
KubeCon
CloudNativeCon
North America 2024

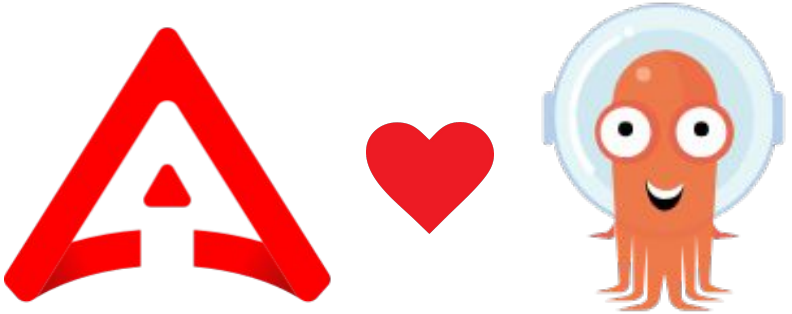
KubeCon | CloudNativeCon

North America 2024

Mastering ApplicationSet: Advanced Argo CD Automation

Alexander Matyushentsev

Introductions



Alexander Matyushentsev

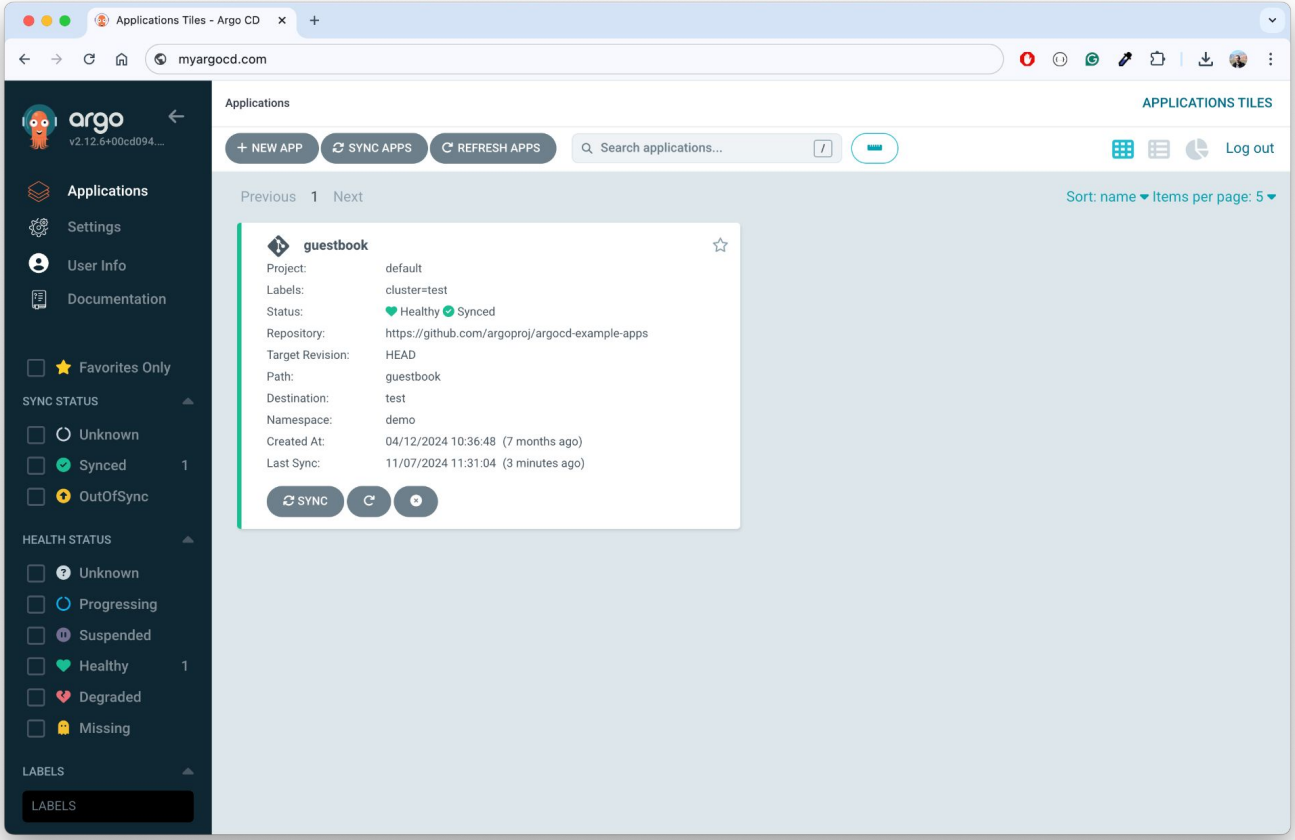
Argo CD Lead

Co-Founder and Chief Software Architect at *Akuity*

Agenda

- 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

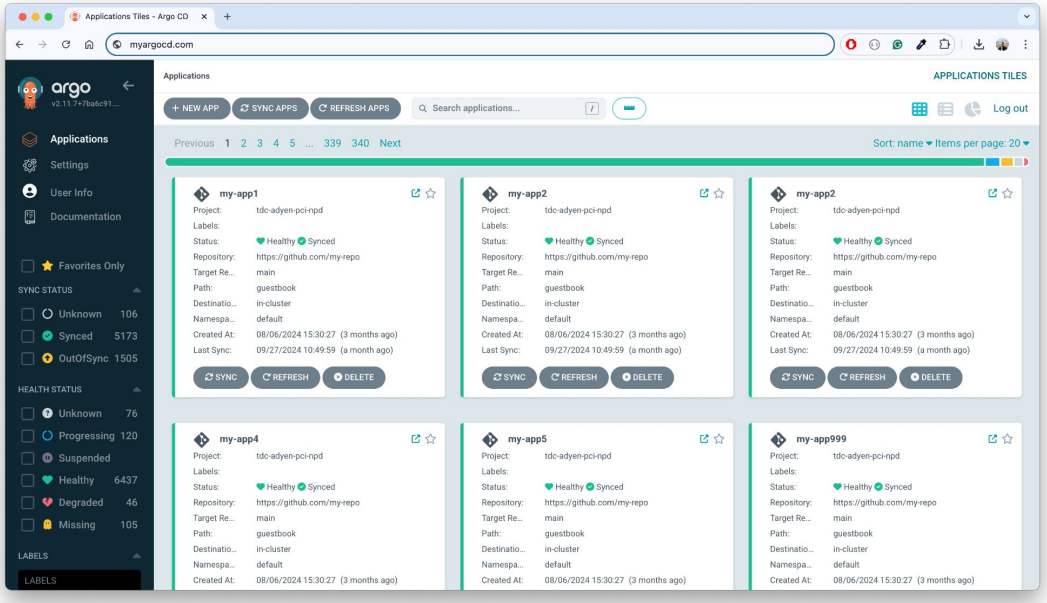


Use Cases

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)



Solutions: CLI/API

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

```yaml
name: Promotion Workflow
on:
  push:
    branches:
      - main

jobs:
  qa-env:
    name: Promote to QA
    runs-on: ubuntu-latest
    timeout-minutes: 10
    steps:
      - uses: actions/checkout@v4
      - run: |
          argocd app create my-service-qa \
          --repo https://github.com/my-org/my-repo \
          --path qa \
          --dest-cluster qa-cluster \
          --dest-namespace default \
          # Idempotently update the app if it already exists
          --upsert
```

Solutions: App of Apps

Git based
✅ Yes - pipeline is stored in Git

Declarative
✅ Yes

Fully Automated
❌ No - enrolling additional environments require manual changes

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-service-qa
spec:
  project: my-service
  source:
    repoURL: https://github.com/my-org/my-service.git
    targetRevision: HEAD
    path: qa
  destination:
    name: qa-cluster
    namespace: my-service-qa
---
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-service-stage
spec:
  project: my-service
  source:
    repoURL: https://github.com/my-org/my-service.git
    targetRevision: HEAD
    path: stage
  destination:
    name: stage-cluster
    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

```
{{- range $i, $value := .Values.apps }}
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-service-{{ $value.name }}
spec:
  project: my-service
  source:
    repoURL: https://github.com/my-org/my-service.git
    targetRevision: HEAD
    path: {{ $value.name }}
  destination:
    name: qa-cluster
    namespace: my-service-{{ $value.name }}
{{- end }}
```

values.yaml

```
apps:
  - name: qa
  - name: stage
  - name: prod
```

Solutions: ApplicationSet

Git based
✅ Yes - pipeline is stored in Git

Declarative
✅ Yes - pipeline logic imperative

Fully Automated
✅ Yes - new apps are created automatically

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: cluster-addons
spec:
  goTemplate: true
  goTemplateOptions: ["missingkey=error"]
  generators:
  - git:
      repoURL: https://github.com/my-org/my-service.git
      revision: HEAD
      directories:
      - path: envs/*
  template:
    metadata:
      name: 'my-service-{{.path.basename}}'
    spec:
      project: my-service
      source:
        repoURL: https://github.com/my-org/my-service.git
        targetRevision: HEAD
        path: '{{.path.path}}'
      destination:
        server: '{{.path.basename}}-cluster'
        namespace: 'my-service{{.path.basename}}'
```

ApplicationSet Features

