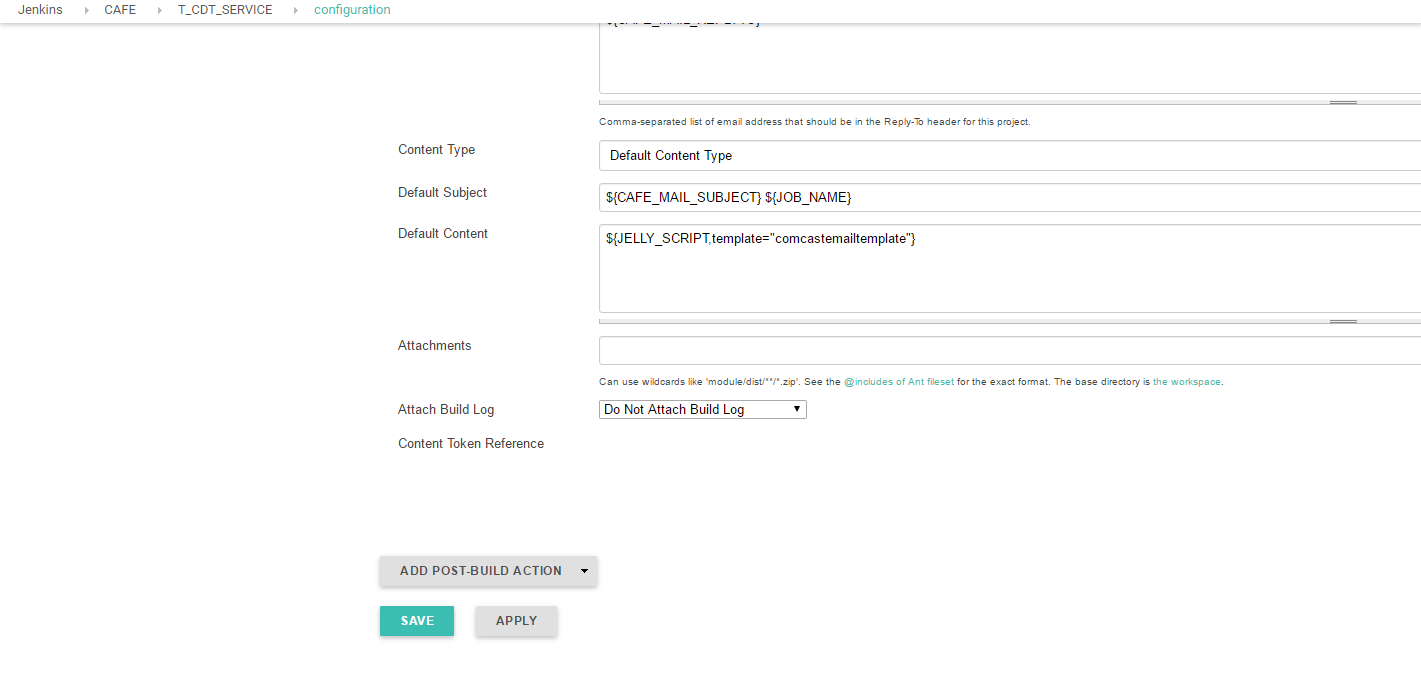
1. Click on **ADD TRIGER** tab **🡪** Add trigger of **Success.**



1. Click on **ADD** tab🡪Add **Recipient List**



1. Click on Advanced setting tab under trigger and provide the necessary details.

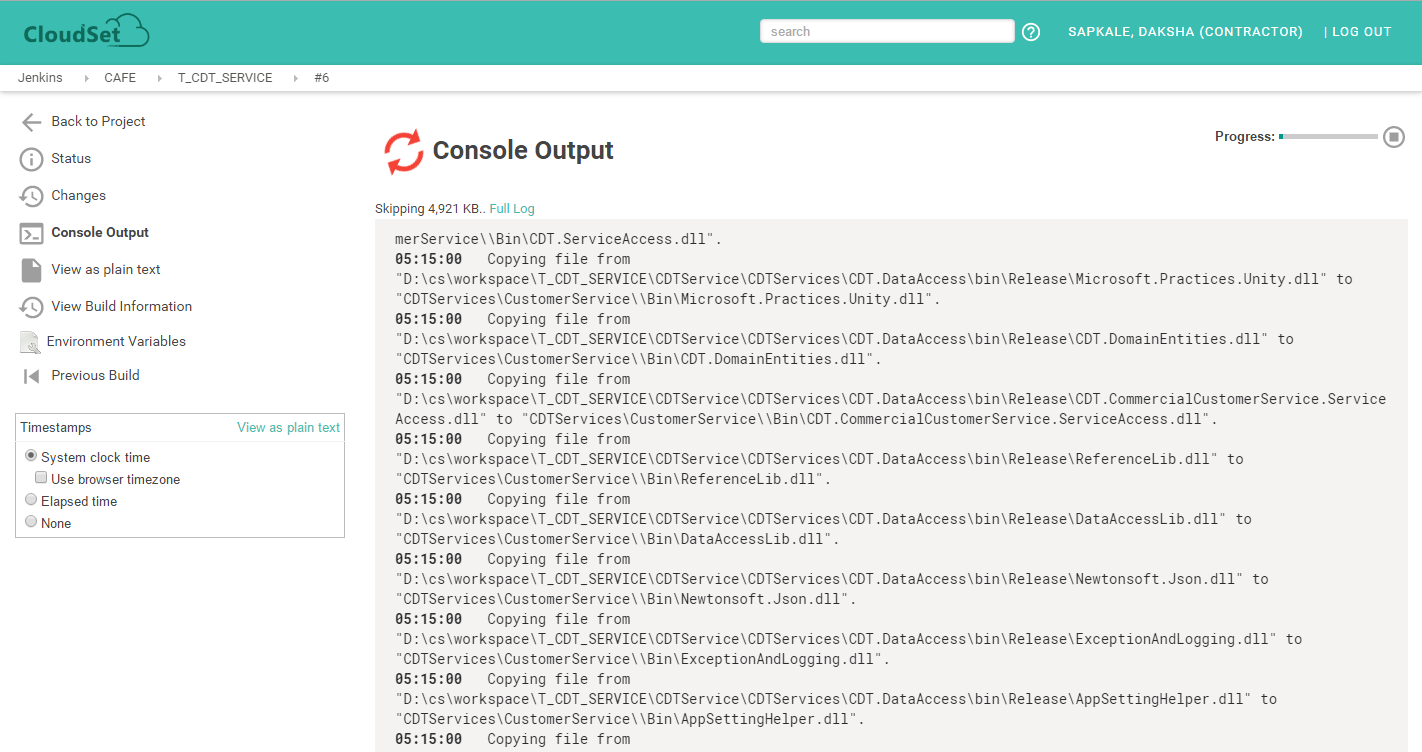


1. Click on **APPLY** 🡪 **SAVE.**

## **3. Build the job**

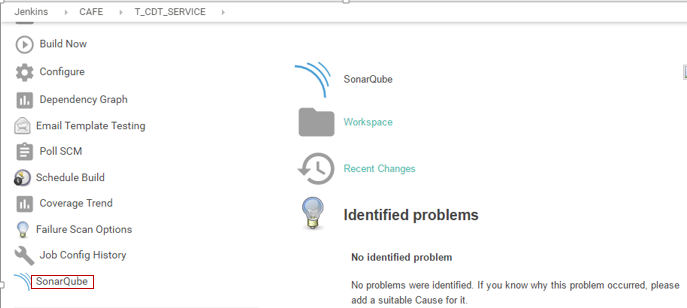
Click on **Build Now** to build the job.

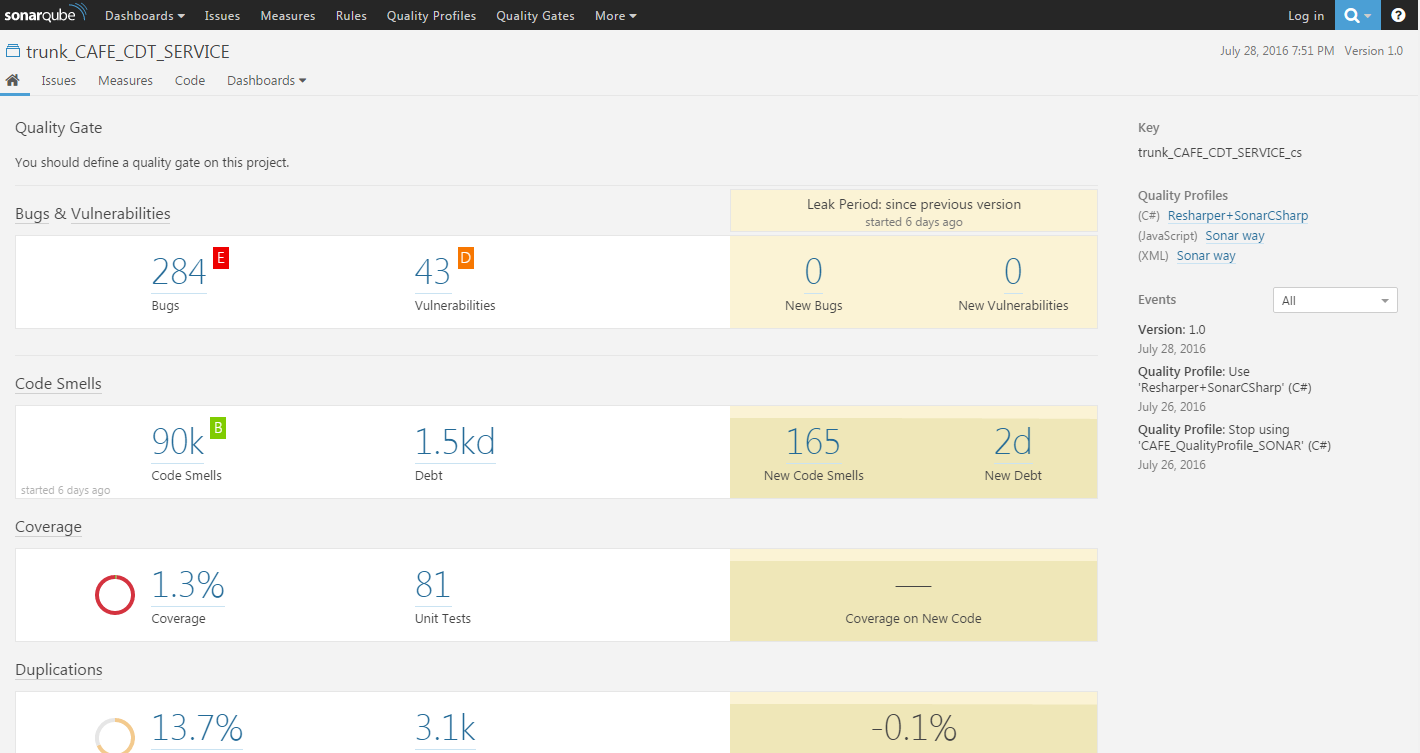




## **4. Check the job in SonarQube**

To check sonarqube analysis of job click on **SonarQube** tab.





# Widget Components

## **1. Create a new job in Cloudset**

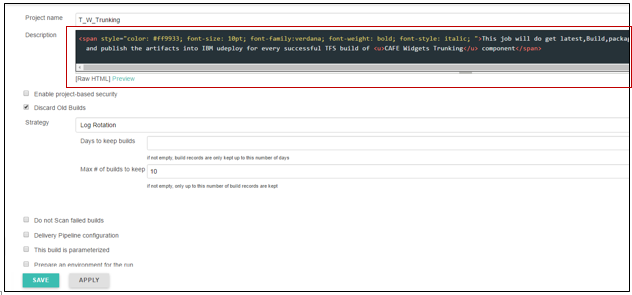
Go to 🡪 **Click here for Create a new job steps**

## **2. Configuration of job in Cloudset**

To configure job 🡪 Click on **Configure**.

**Configurations**

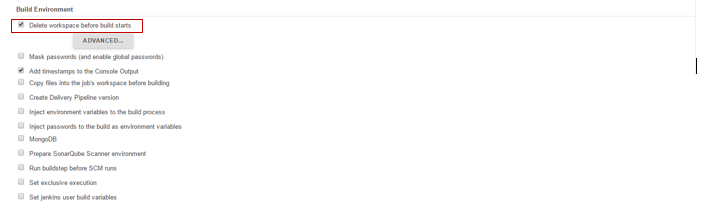
1. **Description**



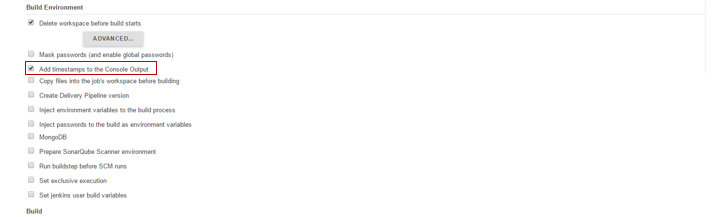
<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Result based on quality gate

and publish the artifacts into IBM udeploy for every successful TFS build of <u>CAFE Widgets Trunking</u> component</span>

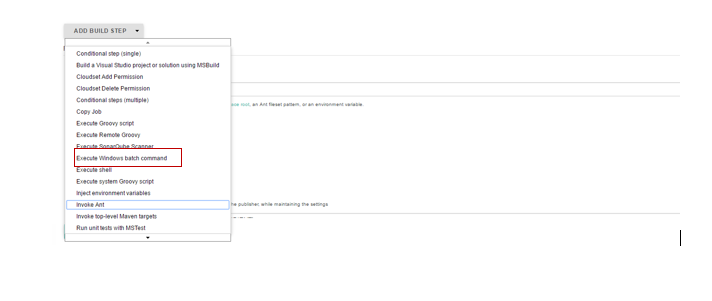
1. In **Build Environment**, check **Delete workspace before build starts.**



1. In **Build Environment**, check **Add timestamps to Console Output**



1. **Add Build Step** 🡪 **Execute Windows batch command** to add the batch command for execution.



1. **Batch command** -

echo D | xcopy "./CDTService\_1" "CDTService" /E

echo D | xcopy "./CommonLibraries\_1" "CommonLibraries" /E

RD /S /Q "./CDTService\_1"

RD /S /Q "./CommonLibraries\_1"

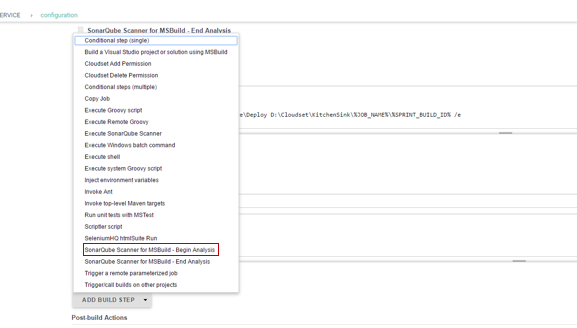
This command is used to copy all library files from particular repository location to the workspace of job.

1. **Execute Groovy Script**

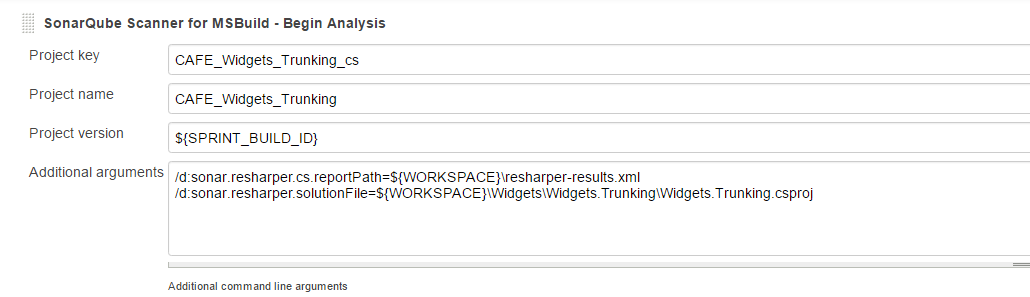
Go to 🡪 **Click here to follow Execute Groovy Script Steps**

1. Invoke SonarQube Scanner –

Go to 🡪 Add Build Step 🡪 **SonarQube Scanner for MSBuild-Begin Analysis**



1. Provide **Project key, Project name, Project version** – SonarQube properties for project.



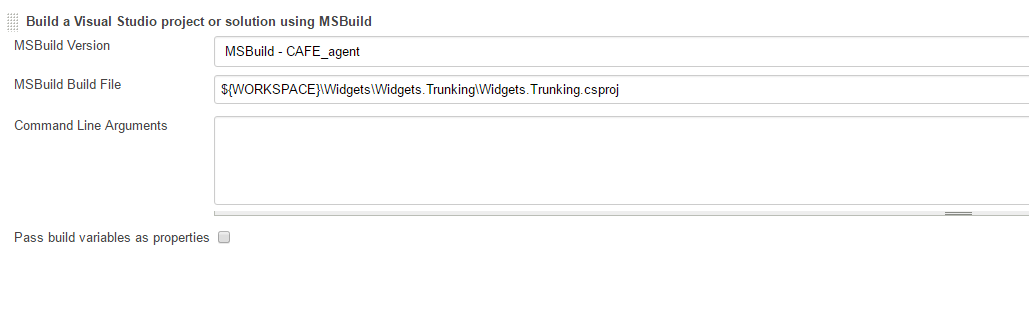
1. Provide the **Additional arguments –**

|  |
| --- |
| /d:sonar.resharper.cs.reportPath=${WORKSPACE}\resharper-results.xml  /d:sonar.resharper.solutionFile=${WORKSPACE}\Widgets\Widgets.Trunking\Widgets.Trunking.csproj |

1. **Build a Visual Studio project or solution using MSBuild**

Go to 🡪 Add the Build step 🡪 **Build a Visual Studio project or solution using MSBuild.**

Provide the build file of job in **MSBuild Build File.**

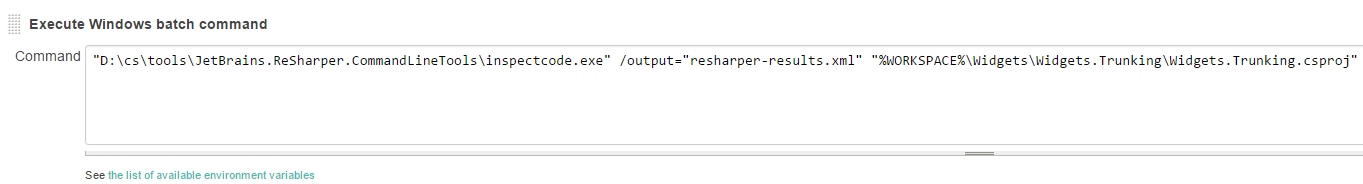


1. **Execute Windows batch command**

**Go to 🡪 Add the Build Step** 🡪 **Execute Windows batch command**

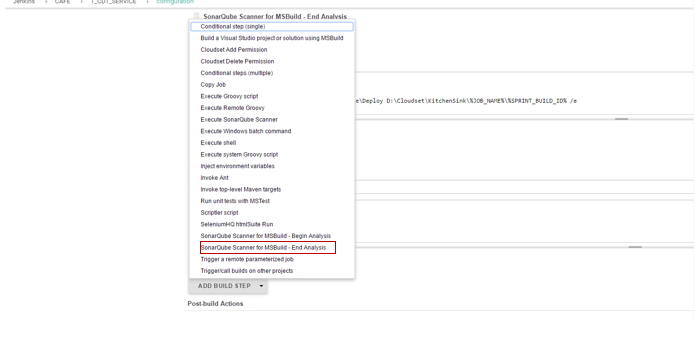
This batch command is used to generate the Resharper-result file.

