# The Apps Script CLI - clasp

**GSP239**

Google Cloud Self-Paced Labs

## Introduction

The Apps Script CLI, or `clasp`, is a tool that lets you to create, edit, and deploy Apps Script projects locally. It allows you to create and publish web apps and add-ons for products like Sheets, Docs, Forms, and Slides from the command line. There are two ways you can develop Apps Script, using script.google.com or locally on your computer. In this lab you will learn how to use `clasp`, the command line tool for Apps Script, to update script.google.com.

## Features

- **Develop Locally**. `clasp` lets you write code on your own computer and upload it to Apps Script when you're done. You can also download existing Apps Script projects and then edit them locally. Once the code is local, you can use your favorite development tools like git to work on Apps Script projects.
- **Manage Deployment Versions**. Create, update, and view multiple deployments of your project.

- **Structure Code**. `clasp` automatically converts your flat project on script.google.com into **folders**. Example (do not copy):

```
# On script.google.com:
├── tests/slides.gs
└── tests/sheets.gs

# Locally:
├── tests/
│    ├─ slides.js
│    └─ sheets.js
```

## What you'll learn

This lab will show you how to do 3 key activities with clasp:

- How to create new Apps Script projects
- How to clone, pull, and push existing projects
- How to manage deployments of your scripts

## Setup and Requirements

**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 Google 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 Google Cloud for the duration of the lab.

**What you need**

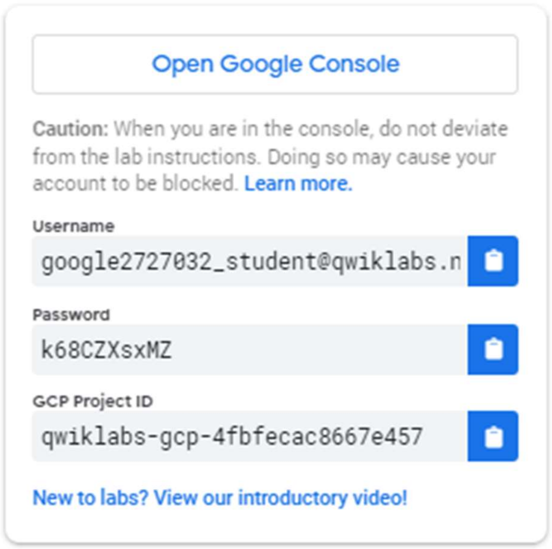
To complete this lab, you need:

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

**Note:** If you are using a Pixelbook, open an Incognito window to run this lab.

**How to start your lab and sign in to the Google Cloud Console**

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Username

```
google2727032_student@qwiklabs.n
```

