Azure CLI LAB

To work on your local PC You need to install Azure CLII:

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>

or You can use azure portal

1. If you're using a local install of the CLI, you need to sign in before performing any other steps.

az login

Complete the authentication process by following the steps displayed in your terminal.

1. Often you will have multiple subscriptions available. Set the subscription to use

az account set --subscription mySubscription

1. List region names

az account list-locations --output=table

az account list-locations --output=table | Select-String -Pattern "\(Europe\)"

1. ~~In Azure, all resources are allocated in a resource management group. Resource groups provide logical groupings of resources that make them easier to work with as a collection. For this tutorial, all of the created resources go into a single group named myResoruceGroup.~~

~~az group create --name myResoruceGroup --location westeurope~~

1. Verify group creation

For many Azure resources, Azure CLI provides a list subcommand to get resource details. For example, the Azure CLI group list command lists your Azure resource groups. This is useful to verify whether resource group creation was successful:

az group list

To get more concise information, you can format the output as a simple table:

az group list --output table

If you have several items in the group list, you can filter the return values by adding a query option using, for example, the following command:

az group list --query "[?name == '<rg name>']"

1. Verify available images

az vm image list

1. Create Windows VM

az vm create --resource-group myResoruceGroup `

--name s0-win-vm01-weu-lab `

--computer-name s0winvm01weulab `

--image Win2019Datacenter `

--admin-username myLabAdminName `

--admin-password myLabPassword1! `

--output json `

--verbose

1. Create Linux VM

az vm create --resource-group myResourceGroup `

--name s0-linux-vm01-weu-lab `

--image UbuntuLTS `

--generate-ssh-keys `

--output json `

--verbose

As the VM is created, you see the local values used and Azure resources being created due to the --verbose option. Once the VM is ready, JSON is returned from the Azure service including the public IP address.

JSONCopy

{

"fqdns": "",

"id": "...",

"location": "eastus",

"macAddress": "...",

"powerState": "VM running",

"privateIpAddress": "...",

"publicIpAddress": <PUBLIC\_IP\_ADDRESS>,

"resourceGroup": "TutorialResources",

"zones": ""

}

Confirm that the VM is running by connecting over SSH.

BashCopy

ssh <PUBLIC\_IP\_ADDRESS>

Go ahead and log out from the VM.

There are other ways to get this IP address after the VM has started. In the next section you will see how to get detailed information on the VM, and how to filter it.

1. Get VM information with queries.

Now that a VM has been created, detailed information about it can be retrieved. The common command for getting information from a resource is show

az vm show --name s0-linux-vm01-weu-lab --resource-group myResoruceGroup

You'll see a lot of information, which can be difficult to parse visually. The returned JSON contains information on authentication, network interfaces, storage, and more. Most importantly, it contains the Azure object IDs for resources that the VM is connected to. Object IDs allow accessing these resources directly to get more information about the VM's configuration and capabilities.

In order to extract the object ID we want, the --query argument is used. Queries are written in the [JMESPath query language](http://jmespath.org/). Start with getting the network interface controller (NIC) object ID.

az vm show --name s0-linux-vm01-weu-lab `

--resource-group myResoruceGroup `

--query 'networkProfile.networkInterfaces[].id' `

--output tsv

There's a lot going on here, just by adding the query. Each part of it references a key in the output JSON, or is a JMESPath operator.

* networkProfile is a key of the top-level JSON, which has networkInterfaces as a subkey. If a JSON value is a dictionary, its keys are referenced from the parent key with the . operator.
* The networkInterfaces value is an array, so it is flattened with the [] operator. This operator runs the remainder of the query on each array element. In this case, it gets the id value of every array element.

The output format tsv (tab-separated values) is guaranteed to only include the result data and whitespace consisting of tabs and newlines. Since the returned value is a single bare string, it's safe to assign directly to an environment variable. Go ahead and assign the NIC object ID to an environment variable now.

$NIC\_ID = az vm show -n s0-linux-vm01-weu-lab -g myResoruceGroup `  
--query 'networkProfile.networkInterfaces[].id' `  
-o tsv

This example also demonstrates the use of short arguments. You may use -g instead of --resource-group, -n instead of --name, and -o instead of --output

1. Set environment variables from CLI output

Now that you have the NIC ID, run az network nic show to get its information. Note that you don't need a resource group here, since the resource group name is contained within the Azure resource ID.

az network nic show --ids $NIC\_ID

This command shows all of the information for the network interface of the VM. This data includes DNS settings, IP information, security settings, and the MAC address. Right now the goal is to obtain the public IP address and subnet object IDs.

az network nic show --ids $NIC\_ID `

--query '{IP:ipConfigurations[].publicIpAddress.id, Subnet:ipConfigurations[].subnet.id}' `

-o json

This command displays a JSON object that has custom keys ('IP' and 'Subnet') for the extracted values. While this style of output might not be useful for command-line tools, it helps with human readability and can be used with custom scripts.

In order to use command-line tools, change the command to remove the custom JSON keys and output as tsv

$IP\_ID = az network nic show `

--ids $NIC\_ID `

--query '[ipConfigurations[].publicIpAddress.id]' `

-o tsv

$SUBNET\_ID = az network nic show `

--ids $NIC\_ID `

--query '[ipConfigurations[].subnet.id]' `

-o tsv

Use the public IP object ID to look up the public IP address and store it in a shell variable.

$VM1\_IP\_ADDR = az network public-ip show --ids $IP\_ID `

--query ipAddress `

-o tsv

Now you have the IP address of the VM stored in a shell variable. Go ahead and check that it is the same value that you used to initially connect to the VM.

$VM1\_IP\_ADDR

1. Creating a new VM on the existing subnet

The second VM uses the existing subnet. You can skip a few steps to get the public IP address of the new VM stored into an environment variable right away, since it's returned in the VM creation information. If you'd need other information about the VM later, it can always be obtained from the az vm show command.

$VM2\_IP\_ADDR= az vm create -g myResoruceGroup `

-n s0-win-vm02-weu-lab `

--image UbuntuLTS `

--generate-ssh-keys `

--subnet $SUBNET\_ID `

--query publicIpAddress `

-o tsv

Using the stored IP address, SSH into the newly created VM.

ssh $VM2\_IP\_ADDR

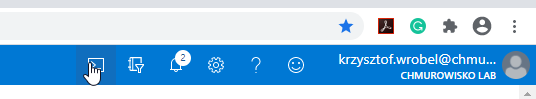
Go ahead and log out from the VM.

1. Cleanup

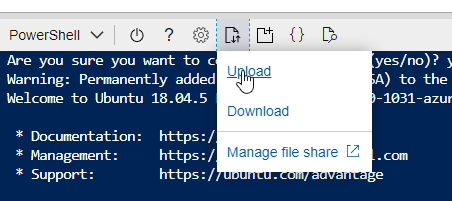
Create a file on your computer called “removeall.json” and paste in the following contents

{  
"$schema": "[https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#](https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json)",  
"contentVersion": "1.0.0.0",  
"parameters": { },  
"variables": { },  
"resources": [ ],  
"outputs": { }  
}

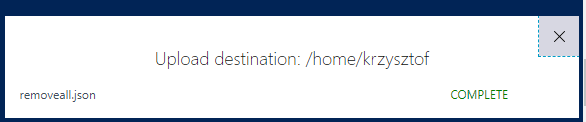
Launch Cloud Shell in the Azure Portal.



Select Upload to upload the file to your Cloud Shell



CD to the path that Cloud Shell indicated was the upload destination



Use the Azure CLI to remove the resources.

az deployment group create --mode complete --template-file ./removeall.json --resource-group myResoruceGroup

1. Extensions

**Find Azure CLI extensions**

az extension list-available --output table

**Install Azure CLI extensions**

az extension add --name <extension-name>

**Uninstall Azure CLI extensions**

az extension remove --name <extension-name>