

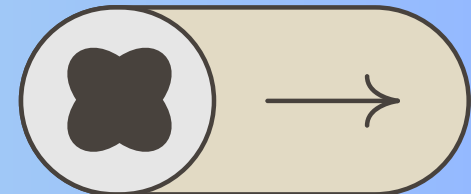
# Execute and Run Bash Scripts in Databricks



**BASH**  
THE BOURNE-AGAIN SHELL

Ganesh R

Azure Data Engineer





You are working in Databricks and need to perform tasks outside of the usual Spark or Python workflows. Perhaps you need to install a certain command-line tool, third party libraries, change some system settings, or just run a script that's quicker in Bash. That's where executing Bash scripts in Databricks becomes useful. But first, what is Bash? Bash—short for Bourne Again Shell—is a command processor that normally runs in a text window and allows users to interact with the core operating system by entering commands or scripts.

So, why would you want to execute these scripts in Databricks? There are several valid reasons:

- 1) Automate your setup. Setting up your Databricks environment can be repetitive. Bash scripts can automate this process by installing specific system packages or configuring settings every time a cluster starts, eliminating the need for manual setup each session.
- 2) Install anything you want. Sometimes, the Python or R libraries you need aren't available through standard package managers like pip or CRAN. Bash allows you to use system-level package managers to install virtually anything you need.
- 3) Get down to the core. Bash commands give low-level access to the cluster's file system and system processes, which Spark and Python environments normally abstract away.
- 4) Leverage command-line magic. There's a vast ecosystem of command-line tools for data processing. Tools for format conversion, data validation, or even calling external APIs might be easier to handle with existing Bash utilities than rewriting them in Spark.



5) Connect to external services. Bash simplifies the process of using command-line tools to communicate with external services or APIs directly from your Databricks cluster.



you'll learn how to run Bash scripts in Databricks four ways: directly in notebooks (using %sh magic command), from stored scripts in Databricks DBFS or cloud storage, through the Databricks Web Terminal, and via automated cluster initialization scripts.

Save up to 50% on your Databricks spend in a few minutes!

### Step-By-Step Guide to Run Bash in Databricks

Let's explore the specifics of running Bash in Databricks. Each method serves a particular use case, so select the one that best suits your needs. We'll cover four different techniques:



Technique 1—Run Bash Scripts Inline in Databricks Notebook

Technique 2—Run Stored Bash File in Databricks

Technique 3—Run Bash Scripts via Databricks Web Terminal


Technique 4—Run Bash Scripts via Cluster Global Init Scripts

Let's dive right into it!

Prerequisites:

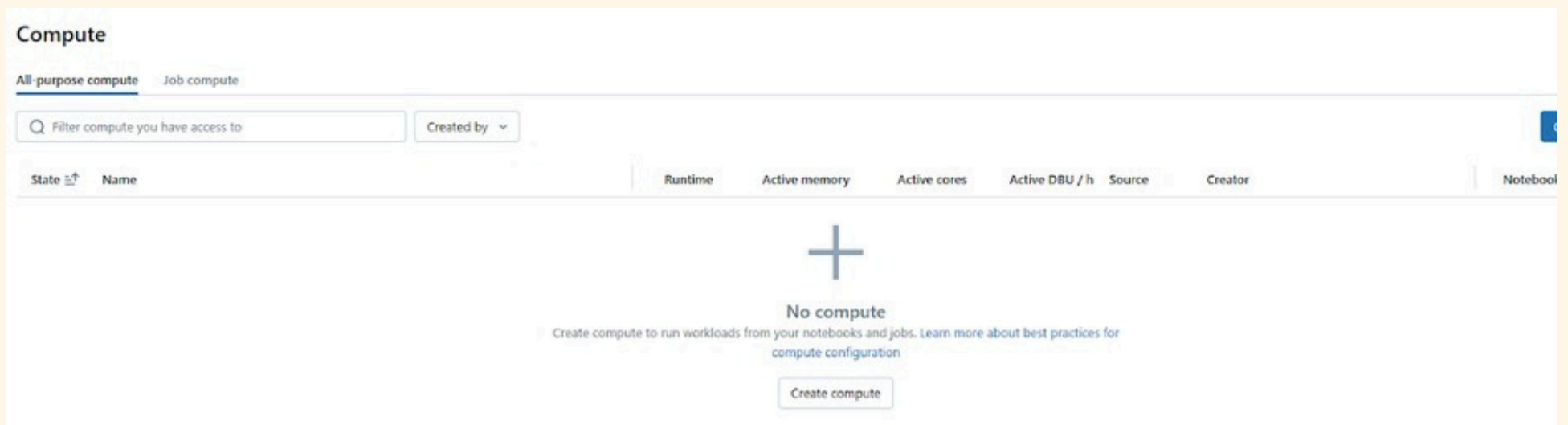
- Before diving into the techniques, make sure you have these basics down: Access to a Databricks workspace and an all-purpose or job cluster.
- Proficiency in basic Bash scripting.
- Knowledge of Databricks DBFS.
- Permissions to execute shell commands, access Databricks Web Terminal, manage Global Init Scripts.
- Databricks Web Terminal must be enabled in workspace settings.



 **Technique 1—Run Bash Scripts Inline in Databricks Notebook using %sh Magic Command** One of the simplest ways to run Bash commands in Databricks is to write them directly in a notebook cell using the Databricks %sh magic command. This way, you can run shell scripts without leaving the notebook. Here is how it works: Whenever you type in %sh at the beginning of a Databricks Notebook cell, Databricks interprets the cell's contents as a shell script. The commands run on the driver node, which means that they only affect the environment on that node.

**Step 1—Login to Databricks** First thing first, start by logging into your Databricks workspace.

**Step 2—Configure Databricks Compute** Once you're in, you need an active Databricks compute cluster. To set this up, go to the "Compute" section on the left sidebar. If you don't have a Databricks compute cluster or need a new one, click on "Create Compute".



Setting up Databricks compute clusters - Run Bash in Databricks


Then, customize the Databricks compute cluster settings according to your needs.



