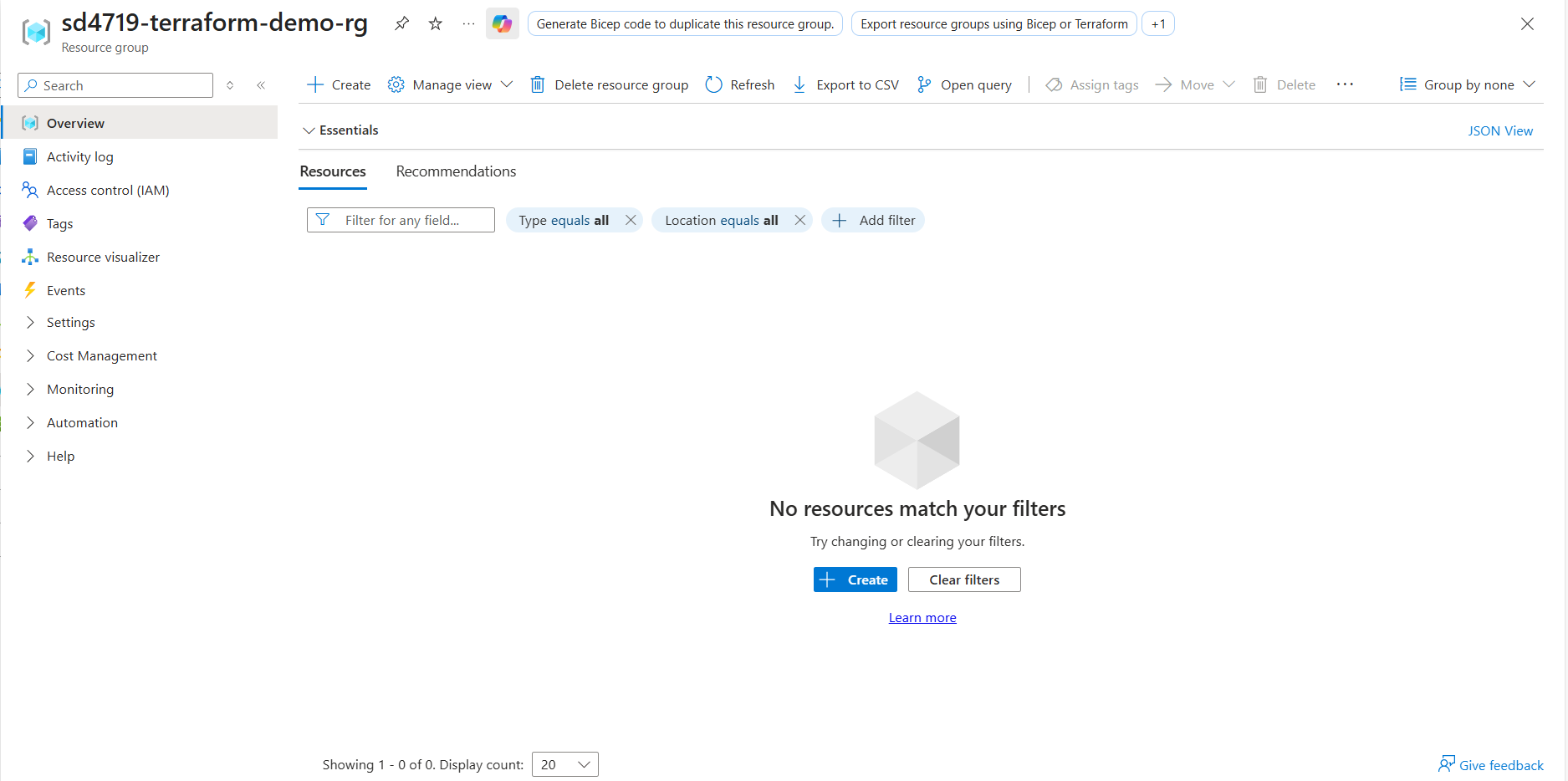
### ****APPLICATION DEPLOY TO AKS****

### ****Step 1: Sign in & Create Prerequisites****

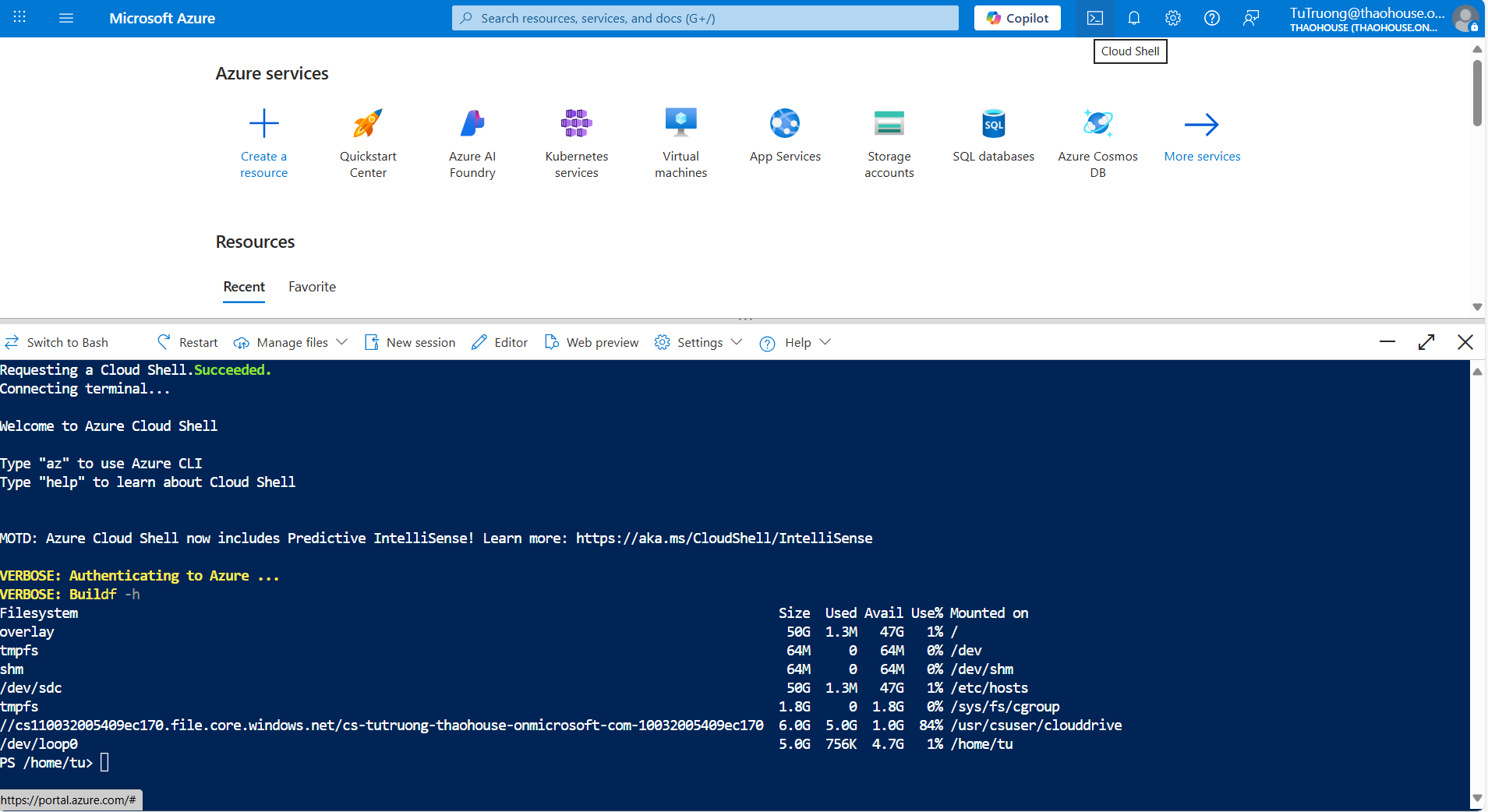
1. Go to https://portal.azure.com.
2. Create a **Resource Group**:
   * Search → “Resource groups” → **+ Create**
   * Example name: terraform-demo-rg
   * Choose a **region** (e.g., Southeast Asia)



### ****Step 2: Create a Service Principal for Terraform****

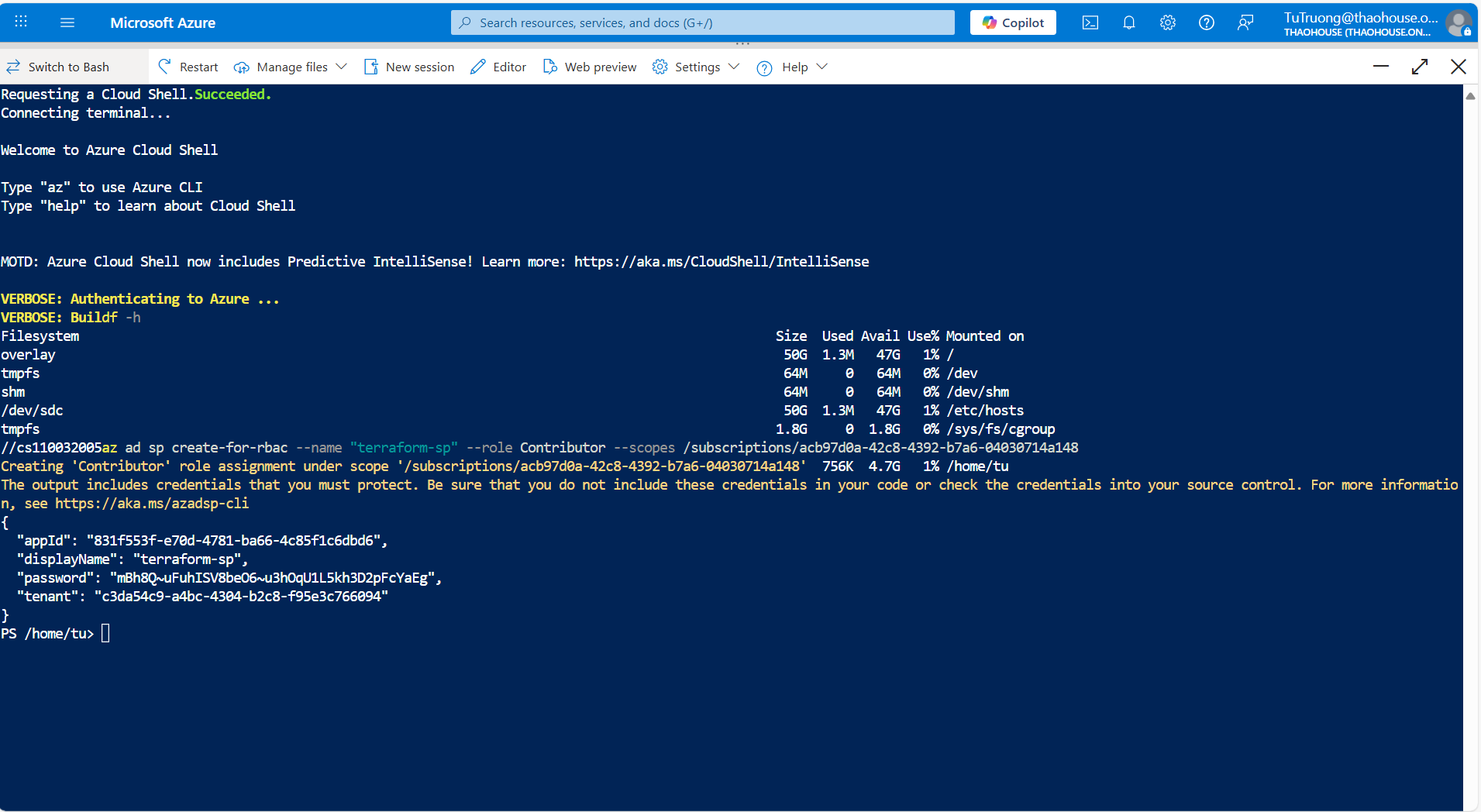
Terraform needs Azure credentials to deploy resources.

1. Open **Cloud Shell** in Azure Portal (top bar, “>\_” icon).



1. Run: az ad sp create-for-rbac --name "terraform-sp" --role Contributor --scopes /subscriptions/acb97d0a-42c8-4392-b7a6-04030714a148

Get an output like:



### ****Step 3: Set Terraform Environment Variables****

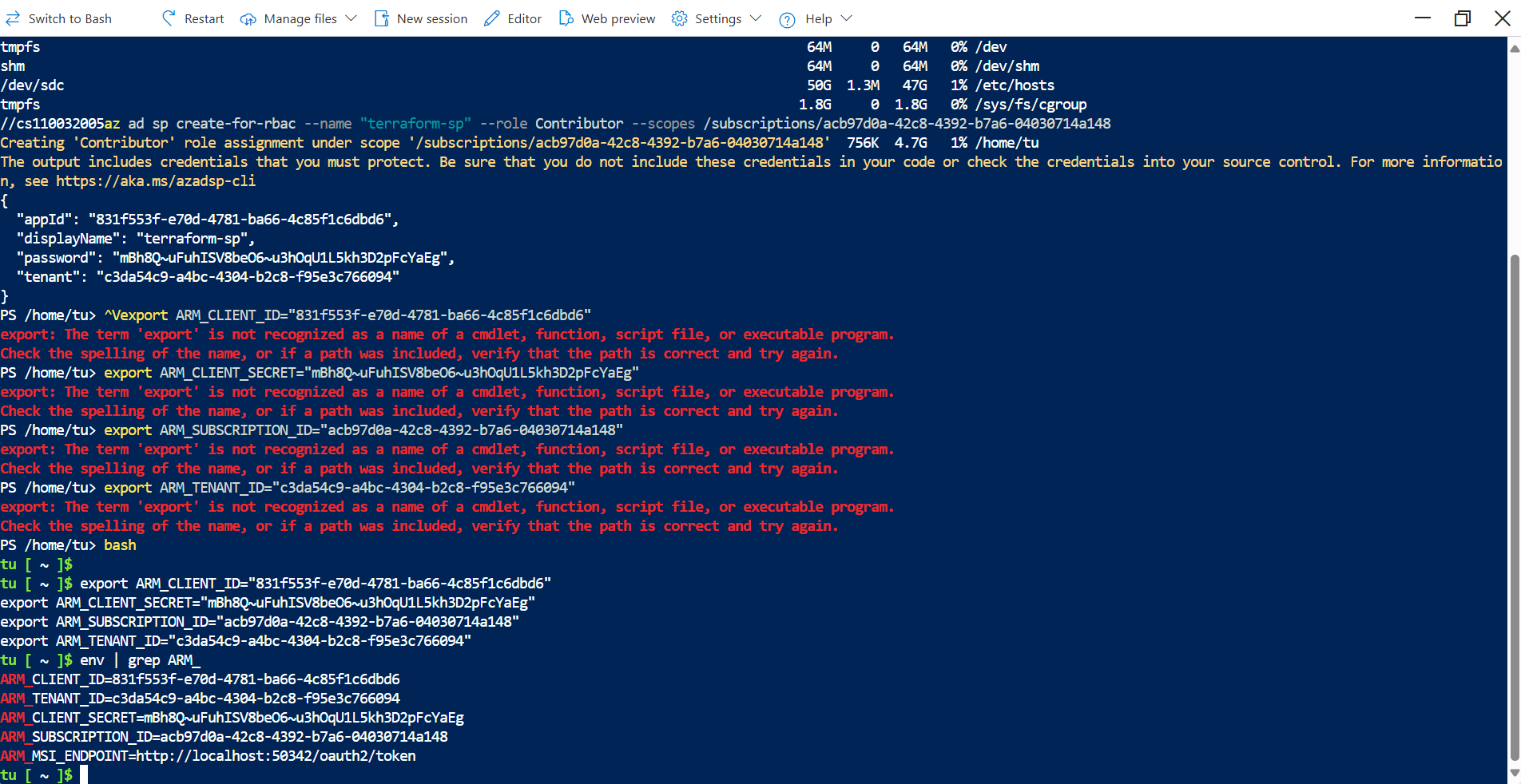
In the same Cloud Shell, switch to Bash and set these:

export ARM\_CLIENT\_ID="831f553f-e70d-4781-ba66-4c85f1c6dbd6" (**appId**)

export ARM\_CLIENT\_SECRET="mBh8Q~uFuhISV8beO6~u3hOqU1L5kh3D2pFcYaEg" (**password**)

export ARM\_SUBSCRIPTION\_ID="acb97d0a-42c8-4392-b7a6-04030714a148"

export ARM\_TENANT\_ID="c3da54c9-a4bc-4304-b2c8-f95e3c766094" (**tenant**)

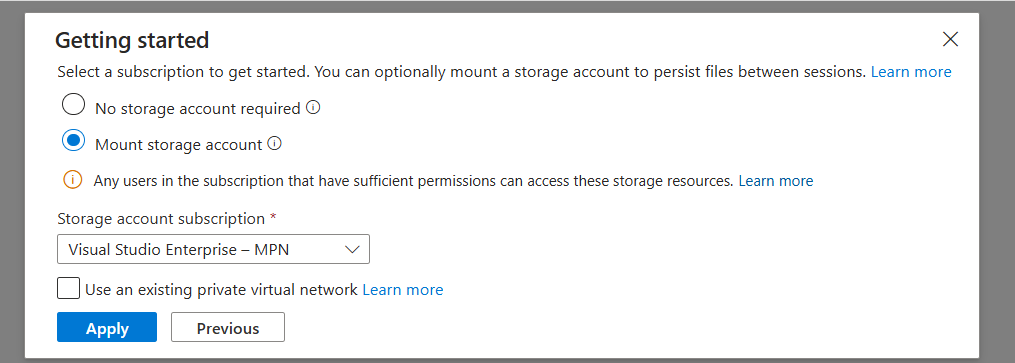


### ****Step 4: Create Terraform Configuration****

### In **Azure Cloud Shell or **Local Machine,** create a folder** (e.g., terraform-aks-demo)

**On Azure Cloud Shell:**

1. In the previous step (**Step 2**), we have created a storage container (e.g., clouddrive) to persist files

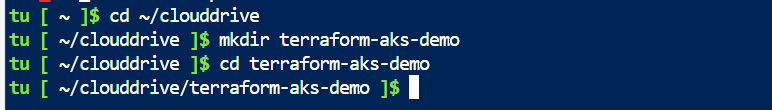


**2. Now we will access the path of** clouddrive **and create a Terraform working folder with commands:**

cd ~/clouddrive

mkdir terraform-aks-demo

cd terraform-aks-demo



**3. Create Terraform files inside with command:**

touch main.tf providers.tf outputs.tf

4. Then we can open and edit files using **VS Code editor inside Cloud Shell with command**:

code main.tf

**On **Local Machine (**I prefer to apply Terraform Configuration on this because it is easier editing, full Linux environment):**

