Lessons Learned using the Operator Pattern to build a Kubernetes Platform

Pavlos Ratis (@dastergon)
Senior Site Reliability Engineer, Red Hat

SREcon21

About

@dastergon | dastergon/awesome-sre |
dastergon/awesome-chaos-engineering |
dastergon/wheel-of-misfortune

Red Hat

Red Hat OpenShift

Flavour of Kubernetes

CC-BY-4.0

Like RHEL for Linux

Red Hat
Enterprise
Linux

Red Hat OpenShift Plans

Red Hat OpenShift Managed Services

We manage it for you

Red Hat OpenShift Service on AWS

Azure Red Hat OpenShift

Red Hat OpenShift on IBM Cloud

Cloud Native offerings jointly managed by Red Hat and Cloud Provider

Red Hat OpenShift Dedicated

Managed by Red Hat

Red Hat OpenShift Container Platform

You manage it, for control and flexibility

Red Hat OpenShift Container Platform

On public cloud, or on-premises on physical or virtual infrastructure[1]

Source: 1 See docs.openshift.com for supported infrastructure options and configurations

Challenges in OpenShift

Service Mesh	App-Services	DB-Services
CI/CD	DNS	Authentication
Monitoring	Kubernetes	Automation
Logging	Registry	Security
Compute	Storage	Network

- Support in a variety of clouds
- Tribal expertise knowledge
- Toil

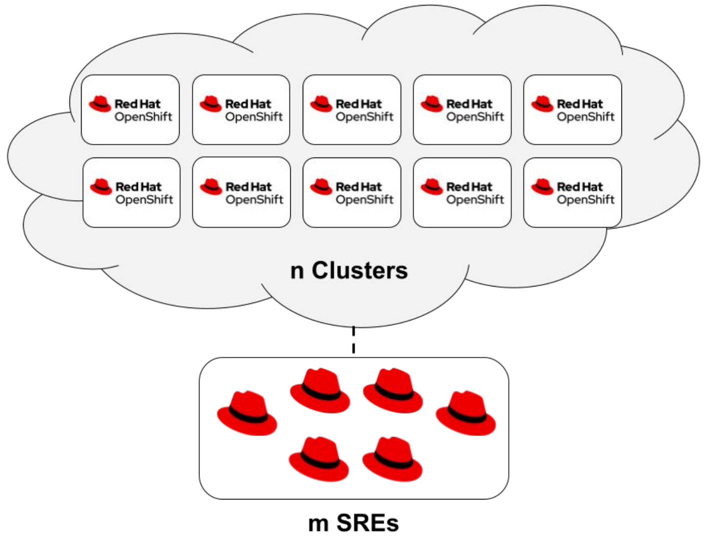
What could toil be in Kubernetes?

Repeatedly running multiple, manual commands to

- upgrade, configure, setup a cluster
- manage state of multiple clusters
- renew certificates
- troubleshoot 1...N clusters

Challenges in SRE

- On-call on a large fleet of clusters
- Manual SRE response to many clusters doesn't scale
- Toil work & maintenance cost us productivity



