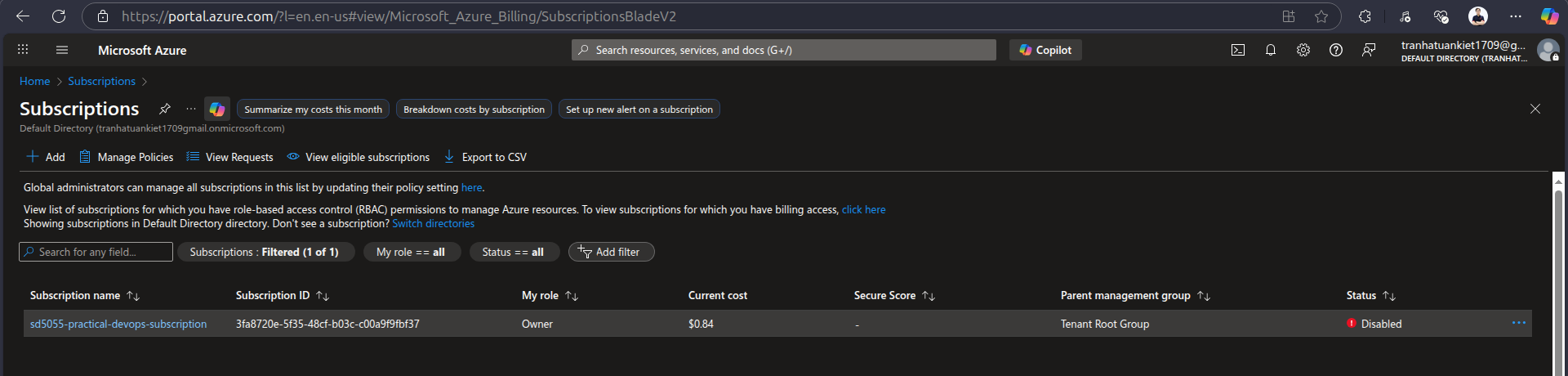
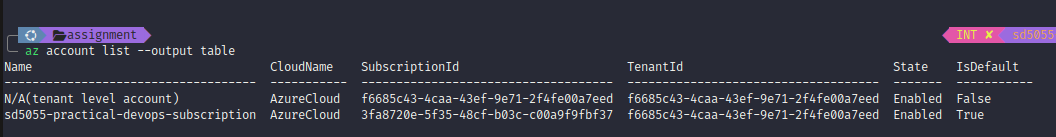
Practical DevOps with Azure infrastructure

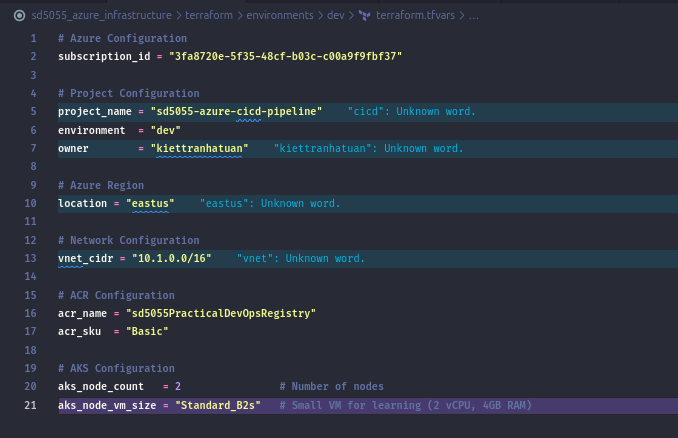
## Create Azure infrastructure with Terraform

Build an Azure infrastructure with Terraform through this repo: https://github.com/tuankiet1709/sd5055\_azure\_infrastructure

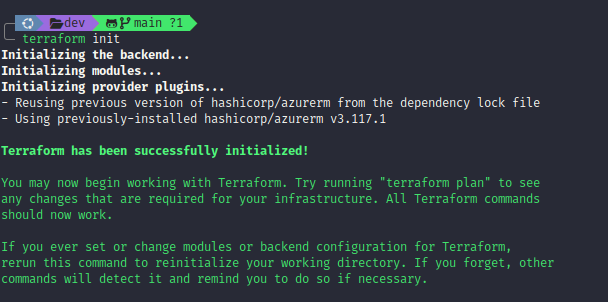
* First, we need to create a subscription if not exist
* Install azure cli (refer to: https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-linux?view=azure-cli-latest&pivots=apt)
* Login to az local with az login
  + I will open a azure login page and we will login on page
* After login, we can use `az account list –output table` to get all subscription in account



* Update subscription\_id to terraform variables



* We will run `terraform init` to init the terraform infrastructure



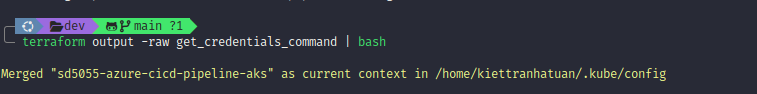
* After initializing terraform successfully, we can use `terraform apply` to create the azure services that we defined in terraform



* After applying terraform successfully, we can see all resources we defined on AWS

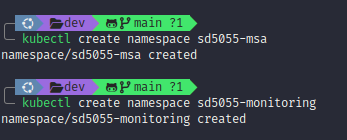


* Add AKS to kube config

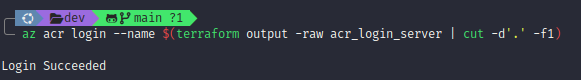


## 

* Create namepsace on aks

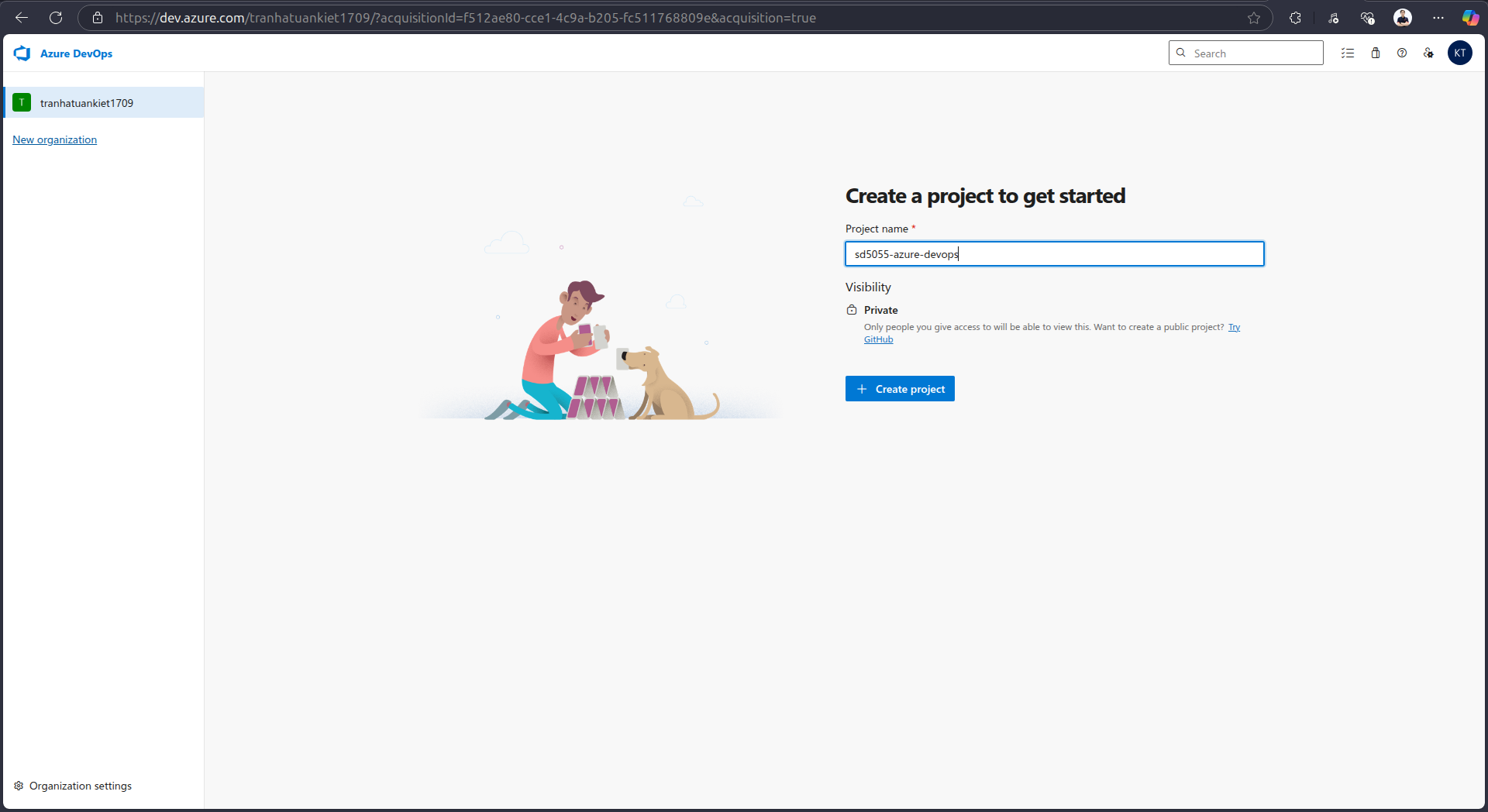


* Login to ACR



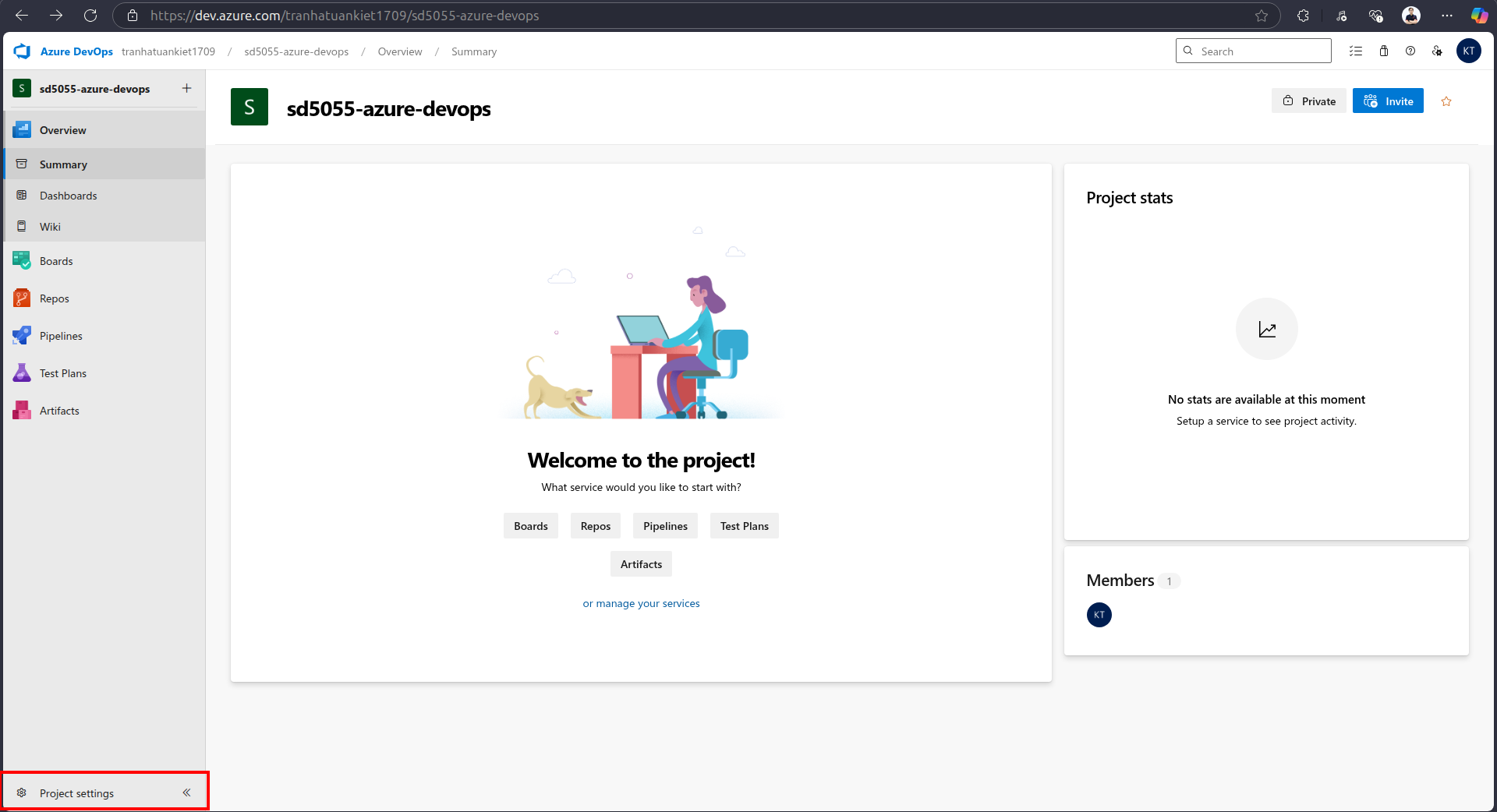
## Deploy MSA application to EKS using Azure DevOps CI/CD

