KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

KubeCon CloudNativeCon
Europe 2022

TikTok's Story:

How To Manage a Thousand Applications on Edge With Argo CD

Qingkun Li - TikTok/Bytedance, Inc.
Jesse Suen - Akuity, Inc.

TikTok's Story:

How To Manage a Thousand Applications on Edge With Argo CD

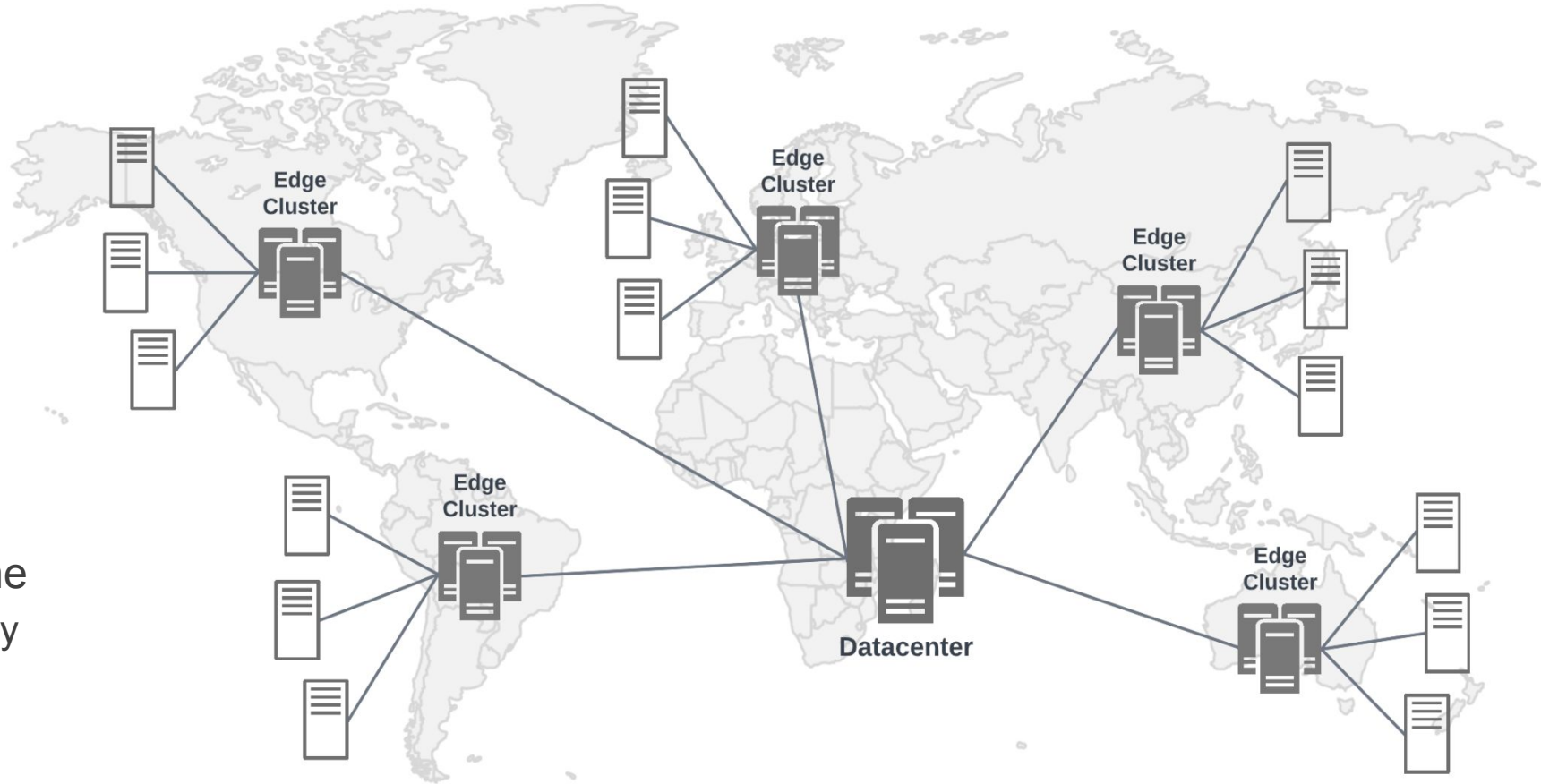
This talk describes a case study of how TikTok manages 3000 applications across 100 global edge clusters with Argo CD. We discuss considerations, tips, and techniques for using Argo CD to manage cluster applications on the edge.

