A Story of GitOps or How Kohl's Manages OpenShift at-scale

Heiko Zuerker / Senior Staff Architect @ Kohl's
Johnathan Kupferer / Solutions Architect @ Red Hat
05/08/2019
1,100+ stores in 49 states

$20B annual revenue

140K employees

$700M to communities
## OUR OPENSHIFT JOURNEY

<table>
<thead>
<tr>
<th>Year</th>
<th>OpenShift Version</th>
<th>Description</th>
<th>Clusters</th>
<th>Nodes</th>
<th>Containers</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td>OpenShift 3.4</td>
<td>Prototype</td>
<td>3 Clusters</td>
<td>&lt;200</td>
<td>&lt;200</td>
<td>Dedicated L2 Support</td>
</tr>
<tr>
<td>2017</td>
<td>OpenShift 3.6</td>
<td>“Sorta” stable</td>
<td>10 Clusters (Planned and staffed for 4)</td>
<td>&lt;200</td>
<td>&lt;200</td>
<td>Dedicated L2 Support</td>
</tr>
<tr>
<td>2018</td>
<td>OpenShift 3.9</td>
<td>Zero issues during “peak”</td>
<td>19 Clusters</td>
<td>1000+</td>
<td>8000+</td>
<td>Dedicated L2 Support</td>
</tr>
<tr>
<td>2019</td>
<td>OpenShift 3.11</td>
<td>Rock-solid, cheap, &amp; stateful</td>
<td>18 Clusters</td>
<td>1200+</td>
<td>25000+</td>
<td>Shared L2 Support</td>
</tr>
</tbody>
</table>

*GitOps for App/Dev*

*DevOps Infrastructure As Code*
**OPENSHIFT @ KOHL’S**
**MAIN DESIGN FEATURES**

- Shared & dedicated clusters
- Agility & Flexibility
- Organic Patching
- Drift Management
- Support snowflakes
- Managed via GitOps
- Automate Everything
- Everything-as-Code
- Idempotent Automation
- Immutable Infrastructure
- Ephemeral Nodes*
- Platform Autoscaling

Applying container principles and best practices to the platform itself!

* Everything except master and gluster nodes (for now)
INNER SOURCE
TAKING OPEN SOURCE PRINCIPLES AND APPLYING THEM INTERNALLY!

Everything that makes Open Source great, for example...

- Open and inviting
- All repos can be viewed
- External changes are encouraged
- Collaboration across teams
- “Outsourcing” of work
- Shared ownership
- Fast turnaround
- Quality code
GitOps = Turning Day 2 Operations into Code!
Thinking about Day 2 on Day 1

Leveraging best practices learned from DevOps and using them to manage “everything-as-code”

Next Generation Change Management
- Version Control
- Peer Reviews
- Audit Trail
- Reproducibility, Consistency, & Reliability

Efficiency
- Disposable Infrastructure
- Scalability of Team
- “Outsource” via Inner Source
Infrastructure Pipeline
SBX → DEV → QA → PROD

Infrastructure Provisioning

OpenShift Prerequisites

OpenShift Installation

Post Installation Tasks

Configure Jobs

Configure Roles

Configure Logging

Configure Monitoring

Configure Namespaces

Configure Certificates

Configure ...
**EXAMPLE: CONFIGURATION**

(SIMPLIFIED)

**Cloud**

**Environment**

**Line-of-Business**

**Release**

**Default**

- Grafana.yaml
- Prometheus.yaml
- Roles.yaml

**Cluster**

- cluster1: routers = 6
- cluster2: routers = 10

**Sandbox:** routers = 1

**Production:** audit_log = true

Managing OpenShift resources via Ansible:
https://github.com/gnuthought/ansible-role-openshift-provision
PLATFORM AUTO SCALING
SCALING THE UNDERLYING NODES BASED ON THE CURRENT RESOURCE DEMANDS (CPU & MEMORY REQUESTS)

- Implemented in GCP for OpenShift 3.6 leveraging kube-autoscaler
- ~3 minutes for new nodes to become available
- Maximum node lifetime 7 days
- No guarantees for how long pods will live

Chaos-Engineering Lite

<table>
<thead>
<tr>
<th>3.6 &amp; 3.9</th>
<th>3.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom cluster-join logic</td>
<td>TLS bootstrapping</td>
</tr>
<tr>
<td>custom scale-down logic</td>
<td>kube-autoscaler</td>
</tr>
<tr>
<td>custom capacity logic</td>
<td>pre-scaling with preemptible pods</td>
</tr>
<tr>
<td>scale-down oldest node</td>
<td>kube-autoscaler</td>
</tr>
</tbody>
</table>
ORGANIC PATCHING

RELY ON PLATFORM AUTOSCALING TO SLOWLY REPLACE NODES, AS THE CLUSTER FLEXES UP AND DOWN

- Cordon “outdated” nodes to force new pods onto new nodes
- “Nudge” the cluster every couple of hours (scale’n’drain)
- Used for configuration changes
- Used for platform patching
“THE LAW” / “TOUGH LOVE”

- Containers are not VMs and will not be treated as such
- Pods must be immutable
- Pods must have health checks and liveness probes
- Pod lifetime not guaranteed & we won’t warn you about shutdowns
- No manual config beyond development environments
- No privileged containers
- You’re responsible for your own HA & DR
- You must manage your own state and backup your own data
EMBRACING FAILURE

“Failure is always an option.”
~ Adam Savage

“It’s a beautiful day at the bomb range. Birds are singing, rabbits are hopping about...and pretty soon there’s going to be a big explosion.”
~ Jamie Hyneman

#redhat #rhsummit
IT’S EARLY DECEMBER...
WE ARE IN PEAK SEASON...

and here comes... CRITICAL CVE-2018-1002105

“Kubernetes privilege escalation and access to sensitive information in OpenShift products and services.”

Patched 19 OpenShift clusters in 2 days
400+ application nodes (VMs) replaced
1000+ business application pods moved
0 (zero) business impact!
GitOps for App/Dev

- Stop funneling requests through a central team
- We don’t care (much) what teams do in “their namespaces”
  - Chargeback
  - Cluster stability & security
- Guardrails via policy automation
- Remove dependencies to external teams
  - Automate DNS entries for custom routes
- Ephemeral namespaces

Kohl’s k8s GitOps Operator soon to be released under: https://github.com/KohlsTechnology
Decentralizing → “OpenShift-in-a-box” / self-managing clusters

Nightly cluster build, validation, & full GitOps test suite

Consistent automation for all k8s platforms

Open Sourcing as much as we can

Less custom engineering

OpenShift 4.x
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos

facebook.com/redhatinc
twitter.com/redhat