Red Hat

Rule Engines and Kubernetes DevOps

Luca Molteni
Principal Software Engineer

Red Hat

Rule Engines in the Cloud

- First idea was to evaluate rules in stateless interactions
- Drools works, but it’s much more powerful than that

Rule Engines in the Cloud

- Applications built in the cloud from the beginning
- Problems of the applications are the problems of the cluster (and vice versa)
- To track problems, we needed something that
  - can easily write sanity rules
  - can run indefinitely with incremental data
  - can correlate different kind of data
- Drools shines in each of these problems

Case Study: Executable SOPs

SOP: standard operating procedure

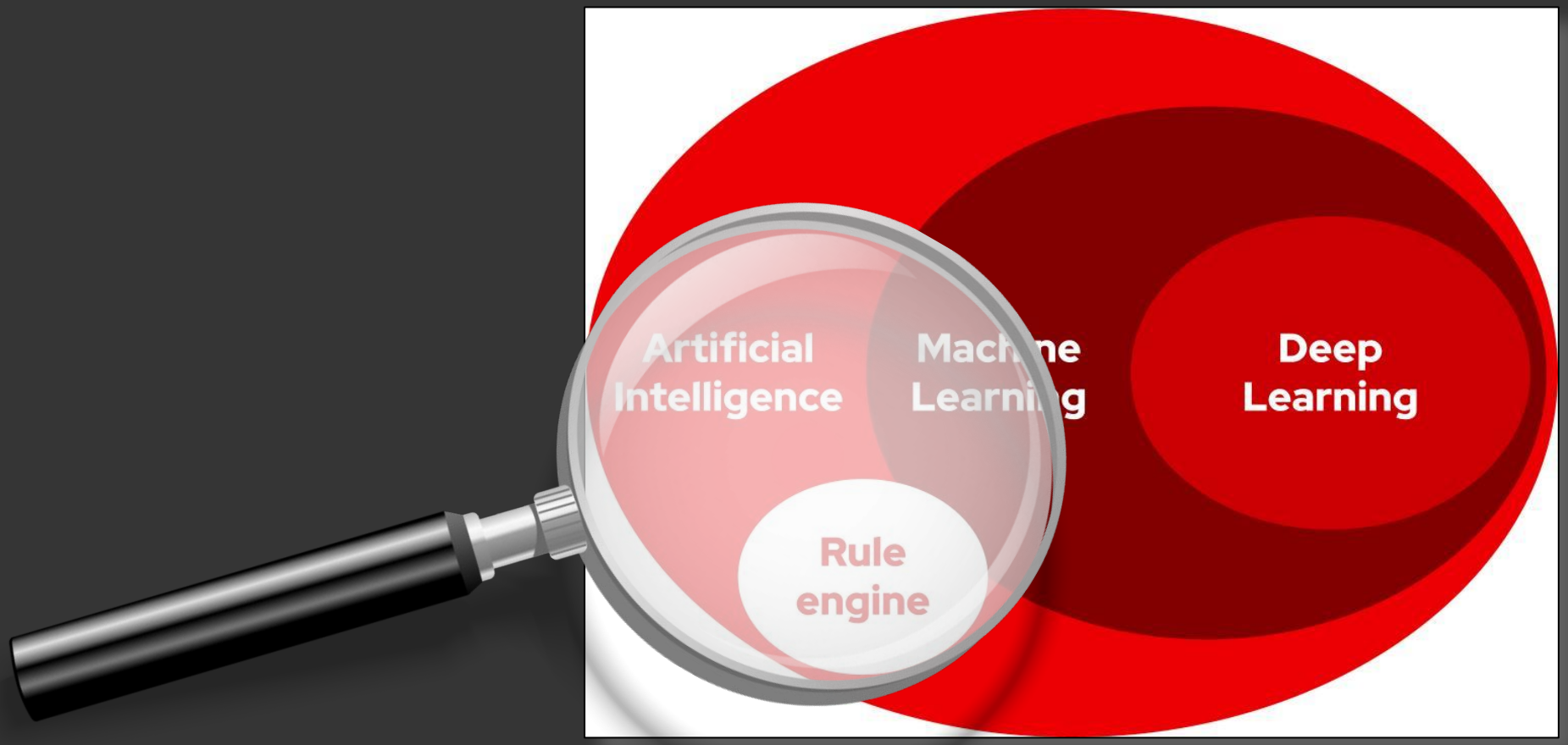
1. Simple error: experienced Devops
2. Simple error: experienced Devops at 3:00 AM
3. Complex error with custom logic
4. Complex error with custom logic at 3:00 AM

The knowledge of the whole system (infrastructure and application logic) is encoded in rules