**1. Create the Terraform files in “terraform-aks-demo**” folder **on local and access it through the VSCode with **WSL terminal** (**WSL: Ubuntu-24.04**):**

**- **If Terraform is not installed, install it with:****

sudo apt update && sudo apt install -y wget unzip

wget -O terraform.zip https://releases.hashicorp.com/terraform/1.9.8/terraform\_1.9.8\_linux\_amd64.zip

sudo unzip terraform.zip -d /usr/local/bin/

terraform -version

- **If az (Azure CLI) isn’t installed:**

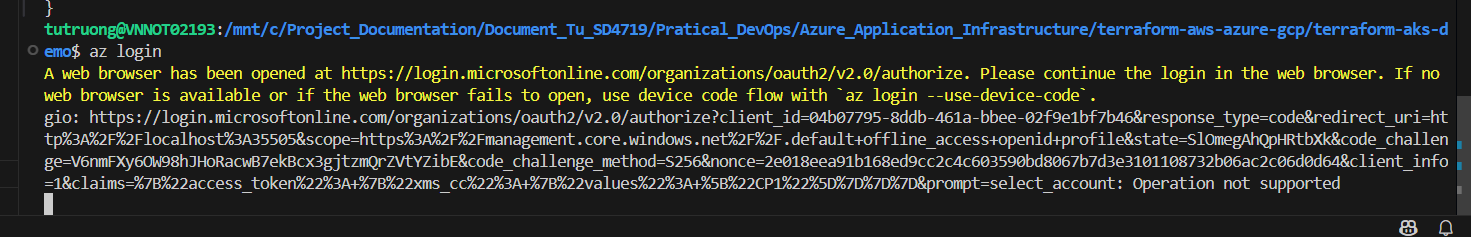
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

2.Inside that folder, create:

* main.tf
* providers.tf
* outputs.tf
* variables.tf

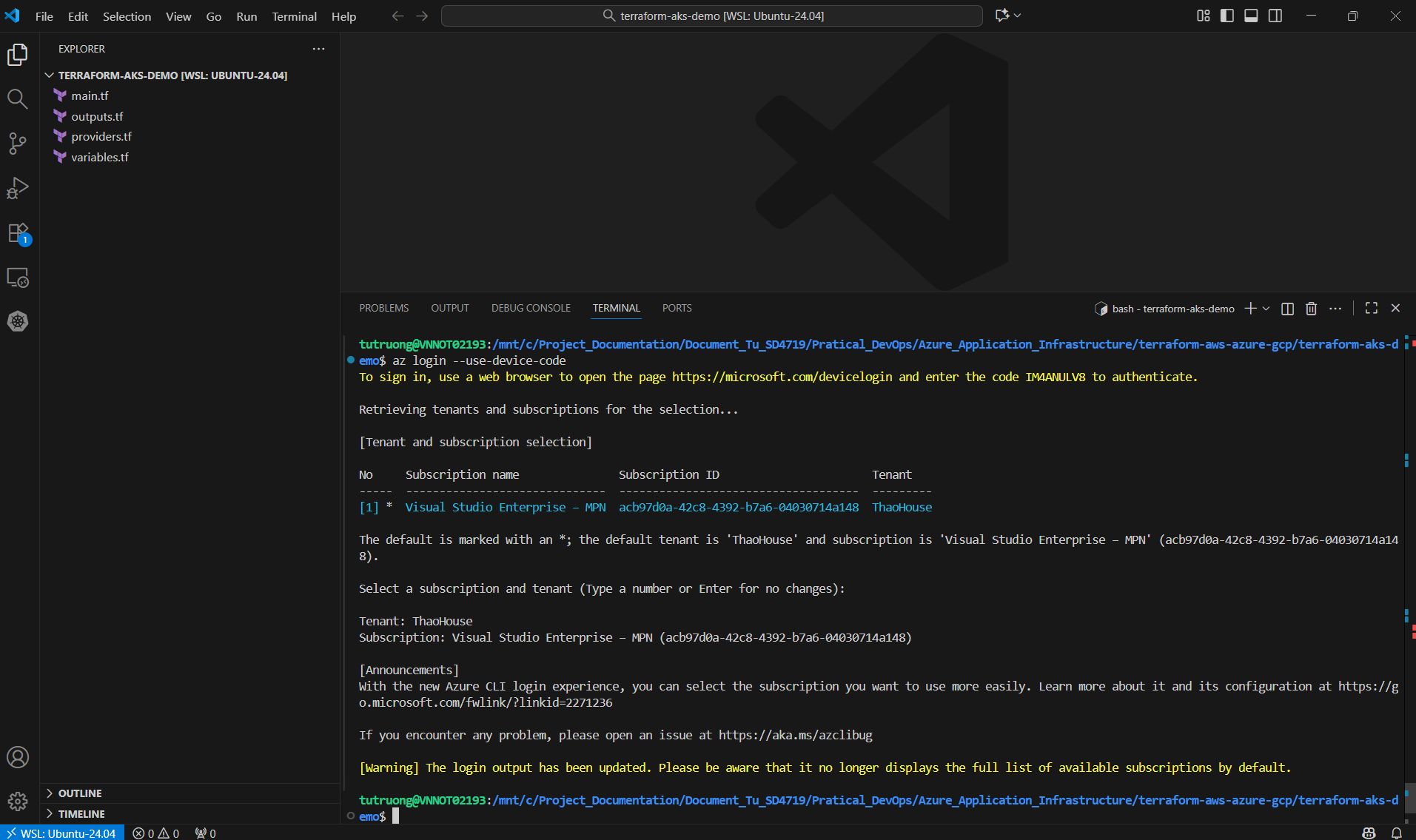
**3. Authenticate to Azure with:**

az login



**If no web browser is available or if the web browser fails to open, use device code flow with `az login –use-device-code` and use a web browser to open the page https://microsoft.com/devicelogin and enter the code IM4ANULV8 to authenticate:**

**az login –use-device-code**



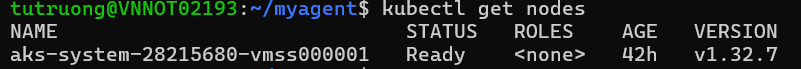
****Get cluster credentials (this updates kubeconfig), run with:****

**az aks get-credentials --resource-group** **sd4719-terraform-demo-rg --name** **demo-aks-cluster --attach-acr mydemoacrregistry**



