PLATFORM · TEMPLATE

GitOps and Progressive Delivery with *Automated* Rollback

KAANSYSTEMS.COM/LIBRARY/GITOPS-PROGRESSIVE-DELIVERY · AUGUST 8, 2026

ABOUT THIS TEMPLATE

GitOps with Argo CD and SLO-gated canary rollouts for regulated teams: repo structure, sync policies, and rollback that writes its own audit trail.

THE TEMPLATE

Deployment-repo layout, a production Rollout with a three-stage canary, and the AnalysisTemplate that gates it. Tune the thresholds to your SLOs, not the other way around.

```
deploy-repo/
├── bootstrap/
│   └── root-app.yaml                 # single Application pointing at argocd/
├── argocd/
│   ├── applicationset.yaml           # one Application per apps/*/overlays/*
│   └── appproject-prod.yaml          # RBAC: who may sync prod
├── platform/
│   ├── argo-rollouts/                # controller install, version pinned
│   ├── external-secrets/             # ESO + ClusterSecretStore (AWS SM)
│   └── monitoring/                   # Prometheus, ServiceMonitors
├── apps/
│   └── api/
│       ├── base/
│       │   ├── kustomization.yaml
│       │   ├── rollout.yaml          # below
│       │   ├── service-stable.yaml
│       │   ├── service-canary.yaml
│       │   ├── ingress.yaml          # ALB, weighted target groups
│       │   └── analysis-template.yaml # below
│       └── overlays/
│           ├── staging/
│           │   └── kustomization.yaml # replicas 2, shorter pauses
│           └── prod/
│               └── kustomization.yaml # replicas 6, image tag pinned here
└── docs/
    └── adr/
        └── 0007-canary-slo-thresholds.md
```

```yaml
# apps/api/base/rollout.yaml
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: api
spec:
  replicas: 6
  revisionHistoryLimit: 5
  selector:
    matchLabels:
      app: api
  strategy:
    canary:
      stableService: api-stable
      canaryService: api-canary
      trafficRouting:
        alb:
          ingress: api
          servicePort: 8080
      steps:
        - setWeight: 10
        - pause: { duration: 5m }
        - analysis:
            templates:
              - templateName: api-slo-check
            args:
              - name: service
                value: api-canary
        - setWeight: 50
        - pause: { duration: 10m }
        - analysis:
            templates:
              - templateName: api-slo-check
            args:
              - name: service
                value: api-canary
        - setWeight: 100
  template:
    metadata:
      labels:
        app: api
    spec:
      containers:
        - name: api
          # tag is set by CI via `kustomize edit set image` in overlays/prod
          image: 123456789012.dkr.ecr.us-east-1.amazonaws.com/api:sha-0000000
          ports:
            - containerPort: 8080
          readinessProbe:
            httpGet:
```

```
              path: /healthz
              port: 8080
            periodSeconds: 5
          resources:
            requests: { cpu: 250m, memory: 256Mi }
            limits: { memory: 512Mi }
```

```
# apps/api/base/analysis-template.yaml
apiVersion: argoproj.io/v1alpha1
kind: AnalysisTemplate
metadata:
  name: api-slo-check
spec:
  args:
    - name: service
  metrics:
    - name: success-rate
      interval: 60s
      count: 5
      # non-5xx share of requests must stay at or above 99%
      successCondition: len(result) > 0 && result[0] ⩾ 0.99
      failureLimit: 2           # tolerates two failed measurements; a third aborts
      provider:
        prometheus:
          address: http://prometheus.monitoring.svc.cluster.local:9090
          query: |
            sum(rate(http_requests_total{service="{{args.service}}",code!~"5.."}[2m]))
            /
            sum(rate(http_requests_total{service="{{args.service}}"}[2m]))
    - name: p99-latency
      interval: 60s
      count: 5
      successCondition: len(result) > 0 && result[0] ⩽ 0.5
      failureLimit: 2
      provider:
        prometheus:
          address: http://prometheus.monitoring.svc.cluster.local:9090
          query: |
            histogram_quantile(0.99,
              sum(rate(http_request_duration_seconds_bucket{service="{{args.service}}"}[2m])) by (
            )
```

On abort, Rollouts shifts all traffic to stable and scales the canary down; the revert PR in the deploy repo is the durable record of the rollback. Wire Argo CD Notifications to Slack so an aborted rollout pings the service owner, not just the platform channel.

—— HOW TO USE IT

Tag bumps are a machine's job

The commit that changes an image tag in the deploy repo should be opened by CI: a bot PR for prod, a direct commit for staging. Argo CD Image Updater can do this, but a 20-line CI job that runs `kustomize edit set image` and opens a PR is easier to audit and easier to debug. Humans review promotions; they should not hand-type tags, because hand-typed tags are where typos meet production. The bot PR also gives your prod promotion the same approval trail as every other change, which is the entire premise.

Secrets never enter the deploy repo

A GitOps repo is read widely by design: every engineer, CI, the controller. Kubernetes Secrets, even base64-encoded, do not belong in it. Run External Secrets Operator pointed at AWS Secrets Manager and commit only ExternalSecret references. Secret values then carry their own audit trail in CloudTrail, separate from config history, which is the separation of duties your auditor expects to find anyway. If a secret does land in git, treat it as leaked and rotate; history rewrites are not remediation.

The first month is an argument with selfHeal

Enabling `selfHeal` surfaces every hand-edit habit your team has, usually within days, usually mid-debugging when someone's live tweak vanishes under them. This is the system working. Announce the cutover, document the break-glass path (disable selfHeal on a single Application, annotate who and why, restore within 24 hours), and hold the line. Teams that carve out permanent exceptions end up operating two systems: GitOps for the services that behave, and vibes for the ones that page.