**Build Release Process**

**Strategy**

C:\Users\admin\Downloads\deployment (1).png

**Testing Environment**: EC2 Amazon Linux AMI (T2 Medium), PHP7.0(optional), MYSQL5.6, Apache 2.4.25(optional)

**Software Requirements** : Jenkins **2.46.1** (LTS), SonarQube 6.1

**Hardware Requirements**:

* Min. 4 GB RAM (2GB is alone required by SonarQube)
* The amount of disk space you need will depend on how much code you analyze with SonarQube. For more detail refer <https://docs.sonarqube.org/display/SONAR/Requirements>.
* Good Computing ablilty is required for sonarqube to run.

**Dependencies**:

* Jenkins requires Java7 or above to function. Java8 is recommended
* SonarQube analysis and the SonarQube Server require specific versions of the JVM. Java8 is recommended.

**Jenkins Plugins**: Git Plugin, [SonarQube Scanner for Jenkins](http://redirect.sonarsource.com/plugins/jenkins.html), Flexible Publish Plugin, S3 publisher plugin, Conditional BuildStep, Git Parameter plugin, Project based Matrix Authorization plugin

**SonarQube Plugins**: SonarPHPCode Analyzer for PHP

**Assumption**: Testing environment is set

**Steps for Installation**

**Java 8**

1. wget --header "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u102-b14/jdk-8u102-linux-x64.rpm

2. yum localinstall jdk-8u102-linux-x64.rpm

3. cd /usr/java/

4. java -version

5. set /etc/profile JAVA\_HOME=/usr/java/jdk

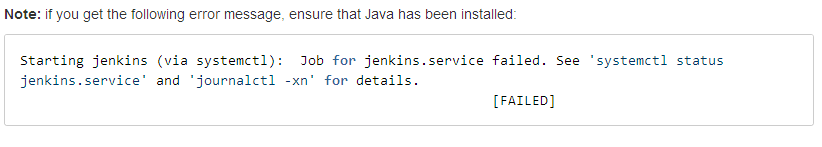
and JRE\_HOME=/usr/java/jdk/jre

6. source /etc/profile to apply changes

You can also refer this link: <https://www.mkyong.com/java/how-to-install-oracle-jdk-8-on-centos/>

**Jenkins 2.46.1**

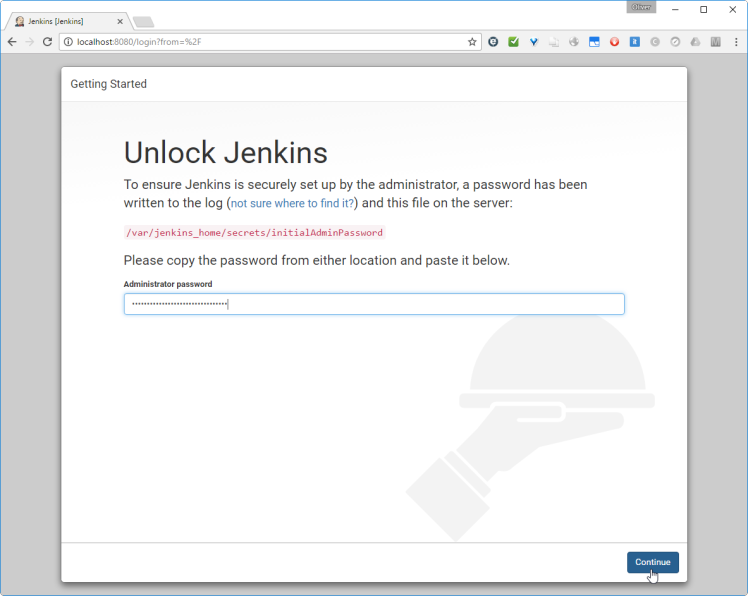
1. sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo
2. sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
3. sudo yum install jenkins
4. sudo service jenkins start/stop/restart
5. sudo chkconfig jenkins on



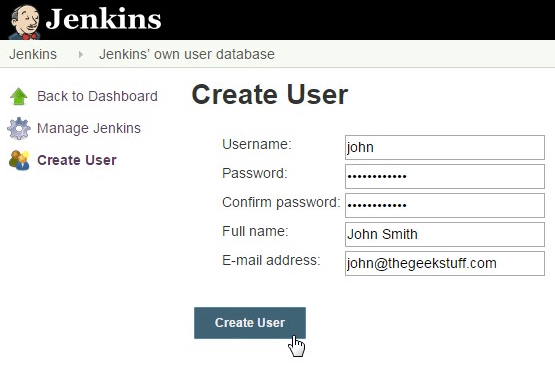
*Important points to note:*

* Jenkins will be launched as a daemon on startup. See **/etc/init.d/jenkins** for more details.
* The **'jenkins' user** is created to run this service. If you change this to a different user via the config file, you must change the owner of /var/log/jenkins, /var/lib/jenkins, and /var/cache/jenkins.
* Log file will be placed in **/var/log/jenkins/jenkins.log**. Check this file if you are troubleshooting Jenkins.
* **/etc/sysconfig/jenkins** will capture configuration parameters for the launch.
* By default, Jenkins listen **on port 8080**. Access this port with your browser to start configuration. Note that the built-in firewall may have to be opened to access this port from other computers. In AWS case security group act as firewall, so ensure that 8080 port is allowed at your instance.
* If in case you are not using AWS instance then to disable firewall refer link <https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions> .
* A Jenkins RPM repository is added in /etc/yum.repos.d/jenkins.repo

1. Now hit aws\_instance\_ip:8080 in your web browser. Then page appear will as



1. Now open file **/var/lib/jenkins/secrets/initialAdminPassword**, copy password from there and click continue.
2. Then jenkins will ask you to custom select plugins you want to install or install some selected plugins automatically. Custom is recommended as some time some plugin may override other’s effect, but we have used preinstalled and found no problem in doing any task mentioned in this document.
3. Then create user screen will prompt as



The user created here will be admin user. Fields are self explainatory.

1. Then jenkins refresh itself and ask for login as admin user.

**Sonarqube 6.1**

**Assumed**

Java8 is installed, MYSQL5.6 or MYSQL5.7 is installed.

For more database options refer link <https://docs.sonarqube.org/display/SONAR/Requirements>

**Installation Steps**

1. There are 2 things required sonarqube server and sonarqube runner(or scanner). Sonarqube runner or scanner is responsible for executing code analysis. We will not install sonarqube runner seperatly as Jenkins automatically integerated sonarqube scanner with its server(we will see later how it will be done).
2. For installing sonarqube server, download zip file from link <https://sonarsource.bintray.com/Distribution/sonarqube/sonarqube-5.6.6.zip>

to see stable version at your time refer link <https://www.sonarqube.org/downloads/>

1. Then extract that .zip file to /opt
2. Sonarqube saves results in its database as such it require database user and database for its working. Make it in mysql as

Login to mysql, then

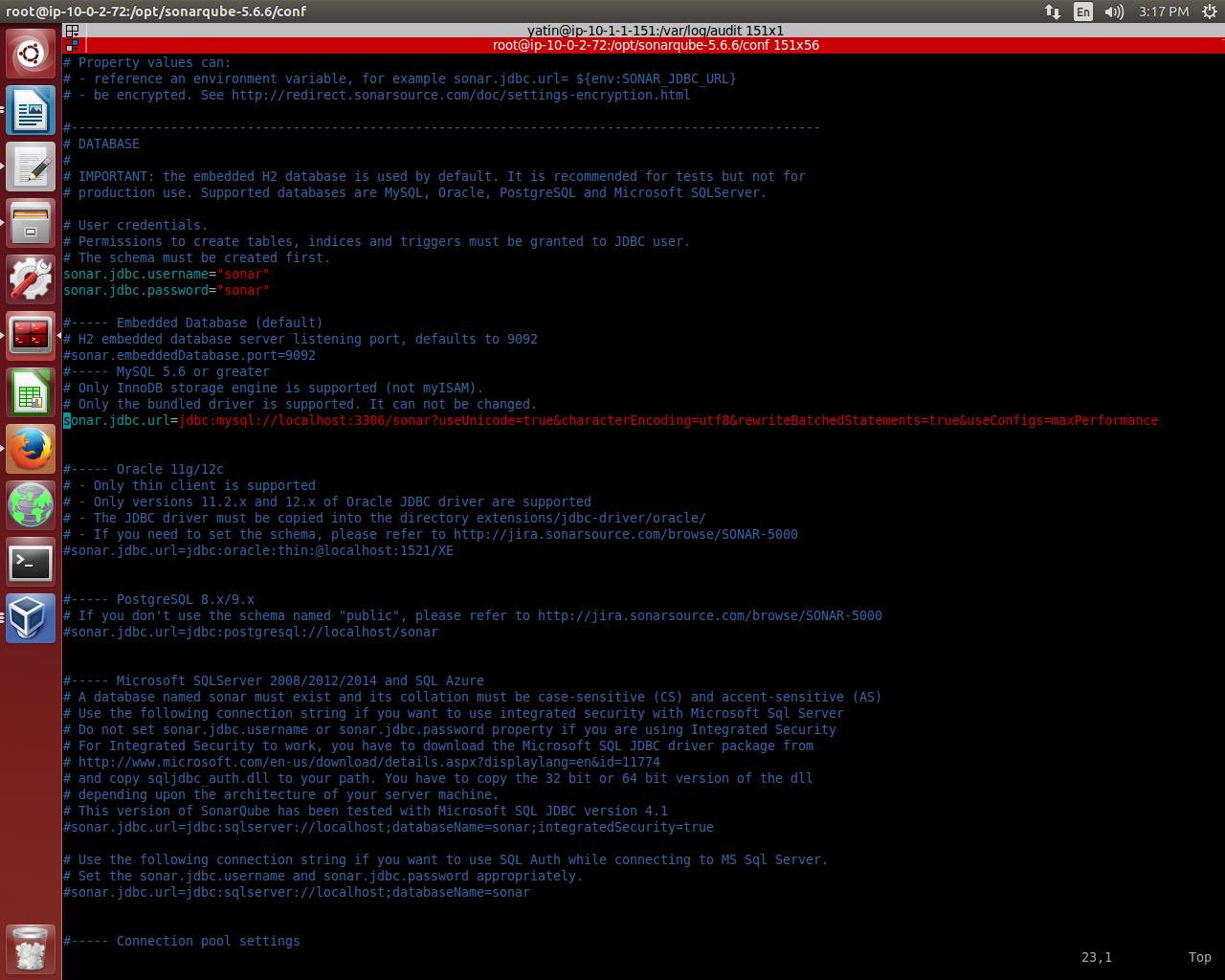
CREATE DATABASE sonar charset=utf8mb4; // charset set mandatory

