MÓDULO 14: KUBERNETES PARA ML

Deployments, Secrets, Resource Limits y Auto-scaling

Guía MLOps v5.0: Senior Edition | DuqueOM | Noviembre 2025

☸ MÓDULO 14: Kubernetes para ML

Orquestación Production-Ready

“Kubernetes no es complicado. Es complejo porque resuelve problemas complejos.”

Duración	Teoría	Práctica
6-7 horas	30%	70%

Lo Que Lograrás

1. **Desplegar** modelos ML en Kubernetes
2. **Configurar** secrets y configmaps
3. **Implementar** resource limits y auto-scaling
4. **Exponer** servicios con Ingress

14.1 Arquitectura K8s para ML

```
╔══════════════════════════════════════════════════════════════════════════════╗
║                    ARQUITECTURA KUBERNETES PARA ML                           ║
╠══════════════════════════════════════════════════════════════════════════════╣
║                                                                              ║
║   Internet                                                                   ║
║       │                                                                      ║
║       ▼                                                                      ║
║   ┌─────────────────┐                                                        ║
║   │    Ingress      │  (NGINX / Traefik)                                     ║
║   └────────┬────────┘                                                        ║
║            │                                                                 ║
║   ┌────────┴────────┐                                                        ║
║   │    Service      │  (ClusterIP / LoadBalancer)                            ║
║   └────────┬────────┘                                                        ║
║            │                                                                 ║
║   ┌────────┴────────┐                                                        ║
║   │   Deployment    │                                                        ║
║   │  ┌──────────┐   │                                                        ║
║   │  │  Pod 1   │   │  ← ReplicaSet manages                                  ║
║   │  │  Pod 2   │   │                                                        ║
║   │  │  Pod 3   │   │                                                        ║
║   │  └──────────┘   │                                                        ║
║   └─────────────────┘                                                        ║
║            │                                                                 ║
║   ┌────────┴────────┐                                                        ║
║   │ ConfigMap/Secret│  (Configuración y credenciales)                        ║
║   └─────────────────┘                                                        ║
║                                                                              ║
╚══════════════════════════════════════════════════════════════════════════════╝
```

14.2 Deployment Completo

```yaml
# k8s/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bankchurn-api
  labels:
    app: bankchurn-api
    version: v1
spec:
  replicas: 3
  selector:
    matchLabels:
      app: bankchurn-api
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    metadata:
      labels:
        app: bankchurn-api
        version: v1
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "8000"
        prometheus.io/path: "/metrics"
    spec:
      serviceAccountName: bankchurn-api

      # ════════════════════════════════════════════════════════════════════
      # SECURITY CONTEXT
      # ════════════════════════════════════════════════════════════════════
      securityContext:
        runAsNonRoot: true
        runAsUser: 1000
        fsGroup: 1000

      containers:
      - name: api
        image: ghcr.io/username/bankchurn:v1.2.3
        imagePullPolicy: Always

        ports:
        - containerPort: 8000
          name: http

        # ════════════════════════════════════════════════════════════════════
        # RESOURCE LIMITS (Crítico para ML)
        # ════════════════════════════════════════════════════════════════════
        resources:
          requests:
            memory: "512Mi"
            cpu: "250m"
          limits:
            memory: "1Gi"
            cpu: "500m"

        # ════════════════════════════════════════════════════════════════════
        # ENVIRONMENT VARIABLES
        # ════════════════════════════════════════════════════════════════════
        env:
        - name: LOG_LEVEL
          valueFrom:
            configMapKeyRef:
              name: bankchurn-config
              key: log_level
        - name: MODEL_VERSION
          value: "1.2.3"
        - name: MLFLOW_TRACKING_URI
          valueFrom:
            secretKeyRef:
              name: bankchurn-secrets
              key: mlflow_uri

        # ════════════════════════════════════════════════════════════════════
        # PROBES
        # ════════════════════════════════════════════════════════════════════
        readinessProbe:
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 10
          periodSeconds: 5
          failureThreshold: 3

        livenessProbe:
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 15
          periodSeconds: 10
          failureThreshold: 3

        startupProbe:
          httpGet:
            path: /health
            port: 8000
          failureThreshold: 30
          periodSeconds: 2

      # ════════════════════════════════════════════════════════════════════
      # TOPOLOGY SPREAD (Distribuir pods en nodos)
      # ════════════════════════════════════════════════════════════════════
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: ScheduleAnyway
        labelSelector:
          matchLabels:
            app: bankchurn-api
```

