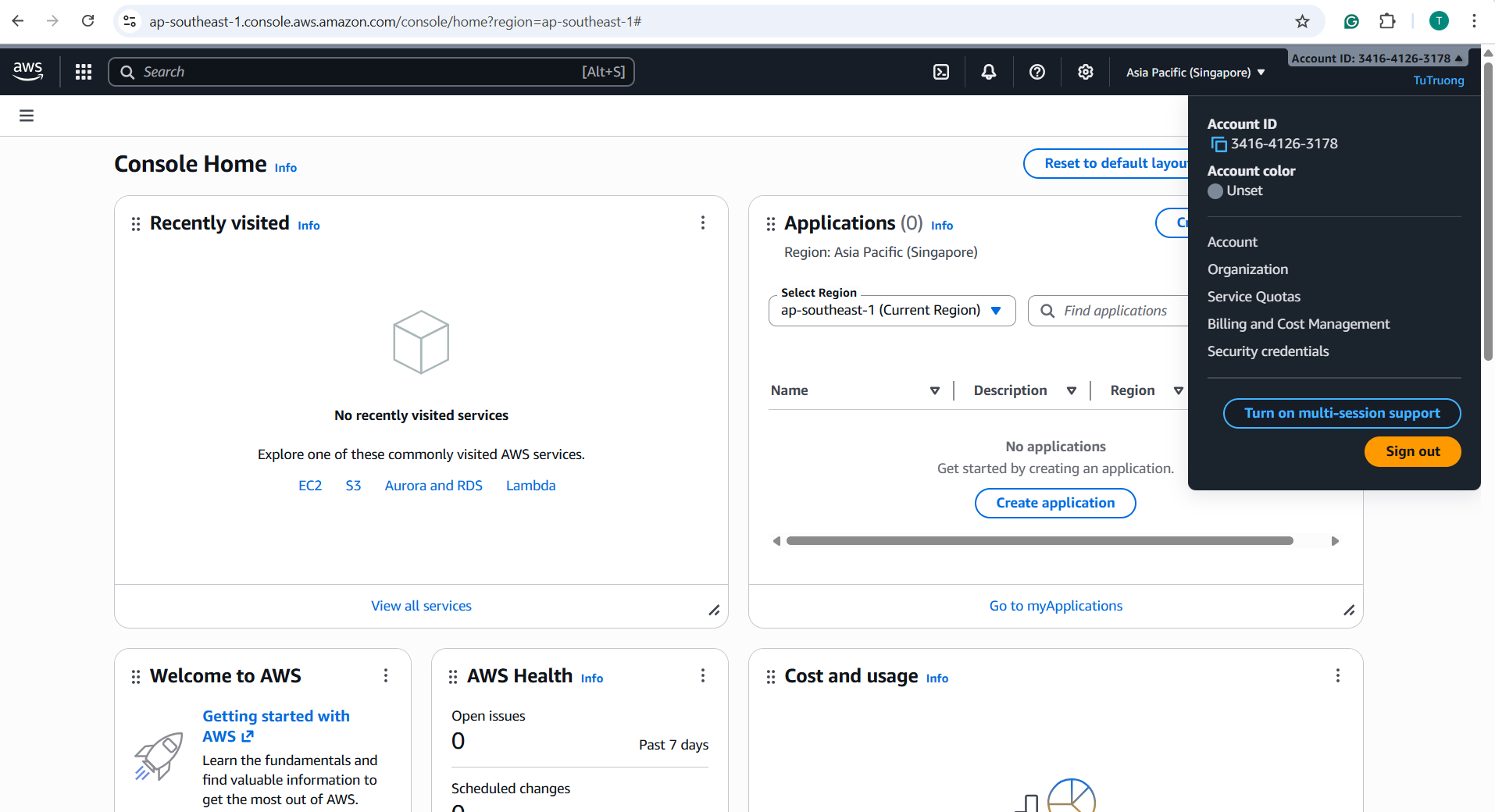
### ****APPLICATION DEPLOY TO EKS****

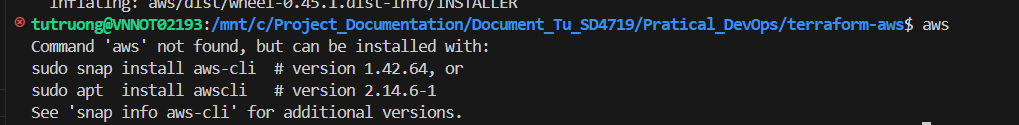
### ****- AWS Account Setup:****

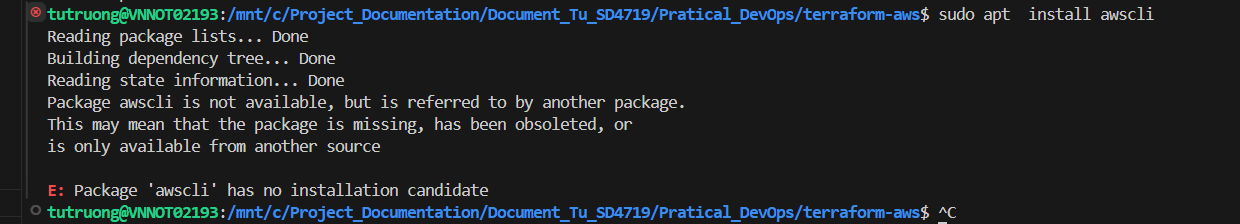
1. Go to [https://aws.amazon.com](https://aws.amazon.com/) => **Create a new account with permission to create infra (VPC / EC2 / ECR / EKS / IAM)**.



1. Verify for installation of Docker, Jenkins, AWS CLI, kubectl and helm.

Currently, on the WSL Terminal I have checked that only the **awscli** is not installed

 And the issue is when installing the **awscli** package because Ubuntu package list is outdated or it is no longer in the default repositories



So, we can install the AWS CLI v2 manually:

- **Remove any old versions:**

sudo apt remove awscli -y

- **Download the AWS CLI v2 package:**

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

- **Unzip:**

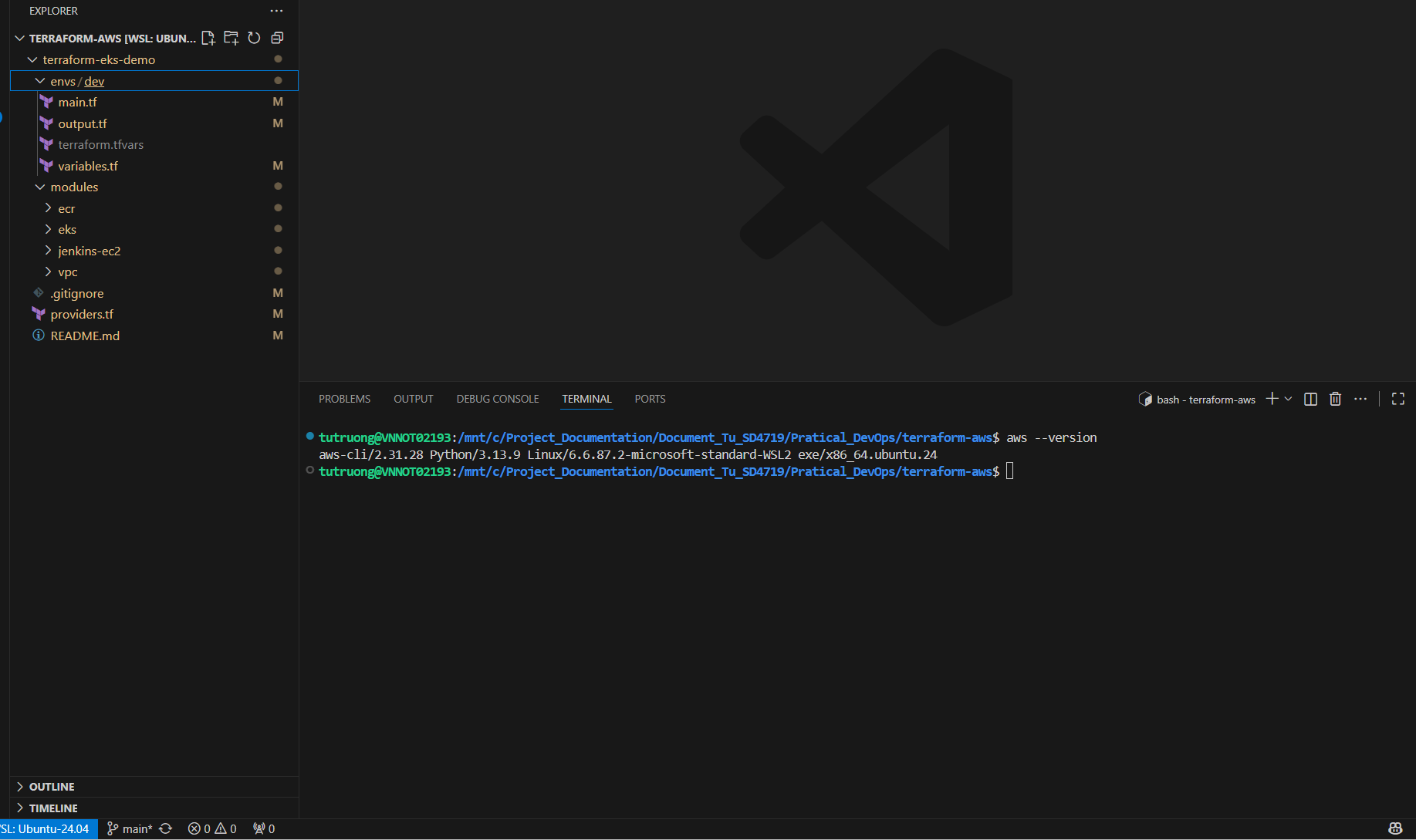
unzip awscliv2.zip

- **Run the installer:**

sudo ./aws/install

- V**erify installation:**

aws –version

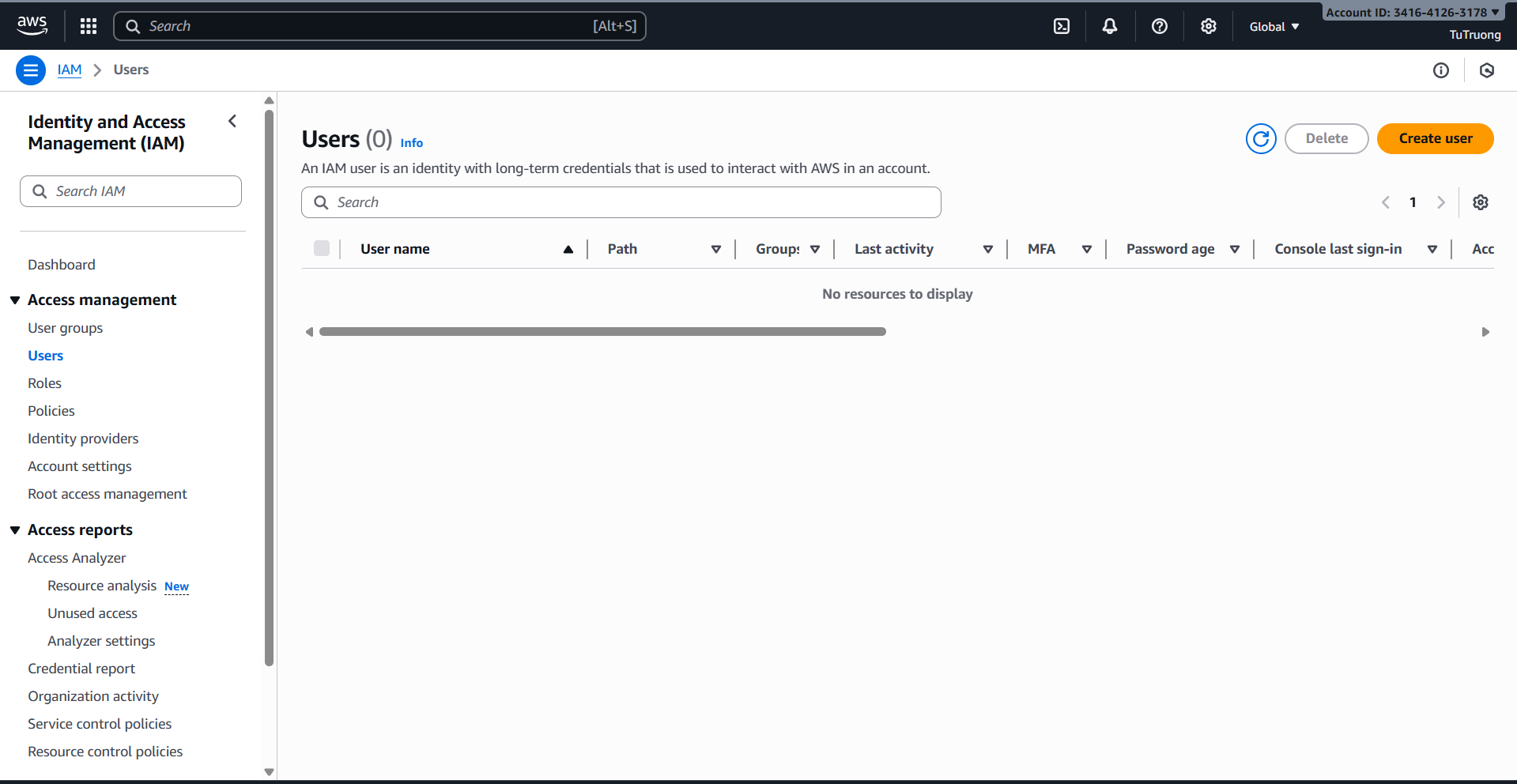


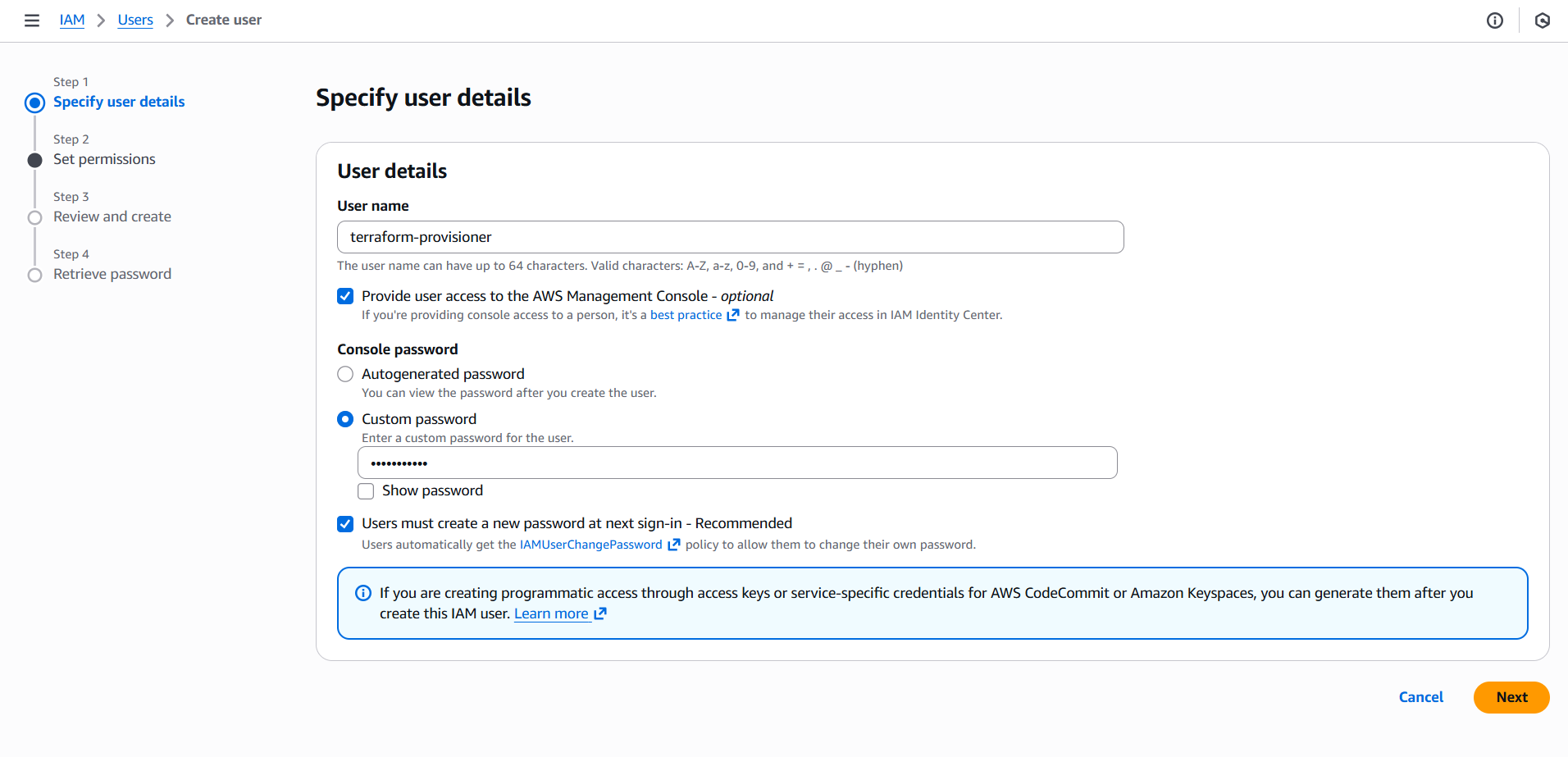
1. Check AWS CLI authentication:

- Create an **IAM** **(Identity and Access Management)** user for Terraform:

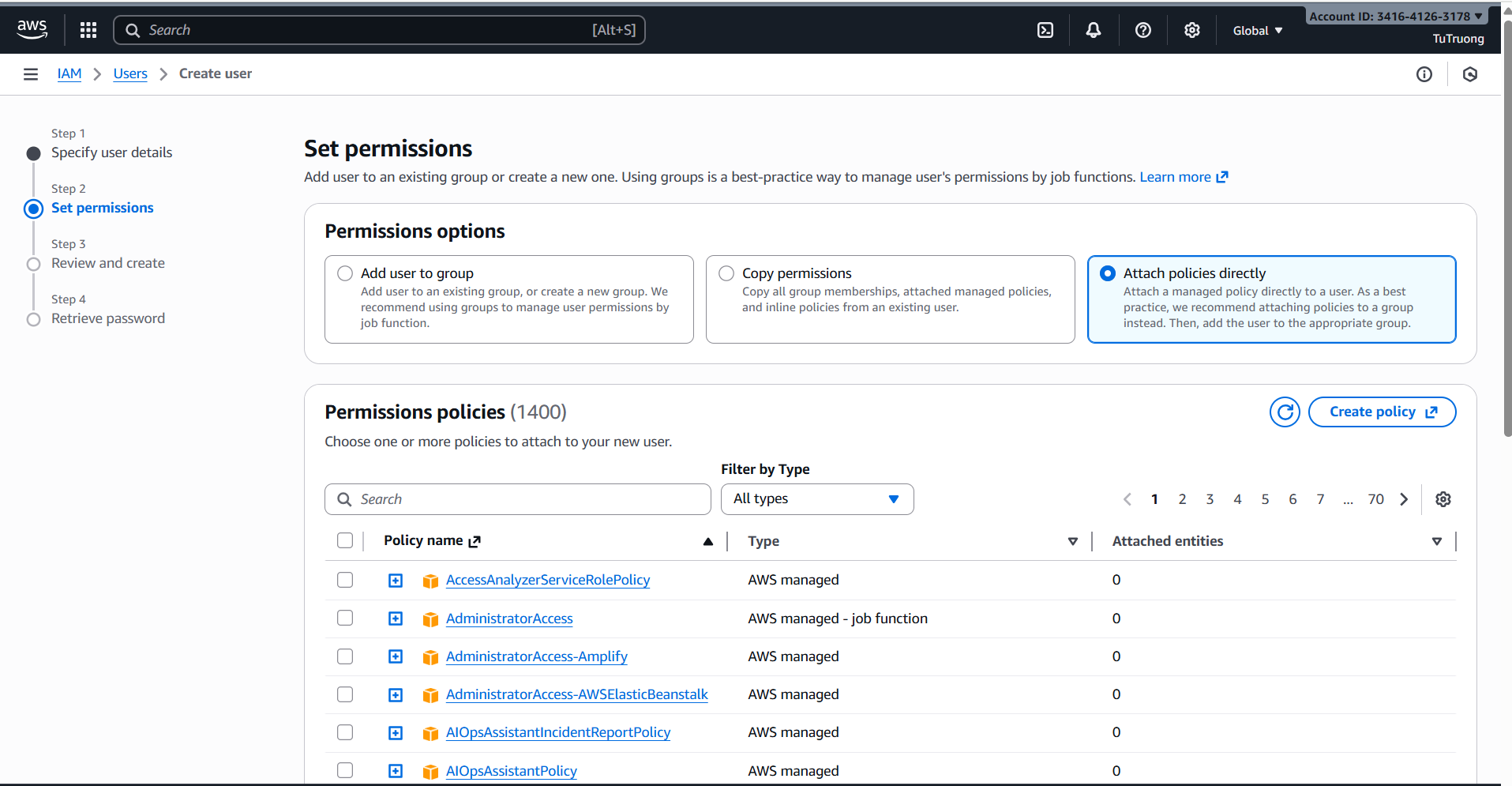
+ Access in to the AWS Management Console

+ Open **IAM** => **Users** => **Add users**.





