**Azure DevOps & Terraform Project: Automating Static Web App Deployment**

**Project Overview**

This project demonstrates the automation of Azure Static Web App deployment using **Azure DevOps Pipelines** and **Terraform**. The Terraform state file is stored in an **Azure Storage Account Container**, and existing resources are imported into Terraform's state management.

**Objectives**

* Automate **Static Web App** deployment with **Terraform** and **Azure DevOps**.
* Store Terraform state files in an **Azure Storage Account Container**.
* Import existing Azure resources into Terraform state management.
* Utilize **Azure DevOps Pipelines** for CI/CD automation.

**Infrastructure Setup**

**Resources Created**

* **Azure Resource Group** (terraformrgforWA)
* **Azure Storage Account** (sathustaccount)
* **Storage Container** (containerst for Terraform state files)
* **Static Web App** (my-static-web-app)

**Terraform Configuration**

**1. Backend Configuration (backend.tf)**

terraform {

backend "azurerm" {

resource\_group\_name = "resourcegroupweb"

storage\_account\_name = "sathustaccount"

container\_name = "containerst"

key = "terraform.tfstate"

}

}

**2. Provider Configuration (provider.tf)**

terraform {

  required\_version = ">=1.0"

  required\_providers {

    azurerm = {

      source  = "hashicorp/azurerm"

      version = ">=3.0.0"  # Upgrade to latest stable version

    }

  }

}

provider "azurerm" {

  features {}

}

**3. Resource Definitions (main.tf)**

resource "azurerm\_resource\_group" "Webapprg" {

name = var.resource\_group\_name

location = var.location

}

resource "azurerm\_static\_web\_app" "WebApp" {

name = var.webapp\_name

location = var.location

resource\_group\_name = azurerm\_resource\_group.Webapprg.name  
   sku\_tier            = "Free”

sku\_size            = "Free"

}

**4. Variables File (variables.tf & var.tfvars)**

variable "resource\_group\_name" {

description = "Resource Group Name"

type = string

}

variable "location" {

description = "Deployment Location"

type = string

}

variable "webapp\_name" {

description = "Static Web App Name"

type = string

}

**var.tfvars:**

location = "centralus"

resource\_group\_name = "terraformrgforWA"

webapp\_name = "my-static-web-app"

**Azure DevOps Pipeline Configuration**

**Pipeline YAML File (azure-pipelines.yml)**

# Starter pipeline

# Start with a minimal pipeline that you can customize to build and deploy your code.

# Add steps that build, run tests, deploy, and more:

# https://aka.ms/yaml

trigger:

- main

pool:

  vmImage: ubuntu-latest

variables:

- group: "static\_Web\_App"

steps:

- script: echo Hello, world!

  displayName: 'Run a one-line script'

- script: |

    echo Add other tasks to build, test, and deploy your project.

    echo See https://aka.ms/yaml

  displayName: 'Run a multi-line script'

- task: ms-devlabs.custom-terraform-tasks.custom-terraform-installer-task.TerraformInstaller@0

  inputs:

    terraformVersion: '1.11.1'

- task: AzureCLI@2

  inputs:

    azureSubscription: 'Mine\_DevOps'

    scriptType: 'bash'

    scriptLocation: 'inlineScript'

    inlineScript: |

            # Create resource group

            az group create --name $(terraformstoragerg) --location "East US"

            # Create storage account

            az storage account create --name $(terraformstorageaccount) --resource-group $(terraformstoragerg) --sku Standard\_LRS --location "East US"

            # Create storage container (Fix: added --account-name)

            az storage container create --name $(terraformcontainer) --account-name $(terraformstorageaccount) --auth-mode login

    addSpnToEnvironment: true

- task: AzureCLI@2

  inputs:

    azureSubscription: 'Mine\_DevOps'

    scriptType: 'bash'

    scriptLocation: 'inlineScript'

    inlineScript: |

      key=$(az storage account keys list --g $terraformstoragerg --name $terraformstorageaccount --query '[0].value' --output tsv)

      echo "##vso[task.setvariable variable=storagekey]$key"

- task: TerraformTaskV4@4

  inputs:

    provider: 'azurerm'

    command: 'init'

    backendServiceArm: 'Mine\_DevOps'

    backendAzureRmResourceGroupName: '$(terraformstoragerg)'

    backendAzureRmStorageAccountName: '$(terraformstorageaccount)'

    backendAzureRmContainerName: '$(terraformcontainer)'

    backendAzureRmKey: 'terraform.tfstate'

