Azure CLI Reference Guide

Version 1

SKYLINES

ACADEMY

Thank you for downloading the Skylines Academy Azure CLI Reference Guide.

We are very excited to be part of your education journey! We hope this guide helps you reach success – whether you are studying for exams, or if you are learning or updating your existing Azure skills.

Interested in learning more? We encourage you to check out our additional courses and study groups at the following:

Training Courses: www.skylinesacademy.com

[Skylines Academy]

[Follow on Twitter]

[Skylines Academy Videos]

[Azure Study Group]

Table of Contents

Introduction

Welcome to the Azure CLI Reference Guide. This guide will provide you with a reference on key Azure CLI commands often used by Azure administrators. Knowledge of the Azure CLI commands are also required to pass the Azure Administrator certification exams from Microsoft.

This guide is made up of several Azure CLI commands which focus on Azure Administration and are also a core part of Microsoft AZ-104, AZ-303 and AZ-304 Certifications.

Note: While we make every effort to test the commands and point out any concerns when deleting objects, be sure to test these out yourself. Before running any of these commands in production, we recommend you test them out in a separate Azure test account so that you are sure you know what they are doing. Some commands are destructive in nature (e.g., removing resource groups, tags, etc.) and you need to ensure you fully understand the commands that you execute.

This guide is divided into the following sections:

- Azure CLI Basics
- Cloud Shell
- Accounts and Subscriptions
- Resource Groups
- Governance
- Storage
- Virtual Machines
- Networking
- Azure Active Directory Commands

If you spot any errors in this guide, please submit them via the Contact Us page on the Skylines Academy web site.

Thank you,

Skylines Academy Team

Azure CLI Basics

Azure CLI was originally introduced as the cross-platform command line interface (CLI) for administrating your Azure environments. In early 2018, Microsoft has released PowerShell Core 6.0 which is also cross-platform; now the choice between Azure CLI and PowerShell is down to personal preference on whether you prefer PowerShell scripting or Bash/Batch scripting

As mentioned above Azure CLI is cross platform and is available for Windows, Linux and macOS. Microsoft also supplies a Docker image if you want to run the CLI shell within a container. If you do not already have Azure CLI installed locally on your computer, then you can download the latest version for your operating system from the following links:

- Installing Azure CLI on Windows
- Installing Azure CLI on macOS
- Installing Azure CLI on Linux with apt
- Running Azure CLI on Docker

It is always advised to keep your administration tools up-to-date, where possible, so you are able to take advantage of any new features.

