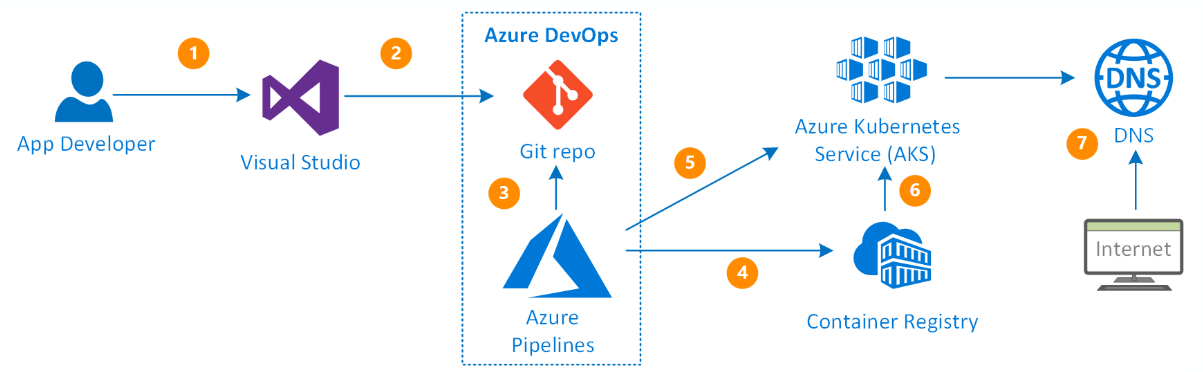
**AZURE DevOps challenge**

**Problem Statement**:  Create a sample application using any technology along with microservices to demonstrate implementation of the DevOps principle using Azure services. Establish a pipeline for continuous integration, continuous testing, and continuous deployment.

**Application** : A very basic ASP.NET Web application which will show “Welcome to Ineuron DevOps Challenge” on web page.



**CI/CD Flow:**

****

**Benefits of using Azure AKS solutions:**

* · Hosts your Kubernetes environment
* · Easy integration with Azure services such as Load balancing, Azure Blob Storage, Azure Active Directory, Application Gateway, Azure Traffic Manager etc.
* · Quick and easy to deploy
* · Hosted control plane
* · Easy and secure containerized applications management.
* · Continuous Integration by adopting Azure Pipeline concept for Docker images creation for faster deployments and reliability
* · Create resources and infrastructure inside the Azure Kubernetes cluster through Deployments and services manifest files
* · AKS management service is free of charge in Microsoft Azure

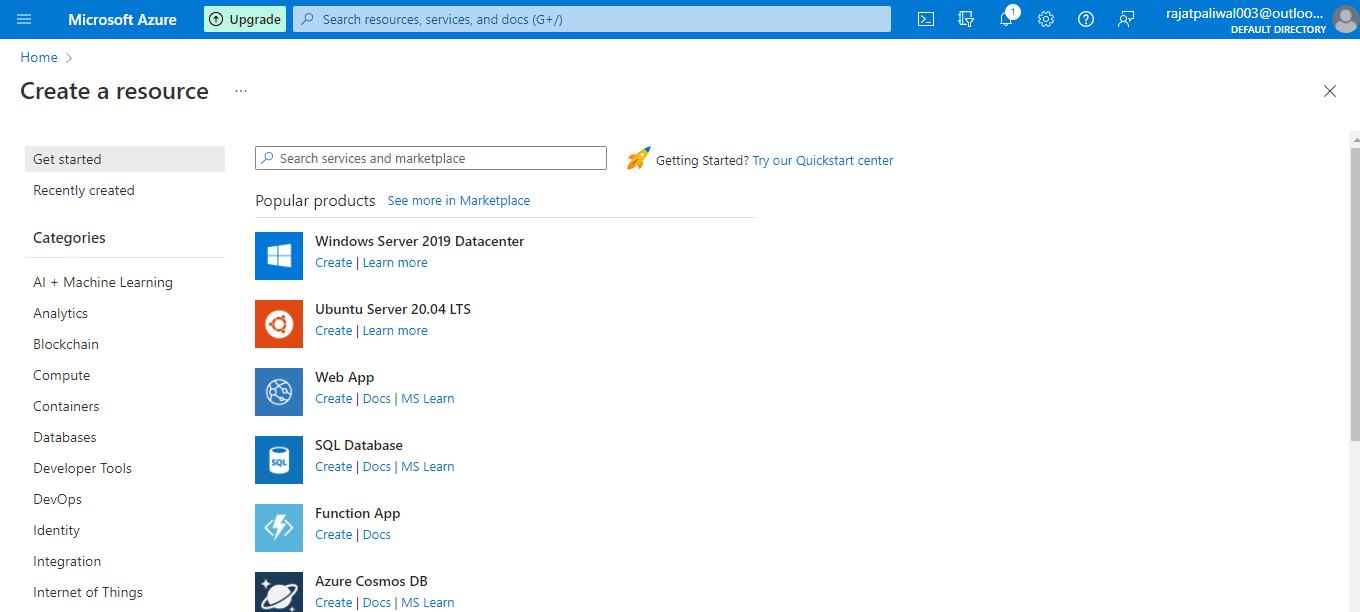
**Scheme:**

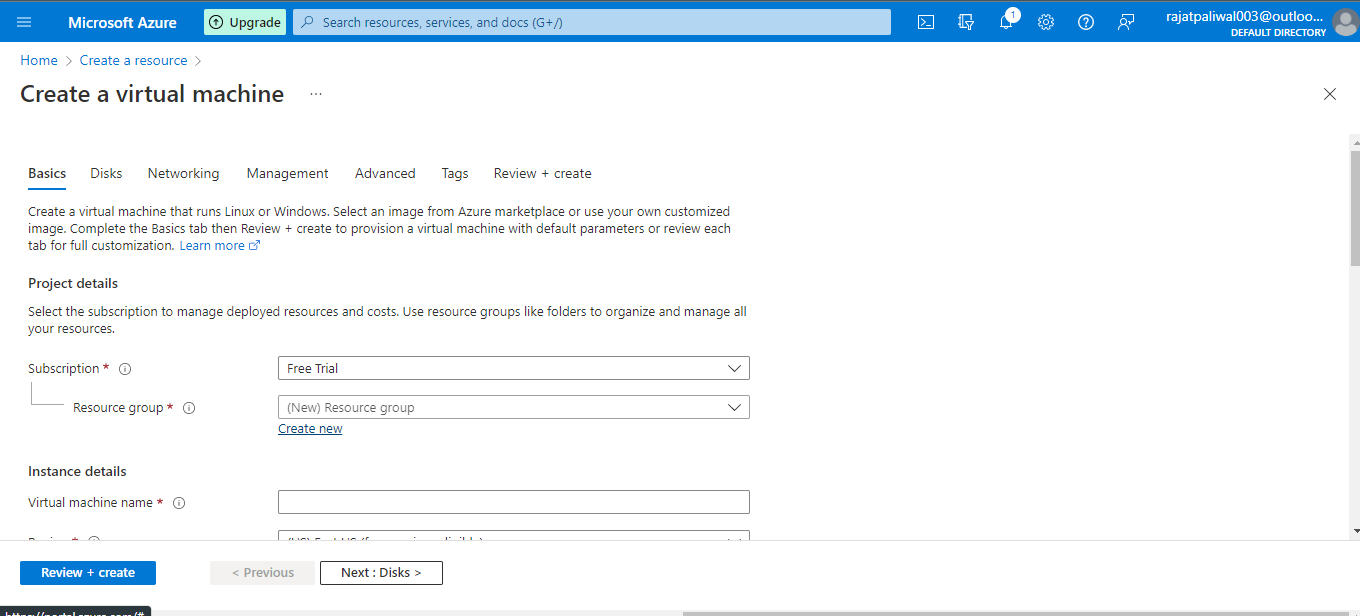
* Create Azure Resource Group
* Create Azure Container Registry
* Create Azure Kubernetes Cluster
* Create a Pipeline For Deployment to Kubernetes

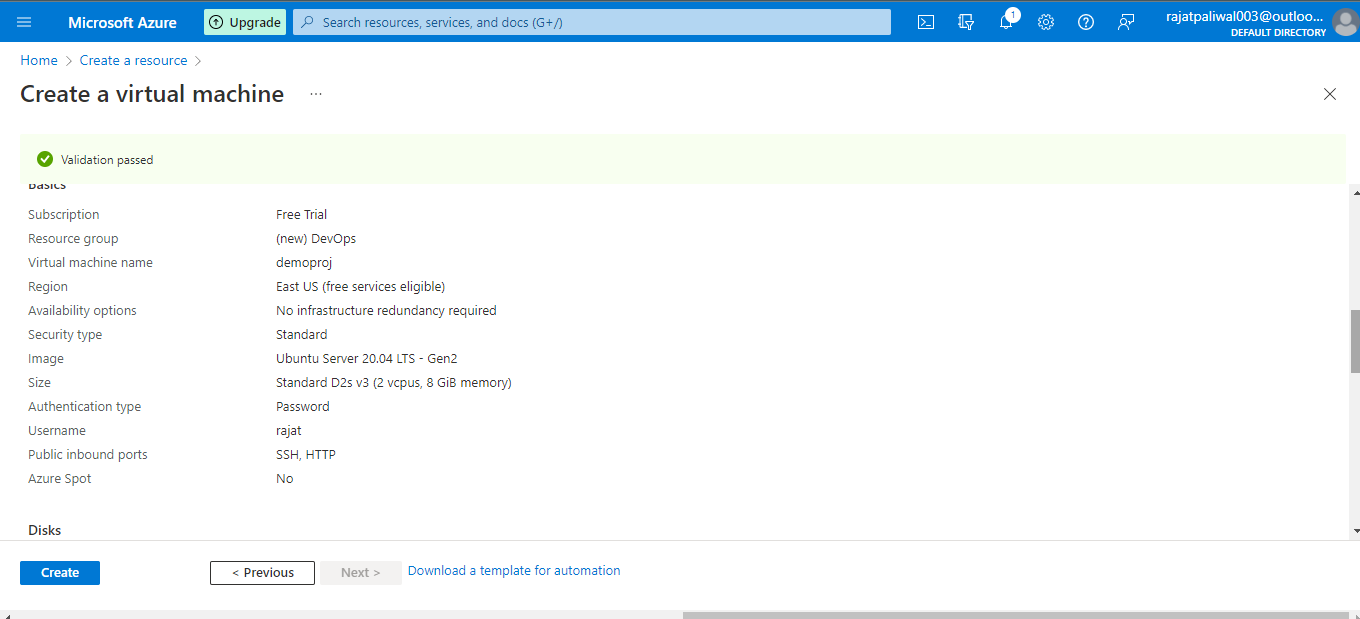
Let's get started,

**Step 1 - Create an Azure Resource Group**

Now we will create a resource group named “DevOps” for our docker images and Kubernetes cluster. We will keep our all resources in this resource group that we are creating.

