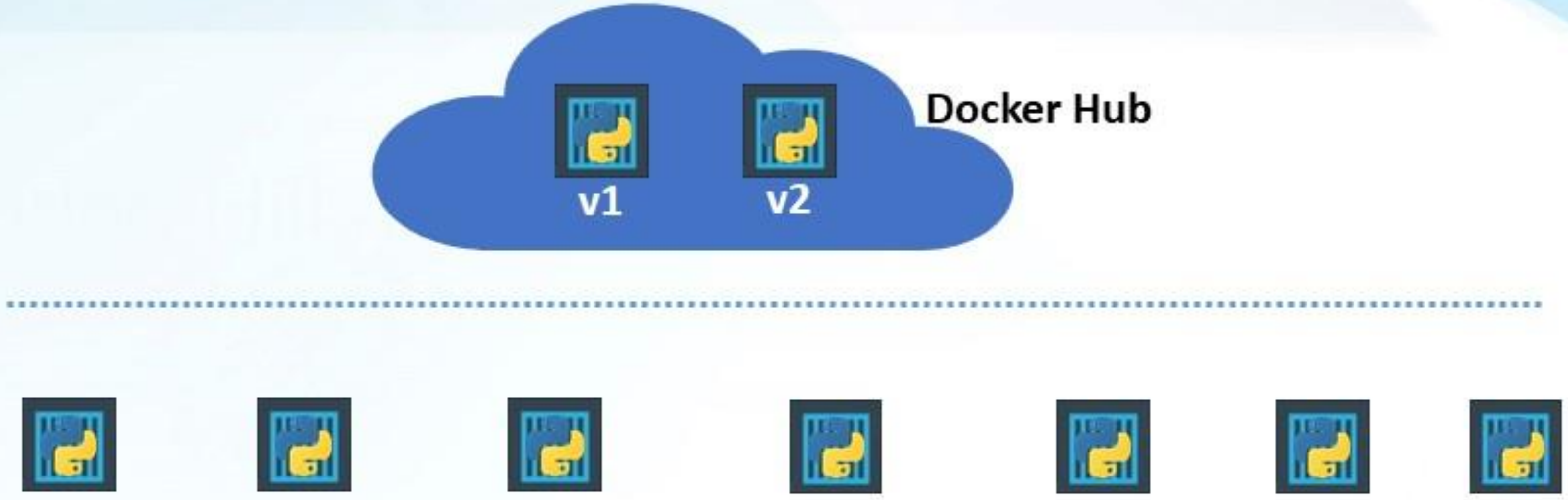
Kubernetes Concepts - Deployment

Deployments

Docker Hub
v1

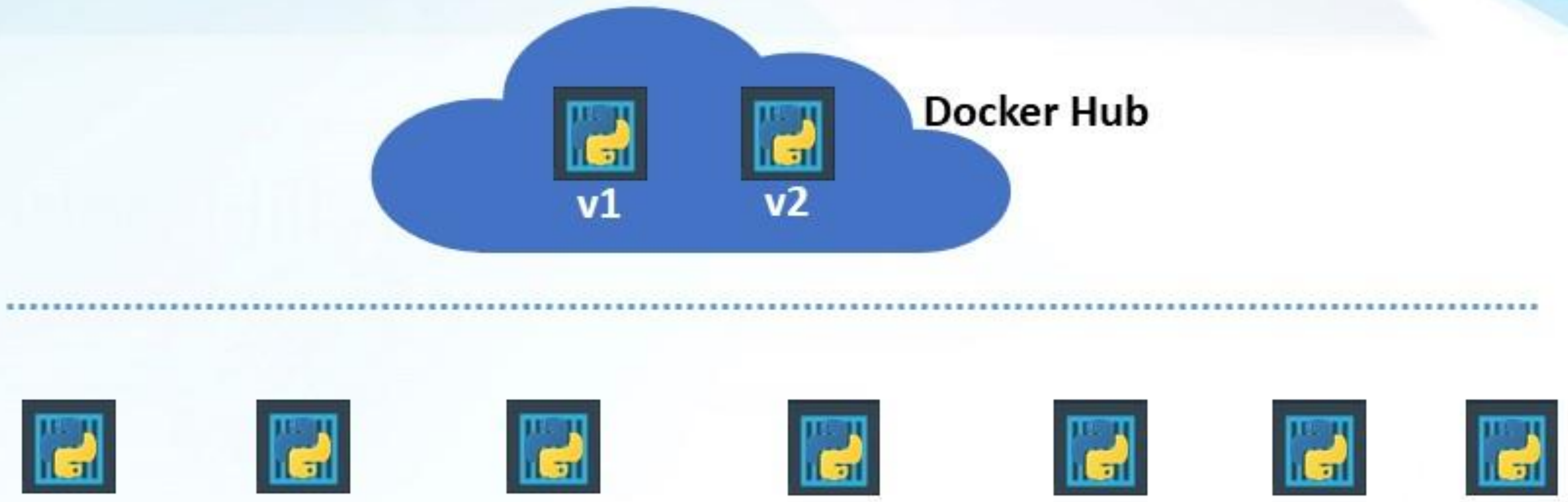
- We have a web server that needs to be deployed in a production environment.
- At later point we may want to deploy many such instances of our web server

