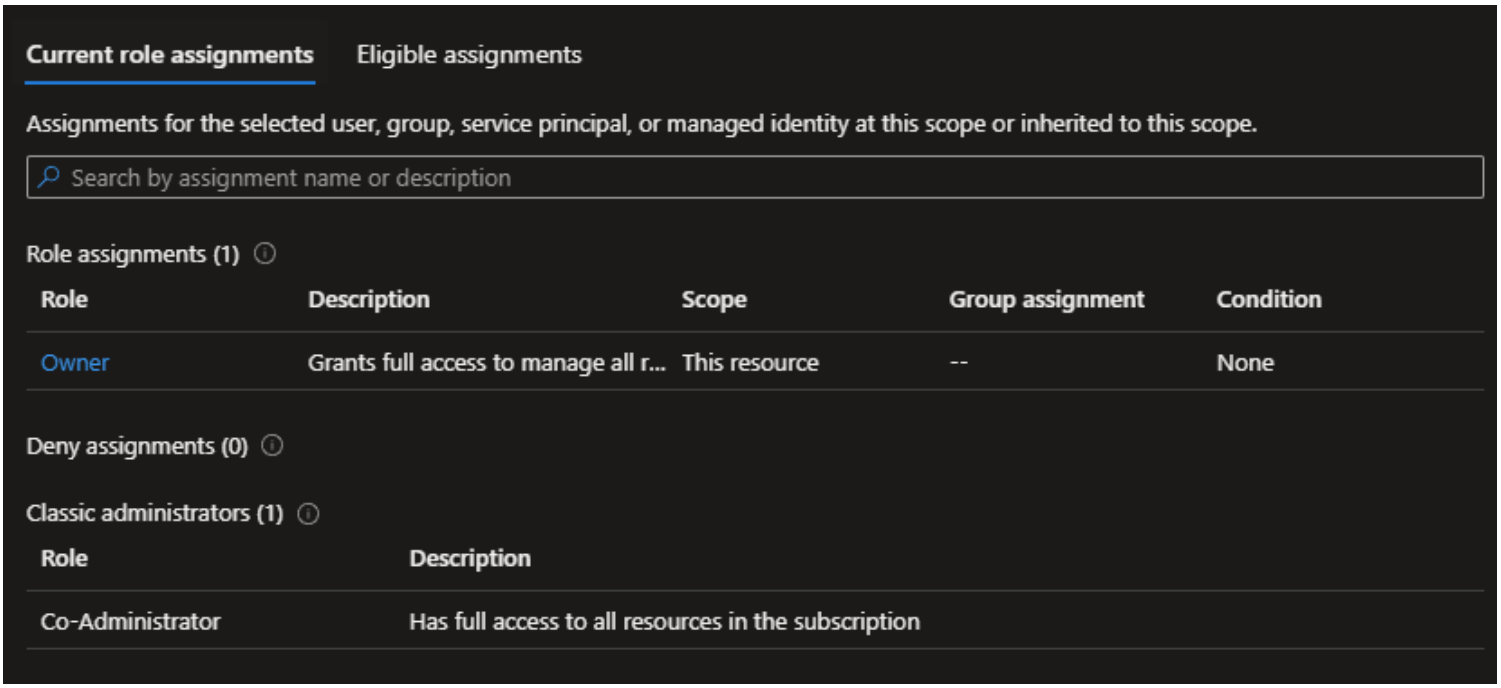
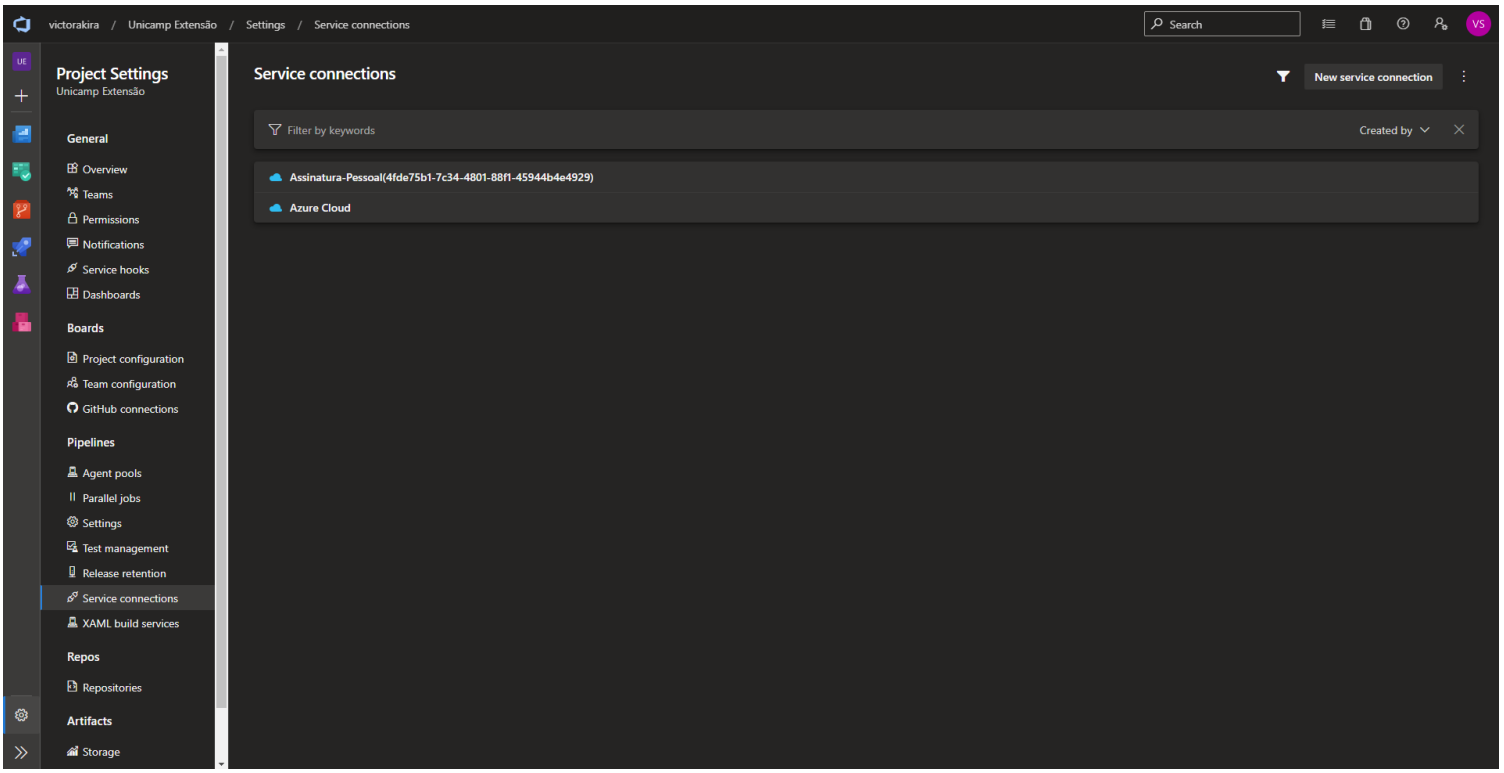
Grupo
Evandro Luis Araujo de Sousa
Leonardo Rener de Oliveira
Victor Akira Hassuda Silva