Remediations

Runbooks

Grow the organization

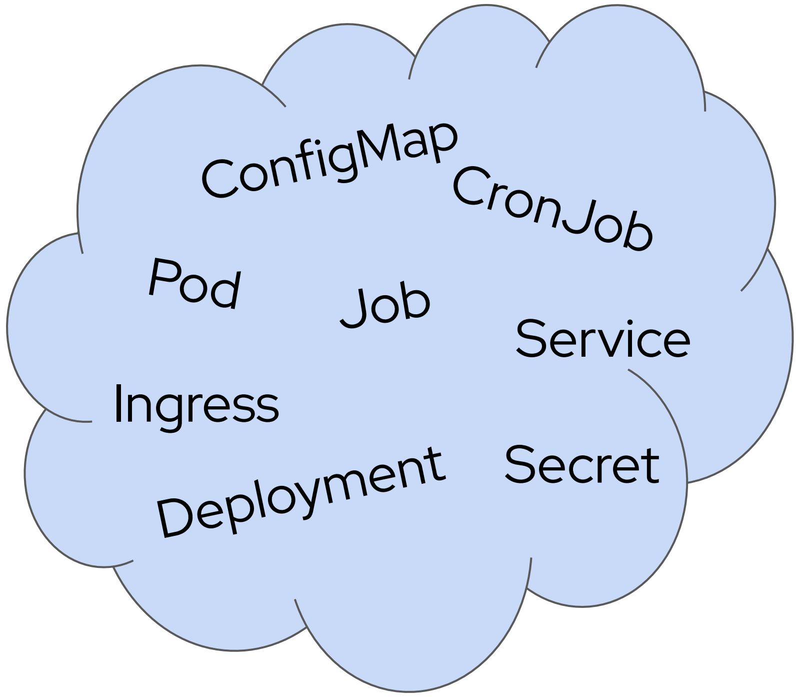
Automation

Operators Definition

*“Operators are **software extensions to Kubernetes** that make use of **custom resources** to **manage applications** and their components.”* – kubernetes.io

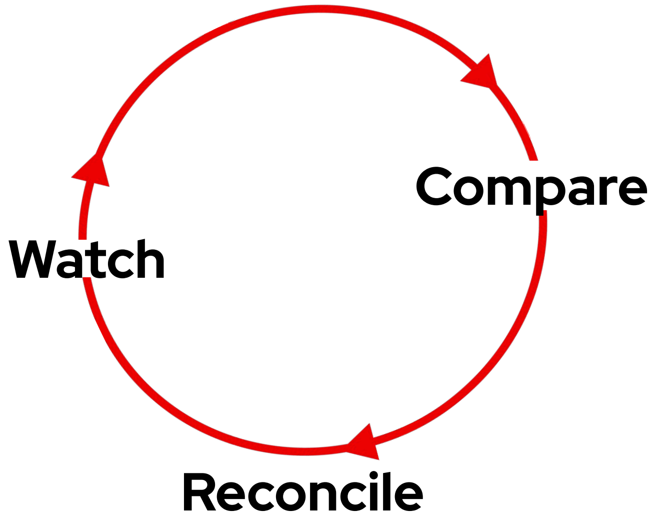
Red Hat

Native Kubernetes Resources



Native Kubernetes Controllers

- Built-in control loops
- Watch for actual and desired state
- Compare & when the states diverge, reconcile



The (Holy) Kubernetes API

- The core of of the Kubernetes control plane
- Everything speaks to it
- Manipulate and query the states of API objects
- kubectl & code to interact with the API

CC-BY-4.0

Red Hat

The Operator Pattern

The Operator Pattern

- A design pattern for Kubernetes introduced by CoreOS
- A method of packaging, deploying and managing a Kubernetes application
- Models a business/application specific domain
    - Stateful Apps (Elasticsearch, Kafka, MySQL)
    - Monitoring (Prometheus)
    - Configuration
    - Logging

