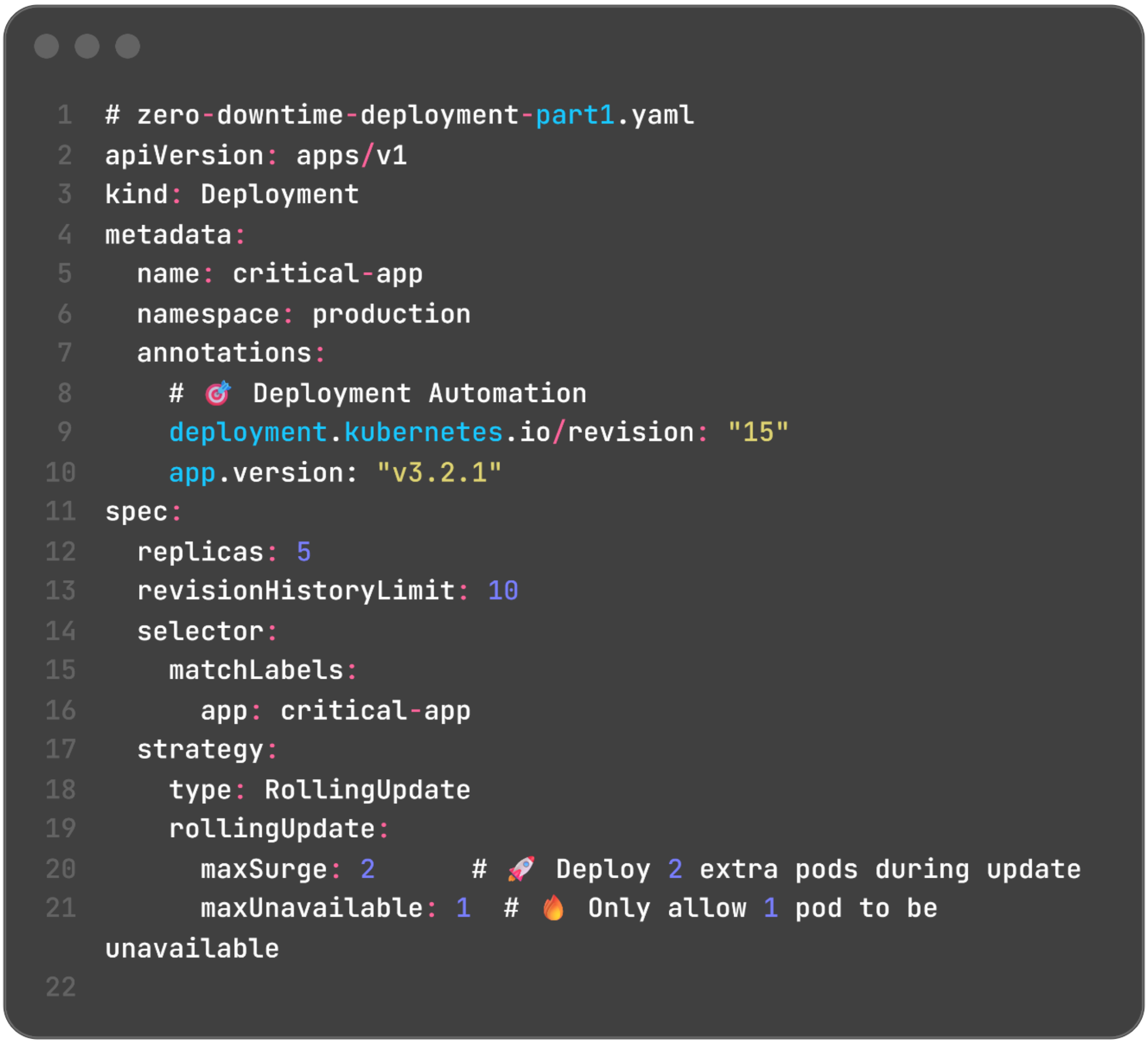
Sometimes, a single YAML tries to do *too much*.

