KubeCon | CloudNativeCon
North America 2023

GoTo Financial's Story:
Towards 10k+ ArgoCD Apps
to Support Billions of $ Transactions

Giri Kuncoro & Yudi Phanama
goto financial

intro

KubeCon | CloudNativeCon
North America 2023

@girikuncoro

@yphanama

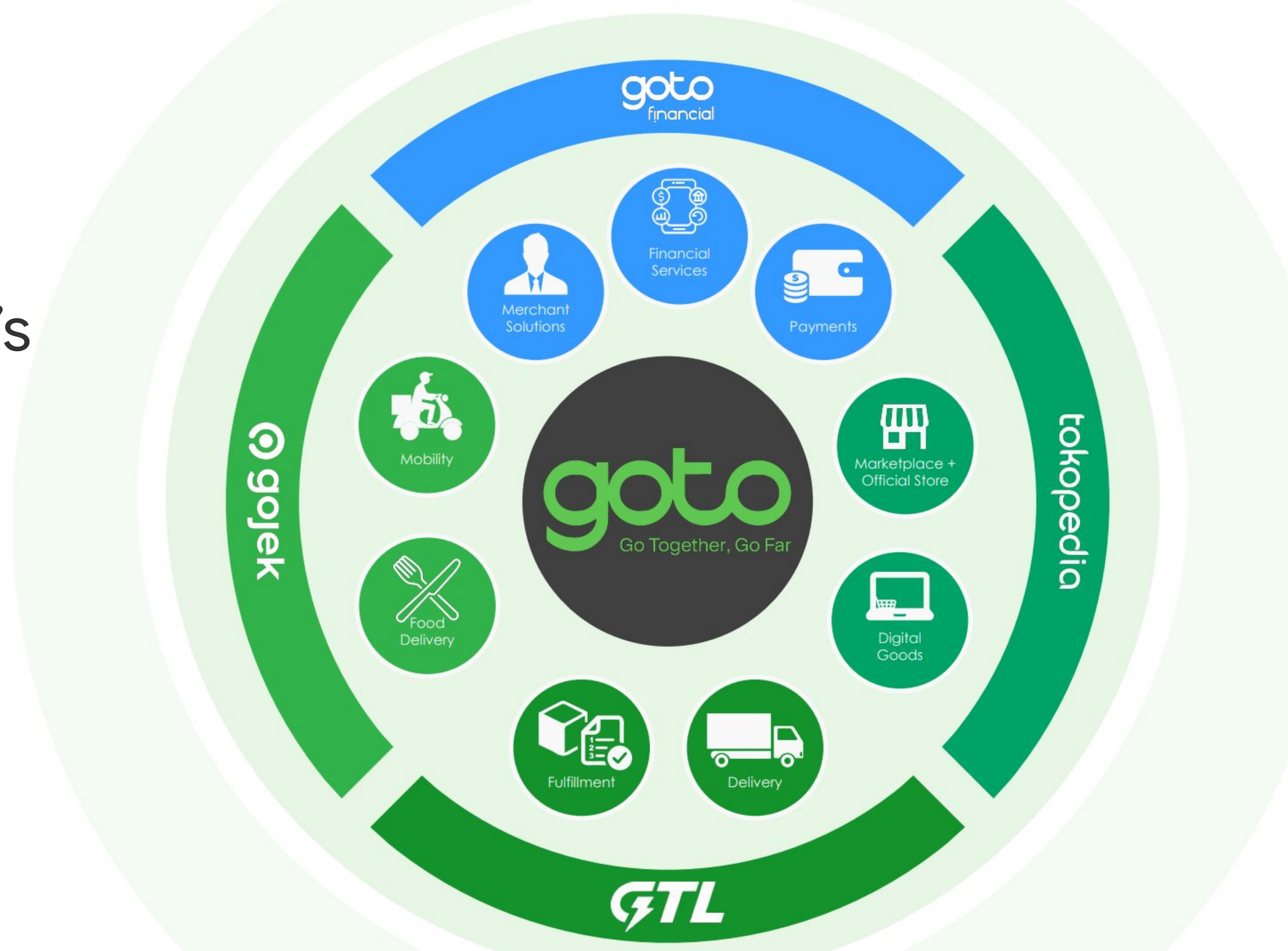
intro

goto financial

Part of goto group,

the leading Indonesia’s

digital ecosystem



ArgoCD Overview

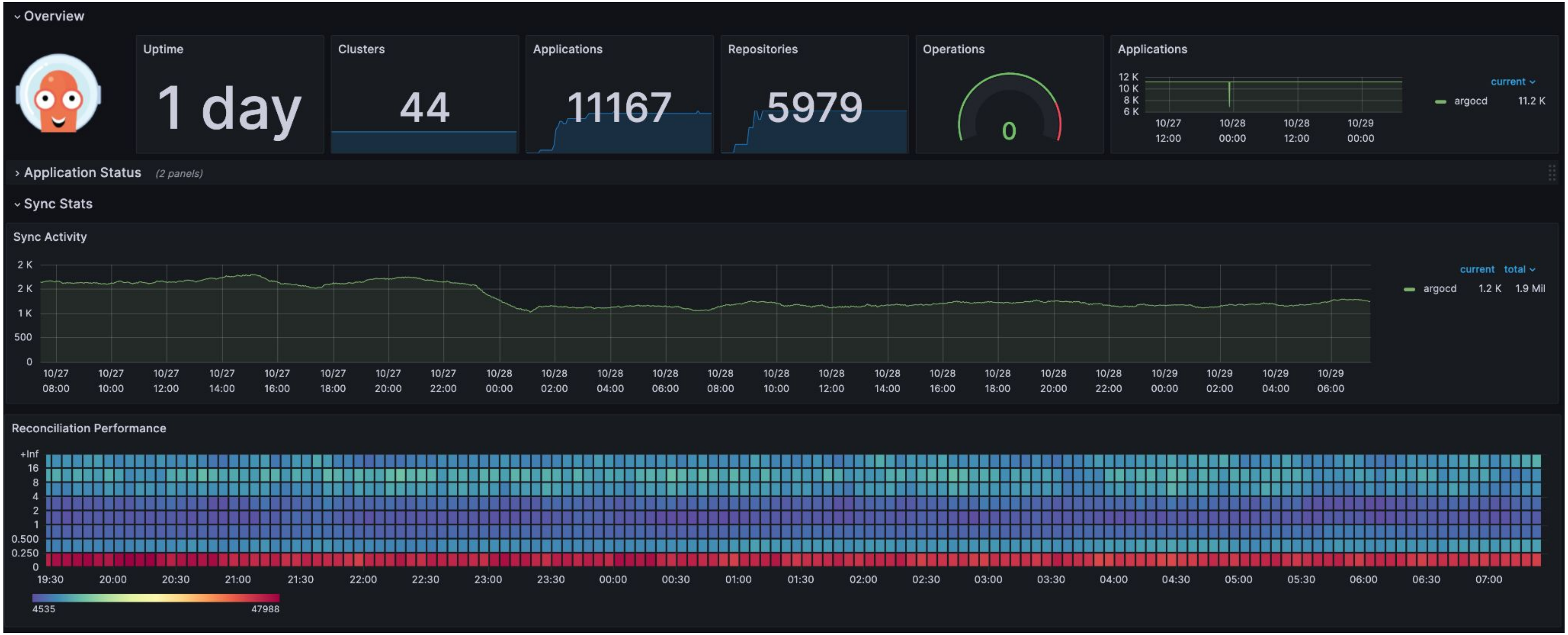
@ goto financial

kubectl describe cluster

- Distributed across Singapore and Indonesia region
  - **~50 Kubernetes clusters** in AWS, GCP, and Private Datacenter
  - **700+** compute nodes
  - **15,000+ CPU**
  - **120+ TB** memory
  - **30,000+** pods

