**Assignment 2 \_**

***Azure Infrastructure-as-Code***

A logo of a letter

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| 2024 - 2025 | Rik Dekkers | 2CCS01 |
|  | Cloud Platforms |  |

Contents

[1. Introduction 3](#_Toc194191114)

[2. Azure Design Diagram 4](#_Toc194191115)

[3. Docker 5](#_Toc194191116)

[3.1 Docker Image 5](#_Toc194191117)

[3.2 Docker Prepare Image 5](#_Toc194191118)

[4. Manual Setup 6](#_Toc194191119)

[4.1 Resource Group 6](#_Toc194191120)

[4.2 Container Registry 7](#_Toc194191121)

[4.2.1 Create Registry 7](#_Toc194191122)

[4.2.2 Add Image to registry 11](#_Toc194191123)

[4.2.3 Enable the Admin account for deployment 13](#_Toc194191124)

[4.3 Deploy a container instance 14](#_Toc194191125)

[5. Basic Infrastructure as Code (IaC) 19](#_Toc194191126)

[5.1 Prerequisites 19](#_Toc194191127)

[5.1.1 Setup Azure CLI on windows 19](#_Toc194191128)

[5.1.2 Installing Bicep 19](#_Toc194191129)

[5.2 Deploying with Bicep 20](#_Toc194191130)

[5.2.1 Defining Parameters 20](#_Toc194191131)

[5.2.2 Defining Resources 20](#_Toc194191132)

[5.2.3 Create Resource Group 25](#_Toc194191133)

[5.2.4 Validate Bicep 26](#_Toc194191134)

[5.2.5 Deploy the templates 26](#_Toc194191135)

[6. Infrastructure as Code (IaC) with best practices 27](#_Toc194191136)

# 1. Introduction

In this assignment we will deploy a flask application in a container. We will use Azure to host and monitor the application. We will try to ensure security while remaining accessibility.

We will first do this manually and show the process of doing so using a few screenshots. After the manual installation we will try to implement it all using IaC (Infrastructure as Code).

The infrastructure as code will be done using Bicep to automatically deploy to Azure. Bicep uses ARM templates and tries to make them more accessible.

ARM templates use JSON, which is mostly taken away in Bicep to make it more clear. Bicep first “compiles” into ARM templates, which are then send to the Azure Resource Manager.

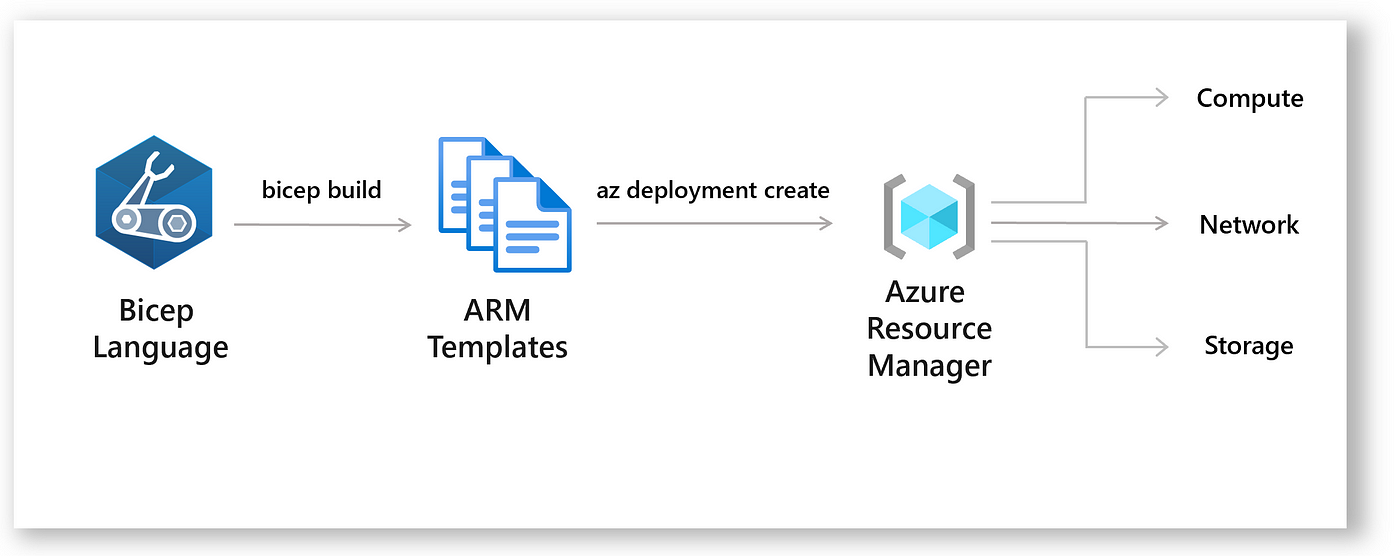


Diagram source: <https://medium.com/codex/arm-templates-or-azure-bicep-what-should-i-use-14e8662d3f27>