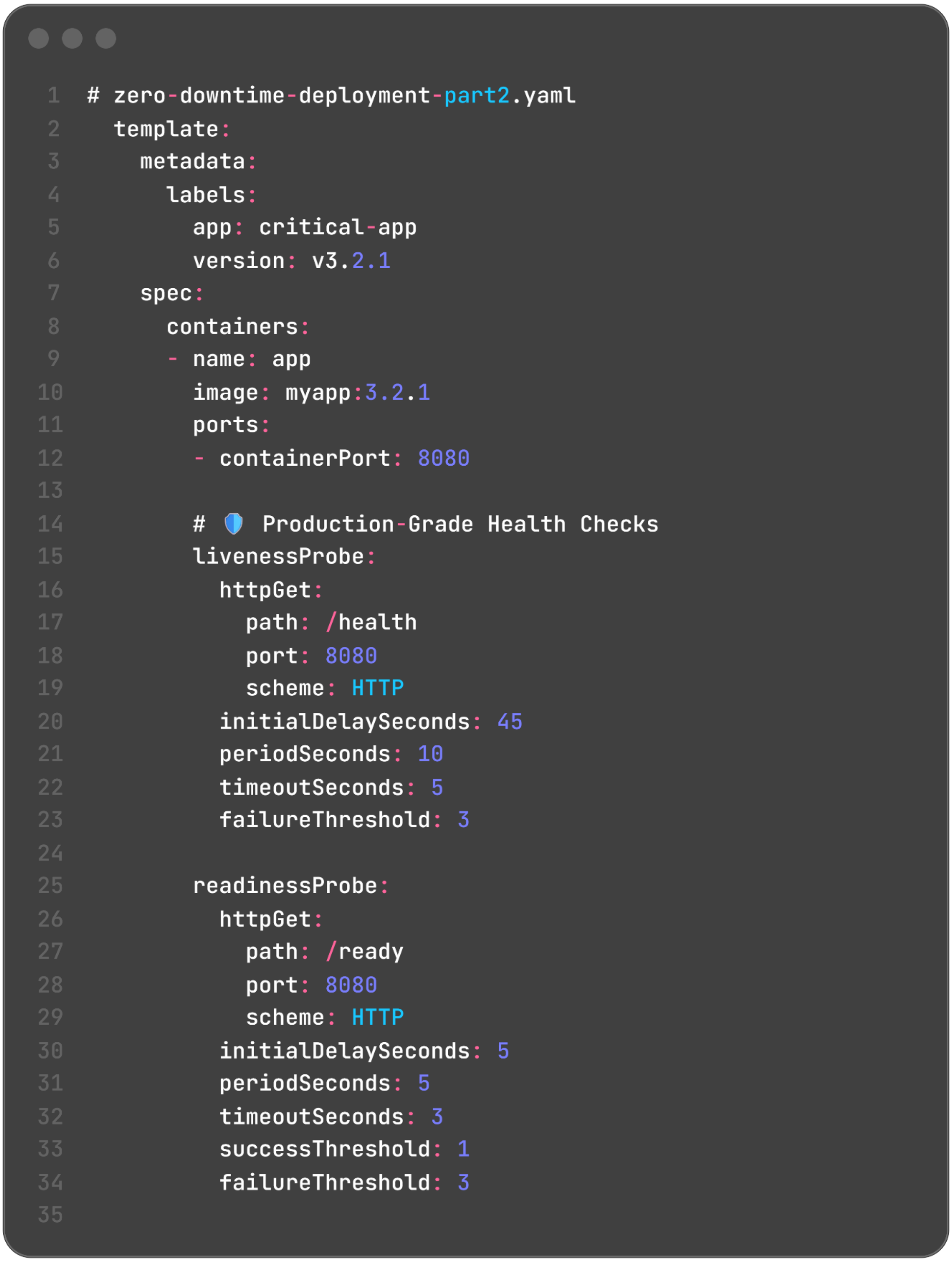
I posted this **full zero-downtime deployment script**, and while many appreciated the depth, several engineers told me:

“It’s too complex can you break it down into smaller, understandable parts?”

