Octopus Deploy

Multi-cluster, multi-app deployments with App Sets

GitOpsCon US

Kostis Kapelonis | December 2025

Kostis Kapelonis

Developer Advocate (Octopus Deploy/Codefresh)

Argo Maintainer (Argo CD, Argo Rollouts)

Co-author GitOps certification

http://learning.octopus.com

Agenda

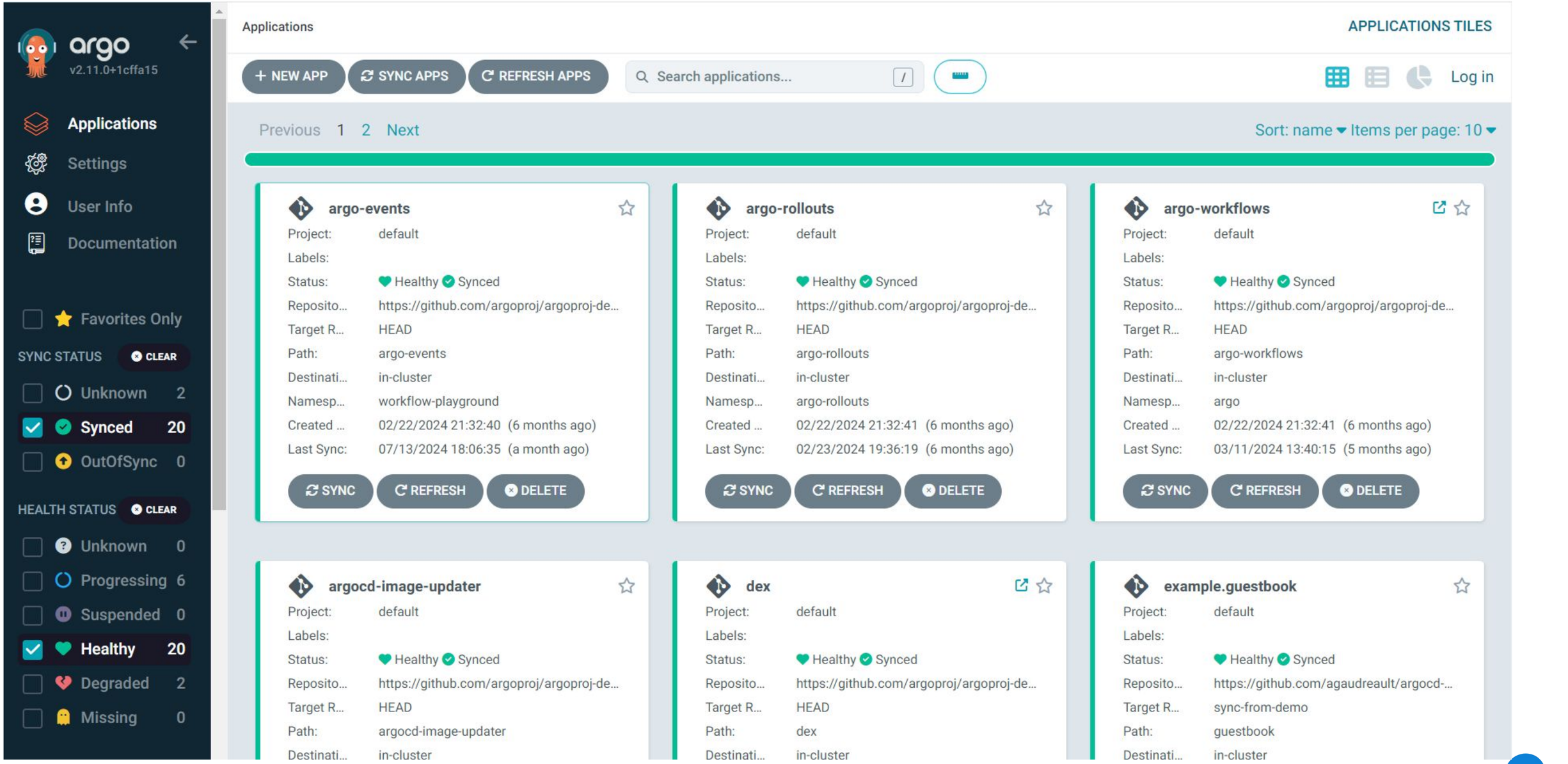
1. Understand Application Sets
2. Group Application Sets with App-of-Apps files
3. Avoid the Helm sandwich
4. Employ Cluster groups with Tags
5. Create Many Application Sets
6. Use the PR Generator for preview envs

Introduction

CNCF End User Survey Finds Argo CD as Majority Adopted GitOps Solution for Kubernetes

Nearly 60% of Kubernetes clusters managed by survey respondents now rely on Argo CD, with strong satisfaction fueled by 3.0 performance and security updates

https://www.cncf.io/announcements/2025/07/24/cncf-end-user-survey-finds-argo-cd-as-majority-adopted-gitops-solution-for-kubernetes/


argo
v2.11.0+1cffa15
Applications
Settings
User Info
Documentation
Favorites Only
SYNC STATUS
CLEAR
Unknown 2
Synced 20
OutOfSync 0
HEALTH STATUS
CLEAR
Unknown 0
Progressing 6
Suspended 0
Healthy 20
Degraded 2
Missing 0
Applications
APPLICATIONS TILES
+ NEW APP
SYNC APPS
REFRESH APPS
Search applications...
/
Log in
Previous 1 2 Next
Sort: name
Items per page: 10
argo-events
Project: default
Labels:
Status: Healthy Synced
Reposito... https://github.com/argoproj/argoproj-de...
Target R... HEAD
Path: argo-events
Destinati... in-cluster
Namesp... workflow-playground
Created ... 02/22/2024 21:32:40 (6 months ago)
Last Sync: 07/13/2024 18:06:35 (a month ago)
SYNC
REFRESH
DELETE
argo-rollouts
Project: default
Labels:
Status: Healthy Synced
Reposito... https://github.com/argoproj/argoproj-de...
Target R... HEAD
Path: argo-rollouts
Destinati... in-cluster
Namesp... argo-rollouts
Created ... 02/22/2024 21:32:41 (6 months ago)
Last Sync: 02/23/2024 19:36:19 (6 months ago)
SYNC
REFRESH
DELETE
argo-workflows
Project: default
Labels:
Status: Healthy Synced
Reposito... https://github.com/argoproj/argoproj-de...
Target R... HEAD
Path: argo-workflows
Destinati... in-cluster
Namesp... argo
Created ... 02/22/2024 21:32:41 (6 months ago)
Last Sync: 03/11/2024 13:40:15 (5 months ago)
SYNC
REFRESH
DELETE
argocd-image-updater
Project: default
Labels:
Status: Healthy Synced
Reposito... https://github.com/argoproj/argoproj-de...
Target R... HEAD
Path: argocd-image-updater
Destinati... in-cluster
dex
Project: default
Labels:
Status: Healthy Synced
Reposito... https://github.com/argoproj/argoproj-de...
Target R... HEAD
Path: dex
Destinati... in-cluster
example.guestbook
Project: default
Labels:
Status: Healthy Synced
Reposito... https://github.com/agaudreault/argocd-...
Target R... sync-from-demo
Path: guestbook
Destinati... in-cluster

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    server: https://kubernetes.default.svc
    namespace: guestbook
```

Name of application

Where to read the Kubernetes manifest from

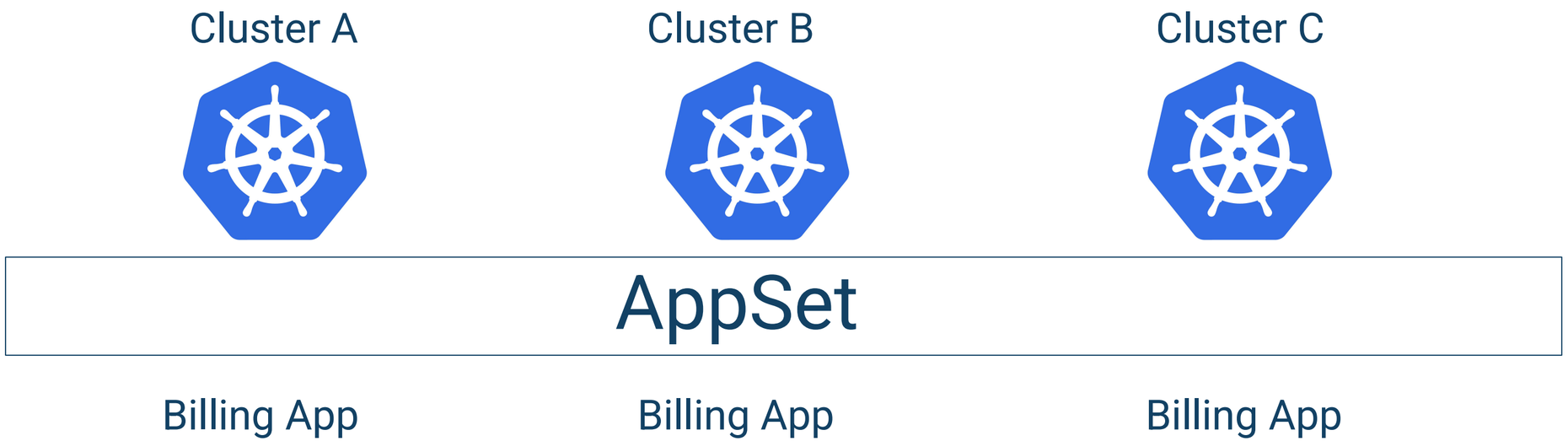
Which cluster to deploy the application to

Always use ApplicationSets

- Understand the generators
- Learn how to combine them
- Don’t miss the PR generator

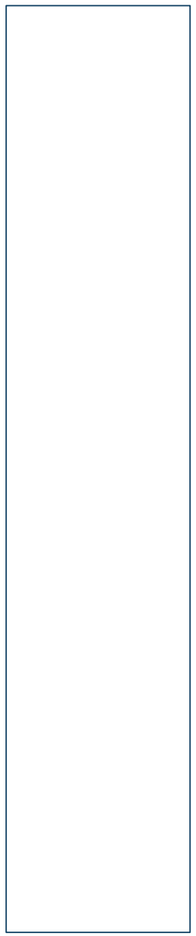
Application Set 101

Same App - different clusters



Same cluster- different apps

AppSet



Cluster

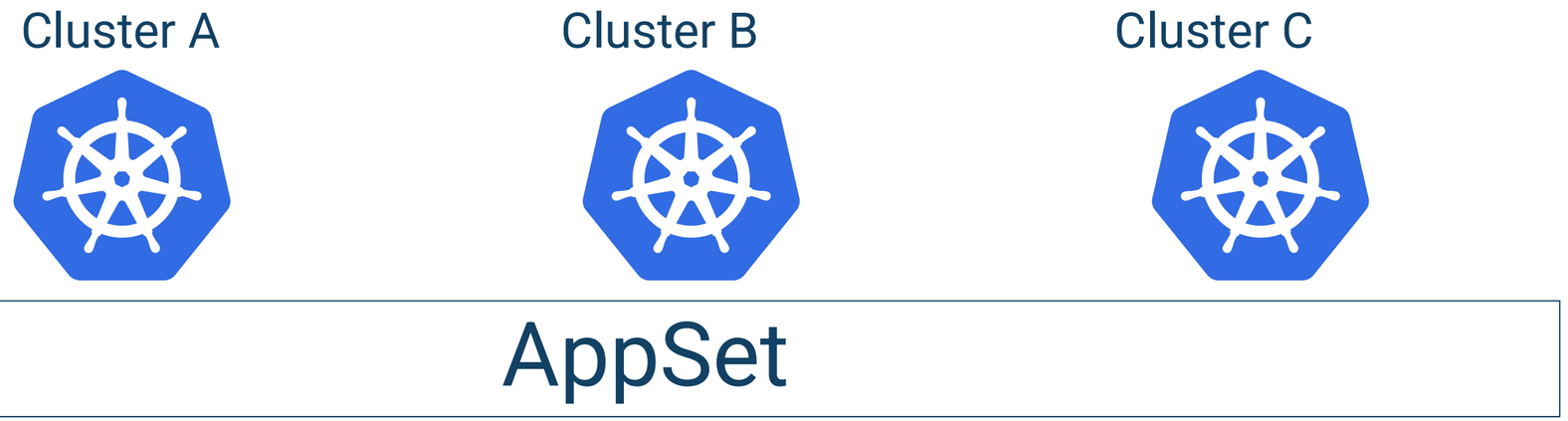
Billing App

Sealed Secrets

Argo Rollouts

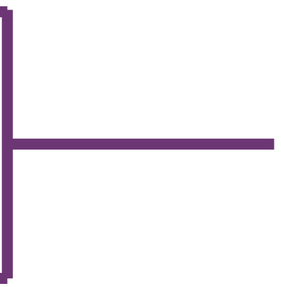
Cert Manager

Many Apps - Many Clusters



Generators

- List
- Cluster
- Git
- SCM Provider
- Pull Request
- Cluster Decision
- Matrix
- Merge
- Plugin



Combine and Extend Generators!

Forget individual Aps

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    server: https://kubernetes.default.svc
    namespace: guestbook
```