Deployments



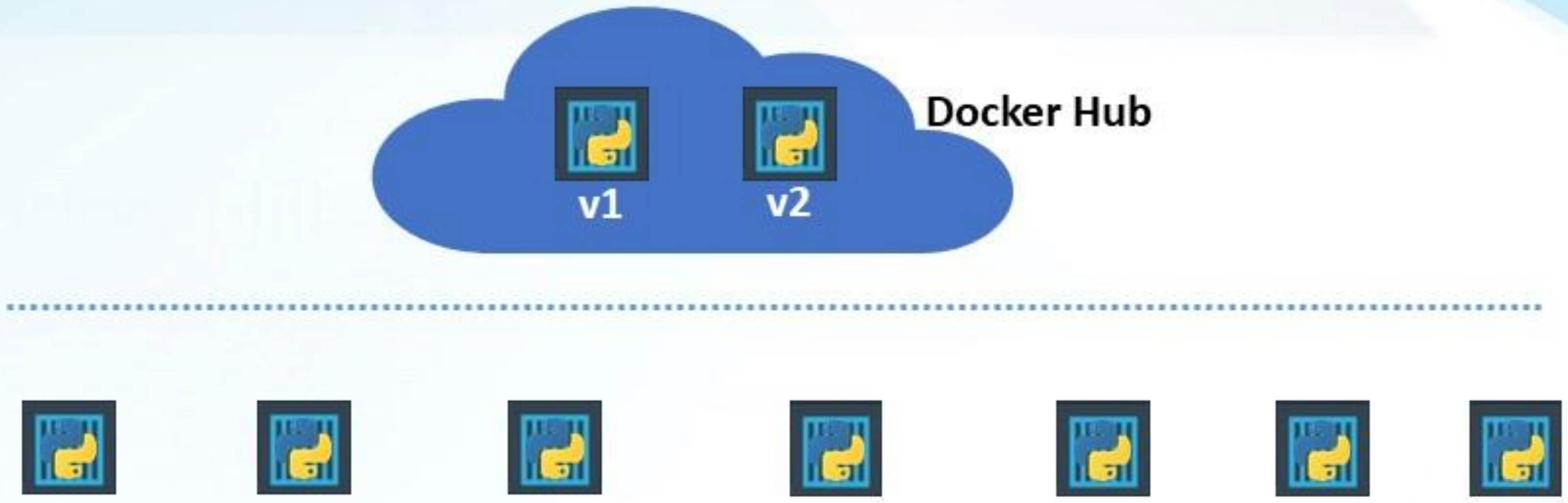
- Whenever newer versions of application builds become available on the docker registry, you would like to upgrade your docker instances seamlessly.
- If we upgrade all instance at once, It will impact users accessing our application.
- We may have to upgrade them one after the other and that kind of upgrade is known as rolling updates

Deployments



- Suppose one of the upgrades we performed resulted in an unexpected error and we need to undo the recent change
- We would like to be able to roll back the changes that were recently carried out.

Deployments



- Finally let's say we would like to make multiple changes to our environment such as upgrading the underlying Web Server versions as well as scaling our environment and also modifying the resource allocations etc.
- We do not want to apply each change immediately after the command is run, instead we want to apply a pause to your environment, make the changes and then resumes so that all the changes are rolled out together.
- All of these capabilities are available with the Kubernetes deployments.

Deployment

- Deployment is a Kubernetes object that comes higher in the hierarchy
- Deployment provides us with the capability to upgrade the underlying instances seamlessly using rolling updates, undo changes and pause and resume changes as required.