Compute > New compute

## Demo Cluster

Databricks runtime version ⓘ

Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2) 

Instance

Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For more configuration options [🔗](#), please upgrade your Databricks subscription. [🔗](#)

Spark

Spark config ⓘ

```
spark.databricks.rocksDB.fileManager.useCommitService false
```

Environment variables ⓘ

```
PYSPARK_PYTHON=/databricks/python3/bin/python3
```

Create compute Cancel

Setting up Databricks compute clusters - Run Bash in Databricks If the Databricks compute cluster isn't already running, start it.

Step 3—Open Databricks Notebook Launch a new or existing Databricks Notebook and make sure it's attached to your active Databricks compute cluster.

Step 4—Execute the Bash Script using Databricks %sh Magic Command



Databricks magic commands are special commands that provide extra functionality within notebook cells. They're like shortcuts to do things beyond the standard notebook language. Magic commands are always prefixed with a percentage sign %. The magic command we're interested in here is Databricks %sh. What Databricks %sh Does? Databricks Shell %sh magic command lets you execute shell commands directly within a notebook cell. "sh" stands for "shell", and in the context of Databricks compute clusters, it typically refers to Bash (or a Bash-compatible shell) running on the driver node of your Databricks compute cluster. Always remember that Databricks %sh commands are executed on the driver node of your Databricks compute cluster. The driver node is the main node that coordinates the Spark jobs across the Databricks compute cluster. This is where the notebook runtime environment resides. So, when you run Databricks %sh ls, for instance, you're listing files on the driver node's file system, not necessarily the distributed file system accessible to all executors. Let's see it in action. Open a cell in your Databricks Notebook and try these examples:

Run this cell ( Shift+Enter ) Cmd/Ctrl+Enter

```
%sh
pwd

/databricks/driver
```

The screenshot shows a Databricks notebook cell interface. At the top, there is a blue button with a play icon and a dropdown arrow, followed by a green checkmark and the text "Just now (1s)". Below this, the cell content is displayed in a light blue background. It starts with the magic command `%sh` in green, followed by the command `pwd` in black. The output of the command is shown in a white box at the bottom, displaying the path `/databricks/driver` in a monospaced font.

Running bash command in Databricks Notebook - Run Bash Script - Run Bash in Databricks

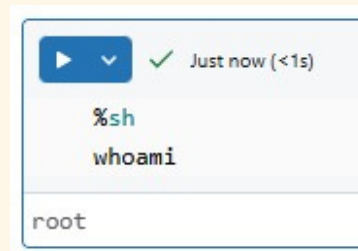


## Running bash command in Databricks Notebook - Run Bash Script - Run Bash in Databricks

You'll see the output, which is the list out the working directory on the driver node.

Try another:

`%sh whoami`



## Running bash command in Databricks Notebook - Run Bash Script - Run Bash in Databricks

This will show you the user account under which the shell commands are being executed. Again, this is the user on the driver node. Let's run a slightly more complex command:

`%sh mkdir test_dir`

As you can see, this command creates a directory named `test_dir` in the current working directory.

`%sh`

`ls -l`





```
▶ Just now (<1s)
%sh
ls -l

total 1304
drwxr-xr-x 2 root root 4096 Feb 12 02:28 azure
drwxr-xr-x 2 root root 4096 Feb 12 02:28 conf
drwxr-xr-x 3 root root 4096 Feb 12 02:30 eventlogs
-r-xr-xr-x 1 root root 2755 Feb 12 02:28 hadoop_accessed_config.lst
drwxr-xr-x 2 root root 4096 Feb 12 02:31 logs
-r-xr-xr-x 1 root root 1306936 Feb 12 02:28 preload class.lst
drwxr-xr-x 2 root root 4096 Feb 12 02:44 test_dir
```

Listing out all the directories using bash in Databricks - Run Bash in Databricks

You can see that this command lists the contents of the current directory in a long listing format. You should see test\_dir

To run a simple inline Bash script, you can put multiple commands within a single

%sh cell:

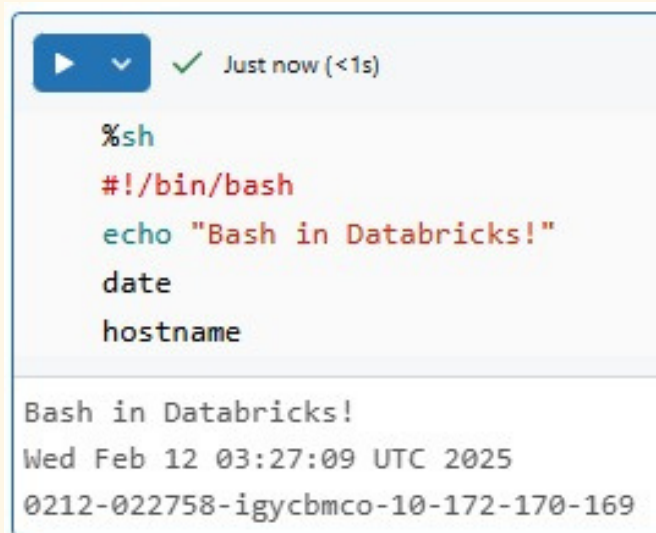
%sh

#!/bin/bash

echo "Hello from Bash in Databricks!" date hostname



Run this cell, and you'll see the output from all three commands.



```
%sh
#!/bin/bash
echo "Bash in Databricks!"
date
hostname
```

Bash in Databricks!  
Wed Feb 12 03:27:09 UTC 2025  
0212-022758-igycbmco-10-172-170-169

Running multiple bash scripts in Databricks - Run Bash Script - Run Bash in Databricks

What Databricks %sh Can't Do (Limitations):

Databricks %sh is useful, but it does not solve every Bash problem with Databricks. There are several limits to keep in mind.