Azure - https://dev.azure.com/victorakira/Unicamp%20Extens%C3%A3o/

Pre-Config
No painel do Azure Cloud é necessário atribuir uma Role de Co-Administrador à Conta da Assinatura que deseja linkar com o Azure Pipeline



1o passo - Criar um Service Connection com o Azure Resource Manager



2o passo - Criar uma pipeline usando YAML na raiz do repositório

```yaml
# Nome da Pipeline
name: $(TeamProject)_$(Build.DefinitionName)_$(SourceBranchName)_$(Date:yyyyMMdd)$(Rev:.rrr)

# Trigger da Pipeline
trigger:
- main

# Recursos a serem utilizados pela Pipeline
resources:
 repositories:
   - repository: mainRepository
     type: git
     name: inf0997-trabalhofinal

# Definição dos Stages
stages:

# Primeiro estágio - Deploy da Infraestrutura
- stage: Deploy_Infrastructure
  displayName: Deploy Infrastructure
```

```yaml
# Utilizando a pool do Ubuntu disponibilizada pela Azure
pool:
  vmImage: ubuntu-latest

# Definição dos jobs da pipeline
jobs:

# Execução do Terraform para criar a infraestrutura
- job:
  displayName: Run Terraform

  steps:
  - checkout: self
    clean: true
    displayName: Checkout repository

  - bash: |
      ls -la
      ls -la terraform
    displayName: Run Scripts

  # Execução do comando Terraform Init para inicializar o Terraform
  # Baixa os comandos do provider na máquina
  - task: AzureCLI@2
    displayName: Init Terraform
    inputs:
      azureSubscription: 'Assinatura-Pessoal(4fde75b1-7c34-4801-88f1-45944b4e4929)'
      scriptType: 'bash'
      scriptLocation: 'inlineScript'
      inlineScript: |
        terraform init
      addSpnToEnvironment: true
      workingDirectory: 'terraform'

  # Execução do planejamento do Terraform
  # Cria toda a estrutura de dependencias para criar a infraestrutura
  - task: AzureCLI@2
    displayName: Plan Terraform
    inputs:
      azureSubscription: 'Assinatura-Pessoal(4fde75b1-7c34-4801-88f1-45944b4e4929)'
```

```
        scriptType: 'bash'
        scriptLocation: 'inlineScript'
        inlineScript: |
          terraform plan -input=false -out deploy.tfplan
        addSpnToEnvironment: true
        workingDirectory: 'terraform'

    # Execução do Terraform apply para criar a infraestrutura no Azure Cloud
    - task: AzureCLI@2
      displayName: Apply Terraform
      inputs:
        azureSubscription: 'Assinatura-Pessoal(4fde75b1-7c34-4801-88f1-45944b4e4929)'
        scriptType: 'bash'
        scriptLocation: 'inlineScript'
        inlineScript: |
          terraform apply -input=false -auto-approve deploy.tfplan
        addSpnToEnvironment: true
        workingDirectory: 'terraform'

# Segundo estágio - Deploy do Web App
# Depende da finalização com sucesso do estágio anterior
- stage: Deploy_WebApp
  displayName: Deploy WebApp
  dependsOn: Deploy_Infrastructure
  condition: succeeded()

  # Utiliza uma vm Windows Server 2019 para executar
  pool:
    vmImage: windows-2019

  jobs:
  - job:
    displayName: 'Deploy Sample Web App'

    # Cria um manifest file para poder instalar outras tools
    steps:
    - task: DotNetCoreCLI@2
      displayName: Dotnet new manifest
      inputs:
        command: 'custom'
        custom: 'new'
```

```
      arguments: 'tool-manifest'
      workingDirectory: 'SampleWebApp/SampleWebApp'

  # Instala a tool do Entity Framework
  - task: DotNetCoreCLI@2
    displayName: Install dotnet-ef
    inputs:
      command: 'custom'
      custom: 'tool'
      arguments: 'install --global dotnet-ef'
      workingDirectory: 'SampleWebApp/SampleWebApp'

  # Executa comando Update-Database do Entity Framework
  # Para criação das tabelas do banco de dados
  - task: DotNetCoreCLI@2
    displayName: Entity Framework Update Database
    inputs:
      command: 'custom'
      custom: 'ef'
      arguments: 'database update'
      workingDirectory: 'SampleWebApp/SampleWebApp'

  # Compila o código em modo Release
  - task: DotNetCoreCLI@2
    displayName: Build Release
    inputs:
      command: 'custom'
      custom: 'build'
      arguments: '--configuration Release'
      workingDirectory: 'SampleWebApp/SampleWebApp'

  # Realiza o Publish da aplicação em modo Release
  - task: DotNetCoreCLI@2
    displayName: Publish Release
    inputs:
      command: 'publish'
      publishWebProjects: true
      arguments: '-f net6.0 -c Release'
      workingDirectory: 'SampleWebApp'

  - powershell: |
```

```
        dir
        dir SampleWebApp/
        dir SampleWebApp/SampleWebApp
      displayName: List Directories

    # Realiza o deploy da Web App no Azure Web Services
    - task: AzureWebApp@1
      displayName: Publish Web App
      inputs:
        azureSubscription: 'Assinatura-Pessoal(4fde75b1-7c34-4801-88f1-45944b4e4929)'
        appType: 'webApp'
        appName: 'inf0997-sample'
        package: '$(System.DefaultWorkingDirectory)/**/*.zip'
        deploymentMethod: 'auto'
```