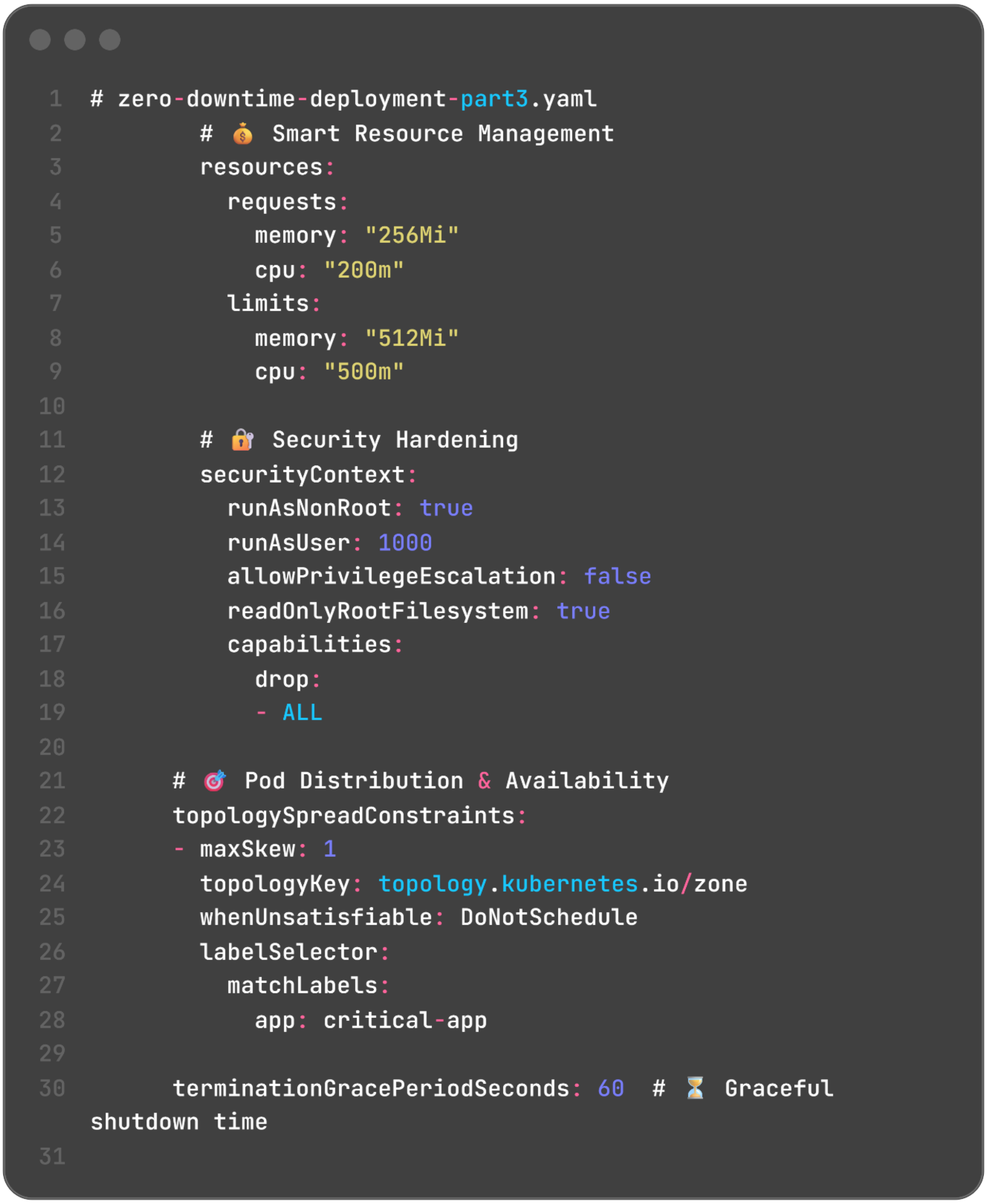
So here’s the full script first -



```yaml
# zero-downtime-deployment-part1.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: critical-app
  namespace: production
  annotations:
    # 🎯 Deployment Automation
    deployment.kubernetes.io/revision: "15"
    app.version: "v3.2.1"
spec:
  replicas: 5
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: critical-app
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 2      # 🚀 Deploy 2 extra pods during update
      maxUnavailable: 1  # 🔥 Only allow 1 pod to be unavailable