### ****Test connection, check deployments and services, run with:****

kubectl get nodes



**kubectl get pods**

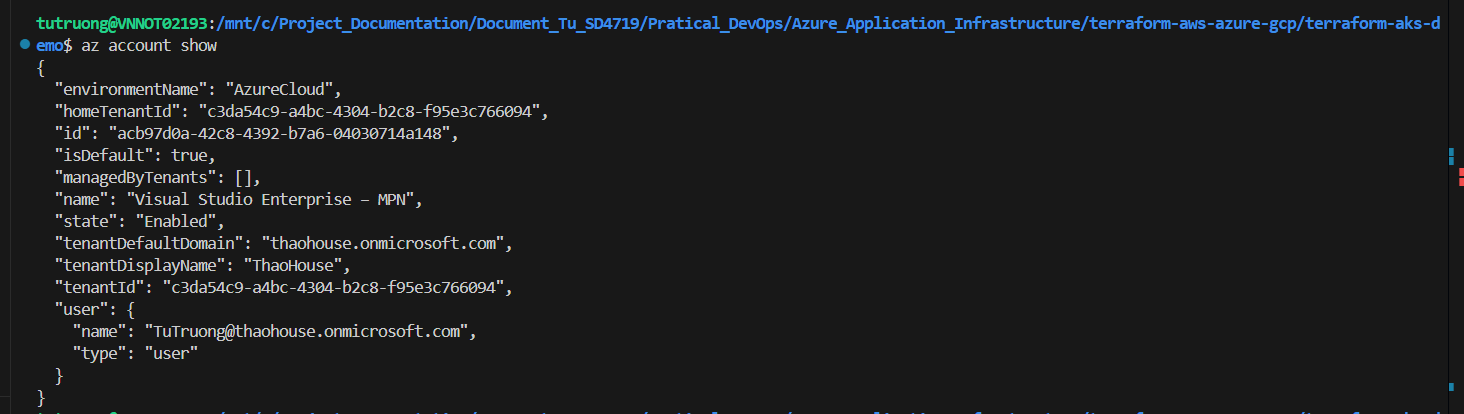
**kubectl get svc**

**kubectl get ingress**



**Verify subscription and tenant IDs (these use in Terraform variables or environment setup) with:**

az account show



4. Back in the Terraform folder and run with:

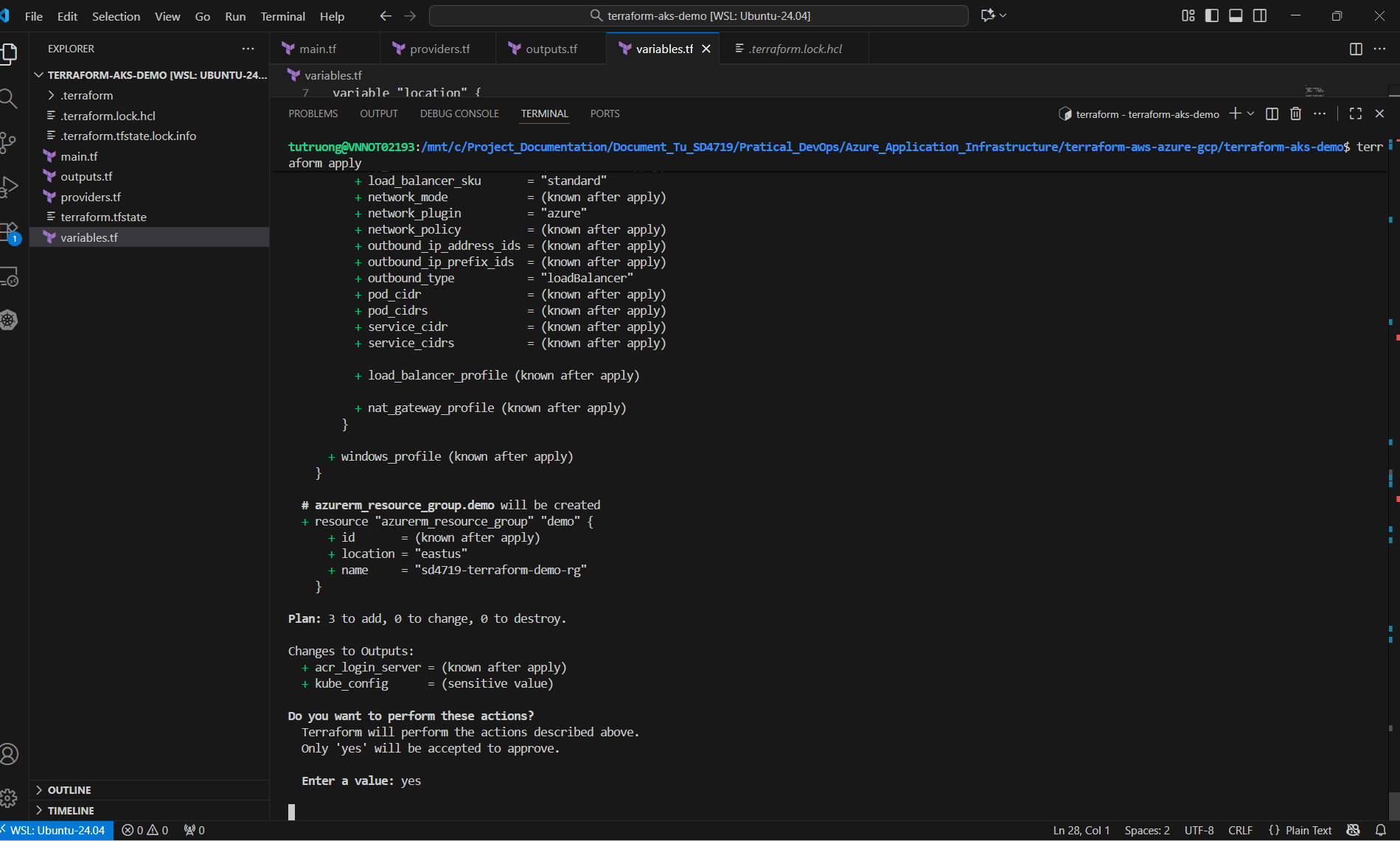
terraform init

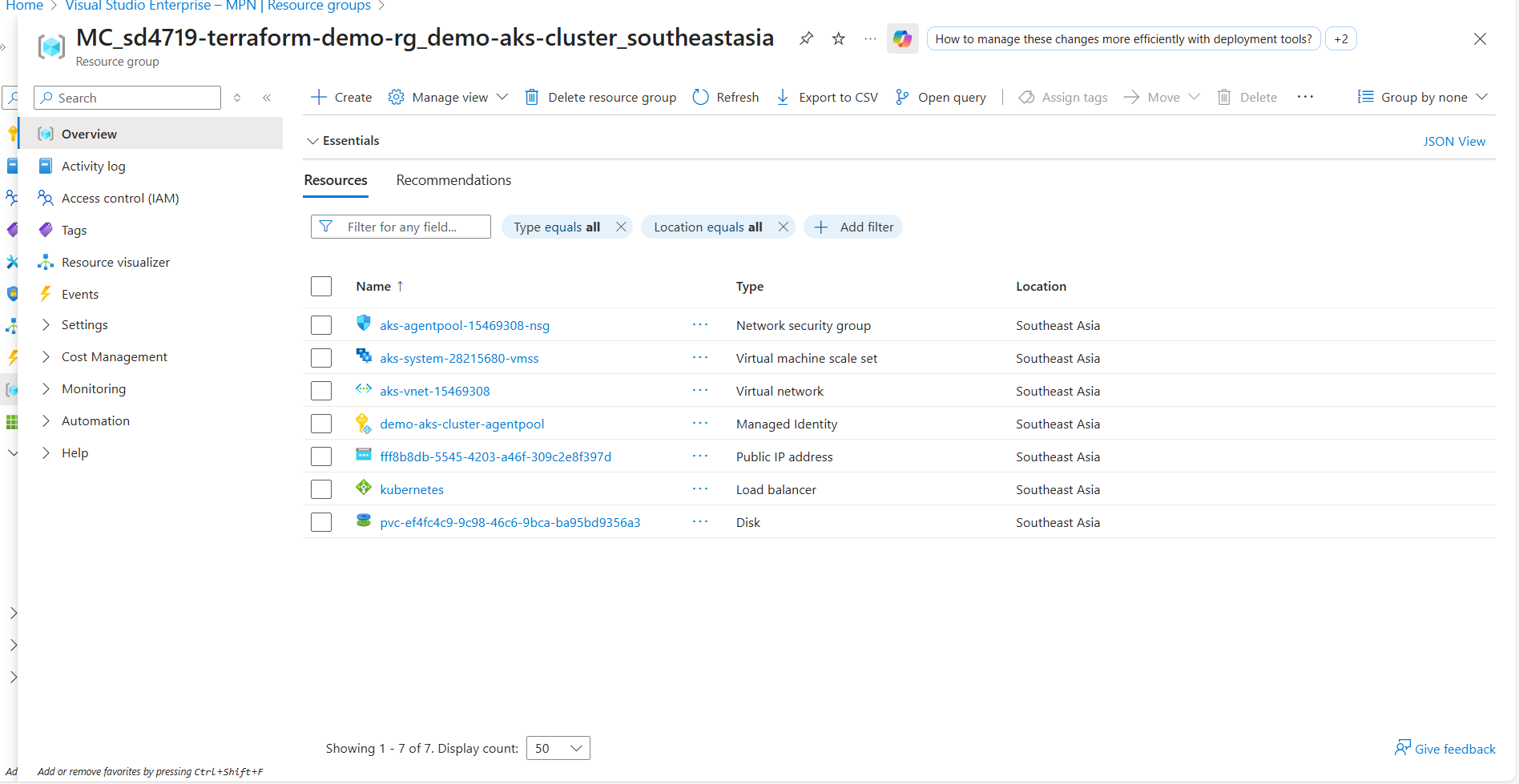
terraform plan

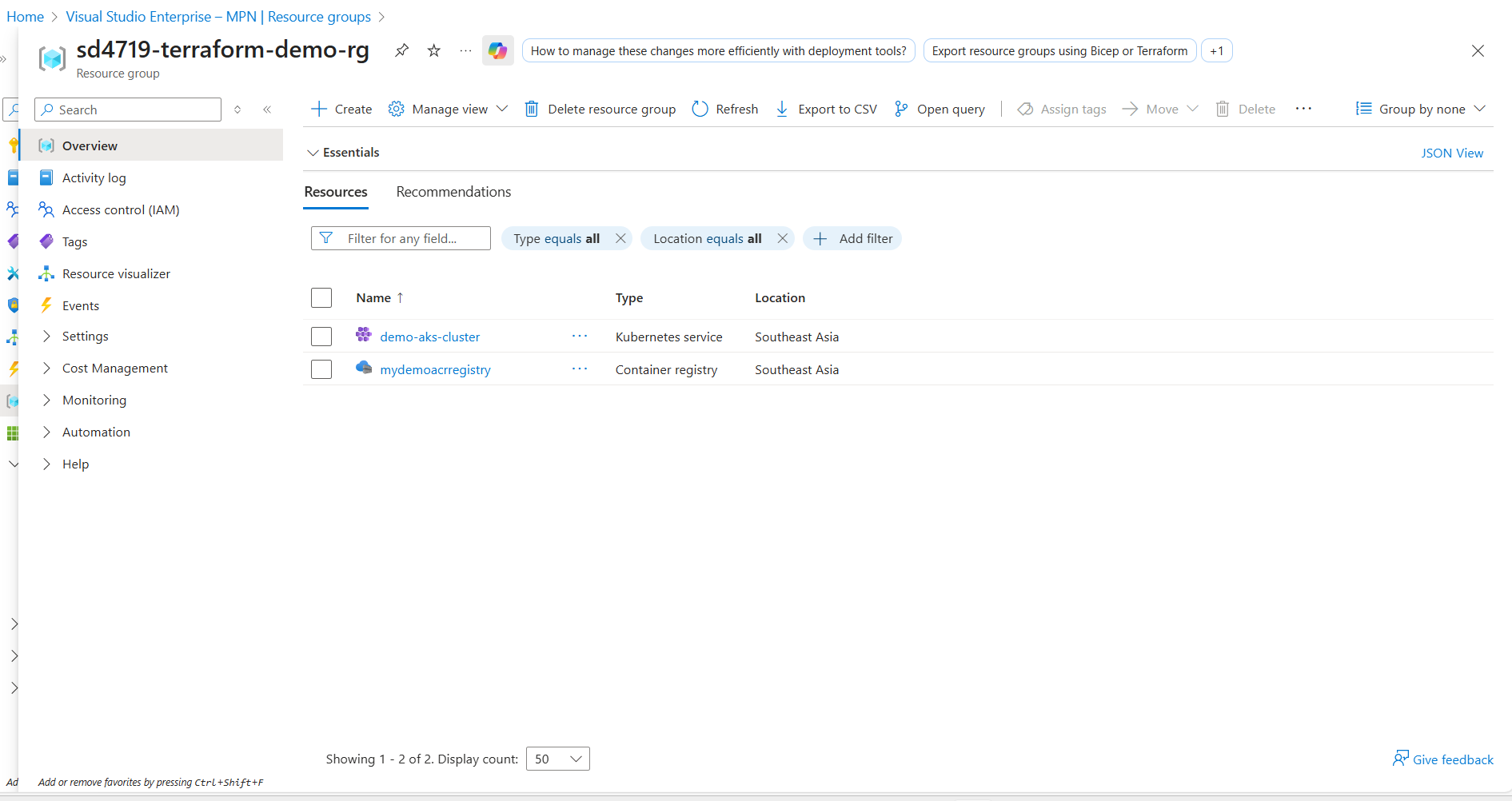
terraform apply

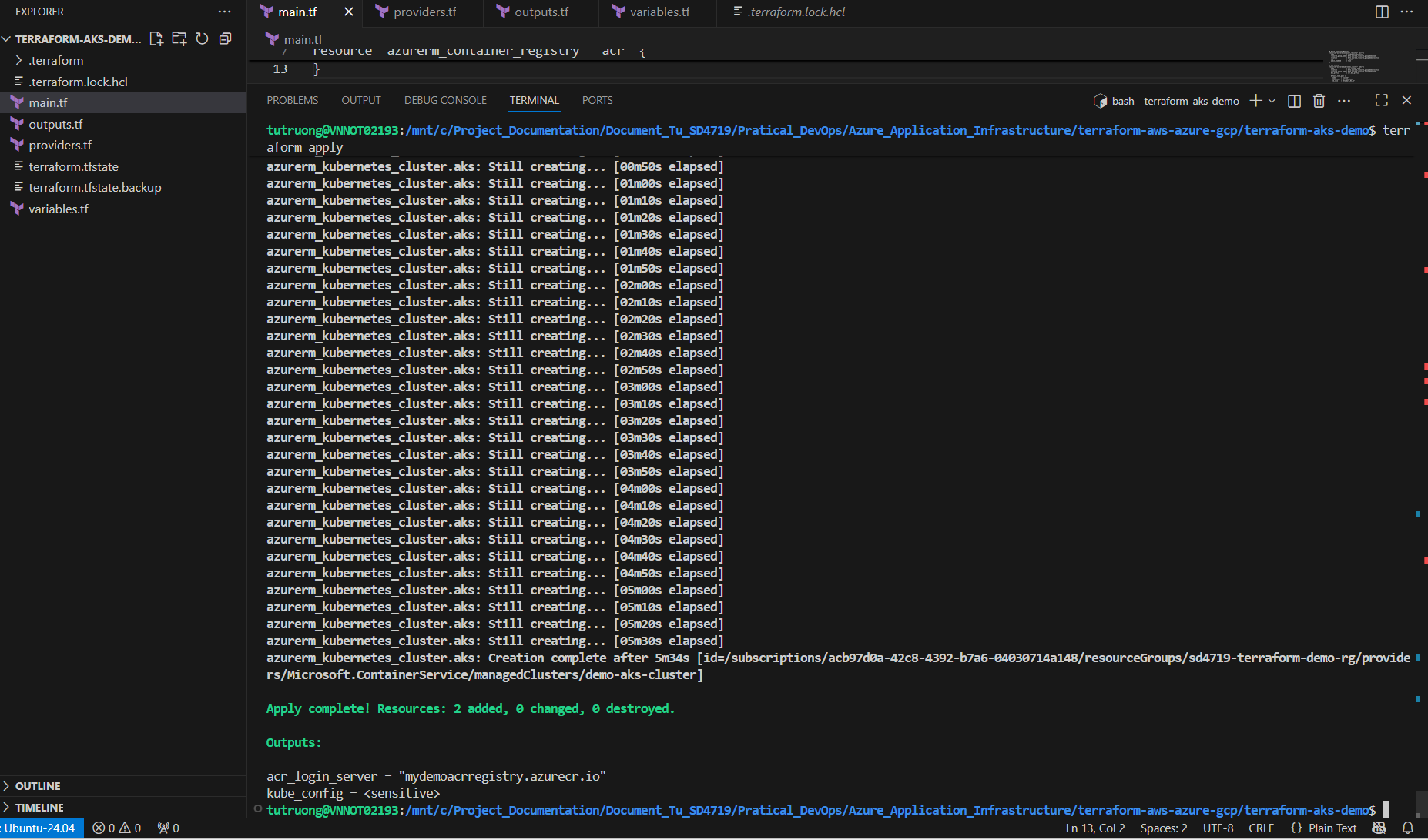
=> The result will get:

* **1 Resource Group** (If the Resource Group is already created manually on Azure Portal, edit in the main.tf file: “use **data** block instead of **resource** block“ to tell Terraform to **use it** rather than re-create it)
* **1 ACR** (for images)
* **1 AKS cluster**









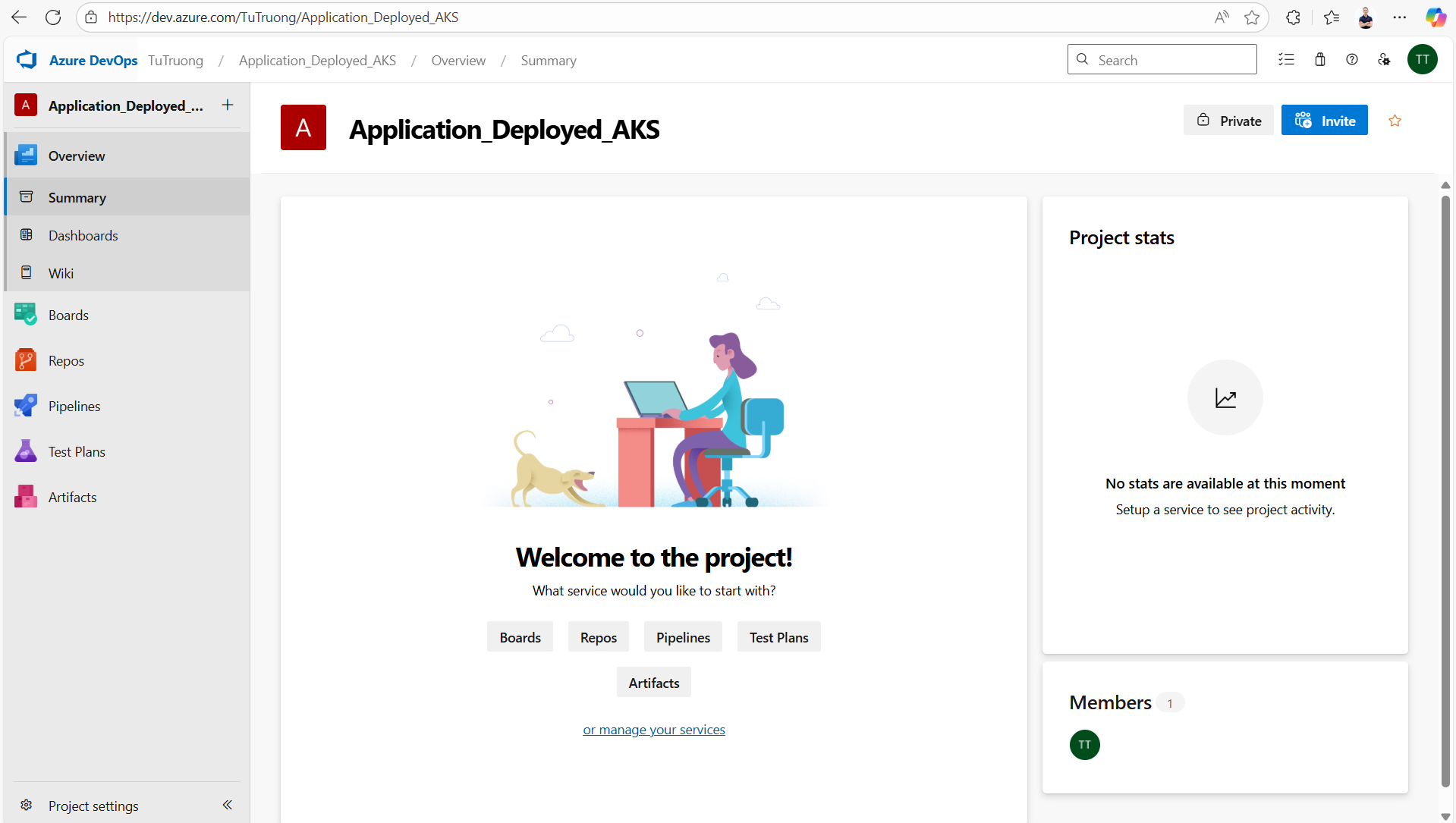
### **5. Clean Up to Avoid Costs:**

When done testing, run with:

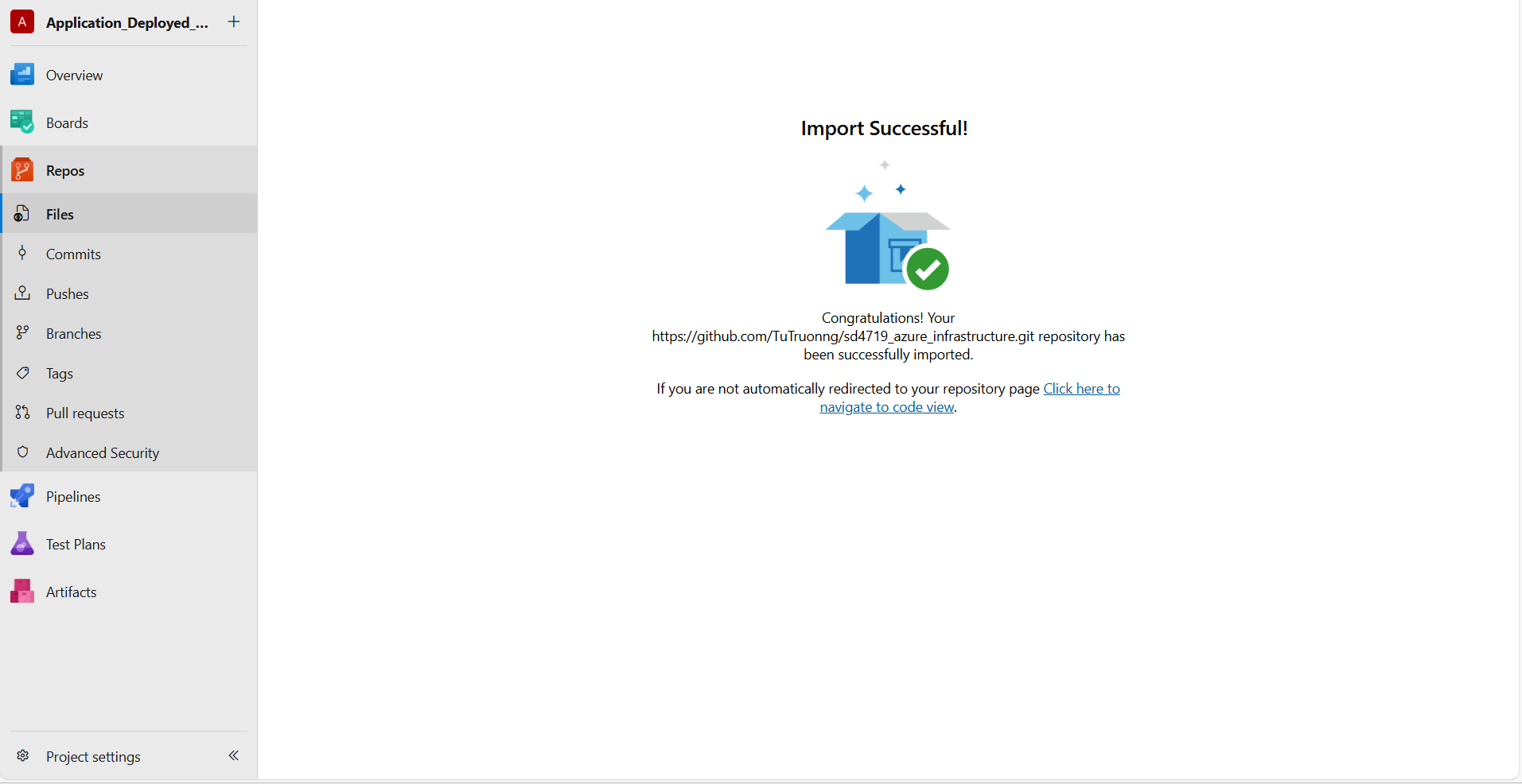
terraform destroy

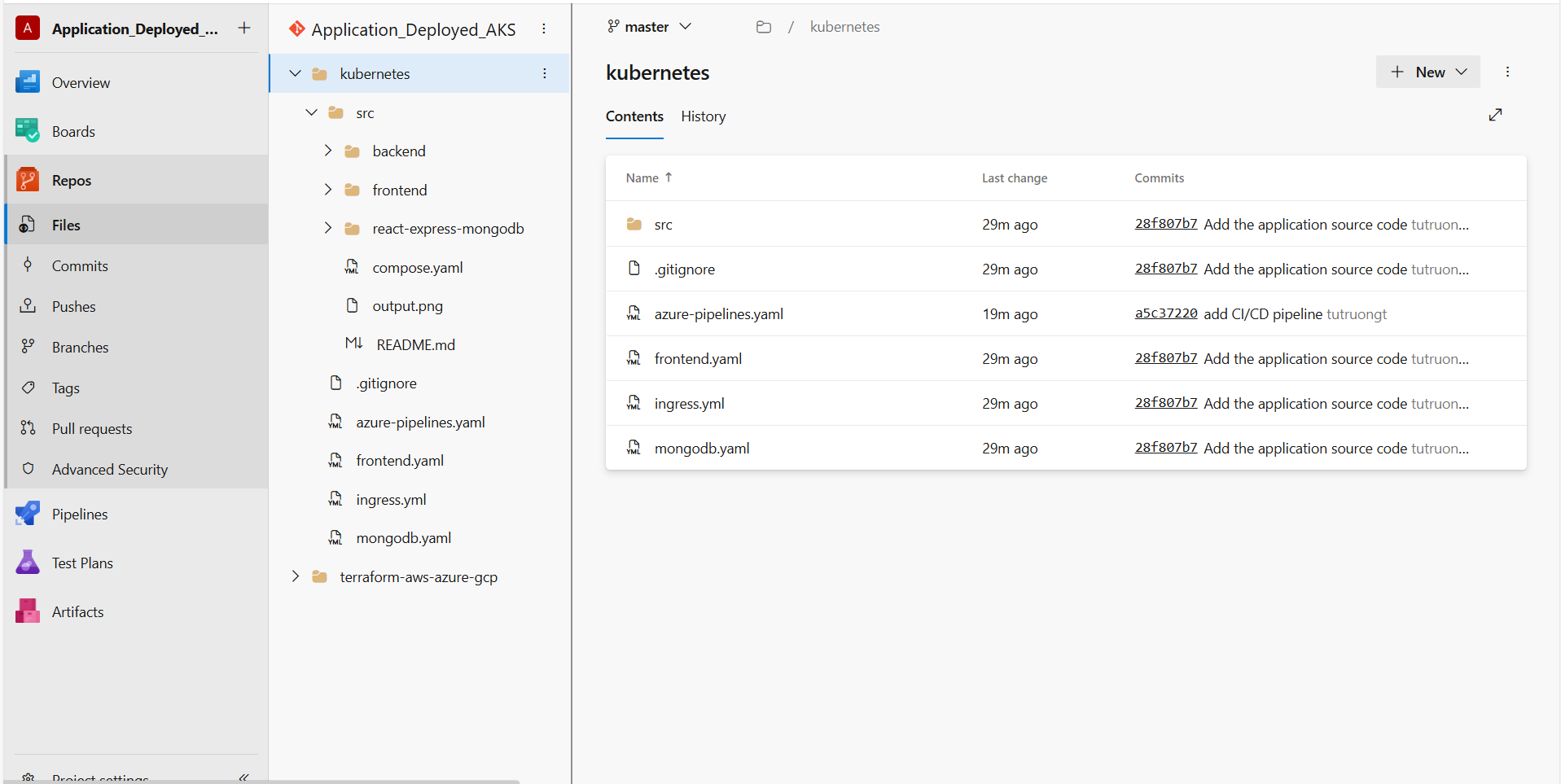
**Step 5: Setup Azure DevOps pipeline for CI/CD**