Qingkun Li
Tech Lead Manager
TikTok/Bytedance, Inc.

Jesse Suen
Co-Founder & CTO
Akuity, Inc.

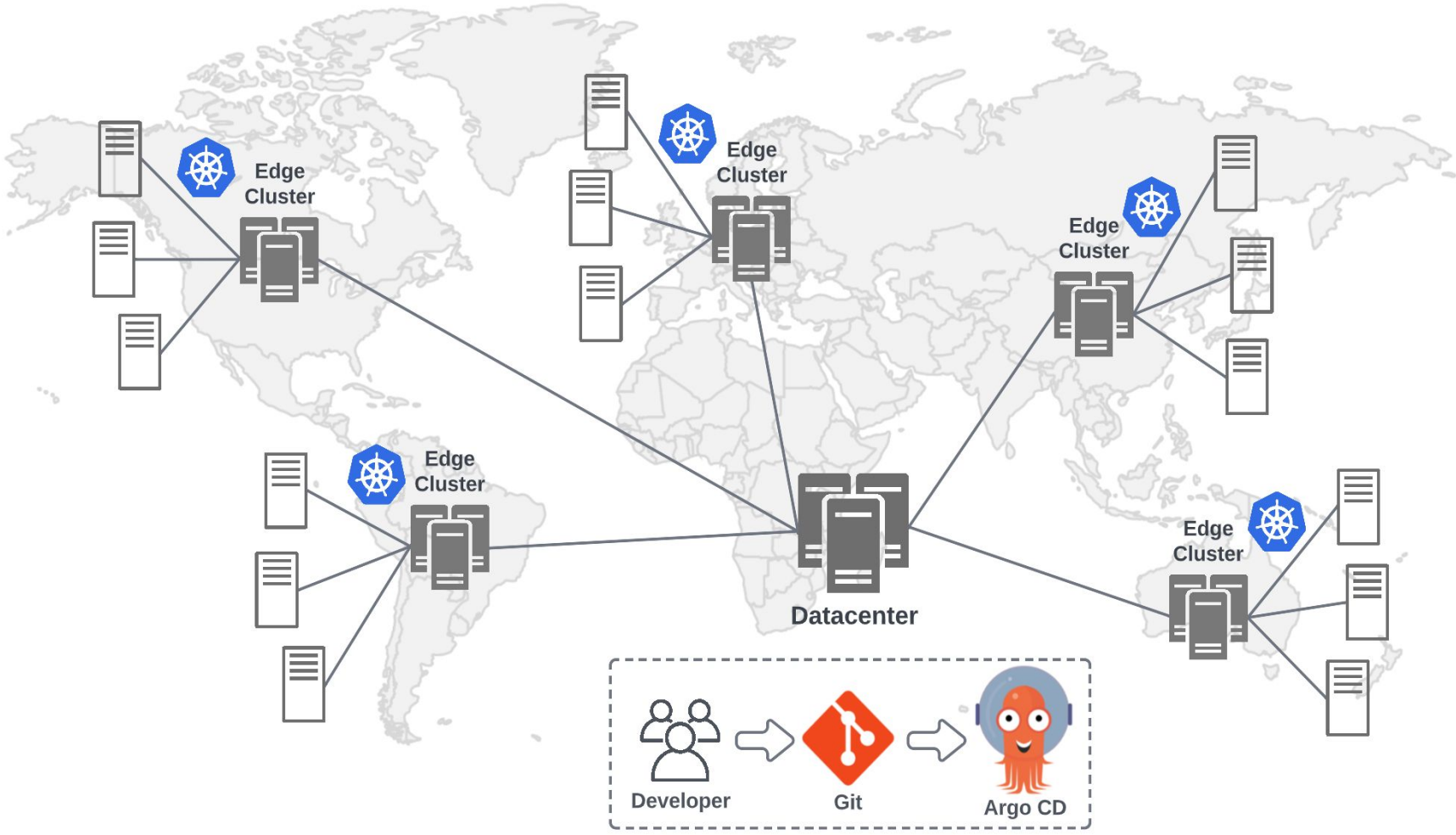
TikTok's Edge Cluster Overview

- Distributed around the world
  - ~100 edge clusters
  - 10-60 nodes on each
- TikTok edge services
  - video CDN cache
  - live streaming
  - game
  - etc.
- Datacenter
  - Management control plane
    - e.g. edge service deploy
  - Service data plane
    - e.g. CDN origin



TikTok's Edge Cluster Deployment

- Each edge is a K8s cluster
- GitOps from datacenter
  1. Developer push K8s configuration to git
  2. Central Argo CD controller pulls config from git and sync to edge K8s clusters



Edge Deployment Pattern

- An edge service is deployed to many edge clusters
- The service functionality and behavior are similar on all edge clusters
- As a result, the K8s configuration pattern of an edge service:
  - Deployment on all edge clusters shares large common configurations
  - There are some cluster-specific configurations
    - replica count
    - resource quota
    - IP address / ingress host
    - etc.

Edge Deployment Pattern - Example

Edge Cluster 1

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:1.19.0
        name: nginx
        resources:
          requests:
            cpu: "100m"
          limits:
            memory: "256Mi"
            cpu: "150m"

---

apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  externalIPs:
    - 1.1.1.1
  ports:
    - name: http
      protocol: TCP
      port: 80
```

Edge Cluster 2

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 6
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:1.19.0
        name: nginx
        resources:
          requests:
            cpu: "100m"
          limits:
            memory: "256Mi"
            cpu: "150m"

---

apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  externalIPs:
    - 2.2.2.2
  ports:
    - name: http
      protocol: TCP
      port: 80
```

Edge Cluster 3

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:1.20.0
        name: nginx
        resources:
          requests:
            cpu: "100m"
          limits:
            memory: "256Mi"
            cpu: "150m"

---

apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  externalIPs:
    - 3.3.3.3
  ports:
    - name: http
      protocol: TCP
      port: 80
```

Edge Deployment Management - Kustomize

- Base Layer Configuration

```
deploy/
 |— base/
 |    |— kustomization.yaml
 |    |— deployment.yaml
 |    |— service.yaml
 |— overlays/
      |— cluster1/
      |    |— kustomization.yaml
      |    |— deployment.yaml
      |    |— service.yam
      |— cluster2/
      |    |— kustomization.yaml
      |    |— deployment.yaml
      |    |— service.yamll
      |— cluster3/
           |— kustomization.yaml
           |— deployment.yaml
           |— service.yaml
```

```yaml
resources:
- deployment.yaml
- service.yaml
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:1.19.0
        name: nginx
        resources:
          requests:
            cpu: "100m"
          limits:
            memory: "256Mi"
            cpu: "150m"
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  ports:
    - name: http
      protocol: TCP
      port: 80
```

Edge Deployment Management - Kustomize

- Overlay Configuration

```
deploy/
 |— base/
 |    |— kustomization.yaml
 |    |— deployment.yaml
 |    |— service.yaml
 |— overlays/
      |— cluster1/
      |    |— kustomization.yaml
      |    |— deployment.yaml
      |    |— service.yam
      |— cluster2/
      |    |— kustomization.yaml
      |    |— deployment.yaml
      |    |— service.yamll
      |— cluster3/
           |— kustomization.yaml
           |— deployment.yaml
           |— service.yaml
```

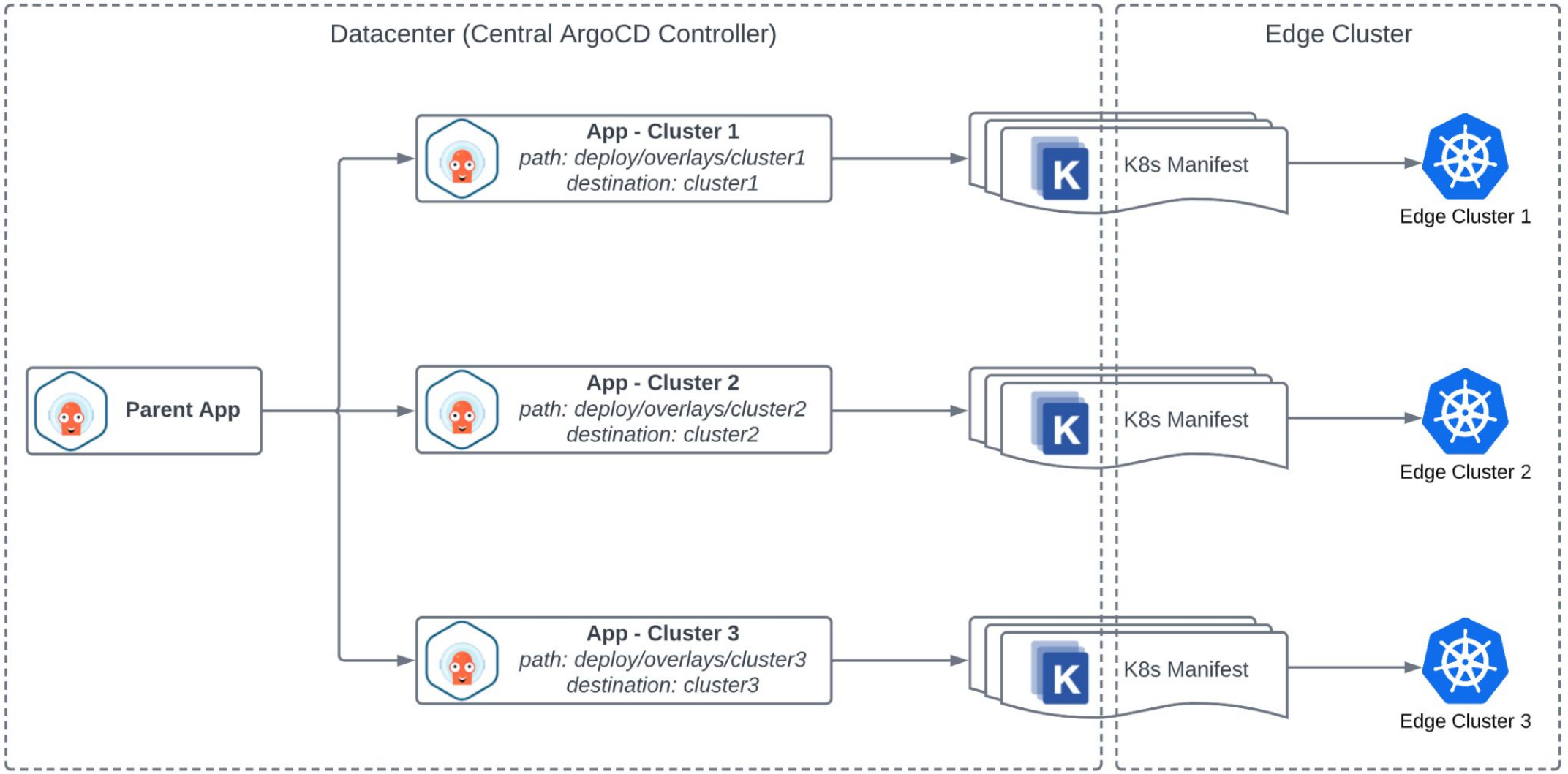
```yaml
resources:
- ../../base
patchesStrategicMerge:
- deployment.yaml
- service.yaml
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  template:
    spec:
      containers:
      - image: nginx:1.20.0
        name: nginx
```

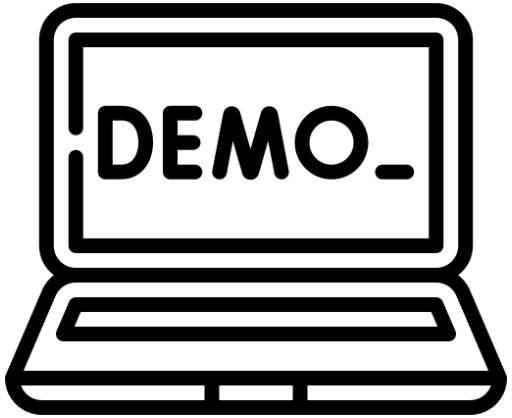
```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  externalIPs:
    - 3.3.3.3
```

Edge Deployment Management - Argo CD

- App of apps



Edge Deployment Management - Argo CD



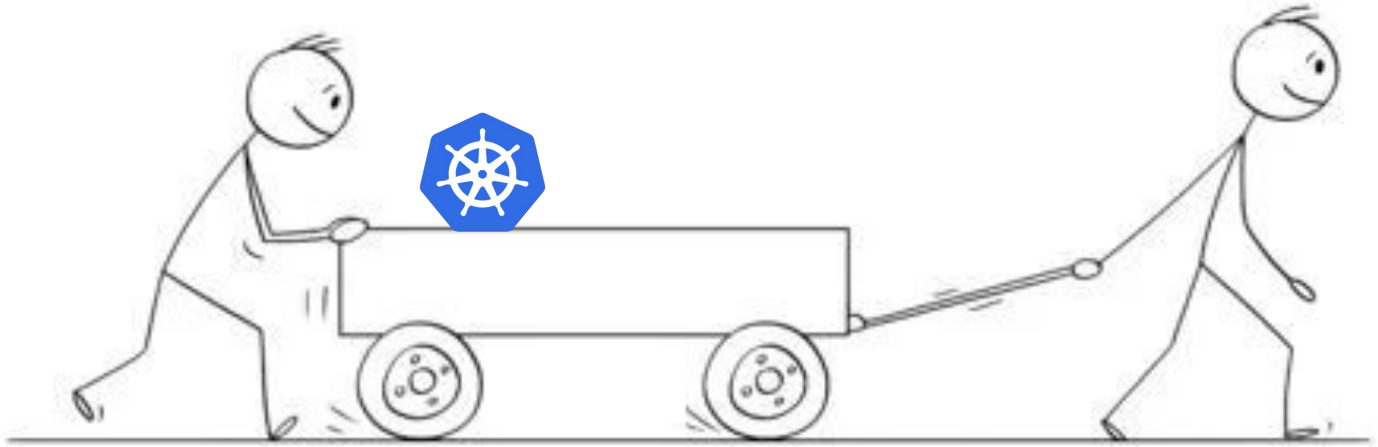
repo: https://github.com/qingkunl/argocd-app-of-apps-demo

Challenges - Argo CD

- Performance
  - Slow to list applications (>10s) when application number gets large (>3000)
- Scalability
  - Imbalanced application handling for different edge clusters
    - Argo CD shards applications to the controller instance by the destination cluster
    - The resource number and type for different edge clusters may vary (big vs small cluster)
- Functionality
  - Need better project-level support (as we use Argo CD project to separate teams/tenants)
    - Application name in different projects could conflict
    - Better project-level view and management, e.g. application, K8s resource, repo, etc.
- Developer Observability
  - Lack of internal observability for tracing, e.g. opentracing
    - Hard to trace and track a single request in source code when troubleshooting

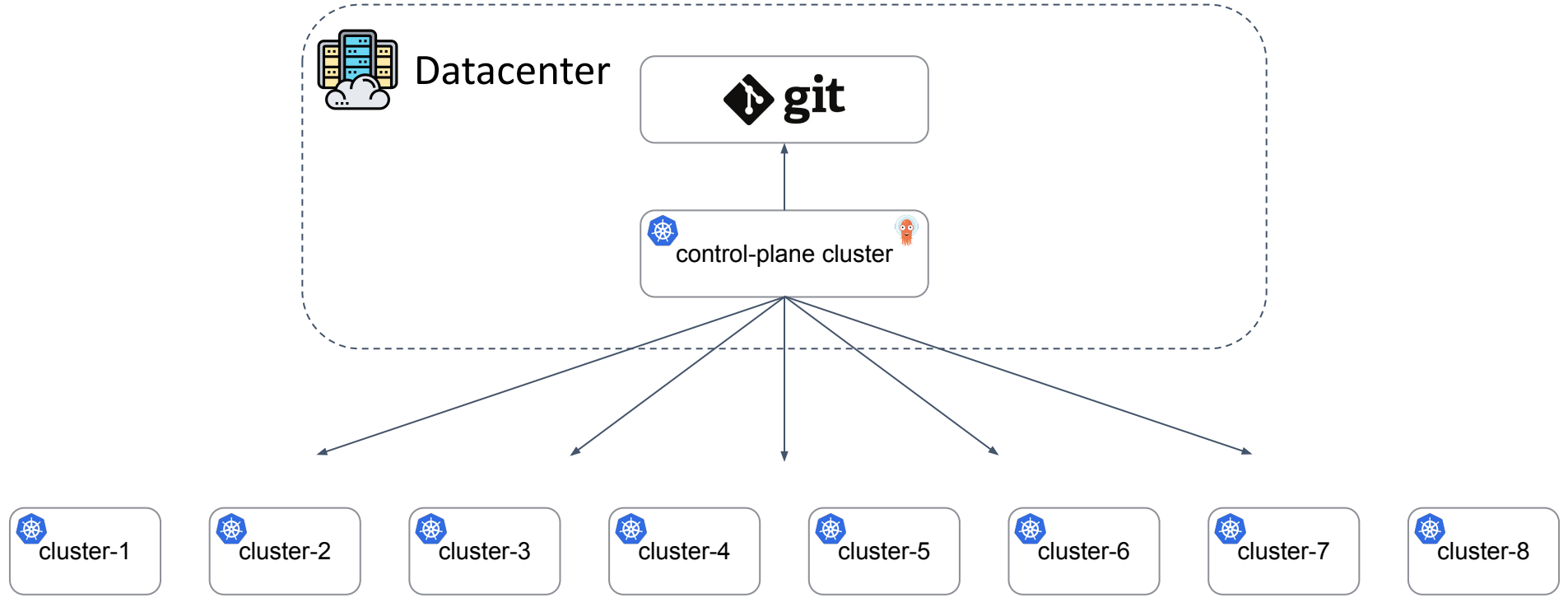
Edge Deployment Models

Centralized Push vs. Distributed Pull



Edge Deployment Model: Centralized Argo CD

Centralized Argo CD ***pushes*** manifests to individual clusters



Edge Deployment Model: Centralized Argo CD

Advantages

- Single pane of glass - view and control applications across all clusters
- Easiest to manage - single instance to maintain
- API & CLI integrations - automate against Argo CD instance running in datacenter

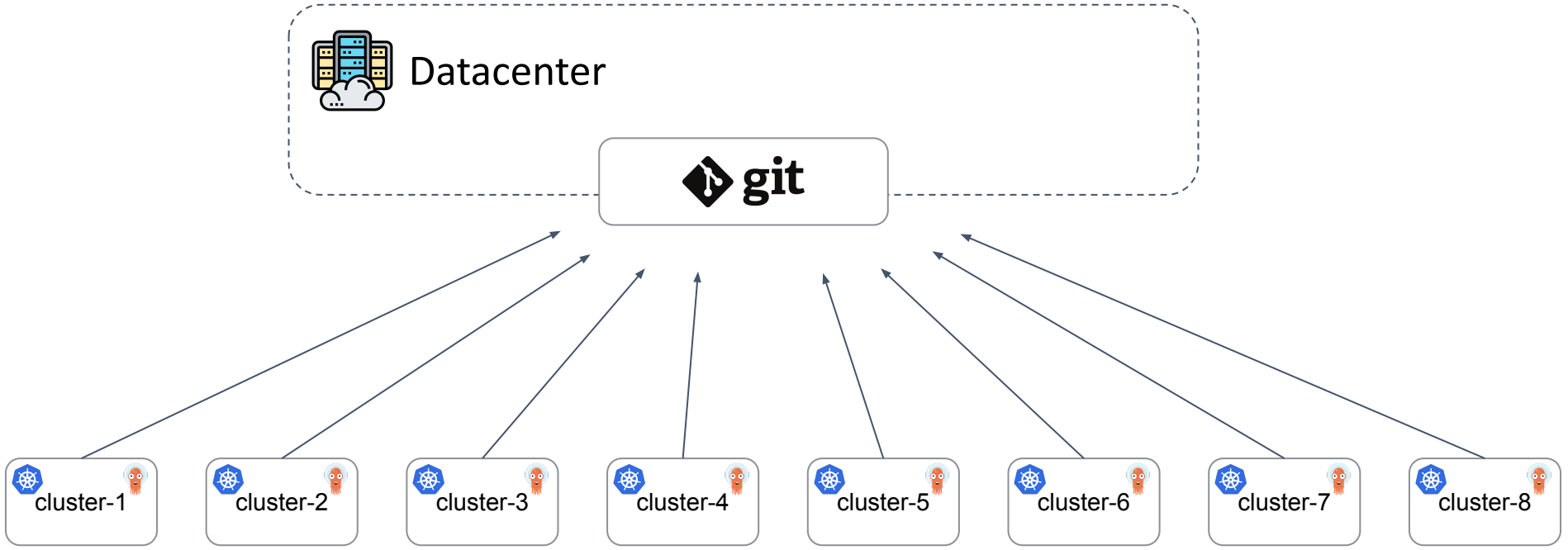
Disadvantages

- Less scalable - requires performance tuning and sharding as cluster fleet grows
- Edge clusters must allow external API Server access
- Single point of failure

Edge Deployment Model: Distributed Argo CD

Argo CD running in edge clusters, ***pulls*** manifests from central git repository

Applications auto-sync changes from cluster specific directories



Edge Deployment Model: Distributed Argo CD

Advantages

- Most scalable - application controller work is distributed to each edge cluster
- Increased security - K8s API servers can be made private

Disadvantages

- No control plane - lack of centralized visibility and management capabilities
- Difficult to manage - many Argo CDs to access/configure/upgrade
- Less flexible - auto-sync requirement reduces flexibility in deploy options
- Git access - clusters requires network access and authentication to the git repository

Edge Techniques: Argo CD Core

A minimal installation mode of Argo CD

- Installs just the minimum necessary components:
  - controller, repo-server, redis
- Great fit for a distributed Argo CD model when UI, API, multi-tenancy are not needed
- UI and CLI are still available with kubeconfig access:

```
export KUBECONFIG=~/.kube/edge-cluster-1
argocd login --core
argocd app list
argocd admin dashboard  # available at http://localhost:8080
```

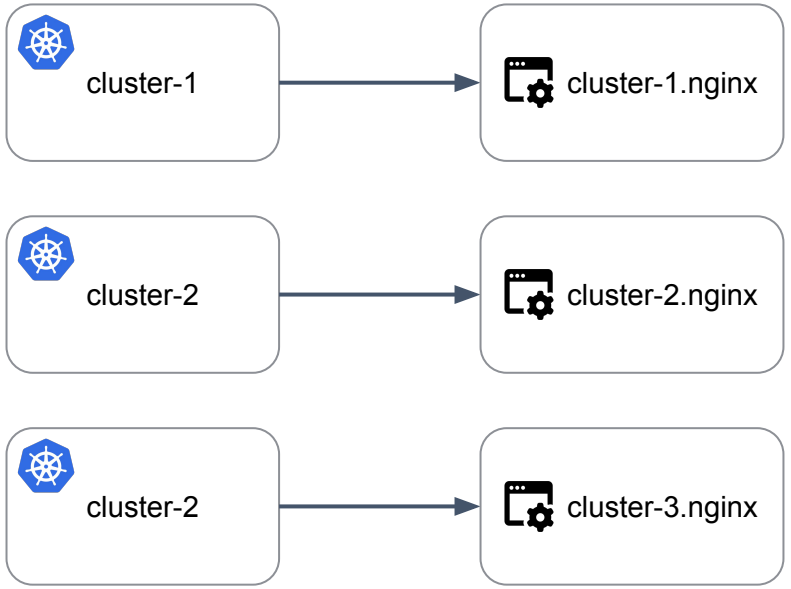
Edge Techniques: ApplicationSets

- Automatically create and deploy applications based on discovery:
  - Paths/files in a git repository
  - Clusters registered to Argo CD
  - Combination of the two
- Created as an alternative to the App-of-Apps
- Works with either the centralized or distributed deployment models

ApplicationSets - Centralized Argo CD

In the central Argo CD, for each cluster registered to Argo CD, create Application(s) for it

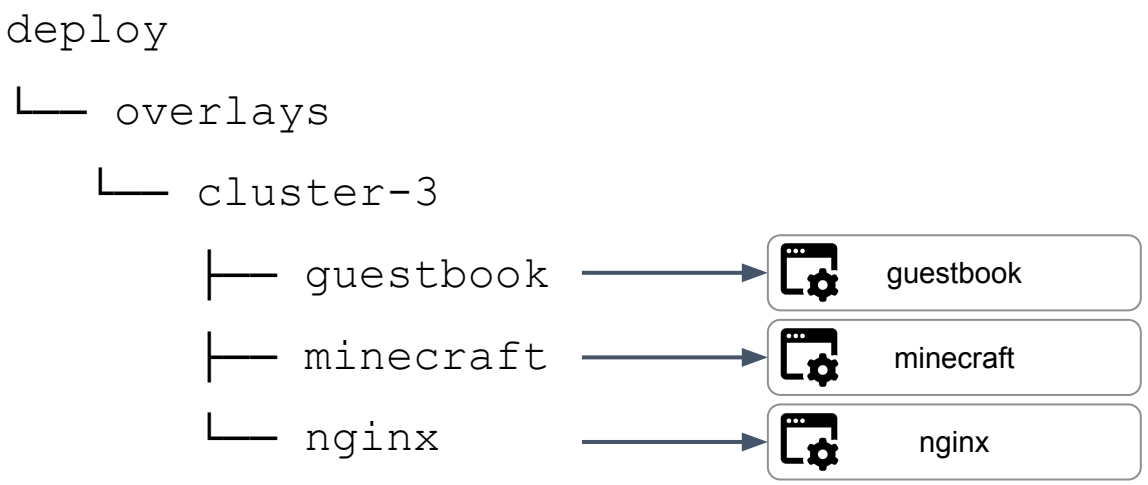
```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
 name: nginx
spec:
 generators:
 - clusters: {}
 template:
   metadata:
     name: '{{name}}.nginx' # name of cluster
   spec:
     project: "default"
     source:
       repoURL: https://github.com/example/repo
       targetRevision: HEAD
       path: deploy/overlays/{{name}}
     destination:
       server: '{{server}}' # K8s server URL
       namespace: nginx
```



ApplicationSets - Distributed Argo CD

In an edge Argo CD, for each sub directory in a cluster specific path, create Application(s) for it

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
 name: cluster-addons
spec:
 generators:
 - git:
     repoURL: https://github.com/example/repo
     revision: HEAD
     directories:
     - path: deploy/overlays/cluster-3/*
 template:
   metadata:
     name: '{{path.basename}}'
   spec:
     project: default
     source:
       repoURL: https://github.com/example/repo
       targetRevision: HEAD
       path: '{{path}}'
     destination:
       server: https://kubernetes.default.svc
```



Recent & Future Improvements

Recent Improvements

- Performance and caching improvements
- Tuning options for unreliable networks

Future Improvements

- Applications in different namespaces
  - Prevent application name collisions
  - Better UI performance
- Utility for automatic balancing of cluster shard
- Open Telemetry
- Web shell (kubectl exec in Argo CD UI)

Thank You!