argocd snapshot

KubeCon | CloudNativeCon
North America 2023



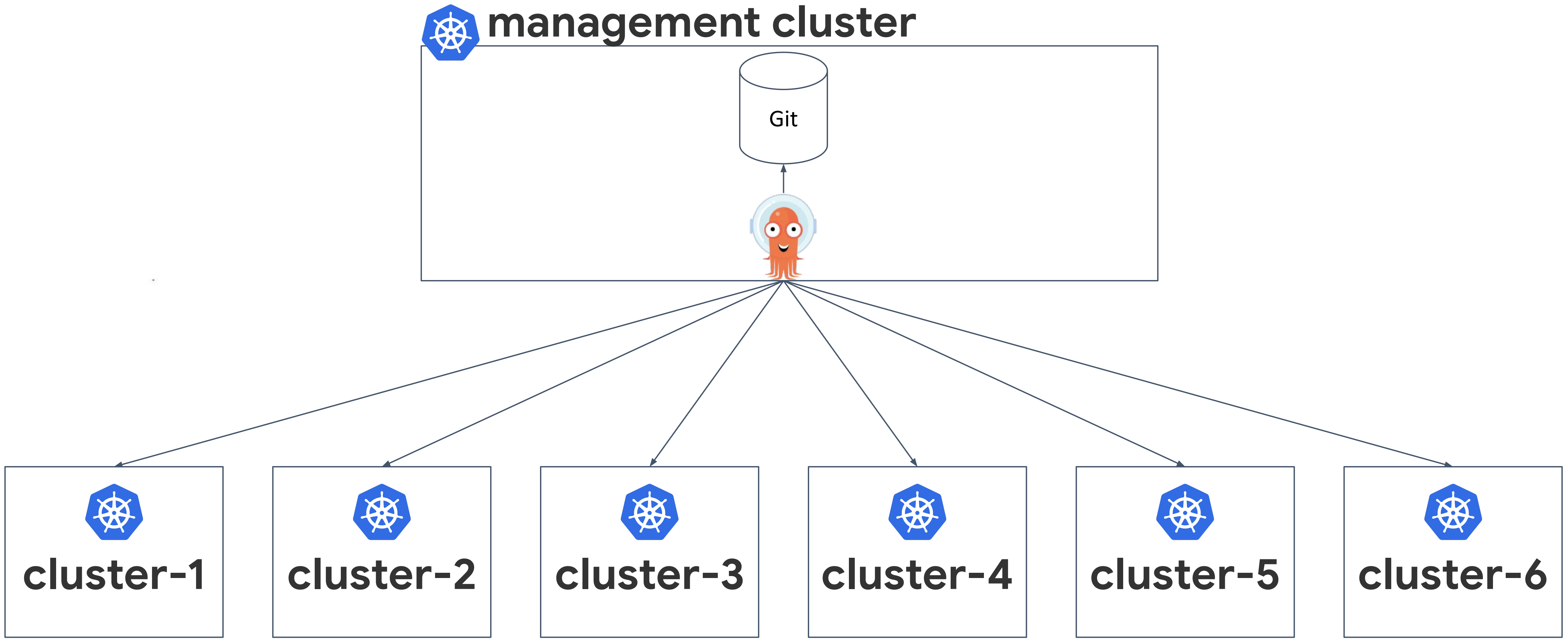
kubectl describe argocd

- **11,000+** applications
- **6,000+** repositories
- **~60** projects
- **380,000+** total objects

on largest cluster

- **2,000** applications
- **40,000+** objects

centralized argocd (push model)