1. **Create a new Organization and a new Project in Azure DevOps:**

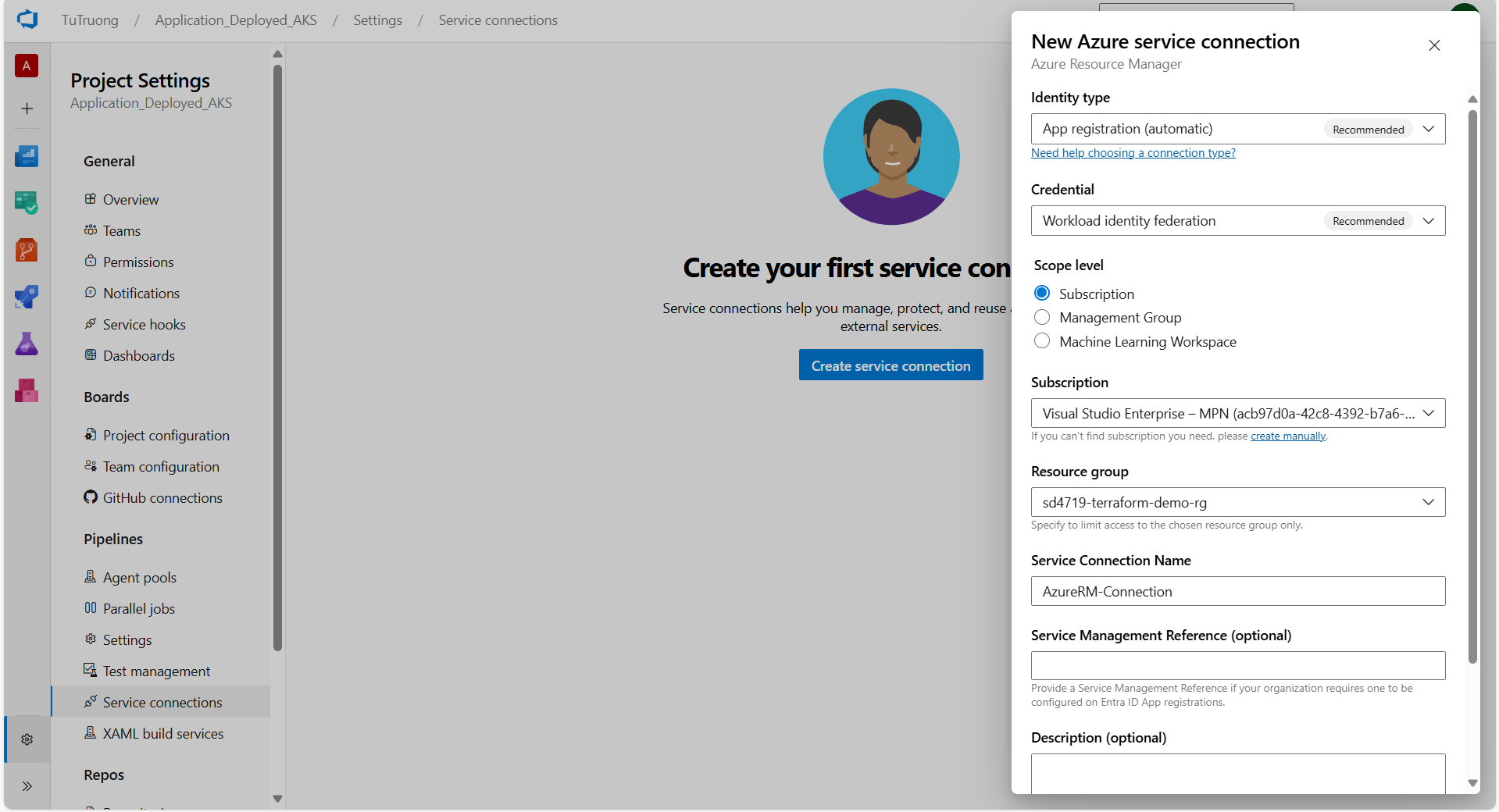
****

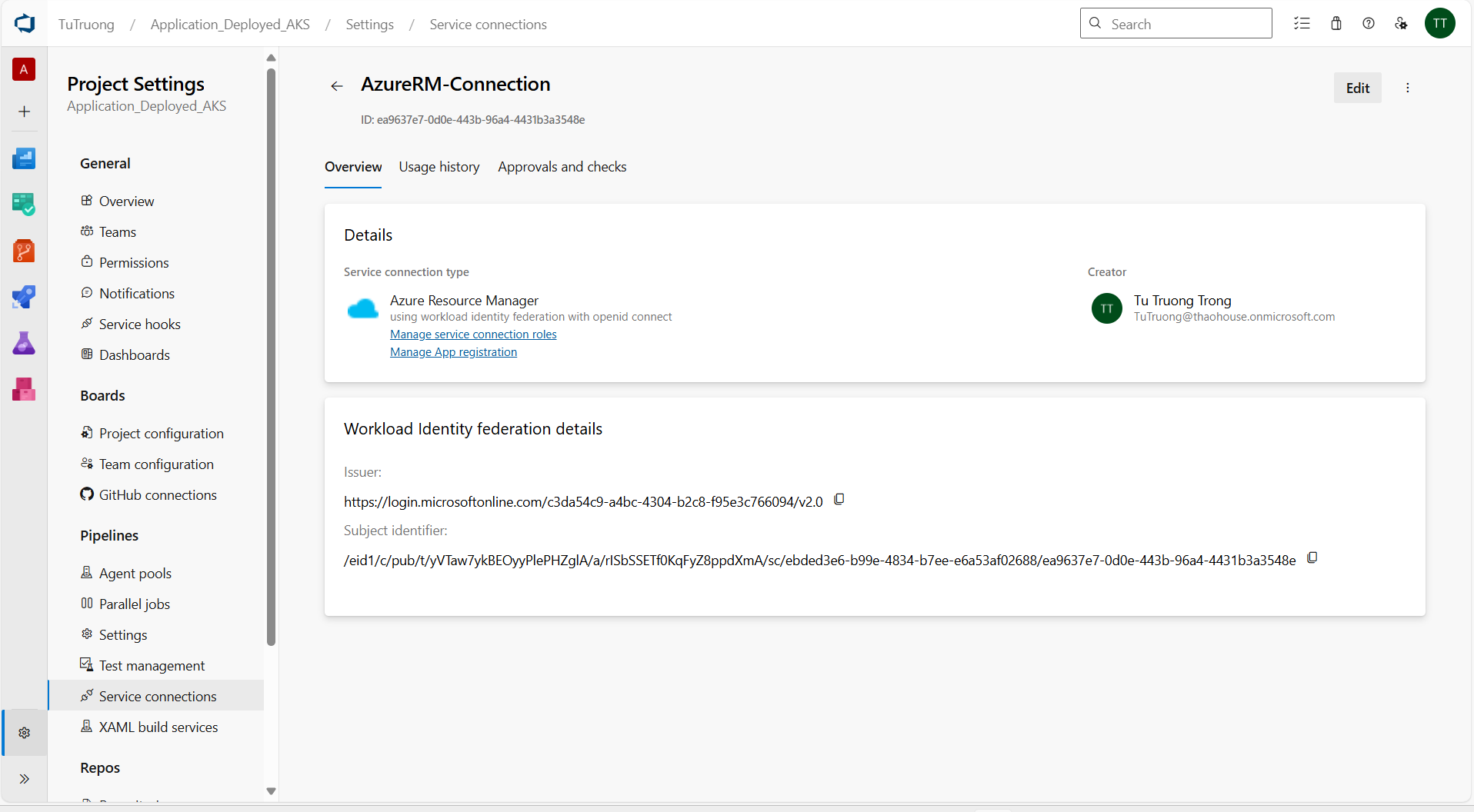
Import source code from GitHub into Repos:

****

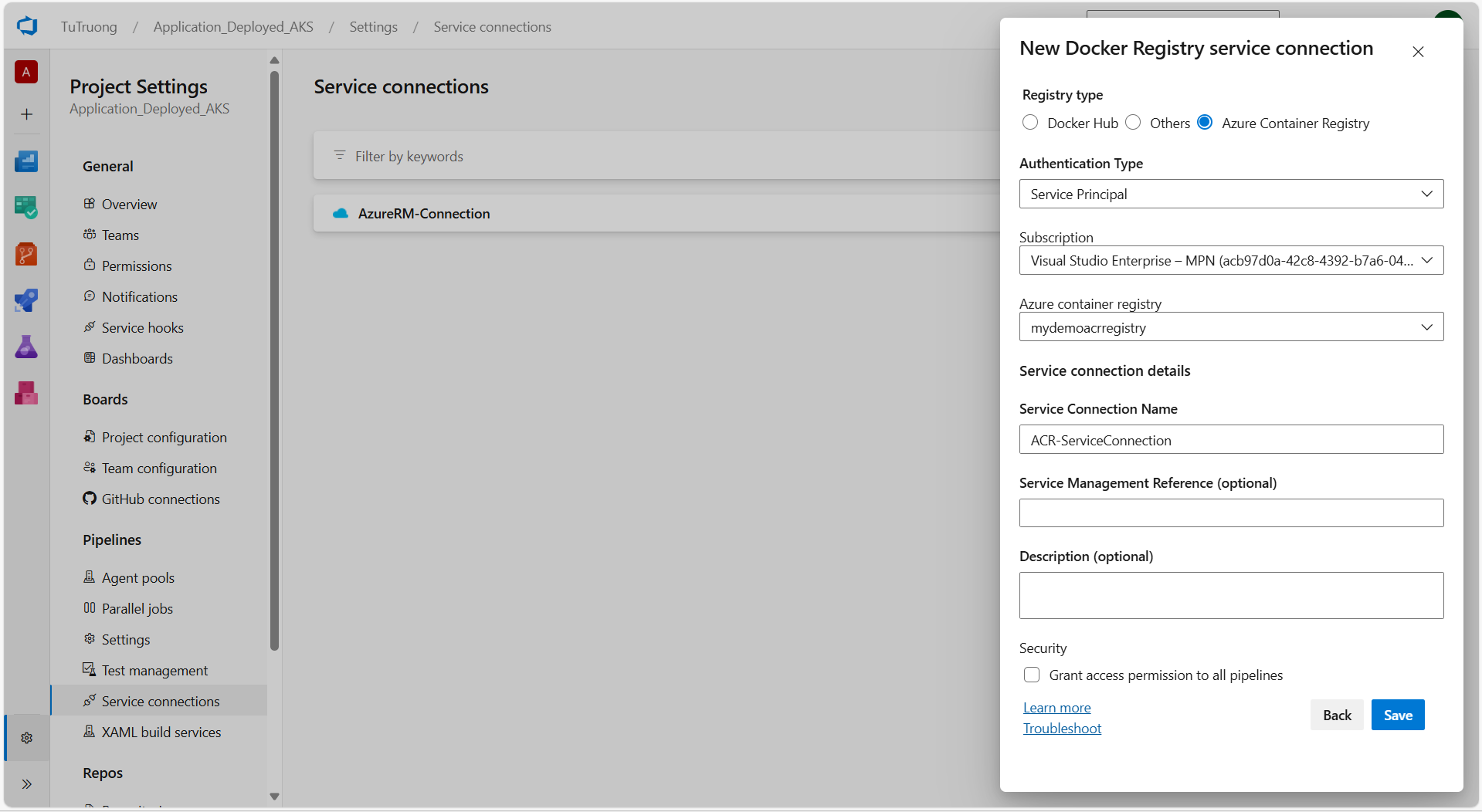
****

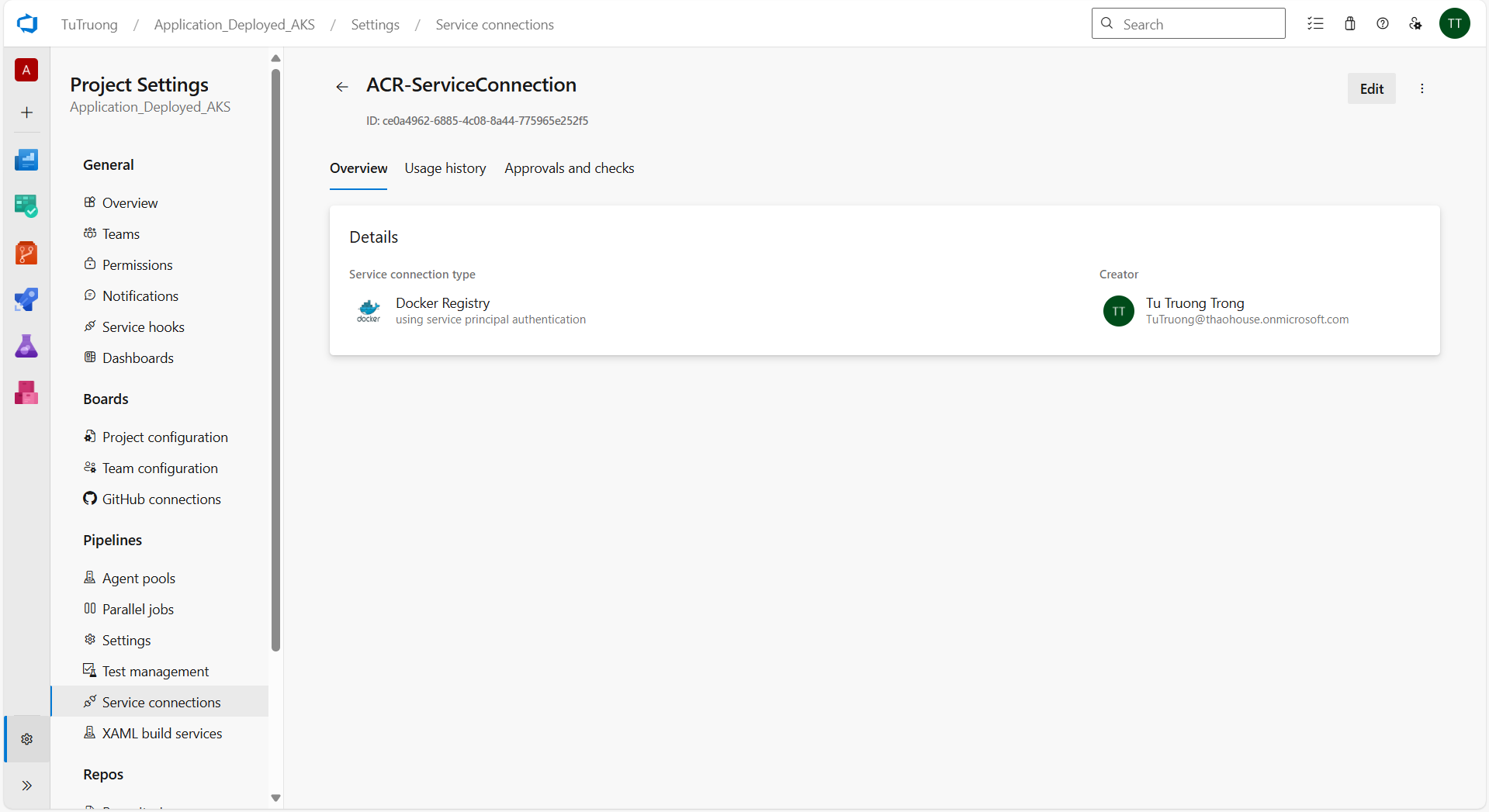
1. **Create an Azure Resource Manager (ARM) Service Connection:**
2. In **Service connections**, click **+ New service connection**.
3. Choose **Azure Resource Manager**.
4. Choose:
   * **Scope level:** Subscription
   * **Subscription name:** Azure subscription name
   * **Resource group: sd4719-terraform-demo-rg**
   * **Name this connection:** AzureRM-Connection
5. Click S**ave**.



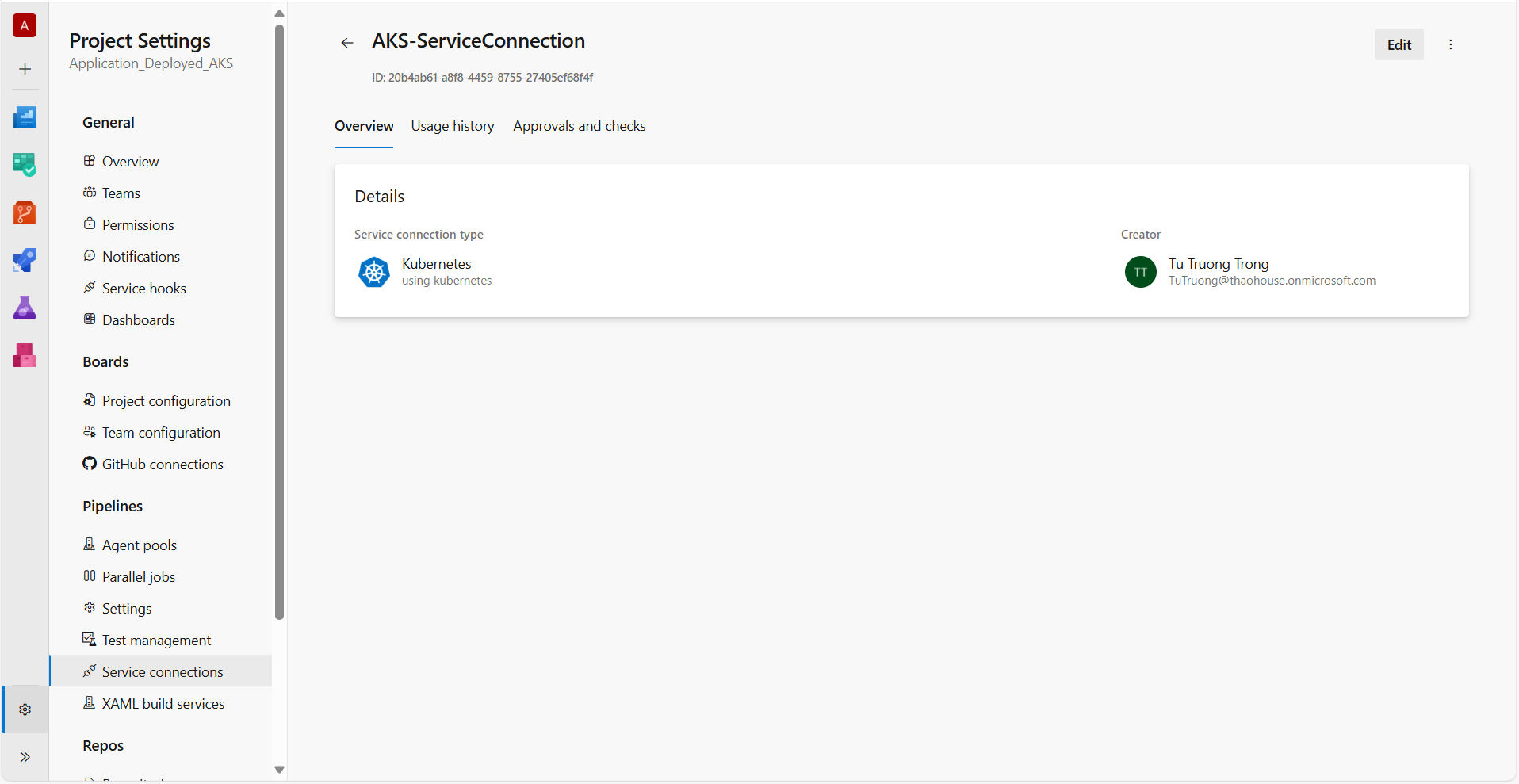


1. **Create a Docker Registry (ACR) Service Connection (to CI/CD build and push container images to ACR):**
2. In **Service connections**, click **+ New service connection** again.
3. Choose **Docker Registry**.
4. Registry type: **Azure Container Registry**.
5. Select Authentication Type: Service Principal and automatic select the **Subscription**.
6. **Name this connection: ACR-ServiceConnection**

