



Red Hat

From Install to Auto Pilot: Exploring Kubernetes Operator Capability Levels

Presented by:

Operator Enablement &
Certified Technology Ecosystem Teams



How Many People Here Know What an Operator is?



Have Built an Operator?



Did you use Operator-SDK?



Did you get **Red Hat Certified** or publish
to **OperatorHub.io** yet?

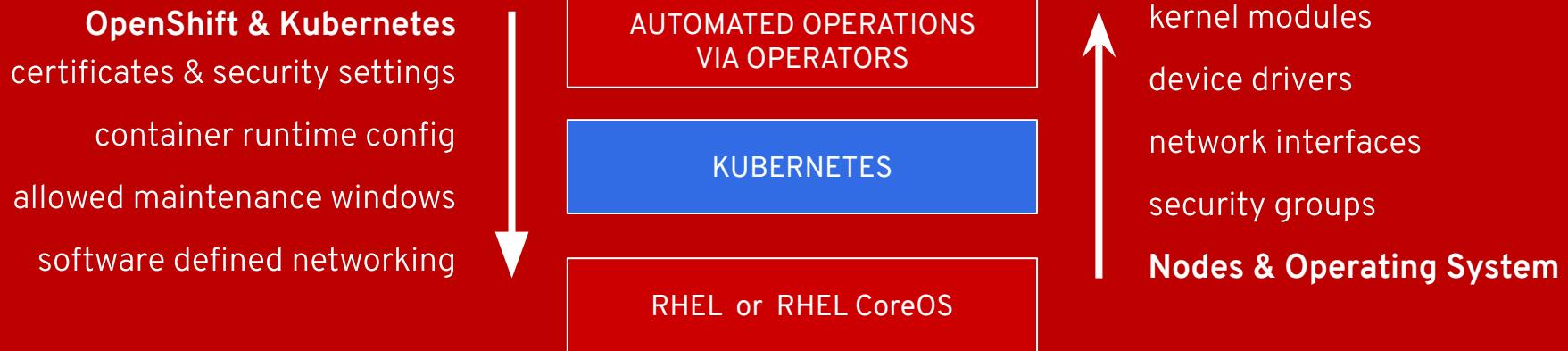


What is an Operator?

Kubernetes Operators are the Magic Behind
OpenShift 4's Automated Operations.

try.openshift.com

OpenShift 4 is aware of the entire infrastructure and brings the Operating System under management



Yeah..but what is it?

Operators

An Operator represents human operational knowledge in software, to reliably manage an application.

Still Confused?

Let's go back 3 years ago.

[← Back to All Blogs](#)

Introducing Operators: Putting Operational Knowledge into Software

November 03, 2016 • By Brandon Philips

Tags: [announcements](#) [Operators](#)

A Site Reliability Engineer (SRE) is a person that operates an application by writing software. They are an engineer, a developer, who knows how to develop software specifically for a particular application domain. The resulting piece of software has an application's operational domain knowledge programmed into it.

Our team has been busy in the Kubernetes community designing and implementing this concept to reliably create, configure, and manage complex application instances atop Kubernetes.

We call this new class of software Operators. An Operator is an application-specific controller that extends the Kubernetes API to create, configure, and manage instances of complex stateful applications on behalf of a Kubernetes user. It builds upon the basic Kubernetes resource controller concepts but includes domain or application-specific knowledge to automate common tasks.

3

It builds upon the basic Kubernetes resource and controller concepts but includes domain or application-specific knowledge to automate common tasks.

1

2

1

(Custom) Resource

2

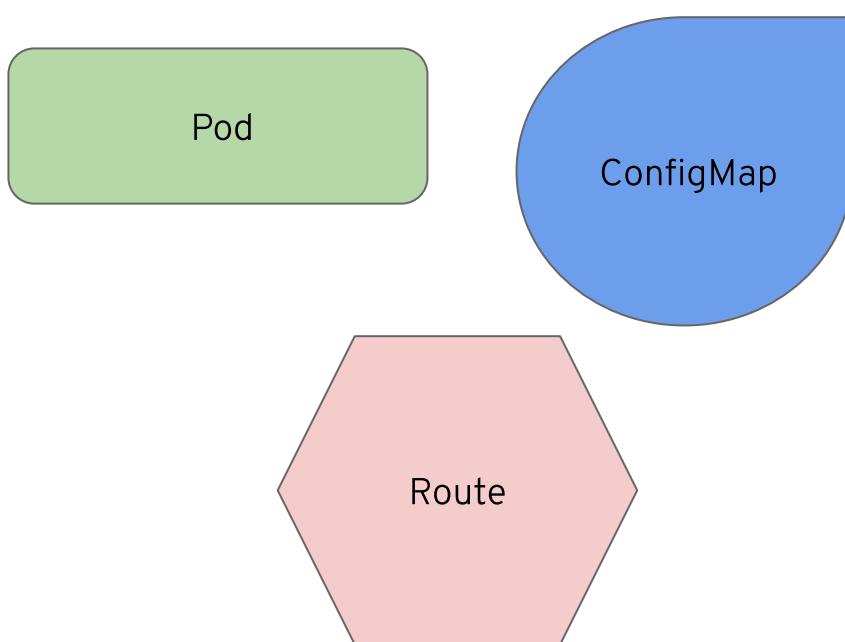
Controller

3

Knowledge

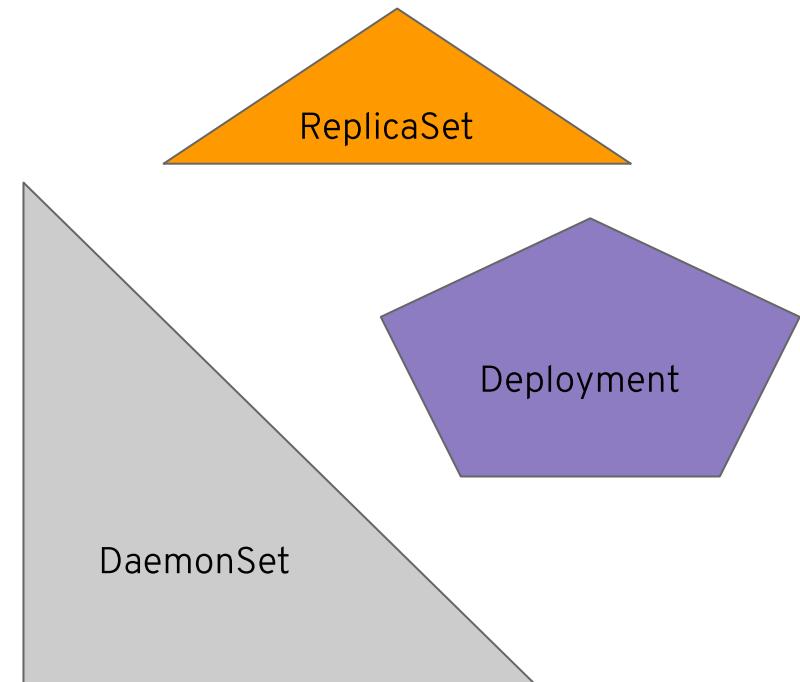
1

Resources



2

Controllers



Domain or Application Specific Knowledge?

Installing.

Self-Heal.

Scale (properly).

Clean Up.

Update.

Backup.

Restore.

etc.

Why do they matter to Red Hat?

We want an “as-a-service” platform experience.

Build an ecosystem of software on Kubernetes that can be
as easy, safe, and reliable to use and operate as a Cloud
Service.

Low-touch, remotely managed, one-click-updates.



OPERATORS IN OPENShift

Operator Hub - Allows administrators to selectively make operators available from curated sources to users in the cluster.

The screenshot shows the Red Hat OpenShift web console with the 'Operator Hub' selected in the sidebar. The main area displays a grid of operator cards. Each card includes the operator name, provider, description, and an 'Installed' status indicator (green checkmark).