# 2. Azure Design Diagram

## 2.1 Simple Setup

A screenshot of a computer

AI-generated content may be incorrect.

## 2.2 Bicep – ARM Flow

A diagram of a process

AI-generated content may be incorrect.

## 2.3 Advanced setup

A screenshot of a computer

AI-generated content may be incorrect.

# 3. Docker

## 3.1 Docker Image

The Dockerfile needs some minor changes, the linux version has been changed.

|  |
| --- |
| FROM ubuntu:latest  # Update installed packages and install python3  RUN apt update && apt upgrade -y && \      apt install -y python3 python3-pip python3-venv  # Set the workdir to /app  WORKDIR /app  # Copy the application files  COPY ./example-flask-crud /app  # Create the virtual environment  RUN python3 -m venv venv  # Install requirements using the virtual environment's pip  RUN ./venv/bin/pip install -r requirements.txt  # Setup environment variables  ENV FLASK\_APP=crudapp.py  # Setup the database  RUN ./venv/bin/flask db init && \      ./venv/bin/flask db migrate -m "entries table" && \      ./venv/bin/flask db upgrade  # Expose flask default port  EXPOSE 80  # Run the application using the virtual environment's Flask  CMD ["./venv/bin/flask", "run", "--host=0.0.0.0", "--port", "80"] |

## 3.2 Docker Prepare Image

|  |
| --- |
| docker build -t cpf/assignment2 . |

# 4. Manual Setup

## 4.1 Resource Group

*Resource groups are used to group resources together.*

Navigate to <https://portal.azure.com/#browse/resourcegroups>

A screenshot of a computer

AI-generated content may be incorrect.

Click “Create”

Select the subscription, provide a name, select a region and click “Review + create”

* Subscription: Azure for students
* Name: r0993343-resource-group
* Region: (Europe) West Europe

A screenshot of a black and white screen

AI-generated content may be incorrect.

On the next page, review your settings and click “Create” again.

## 4.2 Container Registry

*In the container image registry, we will store our container image.*

### 4.2.1 Create Registry

Navigate back to the resource group overview and open our resource group

A screenshot of a computer

AI-generated content may be incorrect.

Click “Create”

A screenshot of a computer

AI-generated content may be incorrect.

Look for container registry and click “Create”

A screenshot of a computer

AI-generated content may be incorrect.

Select the subscription, our resource group. Set a name and a location. Set the pricing plan to “basic”.

A screenshot of a computer

AI-generated content may be incorrect.

*If you have a higher budget, selecting “Premium” will allow you to automatically implement Availability zones for High Availability purposes.*

Click “Next: Networking”

We leave it one Public access, as private access is premium only

A computer screen shot

AI-generated content may be incorrect.

Continue to Encryption and also leave it as is. Then continue to Tags.

No tags are required for now, leave it blank and continue to “Review + create”

A screenshot of a computer

AI-generated content may be incorrect.

Our setup will be verified, to create the resource immediately, click “Create”.

We also notice “Download a template” which is an ARM template which can be used to deploy it automatically.

A screen shot of a computer

AI-generated content may be incorrect.

After the deployment is complete, we can click “Go to resource”

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Click “push an image” to get a guide on how to add your image to the registry”

### 4.2.2 Add Image to registry

**Requirements**

* Docker installed (already satisfied)
* Azure CLI installed

**Azure CLI**

|  |
| --- |
| curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash |

**Log in to Azure**

|  |
| --- |
| az login --use-device-code |

This command provides a link and a code. Open the link, fill in the code and login.

A shark with text overlay

AI-generated content may be incorrect.

**Log in to Registry**

|  |
| --- |
| az acr login --name r0993343ContainerRegistry |

A screen shot of a computer

AI-generated content may be incorrect.