Databricks %sh operates only on the driver node. It doesn't directly reach out to the Spark executors or the distributed computing environment. If you need to do something across your entire Databricks compute cluster, %sh isn't the tool for

Databricks %sh is best for scripts that run from start to finish without needing user input in the middle. While you can run commands that expect input, managing interactive prompts or complex input/output within Databricks %sh gets tricky fast. For fully interactive shell work, Databricks Web Terminal (which we will discuss later) is a preferable option.

Running shell commands, especially arbitrary ones, raises security concerns. In shared Databricks environments or when handling sensitive data, be careful with the commands you run.

➡ Despite these points, Databricks %sh remains an extremely useful tool for many everyday tasks in Databricks Notebooks that require quick access to Bash commands.

### 🧠 Technique 2—Run Stored Bash Scripts from Databricks DBFS or Mounted Storage

For more complex or reusable scripts, it's usually better to store them in separate files and run them from Databricks. You've got a couple of options for storing these scripts—you can use Databricks File System (DBFS) or put them in cloud storage that's mounted to your Databricks workspace.

Let's break down how to do this.

Steps 1, 2, and 3—Setup Your Databricks Environment – Same as Before



These steps are the same as in Technique 1. You need to:

**Step 1—Login to Databricks**

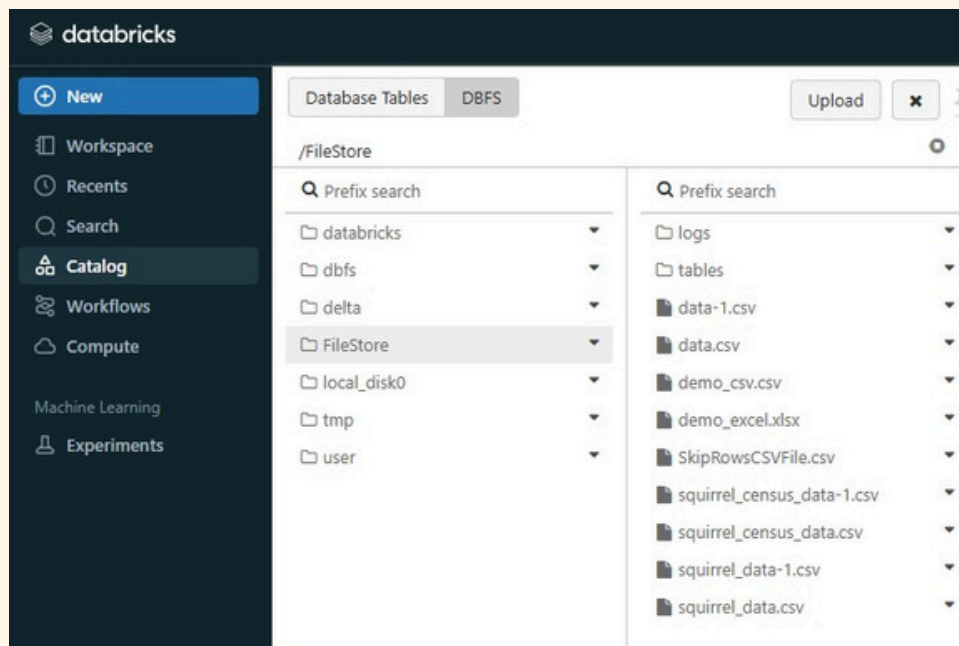
**Step 2—Configure Databricks Compute**

**Step 3—Open Databricks Notebook**

Basically, get your Databricks Notebook environment ready, just like you would for running inline Databricks <sup>bash</sup> commands. Once you've got that setup, the process diverges as we start dealing with external script files.

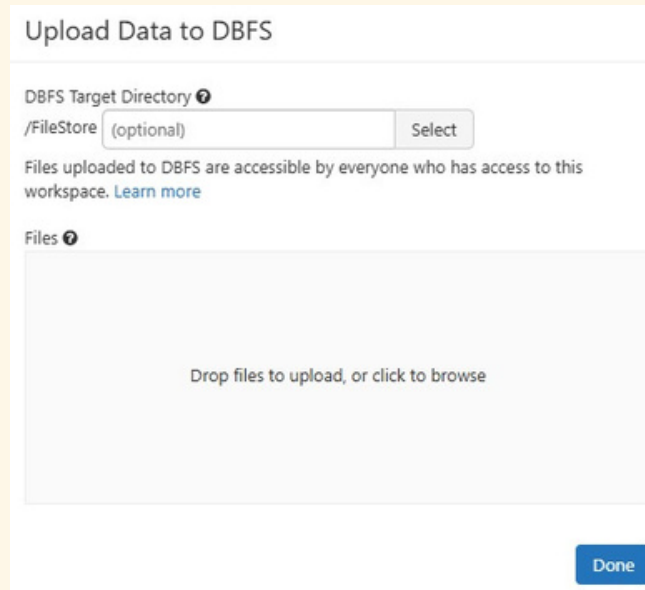
**Step 4—Uploading the Bash Script File to Databricks DBFS (or Mounted Storage)**

To use a stored Bash script, first upload the file to Databricks DBFS. You can do this through the Databricks UI. Navigate to the Catalog section located on the left side, select DBFS, and then click Databricks FileStore.



Navigating to Databricks DBFS - Run Bash in Databricks - Run Bash File

Once you are in the FileStore directory, click the "Upload" button. A file upload window will pop up. You can either drag your .sh script file from your computer directly into this window or click "Browse" to find and select the file using your computer's file explorer.



Uploading bash script to Databricks DBFS - Run Bash in Databricks - Run Bash File

After you've selected your file, hit the "Upload" button in the dialog. After a moment, your script file should appear in the Databricks DBFS directory you chose. You've now successfully uploaded your Bash script to Databricks DBFS using the UI.

For the purpose of this demonstration, this is what we have added to our Bash script file. Make sure to add your own script.



Uploading bash script to Databricks DBFS - Run Bash Script - Run Bash in Databricks For a more detailed visual step-by-step guide, see this article on how to upload a file to DBFS through the UI.

## Step 5—Configure File Permissions

After you've uploaded your Bash script file to Databricks DBFS (or a mount point), there's a step you might need to consider: setting file permissions. Specifically, making the script executable. But first, make sure to copy the path of the Bash script that you have uploaded to Databricks DBFS.



