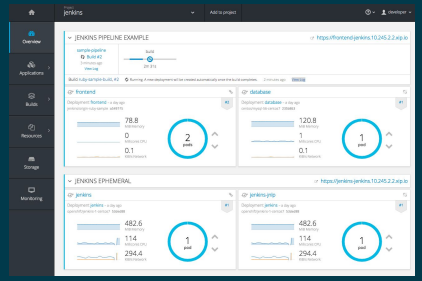
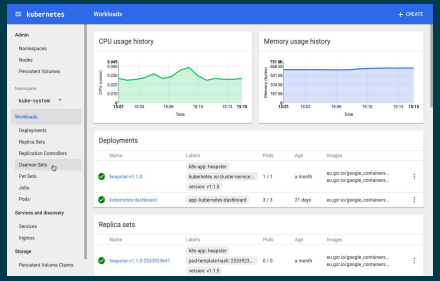
Kubernetes Overview

4

1. Dashboard





2. Command Line Interface

```
$ kubectl apply -f my-new-deployment.yaml
```

3. SDK/Client Libraries

```
pod, err :=
c.Pods(v1.NamespaceDefault).Get("my-pod")

    if err != nil {

        fmt.Println(err)

        return
```

4. Helm

```
$ helm install stable/mariadb
```

API

Red Hat

About Client-Go

A collection of tools/frameworks (in the form of Go packages) for all your Kubernetes programming needs.

