OVHcloud ovhai CLI Cheat Sheet

About ovhai CLI

`ovhai` is a CLI that allows 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.

Installation

Install through curl

Define the region & launch the curl command.

For bash:

```
export REGION=gra
curl
https://cli.$REGION.training.ai.cloud.ovh.net/install.sh |
bash
```

For zsh:

```
export REGION=gra
curl
https://cli.$REGION.training.ai.cloud.ovh.net/install.sh |
zsh
```

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
```

Upgrade the CLI

```
ovhai upgrade
```

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
```

Show all the regions and environments

```
ovhai config ls
```

Switch to another region/environment

```
ovhai config set <REGION>
```

Show available flavors

```
ovhai capabilities flavor ls
```

Jobs

A job is a Docker container running on the OVHcloud AI infrastructure. You can 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.

A job have a start and an end.

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, with 1 CPU and pass your SSH key

```
ovhai job run ubuntu --cpu 1 -s $HOME/.ssh/id_rsa.pub --
bash -c "sleep infinity"
```

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
```

Get job status

```
ovhai job get <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
```

Synchronize - Manually push data from Job to Object Storage (while it is running)

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

Synchronize - Manually pull data from Object Storage to Job (while it is running)

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

Stop a job

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

Data (Object Storage)

Object Storage is a scalable, resilient and secure storage place accessible from anywhere through HTTPS APIs. It is a perfect place to store static files on the long term. Object Storage can be used to persist any data needed by jobs, notebooks or apps.

List Object Storage containers

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

List Object Storage containers that starting with "test%"

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

Push files (objects) to my-container

```
ovhai data upload <REGION> my-container some/local-file
other-file
```

Delete an object on my-container and all their objects

```
ovhai data delete <REGION> my-container my-object --all
```

Delete my-container

```
ovhai data delete <REGION> my-container
```

Delete all of your containers starting by "test%"

```
ovhai data delete <REGION> --prefix test
```

Delete all of your containers

```
ovhai data delete <REGION> --all
```

Deploy

An app is like a job but for API or daemon process that should never stop. An app runs as a group of load balanced Docker containers within OVHcloud AI infrastructure.

You can specify a Docker image (from AI Training shared registry, 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 apps

```
ovhai app ls
```

Run an app and specify we want 3 replicas

```
ovhai app run <registry>/<image>:latest -p 8080 --cpu 1 --
fixed-scale 3
```

Run an app and mount a volume linked to an Object Storage container

```
ovhai app run <docker-image> --gpu 4 --volume my-
container@<REGION>:/data
```

Get app status

```
ovhai app get <app_ID>
```

Get app's URL

```
ovhai app get <app_ID> -o json | jq ".status.url"
```

Stop an app

```
ovhai app stop <app_ID>
```

Delete an app

```
ovhai app delete <app_ID>
```

Warning: you need to stop the app before executing this

Notebooks

Notebook is used to easily work with one of the well-known Machine Learning frameworks on either JupyterLab or VSCode and powerful hardware.

Already installed for you, and that you pay only for your notebooks while they are running.

Display available Machine Learning frameworks

```
ovhai capabilities framework ls
```

Display available editors for notebooks

```
ovhai capabilities editor ls
```

List notebooks

```
ovhai notebook ls
```

Run a notebook using PyTorch and JupyterLab, with 1 GPU and allow access to it without authentication

```
ovhai notebook run --gpu 1 --unsecure-http pytorch
jupyterlab
```

--unsecure-http: allow to bypass authentication with an OpenStack user

Run a notebook with a specified framework version, 1 CPU and mount a volume linked to an Object Storage container

```
ovhai notebook run --framework-version pytorch1.10.1-py39-
cuda10.2-v22-4 --flavor ai1-1-cpu --cpu 1 -v my-
container@<REGION>:/data:ro pytorch jupyterlab
```

Run a notebook and give access to it for people outside of your Public Cloud project

If you want to share and give access to your jobs, apps or notebooks to people outside of your Public Cloud project, you can generate an access token.

Create a token named **my-token**, that will allow to access any notebook that has a label **subject=image-recognition**:

```
ovhai token create my-token --role read --label-selector
subject=image-recognition
```

Run a notebook with **image-recognition** label:

```
ovhai notebook run --gpu 1 --label subject=image-recognition
pytorch vscode
```

Go to the URL on your browser, click on **Login with token** and enter the token.

Get notebook information

```
ovhai notebook get <notebook_ID>
```

Start a stopped notebook

```
ovhai notebook start <notebook_ID>
```

Stop a notebook

```
ovhai notebook stop <notebook_ID>
```

Delete a notebook

```
ovhai notebook delete <notebook_ID>
```

Registries

A registry is a place where you can push and pull your Docker images. By default, you have access to a shared registry scoped to your project, you can only push images and use them for jobs and apps. You can add and delete private registries.

List registries

```
ovhai registry ls
```

Add a private registry

```
ovhai registry add <url>
```

Get registry information

```
ovhai registry get <registry_ID>
```

Delete a private registry

```
ovhai registry delete <registry_ID>
```

Debug

The debug command is useful in order to display logs about a specified command executed by the CLI.

Debug a command

Run a command, for example:

```
ovhai app delete <app_ID>
```

If the command fails, you will have a command number to debug:

```
ovhai debug <command_number>
```