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
    app: myapp
    type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
  replicas: 3
  selector:
  matchLabels:
    type: front-end
```

- How do we create a deployment?
- First, we create the deployment definition file
- The contents of the deployment definition file are exactly similar to the replica set definition file
- Except for the kind which is now going to be “Deployment”.
- Rest everything remains the same

Creating Deployment

- Create a deployment

```
kubectl create –f deployment-definition.yml
```

- View the list of created deployments

```
kubectl get deployments
```

- Deployment automatically creates a replica set. View the list of created replica set

```
kubectl get replicaset
```

- Replica sets ultimately create pods. View the list of created Pods

```
kubectl get pods
```

Creating Deployment

- See all created objects

```
kubectl get all
```

- **Describe Deployment**

```
kubectl describe deployment
```

Deployment with YAML

Exercise 23

Introduction: Let us start with deployments! Given a deployment –definition.yml file.

Instruction: Add all the root level properties to it. **Note**: Only add the properties, not any values yet

deployment-definition.yml

```
```

Exercise 23 - Solution

Introduction: Let us start with deployments! Given a deployment –definition.yml file.

Instruction: Add all the root level properties to it. **Note**: Only add the properties, not any values yet

```
deployment-definition.yml
```

```yaml
apiVersion:
kind:
metadata:
spec:
```

Exercise 24

Introduction: Let us now add the values for Deployment. Deployment is under apiVersion apps/v1

Instruction: Update values for apiVersion and kind

deployment-definition.yml

```
apiVersion:
kind:
metadata:
spec:
```

Exercise 24 - Solution

Introduction: Let us now add the values for Deployment. Deployment is under apiVersion apps/v1

Instruction: Update values for apiVersion and kind

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
spec:
```

Exercise 25

Introduction: Let us now add the values for metadata

Instruction: Name the Deployment frontend. And add labels app=>mywebsite and tier=> frontend

deployment-definition.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
spec:
```

Exercise 25 - Solution

Introduction: Let us now add the values for metadata

Instruction: Name the Deployment frontend. And add labels app=>mywebsite and tier=> frontend

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
```

Exercise 26

Introduction: Let us now get to the specification

Instruction: The spec section for Deployment has 3 fields: replicas, templates and selector. Simply add these properties. Do not add any values

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
```

SkillAssure

Exercise 26 - Solution

Introduction: Let us now get to the specification

Instruction: The spec section for Deployment has 3 fields: replicas, template and selector. Simply add these properties. Do not add any values

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas:
  template:
  selector:
```

Exercise 27

Introduction: Let us update the number of replicas to 4

```
deployment-definition.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas:
  template:
  selector:
```

Exercise 27 - Solution

Introduction: Let us update the number of replicas to 4

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
  selector:
```

Exercise 28

Introduction: The template section expects a Pod definition. Luckily, we have the one we created in the previous set of exercises.

Instruction: Let us copy the contents of the pod-definition.yml file, except for the apiVersion and kind and place it under the template section.

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
     app: myapp
spec:
  containers:
     - name: nginx
       image: nginx
```

deployment-definition.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
     app: mywebsite
     tier: frontend
spec:
  replicas: 4
  template:
  selector:
```

SkillAssure

Exercise 28 - Solution

Introduction: The template section expects a Pod definition. Luckily, we have the one we created in the previous set of exercises.

Instruction: Let us copy the contents of the pod-definition.yml file, except for the apiVersion and kind and place it under the template section.

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
    - name: nginx
      image: nginx
```

deployment-definition.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
    spec:
      containers:
        - name: nginx
          image: nginx
  selector:
```

SkillAssure

Exercise 29

<u>Introduction</u>: Let us now link the pods to the Deployments by updating selectors

Instruction: Add a property “matchLabels”under selector and copy the labels defined in the pod-definition under it.

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
    - name: nginx
      image: nginx
```

deployment-definition.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
    spec:
      containers:
        - name: nginx
          image: nginx
  selector:
```

SkillAssure

Exercise 29 - Solution

Introduction: Let us now link the pods to the Deployments by updating selectors

Instruction: Add a property “matchLabels”under selector and copy the labels defined in the pod-definition under it.

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
    - name: nginx
      image: nginx
```

