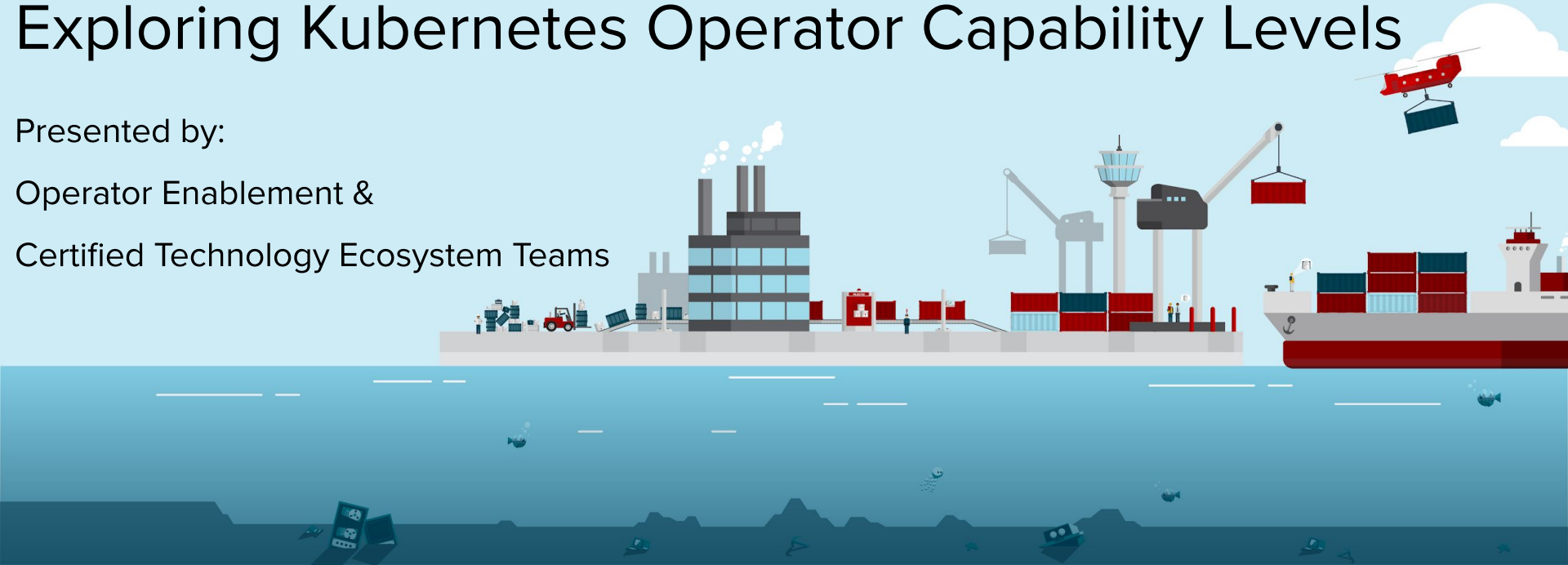
Red Hat

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?

Red Hat

What is an Operator?

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

Red Hat

try.openshift.com

Red Hat

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

OpenShift & Kubernetes

certificates & security settings

container runtime config

allowed maintenance windows

software defined networking

AUTOMATED OPERATIONS VIA OPERATORS

KUBERNETES

RHEL or RHEL CoreOS

kernel modules

device drivers

network interfaces

security groups

Nodes & Operating System

Red Hat

Yeah..but what is it?

Operators

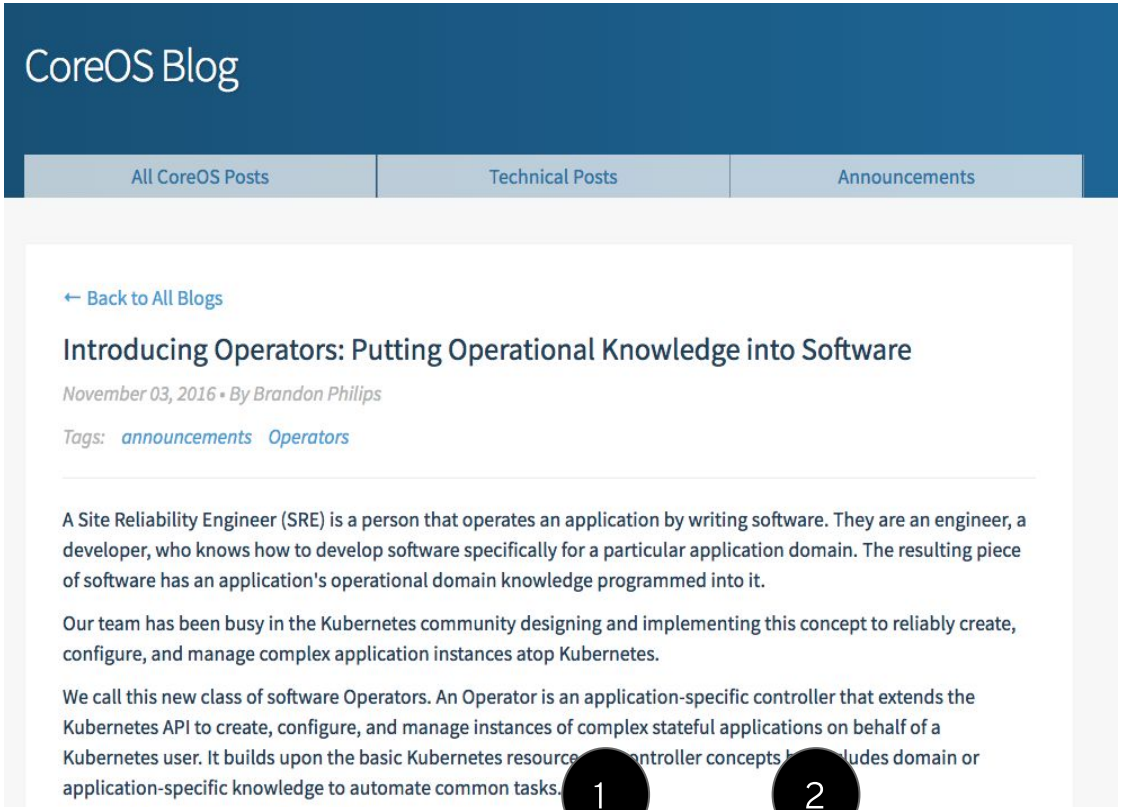
An Operator represents human operational knowledge in software, to reliably manage an application.

Red Hat

Still Confused?

Red Hat

Let’s go back 3 years ago.



3

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

Red Hat

1

(Custom) Resource

2

Controller

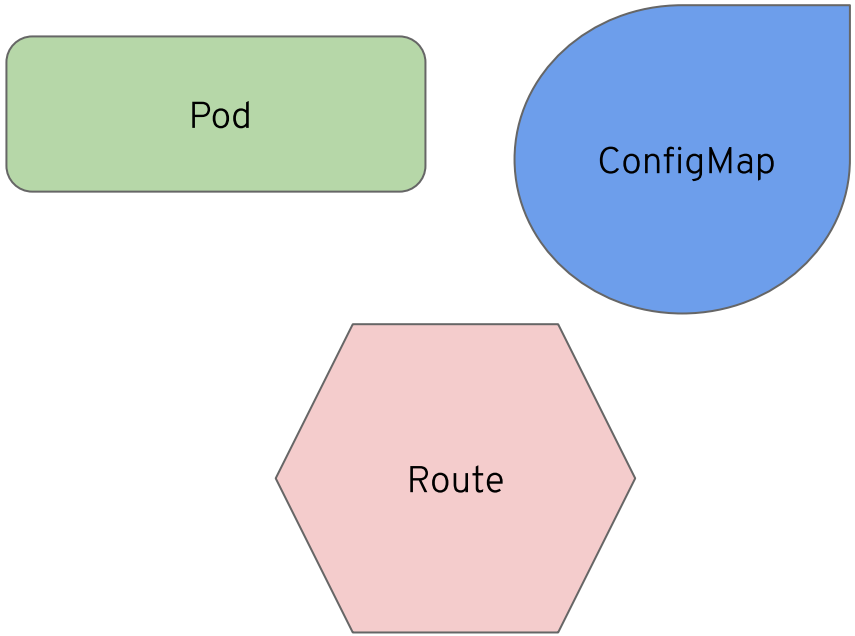
3

Knowledge

Red Hat

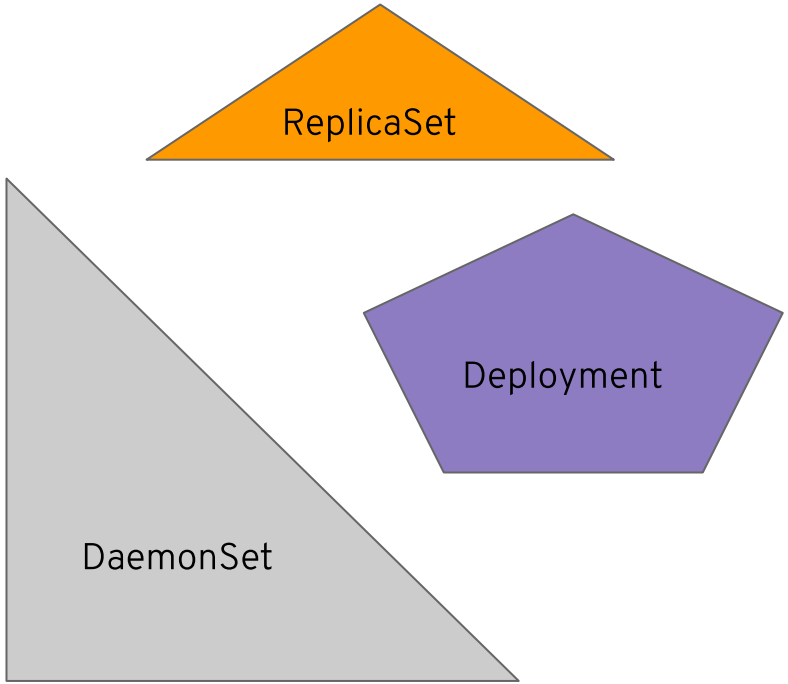
1

Resources



2

Controllers



Red Hat

3

Domain or Application Specific Knowledge?

Installing.

Scale (properly).

Self-Heal.

Clean Up.

Update.

Backup.

Restore.

etc.

Red Hat

Why do they matter to Red Hat?

