# Exploring Pod Parameters

## `metadata`

Used for identification, organization, and management.

```yaml
metadata:
  name: web-server
  namespace: frontend
  labels:
    app: nginx
    tier: web
  annotations:
    description: "Serves static content using NGINX"
```

| Field | Purpose |
|---|---|
| `name` | Unique name of the pod (within the namespace) |
| `namespace` | Isolates the Pod logically |
| `labels` | Key-value pairs used for selecting Pods (e.g., in Services, Deployments) |
| `annotations` | Arbitrary non-identifying metadata used by tools or documentation |

# `spec.containers` (Deep Dive)

Most critical section — defines the actual workload.

```yaml
spec:
  containers:
  - name: nginx
    image: nginx:1.25
    ports:
      - containerPort: 80
    env:
      - name: ENVIRONMENT
        value: "production"
    resources:
      requests:
        cpu: "100m"
        memory: "128Mi"
      limits:
        cpu: "500m"
        memory: "512Mi"
    volumeMounts:
      - name: html-data
        mountPath: /usr/share/nginx/html
    readinessProbe:
      httpGet:
        path: /
        port: 80
      initialDelaySeconds: 5
      periodSeconds: 10
```

## Key Container Subfields

| Field | Description |
|---|---|
| `name` | Unique within the Pod |
| `image` | Docker image used to start the container |
| `command` | Overrides entrypoint (`ENTRYPOINT` in Docker) |
| `args` | Appends to command or overrides `CMD` |
| `ports` | Ports to expose |
| `env` | Set environment variables |
| `envFrom` | Bulk import from ConfigMaps or Secrets |
| `resources` | Requests/limits for CPU/memory |
| `volumeMounts` | Attach volumes to paths |
| `livenessProbe`, `readinessProbe`, `startupProbe` | Probes for health/lifecycle |
| `securityContext` | Set user ID, file permissions, capability drops |
| `lifecycle` | Run hooks like `preStop` or `postStart` |
| `stdin`, `tty` | For interactive containers |

## `spec.volumes`

Must match with `volumeMounts` in containers:

```
volumes:
  - name: html-data
    emptyDir: {}  # ephemeral
```

Common Volume Types:

- `emptyDir` – temporary scratch space
- `hostPath` – maps a host folder (security risk)
- `configMap`, `secret` – mount config or secret
- `persistentVolumeClaim` – durable storage (PVC)
- `nfs`, `projected`, `downwardAPI`, etc.

## `spec.restartPolicy`

```
restartPolicy: OnFailure
```

| Value | Description |
|---|---|
| `Always` (default) | Always restart containers |
| `OnFailure` | Only restart on non-zero exit |
| `Never` | Never restart, useful for debugging jobs |

### `spec.nodeSelector`

Constrains which nodes can run the Pod based on labels.

```yaml
nodeSelector:
  disktype: ssd
```

Use for simple affinity rules.

### `spec.tolerations`

Pods with tolerations can be scheduled to nodes with matching **taints**.

```yaml
tolerations:
- key: "dedicated"
  operator: "Equal"
  value: "gpu"
  effect: "NoSchedule"
```

Common effects:

- `NoSchedule`
- `PreferNoSchedule`
- `NoExecute`

## `spec.affinity`

More flexible than `nodeSelector`:

```yaml
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: disktype
              operator: In
              values: ["ssd"]
```

Also supports `podAffinity` and `podAntiAffinity`.

### `spec.serviceAccountName`

Specifies the identity the Pod will run as (important for RBAC):

```
serviceAccountName: app-access
```

### `spec.terminationGracePeriodSeconds`

How long to wait before forcibly killing a container:

```
terminationGracePeriodSeconds: 30
```

Defaults to `30`, but critical for graceful shutdown.

### `spec.imagePullSecrets`

Used to authenticate to private registries:

```yaml
imagePullSecrets:
- name: dockerhub-secret
```

### `spec.dnsPolicy`, `hostNetwork`, etc.

Advanced networking options:

```yaml
dnsPolicy: ClusterFirst
hostNetwork: false
hostPID: false
hostIPC: false
```

- `hostNetwork: true` puts container in host network namespace.
- `hostPID: true` shares process space.
- Use with care due to security risks.