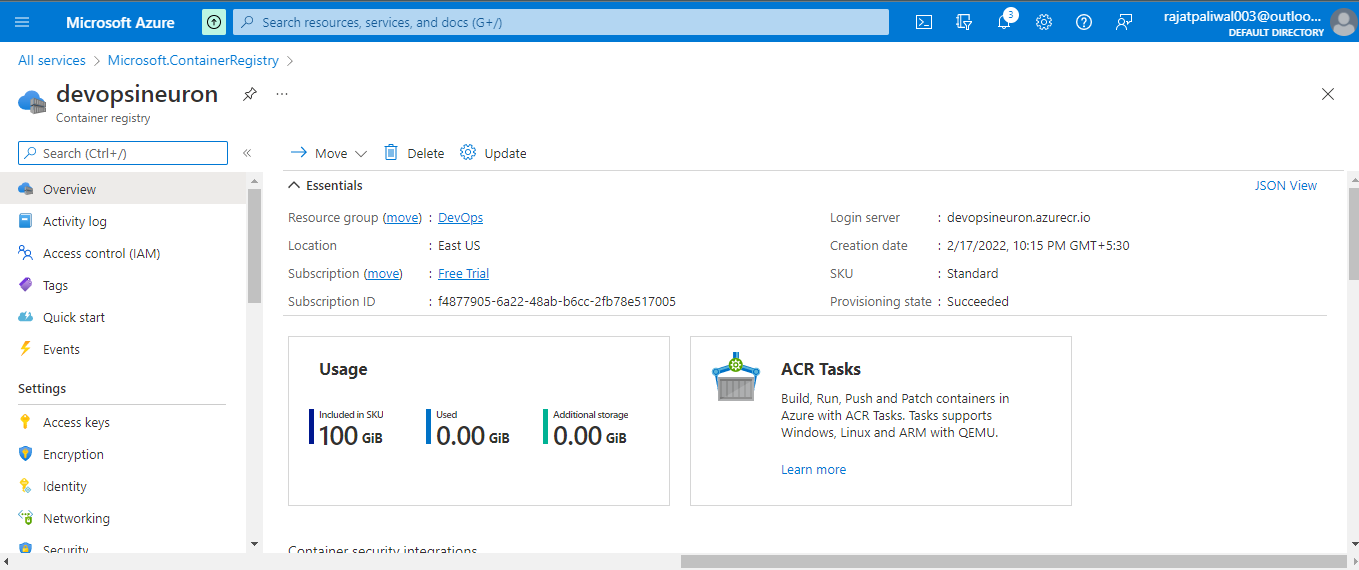






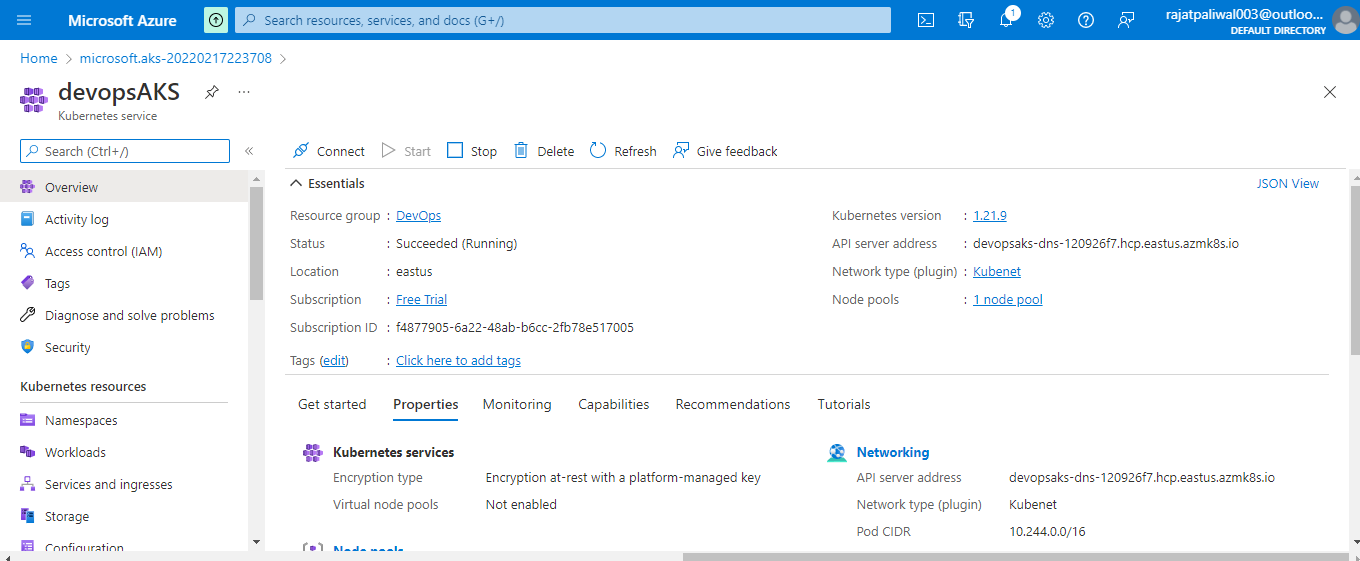
**Step 2 - Create Azure Container Registry**

We have created a resource group. Now for every new resource, we will add to this resource group.



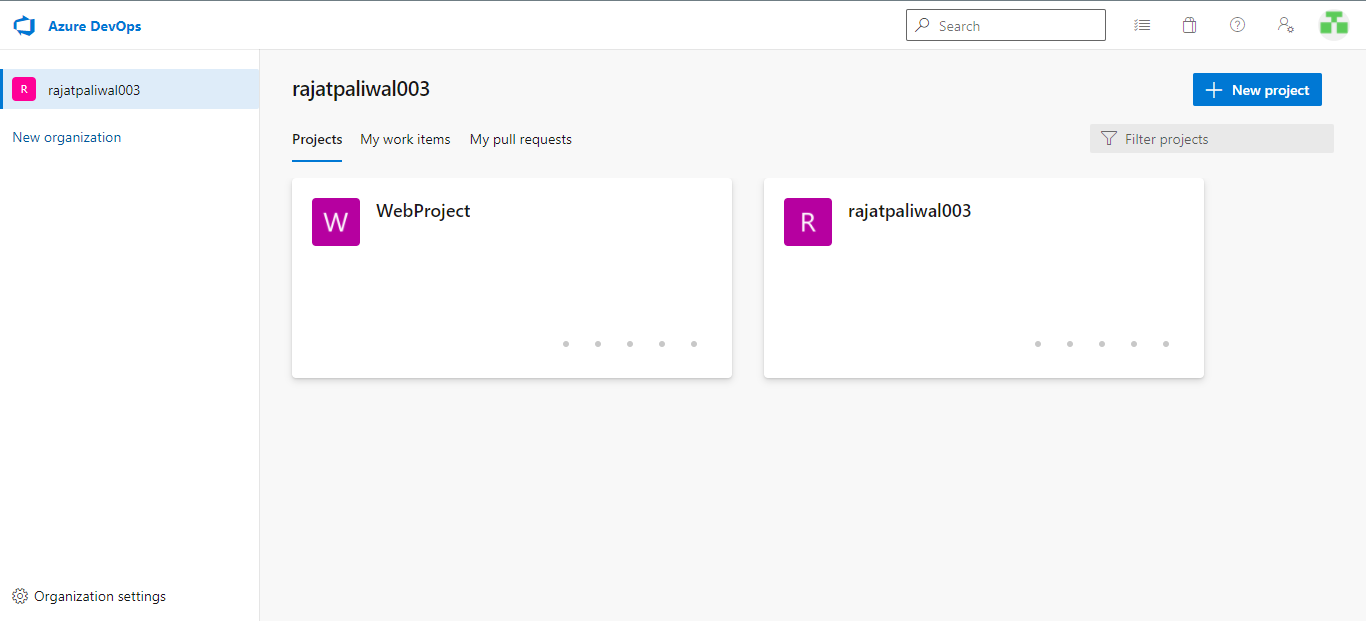
**Step 3 - Create Azure Kubernetes Cluster**

Click on create a resource and choose Kubernetes Service.