+ Create a Least-Privilege Terraform Policy => For the task **Create user**, in the step 2 (Set permisstions) then click **Create policy** button => navigate to **Create polic**y page



+ For the task **Create policy**, in the step 1 (**Specify permissions**) then choose **JSON** tab and paste this policy JSON to keep everything secure, cost-controlled and AWS evaluates these policies when the IAM principal makes a request. Permissions in the policies determine whether the request is allowed or denied:

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "EKSManagement",

"Effect": "Allow",

"Action": [

"eks:\*",

"kms:\*"

],

"Resource": "\*"

},

{

"Sid": "AWSSystemsManager",

"Effect": "Allow",

"Action": [

"ssm:GetParameter",

"ssm:GetParameters",

"ssm:GetParametersByPath"

],

"Resource": [

"arn:aws:ssm:\*:\*:parameter/aws/service/eks/\*"

]

},

{

"Sid": "EC2ForEKSNodes",

"Effect": "Allow",

"Action": [

"ec2:\*"

],

"Resource": "\*"

},

{

"Sid": "ECRAccess",

"Effect": "Allow",

"Action": [

"ecr:GetAuthorizationToken",

"ecr:CreateRepository",

"ecr:DeleteRepository",

"ecr:DescribeRepositories",

"ecr:ListImages",

"ecr:PutImage",

"ecr:BatchDeleteImage",

"ecr:TagResource",

"ecr:ListTagsForResource",

"ecr:PutLifecyclePolicy",

"ecr:GetLifecyclePolicy"

],

"Resource": "\*"

},

{

"Sid": "IAMRoleForEKS",

"Effect": "Allow",

"Action": [

"iam:\*"

],

"Resource": "\*"

},

{

"Sid": "PrometheusMonitoring",

"Effect": "Allow",

"Action": [

"aps:CreateWorkspace",

"aps:DeleteWorkspace",

"aps:DescribeWorkspace",

"aps:ListWorkspaces",

"aps:UpdateWorkspaceAlias",

"aps:PutAlertManagerDefinition",

"aps:PutRuleGroupsNamespace",

"aps:ListRuleGroupsNamespaces",

"aps:DescribeRuleGroupsNamespace",

"aps:DeleteRuleGroupsNamespace"

],

"Resource": "\*"

},

{

"Sid": "CloudWatchLog",

"Effect": "Allow",

"Action": [

"logs:CreateLogGroup",

"logs:DescribeLogGroups",

"logs:PutRetentionPolicy",

"logs:TagResource"

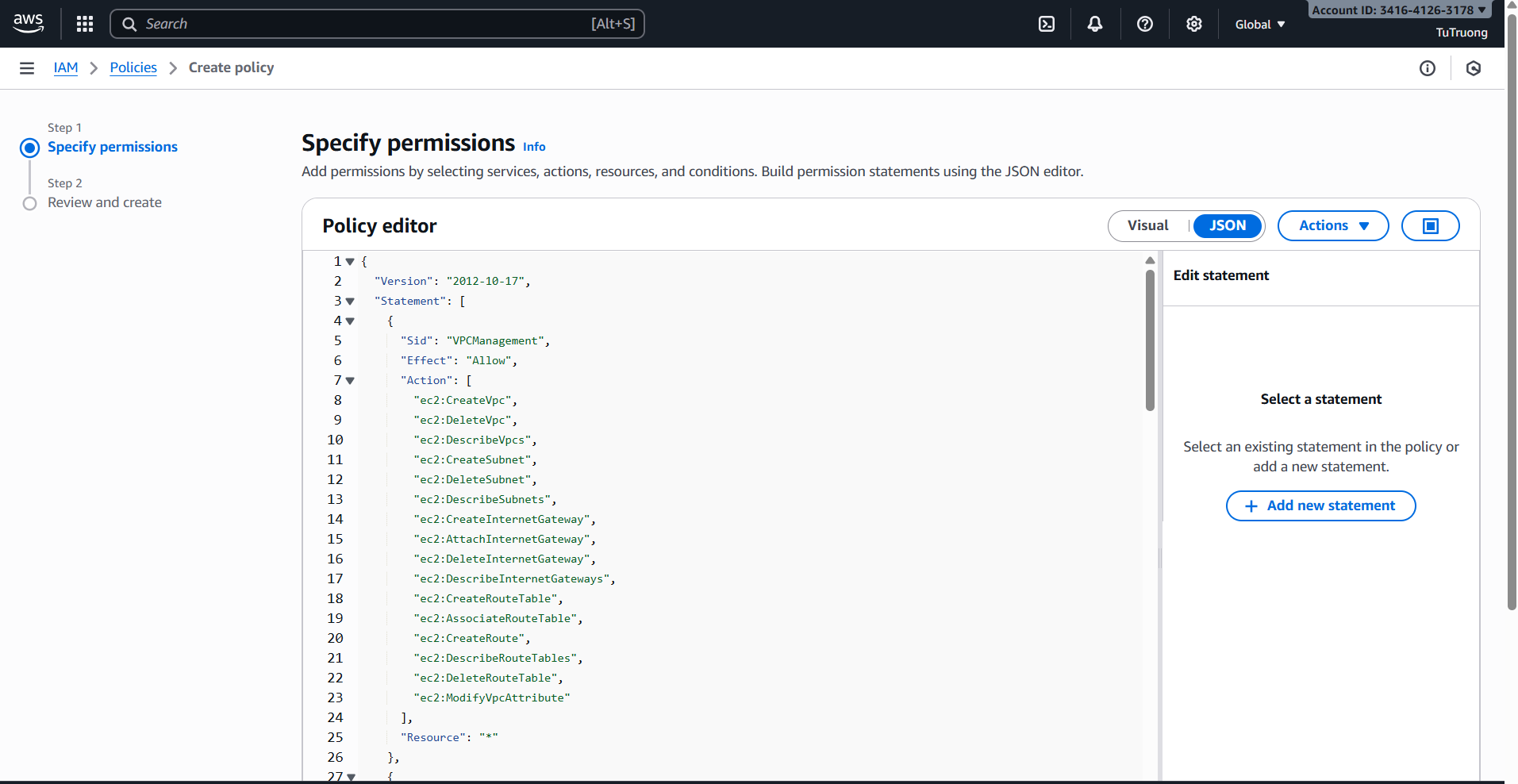
],

"Resource": "\*"

}

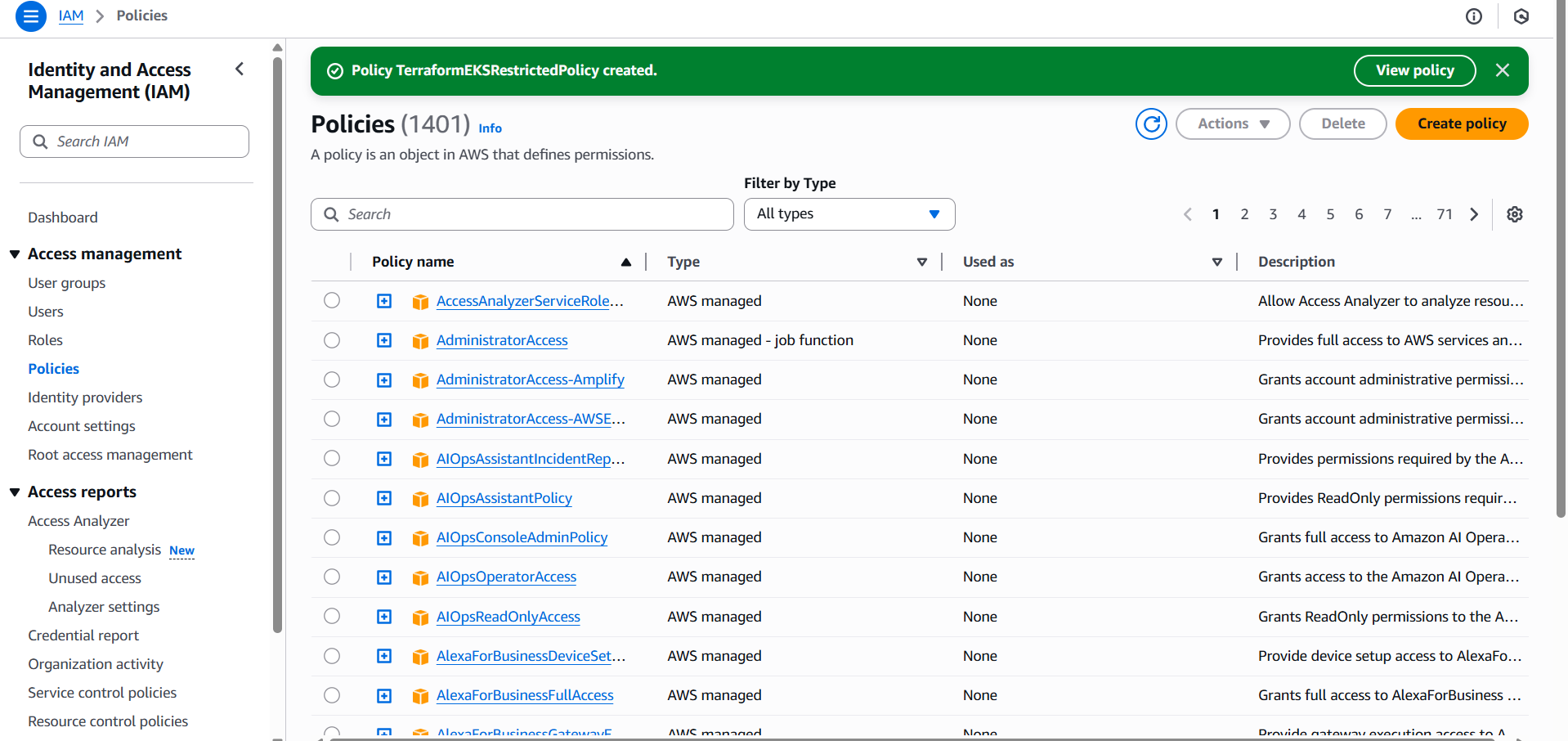
]

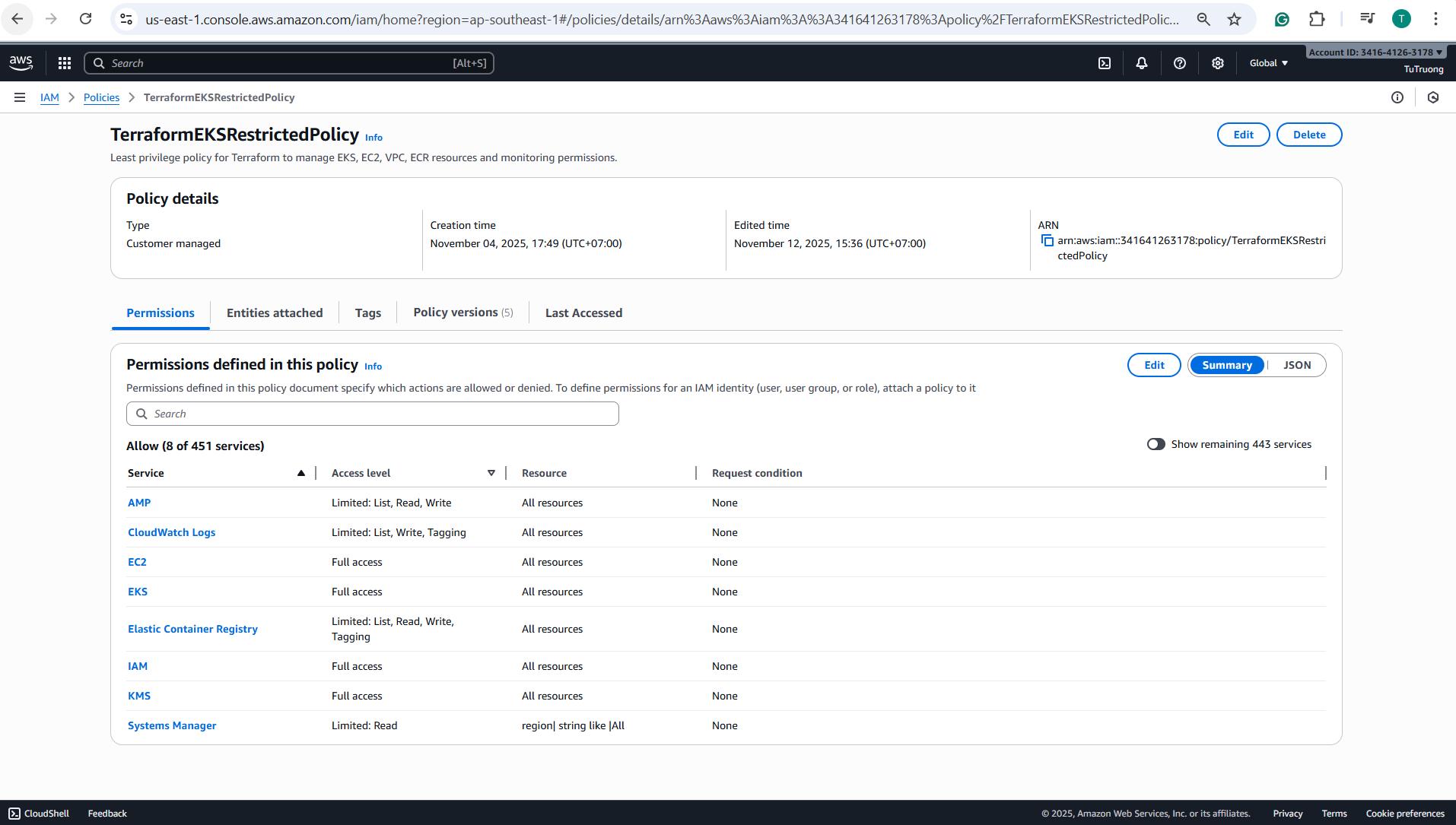
}



+ In the step 2 **(Review and Create)** in task **Create policy:**

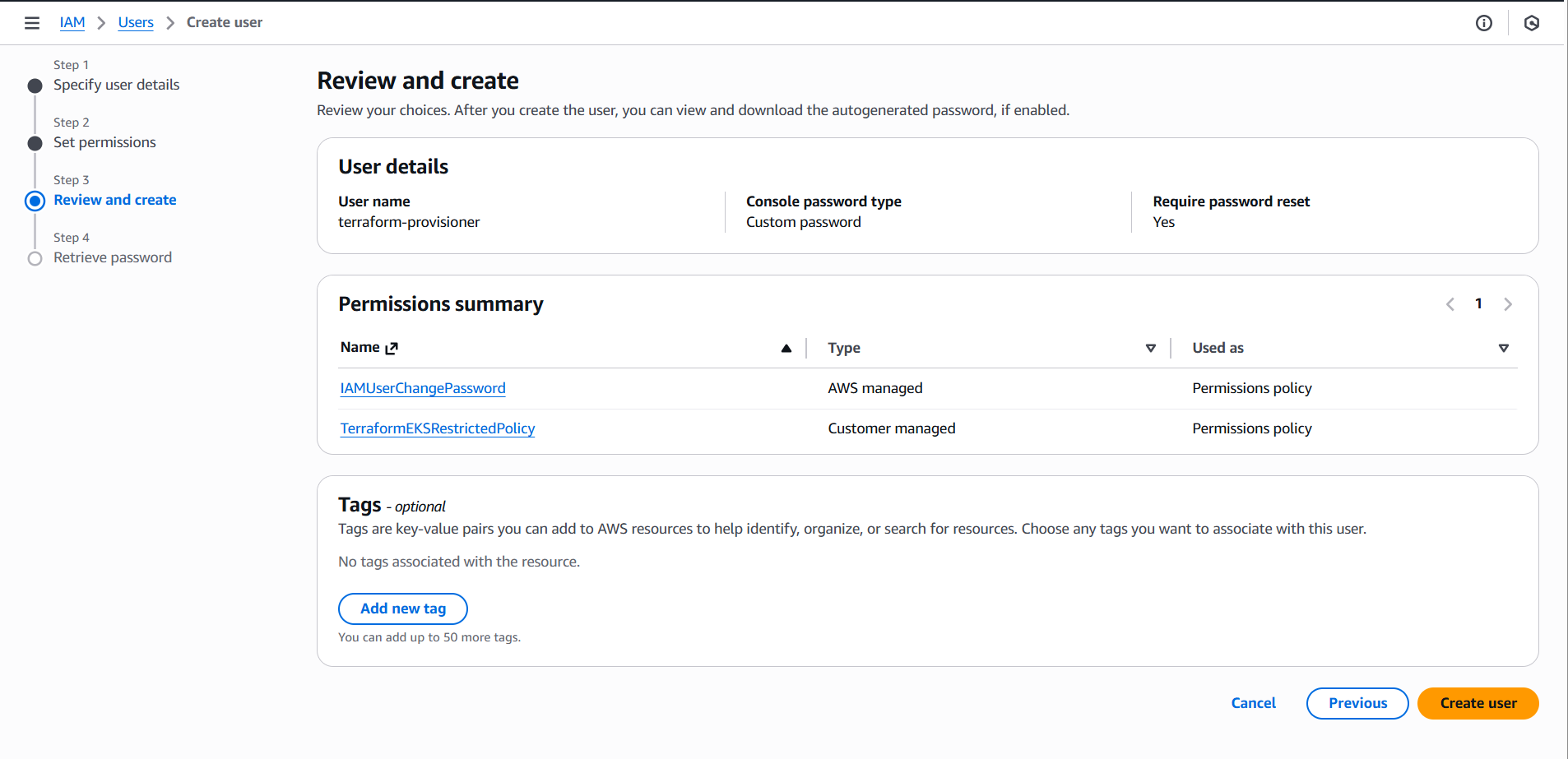
Name and describe for the policty and click **Create policy**

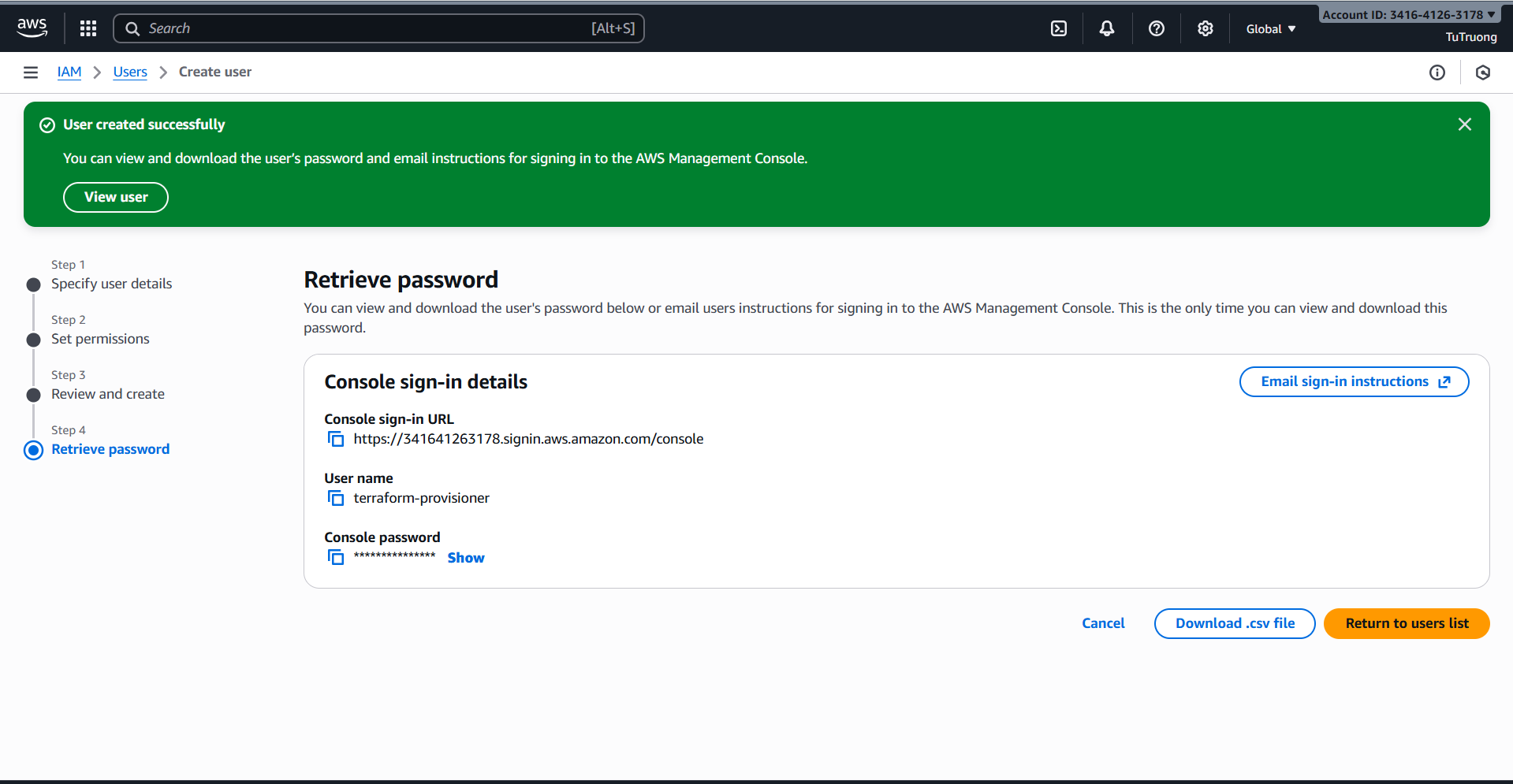




### + Attach this new policy to IAM User:

1. Go back to user creation tab.
2. Click **Refresh** (top right of the policy list).
3. Search for **TerraformEKSRestrictedPolicy**.
4. Check the box to attach it.
5. Continue to the step 3 (**Review and create) => click Create user**.

