README

Kustomize E2E Demo Project Project name `kustomize-e2e-demo` Goal Demonstrate a complete end-to-end Kustomize workflow: - maintain reusable base manifests - apply environment-specific overlays (`dev`, `prod`) - preview generated YAML - deploy, verify, switch versions, and clean up Project structure `text kustomize-e2e-demo/ README.md installation.md base/ namespace.yaml configmap.yaml deployment.yaml service.yaml kustomization.yaml overlays/ dev/ kustomization.yaml patch-deployment.yaml prod/ kustomization.yaml patch-deployment.yaml` Prerequisites Kubernetes cluster running `kubectl` configured Kustomize available (`kubectl kustomize` or standalone `kustomize`) Installation steps: see `kustomize-e2e-demo/installation.md`. What base and overlays do Base (`base/`) Creates namespace `kustomize-demo` Deploys `nginx` app (`web`) Exposes service `web-svc` Adds base labels Defines default ConfigMap values Dev overlay (`overlays/dev`) Uses namespace `kustomize-dev` Adds suffix `-dev` to resource names Sets replicas to 2 Adds `environment: dev` label Overrides ConfigMap message for dev Prod overlay (`overlays/prod`) Uses namespace `kustomize-prod` Adds suffix `-prod` to resource names Sets replicas to 3 Adds `environment: prod` label Adds CPU/memory requests and limits Overrides ConfigMap message for prod Step 1: Preview generated manifests `bash kubectl kustomize kustomize-e2e-demo/overlays/dev kubectl kustomize kustomize-e2e-demo/overlays/prod` (Standalone alternative) `bash kustomize build kustomize-e2e-demo/overlays/dev kustomize build kustomize-e2e-demo/overlays/prod` Step 2: Deploy dev overlay `bash kubectl apply -k kustomize-e2e-demo/overlays/dev` Verify: `bash kubectl get ns | findstr kustomize-dev kubectl get all -n kustomize-dev kubectl get deploy,svc,cm -n kustomize-dev` Step 3: Deploy prod overlay `bash kubectl apply -k kustomize-e2e-demo/overlays/prod` Verify: `bash kubectl get ns | findstr kustomize-prod kubectl get all -n kustomize-prod kubectl get deploy,svc,cm -n kustomize-prod` Step 4: Inspect differences quickly `bash kubectl get deploy -n kustomize-dev -o wide kubectl get deploy -n kustomize-prod -o wide kubectl describe deploy web-dev -n kustomize-dev kubectl describe deploy web-prod -n kustomize-prod` You should see: - dev replicas = 2 - prod replicas = 3 - prod includes resource requests/limits Step 5: Roll out an image update (example) Update tag in overlay `kustomization.yaml` then re-apply: `bash kubectl apply -k kustomize-e2e-demo/overlays/dev kubectl rollout status deploy/web-dev -n kustomize-dev` And for prod: `bash kubectl apply -k kustomize-e2e-demo/overlays/prod kubectl rollout status deploy/web-prod -n kustomize-prod` Useful debug commands `bash kubectl get events -n kustomize-dev --sort-by=.lastTimestamp kubectl get events -n kustomize-prod --sort-by=.lastTimestamp kubectl get pods -n kustomize-dev -o wide kubectl get pods -n kustomize-prod -o wide` Cleanup `bash kubectl delete -k kustomize-e2e-demo/overlays/dev --ignore-not-found=true kubectl delete -k kustomize-e2e-demo/overlays/prod --ignore-not-found=true kubectl delete namespace kustomize-dev --ignore-not-found=true kubectl delete namespace kustomize-prod --ignore-not-found=true` Notes Use overlays for environment-specific configuration only. Keep shared resources and defaults in `base/`. Prefer `kubectl apply -k` for deployment workflows.