Advanced pod concepts

Kubernetes Deep Dive

What’s in this module?

- Pod lifecycle
- Init Containers
- Multi-container pods
- Scheduling

Pod Lifecycle

Pod phases

Pending

- accepted by cluster

Running

- running on a node

Succeeded

- terminated with success

Failed

- terminated with failure

Unknown

- indicative of node communication failure

Container states

Waiting

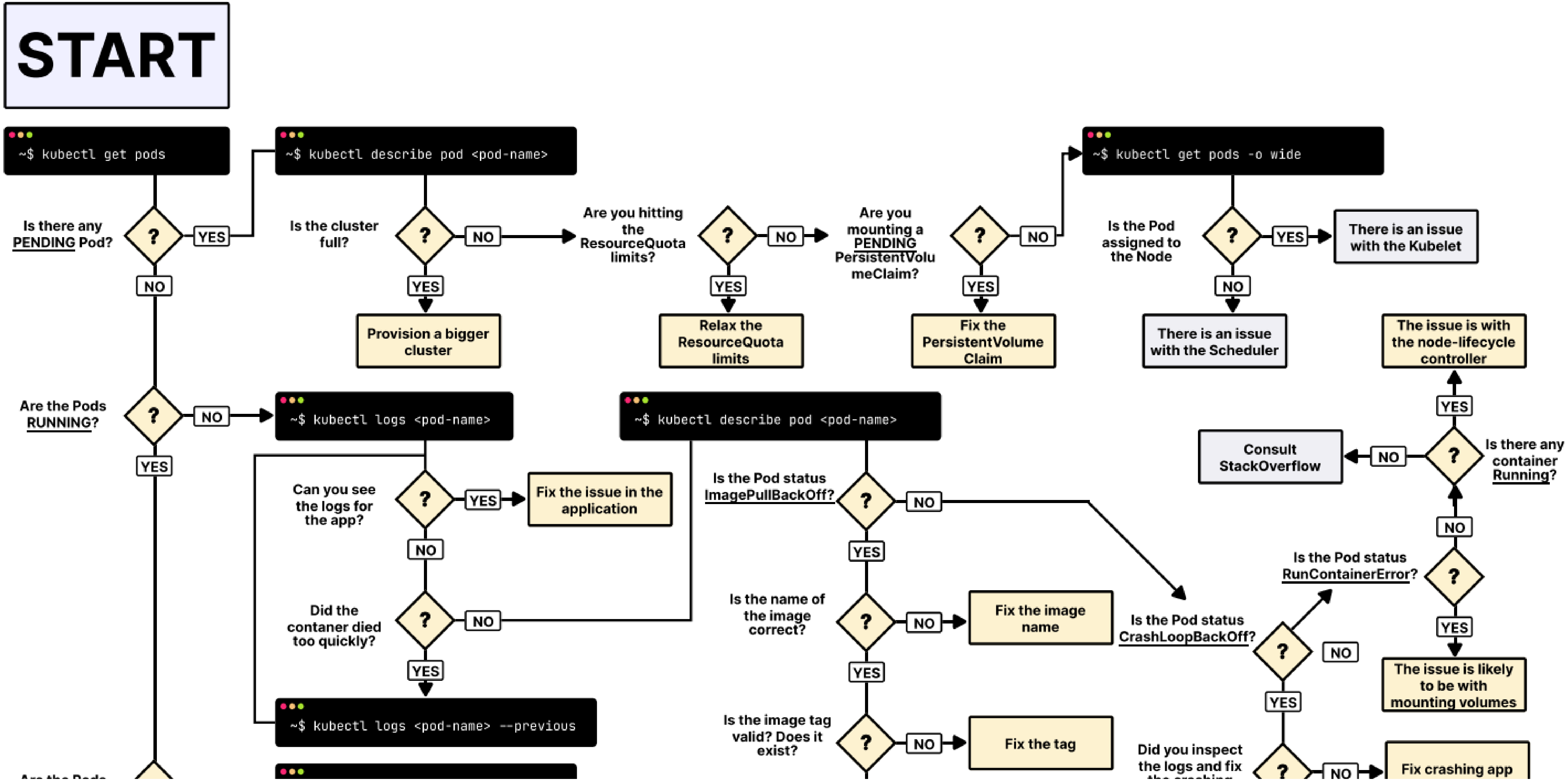
- still doing things to complete start up (e.g., pull)

Running

- container is running

Terminated

- ran to completion or failed



START

```
~$ kubectl get pods
```

Is there any PENDING Pod?

YES

```
~$ kubectl describe pod <pod-name>
```

Is the cluster full?

YES → Provision a bigger cluster

NO → Are you hitting the ResourceQuota limits?

YES → Relax the ResourceQuota limits

NO → Are you mounting a PENDING PersistentVolumeClaim?

