KUBERMATIC

Cloud Native Rejekts [EU’22]

Using defaults for Deployments? Is it safe and sound?

KUBERMATIC

Koray Oksay

Site Reliability Engineer

koray@kubermatic.com

@korayoksay

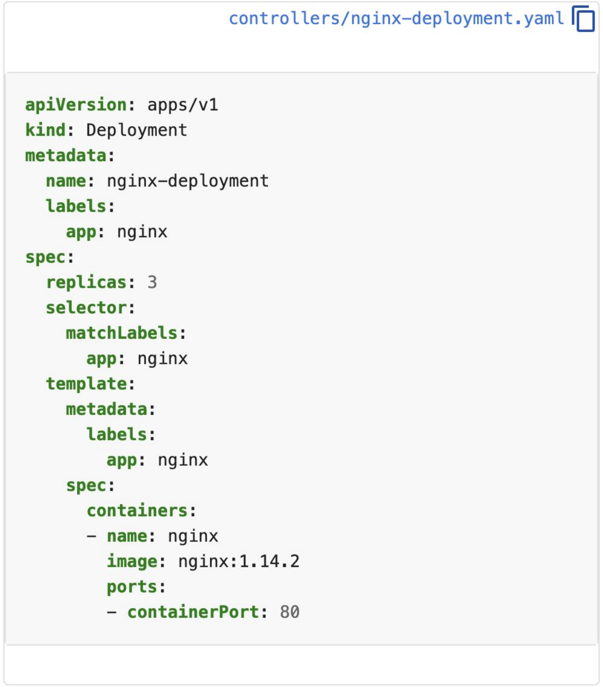
kubermatic/kubermatic

What Else?

- Consultant / Site Reliability Engineer
- Working remotely from Istanbul
- Interested in Linux, Kubernetes, and cloud technologies
- CKA | CKAD | CKS | RHCE