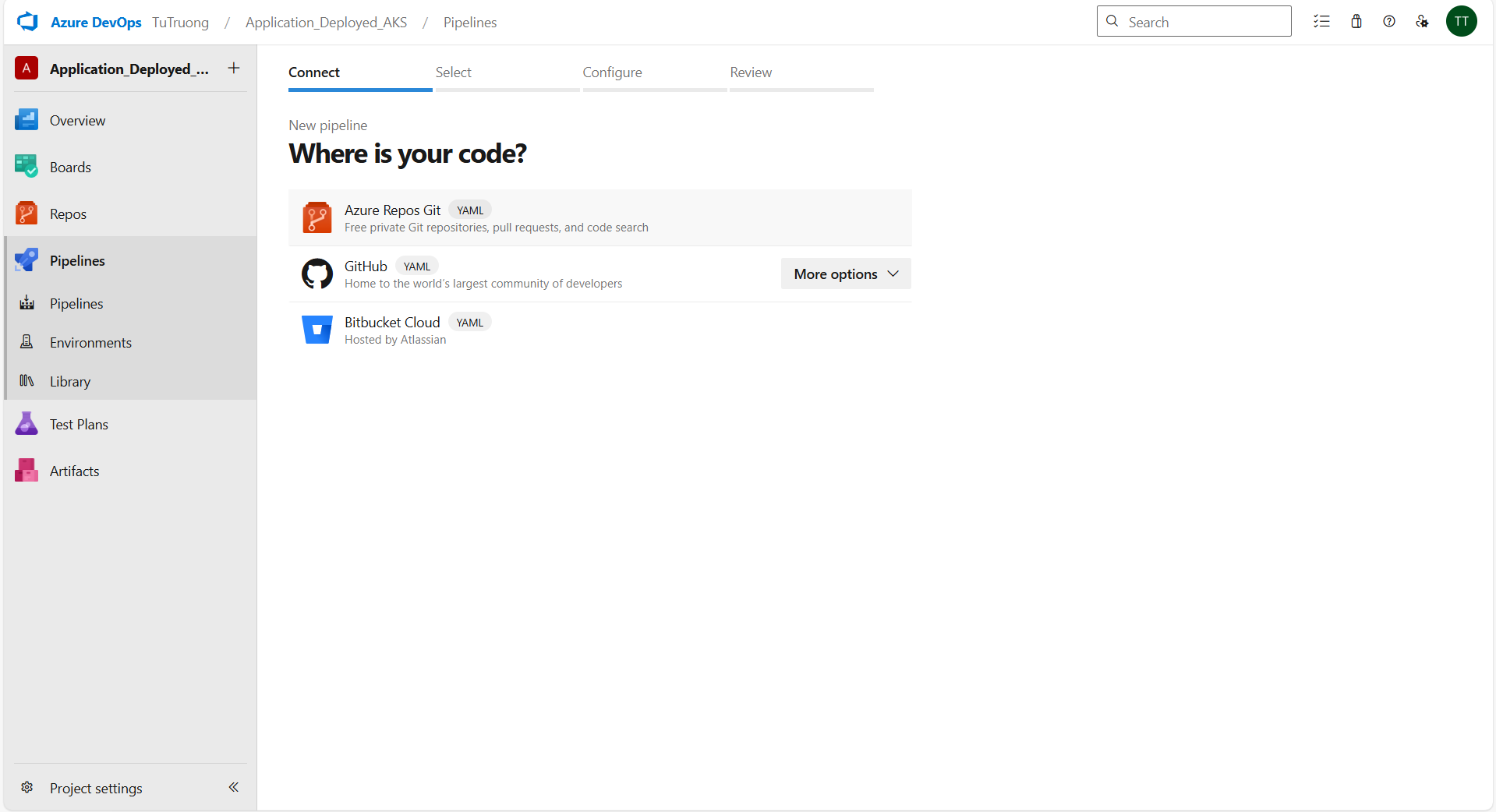




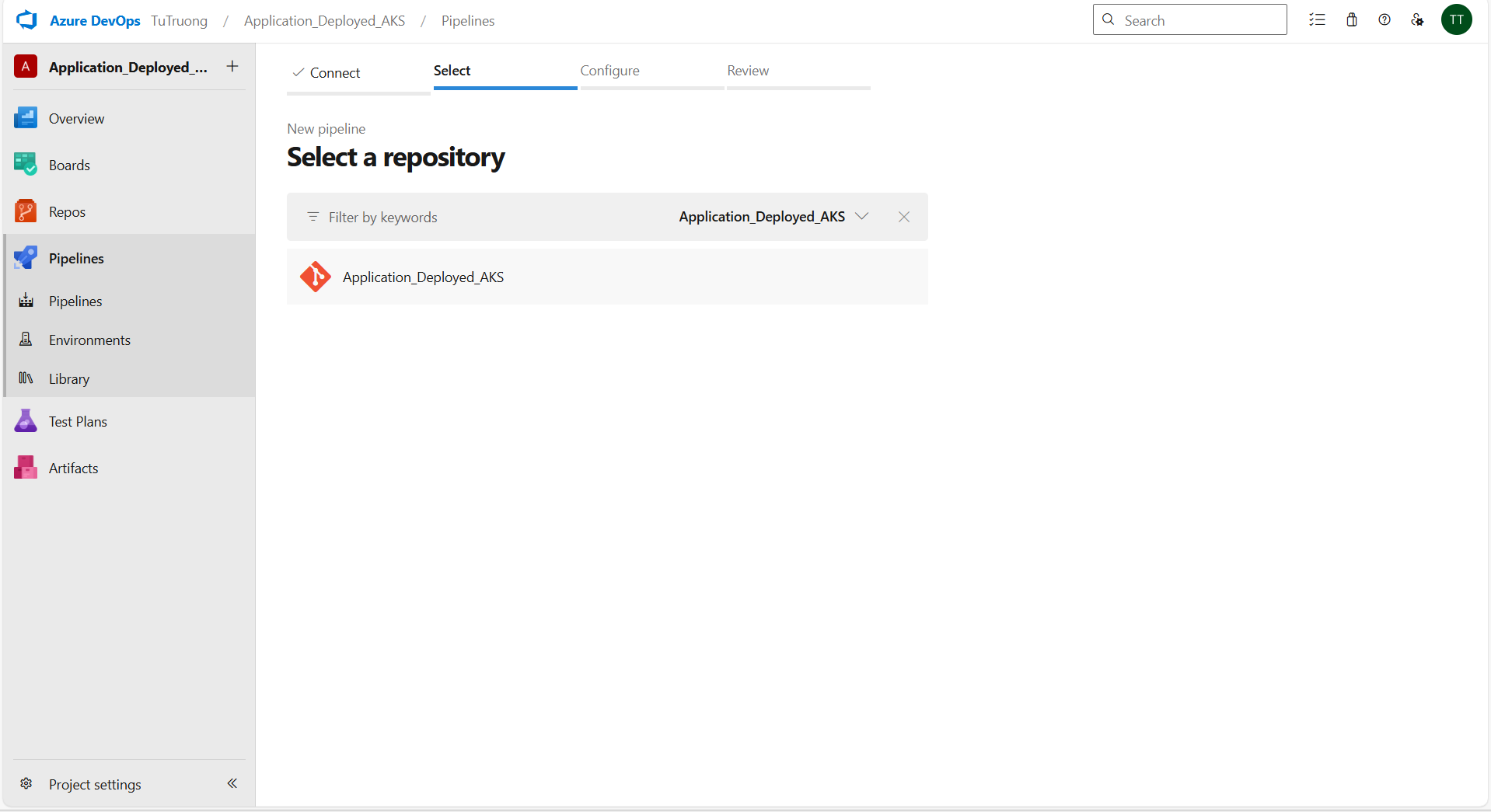
1. Create a Kubernetes Service Connection:
2. In **Service connections**, click **+ New service connection** again.
3. Choose **Kubernetes**.
4. **Name this connection: AKS-ServiceConnection**



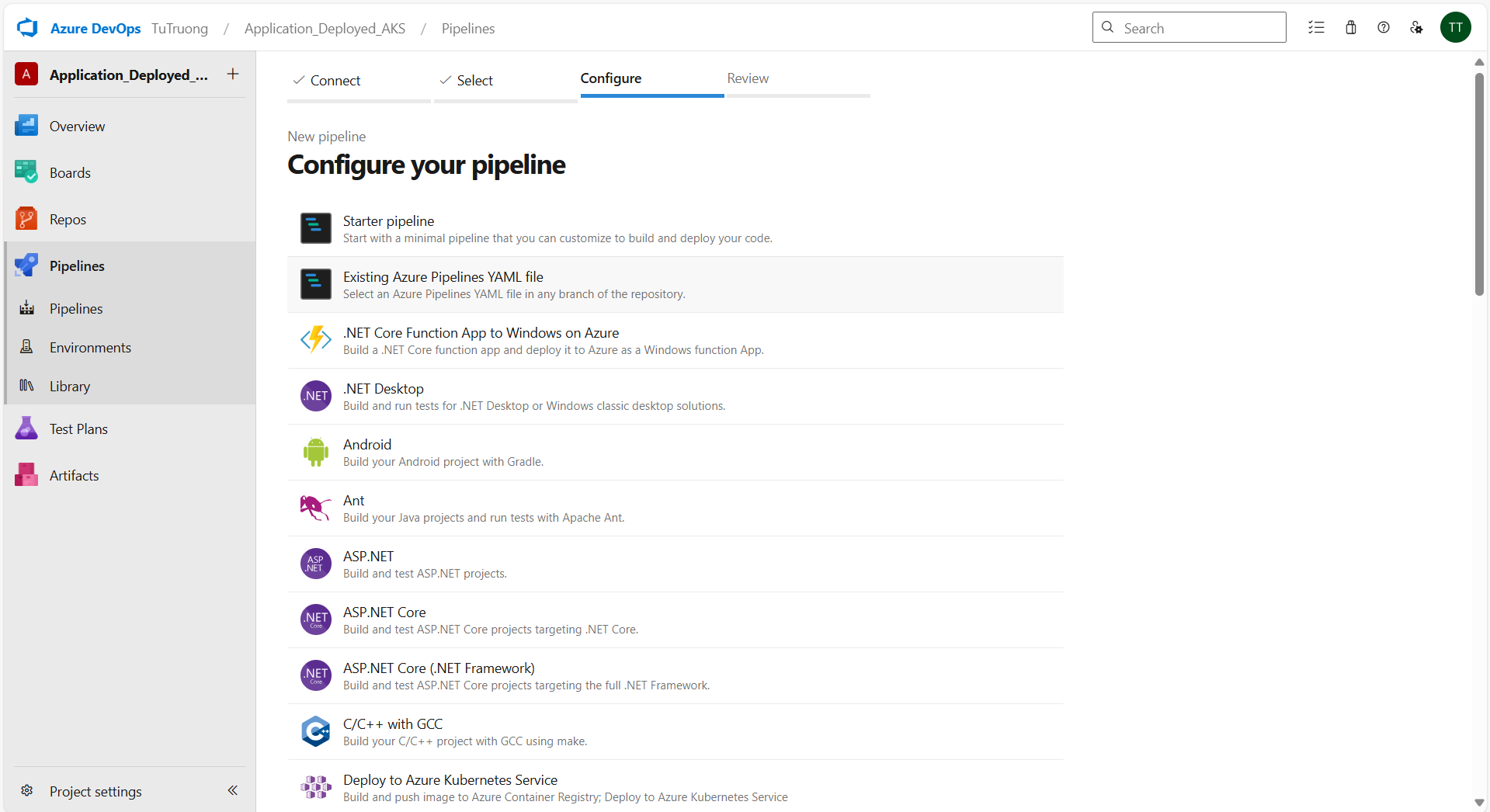
1. Create a new pipeline:
2. In the left bar, choose Pipelines and select Azure Repos Git:

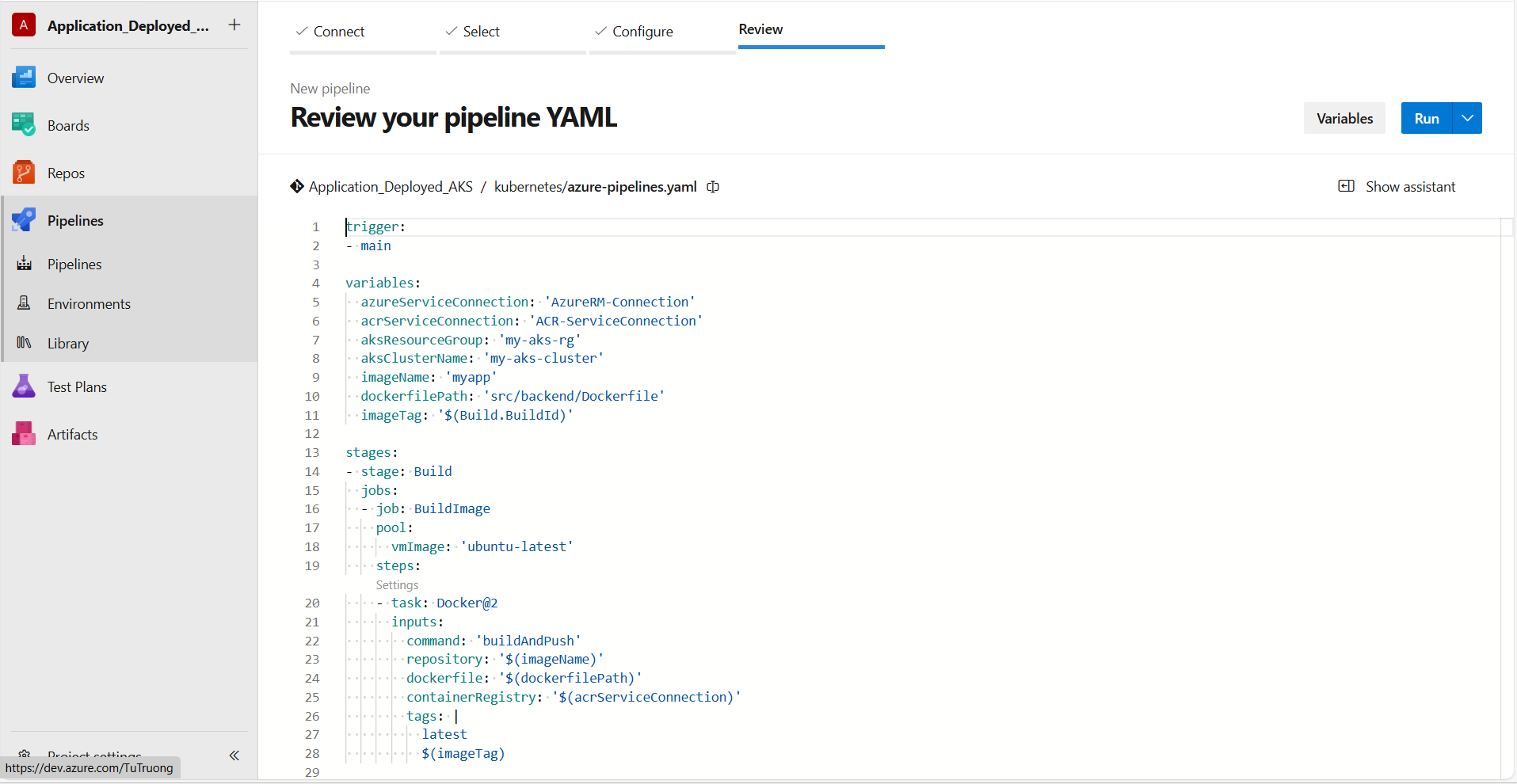


1. Select the repository:

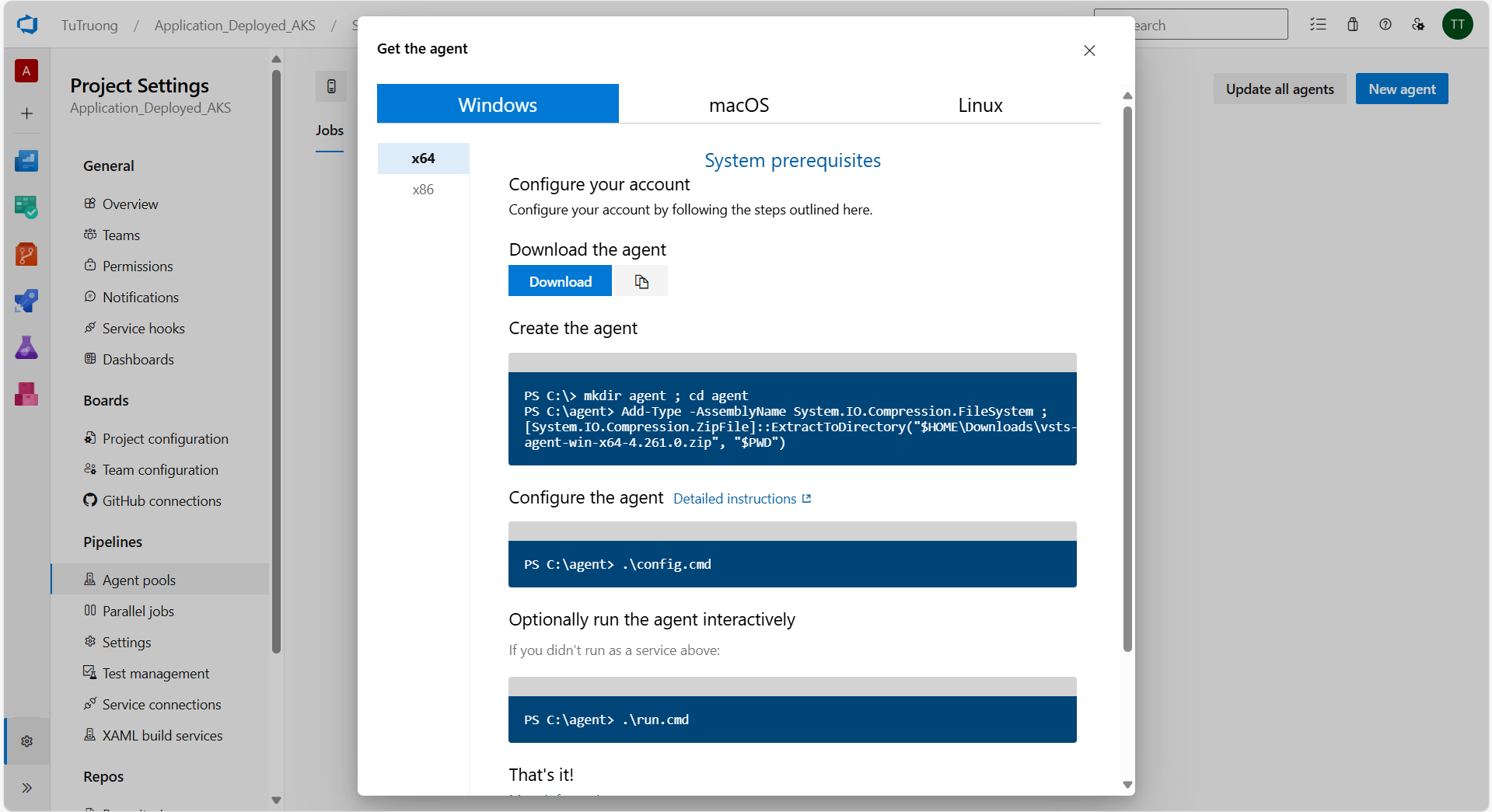


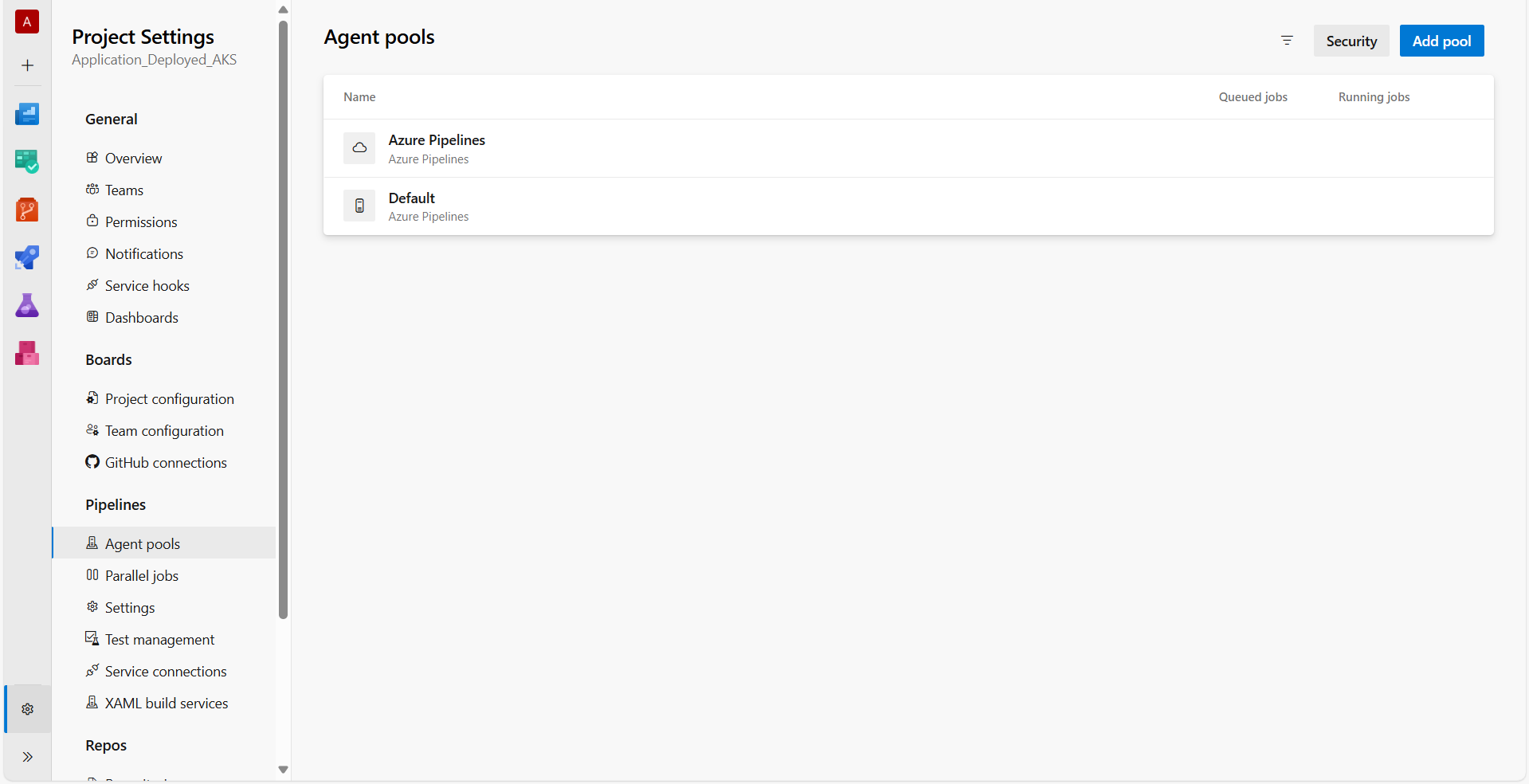
1. Select **“Existing Azure Pipelines YAML file”:**
2. Review pipeline yaml:



****

1. Create a Self-Hosted Agent in Azure DevOps:
2. Open Azure DevOps Organization => select project
3. Click **“Project Settings”** icon
4. Select “**Agent Pools**” => Click on **Default**
5. **Add a New Agent**

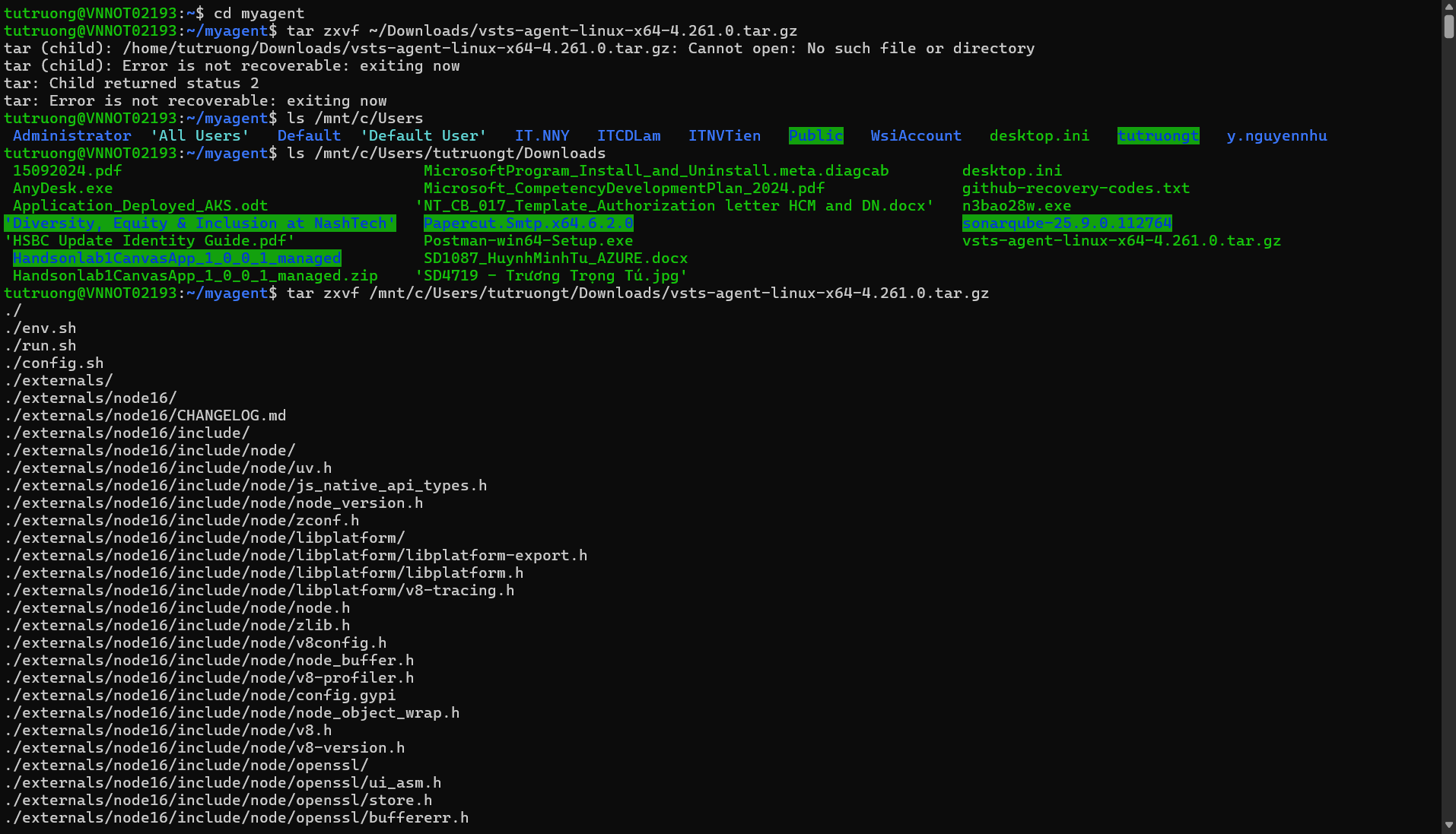




1. Choose OS and Download: using **WSL (Ubuntu)** => choose **Linux**



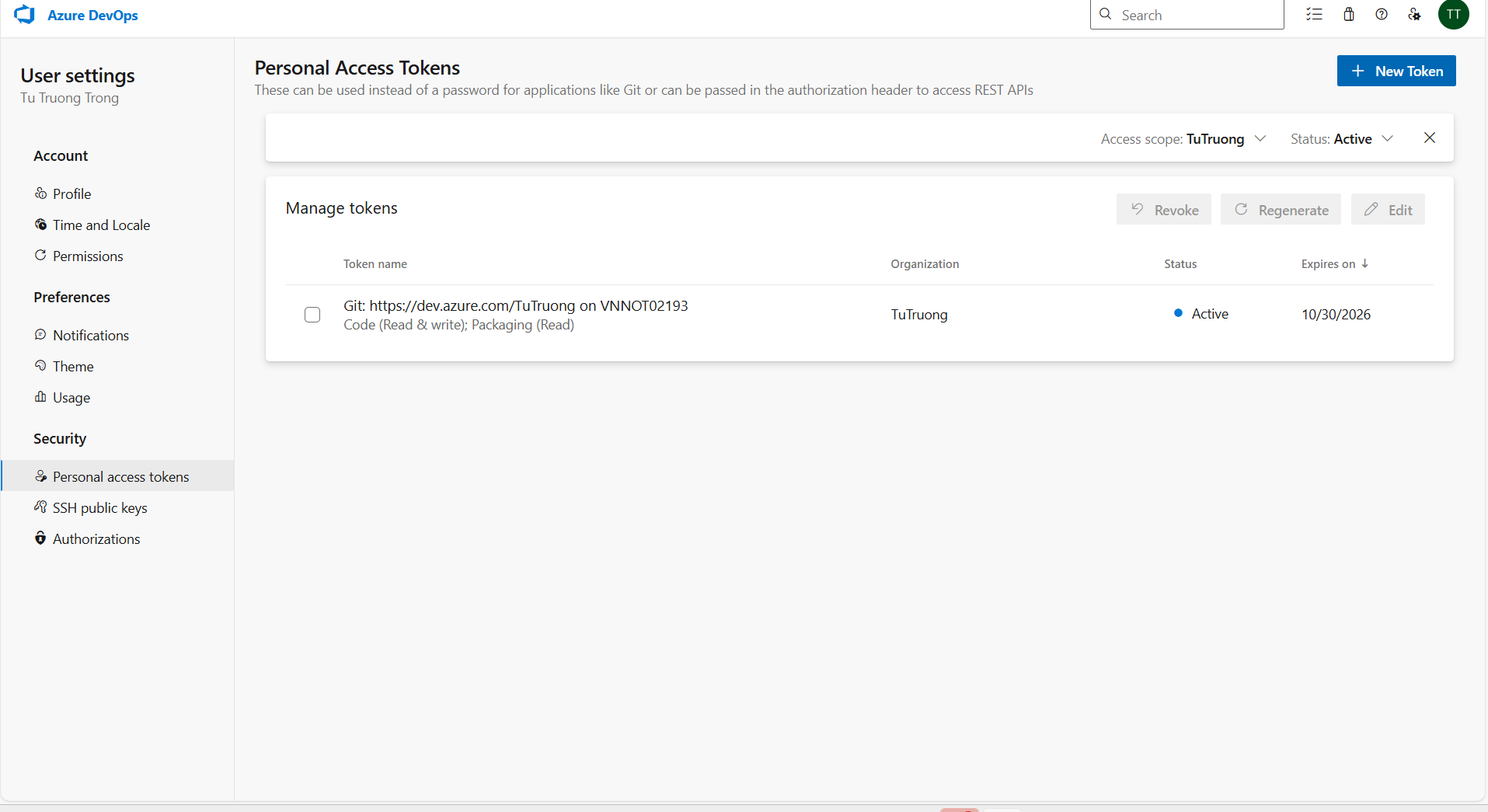
**Run these command on WSL Terminal to create a agent:**

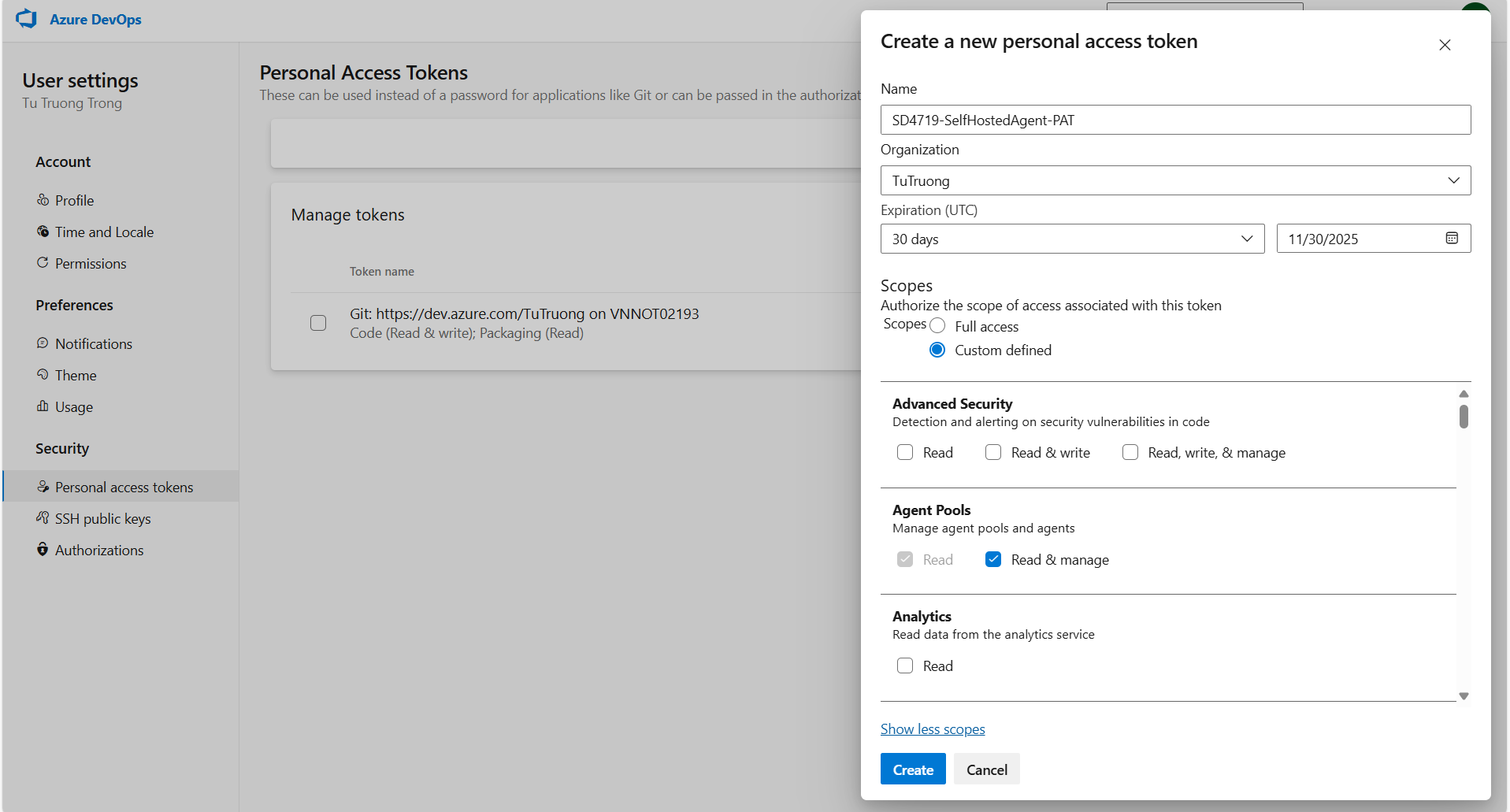


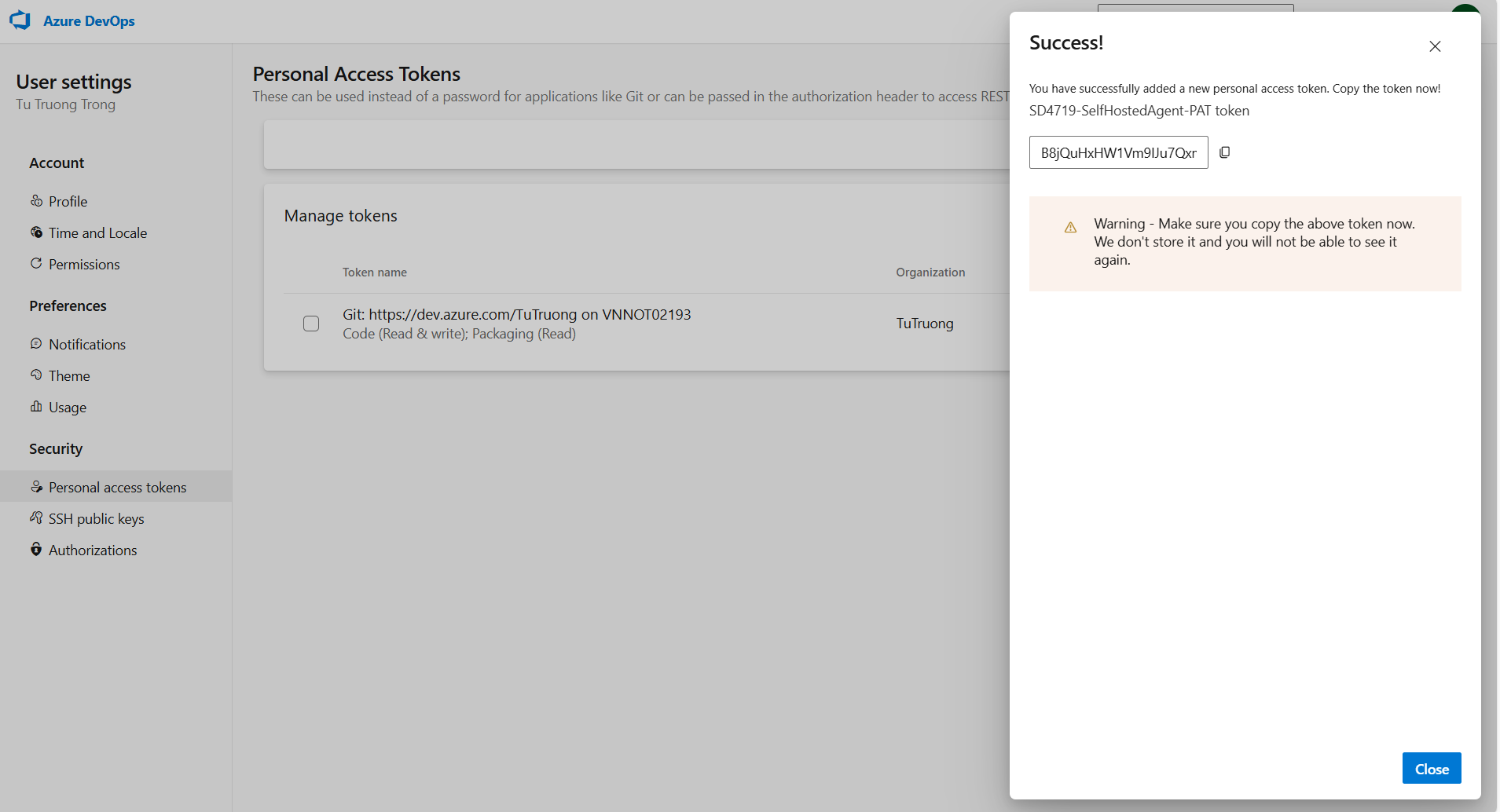
**- Create a Personal Access Token (PAT):**

