**Apache Airflow with Docker**

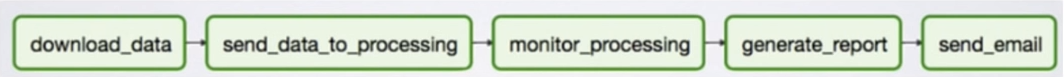
**What is Airflow?**

Airflow is a platform to programmatically author, schedule, and monitor workflows or data pipelines.

**What is a Workflow?**

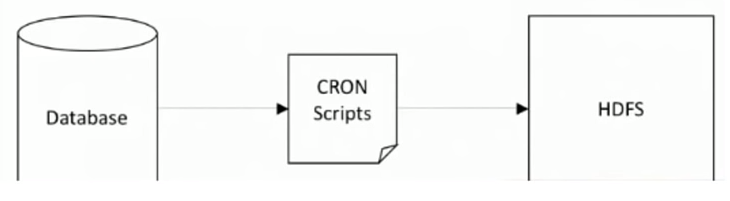
* a sequence of tasks
* started on a scheduler or triggered by an event
* frequently used to handle big data processing pipelines

**A typical Workflow be like**

****

1. download data from source
2. send data somewhere else to process
3. monitor when the process is completed
4. get the result and generate the report
5. send the report out by email

**A traditional ETL approach**:



Examples of naive approach:

* Writing a script to pull data from database and sent it to HDFS to process.
* Schedule the script as a cronjob.

**Problems with naïve approach:**

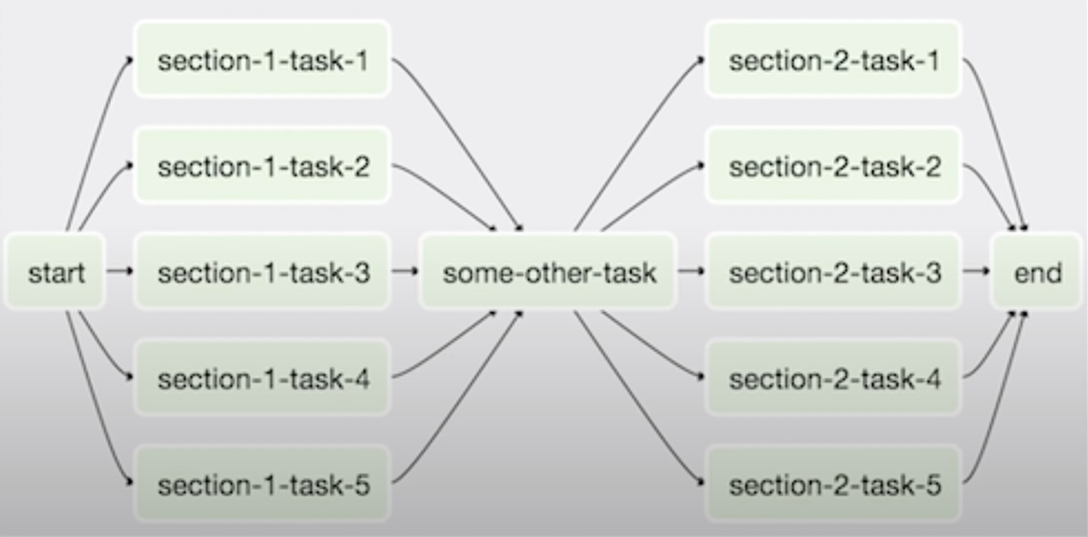
* **Failures**:
  + Retry if failure happens(how many times? how often?)
* **Monitoring**:
  + Success or failure status, how long does the process runs?
* **Dependencies**:
  + Data dependencies: upstream data is missing.
  + Execution dependencies: job 2 runs after job 1 is finished.
* **Scalability**:
  + There is no centralized scheduler between different cron machines.
* **Deployment**:
  + Deploy new changes constantly.
* **Process historic data**:
  + Backfill/rerun historical data.

**Apache Airflow:**

* The project joined the Apache Software Foundation’s incubation program in 2016.
* A workflow (data-pipeline) management system developed by Airbnb
  + A framework to define task and dependencies in python
  + Executing, scheduling and distributing tasks across worker nodes.
  + View of present and past runs, logging feature.
  + Extensible through plugins.
  + Nice UI, possibility to define REST interface.
  + Interact well with database.
* Used by more than 200 companies: Airbnb, Yahoo, Paypal, Intel, Stripe

**Airflow DAG:**

* A workflow as a Directed Acyclic Graph (DAG) with multiple tasks which can be executed independently.
* Airflow DAGs are composed of Tasks.

****

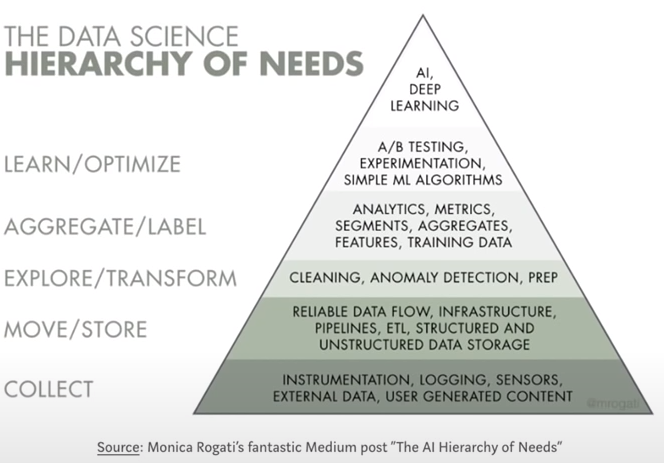
**What makes Airflow great?**

* Can handle upstream/downstream dependencies gracefully (Eg: upstream missing tables)
* Easy to reprocess historical jobs by date, or re-run for specific intervals.
* Jobs can pass parameters to other jobs downstream.
* Handle error and failures gracefully. Automatically retry when a task fails.
* Ease of deployment of workflow changes (continuous integration)
* Integrations with a lot of infrastructure (Hive, Presto, Druid, AWS, Google cloud etc..)
* Data sensors to trigger a DAG when data arrives.
* Job testing through Airflow itself.
* Accessibility of log files and other meta-data through the web GUI.
* Implement trigger rules for tasks.
* Monitoring all jobs status in real time + Email alerts.
* Community support.

**Airflow applications:**

* **Data Warehousing:**
  + Cleanse, organize, data quality check, and publish/stream data.
* **Machine Learning:** 
  + Automate machine learning workflows.
* **Growth Analytics:** 
  + Compute metrics around guest and host engagement as well as growth accounting.
* **Experimentation:** 
  + Compute A/B testing experimentation frameworks logic and aggregates.
* **Email Targeting:** 
  + Apply rules to target and engage users through email campaigns.
* **Sessionization:** 
  + Compute clickstream and time spent datasets.
* **Search:** 
  + Compute search ranking related metrics.
* **Data Infrastructure Maintenance:** 
  + Database scrapes, folder cleanup, applying data retention policies etc.

**The Hierarchy of Data Science:**

****

This framework puts things into perspective. Before a company can optimize the business more efficiently or build data products more intelligently, layers of foundational work need to be built first. Data is the fuel for all data products.

Unfortunately, most data science training program right now only focus on the top of the pyramid of knowledge

**Problems we face with Airflow:**

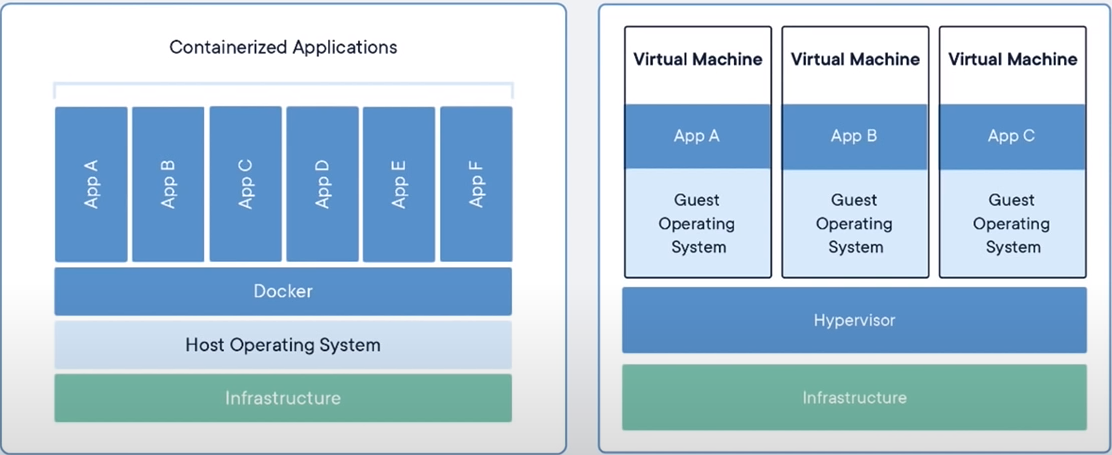
* Airflow is built to integrate with all databases, system, cloud environments,
  + Managing and maintaining all of the dependencies changes will be really difficult.
  + Takes lots of time to setup and configure Airflow env.
  + How to share development and production environments for all developers.

Docker helps to overcome these issues.

Docker is an open platform for developing, shipping and running applications.

Docker provides the ability to package and run an application in a loosely isolated environment called containers simultaneously on a given host, regardless of its operating system: MAC, Windows, PC, cloud, data center etc.

Containers are lightweight because they don’t need the extra load of a hypervisor but run directly within the host machine’s kernel. This means you can run more containers on a given hardware combination than if you were using virtual machines.



**Benefits of using Docker:**

* Docker is freeing us from the task of managing, maintaining all the Airflow dependencies and deployment.
* Easy to share and deploy different versions and environments.
* Keep track through GitHub tags and releases.
* Ease of deployment from testing to production environment.

**Setup Docker:**

Step1: Create an account on [docker hub](https://hub.docker.com/)

Step2: Install [Docker Desktop](https://www.docker.com/products/docker-desktop)

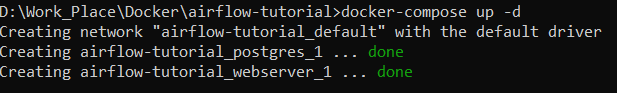
Step3: There will be a small tutorial when starting docker, so follow it up for better understanding.

**Airflow tutorial example:**

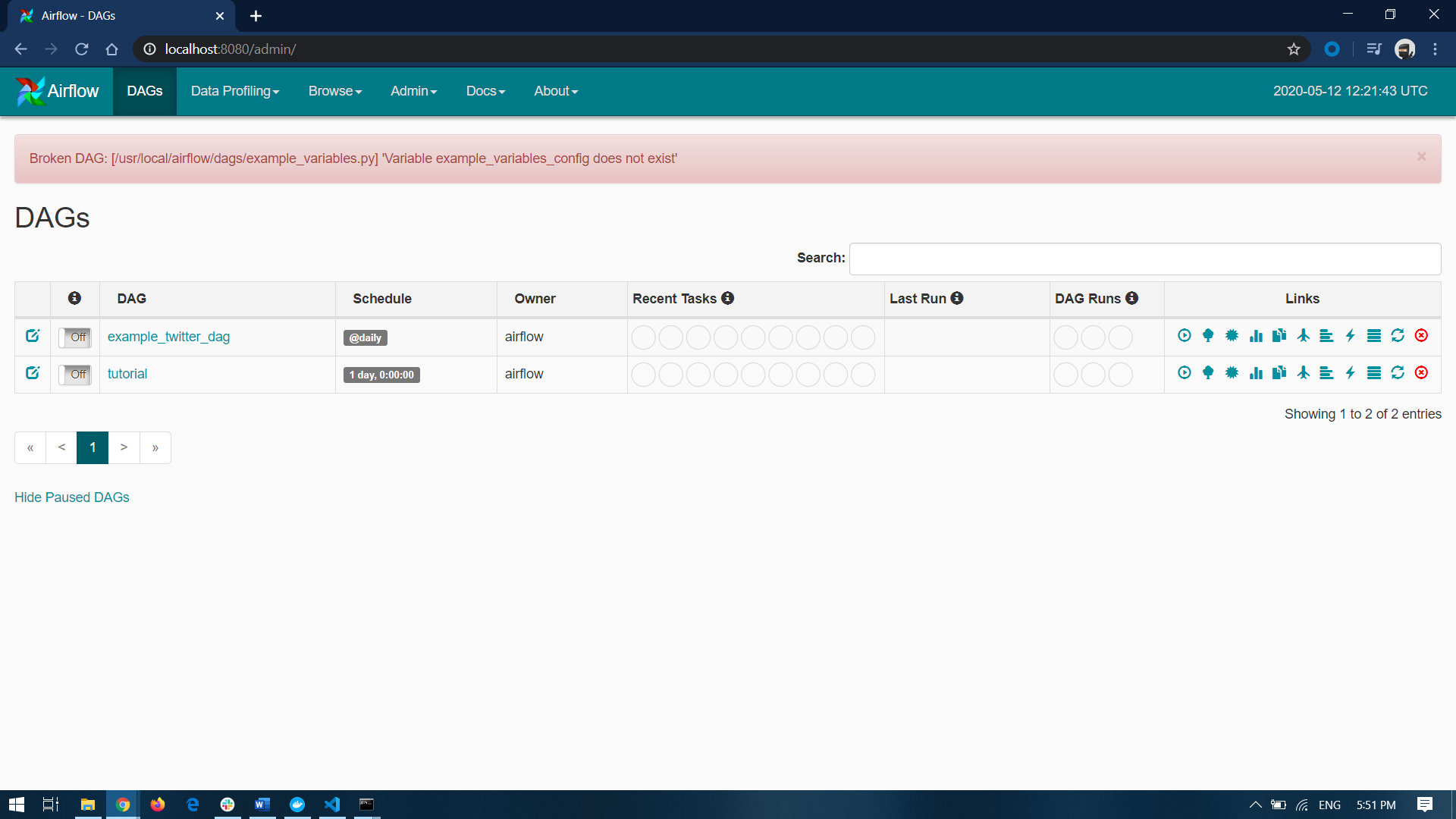
Step1: Clone this [repository](https://github.com/tuanavu/airflow-tutorial.git) in your local by using :

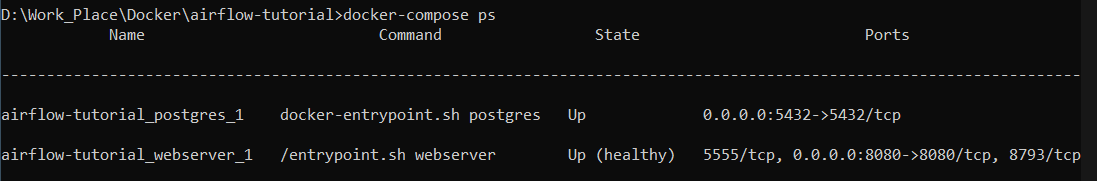
*git clone* <https://github.com/tuanavu/airflow-tutorial.git>

Step2: To start the services, use this command : *docker-compose up -d*

**

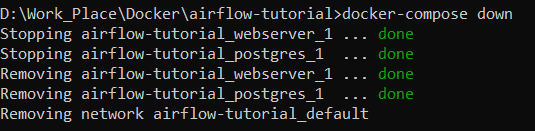
Step3: To land on the Airflow webserver: [*http://localhost:8080/*](http://localhost:8080/)



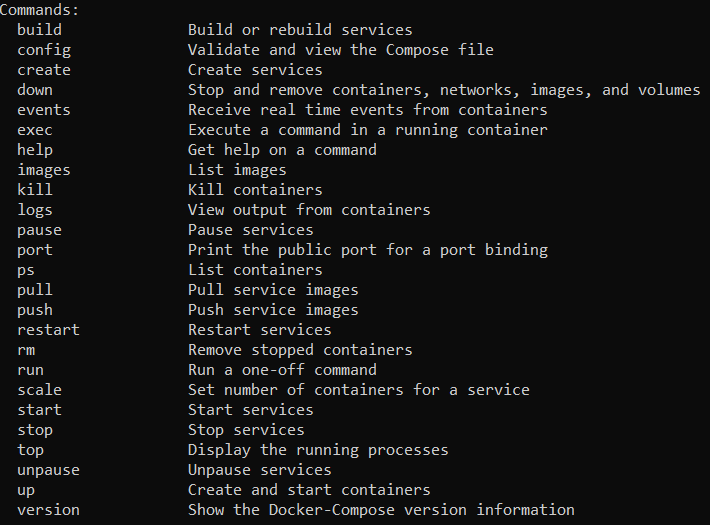
You can also check the list of containers that are running using: *docker-compose ps*

You can also view logs using : *docker-compose logs*

Step3: To stop the services, use this command: *docker-compose down*

**

Here is a list of commands that are helpful



You can also view the containers in Docker Desktop