Name of application

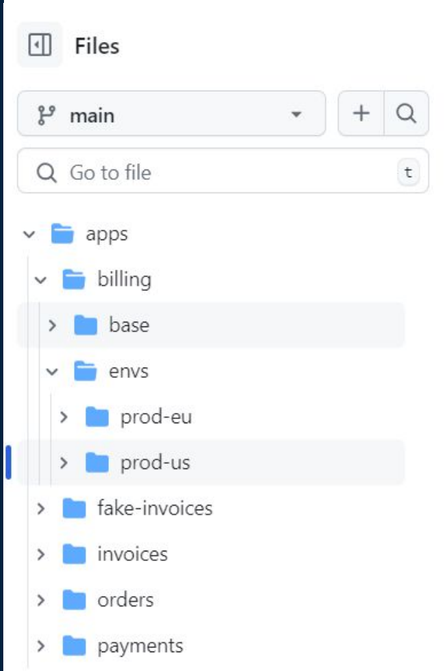
Where to read the Kubernetes manifest

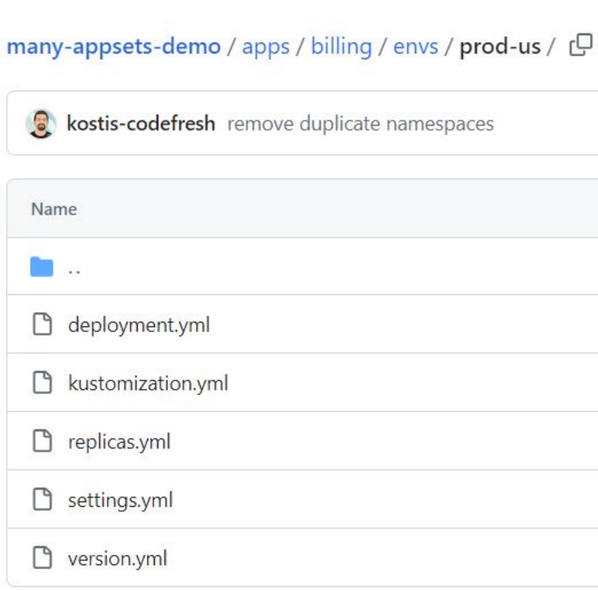
Which cluster to deploy the application to

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: my-qa-appset
  namespace: argocd
spec:
  goTemplate: true
  goTemplateOptions: ["missingkey=error"]
  generators:
  - git:
    repoURL: https://github.com/kostis-codefresh/many-appsets-demo.git
    revision: HEAD
    directories:
    - path: apps/*/envs/qa
  template:
  metadata:
    name: '{{index .path.segments 1}}-{{index .path.segments 3}}'
  spec:
    # The project the application belongs to.
    project: default

    # Source of the application manifests
    source:
      repoURL: https://github.com/kostis-codefresh/many-appsets-demo.git
      targetRevision: HEAD
      path: '{{.path.path}}'

