***SkillUp- Capstone Project***

***Problem statement:***

*Create Azure CI/CD Multistage Pipeline to build .NET application and deploy to DEV, PP and PROD environments. Here are the instructions to follow while creating the pipeline.*

*1. Create Azure DevOps project named "Skillup-Capstone-Project-<name>" where <name> is your name.*

*2. Pull the .NET code from git.*

*3. Build Azure infrastructure (AKS Cluster) using Bicep scripts for three environments DEV, PP and PROD.*

*4. Create Azure Board to create some tickets and assign to yourself. (1 ticket for each environment).*

*5. Build the code to deploy to DEV environment.*

*6. Configure the SONAR server and perform sonar testing before deploying to PP environment.*

*7. Configure the release gates before deploying to PROD environment.*

*8. Perform the autoscaling of AKS cluster in as per the increase or decrease in load.*

*Workaround:*

***Step 1: Create Azure DevOps Project***

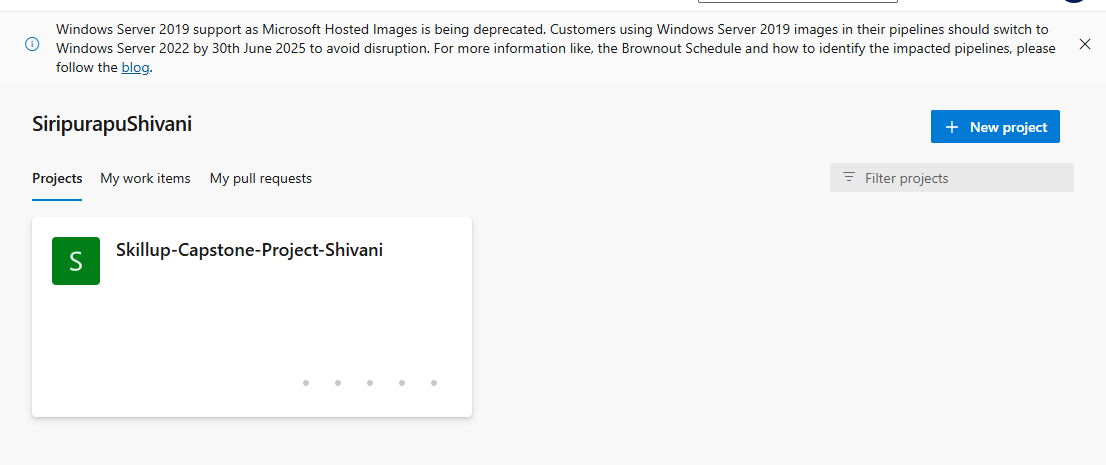
*Go to [Azure DevOps](https://dev.azure.com/).*

*Click on* ***New Project****.*

*Name the project:  
Skillup-Capstone-Project-Shivani*

*Set visibility to* ***Private****.*

*Click* ***Create****.*

**

*We now see Skillup-Capstone-Project-Shivani under SiripurapuShivani Organization.*

***Step 2: Pull .NET Code from Git***

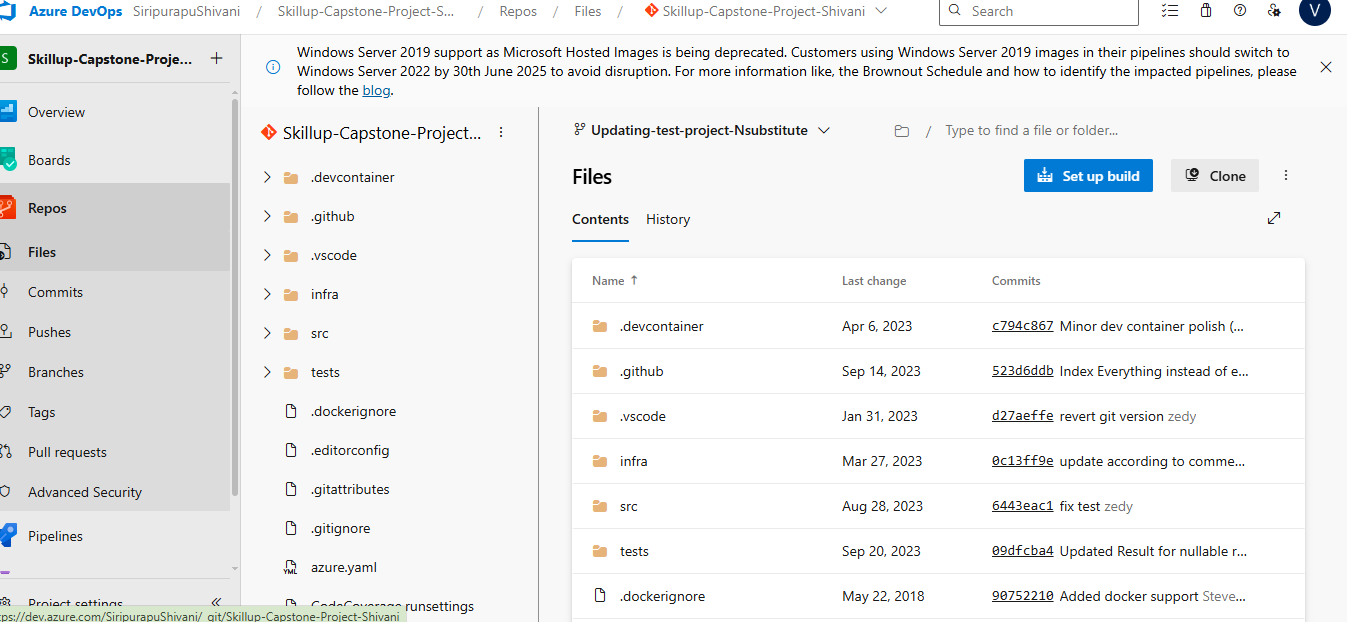
*Navigate to* ***Repos*** *→* ***Import Repository****.*

