KubeCon | CloudNativeCon

North America 2017

Evolving & Supporting Stateful, Multi-Tenant Decisioning Applications in Production

B. Frazier, K. Gasser & G. Mead, Software Engineers, *Capital One*

Agenda

- Intro (Keith)
- Cluster Installation and Operations: State Management and “Rehydration”/Upgrades (Bryce)
- Multi-tenancy and PaaS CLI/DSL (Gavin)

Our experience with K8s

- In production at AWS since 2Q17 (K8s v1.6.x)
  - Single Region, Multi-AZ, homogeneous node types
  - Full-stack (from AMI up) compliance-driven ”re-hydrations” of cluster every 60d
- Supporting four types of workloads
  - T1: Real-time decisioning for transaction streams and analytics
  - T2: Batch-based model refit pipelines
  - T3: Ad hoc analytical queries from data analysts
  - T4: Operational workloads (telemetry stacks, cluster services, housekeeping jobs, etc)

Production Workloads

T1/2: DOMAIN

ZooKeeper

Flink

nifi

kafka

redis

CRATE

Pachyderm

T3: ANALYTIC ENVT

Apache Zeppelin

APACHE DRILL

T4: LOGGING

ELASTIC SEARCH

kibana

fluentd

T4: METRICS

Grafana

HEAPSTER

InfluxDB

T4: SERVICES

Istio

SkyDNS

sTUNNEL

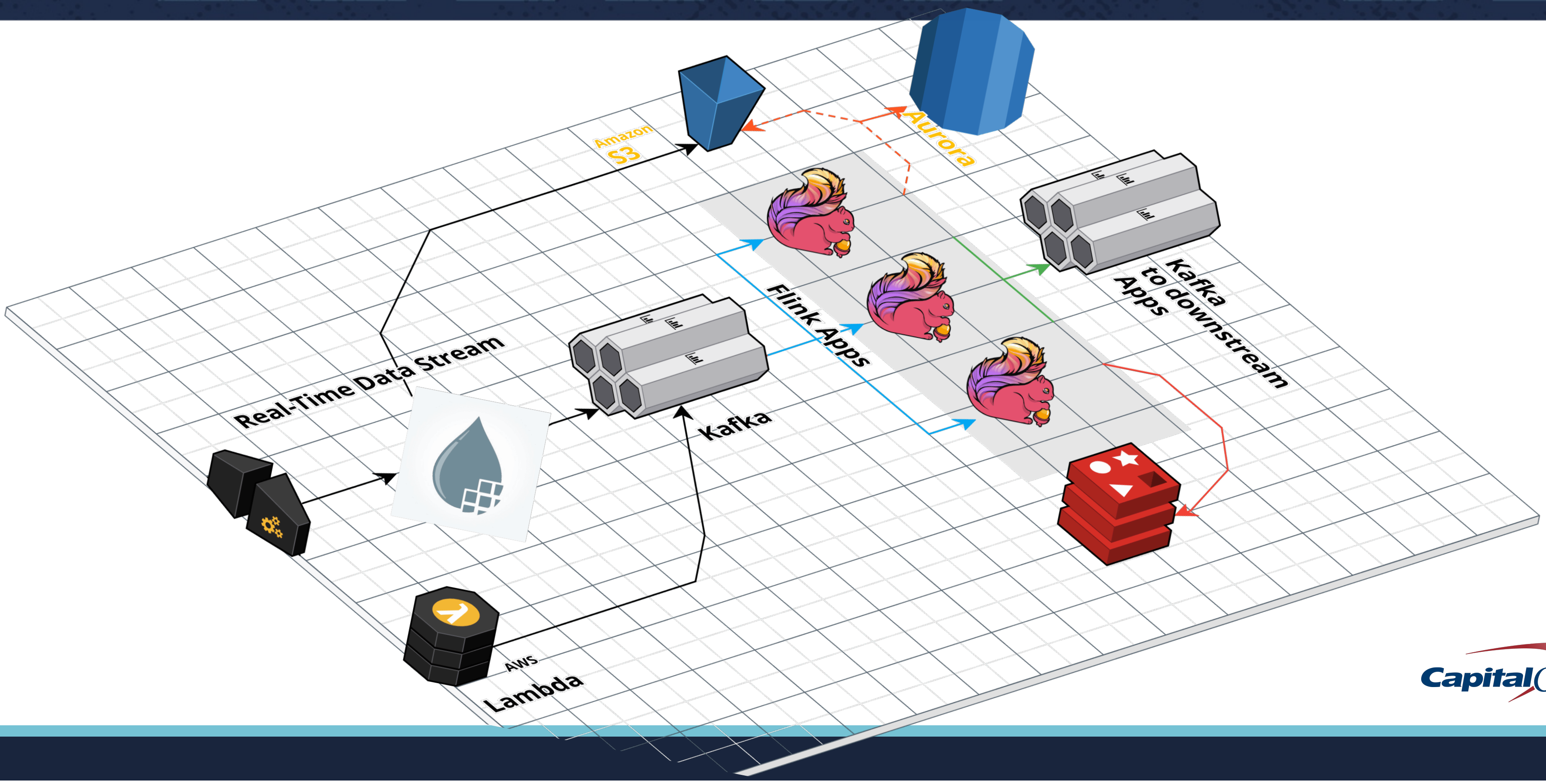
ZIPKIN

etcd

dex

CapitalOne

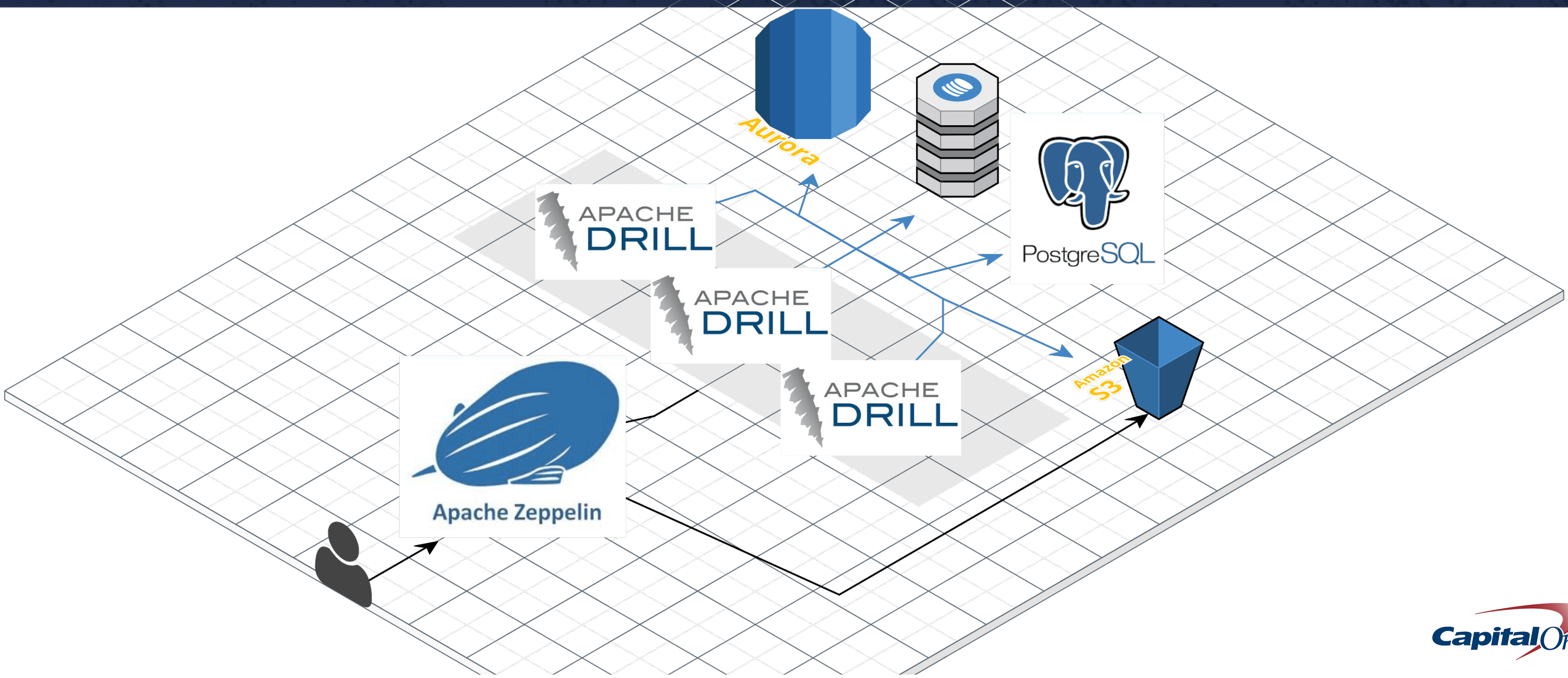
T1: Decisioning Engine



T2: Model Refit with Pachyderm

- Copy on write, similar to Git
- Data Provenance
- Reproducible at scale
- S3-backed
- Reactive batch pipelines

T3: Analytical Environment



T4: Telemetry: Responsive > Reactive

- Metrics and Alerting: Ops and Apps teams share Grafana stack, but separate dashboards.
  - Future state: Separate Grafana stacks isolated by tenant namespace

- Logging: Ops and Apps teams share EFK stack, separate tagging by application, so filterable
  - Future state: Fluentd configurations will forward application log streams to isolated logging aggregators/dashboards as elected by application teams (isolated by tenant namespace if internal to cluster)

Def. State and Multi-Tenancy

- Q: What are stateful workloads?
  - Stateful sets aka ”Petsets”: e.g., Kafka topics
- Q: What do you mean by “multi-tenancy”? Isn’t K8s already multi-tenant?
  - Not without sufficiently isolated workloads
  - Many services designed to be shared (e.g. telemetry stacks, Zookeeper ensemble, Flink cluster)
  - Namespaces don’t solve all forms of isolation
  - Painpoints at scale with differently workload resource demands

Customer interactions…

- “I want my own K8s cluster.”
- “I want my own Flink cluster.”
- “I want access to the K8s dashboard.”
- “I want this much resource…”
- “I want elasticity…”

Your experience?

- How many in production?
- … with state?
- … … with multi-tenancy?

Value to customers – a “managed service”

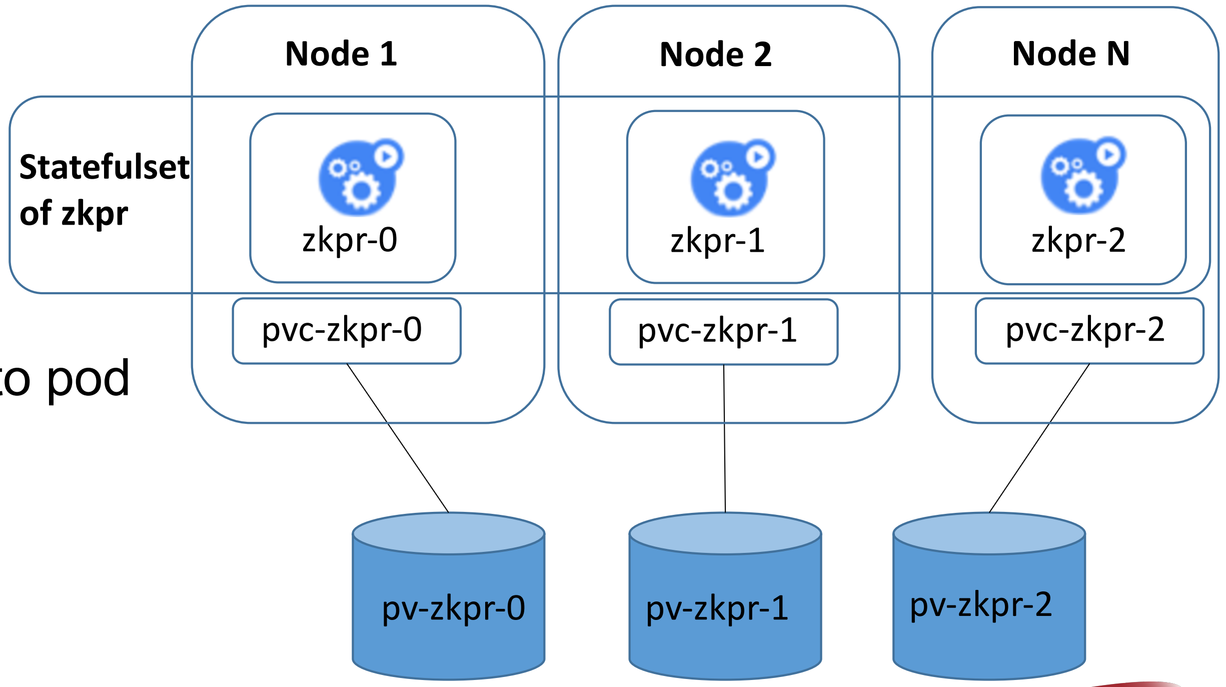
- Free from 60d Compliance “Rehydration” Requirement
- K8s “with benefits”
    - ++Cloud Engineering
        - ++Installation
        - ++Persistent State
        - ++Upgrades/Patching
        - ++Streamlined Security
        - ++Resiliency Engineering
    - ++Common Telemetry Services: Logging/Monitoring
    - ++Common Domain Services: Data + App Infrastructure

Tenant Isolation - Namespacing

- Independent Deployments
- Locked down User policies
  - Authn – Dex
  - Authz – RBAC
- Network Policies via Calico

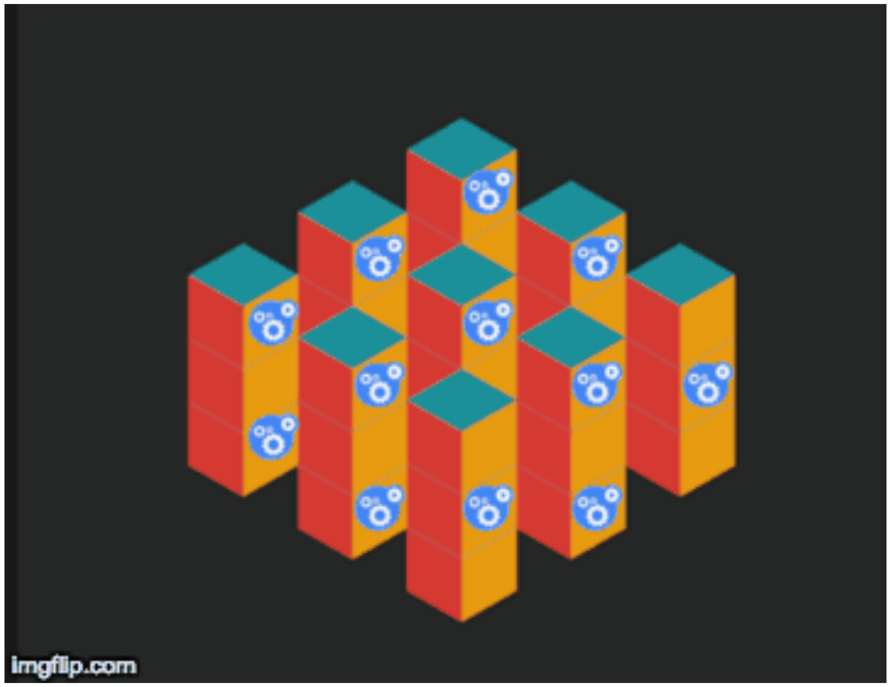
Stateful Applications & Pod-Volume Affinity

- Persistent Volumes
  - Piece of storage, analogous to node
- Persistent Volume Claims
  - A request for storage, similar to pod
- Stateful Sets
  - Unique id
- Storage Classes



Sidebar: Automated Upgrades & “Rehydration”

- Rehydration is a compliance req.
  - AMIs actually deprecated after 60d
- A Kubernetes job (!!!)
- Validates healthy cluster
  BEFORE every step
- Scales out, drains each node, scales in
- ~2.5 hours for full upgrade



Lessons Learned: Safety First!

- Pod Anti-affinity (curse of fat pods)
- Resource Limits, Limit Ranges & Quotas
- Kubelet Resource Management

Future state: Elastic/Dynamic Load

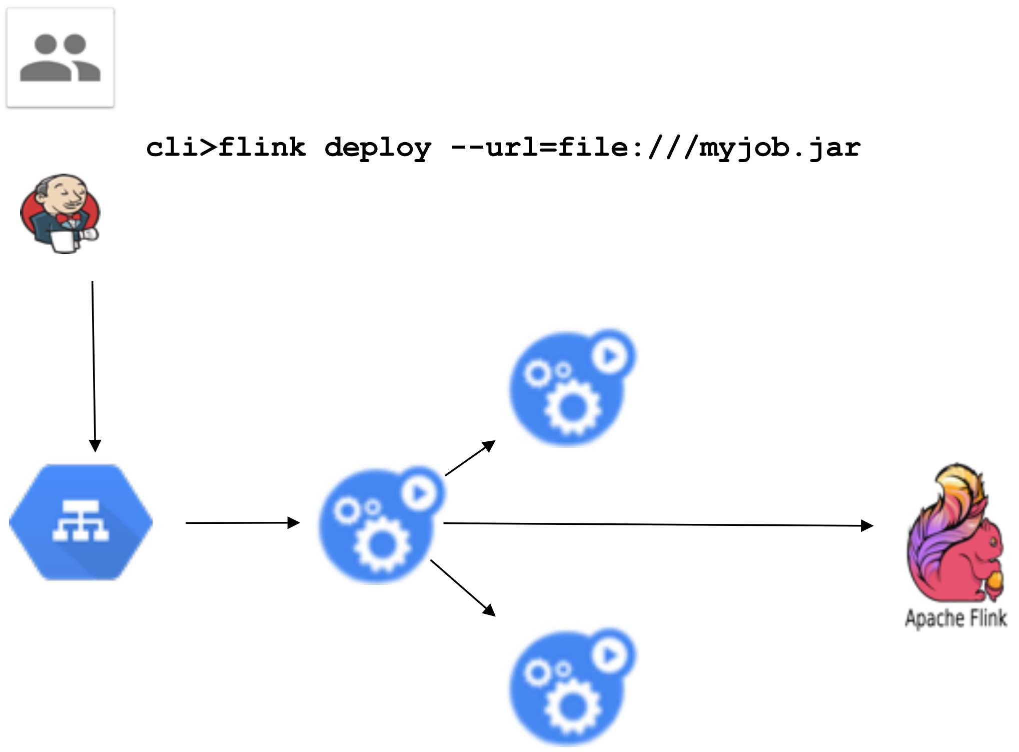
- Pod autoscaler
- Node autoscaler
  - Custom instance types for various loads
  - Taints/Tolerations
- GPU

Kubernetes should be invisible

- Platform is not a general purpose Kubernetes offering
- Kubernetes is an implementation detail of how we deliver our service offerings
- Users were asking for PaaS features like a CLI. So we gave them one.

Current User/Platform Interaction

```
cli>flink deploy --url=file:///myjob.jar
```

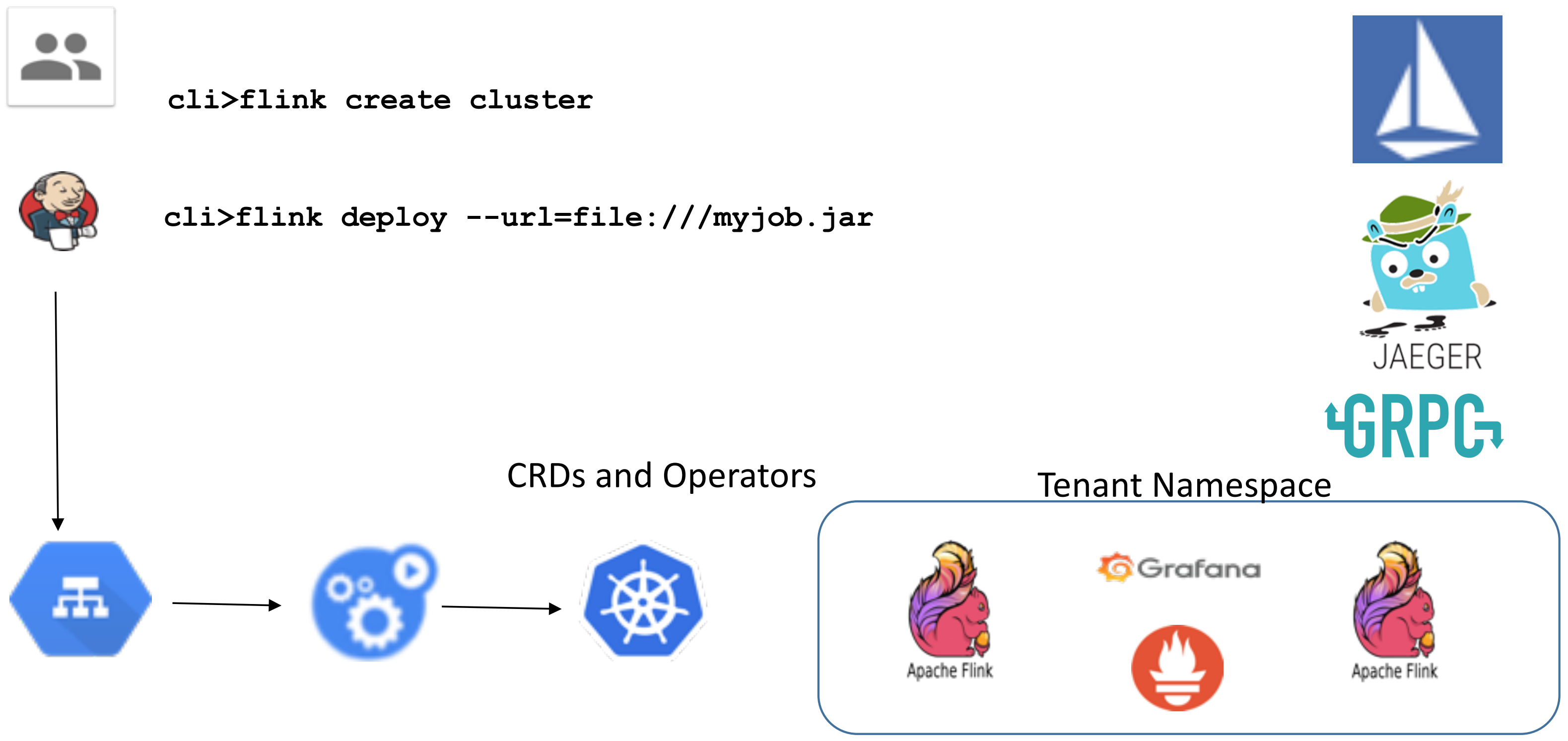


Learning Opportunities

- It is hard to put guard rails around a shared Flink Cluster
- Determining how a Flink Job can affect the overall cluster is difficult
- Users were asking for their own clusters

Future User/Platform Interaction

```
cli>flink create cluster
cli>flink deploy --url=file:///myjob.jar
```



Conclusions

- What is in your K8s “managed service”?
- DSL-based CLI is a good way to insulate users from k8s implementation details
- State will creep in with its “gravity and inertia”
- Unchecked esp. ad hoc workloads have a resource consuming “blast-radius”
- You are likely already multi-tenant, you may not realize it
- Type 4 (ops) workloads will become richer, and continue upward trend in resource consumption (e.g. tracing is now *de rigeur*)
- Clusters supporting streaming services still need R/R services: REST, gRPC
- Given k8s extensibility and WIP, specialized clusters with CDRs, operators for domain-specific needs will emerge

Community Shout-out!

- Sam Brown, Organizer of the NOVA-Kubernetes meetup: https://www.meetup.com/NOVA-Kubernetes/ -- please consider attending if you are in the area!

Thank you!