KUBERNETES 빠른 참조

kubectl, 파드, 디플로이먼트, 서비스, 설정, 디버깅

kubectl 기본

클러스터 정보

```
kubectl cluster-info
kubectl get nodes
kubectl config current-context
kubectl config use-context my-cluster
```

필수 명령어

`kubectl get <resource>`	리소스 목록 조회
`kubectl describe <resource> <name>`	리소스 상세 정보
`kubectl create -f file.yaml`	파일로 리소스 생성
`kubectl apply -f file.yaml`	리소스 생성 또는 업데이트
`kubectl delete -f file.yaml`	파일로 리소스 삭제
`kubectl edit <resource> <name>`	리소스 인플레이스 편집
`kubectl api-resources`	모든 리소스 유형 목록

출력 형식

`-o wide`	추가 컬럼 표시 (IP, 노드)
`-o yaml`	전체 YAML 출력
`-o json`	전체 JSON 출력
`-o jsonpath='{.spec}'`	특정 필드 추출
`--sort-by=.metadata.name`	필드 기준 정렬

파드

파드 작업

```
kubectl get pods
kubectl get pods -A                    # all namespaces
kubectl run nginx --image=nginx        # quick pod
kubectl delete pod nginx
```

파드 YAML

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  labels: { app: myapp }
spec:
  containers:
    - name: app
      image: nginx:1.27
      ports:
        - containerPort: 80
```

파드 상태값

`Running`	모든 컨테이너 시작됨
`Pending`	스케줄링 또는 이미지 풀 대기 중
`CrashLoopBackOff`	컨테이너가 계속 충돌하고 재시작
`ImagePullBackOff`	컨테이너 이미지를 가져올 수 없음
`Completed`	실행 완료 (Job)

디플로이먼트

디플로이먼트 YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels: { app: web }
  template:
    metadata:
      labels: { app: web }
    spec:
      containers:
        - name: web
          image: nginx:1.27
          ports:
            - containerPort: 80
```

디플로이먼트 명령어

`kubectl get deploy`	디플로이먼트 목록
`kubectl scale deploy web --replicas=5`	레플리카 수 조정
`kubectl set image deploy/web web=nginx:1.28`	이미지 업데이트 (롤링)
`kubectl rollout status deploy/web`	롤아웃 진행 상황 확인
`kubectl rollout undo deploy/web`	이전 리비전으로 롤백
`kubectl rollout history deploy/web`	리비전 히스토리 조회

서비스

서비스 유형

`ClusterIP`	클러스터 내부 전용 (기본값)
`NodePort`	각 노드 IP의 정적 포트로 외부 노출
`LoadBalancer`	외부 로드 밸런서 (클라우드)
`ExternalName`	외부 서비스로의 DNS 별칭

서비스 YAML

```
apiVersion: v1
kind: Service
metadata:
  name: web-svc
spec:
  type: ClusterIP
  selector: { app: web }
  ports:
    - port: 80
      targetPort: 80
```

빠른 노출

```
kubectl expose deploy web --port=80 --type=ClusterIP
kubectl expose deploy web --port=80 --type=NodePort
kubectl get svc
```

ConfigMap 및 Secret

ConfigMap

```
kubectl create configmap app-cfg \
  --from-literal=DB_HOST=db.example.com \
  --from-file=config.ini
```

Secret

```
kubectl create secret generic db-creds \
  --from-literal=username=admin \
  --from-literal=password=s3cret
```

파드에서 사용

```
# As environment variables
envFrom:
  - configMapRef: { name: app-cfg }
  - secretRef:    { name: db-creds }

# As volume mount
volumes:
  - name: cfg
    configMap: { name: app-cfg }
```

명령어

`kubectl get cm`	ConfigMap 목록
`kubectl get secret`	Secret 목록
`kubectl describe cm app-cfg`	ConfigMap 데이터 표시
`kubectl get secret db-creds -o yaml`	Secret 표시 (base64 인코딩)

네임스페이스

네임스페이스 명령어

`kubectl get ns`	네임스페이스 목록
`kubectl create ns staging`	네임스페이스 생성
`kubectl delete ns staging`	네임스페이스 및 모든 리소스 삭제
`kubectl get pods -n staging`	네임스페이스의 파드 목록
`kubectl get pods -A`	전체 네임스페이스의 파드 목록

기본 네임스페이스 설정

```
kubectl config set-context --current \
  --namespace=staging
```

볼륨

PersistentVolumeClaim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: data-pvc
spec:
  accessModes: [ReadWriteOnce]
  resources:
    requests: { storage: 10Gi }
```

파드에 마운트

```
volumes:
  - name: data
    persistentVolumeClaim:
      claimName: data-pvc
containers:
  - volumeMounts:
      - name: data
        mountPath: /app/data
```

볼륨 유형

`emptyDir`	임시 디렉터리, 파드 삭제 시 함께 삭제
`hostPath`	호스트 파일시스템 경로 마운트
`persistentVolumeClaim`	영구 스토리지 (PVC)
`configMap`	ConfigMap을 파일로 마운트
`secret`	Secret을 파일로 마운트

인그레스

인그레스 YAML

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web-ingress
spec:
  rules:
    - host: app.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web-svc
                port: { number: 80 }
```

인그레스 참고

`Ingress Controller`	필수 (nginx-ingress, traefik 등)
`pathType: Prefix`	URL 접두사 매칭
`pathType: Exact`	정확한 URL 경로 매칭
`TLS`	Secret 이름으로 `tls:` 섹션 추가

디버깅

진단 명령어

`kubectl logs <pod>`	컨테이너 stdout/stderr
`kubectl logs <pod> -c <ctr>`	특정 컨테이너 로그
`kubectl logs <pod> --previous`	충돌한 컨테이너의 로그
`kubectl describe pod <pod>`	이벤트, 조건, 상태
`kubectl exec -it <pod> -- sh`	컨테이너에 쉘 접속
`kubectl port-forward <pod> 8080:80`	로컬 포트를 파드로 포워딩
`kubectl top pods`	CPU/메모리 사용량 (metrics-server 필요)
`kubectl get events --sort-by=.lastTimestamp`	클러스터 이벤트 타임라인

디버그 파드

```
kubectl run debug --rm -it --image=busybox -- sh
# or attach ephemeral container
kubectl debug -it <pod> --image=busybox
```

주요 패턴

레이블 및 셀렉터

```
kubectl get pods -l app=web
kubectl get pods -l 'env in (prod,staging)'
kubectl label pod myapp env=prod
```

리소스 제한

```
resources:
  requests: { cpu: 100m, memory: 128Mi }
  limits:   { cpu: 500m, memory: 256Mi }
```

활성 및 준비 프로브

```
livenessProbe:
  httpGet: { path: /healthz, port: 8080 }
  initialDelaySeconds: 5
  periodSeconds: 10
readinessProbe:
  httpGet: { path: /ready, port: 8080 }
```

빠른 레시피

`Dry run`	`kubectl apply -f file.yaml --dry-run=client`
`Generate YAML`	`kubectl create deploy web --image=nginx --dry-run=client -o yaml`
`Watch`	`kubectl get pods -w`
`Copy files`	`kubectl cp file.txt pod:/tmp/`
`Restart deploy`	`kubectl rollout restart deploy/web`