*Enter the Git URL of your .NET application.*

*https://github.com/dotnet-architecture/eShopOnWeb.git*

*Click* ***Import****.*

*Confirm the source code is pulled and structured correctly.*

**

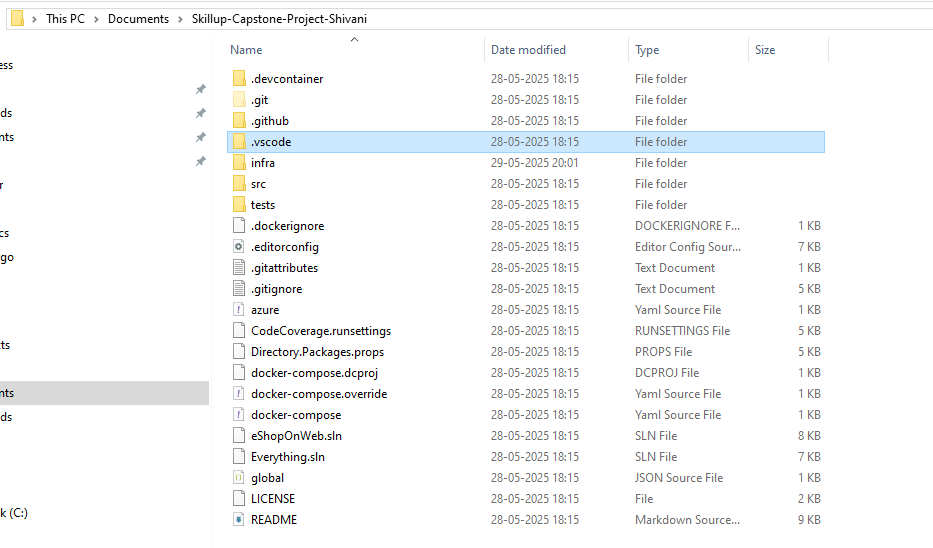
*We now see the source code successfully inside the Azure DevOps project.*

***Step3 : Clone into Your User Directory***

***Command:***

***git clone <https://SiripurapuShivani@dev.azure.com/SiripurapuShivani/Skillup-Capstone/_git/Skillup-Capstone-Project-Shivani>***

***C:\Users\L470\Documents\Skillup-Capstone-Project-Shivani***

******

***We now see the source code clone locally.***

***Scripts for Main.Bicep, Dev-parameters.json, PP-parameters.json and Prod-parameters.json are as below.***

***Main.Bicep***

***targetScope = 'resourceGroup'***

***param environmentName string***

***param location string***

***param adminUsername string***

***@secure()***

***param adminPassword string***

***param vmSize string = 'Standard\_D4a\_v4'***

***resource vnet 'Microsoft.Network/virtualNetworks@2021-05-01' = {***

***name: '${environmentName}-vnet'***

***location: location***

***properties: {***

***addressSpace: {***

***addressPrefixes: [***

***'10.0.0.0/16'***

***]***

***}***

***subnets: [***

***{***

***name: 'default'***

***properties: {***

***addressPrefix: '10.0.0.0/24'***

***}***

***}***

***]***

***}***

***}***

***resource nic 'Microsoft.Network/networkInterfaces@2021-05-01' = {***

***name: '${environmentName}-nic'***

***location: location***

***properties: {***

***ipConfigurations: [***

***{***

***name: 'ipconfig1'***

***properties: {***

***subnet: {***

***id: '${vnet.id}/subnets/default'***

***}***

***privateIPAllocationMethod: 'Dynamic'***

***}***

***}***

***]***

***}***

***}***

***resource vm 'Microsoft.Compute/virtualMachines@2021-07-01' = {***

***name: '${environmentName}-vm'***

***location: location***

***properties: {***

***hardwareProfile: {***

***vmSize: vmSize***

***}***

***osProfile: {***

***computerName: '${environmentName}-vm'***

***adminUsername: adminUsername***

***adminPassword: adminPassword***

***}***

***storageProfile: {***

***imageReference: {***

***publisher: 'MicrosoftWindowsServer'***

***offer: 'WindowsServer'***

***sku: '2019-Datacenter'***

***version: 'latest'***

***}***

***osDisk: {***

***createOption: 'FromImage'***

***}***

***}***

***networkProfile: {***

***networkInterfaces: [***

***{***

***id: nic.id***

***}***

***]***

***}***

***}***

***}***

***Dev-parameters.json***

*{*

*"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",*

*"contentVersion": "1.0.0.0",*

*"parameters": {*

*"environmentName": {*

*"value": "dev"*

*},*

*"location": {*

*"value": "northeurope"*

*},*

*"adminUsername": {*

*"value": "azureuser"*

*},*

*"adminPassword": {*

*"value": "P@ssw0rd1234!"*

*},*

*"vmSize": {*

*"value": "Standard\_B2s"*

*}*

*}*

*}*

***Pp-parameters.json***

*{*

*"adminUsername": {*

*"value": "azureuser"*

*},*

*"adminPassword": {*

*"value": "YourP@ssw0rd123"*

*},*

*"environmentName": {*

*"value": "pp"*

*},*

*"location": {*

*"value": "northeurope"*

*},*

*"vmSize": {*

*"value": "Standard\_B2s"*

*}*

*}*

***Prod-parameters.json***

*{*

*"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",*

*"contentVersion": "1.0.0.0",*

*"parameters": {*

*"environmentName": {*

*"value": "prod"*

*},*

*"location": {*

*"value": "westus2"*

*},*

*"adminUsername": {*

*"value": "azureuser"*

*},*

*"adminPassword": {*

*"value": "YourP@ssw0rd123"*

*},*

*"vmSize": {*

*"value": "Standard\_B1s"*

*}*

*}*

*}*

***Step 4: Create a Resource Group.***

*In the* ***Azure Portal****, search for* ***Resource Groups*** *in the top search bar.*

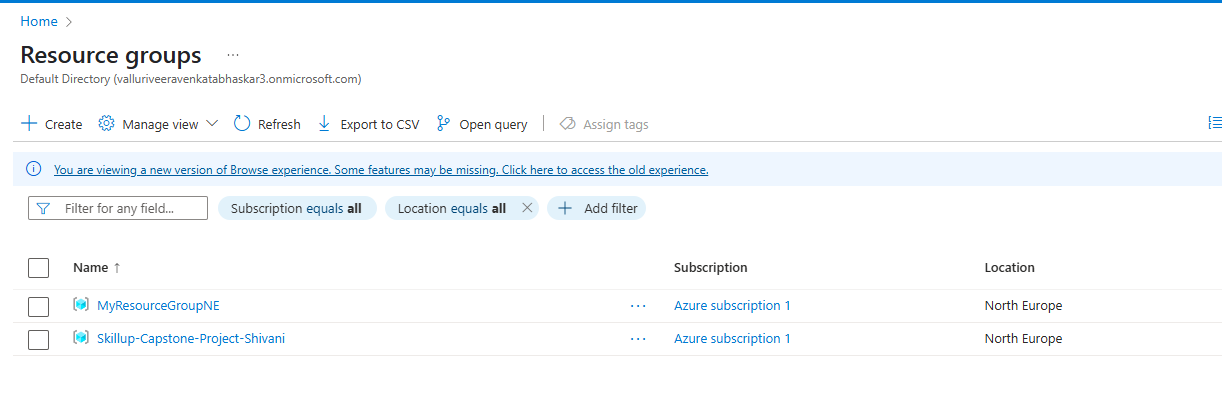
*Click* ***Create*** *and provide the following details:*