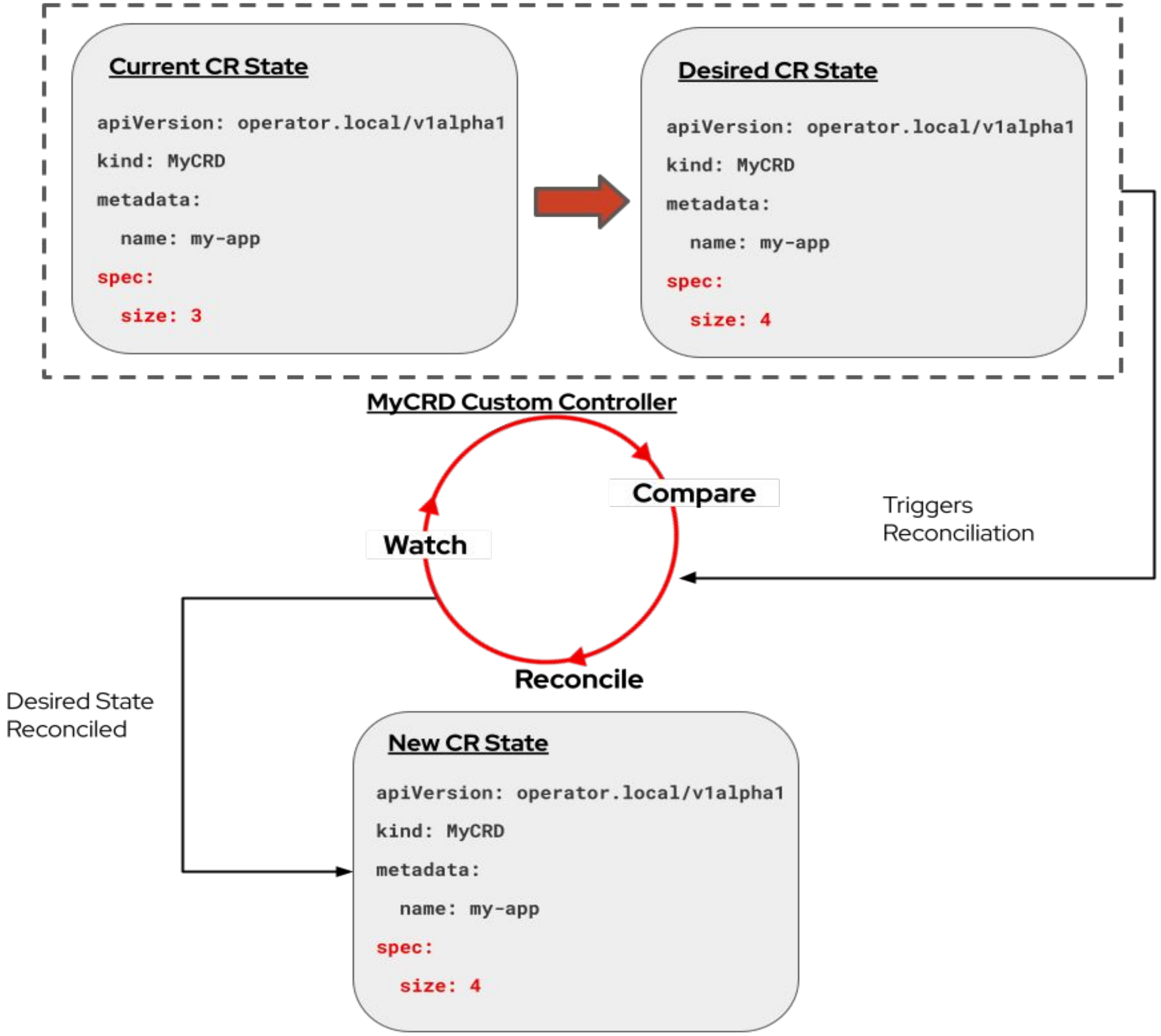
Knowledge Codification

- Transfer human engineering knowledge and operational sane practices for a specific domain to code
- SRE as Code
  - Deploy an application on demand
  - Take care of the backups of the state
  - Interact with some external 3rd party APIs
  - Auto-remediate in case of failures
  - Clean-ups
- Treat operations as a software problem

Operators - The building blocks

- Uses the native **Custom Resource Definition (CRD)** resource to extend the Kubernetes API
- Uses a **custom Controller** to interact with the CRD

Example Operator



Current CR State

```
apiVersion: operator.local/v1alpha1
kind: MyCRD
metadata:
  name: my-app
spec:
  size: 3
```

Desired CR State

```
apiVersion: operator.local/v1alpha1
kind: MyCRD
metadata:
  name: my-app
spec:
  size: 4
```

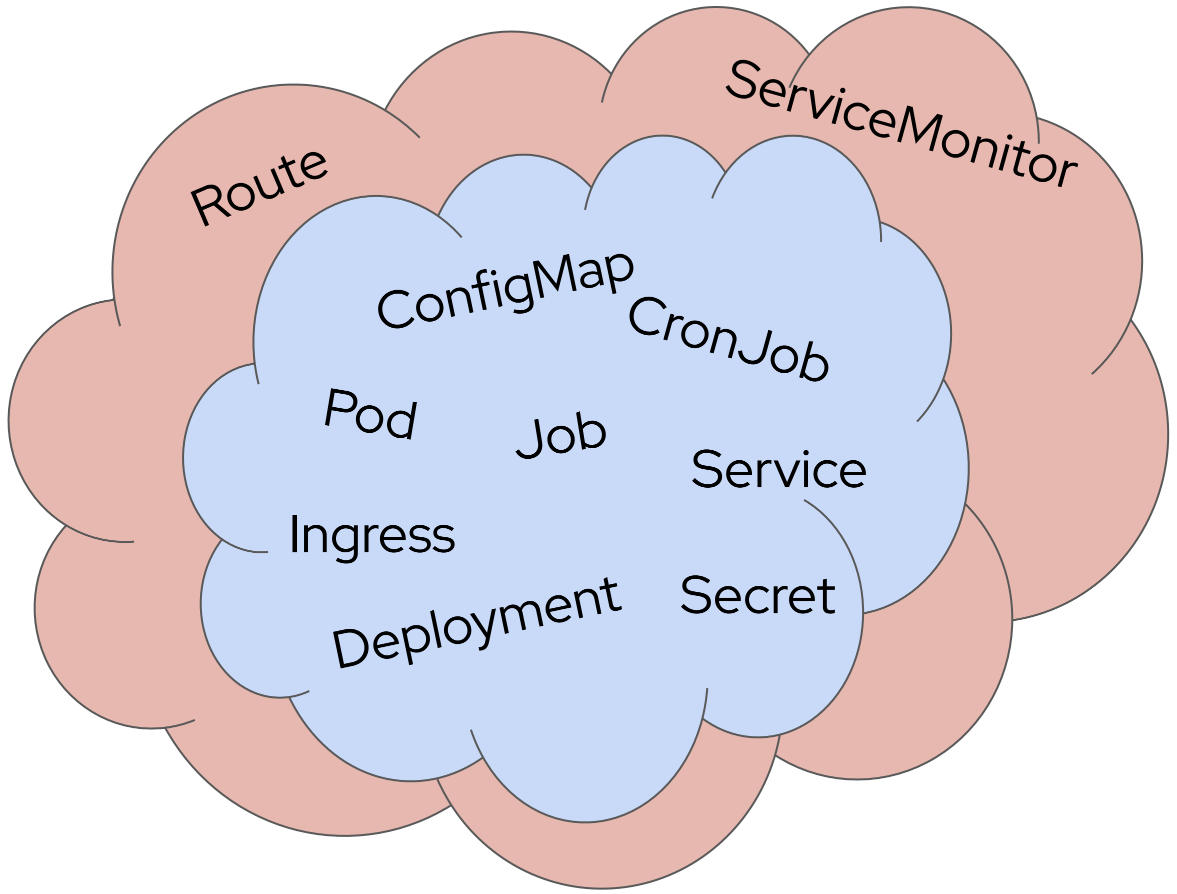
New CR State

```
apiVersion: operator.local/v1alpha1
kind: MyCRD
metadata:
  name: my-app
spec:
  size: 4
```

The Operator Pattern

- Good way to extend the functionality of Kubernetes
- Narrow context software
- Separation of concerns
- Over-the-air upgrades
- Abstraction possibilities

OpenShift CRDs



Example: Route

```
apiVersion: v1
kind: Route
metadata:
  name: route-example
spec:
  host: www.example.com
  path: "/test"
  to:
    kind: Service
    name: service-name
```

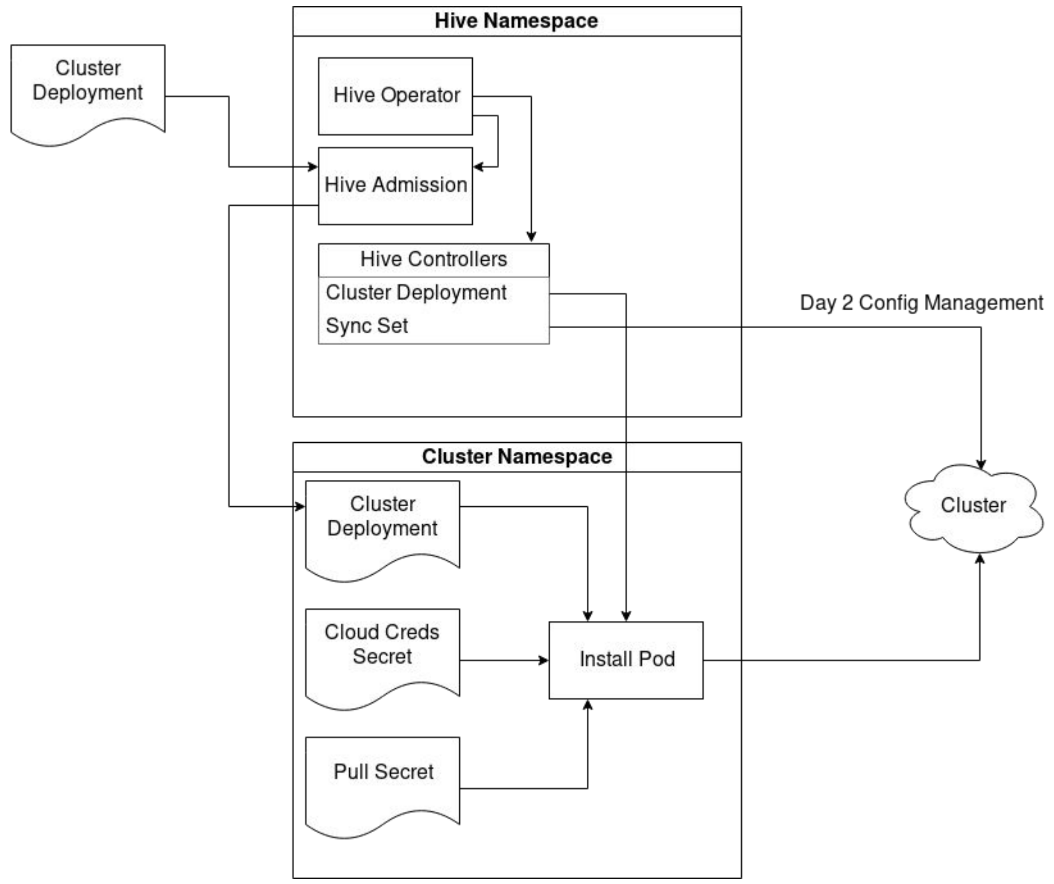
Expose a service by giving it an externally reachable hostname

OpenShift Operators

- cluster-logging-operator
- cluster-monitoring-operator
- cluster-config-operator
- cluster-etcd-operator

Find more at https://github.com/openshift

OpenShift as a Service



https://github.com/openshift/hive

Hive

- Kubernetes operator
- Declarative API to **provision, configure, reshape, and deprovision** OpenShift clusters
- Support for AWS, Azure, GCP.