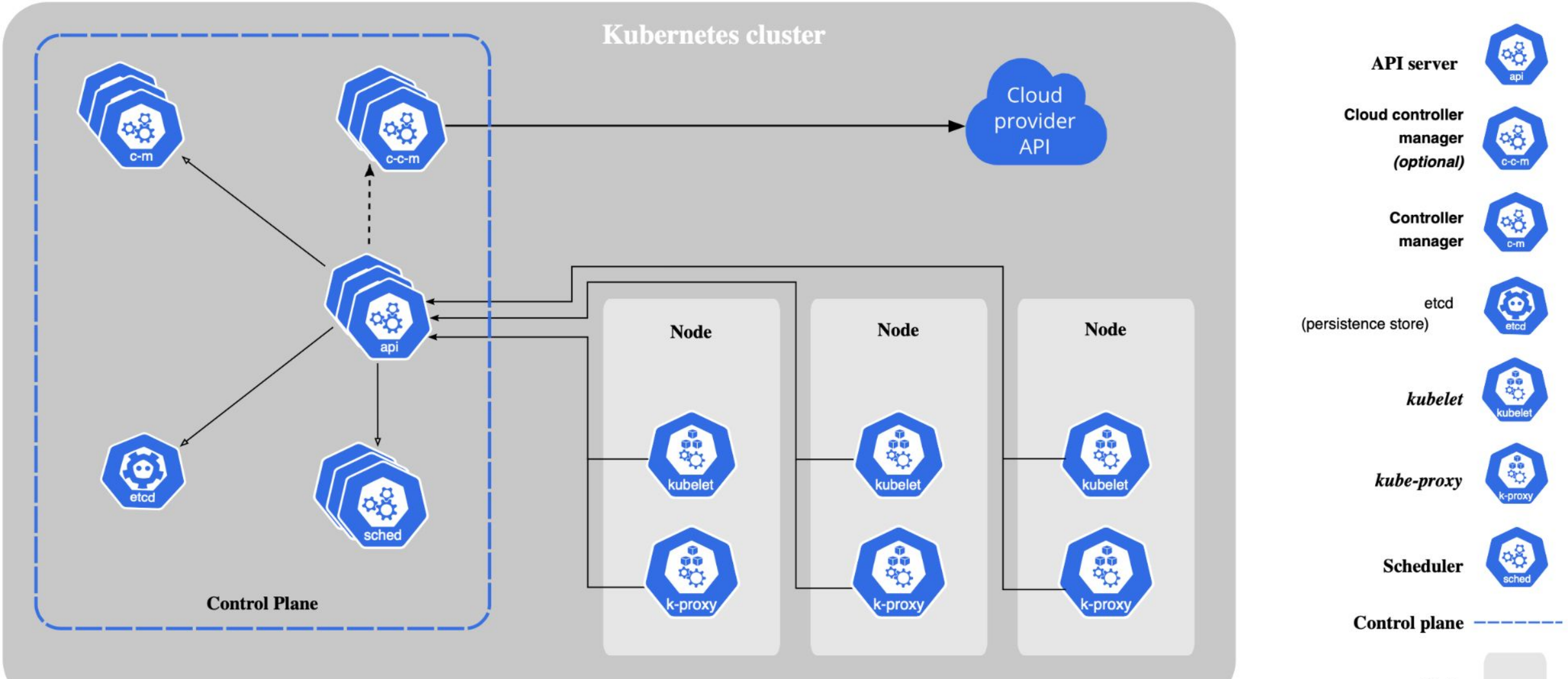
Introducing RuleOps

An exploration space for automating solutions to problematic scenarios using Drools interactions with the Kubernetes Control Plane

Symbolic AI vs Machine Learning



Kubernetes Control Plane



Kubernetes API

- Resources
  - Pods, Deployments, Services
- Endpoints
- Custom Resources
- Java mapped type using the Kubernetes Java Client