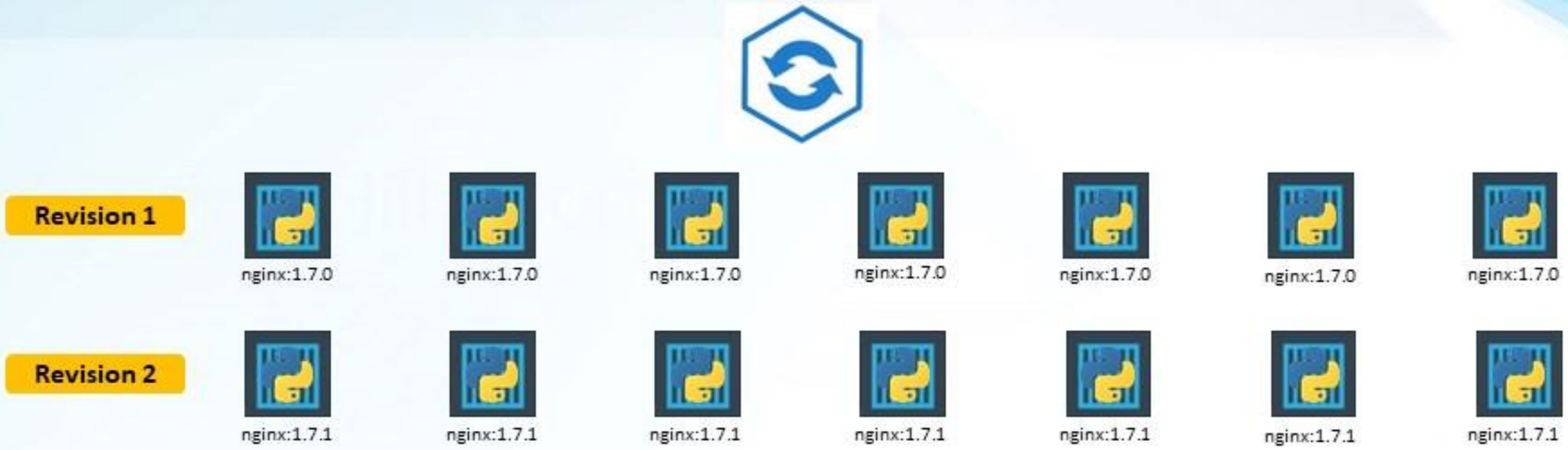
deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
    spec:
      containers:
        - name: nginx
          image: nginx
  selector:
    matchLabels:
      app: myapp
```

Exercise

Deployments
Update & Rollback

Rollout and Versioning



- When you first create a deployment it triggers a rollout
- A new rollout creates a new deployment revision. Let's call it **revision 1**.
- In the future when you upgrade the application meaning when the container version is updated to a new one, a new rollout is triggered, and a new deployment revision is created named **revision 2**.
- This helps us keep track of the changes made to our deployment and enables us to roll back to a previous version of deployment if necessary.

Creating Deployment

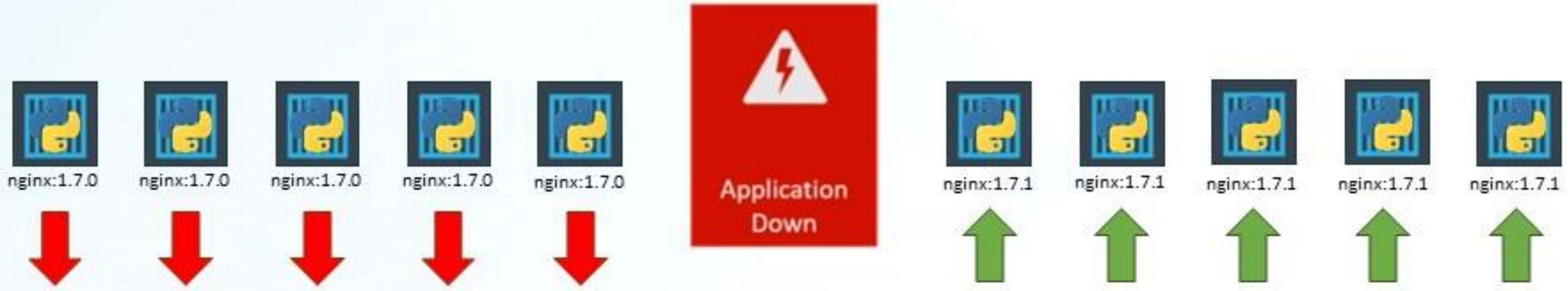
- See status of the rollout

