README

HPA Demo Project (Dynamic Scale Up / Scale Down) Goal Demonstrate `HorizontalPodAutoscaler` behavior in a practical way where: - replicas scale **up** automatically under CPU pressure - replicas scale **down** automatically after load is removed This project is intentionally simple and runnable on local labs (Minikube/KIND) or any Kubernetes cluster with metrics support. Project structure `text hpa-demo/ README.md manifests/ 00-namespace.yaml 01-deployment-service.yaml 02-hpa.yaml 03-load-generator.yaml` How this demo works App deployment: `hpa-web` (CPU request defined: `100m`) HPA target: average CPU utilization = `50%` HPA limits: min `1`, max `10` Load generator: continuously sends HTTP traffic to service `hpa-web-svc` Because HPA uses **utilization relative to CPU request**, setting `resources.requests.cpu` is required. Prerequisites Running Kubernetes cluster `kubectl` configured Metrics pipeline available (typically `metrics-server`) Check metrics quickly: `bash kubectl top nodes kubectl top pods -A` If these commands fail, install metrics-server first (method depends on your cluster distro). If metrics-server is not running (HPA shows `unknown` metrics) Check current status: `bash kubectl get apiservice v1beta1.metrics.k8s.io kubectl -n kube-system get deploy,pods | findstr metrics-server` Install metrics-server (upstream manifest): `bash kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml` For local/lab clusters (KIND, some Minikube setups), kubelet cert validation can block metrics collection. Add these args to `metrics-server` deployment if logs show TLS / x509 / scrape errors: `bash kubectl -n kube-system edit deployment metrics-server` Under `spec.template.spec.containers[0].args`, ensure these flags exist: `text --kubelet-insecure-tls --kubelet-preferred-address-types=InternalIP,Hostname,ExternalIP` Wait for rollout and verify: `bash kubectl -n kube-system rollout status deploy/metrics-server kubectl get apiservice v1beta1.metrics.k8s.io kubectl top nodes kubectl top pods -A` If `kubectl top` works, HPA should start receiving metrics within ~30-90 seconds. Step 1: Deploy app + HPA `bash kubectl apply -f hpa-demo/manifests/00-namespace.yaml kubectl apply -f hpa-demo/manifests/01-deployment-service.yaml kubectl apply -f hpa-demo/manifests/02-hpa.yaml` Verify baseline: `bash kubectl get deploy,hpa,pods -n hpa-demo kubectl describe hpa hpa-web -n hpa-demo` Expected initial state: - deployment replicas around `1` - HPA target shown as `50%` Step 2: Start load and observe scale up Apply load generator: `bash kubectl apply -f hpa-demo/manifests/03-load-generator.yaml` Watch autoscaling behavior: `bash kubectl get hpa hpa-web -n hpa-demo -w` In another terminal: `bash kubectl get deploy hpa-web -n hpa-demo -w` You should see: - `TARGETS` in HPA rise above `50%` - `REPLICAS` increase dynamically (e.g., 1 -> 2 -> 4 ... depending on cluster capacity) Step 3: Stop load and observe scale down Remove load generator: `bash kubectl delete -f hpa-demo/manifests/03-load-generator.yaml` Keep watching HPA/deployment for 1-3 minutes. You should see: - CPU utilization fall - replicas reduce gradually back toward `minReplicas: 1` This project sets `scaleDown.stabilizationWindowSeconds: 30` for quicker classroom/demo feedback. Useful debug commands `bash kubectl top pods -n hpa-demo kubectl describe hpa hpa-web -n hpa-demo kubectl get events -n hpa-demo --sort-by=.lastTimestamp kubectl logs -n hpa-demo deploy/load-generator --tail=20` Practical notes (important) **HPA is reactive, not instant** It takes some time to collect metrics and apply scaling decisions. **CPU request is mandatory for CPU utilization target** Without `resources.requests.cpu`, utilization-based CPU scaling does not work as intended. **Scale down is intentionally slower in real systems** Avoids oscillation/flapping. This demo uses a shorter window (`30s`) for visible lab behavior. **Cluster limits matter** If nodes are resource-constrained, HPA may request replicas that remain Pending. **HPA does not replace Cluster Autoscaler** HPA scales pods; Cluster Autoscaler scales nodes. Cleanup `bash kubectl delete -f hpa-demo/manifests/03-load-generator.yaml --ignore-not-found=true kubectl delete -f hpa-demo/manifests/02-hpa.yaml kubectl delete -f hpa-demo/manifests/01-deployment-service.yaml kubectl`

```
delete -f hpa-demo/manifests/00-namespace.yaml
```