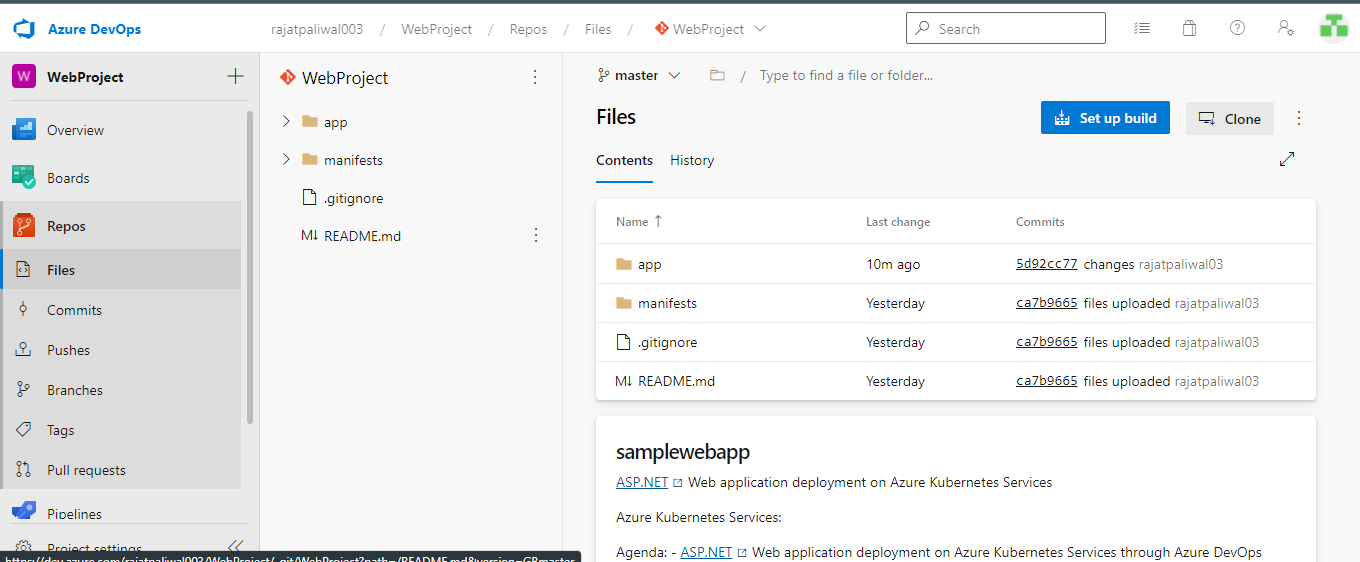


**Step 4 - Create a Pipeline For Deployment to Kubernetes**

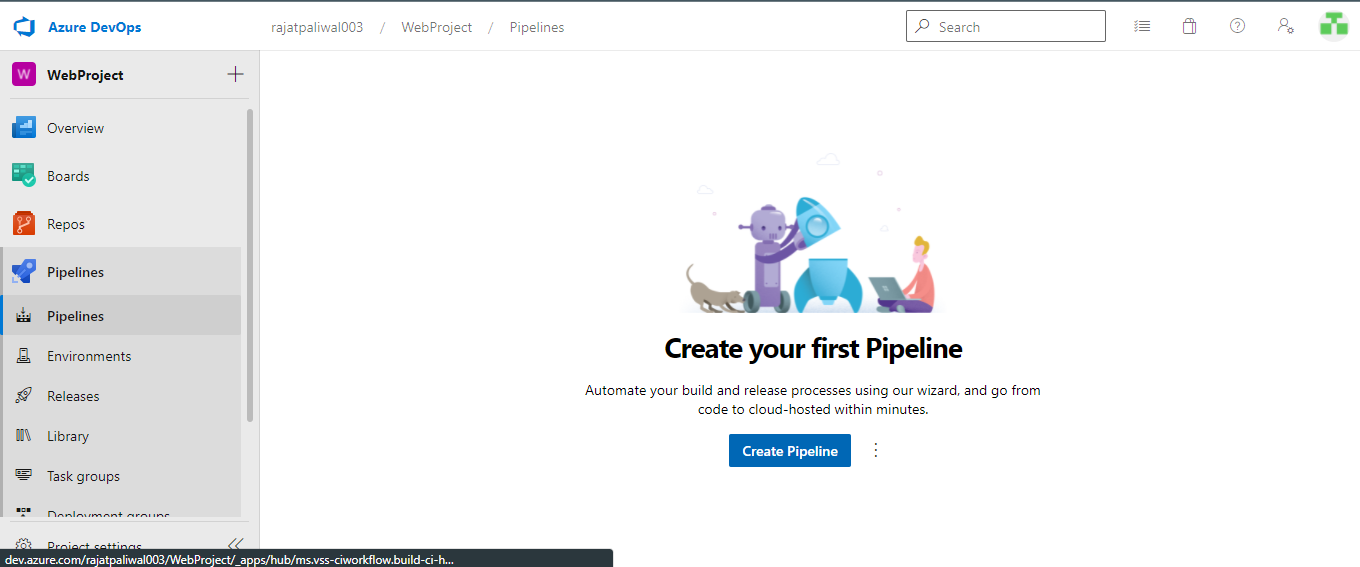
Login to your [DevOps](https://dev.azure.com/) accounts and create a new project named WebProject.

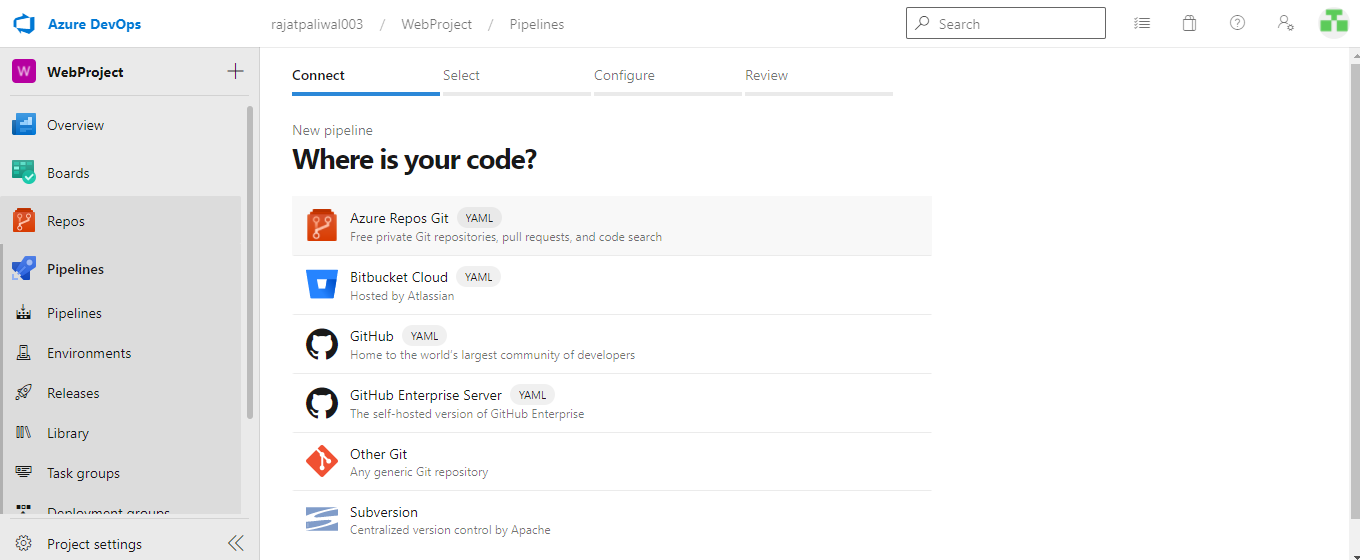


Open this newly created project and add source code to this project repo.

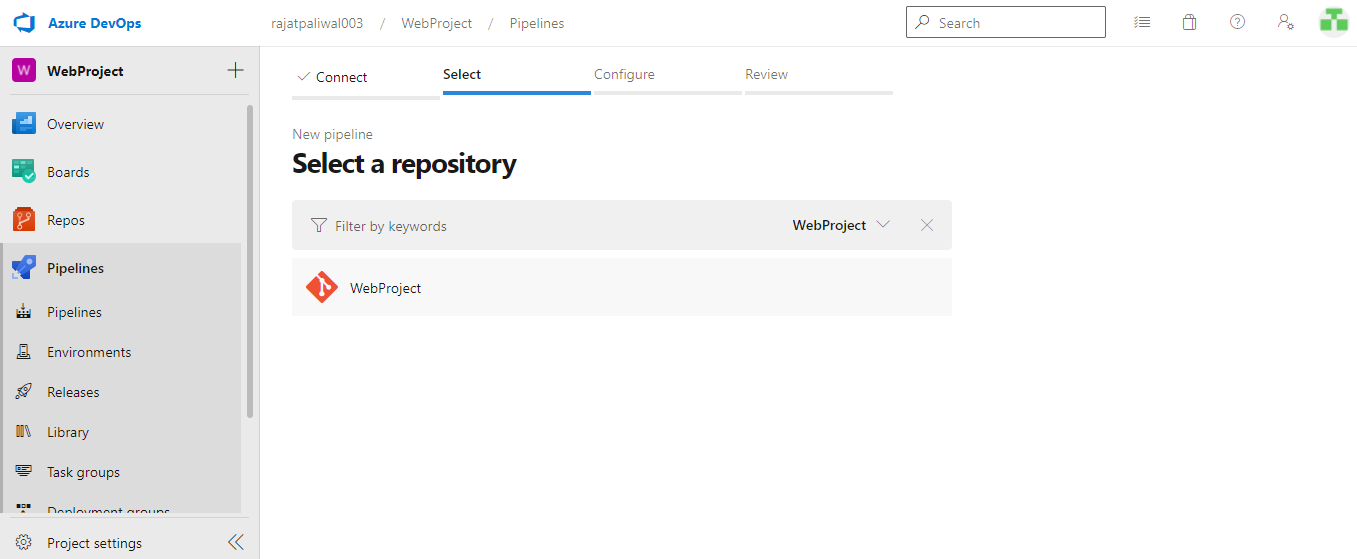


Next, click on the Pipeline tab and then choose the pipelines. This pipeline will build the source code and then create the docker image and will push to the Azure container registry that we created in the last step and create a new deployment to the Azure Kubernetes cluster. Click on create pipeline and choose the source code option. As I mentioned earlier that, I am using Azure Repos for my source code. I will select the Azure repo git option.

