Introduction

Completed 100 XP

- 3 minutes

The Azure portal is great for performing single tasks and for seeing a quick overview of the state of your resources. However, for tasks that need to be repeated daily, or even hourly, using the command line and a set of tested commands or scripts can help get your work done more quickly and avoid errors.

Suppose you work at a company that develops Azure Web Apps. These are applications hosted in Azure, with all the benefits of automatically configured security, load balancing, management, and so on. You're currently testing a web app that generates sales forecasts based on a range of inputs from different databases and other data sources. Your developers use Windows, Linux, and Mac computers, and use a GitHub repository for daily builds of the applications.

As part of the testing, you want to compare app performance for different data sources and for different types of data connections. You've noticed that when your development team uses the Azure portal to create a new test instance of the app, they don't always use exactly the same parameters. You plan to solve this problem by using a set of standard deployment commands for each app test, which can be automated if required, and which will work in the same way across all the computers used by your software team.

In this module, you'll see how to manage Azure resources using the Azure CLI.

Learning objectives

In this module, you'll:

- Install the Azure CLI on Linux, macOS, and/or Windows.
- Connect to an Azure subscription using the Azure CLI.
- Create Azure resources using the Azure CLI.

Prerequisites

- Experience with a command-line interface, such as PowerShell or Bash
- Knowledge of basic Azure concepts, such as resource groups
- Experience administering Azure resources using the Azure portal

Next unit: What is the Azure CLI?

What is the Azure CLI?

Completed100 XP

- 3 minutes

The Azure CLI is a command-line program to connect to Azure and execute administrative commands on Azure resources. It runs on Linux, macOS, and Windows, and allows administrators and developers to execute their commands through a terminal or command-line prompt (or script!) instead of a web browser. For example, to restart a virtual machine (VM), you would use the following command:

Azure CLICopy

```
az vm restart -g MyResourceGroup -n MyVm
```

The Azure CLI provides cross-platform command-line tools for managing Azure resources, and you can easily install it locally on Linux, Mac, or Windows computers. You can also use the Azure CLI from a browser through the Azure Cloud Shell. In both cases, you can use it interactively or scripted. For interactive use, you'll first launch a shell (such as cmd.exe on Windows or Bash on Linux or macOS), then issue the command at the shell prompt. To automate repetitive tasks, you'll assemble the CLI commands into a shell script using the script syntax of your chosen shell, then execute the script.

How to install the Azure CLI

On both Linux and macOS, you'll use a package manager to install the Azure CLI. The recommended package manager differs by OS and distribution:

- Linux: **apt-get** on Ubuntu, **yum** on Red Hat, and **zypper** on OpenSUSE
- Mac: **Homebrew**

The Azure CLI is available in the Microsoft repository, so you'll first need to add that repository to your package manager.

On Windows, you can install the Azure CLI by downloading and running an MSI file.

Using the Azure CLI in scripts

If you want to use the Azure CLI commands in scripts, you need to be aware of any issues around the "shell" (or environment) used for running the script. For example, in a Bash shell, you'll use this syntax when setting variables:

Azure CLICopy

```
variable="value"
variable=integer
```

If you use a PowerShell environment for running Azure CLI scripts, you'll use this syntax for variables:

PowerShellCopy

```
$variable="value"
$variable=integer
```

You must install the Azure CLI before you can use it to manage Azure resources from a local computer. The installation steps vary for Windows, Linux, and macOS, but once installed, the commands are common across platforms.

Next unit: Exercise - Install and run the Azure CLI

Exercise - Install and run the Azure CLI

Completed100 XP

- 10 minutes

Choose your platform

Linux Mac Windows

Let's install the Azure CLI on your local machine, then run a command to verify your installation. The method you use for installing the Azure CLI depends on your computer's operating system. Choose the steps for your operating system.

Note

This exercise guides you through installing the Azure CLI tool locally. The remainder of the module will use the Azure Cloud Shell so you can leverage the free subscription

support in Microsoft Learn. You can consider this exercise as an optional activity and just review the instructions if you prefer.