    # Destination cluster and namespace to deploy the application
    destination:
      server: https://kubernetes.default.svc
      namespace: '{{index .path.segments 1}}-{{index .path.segments 3}}'
```





Generate applications from Git folders

Cluster bootstrapping

argo
v2.10.6+d504d2b

- Applications
- Settings
- User Info
- Documentation

Favorites Only

SYNC STATUS
- Unknown 0
- Synced 13
- OutOfSync 0

HEALTH STATUS
- Unknown 0
- Progressing 12
- Suspended 0
- Healthy 1
- Degraded 0
- Missing 0

LABELS

LABELS

PROJECTS

Applications

APPLICATIONS LIST

+ NEW APP | SYNC APPS | REFRESH APPS | Search applications... / | Log out

Previous 1 2 Next — Sort: name — Items per page: 10

Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/appsets	HEAD	Healthy	
Name: all-apps	Destination:	in-cluster/argocd		Synced	
Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/apps/billing/envs/prod-eu	HEAD	Progressing	
Name: billing-prod-eu	Destination:	in-cluster/billing-prod-eu		Synced	
Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/apps/billing/envs/prod-us	HEAD	Progressing	
Name: billing-prod-us	Destination:	in-cluster/billing-prod-us		Synced	
Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/apps/fake-invoices/envs/qa	HEAD	Progressing	
Name: fake-invoices-qa	Destination:	in-cluster/fake-invoices-qa		Synced	
Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/apps/invoices/envs/prod-eu	HEAD	Progressing	
Name: invoices-prod-eu	Destination:	in-cluster/invoices-prod-eu		Synced	
Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/apps/invoices/envs/prod-us	HEAD	Progressing	
Name: invoices-prod-us	Destination:	in-cluster/invoices-prod-us		Synced	
Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/apps/invoices/envs/qa	HEAD	Progressing	
Name: invoices-qa	Destination:	in-cluster/invoices-qa		Synced	
Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/apps/invoices/envs/staging	HEAD	Progressing	
Name: invoices-staging	Destination:	in-cluster/invoices-staging		Synced	
Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/apps/orders/envs/prod-us	HEAD	Progressing	
Name: orders-prod-us	Destination:	in-cluster/orders-prod-us		Synced	
Project: default	Source:	https://github.com/kostis-codefresh/many-appsets-demo.git/apps/orders/envs/qa	HEAD	Progressing	
Name: orders-qa	Destination:	in-cluster/orders-qa		Synced	

Previous 1 2 Next — Sort: name — Items per page: 10

A good starting point

- Store your applications with Kustomize Overlays OR
- Store your applications with Helm value Hierarchies
- Use the Git generator to load apps
- Use the Cluster generator to assign apps to clusters
- Create one applicationset per environment or cluster type

HOME > BLOG

BEST PRACTICES

How to Structure Your Argo CD Repositories Using Application Sets

21 min read

Kostis Kapelonis May 17, 2024

https://codefresh.io/blog/how-to-structure-your-argo-cd-repositories-using-application-sets/

Confusion with App-of-Apps

They are not mutually exclusive

What to choose?

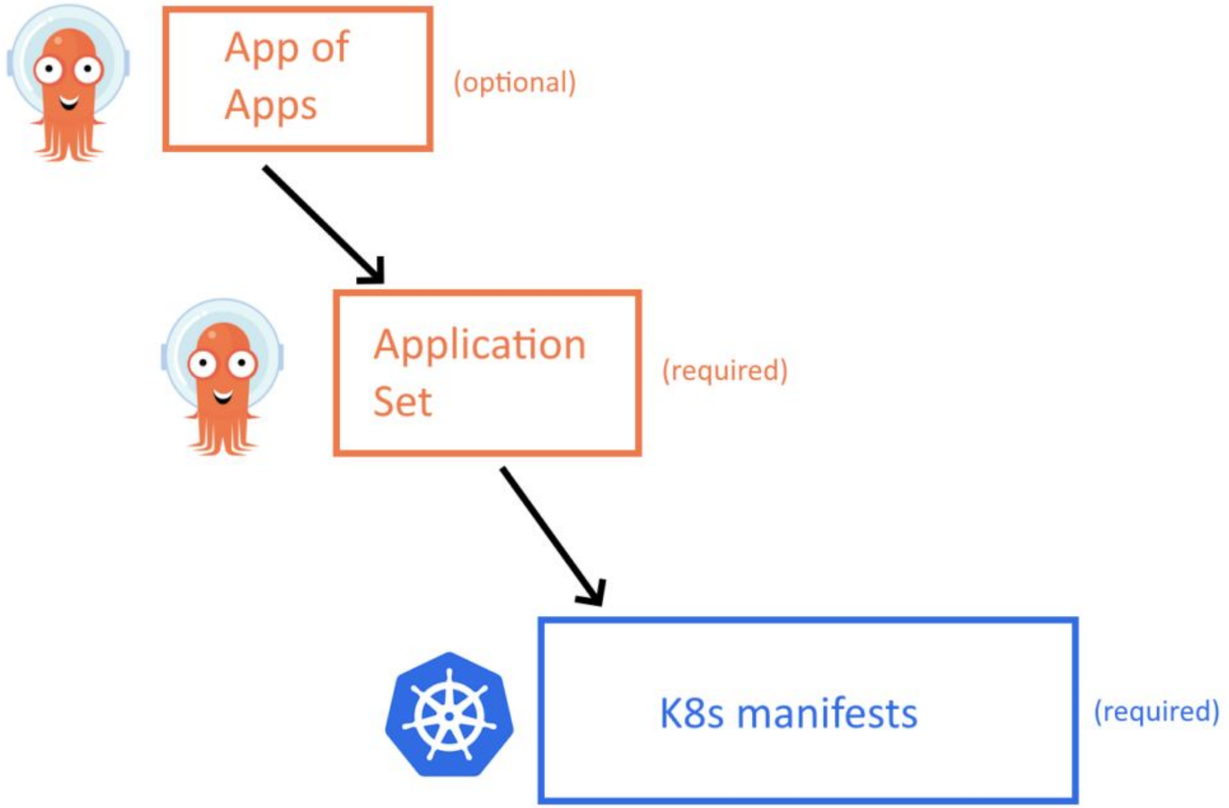
“Should I use Application Sets or App of Apps”?

“Choosing” between App-of-apps and App Sets

- “I prefer app-of-apps and not application sets”
- “I haven’t explored application sets yet. Still using App-of-Apps”
- “I tried Application Sets but found them too complex. Now using App-of-Apps”
- “I am using App-Of-Apps to do X. Can I do X with ApplicationSets?”
- “This is my setup. Do you recommend App-Of-Apps or Application Sets?”

ApplicationSets AND App-Of-Apps

- Application Sets are NOT a replacement for App-of-apps
- You should use them together
- Group many ApplicationSets in a root App
- Great for cluster bootstraping



Don’t use Helm

…for Argo CD manifests

How do we template Kubernetes manifests?

Do

Ops

Let's use Helm !

Ops

Don't

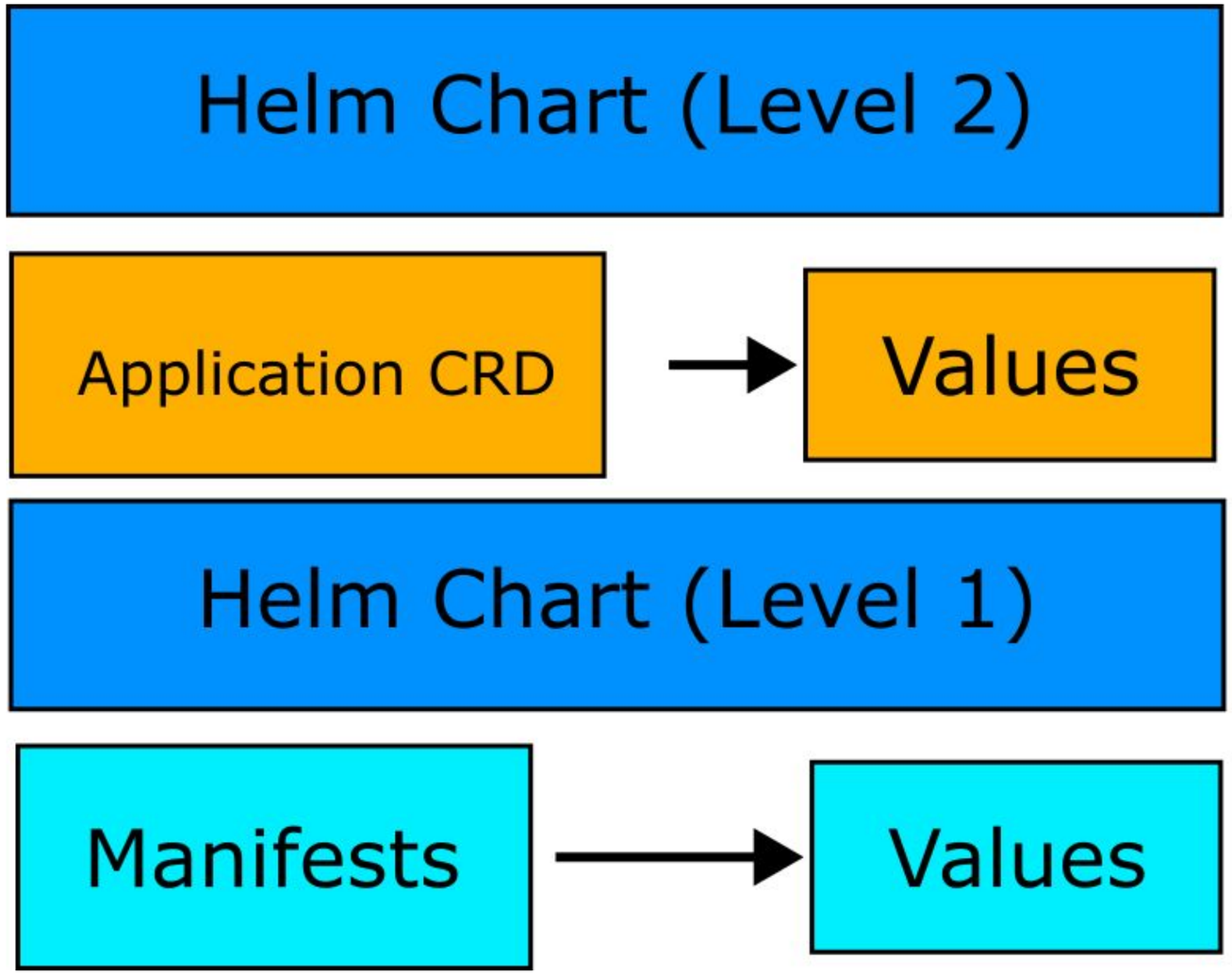
How do we template Argo CD Applications?

Ops

Let's use Helm !

Ops

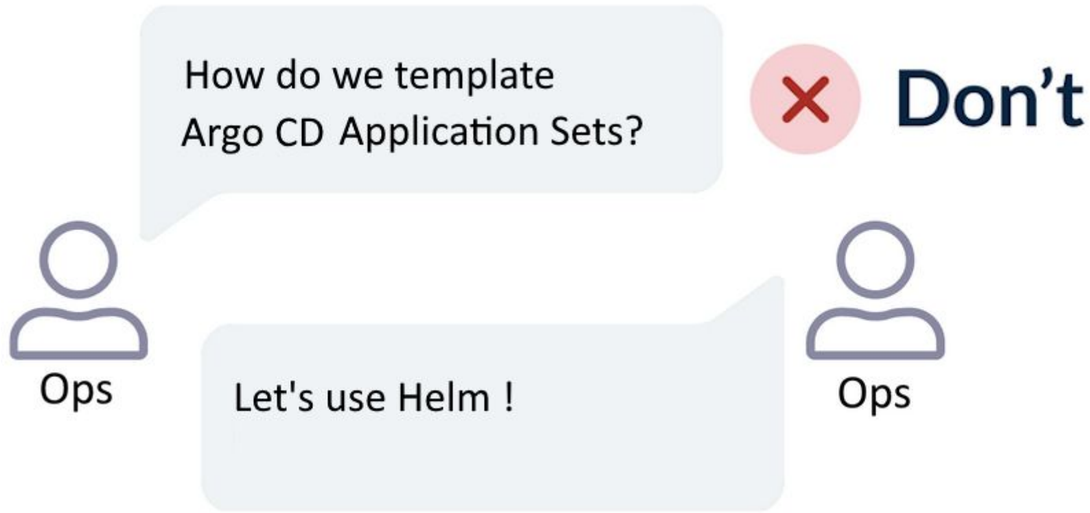
Two layers of Helm - “Helm Sandwich”



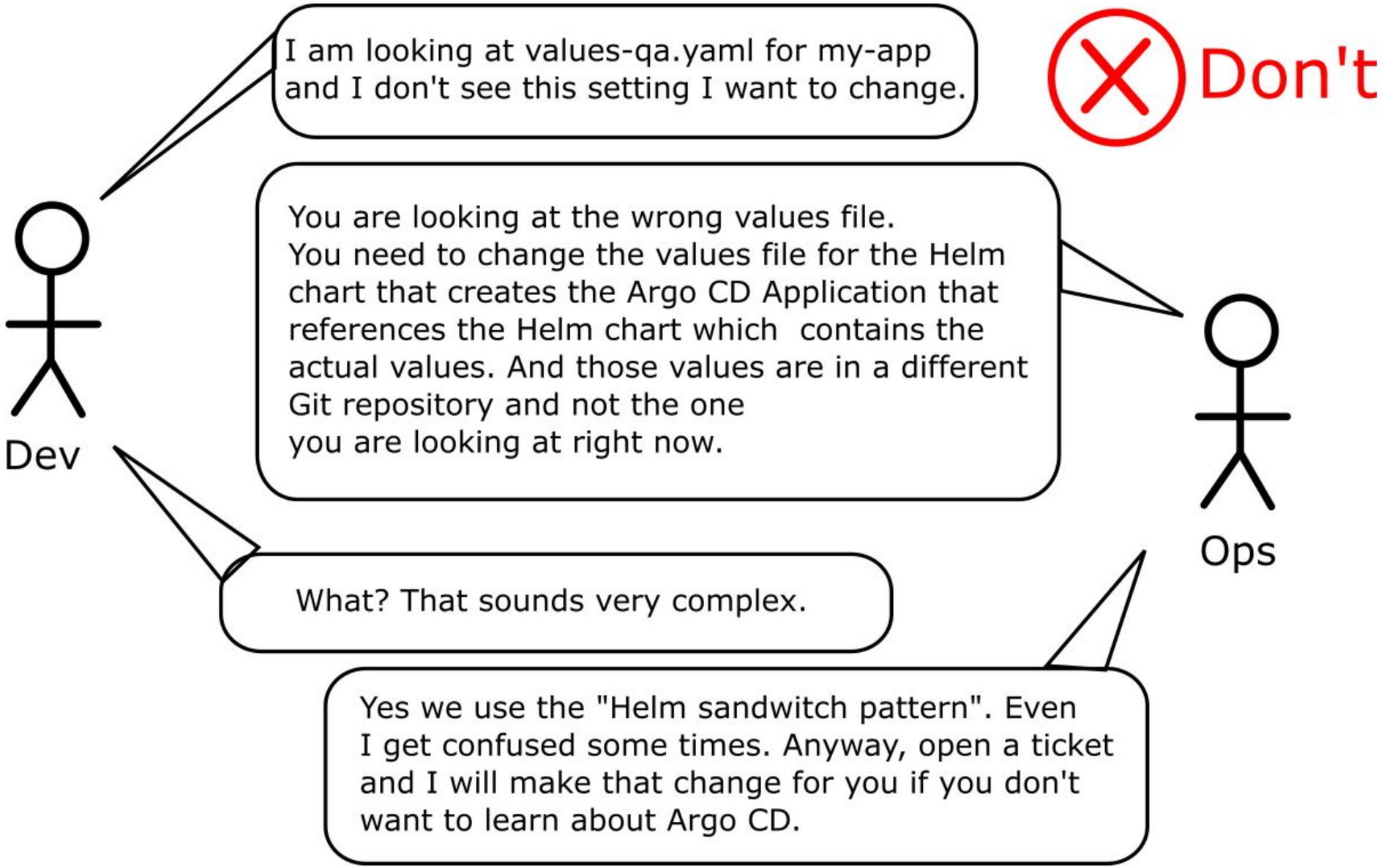
Don't

Application Sets already support templating

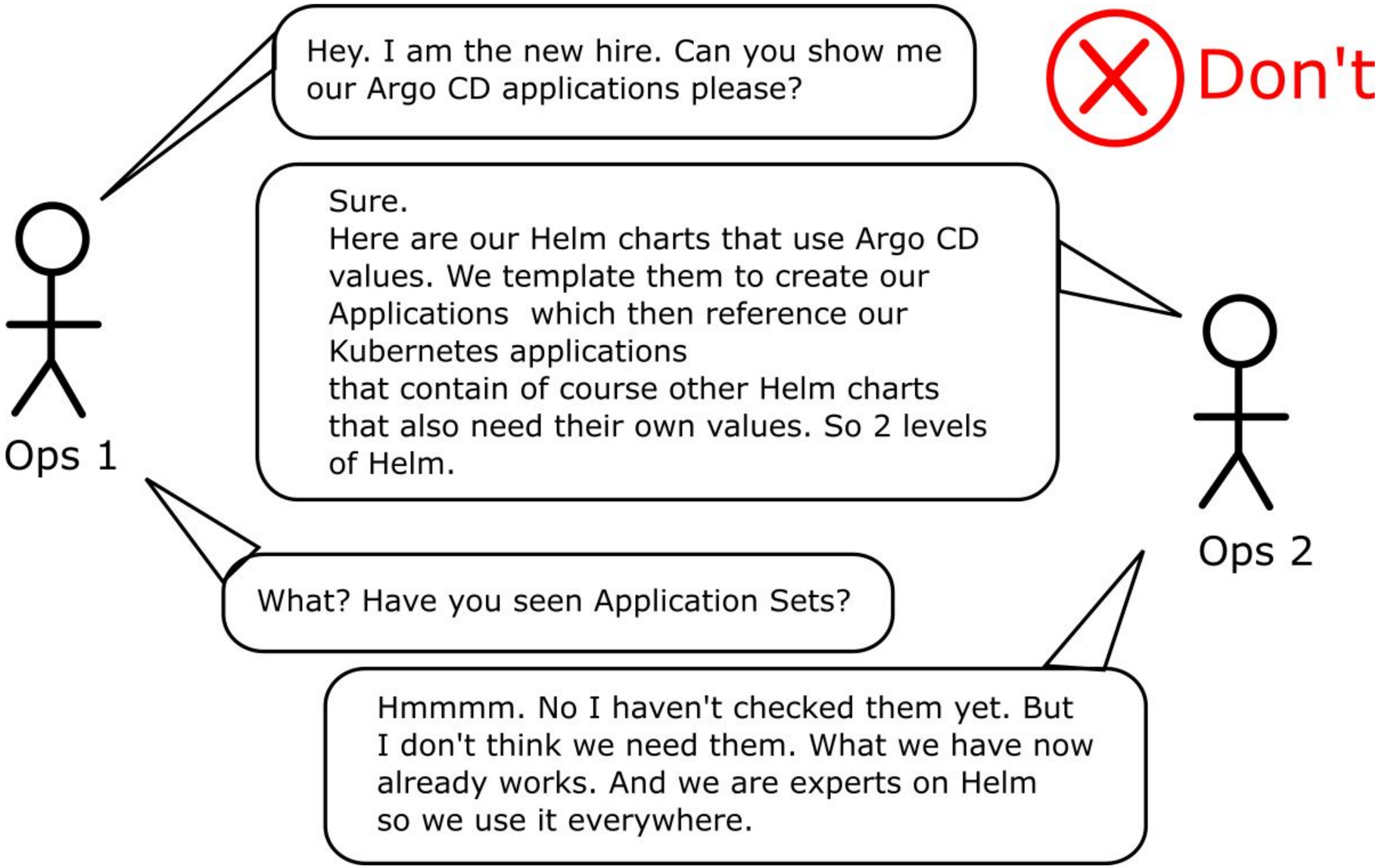
- Don’t use Helm as a hammer…
- No need for extra complexity
- It is difficult to debug Helm Application CRDs that point to Helm charts



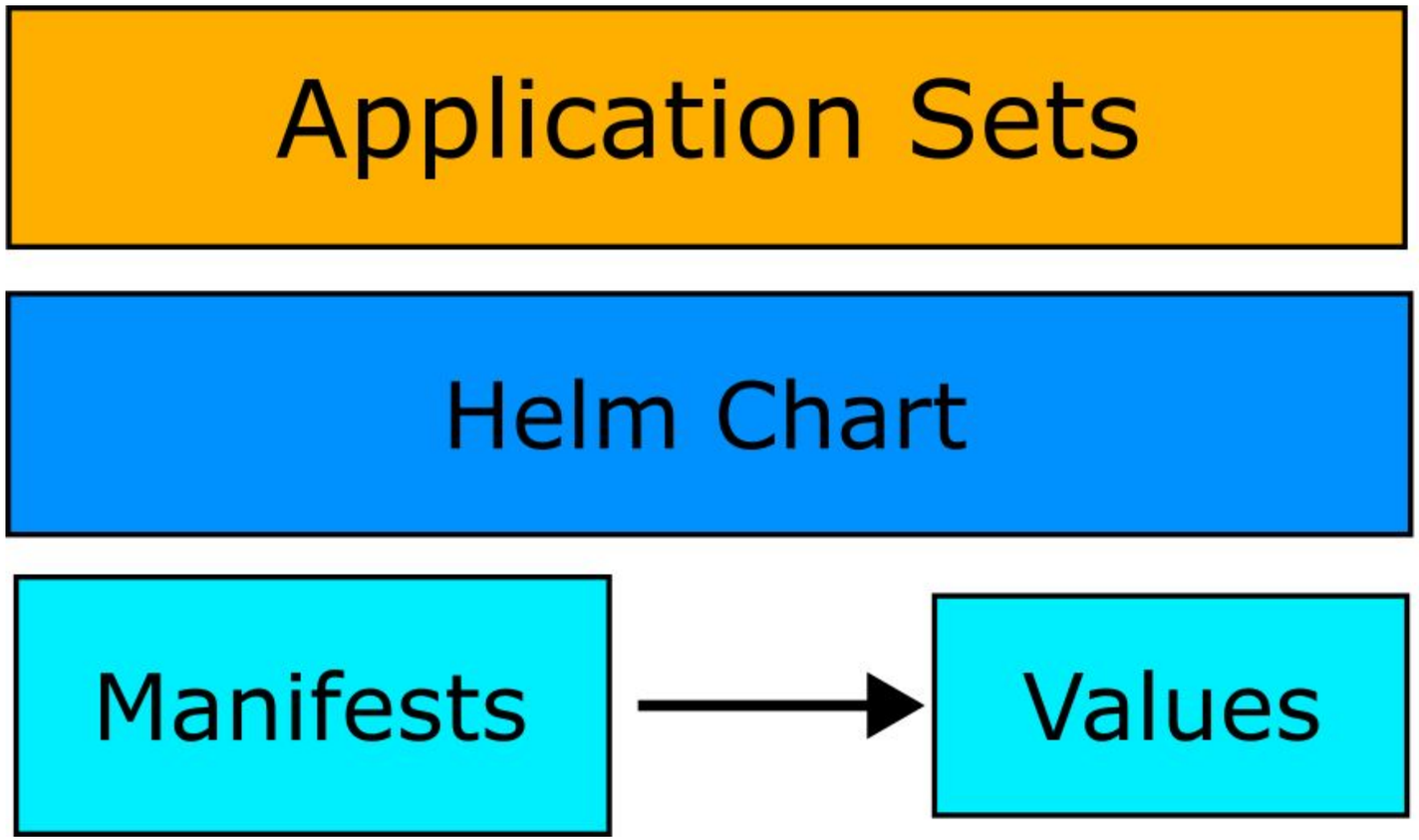
Dev experience



Ops experience



Solution: use Application Sets



Do

Application Set templating

Sprig Function Documentation

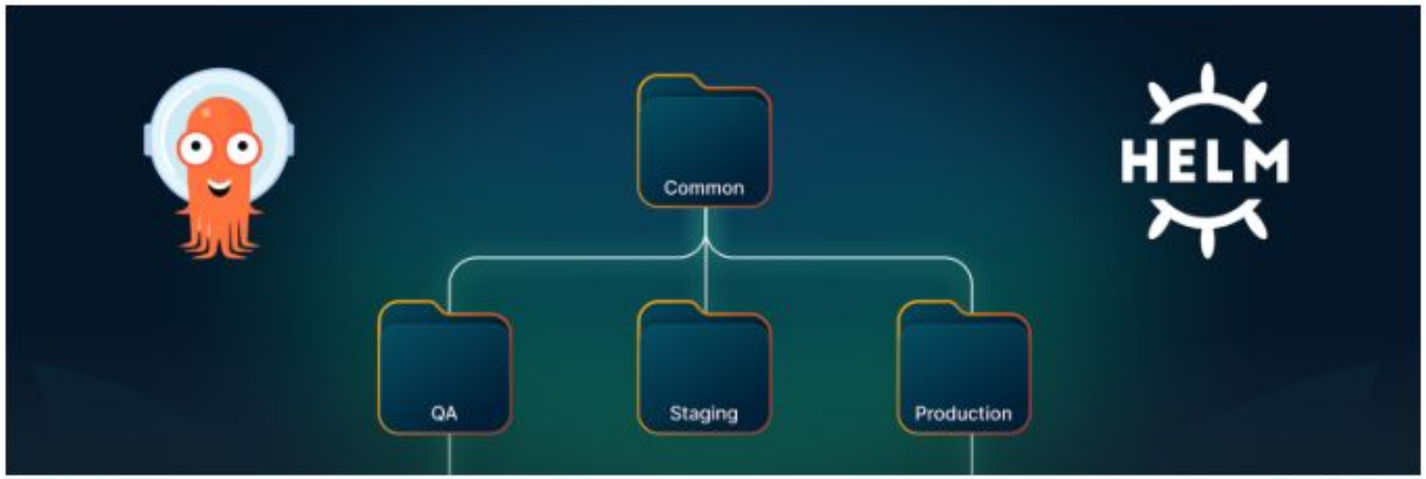
The Sprig library provides over 70 template functions for Go's template language.

- String Functions: `trim`, `wrap`, `randAlpha`, `plural`, etc.
  - String List Functions: `splitList`, `sortAlpha`, etc.
- Integer Math Functions: `add`, `max`, `mul`, etc.
  - Integer Slice Functions: `until`, `untilStep`
- Float Math Functions: `addf`, `maxf`, `mulf`, etc.
- Date Functions: `now`, `date`, etc.
- Defaults Functions: `default`, `empty`, `coalesce`, `fromJson`, `toJson`, `toPrettyJson`, `toRawJson`, `ternary`
- Encoding Functions: `b64enc`, `b64dec`, etc.
- Lists and List Functions: `list`, `first`, `uniq`, etc.
- Dictionaries and Dict Functions: `get`, `set`, `dict`, `hasKey`, `pluck`, `dig`, `deepCopy`, etc.
- Type Conversion Functions: `atoi`, `int64`, `toString`, etc.
- Path and Filepath Functions: `base`, `dir`, `ext`, `clean`, `isAbs`, `osBase`, `osDir`, `osExt`, `osClean`, `osIsAbs`
- Flow Control Functions: `fail`
- Advanced Functions
  - UUID Functions: `uuidv4`
  - OS Functions: `env`, `expandenv`
  - Version Comparison Functions: `semver`, `semverCompare`
  - Reflection: `typeOf`, `kindIs`, `typeIsLike`, etc.
  - Cryptographic and Security Functions: `derivePassword`, `sha256sum`, `genPrivateKey`, etc.
  - Network: `getHostByName`
  - URL: `urlParse`, `urlJoin`

Usage example

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: test-appset
spec:
  ...
  template:
    metadata:
      name: 'hellos3-{{.name}}-{{ cat .branch | slugify 23 }}'
      annotations:
        label-1: '{{ cat .branch | slugify }}'
        label-2: '{{ cat .branch | slugify 23 }}'
        label-3: '{{ cat .branch | slugify 50 false }}'
```