```



```yaml
# zero-downtime-deployment-part2.yaml
  template:
    metadata:
      labels:
        app: critical-app
        version: v3.2.1
    spec:
      containers:
      - name: app
        image: myapp:3.2.1
        ports:
        - containerPort: 8080

        # 🛡️ Production-Grade Health Checks
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 45
          periodSeconds: 10
          timeoutSeconds: 5
          failureThreshold: 3

        readinessProbe:
          httpGet:
            path: /ready
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 5
          periodSeconds: 5
          timeoutSeconds: 3
          successThreshold: 1
          failureThreshold: 3

```



```yaml
# zero-downtime-deployment-part3.yaml
        # 💰 Smart Resource Management
        resources:
          requests:
            memory: "256Mi"
            cpu: "200m"
          limits:
            memory: "512Mi"
            cpu: "500m"

        # 🔐 Security Hardening
        securityContext:
          runAsNonRoot: true
          runAsUser: 1000
          allowPrivilegeEscalation: false
          readOnlyRootFilesystem: true
          capabilities:
            drop:
            - ALL

      # 🎯 Pod Distribution & Availability
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: topology.kubernetes.io/zone
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: critical-app

      terminationGracePeriodSeconds: 60  # ⏳ Graceful shutdown time

```

Why This Was Hard for Many: Because this YAML combines *five different Kubernetes concepts* at once:

- Rolling updates
- Health checks
- Resource management
- Security context
- Pod topology

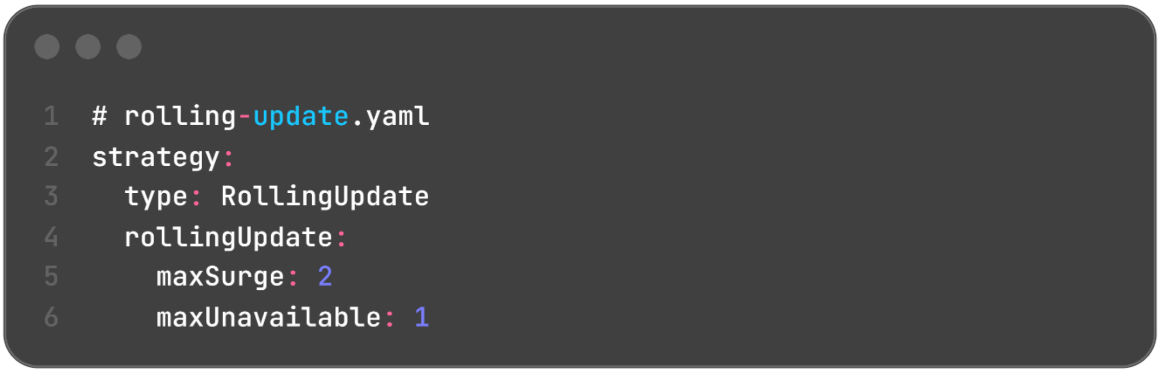
So, instead of one big "all-in-one" file, I broke it into a **Mini Toolkit** — each focused on one skill

---

Zero Downtime Toolkit (Mini Files)

