

***System Admin***

***Training Assignments***

|  |  |
| --- | --- |
| Program Code |  |
| Issue/Revision | x/y |
| Effective date | 04/Aug /2023 |

**Assignment Day 15: Kubernetes Basics**

**Task 1 - Create a Web Application :**

* Create a Kubernetes Pod that runs a simple web application. You can use any web application image available on Docker Hub.
* Create a Persistent Volume (PV) and a Persistent Volume Claim (PVC) to store the web application data.
* Use a ReplicaSet or Deployment to manage the Pods for the web application.
* Expose the web application using a Kubernetes Service of type NodePort.

**Task 2 - Create a Database**:

* Create a Kubernetes Pod that runs a database of your choice (e.g., MySQL or PostgreSQL) from a public Docker image.
* Create a PVC for the database to store its data persistently.
* Ensure that the database Pod can communicate with the web application Pod.

**Task 3 - Set Up Ingress**:

* Create an Ingress resource that allows external access to your web application.
* Define appropriate rules and paths in the Ingress resource to route traffic to your web application.

**Task 4 - Implement Scaling**:

* Use a ReplicaSet or Deployment for both the web application and the database Pods.
* Demonstrate horizontal scaling by increasing and decreasing the number of replicas for the web application Pods.

**Task 5 - Implement StatefulSet and DaemonSet (optional)**:

* Create a StatefulSet for a component of your choice within your web application (e.g., a caching server).
* Create a DaemonSet for a component of your choice within your database (e.g., a log collector).
* Explain the use cases for StatefulSet and DaemonSet in your scenario.

**Solution**

**Task 1 - Create a Web Application**

*# web-app-pod.yaml*

*apiVersion: v1*

*kind: Pod*

*metadata:*

*name: web-app-pod*

*labels:*

*app: web-app*

*spec:*

*containers:*

*- name: web-app-container*

*image: nginx:latest*

*ports:*

*- containerPort: 80*

*volumes:*

*- name: web-app-data*

*emptyDir: {}*

*# pv.yaml*

*apiVersion: v1*

*kind: PersistentVolume*

*metadata:*

*name: web-app-pv*

*spec:*

*capacity:*

*storage: 1Gi*

*accessModes:*

*- ReadWriteOnce*

*persistentVolumeReclaimPolicy: Retain*

*storageClassName: manual*

*# pvc.yaml*

*apiVersion: v1*

*kind: PersistentVolumeClaim*

*metadata:*

*name: web-app-pvc*

*spec:*

*accessModes:*

*- ReadWriteOnce*

*resources:*

*requests:*

*storage: 1Gi*

*storageClassName: manual*

*# web-app-service.yaml*

*apiVersion: v1*

*kind: Service*

*metadata:*

*name: web-app-service*

*spec:*

*selector:*

*app: web-app*

*ports:*

*- protocol: TCP*

*port: 80*

*targetPort: 80*

*type: NodePort*

**Task 2 - Create a Database:**

*# db-pod.yaml*

*apiVersion: v1*

*kind: Pod*

*metadata:*

*name: db-pod*

*labels:*

*app: database*

*spec:*

*containers:*

*- name: db-container*

*image: mysql:latest*

*env:*

*- name: MYSQL\_ROOT\_PASSWORD*

*value: yourpassword*

*volumes:*

*- name: db-data*

*persistentVolumeClaim:*

*claimName: db-pvc*

*# db-pvc.yaml*

*apiVersion: v1*

*kind: PersistentVolumeClaim*

*metadata:*

*name: db-pvc*

*spec:*

*accessModes:*

*- ReadWriteOnce*

*resources:*

*requests:*

*storage: 1Gi*

*storageClassName: manual*

**Task 3 - Set Up Ingress:**

*# ingress.yaml*

*apiVersion: networking.k8s.io/v1*

*kind: Ingress*

*metadata:*

*name: web-app-ingress*

*spec:*

*rules:*

*- host: yourdomain.com*

*http:*

*paths:*

*- path: /*

*pathType: Prefix*

*backend:*

*service:*

*name: web-app-service*

*port:*

*number: 80*

**Task 4 - Implement Scaling:**

Use **kubectl scale** to scale the web application deployment:

*kubectl scale deployment web-app-deployment --replicas=3*

**Task 5 - Implement StatefulSet and DaemonSet:**

*# statefulset.yaml*

*apiVersion: apps/v1*

*kind: StatefulSet*

*metadata:*

*name: stateful-app*

*spec:*

*serviceName: "stateful-app"*

*replicas: 3*

*selector:*

*matchLabels:*

*app: stateful-app*

*template:*

*metadata:*

*labels:*

*app: stateful-app*

*spec:*

*containers:*

*- name: stateful-container*

*image: your-stateful-image:latest*

*# daemonset.yaml*

*apiVersion: apps/v1*

*kind: DaemonSet*

*metadata:*

*name: daemonset-app*

*spec:*

*selector:*

*matchLabels:*

*app: daemonset-app*

*template:*

*metadata:*

*labels:*

*app: daemonset-app*

*spec:*

*containers:*

*- name: daemonset-container*

*image: your-daemonset-image:latest*