Commands

```
kubectl get nodes
```

```
minikube status
```

```
kubectl version
```

```
kubectl get pod
```

```
kubectl get pod -o wide
```

```
kubectl get services
```

```
kubectl create deployment nginx-depl --image=nginx
```

```
kubectl create deployment nginx-depl --image=nginx -o yaml
```

```
kubectl get deployment
```

```
kubectl get replicaset
```

```
kubectl edit deployment nginx-depl
```

```
kubectl logs nginx-depl-$-$
```

```
kubectl create deployment mongo-depl --image=mongo
```

```
kubectl describe pod [pod-name]
```

```
kubectl describe service [service-name]
```

```
kubectl exec -it [pod name] -- bin/bash
```

```
kubectl delete deployment mongo-depl
```

```
kubectl delete -f nginx-deployment.yaml
```

```
kubectl get deployment nginx-deployment -o yaml
```

```
kubectl get pod/[pod-name] -n [namespace-name] -o yaml
```

```
kubectl scale deployment/[deploymen-name] --replicas=3
```

```
kubectl rollout status deployment [deploymen-name]
```

```
kubectl rollout undo deployment [deploymen-name]
```

```
kubectl apply -f [file-name] --namespace=[namespace-name]
```

```
kubectl get namesapces
```

```
kubectl create namesapce [namespace-name]
```

```
kubectl cluster-info
```

(automatically starts the k8s Nginx implementation of Ingress Controller)
(After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1")

```
minikube addons enable ingress
minikube tunnel (mongo-express starts at localhost:8081)
kubectl get pod -n kube-system
```

```
kubectl get ingress -n [name-space]
```

(make secret before deployment references it)

```
kubectl get secrets
```

```
kubectl get all
```

```
kubectl top
```

(assign a external IP using minikube, else it will be pending)

```
minikube service mongo-express-service
```

Parts of a K8s YAML configuration

1. metadata
2. specification
3. status (automatically generated and added by kubernetes) → K8s tries to make the actual state equal to the desired state, this is the basis of the self-healing that K8s provides.

nginx-deployment-result.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"labels":
{"app":"nginx"},"name":"nginx-deployment","namespace":"default"},"spec":{"replicas":2,"selector":{"matchLabels":
{"app":"nginx"}},"template":{"metadata":{"labels":{"app":"nginx"}},"spec":{"containers":
[{"image":"nginx:1.16","name":"nginx","ports":[{"containerPort":8080}]}]}}}}
  creationTimestamp: "2020-01-24T10:54:56Z"
  generation: 1
  labels:
    app: nginx
  name: nginx-deployment
  namespace: default
  resourceVersion: "96574"
  selfLink: /apis/apps/v1/namespaces/default/deployments/nginx-deployment
  uid: e1075fa3-6468-43d0-83c0-63fede0dae51
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
```

```
    spec:
      containers:
      - image: nginx:1.16
        imagePullPolicy: IfNotPresent
        name: nginx
        ports:
        - containerPort: 8080
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: "2020-01-24T10:54:59Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2020-01-24T10:54:56Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: ReplicaSet "nginx-deployment-7d64f4b574" has successfully progressed.
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 1
  readyReplicas: 2
  replicas: 2
  updatedReplicas: 2
```

nginx-deployment.yaml

(the template is the configuration of the pod)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16
        ports:
        - containerPort: 8080
```

```
kubectl apply -f [config-file].yaml
```

nginx-service.yaml

(connections are made with labels and selectors, metadata contains the labels and spec contains the selectors)

(deployment and pods can have their own labels)

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```