



**KubeCon**



**CloudNativeCon**

**North America 2024**





KubeCon



CloudNativeCon

North America 2024

# Mastering ApplicationSet: Advanced Argo CD Automation

Alexander Matyushentsev



**Alexander Matyushentsev**

Argo CD Lead

Co-Founder and Chief Software Architect at *Akuity*

- Day 2 with Argo CD
- Solutions
  - API + CLI
  - App of Apps plus templating
  - ApplicationSet
- ApplicationSet Features
- Building real life ApplicationSet
  - Real life example requires advanced features
  - Getting it to work: debugging and finding errors
  - Testing changes after it runs in production

# Day 1 with Argo CD

The screenshot displays the Argo CD web interface in a browser window. The address bar shows 'myargocd.com'. The left sidebar contains navigation links: Applications, Settings, User Info, and Documentation. Below these are filters for 'Favorites Only', 'SYNC STATUS' (Unknown, Synced, OutOfSync), 'HEALTH STATUS' (Unknown, Progressing, Suspended, Healthy, Degraded, Missing), and 'LABELS'. The main content area is titled 'Applications' and shows a list of applications. The 'guestbook' application is selected, displaying its details in a modal window. The details include: Project: default, Labels: cluster=test, Status: Healthy and Synced, Repository: https://github.com/argoproj/argocd-example-apps, Target Revision: HEAD, Path: guestbook, Destination: test, Namespace: demo, Created At: 04/12/2024 10:36:48 (7 months ago), and Last Sync: 11/07/2024 11:31:04 (3 minutes ago). At the bottom of the modal are buttons for SYNC, REFRESH, and a status icon. The top right of the interface shows 'APPLICATIONS TILES' and a 'Log out' button.

Applications

APPLICATIONS TILES

+ NEW APP + SYNC APPS + REFRESH APPS

Search applications...

Previous 1 Next

Sort: name Items per page: 5

**guestbook**

Project: default

Labels: cluster=test

Status: Healthy Synced

Repository: https://github.com/argoproj/argocd-example-apps

Target Revision: HEAD

Path: guestbook

Destination: test

Namespace: demo

Created At: 04/12/2024 10:36:48 (7 months ago)

Last Sync: 11/07/2024 11:31:04 (3 minutes ago)

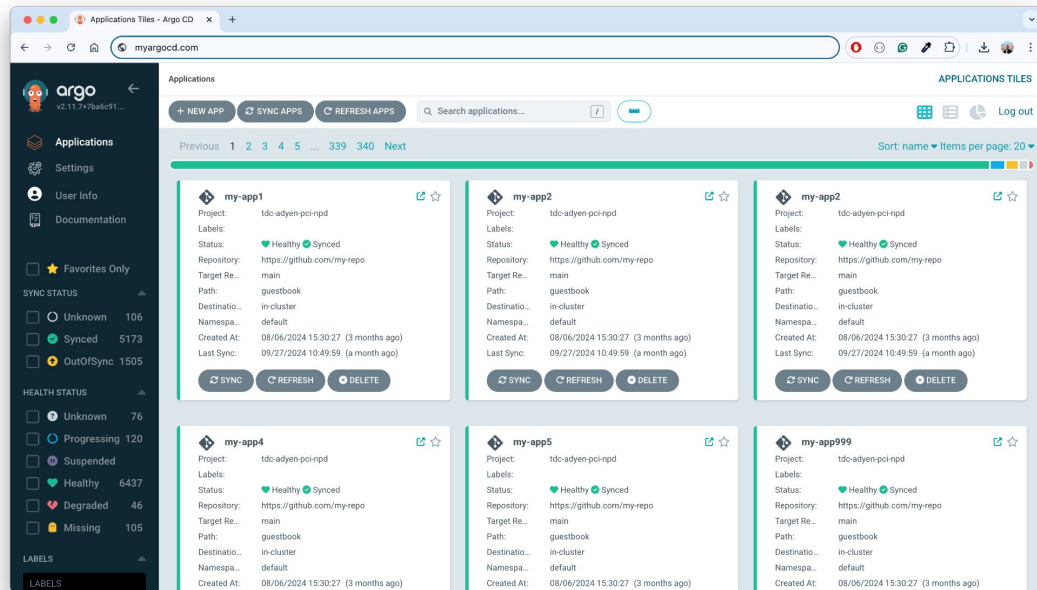
SYNC REFRESH

## Use Cases

**Environment applications** - applications deploying the same application into different environments (qa, prod, stage)

**Cluster addons** - homogeneous set of applications in each cluster

**Ephemeral environments** - applications deployed dynamically for testing/development purposes (e.g. for PR review purposes)



