Kubernetes Workshop

Day 2

Resource Management

Requests and Limits

- Requests: Guaranteed resources for scheduling
- Scheduler uses requests to place pods
- Pod gets at least this much
- Limits: Maximum resources allowed
- CPU limit exceeded → Throttled
- Memory limit exceeded → OOMKilled

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  template:
    spec:
      containers:
        - name: app
          image: my-app:1.0
          resources:
            requests:
              cpu: "250m"
              memory: "256Mi"
            limits:
              cpu: "1"
              memory: "512Mi"
```

LimitRange

- Default and max/min constraints per namespace
- Applied to Pods/Containers that don't specify resources

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: resource-limits
  namespace: dev
spec:
  limits:
    - type: Container
      default:
        cpu: "500m"
        memory: "256Mi"
      defaultRequest:
        cpu: "100m"
        memory: "128Mi"
      max:
        cpu: "2"
        memory: "2Gi"
      min:
        cpu: "50m"
        memory: "64Mi"
```

Health Probes

Health Probes Overview

- Liveness Probe: Is the container alive?
  - Fails → Kubernetes kills and restarts container
  - Use for: detecting deadlocks, hung processes
- Readiness Probe: Is the container ready for traffic?
  - Fails → Removed from Service endpoints (no traffic)
  - Use for: startup warmup, temporary unavailability
- Startup Probe: Has the container started?
  - Disables liveness/readiness until it passes
  - Use for: slow-starting applications

Probe Types

- httpGet - HTTP request to endpoint
  - Success: HTTP 200-399
  - Best for: Web apps, APIs, microservices
- tcpSocket : TCP connection attempt
  - Success: Connection established
  - Best for: Databases, Redis, message queues
- Exec: Run command in container
  - Success: Exit code 0
  - Best for: Custom health checks

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: probe-demo
spec:
  containers:
  - name: app
    image: my-app:1.0
    ports:
    - containerPort: 8080
    readinessProbe:
      httpGet:
        path: /healthz
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 10
      failureThreshold: 3
    livenessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 15
      periodSeconds: 20
      failureThreshold: 3
    startupProbe:
      exec:
        command:
        - cat
        - /tmp/app-initialized
      initialDelaySeconds: 0
      periodSeconds: 5
      failureThreshold: 30
```

Scheduling and Placement

NodeSelector

- Simplest way to control pod placement
- Pod only schedules on nodes with ALL matching labels.

```
kubectl label nodes aks-nodepool1-12345 gpu=true disktype=ssd
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
spec:
  nodeSelector:
    gpu: "true"
    disktype: ssd
  containers:
    - name: ml-training
      image: tensorflow/tensorflow:2.20-gpu
```

Taint

- Blocks all pods by default

```
kubectl taint nodes node1 dedicated=gpu:NoSchedule
kubectl taint nodes node1 dedicated=gpu:NoSchedule
```

Tolerations

- Added to pods to allow scheduling on tainted nodes
- Must match key + value + effect
- Toleration does NOT force pod to that node

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: production-app
spec:
  containers:
  - name: app
    image: my-app:1.0

  tolerations:
  - key: "environment"
    operator: "Equal"
    value: "production"
    effect: "NoSchedule"
```