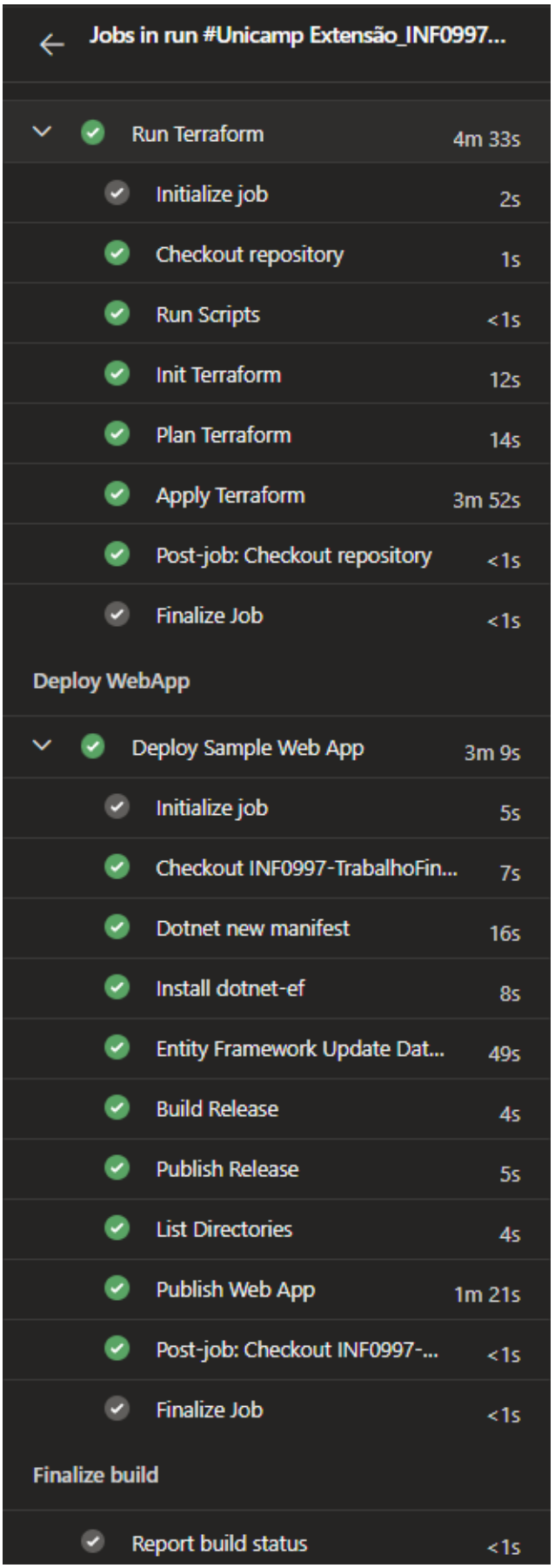
3o passo - Adicionar um arquivo contendo a configuração a ser usada pelo Terraform

```
# Providers necessários para executar o script do Terraform
terraform {
    required_providers {
        azurerm = {
            source = "hashicorp/azurerm"
            version = "3.33.0"
        }
    }
}

# Configuração do provider AzureRM
provider "azurerm" {
  subscription_id = "4fde75b1-7c34-4801-88f1-45944b4e4929"
  client_id = "b228868c-f923-4356-8dbf-28d150bed674"
  client_secret = "u8x8Q~RpjGCxuqTNFzxV5naSPbDPSL7fjjki1bzC"
  tenant_id = "179b5913-afc3-448e-9350-455b204c0296"
  features {

  }
}

# Criação de um Resource Group no Azure Cloud
```

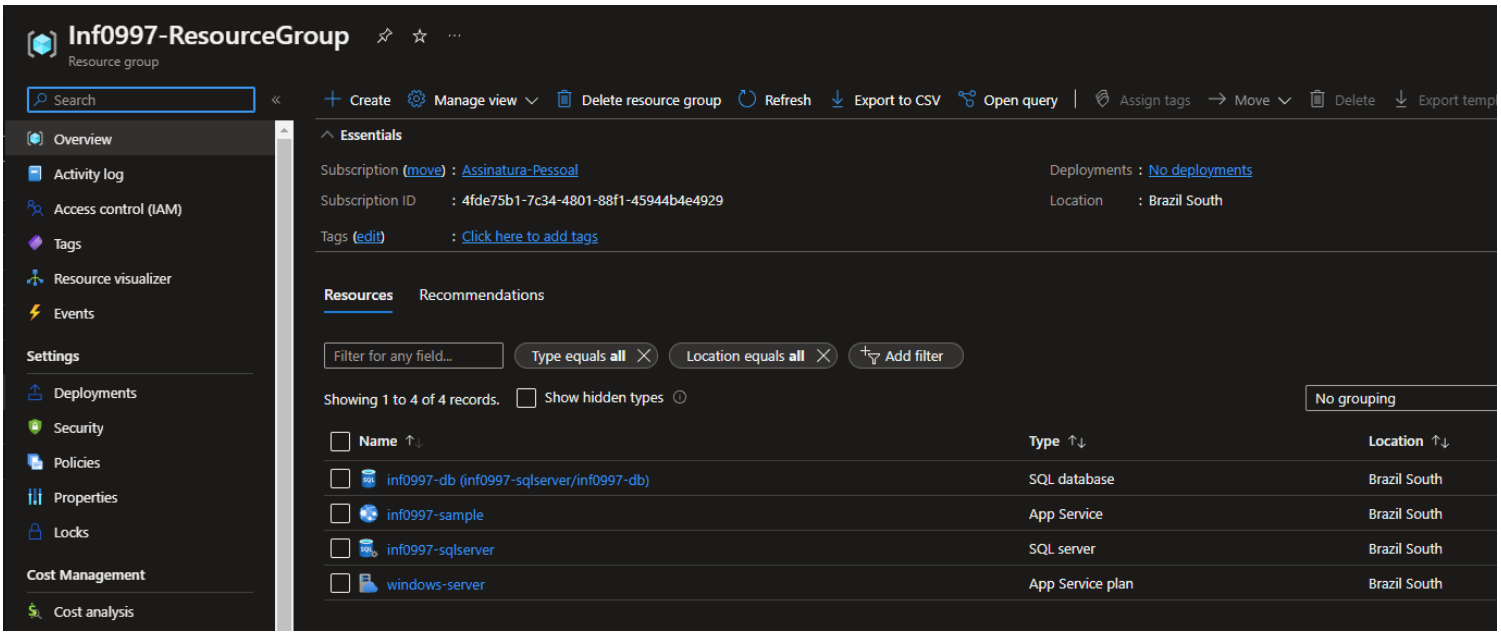
```
resource "azurerm_resource_group" "azure-devops" {
  name = "Inf0997-ResourceGroup"
  location = "Brazil South"
}

# Criação de um servidor para o Banco de Dados SQL Server
resource "azurerm_mssql_server" "azure-sql-server" {
  name = "inf0997-sqlserver"
  resource_group_name = azurerm_resource_group.azure-devops.name
  location = azurerm_resource_group.azure-devops.location
  version = "12.0"
  administrator_login = "inf0997Admin"
  administrator_login_password = "A1k9i9r1aAdmin!"
}

# Configuração do Banco de Dados SQL Server
resource "azurerm_mssql_database" "azure-database" {
  name = "inf0997-db"
  server_id = azurerm_mssql_server.azure-sql-server.id
  collation = "SQL_Latin1_General_CP1_CS_AS"
  max_size_gb = 2
  sku_name = "Basic"
}

# Adição de regras de Firewall do Banco de Dados
resource "azurerm_mssql_firewall_rule" "azure-db-firewall" {
  name             = "inf0997-firewall"
  server_id        = azurerm_mssql_server.azure-sql-server.id
  start_ip_address = "0.0.0.0"
  end_ip_address   = "255.255.255.255"
}



# Adição de um Service Plan para o Windows Server
resource "azurerm_service_plan" "azure-service-plan" {
  name                = "windows-server"
  resource_group_name = azurerm_resource_group.azure-devops.name
  location            = azurerm_resource_group.azure-devops.location
  os_type             = "Windows"
  sku_name            = "B1"
```

```
}

# Criação de um Azure Web App Windows para hospedar um Web App
resource "azurerm_windows_web_app" "azure-webapp-windows" {
  name                = "inf0997-sample"
  resource_group_name = azurerm_resource_group.azure-devops.name
  location            = azurerm_service_plan.azure-service-plan.location
  service_plan_id     = azurerm_service_plan.azure-service-plan.id

  site_config {
    application_stack {
      current_stack = "dotnet"
      dotnet_version = "v6.0"
    }
  }

  identity {
    type = "SystemAssigned"
  }

}
```

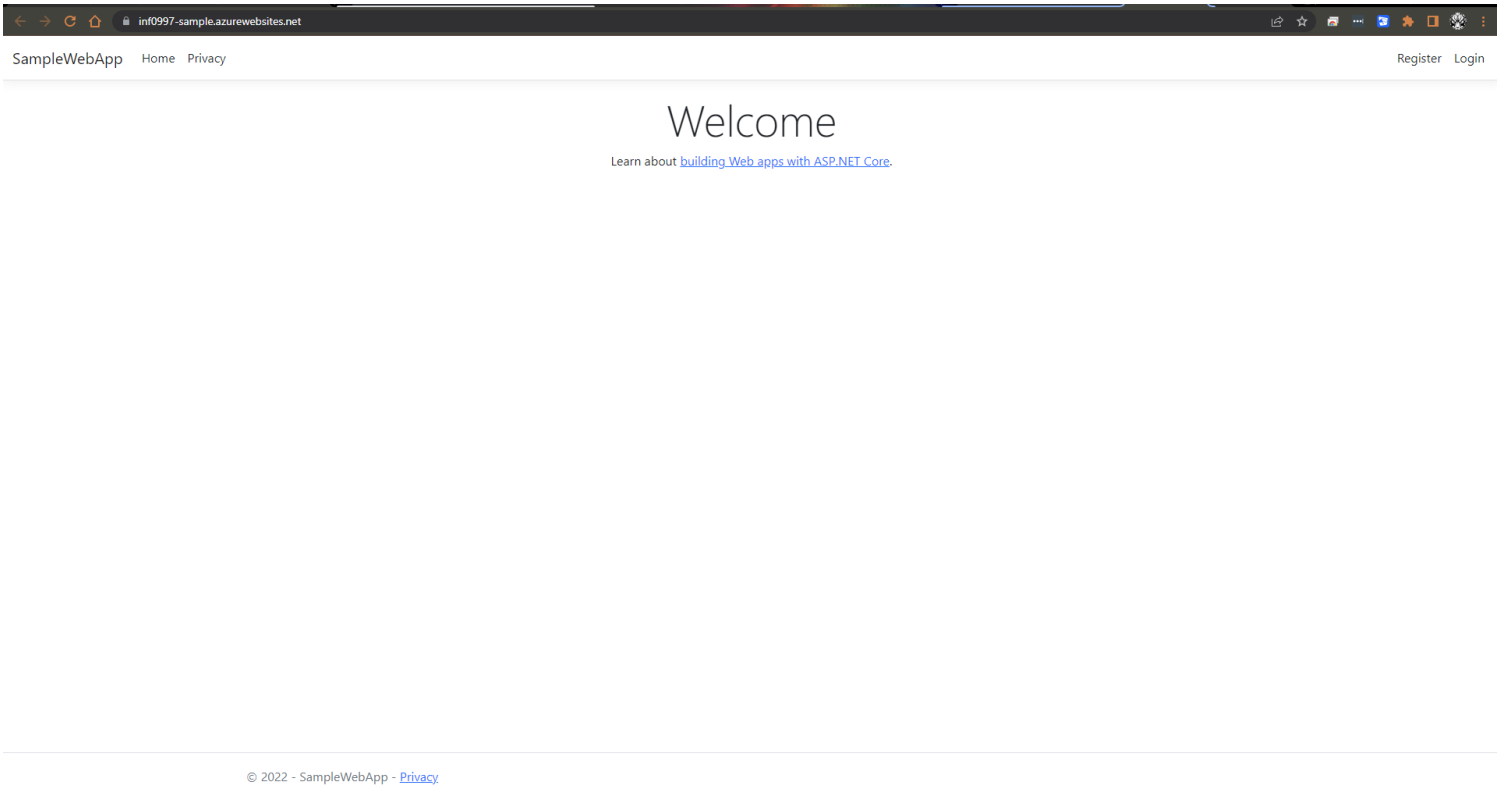
4o passo - Executar o Pipeline ou commitar na Main para execução automática



5o passo - Verificar a execução e criação dos serviços

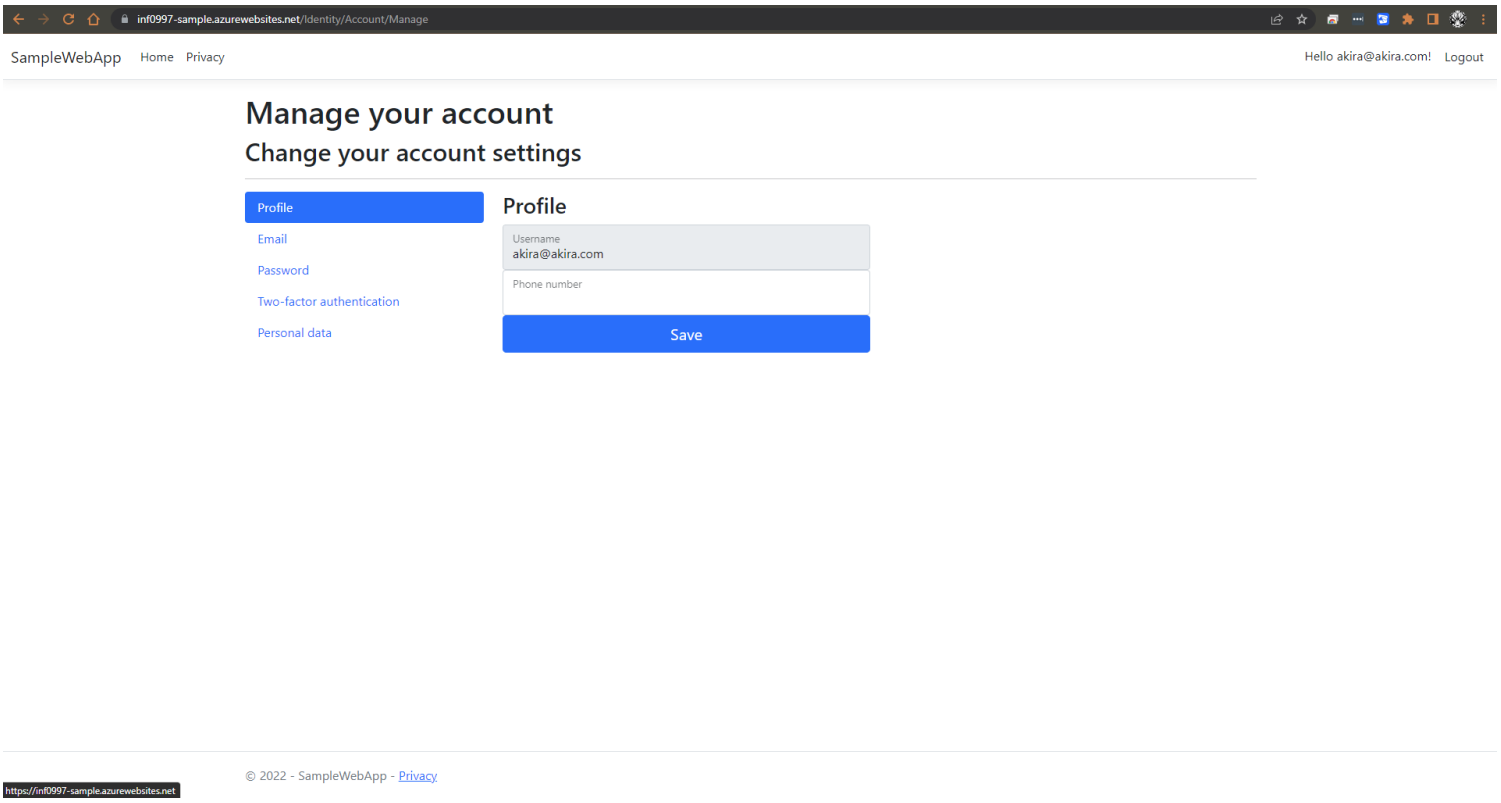


6o passo - Verificar o deploy da Web App



7o passo - Verificar a conexão da Web App com o Banco de Dados

- Criado um usuário com email akira@akira.com
- Acessando a página restrita após realizar o login



8o passo - Verificar a criação das tabelas e usuários no Banco de Dados através do SQL Server Management Studio