The .csproj file need to pass to the command.



|  |
| --- |
| "D:\cs\tools\JetBrains.ReSharper.CommandLineTools\inspectcode.exe" /output="resharper-results.xml" "%WORKSPACE%\Widgets\Widgets.Trunking\Widgets.Trunking.csproj" |

1. Add Build Step **SonarQube Scanner for MSBuild-End Analysis**



1. **Execute Groovy Script**

Go to 🡪 **Click here to follow the Groovy Script steps**

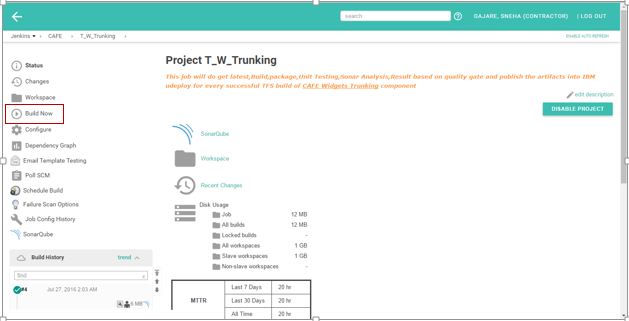
1. **Email notification**

Go to 🡪 **Click here for Email Notification Steps**

1. Click on **APPLY** 🡪 **SAVE.**

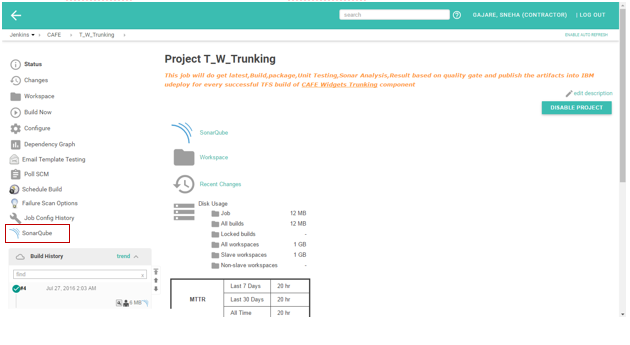
## **3. Build the job in Cloudset**

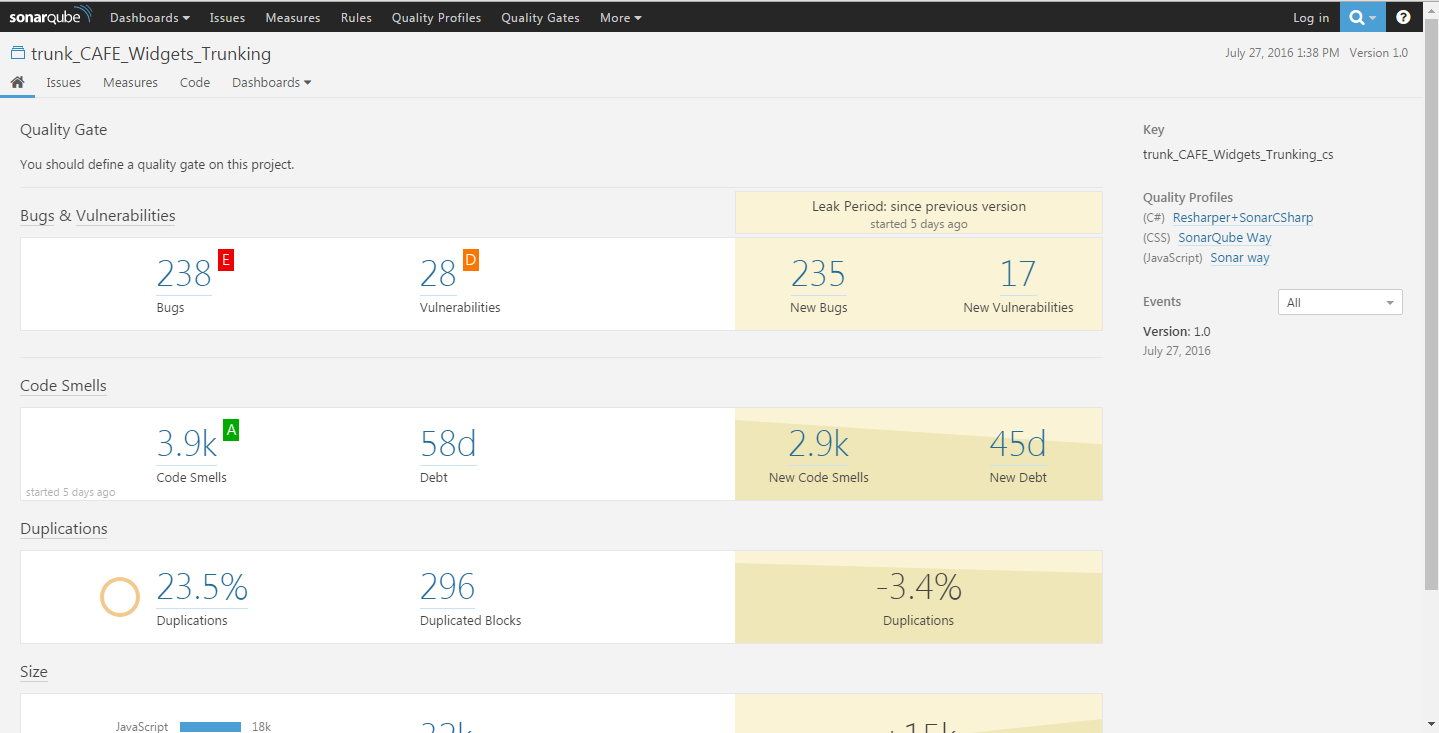
Click on **Build Now** to build the job.



## **4. Check the SonarQube analysis**

To check sonarqube analysis of job click on **SonarQube** tab.





# Database Components

## **1. Create a new job in Cloudset**

Go to 🡪 **Click here for Create a new job steps**

## **2. Configuration of job in Cloudset**

To configure job 🡪 Click on **Configure**.

**Configurations**

1. **Description**

<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Result based on quality gate

and publish the artifacts into IBM udeploy for every successful TFS build of <u>CAFE CDT DATABASE</u> component</span>

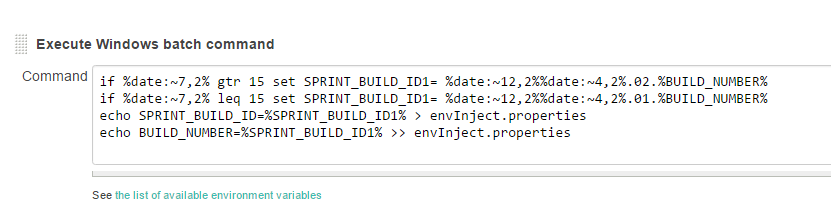


1. **Click here and follow the Steps from 2 to step 4**
2. Execute the batch command –
3. Go to 🡪**Add Build Step** 🡪**Execute Windows batch command**

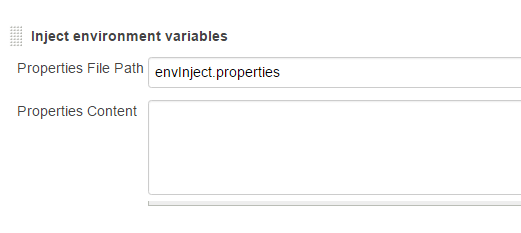
This command is used to copy all files.



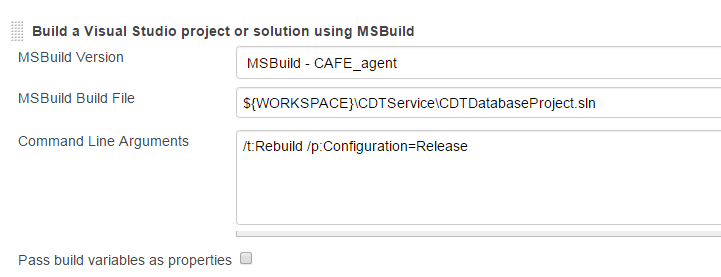
1. Go to 🡪**Add Build Step** 🡪**Execute Windows batch command**



1. **Inject Environment Variable** -

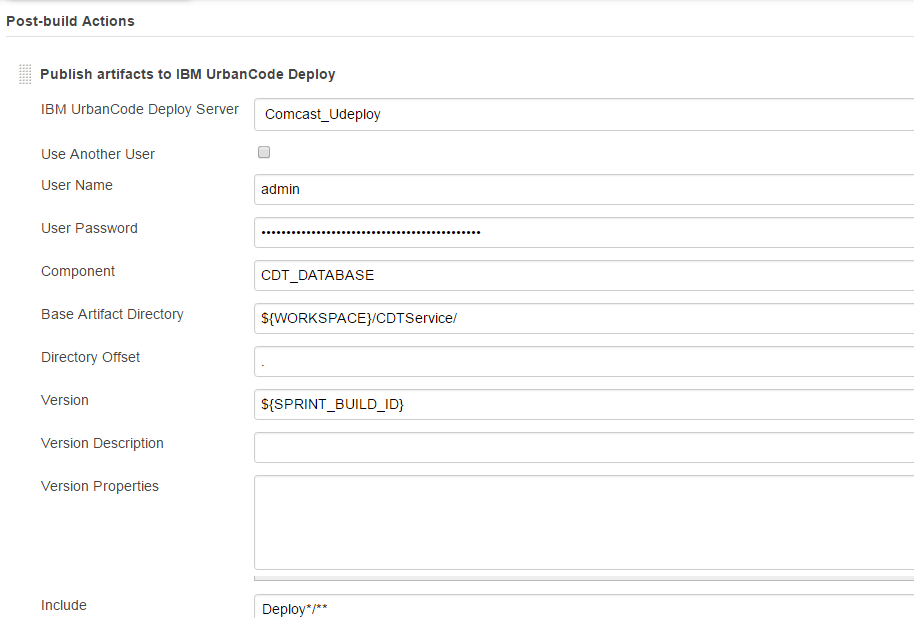


1. **Build a visual studio Project** -



1. **Publish artifact to Udeploy** –

Go to🡪 **Add Post Build steps** 🡪**Publish artifact to IBM UrbanCode Deploy**



1. **Email notification**

Go to 🡪 **Click here for Email Notification Steps**

1. Click on **APPLY** 🡪 **SAVE.**

## **3. Build the job in Cloudset**

Click on **Build Now** to build the job.

# Self Service Job

Self-service job is created to generate the branches of component.

When the service job will trigger it will call 🡪 **admin job** and admin job will call 🡪**template job**

After that the branch will create. There are different template job for service component with test cases, without test cases, widget with test cases, widget without test cases and database components, similarly different admin jobs for different components.

## **1. Create a new job in Cloudset**

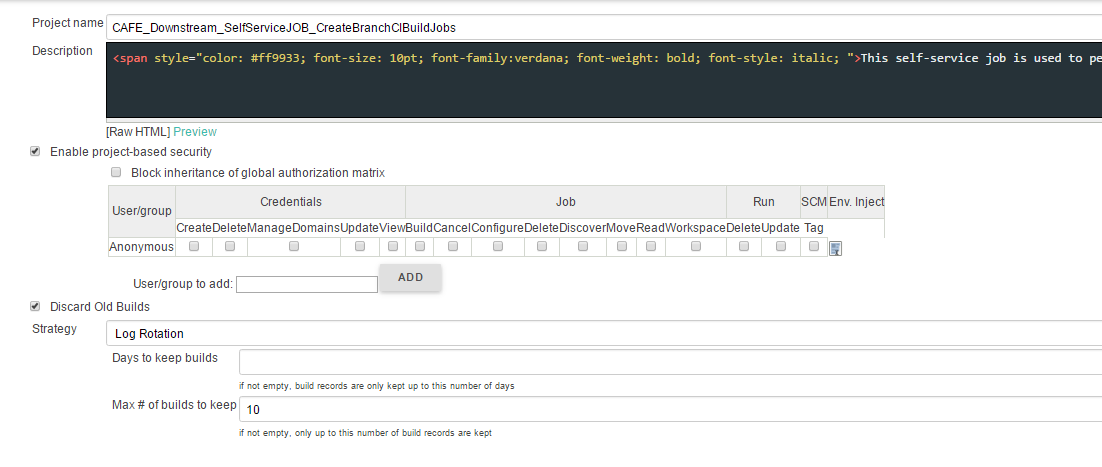
Go to 🡪 **Click here for Create a new job steps**

## **2. Configuration of job in Cloudset**

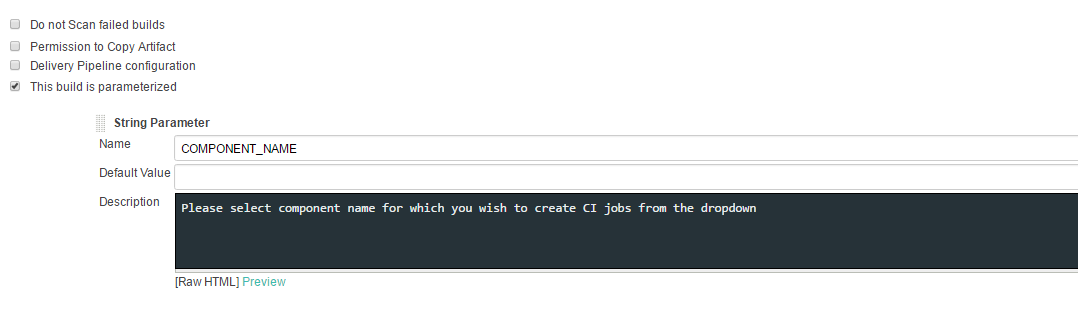
To configure job 🡪 Click on **Configure**.

1. Description –

Fill the description.

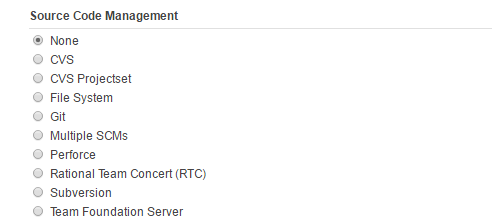


1. Check **Discard old build** and provide the value to **Max # of build to keep**.
2. Check **This build is parameterized**.
3. Provide some input from self-service screen. This input are taken as String parameter in configuration part.
4. **Click here to check the parameter list**



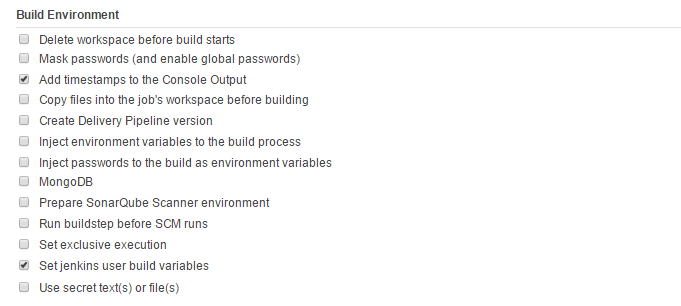
1. **Source code management** –

Select **None** for **Source Code Management**, when no file needs to be fetched from SCM.

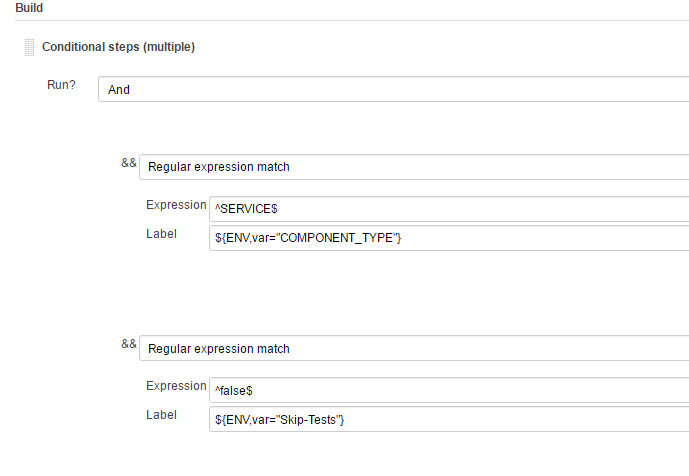


1. **Build Environment** –

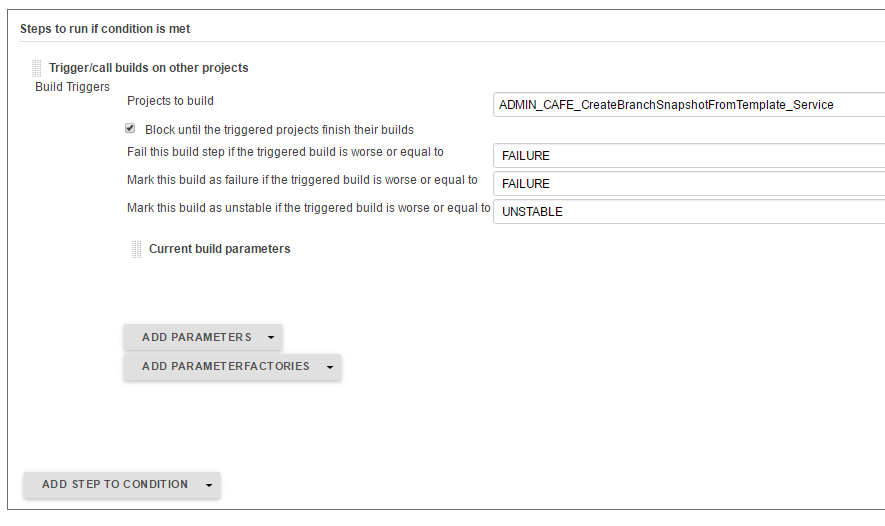
Check two fields, **Add timestamp to console output and set Jenkins user build variables**.



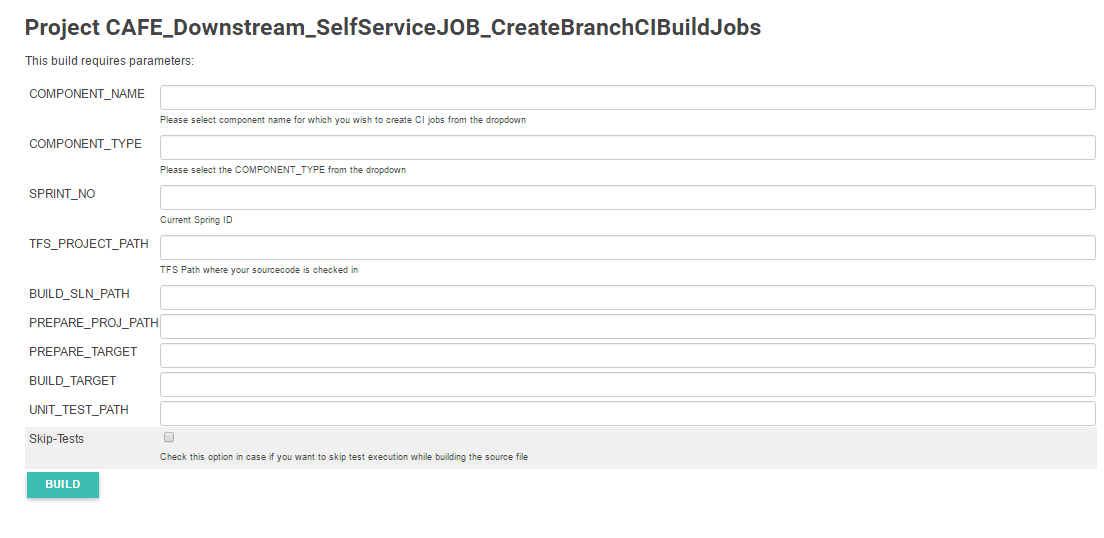
1. **Conditional Steps** –
2. Go to 🡪 **Add build Step** 🡪 **Conditional Steps (Multiple)**



1. Match the job name with regular expression.
2. Expression – Provide the regular expression to match the field name.
3. Label - Check for component types and skip test, this will define the type of job.
4. Trigger other job
5. Go to 🡪 **Add build Step** 🡪 **Trigger build on other project**.