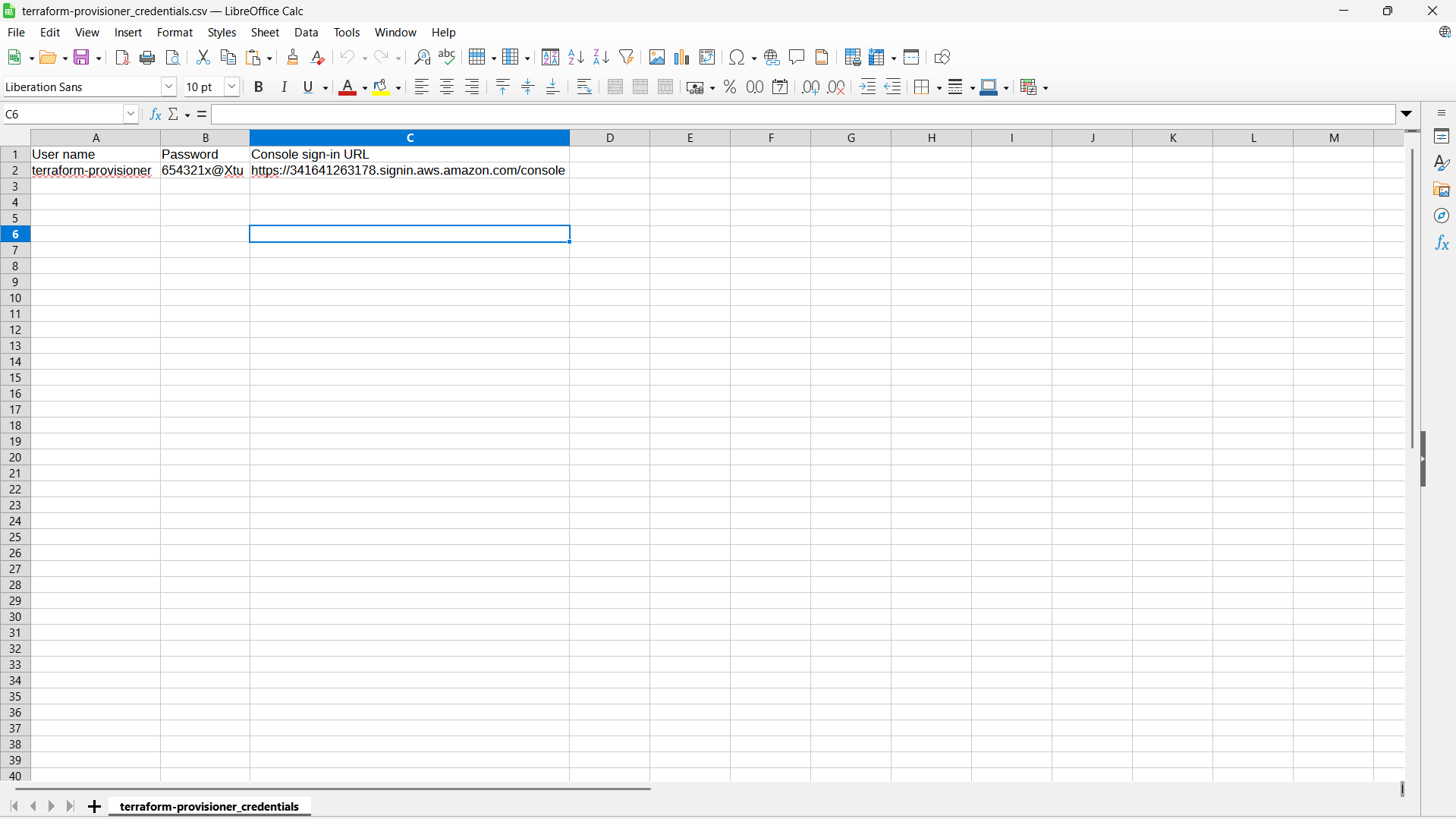




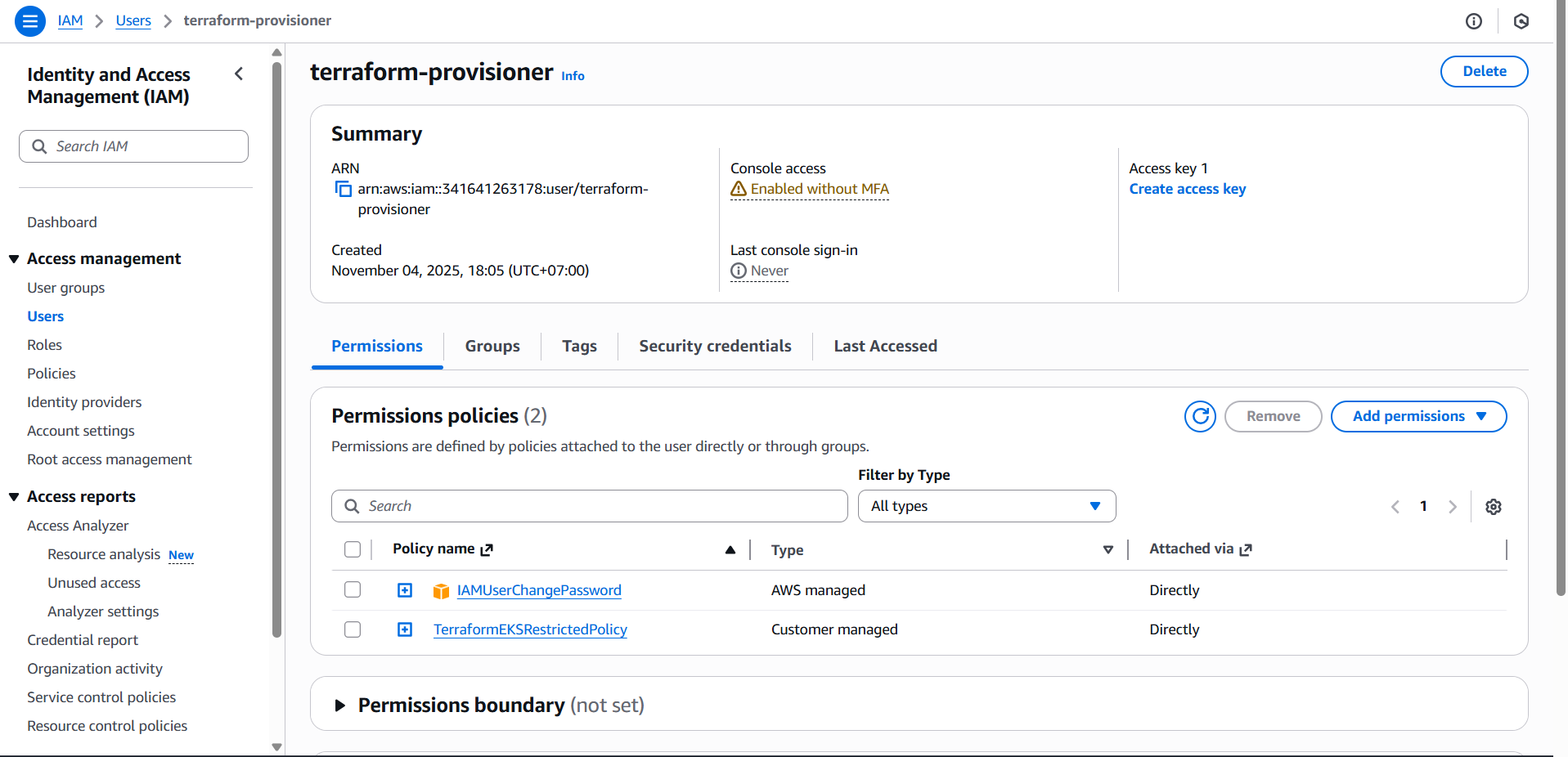
- Download Access Keys:

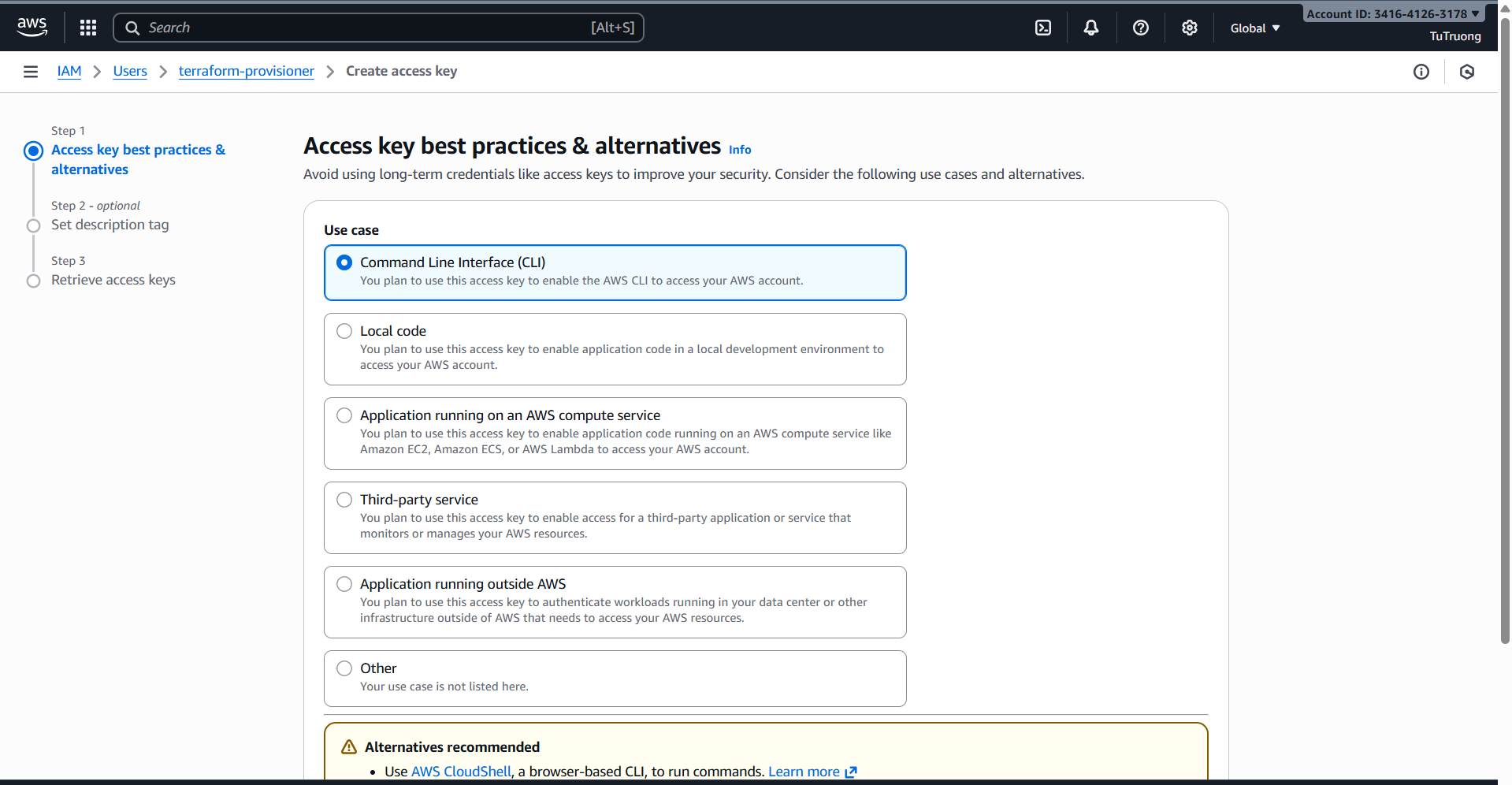
* Download the csv file containing ****Access Key ID**** and ****Secret Access Key**.**
* **If we created the IAM user, we selected “Password – AWS Management Console access” instead of “Access key – Programmatic access.”**

**=> In the CSV File doesn’t contain access keys and it only contains User Name,** **Password and** C**onsole sign-in URL**

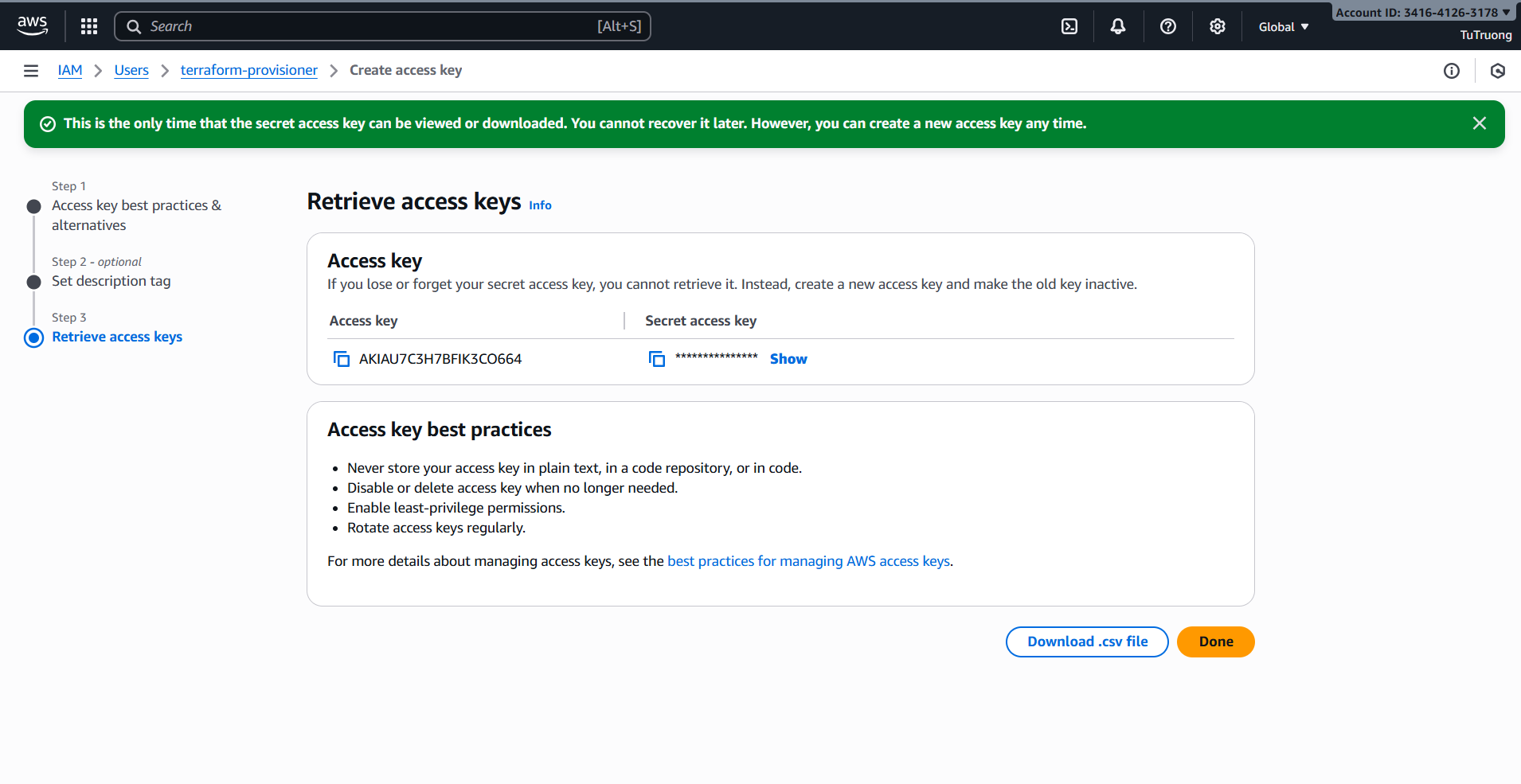


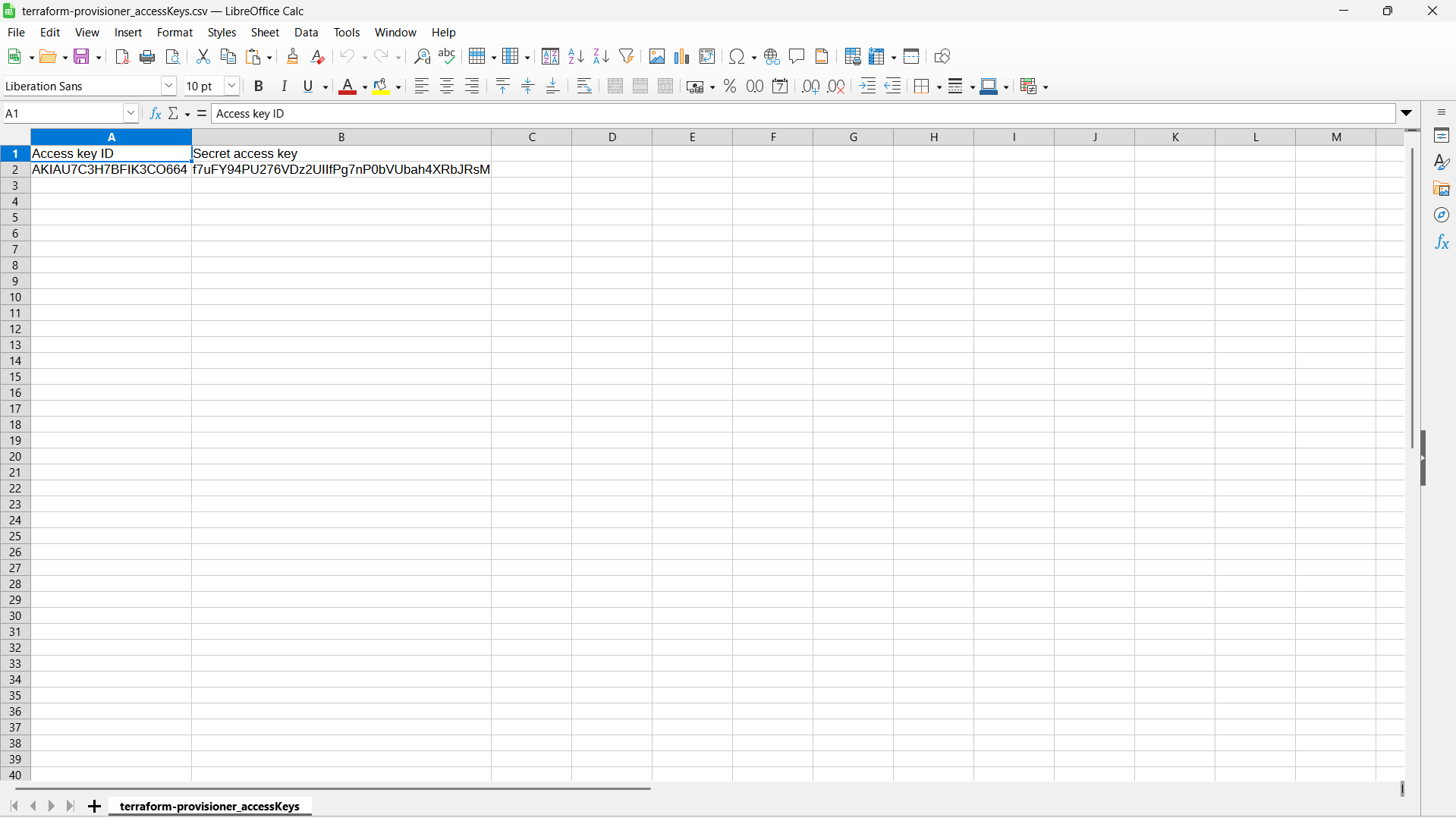
* **So let generate Access Keys in the IAM user (**terraform-provisioner**) just created => click **Create access key** => Choose **“Command Line Interface (CLI)”****





**And then download the CSV file again to get **Access Key ID** and **Secret Access Key****



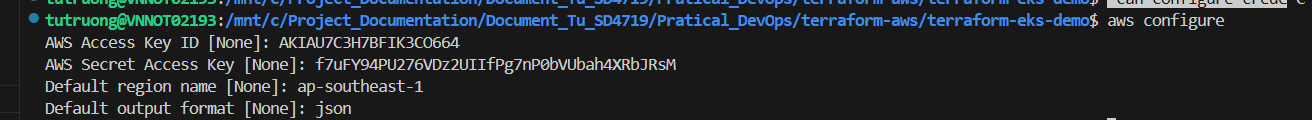


* **Configure Terraform CLI, r**un the command:

aws configure

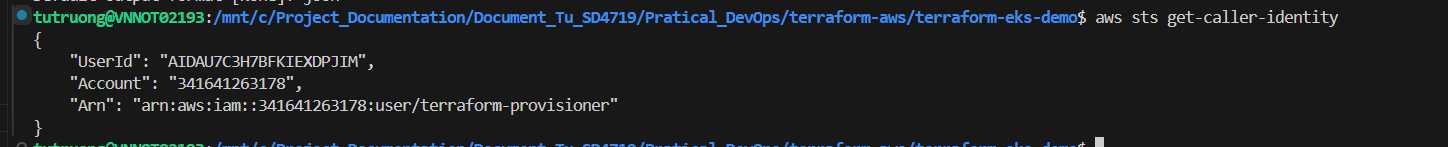
Then input:

* + Access key ID
  + Secret access key
  + Region: ap-southeast-1
  + Output: json



* Verify Terraform Access, run the command:

aws sts get-caller-identity

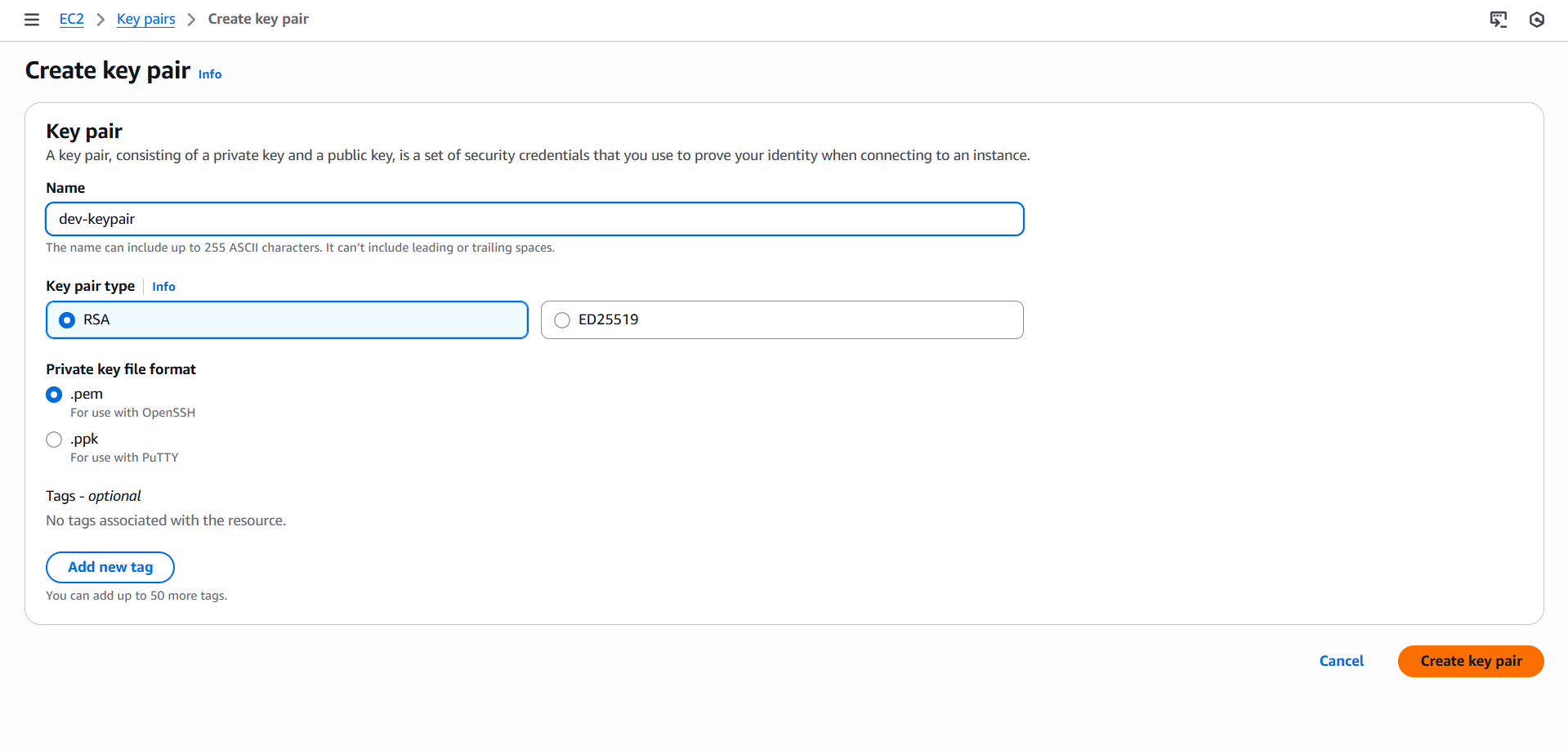


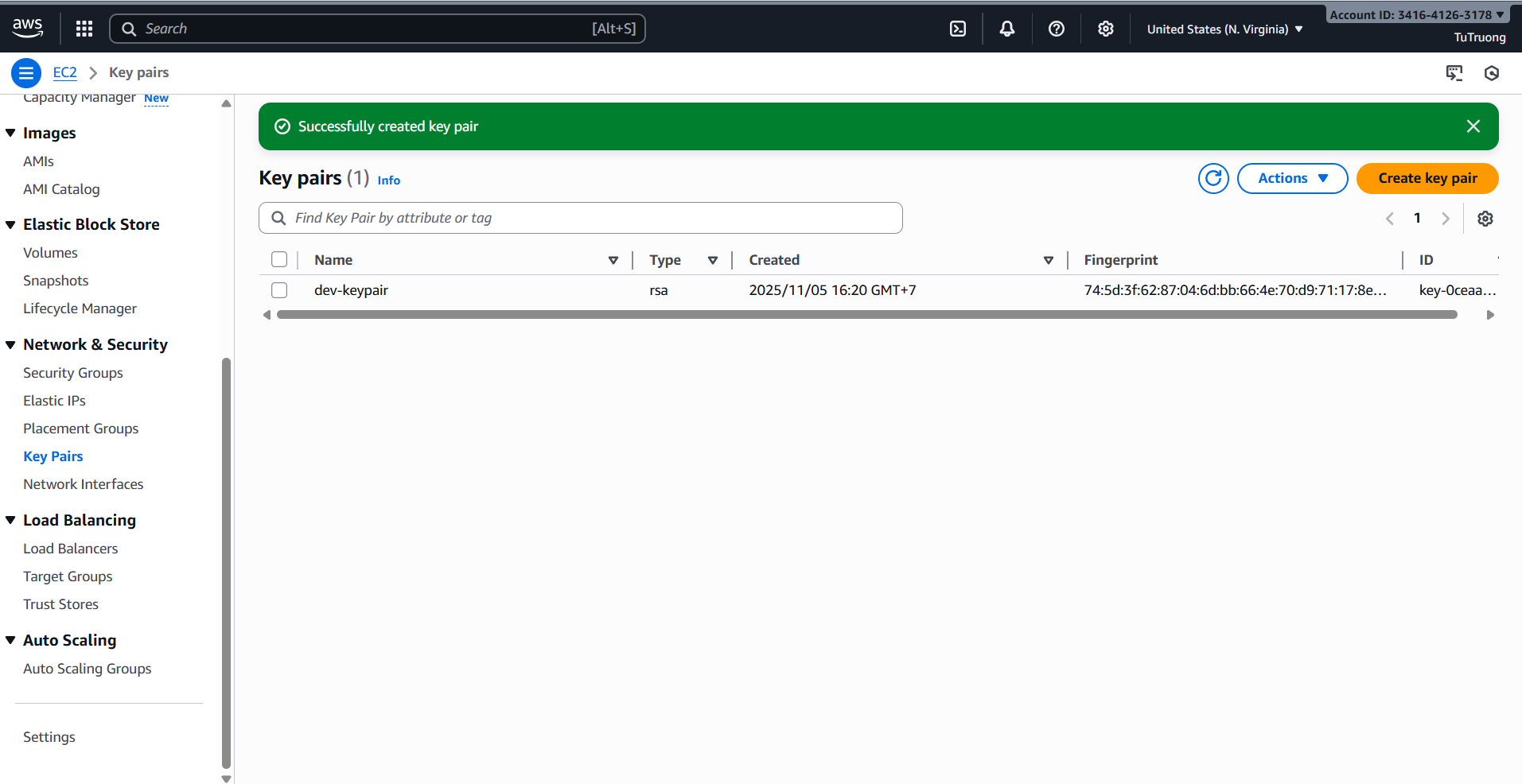
1. Verify Terraform AWS provider can authenticate:

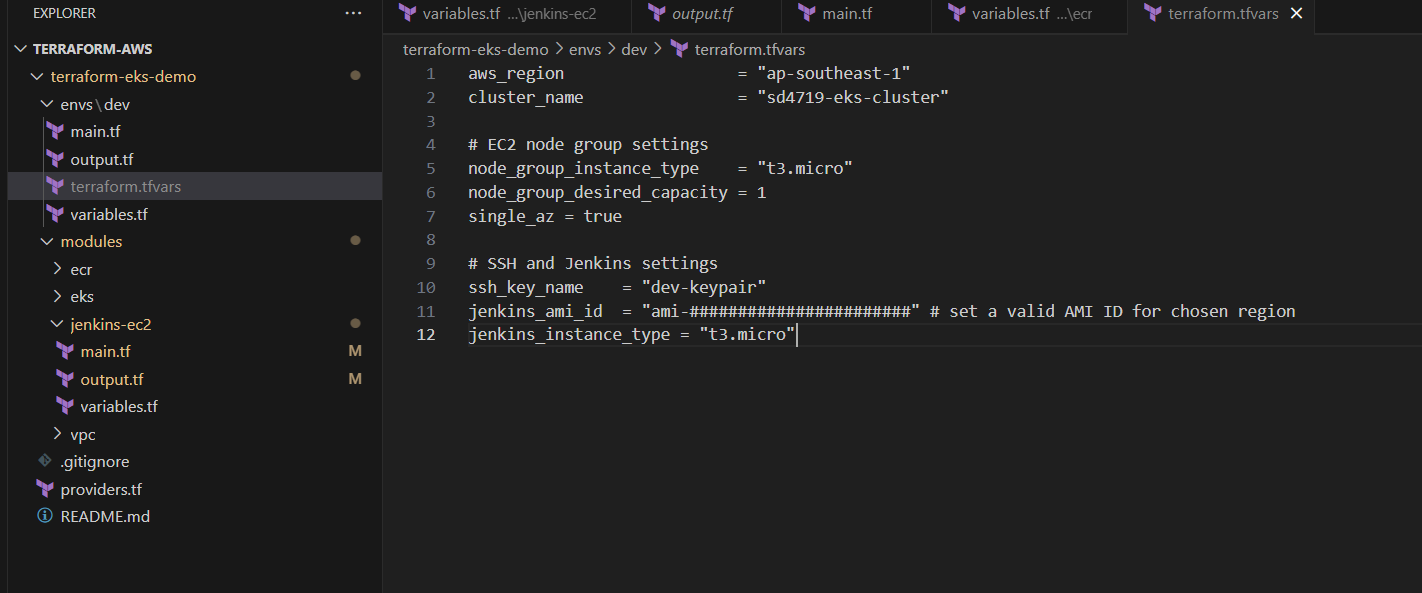
**- Create a new key pair:**

* 1. **In AWS Console, search for EC2 => Key Pairs => Create key pair**.
  2. Name: **dev-keypair**.
  3. Choose **RSA** and **.pem** format (for Linux/macOS) or **.ppk** (for Windows + PuTTY).
  4. Download and store the key securely (need to SSH later).
  5. Then use that key name in **terraform.tfvars**:

ssh\_key\_name = "dev-keypair"





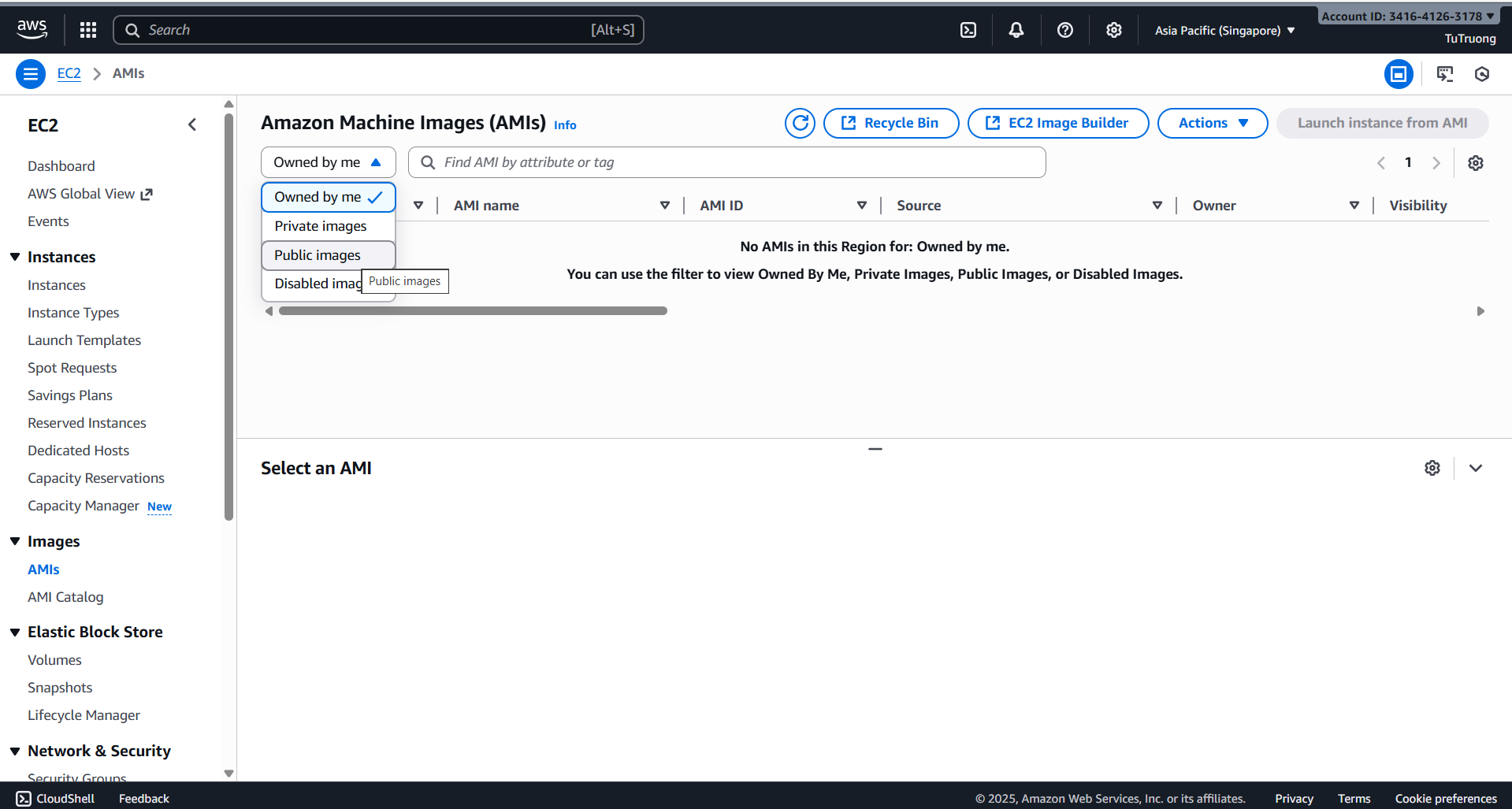


- **Find a valid AMI ID for region (ap-southeast-1):**

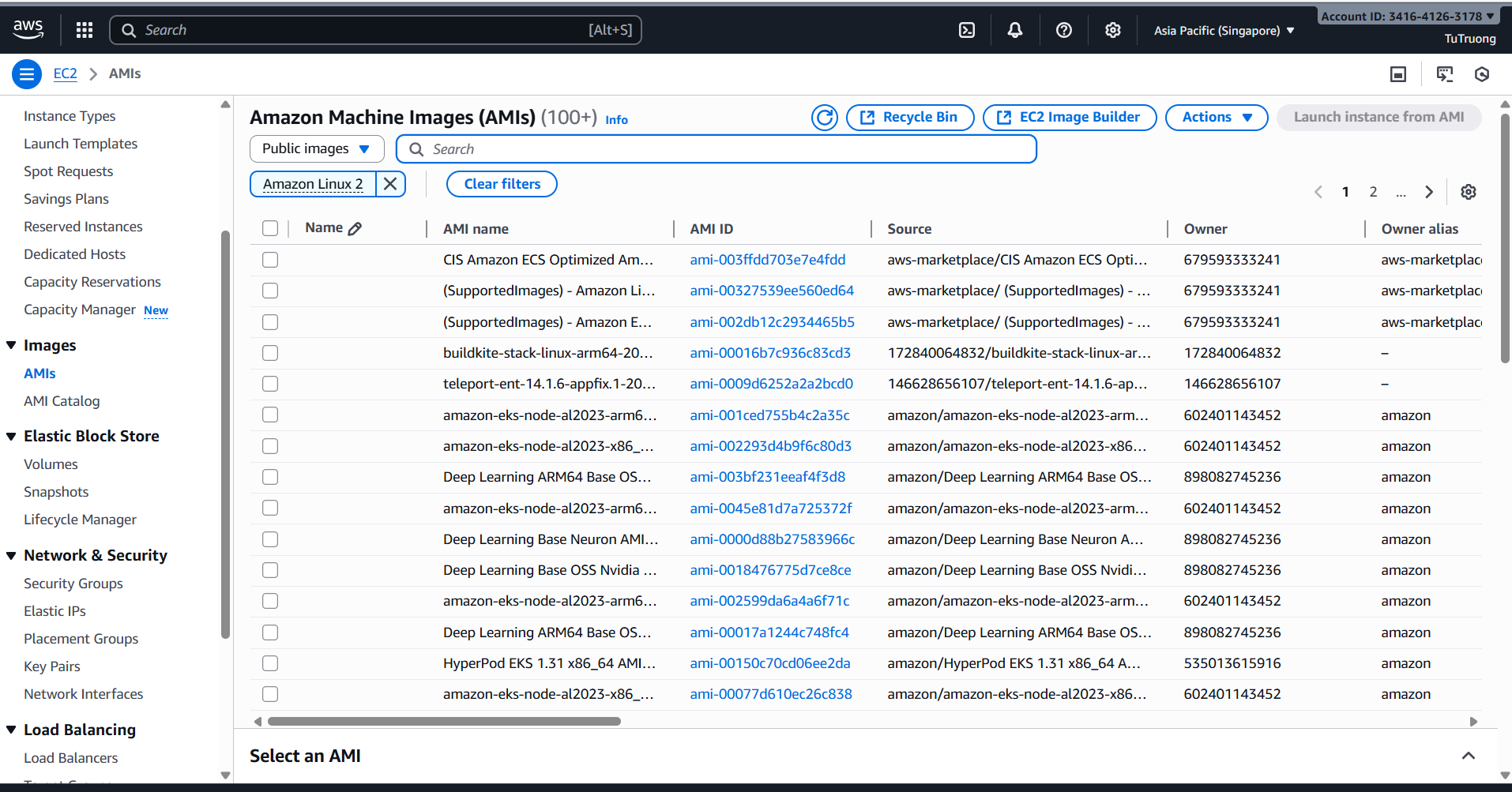
**+ Solution 1: Search for official AMIs (Public) on AWS Console**

**1. In AWS Console, search for EC2 => Images => AMIs.**

2. In the top filter bar, choose the dropdown **Owned by me =>** select **Public images**.



3. Search for “Amazon Linux 2” => select the right AMI based on the official publisher (Amazon)

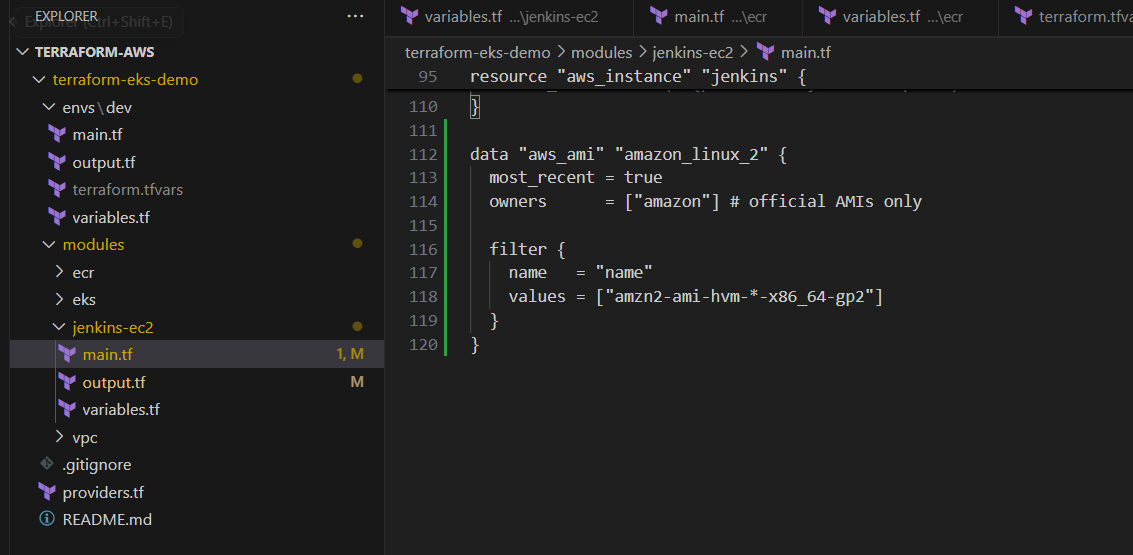


4. Copy and set in terraform.tfvars

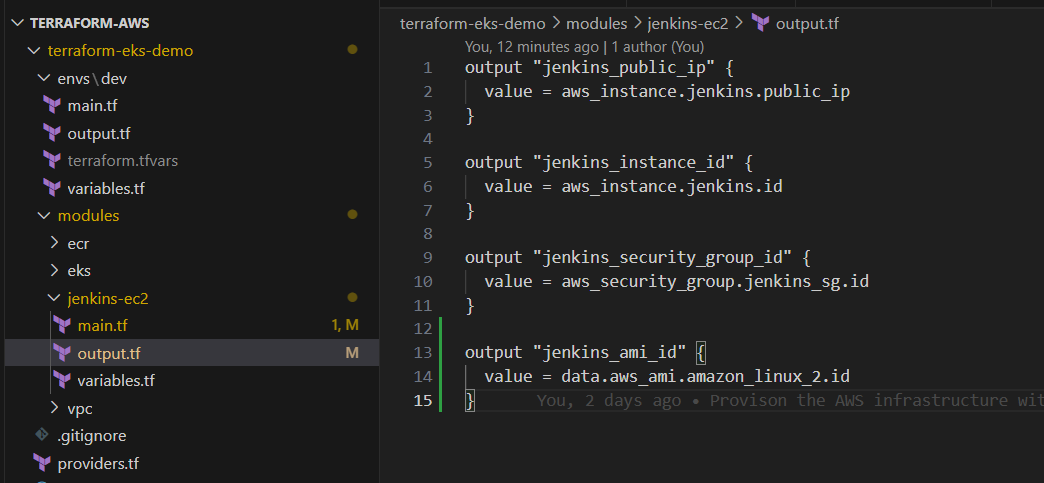
jenkins\_ami\_id = "ami-XXXXXXXXXXXXXXXXXX"

+ **Solution 2 (I prefered): Automatically Lookup Latest Amazon Linux 2 AMI**

1. Use this **Terraform data source** in **jenkins\_ec2/main.tf**:



2. In the **jenkins\_ec2 module**, add the block output "jenkins\_ami\_id" into the **output.tf**:

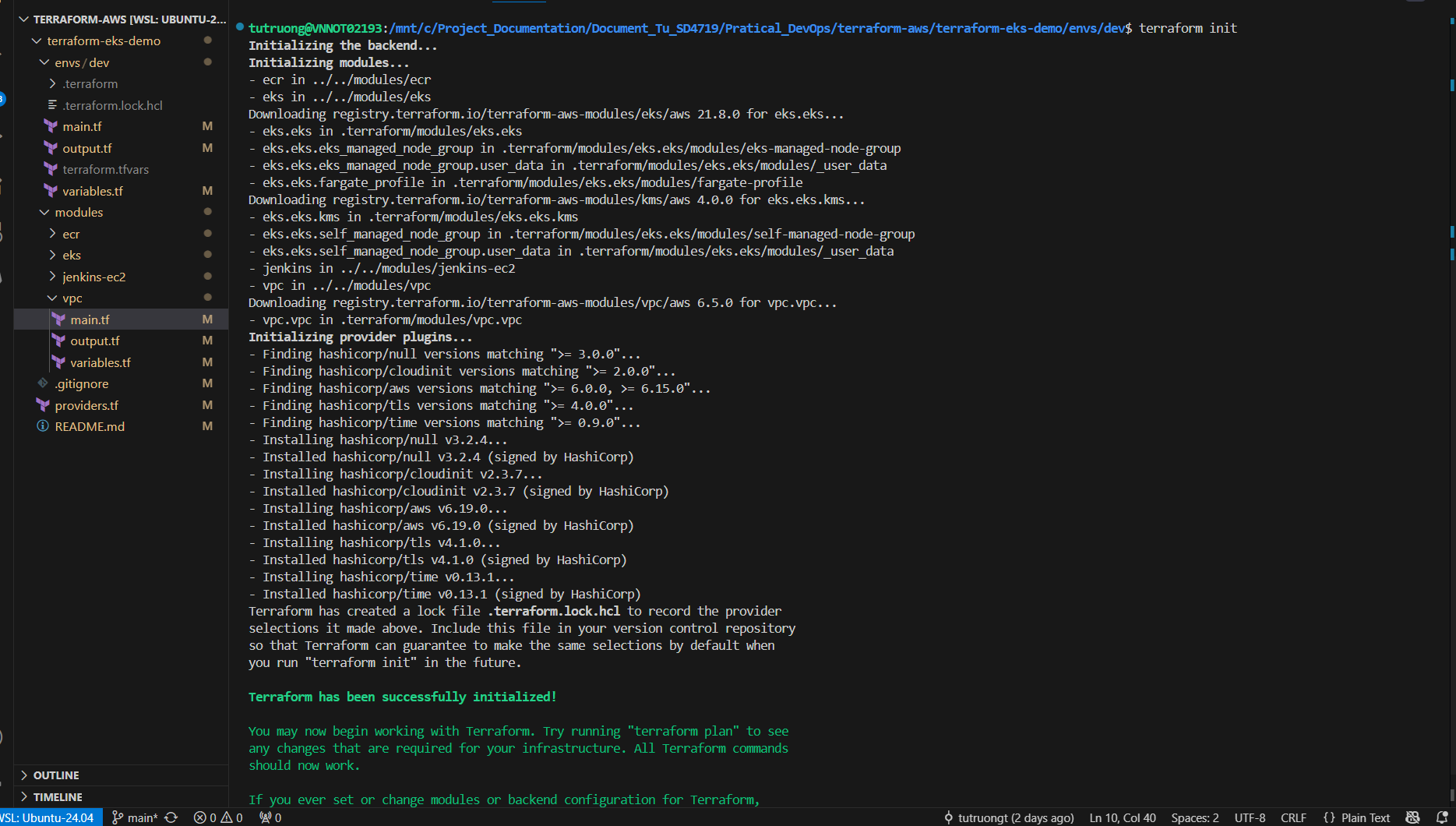


3. After run the command “**terraform apply”**, Terraform will print the output **jenkins\_ami\_id** then we copy and paste it into the variable **jenkins\_ami\_id** in **terraform.tfvars** file**:**

