Provision a New Azure Container Registry Service

In this lab, you will learn how to provision a new Azure Container Registry using the Azure CLI.

Log into the Azure CLI
Understand containers, container images, and Azure Container Registry (ACR)
Provision an Azure Container Registry
Confirm that the new Azure Container Registry was successfully created
Clean up the Container Registry

Understanding Containers and Azure Container Registry

So what are Docker containers, and how can we host them in Microsoft Azure?

According to Docker, "a container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another." You can package your website, and all its dependencies (such as databases) into a Docker image, and run it on your hosting environment. The team does not need to worry about setting up the website dependencies anymore; all they need to do is deploy the container.

According to Docker, "a Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings."

Container images will be pulled into hosts and then become containers at runtime.

In order to host your application in a Docker container, you need to follow these steps:

Package your application and all its dependencies into a Docker container image.
Push your container image into a container registry service such as Docker Hub or the Azure Container Registry (ACR).
Configure your hosting environment to pull the container image from the registry and run it. The following Azure services can host containers:
Azure Container Instances (ACI)
Azure Kubernetes Service (AKS)
Azure App Services
Azure Function Apps
In this lab, we work with the ACR covering steps 1 and 2 of the preceding list. In the next lab, we will introduce Azure Container Instances, which is one of the hosting options in step 3.

Note: You always have only ONE container image, and you can deploy it to multiple hosts.

In this lab, we will create an Azure Container Registry using the Azure CLI, confirm it's created, update its properties, and clean it up at the end.

Provision a New Azure Container Registry
Use the following command to create your first Azure Container Registry:

az acr create --name $acrName --resource-group $resource --sku Standard

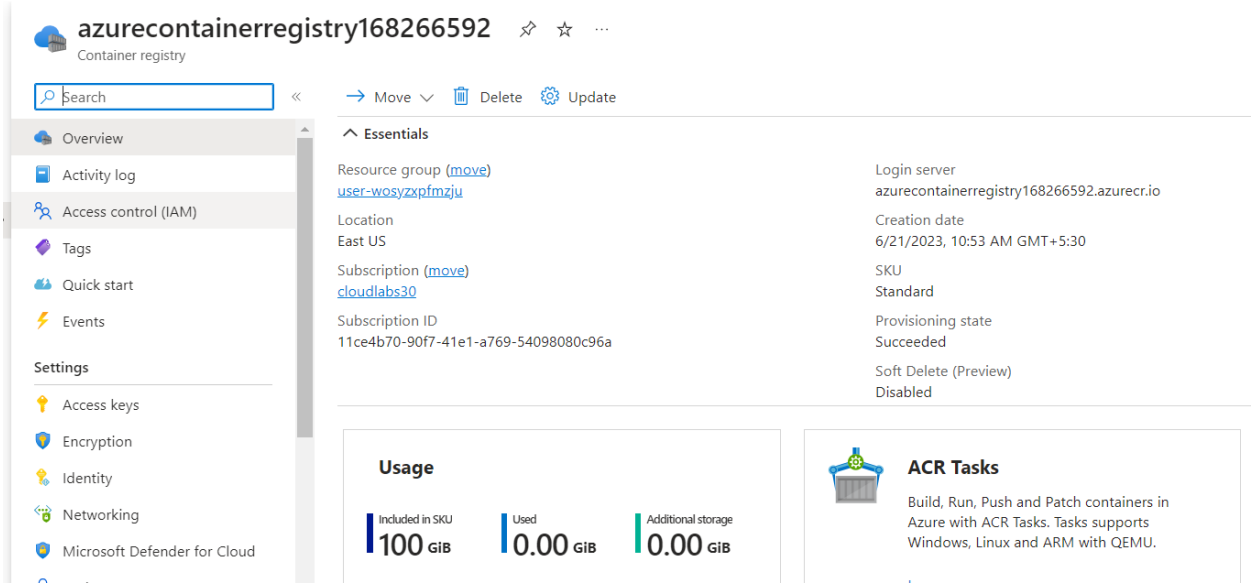
Note: This command might take a few moments!