14.3 Secrets y ConfigMaps

```yaml
# k8s/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bankchurn-config
data:
  log_level: "INFO"
  prediction_threshold: "0.5"
  model_path: "/app/models/pipeline.pkl"
---
# k8s/secret.yaml (NUNCA commitear valores reales)
apiVersion: v1
kind: Secret
metadata:
  name: bankchurn-secrets
type: Opaque
stringData:
  mlflow_uri: "http://mlflow.mlops.svc.cluster.local:5000"
  # En producción, usar External Secrets Operator o Sealed Secrets
```

External Secrets (Producción)

```yaml
# k8s/external-secret.yaml
apiVersion: external-secrets.io/v1beta1
kind: ExternalSecret
metadata:
  name: bankchurn-secrets
spec:
  refreshInterval: 1h
  secretStoreRef:
    name: aws-secrets-manager
    kind: ClusterSecretStore
  target:
    name: bankchurn-secrets
  data:
  - secretKey: mlflow_uri
    remoteRef:
      key: bankchurn/prod
      property: mlflow_uri
```

14.4 Service e Ingress

```yaml
# k8s/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: bankchurn-api
  labels:
    app: bankchurn-api
spec:
  type: ClusterIP
  selector:
    app: bankchurn-api
  ports:
  - name: http
    port: 80
    targetPort: 8000
---
# k8s/ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankchurn-api
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
    cert-manager.io/cluster-issuer: letsencrypt-prod
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - api.bankchurn.example.com
    secretName: bankchurn-tls
  rules:
  - host: api.bankchurn.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: bankchurn-api
            port:
              number: 80
```

14.5 Horizontal Pod Autoscaler

```yaml
# k8s/hpa.yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: bankchurn-api
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: bankchurn-api
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 80
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 300
      policies:
      - type: Percent
        value: 10
        periodSeconds: 60
    scaleUp:
      stabilizationWindowSeconds: 0
      policies:
      - type: Percent
        value: 100
        periodSeconds: 15
```

14.6 Comandos Esenciales

```bash
# Aplicar manifiestos
kubectl apply -f k8s/

# Ver deployments
kubectl get deployments
kubectl describe deployment bankchurn-api

# Ver pods
kubectl get pods -l app=bankchurn-api
kubectl logs -f deployment/bankchurn-api

# Port forward para testing local
kubectl port-forward svc/bankchurn-api 8000:80

# Ver HPA
kubectl get hpa

# Debug pod
kubectl exec -it <pod-name> -- /bin/sh

# Ver eventos
kubectl get events --sort-by='.lastTimestamp'
```

14.7 Ejercicio: Deploy en Minikube

```bash
# Iniciar minikube
minikube start --memory 4096 --cpus 2

# Habilitar ingress
minikube addons enable ingress

# Aplicar manifiestos
kubectl apply -f k8s/

# Verificar
kubectl get all -l app=bankchurn-api

# Acceder
minikube service bankchurn-api --url
```

Checklist

```
DEPLOYMENT:
[ ] Deployment con replicas
[ ] Resource requests/limits
[ ] Probes configurados
[ ] Security context

CONFIGURACIÓN:
[ ] ConfigMap para config
[ ] Secrets para credenciales
[ ] Environment variables

NETWORKING:
[ ] Service tipo ClusterIP
[ ] Ingress con TLS
[ ] HPA configurado
```

Siguiente Paso

Con K8s configurado, es hora de implementar **observabilidad**.

Ir a Módulo 15: Observabilidad →

Módulo 14 completado. Tu API ahora es orquestada profesionalmente.