```
kubectl rollout status deployment/myapp-deployment
```

- See the revision history

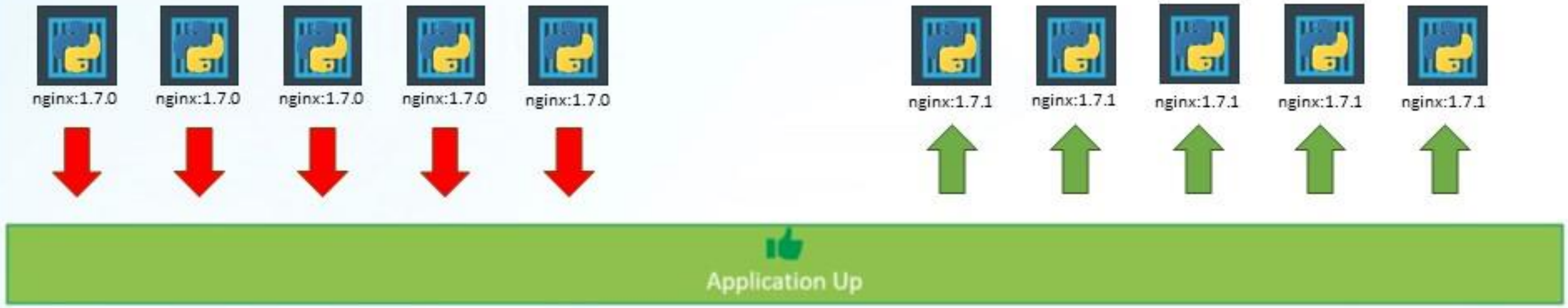
```
kubectl rollout history deployment/myapp-deployment
```

Deployment Strategy - Recreate



- Let's say we have five replicas of your web application instance deployed.
- One way to upgrade these to a newer version is to destroy all of these and then create newer version of application instances.
- The problem with this is that during the period after the older versions are down and before a newer version is up the application is down and inaccessible to users.
- This strategy is known as **Recreate Strategy**
- This is not the default deployment strategy.

Deployment Strategy – Rolling Update



- The second strategy is where we do not destroy all of them at once.
- Instead we take down the older version and bring up a newer version one by one.
- This way the application never goes down and the upgrade is seamless.
- Rolling update is the default deployment strategy, if you do not specify a strategy while creating the deployment it will assume it to be rolling update.

Updating Deployment

deployment-definition.yml

```
apiVersion: v1
kind: ReplicaSet
metadata:
  name: Deployment
  labels:
    app: myapp
    type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
  replicas: 3
  selector:
  matchLabels:
    type: front-end
```

- How do we update our deployment?
- When we say update it could be different things
  - Updating your application version by updating the version of docker containers used
  - Updating their labels or
  - Updating the number of replicas etc..
- Since we already have a deployment definition file it is easy for us to modify these files.

SkillAssure

Updating Deployment

deployment-definition.yml