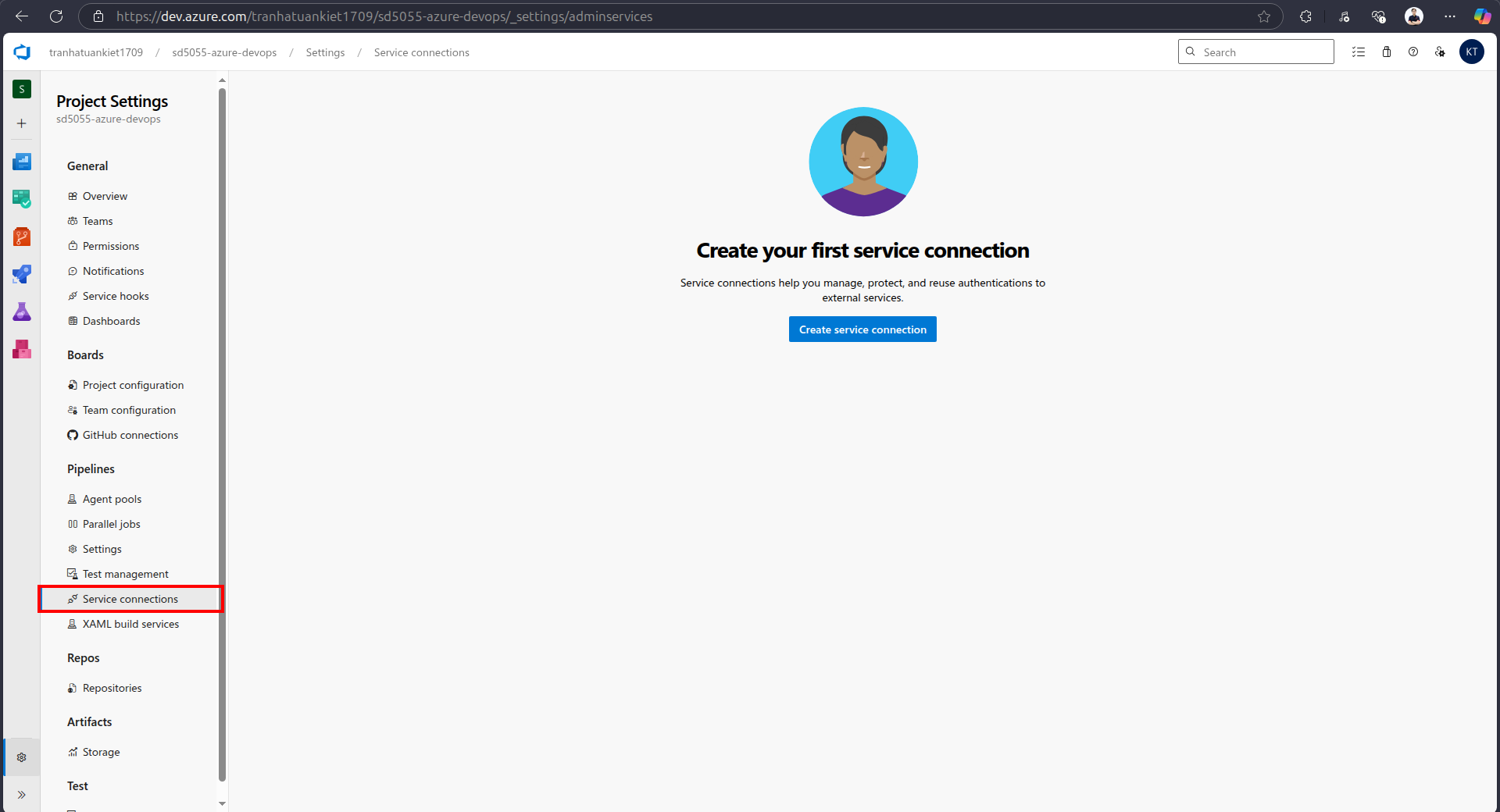
* We need a MSA repo including frontend and backend: <https://github.com/tuankiet1709/sd5055_msa>
* To run CI on azure-pipelines-ci.yml, we need to define a azure-pipelines-ci.yml in BE and FE sources.
* azure-pipelines-ci.yml will define which agent pool use to run CI, the stage of CI is a step to install, build, validate, test, and push to ECR
  + https://github.com/tuankiet1709/sd5055\_msa/blob/main/backend/azure-pipelines-ci.yml
  + https://github.com/tuankiet1709/sd5055\_msa/blob/main/frontend/azure-pipelines-ci.yml
* When we already have a azure-pipelines-ci.yml, we need to create new project in azure devops

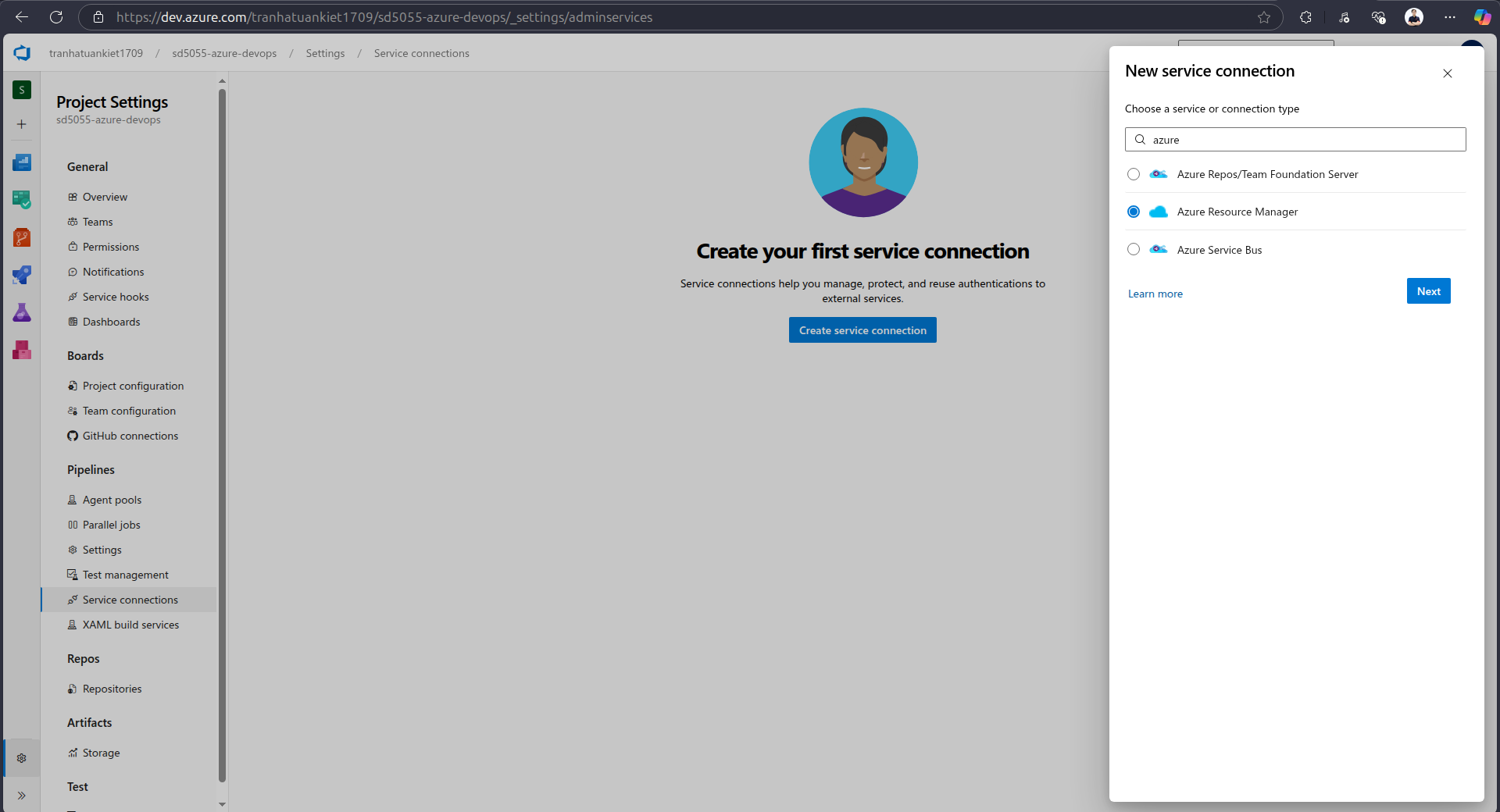


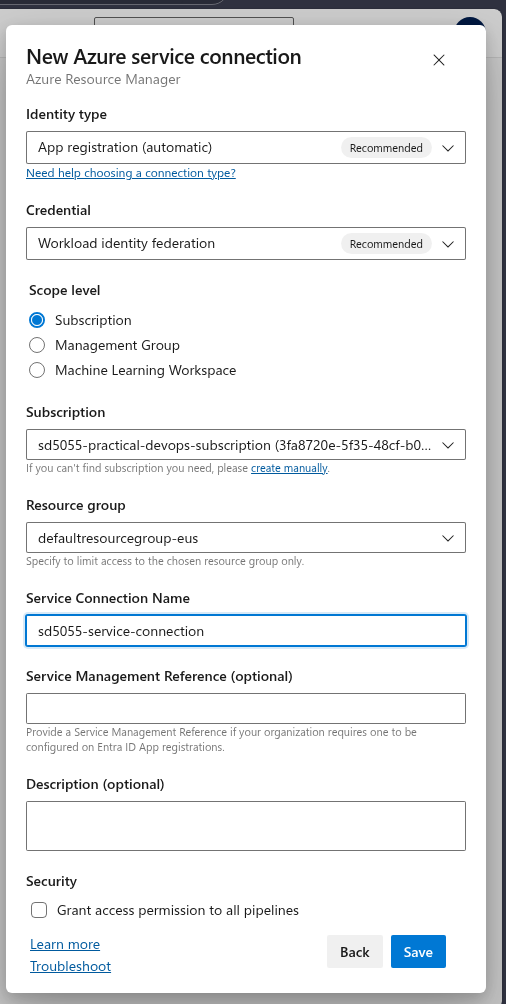
## 

* Go to Project Settings, go to Service Connections under Pipeline section





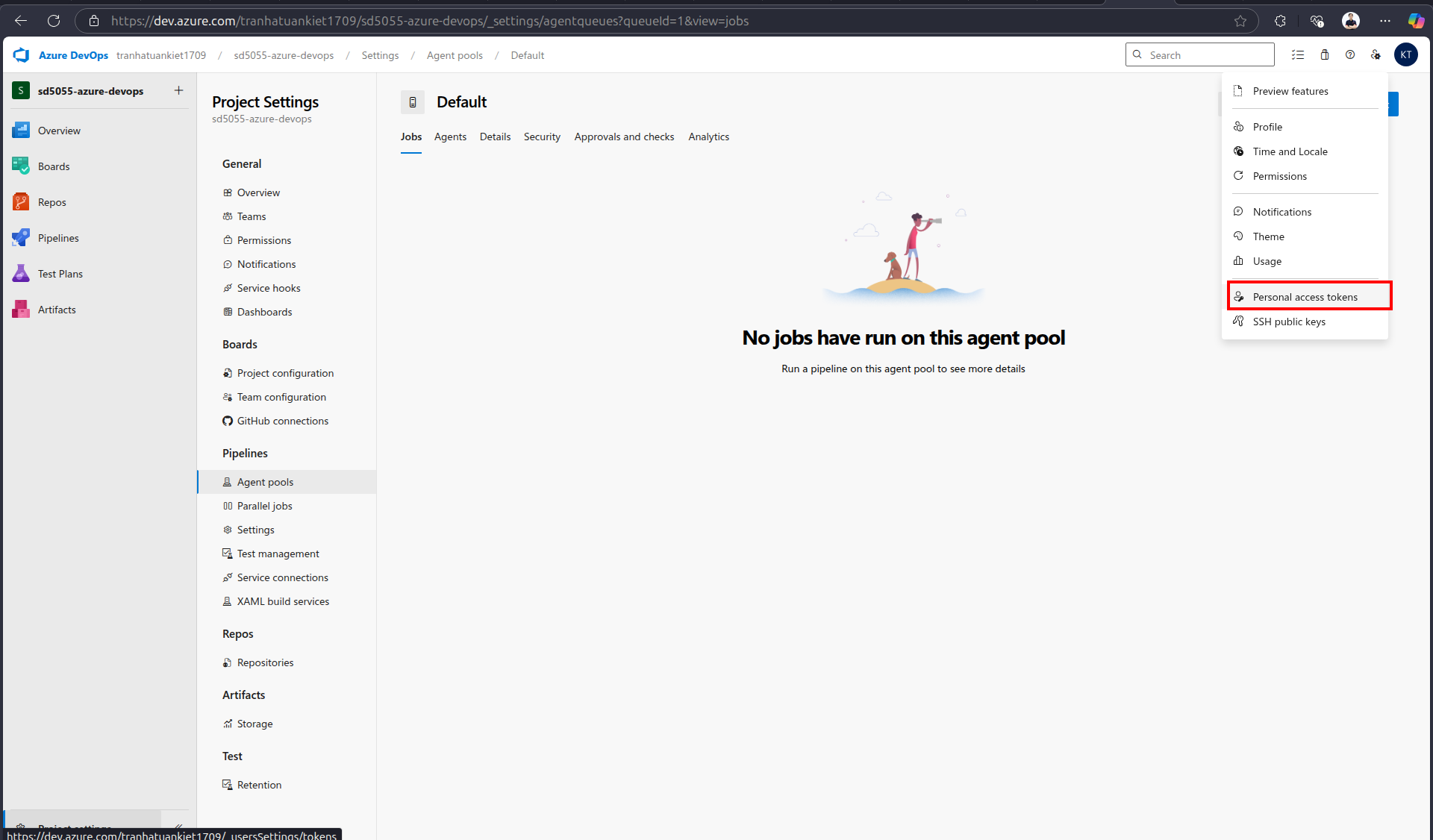




* After create service connection, we need to update to azure-pipelines-ci.yml



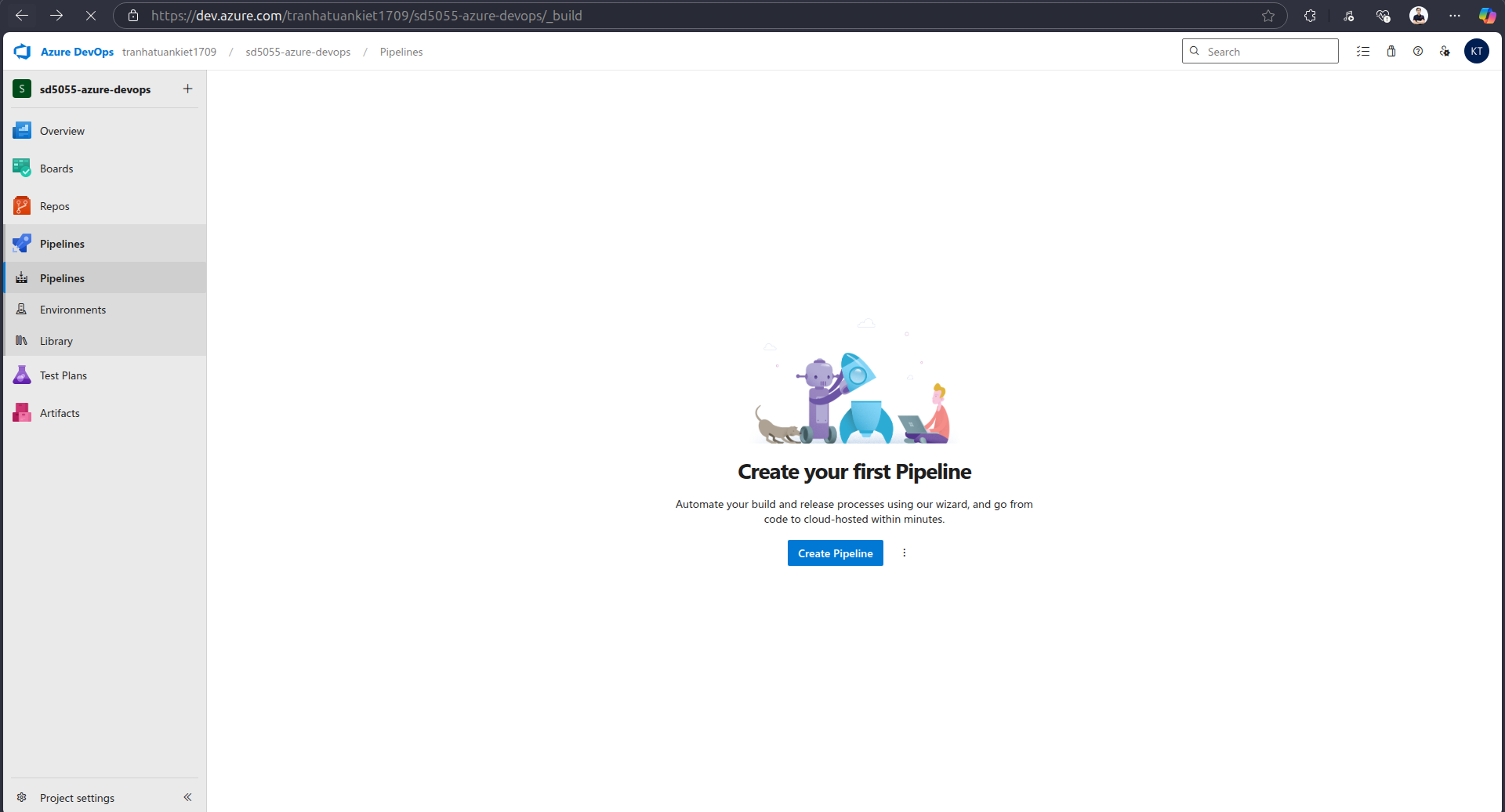
* Create self-hosted agent at local
  + Create Personal access tokens from agent pools in azure devops

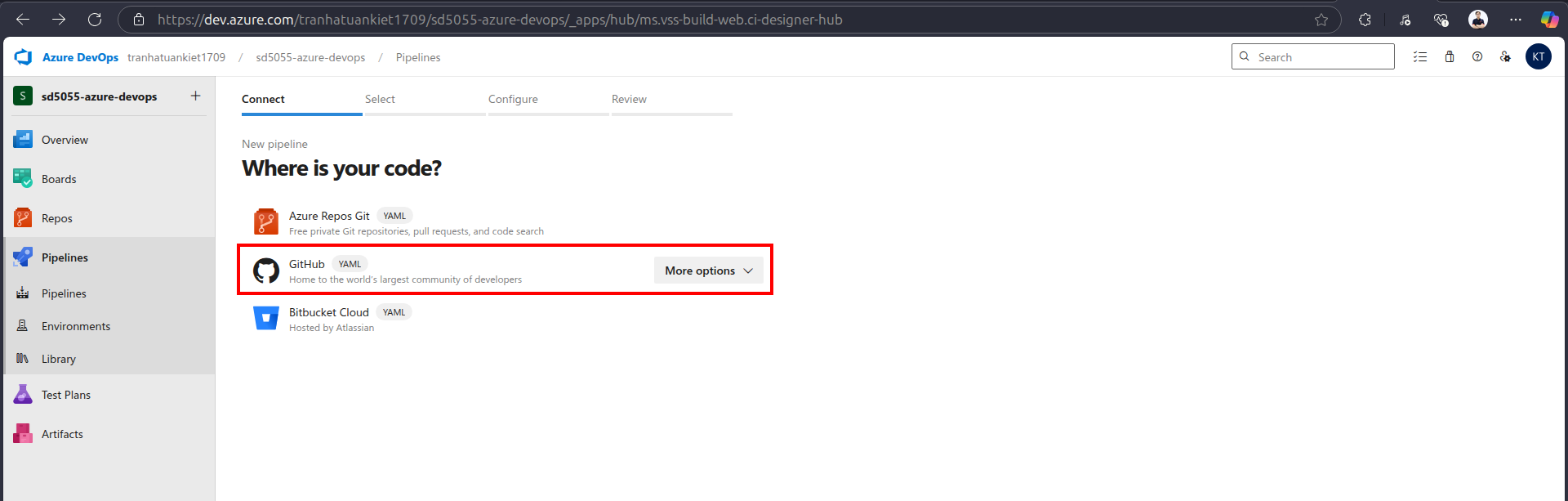


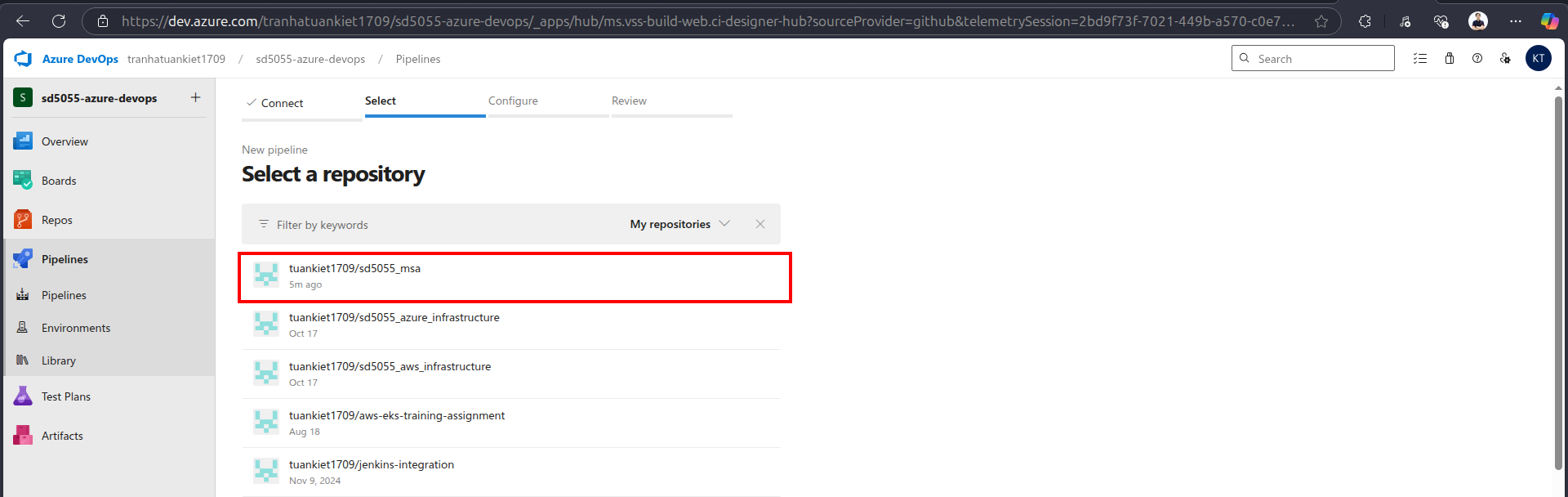
* + Follow the guide in the microsoft page to setup a agent in local (<https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/linux-agent?view=azure-devops&tabs=IP-V4>)

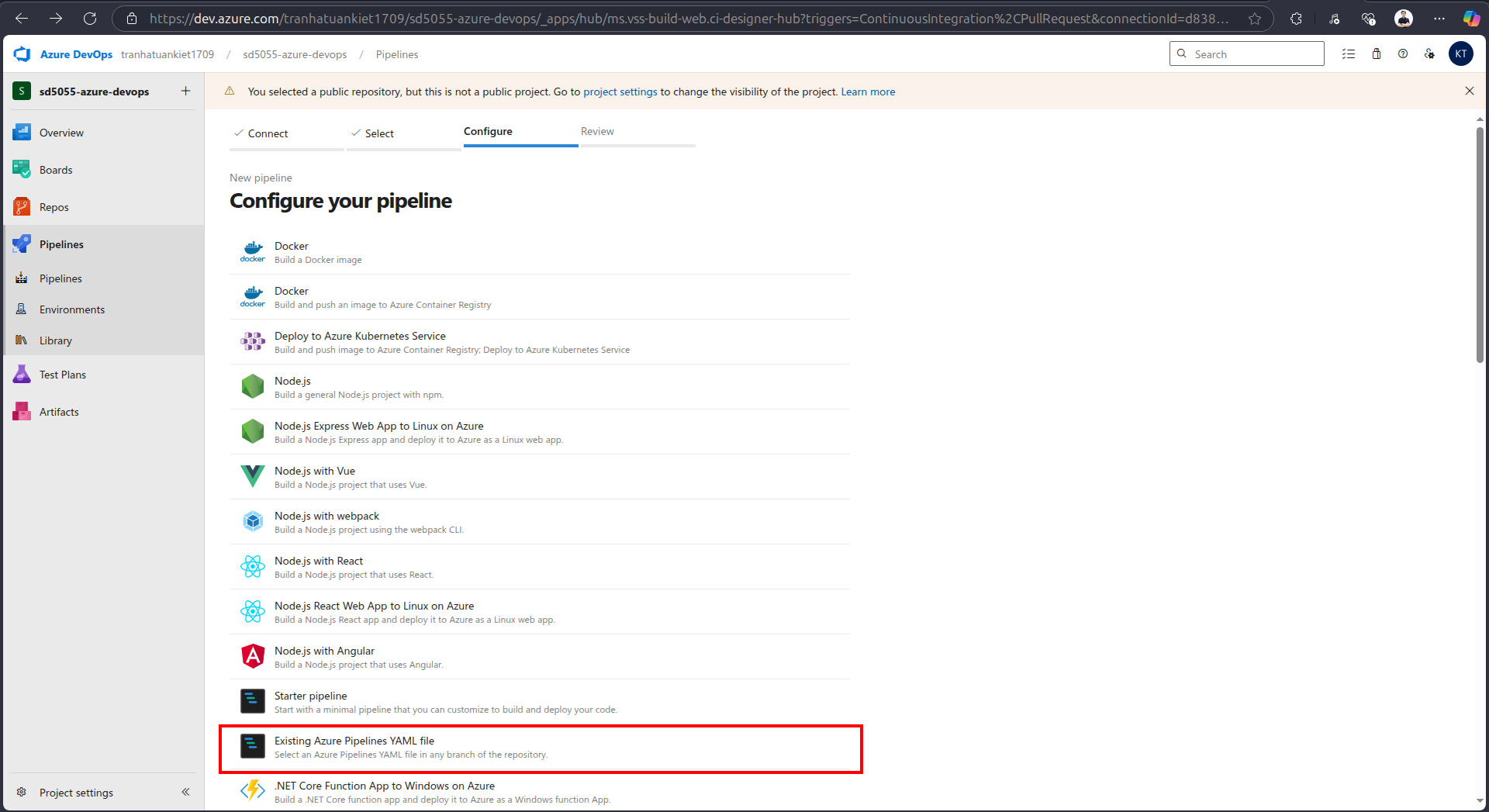


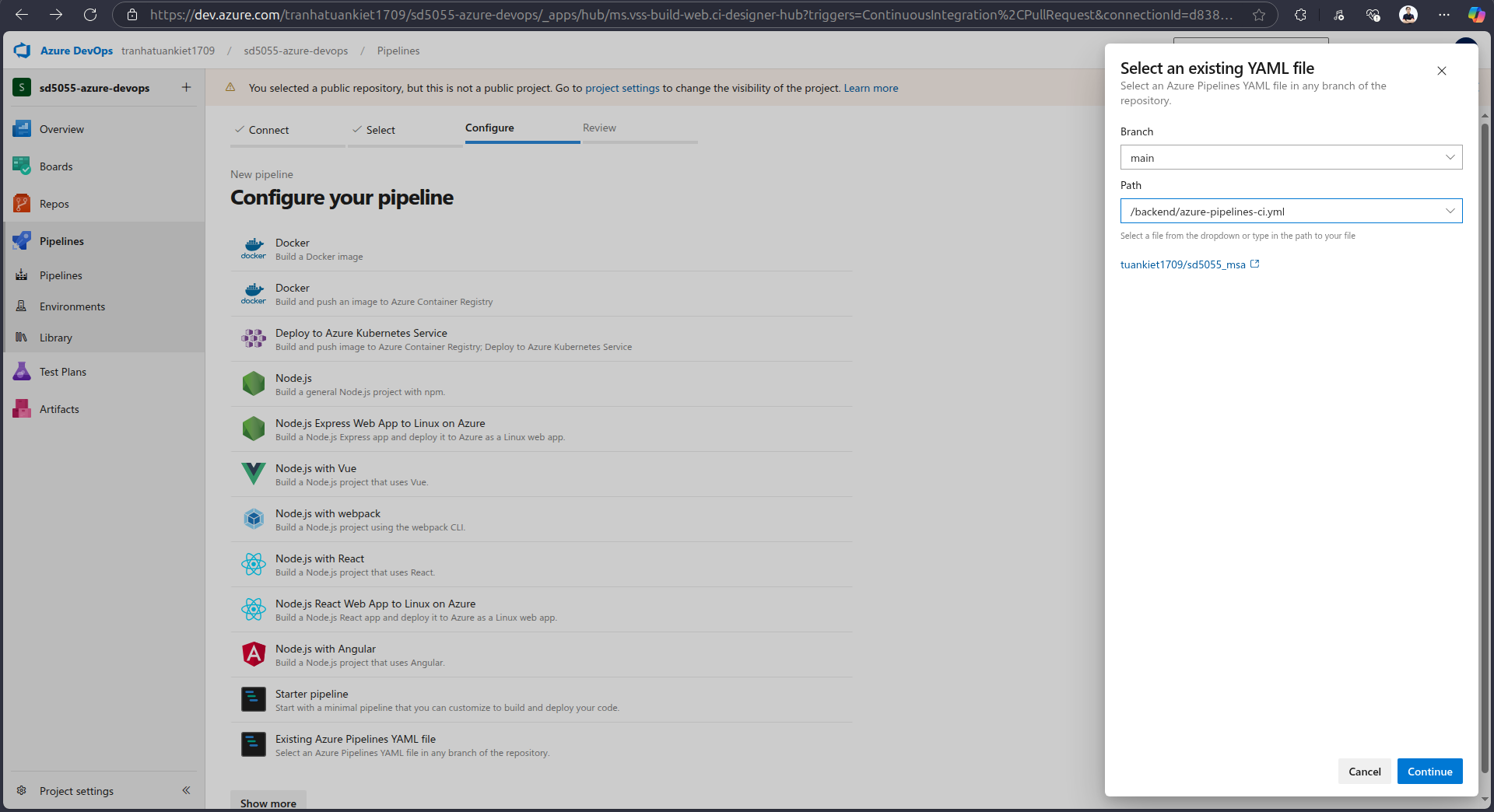
* After setting up the self-hosted agent, we need to update azure-pipelines-ci.yml to use self-hosted agent pool, then we need to create pipeline for backend, frontend, and deployment
  1. Backend
  + Select Create Pipeline in Pipelines section → Select Github for connection → Select sd5055\_msa repo → Select Existing Azure Pipelines YAML file → point to azure-pipelines-ci.yml file in frontend/backend → Review the pipeline YAML file →

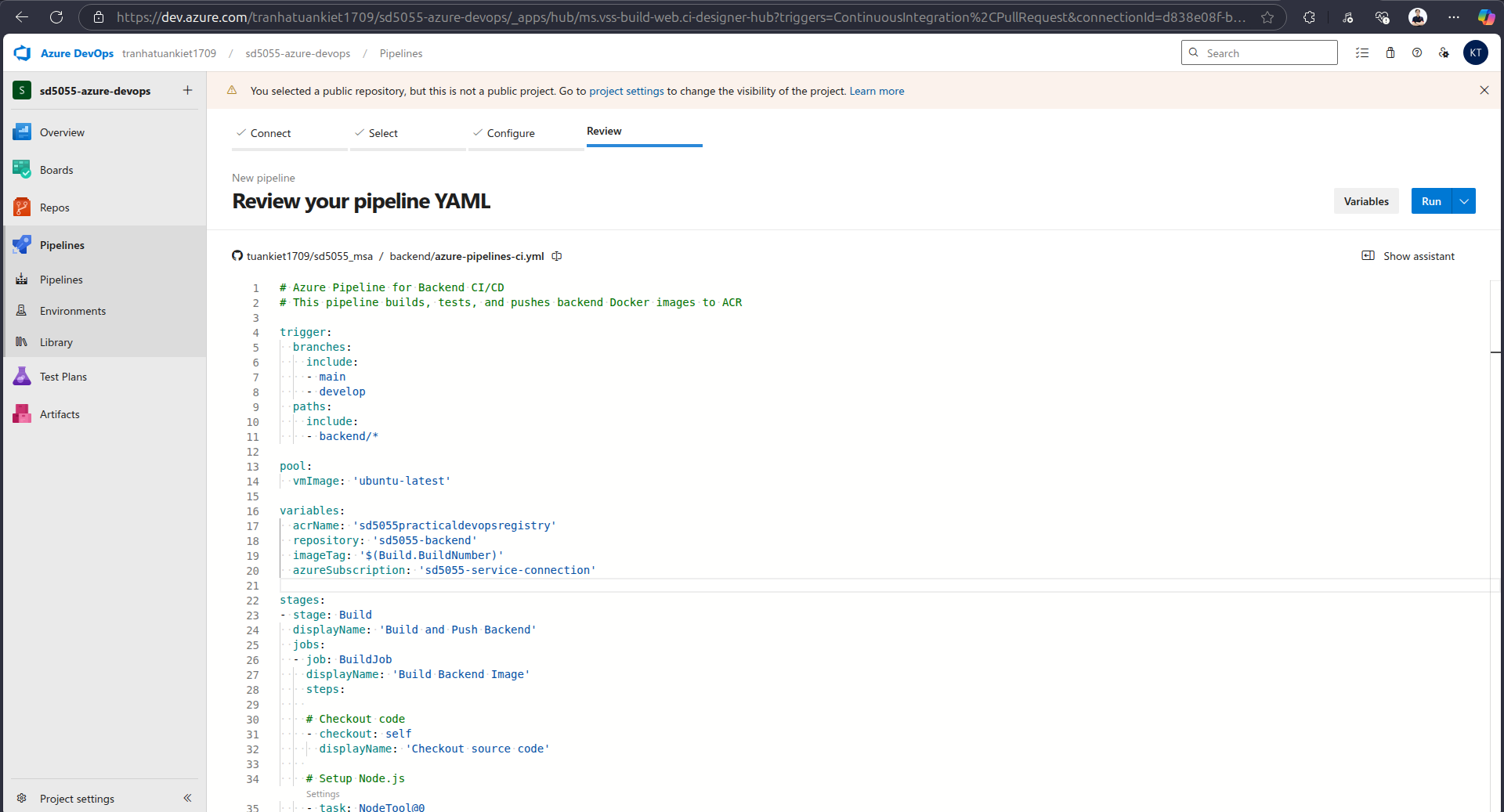


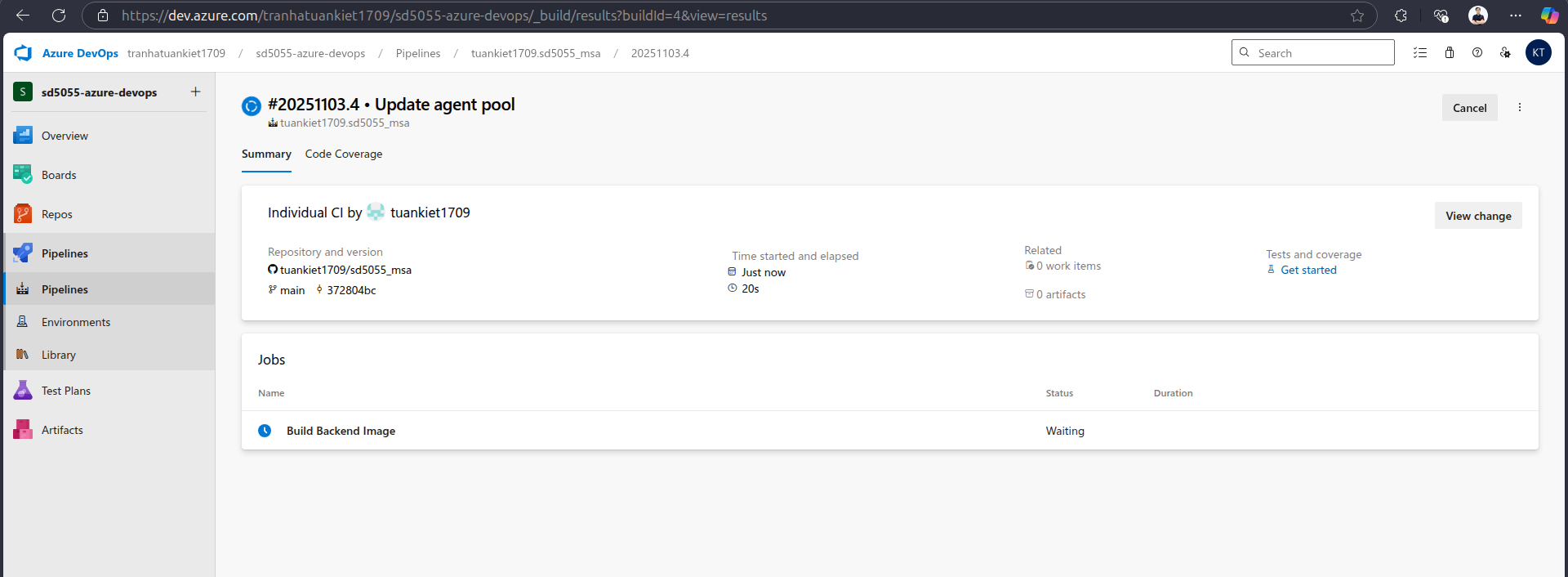


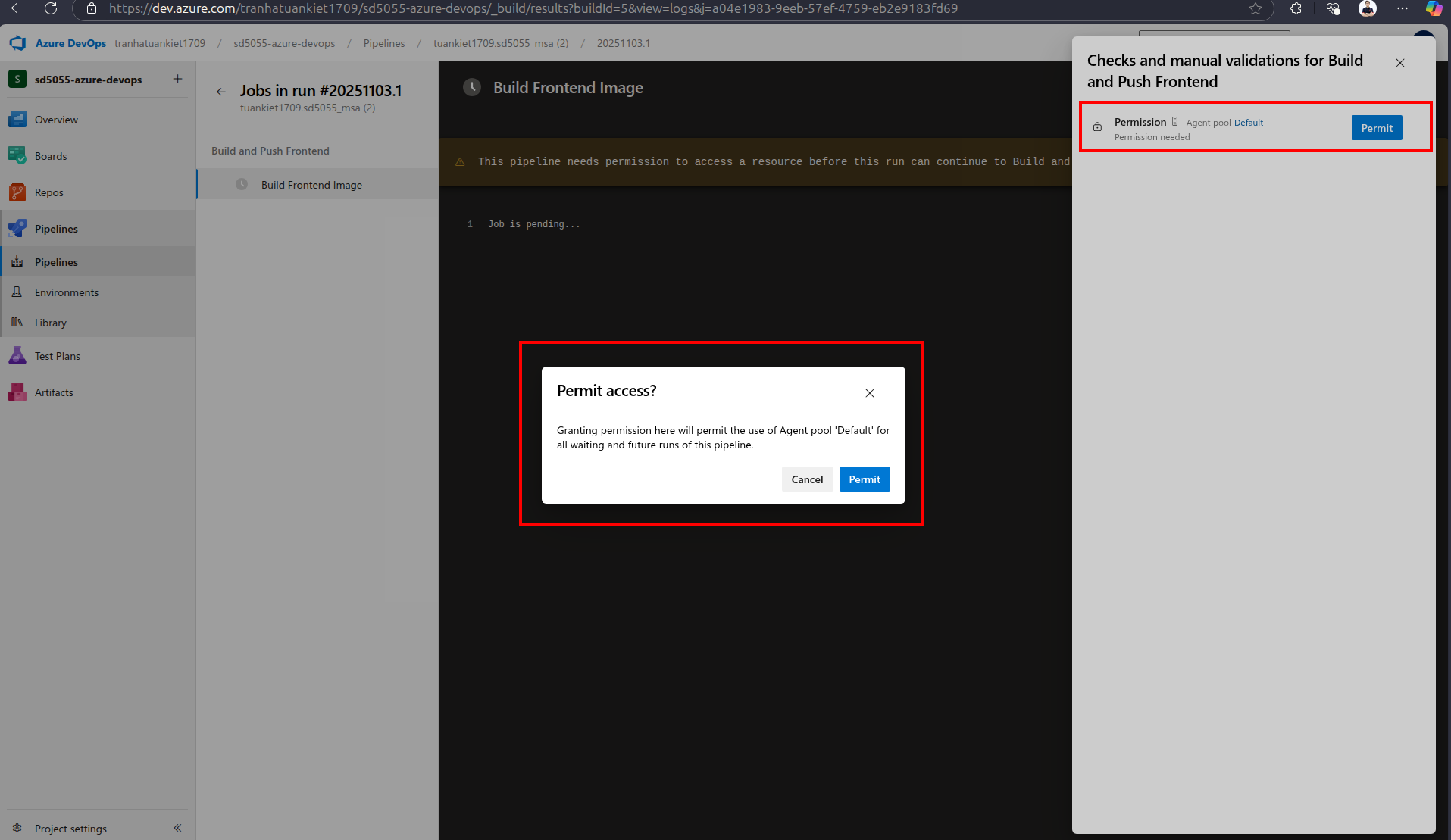


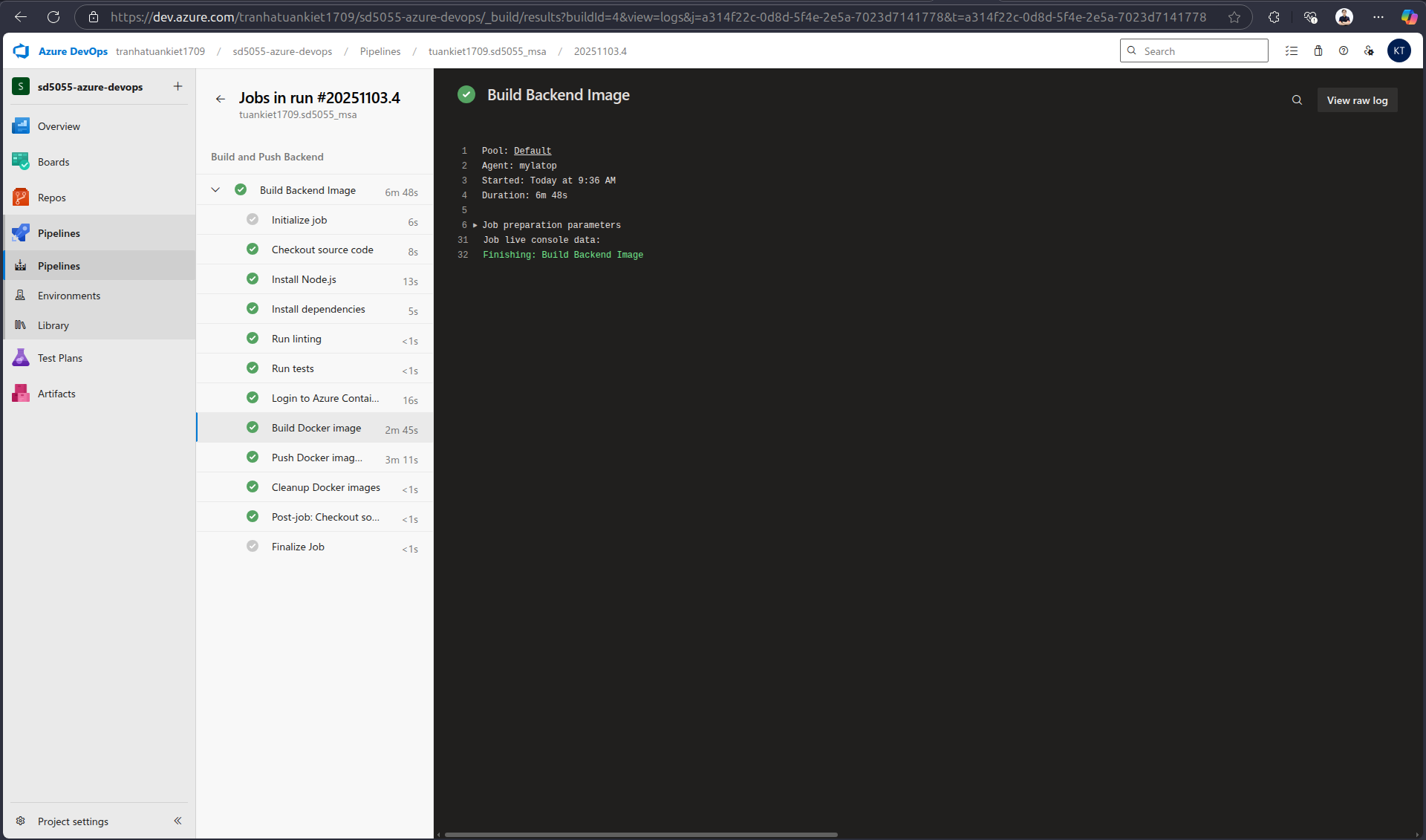


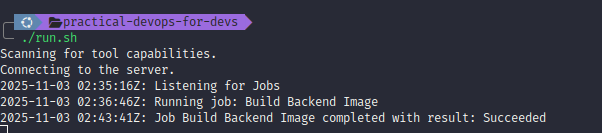




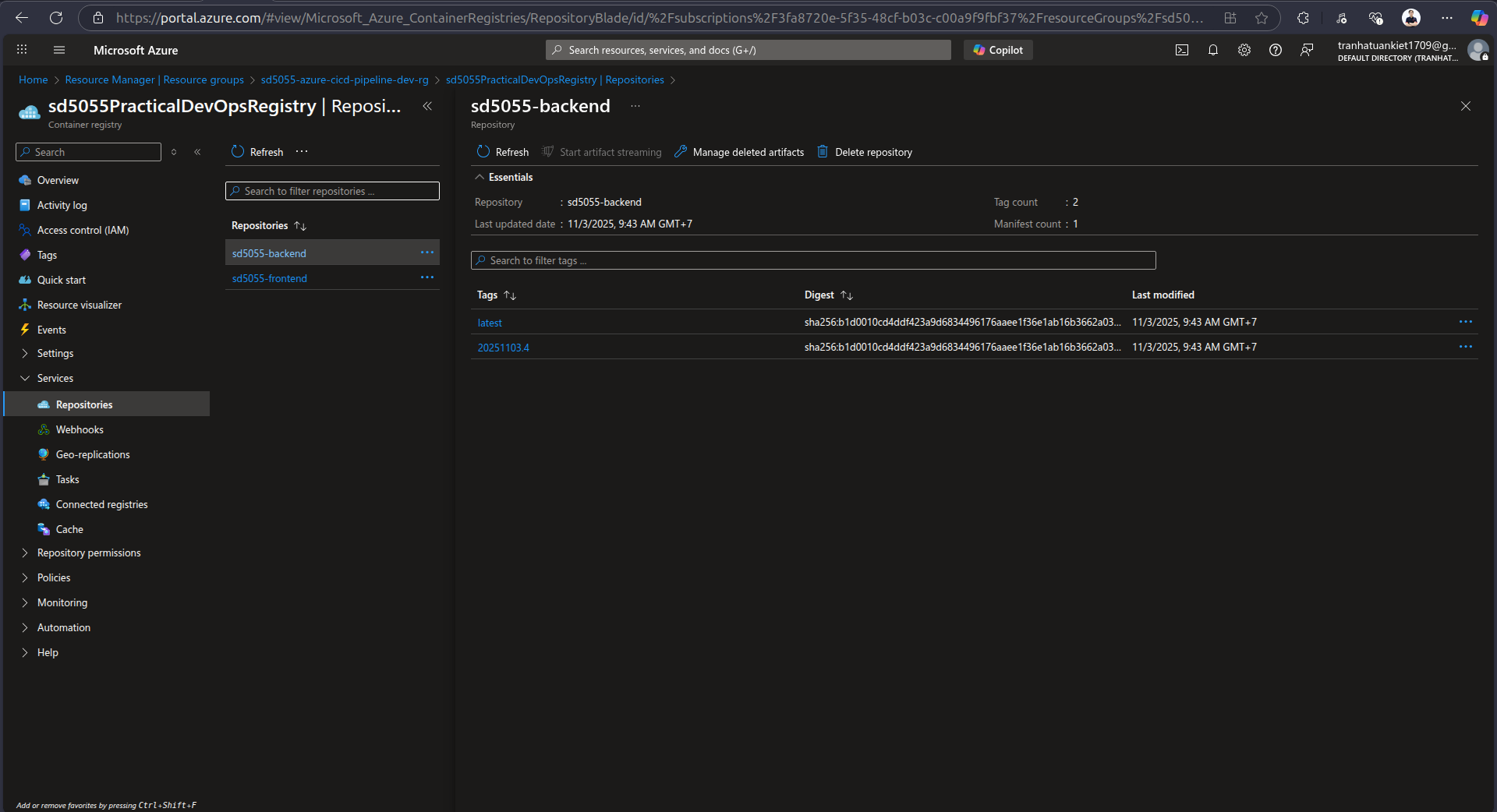




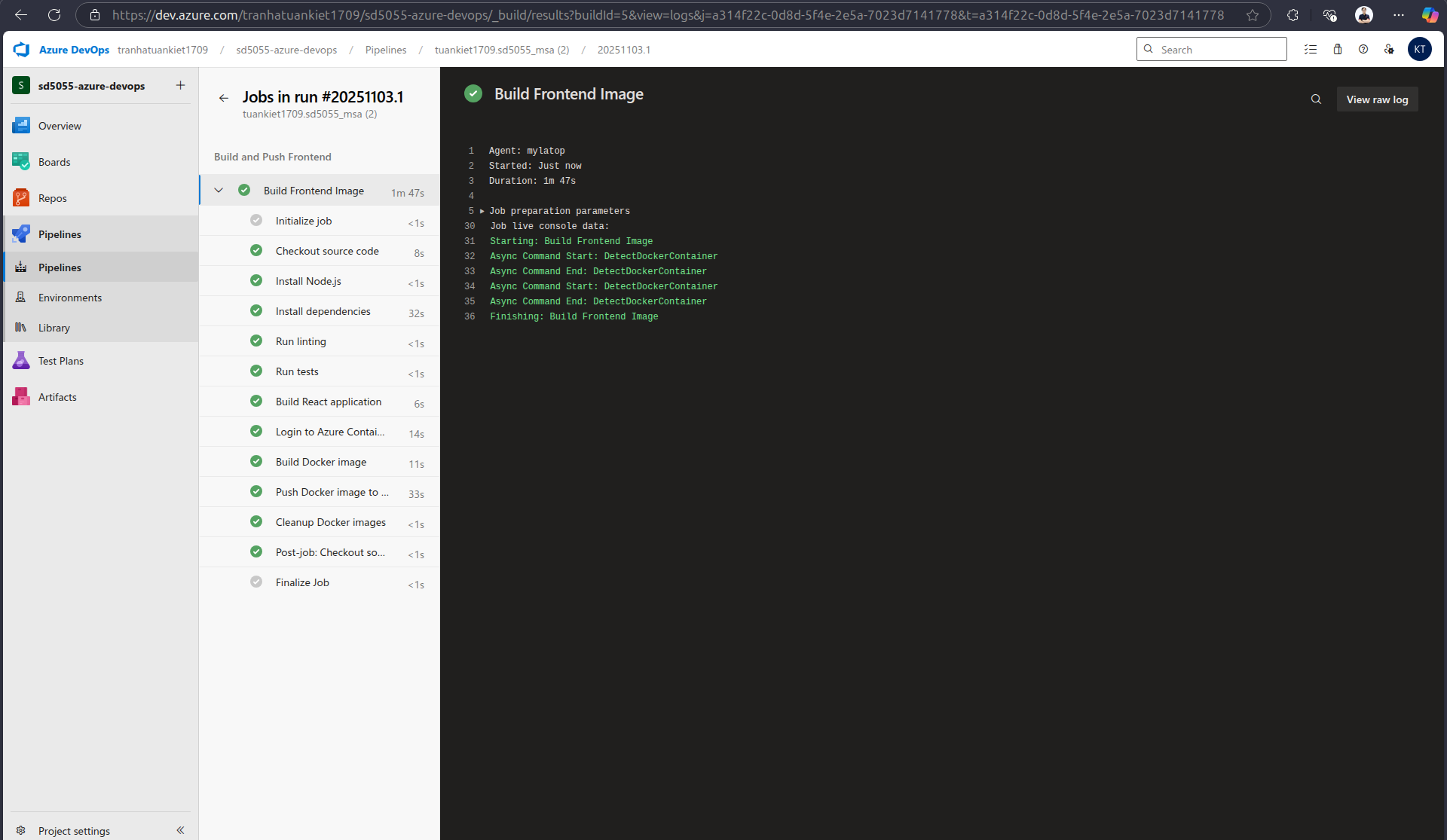


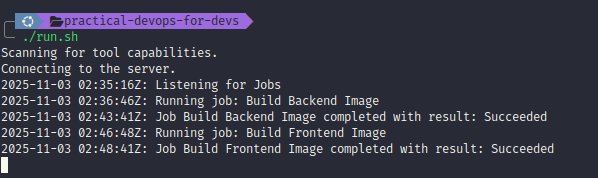


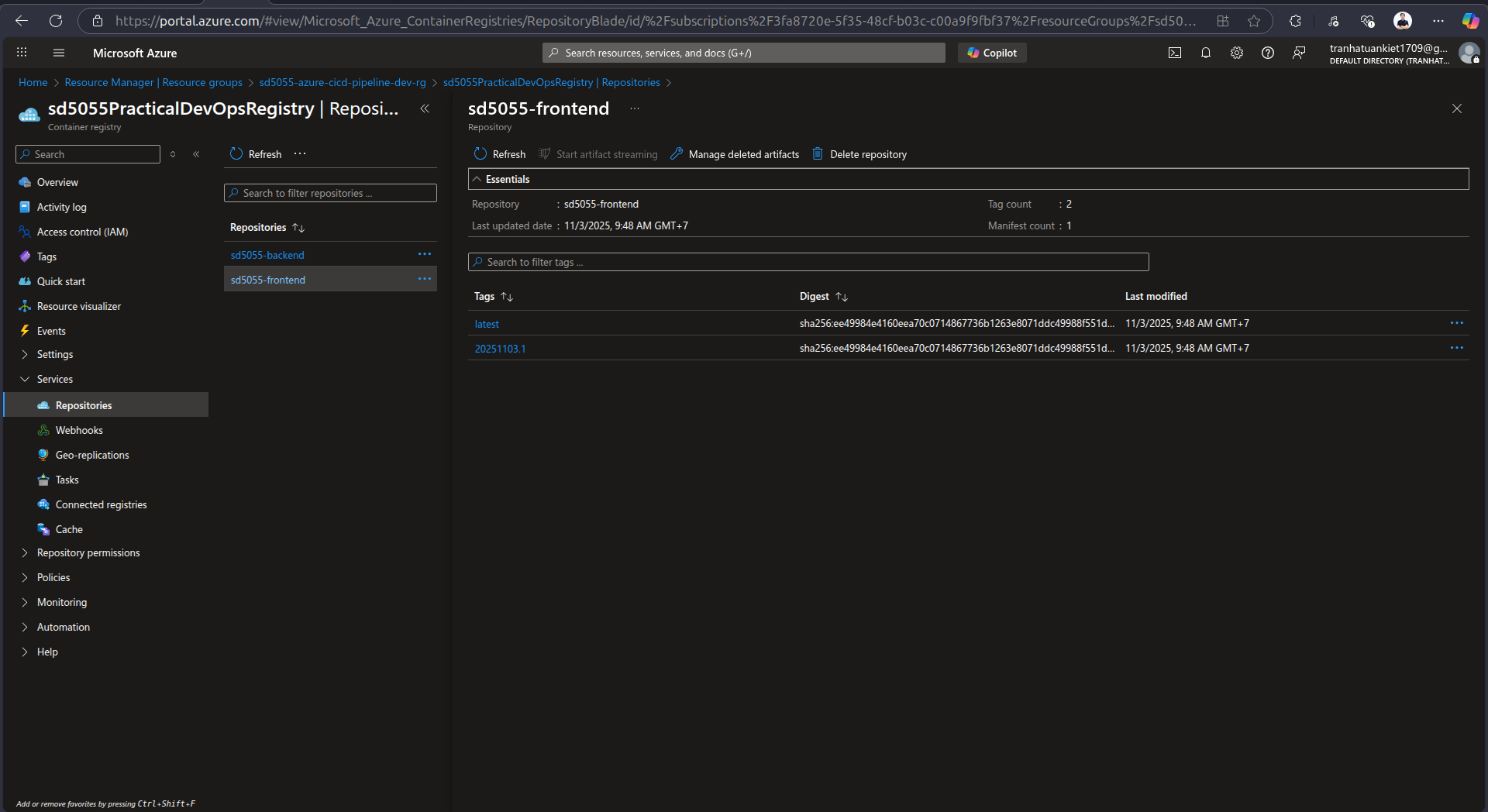
* After running CI successfully, the image is pushed to ACR



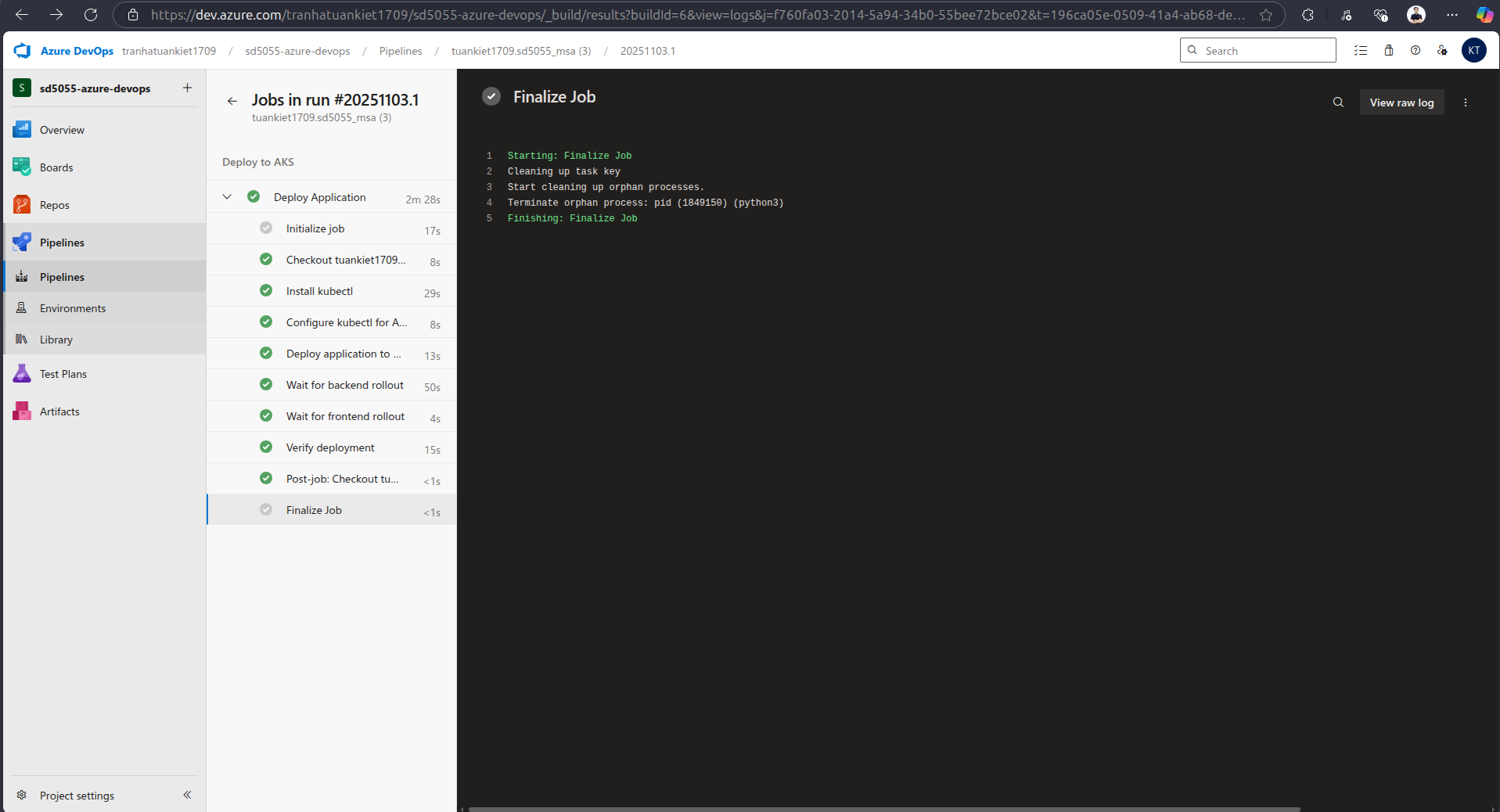
* 1. Frontend
  + Setup pipeline the same as backend pipelines
  + Once frontend pipeline ran successfully, the image is pushed to ACR

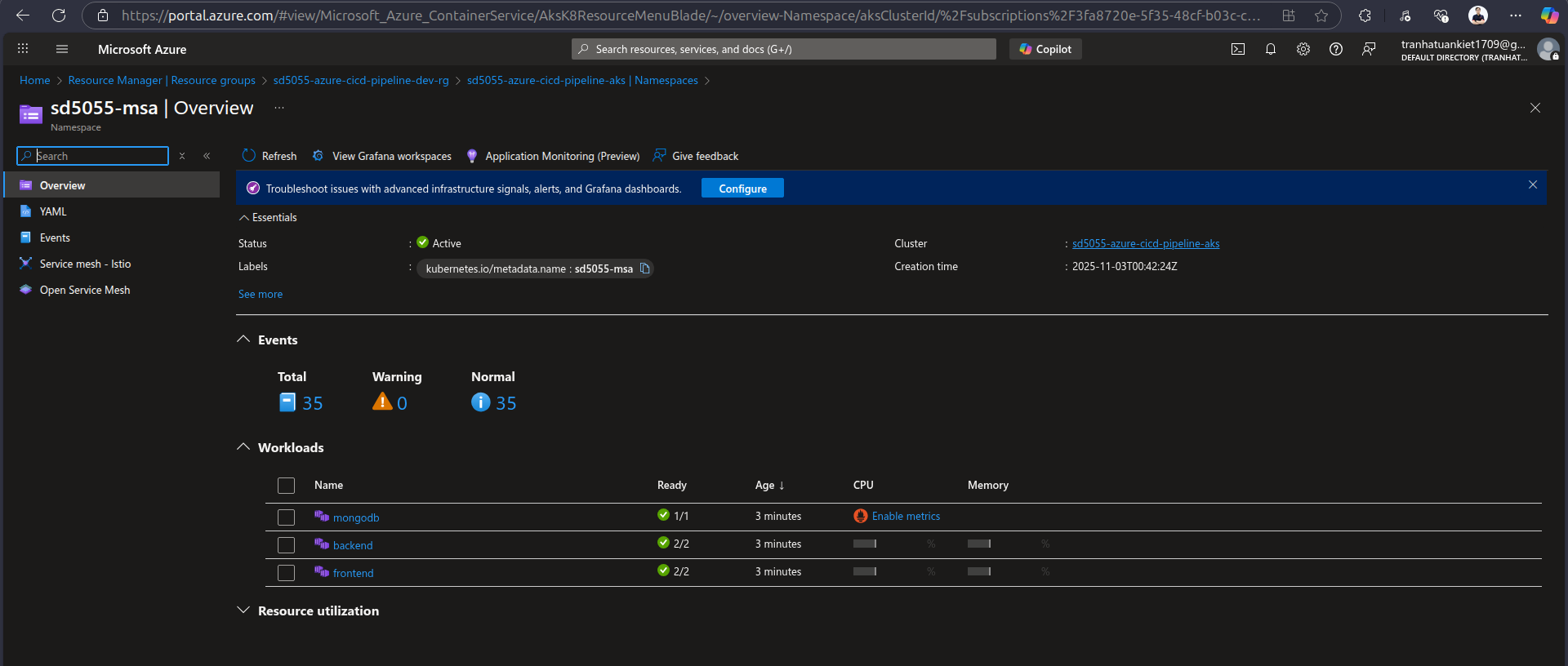
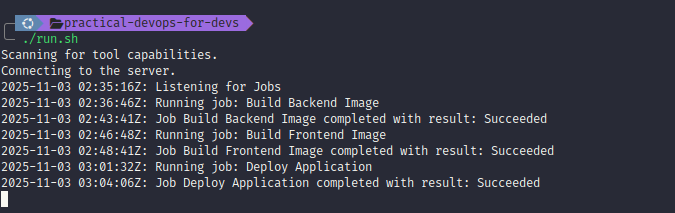




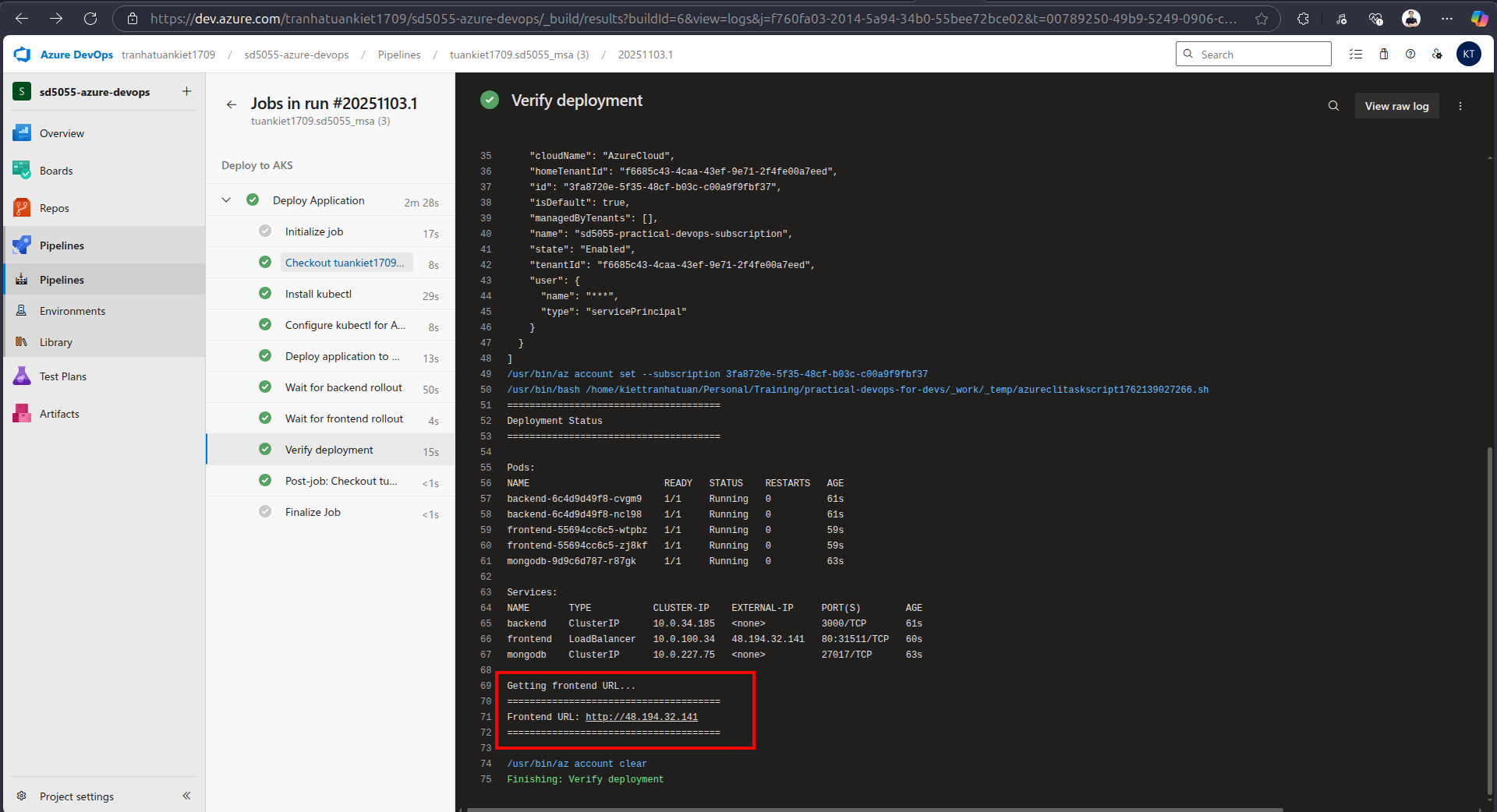


* 1. Run pipeline to deploy MSA to AKS
  + Please refer to source code: <https://github.com/tuankiet1709/sd5055_devops_cicd_deployment/tree/main/azure-deployment>
  + Setup the pipeline for deployment the same as with frontend/backend
  + After running deployment successfully, msa app has been deployed to aks





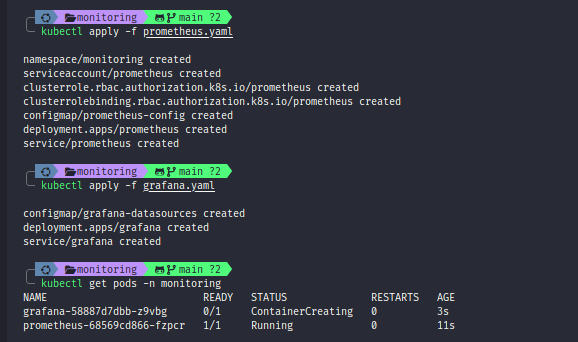
* + From output of deployment pipeline, we can get Frontend URL and able to access to the application



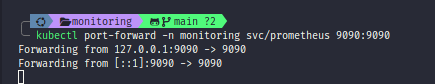


## Monitoring with Prometheus and Grafana

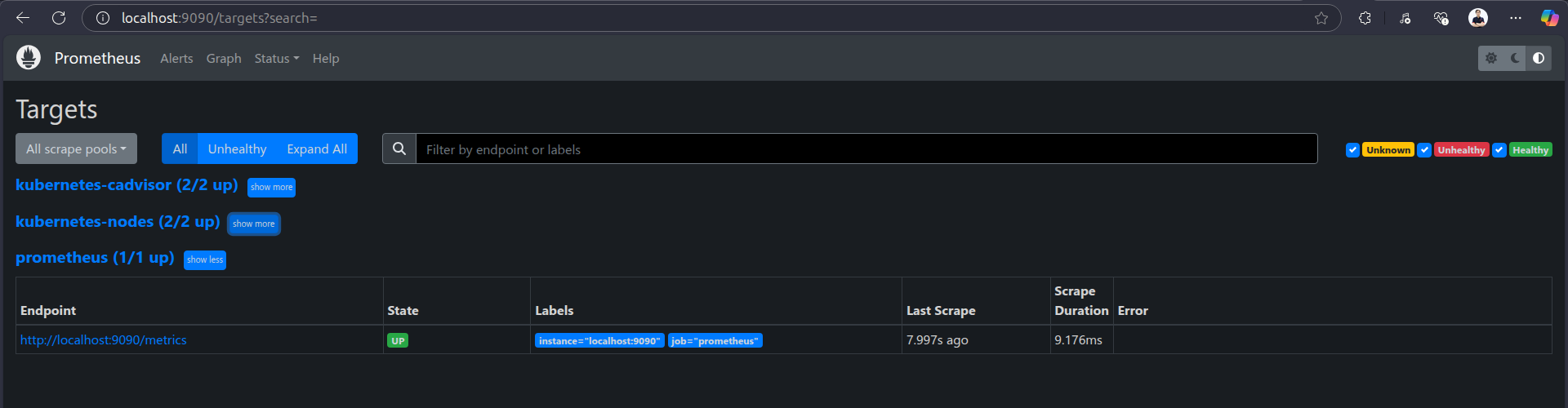
* Define yaml file to deploy Prometheus and Grafana in sd5055\_azure\_infrastructure: <https://github.com/tuankiet1709/sd5055_azure_infrastructure>
* Deploy grafana and prometheus to eks successfully