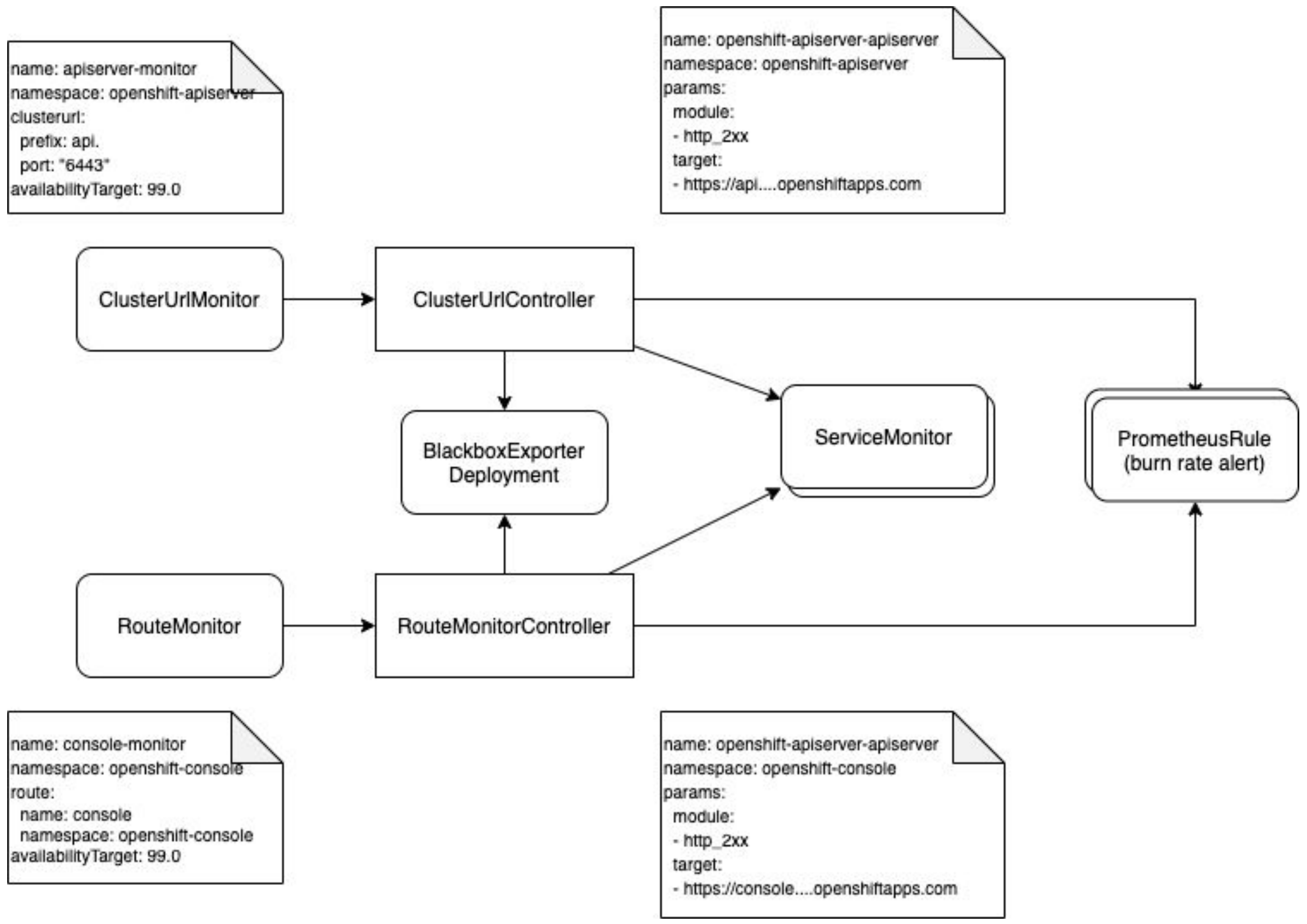
@dastergon

SRE at Red Hat OpenShift

- Automate operations and reduce toil work
- Our SREs are primarily focused on developing software
    - Operators (i.e, route-monitor-operator)
    - Internal tooling (i.e, osdctl, pagerduty-short-circuiter)
- SRE teams are structured as feature development teams and follow the Agile practices
- Part of on-call rotation

OpenShift Route Monitor Operator

- In-cluster operator to monitor liveness of Routes with blackbox probes
- How we set our SLOs for critical components
- Multiwindow, Multi-Burn-Rate Alerts



```
name: apiserver-monitor
namespace: openshift-apiserver
clusterurl:
  prefix: api.
  port: "6443"
availabilityTarget: 99.0
```

```
name: openshift-apiserver-apiserver
namespace: openshift-apiserver
params:
  module:
  - http_2xx
  target:
  - https://api....openshiftapps.com
```

```
name: console-monitor
namespace: openshift-console
route:
  name: console
  namespace: openshift-console
availabilityTarget: 99.0
```

```
name: openshift-apiserver-apiserver
namespace: openshift-console
params:
  module:
  - http_2xx
  target:
  - https://console....openshiftapps.com
```

https://github.com/openshift/route-monitor-operator

@dastergon

Red Hat

Community Operators

- Prometheus Operator
- Elasticsearch (ECK) Operator
- Zalando's Postgres Operator
- Apple's FoundationDB Operator
- Apache Spark Operator

Find more at OperatorHub.io

Operators Development

- The Operator Framework
    - Streamlines Operator development
    - Scaffolding tools (based on kubebuilder)
    - Tooling for basic CRD refactoring
    - Tooling for testing and packaging operators

Who operates the Operator?