---

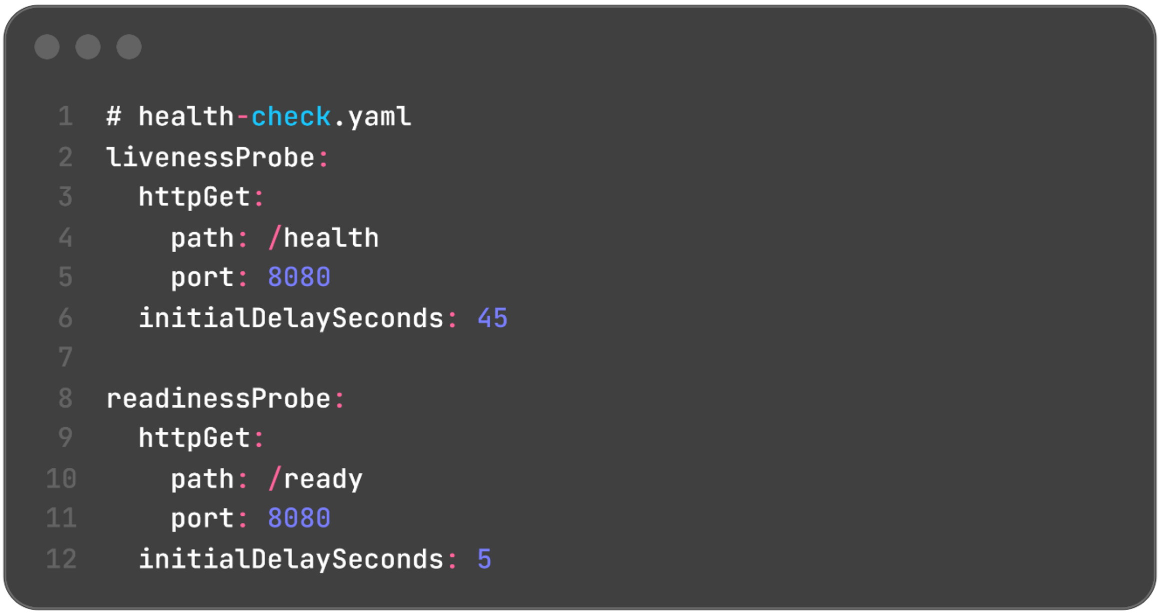
1. Rolling Update Tool *Smooth Deployments*



```yaml
# rolling-update.yaml
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 2
    maxUnavailable: 1
```

✅ Gradual rollout
✅ Zero downtime during upgrades

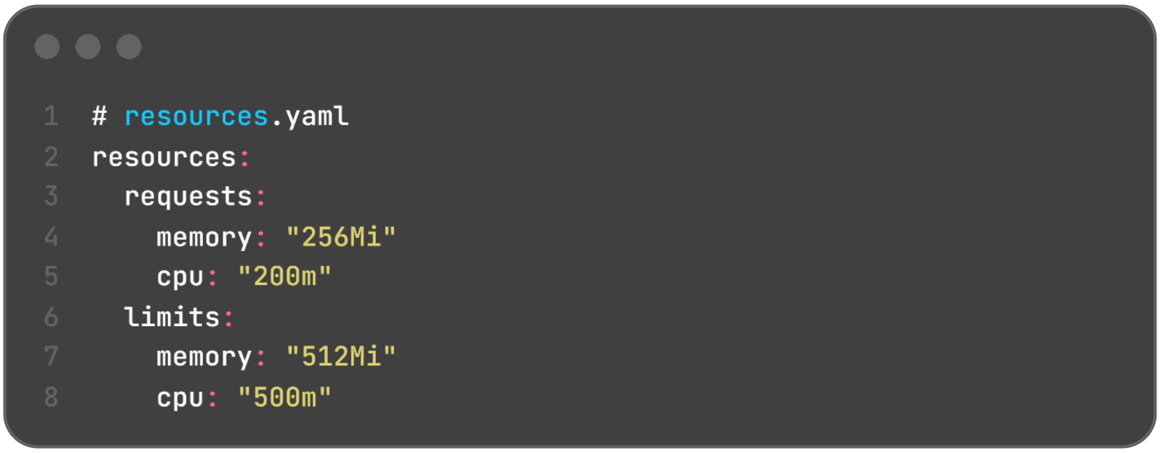
2. Health Check Tool *Self-Healing Apps*



```yaml
# health-check.yaml
livenessProbe:
  httpGet:
    path: /health
    port: 8080
  initialDelaySeconds: 45

readinessProbe:
  httpGet:
    path: /ready
    port: 8080
  initialDelaySeconds: 5
```

