Application life cycle

Deployment

https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#creating-a-deployment

- Desired state is described in a deployment
- **Deployment controller** changes the actual state to the desired state at a controlled rate
- Deployment
    - Creates Replicate Sets
        - Creates pods
- Pod-template-hash label
    - added by the Deployment controller to every POD & ReplicaSet that a Deployment creates
    - This label ensures that child ReplicaSets of a Deployment do not overlap. It is generated by hashing the PodTemplate of the ReplicaSet and using the resulting hash as the label value that is added to the ReplicaSet selector, Pod template labels, and in any existing Pods that the ReplicaSet might have

Use cases

- Create deployment to rollout replica sets
- Updating deployment
    - Deployment rollout is triggered when deployment pod template is changed
    - `.spec.template`

Rollover (Multiple updates in flight)

- Each time a new Deployment is observed by the Deployment controller, a ReplicaSet is created to bring up the desired Pods. If the Deployment is updated, the existing ReplicaSet that controls Pods whose labels match .spec.selector but whose template does not match .spec.template are scaled down. Eventually, the new ReplicaSet is scaled to .spec.replicas and all old ReplicaSets is scaled to 0.
- If you update a Deployment while an existing rollout is in progress, the Deployment creates a new ReplicaSet as per the update and start scaling that up, and rolls over the ReplicaSet that it was scaling up previously – it will add it to its list of old ReplicaSets and start scaling it down.

Label Selector Updates - Not recommended..

- Updating labels on Deployments and pods..
- Try with examples..

Rolling back a deployment

- Deployment's rollout history is kept in the system so that you can rollback anytime you want
- new revision is created if and only if the Deployment's Pod template (.spec.template) is changed,
    - example - label or image. Scaling update do not create a revision

      when you roll back to an earlier revision, only the Deployment's Pod template part is rolled back.

`kubectl rollout status deployment.v1.apps/nginx-deployment`

`kubectl rollout history deployment.v1.apps/nginx-deployment`

`kubectl rollout history deployment.v1.apps/nginx-deployment --revision=2`

`kubectl rollout undo deployment.v1.apps/nginx-deployment --to-revision=2`

- Rollback to old version

Scaling a deployment

`kubectl scale deployment.v1.apps/nginx-deployment --replicas=10`

Autoscaling - Auto increase/decrease replicas based on cpu/memory usage.

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/

`kubectl autoscale deployment.v1.apps/nginx-deployment --min=10 --max=15 --cpu-percent=80`

Proportional scaling

With proportional scaling, you spread the additional replicas across all ReplicaSets. Bigger proportions go to the ReplicaSets with the most replicas and lower proportions go to ReplicaSets with less replicas. Any leftovers are added to the ReplicaSet with the most replicas. ReplicaSets with zero replicas are not scaled up

Pausing and resuming a deployment

```
kubectl rollout pause deployment.v1.apps/nginx-deployment
```

```
kubectl set image deployment.v1.apps/nginx-deployment nginx=nginx:1.9.1
```

```
kubectl set resources deployment.v1.apps/nginx-deployment -c=nginx
--limits=cpu=200m,memory=512Mi
```

```
kubectl rollout resume deployment.v1.apps/nginx-deployment
```

Deployment status

A Deployment enters various states during its lifecycle. It can be progressing while rolling out a new ReplicaSet, it can be complete, or it can fail to progress

Canary deployment

Create another copy of different with same pod label with different name..

The primary, stable release would have a track label with value as stable:

```
  name: frontend
   replicas: 3
   ...
   labels:
      app: guestbook
      tier: frontend
      track: stable
   ...
   image: gb-frontend:v3
```

and then you can create a new release of the guestbook frontend that carries the track label with different value (i.e. canary), so that two sets of pods would not overlap:

```
  name: frontend-canary
  replicas: 1
  ...
  labels:
    app: guestbook
    tier: frontend
    track: canary
  ...
  image: gb-frontend:v4
```

The frontend service would span both sets of replicas by selecting the common subset of their labels (i.e. omitting the track label), so that the traffic will be redirected to both applications:

```
selector:
  app: guestbook
  tier: frontend
```