Contents of Client-Go

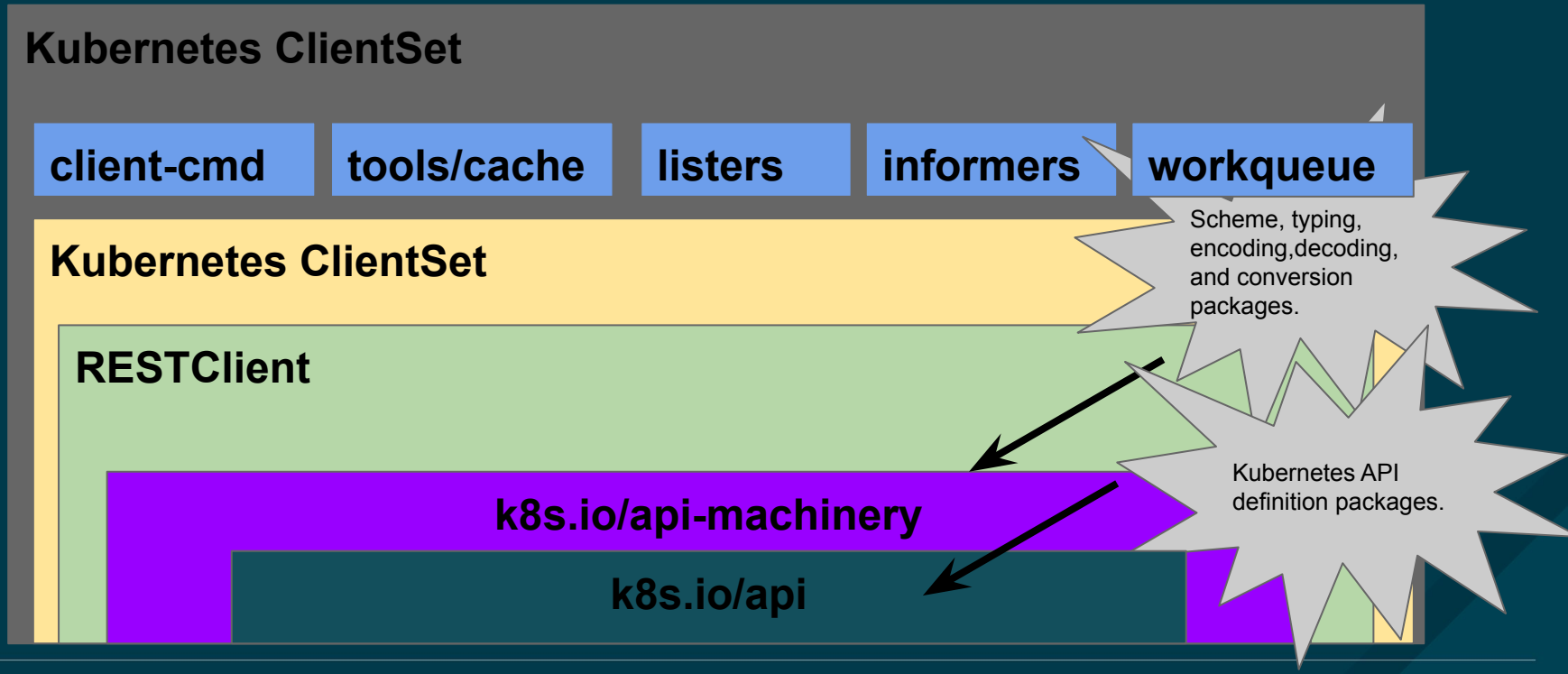
Clients

- Clientset
- Discovery
- RESTclient

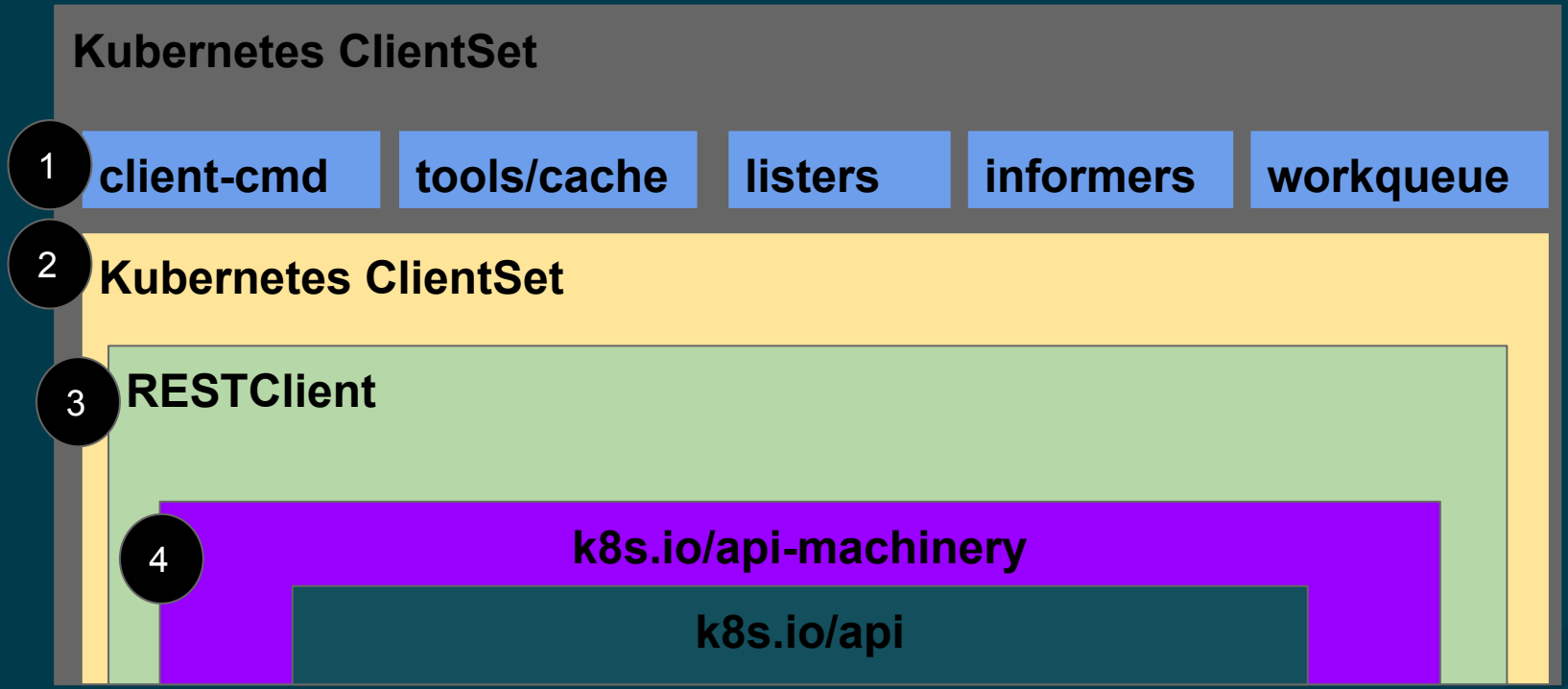
Utilities for Writing Controllers

- Workqueue
- Informers/Shared Informers

Client-Go Implementation



Out-of-Cluster Interaction With Kubernetes API



1 **client-cmd**

Fetch the kube-config file and use current-context

```
func main() {
	var kubeconfig *string
	if home := homeDir(); home != "" {
		kubeconfig = flag.String("kubeconfig", filepath.Join(home, ".kube", "config"), "(optional) absolute path to the kubeconfig file")
	} else {
		kubeconfig = flag.String("kubeconfig", "", "absolute path to the kubeconfig file")
	}
	flag.Parse()

config, err := clientcmd.BuildConfigFromFlags("", *kubeconfig)
	if err != nil {
		panic(err.Error())
	}
```

"k8s.io/client-go/tools/clientcmd"

Create the client-set

```
clientset, err := kubernetes.NewForConfig(config)
        if err != nil {
                panic(err.Error())
        }
```

"k8s.io/client-go/kubernetes"

2 Kubernetes ClientSet

3 RESTClient

4 k8s.io/api-machinery

Retrieve the Corev1 Client via clientset and list all pods in the cluster (across all namespaces)

```
for {
        pods, err := clientset.CoreV1().Pods("").List(metav1.ListOptions{})
        if err != nil {
                panic(err.Error())
        }
        fmt.Printf("There are %d pods in the cluster\n", len(pods.Items))
}
```

```
Verb(verb string) *Request
Post() *Request
Put() *Request
Patch(pt types.PatchType) *Request
Get() *Request
Delete() *Request
APIVersion() schema.GroupVersion
```

```
type ListOptions struct {
    TypeMeta      `json:",inline"`
    LabelSelector    string
`json:"labelSelector,omitempty"`
    FieldSelector    string
`json:"fieldSelector,omitempty"`
protobuf:"bytes,2,opt,name=fieldSele
```