Transactionality

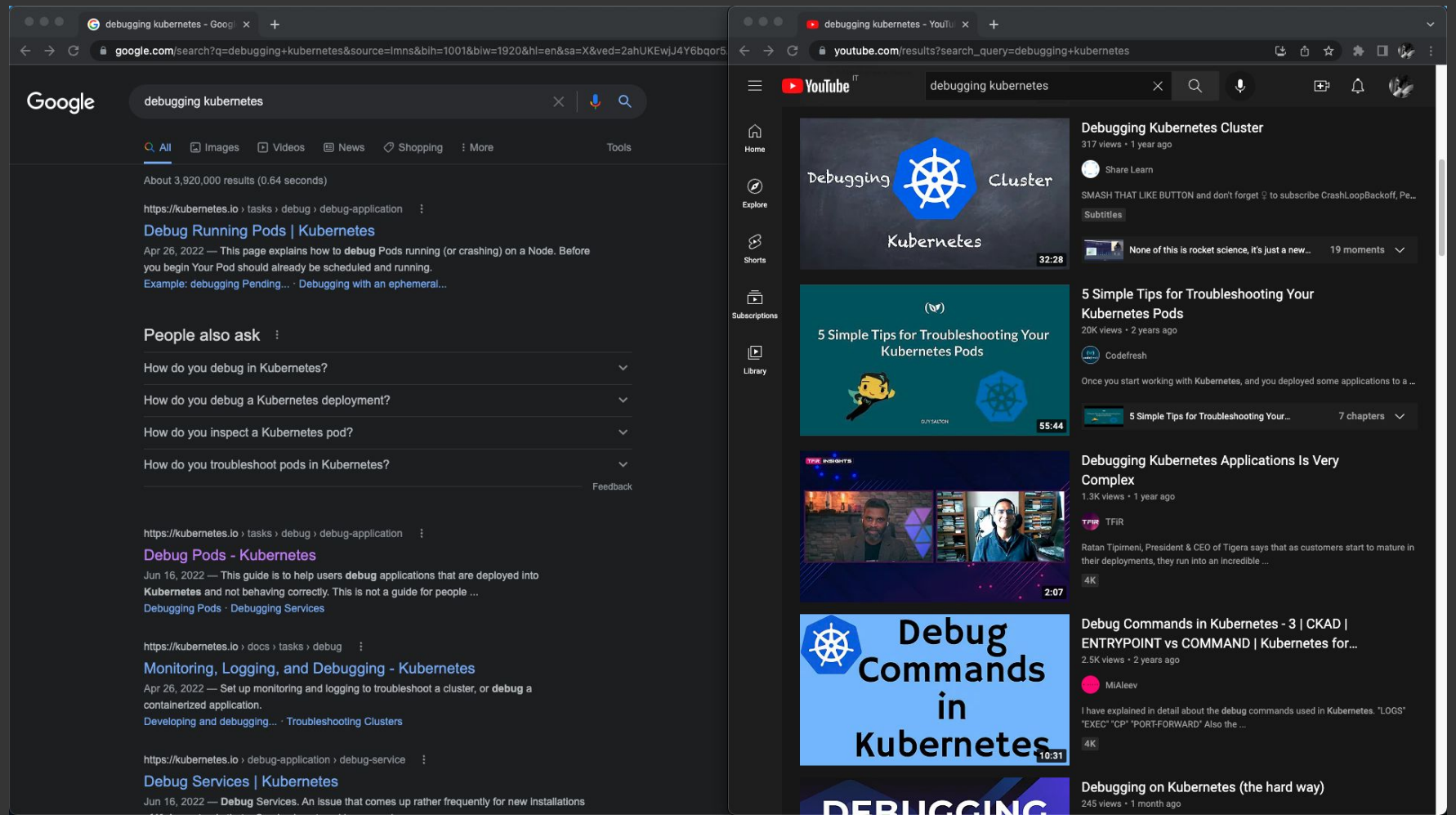
- A typical operation on the control plane involves creating multiple resources
- Having the cluster in an inconsistent state is fine, as Kubernetes will self-heal to the correct state given a correct definition of resources
- If the resources are defined wrongly, the cluster will be broken anyway

RuleOps Correlates Different Resource Types

- Database Rows
- Kubernetes Custom Resources
- Monitoring Data
- Logging

🔍 > “debugging Kubernetes”

