KubeCon | CloudNativeCon

North America 2018

Airflow + Kubernetes

Daniel Imberman, Bloomberg
Barni Seetharaman, Google

Bio: Daniel + Barni

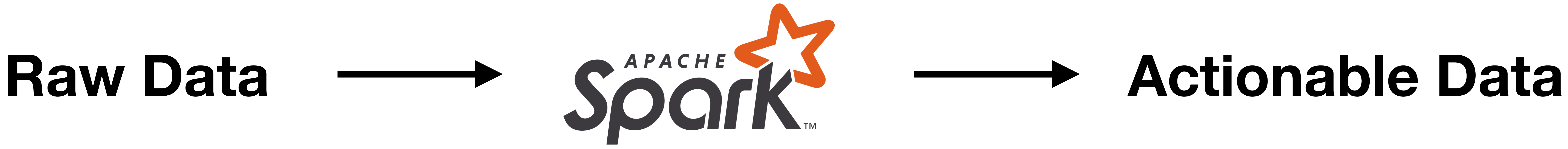
- Daniel
  - Data Science Infrastructure @ Bloomberg LP
  - See our talk tomorrow: Machine Learning the Kubernetes Way
- Barni Seethraman
  - Kubernetes @ Google Cloud
  - Works on Kubernetes Workloads API

Pipelines are hard

Pipelines are hard

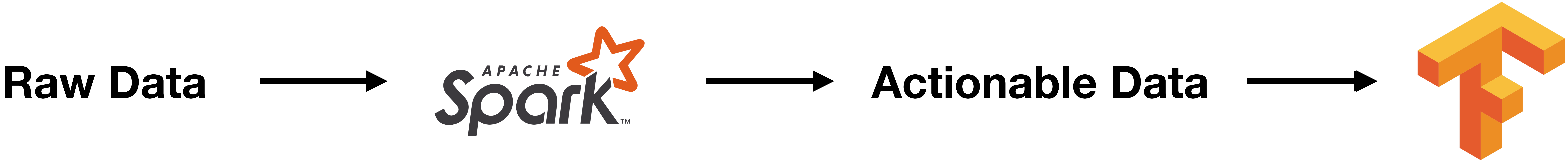
Raw Data → **Actionable Data**

Pipelines are hard



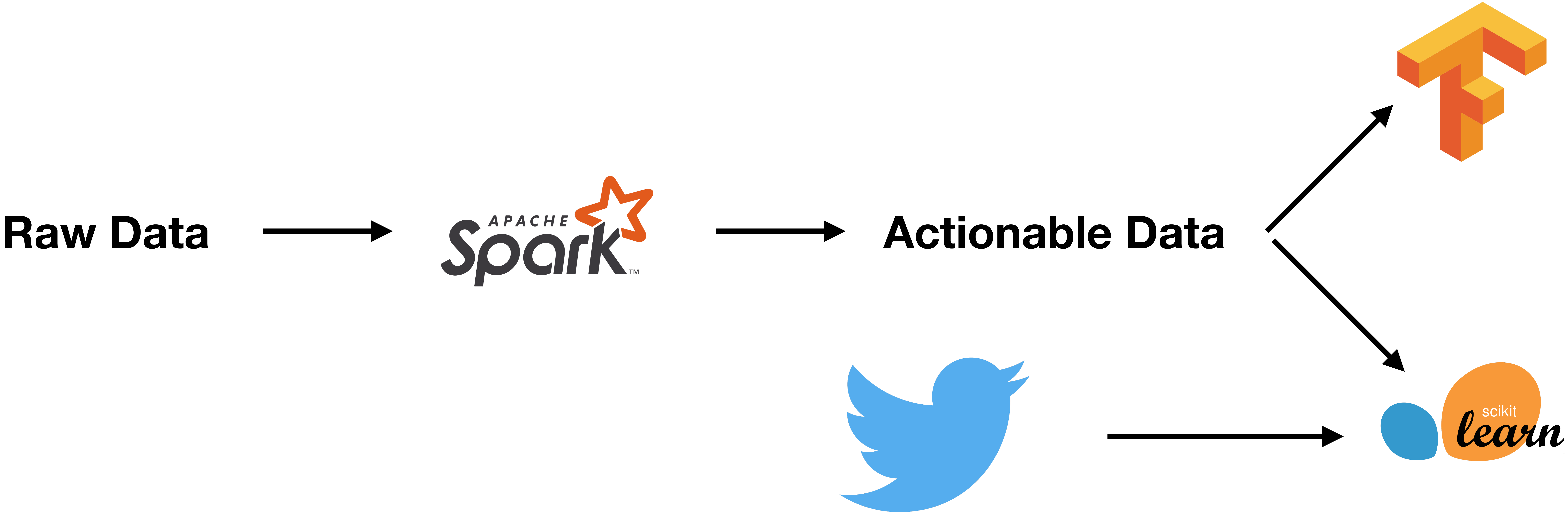
Raw Data → Apache Spark → Actionable Data

Pipelines are hard



Raw Data → Apache Spark → Actionable Data →

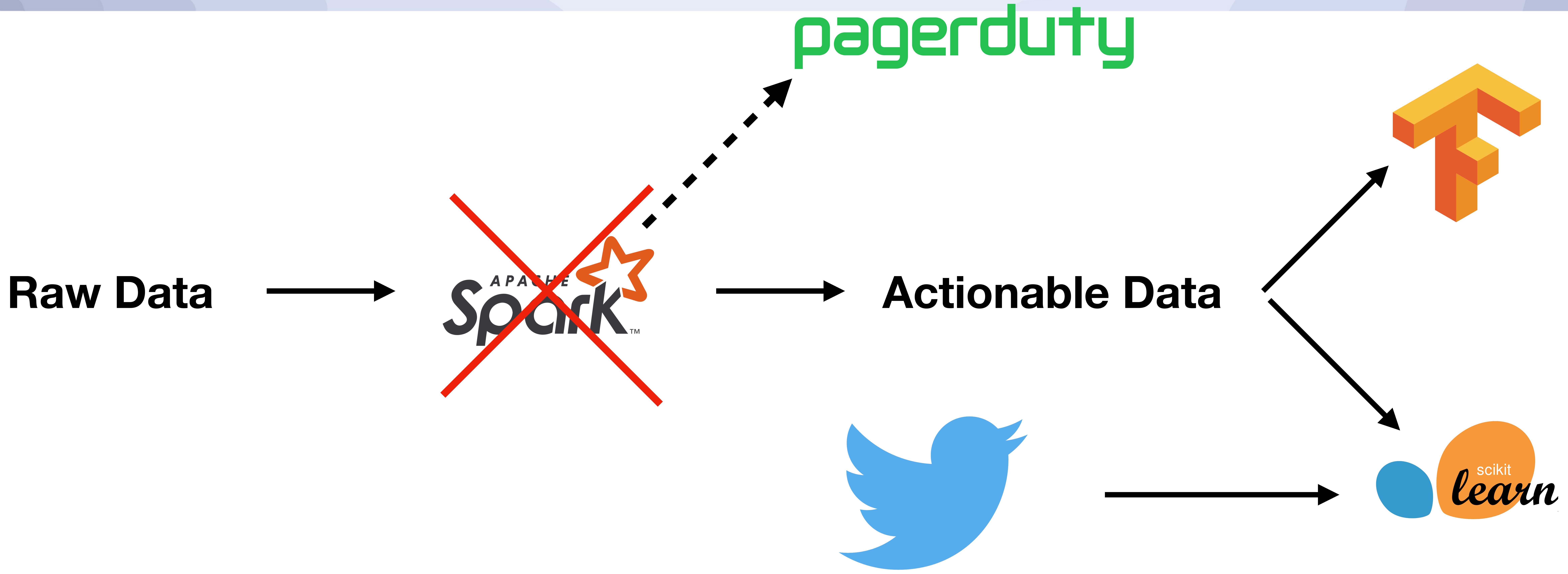
Pipelines are hard



Raw Data → Actionable Data

TWITTER, TWEET, RETWEET and the Twitter logo are trademarks of Twitter, Inc. or its affiliates.

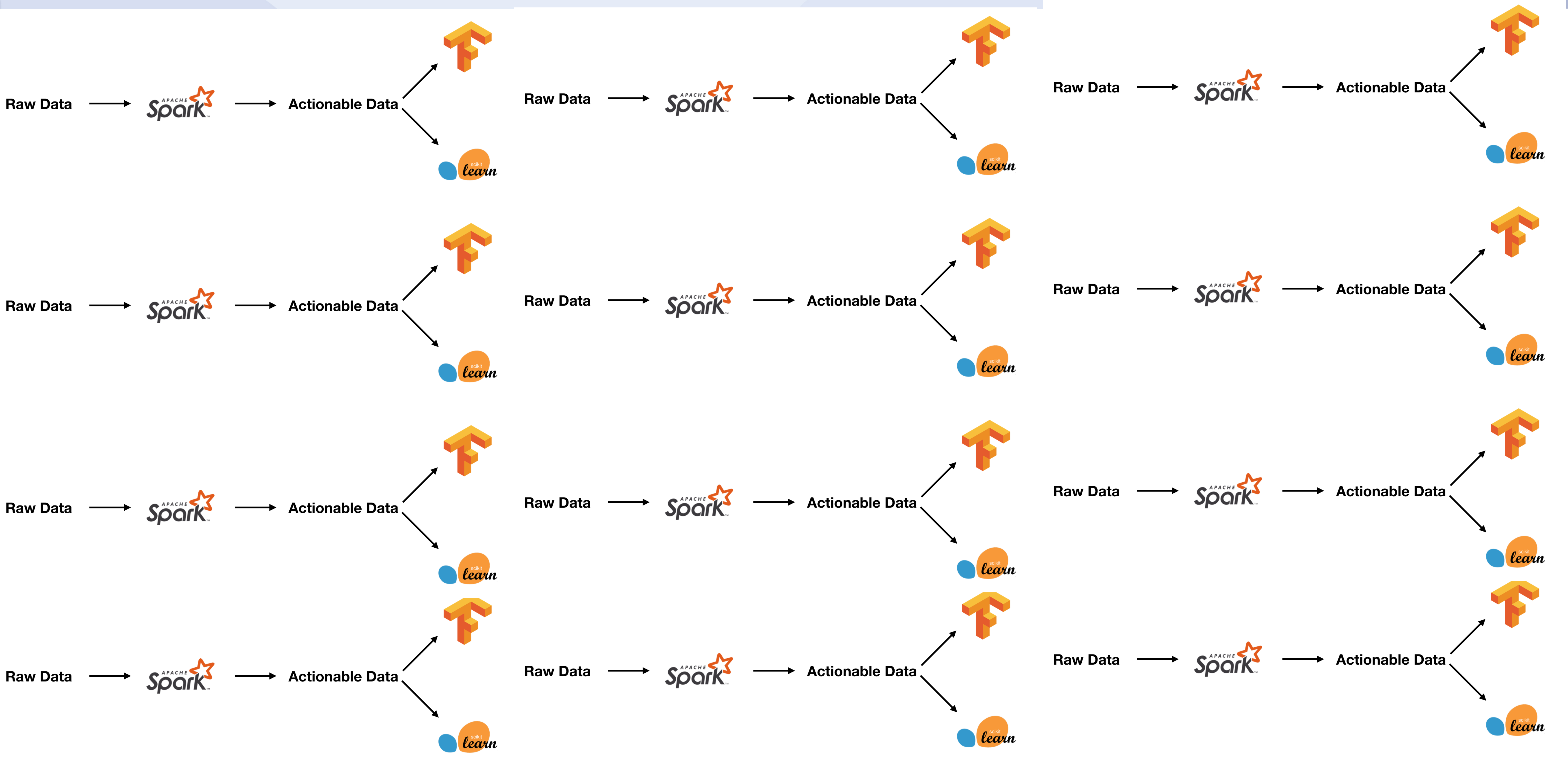
Pipelines are hard



Raw Data

Actionable Data

Lots of pipelines are really hard



Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

Raw Data → Spark → Actionable Data

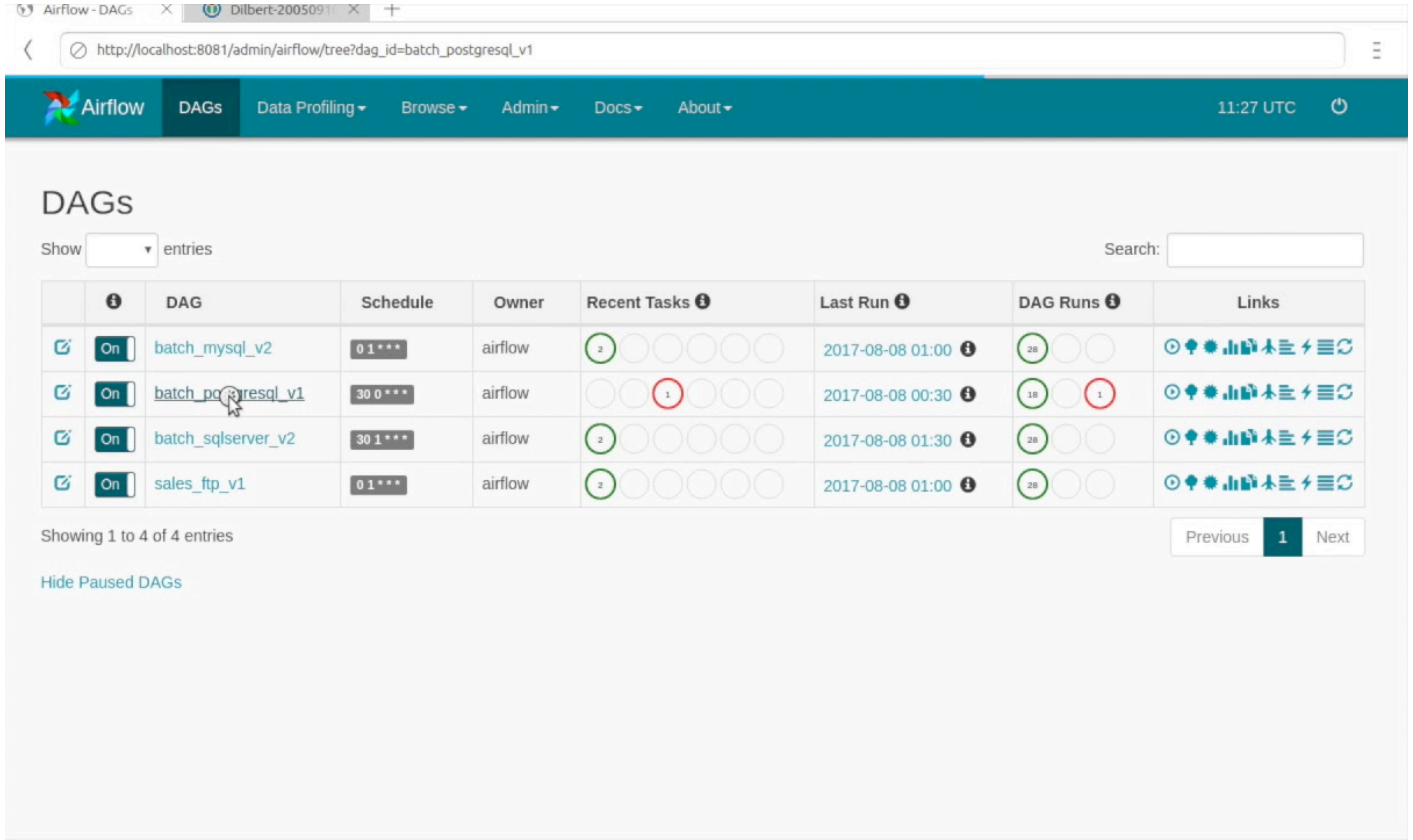
Raw Data → Spark → Actionable Data

Enter Apache Airflow

Apache Airflow

- Workflow Scheduler developed @ Airbnb
- Converts Python code into DAGs
- Has large number of operators/hooks (HDFS, Spark, Bash, Hive, etc…)

Apache Airflow



Airflow - DAGs | Dilbert-200509

http://localhost:8081/admin/airflow/tree?dag_id=batch_postgresql_v1

DAGs

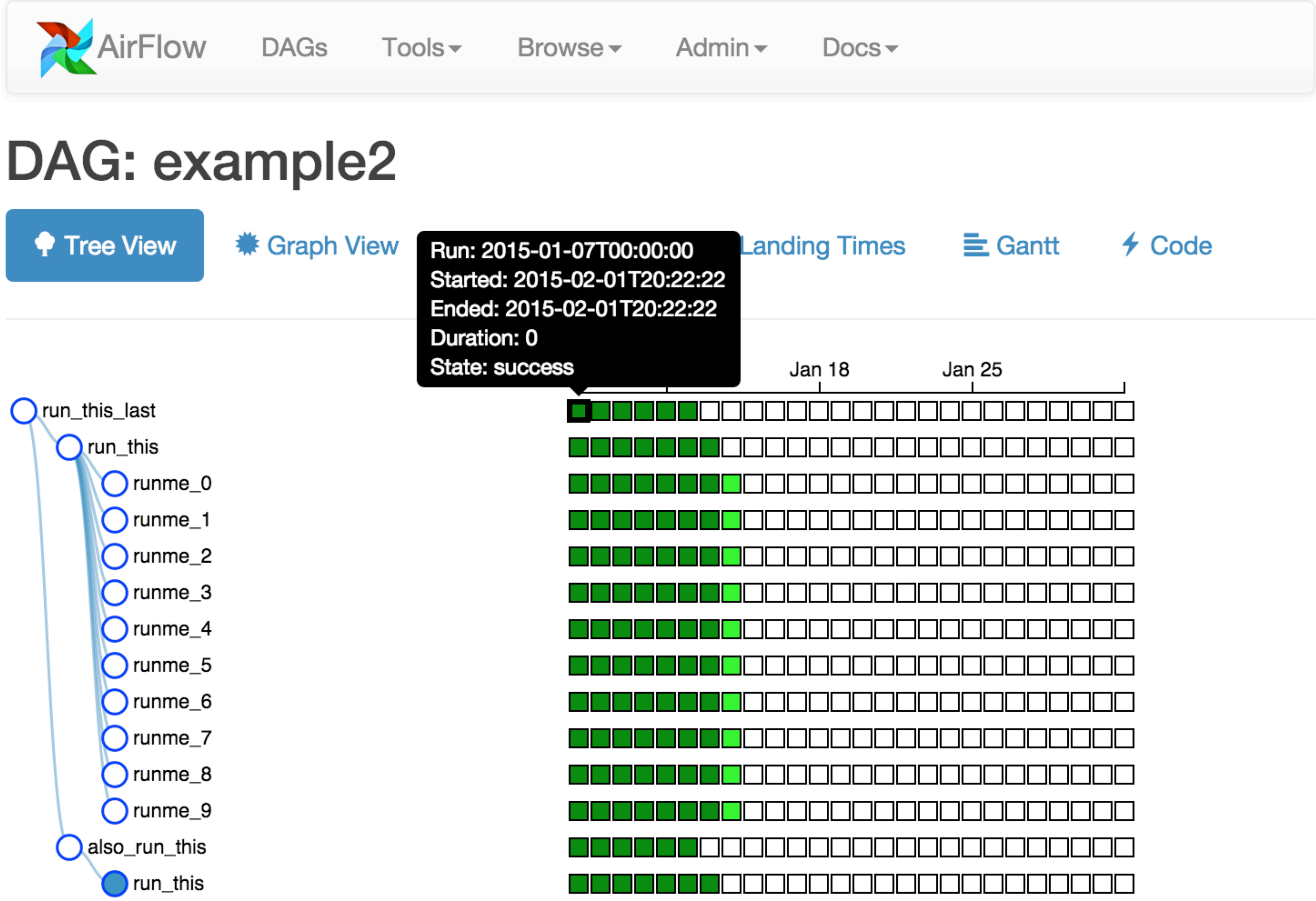
Show entries | Search:

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	On	batch_mysql_v2	0 1 * * *	airflow	2	2017-08-08 01:00	28	
	On	batch_postgresql_v1	30 0 * * *	airflow	1	2017-08-08 00:30	18 1	
	On	batch_sqlserver_v2	30 1 * * *	airflow	2	2017-08-08 01:30	28	
	On	sales_ftp_v1	0 1 * * *	airflow	2	2017-08-08 01:00	28	

Showing 1 to 4 of 4 entries

Hide Paused DAGs

Apache Airflow

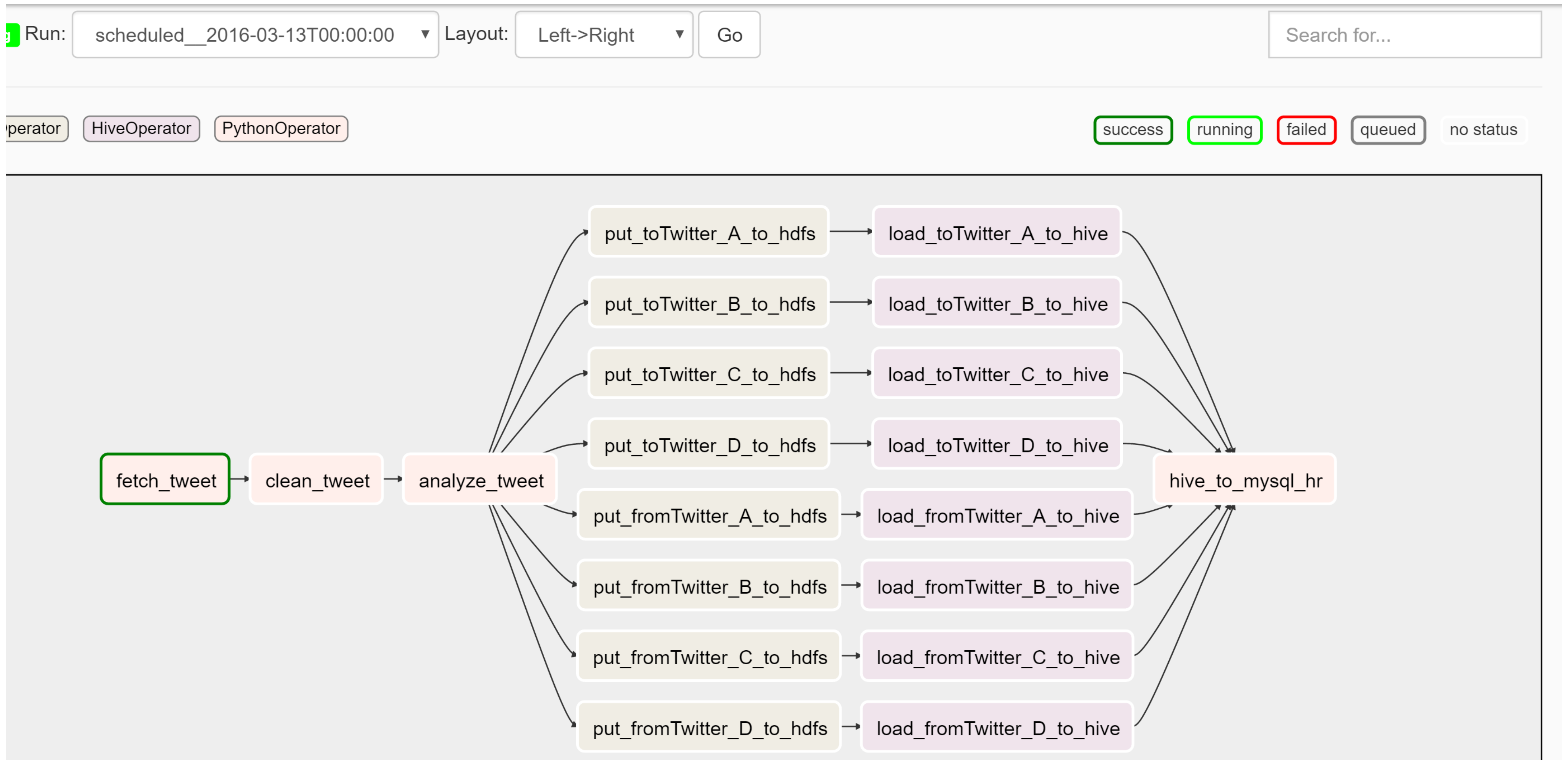


AirFlow DAGs Tools▾ Browse▾ Admin▾ Docs▾

DAG: example2

Tree View | Graph View | Landing Times | Gantt | Code

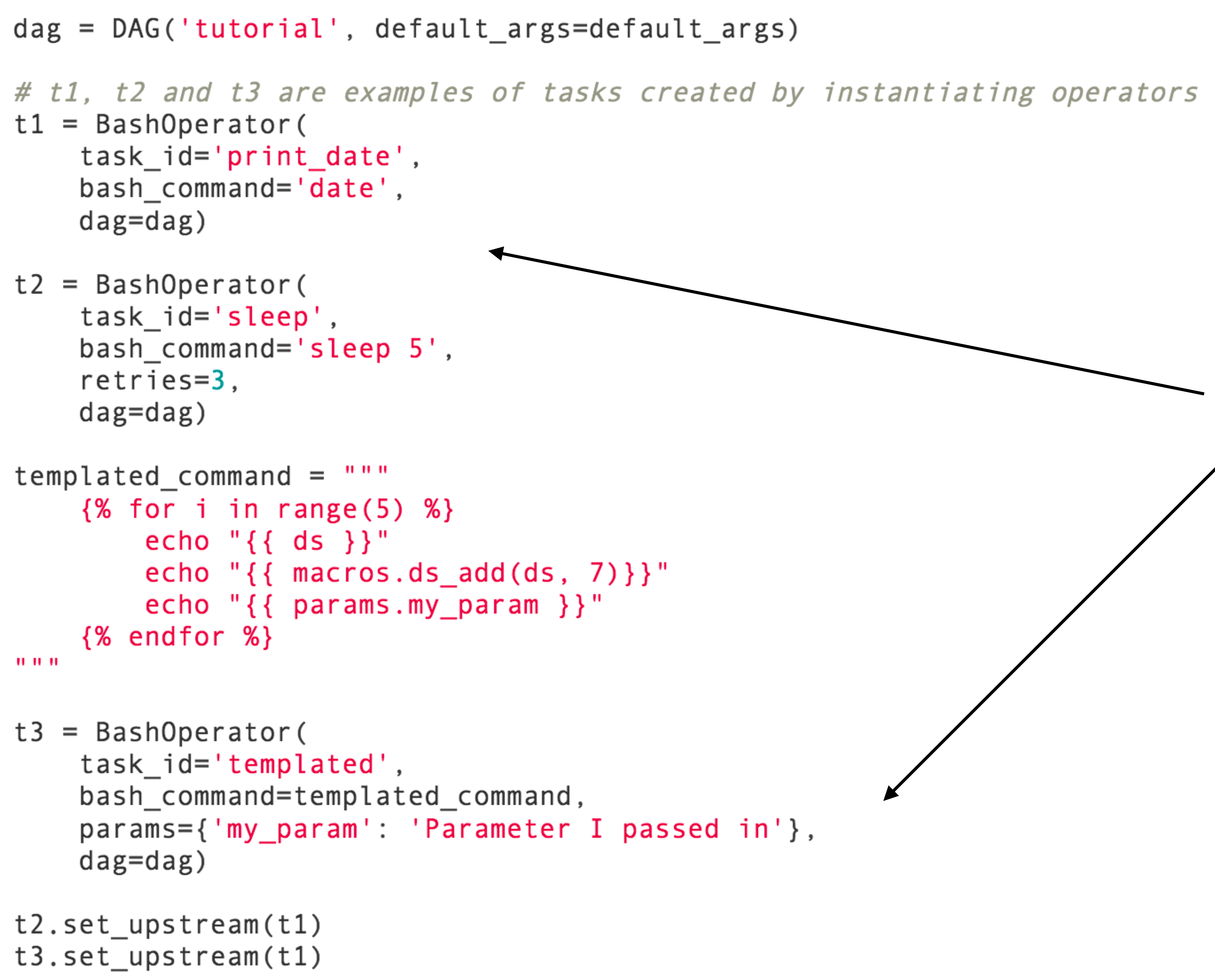
Run: 2015-01-07T00:00:00
Started: 2015-02-01T20:22:22
Ended: 2015-02-01T20:22:22
Duration: 0
State: success



Run: scheduled__2016-03-13T00:00:00 ▾ Layout: Left->Right ▾ Go

Search for...

Creating a pipeline with Airflow

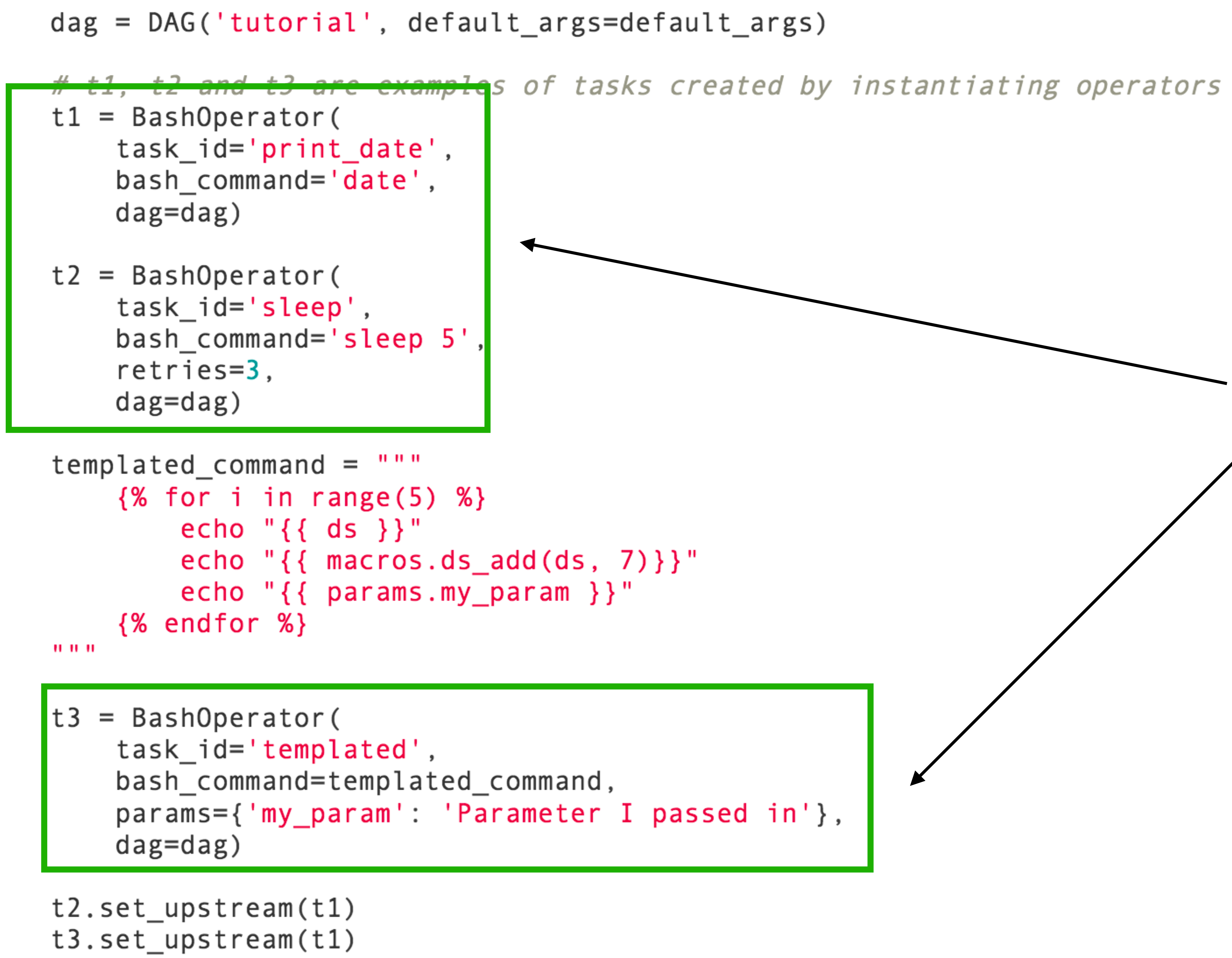


```
dag = DAG('tutorial', default_args=default_args)

# t1, t2 and t3 are examples of tasks created by instantiating operators
t1 = BashOperator(
    task_id='print_date',
    bash_command='date',
    dag=dag)

t2 = BashOperator(
    task_id='sleep',
    bash_command='sleep 5',
    retries=3,
    dag=dag)

templated_command = """
    {% for i in range(5) %}
        echo "{{ ds }}"
        echo "{{ macros.ds_add(ds, 7)}}"
        echo "{{ params.my_param }}"
    {% endfor %}
"""

t3 = BashOperator(
    task_id='templated',
    bash_command=templated_command,
    params={'my_param': 'Parameter I passed in'},
    dag=dag)

t2.set_upstream(t1)
t3.set_upstream(t1)
```

Define Operators

Set Dependencies

Creating a pipeline with Airflow



```
dag = DAG('tutorial', default_args=default_args)

# t1, t2 and t3 are examples of tasks created by instantiating operators
t1 = BashOperator(
    task_id='print_date',
    bash_command='date',
    dag=dag)

t2 = BashOperator(
    task_id='sleep',
    bash_command='sleep 5',
    retries=3,
    dag=dag)

templated_command = """
    {% for i in range(5) %}
        echo "{{ ds }}"
        echo "{{ macros.ds_add(ds, 7)}}"
        echo "{{ params.my_param }}"
    {% endfor %}
"""

t3 = BashOperator(
    task_id='templated',
    bash_command=templated_command,
    params={'my_param': 'Parameter I passed in'},
    dag=dag)

t2.set_upstream(t1)
t3.set_upstream(t1)
```

Define Operators

Set Dependencies

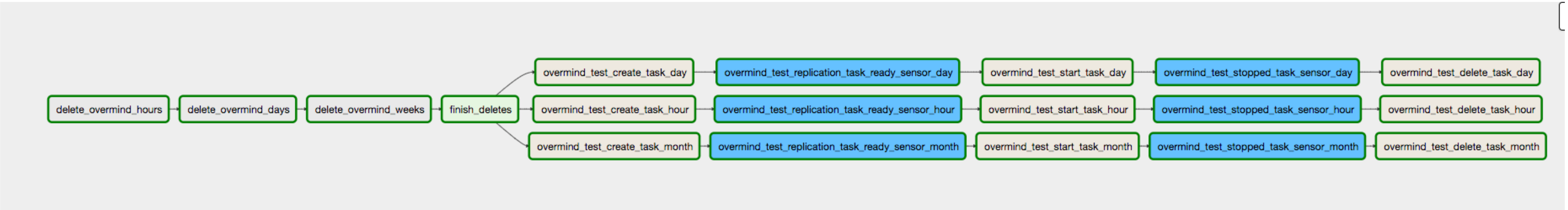
Creating a pipeline with Airflow

```
dag = DAG('tutorial', default_args=default_args)

# t1, t2 and t3 are examples of tasks created by instantiating operators
t1 = BashOperator(
    task_id='print_date',
    bash_command='date',
    dag=dag)

t2 = BashOperator(
    task_id='sleep',
    bash_command='sleep 5',
    retries=3,
    dag=dag)

templated_command = """
    {% for i in range(5) %}
        echo "{{ ds }}"
        echo "{{ macros.ds_add(ds, 7)}}"
        echo "{{ params.my_param }}"
    {% endfor %}
"""

t3 = BashOperator(
    task_id='templated',
    bash_command=templated_command,
    params={'my_param': 'Parameter I passed in'},
    dag=dag)

t2.set_upstream(t1)
t3.set_upstream(t1)
```

Define Operators

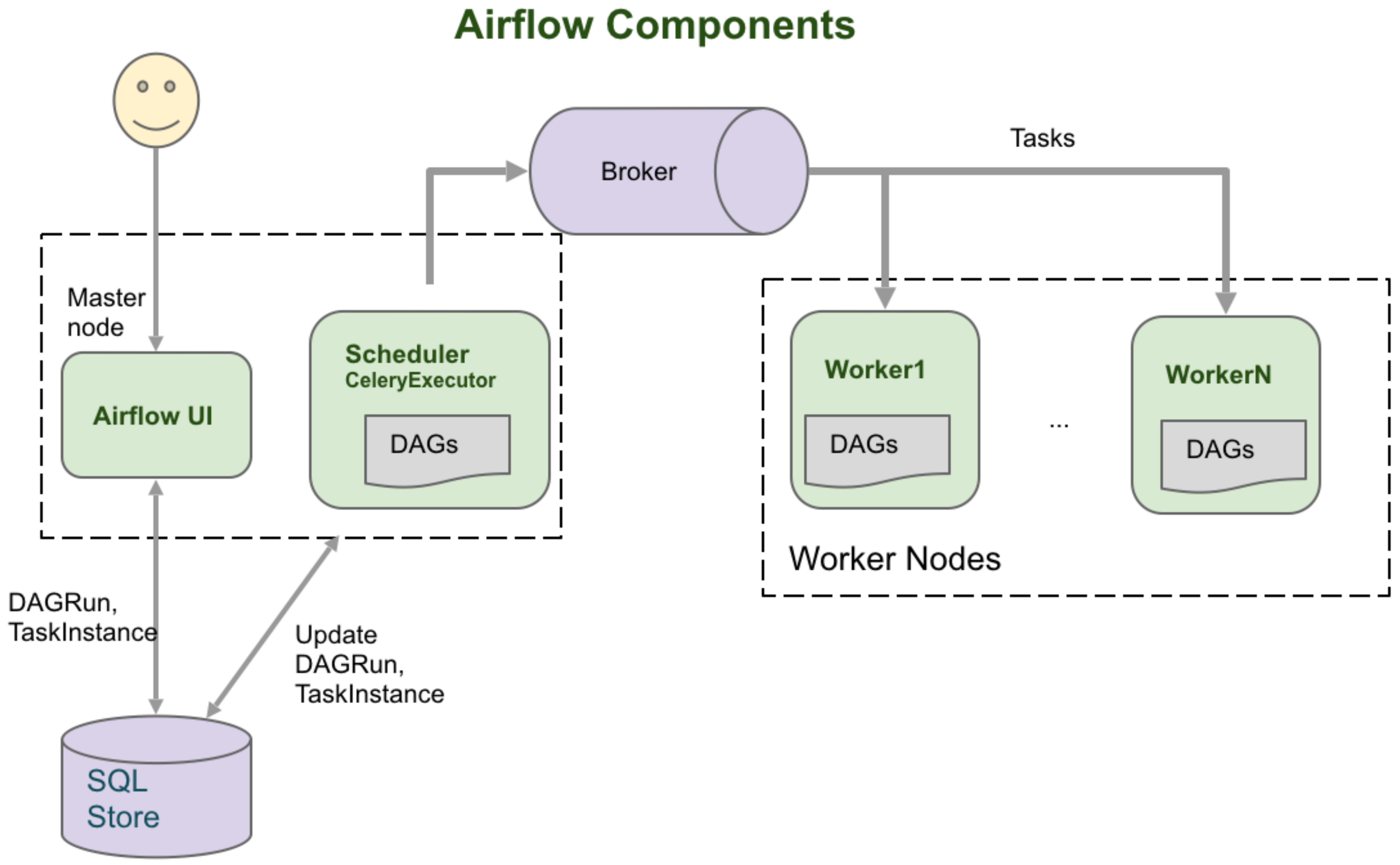
Set Dependencies

Managing a pipeline with Airflow

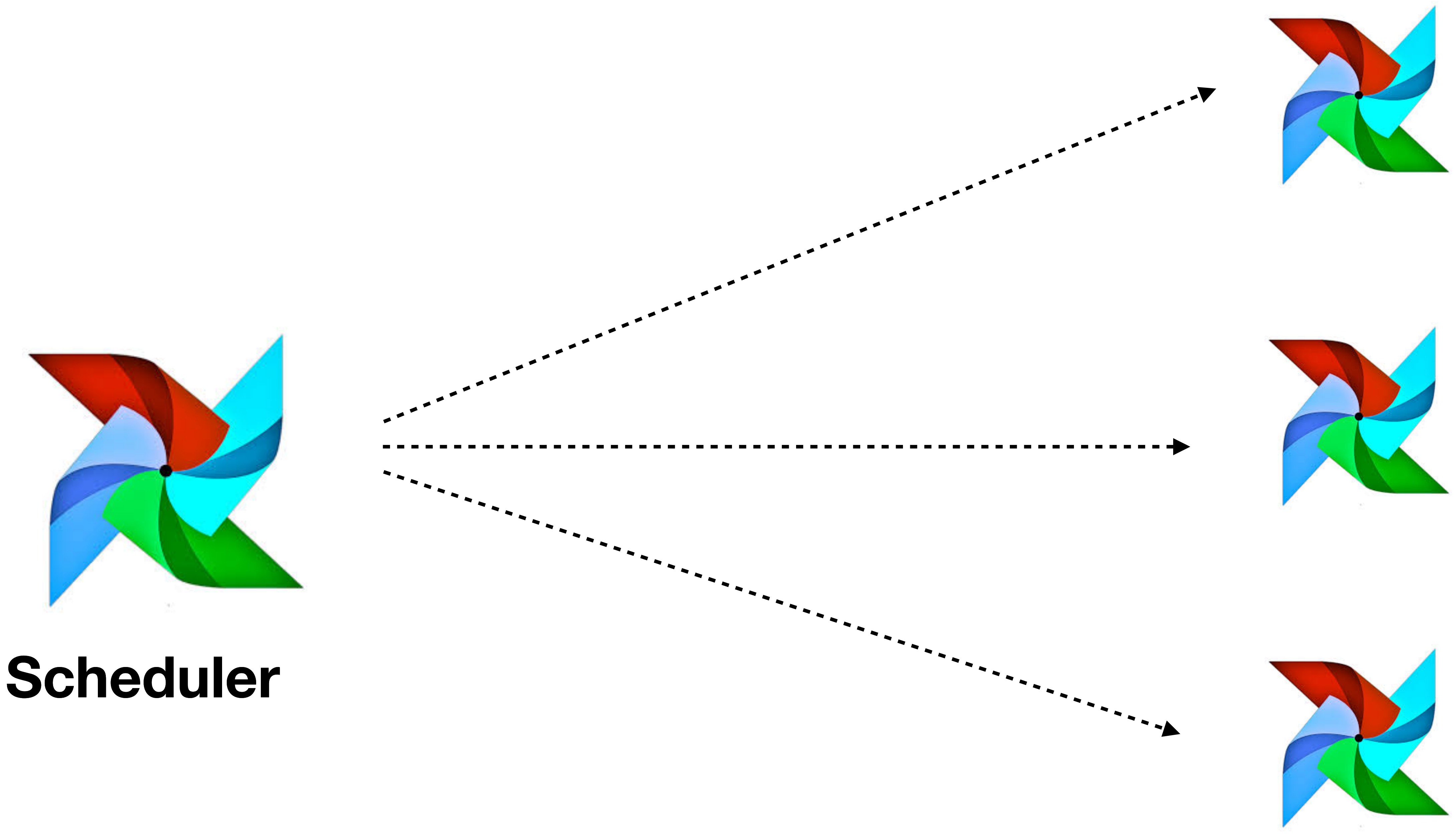

delete_overmind_hours
delete_overmind_days
delete_overmind_weeks
finish_deletes
overmind_test_create_task_day
overmind_test_replication_task_ready_sensor_day
overmind_test_start_task_day
overmind_test_stopped_task_sensor_day
overmind_test_delete_task_day
overmind_test_create_task_hour
overmind_test_replication_task_ready_sensor_hour
overmind_test_start_task_hour
overmind_test_stopped_task_sensor_hour
overmind_test_delete_task_hour
overmind_test_create_task_month
overmind_test_replication_task_ready_sensor_month
overmind_test_start_task_month
overmind_test_stopped_task_sensor_month
overmind_test_delete_task_month

State of things: Complexity

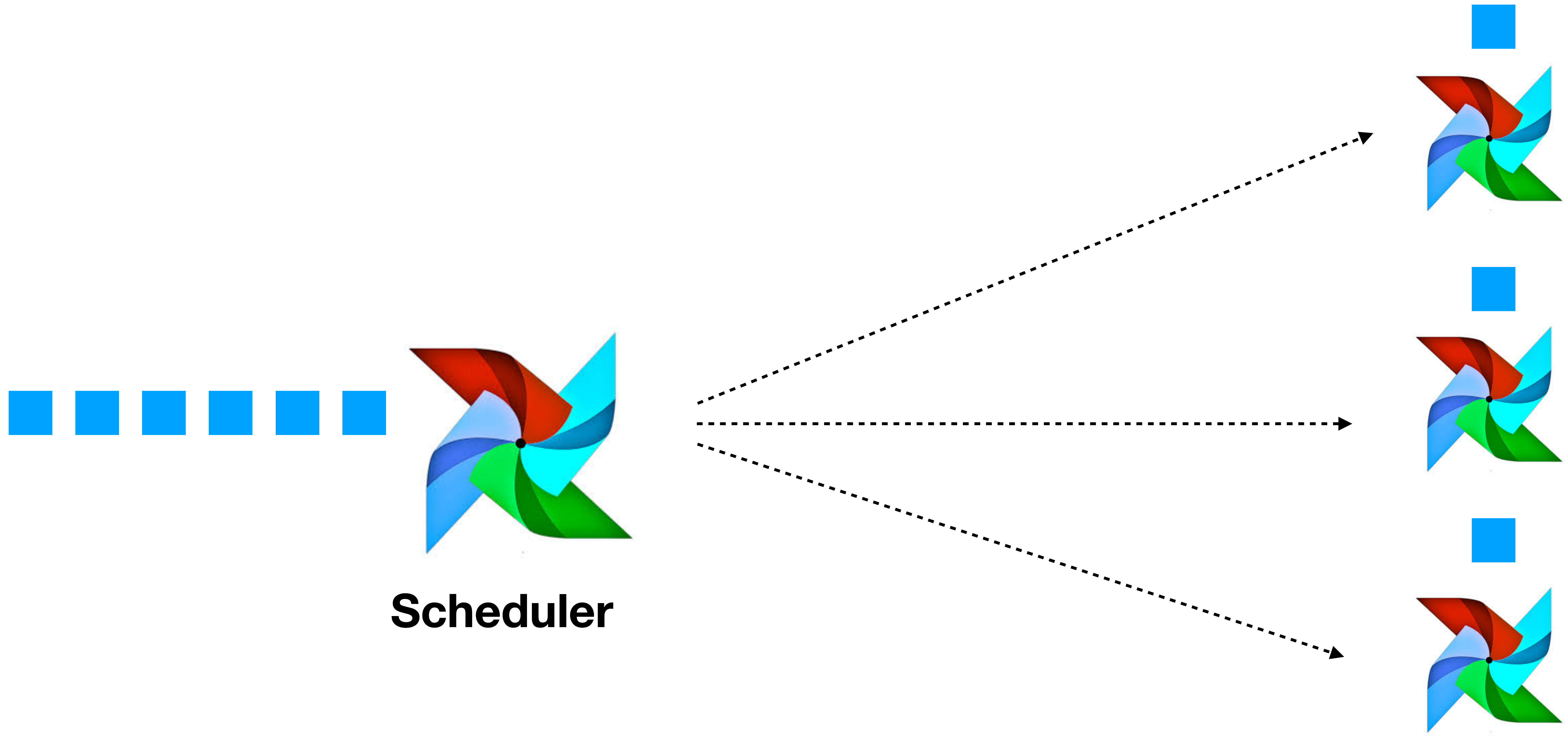
- Complex to deploy and manage
- Multiple components
- Varied configurations
    - Executor
    - DAG source
- Different failure points



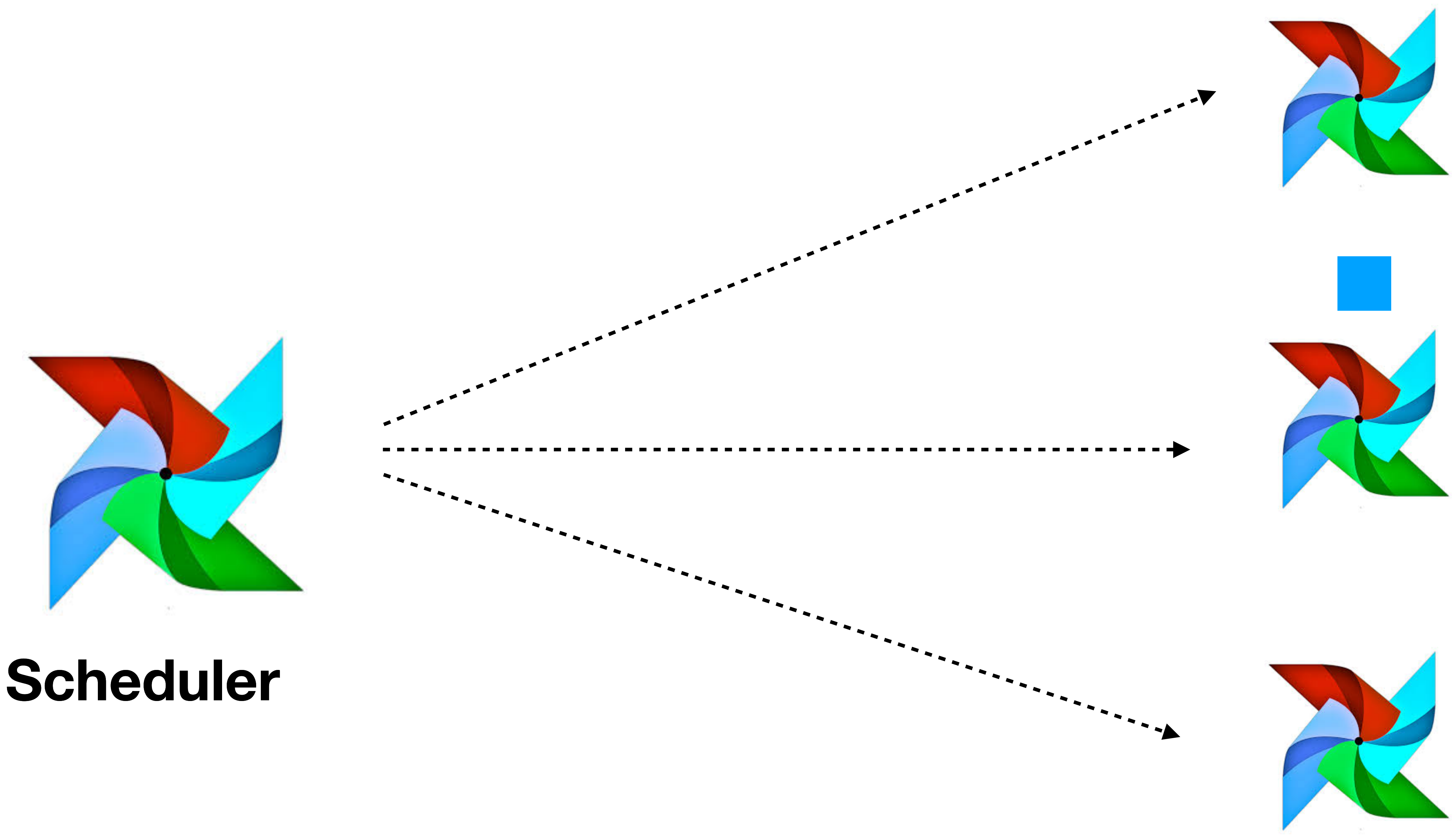
State of things: Static Allocation



State of things: Static Allocation



State of things: Static Allocation

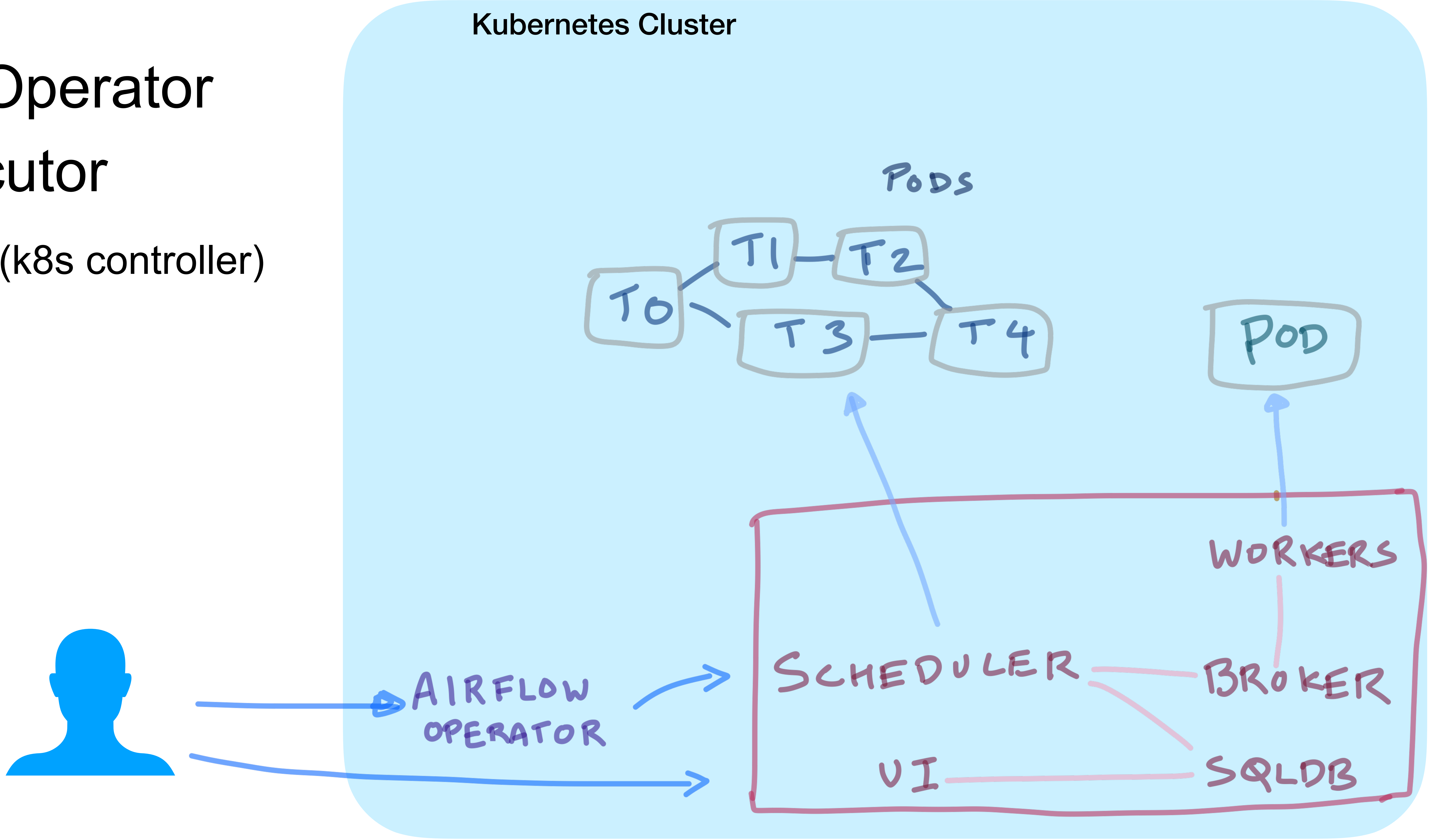


Kubernetes + Airflow

- Modernized stack using containers + k8s
- Reduced Deployment and Management complexity
- Dynamic Resource Allocation
- Automatic Fault remediation
- Improved resource utilization

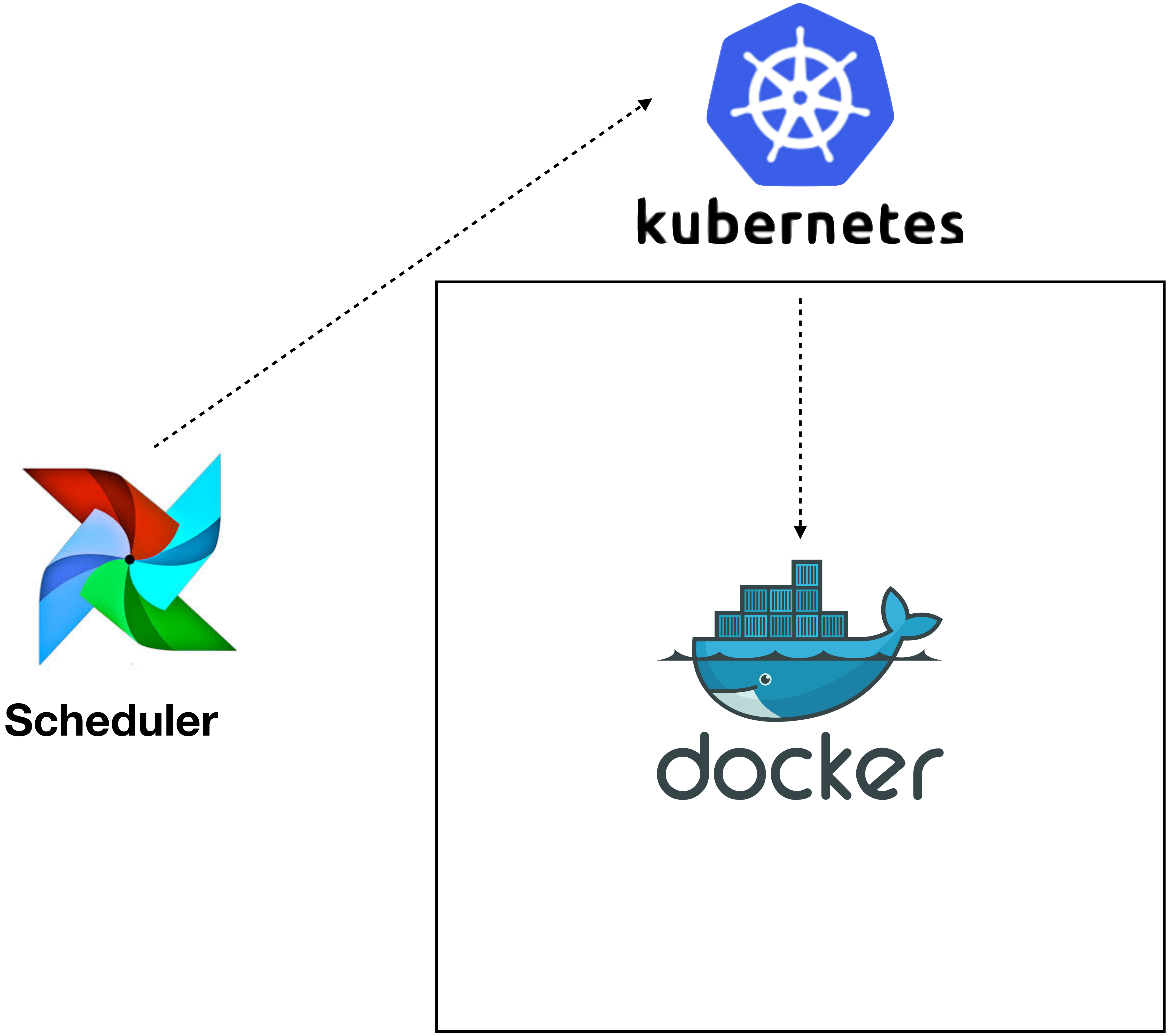
Airflow + Kubernetes

1. KubernetesPodOperator
2. KubernetesExecutor
3. AirflowOperator (k8s controller)



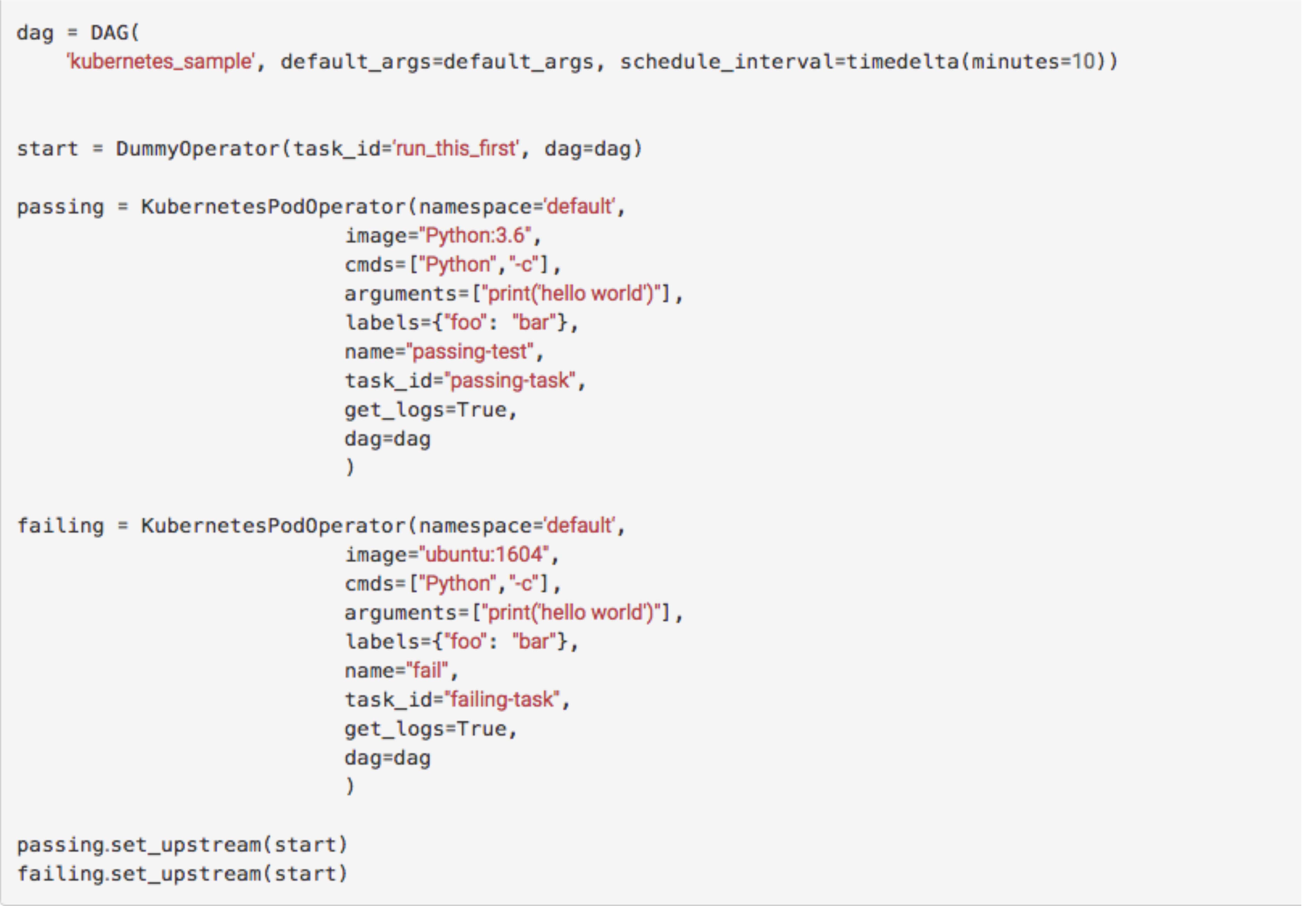
KubernetesPodOperator

- Allow users to deploy arbitrary Docker images
- Users can offload dependencies to containers
- “Lets Airflow focus on scheduling tasks”



KubernetesPodOperator

- Airflow workers are much lighter (don’t require extra libraries)
- Easy rollbacks + deployments through tags

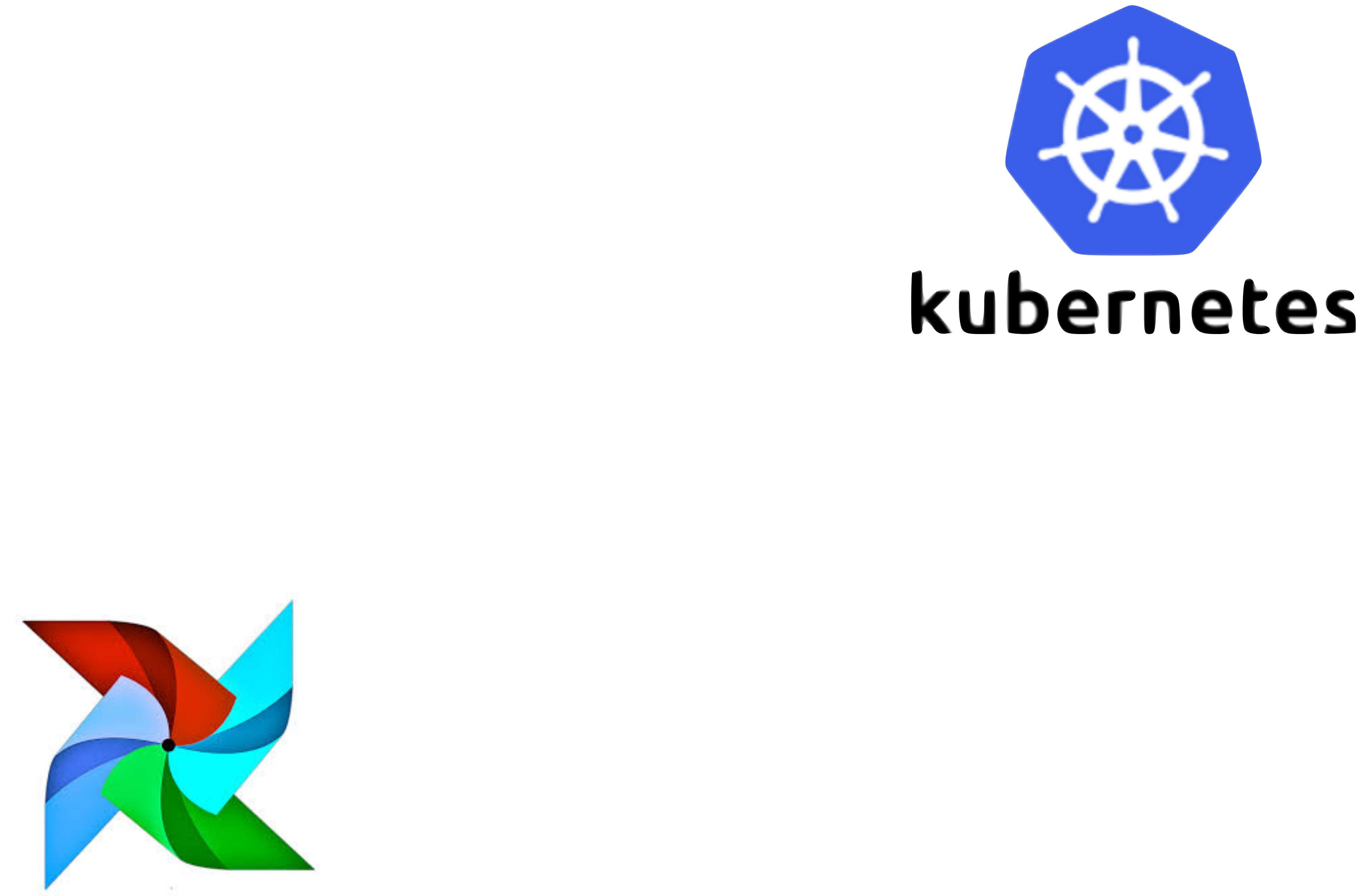


```
dag = DAG(
    'kubernetes_sample', default_args=default_args, schedule_interval=timedelta(minutes=10))


start = DummyOperator(task_id='run_this_first', dag=dag)

passing = KubernetesPodOperator(namespace='default',
                                image="Python:3.6",
                                cmds=["Python","-c"],
                                arguments=["print('hello world')"],
                                labels={"foo": "bar"},
                                name="passing-test",
                                task_id="passing-task",
                                get_logs=True,
                                dag=dag
                                )

failing = KubernetesPodOperator(namespace='default',
                                image="ubuntu:1604",
                                cmds=["Python","-c"],
                                arguments=["print('hello world')"],
                                labels={"foo": "bar"},
                                name="fail",
                                task_id="failing-task",
                                get_logs=True,
                                dag=dag
                                )

passing.set_upstream(start)
failing.set_upstream(start)
```

KubernetesExecutor

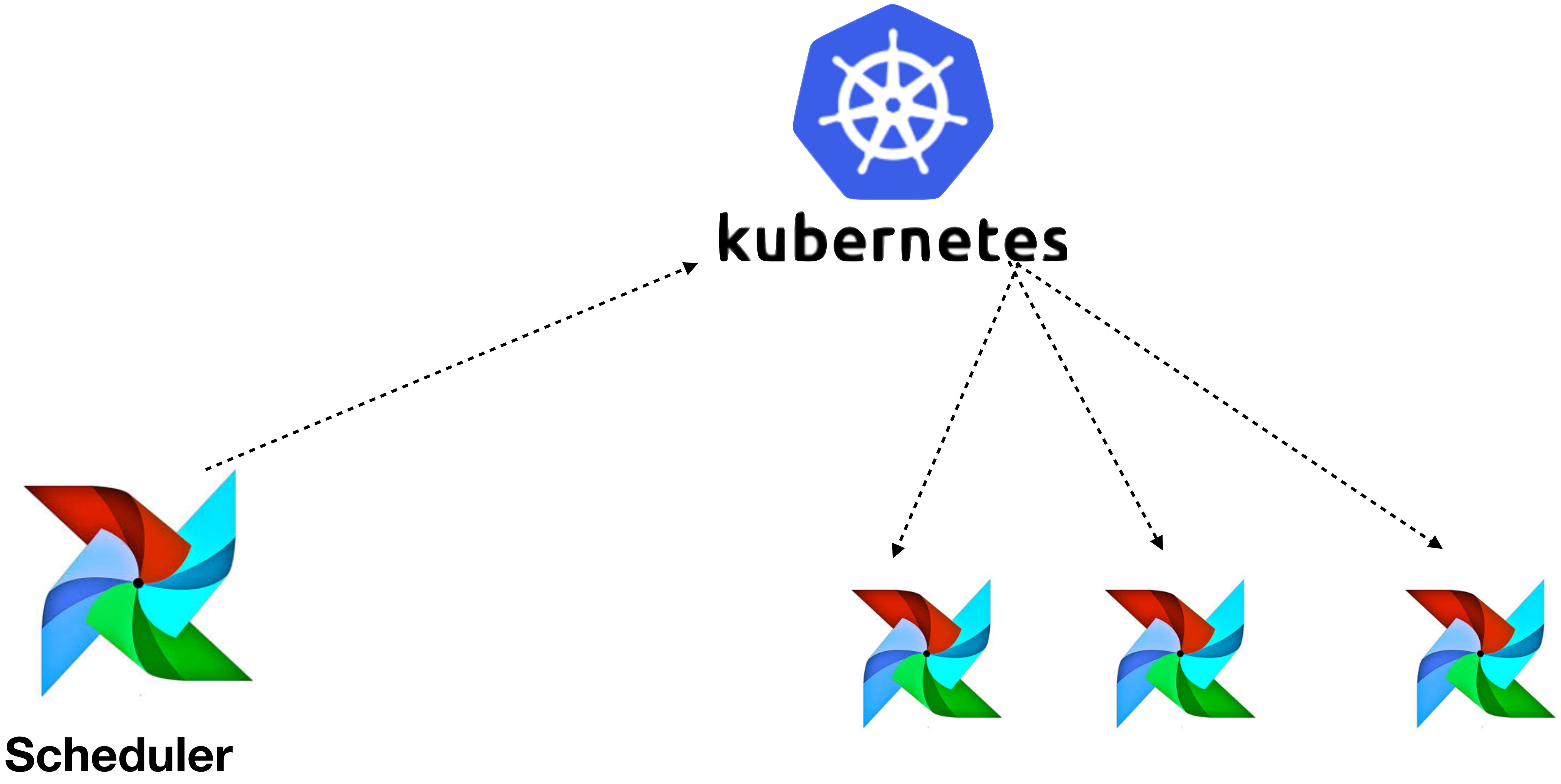
- High levels of parallelism (dynamic allocation)
- Task-level pod configuration
- Fault Tolerance

Dynamic Allocation

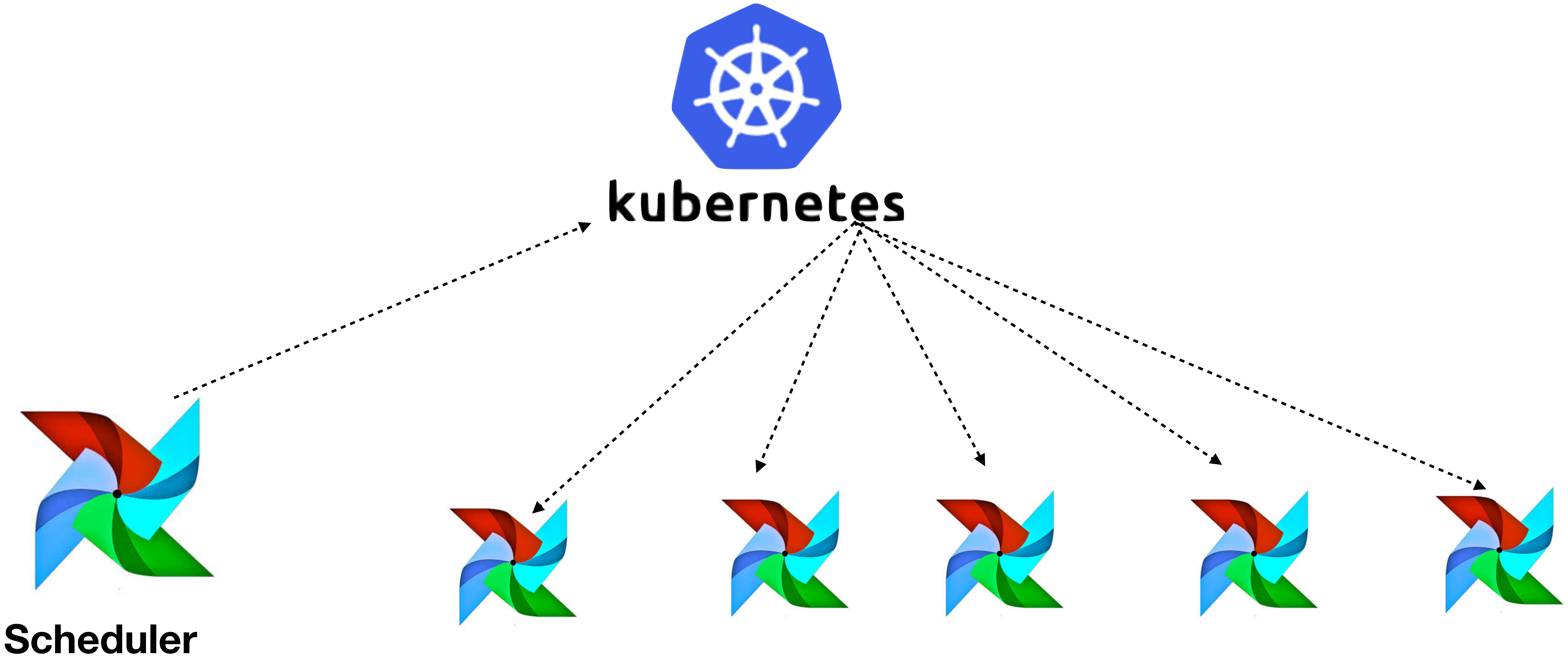


Scheduler

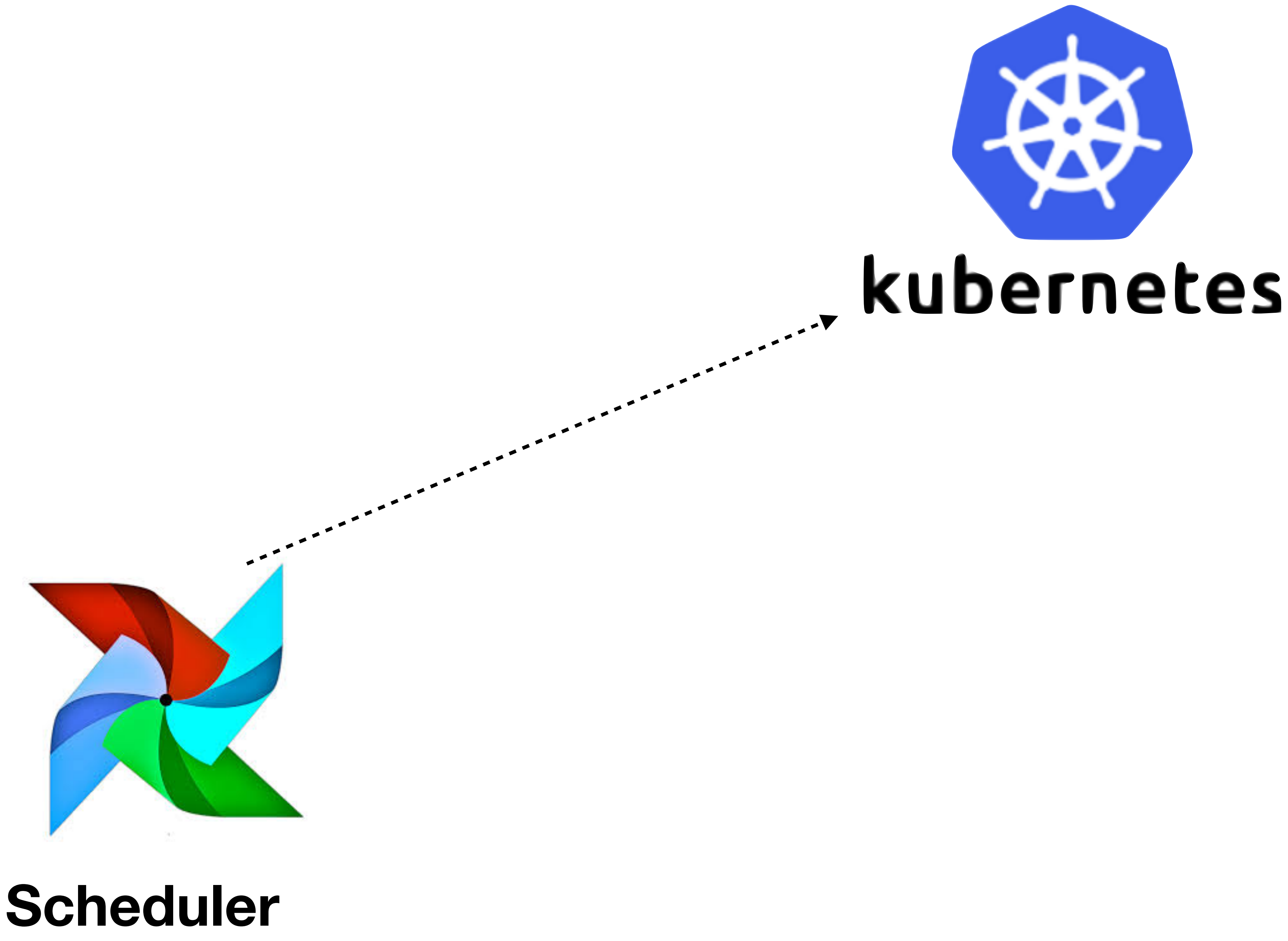
Dynamic Allocation



Dynamic Allocation



Dynamic Allocation



Scheduler

Task Level Configs

```
t = BashOperator(
  task_id = ‘account-test’,
  bash_command = ‘gcloud auth application-default login’,
  dag = dag,

  executor_config = {
  ‘request_memory’: ‘128Mi’,
  ‘limit_memory’: ‘128Mi’
  ‘image’: ‘airflow/scipy:1.1.5’
   ‘gcp-service-account’ : ‘service-account@xxx.iam.gserviceaccount.com’
  }

)
```

Task Level Configs

```
t = BashOperator(
  task_id = ‘account-test’,
  bash_command = ‘gcloud auth application-default login’,
  dag = dag,

  executor_config = {
  ‘request_memory’: ‘128Mi’,
  ‘limit_memory’: ‘128Mi’
  ‘image’: ‘airflow/scipy:1.1.5’
   ‘gcp-service-account’ : ‘service-account@xxx.iam.gserviceaccount.com’
  }

)
```

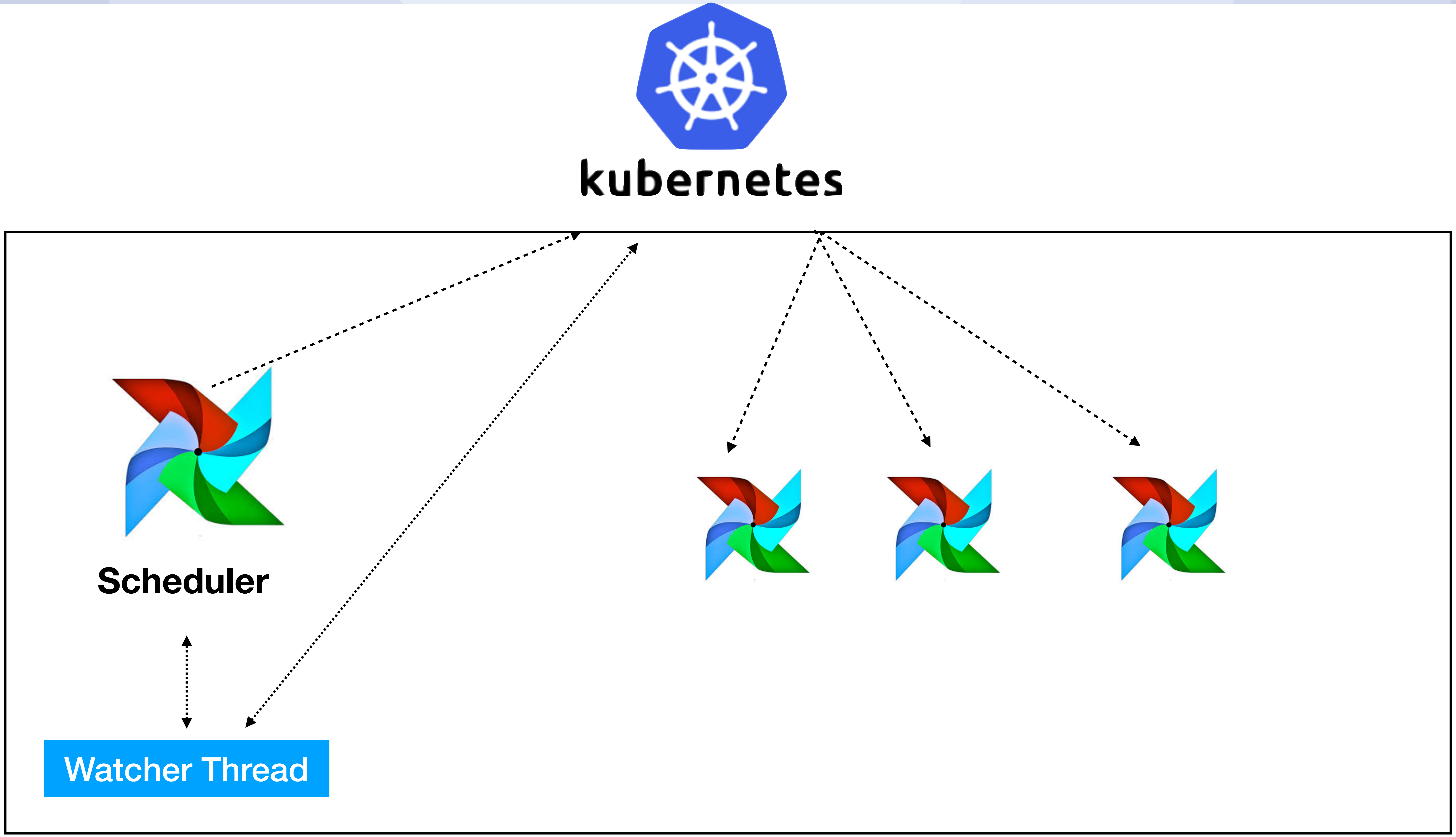
Task Level Configs

```
t = BashOperator(
  task_id = ‘account-test’,
  bash_command = ‘gcloud auth application-default login’,
  dag = dag,

  executor_config = {
  ‘request_memory’: ‘128Mi’,
  ‘limit_memory’: ‘128Mi’
  ‘image’: ‘airflow/scipy:1.1.5’
  ‘gcp-service-account’ : ‘service-account@xxx.iam.gserviceaccount.com’
  }

)
```

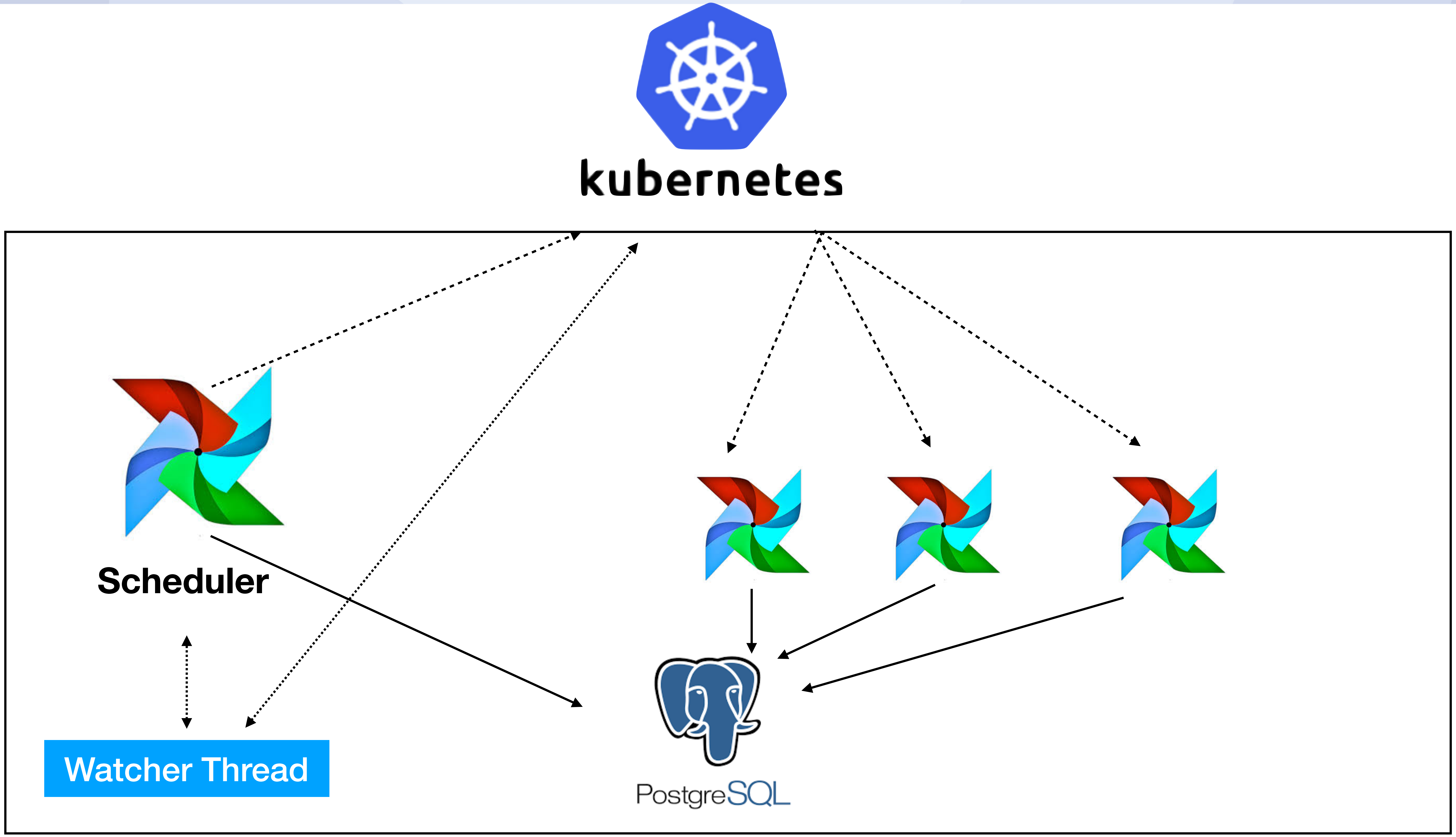
Task Level Configs

```
t = BashOperator(
  task_id = ‘account-test’,
  bash_command = ‘gcloud auth application-default login’,
  dag = dag,

  executor_config = {
  ‘request_memory’: ‘128Mi’,
  ‘limit_memory’: ‘128Mi’
  ‘image’: ‘airflow/scipy:1.1.5’
  ‘gcp-service-account’ : ‘service-account@xxx.iam.gserviceaccount.com’
  }

)
```

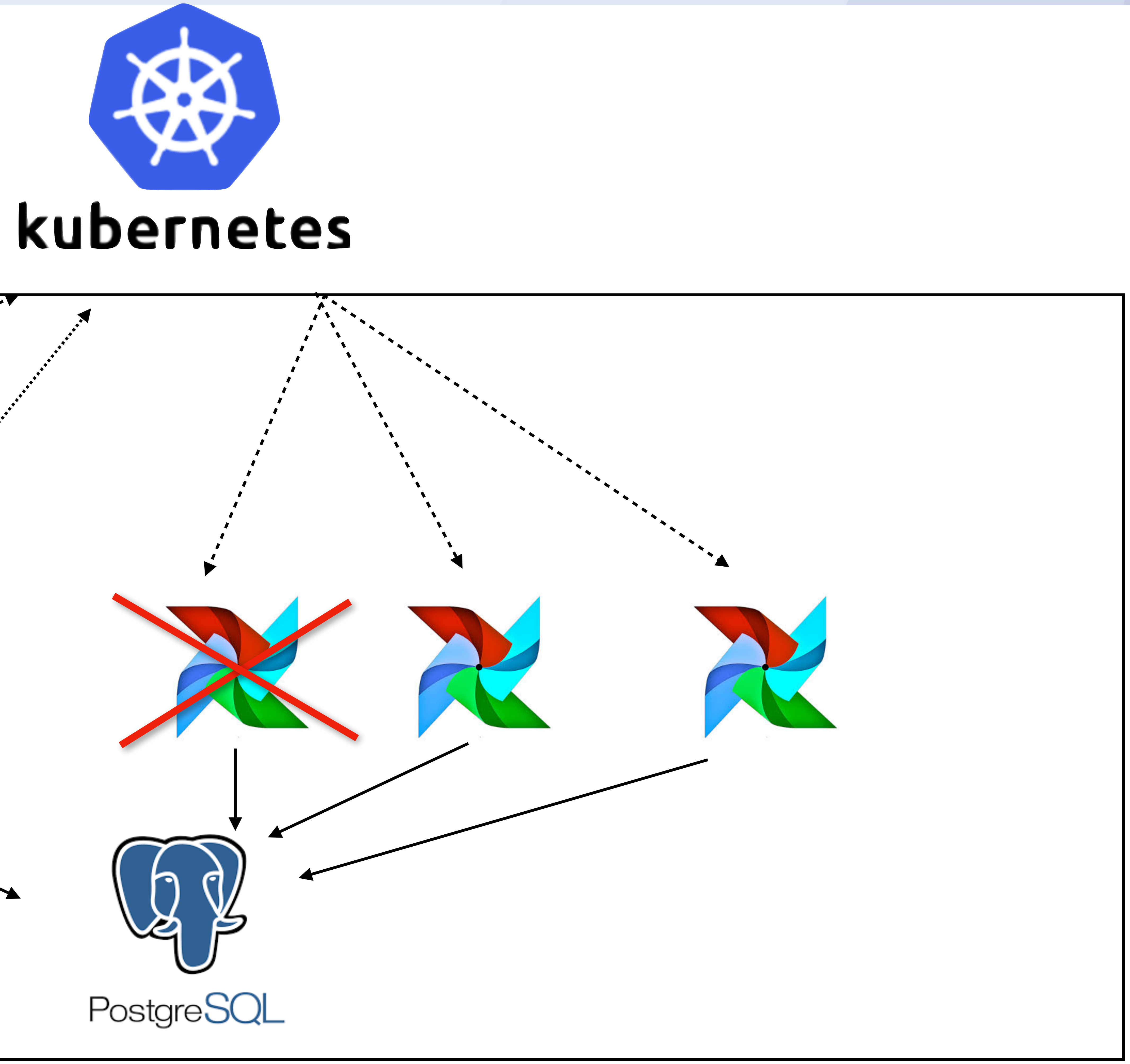
Kubernetes Executor



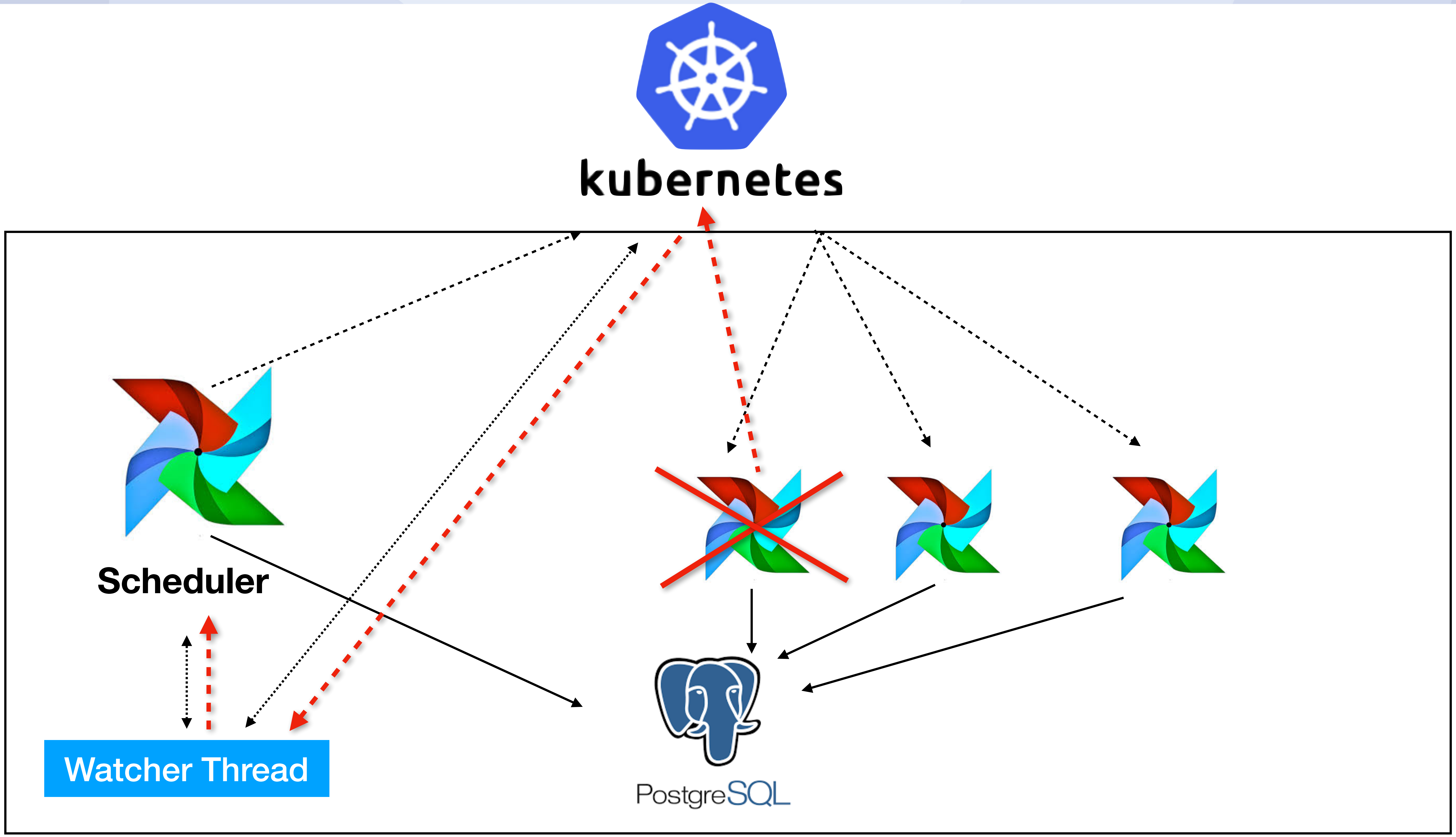
Kubernetes Executor



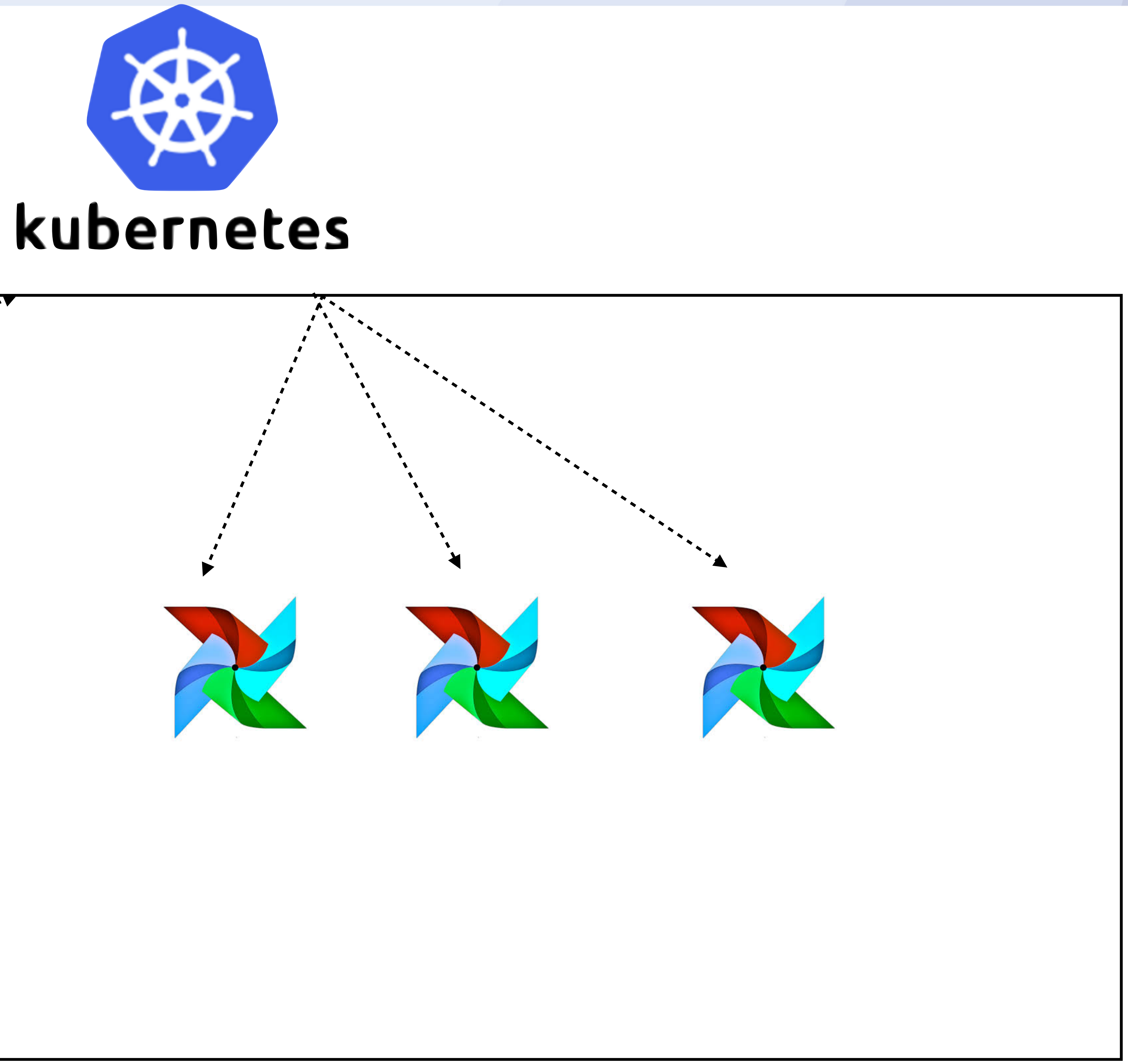
Kubernetes Executor



Kubernetes Executor



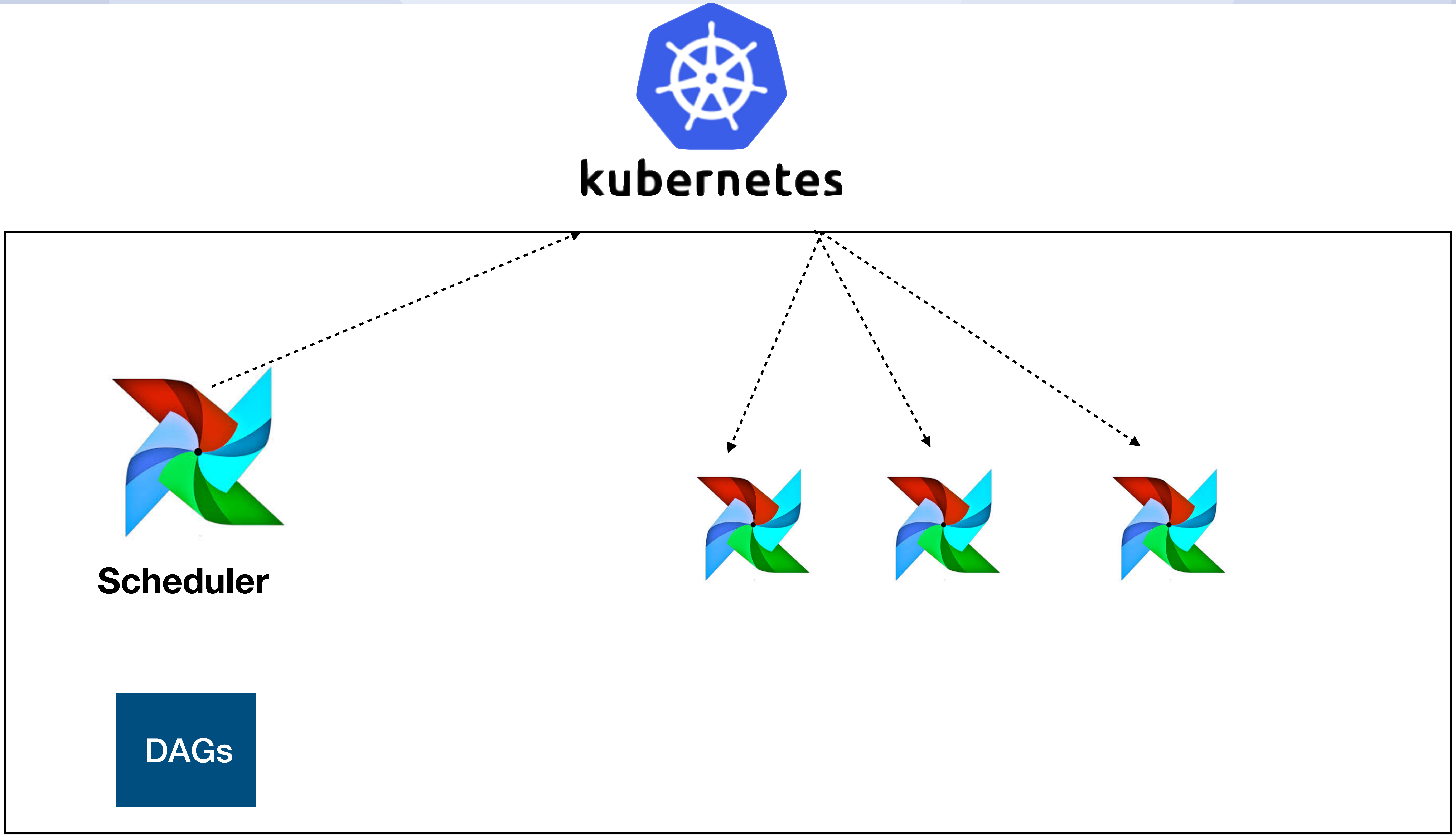
Fault Tolerance



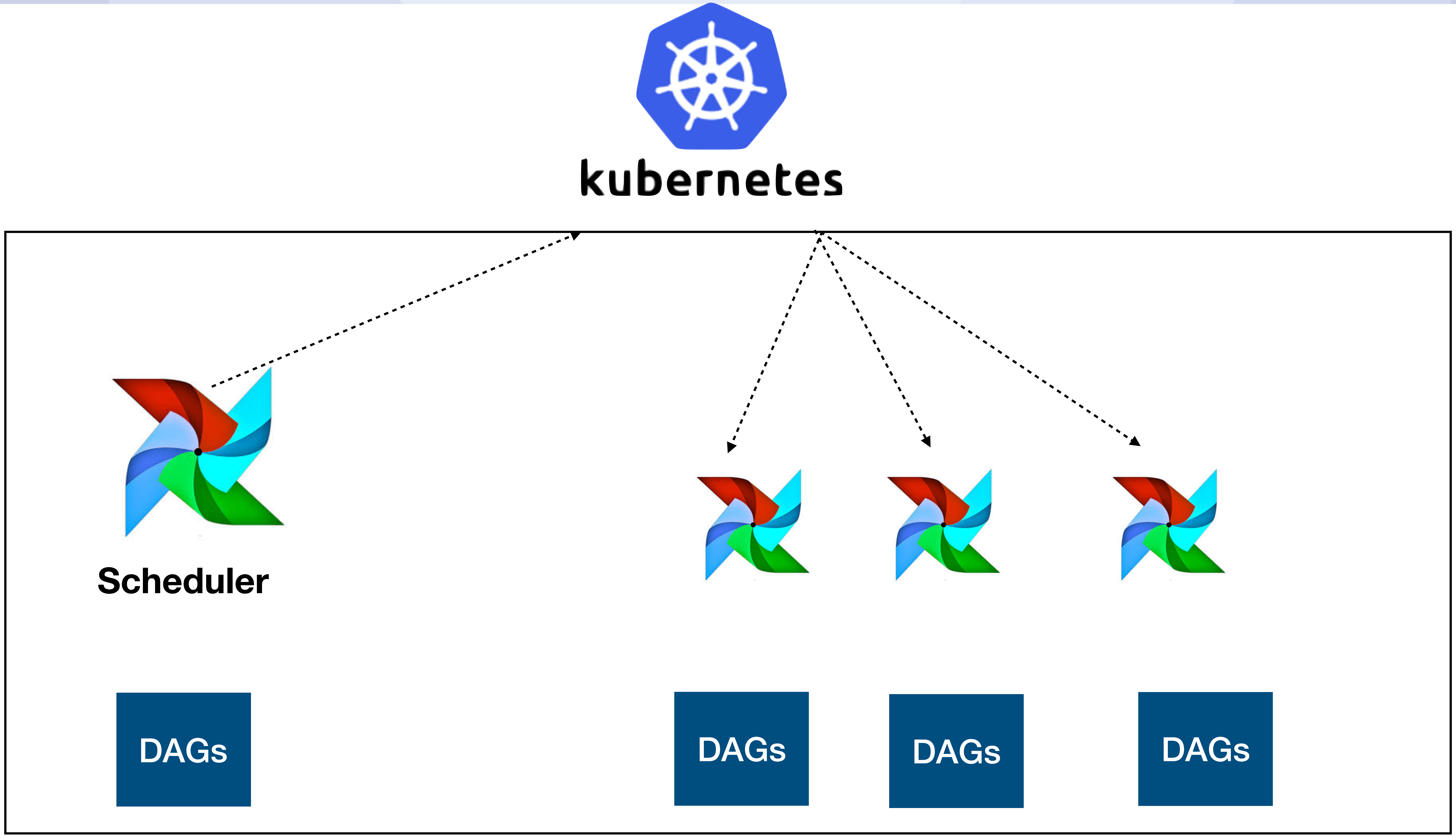
Fault Tolerance

- Uses “resourceVersion” to re-create state
- Maintain a resourceVersion in SQL table for state recovery

DAG Propagation



DAG Propagation



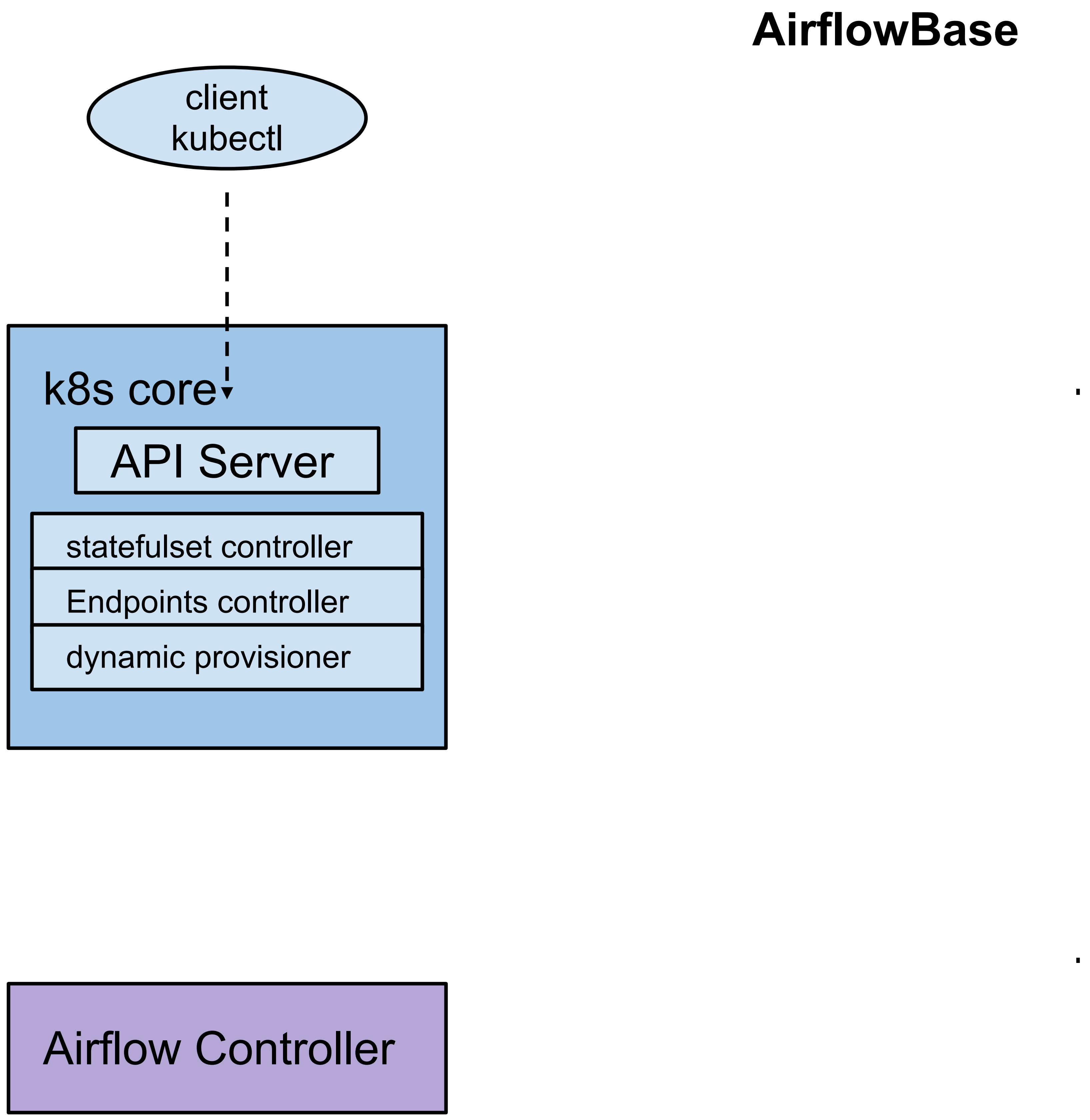
DAG Injection

- Three modes: Git-init mode, persistent volume mode, and “pre-baked” mode (1.10.2)
- Git-init mode + pre-baked is recommended for development and small instances of Airflow, because it does not involve any distributed file systems
- Persistent volume mode recommended for large DAG folders

AirflowOperator (k8s controller)

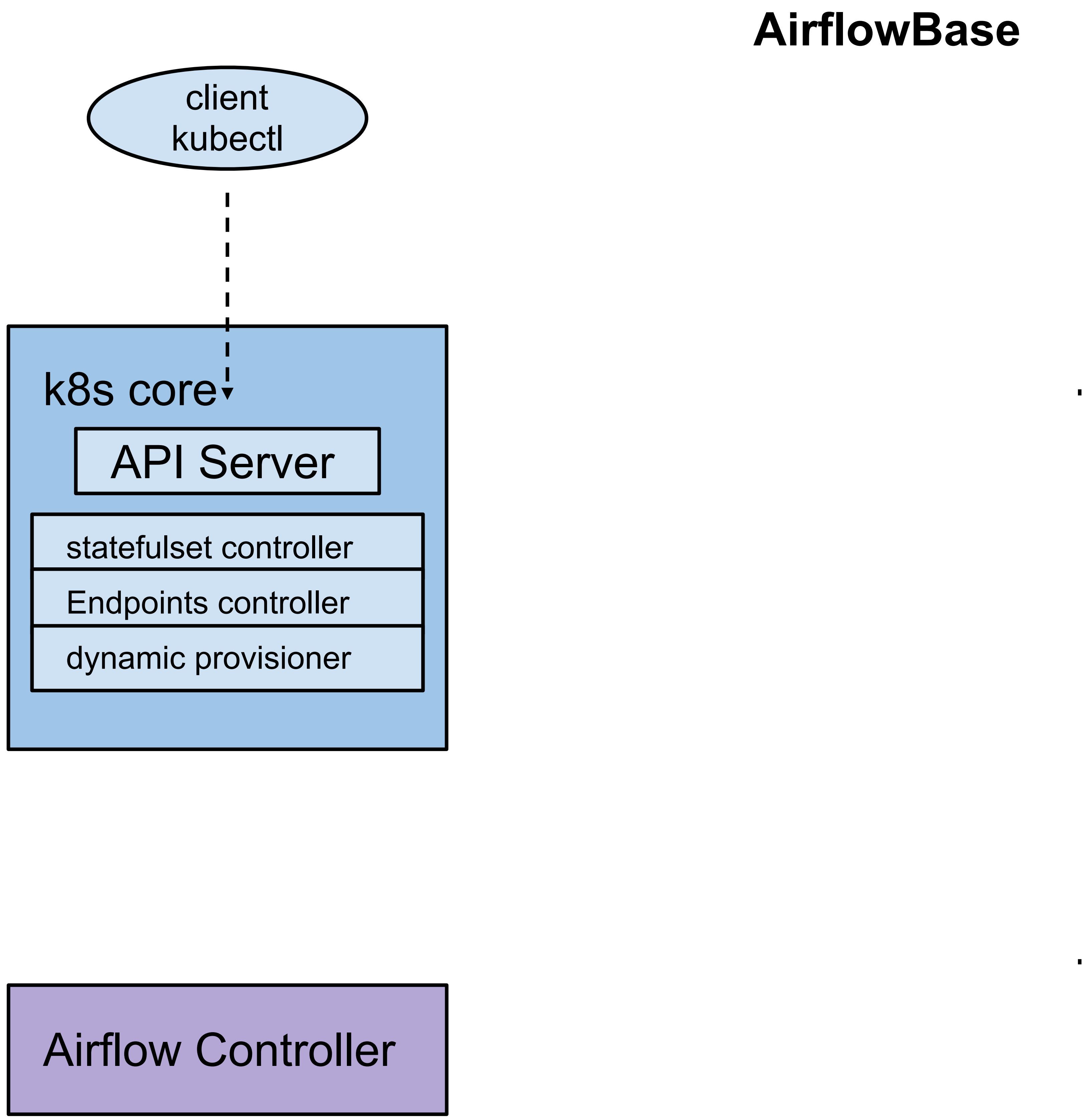
- Simplifies Airflow deployment and management
- Is a Custom Kubernetes controller
- Using CRDs, user creates declarative specs describing his intent
    - AirflowBase
    - AirflowCluster

AirflowBase CRD



- AirflowBase CRD
  - MySQL/Postgres/SQLProxy
  - NFS
- Used by multiple Airflow Clusters

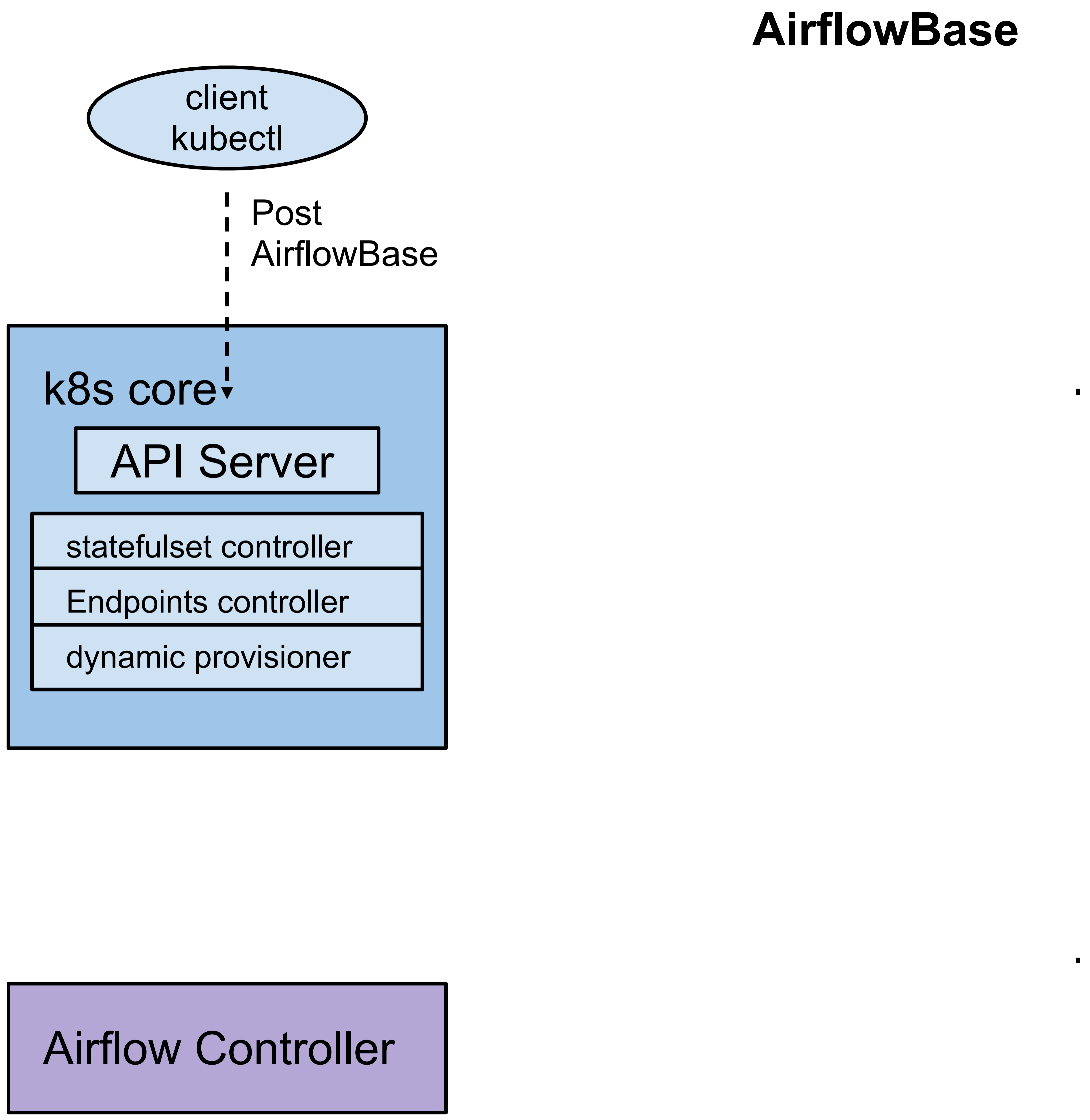
AirflowBase CRD



- AirflowBase CRD
  - MySQL/Postgres/SQLProxy
  - NFS
- Used by multiple Airflow Clusters

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowBase
metadata:
  name: mc-base
spec:
  mysql:
    operator: False
  storage:
    version: ""
```

AirflowBase CRD

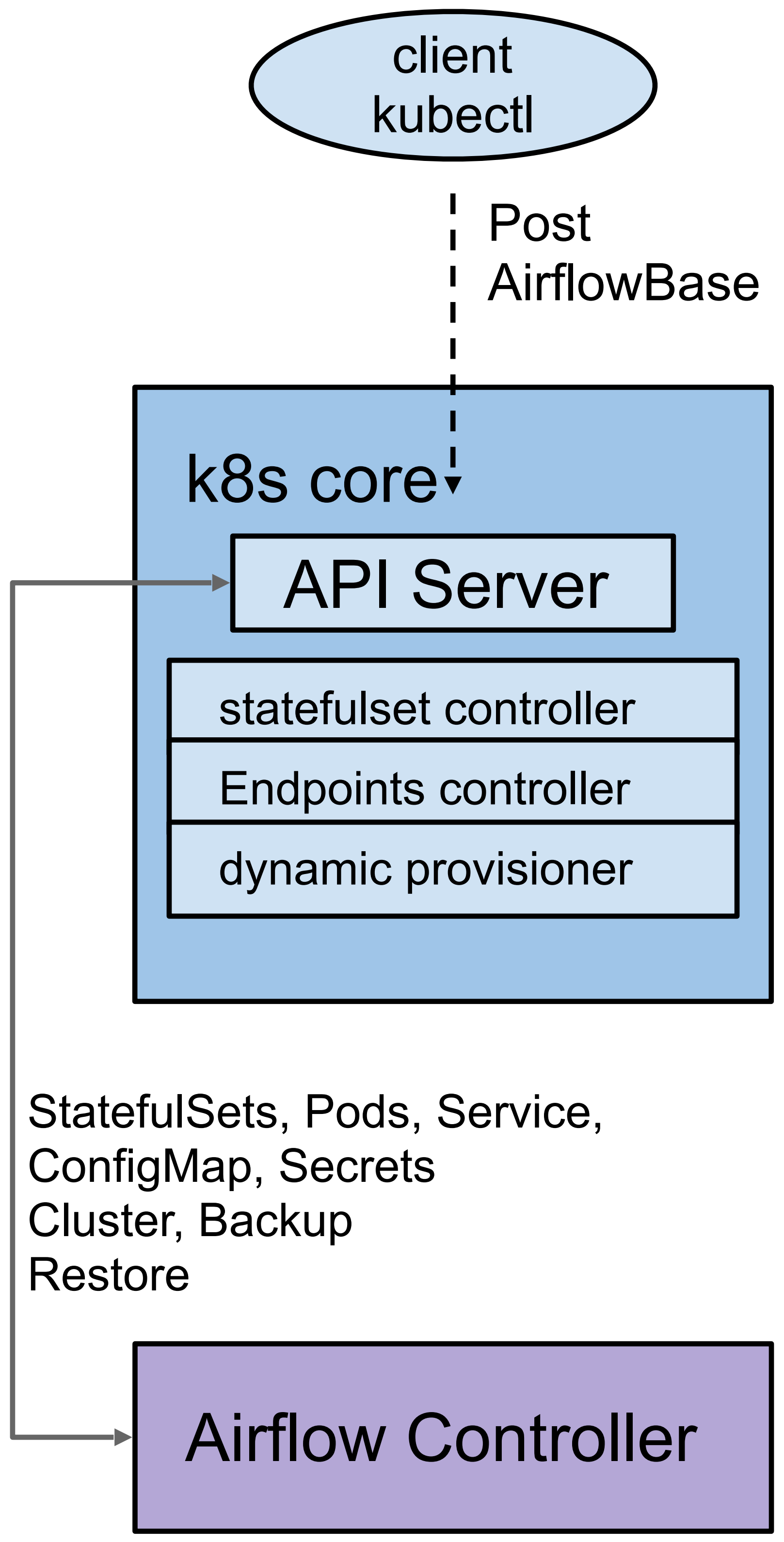


- AirflowBase CRD
  - MySQL/Postgres/SQLProxy
  - NFS
- Used by multiple Airflow Clusters

```yaml
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowBase
metadata:
  name: mc-base
spec:
  mysql:
    operator: False
  storage:
    version: ""
```

AirflowBase CRD

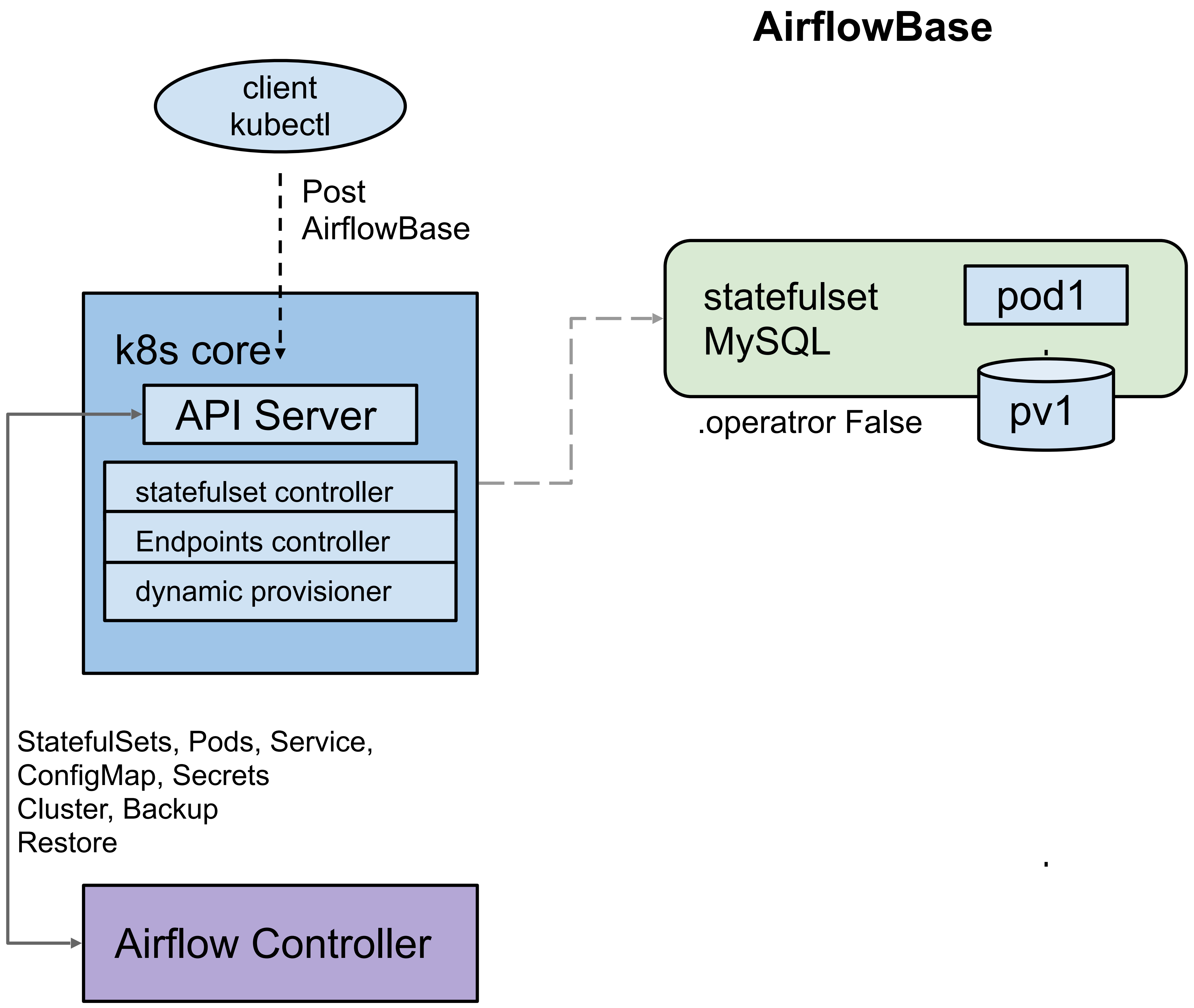
AirflowBase



- AirflowBase CRD
  - MySQL/Postgres/SQLProxy
  - NFS
- Used by multiple Airflow Clusters

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowBase
metadata:
  name: mc-base
spec:
  mysql:
    operator: False
  storage:
    version: ""
```

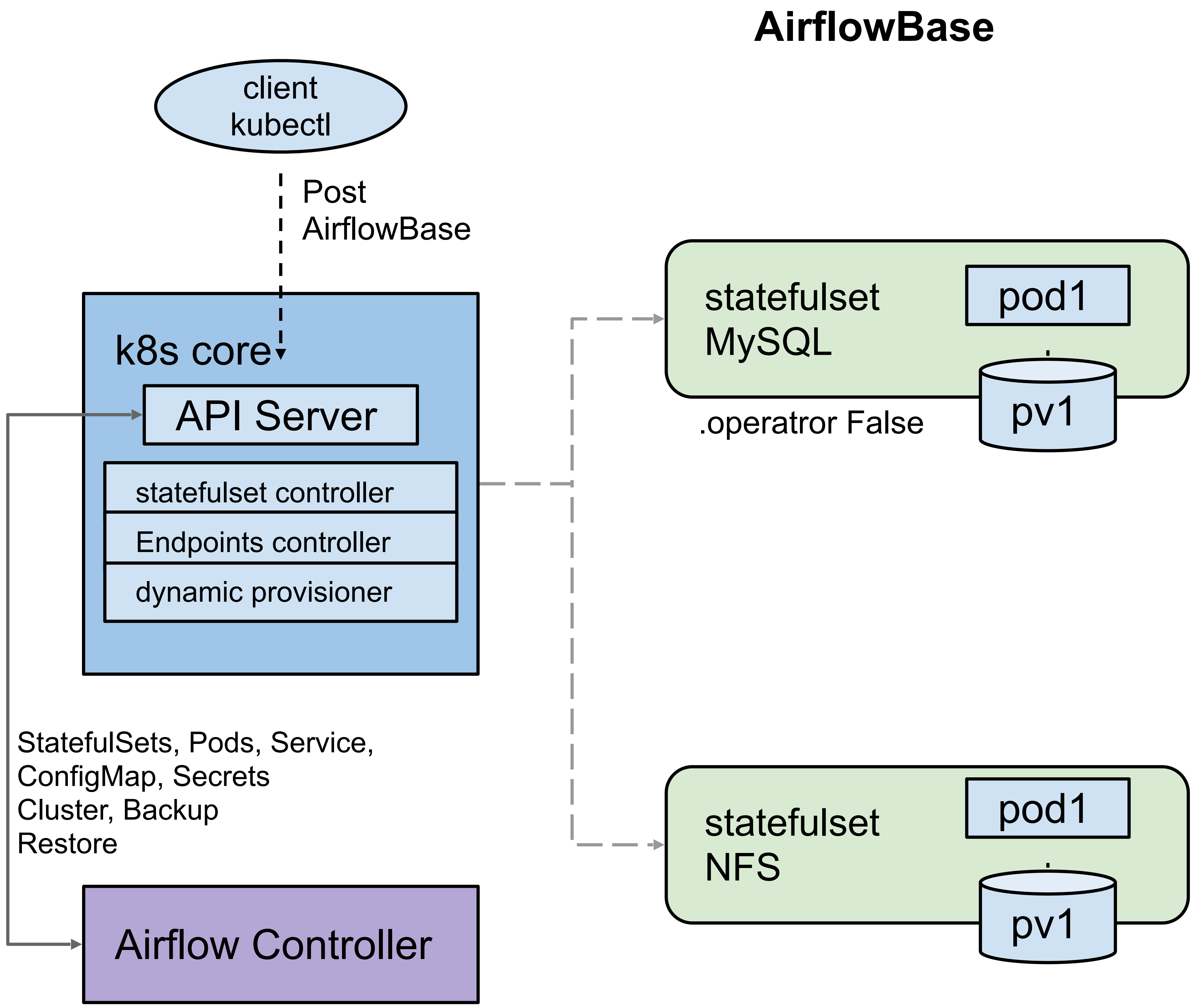
AirflowBase CRD



- AirflowBase CRD
  - MySQL/Postgres/SQLProxy
  - NFS
- Used by multiple Airflow Clusters

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowBase
metadata:
  name: mc-base
spec:
  mysql:
    operator: False
  storage:
    version: ""
```

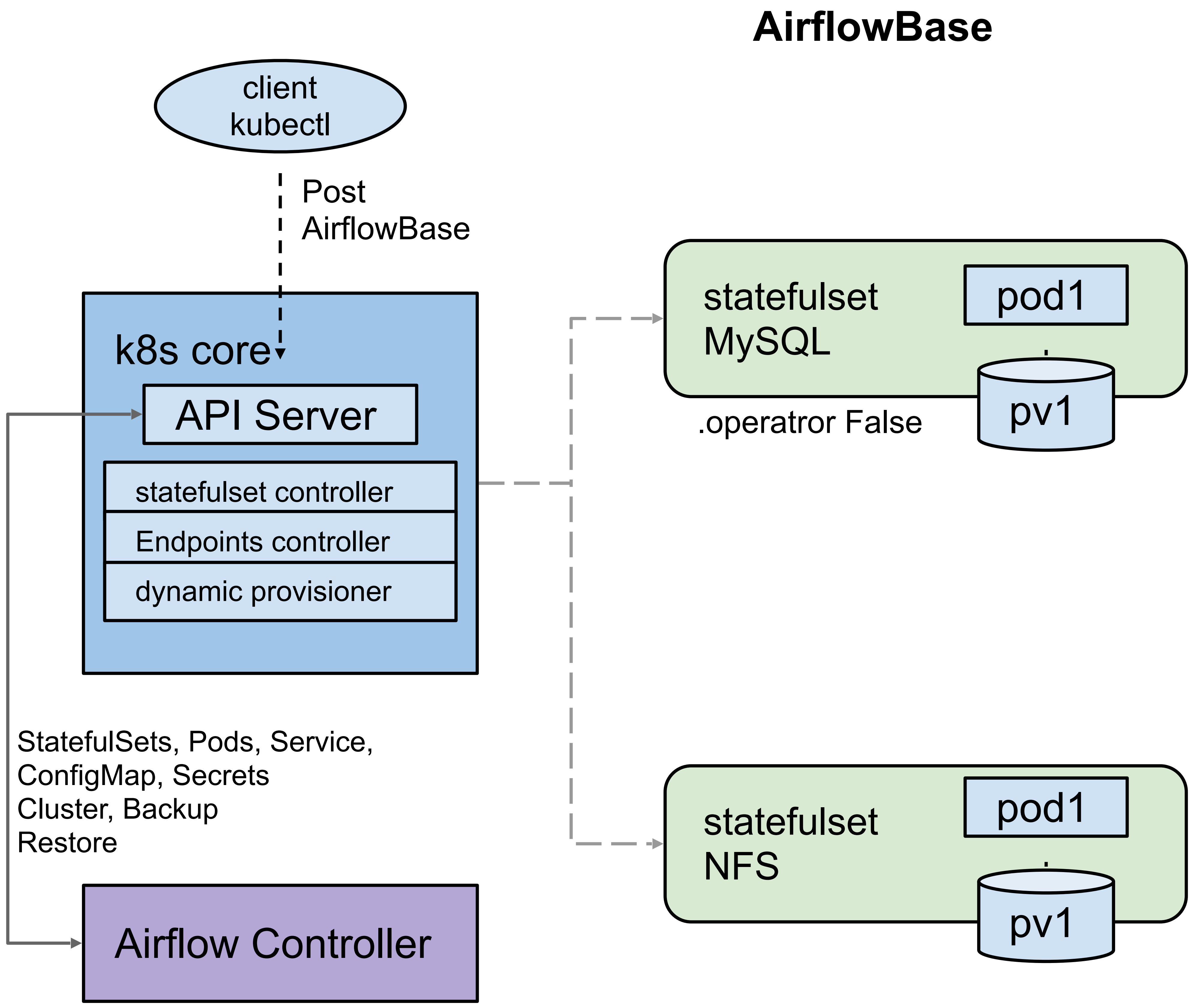
AirflowBase CRD



- AirflowBase CRD
  - MySQL/Postgres/SQLProxy
  - NFS
- Used by multiple Airflow Clusters

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowBase
metadata:
  name: mc-base
spec:
  mysql:
    operator: False
  storage:
    version: ""
```

AirflowBase CRD



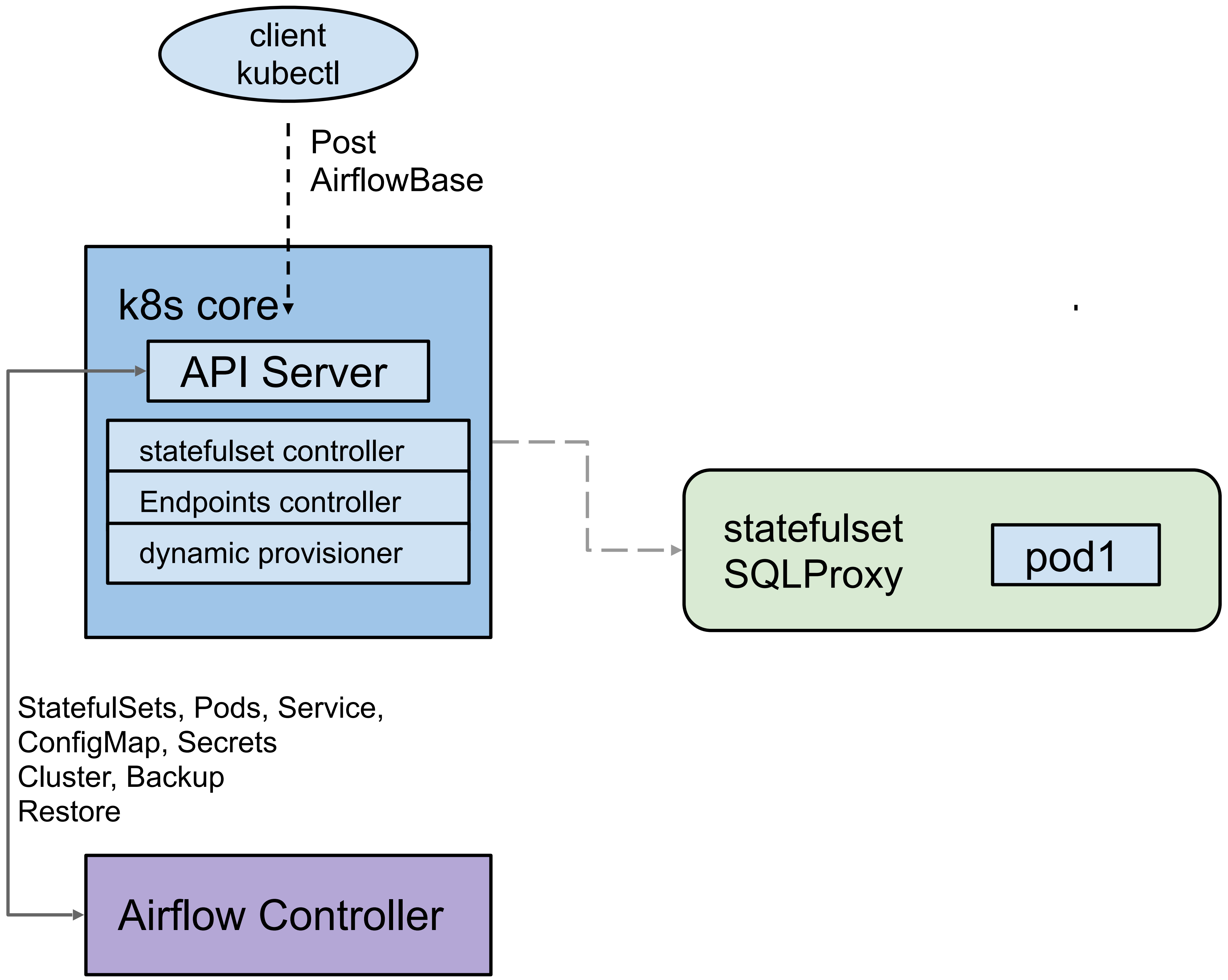
- AirflowBase CRD
  - MySQL/Postgres/SQLProxy
  - NFS
- Used by multiple Airflow Clusters

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowBase
metadata:
  name: mc-base
spec:
  mysql:
    operator: False
  storage:
    version: ""
```

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowBase
metadata:
  name: ck-base
spec:
  sqlproxy:
          project: someproject
          region: us-central1
          instance: testsql-cluster
  storage:
    version: ""
```

AirflowBase CRD

AirflowBase



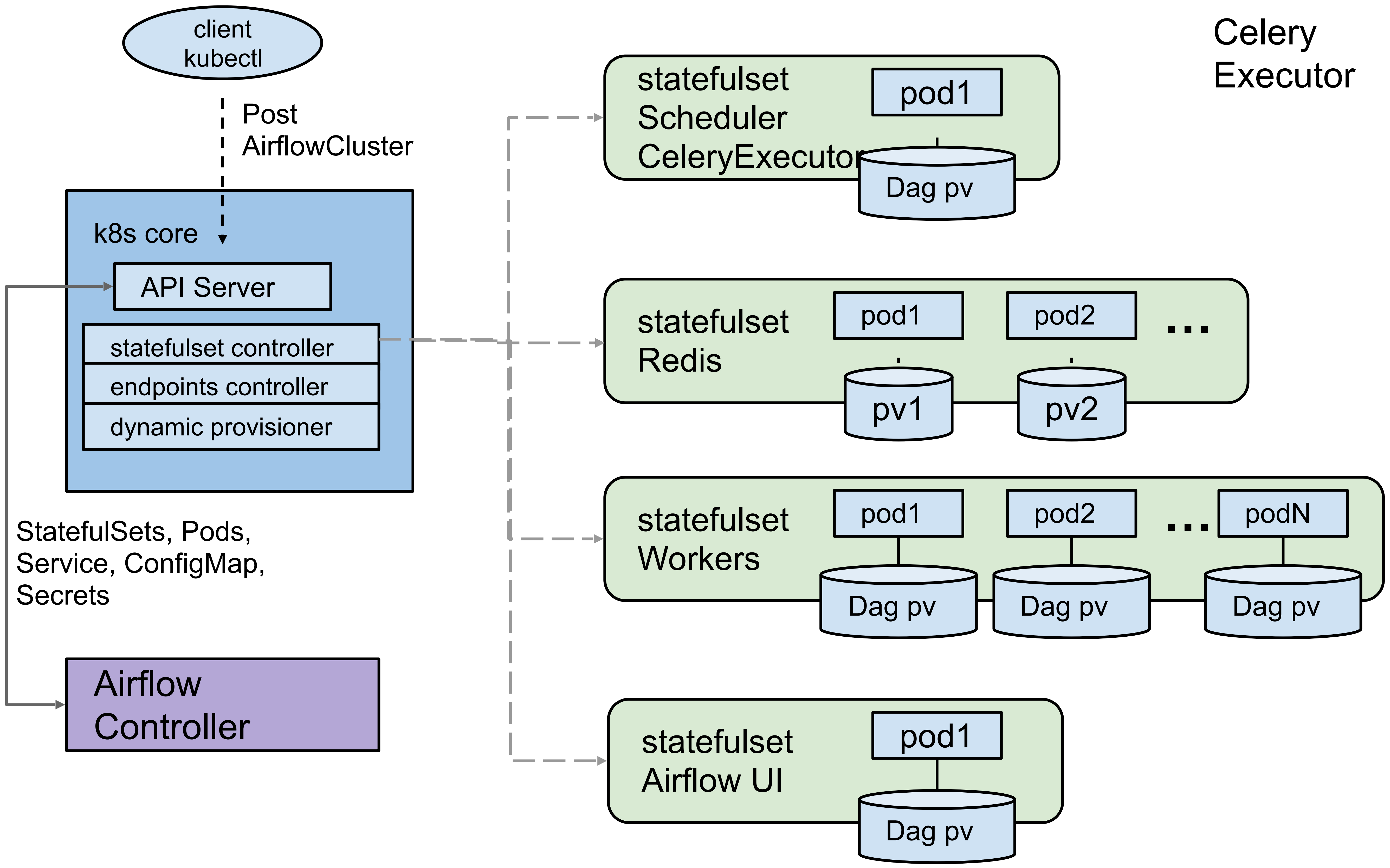
- AirflowBase CRD
  - MySQL/Postgres/SQLProxy
  - NFS
- Used by multiple Airflow Clusters

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowBase
metadata:
  name: mc-base
spec:
  mysql:
    operator: False
  storage:
    version: ""
```

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowBase
metadata:
  name: ck-base
spec:
  sqlproxy:
          project: someproject
          region: us-central1
          instance: testsql-cluster
  storage:
    version: ""
```

AirflowCluster CRD

AirflowCluster

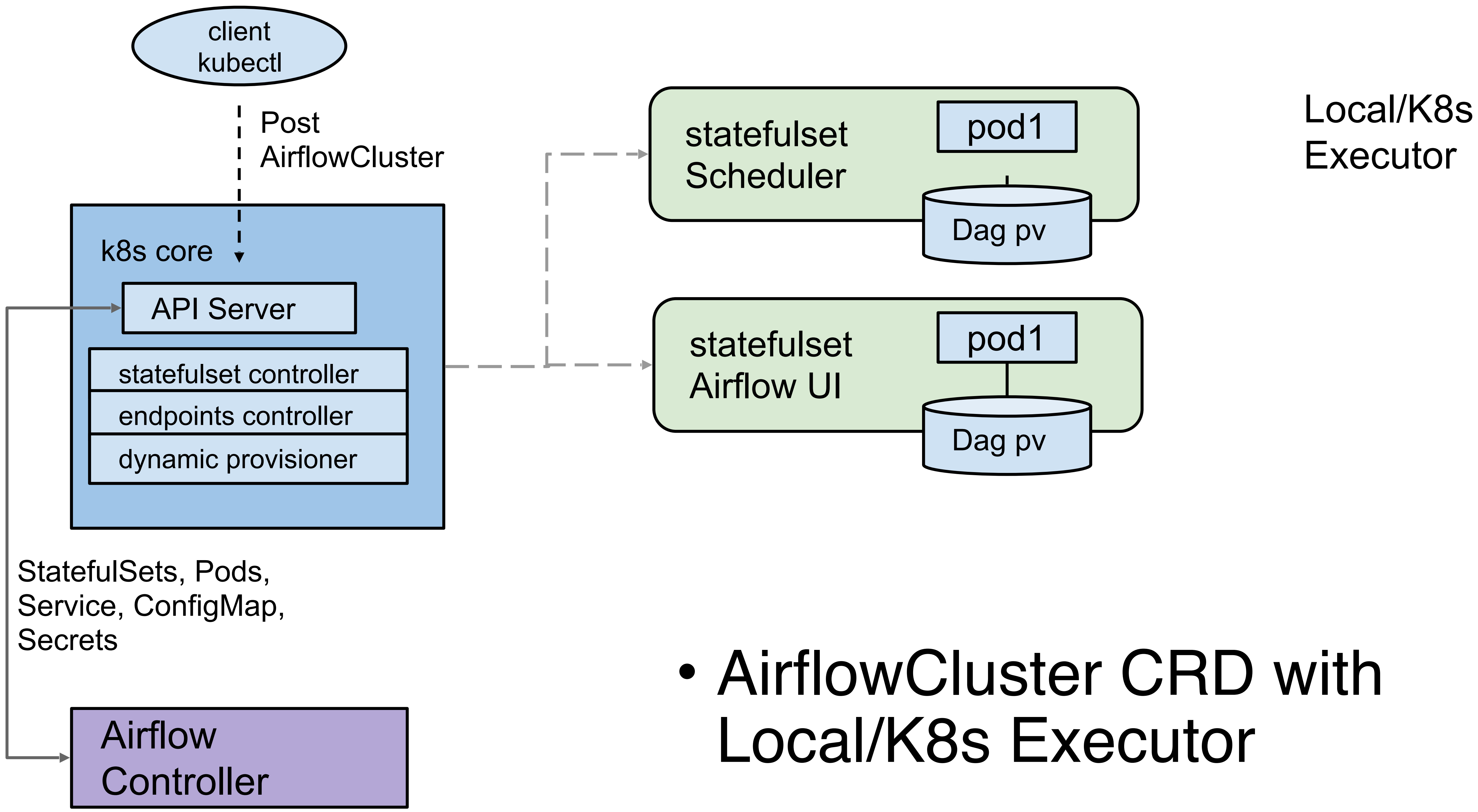


- Celery Executor
  - Redis
  - Airflow UI
  - Airflow Scheduler
  - Airflow Workers
- Each cluster gets its own unique SQL connection string (user:password/dB).

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowCluster
metadata:
  name: mc-cluster
spec:
  executor: Celery
  config:
    airflow:
      AIRFLOW_SOME_CONFIG: SomeValue
  redis:
    operator: False
  scheduler:
    version: "1.10.1"
  ui:
    replicas: 1
    version: "1.10.1"
  worker:
    replicas: 2
    version: "1.10.1"
  flower:
    replicas: 1
    version: "1.10.1"
  dags:
    subdir: "airflow/example_dags/"
    git:
      repo: "https://github.com/apache/incubator
airflow/"
      once: true
  airflowbase:
    name: mc-base
```

AirflowCluster CRD

AirflowCluster



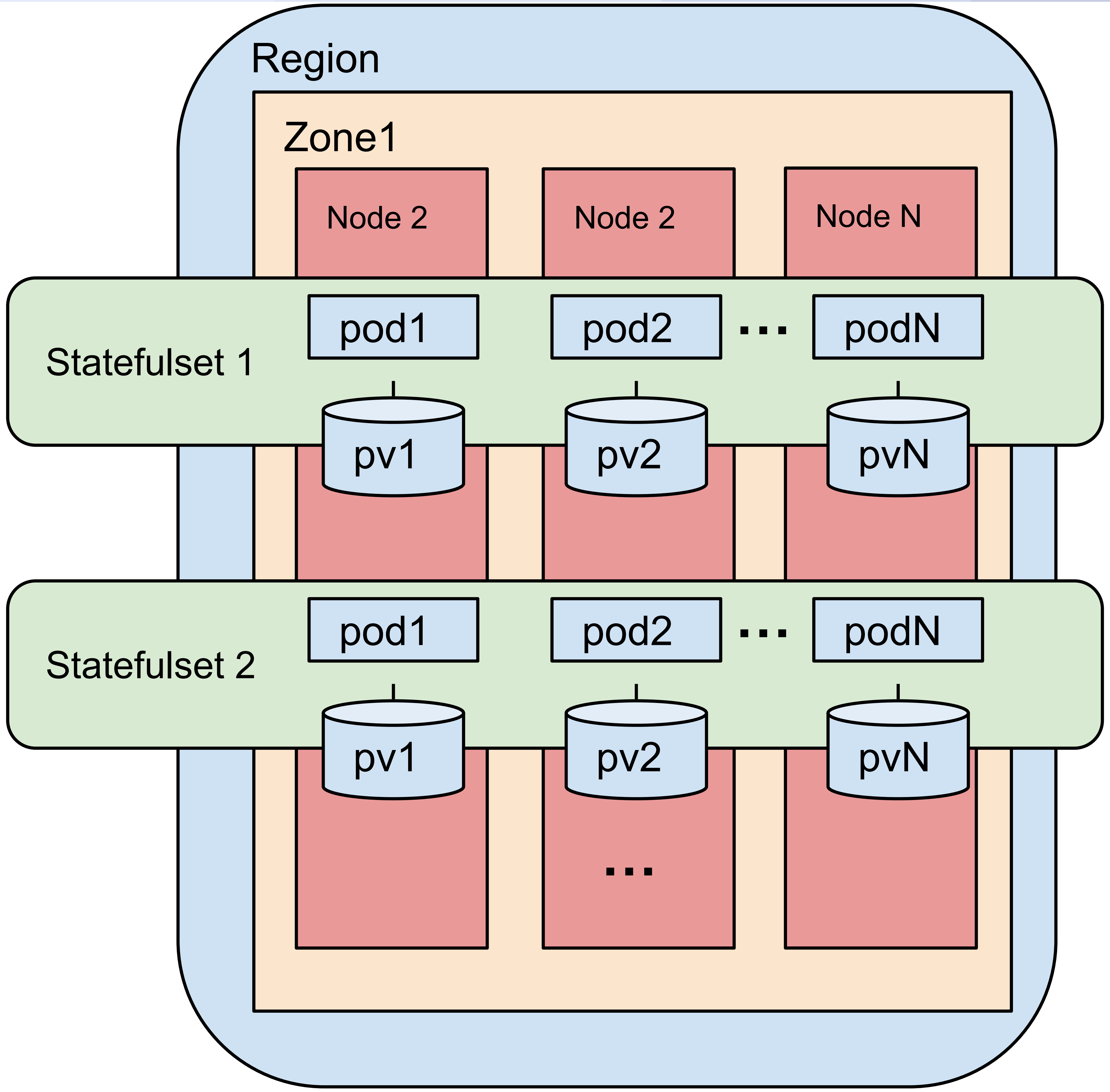
- AirflowCluster CRD with Local/K8s Executor

- Airflow UI
- Airflow Scheduler

```
apiVersion: airflow.k8s.io/v1alpha1
kind: AirflowCluster
metadata:
  name: mk-cluster
spec:
  executor: Kubernetes
  ui:
    replicas: 1
    version: "1.10.1"
  scheduler:
    version: "1.10.1"
  worker:
    version: "1.10.1"
  dags:
    subdir: "airflow/example_dags/"
    git:
      repo: "https://github.com/apache/
incubator-airflow/"
      once: true
      branch: master
  airflowbase:
    name: mc-base
```

AirflowCluster CRD

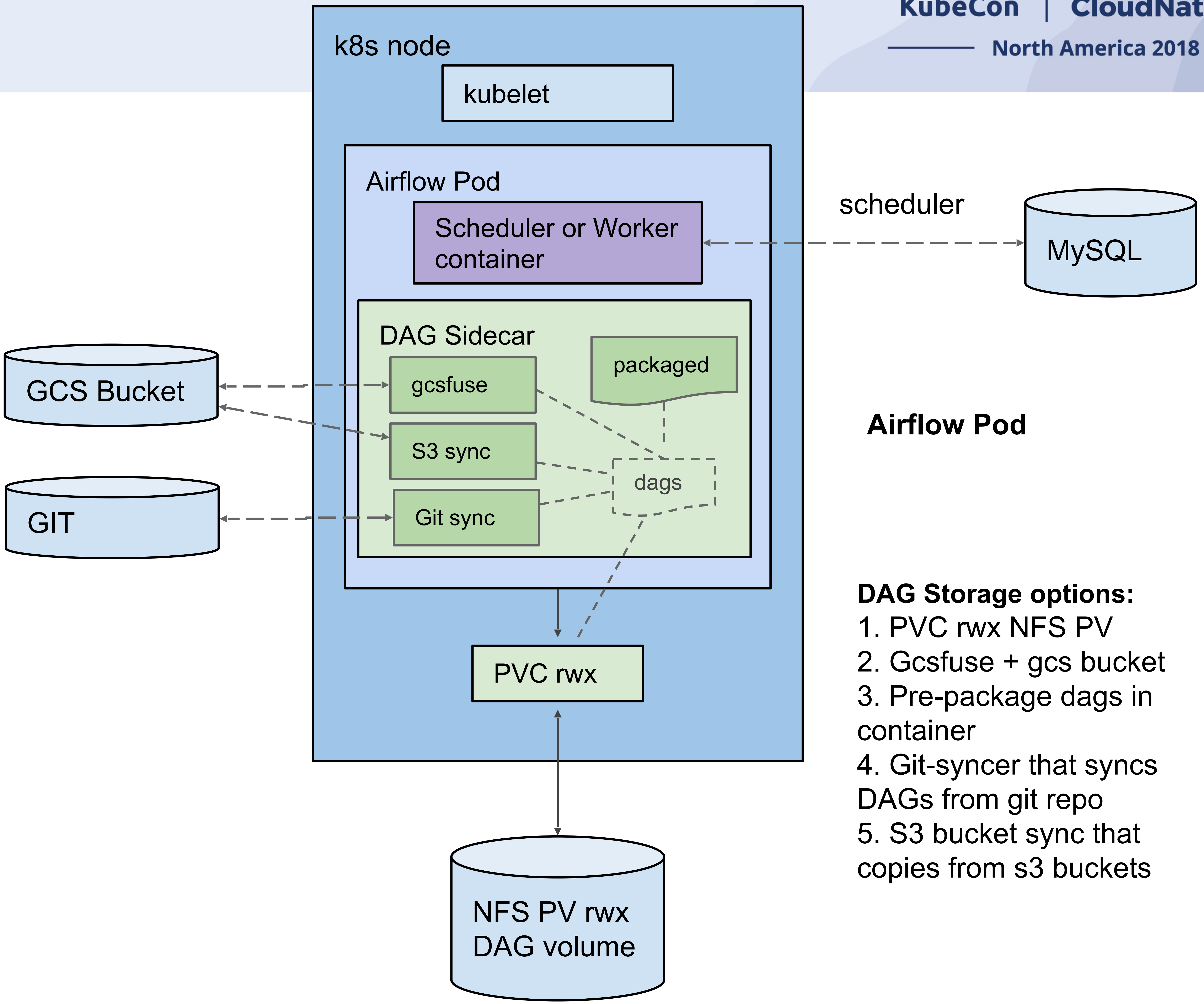
- Pod affinity rules cluster.Spec.Affinity.*.topology can be set to “kubernetes.io/hostname” to spread Pods across Nodes within a Zone.
- Limit the impact of node failures within a zone



Pods spread across Nodes in Zone

AirflowOperator

- Multiple DAG sources are supported via a DAG Sidecar
- Custom Airflow Pod images are supported



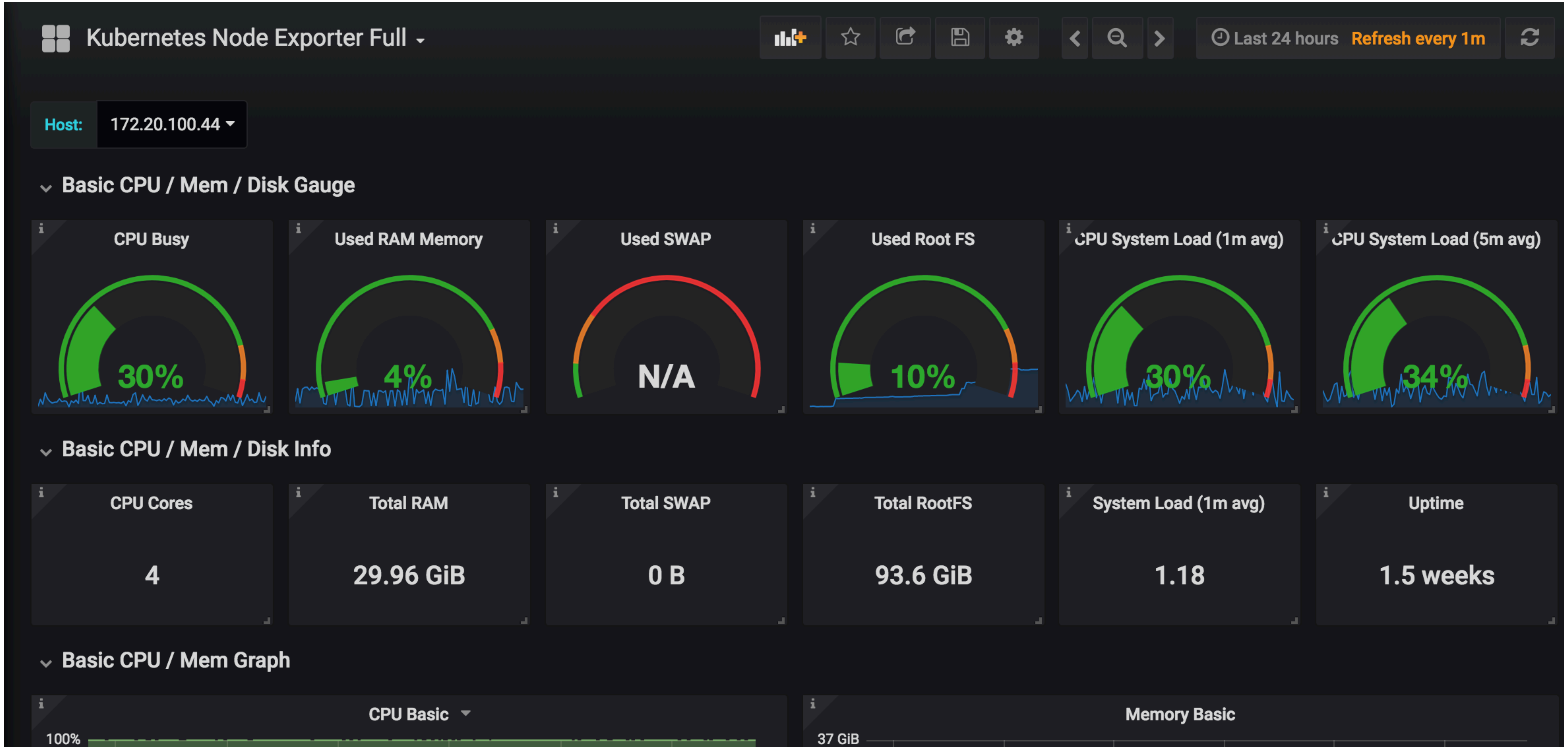
Airflow Pod

DAG Storage options:
1. PVC rwx NFS PV
2. Gcsfuse + gcs bucket
3. Pre-package dags in container
4. Git-syncer that syncs DAGs from git repo
5. S3 bucket sync that copies from s3 buckets

Monitoring

- Can use existing Kubernetes infrastructure
- Only needed to think about Airflow, not machines

Prometheus



Kubernetes Node Exporter Full

Host: 172.20.100.44

Basic CPU / Mem / Disk Gauge

CPU Busy	Used RAM Memory	Used SWAP	Used Root FS	CPU System Load (1m avg)	CPU System Load (5m avg)
30%	4%	N/A	10%	30%	34%

Basic CPU / Mem / Disk Info

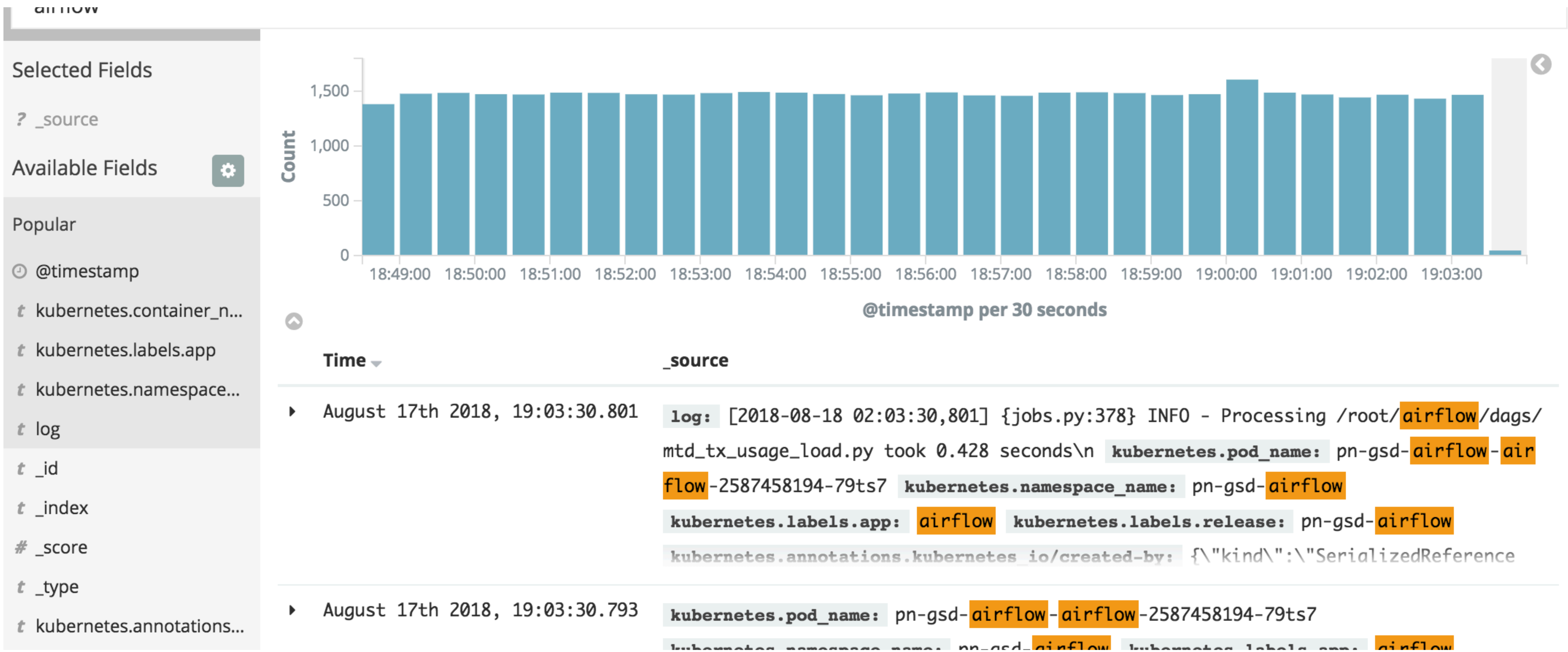
CPU Cores	Total RAM	Total SWAP	Total RootFS	System Load (1m avg)	Uptime
4	29.96 GiB	0 B	93.6 GiB	1.18	1.5 weeks

Basic CPU / Mem Graph

CPU Basic

Memory Basic

Elasticsearch



```
August 17th 2018, 19:03:30.801   log: [2018-08-18 02:03:30,801] {jobs.py:378} INFO - Processing /root/airflow/dags/
                                 mtd_tx_usage_load.py took 0.428 seconds\n kubernetes.pod_name: pn-gsd-airflow-air
                                 flow-2587458194-79ts7 kubernetes.namespace_name: pn-gsd-airflow
                                 kubernetes.labels.app: airflow kubernetes.labels.release: pn-gsd-airflow
                                 kubernetes.annotations.kubernetes_io/created-by: {\"kind\":\"SerializedReference

August 17th 2018, 19:03:30.793   kubernetes.pod_name: pn-gsd-airflow-airflow-2587458194-79ts7
                                 kubernetes.namespace_name: pn-gsd-airflow kubernetes.labels.app: airflow
```

K8sExecutor Status

- Has been released with Airflow 1.10 in experimental mode
- Multiple companies already using in production
- Helm chart in progress
- AirflowOperator by end of 2018
- Active community in #sig-big-data on kubernetes.slack.com
- Seeking beta testers, devs, and brave souls

Airflow Operator Status

- Supports Airflow 1.10.1
- Available on Kubernetes Marketplace in GCP
- Slack channels kubernetes.slack.com

  #sig-big-data

  #airflow-operator

Demo

Thank You

Learn more:

github.com/apache/incubator-airflow/

@danimberman

github.com/GoogleCloudPlatform/airflow-operator

@bharanis

KubeCon | CloudNativeCon

North America 2018