**Tag the image**

|  |
| --- |
| docker tag cpf/assignment2:latest r0993343containerregistry.azurecr.io/cloud-platforms/assignment2:latest |

**Push the image to the registry**

|  |
| --- |
| docker push r0993343containerregistry.azurecr.io/cloud-platforms/assignment2:latest |

**View image**

Visit the Registry page

A screenshot of a computer screen

AI-generated content may be incorrect.

On the left, go to Services > Repositories, Open the repository

A screenshot of a computer

AI-generated content may be incorrect.

### 4.2.3 Enable the Admin account for deployment

Go to the registry page

On the left navigate to Settings > Access Keys and enable Admin User

A screenshot of a computer

AI-generated content may be incorrect.

## 4.3 Deploy a container instance

Navigate to the resource group and click “create”

A screenshot of a computer

AI-generated content may be incorrect.

Look for the module “Container Instances” and click “create”

A screenshot of a computer

AI-generated content may be incorrect.

Set basic information

A screenshot of a computer

AI-generated content may be incorrect.

Set the “Image Source” to “Azure Container Registry”

Select the registry, select the image and the tag

A screenshot of a computer

AI-generated content may be incorrect.

Click “Next” and leave Networking

A screenshot of a computer

AI-generated content may be incorrect.

Leave networking, advanced and tags as is

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer program

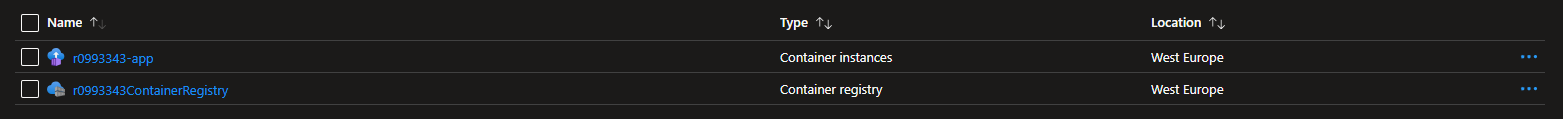
AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

Click “Create” after validation is done A screenshot of a computer

AI-generated content may be incorrect.

If you go back to the resource manager, you can see the deployed application



Open it and open the public IP address

A black rectangular object with white lines

AI-generated content may be incorrect.

We can see that the application is running

A screenshot of a computer

AI-generated content may be incorrect.

# 5. Basic Infrastructure as Code (IaC)

Implementing Bicep on the lowest level, without any “best practices”

## 5.1 Prerequisites

### 5.1.1 Setup Azure CLI on windows

|  |
| --- |
| winget install --exact --id Microsoft.AzureCLI |

|  |
| --- |
| az login --use-device-code |

### 5.1.2 Installing Bicep

|  |
| --- |
| az bicep install |

A screen shot of a computer

AI-generated content may be incorrect.

## 5.2 Deploying with Bicep

### 5.2. Defining Resources

#### 5.2.1.1 Resource types

Azure Repository: <https://github.com/Azure/bicep-types-az/tree/main>

/generated

* List of all possible types

#### 5.2.1.2 Container Registry

* File: ContainerRegistry.bicep

|  |
| --- |
| // Define parameters  param registry\_name string = 'br0993343containerregistry'  // Define resources  // Container Registry  resource Container\_Registry 'Microsoft.ContainerRegistry/registries@2024-11-01-preview' = {    name: registry\_name    location: 'westeurope'    sku: {      name: 'Basic'    }    properties: {      adminUserEnabled: true      publicNetworkAccess: 'Enabled'    }  } |

#### 5.2.1.3 Adding the image

|  |
| --- |
| az acr login --name br0993343containerregistry  docker tag cpf/assignment2:latest br0993343containerregistry.azurecr.io/cloud-platforms/as  signment2:latest  docker push br0993343containerregistry.azurecr.io/cloud-platforms/assignment2:latest |