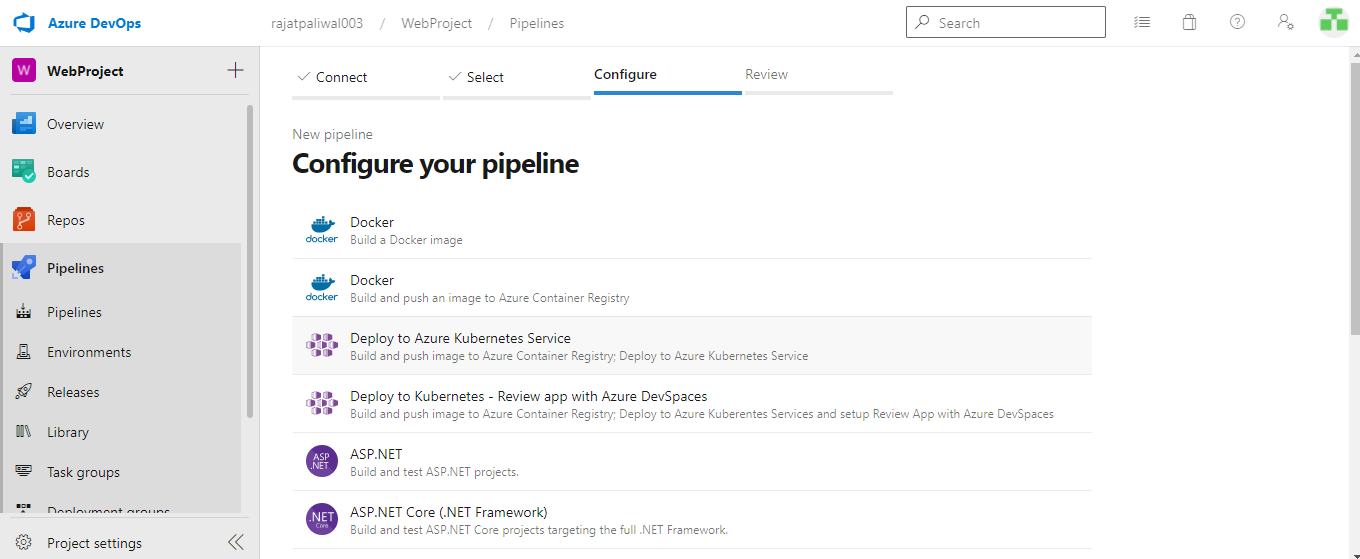




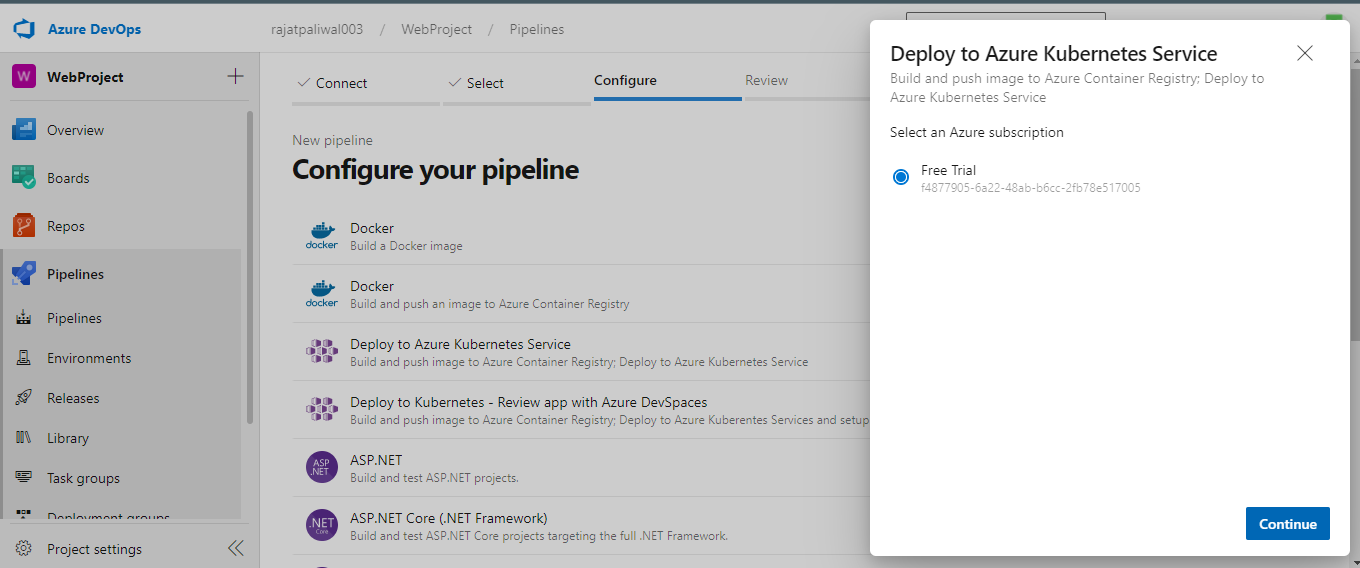
Select your repository and then branch. In my case, I will choose WebProject.



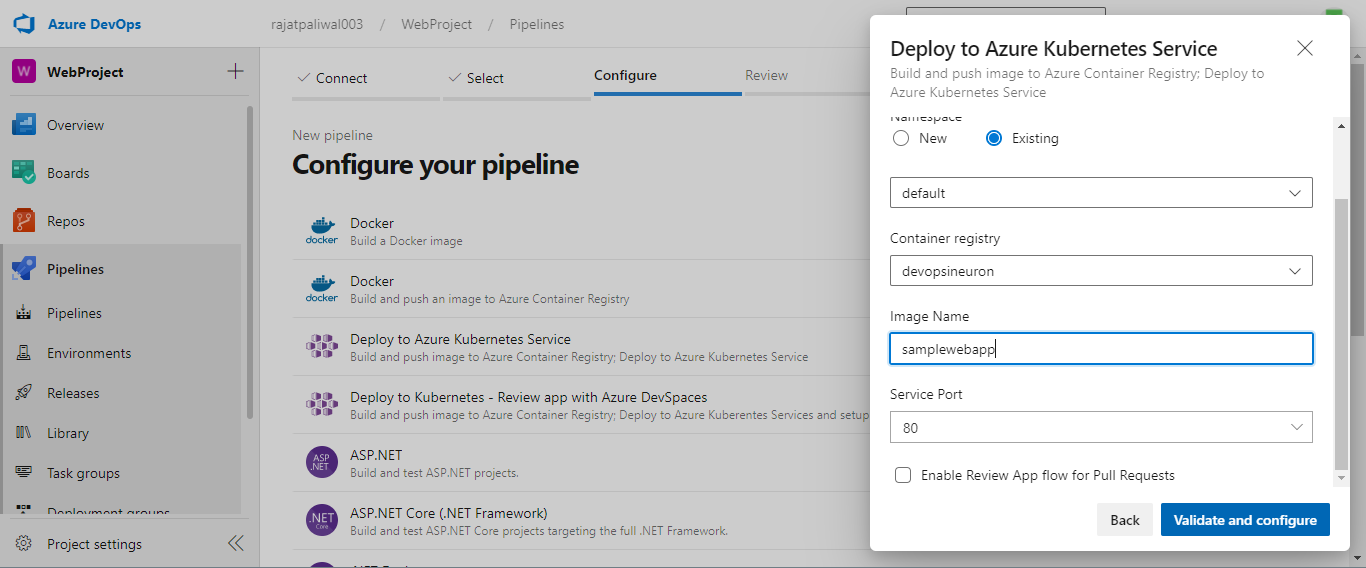
Next, select the Deploy to Azure Kubernetes Service (build and push the image to Azure Container Registry; Deploy to Azure Kubernetes Service) option from the Configure your pipeline page.



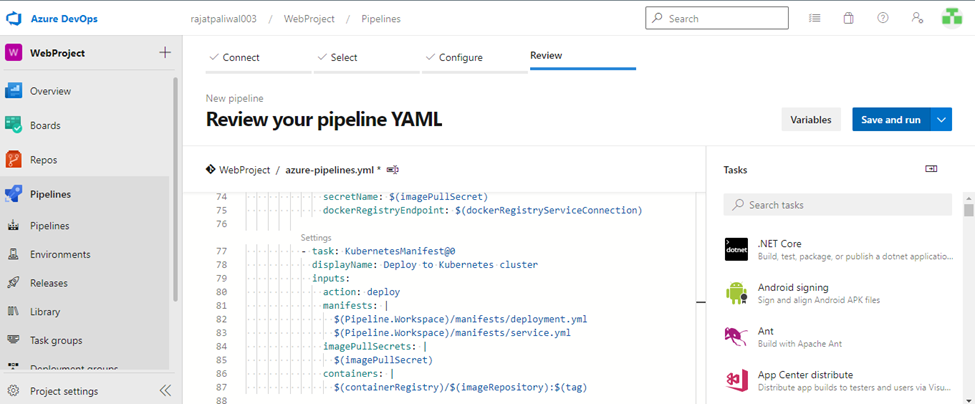
Next, select your active subscription from pop up and hit continue.



