Avionix Airflow

Release 0.2.3+0.gb7acee4.dirty

Zach Brookler

Oct 02, 2020

CONTENTS

Date Oct 02, 2020 **Version**: 0.2.3+0.gb7acee4.dirty

Useful links Source Repository | Issues & Ideas

Why are there so many guides on how to set up airflow when the code could just be given to you?!

Welcome to `avionix_airflow`, an out of the box solution to installing airflow on kubernetes.

`avionix_airflow` comes with support for elasticsearch log and metric storage, grafana monitoring setup, and a fully functional airflow KubernetesExecutor implementation.

CHAPTER

ONE

USER GUIDE

1.1 Minikube

1.1.1 Minikube Setup

For the minikube setup you will need to install minikube and docker.

To start minikube:

```
minikube start --driver=docker
```

To start ingress controller (nginx by default):

```
minikube addons enable ingress
```

1.1.2 Minikube Usage

This is an example of how to install airflow using *avionix_airflow* locally.

```
import logging

from avionix_airflow import add_host, build_airflow_image, get_chart_builder
from avionix_airflow.kubernetes.airflow import AirflowOptions
from avionix_airflow.kubernetes.monitoring import MonitoringOptions
from avionix_airflow.tests.utils import dag_copy_loc, parse_shell_script

logger = logging.getLogger()
logger.setLevel(level=logging.INFO)
logging.basicConfig(format="[%(filename)s:%(lineno)s] %(message)s", level=logging.
↪INFO)


def install_chart():
    build_airflow_image()
    airflow_options = AirflowOptions(
        dag_sync_image="alpine/git",
        dag_sync_command=["/bin/sh", "-c", parse_shell_script(str(dag_copy_loc))],
        dag_sync_schedule="* * * * *",
        default_timezone="est",
        core_executor="KubernetesExecutor",
        open_node_ports=True,
        image_pull_policy="Never",
```

(continues on next page)

(continued from previous page)

```
    )
    add_host(airflow_options)
    builder = get_chart_builder(
        airflow_options=airflow_options,
        monitoring_options=MonitoringOptions(grafana_role="Admin"),
    )
    if builder.is_installed:
        builder.upgrade_chart({"dependency-update": None})
        return
    builder.install_chart(options={"create-namespace": None, "dependency-update":␣
↪None})
```

1.2 AWS EKS Managed Node Group

1.2.1 AWS EKS Setup

Requirements

1. VPC
2. At least 2 subnets in that VPC (These should be marked properly for load balancing, see alb-ingress
3. EKS Cluster

   The above requirements can be created using eksctl

   Avionix_airflow is also preconfigured to use kube2iam, which can be used to assign roles to your kubernetes pods
4. An AWS EFS Volume (Be sure this is available in the same subnets as the cluster and the security group allows traffic from the cluster's security group)
5. An IAM role that is authorized to manipulate ALB's based on the subnet tags you wrote above. You can find a minimal policy for this here
6. An IAM role that can set up an record dns in route53. You can find information about the required policy here
7. An IAM role that can control the autoscaling group. The required policy can be found here

Additional Requirements (Recommended)

1. **AWS ElasticSearch Cluster**
   1. An IAM role that is authorized to connect to the cluster
2. Ensure that the security group for the cluster is allowed TCP access on port 443 from the EKS cluster security group

After all of the above requirements are met, follow the main documentation.

1.2.2 AWS Usage

This is an example of how to install airflow using *avionix_airflow* on AWS.

```
from avionix_airflow import (
    AirflowOptions,
    AwsOptions,
    MonitoringOptions,
    SqlOptions,
    get_chart_builder,
)

builder = get_chart_builder(
    airflow_options=AirflowOptions(
        dag_sync_image="alpine/git",
        dag_sync_command=["/bin/sh", "-c", "my_shell_commands"],
        dag_sync_schedule="* * * * *",
        default_timezone="est",
        core_executor="KubernetesExecutor",
        domain_name="my.airflow.domain.com",
    ),
    monitoring_options=MonitoringOptions(
        elastic_search_uri="https://my-es-vpc.es.amazonaws.com",
    ),
    sql_options=SqlOptions(
        user="postgres-user",
        password="****",
        host="my-postgres-host.amazonaws.com",
        create_database_in_cluster=False,
    ),
    cloud_options=AwsOptions(
        efs_id="fs-12345",
        cluster_name="my-cluster",
        elastic_search_access_role_arn="arn:aws:iam::123456789012:role/es-role",
        default_role_arn="arn:aws:iam::123456789012:role/default-role",
        alb_role_arn="arn:aws:iam::123456789012:role/alb-role",
        external_dns_role_arn="arn:aws:iam::123456789012:role/external-dns-role",
        autoscaling_role_arn="arn:aws:iam::123456789012:role/autoscaling-role",
        domain="my.airflow.domain.com",
        dag_sync_role_arn="arn:aws:iam::123456789012:role/dag_sync_role",
        use_ssl=True,
    ),
)
```

