RESILIENCE

REALIZED

KubeCon | CloudNativeCon

North America 2021

KubeCon | CloudNativeCon

North America 2021

RESILIENCE REALIZED

Manage More Clusters with Less Hassle, with Argo CD Application Sets

Jonathan West

Kshama Jain

Speakers

Jonathan West, Red Hat
Twitter: @JonathanGWest
GitHub: @JGWest

Kshama Jain
Twitter: @kshamajain99
Github: @kshamajain99

ApplicationSets: The Elevator Pitch

ApplicationSets are a new Kubernetes custom resource (CR) and controller, which work alongside an existing Argo CD install.

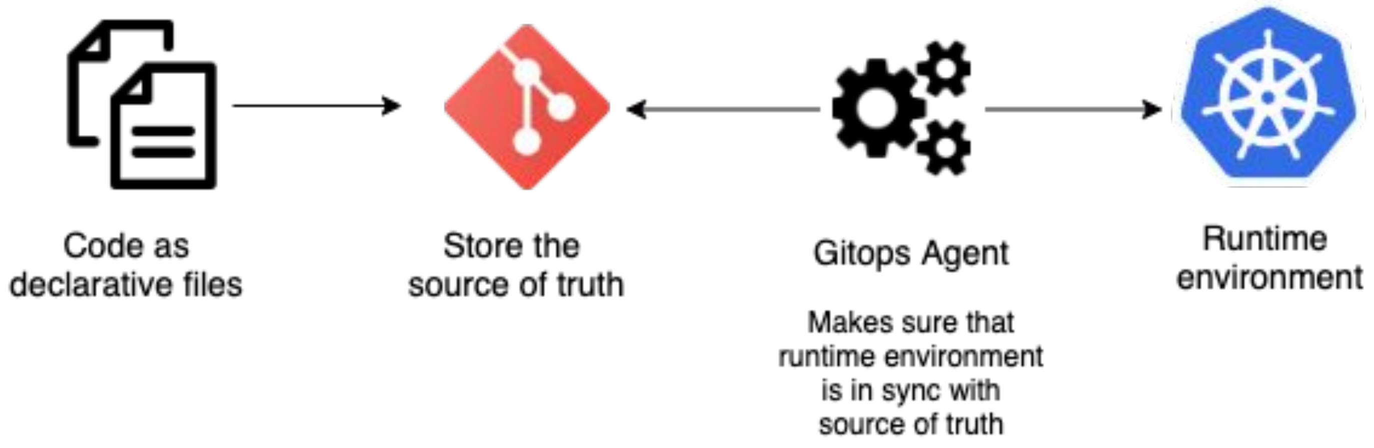
ApplicationSets are a factory for Argo CD Applications: the **ApplicationSet** CR describes the Applications to create/manage, and Argo CD is responsible for deploying them.

Features:

- Manage a large number of Argo CD Applications as a single unit.
- Your cluster deployments may be automated and customized from a variety of data sources:
  - For example, scaling up as new clusters added to your infrastructure, new Git repo are added, new repos are added to GitHub/GitLab org, and more.
- Automatically react to external changes
  - Reconciliation based on external events are quick and customizable: Applications are created/updated/deleted on the fly.
- ApplicationSets are based on Argo CD Applications, harnessing all the existing power that comes with Argo CD