All Items	All Items
Kafka	14 items
Messaging	
Middleware	
Streaming	
Datastore	
Openshift Optional	
Logging	
Aggregated	
Elasticsearch	
Kibana	
Fluentd	
Efk	
Tracing	
Monitoring	
Troubleshooting	
Distributed	
Other	

amq-streams
provided by Red Hat, Inc.
Red Hat AMQ Streams is a massively scalable, distributed, and high performance message broker for the cloud-native era.

automationbroker
provided by Red Hat, Inc.
Ansible Service Broker is an implementation of the [Open Service Broker API] (<https://github.com/open-service-broker/open-service-broker>)

cluster-logging
provided by Red Hat, Inc.
The Cluster Logging Operator for OKD provides a means for configuring and managing your aggregated logging.

couchbase-enterprise
provided by MongoDB, Inc.
The Couchbase Autonomous Operator allows users to easily deploy, manage, and maintain Couchbase.

descheduler
provided by Red Hat, Inc.
An operator to run the OpenShift descheduler

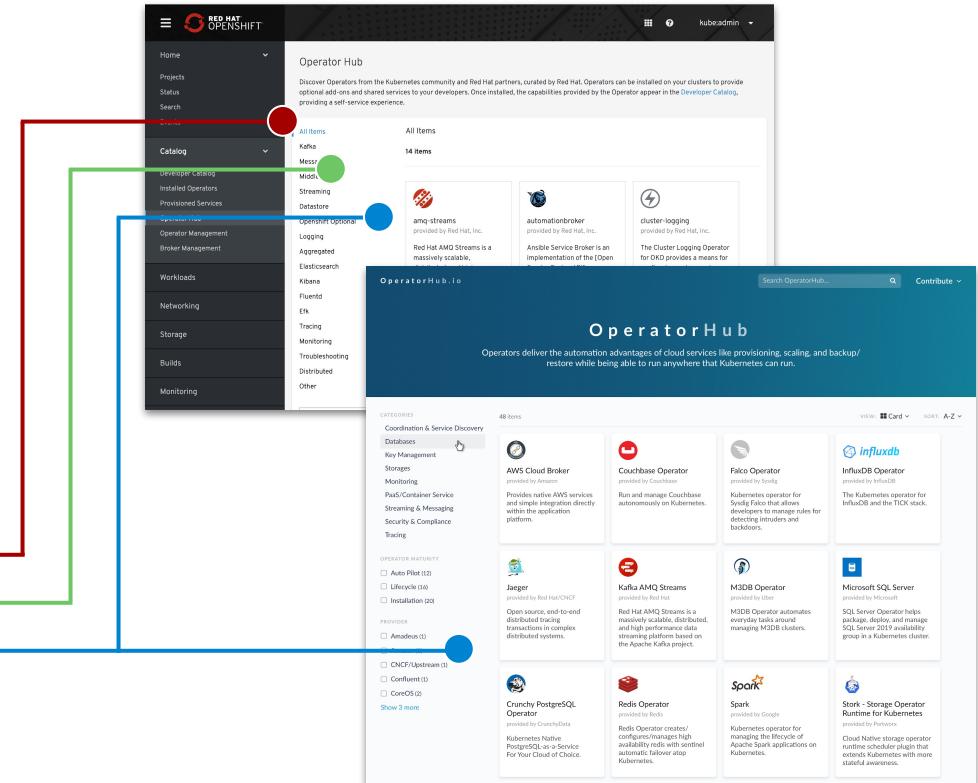
dynatrace-monitoring
provided by MongoDB, Inc.
Install full-stack monitoring of [OpenShift clusters] (<https://www.dynatrace.com/>)

OPERATORHUB

- Accessible to admins only
- Discovery/install of all optional components and apps
- Upstream and downstream offering
- ISV partners will support their Operators

TYPES OF OPERATORS

Red Hat Products
ISV Partners
Community



Red Hat Certified Operators

DEVOPS



APM



DATA SERVICES



DATABASE



SECURITY



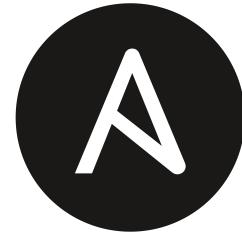
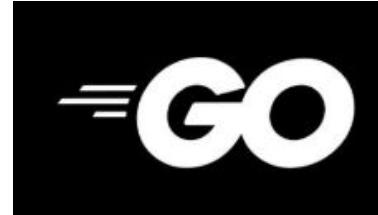
STORAGE



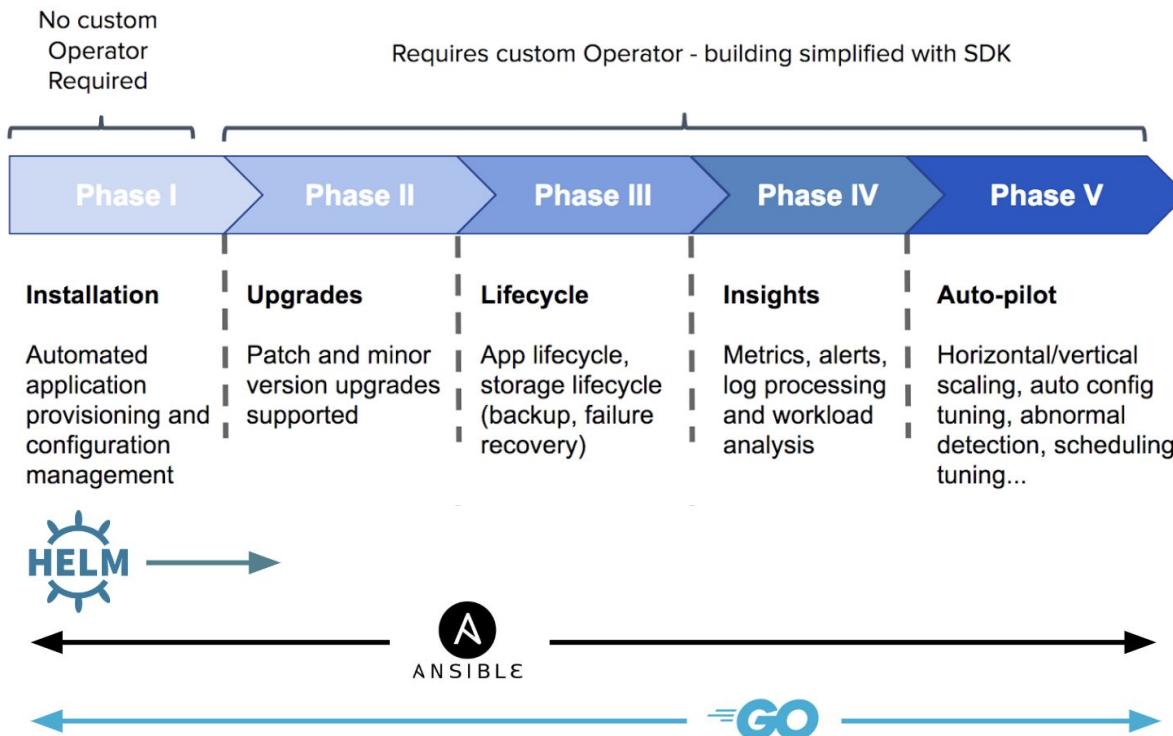


**OPERATOR
FRAMEWORK**

3



TYPES OF OPERATORS



Level 1 Install & Config	Configuration	<ul style="list-style-type: none"> • Can set configuration for installation in the CRD • Change operand config via CRD •
	Installation	<ul style="list-style-type: none"> • Operator can be installed via kubectl • Operator can be installed via OLM • Operator can be installed via Catalog Source
Level 2 Seamless Upgrades	Upgrade	<ul style="list-style-type: none"> • Operator can be upgraded seamlessly • Operand can be upgraded seamlessly •
	Downgrade	<ul style="list-style-type: none"> • Operand can be downgraded seamlessly
Level 3 Resiliency	Backup	<ul style="list-style-type: none"> • Operand can be backed up by the Operator
	Restore	<ul style="list-style-type: none"> • Operator can restore the Operand from backup
Level 4 Deep Insights	Monitoring	<ul style="list-style-type: none"> • Operand exposes monitoring metrics • Operator exposes monitoring metrics •
	Alerting	<ul style="list-style-type: none"> • Operand send useful alerts
Level 5 Auto-Pilot	Metering	<ul style="list-style-type: none"> • Operator leverages Operator Framework Metering
	K8s Events	<ul style="list-style-type: none"> • Custom Resources contain Custom K8s events
Level 5 Auto-Pilot	Auto-scaling	<ul style="list-style-type: none"> • Operator can scale the operand up / out • Operator can scale the operand down / in
	Auto-healing	<ul style="list-style-type: none"> • Operator can automatically heal an unhealthy operand
Level 5 Auto-Pilot	Auto-tuning	<ul style="list-style-type: none"> • Operator is able to tune the operand for ideal performance
	Workload Optimization	<ul style="list-style-type: none"> • Operator dynamically shifts workloads onto the best suited nodes
Level 5 Auto-Pilot	Abnormality Detection	<ul style="list-style-type: none"> • Operator is able to learn the normal performance patterns of operand • Operator is able to detect abnormalities outside of normal patterns •

The screenshot shows the OperatorHub.io interface for the etcd operator. A red box highlights the 'CHANNEL' section, which is set to 'singlenamespace-alpha'. Another red box highlights the 'CAPABILITY LEVEL' section, which includes 'Basic Install', 'Seamless Upgrades', and 'Full Lifecycle' (all checked), while 'Deep Insights' and 'Auto Pilot' are unselected.

CHANNEL
singlenamespace-alpha ▾

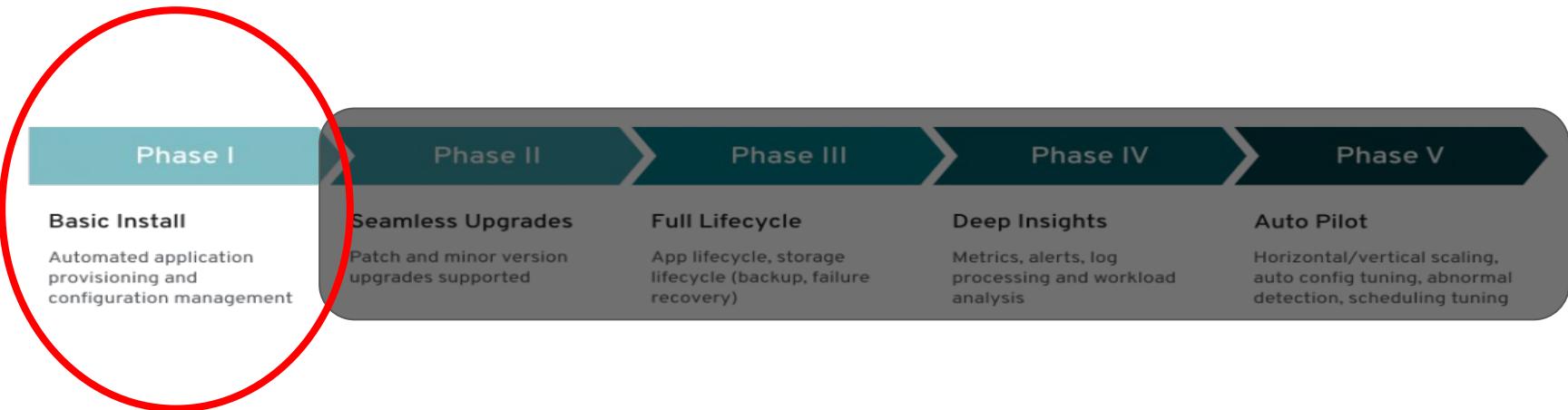
VERSION
0.9.4 (Current) ▾

CAPABILITY LEVEL ⓘ

- Basic Install
- Seamless Upgrades
- Full Lifecycle
- Deep Insights
- Auto Pilot

Let's Explore!

Level 1: Basic Install



We all know installing is pretty simple in the
Kubernetes world.

You may currently install non-Operator
Kubernetes Applications via...

```
kubectl apply -f namespace.yaml  
kubectl apply -f networkpolicy.yaml  
kubectl apply -f serviceaccount.yaml  
    kubectl apply -f role.yaml  
    kubectl apply -f rolebinding.yaml  
    kubectl apply -f configmap1.yaml  
    kubectl apply -f configmap2.yaml  
        kubectl apply -f secret1.yaml  
        kubectl apply -f secret2.yaml  
    kubectl apply -f deployment.yaml
```

```
helm install -f myvalues.yaml ./redis
```

```
oc process -f my-rails-postgresql --param-file=postgres.env
```

How do you install an Operator?

Installing an Operator



Step 1: Apply the CRD(s).

Step 2: Apply a Service Account and RBAC.

Step 3: Apply the controller via a Deployment.

Step 4: Apply the CR(s).

Installing an Operator via OLM

The screenshot shows the Crunchy PostgreSQL Enterprise operator page in the Operator Catalog. At the top left is the Crunchy logo (a blue dog icon). To its right is the title "Crunchy PostgreSQL Enterprise" and below it "4.0.1 provided by CrunchyData.com". A large blue "Install" button is centered at the top. To the left of the button, the "OPERATOR VERSION" is listed as "4.0.1". Below that, the "PROVIDER TYPE" is "Community" and the "PROVIDER" is "CrunchyData.com". On the right side, a box titled "Community Operator" contains the text: "This is a community provided operator. These are operators which have not been vetted or verified by Red Hat. Community Operators should be used with caution because their stability is unknown. Red Hat provides no support for Community Operators." Below this text is a link: "Learn more about Red Hat's third party software support policy". At the bottom of the page, a brief description states: "The PostgreSQL Operator runs within a Kubernetes cluster and provides a means to deploy and manage PostgreSQL clusters."

The screenshot shows the "Create Operator Subscription" dialog. It starts with the instruction: "Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates." Under "Installation Mode", the radio button "A specific namespace on the cluster" is selected, with the note: "Operator will be available in a single namespace only." A dropdown menu is open, showing "default" as the current selection. Below this, the "Update Channel" is set to "alpha". Under "Approval Strategy", the radio button "Automatic" is selected. On the right side, there is a "Provided APIs" section with three items: "Postgres Primary Cluster" (selected), "Postgres SQL Policy", and "Postgres backup task". At the bottom of the dialog are two buttons: "Subscribe" (highlighted with a red arrow) and "Cancel".

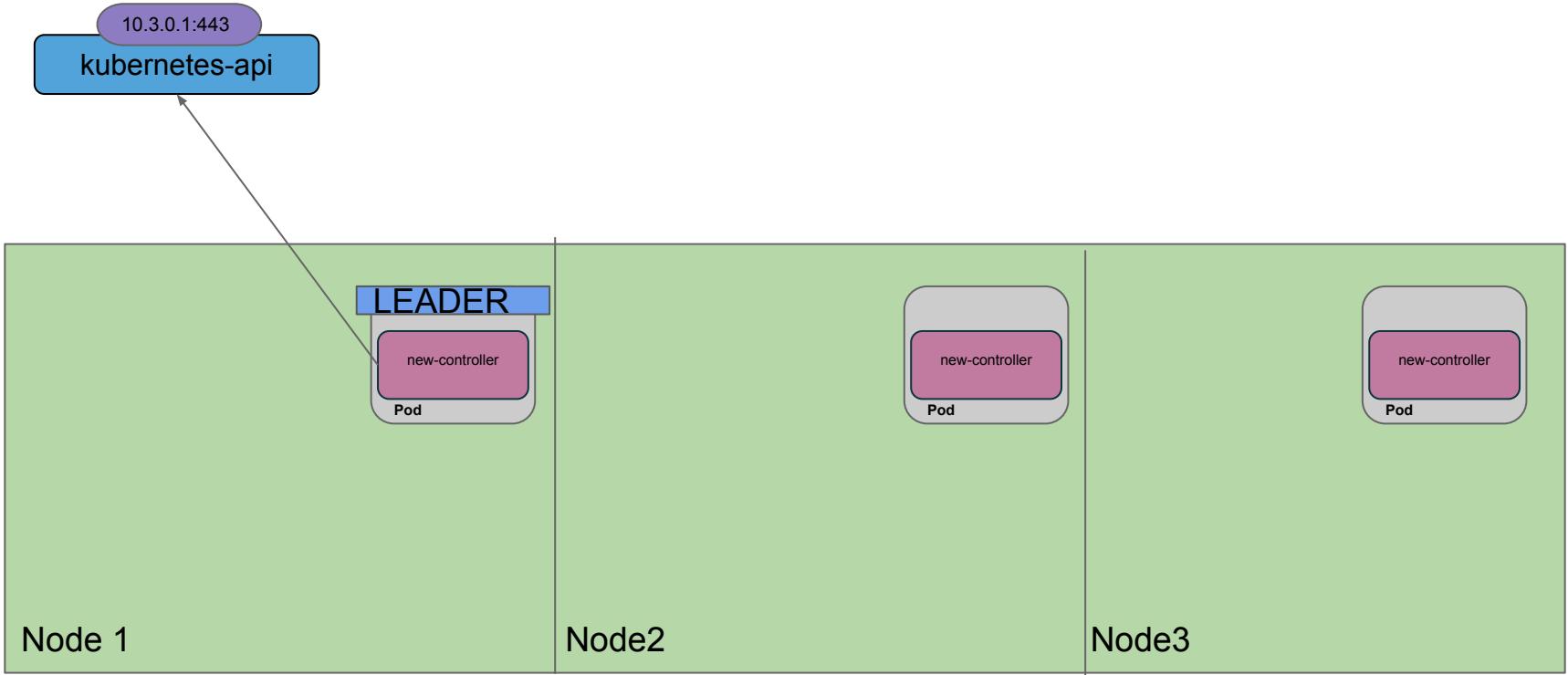


Review Resources Before Installing

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar is titled "Administrator" and includes sections for Home, Dashboards, Projects, Search, Explore, Events, Operators, OperatorHub, Installed Operators, Workloads, Networking, Storage, Builds, Monitoring, Compute, and Administration. The main content area shows a project named "openshift-operators". A message at the top states: "You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in." Below this, it says "Install Plans > Install Plan Details" for "install-2p5s6". The "Components" tab is selected. A modal window titled "Review Manual Install Plan" contains the text: "Once approved, the following resources will be created in order to satisfy the requirements for the components specified in the plan. Click the resource name to view the resource in detail." It has "Approve" and "Deny" buttons. This modal is highlighted with a red rectangle. To the right is a table listing the resources to be created:

Name	Kind	API Version	Status
postgresoperator.v4.0.0-rc10	ClusterServiceVersion	operators.coreos.com/v1alpha1	Unknown
pgbackups.crunchydata.com	CustomResourceDefinition	apiextensions.k8s.io/v1beta1	Unknown
pgclusters.crunchydata.com	CustomResourceDefinition	apiextensions.k8s.io/v1beta1	Unknown
pgpolicies.crunchydata.com	CustomResourceDefinition	apiextensions.k8s.io/v1beta1	Unknown
pgreplicas.crunchydata.com	CustomResourceDefinition	apiextensions.k8s.io/v1beta1	Unknown
pgtasks.crunchydata.com	CustomResourceDefinition	apiextensions.k8s.io/v1beta1	Unknown
postgres-operator	ServiceAccount	core/v1	Unknown
postgresoperator.v4.0.0-rc10-mng7t	Role	rbac.authorization.k8s.io/v1	Unknown
postgresoperator.v4.0.0-rc10-mng7t-postgres-operator-gw2n5	RoleBinding	rbac.authorization.k8s.io/v1	Unknown
postgresoperator.v4.0.0-rc10-mwwct	ClusterRole	rbac.authorization.k8s.io/v1	Unknown
postgresoperator.v4.0.0-rc10-mwwct-postgres-operator-fcz6j	ClusterRoleBinding	rbac.authorization.k8s.io/v1	Unknown
pg-backrest	ServiceAccount	core/v1	Unknown
postgresoperator.v4.0.0-rc10-7f6f4	Role	rbac.authorization.k8s.io/v1	Unknown
nonInresoperator.v4.0.0-rc10-7f6f4-non-backrest-nkn8n	RoleBinding	rbac.authorization.k8s.io/v1	Unknown

CRD and Controller are Applied



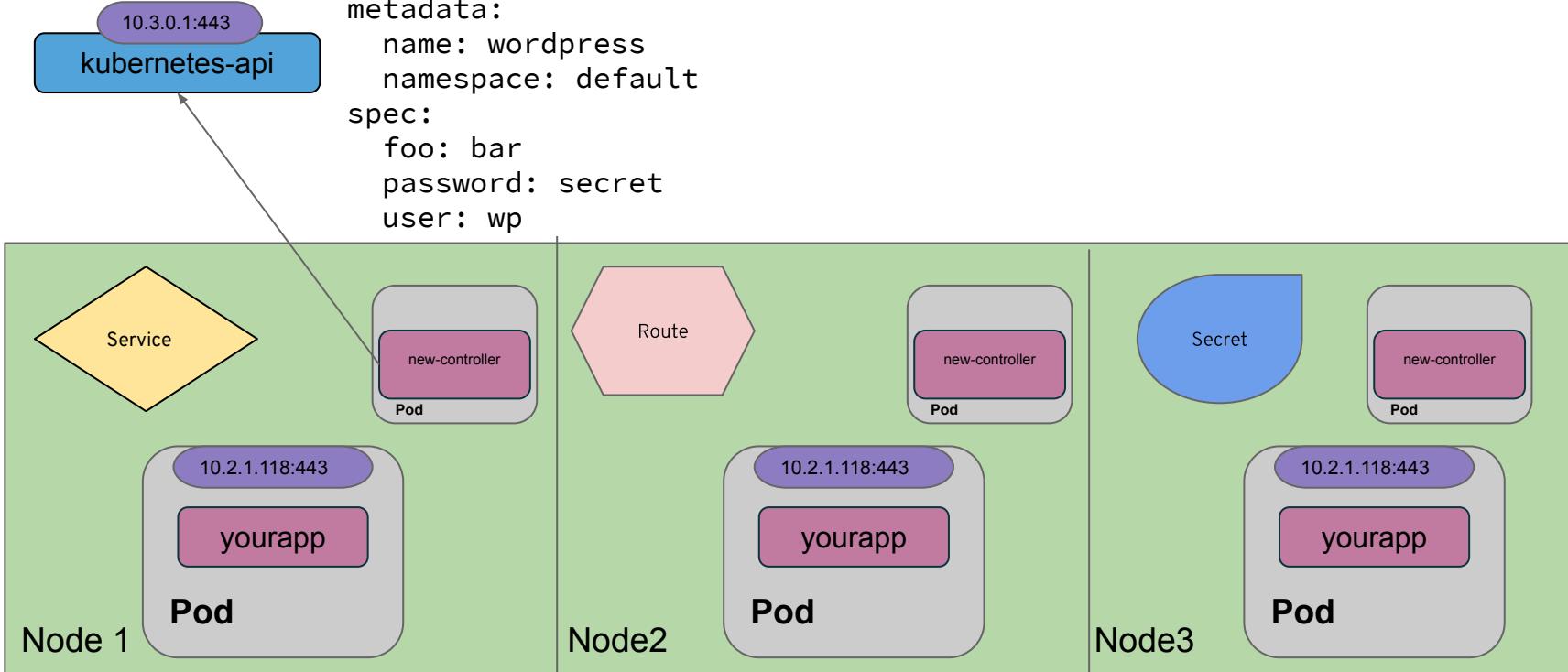
Create the CR(s)

The screenshot shows the Red Hat OpenShift Container Platform web interface. On the left, a sidebar menu includes sections like Administrator, Home, Operators (selected), OperatorHub, Installed Operators (selected), Workloads, Networking, Storage, Builds, Monitoring, Compute, and Administration. The main content area is titled "Project: testing" and shows the "Couchbase Operator > Create Couchbase Cluster" page. A modal window is open, prompting for YAML or JSON definitions. The modal contains fields for "Auto Failover Server Group" (set to False), "Auth Secret" (a dropdown menu listing several secrets, with "builder-dockercfg-qcsm" selected), "External Web Console" (set to False), "Anti Affinity" (set to False), "Show Update Notifications" (set to False), and "Disable Bucket Management". The modal also includes a note about exposing the Couchbase Server Web Console externally. The background shows the "Create Couchbase Cluster" form with a large code editor containing the following YAML configuration:

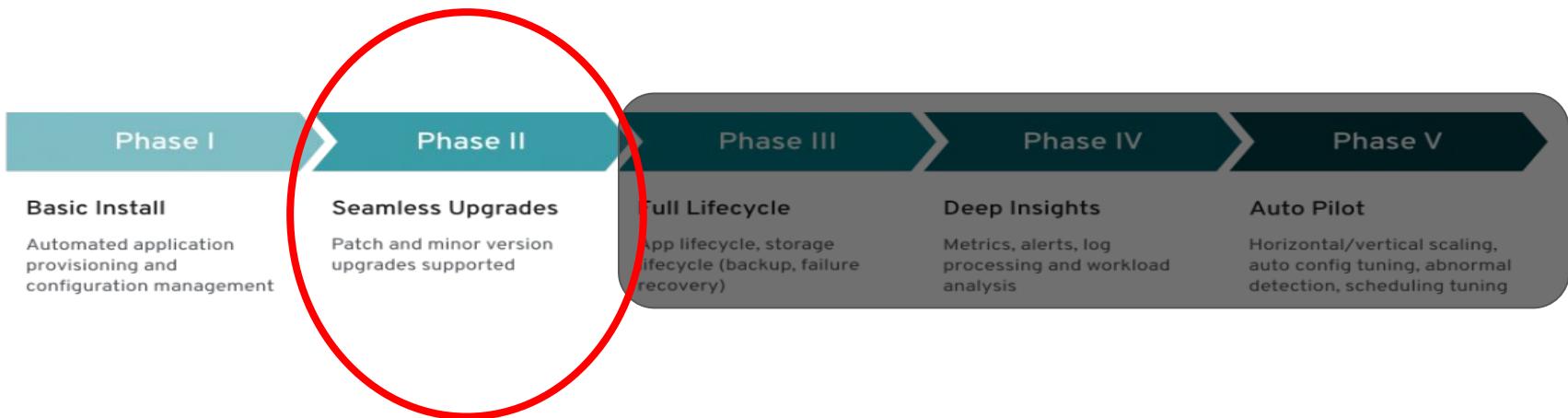
```
1  apiVersion: couchbase.com/v1
2  kind: CouchbaseCluster
3  metadata:
4    name: cb-example
5    namespace: testing
6  spec:
7    authSecret: cb-example-auth
8    baseImage: registry.connect.redhat.com/
9    buckets:
10      - conflictResolution: seqno
11        enableFlush: true
12        evictionPolicy: fullEviction
13        ioPriority: high
14        memoryQuota: 128
15        name: default
16        replicas: 1
17        type: couchbase
18        cluster:
19          analyticsServiceMemoryQuota: 1024
20          autoFailoverMaxCount: 3
21          autoFailoverOnDataDiskIssues: true
22          autoFailoverOnDataDiskIssuesTimePeriod: 10m
23          autoFailoverServerGroup: false
24          autoFailoverTimeout: 120
25          clusterName: cb-example
26          dataServiceMemoryQuota: 256
27          eventingServiceMemoryQuota: 256
28          indexServiceMemoryQuota: 256
29          indexStorageSetting: memory_optimized
30          searchServiceMemoryQuota: 256
```

CR is Applied

```
apiVersion: db.example.com/v1
kind: MySql
metadata:
  name: wordpress
  namespace: default
spec:
  foo: bar
  password: secret
  user: wp
```



Level 2: Seamless Upgrades



Operator Upgrades

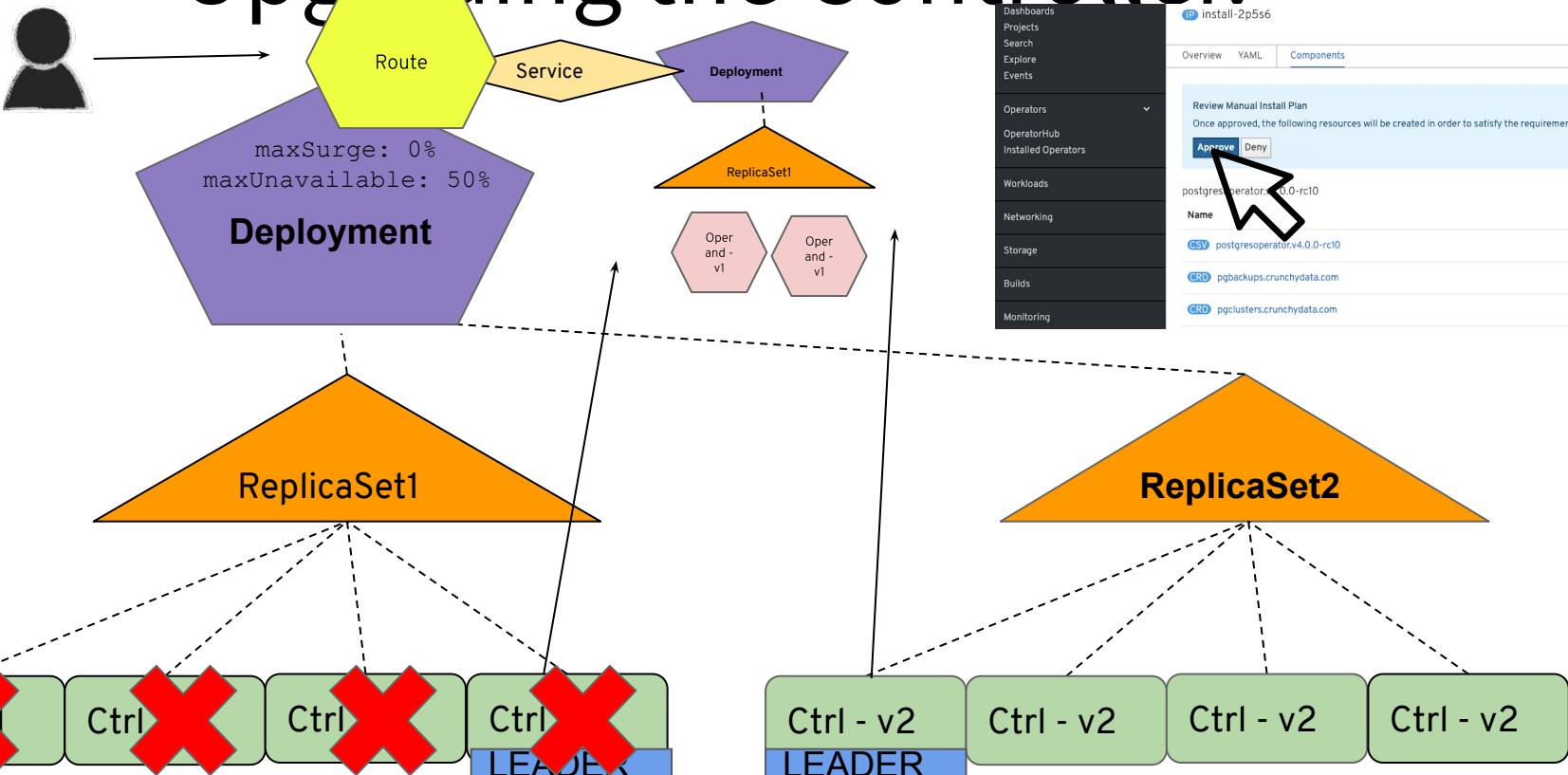
3

Upgrading the Controller.

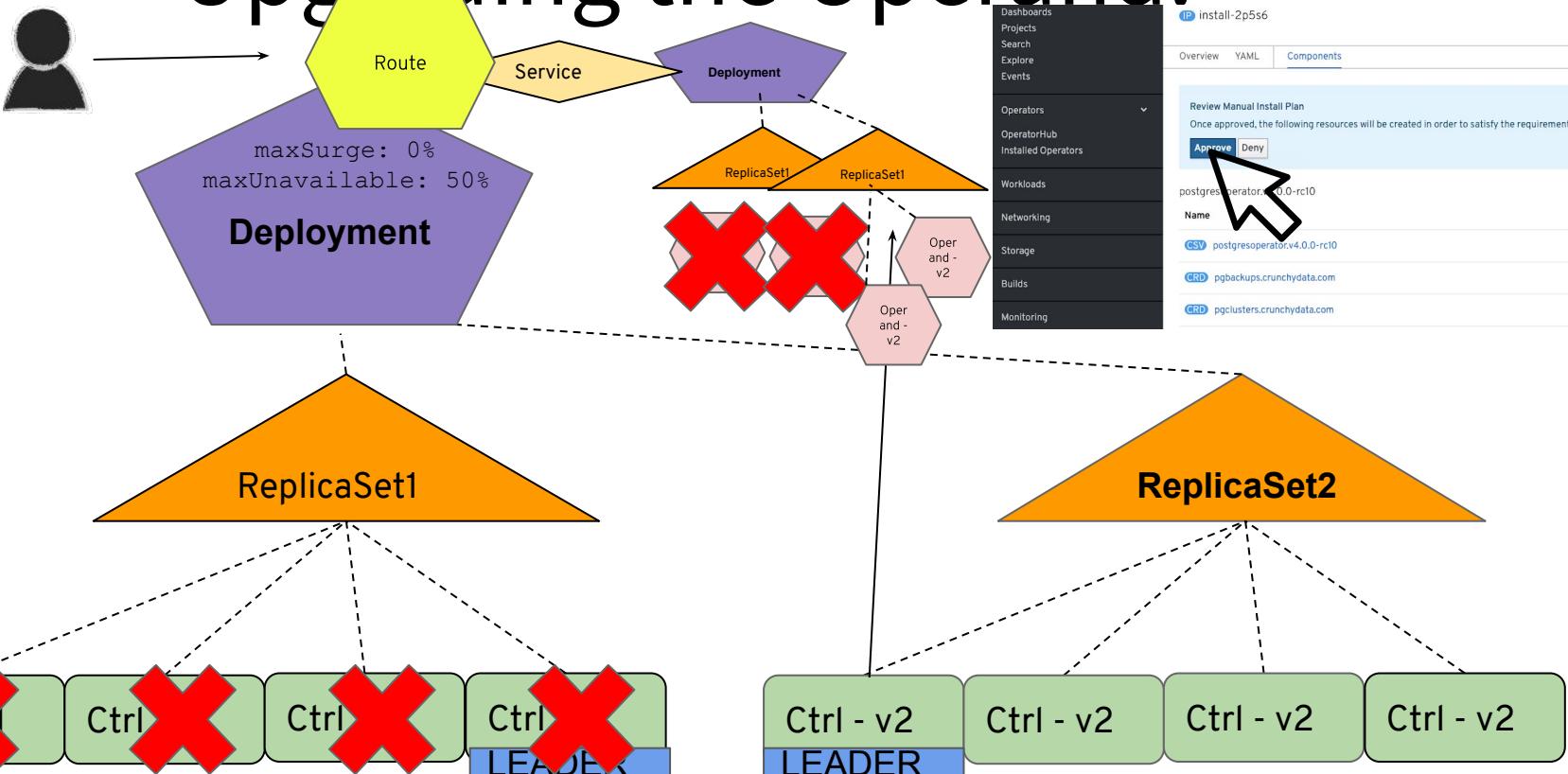
Upgrading the Operand.

Upgrading the CRD version/schema.

Upgrading the Controller.



Upgrading the Operand.



Upgrading the CRD

```
apiVersion: stable.example.com/v1alpha1
kind: CronTab
metadata:
  name: cr1
  namespace: default
spec:
  hostPort: localhost:8080
```

```
apiVersion: stable.example.com/v1alpha2
kind: CronTab
metadata:
  name: cr1
  namespace: default
spec:
  host: localhost
  port: 8080
```

```
Kind: CustomResourceDefinition
metadata:
  name: crontabs.stable.example.com
spec:
  group: stable.example.com
  versions:
    - name: v1alpha1
      served: true
      storage: true
  scope: Namespaced
  names:
    plural: crontabs
    singular: crontab
    kind: CronTab
    shortNames:
      - ct
  validation:
    openAPIV3Schema:
      type: object
      properties:
        spec:
          type: object
          properties:
            hostPort:
              type: string
```



Upgrading the CRD

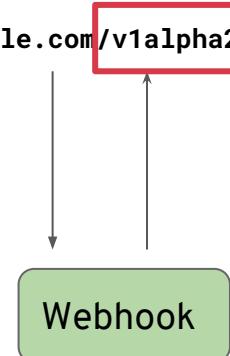


OPERATOR
LIFECYCLE
MANAGER

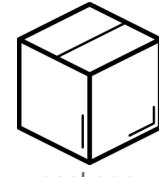
```
Kind: CustomResourceDefinition
metadata:
  name: crontabs.stable.example.com
spec:
...
- name: v1alpha1
  served: true
  storage: true
scope: Namespaced
names:
  plural: crontabs
  singular: crontab
  kind: CronTab
  shortNames:
    - ct
- name: v1alpha2
  served: true
  storage: true
schema:
  openAPIV3Schema:
    type: object
    properties:
      spec:
        type: object
        properties:
          host:
            type: string
          port:
            type: string
```

/apis/stable.example.com/v1alpha2/namespaces/default/crontabs/cr1

Ctrl - v2



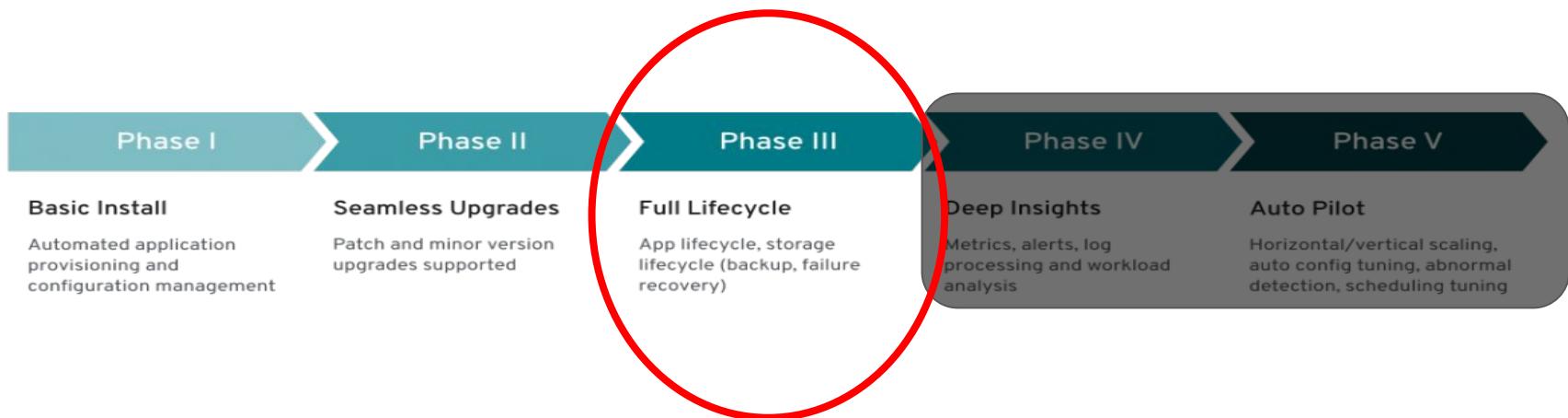
```
apiVersion:
stable.example.com/v1alpha2
kind: CronTab
metadata:
  name: cr1
  namespace: default
spec:
  host: localhost
  port: 8080
```



metadata

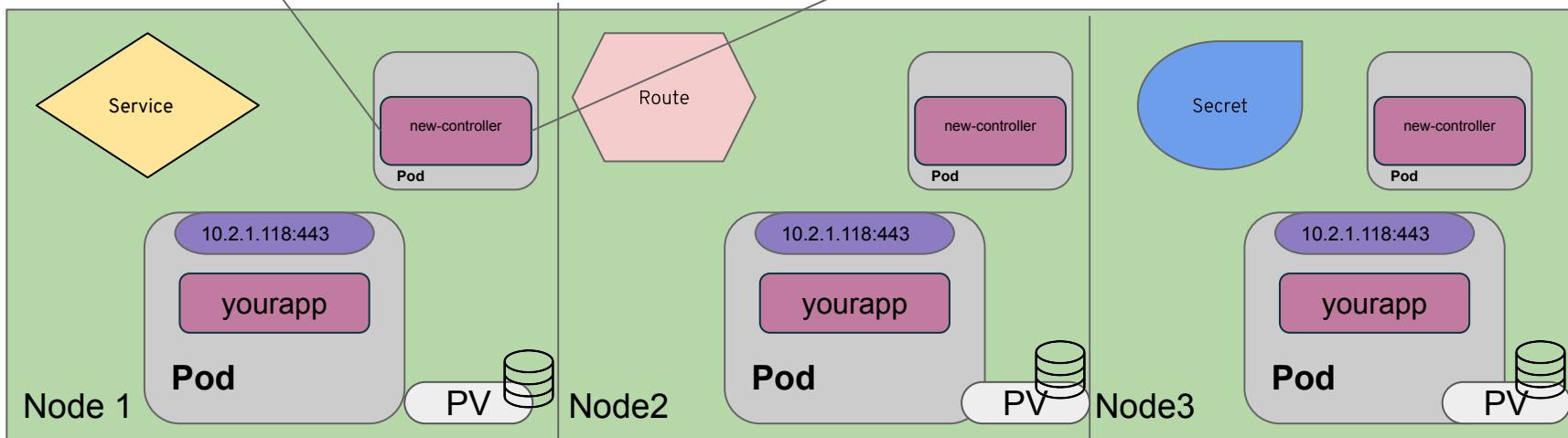
package

Level 3: Full Lifecycle (Backup & Restore)



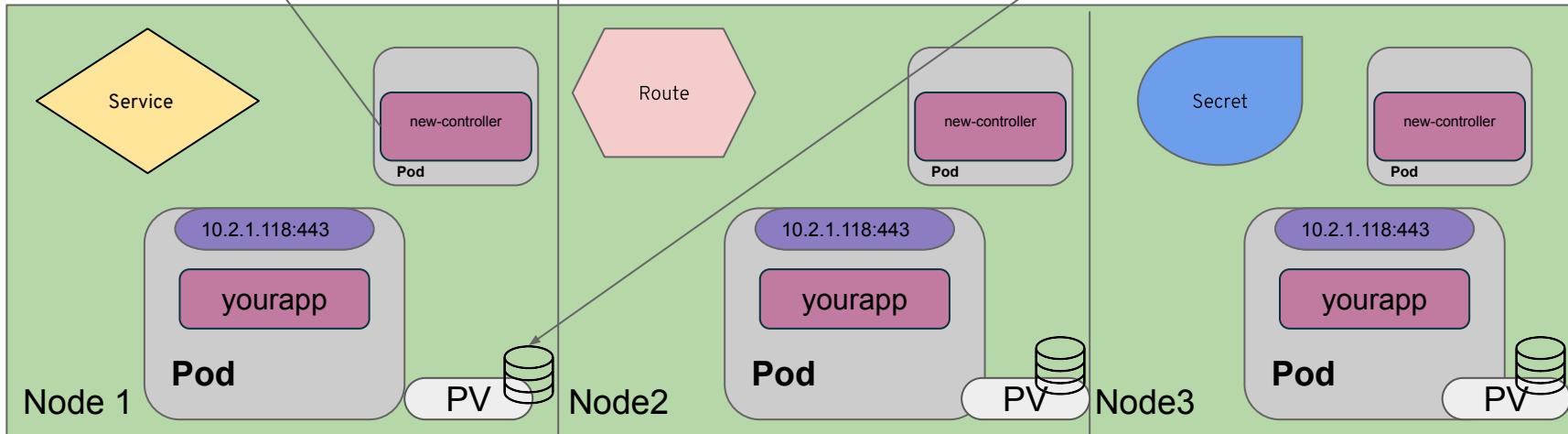
Operator Backup

```
apiVersion: etcd.database.coreos.com/v1beta2
kind: EtcdBackup
metadata:
  name: example-etcd-cluster-backup
spec:
  etcdEndpoints: http://example-etcd-cluster-client:2379
  storageType: S3
  s3: mybucket/etcd.backup
  path: http://s3.aws-region.amazonaws.com
  awsSecret: mysecret
```

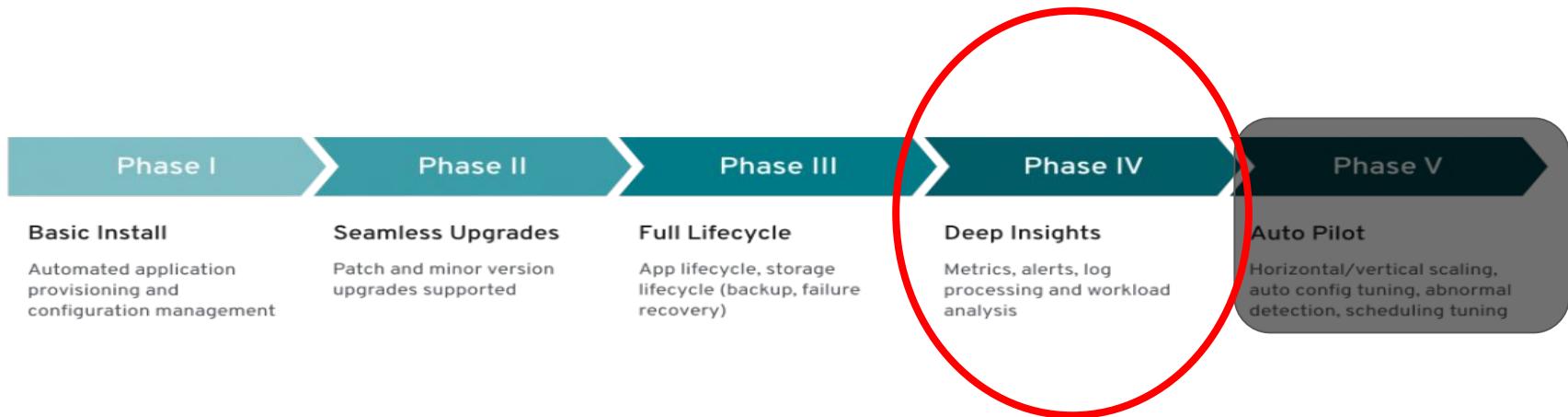


Operator Restore

```
apiVersion: etcd.database.coreos.com/v1beta2
kind: EtcdrRestore
metadata:
  name: example-etcd-cluster-restore
spec:
  etcdCluster: example-etcd-cluster
  backupStorageType: s3
  storageType: S3
  s3:
    bucket: mybucket
    path: http://s3.amazonaws.com/etcd.backup
    awsSecret: mysecret
```



Level 4: Deep Insights



Deep Insights = Observability



Observability = Metrics, Logs, Kubernetes Events,
Tracing, Profiling, etc.



Prometheus



Grafana

Level 4: Metrics!

TIP!

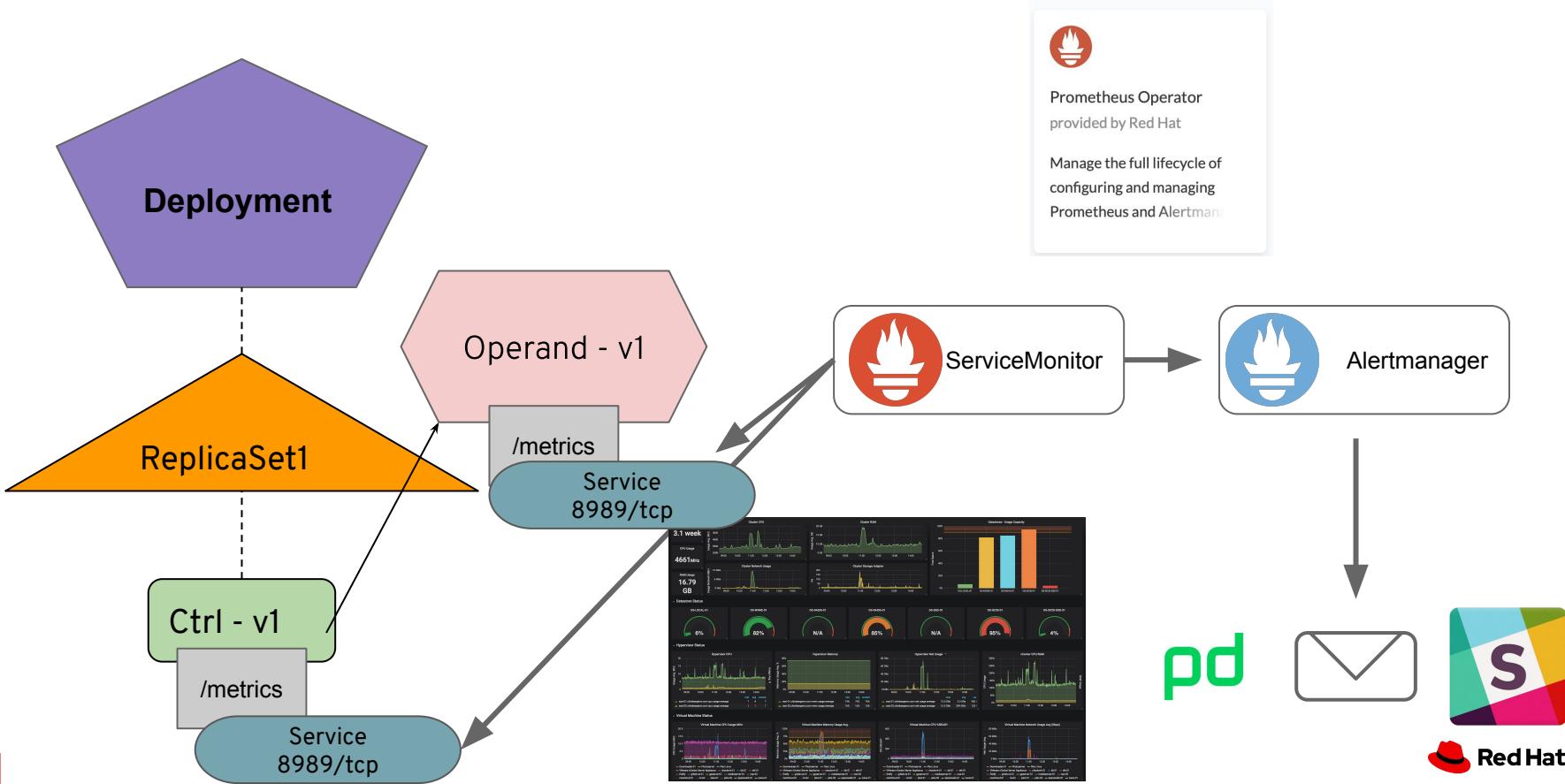
Always register a Prometheus /metrics endpoint for your Operator and Operand.

Operator-SDK does this for you and will automatically give you some default metrics including kube-state-metrics (8383/metrics)

Operator-SDK will expose that endpoint via a Service and allows you to easily create a ServiceMonitor for Prometheus Operator.

You should be utilizing the Prometheus client library to instrument your application with metrics specific to your application. Check out the Prometheus docs.

Operator and Operand Metric Endpoints



```
oc describe mycr example
```

```
.
```

```
.
```

```
.
```

```
.
```

```
.
```

Events: <none>



oc describe etcdcluster example

•
•
•
•
•

Events:

Type	Reason	Age	From	Message
Normal	New Member Added	38s	etcd-operator-594fb565f-r59nh	New member example-p2knrr94d5 added to cluster
Normal	New Member Added	22s	etcd-operator-594fb565f-r59nh	New member example-kx2k2hhft9 added to cluster
Normal	New Member Added	6s	etcd-operator-594fb565f-r59nh	New member example-b2nxkck495 added to cluster



Packages to Write Events

operator-sdk ----->

GoDoc Home About Search

client-go: k8s.io/client-go/tools/record

Index | Files | Directories

package record

import "k8s.io/client-go/tools/record"

Package record has all client logic for recording and reporting "k8s.io/api/core/v1".Event events.

Index

- func EventAggregatorByReasonFunc(event *v1.Event) (string, string)
- func EventAggregatorByReasonMessageFunc(event *v1.Event) string
- type CorrelatorOptions
- type EventAggregator
 - func NewEventAggregator(lruCacheSize int, keyFunc EventAggregatorKeyFunc, messageFunc EventAggregatorMessageFunc, maxEvents int, maxIntervalInSeconds int, clock clock.Clock) *EventAggregator
 - func (e *EventAggregator) EventAggregate(newEvent *v1.Event) (*v1.Event, string)
- type EventAggregatorKeyFunc
- type EventAggregatorMessageFunc
- type EventBroadcaster
 - func NewBroadcaster() EventBroadcaster
 - func NewBroadcasterForTests(sleepDuration time.Duration) EventBroadcaster
 - func NewBroadcasterWithCorrelatorOptions(options CorrelatorOptions) EventBroadcaster
- type EventCorrelateResult
- type EventCorrelator
 - func NewEventCorrelator(clock clock.Clock) *EventCorrelator
 - func NewEventCorrelatorWithOptions(options CorrelatorOptions) *EventCorrelator
 - func (c *EventCorrelator) EventCorrelate(newEvent *v1.Event) (*EventCorrelateResult, error)
 - func (c *EventCorrelator) UpdateState(event *v1.Event)
- type EventFilterFunc
- type EventRecorder

<----- client-go

GoDoc Home About Search

controller-runtime: sigs.k8s.io/controller-runtime/pkg/recorder Index | Examples | Files

package recorder

import "sigs.k8s.io/controller-runtime/pkg/recorder"

Package recorder defines interfaces for working with Kubernetes event recorders.

You can use these to emit Kubernetes events associated with a particular Kubernetes object.

Example (Event)
Example (Eventf)
Example (PastEventf)

Index

type Provider

Examples

package (Event)
package (Eventf)
package (PastEventf)

Package Files

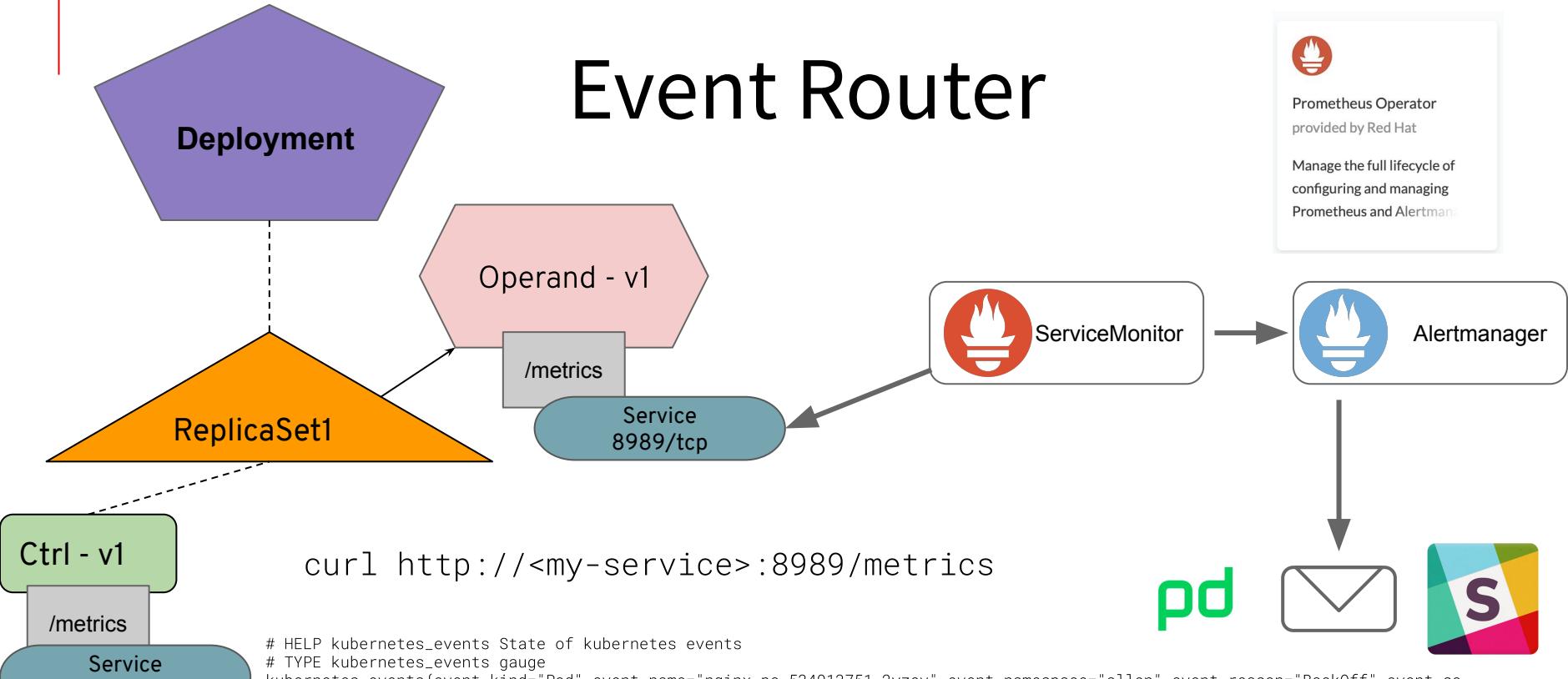
recorder.go

type Provider

```
type Provider interface {
    // NewRecorder returns an EventRecorder with given name.
    GetEventRecorderFor(name string) record.EventRecorder
}
```

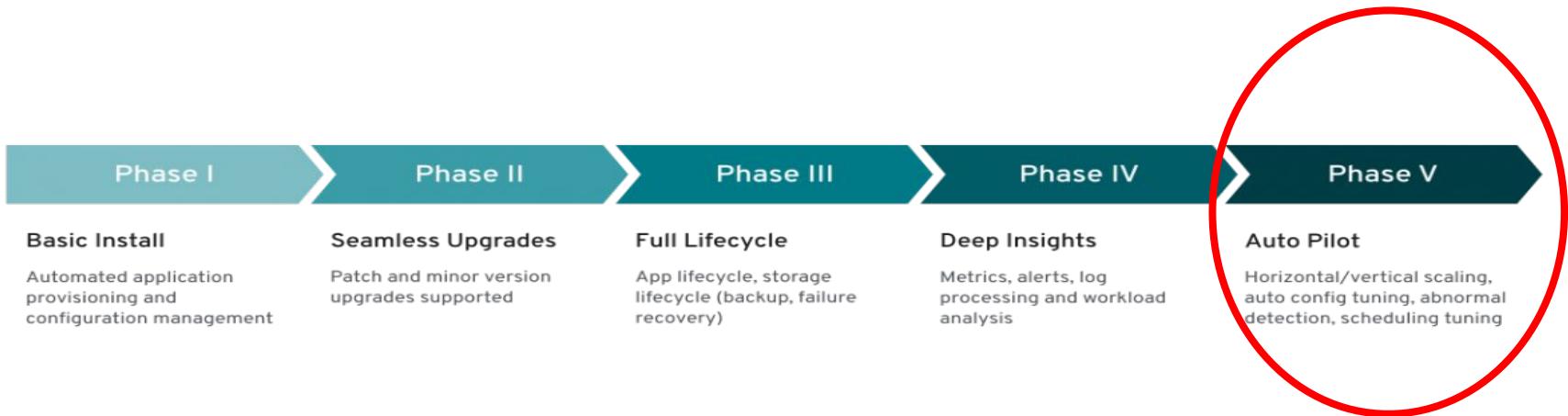


Event Router



```
# HELP kubernetes_events State of kubernetes events
# TYPE kubernetes_events gauge
kubernetes_events{event_kind="Pod",event_name="nginx-pc-534913751-2yzev",event_namespace="allen",event_reason="BackOff",event_source="kube-node-3/kubelet",event_subobject="spec.containers{nginx}",event_type="Normal"} 1
kubernetes_events{event_kind="Pod",event_name="nginx-pc-534913751-2yzev",event_namespace="allen",event_reason="Failed",event_source="kube-node-3/kubelet",event_subobject="spec.containers{nginx}",event_type="Warning"} 0
kubernetes_events{event_kind="Pod",event_name="nginx-pc-534913751-2yzev",event_namespace="allen",event_reason="FailedSync_ErrorImagePull",event_source="kube-node-3/kubelet",event_subobject="",event_type="Warning"} 0
```

Level 5: AutoPilot



Level 5: AutoPilot!!!

- **AUTO-SCALE**

- Operator can scale the Operand based of:
 - CPU/Memory (Horizontal Pod Autoscaler, Vertical Pod Autoscaler)
 - # of schedulable nodes/cores (Cluster Proportional Autoscaler)
 - Custom Metrics (HPA + Prometheus Adapter)

- **AUTO-HEAL**

- Operator runs a series of health checks and customized resolutions against operand (beyond the native liveness/readiness probes).

- **AUTO-TUNE**

- Query metrics-server and Prometheus to tune Operand for ideal performance.

Get Your Kubernetes Operator
Red Hat Certified and Featured in the
OpenShift OperatorHub Today!

THANK YOU!