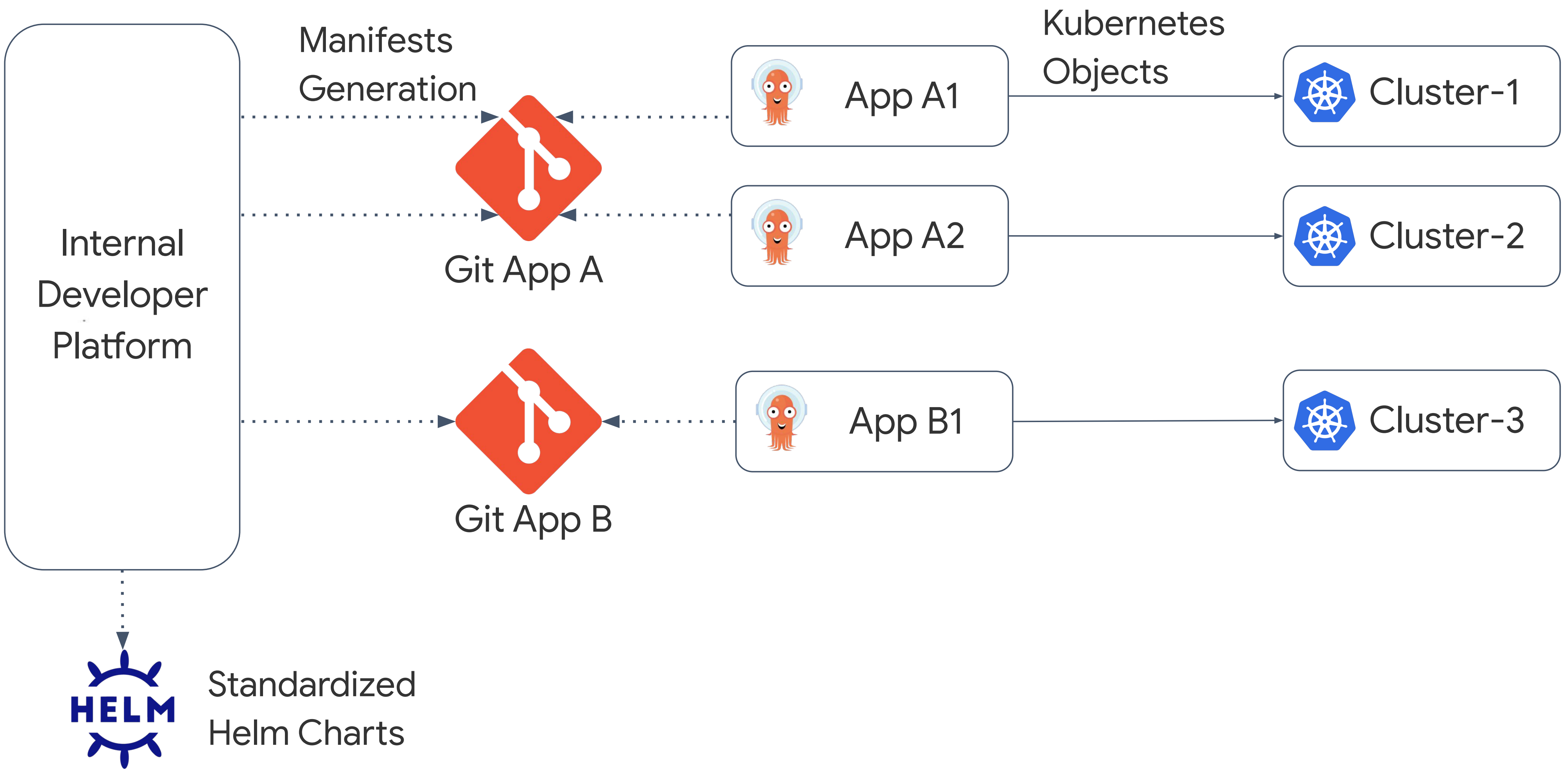


centralized argocd

Pros

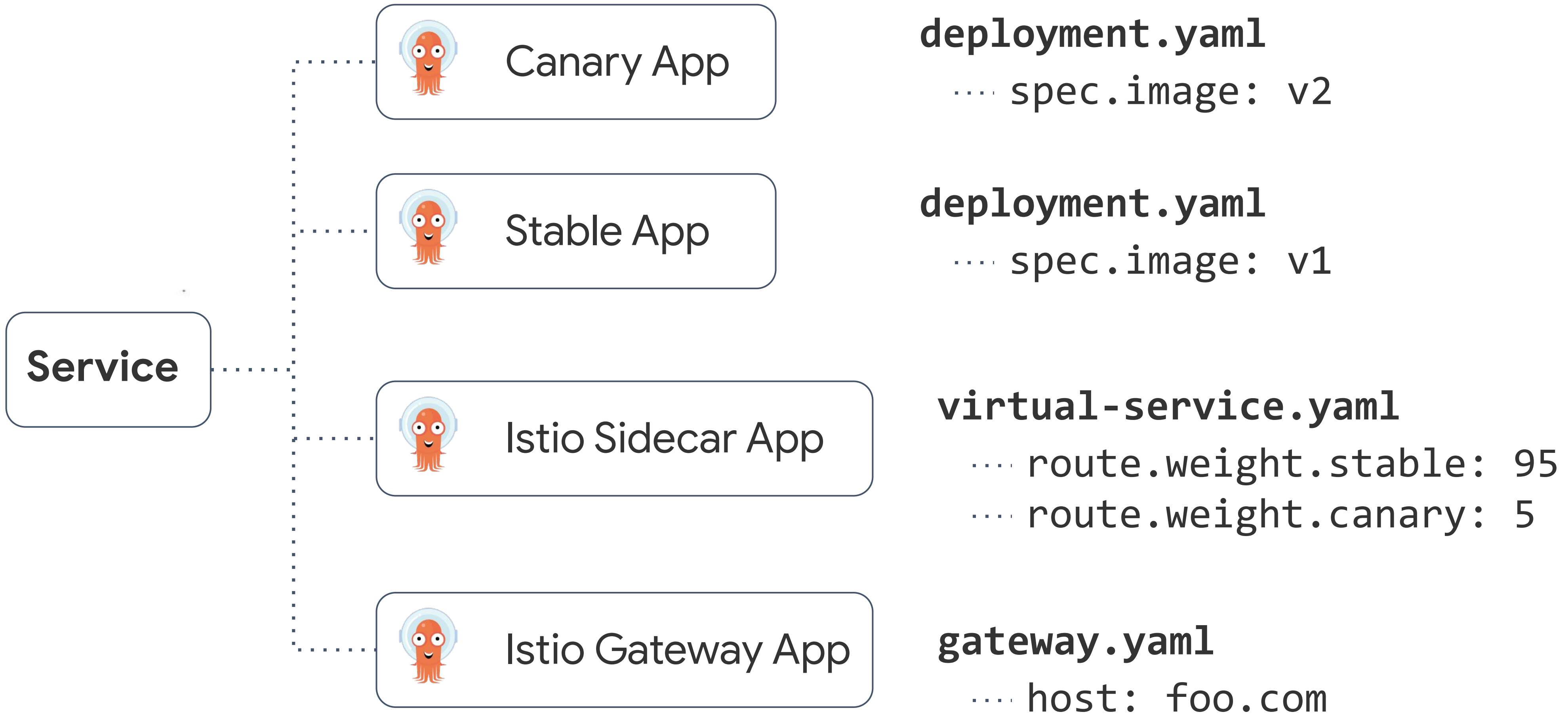
- Easy to maintain and upgrade
- Easy to integrate with our automation and platform
- Easy to manage centralized RBAC
- Single dashboard to view and control all clusters

platform integration



service lifecycle

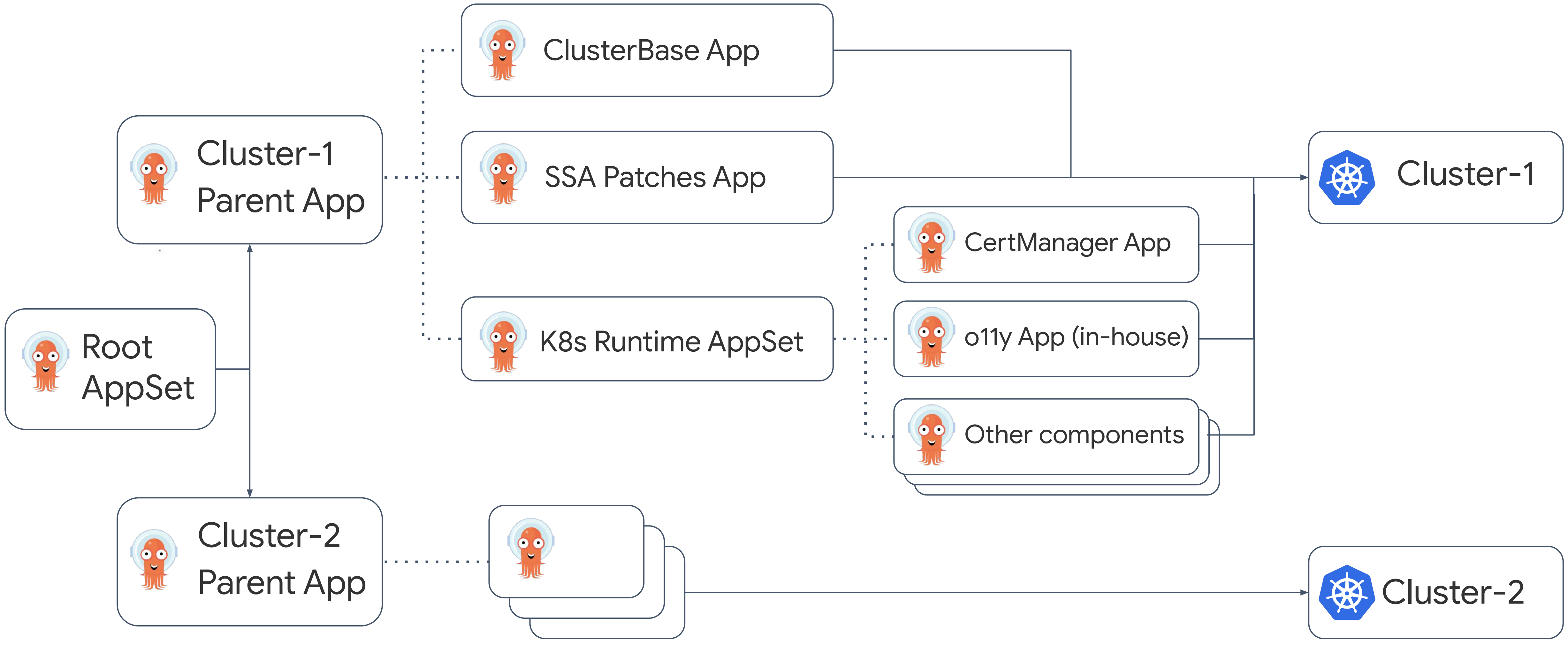
1 service contains 3-5 ArgoCD apps with different lifecycle



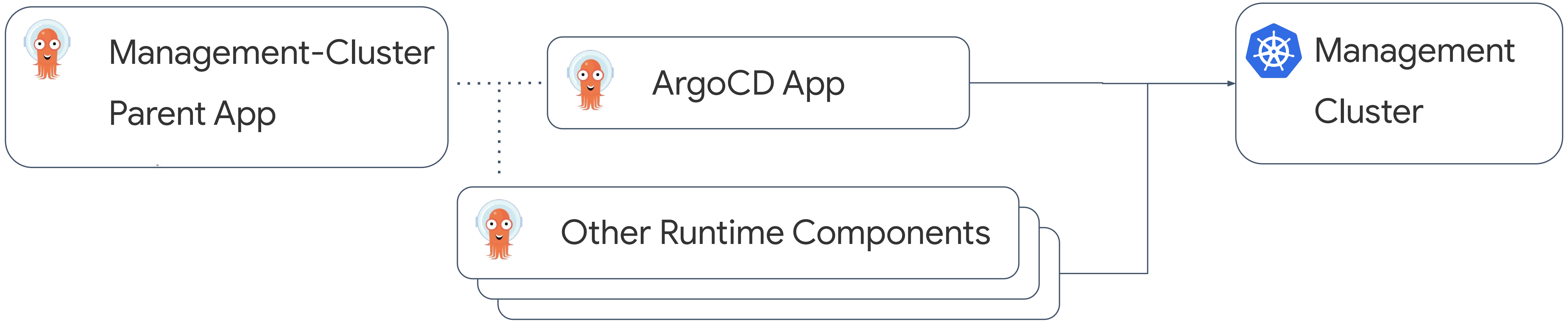
```
deployment.yaml
  spec.image: v2

deployment.yaml
  spec.image: v1

virtual-service.yaml
  route.weight.stable: 95
  route.weight.canary: 5

gateway.yaml
  host: foo.com
```

cluster runtime components

appset & app of apps pattern - monorepo



argocd on argocd



challenges of centralized argocd

Connectivity to all target clusters

- Tunnels / peering
- Public mTLS

Functionalities

- Must maintain unique application name globally (63 chars constraint)
- Single point of failure