Red Hat

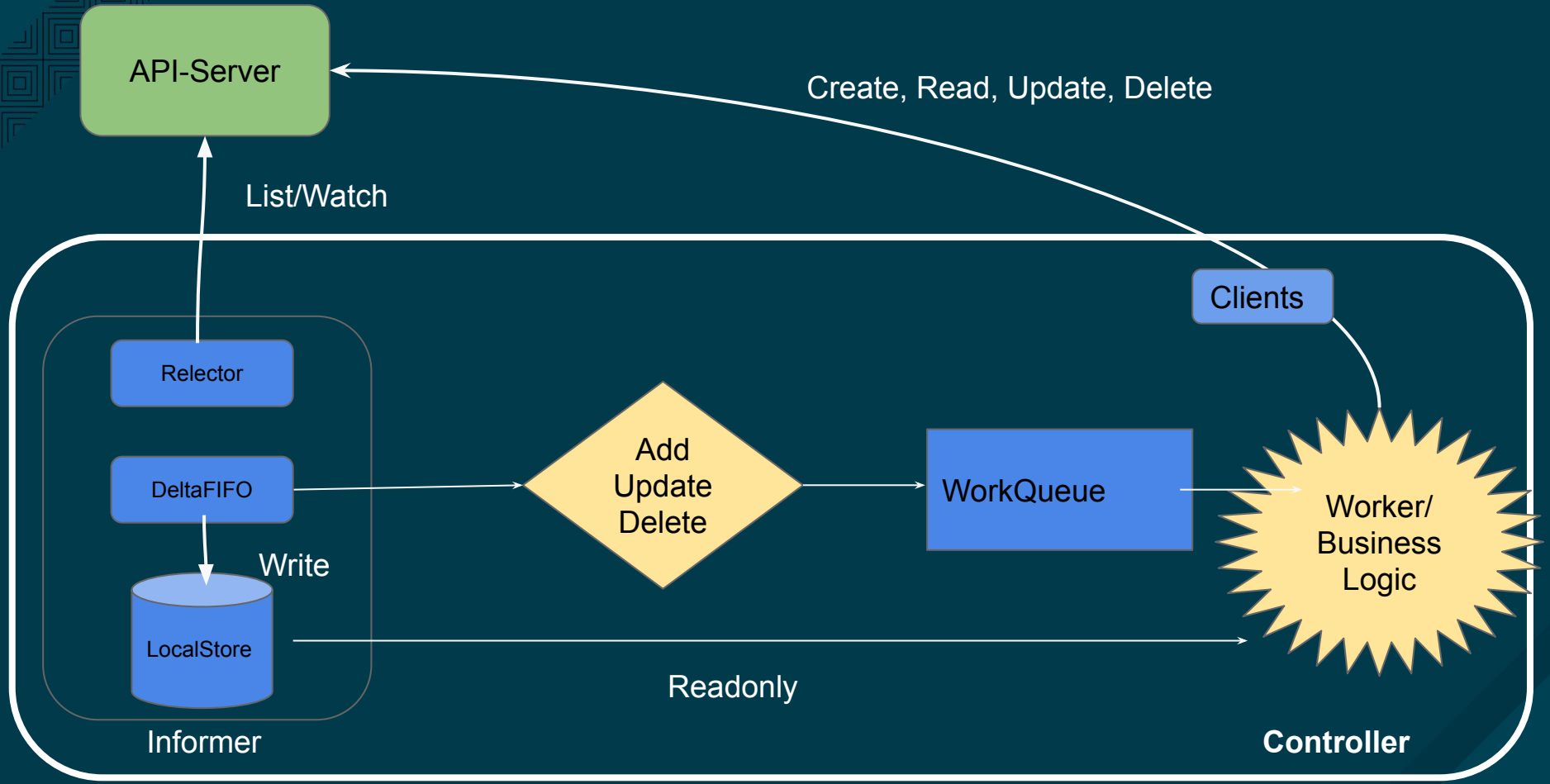
2 Kubernetes ClientSet

3 RESTClient

4 k8s.io/api-machinery

Retrieve the Corev1 Client via clientset and get **spec** for **individual pod** in the **default** namespace.

```
for {
        pod,  err := clientset.CoreV1().Pods("default").Get(“my-pod”, metav1.GetOptions{})
        if err != nil {
                panic(err.Error())
        }
        fmt.Printf("%v\n\n\n\n", pod.spec)
    }
```


API-Server
Create, Read, Update, Delete
List/Watch
Clients
Relector
DeltaFIFO
Write
LocalStore
Add
Update
Delete
WorkQueue
Worker/
Business
Logic
Readonly
Informer
Controller

```
1    while true {
2    receiveInfoAboutAPIObjects()
3    synchronizeRealStateToMatchFetchedInfo()
4    }
```

Kubernetes Concepts

What is a Kubernetes Resource?

Most Common Definition...

Any individual Kubernetes item such as a deployment, pod, service, or secret, etc.

Kubernetes Resources

- Nodes
- Namespaces
- Pods
- Endpoints
- Services
- Deployments
- ReplicaSets
- Persistent Volumes
- PersistentVolumeClaims
- ConfigMaps
- DaemonSets
- StatefulSets
- Events
- PodDisruptionBudgets
- PodSecurityPolicies
- ResourceQuotas
- Service Accounts
- HorizontalPodAutoScalers

A Better Definition...

A Kubernetes Resource is a **declarative API** with well defined Schema structure and endpoints.*

*Because the structure of the Schema and Endpoints are predictable and structured, most Kubernetes tools work with any Kubernetes API even if they are not part of the core (e.g. extensions through CRDs).

```
oc proxy
```

```
curl localhost:8001
```

What is a Declarative API?

Declarative vs. Imperative API

- Declarative expresses a fixed state that the cluster must continually work towards.
- “What to Do”
  - Example: `$ replicas 3`
- Imperative API expresses an operation that may change state but does not define an absolute state that must be maintained.
- “How to Do It”
  - Example: `$ add-pods 2`

ReplicaSet Manifest

Resource Schema Components

GVK aka TypeMeta

Metadata aka ObjectMeta

Spec

Status

```
apiVersion: extensions/v1beta1
kind: ReplicaSet

metadata:
  name: my-first-replica-set
  namespace: myproject

spec:
  selector:
    matchLabels:
     app: nginx
  replicas: 5
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
status:
  availableReplicas: 1
  fullyLabeledReplicas: 1
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
```

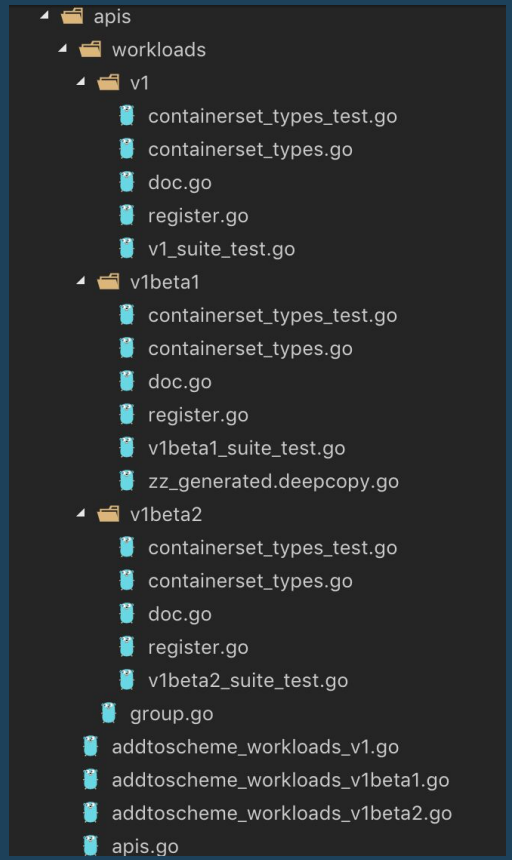
Red Hat

Resource Schema: Group, Version, Kind (GVK)

```
apiVersion: extensions/v1beta1
kind: ReplicaSet
```

- The resource ***Group*** is similar to package in a language. It disambiguates different APIs that may happen to have identically named Kinds. Groups often contain a domain name, such as redhat.com.
- The resource ***Version*** defines the stability of the API and backward compatibility guarantees - such as v1beta1 or v1.
- The resource ***Kind*** is the name of the API - such as Deployment or Service.

A Note About API Versions



Red Hat

Difference between API Version Numbers

i.e. apps/v1beta1, apps/v1beta2

- Unspecified fields may have different defaults.
- The same logical fields may have different names.

```
kubectl explain deployments.spec --api-version="apps/v1beta1"
```

```
revisionHistoryLimit   <integer>
The number of old ReplicaSets to retain to allow rollback. This is a pointer to distinguish between explicit zero and not specified.
Defaults to 2.
```

```
kubectl explain deployments.spec --api-version="apps/v1beta2"
```

```
revisionHistoryLimit   <integer>
The number of old ReplicaSets to retain to allow rollback. This is a pointer to distinguish between explicit zero and not specified.
Defaults to 10.
```

Red Hat

API Versions

Alpha (i.e. v1alpha1)

- May contain bugs. Features may be changed or removed. Field names may also be changed and not supported in the future.
- Only use for short-lived testing clusters.

