

# Polly CLI Documentation

## What is Polly?

The Polly Platform is a cloud-based environment for conducting bioinformatic analyses. It is a central hub for teams to store, analyze, and jointly interpret their bioinformatic data. It allocates storage and compute resources on demand, to meet the needs of ever-growing analyses. To know more about Polly, visit <https://elucidata.io/polly>.

## What is Polly CLI?

The Polly Command Line Interface (Polly CLI) is an open source tool that enables you to interact with Polly services using commands in your command-line shell. Polly CLI lets users run jobs on the Polly cloud infrastructure by scaling computation resources as per need. Users can start and stop jobs and even monitor and view logs.

## Sign up with Polly

To be able to run jobs using Polly CLI, having a Polly account is a must. In order to sign up to Polly, contact [polly@elucidata.io](mailto:polly@elucidata.io).

## Installing Polly CLI

- **OS support**

Polly CLI is supported on Unix systems (Linux and Mac distributions).

- **Dependencies**

- [Node and npm](#)

- **Ubuntu** : For installation on Ubuntu, please follow [this link](#).

- **Mac** : For installation on Mac, please follow [this link](#).

Once npm is installed, please follow the steps mentioned [here](#) to make sure that sudo need not be used for global installation of npm packages.

- **Commands to Install** : To install Polly CLI, run the following commands on Terminal / Command prompt :

**Linux**

```
sudo npm install -g @elucidatainc/pollycli
```

**MacOS**

```
npm install -g @elucidatainc/pollycli
```

- **Commands to Uninstall :** To uninstall Polly CLI, run the following commands on Terminal / Command prompt

#### Linux

```
sudo npm uninstall -g @elucidatainc/pollycli
```

#### MacOS

```
npm uninstall -g @elucidatainc/pollycli
```

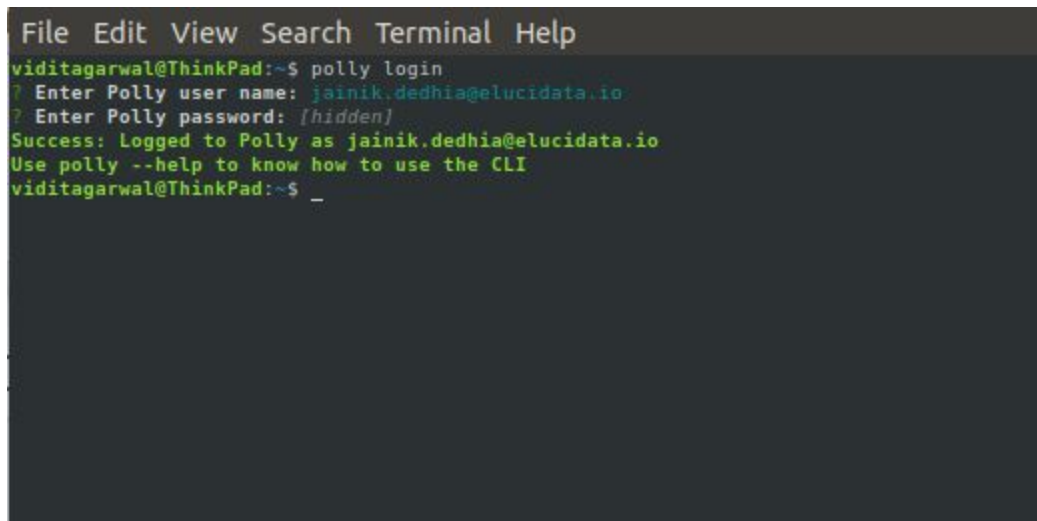
**Note :** While using Polly CLI on servers or instances, **sudo** might have to be used before every Polly command.

## Logging in and out of Polly CLI

- **Log in :** To be able to start using the features of Polly CLI, the first step is to log in to Polly using the Terminal / Command Prompt. Use the following command to log in.

```
polly login
```

Polly Username and Password need to be put in when prompted.

A terminal window with a dark background and light green text. The window has a menu bar at the top with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the following sequence of commands and output: 1. Command: 'polly login'. 2. Prompt: '? Enter Polly user name:'. Input: 'jainik.dedhia@elucidata.io'. 3. Prompt: '? Enter Polly password: [hidden]'. 4. Output: 'Success: Logged to Polly as jainik.dedhia@elucidata.io'. 5. Output: 'Use polly --help to know how to use the CLI'. 6. Prompt: 'viditagarwal@ThinkPad:~\$'. 7. A cursor is visible on the next line.