challenges of centralized argocd

Performance issues

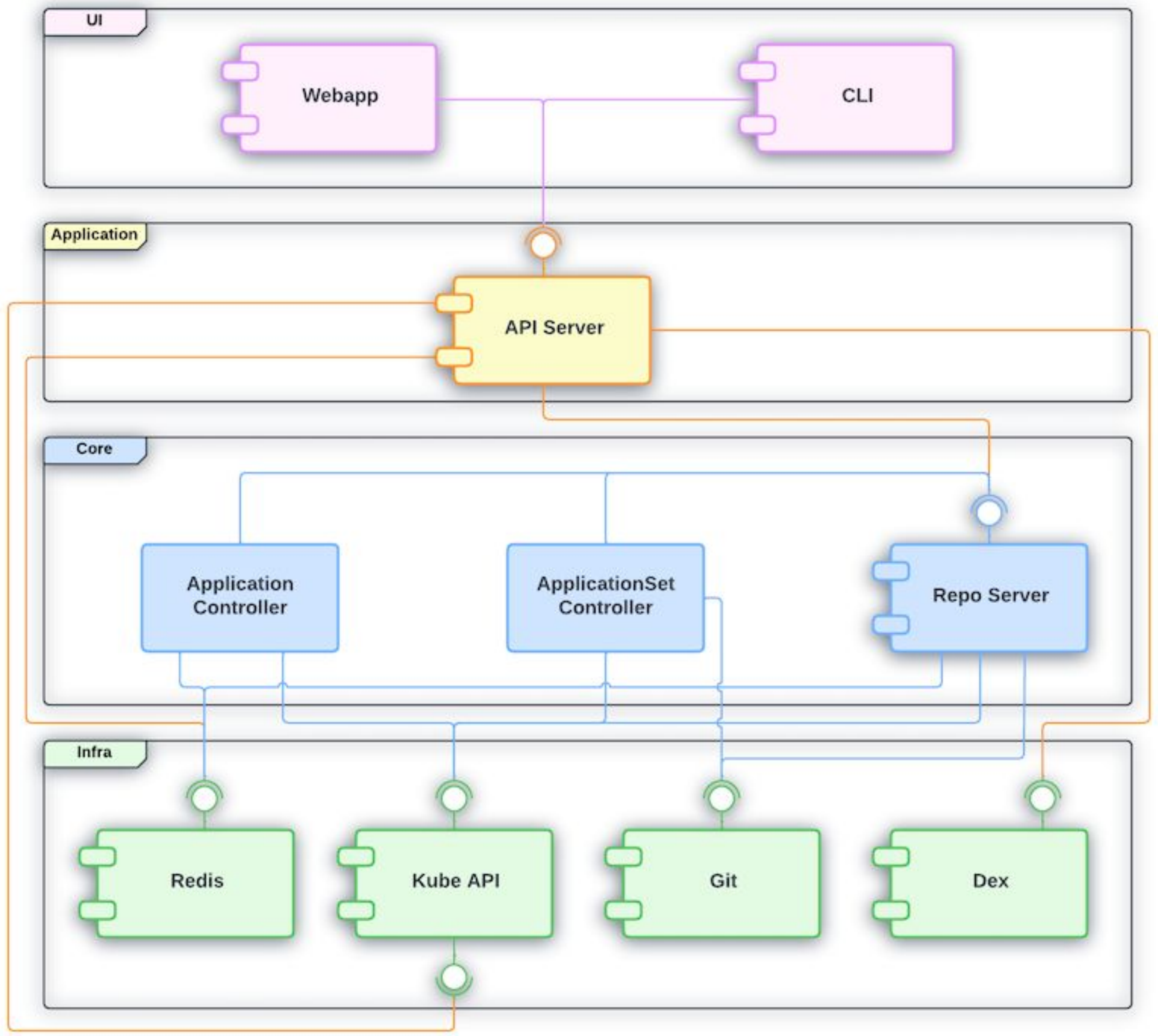
- Slow reconciliation & sync
  - `workqueue_depth`
  - `argocd_app_reconcile_bucket`
- Slow UI loading (> 1 min)
  - very obvious + browser inspect
- Repo server OOM kills
  - `kube events`

challenges of centralized argocd

- High rate of Git API calls (both `ls-remote` and `fetch`)
  - `argocd_git_request_total`
- High repo cache miss
  - `repo-server logs`
- Imbalanced shards & noisy cluster
  - `controller process_cpu_seconds_total`

Performance Tuning Journey 🚀

argocd components



Source:
https://argo-cd.readthedocs.io/en/stable/developer-guide/architecture/components/

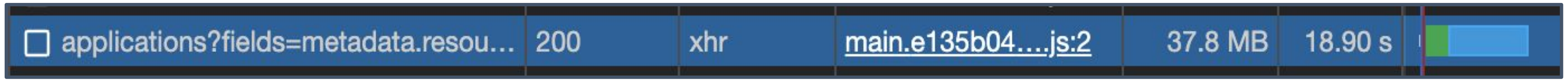
tuning: argocd-server

Problem: Slow UI load

🔧 enable gzip compression

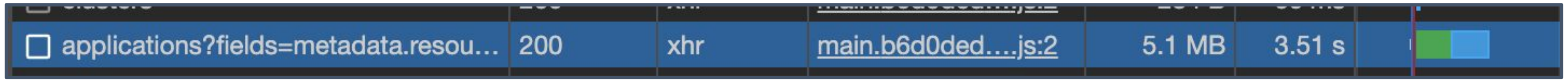
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "argocd-server"
spec:
      containers:
      - name: argocd-server
        env:
        - name: "ARGOCD_SERVER_ENABLE_GZIP"
          value: "true"
```

before:



```
applications?fields=metadata.resou...   200   xhr   main.e135b04....js:2   37.8 MB   18.90 s
```

after (~5x faster load + ~7x smaller data)



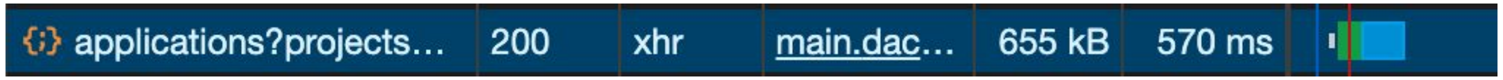
```
applications?fields=metadata.resou...   200   xhr   main.b6d0ded....js:2   5.1 MB   3.51 s
```

tuning: argocd-server

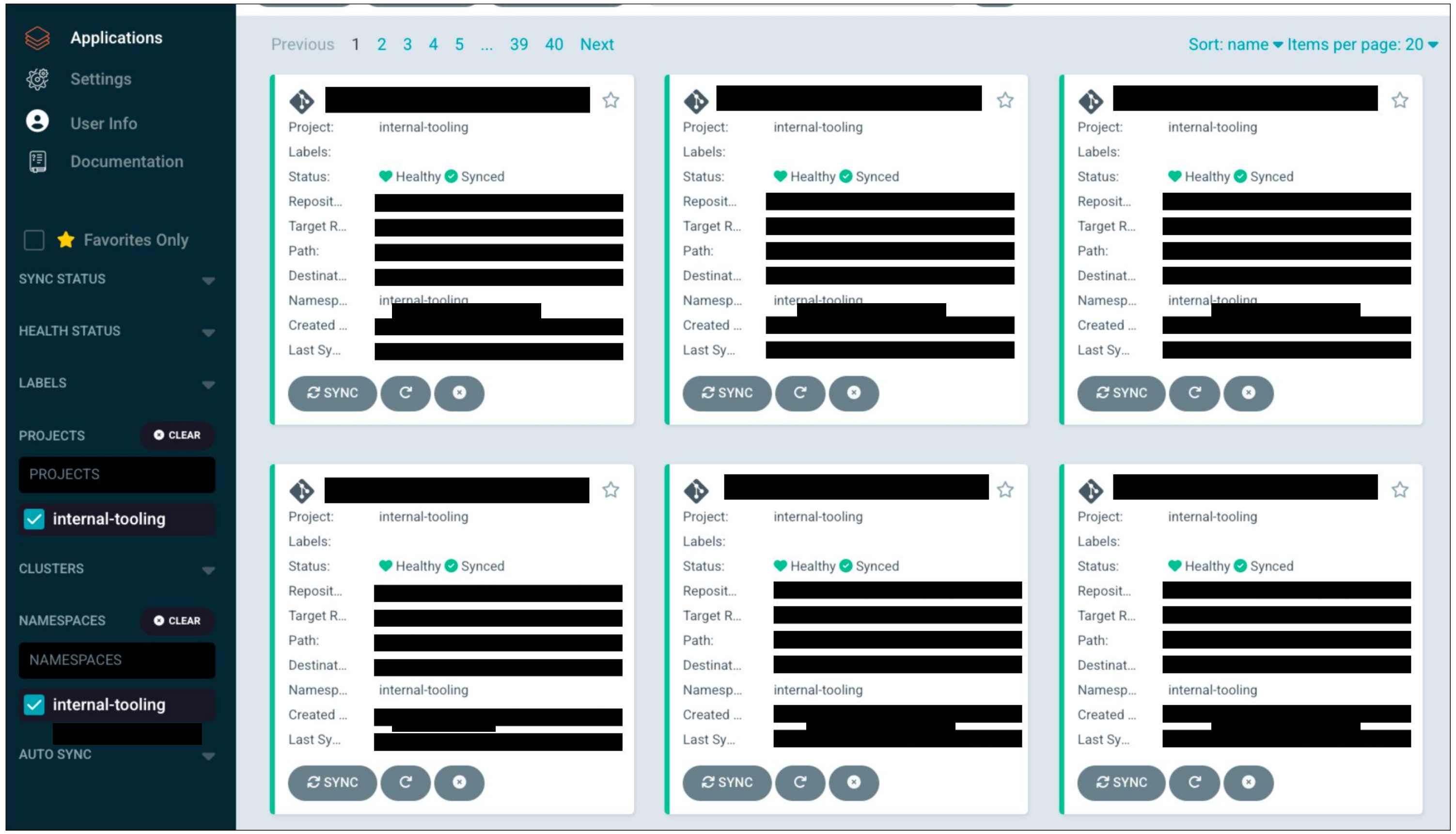
KubeCon CloudNativeCon North America 2023

argocd ui tip - use selectors (labels, projects, namespaces)

🍪 The last selectors are saved the next time we load the ArgoCD UI



