Installing and Configuring Apache Airflow

Posted on December 1st, 2016 by Robert Sanders

|

Apache Airflow is a platform to programmatically author, schedule and monitor workflows – it supports integration with 3rd party platforms so that you, our developer and user community, can adapt it to your needs and stack.

**Additional Documentation:**

Documentation: <https://airflow.incubator.apache.org/>

Install Documentation: <https://airflow.incubator.apache.org/installation.html>

GitHub Repo: <https://github.com/apache/incubator-airflow>

Preparing the Environment

Install all needed system dependencies

Ubuntu

1. SSH onto target machine (s) where you want to install Airflow
2. Login as Root
3. Install Required Libraries

|  |
| --- |
| #Run upgrade  apt-get update    #Unzip  apt-get install unzip    #Build Essentials - GCC Compiler  apt-get install build-essential    #Python Development  apt-get install python-dev    #SASL  apt-get install libsasl2-dev    #Pandas  apt-get install python-pandas |

1. Check Python Version
   1. Run the command:

|  |
| --- |
| python -V |

* 1. If the version comes back as “Python 2.7.X” you can skip the rest of this step
  2. Install Python 2.7.X

|  |
| --- |
| cd /opt  sudo wget --no-check-certificate https://www.python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz  tar xf Python-2.7.6.tar.xz  cd Python-2.7.6  ./configure --prefix=/usr/local  make && make altinstall  ls -ltr /usr/local/bin/python\*  vi ~/.bashrc  #add this line alias python='/usr/local/bin/python2.7'  source ~/.bashrc |

1. Install PIP
   1. Run Install

|  |
| --- |
| cd /tmp/    wget https://bootstrap.pypa.io/ez\_setup.py    python ez\_setup.py  unzip setuptools-\*.zip  cd setuptools-\*  easy\_install pip |

* 1. Verify Installation

|  |
| --- |
| which pip    # Should print out the path to the pip command |

* 1. If you come across an issue where while using pip bellow, its still referring to python2.6, you can follow these instructions
     1. Replace the binaries in the /usr/bin/ directory with the ones that were just installed

|  |
| --- |
| cd /usr/bin/  #Backup old binaries  mv pip pip-BACKUP  mv pip2 pip2-BACKUP  mv pip2.6 pip2.6-BACKUP  #Setup symlinks to the new version of pip that was installed  ln -s /usr/local/bin/pip pip  ln -s /usr/local/bin/pip2 pip2  ln -s /usr/local/bin/pip2.7 pip2.7 |

**Troubleshooting installation on Ubuntu**:

* If you later get the error “**error trying to exec ‘as’: execvp: No such file or directory**” while trying to install airflow with PIP
  + Install the following:

|  |
| --- |
| apt-get install binutils  apt-get install gcc  apt-get install build-essential  pip install pandas |

* + Retry installation
  + If the problem persists, uninstall the packages listed above and reinstall. Then rerun.

CentOS

1. SSH onto target machine(s) where you want to install Airflow
2. Login as Root
3. Install Required Libraries

|  |
| --- |
| yum groupinstall "Development tools"    yum install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel python-devel wget cyrus-sasl-devel.x86\_64 |

1. Check Python Version
   1. Run the command:

|  |
| --- |
| python -V |

* 1. If the version comes back as “Python 2.7.X” you can skip the rest of this step
  2. Install Python 2.7.X

|  |
| --- |
| cd /opt  sudo wget --no-check-certificate https://www.python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz  tar xf Python-2.7.6.tar.xz  cd Python-2.7.6  ./configure --prefix=/usr/local  make && make altinstall  ls -ltr /usr/local/bin/python\*  vi ~/.bashrc  #add this line alias python='/usr/local/bin/python2.7'  source ~/.bashrc |

1. Install PIP
   1. Run Install

|  |
| --- |
| cd /tmp/  wget https://bootstrap.pypa.io/ez\_setup.py  python ez\_setup.py  unzip setuptools-X.X.zip  cd setuptools-X.X  easy\_install pip |

