Kubernetes Operators

as composable parts of the new Drupal SaaS

Konstantinos Samaras-Tsakiris
IT-CDA-WF
Coalescing Web Frameworks on K8s
Unique infrastructure per use case

<table>
<thead>
<tr>
<th>PaaS</th>
<th>Openshift 3 (Kubernetes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebEOS</td>
<td>VMs with custom config</td>
</tr>
<tr>
<td>Drupal</td>
<td>Physical machines with custom config</td>
</tr>
</tbody>
</table>

Low reuse of components
Converging on a cloud native platform

PaaS
WebEOS
Drupal

Openshift 4
Openshift 4
Openshift 4

Many shared components
### Platform components

<table>
<thead>
<tr>
<th>Component</th>
<th>Component</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>argo-cd</td>
<td>cvmfs-csi</td>
<td>okd-console-configuration</td>
</tr>
<tr>
<td>authz-operator</td>
<td>dbod</td>
<td>okd-registry-configuration</td>
</tr>
<tr>
<td>cephfs-csi</td>
<td>dns-manager</td>
<td>paas</td>
</tr>
<tr>
<td>cern-accounts-integration</td>
<td>drupal</td>
<td>reserved-hostnames</td>
</tr>
<tr>
<td>cern-okd-admin-service-account</td>
<td>eosxd</td>
<td>selinux-configuration</td>
</tr>
<tr>
<td>cert-manager</td>
<td>force-clusterversion</td>
<td>shared-image-streams</td>
</tr>
<tr>
<td>cluster-logging</td>
<td>landb-operator</td>
<td>tektoncd</td>
</tr>
<tr>
<td>cluster-state-backup</td>
<td>logviewer</td>
<td>webeos</td>
</tr>
<tr>
<td>custom-ingress-deployment</td>
<td>monitoring</td>
<td>worker-nodes</td>
</tr>
</tbody>
</table>

**Common pattern:** Operator
Drupal @ CERN
Drupal @ CERN

Not just hosting, but fully managed
Software as a Service
Parts of a Drupal site
Drupal SaaS is tough

➢ Take 1500 instances of a complicated thing
➢ Automate business/operational logic
➢ Let users self-provision websites
➢ All this with a very small team!
Drupal sites on K8s
Kubernetes

“Container orchestrator”

- not a workflow engine
- set of independent, composable control processes
- continuously drive the current state towards the provided desired state

Ref: kubernetes.io/docs/concepts/overview/what-is-kubernetes/
OKD4 (OpenShift) cluster

website project

DrupalSite
Operator Pattern
What *should* be in the cluster
What *should* be in the cluster
What *should* be in the cluster

"the world": what is running in the cluster

reconciliation
Making operators

- **k8s event queue**
- **Reconcile logic**
  - Fetch CR
  - Check State
  - Ensure State
  - Update Status

**Watch**
- k8s API server client
  - Reconcile loop
Diving into our operators
apiVersion: drupal.kubecon.cern.ch/eu2021
kind: DrupalSite
metadata:
  name: kubecon
spec:
  drupalVersion: "9.1.x"
publish: true
siteUrl: kubecon.webtest.cern.ch
environment:
  name: "dev"
qosClass: "standard"
dbodClass: "test"
diskSize: "1Gi"
The DrupalSite operator

apiVersion: drupal.kubecon.cern.ch/eu2021
kind: DrupalSite
metadata:
  name: kubecon
spec:
  drupalVersion: "9.1.x"
publish: true
siteUrl: kubecon.webtest.cern.ch
environment:
  name: "dev"
  qosClass: "standard"
  dbodClass: "test"
diskSize: "1Gi"

status:
  conditions:
    - type: Installed
      status: "False"
    - type: Ready
      status: "False"
      reason: DBODError
    - type: UpdateNeeded
      status: Unknown
      reason: k8sAPIClientError
      message: 'k8sAPIClientError: Deployment.apps "kubecon" not found'
Operator Capabilities

**Where we are now**

**Our Goal**

---

**Level I**

**Basic Install**
Automated application provisioning and configuration management

**Level II**

**Seamless Upgrades**
Patch and minor version upgrades supported

**Level III**

**Full Lifecycle**
App lifecycle, storage lifecycle (backup, failure recovery)

**Level IV**

**Deep Insights**
Metrics, alerts, log processing and workload analysis

**Level V**

**Auto Pilot**
Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

Ref: https://sdk.operatorframework.io/docs/overview/
Composing Operators

Not only DrupalSites need to integrate with external services

➢ CRDs make operators composable
Demo: Upgrading Drupal sites
Update workflow

1. Enable site maintenance
2. Take DB snapshot
3. Roll out new images
4. Update DB schema
5. Disable site maintenance
6. Continue reconciliation
7. Rollback update
8. Report in Status
What have we discovered?
Development practices

● GitOps
  ○ Cluster configuration with Helm charts
  ○ Maintain with ArgoCD Applications
  ○ Validation with e2e tests

● Auto-provisioning development clusters
  ○ Almost same configuration as production clusters

● Operator Framework
  ○ Kubebuilder book
Conclusions

- We can provision a highly automated infrastructure to solve a complex problem with a very small team →
  [gitlab.cern.ch/drupal/paas/drupalsite-operator](gitlab.cern.ch/drupal/paas/drupalsite-operator)

- We used the operator model as a critical part of our design.

- Kubernetes as a common API to control many kinds of resources.
That's all Folks!