**Implementation steps**

1. Create a IAM user with administrator access. Generate access key and secret access key for this user

2. Launch an ec2 instance and create an AWS profile on that machine or your local machine.

*aws configure*

3. Install git on this VM and clone the below repository which contains the cloud formation templates to create ECS cluster and Jenkins instance.

4. Change to working directory of the cloned repository and execute below command to create a Cloud formation stack which will create all the resources needed for ECS cluster

*aws cloudformation create-stack --template-body file://ecs-cluster.template --stack-name EcsClusterStack --capabilities CAPABILITY\_IAM --tags Key=Name,Value=ECS --region us-east-1 --parameters ParameterKey=KeyName,ParameterValue=mani1 ParameterKey=EcsCluster,ParameterValue=test-cluster1 ParameterKey=AsgMaxSize,ParameterValue=3*

Below stacks will be created

AWS::IAM::Role

AWS::EC2::VPC

AWS::EC2::InternetGateway

AWS::EC2::VPCGatewayAttachment

AWS::ECS::Cluster

AWS::IAM::InstanceProfile

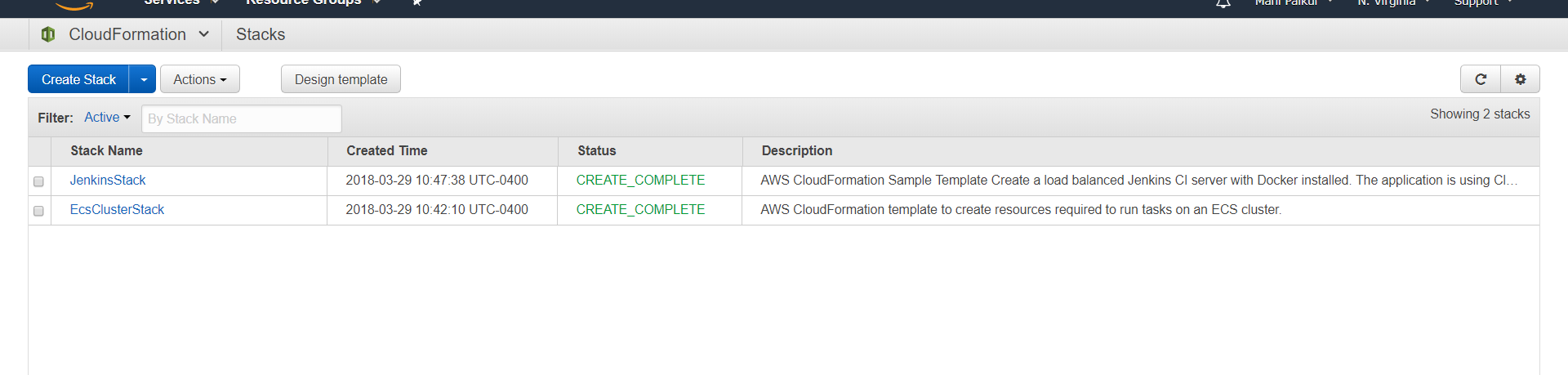
AWS::EC2::Subnet

AWS::EC2::RouteTable

AWS::EC2::SecurityGroup

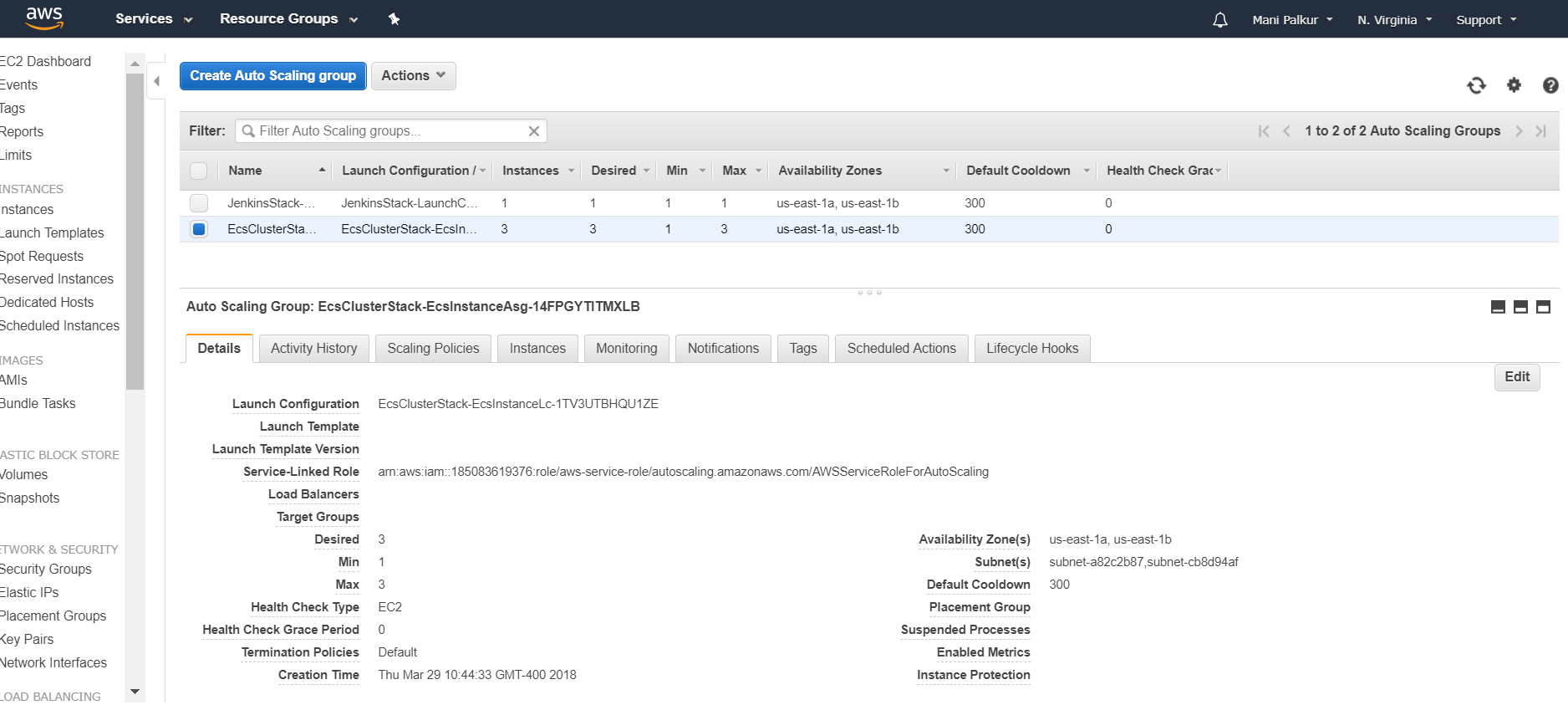
AWS::EC2::SubnetRouteTableAssociation

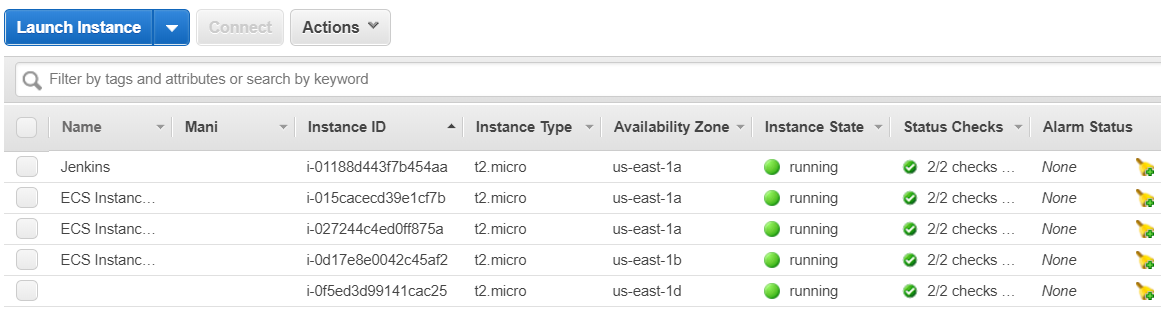
AWS::AutoScaling::AutoScalingGroup



5. The ECS cluster will have min value of 1 and max instances value of 3. The desired state value will be 2

This means at any given time 2 instances will be running as per the desired state. So, if these instances get terminated new instances will be created.