Select a path to copy ⓘ

Spark API Format

File API Format

Selecting the Bash script file path - Run Bash in Databricks - Run Bash File

Then, open a new Databricks Notebook cell and run the following `%sh:` command with Databricks

```
%sh
chmod +x
/dbfs/FileStore/demo_bash_in_databricks.sh
```

As you can see, the `chmod +x <filepath>` command is used to grant execute permissions. `chmod` is short for "change mode ", and `+x` means "add execute permission".

Do You Always Need `chmod +x`?

Interestingly, no, not always. If you explicitly tell Bash to run your script using the `bash` command, you might not strictly need the execute permission set on the script file itself. For example:

```
%sh

bash /dbfs/FileStore/demo_bash_in_databricks.sh
```

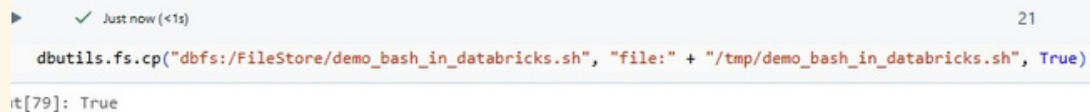


In this case, you're directly calling the bash program and telling it to interpret and execute the script file you're providing as an argument. bash itself is the executable, and it just needs to be able to read your script file. So, in this scenario, execute permissions on the script file becomes less critical.

If you are unable to directly execute a file located in DBFS, here is an alternative way to do it. First, copy the script from Databricks DBFS to the local filesystem of the driver node. You need to bring the script from Databricks DBFS to a

location where the driver node can execute it. A common place for temporary files is the /tmp directory. Here is how to do so:

```
dbutils.fs.cp("dbfs:/FileStore/demo_bash_in_databricks.sh", "file:" + "/tmp/demo_bash_in_databricks.sh", True)
```



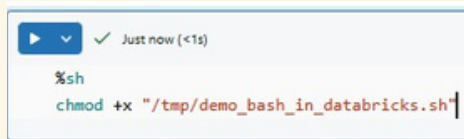
The screenshot shows a Databricks terminal window with a blue header bar containing a play button, a checkmark, and the text "Just now (<1s)". The command `dbutils.fs.cp("dbfs:/FileStore/demo_bash_in_databricks.sh", "file:" + "/tmp/demo_bash_in_databricks.sh", True)` is entered and executed. The output shows the prompt `it[79]:` followed by `True`. A line number "21" is visible in the top right corner.

Copying Bash script to the temporary folder - Run Bash Script - Run Bash in Databricks

After copying, you'll need to make sure and check that the script has execute permissions:

```
%sh
```

```
chmod +x "/tmp/demo_bash_in_databricks.sh"
```



The screenshot shows a Databricks terminal window with a blue header bar containing a play button, a checkmark, and the text "Just now (<1s)". The command `%sh` is entered, followed by `chmod +x "/tmp/demo_bash_in_databricks.sh"` on the next line. The cursor is at the end of the second line.

Making the bash script executable - Run Bash Script - Run Bash in Databricks - Run Bash File

Now, let's move on to the final step on how you can execute this Bash script.



## Step 6—Execute the Script within Databricks Notebook

With your script uploaded (and maybe execute permissions set), you're finally ready to run it from a Databricks Notebook. You'll use the trusty Databricks %sh magic command again, but this time, you'll point it to the path of your stored script file.

Assuming your script is at

/dbfs/FileStore/demo\_bash\_in\_databricks.sh

%sh

or /tmp/demo\_bash\_in\_databricks.sh and you've (optionally) done chmod +x, you can execute it simply by:

sh "/tmp/demo\_bash\_in\_databricks.sh"

```
✓ Just now (<1s) 25
%sh
sh "/tmp/demo_bash_in_databricks.sh"

n Bash Script!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!!
ot
```

Executing the bash script within Databricks Notebook - Run Bash in Databricks

Or by using Databricks bash command:

%sh

bash "/tmp/demo\_bash\_in\_databricks.sh"

```
✓ Just now (<1s) 25
%sh
bash "/tmp/demo_bash_in_databricks.sh"

n Bash Script!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!!
ot
```





./

Executing the bash script within Databricks Notebook - Run Bash in Databricks

If you prefer to use the style for indicating "run this executable in the current directory", you can first use `cd` within your Databricks `%sh` cell to change the working directory:

```
%sh
cd /tmp/
sh ./demo_bash_in_databricks.sh
```

```
Just now (<1s) 26

%sh
cd /tmp/
sh ./demo_bash_in_databricks.sh

Run Bash Script!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!!
root
```

Executing the bash script within Databricks Notebook - Run Bash Script

A Few Things to Note:

- When you're referencing files in Databricks DBFS, you should add `/dbfs/` at the start of your path.
- Running scripts with elevated permissions can be a double-edged sword. Be very very careful of what your script does and who can run it.
- If your script fails, check the Databricks Notebook output for any error messages. It will give you a clue on what went wrong or if there's a typo in your script path.
- That's it. This approach lets you manage your Bash scripts in Databricks in a controlled way, using the same cluster connection as inline commands. Having trouble? Check that the file path is correct and the file permissions are set properly.



- Technique 3—Run Bash Scripts via Databricks Web Terminal
- Now, this technique gives you hands-on access to a Bash shell on a Databricks cluster node. Forget about embedding commands in notebooks—the Databricks Web Terminal is like your regular terminal window, but it's connected directly to your Databricks environment. It's the perfect tool for you when you need to interact with the shell in real-time.

