PROJECT

Deploying a Web Application with DevOps Tools.

Description:

You will create a simple web application, containerize it using Docker, set up infrastructure using Terraform, and use Git for version control.

You'll deploy this application to Azure.

Tools Used:

Git, Docker, Terraform, Azure/AWS
By the end of this project, you will have a fully automated DevOps pipeline that allows you to develop, deploy, and manage your web application using Git, Docker, Terraform, and your chosen cloud platform (Azure/AWS).

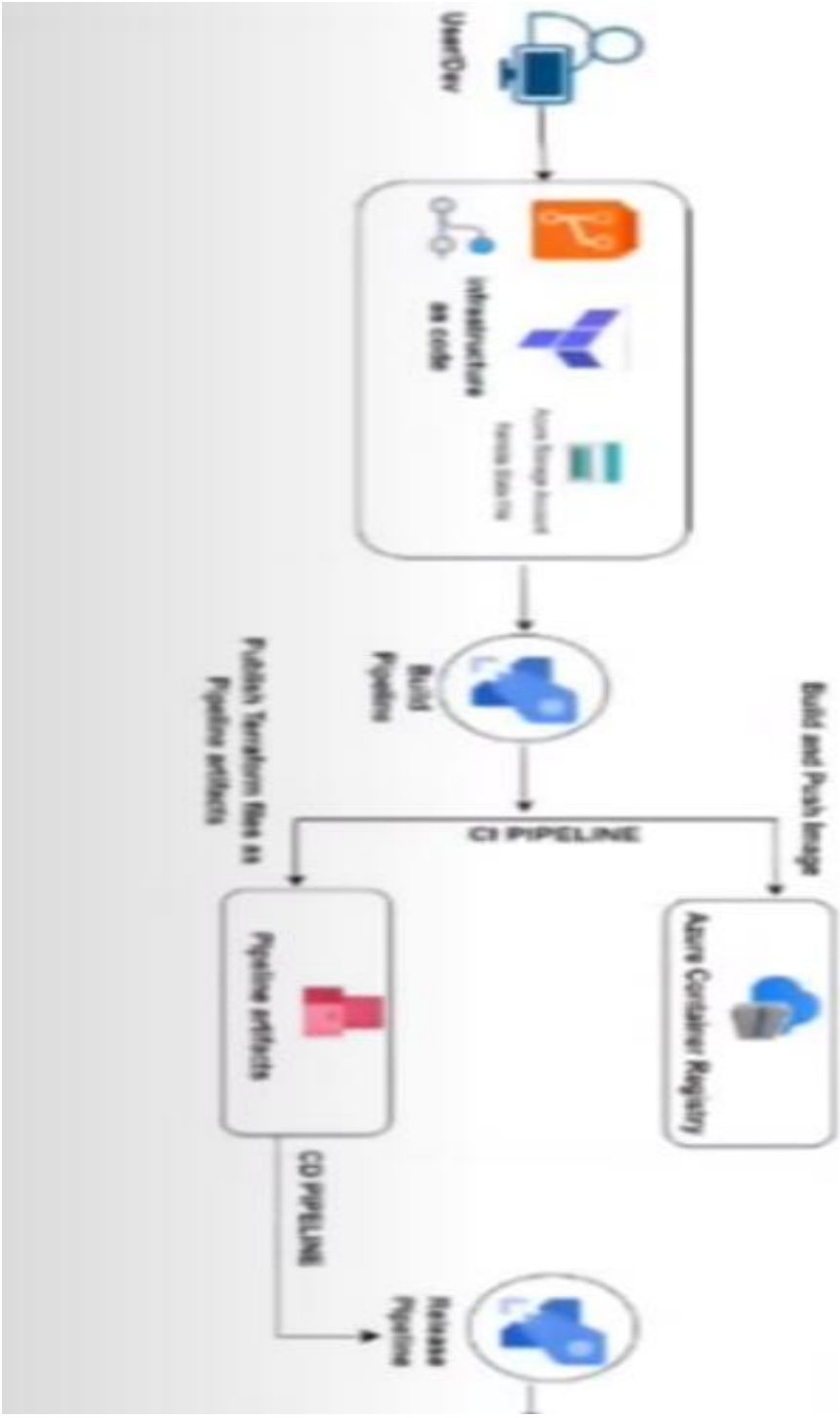
This project provides hands-on experience with key DevOps tools and practices commonly used in the industry.

Steps:

- Set up Git Repository.
- Develop the Web Application: Write a simple web application using any frame work of your choice [ex: node.js] and Create a Docker file to containerize your application.
- Set up Infrastructure with Terraform.
- Deploy to Azure/AWS
- Automate Deployment, set up git hooks or integrate with CI/CD and Configure the CI/CD pipeline to trigger deployment whenever changes are pushed to the Git repository.
- Implement automated tests for your web application.

CONTENTS

Lab secenario



Overview & Key Concepts

In this section we will cover key concepts used in this guide, including

Overview of what's covered in this guide.

Terraform

Terraform is a popular infrastructure-as-code tool that allows you to automate the provisioning and management of infrastructure resources. It uses configuration files written in the HashiCorp Configuration Language (HCL) to define the desired state of your infrastructure, and it uses various commands to apply those configurations and manage your infrastructure resources.

Azure Resources Group

A resource group is a **container that holds related resources for an Azure solution**. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group.

Azure app service

Azure App Service is an **HTTP-based service for hosting web applications, REST APIs, and mobile back ends**. You can develop in your favorite language, be it.NET,.NET Core, Java, Node.js, PHP, and Python. Applications run and scale with ease on both Windows and Linux -based environments.

Azure web service app

Azure App Service is a **fully managed platform for creating and deploying web applications, REST APIs, and mobile back ends**. It supports various languages, such as.NET,.NET Core, Java, Node.js, PHP, and Python. It offers different plans to suit the needs of any application, from small websites to globally scaled web application.

Azure pipeline

Azure Pipelines automatically builds and tests code projects. It supports all major languages and project types and combines continuous integration, continuous delivery, and continuous testing to build, test, and deliver your code to any destination.

PREREQUISITES

1 You need to have a active Azure Account (paid or free).

Note : if you don't have an Azure Account please Check the Activity Guide "create Azure Account & Access Console" under Bonus Module.

2 You need to have a Active Azure DevOps Account.

3 You need to Create a new project based on the Docker template using Azure DevOps Generator , ensure you have enabled parallelism for this project Following instructions. Given in guide "Enable parallelism" Guide.

Terraform

- Write scripts of resource group, web service app, container registry
- Then Execute with Commands

Terraform init.

Terraform validate.

Terraform plan.

Terraform Apply.

➢ After click Yes.

```
Welcome   var.tf   main.tf
var.tf > variable "client_id"
variable "subscription_id" {
    type = string
    default = "b512b3d1-3c24-4726-ba99-97e74a39190c"
    description = "MyazureDevEnv-Subcription"

}
variable "client_id" {
    type = string
    default = "a819a33c-27c7-4017-936d-db01ab3c91cc"
    description = "Dev Client ID"


}
variable "client_secret" {
    type = string
    default =  "_6B8Q~1xubIrTxdnyJrchE_bDhKOOjgJyIeVVdyD"
    description = "Dev client Sceret"

}
variable "tenant_id" {
    type = string
    default = "4f3d58d0-6580-4f24-80f1-b91b6667ab3a"
    description = "Tenant ID"

}
```

```
}
resource "azurerm_resource_group" "rg" {
  name     = "myResourceGroup"
  location = "West Europe"
}

resource "azurerm_container_registry" "acr" {
  name                = "myContainerRegistr789y"
  resource_group_name = azurerm_resource_group.rg.name
  location            = azurerm_resource_group.rg.location
  sku                 = "Premium"
  admin_enabled       = true


}
```