**+ Open User Settings => click profile picture and select Personal access tokens**

**+ Click add new token => choose the options properly to configure the PAT correctly => click **Create****

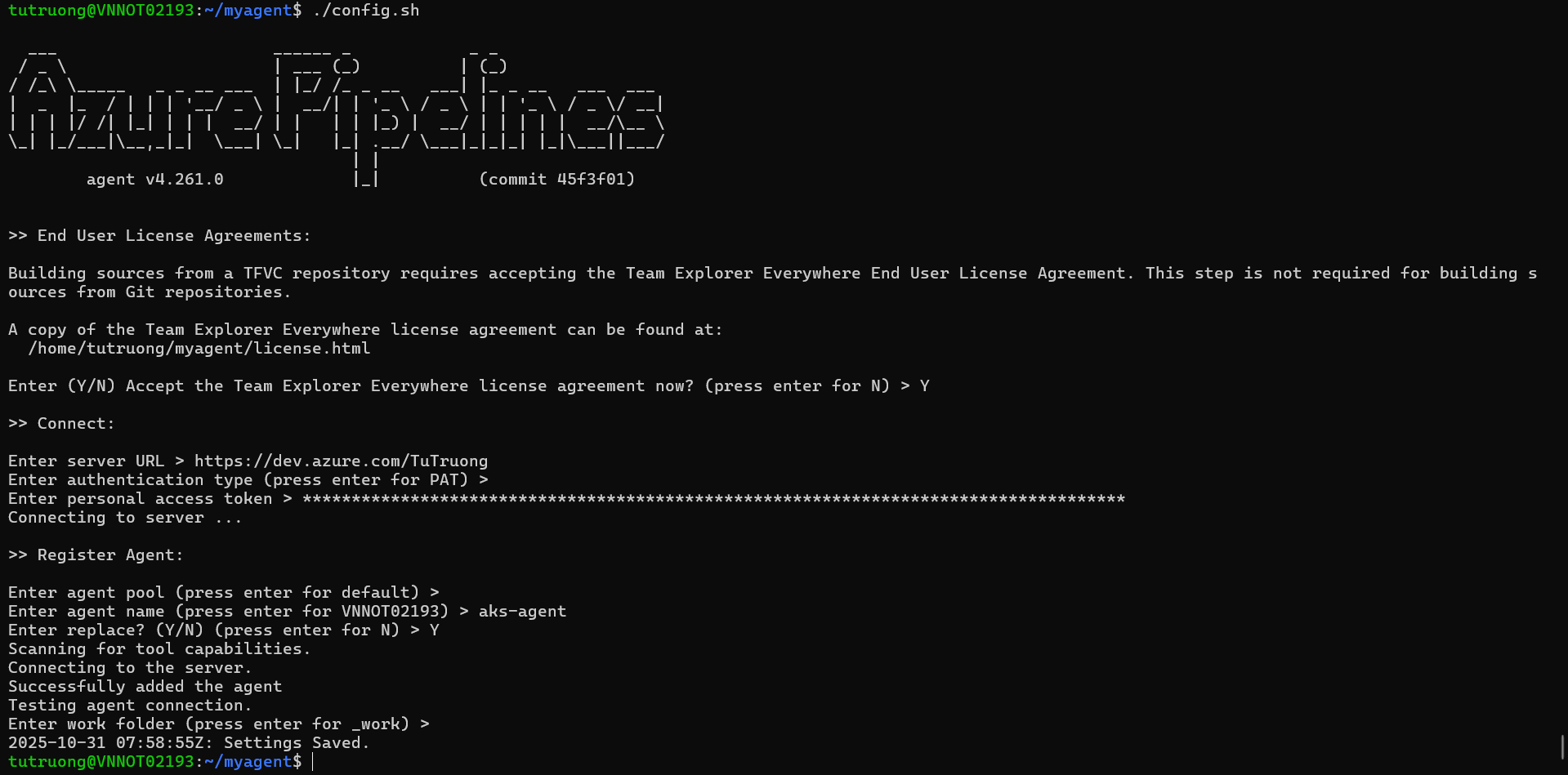






**- In WSL Terminal, run **./config.sh**:**

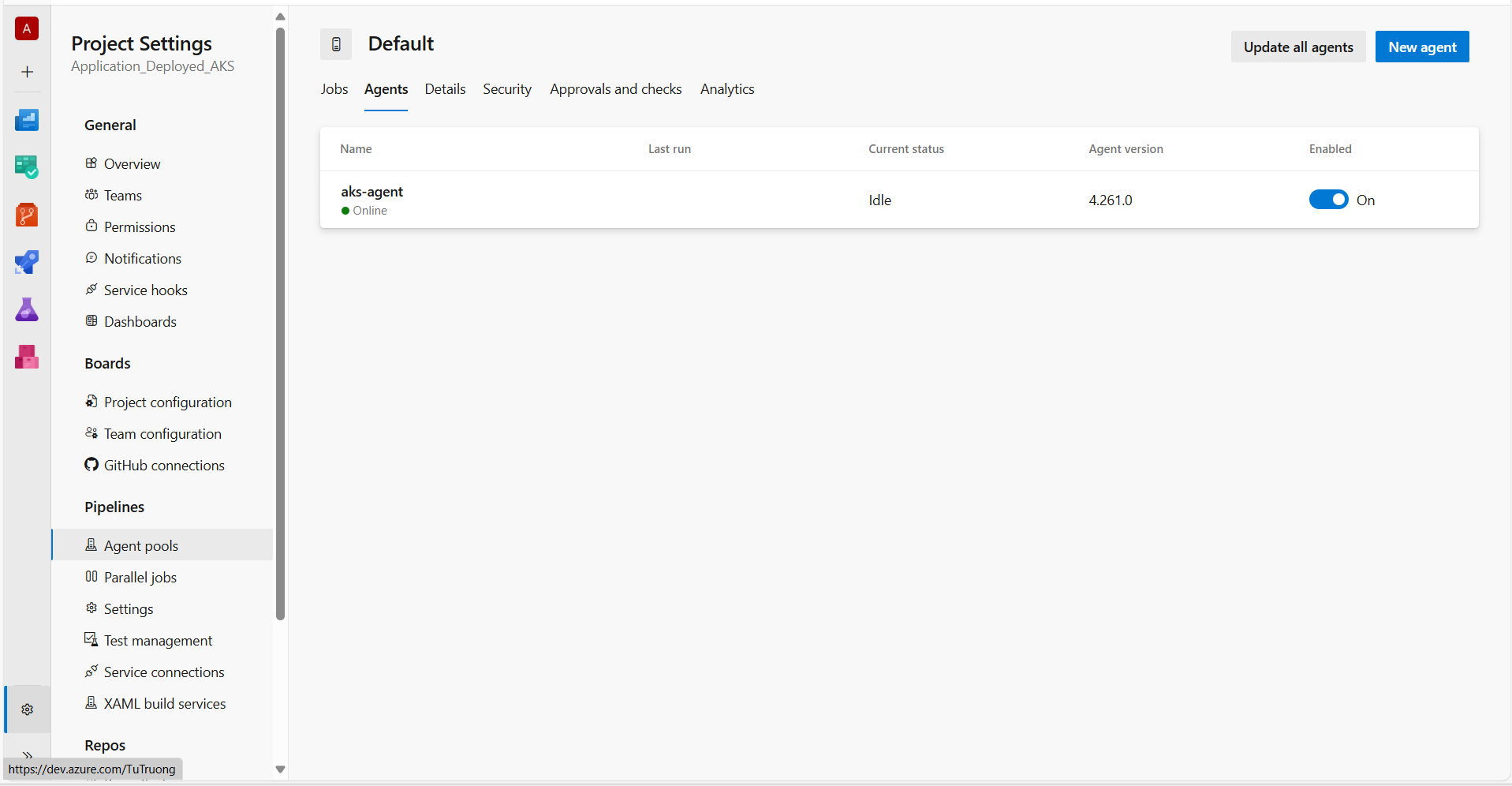
**+ Paste this PAT which was just created into the command and input some remain configurations to register this machine as a new agent inside the “Default” pool**



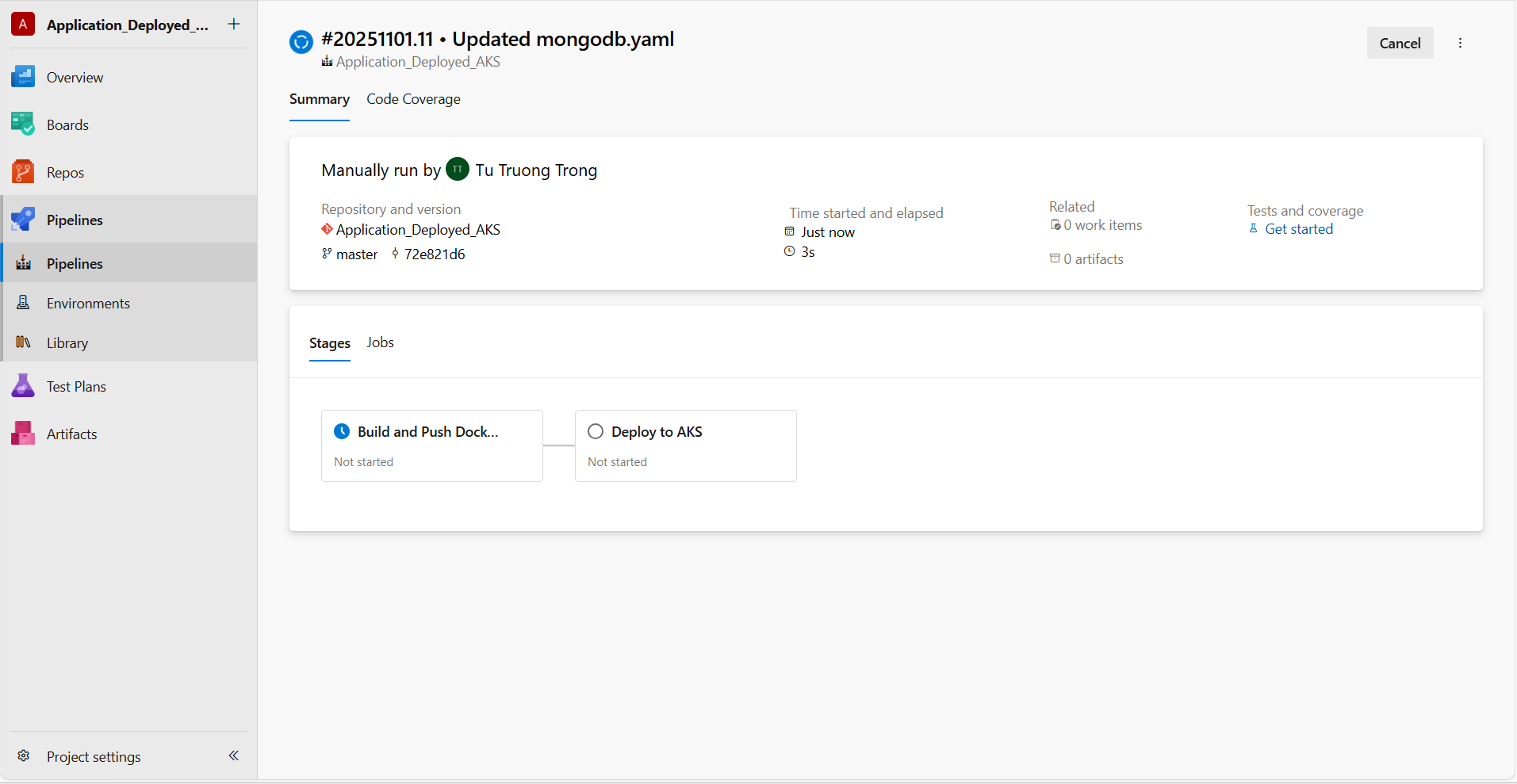
**- In WSL Terminal again, run ./run.sh to start the agent:**



**Now, the **aks-agent** is active**



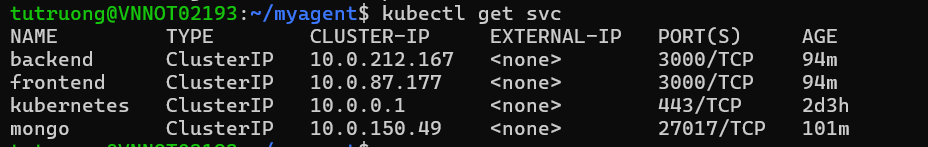
**Update azure-pipelines.yaml file => Run**





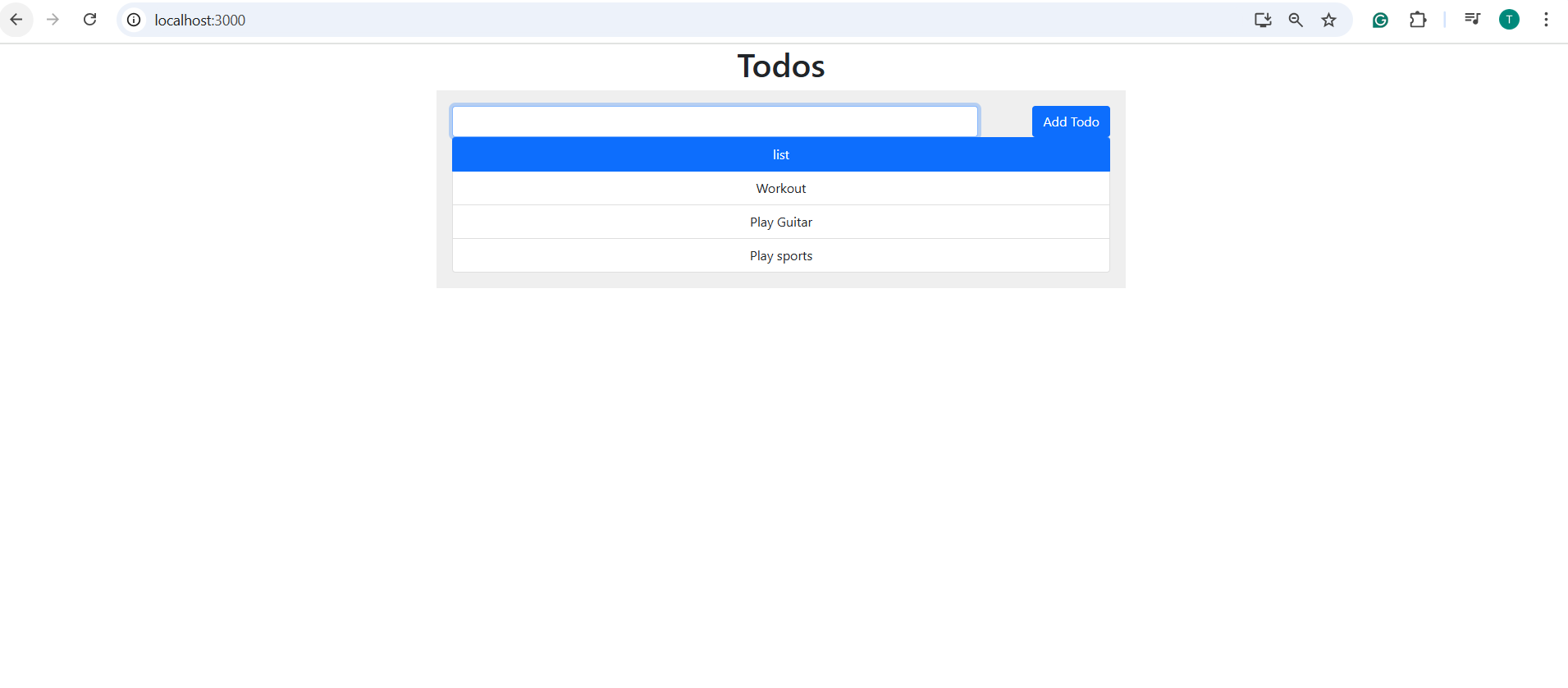
**After Pipeline run successfully => check and run with commands:**

kubectl get svc



**kubectl port-forward service/frontend 3000:3000**

**Then visit: http://localhost:3000**



****Set up Prometheus and Grafana to monitor AKS resources and default AKS resource metric:****

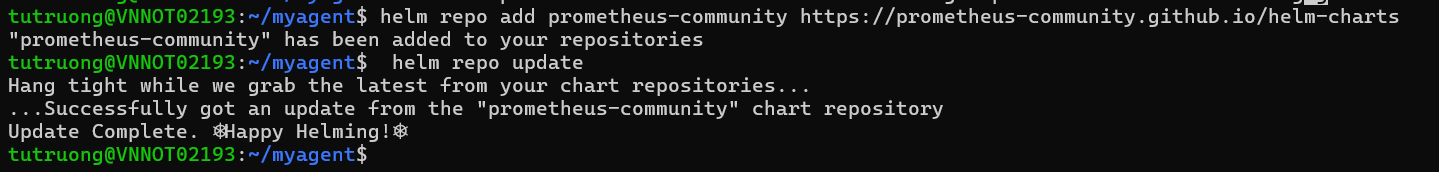
1. **Add Helm repo:**

**helm repo add prometheus-community https://prometheus-community.github.io/helm-charts**

**helm repo update**

### **Create a namespace:**

kubectl create namespace monitoring

****

1. **Create a custom values-monitoring.yaml** (stored in folder kubernetes/monitoring/values-monitoring.yaml)