Steps 1, 2, and 3—Setup Your Databricks Environment – Same as Before

Even though you will not be executing scripts from a Databricks Notebook with this technique, it is still useful to consider starting in a notebook for context. The steps are the same as in Technique 1. You have to:

**Step 1—Login to Databricks**

**Step 2—Configure Databricks Compute**

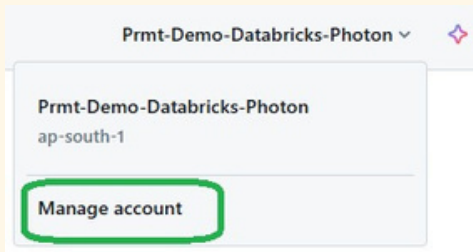
**Step 3—Open Databricks Notebook**

These steps are more about making sure you're in the Databricks environment and ready to work with a cluster. The real action for the Web Terminal happens outside the Databricks Notebook interface itself.



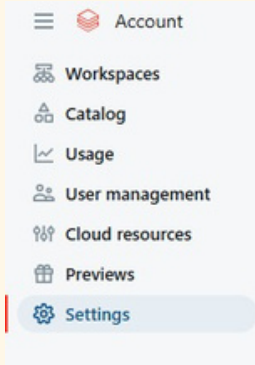
#### Step 4—Enabling Databricks Web Terminal in Workspace Settings

- Here's the first important thing to know: the Databricks Web Terminal isn't automatically enabled for every Databricks workspace. A Workspace Admin needs to enable it. If you're not a Workspace Admin, you'll need to ask someone with those admin rights to turn it on for your workspace.
- For those who are Workspace Admins, here's how to enable the Databricks Web Terminal:
- First, navigate to the top-right of your Databricks workspace. Click on your username. From the dropdown menu, choose "Manage Account". This is where workspace-level settings are managed.



Navigating to Databricks workspace settings - Run Bash in Databricks - Run Bash File

- Once in the workspace-level page, click on the "Settings" tab. It's usually located on the left side of the page.



Navigating to Databricks workspace settings - Run Bash in Databricks - Run Bash File

Click the "Feature Enablement" tab. Keep looking until you find a section labeled " Web Terminal



## Settings

[Subscription & billing](#) [Authentication](#) [User provisioning](#) [App connections](#) **[Feature enablement](#)** [Language settings](#) [My preferences](#) [Security and compliance](#) [Account settings](#)

Some features require you to explicitly opt-in before you can use them. Use this page to enable and configure the following account features.

### Personal Compute

Enables all users to create single-machine compute using the Personal Compute default policy. [Learn more](#)

Enable for all ☒ ?

### Web Terminal

Grants all users access to the web terminal, where they can interact with the command line on their all-purpose compute resources. [Learn more](#)

Enforce ☒ On ▾

### Enable Admin Protection for "No Isolation Shared" Clusters

Enabling admin protection prevents automatic generation of Databricks API credentials for Databricks admins on "No Isolation Shared" clusters. Note that enabling admin protection may prevent some workloads from being run as an admin user on "No Isolation Shared" clusters. [Learn more](#)

Disabled ☐ ?

### Enable partner-powered AI features

Some AI features rely on partner model providers. Model partners do not access or retain your data. [Learn more](#)

Enforce ☒ On ▾

### Predictive Optimization

Databricks intelligently optimizes your Unity Catalog managed tables for performance and cost effectiveness. [Learn more](#)

Default (disabled) ▾

### Marketplace Private Exchange Provider

Enables publishing private listings to specific consumers as a provider. To become a public provider on Databricks Marketplace visit the [Partner Relations Management](#) portal. [Documentation](#) | [Private Exchange Terms](#)

Enable

## Enabling Databricks Web Terminal in Databricks workspace settings - Run Bash in Databricks

You should see a toggle switch next to the "Web Terminal" setting. Use the toggle switch to set the desired state (On or Off). Check the "Enforce" checkbox:

- If enforced: Workspace admins cannot change the Web Terminal setting within their workspaces. The account-level setting will be applied to all workspaces.
- If NOT enforced: Workspace admins can choose to enable or disable the Web Terminal for their individual workspaces.



**Web Terminal**

Grants all users access to the web terminal, where they can interact with the command line on their all-purpose compute resources. [Learn more](#)

---

**Enable Admin Protection for "No Isolation Shared" Clusters**

Enabling admin protection prevents automatic generation of Databricks API credentials for Databricks admins on "No Isolation Shared" clusters. Note that enabling admin protection may prevent some workloads from being run as an admin user on "No Isolation Shared" clusters. [Learn more](#)

Enforce ☒ On

☒ Enforce

Value

Apply the selected value to all workspaces

☒ On ☐ Off

When on, enforce to all workspaces. When off, allow workspace admins to enable or disable web terminal access at the workspace level.

- Enabling Databricks Web Terminal in Databricks workspace settings - Run Bash in Databricks Note: After changing the setting in the Account Console, allow a few minutes for the configuration to take effect.
- In Databricks Community Edition, head over to the Databricks Compute settings tab on the sidebar and toggle the " Web terminal " option.

**Settings**

- Workspace admin
- Identity and access
- Security
- Compute**
- Notifications
- Advanced
- User
- Profile
- Preferences
- Developer
- Notifications

**Compute**

---

**All purpose clusters**

**Global init scripts**  
Enforce consistent cluster configurations across your workspaces

---

**Web terminal**  
Enable or disable the use of web terminals in this workspace. Web terminal allows users to conveniently run shell commands and use editors, such as Vim or Emacs, on the Spark driver node.

