README

Pod Updates Project (Deployment Strategies) Goal Demonstrate different Kubernetes Pod update strategies with runnable YAML manifests and practical notes. Strategies covered: - `RollingUpdate` - `Recreate` - `Blue/Green` - `Canary` (replica-based) Project structure `text pod-updates/ README.md strategy-notes.md manifests/ 00-namespace.yaml 01-rollingupdate.yaml 02-recreate.yaml 03-blue-green.yaml 04-canary.yaml` Prerequisites Running Kubernetes cluster `kubectl` configured and working Apply base namespace `bash kubectl apply -f pod-updates/manifests/00-namespace.yaml` 1) RollingUpdate demo Apply: `bash kubectl apply -f pod-updates/manifests/01-rollingupdate.yaml kubectl get all -n pod-updates` Trigger update: `bash kubectl set image deployment/rolling-web web=nginx:1.28 -n pod-updates kubectl rollout status deployment/rolling-web -n pod-updates kubectl rollout history deployment/rolling-web -n pod-updates` Rollback: `bash kubectl rollout undo deployment/rolling-web -n pod-updates` 2) Recreate demo Apply: `bash kubectl apply -f pod-updates/manifests/02-recreate.yaml` Trigger update (old Pods terminated first, then new Pods created): `bash kubectl set image deployment/recreate-web web=nginx:1.28 -n pod-updates kubectl rollout status deployment/recreate-web -n pod-updates` Rollback: `bash kubectl rollout undo deployment/recreate-web -n pod-updates` 3) Blue/Green demo Apply: `bash kubectl apply -f pod-updates/manifests/03-blue-green.yaml` Initial state: - `blue` is active (`bluegreen-active-svc` points to `color: blue`) - `green` starts with `replicas: 0` Prepare green: `bash kubectl scale deployment bluegreen-green --replicas=3 -n pod-updates kubectl get pods -l app=bluegreen-web -n pod-updates` Switch traffic to green: `bash kubectl patch service bluegreen-active-svc -n pod-updates -p '{"spec":{"selector":{"app":"bluegreen-web","color":"green"}}}'` Rollback switch (to blue): `bash kubectl patch service bluegreen-active-svc -n pod-updates -p '{"spec":{"selector":{"app":"bluegreen-web","color":"blue"}}}'` 4) Canary demo Apply: `bash kubectl apply -f pod-updates/manifests/04-canary.yaml` Initial split is approximate by Pod ratio: - stable replicas: 4 - canary replicas: 1 Increase canary weight: `bash kubectl scale deployment canary-web-canary --replicas=2 -n pod-updates kubectl get deploy -n pod-updates` Promote canary fully: `bash kubectl scale deployment canary-web-stable --replicas=0 -n pod-updates kubectl scale deployment canary-web-canary --replicas=5 -n pod-updates` Rollback canary quickly: `bash kubectl scale deployment canary-web-canary --replicas=0 -n pod-updates kubectl scale deployment canary-web-stable --replicas=4 -n pod-updates` Useful monitoring commands `bash kubectl get deploy,rs,pods,svc -n pod-updates kubectl rollout status deployment/rolling-web -n pod-updates kubectl rollout status deployment/recreate-web -n pod-updates kubectl describe service bluegreen-active-svc -n pod-updates` Cleanup `bash kubectl delete -f pod-updates/manifests/04-canary.yaml kubectl delete -f pod-updates/manifests/03-blue-green.yaml kubectl delete -f pod-updates/manifests/02-recreate.yaml kubectl delete -f pod-updates/manifests/01-rollingupdate.yaml kubectl delete -f pod-updates/manifests/00-namespace.yaml`