Containers | ACI

Running your containers with Azure Container Instances

avanade

Prerequisites and more!

Install *Docker (https://docs.docker.com/get-started/#setup)*

Install *Git (https://git-scm.com/downloads)*

Install *Azure CLI (https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest)*

An *Azure subscription* (free account).

What you will do in this lab?

Download the souce code from git

Build the image from the Docker file

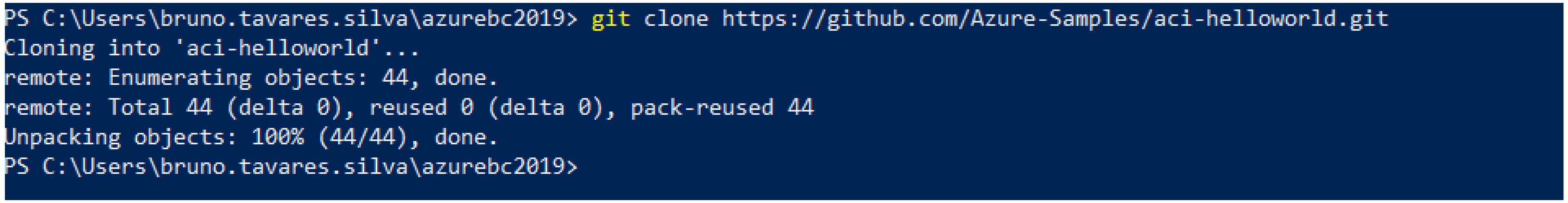
Deploy app to Azure Container Registry

Deploy app to Azure Container Instance

Download the souce code from git

In a terminal window, run the following command to clone the sample app repository to your local machine, then change to the directory that contains the sample code.

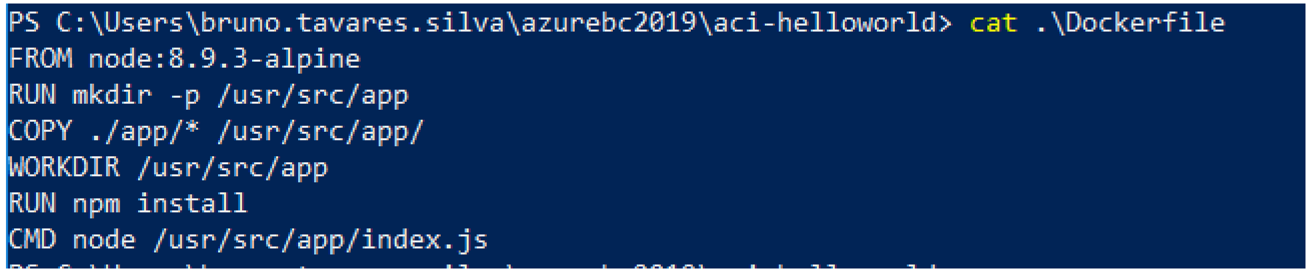
git clone https://github.com/Azure-Samples/aci-helloworld.git



```
PS C:\Users\bruno.tavares.silva\azurebc2019> git clone https://github.com/Azure-Samples/aci-helloworld.git
Cloning into 'aci-helloworld'...
remote: Enumerating objects: 44, done.
remote: Total 44 (delta 0), reused 0 (delta 0), pack-reused 44
Unpacking objects: 100% (44/44), done.
PS C:\Users\bruno.tavares.silva\azurebc2019>
```

Build the image from the Docker file

In the Git repository, take a look at Dockerfile. This file describes the Python environment that is required to run your application.

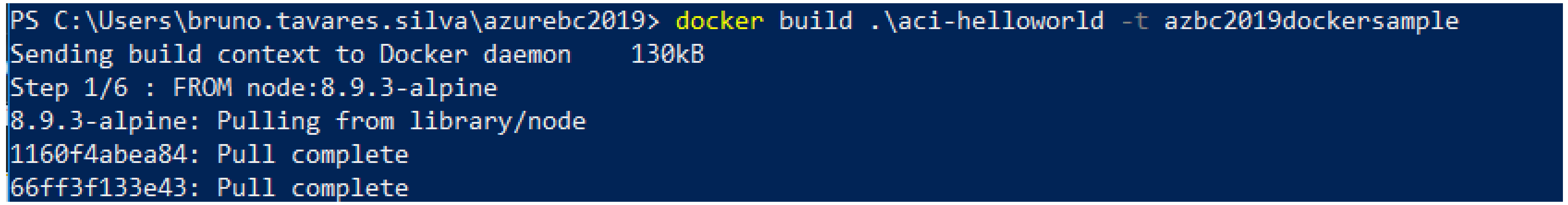


```
PS C:\Users\bruno.tavares.silva\azurebc2019\aci-helloworld> cat .\Dockerfile
FROM node:8.9.3-alpine
RUN mkdir -p /usr/src/app
COPY ./app/* /usr/src/app/
WORKDIR /usr/src/app
RUN npm install
CMD node /usr/src/app/index.js
```

