Day 5: åº”ç”¨å•¥åº·æ£€æŸ¥ä¸Žèµ„æº•é™•åˆ¶

ðŸŽ¯ å¦ä¹ ç›®æ ‡

- **æŠ€èƒ½ç›®æ ‡**: æŽŒæ•¡ä¸º Kubernetes åº”ç”¨é…•ç½®å•¥åº·æ£€æŸ¥å’Œèµ„æº•é…•é¢•çš„æ ¸å¿ƒæ–¹æ³•ã€‚
- **æ ¸å¿ƒæ¦‚å¿µ**: æ·±å…¥ç•†è§£ `Liveness Probe`, `Readiness Probe`, `Startup Probe` çš„åŒºåˆ«å’Œä½œç”¨ï¼Œä»¥å•Š `requests` å’Œ `limits` å¯¹ Pod è°ƒåº¦å’Œç¨³å®šæ€§çš„å½±å“•ã€‚
- **å…·ä½“æˆ•æžœ**:èƒ½å¤Ÿä¸ºä¸€ä¸ª Deployment é…•ç½® `httpGet` ç±»åž‹çš„å˜æ´»æŽ¢é’ˆå’Œå°±ç»ªæŽ¢é’ˆã€‚èƒ½å¤Ÿé€šè¿‡æ¨¡æ‹Ÿå•¥åº·æ£€æŸ¥å¤±è´¥ï¼Œè§‚å¯Ÿå¹¶è§£é‡Š Kubernetes çš„è‡ªæ„ˆè¡Œä¸ºï¼ˆé‡•å•¯ Pod æˆ–å°†å…¶ç§»å‡º Service ç«¯ç‚¹ï¼‰ã€‚èƒ½å¤Ÿä¸ºä¸€ä¸ªå®¹å™¨è®¾ç½®å•ˆç•†çš„ CPU å’Œå†…å˜ `requests` ä¸Ž `limits`ã€‚èƒ½å¤Ÿè§£é‡Š Kubernetes çš„ä¸‰ç§• QoS (Quality of Service) ç‰çº§ã€‚

ðŸ“š ç•†è®ºåŸºç¡€ (40%)

1. ä¸ºä»€ä¹ˆéœ€è¦•å•¥åº·æ£€æŸ¥ï¼Ÿ
ä¸€ä¸ªå®¹å™¨è¿›ç¨‹åœ¨è¿•è¡Œï¼Œä¸•ä»£è¡¨å®ƒæ••ä¾›çš„æœ•åŠ¡å°±ä¸€å®šæ£å¸¸ã€‚ä¾‹å¦‚ï¼š

- åº”ç”¨ç¨‹åº•å•¯èƒ½å•‘ç”Ÿæ»é”•ï¼Œè¿›ç¨‹ä»•åœ¨ä½†æ— æ³•å“•åº”è¯·æ±‚ã€‚
-
åº”ç”¨å•¯èƒ½å› ä¸ºä¾•èµ–çš„å•Žç«¯æœ•åŠ¡ï¼ˆå¦‚æ•°æ•®åº“ï¼‰æ— æ³•è¿žæŽ¥è€Œæš‚æ—¶æ— æ³•æ••ä¾›æœ•åŠ¡ã€‚
-
åº”ç”¨å•¯åŠ¨è¿‡ç¨‹è¾ƒé•¿ï¼Œéœ€è¦•ä¸€æ®µæ—¶é—´æ•¥åŠ è½½æ•°æ•®æˆ–é¢„çƒç¼“å˜ï¼ŒæœŸé—´æ— æ³•å¤„ç•†æµ•é‡•ã€‚

å¦‚æžœ Kubernetes
æ— æ³•æ„ŸçŸ¥åˆ°è¿™äº›å†…éƒ¨çŠ¶æ€•ï¼Œå®ƒå•¯èƒ½ä¼šå°†æµ•é‡•å•‘é€•ç»™ä¸€ä¸ªæ— æ³•å¤„ç•†è¯·æ±‚çš„
Podï¼Œæˆ–è€…æ— æ³•ä»Žä¸€ä¸ªå·²ç»•â€œåƒµæ»â€•çš„åº”ç”¨ä¸æ•¢å¤•ã€‚**å•¥åº·æŽ¢é’ˆ (Probes)** å°±æ˜¯ Kubelet
ç”¨æ•¥æ£€æµ‹å®¹å™¨å†…éƒ¨å•¥åº·çŠ¶å†µçš„æœºåˆ¶ã€‚

2. ä¸‰ç§•æŽ¢é’ˆ (Probes)
Kubelet å•¯ä»¥é…•ç½®ä¸‰ç§•æŽ¢é’ˆæ•¥æ£€æŸ¥å®¹å™¨ï¼š

- **`Liveness Probe` (å˜æ´»æŽ¢é’ˆ)**:**ä½œç”¨**: åˆ¤æ–å®¹å™¨æ˜¯å•¦**å˜æ´»**ã€‚**è¡Œä¸º**: å¦‚æžœå˜æ´»æŽ¢é’ˆ**å¤±è´¥**ï¼ŒKubelet ä¼šè®¤ä¸ºå®¹å™¨å·²ç»•æ»äº¡ï¼Œä¼š**æ•€æ»å¹¶é‡•å•¯**è¯¥å®¹å™¨ã€‚**é€‚ç”¨åœºæ™¯**: ç”¨äºŽæ£€æµ‹åº”ç”¨æ˜¯å•¦å•‘ç”Ÿæ»é”•æˆ–è¿›å…¥ä¸•å•¯æ•¢å¤•çš„æ•…éšœçŠ¶æ€•ï¼Œé€šè¿‡é‡•å•¯æ•¥å°•è¯•æ•¢å¤•æœ•åŠ¡ã€‚
- **`Readiness Probe` (å°±ç»ªæŽ¢é’ˆ)**:**ä½œç”¨**:

å^¤æ–å®¹å™¨æ˜¯å¦**å‡†å¤‡å¥½æŽ¥æ”¶æµ•é‡•**ã€‚**è¡Œä¸º**: å¦‚æžœå°±ç»ªæŽ¢é'ˆ**å¤±è´¥**ï¼ŒKubelet ä¸•ä¼šæ•€æ»å®¹å™¨ï¼Œè€Œæ˜¯å°†è¯¥ Pod ä»Ž Service çš„ Endpoints å^—è¡¨ä¸**ç§»é™¤**ã€‚è¿™æ ·ï¼Œæ–°çš„ç½'ç»œæµ•é‡•å°±ä¸•ä¼šå†•è¢«è½¬å•'å^°è¿™ä¸ª Podã€‚ç›´å^°å°±ç»ªæŽ¢é'ˆå†•æ¬¡æˆ•åŠŸï¼ŒPod æ‰•ä¼šè¢«é‡•æ–°åŠ å›ž Endpoints å^—è¡¨ã€‚**é€‚ç"¨åœºæ™¯**: ç"¨äºŽå¤„ç•†åº"ç"¨å•¯åŠ¨æ…¢ã€•ä¾•èµ–å¤–éƒ¨æœ•åŠ¡ã€•æˆ–éœ€è¦•è¿›è¡Œä¸´æ—¶ç»´æŠ¤çš„åœºæ™¯ã€‚
- **`Startup Probe` (å•¯åŠ¨æŽ¢é'ˆ)**:**ä½œç"¨**: å^¤æ–å®¹å™¨å†…çš„åº"ç"¨æ˜¯å¦å·²ç»•**å•¯åŠ¨æˆ•åŠŸ**ã€‚å®ƒåœ¨å…¶ä»–ä¸¤ç§•æŽ¢é'ˆä¹‹å‰æ‰§è¡Œã€‚**è¡Œä¸º**: å•ªæœ‰å½"å•¯åŠ¨æŽ¢é'ˆ**æˆ•åŠŸ**å•Žï¼Œå˜æ´»æŽ¢é'ˆå'Œå°±ç»ªæŽ¢é'ˆæ‰•ä¼šå¼€å§‹ï¿½ï¿½ä½œã€‚å¦‚æžœå•¯åŠ¨æŽ¢é'ˆåœ¨è®¾å®šçš„ `failureThreshold` * `periodSeconds` æ—¶é—´å†…ä¸€ç›´ä¸•æˆ•åŠŸï¼ŒKubelet å°±ä¼šæ•€æ»å¹¶é‡•å•¯å®¹å™¨ã€‚**é€‚ç"¨åœºæ™¯**: ä¸"é—¨ç"¨äºŽå•¯åŠ¨æ—¶é—´é•žå¸¸é•¿çš„åº"ç"¨ï¼Œå•¯ä»¥ç»™åº"ç"¨è¶³å¤Ÿçš„å•¯åŠ¨æ—¶é—´ï¼Œé•¿å…•è¢«å˜æ´»æŽ¢é'ˆè¿‡æ—©åœ°æ•€æ»ã€‚

3. æŽ¢é'ˆçš„é…•ç½®æ–¹å¼•

æ¯•ç§•æŽ¢é'ˆéƒ½å•¯ä»¥é€šè¿‡ä»¥ä¸‹ä¸‰ç§•æ–¹å¼•ä¹‹ä¸€æ¥é…•ç½®ï¼š

- **`httpGet`**: å•'å®¹å™¨çš„æŒ‡å®šç«¯å•£å'Œè·¯å¾„å•'é€•ä¸€ä¸ª HTTP GET è¯·æ±‚ã€‚å¦‚æžœè¿"å›žçš„ HTTP çŠ¶æ€•ç •åœ¨ 200-399 ä¹‹é—´ï¼Œå^™è®¤ä¸ºæŽ¢æµ‹æˆ•åŠŸã€‚
- **`exec`**: åœ¨å®¹å™¨å†…æ‰§è¡Œä¸€ä¸ªæŒ‡å®šçš„å'½ä»¤ã€‚å¦‚æžœå'½ä»¤çš„é€€å‡ºç •ä¸º 0ï¼Œå^™è®¤ä¸ºæŽ¢æµ‹æˆ•åŠŸã€‚
- **`tcpSocket`**: å°•è¯•ä¸Žå®¹å™¨çš„æŒ‡å®š TCP ç«¯å•£å»ºç«‹è¿žæŽ¥ã€‚å¦‚æžœè¿žæŽ¥èƒ½å¤Ÿæˆ•åŠŸå»ºç«‹ï¼Œå^™è®¤ä¸ºæŽ¢æµ‹æˆ•åŠŸã€‚

4. èµ„æº•è¯·æ±‚ (Requests) ä¸Žé™•å^¶ (Limits)

åœ¨å®šä¹‰ Pod æ—¶ï¼Œä½ å•¯ä»¥ä¸ºæ¯•ä¸ªå®¹å™¨æŒ‡å®šå®ƒéœ€è¦•çš„ CPU å'Œå†…å˜èµ„æº•ã€‚

- **`requests` (èµ„æº•è¯·æ±‚)**:**ä½œç"¨**: å'Šè¯‰è°ƒåº¦å™¨ (Scheduler)ï¼Œè¿™ä¸ªå®¹å™¨**è‡³å°'éœ€è¦•**å¤šå°'èµ„æº•æ‰•èƒ½æ£å¸¸è¿•è¡Œã€‚**è¡Œä¸º**: èºƒåº¦å™¨åœ¨èºƒåº¦ Pod æ—¶ï¼Œä¼šç¡®ä¿•ç›®æ ‡èŠ‚ç‚¹ä¸Šæœ‰è¶³å¤Ÿçš„å•¯ç"¨èµ„æº•æ•¥æ»¡è¶³ Pod æ‰€æœ‰å®¹å™¨çš„ `requests` æ€»å'Œã€‚`requests` æ˜¯ä¸€ä¸ª**æœ‰ä¿•è¯•çš„**èµ„æº•é‡•ã€‚**å••ä½•**: CPU çš„å••ä½•æ˜¯ `cores` (æ ¸å¿ƒæ•°)ï¼Œå•¯ä»¥å†™æˆ• `0.5` æˆ– `500m` (500 millicores)ã€‚å†…å˜çš„ï¿½ï¿½ï¿½ä½•æ˜¯å—èŠ‚ï¼Œé€šå¸¸ä½¿ç"¨ `Mi` (Mebibytes) æˆ– `Gi` (Gibibytes)ã€‚
- **`limits` (èµ„æº•é™•å^¶)**:**ä½œç"¨**: å®šä¹‰ä¸€ä¸ªå®¹å™¨**æœ€å¤šå•¯ä»¥**ä½¿ç"¨å¤šå°'èµ„æº•ã€‚**è¡Œä¸º**:**CPU**: å¦‚æžœå®¹å™¨çš„ CPU ä½¿ç"¨è¯•å›¾è¶…è¿‡ `limits`ï¼Œå®ƒçš„ CPU æ—¶é—´ä¼šè¢«**èŠ‚æµ• (throttled)**ï¼Œå¯¼è‡´æ€§èƒ½ä¸‹é™•ã€‚**å†…å˜**: å¦‚æžœå®¹å™¨çš„å†…å˜ä½¿ç"¨è¶…è¿‡

`limits`ï¼Œå®ƒä¼šè¢«ç³»ç»Ÿ**æ•€æ-»**ï¼ˆOOMKilled, Out of Memory Killedï¼‰ã€‚**æ ¸å¿ƒä»·å€¼**: é˜²æ¢å••ä¸ªæœ‰é—®é¢˜çš„å®¹å™¨ï¼ˆå¦‚å†…å˜æ³„æ¼•ï¼‰è€—å°½æ•´ä¸ªèŠ‚ç‚¹çš„èµ„æº•ï¼Œä»Žè€Œå½±å“•åˆ°èŠ‚ç‚¹ä¸Šå…¶ä»–æ‰€æœ‰ Pod çš„ç¨³å®šæ€§ã€‚

5. QoS (Quality of Service) ç-‰çº§

æ ¹æ•®å®¹å™¨è®¾ç½®çš„ `requests` å'Œ `limits`ï¼ŒKubernetes ä¼šä¸º Pod åˆ†é…•ä¸‰ç§•ä¸•å•Œçš„ QoS ç-‰çº§ï¼š

- **`Guaranteed` (æœ‰ä¿•è¯•çš„)**:**æ•¡ä»¶**: Pod ä¸-çš„**æ¯•ä¸€ä¸ª**å®¹å™¨éƒ½å¿…é¡»å•Œæ—¶è®¾ç½®äº† CPU å'Œå†…å˜çš„ `requests` å'Œ `limits`ï¼Œå¹¶ä¸" `requests` å€¼å¿…é¡»**ç-‰äºŽ** `limits` å€¼ã€‚**å¾…é•‡**: æœ€é«˜ä¼˜å…ˆçº§ã€‚è¿™ç§• Pod æœ€ä¸•å•¯èƒ½åœ¨èŠ‚ç‚¹èµ„æº•ç´§å¼ æ—¶è¢«æ•€æ-»ã€‚
- **`Burstable` (å•¯çª•å•'çš„)**:**æ•¡ä»¶**: Pod ä¸-è‡³å°'æœ‰ä¸€ä¸ªå®¹å™¨è®¾ç½®äº† CPU æˆ–å†…å˜çš„ `requests`ï¼Œä½†ä¸•æ»¡è¶³ `Guaranteed` çš„æ•¡ä»¶ï¼ˆä¾‹å¦‚ï¼Œ`limits` å¤§äºŽ `requests`ï¼Œæˆ–å•ªè®¾ç½®äº† `requests`ï¼‰ã€‚**å¾…é•‡**: ä¸-ç-‰ä¼˜å…ˆçº§ã€‚
- **`BestEffort` (å°½åŠ›è€Œä¸ºçš„)**:**æ•¡ä»¶**: Pod ä¸-çš„æ‰€æœ‰å®¹å™¨éƒ½æ²¡æœ‰è®¾ç½®ä»»ä½• `requests` æˆ– `limits`ã€‚**å¾…é•‡**: æœ€ä½Žä¼˜å…ˆçº§ã€‚å½"èŠ‚ç‚¹èµ„æº•ä¸•è¶³æ—¶ï¼Œè¿™ç§• Pod æ˜¯**æœ€å…ˆè¢«é©±é€•æˆ–æ•€æ-»**çš„ã€‚

æœ€ä½³å®žè·µ: æ€»æ˜¯ä¸ºä½ çš„ç”Ÿäº§åº"ç"¨è®¾ç½® `requests` å'Œ `limits`ï¼Œè‡³å°'è®©å®ƒä»¬æˆ•ä¸º `Burstable`ï¼Œä»¥ä¿•è¯•åŸºæœ¬çš„è¿•è¡Œèµ„æº•å'Œç¨³å®šæ€§ã€‚

ðŸ› ï¸• å®žè·µæ“•ä½œ (50%)

1. ä¸º Deployment æ·»åŠ å•¥åº·æŽ¢é'ˆ

ä¿®æ"¹ Day 2 çš„ `nginx-deployment.yaml`ï¼Œä¸ºå…¶æ·»åŠ `livenessProbe` å'Œ `readinessProbe`ã€‚

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25
        ports:
        - containerPort: 80
        livenessProbe:
          httpGet:
            path: / # æ£€æŸ¥æ ¹è·¯å¾„
            port: 80
          initialDelaySeconds: 5 # Pod å•¯åŠ¨å•Ž 5 ç§’å¼€å§‹ç¬¬ä¸€æ¬¡æŽ¢æµ‹
          periodSeconds: 10    # æ¯• 10 ç§’æŽ¢æµ‹ä¸€æ¬¡
        readinessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 3
          periodSeconds: 5
```

éƒ¨ç½²: `kubectl apply -f nginx-deployment.yaml`

2. æ¨¡æ‹Ÿ Liveness Probe å¤±è´¥

```
# æ‰¾ï¿½ï¿½ä¸€ä¸ª Nginx Pod çš„å••å—
kubectl get pods -l app=nginx

# è¿›å…¥ Podï¼Œæ‰‹åŠ¨åˆ é™¤é¦–é¡µæ–‡ä»¶ï¼Œè®© httpGet / è¿”å›ž 404
kubectl exec -it <nginx-pod-name> -- rm /usr/share/nginx/html/index.html

# è§‚å¯Ÿ Pod çŠ¶æ€•
kubectl get pods -l app=nginx -w
# ä½ ä¼šçœ‹åˆ°è¯¥ Pod çš„ RESTARTS æ¬¡æ•°ä»Ž 0 å•˜ä¸º 1ï¼Œå› ä¸ºå®ƒè¢« Kubelet
é‡•å•¯äº†ã€‚

# æŸ¥çœ‹ Pod äº‹ä»¶ï¼Œå•¯ä»¥çœ‹åˆ° Liveness probe failed çš„è®°å½•
kubectl describe pod <nginx-pod-name>
```

3. æ¨¡æ‹Ÿ Readiness Probe å¤±è´¥

ä¸ºäº†æ›´å¥½åœ°è§‚å¯Ÿï¼Œæˆ'ä»¬å…ˆåˆ›å»ºä¸€ä¸ª Service æŒ‡å'è¿™ä¸ª Deploymentã€‚

```
kubectl expose deployment nginx-deployment --port=80 --type=ClusterIP
```

çŽ°åœ¨ï¼Œå†•æ¬¡åˆ é™¤ä¸€ä¸ª Pod çš„é¦–é¡µæ–‡ä»¶ã€‚

```
# è¿›å…¥å•¦ä¸€ä¸ª Podï¼Œåˆ é™¤é¦–é¡µæ–‡ä»¶
kubectl exec -it <another-nginx-pod-name> -- rm
/usr/share/nginx/html/index.html

# è§‚å¯Ÿ Pod çŠ¶æ€•ï¼ŒREADY åˆ—ä¼šä»Ž 1/1 å•˜ä¸º 0/1
```

```
kubectl get pods -l app=nginx
# NAME                                READY   STATUS    RESTARTS   AGE
# nginx-deployment-xxxx-abcde         1/1     Running   0          10m
# nginx-deployment-xxxx-fghij         0/1     Running   0          5m  <--
å°±ç»ªæŽ¢é’ˆå¤±è´¥

# æŸ¥çœ‹ Service çš„ Endpointsï¼Œä¼šå•‘çŽ°å¤±è´¥çš„ Pod çš„ IP
å·²ç»•è¢«ç§»é™¤äº†
kubectl describe svc nginx-deployment
# Endpoints:          10.244.1.12:80  <-- å•ªå‰©ä¸‹ä¸€ä¸ªå•¥åº·çš„ Pod
```

è¿™è¯•æ˜Žäº†å°±ç»ªæŽ¢é’ˆå¤±è´¥å•Žï¼Œæµ•é‡•å°†ä¸•å†•è¢«å•‘é€•åˆ°æœ‰é—®é¢˜ çš„ Podã€‚

4. è®¾ç½®èµ„æº•è¯·æ±‚å’Œé™•åˆ¶

ä¿®æ”¹ `nginx-deployment.yaml`ï¼Œä¸ºå®¹å™¨æ·»åŠ èµ„æº•é…•ç½®ã€‚

```
# ...
        ports:
        - containerPort: 80
        resources:
          requests:
            memory: "64Mi"
            cpu: "250m" # 1/4 æ ¸
          limits:
            memory: "128Mi"
            cpu: "500m" # 1/2 æ ¸
# ...
```

é‡•æ–°éƒ¨ç½²: `kubectl apply -f nginx-deployment.yaml`

æŸ¥çœ‹ Pod çš„ QoS ç‰çº§ï¼š

```
kubectl get pod <nginx-pod-name> -o yaml
# ...
# status:
#   qosClass: Burstable
```

æŸ¥çœ‹èŠ‚ç‚¹ä¸Šçš„èµ„æº•åˆ†é…•æƒ…å†µï¼š

```
kubectl describe node minikube
# ...
# Allocated resources:
#   (Total limits may be over 100 percent, i.e., overcommitted.)
#   Resource           Requests      Limits
#   --------           --------      ------
#   cpu                500m (25%)    1 (50%)
#   memory             128Mi (1%)    256Mi (3%)
# ...
```

å•¯ä»¥çœ‹åˆ°ï¼Œä¸¤ä¸ª Pod çš„ `requests` å’Œ `limits` éƒ½è¢«ç»Ÿè®¡è¿›åŽ»äº†ã€‚

ðŸ’» Go ç¼–ç¨‹å®žçŽ° (10%)

é¡¹ç›®: k8s-pod-resource-viewer

ç›®æ ‡: ç¼–å†™ä¸€ä¸ª Go ç¨‹åºï¼Œåˆ—å‡ºæŒ‡å®šå‘½åç©ºé—´ä¸‹æ‰€æœ‰ Pod å•Šå…¶å®¹å™¨çš„èµ„æº• `requests` å’Œ `limits`ã€‚

```
package main

import (
 "context"
 "fmt"
 "log"
 "os"
 "path/filepath"

 metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
 "k8s.io/client-go/kubernetes"
 "k8s.io/client-go/tools/clientcmd"
)

func main() {
 if len(os.Args) < 2 {
  fmt.Println("ç”¨æ³•: go run main.go <namespace>")
  os.Exit(1)
 }
 namespace := os.Args[1]

 // --- é…ç½®å’Œåˆ›å»º clientset ---
 userHomeDir, _ := os.UserHomeDir()
 kubeconfig := filepath.Join(userHomeDir, ".kube", "config")
 config, _ := clientcmd.BuildConfigFromFlags("", kubeconfig)
 clientset, _ := kubernetes.NewForConfig(config)

 fmt.Printf("--- Pod Resources in namespace '%s' ---\n", namespace)
 podList, err := clientset.CoreV1().Pods(namespace).List(context.TODO(),
metav1.ListOptions{})
 if err != nil {
  log.Fatal(err)
 }

 for _, pod := range podList.Items {
  fmt.Printf("- Pod: %s\n", pod.Name)
  for _, container := range pod.Spec.Containers {
   fmt.Printf("  - Container: %s\n", container.Name)
   fmt.Printf("    Requests:\n")
   fmt.Printf("      CPU: %s\n", container.Resources.Requests.Cpu().String())
   fmt.Printf("      Memory: %s\n",
container.Resources.Requests.Memory().String())
   fmt.Printf("    Limits:\n")
   fmt.Printf("      CPU: %s\n", container.Resources.Limits.Cpu().String())
   fmt.Printf("      Memory: %s\n",
container.Resources.Limits.Memory().String())
  }
  fmt.Println("--------------------")
 }
}
```

ðŸ”• æ•…éšœæŽ’æŸ¥ä¸Žä¼˜åŒ–

- **Pod å› Liveness Probe å¤±è´¥è¢«å••å¤•é‡•å•¯**:`kubectl describe pod` æŸ¥çœ‹äº‹ä»¶ï¼Œç¡®è®¤æ˜¯å-˜æ´»æŽ¢é’ˆå¤±è´¥ã€‚æ£€æŸ¥æŽ¢é’ˆçš„é…•ç½®æ˜¯å•¦æ-£ç¡®ï¼ˆè·¯å¾„ã€•ç«¯å•£ï¼‰ã€‚å•¯èƒ½æ˜¯åº”ç”¨æœ¬èº«æœ‰é—®é¢˜ï¼Œ`kubectl logs --previous <pod-name>` æŸ¥çœ‹ä¸Šä¸€ä¸ªè¢«æ•€æ-»çš„å®¹å™¨çš„æ—¥å¿—ã€‚å•¯èƒ½æ˜¯ `initialDelaySeconds` è®¾ç½®å¤ªçŸ-ï¼Œåº”ç”¨è¿˜æ²¡å•¯åŠ¨å¥½å°±è¢«æŽ¢æµ‹äº†ã€‚
- **Pod æ— æ³•è¾¾åˆ° Ready çŠ¶æ€•**:`kubectl describe pod` æŸ¥çœ‹äº‹ä»¶ï¼Œç¡®è®¤æ˜¯å°±ç»ªæŽ¢é’ˆå¤±è´¥ã€‚æ£€æŸ¥åº”ç”¨æ˜¯å•¦èƒ½æ-£å¸¸å“•åº”æŽ¢æµ‹è¯·æ±‚ã€‚
- **Pod å› OOMKilled è¢«é‡•å•¯**:`kubectl describe pod` æŸ¥çœ‹ `Reason: OOMKilled`ã€‚è¯´æ˜Žå†…å-˜ `limits` è®¾ç½®å¤ªå°•ï¼Œéœ€è¦•è°ƒå¤§ã€‚

ðŸ• è¯¾å•Žä½œä¸š

1. **ç ”ç©¶ `exec` æŽ¢é’ˆ**: åˆ›å»ºä¸€ä¸ª Podï¼Œä½¿ç”¨ `exec` ç±»åž‹çš„æŽ¢é’ˆã€‚ä¾‹å¦‚ï¼Œ`command: ["cat", "/tmp/healthy"]`ã€‚ç„¶åŽé€šè¿‡ `kubectl exec` è¿›å…¥ Pod åˆ›å»ºæˆ–åˆ é™¤ `/tmp/healthy` æ–‡ä»¶ï¼Œè§‚å¯ŸæŽ¢é’ˆçŠ¶æ€•çš„å•˜åŒ–ã€‚
2. **ç ”ç©¶ `Guaranteed` QoS**: ä¿®æ”¹ä½ çš„ Deploymentï¼Œè®© CPU å’Œå†…å-˜çš„ `requests` å’Œ `limits` å®Œå…¨ç›¸ç-‰ã€‚éƒ¨ç½²å•Žï¼Œä½¿ç”¨ `kubectl get pod <name> -o yaml` éªŒè¯•å…¶ `qosClass` æ˜¯å•¦å•˜ä¸ºäº† `Guaranteed`ã€‚
3. **æ€•èƒƒ**: åœ¨ä»€ä¹ˆæƒ…å†µä¸‹ï¼Œä½ åº”è¯¥åªè®¾ç½® `readinessProbe` è€Œä¸•è®¾ç½® `livenessProbe`ï¼Ÿï¼ˆæ••ç¤ºï¼šè€ƒè™‘ä¸€ä¸ªéœ€è¦•ä»Žé˜Ÿåˆ—ä¸-å¤„ç•†ä»»åŠ¡ï¼Œä½†å¤„ç•†ä¸€ä¸ªä»»åŠ¡å•¯èƒ½è€—æ—¶å¾ˆé•¿çš„ Worker åº”ç”¨ï¼‰ã€‚