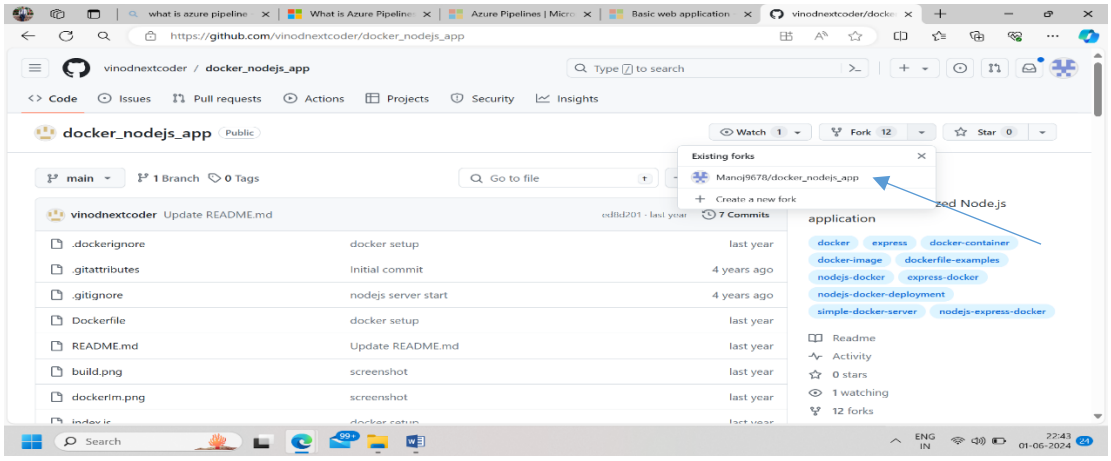
main.tf > resource "azurerm_linux_web_app" "webapp001"

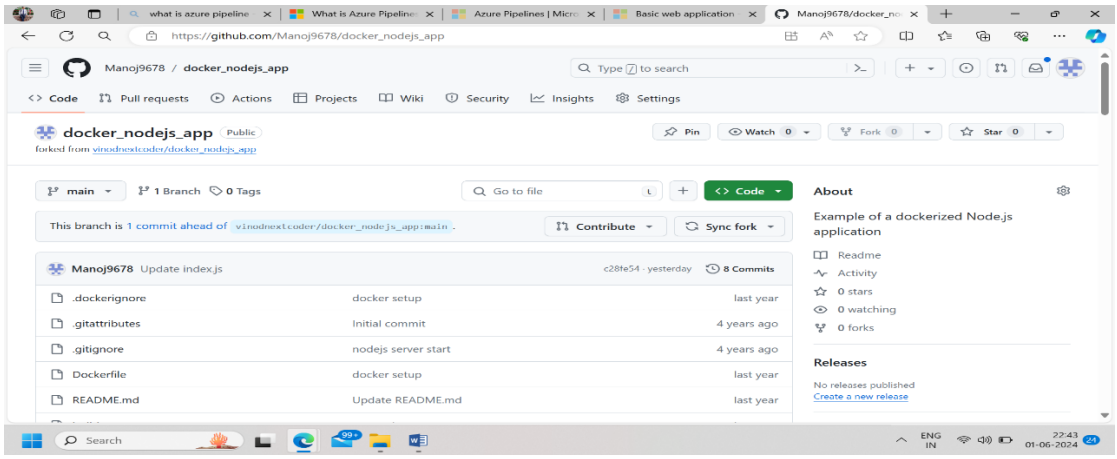
```
terraform {
  required_providers {

    azurerm={
        source = "hashicorp/azurerm"
        version = "3.99.0"
    }
  }
}
#configure the microsoft azure provider
provider "azurerm" {
    features {

    }
    subscription_id = var.subscription_id
    client_id = var.client_id
    client_secret = var.client_secret
    tenant_id = var.tenant_id
}

resource "azurerm_resource_group" "webservice0022" {
  name     = "webservice0022-resources"
  location = "Westus"
}
```

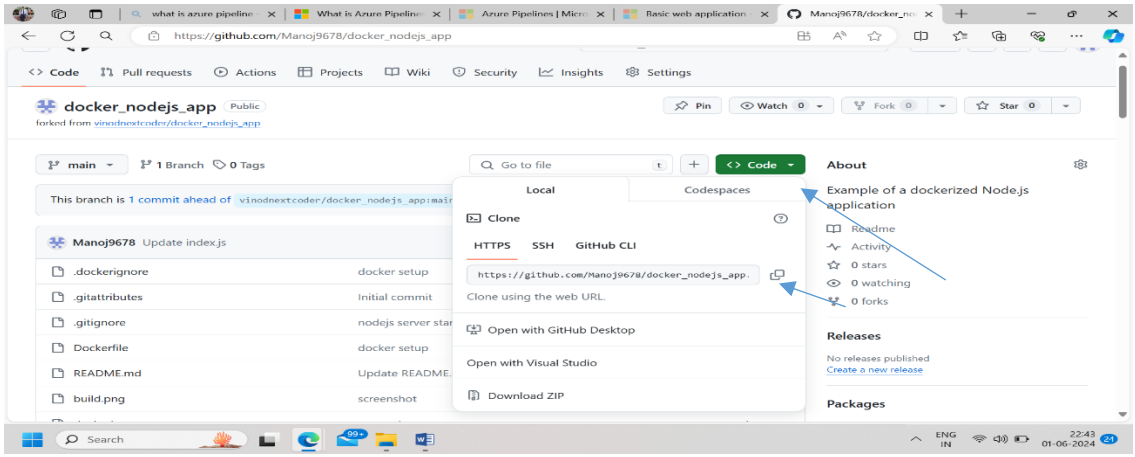
```
resource "azurerm_resource_group" "webservice0022" {
  name     = "webservice0022-resources"
  location = "Westus"
}

resource "azurerm_service_plan" "webserver0022" {
  name                = "webserver001"
  resource_group_name = azurerm_resource_group.webservice0022.name
  location            = azurerm_resource_group.webservice0022.location
  sku_name            = "P1v3"
  os_type             = "Linux"
}

resource "azurerm_linux_web_app" "webapp001" {
  name                = "webappitascode002"
  resource_group_name = azurerm_resource_group.webservice0022.name
  location            = azurerm_service_plan.webserver0022.location
  service_plan_id     = azurerm_service_plan.webserver0022.id


  site_config{
  }
}
```

Github

- Create a github Account.
- Add a fork repository file to in our github.
- web application using any frame work of your choice [ex: node.js] and Create a Docker file to containerize your application

