# Spark on Kubernetes - The Elastic Story

Bowen Li, Huichao Zhao | KubeCon + CloudNativeCon Europe 2022 | 16 May - 20 May 2022

# Agenda

Benefits of Cloud

Design Principles & Architecture of Cloud-Native Spark Service

Autoscaling Spark - Cost Saving & Elasticity Needs

Design of Reactive AutoScaling

Productization of Reactive Autoscaling

Learnings & Future Work

# Benefits of Cloud

| |
|---|
| **Agile**<br>Resources are on-demand, pay as you go |
| **Elastic & Scalable**<br>Almost infinite scale of compute and storage |
| **Strong Resource Isolation**<br>Container-native on K8S |
| **Privacy-First**<br>Leverage cloud security techniques to enforce security |
| **Operation Friendly**<br>Our developers can focus on building and improving services to achieve higher ROI |

# Design Principles

Leverage Cloud Infra

Full Containerizing - Elastic, Agile, Lightweight

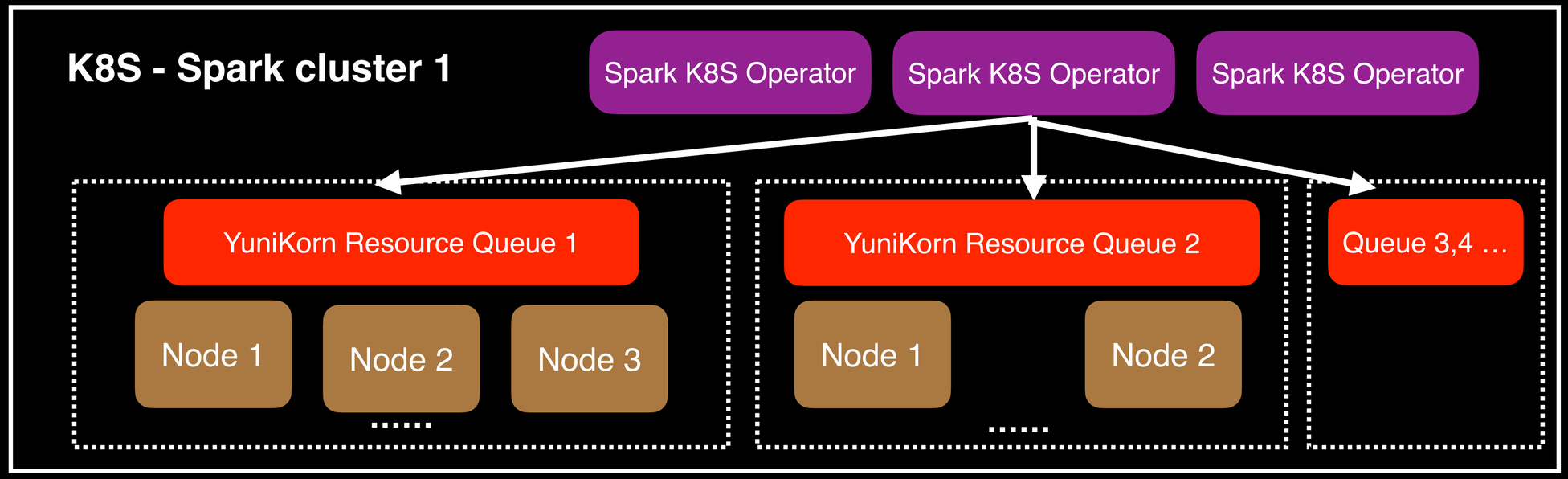
Decouple Compute/Storage, Scale Independently

Developer-Friendly, API Centric

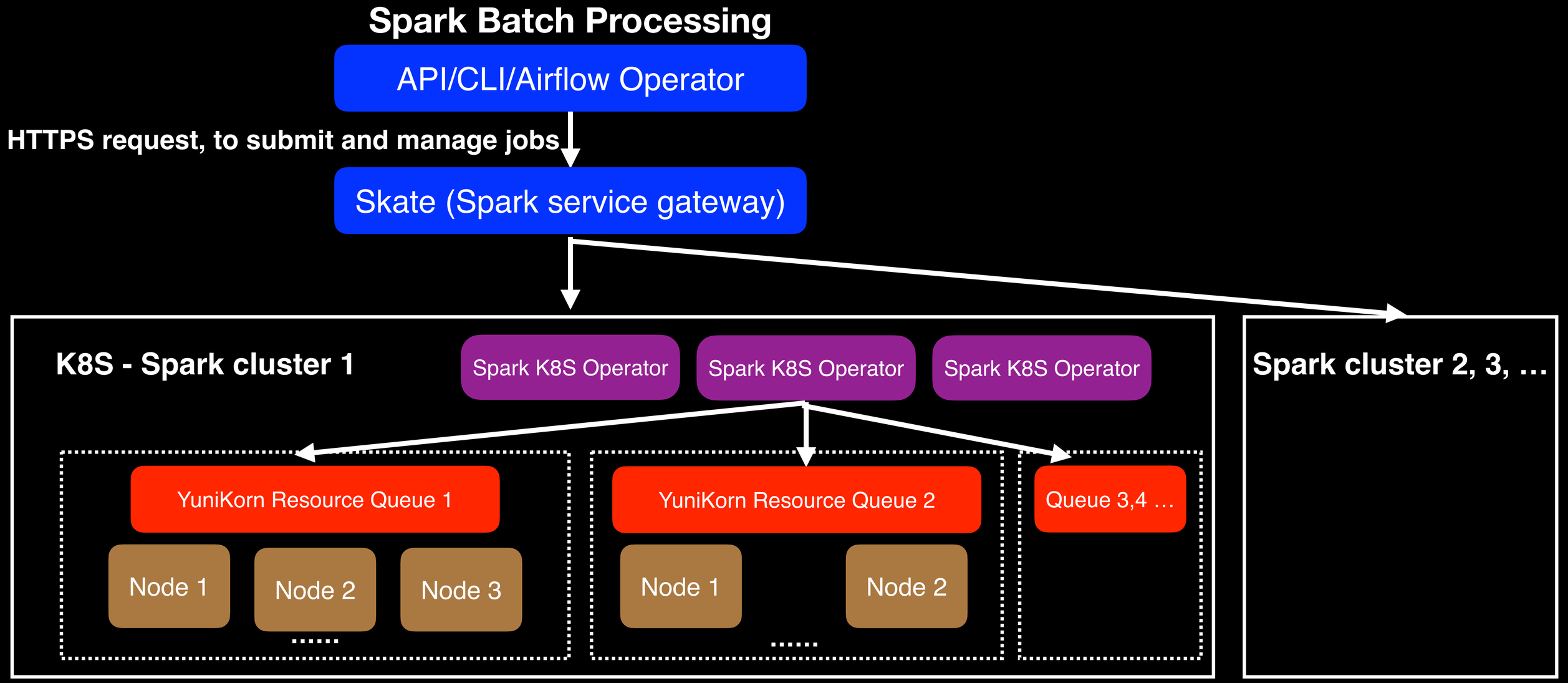
Security & Privacy as First Class Citizen