* + ***Subscription:*** *Select your active subscription.*

***Resource Group Name:*** *AZ204-Project1-RG*

* + ***Region:*** *West US*

*Click* ***Review + Create****, then* ***Create****.*

**

*We now see a Resource Group [Skillup-Capstone-Project-Shivani](https://portal.azure.com/) in North Europe location.*

***Step 5: Build Infrastructure Using Bicep***

1. *Make sure there is bicep template in your repo.*
2. *Make sure it has defined a parameterized Bicep file for AKS*
3. *Create environment-specific parameter files:*

*dev-parameters.json*

*pp-parameters.json*

*prod-parameters.json*

*4. Use Azure CLI or a deployment task in your pipeline to deploy:*

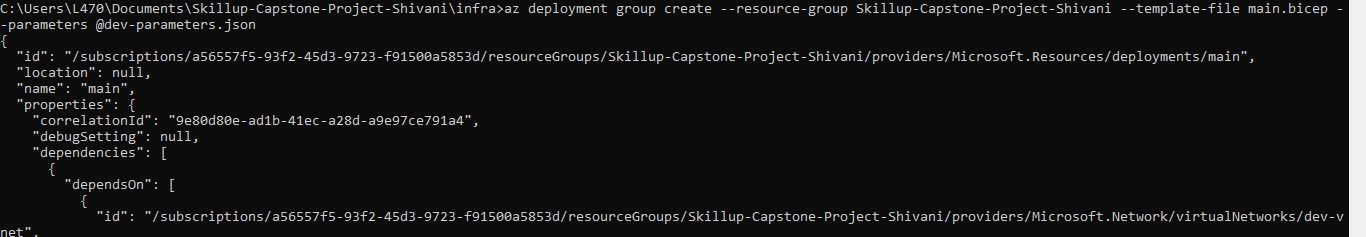
*az deployment sub create \*

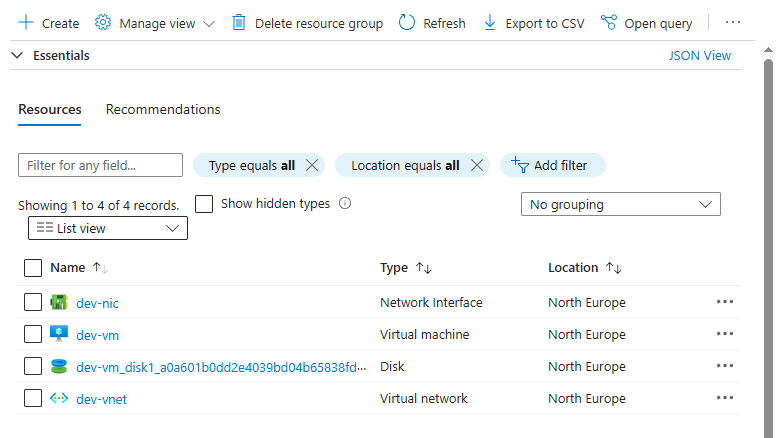
*--location eastus \*

*--template-file main.bicep \*

*--parameters @dev-parameters.json*

*Command ran successfully.*

**

**

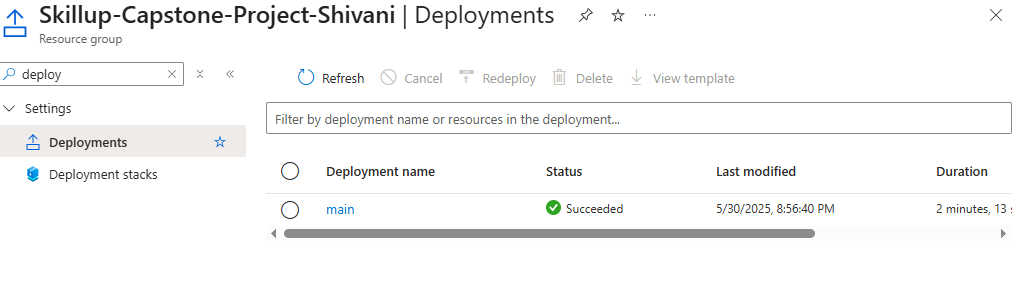
*We now see Dev resources successfully.*

*We have successfully deployed the following resources in the northeurope region:*

***Virtual Network****: dev-vnet*

***Network Interface****: dev-nic*

***Virtual Machine****: dev-vm (Size: Standard\_B2s, Admin: azureuser)*

**

*We now see that our deployment using main.bicep and dev-parameters.json to the resource group Skillup-Capstone-Project-Shivani has* ***succeeded****.*

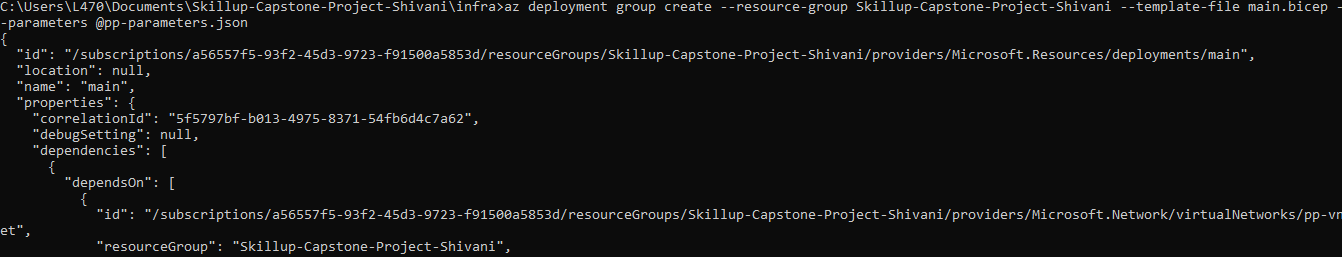
***Step 6 : Repeat the same process for PP and Prod environments.***