* 1. Verify Installation

|  |
| --- |
| which pip    #Should print out "/usr/local/bin/pip" |

**Troubleshooting on CentOS**:

* If you get an error saying **ImportError: No module named extern**while Installing PIP with easy\_install
  1. Reinstall python-setuptools:

|  |
| --- |
| yum reinstall python-setuptools |

* 1. Retry installation

Install Airflow

Login as Root and run:

|  |
| --- |
| pip install airflow==1.7.0  pip install airflow[hive]==1.7.0  pip install airflow[celery]==1.7.0 |

Update: Common Issue with Celery

Recently there were some updates to the dependencies of Airflow where if you were to install the airflow[celery] dependency for Airflow 1.7.x, pip would install celery version 4.0.2. This version of celery is incompatible with Airflow 1.7.x. This would result in various types of errors including messages saying that the CeleryExecutor can’t be loaded or that tasks are not getting executed as they should.

To get around this issue, install an older version of celery using pip:

|  |
| --- |
| pip install celery==3.1.17 |

Install RabbitMQ

If you intend to use RabbitMQ as a message broker you will need to install RabbitMQ.If you don’t intend to, you can skip this step. For production it is recommended that you use **CeleryExecutors** which requires a message broker such as RabbitMQ.

Setup

Follow these steps: [Install RabbitMQ](http://site.clairvoyantsoft.com/installing-rabbitmq/)

Recovering from a RabbitMQ Node Failure

If you’ve opted to setup RabbitMQ to run on as a cluster, and one of those cluster nodes fails, you can follow these steps to recover on airflow:

1. Bring the RabbitMQ node and daemon back up
2. Navigate to the RabbitMQ Management UI
3. Click on Queues
4. Delete the “Default” queue
5. Restart Airflow Scheduler service

Install MySQL Dependencies

If you intend to use MySQL as an DB repo you will need to install some MySQL dependencies. If you don’t intend to, you can skip this step.

Install MySQL Dependencies on Ubuntu

1. Install MySQL Dependencies

|  |
| --- |
| apt-get install python-dev libmysqlclient-dev  pip install MySQL-python |

Install MySQL Dependencies on CentOS

1. Install MySQL Dependencies

|  |
| --- |
| yum install -y mysql-devel python-devel python-setuptools  pip install MySQL-python |

Configuring Airflow

Its recommended to use RabbitMQ.

Apache Airflow needs a home, ~/airflow is the default, but you can lay foundation somewhere else if you prefer **(OPTIONAL)**

|  |
| --- |
| export AIRFLOW\_HOME=~/airflow |

Run the following as the desired user (who ever you want executing the Airflow jobs) to setup the airflow directories and default configs

|  |
| --- |
| airflow initdb    #note: When you run this the first time, it will generate a sqlite file (airflow.db) in the AIRFLOW\_HOME directory for the Airflow Metastore. If you don't intend to use sqlite as the Metastore then you can remove this file. |

Make the following changes to the {AIRFLOW\_HOME}/airflow.cfg file

1. Change the Executor to CeleryExecutor (Recommended for production)

|  |
| --- |
| executor = CeleryExecutor |

1. Point SQL Alchemy to MySQL (if using MySQL)

|  |
| --- |
| sql\_alchemy\_conn = mysql://{USERNAME}:{PASSWORD}@{MYSQL\_HOST}:3306/airflow |

1. Set dags are paused on startup. This is a good idea to avoid unwanted runs of the workflow. (Recommended)

|  |
| --- |
| # Are DAGs paused by default at creation  dags\_are\_paused\_at\_creation = True |

1. Don’t load examples

|  |
| --- |
| load\_examples = False |

1. Set the Broker URL (If you’re using CeleryExecutors)
   1. If you’re using RabbitMQ:

|  |
| --- |
| broker\_url = amqp://guest:guest@{RABBITMQ\_HOST}:5672/ |

* 1. If you’re using AWS SQS:

|  |
| --- |
| broker\_url = sqs://{ACCESS\_KEY\_ID}:{SECRET\_KEY}@    # Note: You will also need to install boto:  $ pip install -U boto |

