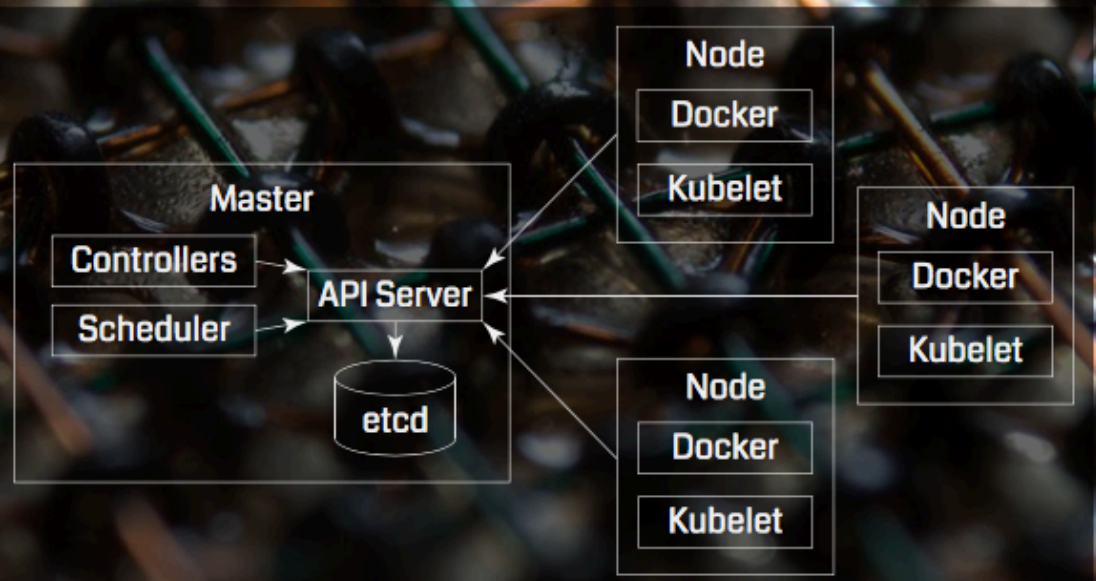
Kubernetes

tuna

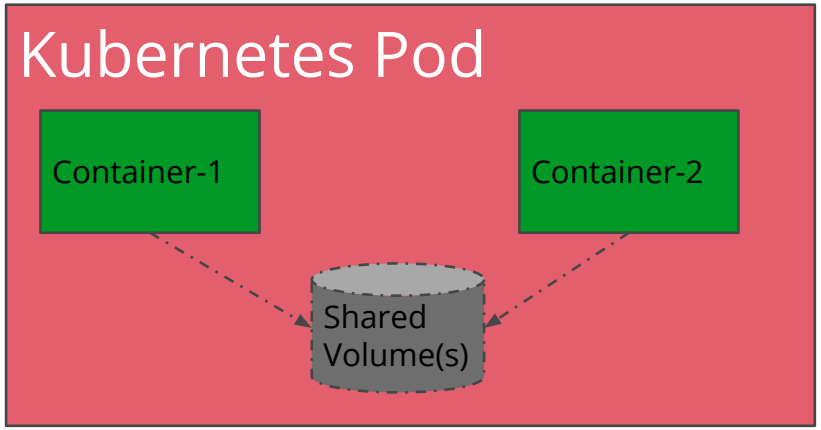

Kubernetes
Open Source
https://github.com/kubernetes/kubernetes
Very active open source project
12k stars, 750+ contributors
Apache 2 licensed
Written in Go
In progress: donate to the Cloud Native Computing Foundation

Core Concepts

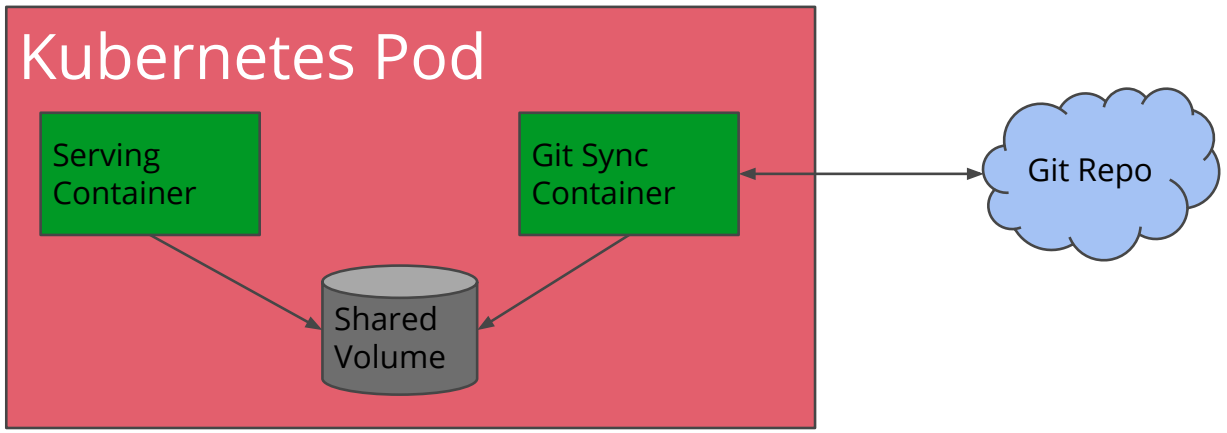
Cluster



Components: Pods

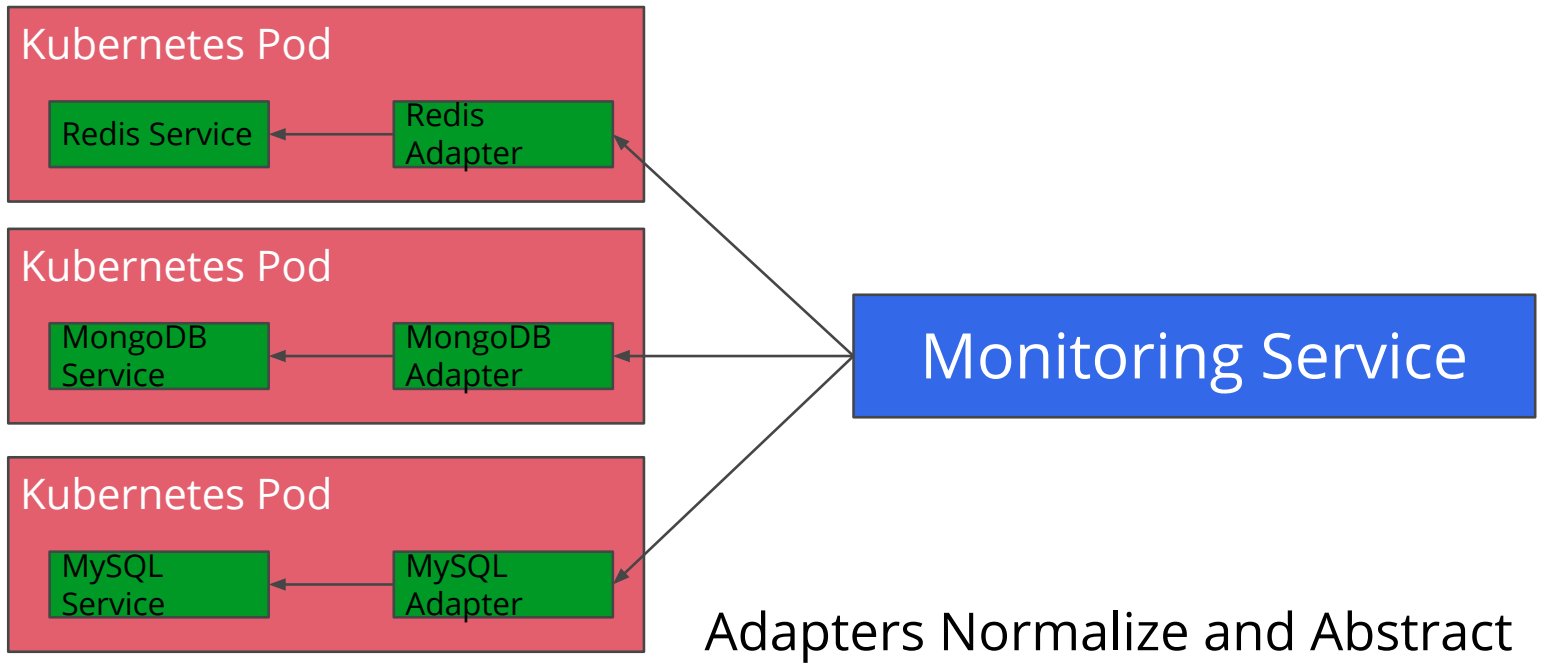


Components: Sidecars



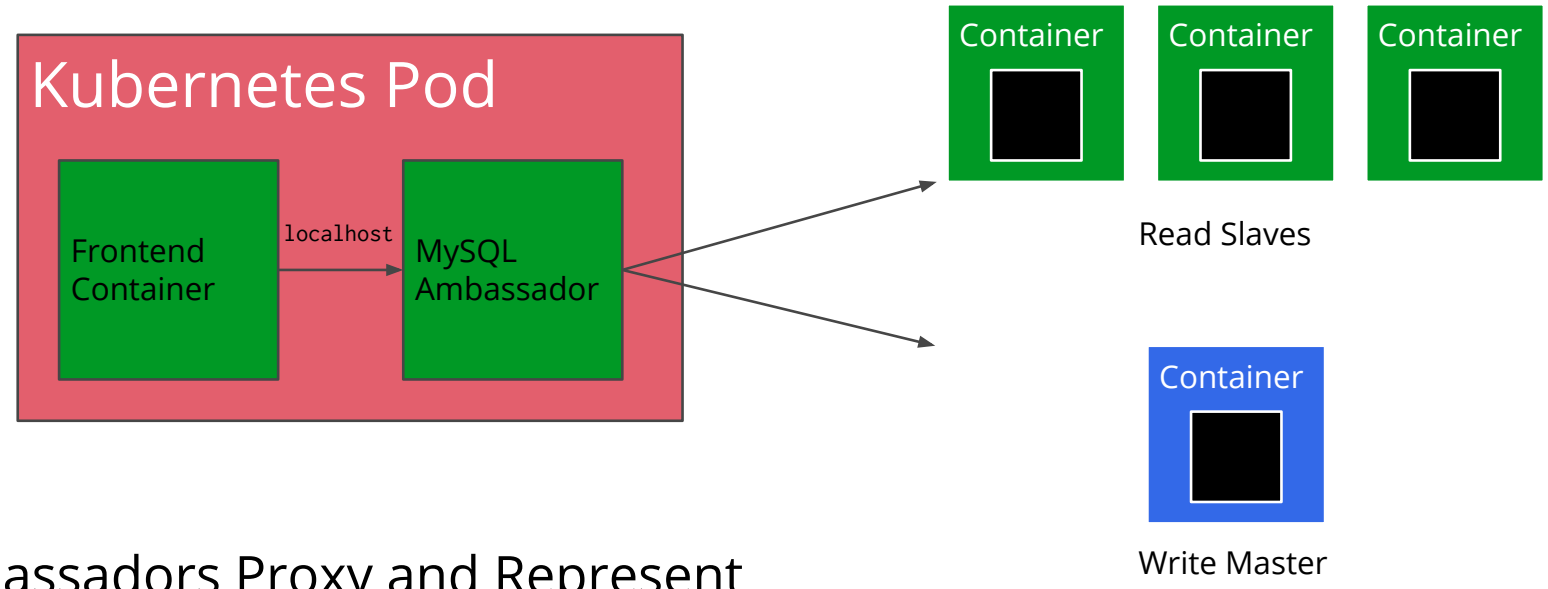
Sidecars Extend and Enhance

Components: Adapters



Adapters Normalize and Abstract

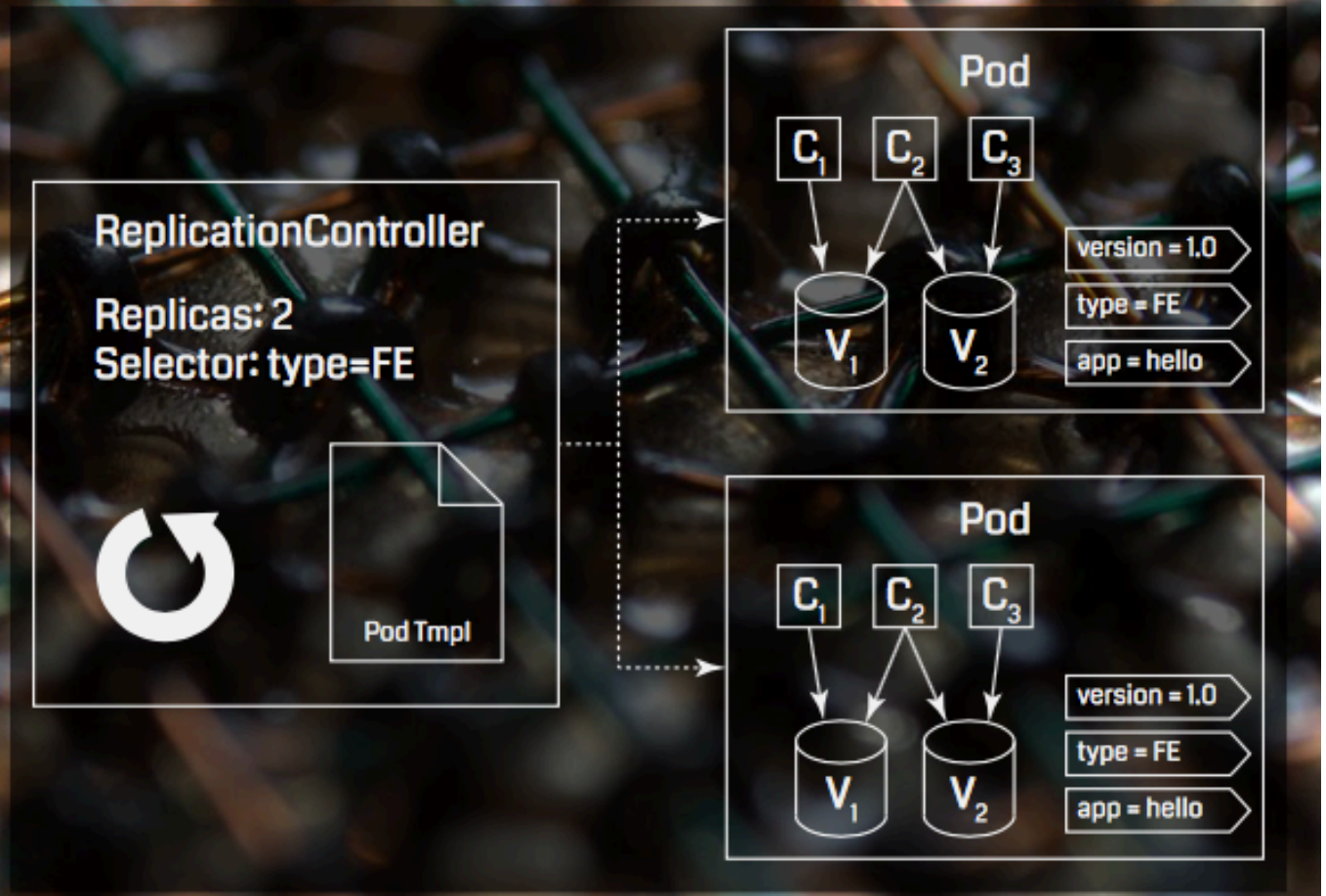
Components: Ambassadors



Ambassadors Proxy and Represent

Core Concepts

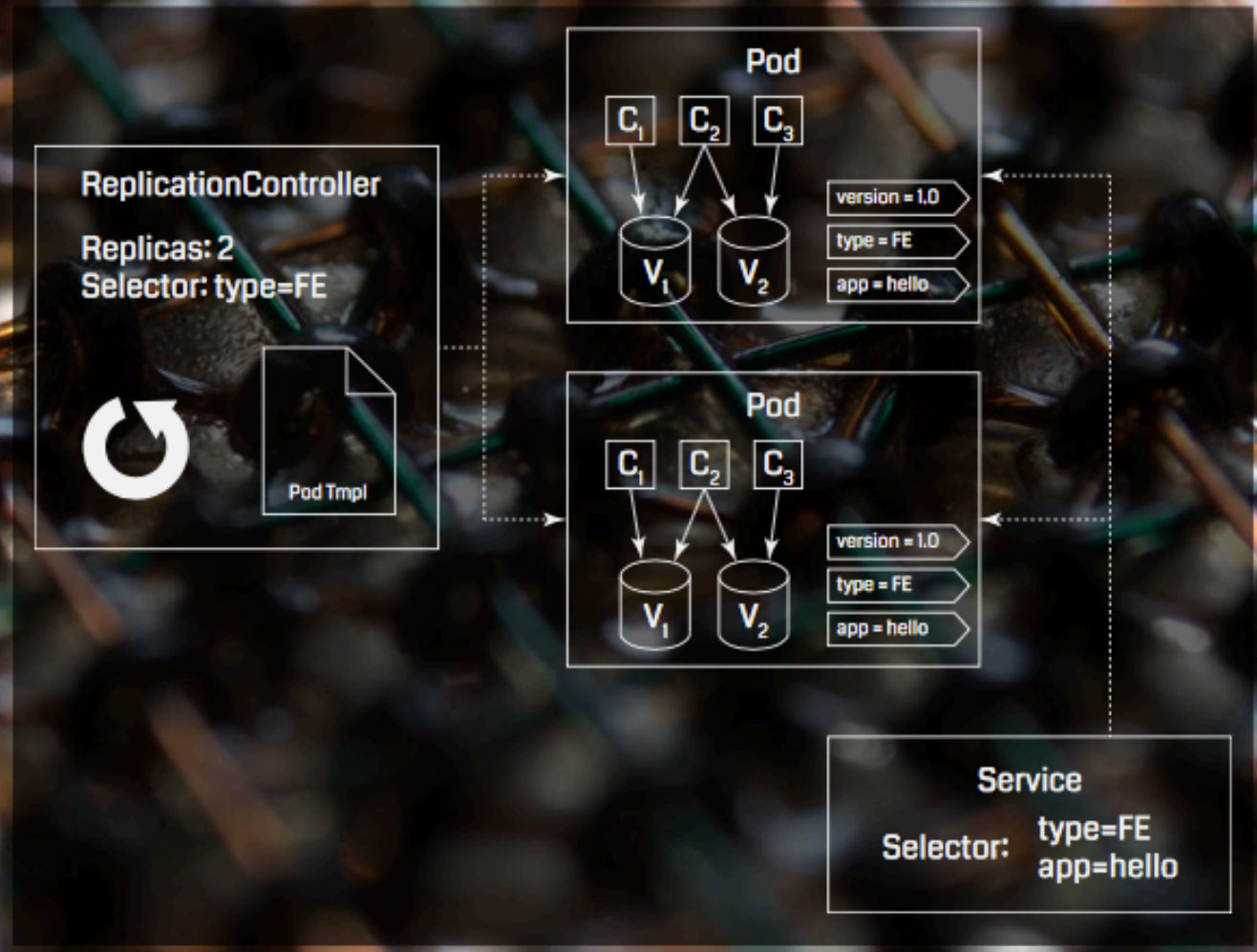
Replication Controller



```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

Core Concepts

Service



Defining a service

Suppose you have a set of `pods` that each expose port `9376` and carry a label "app=MyApp".

```
{
    "kind": "Service",
    "apiVersion": "v1",
    "metadata": {
        "name": "my-service"
    },
    "spec": {
        "selector": {
            "app": "MyApp"
        },
        "ports": [
            {
                "protocol": "TCP",
                "port": 80,
                "targetPort": 9376
            }
        ]
    }
}
```

In any of these scenarios you can define a `service` without a `selector`:

```
{
    "kind": "Service",
    "apiVersion": "v1",
    "metadata": {
        "name": "my-service"
    },
    "spec": {
        "ports": [
            {
                "protocol": "TCP",
                "port": 80,
                "targetPort": 9376
            }
        ]
    }
}
```

```
{
    "kind": "Endpoints",
    "apiVersion": "v1",
    "metadata": {
        "name": "my-service"
    },
    "subsets": [
        {
            "addresses": [
                { "ip": "1.2.3.4" }
            ],
            "ports": [
                { "port": 9376 }
            ]
        }
    ]
}
```

Discovering services

Environment variables

```
REDIS_MASTER_SERVICE_HOST=10.0.0.11
REDIS_MASTER_SERVICE_PORT=6379
REDIS_MASTER_PORT=tcp://10.0.0.11:6379
REDIS_MASTER_PORT_6379_TCP=tcp://10.0.0.11:6379
REDIS_MASTER_PORT_6379_TCP_PROTO=tcp
REDIS_MASTER_PORT_6379_TCP_PORT=6379
REDIS_MASTER_PORT_6379_TCP_ADDR=10.0.0.11
```

DNS

my-service.my-namespace

Publishing services - service types

Service could be exposed onto an external or internal network. Kubernetes `ServiceTypes` allow you to specify what kind of service you want.

Valid values for the **ServiceType** field are:

- **ClusterIP**: use a cluster-internal IP only - this is the default and is discussed above. Choosing this value means that you want this service to be reachable only from inside of the cluster.
- **NodePort**: on top of having a cluster-internal IP, expose the service on a port on each node of the cluster (the same port on each node). You'll be able to contact the service on any **NodeIP:NodePort** address.
- **LoadBalancer**: on top of having a cluster-internal IP and exposing service on a NodePort also, ask the cloud provider for a load balancer which forwards to the `Service` exposed as a **NodeIP:NodePort** for each Node.

Type LoadBalancer

```
{
    "kind": "Service",
    "apiVersion": "v1",
    "metadata": {
        "name": "my-service"
    },
    "spec": {
        "selector": {
            "app": "MyApp"
        },
        "ports": [
            {
                "protocol": "TCP",
                "port": 80,
                "targetPort": 9376,
                "nodePort": 30061
            }
        ],
        "clusterIP": "10.0.171.239",
        "loadBalancerIP": "78.11.24.19",
        "type": "LoadBalancer"
    },
    "status": {
        "loadBalancer": {
            "ingress": [
                {
                    "ip": "146.148.47.155"
                }
            ]
        }
    }
}
```

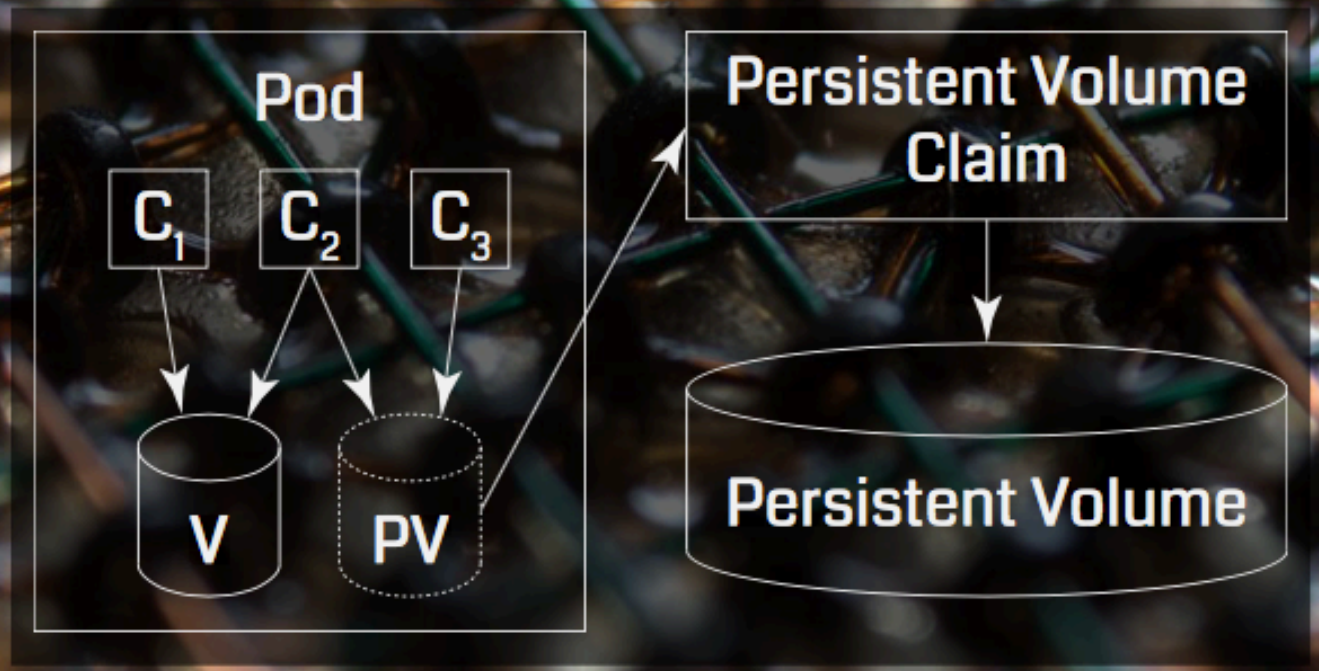
External IPs

If there are external IPs that route to one or more cluster nodes, Kubernetes services can be exposed on those `externalIPs`. `externalIPs` are not managed by Kubernetes and are the responsibility of the cluster administrator. In the example below, my-service can be accessed by clients on 80.11.12.10:80 (externalIP:port)

```
{
    "kind": "Service",
    "apiVersion": "v1",
    "metadata": {
        "name": "my-service"
    },
    "spec": {
        "selector": {
            "app": "MyApp"
        },
        "ports": [
            {
                "name": "http",
                "protocol": "TCP",
                "port": 80,
                "targetPort": 9376
            }
        ],
        "externalIPs" : [
            "80.11.12.10"
        ]
    }
}
```

Core Concepts

Persistent Volumes



Types of Persistent Volumes

`PersistentVolume` types are implemented as plugins. Kubernetes currently supports the following plugins:

- GCEPersistentDisk
- AWSElasticBlockStore
- NFS
- iSCSI
- RBD (Ceph Block Device)
- Glusterfs
- HostPath (single node testing only – local storage is not supported in any way and WILL NOT WORK in a multi-node cluster)

awsElasticBlockStore

An `awsElasticBlockStore` volume mounts an Amazon Web Services (AWS) EBS Volume into your pod. Unlike `emptyDir`, which is erased when a Pod is removed, the contents of an EBS volume are preserved and the volume is merely unmounted. This means that an EBS volume can be pre-populated with data, and that data can be "handed off" between pods.

Important: You must create an EBS volume using `aws ec2 create-volume` or the AWS API before you can use it

There are some restrictions when using an awsElasticBlockStore volume:

- the nodes on which pods are running must be AWS EC2 instances
- those instances need to be in the same region and availability-zone as the EBS volume
- EBS only supports a single EC2 instance mounting a volume

AWS EBS Example configuration

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-ebs
spec:
  containers:
  - image: gcr.io/google_containers/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-ebs
      name: test-volume
  volumes:
  - name: test-volume
    # This AWS EBS volume must already exist.
    awsElasticBlockStore:
      volumeID: <volume-id>
      fsType: ext4
```