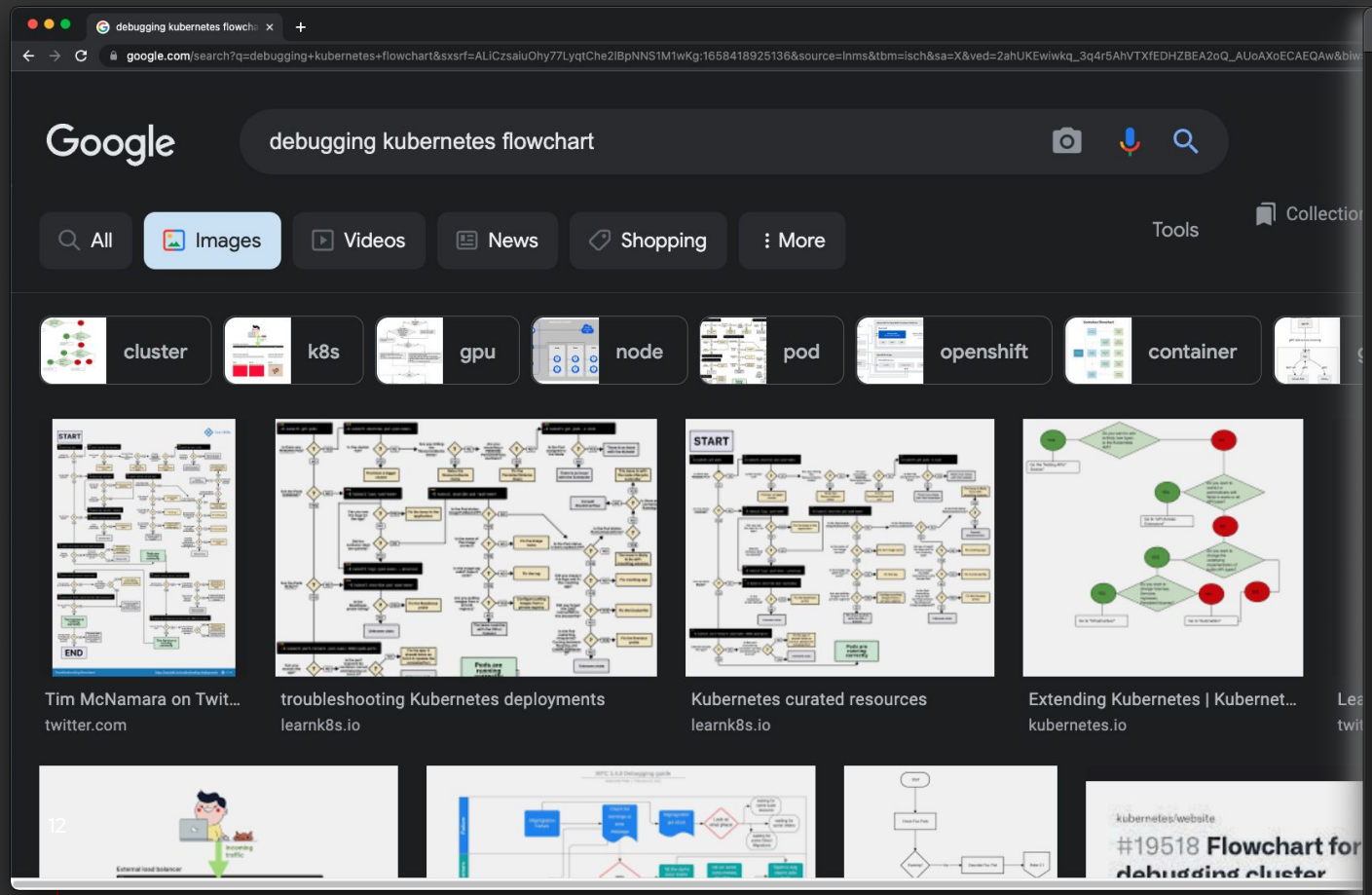
3+ Mil results indexed, counteless videos, “very complex”, “the hard way”, ...