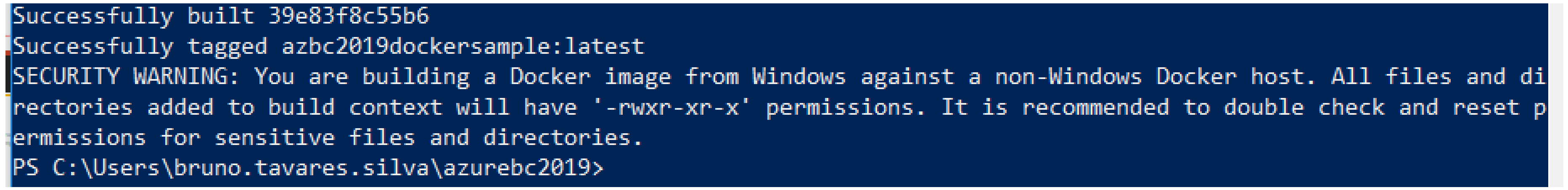
Build the image from the Docker file

Build the Docker image with the ***docker build*** command.

docker build .\aci-helloworld -t mydockerimage



```
PS C:\Users\bruno.tavares.silva\azurebc2019> docker build .\aci-helloworld -t azbc2019dockersample
Sending build context to Docker daemon    130kB
Step 1/6 : FROM node:8.9.3-alpine
8.9.3-alpine: Pulling from library/node
1160f4abea84: Pull complete
66ff3f133e43: Pull complete
```

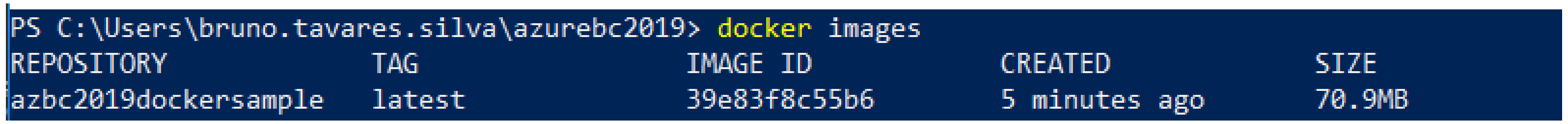


```
Successfully built 39e83f8c55b6
Successfully tagged azbc2019dockersample:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and di
rectories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset p
ermissions for sensitive files and directories.
PS C:\Users\bruno.tavares.silva\azurebc2019>
```

Build the image from the Docker file

Use the *docker images* command to see the built image:

docker images

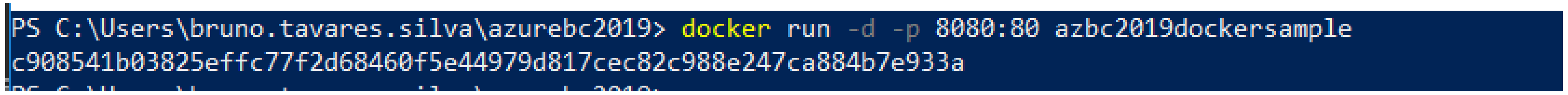


```
PS C:\Users\bruno.tavares.silva\azurebc2019> docker images
REPOSITORY             TAG                 IMAGE ID            CREATED             SIZE
azbc2019dockersample   latest              39e83f8c55b6        5 minutes ago       70.9MB
```