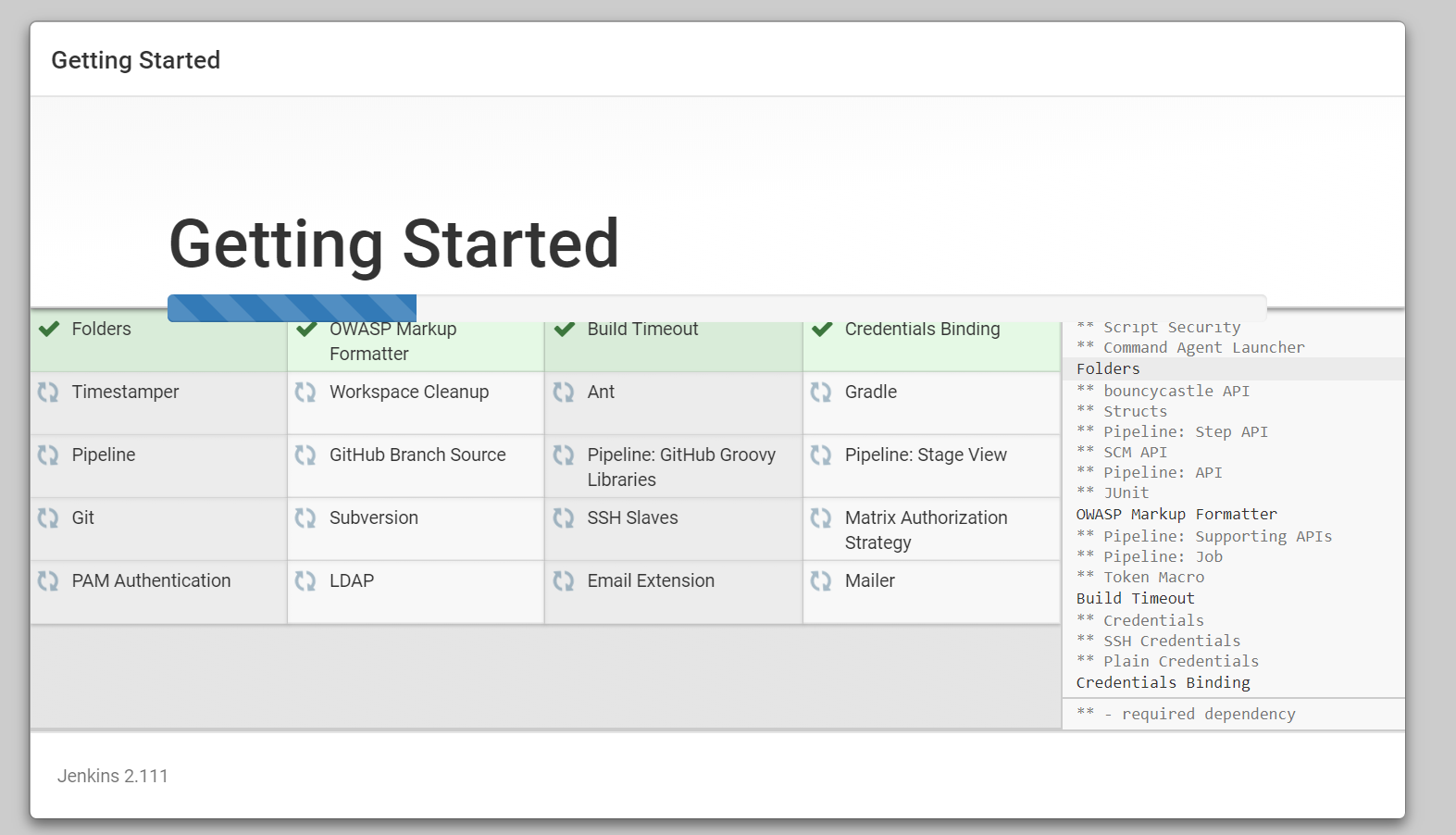


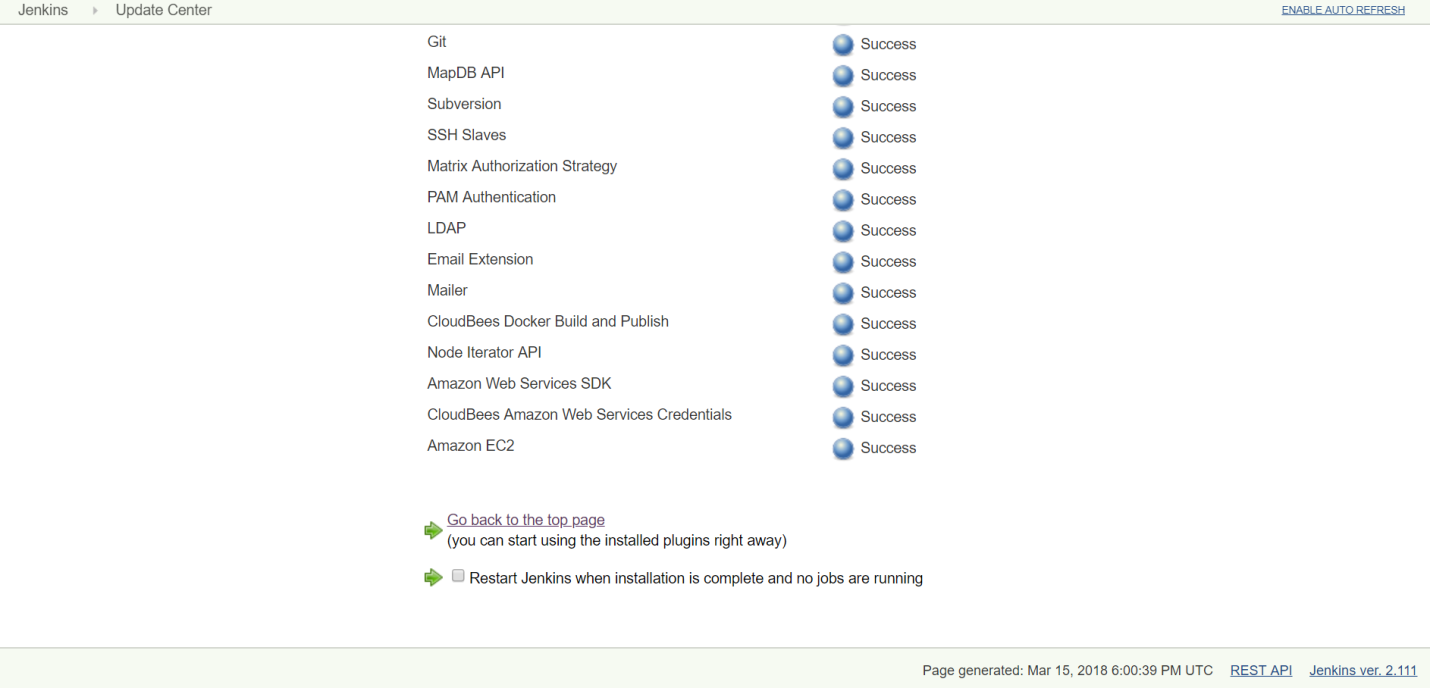


6. Next we create a Cloud formation stack. This stack will use the Jenkins template to create a VM and does the Jenkins installation and setup in it. Below is the command

*aws cloudformation create-stack --template-body file://ecs-jenkins-demo.template --stack-name JenkinsStack --capabilities CAPABILITY\_IAM --tags Key=Name,Value=Jenkins --region us-east-1 --parameters ParameterKey=EcsStackName,ParameterValue=EcsClusterStack*

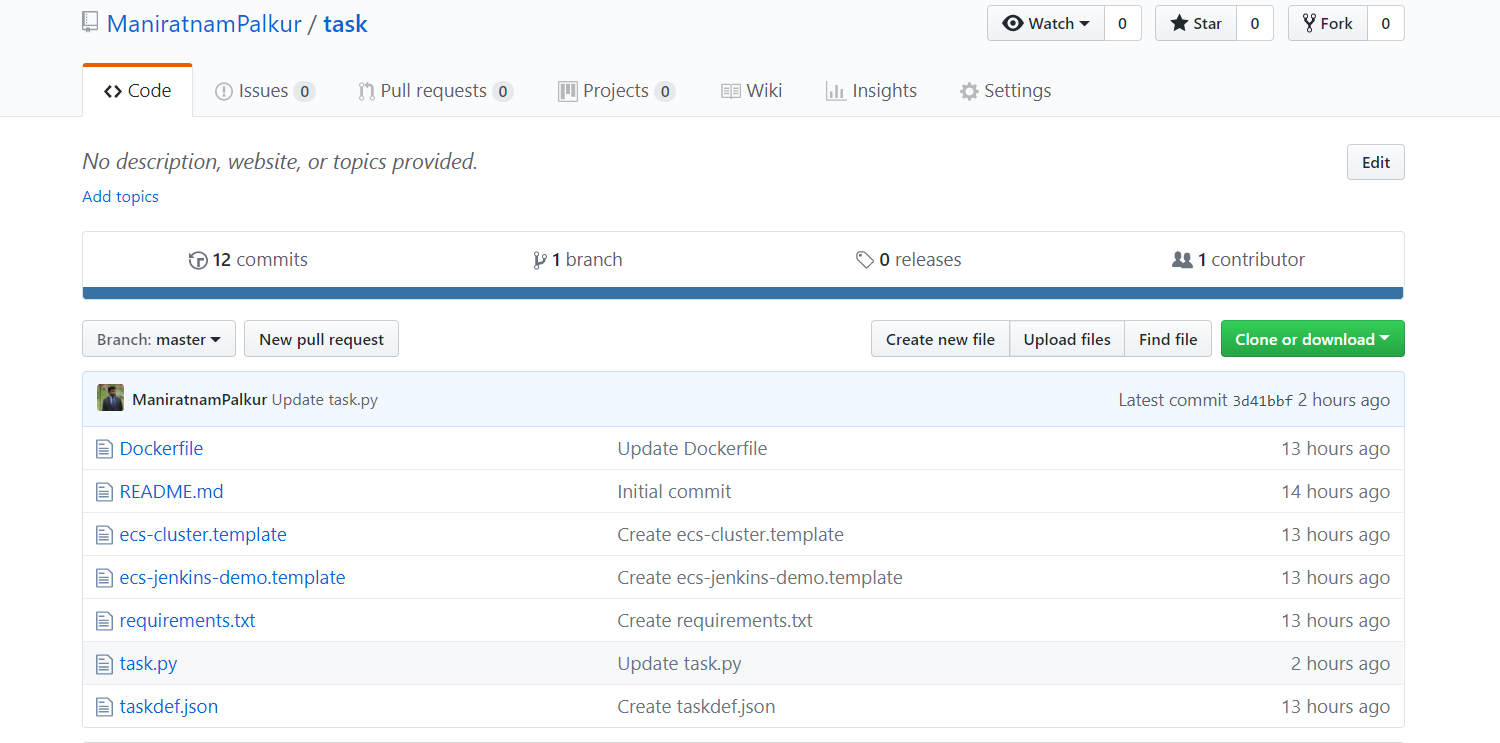
7. Once the stack is finished you will see new Jenkins EC2 instance.We can use the public-ip of it and open the Jenkins Dashboard in browser.Install the Amazon ECR and Cloudbees docker build and publish plugins in it.



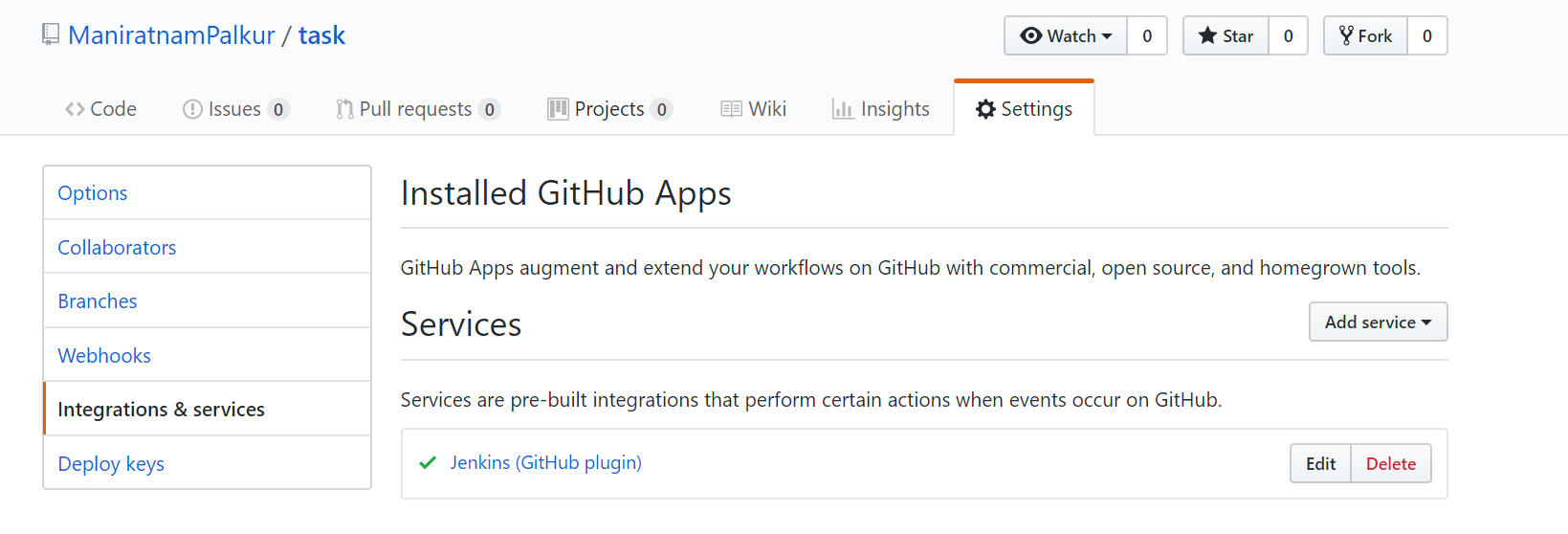


8. Create a Repository in GitHub and push the code into this repository

<https://github.com/ManiratnamPalkur/task.git>



9. Add Jenkins url into the Github repository as github-webhook



10. Create a Amazon ECR repository to store the docker images. Below is the command

*aws ecr create-repository --repository-name hello-world --region us-east-1*

11. Create a new Jenkins job with freestyle project

Select Git under SCM. Give the repository url and credentials

Select “GitHub hook trigger for GITScm polling”

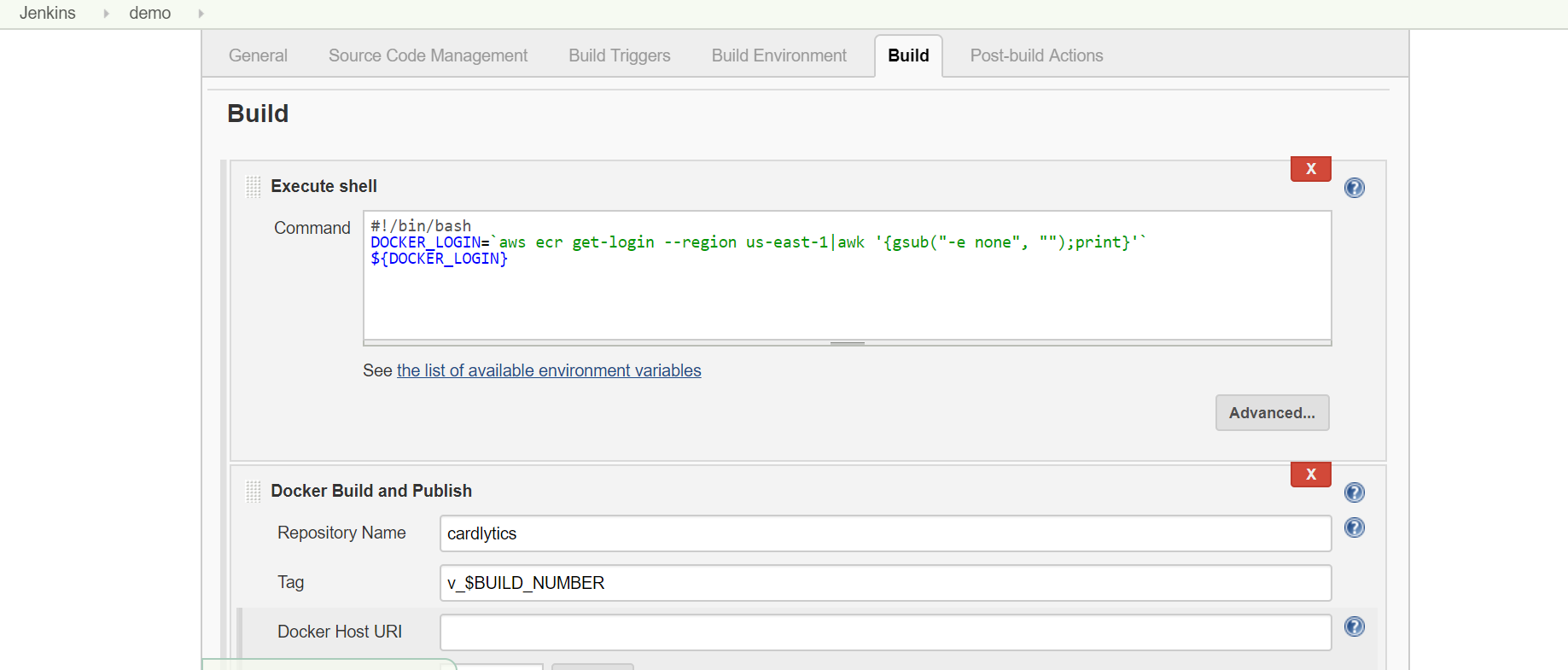
Under Build, select “Execute shell” and give below command

#!/bin/bash

DOCKER\_LOGIN=`aws ecr get-login --region us-east-1|awk '{gsub("-e none", "");print}'`

${DOCKER\_LOGIN}

This shell command will get the password to login to Docker



Next select “Docker Build and Publish” and give the required details in the field

*#!/bin/bash*

*#Constants*

*REGION=us-east-1*

*REPOSITORY\_NAME=cardylitcs*

*CLUSTER=test-cluster1*

*FAMILY=`sed -n 's/.\*"family": "\(.\*\)",/\1/p' taskdef.json`*

*NAME=`sed -n 's/.\*"name": "\(.\*\)",/\1/p' taskdef.json`*

*SERVICE\_NAME=${NAME}-service*

*#Store the repositoryUri as a variable*

*REPOSITORY\_URI=`aws ecr describe-repositories --repository-names ${REPOSITORY\_NAME} --region ${REGION} | jq .repositories[].repositoryUri | tr -d '"'`*

*#Replace the build number and respository URI placeholders with the constants above*

*sed -e "s;%BUILD\_NUMBER%;${BUILD\_NUMBER};g" -e "s;%REPOSITORY\_URI%;${REPOSITORY\_URI};g" taskdef.json > ${NAME}-v\_${BUILD\_NUMBER}.json*

*#Register the task definition in the repository*

*aws ecs register-task-definition --family ${FAMILY} --cli-input-json file://${WORKSPACE}/${NAME}-v\_${BUILD\_NUMBER}.json --region ${REGION}*

*SERVICES=`aws ecs describe-services --services ${SERVICE\_NAME} --cluster ${CLUSTER} --region ${REGION} | jq .failures[]`*

*#Get latest revision*

*REVISION=`aws ecs describe-task-definition --task-definition ${NAME} --region ${REGION} | jq .taskDefinition.revision`*

*#Create or update service*

*if [ "$SERVICES" == "" ]; then*

*echo "entered existing service"*

*DESIRED\_COUNT=`aws ecs describe-services --services ${SERVICE\_NAME} --cluster ${CLUSTER} --region ${REGION} | jq .services[].desiredCount`*

*if [ ${DESIRED\_COUNT} = "0" ]; then*

*DESIRED\_COUNT="1"*

*fi*

*aws ecs update-service --cluster ${CLUSTER} --region ${REGION} --service ${SERVICE\_NAME} --task-definition ${FAMILY}:${REVISION} --desired-count ${DESIRED\_COUNT} --force-new-deployment*

*else*

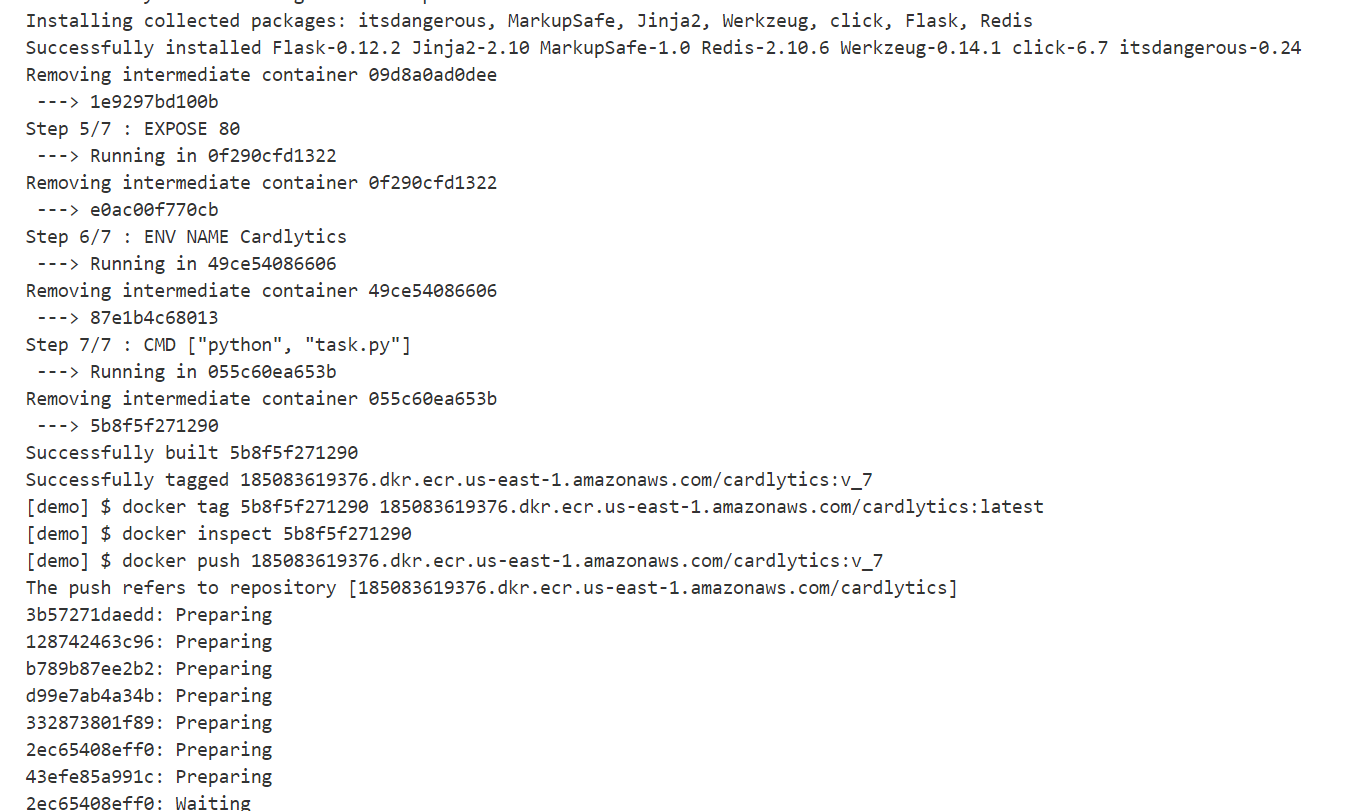
*echo "entered new service"*

*aws ecs create-service --service-name ${SERVICE\_NAME} --desired-count 1 --task-definition ${FAMILY} --cluster ${CLUSTER} --region ${REGION}*