- task: TerraformTaskV4@4

  inputs:

    provider: 'azurerm'

    command: 'plan'

    commandOptions: '-var-file=var.tfvars'

    environmentServiceNameAzureRM: 'Mine\_DevOps'

- task: TerraformTaskV4@4

  inputs:

    provider: 'azurerm'

    command: 'apply'

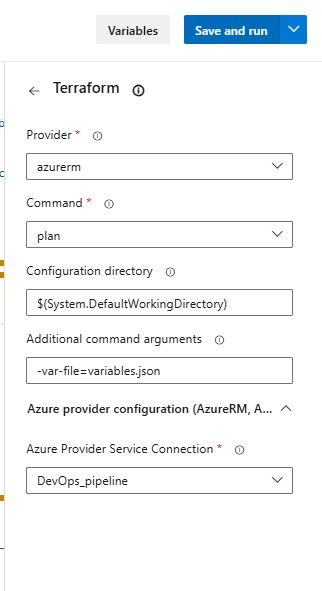
    commandOptions: '-var-file=var.tfvars -auto-approve'

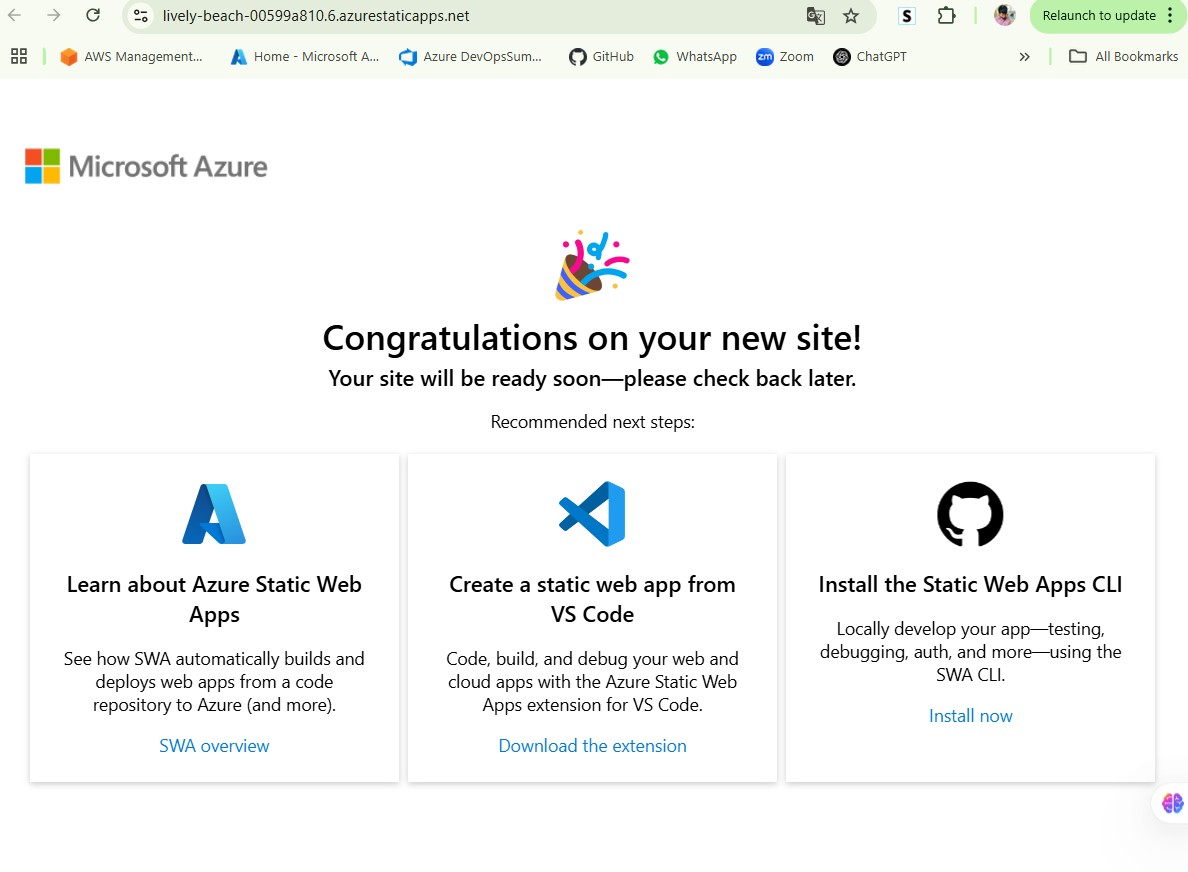
