**Deploy applications on AWS EKS**

[**1.** **Prerequisites** 2](#_Toc175561860)

[**2.** **Setup infrastructure with Terraform** 2](#_Toc175561861)

[**2.1. Set up AWS credential configure** 2](#_Toc175561862)

[**2.2. Initialize the module** 2](#_Toc175561863)

[**2.3. Verify the terraform plan** 4](#_Toc175561864)

[**2.3.1. Verify Cluster using kubectl** 4](#_Toc175561865)

[**2.3.2. Verify on AWS UI** 4](#_Toc175561866)

[**3.** **Setup Jenkins server with Terraform** 7](#_Toc175561867)

[**3.1. Initialize the module** 7](#_Toc175561868)

[**3.2. Verify the terraform plan** 8](#_Toc175561869)

[**3.3. Setup Jenkins account** 9](#_Toc175561870)

[**4.** **Setup Jenkins pipeline (Push images to AWS ACR)** 10](#_Toc175561871)

[**5.** **EKS Setup** 14](#_Toc175561872)

[**6.** **Monitoring with Prometheus and Grafana** 14](#_Toc175561873)

1. **Prerequisites**

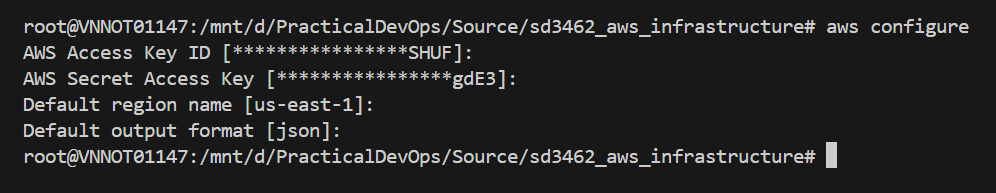
* AWS account (IAM user)
* AWS CLI
* Terraform
* Docker
* Kubectl
* Eksctl
* Helm

1. **Setup infrastructure with Terraform**

**2.1. Set up AWS credential configure**

**-** Run the command to set up your AWS CLI: aws configure

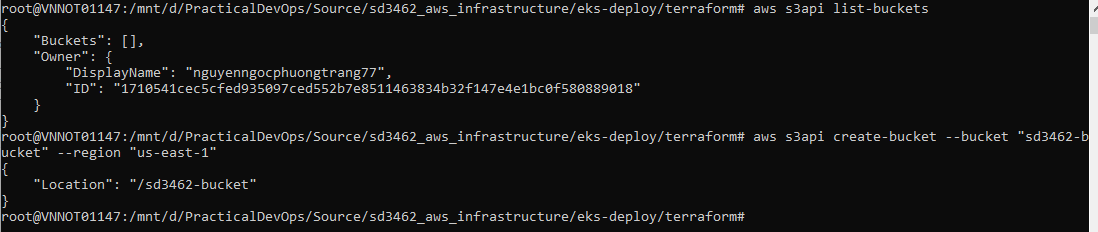
**-** Enter the access key of your IAM account, if you already configured press enter.

****

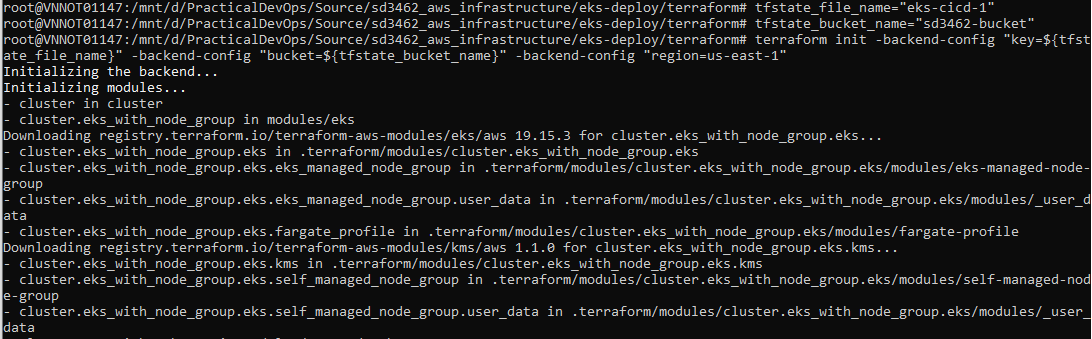
**2.2. Initialize the module**

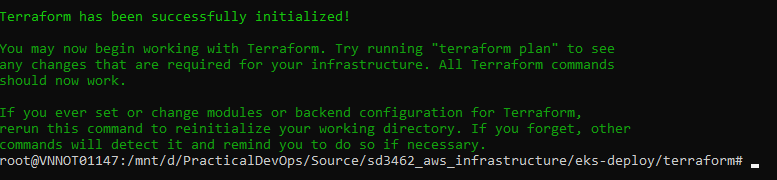
**-** Move to the **“./eks-deploy/terraform”** directory to perform commands.

**-** Create the s3 bucket to store the *tfstate* file.

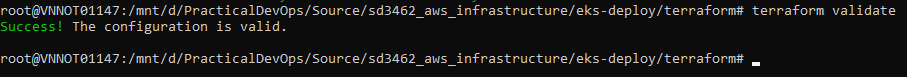


**-** Run the command terraform init -backend-config "key=${tfstate\_file\_name}" -backend-config "bucket=${tfstate\_bucket\_name}" -backend-config "region=us-east-1" to initializing the backend.

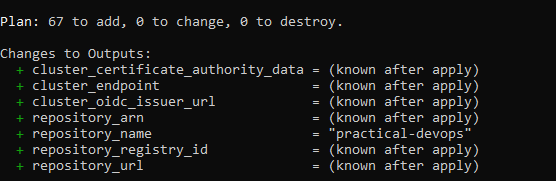




**-** After successfully initializing, run terraform validate to check whether the configuration is valid to ensure no issue when applying the terraform plan.



**-** Run terraform plan -var-file=”dev.tfvars” to apply the execution plan.



- Run terraform apply –var-file=”dev.tfvars” to create all the resource.



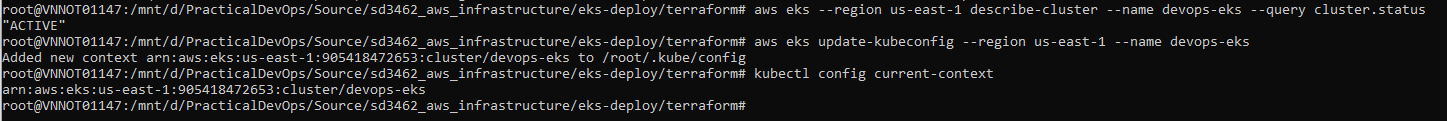
**2.3. Verify the terraform plan**

**2.3.1. Verify Cluster using kubectl**

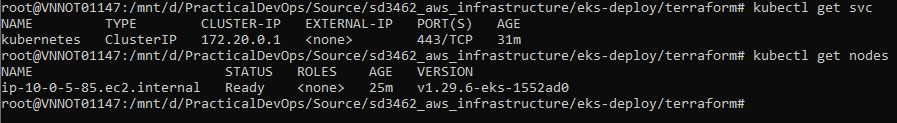
- Check status of the Cluster aws eks --region us-east-1 describe-cluster --name devops-eks --query cluster.status.

- Configure kubectl with EKS API server credentials aws eks update-kubeconfig --region us-east-1 --name devops-eks.

- Check if you are pointing to the right kubernetes cluster kubectl config current-context

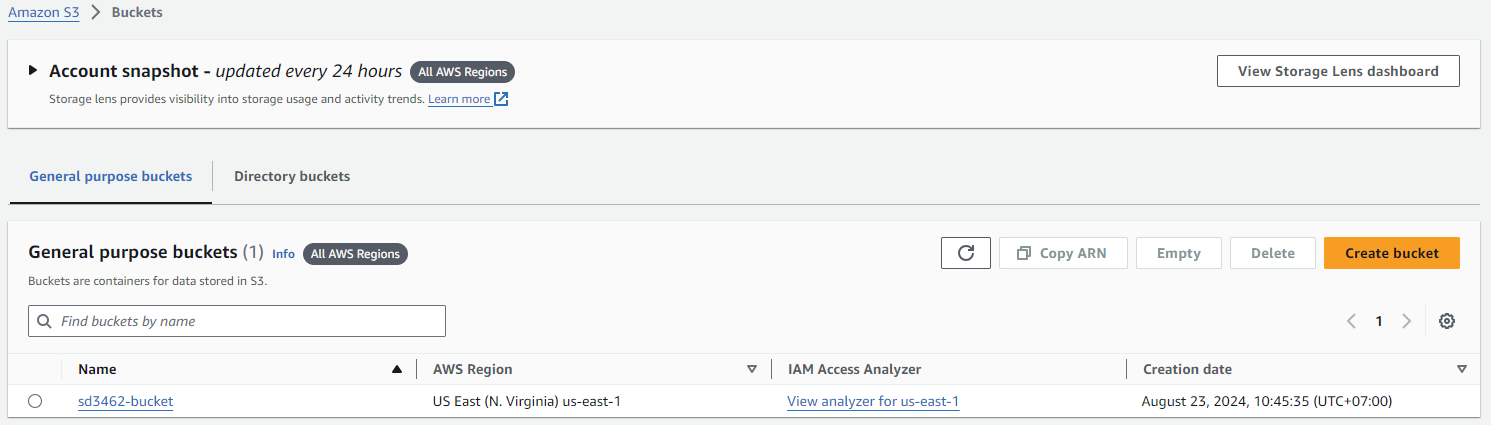


- Validate kubectl configuration master node.