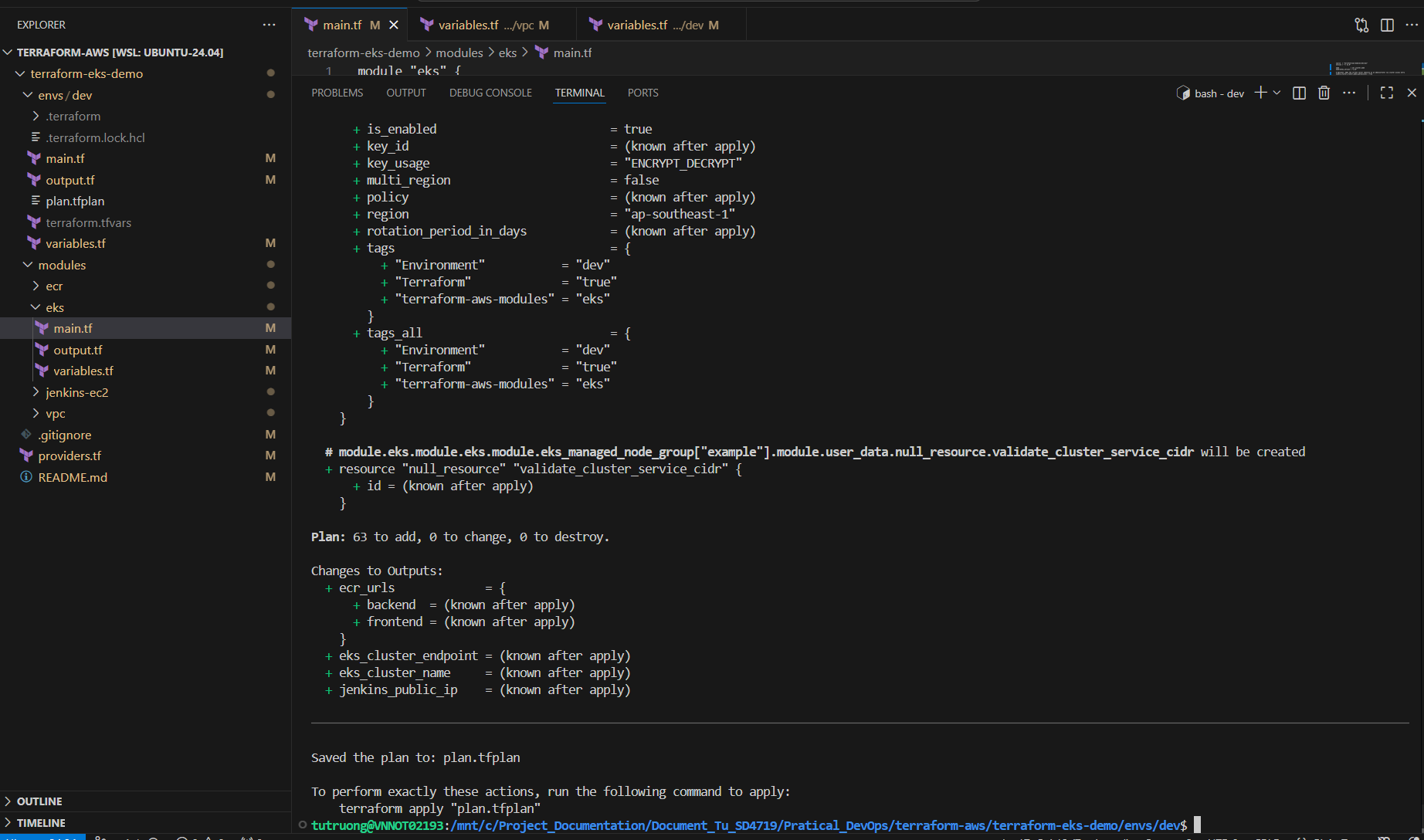
jenkins\_ami\_id = "ami-XXXXXXXXXXXXXXXXXX"

- In the Terraform project (**terraform-eks-demo/envs/dev**), run the command:

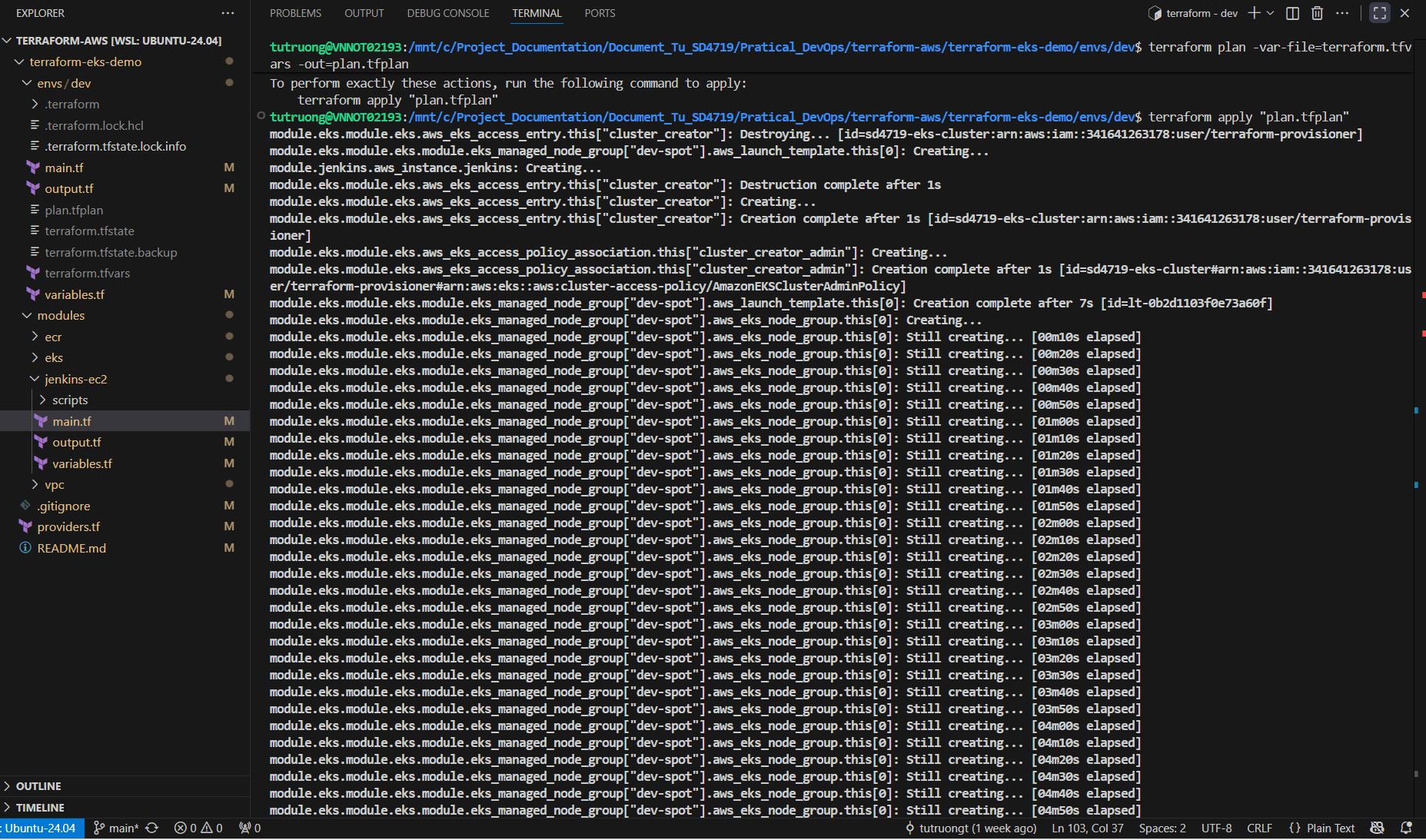
terraform init

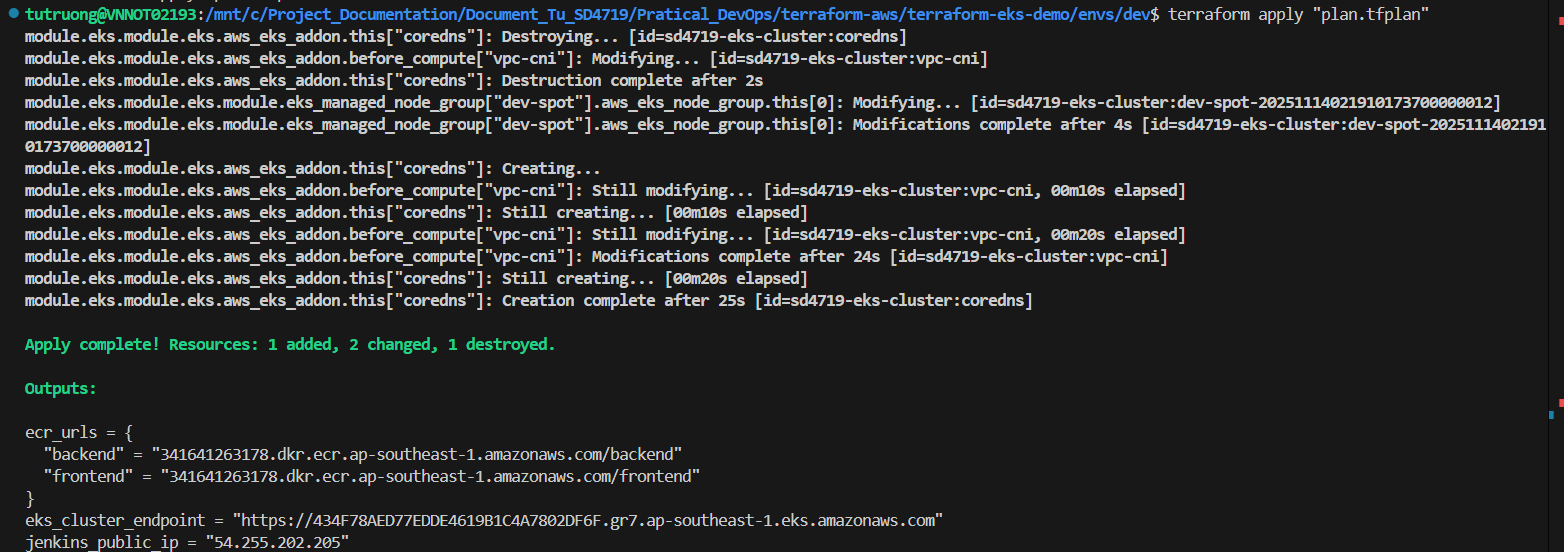


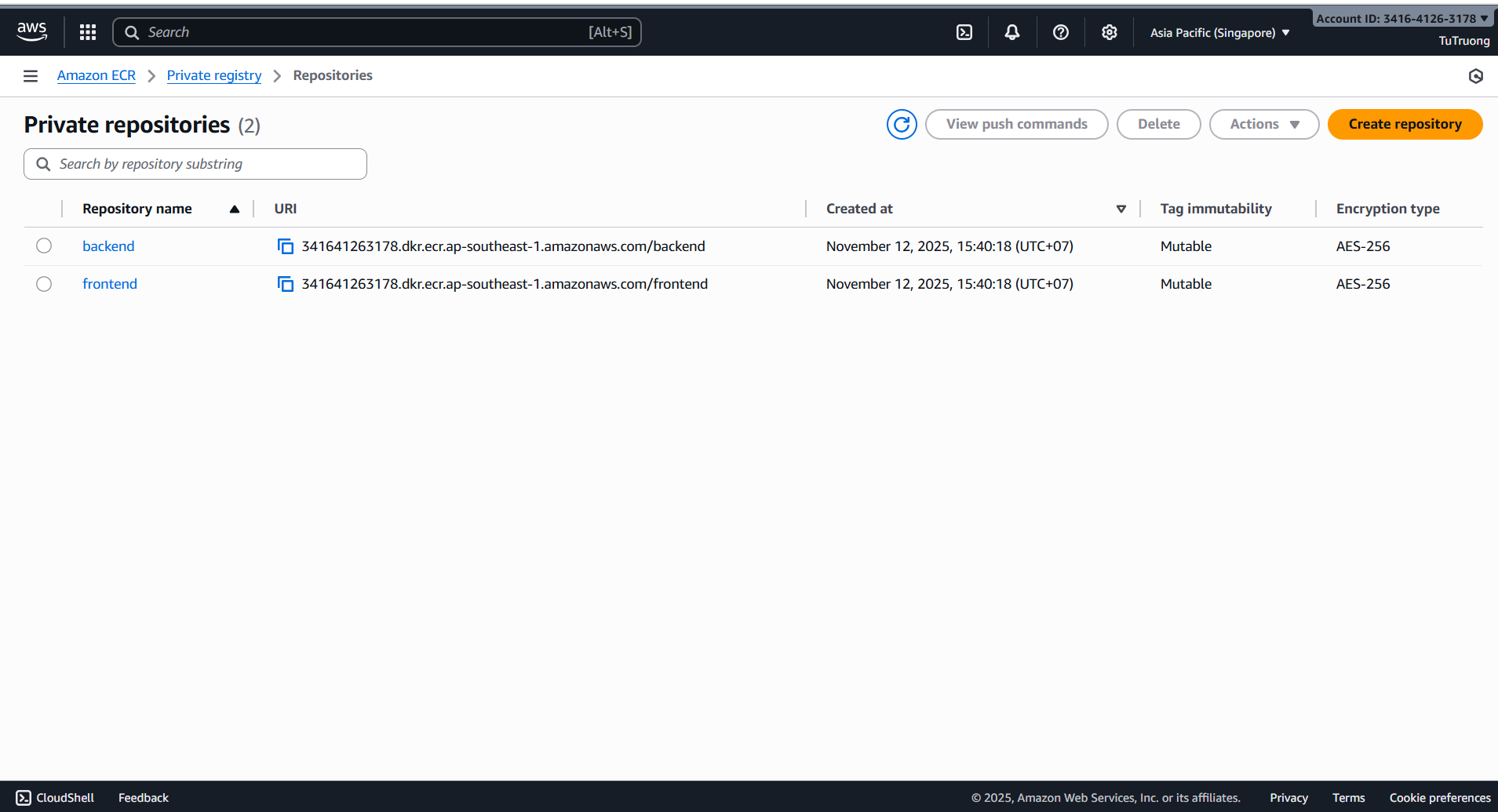
terraform plan -var-file=terraform.tfvars -out=plan.tfplan

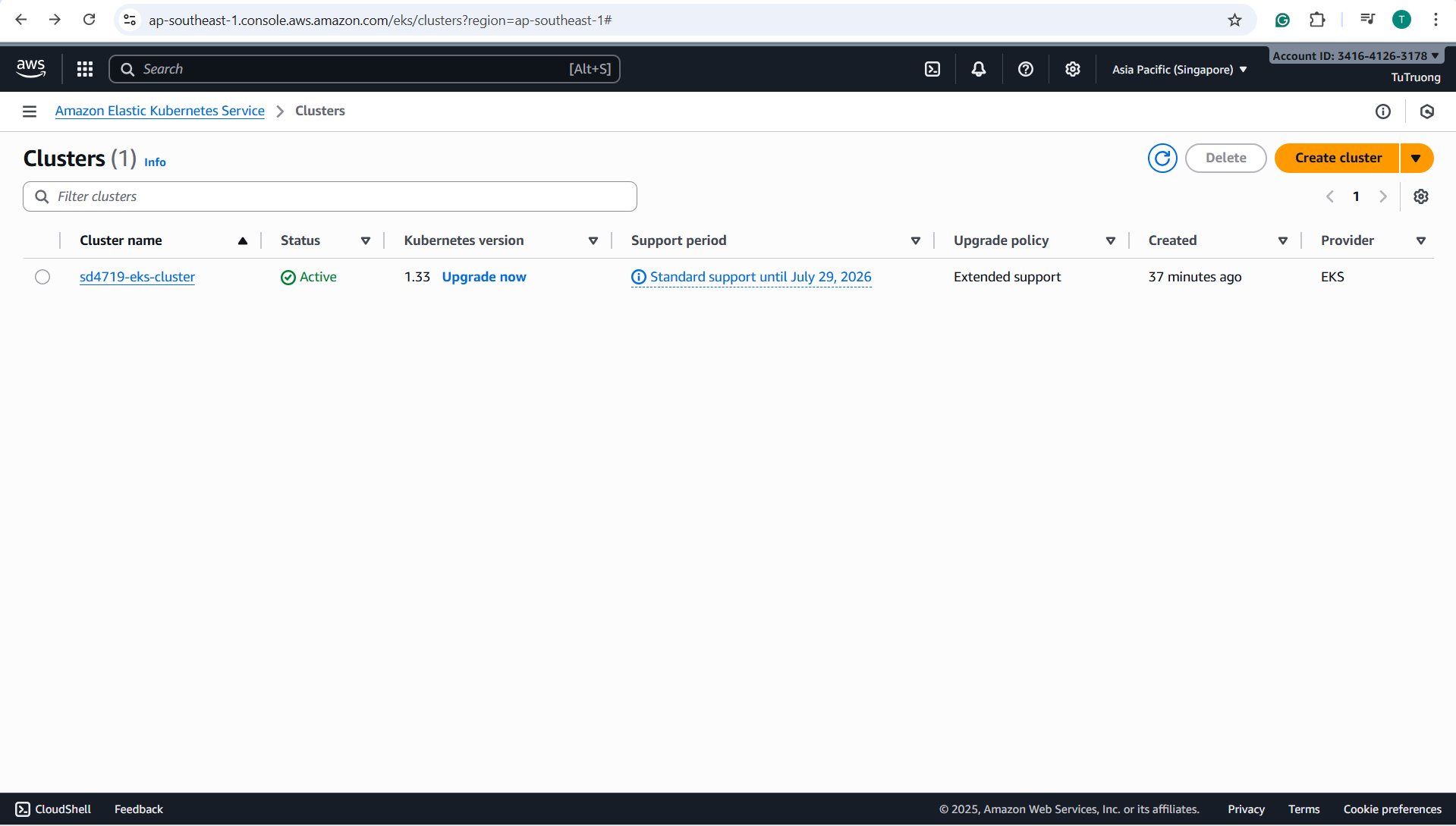


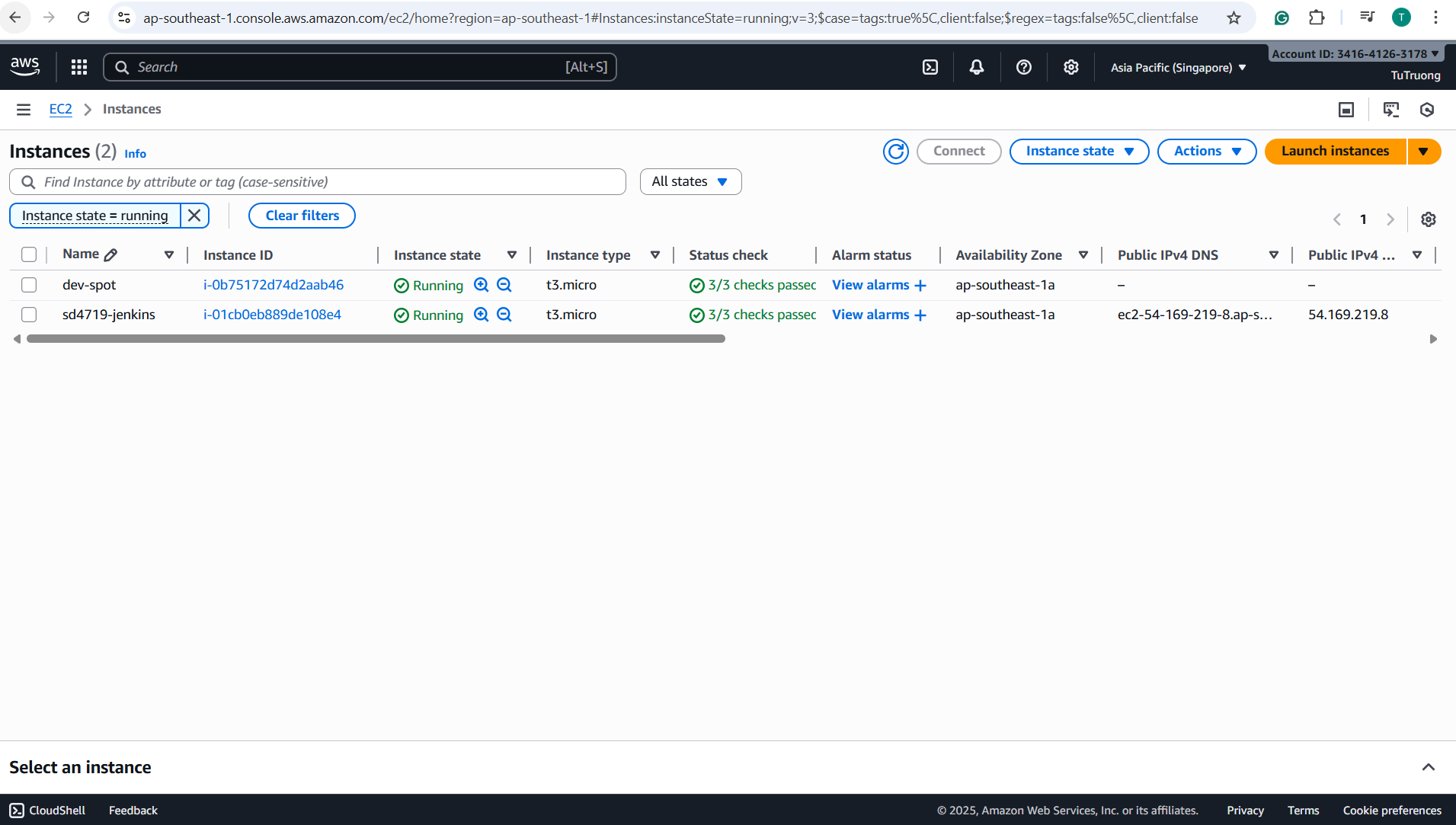
terraform apply plan.tfplan

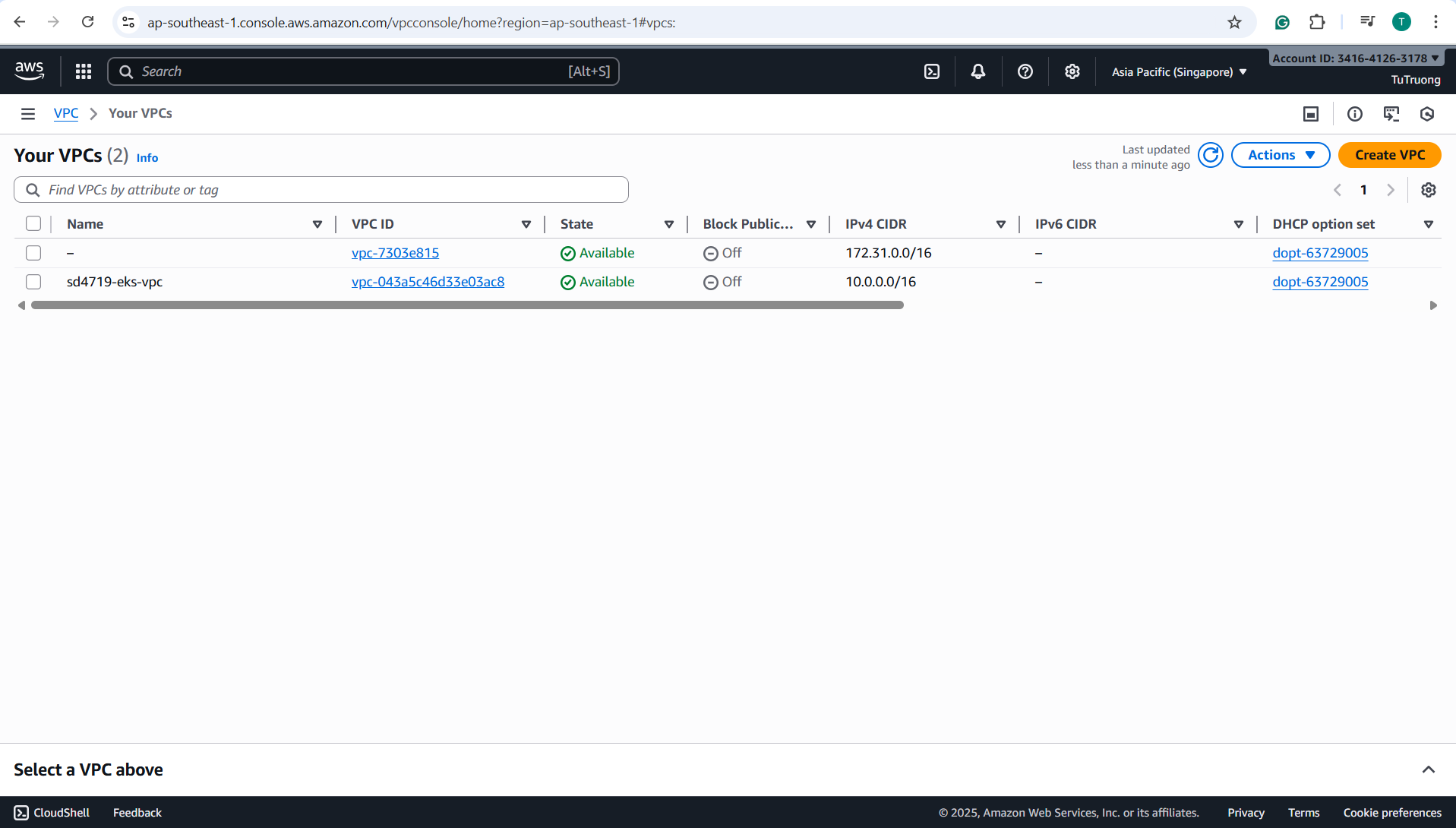








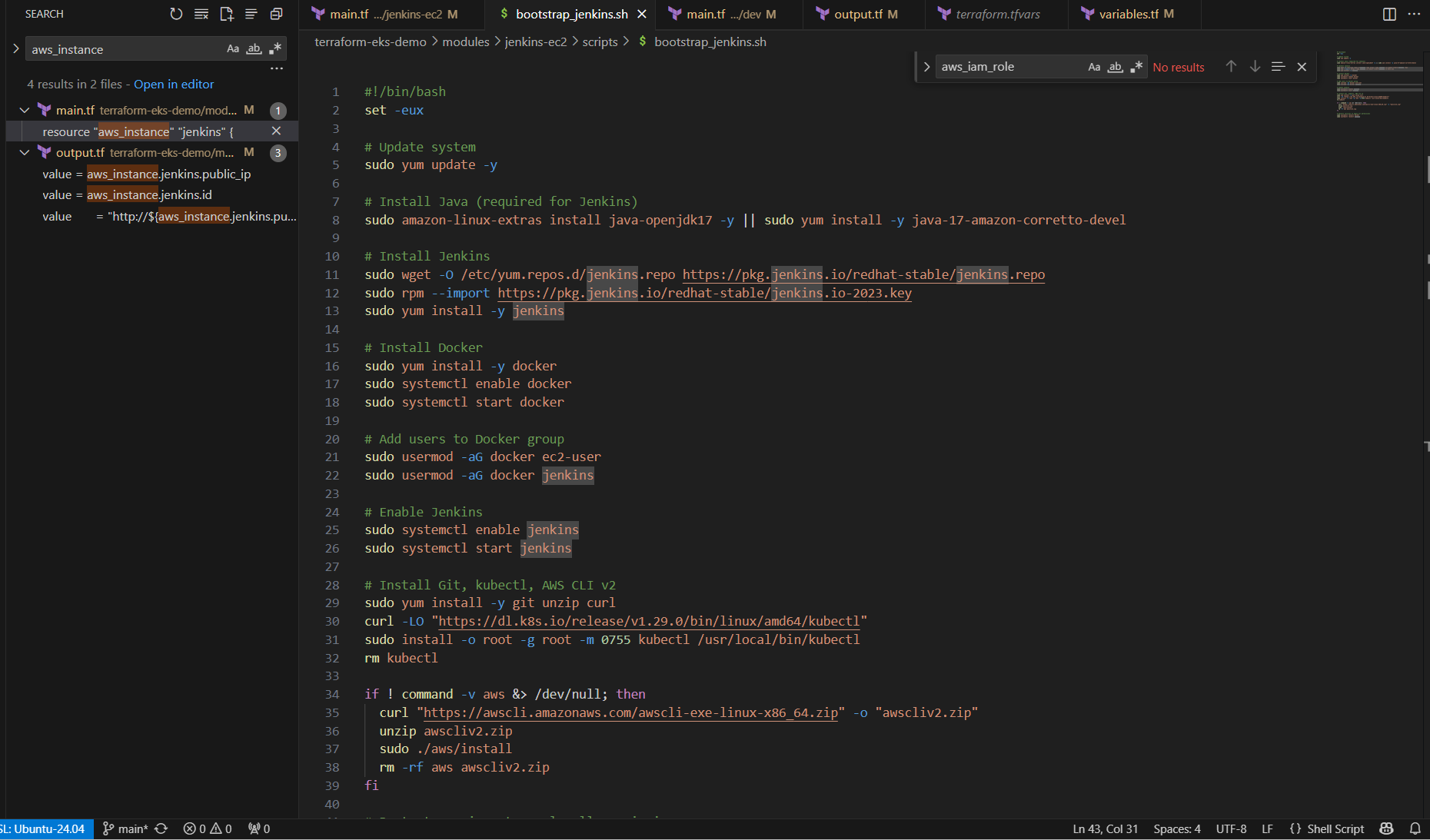




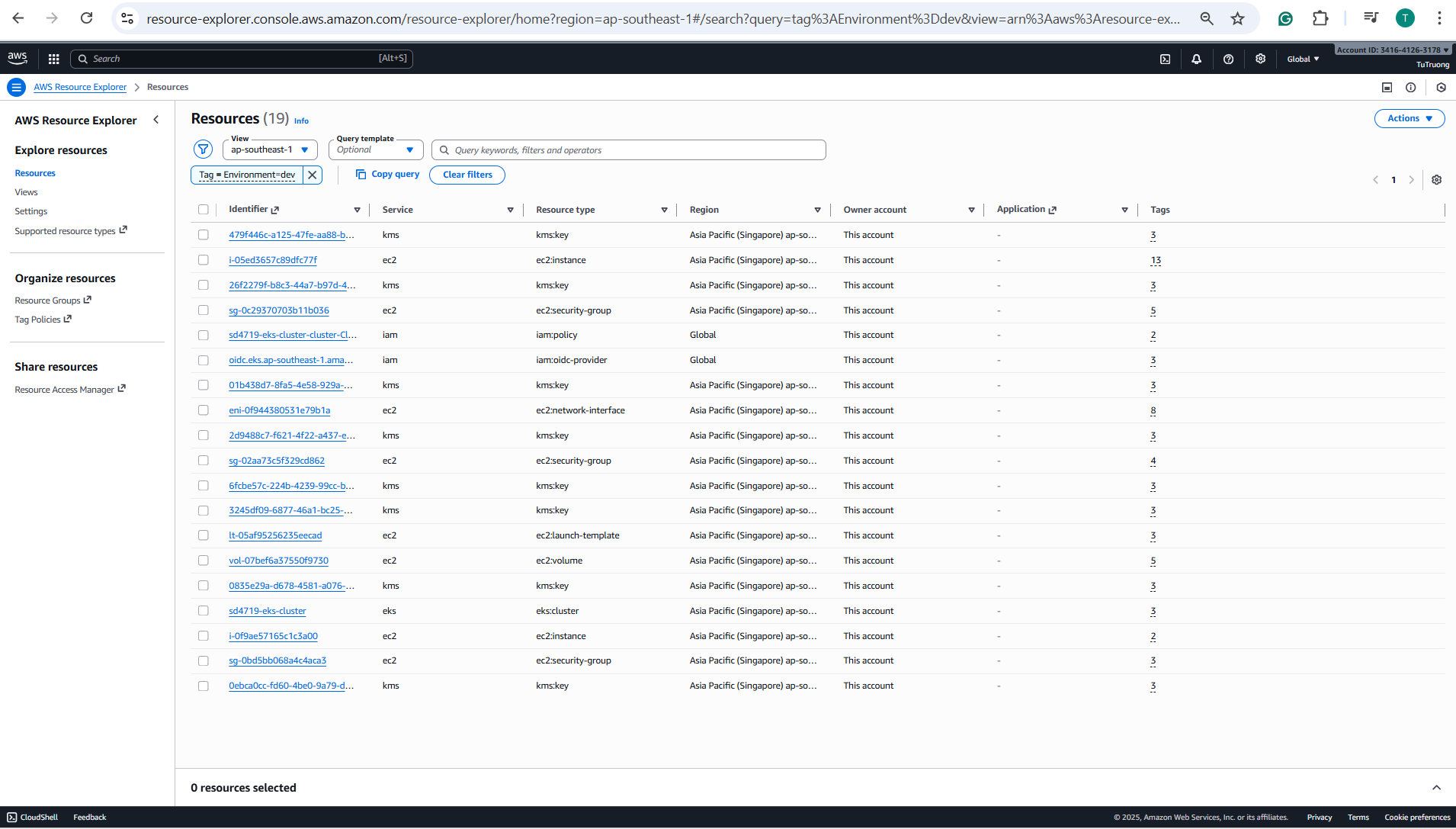
1. Use a **user\_data** script to install Jenkins/Docker automatically during Terraform provisioning:

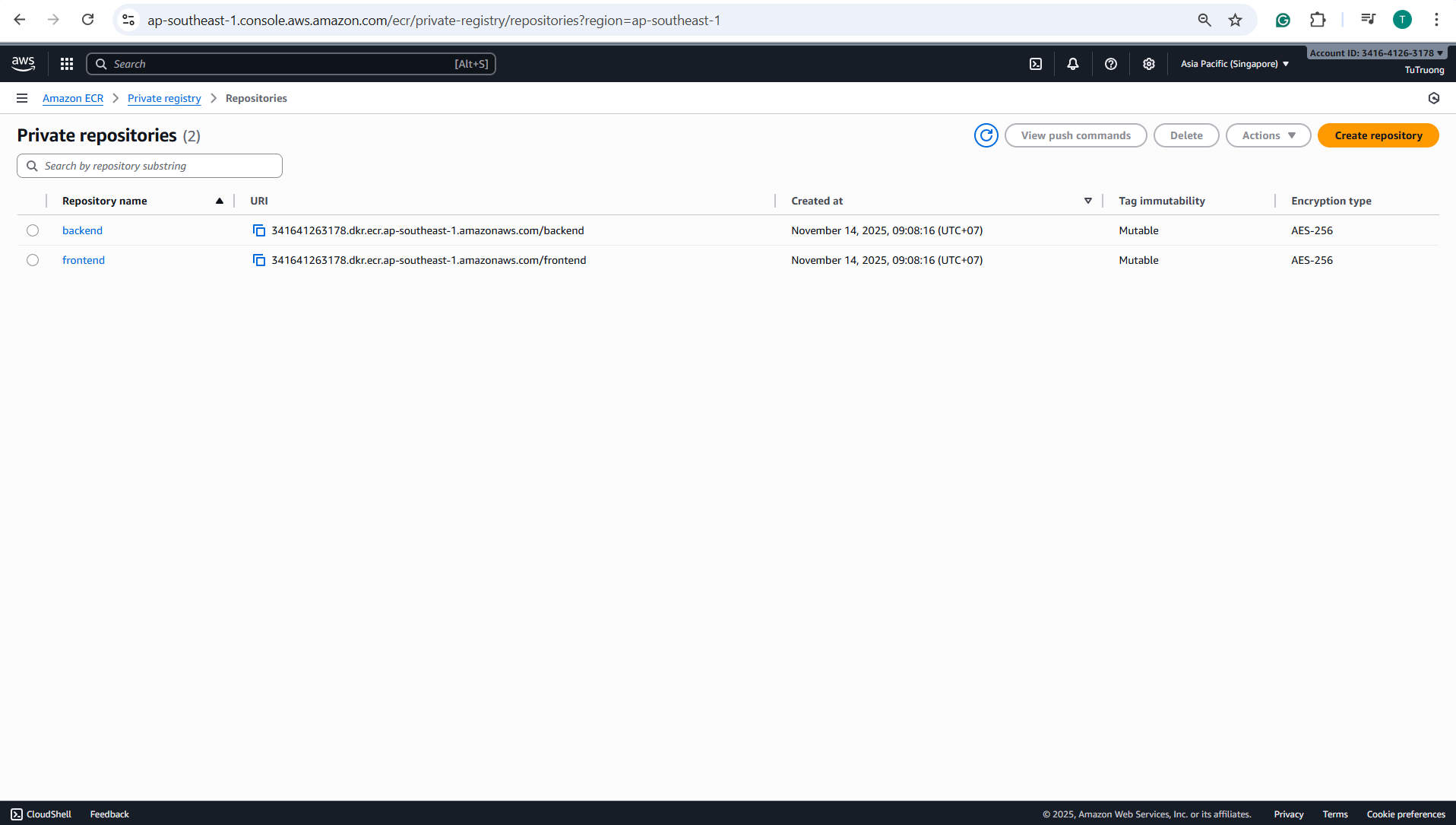
- Provisions EC2 instance and passes a small **user\_data** script (an external bootstrap script) to install **Docker + Jenkins** base:

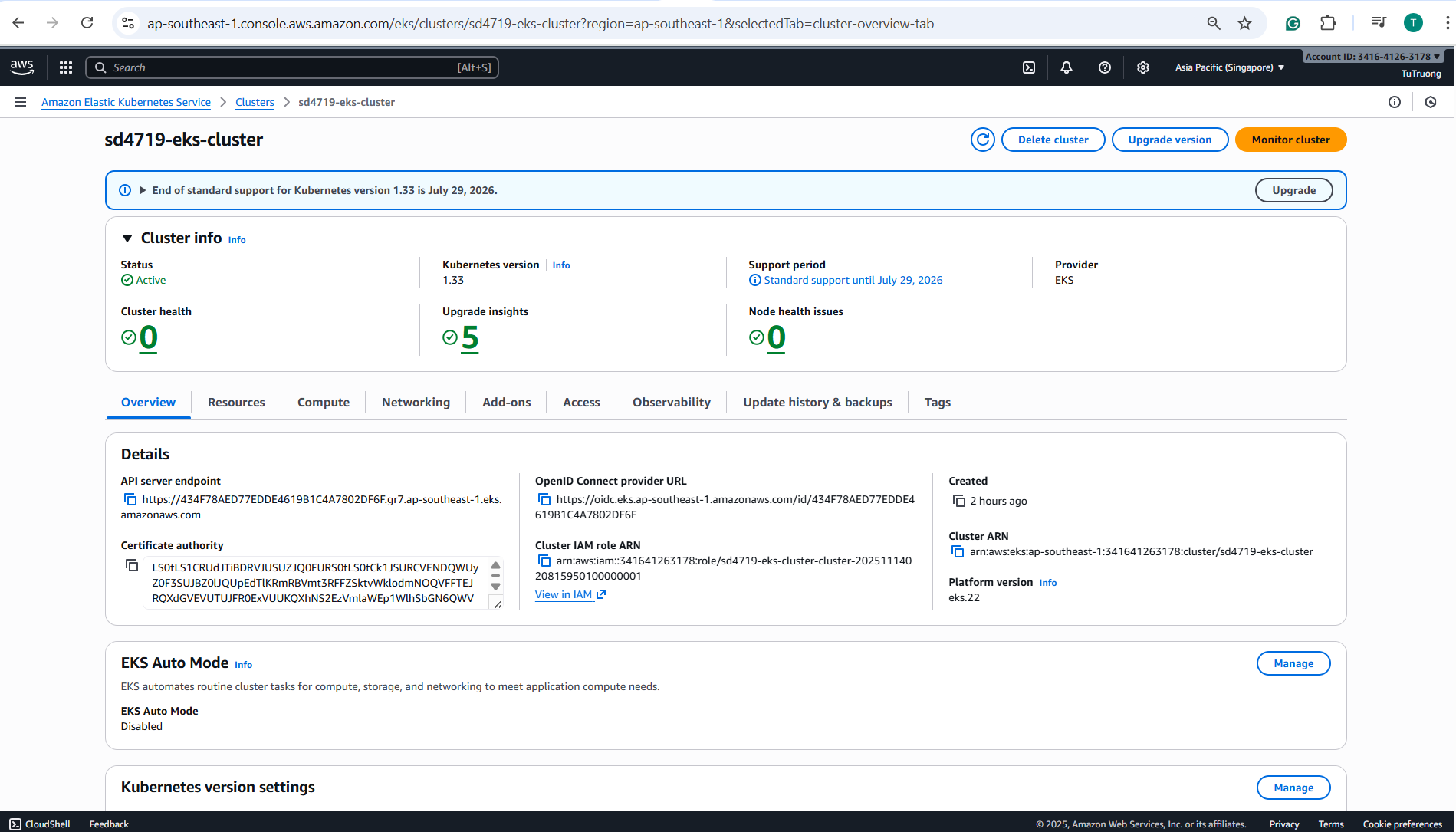
user\_data = file("${path.module}/scripts/bootstrap\_jenkins.sh")

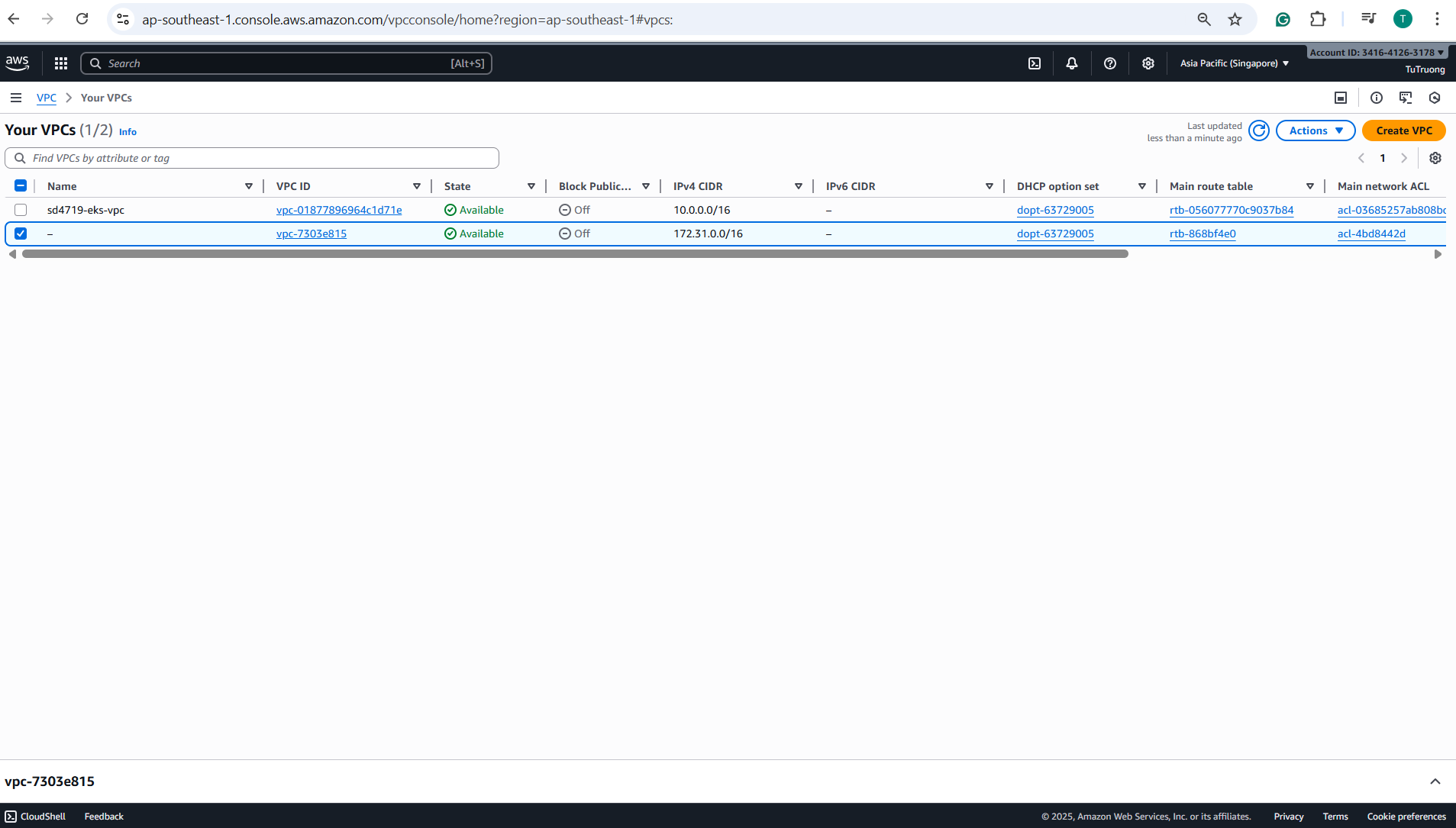


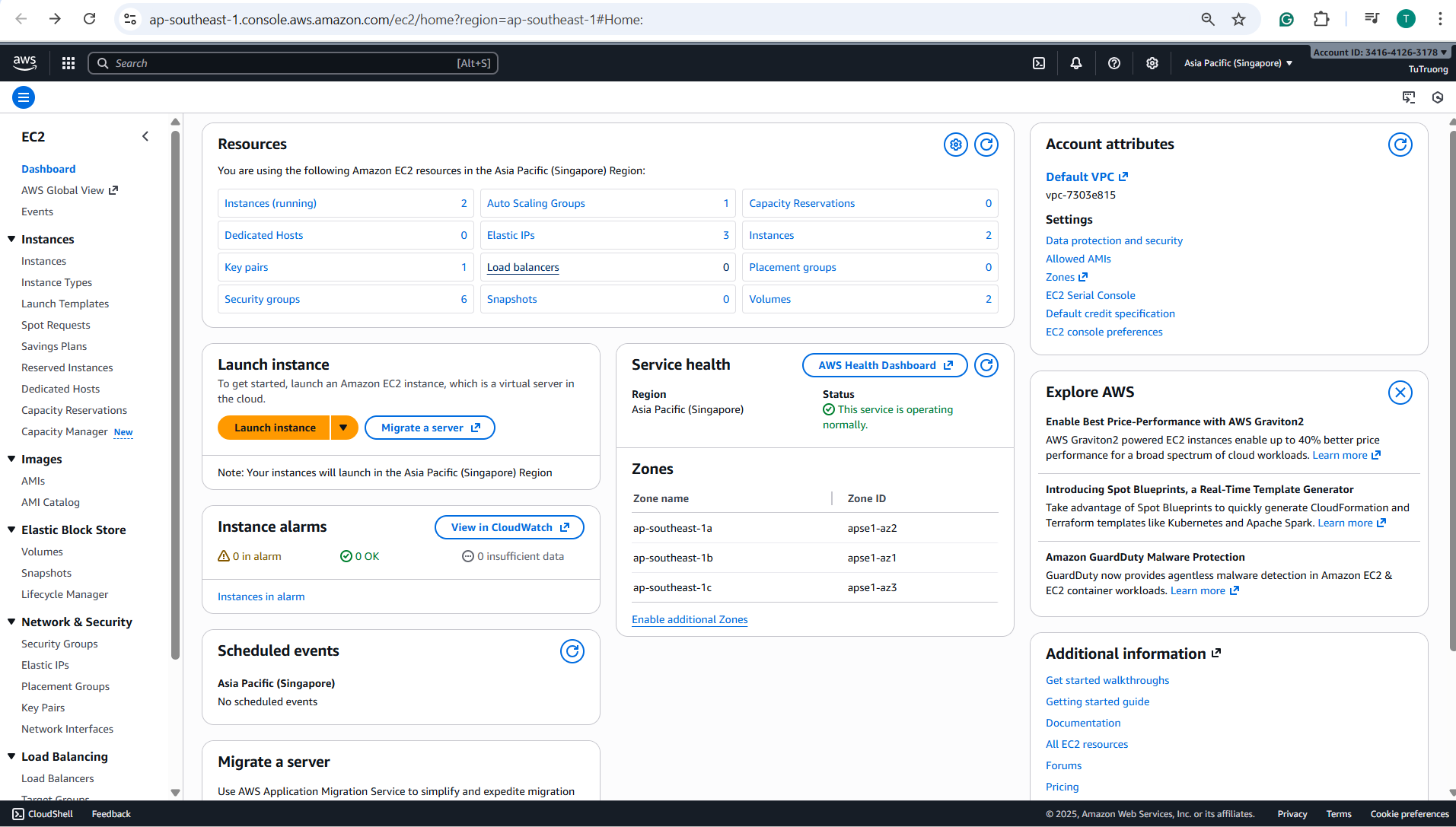
- **After Terraform creates the EC2 instance => EC2 executes the user\_data script automatically => **Docker/Jenkins** installation happens inside the EC2 instance during its first boot => the instance is ready:** Visit **http://54.255.202.205:8080**