Let's take a look at the command parameters:

--resource-group: The name of the parent resource group to create the new Azure Container Registry (ACR) in.
--name: The name of the new ACR.
--location: The region to create the ACR in.
--sku: The pricing tier for the registry. Accepted values are Basic, Classic, Standard, and Premium.
Here are a few more properties which you can set when creating the ACR:

--public-network-enabled: Allow public network access for the container registry (the default is allow).
--zone-redundancy: When Enabled, the ACR will be replicated among multiple availability zones in the region.
--allow-trusted-services: Allow services such as Azure Container Instances to access the ACR. See this page for more details.
--admin-enabled: Indicates whether the ACR admin user is enabled.
Wait for the command to succeed. In the next step, we'll confirm the ACR creation using the Azure CLI.

```
]
$ az acr create --name $acrName --resource-group $resource --sku Standard
{
  "adminUserEnabled": false,
  "anonymousPullEnabled": false,
  "creationDate": "2023-06-21T05:23:06.165239+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
```



Confirm that the Azure Container Registry is Provisioned
Use the following command to confirm the new ACR (Azure Container Registry) has been created:

az acr show --name $acrName --query "{Name:name}"

```
}
$ az acr show --name $acrName --query "{Name:name}"
{
  "Name": "azurecontainerregistry168266592"
}
$
```

You can also use the following command to list all the ACRs under your test subscription:

az acr list --resource-group $resource --query "[].{Name:name}"

```
}
$ az acr list --resource-group $resource --query "[].{Name:name}"
[
  {
    "Name": "azurecontainerregistry168266592"
  }
]
```

Now your clients can push or pull container images to your ACR service, provided that they have the right access.

Use the following command to clean up the ACR:

az acr delete --name $acrName

```
$ az acr delete --name $acrName
Are you sure you want to delete the regi
```

Manage an Existing Azure Container Registry

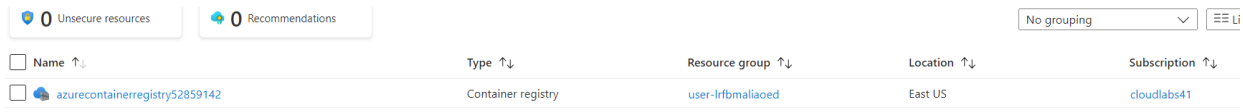
In this lab, you will learn how to manage a new Azure Container Registry using the Azure CLI.

Azure Container Registry is a private Docker image registry hosted in Microsoft Azure. In Microsoft's words, you can "create and maintain Azure container registries to store and manage your private Docker container images and related artifacts."

Log into the Azure CLI
Update the Configuration for an Azure Container Registry
Confirm that the Azure Container Registry was successfully updated
Clean up the Container Registry

Use the following command to confirm that the ACR is present:

az acr list --resource-group $resource --query "[].{Name:name}"



In this lab, we will manage this ACR instance by updating its configuration using the Azure CLI.

Updating Configuration for an Azure Container Registry
You can change an existing ACR configuration using the Azure CLI.

For instance, use the following command to enable the admin user for your Azure Container Registry:

az acr update --name $acrName --resource-group $resource --admin-enabled true

Let's take a look at the command parameters:

--resource-group: The name of the parent resource group for the Azure Container Registry (ACR).
--name: The name for the existing ACR.
--admin-enabled: Enable, or disable the admin user for the existing ACR.
Here are a few more properties which you can set when updating the ACR:

--tags: Add new resource tags to the ACR. Resource tags are useful to label and manage your Azure resources.
--anonymous-pull-enabled: When Enabled, the ACR will be replicated among multiple availability zones in the region.
--allow-trusted-services: Allow services such as Azure Container Instances to access the ACR

```
$ az acr update --name $acrName --resource-group $resource --admin-enabled true
{
  "adminUserEnabled": true,
  "anonymousPullEnabled": false,
  "creationDate": "2023-06-21T05:32:10.409025+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
    "status": "disabled"
```

Confirm that the Azure Container Registry is Updated
Use the following command to confirm the updated ACR configuration:

az acr show --name $acrName --query "{Name:name, AdminUserEnabled:adminUserEnabled}"

```
$ az acr show --name $acrName --query "{Name:name, AdminUserEnabled:adminUserEnabled}"
{
  "AdminUserEnabled": true,
  "Name": "azurecontainerregistry52859142"
}
```

Delete the Azure Container Registry
Time to clean up our ACR. Use the following command to clean up the ACR:

az acr delete --name $acrName

```
$ az acr delete --name $acrName
Are you sure you want to delete the registry 'azurecontainerregistry52859142'? (y/n): y
$
```

Provision a New Azure Container Instances Service

In this lab, you will learn how to create new Azure Container Instances using the Azure CLI.

Azure Container Instances (ACI) is an Azure service allowing you to host individual containers in the cloud.

