Install Airflow:

- pip install apache-airflow
- To install extra dependencies
  `pip install 'apache-airflow[gcp]'` For google cloud support.
  For other packages
  (https://airflow.apache.org/docs/stable/installation.html)
- airflow version(to check installation)

Create server:

- First create a `airflow_home` folder inside your project.
- Then set airflowPath for your current project:
  `export AIRFLOW_HOME=$(pwd)/airflow_home.
- Then initiate airflow :
  `airflow initdb`
  This will create airflow configuration, database, logs and unittest files.
- Then create `dags` folder inside airflow_home folder. Inside dags folder we will store our dags.
- (optional) if you want to remove example dags then change `load_examples=False` inside airflow.cfg file.
- To start server run this command:
  `airflow webserver`
  `airflow scheduler`
  Shcheduler is used to run dags and update dags list.

  Note:if u face any database exception then run airflow initdb command again

Parts of airflow code:

- **DAGs** do not perform any actual computation. Instead, **Operators** determine what actually gets done.

  (Dag is directed acyclic graph, i.e it always points to a single direction.)

  (different operators: https://github.com/apache/airflow/tree/master/airflow/contrib/operators)

- **Task:** Once an operator is instantiated, it is referred to as a “task”. An operator describes a single task in a workflow.
  - Instantiating a task requires providing a unique task_id and DAG container
- A **DAG** is a container that is used to organize tasks and set their execution context

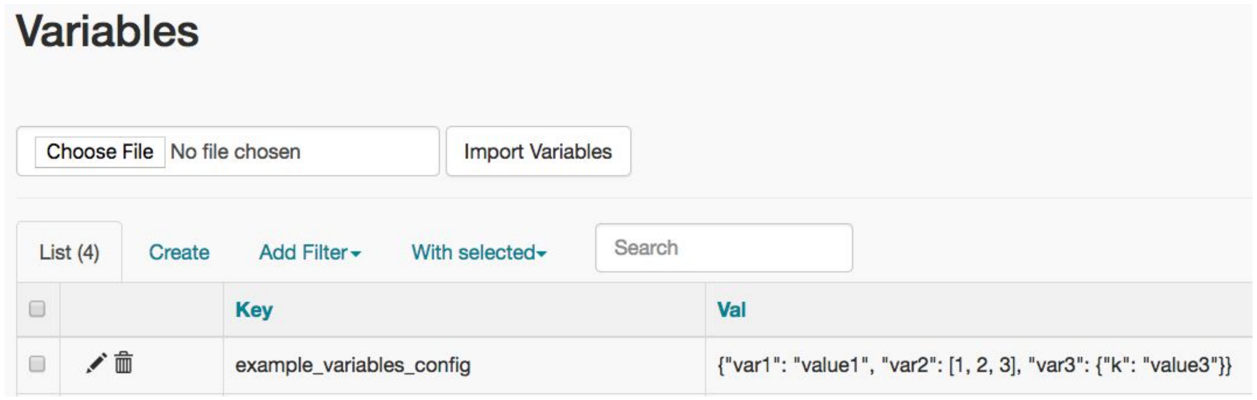
```
t1 = BashOperator(
    task_id='print_date',
    bash_command='date',
    dag=dag,
)
```

ADDING CONNECTION TO OTHER SERVICES:

- SETTING UP CONNECTION WITH GOOGLe apis:-> https://cloud.google.com/composer/docs/how-to/managing/connections
  Now we can use the above-created connection_id inside our google operator in our airflow code.

Working with Variables

- Variables can be listed, created, updated and deleted from the UI (`Admin -> Variables`).
- In addition, json settings files can be bulk uploaded through the UI. Please look at an example here for a variable json setting file


Variables

		Key	Val
		example_variables_config	{"var1": "value1", "var2": [1, 2, 3], "var3": {"k": "value3"}}

Note: best practice is to use a single json for a given project (i.e use single airflow variable for all variables in a single project. Like we did above by uploading json for single variable.)

Fetching vars:

```
## Recommended way

dag_config = Variable.get("example_variables_config",
deserialize_json=True)

var1 = dag_config["var1"]

var2 = dag_config["var2"]

var3 = dag_config["var3"]
```