

a Tech Axolotl cheat sheet

AUTOMATIC DEPLOYMENT

the GitOps way



From `git push` to a running pod, and nobody types `kubectl apply`. The pipeline builds the image, **Argo CD** deploys it, **External Secrets** brings the passwords and **cert-manager** keeps TLS valid.



Why pull, not push

With **push**, CI holds a kubeconfig and runs `helm upgrade`: whoever owns the pipeline owns prod.

With **pull**, Argo CD lives in the cluster and reads git. CI only writes a commit.

on the next pages

- 2 six rules
- 3 the pipeline
- 4 Argo CD

- 5 secrets and certificates
- 6 rollouts and fixes

before any YAML SIX RULES

02



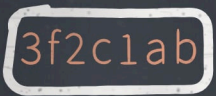
1 Git is the only door to prod

CI writes commits, Argo CD applies them. Nobody runs `kubectl apply` against prod, so every change has an author and a history.



2 Build once, ship the same image

Staging and prod run the exact image that passed the tests. No rebuild per environment.



3 Tag with the sha, never :latest

Every running pod points back to one commit. Make tags immutable in the registry, or pin the digest.



4 No secrets in git

Commit the ExternalSecret, which is only a pointer. The value stays in Vault or your cloud's secret store.



5 Every pod gets a readiness probe

Kubernetes only stops a bad rollout when it can tell that the new pods are not ready.



6 Rollback is git revert

Argo CD's selfHeal undoes a `kubectl fix` within seconds. Revert the commit and let it sync.

app repo: .github/workflows/release.yml

```

name: release
on:
  push:
    branches: [main]
permissions:
  contents: read
  packages: write      # push to ghcr.io
jobs:
  ship:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v7
      - run: make test      # 1
      - uses: docker/setup-buildx-action@v4
      - uses: docker/login-action@v4
        with:
          registry: ghcr.io
          username: ${GITHUB_ACTOR}
          password: ${GITHUB_TOKEN}
      - uses: docker/build-push-action@v4 # 2
        with:
          push: true
          tags: ghcr.io/acme/shop:${GITHUB_SHA}
          cache-from: type=gha
          cache-to: type=gha,mode=max
      - uses: aquasecurity/trivy-action@v0.36.0 # 3
        with:
          image-ref: ghcr.io/acme/shop:${GITHUB_SHA}
          severity: CRITICAL,HIGH
          exit-code: '1'
      - uses: actions/checkout@v7 # 4
        with:
          repository: acme/gitops
          token: ${GITOPS_TOKEN}
          path: gitops
      - working-directory: gitops/apps/shop/staging
        run: |
          kustomize edit set image \
            ghcr.io/acme/shop:${GITHUB_SHA}
          git config user.name ci-bot
          git config user.email ci-bot@acme.dev
          git commit -am "deploy shop ${GITHUB_SHA}"
          git push

```

① test: a red test stops everything

② build and push one image, tagged with the commit sha

③ scan: a HIGH or CRITICAL CVE fails the job, nothing is deployed

④ bump: CI commits the new tag to the gitops repo. That commit is the deploy.

staging to prod



The GITOPS_TOKEN

Write access to the gitops repo and nothing else: a GitHub App, or a fine-grained token with `contents: write` on that one repo. CI never gets a kubeconfig.

step 5 ARGO CD

04

```
gitops repo: argocd/shop-staging.yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: shop-staging
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/acme/gitops
    targetRevision: main
    path: apps/shop/staging
  destination:
    server: https://kubernetes.default.svc
    namespace: shop
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - CreateNamespace=true
  retry:
    limit: 5
    backoff:
      duration: 10s
      factor: 2
      maxDuration: 3m
```

app of apps



Apply **one** root Application by hand. It points at a folder of Applications.

- prune: deletes what was deleted in git
- selfHeal: a hand edit is put back in seconds
- retry: 10 s, 20 s, 40 s... at most 3 min apart

argocd CLI: the eight you will actually use

argocd app list	every app, its sync and health
argocd app get shop-staging	resources, last sync, errors
argocd app diff shop-staging	what git would change right now
argocd app sync shop-staging	sync now instead of waiting for the 3 min poll
argocd app get shop-staging --hard-refresh	drop the cache and read git again
argocd app history shop-staging	which revision ran, and when
argocd app rollback shop-staging 12	only with auto sync off; in GitOps, revert instead
argocd app terminate-op shop-staging	stop a sync that hangs

two statuses, not one

Sync: does the cluster match git? **Synced** or **OutOfSync**.

Health: does it work? **Healthy**, **Progressing**, **Degraded** or **Missing**.

A bad release often shows **Synced** and **Progressing**: git was applied fine, the new pods just never get Ready. More on page 6.

the helpers: External Secrets and cert-manager

SECRETS AND TLS

05

External Secrets: one store, then one per secret

```
apiVersion: external-secrets.io/v1
kind: ClusterSecretStore
metadata:
  name: vault
spec:
  provider:
    vault:
      server: https://vault.acme.internal
      path: secret
      version: v2
      auth:
        kubernetes:
          mountPath: kubernetes
          role: eso
```

```
---
apiVersion: external-secrets.io/v1
kind: ExternalSecret
metadata:
  name: shop-db
  namespace: shop
spec:
  refreshInterval: 1h
  secretStoreRef:
    kind: ClusterSecretStore
    name: vault
  target:
    name: shop-db
  data:
    - secretKey: password
      remoteRef:
        key: shop/db
        property: password
```



↘ Vault role "eso" must trust the service account ESO runs as

↘ rotated in Vault? picked up within 1 h

↗ ESO creates this Secret; the pod uses it like any other

cert-manager: one issuer, then one annotation per Ingress

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: ops@acme.dev
    privateKeySecretRef:
      name: letsencrypt-account
    solvers:
      - http01:
          ingress:
            ingressClassName: traefik
```

on the shop Ingress

```
metadata:
  annotations:
    cert-manager.io/cluster-issuer: letsencrypt
spec:
  tls:
    - hosts: [shop.acme.dev]
      secretName: shop-tls
```

↘ Let's Encrypt checks you on port 80: the Ingress must answer

↘ this one line orders the certificate

↘ created for you, renewed at 2/3 of its lifetime

step 6, and the bad days ROLLOUTS AND FIXES

06

a rollout with a brake



its readiness probe fails,
so no v1 pod is removed:
users never notice



Argo CD: Progressing,
after 10 min Degraded

```
in the Deployment
spec:
  progressDeadlineSeconds: 600
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 0
  template:
    spec:
      containers:
        - name: shop
          image: ghcr.io/acme/shop:<sha>
          readinessProbe:
            httpGet:
              path: /healthz
              port: 8080
```

when it breaks: the first command

OutOfSync forever `argocd app diff shop-staging`

a controller changes a field after sync: add
`ignoreDifferences`

Stuck Progressing `kubectl rollout status deploy/shop -n shop`

then describe a new pod; mostly the readiness
probe

ImagePullBackOff `kubectl describe pod <pod> -n shop`

the tag does not exist, or the pull secret is missing

CrashLoopBackOff `kubectl logs <pod> -n shop --previous`

the log of the run that crashed (`kubectl get pods`
finds it)

Secret missing `kubectl describe externalsecret shop-db -n shop`

the events name the store, the path or the
permission

No TLS `kubectl get certificate,challenge -n shop`

a pending challenge: Let's Encrypt cannot reach you

Bad release `git revert <sha> && git push`

the real rollback; `kubectl rollout undo` is put
back by selfHeal

more drawn explainers: @thetechaxolotl