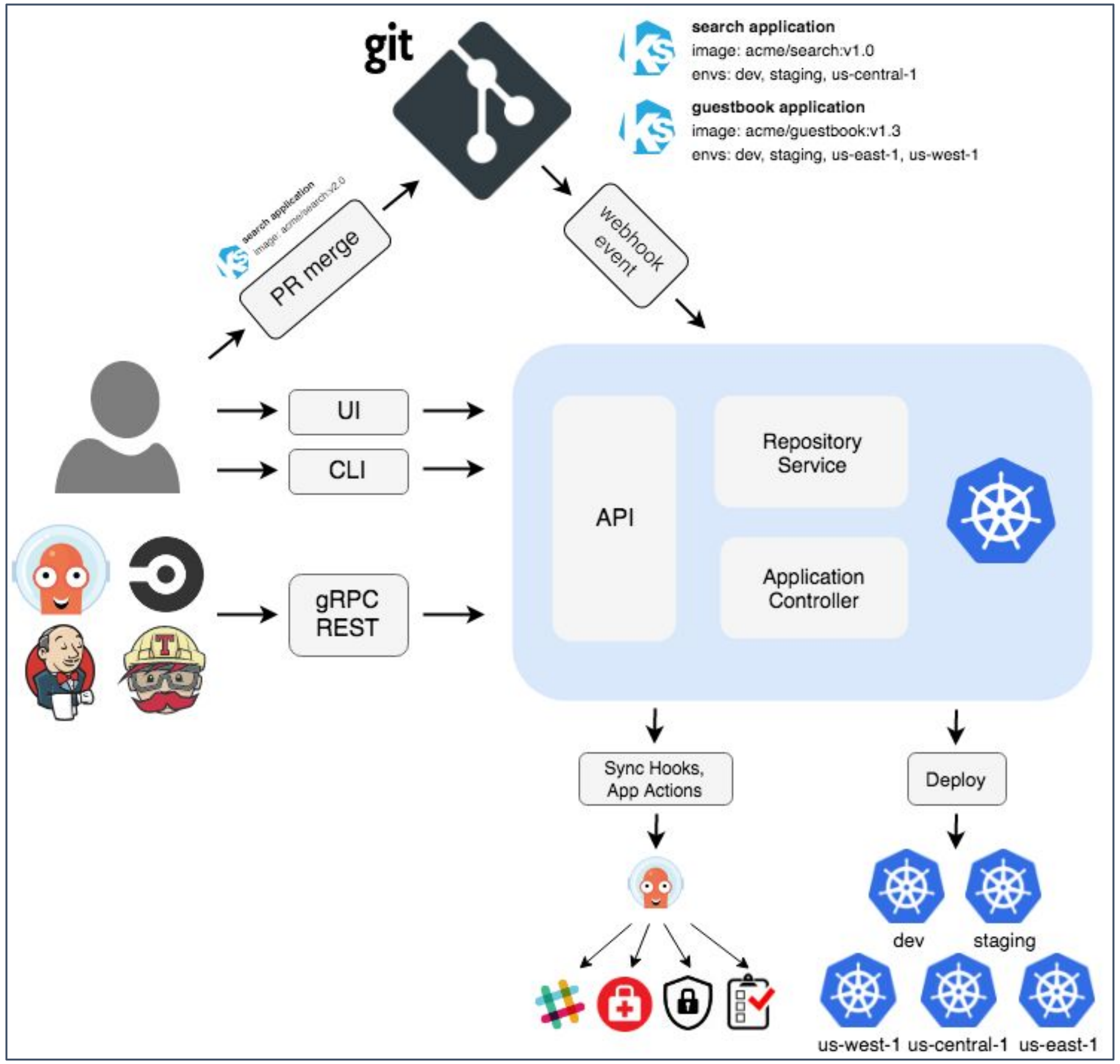
Introduction to GitOps

- Git is source of truth
- The desired software state is stored in form of files in a Git repository
- An gitops agent will make sure that desired software state is deployed on the runtime environment


Code as declarative files
Store the source of truth
Gitops Agent
Makes sure that runtime environment is in sync with source of truth
Runtime environment