Beta (i.e. v1beta1)

- Considered safe. Backwards compatibility on field names.
- Support for the feature will not be dropped, though details may change.

Stable (i.e. v1,v2)

- Stable versions of features will appear in many subsequent versions.

Not Flexible

```
http://kubernetes:6443/api/v1/pods

http://kubernetes:6443/api/v1/replicasets

http://kubernetes:6443/api/v1/services

http://kubernetes:6443/api/v1/deployments
```

Flexible

```
curl kubernetes:6443
```

```
"/api/v1"
"/apis/authentication.k8s.io/v1"
"/apis/authentication.k8s.io/v1beta1"
"/apis/authorization.k8s.io/v1"
"/apis/authorization.k8s.io/v1beta1"
"/apis/certificates.k8s.io/v1beta1"
"/apis/certificates.k8s.io"
"/apis/extensions/v1beta1"
"/apis/policy/v1beta1"
"/apis/rbac.authorization.k8s.io/v1beta1"
"/apis/rbac.authorization.k8s.io/v1alpha1"
"/apis/storage.k8s.io/v1"
"/apis/storage.k8s.io/v1beta1"
```

Allows the program to move, change, and grow over time.

Engineers can advertise to support older API versions, and offer backward-compatibility guarantees.

See Current API-Versions

```
oc api-versions
```

```
"/api/v1"
"/apis/authentication.k8s.io/v1"
"/apis/authentication.k8s.io/v1beta1"
"/apis/authorization.k8s.io/v1"
"/apis/authorization.k8s.io/v1beta1"
"/apis/certificates.k8s.io/v1beta1"
"/apis/certificates.k8s.io"
"/apis/extensions/v1beta1"
"/apis/policy/v1beta1"
"/apis/rbac.authorization.k8s.io/v1beta1"
"/apis/rbac.authorization.k8s.io/v1alpha1"
"/apis/storage.k8s.io/v1"
"/apis/storage.k8s.io/v1beta1"
```

News Snippet About Introduction of v1 NetworkPolicy

Two of the changes you need to be aware of are:

» The v1beta1 NetworkPolicy API Has Been Deprecated

The v1beta1 version of the NetworkPolicy API has been deprecated in favor of moving forward with the new behaviors and updating the behavior of the *extensions* API to allow for future expansion and development.Keep in mind that while the v1 NetworkPolicy API eclipses the existing beta, the new API endpoint will only be available on Kubernetes 1.7+ (as older versions do not include the v1 API code). As such, as you work towards upgrading, you'll want to ensure that you are using the correct version of Project Calico for the NetworkPolicy behavior you want.

» The DefaultDeny Annotation Has Been Removed

One of the bigger changes in Kubernetes 1.7 is the removal of the DefaultDeny annotation. This means that when upgrading, you should **first delete any existing NetworkPolicy** objects in namespaces that previously **did not have** the "DefaultDeny" annotation (as this may cause Kubernetes to unintentionally block traffic now).