✅ Ensures the app is running
✅ Routes traffic only when ready

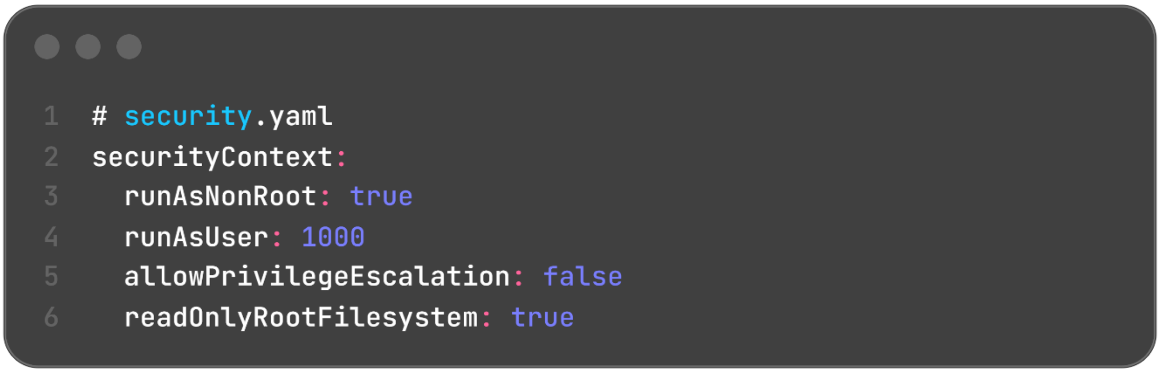
3. Resource Tool — *Smart Resource Allocation*



```yaml
# resources.yaml
resources:
  requests:
    memory: "256Mi"
    cpu: "200m"
  limits:
    memory: "512Mi"
    cpu: "500m"
```

✅ Prevents overuse
✅ Keeps cluster costs predictable

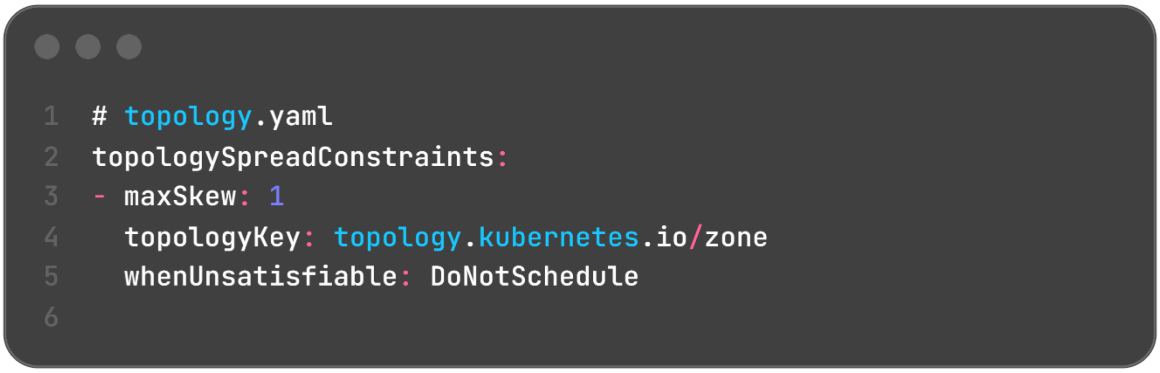
4. Security Tool *Hardened Pods*



```yaml
# security.yaml
securityContext:
  runAsNonRoot: true
  runAsUser: 1000
  allowPrivilegeEscalation: false
  readOnlyRootFilesystem: true
```

