Back to Basics: Hands-On Deployment of Stateful Workloads on Kubernetes

David Zhu, Google & Jan Šafránek, Red Hat

Agenda

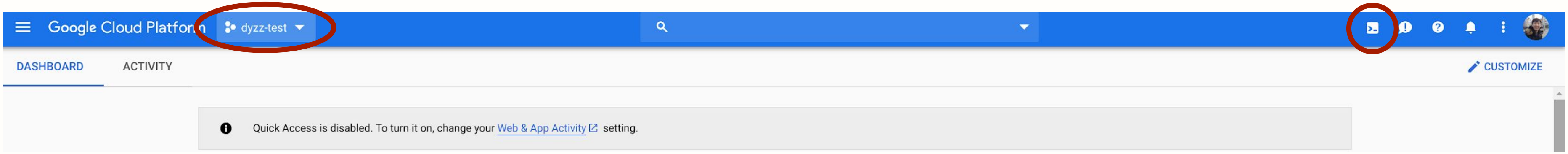
- Create a Cluster
- Basic Stateful Workload Concepts
- Dynamic Provisioning
- Higher Level Workload Concepts
- Kubectl
- Common Debugging Techniques
- Our Cassandra Demo App Hands-On
- Other databases
- Advanced Topics

First we need a Kubernetes cluster.

Log in to Google Cloud Platform

`https://console.cloud.google.com`

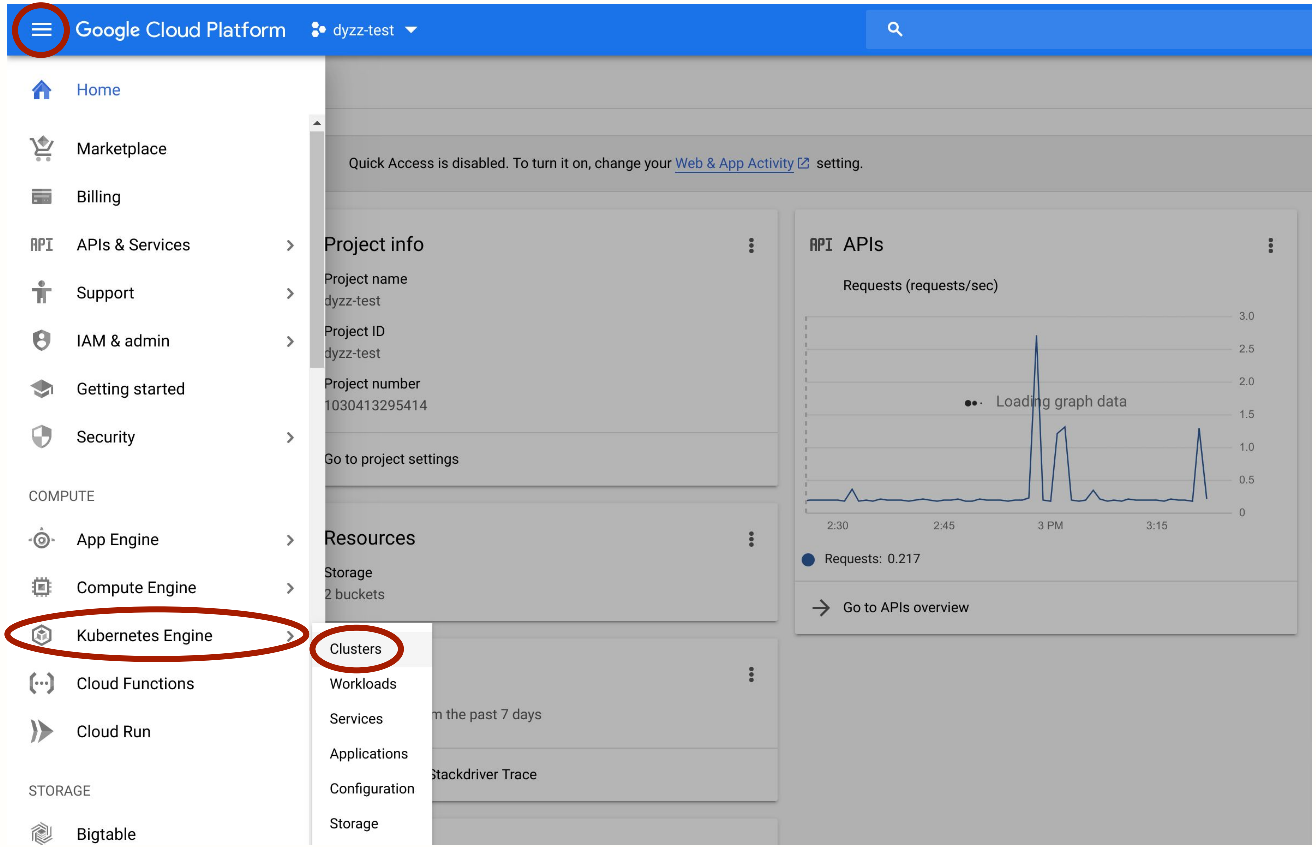
Select your project & Start cloud console



Quick Access is disabled. To turn it on, change your Web & App Activity setting.

A Basic Stateful Workload

Navigate to Kubernetes Engine



Quick Access is disabled. To turn it on, change your Web & App Activity setting.

Project info

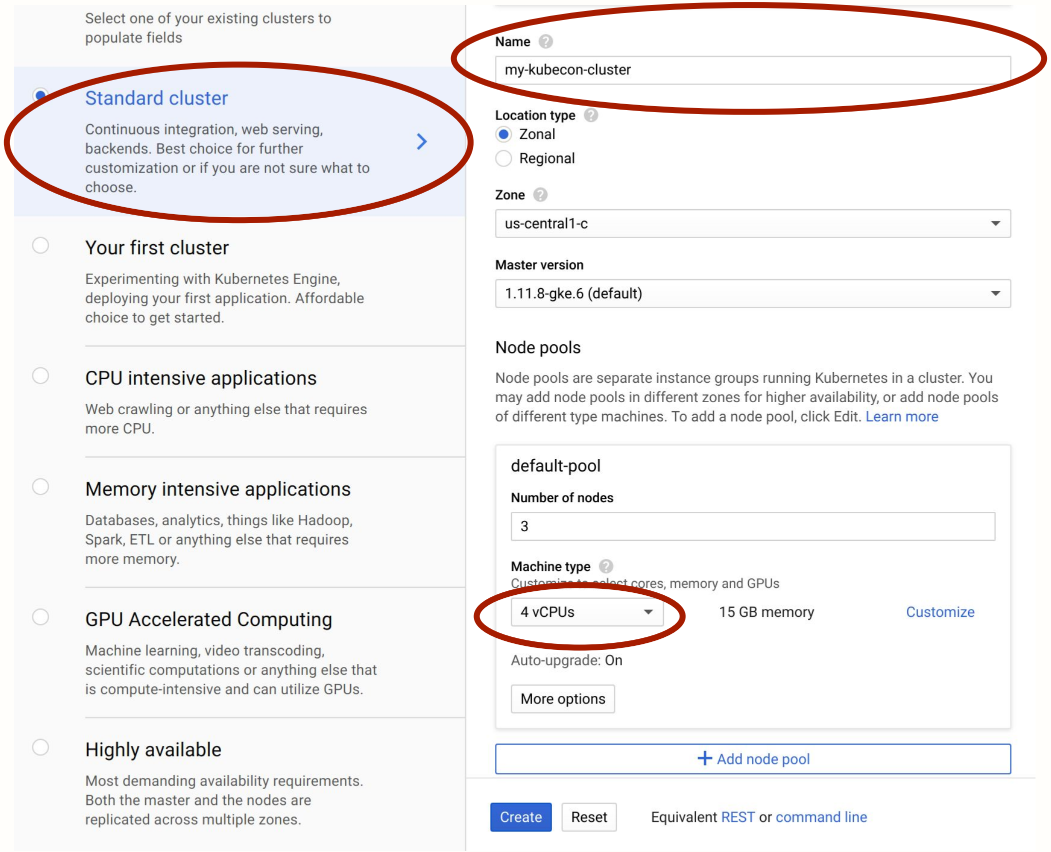
Project name
dyzz-test

Project ID
dyzz-test

Project number
1030413295414

Storage
2 buckets

Create a cluster



Select one of your existing clusters to populate fields

● **Standard cluster**

Continuous integration, web serving, backends. Best choice for further customization or if you are not sure what to choose.

○ **Your first cluster**

Experimenting with Kubernetes Engine, deploying your first application. Affordable choice to get started.

○ **CPU intensive applications**

Web crawling or anything else that requires more CPU.

○ **Memory intensive applications**

Databases, analytics, things like Hadoop, Spark, ETL or anything else that requires more memory.

○ **GPU Accelerated Computing**

Machine learning, video transcoding, scientific computations or anything else that is compute-intensive and can utilize GPUs.

○ **Highly available**

Most demanding availability requirements. Both the master and the nodes are replicated across multiple zones.

Name

my-kubecon-cluster

Location type

● Zonal

○ Regional

Zone

us-central1-c

Master version

1.11.8-gke.6 (default)

Node pools

Node pools are separate instance groups running Kubernetes in a cluster. You may add node pools in different zones for higher availability, or add node pools of different type machines. To add a node pool, click Edit. Learn more

default-pool

Number of nodes

3

Machine type

Customize to select cores, memory and GPUs

4 vCPUs 15 GB memory Customize

Auto-upgrade: On

Equivalent REST or command line

Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sleepypod
spec:
  containers:
    - name: sleepycontainer
      image: gcr.io/google_containers/busybox
      command:
        - sleep
        - "6000"
      volumeMounts:
        - name: data
          mountPath: /data
          readOnly: false
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: mypvc
```

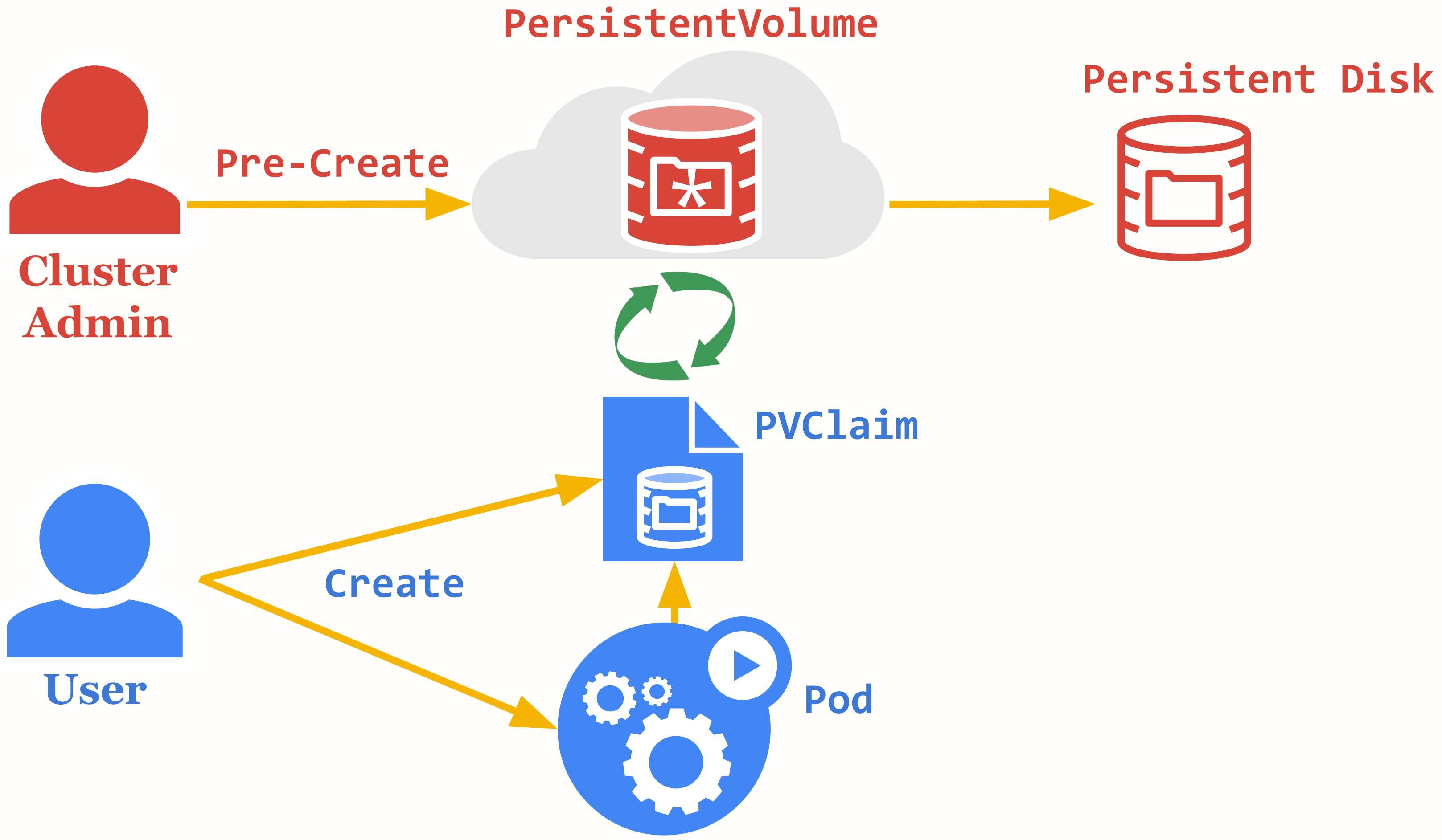
PersistentVolumeClaim

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mypvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 5Gi
```

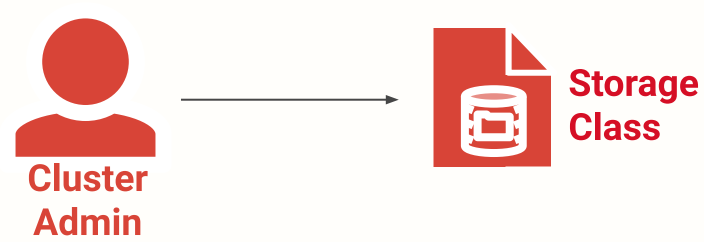
Binder

PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mypv
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  nfs:
    path: /tmp
    server: 172.17.0.2
```


PersistentVolume
Persistent Disk
Pre-Create
Cluster Admin
PVClaim
Create
User
Pod

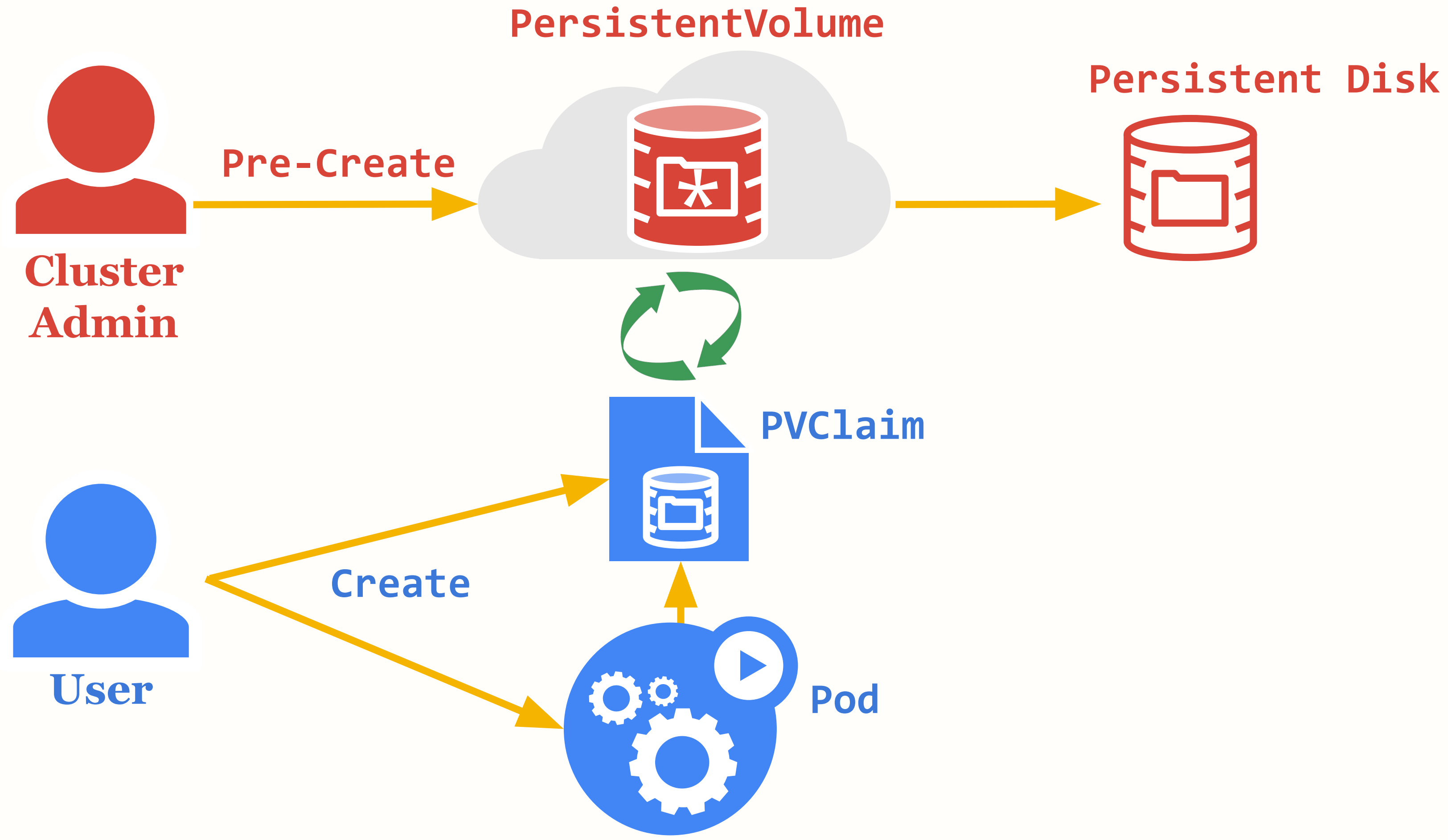
Dynamic Provisioning



Cluster Admin

Storage Class

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard (hdd)

--

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: fast
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
```



PersistentVolume

Persistent Disk

Pre-Create

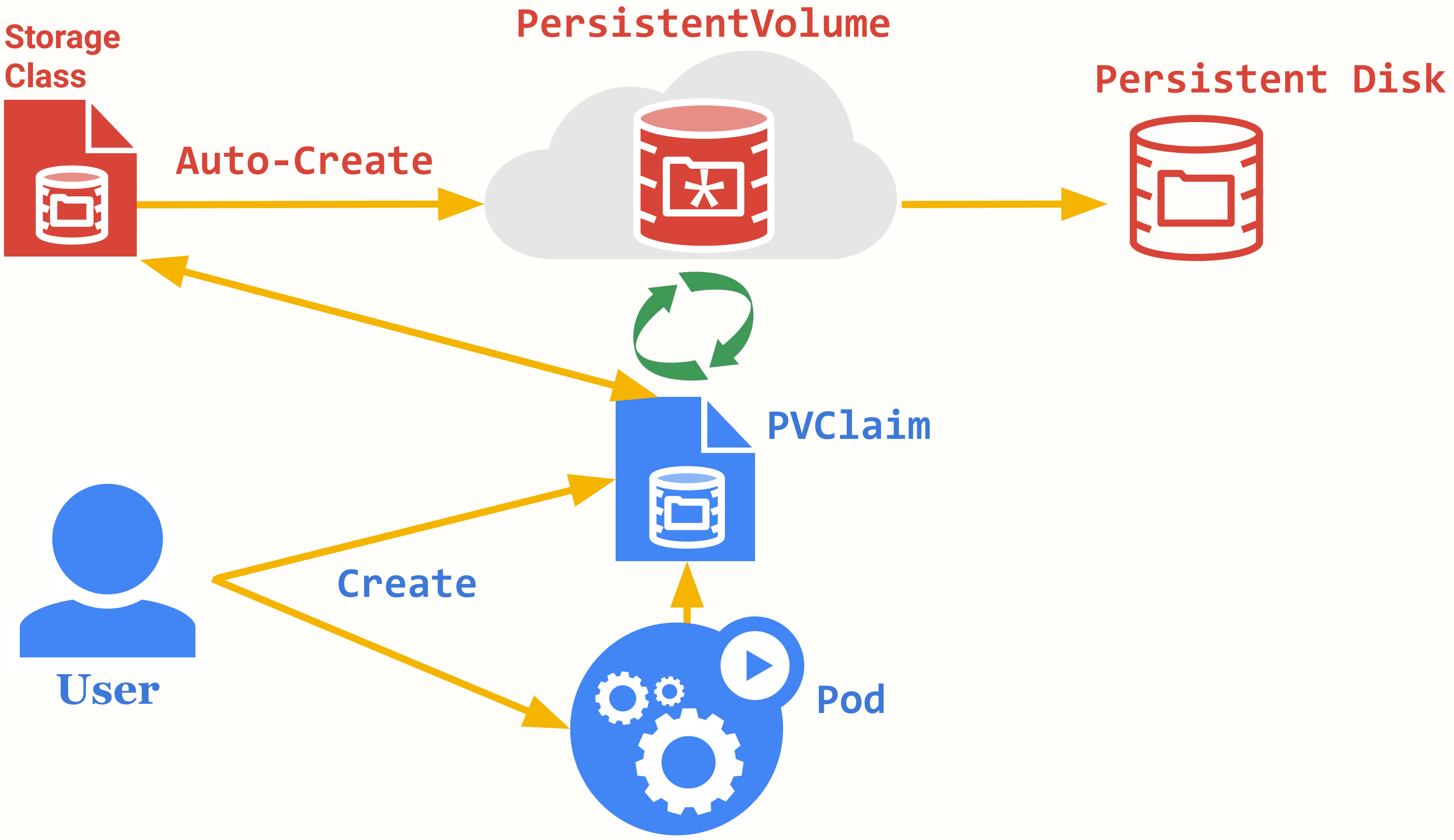
Cluster Admin

PVClaim

Create

User

Pod


Storage Class
PersistentVolume
Persistent Disk
Auto-Create
PVClaim
User
Create
Pod

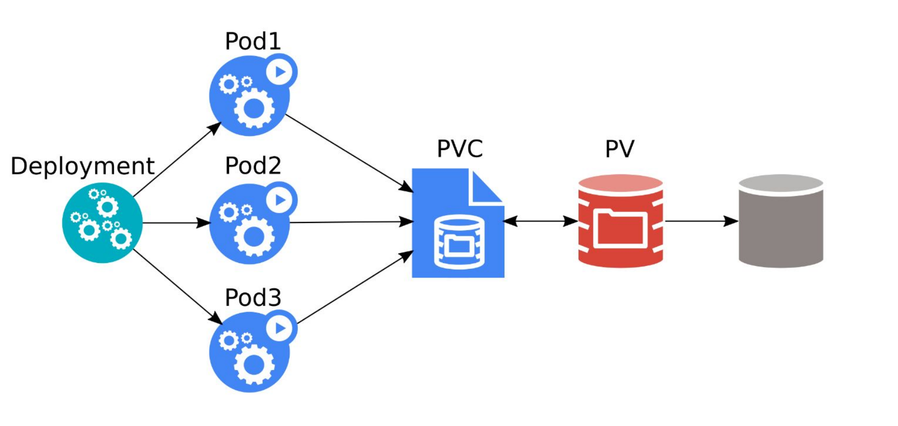
Benefits of Dynamic Provisioning

- Decrease overhead by only creating disks (on clouds) when they are requested by a workload
- Grouping of volumes by storage characteristics
- Decrease cluster admin burden of pre-provisioning Persistent Volumes for each underlying infrastructure disk

Higher Level Workloads

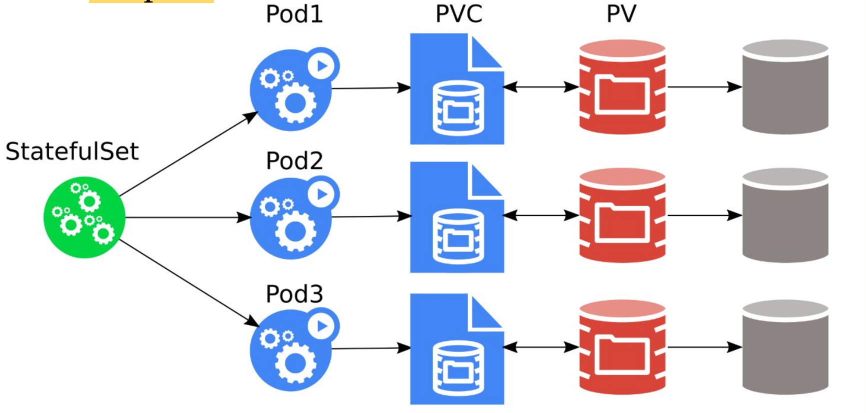
Deployments

- Runs X replicas of a single Pod template
- When a pod is deleted, Deployment automatically creates a new one
- Scalable up and down
- All pods share the same PVC



Statefulset

- Runs X replicas of a single Pod template
- When a pod is deleted, StatefulSet automatically creates a new one
- Each pod has a stable identity
- Scalable up and down
- Each pod gets its own PVC(s) from a PVC template



Services

Load Balancer

- Defines logical set of pods and a policy by which to access them (micro-service)
- Abstracts away fungible ephemeral pods
- Exposes a single cluster IP externally using the cloud providers load balancer automatically

Headless

- Defines logical set of pods and a policy by which to access them (micro-service)
- Exposes IPs of each pod for discoverability

What do I do with all these Objects?

kubectl

Useful `kubectl` commands:

```
kubectl apply -f {YAMLFile}
```

- Apply an object defined by YAMLFile onto your cluster

```
kubectl delete -f {YAMLFile}
```

- Delete an object with name/type defined by YAMLFile in the cluster

```
kubectl get {APIObject}
```

- Get basic list of API objects

```
kubectl describe {APIObject}
```

- Get more details and error events

```
kubectl logs {PodName} {ContainerName}
```

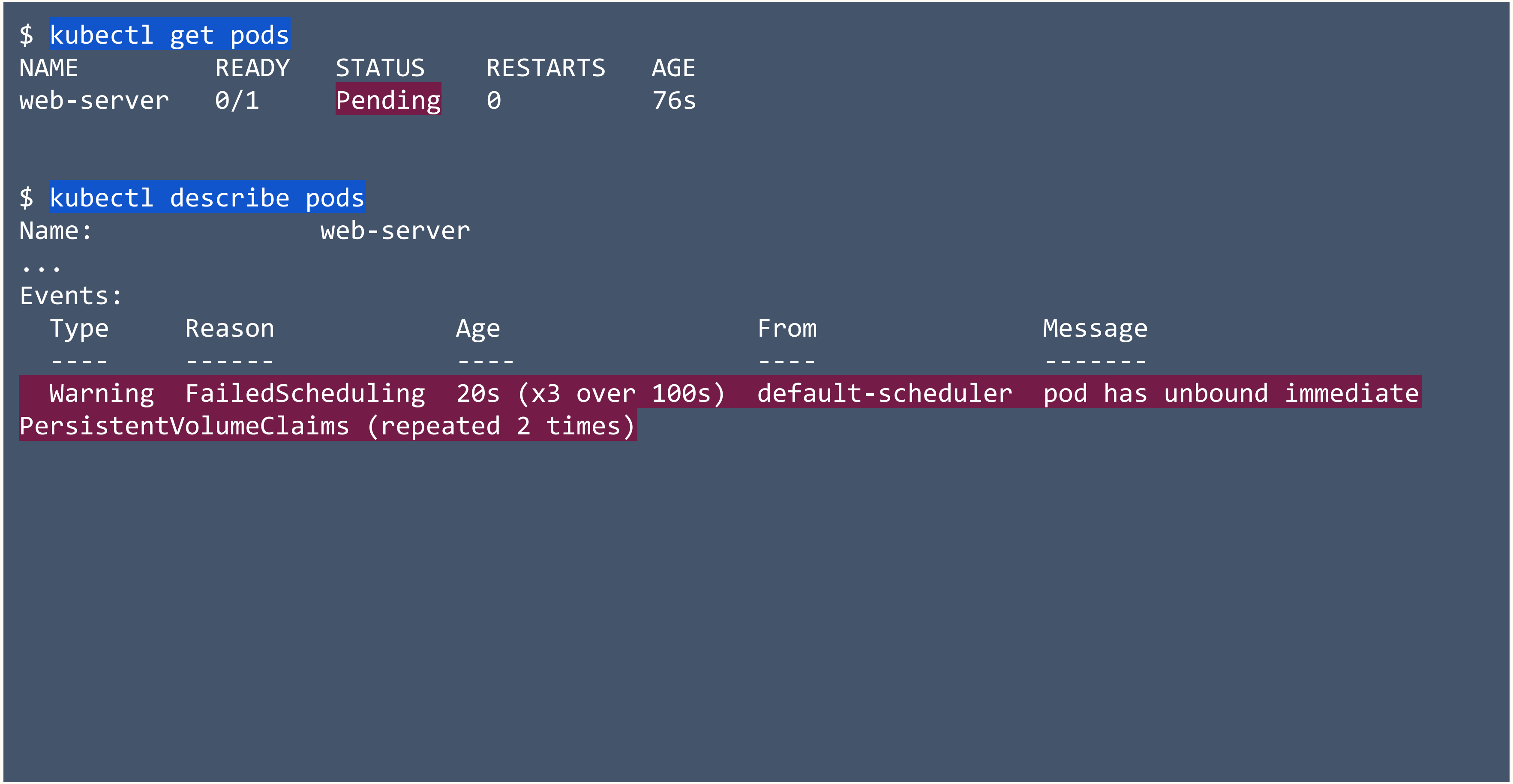
- Get stdout from container for debugging

```
kubectl exec {PodName} -c {ContainerName} -- {Command}
```

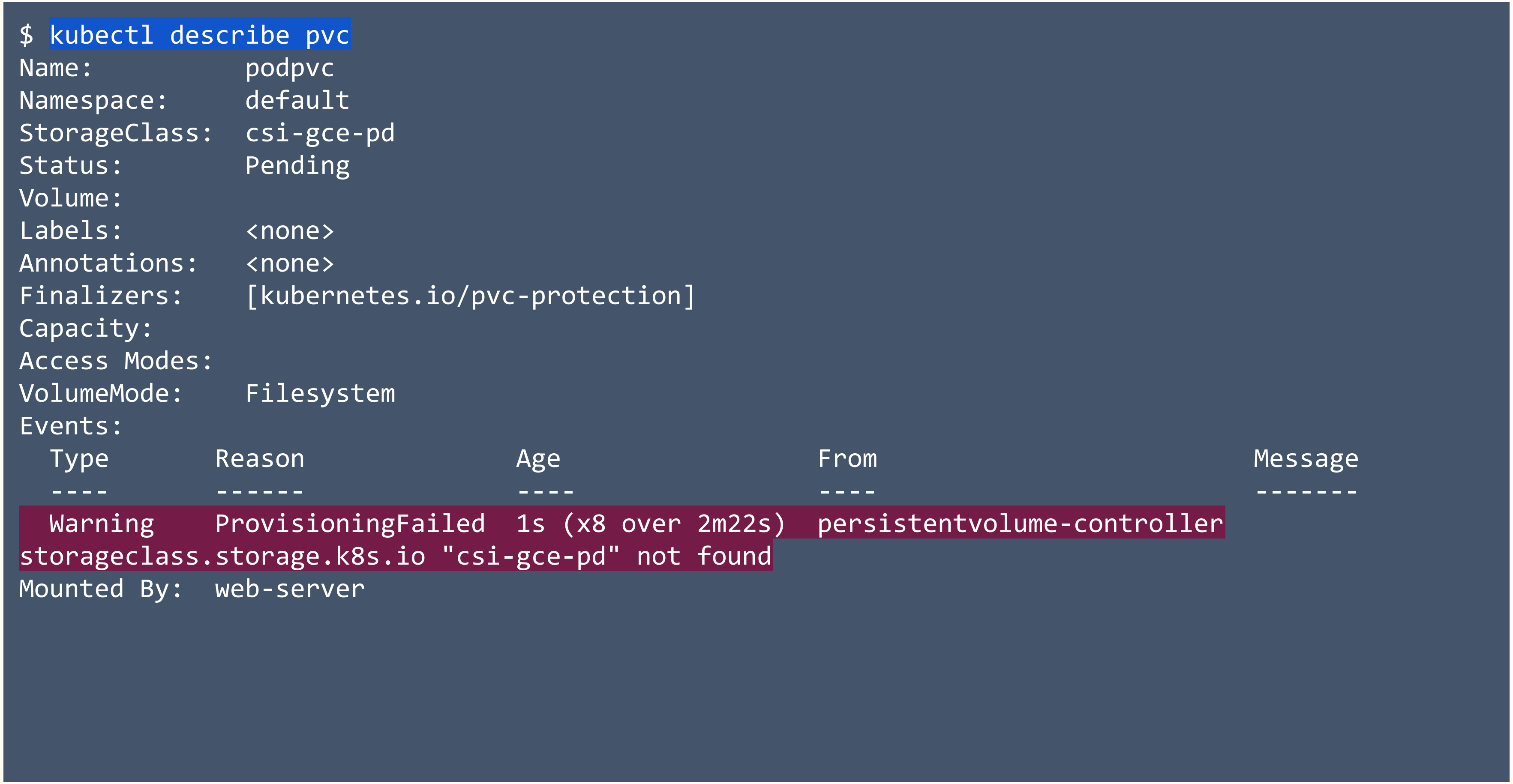
- Execute command directly in a container, such as “ls” or “/bin/sh”

https://kubernetes.io/docs/reference/kubectl/cheatsheet/

A Common Debugging Technique



```
$ kubectl get pods
NAME         READY   STATUS     RESTARTS   AGE
web-server   0/1     Pending    0          76s


$ kubectl describe pods
Name:                web-server
...
Events:
  Type     Reason            Age                 From               Message
  ----     ------            ----                ----               -------
  Warning  FailedScheduling  20s (x3 over 100s)  default-scheduler  pod has unbound immediate
PersistentVolumeClaims (repeated 2 times)
```



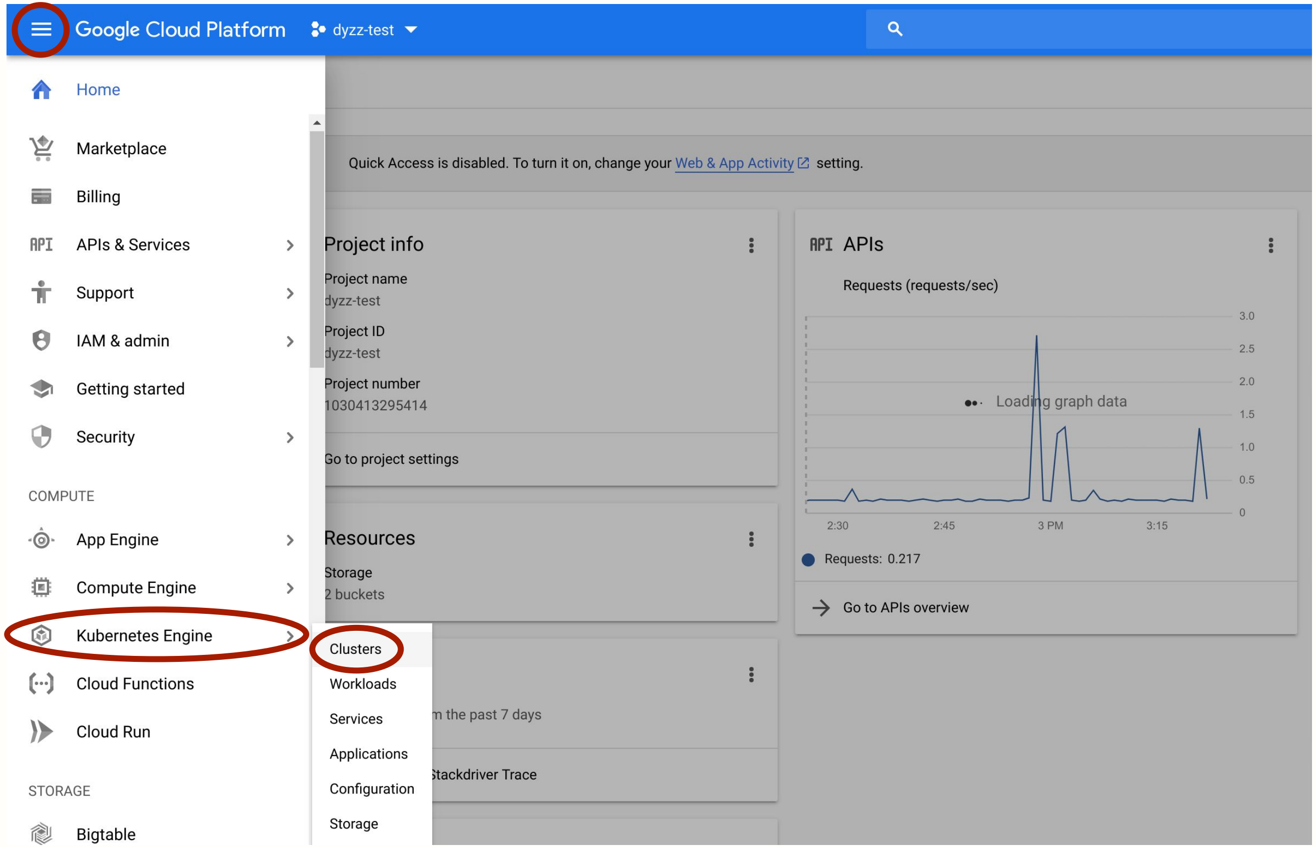
```
$ kubectl describe pvc
Name:          podpvc
Namespace:     default
StorageClass:  csi-gce-pd
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Events:
  Type       Reason              Age                 From                         Message
  ----       ------              ----                ----                         -------
  Warning    ProvisioningFailed  1s (x8 over 2m22s)  persistentvolume-controller
storageclass.storage.k8s.io "csi-gce-pd" not found
Mounted By:  web-server
```

```
$ kubectl get storageclass
No resources found.

$ kubectl apply -f examples/kubernetes/demo-zonal-sc.yaml
storageclass.storage.k8s.io/csi-gce-pd created

$ kubectl describe storageclass
Name:            csi-gce-pd
IsDefaultClass:  No
...
Provisioner:           pd.csi.storage.gke.io
Parameters:            type=pd-standard
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
Events:                <none>
```

```
$ kubectl describe pods web-server
Name:               web-server
...
Events:
  Type     Reason                  Age                 From
Message
  ----     ------                  ----                ----
-------
  Warning  FailedScheduling        8m3s (x4 over 10m)  default-scheduler                      pod
has unbound immediate PersistentVolumeClaims (repeated 2 times)
  Normal   Scheduled               67s                 default-scheduler
Successfully assigned default/web-server to kubernetes-minion-group-mvmb
  Normal   SuccessfulAttachVolume  54s                 attachdetach-controller
AttachVolume.Attach succeeded for volume "pvc-58880c4d-92a5-45b1-a1b0-baf3ab2e91dd"
  Normal   Pulling                 49s                 kubelet, kubernetes-minion-group-mvmb
Pulling image "nginx"
  Normal   Pulled                  47s                 kubelet, kubernetes-minion-group-mvmb
Successfully pulled image "nginx"
  Normal   Created                 46s                 kubelet, kubernetes-minion-group-mvmb
Created container web-server
  Normal   Started                 46s                 kubelet, kubernetes-minion-group-mvmb
Started container web-server
```

Let’s try the demo together.

Navigate to Kubernetes Engine



Quick Access is disabled. To turn it on, change your Web & App Activity setting.

Project name
dyzz-test

Project ID
dyzz-test

Project number
1030413295414

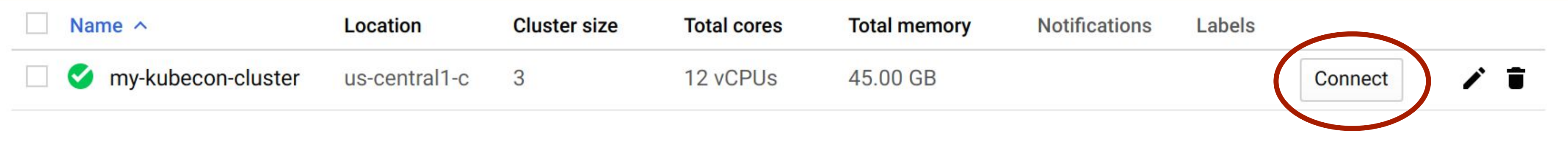
Storage
2 buckets

m the past 7 days

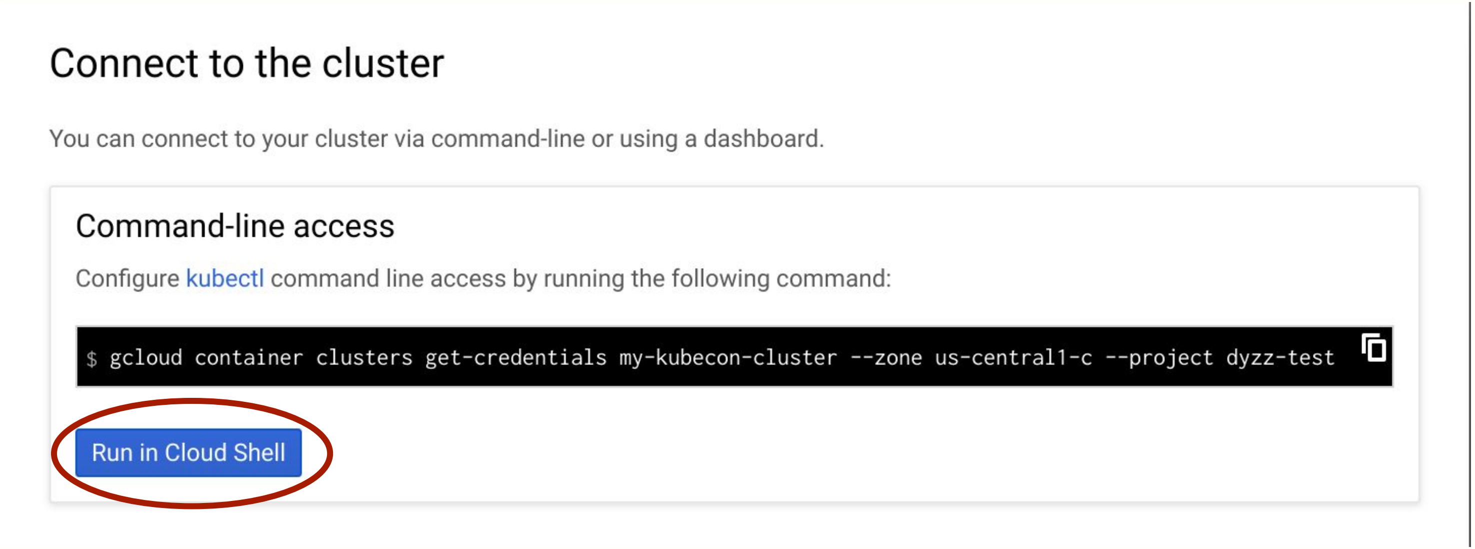
Stackdriver Trace

Requests: 0.217

Connect to your cluster



□	Name ^	Location	Cluster size	Total cores	Total memory	Notifications	Labels		
□	✓ my-kubecon-cluster	us-central1-c	3	12 vCPUs	45.00 GB			Connect	✎ 🗑



Connect to the cluster

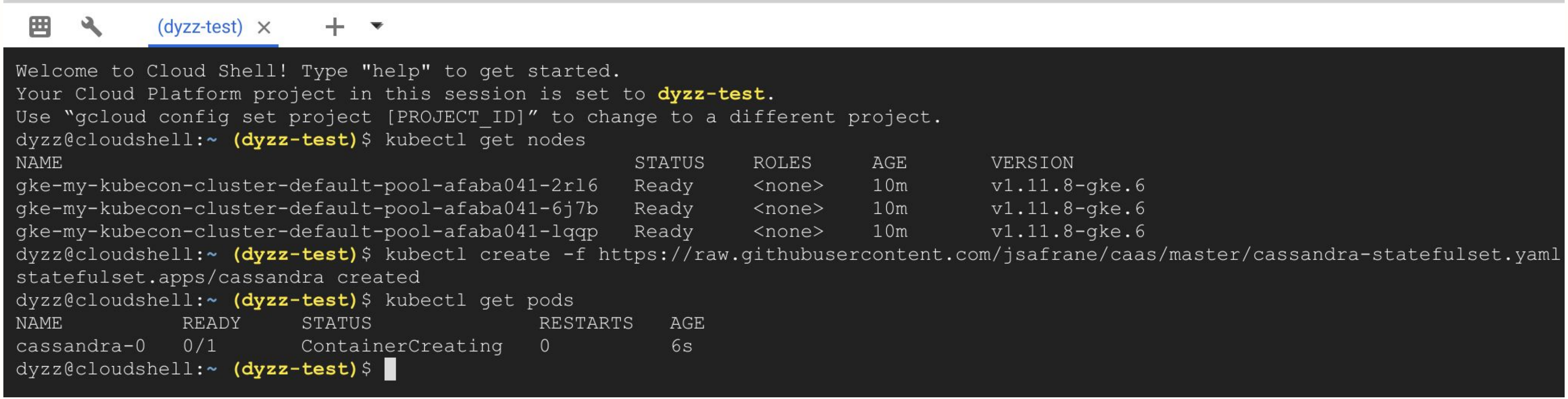
You can connect to your cluster via command-line or using a dashboard.

Command-line access

Configure kubectl command line access by running the following command:

```
$ gcloud container clusters get-credentials my-kubecon-cluster --zone us-central1-c --project dyzz-test
```

Verify.



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to dyzz-test.
Use “gcloud config set project [PROJECT_ID]” to change to a different project.
dyzz@cloudshell:~ (dyzz-test)$ kubectl get nodes
NAME                                                 STATUS    ROLES     AGE       VERSION
gke-my-kubecon-cluster-default-pool-afaba041-2rl6   Ready     <none>    10m       v1.11.8-gke.6
gke-my-kubecon-cluster-default-pool-afaba041-6j7b   Ready     <none>    10m       v1.11.8-gke.6
gke-my-kubecon-cluster-default-pool-afaba041-lqqp   Ready     <none>    10m       v1.11.8-gke.6
dyzz@cloudshell:~ (dyzz-test)$ kubectl create -f https://raw.githubusercontent.com/jsafrane/caas/master/cassandra-statefulset.yaml
statefulset.apps/cassandra created
dyzz@cloudshell:~ (dyzz-test)$ kubectl get pods
NAME          READY     STATUS              RESTARTS   AGE
cassandra-0   0/1       ContainerCreating   0          6s
dyzz@cloudshell:~ (dyzz-test)$
```

We will now interact with Kubernetes through `kubectl`. The following is consistent across any conformant Kubernetes cluster.

Let’s deploy a stateful app.

github.com/jsafrane/caas

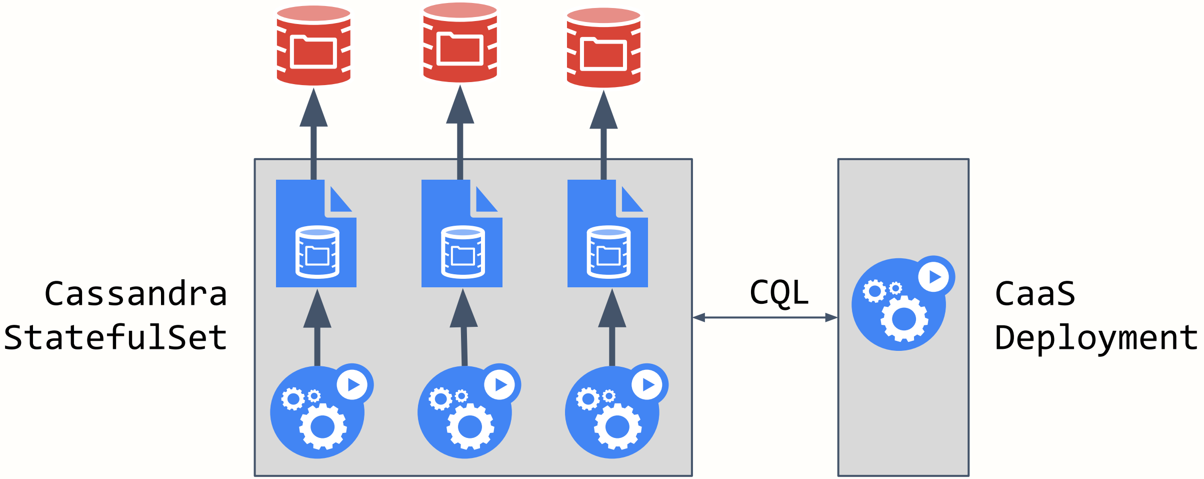
Goal:

Very simple web application: “Counter as a Service”.

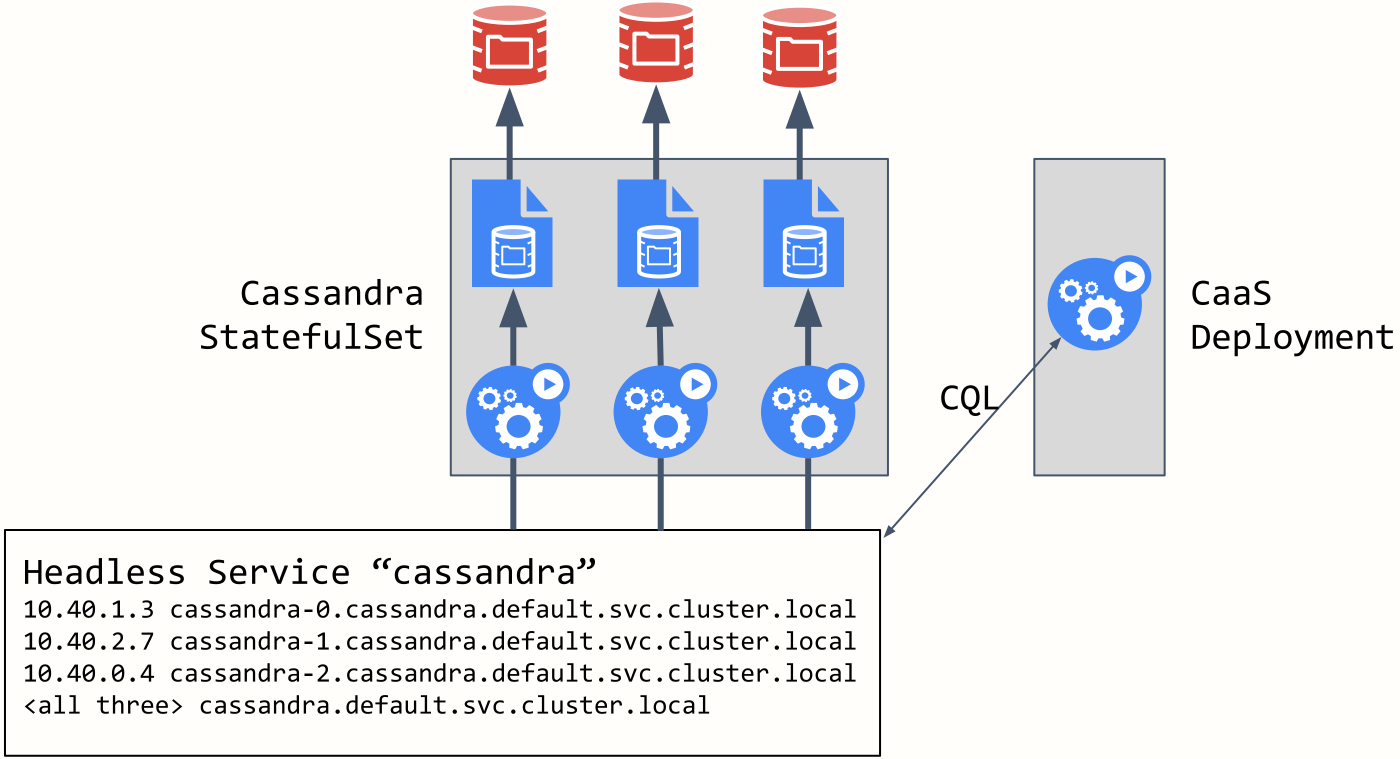
- Storing one integer in a DB.
- DB = Apache Cassandra
  - Easy to set up.
  - Easy to show StatefulSet concepts.

github.com/jsafrane/caas

Goal



Goal



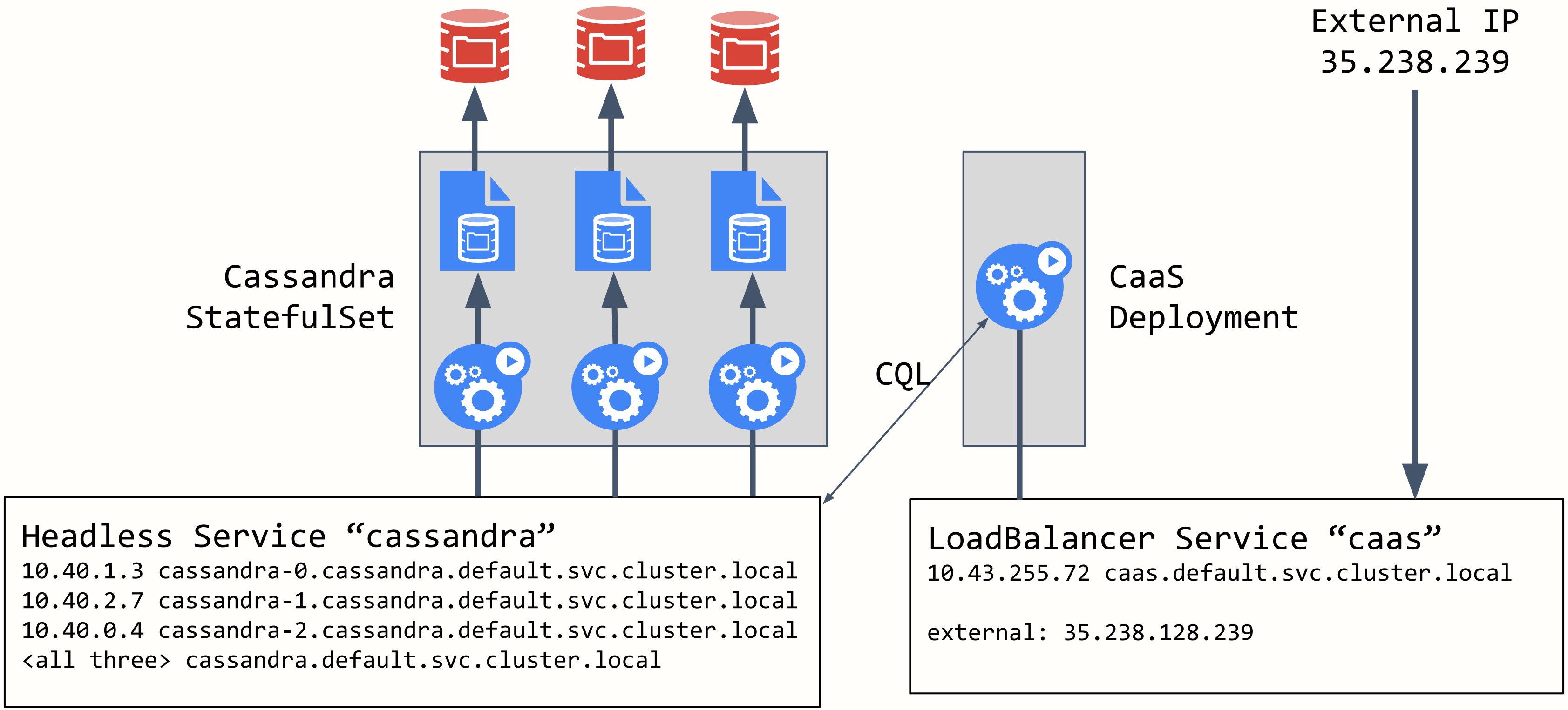
Cassandra
StatefulSet

CQL

CaaS
Deployment

Headless Service “cassandra”
10.40.1.3 cassandra-0.cassandra.default.svc.cluster.local
10.40.2.7 cassandra-1.cassandra.default.svc.cluster.local
10.40.0.4 cassandra-2.cassandra.default.svc.cluster.local
<all three> cassandra.default.svc.cluster.local

Goal



External IP
35.238.239

Cassandra
StatefulSet

CaaS
Deployment

CQL

Headless Service “cassandra”
10.40.1.3 cassandra-0.cassandra.default.svc.cluster.local
10.40.2.7 cassandra-1.cassandra.default.svc.cluster.local
10.40.0.4 cassandra-2.cassandra.default.svc.cluster.local
<all three> cassandra.default.svc.cluster.local

LoadBalancer Service “caas”
10.43.255.72 caas.default.svc.cluster.local

external: 35.238.128.239

Cassandra App Backend StatefulSet

Launch 3 copies of the template

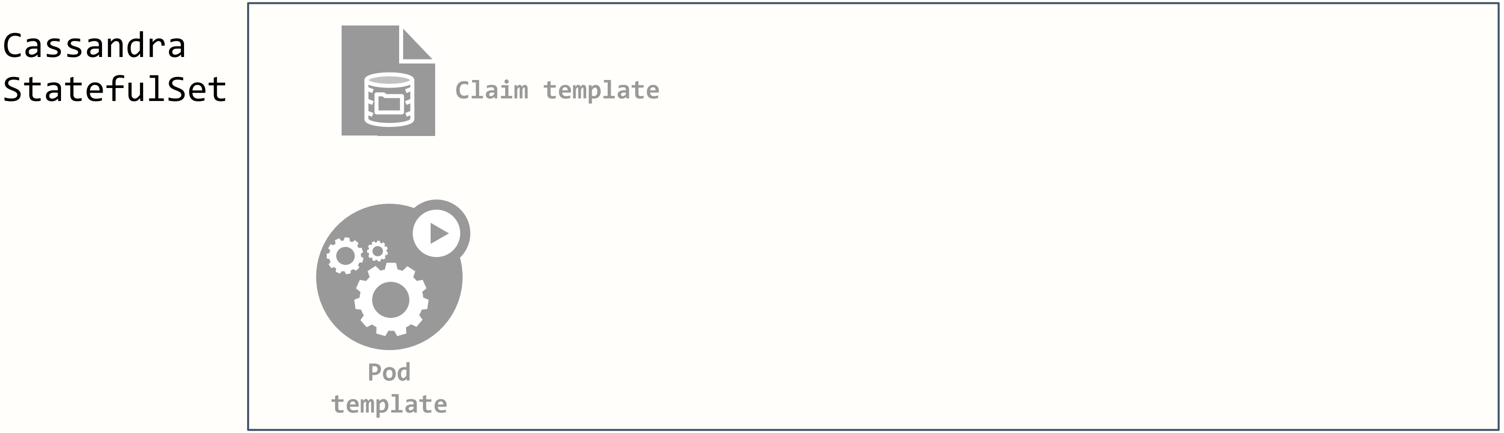
Kubernetes creates one PersistentVolume for each VolumeClaimTemplate

```
kind: StatefulSet
metadata:
  name: cassandra
spec:
  …
  replicas: 3
  template:
    spec:
      containers:
      - name: cassandra
        image: gcr.io/google-samples/cassandra:v13
        ...
        volumeMounts:
        - name: cassandra-data
          mountPath: /cassandra_data
  volumeClaimTemplates:
  - metadata:
      name: cassandra-data
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "standard"
      resources:
        requests:
          storage: 1Gi
```

Where to mount each volume in each container replica

Storageclass used to provision the 3 volumes

Initial StatefulSet

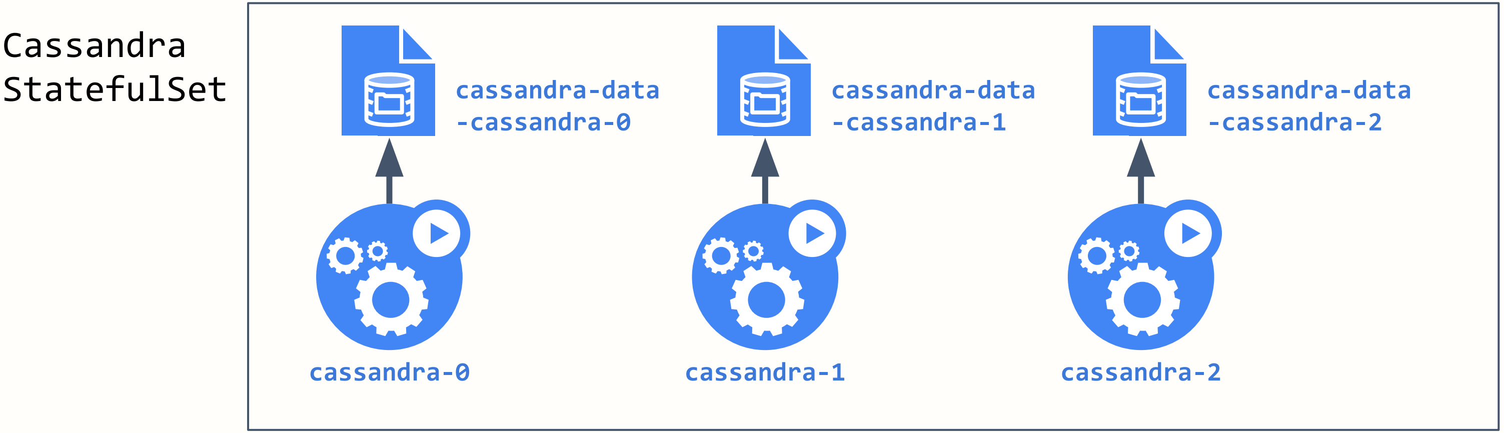


Cassandra
StatefulSet

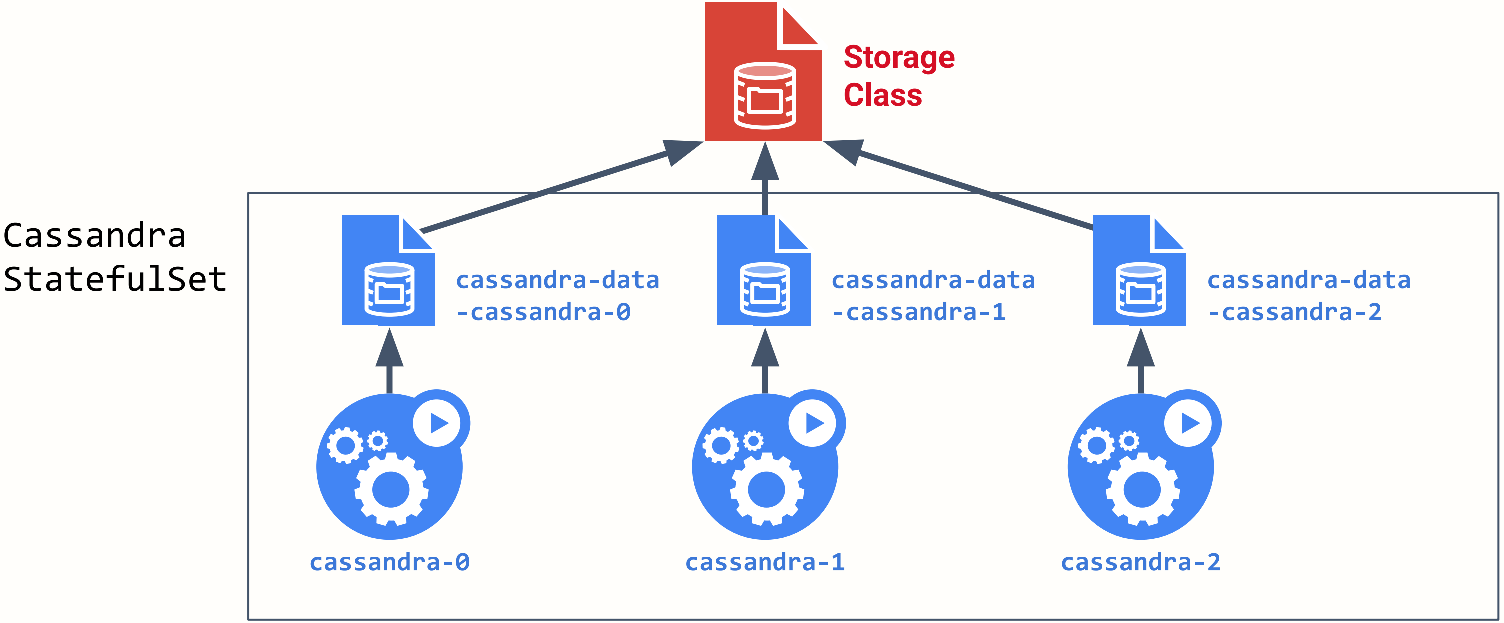
Claim template

Pod
template

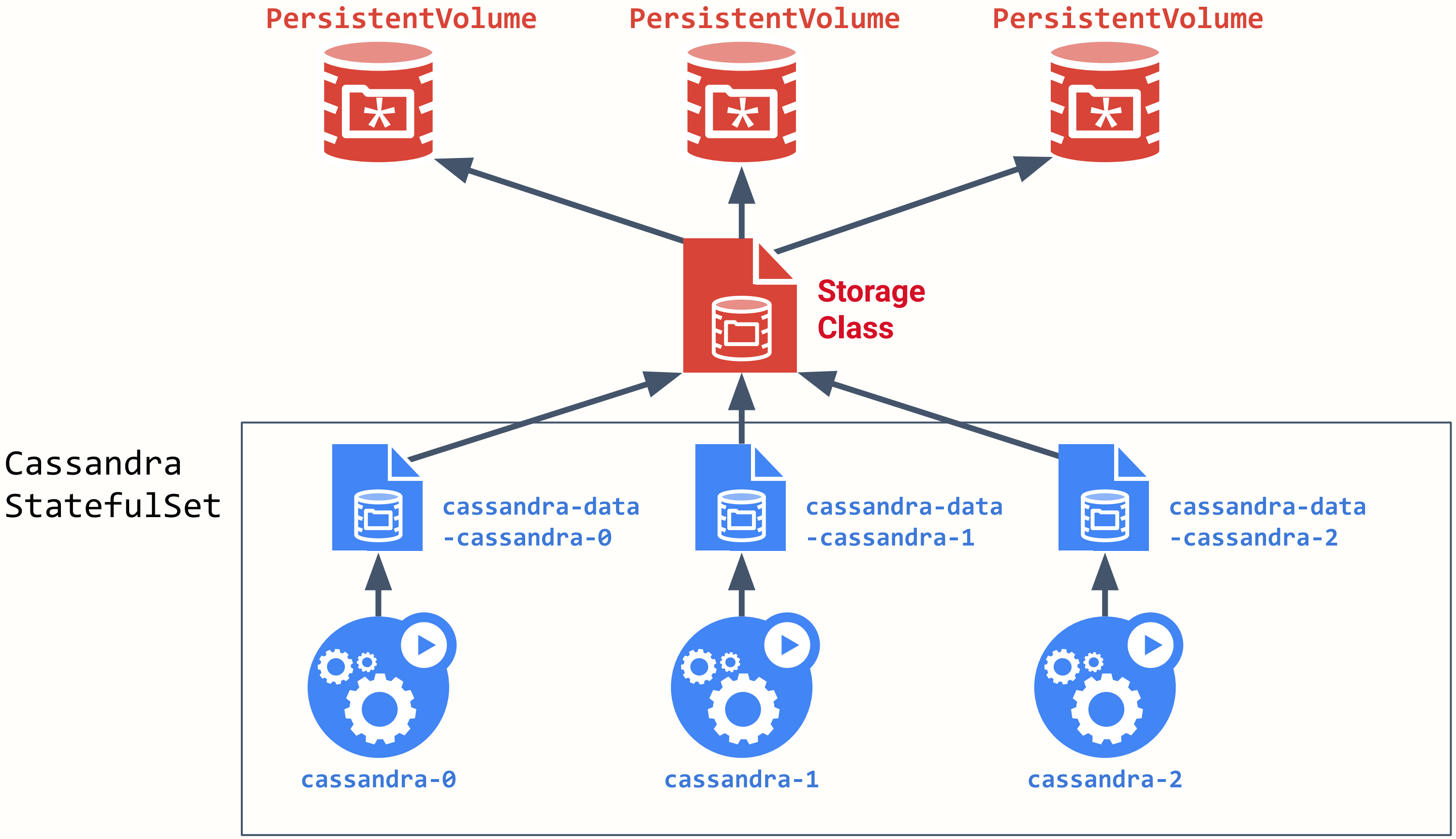
Pods & PVCs are created


Cassandra StatefulSet
cassandra-data -cassandra-0
cassandra-data -cassandra-1
cassandra-data -cassandra-2
cassandra-0
cassandra-1
cassandra-2

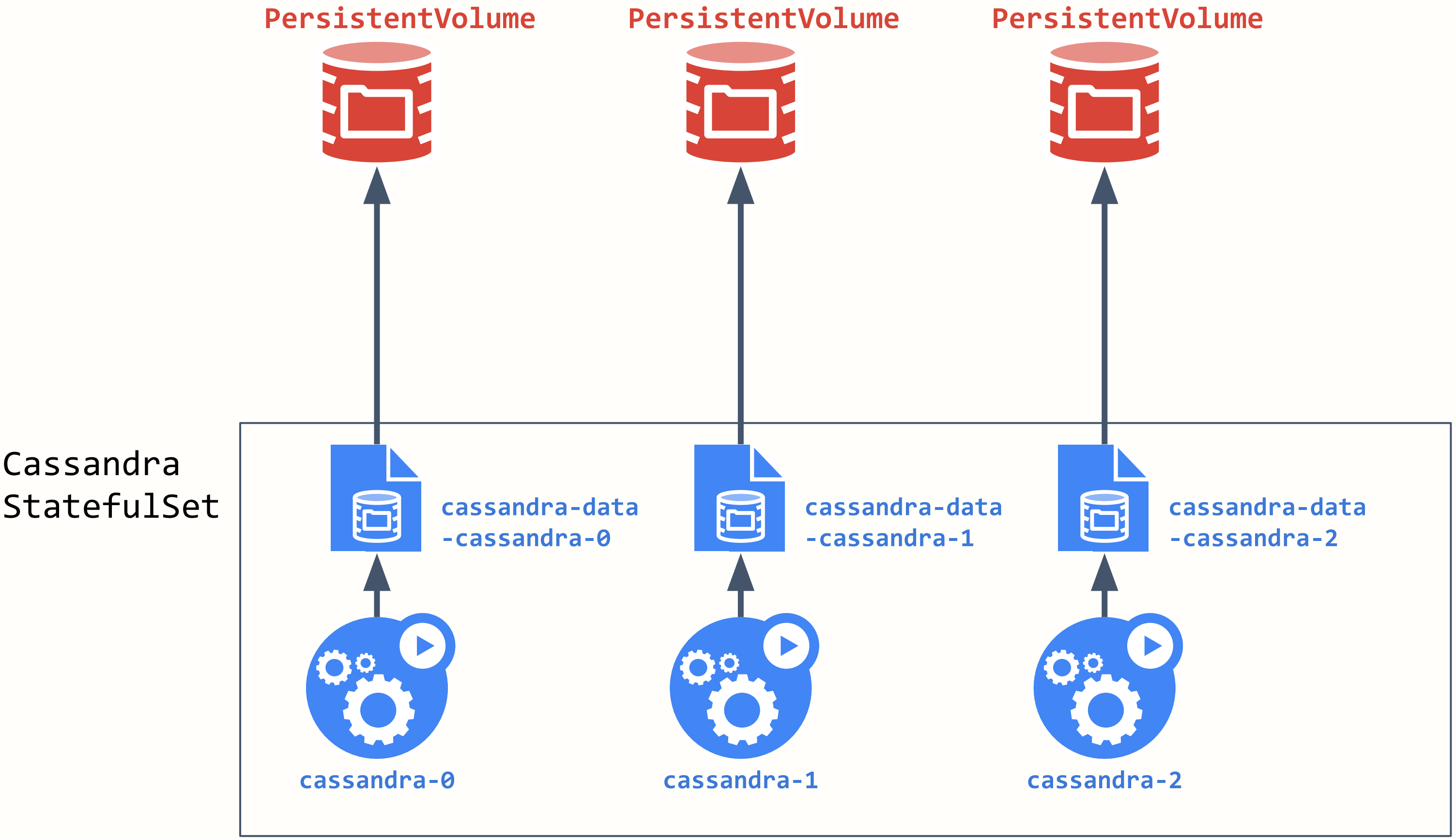
Volumes are Provisioned


Storage Class
Cassandra StatefulSet
cassandra-data -cassandra-0
cassandra-data -cassandra-1
cassandra-data -cassandra-2
cassandra-0
cassandra-1
cassandra-2

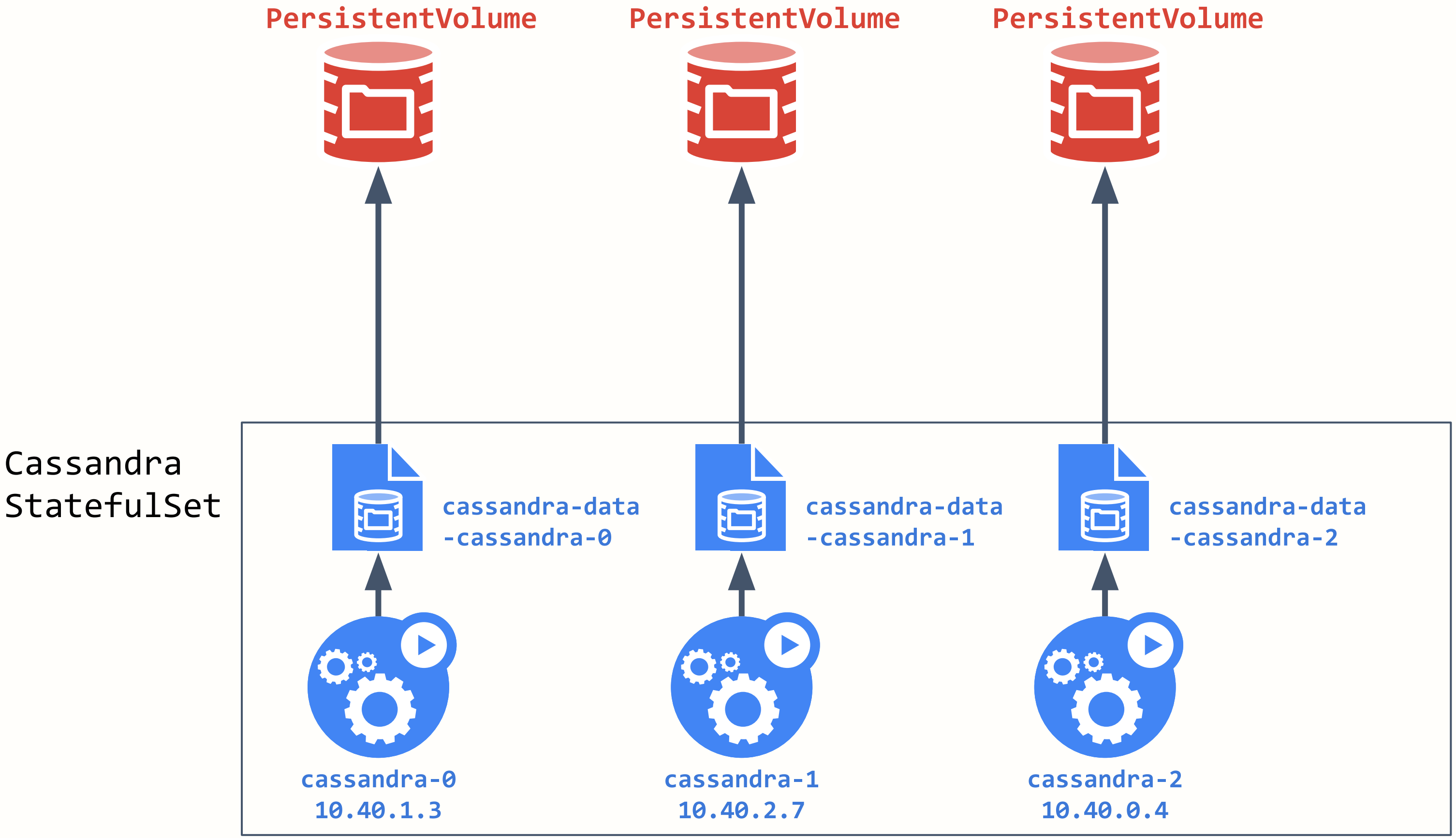
Volumes are Provisioned



Volumes are Attached/Mounted



Pods are started



PersistentVolume

PersistentVolume

PersistentVolume

Cassandra
StatefulSet

cassandra-data
-cassandra-0

cassandra-data
-cassandra-1

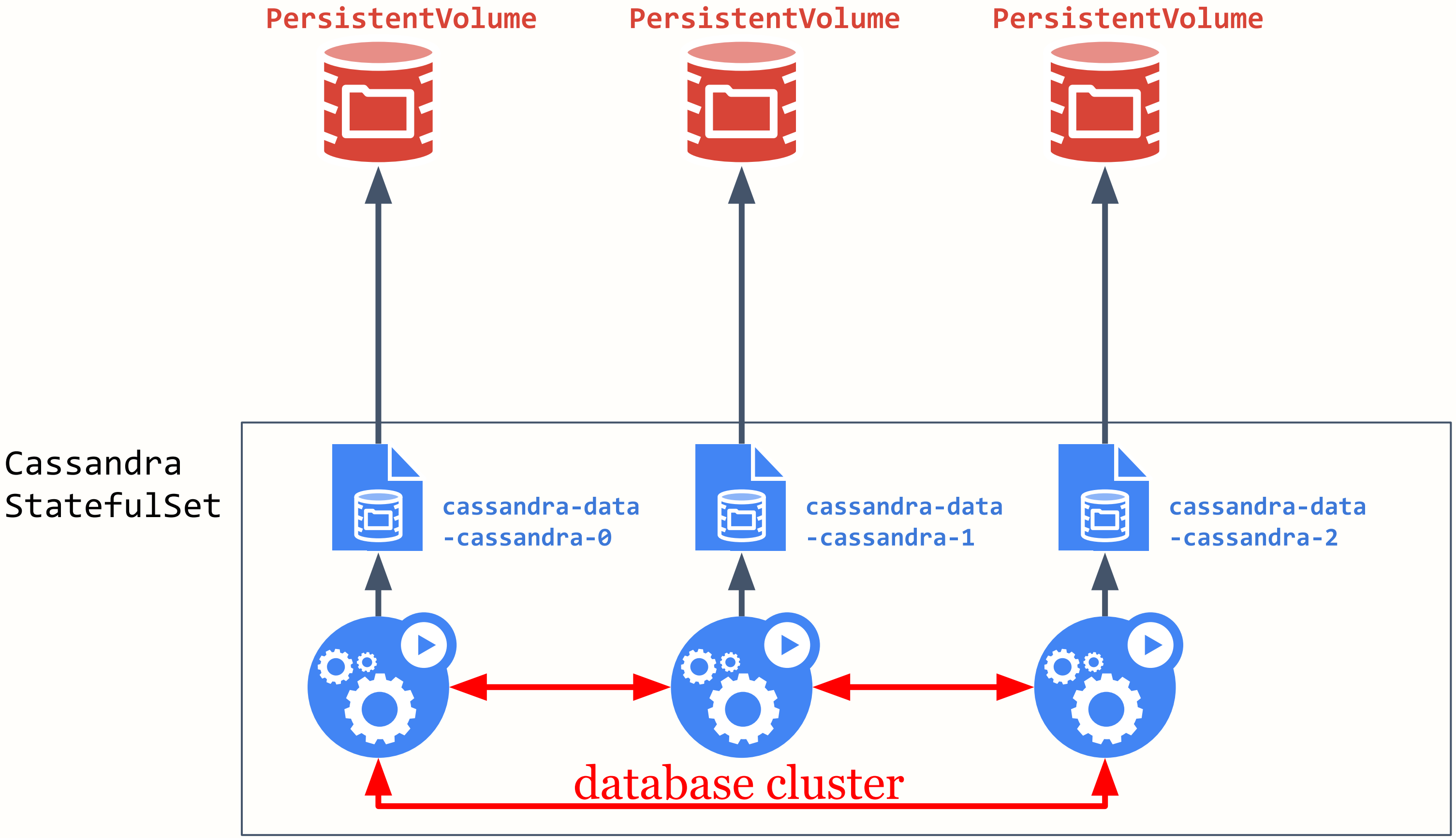
cassandra-data
-cassandra-2

cassandra-0
10.40.1.3

cassandra-1
10.40.2.7

cassandra-2
10.40.0.4

Pods *should* form a cluster



That was Cassandra,
what about the others?

MySQL

https://kubernetes.io/docs/tasks/run-application/run-replicated-stateful-application/

- Single read/write master.
- Multiple read-only slaves.
- -> Single point of failure.

PostgreSQL

https://github.com/CrunchyData/crunchy-containers

- Not tested by us.
- Optional operator: https://github.com/CrunchyData/postgres-operator
- Looks solid!

Mongo

https://codelabs.developers.google.com/codelabs/cloud-mongodb-statefulset/

- Not tested by us.
- Needs sidecar:

```
 - name: mongo-sidecar
   image: cvallance/mongo-k8s-sidecar
   env:
   - name: MONGO_SIDECAR_POD_LABELS
     value: "role=mongo,environment=test"
```

- Requires all replicas in Mongo connection URI.
  “mongodb://mongo-0.mongo,mongo-1.mongo,mongo-2.mongo:27017/dbname_?"

Cloud-native databases

- CockroachDB
- FoundationDB
- TiDB
- Vitess
- YugaDB
- …

Advanced Topics

Helm chart

- Template of YAML files.
- Simplify deployment of Kubernetes applications.

Operator

- Small application running in the cluster.
- Simplify deployment and maintenance of Kubernetes applications.

Advanced Topics

Updates

- How to update to a new version?
- Will StatefulSet rolling update strategy work for my DB?
- Or should I update manually? How?

Advanced Topics

Networking

- IP addresses of pods can change.
    - Only DNS name is stable.
    - Should applications always resolve address of a service for each request?
- Network partition.

Advanced Topics

Backup

- How do I backup my data?
- Does the app / database support consistent dump?
- Can I use snapshots?
- How do I recover from data loss?

Advanced Topics

Availability

- What happens if one pod dies?
- What happens if one node dies?
- What happens if whole datacenter dies?
- Anti-affinity

Advanced Topics

Security

- What pods / machines can talk to the database?
- What other pods can run on the same machine as the database pods?
  - Are they trustworthy?
  - What happens when one of them escapes its container?

There is no silver bullet. Compare options and make informed tradeoffs.

Questions?

Reach Out

David Zhu

- Email: dyzz@google.com
- Github: davidz627
- Twitter: dyzzhu

Jan Šafránek

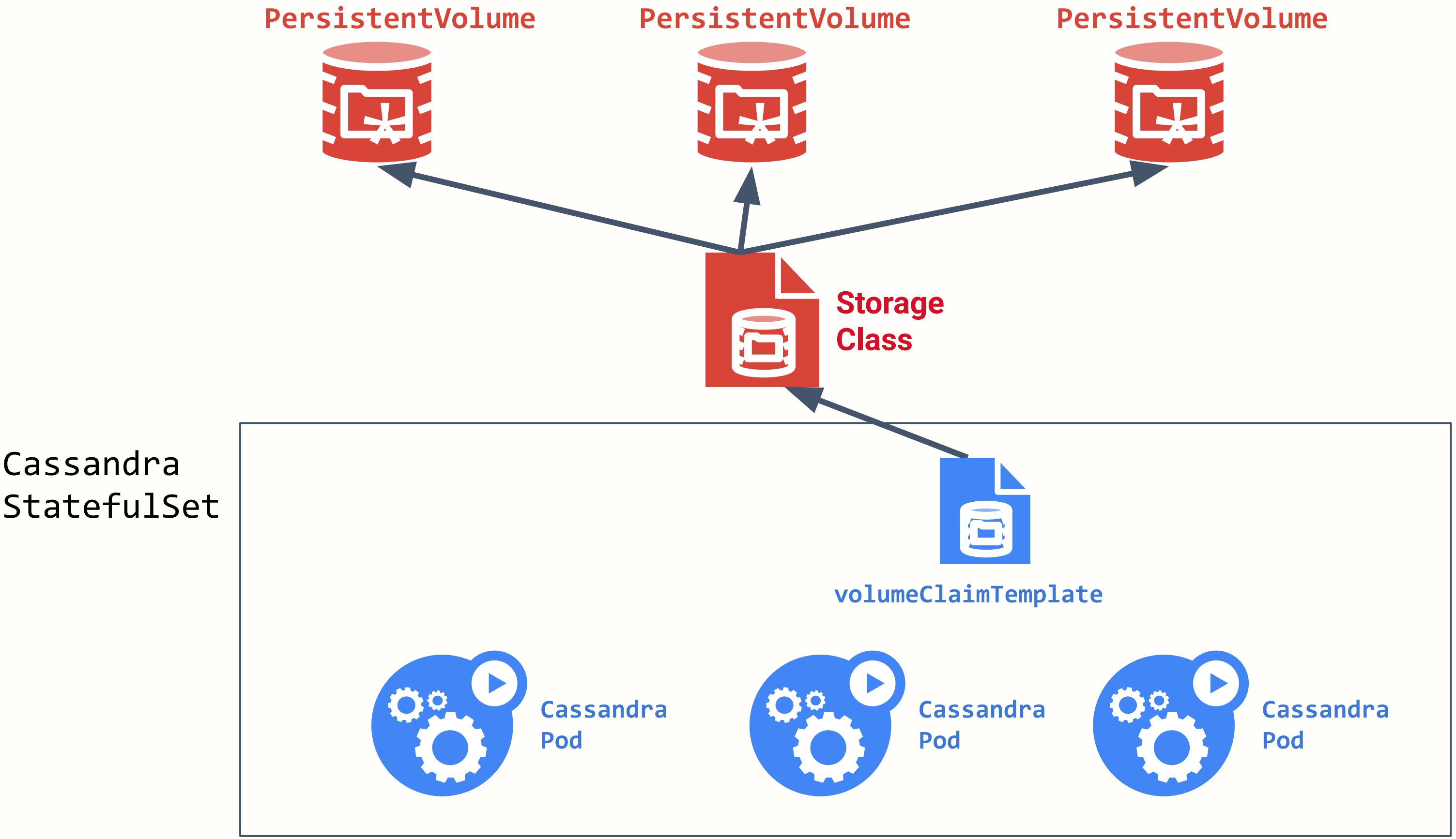
- Email: jsafrane@redhat.com
- Github: jsafrane

Sig-Storage

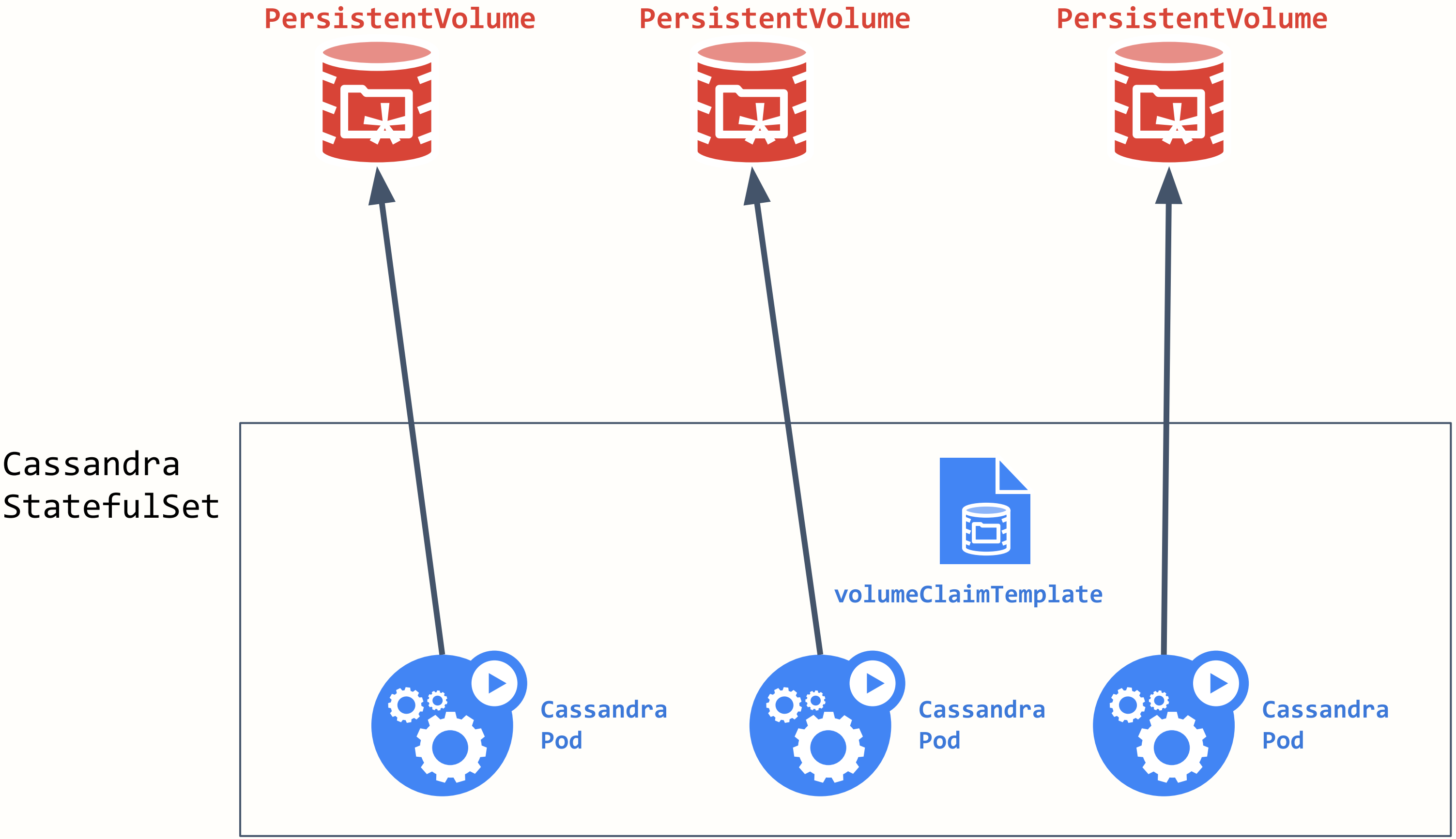
- Mailing List
  - kubernetes-sig-storage@googlegroups.com
- Bi-Weekly Meetings
  - 4:00pm-5:00pm GMT Every Second Thursday (next 23 May 2019)
  - https://zoom.us/j/614261834
- Slack
  - kubernetes.slack.com
    - #sig-storage

Junkyard

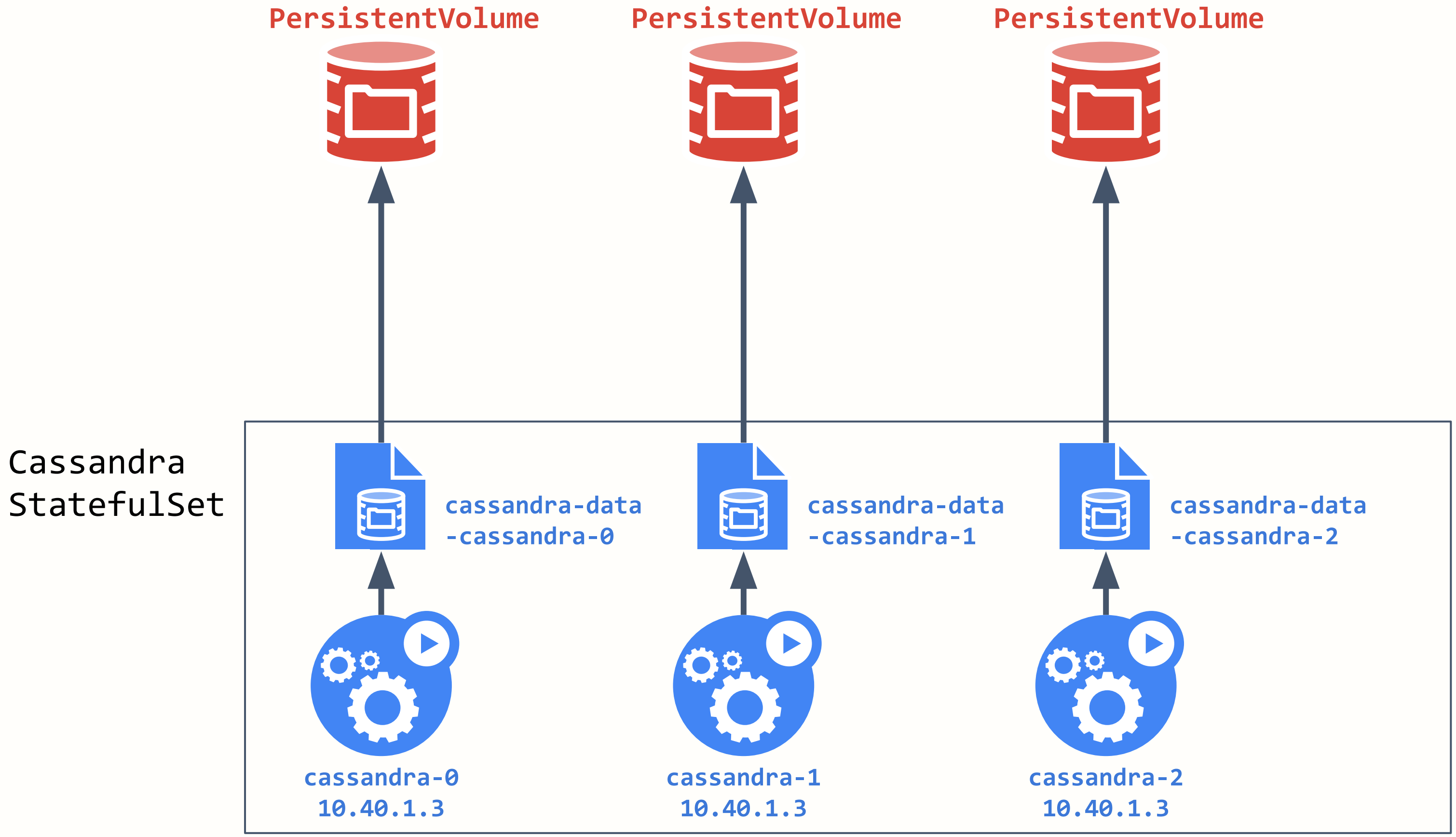
Volumes are Provisioned



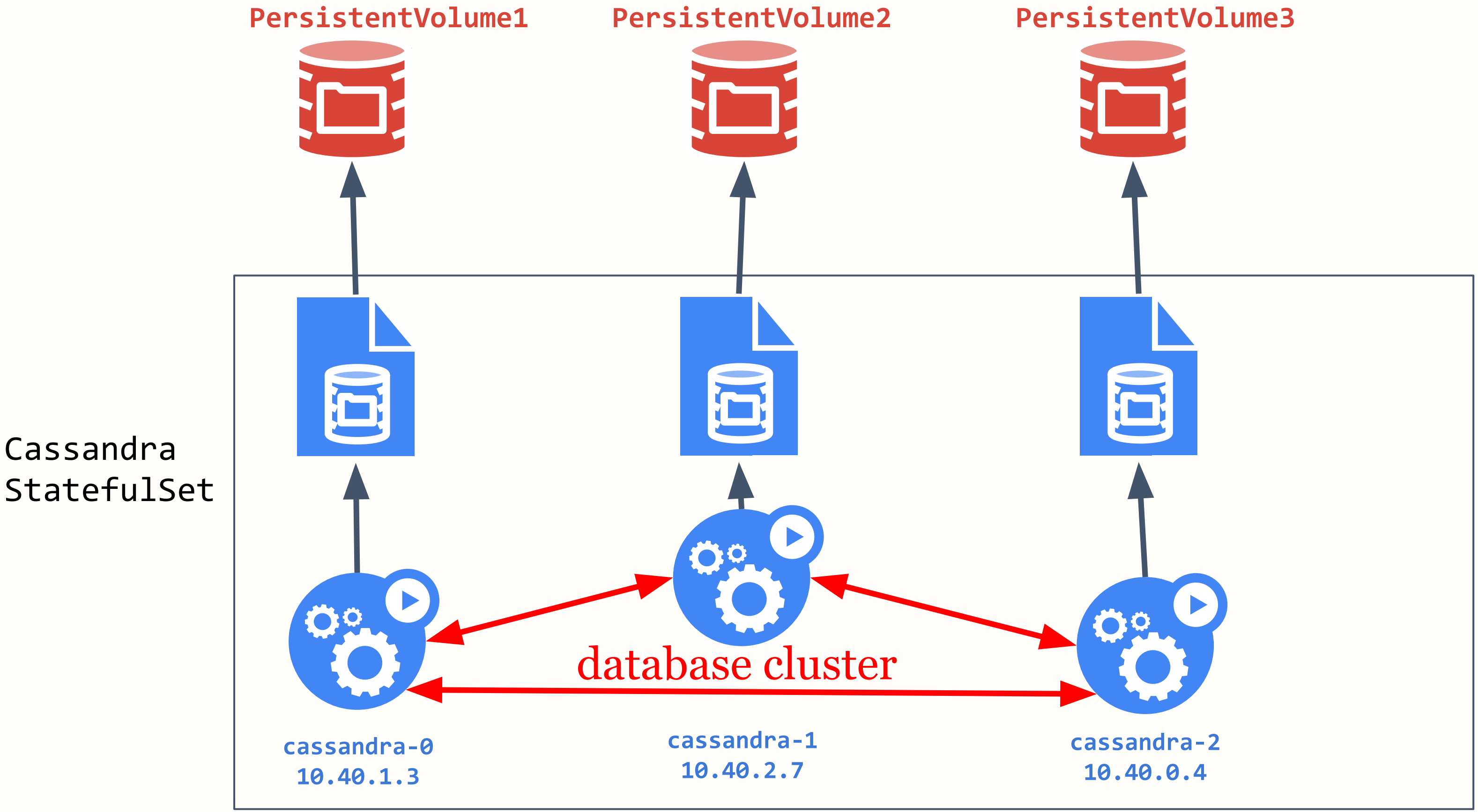
Volumes are Attached/Mounted



Volumes are Attached/Mounted



StatefulSet


PersistentVolume1
PersistentVolume2
PersistentVolume3
Cassandra
StatefulSet
database cluster
cassandra-0
10.40.1.3
cassandra-1
10.40.2.7
cassandra-2
10.40.0.4