HOME > BLOG



BEST PRACTICES

Using Helm Hierarchies in Multi-Source Argo CD Applications for Promoting to Different GitOps Environments

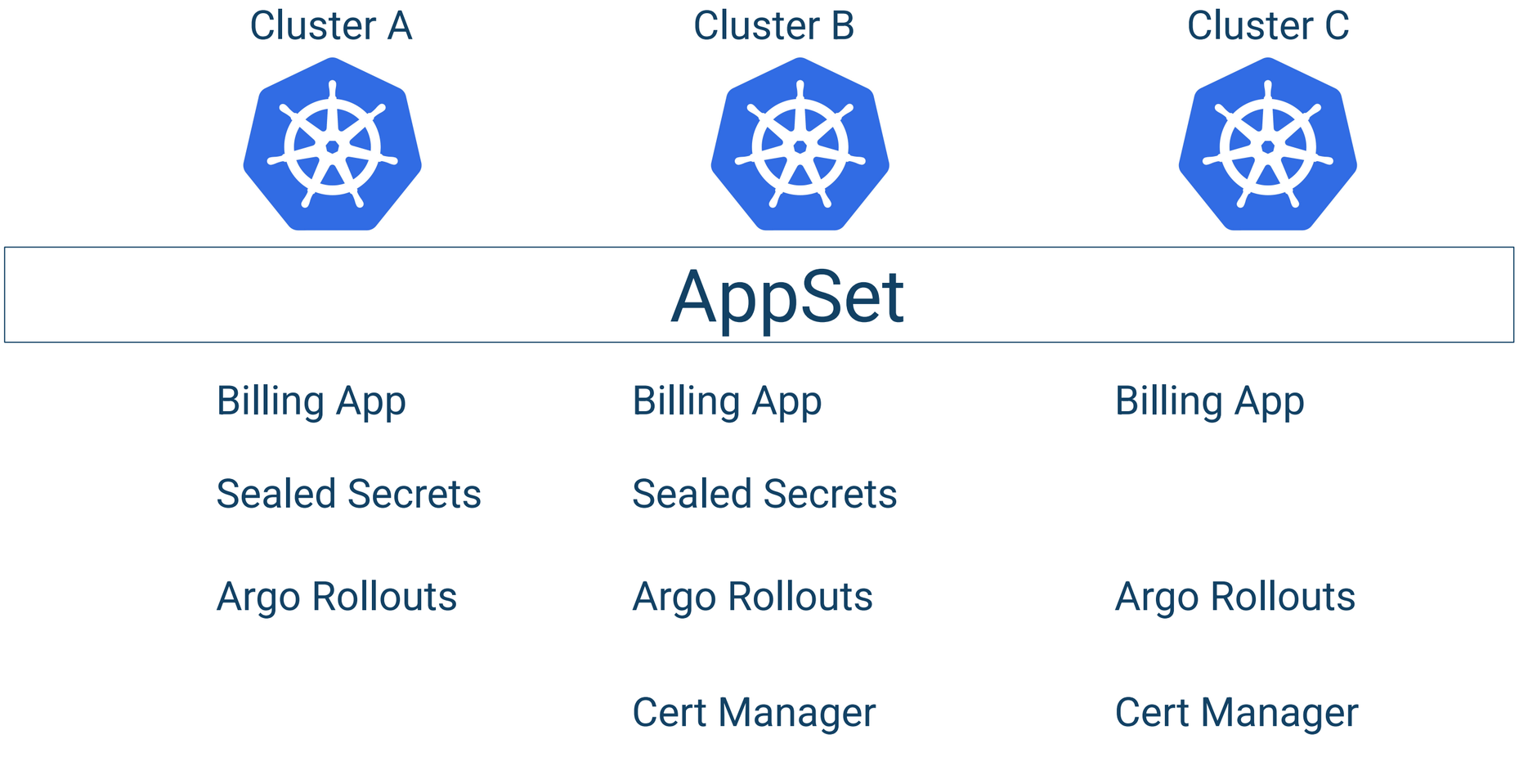
14 min read

https://codefresh.io/blog/helm-values-argocd/

Use Cluster Groups with tags

Pet vs Cattle

Many Apps - Many Clusters



```yaml
- merge:
    mergeKeys:
      - app
    generators:
      - list:
          elements:
            - app: external-dns
              appPath: infra/helm-charts/external-dns
              namespace: dns
            - app: argocd
              appPath: infra/helm-charts/argocd
              namespace: argocd
            - app: external-secrets
              appPath: infra/helm-charts/external-secrets
              namespace: external-secrets
            - app: kyverno
              appPath: infra/helm-charts/kyverno
              namespace: kyverno
      - list:
          elements:
            - app: external-dns
              enabled: "true"
            - app: argocd
              enabled: "true"
            - app: external-secrets
              enabled: "false"
            - app: kyverno
              enabled: "true"
  selector:
    matchLabels:
      enabled: "true"
```

Trying to define exactly what goes into each cluster

Tag your clusters

QA

Non-prod

AWS

Region B

Region A

Non-prod

GCP

Staging

US-East

Region A

Prod

GCP

US-West

Region A

Prod

GCP

Non-prod

AWS

Region B

Load Testing

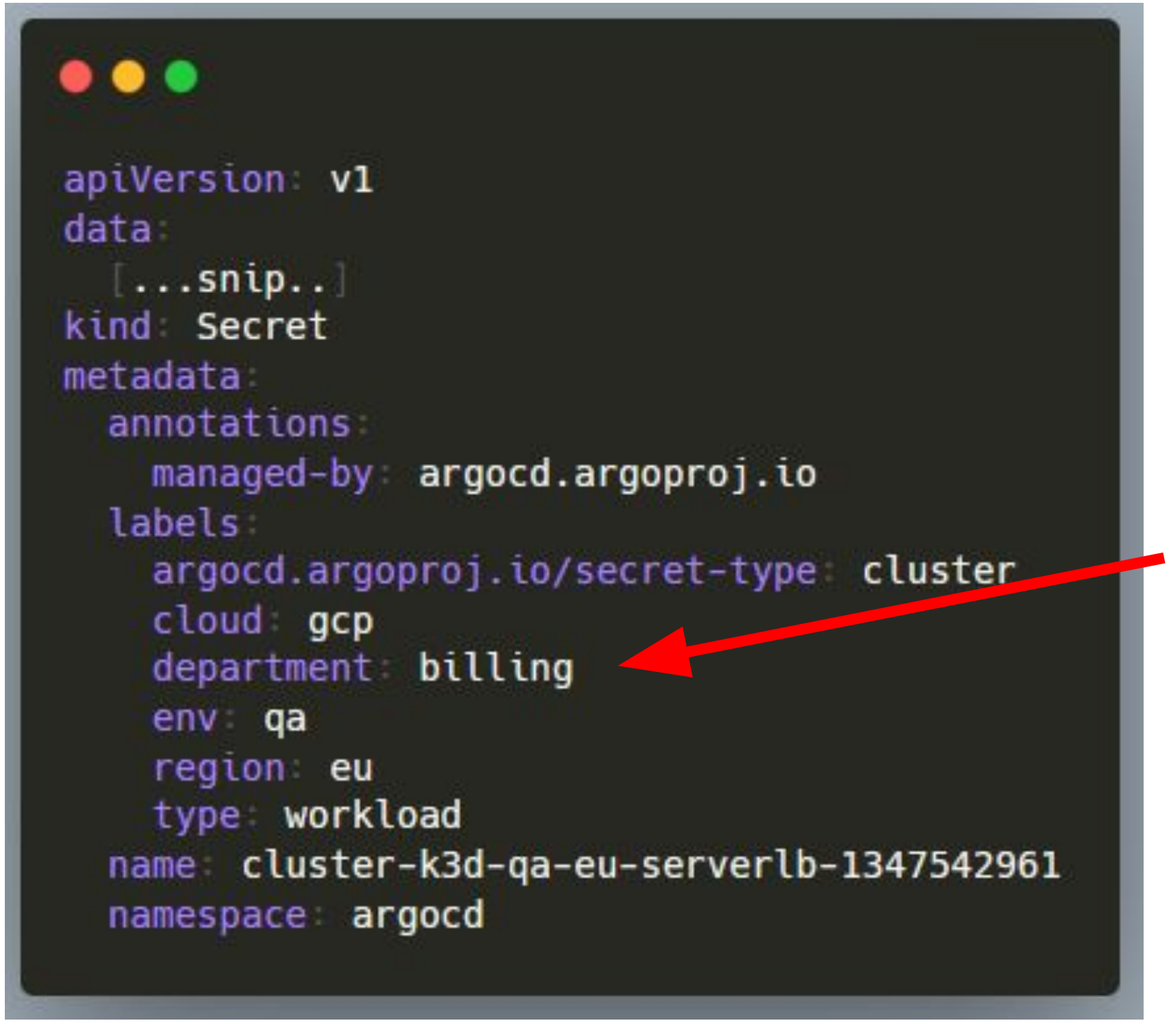
Prod

AWS

Region B

EU

Labels are in cluster secrets



This cluster belongs to the:

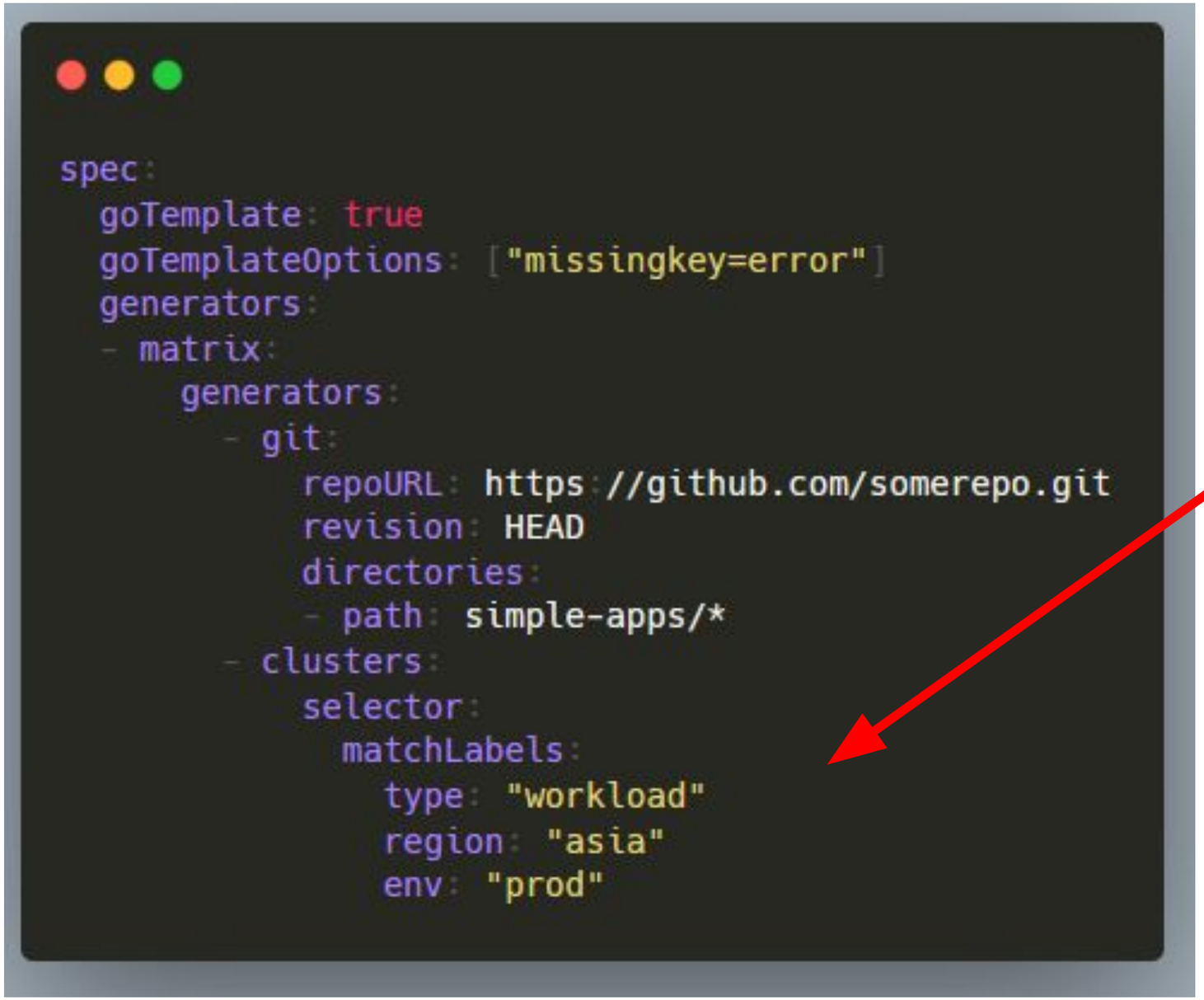
- gcp
- qa
- EU
- workload
- billing

…cluster groups

Use Tags with apps

- “In all my AWS clusters I want these apps” -> Create application Set
- “In all my Prod clusters I want these apps “ -> Create Application Set
- “In all my US Region clusters I want these apps” -> Create Application Set
- “In all my Staging clusters I want these apps” -> Create Application Set

Choose cluster group (And mode)



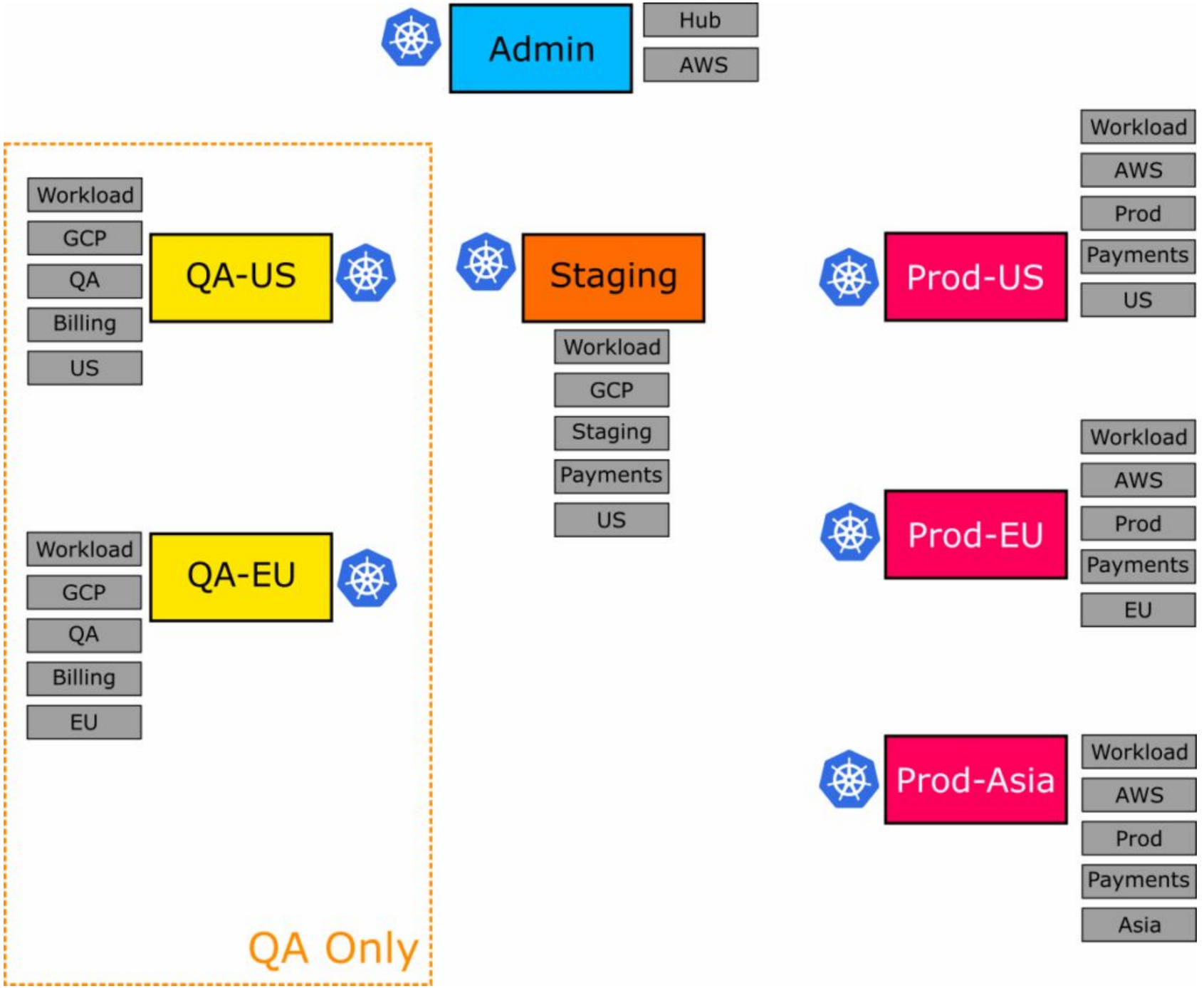
```yaml
spec:
  goTemplate: true
  goTemplateOptions: ["missingkey=error"]
  generators:
  - matrix:
      generators:
        - git:
            repoURL: https://github.com/somerepo.git
            revision: HEAD
            directories:
            - path: simple-apps/*
        - clusters:
            selector:
              matchLabels:
                type: "workload"
                region: "asia"
                env: "prod"
```

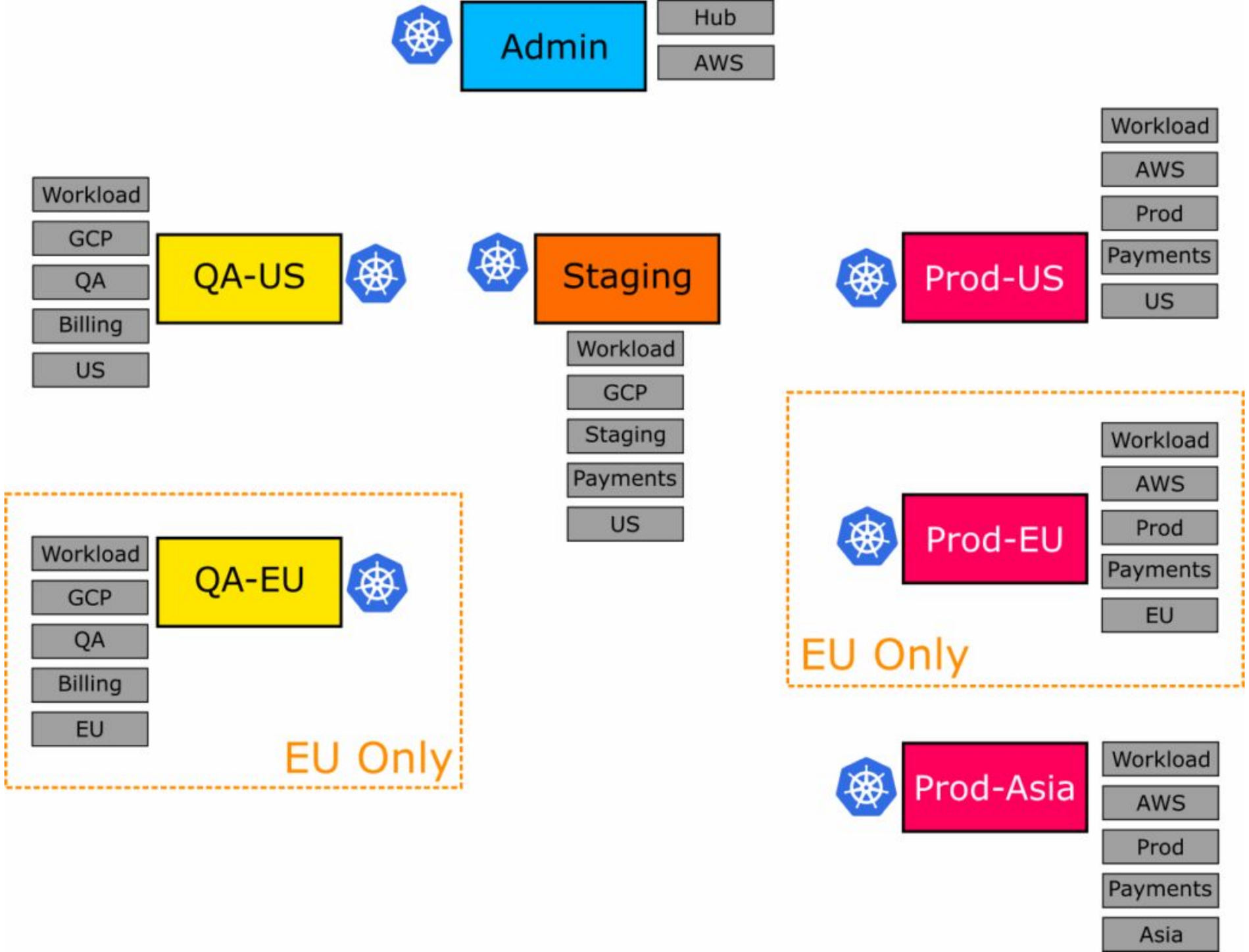
Choose all clusters that are in Production AND in Asia AND with type “workload”

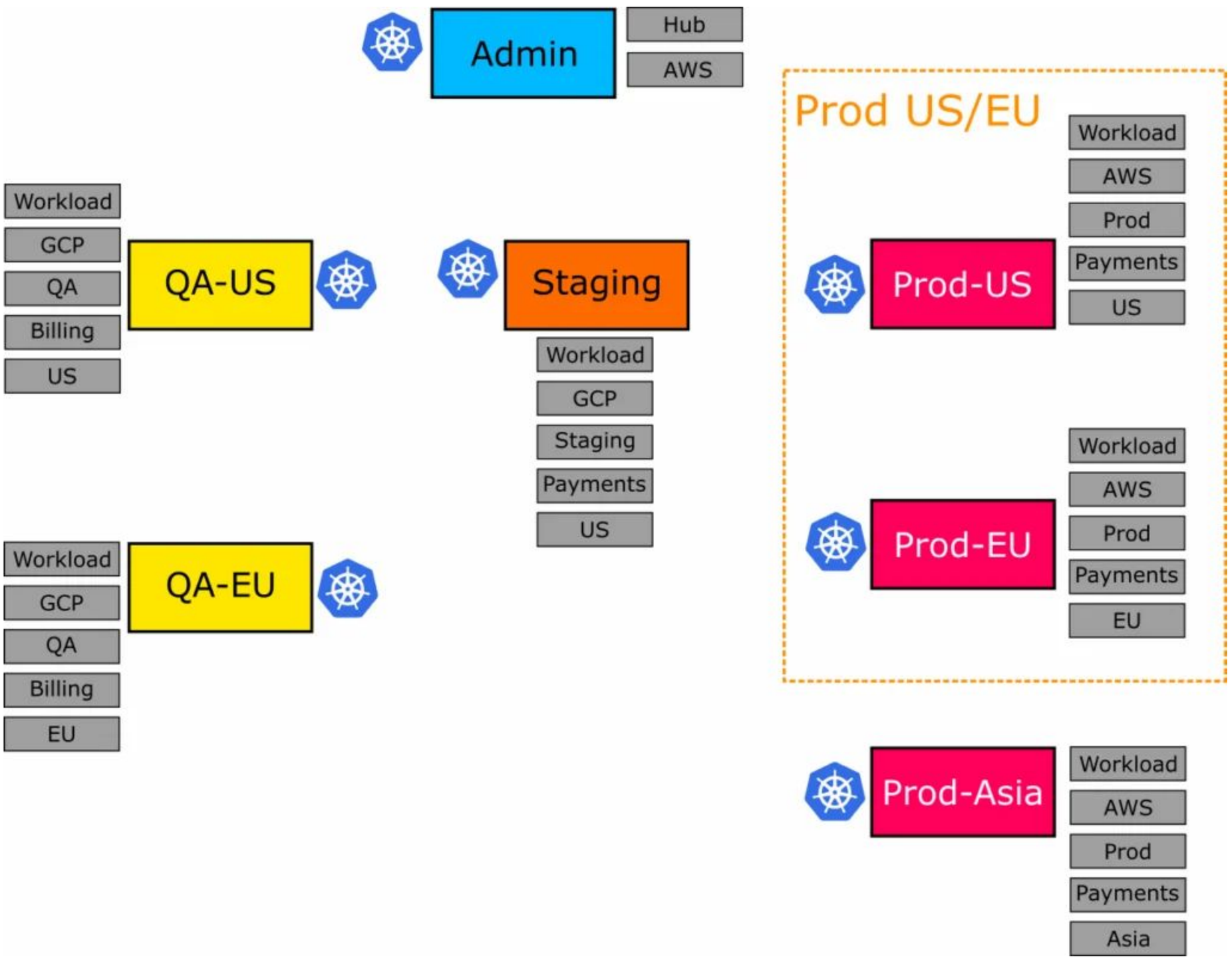
Choose cluster group (Or mode)


```yaml
spec:
  goTemplate: true
  goTemplateOptions: ["missingkey=error"]
  generators:
  - matrix:
      generators:
        - clusters:
            selector:
              matchLabels:
                type: "workload"
                env: "prod"
              matchExpressions:
              - key: region
                operator: In
                values:
                  - "eu"
                  - "us"
        - git:
            repoURL: https://github.com/somerepo.git
            revision: HEAD
            directories:
            - path: 'kustomize-apps/*/envs/{{.name}}'
```

Choose all clusters that are in Production AND with type “workload” and in region EU OR US)


Admin
Hub
AWS
Workload
GCP
QA
Billing
US
QA-US
Staging
Workload
GCP
Staging
Payments
US
Prod-US
Workload
AWS
Prod
Payments
US
Workload
GCP
QA
Billing
EU
QA-EU
Prod-EU
Workload
AWS
Prod
Payments
EU
Prod-Asia
Workload
AWS
Prod
Payments
Asia
QA Only


Admin
Hub
AWS
Workload
GCP
QA
Billing
US
QA-US
Staging
Workload
GCP
Staging
Payments
US
Workload
AWS
Prod
Payments
US
Prod-US
Workload
GCP
QA
Billing
EU
QA-EU
EU Only
Workload
AWS
Prod
Payments
EU
Prod-EU
EU Only
Prod-Asia
Workload
AWS
Prod
Payments
Asia


Admin
Hub
AWS
Prod US/EU
Workload
AWS
Prod
Payments
US
Prod-US
Workload
GCP
QA
Billing
US
QA-US
Staging
Workload
GCP
Staging
Payments
US
Workload
AWS
Prod
Payments
EU
Prod-EU
Workload
GCP
QA
Billing
EU
QA-EU
Prod-Asia
Workload
AWS
Prod
Payments
Asia

New cluster?

Region A

Non-prod GCP

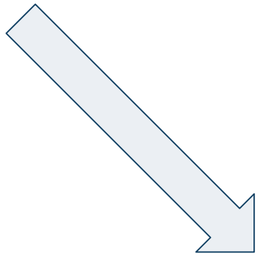
Staging

QA

Non-prod

AWS

Region B



US-East

Region A

Prod GCP

US-West

Region A

Prod GCP

Non-prod

AWS

Region B

Load Testing

?

?

?

Brand new Cluster

EU

Prod AWS

Region B

New Cluster bootstrapping

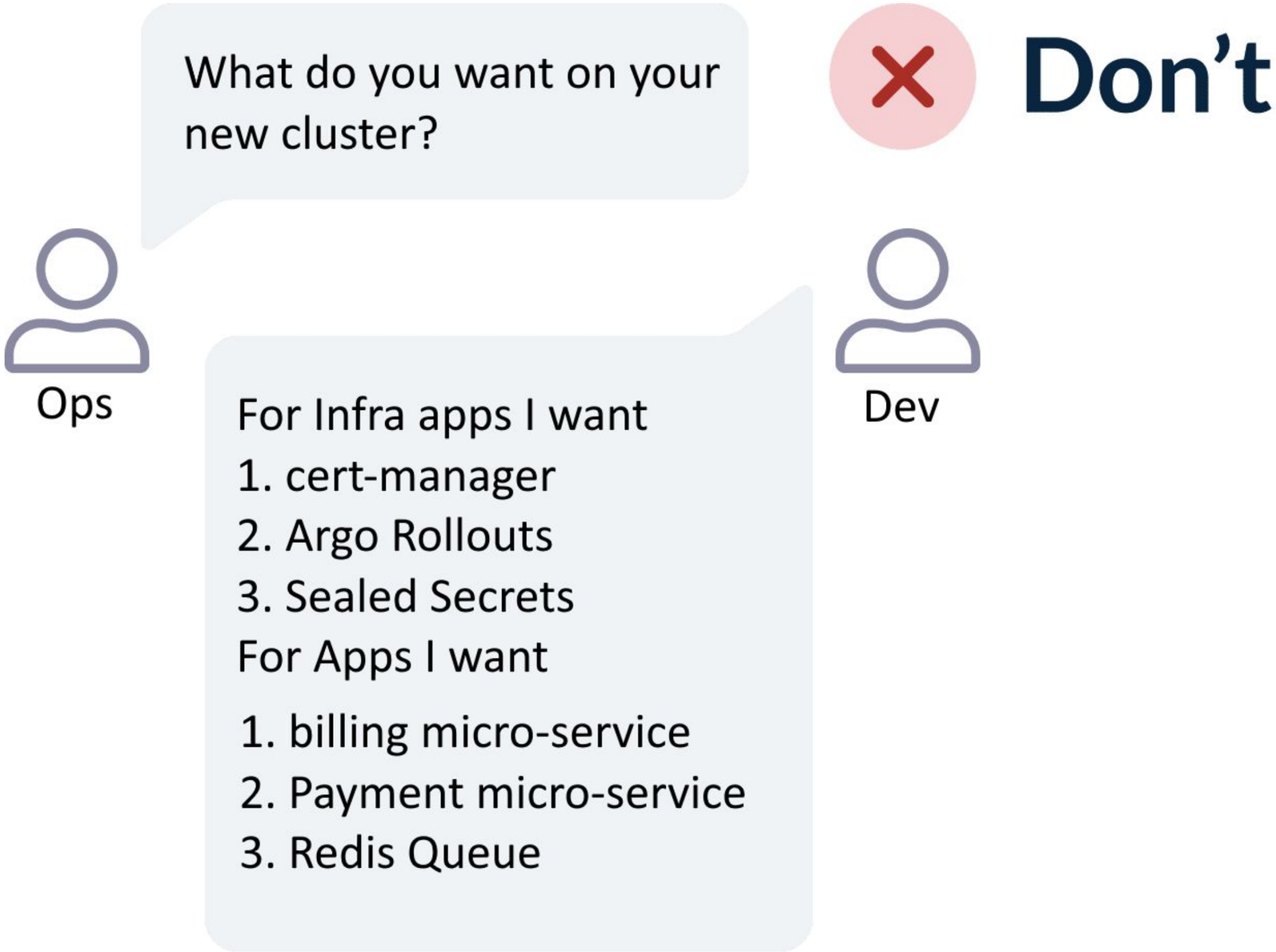
1. Create cluster with Terraform/Pulumi/Crossplane etc.

2. Assign tags to cluster “This is a prod cluster in GCP”

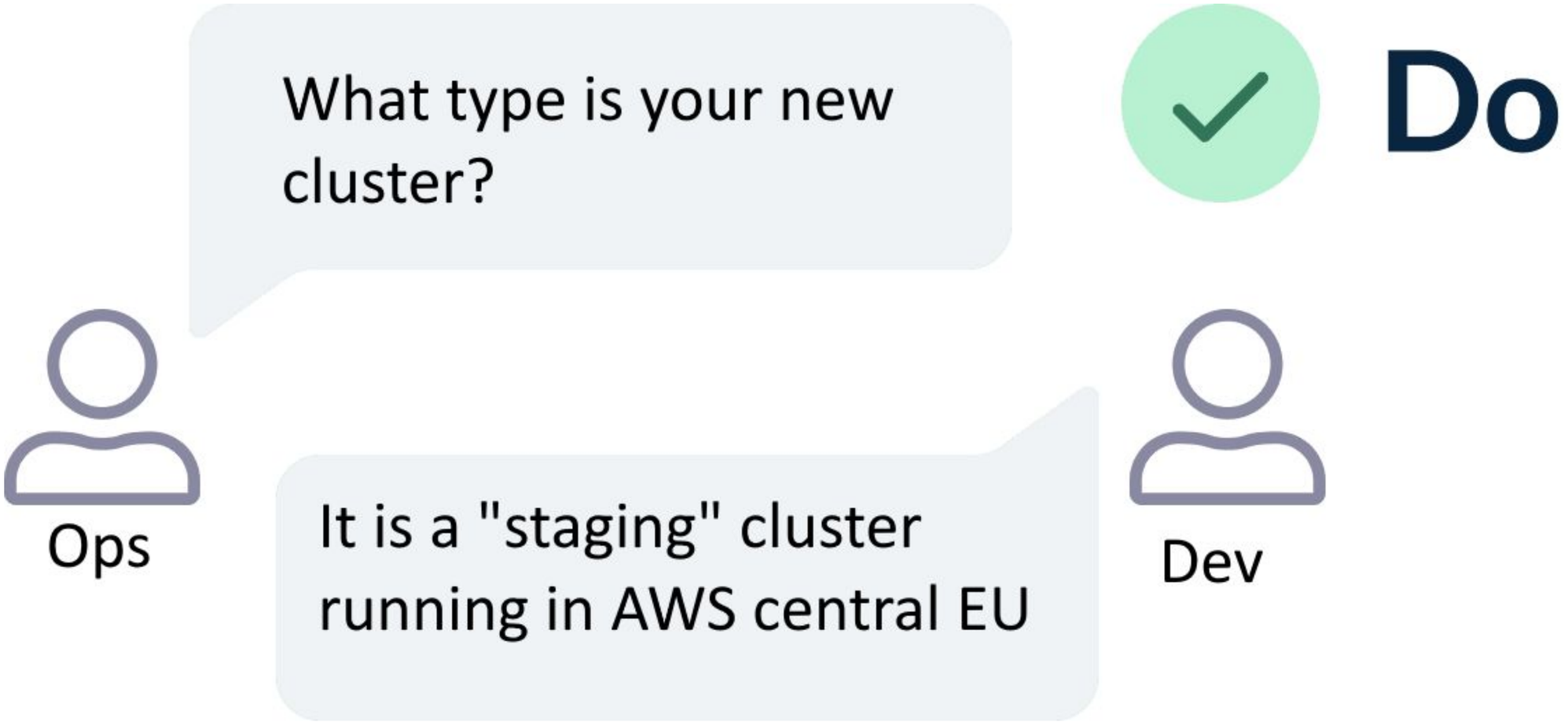
3. Argo CD will take care of everything else

Finished

Communication with developers



Communication with developers



BEST PRACTICES

Distribute Your Argo CD Applications to Different Kubernetes Clusters Using Application Sets

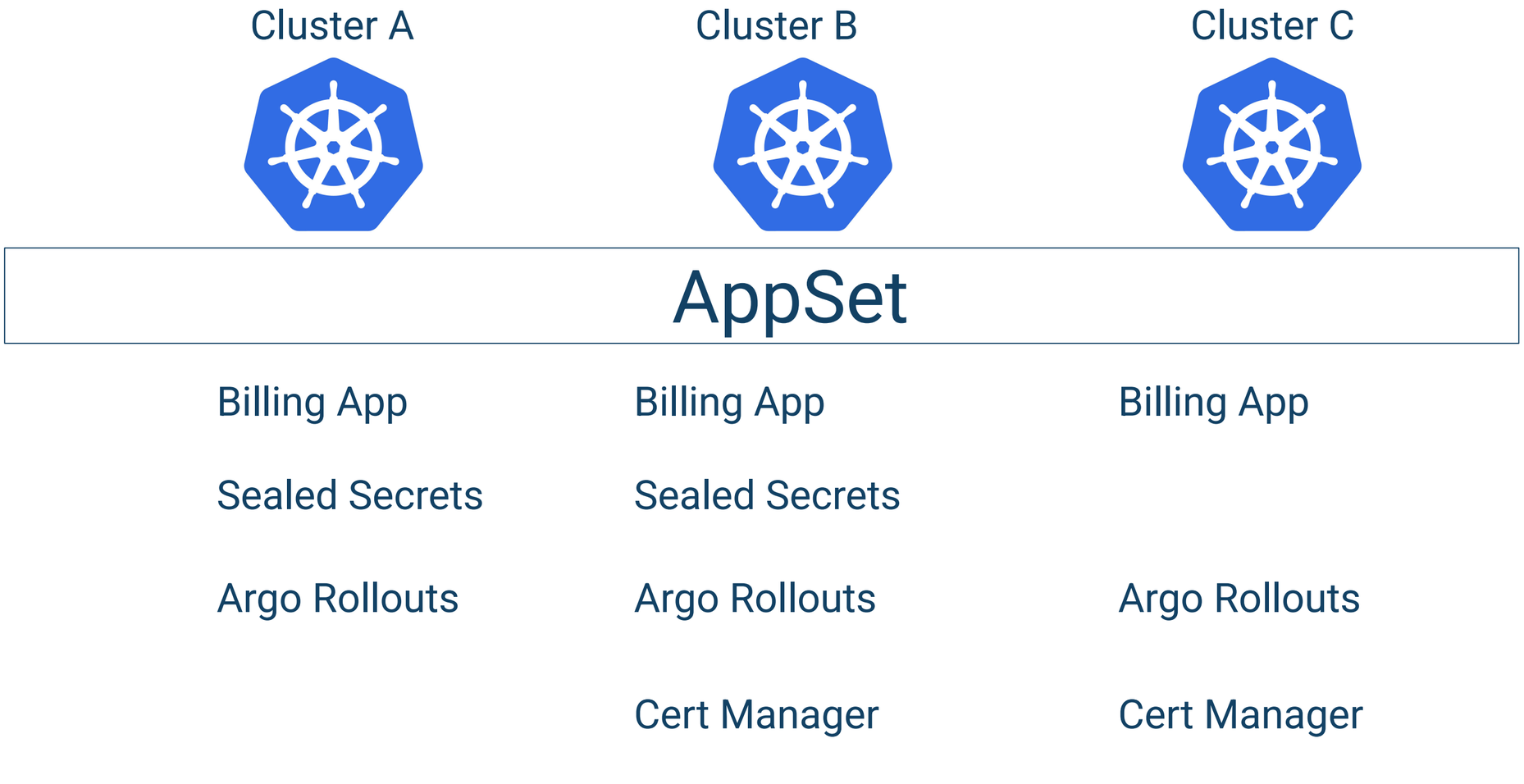
23 min read

https://codefresh.io/blog/argocd-clusters-labels-with-apps/

Create many Application Sets

Per team/environment/cloud/department etc.