Build the image from the Docker file

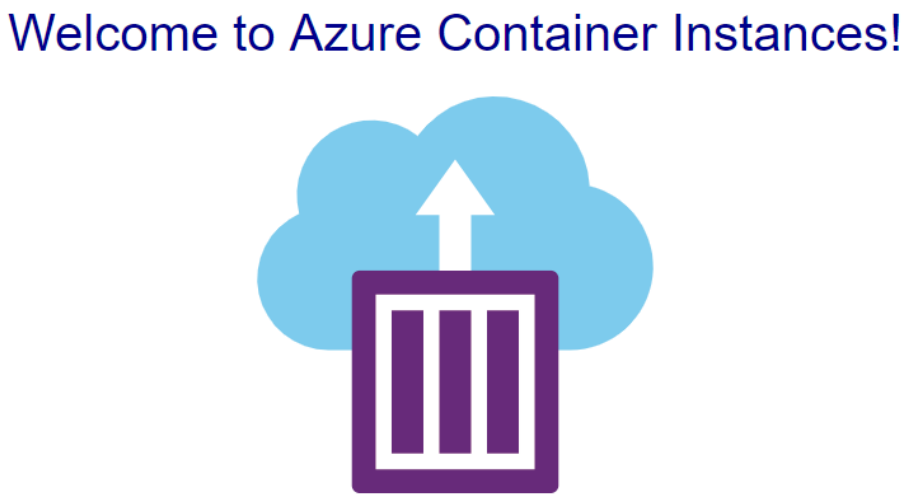
Test that the build works by running the Docker container. Issue the docker run command and pass the name and tag of the image to it. Be sure to specify the port using the -p argument.

docker run -p 8080:80 mydockerimage



```
PS C:\Users\bruno.tavares.silva\azurebc2019> docker run -d -p 8080:80 azbc2019dockersample
c908541b03825effc77f2d68460f5e44979d817cec82c988e247ca884b7e933a
```

Verify the web app and container are functioning correctly by browsing to http://localhost:8080.



Welcome to Azure Container Instances!

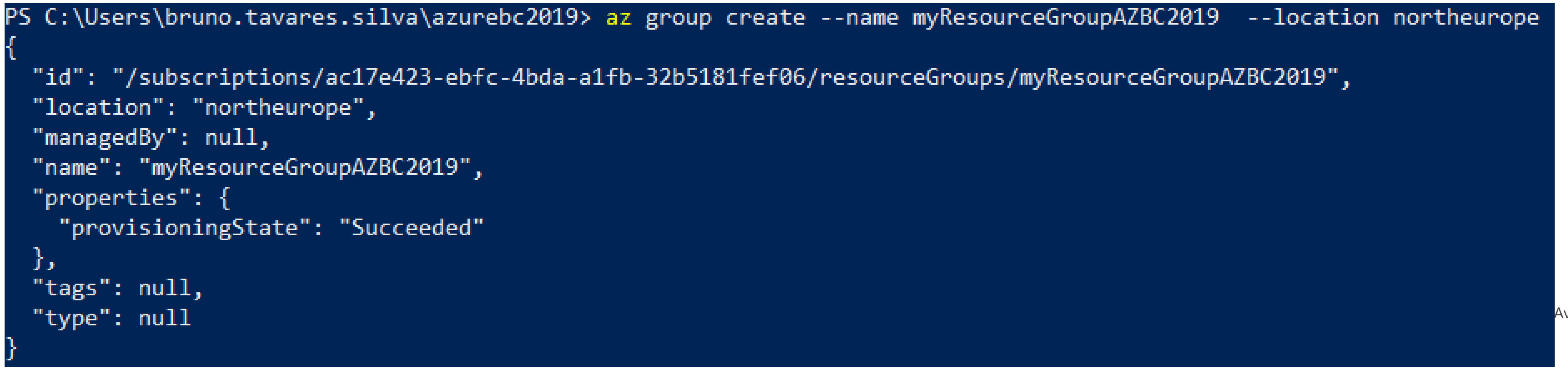
Deploy app to Azure Container Registry

To create an app that uses the image you just created, you run Azure CLI commands that create a resource group, pushes the image, and then creates the App Service plan web app to run it.