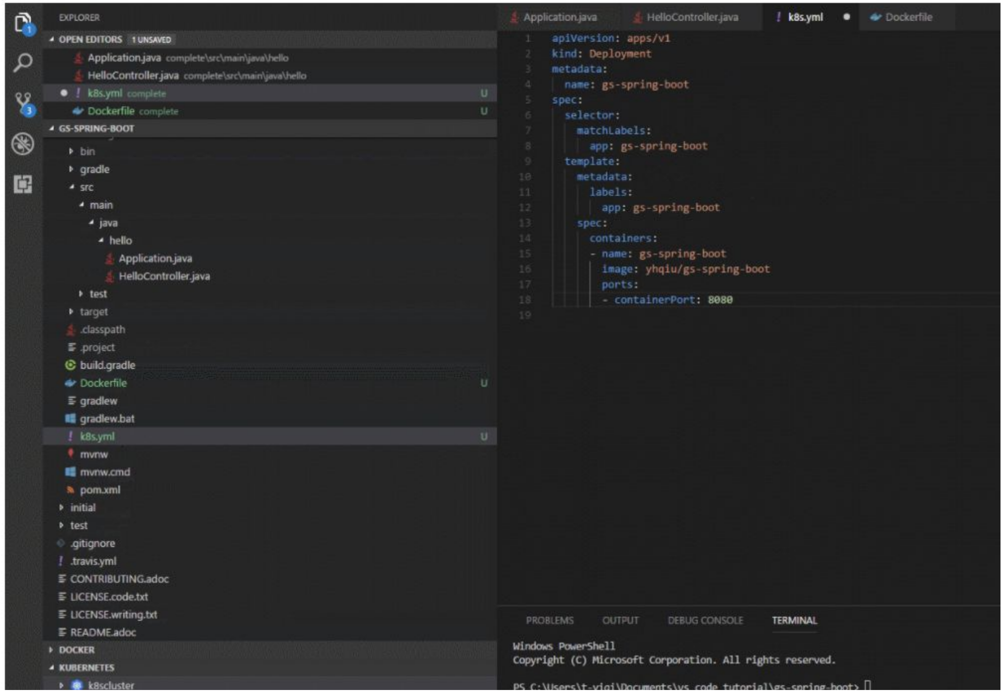
Kubernetes Deployment

- kubernetes.io/docs
- VS Code Kubernetes plugin



controllers/nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gs-spring-boot
spec:
  selector:
    matchLabels:
      app: gs-spring-boot
  template:
    metadata:
      labels:
        app: gs-spring-boot
    spec:
      containers:
      - name: gs-spring-boot
        image: yhqiu/gs-spring-boot
        ports:
        - containerPort: 8080
```

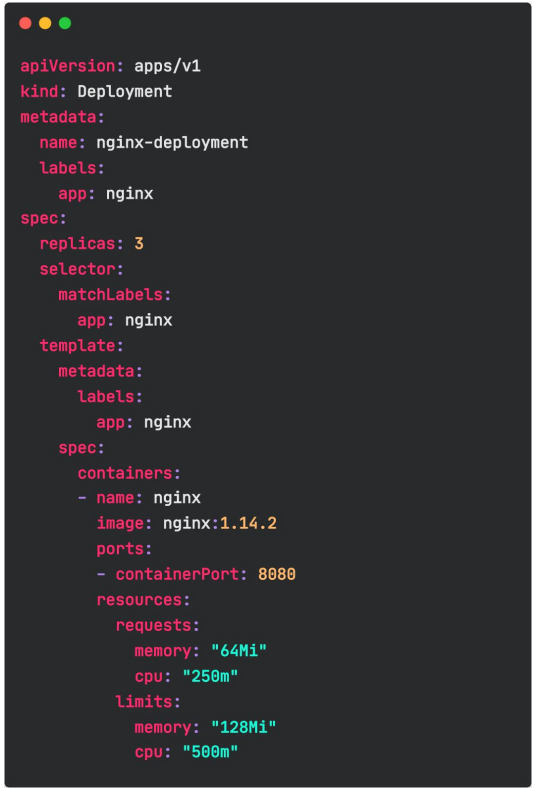
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\t-yiqi\Documents\vs code tutorial\gs-spring-boot>
```

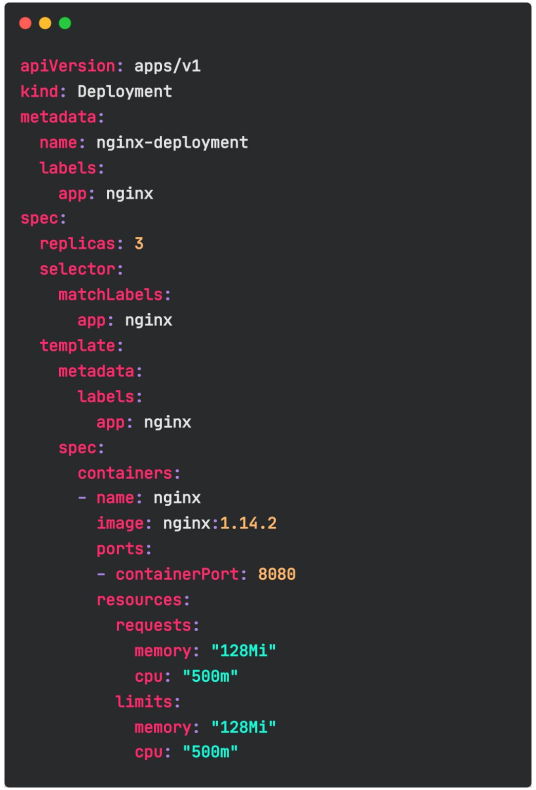
Quality of Service Classes

- Guaranteed:
  - Top priority
  - Pods will not be killed until they exceed their limits
- Burstable:
  - Minimal resource guarantee
  - Pods will not be killed until they exceed their limits and not any best-effort pods exist.
- **Best Effort:**
  - **Lowest priority**
  - **First to kill when the system is out of memory**

Add Resource Requests and Limits



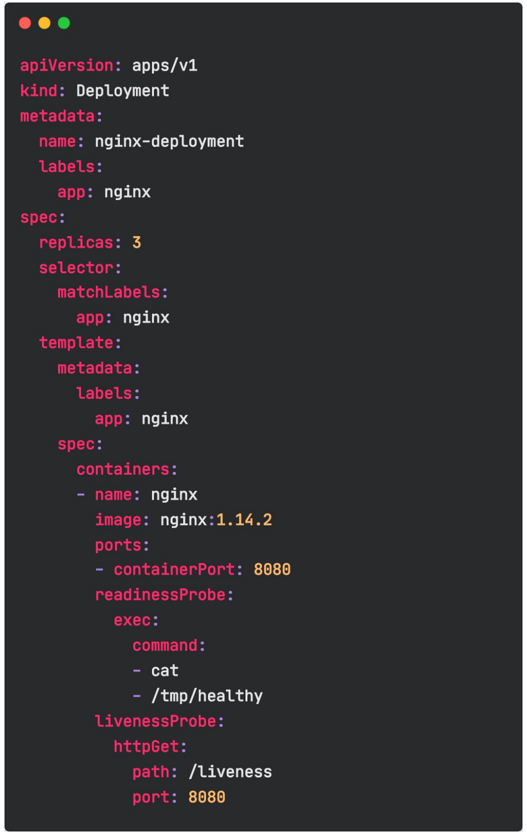
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 8080
        resources:
          requests:
            memory: "64Mi"
            cpu: "250m"
          limits:
            memory: "128Mi"
            cpu: "500m"
```



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 8080
        resources:
          requests:
            memory: "128Mi"
            cpu: "500m"
          limits:
            memory: "128Mi"
            cpu: "500m"
```

Define Probes

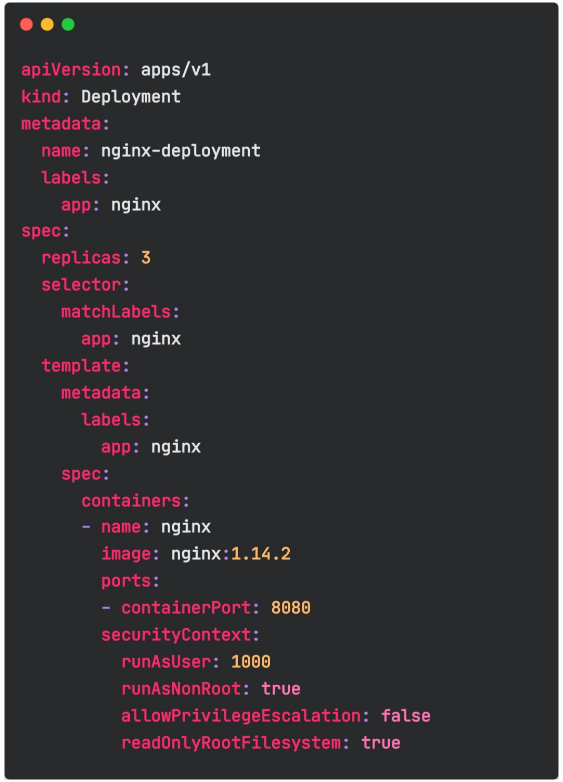
- Readiness:
  - Checks if the container is ready to accept traffic
- Liveness:
  - Checks if the application is running and makes progress



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 8080
        readinessProbe:
          exec:
            command:
            - cat
            - /tmp/healthy
        livenessProbe:
          httpGet:
            path: /liveness
            port: 8080
```

Add Security Context

- Make sure you run your container as non-root user (Container escape?).
- Have read-only root filesystem (do not change filesystem of your container).
- Process should not gain more privileges than its parent.



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 8080
        securityContext:
          runAsUser: 1000
          runAsNonRoot: true
          allowPrivilegeEscalation: false
          readOnlyRootFilesystem: true
```

KUBERMATIC

Terminate Your Pods Gracefully

The steps to terminate a pod:

- Pod state is changed to **`Terminating`** and **`preStop`** hook is executed (if defined)
- Send **`SIGTERM`** to PID 1 of all containers (This might be tricky!)
- Wait for 30 seconds (Default value for **`TerminationGracePeriodSeconds`**)
- Send **`SIGKILL`**