Kubernetes API Actions and HTTP Method

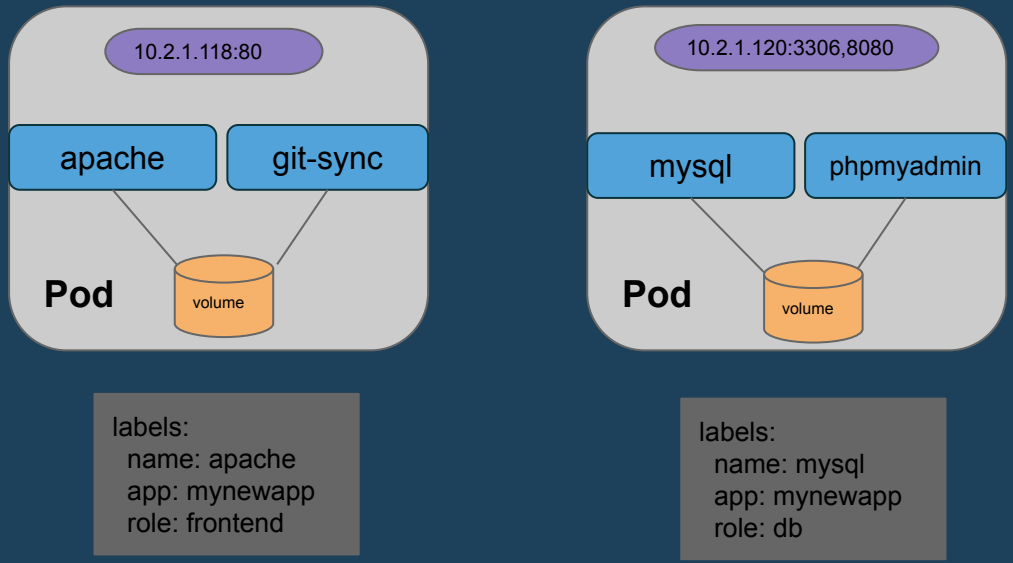
Verb	HTTP Method
Get	GET
List	GET
Watch	GET
Create	POST
Update	PUT
Patch	PATCH
Delete	DELETE

Labels/Selectors

Key/value pairs attached to resources.

Used for grouping, viewing, and operating.

Labels: Grouping




```
labels:
  name: apache
  app: mynewapp
  role: frontend
```

```
labels:
  name: mysql
  app: mynewapp
  role: db
```

Labels: Viewing

```
kubectl get pods --show-labels
```

```
db-dev                          1/1       Running             0          6s        app=my-app,environment=dev,tier=backend
www-dev                         1/1       Running             0          6s        app=my-app,environment=dev,tier=frontend
www-prod                        1/1       Running             0          6s        app=my-app,environment=production,tier=frontend
```

```
kubectl get pods -L app,environment,tier -l environment!=dev
```

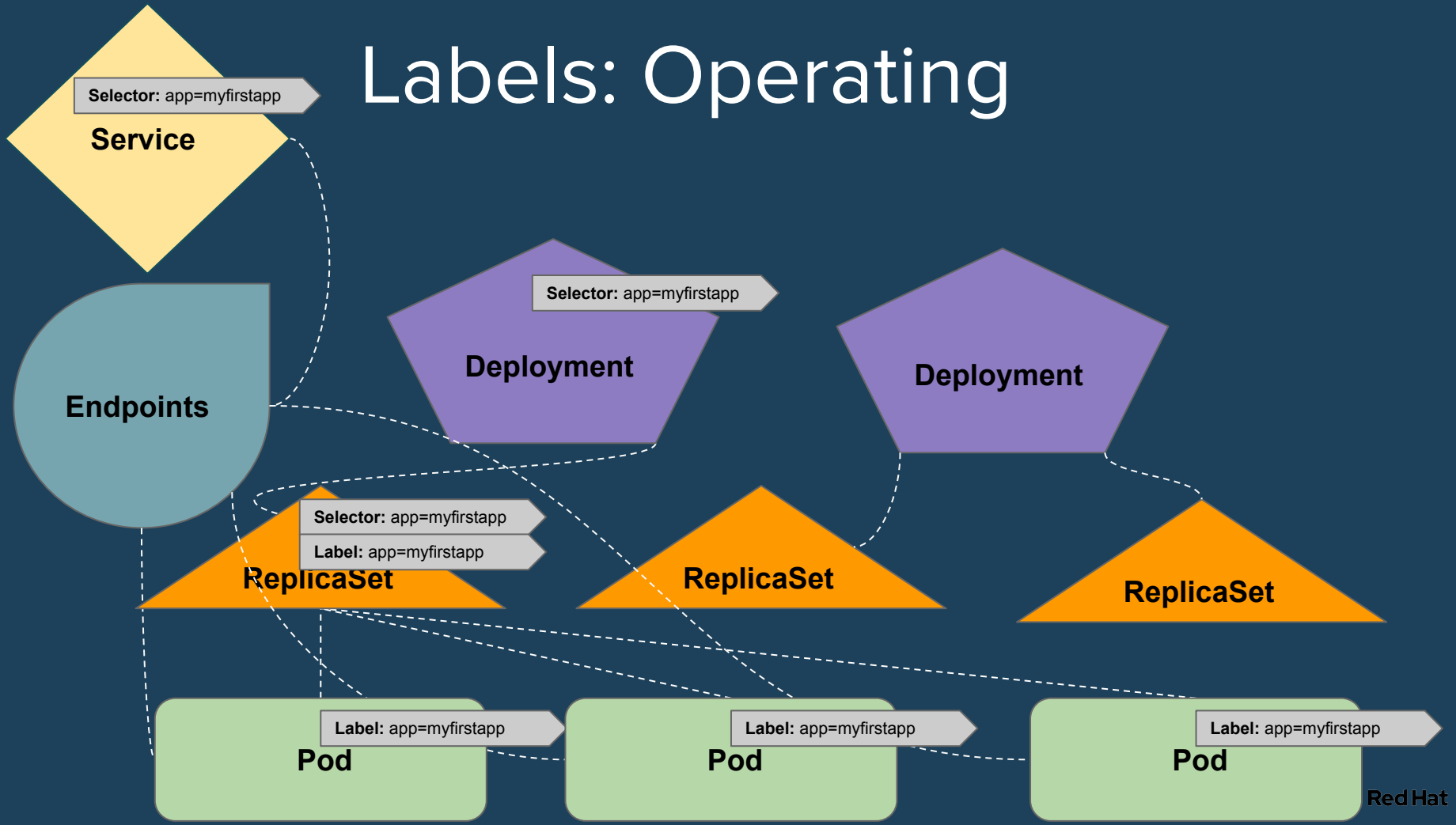
```
www-prod                        1/1       Running             0          4m        my-app    production    frontend
```

```
kubectl get pods -l "tier notin (backend,cache),environment in (dev)"
```

```
www-dev                         1/1       Running   0          6m
```

Red Hat

Labels: Operating



Using Labels

Labels can be displayed as a column in output with -L option:

```
$ kubectl get pods -L tier
NAME                          READY     STATUS    RESTARTS   AGE       TIER
my-nginx-3800858182-1v53o     1/1       Running   0          46s       backend
my-nginx-3800858182-2ds1q     1/1       Running   0          46s       backend
```

http://kubernetes.io/docs/user-guide/labels/

Using Labels

Labels can be displayed as a column in output with -L option:

```
$ kubectl get pods -L tier
NAME                          READY     STATUS    RESTARTS   AGE       TIER
my-nginx-3800858182-1v53o     1/1       Running   0          46s       backend
my-nginx-3800858182-2ds1q     1/1       Running   0          46s       backend
```

http://kubernetes.io/docs/user-guide/labels/

Using Labels

Labels can be displayed as a column in output with -L option:

```
$ kubectl get pods -L tier
NAME                          READY     STATUS    RESTARTS   AGE       TIER
my-nginx-3800858182-1v53o     1/1       Running   0          46s       backend
my-nginx-3800858182-2ds1q     1/1       Running   0          46s       backend
```

http://kubernetes.io/docs/user-guide/labels/

Using Labels

Labels can be displayed as a column in output with -L option:

```
$ kubectl get pods -L tier
NAME                          READY     STATUS    RESTARTS   AGE       TIER
my-nginx-3800858182-1v53o     1/1       Running   0          46s       backend
my-nginx-3800858182-2ds1q     1/1       Running   0          46s       backend
```

http://kubernetes.io/docs/user-guide/labels/

Using Labels Effectively

Examples of multiple labels for app, tier and role:

```
labels:
        app: guestbook
        tier: frontend
```

```
labels:
        app: guestbook
        tier: backend
        role: master
```

```
labels:
        app: guestbook
        tier: backend
        role: slave
```

Other example labels:

- "release" : "stable" or "canary"
- "partition" : "customerA" or "customerB"
- "track" : "daily" or "weekly"