Deep Dive into Kubernetes Part 1

Imesh Gunaratne, WSO2

Agenda

- Kubernetes Architecture
- Container Orchestration:
  - Pods
  - Replica Sets
  - Deployments
- Internal Routing
  - Services
- External Routing
  - Ingresses & Ingress Controllers

Agenda Cont.

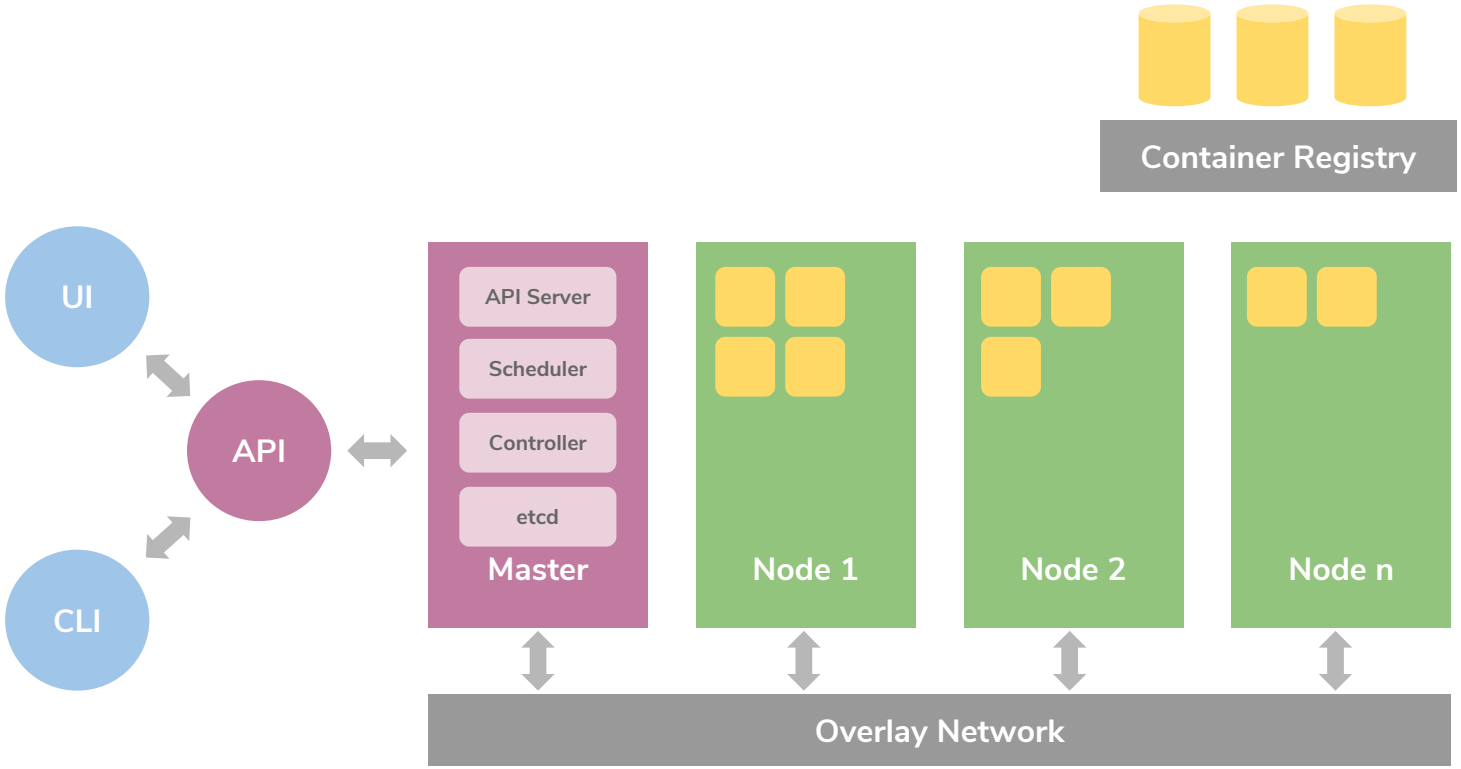
- Configuration Management
  - Config Maps
- Credentials Management
  - Secrets
- Persistent Volumes
- Rolling Out Updates
- Autoscaling
  - Horizontal Pod Autoscalers

Agenda Cont.

- Package Management
  - Helm
- Hello World Example

Kubernetes Architecture

Kubernetes Architecture



Container Orchestration

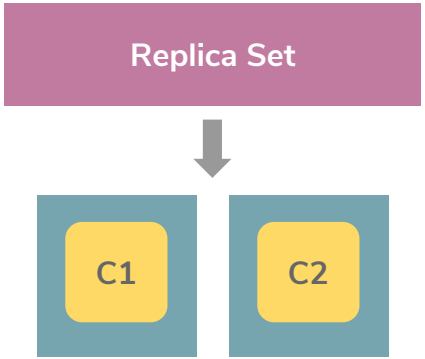
Pods

- A pod is a group of containers that share the file system, users, network interfaces, etc
- By default a pod will include two containers: one for the given docker image and other for the network interface


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo
Hello Kubernetes! && sleep 3600']
```

Replica Sets

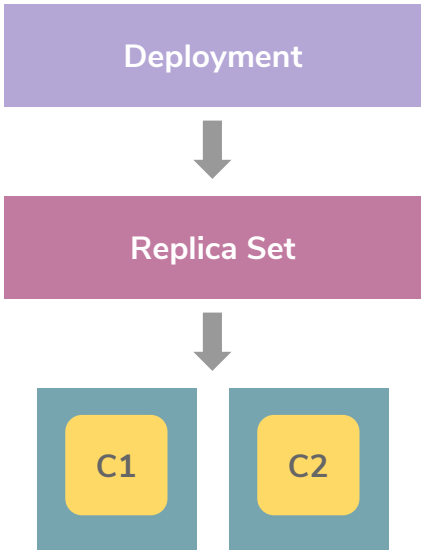
- Replica Sets are used for orchestrating pods
- They define the docker images, resources, env. variables, ports, etc required for creating pods




