Why do you need deployment?

If all that a `deployment` does is create a `replicaset`, why cant we just create `rs`?

Experiment

Update the `image` to be `nginx2` (an image that doesnt exist)

```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: nginx2:latest
        ports:
        - containerPort: 80
```

- Apply the new deployment

- Check the new `rs` now


```
kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-576c6b7b6    3         3         3       14m
nginx-deployment-5fbd4799cb   1         1         0       10m
```

- Check the pods

```
kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
nginx-deployment-576c6b7b6-9nlnq    1/1     Running            0          15m
nginx-deployment-576c6b7b6-m8ttl    1/1     Running            0          16m
nginx-deployment-576c6b7b6-n9cx4    1/1     Running            0          16m
nginx-deployment-5fbd4799cb-fmt4f   0/1     ImagePullBackOff   0          12m
```

Role of deployment

Deployment ensures that there is a smooth deployment, and if the new image fails for some reason, the old replicaset is maintained.

Even though the `rs` is what does `pod management` , `deployment` is what does `rs management`

Rollbacks

- Check the history of deployment

```
kubectl rollout history deployment/nginx-deployment
```

- Undo the last deployment

```
kubectl rollout undo deployment/nginx-deployment
```

C

- Replace the image to be `postgres`

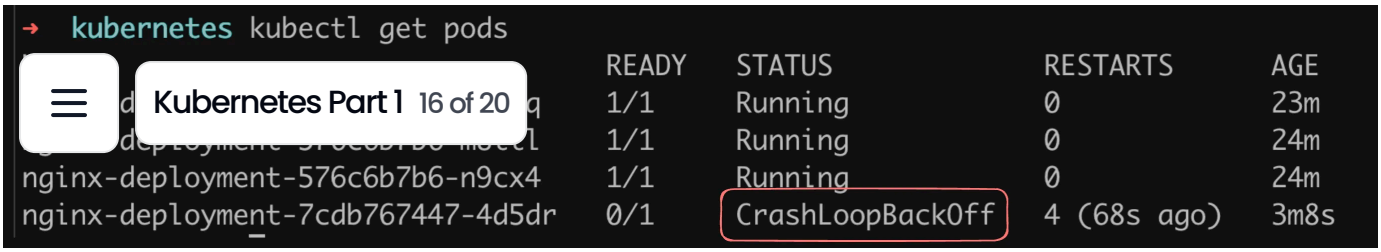
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: postgres:latest
        ports:
     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

- Check the new set or rs

```
→  kubernetes kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-576c6b7b6    3         3         3       21m
nginx-deployment-5fbd4799cb   0         0         0       17m
nginx-deployment-7cdb767447   1         1         0       4s
→  kubernetes kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
nginx-deployment-576c6b7b6-9nlnq    1/1     Running             0          20m
nginx-deployment-576c6b7b6-m8ttl    1/1     Running             0          21m
nginx-deployment-576c6b7b6-n9cx4    1/1     Running             0          21m
nginx-deployment-7cdb767447-4d5dr   0/1     ContainerCreating   0          7s
```

- Check the pods

```
→  kubernetes kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
nginx-deployment-576c6b7b6-9nlnq    1/1     Running             0          20m
nginx-deployment-576c6b7b6-m8ttl    1/1     Running             0          21m
nginx-deployment-576c6b7b6-n9cx4    1/1     Running             0          21m
nginx-deployment-7cdb767447-4d5dr   0/1     ContainerCreating   0          7s
```



- Check the logs

```
kubectl logs -f nginx-deployment-7cdb767447-4d5dr
Error: Database is uninitialized and superuser password is not specified.
    You must specify POSTGRES_PASSWORD to a non-empty value for the
    superuser. For example, "-e POSTGRES_PASSWORD=password" on "docker r

    You may also use "POSTGRES_HOST_AUTH_METHOD=trust" to allow all
    connections without a password. This is *not* recommended.

    See PostgreSQL documentation about "trust":
    https://www.postgresql.org/docs/current/auth-trust.html
```

- Update the manifest to pass `POSTGRES_PASSWORD`

```
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: postgres:latest
        ports:
        - containerPort: 80
        env:
```

- Check pods now


```
→  kubernetes kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
nginx-deployment-576c6b7b6-9nlnq    1/1     Running             0          25m
nginx-deployment-576c6b7b6-m8ttl    1/1     Running             0          26m
nginx-deployment-576c6b7b6-n9cx4    1/1     Running             0          26m
nginx-deployment-58ff5599d8-hq9r2   0/1     ContainerCreating   0          14s
```

- Try after some time

```
→  kubernetes kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-58ff5599d8-hq9r2   1/1     Running   0          44s
nginx-deployment-58ff5599d8-n4zxh   1/1     Running   0          25s
nginx-deployment-58ff5599d8-vbpv9   1/1     Running   0          22s
```

Postgres is running correctly

- Check the rs

```
→  kubernetes kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-576c6b7b6    0         0         0       26m
nginx-deployment-58ff5599d8   3         3         3       57s
nginx-deployment-5fbd4799cb   0         0         0       22m
nginx-deployment-7cdb767447   0         0         0       5m38s
```