CHAPTER

TWO

AVIONIX AIRFLOW REFERENCE

Avionix airflow functions using two main components.

1. A functions that provides an avionix `ChartBuilder` object
2. Options objects that set different values in the kubernetes components

2.1 Chart Builder Generation

This is the function that will return an avionix `ChartBuilder` object

avionix_airflow.**get_chart_builder**(*airflow_options*, *sql_options=<avionix_airflow.kubernetes.postgres.sql_options.SqlO object>*, *redis_options=<avionix_airflow.kubernetes.redis.redis_options.RedisOptio object>*, *monitoring_options=<avionix_airflow.kubernetes.monitoring.monitoring_o object>*, *cloud_options=<avionix_airflow.kubernetes.cloud.local.local_options.Loca object>*)

Parameters

- **sql_options** (`SqlOptions`) – An SqlOptions object
- **redis_options** (`RedisOptions`) – A RedisOptions object
- **airflow_options** (`AirflowOptions`) – An AirflowOptions object
- **monitoring_options** (`MonitoringOptions`) – A MonitoringOptions object
- **cloud_options** (`CloudOptions`) – A CloudOptions object

Return type `ChartBuilder`

Returns Avionix ChartBuilder object that can be used to install airflow

Documentation on how to use the `ChartBuilder` can be found here

2.2 Avionix Airflow Options

avionix_airflow's configuration can be changed by using option objects. For the local implementation most settings will not need to be changed, but for cloud implementations many will likely need to be set, in particular the `AwsOptions` and `AirflowOptions`.

class avionix_airflow.**AirflowOptions**(*dag_sync_image*, *dag_sync_command*, *dag_sync_schedule*, *dag_storage='50Mi'*, *log_storage='50Mi'*, *external_storage='50Mi'*, *default_executor_cpu=5*, *default_executor_memory=2*, *access_modes=None*, *default_timezone='utc'*, *core_executor='CeleryExecutor'*, *namespace='airflow'*, *domain_name='www.avionix-airflow.com'*, *additional_vars=None*, *fernet_key=''*, *dags_paused_at_creation=True*, *worker_image=''*, *worker_image_tag=''*, *open_node_ports=False*, *local_mode=False*, *smtp_notification_options=None*, *git_ssh_key=None*, *image_pull_policy='IfNotPresent'*, *master_image=''*, *master_image_tag=''*, *delete_pods_on_failure=False*)

Class for storing airflow options. Note that for storage specification the default unit is bytes, and you can use Mi or Gi to specify larger units.

Parameters

- **dag_sync_image** (str) – The image to use for syncing dags
- **dag_sync_command** (List[str]) – The commands to use for syncing dags
- **dag_sync_schedule** (str) – Cron format schedule for how often to sync dags
- **dag_storage** (str) – How much persistent storage to allocate for dags
- **logs_storage** – How much persistent storage to allocate for dags
- **external_storage** (str) – How much external storage to allocate for dags
- **default_executor_cpu** (int) – Default CPU to allocate to the executor This is specified in number of vCPUs. You can fractions of CPUs as well or milliCPUs (eg. "500m" for 500 milliCPUs)
- **default_executor_memory** (int) – Default memory to allocate to the executor This can be specified in bytes (default) or using standard prefix (eg. "20Mi")
- **access_modes** (Optional[InitVar]) – Used to set access modes on the dag and logs volumes
- **default_timezone** (str) – The default timezone
- **core_executor** (str) – The core executor to use, supported executors are "KubernetesExecutor" and "CeleryExecutor"
- **namespace** (str) – The kubernetes namespace to use for the installation
- **domain_name** (Optional[str]) – The domain name to place on the ingress controller
- **additional_vars** (Optional[InitVar]) – Any additional environment variables to place on the airflow nodes. Note that you can add additional airflow configuration using this setting. For more info on this see the airflow config documentation
- **fernet_key** (InitVar) – The fernet key to use for airflow, this is autogenerated by default, so unless you need to use an old fernet key, just leave this
- **dags_paused_at_creation** (bool) – Whether or not dags should be paused on creation
- **worker_image** (str) – The docker image to use as the airflow worker
- **worker_image_tag** (str) – The docker tag of the airflow worker image

- **open_node_ports** (bool) – Whether to expose the node ports
- **local_mode** (bool) – Whether or not to run in local mode
- **image_pull_policy** (str) – Policy for pulling docker image. Can be one of "Never", "Always", "IfNotPresent"
- **master_image_tag** (str) – The docker tag to use for the master image
- **delete_pods_on_failure** (bool) – Whether or not to terminate pods if KubernetesExecutor task fails

class avionix_airflow.**AwsOptions**(*efs_id*, *cluster_name*, *elastic_search_access_role_arn*, *default_role_arn*, *alb_role_arn*, *external_dns_role_arn*, *autoscaling_role_arn*, *dag_sync_role_arn*, *domain*, *domain_filters=None*, *use_ssl=False*)

Settings for the AWS Managed Node Group setup

Parameters

- **efs_id** (str) – The id of the EFS file system to use as the storage provider
- **cluster_name** (str) – The name of the EKS cluster
- **elastic_search_access_role_arn** (str) – The IAM role arn for accessing elastic search
- **default_role_arn** (str) – The default IAM role arn for pods in the cluster
- **alb_role_arn** (str) – The IAM role setting up application load balancing
- **external_dns_role_arn** (str) – The IAM role for setting the domain name
- **autoscaling_role_arn** (str) – The IAM role for controlling the cluster scaling
- **domain** (str) – The AWS domain name to use
- **dag_sync_role_arn** (str) – The IAM role use for controlling retrieval of dags
- **domain_filters** (Optional[List[str]]) – A list used to filter out route53 domain names
- **use_ssl** (bool) – Whether or not to use SSL encryption, (Recommended to be True)

class avionix_airflow.**MonitoringOptions**(*enabled=True*, *elastic_search_uri='http://elasticsearch-master:9200'*, *grafana_role='Viewer'*, *elastic_search_proxy_uri='elasticsearch-master'*)

Configurations for monitoring airflow

Parameters

- **enabled** (bool) – Whether or not monitoring should be enabled
- **elastic_search_uri** (str) – The uri to use for elastic search
- **grafana_role** (str) – The role to use for grafana can be one of ["Viewer", "Editor", "Admin"]
- **elastic_search_proxy_uri** (str) – The uri to use as the elastic search proxy, this shouldn't be changed unless using an external elastic search provider

class avionix_airflow.**RedisOptions**(*port=6379*, *host='redis-svc'*, *proto='redis://'*, *password=''*, *db_num=1*)

Configuration for redis if using "CeleryExecutor" option in AirflowOptions

Parameters

- **port** (int) – The port to use for redis
- **host** (str) – Name of the redis host, defaults to the internal service name
- **proto** (str) – Redis protocol
- **password** (str) – Redis password
- **db_num** (int) – Redis db number

class avionix_airflow.**SqlOptions**(*user='airflow'*, *password='airflow'*, *host='airflow-database-connection'*, *port=5432*, *db='airflow'*, *create_database_in_cluster=True*, *extras=''*)

Provides configuration for the airflow backend database. Currently only postgres is supported. Note that all connection related strings will be stored in kubernetes secrets on the cluster.

Parameters

- **user** (str) – The username for database
- **password** (str) – The pass for the database
- **host** (str) – The hostname for the database
- **port** (int) – The port to keep open for the database
- **db** (str) – The database name
- **create_database_in_cluster** (bool) – Whether airflow should create the database
- **extras** – Any extra params to be appended to the end of the db connection string

PYTHON MODULE INDEX

a

INDEX