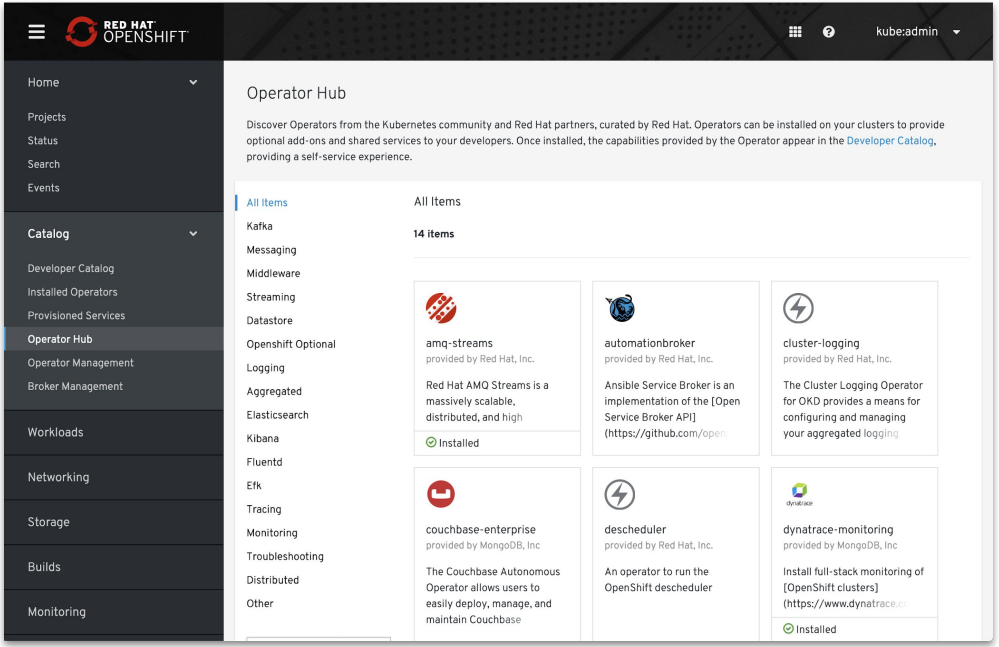
We want an “as-a-service” platform experience.

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.



