**Requirement:**

**## Create a Jenkins Job that will take two inputs:**

  1) Instance Id

  2) Instance Type

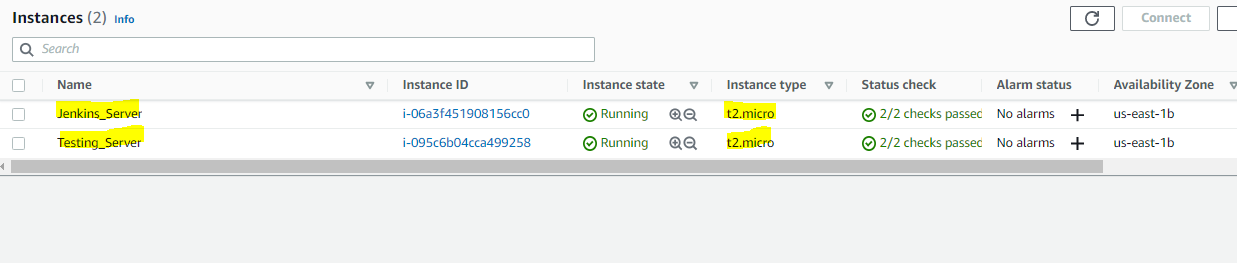
**## The Jenkins Job should perform 3 operations:**

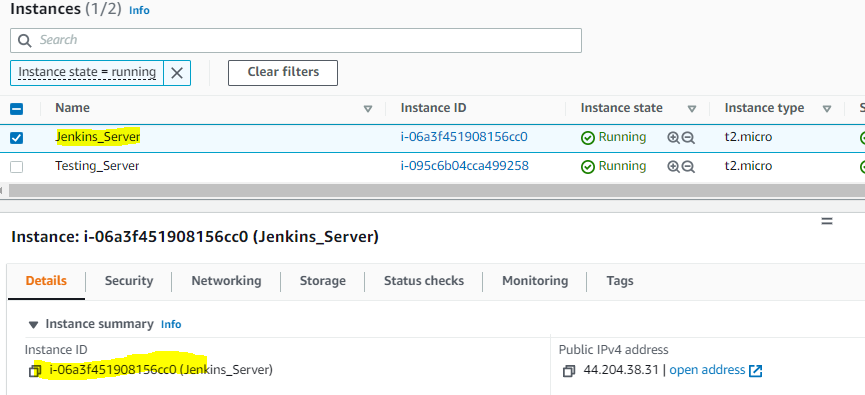
  1) Change the instance type of instance with the provided instance type.

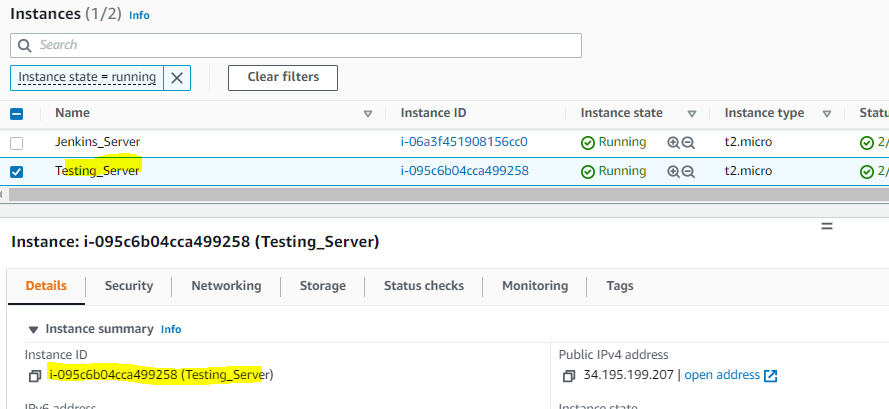
  2) Perform an operation using ansible on the provided instance

  3) Scale back instance to the previous instance type.