```
applications?projects…   200   xhr   main.dac…   655 kB   570 ms
```



Projects: ✓ internal-tooling

Namespaces: ✓ internal-tooling [redacted]

Previous 1 2 3 4 5 ... 39 40 Next

Sort: name Items per page: 20

[redacted]
Project: internal-tooling
Labels:
Status: Healthy Synced
Reposit... Target R... Path: Destinat... [redacted]
Namesp... internal-tooling [redacted]
Created ... Last Sy... [redacted]
SYNC

[redacted]
Project: internal-tooling
Labels:
Status: Healthy Synced
Reposit... Target R... Path: Destinat... [redacted]
Namesp... internal-tooling [redacted]
Created ... Last Sy... [redacted]
SYNC

[redacted]
Project: internal-tooling
Labels:
Status: Healthy Synced
Reposit... Target R... Path: Destinat... [redacted]
Namesp... internal-tooling [redacted]
Created ... Last Sy... [redacted]
SYNC

[redacted]
Project: internal-tooling
Labels:
Status: Healthy Synced
Reposit... Target R... Path: Destinat... [redacted]
Namesp... internal-tooling
Created ... Last Sy... [redacted]
SYNC

[redacted]
Project: internal-tooling
Labels:
Status: Healthy Synced
Reposit... Target R... Path: Destinat... [redacted]
Namesp... internal-tooling
Created ... Last Sy... [redacted]
SYNC

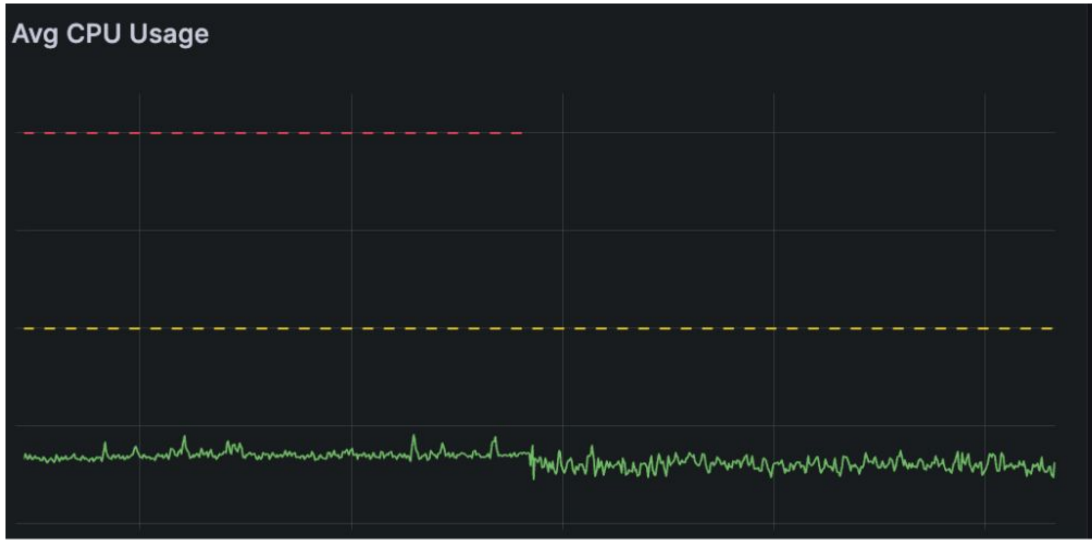
[redacted]
Project: internal-tooling
Labels:
Status: Healthy Synced
Reposit... Target R... Path: Destinat... [redacted]
Namesp... internal-tooling
Created ... Last Sy... [redacted]
SYNC

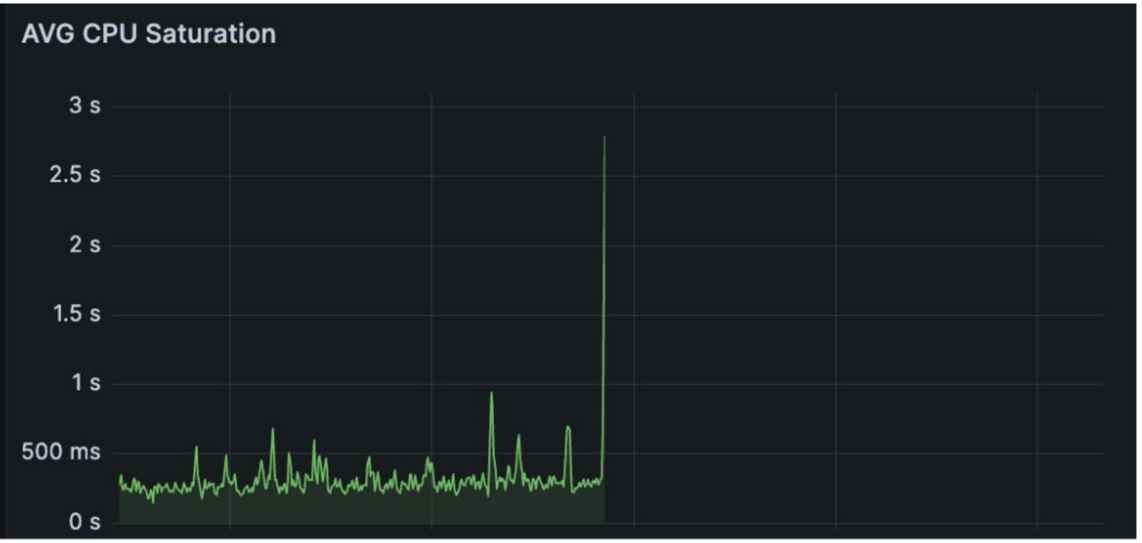
tuning: k8s cpu limits

Problem: CPU gets throttled across all components

🔧 remove k8s CPU limit, use k8s requests only

- k8s request & limit uses cgroup, which uses CFS [1]
- CFS guarantees or throttle CPU proportional to container shares in a Node
- ref [2][3]





tuning: repo-server

Problem: repo-server got OOMkilled frequently

🔧 Increase replicas and use HPA