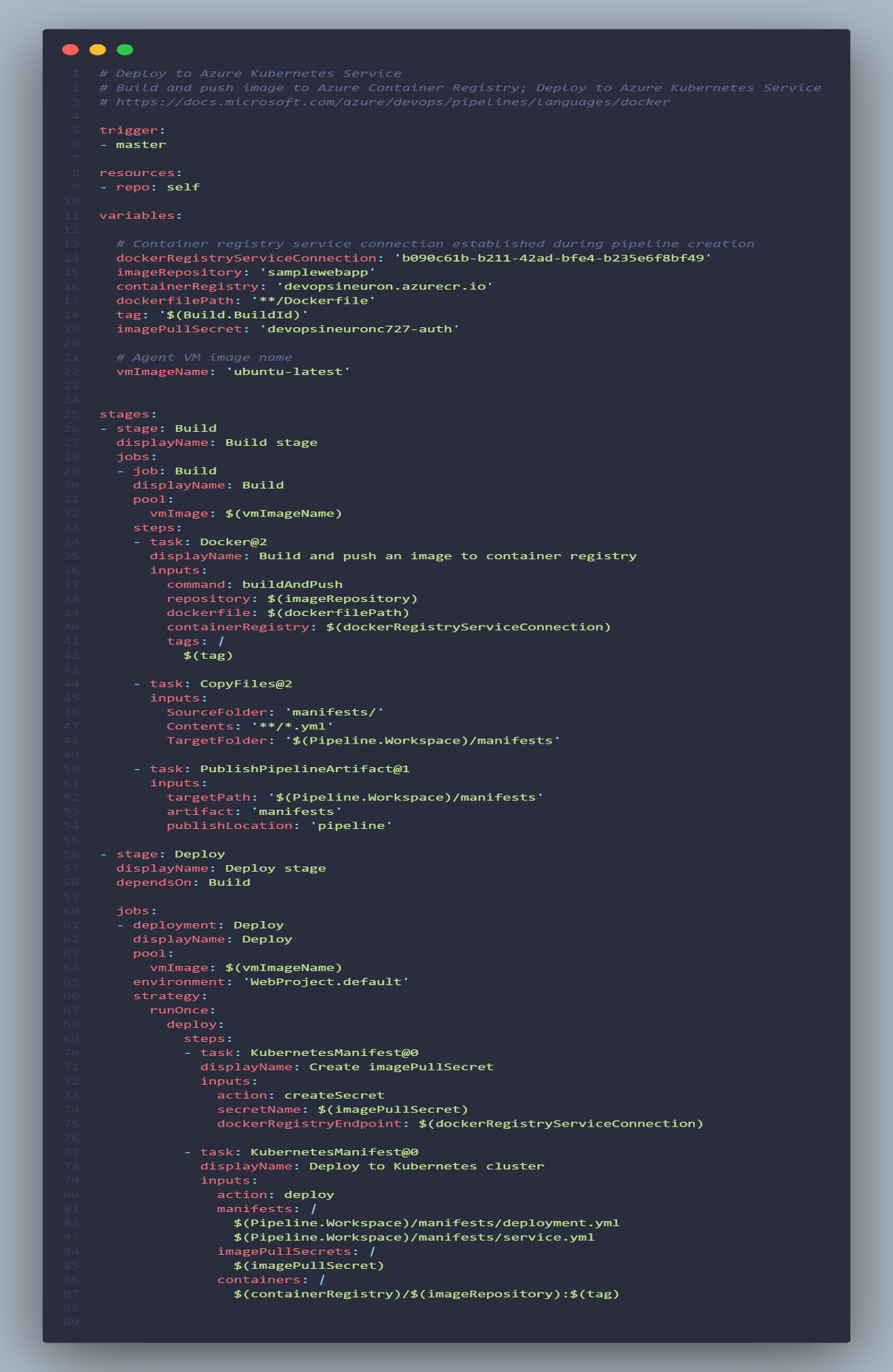
Next, select your Cluster name from the dropdown and choose the Namespace (existing or new), and select the container registry and enter the image name that you want to use. The service Port option leaves it with a default value. Finally, click on validate and configure.



Next, you will see an azure-pipelines.yml file that is a predefined template for building and pushing images to the Azure container registry and deploying them to the Kubernetes cluster. Just update the branch trigger name main to master or any branch name that you want to use for this build pipeline. Now we are good to go and we can save this pipeline as-is. Click on save and run.

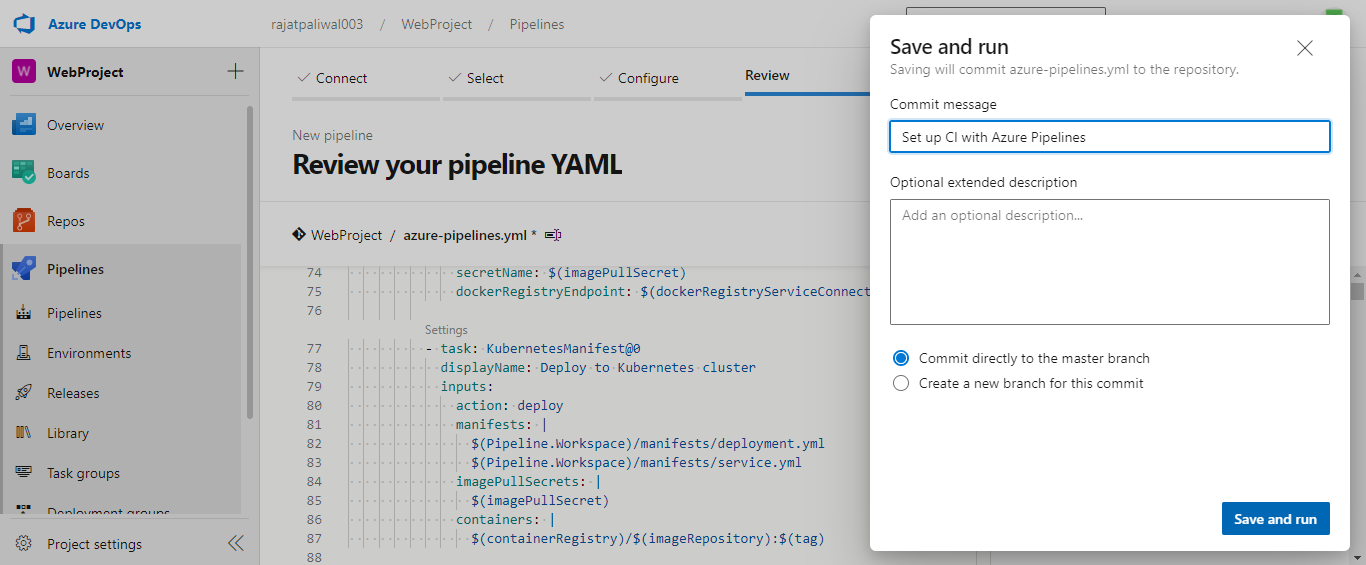


**azure-pipelines.yml :**

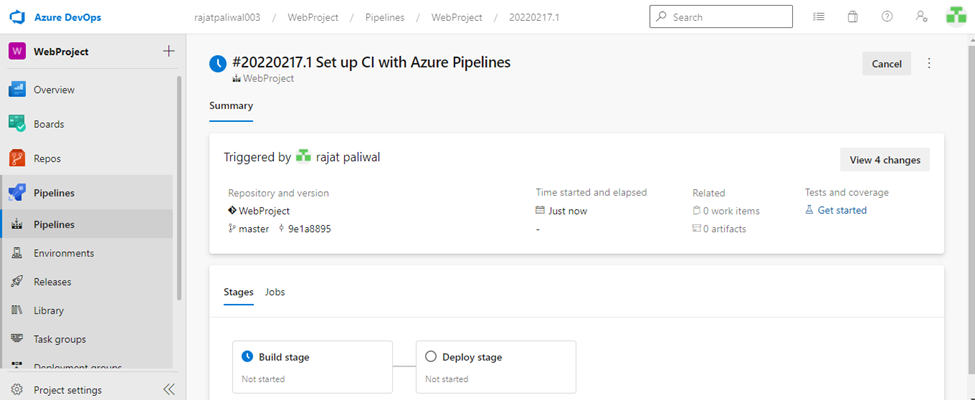


Here you will see in the output popup the three files will be added to our repository (azure-pipeline, deployment.yml, service.yml) choose the commit directly to the master branch option and hit Save and Run.

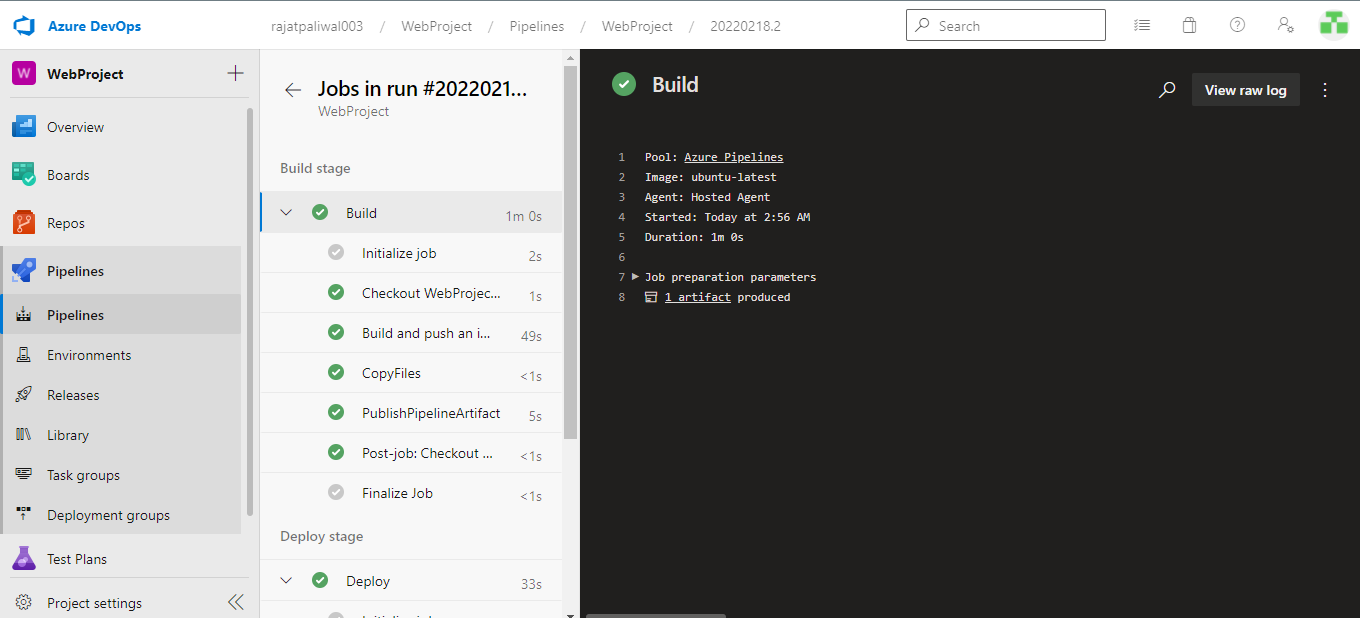
 



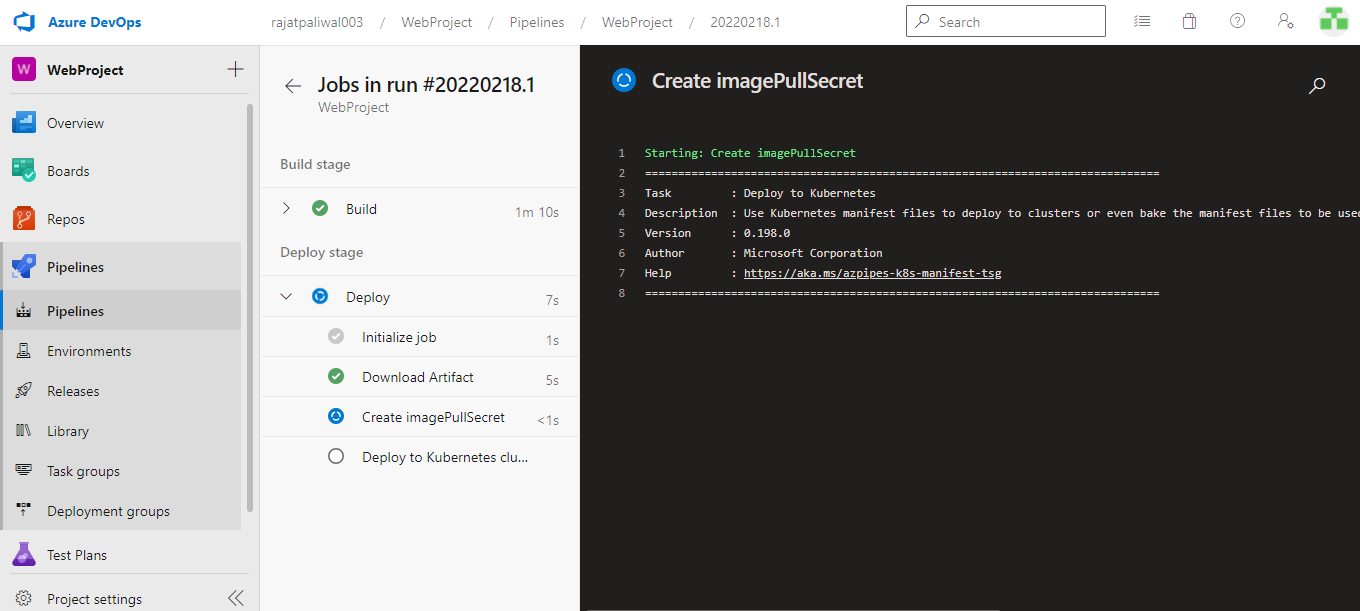
After Save and Run, you will see two stages. In the first stage, our code will be built and the docker image will be created and pushed to the Azure container registry. In the second stage, it will be deployed to the Kubernetes cluster.



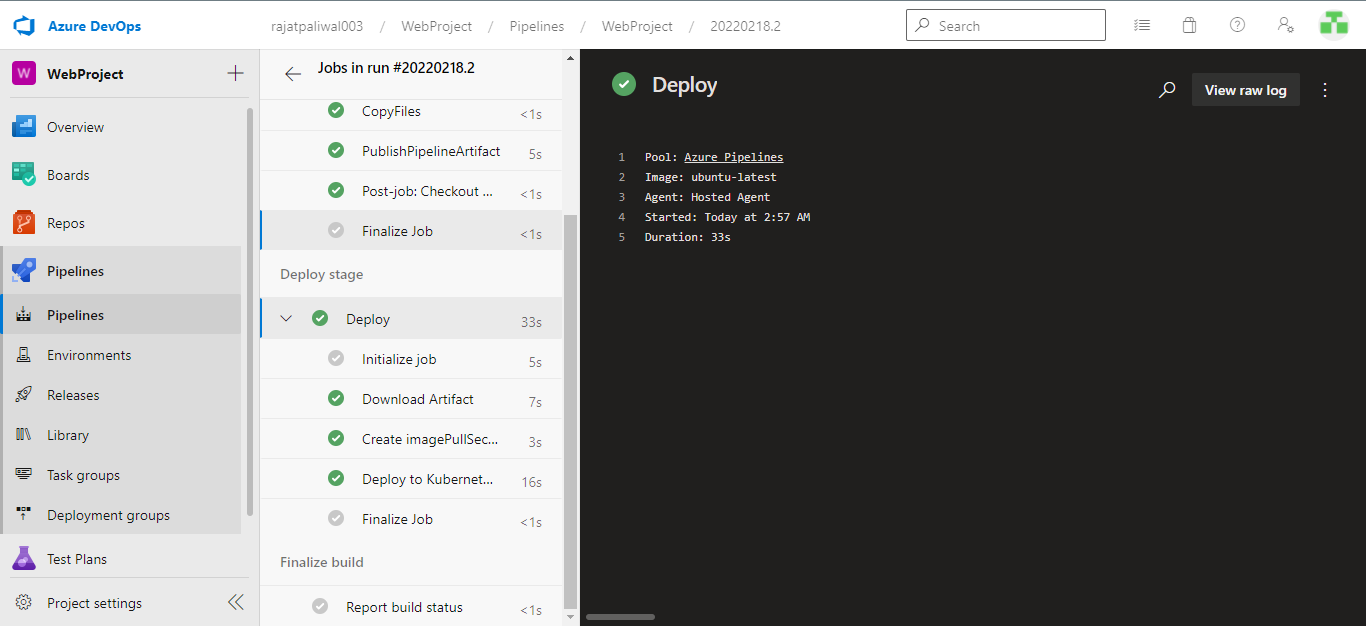
The stage of building an image and pushing it to the Azure container registry is done.