Use Apple Internal Spark distribution

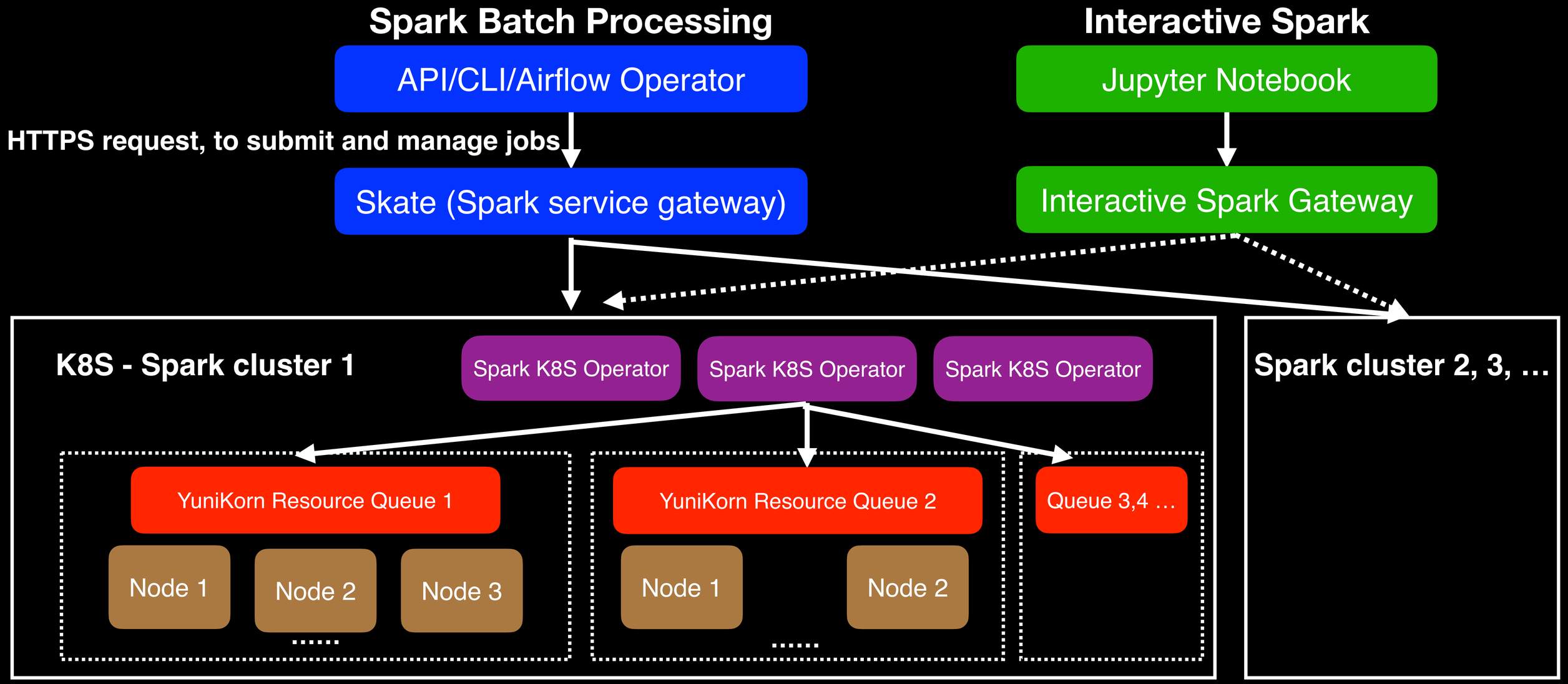
# Architecture of Cloud-Native Spark Service