YES → Fix the PersistentVolume Claim

NO

```
~$ kubectl get pods -o wide
```

Is the Pod assigned to the Node

YES → There is an issue with the Kubelet

NO → There is an issue with the Scheduler

NO

Are the Pods RUNNING?

NO

```
~$ kubectl logs <pod-name>
```

Can you see the logs for the app?

YES → Fix the issue in the application

NO → Did the contaner died too quickly?

YES

```
~$ kubectl logs <pod-name> --previous
```

NO

```
~$ kubectl describe pod <pod-name>
```

Is the Pod status ImagePullBackOff?

YES → Is the name of the image correct?

NO

Fix the image name

YES → Is the image tag valid? Does it exist?

NO

Fix the tag

NO → Is the Pod status CrashLoopBackOff?

YES → Did you inspect the logs and fix the crashing

NO → Fix crashing app

NO → Is the Pod status RunContainerError?

YES → The issue is likely to be with mounting volumes

NO → Is there any container Running?

YES → The issue is with the node-lifecycle controller

NO → Consult StackOverflow

YES

Are the Pods

Init containers

Init Containers

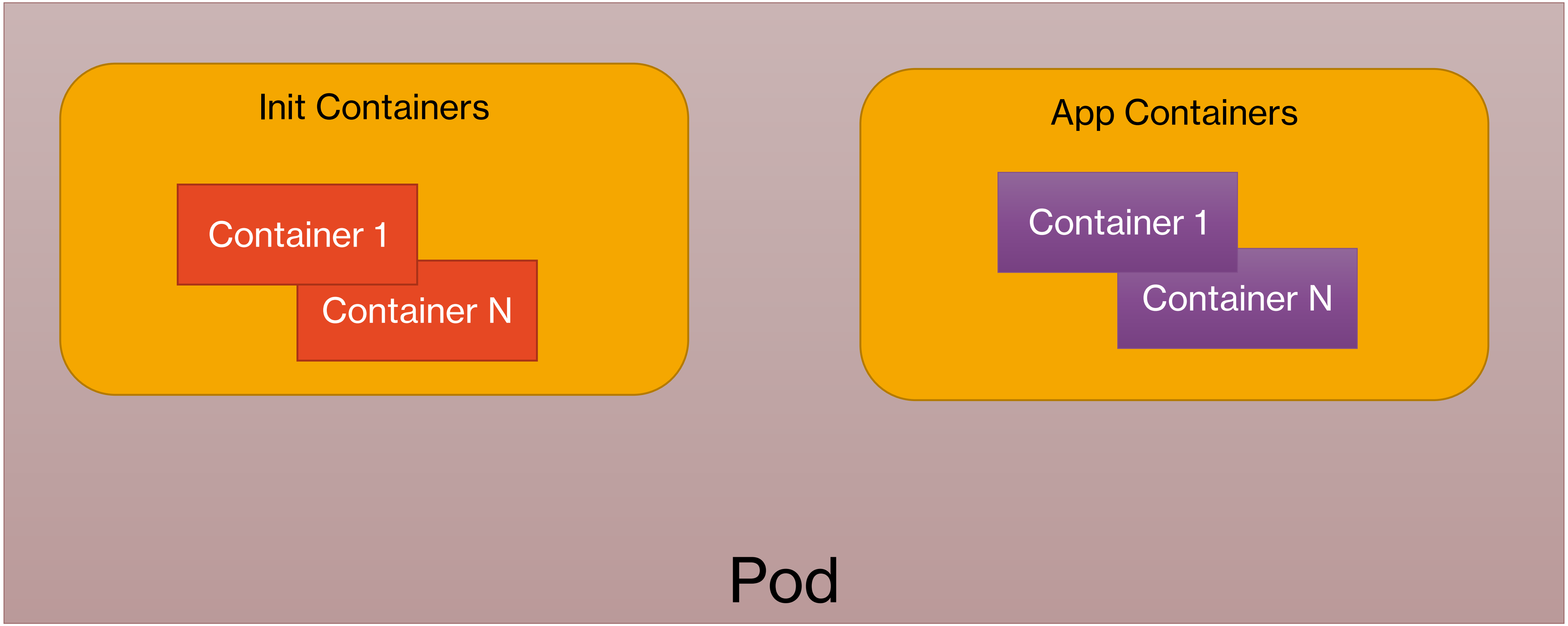
Init containers run before the application containers