```
apiVersion: v1
kind: ReplicaSet
metadata:
  name: Deployment
  labels:
    app: myapp
    type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.7.1
  replicas: 3
  selector:
  matchLabels:
    type: front-end
```

- Apply Updates

  kubectl apply -f deployment-definition.yml

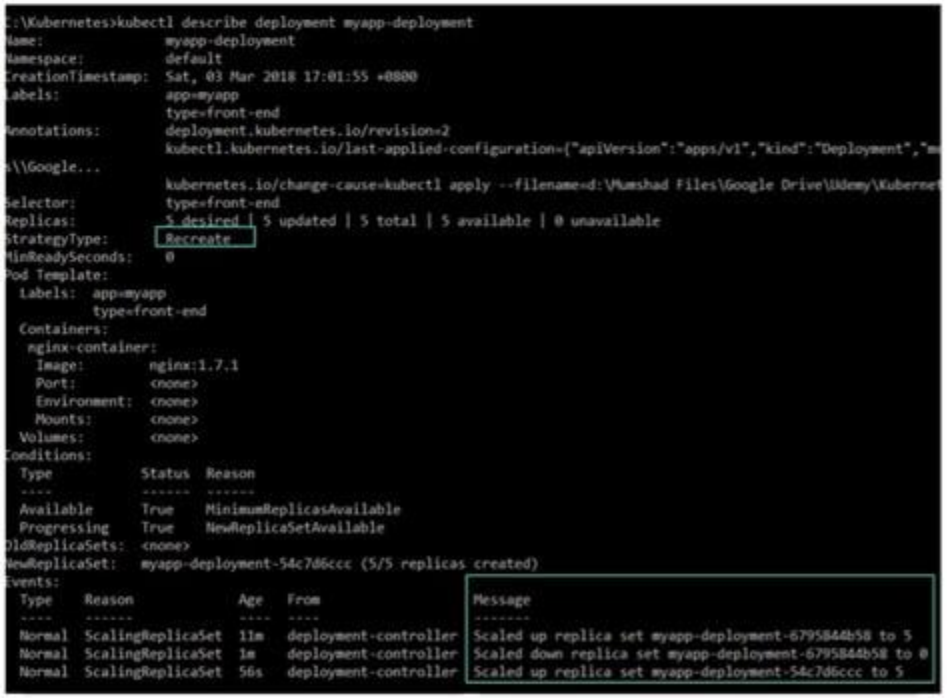
- A new rollout will be triggered, and a new revision of the deployment is created.

- Option 2

  kubectl set image deployment/myapp-deployment \ nginx=nginx:1.7.1

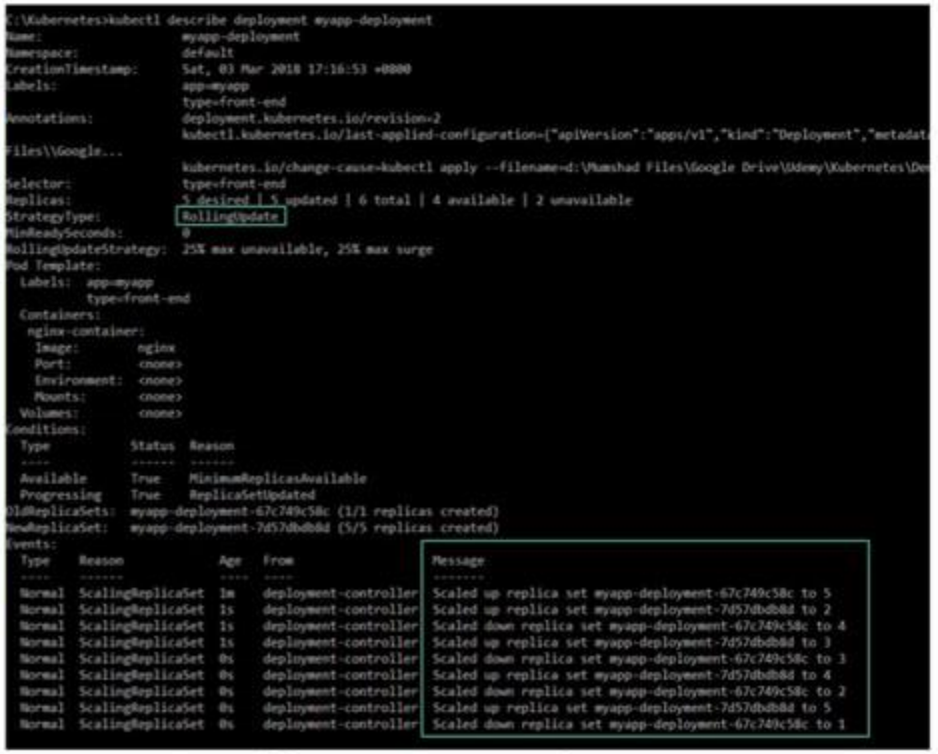
- Remember file will not get updated

Updating Deployment



```
:\Kubernetes>kubectl describe deployment myapp-deployment
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Sat, 03 Mar 2018 17:01:55 +0800
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision=2
                        kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","m
s\\Google...
                        kubernetes.io/change-cause=kubectl apply --filename=d:\Mumshad Files\Google Drive\Udemy\Kubernet
Selector:               type=front-end
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:           Recreate
MinReadySeconds:        0
Pod Template:
  Labels:  app=myapp
           type=front-end
  Containers:
   nginx-container:
    Image:        nginx:1.7.1
    Port:         <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   myapp-deployment-54c7d6ccc (5/5 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  11m   deployment-controller  Scaled up replica set myapp-deployment-6795844b58 to 5
  Normal  ScalingReplicaSet  1m    deployment-controller  Scaled down replica set myapp-deployment-6795844b58 to 0
  Normal  ScalingReplicaSet  56s   deployment-controller  Scaled up replica set myapp-deployment-54c7d6ccc to 5
```

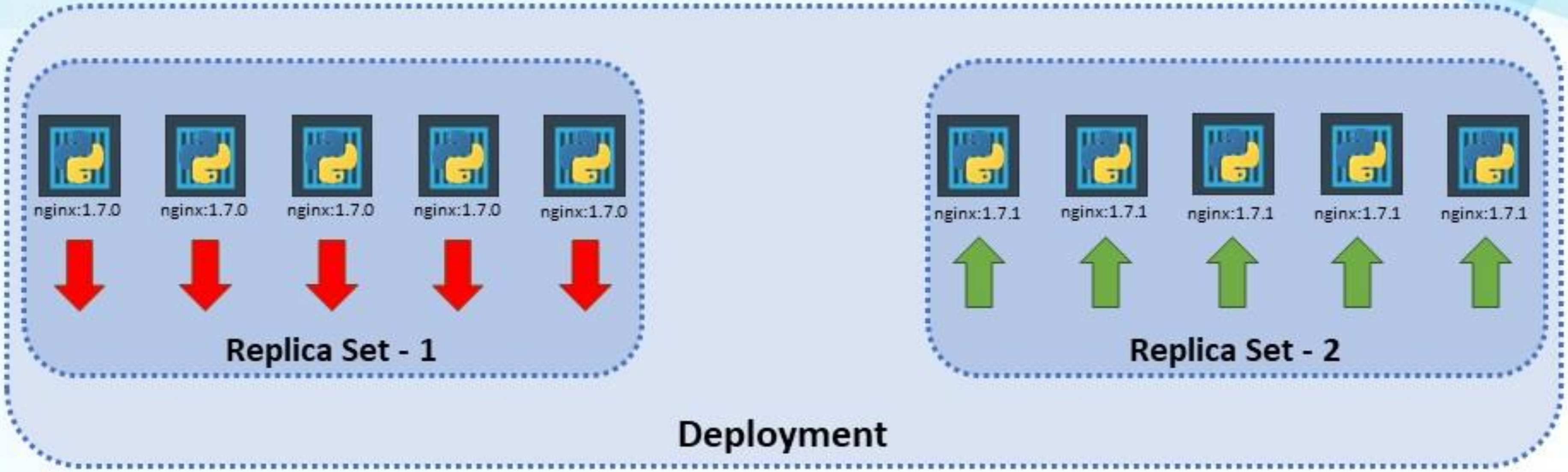
Recreate