Once logged in, the user does not get automatically logged out and will have to log out manually with the command to log out.

- **Log out :** In order to log out of Polly, use the following command.

```
polly logout
```

## Polly Workspaces

- **What are Polly Workspaces?**

Polly workspaces are online workspaces that contain data, analyses, code, logs etc for a specific project or experiment. Data is stored and Analysis is performed within a user chosen workspace.

Workspaces can be managed on [polly.elucidata.io](https://polly.elucidata.io) or Polly CLI. A Workspace contains Workspace ID, Workspace name and Workspace description.

- **Create a new Workspace**

To create a new workspace, use the following command :

```
polly workspaces create
```

Users will be asked to name the Workspace and provide a description. Once the workspace is created, the workspace name and ID will be shown on the screen. This Workspace ID will be needed while creating a json file for running jobs.

```
viditagarwal@ThinkPad:~$ polly workspaces create
? Enter the name of the workspace you would like to create: New Workspace
? Enter the description of the workspace that you would like to create: This is a workspace intended
  for running all the new jobs
Success: Project created
```

Workspace ID	Workspace name
3108	New Workspace

```
viditagarwal@ThinkPad:~$ _
```

- **View existing Workspaces**

To view all the existing Workspaces with access, use the following command :

```
polly workspaces list
```

Users will be asked to select which Workspaces to list - **all, latest or oldest**. On selecting all, all the Workspaces will be listed. On selecting, oldest or latest, the user will be asked to enter the number of workspaces to be shown as shown in the image below.

```
viditagarwal@ThinkPad:~$ polly workspaces list
? You want to see all, latest or older workspaces: latest
? How many latest workspaces you would like to see: 2
```

Workspace ID	Workspace name
3108	New Workspace
3107	CLI test

```
viditagarwal@ThinkPad:~$ _
```

## Start Jobs

- **Create job description json file**

Json file is needed to describe the job to be run on Polly. This file should contain the information about the computational resources (machine), docker image, the name of the job and specific commands (if required) to be run after the docker has been run, as keys. Text can be copy pasted from the example below to create the json file.

```
{
  "machineType" : "gp",
  "cpu": 1,
  "memory": "1Gi",
  "image": "docker/whalesay",
  "tag": "latest",
  "name": "Single Cell RNA",
  "command": [
    "cowsay", "hello world"
  ]
}
```

- **machineType** : Name of the machine required to run the job needs to be mentioned as per the following table.

machineType	No. of vCPUs	Memory (RAM)	No. of GPUs
gp	4	16 GB	-
ci2xlarge	16	32 GB	-
ci3xlarge	36	72 GB	-
mi2xlarge	4	32 GB	-
mi3xlarge	8	64 GB	-
mi4xlarge	16	122 GB	-

More machines (including some with GPUs) will be added soon.

If computational power required is less than 2 vCPUs and 8 GB RAM, use the keys “**cpu**” and “**memory**” in the json file instead of the key “**machineType**”. If all 3 keys are present, “**machineType**” takes priority and the machine will be assigned accordingly. In the

example json (image) mentioned above, machine selected will be **“gp”** with 4 vCPUs and 16 GB RAM and **NOT** 1vCPU and 1 GB RAM.

- **cpu** : Mention the number of CPUs needed here. For smaller jobs, just a part of the CPU can also be chosen. For example, if 0.1 vCPUs are required for the job, the number of CPUs can be mentioned as **“100m”**. If more than 2 CPUs are required for the job, use the key **“machinType”** to choose the relevant machine instead of **“cpu”** and **“memory”**.
- **memory** : RAM required needs to be mentioned in text (eg - “1Gi” or “500 Mi”) in this key. If memory needed is more than 8 GB, use the key **“machinType”** to choose the relevant machine instead of **“cpu”** and **“memory”**.
- **image** : The path to the docker image present in DockerHub or ECR needs to be mentioned in this key .
- **tag** : Tag of the docker image needs to be mentioned in this key.
- **name** : Name that user wants to provide to the job has to be mentioned in this key.
- **command** : Any commands to be executed after the docker has been run can be mentioned in this key.

- **Docker building guidelines :**

While creating a docker to be run on Polly, the following must be taken care of :

- Dockers must be present in either Docker Hub or Amazon ECR.  
Soon users will be able to have Dockers directly on Polly.
- Only self contained dockers can be run on Polly. A self contained docker is one which has the code to get input files as well as upload output files back contained in the docker..
- Public as well as Private dockers are supported. In order to run Private dockers, **“secret”** should be passed as a key in the json file. To get the secret key for the private docker, the following steps need to be followed.
  - For MacOS, users need to remove the key value pair **“credsStore”**:  
**“osxkeychain”** from the **config.json** file present in the directory **“/Users/<username>/docker”**.
  - The user needs to be logged in to DockerHub or ECR through the terminal. If not, the user will need to log in.
  - Run the command **“sudo polly”** on the Terminal.
  - Select the option **“miscellaneous”** followed by **“create secret for docker”**.
  - Provide the path to the docker config file (the usual path for docker config is **/Users/<username>/docker/config.json** in Mac and

/home/<username>/.docker/config.json in Linux).

**Note** - Relative paths are not supported.

- Select the account in which the docker to be run is present.
- Copy the long text string (secret key) output to the json file in the key “**secret**”.

```
{
  "cpu": 1,
  "memory": "1Gi",
  "image": "docker/whalesay",
  "tag": "latest",
  "Secret": "ewoAImFldVnphR0ZzWjNWd2RHRTZSVkJKUXlOcFlXMGsiCgkw==",
  "name": "exampleName",
  "command": [
    "cowsay", "hello world"
  ]
}
```

- **Passing Environment variables** : Two types of Environment variables can be passed in the json file.
  - **Normal environment variables** are saved in a database for future references. These can be passed in the parameter “**env**” in the json file.
  - **Private environment variables** are not saved in any database. These can be used for passing credentials in the json file. These can be passed in the parameter “**secret\_env**” in the json file.

**Note** - The value of Environment variables should always be string. For example, the correct way to assign Environment variable is {“parallel\_threads” : “2”} and **NOT** {“parallel\_threads” : 2}

```
{
  "cpu": "100m",
  "memory": "64Mi",
  "image": "your_docker",
  "tag": "latest",
  "env": {
    "ENV1": "ENV_VALUE1",
    "ENV2": "ENV_VALUE2"
  },
  "secret_env": {
    "SECRET_ENV1": "SECRET_ENV_VALUE1",
  }
}
```

```

"SECRET_ENV2": "SECRET_ENV_VALUE2"
},
"name": "docker running"
}

```

- **Execute job :** To execute the job, enter the following command :

```
polly jobs submit
```

On executing this command, the user will be asked to enter the id of the workspace where the job should be run and the path to the job description json file. With this, the job will be submitted to run and Job ID will be created. This Job ID will be needed to check the status and the logs of the submitted job.

```

viditagarwal@ThinkPad:~$ polly jobs submit
? Enter the workspace ID: 3108
? Enter path to the job discription file (.json): /home/viditagarwal/Desktop/job.json
Success: Submitted the job to run!

```

Workspace ID	Job ID
3108	748f6cb3338542e1ac194c695b4e4591

```

Useful commands:
status : polly jobs status --workspace-id 3108 --job-id 748f6cb3338542e1ac194c695b4e4591
viditagarwal@ThinkPad:~$ _

```

## Monitor Job status

- **Get job status :**
  - The following command can be used to view the status of a particular job :

```
polly jobs status --workspace-id <workspace id> --job-id <job id>
```

This will give the status of the job as shown below.

```

viditagarwal@ThinkPad:~$ polly jobs status --workspace-id 3108 --job-id 748f6cb3338542e1ac194c695b4e4591

```

Job ID	Job Name	State
748f6cb3338542e1ac194c695b4e4591	exampleName	SUCCESS

- The following command can be used to view the statuses of all the jobs in a workspace :

```
polly jobs status --workspace-id <workspace id>
```

A prompt to enter job id will appear which when kept

```

viditagarwal@ThinkPad:~$ polly jobs status --workspace-id 4710
? Enter the job ID:

```

Job ID	Job Name	State
f9ced1780fbd46d78256f8089592c069	Single Cell RNA	SUCCESS
3e7c2647462c4b7f86395f0104bd5130	Fluxomics	SUCCESS
06ab26f9e62e43519bc531b6f43444c6	RNAseq	SUCCESS
e546e084153a4848a78cecl1a4f0184e4	exampleName	SUCCESS
9c7dfc3f70e24822afc4724b0d5396ab	exampleName	SUCCESS

- polly.jobs.logs workspace\_id <workspace id> job\_id <job id>

```
viditagarwal@ThinkPad:~$ polly jobs logs --workspace-id 3108 --job-id 748f6cb3338542e1ac194c695b
```

More details

- PollyCLI help

- If help is needed for any command, just type `--help` at the end of the command and execute.

```
viditagarwal@ThinkPad:~$ polly jobs submit --help
```