Create a Resource Group

A resource group is a logical container into which Azure resources like web apps, databases, and storage accounts are deployed and managed. For example, you can choose to delete the entire resource group in one simple step later.

az group create --name myResourceGroup --location northeurope



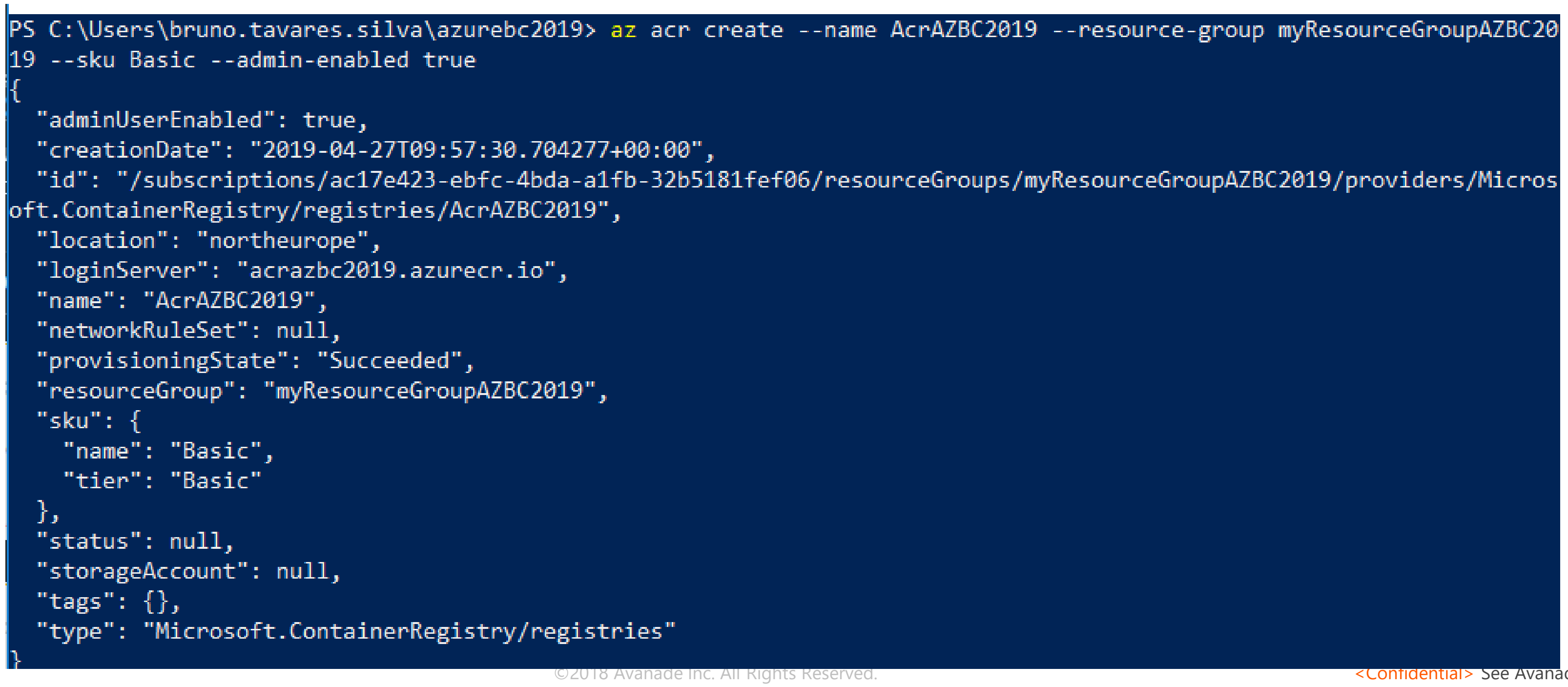
```
PS C:\Users\bruno.tavares.silva\azurebc2019> az group create --name myResourceGroupAZBC2019  --location northeurope
{
  "id": "/subscriptions/ac17e423-ebfc-4bda-a1fb-32b5181fef06/resourceGroups/myResourceGroupAZBC2019",
  "location": "northeurope",
  "managedBy": null,
  "name": "myResourceGroupAZBC2019",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": null
}
```

Deploy app to Azure Container Registry

Create an Azure Container Registry

In the Cloud Shell, use the az acr create command to create an Azure Container Registry.

az acr create --name <azure-container-registry-name> --resource-group myResourceGroup --sku Basic --admin-enabled true



```
PS C:\Users\bruno.tavares.silva\azurebc2019> az acr create --name AcrAZBC2019 --resource-group myResourceGroupAZBC20
19 --sku Basic --admin-enabled true
{
  "adminUserEnabled": true,
  "creationDate": "2019-04-27T09:57:30.704277+00:00",
  "id": "/subscriptions/ac17e423-ebfc-4bda-a1fb-32b5181fef06/resourceGroups/myResourceGroupAZBC2019/providers/Micros
oft.ContainerRegistry/registries/AcrAZBC2019",
  "location": "northeurope",
  "loginServer": "acrazbc2019.azurecr.io",
  "name": "AcrAZBC2019",
  "networkRuleSet": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "myResourceGroupAZBC2019",
  "sku": {
    "name": "Basic",
    "tier": "Basic"
  },
  "status": null,
  "storageAccount": null,
  "tags": {},
  "type": "Microsoft.ContainerRegistry/registries"
}
```

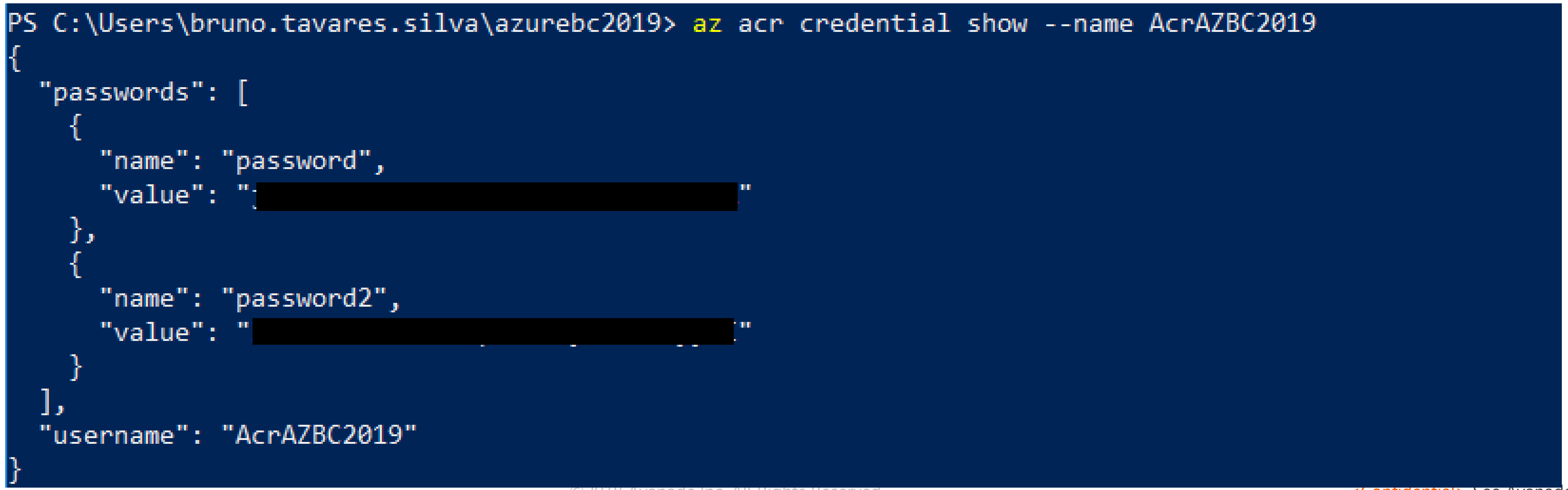
Deploy app to Azure Container Registry

Sign in to Azure Container Registry

To push an image to the registry, you need to authenticate with the private registry. In the Cloud Shell, use the az acr show command to retrieve the credentials from the registry you created.

az acr credential show --name <azure-container-registry-name>

The output reveals two passwords along with the user name. (ATTENTION! Please save this passwords)



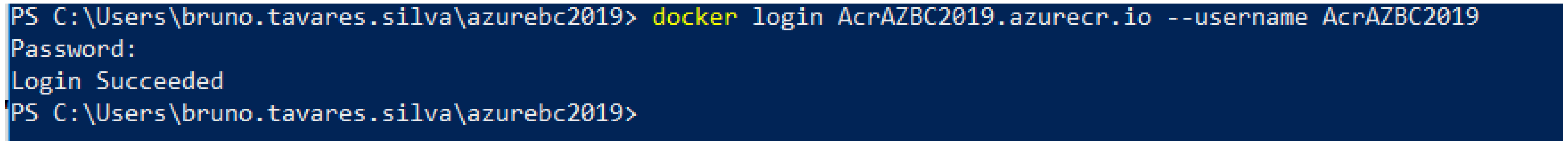
```
PS C:\Users\bruno.tavares.silva\azurebc2019> az acr credential show --name AcrAZBC2019
{
  "passwords": [
    {
      "name": "password",
      "value": "[redacted]"
    },
    {
      "name": "password2",
      "value": "[redacted]"
    }
  ],
  "username": "AcrAZBC2019"
}
```

Deploy app to Azure Container Registry

Sign in to Azure Container Registry

From your local terminal window, sign in to the Azure Container Registry using the docker login command, as shown in the following example. Replace <azure-container-registry-name> and <registry-username> with values for your registry. When prompted, type in one of the passwords from the previous step.

docker login <azure-container-registry-name>.azurecr.io --username <registry-username>



```
PS C:\Users\bruno.tavares.silva\azurebc2019> docker login AcrAZBC2019.azurecr.io --username AcrAZBC2019
Password:
Login Succeeded
PS C:\Users\bruno.tavares.silva\azurebc2019>
```

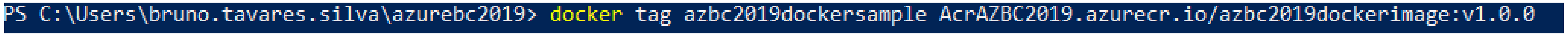
Confirm that the login succeeds.

Deploy app to Azure Container Registry

Push image to Azure Container Registry

Tag your local image for the Azure Container Registry. For example:

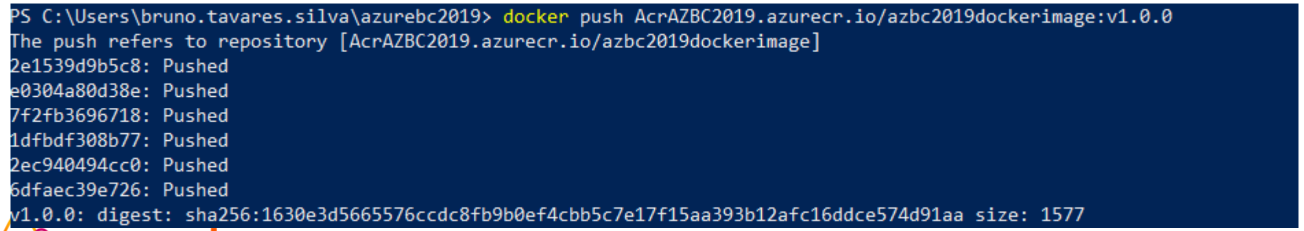
docker tag mydockerimage <azure-container-registry-name>.azurecr.io/mydockerimage:v1.0.0



```
PS C:\Users\bruno.tavares.silva\azurebc2019> docker tag azbc2019dockersample AcrAZBC2019.azurecr.io/azbc2019dockerimage:v1.0.0
```

Push the image by using the docker push command. Tag the image with the name of the registry, followed by your image name and tag.

docker push <azure-container-registry-name>.azurecr.io/mydockerimage:v1.0.0



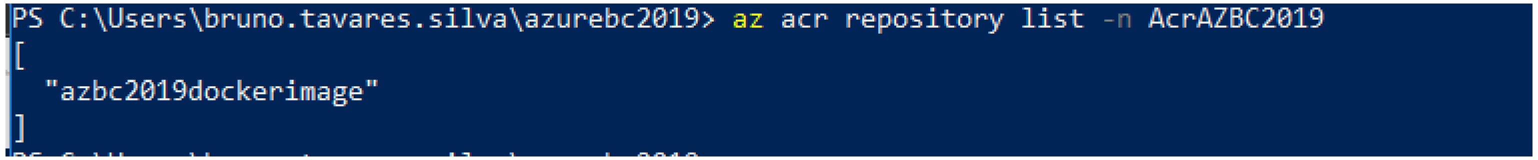
```
PS C:\Users\bruno.tavares.silva\azurebc2019> docker push AcrAZBC2019.azurecr.io/azbc2019dockerimage:v1.0.0
The push refers to repository [AcrAZBC2019.azurecr.io/azbc2019dockerimage]
2e1539d9b5c8: Pushed
e0304a80d38e: Pushed
7f2fb3696718: Pushed
1dfbdf308b77: Pushed
2ec940494cc0: Pushed
6dfaec39e726: Pushed
v1.0.0: digest: sha256:1630e3d5665576ccdc8fb9b0ef4cbb5c7e17f15aa393b12afc16ddce574d91aa size: 1577
```

Deploy app to Azure Container Registry

Push image to Azure Container Registry

Verify that the push is successful.

az acr repository list -n <azure-container-registry-name>



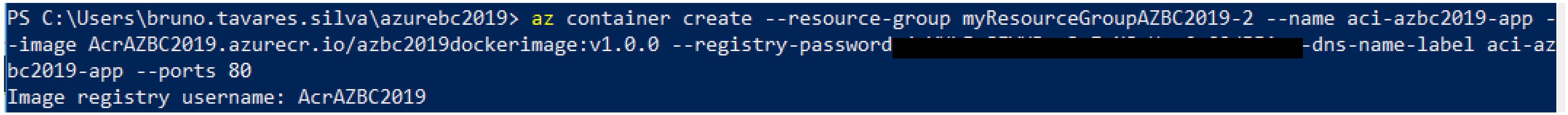
```
PS C:\Users\bruno.tavares.silva\azurebc2019> az acr repository list -n AcrAZBC2019
[
  "azbc2019dockerimage"
]
```

Deploy app to Azure Container Instance

Create a container in Azure Container Instance

Now that you have a resource group and your image in ACR you can run a container in Azure. To create a container instance with the Azure CLI, provide a resource group name, container instance name, and Docker container image to the az container create command.

az container create --resource-group myResourceGroup --name aci-tutorial-app --image <acrLoginServer>/aci-tutorial-app:v1 --cpu 1 --memory 0.5 --registry-password <acrPassword> --dns-name-label aci-azbc2019-app --ports 80



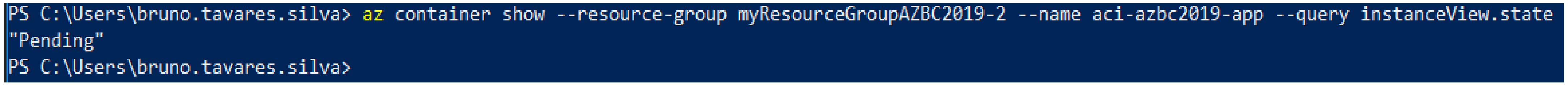
```
PS C:\Users\bruno.tavares.silva\azurebc2019> az container create --resource-group myResourceGroupAZBC2019-2 --name aci-azbc2019-app -
-image AcrAZBC2019.azurecr.io/azbc2019dockerimage:v1.0.0 --registry-password [redacted]-dns-name-label aci-az
bc2019-app --ports 80
Image registry username: AcrAZBC2019
```

Deploy app to Azure Container Instance

Create a container in Azure Container Instance

Within a few seconds, you should receive an initial response from Azure Resource Manager. To view the state of the deployment, use az container show:

az container show --resource-group myResourceGroup --name aci-tutorial-app --query instanceView.state



```
PS C:\Users\bruno.tavares.silva> az container show --resource-group myResourceGroupAZBC2019-2 --name aci-azbc2019-app --query instanceView.state
"Pending"
PS C:\Users\bruno.tavares.silva>
```

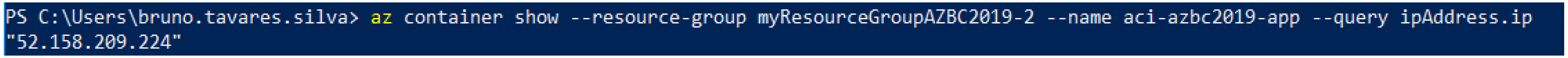
Repeat the az container show command until the state changes from Pending to Running, which should take under a minute. When the container is Running, proceed to the next step.

Deploy app to Azure Container Instance

Create a container in Azure Container Instance

Once the deployment succeeds, display the container's public IP address with the az container show command:

az container show --resource-group myResourceGroup --name aci-tutorial-app --query ipAddress.ip



```
PS C:\Users\bruno.tavares.silva> az container show --resource-group myResourceGroupAZBC2019-2 --name aci-azbc2019-app --query ipAddress.ip
"52.158.209.224"
```

Verify the web app and container are functioning correctly by browsing to http://55.158.209.224 (sample)

Welcome to Azure Container Instances!