```yaml
apiVersion: apps/v1
kind: HorizontalPodAutoscaler
metadata:
  name: "argocd-repo-server"
spec:
...
  maxReplicas: 15
  minReplicas: 5
  metrics:
  - resource:
      name: memory
      target:
        averageUtilization: 75
      type: Resource
  - resource:
      name: cpu
      target:
        averageUtilization: 75
      type: Resource
...
```

- Automatically scales with memory usage
- Distributes requests to more pods
- Alternatively, you could use the `--parallelismlimit` flag to control how many manifest generation requests that can be served in parallel and help avoid OOM kills.

tuning: repo-server

Problem: repo-server timeout errors from our logs

🔧 increase repo server client timeout

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "argocd-server"
spec:
…
      containers:
      - name: argocd-server
        env:
        - name: "ARGOCD_SERVER_REPO_SERVER_TIMEOUT_SECONDS"
          value: "120" #default is 60

…
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: "argocd-application-controller"
spec:
…
      containers:
      - name: argocd-application-controller
        env:
        - name: "ARGOCD_APPLICATION_CONTROLLER_REPO_SERVER_TIMEOUT_SECONDS"
          value: "120" #default is 60

…
```

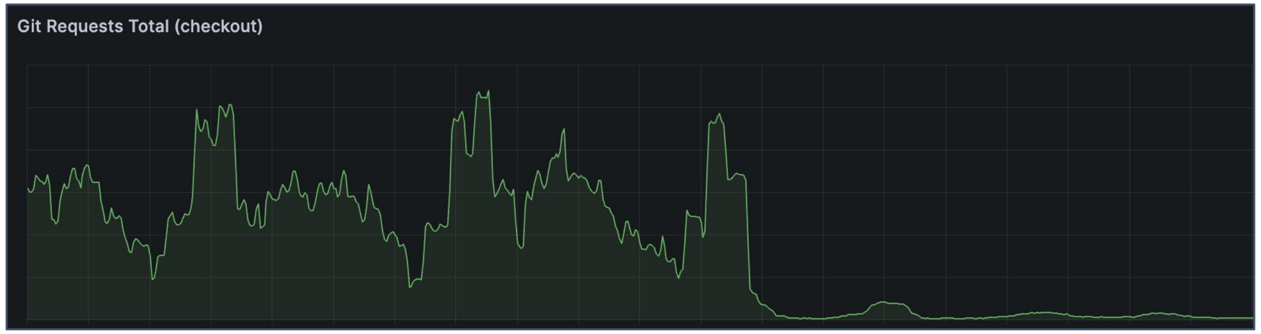
- argocd-server and app-controller talks to the repo-server for manifests generation.
- We started seeing these timeout errors from them when syncing or refreshing apps.
- Increase it in **both** the argocd-server and argocd-application-controller.

tuning: repo-server

Problem: persistently high git fetch requests

🔧 extend repo cache expiration

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "argocd-repo-server"
spec:
…
      containers:
      - name: argocd-repo-server
        env:
        - name: "ARGOCD_REPO_CACHE_EXPIRATION"
          value: "24h"

…
```



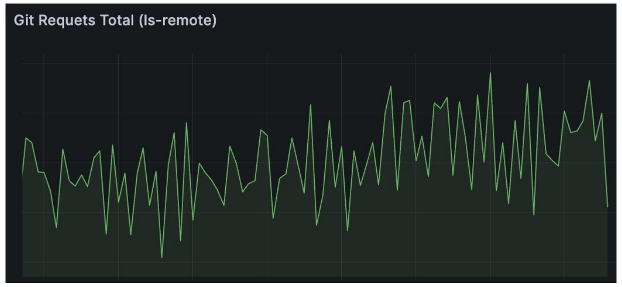
- ArgoCD caches generated manifests (24h).
- When **remote files change** often **even though** the repository **tag hasn't changed**, shorter expiry is desirable to pick up the updates.
- We can use higher expiry if the Helm/Kustomize/Git remote refs are already hermetic.
- Set ARGOCD_REPO_CACHE_EXPIRATION to extend repo cache expiration.

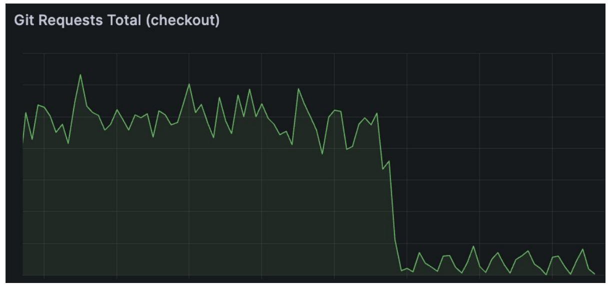
tuning: monorepo usage

KubeCon CloudNativeCon North America 2023

Problem: Multi Sources apps caused high git requests

```
sources:
  - directory:
      exclude: "*"
    path: "my-app/"
    repoURL: https://REPO_URL
    targetRevision: HEAD
    ref: values
  - chart: CHART_NAME
    repoURL: CHART_URL
    targetRevision: v0.0.1
    helm:
      version: v3
      releaseName: RELEASE_NAME
      valueFiles:
      - $values/my-app/values.yaml
```





🔧🔧🔧

- Very high *git-fetch* and *git-ls-remote* requests.
- Potentially a bug. We implemented an undocumented workaround #14725.
- Our git-fetch requests dropped dramatically.
- We're still seeing high ls-remote requests. The bug issue is still open.

tuning: monorepo usage w/webhook

Problem: all monorepo apps refreshed every time there's commit to unrelated apps

🔧 use manifest paths annotation

```
…
kind: Application
metadata:
  name: my-kustomize-app
  namespace: argocd
  annotations:
    argocd.argoproj.io/manifest-generate-paths: "./;/bases/my-base;"
spec:
…
```

- (in monorepo) ArgoCD webhook server refresh all apps when it receive a webhook.
- In the refresh process, ArgoCD invalidates cache for all apps and calls k8s API to annotate all Application objects, slowing update process when having 1k+ apps.
- Using the annotation filters out unrelated apps, speeding up the update process.

tuning: argocd-application-controller

Problem: workqueue depth started piling up

🔧 Increase # of operation processors and status processors

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app.kubernetes.io/name: argocd-cmd-params-cm
    app.kubernetes.io/part-of: argocd
  name: argocd-cmd-params-cm
data:
...
  # ARGOCD_APPLICATION_CONTROLLER_STATUS_PROCESSORS
  controller.status.processors: '500'
  # ARGOCD_APPLICATION_CONTROLLER_OPERATION_PROCESSORS
  controller.operation.processors: '250'
...
```