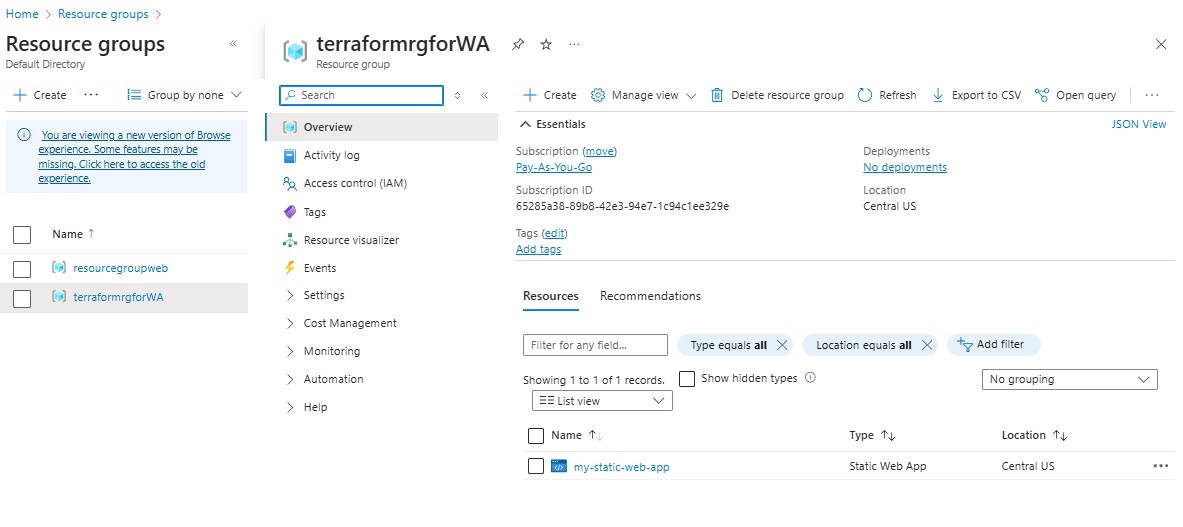
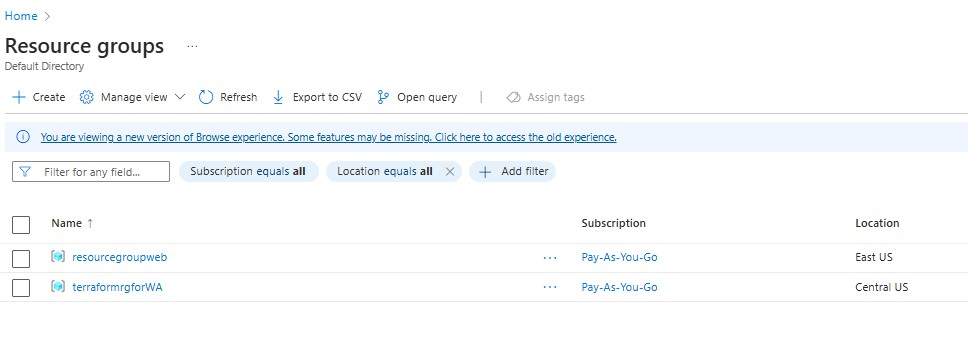
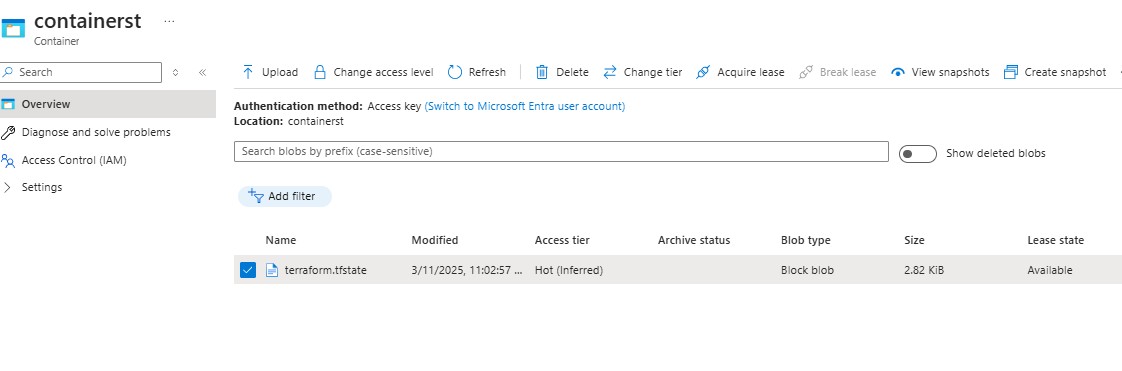
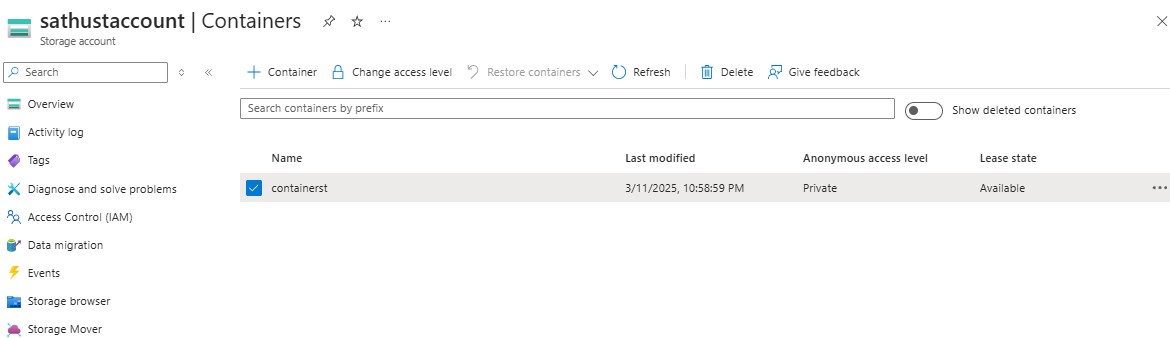
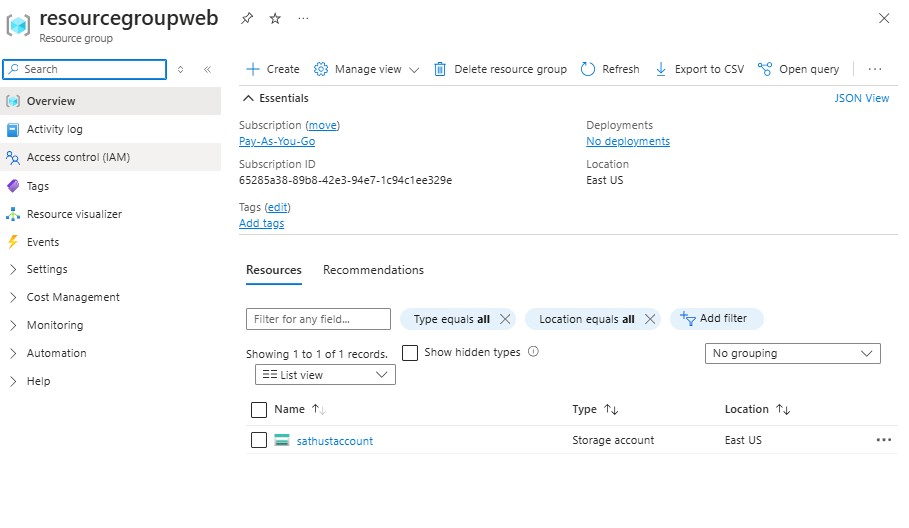
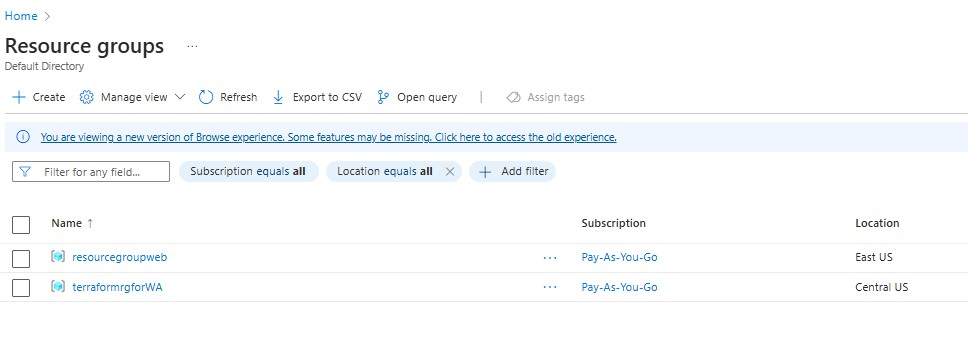
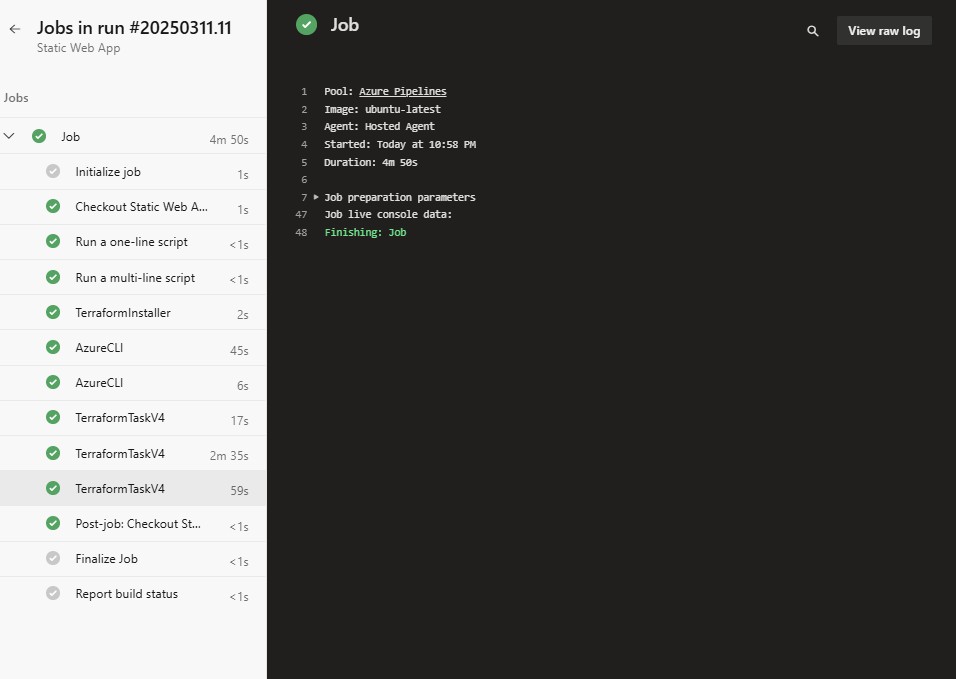
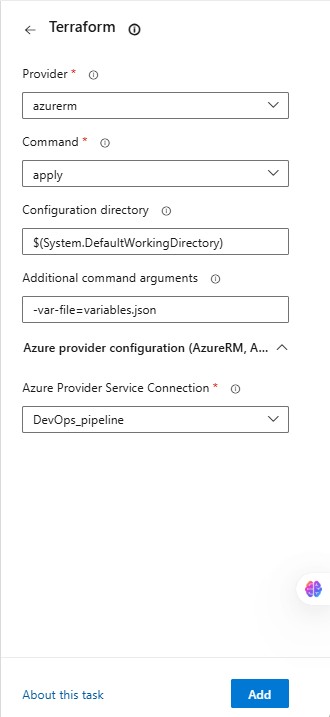
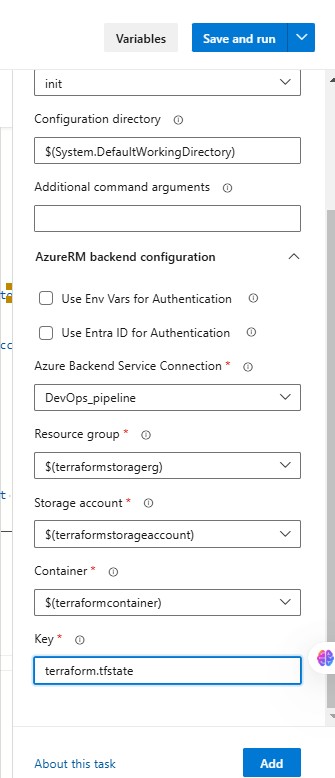