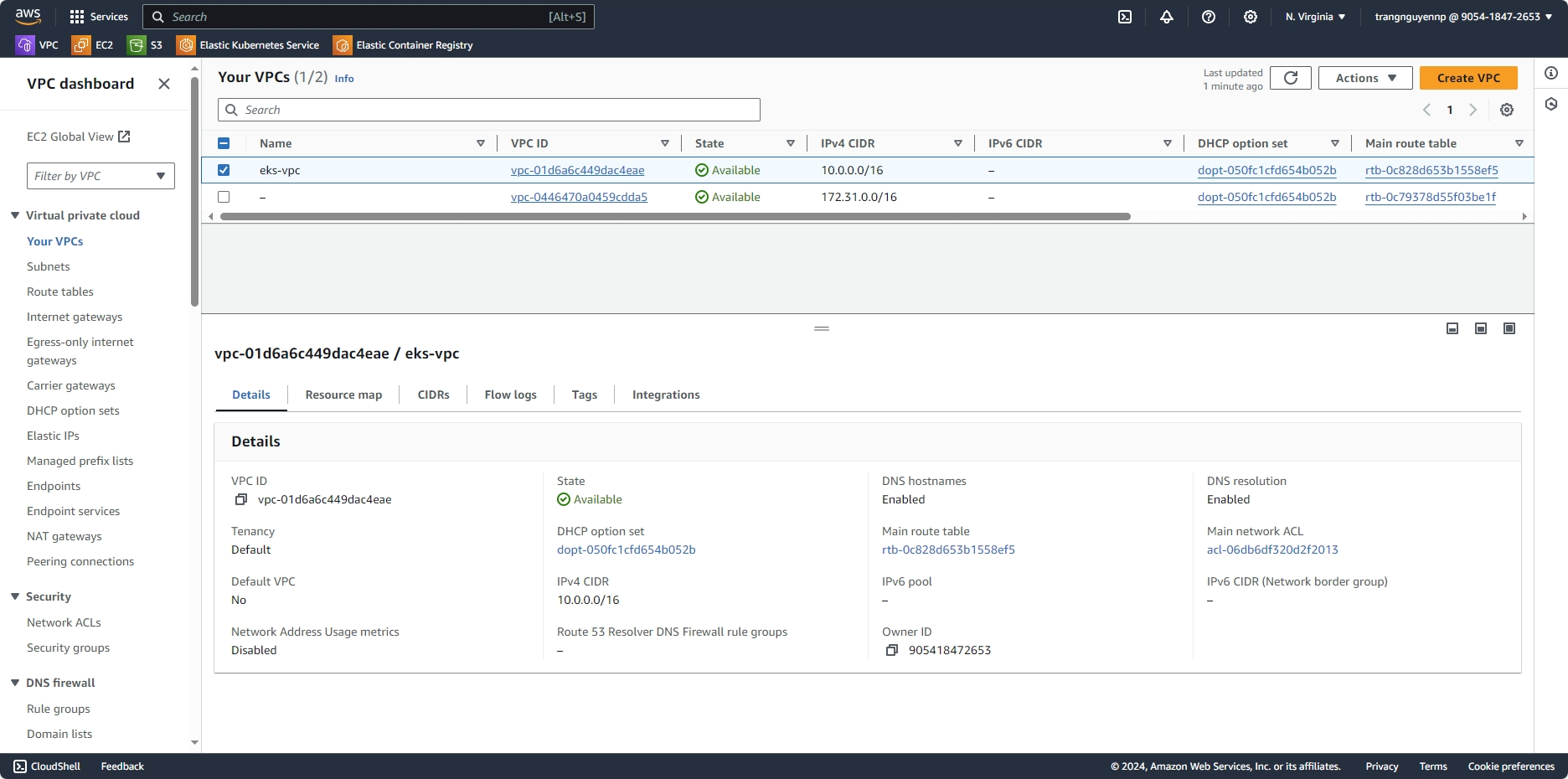


**2.3.2. Verify on AWS UI**

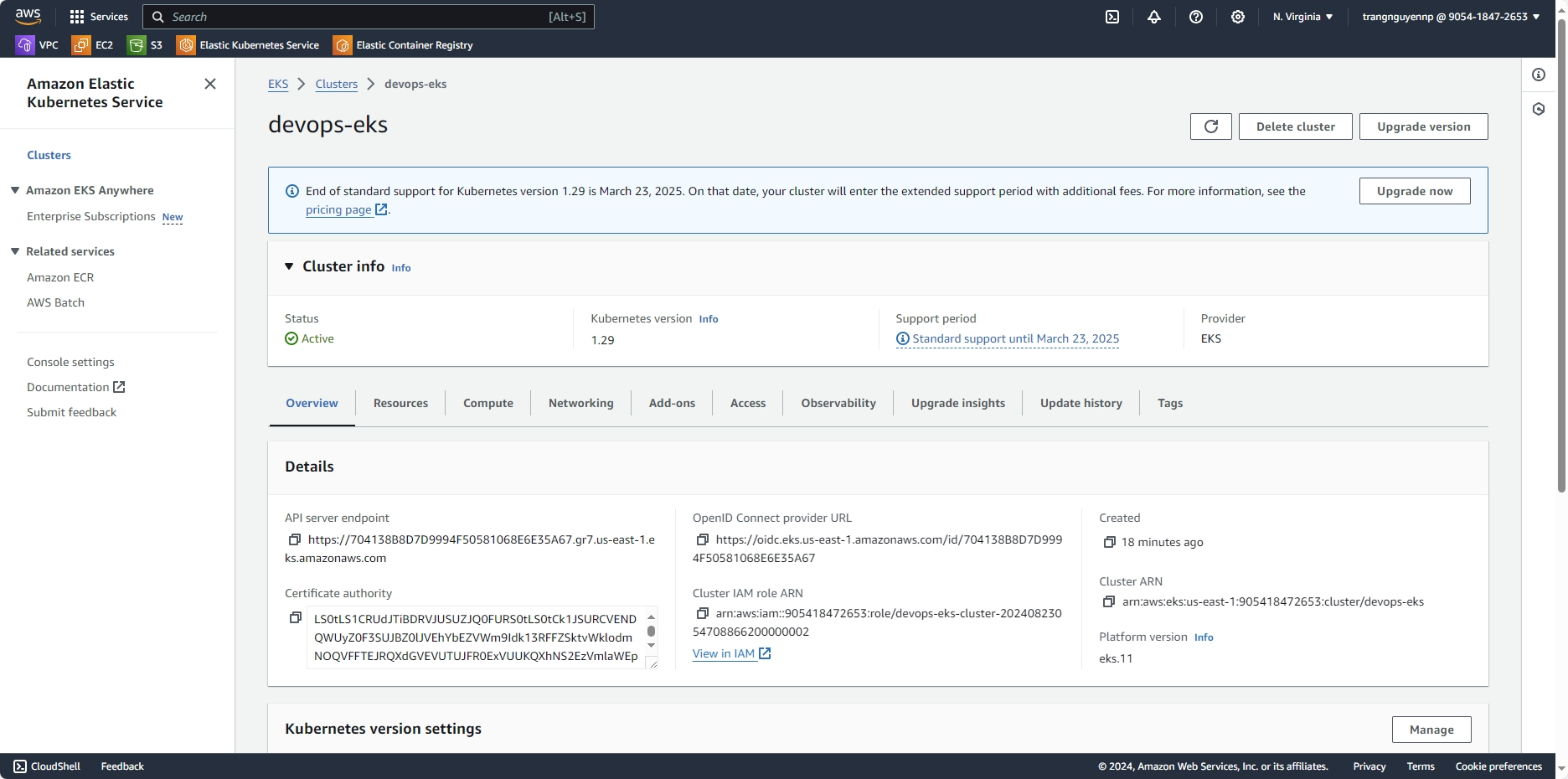
* S3 bucket.



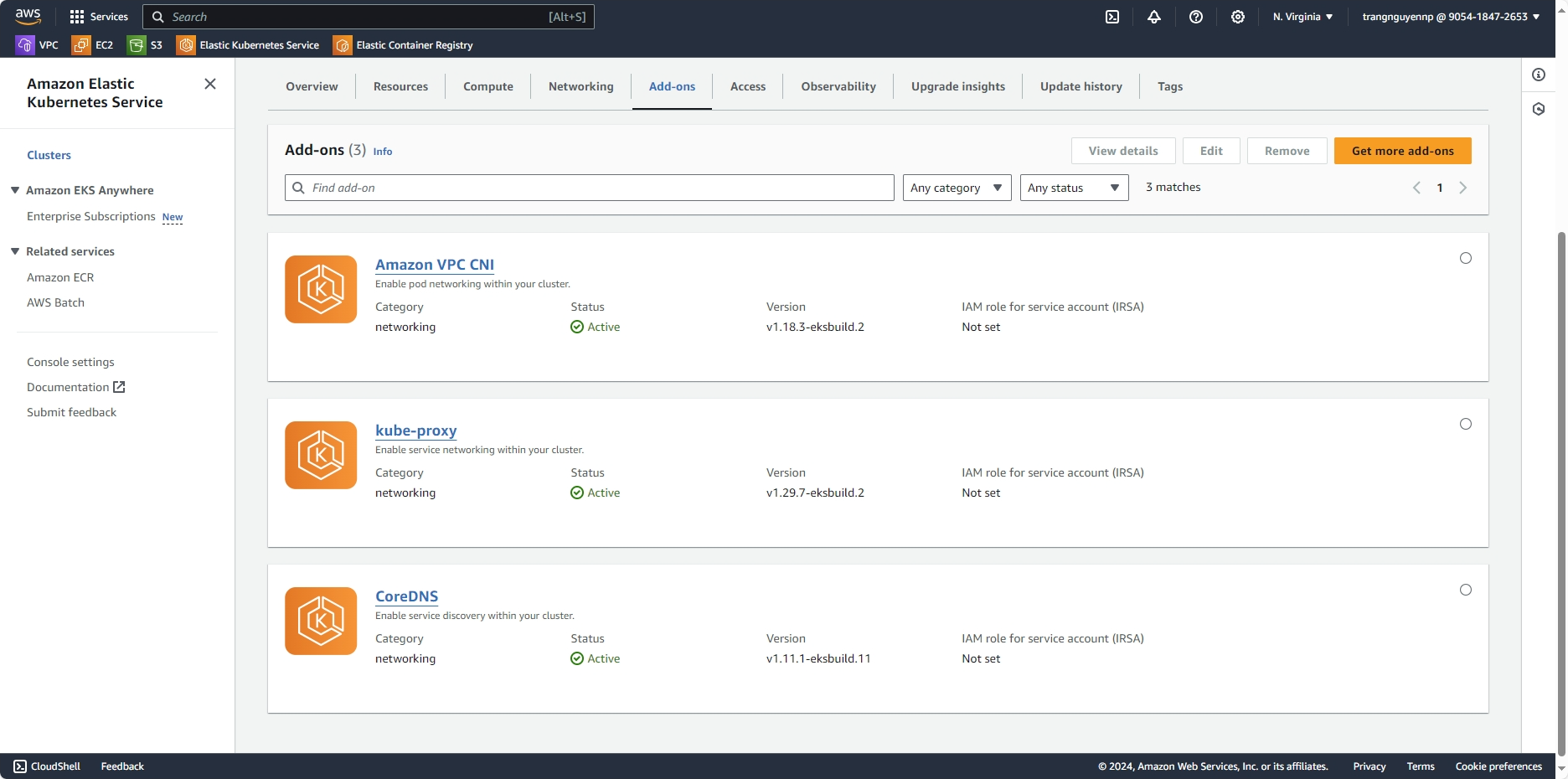
* Virtual private cloud.



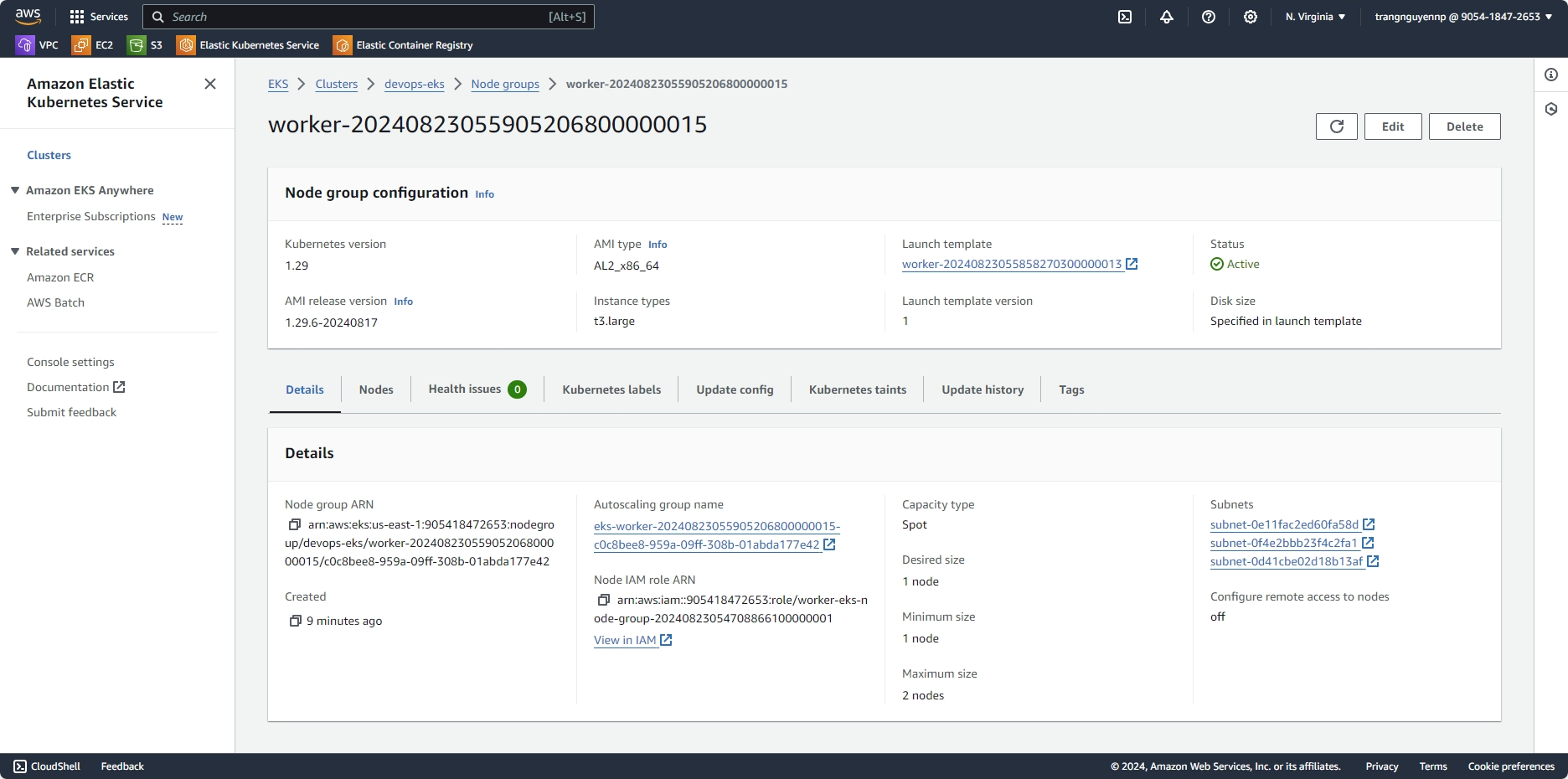
* EKS Cluster with OIDC Provider.