Self healing

Once the application instances are created, a Kubernetes **Deployment Controller** continuously monitors those instances. If the Node hosting an instance goes down or is deleted, the Deployment controller replaces the instance with an instance on another Node in the cluster. **This provides a self-healing mechanism to address machine failure or maintenance.**

Replicate Sets

- **A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical Pods**
- **The link a ReplicaSet has to its Pods is via the Pods' metadata.ownerReferences field, which specifies what resource the current object is owned by.**
- **A ReplicaSet identifies new Pods to acquire by using its selector. If there is a Pod that has no OwnerReference or the OwnerReference is not a controller and it matches a ReplicaSet's selector, it will be immediately acquired by said ReplicaSet**
- Non-Template Pod acquisitions

- While you can create bare Pods with no problems, it is strongly recommended to make sure that the bare Pods do not have labels which match the selector of one of your ReplicaSets. The reason for this is because a ReplicaSet is not limited to owning Pods specified by its template– it can acquire other Pods
- To delete a ReplicaSet and all of its Pods, use kubectl delete. The Garbage collector automatically deletes all of the dependent Pods by default.
- You can delete a ReplicaSet without affecting any of its Pods using kubectl delete with the --cascade=false option.
- **You can remove Pods from a ReplicaSet by changing their labels. This technique may be used to remove Pods from service for debugging, data recovery, etc**
- **Autoscaling**

  kubectl autoscale rs frontend --max=10

Job

- **Use a Job instead of a ReplicaSet for Pods that are expected to terminate on their own (that is, batch jobs)**
- A Job creates one or more Pods and ensures that a specified number of them successfully terminate
- The Job object will start a new Pod if the first Pod fails or is deleted (for example due to a node hardware failure or a node reboot).
- You can also use a Job to run multiple Pods in parallel.
- When a Job completes, no more Pods are created, but the Pods are not deleted either. Keeping them around allows you to still view the logs of completed pods to check for errors, warnings, or other diagnostic output. The job object also remains after it is completed so that you can view its status. It is up to the user to delete old jobs after noting their status.

CronJob

- A Cron Job creates Jobs on a time-based schedule.
- One CronJob object is like one line of a crontab (cron table) file. It runs a job periodically on a given schedule, written in Cron format.
- The Cronjob is only responsible for creating Jobs that match its schedule, and the Job in turn is responsible for the management of the Pods it represents.

DaemonSet

Use a DaemonSet instead of a ReplicaSet for Pods that provide a machine-level function, such as machine monitoring or machine logging. These Pods have a lifetime that is tied to a machine lifetime:

ReplicationController

ReplicaSets are the successors to *ReplicationControllers*. The two serve the same purpose, and behave similarly, except that a ReplicationController does not support set-based selector requirements as described i

Stateful Sets

- **StatefulSet is the workload API object used to manage stateful applications.**
- **Manages the deployment and scaling of a set of Pods , *and provides guarantees about the ordering and uniqueness* of these Pods.**
- **Like a Deployment , a StatefulSet manages Pods that are based on an identical container spec. Unlike a Deployment, a StatefulSet maintains a sticky identity for each of their Pods.**
- **These pods are created from the same spec, but are not interchangeable: each has a persistent identifier that it maintains across any rescheduling.**

USes:

StatefulSets are valuable for applications that require one or more of the following.

- **Stable, unique network identifiers.**
- **Stable, persistent storage.**
- **Ordered, graceful deployment and scaling.**
- **Ordered, automated rolling updates.**

Exercise

https://kubernetes.io/docs/tutorials/stateful-application/basic-stateful-set/

Components

- Headless service
- statefulSet
- VolumeClaimtemplates

Pod Identity

Ordinal Index

- **For a StatefulSet with N replicas, each Pod in the StatefulSet will be assigned an integer ordinal, from 0 up through N-1, that is unique over the Set.**

Stable Network ID

Each Pod in a StatefulSet derives its hostname from the name of the StatefulSet and the ordinal of the Pod. The pattern for the constructed hostname is $(statefulset name)-$(ordinal). The example above will create three Pods named web-0,web-1,web-2

A StatefulSet can use a Headless Service to control the domain of its Pods. The domain managed by this Service takes the form: $(service name).$(namespace).svc.cluster.local, where "cluster.local" is the cluster domain.

