The way of the force: GitOps on JEEDY

Antonio Nappi
Ioannis Panagiotidis

What we do

- Hosting CERN JAVA web applications on Tomcat/WebLogic running in Kubernetes
  - ~25 Kubernetes Clusters
  - more than 400 nodes
  - more than 3000 pods
- Some of applications that we hosts
  - EDH, Phonebook, EDMS, AISMEDIA...
- Our users are developers from different Departments

Which was the problem ?

- Prometheus servers:
  - for each K8s cluster
  - for each user community a dedicated Prometheus that federates the ones running on all their clusters
- Users wanted to:
  - Define custom alerting/recording rules
  - Define scraping endpoints
  - Define alertmanager configuration

Previous Solution

- Building rundeck jobs for each of the above requests, to deploy to our Kubernetes clusters
  - not scalable
  - hard to mantain
  - hard to track changes

Solution: GitOps

- Profit of all Git advantages:
  - tracebility
  - versioning

    - easy to roll back
- Build CI to validate content of a git repo
    - valid yaml, json etc..
- Works with declarative infrastructure tools

Implementation: ArgoCD

- Seemed more mature than Flux 1
- Faster growing community
- As today we use ArgoCD to manage
    - users and internal monitoring/alerting systems
    - user cronjob submissions
    - efiles.cern.ch deployment

JEEDY Repositories structure

3 Kind of repositories

1. Sources (e.g. prometheus-sources)
    - Helm Charts
    - Jsonnet
    - Kustomize
    - JSON/YAML files
2. Users (e.g. ais-users)
    - As sources but used to apply user customization to our Kubernetes Clusters
3. Applications (e.g. prometheus-applications)
    - Contains ArgoCD application definitions
        - Using pattern of application of applications
    - Using ArgoCD Application Sets as alternative

Managing cronjobs via GitOps

- ArgoCD is used to deploy ArgoWorkflows
    - ArgoWorkflows is a workflow engine that is used to orchestrate jobs in Kubernetes
    - Is used by Jeedy and DIR teams, other users are migrating to it
- Advantages:
    - Designed for containers, is implemeted as a CRD

  - Cloud agnostic
  - Offers a web UI and many extras: run jobs with a click, view logs, disable jobs etc

---

Demo

- With just 2 small commits we will:

1. As admins: Deploy a complete ArgoWorkflows instance (with custom configmaps, SSO functionality, Ingress, RBAC etc) to a new cluster
2. As users: Add/remove jobs

ArgoCD impressions (Part 1)

- Documentation
  - it is getting better but initially was pretty bad
- Scalability
  - Fixed in more recent versions
  - ArgoCD managing other ArgoCD instances
- Plugins
  - Really powerful, allows you to extend ArgoCD as you wish
  - When to move to BYOI (Build your own image) ?

ArgoCD impressions (Part 2)

- Installation is sold as GitOps oriented but not really easy to achieve
  - Clusters as stored as K8s secrets but have a no sense format
  - Repositories a bit better but still some bricolage to do
- ArgoCD is maturing

Secret Management: current solution

- ArgoCD Vault Pluging
  - Private instance of Vault
    - Not opened outside the group
  - It works pretty well but cannot be shared secrets with users
    - They first update secrets to Teigi and then we add them to Vault

---

Secret Management: possible alternative

1. Custom plugin that interacts with Teigi
    - BYOI of ArgoCD
2. Use KSOPS
    - Quite easy to integrate with ArgoCD
3. Sealed Secrets :x:
    - It is a HELL when you have many clusters
        - Operator in each cluster
        - Key rotation

Lessons Learned (Part 1)

- No Golden rule to structure repositories
    - base/overlays to minimize code
- Mono repo or multiple ones ?
    - Multiple
        - P: isolate different use cases and applications
        - C: tracking can be more difficult
            - what is managing what?
- One branch or multiple ones ?
    - One branch
        - P: easier to maintain and to update
        - C: no isolation between environments

Lessons Learned (Part 2)

- Upgrades
    - identical development instance of ArgoCD where we test upgrades
        - need to keep a real example of applications running on the Production instance
- Pruning
    - It is useful but at the same time also dangerous
        - Mostly disabled
- GitOps is not only Kubernetes

    - Deployment of K8s clusters via Terraform and gitlab
    - (Previous) Rundeck jobs stored in git
    - It isn't panacea

What's next

- Stabilize secret management
- Manage ArgoCD clusters via Git
- Enabling GitOps for more applications

Overall impression

- Really happy about GitOps adoption

    - It speed up the deployment of infrastructure components

        - Faster recover after a major incident (e.g. Kubernetes deletion incident (https://indico.cern.ch/event/1140863/contributions/4794756/attachmer

    - Facilitate customization
    - Better control on what is applied to the infrastructure

---

<font size="25">
<b style="text-align: center;" >Thank you </b>
</font>
</br>
<b style="text-align: center;" >antonio.nappi@cern.ch</b>
</br>
<b style="text-align: center;" >ioannis.panagiotidis@cern.ch</b>