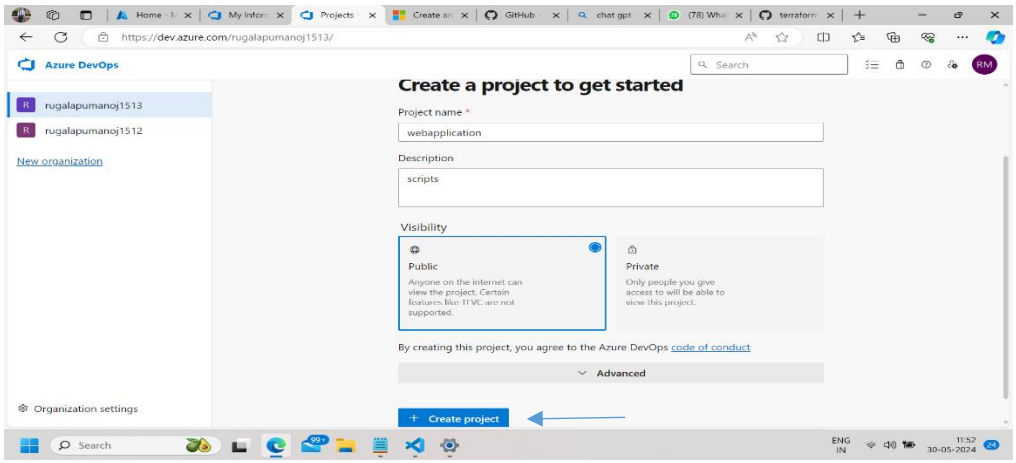




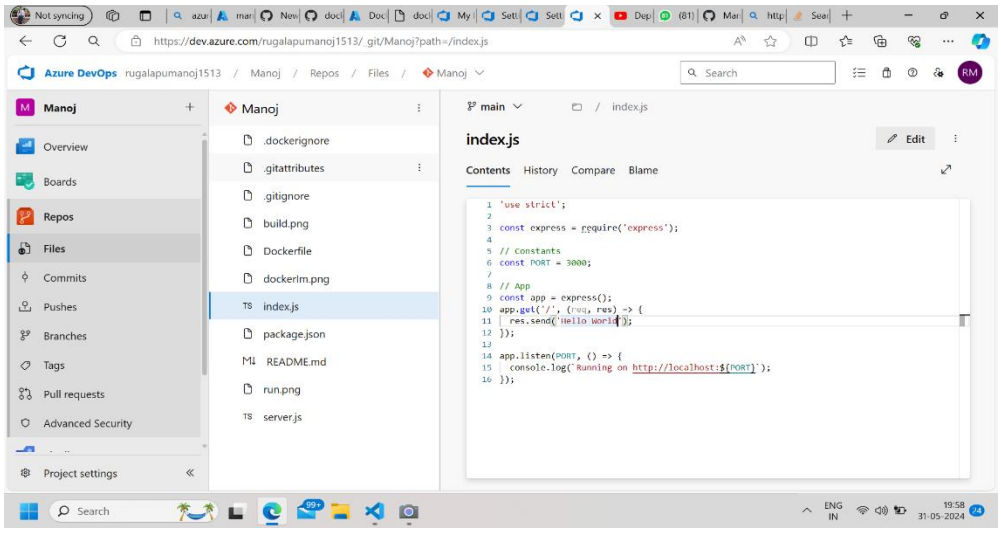


Deploying a Web Application in pipeline

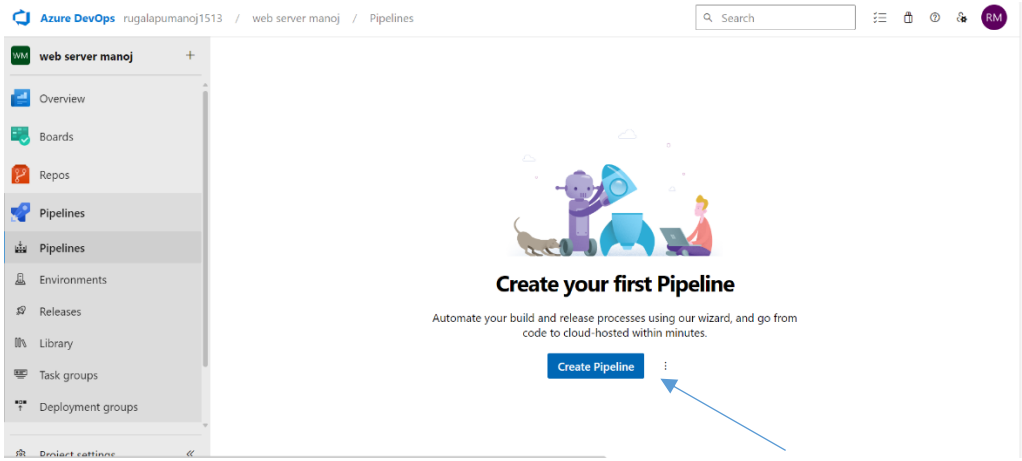
- open a Azure portal
- Create a Azure DevOps Organsiations.



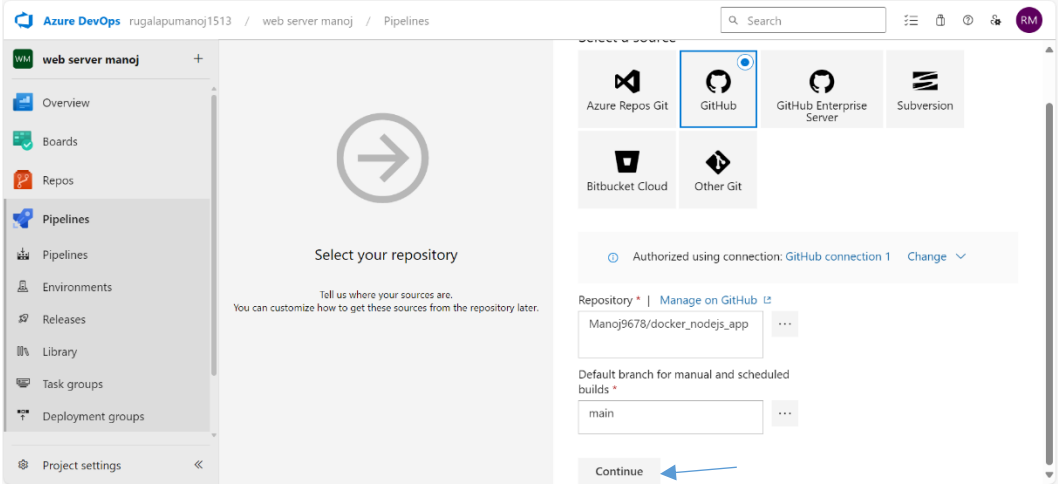
- Let's import from github repository file.



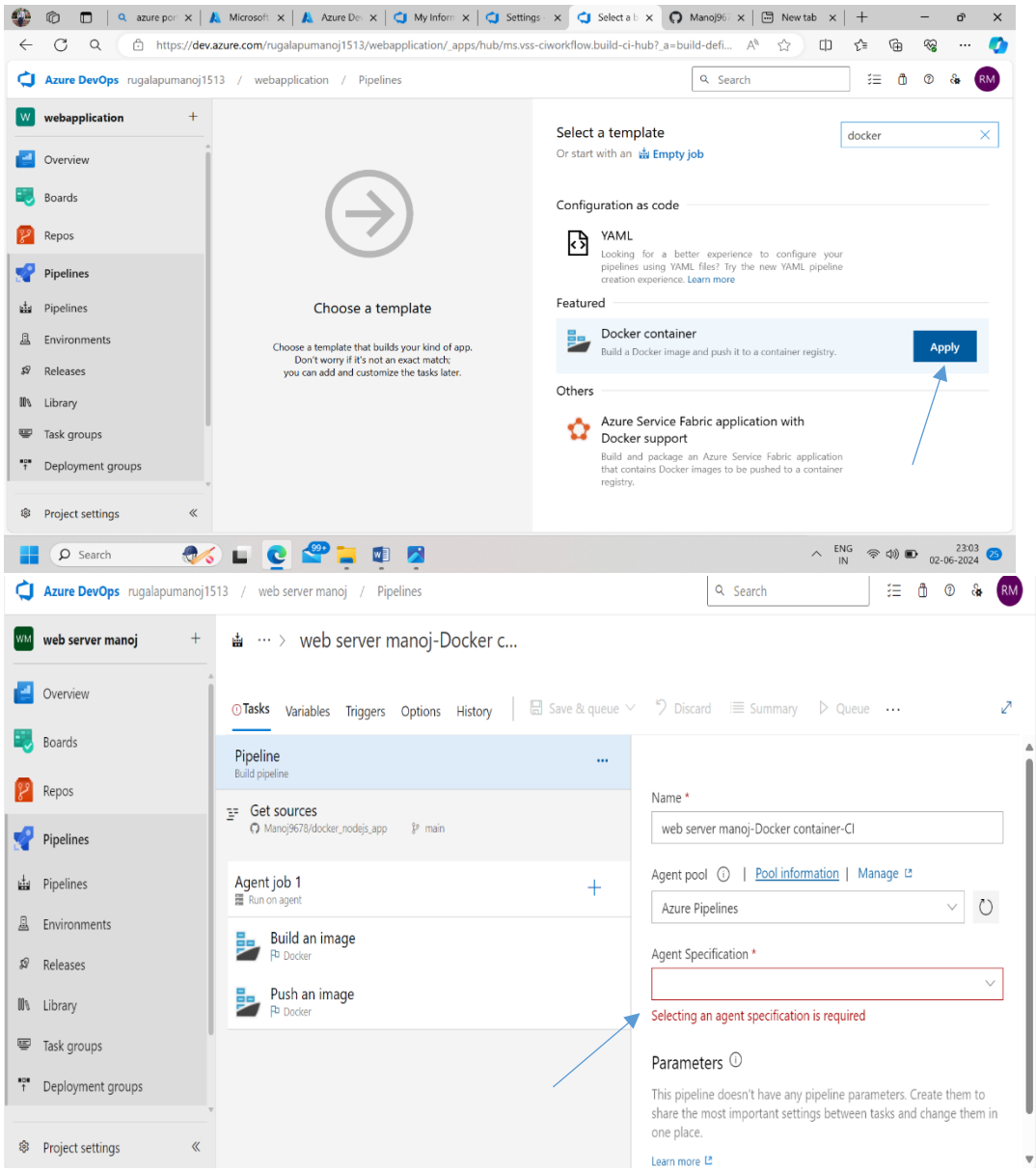
- Create a pipeline .



