Deploy Docker Image to Azure

# Optional: Create docker registry in Azure

Defining your private Docker registry in Azure is easy. Just type Container registries into the Azure

search bar and select Container registries. On the page that appears, click on the Create button.

The following form will appear:

Graphical user interface, text, application, email

Description automatically generated

*Figure 5.4: Creating an Azure private Docker registry*

The name you select is used to compose the overall registry URI: <name>.azurecr.io. As usual, you

can specify the subscription, resource group, and location. The SKU dropdown lets you choose from

various levels of offerings that differ in terms of performance, available memory, and a few other

auxiliary features.

Whenever you mention image names in Docker commands or in a Visual Studio publish form, you

must prefix them with the registry URI: <name>.azurecr.io/<my imagename>.

If images are created with Visual Studio, then they can be published by following the instructions

that appear once you’ve published the project. Otherwise, you must use docker commands to push

them into your registry.

The easiest way to use Docker commands that interact with the Azure registry is by installing the

Azure CLI on your computer. Download the installer from https://aka.ms/installazurecliwindows

and execute it. Once the Azure CLI has been installed, you can use the az command from Windows

Command Prompt or PowerShell. In order to connect with your Azure account, you must execute the

following login command:

Graphical user interface, text, application

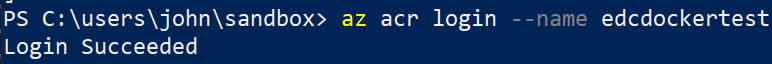
Description automatically generated

Need to use az command from Azure CLI to login to Azure first.

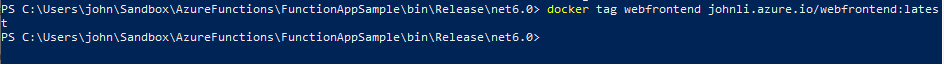
Text

Description automatically generated

Then login to Azure Container Registry (ACR)



Then docker tag (can be tag as a new image name, in this case it’s “dotnetcoretest”



Graphical user interface, text

Description automatically generated with medium confidence

The use docker push to push it to ACR repository:

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, email

Description automatically generated

Now you can see the renamed image “dotnetcoretest” has been deployed to ACR Repository “edcdpclertest”

Graphical user interface, text, application, email

Description automatically generated

# Run Docker Image from Azure Container Repository in a ACI (Azure Container Instance) from Azure website UI

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Open browser to public IP address: 20.232.51.160 and you will see the application running on the Azure Container Instance:

Graphical user interface, text, application

Description automatically generated

Or use Container instances 🡪 Create container instance

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Open the FQDN on a browser:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

# Create Azure Container Instance using Azure CLI – AZ command

**Command:**

$RES\_GROUP="dev"

$ACR\_NAME="edcdockertest"

$AKV\_NAME="edctestkeyvault"

az keyvault create -g $RES\_GROUP -n $AKV\_NAME

**Result:**

{

"id": "/subscriptions/8e3a03d4-b34f-4e02-9004-ea5707bd7752/resourceGroups/dev/providers/Microsoft.KeyVault/vaults/edctestkeyvault",

"location": "canadaeast",

"name": "edctestkeyvault",

"properties": {

"accessPolicies": [

{

"applicationId": null,

"objectId": "7af16ec0-1ac2-4cbe-96dd-920f149fc15a",

"permissions": {

"certificates": [

"all"

],

"keys": [

"all"

],

"secrets": [

"all"

],

"storage": [

"all"

]

},

"tenantId": "d4167d58-713e-4e8f-8ba2-869b5491fb80"

}

],

"createMode": null,

"enablePurgeProtection": null,

"enableRbacAuthorization": null,

"enableSoftDelete": true,

"enabledForDeployment": false,

"enabledForDiskEncryption": null,

"enabledForTemplateDeployment": null,

"hsmPoolResourceId": null,

"networkAcls": null,

"privateEndpointConnections": null,

"provisioningState": "Succeeded",

"publicNetworkAccess": "Enabled",

"sku": {

"family": "A",

"name": "standard"

},

"softDeleteRetentionInDays": 90,

"tenantId": "d4167d58-713e-4e8f-8ba2-869b5491fb80",

"vaultUri": "https://edctestkeyvault.vault.azure.net/"

},

"resourceGroup": "dev",

"systemData": {

"createdAt": "2022-09-02T17:10:09.915000+00:00",

"createdBy": "lixxjo@edc.ca",

"createdByType": "User",

"lastModifiedAt": "2022-09-02T17:10:09.915000+00:00",

"lastModifiedBy": "lixxjo@edc.ca",

"lastModifiedByType": "User"

},

"tags": {},

"type": "Microsoft.KeyVault/vaults"

}

**Command:**

az ad sp create-for-rbac --name http://$ACR\_NAME-pull --scopes $(az acr show --name $ACR\_NAME --query id --output tsv) --role acrpull

**Result**:

Creating 'acrpull' role assignment under scope '/subscriptions/8e3a03d4-b34f-4e02-9004-ea5707bd7752/resourceGroups/DEV/providers/Microsoft.ContainerRegistry/registries/edcdockertest'

The output includes credentials that you must protect. Be sure that you do not include these credentials in your code or check the credentials into your source control. For more information, see https://aka.ms/azadsp-cli

{

"appId": "c5317979-1c6a-432f-82ee-9e8436d00233",

"displayName": "http://edcdockertest-pull",

"password": "cKa8Q~dZDCeXP1slaWs7.LuaOdy2K0SOzBX1sdlz",

"tenant": "d4167d58-713e-4e8f-8ba2-869b5491fb80"

}

**Command:**

SP\_ID="c5317979-1c6a-432f-82ee-9e8436d00233"

az keyvault secret set --vault-name $AKV\_NAME --name $ACR\_NAME-pull-pwd --value "cKa8Q~dZDCeXP1slaWs7.LuaOdy2K0SOzBX1sdlz"

**Result:**

value "cKa8Q~dZDCeXP1slaWs7.LuaOdy2K0SOzBX1sdlz"

{

"attributes": {

"created": "2022-09-02T17:25:57+00:00",

"enabled": true,

"expires": null,

"notBefore": null,

"recoveryLevel": "Recoverable+Purgeable",

"updated": "2022-09-02T17:25:57+00:00"

},

"contentType": null,

"id": "https://edctestkeyvault.vault.azure.net/secrets/edcdockertest-pull-pwd/37bb29784e084f2aabee783f2d63bb8a",

"kid": null,

"managed": null,

"name": "edcdockertest-pull-pwd",

"tags": {

"file-encoding": "utf-8"

},

"value": "cKa8Q~dZDCeXP1slaWs7.LuaOdy2K0SOzBX1sdlz"

}

**Command:**

az keyvault secret set --vault-name $AKV\_NAME --name $ACR\_NAME-pull-usr --value $(az ad sp show --id $SP\_ID --query appId --output tsv)

**Result:**

{

"attributes": {

"created": "2022-09-02T17:50:22+00:00",

"enabled": true,

"expires": null,

"notBefore": null,

"recoveryLevel": "Recoverable+Purgeable",

"updated": "2022-09-02T17:50:22+00:00"

},

"contentType": null,

"id": "https://edctestkeyvault.vault.azure.net/secrets/edcdockertest-pull-usr/8182ac88ed9a449bae47129beb1114ec",

"kid": null,

"managed": null,

"name": "edcdockertest-pull-usr",

"tags": {

"file-encoding": "utf-8"

},

"value": "c5317979-1c6a-432f-82ee-9e8436d00233"

}

**Command:**

$ACR\_LOGIN\_SERVER=$(az acr show --name $ACR\_NAME --resource-group $RES\_GROUP --query "loginServer" --output tsv)

**Result:**

edcdockertest.azurecr.io

**Command:**

az container create `

--name aci-demo `

--resource-group $RES\_GROUP `

--image $ACR\_LOGIN\_SERVER/dotnetcoretest `

--registry-login-server $ACR\_LOGIN\_SERVER `

--registry-username $(az keyvault secret show --vault-name $AKV\_NAME -n $ACR\_NAME-pull-usr --query value -o tsv) `

--registry-password $(az keyvault secret show --vault-name $AKV\_NAME -n $ACR\_NAME-pull-pwd --query value -o tsv) `

--dns-name-label aci-demo-dotnetcore `

--query ipAddress.fqdn

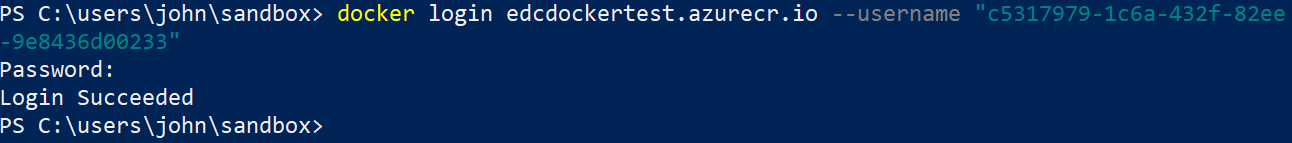
**Result:**

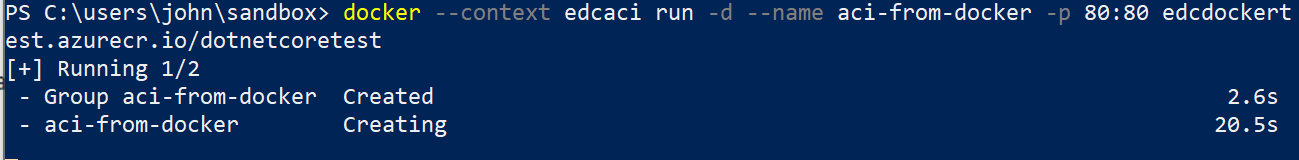
"aci-demo-dotnetcore.canadaeast.azurecontainer.io"

Browse to: http://aci-demo-dotnetcore.canadaeast.azurecontainer.io/

working!!

# Create Azure Container Instance through Docker CLI





Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Working!!

**You can also use dnsname parameter to specify and DNS for the container instance:**

Text

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Working!!