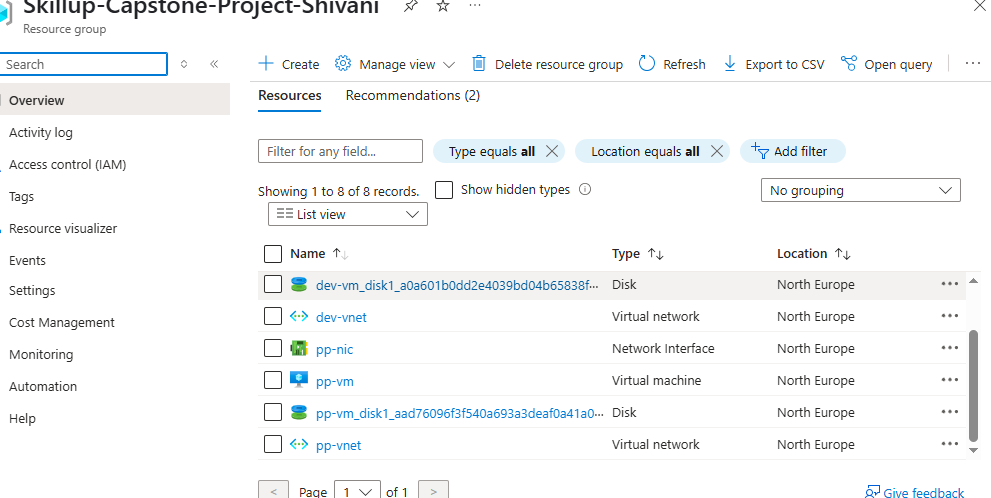
***Deploy to PP Environment:***

*Command:*

*az deployment group create --resource-group Skillup-Capstone-Project-Shivani --template-file main.bicep --parameters @pp-parameters.json*

*Command ran successfully.*

**

**

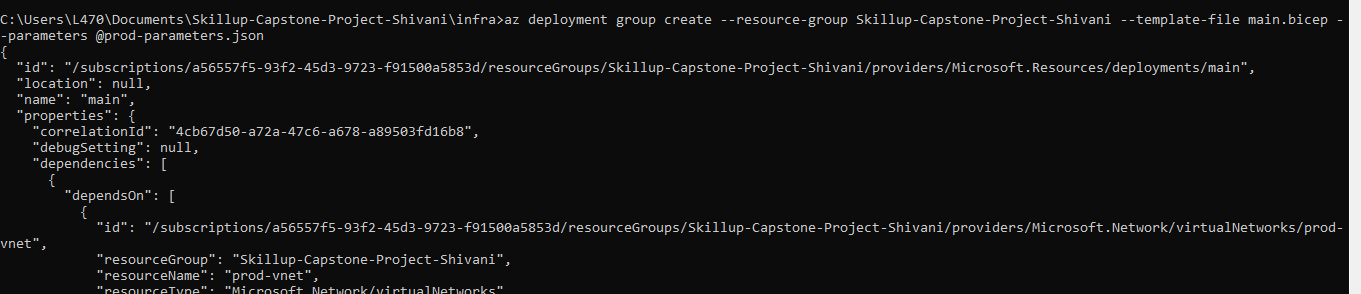
*We see PP resources.*

***Deploy to PROD Environment:***

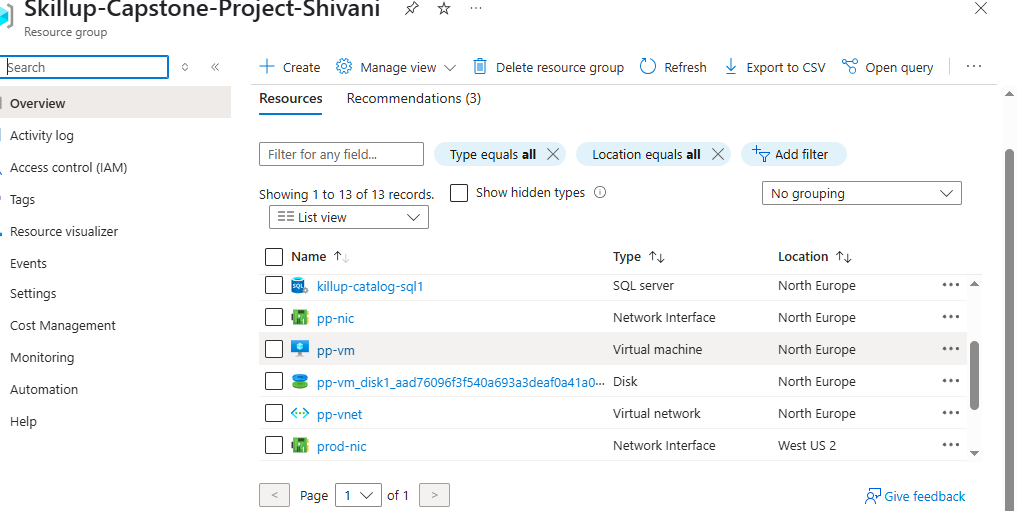
*Command:*

*az deployment group create --resource-group Skillup-Capstone-Project-Shivani --template-file main.bicep --parameters @prod-parameters.json*

