1-

```
controlplane $ kubectl create namespace sprints-devops
namespace/sprints-devops created
controlplane $ kubectl get ns
NAME              STATUS   AGE
default           Active   13d
kube-node-lease   Active   13d
kube-public       Active   13d
kube-system       Active   13d
sprints-devops    Active   7s
```

2-

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: sprints-sa-devops
  namespace: sprints-devops
```

```
controlplane $ vim sa.yml
controlplane $ kubectl create -f sa.yml
serviceaccount/sprints-sa-devops created
```

3-

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-role-devops
rules:
- apiGroups: [""]
  #
  # at the HTTP level, the name of the resource for accessing Secret
  # objects is "secrets"
  resources: ["secrets", "configMaps", "secrets", "endpoints", "nodes","pods", "services", "namespaces", "events", "serviceAccount"]
  verbs: ["get", "watch", "list","create","patch","update"]
```

```
controlplane $ kubectl create -f clusterrole.yml
clusterrole.rbac.authorization.k8s.io/cluster-role-devops created
controlplane $
```

4-

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-role-binding-devops
  namespace: sprints-devops
subjects:
- kind: ServiceAccount
  name: sprints-sa-devops
  namespace: sprints-devops
#  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-role-devops
  apiGroup: rbac.authorization.k8s.io
```

```
controlplane $ kubectl create -f crb.yml
clusterrolebinding.rbac.authorization.k8s.io/cluster-role-binding-devops created
```

5-

ASPECT	DEPLOYMENT	STATEFULSET
Data persistence	Stateless	Stateful
Pod name and identity	Pods are assigned an ID that consists of the deployment name and a random hash to generate a temporarily unique identity	Each pod gets a persistent identity consisting of the StatefulSet name and a sequence number
Interchangeability	Pods are identical and can be interchanged	Pods in a StatefulSet are neither identical nor interchangeable
Behavior	A pod can be replaced by a new replica at any time	Pods retain their identity when rescheduled on another node
Volume claim	All replicas share a PVC and a volume	Each pod gets a unique volume and PVC
Pod interaction	Requires a service to interact with the pods	The headless service handles pod network identities
Order of pod creation	Pods are created and deleted randomly	Pods are created in a strict sequence and cannot be deleted randomly

6-

```
controlplane $ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.5.1/deploy/static/provider/cloud/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
```

```
controlplane $ kubectl get pods --namespace=ingress-nginx
NAME                                        READY   STATUS      RESTARTS   AGE
ingress-nginx-admission-create-k8d4g        0/1     Completed   0          103s
ingress-nginx-admission-patch-tf6lc         0/1     Completed   0          103s
ingress-nginx-controller-65dc77f88f-6xv2h   1/1     Running     0          103s
controlplane $ kubectl create deployment web --image=gcr.io/google-samples/hello-app:1.0
deployment.apps/web created
controlplane $ kubectl expose deployment web --type=NodePort --port=8080
service/web exposed
controlplane $ kubectl get service web
NAME   TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
web    NodePort   10.99.146.245   <none>        8080:32575/TCP   7s
```