`$(podname).$(governing service domain)`,

Cluster Domain	Service (ns/name)	StatefulSet (ns/name)	StatefulSet Domain	Pod DNS	Pod Hostname
cluster.local	default/nginx	default/web	nginx.default.svc.cluster.local	web-{0..N-1}.nginx.default.svc.cluster.local	web-{0..N-1}
cluster.local	foo/nginx	foo/web	nginx.foo.svc.cluster.local	web-{0..N-1}.nginx.foo.svc.cluster.local	web-{0..N-1}
kube.local	foo/nginx	foo/web	nginx.foo.svc.kube.local	web-{0..N-1}.nginx.foo.svc.kube.local	web-{0..N-1}

Stable Storage

Pod Name Label

Deployment and scaling

- **For a StatefulSet with N replicas, when Pods are being deployed, they are created sequentially, in order from {0..N-1}.**
- **When Pods are being deleted, they are terminated in reverse order, from {N-1..0}.**
- **Before a scaling operation is applied to a Pod, all of its predecessors must be Running and Ready.**
- **Before a Pod is terminated, all of its successors must be completely shutdown.**

Limitations

- StatefulSets currently require a Headless Service to be responsible for the network identity of the Pods. You are responsible for creating this Service.
- StatefulSets do not provide any guarantees on the termination of pods when a StatefulSet is deleted. To achieve ordered and graceful termination of the pods in the StatefulSet, it is possible to scale the StatefulSet down to 0 prior to deletion.
- Deleting and/or scaling a StatefulSet down will *not* delete the volumes associated with the StatefulSet. This is done to ensure data safety, which is generally more valuable than an automatic purge of all related StatefulSet resources.
- The storage for a given Pod must either be provisioned by a PersistentVolume Provisioner based on the requested `storage class`, or pre-provisioned by an admin.

Update strategies

On Delete

- When a StatefulSet's `.spec.updateStrategy.type` is set to `OnDelete`, the StatefulSet controller will not automatically update the Pods in a StatefulSet. Users must manually delete Pods to cause the controller to create new Pods that reflect modifications made to a StatefulSet's `.spec.template`

Rolling Update

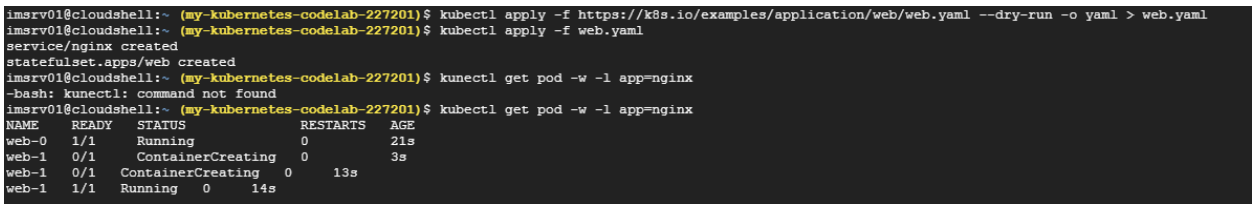
- When a StatefulSet's `.spec.updateStrategy.type` is set to `RollingUpdate`, the StatefulSet controller will delete and recreate each Pod in the StatefulSet. It will proceed in the same order as Pod termination (from the largest ordinal to the smallest), updating each Pod one at a time. It will wait until an updated Pod is Running and Ready prior to updating its predecessor

Partitions

- The `RollingUpdate` update strategy can be partitioned, by specifying a `.spec.updateStrategy.rollingUpdate.partition`. If a partition is specified, all Pods with an ordinal that is greater than or equal to the partition will be updated when the StatefulSet's `.spec.template` is updated. All Pods with an ordinal that is less than the partition will not be updated, and, even if they are deleted, they will be recreated at the previous version.