On ☒

---

**RStudio home directory**  
Set the RStudio home directory. Users can use RStudio, a popular integrated development environment (IDE) for R, to connect to Databricks compute resources within Databricks workspaces. /home

---

**EBS SSD gp3**  
Enable the use of gp3 as the AWS EBS SSD volume type for all clusters in a Databricks workspace. Off ☐

---

**Enforce AWS instance metadata v2**  
Enforce the use of AWS Instance metadata V2 service to retrieve the instance metadata. Off ☐

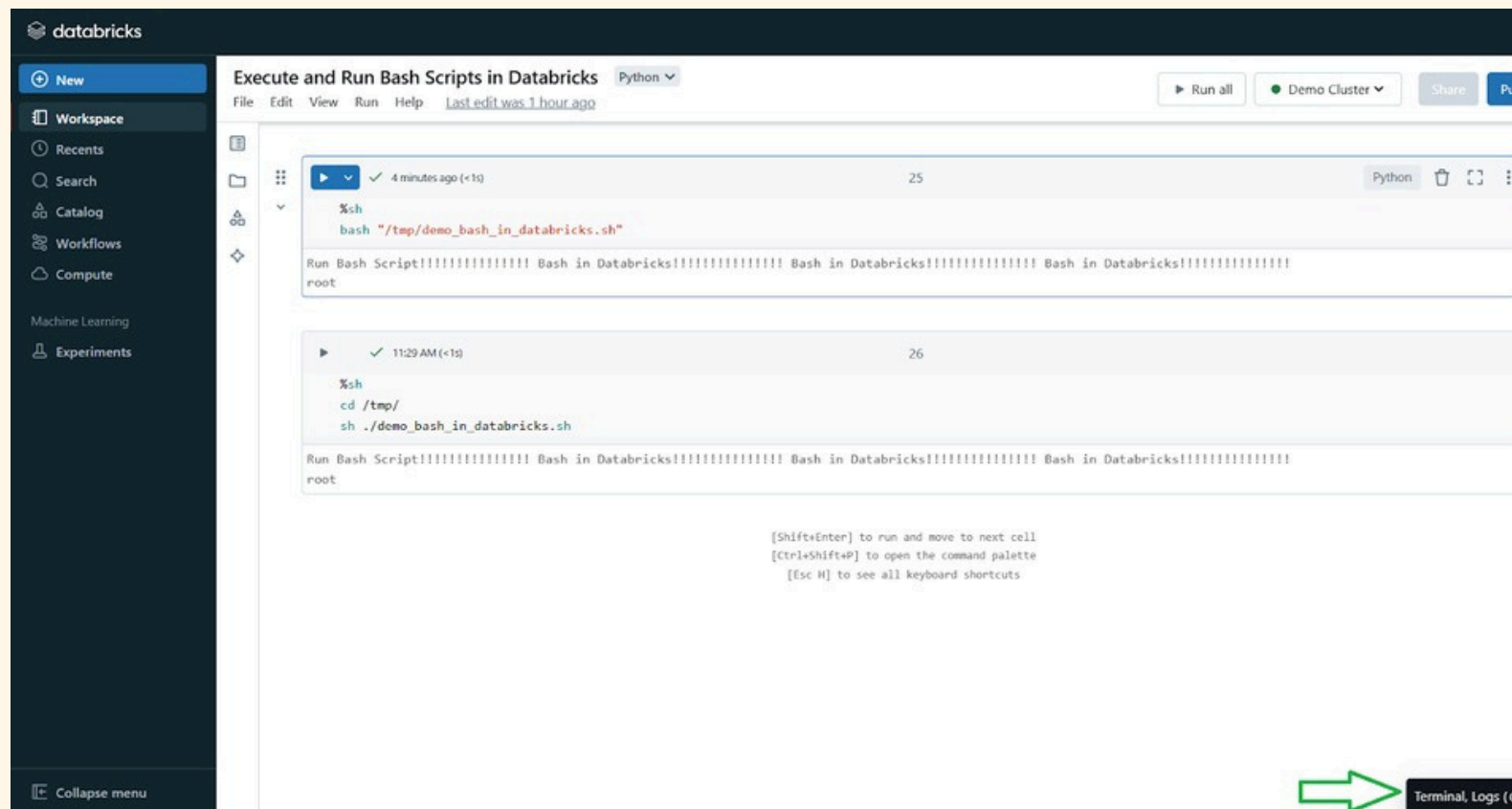
---

- Enabling Databricks Web Terminal in Databricks workspace settings - Run Bash in Databricks



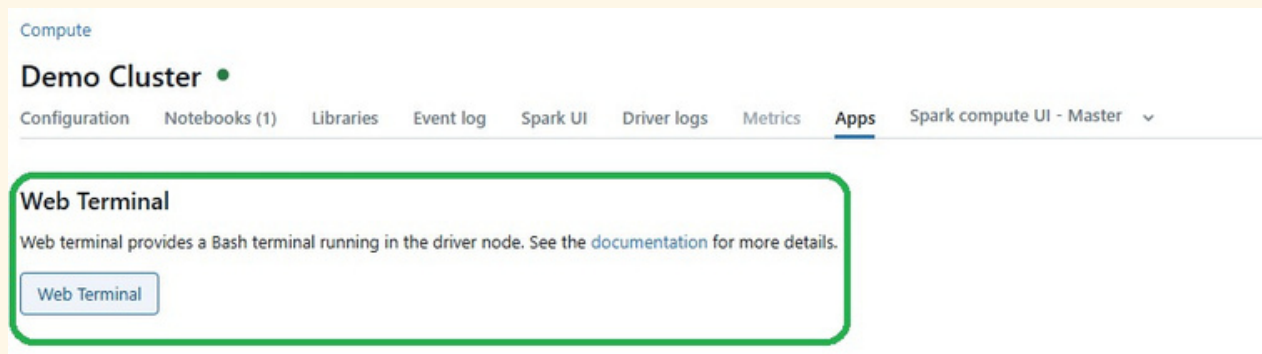
## Step 5—Launch the Web Terminal

- With the Web Terminal now enabled for your workspace, and with a Databricks compute cluster running, you can launch the Web Terminal specifically for that cluster. You have two options here: Open it directly from your Databricks Notebook by clicking the terminal icon (often found in the bottom right side panel). Or press Ctrl + `



Launching Databricks Web Terminal in Databricks - Run Bash in Databricks - Run Bash File

Alternatively, you can go to the Databricks Compute Cluster UI, select your cluster, and click on the "Web Terminal" link in the Apps tab.



Launching Databricks Web Terminal in Databricks - Run Bash in Databricks

Either way, a new tab opens with a Bash prompt on the driver node.

### Step 6—Navigating the File System and Executing Bash Commands/Scripts

Once you have the terminal open, you can move around the file system as you would in any Linux shell. Use commands like `cd` and `ls` to check directories and list files.

```
Welcome to Databricks web terminal! Please read the following instructions:

* This terminal session is ephemeral, so it will go away if you close or refresh the browser tab.
* If you want to have a persistent terminal session on this cluster, please use 'tmux'.
* There is an idle timeout if no client- or server-side changes are made to the terminal session.
  Refreshing the tab would launch a new session.

root@0212-022758-igycbmco-10-172-170-169:/databricks/driver# ls
azure conf eventlogs ganglia hadoop_accessed_config.lst logs metastore_db preload_class.lst test_dir
root@0212-022758-igycbmco-10-172-170-169:/databricks/driver#
```

Launching Databricks Web Terminal in Databricks - Run Bash Script - Run Bash in Databricks



If you've uploaded Bash scripts to DBFS or cloud storage (as discussed in Technique 2), you can access and run them from the Web Terminal, provided they are within the file system that the terminal can see. Let's say you uploaded `demo_bash_in_databricks.sh` to `/dbfs/FileStore/` or `/tmp/` earlier. Here's how you might run it in the Databricks Web Terminal:

### 1) Change Directory

You don't always have to, but if you want to be in the same directory as your script, use:

```
cd /dbfs/FileStore/
```

Or

```
cd /tmp/
```

```
root@0212-022758-igycbmco-10-172-170-169:/databricks/driver# cd /tmp/  
root@0212-022758-igycbmco-10-172-170-169:/tmp#
```

Executing Bash command in Databricks Web Terminal - Run Bash Script

### 2) Run the Script

If you made the script executable using `chmod +x` (as covered in Technique 2), you can run it directly using:

```
sh ./demo_bash_in_databricks.sh
```





If you didn't cd first, use the full path:

```
sh /dbfs/FileStore/demo_bash_in_databricks.sh
```

or

```
sh /tmp/demo_bash_in_databricks.sh
```

```
root@0212-022758-igycbmco-10-172-170-169:/tmp# sh ./demo_bash_in_databricks.sh
Run Bash Script!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!!
root
root@0212-022758-igycbmco-10-172-170-169:/tmp# █
```

Executing Bash script in Databricks Web Terminal - Run Bash Script - Run Bash in Databricks

If you skipped the chmod +x step, you could use:

```
bash /tmp/demo_bash_in_databricks.sh
```

```
root@0212-022758-igycbmco-10-172-170-169:/tmp# bash ./demo_bash_in_databricks.sh
Run Bash Script!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!! Bash in Databricks!!!!!!!!!!!!!!
root
root@0212-022758-igycbmco-10-172-170-169:/tmp# █
```

Executing Bash script in Databricks Web Terminal - Run Bash Script - Run Bash in Databricks



### **Some factors to consider:**

Since you're running commands with direct access to the node, understand the implications. It's similar to giving someone the remote control for your TV; be sure they only change the channel, not your settings.

Databricks Web Terminals are designed for interactive use, as opposed to running scripts from a Databricks notebook. You can modify files, test commands, or debug on the fly.

Databricks Web Terminal is excellent for interactive exploration, quick system checks, and any task where you need a direct, command-line interface to your cluster environment. It's especially useful when you're troubleshooting, setting up configurations manually, or using command-line tools directly on the cluster node.

### **Technique 4—Run Bash Scripts via Cluster Global Init Scripts**

Do you know you can configure settings just once and have them automatically apply to ~~every Databricks~~ compute cluster you use? Databricks Global Init Scripts make this possible. They run on every cluster in your Databricks workspace. Since this is a workspace-level setting, any Global Init Script you write in the UI will execute on all Databricks clusters you launch.

Databricks Global Init Scripts affect all nodes. If you need node-specific actions or require interactive execution, consider using one of the other techniques.

#### **Step 1—Login to Databricks**

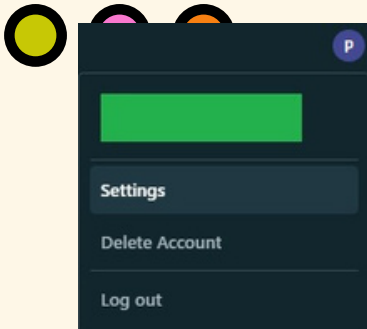
First things first, sign in to your Databricks workspace with your credentials.

#### **Step 2—Configure Databricks Compute**

Just like before, now, head to your Databricks cluster settings and check that your cluster is running or is ready to start. Your commands in the init script will run on every node when the cluster restarts.

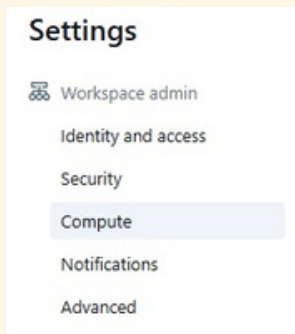
#### **Step 3—Open the Global Init Script Editor in the UI**

In the upper-right corner, click your user profile. A dropdown appears; select the settings option.

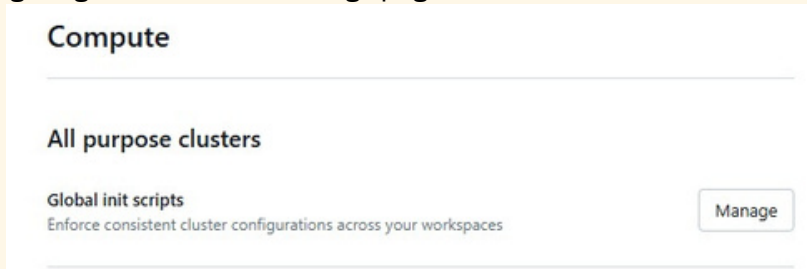


Navigating to Databricks settings page - Run Bash in Databricks

Next, go to the Databricks Compute settings tab on the sidebar and click on the Manage button for Global Init Scripts.



Navigating to Databricks settings page - Run Bash in Databricks



Configuring Databricks Global Init Scripts - Run Bash in Databricks

Here, click the Add button and provide a name for your new script.



Workspace settings > Compute >

## Global init scripts

**Note:** Changes to global init scripts do not take effect on running clusters until they restart.

Databricks recommends configuring all init scripts as cluster-scoped init scripts and managing them across your workspace using cluster policies.

Global init scripts run on all clusters in your workspace that are configured with "single user" or "legacy no-isolation shared" access mode. [Learn more](#)

Filter within all init scripts

Reset order

Apply order

Add

Order

Enabled

Name

Created

Updated



No init script configured for this workspace.

You can click Add on top right of the table to start configuring an init script.

Configuring Databricks Global Init Scripts - Run Bash in Databricks



#### Step 4—Write Your Bash Script





In the editor, type your Bash commands. This script runs on every Databricks compute cluster node at startup. For example, you can write a simple command:

```
echo "Global Init Script ran successfully"
```

This simple script helps you confirm that the init process works on every node.

#### Step 5—Toggle the Global Init Script Option

Switch the Global Init Script setting to ON. This tells Databricks to run your script on every cluster startup.

Order	Enabled	Name	Created	Updated
0	<input checked="" type="checkbox"/>	demo script	a few seconds ago by 	a few seconds ago by   

Toggling Databricks Global Init Scripts - Run Bash Script - Run Bash in Databricks

#### Step 6—Restart Your Databricks Compute Cluster

Restart your cluster. Global Init Scripts execute only during cluster startup, so a restart is required for the script to run.

#### Step 7—Monitor Init Script Execution and Troubleshoot

After the cluster restarts, open the cluster logs. Look for the message " your script accordingly.

When to Use Global Init Scripts? Global Init Script ran successfully " to confirm that your script executed on all nodes. If the expected output doesn't appear, review the logs for error messages and adjust



- Use these scripts for tasks that must run on every node, such as setting up specific environment variables or installing system libraries.
- Global Init Scripts must run without user input, so all commands should be non-interactive.
- Since these scripts execute on startup, they are not ideal for tasks you want to run repeatedly without restarting the cluster.

### **So, which technique should you choose? Here's a quick rundown:**

- 🧑‍💻 Inline Execution (%sh): Use this for quick tests or when you need to run a few commands on the driver node. It is ideal for simple file operations and command tests.
- 🧑‍💻 Stored Scripts from DBFS: Use this when you want to keep your script separate from your notebook.
- 🧑‍💻 Web Terminal: Use this for interactive sessions. It is useful when you need to debug scripts, run commands interactively, or edit files on the fly.
- 🧑‍💻 Global Init Scripts: Use these for tasks that run on every node during cluster startup. They work well for one-time setup tasks that do not require interaction.

### **Conclusion**

And that's a wrap! You've now seen how to run Bash scripts in Databricks, bringing command-line power right to your fingertips. Think about testing commands inline, running stored scripts from DBFS, or having a full interactive terminal— it's all possible. You can even apply settings to every node with Global Init Scripts. Each approach offers a different way to simplify your workflow and reduce manual steps.

In this article, we have covered:

#### **Step-By-Step Guide to Run Bash in Databricks:**

- Technique 1—Run Bash Scripts Inline in Databricks Notebook
- Technique 2—Run Stored Bash File in Databricks
- Technique 3—Run Bash Scripts via Databricks Web Terminal
- Technique 4—Run Bash Scripts via Cluster Global Init Scripts

... and so much more!



## FAQs

### What are Bash commands?

- Bash commands are text instructions you type in a Unix shell to perform tasks like file operations, process management, or system configuration.
- Can we run shell script in databricks?
- Yes, you can run shell scripts in Databricks using multiple methods such as inline in a notebook with %sh, from DBFS or mounted storage, via the Web Terminal, or as Global Init Scripts.
- What is Databricks magic command?
- Databricks Magic commands in Databricks Notebooks are special commands, prefixed with %, that extend notebook capabilities beyond the standard language (like Python or Scala). They provide shortcuts for actions like executing shell commands (%sh), running SQL queries (%sql), or changing the notebook's language context.
- What is a web terminal in Databricks?
- Databricks Web Terminal provides you with a full Linux command-line interface on the driver node. It lets you run and interact with Bash commands as if you were using a standard terminal.
- How to run terminal commands in Databricks Notebooks using %sh?
- To run terminal commands, start a notebook cell with %sh followed by your Bash commands. Your commands execute on the driver node, and the output displays directly in the cell.
- How do I troubleshoot errors when using the %sh command?
- Check the cell output for error messages, use commands like ls to verify file paths, and add simple logging with echo to track progress. For persistent issues, review the cluster logs.
- Can I run interactive Bash scripts in Databricks?
- Databricks Notebooks aren't designed for interactive Bash scripts. Use the Web Terminal if you need to run commands that require manual input.
- How do I manage environment variables in Bash scripts executed via init scripts?



Set environment variables within your script using the export command ( `export MY_VAR="value"` ), or configure them via your cluster settings. This way, the variables become available to all commands in your script.

Are there security concerns when using the Web Terminal to run Bash commands?

Yes. Run commands only from trusted sources. Since the Web Terminal gives you direct access to the driver node, restrict its use to authorized users and avoid hard-coding sensitive data.

Where is the best place to store Bash scripts for maintainability?

Store your Bash scripts in DBFS or on mounted storage. This method keeps your scripts accessible across notebooks and supports version control practices externally.

How do I create and enable a Global Init Script?

To add a global init script using the Databricks UI Admin Settings:

- Go to the Admin Settings and click the Databricks Compute tab.
- Next to "Global init scripts", click Manage.
- Click + Add.
- Name the script and enter its content in the Script field(script limit: 64KB).
- If multiple global init scripts exist, set the run order for this new script.
- To activate the script immediately for all new and restarted clusters, toggle Enabled to the ON position.
- click add save.

What are Cluster Global Init Scripts and how do they work?

Cluster Global Init Scripts run automatically on every node at cluster startup. They handle tasks like installing packages or setting environment variables before your user code runs.



**Follow for more content like this Azure  
Cloud for Data Engineering**



**Ganesh R**

**Azure Data Engineer**