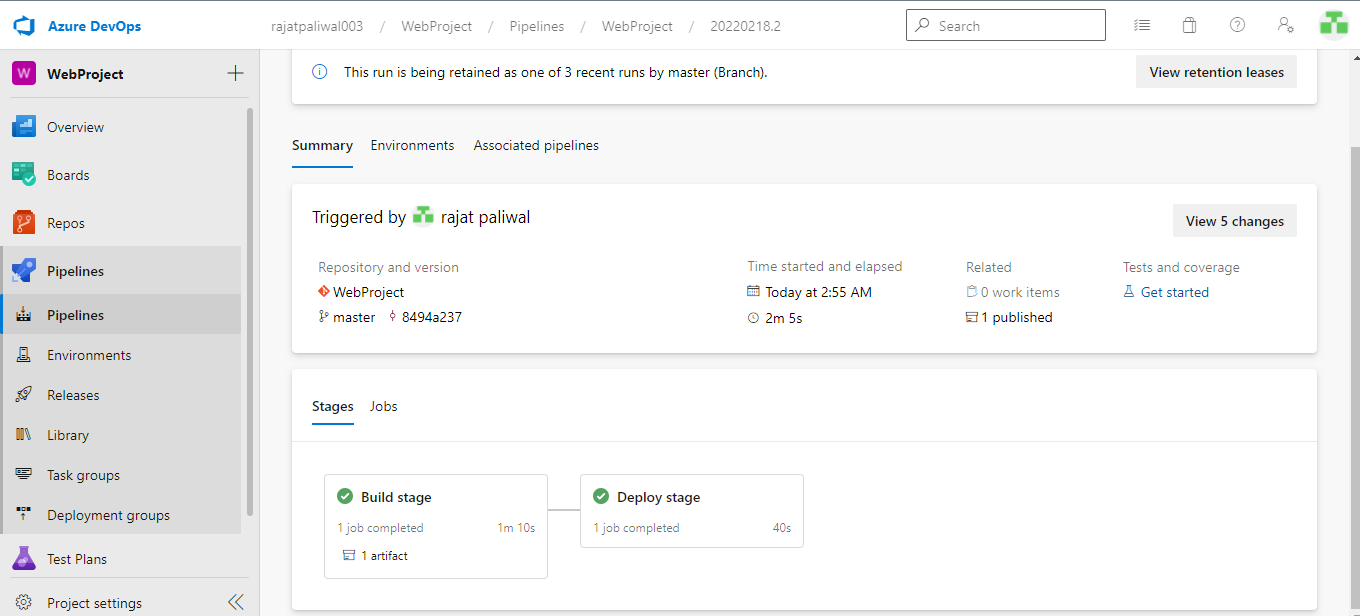


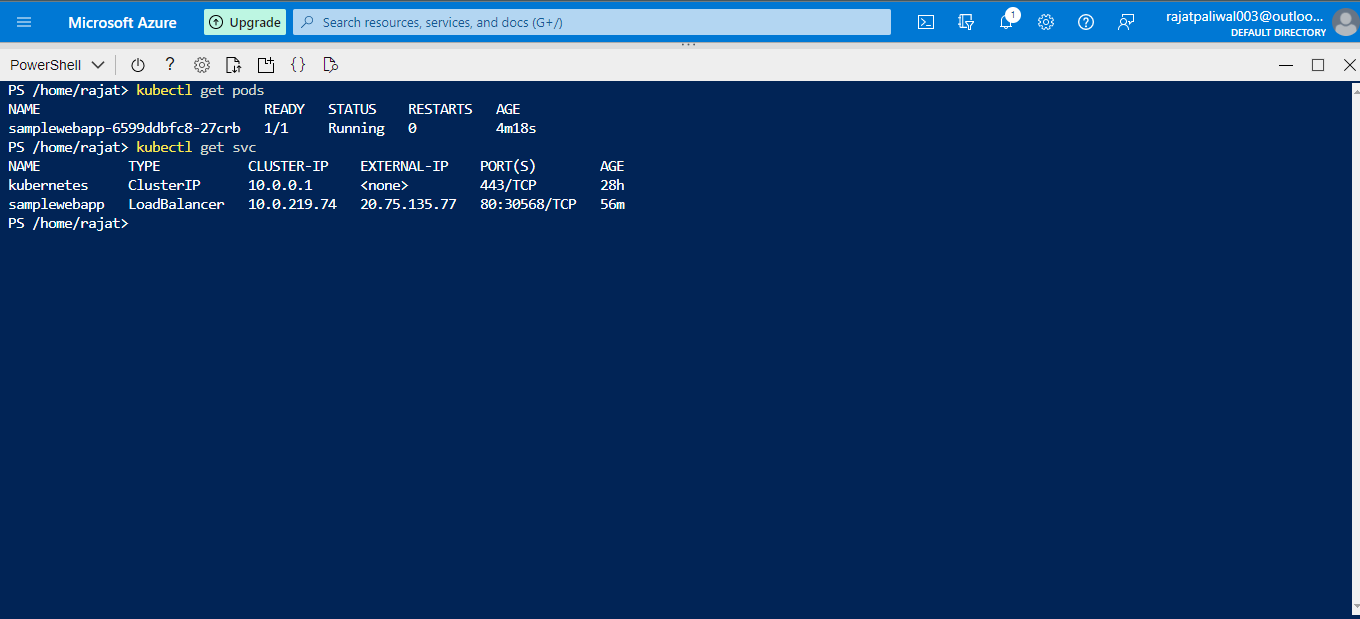
The process of deploying to the Kubernetes cluster is in progress.



The stage of deploying to the Kubernetes cluster is done.

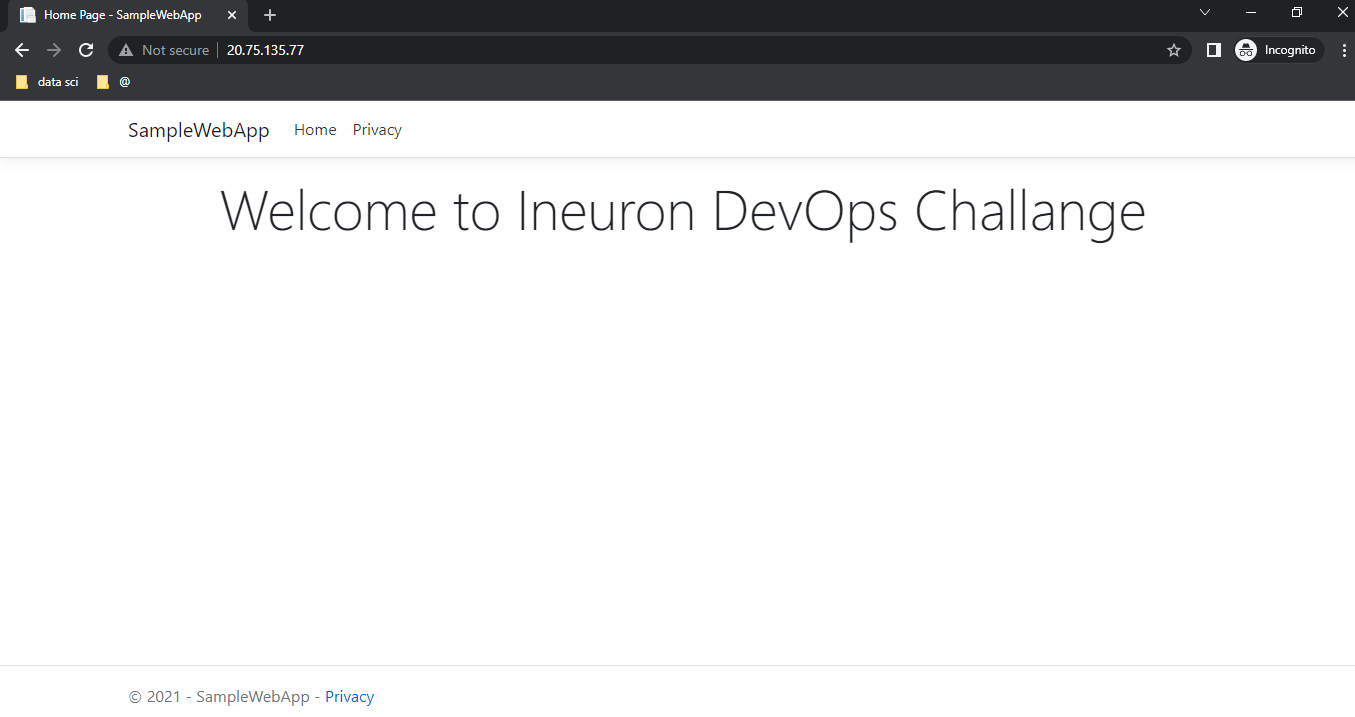






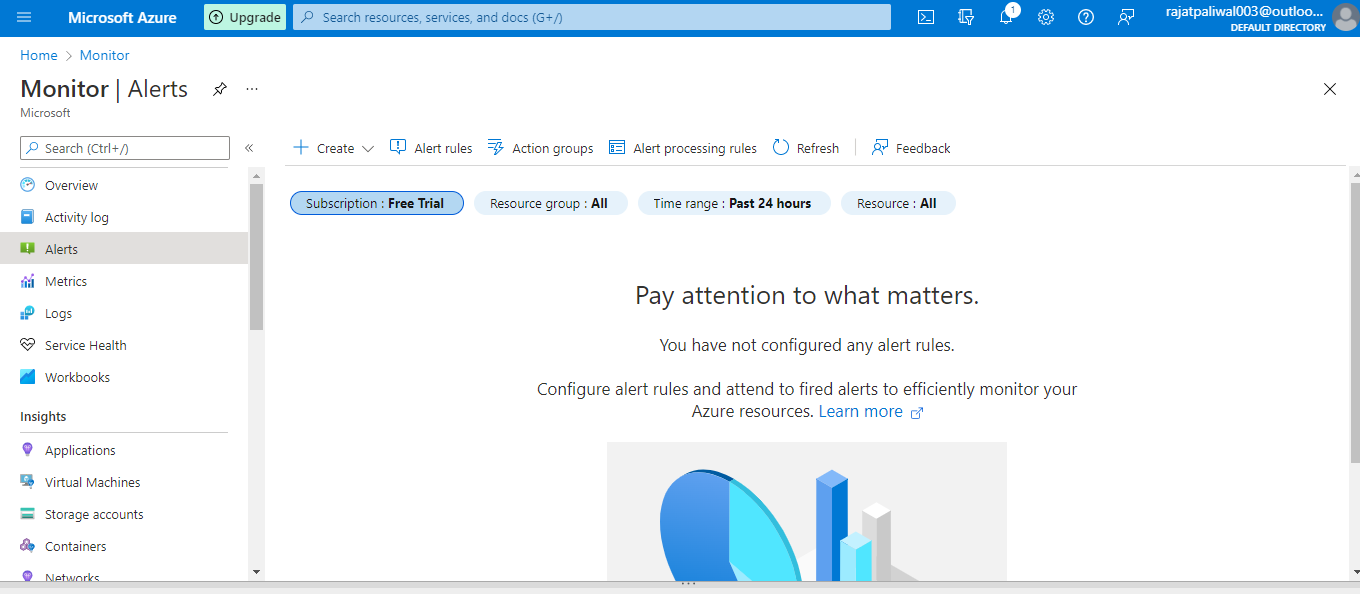
Grab the External IP address of your service and open it in any browser. Amazing, the Azure Kubernetes cluster is up and running in the Azure portal. So, this was the process of building a CI/CD pipeline for Azure Kubernetes Cluster.