- Operator Lifecycle Manager (OLM)
  - Declarative way to install, manage, and upgrade Operators and their dependencies in a cluster.
  - Oversees and manages the lifecycle of all of the operators

Lessons Learned

Sane Practices

- Use an SDK framework (operator-sdk, kubebuilder, metacontroller)
- Create Operators based on business needs
- Use 1 operator: 1 application (Elasticsearch, Kafka etc.)
  - An operator can have multiple controllers and CRDs though
- Standardize conventions & tooling
- Follow the same versioning scheme
- Monitor, log and alert like you would in a microservice

Pitfalls

- The pattern could be abused
  - The curse of autonomy
  - Operator all things!
- Different teams, different operators, following different
  - conventions
  - SDK versions
  - testing frameworks & methods
- Compatibility issues
  - Resource incompatibility (version v1alpha1 vs version v1beta1)
  - Code incompatibilities
- Not testing early enough

Just like any software...

- Software rots over time
  - Many changing parts
    - Requirements might change
    - Dependencies change
    - SREs in the team come and go
  - Needs constant care

SRE the Operators

- Out-of-the box support for metrics
  - Establish meaningful SLIs
- A dashboard per operator
- Logging in all layers
- Alert on symptoms
  - PersistentVolume Filling Up
  - Operator is degraded
- Check the volume of CRs your operator will create over time and clean up if necessary

Standardization

- Standardize code conventions
    - Use scaffolding tools (i.e., operator-sdk) when creating new operators
    - Create Operator Development Guidelines
- Unify tooling
    - Compile, build, test and deploy all the operators the same way
- https://github.com/openshift/boilerplate

Code Quality

- Golang CI-lint in our CI
- Security code checks: gosec
- Image Vulnerability Scans: Quay.io
- Delve for debugging
- pprof for performance diagnostics

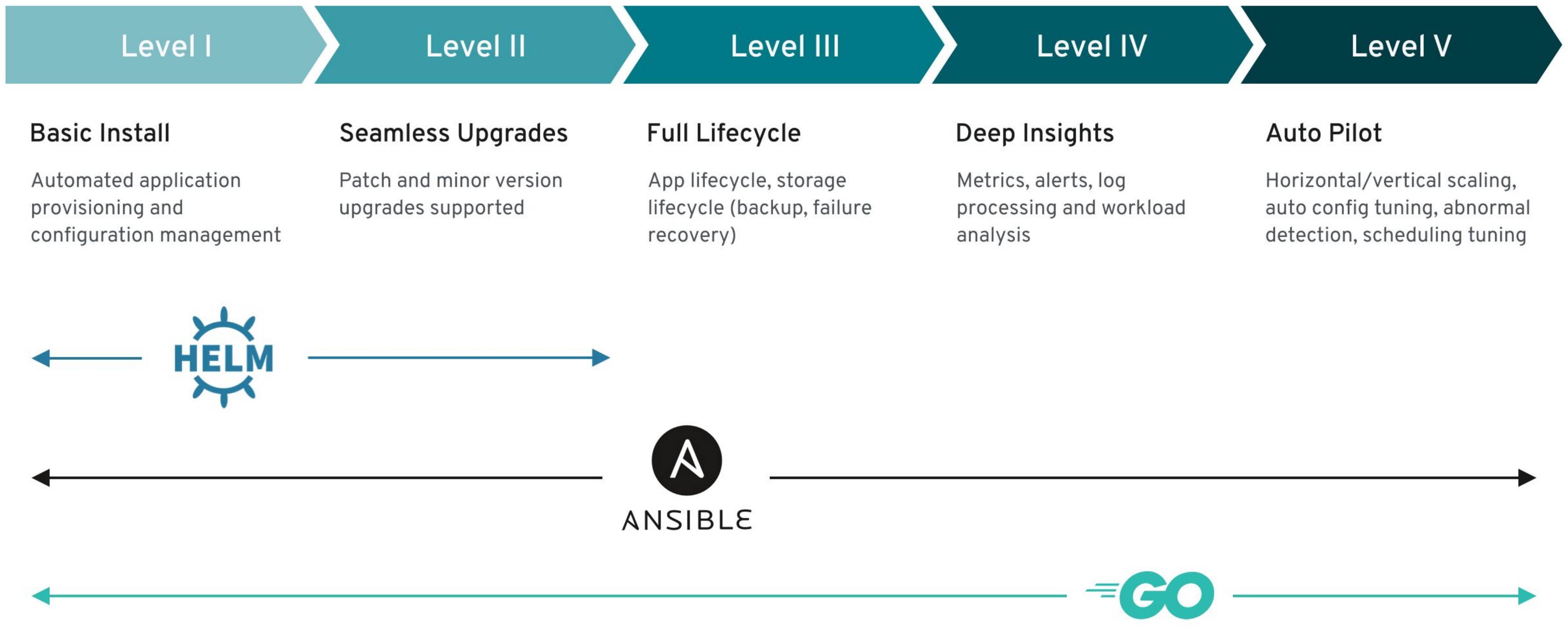
Red Hat

Testing

- Testing libraries
    - Go’s native test library
    - Ginkgo (BDD)
- Fake/mock libraries for unit testing
    - k8s fake package
    - kubebuilder’s envtest
- Local testing (Kind, crc) and staging clusters for integration tests
- Test the operators end-to-end
    - OSDe2e: Automated validation of all new OpenShift releases
    - https://github.com/openshift/osde2e

Red Hat

Excuse me, what about Helm?


Level I
Level II
Level III
Level IV
Level V
Basic Install
Automated application provisioning and configuration management
Seamless Upgrades
Patch and minor version upgrades supported
Full Lifecycle
App lifecycle, storage lifecycle (backup, failure recovery)
Deep Insights
Metrics, alerts, log processing and workload analysis
Auto Pilot
Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning
HELM
ANSIBLE
GO

Source:
https://sdk.operatorframework.io/docs/overview/operator-capabilities/

Microservices vs Operators?

- Operators are microservices that use Kubernetes CRs as API
- Operators
  - good for extending Kubernetes capabilities
  - event subscription through the Kubernetes API
  - concurrency control (optimistic locking)
  - integrate with Kubernetes' RBAC system
- But…
  - coupled to Kubernetes
  - shouldn't replace your current microservice architecture
  - migrating a running operator (+CRs) to a new cluster (data migration) is a big challenge
  - What if we need to move state from one cluster to another in another region?
- We plan to convert a few of our SRE-developed operators to microservices for some the above reasons

Automate all things?

“Ironically, although intended to relieve SREs of work, **automation adds to systems complexity** and can easily make that work even more difficult.” - Allspaw, John & Cook, Richard. (2018). SRE Cognitive Work.

Operators or not?

- Kubernetes native capabilities
- Kubectl plugins
- Helm Charts
- Off-the-shelf Operators
- DIY Operators

Red Hat

Resources

- CoreOS' original article
- Kubernetes Operators official page
- CNCF Operator White Paper
- Kubernetes Operators book
- Red Hat's article on Operators
- Operator Best Practices
- Is there a Helm and Operators showdown?

Red Hat

Resources on Red Hat SRE

- From Ops to SRE: Evolution of the OpenShift Dedicated Team
- 5 Agile Practices Every SRE Team Should Adopt
- 7 Best Practices for Writing Kubernetes Operators: An SRE Perspective
- Closed Box Monitoring, the Artist Formerly Known as Black Box Monitoring

Thank you!

We are hiring!

jobs.redhat.com

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat