Running Apache Airflow reliably with Kubernetes

...

and other open source software

April 17, 2019
Data Council
San Francisco, CA

Greg Neiheisel, CTO

ASTRONOMER

On Deck

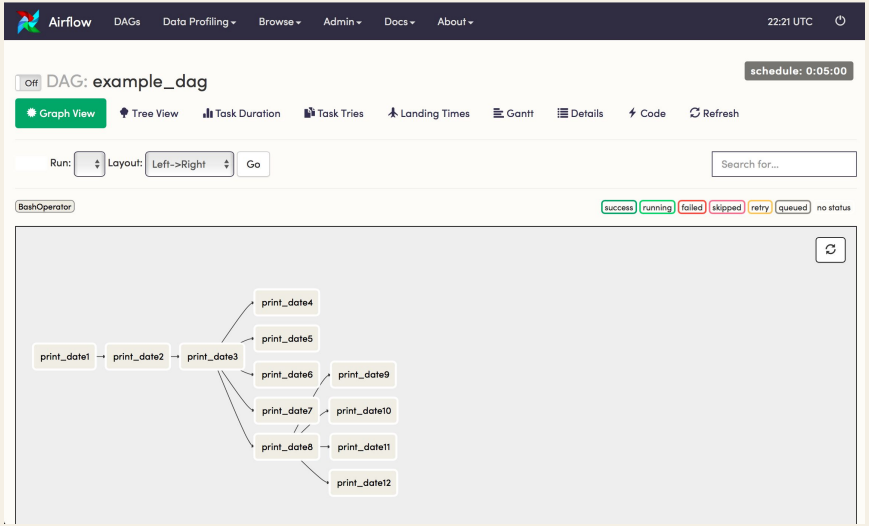
- Quick Airflow / Kubernetes overview
- Running Airflow at Scale
- Major system design considerations
- Lessons and best practices we've learned along the way

What is Apache Airflow?

- A task scheduler written in Python to programatically author, schedule, and monitor dependency driven workflows (DAGs)
  - Pluggable architecture, focused on ETL, ML use-cases
  - Lots of existing building blocks
- Top-level Apache Project
  - 11,000+ stars on github
  - 6,000+ commits
  - 700+ contributors

Airflow core concepts

- **DAGs** - created in code, typically associated with a cron schedule
- **DAG Runs** - typically execution of a dag for a given execution date
- **Task Instances** - represents an execution of a node in the DAG



1: 2: 3: 9: 0% 12 9% 117°F 0.4 / 0.4 KB/s Enabled 91% 49% Thu Apr 11 12:57:03 PM

Airflow - DAGs

galactic-cosmology-3441-airflow.datarouter.ai/admin/airflow/graph?dag_id=example_dag_two&root=&execution_date=2018-01-03T07%3A25%3A00%2B00%3A00

Airflow DAGs Data Profiling Browse Admin Docs About Astronomer 2019-04-11 16:57:03 UTC

On DAG: example_dag_two

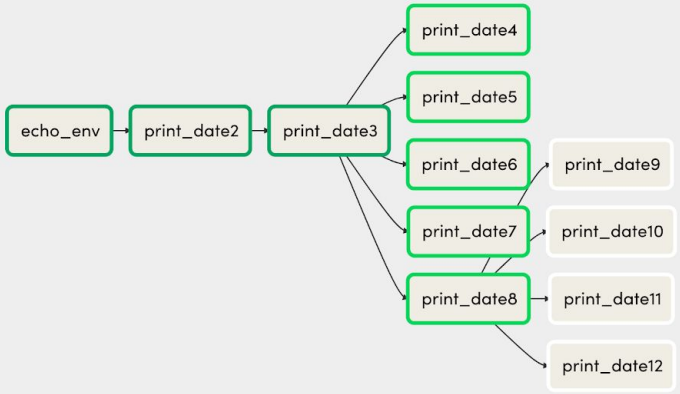
schedule: 0:05:00

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Refresh Delete

running Base date: 2018-01-03 07:25:01 Number of runs: 25 Run: scheduled__2018-01-03T07:25:00+00:00 Layout: Left->Right Go Search for...

BashOperator

success running failed skipped retry queued no status



1: 2: 3: 9: 0% 12 10% 118°F 931.2/ 13.3 KB/s Enabled 92% 48% Thu Apr 11 12:58:31 PM

Airflow - DAGs

galactic-cosmology-3441-airflow.datarouter.ai/admin/airflow/tree?dag_id=example_dag_two&num_runs=&root=

Airflow DAGs Data Profiling Browse Admin Docs About Astronomer 2019-04-11 16:58:30 UTC

On DAG: example_dag_two

schedule: 0:05:00

Graph View | Tree View | Task Duration | Task Tries | Landing Times | Gantt | Details | Code | Refresh | Delete

Base date: 2018-01-03 07:35:00 Number of runs: 25 Go

BashOperator

success running failed skipped retry queued no status

1: 2: 3: 9: 0% 12 5% 113°F 0.7 / 0.4 KB/s Enabled 92% 53% Thu Apr 11 12:53:26 PM

Airflow - DAGs

galactic-cosmology-3441-airflow.datarouter.ai/admin/airflow/log?dag_id=example_dag_two&task_id=echo_env&execution_date=2018-01-03T07%3A25%3A00%2B00%3A00

Airflow DAGs Data Profiling Browse Admin Docs About Astronomer 2019-04-11 16:53:26 UTC

On DAG: example_dag_two

schedule: 0:05:00

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Refresh Delete

Task Instance: echo_env 2018-01-03 07:25:00

Task Instance Details Rendered Template Log XCom

Log by attempts

1

```
Dependencies all met for <TaskInstance: example_dag_two.echo_env 2018-01-03T07:25:00+00:00 [queued]>
Dependencies all met for <TaskInstance: example_dag_two.echo_env 2018-01-03T07:25:00+00:00 [queued]>

--------------------------------------------------------------------------------
Starting attempt 1 of 2
--------------------------------------------------------------------------------

Executing <Task(BashOperator): echo_env> on 2018-01-03T07:25:00+00:00
Running: ['bash', '-c', 'airflow run example_dag_two echo_env 2018-01-03T07:25:00+00:00 --job_id 7976 --raw -sd DAGS_FOLDER/example-dag.py --cfg_path /tmp/tmptu2qbsac']
Job 7976: Subtask echo_env [2019-04-11 16:48:20,812] {settings.py:174} INFO - settings.configure_orm(): Using pool settings. pool_size=5, pool_recycle=1800, pid=55847
Job 7976: Subtask echo_env [2019-04-11 16:48:21,510] {__init__.py:51} INFO - Using executor CeleryExecutor
Job 7976: Subtask echo_env [2019-04-11 16:48:21,777] {models.py:273} INFO - Filling up the DagBag from /usr/local/airflow/dags/example-dag.py
Job 7976: Subtask echo_env [2019-04-11 16:48:21,806] {cli.py:520} INFO - Running <TaskInstance: example_dag_two.echo_env 2018-01-03T07:25:00+00:00 [running]> on host galactic-cosmology-3441-
Job 7976: Subtask echo_env [2019-04-11 16:48:21,842] {bash_operator.py:77} INFO - Tmp dir root location:
Job 7976: Subtask echo_env  /tmp
Job 7976: Subtask echo_env [2019-04-11 16:48:21,842] {bash_operator.py:86} INFO - Exporting the following env vars:
Job 7976: Subtask echo_env AIRFLOW_CTX_DAG_ID=example_dag_two
Job 7976: Subtask echo_env AIRFLOW_CTX_TASK_ID=echo_env
Job 7976: Subtask echo_env AIRFLOW_CTX_EXECUTION_DATE=2018-01-03T07:25:00+00:00
Job 7976: Subtask echo_env AIRFLOW_CTX_DAG_RUN_ID=scheduled__2018-01-03T07:25:00+00:00
```

On DAG: example_dag_two

schedule: 0:05:00

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Refresh Delete

example_dag_two

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta

default_args = {
        'owner': 'schnie',
        'depends_on_past': False,
        'start_date': datetime(2018, 1, 1),
        'email_on_failure': False,
        'email_on_retry': False,
        'retries': 1,
        'retry_delay': timedelta(minutes=5),
}

dag = DAG('example_dag_two',
                        max_active_runs=3,
                        schedule_interval=timedelta(minutes=5),
                        default_args=default_args)

t1 = BashOperator(
        task_id='echo_env',
        bash_command='sleep 120s',
        dag=dag)

t2 = BashOperator(
        task_id='print_date2',
        bash_command='sleep 120s',
        dag=dag)
```

Times are changing

- Wider Use Cases
  - ETL, ML, Reporting, Data Integrity
- Higher Usage
  - More teams with different skill sets and goals for Airflow usage
  - More DAGs running more frequently
- Stricter SLAs
- More complex core components (executors, operators, etc)
  - Kubernetes, Mesos, Spark, etc.
- Immutable infrastructure

10 data engineers
240+ active DAGs
5400+ tasks per day

...as of April '18...

https://speakerdeck.com/vananth22/operating-data-pipeline-with-airflow-at-slack?slide=6

Airflow is a highly-available, mission-critical service

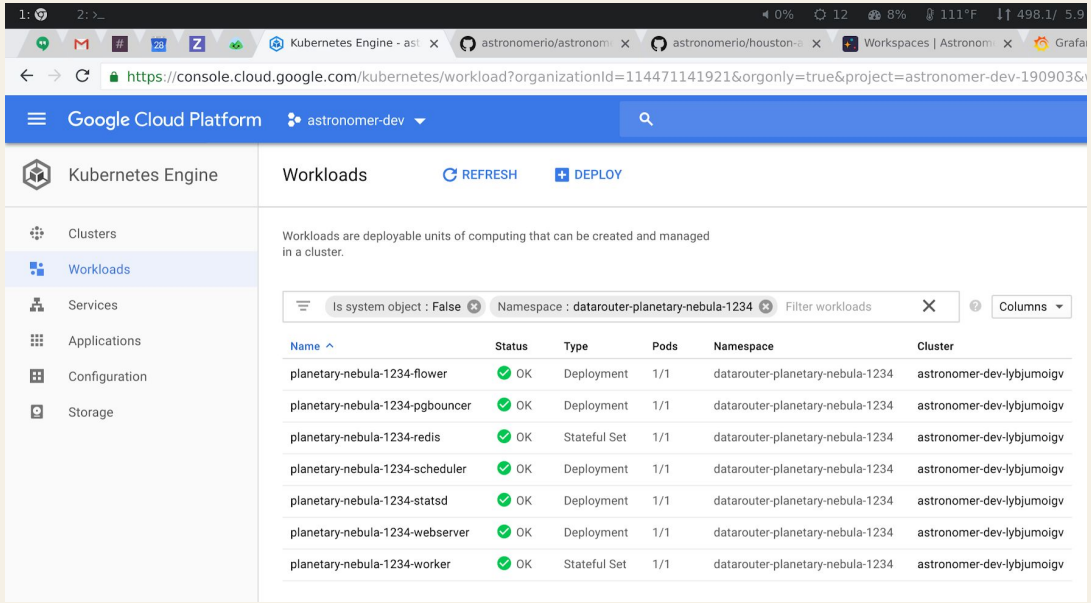
- Automated Airflow deployments
- Continuous delivery
- Support 100s of users and 1,000s of tasks per day
- Security
- Access controls
- Observability (Metrics / Logs)
- Autoscaling / Scale to zero-ish

Kubernetes

Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation

Kubernetes

Applications are broken into smaller, independent pieces and can be deployed and managed dynamically



Kubernetes

- **Pod** - One or more colocated containers, share volumes, ports
- **Deployment** - Higher level abstraction, manages pods, replica sets
- **Stateful Set** - Similar to Deployment, except each replica gets a stable hostname and can mount persistent volumes
- **Daemon Set** - Replica pods deployed to each node
- **Namespace** - Virtual cluster backed by the same physical cluster

Declarative Service Definition with Kubernetes / Helm

Helm helps you manage Kubernetes applications — Helm Charts helps you define, install, and upgrade even the most complex Kubernetes application.

helm.astronomer.io > worker-statefulset.yaml

```yaml
#################################
## Airflow Worker StatefulSet
#################################
{{- if eq .Values.executor "CeleryExecutor" }}
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: {{ .Release.Name }}-worker
  labels:
    tier: airflow
    component: worker
    release: {{ .Release.Name }}
    workspace: {{ .Values.platform.workspace | quote }}
    chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
    heritage: {{ .Release.Service }}
spec:
  serviceName: {{ .Release.Name }}-worker
  replicas: {{ .Values.workers.replicas }}
  podManagementPolicy: Parallel
  selector:
    matchLabels:
      tier: airflow
      component: worker
      release: {{ .Release.Name }}
      workspace: {{ .Values.platform.workspace | quote }}
```

https://github.com/astronomerio/helm.astronomer.io/tree/master/charts/airflow

```
helm install -n airflow-prod charts/airflow
```

Airflow Executors

A pluggable way to scale out Airflow workloads. Responsible for running

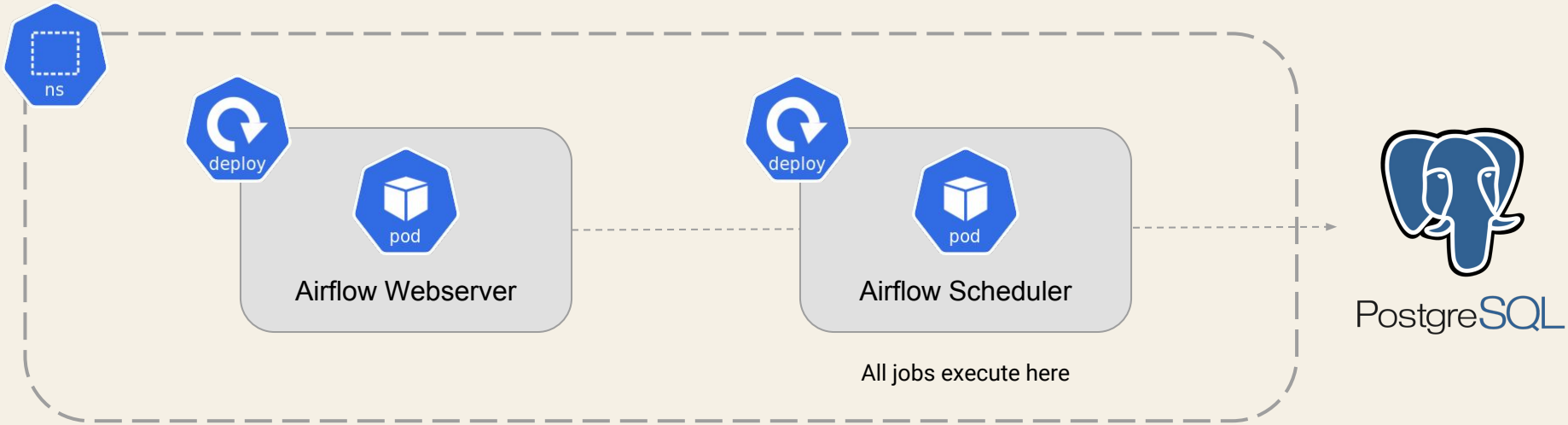
```
airflow run ${dag_id} ${task_id} ${execution_date}
```

somewhere.

Executors - Sequential/Local

- Fork off and run tasks in subprocess
- Good for simple workloads
- Eventually things need to scale out

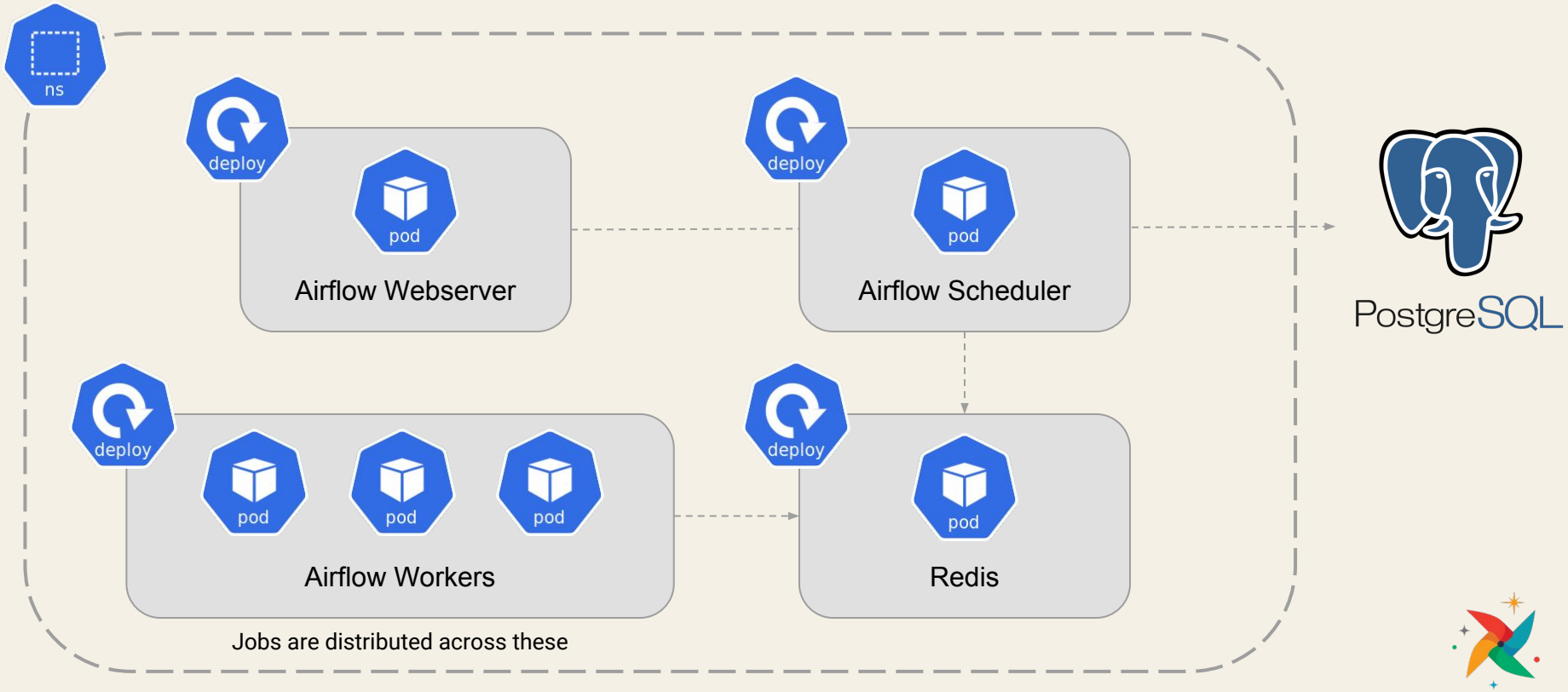
Executors - Local Executor



Executors - Celery Executor

- Distributed Task Queue
- Redis, RabbitMQ, etc dependency
- Configure number of workers
  - Kubernetes HorizontalPodAutoscaler
- Configure worker size
  - Kubernetes resource requests / limits

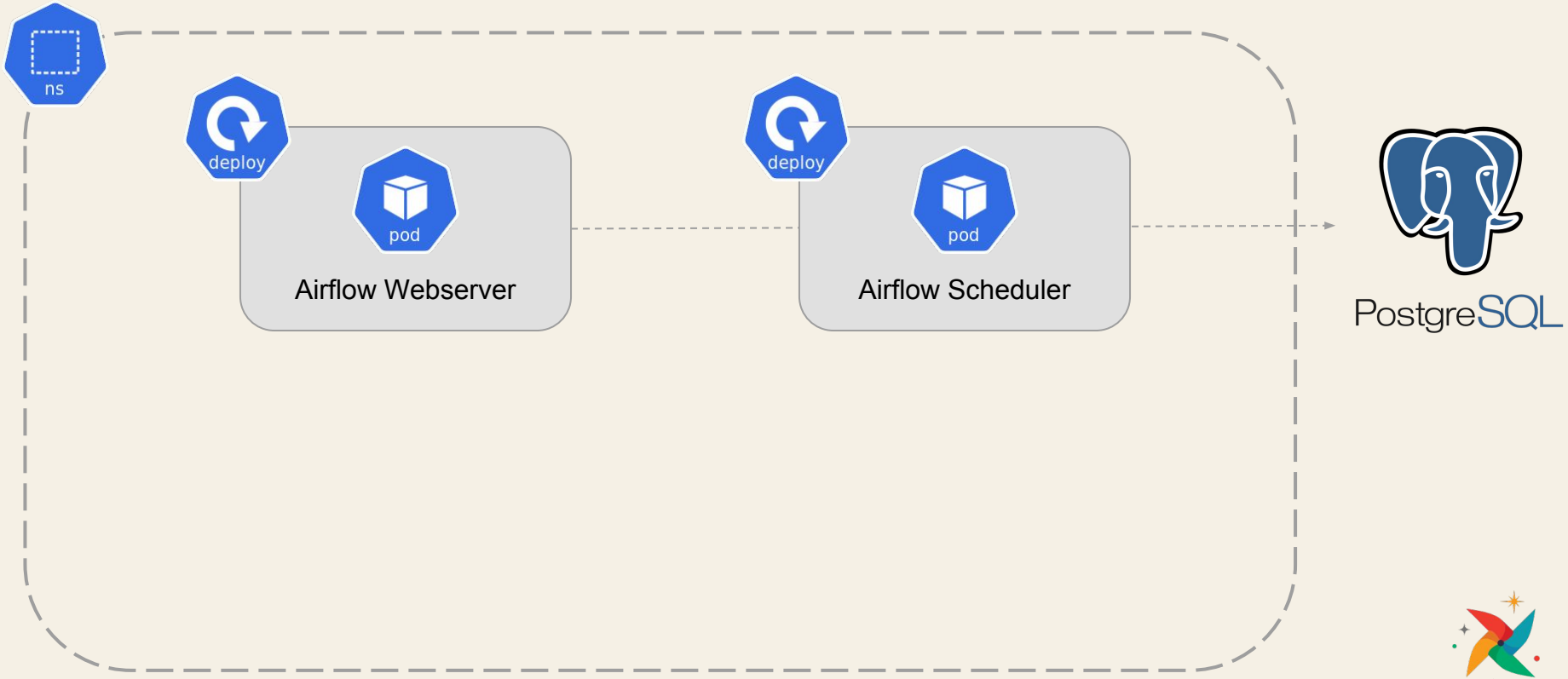
Executors - Celery Executor



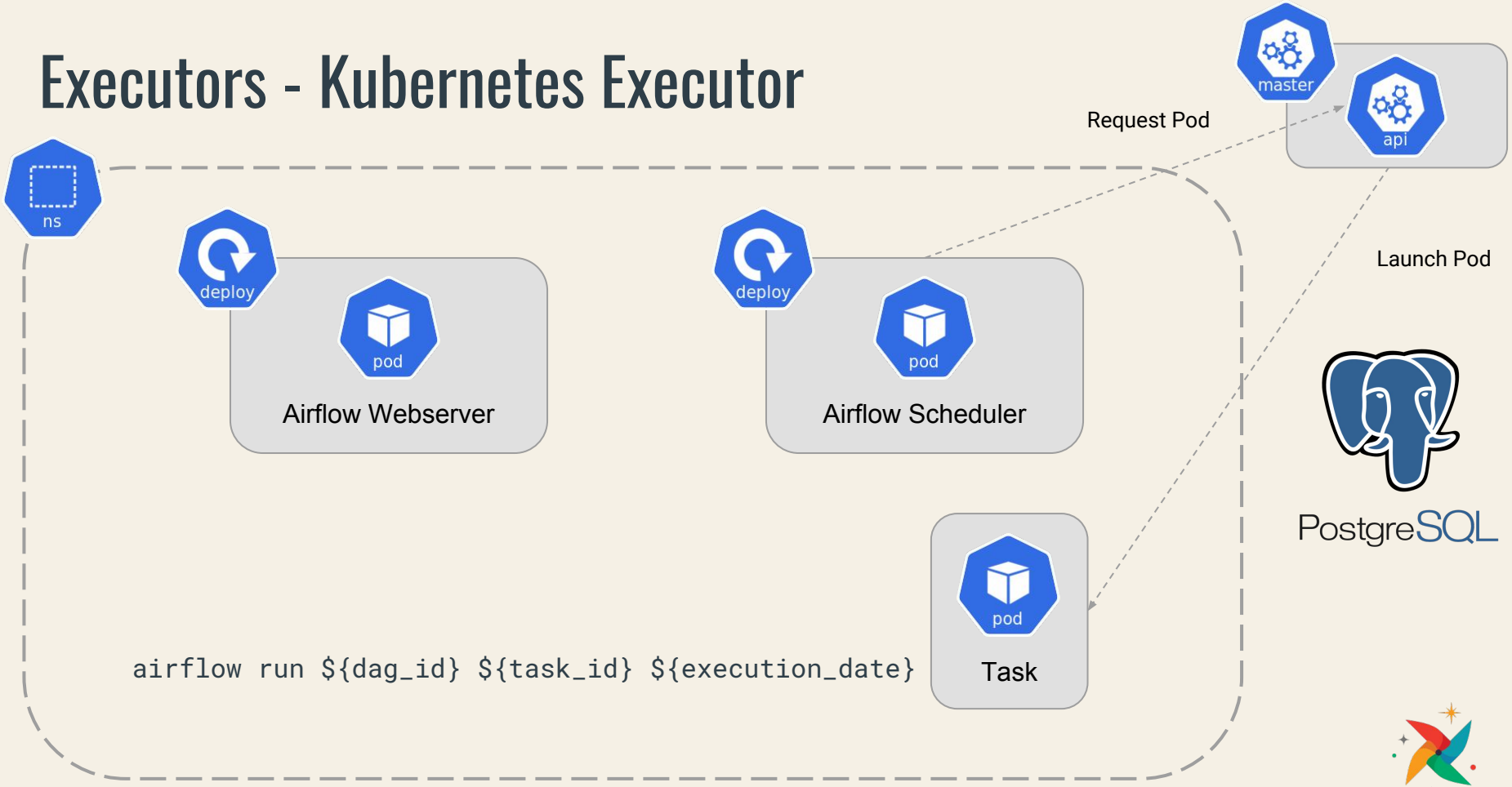
Executors - Kubernetes Executor

- Scale to zero / near-zero
- Each task runs in a new pod
  - Configurable resource requests (cpu/mem)
- Scheduler subscribes to Kubernetes event stream
- Pods run to completion
- Straightforward and natural
- DAG distribution
  - Git clone with init container for each pod
  - Mount volume with DAGs
  - Ensure the image already contains the DAG code

Executors - Kubernetes Executor

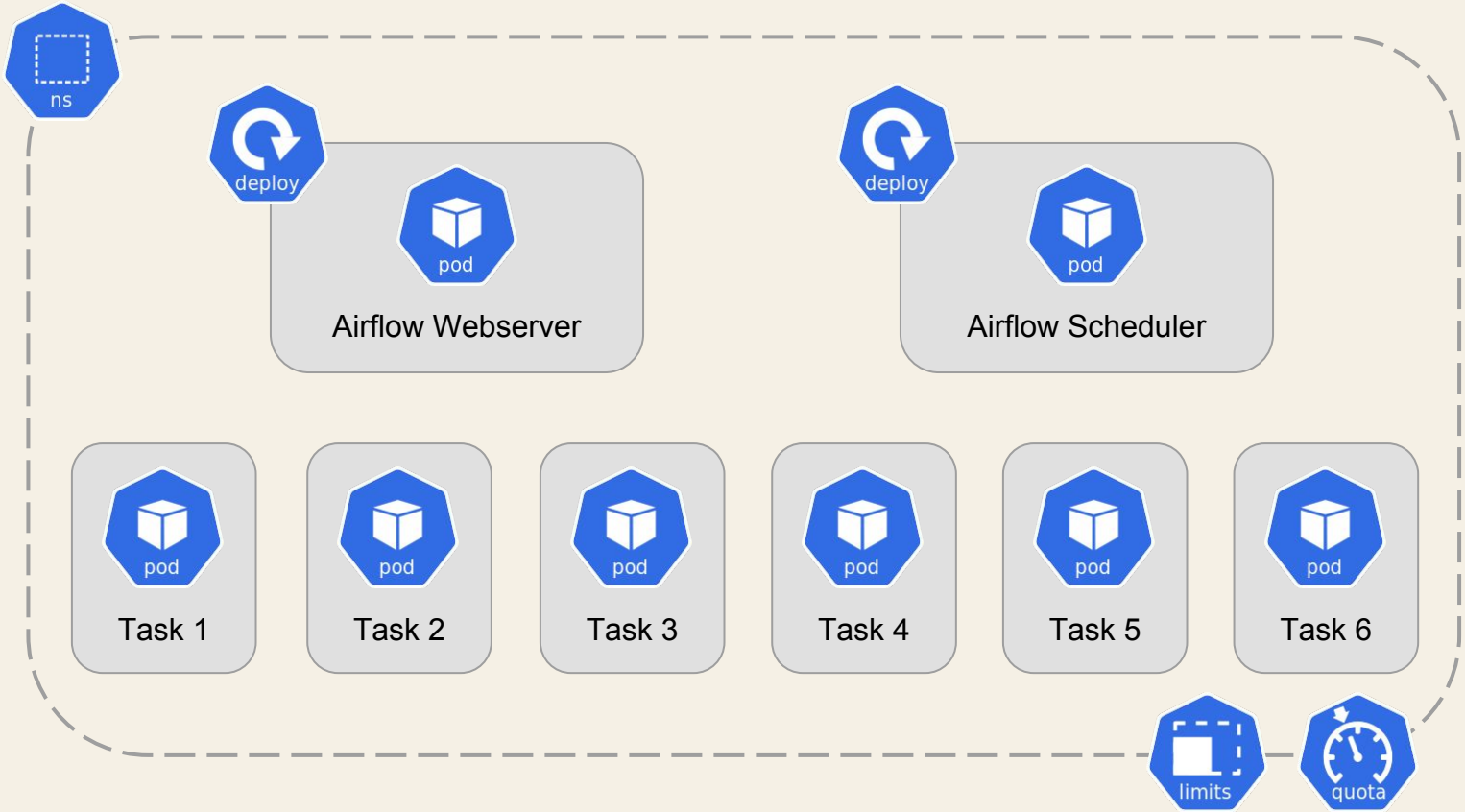


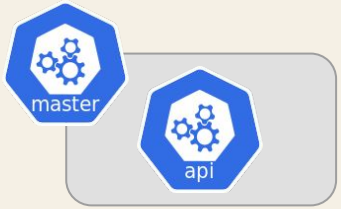
Executors - Kubernetes Executor




```
airflow run ${dag_id} ${task_id} ${execution_date}
```

Executors - Kubernetes Executor

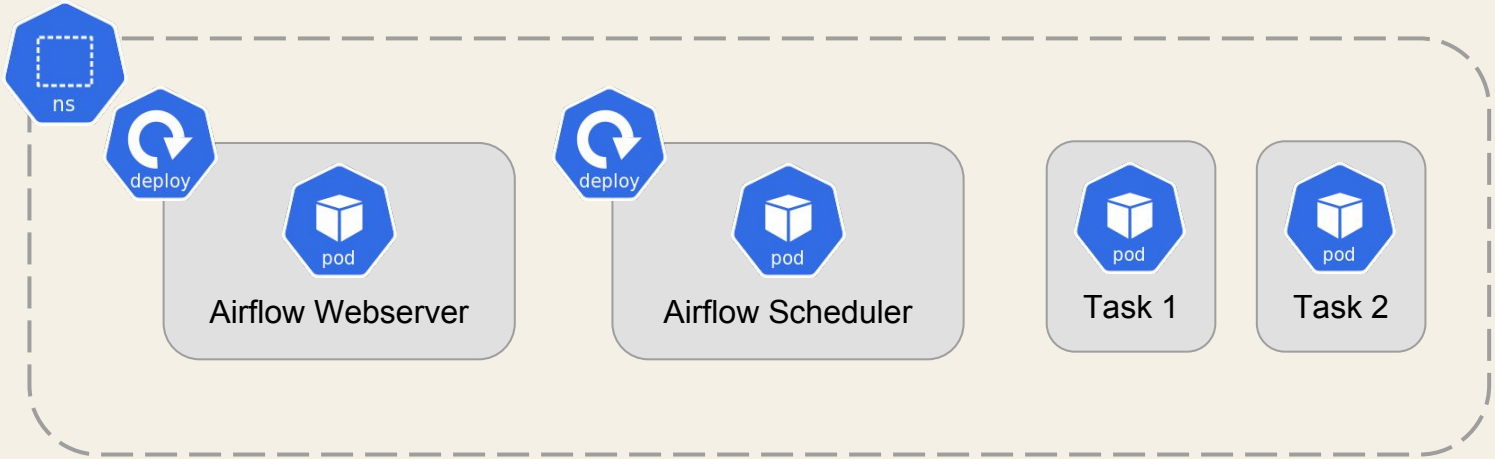




How do we deploy DAG updates to a running environment?

```
helm upgrade airflow-prod charts/airflow --set tag=v0.0.2
```

DAG Updates



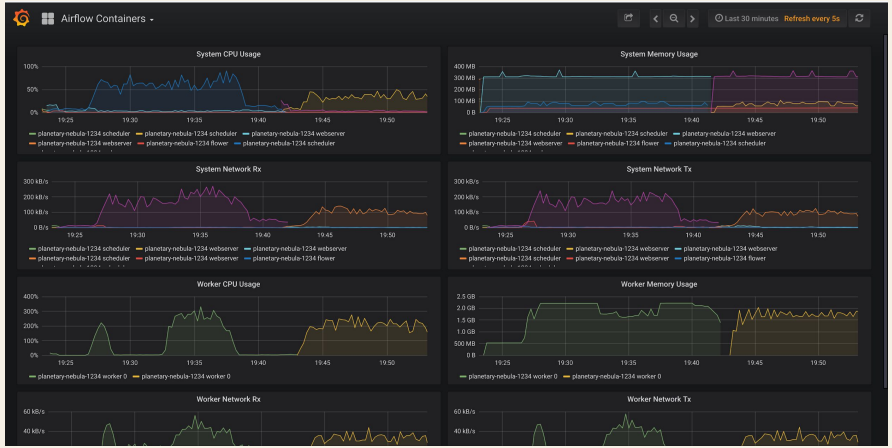
- `helm upgrade` updates the Deployments state in Kubernetes
- Kubernetes gracefully terminates the webserver and scheduler and reboots pods with updated image tag
- Task pods continue running to completion
- You experience negligible amount of downtime
- Can be automated via CI/CD tooling

How do we monitor and alert across a number of Airflow deployments?

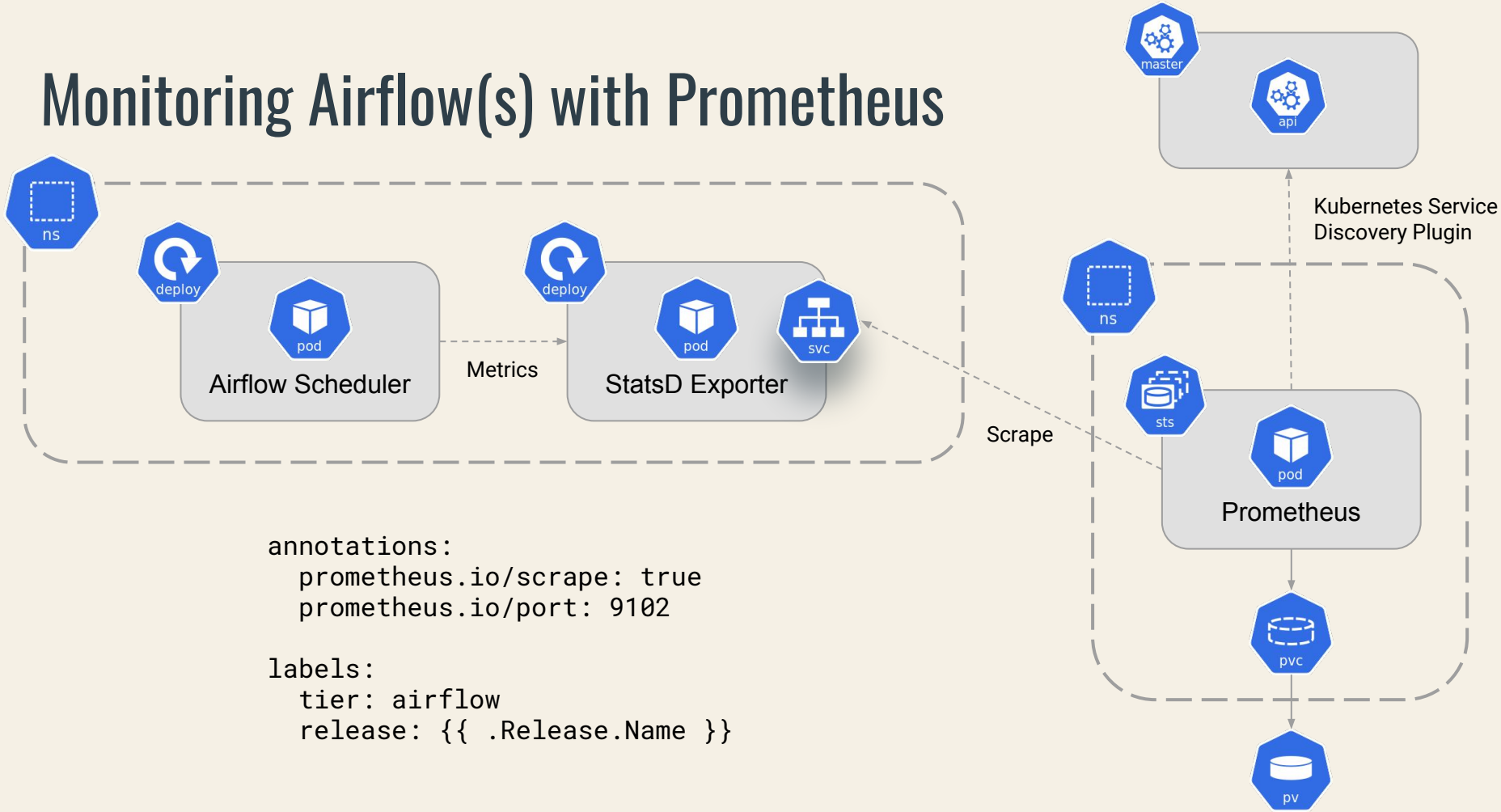
```
helm install stable/prometheus
```

Monitoring Airflow(s) with Prometheus

- Prometheus
  - Also CNCF project
  - Time series database
  - Pull-based
  - Auto-scrape with kubernetes annotations and SD plugin
  - Works great with Grafana →
- Airflow natively exports statsd metrics
- Statsd Exporter as a bridge to Prometheus

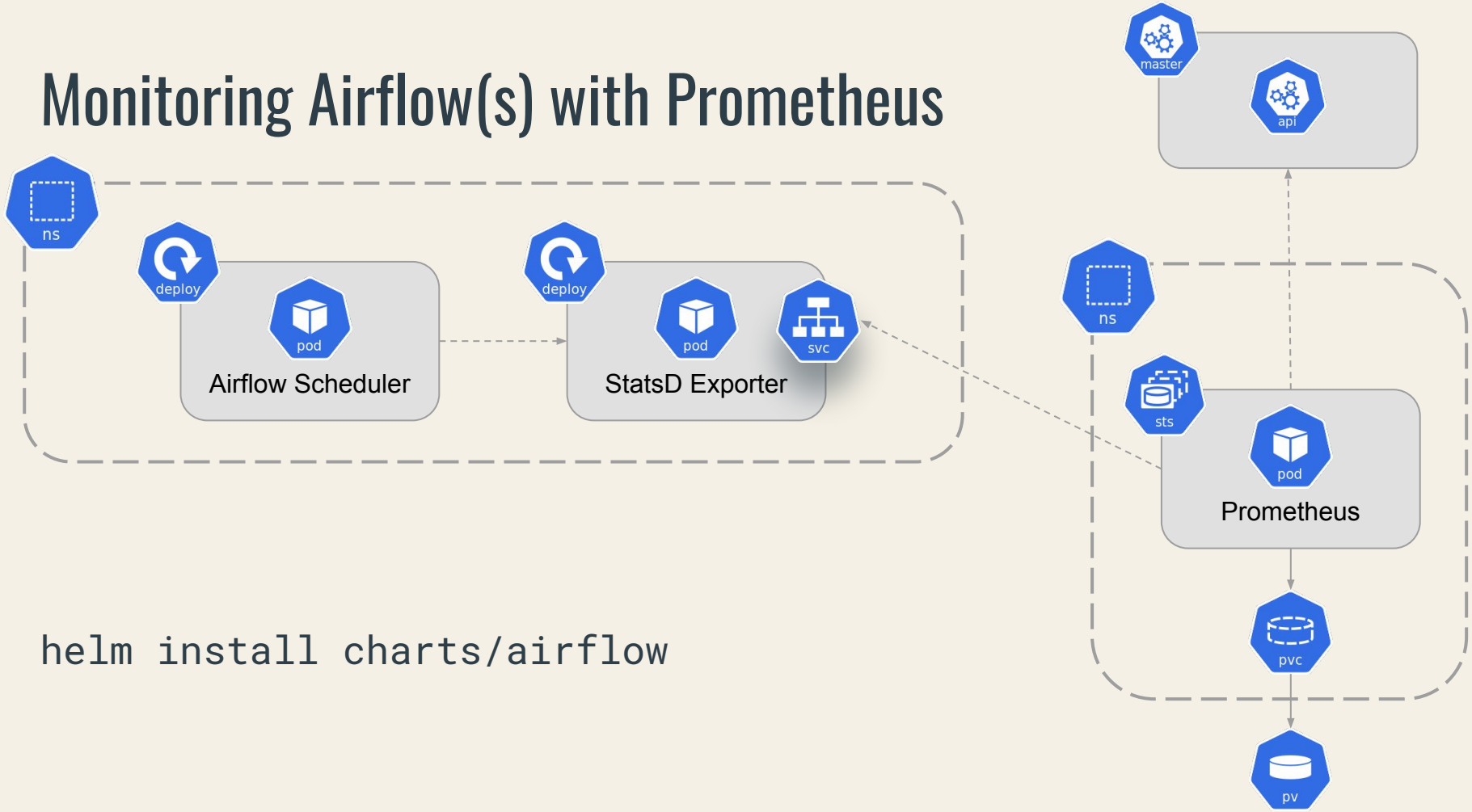


Monitoring Airflow(s) with Prometheus



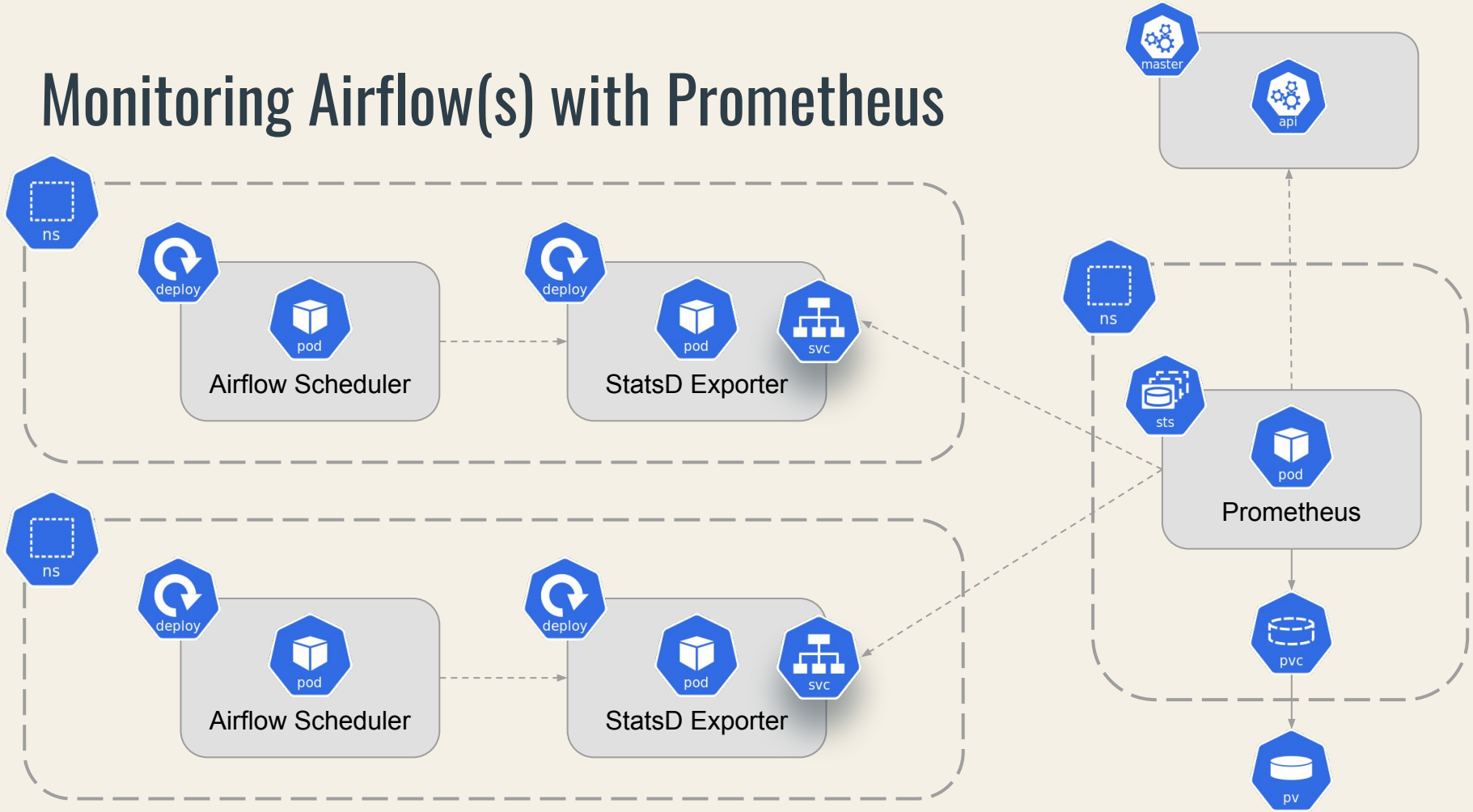

```
annotations:
  prometheus.io/scrape: true
  prometheus.io/port: 9102

labels:
  tier: airflow
  release: {{ .Release.Name }}
```

Monitoring Airflow(s) with Prometheus




```
helm install charts/airflow
```

Monitoring Airflow(s) with Prometheus



1: 2: 3: 9: 0% 12 10% 104°F 1.2 / 1.1 KB/s Enabled 95% 100% Tue Apr 09 8:21:11 PM

Airflow Scheduler - Grafana

grafana.datarouter.ai/d/I6_wigTmk/airflow-scheduler?orgId=1

Airflow Scheduler

Last 30 minutes Refresh every 10s

Deployment

- Selected (1)
- All
- energetic-exposure-4830
- galactic-cosmology-3441
- observational-proton-7027

Ongoing Local Task Jobs

3

Zombies Killed

0

Dagbag Size

2

Collect DAGs

15.5 ms

DAG Bag Import Errors

0

Successful Task Instances

878

Failed Task Instances

0

Operator Success Rate

0.125 ops
0.100 ops
0.075 ops
0.050 ops

Operator Failure Rate

1.0 ops
0.8 ops
0.6 ops
0.4 ops

No data points

1: 2: 3: 9: 0% 12 23% 108°F 1.8 / 2.2 KB/s Enabled 97% 100% Tue Apr 09 8:22:39 PM

Airflow Deployment Overview

grafana.datarouter.ai/d/E5LsXoamz/airflow-deployment-overview?orgId=1&var-deployment=galactic-cosmology-3441

Airflow Deployment Overview

Last 30 minutes Refresh every 10s

Deployment galactic-cosmology-3441

At a Glance

Deployment Status

Healthy

Scheduler Heartbeat

Healthy

Quotas

Max Pods

14 pods

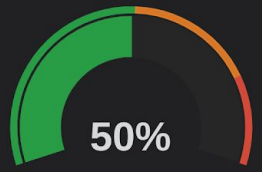
Max CPU

5.4 cores

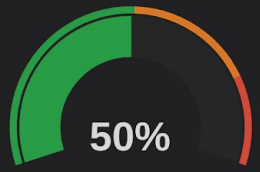
Max Memory

21.7 GB

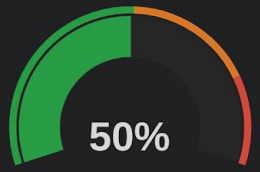
Running Pods



Reserved CPU



Reserved Memory



Scheduler

Scheduler Heartbeat

157151

Dagbag Size

1

Zombies Killed

0

Successful Tasks

Task Success Rate

0.15 ops

Task Failure Rate

1.00 ops

Failed Tasks

1: 2: 3: 9: 0% 12 8% 106°F 0.5 / 0.5 KB/s Enabled 100% 100% Tue Apr 09 8:24:13 PM

Airflow Deployment Overview

grafana.datarouter.ai/d/E5LsXoamz/airflow-deployment-overview?orgId=1&var-deployment=galactic-cosmology-3441

Airflow Deployment Overview

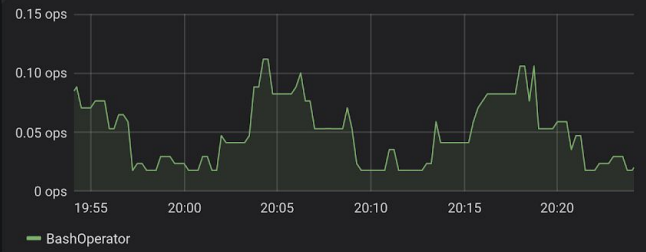
Last 30 minutes Refresh every 10s

Scheduler Heartbeat
157168

Dagbag Size
1

Zombies Killed
0

Successful Tasks
872

Task Success Rate


Task Failure Rate

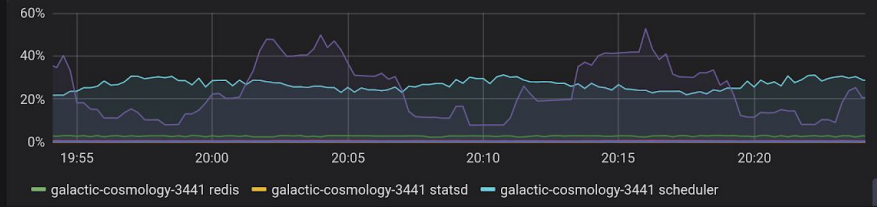

Failed Tasks
0

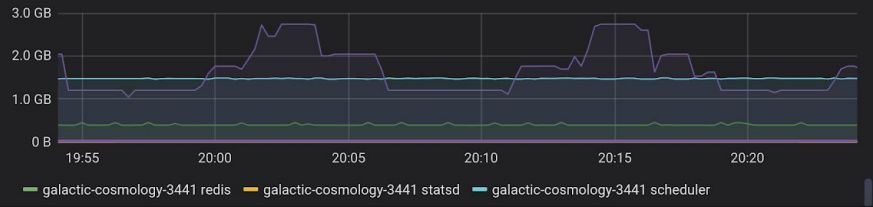
Database Connections

Total Database Connections
7

Total Waiting Clients
0

Resource Utilization

CPU Usage


Memory Usage


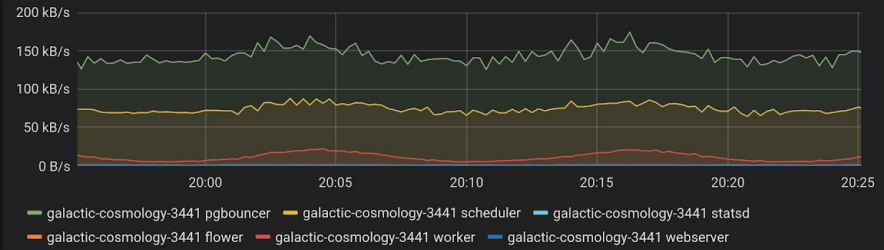
Airflow Deployment Overview

Last 30 minutes Refresh every 10s

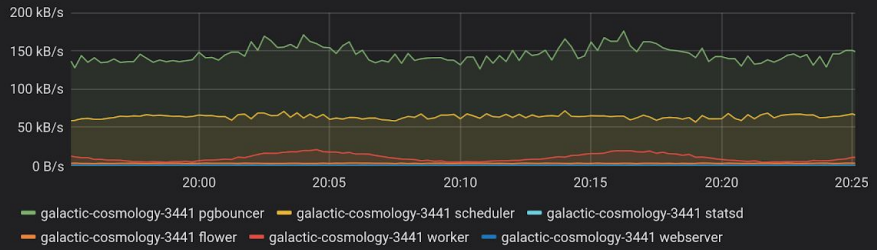
galactic-cosmology-3441 redis galactic-cosmology-3441 statsd galactic-cosmology-3441 scheduler galactic-cosmology-3441 scheduler-gc galactic-cosmology-3441 pgbouncer galactic-cosmology-3441 pgbouncer-metrics galactic-cosmology-3441 flower galactic-cosmology-3441 worker galactic-cosmology-3441 webserver

galactic-cosmology-3441 redis galactic-cosmology-3441 statsd galactic-cosmology-3441 scheduler galactic-cosmology-3441 scheduler-gc galactic-cosmology-3441 pgbouncer galactic-cosmology-3441 pgbouncer-metrics galactic-cosmology-3441 flower galactic-cosmology-3441 worker galactic-cosmology-3441 webserver

Network Rx



Network Tx



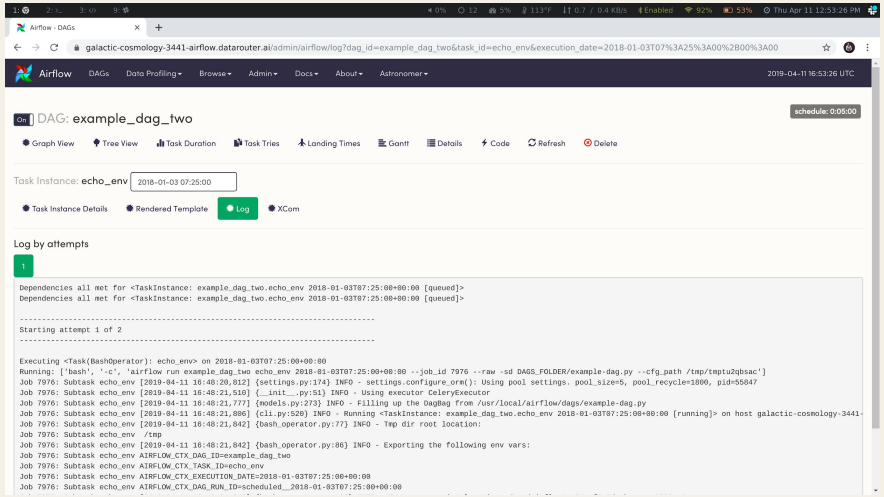
Container Status

Container Status

Container	Pod	Status
galactic-cosmology-3441-redis	galactic-cosmology-3441-redis-0	Healthy
galactic-cosmology-3441-scheduler	galactic-cosmology-3441-scheduler-55fcf74956-bgggp	Healthy
galactic-cosmology-3441-pgbouncer	galactic-cosmology-3441-pgbouncer-55874674fc-fslcj	Healthy
galactic-cosmology-3441-statsd	galactic-cosmology-3441-statsd-7656dc98f5-9cng2	Healthy
galactic-cosmology-3441-flower	galactic-cosmology-3441-flower-7b7c9f8598-gcv7l	Healthy
galactic-cosmology-3441-webserver	galactic-cosmology-3441-webserver-dd858849d-knw7m	Healthy
galactic-cosmology-3441-pgbouncer-metrics	galactic-cosmology-3441-pgbouncer-55874674fc-fslcj	Healthy
galactic-cosmology-3441-scheduler-gc	galactic-cosmology-3441-scheduler-55fcf74956-bgggp	Healthy
galactic-cosmology-3441-worker	galactic-cosmology-3441-worker-85f6f4d96b-jld7z	Healthy

Airflow Logging

- Powers the task log view in Airflow UI
- KubernetesExecutor requires remote logging plugin
- Several remote logging backend plugins available
  - Object Storage (S3, GCS, WASB)
  - Elasticsearch



1: 2: 3: 9: 0% 12 5% 113°F 0.7 / 0.4 KB/s Enabled 92% 53% Thu Apr 11 12:53:26 PM

Airflow - DAGs

galactic-cosmology-3441-airflow.datarouter.ai/admin/airflow/log?dag_id=example_dag_two&task_id=echo_env&execution_date=2018-01-03T07%3A25%3A00%2B00%3A00

Airflow DAGs Data Profiling Browse Admin Docs About Astronomer 2019-04-11 16:53:26 UTC

On DAG: example_dag_two

schedule: 0:05:00

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Refresh Delete

Task Instance: echo_env 2018-01-03 07:25:00

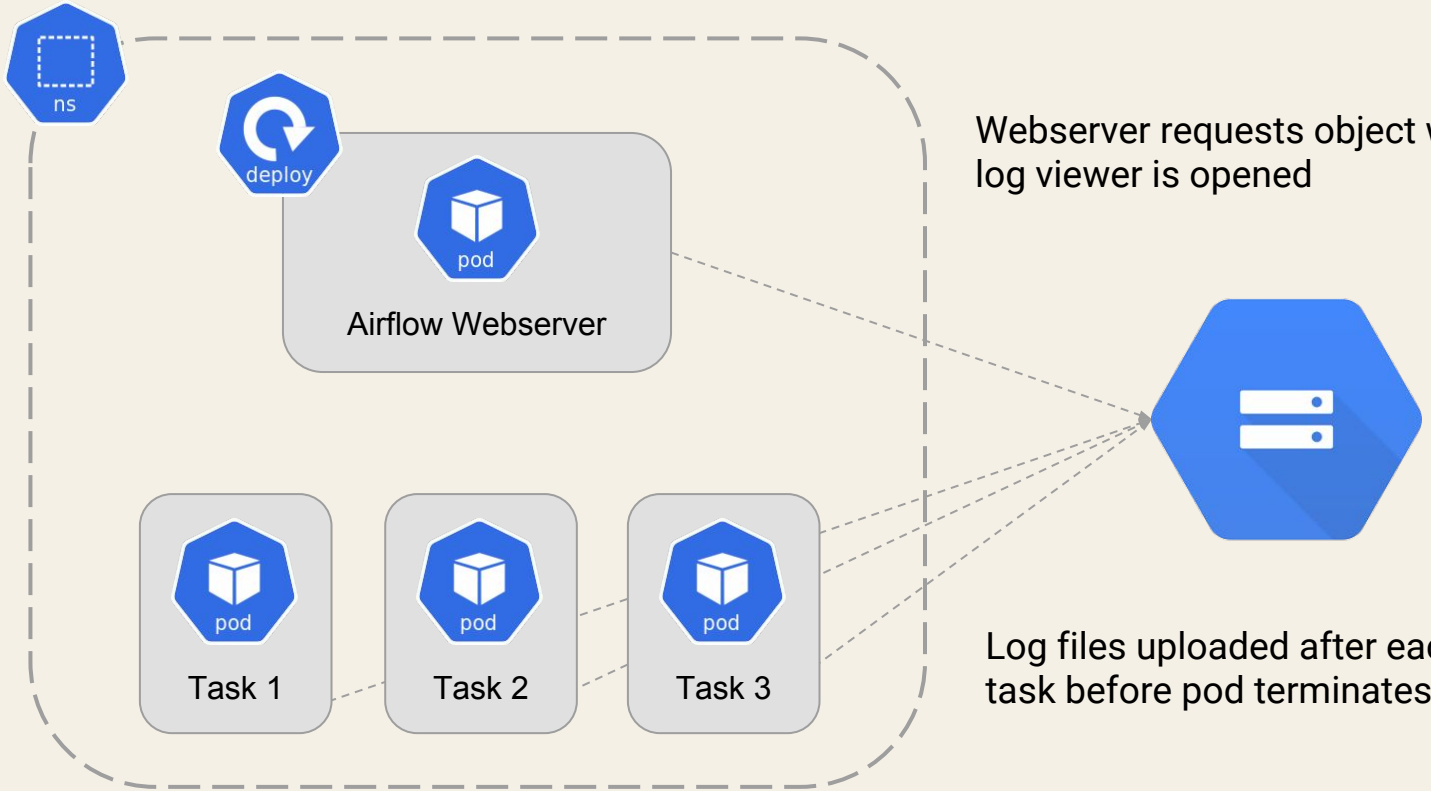
Task Instance Details Rendered Template Log XCom

Log by attempts

1