Node Affinity

- More expressive than nodeSelector

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: required-affinity-pod
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - westeurope-1
            - westeurope-2
  containers:
  - name: app
    image: nginx
```

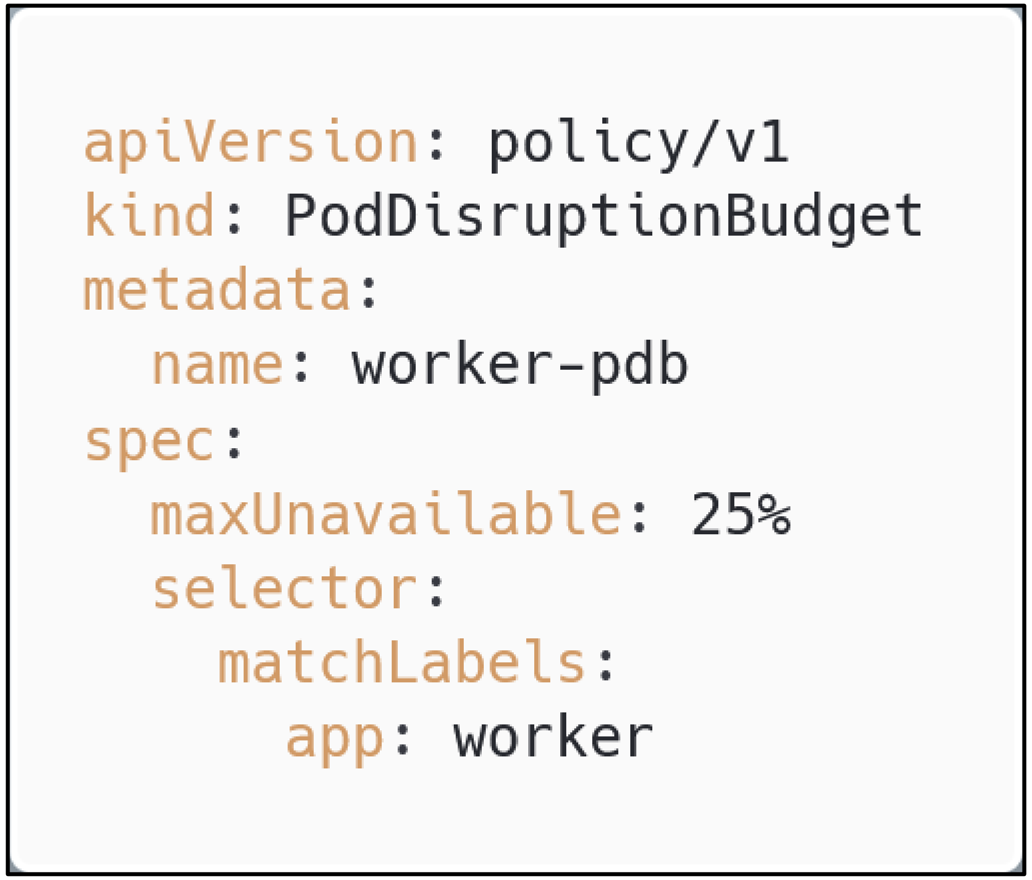
Pod Affinity and Anti-Affinity

- Pod Affinity: Schedule pods together
  - Use case: Web server + cache on same node (low latency)
- Pod Anti-Affinity: Spread pods apart
  - Use case: Database replicas on different nodes (HA)
- topologyKey defines the domain:
  - kubernetes.io/hostname = same/different node
  - topology.kubernetes.io/zone = same/different AZ

Pod Disruption Budget

Pod Disruption Budgets (PDB)

- Protect availability during voluntary disruptions
- Voluntary: node drains, cluster upgrades, autoscaler
- Does not protect against node crashes, pod deletions, app failures
- minAvailable: At least N pods must stay running
- maxUnavailable: At most N pods can be down