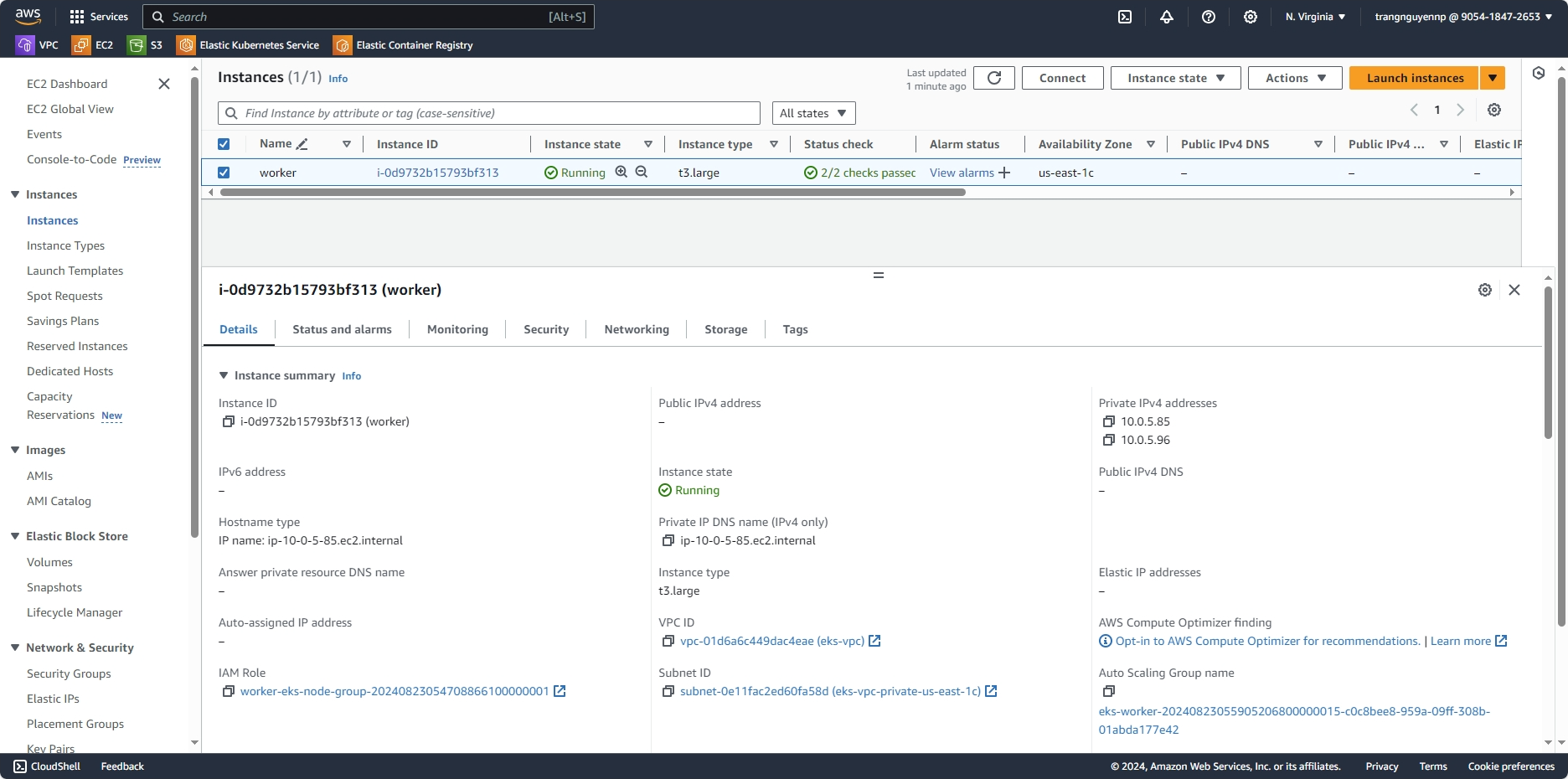
* EKS Managed AddOns.



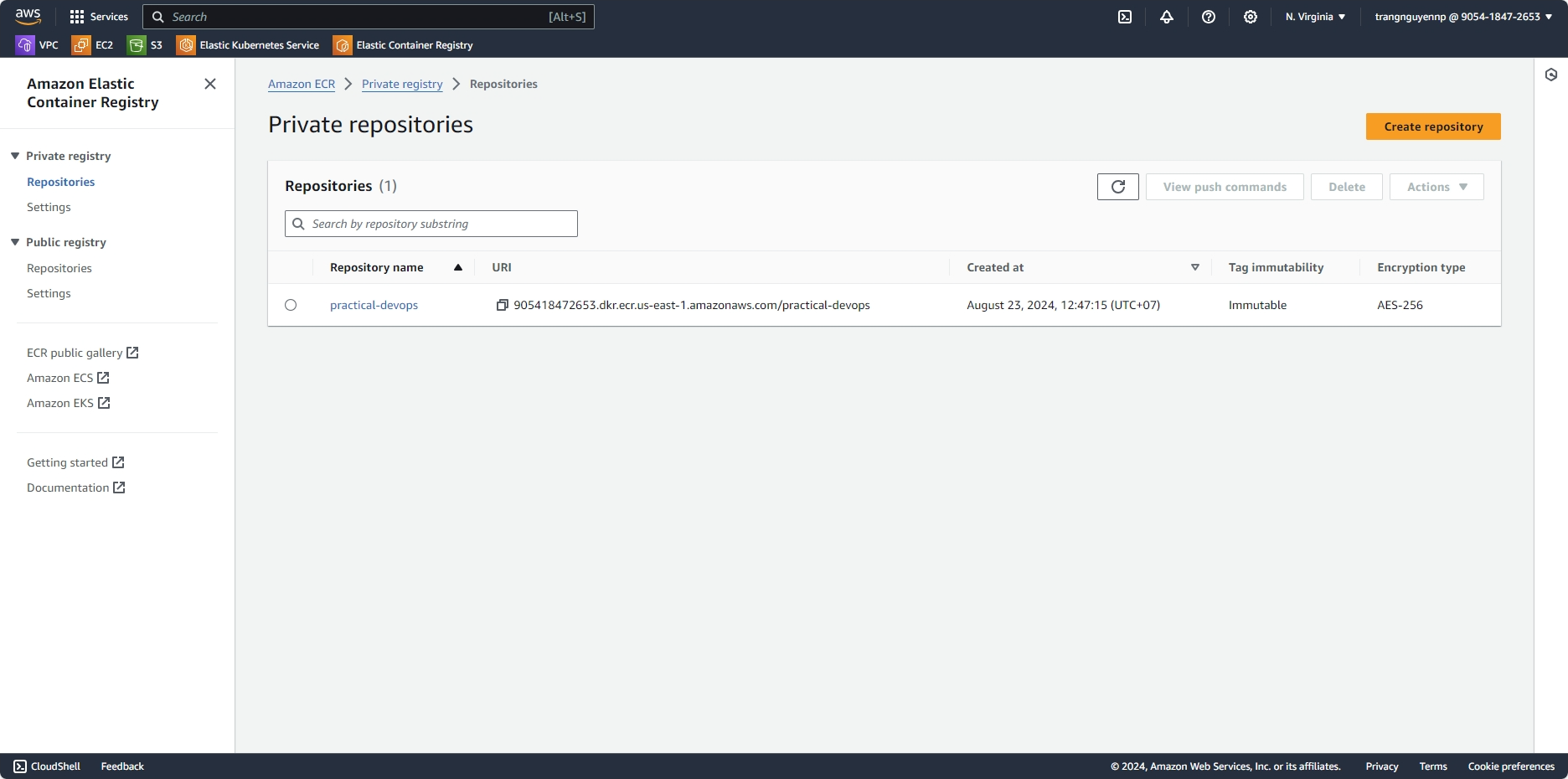
* EKS Managed node group.



* EC2: worker nodes



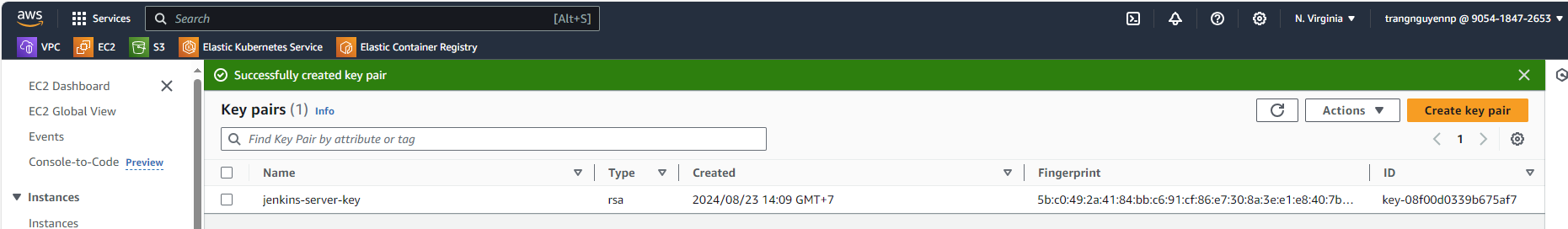
* ECR repositories



# **Setup Jenkins server with Terraform**

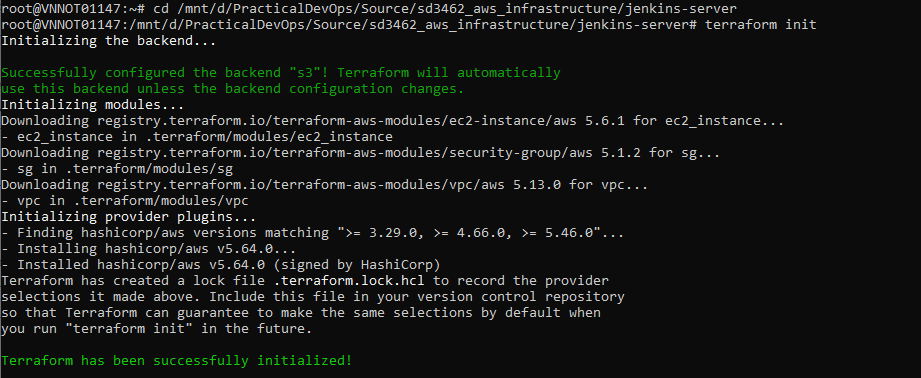
## **3.1. Initialize the module**

- Go to AWS UI page, create new key pair name **“jenkins-server-key”** in EC2 page.

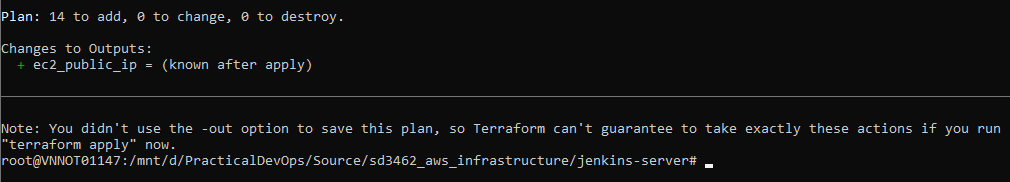


- Move to the **“./jenkins-server”** directory to perform commands.

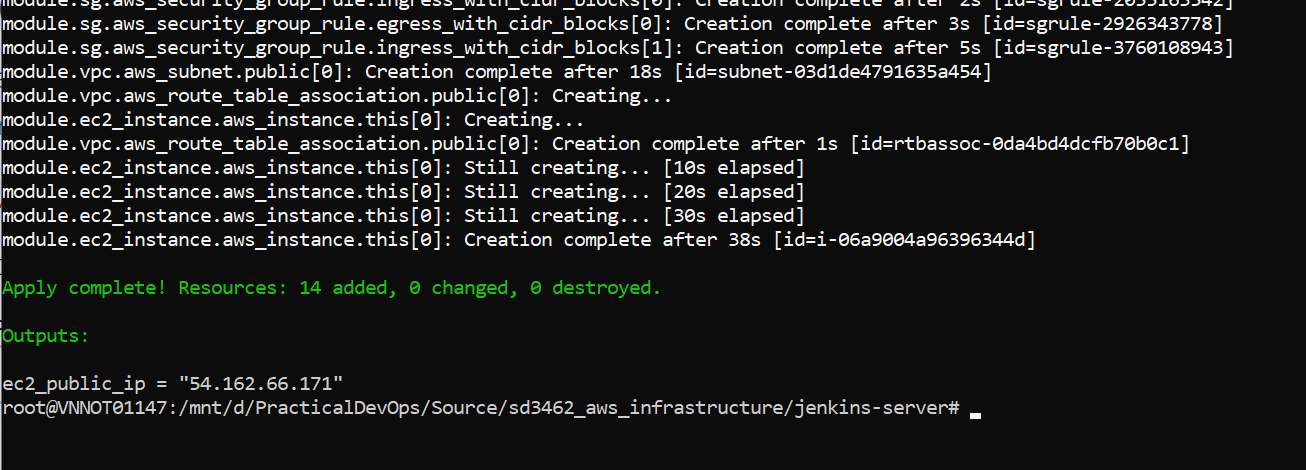
- Run the command terraform init to initialize backend.



- After successfully initializing, run terraform validate to check whether the configuration is valid to ensure no issue when applying the terraform plan.

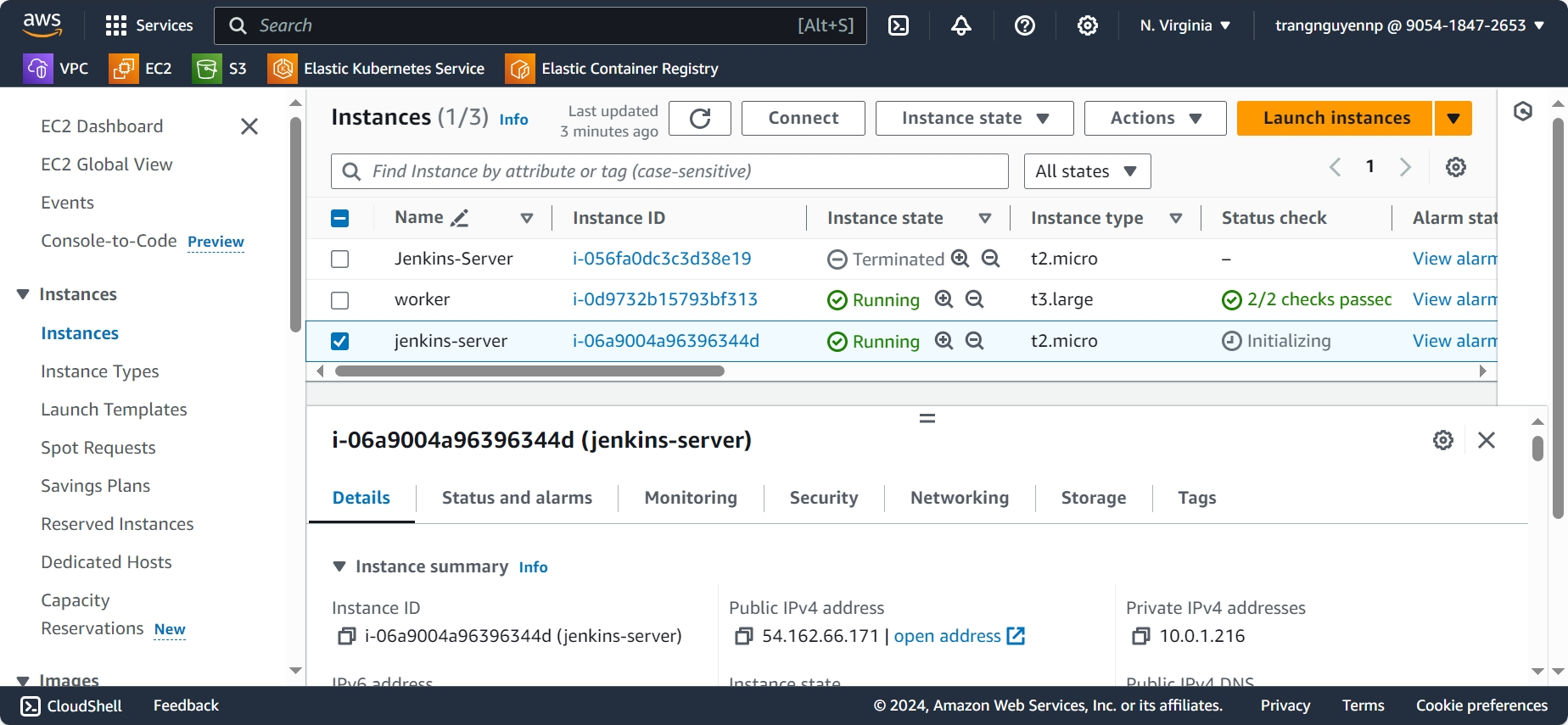
- Run terraform plan -var-file=”dev.tfvars” to apply the execution plan.

- Run terraform apply -var-file=”dev.tfvars” to create all the resource.

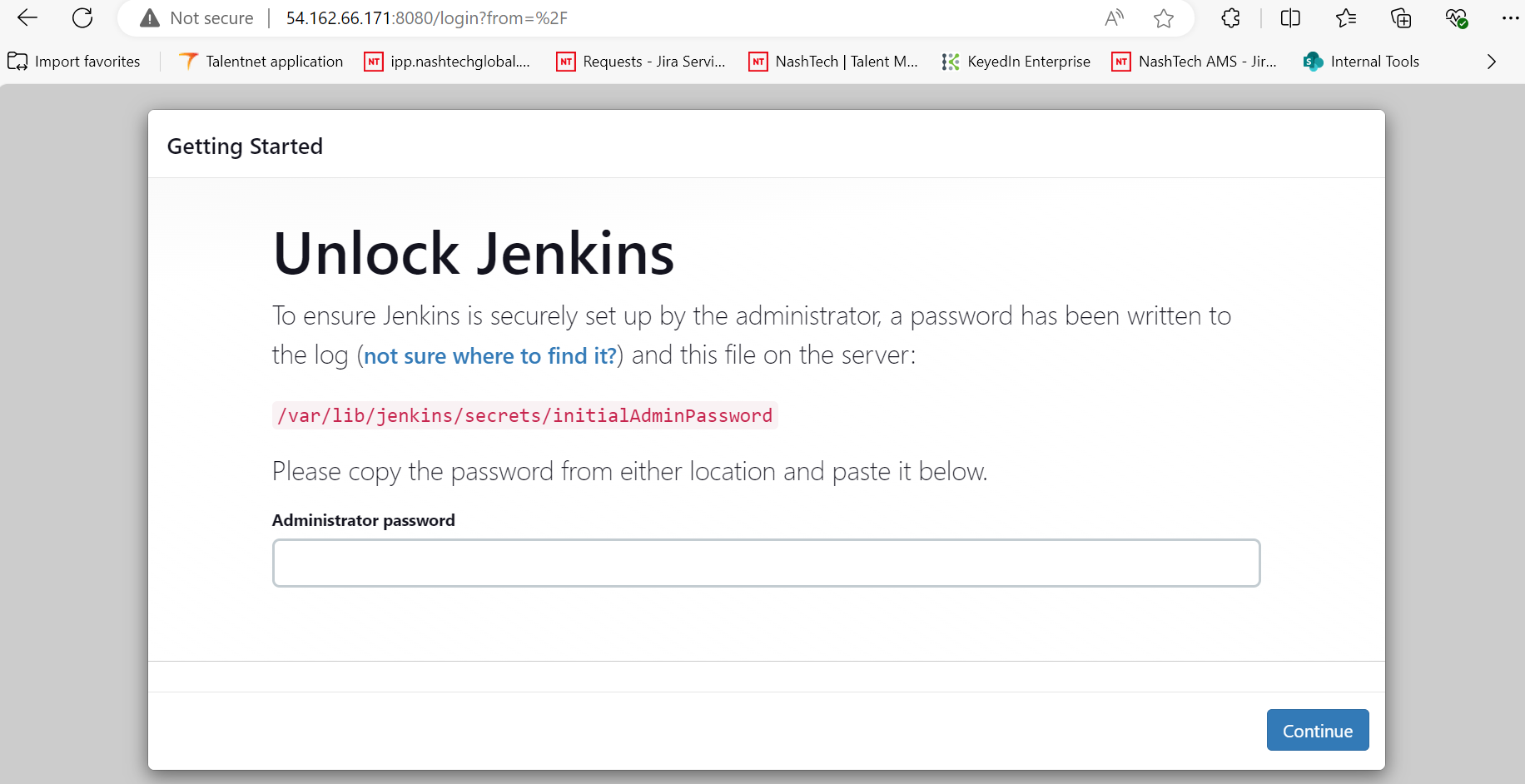


## **3.2. Verify the terraform plan**

- On EC2 page you will see the **jenkins-server** instance created.

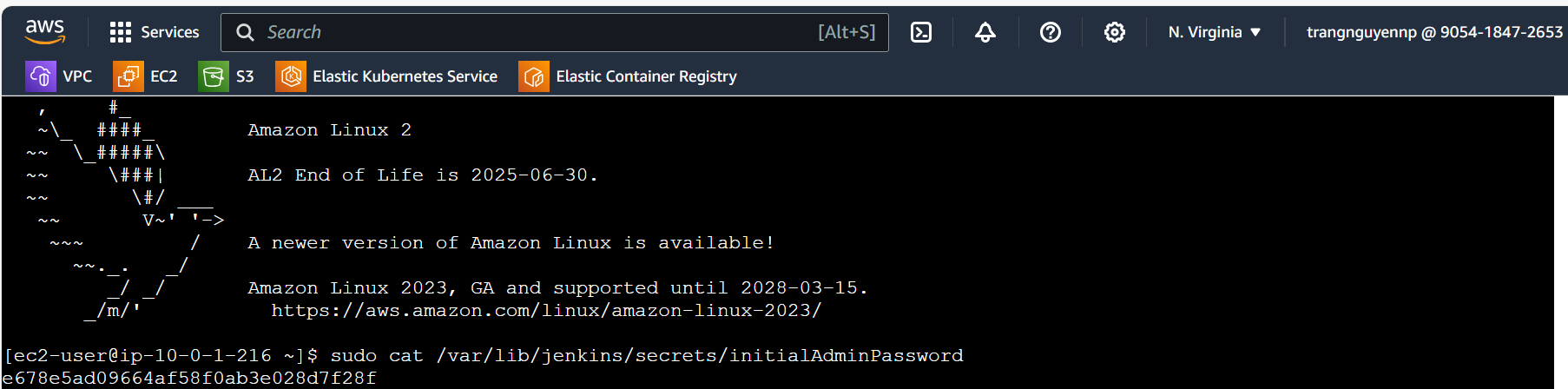


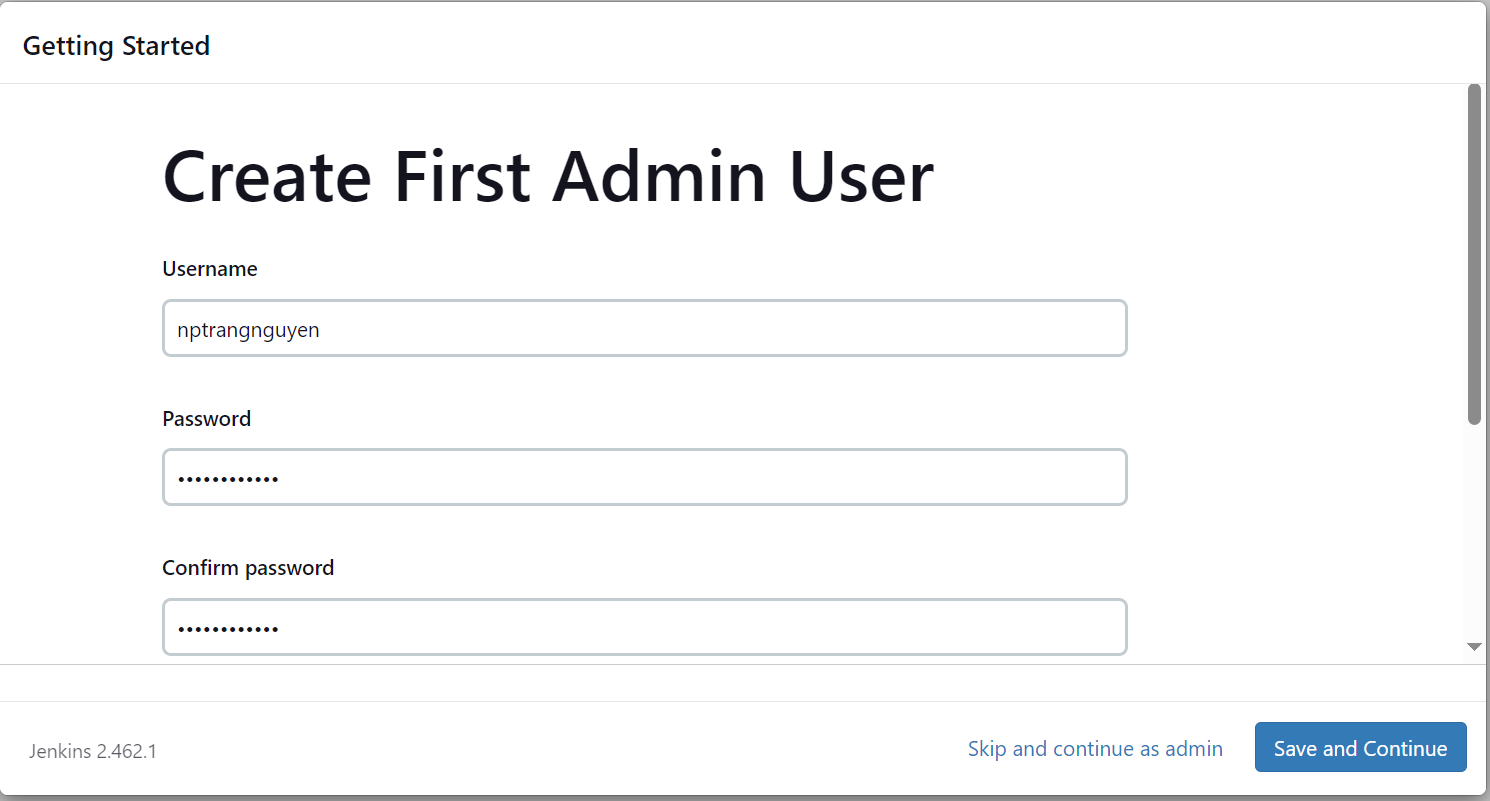
- Open address <http://ec2-public-ip:8080> to start the Jenkins.



## **3.3. Setup Jenkins account**

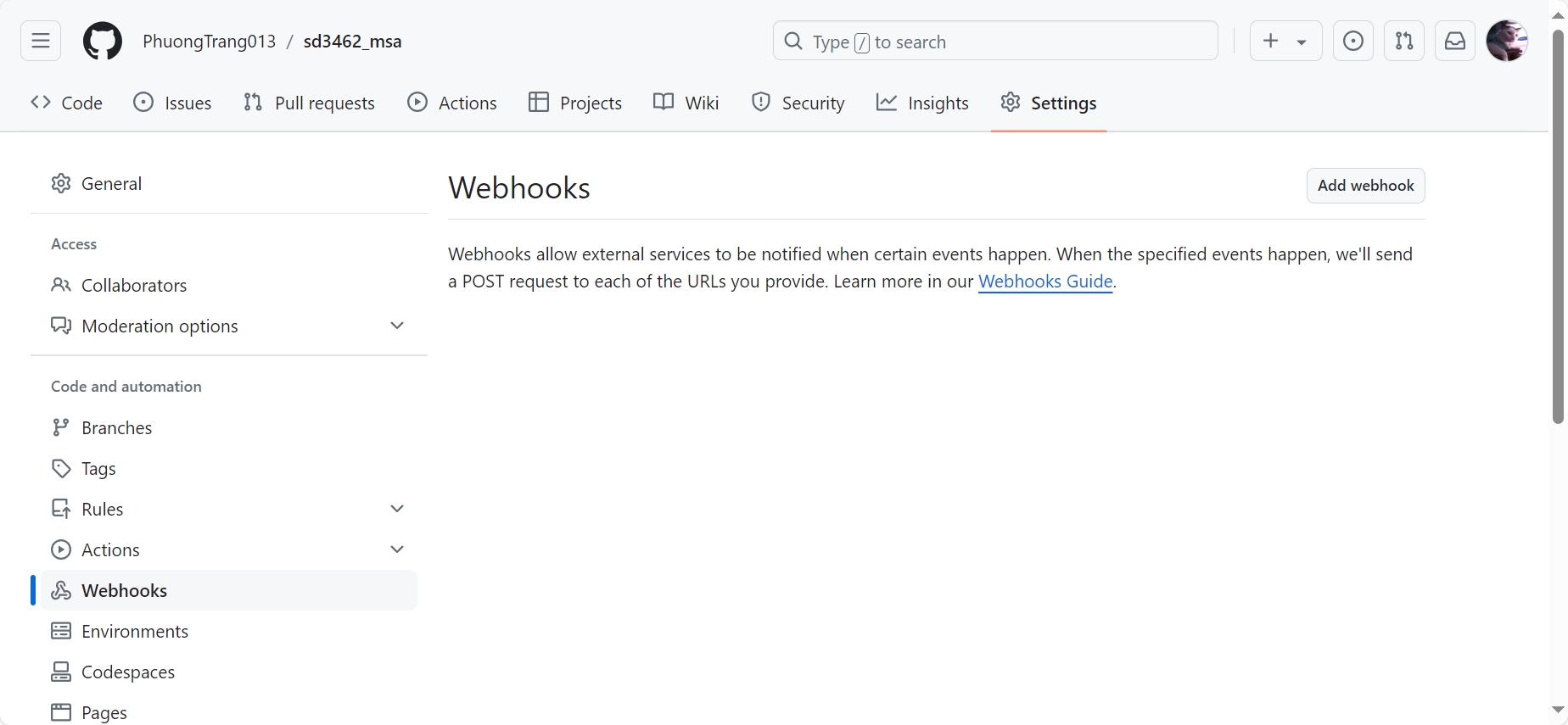
- Following the getting started page to get the administrator password then setup your own username and password.