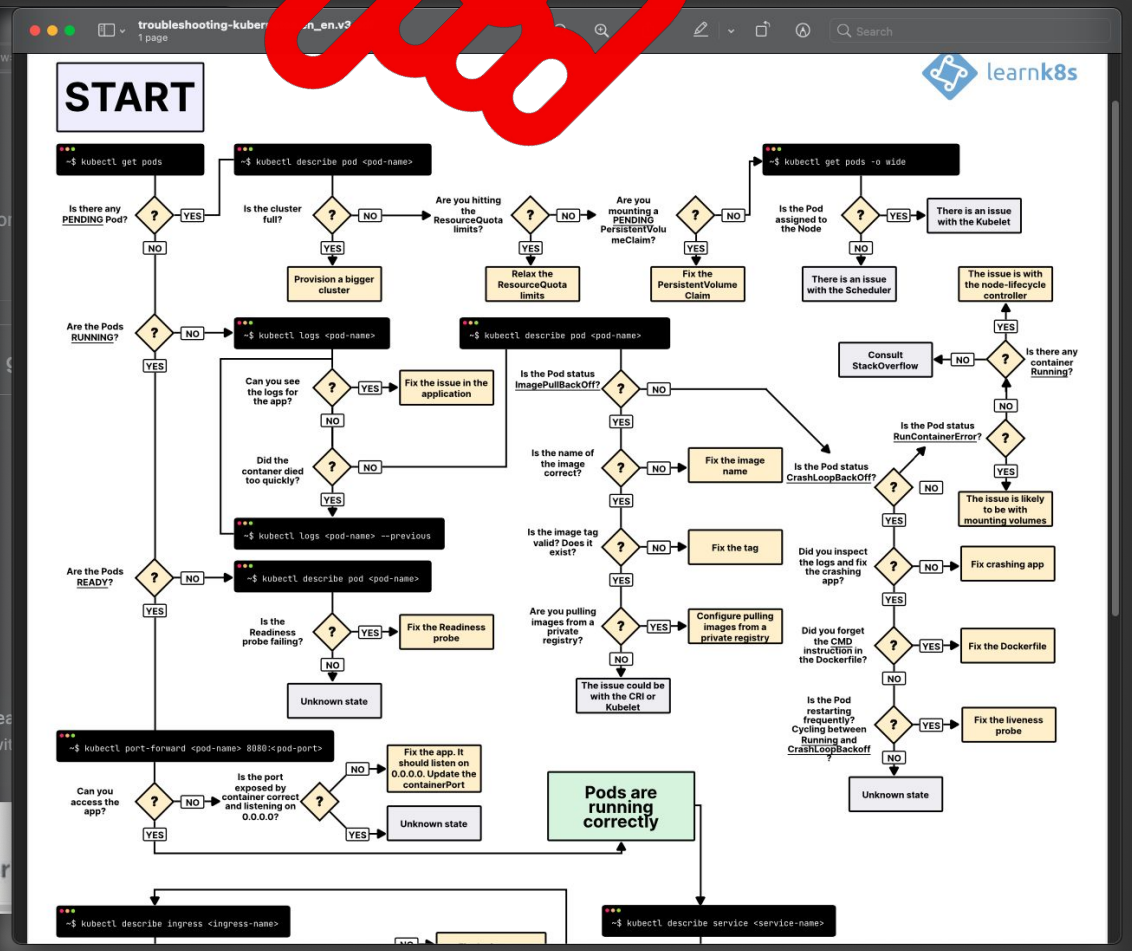


Red Hat

Some helpful resources online:

Notice a pattern here? 😉 automate it!





Procedural vs Declarative

Show me the code!

Demo 1:

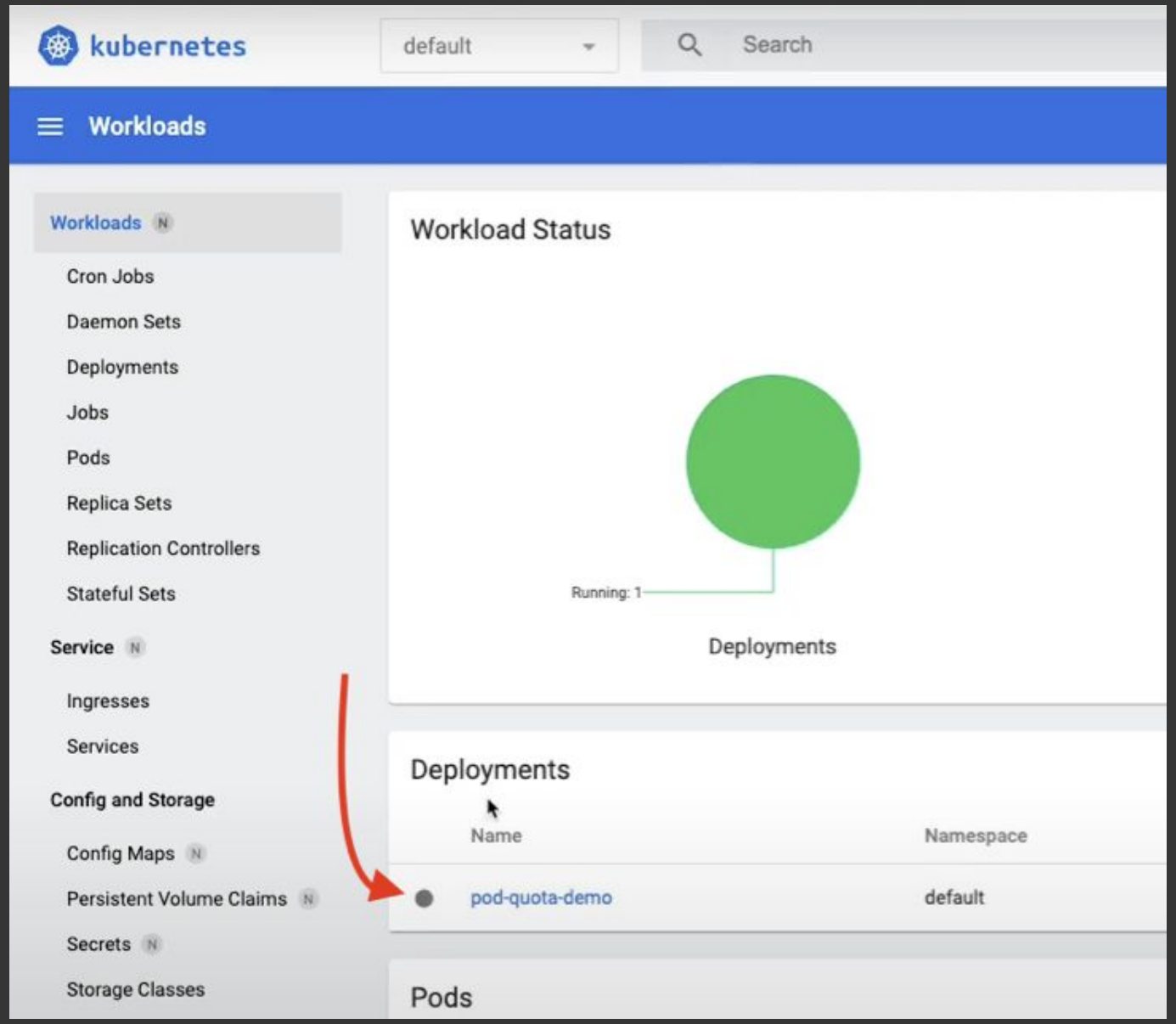
Inconsistent Number of Pods

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
 name: pod-demo
spec:
 hard:
  pods: "2"
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: pod-quota-demo
spec:
 selector:
   matchLabels:
     purpose: quota-demo
 replicas: 3
 template:
   metadata:
     labels:
       purpose: quota-demo
   spec:
     containers:
     - name: pod-quota-demo
       image: nginx
```


Red Hat

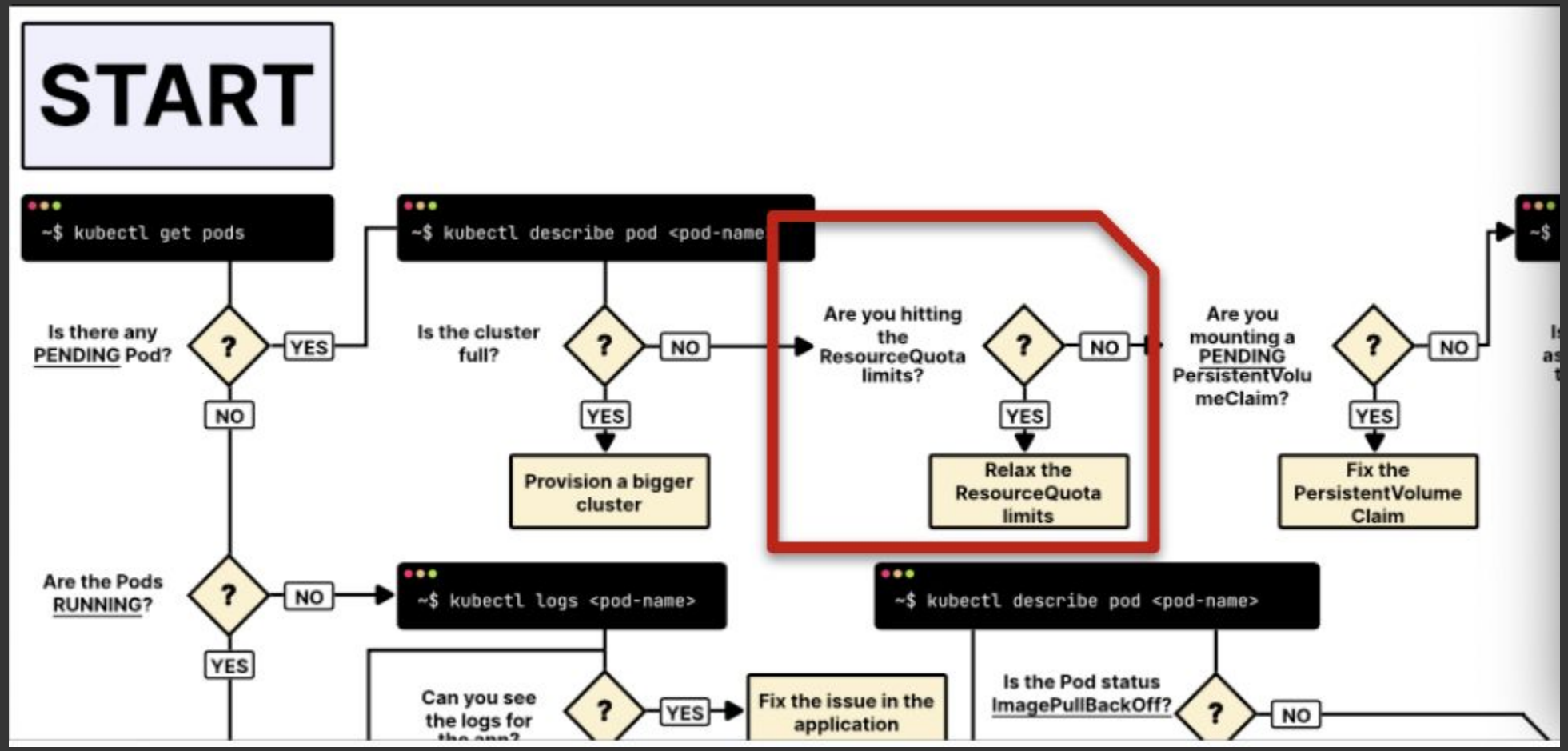
Demo I - Quota Limits



Red Hat

