Managing and Executing gcloud Commands in JupyterLab

Christoph Bussler Just now · 6 min read

tl;dr For the longest time I managed gcloud commands in documents or text files, especially during prototyping or initial research. From there I copy and paste them one-by-one or in blocks to a terminal or to the Cloud Shell for execution. This is tedious, error prone and very inefficient — so I changed my approach: in this blog I briefly show you how you can combine both in JupyterLab. You can manage as well as execute gcloud commands in a notebook. I list a few interesting use cases as well in addition to some installation caveats that I came across.

Example: Cloud Spanner — from zero to query in 5 commands

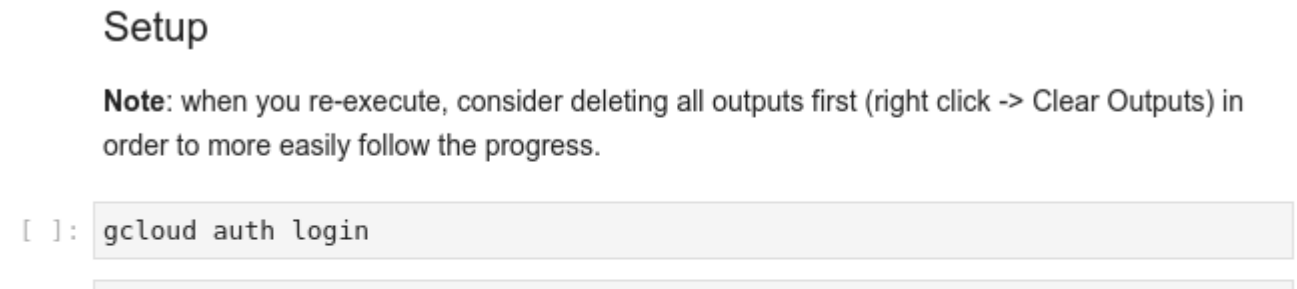
In the following you will see an example of using gcloud commands in a JupyterLab bash notebook. This requires to install the bash kernel in JupyterLab (see installation section below).

There are three sections in my notebook:

- **Setup**. This section has the necessary commands to setup the notebook
- **Demo**. This section shows five gcloud commands to setup a Cloud Spanner instance, setup a database, create tables and run a query.
- **Teardown**. This section removes all resources that were created in the notebook.

Section: Setup

The following screenshot shows the setup section:



```
[ ]: gcloud config set project jupyter-blog-007

[ ]: gcloud config set disable_prompts true

[ ]: gcloud services enable spanner.googleapis.com
```

Setup section in notebook

The commands execute the following (in order top to bottom):

1. Authenticate and authorize the user
2. Set the project for which the commands are executed
3. Disable any gcloud command prompts and use the default
4. Enable the Cloud Spanner APIs

At this point all necessary setup is completed.

Section: Demo

The next screenshot shows the demo section of the notebook:

Demo

```
[5]: export instance_name=blog-instance
     export database_name=blog-database
     export instance_region=regional-us-west1

[6]: gcloud spanner instances create $instance_name \
       --description=$instance_name \
       --config=$instance_region \
       --nodes=1

     Creating instance...done.

[7]: gcloud spanner databases create $database_name \
       --instance=$instance_name

     Creating database...done.

[8]: gcloud spanner databases ddl update $database_name \
       --instance=$instance_name \
       --ddl='CREATE TABLE blog_entity (k INT64, v STRING(1024)) PRIMARY KEY(k)'

     Schema updating...done.

[9]: gcloud spanner databases execute-sql $database_name \
       --instance=$instance_name \
       --sql="INSERT blog_entity (k, v) VALUES (1, 'first entity')"
```

```
Statement modified 1 row

[10]: gcloud spanner databases execute-sql $database_name \
        --instance=$instance_name \
        --sql='SELECT * from blog_entity'

k  v
1  first entity
```

Demo section in notebook

This demo section is self-explanatory. A few shell variables are set and used in the subsequent commands. The commands go through a simple Cloud Spanner use case. As you can see, the output of every command is displayed in the notebook automatically after it is run.

Section: Teardown

The next screenshot shows the teardown section of the notebook:

Teardown

```
[11]: gcloud spanner instances delete $instance_name
```

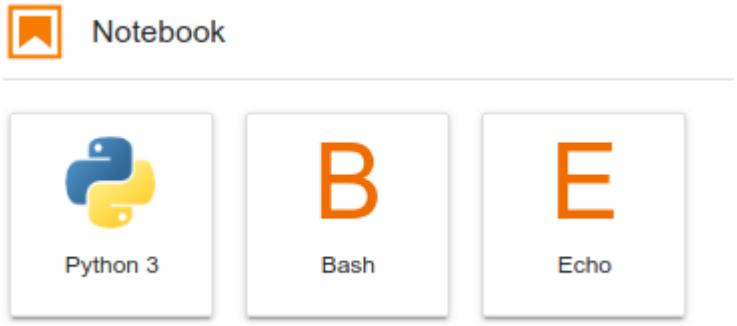
Teardown section in notebook

The one command removes the Cloud Spanner instance and all its databases.

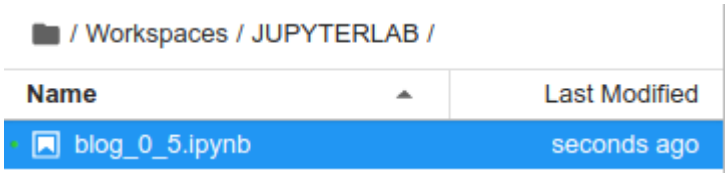
Workflow and best practices

The overall workflow is as follows:

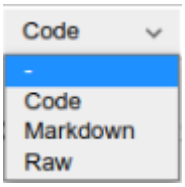
1. **Create a bash notebook**. This is done by clicking the **Bash** icon in JupyterLab.



2. **Name the notebook**. Provide a suitable name for the notebook.



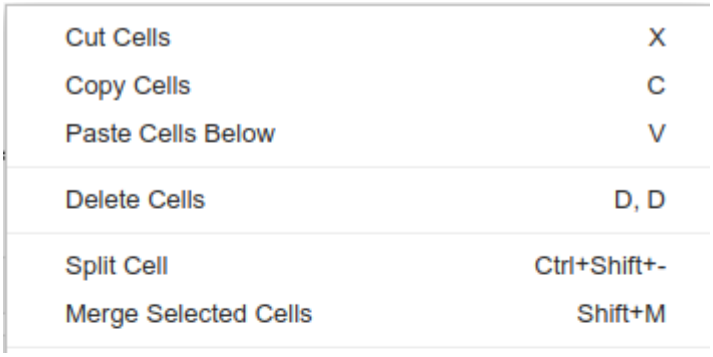
3. **Add a first command**. Click "+" to create a notebook cell and write the command into it. In the drop down on the right, you can select whether the command is raw text, a text following markdown notation or a code command like gcloud.

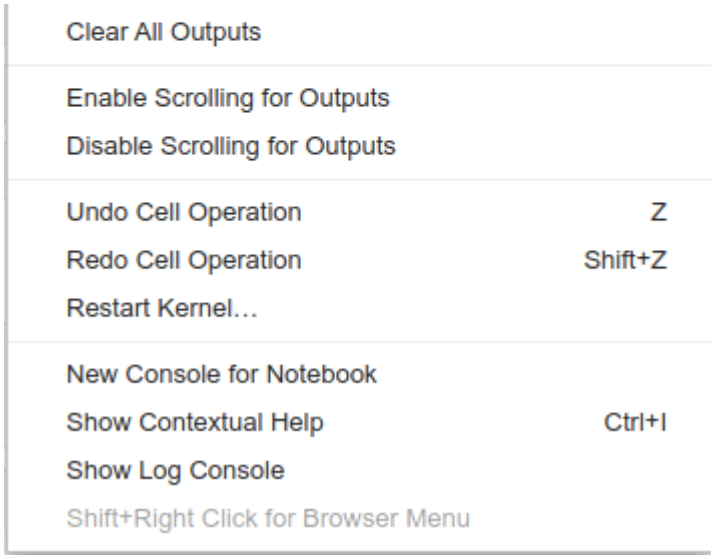


4. **Execute the command**. Place the cursor into the cell and click on the right arrow to execute the command.

5. **Keep adding commands**. Depending on your needs create additional raw, markdown or code commands by adding notebook cells.

6. **Execute from the beginning**. It is possible to remove all output by selecting "Delete all output" and this will leave only the commands in the notebook. You can re-execute all commands and the new output will be dynamically added to the notebook.





From a best practice perspective consider the following recommendations

- Start a notebook with a dedicated setup section
- Parameterize the commands by setting shell variables
- Document the commands as needed
- Provide a cleanup or teardown section that cleans up all created resources

These basic best practices are a good start and you can refine those over time to your needs.

Use cases

There are quite a number of use cases for this approach of managing and executing gcloud commands with JupyterLab in notebooks. Some important use cases are:

- **Customer Support**. Using JupyterLab customers can communicate executed gcloud commands in notebooks and describe the encountered problem without having to share their Google Cloud project. You can rerun those commands in your project for investigation.
- **Iterative development**. When you experiment, create POCs or do initial research on cloud resources, notebooks give you the ability to easily modify gcloud commands, keep track of the gcloud commands that succeeded (and failed) as well as their output.
- **Demonstrations**. A notebook can serve as a demo script that provides you the ability to not only list and execute the commands, but also document noteworthy details.

- **Documentation**. You can create one or several notebooks for you to document gcloud commands for yourself or others with details beyond the official product documentation.
- **Collaboration**. When checked into source control, notebooks can serve as the basis for collaboration within and across teams.
- **Teaching**. A notebook can be used to teach one or more lessons about cloud resources and the gcloud commands that go with them.
- **Learning**. When following instructions, you could collect the gcloud commands in a notebook to keep a trail of what you have executed and for possible repeated executions.

These are some examples of how gcloud commands may be used in notebooks. You will definitely find more important use cases over time.

Installation

These are my installation notes and some caveats I found during installing JupyterLabs on my Linux laptop. Details might vary in your situation; the following is only an experience report for my situation.

Installing JupyterLab. The best experience for me was to use `pip` for installation as described here. You might have to install `pip` first.

Installing echo and bash kernels. Kernels provide the functionality available in notebooks and I installed two: the echo kernel for testing, and afterwards the bashkernel. The bash kernel is the one used to run gcloud commands (above example was implemented using the bash kernel).

Starting JupyterLab. Run the command jupyter lab on a console and JupyterLab starts.

There are come caveats based on my experience:

- **Bracketed paste**. At one point unexpected character sequences appeared in some environment variables as shown here. This was very frustrating as I was not sure where it was coming from. After some search, I found that this comes from the concept of

bracketed paste. Once I turned bracketed paste off, I did not encounter this issue anymore. My approach was to add this line to the `.bashrc` file:

```
bind 'set enable-bracketed-paste off'
```

- **Prompts**. If a command asks for input during its execution, then this prompt will not be propagated to the notebook. From the viewpoint of the notebook, the command appears to be hanging, but it is actually waiting for user input. This is the reason to switch off the prompt in the setup section.

Your imagination

Experiment with managing and executing gcloud commands in JupyterLab notebooks, and see what other use cases, functionality, and features you might come up with.