Command Line Interface and Environment Variables Reference

Co
En

Command Line Interface

Airflow has a very rich command line interface that allows for many types of operation on a DAG, starting services, and supporting development and testing.

Note

For more information on usage CLI, see Using the Command Line Interface

Content

```
usage: airflow [-h] GROUP_OR_COMMAND ...
```

Positional Arguments

`GROUP_OR_COMMAND`

Possible choices: celery, cheat-sheet, config, connections, dags, db, info, kerberos, kubernetes, plugins, pools, providers, roles, rotate-fernet-key, scheduler, sync-perm, tasks, users, variables, version, webserver

Sub-commands:

celery

Start celery components. Works only when using CeleryExecutor. For more information, see https://airflow.apache.org/docs/stable/executor/celery.html

```
airflow celery [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: flower, stop, worker

Sub-commands:

flower

Start a Celery Flower

```
airflow celery flower [-h] [-A BASIC_AUTH] [-a BROKER_API] [-D]
                      [-c FLOWER_CONF] [-H HOSTNAME] [-l LOG_FILE]
                      [--pid [PID]] [-p PORT] [--stderr STDERR]
                      [--stdout STDOUT] [-u URL_PREFIX]
```

Named Arguments¶

`-A, --basic-auth`

Securing Flower with Basic Authentication. Accepts user:password pairs separated by a comma. Example: flower_basic_auth = user1:password1,user2:password2

Default: “”

`-a, --broker-api`

Broker API

`-D, --daemon`

Daemonize instead of running in the foreground

Default: False

`-c, --flower-conf`

Configuration file for flower

`-H, --hostname`

Set the hostname on which to run the server

Default: “0.0.0.0”

`-l, --log-file`

Location of the log file

`--pid`

PID file location

`-p, --port`

The port on which to run the server

Default: 5555

`--stderr`

Redirect stderr to this file

`--stdout`

Redirect stdout to this file

`-u, --url-prefix`

URL prefix for Flower

Default: “”

stop

Stop the Celery worker gracefully

```
airflow celery stop [-h]
```

worker

Start a Celery worker node

```
airflow celery worker [-h] [-a AUTOSCALE] [-H CELERY_HOSTNAME]
                      [-c CONCURRENCY] [-D] [-l LOG_FILE] [--pid [PID]]
                      [-q QUEUES] [-s] [--stderr STDERR] [--stdout STDOUT]
                      [-u UMASK]
```

Named Arguments¶

`-a, --autoscale`

Minimum and Maximum number of worker to autoscale

`-H, --celery-hostname`

Set the hostname of celery worker if you have multiple workers on a single machine

`-c, --concurrency`

The number of worker processes

Default: 16

`-D, --daemon`

Daemonize instead of running in the foreground

Default: False

`-l, --log-file`

Location of the log file

`--pid`

PID file location

`-q, --queues`

Comma delimited list of queues to serve

Default: “default”

`-s, --skip-serve-logs`

Don’t start the serve logs process along with the workers

Default: False

`--stderr`

Redirect stderr to this file

`--stdout`

Redirect stdout to this file

`-u, --umask`

Set the umask of celery worker in daemon mode

Default: “0o077”

cheat-sheet

Display cheat sheet

```
airflow cheat-sheet [-h]
```

config

View configuration

```
airflow config [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: get-value, list

Sub-commands:

get-value

Print the value of the configuration

```
airflow config get-value [-h] section option
```

Positional Arguments¶

`section`

The section name

`option`

The option name

list

List options for the configuration

```
airflow config list [-h] [--color {auto,off,on}]
```

Named Arguments¶

`--color`

Possible choices: auto, off, on

Do emit colored output (default: auto)

Default: “auto”

connections

Manage connections

```
airflow connections [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: add, delete, export, get, list

Sub-commands:

add

Add a connection

```
airflow connections add [-h] [--conn-description CONN_DESCRIPTION]
                        [--conn-extra CONN_EXTRA] [--conn-host CONN_HOST]
                        [--conn-login CONN_LOGIN]
                        [--conn-password CONN_PASSWORD]
                        [--conn-port CONN_PORT] [--conn-schema CONN_SCHEMA]
                        [--conn-type CONN_TYPE] [--conn-uri CONN_URI]
                        conn_id
```

Positional Arguments¶

`conn_id`

Connection id, required to get/add/delete a connection

Named Arguments¶

`--conn-description`

Connection description, optional when adding a connection

`--conn-extra`

Connection *Extra* field, optional when adding a connection

`--conn-host`

Connection host, optional when adding a connection

`--conn-login`

Connection login, optional when adding a connection

`--conn-password`

Connection password, optional when adding a connection

`--conn-port`

Connection port, optional when adding a connection

`--conn-schema`

Connection schema, optional when adding a connection

`--conn-type`

Connection type, required to add a connection without conn_uri

`--conn-uri`

Connection URI, required to add a connection without conn_type

delete

Delete a connection