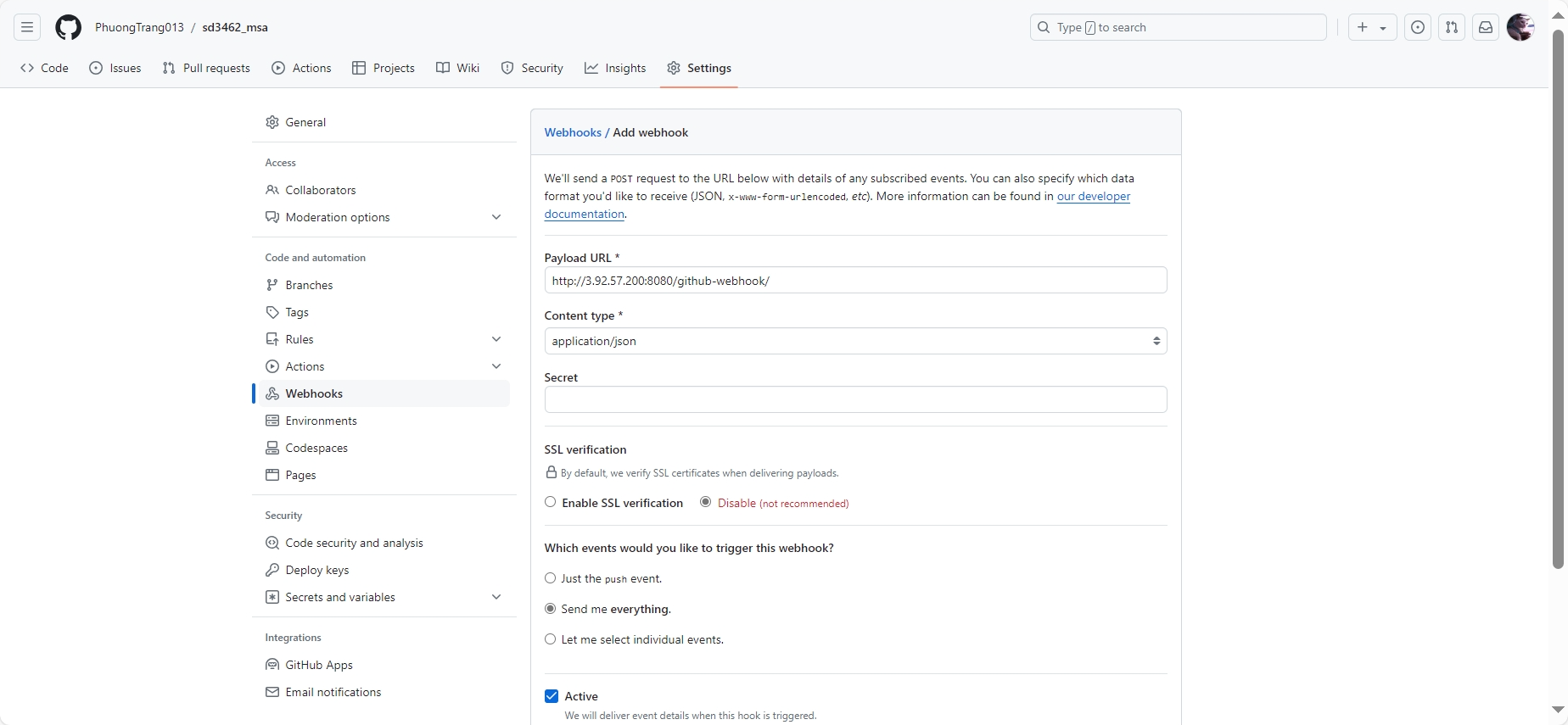


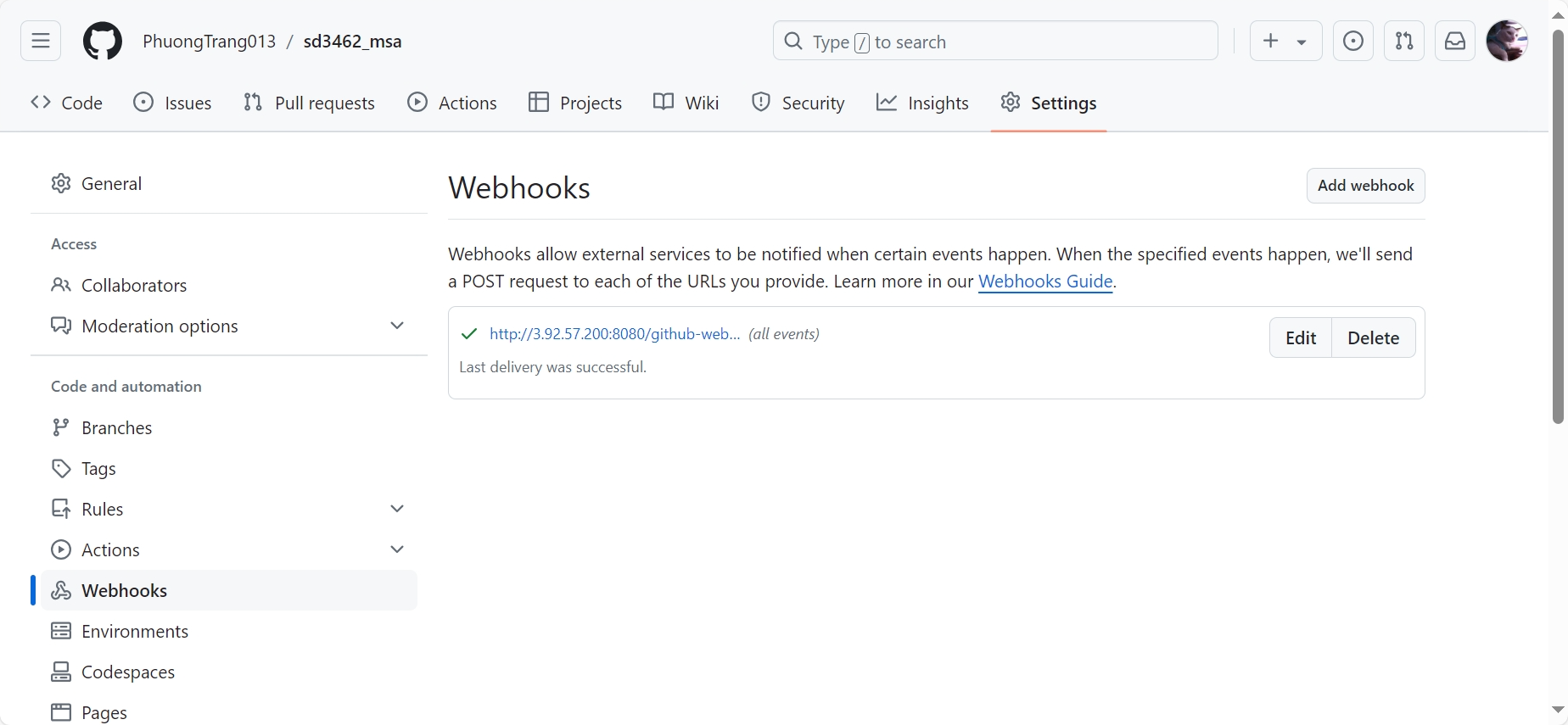
1. **Setup Jenkins pipeline (Push images to AWS ACR)**

* Add webhook for git repository. Click on **Repository settings** > **Webhook** and click **Add webhook.**

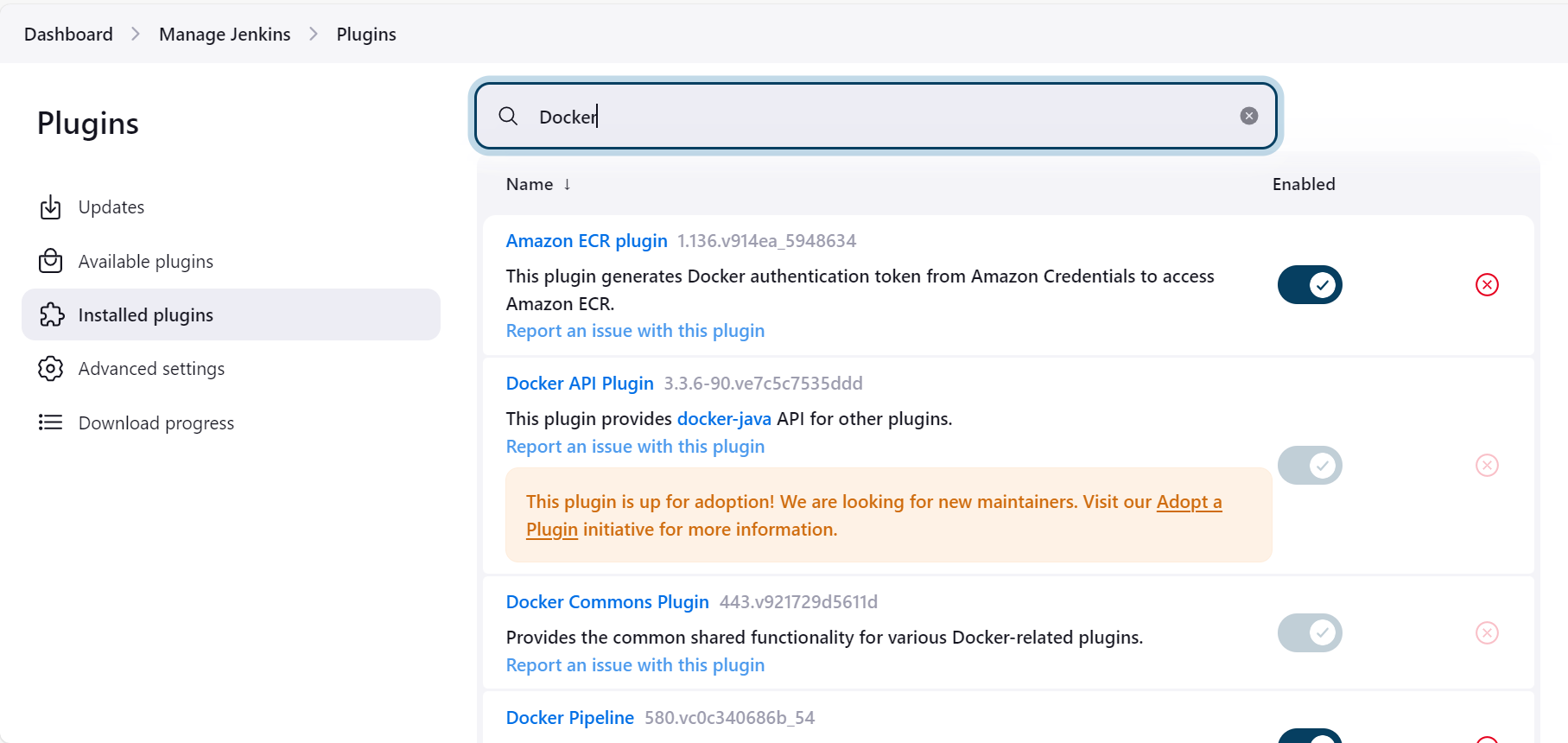


* Give a name to webhook > paste the url of the jenkins in the url section > click on generate secret > create webhook.

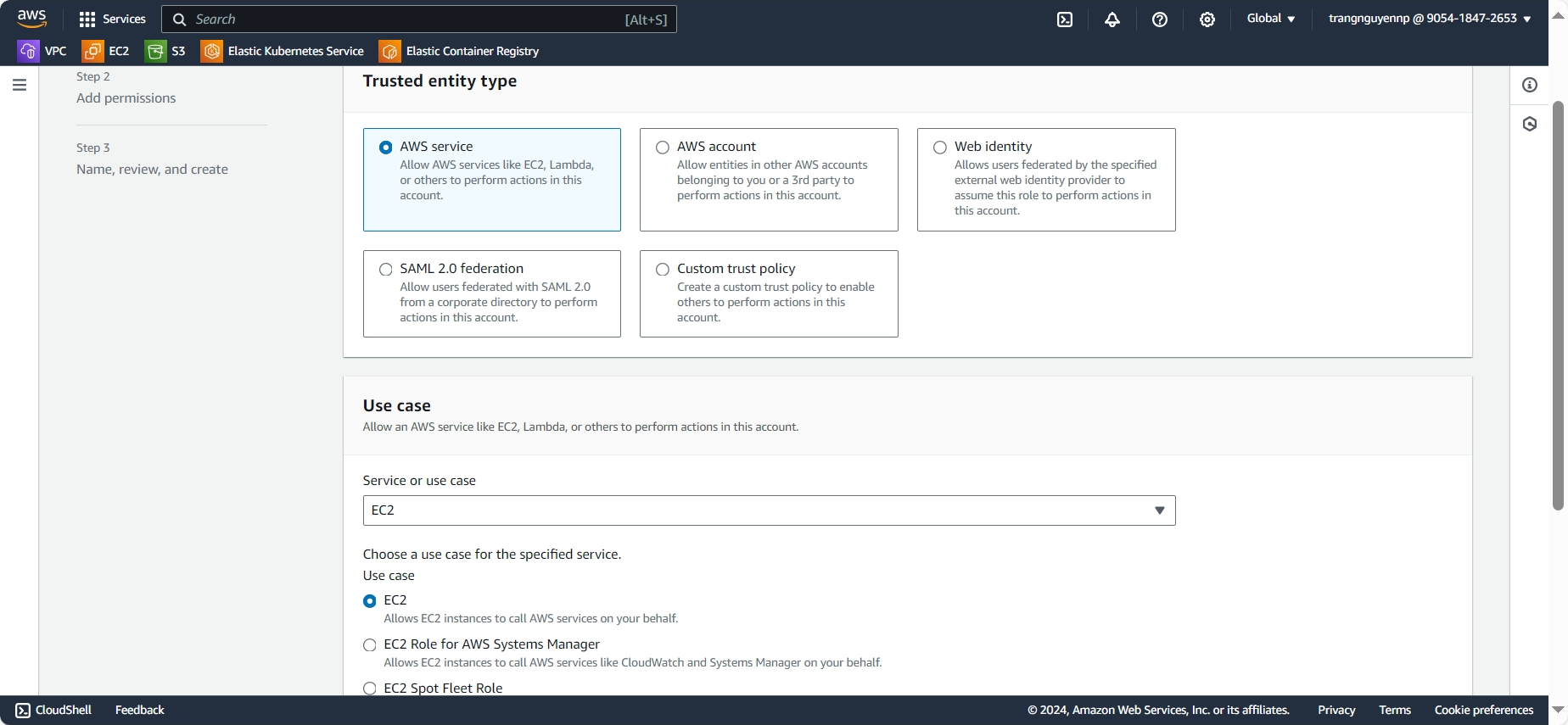




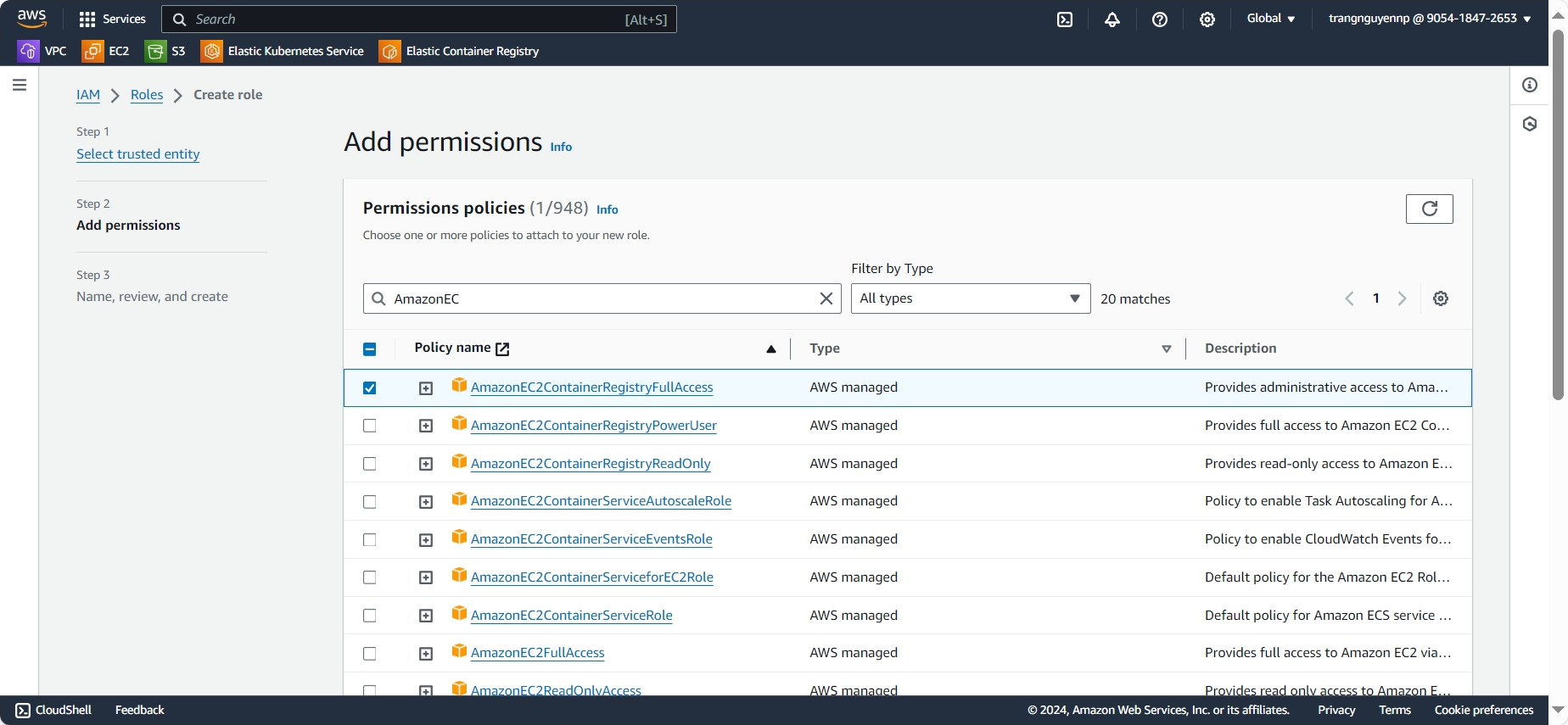
* On jenkins install necessary plugins to run docker and kubernetes commands.(docker, docker pipeline, amazon ecr )



