Contents

[Introduction and Code Base Hierarchy 1](#_Toc107913668)

[Steps Required 1](#_Toc107913669)

[Results 7](#_Toc107913670)

# Introduction and Code Base Hierarchy

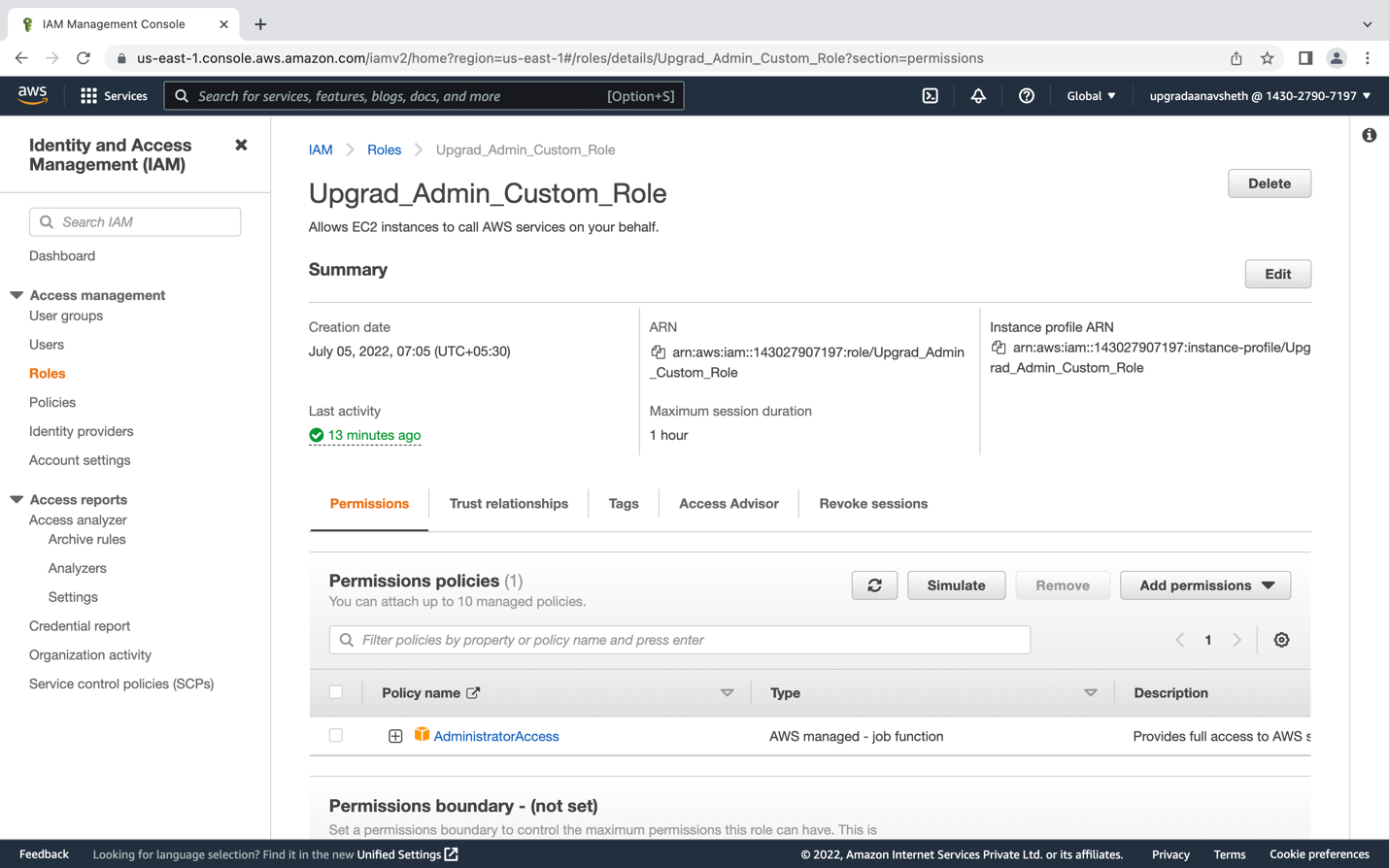
S3 bucket needs to be created manually by CLI. Terraform code sets S3 Bucket as backend. The terraform codebase is standardized and decoupled such as separate TF files for each EC2 instance using Modules, Elastic IP, each Security Group, ALB, VPC. Please note that ALB is created by the code as well.

“ansible-playbook” directory contains inventory and ProxyJump Config file auto-generated by “null\_resource” provisioner with local-exec. It contains “setup\_ec2\_instances.yml” playbook that setup Jenkins and App Instances. It even installs Jenkins and configures it to run on “/jenkins”. It also creates an ssh-keygen for root user on Jenkins and copies the private key to APP EC2 so that Jenkins can SSH App server.