```
airflow connections delete [-h] [--color {auto,off,on}] conn_id
```

Positional Arguments¶

`conn_id`

Connection id, required to get/add/delete a connection

Named Arguments¶

`--color`

Possible choices: auto, off, on

Do emit colored output (default: auto)

Default: “auto”

export

All connections can be exported in STDOUT using the following command: airflow connections export - The file format can be determined by the provided file extension. eg, The following command will export the connections in JSON format: airflow connections export /tmp/connections.json The –format parameter can be used to mention the connections format. eg, the default format is JSON in STDOUT mode, which can be overridden using: airflow connections export - –format yaml The –format parameter can also be used for the files, for example: airflow connections export /tmp/connections –format json

```
airflow connections export [-h] [--format {json,yaml,env}] file
```

Positional Arguments¶

`file`

Output file path for exporting the connections

Named Arguments¶

`--format`

Possible choices: json, yaml, env

Format of the connections data in file

get

Get a connection

```
airflow connections get [-h] [--color {auto,off,on}] [-o table, json, yaml]
                        conn_id
```

Positional Arguments¶

`conn_id`

Connection id, required to get/add/delete a connection

Named Arguments¶

`--color`

Possible choices: auto, off, on

Do emit colored output (default: auto)

Default: “auto”

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

list

List connections

```
airflow connections list [-h] [--conn-id CONN_ID] [-o table, json, yaml]
```

Named Arguments¶

`--conn-id`

If passed, only items with the specified connection ID will be displayed

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: "table"

dags

Manage DAGs

```
airflow dags [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: backfill, delete, list, list-jobs, list-runs, next-execution, pause, report, show, state, test, trigger, unpause

Sub-commands:

backfill

Run subsections of a DAG for a specified date range. If reset_dag_run option is used, backfill will first prompt users whether airflow should clear all the previous dag_run and task_instances within the backfill date range. If rerun_failed_tasks is used, backfill will auto re-run the previous failed task instances within the backfill date range

```
airflow dags backfill [-h] [-c CONF] [--delay-on-limit DELAY_ON_LIMIT] [-x]
                      [-n] [-e END_DATE] [-i] [-I] [-l] [-m] [--pool POOL]
                      [--rerun-failed-tasks] [--reset-dagruns] [-B]
                      [-s START_DATE] [-S SUBDIR] [-t TASK_REGEX] [-v] [-y]
                      dag_id
```

Positional Arguments¶

`dag_id`

The id of the dag

Named Arguments¶

`-c, --conf`

JSON string that gets pickled into the DagRun's conf attribute

`--delay-on-limit`

Amount of time in seconds to wait when the limit on maximum active dag runs (max_active_runs) has been reached before trying to execute a dag run again

Default: 1.0

`-x, --donot-pickle`

Do not attempt to pickle the DAG object to send over to the workers, just tell the workers to run their version of the code

Default: False

`-n, --dry-run`

Perform a dry run for each task. Only renders Template Fields for each task, nothing else

Default: False

`-e, --end-date`

Override end_date YYYY-MM-DD

`-i, --ignore-dependencies`

Skip upstream tasks, run only the tasks matching the regexp. Only works in conjunction with task_regex

Default: False

`-I, --ignore-first-depends-on-past`

Ignores depends_on_past dependencies for the first set of tasks only (subsequent executions in the backfill DO respect depends_on_past)

Default: False

`-l, --local`

Run the task using the LocalExecutor

Default: False

`-m, --mark-success`

Mark jobs as succeeded without running them

Default: False

`--pool`

Resource pool to use

`--rerun-failed-tasks`

if set, the backfill will auto-rerun all the failed tasks for the backfill date range instead of throwing exceptions

Default: False

`--reset-dagruns`

if set, the backfill will delete existing backfill-related DAG runs and start anew with fresh, running DAG runs

Default: False

`-B, --run-backwards`

if set, the backfill will run tasks from the most recent day first. if there are tasks that depend_on_past this option will throw an exception

Default: False

`-s, --start-date`

Override start_date YYYY-MM-DD

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

`-t, --task-regex`

The regex to filter specific task_ids to backfill (optional)

`-v, --verbose`

Make logging output more verbose

Default: False

`-y, --yes`

Do not prompt to confirm reset. Use with care!

Default: False

delete

Delete all DB records related to the specified DAG

```
airflow dags delete [-h] [-y] dag_id
```

Positional Arguments¶

`dag_id`

The id of the dag

Named Arguments¶

`-y, --yes`

Do not prompt to confirm reset. Use with care!

Default: False

list

List all the DAGs

```
airflow dags list [-h] [-o table, json, yaml] [-S SUBDIR]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to ‘[AIRFLOW_HOME]/dags’ where [AIRFLOW_HOME] is the value you set for ‘AIRFLOW_HOME’ config you set in ‘airflow.cfg’

Default: “[AIRFLOW_HOME]/dags”

list-jobs

List the jobs

```
airflow dags list-jobs [-h] [-d DAG_ID] [--limit LIMIT] [-o table, json, yaml]
                       [--state STATE]
```

Named Arguments¶

`-d, --dag-id`

The id of the dag

`--limit`

Return a limited number of records

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

`--state`

Only list the dag runs corresponding to the state

list-runs

List DAG runs given a DAG id. If state option is given, it will only search for all the dagruns with the given state. If no_backfill option is given, it will filter out all backfill dagruns for given dag id. If start_date is given, it will filter out all the dagruns that were executed before this date. If end_date is given, it will filter out all the dagruns that were executed after this date.

```
airflow dags list-runs [-h] [-d DAG_ID] [-e END_DATE] [--no-backfill]
                       [-o table, json, yaml] [-s START_DATE] [--state STATE]
```

Named Arguments¶

`-d, --dag-id`

The id of the dag

`-e, --end-date`

Override end_date YYYY-MM-DD

`--no-backfill`

filter all the backfill dagruns given the dag id

Default: False

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: "table"

`-s, --start-date`

Override start_date YYYY-MM-DD

`--state`

Only list the dag runs corresponding to the state

next-execution

Get the next execution datetimes of a DAG. It returns one execution unless the num-executions option is given

```
airflow dags next-execution [-h] [-n NUM_EXECUTIONS] [-S SUBDIR] dag_id
```

Positional Arguments¶

`dag_id`

The id of the dag

Named Arguments¶

`-n, --num-executions`

The number of next execution datetimes to show

Default: 1

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

pause

Pause a DAG

```
airflow dags pause [-h] [-S SUBDIR] dag_id
```

Positional Arguments¶

dag_id

The id of the dag

Named Arguments¶

-S, --subdir

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

report

Show DagBag loading report

```
airflow dags report [-h] [-o table, json, yaml] [-S SUBDIR]
```

Named Arguments¶

-o, --output

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: "table"

-S, --subdir

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

show

The –imgcat option only works in iTerm.

For more information, see: https://www.iterm2.com/documentation-images.html

The –save option saves the result to the indicated file.

The file format is determined by the file extension. For more information about supported format, see: https://www.graphviz.org/doc/info/output.html

If you want to create a PNG file then you should execute the following command: airflow dags show <DAG_ID> –save output.png

If you want to create a DOT file then you should execute the following command: airflow dags show <DAG_ID> –save output.dot

```
airflow dags show [-h] [--imgcat] [-s SAVE] [-S SUBDIR] dag_id
```

Positional Arguments¶

dag_id

The id of the dag

Named Arguments¶

--imgcat

Displays graph using the imgcat tool.

Default: False

`-s, --save`

Saves the result to the indicated file.

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

state

Get the status of a dag run

```
airflow dags state [-h] [-S SUBDIR] dag_id execution_date
```

Positional Arguments¶

`dag_id`

The id of the dag

`execution_date`

The execution date of the DAG

Named Arguments¶

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

test

Execute one single DagRun for a given DAG and execution date, using the DebugExecutor.

The –imgcat-dagrun option only works in iTerm.

For more information, see: https://www.iterm2.com/documentation-images.html

If –save-dagrun is used, then, after completing the backfill, saves the diagram for current DAG Run to the indicated file. The file format is determined by the file extension. For more information about supported format, see: https://www.graphviz.org/doc/info/output.html

If you want to create a PNG file then you should execute the following command: airflow dags test <DAG_ID> <EXECUTION_DATE> –save-dagrun output.png

If you want to create a DOT file then you should execute the following command: airflow dags test <DAG_ID> <EXECUTION_DATE> –save-dagrun output.dot

```
airflow dags test [-h] [--imgcat-dagrun] [--save-dagrun SAVE_DAGRUN]
                  [--show-dagrun] [-S SUBDIR]
                  dag_id execution_date
```

Positional Arguments¶

`dag_id`

The id of the dag

`execution_date`

The execution date of the DAG

Named Arguments¶

`--imgcat-dagrun`

After completing the dag run, prints a diagram on the screen for the current DAG Run using the imgcat tool.

Default: False

`--save-dagrun`

After completing the backfill, saves the diagram for current DAG Run to the indicated file.

`--show-dagrun`

After completing the backfill, shows the diagram for current DAG Run.

The diagram is in DOT language

Default: False

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

trigger

Trigger a DAG run

```
airflow dags trigger [-h] [-c CONF] [-e EXEC_DATE] [-r RUN_ID] [-S SUBDIR]
                     dag_id
```

Positional Arguments¶

`dag_id`

The id of the dag

Named Arguments¶

`-c, --conf`

JSON string that gets pickled into the DagRun's conf attribute

`-e, --exec-date`

The execution date of the DAG

`-r, --run-id`

Helps to identify this run

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

unpause

Resume a paused DAG

```
airflow dags unpause [-h] [-S SUBDIR] dag_id
```

Positional Arguments¶

`dag_id`

The id of the dag

Named Arguments¶

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

db

Database operations

```
airflow db [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: check, check-migrations, init, reset, shell, upgrade

Sub-commands:

check

Check if the database can be reached

```
airflow db check [-h]
```

check-migrations

Check if migration have finished (or continually check until timeout)

```
airflow db check-migrations [-h] [-t MIGRATION_WAIT_TIMEOUT]
```

Named Arguments¶

`-t, --migration-wait-timeout`

timeout to wait for db to migrate

Default: 0

init

Initialize the metadata database

```
airflow db init [-h]
```

reset

Burn down and rebuild the metadata database

```
airflow db reset [-h] [-y]
```

Named Arguments¶

`-y, --yes`

Do not prompt to confirm reset. Use with care!

Default: False

shell

Runs a shell to access the database

```
airflow db shell [-h]
```

upgrade

Upgrade the metadata database to latest version

```
airflow db upgrade [-h]
```

info

Show information about current Airflow and environment

```
airflow info [-h] [--anonymize] [--file-io]
```

Named Arguments

`--anonymize`

Minimize any personal identifiable information. Use it when sharing output with others.

Default: False

`--file-io`

Send output to file.io service and returns link.

Default: False

kerberos

Start a kerberos ticket renewer

```
airflow kerberos [-h] [-D] [-k [KEYTAB]] [-l LOG_FILE] [--pid [PID]]
                 [--stderr STDERR] [--stdout STDOUT]
                 [principal]
```

Positional Arguments

`principal`

kerberos principal

Named Arguments

`-D, --daemon`

Daemonize instead of running in the foreground

Default: False

`-k, --keytab`

keytab

Default: “airflow.keytab”

`-l, --log-file`

Location of the log file

`--pid`

PID file location

`--stderr`

Redirect stderr to this file

`--stdout`

Redirect stdout to this file

kubernetes

Tools to help run the KubernetesExecutor

```
airflow kubernetes [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: cleanup-pods, generate-dag-yaml

Sub-commands:

cleanup-pods

Clean up Kubernetes pods in evicted/failed/succeeded states

```
airflow kubernetes cleanup-pods [-h] [--namespace NAMESPACE]
```

Named Arguments¶

`--namespace`

Kubernetes Namespace

Default: “default”

generate-dag-yaml

Generate YAML files for all tasks in DAG. Useful for debugging tasks without launching into a cluster

```
airflow kubernetes generate-dag-yaml [-h] [-o OUTPUT_PATH] [-S SUBDIR]
                                     dag_id execution_date
```

Positional Arguments¶

`dag_id`

The id of the dag

`execution_date`

The execution date of the DAG

Named Arguments¶

`-o, --output-path`

The output for generated yaml files

Default: “[CWD]”

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to ‘[AIRFLOW_HOME]/dags’ where [AIRFLOW_HOME] is the value you set for ‘AIRFLOW_HOME’ config you set in ‘airflow.cfg’

Default: “[AIRFLOW_HOME]/dags”

plugins

Dump information about loaded plugins

```
airflow plugins [-h] [-o table, json, yaml]
```

Named Arguments

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

pools

Manage pools

```
airflow pools [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: delete, export, get, import, list, set

Sub-commands:

delete

Delete pool

```
airflow pools delete [-h] [-o table, json, yaml] NAME
```

Positional Arguments¶

`NAME`

Pool name

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

export

Export all pools

```
airflow pools export [-h] FILEPATH
```

Positional Arguments¶

`FILEPATH`

Export all pools to JSON file

get

Get pool size

```
airflow pools get [-h] [-o table, json, yaml] NAME
```

Positional Arguments¶

`NAME`

Pool name

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

import

Import pools

```
airflow pools import [-h] FILEPATH
```

Positional Arguments¶

`FILEPATH`

Import pools from JSON file

list

List pools

```
airflow pools list [-h] [-o table, json, yaml]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

set

Configure pool

```
airflow pools set [-h] [-o table, json, yaml] NAME slots description
```

Positional Arguments¶

`NAME`

Pool name

`slots`

Pool slots

`description`

Pool description

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

providers

Display providers

```
airflow providers [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: behaviours, get, hooks, links, list, widgets

Sub-commands:

behaviours

Get information about registered connection types with custom behaviours

```
airflow providers behaviours [-h] [-o table, json, yaml]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

get

Get detailed information about a provider

```
airflow providers get [-h] [--color {auto,off,on}] [-f] [-o table, json, yaml]
                      provider_name
```

Positional Arguments¶

`provider_name`

Provider name, required to get provider information

Named Arguments¶

`--color`

Possible choices: auto, off, on

Do emit colored output (default: auto)

Default: “auto”

`-f, --full`

Full information about the provider, including documentation information.

Default: False

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

hooks

List registered provider hooks

```
airflow providers hooks [-h] [-o table, json, yaml]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

links

List extra links registered by the providers

```
airflow providers links [-h] [-o table, json, yaml]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

list

List installed providers

```
airflow providers list [-h] [-o table, json, yaml]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

widgets

Get information about registered connection form widgets

```
airflow providers widgets [-h] [-o table, json, yaml]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

roles

Manage roles

```
airflow roles [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: create, list

Sub-commands:

create

Create role

```
airflow roles create [-h] [role [role ...]]
```

Positional Arguments¶

`role`

The name of a role

list

List roles

```
airflow roles list [-h] [-o table, json, yaml]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

rotate-fernet-key

Rotate all encrypted connection credentials and variables; see https://airflow.apache.org/docs/stable/howto/secure-connections.html#rotating-encryption-keys

```
airflow rotate-fernet-key [-h]
```

scheduler

Start a scheduler instance

```
airflow scheduler [-h] [-D] [-p] [-l LOG_FILE] [-n NUM_RUNS] [--pid [PID]]
                  [--stderr STDERR] [--stdout STDOUT] [-S SUBDIR]
```

Named Arguments

`-D, --daemon`

Daemonize instead of running in the foreground

Default: False

`-p, --do-pickle`

Attempt to pickle the DAG object to send over to the workers, instead of letting workers run their version of the code

Default: False

`-l, --log-file`

Location of the log file

`-n, --num-runs`

Set the number of runs to execute before exiting

Default: -1

`--pid`

PID file location

`--stderr`

Redirect stderr to this file

`--stdout`

Redirect stdout to this file

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to ‘[AIRFLOW_HOME]/dags’ where [AIRFLOW_HOME] is the value you set for ‘AIRFLOW_HOME’ config you set in ‘airflow.cfg’

Default: “[AIRFLOW_HOME]/dags”

Signals:

- SIGUSR2: Dump a snapshot of task state being tracked by the executor.

  `Example:`

pkill -f -USR2 “airflow scheduler”

sync-perm

Update permissions for existing roles and DAGs

```
airflow sync-perm [-h]
```

tasks

Manage tasks

```
airflow tasks [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: clear, failed-deps, list, render, run, state, states-for-dag-run, test

Sub-commands:

clear

Clear a set of task instance, as if they never ran

```
airflow tasks clear [-h] [-R] [-d] [-e END_DATE] [-X] [-x] [-f] [-r]
                    [-s START_DATE] [-S SUBDIR] [-t TASK_REGEX] [-u] [-y]
                    dag_id
```

Positional Arguments¶

`dag_id`

The id of the dag

Named Arguments¶

`-R, --dag-regex`

Search dag_id as regex instead of exact string

Default: False

`-d, --downstream`

Include downstream tasks

Default: False

`-e, --end-date`

Override end_date YYYY-MM-DD

`-X, --exclude-parentdag`

Exclude ParentDAGS if the task cleared is a part of a SubDAG

Default: False

`-x, --exclude-subdags`

Exclude subdags

Default: False

`-f, --only-failed`

Only failed jobs

Default: False

`-r, --only-running`

Only running jobs

Default: False

`-s, --start-date`

Override start_date YYYY-MM-DD

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

`-t, --task-regex`

The regex to filter specific task_ids to backfill (optional)

`-u, --upstream`

Include upstream tasks

Default: False

`-y, --yes`

Do not prompt to confirm reset. Use with care!

Default: False

failed-deps

Returns the unmet dependencies for a task instance from the perspective of the scheduler. In other words, why a task instance doesn't get scheduled and then queued by the scheduler, and then run by an executor.

```
airflow tasks failed-deps [-h] [-S SUBDIR] dag_id task_id execution_date
```

Positional Arguments¶

`dag_id`

The id of the dag

`task_id`

The id of the task

`execution_date`

The execution date of the DAG

Named Arguments¶

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

list

List the tasks within a DAG

```
airflow tasks list [-h] [-S SUBDIR] [-t] dag_id
```

Positional Arguments¶

`dag_id`

The id of the dag

Named Arguments¶

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

`-t, --tree`

Tree view

Default: False

render

Render a task instance's template(s)

```
airflow tasks render [-h] [-S SUBDIR] dag_id task_id execution_date
```

Positional Arguments¶

`dag_id`

The id of the dag

`task_id`

The id of the task

`execution_date`

The execution date of the DAG

Named Arguments¶

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

run

Run a single task instance

```
airflow tasks run [-h] [--cfg-path CFG_PATH] [--error-file ERROR_FILE] [-f]
                  [-A] [-i] [-I] [-N] [-l] [-m] [-p PICKLE] [--pool POOL]
                  [--ship-dag] [-S SUBDIR]
                  dag_id task_id execution_date
```

Positional Arguments¶

`dag_id`

The id of the dag

`task_id`

The id of the task

`execution_date`

The execution date of the DAG

Named Arguments¶

`--cfg-path`

Path to config file to use instead of airflow.cfg

`--error-file`

File to store task failure error

`-f, --force`

Ignore previous task instance state, rerun regardless if task already succeeded/failed

Default: False

`-A, --ignore-all-dependencies`

Ignores all non-critical dependencies, including ignore_ti_state and ignore_task_deps

Default: False

`-i, --ignore-dependencies`

Ignore task-specific dependencies, e.g. upstream, depends_on_past, and retry delay dependencies

Default: False

`-I, --ignore-depends-on-past`

Ignore depends_on_past dependencies (but respect upstream dependencies)

Default: False

`-N, --interactive`

Do not capture standard output and error streams (useful for interactive debugging)

Default: False

`-l, --local`

Run the task using the LocalExecutor

Default: False

`-m, --mark-success`

Mark jobs as succeeded without running them

Default: False

`-p, --pickle`

Serialized pickle object of the entire dag (used internally)

`--pool`

Resource pool to use

`--ship-dag`

Pickles (serializes) the DAG and ships it to the worker

Default: False

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

state

Get the status of a task instance

```
airflow tasks state [-h] [-S SUBDIR] dag_id task_id execution_date
```

Positional Arguments¶

`dag_id`

The id of the dag

`task_id`

The id of the task

`execution_date`

The execution date of the DAG

Named Arguments¶

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

states-for-dag-run

Get the status of all task instances in a dag run

```
airflow tasks states-for-dag-run [-h] [-o table, json, yaml]
                                 dag_id execution_date
```

Positional Arguments¶

`dag_id`

The id of the dag

`execution_date`

The execution date of the DAG

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: "table"

test

Test a task instance. This will run a task without checking for dependencies or recording its state in the database

```
airflow tasks test [-h] [-n] [--env-vars ENV_VARS] [-m] [-S SUBDIR]
                   [-t TASK_PARAMS]
                   dag_id task_id execution_date
```

Positional Arguments¶

`dag_id`

The id of the dag

`task_id`

The id of the task

`execution_date`

The execution date of the DAG

Named Arguments¶

`-n, --dry-run`

Perform a dry run for each task. Only renders Template Fields for each task, nothing else

Default: False

`--env-vars`

Set env var in both parsing time and runtime for each of entry supplied in a JSON dict

`-m, --post-mortem`

Open debugger on uncaught exception

Default: False

`-S, --subdir`

File location or directory from which to look for the dag. Defaults to '[AIRFLOW_HOME]/dags' where [AIRFLOW_HOME] is the value you set for 'AIRFLOW_HOME' config you set in 'airflow.cfg'

Default: "[AIRFLOW_HOME]/dags"

`-t, --task-params`

Sends a JSON params dict to the task

users

Manage users

```
airflow users [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: add-role, create, delete, export, import, list, remove-role

Sub-commands:

add-role

Add role to a user

```
airflow users add-role [-h] [-e EMAIL] -r ROLE [-u USERNAME]
```

Named Arguments¶

`-e, --email`

Email of the user

`-r, --role`

Role of the user. Existing roles include Admin, User, Op, Viewer, and Public

`-u, --username`

Username of the user

create

Create a user

```
airflow users create [-h] -e EMAIL -f FIRSTNAME -l LASTNAME [-p PASSWORD] -r
                     ROLE [--use-random-password] -u USERNAME
```

Named Arguments¶

`-e, --email`
: Email of the user

`-f, --firstname`
: First name of the user

`-l, --lastname`
: Last name of the user

`-p, --password`
: Password of the user, required to create a user without –use-random-password

`-r, --role`
: Role of the user. Existing roles include Admin, User, Op, Viewer, and Public

`--use-random-password`
: Do not prompt for password. Use random string instead. Required to create a user without –password

  Default: False

`-u, --username`
: Username of the user

examples: To create an user with “Admin” role and username equals to “admin”, run:

> `$ airflow users create`
>
> –username admin –firstname FIRST_NAME –lastname LAST_NAME –role Admin –email admin@example.org

delete

Delete a user

```
airflow users delete [-h] -u USERNAME
```

Named Arguments¶

`-u, --username`
: Username of the user

export

Export all users

```
airflow users export [-h] FILEPATH
```

Positional Arguments¶

`FILEPATH`
: Export all users to JSON file

import

Import users

```
airflow users import [-h] FILEPATH
```

Positional Arguments¶

`FILEPATH`

Import users from JSON file. Example format:

```
[
    {
        "email": "foo@bar.org",
        "firstname": "Jon",
        "lastname": "Doe",
        "roles": ["Public"],
        "username": "jondoe"
    }
]
```

list

List users

```
airflow users list [-h] [-o table, json, yaml]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

remove-role

Remove role from a user

```
airflow users remove-role [-h] [-e EMAIL] -r ROLE [-u USERNAME]
```

Named Arguments¶

`-e, --email`

Email of the user

`-r, --role`

Role of the user. Existing roles include Admin, User, Op, Viewer, and Public

`-u, --username`

Username of the user

variables

Manage variables

```
airflow variables [-h] COMMAND ...
```

Positional Arguments

`COMMAND`

Possible choices: delete, export, get, import, list, set

Sub-commands:

delete

Delete variable

```
airflow variables delete [-h] key
```

Positional Arguments¶

`key`

Variable key

export

Export all variables

```
airflow variables export [-h] file
```

Positional Arguments¶

`file`

Export all variables to JSON file

get

Get variable

```
airflow variables get [-h] [-d VAL] [-j] key
```

Positional Arguments¶

`key`

Variable key

Named Arguments¶

`-d, --default`

Default value returned if variable does not exist

`-j, --json`

Deserialize JSON variable

Default: False

import

Import variables

```
airflow variables import [-h] file
```

Positional Arguments¶

`file`

Import variables from JSON file

list

List variables

```
airflow variables list [-h] [-o table, json, yaml]
```

Named Arguments¶

`-o, --output`

Possible choices: table, json, yaml

Output format. Allowed values: json, yaml, table (default: table)

Default: “table”

set

Set variable

```
airflow variables set [-h] [-j] key VALUE
```

Positional Arguments¶

`key`

Variable key

`VALUE`

Variable value

Named Arguments¶

`-j, --json`

Deserialize JSON variable

Default: False

version

Show the version

```
airflow version [-h]
```

webserver

Start a Airflow webserver instance

```
airflow webserver [-h] [-A ACCESS_LOGFILE] [-L ACCESS_LOGFORMAT] [-D] [-d]
                  [-E ERROR_LOGFILE] [-H HOSTNAME] [-l LOG_FILE] [--pid [PID]]
                  [-p PORT] [--ssl-cert SSL_CERT] [--ssl-key SSL_KEY]
                  [--stderr STDERR] [--stdout STDOUT] [-t WORKER_TIMEOUT]
                  [-k {sync,eventlet,gevent,tornado}] [-w WORKERS]
```

Named Arguments

`-A, --access-logfile`

The logfile to store the webserver access log. Use ‘-‘ to print to stderr

Default: “-“

`-L, --access-logformat`

The access log format for gunicorn logs

Default: “”

`-D, --daemon`

Daemonize instead of running in the foreground

Default: False

`-d, --debug`

Use the server that ships with Flask in debug mode

Default: False

`-E, --error-logfile`

The logfile to store the webserver error log. Use ‘-‘ to print to stderr

Default: “-“

`-H, --hostname`

Set the hostname on which to run the web server

Default: “0.0.0.0”

`-l, --log-file`

Location of the log file

`--pid`

PID file location

`-p, --port`

The port on which to run the server

Default: 8080

`--ssl-cert`

Path to the SSL certificate for the webserver

Default: “”

`--ssl-key`

Path to the key to use with the SSL certificate

Default: “”

`--stderr`

Redirect stderr to this file

`--stdout`

Redirect stdout to this file

`-t, --worker-timeout`

The timeout for waiting on webserver workers

Default: 120

`-k, --workerclass`

Possible choices: sync, eventlet, gevent, tornado

The worker class to use for Gunicorn

Default: “sync”

`-w, --workers`

Number of workers to run the webserver on

Default: 4

Environment Variables

`AIRFLOW__{SECTION}__{KEY}`

Sets options in the Airflow configuration. This takes priority over the value in the `airflow.cfg` file.

Replace the `{SECTION}` placeholder with any section and the `{KEY}` placeholder with any key in that specified section.

For example, if you want to set the `dags_folder` options in `[core]` section, then you should set the `AIRFLOW__CORE__DAGS_FOLDER` environment variable.

For more information, see: Setting Configuration Options.

`AIRFLOW__{SECTION}__{KEY}_CMD`

For any specific key in a section in Airflow, execute the command the key is pointing to. The result of the command is used as a value of the `AIRFLOW__{SECTION}__{KEY}` environment variable.

This is only supported by the following config options:

- `sql_alchemy_conn` in `[core]` section
- `fernet_key` in `[core]` section
- `broker_url` in `[celery]` section
- `flower_basic_auth` in `[celery]` section
- `result_backend` in `[celery]` section
- `password` in `[atlas]` section
- `smtp_password` in `[smtp]` section
- `secret_key` in `[webserver]` section

`AIRFLOW__{SECTION}__{KEY}_SECRET`

For any specific key in a section in Airflow, retrieve the secret from the configured secrets backend. The returned value will be used as the value of the `AIRFLOW__{SECTION}__{KEY}` environment variable.

See Secrets Backends for more information on available secrets backends.

This form of environment variable configuration is only supported for the same subset of config options as `AIRFLOW__{SECTION}__{KEY}_CMD`

`AIRFLOW_CONFIG`

The path to the Airflow configuration file.

`AIRFLOW_CONN_{CONN_ID}`

Defines a new connection with the name `{CONN_ID}` using the URI value.

For example, if you want to create a connection named `PROXY_POSTGRES_TCP`, you can create a key `AIRFLOW_CONN_PROXY_POSTGRES_TCP` with the connection URI as the value.

For more information, see: Storing a Connection in Environment Variables.

`AIRFLOW_HOME`

The root directory for the Airflow content. This is the default parent directory for Airflow assets such as DAGs and logs.

`AIRFLOW_VAR_{KEY}`

Defines an Airflow variable. Replace the `{KEY}` placeholder with the variable name.

For more information, see: Managing Variables.

Previous Next

Was this entry helpful?

☆ ☆ ☆ ☆ ☆

Want to be a part of Apache Airflow? Join community

License Donate Thanks
Security