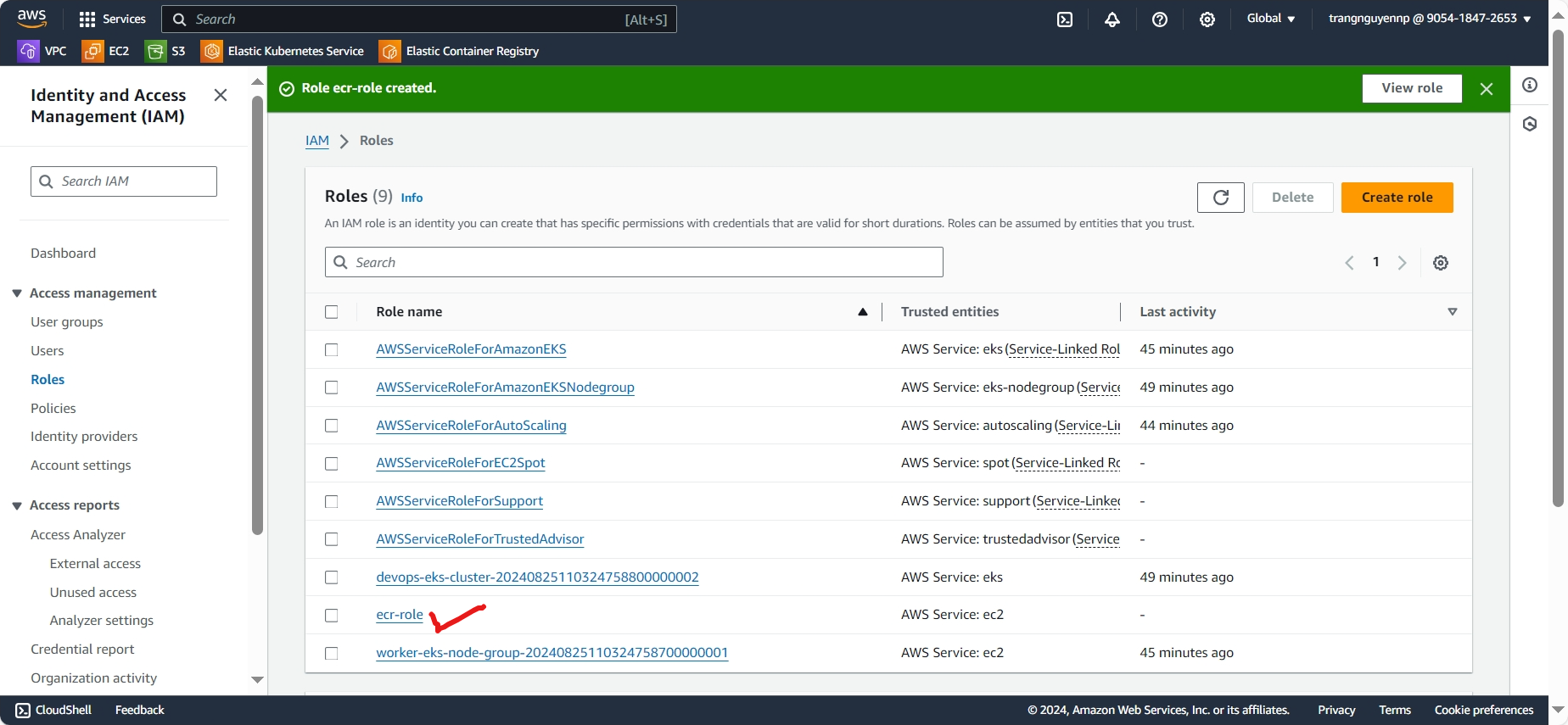
* Create IAM Role, go to **IAM** page and click **Create Role.** At select **trusted entity section**, choose **AWS service** for trusted type and **EC2** for the use case then click **Next**.



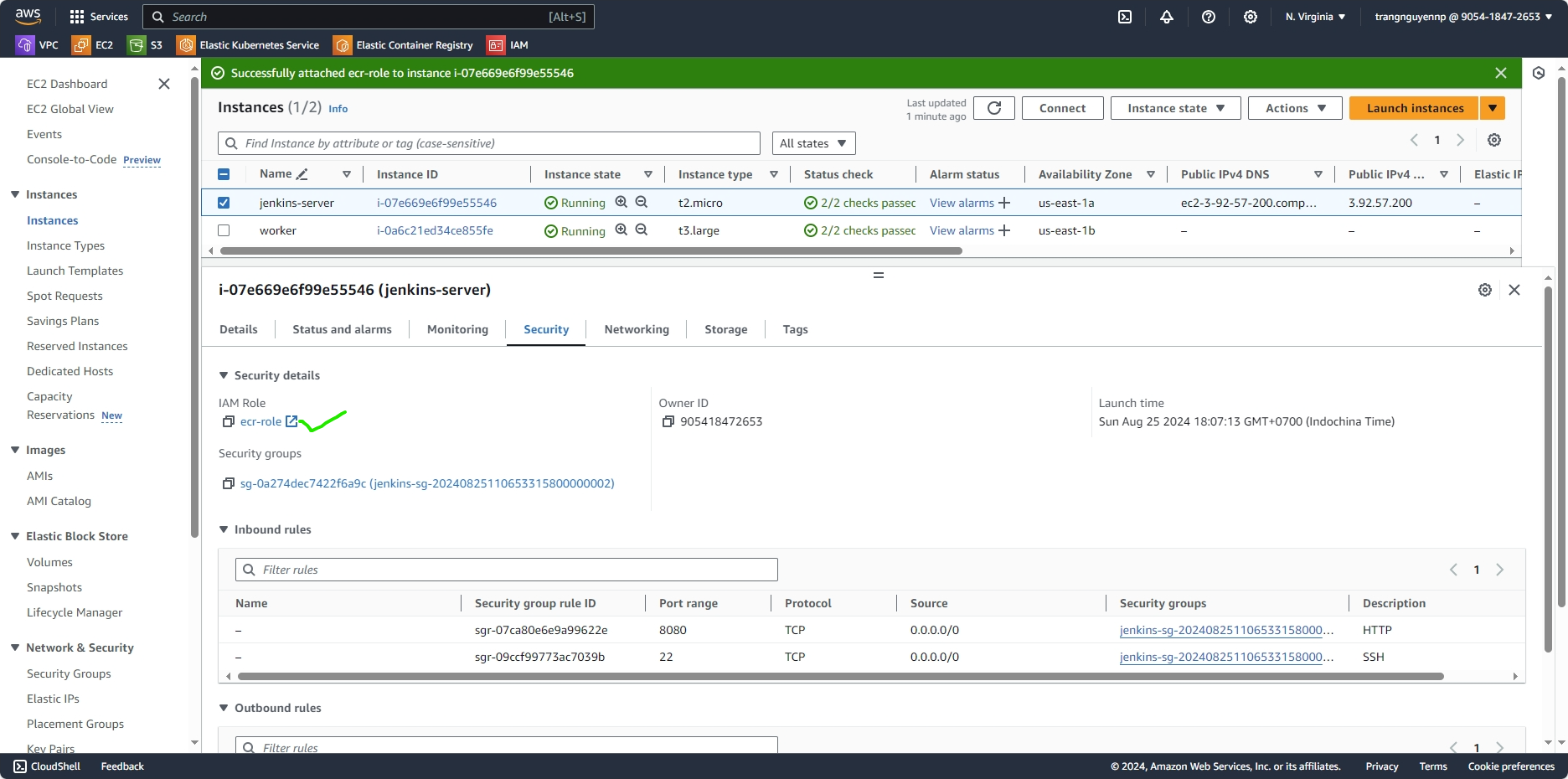
* On Add permissions section, add **AmazonEC2ContainerRegistryFullAccess** then click **Next**.



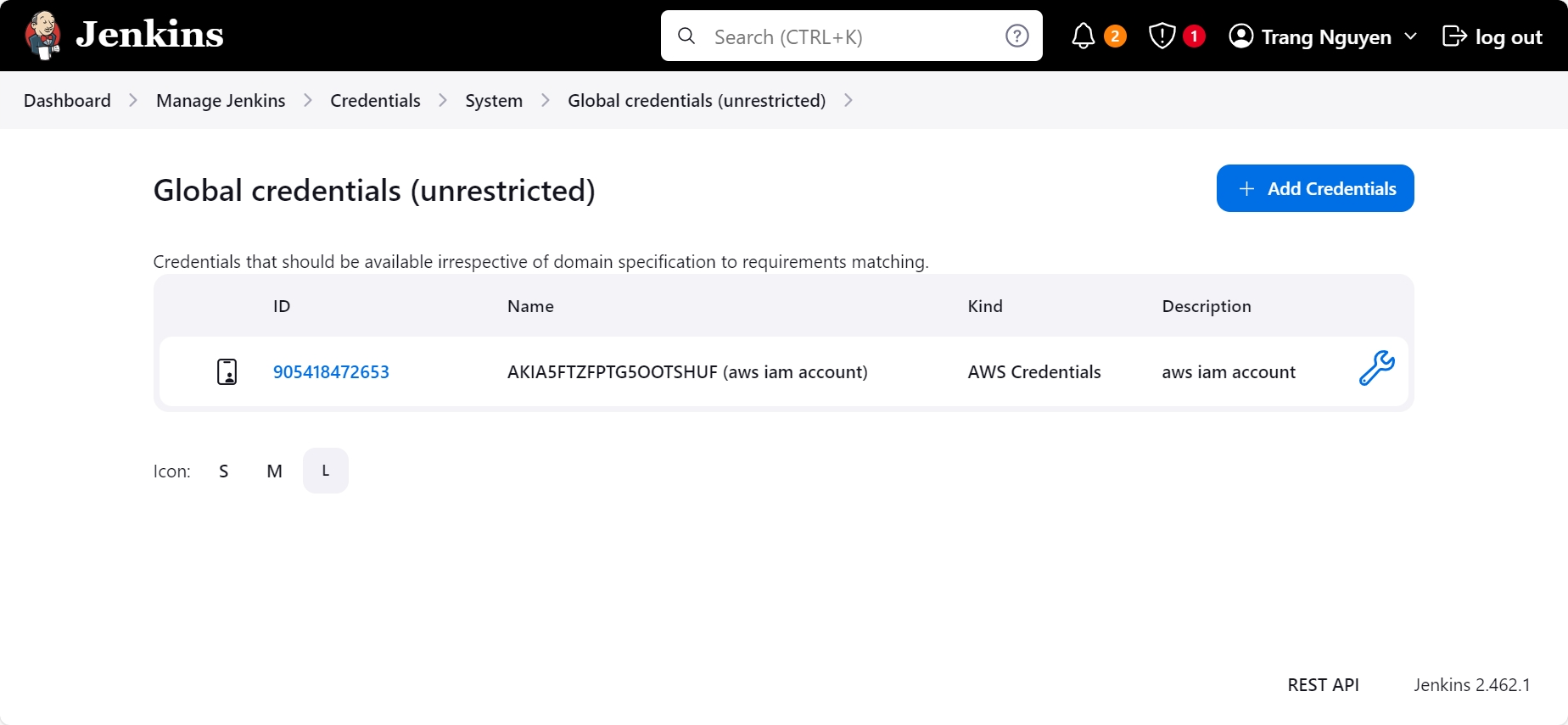
* Give a role name, review and click **Create role**.