```
Dependencies all met for <TaskInstance: example_dag_two.echo_env 2018-01-03T07:25:00+00:00 [queued]>
Dependencies all met for <TaskInstance: example_dag_two.echo_env 2018-01-03T07:25:00+00:00 [queued]>

--------------------------------------------------------------------------------
Starting attempt 1 of 2
--------------------------------------------------------------------------------

Executing <Task(BashOperator): echo_env> on 2018-01-03T07:25:00+00:00
Running: ['bash', '-c', 'airflow run example_dag_two echo_env 2018-01-03T07:25:00+00:00 --job_id 7976 --raw -sd DAGS_FOLDER/example-dag.py --cfg_path /tmp/tmptu2qbsac']
Job 7976: Subtask echo_env [2019-04-11 16:48:20,812] {settings.py:174} INFO - settings.configure_orm(): Using pool settings. pool_size=5, pool_recycle=1800, pid=55847
Job 7976: Subtask echo_env [2019-04-11 16:48:21,510] {__init__.py:51} INFO - Using executor CeleryExecutor
Job 7976: Subtask echo_env [2019-04-11 16:48:21,777] {models.py:273} INFO - Filling up the DagBag from /usr/local/airflow/dags/example-dag.py
Job 7976: Subtask echo_env [2019-04-11 16:48:21,806] {cli.py:520} INFO - Running <TaskInstance: example_dag_two.echo_env 2018-01-03T07:25:00+00:00 [running]> on host galactic-cosmology-3441-
Job 7976: Subtask echo_env [2019-04-11 16:48:21,842] {bash_operator.py:77} INFO - Tmp dir root location:
Job 7976: Subtask echo_env  /tmp
Job 7976: Subtask echo_env [2019-04-11 16:48:21,842] {bash_operator.py:86} INFO - Exporting the following env vars:
Job 7976: Subtask echo_env AIRFLOW_CTX_DAG_ID=example_dag_two
Job 7976: Subtask echo_env AIRFLOW_CTX_TASK_ID=echo_env
Job 7976: Subtask echo_env AIRFLOW_CTX_EXECUTION_DATE=2018-01-03T07:25:00+00:00
Job 7976: Subtask echo_env AIRFLOW_CTX_DAG_RUN_ID=scheduled__2018-01-03T07:25:00+00:00
```

Airflow Logging - Object Storage



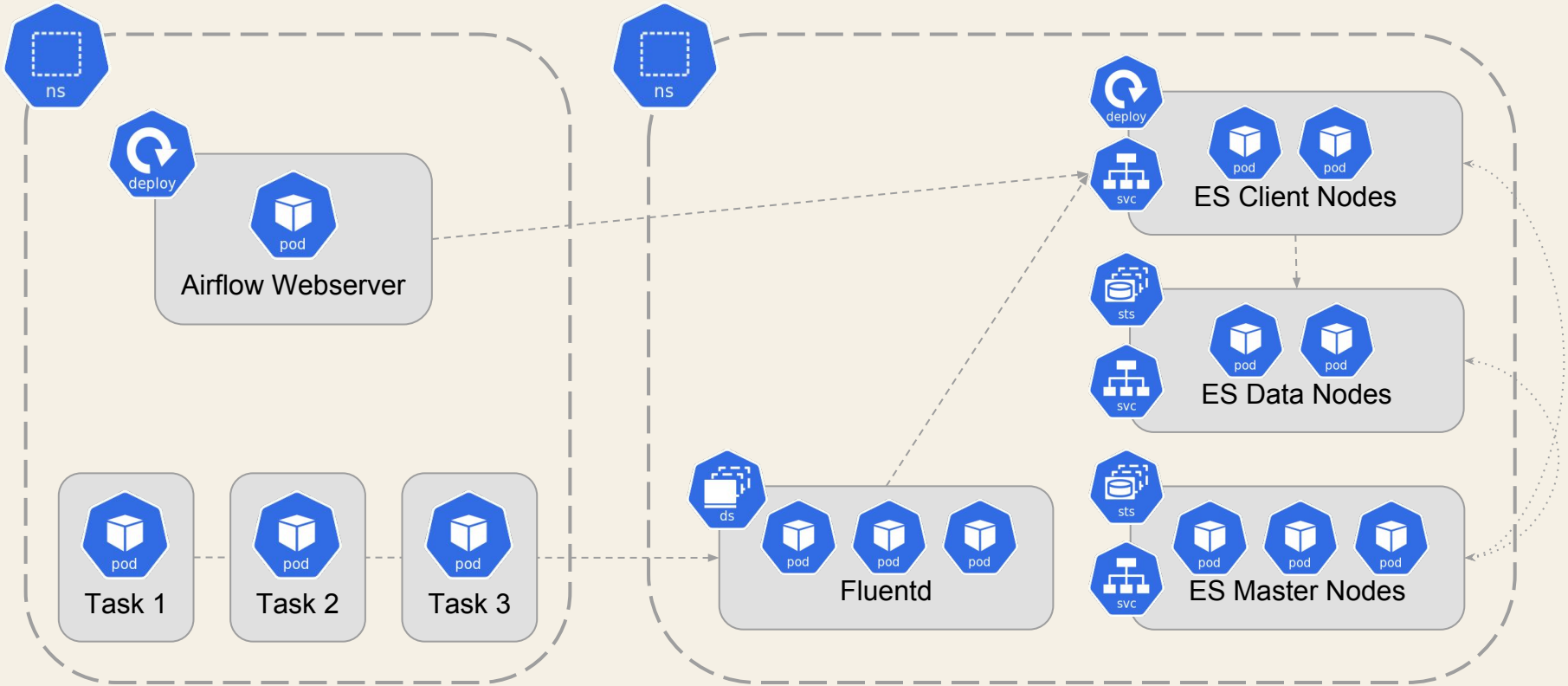
Webserver requests object when log viewer is opened

Log files uploaded after each task before pod terminates

Airflow Logging - Elasticsearch

```
helm install stable/elasticsearch

helm install stable/fluentd
```

Airflow Logging - Elasticsearch



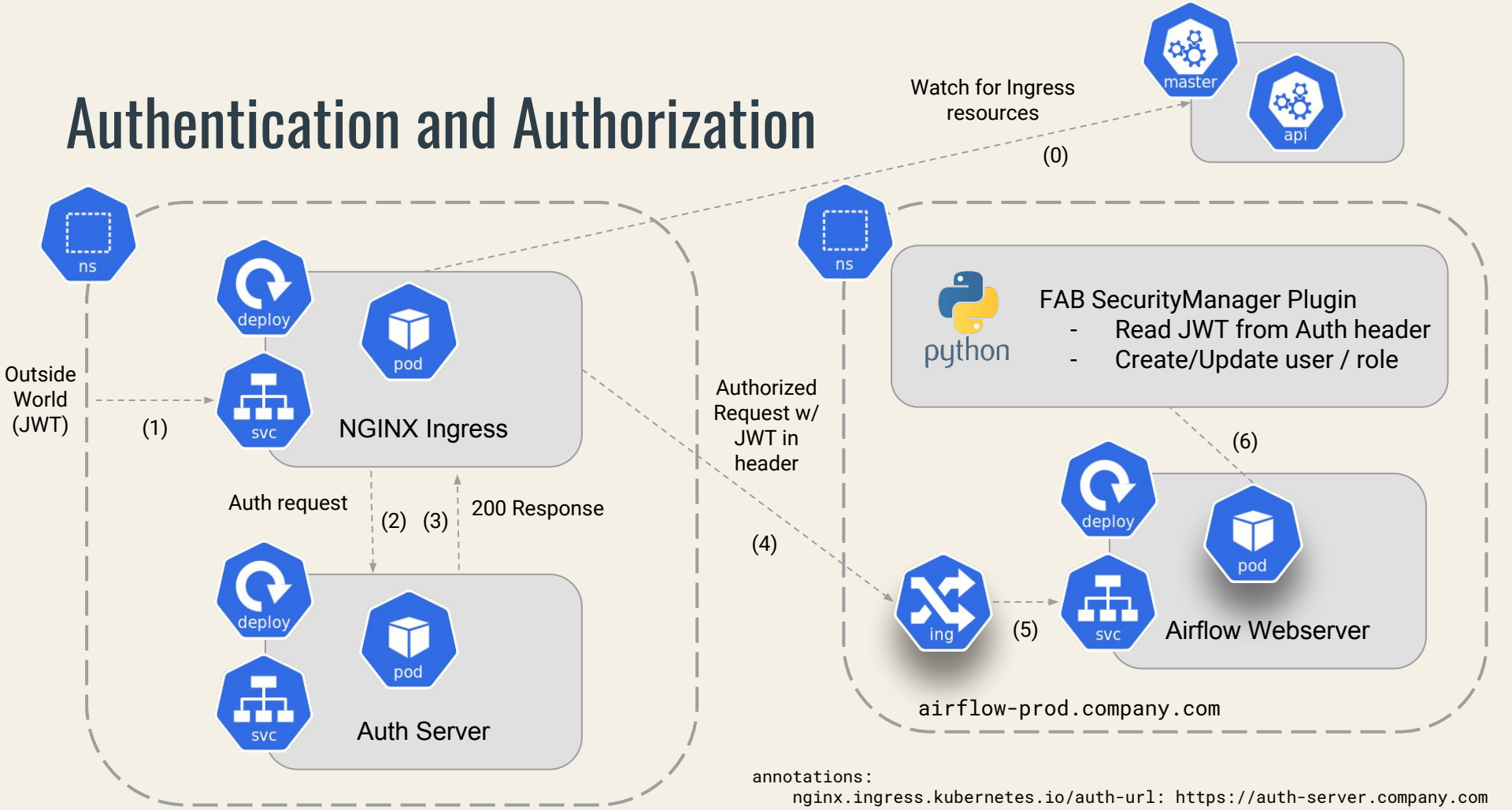
AIRFLOW-3370 - https://issues.apache.org/jira/browse/AIRFLOW-3370

Authentication and Authorization

- Ingress Controllers
  - Exposes a Kubernetes service to the outside world
  - Fulfulls Kubernetes Ingress resources

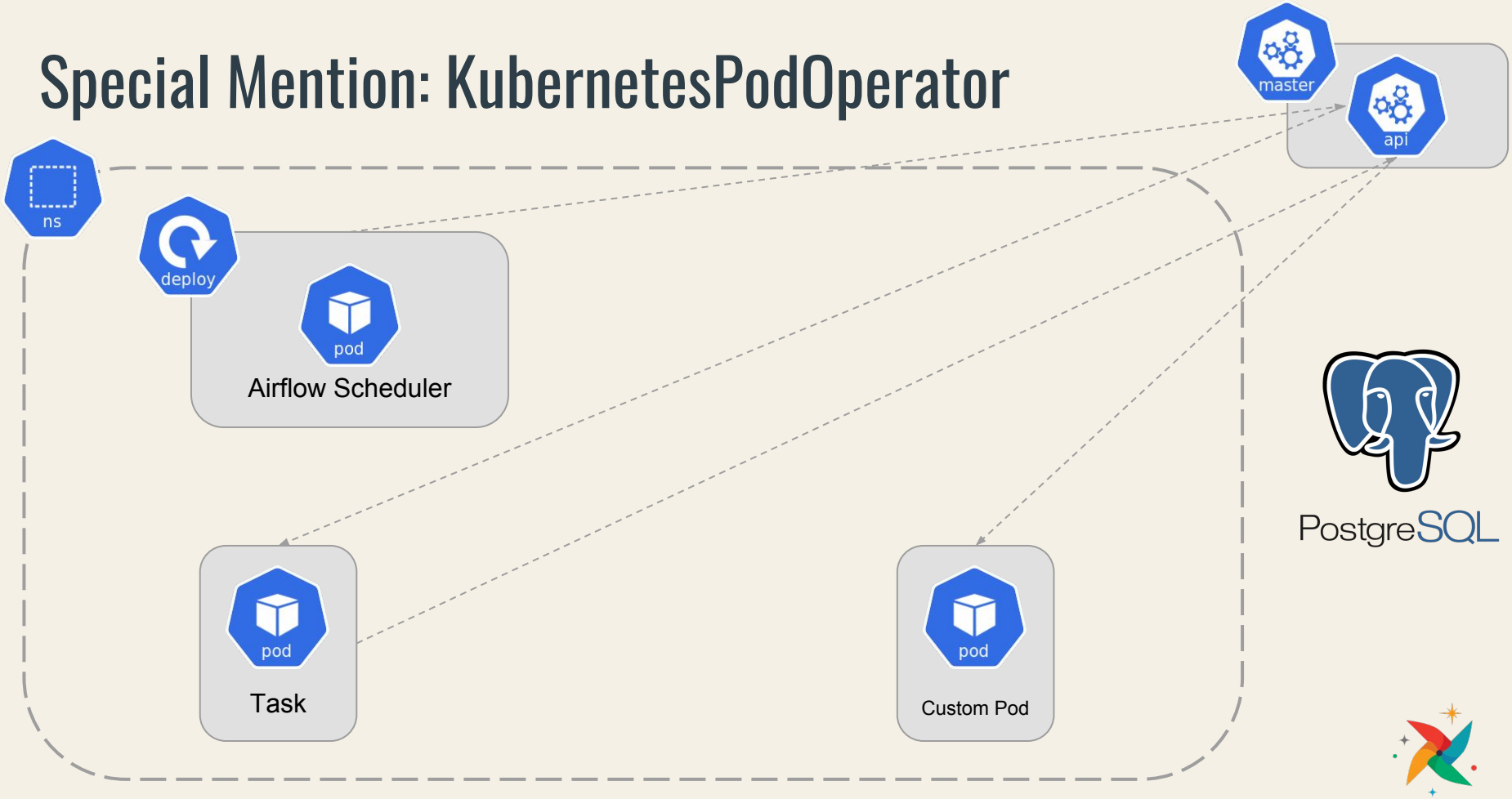
```
helm install stable/nginx-ingress
```

Authentication and Authorization




```
annotations:
    nginx.ingress.kubernetes.io/auth-url: https://auth-server.company.com
```

Special Mention: KubernetesPodOperator



Thank you!

greg@astronomer.io