DEVESH CHOUDHARY

DevOps & Fullstack Enthusiast

Collection of Kubernetes Manifest file

pod.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: <pod-name>           # Replace with your desired pod name
  labels:                    # Optional labels for filtering and RBAC
    app: <your-app-name>   # Add your labels here (e.g., app: my-app)
spec:
  containers:
     - name: <container-name>
       image: <image-name>:<tag>        # Replace with the image name
       ports:                       # Optional: Define ports exposed by the container
         - containerPort: <port-number>
           name: <port-name>           # Optional: Define a name for the port
       env:                  # Optional: Define environment variables for the container
         - name: <env-var-name>
           value: "<env-var-value>"
       resources: # Optional: Define resource requests and limits for the container
           requests:
              memory: "1Gi"
              cpu: "1"
           limits:
              memory: "2Gi"
              cpu: "2"
       volumeMounts:            # Optional: Define volume mounts for the container
         - name: <volume-name>
           mountPath: /<path/in/container>
  volumes:                # Optional: Define volumes for the pod (persistent storage)
     - name: <volume-name>
       persistentVolumeClaim:
           laimName: <your-claim-name>
```

deployment.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: <deployment-name>
  labels: null
spec:
  replicas: 2
  selector:
    matchLabels:
      app: <pod-name>
  template:
    metadata:
      labels:
        app: <pod-name>
    spec:
      containers:
        - name: <container-name>
          image: '<image-name>:<tag>'
          ports:
            - containerPort: <port-number>
              name: <port-name>
          env:
            - name: <env-var-name>
              value: <env-var-value>
          livenessProbe:
            httpGet:
              path: /
              port: <port-number>
            initialDelaySeconds: 15
            periodSeconds: 20
            failureThreshold: 3
```

deployment.yml (continue)

```yaml
      readinessProbe:
        httpGet:
          path: /
          port: <port-number>
        initialDelaySeconds: 5
        periodSeconds: 10
        failureThreshold: 2
      resources:  #Optional: Define resource requests and limits
        requests:
          memory: 1Gi
          cpu: '1'
        limits:
          memory: 2Gi
          cpu: '2'
      volumeMounts:
         - name: <volume-name>
           mountPath: /<path/in/container>
  volumes:
    - name: <volume-name>
      persistentVolumeClaim:
        claimName: <pvc-name>
```

service.yml

```
# This template defines a Kubernetes Service for exposing
your application


apiVersion: v1
kind: Service
metadata:
  name: <service-name>
spec:
  selector:
    app:  <your-app-name>  #Replace with pod selection criteria
  ports:
    - protocol: TCP # Protocol (typically TCP)
      port: <service-port>       #External service expose port
      targetPort: <container-port> #Pod listning port
  type: <service-type>    #ClusterIP, NodePort, LoadBalancer
```

Config-map.yml & Secrets.yml

```
# This template defines a ConfigMap for storing non-sensitive
configuration data

apiVersion: v1
kind: ConfigMap
metadata:
  name: <configmap-name> #Descriptive name
data:
  <key1>: <value1>
  <key2>: <value2>
  # ... (add more key-value pairs as needed)


# This template defines a Secret for storing sensitive data in
Kubernetes

apiVersion: v1
kind: Secret
metadata:
  name: <secret-name> #Descriptive name for your secret
stringData:
  # Replace with key-value pairs for your sensitive data
  <key1>: <value1> (base64 encoded)
  <key2>: <value2> (base64 encoded)
  # ... (add more key-value pairs as needed)
```

Service-account & Role.yml

# This template defines a ServiceAccount for your application with Namespaces limited

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: <service-account-name>          #Descriptive service-account
  namespace: <target-namespace>         #Replace your name-space
```

# This template defines a Role for granting least privilege access within the cluster

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
   name: <role-name>          # Replace with a descriptive role name
rules:
   - apiGroups: ["apps"]             # (adjust based on your needs)
     resources: ["deployments","”pods”] # (adjust based on your needs)
     verbs: ["get", "list", "watch"] #(adjust based on your needs)

  # We can add additional rules for specific access control needs,
following least privilege
```

Role-binding.yml (with service-account)

```
# This template defines a RoleBinding to associate
a Namespaced Service Account with a Role

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
   name: <rolebinding-name>      #Descriptive name
subjects:
   - kind: ServiceAccount
     name: <service-account-name> #service account Ref
     namespace: <target-namespace> #service-account
roleRef:
   apiGroup: rbac.authorization.k8s.io
   kind: Role
   name: <role-name> # Reference the role
```