run to completion

run one after the other

Each init container can have a separate image

run any tool you want without needing to add it to app image



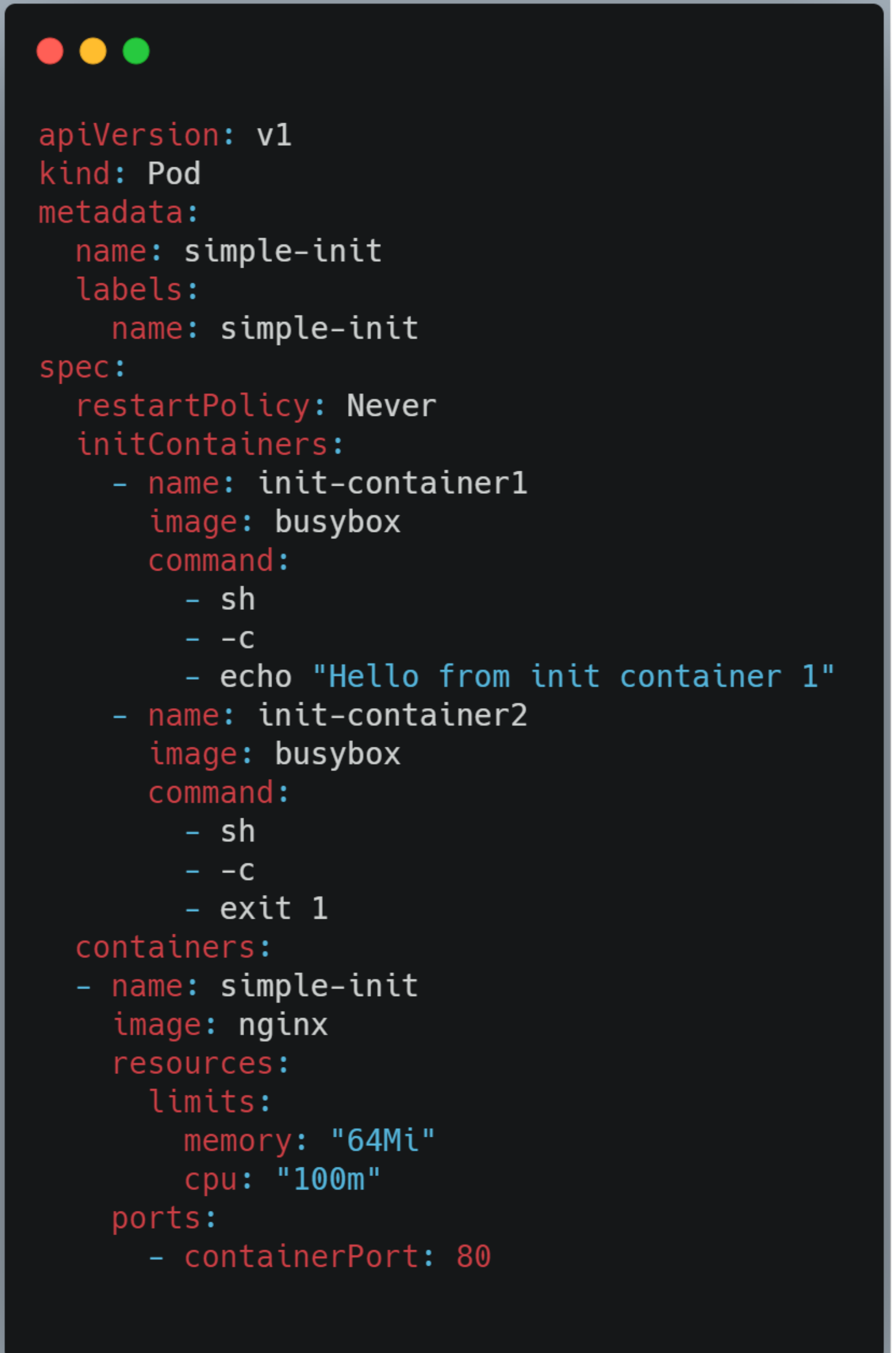
Init containers run sequentially
<u>before</u> the app containers run

Use cases

- Modify network rules
- Wait for events or sleep
- Register pods with remote systems
- Download files to a volume

  e.g. git clone

Example



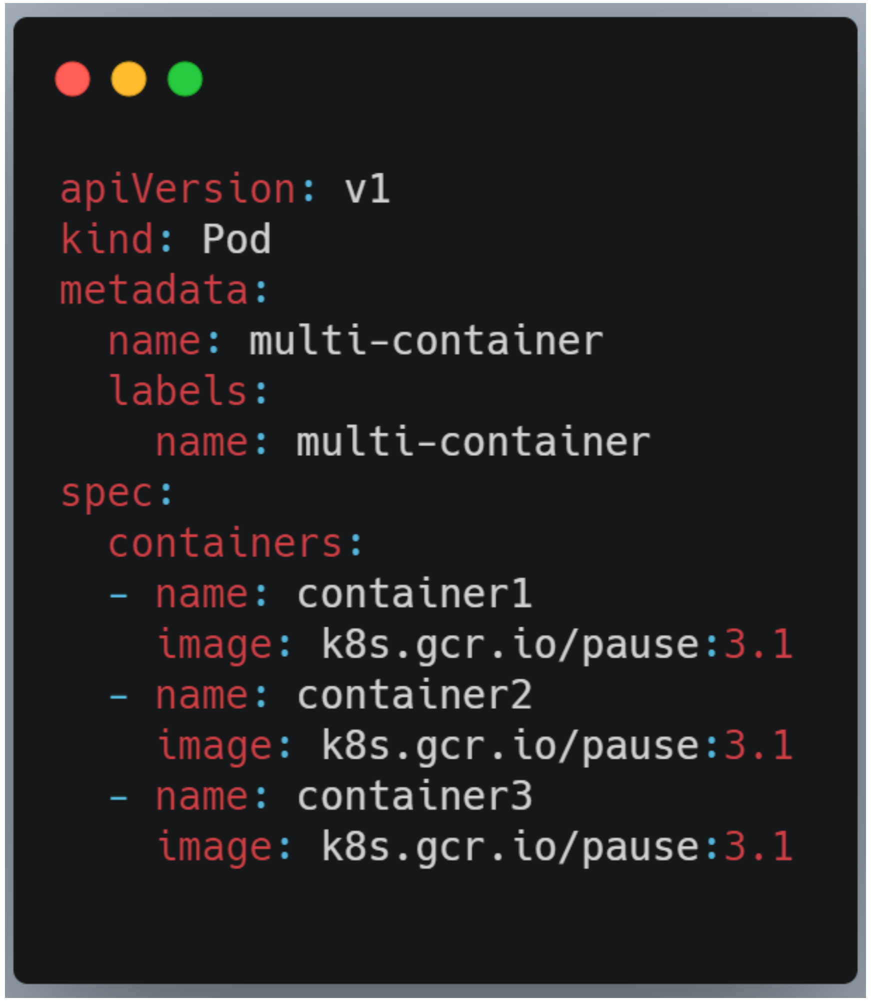
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-init
  labels:
    name: simple-init
spec:
  restartPolicy: Never
  initContainers:
    - name: init-container1
      image: busybox
      command:
        - sh
        - -c
        - echo "Hello from init container 1"
    - name: init-container2
      image: busybox
      command:
        - sh
        - -c
        - exit 1
  containers:
  - name: simple-init
    image: nginx
    resources:
      limits:
        memory: "64Mi"
        cpu: "100m"
    ports:
      - containerPort: 80
```

- Two init containers that run sequentially
- Second init container has non-zero exit code
- kubelet will restart the second init container unless restartPolicy=Never
- **Important:** set requests/limits in case namespace has ResourceQuota

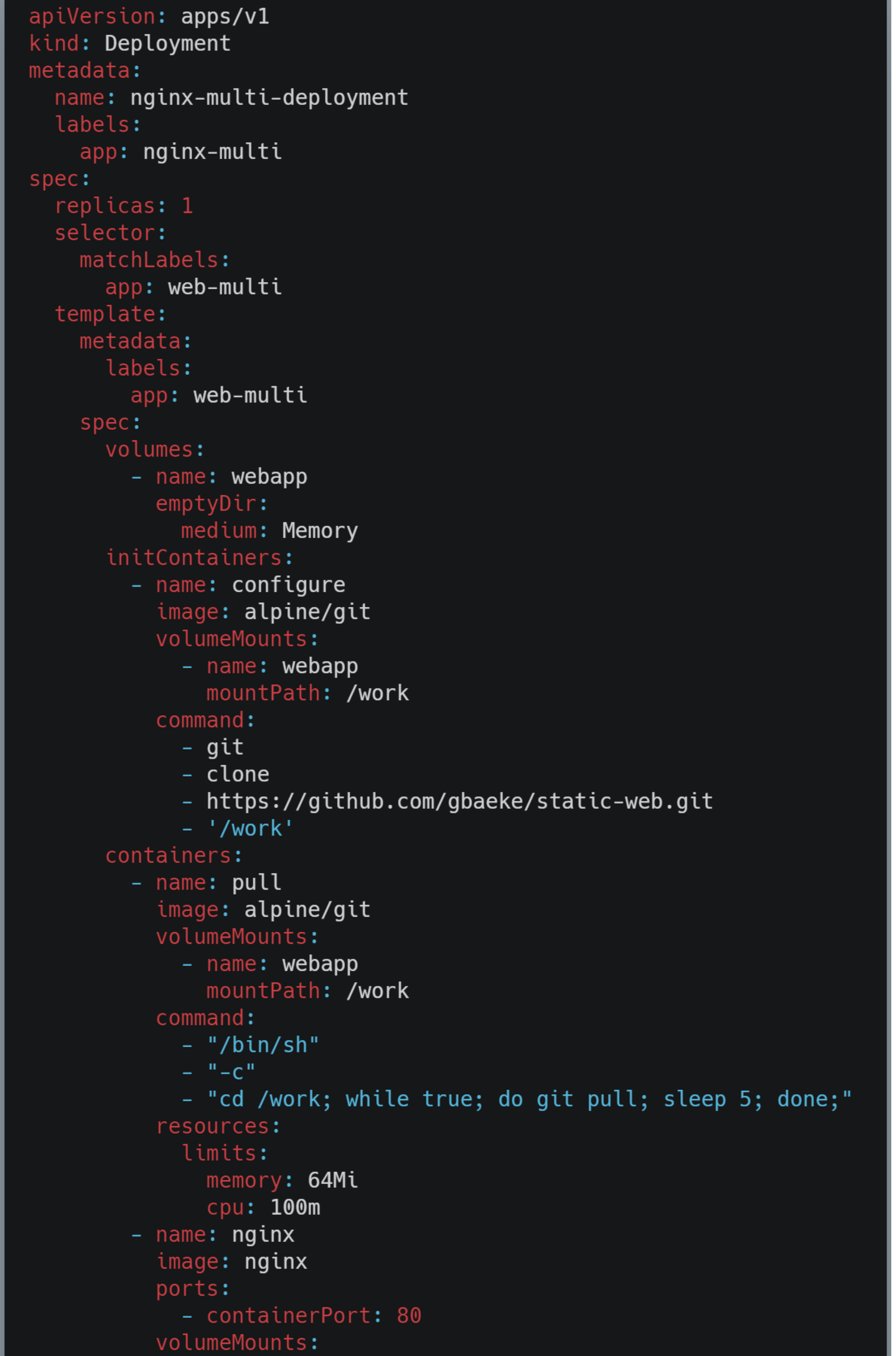
Multi container pods

Multi-container pods

- Run multiple co-located containers in a pod
- Containers run in parallel and can access shared resources
  - Storage: shared volumes
  - Network: shared network namespace (IP, network ports, …)



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: multi-container
  labels:
    name: multi-container
spec:
  containers:
  - name: container1
    image: k8s.gcr.io/pause:3.1
  - name: container2
    image: k8s.gcr.io/pause:3.1
  - name: container3
    image: k8s.gcr.io/pause:3.1
```