For every 1000 application, use

`--status-processors=50` and

`--operation-processors=25`

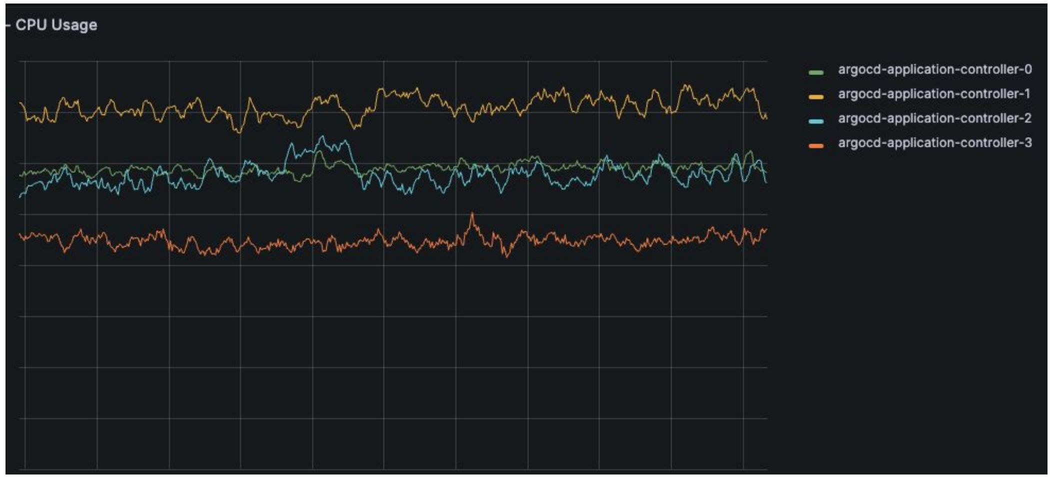
tuning: argocd-application-controller

Problem: scaling argocd-application-controller

🔧 shard controllers to multiple pods - distribute load

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: "argocd-application-controller"
spec:
  replicas: 4
  template:
    spec:
      containers:
      - args:
        - "/usr/local/bin/argocd-application-controller"
        env:
        - name: "ARGOCD_CONTROLLER_REPLICAS"
          value: "4"
...
```

- argocd-app-controller is horizontally shardable
- sharding algorithm is on the cluster-level
- increase replicas and set **ARGOCD_CONTROLLER_REPLICAS**

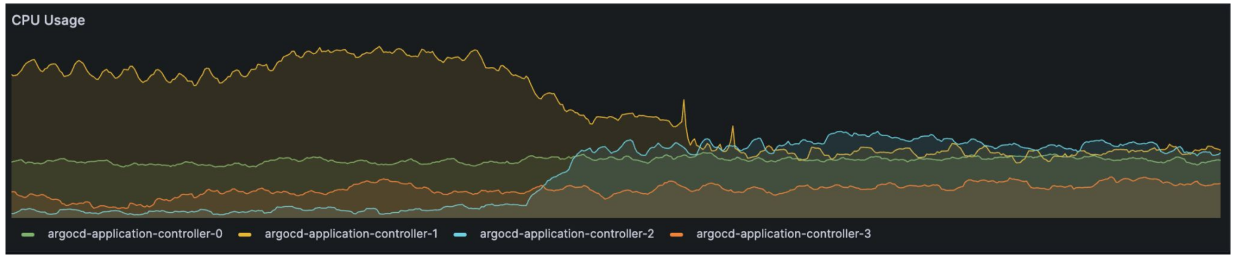


tuning: argocd-application-controller

Problem: uneven shards CPU usage, some shards are higher than others

🔧 manual shard assignment

```
...
apiVersion: v1
kind: Secret
metadata:
  annotations:
    argocd.argoproj.io/sync-options: ServerSideApply=true
  name: CLUSTER_A_SECRET_NAME
  namespace: argocd
type: Opaque
stringData:
  shard: "2" # the shard number

...
```



- ArgoCD shards per cluster, not per app. Large clusters could be hosted by the same shard.
- The round-robin sharding might not help much either. There's still chance large clusters could get into same shards.
- We did manual shard allocation instead to fine-tune shard resources.
- ArgoCD app-level sharding as a feature would be really great. See discussion [1].

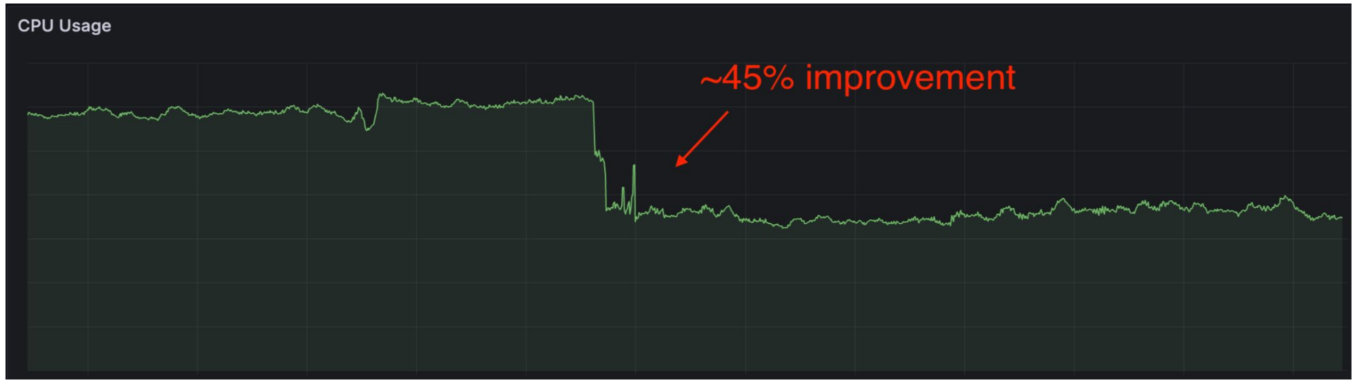
tuning: argocd-application-controller

Problem: we still see very high app-controller CPU usage, slow reconciles

🔧 optimize high-churn Reconciliations

```yaml
resource.customizations.ignoreDifferences.all: |
  managedFieldsManagers:
  - kube-controller-manager
  jsonPointers:
  - /spec/replicas

resource.ignoreResourceUpdatesEnabled: "true"

resource.customizations.ignoreResourceUpdates.all: |
  jsonPointers:
  - /status

resource.customizations.ignoreResourceUpdates.autoscaling_HorizontalPodAutoscaler: |
  jsonPointers: #for autoscaling/v1 compatibility
  - /metadata/annotations/autoscaling.alpha.kubernetes.io~1behavior
  - /metadata/annotations/autoscaling.alpha.kubernetes.io~1conditions
  - /metadata/annotations/autoscaling.alpha.kubernetes.io~1metrics
  - /metadata/annotations/autoscaling.alpha.kubernetes.io~1current-metrics

resource.customizations.ignoreResourceUpdates.apps_ReplicaSet: |
  jsonPointers:
    - /metadata/annotations/deployment.kubernetes.io~desired-replicas
    - /metadata/annotations/deployment.kubernetes.io~max-replicas
    - /metadata/annotations/deployment.kubernetes.io~revision

resource.customizations.ignoreResourceUpdates.discovery.k8s.io_EndpointSlice: |
  jsonPointers:
  - /endpoints
  - /ports
  - /metadata/annotations