Example:

Pods are created in order, second pod is created only when the first one is ready and running



headless service and stateful set

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get svc nginx
NAME    TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
nginx   ClusterIP   None         <none>        80/TCP    10m
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get stafulset web
error: the server doesn't have a resource type "stafulset"
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get statefulset web
NAME   DESIRED   CURRENT   AGE
web    2         2         10m
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$
```

Pods have stable network identities - hostname

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ for i in 0 1; do kubectl exec web-$i -- sh -c 'hostname'; done
web-0
web-1
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$
```

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl run test-$RANDOM --image=busybox:1.28 --restart=Never --rm -it -- sh
If you don't see a command prompt, try pressing enter.
/ # nslookup web-0.nginx
Server:    10.12.0.10
Address 1: 10.12.0.10 kube-dns.kube-system.svc.cluster.local

Name:      web-0.nginx
Address 1: 10.8.0.22 web-0.nginx.default.svc.cluster.local
/ #
```

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl run test-$RANDOM --image=busybox:1.28 --restart=Never --rm -it -- sh
If you don't see a command prompt, try pressing enter.
/ # nslookup web-1.nginx
Server:    10.12.0.10
Address 1: 10.12.0.10 kube-dns.kube-system.svc.cluster.local

Name:      web-1.nginx
Address 1: 10.8.0.23 web-1.nginx.default.svc.cluster.local
/ #
```

The CNAME of the headless service points to SRV records (one for each Pod that is Running and Ready). The SRV records point to A record entries that contain the Pods' IP addresses.

Deletion, IP gets changes but hostname and DNS entry do not get changed

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl delete pod -l app=nginx
pod "web-0" deleted
pod "web-1" deleted
^C
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get pod -l app=nginx -w
NAME    READY   STATUS              RESTARTS   AGE
web-0   1/1     Running             0          13s
web-1   0/1     ContainerCreating   0          9s
web-1   0/1   ContainerCreating   0     10s
web-1   1/1   Running   0     11s
```

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ for i in 0 1; do kubectl exec web-$i -- sh -c 'echo $(hostname) > /usr/share/nginx/html/index.html'; done
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ for i in 0 1; do kubectl exec -it web-$i -- curl localhost; done
web-0
web-1web-0
web-1
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ for i in 0 1; do kubectl exec -it web-$i -- curl localhost; done
web-0
web-1
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$
```

Scaling

the StatefulSet controller created each Pod sequentially with respect to its ordinal index, and it waited for each Pod's predecessor to be Running and Ready before launching the subsequent Pod.

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get pods -w -l app=nginx
NAME    READY   STATUS              RESTARTS   AGE
web-0   1/1     Running             0          5m29s
web-1   1/1     Running             0          5m25s
web-2   0/1     ContainerCreating   0          3s
web-2   0/1   ContainerCreating   0     12s
web-2   1/1   Running   0     12s
web-3   0/1   Pending   0     0s
web-3   0/1   Pending   0     0s
web-3   0/1   Pending   0     3s
web-3   0/1   Pending   0     3s
web-3   0/1   ContainerCreating   0     3s
web-3   0/1   ContainerCreating   0     13s
web-3   1/1   Running   0     13s
web-4   0/1   Pending   0     0s
web-4   0/1   Pending   0     0s
web-4   0/1   Pending   0     2s
web-4   0/1   Pending   0     2s
web-4   0/1   ContainerCreating   0     2s
web-4   0/1   ContainerCreating   0     20s
web-4   1/1   Running   0     21s
```

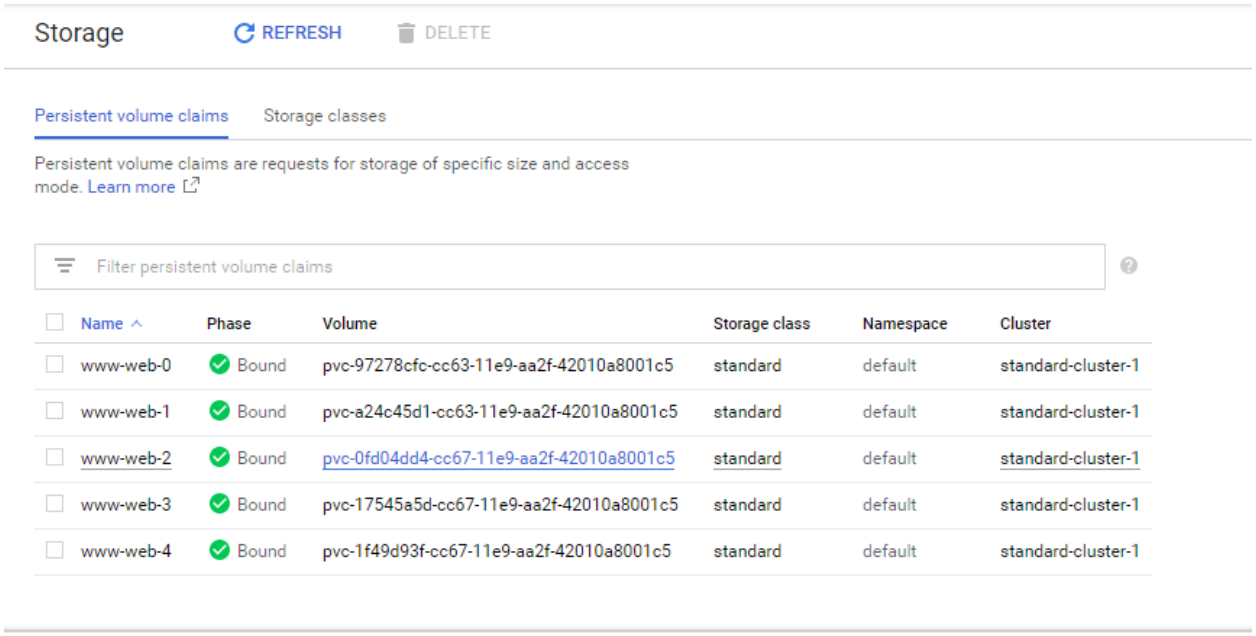
Scale down

The controller deleted one Pod at a time, in reverse order with respect to its ordinal index, and it waited for each to be completely shutdown before deleting the next.