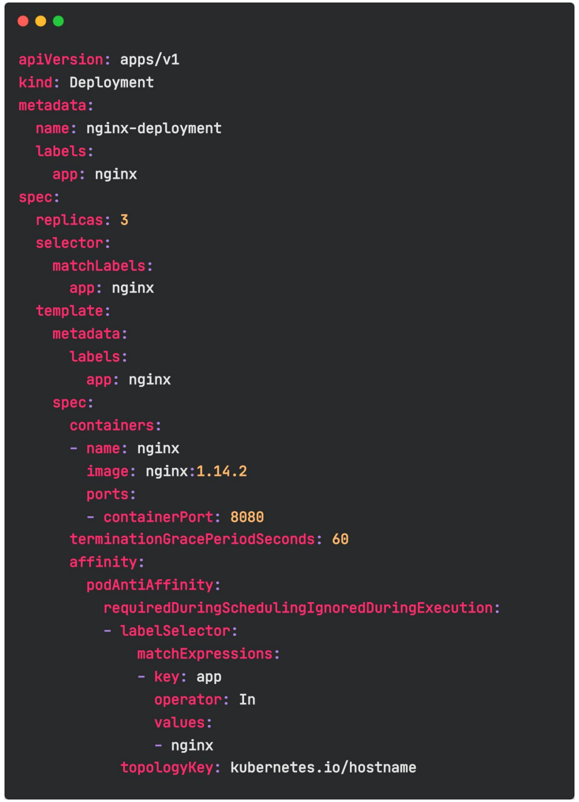
Tricky part: if your pods are always wait for 30 seconds (or other defined value)

- Check your Dockerfile for the CMD instruction:
  - **`CMD ./app.py`** → This starts a shell and runs the app (`/bin/sh -c ./app.py`)
  - **`CMD [“./app.py”]`** → This runs the app without a shell, so it gets **PID 1**

Distribute Your Pods

- By default, kube-scheduler runs the pods on any available node
- This could cause all your 3 replicas running on the same node!

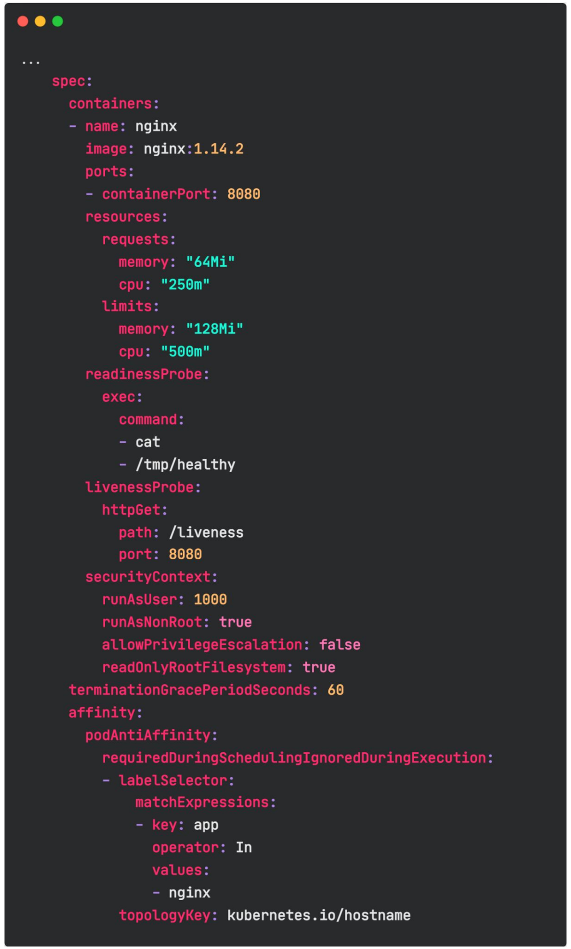
- **Ask the scheduler to run your pods on different nodes - a.k.a. AntiAffinity**



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 8080
      terminationGracePeriodSeconds: 60
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - nginx
            topologyKey: kubernetes.io/hostname
```

KUBERMATIC

All Combined



```
...
  spec:
    containers:
    - name: nginx
      image: nginx:1.14.2
      ports:
      - containerPort: 8080
      resources:
        requests:
          memory: "64Mi"
          cpu: "250m"
        limits:
          memory: "128Mi"
          cpu: "500m"
      readinessProbe:
        exec:
          command:
          - cat
          - /tmp/healthy
      livenessProbe:
        httpGet:
          path: /liveness
          port: 8080
      securityContext:
        runAsUser: 1000
        runAsNonRoot: true
        allowPrivilegeEscalation: false
        readOnlyRootFilesystem: true
    terminationGracePeriodSeconds: 60
    affinity:
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - nginx
          topologyKey: kubernetes.io/hostname
```

KUBERMATIC

Thank you!

koray@kubermatic.com

@kubermatic

kubermatic/kubermatic