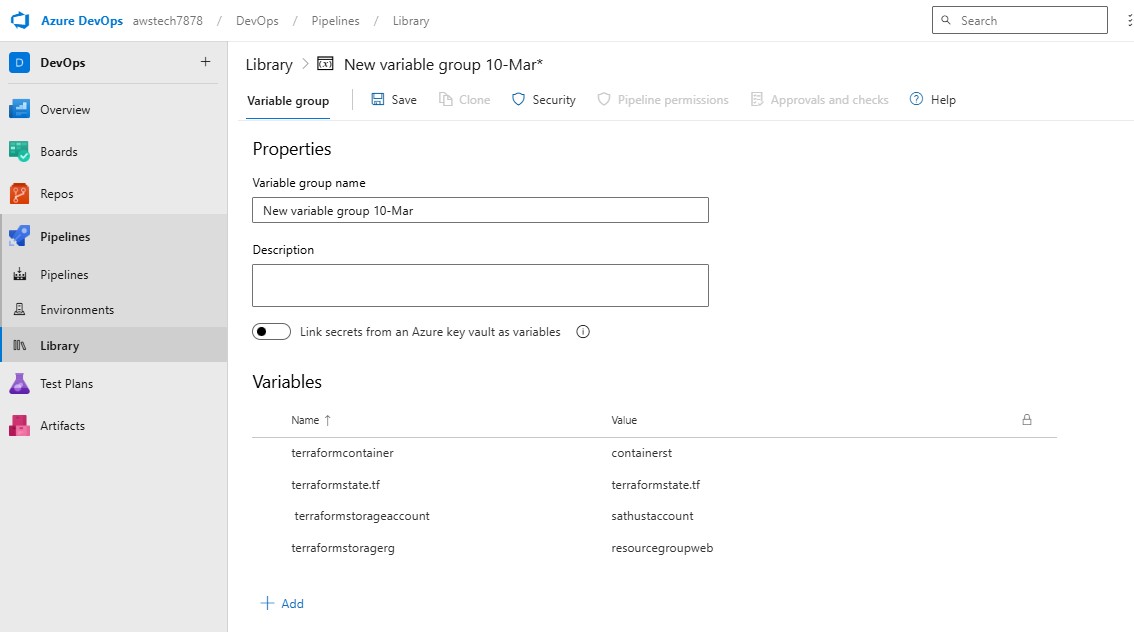
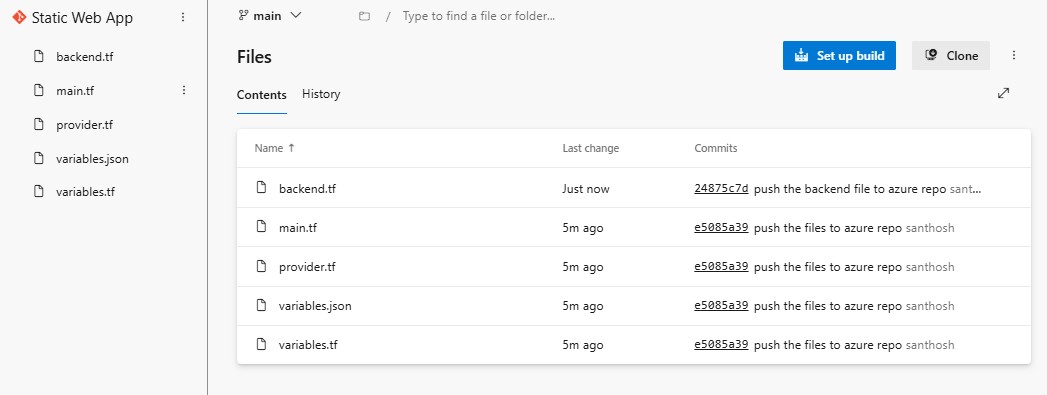
    environmentServiceNameAzureRM: 'Mine\_DevOps'

**Project Execution and Results**

* The **Azure DevOps Pipeline** successfully initializes Terraform, creates the resources, and applies configurations.
* The **Terraform state file** (terraform.tfstate) is successfully stored in **Azure Storage Container**.
* The **Static Web App** (my-static-web-app) is deployed successfully and accessible via Azure Static Web Apps URL.

**Screenshots & Evidence**

1. **Azure DevOps Pipeline Configuration & Execution**
2. **Terraform State File Stored in Azure Storage**
3. **Azure Resource Group & Static Web App Created**
4. **Successfully Hosted Static Web App Page  
     
     
     
   **



**Conclusion**

This project successfully automated the deployment of **Azure Static Web Apps** using **Terraform and Azure DevOps Pipelines**. The Terraform state file was securely stored in **Azure Storage**, and existing resources were efficiently managed via Terraform state imports. This approach streamlines **CI/CD** for Azure cloud resources while maintaining infrastructure as code.