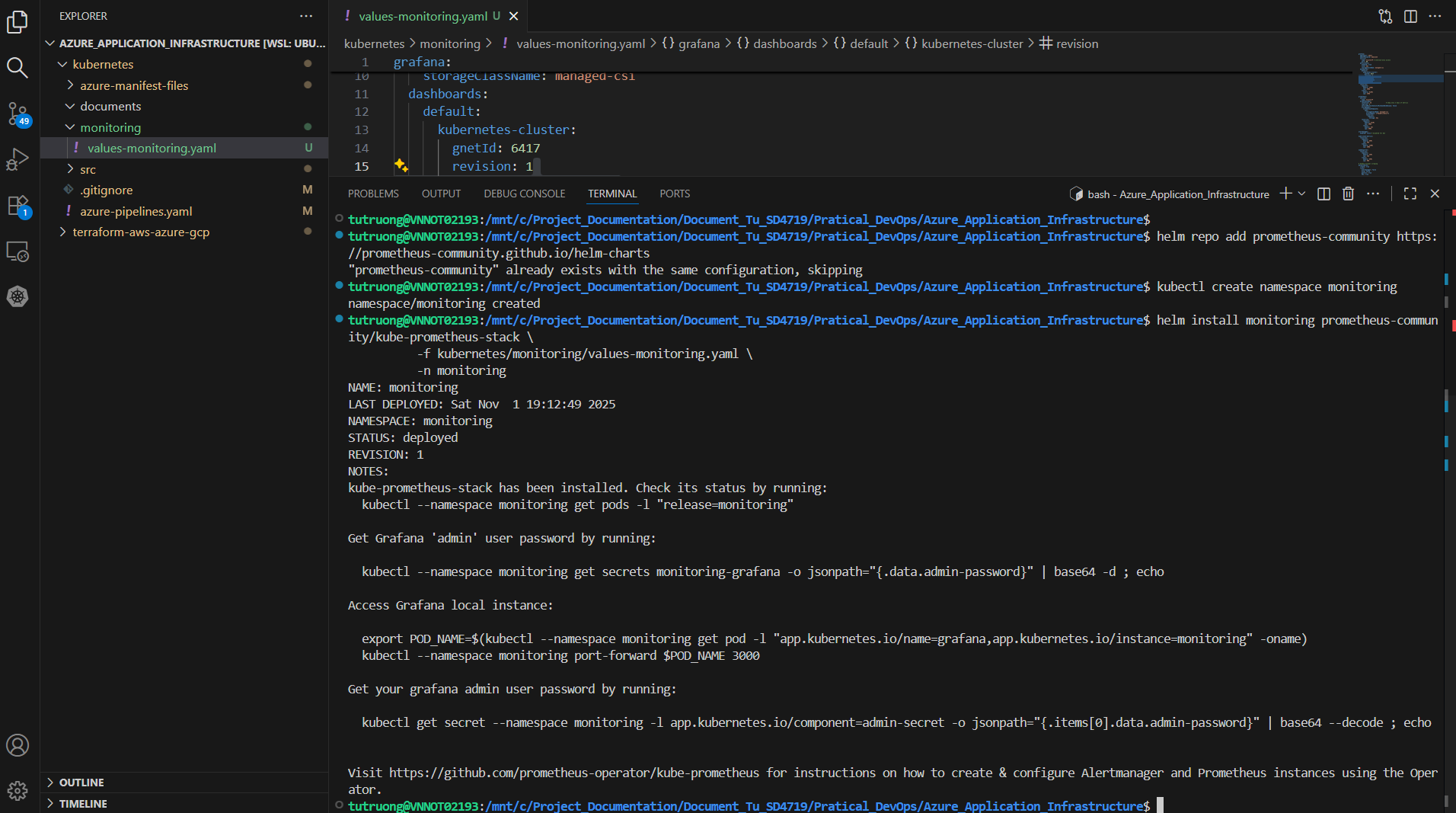
### **Install the Helm chart:**

**I share my source code folder with WSL to re-create a namespace monitoring and run this command to install monitoring with **helm**:**

helm install monitoring prometheus-community/kube-prometheus-stack \

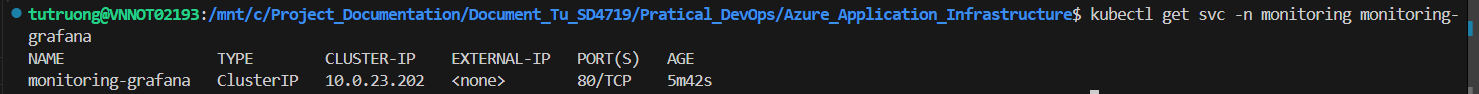
-f kubernetes/monitoring/values-monitoring.yaml \

-n monitoring

****

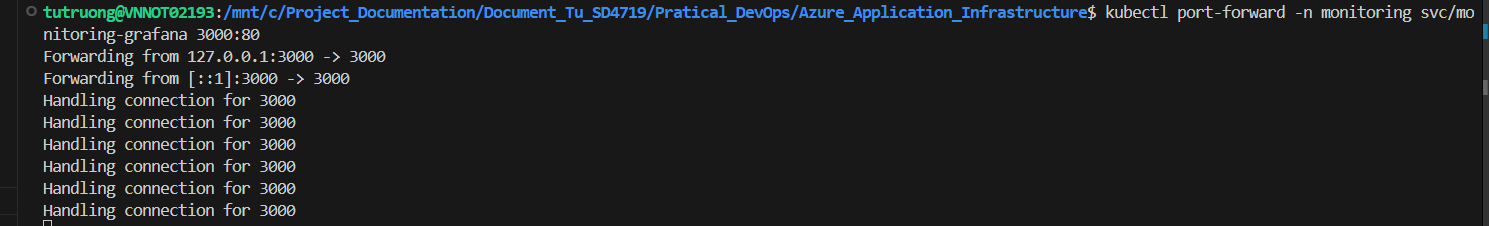
1. ****Access Grafana Dashboard:****
   1. **Get **Grafana** service, run with:**

**kubectl get svc -n monitoring monitoring-grafana**



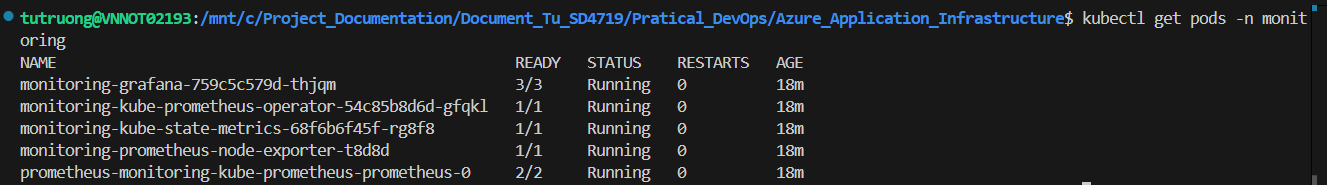
* 1. **Use port-forward to redirect into Grafana Dashboard, run with:**

**kubectl port-forward -n monitoring svc/monitoring-grafana 3000:80**

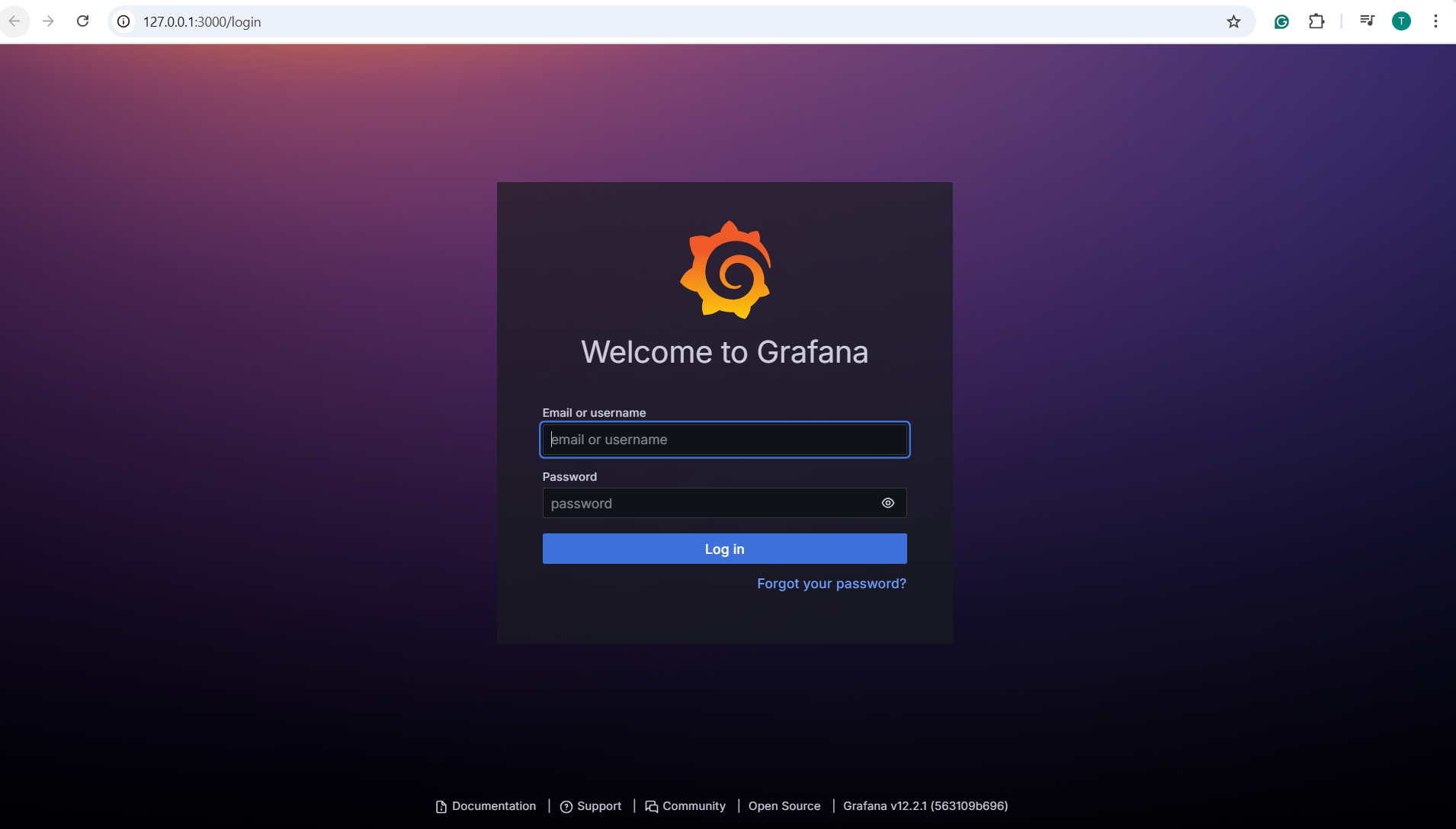


**3. Verify Prometheus is scraping AKS metrics, run with:**

**kubectl get pods -n monitoring**



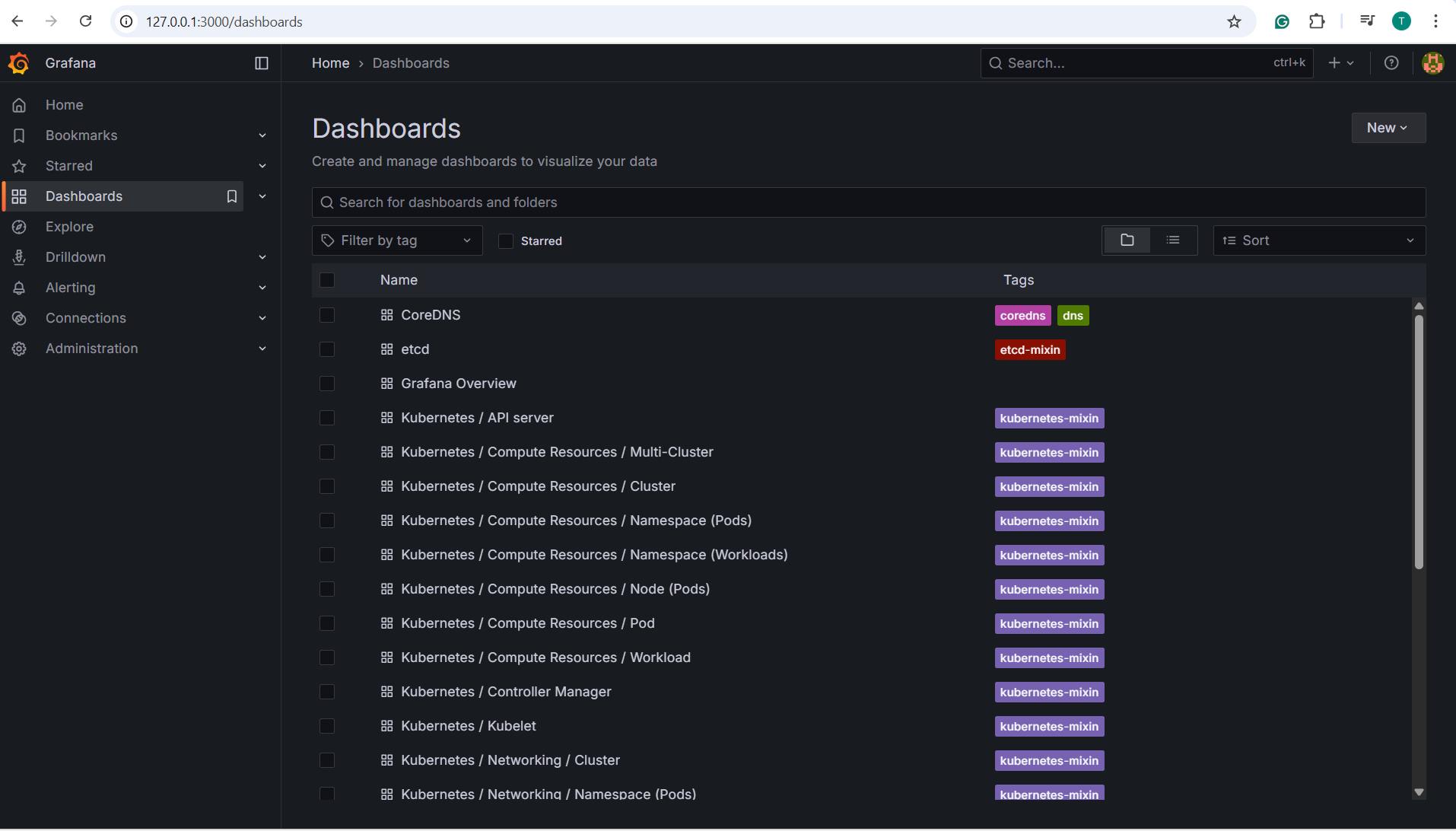
**Then visit: http://127.0.0.1:3000/login**

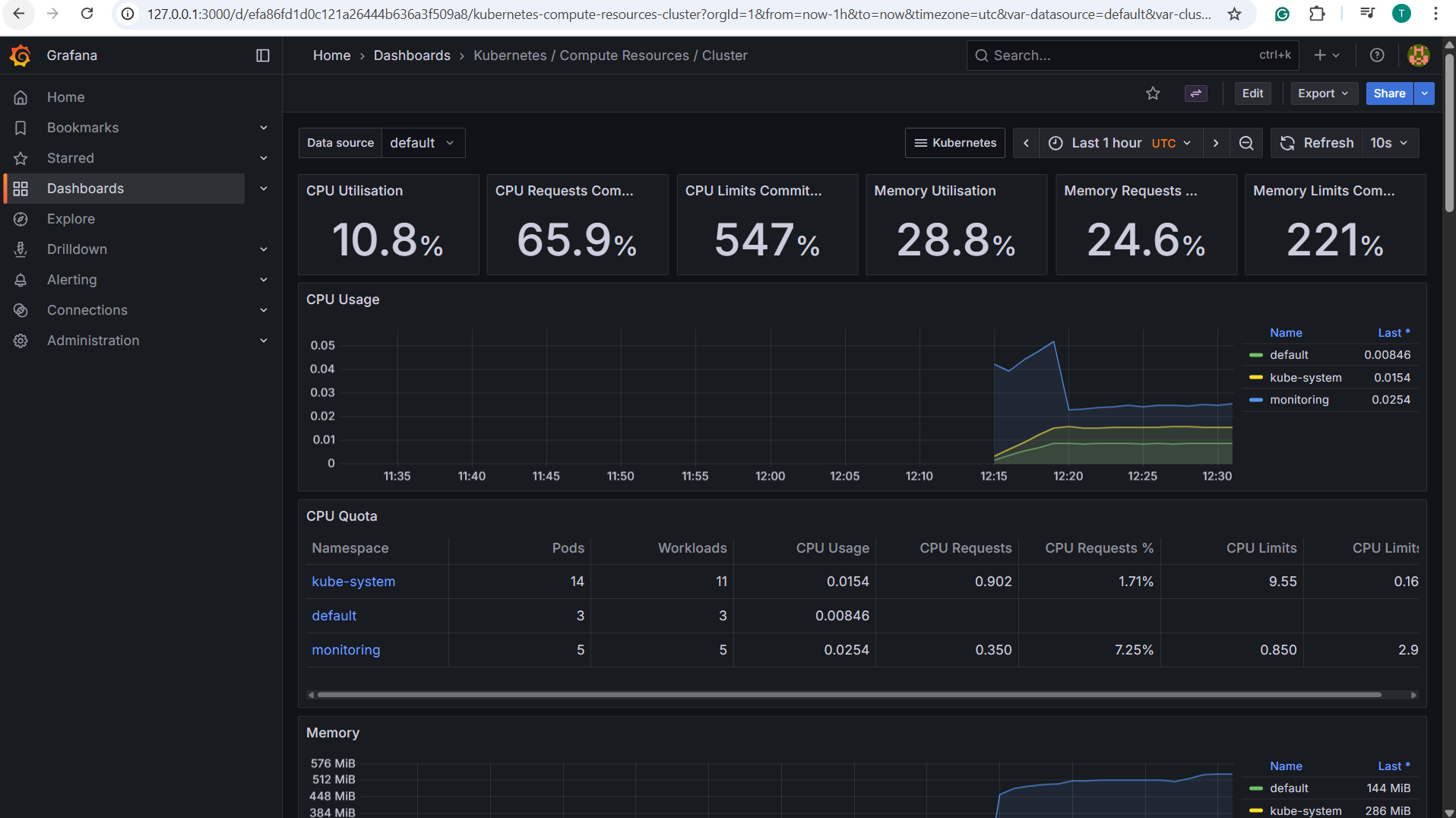


**Now, we can login into the Grafana Dashboard with the account info that defined in the **values-monitoring.yaml** file:**

****User: admin****

****Password: admin123****





****If any updates are in the values-monitoring.yaml file, run with this command to update:****

**helm upgrade monitoring prometheus-community/kube-prometheus-stack \**

**-f kubernetes/monitoring/values-monitoring.yaml \**

**-n monitoring**