Windows

Here, you'll install the Azure CLI on Windows using the MSI installer.

1. Go to https://aka.ms/installazurecliwindows, and in the browser security dialog box, select **Run** or **Open file**.
2. In the installer, accept the license terms, then select **Install**.
3. In the **User Account Control** dialog, select **Yes**.

Running the Azure CLI

You run the Azure CLI by opening a bash shell (Linux and macOS), or from the command prompt or PowerShell (Windows).

1. Start the Azure CLI and verify your installation by running the version check.

   Azure CLICopy
   ```
   az --version
   ```

Tip

Running the Azure CLI from PowerShell has some advantages over running the Azure CLI from the Windows command prompt. PowerShell provides more tab completion features than those available from the command prompt.

You've set up your local machines to administer Azure resources with the Azure CLI. You can now use the Azure CLI locally to enter commands or execute scripts. The Azure CLI will forward your commands to the Azure datacenters, where they'll run inside your Azure subscription.

Next unit: Work with the Azure CLI

Work with the Azure CLI

Completed 100 XP

- 5 minutes

The Azure CLI lets you type commands and execute them immediately from the command line. Recall that the overall goal in the software-development example is to deploy new builds of a web app for testing. Let's talk about the sorts of tasks you can do with the Azure CLI.

What Azure resources can be managed using the Azure CLI?

The Azure CLI lets you control nearly every aspect of every Azure resource. You can work with resource groups, storage, virtual machines, Azure Active Directory (Azure AD), containers, machine learning, and so on.

Commands in the CLI are structured in *groups* and *subgroups*. Each group represents a service provided by Azure, and the subgroups divide commands for these services into logical groupings. For example, the `storage` group contains subgroups including **account**, **blob**, and **queue**.

So, how do you find the particular commands you need? One way is to use `az find`, the AI robot that uses the Azure documentation to tell you more about commands, the CLI, and more.

Example: find the most popular commands related to the word **blob**.

Azure CLICopy

```
az find blob
```

Example: Show me the most popular commands for an Azure CLI command group, such as `az vm`.

Azure CLICopy

```
az find "az vm"
```

Example: Show me the most popular parameters and subcommands for an Azure CLI command.

Azure CLICopy

```
az find "az vm create"
```

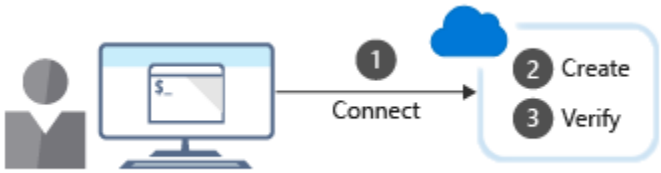
If you already know the name of the command you want, the `--help` argument for that command will get you more detailed information on the command, and a list of the available subcommands for a command group. So, with our storage example, here's how you can get a list of the subgroups and commands for managing blob storage:

Azure CLICopy

```
az storage blob --help
```

How to create an Azure resource

When you're creating a new Azure resource, there are typically three steps: connect to your Azure subscription, create the resource, and verify that creation was successful. The following illustration shows a high-level overview of the process.



Each step corresponds to a different Azure CLI command.

Connect

Since you're working with a local install of the Azure CLI, you'll need to authenticate before you can execute Azure commands by using the Azure CLI **login** command.

Azure CLICopy

```
az login
```

The Azure CLI will typically launch your default browser to open the Azure sign-in page. If this doesn't work, follow the command-line instructions and enter an authorization code at https://aka.ms/devicelogin.

After a successful sign-in, you'll be connected to your Azure subscription.

Create

You'll often need to create a new resource group before you create a new Azure service, so we'll use resource groups as an example to show how to create Azure resources from the CLI.

The Azure CLI **group create** command creates a resource group. You must specify a name and location. The name must be unique within your subscription. The location determines where the metadata for your resource group will be stored. You use strings like "West US", "North Europe", or "West India" to specify the location; alternatively, you can use single-word equivalents, such as westus, northeurope, or westindia. The core syntax is:

Azure CLICopy

```
az group create --name <name> --location <location>
```

Important

You don't need to create a resource group when using the free Azure sandbox. Instead, you'll use a pre-created resource group.

Verify

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

Azure CLICopy

```
az group list
```

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

Azure CLICopy

```
az group list --output table
```

Next unit: Exercise - Create an Azure website using the CLI

Exercise - Create an Azure website using the CLI

Completed 100 XP

- 10 minutes

38% Creating resource group...

Next, let's use the Azure CLI to create a resource group, then to deploy a web app into this resource group.

Using a resource group

When you're working with your own machine and Azure subscription, you'll need to first sign in to Azure using the `az login` command. However, signing in is unnecessary when you're using the browser-based Cloud Shell sandbox environment.

Next, you'd normally create a resource group for all your related Azure resources with an `az group create` command, but for this exercise, the following resource group has been created for you: **[sandbox resource group name]**.

1. Your first step in this exercise will be to create several variables that you'll use in later commands.

   BashCopy

   ```
   export RESOURCE_GROUP=[sandbox resource group name]
   export AZURE_REGION=centralus
   export AZURE_APP_PLAN=popupappplan-$RANDOM
   export AZURE_WEB_APP=popupwebapp-$RANDOM
   ```

2. You can ask the Azure CLI to list all your resource groups in a table. There should just be one while you're in the free Azure sandbox.

   Azure CLICopy

   ```
   az group list --output table
   ```

   Tip

   You can use the **Copy** button to copy commands to the clipboard. To paste, right-click on a new line in the Cloud Shell terminal and select **Paste**, or use the Shift+Insert keyboard shortcut (⌘+V on macOS).

3. As you do more Azure development, you can end up with several resource groups. If you have several items in the group list, you can filter the return values by adding a `--query` option. Try the following command:

   Azure CLICopy

   ```
   az group list --query "[?name == '$RESOURCE_GROUP']"
   ```

   The query is formatted using **JMESPath**, which is a standard query language for JSON requests. You can learn more about this powerful filter language at http://jmespath.org/. We also cover queries in more depth in the **Manage VMs with the Azure CLI** module.

Steps to create a service plan

When you run Web Apps using the Azure App Service, you pay for the Azure compute resources that the app uses, and the resource costs depend on the App Service plan associated with your Web Apps. Service plans determine the region used for the app datacenter, number of VMs used, and pricing tier.

1. Create an App Service plan to run your app. The following command specifies the free pricing tier, but you can run `az appservice plan create --help` to see the other pricing tiers.

   Note

   The name of the app and plan must be *unique* in all of Azure. The variables that you created earlier will assign random values as suffixes to make sure they're unique. However, if you receive an error when you're creating any resources, you should run the commands listed earlier to reset all of the variables with new random values.

   If you receive an error about the resource group, run the commands listed earlier with a different resource group value.

   Azure CLICopy

   ```
   az appservice plan create --name $AZURE_APP_PLAN --resource-group
   $RESOURCE_GROUP --location $AZURE_REGION --sku FREE
   ```

   This command can take several minutes to complete.

2. Verify that the service plan was created successfully by listing all your plans in a table.

   Azure CLICopy

   ```
   az appservice plan list --output table
   ```

   You'll get a response like the following example:

   OutputCopy

   ```
   Kind    Location    MaximumNumberOfWorkers    Name
   NumberOfSites    ResourceGroup                                Status
   ------  ----------  ------------------------  ------------------  -------
   --------  ----------------------------------------  --------
   app     Central US  3                         popupappplan-54321  0
   Learn-12345678-1234-1234-1234-123456789abc  Ready
   ```

Steps to create a web app

Next, you'll create the web app in your service plan. You can deploy the code at the same time, but for our example, we'll create the web app and deploy the code as separate steps.

1. To create the web app, you'll supply the web app name and the name of the app plan you created above. Just like the app plan name, the web app name must be unique. The variables that you created earlier will assign random values that should be sufficient for this exercise.

   Azure CLICopy

   ```
   az webapp create --name $AZURE_WEB_APP --resource-group $RESOURCE_GROUP -
   -plan $AZURE_APP_PLAN
   ```

2. Verify that the app was created successfully by listing all your apps in a table.

   Azure CLICopy

   ```
   az webapp list --output table
   ```

   You'll get a response like the following example:

   OutputCopy

   ```
   Name               Location    State    ResourceGroup
   DefaultHostName                     AppServicePlan
   ```

```
----------------  ----------  -------  --------------------------------
---------  ---------------------------------  -----------------
popupwebapp-12345  Central US  Running  Learn-12345678-1234-1234-1234-
123456789abc  popupwebapp-12345.azurewebsites.net  popupappplan-54321
```

Make a note of the **DefaultHostName** listed in the table; this address is the URL for the new website. Azure will make your website available through the unique app name in the `azurewebsites.net` domain. For example, if your app name was "popupwebapp-12345", then your website URL would be: `http://popupwebapp-12345.azurewebsites.net`.

3. Your site has a "quickstart" page created by Azure that you can see either in a browser, or with CURL, just use the **DefaultHostName**:

   BashCopy

   ```
   curl $AZURE_WEB_APP.azurewebsites.net
   ```

   You'll get the default HTML for the sample app returned.

Steps to deploy code from GitHub

1. The final step is to deploy code from a GitHub repository to the web app. Let's use a basic PHP page available in the Azure Samples GitHub repository that displays "Hello World!" when it executes. Make sure to use the web app name you created.

   Azure CLICopy

   ```
   az webapp deployment source config --name $AZURE_WEB_APP --resource-group
   $RESOURCE_GROUP --repo-url "https://github.com/Azure-Samples/php-docs-
   hello-world" --branch master --manual-integration
   ```

2. Once it's deployed, hit your site again with a browser or CURL.

   BashCopy

   ```
   curl $AZURE_WEB_APP.azurewebsites.net
   ```

   The page displays "Hello World!"

   OutputCopy

   ```
   Hello World!
   ```

This exercise demonstrated a typical pattern for an interactive Azure CLI session. You first used a standard command to create a new resource group. You then used a set of commands to deploy a resource (in this example, a web app) into this resource group. You could easily combine this set of commands into a shell script and execute it every time you need to create the same resource.

Next unit: Summary

Summary

Completed200 XP

- 3 minutes

The Azure CLI is a good choice for anyone new to Azure command line and scripting. Its simple syntax and cross-platform compatibility help reduce the risk of errors when performing regular and repetitive tasks. In this module, you used the Azure CLI commands to create a resource group and deploy a web app by using a small set of commands. These commands could be combined into a shell script as part of automation solution.

Clean up

The sandbox automatically cleans up your resources when you're finished with this module.

When you're working in your own subscription, it's a good idea at the end of a project to identify whether you still need the resources you created. Resources that you leave running can cost you money. You can delete resources individually or delete the resource group to delete the entire set of resources.

Check your knowledge

1.

What do you need to install on your machine to let you execute Azure CLI commands locally?

The Azure Cloud Shell

The Azure CLI and Azure PowerShell

Only the Azure CLI

You only need to install the Azure CLI. You will use a shell to issue the CLI commands, but every platform has at least one built-in shell.

2.

True or false: The Azure CLI can be installed on Linux, macOS, and Windows, and the CLI commands you use are the same in all platforms.

True

The CLI is cross-platform and can be installed on Linux, macOS, and Windows. After installation, the CLI commands that you run are the same everywhere. This means you can learn the commands once and use them with any local installation or in the Azure Cloud Shell.

False

3.

Which parameter value can you add to most CLI commands to get concise, formatted output?

list

table

The table parameter formats the output as a table. This can make things much more readable for commands that produce a large amount of output.

group

Explore other modules

- Manage resources in Azure

- AZ-104: Prerequisites for Azure administrators
- Prerequisites for Azure administrators