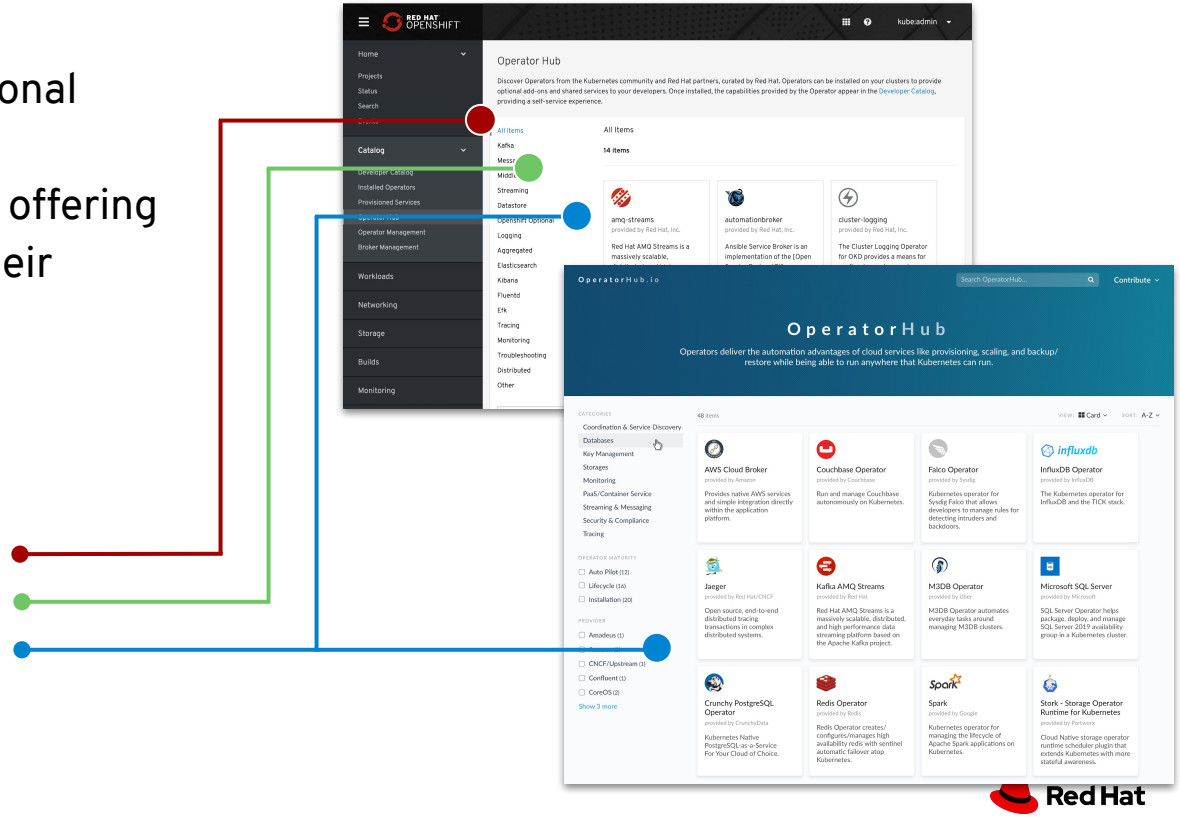
Red Hat

OPERATORHUB

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

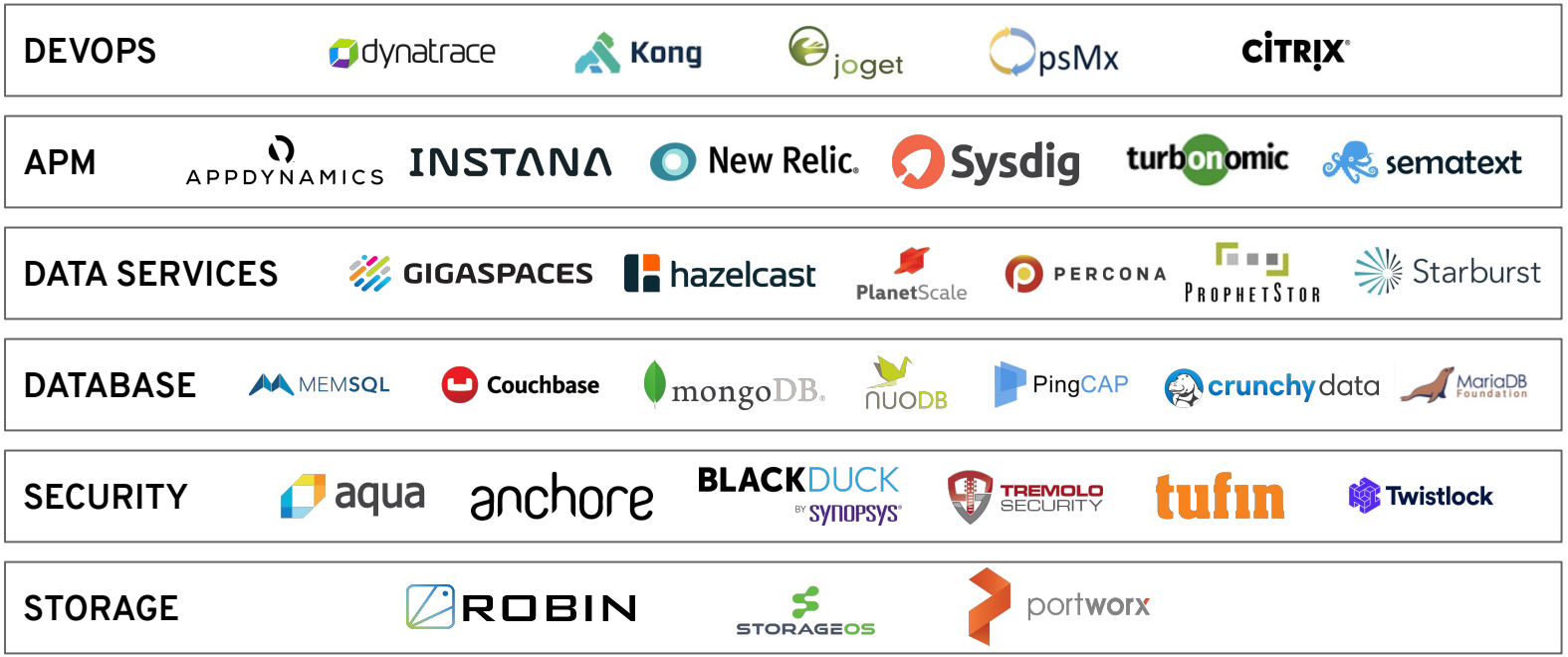
TYPES OF OPERATORS

Red Hat Products
ISV Partners
Community



Red Hat

Red Hat Certified Operators



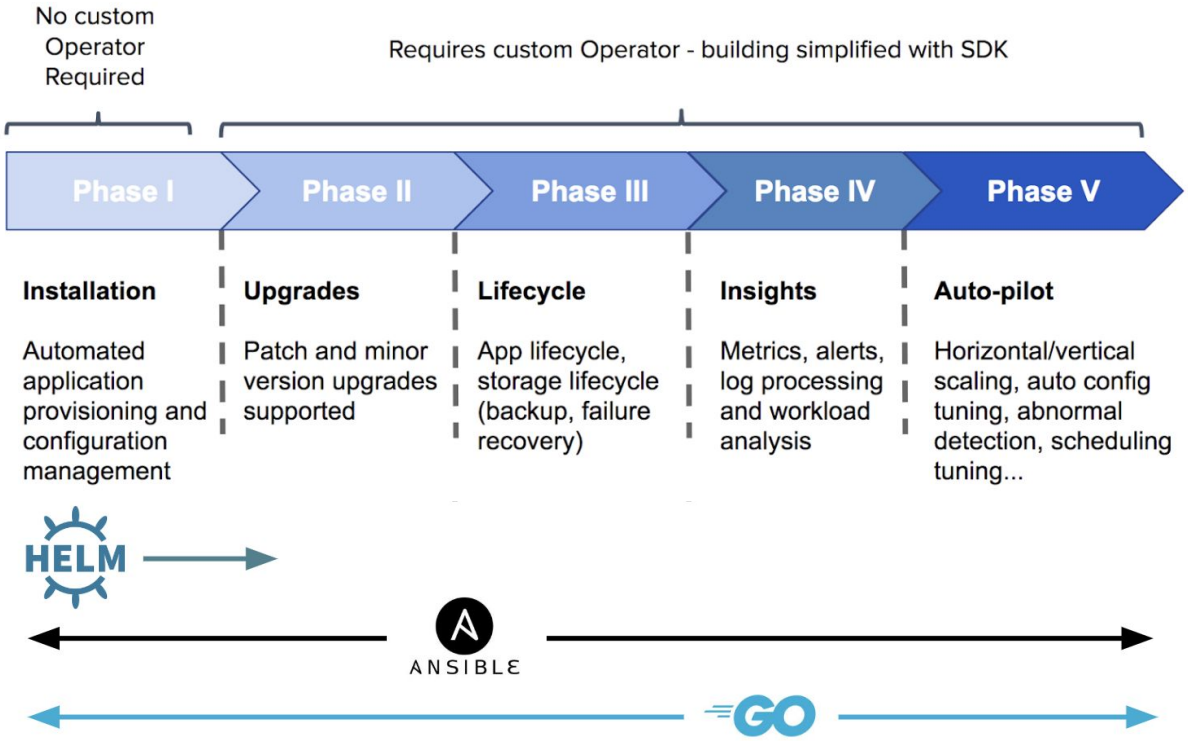
OPERATOR FRAMEWORK

Red Hat

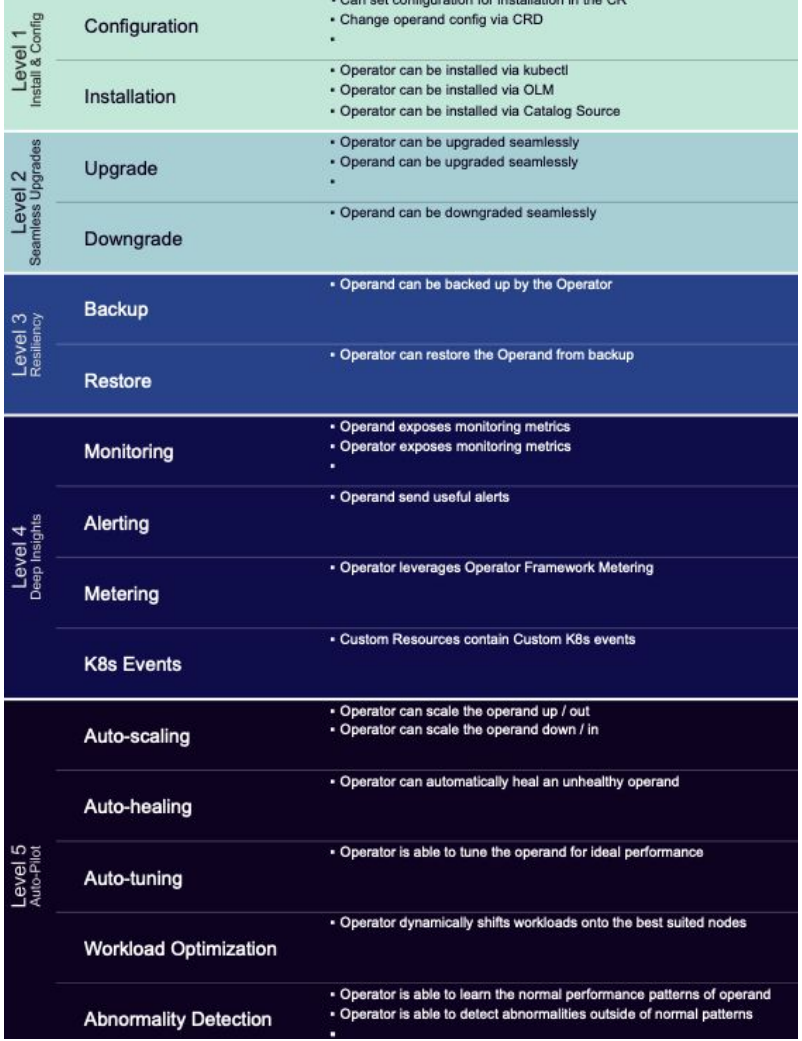
3

Red Hat

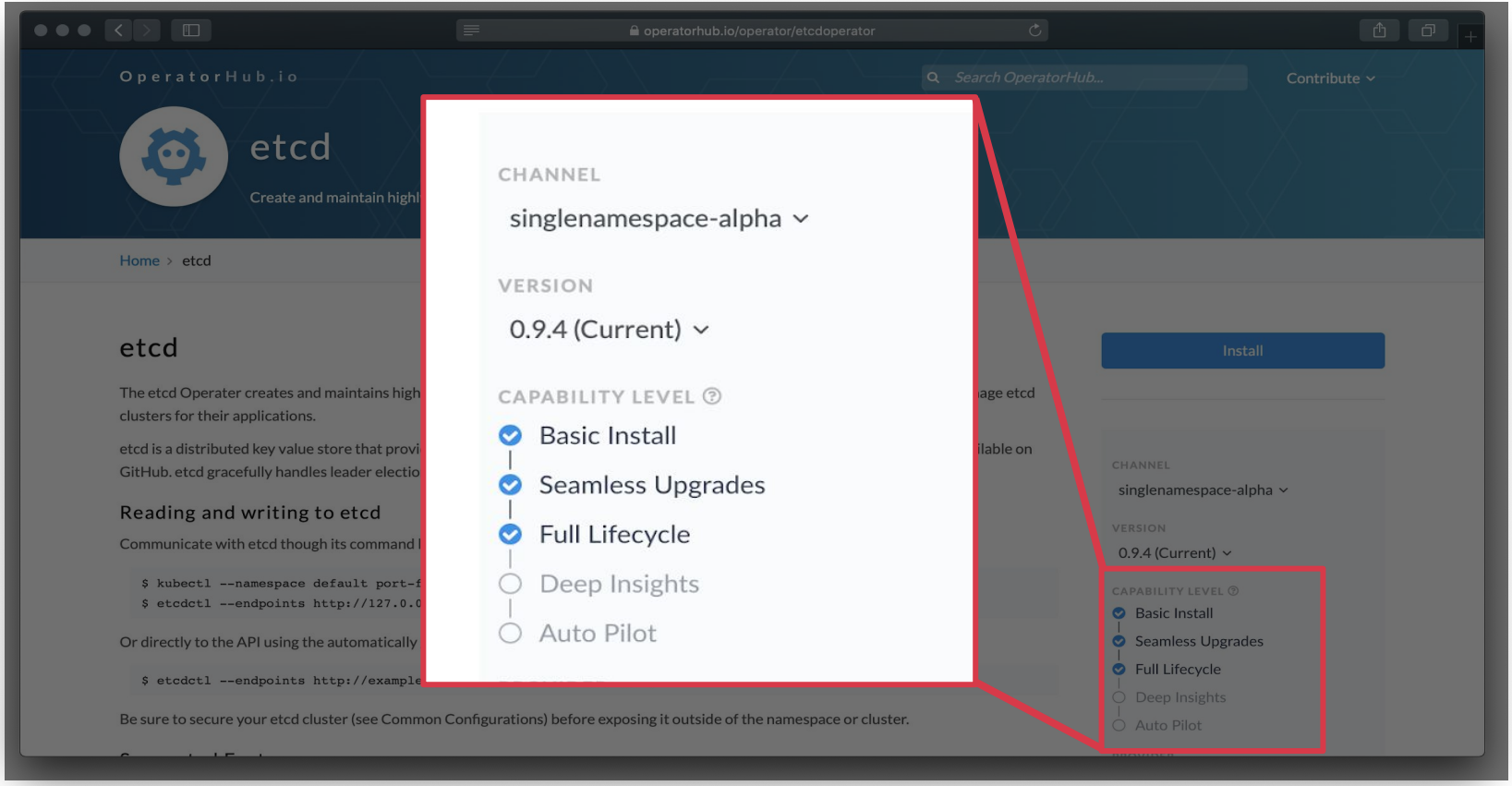
TYPES OF OPERATORS



Red Hat


Level 1
Install & Config
Configuration
• Change operand config via CRD
Installation
• Operator can be installed via kubectl
• Operator can be installed via OLM
• Operator can be installed via Catalog Source
Level 2
Seamless Upgrades
Upgrade
• Operator can be upgraded seamlessly
• Operand can be upgraded seamlessly
Downgrade
• Operand can be downgraded seamlessly
Level 3
Resiliency
Backup
• Operand can be backed up by the Operator
Restore
• Operator can restore the Operand from backup
Level 4
Deep Insights
Monitoring
• Operand exposes monitoring metrics
• Operator exposes monitoring metrics
Alerting
• Operand send useful alerts
Metering
• Operator leverages Operator Framework Metering
K8s Events
• Custom Resources contain Custom K8s events
Level 5
Auto-Pilot
Auto-scaling
• Operator can scale the operand up / out
• Operator can scale the operand down / in
Auto-healing
• Operator can automatically heal an unhealthy operand
Auto-tuning
• Operator is able to tune the operand for ideal performance
Workload Optimization
• Operator dynamically shifts workloads onto the best suited nodes
Abnormality Detection
• Operator is able to learn the normal performance patterns of operand
• Operator is able to detect abnormalities outside of normal patterns

Red Hat

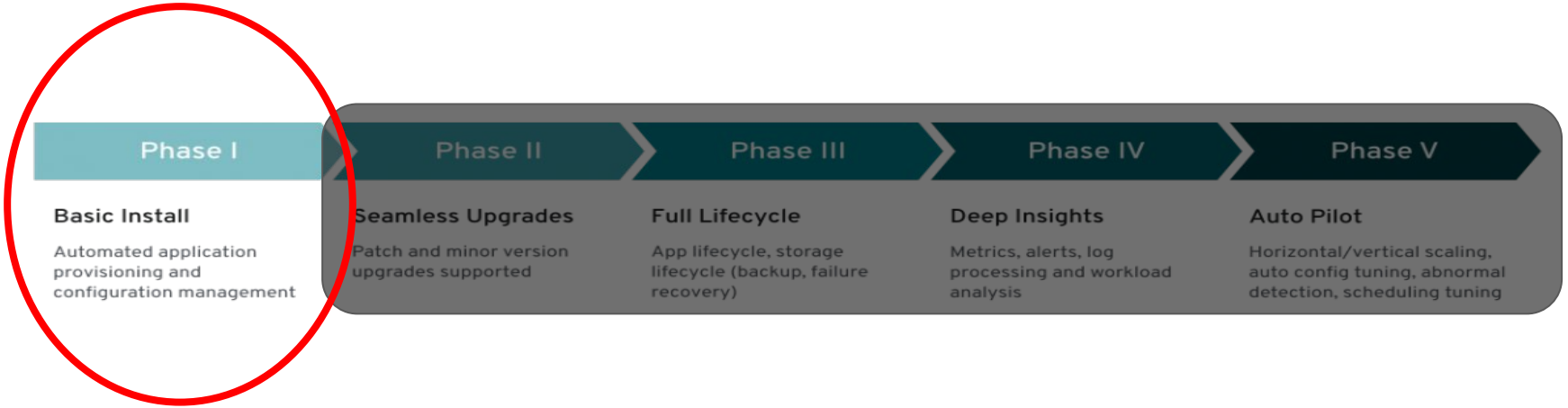

operatorhub.io/operator/etcdoperator
OperatorHub.io
Search OperatorHub...
Contribute
etcd
Create and maintain high
Home > etcd
etcd
The etcd Operater creates and maintains high
clusters for their applications.
etcd is a distributed key value store that provi
GitHub. etcd gracefully handles leader electio
Reading and writing to etcd
Communicate with etcd though its command l
$ kubectl --namespace default port-f
$ etcdctl --endpoints http://127.0.0
Or directly to the API using the automatically
$ etcdctl --endpoints http://example
Be sure to secure your etcd cluster (see Common Configurations) before exposing it outside of the namespace or cluster.
CHANNEL
singlenamespace-alpha
VERSION
0.9.4 (Current)
CAPABILITY LEVEL
Basic Install
Seamless Upgrades
Full Lifecycle
Deep Insights
Auto Pilot
Install
CHANNEL
singlenamespace-alpha
VERSION
0.9.4 (Current)
CAPABILITY LEVEL
Basic Install
Seamless Upgrades
Full Lifecycle
Deep Insights
Auto Pilot

Red Hat

Let’s Explore!

Red Hat

Level 1: Basic Install



Red Hat

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

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

Red Hat

```
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
```

Red Hat

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

Red Hat

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

Red Hat

How do you install an Operator?

Installing an Operator

4

Step 1: Apply the CRD(s).

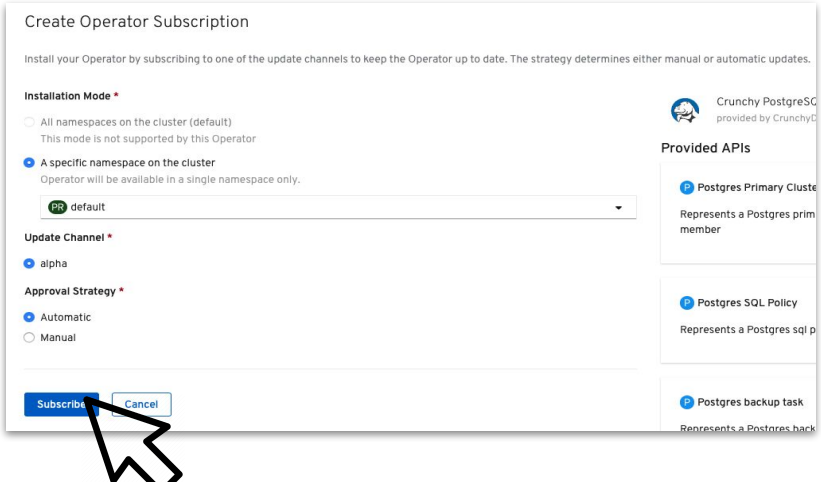
Step 2: Apply a Service Account and RBAC.

Step 3: Apply the controller via a Deployment.

Step 4: Apply the CR(s).

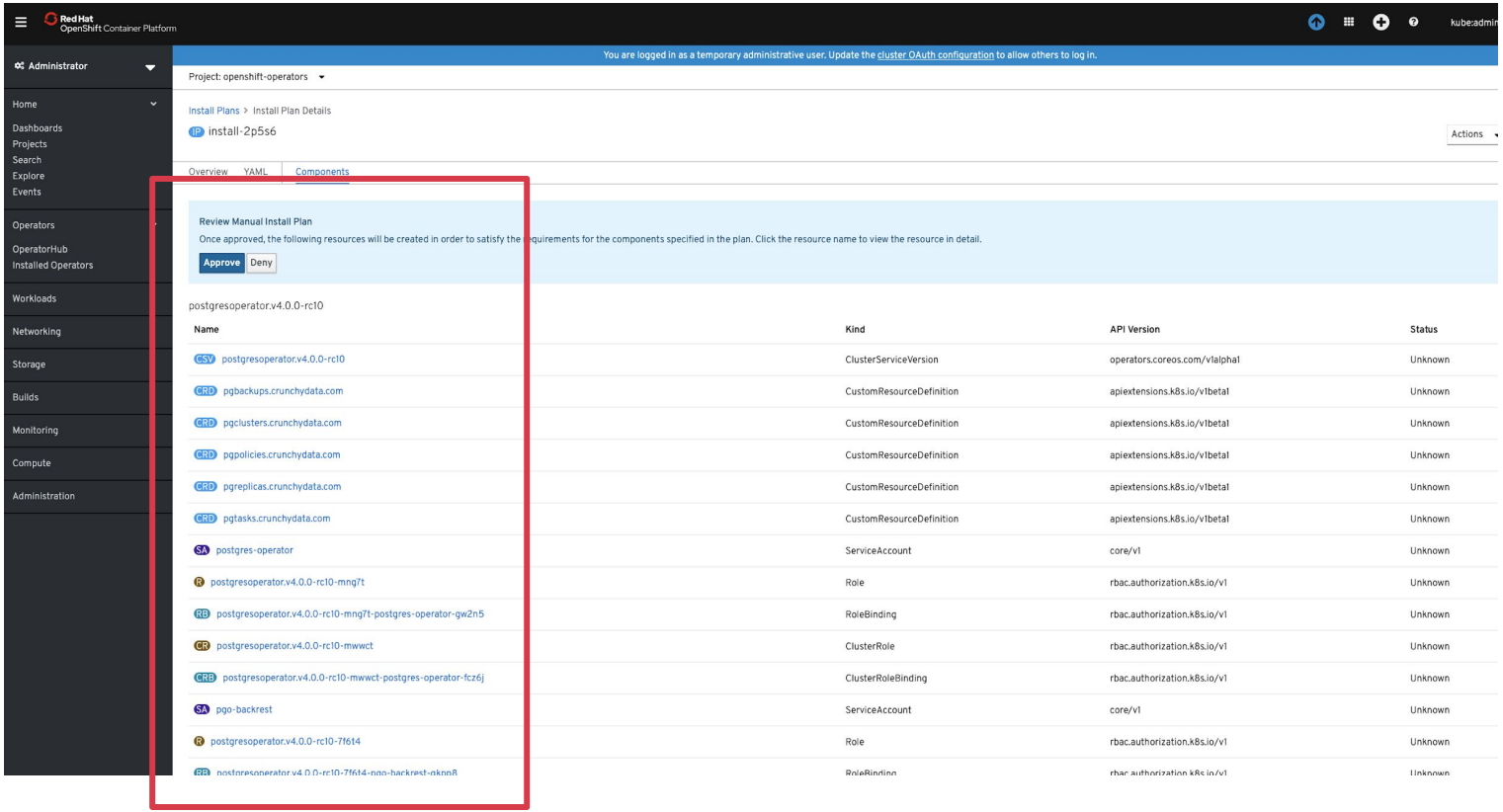
Red Hat

Installing an Operator via OLM



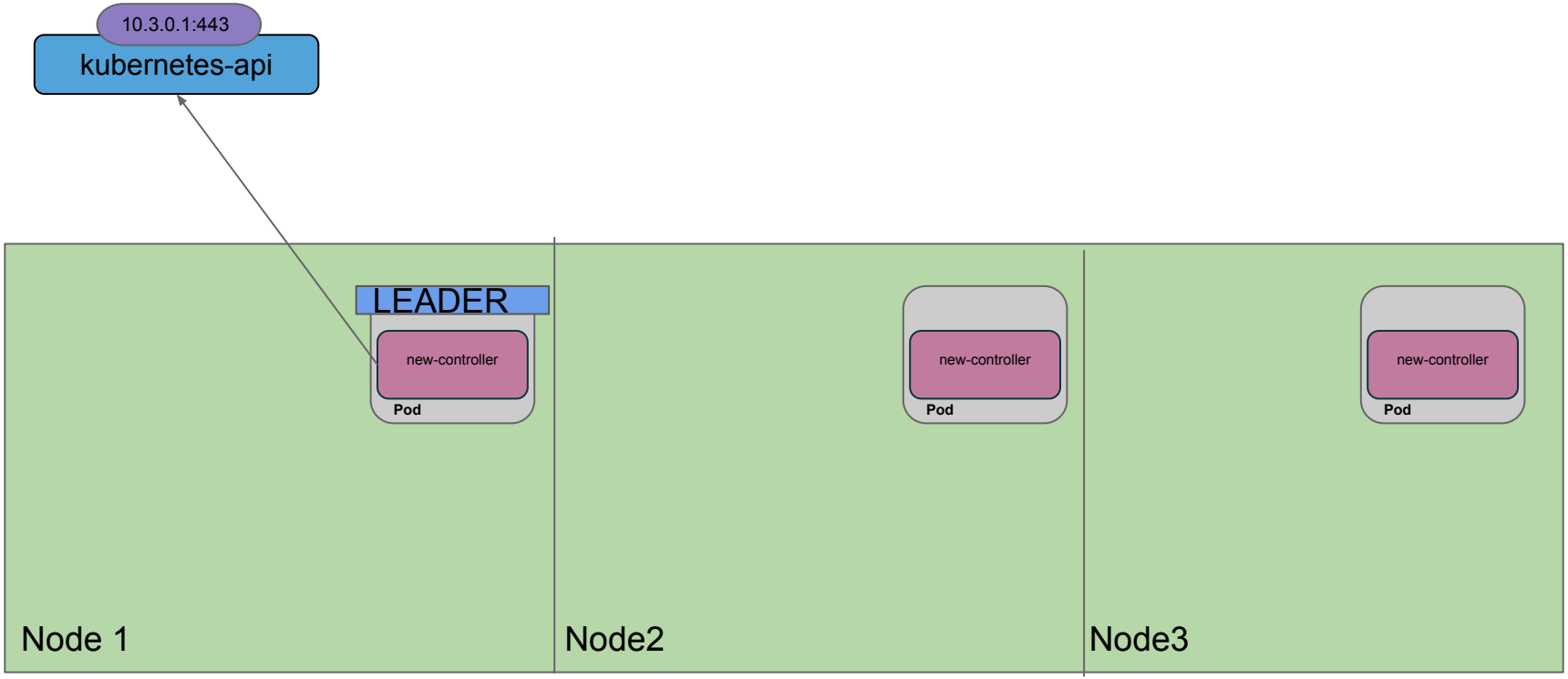
Red Hat

Review Resources Before Installing



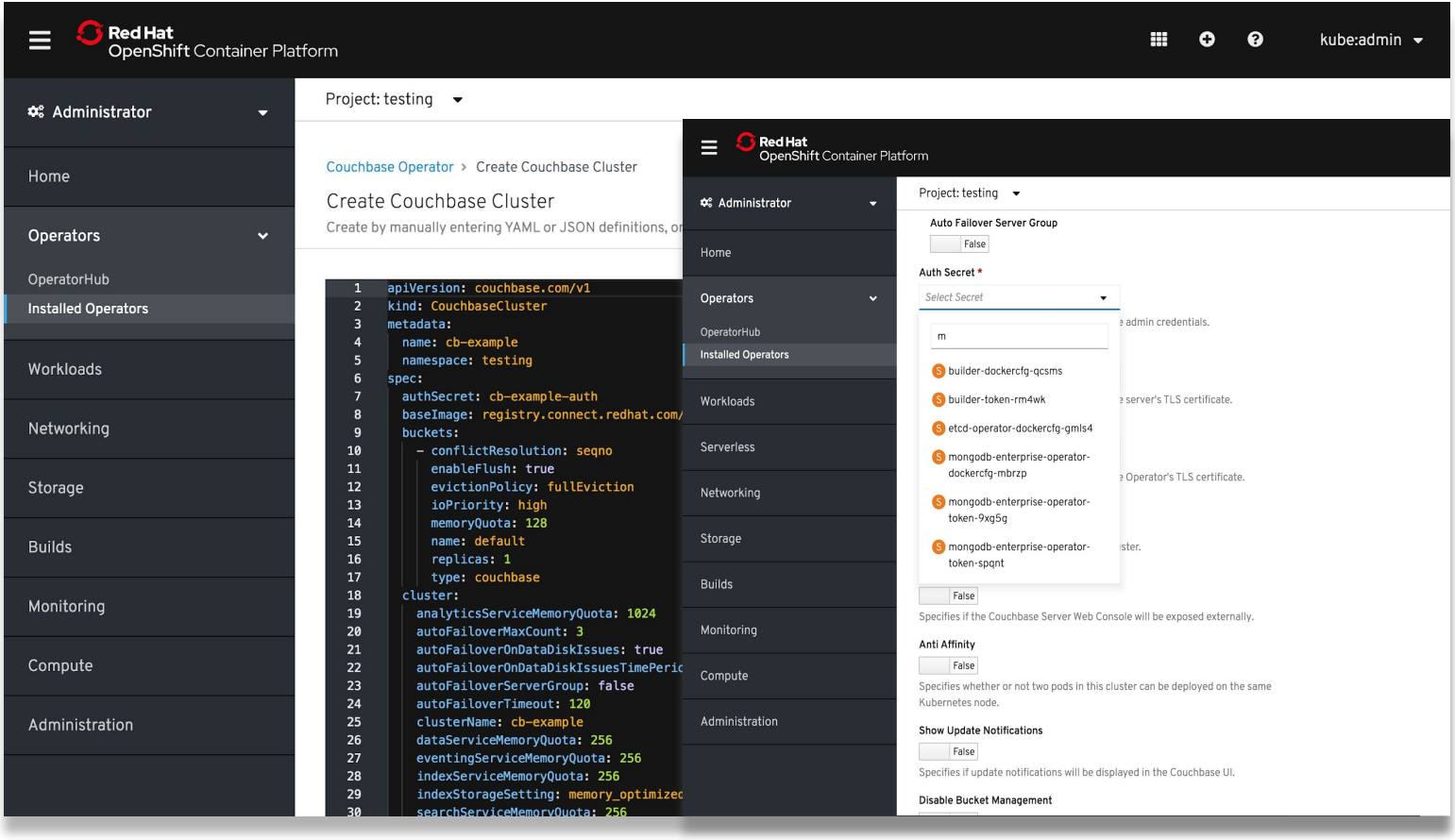
Red Hat

CRD and Controller are Applied



Red Hat

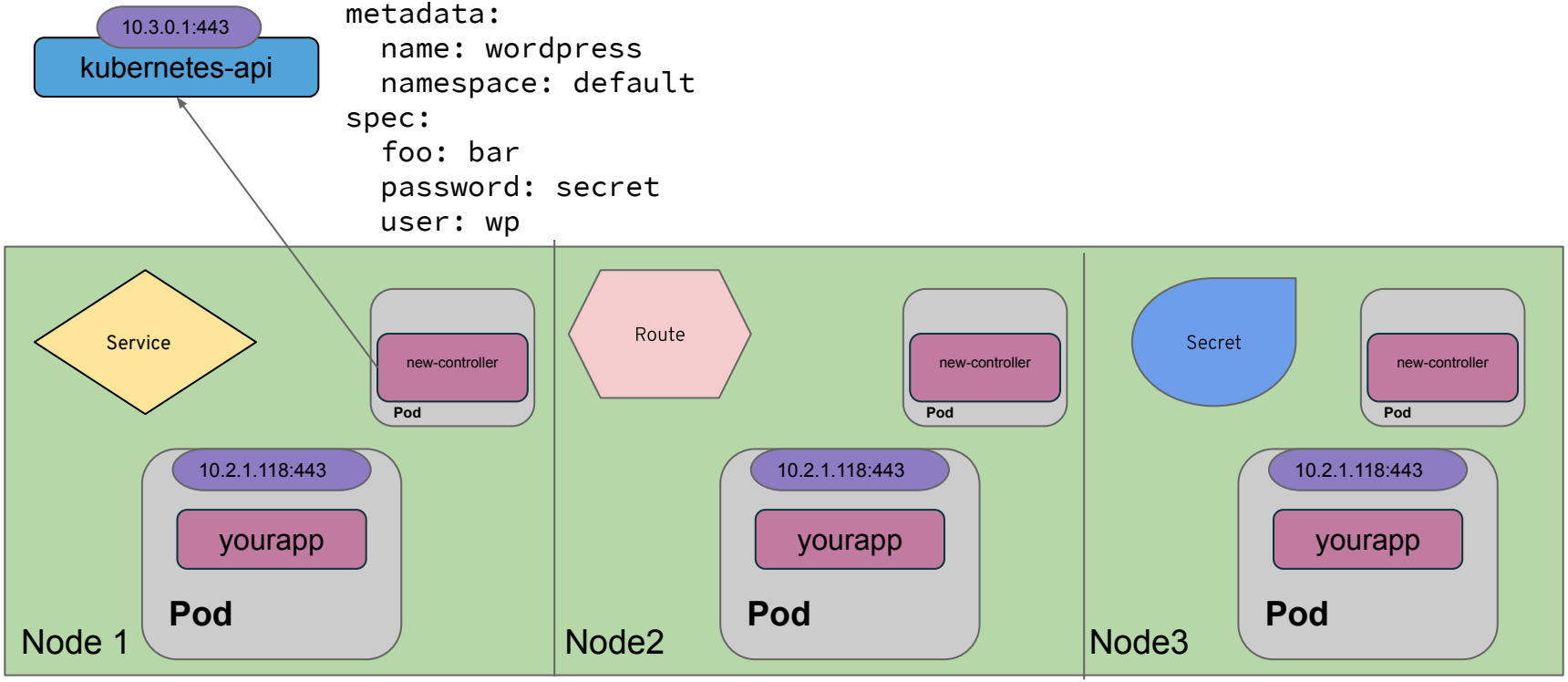
Create the CR(s)



Red Hat

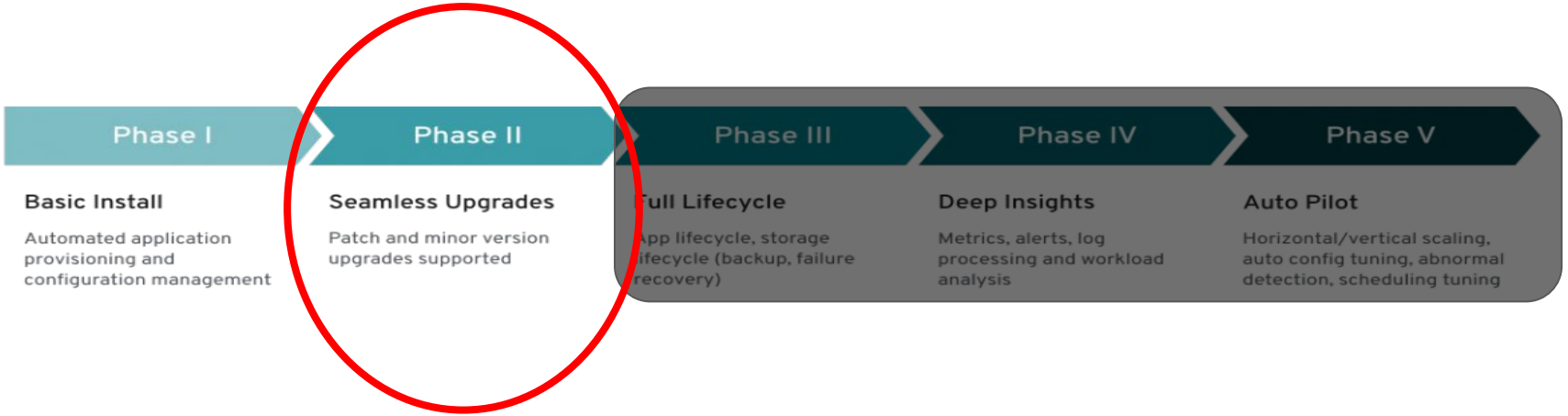
CR is Applied

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



Red Hat

Level 2: Seamless Upgrades



Red Hat

Operator Upgrades

3

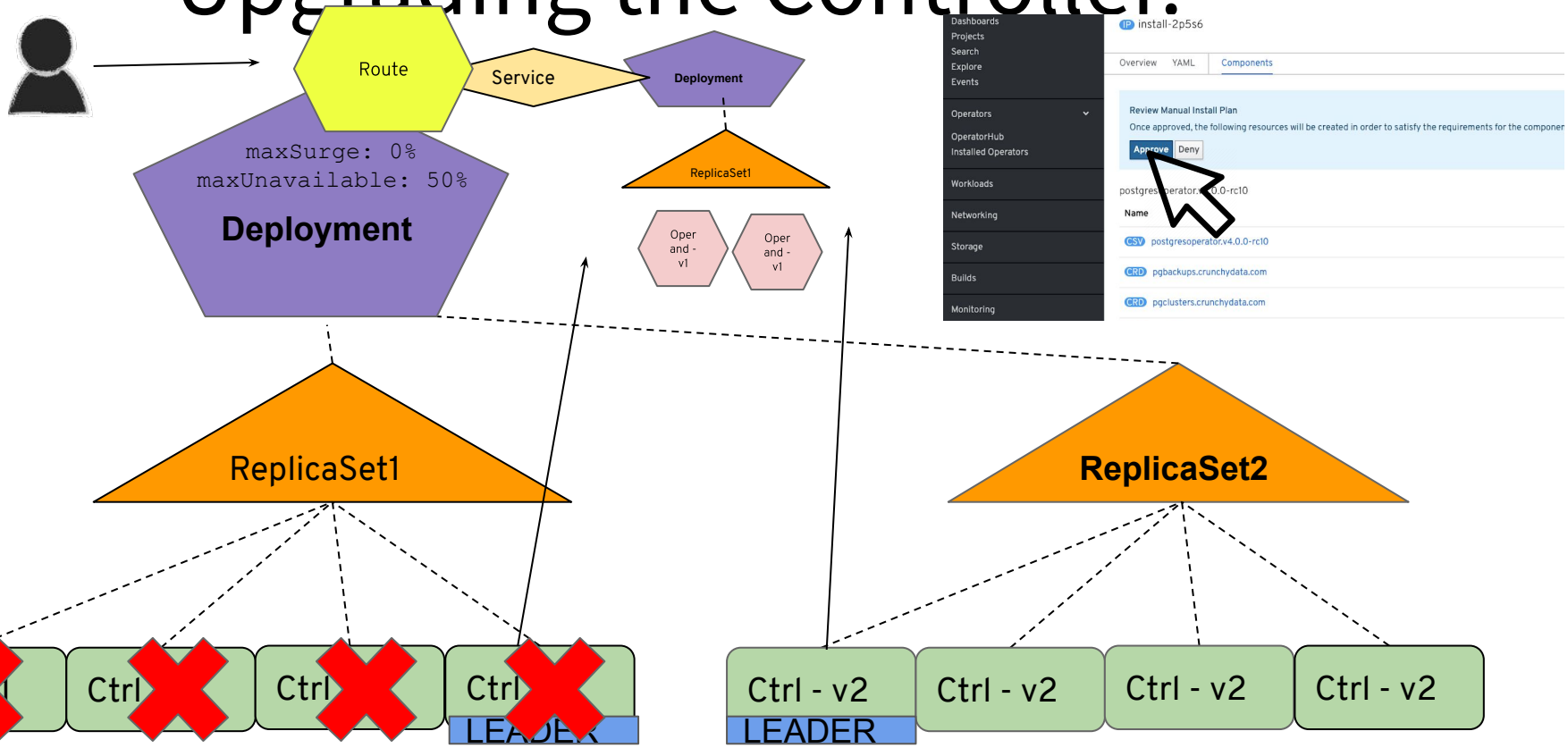
Upgrading the Controller.