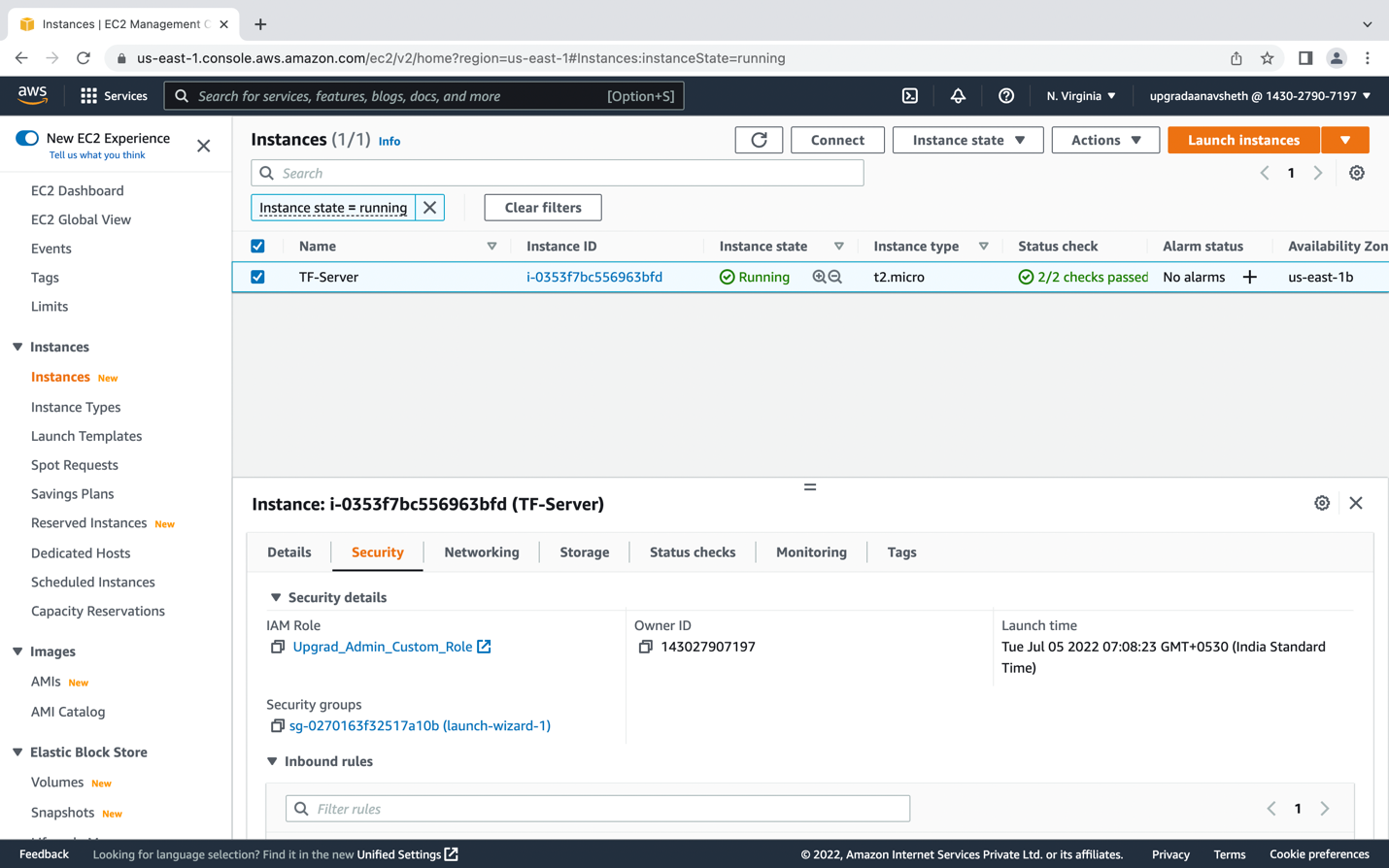
Terraform Code : - <https://github.com/ms-sourcetech/terraform-ansible-jenkins.git>

# Steps Required

* Create custom role having “AdministratorAccess”. Attach it to the local machine which is also an EC2 instance but in default VPC.



* Attach the role to EC2 instance



* Create S3 bucket named “manav-assignment-backend” using the below CLI Command.

aws s3api create-bucket --object-lock-enabled-for-bucket --bucket manav-assignment-backend

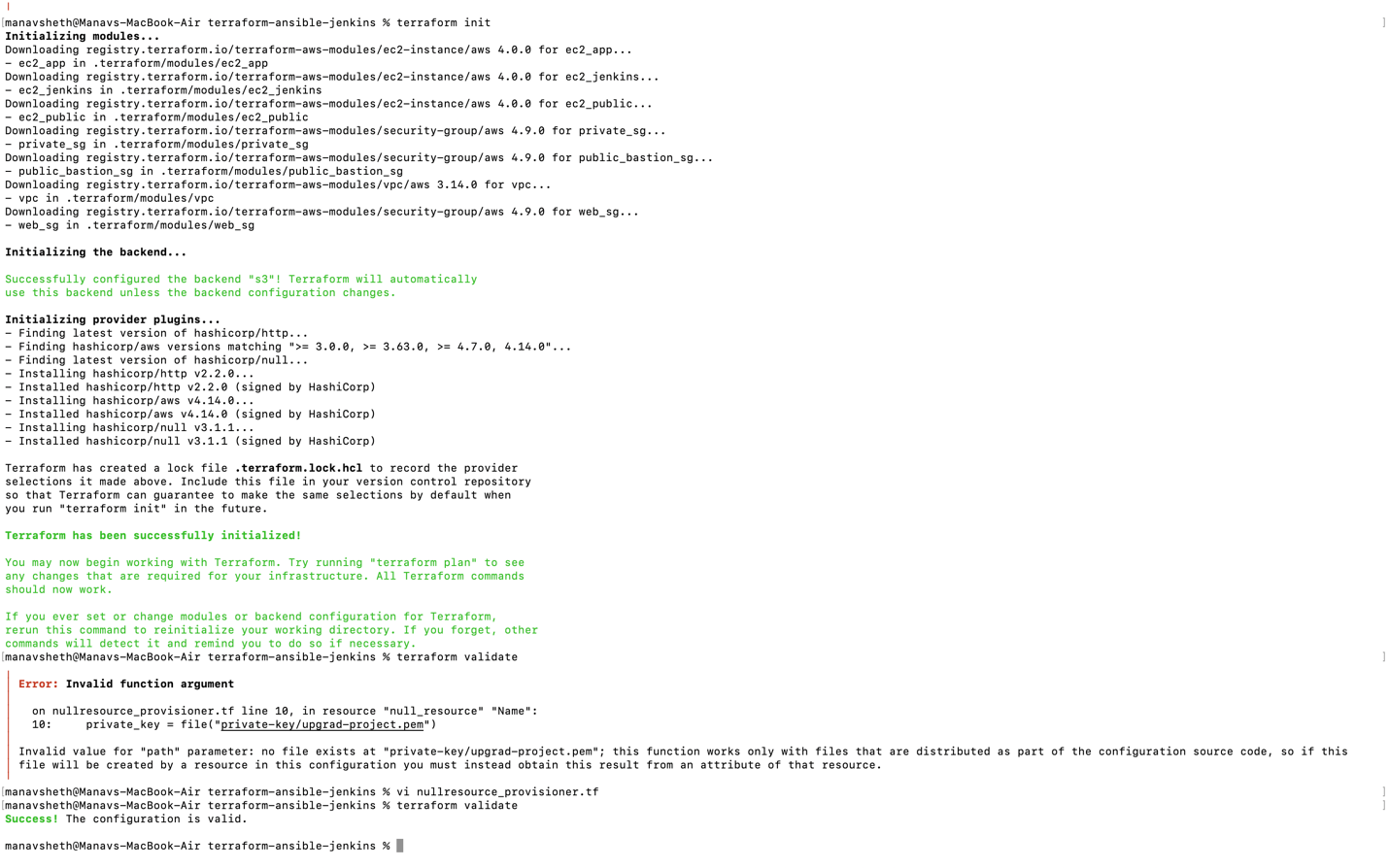
Text, letter

Description automatically generated

* Clone the repository from github which is having terraform and ansible codebase.

git clone <https://github.com/ms-sourcetech/terraform-ansible-jenkins.git>

* Change directory to assignment “cd terraform-ansible-jenkins”
* Initialize terraform code using “terraform init”



* Validate Terraform code

Text

Description automatically generated

* Apply the terraform config by “terraform apply” and then enter “yes”

Text

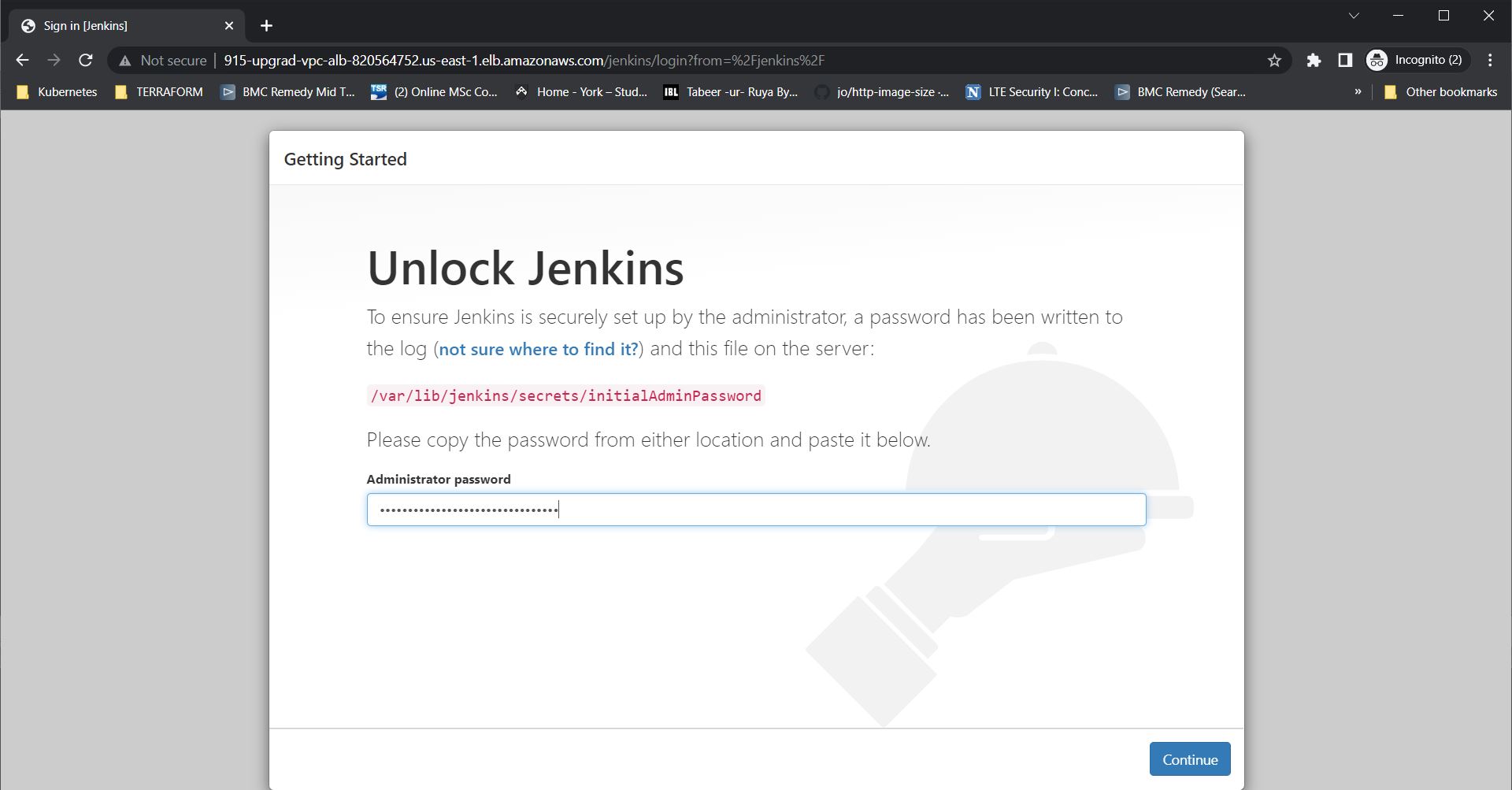
Description automatically generated with medium confidence

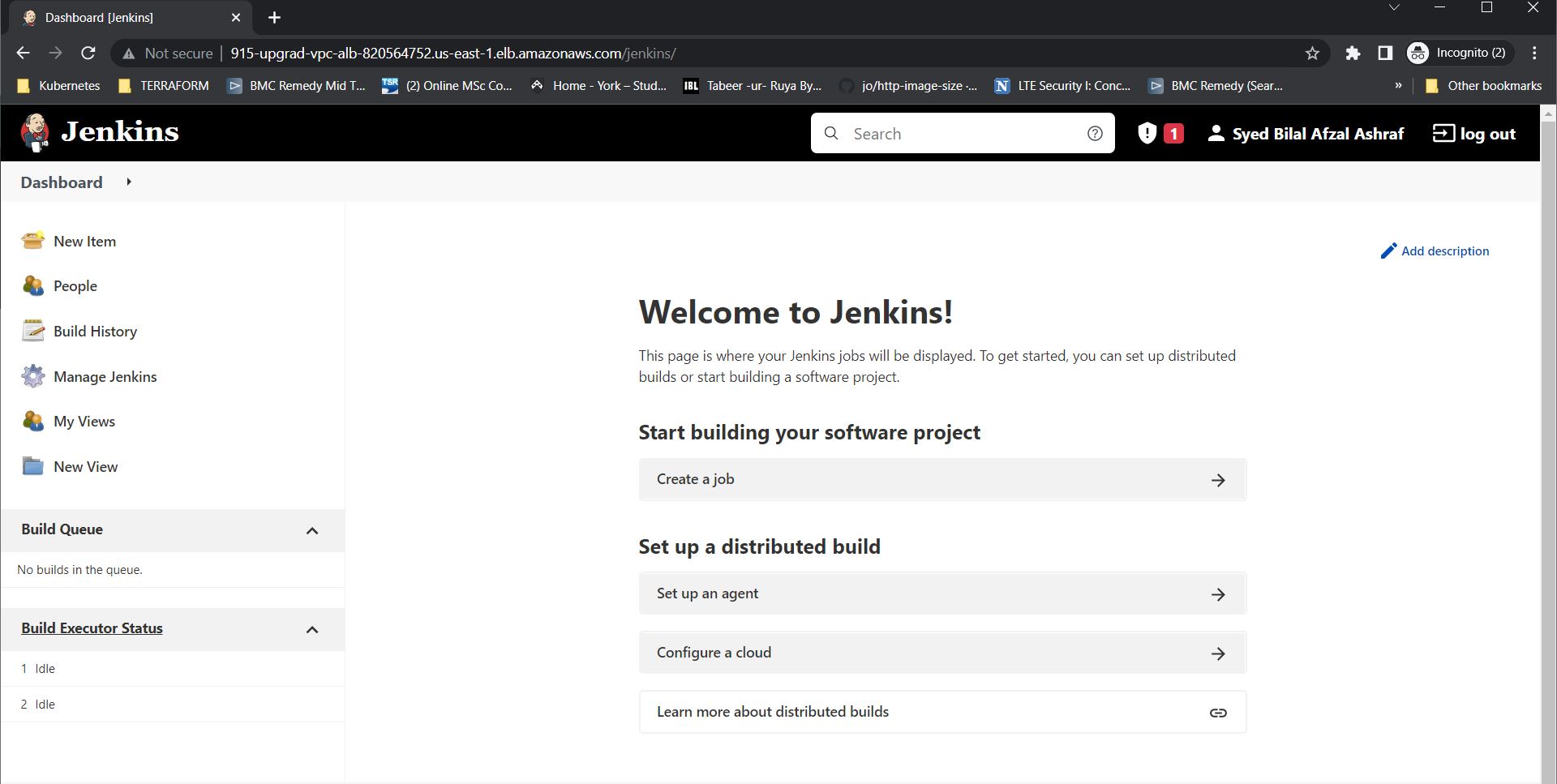
* Change directory to “ansible-playbook”. This directory is having terraform generated Ansible Inventory and SSH ProxyJump (config) files. This directory is also having EC2 Instances Private Key, Ansible config and Playbook files.
* Use playbook “setup\_ec2\_instances.yml” to install docker on both Jenkins and App EC2 instances. It also changes the hostnames to “jenkins” and “app”. It also creates /etc/hosts files. It creates SSH Keypair and copy public key to App EC2, so that Jenkins can SSH to App Instance. This playbook configures Jenkins prefix as “/jenkins”
* Create a custom role with policy “EC2InstanceProfileForImageBuilderECRContainerBuilds” and assign it to “jenkins” and “app” instances.

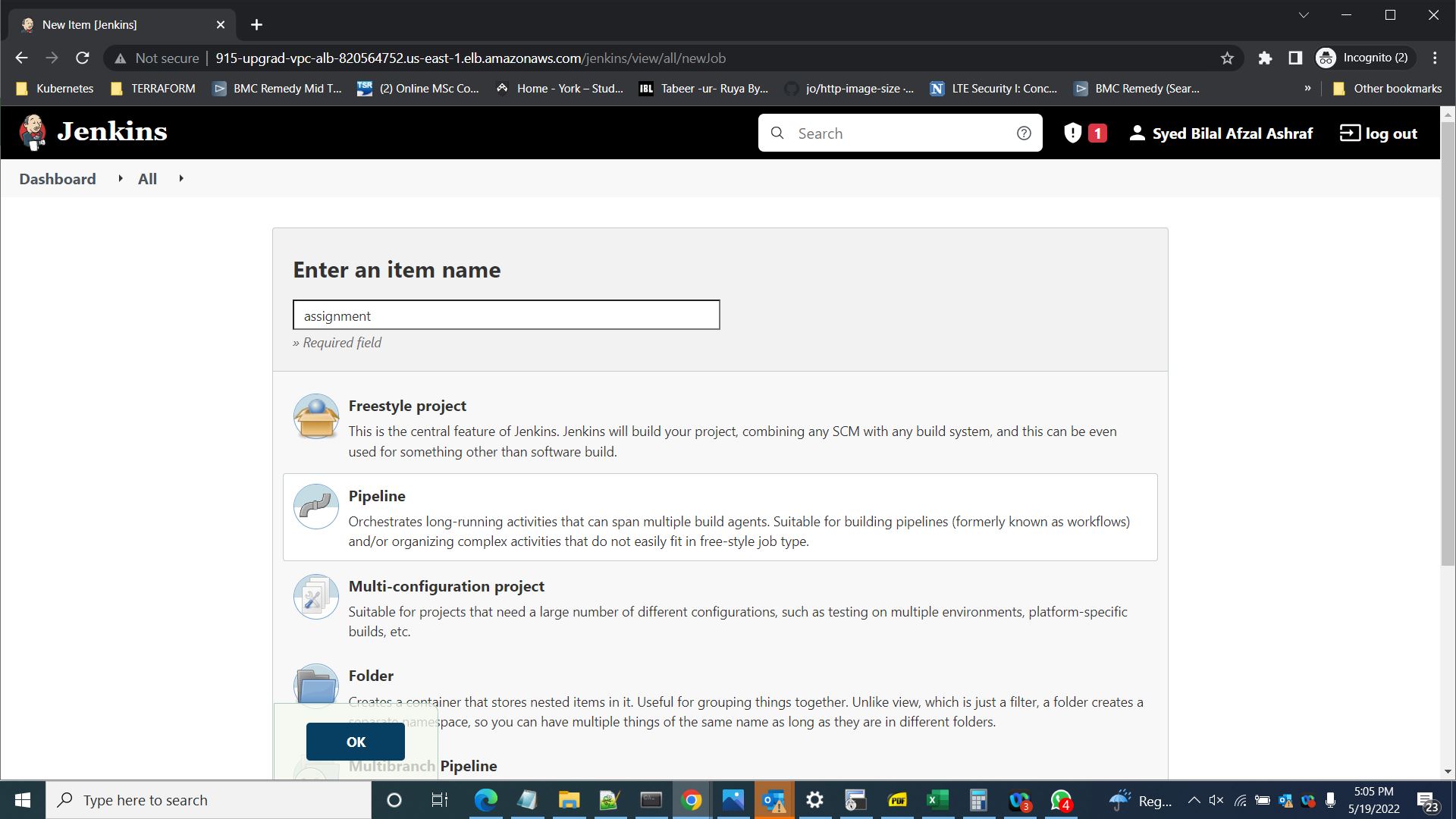
Graphical user interface, text, application

Description automatically generated

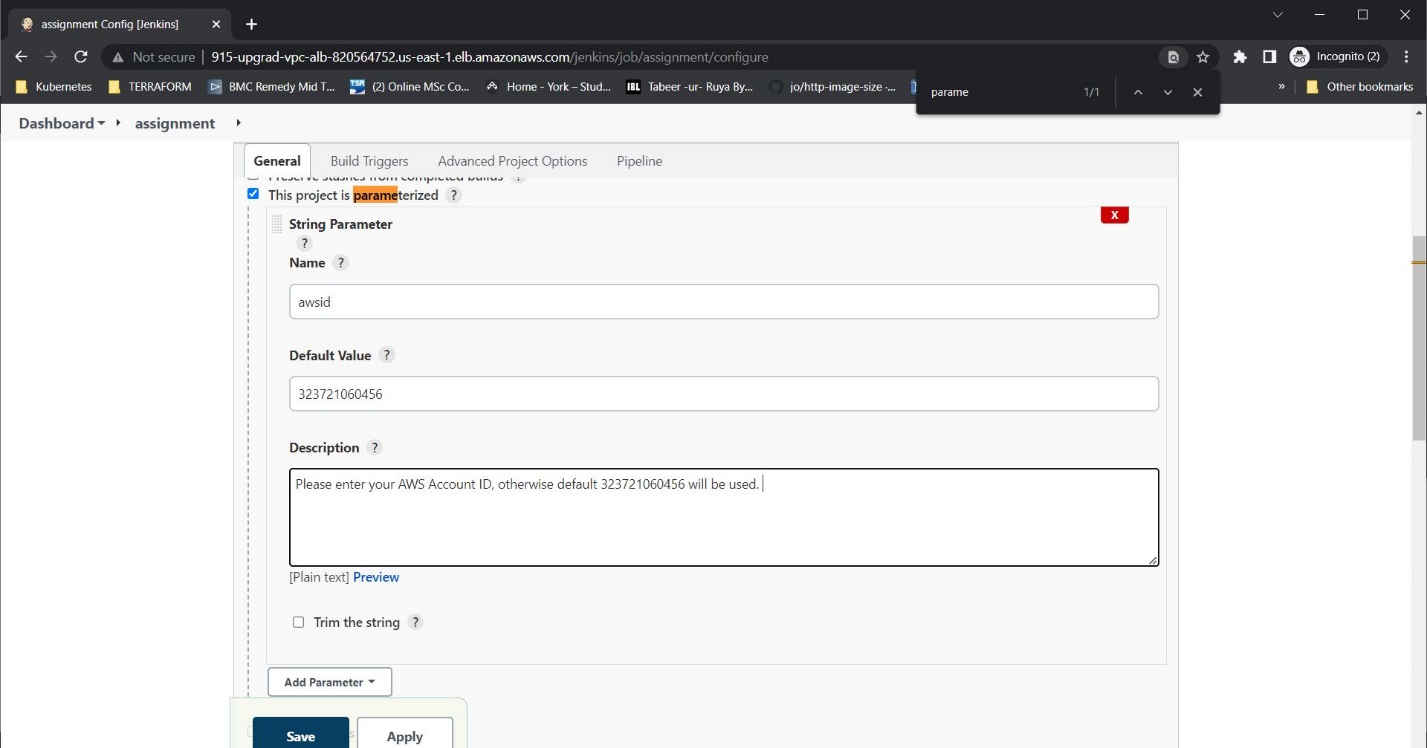
* Get Application Load Balancer Public DNS IP
* Use the ALB DNS URL and open it on browser. Get the initial password using next steps.

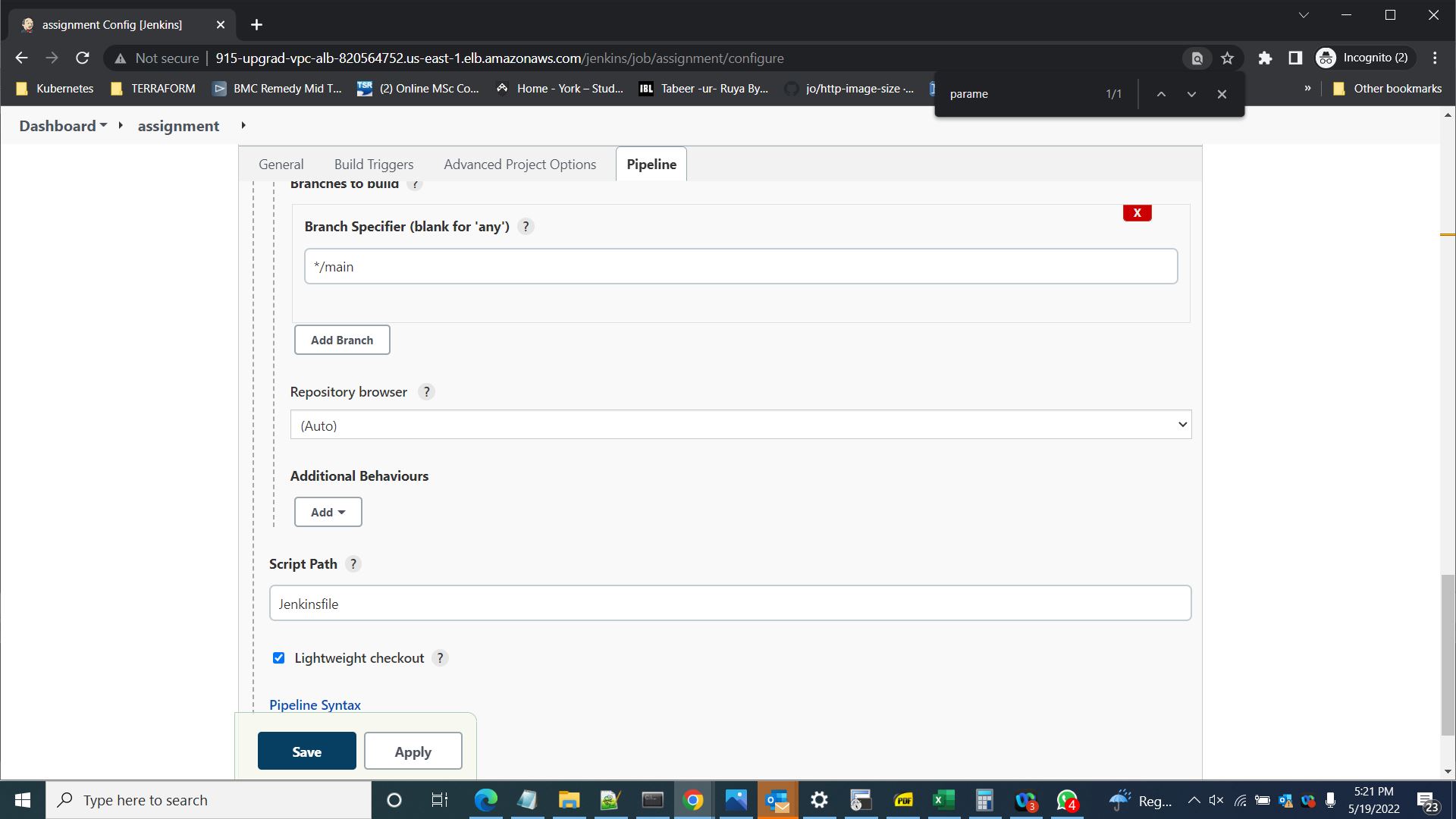


* Now SSH to Jenkins EC2 to get password using ssh config file generated by terraform.
* Get Jenkins initial admin password, copy the password.
* Click “New Item.
* Enter name and select “pipeline” then click OK.