* Launch an ec2 instance for Jenkins\_server and testing\_server – for testing\_server I’ve associated ElasticIP







* Login to Jenkins\_server machine and install pre-requisites for Jenkins

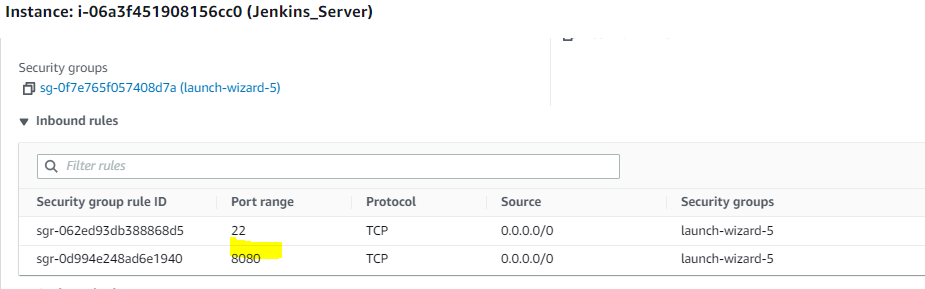
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

I’ve launched an ec2 instance of type amazon linux so installed jdk from amazon-linux-extras

1. amazon-linux-extras list
2. amazon-linux-extras install java-openjdk11 -y
3. yum install jenkins -y
4. service jenkins status
5. service jenkins start
6. service jenkins status
7. netstat -tupln
8. chkconfig jenkins on
9. cat /var/lib/jenkins/secrets/initialAdminPassword

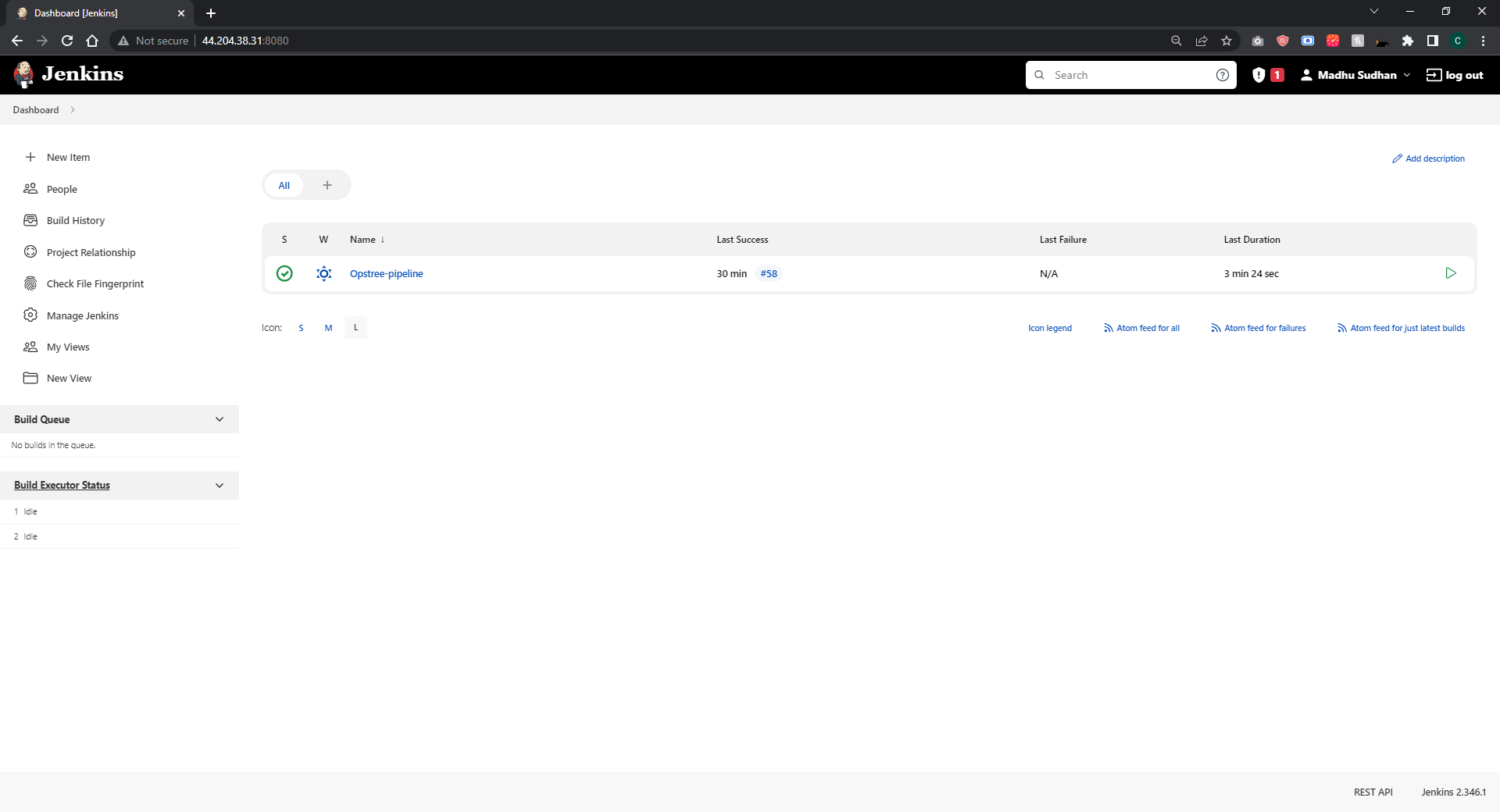
Access Jenkins GUI by allowing port 8080 in the security group (Source I’ve given as 0.0.0.0/0 for demonstration purpose)



Install the plugins suggested by Jenkins when the setup is done for the first type

Create a user name and password and then log in

Created a Pipeline project called - [Opstree-pipeline](http://44.204.38.31:8080/job/Opstree-pipeline/)

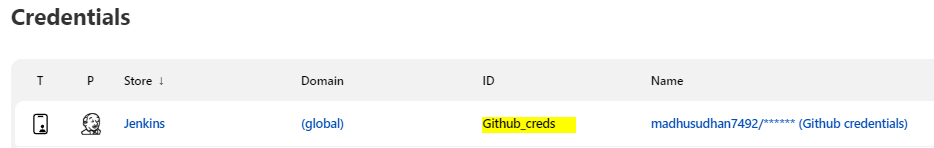


For two inputs – Instance\_Id and Instance\_Type, I’ve given them as parameterized inputs in Jenkins

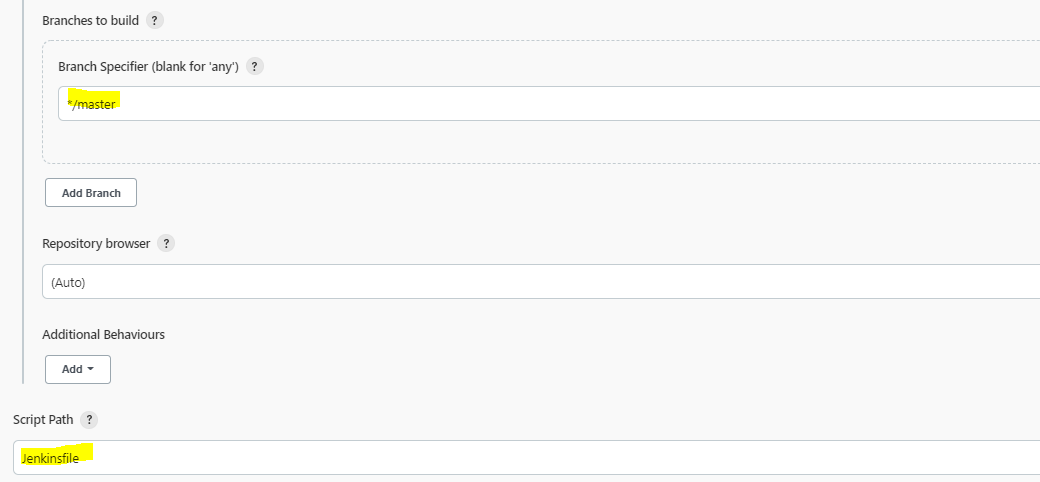


As this is a pipeline project, We need git to pull the code from the remote repository

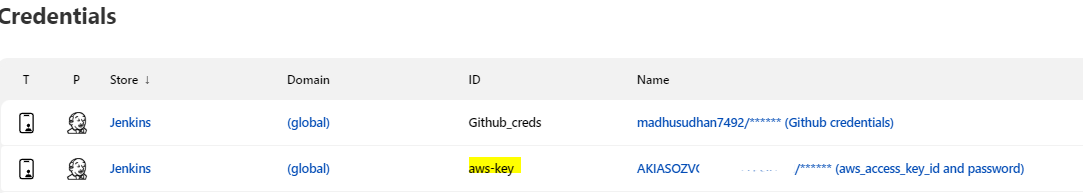
For credentials – I’ve configured username and password as global credentials





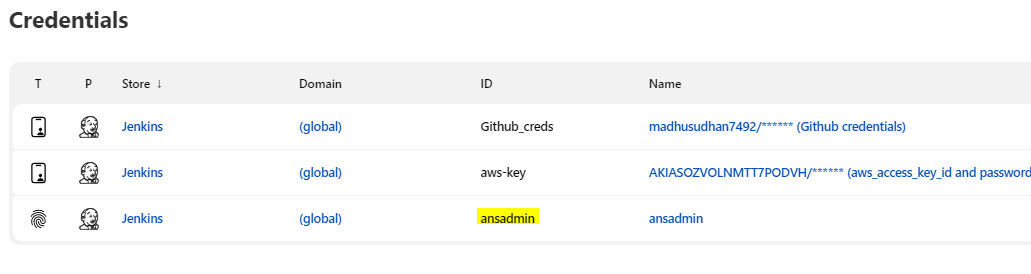


As Jenkins has to run aws CLI commands so configured aws-keys in global credentials

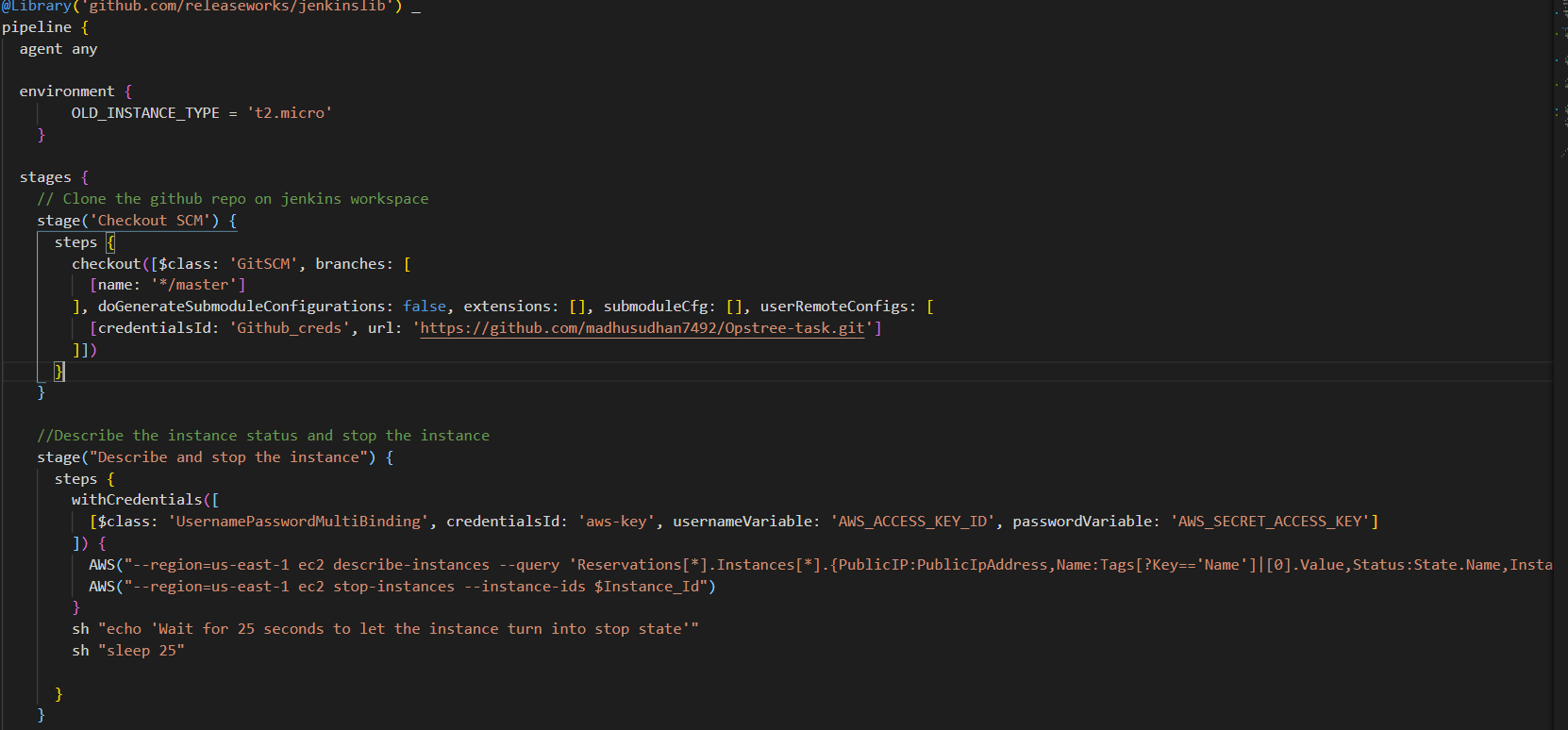


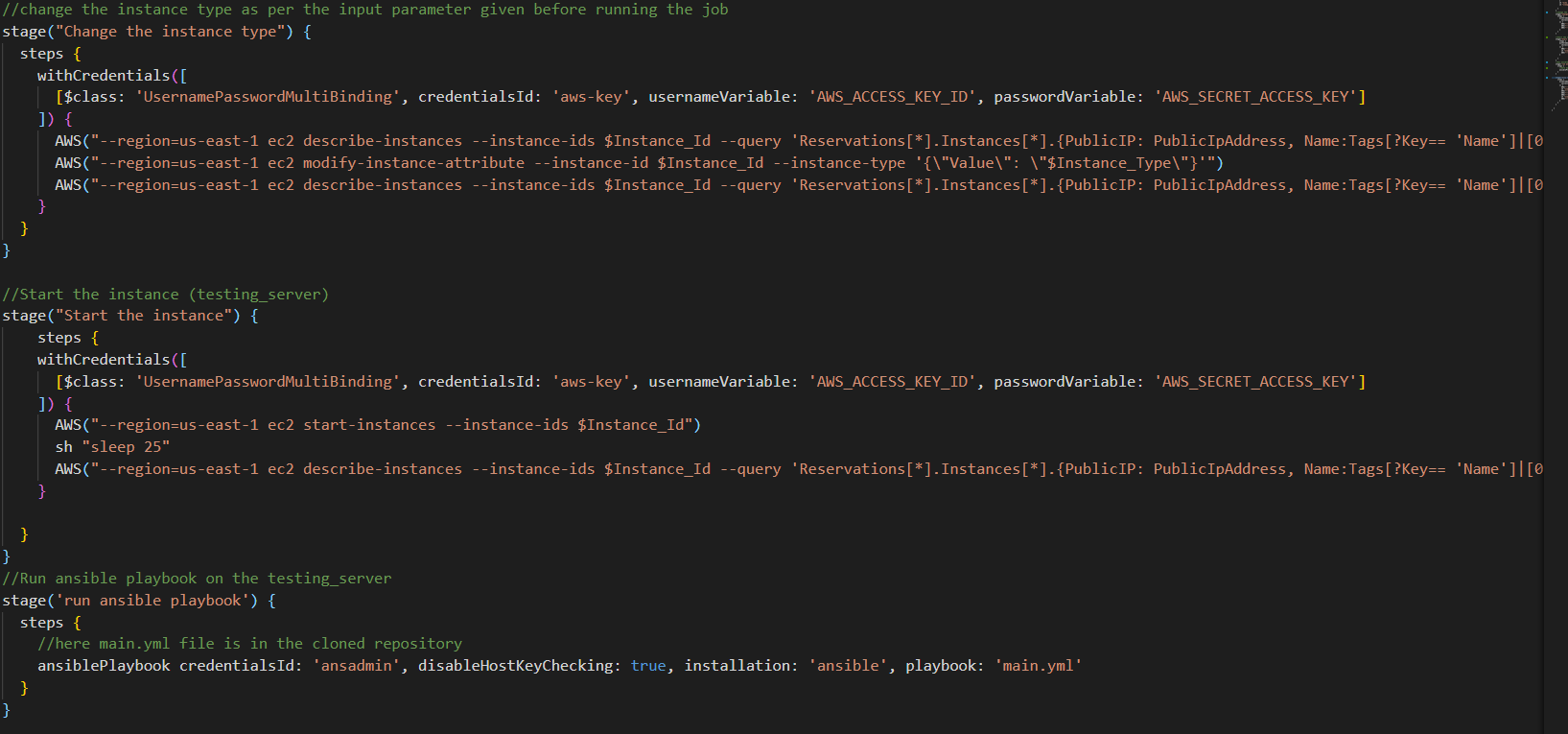
Below is the Jenkins file: <https://github.com/madhusudhan7492/Opstree-task/blob/master/Jenkinsfile>

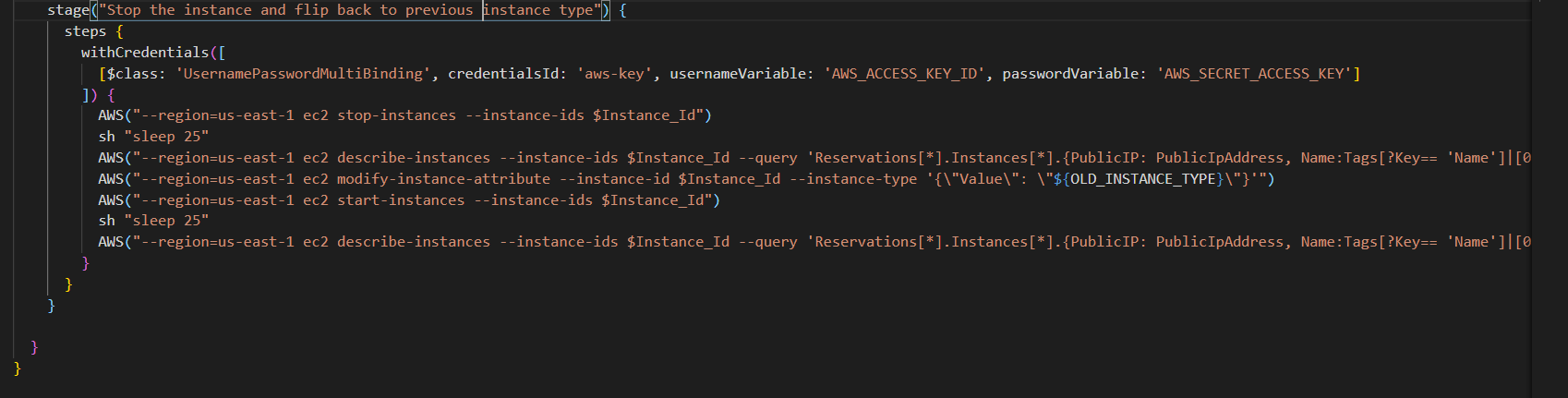
1. Checkout SCM – clone the given repo in Jenkins workspace and use Github\_creds which were configured as global credentials
2. Describe the instance and stop it – how many instances are there in the account and then stop the instance\_id which was given as input
3. Change the instance\_type which was given as input
4. Start the instance
5. Run the ansible-playbook (for ansible I’ve created ansadmin user in jenkins\_server and testing\_server then created and copied ssh public key from ansadmin user in jenkins\_server and pasted it in testing\_server
   1. In jenkins\_server
   2. Install ansible from amazon-linux-extras in Jenkins\_server
      1. useradd ansadmin
      2. passwd ansadmin
      3. su – ansadmin
      4. ssh-keygen -t rsa
      5. Copied id\_rsa.pub (public key) and paste in target node (testing\_server)
   3. In testing\_server
      1. useradd ansadmin
      2. passwd ansadmin
      3. su – ansadmin
      4. mkdir .ssh
      5. cd .ssh
      6. vi authorized\_keys
      7. Paste the copied public key
      8. Change the permission of .ssh folder and authorized\_keys file
      9. chmod 700 .ssh (read, write and execute permission for ansadmin)
      10. chmod 600 authorized\_keys (read and write permission for ansadmin)
   4. In jenkins\_server (control node) and testing\_server (target node) to check the connectivity
      1. ssh ansadmin@TargetnodeIP
   5. In jenkins\_server and testing\_server we need to give sudo root access to ansadmin so adding to wheel group
   6. Visudo from the root and uncomment - %wheel ALL=(ALL) NOPASSWD: ALL
      1. usermod -a -G wheel ansadmin (adding secondary group as the wheel)
   7. Also given ansadmin username and password as global credentials
   8. In ansible playbook, I’ve written a task to install httpd and start the service in main.yaml in the dev host group (I’ve given testing\_server IP in /etc/ansible/hosts file)