✅ Runs as non-root
✅ Locks down privileges

5. Topology Tool — *High Availability Spread*



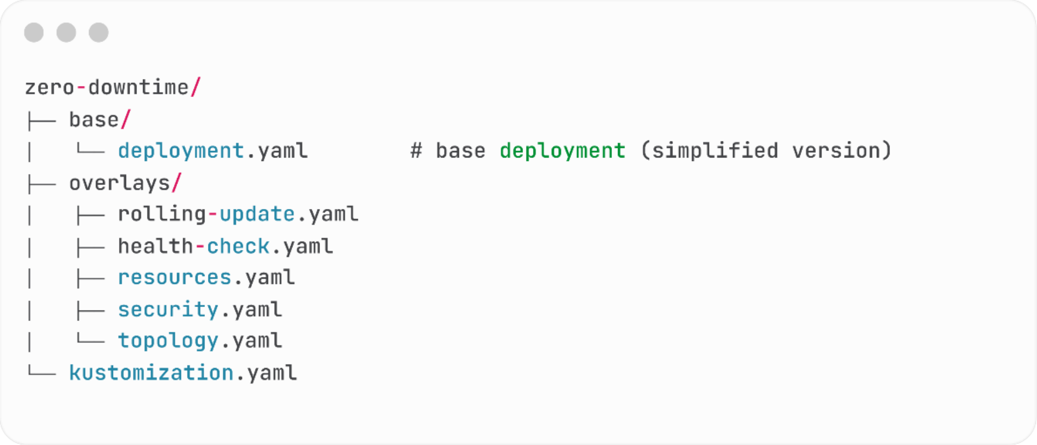
```yaml
# topology.yaml
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: topology.kubernetes.io/zone
  whenUnsatisfiable: DoNotSchedule

```

✅ Distributes pods across zones
✅ Avoids single-zone failure

How to Implement All Mini Files Together

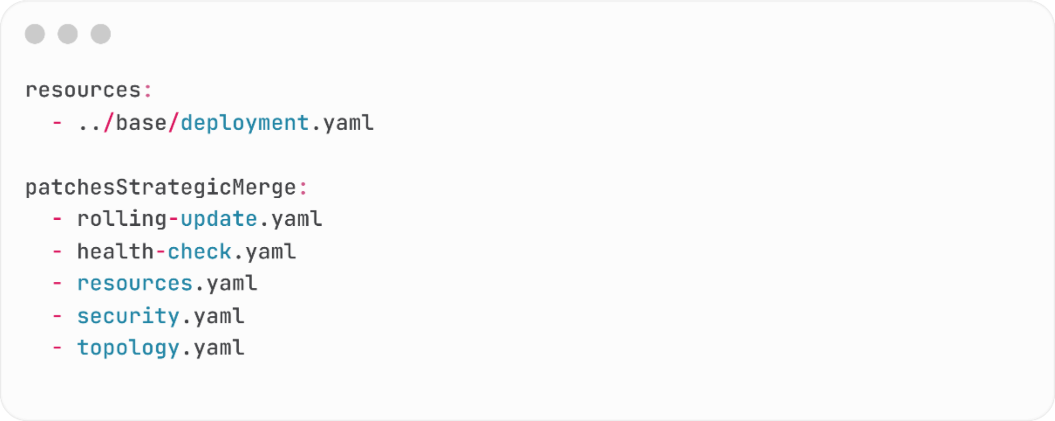
Now, here’s the fun part you can make these 5 files **work exactly like the full deployment** using **Kustomize**.

📁 Folder Structure



```
zero-downtime/
├── base/
|   └── deployment.yaml         # base deployment (simplified version)
├── overlays/
|   ├── rolling-update.yaml
|   ├── health-check.yaml
|   ├── resources.yaml
|   ├── security.yaml
|   └── topology.yaml
└── kustomization.yaml
```

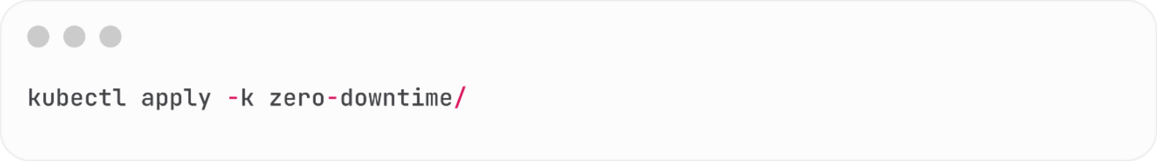
kustomization.yaml



```yaml
resources:
  - ../base/deployment.yaml

patchesStrategicMerge:
  - rolling-update.yaml
  - health-check.yaml
  - resources.yaml
  - security.yaml
  - topology.yaml
```

Apply All at Once

Run:



```
kubectl apply -k zero-downtime/
```