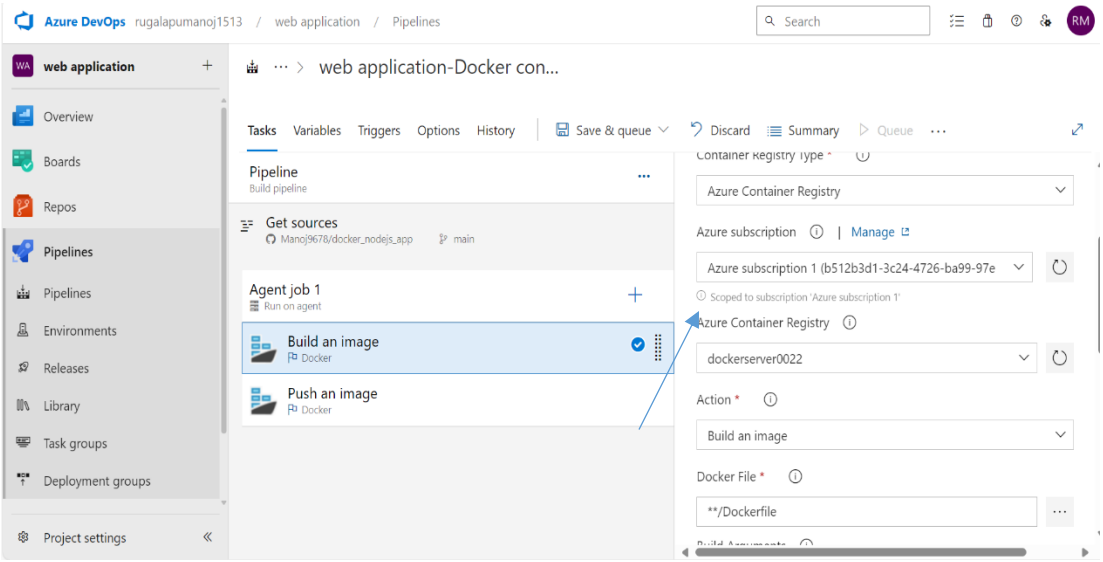
- Add a Github authrozsied with pipeline.



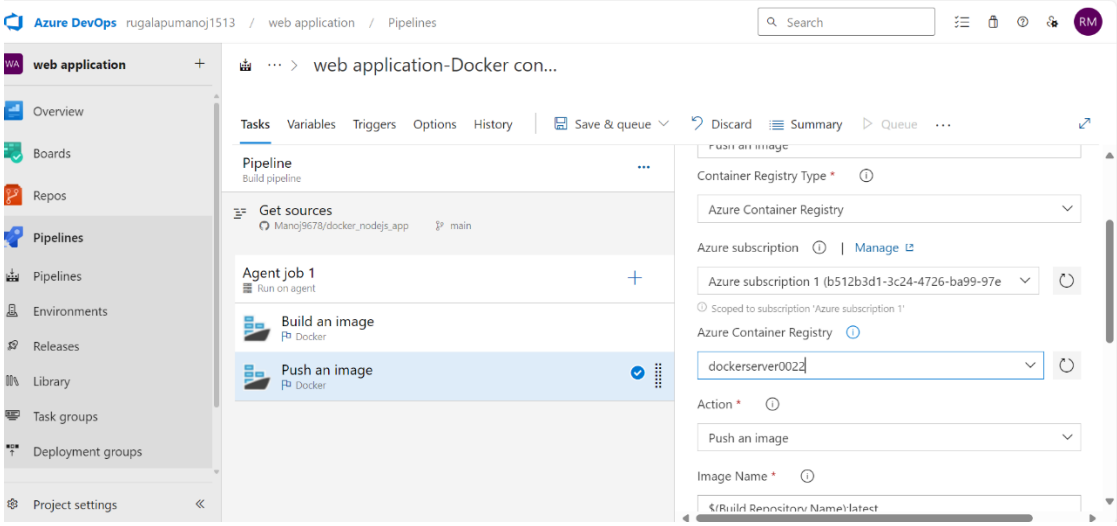
- Then we add Docker container Apply it.



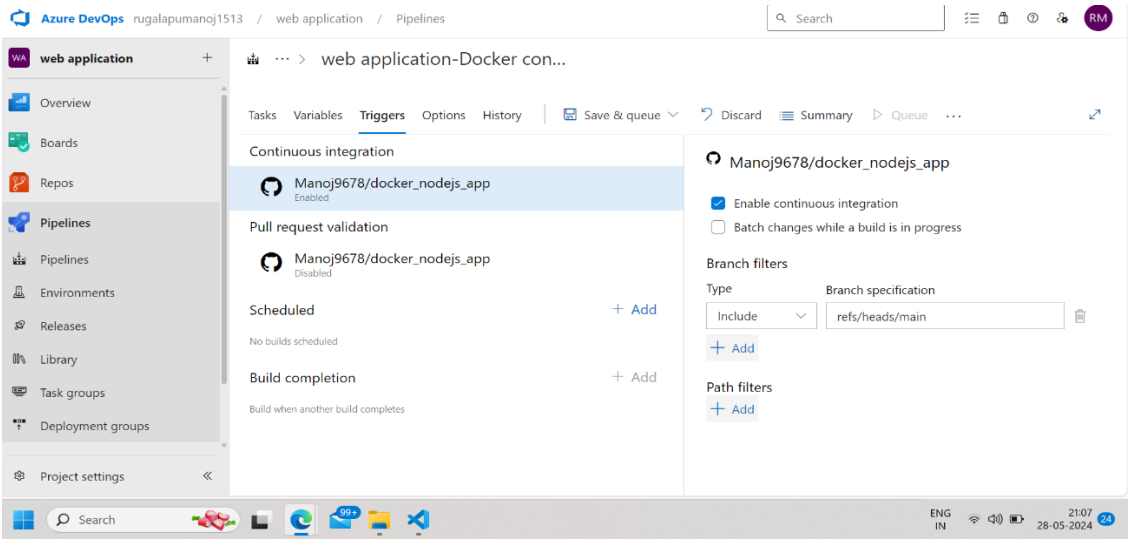
- Build an image requirements .
- Authorzise for subscription.