## Git based

✓ Yes - pipeline is stored in Git

## Declarative

✗ No - pipeline logic imperative

## Fully Automated

✓ Yes - new apps are created automatically as teams start using Argo CD

```
1  name: Promotion Workflow
2  on:
3    push:
4      branches:
5        - main
6
7  jobs:
8    qa-env:
9      name: Promote to QA
10     runs-on: ubuntu-latest
11     timeout-minutes: 10
12     steps:
13       - uses: actions/checkout@v4
14       - run: |
15           argocd app create my-service-qa \
16             --repo https://github.com/my-org/my-repo \
17             --path qa \
18             --dest-cluster qa-cluster \
19             --dest-namespace default \
20             # Idempotently update the app if it already exists
21             --upsert
```

# Solutions: App of Apps

## Git based

✓ Yes - pipeline is stored in Git

## Declarative

✓ Yes

## Fully Automated

✗ No - enrolling additional environments require manual changes

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Application
3  metadata:
4    name: my-service-qa
5  spec:
6    project: my-service
7    source:
8      repoURL: https://github.com/my-org/my-service.git
9      targetRevision: HEAD
10     path: qa
11   destination:
12     name: qa-cluster
13     namespace: my-service-qa
14   ---
15  apiVersion: argoproj.io/v1alpha1
16  kind: Application
17  metadata:
18    name: my-service-stage
19  spec:
20    project: my-service
21    source:
22      repoURL: https://github.com/my-org/my-service.git
23      targetRevision: HEAD
24      path: stage
25    destination:
26      name: stage-cluster
27      namespace: my-service-stage
```



# Solutions: App of Apps + Templating

## Git based



Yes - pipeline is stored in Git

## Declarative



Yes

## Fully Automated



Still no - enrolling additional environments require manual changes of values.yaml

### templates/application.yaml

```
1  {{- range $i, $value := .Values.apps }}
2  apiVersion: argoproj.io/v1alpha1
3  kind: Application
4  metadata:
5    name: my-service-{{ $value.name }}
6  spec:
7    project: my-service
8    source:
9      repoURL: https://github.com/my-org/my-service.git
10     targetRevision: HEAD
11     path: {{ $value.name }}
12     destination:
13       name: qa-cluster
14       namespace: my-service-{{ $value.name }}
15  {{- end }}
```

### values.yaml

```
1  apps:
2    - name: qa
3    - name: stage
4    - name: prod
```

## Git based

✓ Yes - pipeline is stored in Git

## Declarative

✓ Yes - pipeline logic imperative

## Fully Automated

✓ Yes - new apps are created automatically

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: ApplicationSet
3  metadata:
4    name: cluster-addons
5  spec:
6    goTemplate: true
7    goTemplateOptions: ["missingkey=error"]
8    generators:
9      - git:
10        repoURL: https://github.com/my-org/my-service.git
11        revision: HEAD
12        directories:
13          - path: envs/*
14      template:
15        metadata:
16          name: 'my-service-{{.path.basename}}'
17        spec:
18          project: my-service
19          source:
20            repoURL: https://github.com/my-org/my-service.git
21            targetRevision: HEAD
22            path: '{{.path.path}}'
23          destination:
24            server: '{{.path.basename}}-cluster'
25            namespace: 'my-service{{.path.basename}}'
```

## Template

An Argo CD application spec  
Go-based template.

## Generator

Use case-specific producer  
of values that are fed into  
the template

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: ApplicationSet
3  metadata:
4    name: guestbook
5  spec:
6    goTemplate: true
7    goTemplateOptions: ["missingkey=error"]
8    generators:
9      - list:
10        elements:
11          - cluster: engineering-dev
12            url: https://1.2.3.4
13      template:
14        metadata:
15          name: '{{.cluster}}-guestbook'
16        spec:
17          project: my-project
18          source:
19            repoURL: https://github.com/infra-team/cluster-deployments.git
20            targetRevision: HEAD
21            path: guestbook/{{.cluster}}
22          destination:
23            server: '{{.url}}'
24            namespace: guestbook
```

generator

template

## Cluster

Produces parameters based on the list of registered Argo CD clusters

## Use Case:

Cluster addons - automatically produce apps for each new cluster

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: ApplicationSet
3  metadata:
4    | name: guestbook
5  spec:
6    | generators:
7    | - clusters:
8    |   | selector:
9    |   |   matchLabels:
10   |   |   staging: "true"
```

## Git

Generate parameters based on Git repository content

## Use Case:

Application developer teams  
self-servicing - automatically produces  
apps for each new directory/file in Git

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: ApplicationSet
3  metadata:
4    | name: guestbook
5  spec:
6    | generators:
7    | - git:
8    |   | repoURL: https://github.com/my-org/my-repo
9    |   | revision: HEAD
10   |   | directories:
11   |   | - path: cluster-addons/*
```

## List

Generates parameters based on an arbitrary list of key/value pairs

## Use Case

One off customizations

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: ApplicationSet
3  metadata:
4    name: guestbook
5  spec:
6    generators:
7      - list:
8          elements:
9            - cluster: engineering-dev
10              url: https://1.2.3.4
11              - cluster: engineering-prod
12                url: https://5.6.7.8
```

## Merge and Matrix

Allows to combine various generators together

## Use Case

Enables complex, real-life requirements based on multiple input sources

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: ApplicationSet
3  metadata:
4    name: guestbook
5  spec:
6    generators:
7      - merge:
8        mergeKeys:
9          - my-key
10         generators:
11           - # generator1
12           - # generator2
```

## Challenges/Community Feedback

- Real ApplicationSet is a complex program
- Hard to troubleshoot errors
  - Lack of visibility
  - Long retry cycle
- Difficult to make changes in production
  - Any mistake affects critical applications
- Difficult to get logic right
  - Don't produce apps at all
  - Produces wrong set of apps

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: ApplicationSet
3  metadata:
4    name: cluster-apps
5  spec:
6    generators:
7      - merge:
8        mergeKeys:
9          - metadata.labels.env
10         - path.basename
11        generators:
12          - matrix:
13            generators:
14              - clusters: {}
15            - git:
16              repoURL: &repo https://github.com/alexmt/kubecon-2024-us.git
17              directories:
18                - path: clusters/base/*
19          - merge:
20            mergeKeys:
21              - path
22            generators:
23              - git:
24                repoURL: *repo
25                revision: HEAD
26                files:
27                  - path: clusters/groups/*.env.yaml
28              - git:
29                repoURL: *repo
30                revision: HEAD
31                directories:
32                  - path: clusters/groups/*/*
```



# Which Solution Is Best For You?

- **CLI/API Based Automation**
  - Very flexible
  - Powered by a programming language and support any edge case
  - Great troubleshooting toolset
  - **Unfortunately will require some work to get it done**
- **App Of Apps**
  - Very simple and fully declarative
  - Does not require troubleshooting
  - **Does not provide the best end users experience**
- **ApplicationSet**
  - Fully declarative
  - Support 80% of use cases without much work
  - **Will require some work to support remaining 20%**

## Check error message in ApplicationSet conditions

*kubectl get appset cluster-apps -o=yaml*

```
status:
  conditions:
  - lastTransitionTime: "2024-11-08T23:11:04Z"
    message: 'error getting param sets from generators: error getting params from
      generator 1 of 2: child generator returned an error on parameter generation:
      failed to get params for second generator in the matrix generator: child generator
      returned an error on parameter generation: error getting project system: AppProject.argoproj.io
      "system" not found'
    reason: ApplicationGenerationFromParamsError
    status: "True"
    type: ErrorOccurred
```

## Get more details from argocd-applicationset-controller logs

*kubectl logs deploy/argocd-applicationset-controller*

*time="2024-11-08T16:42:58-07:00" level=error msg="error generating application from params" applicationset=argocd/cluster-apps error="error getting param sets from generators: error getting params from generator 1 of 2: child generator returned an error on parameter generation: failed to get params for second generator in the matrix generator: **child generator returned an error on parameter generation: error getting project system: AppProject.argoproj.io \"system\" not found**"*

## The `--dry-run` flag in `argocd appset create` CLI

- Provides basic appset spec validation
- Prints errors if app generation fails

```
$ argocd appset create ~/applicationset.yaml --upsert --dry-run  
FATA[0000] rpc error: code = InvalidArgument desc =  
ApplicationSet references project system which does not exist
```

## The `argocd appset generate` command

- **Validates application set and generate apps**
- **Returns generated applications and enables full application set testing**

```
$ argocd appset generate ./applicationset.yaml
```

```
traefik https://kubernetes.default.svc argocd default Auto-Prune  
<none> https://github.com/alexmt/kubecon-2024-us.git clusters/base/traefik  
grafana https://kubernetes.default.svc argocd default Auto-Prune  
<none> https://github.com/alexmt/kubecon-2024-us.git clusters/base/grafana
```

# Expected In Upcoming Releases

- An ability to test against local Git repositories
- An ability to create unit tests
- Your idea if you create an issue for it!

<https://github.com/argoproj/argo-cd/issues/new>

