Assignment 2: GitOps-deploy your app with Argo CD

Goal: Use Argo CD to deploy the image you built in Assignment 1 onto the lab's live Kubernetes cluster, the pull-based GitOps way — without ever running `kubectl apply` against production yourself.

Where: A Git repo on the lab's Gitea server holds your Kubernetes manifests. Argo CD runs inside the lab cluster (master `10.100.100.7`) and pulls from that repo. Your image lives in the registry at `10.100.100.6`.

Tasks

1. Create a new Gitea repo (e.g. `<yourname>-deploy`) to hold deployment manifests. This is your GitOps source of truth — keep it separate from your app code.
2. Add a `k8s/` folder with Kubernetes manifests for your app: at minimum a `Deployment` and a `Service`. Point the Deployment's container `image:` at your SHA-tagged image from Assignment 1: `10.100.100.6/<yourname>-app:<sha>`.
3. Commit and push the manifests to `main`.
4. Create an Argo CD `Application` (a YAML manifest, as in chapter 4 section 3) that:
   - sets `source.repoURL` to your deploy repo, `targetRevision: main`, `path: k8s`;
   - sets `destination` to the in-cluster server and a namespace named after you;
   - enables `syncPolicy.automated` with `prune: true` and `selfHeal: true`.
5. Apply the Application to Argo CD (your mentor will tell you how to reach Argo CD — its UI or `argocd app create`). Confirm the Application appears and reaches **Synced** and **Healthy**.
6. Verify your app is actually running in the cluster (check the pods are Running and the Service responds).
7. **Demonstrate drift detection:** manually change the live Deployment (e.g. scale replicas with `kubectl scale`). Watch Argo CD report **OutOfSync** and then `selfHeal` revert it. Capture what you observed.
8. **Demonstrate a GitOps update:** change the image tag in your manifest to a newer SHA (rebuild via Assignment 1 if needed), commit, and watch Argo CD sync the new version automatically.

Deliverable

A link to your deploy repo (with `k8s/` manifests and the Argo CD `Application` YAML), plus a short note describing what you saw during the drift-detection test (steps 7) and the GitOps update (step 8). Include the Argo CD Application status showing Synced/Healthy.

Acceptance criteria — you're done when:

☐ A separate deploy repo exists with `k8s/` manifests (Deployment + Service) referencing your SHA-tagged image at `10.100.100.6`.

☐ An Argo CD `Application` manifest exists with the correct `source` (your repo, `path: k8s`, `main`) and `destination` (your namespace in the cluster).

☐ `syncPolicy.automated` is set with `prune: true` and `selfHeal: true`.

☐ The Application shows **Synced** and **Healthy** in Argo CD.

☐ Your app's pods are Running in the cluster and the Service responds.

☐ You triggered drift (a manual `kubectl` change) and observed Argo CD mark it OutOfSync and self-heal it back.

☐ You changed the image tag in Git, committed, and Argo CD synced the new version automatically — you never ran `kubectl apply` to deploy.

Hints

- Re-read chapter 4 sections 3–5; your Application is the example with your repo URL and namespace swapped in.
- Two repos is intentional: app code (Assignment 1) and deploy manifests (here). Mixing them is a common beginner mistake and muddles the "source of truth."
- If the pod won't start with `ImagePullBackOff`, the cluster can't pull from `10.100.100.6` — check the image name/tag is exactly right and ask your mentor about registry pull access from the cluster.
- "Synced but not Healthy" usually means the manifest applied but the app is crashing — check the pod logs, not Argo CD.
- For the drift test, `kubectl scale deployment/<name> --replicas=5` is an easy way to create drift; Argo CD should pull it back to whatever your manifest says.
- Remember: with GitOps, the *only* way you change production is a Git commit. If you're tempted to `kubectl apply` to fix something, change the manifest in Git instead.
- Blocked for >~30 min after re-reading the lessons? Bring what you've tried to your mentor.

Revision #3
Created 2026-05-26 11:16:00 UTC by Eslam Shapsough
Updated 2026-05-26 11:20:00 UTC by Eslam Shapsough