Upgrading the Operand.

Upgrading the CRD version/schema.

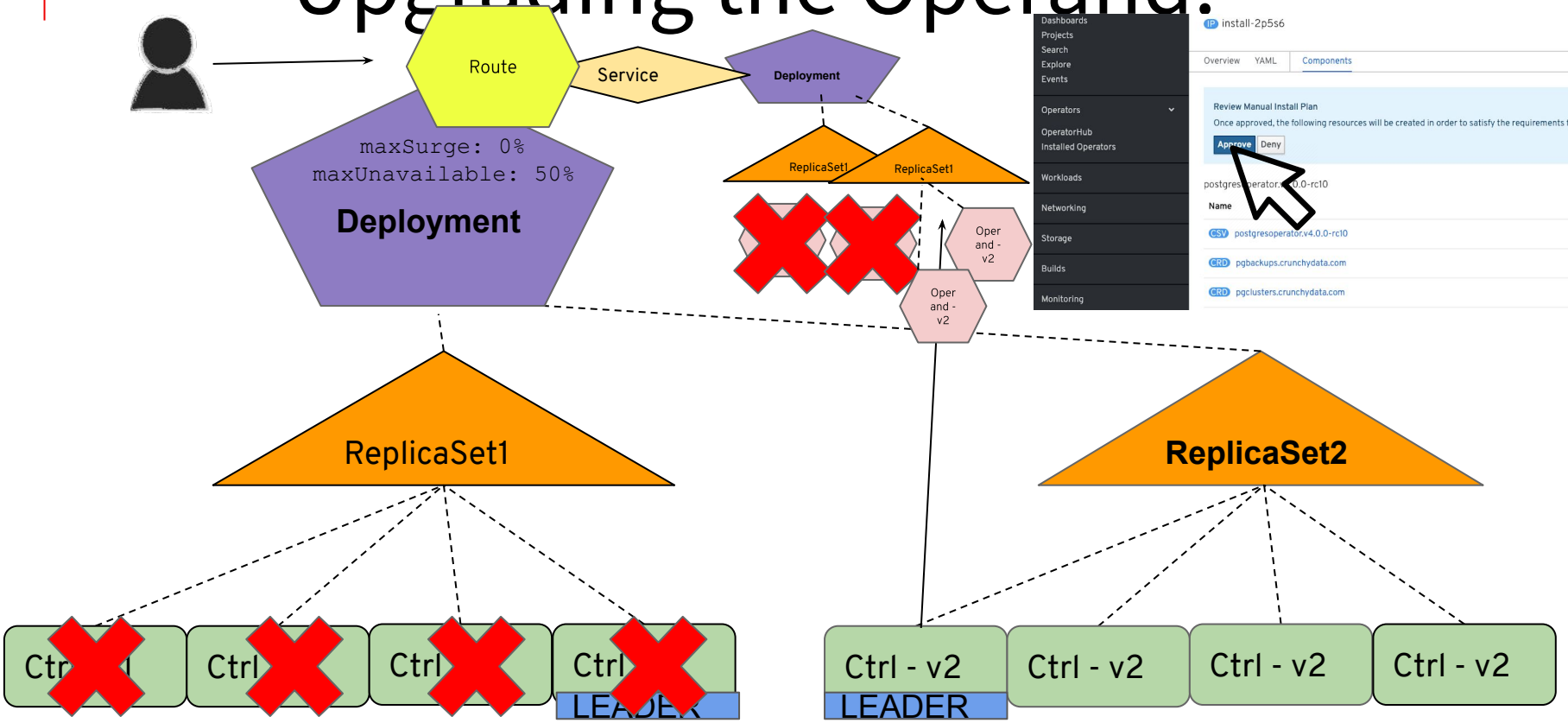
Red Hat

Upgrading the Controller.



Red Hat

Upgrading the Operand.



Red Hat

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
```

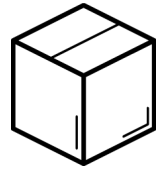
```
Kind: CustomResourceDefinition
metadata:
  name: crontabs.stable.example.com
spec:
  group: stable.example.com
  versions:
  - name: v1alpha1
    served: true
    storage: true
  scope: Namespaced
  names:
    plural: crontabs
    singular: crontab
    kind: CronTab
    shortNames:
    - ct
  validation: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

Red Hat

Upgrading the CRD

OPERATOR LIFECYCLE MANAGER

metadata



package

```
 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

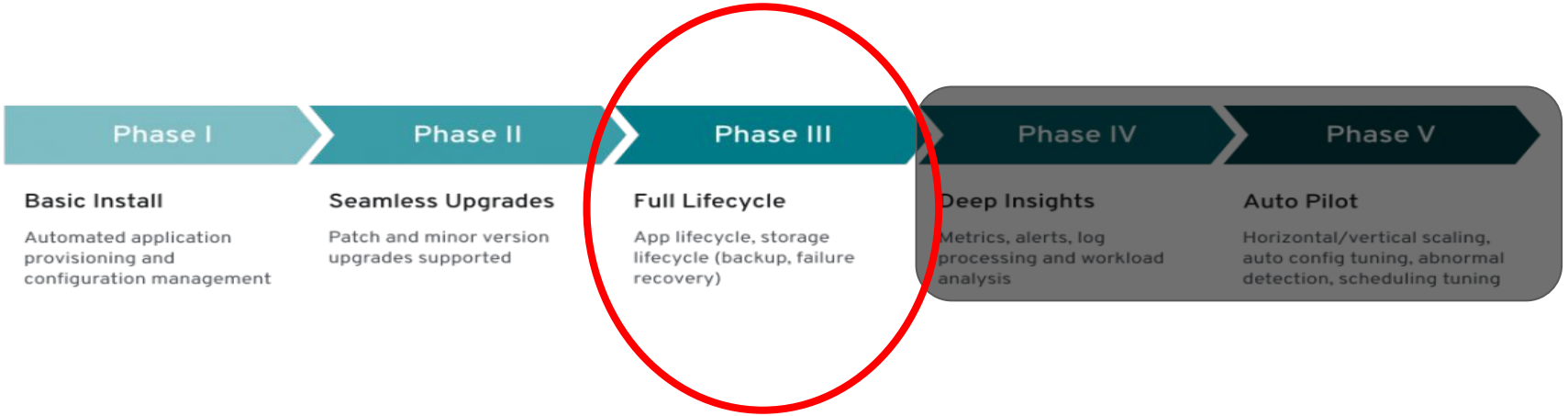
Webhook

Ctrl - v2

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

Red Hat

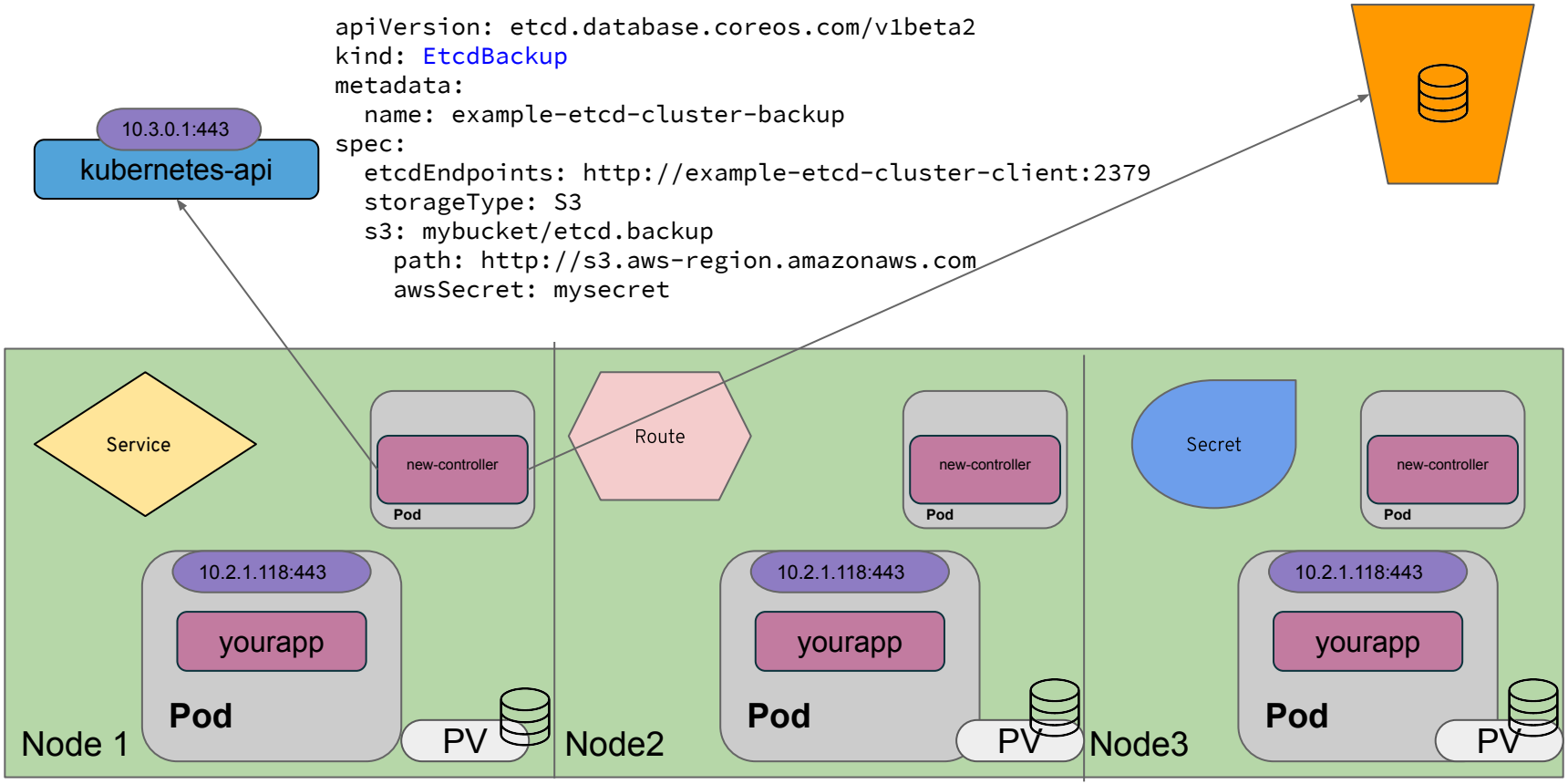
Level 3: Full Lifecycle (Backup & Restore)



Red Hat

Operator Backup

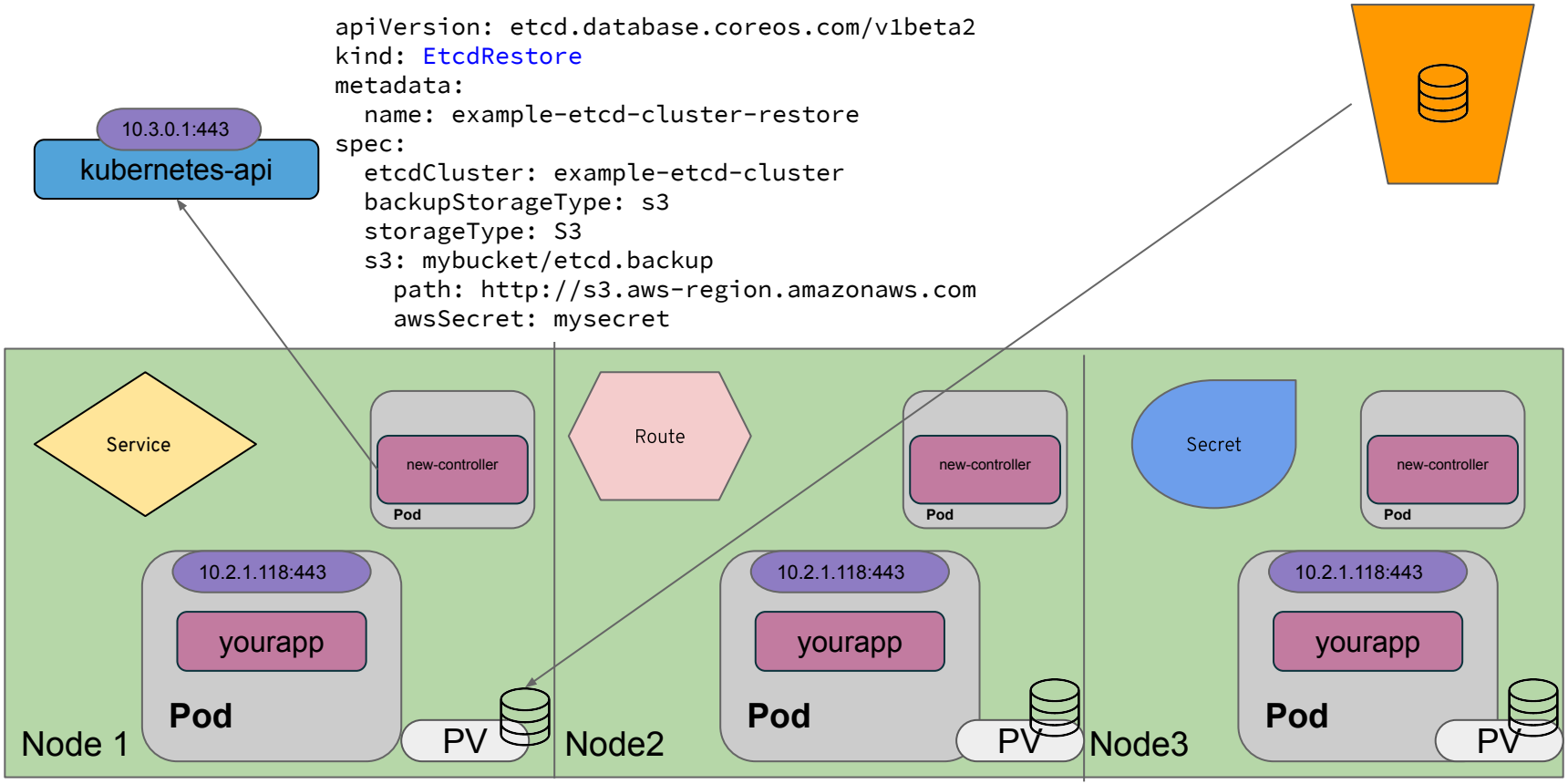
```
apiVersion: etcd.database.coreos.com/v1beta2
kind: EtcdBackup
metadata:
  name: example-etcd-cluster-backup
spec:
  etcdEndpoints: http://example-etcd-cluster-client:2379
  storageType: S3
  s3: mybucket/etcd.backup
    path: http://s3.aws-region.amazonaws.com
    awsSecret: mysecret
```



Red Hat

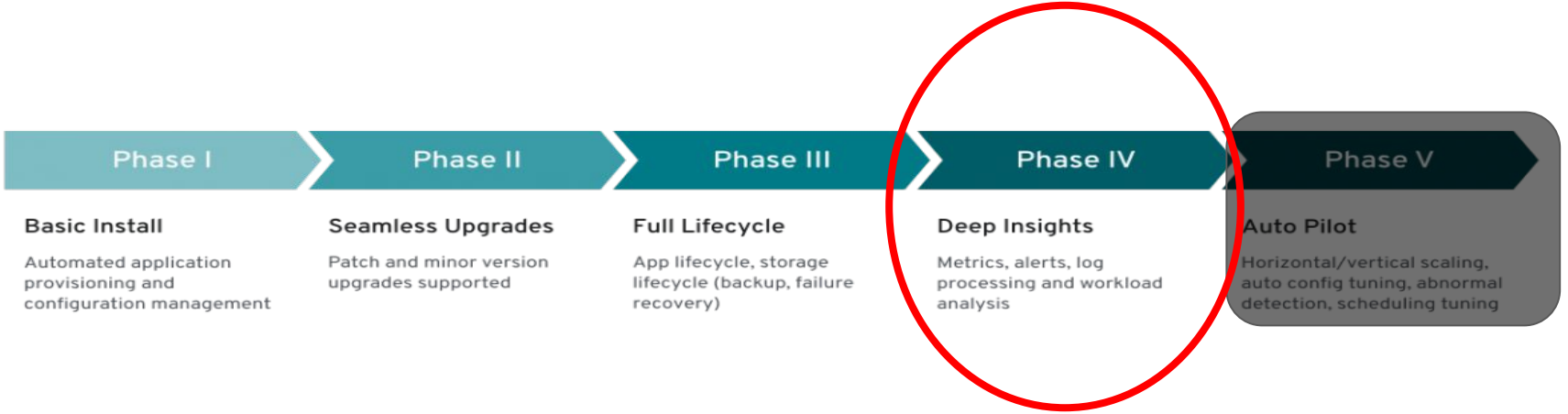
Operator Restore

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



Red Hat

Level 4: Deep Insights

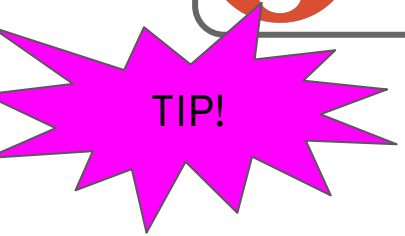


Red Hat

Deep Insights = Observability

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

Level 4: Metrics!



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

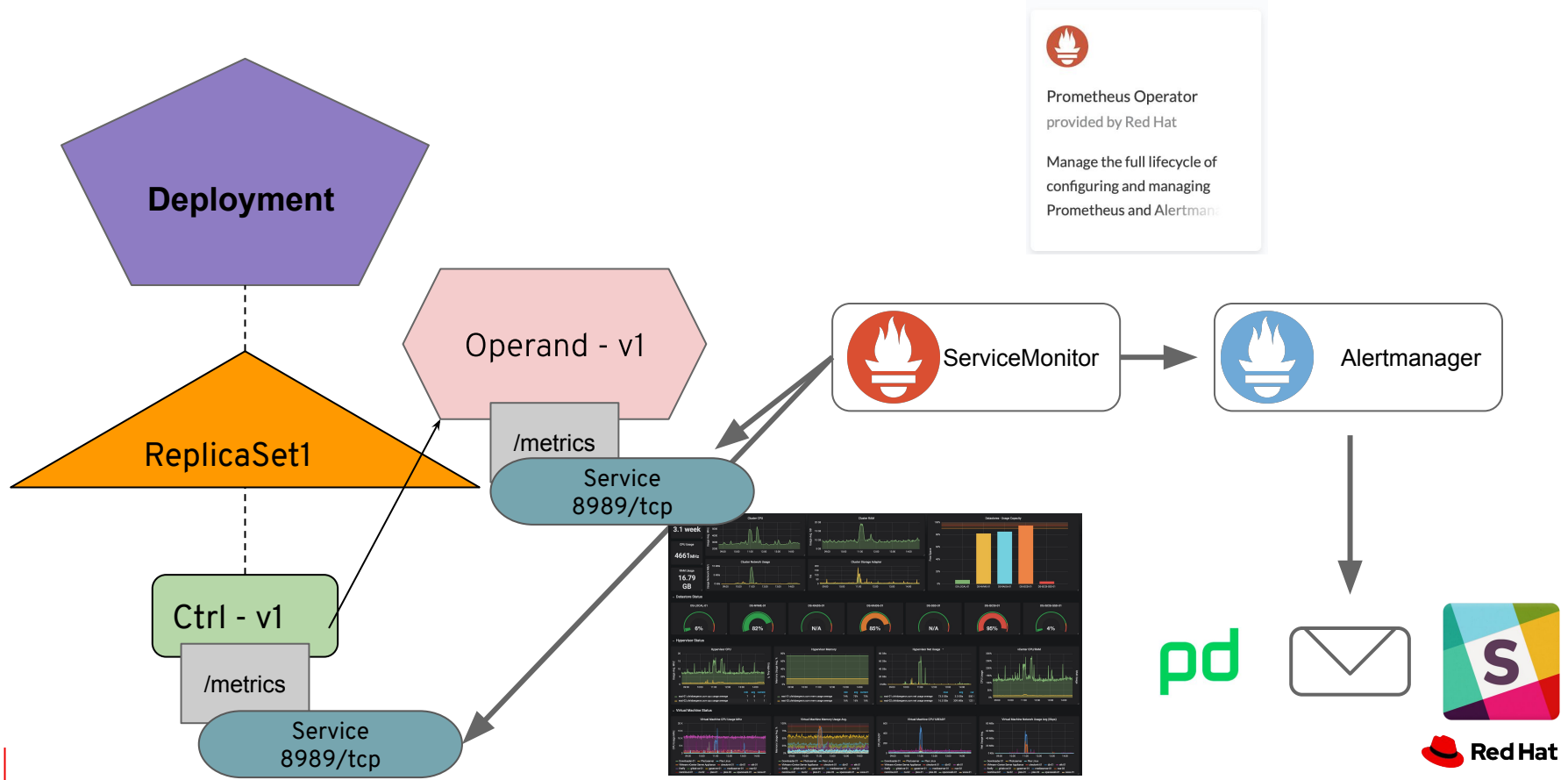
Operator-SDK does this for you and will automatically give you some default metrics including kube-state-metrics (8383/metrics)

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.

Red Hat

Operator and Operand Metric Endpoints



```
oc describe mycr example
.
.
.
.
.
Events:  <none>
```

Red Hat

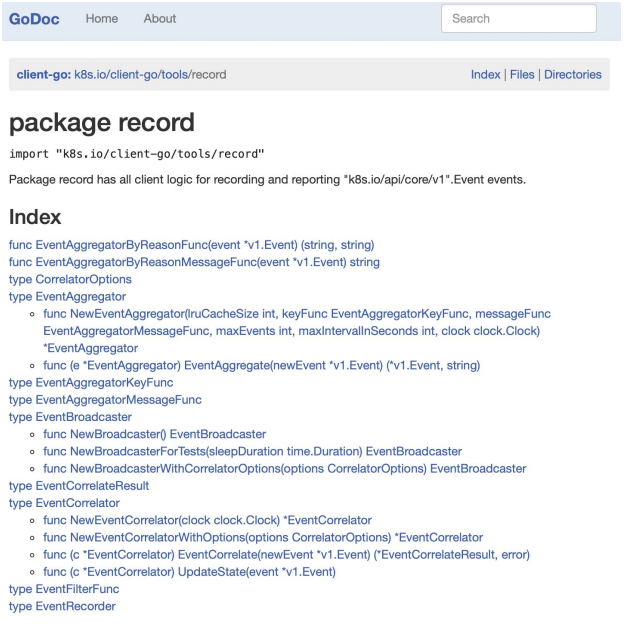
oc describe etcdcluster example

```
Events:
  Type    Reason             Age   From                          Message
  ----    ------             ----  ----                          -------
  Normal  New Member Added  38s   etcd-operator-594fbd565f-r59nh  New member example-p2knrr94d5 added to
cluster
  Normal  New Member Added  22s   etcd-operator-594fbd565f-r59nh  New member example-kx2k2hhft9 added to
cluster
  Normal  New Member Added  6s    etcd-operator-594fbd565f-r59nh  New member example-b2nxkck495 added to
cluster
```

Red Hat

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

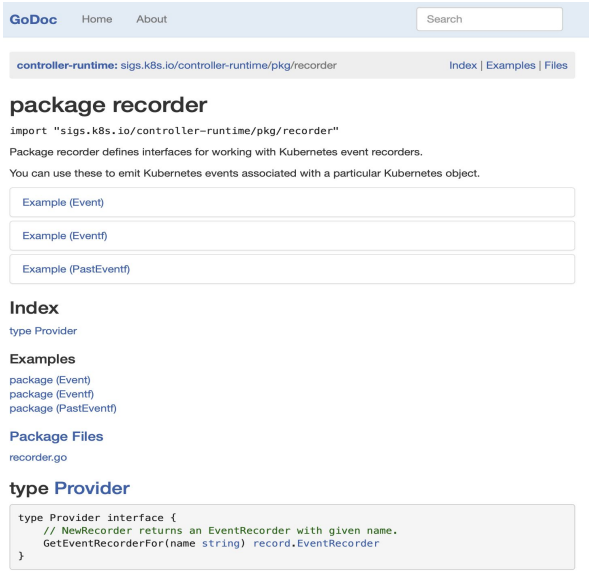
func EventAggregatorByReasonFunc(event *v1.Event) (string, string)
func EventAggregatorByReasonMessageFunc(event *v1.Event) string
type CorrelatorOptions
type EventAggregator
- func NewEventAggregator(lruCacheSize int, keyFunc EventAggregatorKeyFunc, messageFunc EventAggregatorMessageFunc, maxEvents 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

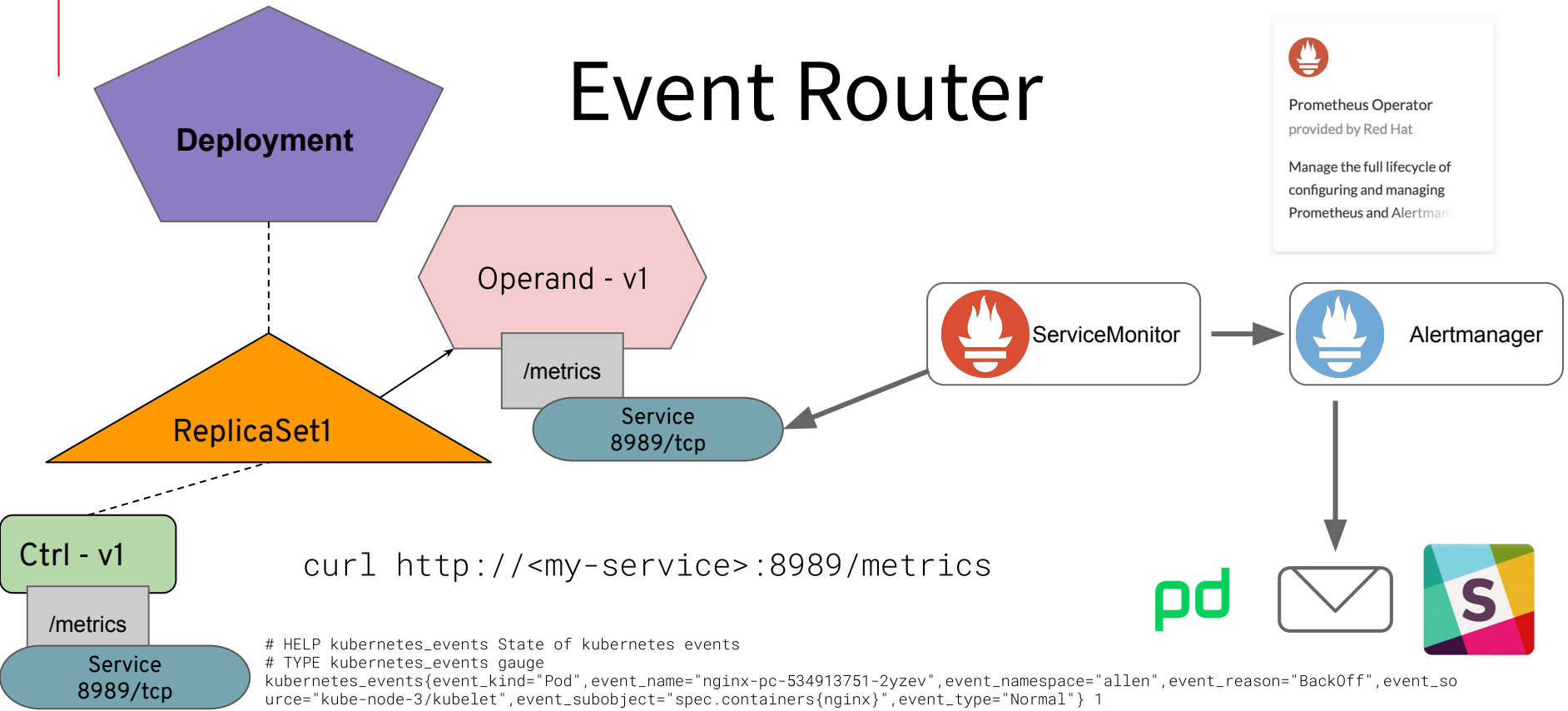
recorder.go

type Provider

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

Red Hat

Event Router

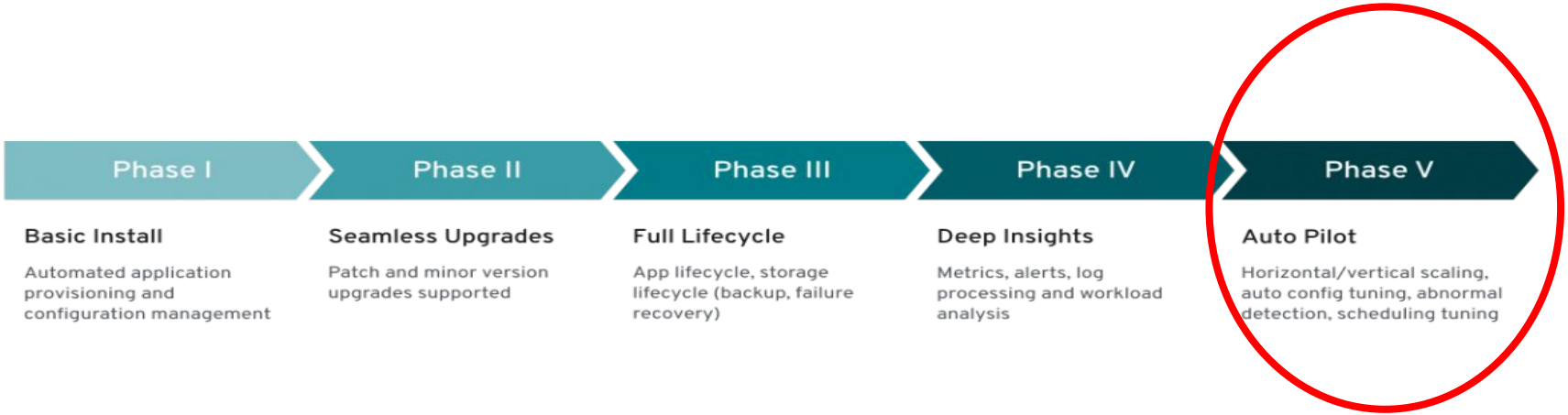



```
curl http://<my-service>:8989/metrics
```

```
# 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_ErrImagePull",event_source="kube-node-3/kubelet",event_subobject="",event_type="Warning"} 0
```

Red Hat

Level 5: AutoPilot



Red Hat

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.

Red Hat

Get Your Kubernetes Operator

Red Hat Certified and Featured in the

OpenShift OperatorHub Today!

Red Hat

THANK YOU!

Red Hat