*Command ran successfully.*

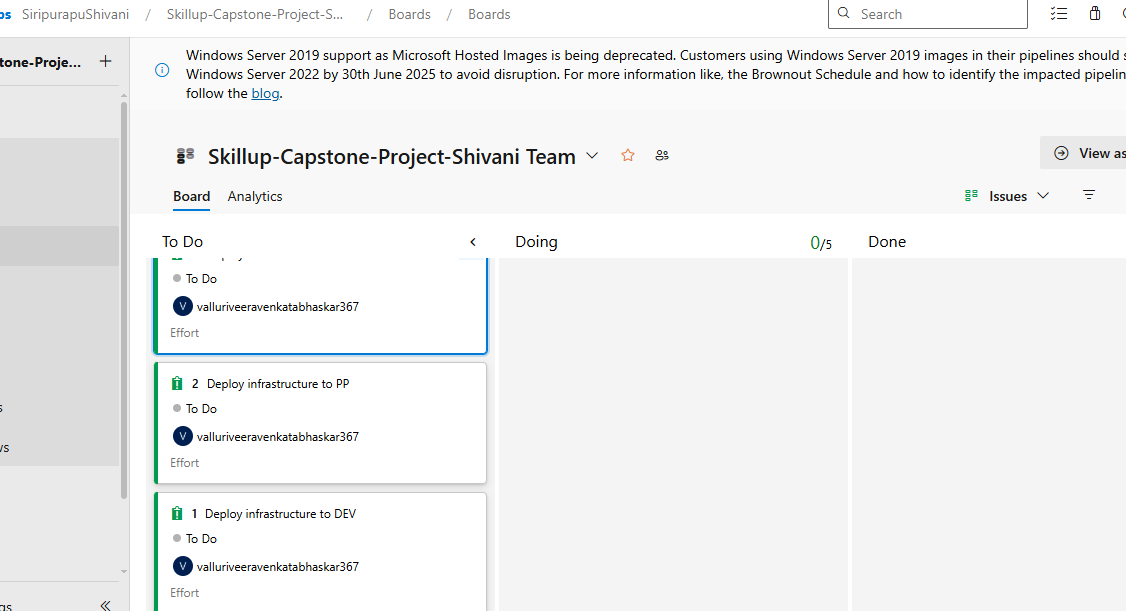
**

*We see PP resources.*

**

***Step 7 :Create Azure Boards Tickets.***

* *Navigate to* ***Boards*** *→* ***Work Items****.*
* *Create three tasks:*
  + *"Deploy infrastructure to DEV"*
  + *"Deploy infrastructure to PP"*
  + *"Deploy infrastructure to PROD"*
* *Assign all tasks to yourself*

******

***We now see the tickets created foe three environments and assigned to self( This Azure account is created under other’s name. Hence, it is reflecting the account creator’s name.)***

***Step 8 :Build & Deploy to DEV***

*Go to* ***Pipelines*** *→* ***Create Pipeline****.*

*Select your repository.*

*Use the classic editor or YAML.*

*I chose ASP.NET as we are focusing on .NET application.*

*YAML pipeline script used as below:*

*azure-pipelines.yml*

*trigger:*

*- main*

*pool:*

*vmImage: 'windows-latest'*

*steps:*

*- task: UseDotNet@2*

*inputs:*

*packageType: 'sdk'*

*version: '8.0.x'*

*installationPath: $(Agent.ToolsDirectory)/dotnet*

*- task: CmdLine@2*

*inputs:*

*script: dotnet build src/Web/Web.csproj*

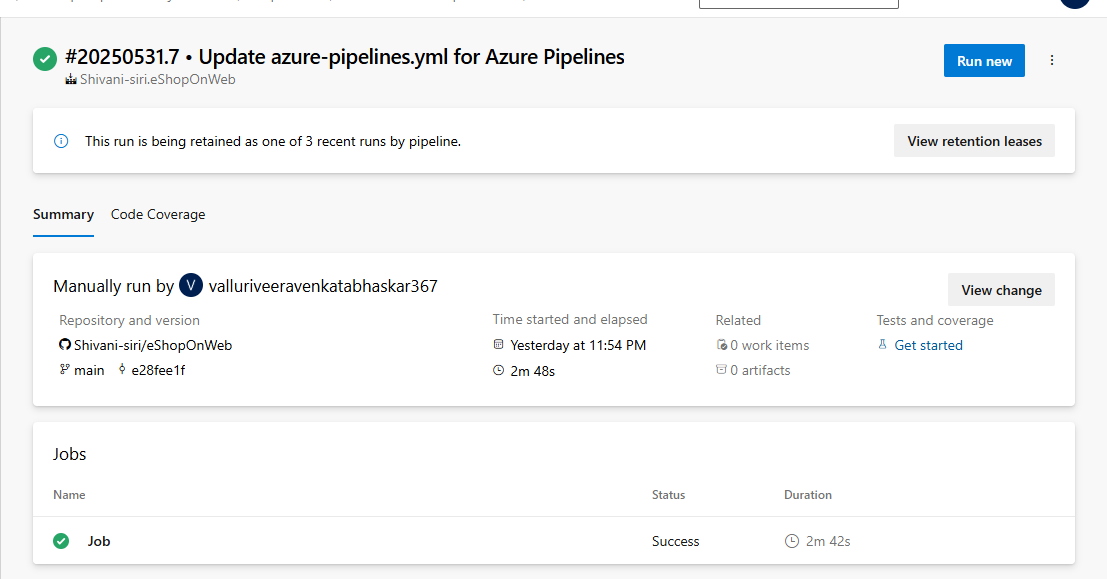
*displayName: 'Build Web.csproj'*

*- task: CmdLine@2*

*inputs:*

*script: dotnet build src/Web/Web.csproj*

*displayName: 'Build Web.csproj'*

**

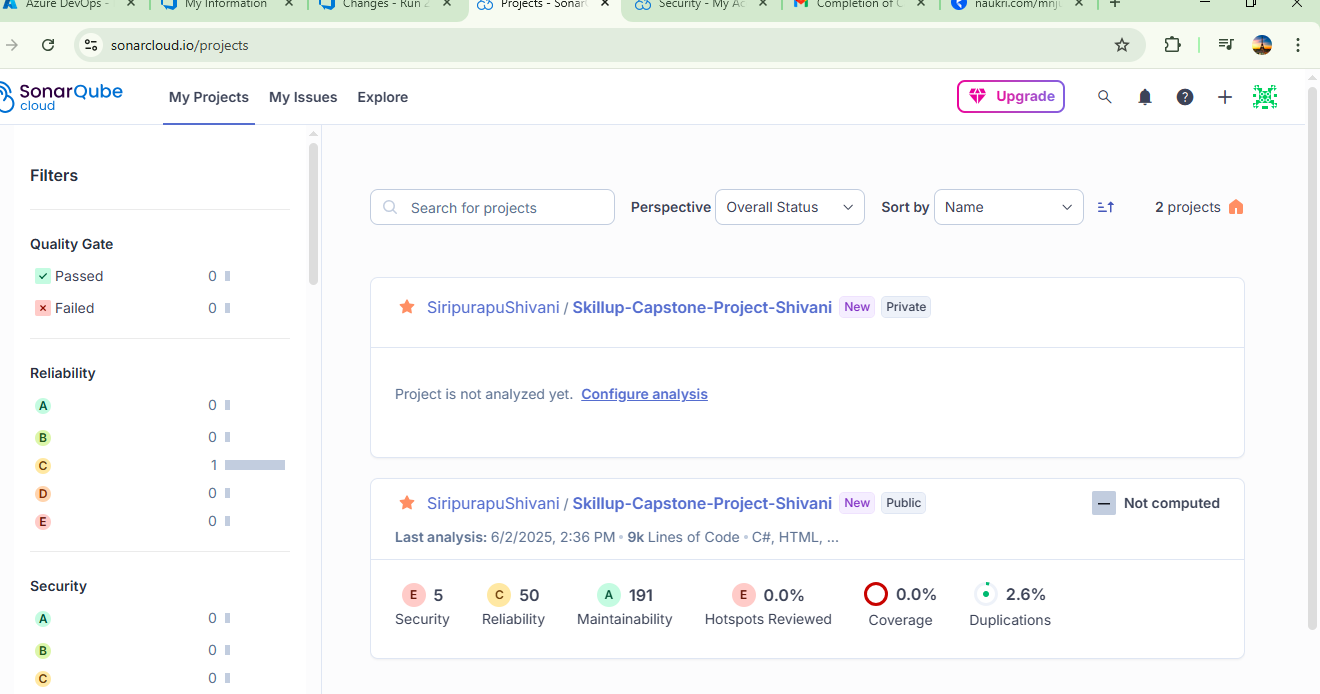
*We now see that pipeline is successfully built.*

***Step 9: Configure SonarQube Before PP Deployment***

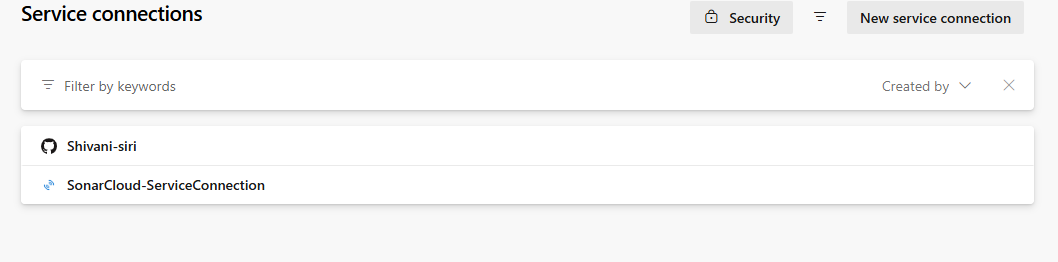
*Create a SonarQube project on your SonarQube server.*

*Add SONAR\_TOKEN to Azure DevOps Pipelines → Library → Variables.*

*Add the Sonar tasks in your YAML before PP deployment:*

**

*We see that our Capstone Project is successfully configured on the SonarQube server.*