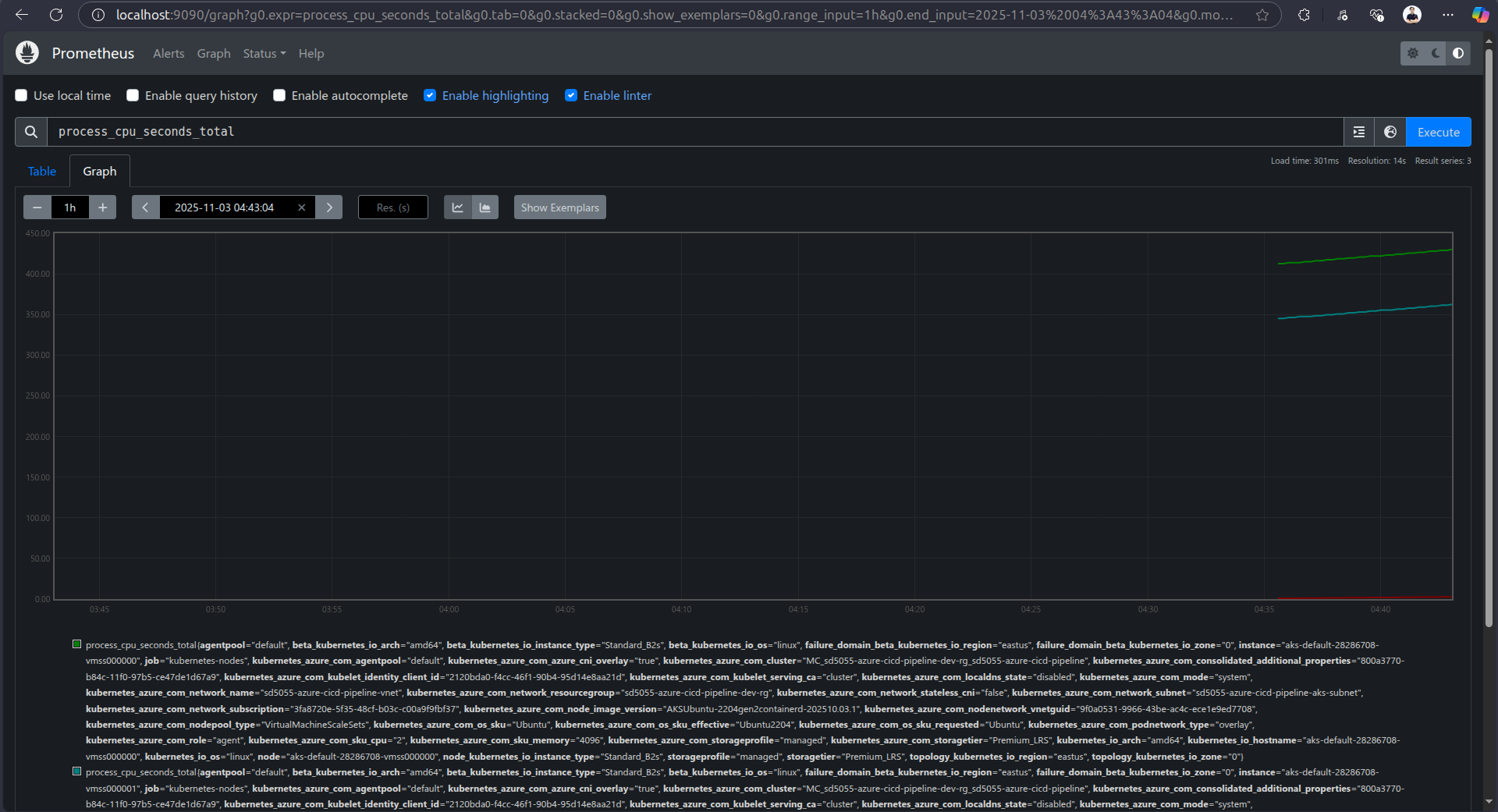


* Forward prometheus port from eks to local

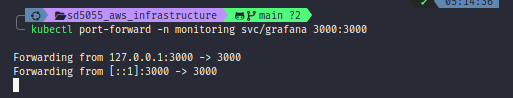


* We can monitoring with prometheus now

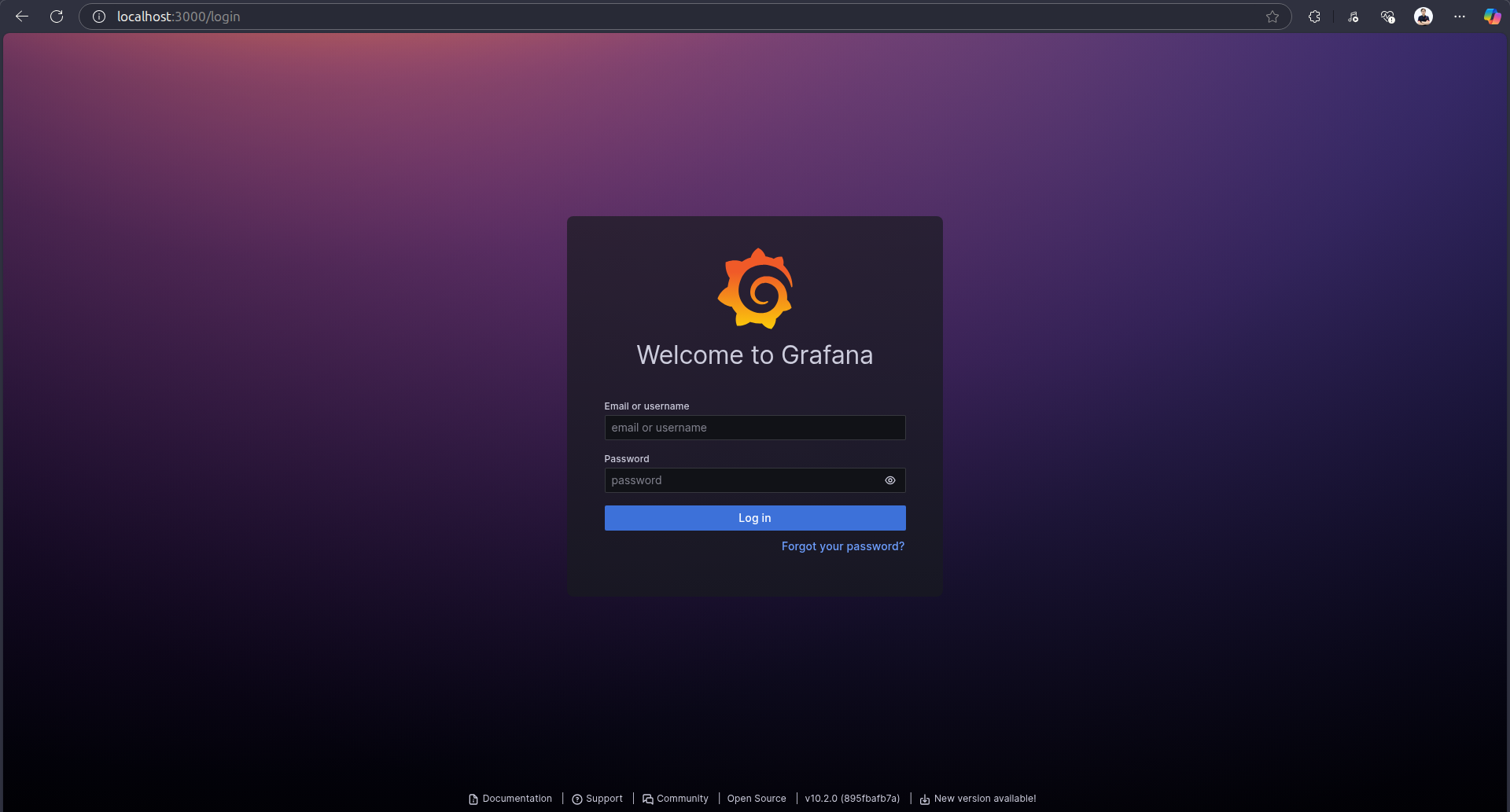




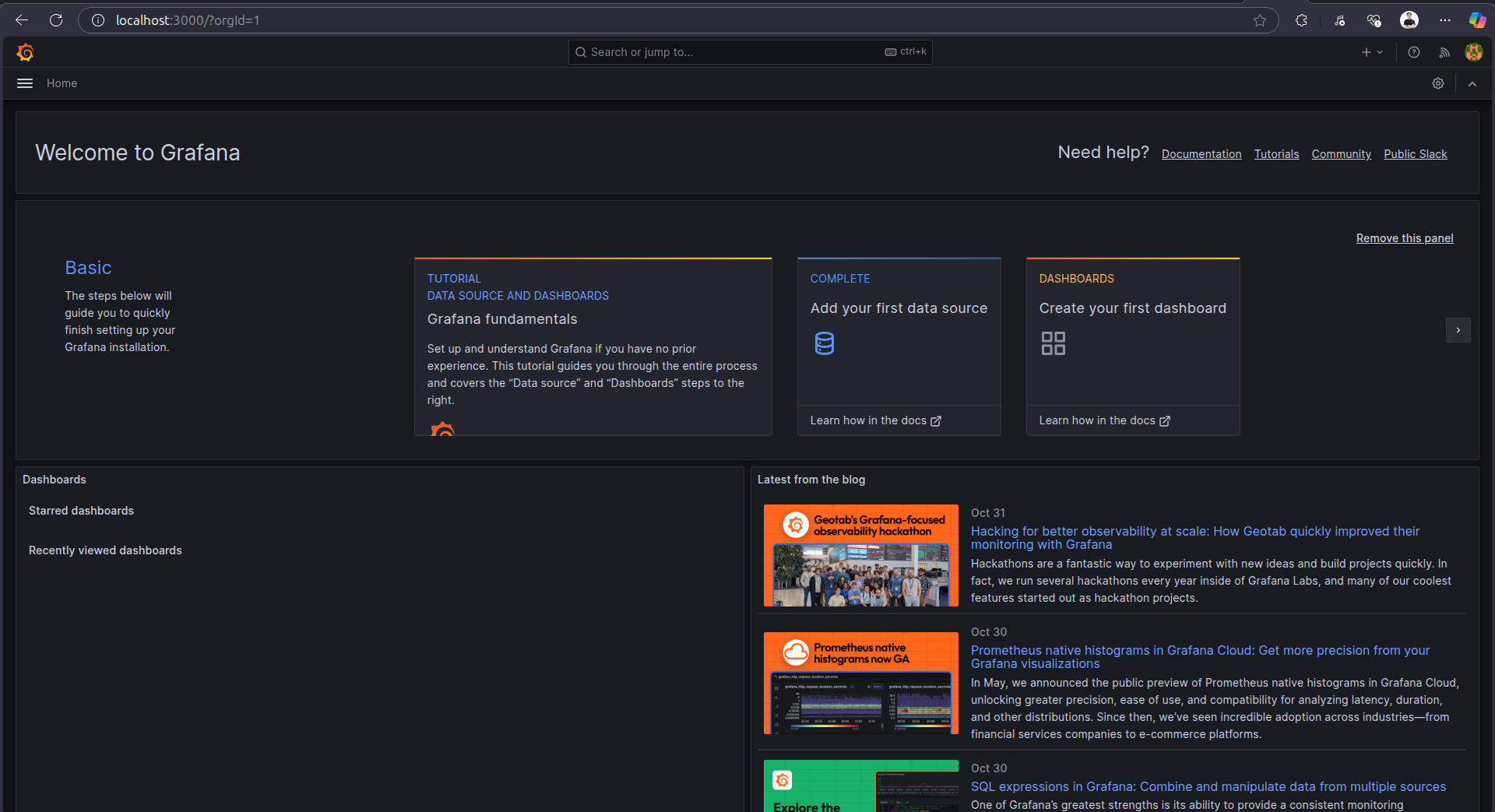
* Forward Grafana port from eks to local



* We can access to Grafana now



* Logged in the Grafana successfully



* We can monitor with Grafana now

