Kubernetes Project (v1.)

1.

```
minikube start --nodes=5
```

2.

```
kubectl create namespace test

kubectl create namespace production
```

3.

Öncelikle işe, .yaml dosyalarını oluşturmakla başlayalım.

jr-production-rb.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
# Kubernetes API versiyonunu belirtir.
kind: RoleBinding
# Bu YAML belgesinde tanımlanan nesnenin türünü belirtir, burada bir RoleBinding (Rol Atama) tanımlanmıştır.
metadata:
  name: jr-production-rb
  # RoleBinding'in adı "jr-production-rb" olarak belirlenmiştir.
  namespace: production
  # RoleBinding'in ait olduğu namespace "production" olarak belirlenmiştir.
subjects:
- kind: Group
  # Atanan yetkinin türü "Group" olarak belirtilmiştir.
  name: junior
  # "Junior" adlı bir grup, yani kullanıcı grubu belirlenmiştir.
  apiGroup: rbac.authorization.k8s.io
  # Yetki verilen grubun API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
roleRef:
  kind: ClusterRole
  # Atanan Role türü "ClusterRole" olarak belirtilmiştir.
  name: view
  # Verilen yetki "view" adlı bir Role ile sınırlıdır.
  apiGroup: rbac.authorization.k8s.io
  # Role'un API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
```

jr-test-rb.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
# Kubernetes API versiyonunu belirtir.
kind: RoleBinding
# Bu YAML belgesinde tanımlanan nesnenin türünü belirtir, burada bir RoleBinding (Rol Atama) tanımlanmıştır.
metadata:
  name: jr-test-rb
  # RoleBinding'in adı "jr-test-rb" olarak belirlenmiştir.
  namespace: test
  # RoleBinding'in ait olduğu namespace "test" olarak belirlenmiştir.
subjects:
- kind: Group
  # Atanan yetkinin türü "Group" olarak belirtilmiştir.
  name: junior
  # "Junior" adlı bir grup, yani kullanıcı grubu belirlenmiştir.
```

```
  apiGroup: rbac.authorization.k8s.io
  # Yetki verilen grubun API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
roleRef:
  kind: ClusterRole
  # Atanan Role türü "ClusterRole" olarak belirtilmiştir.
  name: edit
  # Verilen yetki "edit" adlı bir ClusterRole ile sınırlıdır.
  apiGroup: rbac.authorization.k8s.io
  # Role'un API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
```

sr-production-rb.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
# Kubernetes API versiyonunu belirtir.
kind: RoleBinding
# Bu YAML belgesinde tanımlanan nesnenin türünü belirtir, burada bir RoleBinding (Rol Atama) tanımlanmıştır.
metadata:
  name: sr-production-rb
  # RoleBinding'in adı "sr-production-rb" olarak belirlenmiştir.
  namespace: production
  # RoleBinding'in ait olduğu namespace "production" olarak belirlenmiştir.
subjects:
- kind: Group
  # Atanan yetkinin türü "Group" olarak belirtilmiştir.
  name: senior
  # "Senior" adlı bir grup, yani kullanıcı grubu belirlenmiştir.
  apiGroup: rbac.authorization.k8s.io
  # Yetki verilen grubun API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
roleRef:
  kind: ClusterRole
  # Atanan Role türü "ClusterRole" olarak belirtilmiştir.
  name: edit
  # Verilen yetki "edit" adlı bir ClusterRole ile sınırlıdır.
  apiGroup: rbac.authorization.k8s.io
  # Role'un API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
```

sr-test-rb.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
# Kubernetes API sürümünü belirtir.
kind: RoleBinding
# Bu YAML belgesinde tanımlanan nesnenin türünü belirtir, burada bir RoleBinding (Rol Atama) tanımlanmıştır.
metadata:
  name: sr-test-rb
  # RoleBinding'in adı "sr-test-rb" olarak belirlenmiştir.
  namespace: test
  # RoleBinding'in ait olduğu namespace "test" olarak belirlenmiştir.
subjects:
- kind: Group
  # Atanan yetkinin türü "Group" olarak belirtilmiştir.
  name: senior
  # "Senior" adlı bir grup, yani kullanıcı grubu belirlenmiştir.
  apiGroup: rbac.authorization.k8s.io
  # Yetki verilen grubun API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
roleRef:
  kind: ClusterRole
  # Atanan Role türü "ClusterRole" olarak belirtilmiştir.
  name: edit
  # Verilen yetki "edit" adlı bir ClusterRole ile sınırlıdır.
  apiGroup: rbac.authorization.k8s.io
  # Role'un API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
```

sr-cluster-crb.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
# Kubernetes API sürümünü belirtir.
```

```
kind: ClusterRoleBinding
# Bu YAML belgesinde tanımlanan nesnenin türünü belirtir, burada bir ClusterRoleBinding (Küme Rol Atama) tanımlanmıştır.
metadata:
  name: sr-cluster-crb
  # ClusterRoleBinding'in adı "sr-cluster-crb" olarak belirlenmiştir.
subjects:
- kind: Group
  # Atanan yetkinin türü "Group" olarak belirtilmiştir.
  name: senior
  # "Senior" adlı bir grup, yani kullanıcı grubu belirlenmiştir.
  apiGroup: rbac.authorization.k8s.io
  # Yetki verilen grubun API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
roleRef:
  kind: ClusterRole
  # Atanan Role türü "ClusterRole" olarak belirtilmiştir.
  name: view
  # Verilen yetki "view" adlı bir ClusterRole ile sınırlıdır.
  apiGroup: rbac.authorization.k8s.io
  # Role'un API grubu "rbac.authorization.k8s.io" olarak belirtilmiştir.
```

apply

```
kubectl apply -f ./yaml/jr-production-rb.yaml

kubectl apply -f ./yaml/jr-test-rb.yaml

kubectl apply -f ./yaml/sr-cluster-crb.yaml

kubectl apply -f ./yaml/sr-production-rb.yaml

kubectl apply -f ./yaml/sr-test-rb.yaml
```

4.

Nginx: https://kubernetes.github.io/ingress-nginx/deploy/

```
minikube addons enable ingress
```

```
# OUTPUT
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image registry.k8s.io/ingress-nginx/controller:v1.7.0
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230312-helm-chart-4.5.2-28-g66a760794
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230312-helm-chart-4.5.2-28-g66a760794
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

5.

```
node1=$(kubectl get no -o jsonpath="{.items[1].metadata.name}")
```

Bu komut, `kubectl get nodes` komutunun çıktısından sırasıyla ikinci düğümün adını alarak "node1" adlı bir değişkene atar.

```
node2=$(kubectl get no -o jsonpath="{.items[2].metadata.name}")
```

Bu komut, `kubectl get nodes` komutunun çıktısından sırasıyla üçüncü düğümün adını alarak "node2" adlı bir değişkene atar.

```
node3=$(kubectl get no -o jsonpath="{.items[3].metadata.name}")
```

Bu komut, `kubectl get nodes` komutunun çıktısından sırasıyla dördüncü düğümün adını alarak "node3" adlı bir değişkene atar.

```
kubectl taint node $node1 tier=production:NoSchedule
```

Bu komut, "node1" adlı düğümü "tier=production:NoSchedule" şeklinde bir "NoSchedule" taint ile işaretler. Bu, düğümün üzerinde çalışacak Pod'ların belirli taintleri olan Pod'lara zaman zaman zamanlama yapmasını engelleyecektir.

```
kubectl taint node $node2 tier=production:NoSchedule
```

Bu komut, "node2" adlı düğümü "tier=production:NoSchedule" şeklinde bir "NoSchedule" taint ile işaretler. Bu, düğümün üzerinde çalışacak Pod'ların belirli taintleri olan Pod'lara zaman zaman zamanlama yapmasını engelleyecektir.

```
kubectl taint node $node3 tier=production:NoSchedule
```

Bu komut, "node3" adlı düğümü "tier=production:NoSchedule" şeklinde bir "NoSchedule" taint ile işaretler. Bu, düğümün üzerinde çalışacak Pod'ların belirli taintleri olan Pod'lara zaman zaman zamanlama yapmasını engelleyecektir.

```
kubectl label node $node1 tier=production
```

Bu komut, "node1" adlı düğüme "tier=production" şeklinde "production" etiketi ekler. Bu, düğümü "production" ortamı için ayrılmış olarak işaretlemeye yardımcı olur.

```
kubectl label node $node2 tier=production
```

Bu komut, "node2" adlı düğüme "tier=production" şeklinde "production" etiketi ekler. Bu, düğümü "production" ortamı için ayrılmış olarak işaretlemeye yardımcı olur.

```
kubectl label node $node3 tier=production
```

Bu komut, "node3" adlı düğüme "tier=production" şeklinde "production" etiketi ekler. Bu, düğümü "production" ortamı için ayrılmış olarak işaretlemeye yardımcı olur.

6.

MYSQL_USER.txt

```
mysqlwpadmin
```

MYSQL_PASSWORD.txt

```
P@ssw0rd1!
```

MYSQL_ROOT_PASSWORD.txt

```
P@ssw0rd1!
```

MYSQL_DATABASE.txt

```
wordpressdb
```

wptest.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-test-claim
  namespace: test
  labels:
    tier: test
    app: mysql
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 5Gi
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-test-claim
  namespace: test
  labels:
    tier: test
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 5Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  namespace: test
  labels:
    tier: test
    app: mysql
spec:
  selector:
    matchLabels:
      tier: test
```

```
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        tier: test
        app: mysql
    spec:
      containers:
      - name: mysql-test
        image: mysql:5.6
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-test-secret
              key: MYSQL_ROOT_PASSWORD
        - name: MYSQL_USER
          valueFrom:
            secretKeyRef:
              name: mysql-test-secret
              key: MYSQL_USER
        - name: MYSQL_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-test-secret
              key: MYSQL_PASSWORD
        - name: MYSQL_DATABASE
          valueFrom:
            secretKeyRef:
              name: mysql-test-secret
              key: MYSQL_DATABASE
        resources:
          limits:
            memory: "1Gi"
            cpu: "250m"
        ports:
          - containerPort: 3306
            name: mysql-test-port
        volumeMounts:
          - name: mysql-test-pv
            mountPath: /var/lib/mysql
      volumes:
        - name: mysql-test-pv
          persistentVolumeClaim:
            claimName: mysql-test-claim
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wp-deployment
  namespace: test
  labels:
    tier: test
    app: wordpress
spec:
  selector:
    matchLabels:
      tier: test
      app: wordpress
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        tier: test
        app: wordpress
    spec:
      containers:
      - image: wordpress:5.6
        name: wp-test
        env:
        - name: WORDPRESS_DB_HOST
          value: mysql-test-svc
        - name: WORDPRESS_DB_USER
          valueFrom:
            secretKeyRef:
              name: mysql-test-secret
              key: MYSQL_USER
        - name: WORDPRESS_DB_PASSWORD
```

```
          valueFrom:
            secretKeyRef:
              name: mysql-test-secret
              key: MYSQL_PASSWORD
        - name: WORDPRESS_DB_NAME
          valueFrom:
            secretKeyRef:
              name: mysql-test-secret
              key: MYSQL_DATABASE
        resources:
          limits:
            memory: "1Gi"
            cpu: "250m"
        ports:
        - containerPort: 80
          name: wp-test-port
        volumeMounts:
        - name: wp-test-pv
          mountPath: /var/www/html
      volumes:
      - name: wp-test-pv
        persistentVolumeClaim:
          claimName: wp-test-claim
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - mysql
            topologyKey: kubernetes.io/hostname
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-test-svc
  namespace: test
  labels:
    tier: test
spec:
  selector:
    tier: test
    app: mysql
  ports:
  - port: 3306
    targetPort: 3306
---
apiVersion: v1
kind: Service
metadata:
  name: wp-test-svc
  namespace: test
  labels:
    tier: test
    app: wordpress
spec:
  selector:
    tier: test
    app: wordpress
  ports:
  - port: 80
    targetPort: 80
```

wptest.yaml açıklama

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-test-claim
  namespace: test
  labels:
    tier: test
    app: mysql
spec:
```

```
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 5Gi
```

Bu kısım, "mysql-test-claim" adında bir kalıcı disk talebi oluşturur. Bu talep, "test" adlı namespace içinde "mysql" uygulaması için 5GB boyutunda bir depolama alanı talep eder.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-test-claim
  namespace: test
  labels:
    tier: test
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 5Gi
```

Bu kısım, "wp-test-claim" adında bir kalıcı disk talebi oluşturur. Bu talep, "test" adlı namespace içinde "wordpress" uygulaması için 5GB boyutunda bir depolama alanı talep eder.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  namespace: test
  labels:
    tier: test
    app: mysql
spec:
  ...
```

Bu kısım, "mysql" uygulamasını "test" adlı namespace içinde çalıştıran bir Deployment (Dağıtım) oluşturur. MySQL 5.6 resmi Docker imajını kullanarak bir konteyner başlatır ve belirli ortam değişkenlerini gizli (secret) nesnelerinden alır. Ayrıca 1GB RAM ve 250m CPU sınırlaması ile düğümde çalışır.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wp-deployment
  namespace: test
  labels:
    tier: test
    app: wordpress
spec:
  ...
```

Bu kısım, "wordpress" uygulamasını "test" adlı namespace içinde çalıştıran bir Deployment (Dağıtım) oluşturur. WordPress 5.6 resmi Docker imajını kullanarak bir konteyner başlatır ve belirli ortam değişkenlerini gizli (secret) nesnelerinden alır. Ayrıca 1GB RAM ve 250m CPU sınırlaması ile düğümde çalışır. Düğümlerde "mysql" etiketine sahip Pod'larla aynı düğümde çalışması için podAffinity kullanır.

```
apiVersion: v1
kind: Service
metadata:
  name: mysql-test-svc
  namespace: test
  labels:
    tier: test
spec:
  ...
```

Bu kısım, "mysql" uygulamasını temsil eden bir Service (Hizmet) oluşturur. Bu hizmet, "test" adlı namespace içindeki "mysql" Pod'larına erişim sağlamak için 3306 numaralı portu kullanır.

```
apiVersion: v1
kind: Service
metadata:
  name: wp-test-svc
  namespace: test
  labels:
    tier: test
spec:
  ...
```

Bu kısım, "wordpress" uygulamasını temsil eden bir Service (Hizmet) oluşturur. Bu hizmet, "test" adlı namespace içindeki "wordpress" Pod'larına erişim sağlamak için 80 numaralı portu kullanır.

wpprod.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-prod-claim
  namespace: production
  labels:
    tier: production
    app: mysql
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 5Gi
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-prod-claim
  namespace: production
  labels:
    tier: production
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 5Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  namespace: production
  labels:
    tier: production
    app: mysql
spec:
```

```
  selector:
    matchLabels:
      tier: production
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        tier: production
        app: mysql
    spec:
      containers:
      - name: mysql-prod
        image: mysql:5.6
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-prod-secret
              key: MYSQL_ROOT_PASSWORD
        - name: MYSQL_USER
          valueFrom:
            secretKeyRef:
              name: mysql-prod-secret
              key: MYSQL_USER
        - name: MYSQL_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-prod-secret
              key: MYSQL_PASSWORD
        - name: MYSQL_DATABASE
          valueFrom:
            secretKeyRef:
              name: mysql-prod-secret
              key: MYSQL_DATABASE
        resources:
          limits:
            memory: "1Gi"
            cpu: "250m"
        ports:
          - containerPort: 3306
            name: mysql-prod-port
        volumeMounts:
          - name: mysql-prod-pv
            mountPath: /var/lib/mysql
      nodeSelector:
        tier: production
      tolerations:
      - key: "tier"
        operator: "Equal"
        value: "production"
        effect: "NoSchedule"
      volumes:
        - name: mysql-prod-pv
          persistentVolumeClaim:
            claimName: mysql-prod-claim
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wp-deployment
  namespace: production
  labels:
    tier: production
    app: wordpress
spec:
  selector:
    matchLabels:
      tier: production
      app: wordpress
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        tier: production
        app: wordpress
    spec:
      containers:
      - image: wordpress:5.6
```

```
        name: wp-prod
        env:
        - name: WORDPRESS_DB_HOST
          value: mysql-prod-svc
        - name: WORDPRESS_DB_USER
          valueFrom:
            secretKeyRef:
              name: mysql-prod-secret
              key: MYSQL_USER
        - name: WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-prod-secret
              key: MYSQL_PASSWORD
        - name: WORDPRESS_DB_NAME
          valueFrom:
            secretKeyRef:
              name: mysql-prod-secret
              key: MYSQL_DATABASE
        resources:
          limits:
            memory: "1Gi"
            cpu: "250m"
        ports:
        - containerPort: 80
          name: wp-prod-port
        volumeMounts:
        - name: wp-prod-pv
          mountPath: /var/www/html
      nodeSelector:
        tier: production
      tolerations:
      - key: "tier"
        operator: "Equal"
        value: "production"
        effect: "NoSchedule"
      volumes:
      - name: wp-prod-pv
        persistentVolumeClaim:
          claimName: wp-prod-claim
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - mysql
            topologyKey: kubernetes.io/hostname
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-prod-svc
  namespace: production
  labels:
    tier: production
spec:
  selector:
    tier: production
    app: mysql
  ports:
  - port: 3306
    targetPort: 3306
---
apiVersion: v1
kind: Service
metadata:
  name: wp-prod-svc
  namespace: production
  labels:
    tier: production
    app: wordpress
spec:
  selector:
    tier: production
    app: wordpress
  ports:
  - port: 80
    targetPort: 80
```

wpprod.yaml açıklama

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-prod-claim
  namespace: production
  labels:
    tier: production
    app: mysql
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 5Gi
```

Bu kısım, "mysql-prod-claim" adında bir kalıcı disk talebi oluşturur. "production" adlı namespace içinde "mysql" uygulaması için 5GB boyutunda bir depolama alanı talep eder.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-prod-claim
  namespace: production
  labels:
    tier: production
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 5Gi
```

Bu kısım, "wp-prod-claim" adında bir kalıcı disk talebi oluşturur. "production" adlı namespace içinde "wordpress" uygulaması için 5GB boyutunda bir depolama alanı talep eder.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  namespace: production
  labels:
    tier: production
    app: mysql
spec:
  ...
```

Bu kısım, "mysql" uygulamasını "production" adlı namespace içinde çalıştıran bir Deployment (Dağıtım) oluşturur. MySQL 5.6 resmi Docker imajını kullanarak bir konteyner başlatır ve belirli ortam değişkenlerini gizli (secret) nesnelerinden alır. Ayrıca 1GB RAM ve 250m CPU sınırlaması ile düğümde çalışır. Bu Deployment, düğümlere "tier: production" etiketi atanmış olanlarda çalışacak ve NoSchedule tolerasyonu ile işaretlenmiş düğümlerde zamanlama yapmayacaktır.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wp-deployment
  namespace: production
  labels:
    tier: production
    app: wordpress
spec:
  ...
```

Bu kısım, "wordpress" uygulamasını "production" adlı namespace içinde çalıştıran bir Deployment (Dağıtım) oluşturur. WordPress 5.6 resmi Docker imajını kullanarak bir konteyner başlatır ve belirli ortam değişkenlerini gizli (secret) nesnelerinden alır. Ayrıca 1GB RAM ve 250m CPU sınırlaması ile düğümde çalışır. Bu Deployment, düğümlere "tier: production" etiketi atanmış olanlarda çalışacak ve NoSchedule tolerasyonu ile işaretlenmiş düğümlerde zamanlama yapmayacaktır. Ayrıca bu Deployment, "mysql" uygulamasını temsil eden Pod'larla aynı düğümde çalışması için podAffinity kullanacaktır.

```
apiVersion: v1
kind: Service
metadata:
  name: mysql-prod-svc
  namespace: production
  labels:
    tier: production
spec:
  ...
```

Bu kısım, "mysql" uygulamasını temsil eden bir Service (Hizmet) oluşturur. Bu hizmet, "production" adlı namespace içindeki "mysql" Pod'larına erişim sağlamak için 3306 numaralı portu kullanacaktır.

```
apiVersion: v1
kind: Service
metadata:
  name: wp-prod-svc
  namespace: production
  labels:
    tier: production
    app: wordpress
spec:
  ...
```

Bu kısım, "wordpress" uygulamasını temsil eden bir Service (Hizmet) oluşturur. Bu hizmet, "production" adlı namespace içindeki "wordpress" Pod'larına erişim sağlamak için 80 numaralı portu kullanacaktır.

```
$ kubectl create secret generic mysql-test-secret -n test --from-file=MYSQL_ROOT_PASSWORD=./yaml/mysql_root_password.txt --from-file=MYSQL_

$ kubectl create secret generic mysql-prod-secret -n production --from-file=MYSQL_ROOT_PASSWORD=./yaml/mysql_root_password.txt --from-file=

$ kubectl apply -f ./yaml/wptest.yaml

$ kubectl apply -f ./yaml/wpprod.yaml
```

7.

wpingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wp-test-ingress
  labels:
    tier: test
    app: wordpress
  namespace: test
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-redirect: "false" # SSL yönlendirmesini devre dışı bırakır.
    nginx.ingress.kubernetes.io/rewrite-target: / # URL yeniden yazma hedefi olarak kök dizini kullanır.
spec:
  rules:
  - host: testblog.example.com # Bu kural, testblog.example.com için geçerlidir.
    http:
      paths:
      - pathType: Prefix
        path: / # Gelen isteklerin hepsi bu yolu takip eder.
        backend:
          service:
            name: wp-test-svc
            port:
              number: 80 # İstekler, wp-test-svc adlı servise 80 numaralı porttan yönlendirilir.
---

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wp-prod-ingress
  labels:
    tier: production
    app: wordpress
  namespace: production
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-redirect: "false" # SSL yönlendirmesini devre dışı bırakır.
    nginx.ingress.kubernetes.io/rewrite-target: / # URL yeniden yazma hedefi olarak kök dizini kullanır.
spec:
  rules:
  - host: companyblog.example.com # Bu kural, companyblog.example.com için geçerlidir.
    http:
      paths:
      - pathType: Prefix
        path: / # Gelen isteklerin hepsi bu yolu takip eder.
        backend:
          service:
            name: wp-prod-svc
            port:
              number: 80 # İstekler, wp-prod-svc adlı servise 80 numaralı porttan yönlendirilir.
```

```
kubectl apply -f ./yaml/wpingress.yaml
```

8.

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-deployment
  labels:
    app: k8s
  namespace: production
spec:
  replicas: 5 # 5 adet kopya (pod) oluşturacak şekilde konfigüre edilmiştir.
```

```
  selector:
    matchLabels:
      app: k8s # Deployment, etiketleri 'app: k8s' olan podları yönetir.
  strategy:
    type: RollingUpdate # Güncellemeleri yavaş yavaş gerçekleştirirken çalışan uygulamaları etkilemez (RollingUpdate).
    rollingUpdate:
      maxUnavailable: 0 # Güncelleme sırasında hiç pod etkin değil olamaz, yani tüm podlar daima çalışır.
      maxSurge: 2 # Birden fazla pod aynı anda çalıştırılabilir. Bu durum, yeni podlar oluşturulurken eski podların hala çalışmasını sağlar
  template:
    metadata:
      labels:
        app: k8s # Pod etiketi olarak 'app: k8s' kullanılır.
    spec:
      containers:
      - name: k8s
        image: ozgurozturknet/k8s:v1 # K8s adlı container, 'ozgurozturknet/k8s' Docker imajından oluşturulur.
        ports:
        - containerPort: 80 # K8s container'ı 80 numaralı portu dinler.
        livenessProbe:
          httpGet:
            path: /healthcheck # Canlılık probu, /healthcheck yoluna HTTP GET isteği gönderir.
            port: 80
          initialDelaySeconds: 5 # İlk canlılık kontrolü, pod başlatıldıktan 5 saniye sonra gerçekleştirilir.
          periodSeconds: 5 # Ardından her 5 saniyede bir tekrarlanır.
        readinessProbe:
          httpGet:
            path: /ready # Hazır olma probu, /ready yoluna HTTP GET isteği gönderir.
            port: 80
          initialDelaySeconds: 20 # İlk hazır olma kontrolü, pod başlatıldıktan 20 saniye sonra gerçekleştirilir.
          periodSeconds: 3 # Ardından her 3 saniyede bir tekrarlanır.
      nodeSelector:
        tier: production # Podlar, 'tier: production' etiketine sahip nodlarda çalışır.
      tolerations:
      - key: "tier"
        operator: "Equal"
        value: "production"
        effect: "NoSchedule" # Podlar, 'tier: production' etiketine sahip nodlarda tolerasyon (NoSchedule) sağlar.
```

```
kubectl apply -f ./yaml/deployment.yaml
```

9.

```
kubectl expose deployment k8s-deployment --type=LoadBalancer -n production
```

10.

```
kubectl scale deployment k8s-deployment --replicas=3 -n production

kubectl scale deployment k8s-deployment --replicas=10 -n production

kubectl set image deployment/k8s-deployment k8s=ozgurozturknet/k8s:v2 -n production
```

11.

daemonset.yaml

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: logdaemonset
```

```
  labels:
    app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch # DaemonSet, etiketleri 'name: fluentd-elasticsearch' olan nodları seçer.
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch # Pod etiketi olarak 'name: fluentd-elasticsearch' kullanılır.
    spec:
      tolerations:
      - key: "tier"
        operator: "Equal"
        value: "production"
        effect: "NoSchedule" # Podlar, 'tier: production' etiketine sahip nodlarda tolerasyon (NoSchedule) sağlar.
      containers:
      - name: fluentd-elasticsearch
        image: quay.io/fluentd_elasticsearch/fluentd:v2.5.2 # Fluentd Elasticsearch için kullanılan imaj.
        resources:
          limits:
            memory: 200Mi # Fluentd container'ı için maksimum 200 MiB bellek sınırı.
          requests:
            cpu: 100m # Fluentd container'ı için minimum 100 millicpu (1/10 CPU çekirdeği) talebi.
            memory: 200Mi # Fluentd container'ı için minimum 200 MiB bellek talebi.
        volumeMounts:
        - name: varlog
          mountPath: /var/log # Fluentd, /var/log yoluna erişebilir.
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true # Fluentd, /var/lib/docker/containers yoluna salt okunur şekilde erişebilir.
      terminationGracePeriodSeconds: 30 # Konteyner kapanırken 30 saniyelik bir grace dönemi (kapanma öncesi bekleme süresi) sağlar.
      volumes:
      - name: varlog
        hostPath:
          path: /var/log # Host makinedeki /var/log yolunu Fluentd container'ına bağlar.
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers # Host makinedeki /var/lib/docker/containers yolunu Fluentd container'ına bağlar.
```

```
kubectl apply -f ./yaml/daemonset.yaml
```

12.

statefulset.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mongo-svc
  labels:
    app: mongo
spec:
  ports:
  - port: 27017 # Servis, 27017 numaralı porta gelen istekleri dinler.
    targetPort: 27017 # Gelen istekleri hedef olarak 27017 numaralı porta yönlendirir.
  clusterIP: None # Servis, küme IP'si olmadan başlatılır, yani Cluster IP ataması yapılmaz.
  selector:
    app: mongo # Bu servis, etiketi 'app: mongo' olan podları hedef alır.
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongostatefulset
spec:
  serviceName: mongo-svc # StatefulSet'in hizmet adı "mongo-svc" ile eşleşir.
  replicas: 2 # 2 adet pod oluşturacak şekilde StatefulSet yapılandırılır.
  selector:
    matchLabels:
```

```
      app: mongo # StatefulSet, etiketleri 'app: mongo' olan podları yönetir.
  template:
    metadata:
      labels:
        app: mongo
        environment: test
        replicaset: MainRepSet # Podlar, 'app: mongo', 'environment: test' ve 'replicaset: MainRepSet' etiketlerine sahip olacaktır.
    spec:
      terminationGracePeriodSeconds: 10 # Konteyner kapanırken 10 saniyelik bir grace dönemi (kapanma öncesi bekleme süresi) sağlar.
      nodeSelector:
        tier: production # Podlar, 'tier: production' etiketine sahip nodlarda çalışır.
      tolerations:
      - key: "tier"
        operator: "Equal"
        value: "production"
        effect: "NoSchedule" # Podlar, 'tier: production' etiketine sahip nodlarda tolerasyon (NoSchedule) sağlar.
      containers:
        - name: mongo-container
          image: mongo # Mongo veritabanını çalıştırmak için kullanılan Docker imajı.
          command:
            - "mongod" # Mongo'da çalışacak komut, "mongod" (Mongo veritabanı sunucusu).
            - "--bind_ip"
            - "0.0.0.0" # Sunucunun herhangi bir IP adresini dinlemesine izin verir.
            - "--replSet"
            - "MainRepSet" # Replica Set adı, veritabanının diğer node'larla iletişim kurmasını sağlar.
          resources:
            requests:
              cpu: 200m # Minimum 200 millicpu (1/5 CPU çekirdeği) talebi.
              memory: 200Mi # Minimum 200 MiB bellek talebi.
          ports:
            - containerPort: 27017 # Mongo veritabanı, 27017 numaralı portu dinler.
          volumeMounts:
            - name: mongo-pv-claim
              mountPath: /data/db # Veritabanı verilerini saklamak için kullanılan bağlama noktası.
  volumeClaimTemplates:
  - metadata:
      name: mongo-pv-claim
      annotations:
        volume.beta.kubernetes.io/storage-class: "default" # Depolama sınıfı, "default" olarak ayarlanmıştır.
    spec:
      accessModes:
        - ReadWriteOnce # Bir kez okuma ve yazma için uygun talep modu.
      volumeMode: Filesystem # Hacim modu, dosya sistemi olarak ayarlanmıştır.
      resources:
        requests:
          storage: 2Gi # Minimum 2 gigabayt depolama alanı talebi.
```

```
kubectl apply -f ./yaml/statefulset.yaml
```

```
kubectl exec -it mongostatefulset-0 -- bash

root@mongostatefulset-0:/# mongo

> rs.initiate({ _id: "MainRepSet", version: 1,
members: [
 { _id: 0, host: "mongostatefulset-0.mongo-svc.default.svc.cluster.local:27017" },
 { _id: 1, host: "mongostatefulset-1.mongo-svc.default.svc.cluster.local:27017" } ]});

MainRepSet:PRIMARY> db.getSiblingDB("admin").createUser({
...       user : "mongoadmin",
...       pwd  : "P@ssw0rd!1",
...       roles: [ { role: "root", db: "admin" } ]
...  });

MainRepSet:PRIMARY> rs.status();
```

13.

serviceaccount.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: sa-project
  namespace: default # Hizmet Hesabı, "default" adlı isim alanında oluşturulur.
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rb-project
  namespace: default # Küme Rol Bağlama işlemi, "default" adlı isim alanında yapılır.
subjects:
- kind: ServiceAccount
  name: sa-project # Küme Rol Bağlama işlemi, "sa-project" adlı hizmet hesabını hedef alır.
  namespace: default # Hizmet Hesabı, "default" adlı isim alanındadır.
  apiGroup: "" # Kullanılan API grubu belirtilmemiş, varsayılan olarak boş bırakılır.
roleRef:
  kind: ClusterRole
  name: view # "view" adlı Cluster Rolü, hizmet hesabına atanır.
  apiGroup: rbac.authorization.k8s.io # Rol API grubu belirtilir.
---
apiVersion: v1
kind: Pod
metadata:
  name: pod-proje
  namespace: default # Pod, "default" adlı isim alanında oluşturulur.
spec:
  serviceAccountName: sa-project # Pod, "sa-project" adlı hizmet hesabını kullanır.
  containers:
  - name: projecontainer
    image: ozgurozturknet/k8s:latest # "ozgurozturknet/k8s" Docker imajını kullanarak "projecontainer" adlı bir konteyner oluşturur.
    ports:
    - containerPort: 80 # Konteyner, 80 numaralı portu dinler.
```

```
kubectl apply -f ./yaml/serviceaccount.yaml
```

```
kubectl exec -it pod-proje -- bash

bash-5.0# CERT=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt

bash-5.0# TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)

bash-5.0# curl --cacert $CERT https://kubernetes/api/v1/pods --header "Authorization:Bearer $TOKEN" | jq '.items[].metadata.name'
```

Açıklama

Kubernetes API sunucusuna güvenli bir şekilde bağlanmak için kullanılacak olan sertifika dosyasının yolu.

```
bash-5.0# CERT=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
```

Kubernetes API sunucusuna kimlik doğrulaması için kullanılacak olan kimlik belirtecinin alınması.

```
bash-5.0# TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
```

Kubernetes API sunucusuna, pod listesini almak için yapılan bir HTTP isteği. "ca.crt" sertifikasını kullanarak doğrulama yapar ve kimlik doğrulama için elde edilen kimlik belirteci olan "TOKEN" kullanılır. Sonuçlar "jq" aracı ile işlenir ve her podun adı listelenir.

```
bash-5.0# curl --cacert $CERT https://kubernetes/api/v1/pods --header "Authorization:Bearer $TOKEN" | jq '.items[].metadata.name'
```

14.

```
$ nodedrain=$(kubectl get no -o jsonpath="{.items[3].metadata.name}")

$ kubectl drain $nodedrain --ignore-daemonsets --delete-local-data

$ kubectl cordon $nodedrain
```

Açıklama

Kubernetes kümesindeki düğümleri listeler ve dördüncü düğümün adını alır ve "nodedrain" adlı değişkene atar.

```
$ nodedrain=$(kubectl get no -o jsonpath="{.items[3].metadata.name}")
```

Belirtilen düğümdeki tüm çalışan podları durdurur ve varsa DaemonSet'ler hariç tutulur. Ayrıca, düğüme ait verileri siler.

```
$ kubectl drain $nodedrain --ignore-daemonsets --delete-local-data
```

Belirtilen düğümü işlem görmeyecek (cordoned) duruma getirir, yani yeni podlar o düğüme dağıtılmaz. Mevcut podlar etkilenmez ve çalışmaya devam eder.

```
$ kubectl cordon $nodedrain
```