1. After the conditional step call admin job.
2. According to component type provide the Admin job in Project to build field.
3. Click on **APPLY** 🡪 **SAVE.**
4. Go to 🡪 **Build with parameter**



The Above screen is self-service screen Provide following parameters.

1. Component name – Provide the component name for which branch will get create.
2. Component type – Provide type of components as service, widget or database component.
3. Sprint\_No – Provide the sprint no with which branch will get create.
4. TFS project path – Provide the Project path in TFS.
5. Build\_Solution\_Path - Provide the Build Solution path
6. Prepared project path - Provide the Prepared project path
7. Prepare target – Provide the prepare target specified for component.
8. Build target - Provide the Build target specified for component.
9. Unit test path - Provide the Unit test path specified for component.
10. Skip Test – If test cases are not present for the component tick this.
11. Click on🡪 **Build**

This will create the branches for job**.**

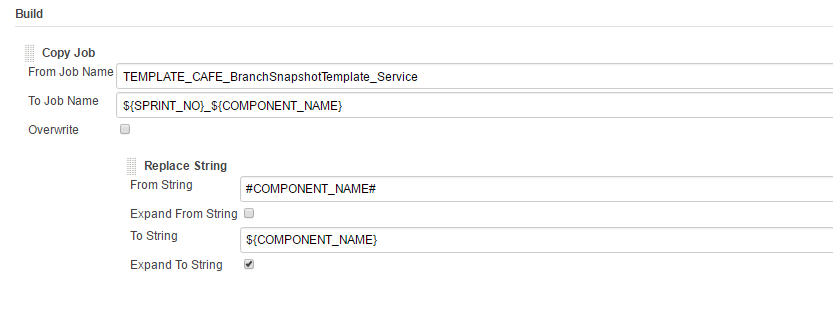
# Admin Job

## **1. Create a new job in Cloudset**

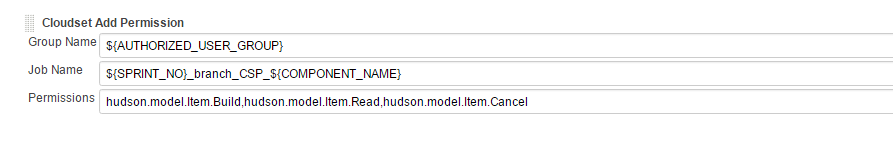
Go to 🡪 **Click here for Create a new job steps**

## **2. Configuration of job in Cloudset**

1. **Click here and follow the steps 1 to step 5**
2. **Build**
3. In admin job, copy the template job.
4. Go to 🡪 **Add build step** 🡪 **Copy job**



1. From job name – Provide template job name
2. To job name – Provide job name to be created according to **${Sprint\_No}\_${Component Name}**
3. Replace string – Provide string to be replace from template job
4. Cloudset Add Permission
5. Go to 🡪 **Add build step** 🡪 **Cloudset add permission**
6. Group name – Provide group name
7. Job name – Provide job name
8. Permission – Provide permission



1. Click on **APPLY** 🡪 **SAVE.**

# Template Job

## **1. Template For Service components With Test Cases**

### **Create the template Component**

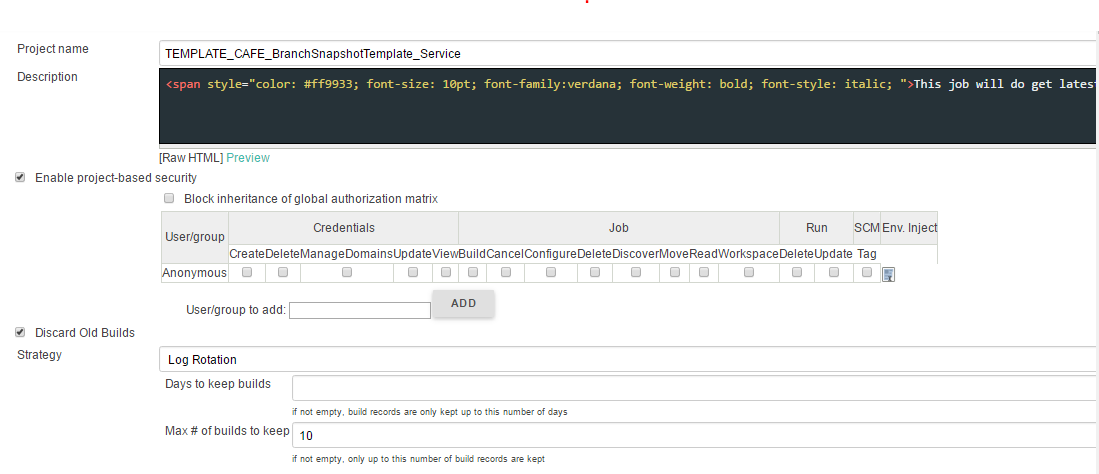
Go to **New item** 🡪 Give **Item Name** 🡪 **OK**

### **Configure the Template component**

Click on Component Name 🡪 Click on **Configure**

**Configuration**

1. Description – Provide the description for template job
2. Check **discard old build** and provide the number of build need to keep in **Max # of build to keep** part

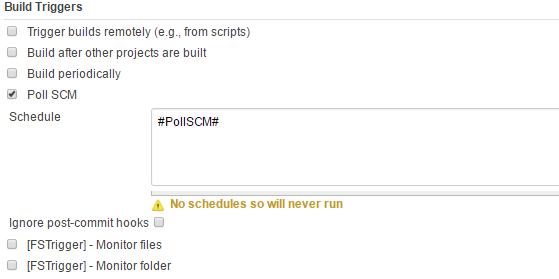


<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Quality gate based on code metrics and publish the artifacts into IBM udeploy every 30 mins in case of any code changes in <u>#COMPONENT\_NAME#</u> component of CAFE application</span>

1. JDK –Here need to provide agent which is **JAVA\_CAFE\_Agent\_a5q.**
2. Select the **Multiple SCMs** In **Source Code Management**, select **Team Foundation Server**, and provide **Repository URL**, and credential of TFS.

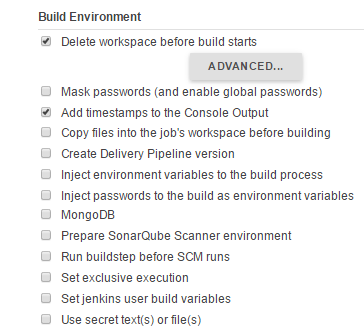


1. Go to 🡪**Build Triggers** 🡪Tick the **Poll SCM checkbox**



The value **#PollSCM#** is taken as input from self-service job.

1. In **Build Environment**, tick **Delete workspace before build starts**
2. In **Build Environment**, tick **Add timestamps to Console Output**



1. **Execute Windows batch command**

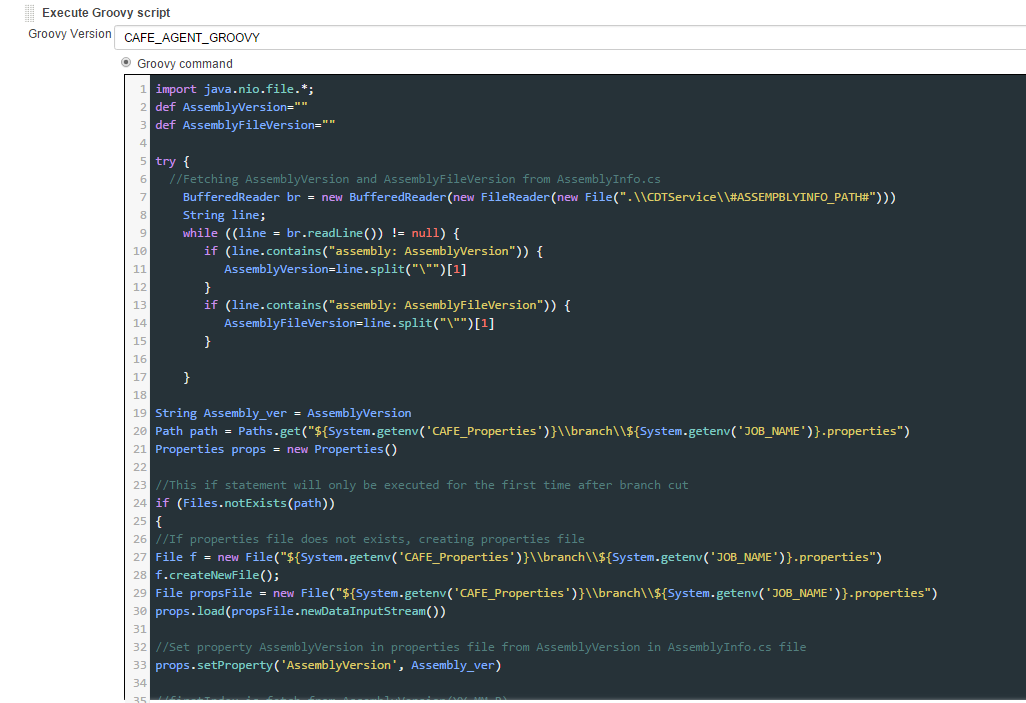
Add the Build Step as **Execute Windows batch command** to add the batch command for execution.



This command is used to copy all library files from particular repository location to the workspace of job.

1. **Execute Groovy Script**
2. Go to **🡪 Add Build Step 🡪 Execute Groovy Script**

To update the value of Assembly Version in Assemblyinfo.cs file this groovy script is used.



Steps for the script –

1. To get assembly version from Assemblyinfo.cs file.
2. If the component is trunk job then have to initialize the build identifier with 1.
3. Then update the property file with build identifier, build number and Assembly version values.
4. If the job is branch job have to initialize the value of build identifier with 200.

**The groovy script–**

import java.nio.file.\*;

def AssemblyVersion=""

def AssemblyFileVersion=""

try {

//Fetching AssemblyVersion and AssemblyFileVersion from AssemblyInfo.cs

BufferedReader br = new BufferedReader(new FileReader(new File(".\\CDTService\\#ASSEMPBLYINFO\_PATH#")))

String line;

while ((line = br.readLine()) != null) {

if (line.contains("assembly: AssemblyVersion")) {

AssemblyVersion=line.split("\"")[1]

}

if (line.contains("assembly: AssemblyFileVersion")) {

AssemblyFileVersion=line.split("\"")[1]

}

}

String Assembly\_ver = AssemblyVersion

Path path = Paths.get("${System.getenv('CAFE\_Properties')}\\branch\\${System.getenv('JOB\_NAME')}.properties")

Properties props = new Properties()

//This if statement will only be executed for the first time after branch cut

if (Files.notExists(path))

{

//If properties file does not exists, creating properties file

File f = new File("${System.getenv('CAFE\_Properties')}\\branch\\${System.getenv('JOB\_NAME')}.properties")

f.createNewFile();

File propsFile = new File("${System.getenv('CAFE\_Properties')}\\branch\\${System.getenv('JOB\_NAME')}.properties")

props.load(propsFile.newDataInputStream())

//Set property AssemblyVersion in properties file from AssemblyVersion in AssemblyInfo.cs file

props.setProperty('AssemblyVersion', Assembly\_ver)

//firstIndex is fetch from AssemblyVersion(YY.MM.R)

firstIndex = Assembly\_ver.substring(0,Assembly\_ver.lastIndexOf("."))

//Initialize build\_identifierNo to 201(branch job)

build\_identifierNo = 201

//Build number is contructed from AssemblyVersion and build\_identifierNo

Build\_Number = firstIndex + "." + build\_identifierNo.toString()

props.setProperty('BuildIdentifier', build\_identifierNo.toString())

props.setProperty('BUILD\_NUMBER',Build\_Number)

props.store(propsFile.newWriter(), null)

println "Assembly Version : " +AssemblyVersion

println "Build Number : " +Build\_Number

}

//If properties file exists, setting property AssemblyVersion in properties file

else

{

File propsFile = new File("${System.getenv('CAFE\_Properties')}\\branch\\${System.getenv('JOB\_NAME')}.properties")

props.load(propsFile.newDataInputStream())

props.setProperty('AssemblyVersion', Assembly\_ver)

props.store(propsFile.newWriter(), null)

println "Assembly Version : " +Assembly\_ver

}

} catch (Exception e) {e.printStackTrace()}

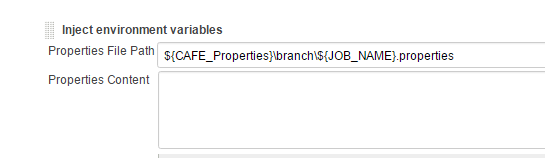
1. Inject Environment Variable

Go to 🡪 **Add Build Step** 🡪 **Inject environment variables.**

Provide properties File path, here that file path is constructed as below

**${CAFÉ\_Properties}\branch\${JOB\_NAME}.properties**

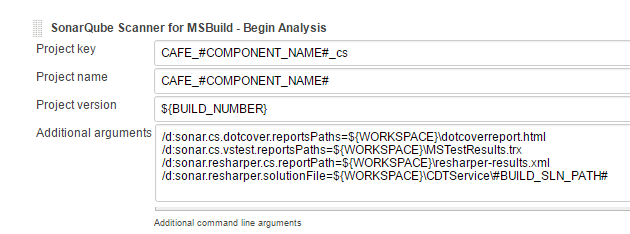
In workspace there is folder café\_properties 🡪 branch 🡪 JOB\_NAME.properties



1. Invoke SonarQube –

Go to 🡪 Add Build Step 🡪 **SonarQube Scanner for MSBuild-Begin Analysis**

1. Provide **Project key, Project name, Project version** – SonarQube properties for project.



1. Provide the **Additional arguments –**
2. Dotcover report path
3. Vstest report path
4. Resharper report path
5. Resharper solution file

/d:sonar.cs.dotcover.reportsPaths=${WORKSPACE}\dotcoverreport.html

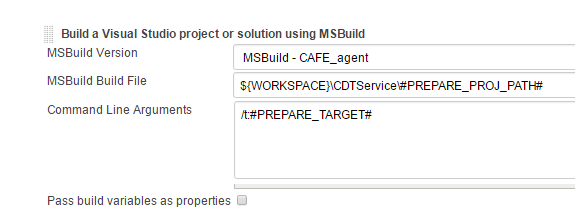
/d:sonar.cs.vstest.reportsPaths=${WORKSPACE}\MSTestResults.trx

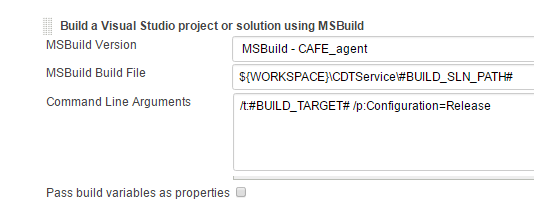
/d:sonar.resharper.cs.reportPath=${WORKSPACE}\resharper-results.xml

/d:sonar.resharper.solutionFile=${WORKSPACE}\CDTService\#BUILD\_SLN\_PATH#

The Value of #BUILD\_SLN\_PATH# is fetched from input taken in self-service job.

1. Add the Build step – **Build a Visual Studio project or solution using MSBuild.**
2. Provide the build file of job in **MSBuild Build File.**
3. Provide target in command line argument field.





1. Execute windows batch command –
2. Go to 🡪 **Add build Steps** 🡪 **Execute windows batch command**

"C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\MSTest.exe" /testcontainer:%WORKSPACE%\CDTService\#UNIT\_TEST\_PATH# /resultsfile:MSTestResults.trx

The #UNIT\_TEST\_PATH# is provided as input from self-Service.

1. Go to 🡪 **Add build Steps** 🡪 **Execute windows batch command**

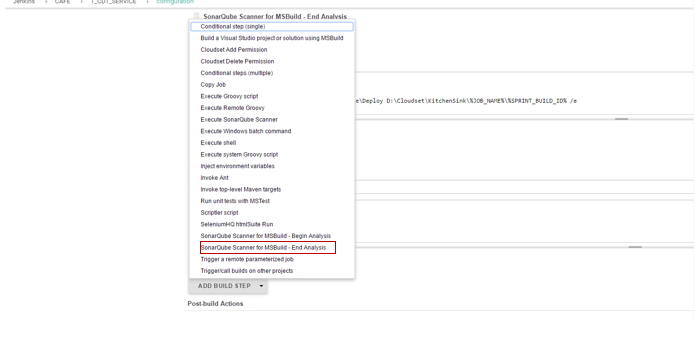
"C:\BuildAgent4\tools\dotCover\dotCover.exe" analyse /ReportType=HTML /Output="%WORKSPACE%\dotcoverreport.html" "/TargetExecutable=C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\MSTest.exe" /TargetWorkingDir=. "/TargetArguments=/testcontainer:%WORKSPACE%\CDTService\#UNIT\_TEST\_PATH#"

1. Go to 🡪 **Add build Steps** 🡪 **Execute windows batch command**

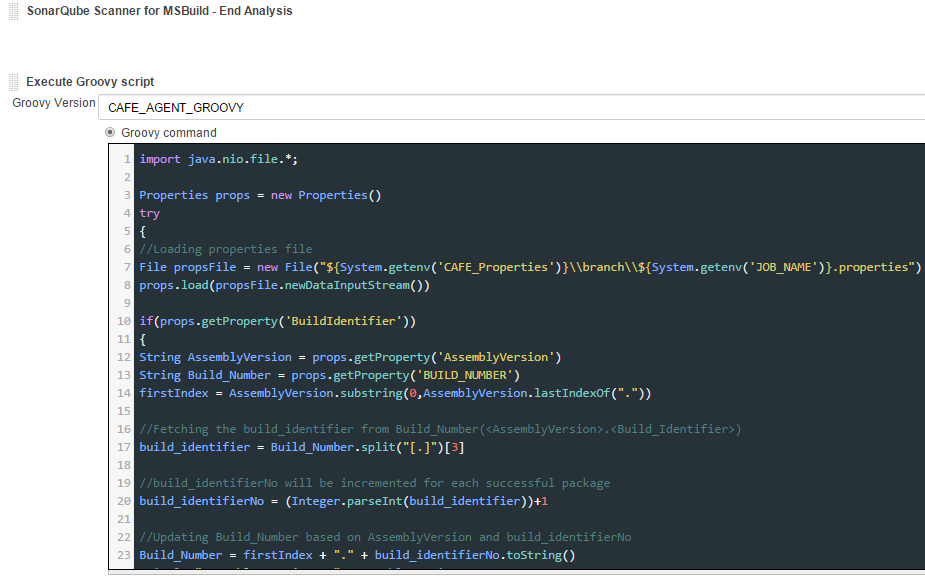
"D:\cs\tools\JetBrains.ReSharper.CommandLineTools\inspectcode.exe" /output=

"resharper-results.xml" "%WORKSPACE%\CDTService\#BUILD\_SLN\_PATH#"

1. Add Build Step **SonarQube Scanner for MSBuild-End Analysis**



1. **Groovy Script -**



Increment Build Number based on build success or failure -

1. The groovy script will only be executed when all above steps have been built successfully
2. In this step, we are fetching the Assembly Version and Build Number from properties file.
3. The Build Identifier number keeps track of last successful build. It is assign with the last Index value we get after splitting the Build Number.
4. This value is incremented for each successful build. The Build Number is updated accordingly.
5. The updated values for Build Identifier No and Build Number are written back to property file.