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-multi-deployment
  labels:
    app: nginx-multi
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-multi
  template:
    metadata:
      labels:
        app: web-multi
    spec:
      volumes:
        - name: webapp
          emptyDir:
            medium: Memory
      initContainers:
        - name: configure
          image: alpine/git
          volumeMounts:
            - name: webapp
              mountPath: /work
          command:
            - git
            - clone
            - https://github.com/gbaeke/static-web.git
            - '/work'
      containers:
        - name: pull
          image: alpine/git
          volumeMounts:
            - name: webapp
              mountPath: /work
          command:
            - "/bin/sh"
            - "-c"
            - "cd /work; while true; do git pull; sleep 5; done;"
          resources:
            limits:
              memory: 64Mi
              cpu: 100m
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
          volumeMounts:
```

Example

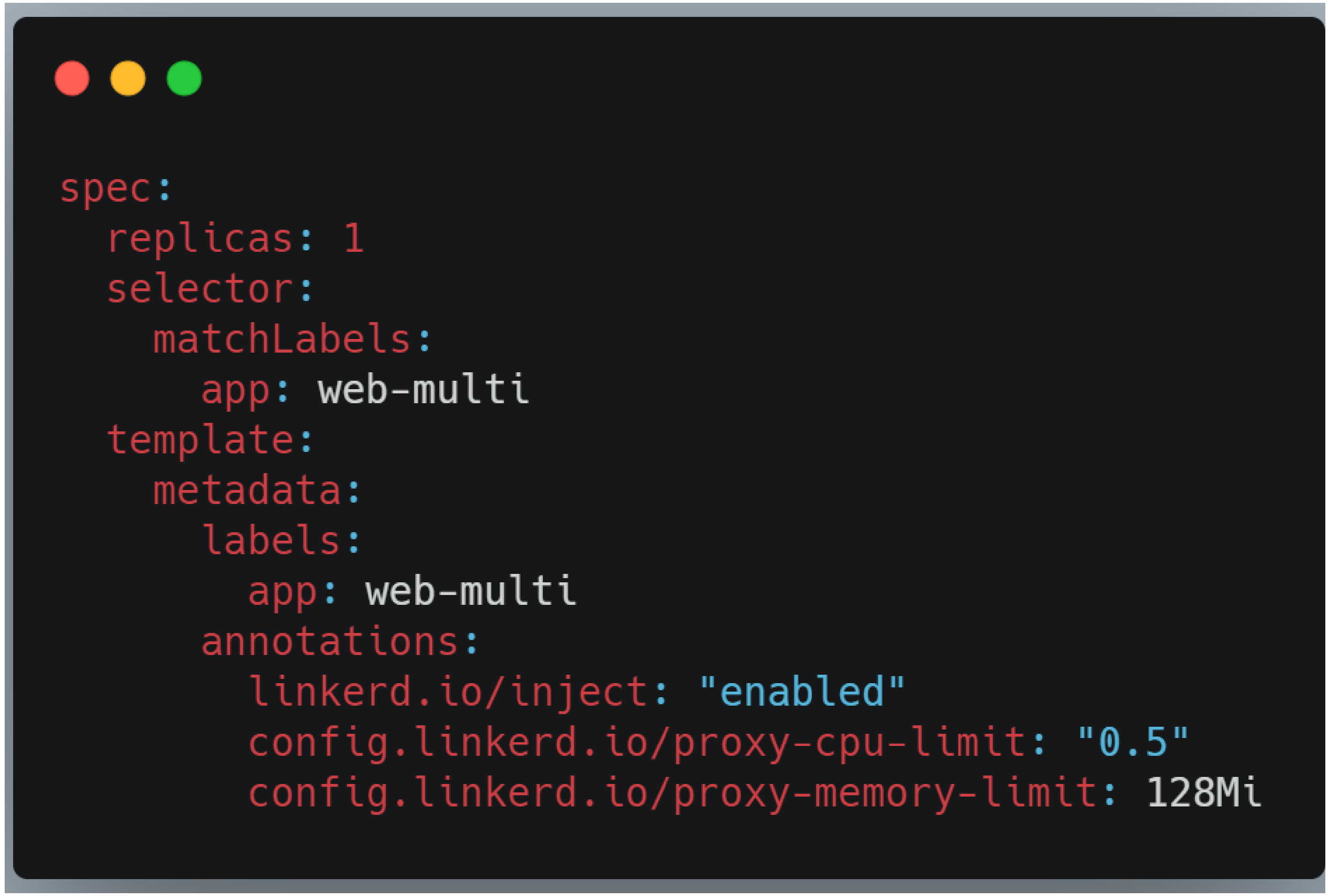
- Init container to clone the repo
- Container to run **git pull**
- Main container **nginx** to serve the contents of the git repo

Sidecars

Sidecar

- One or more containers that run along the main container
- Support the main container without changing it
- Important:
  - configure health checks
  - set appropriate requests/limits

Sidecar request and limits



```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-multi
  template:
    metadata:
      labels:
        app: web-multi
      annotations:
        linkerd.io/inject: "enabled"
        config.linkerd.io/proxy-cpu-limit: "0.5"
        config.linkerd.io/proxy-memory-limit: 128Mi
```

- Especially important in combination with ResourceQuota
- Sidecar injectors might need additional configuration

Ephemeral containers

Ephemeral Containers

- Alpha feature in Kubernetes 1.22
- Run a container in a pod temporarily
- Useful for interactive troubleshooting
  - Container crashed
  - Containers without debugging tools or shells (scratch & distroless)
- How?

→ **kubectl debug –it name --image busybox --target targetpod**

Disruptions

Disruptions

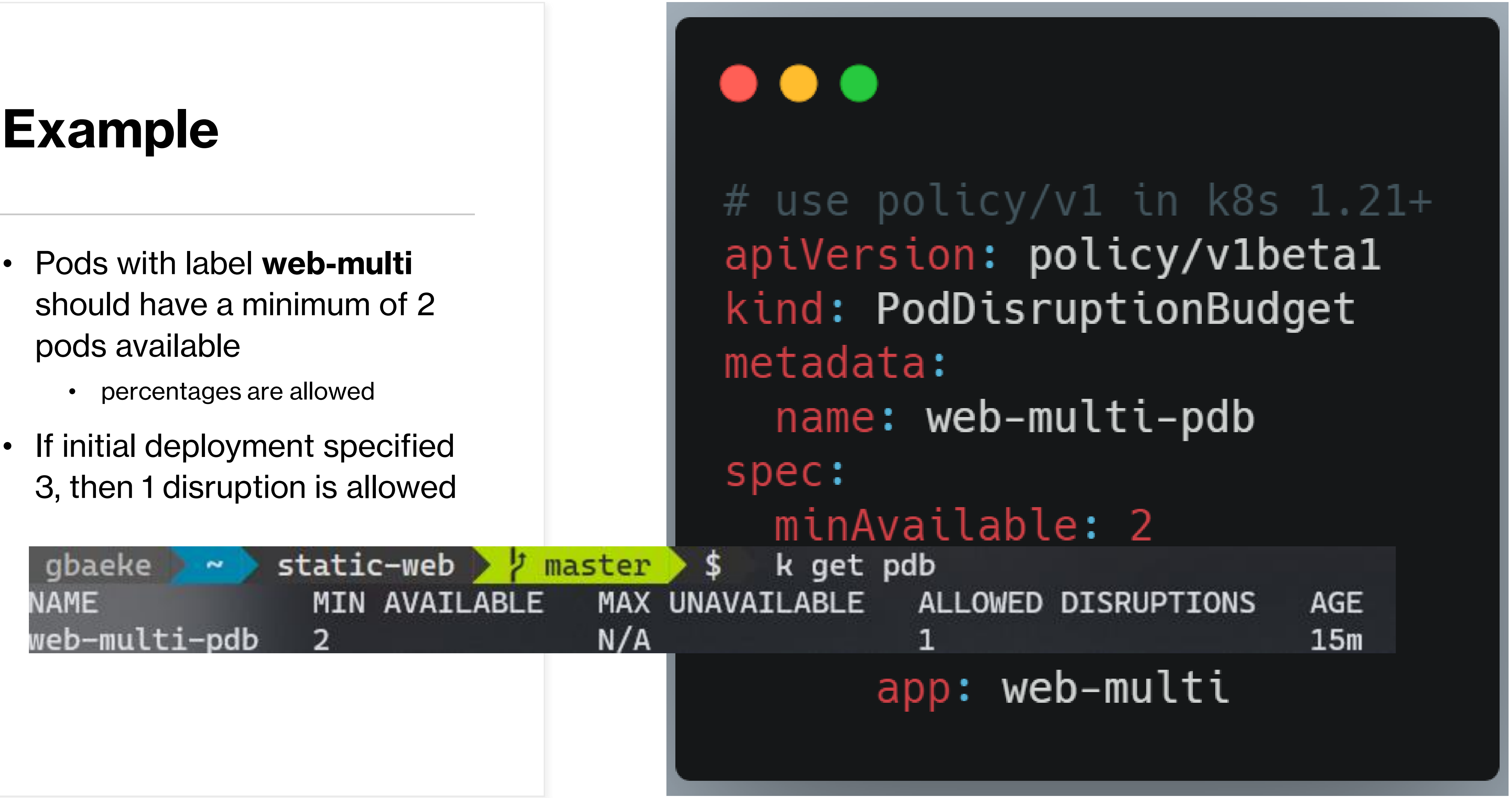
- Voluntary versus involuntary
- Use **PodDisruptionBudget** to limit the number of pods down from involuntary disruptions

  → set the number of replicas you tolerate having, versus the intended number

  → used by tools that use the eviction API (e.g. kubectl drain)
- Example:
  - Replicas in spec: 10
  - PodDisruptionBudget: 6
  - Eviction API will allow disruption of 4 pods at a time

Example

- Pods with label **web-multi** should have a minimum of 2 pods available
    - percentages are allowed
- If initial deployment specified 3, then 1 disruption is allowed



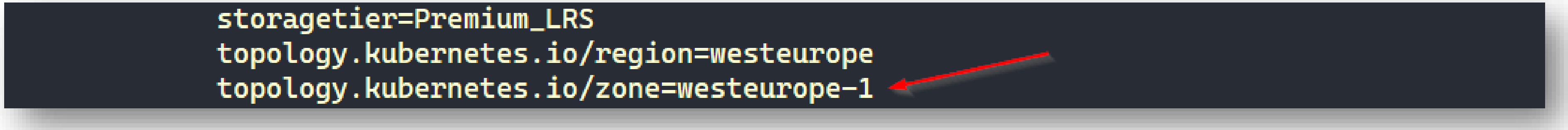
```yaml
# use policy/v1 in k8s 1.21+
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
metadata:
  name: web-multi-pdb
spec:
  minAvailable: 2
      app: web-multi
```

