KUBERNETES クイックリファレンス

kubectl、Pod、デプロイメント、サービス、設定、デバッグ

kubectl の基本

クラスター情報

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

必須コマンド

`kubectl get <resource>`	リソースを一覧表示
`kubectl describe <resource> <name>`	リソースの詳細情報
`kubectl create -f file.yaml`	ファイルからリソースを作成
`kubectl apply -f file.yaml`	リソースを作成または更新
`kubectl delete -f file.yaml`	ファイルからリソースを削除
`kubectl edit <resource> <name>`	リソースをその場で編集
`kubectl api-resources`	すべてのリソースタイプを一覧表示

出力フォーマット

`-o wide`	追加列（IP、ノード）
`-o yaml`	完全な YAML 出力
`-o json`	完全な JSON 出力
`-o jsonpath='{.spec}'`	特定フィールドを抽出
`--sort-by=.metadata.name`	フィールドで出力をソート

Pod

Pod 操作

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

Pod YAML

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

Pod ステータス値

`Running`	すべてのコンテナが起動済み
`Pending`	スケジューリングまたはイメージのプルを待機中
`CrashLoopBackOff`	コンテナがクラッシュし再起動を繰り返している
`ImagePullBackOff`	コンテナイメージをプルできない
`Completed`	完了まで実行された（Job）

デプロイメント

デプロイメント 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
```

デプロイメントコマンド

`kubectl get deploy`	デプロイメントを一覧表示
`kubectl scale deploy web --replicas=5`	レプリカ数をスケール
`kubectl set image deploy/web web=nginx:1.28`	イメージを更新（ローリング）
`kubectl rollout status deploy/web`	ロールアウトの進行状況を監視
`kubectl rollout undo deploy/web`	前のリビジョンにロールバック
`kubectl rollout history deploy/web`	リビジョン履歴を表示

サービス

サービスタイプ

`ClusterIP`	内部のみ（デフォルト）
`NodePort`	各ノードの IP の固定ポートで公開
`LoadBalancer`	外部ロードバランサー（クラウド）
`ExternalName`	外部サービスへの DNS エイリアス

サービス YAML

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

クイック公開

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

ConfigMap と 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
```

Pod での使用

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

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

コマンド

`kubectl get cm`	ConfigMap を一覧表示
`kubectl get secret`	Secret を一覧表示
`kubectl describe cm app-cfg`	ConfigMap のデータを表示
`kubectl get secret db-creds -o yaml`	Secret を表示（base64 エンコード）

ネームスペース

ネームスペースコマンド

`kubectl get ns`	ネームスペースを一覧表示
`kubectl create ns staging`	ネームスペースを作成
`kubectl delete ns staging`	ネームスペースとすべてのリソースを削除
`kubectl get pods -n staging`	ネームスペース内の Pod を一覧表示
`kubectl get pods -A`	すべてのネームスペースの Pod を一覧表示

デフォルトネームスペースを設定

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

ボリューム

PersistentVolumeClaim

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

Pod へのマウント

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

ボリュームタイプ

`emptyDir`	一時ディレクトリ、Pod と共に削除
`hostPath`	ホストファイルシステムパスをマウント
`persistentVolumeClaim`	永続ストレージ（PVC）
`configMap`	ConfigMap をファイルとしてマウント
`secret`	Secret をファイルとしてマウント

Ingress

Ingress 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 }
```

Ingress の注意点

`Ingress Controller`	必須（nginx-ingress、traefik など）
`pathType: Prefix`	URL プレフィックスにマッチ
`pathType: Exact`	正確な URL パスにマッチ
`TLS`	Secret 名を指定した `tls:` セクションを追加

デバッグ

診断コマンド

`kubectl logs <pod>`	コンテナの stdout/stderr
`kubectl logs <pod> -c <ctr>`	特定コンテナのログ
`kubectl logs <pod> --previous`	クラッシュしたコンテナのログ
`kubectl describe pod <pod>`	イベント、条件、ステータス
`kubectl exec -it <pod> -- sh`	コンテナへのシェル接続
`kubectl port-forward <pod> 8080:80`	ローカルポートを Pod にフォワード
`kubectl top pods`	CPU / メモリ使用量（metrics-server）
`kubectl get events --sort-by=.lastTimestamp`	クラスターイベントのタイムライン

デバッグ Pod

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

よく使うパターン

ラベルとセレクター

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

リソース制限

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

ライブネスとレディネス

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

クイックレシピ

`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`