



When Airflow Meets YuniKorn

Enhancing Airflow on Kubernetes with YuniKorn for Higher Efficiency

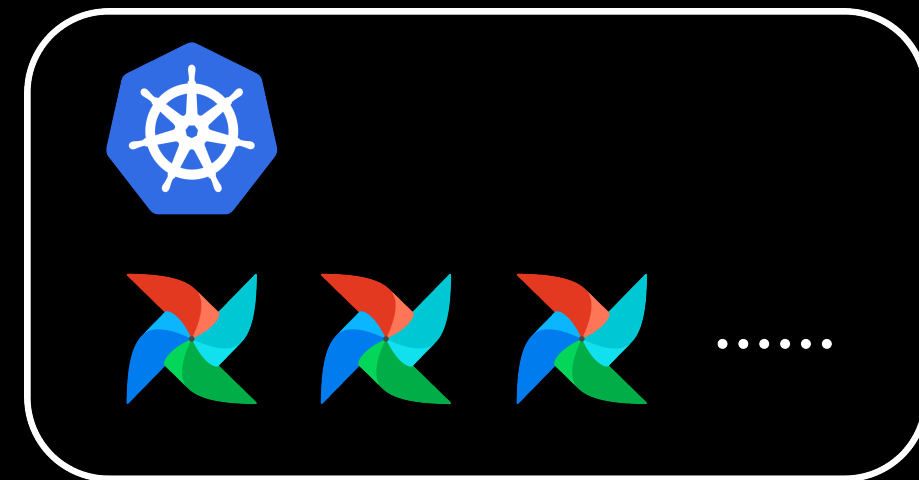


Xiaodong DENG (@XD-DENG, xddeng@apache.org)

About me

- Staff Software Engineer at Apple's Data Platform
- Airflow Committer (since 2019) & PMC member (since 2020)
- Off the clock: I'm a trail runner and paraglider pilot

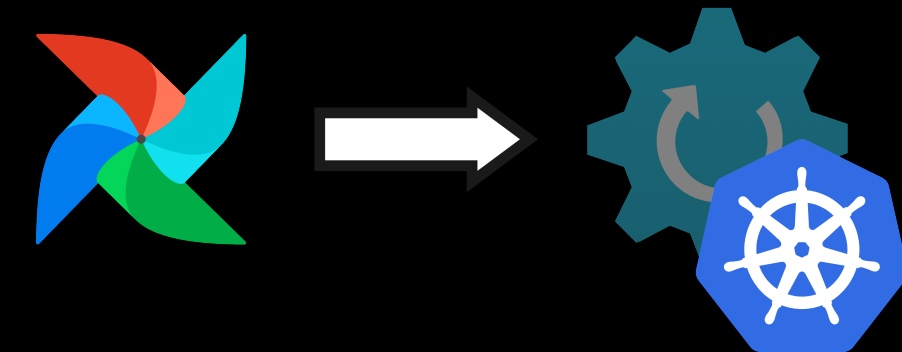
Why using Kubernetes is essential in Airflow?



**Deploy Airflow itself
on Kubernetes**



KubernetesExecutor

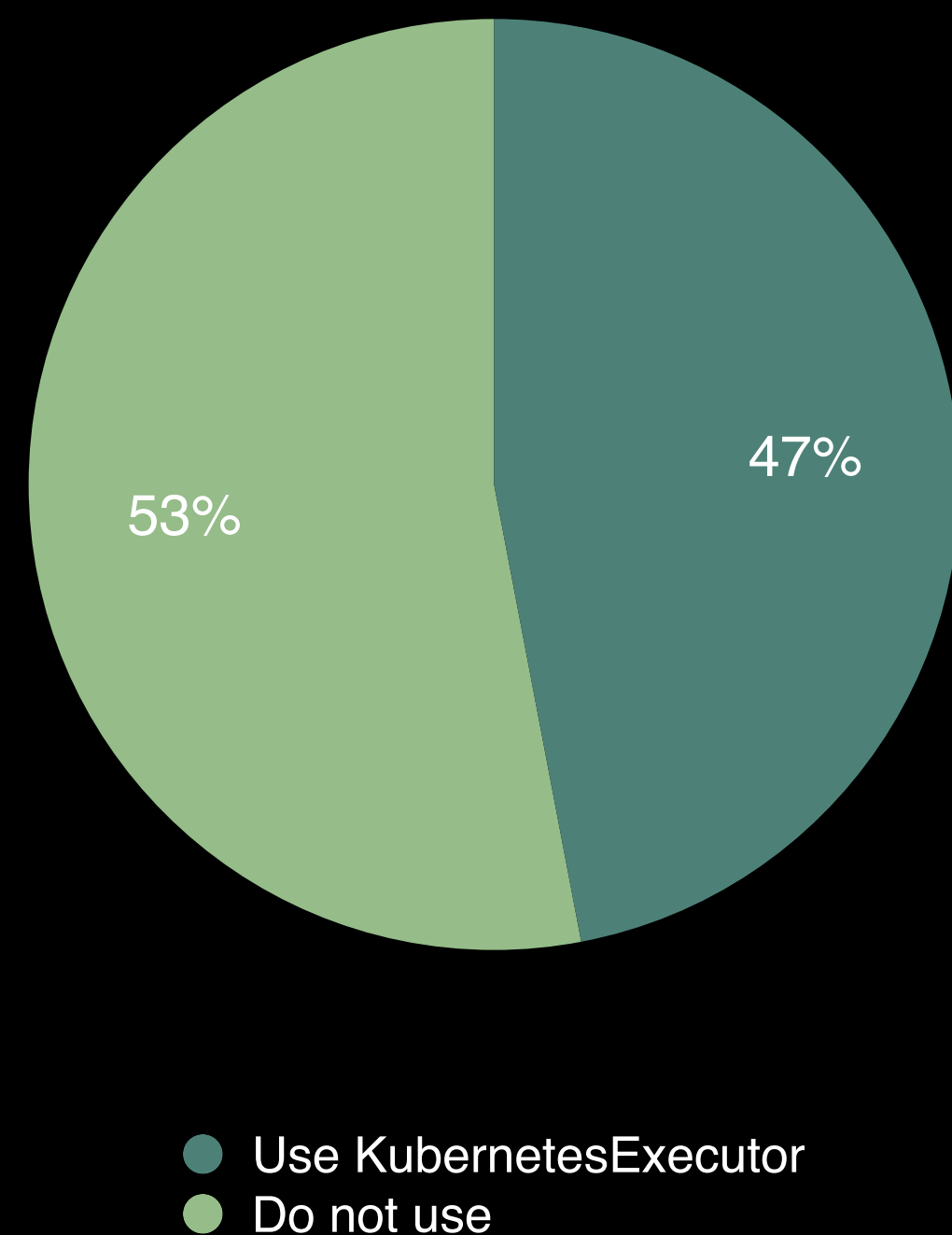


KubernetesPodOperator

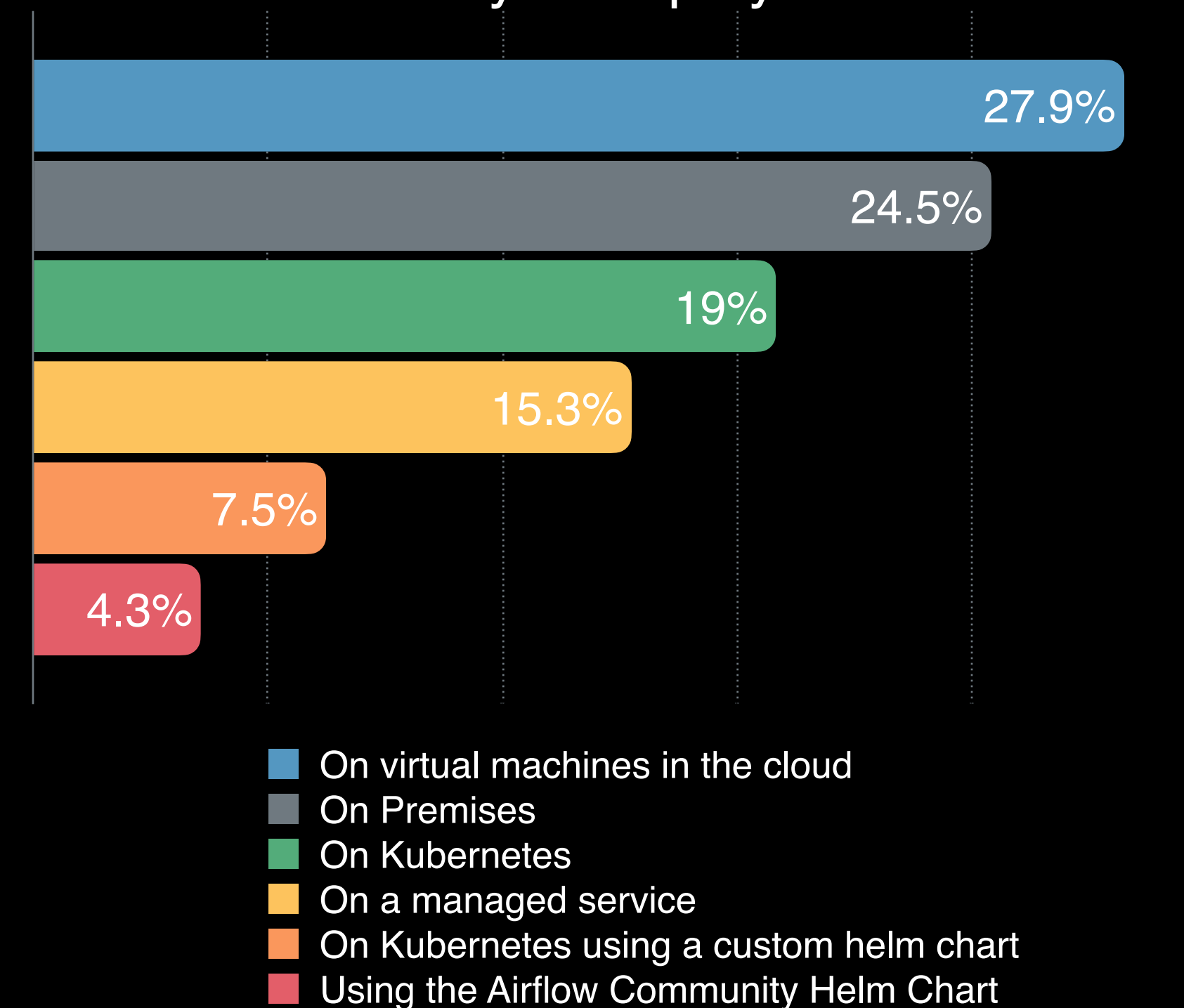
Why using Kubernetes is essential in Airflow?

Insights from 2025 Airflow Survey

KubernetesExecutor



How do you deploy Airflow



“Vanilla” Kubernetes Scheduler may not be enough?

- Lack of “Queue” concept in Kubernetes (meanwhile, in Airflow, Queue is supported in limited Executors)
- Inflexible ResourceQuota
- Suboptimal support for Multi-Tenant Fairness
- Not optimized for handling large-scale batch jobs

Options out there



Kueue



Volcano

And quite many others

...

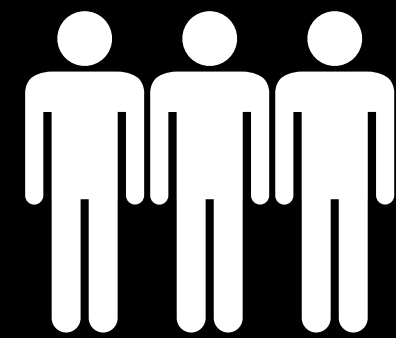
What is Apache YuniKorn?

- Apache YuniKorn is a Kubernetes-native resource scheduler that optimizes efficiency and fairness for mixed workloads in cloud environments.
- Excels at complex scheduling with fairness and high resource utilization
- Key strength in managing multi-tenant, shared clusters for data-intensive applications.

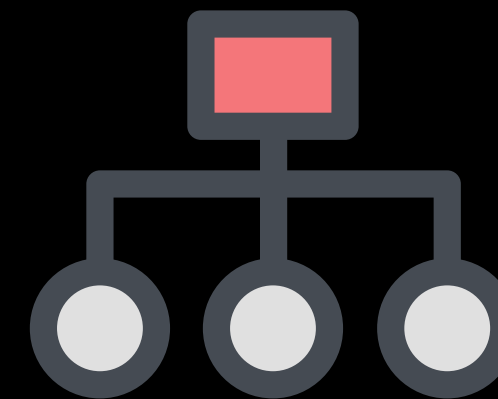
How Does YuniKorn Help?



Application
Awareness



Gang
Scheduling



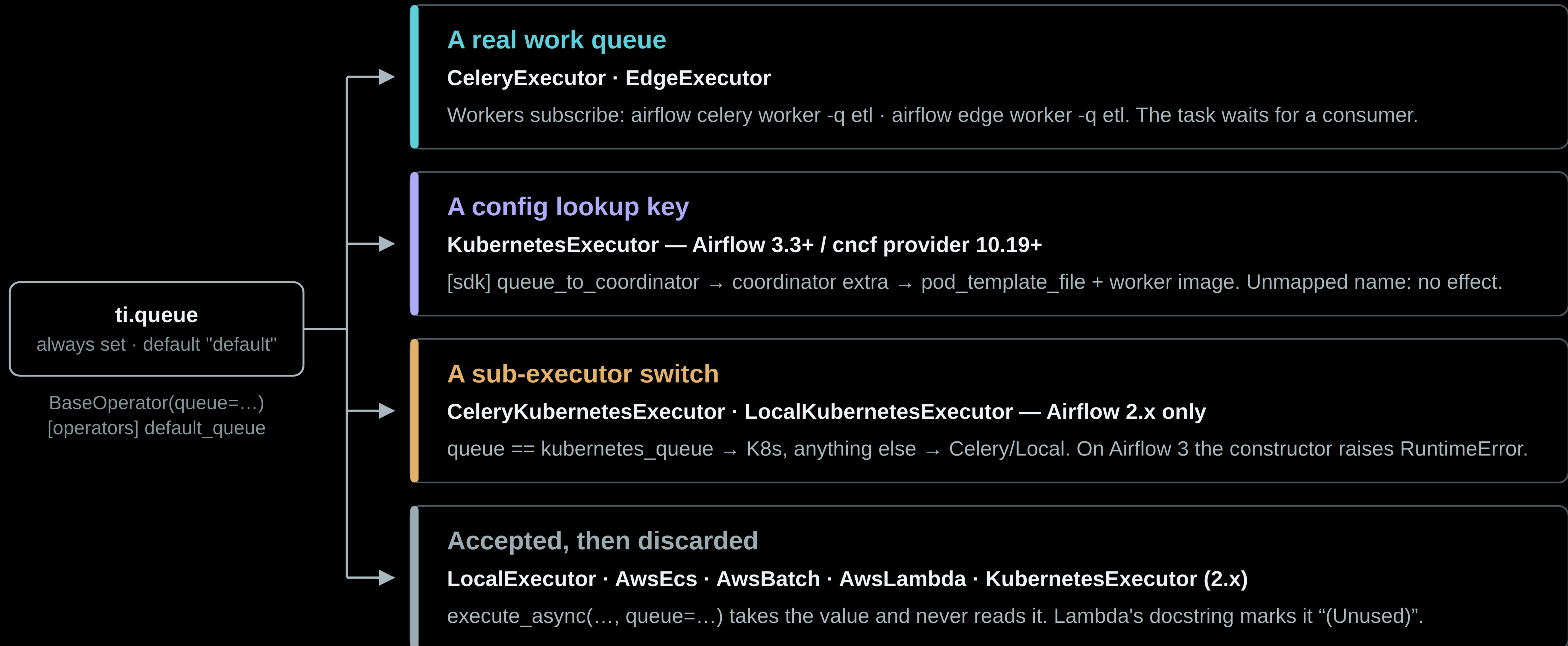
Flexible Resource Queues

- Quota enforcement
- Tenant Fairness
- Flexible mapping to namespaces
- Hierarchical Queues
- Dynamic creation of Child Queues



Ease of
maintenance
& operations

How Does YuniKorn Help - Queue



How Does YuniKorn Help - Queue

Airflow has limited support towards “queue” on Kubernetes.

Plus Kubernetes itself does not support queue.

What if I have use cases below on KubernetesExecutor:

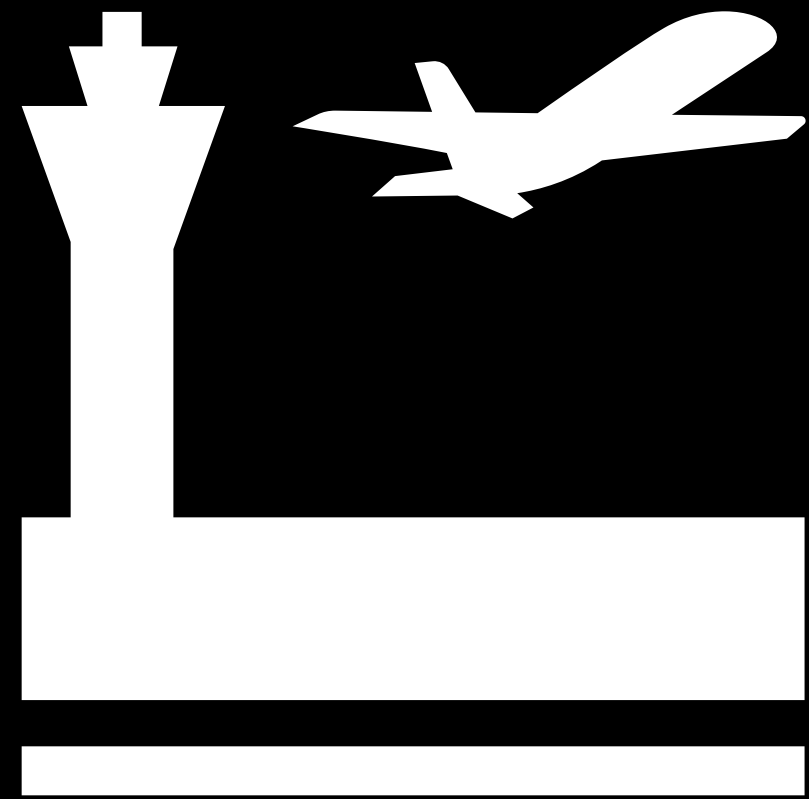
- *“Each tenant/group of tasks should only use certain amount of resources”*
- *“Isolate jobs to avoid noisy neighbors”*

How Does YuniKorn Help - Queue

```
apiVersion: v1
kind: Pod
metadata:
  name: your-pod
  namespace: your-namespace
  labels:
    yunikorn.apache.org/queue: your-queue
spec:
  ... ..
```

Let's demo

How Does YuniKorn Help - Gang Scheduling



Let's take a **flight** as example.

- Will you fly away without your passengers?
- Will you fly without your pilots?
- Can you fly without your airplane?

How Does YuniKorn Help - Gang Scheduling

A: "I have multiple Task Instances in different DAGs. For reasons a/b/c/...., I would like to let all of them to be scheduled at the same time"

B: "You can re-structure your DAGs, e.g. to put all of them into one DAG, and use the DAG structure to achieve what you want"

A: "But how can we guarantee all these worker pods will be scheduled at the same time?"

How Does YuniKorn Help - Gang Scheduling

```
apiVersion: v1
kind: Pod
metadata:
  name: airplane
  labels:
    yunikorn.apache.org/app-id: "flight"
  annotations:
    yunikorn.apache.org/task-group-name: airplane
    yunikorn.apache.org/task-groups: |-
      [{
        "name": "airplne",
        "minMember": 1,
        "minResource": {
          "cpu": "5",
          "memory": "20Gi"
        }
      },
      {
        "name": "pilots",
        "minMember": 2
      },
      {
        "name": "flight-attendants",
        "minMember": 4
      },
      {
        "name": "passengers",
        "minMember": 50
      }
    ]
spec:
  schedulerName: yunikorn
  ...
```

```
---
apiVersion: v1
kind: Pod
metadata:
  name: pilot
  labels:
    yunikorn.apache.org/app-id: "flight"
  annotations:
    yunikorn.apache.org/task-group-name: pilots
spec:
  schedulerName: yunikorn
  ...
```

```
---
apiVersion: v1
kind: Pod
metadata:
  name: flight-attendant
  labels:
    yunikorn.apache.org/app-id: "flight"
  annotations:
    yunikorn.apache.org/task-group-name: flight-attendants
spec:
  schedulerName: yunikorn
  ...
```

```
---
apiVersion: v1
kind: Pod
metadata:
  name: passenger
  labels:
    yunikorn.apache.org/app-id: "flight"
  annotations:
    yunikorn.apache.org/task-group-name: passengers
spec:
  schedulerName: yunikorn
  ...
```

```
apiVersion: v1
kind: Pod
metadata:
  name: airplane
  labels:
    yunikorn.apache.org/app-id: "flight"
  annotations:
    yunikorn.apache.org/task-group-name: airplane
    yunikorn.apache.org/task-groups: |-
      [{
        "name": "airplane",
        "minMember": 1,
        "minResource": {
          "cpu": "5",
          "memory": "20Gi"
        }
      },
      {
        "name": "pilots",
        "minMember": 2
      },
      {
        "name": "flight-attendants",
        "minMember": 4
      },
      {
        "name": "passengers",
        "minMember": 50
      }
    ]
spec:
  schedulerName: yunikorn
  ...
```

```
apiVersion: v1
kind: Pod
metadata:
  name: airplane
  labels:
    yunikorn.apache.org/app-id: "flight"
  annotations:
    yunikorn.apache.org/task-group-name: airplane
    yunikorn.apache.org/task-groups: |-
      [{
        "name": "airplane",
        "minMember": 1,
        "minResource": {
          "cpu": "5",
          "memory": "20Gi"
        }
      },
      {
        "name": "pilots",
        "minMember": 2
      },
      {
        "name": "flight-attendants",
        "minMember": 4
      },
      {
        "name": "passengers",
        "minMember": 50
      }
    ]
```

spec:

```
schedulerName: yunikorn
```

...

Application ID

Task Groups Sepc

Indicate to use Yunikorn as the scheduler

How Does YuniKorn Help - Gang Scheduling

```
with DAG(
    dag_id="demo",
    schedule=None,
    start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),
) as dag:

    executor_config_template = {
        "pod_override": k8s.V1Pod(
            spec=k8s.V1PodSpec(
                scheduler_name="yunikorn"
            ),
            metadata=k8s.V1ObjectMeta(
                labels={
                    "yunikorn.apache.org/app-id": "flight"
                },
                annotations={
                    "yunikorn.apache.org/task-group-name": "pilots"
                }
            )
        ),
    }

    @task(executor_config=executor_config_template)
    def pilot():
        pilot_get_ready()
```

Let's demo

Looking Ahead



Thanks