```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
    matchExpressions:
      - {key: tier, operator: In}
  template:
    metadata:
      labels:
        ...
    spec:
      containers:
      - name: php-redis
        image: foo:bar
        ports:
        - containerPort: 80
```

Deployments

- A deployment is used for orchestrating pods via replica sets:

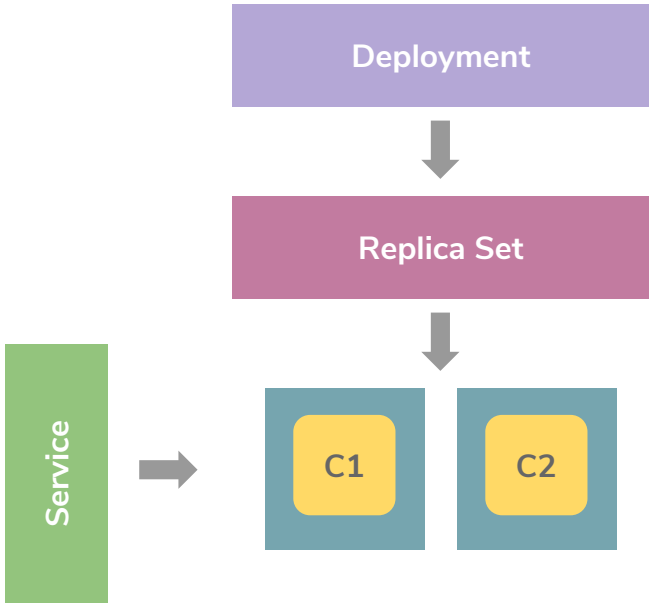



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Internal Routing

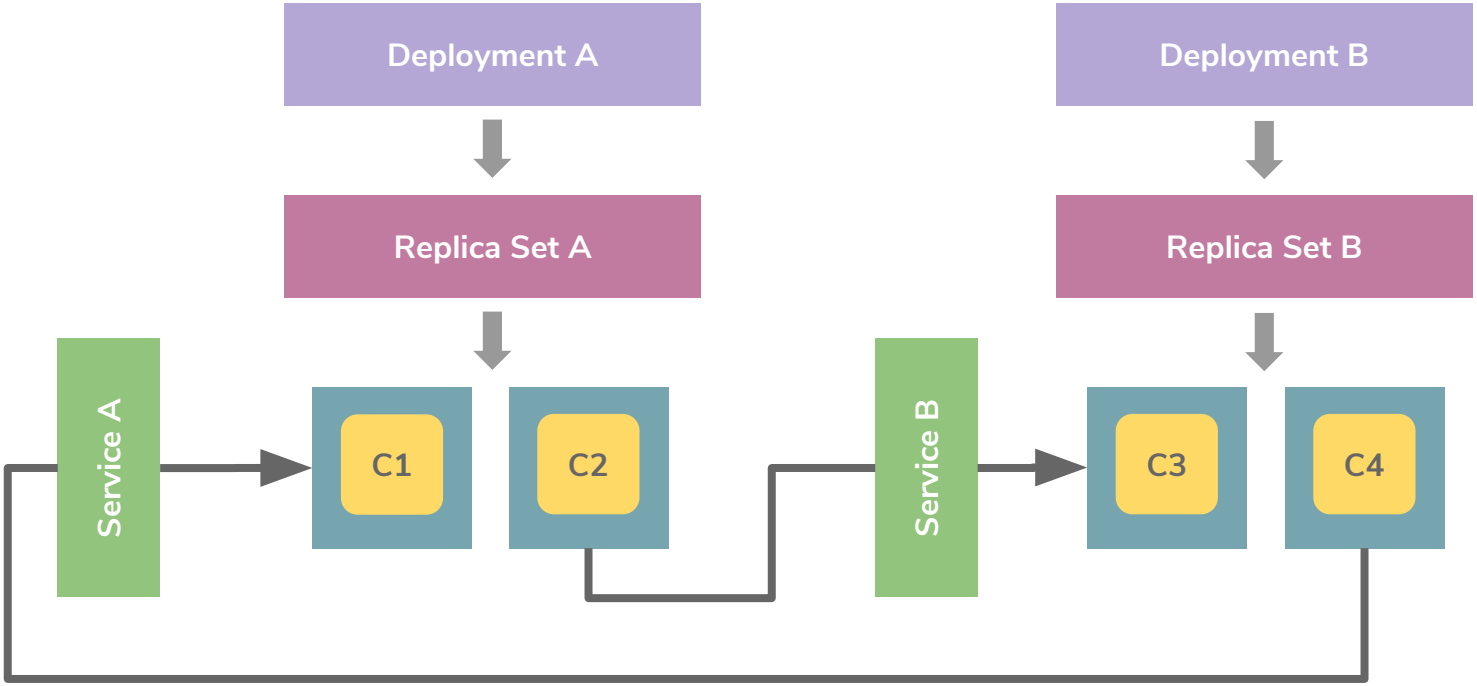
Services

- A service provides a layer 4 load balancer for pods:




```yaml
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

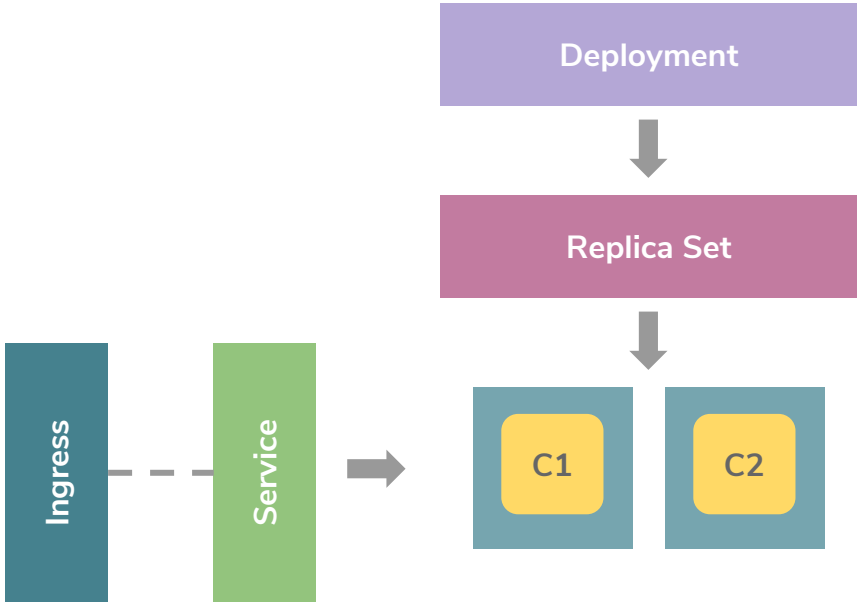
Pod to Pod Communication



External Routing

Ingresses

- An ingress is used for configuring a load balancer for external routing

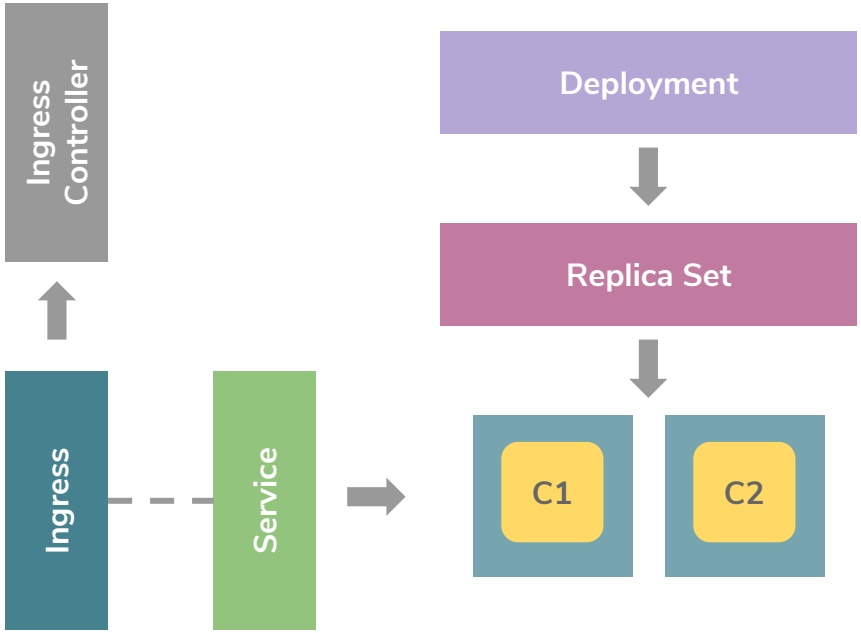



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:

nginx.ingress.kubernetes.io/rewri
te-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```

Ingresses

- An ingress is used for configuring a load balancer for external routing

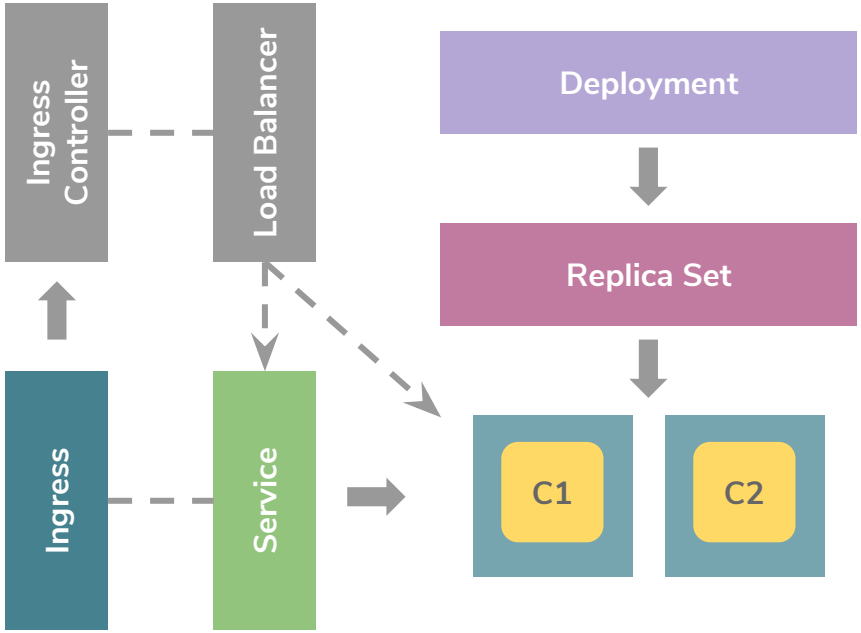



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:

nginx.ingress.kubernetes.io/rewri
te-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```

Ingresses

- An ingress is used for configuring a load balancer for external routing

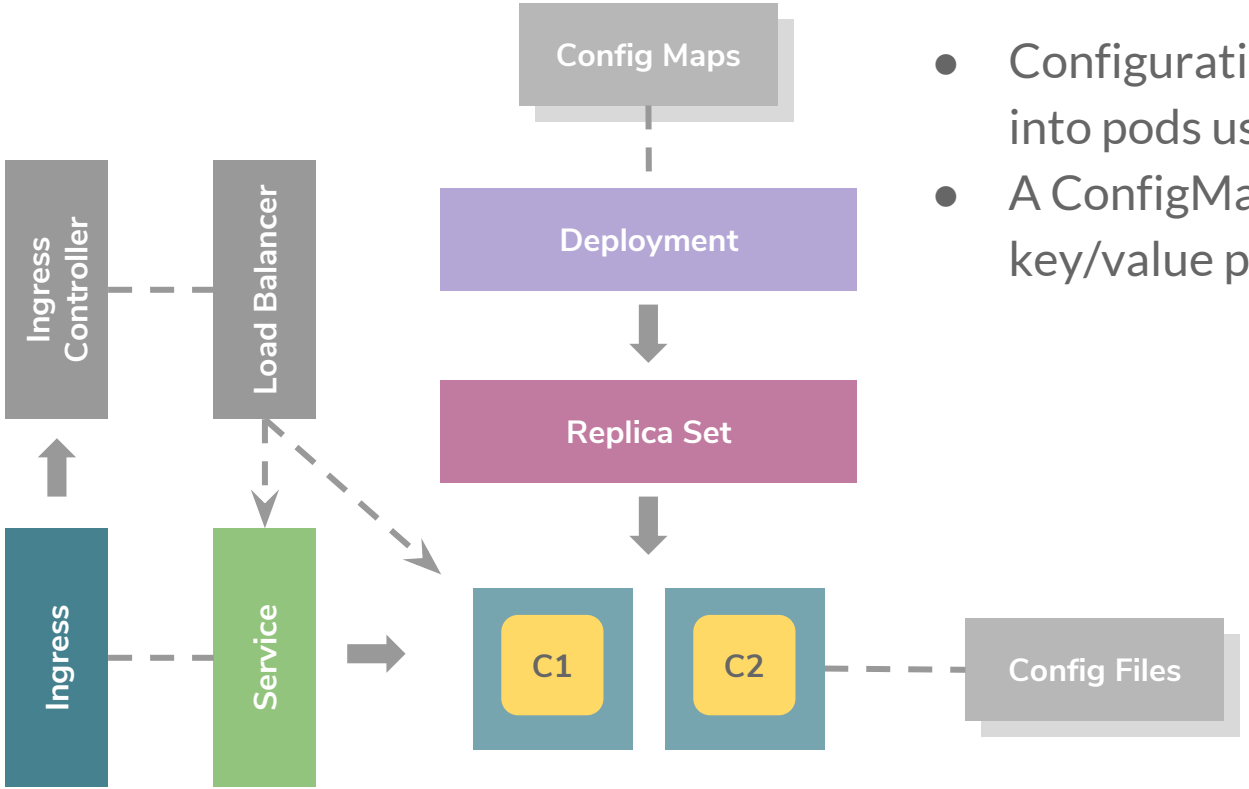



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:

nginx.ingress.kubernetes.io/rewri
te-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```