"Azure Container Instances is a great solution for any scenario that can operate in isolated containers, including simple applications, task automation, and build jobs."

Log into the Azure CLI
Understand Azure Container Instances (ACI)
Provision Azure Container Instances
Confirm that the new Azure Container Instances Service was successfully created
Clean up the Container Instances

Understanding Azure Container Instances
In the previous lab, we learned about the Azure Container Registry (ACR). You can keep your container images in the ACR, but you still need a service to host and run these containers. Azure Container Instances (ACI) can help you achieve this.

You can use ACI to host isolated and individual containers running simple applications. You can configure ACI to pull your container images from the Azure Container Registry (ACR) and run them.

As a reminder, container images will be pulled into hosts and then become containers at runtime.

In order to host your application in a Docker container, you need to follow these steps:

Package your application and all its dependencies into a Docker container image.
Push your container image into a container registry service such as Docker Hub or the Azure Container Registry (ACR).
Configure your hosting environment to pull the container image from the registry, and run it. The following Azure services can host containers:
Azure Container Instances (ACI)
Azure Kubernetes Service (AKS)
Azure App Services
Azure Function Apps
In this lab, we work with ACI. This covers one of the container hosting options in step 3 of the preceding list.

Reminder: You always have only ONE container image, and you can deploy it to multiple hosts.

In this lab, we will create an Azure Container Instances (ACI) service using the Azure CLI, confirm it's created, update its properties, and finally will clean it up at the end.

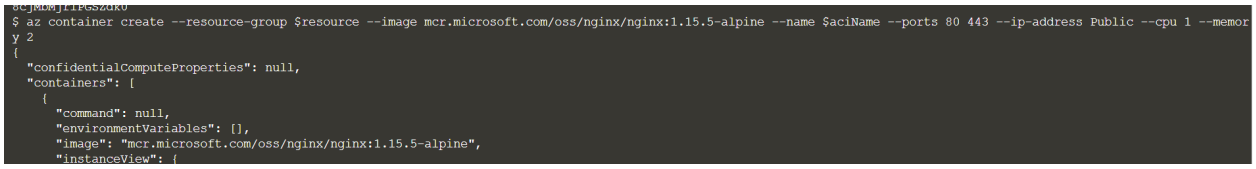
Provision a New Azure Container Instances
Use the following command to create your first Azure Container Instances (ACI):

Note: This command might take a few moments!

az container create --resource-group $resource --image mcr.microsoft.com/oss/nginx/nginx:1.15.5-alpine --name $aciName --ports 80 443 --ip-address Public --cpu 1 --memory 2

Let's take a look at the command parameters:

--resource-group: The name of the parent resource group to create the new Azure Container Instances (ACI) in.
--name: The name for the new ACI instance.
--location: The region to create the ACI in.
--cpu: The number of the CPU cores dedicated to this ACI instance.
--memory: Memory in GB to dedicate to this ACI instance.
--image: The name and version of the container image to pull from a container registry and run. In this example, we are pulling a public image from Microsoft. No credentials are needed to pull and run this image.
--ports: Ports to open to the public. You need ports 443 and/or 80 if running a web application.
--ip-address: The IP address type for the ACI. Accepted values are Private and Public. For this demo, we need a public IP address so we can hit the hosted application.



If you were pulling a private image from ACR (or Docker Hub), you would need to add the following parameters to enable ACI to authenticate to the container registry and pull the container image:

--registry-login-server: The URL of the ACR instance hosting the container image to pull.
--registry-username: The username of the ACR instance hosting the container image to pull.
--registry-password: The password of the ACR instance hosting the container image to pull.

Confirm the Azure Container Instances is provisioned
Use the following command to confirm the new ACI has been created:

az container show --name $aciName --resource-group $resource --query "{Name:name}"

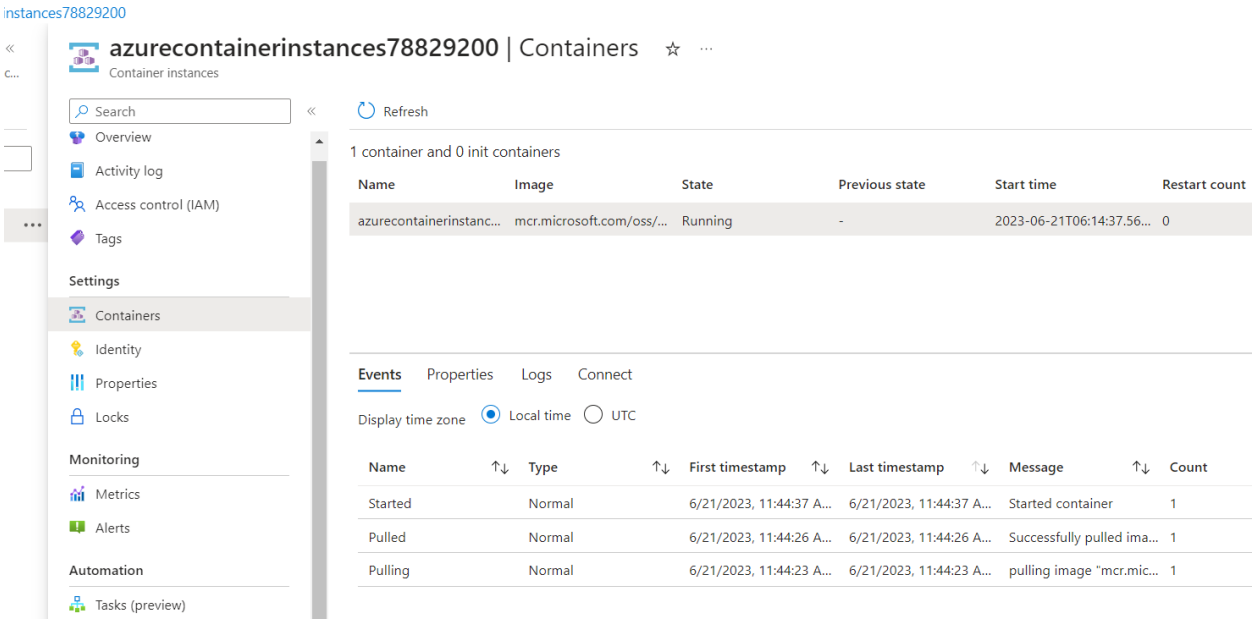
```
$ az container show --name $aciName --resource-group $resource --query "{Name:name}"
{
  "Name": "azurecontainerinstances78829200"
}
$
```

az container list --resource-group $resource --query "[].{Name:name}"

```
$ az container list --resource-group $resource --query "[].{Name:name}"
[
  {
    "Name": "azurecontainerinstances78829200"
  }
]
$
```

Now, your clients can pull container images from ACR to this ACI instance and run them.

We can also confirm the ACI creation in the Azure portal. We'll do so in the next step.



Delete the Azure Container Instances
Time to clean up our ACI. Use the following command to clean up the ACI:

az container delete --name $aciName --resource-group $resource

```
]
$ az container delete --name $aciName --resource-group $resource
Are you sure you want to perform this operation? (y/n): y
{
  "confidentialComputeProperties": null,
  "containers": [
    {
      "command": null,
      "environmentVariables": [],
```

Manage an Existing Azure Container Instances Service

In this lab, you will learn how to manage an ACI service using the Azure CLI.

Azure Container Instances (ACI) is an Azure service allowing you to host individual containers in the cloud. In Microsoft's words, "Azure Container Instances is a great solution for any scenario that can operate in isolated containers, including simple applications, task automation, and build jobs."

Log into the Azure CLI
Stop, start, and restart an Azure Container Instances (ACI) service
Confirm the Azure Container Instances State
Clean up the ACI

Azure Container Instances (ACI) service has already been created for you. Use the following command to confirm that the ACI is present:

az container list --resource-group $resource --query "[].{Name:name}"

```
$ az container list --resource-group $resource --query "[].{Name:name}"
[
  {
    "Name": "azurecontainerinstances183748500"
  }
]
$
```

In this lab, we will manage this ACI instance by stopping, starting, and restarting it using the Azure CLI.

Start, Stop, and Restart an ACI
You can manage an existing ACI service using the Azure CLI. For instance, use the following command to stop an ACI instance. This will stop any hosted application from functioning:

az container stop --name $aciName --resource-group $resource

az container show --name $aciName --resource-group $resource --query "{Name:name, State:instanceView.state}"

Now, use the az container command again to start the ACI:

az container start --name $aciName --resource-group $resource

az container show --name $aciName --resource-group $resource --query "{Name:name, State:instanceView.state}"

And finally, you can also use this command to restart your ACI service:

az container restart --name $aciName --resource-group $resource

Delete the ACI Service
Time to clean up our ACI.

Note: This command might take a few moments!

Use the following command to clean up the ACI:

az container delete --name $aciName --resource-group $resource

az container list --resource-group $resource --query "[].{Name:name}"