**

*We have also added Sonar token to our project.*

*Now, we have added Sonar tasks to the YAML as below:*

*# ASP.NET*

*# Build and test ASP.NET projects.*

*# Add steps that publish symbols, save build artifacts, deploy, and more:*

*# https://docs.microsoft.com/azure/devops/pipelines/apps/aspnet/build-aspnet-4*

*# ASP.NET*

*trigger:*

*- main*

*pool:*

*vmImage: 'windows-latest'*

*variables:*

*buildConfiguration: 'Release'*

*stages:*

*# 🔍 Step 1: SonarCloud Code Analysis*

*- stage: CodeAnalysis*

*displayName: 'SonarCloud Analysis'*

*jobs:*

*- job: SonarScan*

*displayName: 'Run SonarCloud Scan'*

*steps:*

*- task: UseDotNet@2*

*displayName: 'Install .NET 8.0 SDK'*

*inputs:*

*packageType: sdk*

*version: '8.0.x'*

*- task: SonarCloudPrepare@1*

*inputs:*

*SonarCloud: 'SonarCloud-ServiceConnection'*

*organization: 'siripurapushivani'*

*scannerMode: 'MSBuild'*

*projectKey: 'Skillup-Capstone-Project-Shivani'*

*projectName: 'Skillup-Capstone-Project-Shivani'*

*- task: DotNetCoreCLI@2*

*displayName: 'Build Solution'*

*inputs:*

*command: 'build'*

*projects: '\*\*/\*.sln'*

*arguments: '--configuration $(buildConfiguration)'*

*- task: SonarCloudAnalyze@1*

*displayName: 'Run SonarCloud Analysis'*

*# 🚀 Step 2: Deploy to Pre-Production*

*- stage: DeployPP*

*displayName: 'Deploy to Pre-Production'*

*dependsOn: CodeAnalysis*

*condition: succeeded()*

*jobs:*

*- job: DeployPP*

*displayName: 'Deploy to PP'*

*steps:*

*- script: echo "Deploying to Pre-Production..."*

*displayName: 'Stub: Deployment Script Here'*

*# 🚀 Step 3: Deploy to Production*

*- stage: DeployProd*

*displayName: 'Deploy to Production'*

*dependsOn: DeployPP*

*condition: succeeded()*

*jobs:*

*- job: DeployProd*

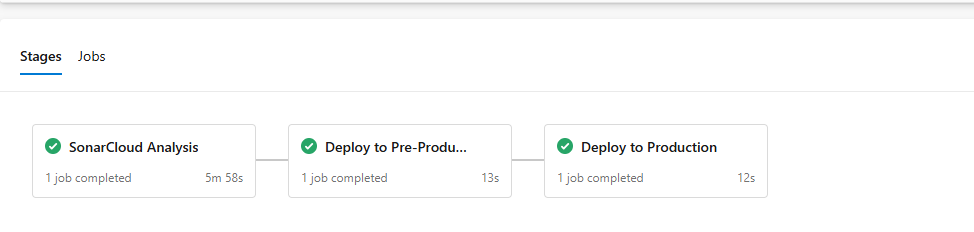
*displayName: 'Deploy to PROD'*

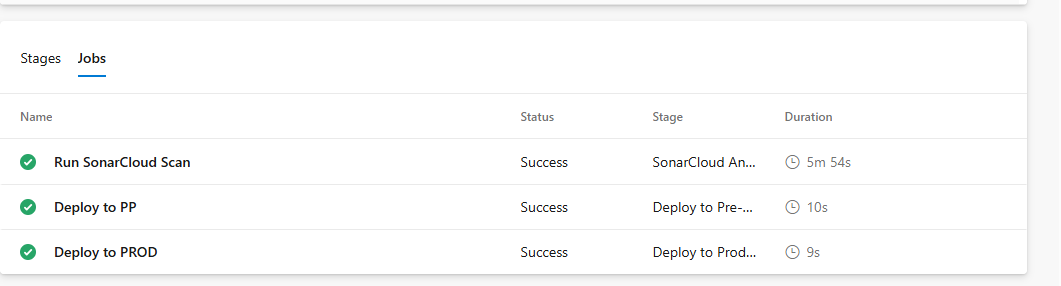
*steps:*

*- script: echo "Deploying to Production..."*

*displayName: 'Stub: Deployment Script Here'*

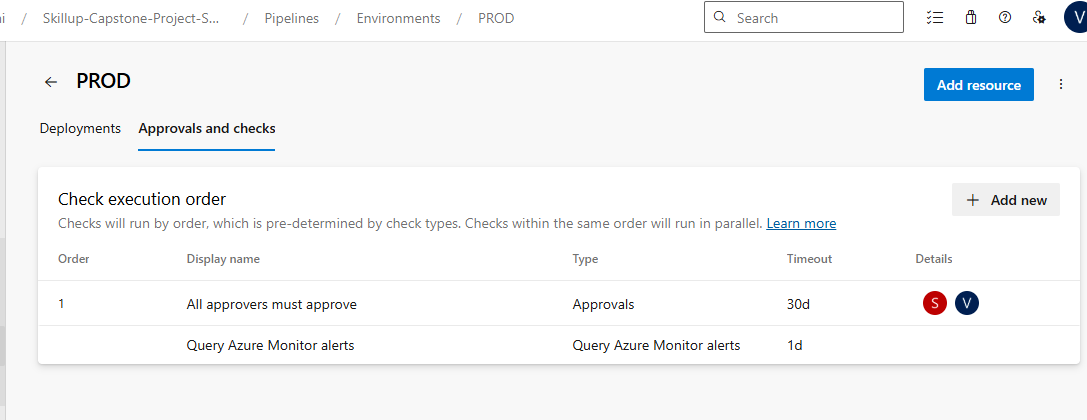
*After performing sonar tests, we see that jobs are successful as shown in the below images.*

**

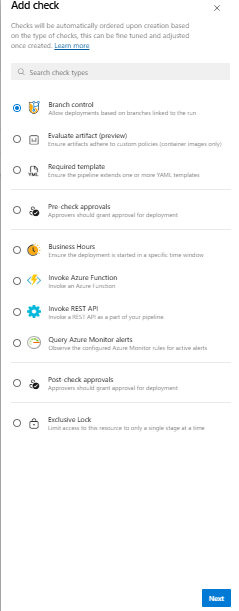
**

***Step 10: Configure Release Gates Before PROD***

1. *In Azure Pipelines → Environments → Create environment PROD.*
2. *Add* ***Approvals and Checks****:*
   * *Manual approval*
   * *Azure Monitor alert*
   * *Query work item or SonarQube Quality Gate*

**

***Note: I have added Manual approval and Azure Monitor Alerts. However, to add Query Work Item or SonarQube Gate, we need to use an Agile, Scrum, or CMMI process.I am still on Basic, so this option won't appear. Hence, skipping this part.***

******

1. *Update your azure-pipelines.yaml to include:*

*# ASP.NET Pipeline with SonarCloud and Deployment Stages*

*trigger:*

*- main*

*pool:*

*vmImage: 'windows-latest'*

*variables:*

*buildConfiguration: 'Release'*

*stages:*

*# 🔍 Step 1: SonarCloud Code Analysis*

*- stage: CodeAnalysis*

*displayName: 'SonarCloud Analysis'*

*jobs:*

*- job: SonarScan*

*displayName: 'Run SonarCloud Scan'*

*steps:*

*- task: UseDotNet@2*

*displayName: 'Install .NET 8.0 SDK'*

*inputs:*

*packageType: sdk*

*version: '8.0.x'*

*- task: SonarCloudPrepare@1*

*inputs:*

*SonarCloud: 'SonarCloud-ServiceConnection'*

*organization: 'siripurapushivani'*

*scannerMode: 'MSBuild'*

*projectKey: 'Skillup-Capstone-Project-Shivani'*

*projectName: 'Skillup-Capstone-Project-Shivani'*

*- task: DotNetCoreCLI@2*

*displayName: 'Build Solution'*

*inputs:*

*command: 'build'*

*projects: '\*\*/\*.sln'*

*arguments: '--configuration $(buildConfiguration)'*

*- task: SonarCloudAnalyze@1*

*displayName: 'Run SonarCloud Analysis'*

*# 🚀 Step 2: Deploy to Pre-Production*

*- stage: DeployPP*

*displayName: 'Deploy to Pre-Production'*

*dependsOn: CodeAnalysis*

*condition: succeeded()*

*jobs:*

*- job: DeployPP*

*displayName: 'Deploy to PP'*

*steps:*

*- script: echo "Deploying to Pre-Production..."*

*displayName: 'Stub: Deployment Script Here'*

*# 🚀 Step 3: Deploy to Production with Environment and Release Gates*

*- stage: Deploy\_PROD*

*displayName: 'Deploy to Production'*

*dependsOn: DeployPP*

*condition: succeeded()*

*jobs:*

*- deployment: DeployProd*

*displayName: 'Deploy to PROD Environment'*

*environment: 'PROD' # This ties to the environment you created in Azure DevOps*

*strategy:*

*runOnce:*

*deploy:*

*steps:*

*- task: UseDotNet@2*

*displayName: 'Install .NET SDK'*

*inputs:*

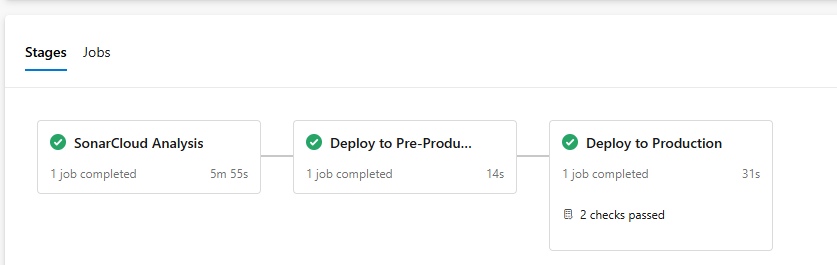
*packageType: 'sdk'*

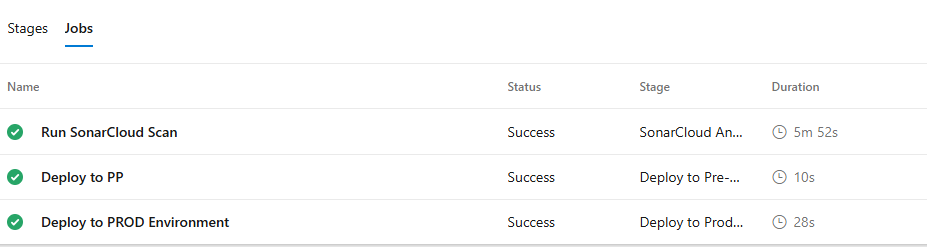
*version: '8.0.x'*

*- script: echo "Deploying to Production..."*

*displayName: 'Stub: Replace with your deployment script'*

*When you deploy, the* ***gates will pause*** *the deployment to PROD until they are satisfied.*