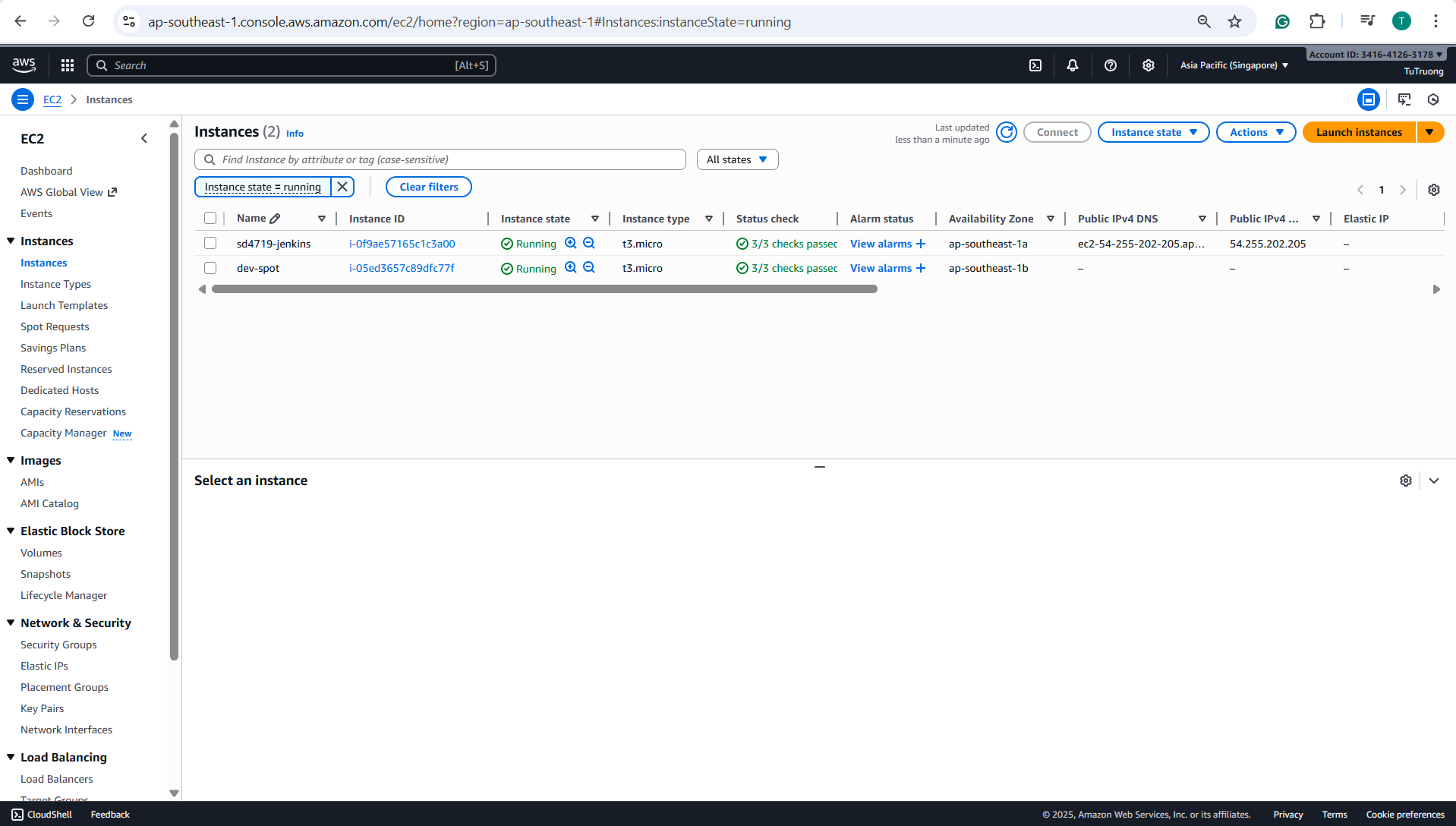


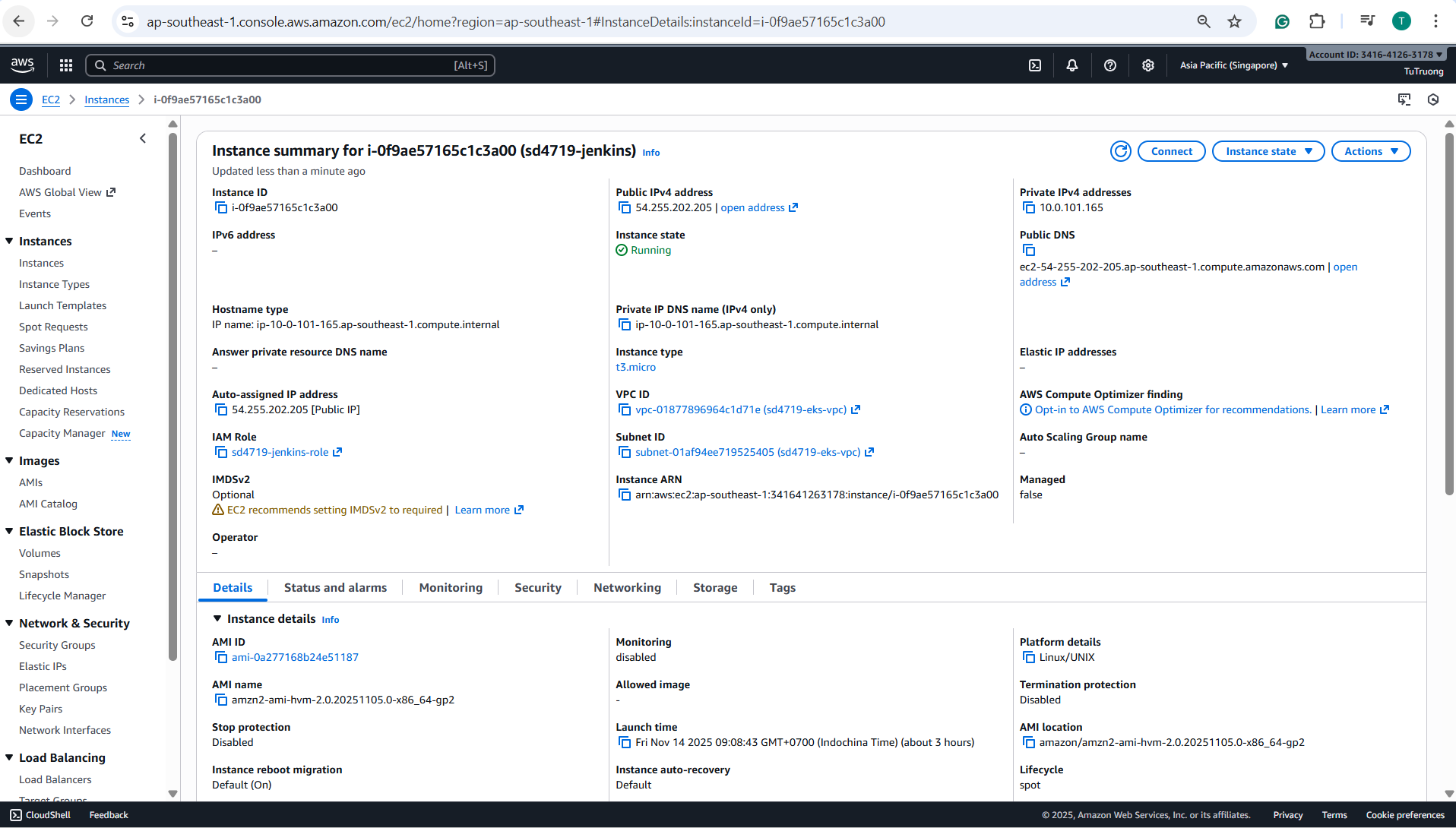




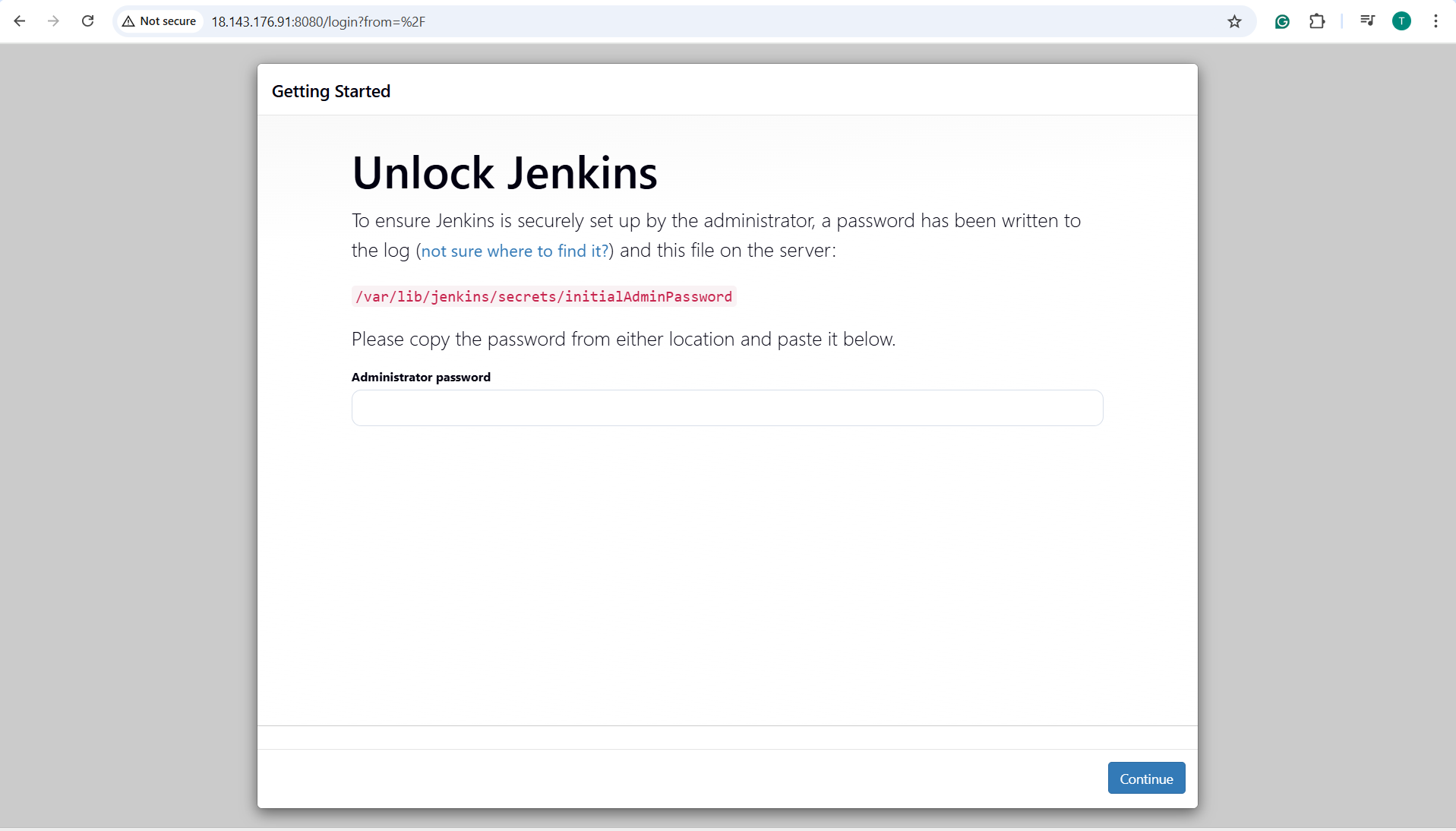








1. **Unlock** and complete the **CI/CD pipeline setup** with **Jenkins:**



**- **Get the Jenkins initial admin password: use SSH into the Jenkins EC2 instance:****

**+ Move the PEM file into Linux home folder, run with:**

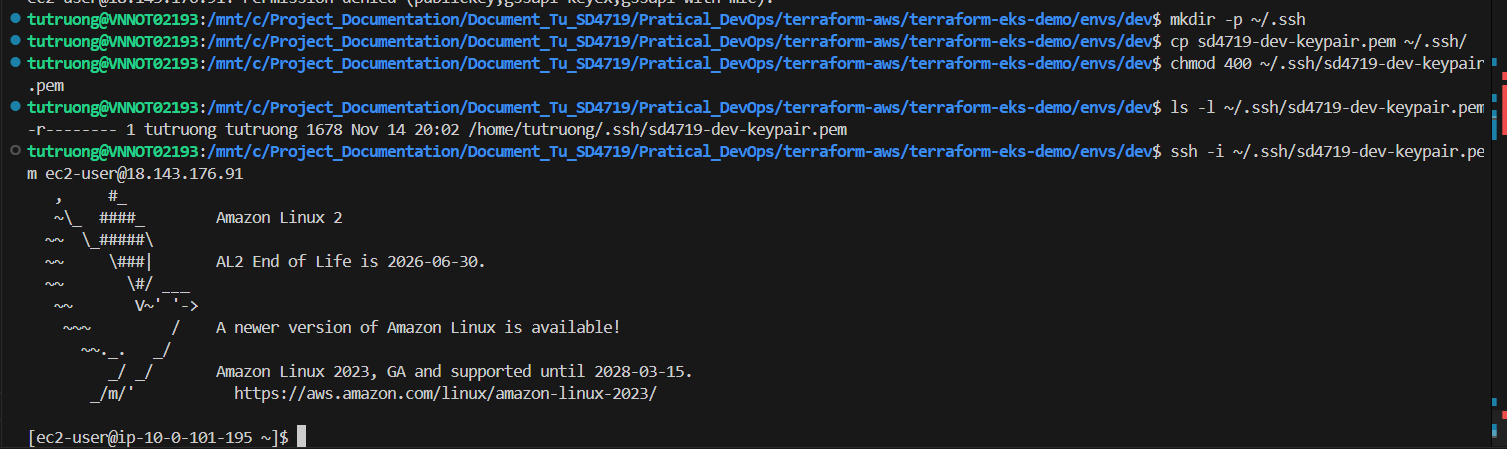
***mkdir -p ~/.ssh***

***cp sd4719-dev-keypair.pem ~/.ssh/***

***chmod 400 ~/.ssh/sd4719-dev-keypair.pem***

**+ Verify sd4719-dev-keypair.pem in the .ssh folder, run with:  
 *ls -l ~/.ssh/sd4719-dev-keypair.pem***

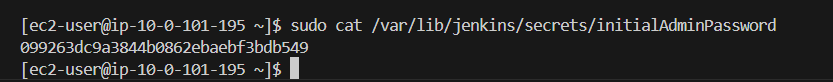
**Then run: *ssh -i ~/.ssh/sd4719-dev-keypair.pem ec2-user@18.143.176.91***



**+ Get the password, run with:**

***sudo cat /var/lib/jenkins/secrets/initialAdminPassword***

***=>* ***output: 099263dc9a3844b0862ebaebf3bdb549*****

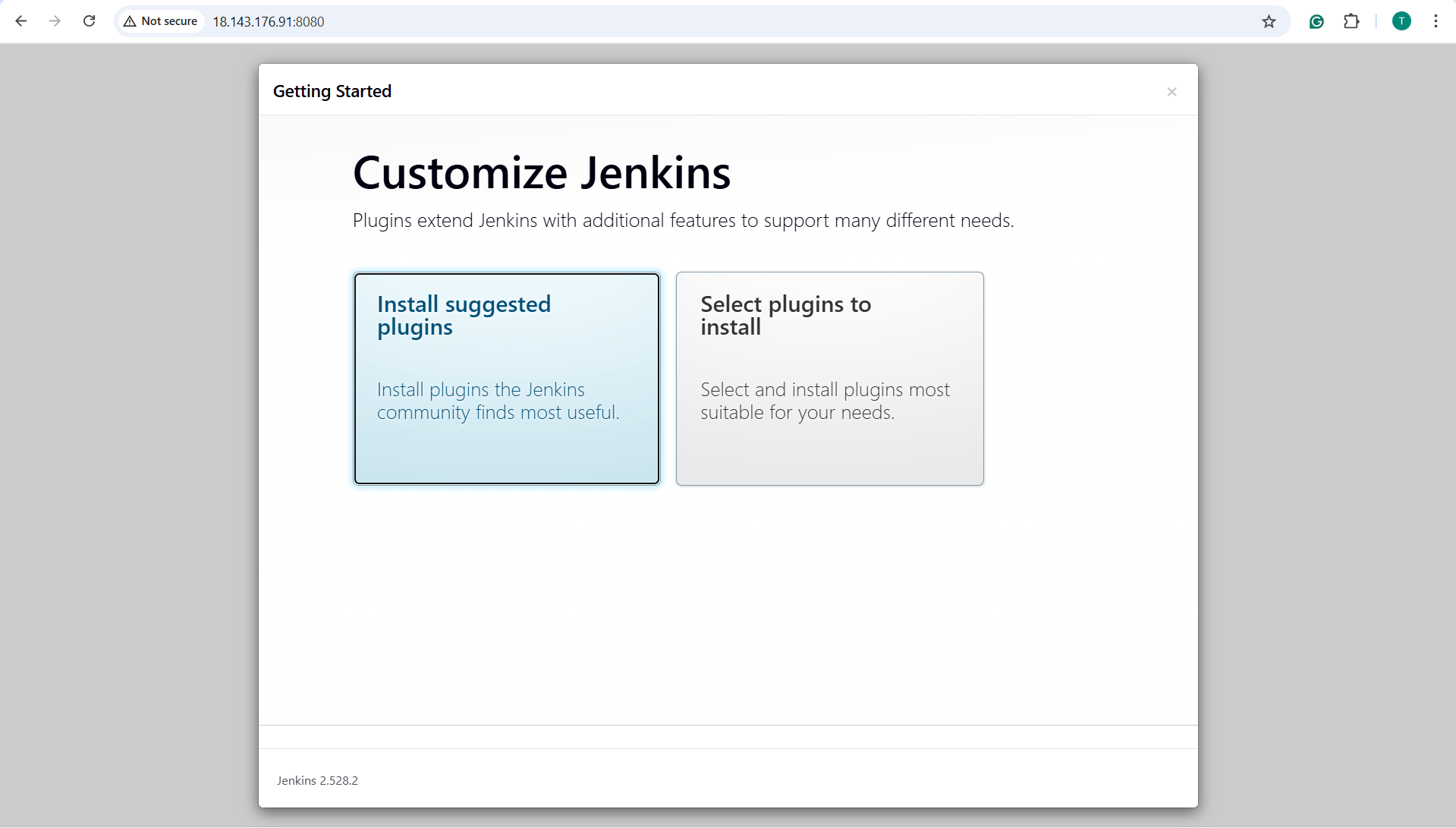


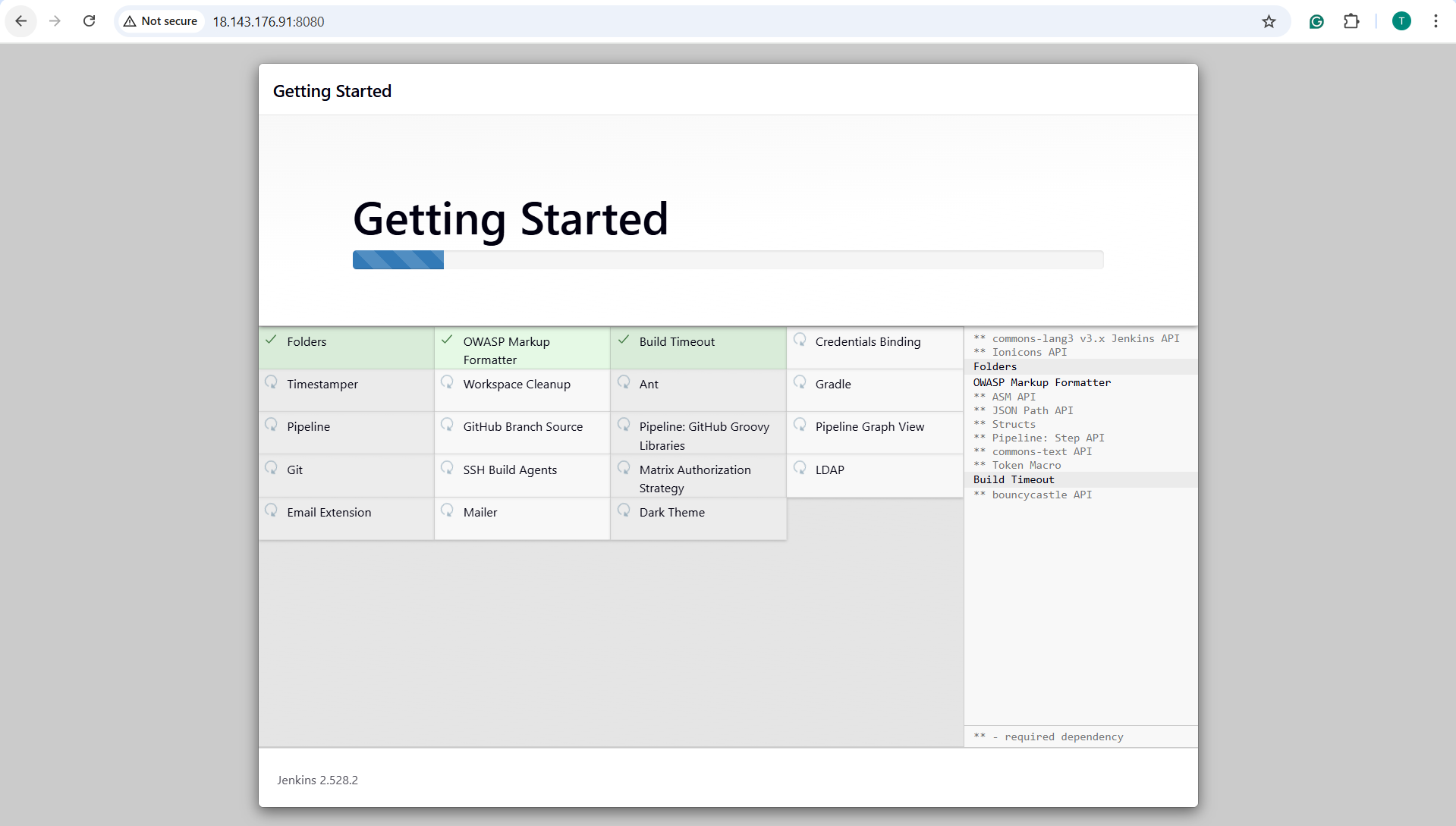
***+* Copy that value and paste it into the web form (**Administrator password**) => click Continue**