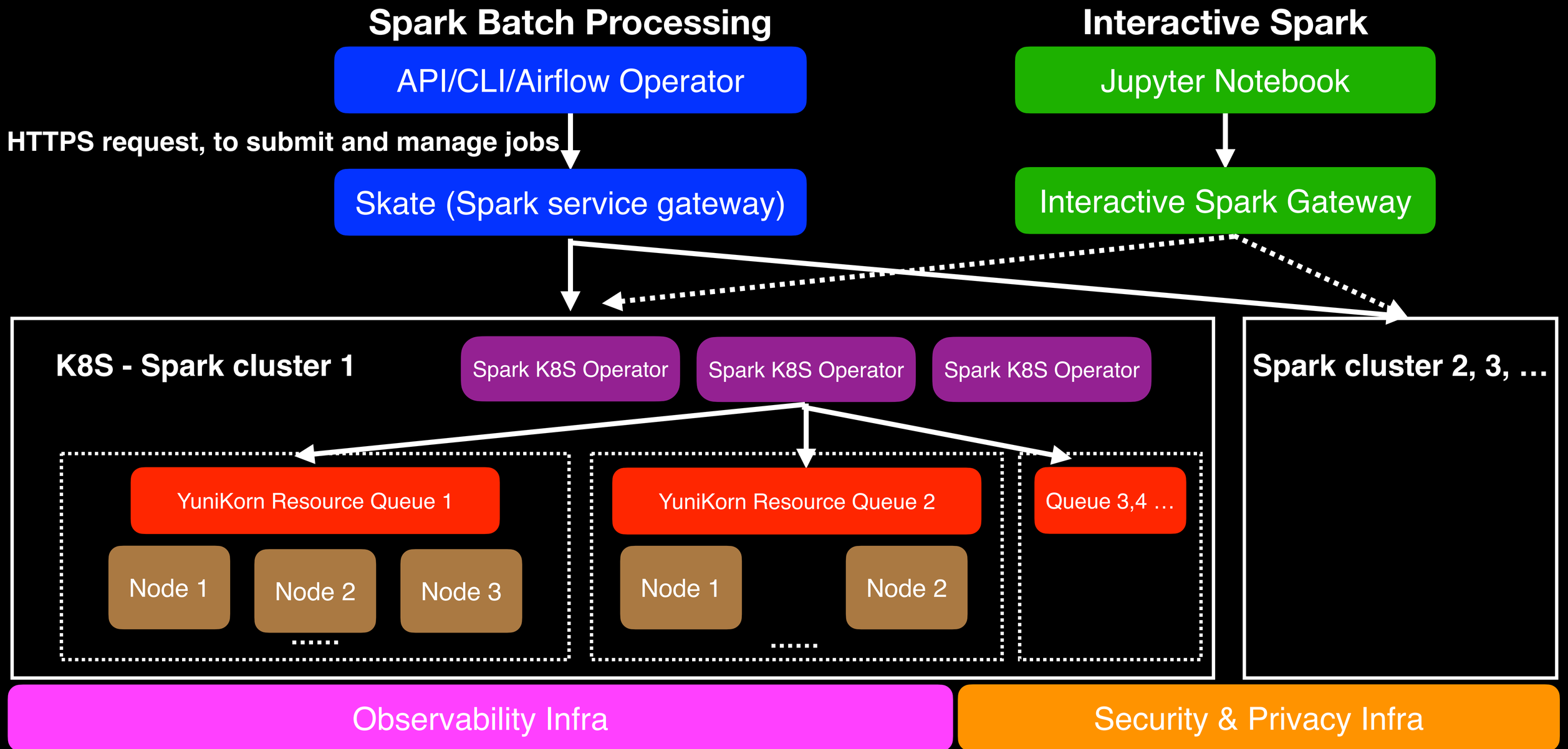
# Architecture of Cloud-Native Spark Service



# Architecture of Cloud-Native Spark Service



# Architecture of Cloud-Native Spark Service



# Cost saving and Elasticity Needs

Varying workload pattern: fluctuating within a day and/or a week

Different use cases: daily/weekly scheduled jobs, ad hoc jobs, scheduled + adhoc, backfill

Fixed amount of resources must account for max usage, which causes resource waste

