

About ovhai CLI

ovhai is a CLI that allow you to interact with OVHcloud AI Training (<https://www.ovhcloud.com/fr/public-cloud/ai-training/>). OVHcloud AI training is a cloud service that allows Data Scientists, ML Engineers and Deep Learning practitioners to create and allocate training ressources on demand.

With the CLI you can: - manage jobs (list, run, stop, execute commands inside...) - manage AI notebooks (start, stop, sync...) - manage object storage (upload data to object storage, attach data to a notebook, delete a bucket/container ...) - manage apps (list, run, delete...) - manage containers registries (list, add a private registry...) - manage access tokens - ...

Installation

Install through curl

Define the region and then download the binary according to your OS:

```
$ export REGION=gra ; curl -O https://cli.$REGION.trai
$ unzip ovhai-darwin.zip
$ sudo mv ovhai /usr/local/bin/ovhai
$ rm ovhai-darwin.zip
```

Prerequisites

Create an OpenStack user

Create an OpenStack User with the roles "AI Training Operator" and "ObjectStore operator". Follow the guide: <https://docs.ovh.com/gb/en/ai-training/create-user/>

Usage

Show version

```
$ ovhai --version
```

```
ovhai 2.3.1
```

Login

```
$ ovhai login
```

It's the first command you need to execute. To login into AI Training platform you need the credentials of a user with the roles "AI Training Operator" and ObjectStore operator. Enter the user name and password.

Show current user information

```
$ ovhai me
```

```
Name:      user-NnnmWdWTFjn3
Tenant:    a123a4e56b789c1ba23a456b789fcf01
Trust Id:  bealc2cc345e67a89adf1023c45ccd67
```

Show all the environments

```
$ ovhai config ls
```

```
NAME URL
BHS  https://bhs.training.ai.cloud.ovh.net/v1
GRA  https://gra.training.ai.cloud.ovh.net/v1
```

Jobs

A job is a Docker container running on the OVHcloud AI infrastructure. You can you specify a Docker image (from Docker Hub or a private registry), allocate the request hardware requirements (number of GPUs, number of CPUs), link the data in input and output to an object storage through volumes, and run it.

List jobs

```
$ ovhai job ls
```

Run a simple job

```
$ ovhai job run ubuntu-bash -c "while true; do
echo toto; sleep 2; done"
```

Run a job and pass your SSH key

```
$ ovhai job run ubuntu -s $HOME/.ssh/id_rsa.pub
-bash -c "sleep infinity"
```

Run a job, with 1 CPU

```
$ ovhai job run <docker_image> --cpu 1-sleep
infinity
```

-c, --cpu: number of CPUs (ignored if GPUs is specified)

Run a job, with 2 GPUs and plug it to an object storage container (bucket) in read only access mode

```
$ ovhai job run <docker_image> --gpu 2 -v
<container_name>@<REGION>:/data:ro-sleep
infinity
```

-g, --gpu: number of GPUs (default 1)

Get job status

```
$ ovhai job get <job_ID>
```

Diplay job logs

```
$ ovhai job logs <job_ID>
```

Diplay job logs and stream the logs

```
$ ovhai job logs -f <job_ID>
```

Execute commands in a job

```
$ ovhai job exec -it <job_ID>-bash
```

Manually push data from tObject Storage to the job (while it is running)

```
$ ovhai job push-data <job_ID>
```

Manually pull data from Object Storage to the job (while it is running)

```
$ ovhai job pull-data <job_ID>
```

Stop a job

```
$ ovhai job stop <job_ID>
```

Notebooks

```
xxx
```

```
xxx
```

Apps

```
xxxx
```

```
xxx
```

Data (Object Storage)

Jobs are containers so by default there are ephemeral, it means that if you store data inside jobs, you can lose them. The solution is to link your jobs and notebooks to Object Storage containers (buckets).

List Object Storage containers

```
$ ovhai data ls <REGION>
```

List Object Storage containers that starting with "test%"

```
$ ovhai data ls <REGION> --prefix test
```

Push a file (an object) to my-bucket on

```
$ ovhai data upload <REGION> my-bucket
some/local-file
```

TODO: xxx

How to upload several objects ? > ovhai data upload GRA my-container some/local other-local --add-prefix a-prefix/

How to delete an object ? > ovhai data delete GRA my-container some-object

How to delete several objects ? > ovhai data delete GRA my-container --prefix a-prefix > ovhai data delete GRA my-container --all

How to delete a container ? > ovhai data delete GRA my-container

How to delete several containers ? > ovhai data delete GRA --prefix a-prefix > ovhai data delete GRA --all

xxx

Registries

xxx

xxx

...

...

This command is useful when you have defined some modules. Modules are vendored so when you edit them, you need to get again modules content.

```
$ terraform get -update=true
```

When you use modules, the first thing you'll have to do is to do a terraform get. This pulls modules into the .terraform directory. Once you do that, unless you do another terraform get -update=true, you've essentially vendored those modules.

Plan

The plan step check configuration to execute and write a plan to apply to target infrastructure provider.

```
$ terraform plan -out plan.out
```

It's an important feature of Terraform that allows a user to see which actions Terraform will perform prior to making any changes, increasing confidence that a change will have the desired effect once applied.

When you execute terraform plan command, terraform will scan all *.tf files in your directory and create the plan.


Apply

Now you have the desired state so you can execute the plan.

```
$ terraform apply plan.out
```

```
$ terraform apply
```

Authors :

 **@aurelievache**
DevRel at OVHcloud

v1.0.0