Persistance-volume.yml & PV-claim.yml

```
# This template defines a Persistent Volume for production
apiVersion: v1
kind: PersistentVolume
metadata:
  name: <pv-name>    #descriptive name for your PV
spec:
  capacity:
    storage: 1Gi        # Desired storage capacity
  accessModes:
    - ReadWriteOnce # Access mode
  persistentVolumeReclaimPolicy: Retain         #Retain storage even after PV becomes unbound
```

```
# This template defines a Persistent Volume Claim to request storage from a StorageClass or hostpath volume.
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
    name: my-pvc # Name of the Persistent Volume Claim
spec:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
          storage: 1Gi # Requested storage
```

HorizontalPodAutoscaler.yml

```
# This template defines a Horizontal Pod Autoscaler (HPA) for
production use

apiVersion: apps/v1
kind: HorizontalPodAutoscaler
metadata:
  name: <hpa-name>          # Replace with a name for your HPA
spec:
  scaleTargetRef:
    apiVersion: apps/v1       # Adjust for ReplicaSets
    kind: Deployment          # Adjust for ReplicaSets (ReplicaSet)
    name: <target-deployment-name>   #Actual deployment name
  minReplicas: <min-replicas>        #Minimum desired pod count
  maxReplicas: <max-replicas>       #Maximum desired pod count

  metrics:             # Define metrics for autoscaling
    - type: Resource
      resource:
        name: CPU
        target:
          type: Utilization
          averageUtilization: <target-cpu-utilization> # Replace with
desired average CPU utilization (e.g., 80)

    # You can add additional metrics here for combined scaling
decisions (optional)
    # - type: ...
```

VerticalPodAutoscaler.yml

```yaml
# This template defines a Vertical Pod Autoscaler (VPA) for production use

apiVersion: autoscaling.k8s.io/v2beta2
kind: VerticalPodAutoscaler
metadata:
  name: <vpa-name>      #Descriptive name for your VPA
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment            #Adjust for ReplicaSets
    name: <target-deployment-name>   #Replace name
  updatePolicy:
    updateMode: Auto            #Recommended for production
  metrics:   #monitor for autoscaling
    - type: Resource
      resource:
        name: memory
        target:
          type: Utilization
          averageUtilization: <target-memory-utilization>
  minResources:
    memory: <minimum-memory-request>
  resources:
    memory: <initial-memory-request> # Set memory request
```

ClusterRoleBinding (Service-acc, Cluster-role)

#Creating a Service Account:

```
kubectl create serviceaccount my-service-account
```

#Creating a ClusterRole:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <my-cluster-role> #Replace as your choice
rules:
  - apiGroups: [""]
    resources: ["pods"]         #We can add more services
    verbs: ["get", "list", "watch"]
```

# This template defines a (restricted) ClusterRoleBinding for a Service Account with limited ClusterRole access.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
   name: <clusterrolebinding-name> # Replace with a descriptive name.
subjects:
   - kind: ServiceAccount
     name: <my-service-account> # Reference the service account.
roleRef:
   apiGroup: rbac.authorization.k8s.io
   kind: ClusterRole
   name: <my-cluster-role> # Reference the ClusterRole
```

UserRoleBinding (Service-acc, Cluster-role)

```
# This RoleBinding grants permissions to a User within a
namespace.

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding    # Kind of object (in this case, a RoleBinding)
metadata:
  name: <role-binding-name>  # (replace with your desired name)
  namespace: <namespace>   #(replace with the namespace)

subjects:
 # Who is granted the permissions (a User)
  - kind: User           # Kind of subject (User in this case)
    name: <user-name>  # Username (replace with the actual one)
    apiGroup: rbac.authorization.k8s.io     # API group of the User

roleRef:
 # Reference to the Role that defines the permissions
  apiGroup: rbac.authorization.k8s.io     # API group of the Role
  kind: Role                    # Kind of object (Role in this case)
  name: <role-name>             # Name of the Role
```

YOU'VE SUCCESSFULLY NAVIGATED THROUGH A COLLECTION OF KUBERNETES MANIFESTS. THIS IS A SIGNIFICANT STEP IN UNDERSTANDING THE BUILDING BLOCKS OF KUBERNETES APPLICATIONS.

If You liked the post, you can follow me below

Thank you