```
^Cimsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl patch sts web -p '{"spec":{"replicas": 3}}'
statefulset.apps/web patched
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get pods -w -l app=nginx
NAME    READY   STATUS        RESTARTS   AGE
web-0   1/1     Running       0          7m52s
web-1   1/1     Running       0          7m48s
web-2   1/1     Running       0          2m26s
web-3   1/1     Running       0          2m14s
web-4   0/1     Terminating   0          2m
web-4   0/1   Terminating   0     2m8s
web-4   0/1   Terminating   0     2m8s
web-3   1/1   Terminating   0     2m22s
web-3   0/1   Terminating   0     2m23s
web-3   0/1   Terminating   0     2m24s
web-3   0/1   Terminating   0     2m24s
```

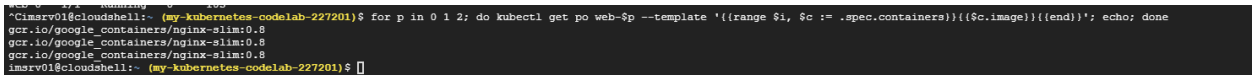
PVC were not delete

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get pvc
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
www-web-0   Bound    pvc-97278cfc-cc63-11e9-aa2f-42010a8001c5   1Gi        RWO            standard       28m
www-web-1   Bound    pvc-a24c45d1-cc63-11e9-aa2f-42010a8001c5   1Gi        RWO            standard       28m
www-web-2   Bound    pvc-0fd04dd4-cc67-11e9-aa2f-42010a8001c5   1Gi        RWO            standard       3m46s
www-web-3   Bound    pvc-17545a5d-cc67-11e9-aa2f-42010a8001c5   1Gi        RWO            standard       3m34s
www-web-4   Bound    pvc-1f49d93f-cc67-11e9-aa2f-42010a8001c5   1Gi        RWO            standard       3m20s
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$
```

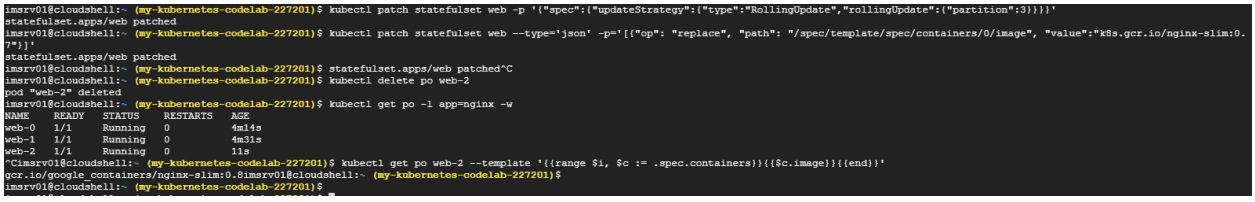


Patching - image updated

The Pods in the StatefulSet are updated in reverse ordinal order. The StatefulSet controller terminates each Pod, and waits for it to transition to Running and Ready prior to updating the next Pod.