*fi*

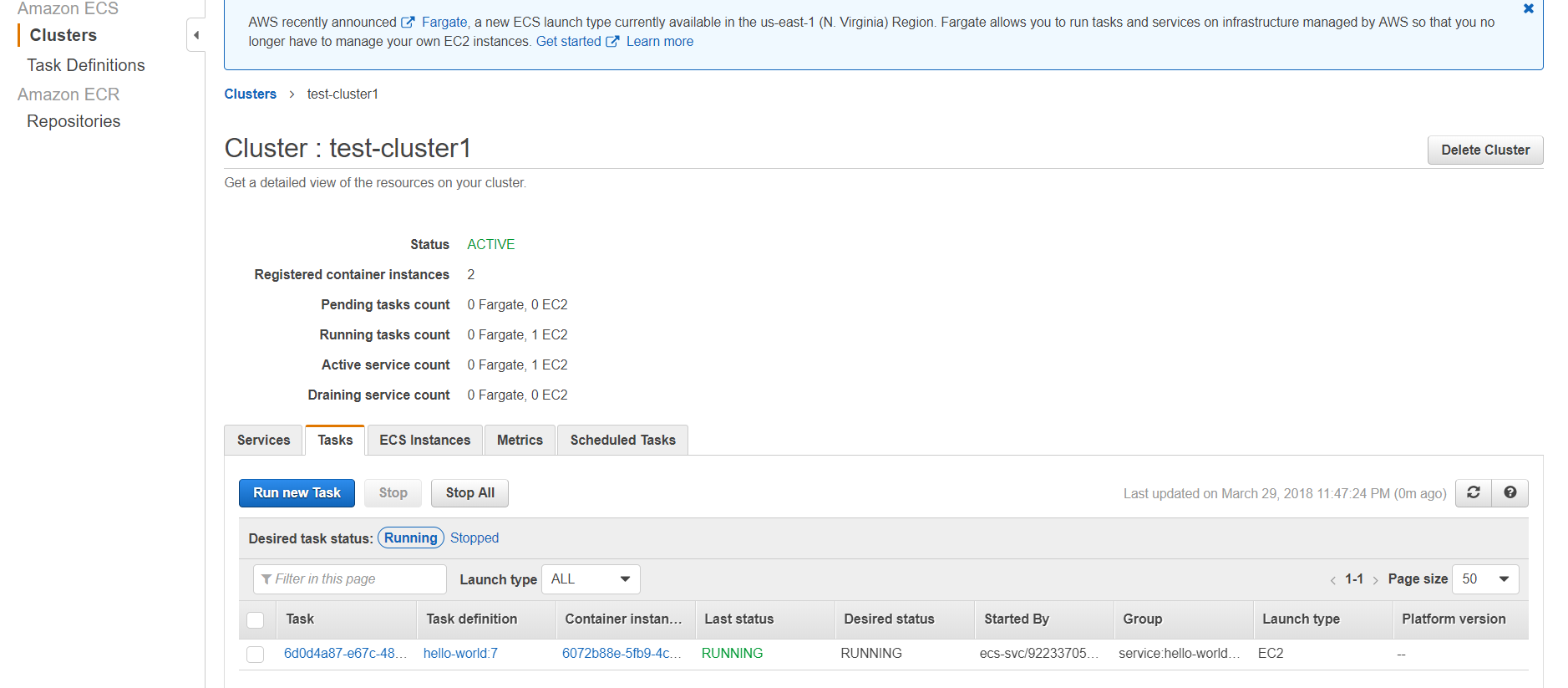
12. Next make a change to the source code and push the changes to GitHub. This will trigger the Jenkins job automatically. The job will invoke the Docker build and publish plugin which will build the docker image by using the Dockerfile.

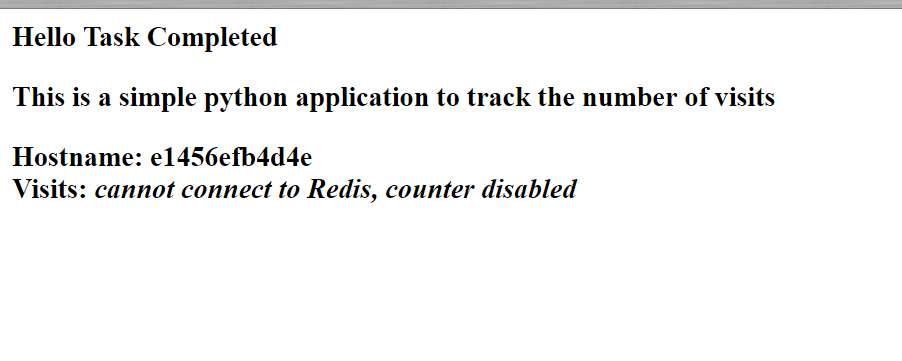
The Docker image will be pushed the Amazon ECR which we have created.



Next a service will be created in ECS cluster and the latest docker image will be pulled and deployed into container.

We can access the webpage in the service using the External link ip-address of the task.





Reference: <https://docs.aws.amazon.com/AWSGettingStartedContinuousDeliveryPipeline/latest/GettingStarted/CICD_Jenkins_Pipeline.html>