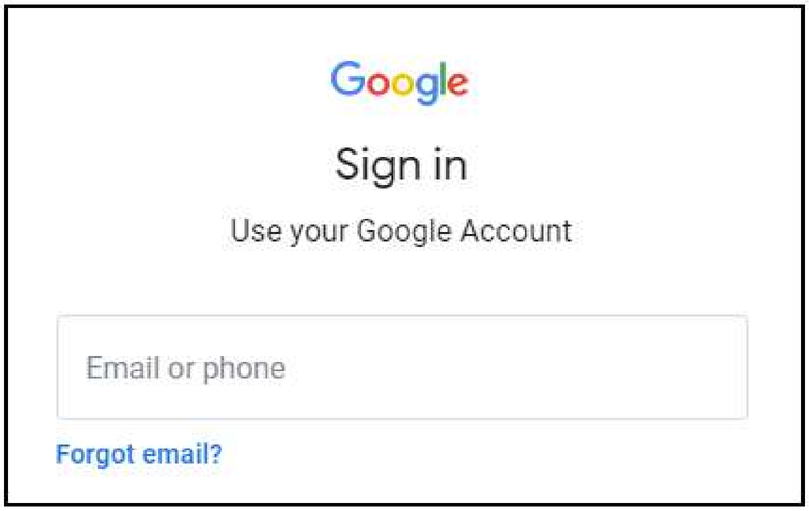
Password

```
k68CZXsxMZ
```

GCP Project ID

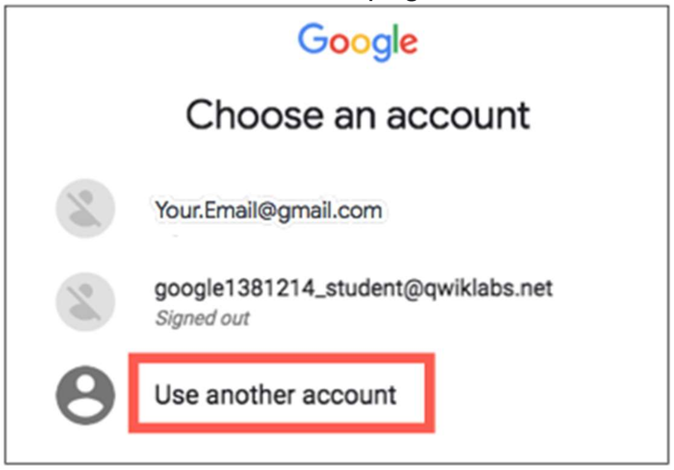
```
qwiklabs-gcp-4fbfecac8667e457
```

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



***Tip:*** Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another**



**Account**.

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

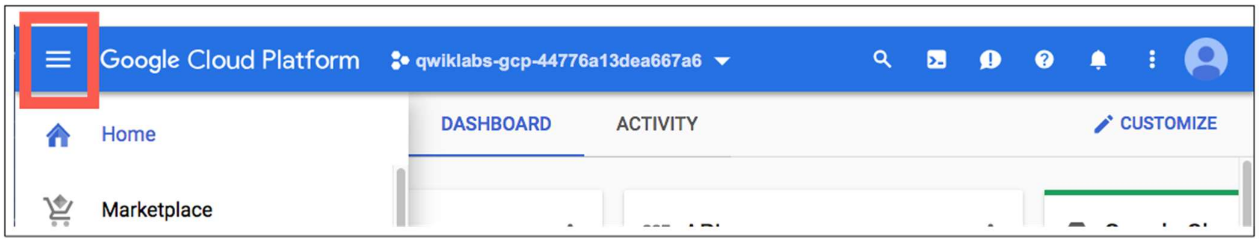
   ***Important:*** You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

   - Accept the terms and conditions.
   - Do not add recovery options or two-factor authentication (because this is a temporary account).
   - Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

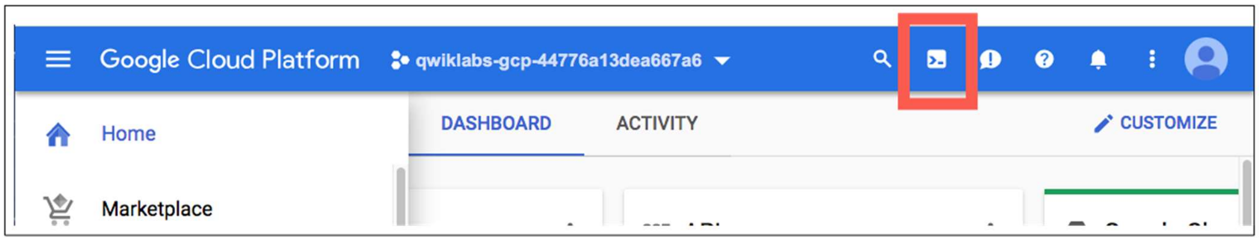
**Note:** You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-
left.



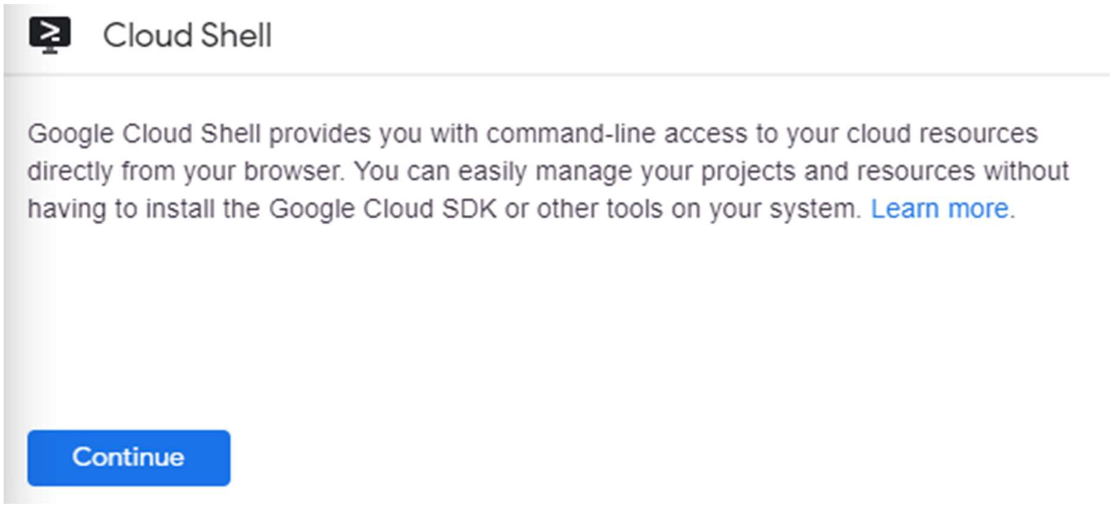
## Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.

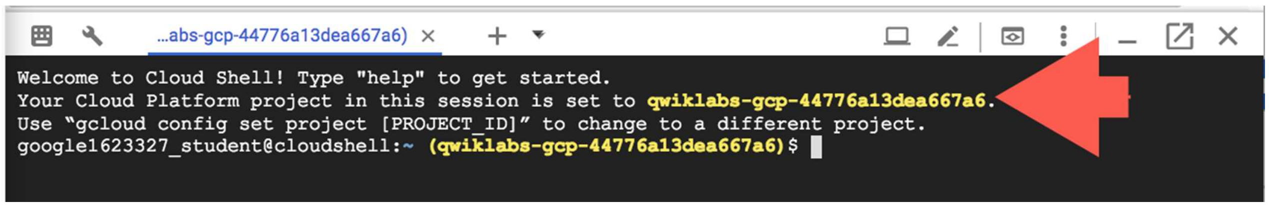


Click **Continue**.


Cloud Shell

Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. Learn more.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6)$
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:
 - <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]
project = <project_ID>
```

(Example output)

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the gcloud command-line tool overview.

## Download the CLI

Google Cloud comes with Node.js. The Apps Script CLI (clasp) requires Node.js >= v4.7.4.

Verify your Node.js version:

```
node --version
```

You should see a v8.9.4 or more.

Install the CLI globally (alias clasp):

```
npm i @google/clasp -g
```

## Login

Try out clasp! The only command you need to remember is `clasp`.

```
clasp
```

You should see detailed information about the `clasp` command.

Before you start using the command line tool, run this command to log in:

```
clasp login --no-localhost
```

The output directs you to authorize clasp by visiting a url. Copy the URL and paste it into a new browser window.

Click on your user name for this lab.

In the **Sign in with Google** page click **Allow**.

Copy the code provided in the **Sign in** page.

Return to Cloud Shell and paste it in the command line at the prompt.

## Create a new standalone project

Start out by creating a stand alone Google Apps Script project.

Run the following to make and then navigate to the `clasp_lab` directory:

```
mkdir clasp_lab && cd clasp_lab
```

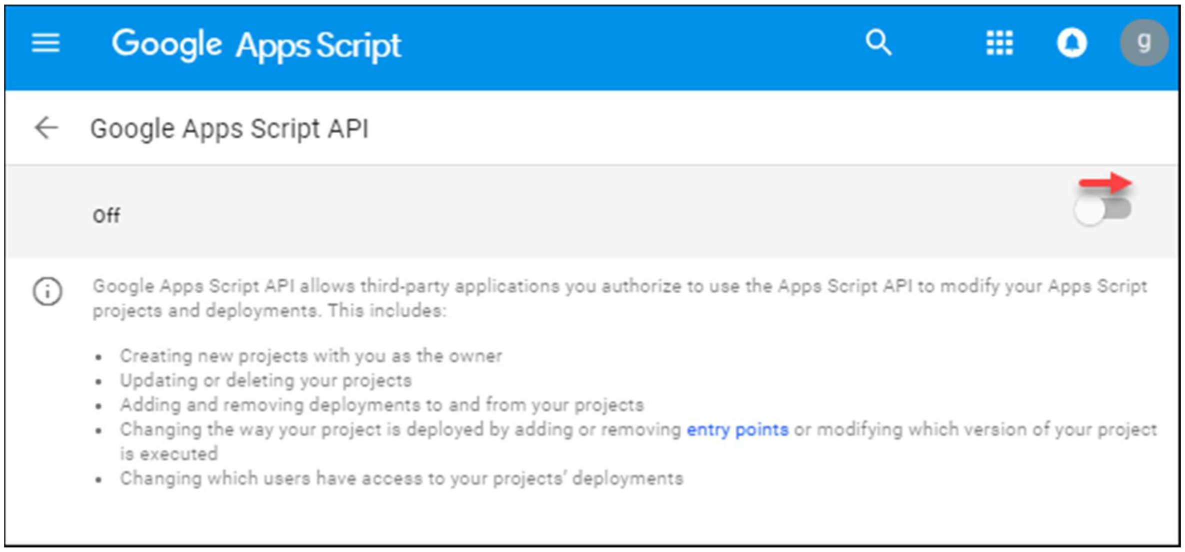
Create the project:

```
clasp create "clasp Lab"
```

Since you're creating a stand alone project, use the arrow keys to select `standalone` then press **Enter**.

You'll be notified that the Apps Script API has not been enabled. Click on the URL provided to enable it.

Click **VIEW DASHBOARD**, then click **Google Apps Script API**. Move the button right to enable Apps Script API.



Google Apps Script API allows third-party applications you authorize to use the Apps Script API to modify your Apps Script projects and deployments. This includes:

- Creating new projects with you as the owner
- Updating or deleting your projects
- Adding and removing deployments to and from your projects
- Changing the way your project is deployed by adding or removing entry points or modifying which version of your project is executed
- Changing which users have access to your projects' deployments

Back in Cloud Shell, re-run the command to create the project:

```
clasp create "clasp Lab"
```

Example output:

```
Created new standalone script:
https://script.google.com/d/1LL5L_Lj1FIvIZ5HMK65JE7ATQXJexVzqbYtLaZK0vDiTURemUjfXIO9x/e
dit
Cloned 1 file.
└─ appsscript.json
```

You just created an Apps Script Project in the folder **clasp_lab**!

Note: The `clasp create` command created a standalone script. This means the project is not connected to a Google Sheet, Doc, Form, or Presentation. How to create a bound script is covered in the next section.
Remove the project you just created before moving to the next section:

```
rm .clasp.json
```

## Clone an existing project

Now create a container-bound script for a Google Slides Add-on.

Go to slides.google.com and create a new presentation.
Name the presentaiton "clasp Code Test".

In the header, under **Tools**, select **Script Editor**.

This opens your Apps Script project at `script.google.com`. To clone a project, you need the Script ID. Find this value by going to **File** > **Project properties**.

You will be prompted for a project name. For this lab type "clasp Code Test", then click **OK**.

In the window that pops up, in the **Info** tab, look for the **Script ID** (this value is also found in the URL). Copy the value and then click **Cancel**.

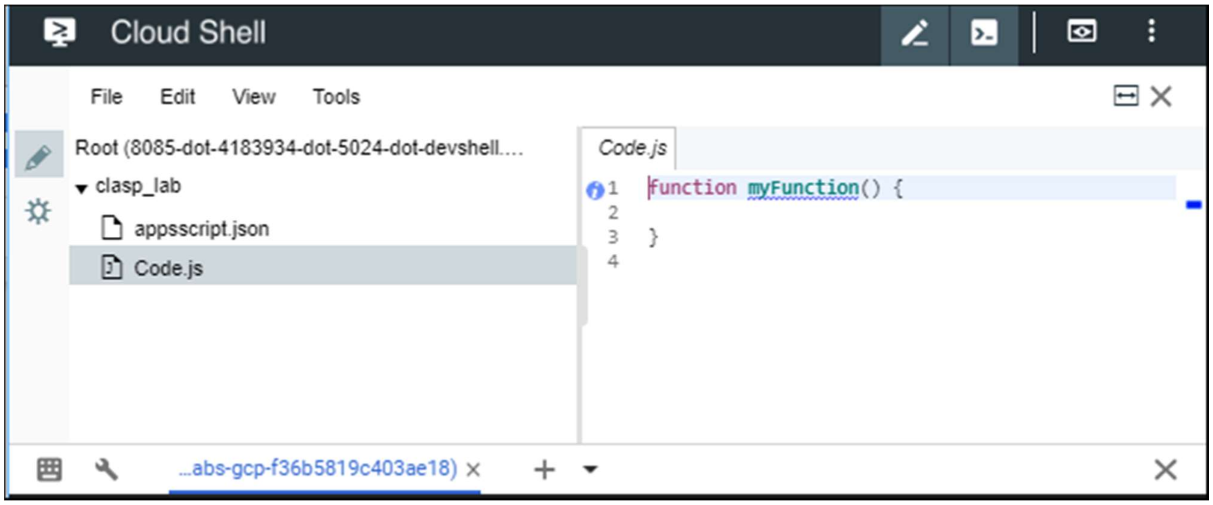
Back in Cloud Shell, clone the project, replacing with the value you copied:

```
clasp clone <scriptID>
```

Open the Code Editor in Cloud Shell by clicking on the pencil icon.

You should see the cloned files `Code.js` and `appscript.json`.

`Code.js` viewed in Cloud Shell code editor:


Cloud Shell
File
Edit
View
Tools
Root (8085-dot-4183934-dot-5024-dot-devshell....
clasp_lab
appsscript.json
Code.js
Code.js
...abs-gcp-f36b5819c403ae18)

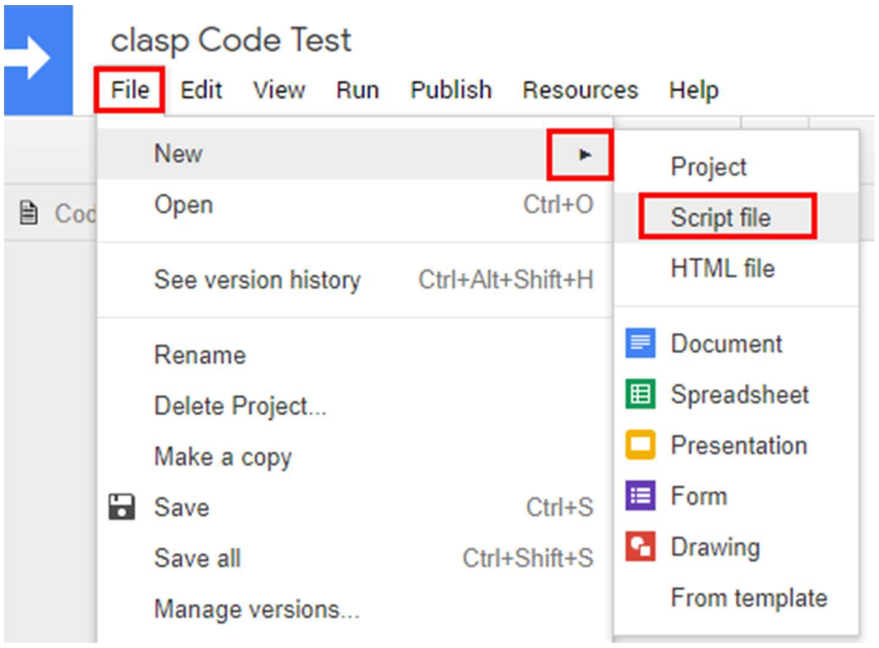
```
function myFunction() {

}
```

You have downloaded the project in your current directory. Use your favorite editor to view the contents of `Code.gs` (an empty function).

# Pulling and pushing files - edit code on script.google.com

Next, learn how to pull and push files. You will edit on the cloud via script.google.com and pull locally to your computer using `clasp`.

In the script editor, create a new file by choosing **File** > **New** > **Script file**.



Enter the name "utils/strings".

In the newly created file, `utils/strings.js`, replace the code with the following:

```
var world = "世界";
```

Click on the `Code.gs`file, replace the existing code with the following:

```
function hello() {
  Logger.log("Hello, " + world);
}
```

Be sure to save all the files.

Try running the function by clicking **Run** > **Run function** > **hello**.

Go to **View** > **Logs** to see the greeting.

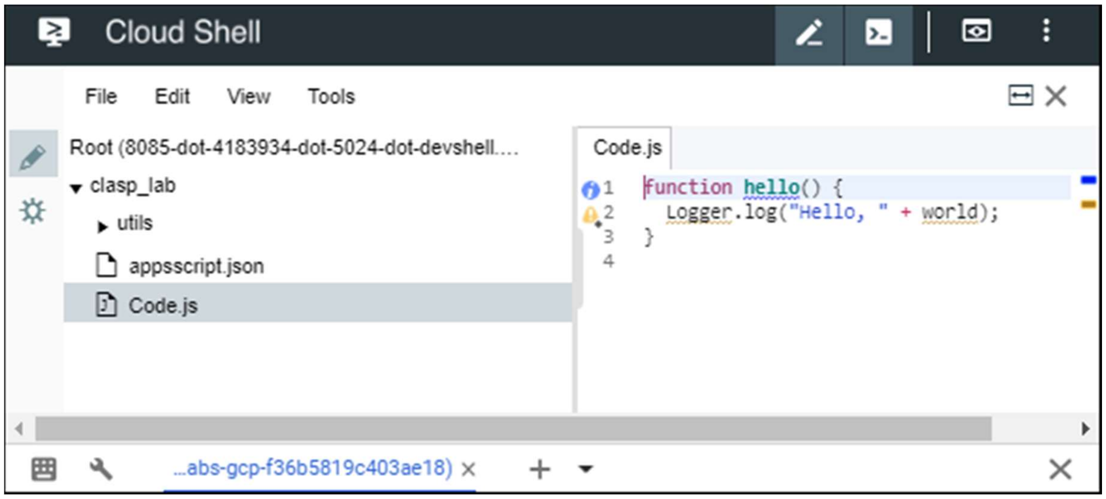
# Edit code locally

Go back to the Cloud Shell command line where you last cloned the project. You may notice that your code is now out of sync with the online editor. To fix that, pull the code from your online project:

```
clasp pull
```

Now go back to the code. You should notice there's a `utils` folder now. The clasp CLI automatically converts the slash character `/` to folders on the local filesystem.

Pulled files in Cloud Shell code editor:


Cloud Shell
File
Edit
View
Tools
Root (8085-dot-4183934-dot-5024-dot-devshell....
clasp_lab
utils
appsscript.json
Code.js
...abs-gcp-f36b5819c403ae18)

Code.js

```
function hello() {
  Logger.log("Hello, " + world);
}
```

In your favorite text editor:

- Navigate to `util/strings.js` and replace the variable name `world` to `mundus`.
- Update the `Code.js` by replacing "world" with "mundus".
- Push your edited code to update the code on script.google.com:

```
clasp push
```

And that's it! Your code is now updated on script.google.com.

`clasp push` replaces code that is on script.google.com and `clasp pull` replaces all files locally. For this reason, follow these guidelines:

- Do not concurrently edit code locally and on script.google.com.
- Use a version control system, like git.

# Versioning and Deploying

`clasp` allows you to manage versions and deployments. First, some vocabulary:

- **Version**: A "snapshot" of a script project. A version can be considered a read-only branch used for deployments.
- **Deployment**: A published release of a script project (often as an add-on or web app). Requires a version number.
  In Cloud Shell, create a version of your script:

```
clasp version "First version"
```

Using the logged version string you created in place of [version], deploy the script:

```
clasp deploy 1 "First deployment"
```

The `clasp deploy` command looks at your manifest and creates a new versioned deployment.
Your code is now deployed as an executable. Learn more about this in the deployments guide.
You can now choose what version of your project to deploy.

## Congratulations!

The Apps Script CLI is a simple tool to help you manage Apps Script projects. In this lab you performed the following:

- Created a new Apps Script project
- Pushed and pulled existing projects
- Managed deployments of your scripts
  clasp is available on GitHub and welcomes new features and patches.

### Finish Your Quest

This self-paced lab is part of the Workspace Integrations Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in this Quest and get immediate completion credit if you've taken this lab. See other available Qwiklabs Quests.

### Take your next lab

Continue your Quest with these labs:

- Google Apps Script: Access Google Sheets, Maps & Gmail in 4 Lines of Code
- Google Sheets as a Reporting Tool - Sheets API

### Learn more

- Read the Google Sheets API developer documentation.
- Post questions and find answers on Stackoverflow under the google-sheets-api tag.
- Google Workspace Learning Center

### Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.
Manual Last Updated June 25, 2019
Lab Last Tested June 25, 2019