```
:\Kubernetes>kubectl describe deployment myapp-deployment
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Sat, 03 Mar 2018 17:16:53 +0800
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision=2
                        kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","metadat
Files\\Google...
                        kubernetes.io/change-cause=kubectl apply --filename=d:\Mumshad Files\Google Drive\Udemy\Kubernetes\De
Selector:               type=front-end
Replicas:               5 desired | 5 updated | 6 total | 4 available | 2 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=myapp
           type=front-end
  Containers:
   nginx-container:
    Image:        nginx
    Port:         <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:  myapp-deployment-67c749c58c (1/1 replicas created)
NewReplicaSet:   myapp-deployment-7d57dbdb8d (5/5 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  1m    deployment-controller  Scaled up replica set myapp-deployment-67c749c58c to 5
  Normal  ScalingReplicaSet  1s    deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 2
  Normal  ScalingReplicaSet  1s    deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 4
  Normal  ScalingReplicaSet  1s    deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 3
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 3
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 4
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 2
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 5
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 1
```

RollingUpdate

Deployment - Upgrades

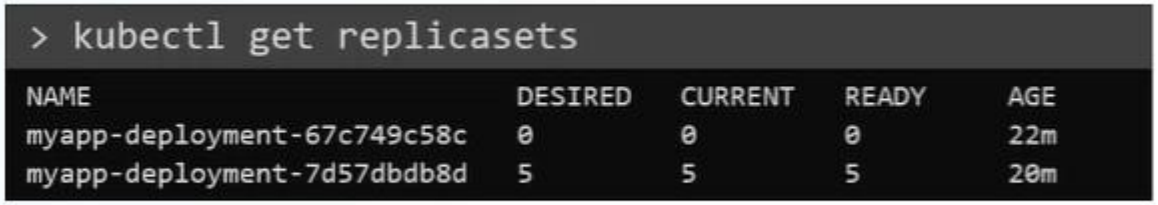


- When a new deployment is created
  - It first creates a replica set automatically
  - Then it creates the number of Pods required to meet the number of replicas

- When you upgrade your application.
  - Kubernetes deployment object creates a new replica set under the hood
  - Then starts deploying the containers there at the same time taking down the pods in the old replica set following a rolling update strategy.

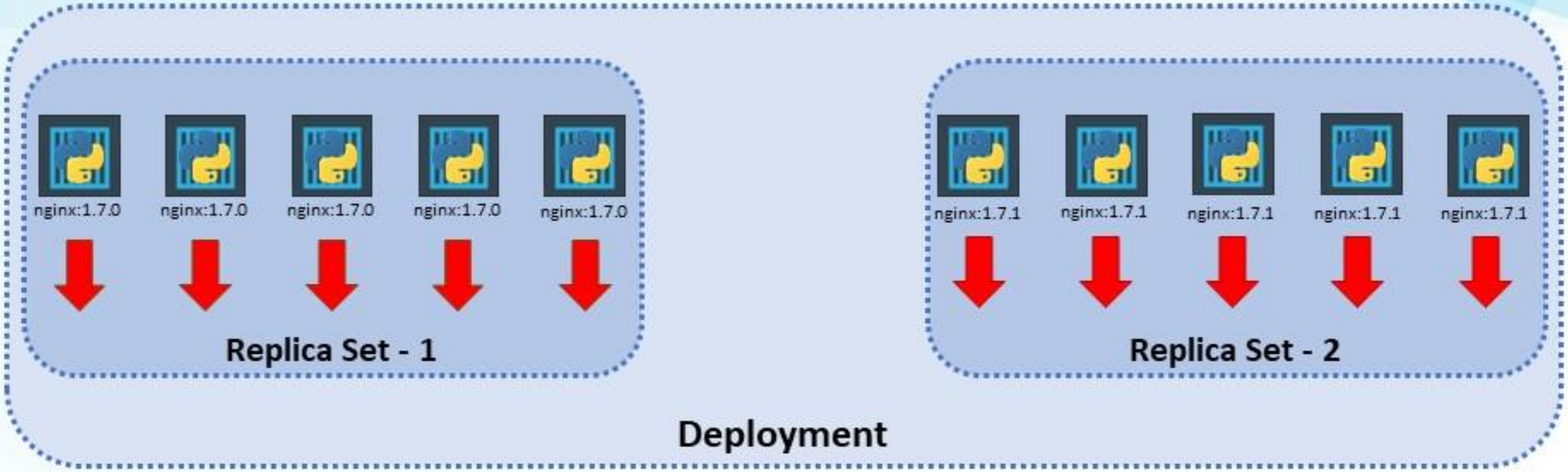
Deployment - Upgrades

- This can be seen when you try to list the replica sets