1. Point Celery to MySQL (if using MySQL)

|  |
| --- |
| celery\_result\_backend = db+mysql://{USERNAME}:{PASSWORD}@{MYSQL\_HOST}:3306/airflow |

1. Set the default\_queue name used by CeleryExecutors (Optional: Primarily for if you have a preference of the default queue name or plan on using the same broker for multiple airflow instances)

|  |
| --- |
| # Default queue that tasks get assigned to and that worker listen on.  default\_queue = {YOUR\_QUEUE\_NAME\_HERE} |

1. Setup MySQL (if using MySQL)
   1. Login to the mysql machine
   2. Create the airflow database if it doesn’t exist

|  |
| --- |
| CREATE DATABASE airflow CHARACTER SET utf8 COLLATE utf8\_unicode\_ci; |

* 1. Grant access

|  |
| --- |
| grant all on airflow.\* TO ‘USERNAME'@'%' IDENTIFIED BY ‘{password}'; |

1. Run initdb to setup the database tables

|  |
| --- |
| airflow initdb |

1. Create needed directories

|  |
| --- |
| cd {AIRFLOW\_HOME}  mkdir dags  mkdir logs |

Configuring Airflow – Advanced (Optional)

Email Alerting

Allow Email alerting for if a task or job fails.

1. Edit the {AIRFLOW\_HOME}/airflow.cfg file
2. Set the properties
   1. Properties
      * SMTP\_HOST - Host of the SMTP Server
      * SMTP\_TLS - Whether to use TLS when connecting to the SMTP Server
      * SMTP\_USE\_SSL - Whether to use SSL when connecting to the SMTP Server
      * STMP\_USER - Username for connecting to SMTP Server
      * SMTP\_PORT - Port to use for SMTP Server
      * SMTP\_PASSWORD - Password associated with the user thats used to connect to SMTP Server
      * SMTP\_EMAIL\_FROM - Email to send Alert Emails as
   2. Example

|  |
| --- |
| [email]  email\_backend = airflow.utils.send\_email\_smtp  [smtp]  # If you want airflow to send emails on retries, failure, and you want to  # the airflow.utils.send\_email function, you have to configure an smtp  # server here  smtp\_host = {SMTP\_HOST}  smtp\_starttls = {SMTP\_TLS: True or False}  smtp\_ssl = {SMTP\_USE\_SSL: True or False}  smtp\_user = {STMP\_USER}  smtp\_port = {SMTP\_PORT}  smtp\_password = {SMTP\_PASSWORD}  smtp\_mail\_from = {SMTP\_EMAIL\_FROM} |

Password Authentication

To enable password authentication for the web app.

Follow these instructions: <http://airflow.incubator.apache.org/security.html>

Controlling Airflow Services

By default you have to use the Airflow Command line Tool to startup the services. You can use the bellow commands to startup the processes in the background and dump the output to log files.

Starting Services

1. Start Web Server

|  |
| --- |
| nohup airflow webserver $\* >> ~/airflow/logs/webserver.logs & |

1. Start Celery Workers

|  |
| --- |
| nohup airflow worker $\* >> ~/airflow/logs/worker.logs & |

1. Start Scheduler

|  |
| --- |
| nohup airflow scheduler >> ~/airflow/logs/scheduler.logs & |

1. Navigate to the Airflow UI
   * http://{HOSTNAME}:8080/admin/
2. Start Flower (Optional)
   * Flower is a web UI built on top of Celery, to monitor your workers.

|  |
| --- |
| nohup airflow flower >> ~/airflow/logs/flower.logs & |

1. Navigate to the Flower UI (Optional)
   * http://{HOSTNAME}:5555/

Stopping Services

Search for the service and run the kill command:

|  |
| --- |
| # Get the PID of the service you want to stop  ps -eaf | grep airflow  # Kill the process  kill -9 {PID} |

Setting up Systemd to Run Airflow

Deploy Systemd Scripts

1. Login as Root
2. Get the zipped up Airflow

|  |
| --- |
| cd /tmp/  wget https://github.com/apache/incubator-airflow/archive/{AIRFLOW\_VERSION}.zip  #Example: "wget https://github.com/apache/incubator-airflow/archive/1.7.0.zip" |

1. Unzip the file

|  |
| --- |
| unzip {AIRFLOW\_VERSION}.zip  # This will output extract the contents into: incubator-airflow-{AIRFLOW\_VERSION} |

1. Distribute the Systemd files

|  |
| --- |
| cd incubator-airflow-{AIRFLOW\_VERSION}/scripts/systemd/  # Update the contents of the airflow file.  # Set the AIRFLOW\_HOME if its anything other then the default  vi airflow  # Copy the airflow property file to the target location  cp airflow /etc/sysconfig/  # Update the contents of the airflow-\*.service files  # Set the User and Group values to the user and group you want the airflow service to run as  vi airflow-\*.service  # Copy the airflow services files to the target location  cp airflow-\*.service /etc/systemd/system/ |

How to Use Systemd

Webserver

|  |
| --- |
| # Starting up the Service  service airflow-webserver start  # Stopping the Service  service airflow-webserver stop  # Restarting the Service  service airflow-webserver restart  # Checking the Status of the Service  service airflow-webserver status |

|  |
| --- |
| # Viewing the Logs  journalctl -u airflow-webserver -e |

Celery Worker

|  |
| --- |
| # Starting up the Service  service airflow-worker start  # Stopping the Service  service airflow-worker stop  # Restarting the Service  service airflow-worker restart  # Checking the Status of the Service  service airflow-worker status |

|  |
| --- |
| # Viewing the Logs  journalctl -u airflow-worker -e |

Scheduler

|  |
| --- |
| # Starting up the Service  service airflow-scheduler start  # Stopping the Service  service airflow-scheduler stop  # Restarting the Service  service airflow-scheduler restart  # Checking the Status of the Service  service airflow-scheduler status |

|  |
| --- |
| # Viewing the Logs  journalctl -u airflow-scheduler -e |

Flower (Optional)

|  |
| --- |
| # Starting up the Service  service airflow-flower start  # Stopping the Service  service airflow-flower stop  # Restarting the Service  service airflow-flower restart  # Checking the Status of the Service  service airflow-flower status |

|  |
| --- |
| # Viewing the Logs  journalctl -u airflow-flower -e |

Setting up Airflow Services to Run on Machine Startup

Webserver

|  |
| --- |
| chkconfig airflow-webserver on |

Celery Worker

|  |
| --- |
| chkconfig airflow-worker on |

Scheduler

|  |
| --- |
| chkconfig airflow-scheduler on |

Flower (Optional)

|  |
| --- |
| chkconfig airflow-flower on |

Testing Airflow

Example Dags

<https://github.com/apache/incubator-airflow/tree/master/airflow/example_dags>

High Level Testing

Note: You will need to deploy the [tutorial.py](https://github.com/apache/incubator-airflow/blob/master/airflow/example_dags/tutorial.py) dag.

|  |
| --- |
| airflow test tutorial print\_date 2016-03-30  #[2016-03-30 18:39:46,621] {bash\_operator.py:72} INFO - Output:  #[2016-03-30 18:39:46,623] {bash\_operator.py:76} INFO - Wed Mar 30 18:39:46 UTC 2016 |

Running a Sample Airflow DAG

Assume the following code is in the dag at {AIRFLOW\_HOME}/dags/sample.py

|  |
| --- |
| from airflow import DAG  from airflow.operators import DummyOperator  from datetime import datetime, timedelta  default\_args = {  'owner': 'airflow',  'start\_date': datetime.now() - timedelta(seconds=10),  'retries': 0  }  dag = DAG('sample', default\_args=default\_args, start\_date=datetime.now() - timedelta(seconds=10))  op = DummyOperator(task\_id='dummy', dag=dag) |

Verify the DAG is Available

Verify that the DAG you deployed is available in the list of DAGs

|  |
| --- |
| airflow list\_dags |

The output should list the ‘sample’ DAG

Running a Test

Let’s test by running the actual task instances on a specific date. The date specified in this context is an execution\_date, which simulates the scheduler running your task or dag at a specific date + time:

|  |
| --- |
| airflow test sample dummy 2016-03-30 |

Run

Heres how to run a particular task. Note: It might fail if the dependent tasks are not run successfully.

|  |
| --- |
| airflow run sample dummy 2016-04-22T00:00:00 --local |

Trigger DAG

Trigger a DAG run

|  |
| --- |
| airflow trigger\_dag sample |

Backfill

Backfill will respect your dependencies, emit logs into files and talk to the database to record status. If you do have a webserver up, you’ll be able to track the progress. airflow webserver will start a web server if you are interested in tracking the progress visually as your backfill progresses.

|  |
| --- |
| airflow backfill sample -s 2016-08-21 |

Helpful Operations

Getting Airflow Version

|  |
| --- |
| airflow version |

Find Airflow Site-Packages Installation Location

Sometimes it might be helpful to find the source code so you can perform some other operations to help customize the experience in Airflow. This is how you can find the location of where the airflow source code is installed:

1. Start up a Python CLI

|  |
| --- |
| python |

1. Run the following code to find where the airflow source code is installed

|  |
| --- |
| import site  import os  SITE\_PACKAGES = site.getsitepackages()  print "All Site Packages: " + str(SITE\_PACKAGES)  for site\_package in SITE\_PACKAGES:  test\_path = site\_package + "/airflow"  if os.path.exists(test\_path):  AIRFLOW\_INSTALL\_DIR = test\_path  print "Site Page Containing Airflow: " + str(AIRFLOW\_INSTALL\_DIR) |

Usual Site Package Paths:

* Centos
  + /usr/lib/python2.7/site-packages

Change Alert Email Subject

By default, the Airflow Alert Emails are always sent with the subject like: **Airflow alert: <TaskInstance: [DAG\_NAME].[TASK\_ID] [DATE] [failed]>**. If you would like to change this to provide more information as to which Airflow cluster you’re working with you can follow these steps.

Note: It requires a very small modification of the Airflow Source Code.

1. Go to the Airflow Site-Packages Installation Location
   1. Example Path: /usr/lib/python2.7/site-packages/airflow
2. Edit the models.py file
3. Search for the text “Airflow alert: ”
   1. Using nano
      1. Open the file
      2. Hit CTRL+w
      3. Type in “Airflow alert” and hit enter
4. Modify this string to whatever you would like.
   1. Original value ‘title = “*Airflow alert: {self}”.format(\*\*locals())”‘* will produce ‘Airflow alert: <TaskInstance: [DAG\_NAME].[TASK\_ID] [DATE] [failed]>’
   2. An updated value like ‘title = “*Test Updated Airflow alert: {self}”.format(\*\*locals())”*‘ will produce ‘Test Updated Airflow alert: <TaskInstance: [DAG\_NAME].[TASK\_ID] [DATE] [failed]>’

Set Logging Level

If you want to get more information in the logs (debug) or log less information (warn) you can follow these steps to set the logging level

Note: It requires a very small modification of the Airflow Source Code.

1. Go to the Airflow Site-Packages Installation Location of airflow
2. Edit the settings.py file
3. Set the LOGGING\_LEVEL variable to your desired value
   1. debug → logging.DEBUG
   2. info → logging.INFO
   3. warn → logging.WARN
4. Restart the Airflow Services

[Share on Facebook Share](https://www.facebook.com/sharer/sharer.php?u=http://site.clairvoyantsoft.com/installing-and-configuring-apache-airflow/" \o "Share on Facebook" \t "_blank)

[Share on TwitterTweet](https://twitter.com/intent/tweet?text=Installing%20and%20Configuring%20Apache%20Airflow&url=http%3A%2F%2Fsite.clairvoyantsoft.com%2Finstalling-and-configuring-apache-airflow%2F&" \o "Share on Twitter" \t "_blank)