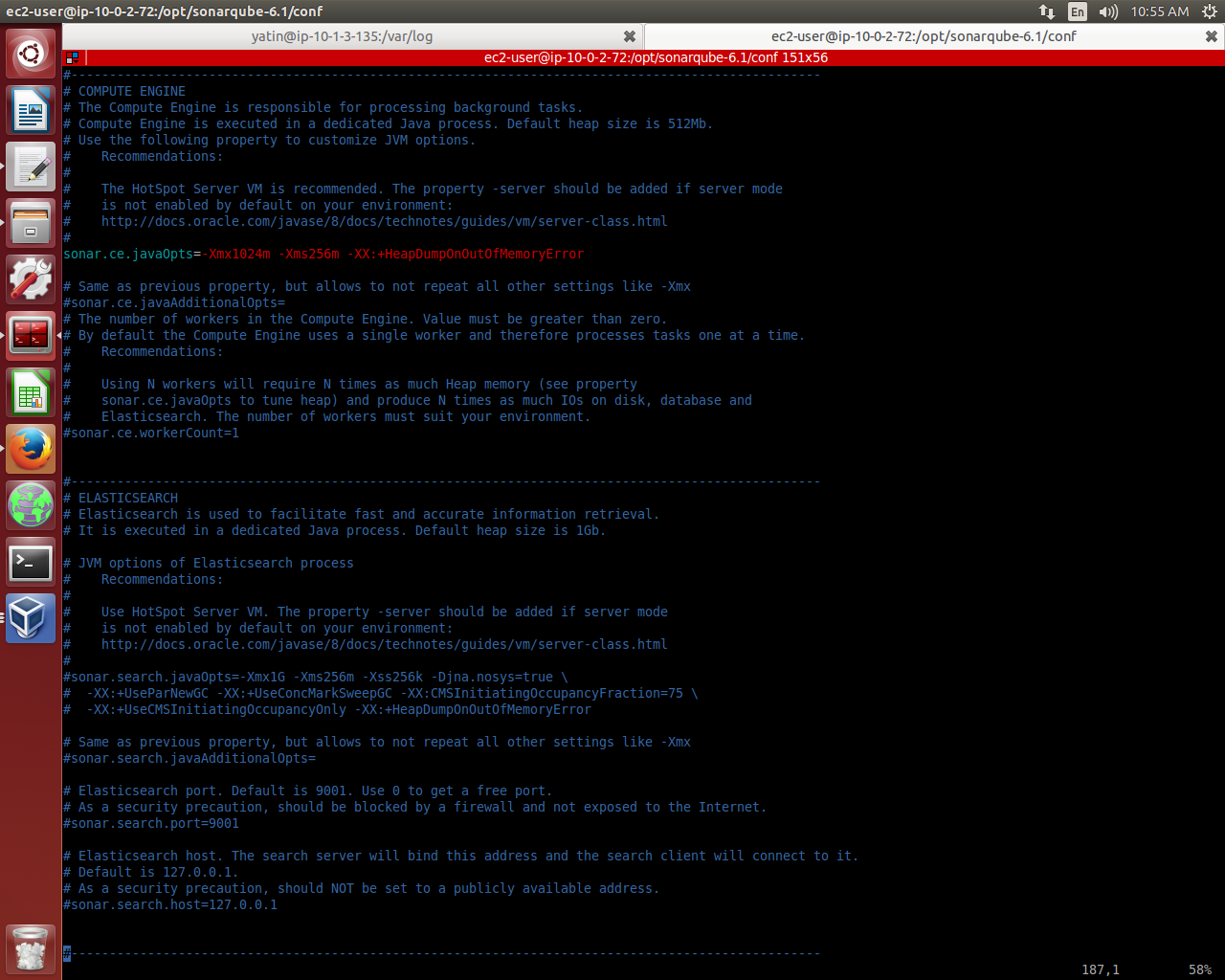
CREATE USER ‘sonar’ IDENTIFIED BY ‘sonar’;

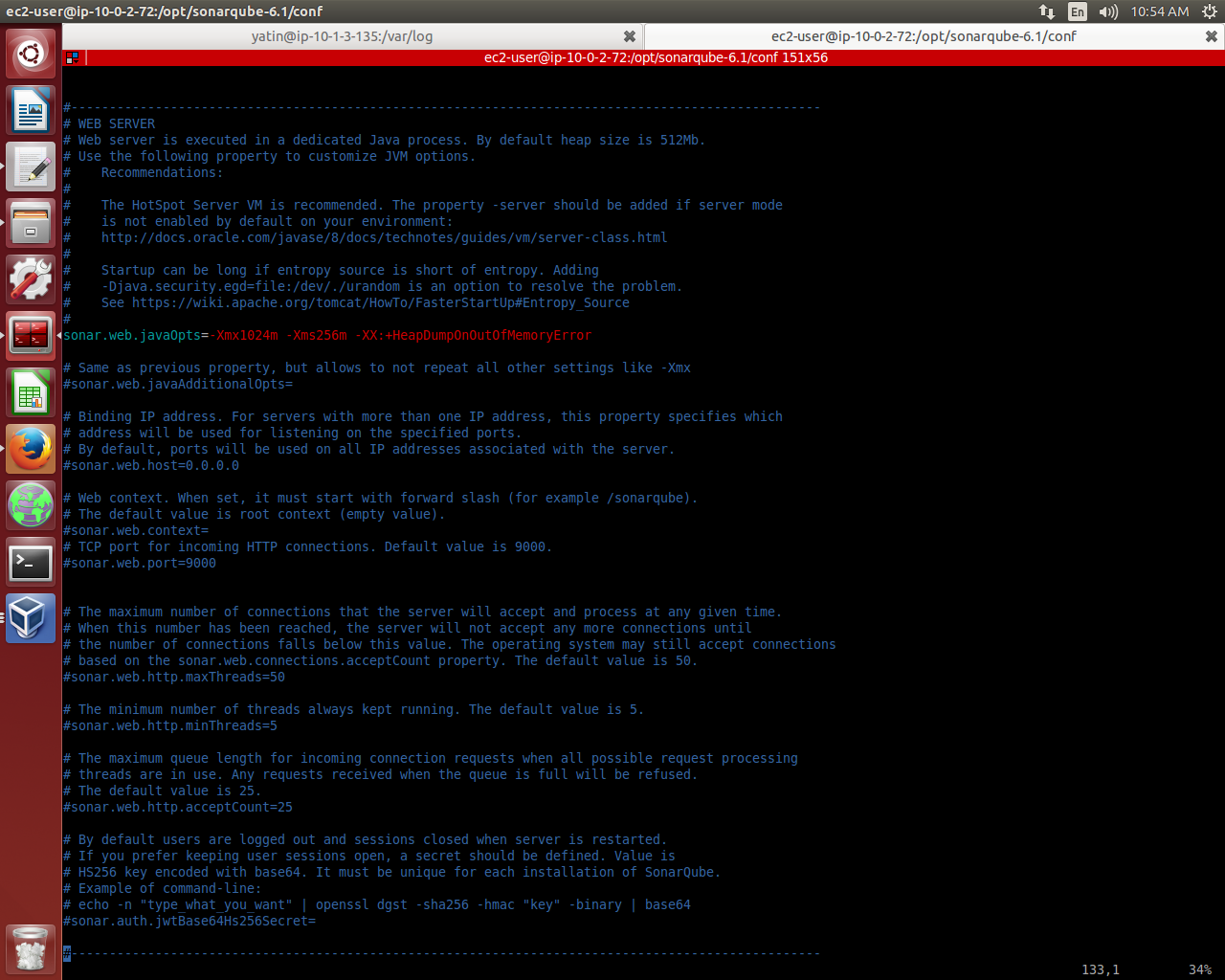
GRANT ALL ON sonar.\* TO ‘sonar’@’%’ IDENTIFIED BY ‘sonar’;

GRANT ALL ON sonar.\* TO ‘sonar’@’localhost’ IDENTIFIED BY ‘sonar’;

Now logout of mysql. Now go to /opt/sonarqube-5.6.6/conf and edit sonar.properties and uncomment sonar.jdbc.username and sonar.jdbc.password, and provide sonar username and its password in it. (In our case it is sonar and sonar) . It will be seen as







1. Set sonar environment variable as



Shown in **PATH** variable of snapshot. This change is done in **/etc/profile**. By changing in this file change will get automatically reflected to every user. Also note JENKINS and JAVA environment variables are also set. To change it only for current user, in home directory there is .bash\_profile, set it there.

Then to apply changes in /etc/profile use command

source /etc/profile

now you can check it by echo $PATH

1. Now go to /opt/sonarqube/bin/ and their according to os you are using select folder. As we are using **64 bit** t2 medium Amazon **Linux** so we will go for linux-x86-64 and from there start sonarqube as

./sonar.sh start

To stop use

./sonar.sh stop

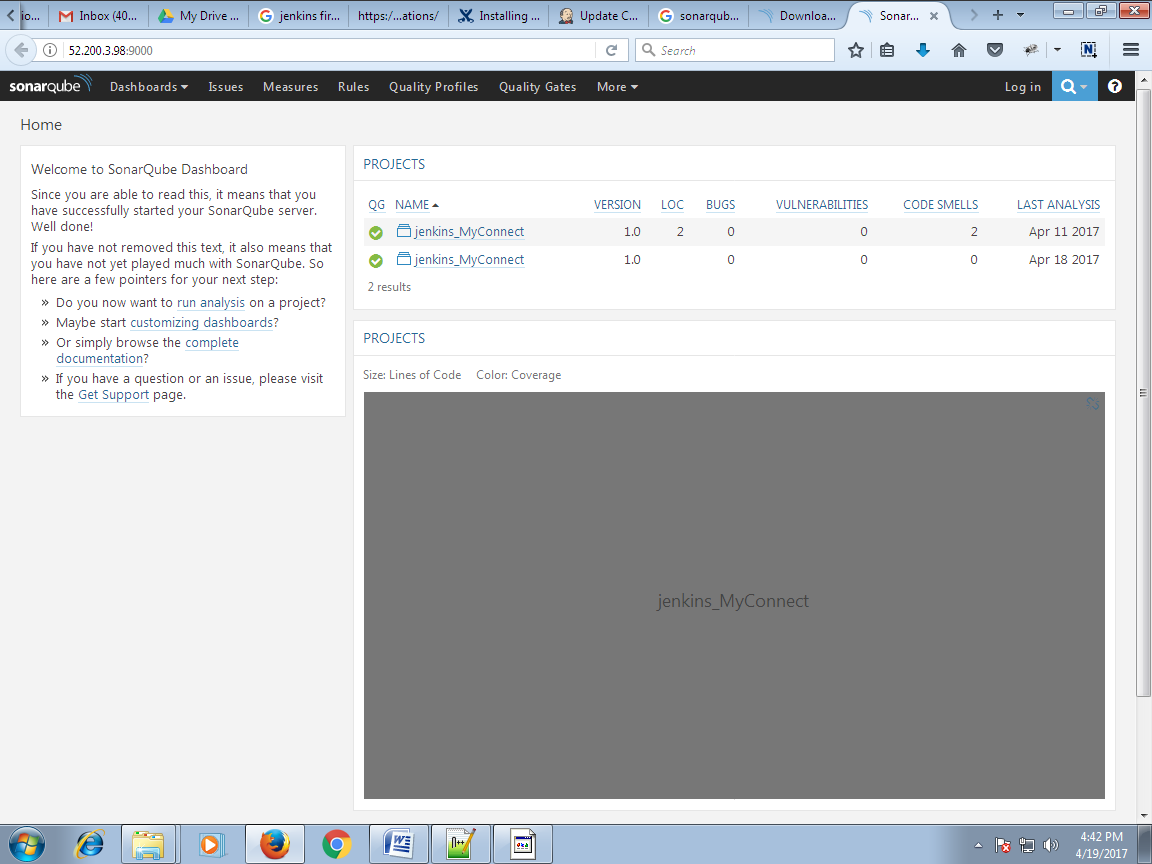
And to restart use

./sonar.sh restart

1. Sonarqube by default runs on **9000 port**. Ensure that your firewall(in aws security group) don’t block request at 9000 port. Type url

aws\_ip:9000

and hit. Sonarqube page will appear as



You can login by credentials “admin” and “admin” and come into administrative mode.

**Note**: As you can see in snap above, LOC is line of code, Vulnerability is some hacking points in code(like if password is hard coded), Code Smells is coding standard error, there is one more term technical debt i.e time it should take to eliminate those code smells.

**Also you need to set mysql size according to your project size**. To set it set /etc/my.cnf (mysql configuration file) as



In it notice “max\_allowed\_packet” and “innodb\_log\_file”. Set it 256M and restart mysql. To check it login to mysql and run

SELECT @@max\_allowed\_packet;

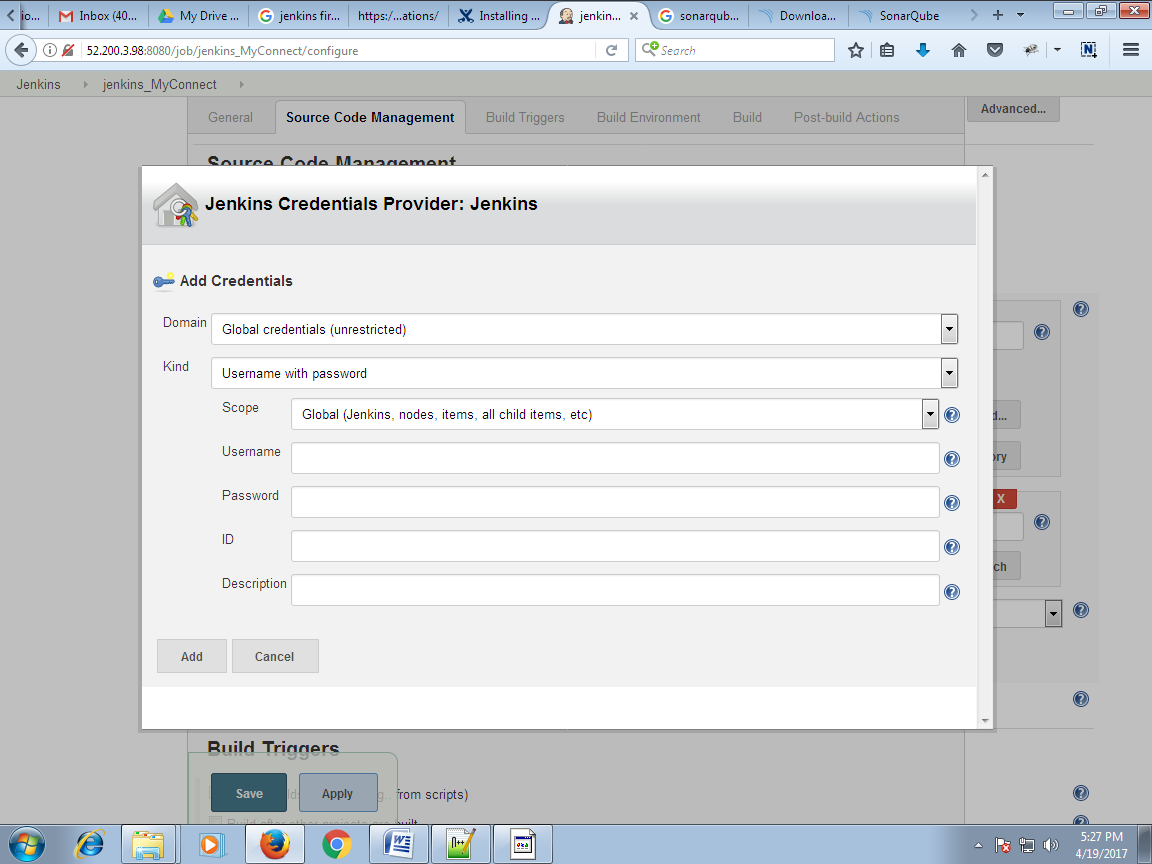
**Assumptions**

* You have a github account where your php project that is to deployed lies.
* You know “how to make jobs in Jenkins”
* You know “how to install plugins in jenkins”
* You know “how to work with s3 buckets, and have knowledge of codepipeline and elasticbeanstalk”
* ***First we introduce way to install different plugins and then explain how everything is going in operation.(IMPORTANT)***

**Configuring Git with Jenkins Jobs**

**Purpose**: We are configuring git with Jenkins, as we want that Jenkins should automatically carry code from git repository to its workspace and make its build.

1. To configure, you require “**Git Plugin**”. Install it. It may also come preinstalled. To check it, type its name in filter of manage plugin section under “Installed” tab. If plugin appears, means it is installed. You can check other plugins in same way.
2. Now when you make new freestyle job or configure job, and click on “Source Code Management” tab, you will see extra option Git there(which was not there before plugin installed). Now Here you can configure project at github or if you have in codecommit, then that also.
3. Now mention repository url as <https://github.com/username/project.git> and to provide credentials, click on Add>Jenkins and pop up appear as



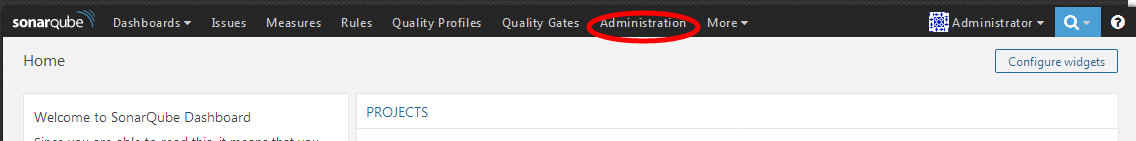
In it username is git username, password is git account password, id from which credentials will be uniquely identified. Description can be anything about credential(something reasonable and don’t leave it blank).

If in case you want to give keys(in case of codecommit) then you can select keys from type of credentials(like here it is username with password).

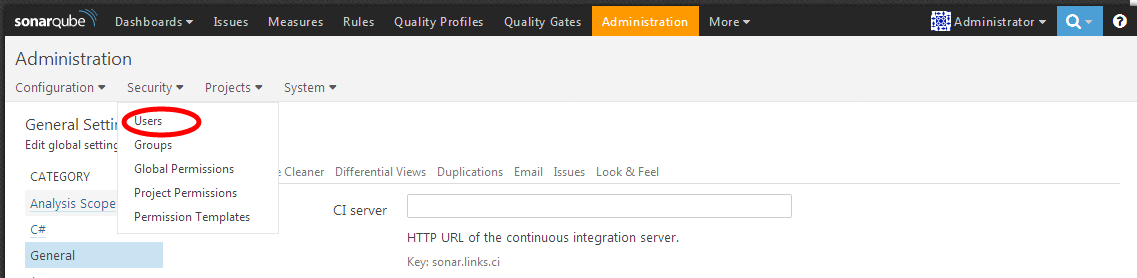
1. In Branch mention branch from which build will be prepared.
2. Apply, Save and build job, then you can check in /var/lib/jenkins/jobName/, there all code from git should carried as job is build.

**Configuring SonarQube with Jenkins**

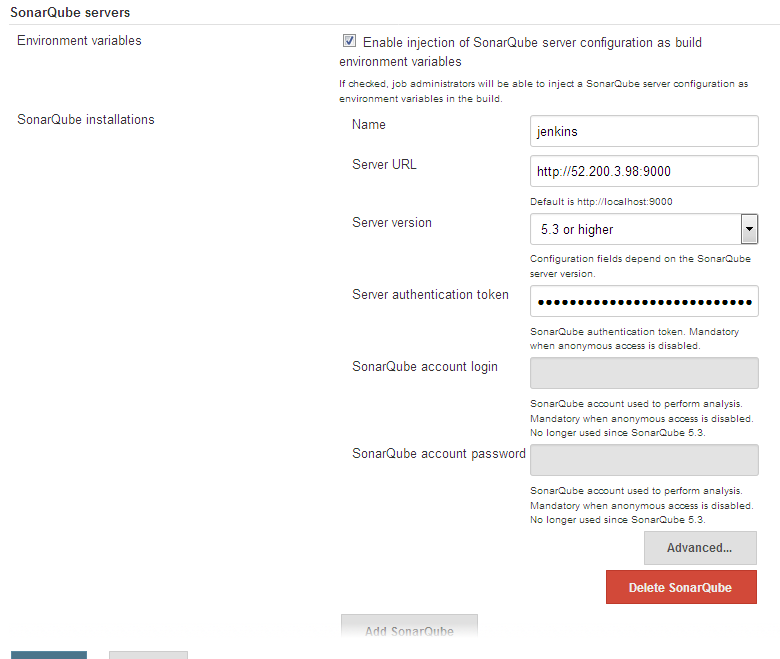
1. To configure, you require “[SonarQube Scanner for Jenkins](http://redirect.sonarsource.com/plugins/jenkins.html)”. Install it.
2. Now open SonarQube page(Ensure sonarqube service is running, and port 9000 is allowed by your firewall), Login by credentials “admin” and “admin”, and select Administration option as



1. Select Security>Users as



1. Now click on create user button. From here you can generate **authorization token** . Save this token as it will be used to configure Jenkins with sonarqube.
2. Now open Jenkins, Go to Manage Jenkins->Configure System and there you will see a SonarQube section(only visible after installation of plugin). Click on Add SonarQube and it will be seen as



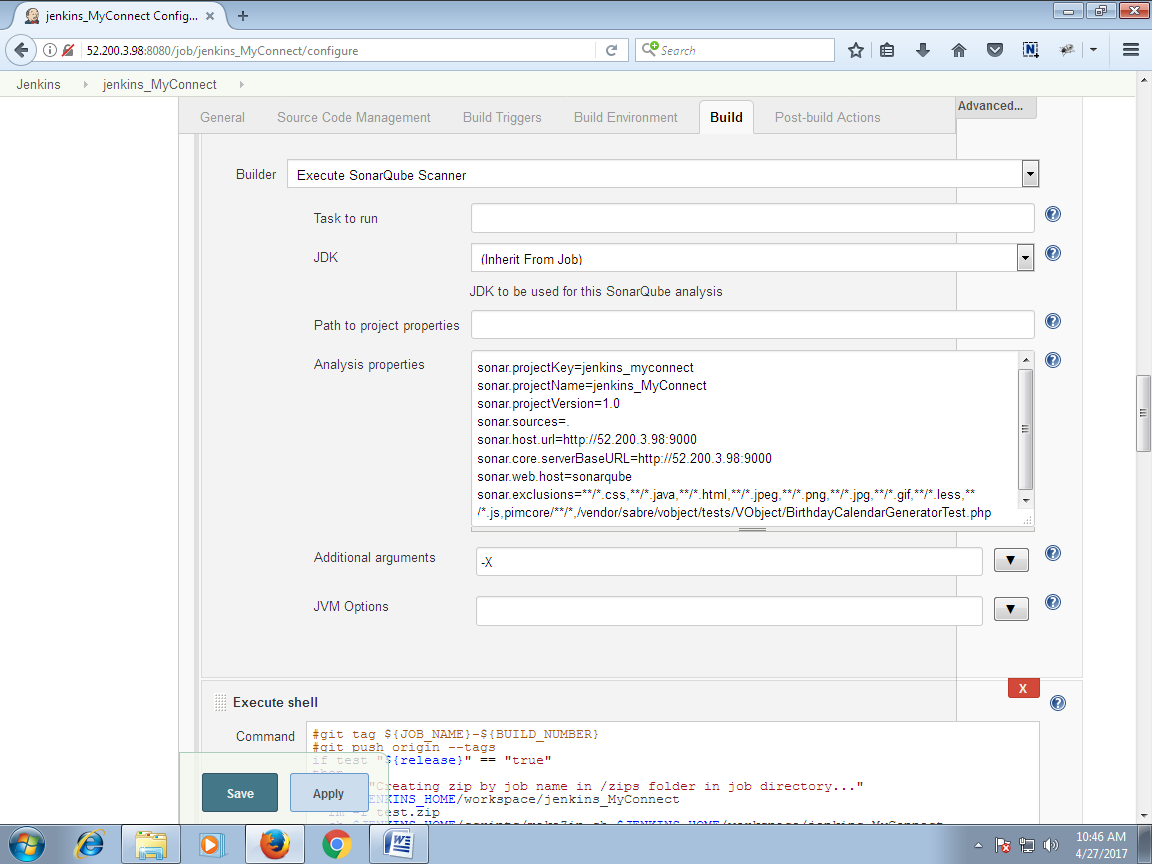
Point to note:

* Don’t miss http from url

As SonarQube version is 5.6.6 thus select 5.3 or higher

Then provide authentication token previously generated and provide in it. Apply and save it.

1. Now when job is made and want to run code quality analysis with it, click configure job and select Build, in which there is a option Execute with Sonarqube Scanner. Select it as configure it as



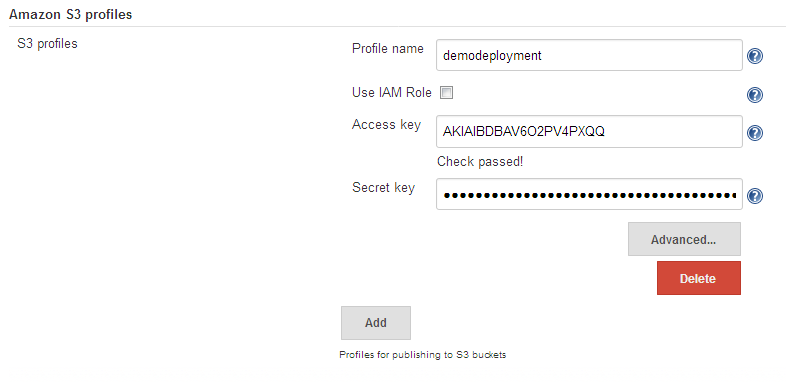
Here “sonar.exclusion” will exclude these files. For see more exclusion rules refer url “https://docs.sonarqube.org/display/SONAR/Narrowing+the+Focus”

(Options will be same as snapshot is showing, UI can be different). In Analysis properties sonar properties are set which need to be set before code analysis can be carried by sonar.

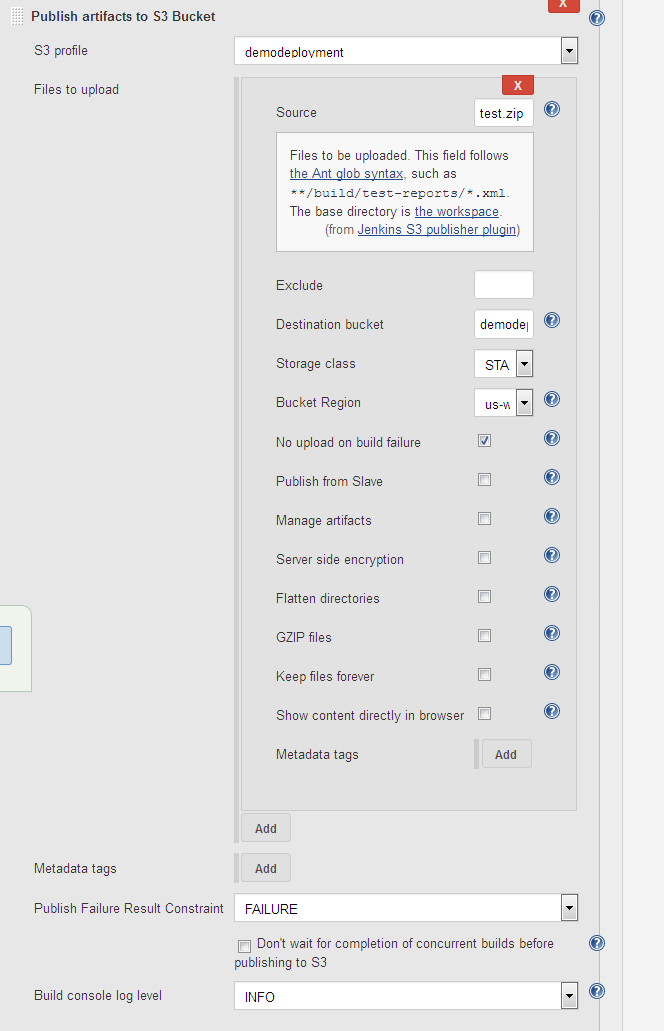
**Configure S3 with Jenkins**

**Purpose**: We want to push zip of build to s3 bucket.

1. For it you need “S3 Publisher plugin”.
2. For it you must have IAM Access Key and Secret Key and must be ready with bucket where zip need to be pushed. In our case bucket name is “demodeployment”.
3. Now go to Manage Jenkins>Configure System, there you will see Amazon s3 Profiles option as

here profile name is bucket name. You can add as many buckets correspond to as many users you want. Your bucket is not required to keep public files as all authorization you are specifying here. Now apply and save.

1. Now you can perform post build actions on s3 bucket.



UI may seem different as it is after flexible publish plugin installed. If you want you can first install flexible publish plugin and then do s3 configuration.

In **Source** we specify filename that is need to be pushed in s3 bucket. Filename cannot be path. Here we provide “test.zip” because it lies in Jenkins job workspace.

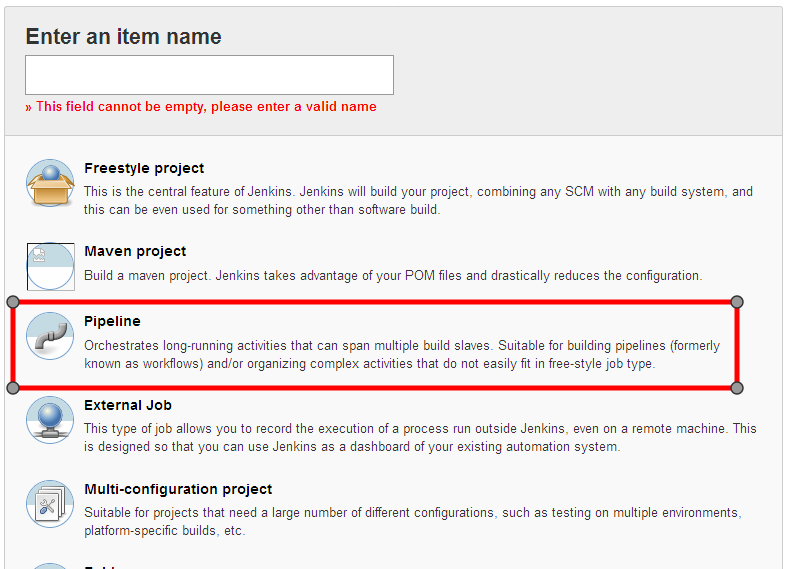
Everything else is self explanatory. Click apply and save.

**Pipeline in Jenkins**

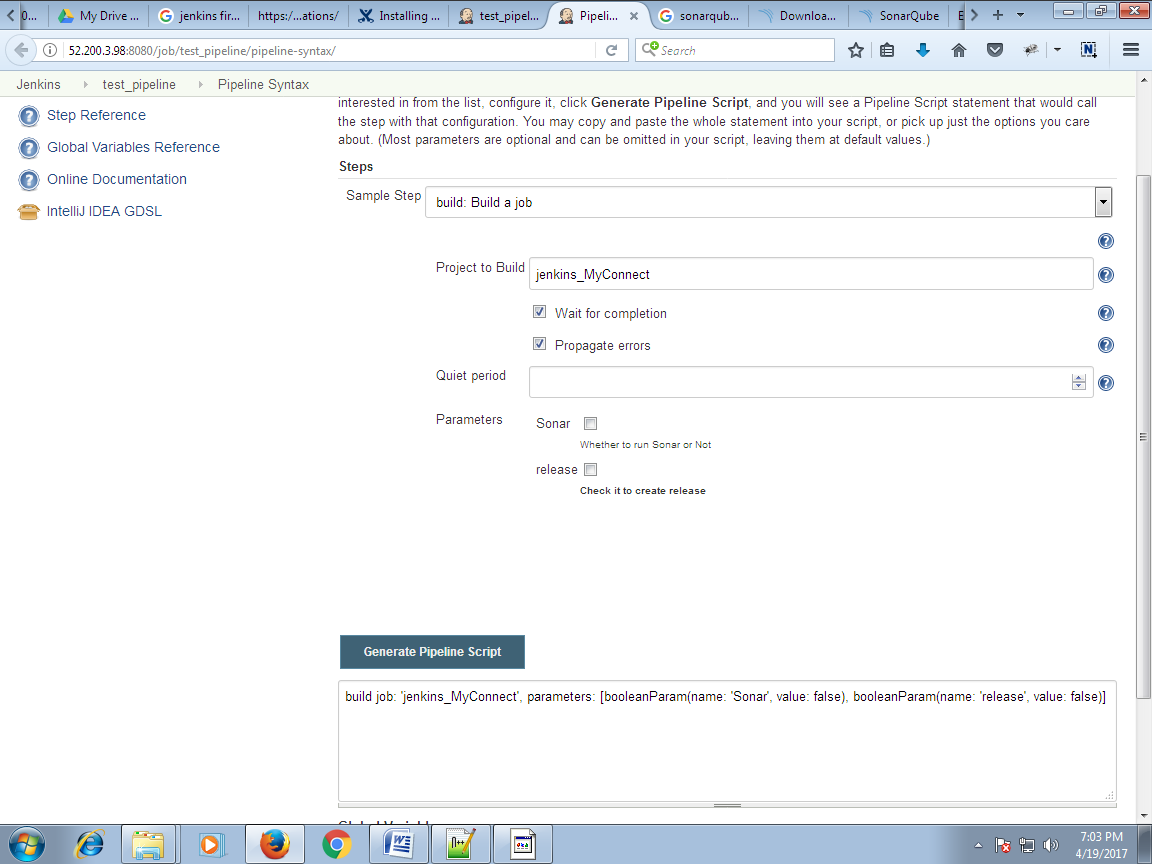
**Purpose**: Purpose of pipeline is to divide whole deployment process into different stages and able to track each stage independently. We have used pipeline in a way that we can put conditions on build and type of job to run, and then pipeline do everything for us.

Note: “Pipeline” is ultimately a “job”.

1. Install “Pipeline” plugin in Jenkins. Now when you create job a Pipeline option also appears(which was not available before plugin is not installed).



1. In pipeline that we made, there is only stage i.e Build now.
2. In Jenkins pipeline to divide deployment into stages and do different task in stages we have to write a pipeline script. This pipeline script is in groovy and Jenkins provide a sample groovy script also. Moreover we can use “Pipeline Syntax” link to generate groovy code for things we want to do in each stage. Like



Now copy this code in your groovy script.

**Conditional BuildStep**

**Purpose**: A buildstep wrapping any number of other buildsteps, controlling their execution based on a defined condition (e.g. BuildParameter).

1. Install this plugin, there is nothing need to configure with this plugin.

**Flexible Publish plugin in Jenkins**

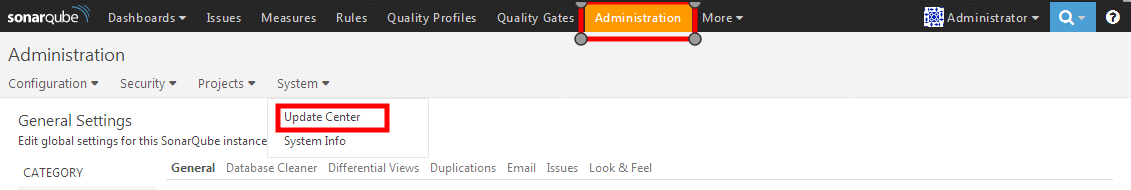
**Purpose**: This plugin is required to execute conditionally post build actions.

1. Install this plugin, there is nothing need to configure with this plugin.

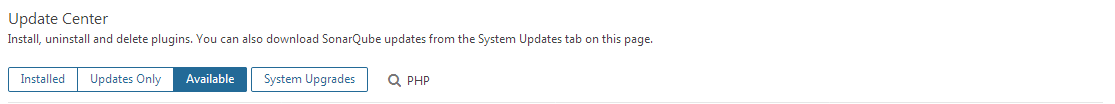
**SonarPHPCode Analyzer for PHP**

Purpose: SonarQube by default don’t support PHP code quality analysis, to provide it PHP support you need this plugin in SonarQube.

1. Login to SonarQube by credentials “admin” and “admin”. By it you will get into Administrative mode.



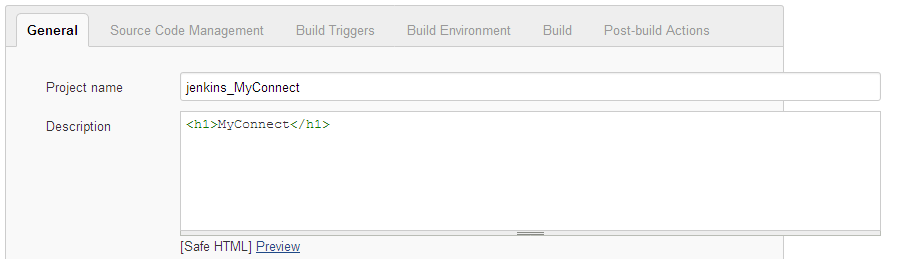
1. Go to Administration>System>Update Center and then click on Available tab, type PHP in filter(or search) and this plugin will appear, install it from there, and restart sonarqube.



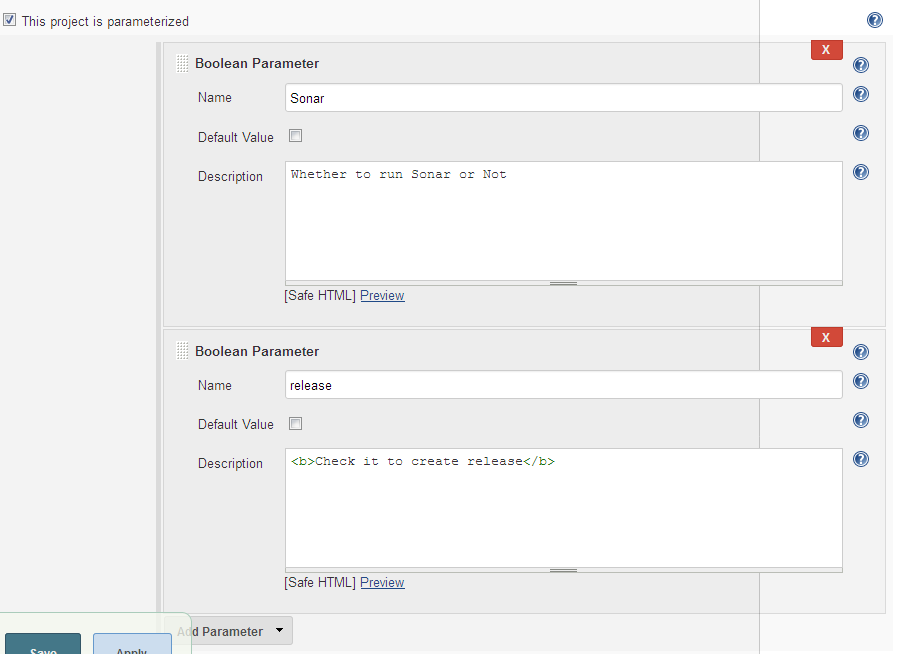
If in case it is already installed, it will not appear in Available tab but will appear in Installed tab.

Starting Operation

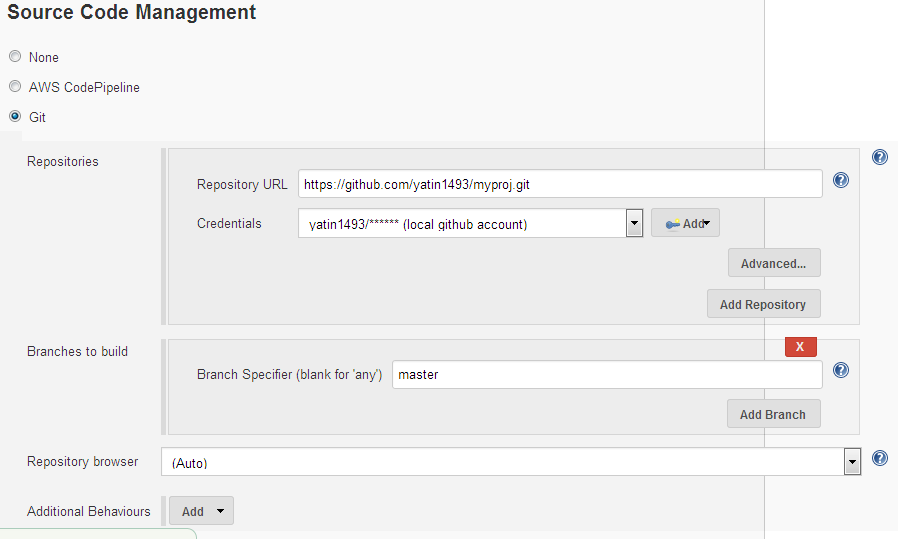
1. There will be a job correspond to your project. In our case it is “Jenkins\_MyConnect” . The way it is configured is shown in snapshots:



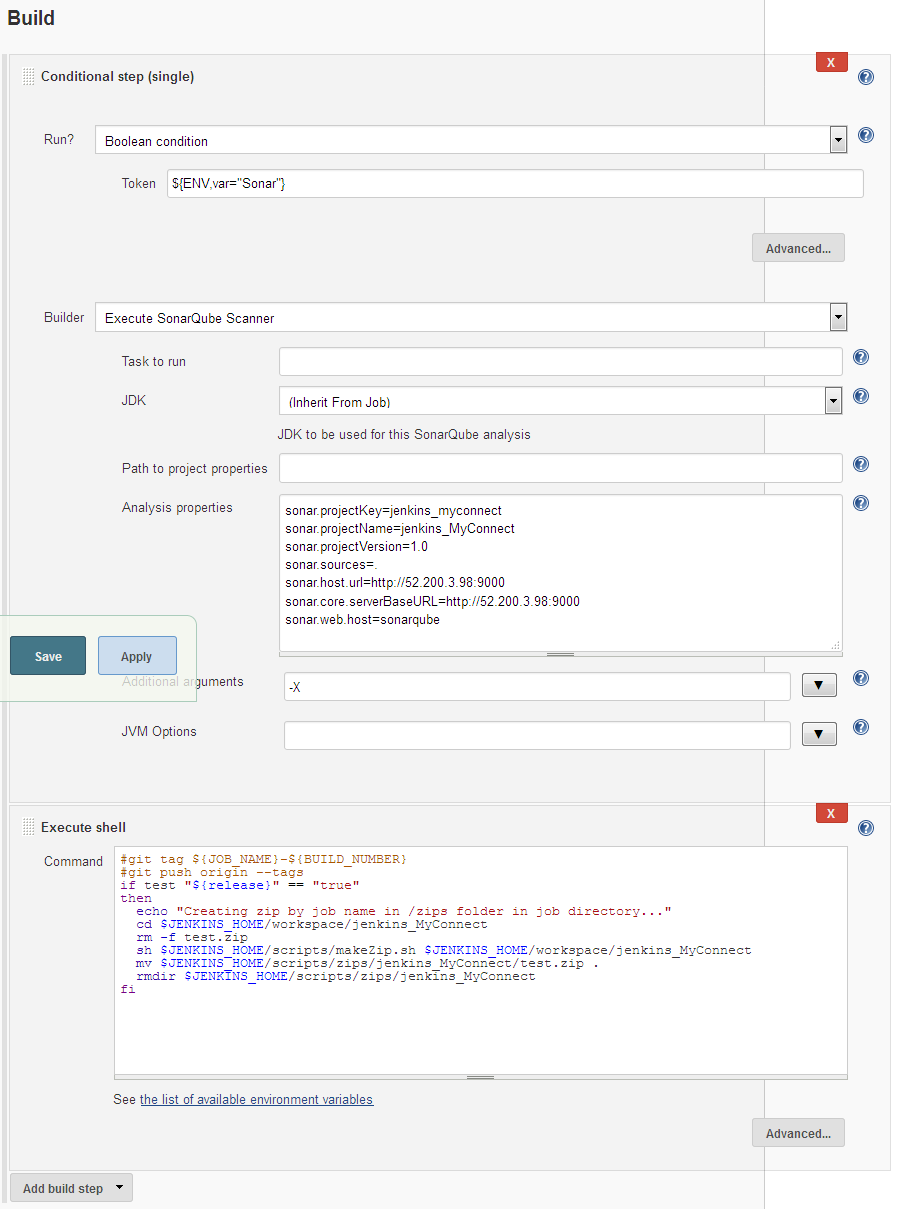
Here description must be meaningful, as it will appear to user when he will open job. By default it can be in plain text, but we have enabled option of HTML formatting by selecting “Safe HTML” in Markup formatter field in “Configure Global Security”.



We have specified parameters Sonar and release which need to be selected before build job. These are “Boolean parameters”. These parameter can be used as ${ENV,var="Sonar"} to get their value. Their value will be either true or false.



Here we specify git credentials as explained in “Configure Git with Jenkins” section.



Marked step specify conditional build which feature come by conditional plugin. We specify Boolean parameter here that check for value in “Token”. Value to token comes from variable ${ENV,var=”Sonar”} . If this value is true, build action that this conditional step specify execute else not.

Then we select execute with SonarQube Scanner(this option again comes from SonarQube plugin. It will work only when you had configured sonarqube with jenkins).

Then we are executing set of commands in shell. This command is checking that if release variable set to true then previous zip is deleted and new zip of code inside Jenkins workspace is made using makeZip script. This zip temporarily added to /var/lib/jenkins/scripts/zips, then copy from there to Jenkins workspace. (This will ensure that there will always be at max one zip present of whole code in Jenkins workspace).

Code to makeZip.sh is

#$1 consist of folder name with its complete path or just folder name if in case script execute in same dir where folder exists

filename=$(basename $1) # get folder name from folder path

jenkinshomedir="/var/lib/jenkins"

zipLocation="${jenkinshomedir}/scripts/zips"

if test -d "$1" # check if folder exists

then

cd $zipLocation # go to zip location

if ! test -d "${filename}" # check if in current job dir project zip folder exists

then

mkdir "$filename" # make empty project zip folder

fi

cd "${zipLocation}/${filename}" # go inside location where to form zip

if test -f "test.zip" # if test.zip exists

then

rm "test.zip" # remove previous zip

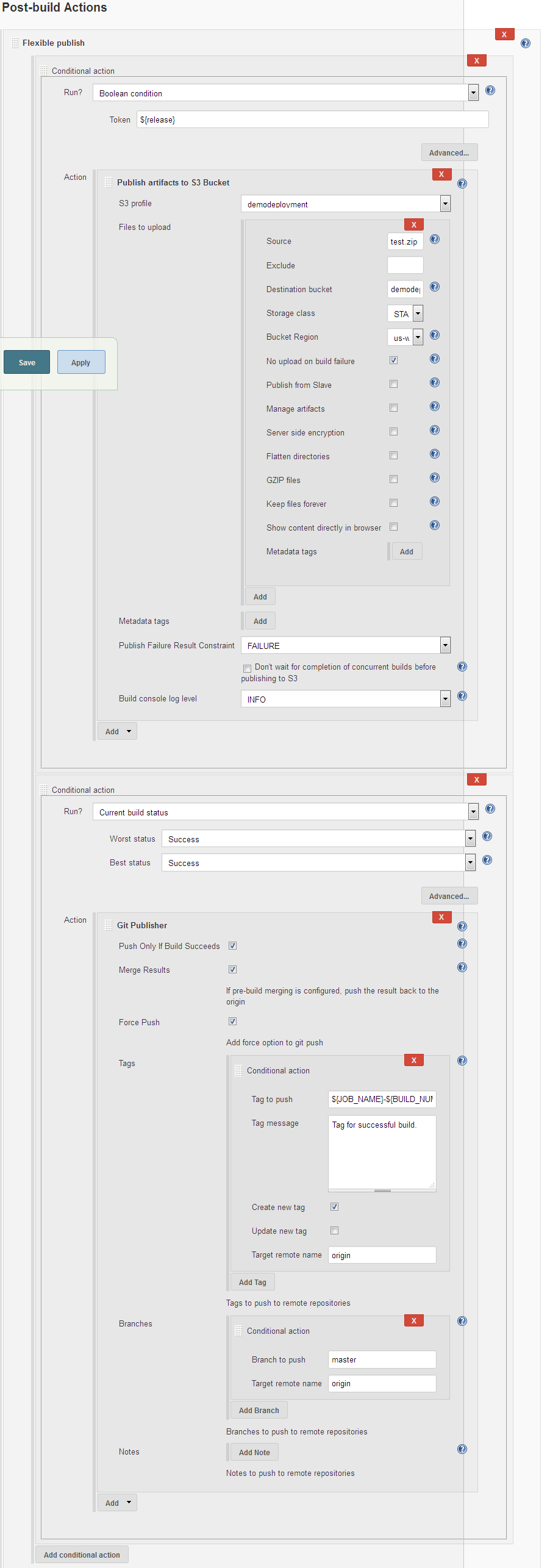
fi

export zipsSaveDir=`pwd` # get complete path to zips project folder

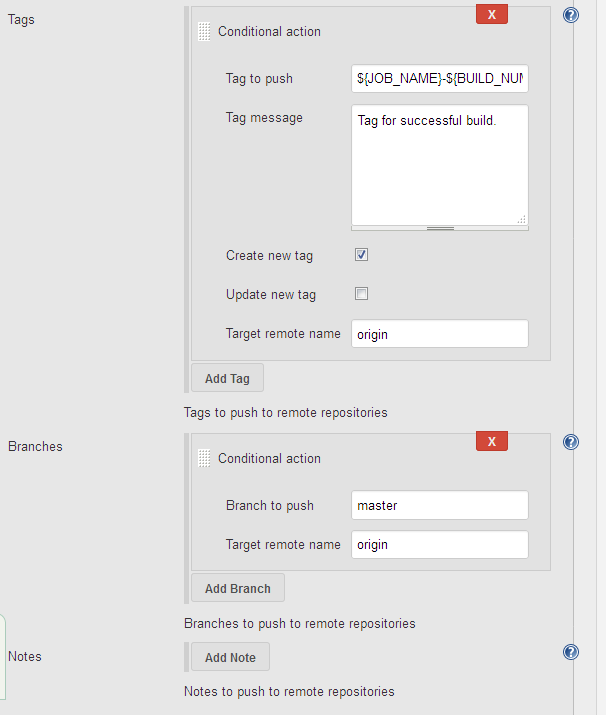
cd `dirname "${1}"` # move to parent directory of folder of which zip to make

zip "${zipsSaveDir}/test.zip" -r $filename -x \*.git\* -x .elasticbeanstalk -x .sonar # make zip of folder in zips folder excluding .git file from folder

fi



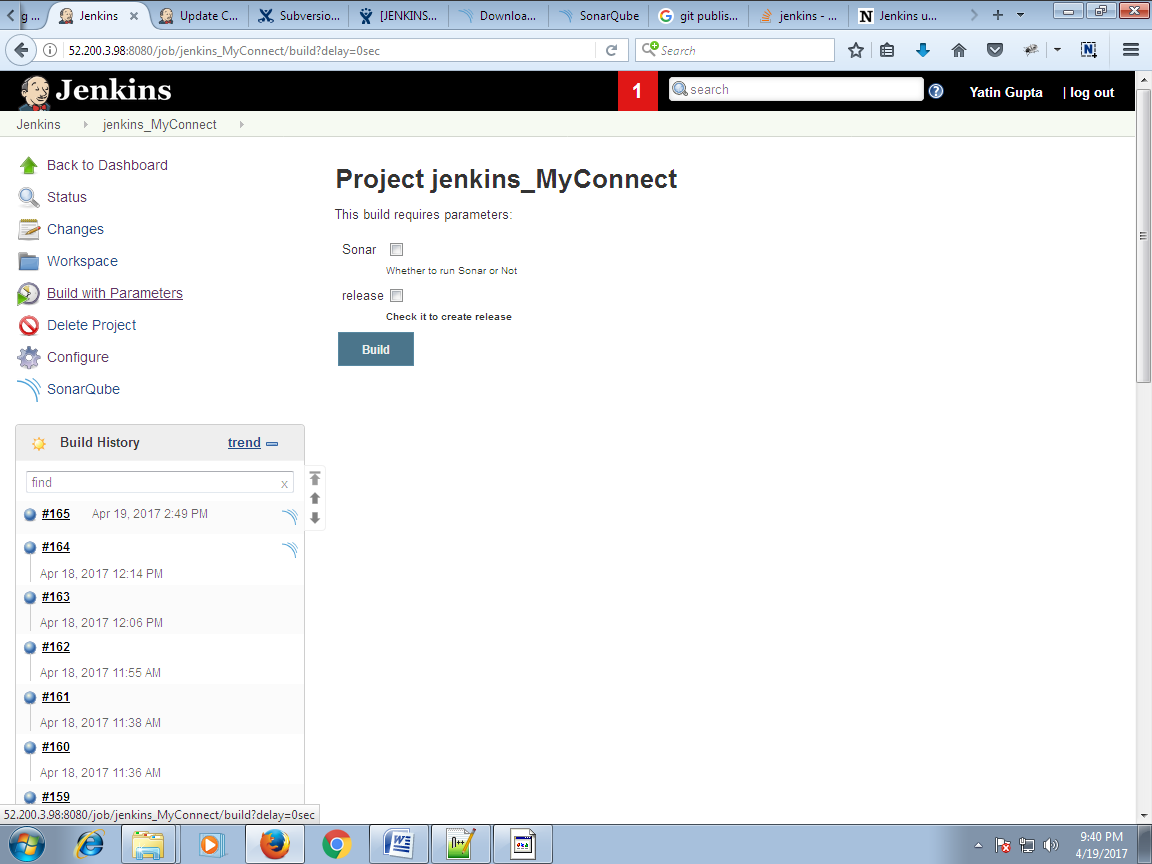
In Action select Git Publish



* **Green** marked comes from “Flexible Publish Plugin”
* **Red** marked comes from “S3 Publisher Plugin”. In S3 Published plugin we mention “profile” as bucket name and in “Destination Bucket” field, if your bucket destination is as s3://demodeployment/myconnect then field value will be demodeployment/myconnect and other is explained in “Configure S3 with Jenkins”.
* **Black** marked comes from Git Publisher plugin which comes from git plugin. It is responsible for pushing tag on git.
* Tag will be pushed by name taken by built in environment variable JOB\_NAME and BUILD\_NUMBER.
* “Branch” is about branch at which tag will be pushed.

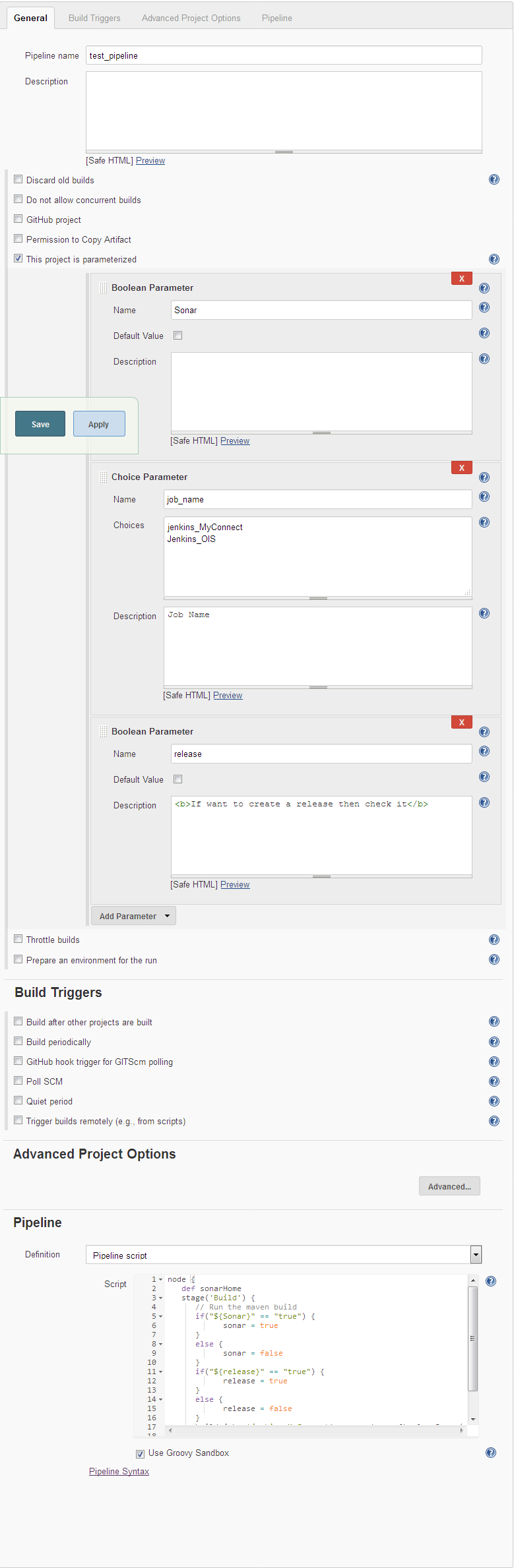
Now click Apply and Save and click on “**Build with Parameter**”.

A screen appear as



* Now you can select ***optionally*** Sonar or release and prepare build. If Sonar is selected then code quality analysis is carried and if release is selected then build zip is carried to s3 bucket destination as specified.

1. Now we need to develop pipeline. Go to New Item>Pipeline and mention project name and then select OK. Pipeline is configured this way:



Parameters are made here in same way as in freestyle job, but to be used in pipeline script. Groovy that we write is

node {

stage('Build') {

// Run the maven build

if("${Sonar}" == "true") {

sonar = true

}

else {

sonar = false

}

if("${release}" == "true") {

release = true

}

else {

release = false

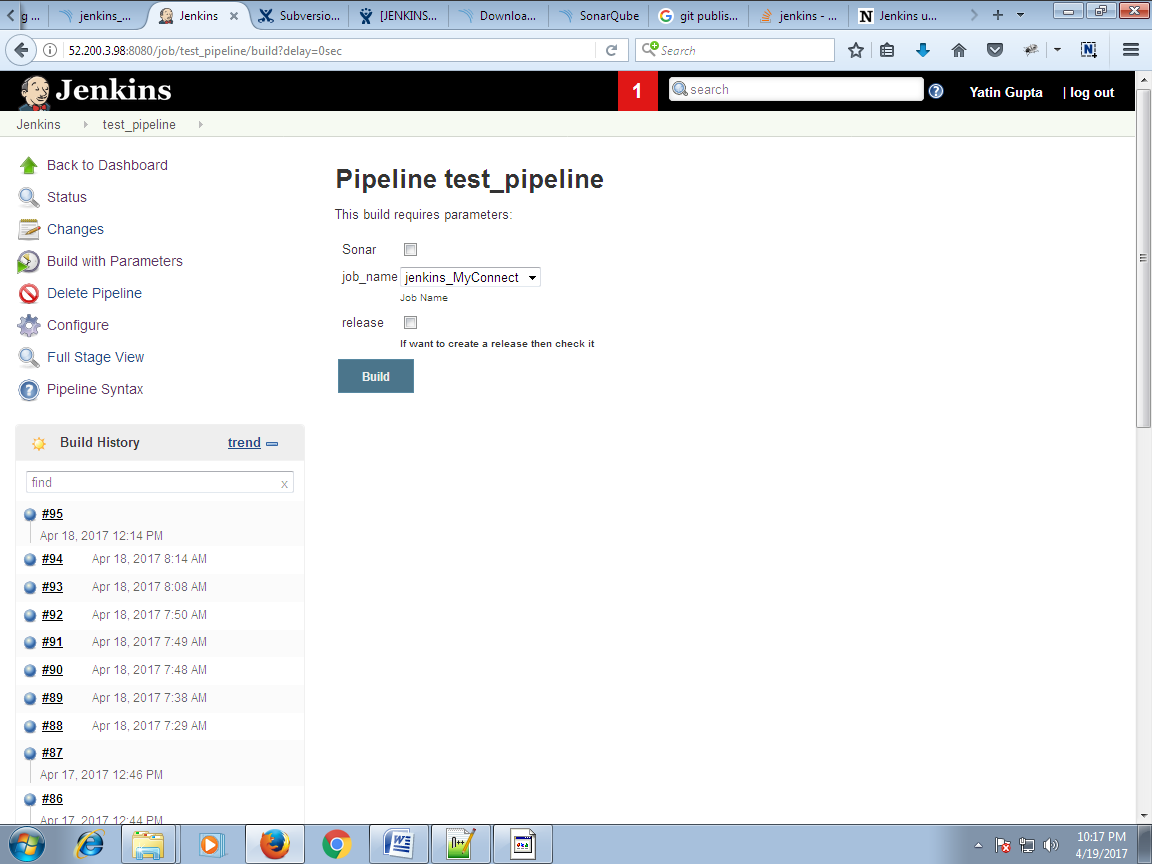
}

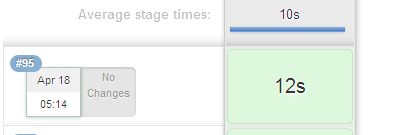
build job: 'jenkins\_MyConnect', parameters: [booleanParam(name: 'Sonar', value: sonar), booleanParam(name: 'release', value: release)]

}

}

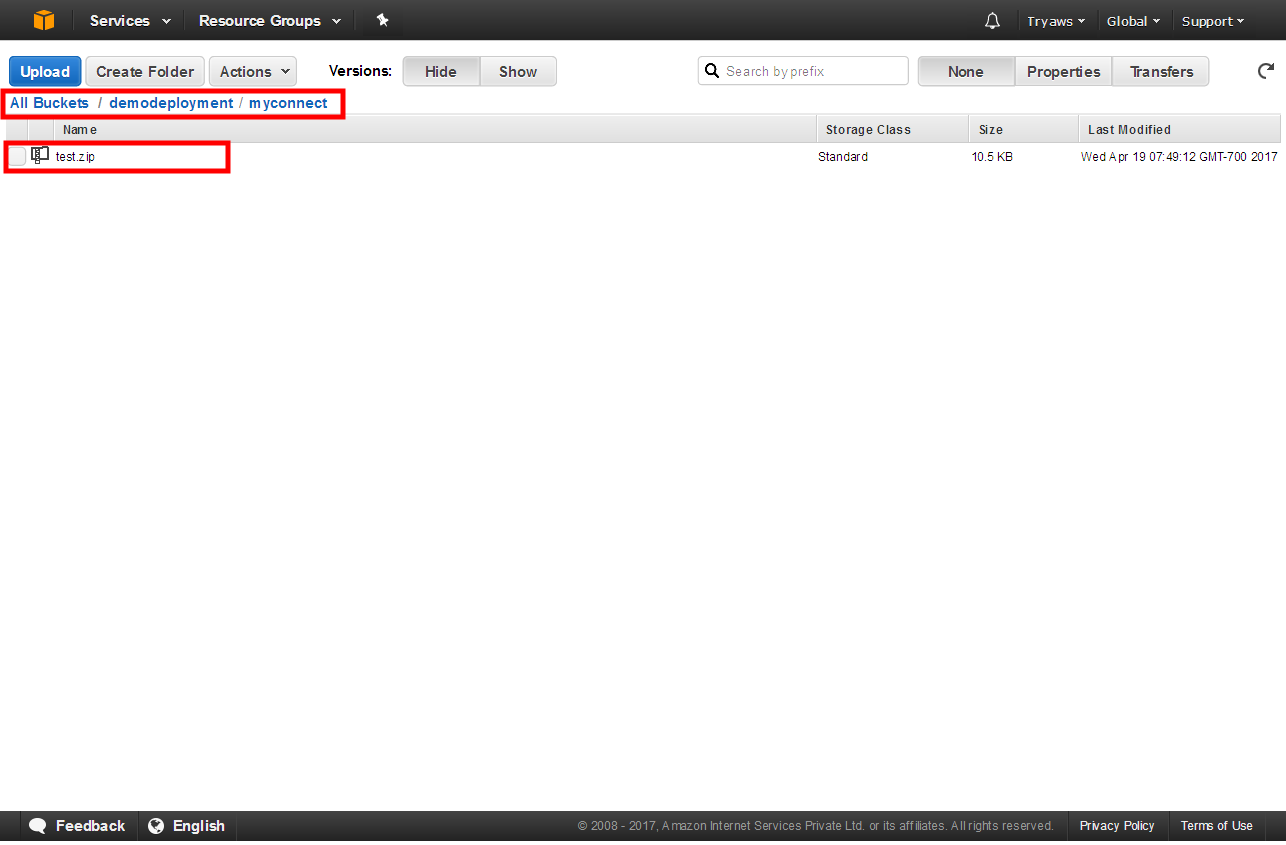
Now build with parameter, screen come as





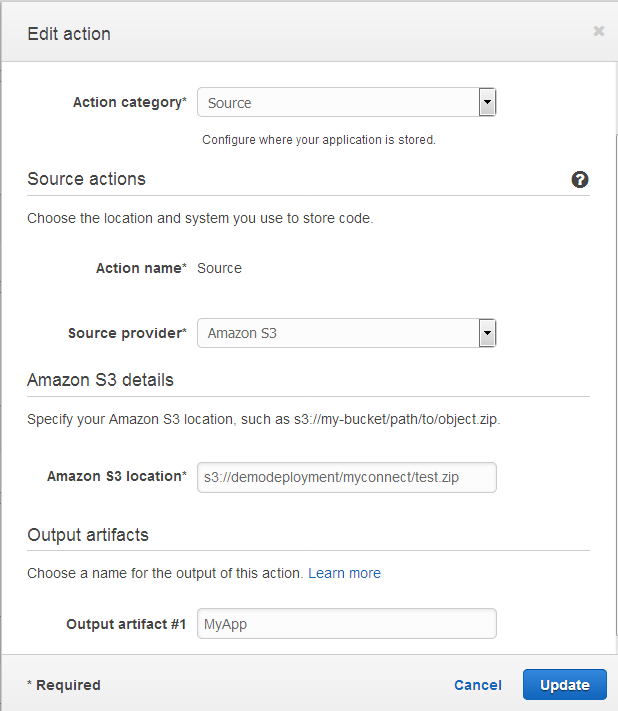
If you click on green part you can also see logs.

1. Now If we check “release” at time of making build then zip will be prepared of build and pushed to s3 bucket “deployment pipe” as shown



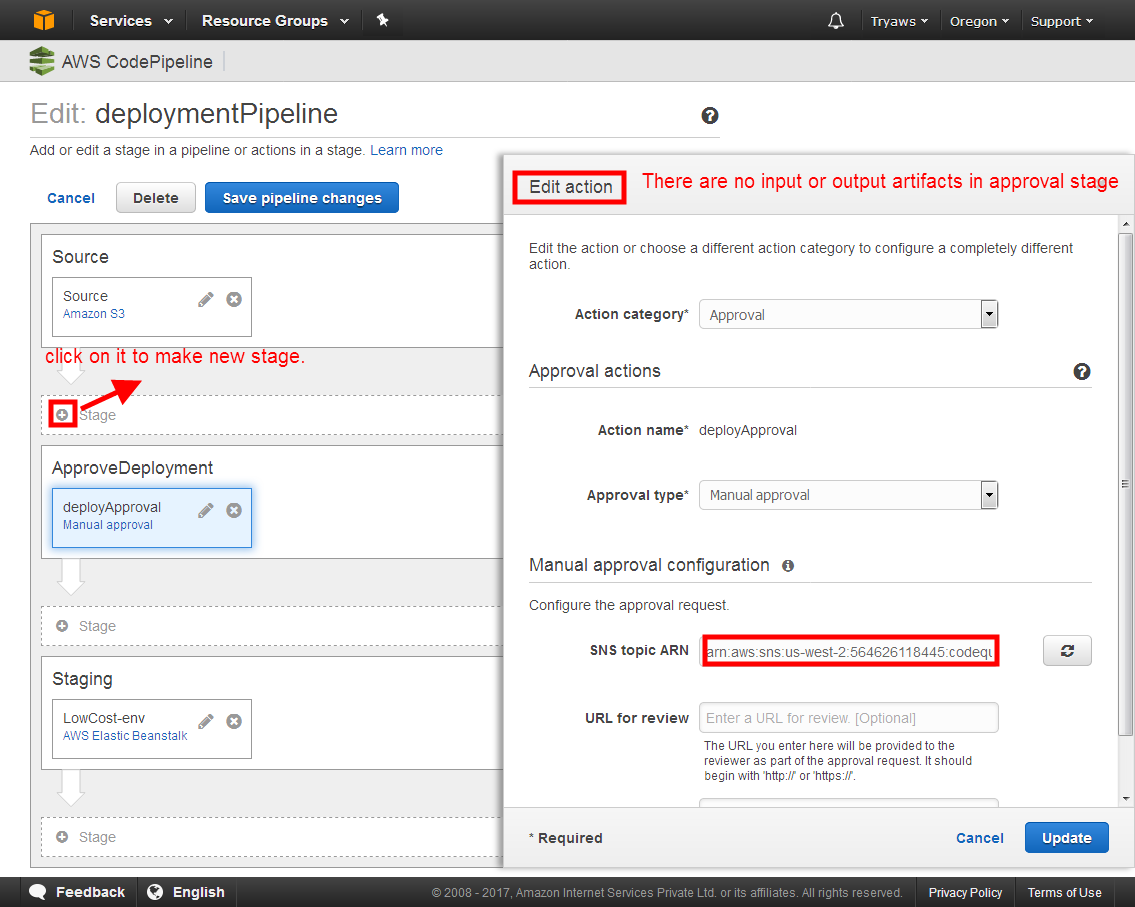
As can be seen in snapshot, zip form in myconnect folder in demodeployment bucket, (this is path we specify in Destination Bucket in job configuration). Every project zip should go in same bucket in folder on project name as here goes.

1. Its time to work with codepipeline. Codepipeline will work as, As demodeployment bucket myconnect folder get new file, codepipeline get triggered, it carry or source code from s3 bucket to its bucket(codepipeline make its own bucket which is region specific, and source all code in this bucket). Then it ask for approval by mail. We can approve by clicking on link in mail or manually by aws console. If want to deploy build on elasticbeanstalk then approve else reject it there.
2. There are 2 stages that are made in codepipeline, one is source and other is deployment with approval stage in between. Configuration for source stage will be

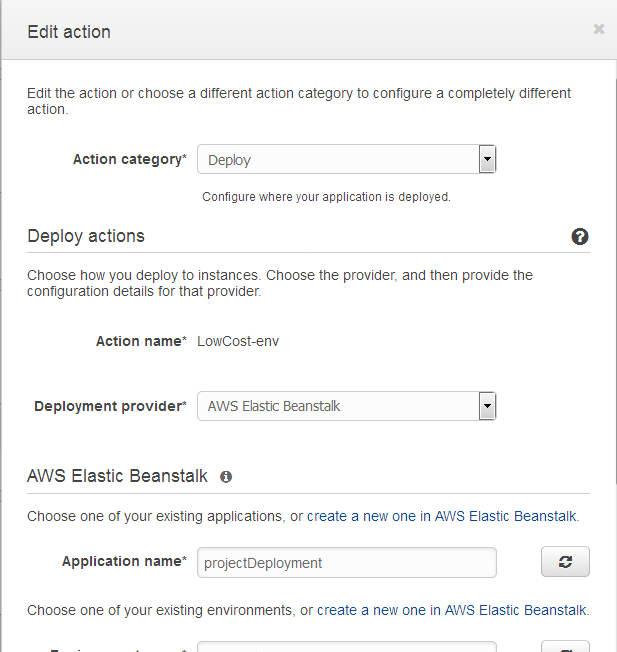
\*Note: When you make new codepipeline, you will be asked for build provider, as we are not keeping build step so keep it none.

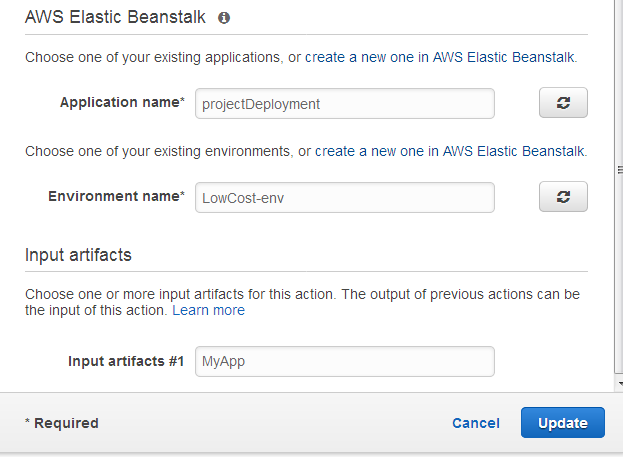
Here all the source code passes in form of MyApp output artifact to next step(elasticbeanstalk stage not approval[Approval is not stage]). Actually MyApp is name of folder inside codepipeline bucket in which codepipeline saves all code. So keep these names carefully.

If you want to create email notifications on every approval, you need to create SNS. We have prepared a SNS for email notification at [yatin.gupta@osscube.com](mailto:yatin.gupta@osscube.com), so everytime source stage succeed, request to approve comes at this email. Approval configuration will be seen as:



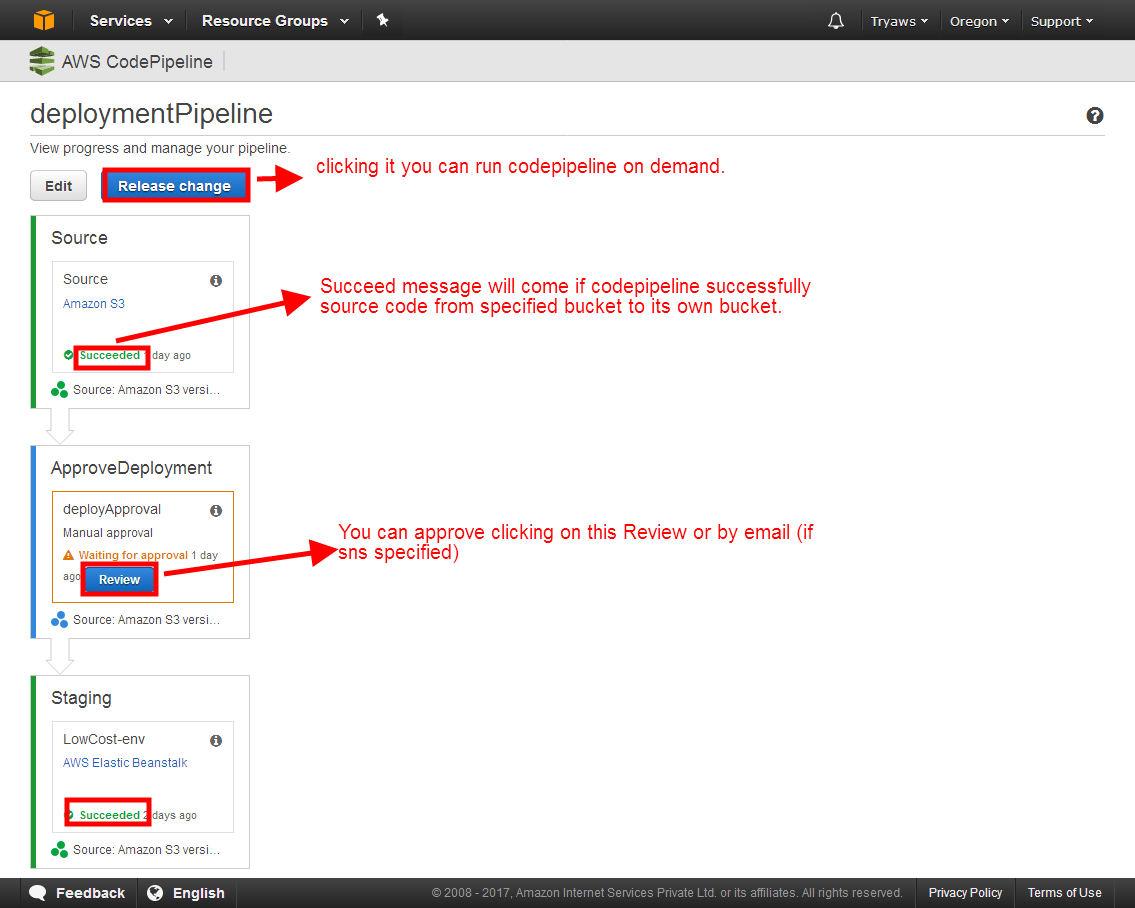
And then elastic beanstalk configuration will be seen as:





projectDeployment is elasticbeanstalk app that we make previously.

1. Now its times to run codepipeline.



Finally deployment process get over.

\*Note: In codepipeline ensure that what is output artifact of first stage, that should be input artifact of next stage or stages. First stage is always source stage, you cannot change it. First stage don’t have input artifact and last stage don’t have output artifact and approval stage don’t have any.

Now you can check by clicking on eb link and see deployed project.

Go to Elastic Beanstalk, Select your application then environment.



If in case following errors come:

# Sonarqube Container crashes with code 137 when given high load or Sonarqube exit with code 1

# Then you need to decrease memory taken by java runtime environment in running sonarqube which is done by open /opt/sonarqube-6.1/conf/sonar.properties and set as

# 

# 

END BUILD RELEASE DOCUMENT