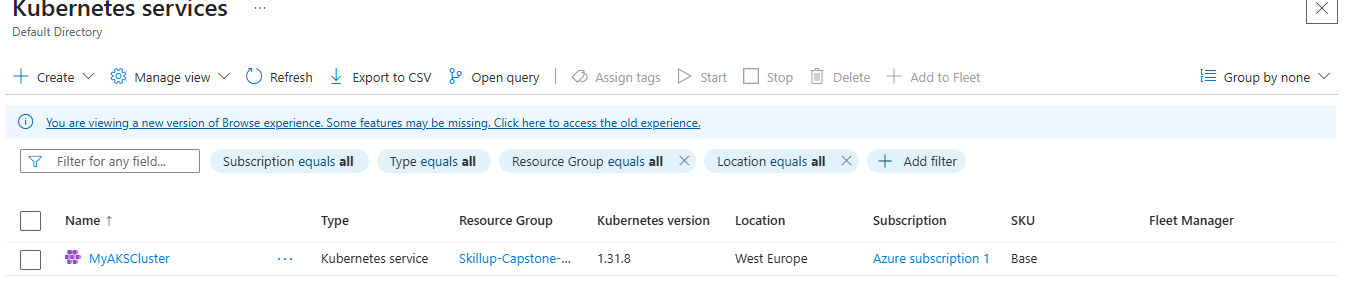
**

**

*We see that stages are completed and jobs are successful after updating yaml.*

***Step 11: Configure AKS Autoscaling.***

*I already have an AKS, hence I am updating it using below script.*

**

*az aks update \*

*--resource-group <your-resource-group> \*

*--name <your-cluster-name> \*

*--enable-cluster-autoscaler \*

*--min-count 1 \*

*--max-count 3*

***Deploy metrics server***

*Command:*

*kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml*

*Verification:*

*Command:*

*kubectl apply -f <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>*

***Create a deployment and expose it .***

*nano nginx-deployment.yaml*

*Below used is the YAML script.*

*# nginx-deployment.yaml*

*apiVersion: apps/v1*

*kind: Deployment*

*metadata:*

*name: nginx-deployment*

*spec:*

*replicas: 1*

*selector:*

*matchLabels:*

*app: nginx*

*template:*

*metadata:*

*labels:*

*app: nginx*

*spec:*

*containers:*

*- name: nginx*

*image: nginx*

*resources:*

*requests:*

*cpu: 100m*

*limits:*

*cpu: 200m*

*ports:*

*- containerPort: 80*

*Save and exit (CTRL+O, Enter, then CTRL+X).*

***Apply the above script.***

*kubectl apply -f nginx-deployment.yaml*

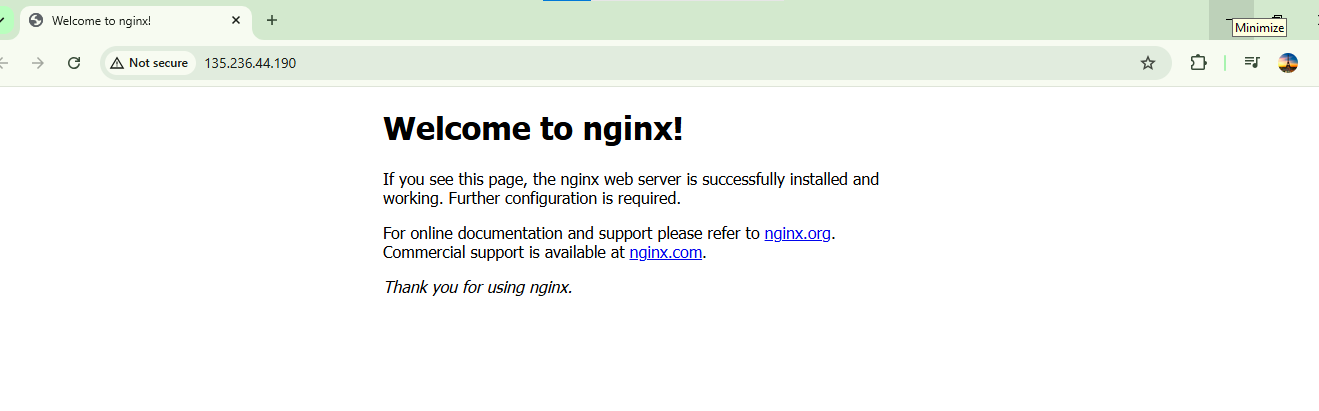
***Expose the deployment***

*Command*

*kubectl expose deployment nginx-deployment --port=80 --type=LoadBalancer*

*nginx-deployment LoadBalancer 10.0.15.18 135.236.44.190 80:31820/TCP 56s*

*Our* ***Nginx service is now live*** *at: <http://135.236.44.190>*

**

*kubectl expose deployment nginx-deployment --port=80 --type=LoadBalancer*

***Create a Horizontal Pod Autoscaler (HPA)***

*You can auto-scale pods based on CPU usage.*

*Command:*

*kubectl autoscale deployment nginx-deployment \*

*--cpu-percent=50 \*

*--min=1 \*

*--max=5*

*Verify*

*Command: kubectl get hpa*

***Simulate Load to Test Autoscaling***

*Use a load generator like kubectl run*

*Command*

*kubectl run -i --tty load-generator --image=busybox /bin/sh*

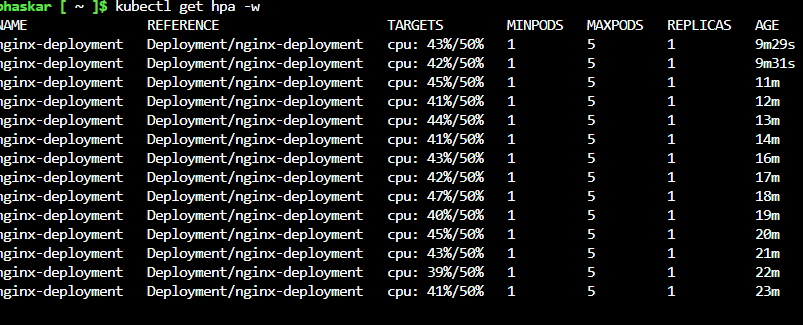
*Inside the shell, run*

*while true; do wget -q -O- http://nginx-deployment; done*

*Open a new terminal tab and monitor the HPA:*

*Command*

*kubectl get hpa -w*

**

*We are able to see the CPU usage go up, and the number of pods (REPLICAS) increase.*

*Shivani S*

*9502735637*