```yaml
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: worker-pdb
spec:
  maxUnavailable: 25%
  selector:
    matchLabels:
      app: worker
```

Labs

- 2.01: StatefulSets
- 2.02: Requests and Limits
- 2.03: Probes
- 2.04: Scheduling and placement

Security and RBAC

Service Accounts

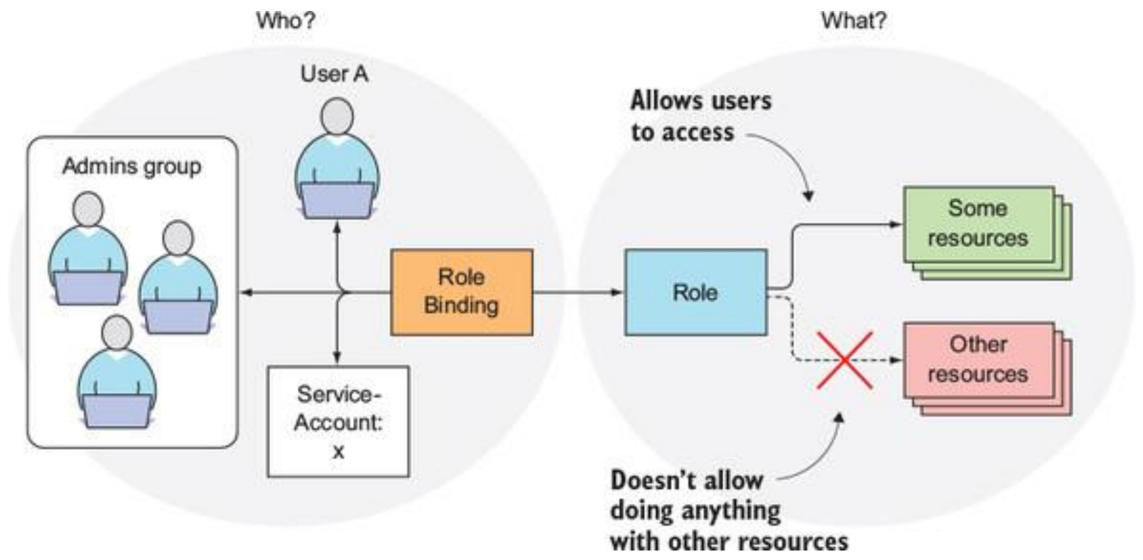
- Identity for pods (not humans)
- Every namespace has a 'default' ServiceAccount
- Pods automatically use 'default' unless specified
- Token mounted at /var/run/secrets/kubernetes.io/serviceaccount/
- Best practice: Create dedicated SA per application
- Don't use 'default' SA for production workloads

RBAC

- Role-Based Access Control
- Default: Deny everything
- Who? User, Group, or ServiceAccount
- What? Verbs: get, list, watch, create, update, delete
- Where? Resources: pods, deployments, secrets, configmaps
- Role: Permissions in ONE namespace

RBAC

- Role: Permissions in one namespace
- RoleBinding: Grants Role to subject
- ClusterRole: Permissions cluster-wide
- ClusterRoleBinding: Grants ClusterRole cluster-wide



Custom Resource Definition

Custom Resource Definition (CRD)

- Extend Kubernetes API with your own types
- Examples: Database, Certificate, Backup
- Stored and served like built-in resources

Finalizers

Finalizers

- Prevents resource deletion until cleanup is done
- Resource stuck in "Terminating" until finalizers removed

Helm

Helm

- Package manager for Kubernetes
- Chart: Package of K8s manifests + templates
- Release: Installed instance of a chart
- Repository: Collection of charts
- Values: Configuration for customization
- Templating with Go templates
- Versioning and rollback support

GitOps

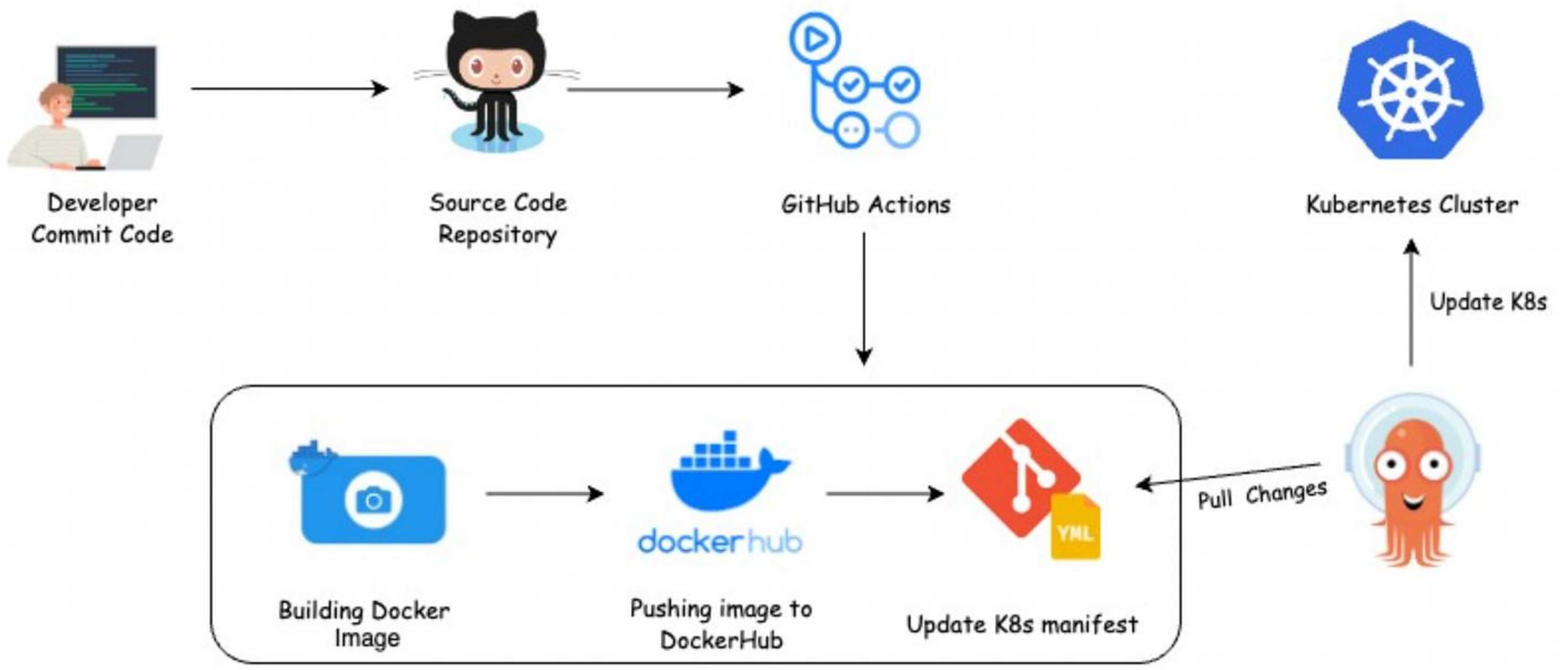
What is GitOps?

- Git = Single source of truth for infrastructure
- Desired state stored in Git repository
- Automated sync from Git to cluster
- All changes through Git (PRs, reviews, approvals)
- Benefits
  - Audit trail built-in (git history)
  - Rollback
  - Consistency across environments
  - No direct cluster access needed

ArgoCD

- Declarative, Git-based deployments
- CNCF project
- CLI and web UI
- Monitors Git repos for changes
- Syncs desired state to cluster automatically
- Multi-cluster support

ArgoCD


Developer Commit Code
Source Code Repository
GitHub Actions
Kubernetes Cluster
Update K8s
Building Docker Image
docker hub
Pushing image to DockerHub
YML
Update K8s manifest
Pull Changes

Labs

- 2.05: Service Accounts
- 2.06: Finalizers
- 2.07: Troubleshooting