```
> kubectl get replicasets
NAME                          DESIRED   CURRENT   READY   AGE
myapp-deployment-67c749c58c   0         0         0       22m
myapp-deployment-7d57dbdb8d   5         5         5       20m
```

Deployment - Rollback



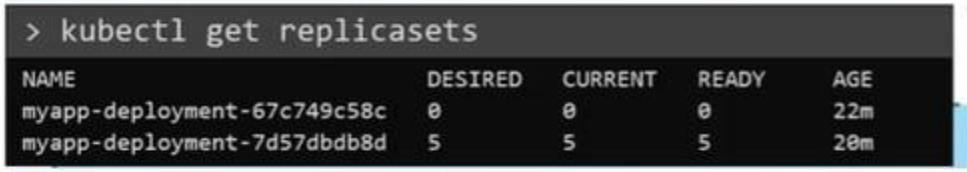
- After upgrade if you realize something is wrong with the new version of build, we can simply rollback
- Kubernetes deployments allows to roll back to a previous revision
- To undo a change, run

```
kubectl rollout undo deployment/myapp-deployment
```

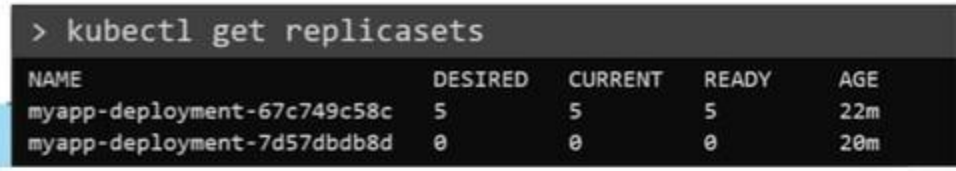
- The deployment will then destroy the pod in the new replica set and bring the older ones up in the old replica set.
- The application will be back to its older format

Deployment - Upgrades

- When we compare the output of the **kubectl get replicasets** command before and after to roll back. We notice the difference



```
> kubectl get replicasets
NAME                           DESIRED   CURRENT   READY   AGE
myapp-deployment-67c749c58c    0         0         0       22m
myapp-deployment-7d57dbdb8d    5         5         5       20m
```



```
> kubectl get replicasets
NAME                           DESIRED   CURRENT   READY   AGE
myapp-deployment-67c749c58c    5         5         5       22m
myapp-deployment-7d57dbdb8d    0         0         0       20m
```

- Check Rollout status

```
kubectl rollout status deployment/myapp-deployment
```

- Check Rollout history

```
kubectl rollout history deployment/myapp-deployment
```

SkillAssure

Summarize Commands

Create

```
> kubectl create –f deployment-definition.yml
```

Get

```
> kubectl get deployments
```

Update

```
> kubectl apply –f deployment-definition.yml
```

```
> kubectl set image deployment/myapp-deployment nginx=nginx:1.9.1
```

Status

```
> kubectl rollout status deployment/myapp-deployment
```

```
> kubectl rollout history deployment/myapp-deployment
```

Rollback

```
> kubectl rollout undo deployment/myapp-deployment
```

Thank You