resource.compareoptions: |
  ignoreDifferencesOnResourceUpdates: true
```

- ArgoCD watches all field changes of tracked objects.
- K8s fields could get very concise or frequently update, even for fields that we don't really need to reconcile.
- Use the features: ignoreResourceUpdates (available v2.8) [1] and, ignoreDifferences [2].
- 🔍 might need to enable debug log to find org-specific high-churn objects



fix: argocd apiclient

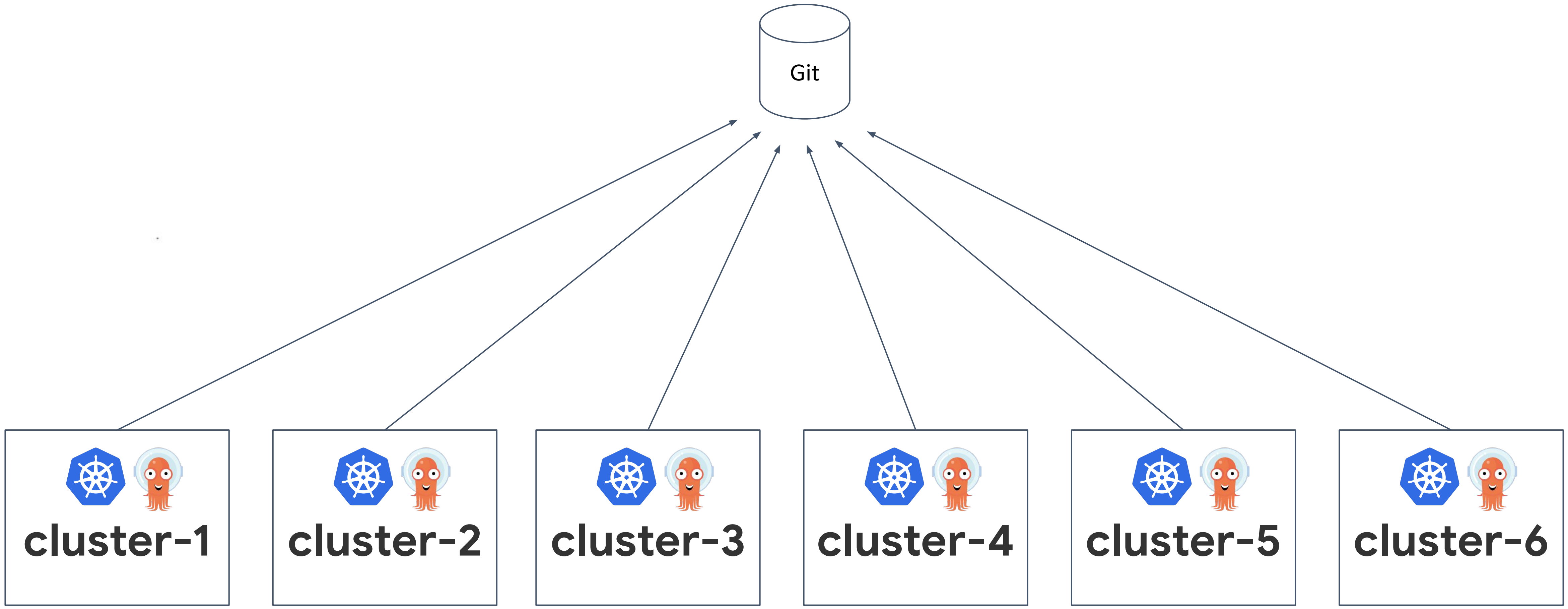
Problem: http2 GOAWAY / grpc ENHANCE_YOUR_CALM errors

🐞🔧 Fix missing grpc parameter when using `grpcWeb=true`

- Our in-house developer platform implements argocd's apiclient library.
- As we scale, we started seeing http2 GOAWAY errors from our platform.
- We investigated, found and fixed a bug in the apiclient library when using `grpcWeb=true` (#15707).
- Also, the argocd CLI may fallback to use `grpcWeb=true` even though `--grpc-web` flag is not specified.
- It may also implicitly use `grpcWeb=true`. Check your `.config/argocd/config` file!
- Alternatively, use native grpc.

Future Improvements

decentralized argocd (pull model)



decentralized argocd

Pros

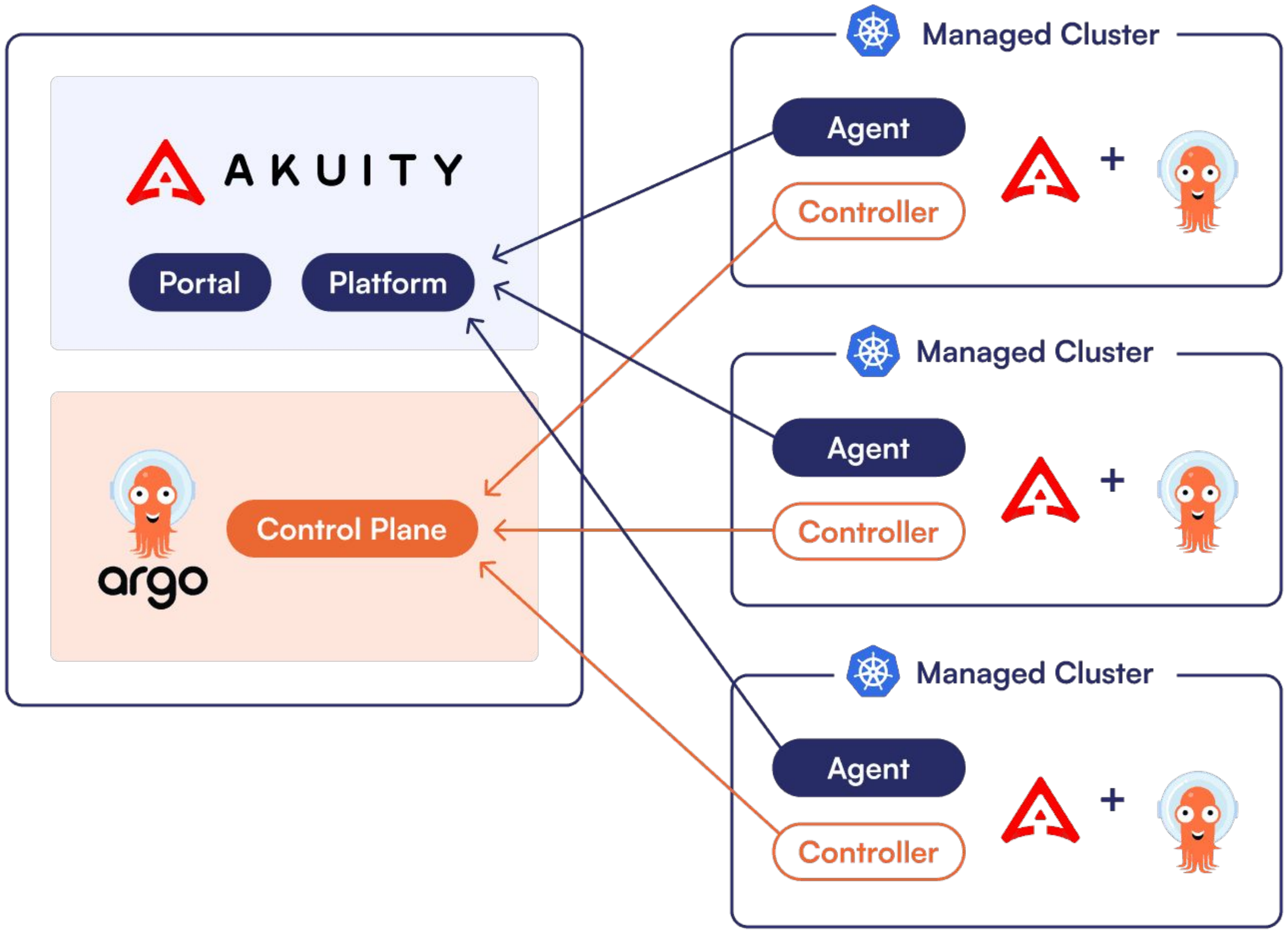
- Application controller workload distributed across clusters
- Access to Kubernetes API server is local only

Cons

- Maintenance and upgrade headache
- Automation headache: maintain multiple argocd client versions
- No more centralized dashboard
- Still require tuning for large clusters

agent-based argocd (hybrid model)

popularized by Akuity platform



Source: https://akuity.io/blog/argo-cd-architectures-explained/

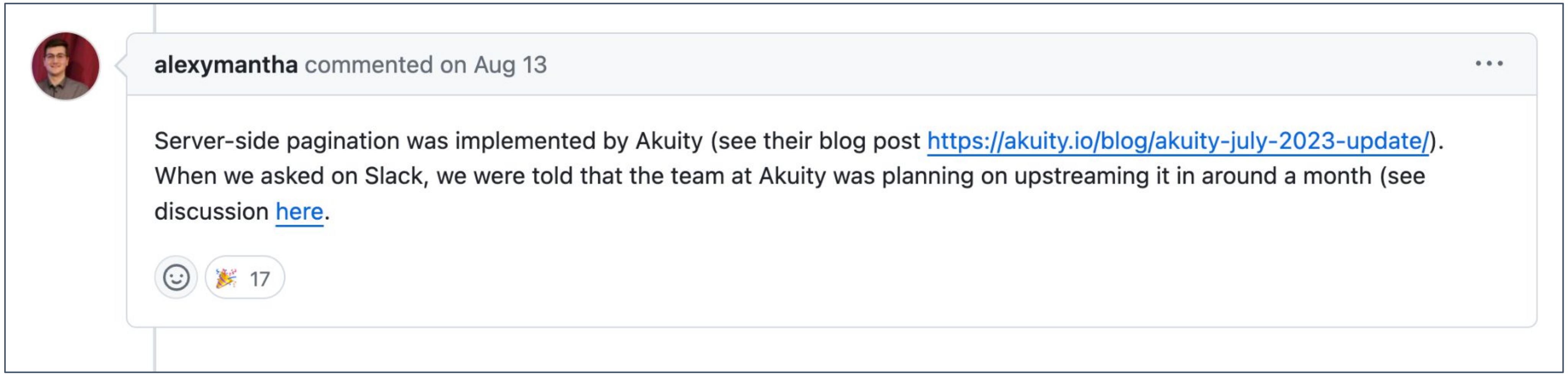
agent-based argocd (hybrid model)

WIP in the community

- Optional Pull Mechanism for ApplicationSet (merged: #10908)
- Centralized UI for Multiple ArgoCD Instances (open: #11498)

argocd ui

- Server-Side Pagination 🚀 (open: #14947)
- Already on Akuity, and they planned bring it to upstream 🎉



alexymantha commented on Aug 13

Server-side pagination was implemented by Akuity (see their blog post https://akuity.io/blog/akuity-july-2023-update/).
When we asked on Slack, we were told that the team at Akuity was planning on upstreaming it in around a month (see discussion here.

🎉 17

references

- TikTok, Managing Thousands of Apps with ArgoCD
- Adobe, Managing Hundreds of Clusters with ArgoCD
- Alexander Matyushentsev, ArgoCD Best Practices
- ArgoCD High Availability Documentation

Please scan the QR Code above
to leave feedback on this session

@girikuncoro

@yphanama

KubeCon | CloudNativeCon

North America 2023