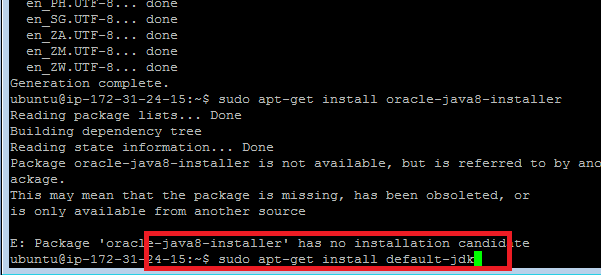
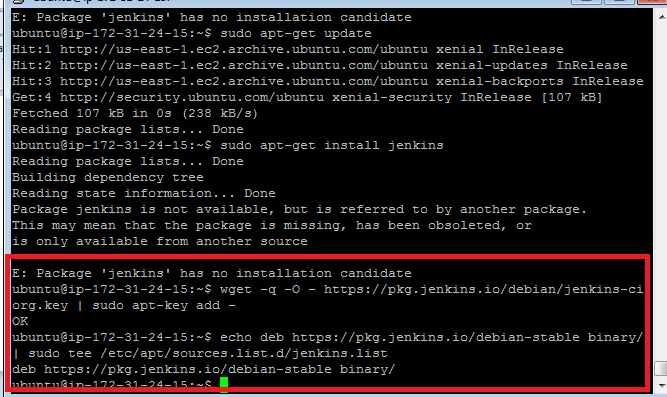


Install java:

  
First, we'll add the repository key to the system.

* wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -

When the key is added, the system will return OK. Next, we'll append the Debian package repository address to the server's sources.list:

* echo deb https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list

When both of these are in place, we'll run update so that apt-get will use the new repository:

* sudo apt-get update

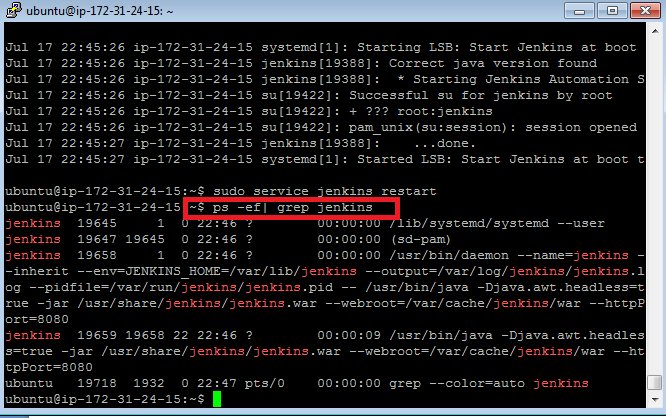
Finally, we'll install Jenkins and its dependencies, including Java:

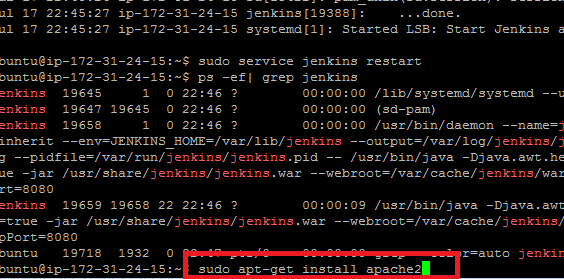
* sudo apt-get install jenkins

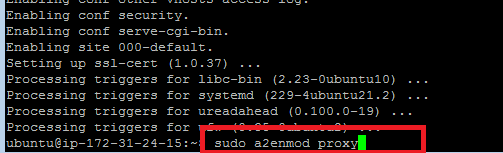
Now that Jenkins and its dependencies are in place, we'll start the Jenkins server.

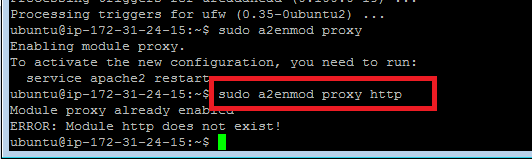
Sudo service Jenkins status

Sudo service Jenkins restart

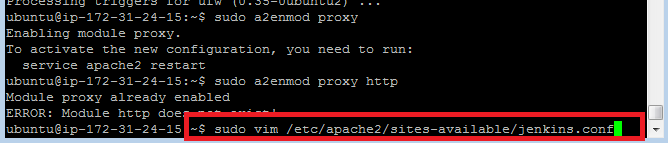
  
Install and configure apache:







sudo ae2nmod proxy\_http



Paste it:

<VirtualHost \*:80>

ServerName ec2-107-23-156-28.compute-1.amazonaws.com

ProxyRequests Off

<Proxy \*>

Order deny,allow

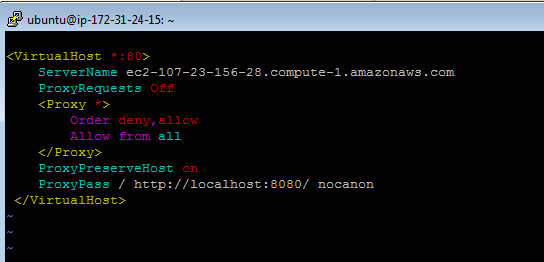
Allow from all

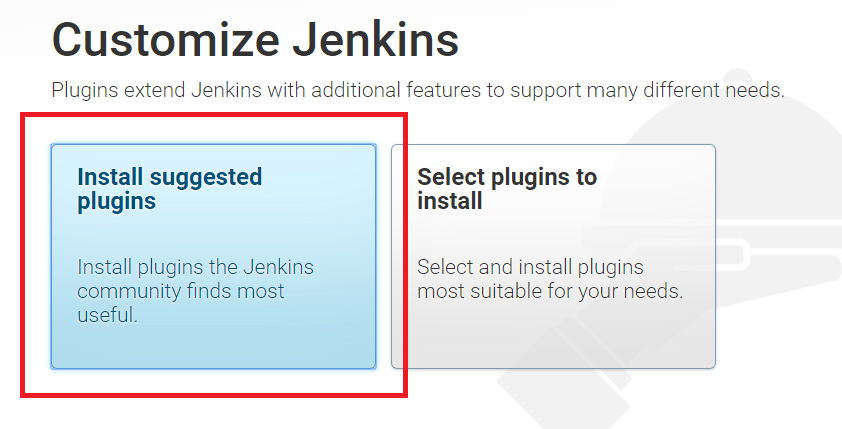
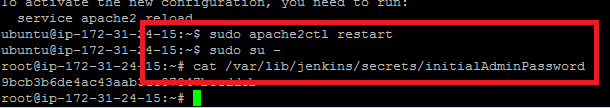
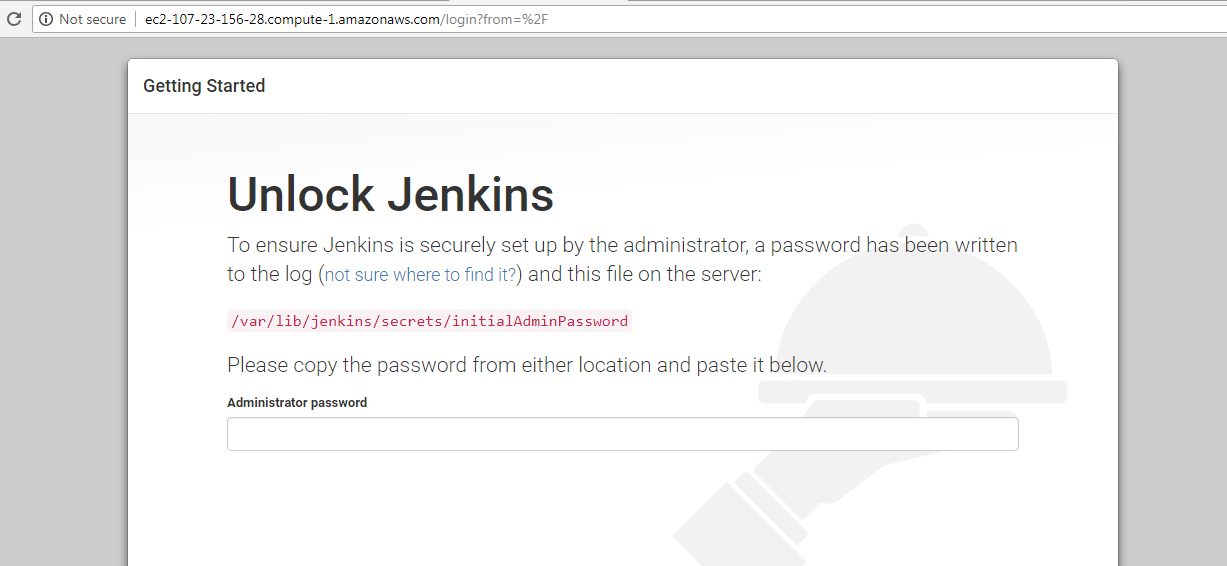
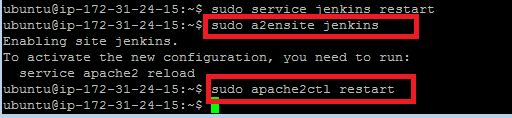
</Proxy>

ProxyPreserveHost on

ProxyPass / http://localhost:8080/ nocanon

</VirtualHost>





IAM role: codedeploy: amazonec2 role-> Permission: custom policy: name: codedeploy, edit trust policy

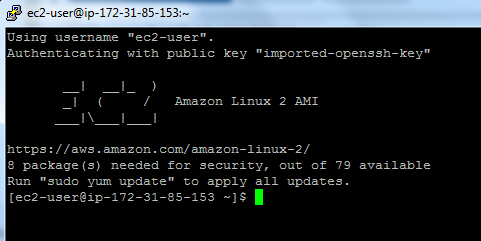
Ec2codedeploy-> amazon ec2 role-> custom policy: ec2codedeploy:

Create ec2 instance: linux: has amazon cli. IAM role: ec2 code deploy

<https://github.com/andrewpuch/code_deploy_example/>

|  |
| --- |
|  |
|  | { |
|  | "Version": "2012-10-17", |
|  | "Statement": [ |
|  | { |
|  | "Action": [ |
|  | "autoscaling:PutLifecycleHook", |
|  | "autoscaling:DeleteLifecycleHook", |
|  | "autoscaling:RecordLifecycleActionHeartbeat", |
|  | "autoscaling:CompleteLifecycleAction", |
|  | "autoscaling:DescribeAutoscalingGroups", |
|  | "autoscaling:PutInstanceInStandby", |
|  | "autoscaling:PutInstanceInService", |
|  | "ec2:Describe\*" |
|  | ], |
|  | "Effect": "Allow", |
|  | "Resource": "\*" |
|  | } |
|  | ] |
|  | } |
|  |  |
|  | // Policy Trust for Code Deploy |
|  | { |
|  | "Version": "2012-10-17", |
|  | "Statement": [ |
|  | { |
|  | "Sid": "", |
|  | "Effect": "Allow", |
|  | "Principal": { |
|  | "Service": [ |
|  | "codedeploy.us-west-2.amazonaws.com", |
|  | "codedeploy.us-east-1.amazonaws.com" |
|  | ] |
|  | }, |
|  | "Action": "sts:AssumeRole" |
|  | } |
|  | ] |
|  | } |
|  |  |
|  | // Instance Role for EC2 Instance |
|  | { |
|  | "Version": "2012-10-17", |
|  | "Statement": [ |
|  | { |
|  | "Action": [ |
|  | "s3:Get\*", |
|  | "s3:List\*" |
|  | ], |
|  | "Effect": "Allow", |
|  | "Resource": "\*" |
|  | } |
|  | ] |
|  | } |

Log into instance



When server is booted run the following commands as root.

Sudo su

yum -y update

yum install -y aws-cli

cd /home/ec2-user

2.) Here you will setup your AWS access, secret, and region.

aws configure

aws s3 cp s3://aws-codedeploy-us-east-1/latest/install . --region us-east-1

chmod +x ./install

3.) This is simply a quick hack to get the agent running faster.

sed -i "s/sleep(.\*)/sleep(10)/" install

./install auto

4.) Verify it is running.

service codedeploy-agent status

secret : JUWARzQB3fHAUbIcoxdrf9BirvMX24efX6Y93NBi

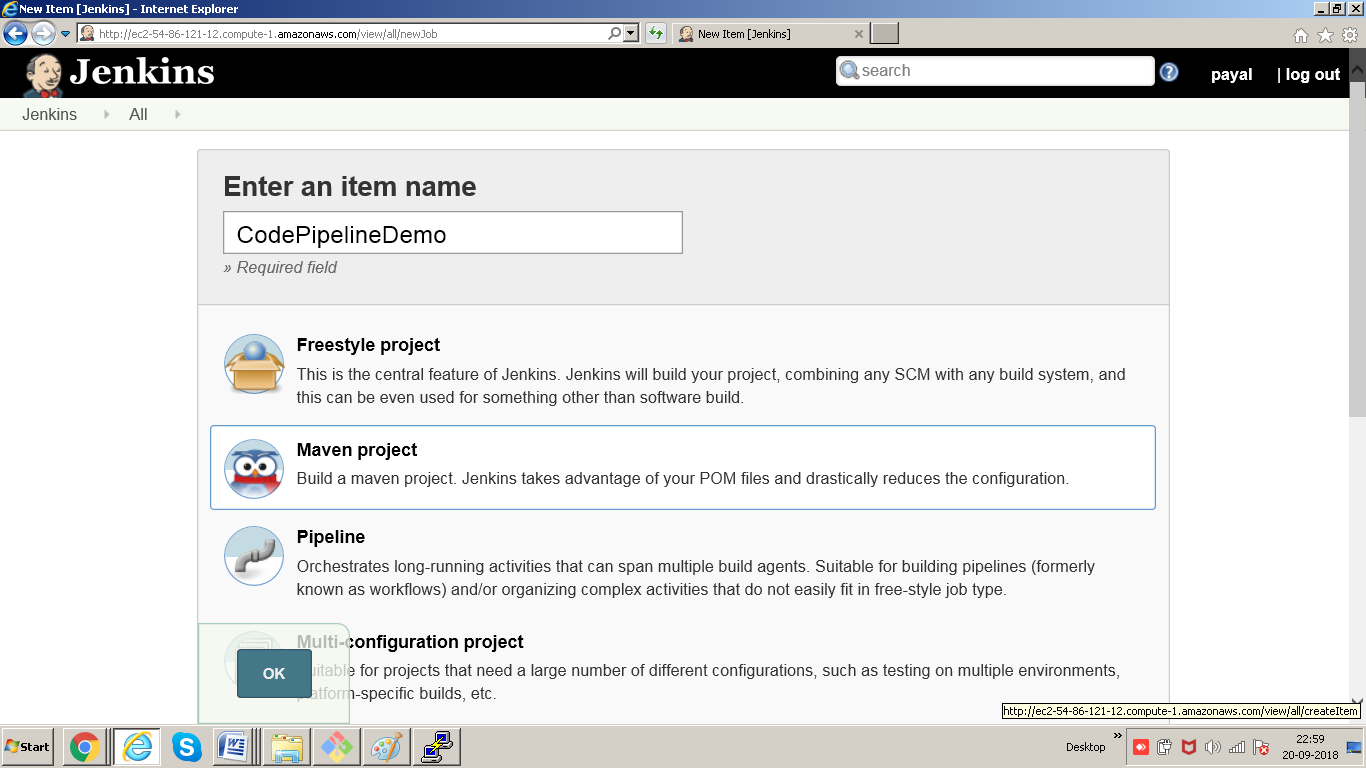
access key id: AKIAIRUJC663TMJA5WYQ

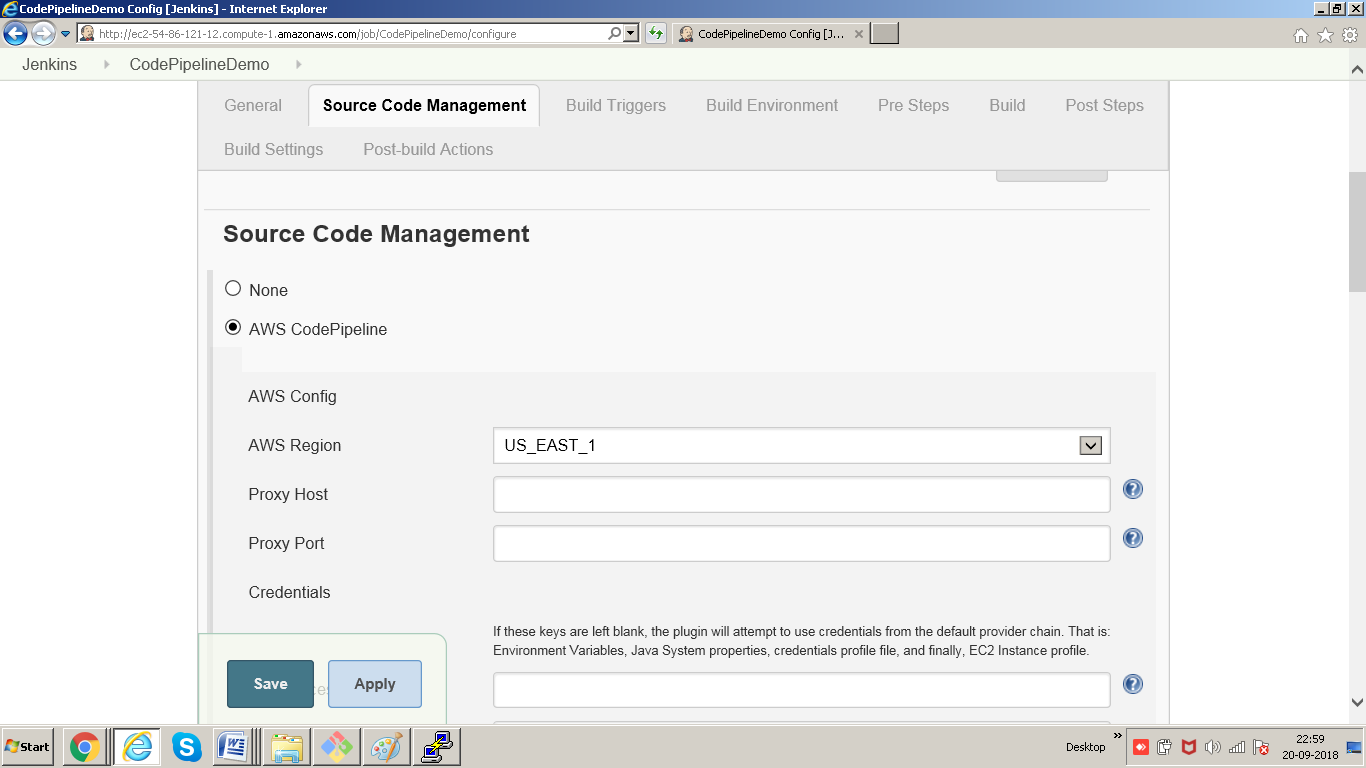
Jenkins: username:payal

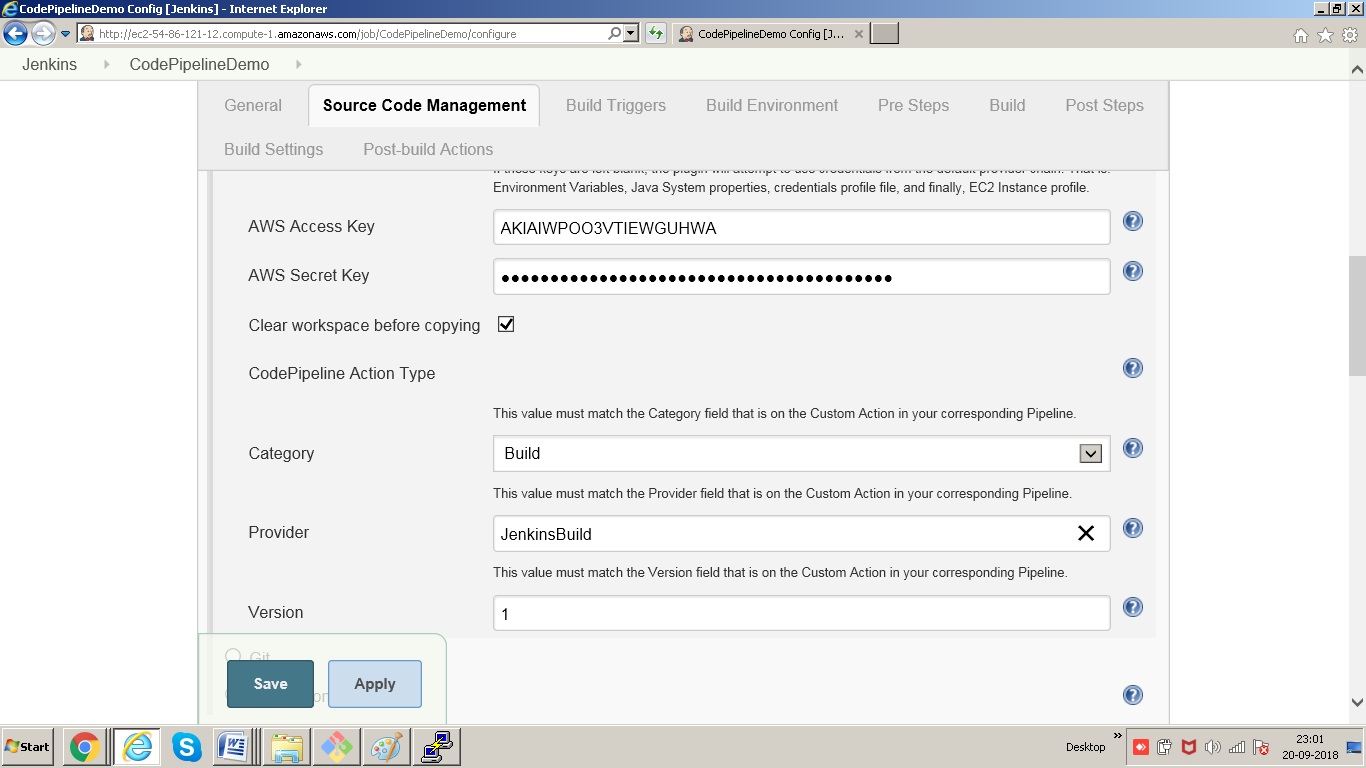
password: payal

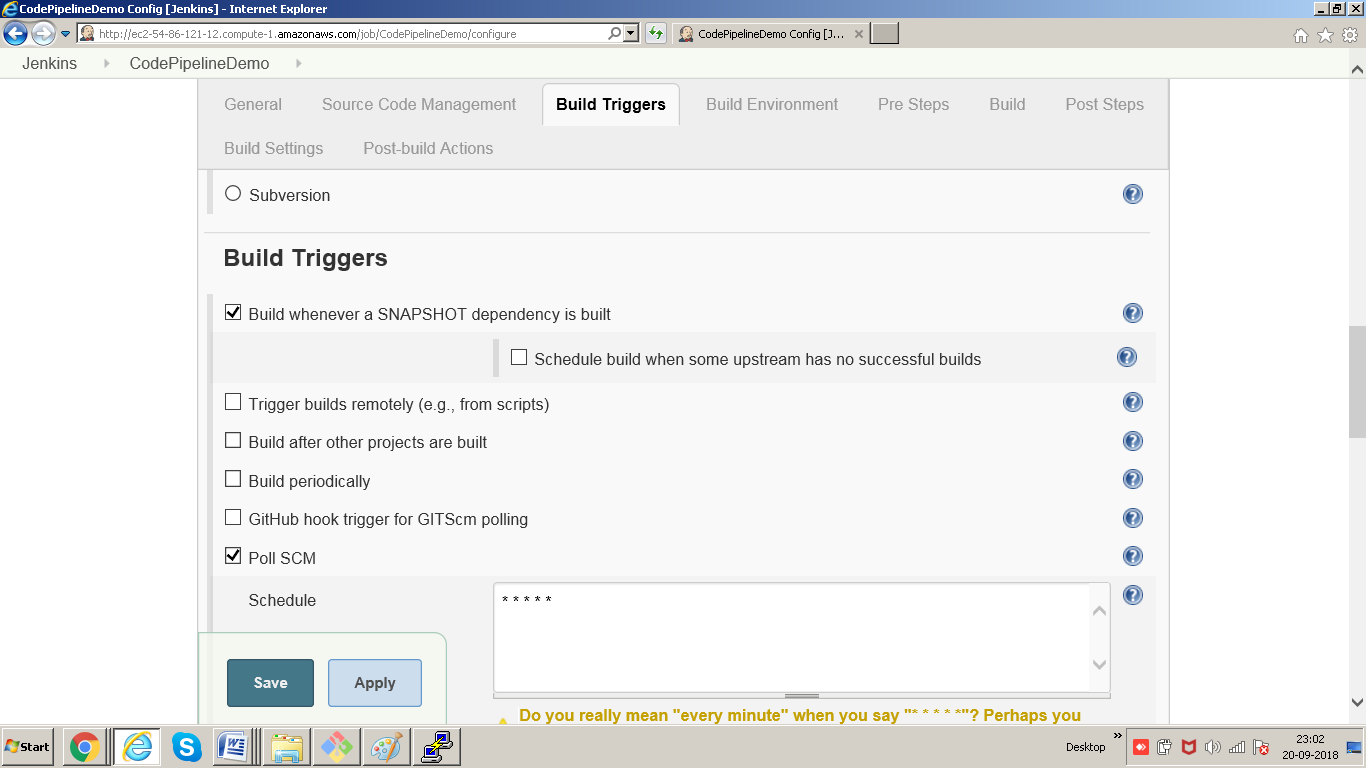
1) Manage jenkins: Install Maven Integration plugin, awscodepipeline plugin,

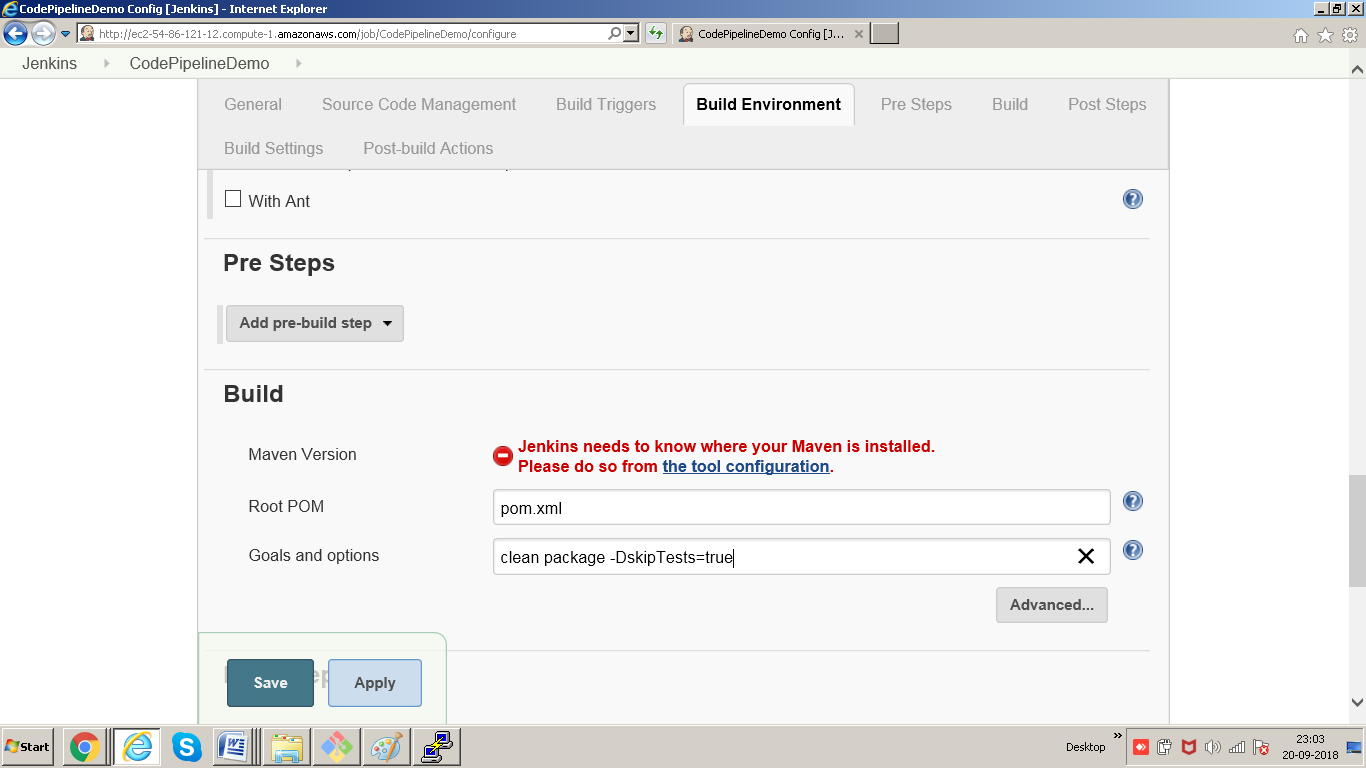
2) New item: Build a maven project

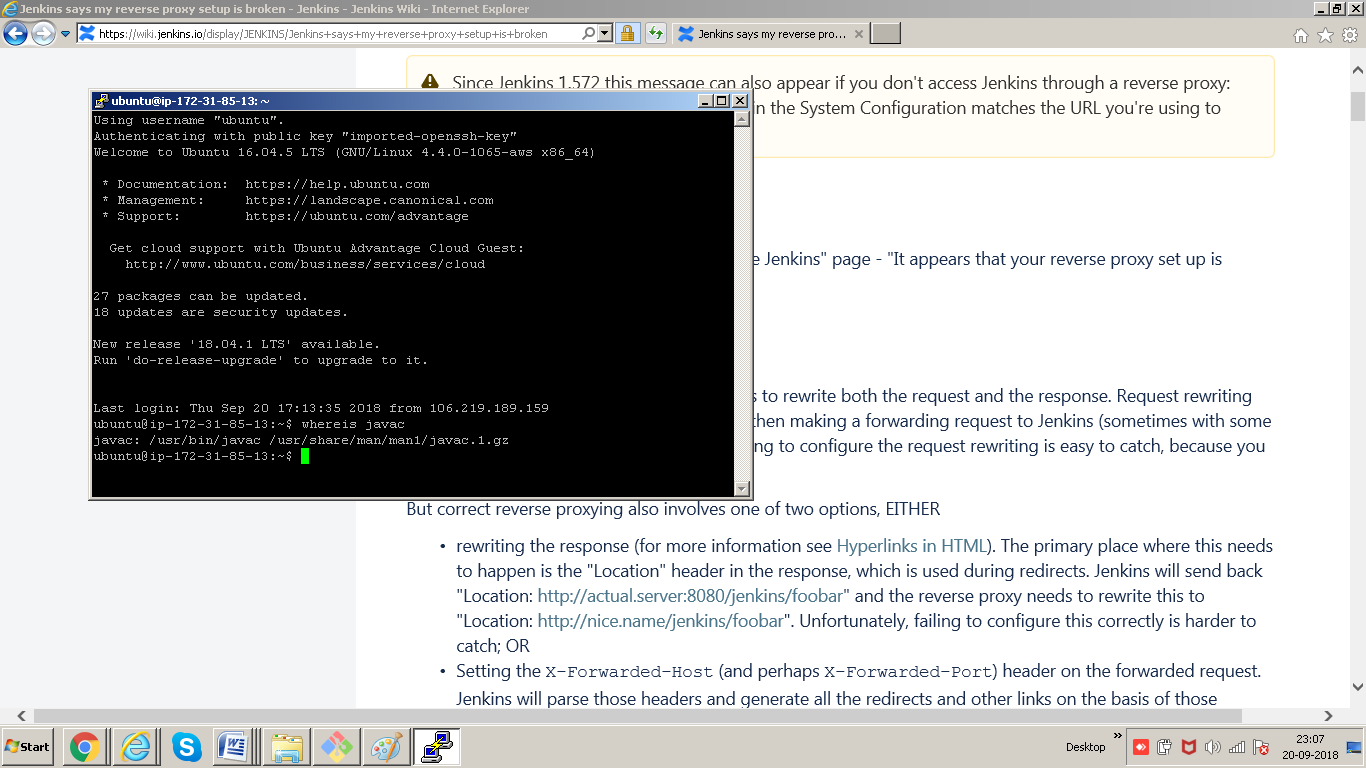


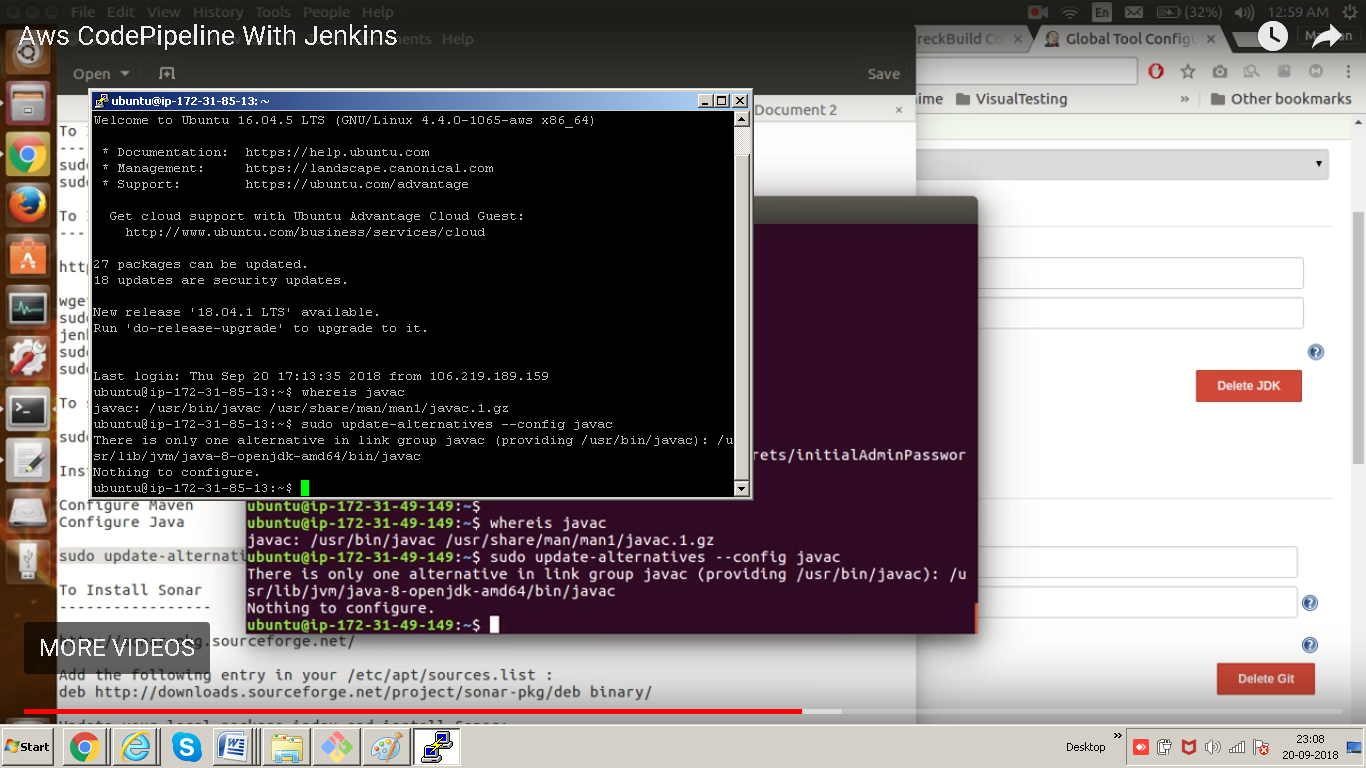


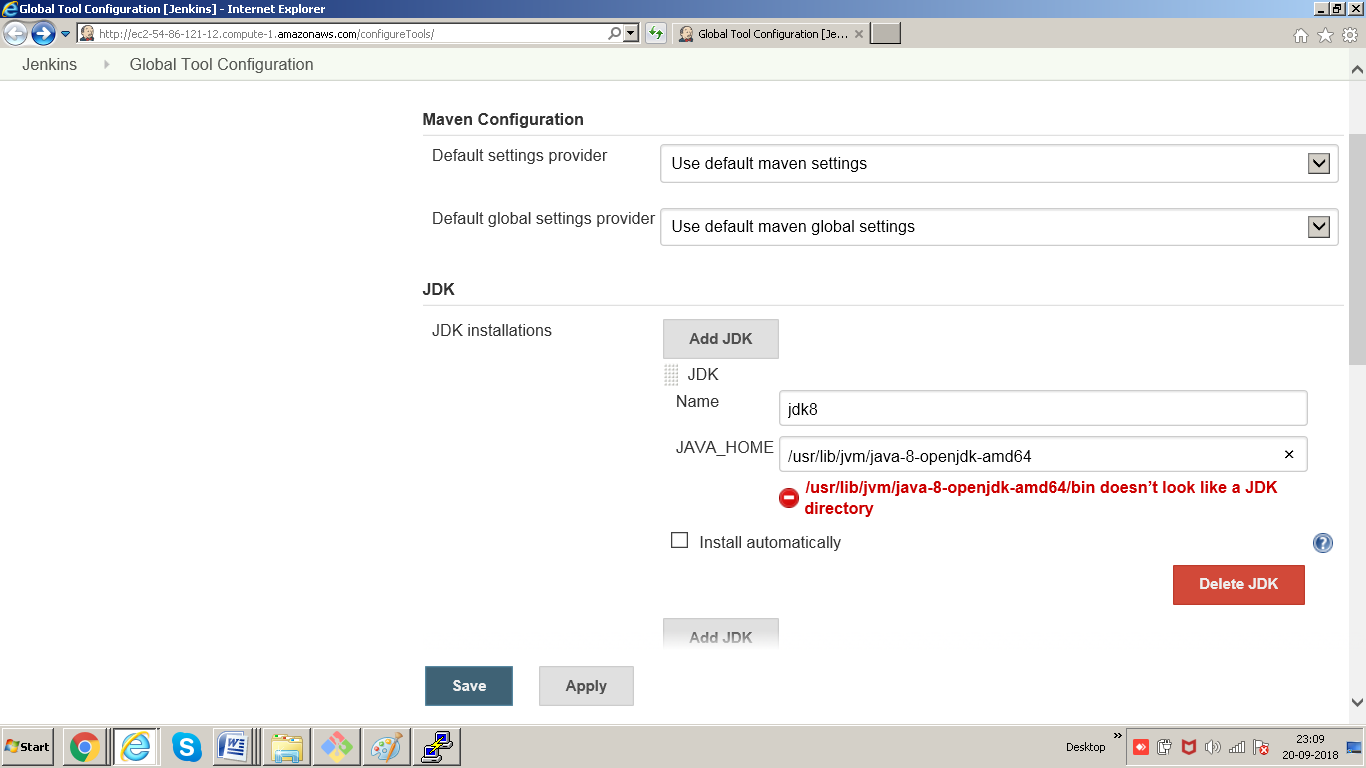


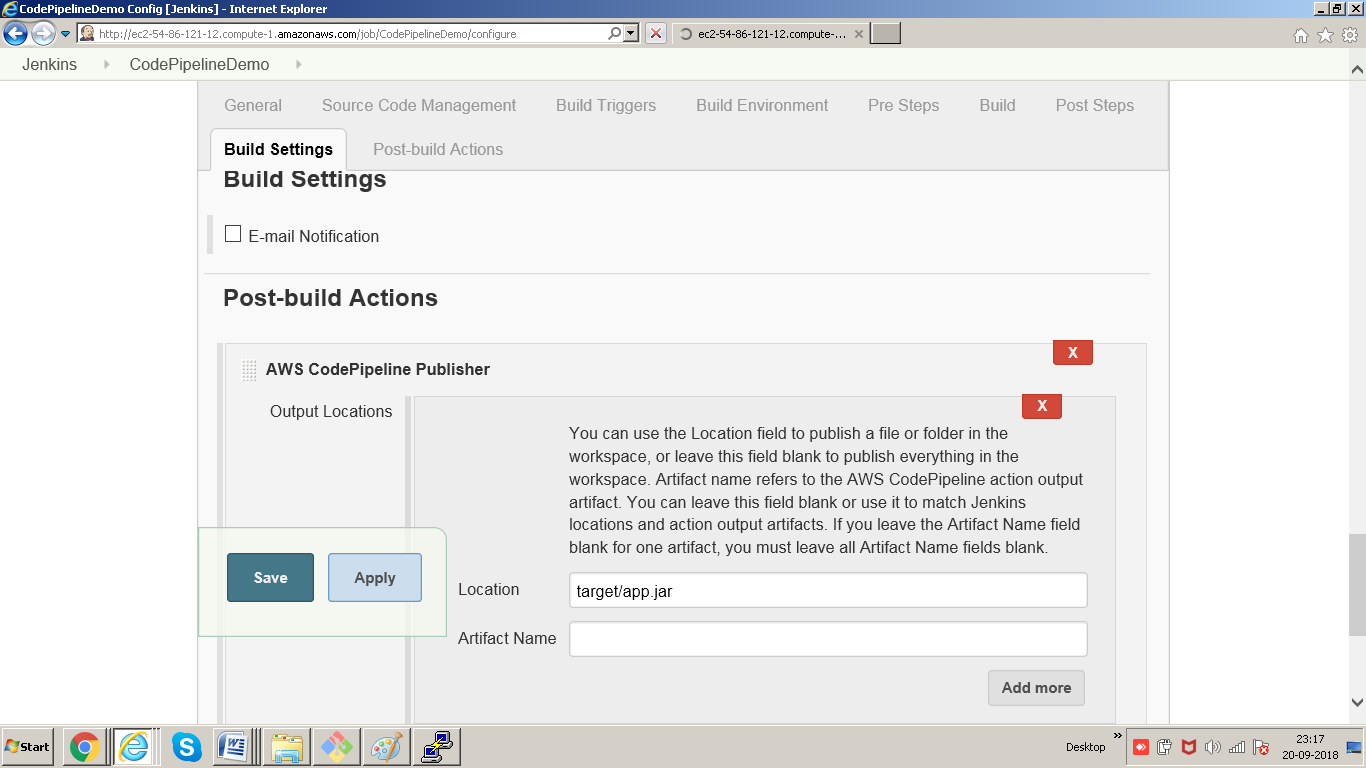








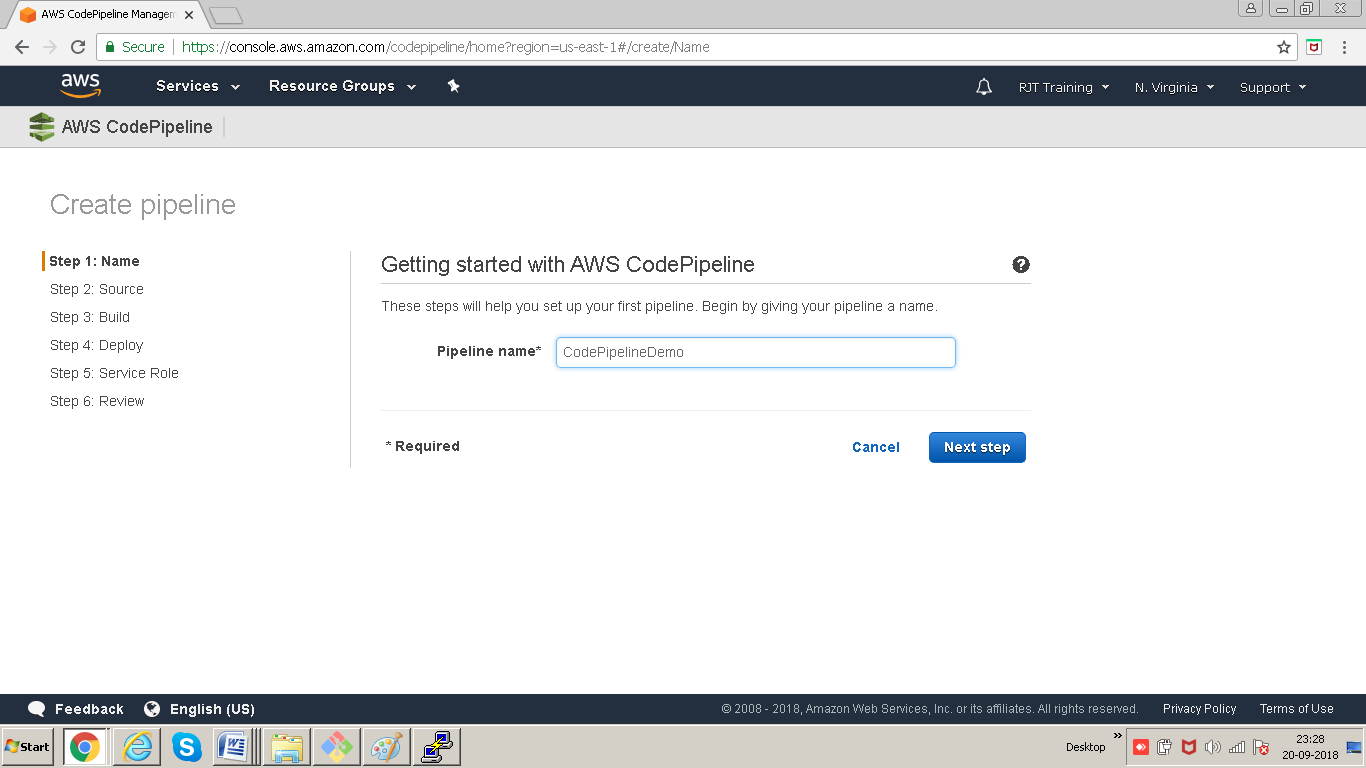


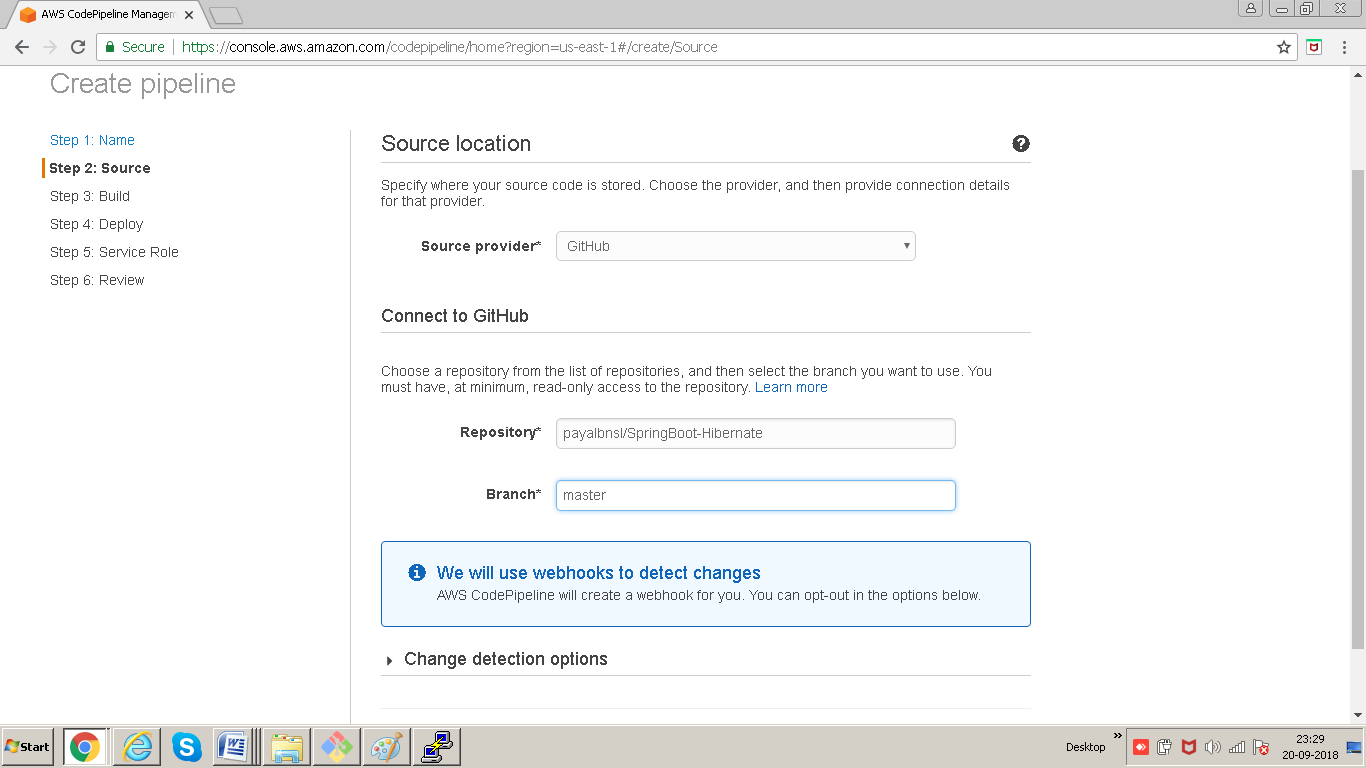


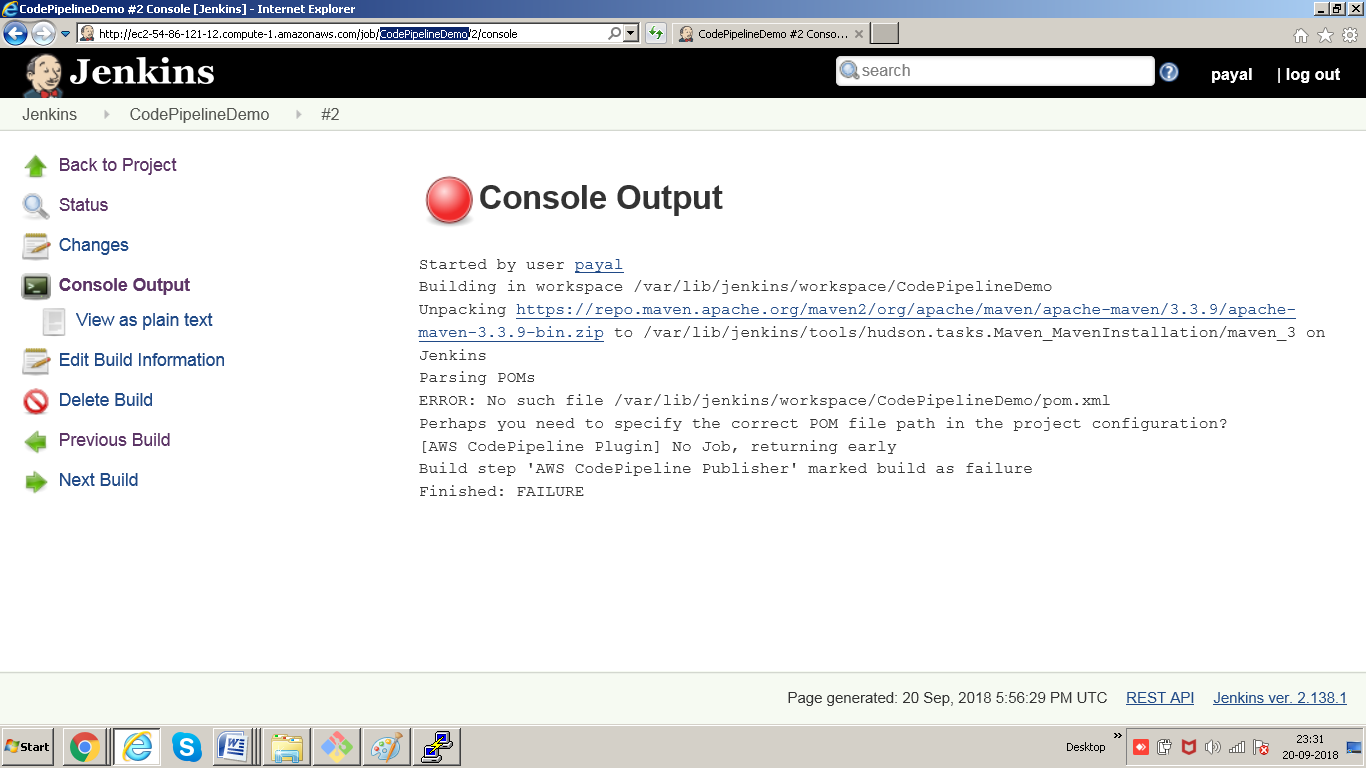
Jenkins is configured. Moving on to CodePipeline:

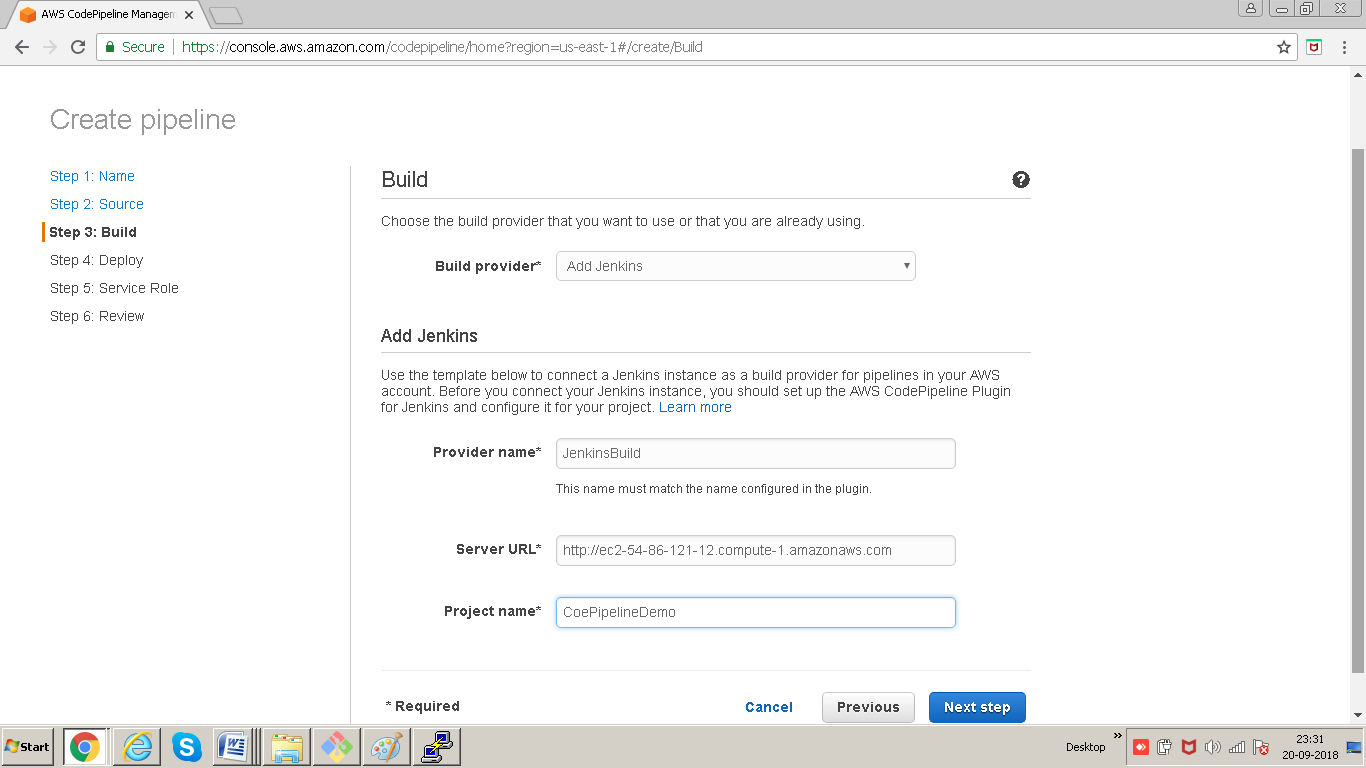


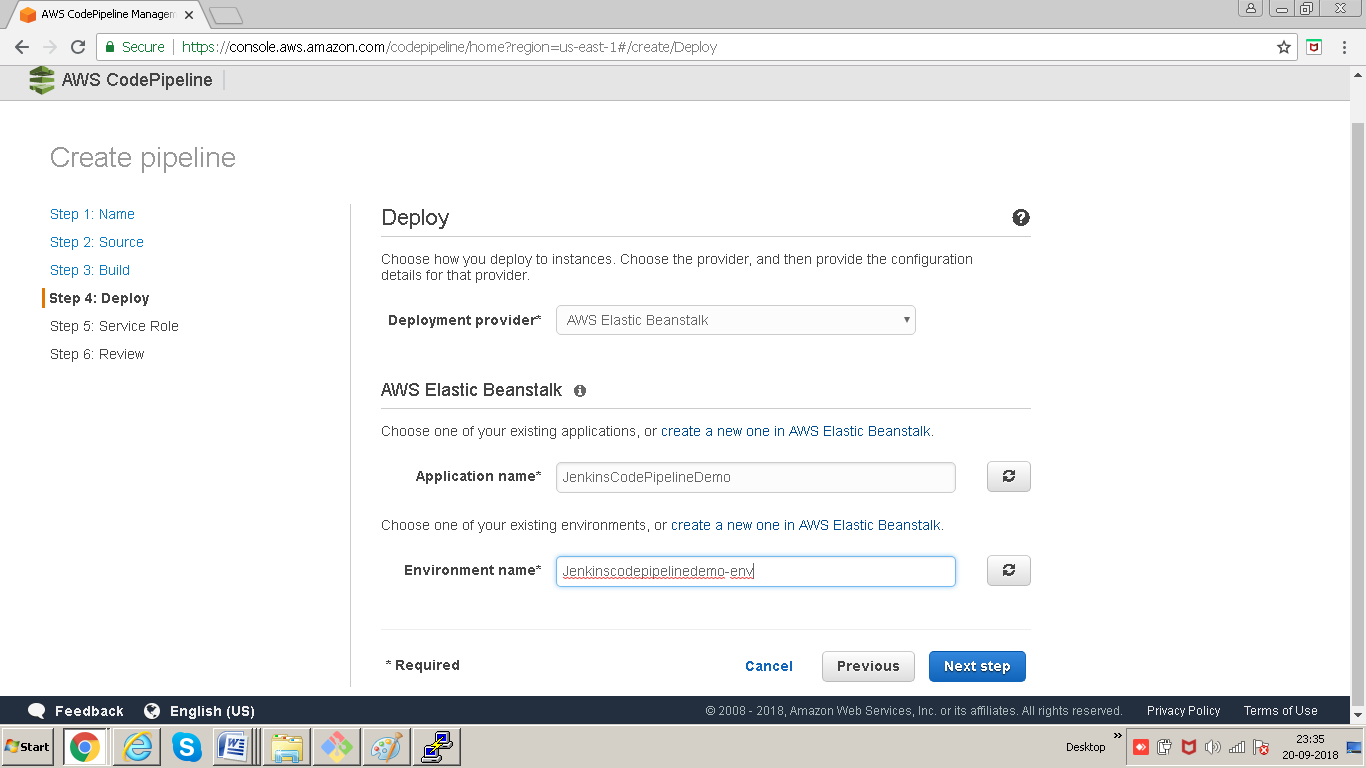
CodePipeline:



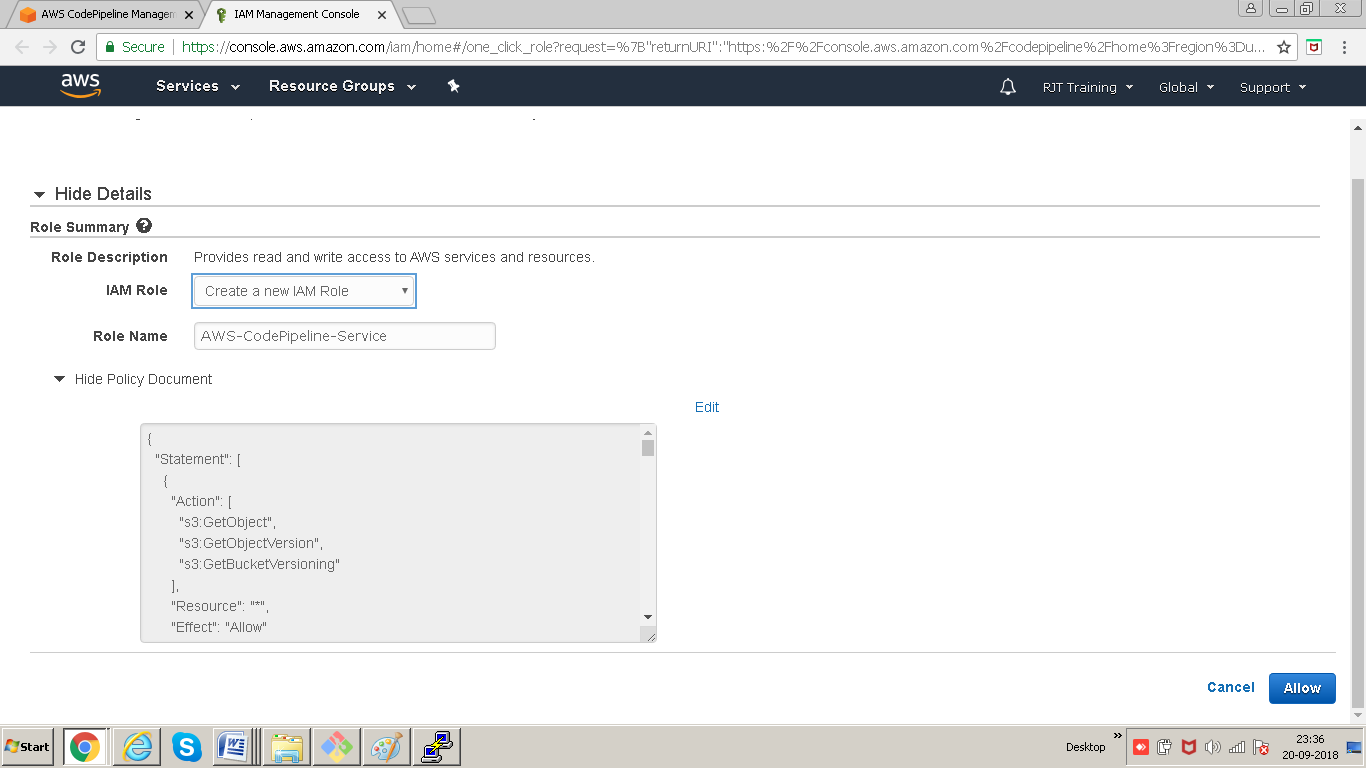


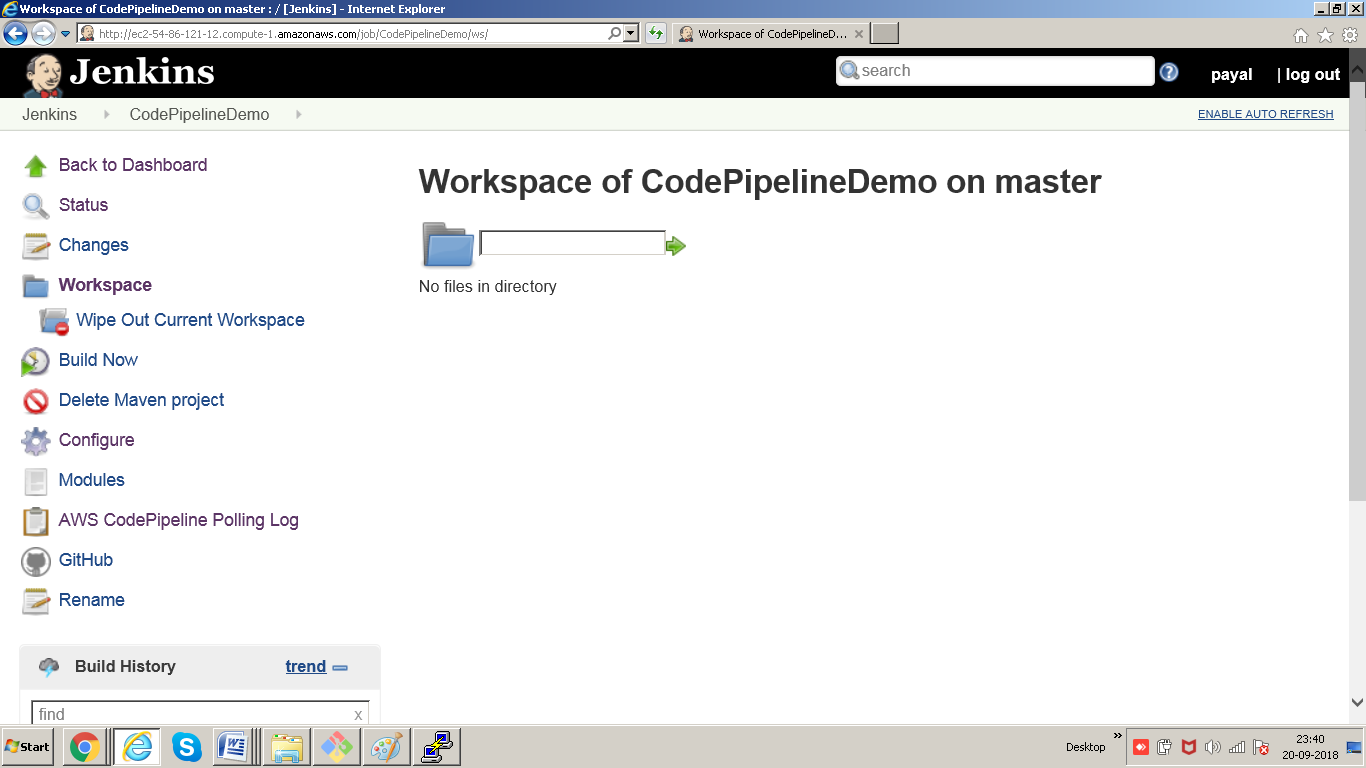
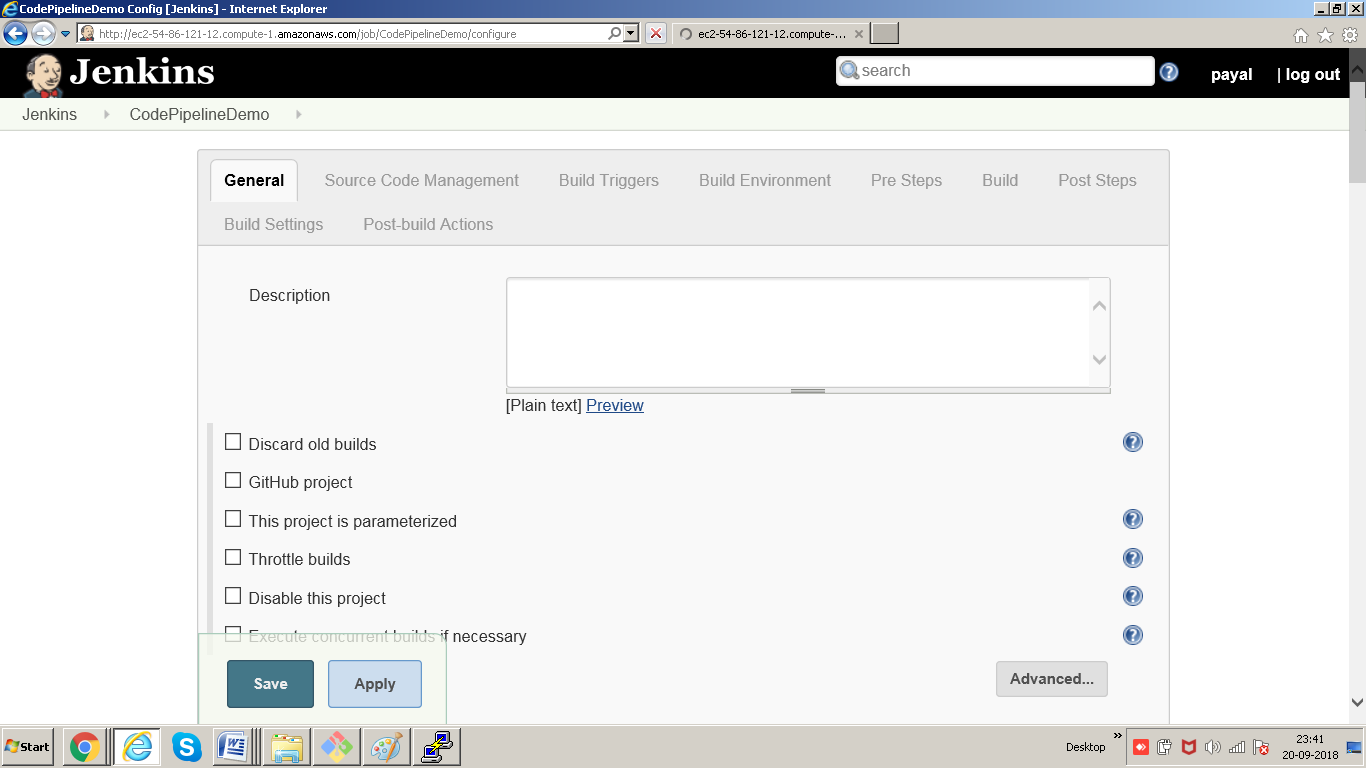


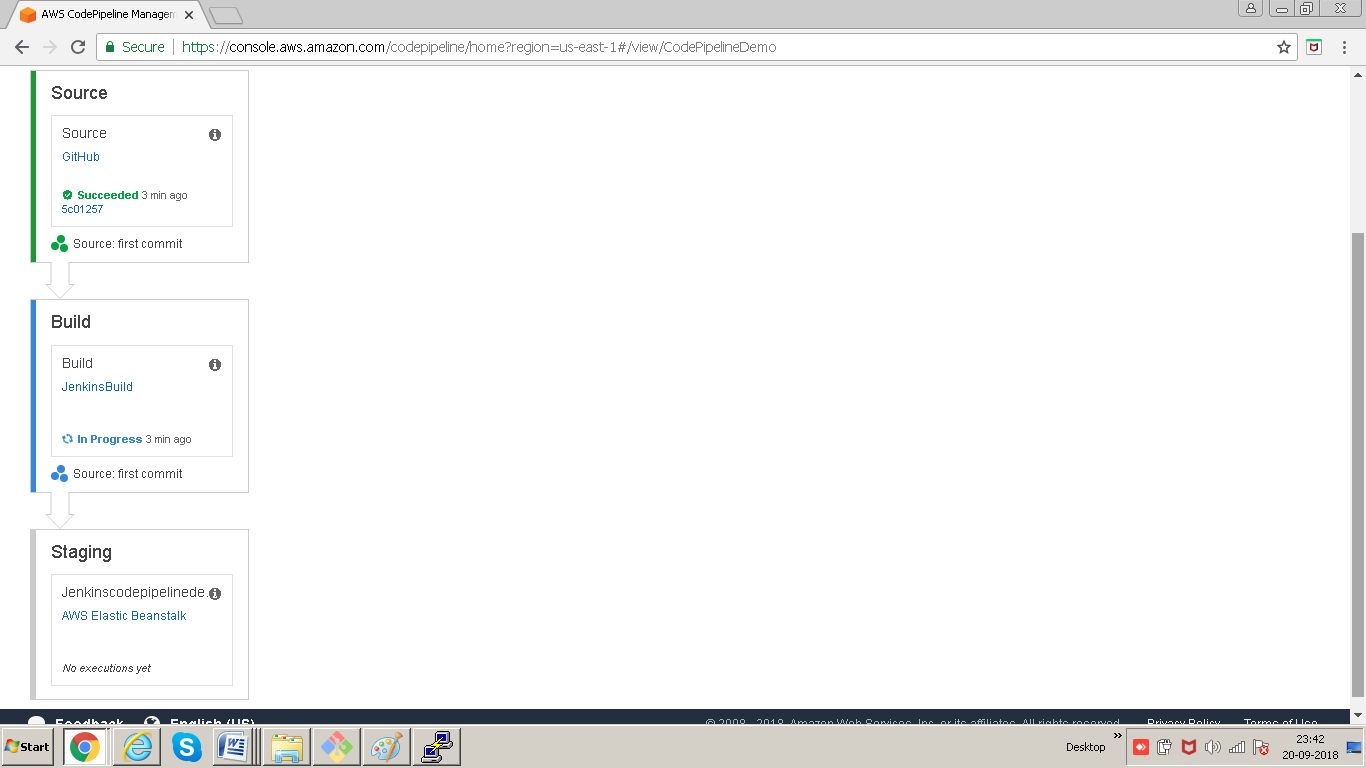




RoleName: create role





Aws codepipeline will checkout project from github, pass it to jenkins Jenkins will build and giveback artifact to the pipeline. Pipeline will deploy it to EBS.

To integrate with Jenkins, AWS CodePipeline requires you to install the AWS CodePipeline Plugin for Jenkins on any instance of Jenkins you want to use with AWS CodePipeline. You should also configure a dedicated IAM user to use for permissions between your Jenkins project and AWS CodePipeline. The easiest way to integrate Jenkins and AWS CodePipeline is to install Jenkins on an Amazon EC2 instance that uses an IAM instance role that you create for Jenkins integration. In order for links in the pipeline for Jenkins actions to successfully connect, you must configure proxy and firewall settings on the server or Amazon EC2 instance to allow inbound connections to the port used by your Jenkins project. Make sure you have configured Jenkins to authenticate users and enforce access control before you allow connections on those ports (for example, 443 and 8443 if you have secured Jenkins to only use HTTPS connections, or 80 and 8080 if you allow HTTP connections).