even though the update strategy is `RollingUpdate` the StatefulSet controller restored the Pod with its original container. This is because the ordinal of the Pod is less than the `partition` specified by the `updateStrategy`.



Rolling out a canary

When you changed the partition, the StatefulSet controller automatically updated the web-2 Pod because the Pod's ordinal was greater than or equal to the partition.

web-1 was restored to its original configuration because the Pod's ordinal was less than the partition. When a partition is specified, all Pods with an ordinal that is greater than or equal to the partition will be

updated when the StatefulSet's .spec.template is updated. If a Pod that has an ordinal less than the partition is deleted or otherwise terminated, it will be restored to its original configuration.

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl patch statefulset web -p '{"spec":{"updateStrategy":{"type":"RollingUpdate","rollingUpdate":{"partition":2}}}}'
statefulset.apps/web patched
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get po -l app=nginx -w
NAME    READY   STATUS              RESTARTS   AGE
web-0   1/1     Running             0          6m8s
web-1   1/1     Running             0          6m25s
web-2   0/1     ContainerCreating   0          6s
web-2   0/1   ContainerCreating   0     10s
web-2   1/1   Running   0     12s
^Cimsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get po web-2 --template '{{range $i, $c := .spec.containers}}{{$c.image}}{{end}}
> ^C
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get po web-2 --template '{{range $i, $c := .spec.containers}}{{$c.image}}{{end}}'
k8s.gcr.io/nginx-slim:0.7imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl delete po web-1
pod "web-1" deleted
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get po -l app=nginx -w
NAME    READY   STATUS              RESTARTS   AGE
web-0   1/1     Running             0          6m59s
web-1   0/1     ContainerCreating   0          6s
web-2   1/1     Running             0          57s
web-1   0/1   ContainerCreating   0     9s
web-1   1/1   Running   0     10s
^Cimsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get po web-1 --template '{{range $i, $c := .spec.containers}}{{$c.image}}{{end}}'
gcr.io/google_containers/nginx-slim:0.8imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$
```

Phased Roll Outs

You can perform a phased roll out (e.g. a linear, geometric, or exponential roll out) using a partitioned rolling update in a similar manner to how you rolled out a canary.

To perform a phased roll out, set the `partition` to the ordinal at which you want the controller to pause the update.

The partition is currently set to `2`. Set the partition to `0`.

```
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl patch statefulset web -p '{"spec":{"updateStrategy":{"type":"RollingUpdate","rollingUpdate":{"partition":0}}}}'
statefulset.apps/web patched
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ kubectl get po -l app=nginx -w
NAME    READY   STATUS              RESTARTS   AGE
web-0   1/1     Running             0          9m7s
web-1   0/1     ContainerCreating   0          5s
web-2   1/1     Running             0          3m5s
web-1   0/1   ContainerCreating   0     10s
web-1   1/1   Running   0     11s
web-0   1/1   Terminating   0     9m13s
web-0   0/1   Terminating   0     9m14s
web-0   0/1   Terminating   0     9m15s
web-0   0/1   Terminating   0     9m15s
web-0   0/1   Pending   0     0s
web-0   0/1   Pending   0     0s
web-0   0/1   ContainerCreating   0     0s
web-0   0/1   ContainerCreating   0     6s
web-0   1/1   Running   0     7s
```

```
^Cimsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$ for p in 0 1 2; do kubectl get po web-$p --template '{{range $i, $c := .spec.containers}}{{$c.image}}{{end}}'; echo; done
k8s.gcr.io/nginx-slim:0.7
k8s.gcr.io/nginx-slim:0.7
k8s.gcr.io/nginx-slim:0.7
imsrv01@cloudshell:~ (my-kubernetes-codelab-227201)$
```