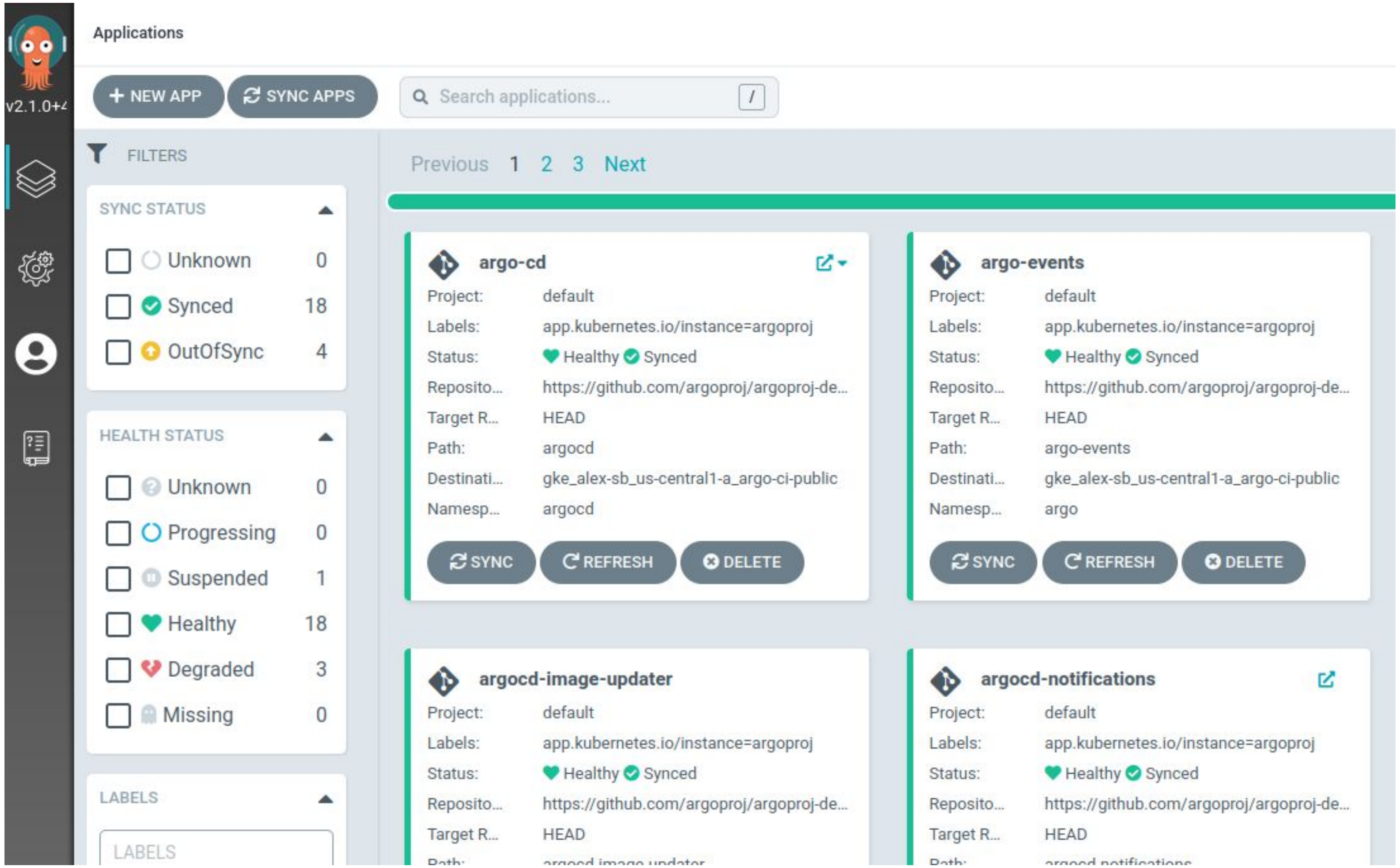
Introduction to Argo CD

- GitOps based continuous delivery tool for k8s.
- Uses Git repositories as source of truth.
- Declarative continuous delivery tool.
- Supports various configuration management tools (ksonnet/jsonnet, kustomize, helm).
- Implemented as a k8s controller and CRD.
- Provides a Kubernetes dashboard with powerful set of resource management features.
- Automates the synchronization of the desired application state with the current live state.



Argo CD Demo

KubeCon | CloudNativeCon North America 2021



v2.1.0+4

Applications

+ NEW APP | SYNC APPS | Search applications...

FILTERS

SYNC STATUS

Unknown	0
Synced	18
OutOfSync	4

HEALTH STATUS

Unknown	0
Progressing	0
Suspended	1
Healthy	18
Degraded	3
Missing	0

LABELS

Previous 1 2 3 Next

argo-cd

Project:	default
Labels:	app.kubernetes.io/instance=argoproj
Status:	Healthy Synced
Reposito...	https://github.com/argoproj/argoproj-de...
Target R...	HEAD
Path:	argocd
Destinati...	gke_alex-sb_us-central1-a_argo-ci-public
Namesp...	argocd

SYNC | REFRESH | DELETE

argo-events

Project:	default
Labels:	app.kubernetes.io/instance=argoproj
Status:	Healthy Synced
Reposito...	https://github.com/argoproj/argoproj-de...
Target R...	HEAD
Path:	argo-events
Destinati...	gke_alex-sb_us-central1-a_argo-ci-public
Namesp...	argo

SYNC | REFRESH | DELETE

argocd-image-updater

Project:	default
Labels:	app.kubernetes.io/instance=argoproj
Status:	Healthy Synced
Reposito...	https://github.com/argoproj/argoproj-de...
Target R...	HEAD

argocd-notifications

Project:	default
Labels:	app.kubernetes.io/instance=argoproj
Status:	Healthy Synced
Reposito...	https://github.com/argoproj/argoproj-de...
Target R...	HEAD

Argo CD Key Features

- Powerful, real-time web UI
- Support for multiple config management/templating tools
- Multi-cluster support
- SSO Integration
- Resource health assessments
- Automated configuration drift detection and diffing
- Automated or manual syncing of applications to its desired state
- CLI for automation and CI integration
- Audit trails of activity
- Prometheus metrics
- GnuPG signature verification
- Pre/Post sync hooks
- Multi-tenancy and RBAC policies for authorization

And many more ...

Argo CD Application Custom Resource

The central entity which defines what one can do in Argo CD is the **Application** CR.

```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
```

Argo CD Applications are like the 'glue' between your Kubernetes cluster and Git:
"Take these K8s resources defined in Git, and apply them to my cluster. Keep them in sync."

But notice there's only a single source (Git repository), and only a single destination (a single namespace of a single cluster).

Introduction to ApplicationSets

Unlike Argo CD Applications, which only connect a single Git repository path to a single Kubernetes cluster namespace, ApplicationSets allow you to define many such connections (as Argo CD Applications), and manage them all as a single unit.

The ApplicationSet controller is open source, and is an Argo CD sub-project, hosted within the *argoproj-labs* organization, and is built from contributions from many different folks, including individual contributions from folks from Red Hat, Snyk, Intuit, MLB, Ali Baba Cloud, and more.

The ApplicationSet controller requires an existing Argo CD installation, and works alongside it.

ApplicationSet Custom Resource (CR)

KubeCon CloudNativeCon North America 2021

The **generator** field references specifies a *generator,* which is responsible for generating template **parameters**.

template parameters are **key**-value pairs that will be substituted into the template:

- **cluster**: engineering-dev
- **url**: https://kubernetes.default.svc

The parameters are rendered into the corresponding **{{ parameter name }}** fields of the template.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - list:
      elements:
      - cluster: engineering-dev
        url: https://kubernetes.default.svc
  template:
    metadata:
      name: '{{cluster}}-guestbook'
    spec:
      project: default
      source:
        repoURL: https://github.com/argoproj-labs/applicationset.git
        targetRevision: HEAD
        path: examples/list-generator/guestbook/{{cluster}}
      destination:
        server: '{{url}}'
        namespace: guestbook
```

List Generator

What: A simple list of key-values, that are substituted directly into the template, producing Applications.

Why: The simplest generator. Good for initial experimentation with ApplicationSets and very basic deployment scenarios.

Less YAML than the equivalent Argo CD Applications, but still less powerful than other generators.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - list:
      elements:
      - cluster: engineering-dev
        url: https://kubernetes.default.svc
      - cluster: engineering-prod
        url: https://kubernetes.default.svc
  template:
    metadata:
      name: '{{cluster}}-guestbook'
    spec:
      project: default
      source:
        repoURL: https://github.com/argoproj-labs/applicationset.git
        targetRevision: HEAD
        path: examples/list-generator/guestbook/{{cluster}}
      destination:
        server: '{{url}}'
        namespace: guestbook
```

Generators

Generators are responsible for generating parameters, which are then rendered into the template: fields of the ApplicationSet resource.

Generators are primarily based on the data source that they use to generate the template parameters:

- the *List* generator provides a set of parameters from a literal list
- the *Cluster* generator uses the Argo CD cluster list as a source
- the *Git* generator uses files/directories from a Git repository
- ... and more.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - list:
      elements:
      - cluster: engineering-dev
        url: https://kubernetes.default.svc
      - cluster: engineering-prod
        url: https://kubernetes.default.svc
  template:
    # (...)
```

The generator generates two pairs of parameters

cluster	url
engineering-dev	https://kubernetes.default.svc
engineering-prod	https://kubernetes.default.svc

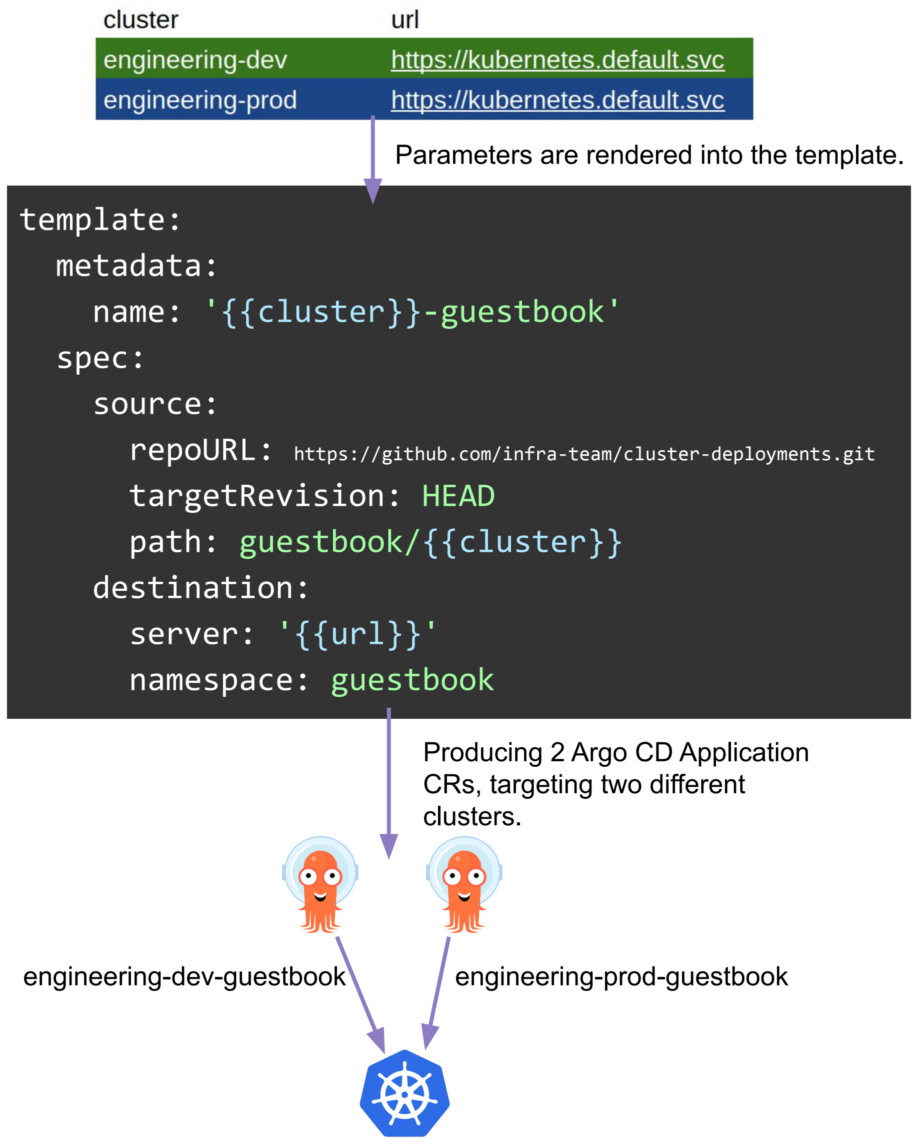
Parameters are substituted into the template.

Template

The template fields of the ApplicationSet spec are used to generate Argo CD Application resources.

An Argo CD Application is created by combining the parameters from the generator with fields of the template (via **{{values}}**), and from that a concrete Application resource is produced and applied to the cluster.

The template subfields correspond directly to the spec of an [Argo CD Application resource](#) ([example](#)).

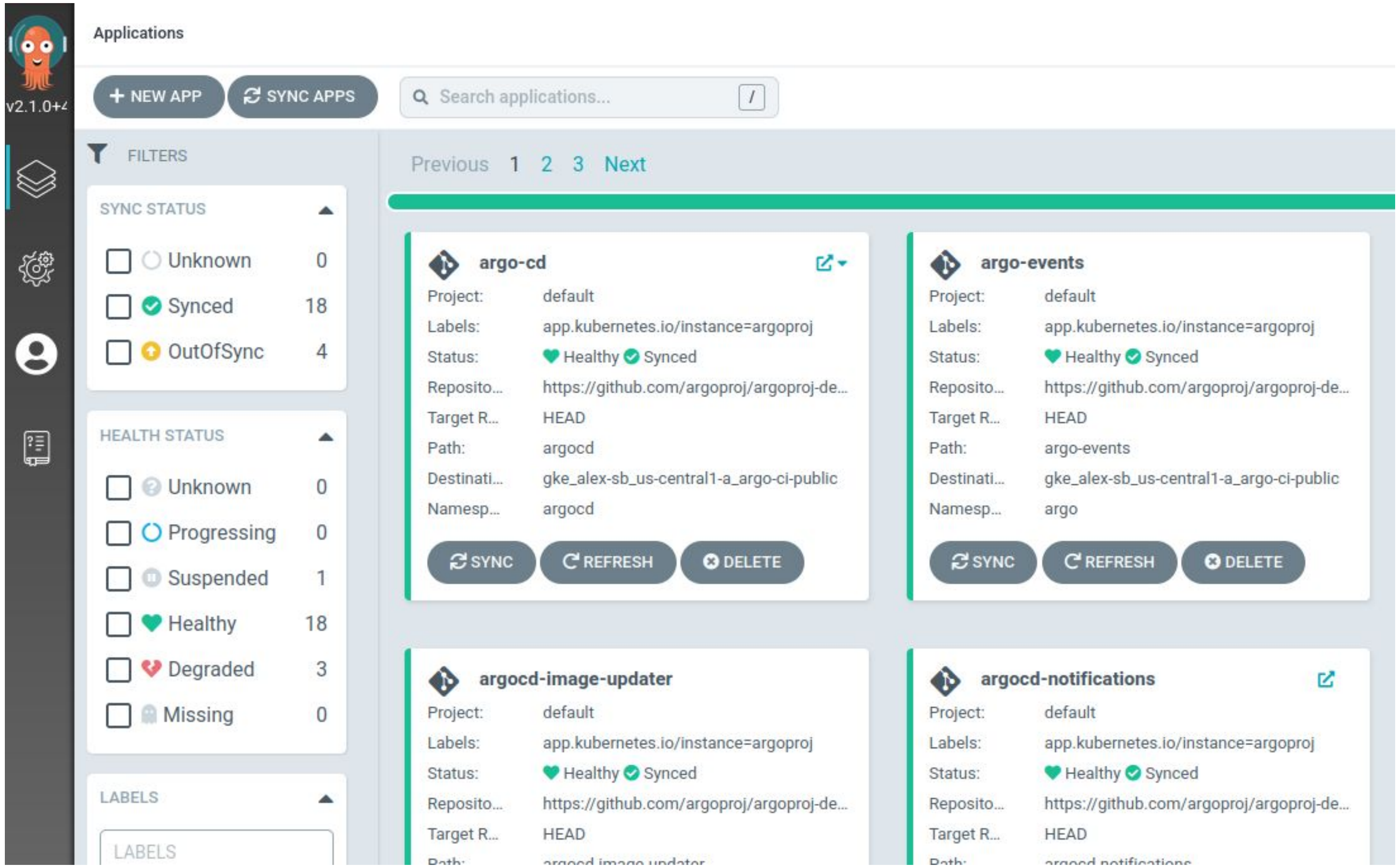


cluster	url
engineering-dev	https://kubernetes.default.svc
engineering-prod	https://kubernetes.default.svc

```
template:
  metadata:
    name: '{{cluster}}-guestbook'
  spec:
    source:
      repoURL: https://github.com/infra-team/cluster-deployments.git
      targetRevision: HEAD
      path: guestbook/{{cluster}}
    destination:
      server: '{{url}}'
      namespace: guestbook
```

Parameters are rendered into the template.
Producing 2 Argo CD Application CRs, targeting two different clusters.
engineering-dev-guestbook
engineering-prod-guestbook

ApplicationSet Demo



v2.1.0+4

Applications

+ NEW APP SYNC APPS Search applications...

FILTERS

SYNC STATUS

Unknown	0
Synced	18
OutOfSync	4

HEALTH STATUS

Unknown	0
Progressing	0
Suspended	1
Healthy	18
Degraded	3
Missing	0

LABELS

LABELS

Previous 1 2 3 Next

argo-cd

Project:	default
Labels:	app.kubernetes.io/instance=argoproj
Status:	Healthy Synced
Reposito...	https://github.com/argoproj/argoproj-de...
Target R...	HEAD
Path:	argocd
Destinati...	gke_alex-sb_us-central1-a_argo-ci-public
Namesp...	argocd

SYNC REFRESH DELETE

argo-events

Project:	default
Labels:	app.kubernetes.io/instance=argoproj
Status:	Healthy Synced
Reposito...	https://github.com/argoproj/argoproj-de...
Target R...	HEAD
Path:	argo-events
Destinati...	gke_alex-sb_us-central1-a_argo-ci-public
Namesp...	argo

SYNC REFRESH DELETE

argocd-image-updater

Project:	default
Labels:	app.kubernetes.io/instance=argoproj
Status:	Healthy Synced
Reposito...	https://github.com/argoproj/argoproj-de...
Target R...	HEAD

argocd-notifications

Project:	default
Labels:	app.kubernetes.io/instance=argoproj
Status:	Healthy Synced
Reposito...	https://github.com/argoproj/argoproj-de...
Target R...	HEAD

Putting it all together

Changes made to the parent **ApplicationSet** are automatically applied all the child Argo CD **Applications**.

- This allows you to manage the content of many **Applications** from a single **ApplicationSet**.

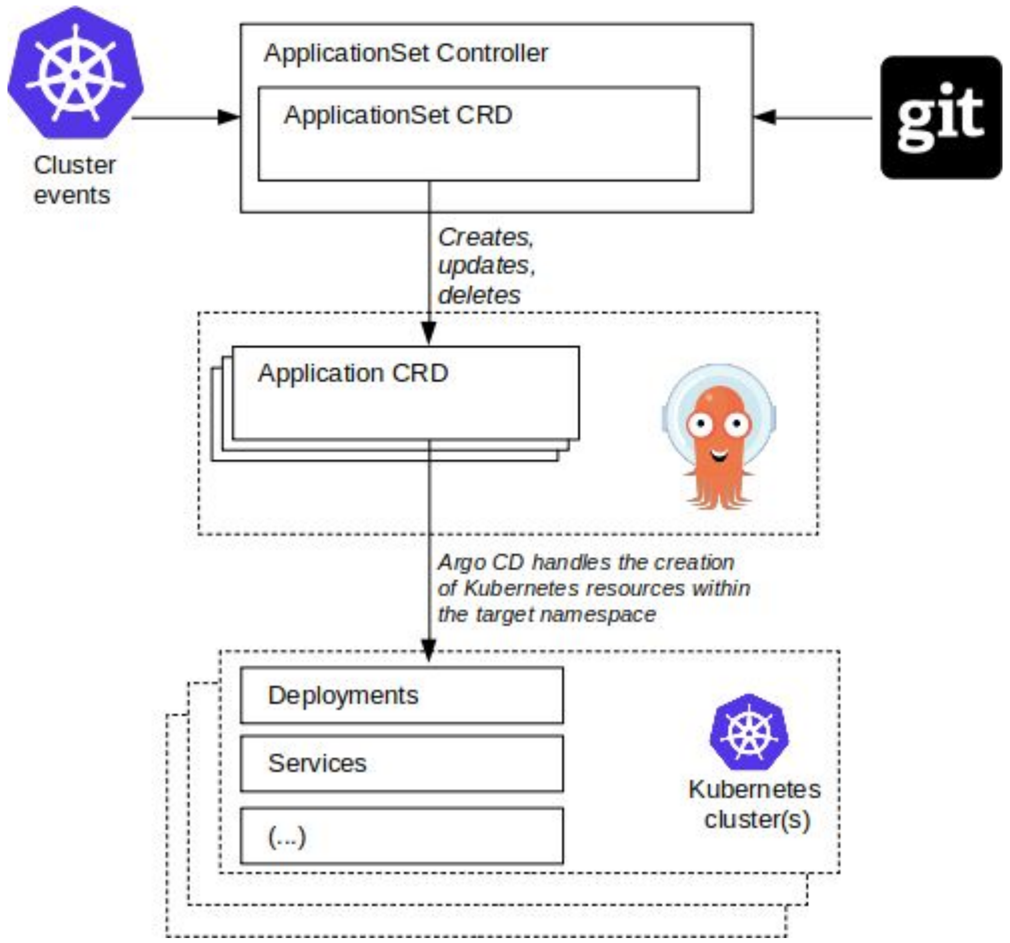
The ApplicationSet controller is responsible for reconciling the **ApplicationSets**.

The controller generates one or more **Application**, based on the contents of the template field of the **ApplicationSet**.

- This is where the ApplicationSet controller's responsibility ends.

Argo CD is responsible for its traditional responsibility of deploying the Application resources.

- Reading Kubernetes resources from Git
- Applying those resources to the target cluster/NS


Cluster events
ApplicationSet Controller
ApplicationSet CRD
git
Creates, updates, deletes
Application CRD
Argo CD handles the creation of Kubernetes resources within the target namespace
Deployments
Services
(...)
Kubernetes cluster(s)

Cluster Generator

What: Automatically reads the list of clusters defined within Argo CD, and create Applications for each Cluster.

Why: Allows you to automatically deploy Argo CD Applications to clusters as they are added to Argo CD, and remove those Applications as clusters are removed.

You can likewise target a subset of clusters using cluster annotations.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - clusters: {}
  template:
    metadata:
      name: '{{name}}-guestbook'
    spec:
      project: "default"
      source:
       repoURL: https://github.com/argoproj/argocd-example-apps/
       targetRevision: HEAD
       path: guestbook
     destination:
       server: '{{server}}'
       namespace: guestbook
```

Git Directory Generator

What: Scans through the contents of a Git repository, and for each matching directory, creates an Argo CD Application.

Why: For folks that prefer to define the K8s resources for multiple workloads / components / microservices within a single Git repo, this generator will automatically locate them within the repo. Additions/removal from Git will automatically be detected and reflected within K8s.

For the opposite approach, where each application component/microservice gets its own repository, see the SCM Provider generator.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: cluster-addons
spec:
  generators:
  - git:
      repoURL: https://github.com/argoproj-labs/applicationset.git
      revision: HEAD
      directories:
      - path: examples/git-generator-directory/cluster-addons/*
  template:
    metadata:
      name: '{{path.basename}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/argoproj-labs/applicationset.git
        targetRevision: HEAD
        path: '{{path}}'
      destination:
        server: https://kubernetes.default.svc
        namespace: '{{path.basename}}'
```

SCM Provider Generator

What: Scans the list of repositories of an organization within GitHub/GitLab, and create Applications based on matching repositories.

Why: Many folks prefer to split their application/microservice deployments across multiple repositories of a GitHub/GitLab organization. This generator will automatically scan an organization allowing deployment of all an organization's applications to a target location.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - scmProvider:
      github:
        organization: argoproj
      cloneProtocol: https
      filters:
        - repositoryMatch: example-apps
  template:
    metadata:
      name: '{{ repository }}-guestbook'
    spec:
      project: "default"
      source:
        repoURL: '{{ url }}'
        targetRevision: '{{ branch }}'
        path: guestbook
      destination:
        server: https://kubernetes.default.svc
        namespace: guestbook
```

Git File Generator

KubeCon | CloudNativeCon North America 2021

What: Scan through the contents of a Git repository, and for each matching JSON/YAML file, create an Argo CD Application.

Why: Defines a list of configuration files that describe the specific Applications to create. This gives fine-grained control over individual fields via Git commits.

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
    - git:
        repoURL: https://github.com/argoproj-labs/applicationset.git
        revision: HEAD
        files:
          - path: |
                "examples/git-generator-files-discovery/cluster-config/**/config.json"
  template:
    metadata:
      name: '{{cluster.name}}-guestbook'
    spec:
      project: default
      source:
        repoURL: https://github.com/argoproj-labs/applicationset.git
        targetRevision: HEAD
        path: "examples/git-generator-files-discovery/apps/guestbook"
      destination:
        server: https://kubernetes.default.svc
        namespace: guestbook
```

Matrix Generator

What: Combine the output of two generators, thus gaining the advantages of both.

Why: Highly flexible mechanism for combining generators, for example:

- Deploy every application in Git to every Argo CD cluster.
  - **Git Directory**: For every application in this Git repo…
  - **Cluster**: For every Argo CD cluster…
- Scan through an GitHub org and deploy repositories to a list of environments defined with YAML files in Git.
  - **Git File**: For every matching YAML files in a repo, convert the content to parameters...
  - **SCM Provider**: Scan through an organization and convert to parameters.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: cluster-git
spec:
  generators:
  - matrix:
      generators:
        - git:
            repoURL: https://github.com/argoproj-labs/applicationset.git
            revision: HEAD
            directories:
            - path: examples/matrix/cluster-addons/*
        - clusters:
            selector:
              matchLabels:
                argocd.argoproj.io/secret-type: cluster
  template:
    metadata:
      name: '{{path.basename}}-{{name}}'
    spec:
      project: '{{metadata.labels.environment}}'
      source:
        repoURL: https://github.com/argoproj-labs/applicationset.git
        targetRevision: HEAD
        path: '{{path}}'
      destination:
        server: '{{server}}'
        namespace: '{{path.basename}}'
```

Cluster Decision Resource Generator

What: ‘Outsource’ the logic of which clusters to place applications on, to an external CR.

Why: Perhaps the most complex generator (but also most powerful). Using it requires writing your own custom controller, defining a custom resource definition (CRD), and using your controller to modify the status field of the CR to indicate which clusters Applications should be deployed to.

Supports integration with Open Cluster Management project’s Placement Rule CR (and these are the folks that contributed the generator).

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: book-import
spec:
  generators:
    - clusterDecisionResource:
        configMapRef: ocm-placement
        name: test-placement
        requeueAfterSeconds: 30
  template:
    metadata:
      name: '{{clusterName}}-book-import'
    spec:
      project: "default"
      source:
        repoURL: |
             https://github.com/open-cluster-management/application-samples.git
        targetRevision: HEAD
        path: book-import
      destination:
        name: '{{clusterName}}'
        namespace: bookimport
```

ApplicationSets at Red Hat

ApplicationSets are built in to **Red Hat OpenShift GitOps**, along with Argo CD.

Learn more about how to deploy using Argo CD, and ApplicationSets, with OpenShift GitOps:

- An Introduction to ApplicationSets in OpenShift GitOps by Jonathan West & Dewan Ahmed
- Getting Started with ApplicationSets by Christian Hernandez

Red Hat OpenShift GitOps

ApplicationSets at Intuit

- Automate the addon installation customized across multiple kubernetes clusters.
- Git File Generator for Argo rollouts installation.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: argo-rollouts
spec:
  generators:
    - git:
        repoURL: 'https://github.com/argoproj-labs/applicationset.git'
        revision: HEAD
        files:
          - path: argo-rollouts/env/*/config.json
  template:
    metadata:
      labels:
        appset: argo-rollouts
        region: '{{region}}'
      name: 'argo-rollouts.{{name}}'
    spec:
      destination:
        namespace: argo-rollouts
        server: '{{server}}'
      info:
        - name: cluster-url
          value: 'https://kubernetes.default.svc'
      project: argo-rollouts
      source:
        path: 'argo-rollouts/env/{{name}}'
        repoURL: 'https://github.com/argoproj-labs/applicationset.git'
        targetRevision: HEAD
```

Jump In

Get started:

- ApplicationSets Getting Started: Introduction and quick start

Learn more:

- ApplicationSet docs: templates, generators, architecture, and more!

Join us:

- Argo CD ApplicationSet project on GitHub
- #argo-cd-appset on CNCF Slack

RESILIENCE
REALIZED

KubeCon | CloudNativeCon
North America 2021

Thanks for watching!
Questions?