```
gbaeke  ~  static-web  master  $  k get pdb
NAME            MIN AVAILABLE   MAX UNAVAILABLE   ALLOWED DISRUPTIONS   AGE
web-multi-pdb   2               N/A               1                     15m
```

Topology spread constraints

Pod spreading

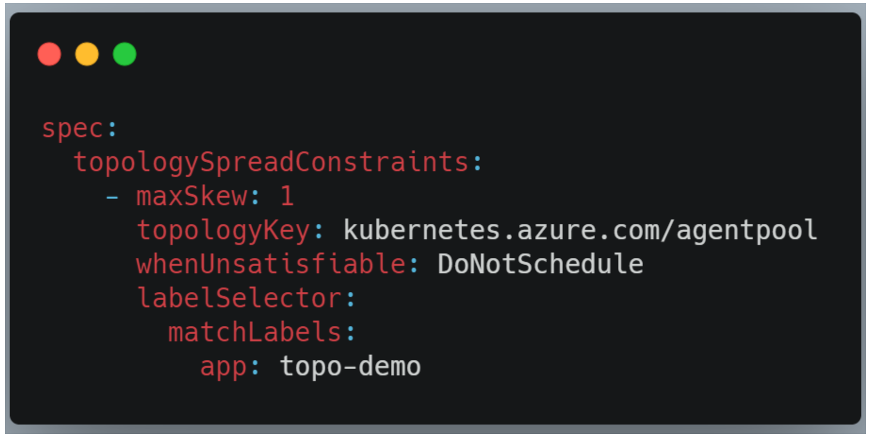
- Control how pods are spread across a target **topology** of the cluster
  - Regions, zones, nodes, …
- AKS uses the **topology.kubernetes.io/zone** label to:
  - Identify the zone of the node (if Availability Zones are used)
  - Distribute the pods across the zones