**The groovy script–**

import java.nio.file.\*;

Properties props = new Properties()

try

{

//Loading properties file

File propsFile = new File("${System.getenv('CAFE\_Properties')}\\branch\\${System.getenv('JOB\_NAME')}.properties")

props.load(propsFile.newDataInputStream())

if(props.getProperty('BuildIdentifier'))

{

String AssemblyVersion = props.getProperty('AssemblyVersion')

String Build\_Number = props.getProperty('BUILD\_NUMBER')

firstIndex = AssemblyVersion.substring(0,AssemblyVersion.lastIndexOf("."))

//Fetching the build\_identifier from Build\_Number(<AssemblyVersion>.<Build\_Identifier>)

build\_identifier = Build\_Number.split("[.]")[3]

//build\_identifierNo will be incremented for each successful package

build\_identifierNo = (Integer.parseInt(build\_identifier))+1

//Updating Build\_Number based on AssemblyVersion and build\_identifierNo

Build\_Number = firstIndex + "." + build\_identifierNo.toString()

println "Assembly Version : " +AssemblyVersion

println "Build Number : " +Build\_Number

//Setting updated values of BuildIdentifier and BUILD\_NUMBER to properties file

props.setProperty('BuildIdentifier', build\_identifierNo.toString())

props.setProperty('BUILD\_NUMBER',Build\_Number)

props.store(propsFile.newWriter(), null)

}

} catch (Exception e) {e.printStackTrace()}

1. **Execute Windows batch command**

Go to 🡪 Add the Build Step 🡪 **Execute Windows batch command**.

In this batch command storing the files to one folder.

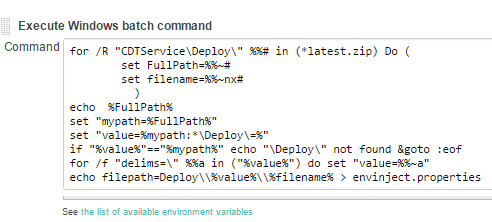


cd D:\Cloudset\KitchenSink\

mkdir %JOB\_NAME%\%SPRINT\_BUILD\_ID%

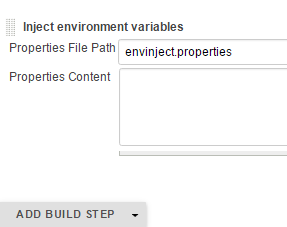
echo D | xcopy %WORKSPACE%\CDTService\Deploy D:\Cloudset\KitchenSink\%JOB\_NAME%\%SPRINT\_BUILD\_ID% /e

1. **Execute windows batch command**

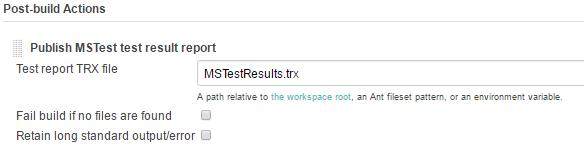


This batch script is written to get the file path which need to pass to Udeploy as a property “**Package path**”.

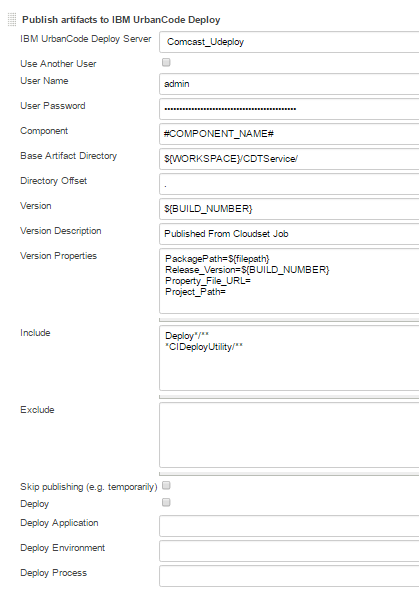
1. Inject the filepath variable to envinject.properties file



**17**. **Add Post Build action** – **Publish MSTest test result report**



**18**. **Publish Artifacts to IBM UrbanCode Deploy**



**Configuration** -

|  |  |
| --- | --- |
|  | 1. Component – Provide Job Name to Component. 2. Version – Provide the Build\_Number 3. Package path – The Package path is file path which is constructed with batch script and injected to environment properties. 4. Release version –Provide Build number 5. Include - Provide all files under deploy older as well as deploy utility folder. |

1. **Email notification**

**Click here to follow Email notification Steps**

1. Step Click on **APPLY** 🡪 **SAVE.**

## **2. Template For Service components Without Test Cases**

### **Create the template Component**

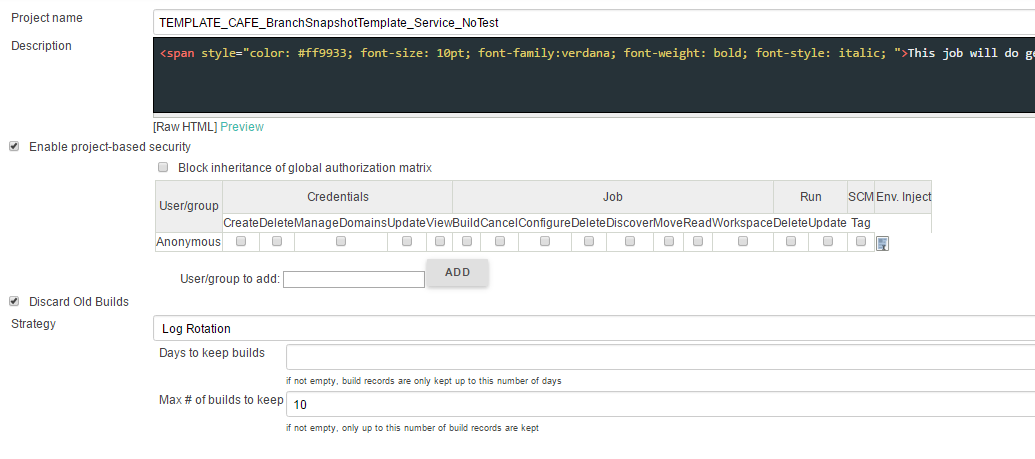
Go To **New item** 🡪 Give **Item Name** 🡪 **OK**

### **Configure the Template component**

Click on Component Name 🡪 Click on **Configure**

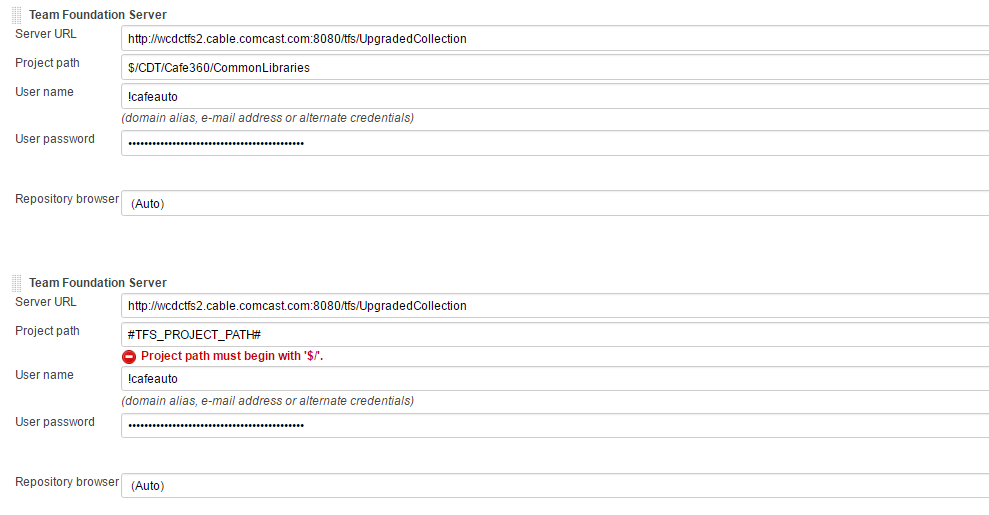
**Configuration**

1. Description – Provide the description for template job
2. Check discard old build and provide the number of build need to keep in **Max # of build to keep** part

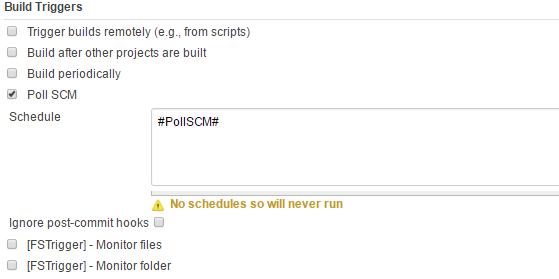


<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Quality gate based on code metrics and publish the artifacts into IBM udeploy every 30 mins in case of any code changes in <u>#COMPONENT\_NAME#</u> component of CAFE application</span>

1. JDK –Here need to provide agent which is **JAVA\_CAFE\_Agent\_a5q.**
2. Select the **Multiple SCMs** In **Source Code Management**, select **Team Foundation Server**, and provide **Repository URL**, and credential of TFS.

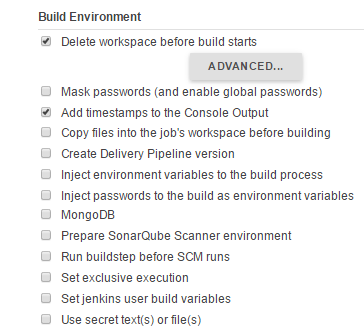


1. Go to 🡪**Build Triggers** 🡪Tick the **Poll SCM checkbox**



The value **#PollSCM#** is taken as input from self-service job.

1. In **Build Environment**, tick **Delete workspace before build starts**
2. In **Build Environment**, tick **Add timestamps to Console Output**

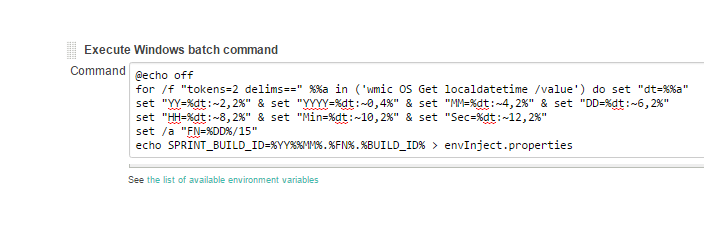


1. **Execute Windows batch command**
2. Go to🡪Add the Build Step 🡪 **Execute Windows batch command**

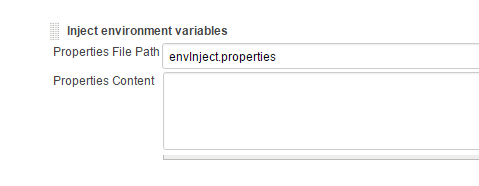


This command is used to copy all library files from particular repository location to the workspace of job.

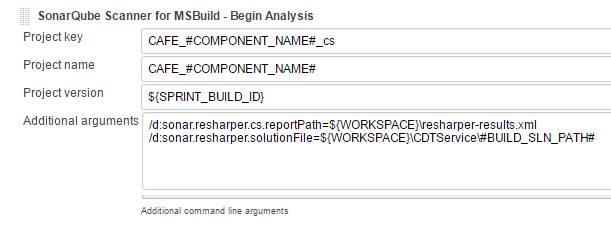
1. Go to🡪Add the Build Step 🡪 **Execute Windows batch command**



This batch command is used to construct the Sprint Build Id.This Sprint Build Id is   
stored in Inject Environment .properties file.so the value of Sprint Build id is stored in Environment variables of cloudSet.



1. Go to 🡪 **Add Build Step** 🡪 **SonarQube Scanner for MSBuild**
2. Provide **Project key, Project name, Project version** – SonarQube properties for project.



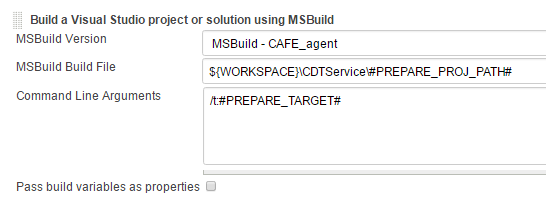
1. Provide the **Additional arguments –**
2. Resharper report path
3. Resharper solution file

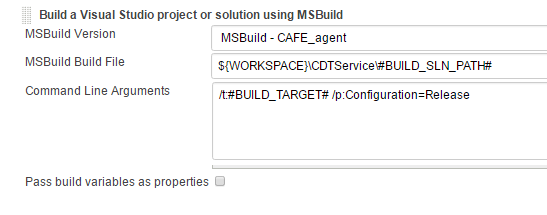
/d:sonar.resharper.cs.reportPath=${WORKSPACE}\resharper-results.xml

/d:sonar.resharper.solutionFile=${WORKSPACE}\CDTService\#BUILD\_SLN\_PATH#

The Value of #BUILD\_SLN\_PATH# is fetched from input taken in self-service job.

1. Go to 🡪 **Add build step** 🡪 **Build a visual Studio Project or solution using MSBuild**





1. Add windows batch command.

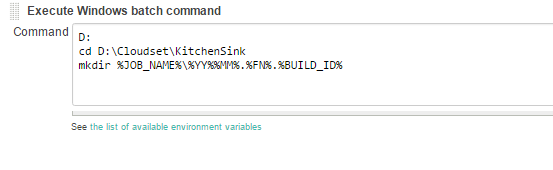
Go to🡪 **Add Build Step** 🡪 **Execute Windows batch command**



This batch command is used to produce the Resharper-Result.xml file.

1. Add SonarQube End analysis step.
2. Execute windows batch command –

Go to 🡪 **Add Build step** 🡪 **Execute windows batch command**



Storing all the file in one folder which is constructed with job name and version number.

1. Email notification –

**Click here for Email Notification steps**

1. Click on **APPLY** 🡪 **SAVE.**

## **3. Template For Database components**

### **Create the template Component**

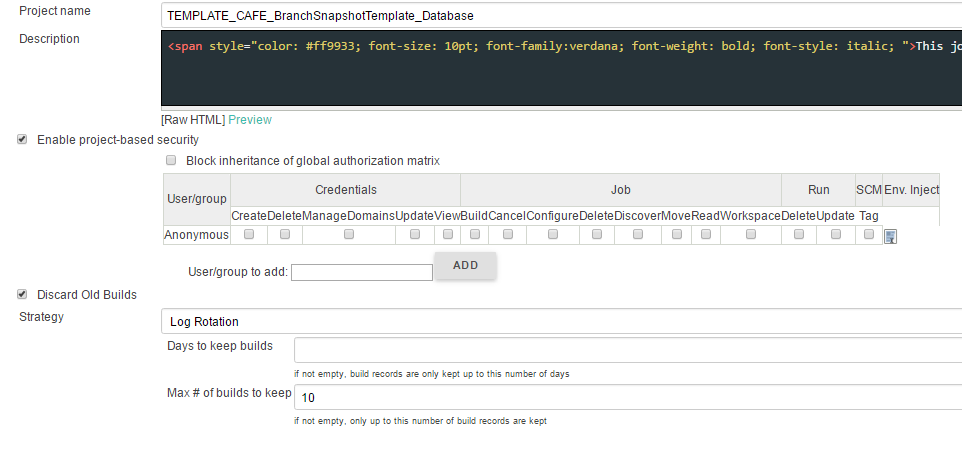
Go To **New item** 🡪 Give **Item Name** 🡪 **OK**

### **Configure the Template component**

Click on Component Name 🡪 Click on **Configure**

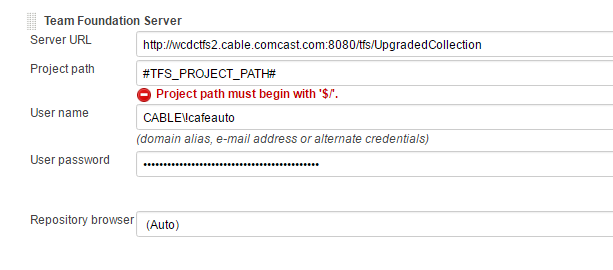
**Configuration**

1. Description – Provide the description for template job
2. Check discard old build and provide the number of build need to keep in **Max # of build to keep** part

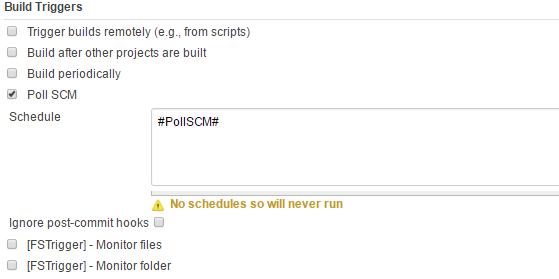


<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Quality gate based on code metrics and publish the artifacts into IBM udeploy every 30 mins in case of any code changes in <u>#COMPONENT\_NAME#</u> component of CAFE application</span>

1. JDK –Here need to provide agent which is **JAVA\_CAFE\_Agent\_a5q.**
2. Select the **Multiple SCMs** In **Source Code Management**, select **Team Foundation Server**, and provide **Repository URL**, and credential of TFS.

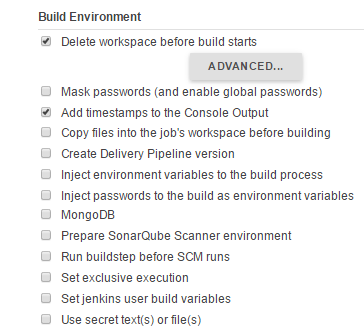


1. Go to 🡪**Build Triggers 🡪** Tick the **Poll SCM checkbox**



The value **#PollSCM#** is taken as input from self-service job.

1. In **Build Environment**, tick **Delete workspace before build starts**
2. . In **Build Environment**, tick **Add timestamps to Console Output**

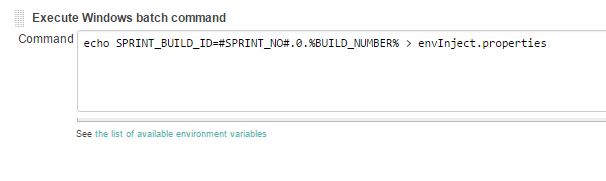


1. **Execute Windows batch command**
2. Go to 🡪 Add Build Step 🡪 **Execute Windows batch command**



This command is used to copy all the files from particular repository location to the workspace of job.

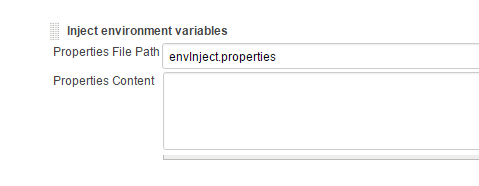
1. Go to 🡪 Add Build Step 🡪 **Execute Windows batch command**



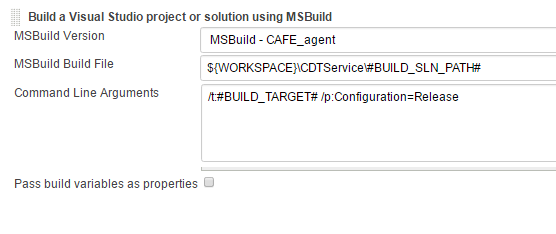
This batch command is used to inject the Sprint Build Id to properties file.

1. Inject Environment Variable

Go to 🡪 **Add Build Step** 🡪 **inject environment variable**



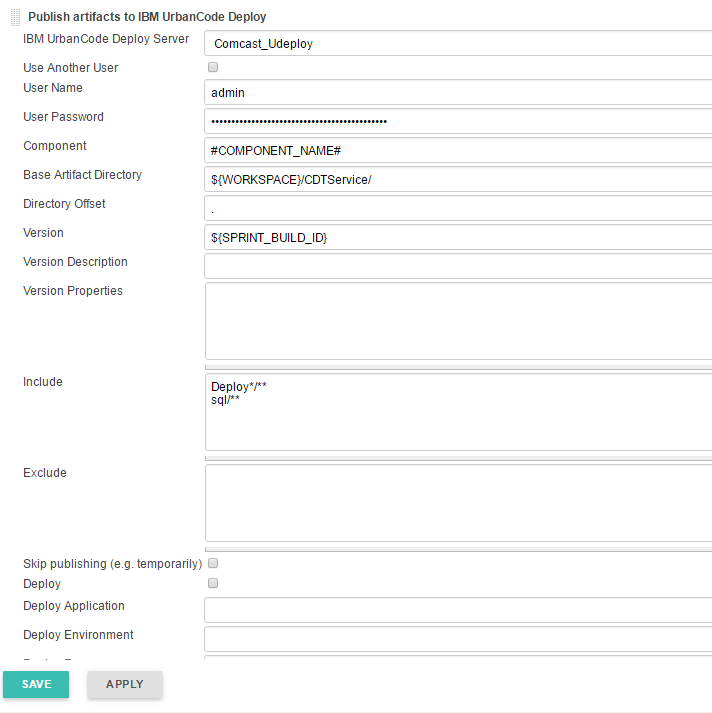
1. Invoke SonarQube Scanner
2. Go to 🡪 **Add Build Step** 🡪 **Build a visual Studio Project or solution using MSBuild**



The Value of #BUILD\_SLN\_PATH# is fetched from input taken in self-service job.

1. Publish artifact to Udeploy

Go to 🡪**Add post build Step** 🡪**Publish artifacts to IBM UrbanCode Deploy**



**Configuration** -

|  |  |
| --- | --- |
|  | 1. Component – Provide #Component\_Name# which is taken as input from self-service. 2. Version – Provide the Sprint\_Build\_Id 3. Include – Need to include all files under deploy older as well as all sql files. |

1. Email notification –

**Click here for Email Notification steps**

1. Step Click on **APPLY** 🡪 **SAVE.**

## **4. Template For Widgets componentes without test cases**

### **Create the template Component**

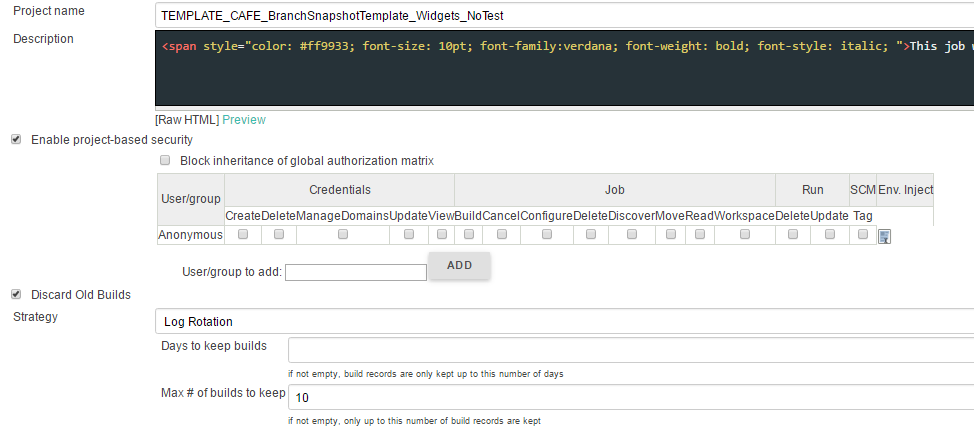
Go To **New item** 🡪 Give **Item Name** 🡪 **OK**

### **Configure the Template component**

Click on Component Name 🡪 Click on **Configure**

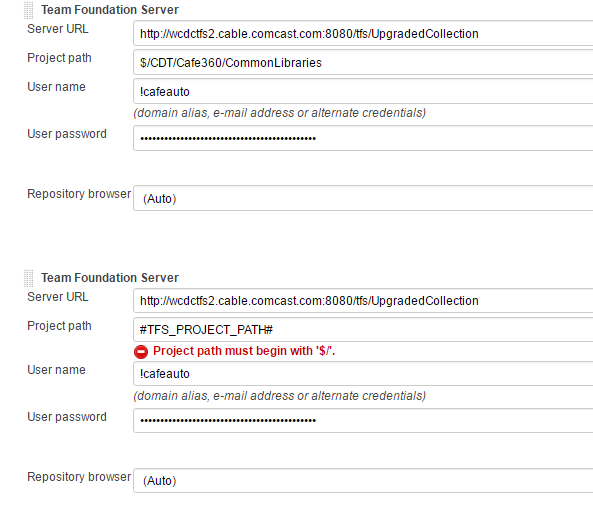
**Configuration**

1. Description – Provide the description for template job
2. Check discard old build and provide the number of build need to keep in **Max # of build to keep** part

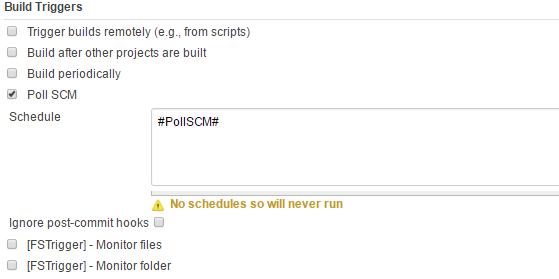


<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Quality gate based on code metrics and publish the artifacts into IBM udeploy every 30 mins in case of any code changes in <u>#COMPONENT\_NAME#</u> component of CAFE application</span>

1. JDK –Here need to provide agent which is **JAVA\_CAFE\_Agent\_a5q.**
2. Select the **Multiple SCMs** In **Source Code Management**, select **Team Foundation Server**, and provide **Repository URL**, and credential of TFS.

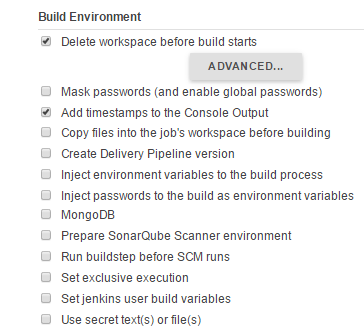


1. Go to 🡪**Build Triggers** 🡪Tick the **Poll SCM checkbox**



The value **#PollSCM#** is taken as input from self-service job.

1. In **Build Environment**, tick **Delete workspace before build starts**
2. . In **Build Environment**, tick **Add timestamps to Console Output**



1. **Click here and follow steps 8 to 13 for the configuration**
2. Email notification –

**Click here for Email Notification steps**

1. Step Click on **APPLY** 🡪 **SAVE.**

## **5. Template For Widgets Components with Test cases**

### **Create the template Component**

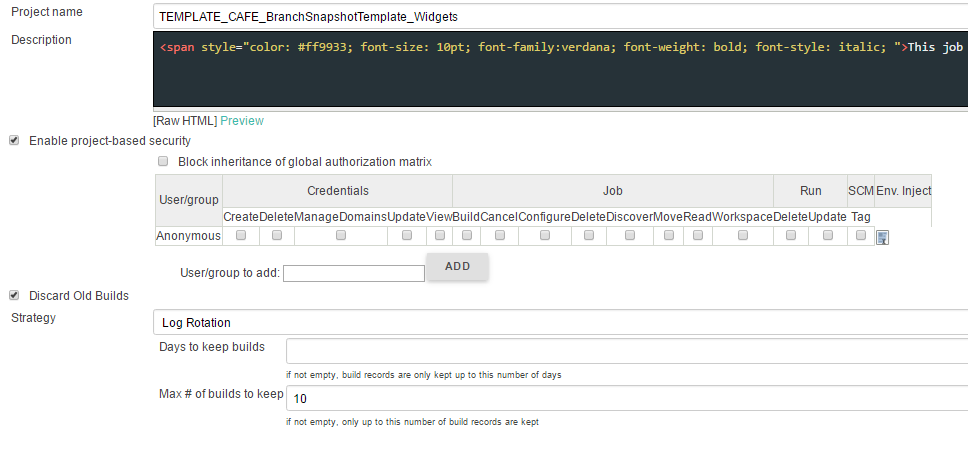
Go To **New item** 🡪 Give **Item Name** 🡪 **OK**

### **Configure the Template component**

Click on Component Name 🡪 Click on **Configure**

**Configuration**

1. Description – Provide the description for template job
2. Check discard old build and provide the number of build need to keep in **Max # of build to keep** part

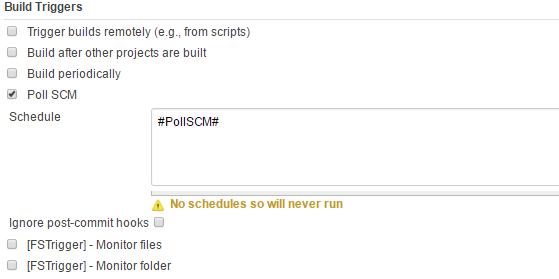


<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Quality gate based on code metrics and publish the artifacts into IBM udeploy every 30 mins in case of any code changes in CAFE <u>#COMPONENT\_NAME#</u> component of CAFE application</span>

1. JDK –Here need to provide agent which is **JAVA\_CAFE\_Agent\_a5q.**
2. Select the **Multiple SCMs** In **Source Code Management**, select **Team Foundation Server**, and provide **Repository URL**, and credential of TFS.

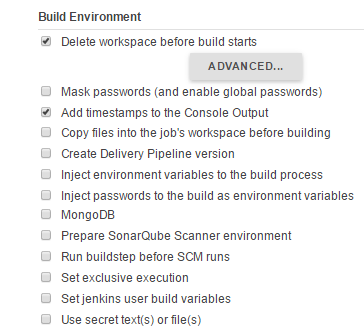


1. Go to 🡪**Build Triggers** 🡪Tick the **Poll SCM checkbox**



The value **#PollSCM#** is taken as input from self-service job.

1. In **Build Environment**, tick **Delete workspace before build starts**
2. In **Build Environment**, tick **Add timestamps to Console Output**

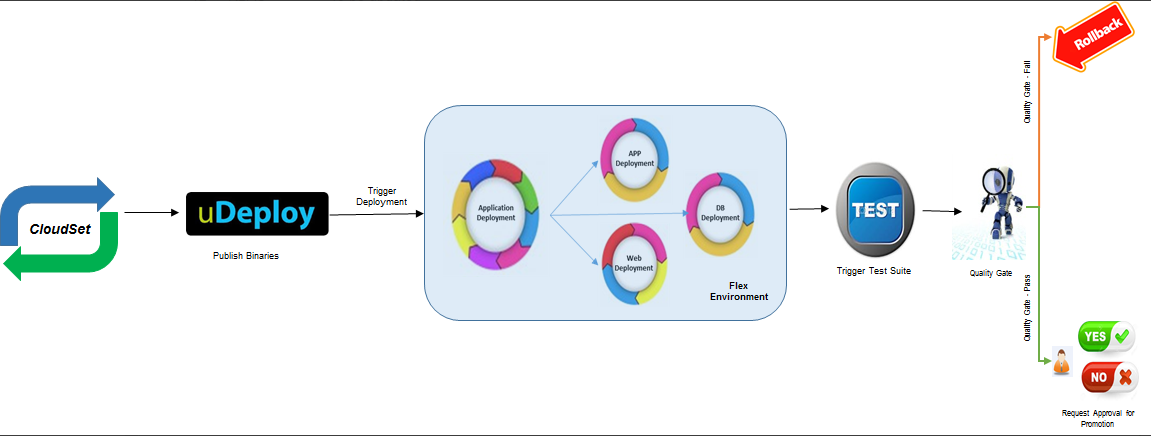


1. **Click here and follow steps 8 to 13 for the configuration**
2. Email notification –

**Click here for Email Notification steps**

1. Step Click on **APPLY** 🡪 **SAVE.**

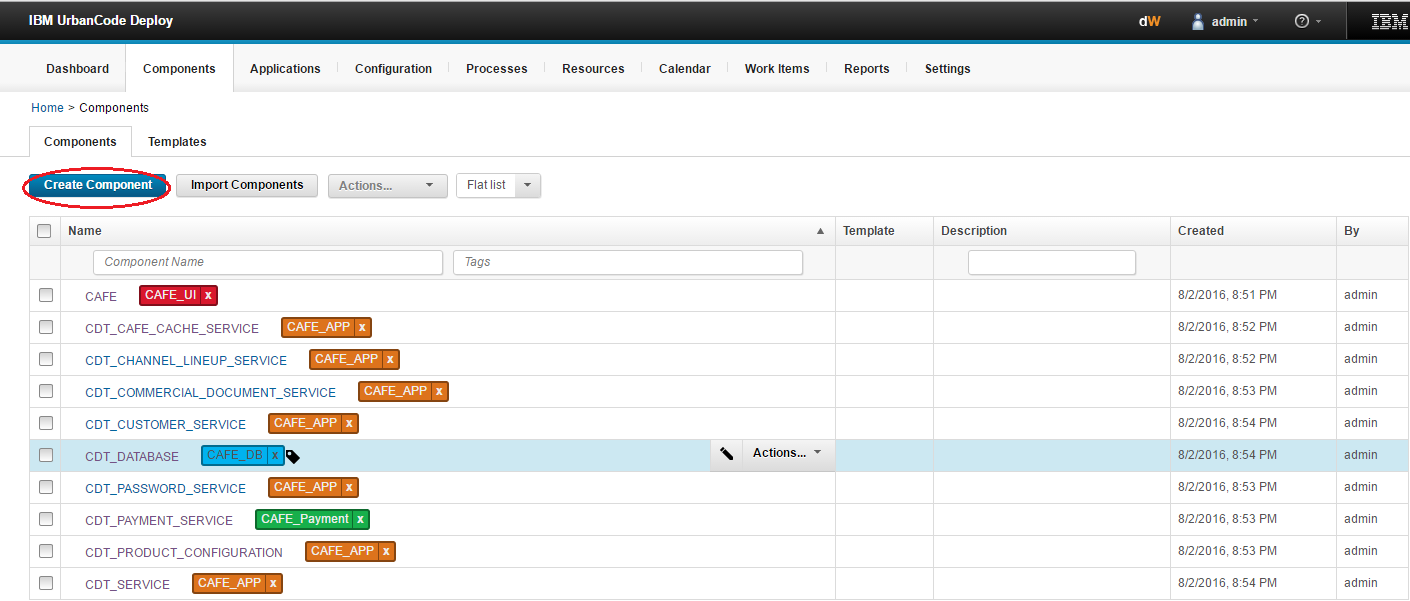
# Overview and CD Architecture



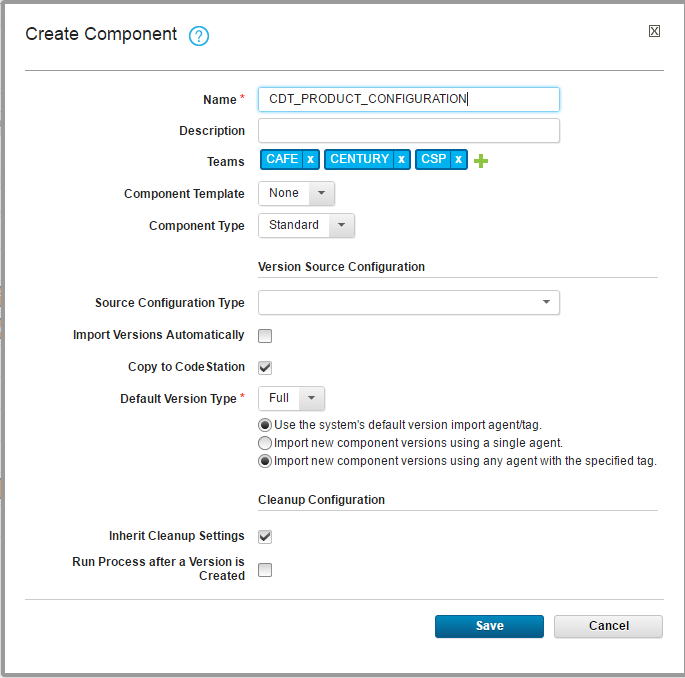
# Component

## **1. Create Components**

Go to 🡪 IBM Udeploy 🡪 click on **Component** 🡪 Click on **Create Component**



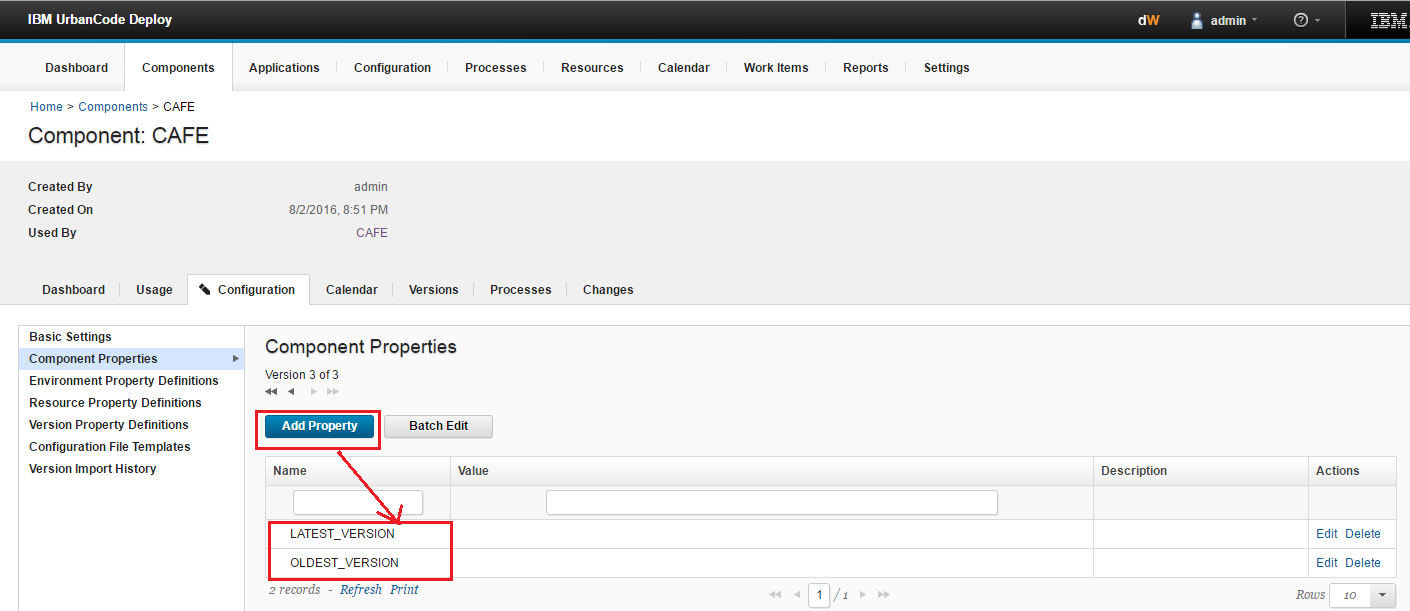
Provide the necessary details as shown in screenshot.



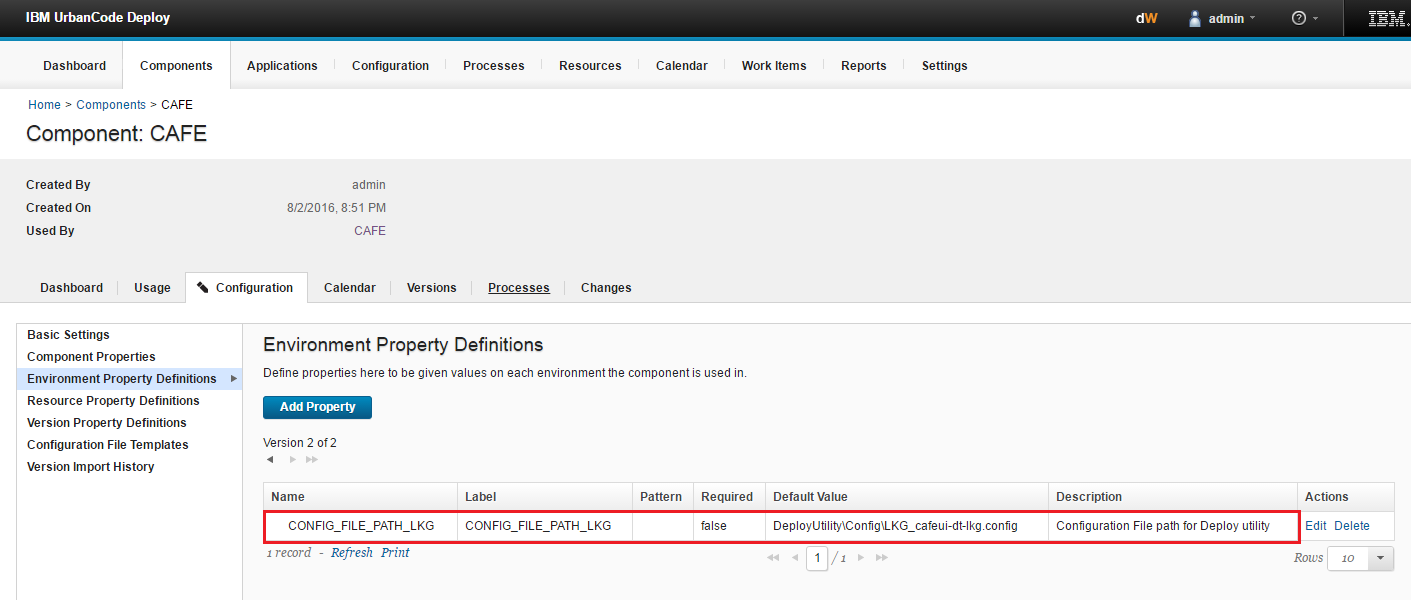
## **2. Add Properties**

* 1. Go to 🡪 Component 🡪 configuration 🡪component properties 🡪 add property

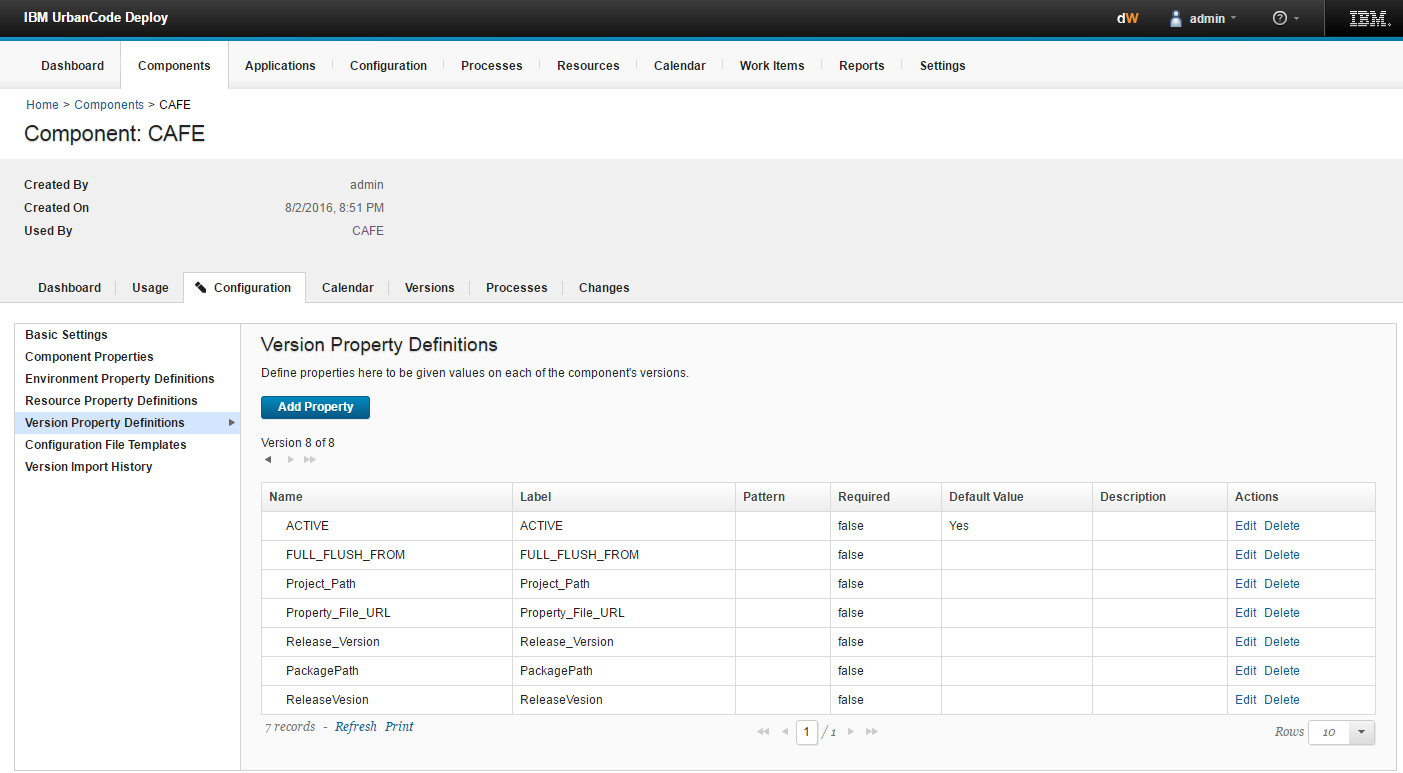
Add Latest \_Version and Oldest\_Version property



* 1. Go to 🡪 Component 🡪 configuration 🡪Environment Property definitions 🡪 add property

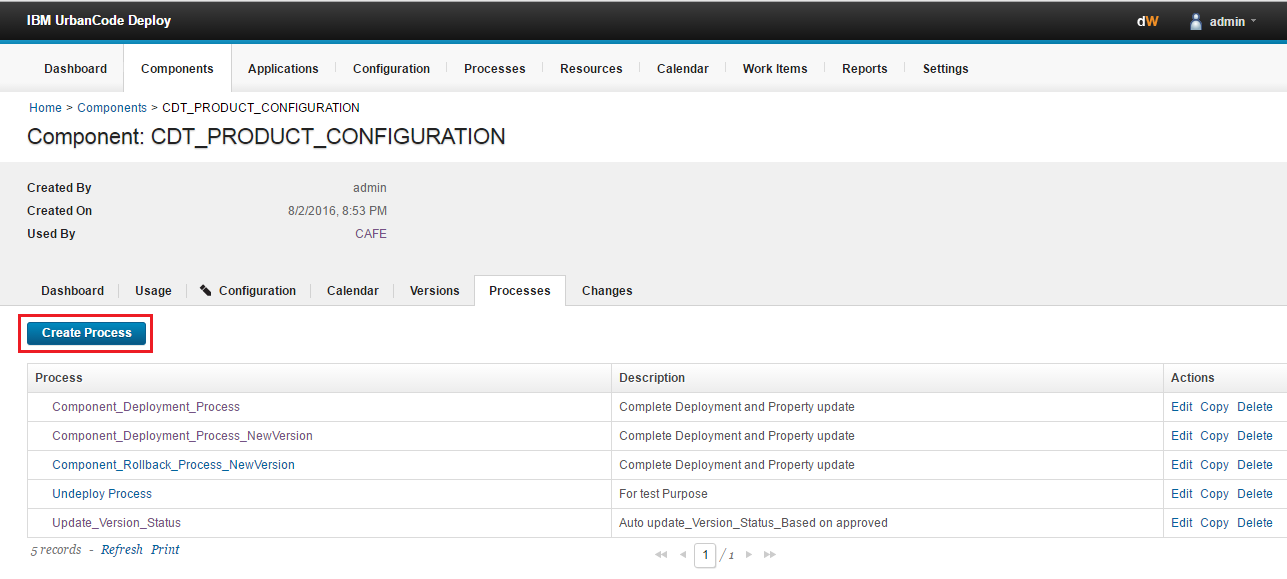


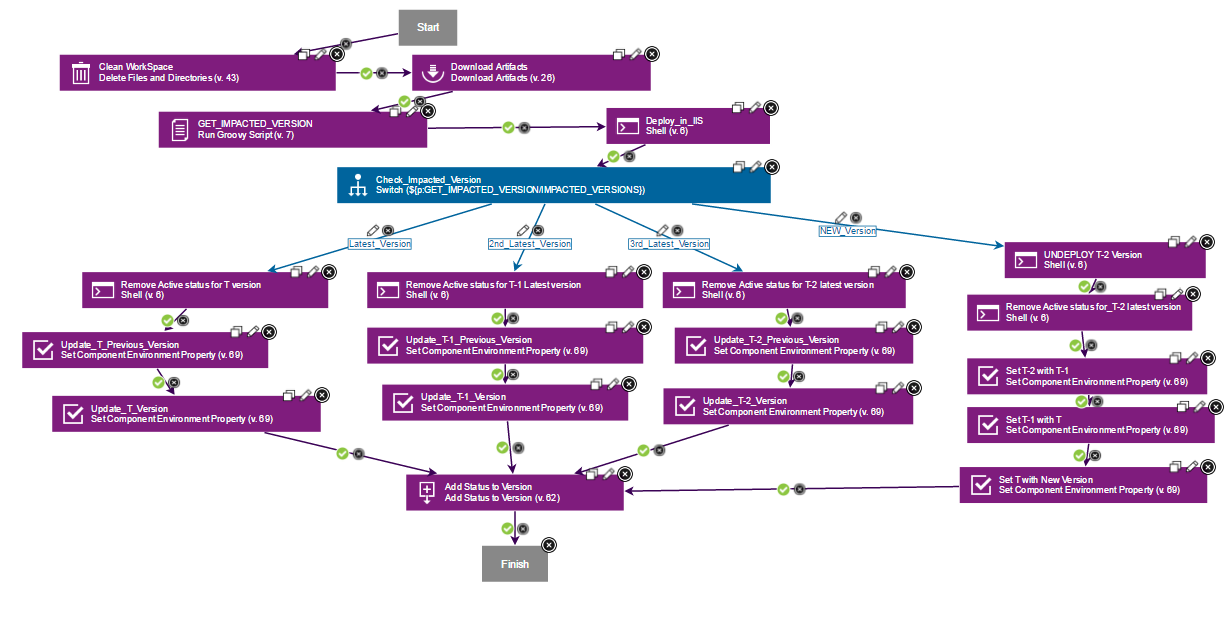
* 1. Go to 🡪 **Component** 🡪 **configuration** 🡪**Version Property definitions** 🡪 add property



## **3. Component Process**

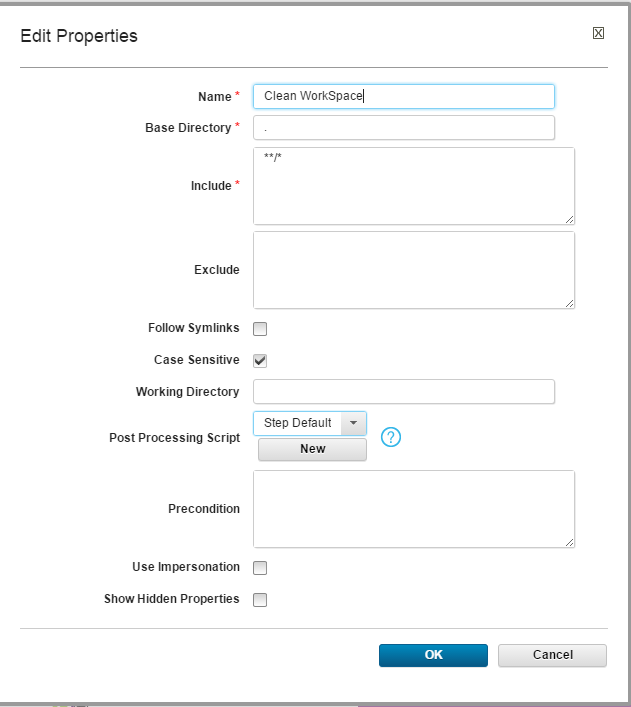
Go to 🡪 **Component Name** 🡪**Processes**🡪 **Create process**





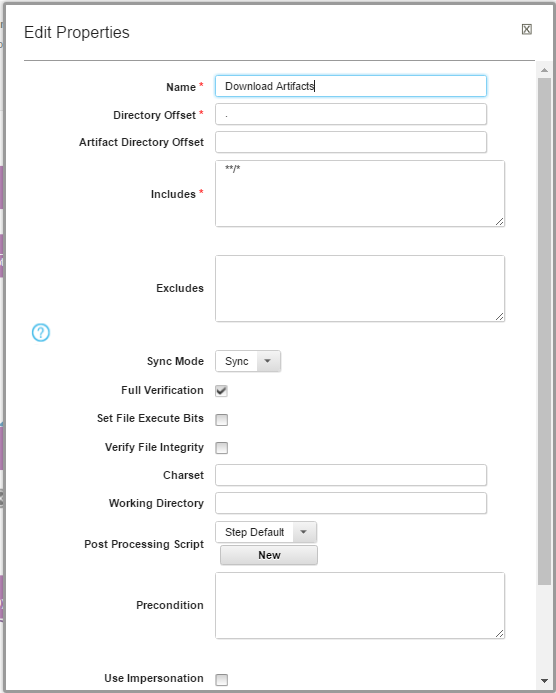
Steps Designed in process with configurations –

* 1. Clean Workspace –

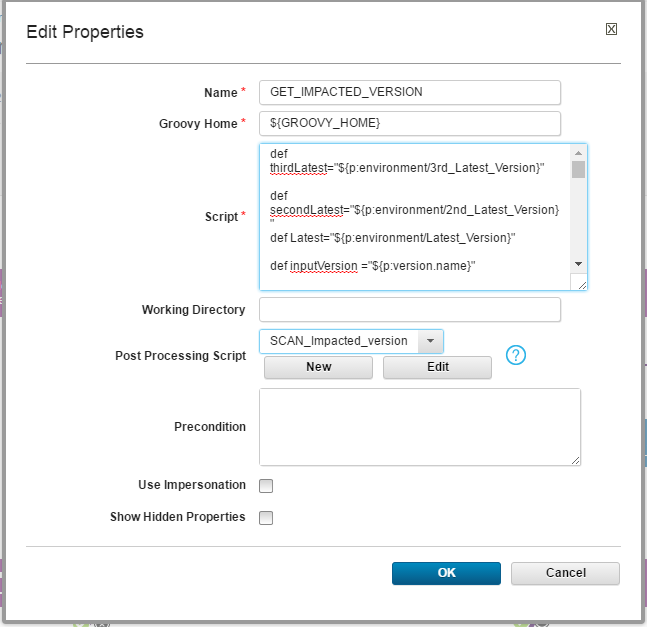


* 1. Download artifact – Download the artifact to code station.

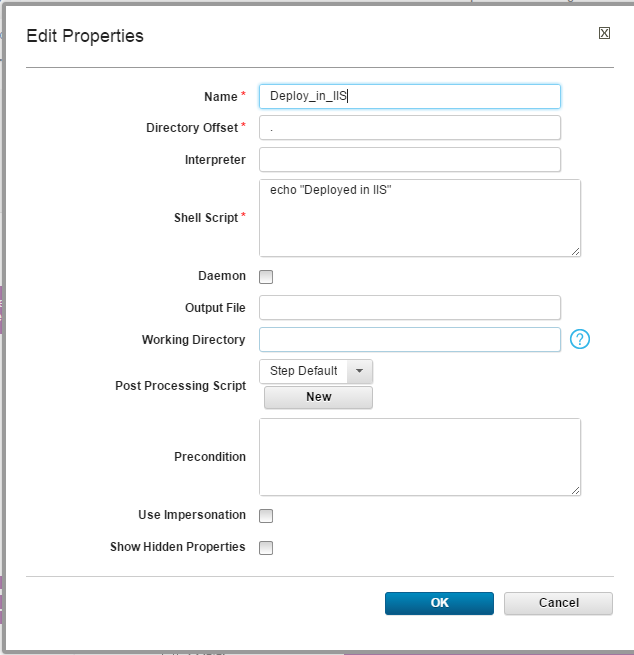
Provide the Name 🡪 **Save**.



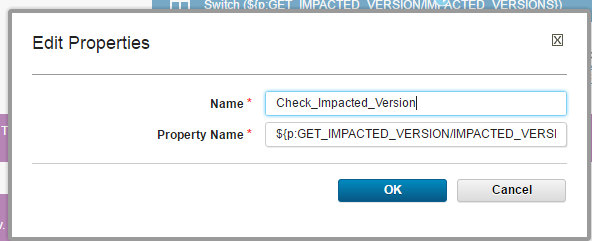
* 1. Get Impacted version –
     1. Name - Provide name of the step
     2. Groovy name - Provide the name of groovy Present in system
     3. Script - To get impacted version, provide the groovy script which will match the versions.
     4. Post processing script – Select the script from dropdown list.



* 1. Deploy in IIS –



* 1. Switch statement –



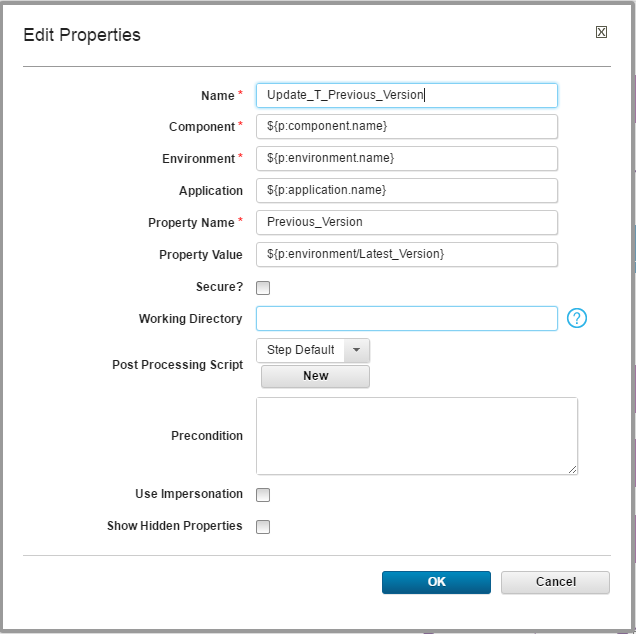
To check the impacted version use Switch.

* 1. 1st Latest Version –
     1. Remove artifact status – Provide the shell Script to remove the old version status and provide the necessary tags.

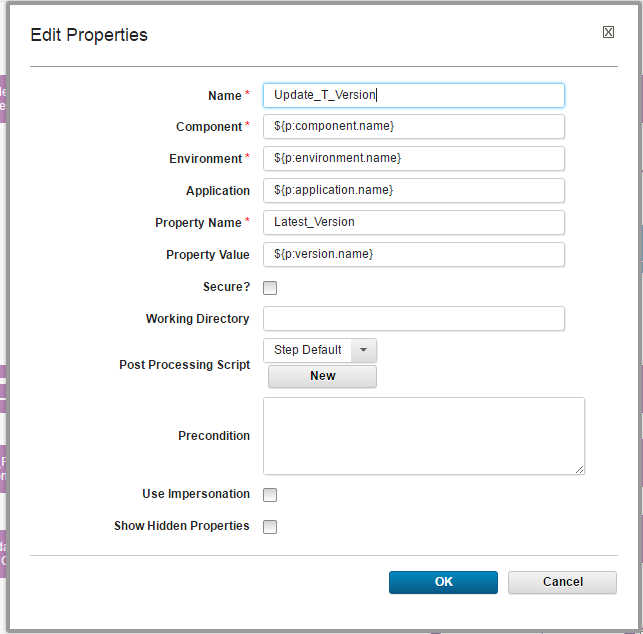


* + 1. Update\_Previous\_Version – Provide Name, Component Name, Environment and property name.

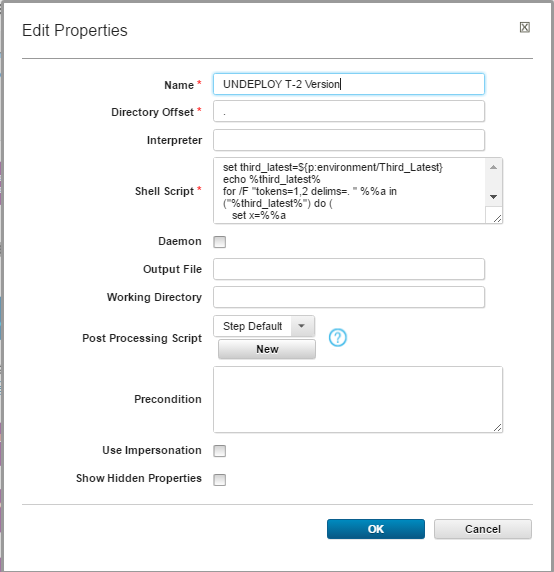
Provide the properties to respective fields.



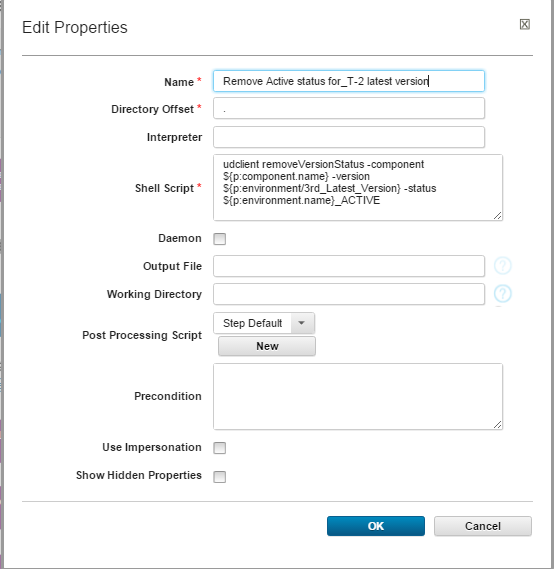
* + 1. Update\_T\_Version – Provide the properties to respective fields.



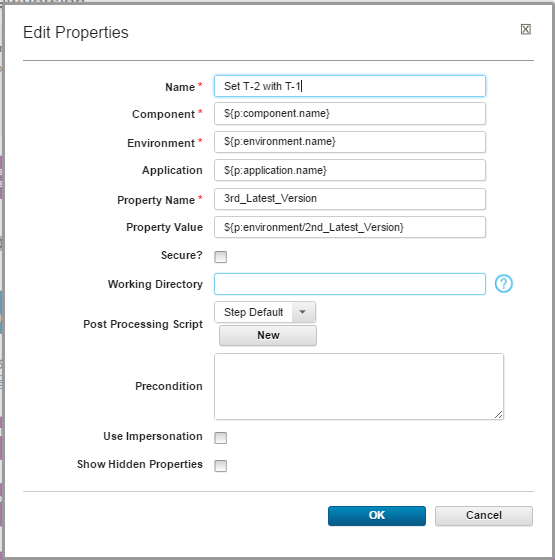
* 1. Repeat step vi for 2ed latest and 3rd latest versions
  2. New Version –
     1. Undeploy Version – Provide the shell script, to undeploy the third latest version.



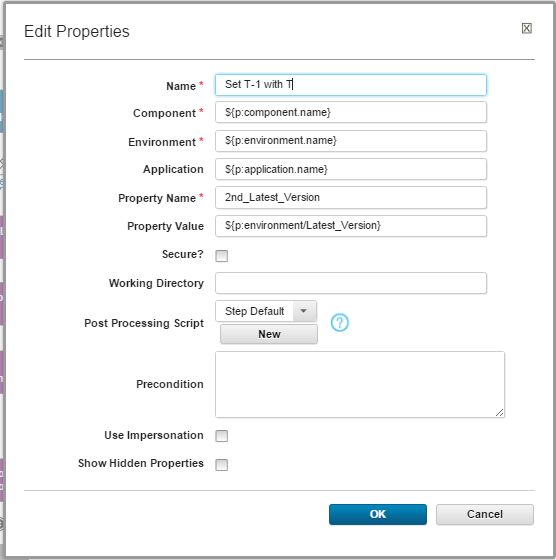
* + 1. Remove active status –



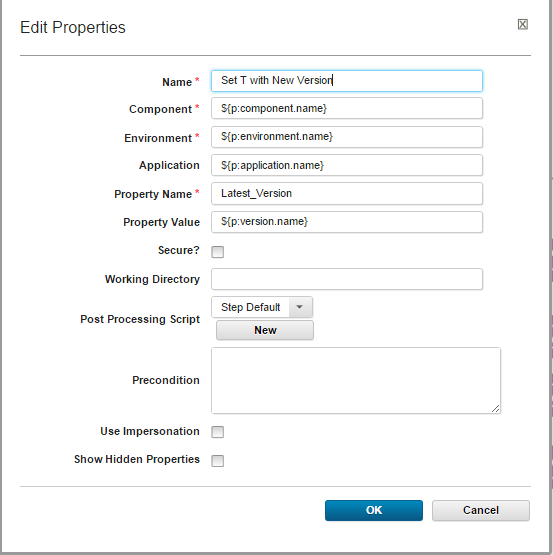
* + 1. Set T -2 with T1



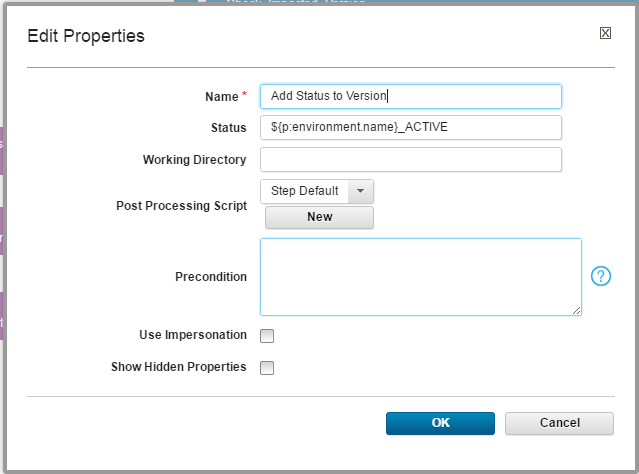
* + 1. Set T-1 with T



* + 1. Set T with new version

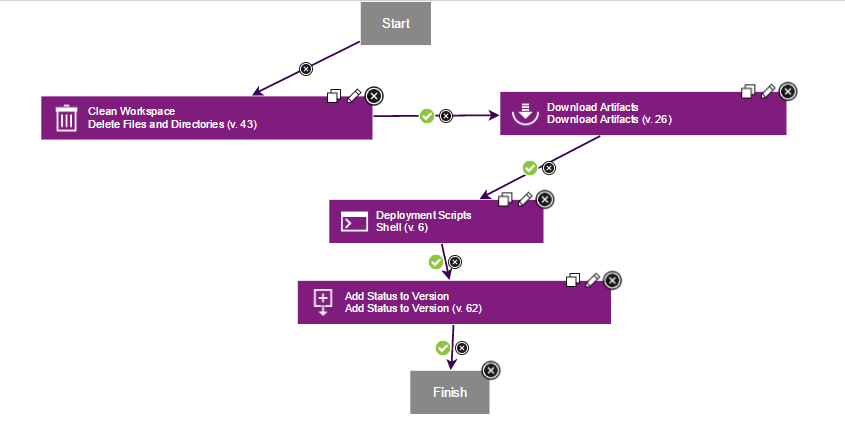


* 1. Add status –



* 1. Finish

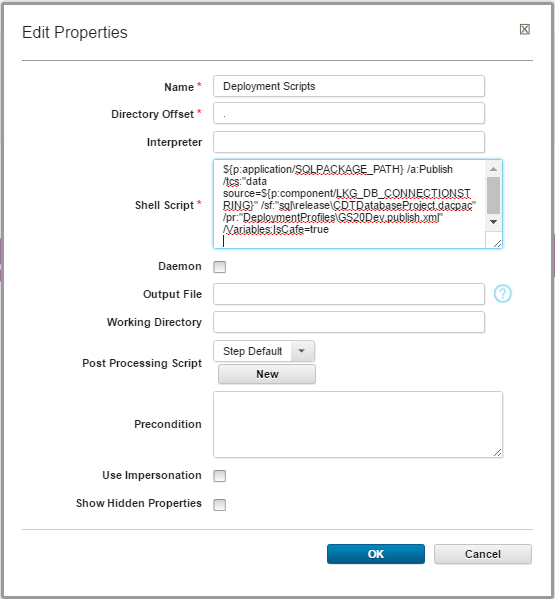
## **4. DB\_Deployment Process**



Steps –

1. Deployment Script – Provide the shell script for deployment of database. Here The shell script is as follow

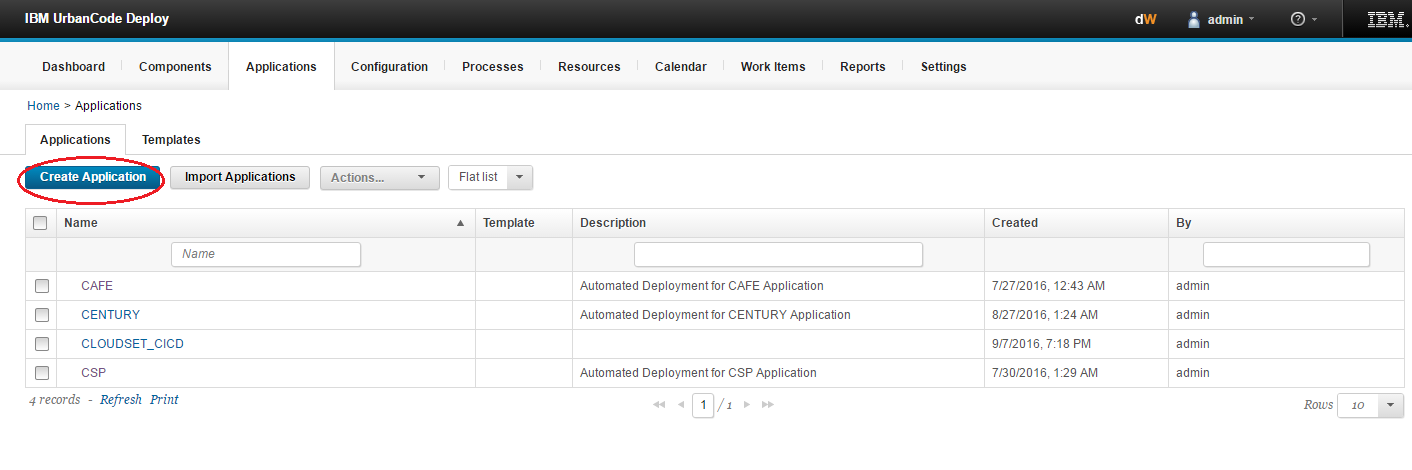
${p:application/SQLPACKAGE\_PATH} /a:Publish /tcs:"data source=${p:component/LKG\_DB\_CONNECTIONSTRING}" /sf:"sql\release\CDTDatabaseProject.dacpac" /pr:"DeploymentProfiles\GS20Dev.publish.xml" /Variables:IsCafe=true



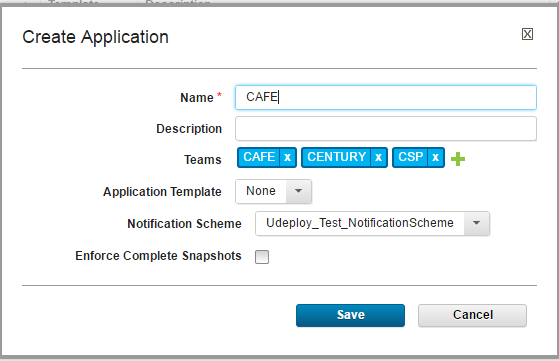
# Application

## **1. Create Application**

Go to 🡪Application 🡪 Create application



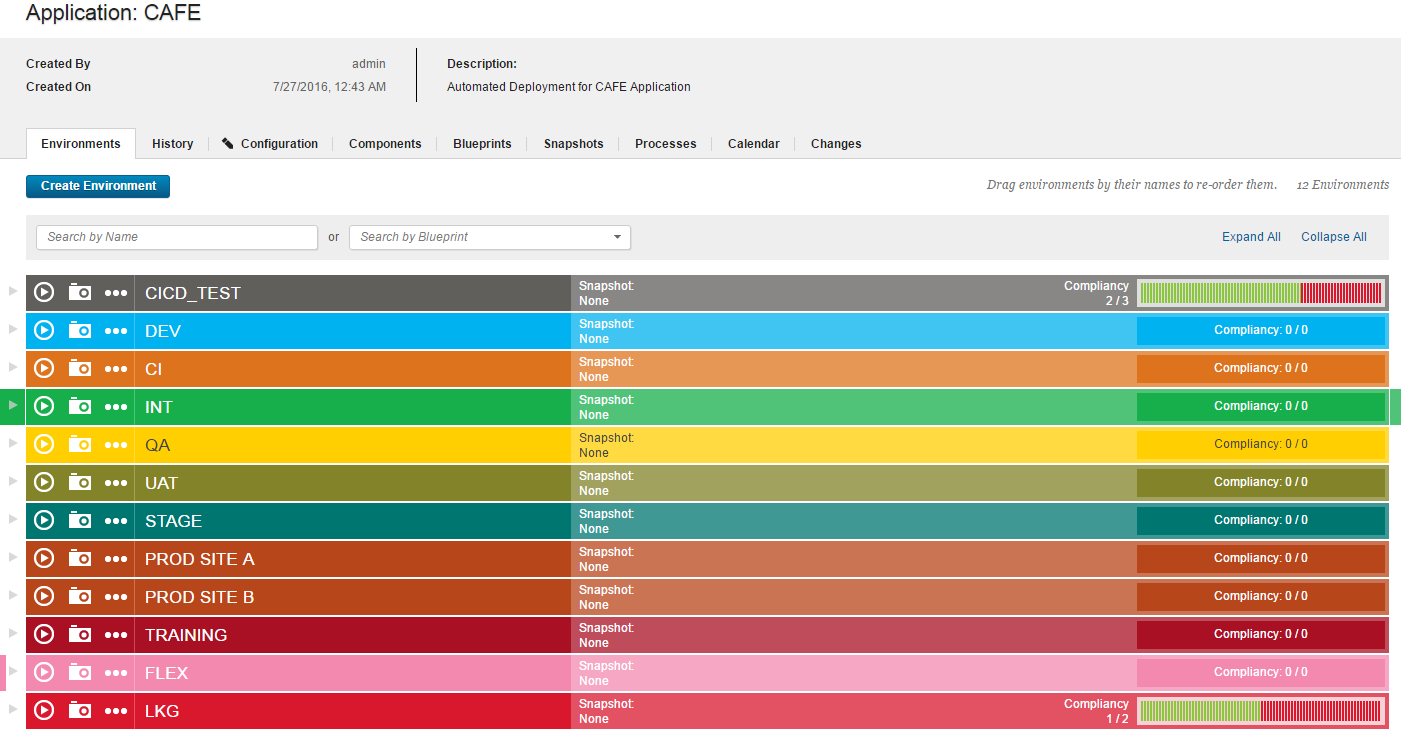
Provide necessary details.



## **2. Create Environment**

Go to 🡪 **Application** 🡪 click on **Application name** 🡪 **Create Environment**

Create the environment for different Stages as shown below.

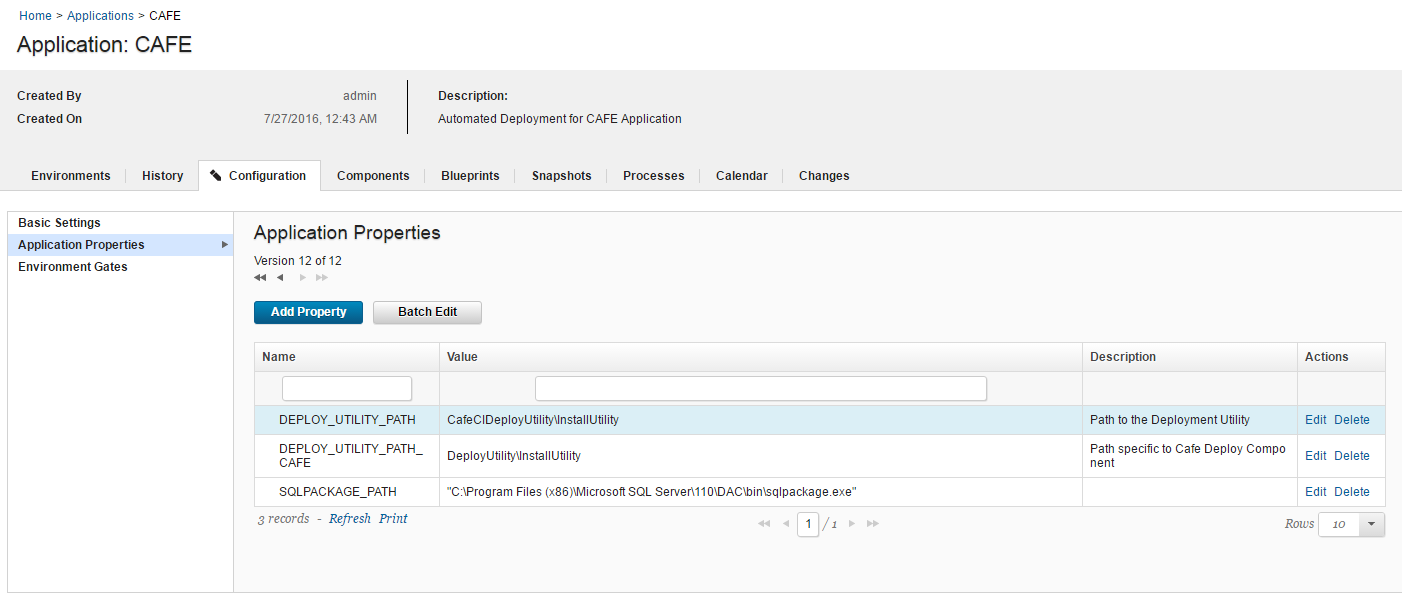


## **3. Add Properties**

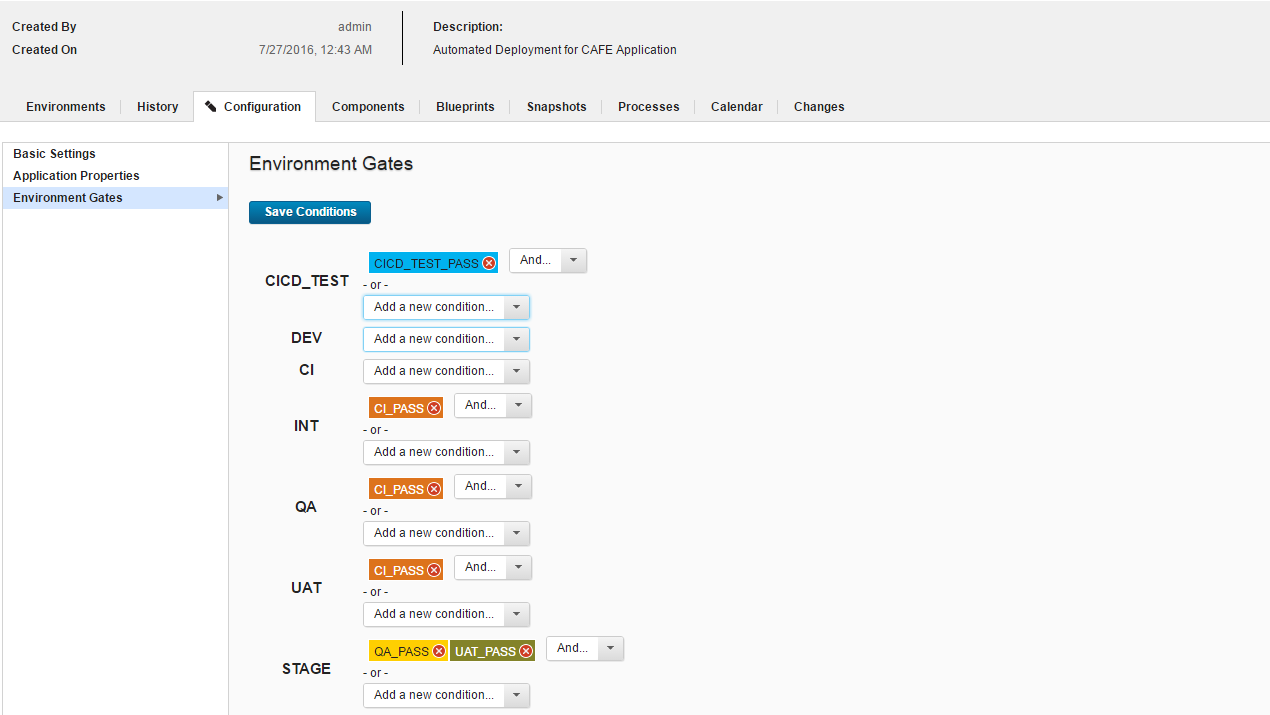
* 1. Go to 🡪 **Application** 🡪 **Application name** 🡪 **Configuration** 🡪**Application Properties** 🡪 **Add property**

Provide properties for

* + 1. Deploy utility path
    2. SQL package path



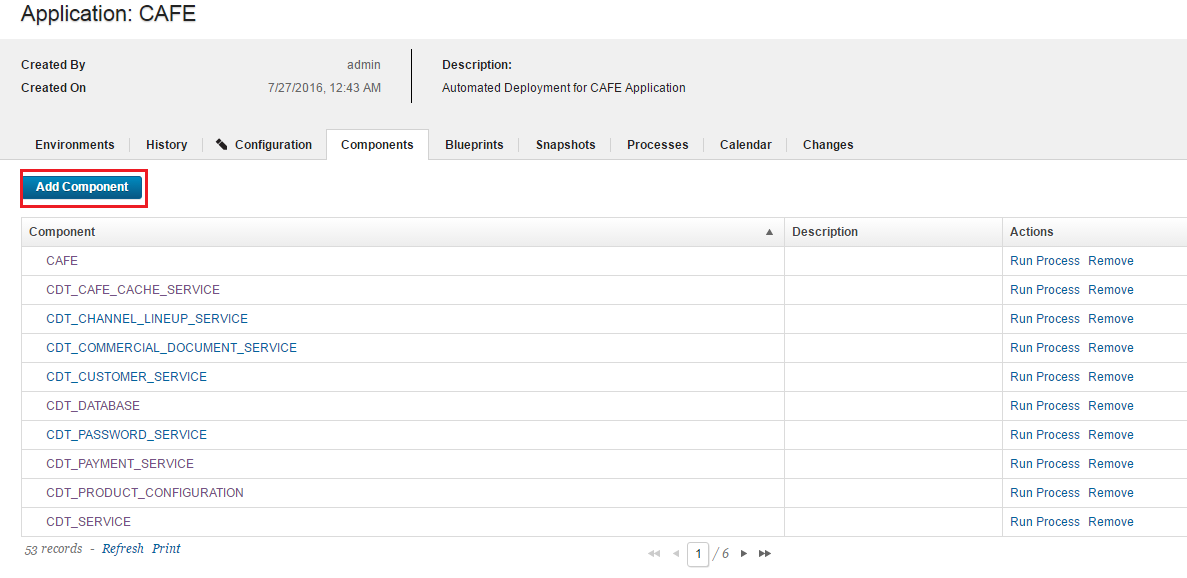
* 1. Go to 🡪 **Application** 🡪 **Application name** 🡪 **Configuration** 🡪**Environment gates** 🡪 **Save** conditions



## **4. Map Component**

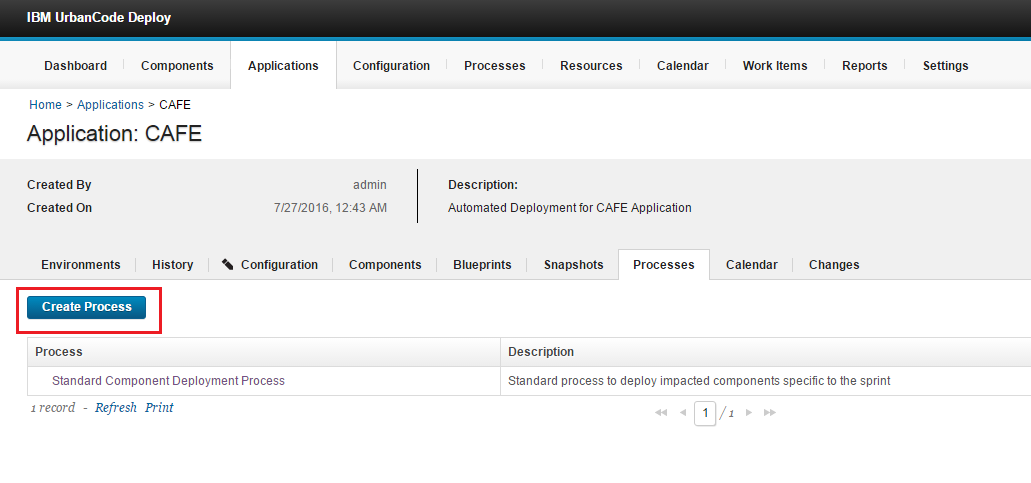
Go to 🡪 **Application name**🡪 **Components** 🡪 Add components

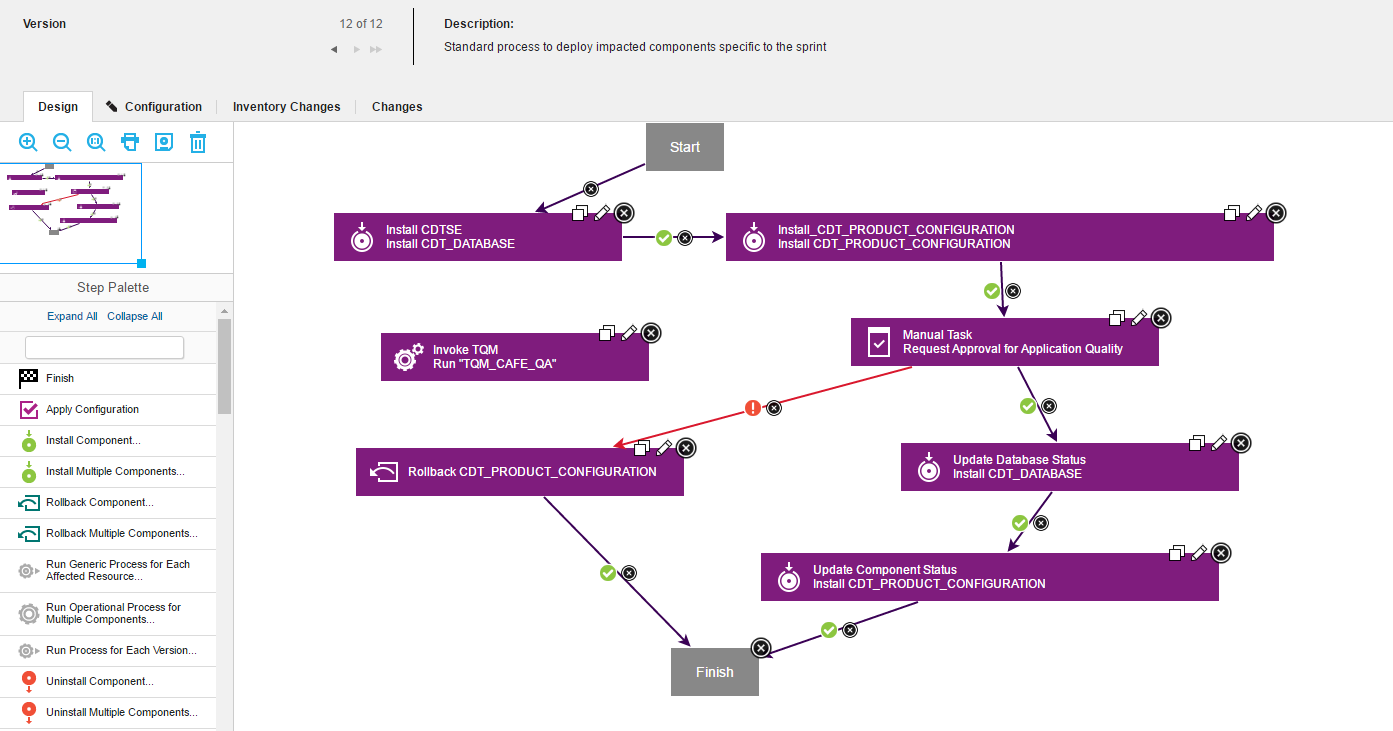
Map all café components to application.



## **5. Application Process**

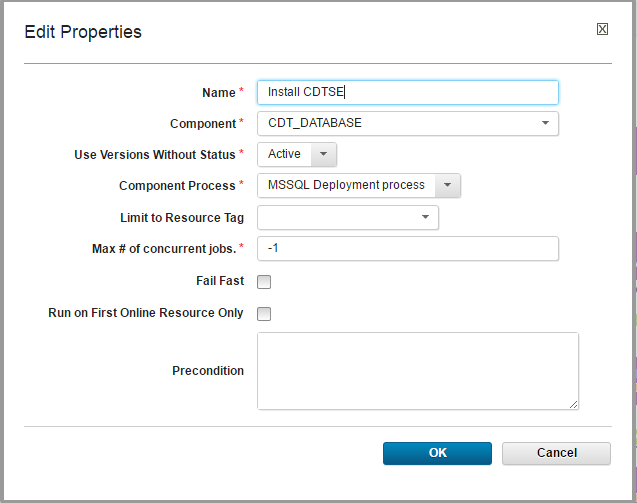
Go to 🡪 **Application** 🡪 **Processes** 🡪**Create process**



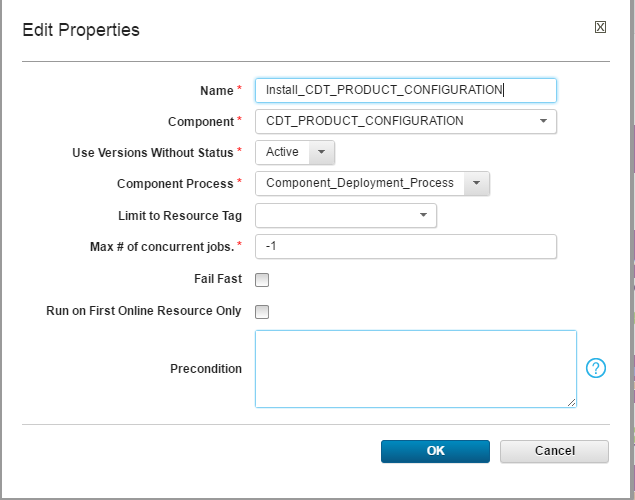


Steps –

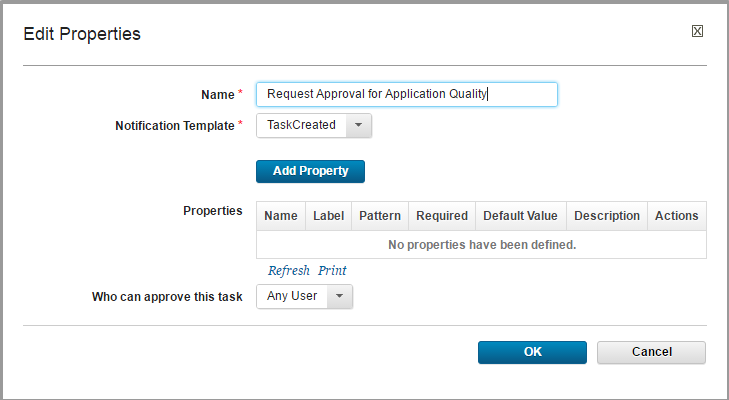
1. Install database –
   * 1. Name –Provide name of the step.
     2. Component – Provide component name.
     3. Use version – Provide version.
     4. Component process – Provide the DB\_Component Processs here.



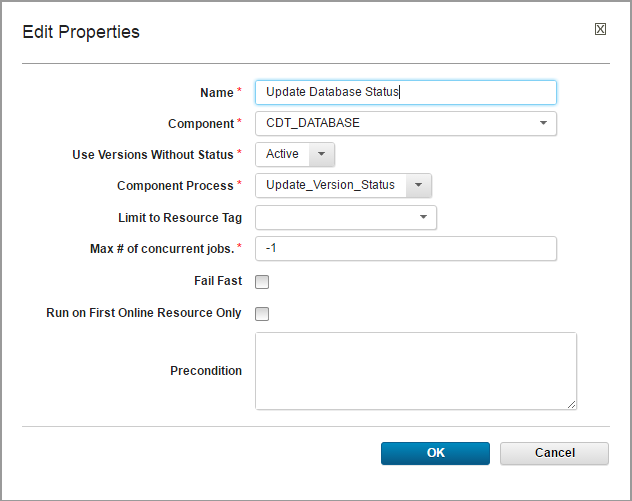
1. Install Components – Provide all the Components parallely or serially.



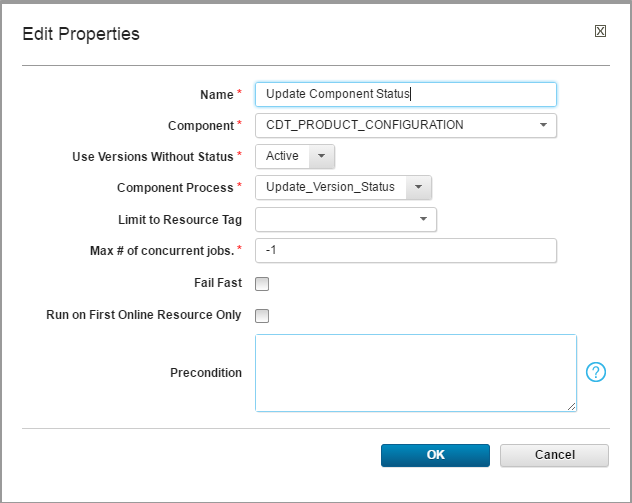
1. Approval request – Provide the Step which will ask for approval, to deploy the component in upper environment.



1. Update database status – This step will update the status of database component. Provide the “Update\_Version\_Status” component process.



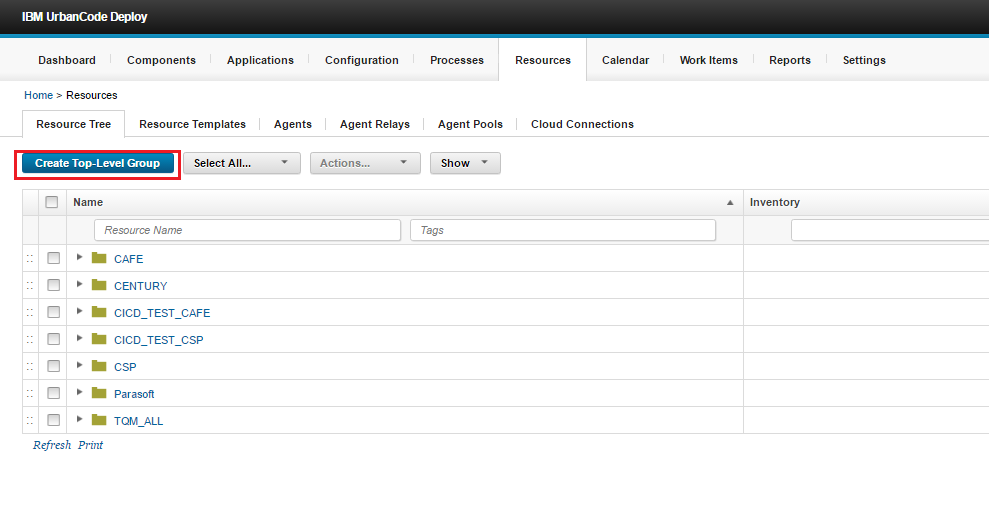
1. Update component Status - This step will update the status of component. Provide the “Update\_Version\_Status” component process.



# Resource

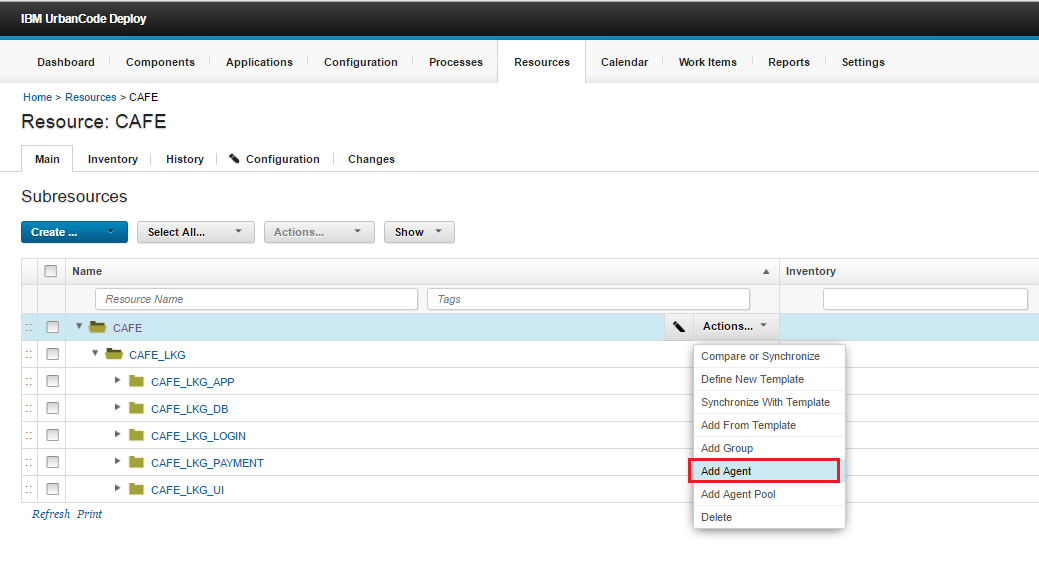
## **1. Create Resource**

Go to 🡪Resource 🡪 create top level group



Go to **Created resource** 🡪 click on **Action** 🡪 add **Agent**

Different agents are provided for different type of applications, map this agent according to that.



## **2. Add Resource to application**

Go to 🡪 **Application Name** 🡪 click on **Environment** 🡪 **Add base resource** 🡪 **Add resource**