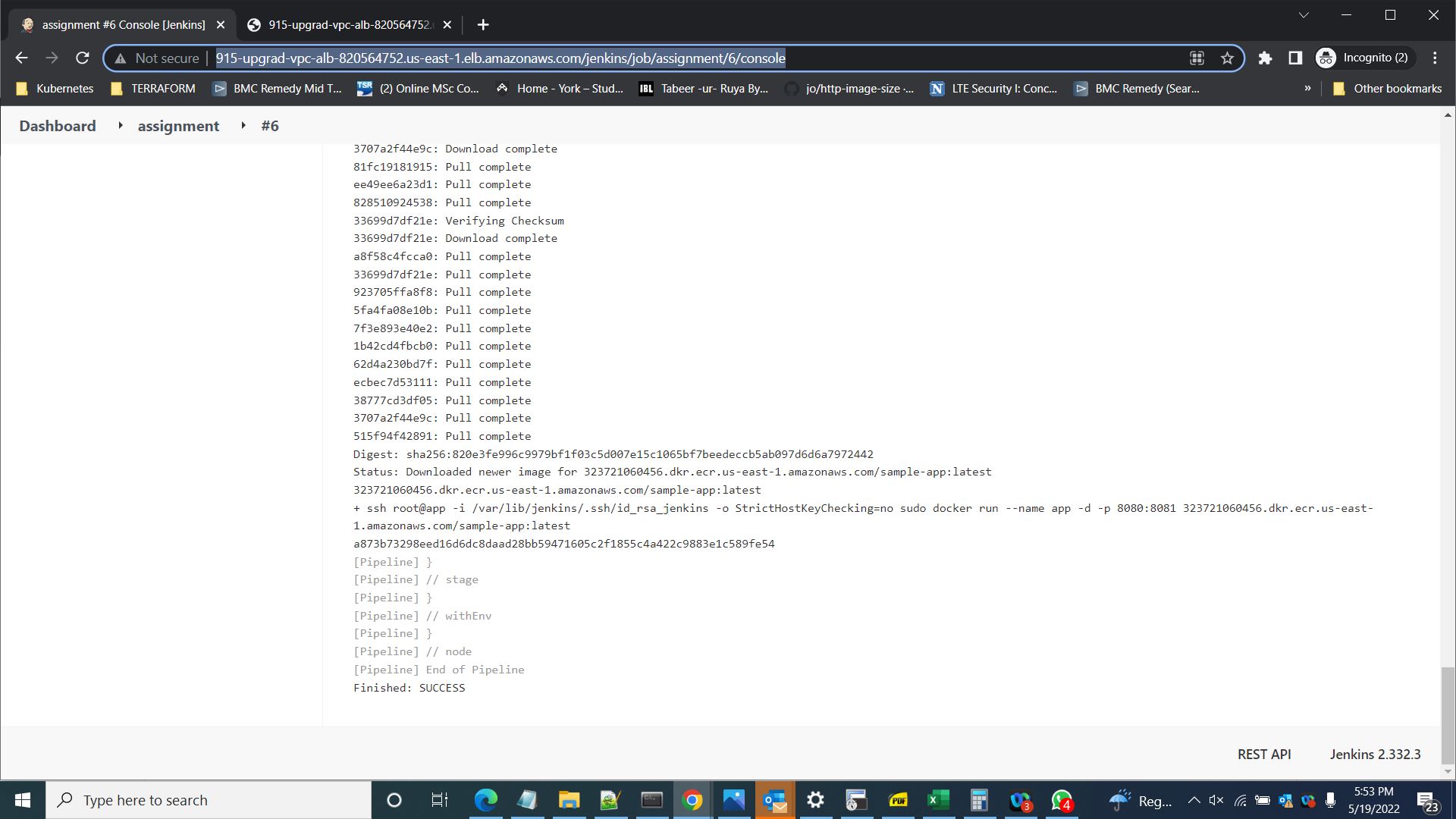
* Select “This project is parameterized”. Enter “awsid” under name box.



* Get the AWS account ID to set the default value of “awsid” parameter.
* Under Pipeline Section, select “Pipeline script from SCM”. Select “Git” under SCM. Enter the repository URL <https://github.com/ms-sourcetech/terraform-ansible-jenkins.git> . The repo is public hence no need to add credentials.
* Scroll down and enter “\*/main” under “Branch Specifier”. Also enter “Jenkinsfile” under Script Path. Click Save. 
* Go to dashboard and select the created pipeline. Then click “build with parameters”. It will show default AWS account ID, change the ID to your own AWS account ID and then click build.

# Results

* Console Output



* Successful APP URL. http://<alb-dns>:80/app URL.
* Use “terraform destroy” to destroy the infrastructure. Enter “yes” to approve.