**- **Install Jenkins recommended plugins:****

**Choose “Install suggested plugins” — this installs common ones like:**

* + Git
  + Pipeline
  + GitHub integration
  + Credentials binding
  + SSH Build Agents
  + Docker Pipeline

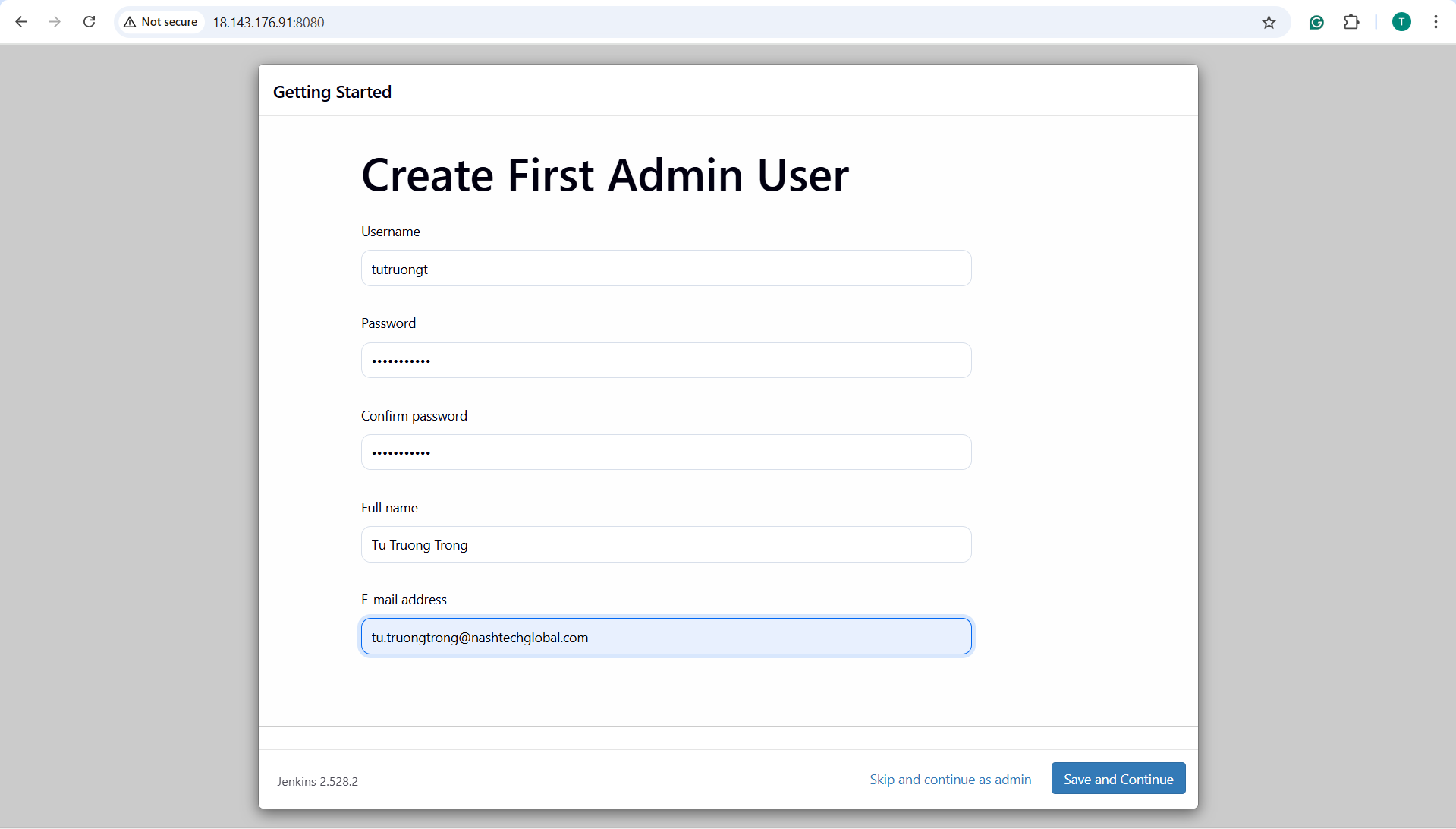


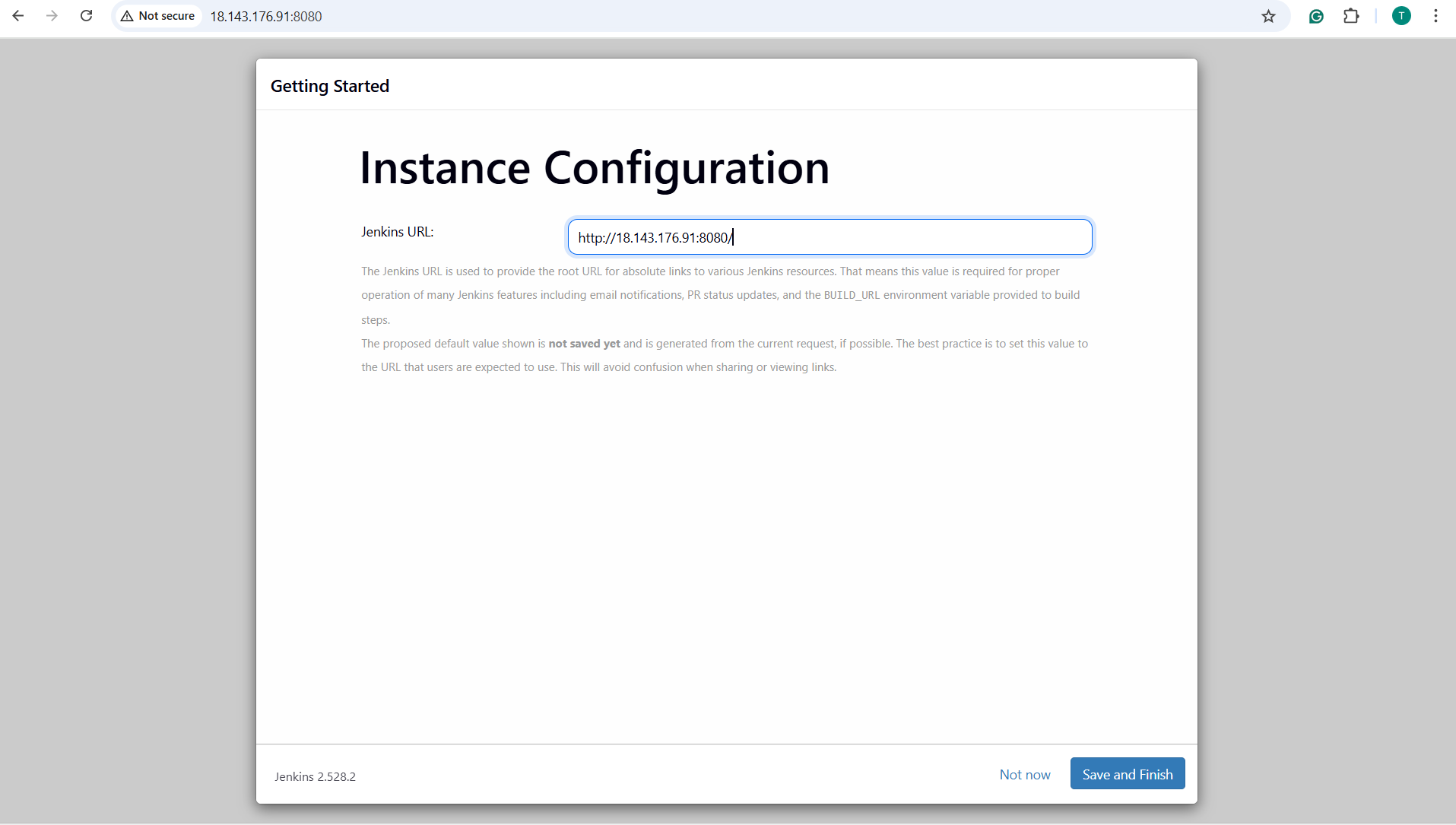


## ****- Create the first admin user:****

Once plugins are installed, Jenkins will ask to:

* + Create a username, password, and email.
  + Save and continue.



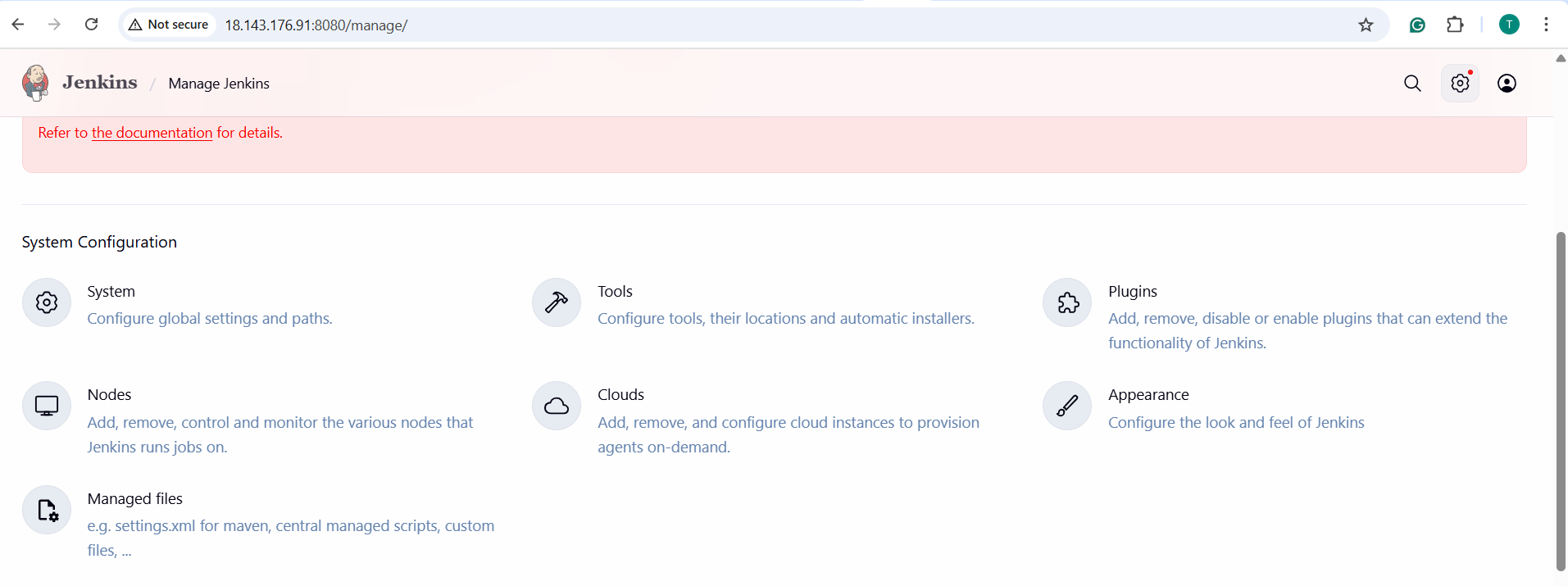


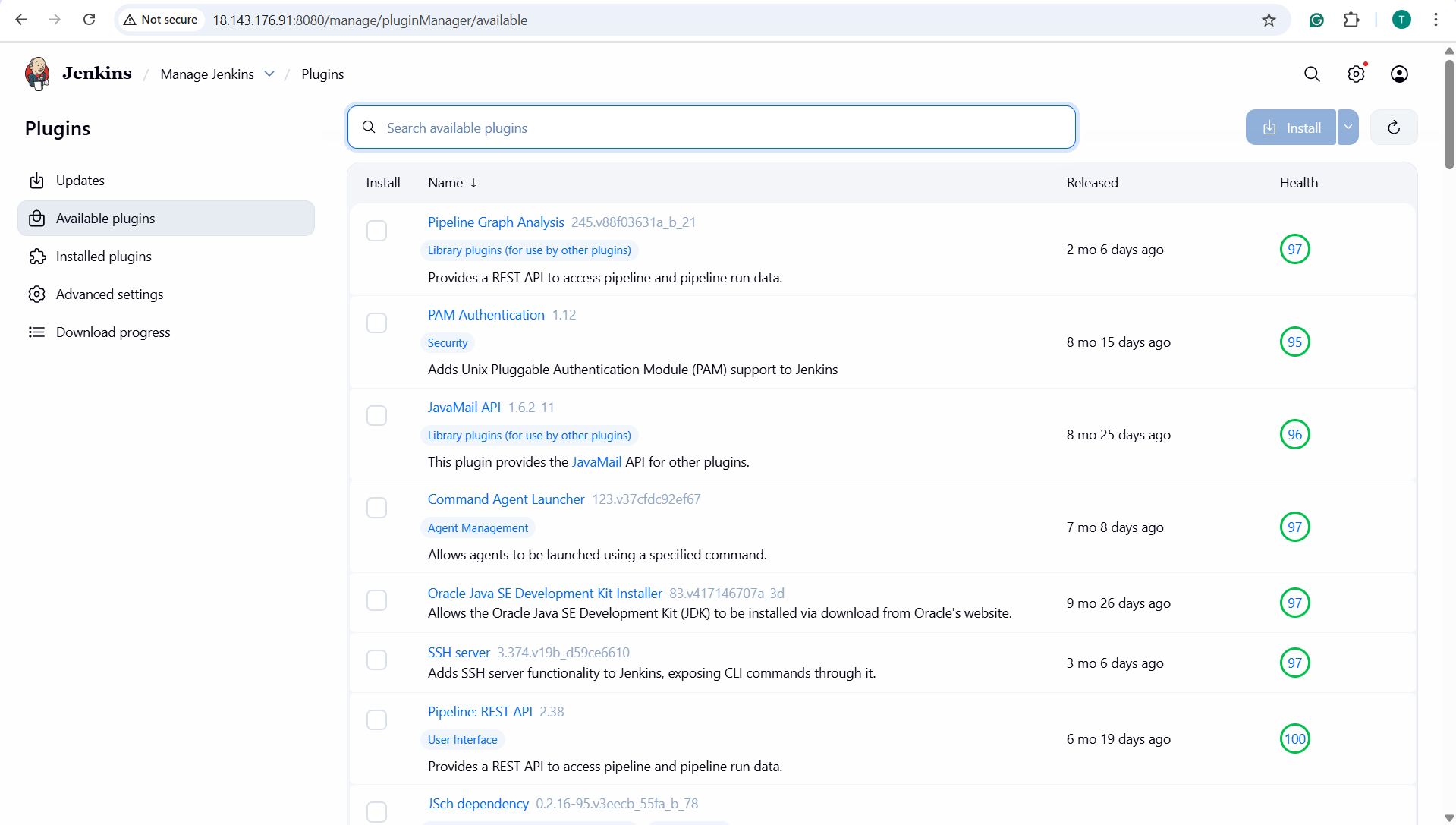
1. Set up Jenkins for CI/CD:

### - Install required Jenkins plugins:

From ****Manage Jenkins** => **Plugins => Available**:**

* + Kubernetes CLI
  + Docker Pipeline
  + Git Plugin
  + Amazon ECR
  + Pipeline: AWS Steps
  + NodeJS Plugin



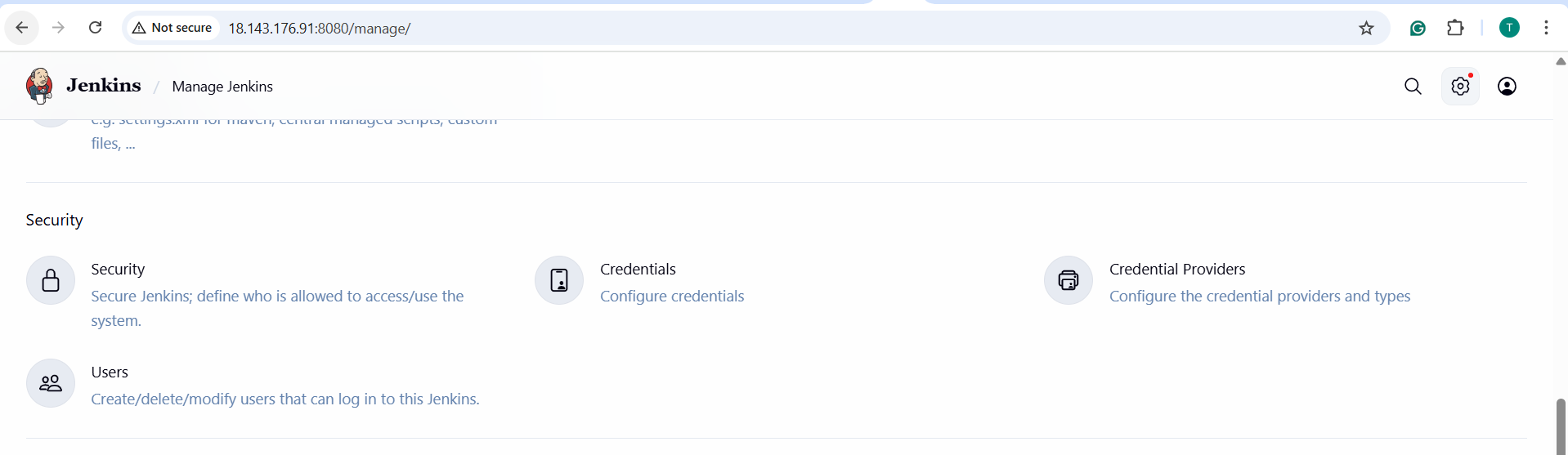


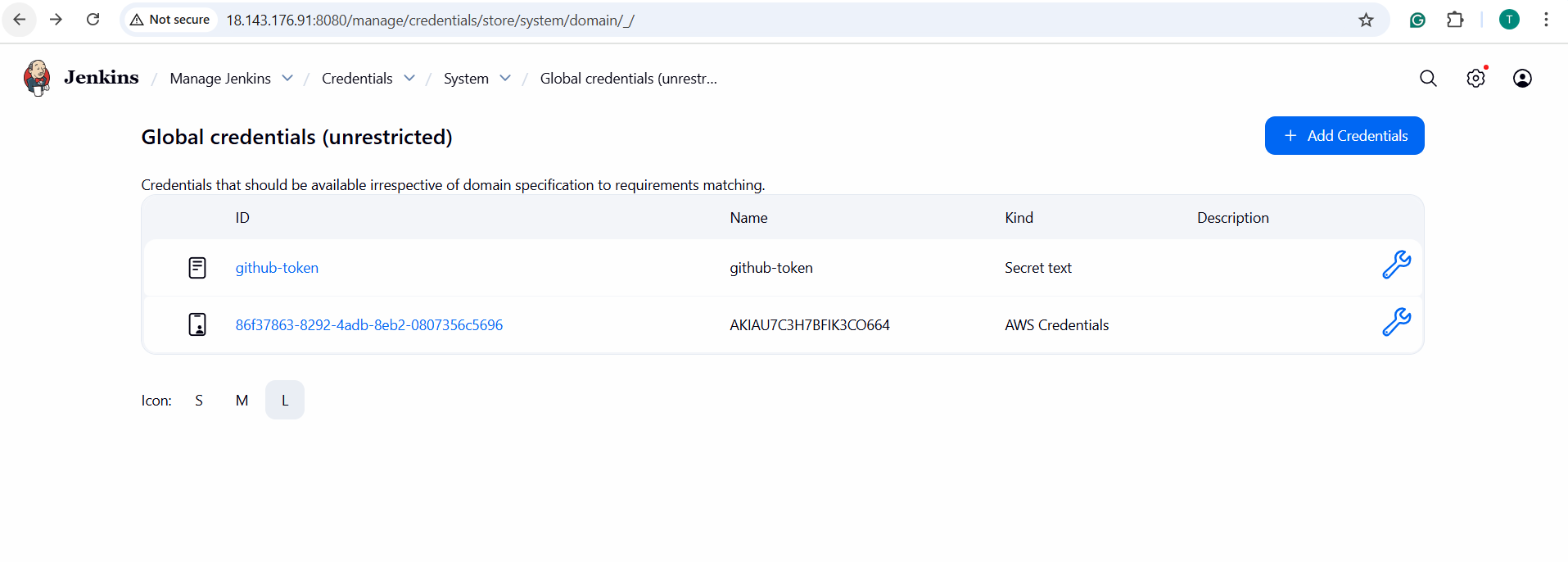
### - Configure Jenkins credentials:

Go to **Manage Jenkins => Credentials => click (global) => Add Credentials:**

Add:

* + **GitHub token** (for code checkout)
  + **AWS credentials** (Access Key ID + Secret Key)
  + **DockerHub credentials** (if pushing images)





### - Connect Jenkins to your GitHub repo

Create a new Jenkins pipeline job:

* + Name: demo-eks-pipeline
  + Type: Pipeline
  + Choose “Pipeline script from SCM”
  + SCM: Git
  + Repository URL: https://github.com/TuTruonng/sd4719\_msa/blob/master/kubernetes
* Branch: main or master

1. f
2. f
3. f
4. f