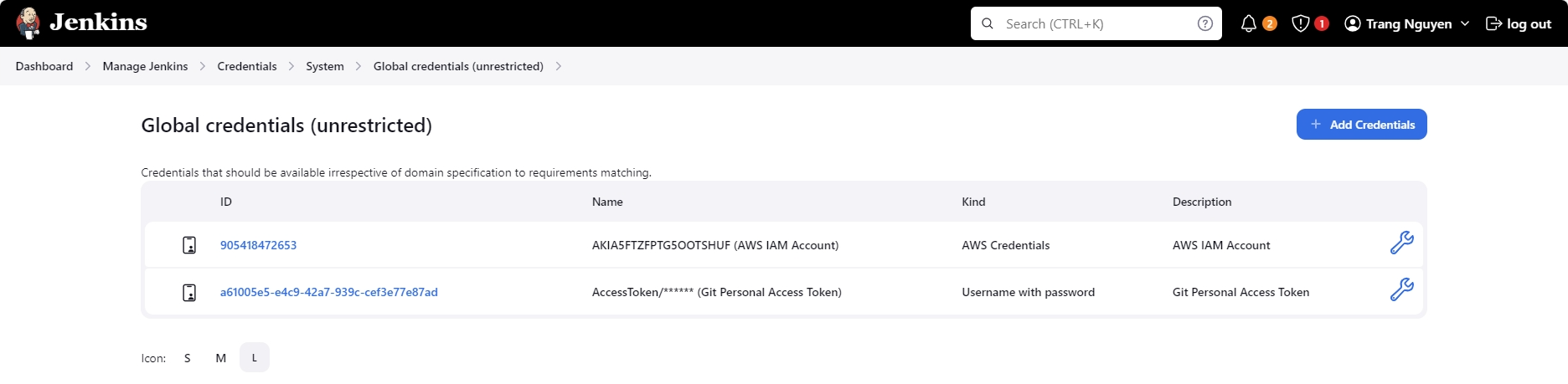


* Then add IAM role to “jenkins-server” on EC2 instances page.

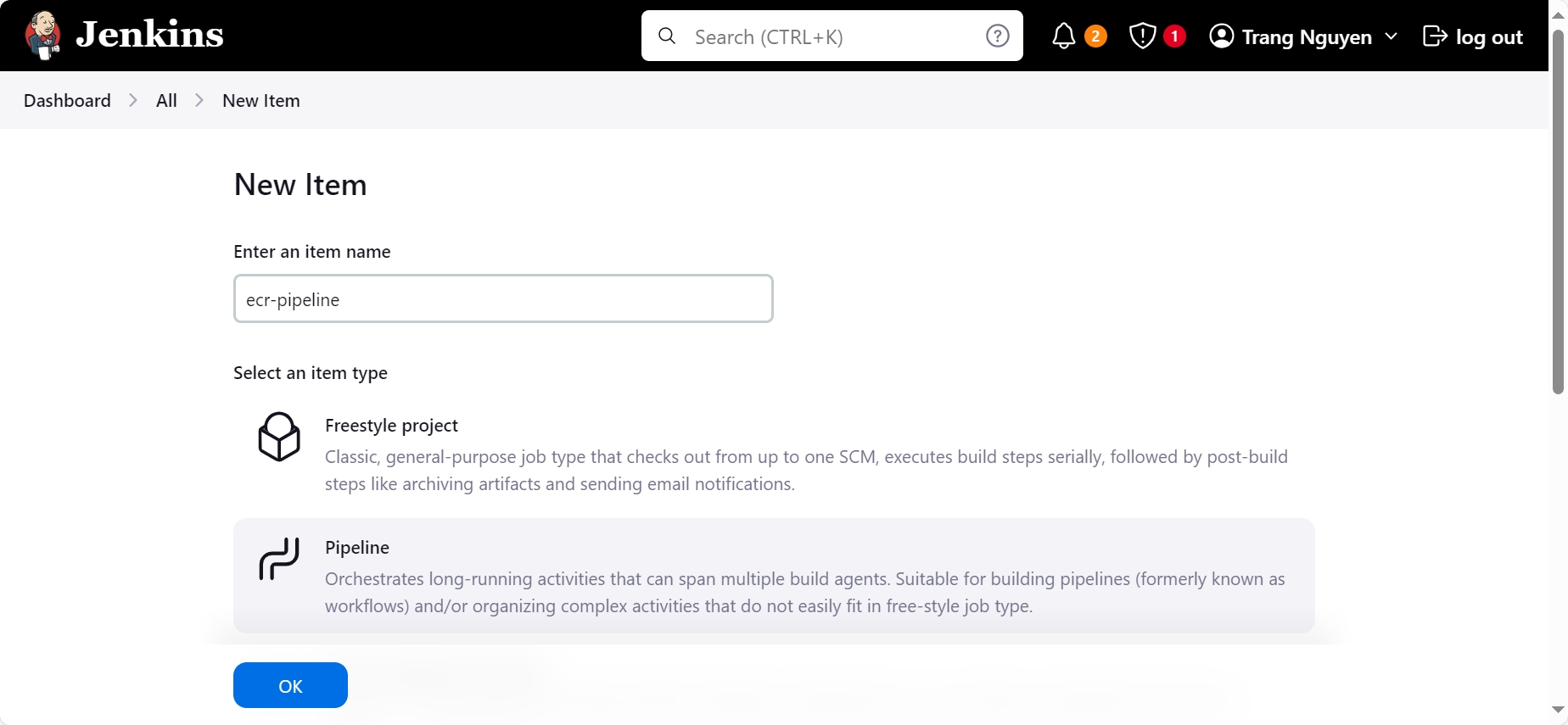


* Add **AWS** **credential** and **Github credential** (personal access token), go to the Manage Jenkins>>Credentials>>system>>Global credentials.

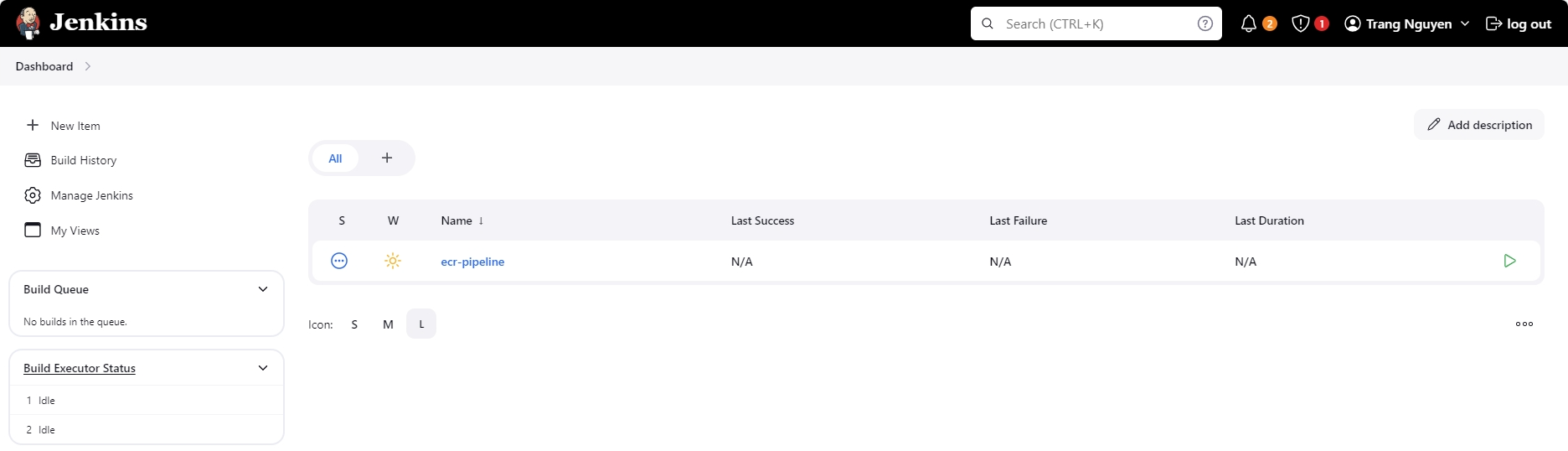




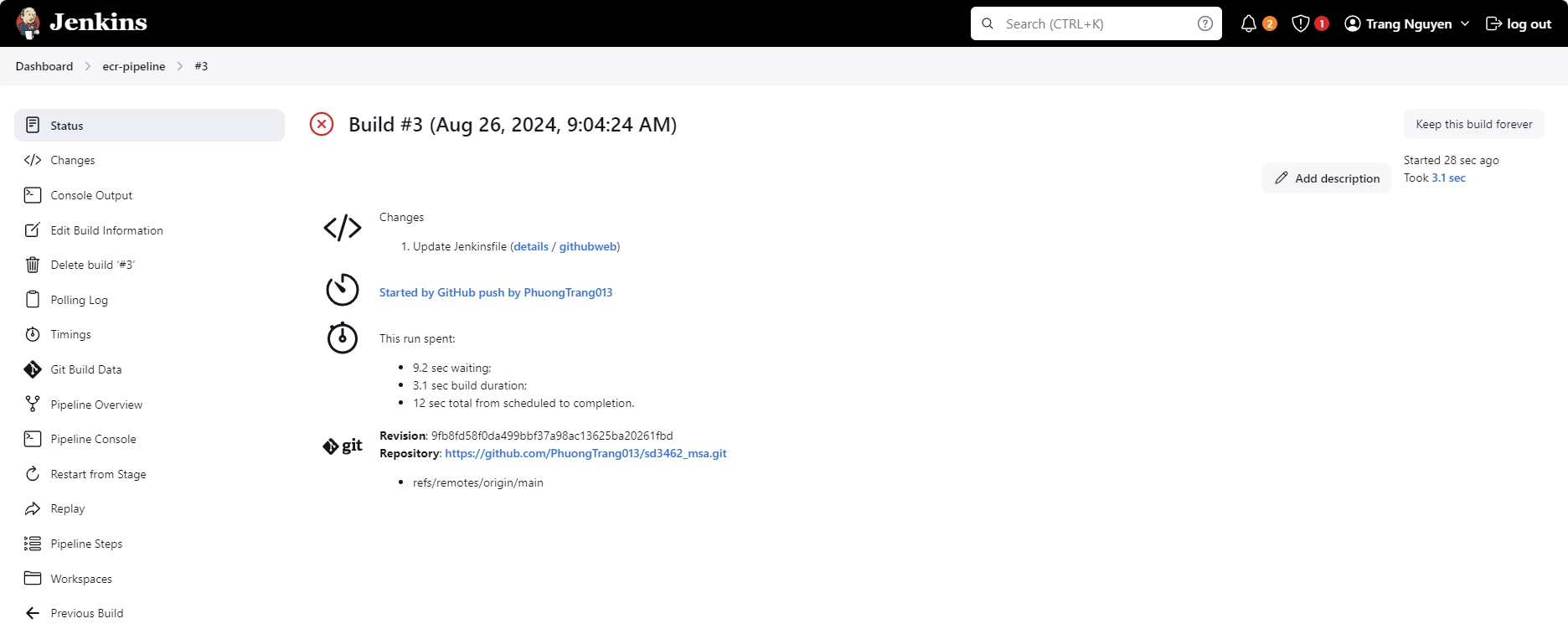
* Create Jenkins pipeline, go to the Jenkins dashboard then click on “**New Item**” and choose “**Pipeline**” from the options.



* Fill out the following fields for the pipeline, leaving everything else as default then Save.
* **GitHub hook trigger for GITScm polling** - check the box
* **Pipeline - Definition** - pipeline script from SCM
* **SCM** - Git
* **Repository URL** - the URL of your repo
* **Credentials** – github credential
* **Branch Specifier** - \*/main
* **Script path** - Jenkinsfile



* When we commit on repo this will trigger a build on Jenkins.



1. **EKS Setup**
2. **Monitoring with Prometheus and Grafana**