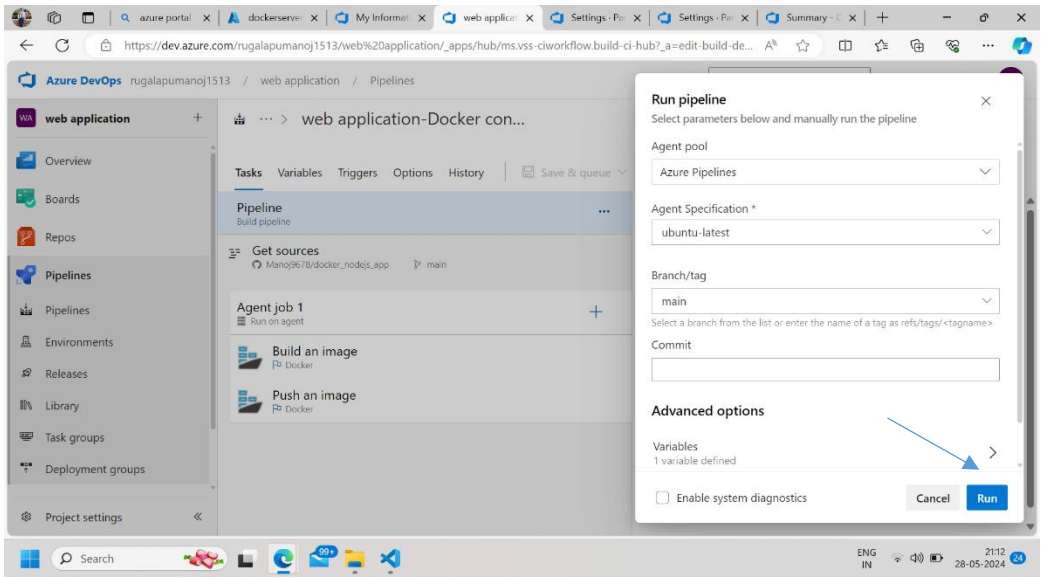
- push an image requirements .
- Authorzise for subscription.

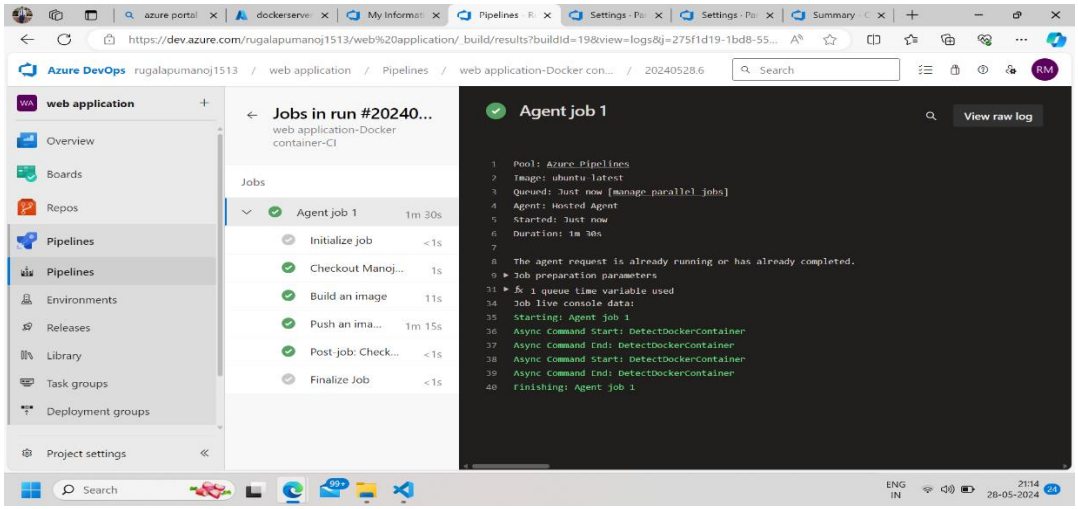


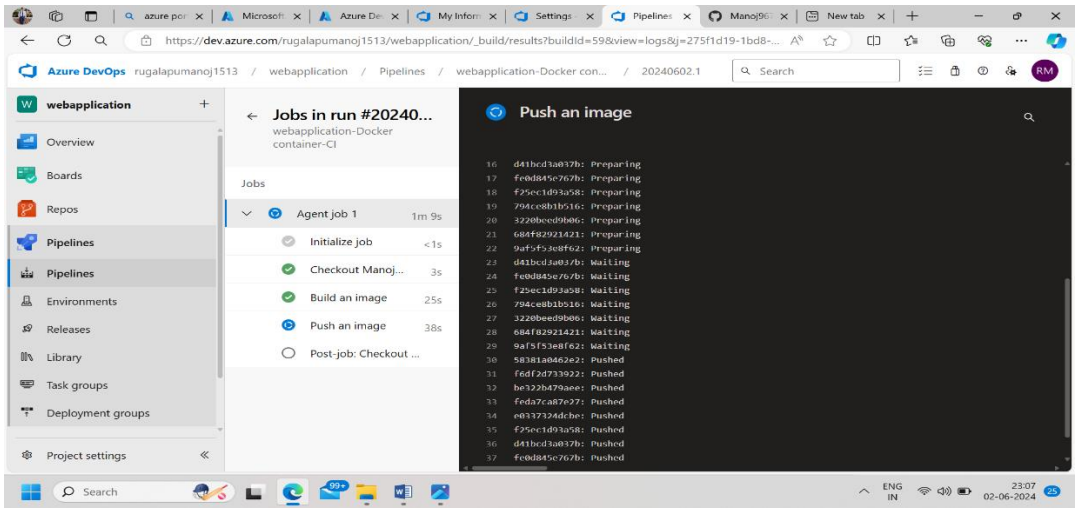
- Go to triggers Then enable it.



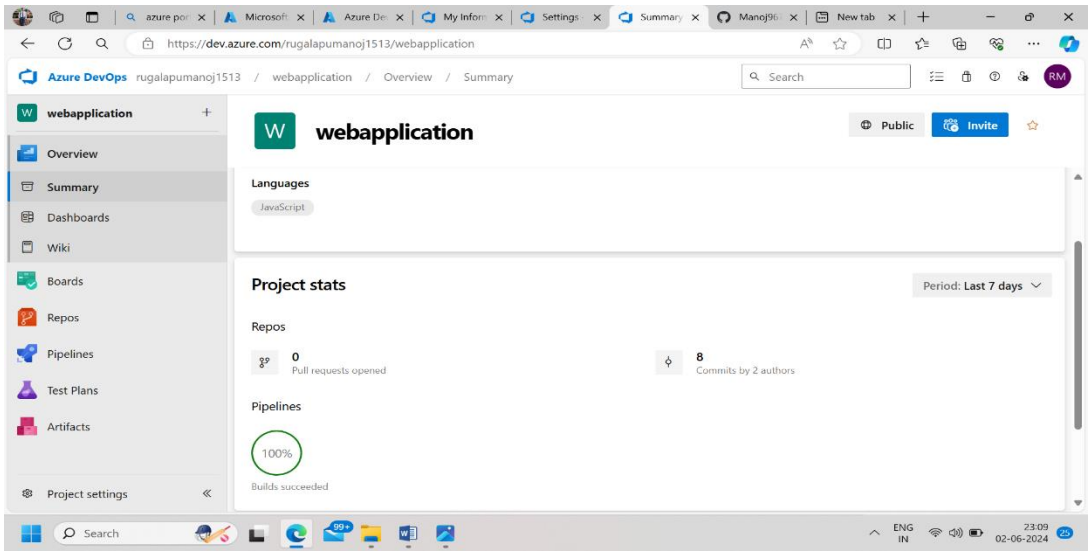
- Save it .
- Let’s we run it.