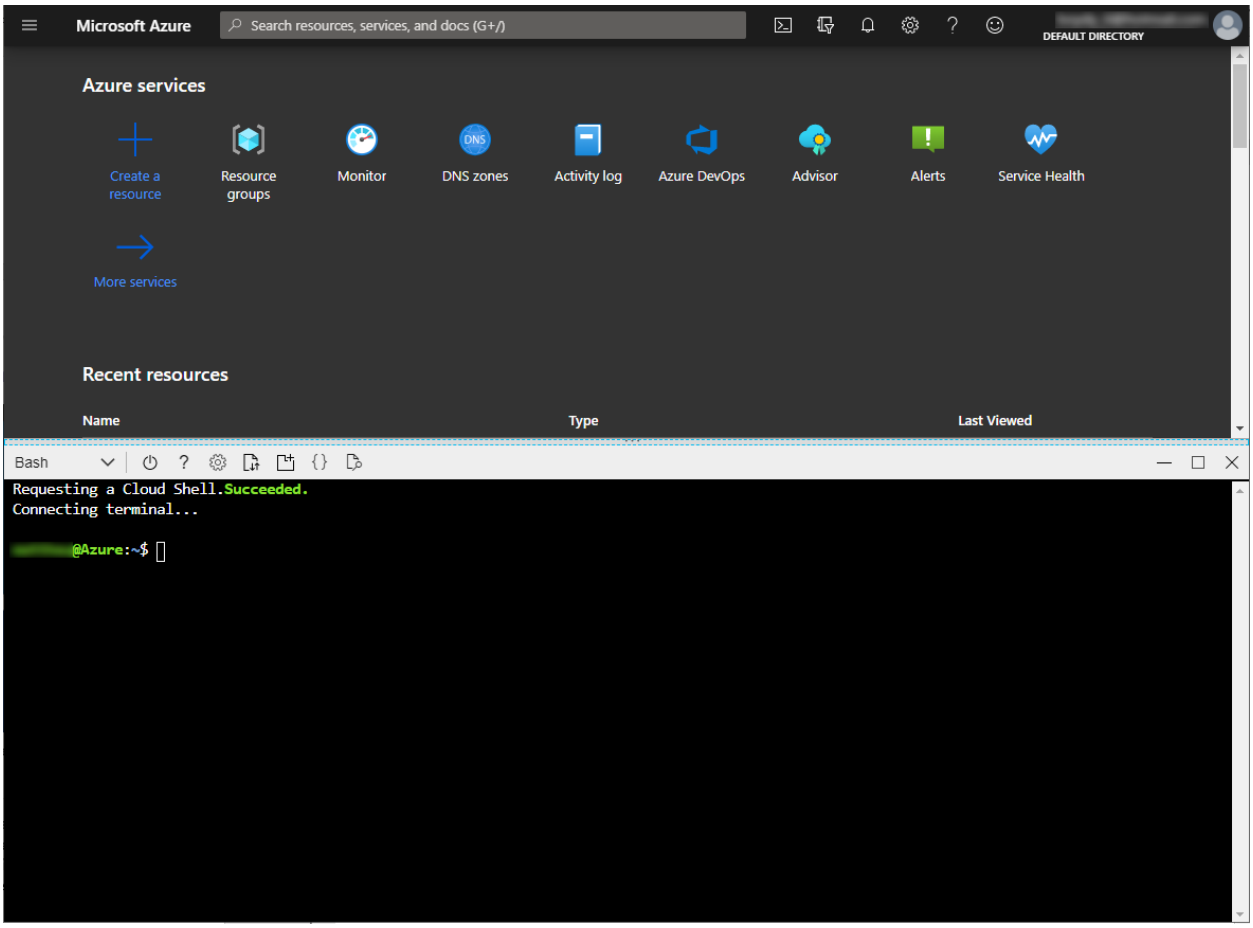
Once you have installed the Azure CLI you can check the version by running the command:

```
az --version
```

Cloud Shell

Before you install Azure CLI locally, it is worth also noting that you can execute Azure CLI commands and scripts from the Azure Cloud Shell. Azure Cloud Shell is a ***browser-based*** shell for managing Azure resources and provides two primary mechanisms for interacting with your Azure environment, either Bash or PowerShell. What this means is that you can execute your Azure CLI commands directly from inside the Azure portal by opening Cloud Shell.

You can also access the shell directly by going to https://shell.Azure.com



Once you open the shell from the portal, you can now execute commands directly with instant authentication since you have already signed into the Azure Portal.

It is also important to know that while the Cloud Shell is temporary in nature, if you wish to store any scripts you create, you can do so by mounting the "cloud drive" share. You will notice the very first time you open Cloud Shell, it will prompt you to create a resource group,

storage account, and Azure file share. This only needs to happen the first time you open Cloud Shell and will then be automatically attached to every subsequent session you open up.

Some key concepts you should also be aware of are:

- Cloud Shell runs on a temporary host provided on a **per-session**, **per-user basis**.
- Cloud Shell times out after **20 minutes** without interactive activity.
- Cloud Shell **requires an Azure file share** to be mounted.
- Cloud Shell uses the **same Azure file share** for both **Bash** and **PowerShell**.
- Cloud Shell is assigned **one machine per user account**.
- Cloud Shell persists **$HOME using a 5-GB image** held in your file share.

Accounts and Subscriptions

Before you perform any tasks within Azure, it is important to ensure you know how to connect and disconnect from your Azure account. You will not be able to run subsequent commands if you are not connected to an Azure Account.

Azure Accounts

Task	Command	Additional Explanation
Connect to Azure Account	`az login`	Connects to Azure with your Azure account to allow use of Azure Resource Manager Cmdlets. Note: Upon entering this command, you will be redirected to https://microsoft.com/devicelogin and presented with a popup window to complete your login process and any MFA requirements.
Disconnect from Azure Account	`az logout`	Terminates the session, disconnecting you from your Azure account

Subscriptions

Once you are connected to your account, you will often want to work on or in the context of an Azure subscription where you will be placing resources. These commands show you how to retrieve a list of all your subscriptions, look at specific subscriptions in a tenant, and then select the subscription you want to work on.

List all subscriptions the account can access	`az account list`	
Choose subscription	`az account set --subscription "My Subscription"`	Subscription parameter can be either Subscription ID or Subscription Name

Create New Subscription from Enrollment account (Requires EA)	For this command you will need to add an extension to azure cli first. (The subscription extension is still in preview at time of writing)
	
	`az extension add --name subscription`
	
	`az account create --offer-type "MS-XXX-00000" --display-name "My Subscription" --enrollment-account-object-id "<enrollmentAccountObjectId>"`
		Advanced command used to create subscriptions off a specific enrollment account which has permissions to do so. Typically used in larger enterprises.

Resource Groups

Now that we can login to our Account and select subscriptions, we are ready to work with resources. First let's look at the commands to list out resources in our environment. These are also fantastic commands to practice with as the risk of doing anything unwanted is extremely low since you are just pulling information about your environment.

Retrieving Resource Groups

Task	Command	Additional Explanation
List All Resource Groups	`az group list`	Lists all resource group.
Retrieve Specific Resource Group	`az group show --name "SkylinesRG”`	Used to find a specific resource group based on name.
List All Resource Groups based on string	`az group list --query "[?contains(name, ‘Skylines’)].[name]"`	Used to find a specific resource group based on a string, but does not need to have an exact match.
Resource Group by Location	`az group list --query "sort_by([], &location)"`	Find resource groups based on Azure region.
Resource Group by Location (Formatted as Table)	`az group list --query "sort_by([], &location)" --output table`	As above with extra formatting added on.

Resource Group Provisioning & Management

Now we know how to look at resource groups, let's look at how we can create new resource groups.

Task	Command	Additional Explanation
Create a new Resource Group	`az group create --location 'northcentralus' --name 'SkylinesRG'`	Creates a new resource group in North Central called "SkylinesRG"
Delete a Resource Group	`az group delete --name "SLRGToDelete"`	Removes a resource group and all resources contained inside it

Note: Remember even though we have to define a regional location for a resource group, this is purely to store the meta data. The resources inside the resource group can be deployed in other regions. The resource group is a management construct allowing us to enforce RBAC rights and also group resources by application lifecycle.

Resource Group Tags

As you know, the resource group itself is a management construct allowing us to enforce RBAC rights and store meta data about the resources within the group. One of the key pieces of meta data are tags.

They allow us to perform the following commands:

Task	Command	Additional Explanation
Display Tags associated with a specific **resource group** name	`az group show --name "SkylinesRG" --query [tags]`	Lists out all the tags for a specific resource group.
To get all Azure **resource groups** with a specific tag	`az group list --tag 'Owner=Skylines Academy'`	Helps you locate all the resource groups with a specific tag assigned.
To get specific **resources** with a specific tag	`az resource list --tag "Dept=Finance"`	

Adding Tags

Task	Command	Additional Explanation
Add Tags to an existing resource group that has no tags	`az group update --name SkylinesRG --tags Dept=IT Environment=Test`	Used when your resource group has NO tags currently assigned. Be careful as not to override existing tags.

Adding tags to an existing resource group that has tags
1. Get Tags
2. Modify Format
3. Update/Apply Tags
	`tags=$(az group show --resource-group slresourcegroup --query tags)`
	
	`tags=$(echo $tags | tr -d '"{},' | sed 's/: /=/g')`
	
	`az group update --resource-group slresourcegroup --tags $tags Status=Approved`
		Used when your resource group already has **some** existing tags.
		
		Finds JSON record for current tags, removes quotation marks, curly brackets and commas and then replaces colons with equal signs.

Add tags to a specific resource that has no tags	`az resource tag --resource-group examplegroup --name examplevnet --tag Dept="IT Environment="Test" --resource-type Microsoft.Network/virtualNetworks`	Tag specific resources that have NO tags currently assigned. Be careful as not to override existing tags.

Apply all tags from an existing resource group to the resources beneath.	`tags=$(az group show --resource-group slresourcegroup --query tags)`
	
	`tags=$(echo $tags | tr -d '"{},' | sed 's/: /=/g')`
	
	`az resource tag --resource-group examplegroup --name examplevnet --tag $tags --resource-type Microsoft.Network/virtualNetworks`
		Takes the resource group tags at the parent RG and assigns them to resource inside the RG.

Apply all tags from an existing resource group to the resources beneath. Keeping existing resource level tags	`rgtags=$(az group show --resource-group slresourcegroup --query tags)`
	
	`tags=$(az resource show --resource-group slresourcegroup --name vmname --resource-type Microsoft.Compute/virtualMachines --query tags)`
	
	`tags=$(echo $rgtags $tags | tr -d '"{},' | sed 's/: /=/g')`
		Takes the resource group tags and adds them to the resource keeping currently assigned tags too.

	`az resource tag --resource-group slresourcegroup --name vmname --tag $tags --resource-type Microsoft.Compute/virtualMachines`	

Remove All Tags (Caution)

You may want to remove tags. If you need to, you can use this command, but do so with caution. This command will erase all the tags from your resource groups, and it is not recoverable.

Removes all tags by passing an empty hash	`az group update --name SkylinesRG --tags`	Removes all tags. Use with caution!

Resources within RGs

What we are often concerned with are specific resources inside our Resource Group. In the cloud world, it is not uncommon for resources to grow quickly and get out of control. These initial commands will help you find what you have inside your Resource Group, as well as locate resources of specific types.

The following two commands are also very safe to use as they are focused on listing out resources in your environment.

Task	**Command**	**Additional Explanation**
Find resources in a resource **group** with a specific name	`az resource list --resource-group "SkylinesRG"`	Find resources in a resource group.

Moving Resources from One Resource Group to Another

Moving resources between resource groups is another common task that you might be expected to perform. It is certainly possible from the portal, but can get tedious when you have a lot of moves to complete.

Note: There are also some restrictions with Azure when moving resources between groups and you may get an error when moving certain types.

Task	**Command**	**Additional Explanation**
Step 1: Retrieve existing Resource	`az resource list --resource-group SL-OldRG --resource-type Microsoft.Storage/storageAccounts --name SkylinesStorageAccount --query "[].id" -o tsv`	# Retrieves a storage account called "SkylinesStorageAccount"
Step 2: Move the Resource to the New Group	`az resource move --destination-group SL-NewRG --ids $(az resource list --resource-group SL-OldRG --resource-type Microsoft.Storage/storageAccounts --name SkylinesStorageAccount --query "[].id" -o tsv)`	# Uses command in step 1 to move the resource from Step 1 into the destination resource group "SL-NewRG"

Governance

Governance has become critical for users of the cloud. As mentioned before, resources can get out of control very quickly and having a complete governance strategy for your environment is essential.

Thankfully, Microsoft provides a number of free mechanisms to help enforce policies in your environment. In addition, you should also check out the Microsoft Virtual Datacenter Guide. This guide includes a complete approach for managing your Enterprise environment and is a must for large organizations.

Either way, you will need to know about Azure policies and how they can be used to help maintain your environment. Administering them is a daily job of any Azure Administrator as you deal with the demands from application teams with varying requirements.

Azure Policies: View Policies and Assignments

These commands allow you to look at your existing policies and assignments:

Task	Command	Additional Explanation
See all policy definitions in your subscription	`az policy definition list`	Find all your existing policy definitions. You can then assign these to resource groups or subscriptions.
Retrieve assignments for a specific resource group	`az policy assignment list --resource-group ExampleGroup`	Look up all your existing assignments for a specific resourcegroup. Using -Name allows you to narrow down to a specific policy assignment.

Creating Policies

This is a two-step process. First, you need to create your policies in JSON syntax, and then create a definition from them. In step 2 below, you will see two options for referencing the JSON policy, either via GitHub repository, or via a local file on your desktop. I encourage you to test this out and think about how you will go about managing your policy templates long term in your organization.

Task	Command	Additional Explanation
Step 1: Create JSON Policy	Create the policy in JSON	See JSON policies - https://docs.microsoft.com/en-us/azure/governance/policy/tutorials/create-custom-policy-definition

Step 2: Create Policy Definition (Local Reference)	`az policy definition create --name denyCoolTiering --description "Deny cool access tiering for storage" --rules ~\policies\coolAccessTier.json`	Pass your JSON file using local file
Step 2: Create Policy Definition (Code Repo Reference)	`az policy definition create --name denyRegions --description " Deny specific regions " --rules 'https://githublocation.com/azurepolicy.rules.json'`	Pass your JSON file using GitHub

Assign Policies

Now it is time to apply our policy. This involves ASSIGNING the policy to a resource group or subscription. In this example, we first retrieve our resource group and store it as a variable, then reference that variable when creating the policy assignment.

Task	**Command**	**Additional Explanation**

Assign Azure Policy	`policyName=$(az policy definition list --query "[?displayName == 'Allowed locations'].name" --output tsv)`
	
	`az policy assignment create --resource-group slresourcegroup --name allowedLocations --policy $policyName --params "{ \"listofAllowedLocations\": { \"value\": [ \"northeurope\", \"eastus\", \"westeurope\" ] } }"`
		Creates a new policy assignment to a resource group you specified by name. The policy assigned is the built-in "Allowed locations" policy

Resource Locks

Task	**Command**	**Additional Explanation**
Create a new resource lock	`az resource lock create --lock-type ReadOnly --notes "Notes about the lock" --name SL-WebSiteLock --resource-name SL-WebSite --resource-type Microsoft.Web/sites --resource-group SL-RGWebSite`	Creates a new resource lock on a specific resource. In this example, it creates a new ReadOnly resource lock on a website resource.
Retrieve a resource lock	`az resource lock show --name SL-WebSiteLock --resource-name SL-WebSite --resource-type Microsoft.Web/sites --resource-group SL-RGWebSite`	Look up a specific resource lock.

Storage

Retrieving Storage Accounts

Lists all storage accounts in the current subscription	`az storage account list`	Find all of your storage accounts – you will probably have a lot in a large environment.

Create Storage Account

Task	**Command**	**Additional Explanation**
Create Storage Account Requires the resource group name, storage account name, valid Azure location, and type (SkuName).	`az storage account create --name slstorage1 --resource-group slstoragerg --location eastus --sku Standard_LRS`	Creates a new storage account in a resource group. You specific the region and storage account SKU to decide on the type of account.
SKU Options	• `Standard_LRS`. Locally-redundant storage. • `Standard_ZRS`. Zone-redundant storage. • `Standard_GRS`. Geo-redundant storage. • `Standard_GZRS`, Geo-Zone redundant storage • `Standard_RAGRS`. Read access geo-redundant storage. • `Standard_RAGZRS`, Read access geo-zone redundant storage. • `Premium_LRS`. Premium locally-redundant storage. • `Premium_ZRS`, Premium zone-redundant storage.	
Optional Key Parameters	`--kind` The kind parameter will allow you to specify the type of Storage Account. • `Storage` - General purpose Storage account that supports storage of Blobs, Tables, Queues, Files and Disks. • `StorageV2` - General Purpose Version 2 (GPv2) Storage account that supports Blobs, Tables, Queues, Files, and Disks, with advanced features like data tiering. • `BlobStorage` -Blob Storage account which supports storage of Blobs only. The default value is Storage. • `BlockBlobStorage`	

	• `FileStorage` `--access-tier` Some storage account types allow for specification of access-tier for use with lower access frequency data. • `Hot` • `Cool`	
Create a storage container in a storage account	`az storage container create –name slcontainer --account-name slstorageaccount`	

Remove Accounts and Containers

Task	**Command**	**Additional Explanation**
Delete a storage account	`az storage account delete --name mystorageaccountname --resource-group resourceGroup`	Deletes your storage account
Delete a storage container using storage account name and container name	`az storage container delete --name "slcontainer" --account-name "slstorageaccount"`	

Deploy and Manage Virtual Machines

Get Information about VMs

Task	Command	Additional Explanation
List all VMs in current subscription	`az vm list`	Get all of your Azure virtual machines.
List VMs in a resource group (See Resource Groups section above)	`az vm list --resource-group "MyResourceGroup”`	List all of your VMs inside a specific resource group.
Get a specific virtual machine	`az vm show --resource-group “slresourcegroup” --name MyVm`	Find a specific VM by its name inside of a resource group.

Create a VM – Simplified

We put this command here as it is a quick way to create a VM, but you are far better off using VM configurations to create your VMs with more specific parameters applied. Try out both of them and you will see the difference.

Task	Command	Additional Explanation
Create a simple Windows Server 2019 VM	`az vm create --name vmname --resource-group slresourcegroup --image Win2019Datacenter`	Typing in this simple command will create a VM and populate names for all the associated objects based on the VM name specified.

Create a VM Configuration Before Creating the Virtual Machine

Use the following tasks to create a new VM configuration before creating your Virtual Machine based on that config.

Task	Command	Additional Explanation
Create a VM with a specific size	`az vm create --name vmname --resource-group slresourcegroup --image Win2019Datacenter --size Standard_D1_v2`	
Create Ubuntu VM with auto-generated SSH keys	`az vm create --name vmname --resource-group slresourcegroup --image UbuntuLTS --generate-ssh-keys`	
Create VM on an existing virtual network	`az vm create --name vmname --resource-group slresourcegroup --image Win2019Datacenter --vnet-name slVNet --subnet slSubnet1 --admin-password Password1234`	
Create a VM without a public IP address	`az vm create --name vmname --resource-group slresourcegroup --public-ip-address "" --image Win2012R2Datacenter`	

VM Operations

Task	Command	Additional Explanation
Start a VM	`az vm start --resource-group MyResourceGroup --name MyVm`	Power On
Stop a VM	`az vm stop --resource-group MyResourceGroup --name MyVm`	Power Off, billing will continue for the virtual machine.
Restart a running VM	`az vm restart --resource-group MyResourceGroup --name MyVm`	Soft Restart
Deallocate a VM	`az vm deallocate --resource-group MyResourceGroup --name MyVm`	Deallocates the VM's resources and stops the Billing cycle for this VM.
Delete a VM	`az vm delete --resource-group MyResourceGroup --name MyVm`	Destroys the VM

Networking

Get/List Networking

Task	Command	Additional Explanation
List virtual networks	`az network vnet list --resource-group "slresourcegroup"`	Lists all the virtual networks in the resource group.
Get information about a virtual network	`az network vnet show --resource-group slresourcegroup --name MyVNet`	Retrieves details of a specific VNET by name.
List subnets in a virtual network	`az network vnet subnet list --resource-group slresourcegroup --name MyVNet`	Filters down to the subnets inside of the VNET
Get information about a subnet	`az network vnet subnet show --resource-group slresourcegroup --name mySubnet1 --vnet-name MyVNet`	Gets information about the subnet in the specified virtual network. The $vnet value represents the object returned by Get-AzVirtualNetwork you used previously.
Get all IP addresses from a resource group	`az network public-ip list --resource-group SkylinesRG`	
Get all load balancers from a resource group	`az network lb list --resource-group SkylinesRG`	
Get all network interfaces from a resource group	`az network nic list --resource-group SkylinesRG`	
Get information about a network interface	`az network nic show --resource-group SkylinesRG --name slNIC`	
Get the IP configuration of a network interface	`az network nic ip-config list --resource-group SkylinesRG --nic-name slNIC`	

Create Network Resources

Task	Command	Additional Explanation
Create a virtual network	`az network vnet create --name slVNet --resource-group SkylinesRG --address-prefixes XXX.XXX.XXX.XXX/YY`	
Create subnet configurations	`az network vnet subnet create --address-prefixes XXX.XXX.XXX.XXX/YY --name slSubnet1 --resource-group SkylinesRG --vnet-name slVNet`	
Create a virtual network with subnet on creation	`az network vnet create --name slVNet --resource-group SkylinesRG --address-prefixes XXX.XXX.XXX.XXX/YY --subnet-name slSubnet1 --subnet-prefix XXX.XXX.XXX.XXX/YY`	
Create a public IP address	`az network public-ip create --name myPublicIp --resource-group slresourcegroup --dns-name skylines --locations northcentralus --allocation-method Dynamic`	The public IP address uses the domain name that you previously tested and is used by the frontend configuration of the load balancer.
Create a load balancer	`az network lb create --name myLoadBalancer --resource-group slresourcegroup`	
Create a frontend IP configuration	`az network lb frontend-ip create --lb-name myLoadBalancer --name myFrontendIP --resource-group slresourcegroup --public-ip-address myPublicIp`	The frontend configuration includes the public IP address that you previously created for incoming network traffic.

Create a backend address pool	`az network lb address-pool create --lb-name myLoadBalancer --name myBackendAddressPool --resource-group slresourcegroup`	Provides internal addresses for the backend of the load balancer that are accessed through a network interface.
Create a probe	`az network lb probe create --lb-name myLoadBalancer --name myProbe --port 80 --protocol Http --resource-group slresourcegroup --interval 15 --path HealthProbe.aspx`	Contains health probes used to check availability of virtual machines instances in the backend address pool.
Create a load balancing rule	`az network lb rule create --backend-port 80 --frontend-port 80 --lb-name myLoadBalancer --name HTTP --protocol Tcp --resource-group slresourcegroup --probe-name myProbe --frontend-ip-name myFrontendIP --backend-pool-name slBackendAddressPool`	Contains rules that assign a public port on the load balancer to a port in the backend address pool.
Create an inbound NAT rule	`az network lb inbound-nat-rule create --backend-port 3389 --frontend-port 3441 --lb-name myLoadBalancer --name myInboundRule1 --protocol Tcp --resource-group slresourcegroup –frontend-ip-name myFrontendIP`	Contains rules mapping a public port on the load balancer to a port for a specific virtual machine in the backend address pool.
Create a network interface	`az network nic create --name myNIC --resource-group slresourcegroup --subnet slSubnet1 --lb-inbound-nat-rules myInboundRule1 --lb-address-pools myBackendAddressPool --private-ip-address XXX.XXX.XXX.XXX --lb-name myLoadBalancer --vnet-name slVNet`	Create a network interface using the public IP address and virtual network subnet that you previously created.

Remove Network Resources

Task	Command	Additional Explanation
Delete a network interface	`az network nic delete --resource-group slresourcegroup --name myNIC`	Removes the specified network interface from the resource group.
Delete a virtual network	`az network vnet delete --resource-group slresourcegroup --name myVNet`	Removes the specified virtual network from the resource group.
Delete a load balancer	`az network lb delete --resource-group slresourcegroup --name myLoadBalancer`	Removes the specified load balancer from the resource group.
Delete a public IP address	`az network public-ip delete --resource-group slresourcegroup --name myIPAddress`	Removes the specified public IP address from the resource group.

Azure Active Directory Commands

User and Service Principal Management

Task	Command	Additional Explanation
Get all users	`az ad user list`	
Get specific user	`az ad user list --upn "user@skylinesexam.com"`	
Remove User	`az ad user delete --id "user@skylinesexam.com"`	
New User Creation	`az ad user create --display-name "New User" --password "Password1234" --user-principal-name user@skylinesexam.com --force-change-password-next-login true --mail-nickname "Newuser"`	This is a three-step process that requires first creating a password profile, setting the password, and then passing these into the New-AzureADUser command

Service Principal Creation	First you need to create your application registration in AzureAD then you retrieve it with this command.
	
	`az ad app list --display-name "SLAppRegistrationName"`
	
	Once you have the application ID for the App registration, you can use it to create the SPN (Service Principal)
	
	`az ad sp create –id <ID>`	

Assign Role	`az role assignment create --assignee 11111111-1111-1111-1111-11111111111 --resource-group slresourcegroup --role Reader`	This will be scoped to the resource group name you type in with the role definition assigned to the SPN --assignee can be objectId, sign-in name or service principal name.
View Current Role Assignment	`az role assignment list --assignee 11111111-1111-1111-1111-11111111111 --resource-group slresourcegroup`	