#### 5.2.1.4 Container App

|  |
| --- |
| // Define parameters  @secure()  param registry\_username string // allow admin to insert username  @secure()  param registry\_password string // allow admin to insert password  param app\_name string = 'b-r0993343-app'  resource Container\_App 'Microsoft.ContainerInstance/containerGroups@2024-10-01-preview' = {    name: app\_name    location: 'westeurope'    zones: [      '2'    ]    properties: {      sku: 'Standard'      containers: [        {          name: app\_name          properties: {            image: 'br0993343containerregistry.azurecr.io/cloud-platforms/assignment2:latest'            ports: [              {                protocol: 'TCP'                port: 80              }            ]            environmentVariables: []            resources: {              requests: {                memoryInGB: json('1.5')                cpu: json('1')              }            }          }        }      ]      initContainers: []      imageRegistryCredentials: [        {          server: 'br0993343containerregistry.azurecr.io'          username: registry\_username          password: registry\_password        }      ]        restartPolicy: 'OnFailure'      ipAddress: {        ports: [          {            protocol: 'TCP'            port: 80          }        ]        type: 'Public'      }      osType: 'Linux'    }  } |

### 5.2.3 Create Resource Group

|  |
| --- |
| az group create --name b-r0993343-resource-group --location westeurope |

A screen shot of a computer

AI-generated content may be incorrect.

### 5.2.4 Validate Bicep

**Container registry**

|  |
| --- |
| az deployment group validate --resource-group b-r0993343-resource-group --template-file ContainerRegistry.bicep |

**Container Instance**

|  |
| --- |
| az deployment group validate --resource-group b-r0993343-resource-group --template-file .\DeployApp.bicep |

### 5.2.5 Deploy the templates

**Container registry**

|  |
| --- |
| az deployment group create --resource-group b-r0993343-resource-group --template-file ContainerRegistry.bicep |

**Container instance**

*Grab the admin username and password from registry > access keys and put them in the prompts*

|  |
| --- |
| az deployment group create --resource-group b-r0993343-resource-group --template-file .\DeployApp.bicep |

# 6. Infrastructure as Code (IaC) with best practices

## 6.1 New structure

We first deploy the registry, then after that everything else with main.bicep.

* main.bicep
  + DeployApp.bicep
  + Networking.bicep
  + Security.bicep
* ContainerRegistry.bicep

## 6.2 ContainerRegistry.bicep

With this file we deploy the container registry. It is separate as we need to also put the image in the registry from the CLI.

|  |
| --- |
| // Define parameters  param location string = 'westeurope'  param registry\_name string = 'br0993343containerregistry'  // Define resources  // Container Registry  resource Container\_Registry 'Microsoft.ContainerRegistry/registries@2024-11-01-preview' = {    name: registry\_name    location: location    sku: {      name: 'Basic' // Tier of service    }    properties: {      adminUserEnabled: true      publicNetworkAccess: 'Enabled'    }  }  output registry\_name string = Container\_Registry.name  output registry\_login\_server string = Container\_Registry.properties.loginServer |

*SKU = Stock keeping unit = version of offering of a resource*

## 6.3 main.bicep

In this file we define global parameters = parameters used everywhere, secure parameters for the username and password and module definitions.

|  |
| --- |
| // Static definitions  param location string = 'westeurope'  param registry\_login\_server string = 'br0993343containerregistry.azurecr.io'  // Prompt registry information  @secure()  param registry\_username string // allow admin to insert username  @secure()  param registry\_password string // allow admin to insert password  // Module definitions  module network 'Networking.bicep' = {    name: 'networkDeployment'    params: {      location: location    }  }  module security 'Security.bicep' = {    name: 'securityDeployment'    params: {      location: location      subnet\_id: network.outputs.subnet\_id    }  }  module container 'DeployApp.bicep' = {    name: 'containerDeployment'    params: {      registry\_username: registry\_username      registry\_password: registry\_password      location: location      subnet\_id: network.outputs.subnet\_id      log\_analytics\_id: network.outputs.log\_analytics\_id      registry\_login\_server: registry\_login\_server    }  } |

We pass parameters into the modules, these are predefined at top or outputted by the modules.

## 6.4 Networking.bicep

|  |
| --- |
| param location string  resource vnet 'Microsoft.Network/virtualNetworks@2023-11-01' = {    name: 'b-r0993343-vnet'    location: location    properties: {      addressSpace: {        addressPrefixes: ['10.0.0.0/16']      }      subnets: [        {          name: 'b-r0993343-subnet'          properties: {            addressPrefix: '10.0.1.0/24'            delegations: [              {                name: 'Microsoft.ContainerInstance/containerGroups'                properties: {                  serviceName: 'Microsoft.ContainerInstance/containerGroups'                }                type: 'Microsoft.Network/virtualNetworks/subnets/delegations'              }            ]          }        }      ]    }  }  resource publicIP 'Microsoft.Network/publicIPAddresses@2023-11-01' = {    name: 'b-r0993343-publicIP'    location: location    properties: {      publicIPAllocationMethod: 'Static'    }  }  resource logAnalytics 'Microsoft.OperationalInsights/workspaces@2023-09-01' = {    name: 'b-r0993343-logs'    location: location    properties: {}  }  output vnet object = vnet  output subnet\_id string = vnet.properties.subnets[0].id  output public\_ip\_id string = publicIP.id  output log\_analytics\_id string = logAnalytics.id |

We create a vNet (vnet is similar to VPC in AWS), containing the public subnet definition. It also has a delegation so it will work with the container application. The subnet fits in the address space of the vnet. We create a publicIP definition which has a static public ip used for the application or the subnet depending on the implementation.

## 6.5 Security.bicep

|  |
| --- |
| param location string  param subnet\_id string  resource nsg 'Microsoft.Network/networkSecurityGroups@2023-11-01' = {    name: 'b-r0993343-nsg'    location: location    properties: {      securityRules: [        { //Rule: allow http on inbound          name: 'AllowHTTP'          properties: {            priority: 100            direction: 'Inbound'            access: 'Allow'            protocol: 'Tcp'            sourcePortRange: '\*'            destinationPortRange: '80'            sourceAddressPrefix: '\*'            destinationAddressPrefix: '\*'          }        }, { //Rule: deny all (higher prio => less important => allow only http)          name: 'DenyAllInbound'          properties: {            priority: 200            direction: 'Inbound'            access: 'Deny'            protocol: '\*'            sourcePortRange: '\*'            destinationPortRange: '\*'            sourceAddressPrefix: '\*'            destinationAddressPrefix: '\*'          }        }      ]    }  }  output nsg\_id string = nsg.id |

We create a security group which has a rule to allow inbound http traffic and a denyall after it.

## 6.6 DeployApp.bicep

|  |
| --- |
| // Define parameters  // Secure  @secure()  param registry\_username string // allow admin to insert username  @secure()  param registry\_password string // allow admin to insert password  // Inherited  param location string  param subnet\_id string  param log\_analytics\_id string  param registry\_login\_server string  // Open  param app\_name string = 'b-r0993343-app'  param subscriptionId string = 'b2b2ae69-a262-4b30-b329-c51cd8d65d85'  resource Container\_App 'Microsoft.ContainerInstance/containerGroups@2024-10-01-preview' = {    name: app\_name    location: location    zones: [      '2'    ]    properties: {      sku: 'Standard'      containers: [        {          name: app\_name          properties: {            image: '${registry\_login\_server}/cloud-platforms/assignment2:latest'            ports: [              {                protocol: 'TCP'                port: 80              }            ]            environmentVariables: []            resources: {              requests: {                memoryInGB: json('1.5')                cpu: json('1')              }            }          }        }      ]      initContainers: []      imageRegistryCredentials: [        {          server: registry\_login\_server          username: registry\_username          password: registry\_password        }      ]        restartPolicy: 'OnFailure'      ipAddress: {        type: 'Private'  // Use Private IP type        ports: [          {            protocol: 'TCP'            port: 80          }        ]      }      subnetIds: [        {          id: subnet\_id        }      ]      osType: 'Linux'    }  } |

We prompt for the registry credentials, then we have a definition for the inherited parameters from main.bicep and finally we also have a few optional open parameters.

We define the resource, give it 2 availability zones. It pulls the image from our internal server using the credentials we put it. It has limited resources. It will restart on failure and is placed in the required private subnet.

## 6.7 Deploying

**Container registry**

|  |
| --- |
| az deployment group create --resource-group b-r0993343-resource-group --template-file ContainerRegistry.bicep |

**Everything else**

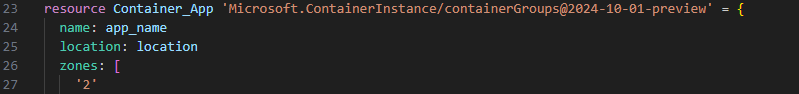
|  |
| --- |
| az deployment group create --resource-group b-r0993343-resource-group --template-file .\main.bicep |

A screenshot of a computer

AI-generated content may be incorrect.

# 7. Extras

In order to have high availability, we use two availability zones in our container deployment.



We therefor mirror it on two locations.

# 8. GitHub

<https://github.com/NanoSievert/cpf-assignment2>