1. Once the ansible-playbook has run, next step is to stop the instance and flip to the previous instance type (here I’ve considered an env variable as OLD\_INSTANCE\_TYPE as “t2.micro”)
2. For describe-instances, start instance, stop instance, modify instance type – I’ve configured AWS Access key and AWS Secret key as global credentials and imported @Library('github.com/releaseworks/jenkinslib') \_ - This will load the Releaseworks Jenkins library, that includes the helper function that we will use. And then used withCredentials to get the aws-key and executed commands on specified region and also to execute from jenkins user installed docker and added jenkins user to docker.sock and changed permission to docker.sock file - chmod 777 /var/run/docker.sock
3. As part of the plugins i’ve installed Ansible plugin and given the location of ansible /usr/bin/ in the Global Tool configuration
4. Link to latest build output - http://44.204.38.31:8080/job/Opstree-pipeline/58/consoleText

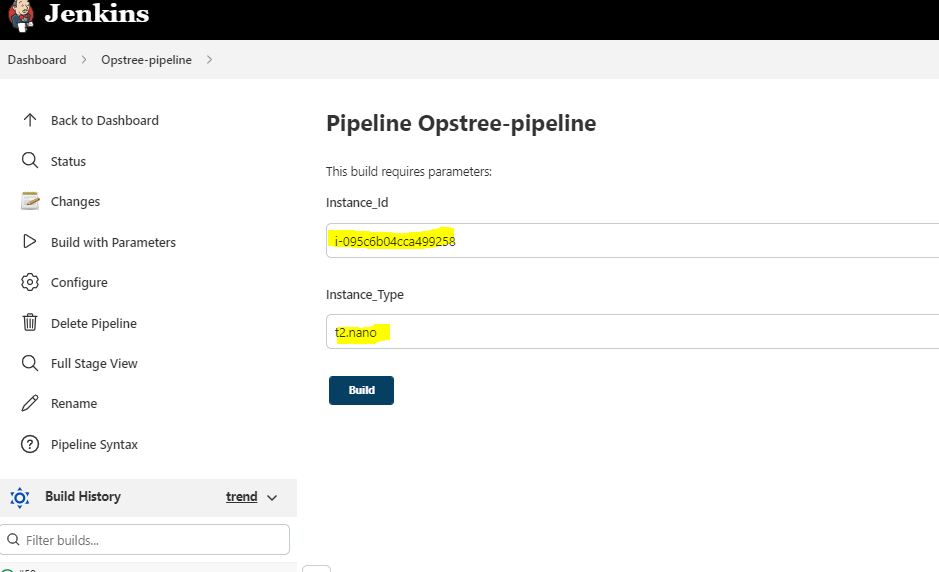


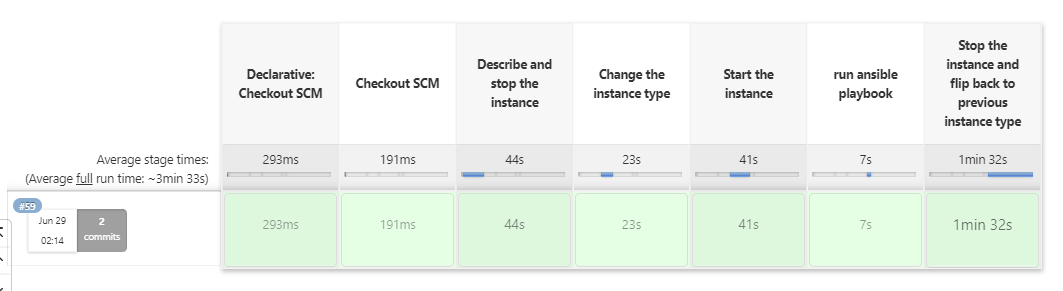




**Demo:**

1. Build with the below parameters





**Build output:** 

**Further improvements:**

1. Instead of one playbook, writing ansible roles
2. Instead of one instance id, writing for loop and using aws describe\_instances method to get all the instance\_ids for that region and change the instance\_type and invoke then Ansible role to them
3. We can also take region as an input parameter and invoke that in Jenkinsfile
4. In Jenkinsfile I’ve hardcoded the OLD\_INSTANCE\_TYPE, which can be further improved by running aws describe\_instances and getting the instance type and storing it in some variable, and then calling it while flipping back