

# Working with the Google Cloud Console and Cloud Shell

## Overview

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

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

- The Cloud Console is under continuous development, so occasionally the graphical layout changes. This is most often to accommodate new Google Cloud features or changes in the technology, resulting in a slightly different workflow.
- You can perform most common Google Cloud actions in the Cloud 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

Cloud Console. In these cases, the command line or the API is the best alternative.

- The Cloud Console is extremely fast for some activities. The Cloud 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 lot of typing and would be susceptible to typing errors.
- The Cloud Console is able to reduce errors by offering only valid options through its menus. It can 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 Google Cloud.
- Use the Cloud Console to create a Cloud Storage bucket.
- Use Cloud Shell to create a Cloud Storage bucket.
- Become familiar with Cloud Shell features.

# Task 1. Use the Cloud Console to create a bucket

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 Cloud Console interface.

## Navigate to the Storage service and create the bucket

1. In the Cloud Console, on the **Navigation menu** () click **Cloud Storage > Bucket**.
2. Click **Create**.
3. For **Name**, type a globally unique bucket name; leave all other values as their defaults.
4. Click **Create**.
5. If prompted `Public access will be prevented`

# 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:

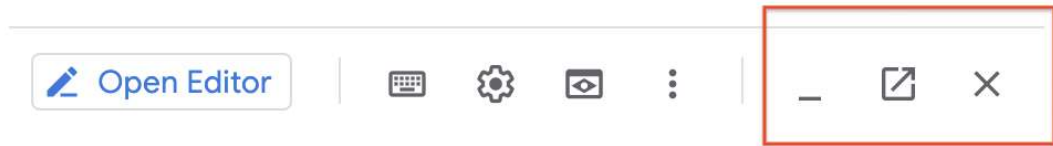
- Temporary Compute Engine VM
- Command-line access to the instance via a browser
- 5 GB of persistent disk storage (\$HOME dir)
- Pre-installed Cloud SDK and other tools
- gcloud: for working with Compute Engine and many Google Cloud services
- gcloud storage: for working with Cloud Storage
- kubectl: for working with Google Kubernetes 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

Learn more about Cloud Shell from the Google Cloud [Cloud Shell Documentation](#).

**Note:** 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 menu, click **Activate Cloud Shell** (  ). If prompted, click **Continue**. Cloud Shell opens at the bottom of the Cloud Console window.  
There are three buttons on the far right of the Cloud Shell toolbar:



sutil

- **Minimize/Restore:** The first one minimizes or restores the window, giving you full access to the Cloud Console without closing Cloud Shell.
- **Open in a new window:** Having Cloud Shell at the bottom of the Cloud 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 button pops the Cloud Shell out into a full-sized terminal window.
- **Close terminal:** This button closes Cloud Shell. Every time you close Cloud Shell, the virtual machine is reset and all machine context is lost.

## Task 3. Use Cloud Shell to create a Cloud Storage bucket

Create a second bucket and verify in the Cloud Console

1. Open Cloud Shell again.
2. Use the `gcloud storage` 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**):

```
gcloud storage buckets create gs://<BUCKET_NAME>
```

Copied!

content\_copy

3. If prompted, click **Authorize**.
4. In the Cloud Console, on the **Navigation menu** (≡), click **Cloud Storage > Bucket**, or click **Refresh** if you are already in the Storage browser. The second bucket should be displayed in the **Buckets** list.

**Note:** You have performed equivalent actions using the Cloud Console and Cloud Shell. You created a bucket using the Cloud Console and another bucket using Cloud Shell.

## Task 4. Explore more Cloud Shell features

### Upload a file

1. Open Cloud Shell.
2. Click the **More** button (⋮) in the Cloud Shell toolbar to display further options.
3. Click **Upload**. 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:

```
gcloud storage cp [MY_FILE] gs://[BUCKET_NAME]
```

Copied!

content\_copy

If your filename has whitespaces, be sure to place single quotes around the filename. For example, `gcloud storage cp 'my file.txt' gs://[BUCKET_NAME]`

**Note:** 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.

## 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 you to specify values such as a **Region**, **Zone**, or **Project ID**. Entering them repeatedly increases the chance of making typing errors. If you use

Cloud Shell frequently, 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 Cloud Console. Note that this allocates a new VM for you.
2. To list available regions, execute the following command:

```
gcloud compute regions list
```

Copied!

content\_copy

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]
```

Copied!

content\_copy

2. Verify it with echo:

```
echo $INFRACLASS_REGION
```



Copied!

content\_copy

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.

**Note:** 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 reset.

## Append the environment variable to a file

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

```
mkdir infraclass
```

Copied!

content\_copy

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

```
touch infraclass/config
```

Copied!

content\_copy

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

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

Copied!

content\_copy

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 Cloud Console Home page.

```
INFRACCLASS_PROJECT_ID=[YOUR_PROJECT_ID]
```

Copied!

content\_copy

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

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

Copied!

content\_copy

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 $INFRACCLASS_PROJECT_ID
```

Copied!

content\_copy

**Note:** This gives you a method to create environment variables and to easily recreate them if the Cloud Shell is recycled or reset. However, you will still need to remember

to issue the source command each time Cloud Shell is opened. In the next step, you modify the .profile file so that the source command is issued automatically every 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
```

Copied!

content\_copy

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
```

Copied!

content\_copy

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

```
source infraclass/config
```

Copied!

content\_copy

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 reset the VM.

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

```
echo $INFRACLASS_PROJECT_ID
```

Copied!

content\_copy

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

**Note:** If your Cloud Shell environment is ever corrupted, instructions on resetting it are in the Cloud Shell Documentation article titled [Disabling or Resetting Cloud Shell](#). This will cause everything in your Cloud Shell environment to be set back to its original default state.

## Task 6. Review the Google Cloud interface

Cloud Shell is an excellent interactive environment for exploring Google Cloud by using Google Cloud SDK commands like `gcloud` and `gcloud storage`.

You can install the Google Cloud SDK on a computer or on a VM instance in Google Cloud. The `gcloud` and `gcloud storage` commands can be automated by 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 an implementation guide in the SDK using one of the supported languages.

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

The Console:

- Provides a fast way to perform tasks.
- 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
- A complete range of options and features
- A path to automation through scripting