K8s Workload

Table of Contents

Pod Metadata

Defined under `metadata`:

```yaml
metadata:
  name: my-pod
  labels:
    app: frontend
  annotations:
    description: "Test pod"
```

- `name`: Unique name for the Pod
- `labels`: Key-value pairs for selection & grouping
- `annotations`: Non-identifying metadata

Pod Spec Overview

All core configuration lives under `spec`:

```
spec:
  containers:
  - name: app
    image: nginx
```

Key fields include:

- `containers`: Main container definitions
- `volumes`: Shared storage between containers
- `restartPolicy`: Always, OnFailure, Never
- `nodeSelector`, `affinity`, `tolerations`: Scheduling rules

Containers Section

Inside `spec.containers[]`:

```yaml
containers:
- name: app
  image: nginx
  ports:
  - containerPort: 80
  env:
  - name: ENV
    value: prod
  resources:
    requests:
      cpu: "100m"
      memory: "64Mi"
```

- `image`: Container image
- `env`: Environment variables
- `ports`: Exposed ports
- `resources`: CPU/memory requests & limits
- `volumeMounts`: Attach shared volumes

Volumes

Defined under `spec.volumes`:

```
volumes:
- name: config-vol
  configMap:
    name: my-config
```

Mount inside a container:

```
volumeMounts:
- name: config-vol
  mountPath: /etc/config
```

Types include:

- `emptyDir`, `hostPath`, `configMap`, `secret`, `persistentVolumeClaim`, etc.

Pod Networking

All containers in a Pod:

- Share **same IP address**
- Communicate via `localhost`
- Can expose `ports` (for internal or service use)

Other fields:

- `hostname`: Custom Pod hostname
- `subdomain`: DNS within a `headless` Service

Scheduling & Placement

Control **where** Pods run:

```
nodeSelector:
  disktype: ssd

affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - ...
```

- `nodeSelector`: Simple key-value match
- `affinity`: Advanced node/pod affinity rules
- `tolerations`: Match tainted nodes
- `priorityClassName`: Influence scheduling priority

Security Context

Control container & Pod privileges:

```
securityContext:
  runAsUser: 1000
  runAsGroup: 3000
  fsGroup: 2000
```

Container-level:

```
containers:
- name: app
  securityContext:
    allowPrivilegeEscalation: false
```

- Use to enforce non-root access
- Enable SELinux, AppArmor profiles

Probes & Lifecycle

Health checks:

```
livenessProbe:
  httpGet:
    path: /health
    port: 8080
readinessProbe:
  exec:
    command: [ "cat", "/tmp/ready" ]
```

- `livenessProbe`: Restart if check fails
- `readinessProbe`: Control Service traffic
- `startupProbe`: For slow-starting apps

Lifecycle Hooks:

- `postStart`, `preStop`