# Agenda

Benefits of Cloud

Design Principles & Architecture of Cloud-Native Spark Service

Autoscaling Spark - Cost Saving & Elasticity Needs

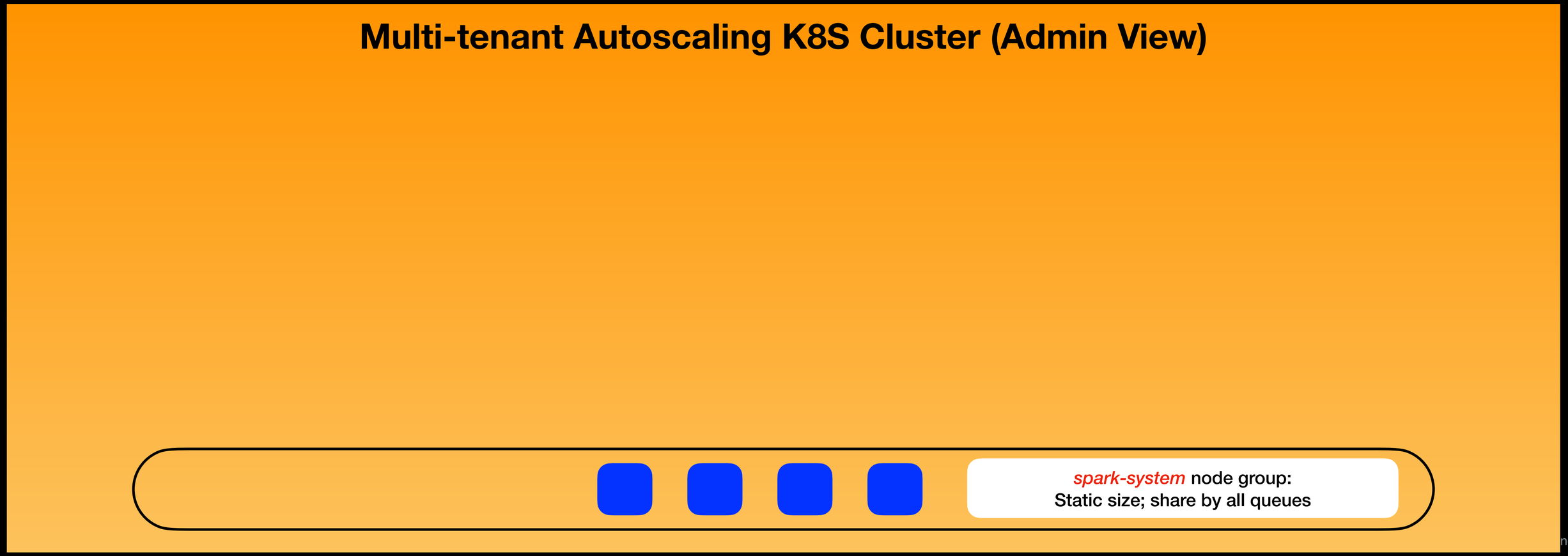
Design of Reactive AutoScaling

Productization of Reactive Autoscaling

Learnings & Future Work

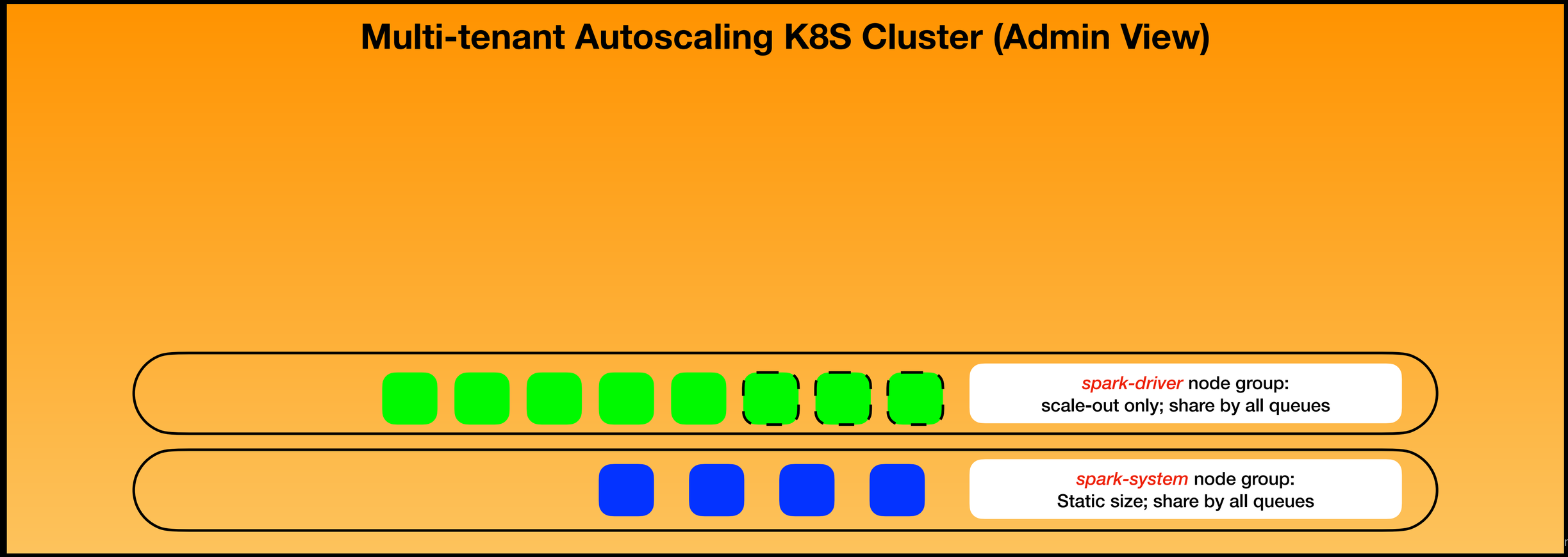
# Reactive AutoScaling Cluster NodeGroups Layout

- Physical isolation:  Minimize potential impact



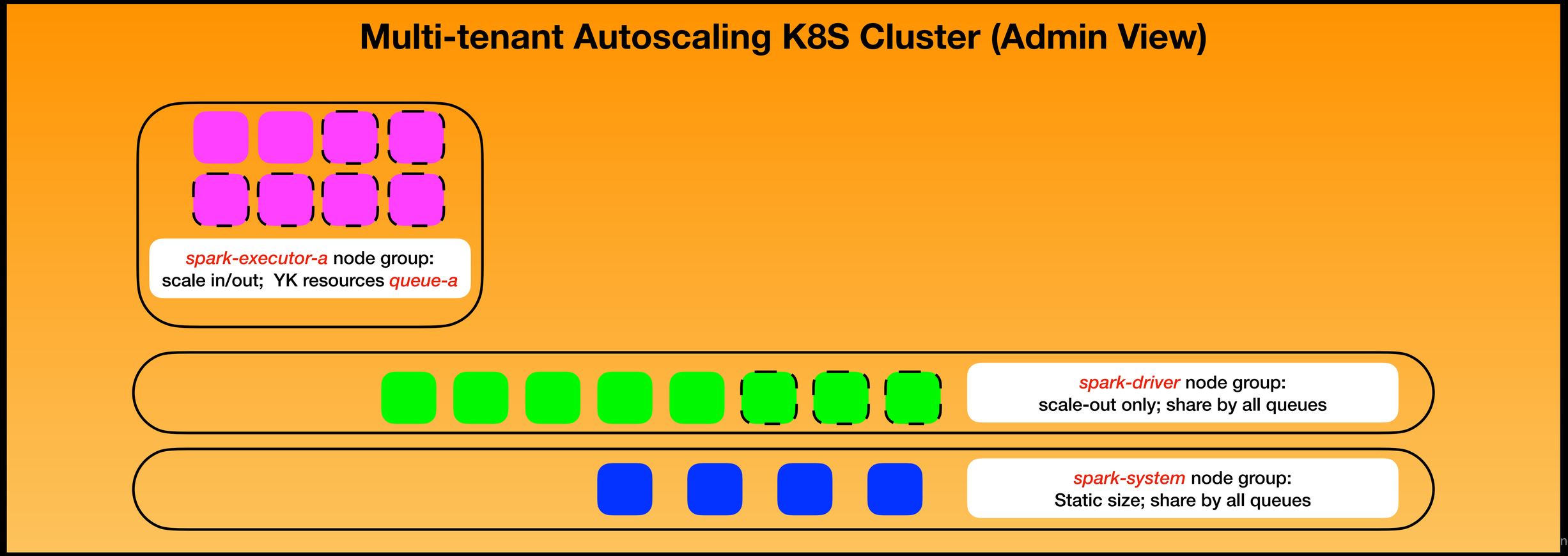
# Reactive AutoScaling Cluster NodeGroups Layout

- Physical isolation:  Minimize potential impact



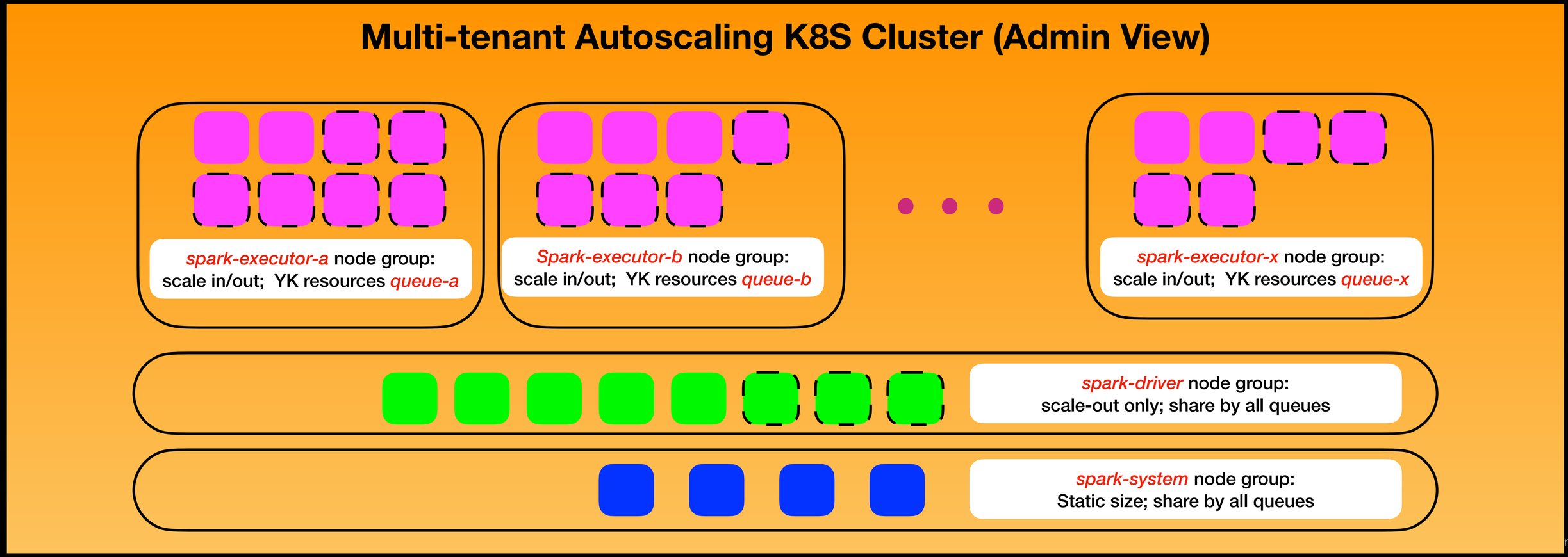
# Reactive AutoScaling Cluster NodeGroups Layout

- Physical isolation:  Minimize potential impact



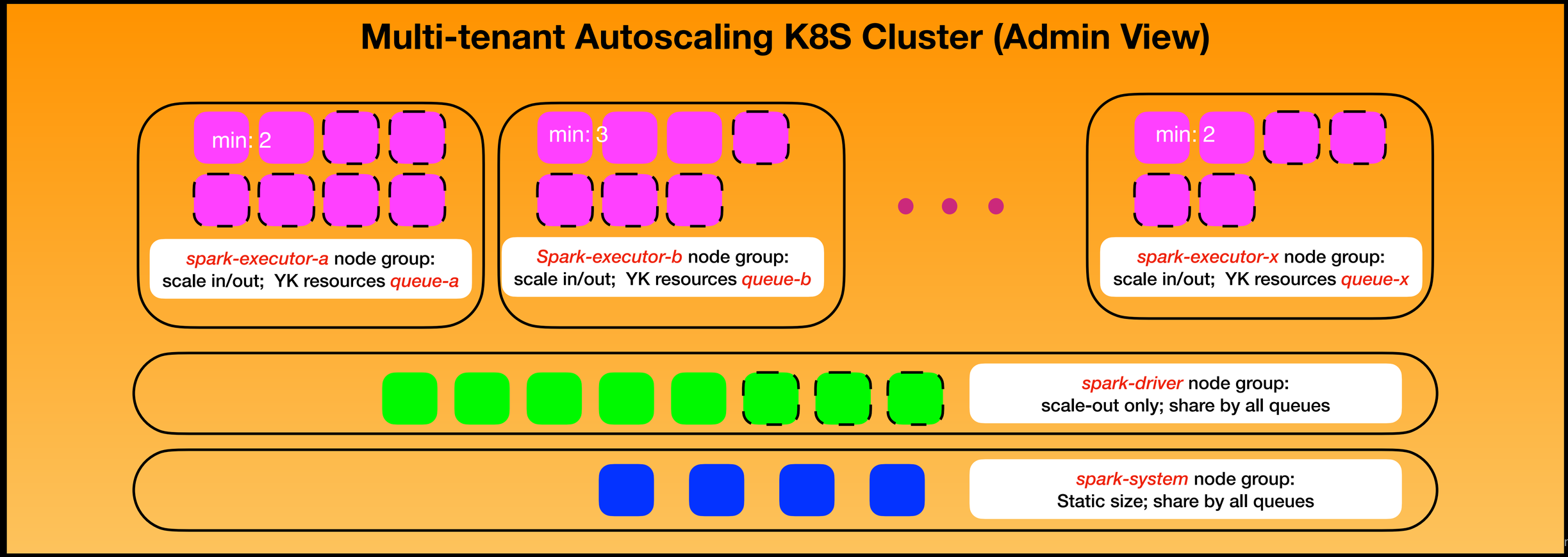
# Reactive AutoScaling Cluster NodeGroups Layout