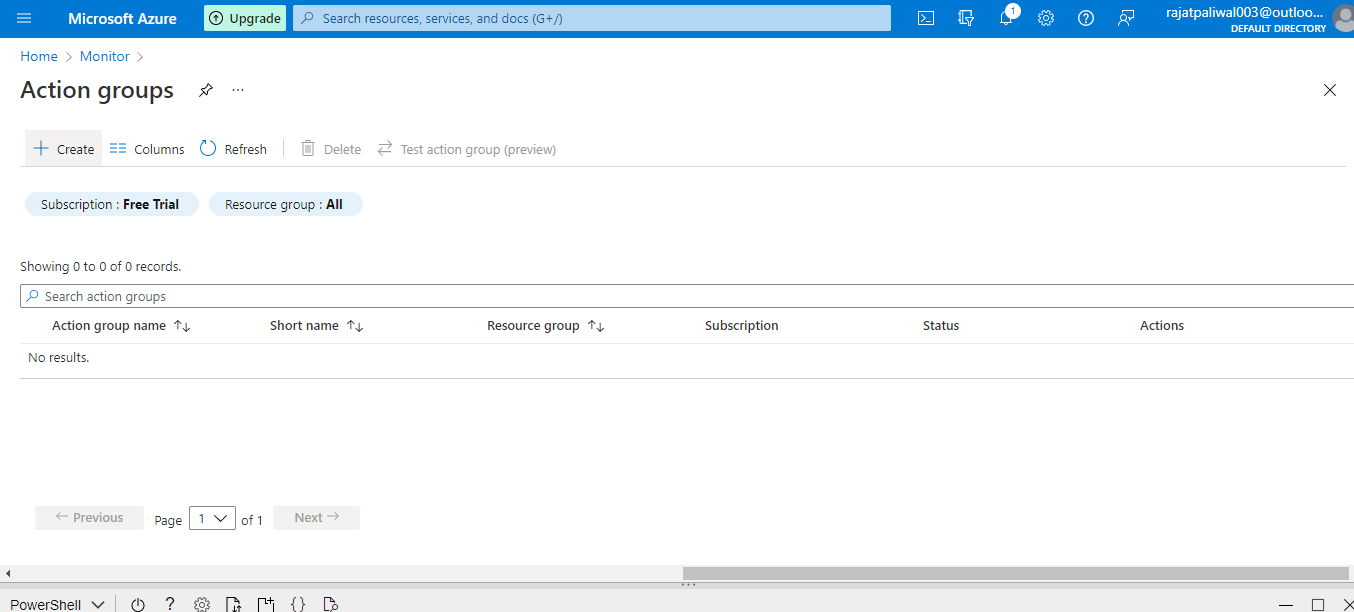
**Web App:**

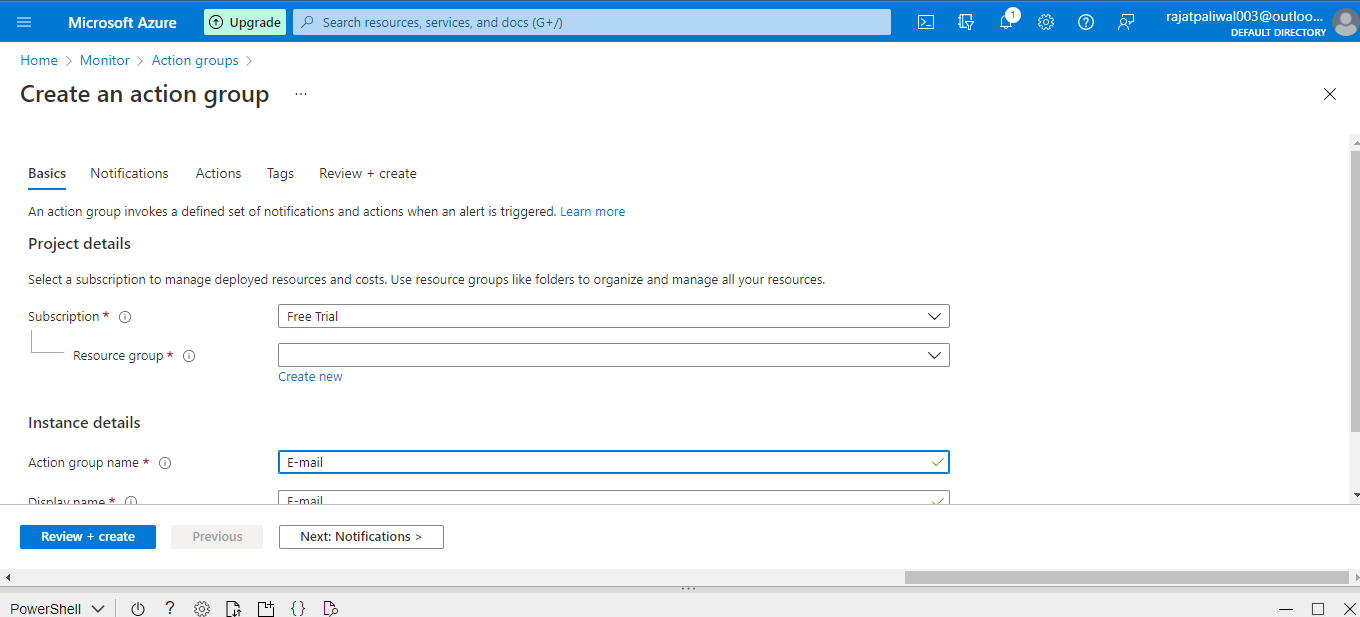


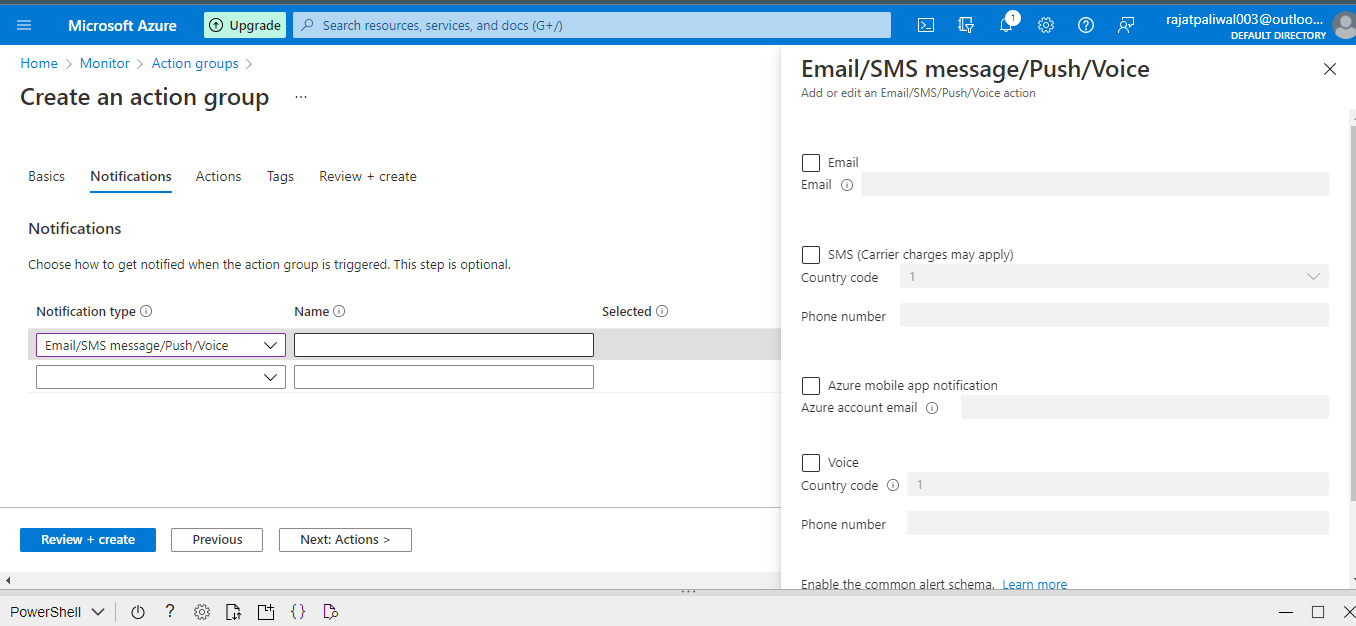
**Monitoring logs and email alerts:**

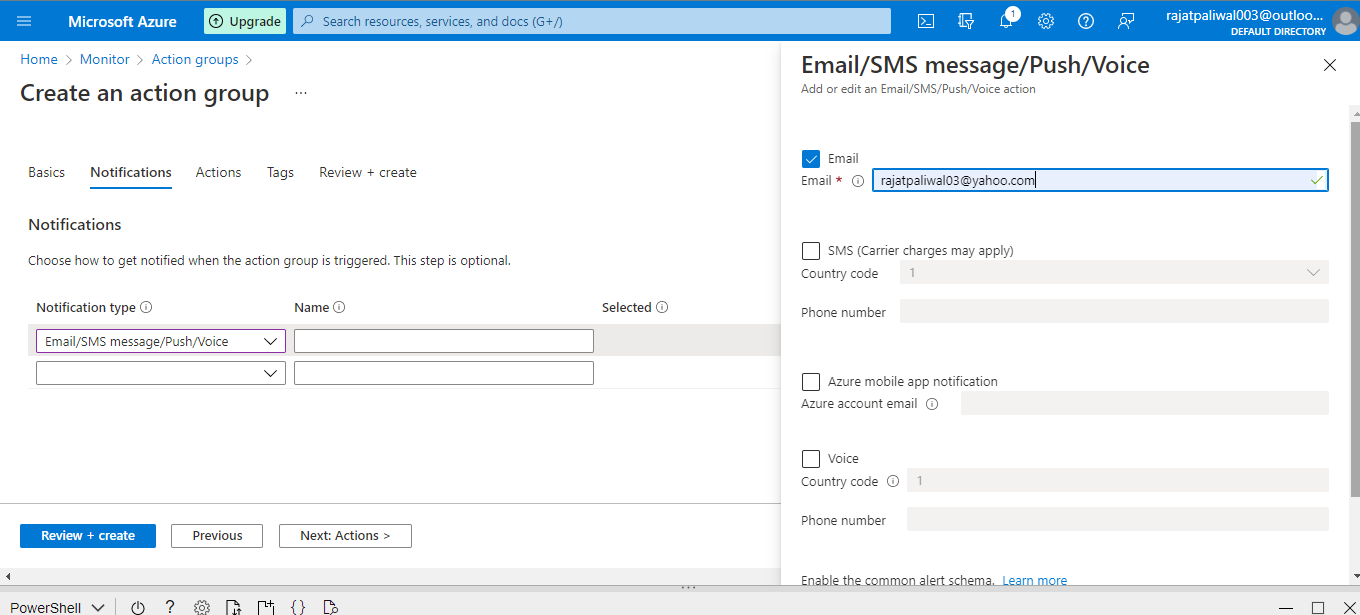
1. In the portal, select the relevant resource.
2. Under **Monitoring**, select **Alerts**.
3. From the top command bar, select **Alert rules**.
4. Select the alert rule that you want to edit.
5. Edit any fields necessary, then select **Save** on the top command bar.











After entering e-mail, click review + create .