```
storagetier=Premium_LRS
topology.kubernetes.io/region=westeurope
topology.kubernetes.io/zone=westeurope-1
```

Topology Spread Constraints

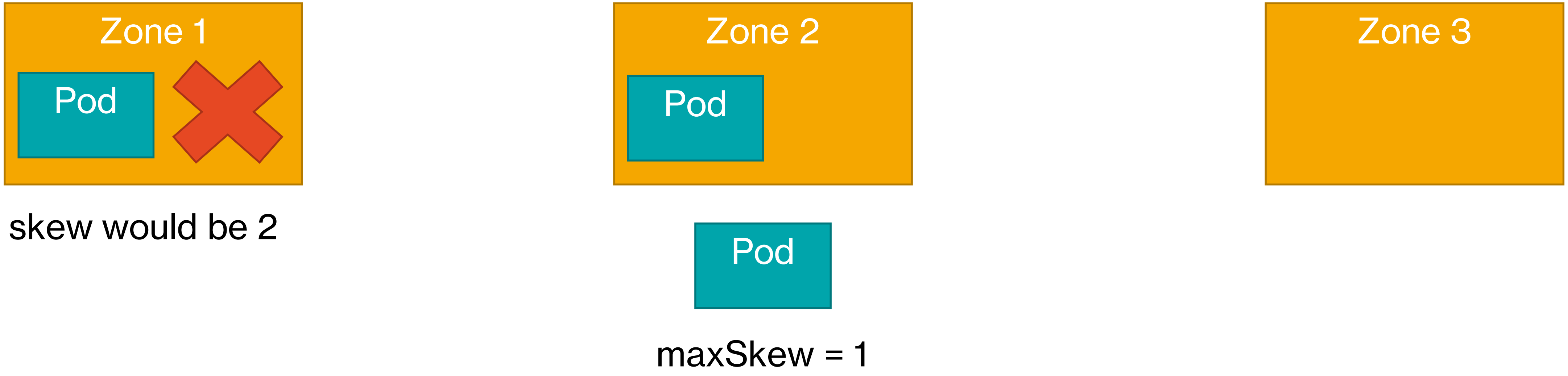
- New field in **pod spec**: topologySpreadConstraints
  - Applies to pods based on selector
- Topology is defined by a key of a node label
  - kubectl get nodes --show-labels
  - kubectl describe nodes
- Topology constraints at cluster level are **not possible** in a managed offering like AKS



```
spec:
  topologySpreadConstraints:
    - maxSkew: 1
      topologyKey: kubernetes.azure.com/agentpool
      whenUnsatisfiable: DoNotSchedule
      labelSelector:
        matchLabels:
          app: topo-demo
```

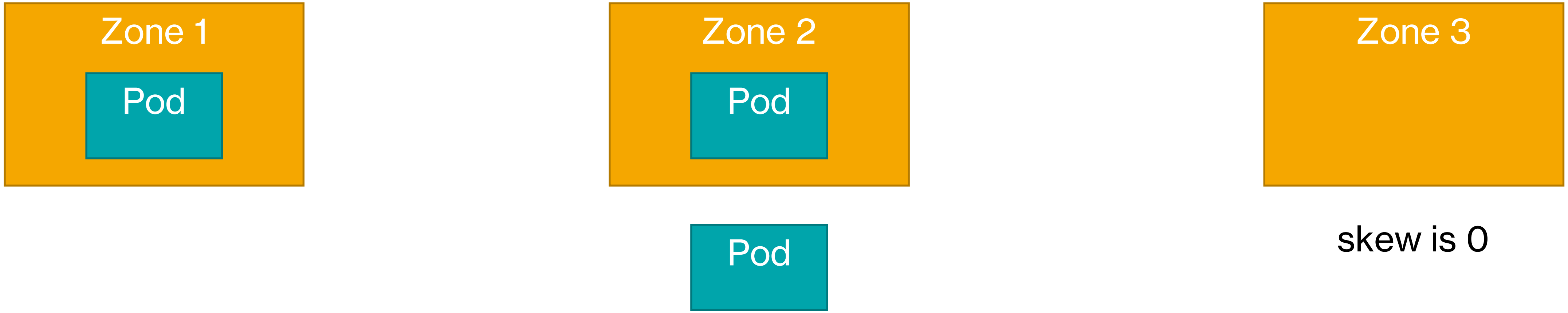
maxSkew

Maximum permitted difference between # of matching pods in the target topology and global minimum



maxSkew

Maximum permitted difference between # of matching pods in the target topology and global minimum



skew is 0

maxSkew = 1

Now onto...