- Physical isolation:  Minimize potential impact



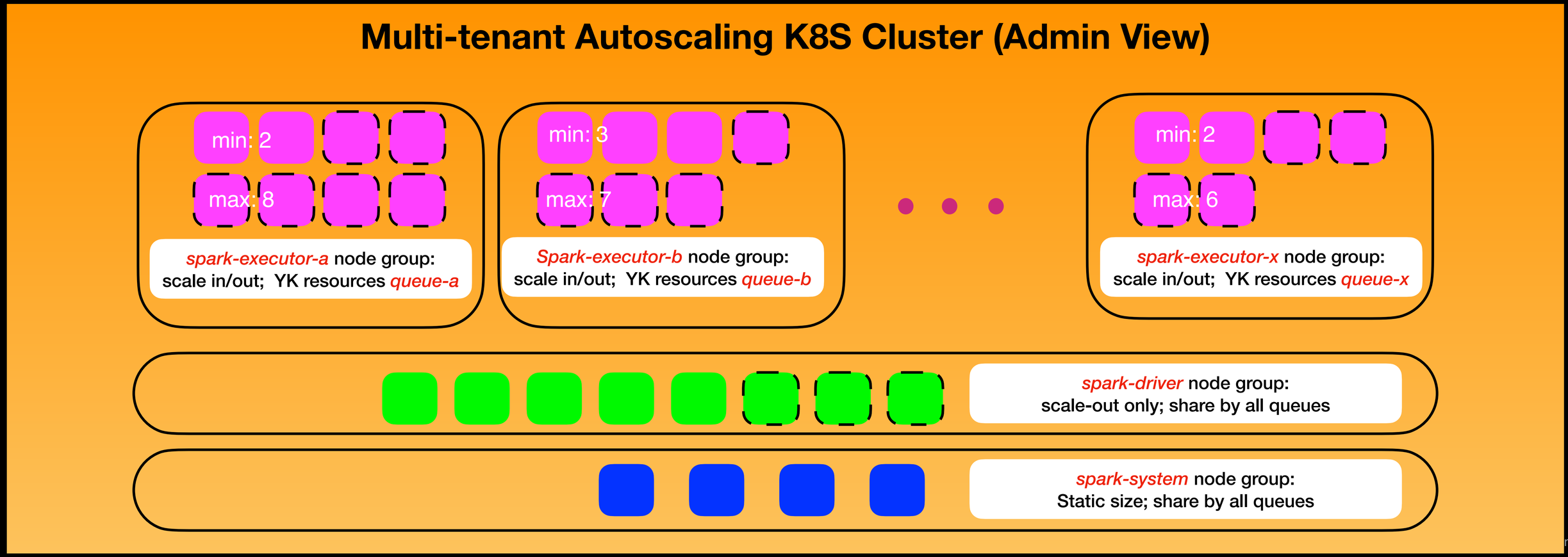
# Reactive AutoScaling Cluster NodeGroups Layout

- Physical isolation: Minimize potential impact
- Minimum capacity: Guaranteed at anytime

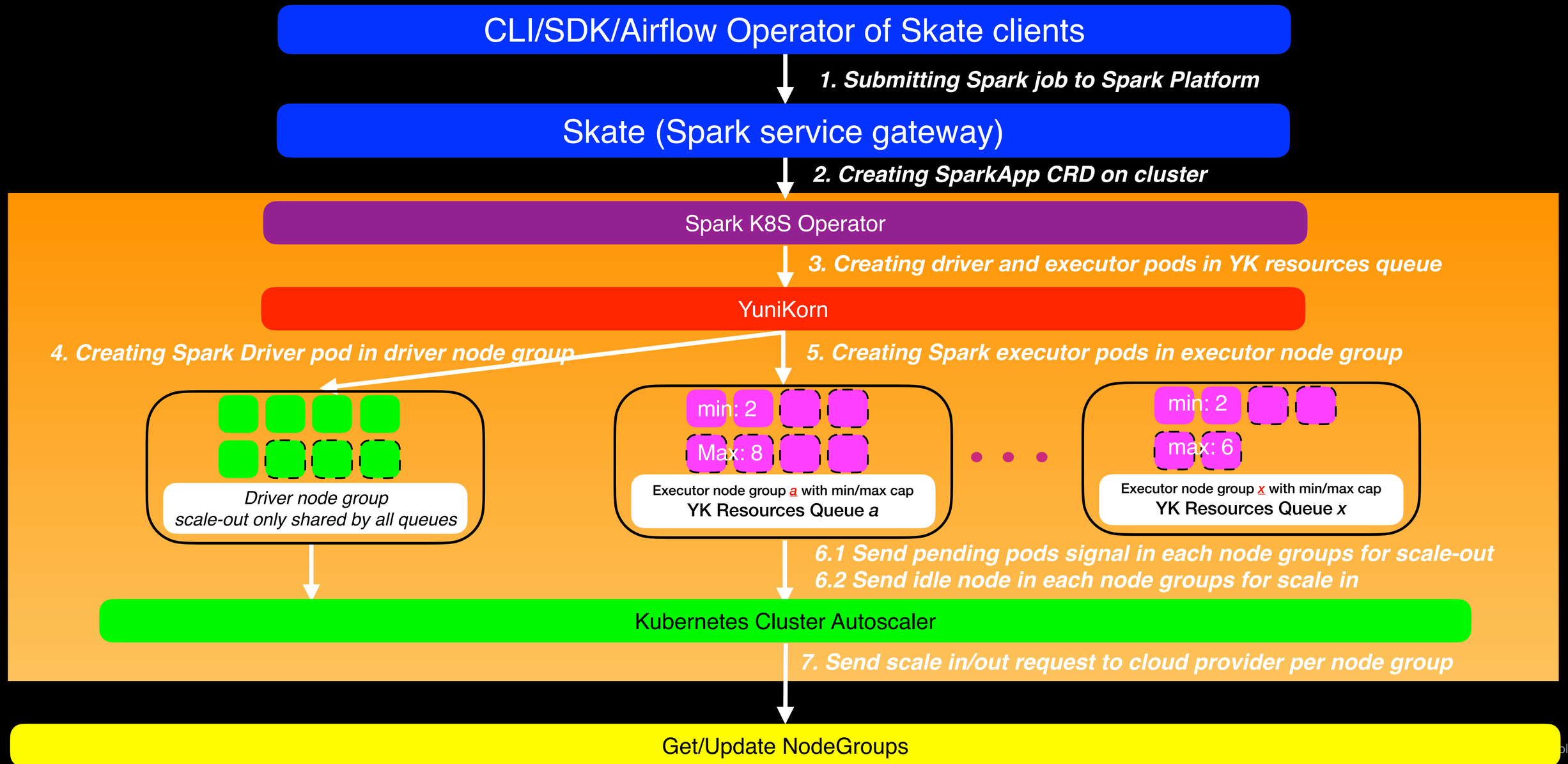


# Reactive AutoScaling Cluster NodeGroups Layout

- Physical isolation:  Minimize potential impact
- Minimum capacity:  Guaranteed at anytime
- Maximum capacity:  Jobs will be queued if exceed



# Reactive AutoScaling Workflow



# Reactive AutoScaling Scale in/out  Controls

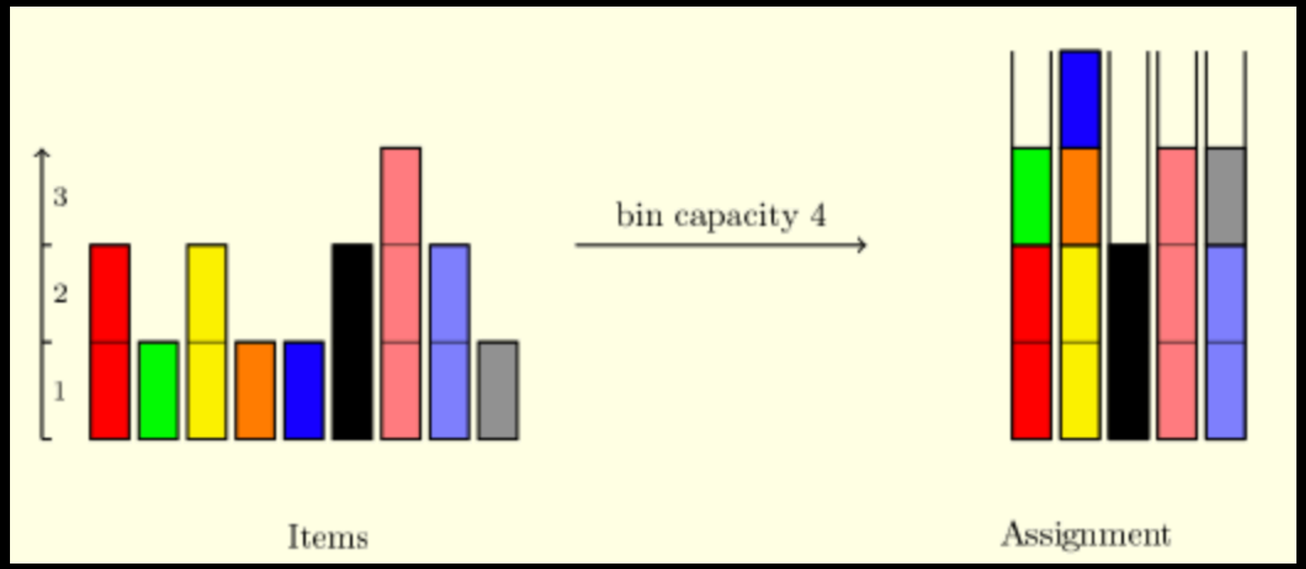
## Scale in Controls

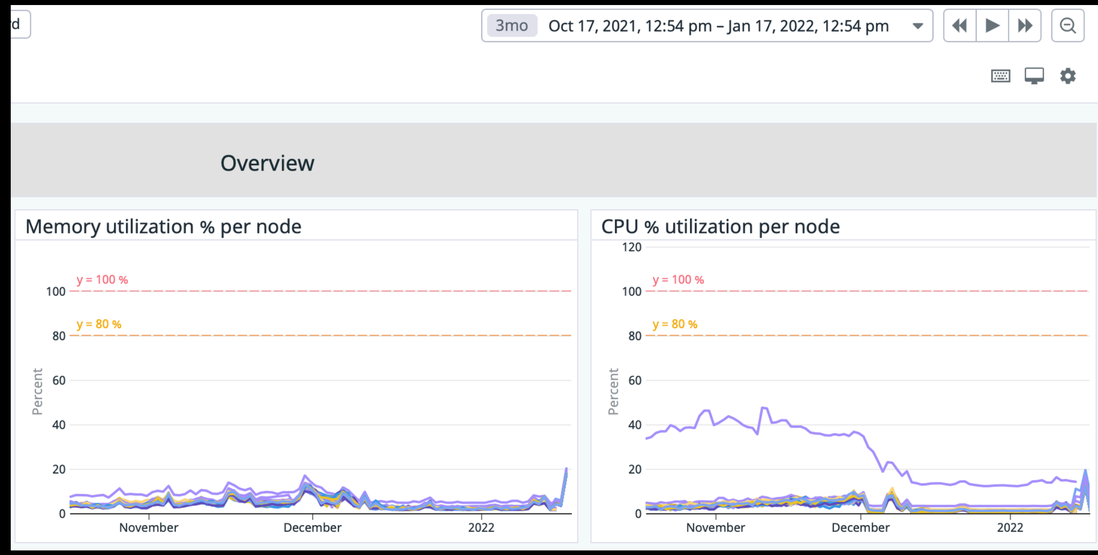
- Only when no running executor pods on the node

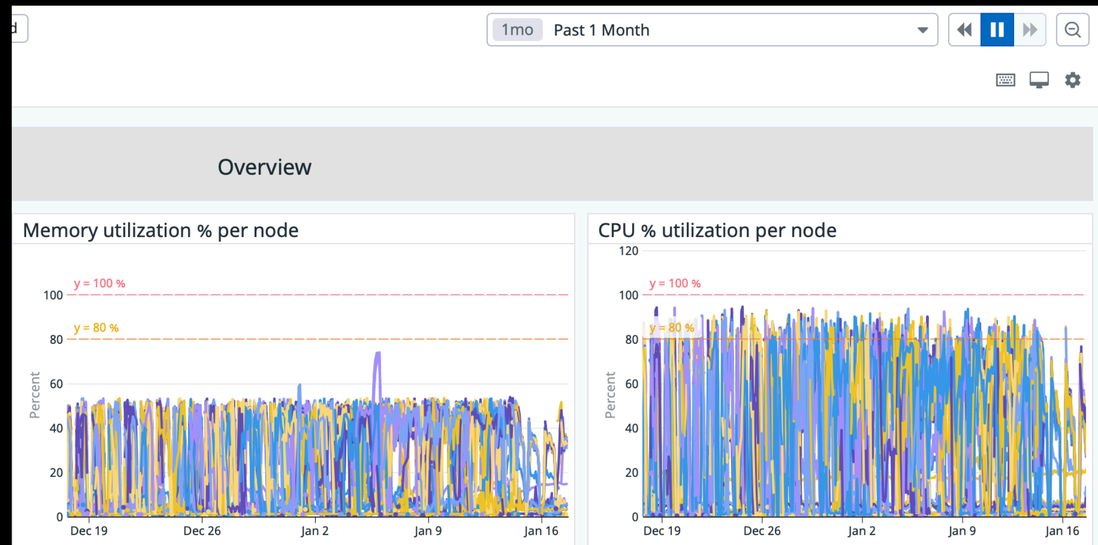
# Reactive AutoScaling Scale in/out  Controls

## Scale in Controls

- Only when no running executor pods on the node
- Enabled Apache YuniKorn bin-packing in resource scheduling



# Reactive AutoScaling Scale in/out  Controls

## Scale out Controls

- Spark-driver node group scale out only
- Speed up executor pods allocation size config of Spark

# Production Status

**In production for 3+ months**

**Cost saving report**

- Cost saving percentage per queue is located in 20% - 70%

# Migration findings 1

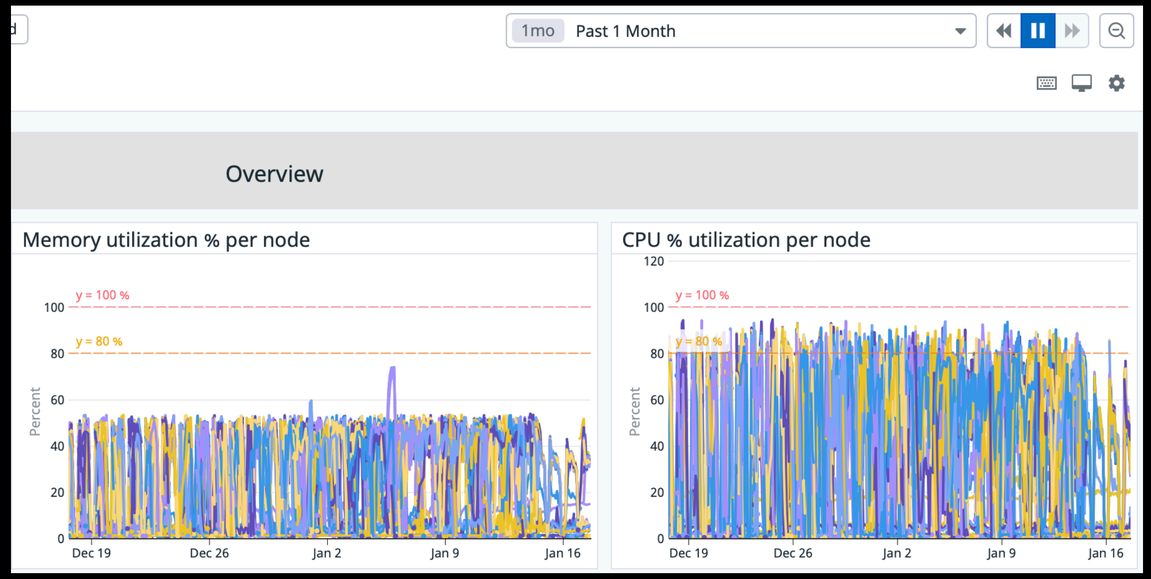
**Scale in/out events are stable**

- Tens of thousands of job per week running successfully
- All scale-in events works as expected
- Scale out latency is consistent ( <= 5 mins  from 2 to 200)

# Migration findings 2

**Compared to massive over-provisioning approach before, runtime of workloads with autoscaling enabled may increase**

- Mostly negligible, a couple jobs increased ~20%
- Users need to take this into consideration and optimize jobs if there's strict data delivery time SLO



# Challenges, Solutions, Learnings

- Physical Isolation and min/max capacity setting
- How to guarantee no impact to existing Spark jobs when scale-in
- How to speed up scale-out latency and always allow Spark driver getting start
- Monitoring autoscaling performance

# Top Community Feature Requests

- Mixed instance type support
- Dynamic Allocation support
- Spot instance support with Remote Shuffle Service
- Predictive Autoscaling leveraging the platform