Template
An Argo CD application spec Go-based template.

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

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  goTemplate: true
  goTemplateOptions: ["missingkey=error"]
  generators:
  - list:
      elements:
      - cluster: engineering-dev
        url: https://1.2.3.4
  template:
    metadata:
      name: '{{.cluster}}-guestbook'
    spec:
      project: my-project
      source:
        repoURL: https://github.com/infra-team/cluster-deployments.git
        targetRevision: HEAD
        path: guestbook/{{.cluster}}
      destination:
        server: '{{.url}}'
        namespace: guestbook
```

generator

template

ApplicationSet Generators

Cluster
Produces parameters based on the list of registered Argo CD clusters

Use Case:
Cluster addons - automatically produce apps for each new cluster

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - clusters:
      selector:
        matchLabels:
          staging: "true"
```

ApplicationSet Generators

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

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - git:
      repoURL: https://github.com/my-org/my-repo
      revision: HEAD
      directories:
      - path: cluster-addons/*
```

ApplicationSet Generators

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

Use Case
One off customizations

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - list:
      elements:
      - cluster: engineering-dev
        url: https://1.2.3.4
      - cluster: engineering-prod
        url: https://5.6.7.8
```

ApplicationSet Generators

Merge and Matrix
Allows to combine various generators together

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

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - merge:
      mergeKeys:
        - my-key
      generators:
        - # generator1
        - # generator2
```

Real Life ApplicationSet

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

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: cluster-apps
spec:
  generators:
  - merge:
      mergeKeys:
        - metadata.labels.env
        - path.basename
      generators:
      - matrix:
          generators:
          - clusters: {}
          - git:
              repoURL: &repo https://github.com/alexmt/kubecon-2024-us.git
              directories:
              - path: clusters/base/*
      - merge:
          mergeKeys:
            - path
          generators:
          - git:
              repoURL: *repo
              revision: HEAD
              files:
              - path: clusters/groups/*/.env.yaml
          - git:
              repoURL: *repo
              revision: HEAD
              directories:
              - 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%**

Troubleshooting Tips

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
```

Troubleshooting Tips

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"***

New In Argo CD v2.13!

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
```

New In Argo CD v2.13!

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