Many Apps - Many Clusters



Use Tags with apps

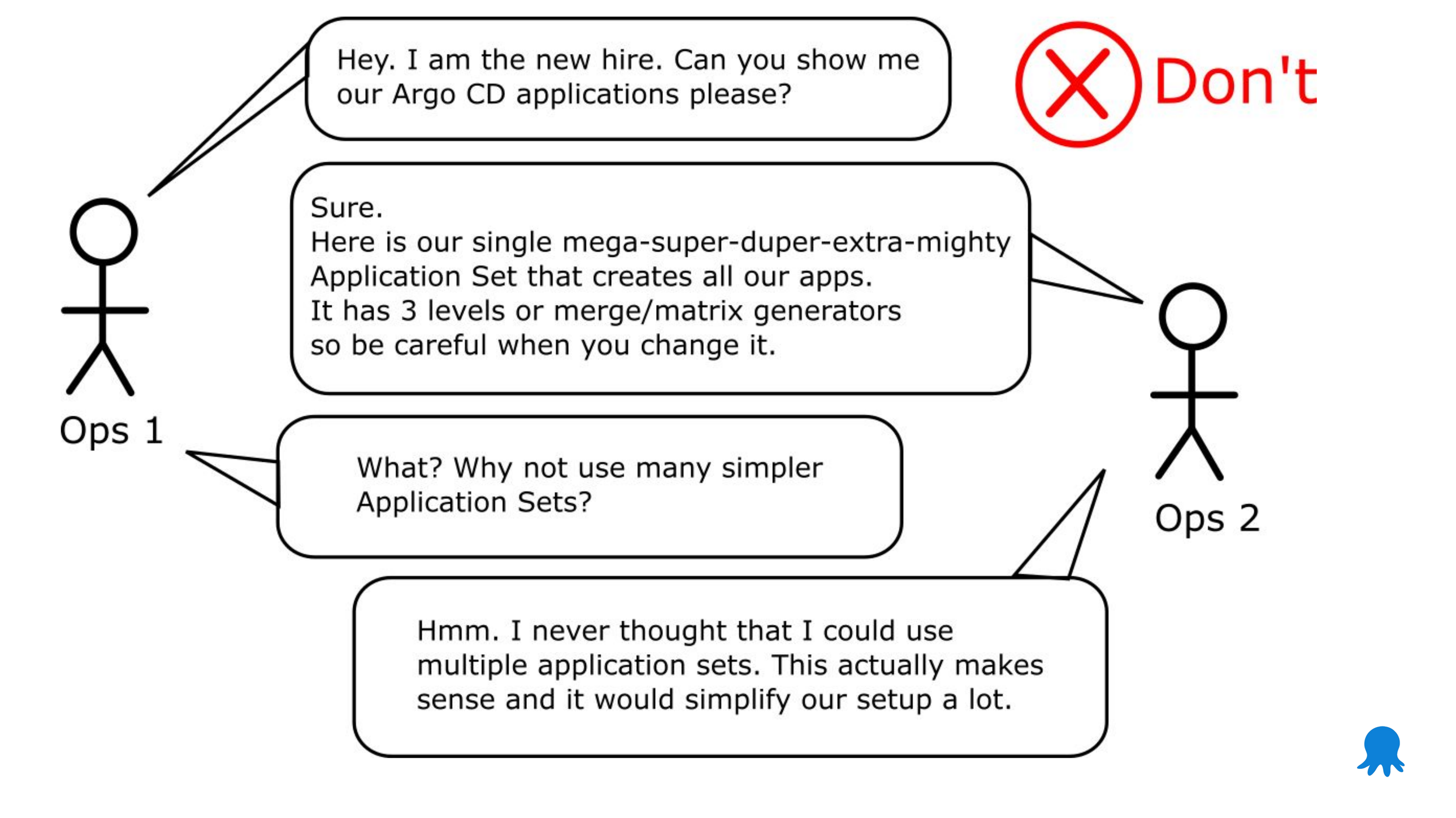
- “In all my AWS clusters I want these apps” -> Create application Set
- “In all my Prod clusters I want these apps “ -> Create Application Set
- “In all my US Region clusters I want these apps” -> Create Application Set
- “In all my Staging clusters I want these apps” -> Create Application Set

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: appset-{{ .Values.application.name }}
  namespace: argo
spec:
  goTemplate: true
  goTemplateOptions: ["missingkey=error"]
  generators:
    - list:
        elements:
          - cluster: 01
            url: https://qa-svc.privatelink.eastwest.azmk8s.io:443
  template:
    metadata:
      name: "{{ .Values.application.name }}-{{`{{ .cluster }}`}}"
    spec:
      destination:
        namespace: "{{ .Values.spec.namespace }}"
        server: "{{`{{ .url }}`}}"
      project: "{{ .Values.spec.project }}"
      syncPolicy:
        automated:
          selfHeal: true
      source:
        repoURL: "{{ .Values.spec.source.repoURL }}"
        targetRevision: "{{ .Values.spec.source.targetRevision }}"
        path: "{{ .Values.spec.source.chart }}"
        helm:
          parameters:
            - name: application.name
              value: "{{ .Values.application.name }}"
            {{- range $key1, $value1 := .Values.params }}
              {{- if typeIs "map[string]interface {}" $value1 }}
                {{- range $key2, $value2 := $value1 }}
                  {{- if typeIs "map[string]interface {}" $value2 }}
                    {{- range $key3, $value3 := $value2 }}
                      {{- if typeIs "map[string]interface {}" $value3 }}
                        {{- range $key4, $value4 := $value3 }}
                          {{- if typeIs "map[string]interface {}" $value4 }}
                            {{- range $key5, $value5 := $value4 }}
                              {{- if typeIs "map[string]interface {}" $value5 }}
                                {{- range $key6, $value6 := $value5 }}
            - name: {{ $key1 }}.{{ $key2 }}.{{ $key3 }}.{{ $key4 }}.{{ $key5 }}.{{ $key6 }}
              value: {{ $value6 | quote}}
                                {{- end }}
                              {{- else }}
            - name: {{ $key1 }}.{{ $key2 }}.{{ $key3 }}.{{ $key4 }}.{{ $key5 }}
              value: {{ $value5 | quote}}
                              {{- end }}
                            {{- end }}
                          {{- else }}
            - name: {{ $key1 }}.{{ $key2 }}.{{ $key3 }}.{{ $key4 }}
              value: {{ $value4 | quote}}
                          {{- end }}
                        {{- end }}
                      {{- else }}
            - name: {{ $key1 }}.{{ $key2 }}.{{ $key3 }}
              value: {{ $value3 | quote}}
                      {{- end }}
                    {{- end }}
                  {{- else }}
            - name: {{ $key1}}.{{ $key2 }}
              value: {{ $value2 | quote}}
                  {{- end }}
                {{- end }}
              {{- else }}
            - name: {{ $key1 }}
              value: {{ $value1 | quote}}
              {{- end }}
            {{- end }}
```

"Here is our single Application Set that controls all our apps and clusters"

REJECTED

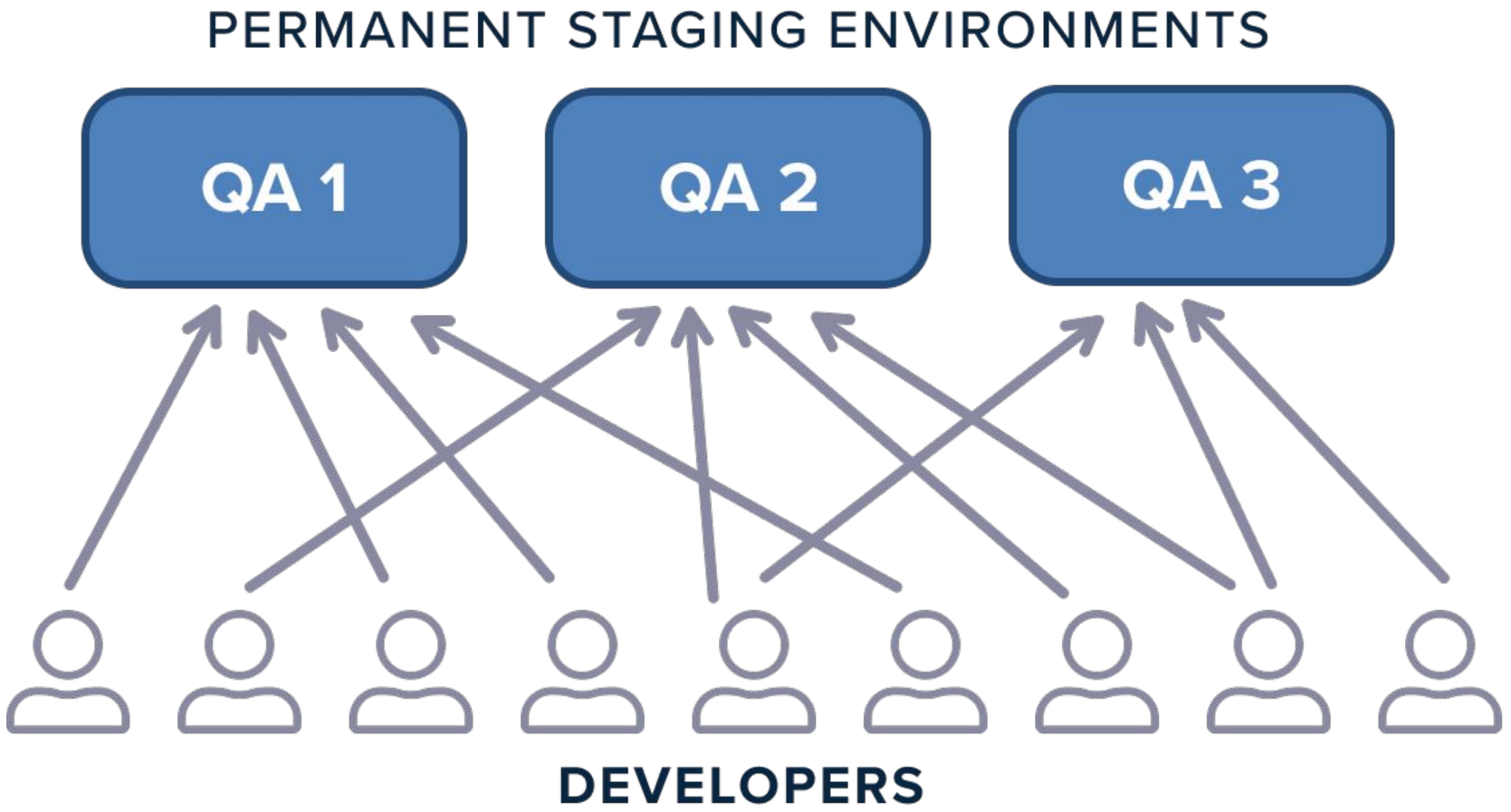
```yaml
      helm:
        parameters:
          - name: application.name
            value: "{{ .Values.application.name }}"
          {{- range $key1, $value1 := .Values.params }}
            {{- if typeIs "map[string]interface {}" $value1 }}
              {{- range $key2, $value2 := $value1 }}
                {{- if typeIs "map[string]interface {}" $value2 }}
                  {{- range $key3, $value3 := $value2 }}
                    {{- if typeIs "map[string]interface {}" $value3 }}
                      {{- range $key4, $value4 := $value3 }}
                        {{- if typeIs "map[string]interface {}" $value4 }}
                          {{- range $key5, $value5 := $value4 }}
                            {{- if typeIs "map[string]interface {}" $value5 }}
                              {{- range $key6, $value6 := $value5 }}
          - name: {{ $key1 }}.{{ $key2 }}.{{ $key3 }}.{{ $key4 }}.{{ $key5 }}.{{ $key6 }}
            value: {{ $value6 | quote}}
                              {{- end }}
                            {{- else }}
          - name: {{ $key1 }}.{{ $key2 }}.{{ $key3 }}.{{ $key4 }}.{{ $key5 }}
            value: {{ $value5 | quote}}
                            {{- end }}
                          {{- end }}
                        {{- else }}
          - name: {{ $key1 }}.{{ $key2 }}.{{ $key3 }}.{{ $key4 }}
            value: {{ $value4 | quote}}
                        {{- end }}
                      {{- end }}
                    {{- else }}
          - name: {{ $key1 }}.{{ $key2 }}.{{ $key3 }}
            value: {{ $value3 | quote}}
                    {{- end }}
                  {{- end }}
                {{- else }}
          - name: {{ $key1 }}.{{ $key2 }}
            value: {{ $value2 | quote}}
                {{- end }}
              {{- end }}
            {{- else }}
          - name: {{ $key1 }}
            value: {{ $value1 | quote}}
            {{- end }}
          {{- end }}
```


Hey. I am the new hire. Can you show me our Argo CD applications please?
Don't
Sure.
Here is our single mega-super-duper-extra-mighty Application Set that creates all our apps.
It has 3 levels or merge/matrix generators so be careful when you change it.
Ops 1
What? Why not use many simpler Application Sets?
Ops 2
Hmm. I never thought that I could use multiple application sets. This actually makes sense and it would simplify our setup a lot.

Understand the PR generator

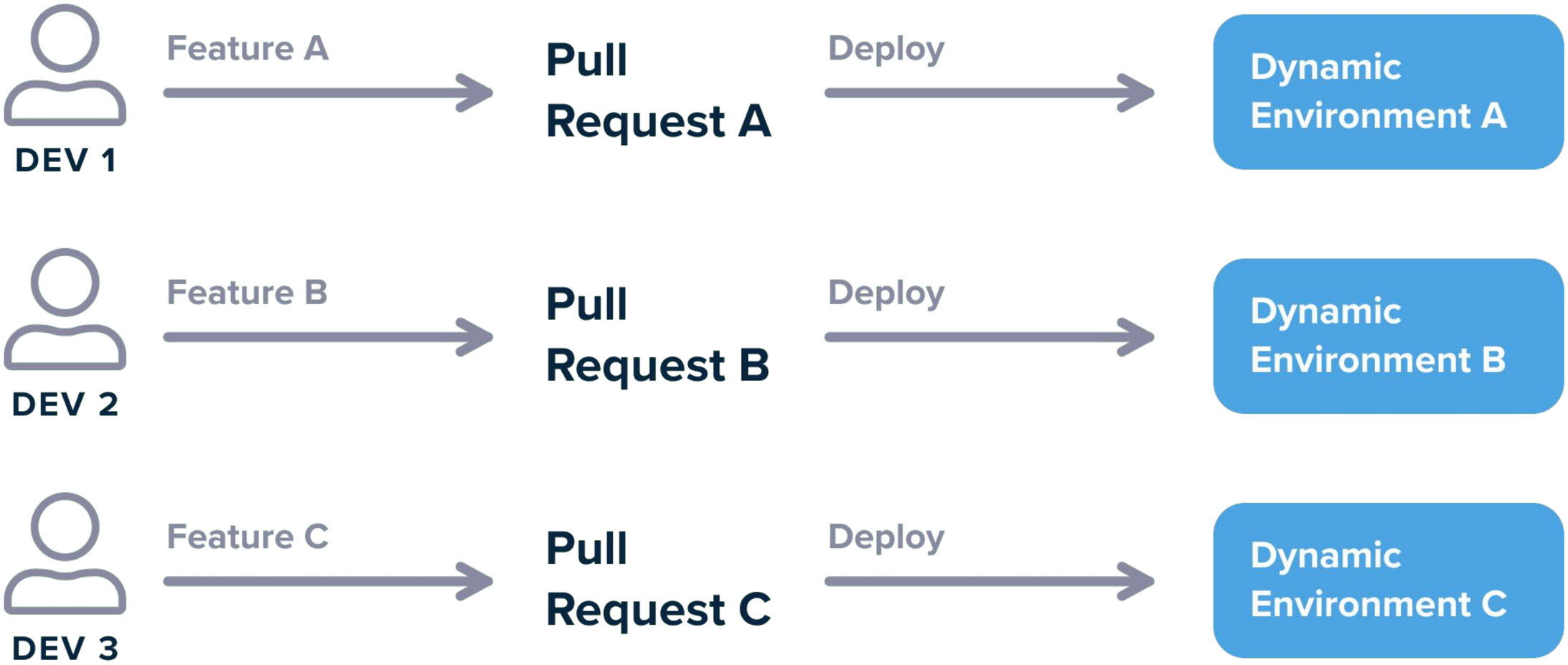
It is perfect for preview/emphemeral/temporary environments

The old way

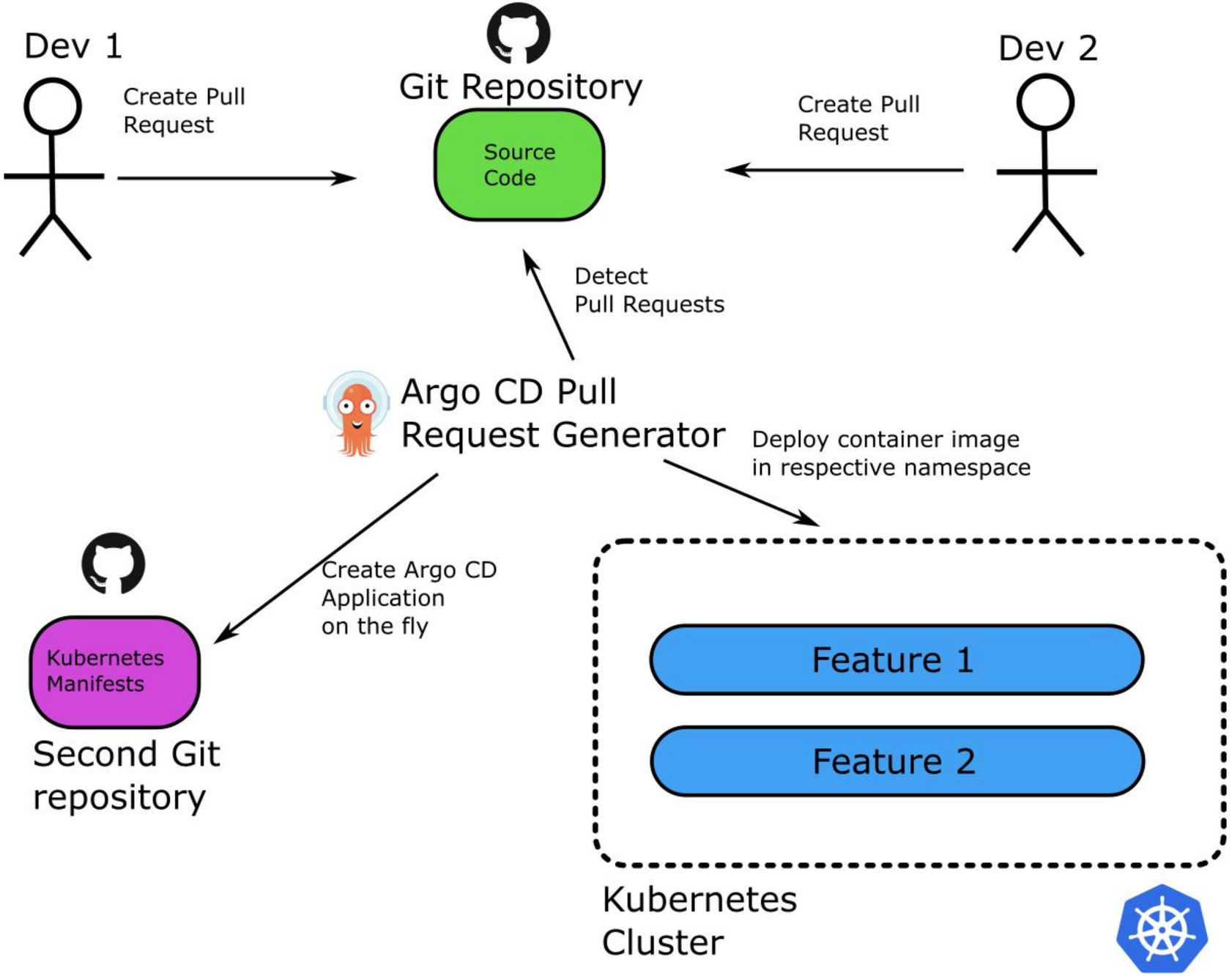


Don’t

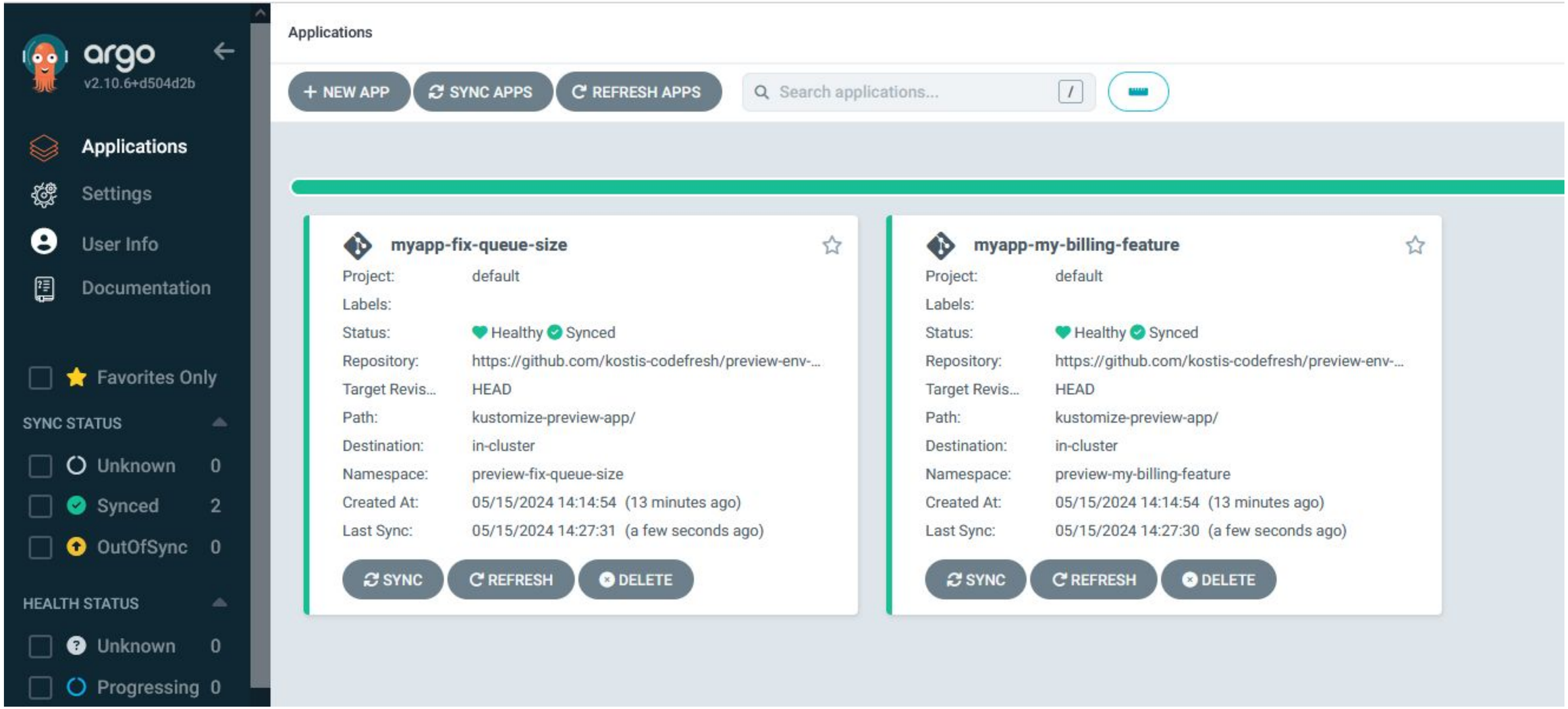
The correct way



Do Each feature is tested on its own

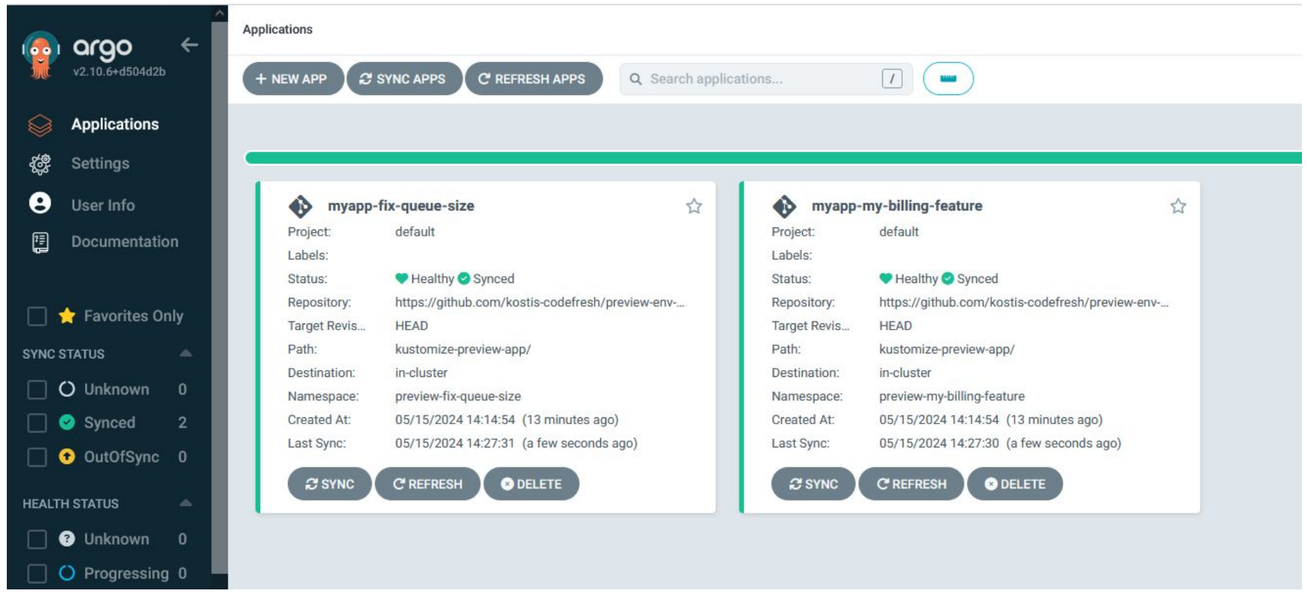

Dev 1
Create Pull Request
Git Repository
Source Code
Create Pull Request
Dev 2
Detect Pull Requests
Argo CD Pull Request Generator
Deploy container image in respective namespace
Create Argo CD Application on the fly
Kubernetes Manifests
Second Git repository
Feature 1
Feature 2
Kubernetes Cluster

Dynamic/temporary deployments



Dynamic/temporary deployments

- Create a PR request -> Application deployed
- Close PR -> Application removed
- Merge PR -> Application removed
- Commit again -> Application updated
- Works great with vCluster



TECHNICAL GUIDES

Creating Temporary Preview Environments Based On Pull Requests With Argo CD And Codefresh

11 min read

Kostis Kapelonis May 30, 2024

https://codefresh.io/blog/creating-temporary-preview-environments-based-pull-requests-argo-cd-codefresh/

Conclusion

1. Understand Application Sets
2. Group Application Sets with App-of-Apps files
3. Avoid the Helm sandwich
4. Employ Cluster groups with Tags
5. Create Many Application Sets
6. Use the PR Generator for preview envs

Thank you!

Questions: kostis.kapelonis@octopus.com

GitOps/Argo CD certification learning.octopus.com

CNCF Slack https://slack.cncf.io/

Blog https://blog.argoproj.io/

Octopus Deploy