Console and Cloud Shell

In this lab, you become familiar with the Google Cloud Platform (GCP) web-based interface. There are two integrated environments: a GUI (graphical user interface) environment called the GCP Console, and a CLI (command-line interface) called Cloud Shell. In this class you use both environments.

Here are a few things you need to know about the GCP Console:

- The GCP Console is under continuous development, so occasionally the graphical layout changes. This is most often to accommodate new GCP features or changes in the technology, resulting in a slightly different workflow.
- You can perform most common GCP actions in the GCP Console, but not all actions. In particular, very new technologies or sometimes detailed API or command options are not implemented (or not yet implemented) in the GCP Console. In these cases, the command line or the API is the best alternative.
- The GCP Console is extremely fast for some activities. The GCP Console can perform multiple actions on your behalf that might require many CLI commands. It can also perform repetitive actions. In a few clicks you can accomplish activities that would require a great deal of typing and would be prone to typing errors.
- The GCP Console is able to reduce errors by only offering up through its menus valid options. It is able to leverage access to the platform "behind the scenes" through the SDK to validate configuration before submitting changes. A command line can't do this kind of dynamic validation.

Objectives

In this lab, you learn how to perform the following tasks:

- Get access to GCP.
- Create a Cloud Storage bucket using the GCP Console.
- Create a Cloud Storage bucket using Cloud Shell.
- Become familiar with Cloud Shell features.

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click Start Lab, shows how long Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access the Google Cloud Platform for the duration of the lab.

What you need

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.
  Note: If you already have your own personal GCP account or project, do not use it for this lab.

Task 1: Create a bucket using the GCP Console

In this task, you create a bucket. However, the text also helps you become familiar with how actions are presented in the lab instructions in this class and teaches you about the GCP Console interface.

Navigate to the Storage service and create the bucket

1. In the GCP Console, on the **Navigation menu** (☰), click **Storage** > **Browser**.
2. Click **Create bucket**.
3. For **Name**, type a globally unique bucket name, and leave all other values as their defaults.
4. Click **Create**.

Explore features in the GCP Console

The Google Cloud Platform menu contains a Notifications (🔔) icon. Sometimes, feedback from the underlying commands is provided there. If you are not sure what is happening, check Notifications for additional information and history.

Click *Check my progress* to verify the objective.

Create a bucket using the GCP Console

Check my progress

Task 2: Access Cloud Shell

In this section, you explore Cloud Shell and some of its features.

You can use the Cloud Shell to manage projects and resources via command line, without having to install the Cloud SDK and other tools on your computer.

Cloud shell provides the following:

- Temporary Compute Engine VM
- Command-line access to the instance via a browser
- 5 GB of persistent disk storage ($HOME dir)
- Pre-installed Cloud SDK and other tools
- gcloud: for working with Google Compute Engine and many GCP services
- gsutil: for working with Cloud Storage
- kubectl: for working with Google Container Engine and Kubernetes
- bq: for working with BigQuery
- Language support for Java, Go, Python, Node.js, PHP, and Ruby
- Web preview functionality
- Built-in authorization for access to resources and instances

  After 1 hour of inactivity, the Cloud Shell instance is recycled. Only the /home directory persists. Any changes made to the system configuration, including environment variables, are lost between sessions.

Open Cloud Shell and explore features

1. In the Google Cloud Platform menu, click **Activate Cloud Shell** (). If prompted, click **Continue**. Cloud Shell opens at the bottom of the GCP Console window.

There are three icons to the far right of the Cloud Shell toolbar:

- **Hide/Restore:** The first one hides and restores the window, giving you full access to the GCP Console without closing Cloud Shell.
- **Open in new window:** Having Cloud Shell at the bottom of the GCP Console is useful when you are issuing individual commands. However, sometimes you will be editing files or want to see the full output of a command. For these uses, this icon pops the Cloud Shell out into a full-sized terminal window.
- **Close all tabs:** This icon closes Cloud Shell. Every time you close Cloud Shell, the virtual machine is recycled and all machine context is lost.

2. Close Cloud Shell now.

☐ ıd Shell provides you with which of the following? (Select all that apply).

☐ ımmand-line tool that requires you to install Cloud SDK

☐ 3 of persistent storage (/home)

☐ :-in authorization for access to resources and instances

Command-line access to a free temporary Compute Engine VM

Submit

Task 3: Create a bucket using Cloud Shell

Create a second bucket and verify in the GCP Console

1. Open Cloud Shell again.
2. Use the gsutil command to create another bucket. Replace <BUCKET_NAME> with a globally unique name (you can append a 2 to the globally unique bucket name you used previously):

```
gsutil mb gs://<BUCKET_NAME>
```

3. In the GCP Console, on the **Navigation menu** (☰), click **Storage** > **Browser**, or click **Refresh** if you are already in the Storage Browser. The second bucket should be displayed in the **Buckets** list.

You have performed equivalent actions using the GCP Console and Cloud Shell.

You created a bucket using the GCP Console and another bucket using Cloud Shell.
Click *Check my progress* to verify the objective.