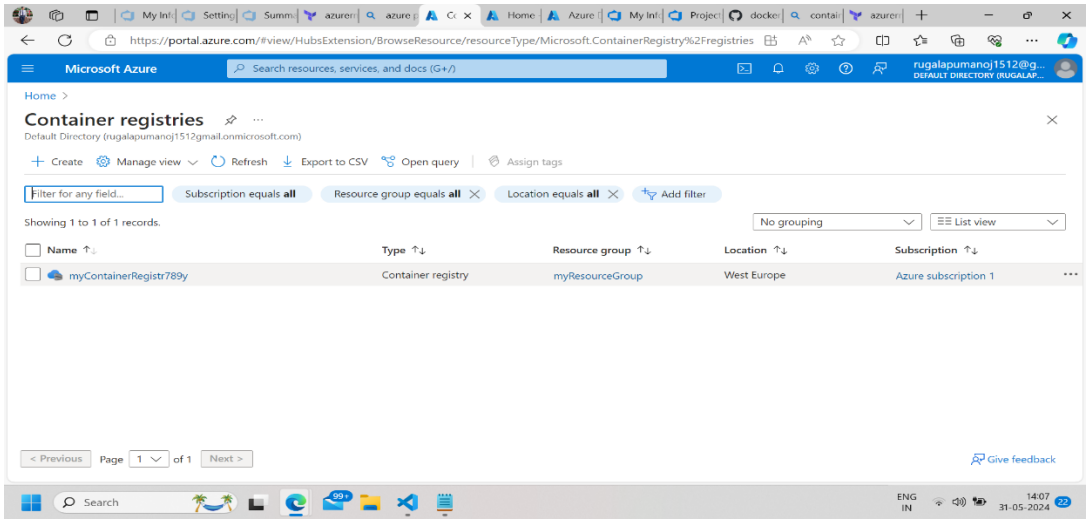


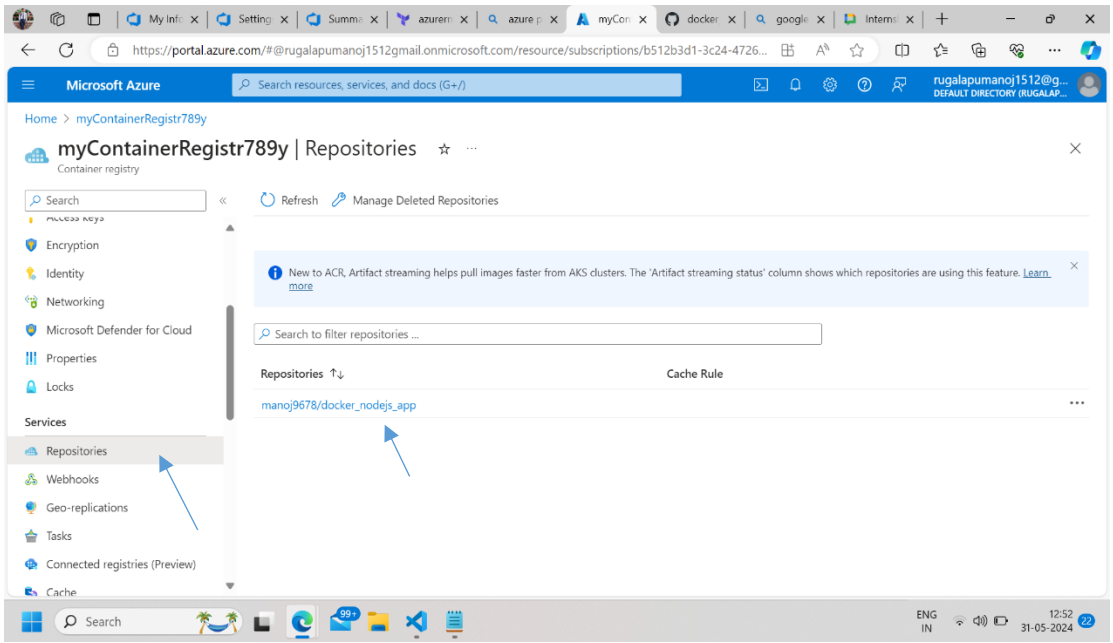


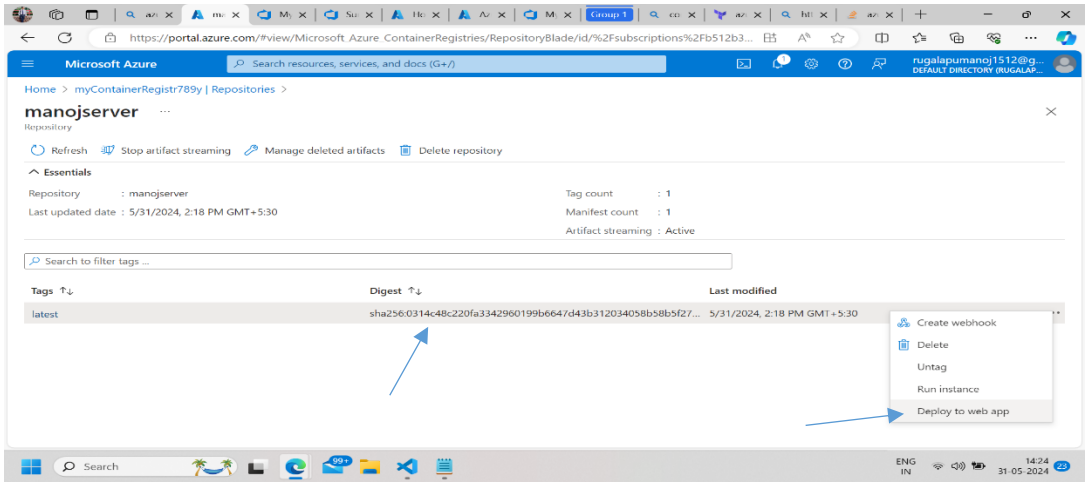
- We Created a pipeline then we checked it in repository.



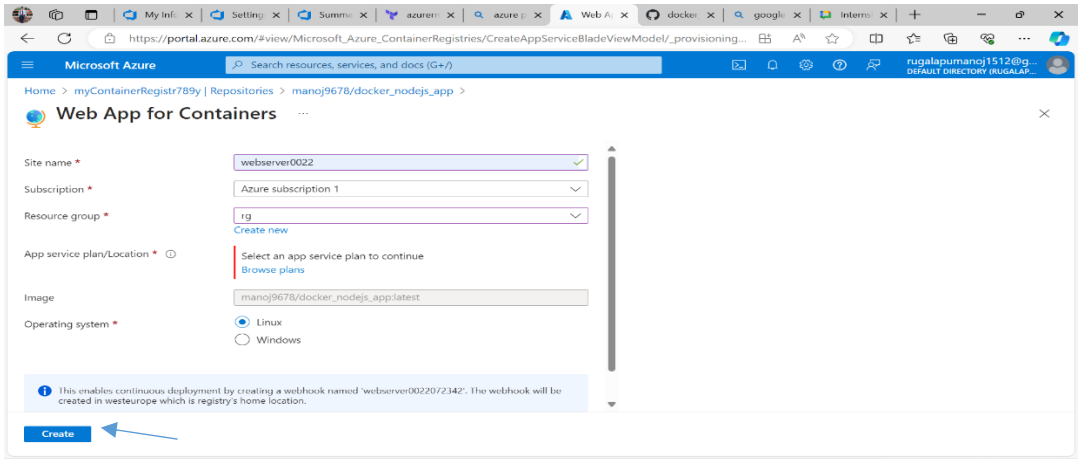
- I am created a image shown in a container.
- Let's we check it.

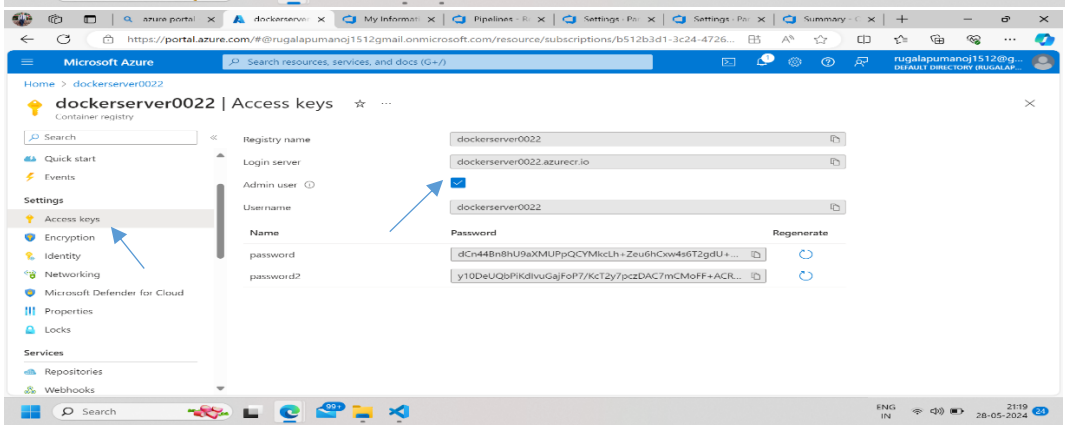


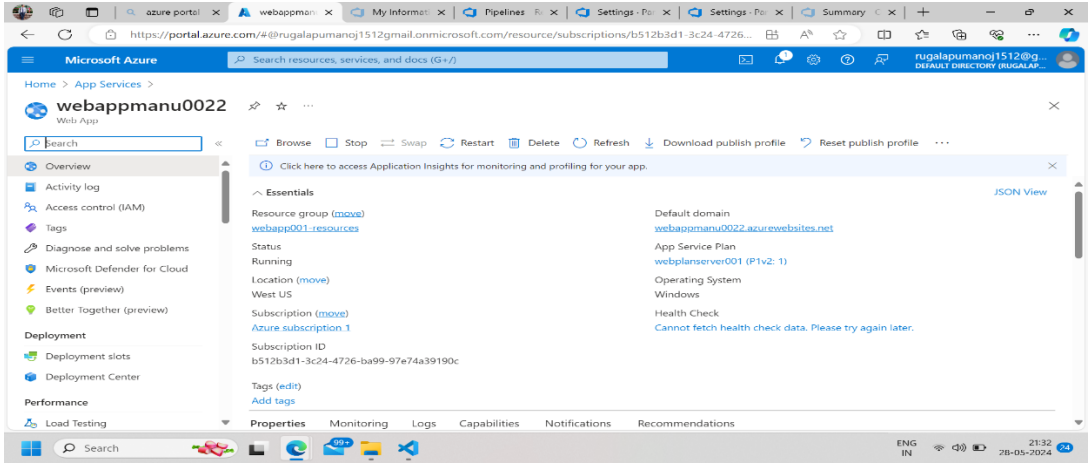




- We Deploy to web app . Then create it.

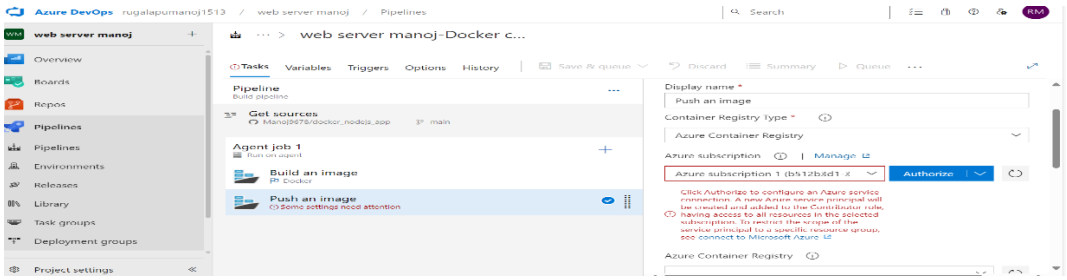




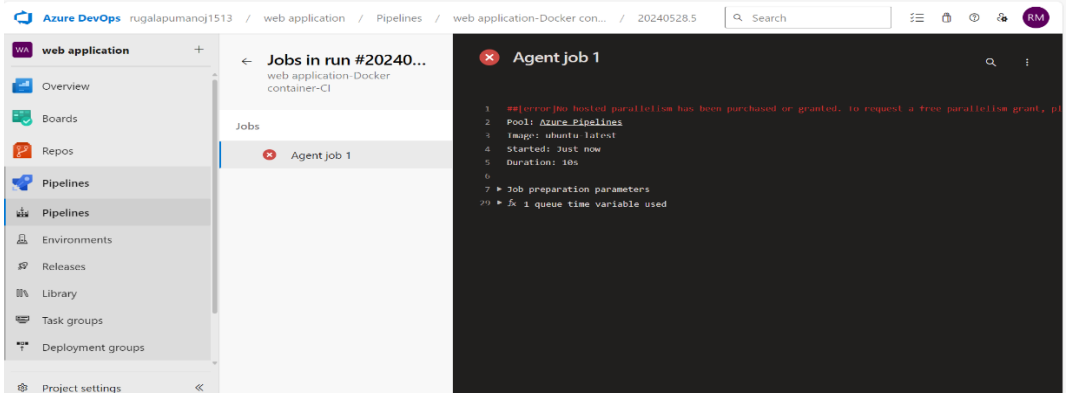


Troubleshooting Section

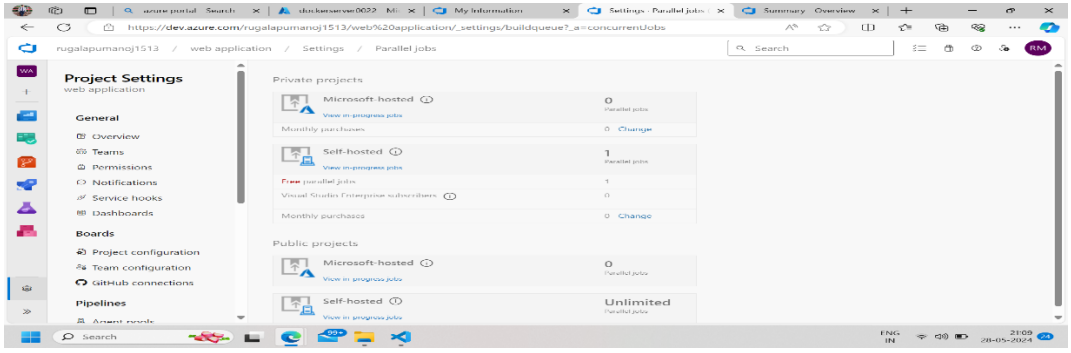
- Authorzised problem in pipeline.



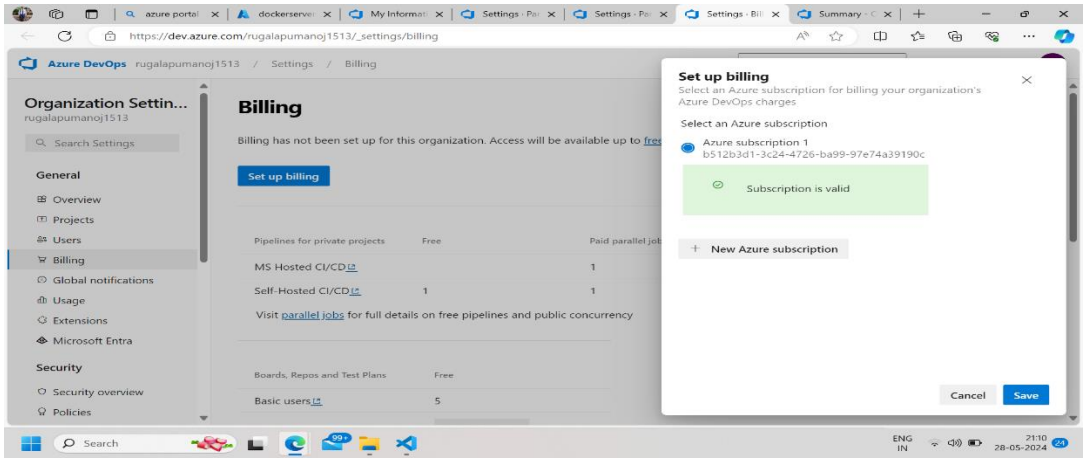
- We correct it using same subscription ,same container server name.
- Problem in parallelism host.



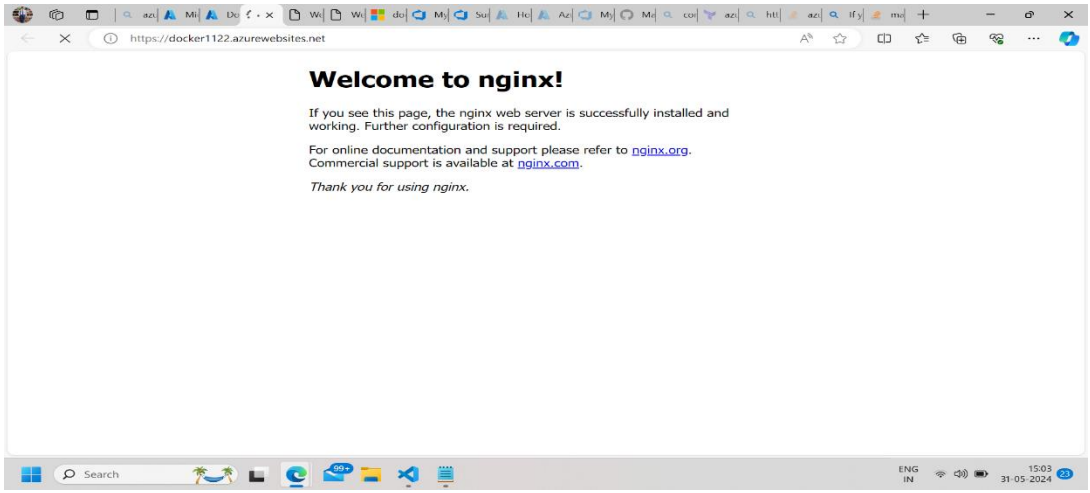
- Goto settings then click it Microsoft host change



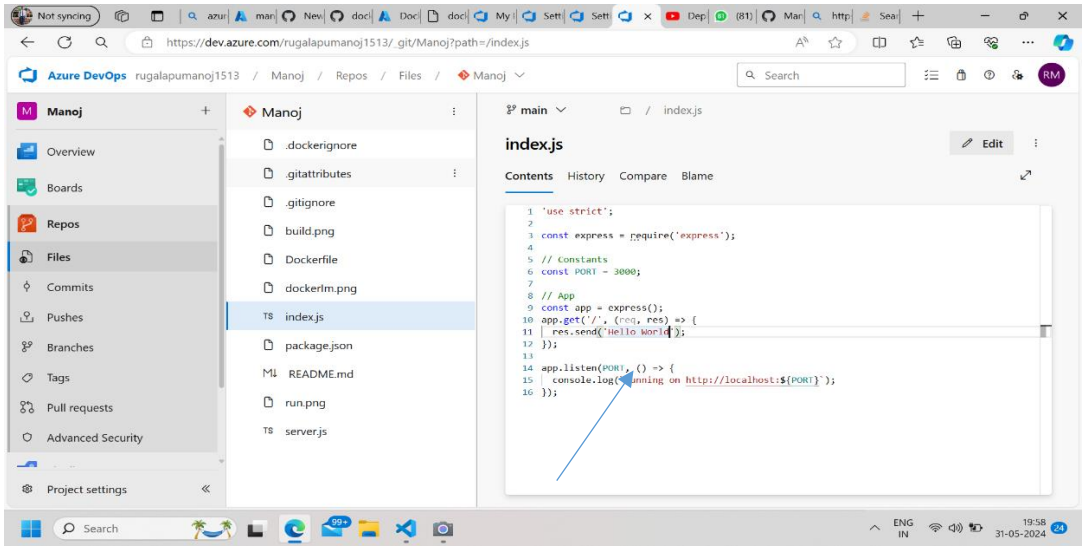
- Set up billing and save it .



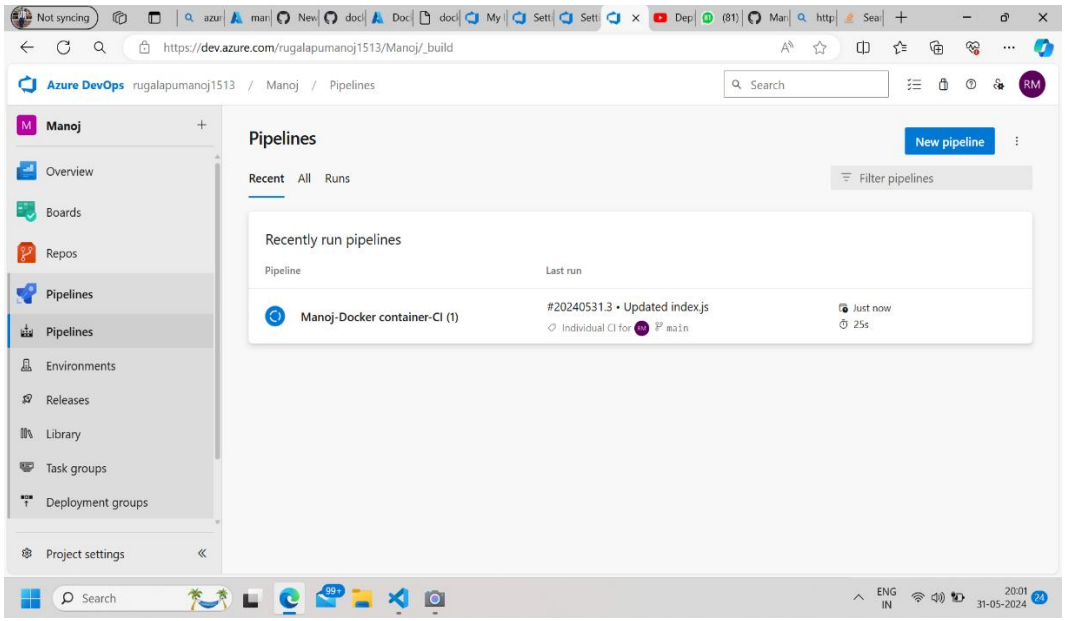
OUTPUT'S

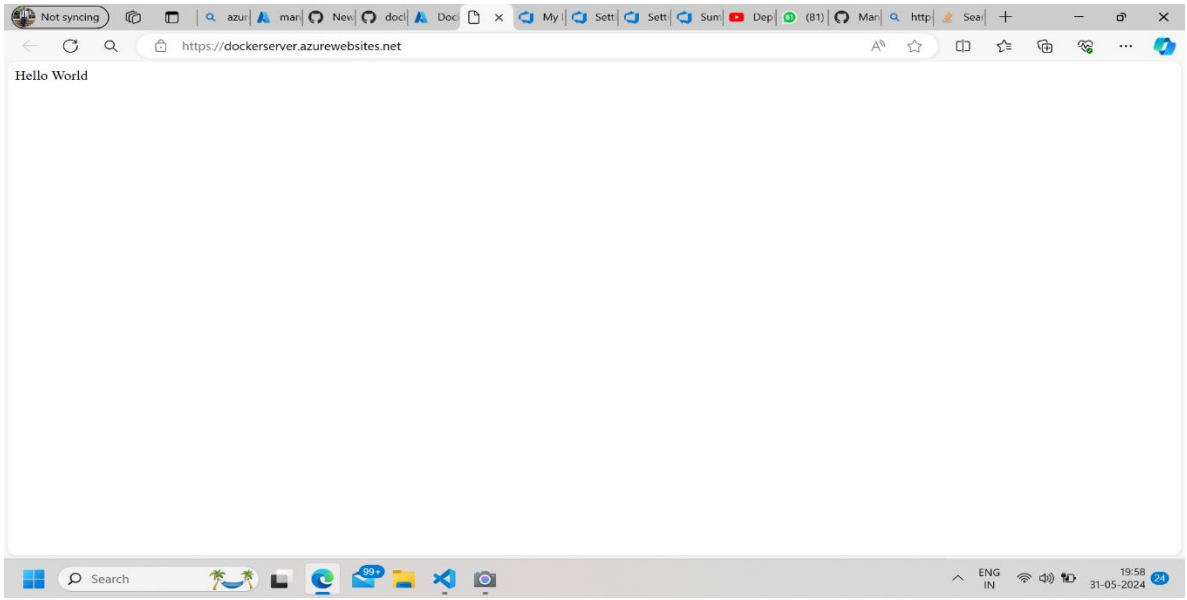


- It's first output
- Then we Automate it any change in scripts or write code.



- Edit write it Next word save it.





Summary's

- It is Implemented a simple web application, containerize it using Docker, set up infrastructure using Terraform, and use Git for version control. I will have a fully automated DevOps pipeline that allows you to develop, deploy, and manage your web application, check out the outputs it fully automated.

- It is implemented Deployed a Web Application with DevOps Tools.