Configuration Management

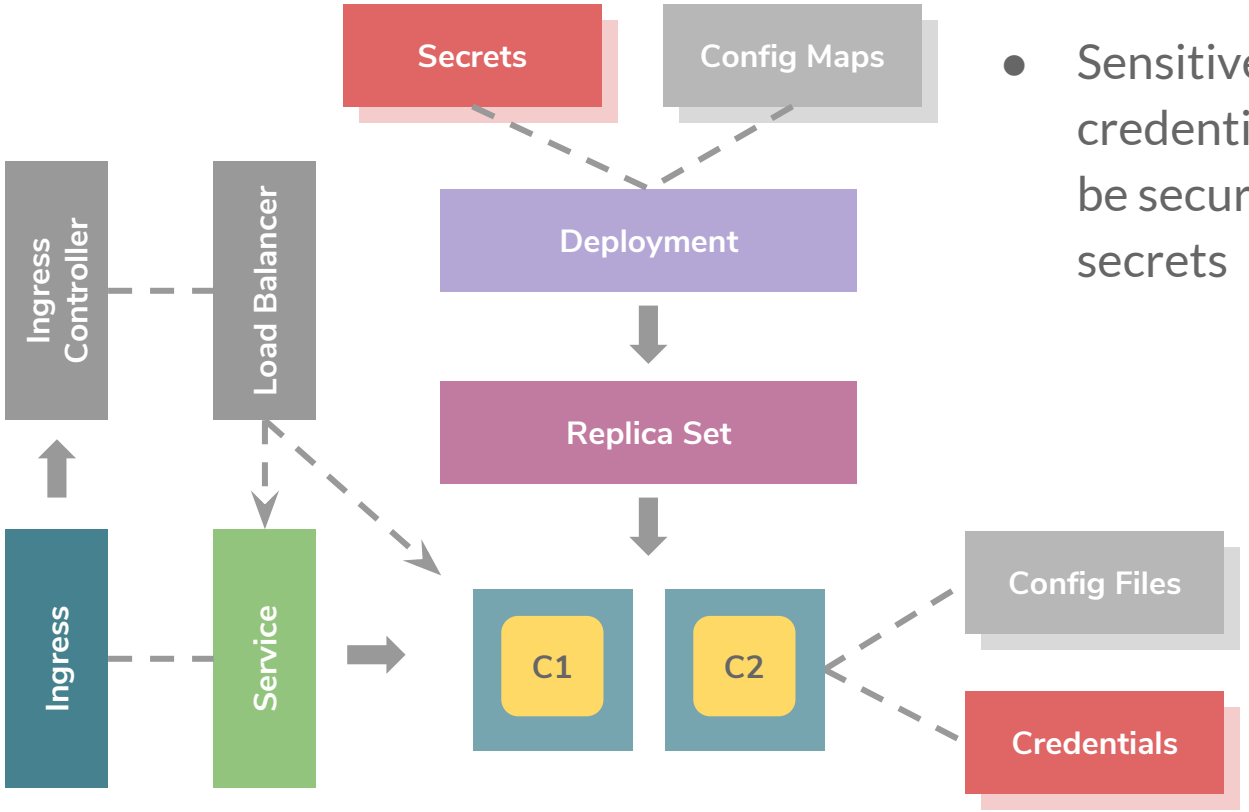
ConfigMaps



- Configuration files can be injected into pods using ConfigMaps
- A ConfigMap can be created for key/value pairs, files and folders

Credentials Management

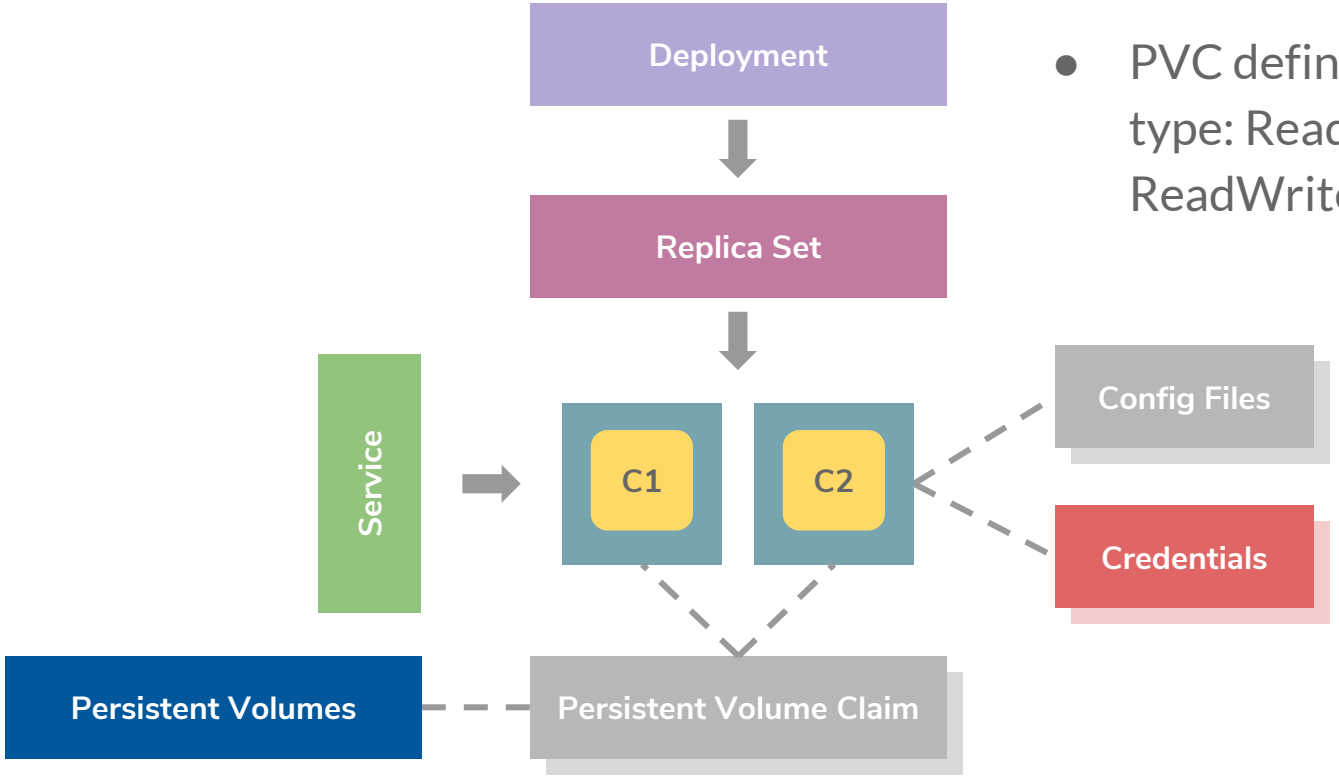
Secrets



- Sensitive data such as credentials, encryption keys can be securely injected using secrets

Persistent Volumes

Persistent Volumes



- PVC defines volume size, and type: ReadOnly, ReadWrite, ReadWriteMany

Persistent Volume Types

- GCEPersistentDisk
- AWSElasticBlockStore
- AzureFile
- AzureDisk
- FC (Fibre Channel)**
- FlexVolume
- Flocker
- NFS
- iSCSI
- RBD (Ceph Block Device)
- CephFS
- Cinder (OpenStack block storage)
- Glusterfs
- VsphereVolume
- Quobyte Volumes
- VMware Photon
- Portworx Volumes
- ScaleIO Volumes
- StorageOS

https://kubernetes.io/docs/concepts/storage/persistent-volumes/#types-of-persistent-volumes

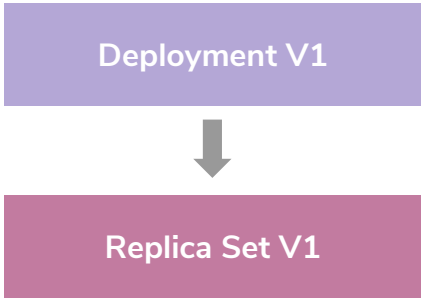
Rolling Out Updates

Deployment Process

Deployment V1

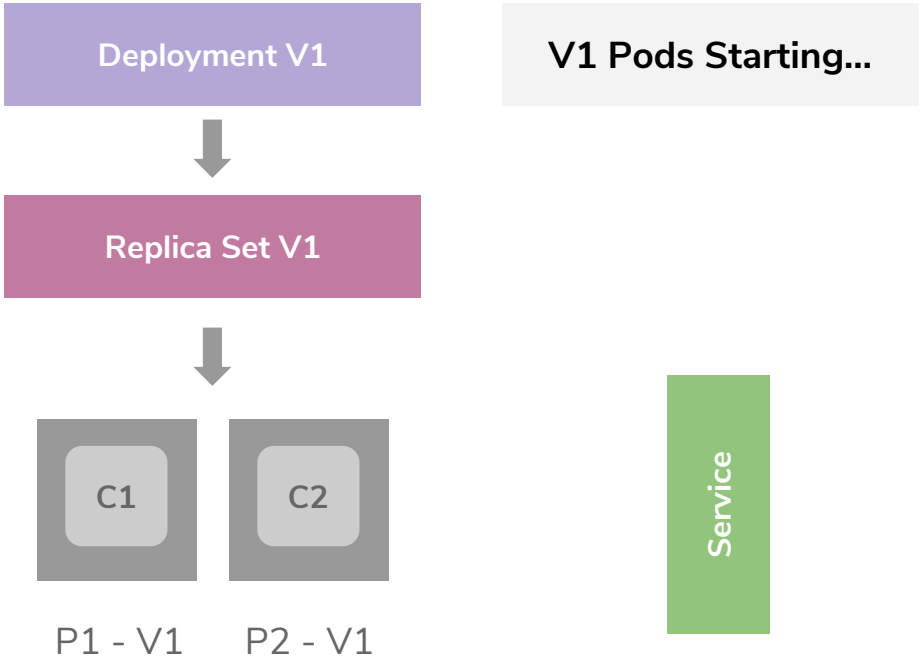
V1 Deploying...

Deployment Process



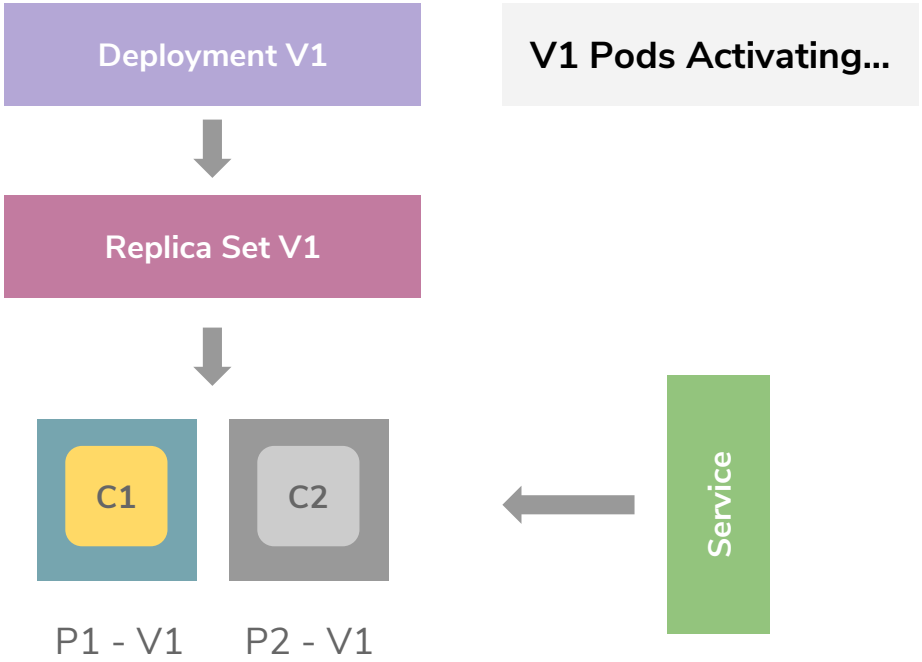
V1 Deploying...

Deployment Process

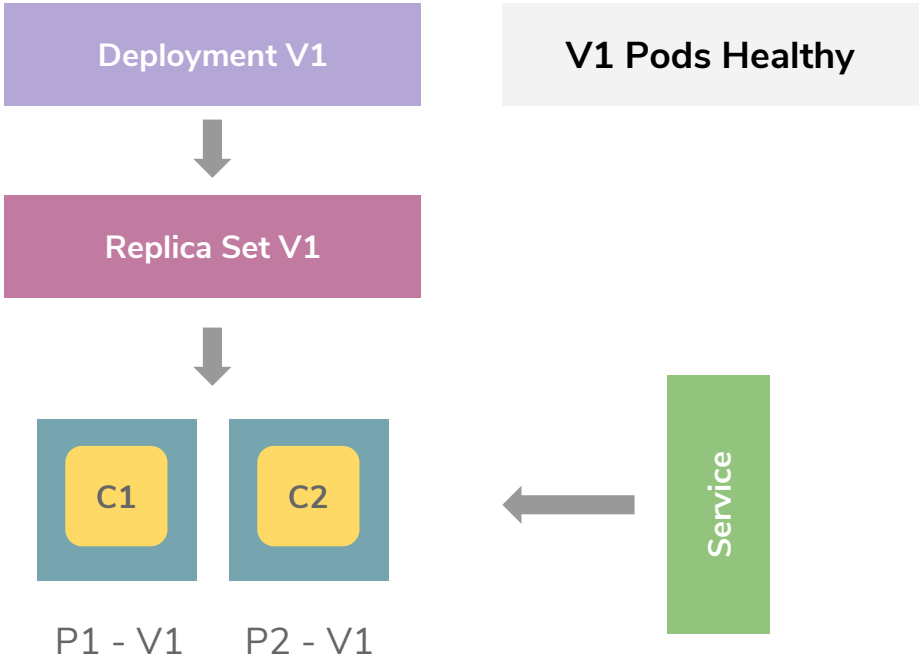


V1 Pods Starting...

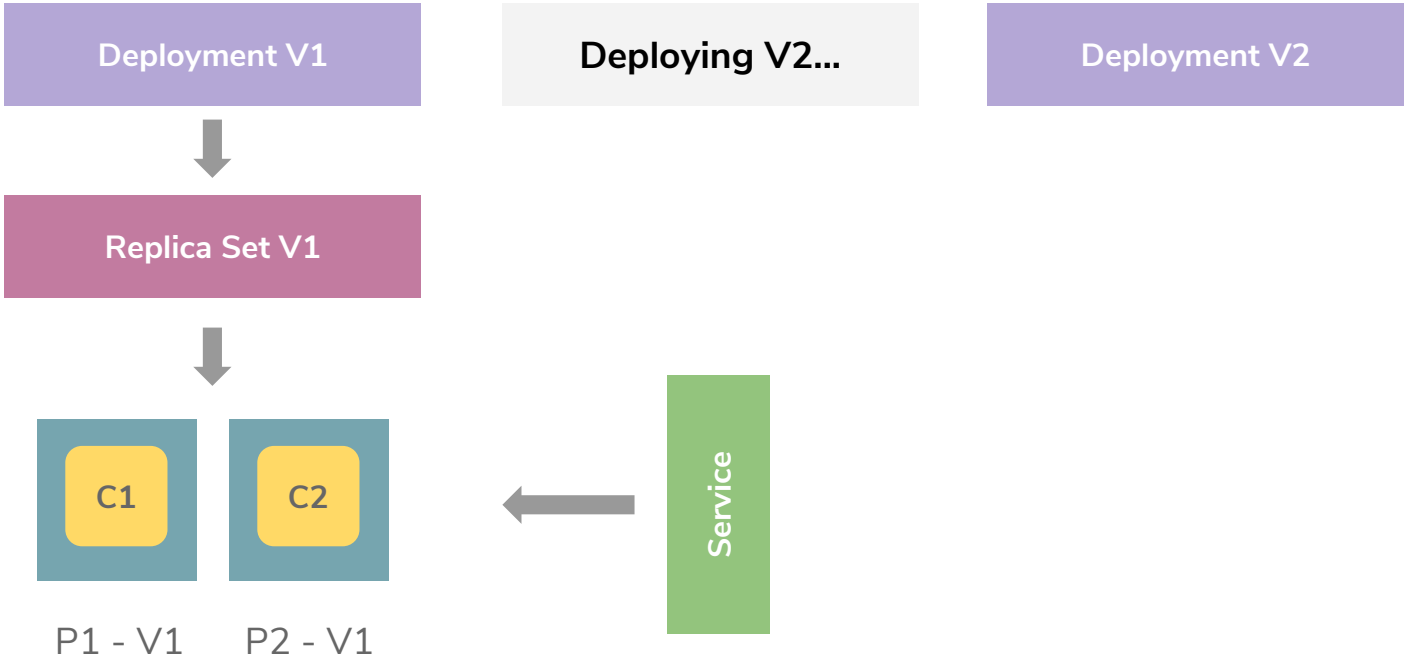
Deployment Process



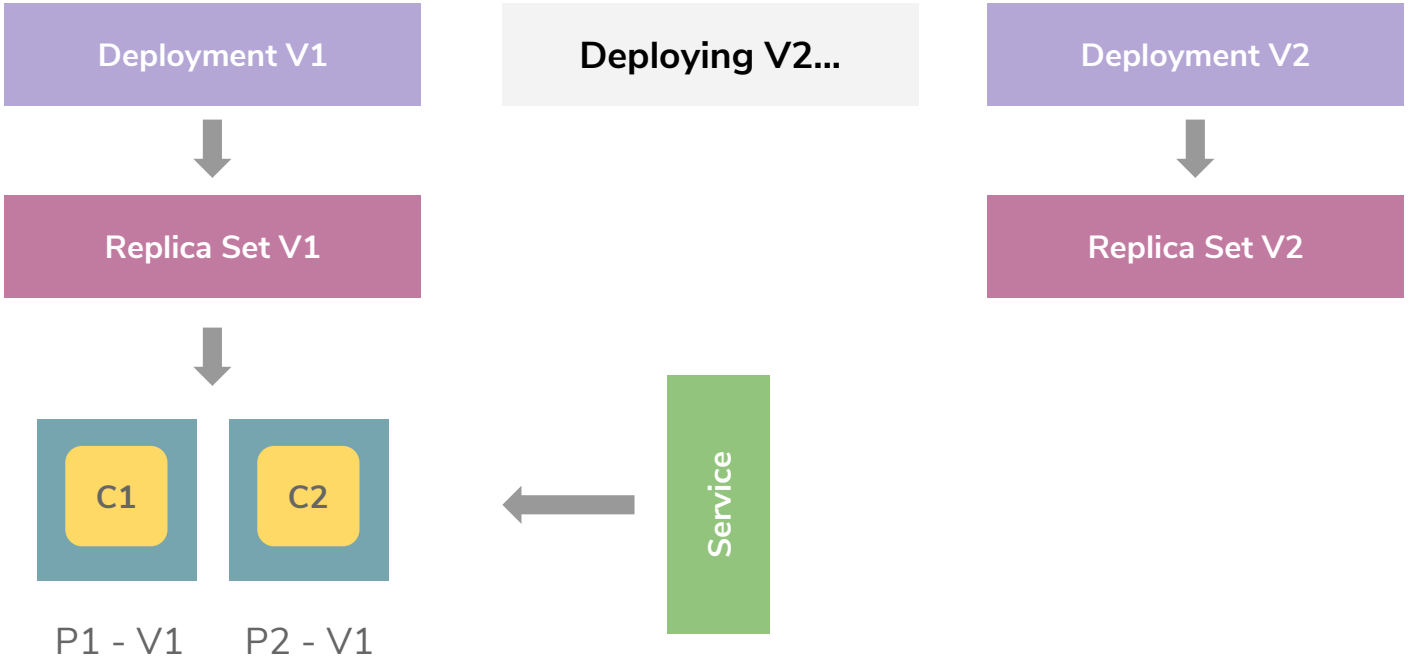
Deployment Process



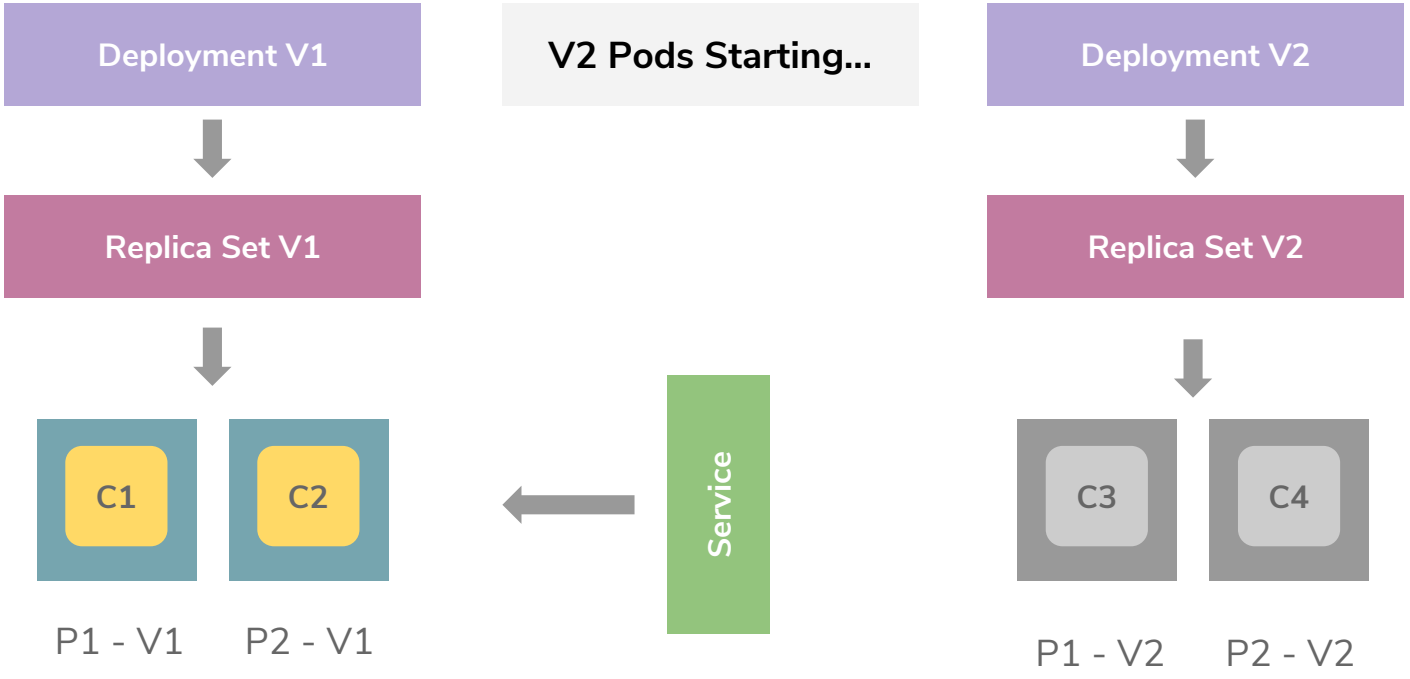
Rolling Update Process



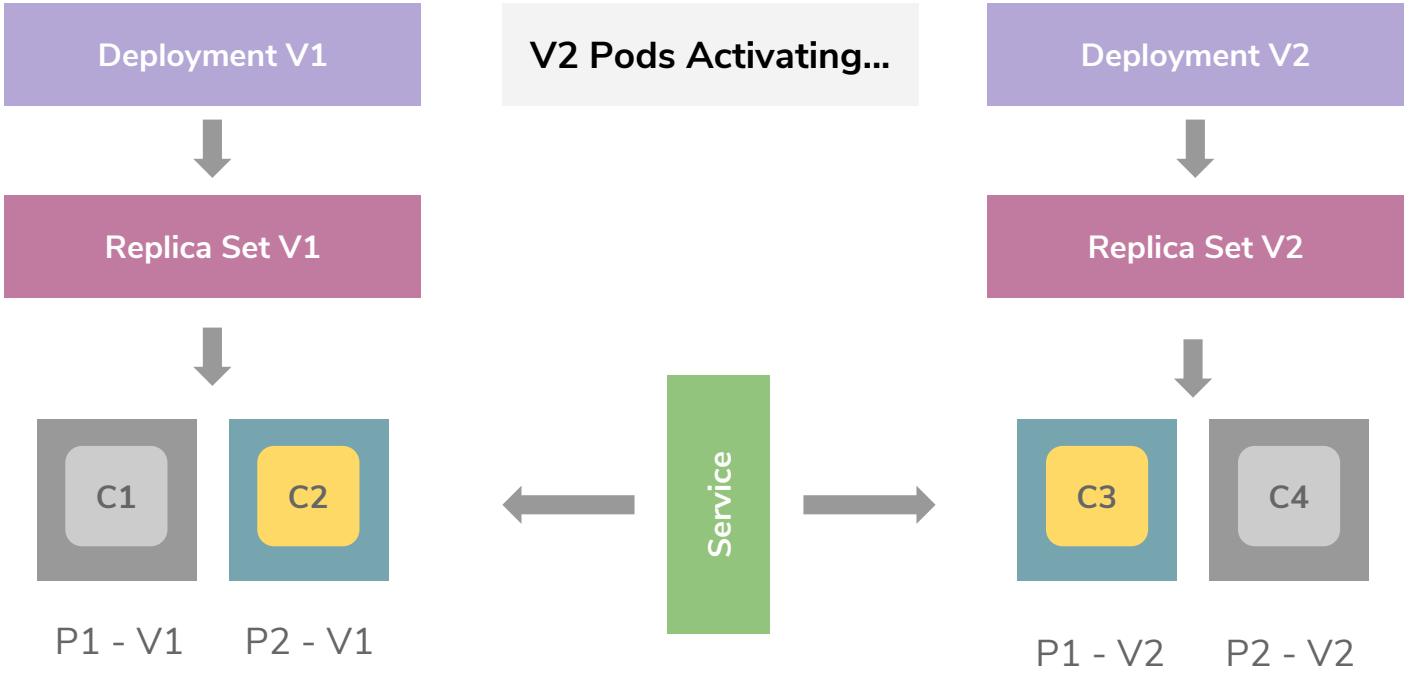
Rolling Update Process



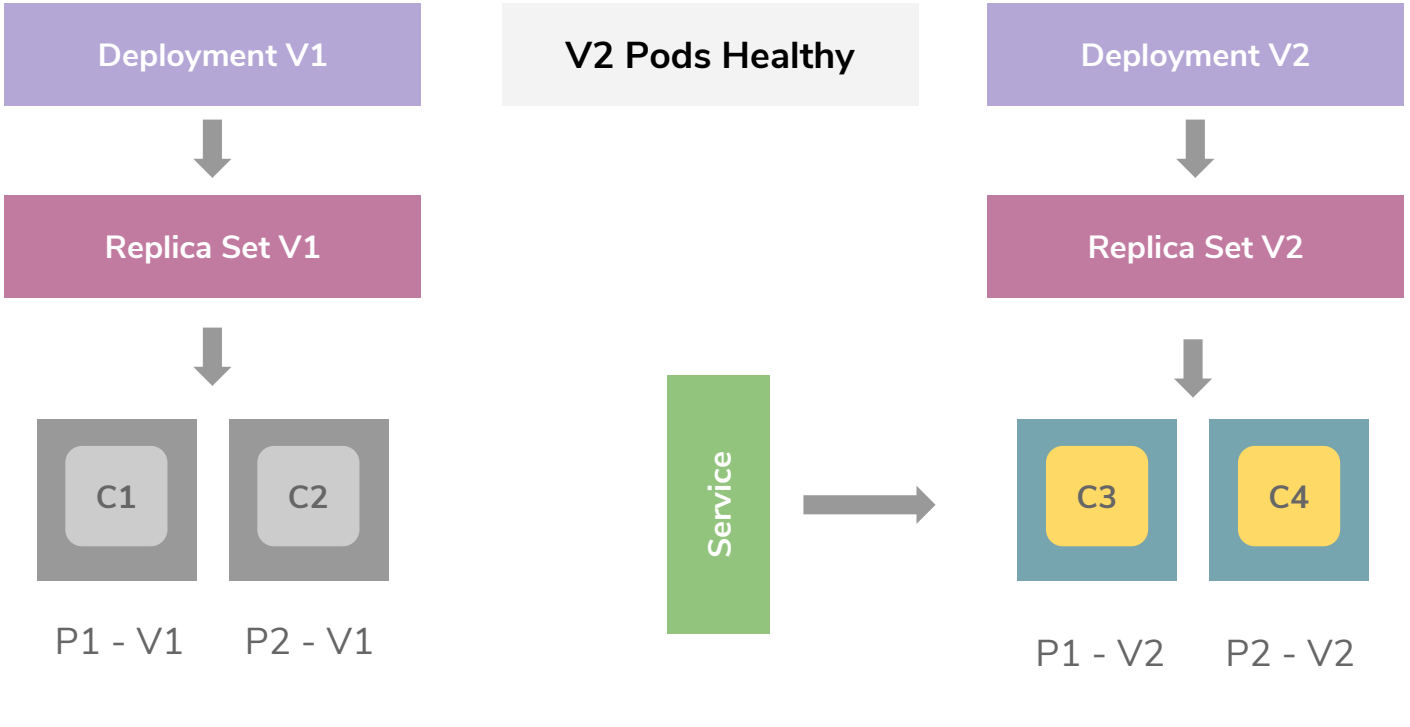
Rolling Update Process



Rolling Update Process



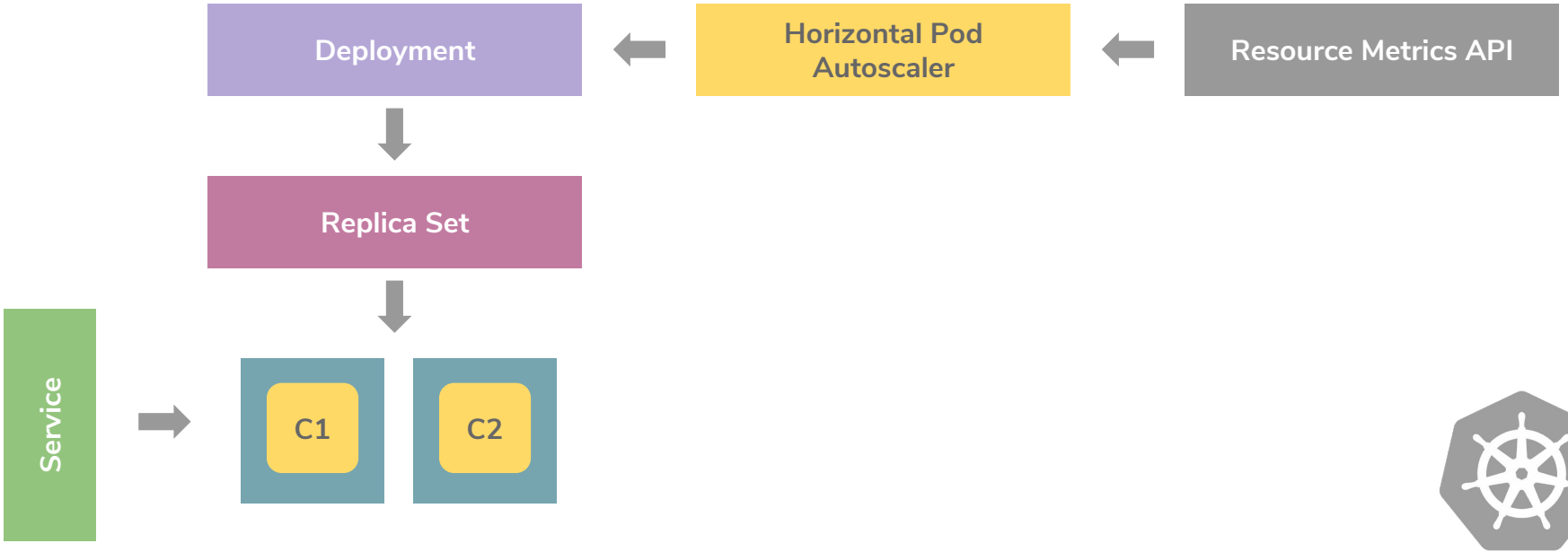
Rolling Update Process



Autoscaling

Horizontal Pod Autoscalers

- Enable autoscaling for pods based on CPU utilization



Package Management

Helm

- Helm is the Kubernetes package manager.
- It uses Charts for defining, installing and upgrading applications on Kubernetes.
- Runtime configurations can be templated and parameterized.
- Existing Charts can be reused and added as dependencies to new Charts.
- Helm is managed by CNCF.

https://docs.helm.sh

Helm Hello World

```
# chart.yaml

name: apps/v1
version:
```

```
# templates/deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  Name: hello-world
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
      - name: hello-world
        image:
gcr.io/google-samples/node-hello
:1.0
        ports:
        - containerPort: 8080
          protocol: TCP
```

```
# templates/service.yaml

kind: Service
apiVersion: v1
metadata:
  name: hello-world
spec:
  type: NodePort
  selector:
    app: hello-world
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 8080
```

Hello World Demo

Questions & Feedback

References

References

- Kubernetes Documentation:
  - https://kubernetes.io/docs/
- An Introduction to Kubernetes:
  - https://www.slideshare.net/imesh/an-introduction-to-kubernetes
- WSO2Con US 2015 Kubernetes: a platform for automating deployment, scaling, and operations:
  - https://www.slideshare.net/BrianGrant11/wso2con-us-2015-kubernetes-a-platform-for-automating-deployment-scaling-and-operations
- Kubernetes: An Overview:
  - https://thenewstack.io/kubernetes-an-overview/

References Cont.

- Helm Documentation:
  - https://docs.helm.sh
- The missing CI/CD Kubernetes component: Helm package manager
  - https://medium.com/@gajus/the-missing-ci-cd-kubernetes-component-helm-package-manager-1fe002aac680

Thank You!