Create a bucket using Cloud Shell

Check my progress

Task 4: Explore more Cloud Shell features

Upload a file

1. Open Cloud Shell.

2. Click the three dots (⋮) icon in the Cloud Shell toolbar to display further options.

3. Click **Upload file**. Upload any file from your local machine to the Cloud Shell VM. This file will be referred to as [MY_FILE].
4. In Cloud Shell, type `ls` to confirm that the file was uploaded.
5. Copy the file into one of the buckets you created earlier in the lab. Replace [MY_FILE] with the file you uploaded and [BUCKET_NAME] with one of your bucket names:

```
gsutil cp [MY_FILE] gs://[BUCKET_NAME]
```

If your filename has whitespaces, ensure to place single quotes around the filename. For example, `gsutil cp ‘my file.txt' gs://[BUCKET_NAME]`
You have uploaded a file to the Cloud Shell VM and copied it to a bucket.

6. Explore the options available in Cloud Shell by clicking on different icons in the Cloud Shell toolbar.
7. Close all the Cloud Shell sessions.

Click *Check my progress* to verify the objective.

Upload a file to Storage bucket

Check my progress

Task 5: Create a persistent state in Cloud Shell

In this section you will learn a best practice for using Cloud Shell. The gcloud command often requires specifying values such as a **Region** or **Zone** or **Project ID**. Entering them repeatedly increases the chances of making typing errors. If you use Cloud Shell a lot, you may want to set common values in environment variables and use them instead of typing the actual values.

Identify available regions

1. Open Cloud Shell from the GCP Console. Note that this allocates a new VM for you.

2. To list available regions, execute the following command:

```
gcloud compute regions list
```

3. Select a region from the list and note the value in any text editor. This region will now be referred to as [YOUR_REGION] in the remainder of the lab.

Create and verify an environment variable

1. Create an environment variable and replace [YOUR_REGION] with the region you selected in the previous step:

```
INFRACLASS_REGION=[YOUR_REGION]
```

2. Verify it with echo:

```
echo $INFRACLASS_REGION
```

You can use environment variables like this in gcloud commands to reduce the opportunities for typos, and so that you won't have to remember a lot of detailed information.

Every time you close Cloud Shell and reopen it, a new VM is allocated, and the environment variable you just set disappears. In the next steps, you create a file to set the value so that you won't have to enter the command each time Cloud Shell is cycled.

Append the environment variable to a file

1. Create a subdirectory for materials used in this class:

```
mkdir infraclass
```

2. Create a file called `config` in the infraclass directory:

```
touch infraclass/config
```

3. Append the value of your Region environment variable to the `config` file:

```
echo INFRACLASS_REGION=$INFRACLASS_REGION >> ~/infraclass/config
```

4. Create a second environment variable for your Project ID, replacing [YOUR_PROJECT_ID] with your Project ID. You can find the project ID on the GCP Console Home page.

```
INFRACLASS_PROJECT_ID=[YOUR_PROJECT_ID]
```

5. Append the value of your Project ID environment variable to the `config` file:

```
echo INFRACLASS_PROJECT_ID=$INFRACLASS_PROJECT_ID >> ~/infraclass/config
```

6. Use the source command to set the environment variables, and use the echo command to verify that the project variable was set:

```
source infraclass/config
echo $INFRACLASS_PROJECT_ID
```

This gives you a method to create environment variables and to easily recreate them if the Cloud Shell is cycled. However, you will still need to remember to issue the source command each time Cloud Shell is opened.

In the next step you will modify the .profile file so that the source command is issued automatically any time a terminal to Cloud Shell is opened.

7. Close and re-open Cloud Shell. Then issue the echo command again:

```
echo $INFRACLASS_PROJECT_ID
```

There will be no output because the environment variable no longer exists.

Modify the bash profile and create persistence

1. Edit the shell profile with the following command:

```
nano .profile
```

2. Add the following line to the end of the file:

```
source infraclass/config
```

3. Press **Ctrl+O**, **ENTER** to save the file, and then press **Ctrl+X** to exit nano.
4. Close and then re-open Cloud Shell to cycle the VM.

5. Use the echo command to verify that the variable is still set:

```
echo $INFRACLASS_PROJECT_ID
```

You should now see the expected value that you set in the config file.

If you ever find your Cloud Shell environment corrupted, you can find instructions on resetting it here:

Resetting Cloud Shell
This will cause everything in your Cloud Shell environment to be set back to its original default state.

To create a persistent state in Cloud Shell, which file would you configure?

○ .my_variables

○ .bashrc

○ .config

○ [✓ .profile]

Submit

Task 6: Review the GCP interface

Cloud Shell is an excellent interactive environment for exploring GCP using Google Cloud SDK commands like `gcloud` and `gsutil`.
You can install the Google Cloud SDK on a computer or on a VM instance in GCP. The gcloud and gsutil commands can be automated using a scripting language like bash (Linux) or Powershell (Windows). You can also explore using the command-line tools in Cloud Shell, and then use the parameters as a guide for re-implementing in the SDK using one of the supported languages.

The GCP interface consists of two parts: the GCP Console and Cloud Shell.

The Console:

- Provides a fast way to get things done

- Presents options to you, instead of requiring you to know them
- Performs behind-the-scenes validation before submitting the commands

  Cloud Shell provides:

- Detailed control
- Complete range of options and features
- A path to automation through scripting