```
- name: Relax the ResourceQuota limits Deployment PENDING
  when:
  - given: Deployment
    as: $d
  - given: DeploymentCondition
    having:
    - type == "Available"
    - status == "False"
    from: $d.status.conditions
  - given: DeploymentCondition
    having:
    - message contains "exceeded quota"
    from: $d.status.conditions
  then: |
    insert(new Advice("Relax the ResourceQuota limits", ...
```

Demo I - Quota Limits



Red Hat

Demo 2:

Resource Quota Hit

Demo II - Resource Quota Hit

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
 name: mem-cpu-demo
spec:
 hard:
   requests.cpu: "1"
   requests.memory: 1Gi
   limits.cpu: "2"
   limits.memory: 2Gi
```


Red Hat

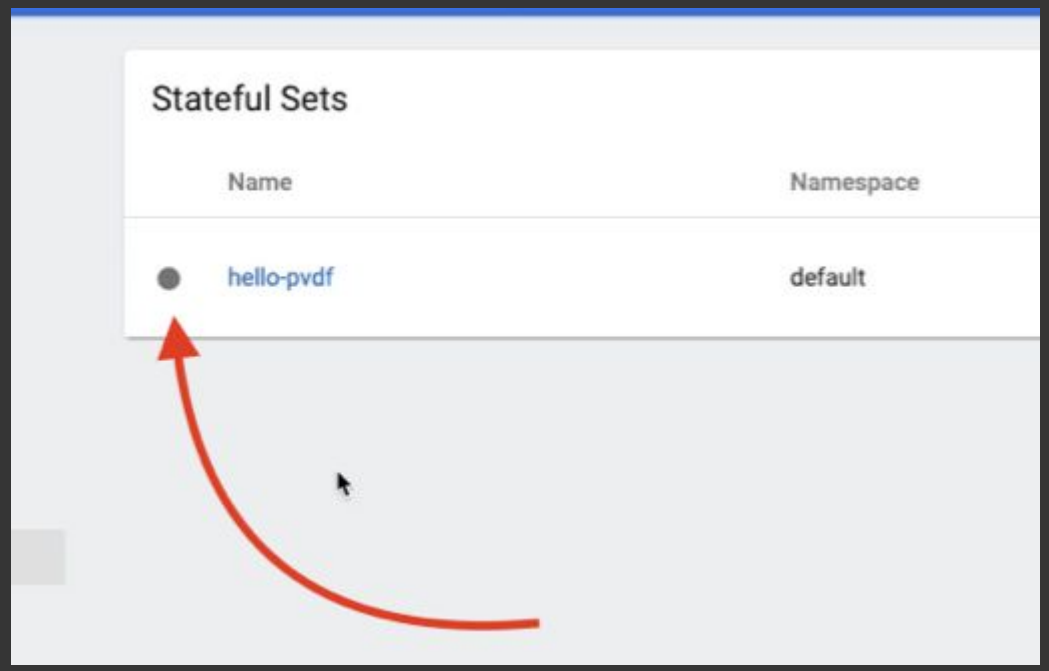
Demo II - Resource Quota Hit

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: quota-mem-cpu-demo
spec:
 containers:
   - name: quota-mem-cpu-demo-ctr
     image: nginx
     resources:
       limits:
         memory: "1Gi"
         cpu: "1"
       requests:
         memory: "600Mi"
         cpu: "500m"
```


Red Hat

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
 labels:
   app.kubernetes.io/version: 1.0.0-SNAPSHOT
   app.kubernetes.io/name: hello-pvdf
 name: hello-pvdf
 namespace: default
spec:
        resources:
          limits:
            cpu: 2000m
            memory: 2Gi
          requests:
            cpu: 500m
            memory: 500Mi
        volumeMounts:
          - mountPath: /mnt/data
            name: my-pvc-claim
            readOnly: false
```

Demo II - Resource Quota Hit



Red Hat

```
- name: Relax the ResourceQuota limits StatefulSet PENDING
  when:
  - given: StatefulSet
    as: $s
    having:
    - spec.replicas != status.replicas
  - given: Event
    having:
    - message contains "exceeded quota"
    from: DroolsK8sClient.eventsFor($s)
  then: |
    insert(new Advice("Relax the ResourceQuota limits", ...
```

Demo 3:

Persistence Volume not set

Demo III - Persistence Volume Not set

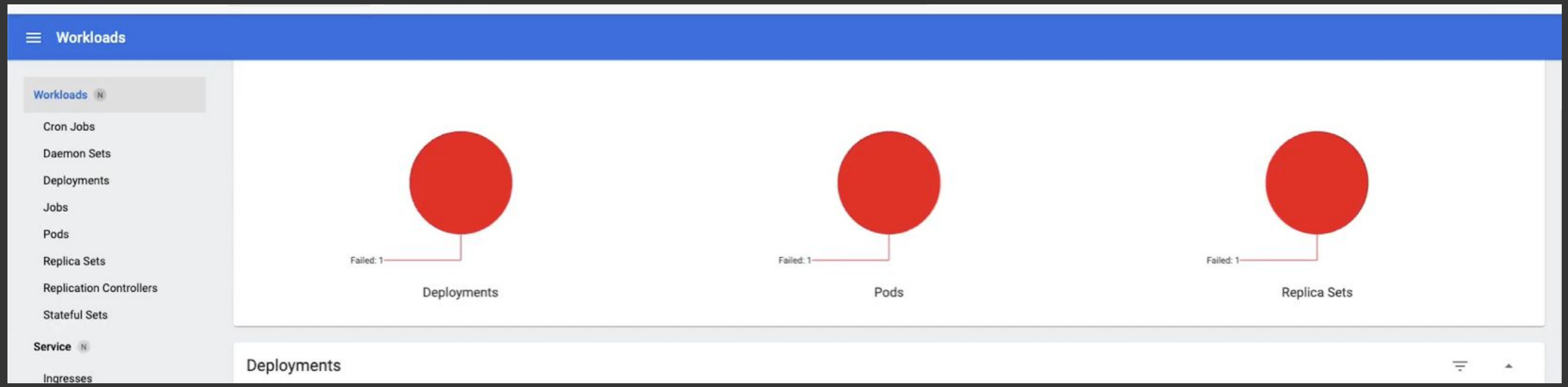
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: task-pv-claim
spec:
 storageClassName: manual
 accessModes:
   - ReadWriteOnce
 resources:
   requests:
     storage: 3Gi
```


Red Hat

Demo III - Persistence Volume Not set

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: pod-quota-demo
spec:
 selector:
   matchLabels:
     purpose: quota-demo
 replicas: 1
 template:
   metadata:
     labels:
       purpose: quota-demo
   spec:
     volumes:
     - name: task-pv-storage
       persistentVolumeClaim:
         claimName: task-pv-claim
     containers:
     - name: pod-quota-demo
       image: nginx
       ports:
         - containerPort: 80
           name: "http-server"
       volumeMounts:
         - mountPath: "/usr/share/nginx/html"
           name: task-pv-storage
```

Demo III - Persistence Volume Not set



Red Hat

```
- name: Fix the PersistentVolumeClaim Pod PENDING
  when:
  - given: PersistentVolumeClaim
    as: $pvc
    having:
    - status.phase == "Pending"
  - given: Pod
    as: $pod
    having:
    - status.phase == "Pending"
  - given: Volume
    having:
    - persistentVolumeClaim!.claimName == $pvc.metadata.name
    from: $pod.spec.volumes
  then: |
    insert(new Advice("Fix the PersistentVolume", ...
```

Further development of RuleOps

CEP (Complex Events Processing)

- Correlating events of a monitoring system
- Temporal operators
  - Before, After
  - Interval ranges
- Can highlight internals of Kubernetes while used with the controllers' API

RuleOps Deployment Models

- Sidecar application
- CLI
- As a Kubernetes Controller
- As a Kubernetes plugin?

Performance / Efficiency

- Drools is fast
- Based on Drools Executable Model
- Fetch only the needed Kubernetes resource types
  - If a rule validates Pods, it will fetch only Pods from the Control Plane, therefore reducing the amount of data passing

Level trigger vs Edge Trigger

https://medium.com/hackernoon/level-triggering-and-reconciliation-in-kubernetes-1f17fe30333d

36

Red Hat

Matteo Mortari

Principal Sw Engineer

Luca Molteni

Principal Sw Engineer

Daniele Zonca

Sr Principal Sw Engineer, Architect

Red Hat

https://github.com/kiegroup/ruleops

38

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat