final p3 eval check

Here is your complete final P3 evaluation check — every checkbox from the evalsheet mapped to your exact implementation.

Configuration Checks

☑ Start up infrastructure

bash p3/scripts/install_k3d_argocd.sh

Status: Script exists — apply the fixed version from previous review (docker group fix)

☑ Config files present & explainable

```
ls p3/confs/      # → argocd-app.yaml
ls p3/dev-app/    # → deployment.yaml  service.yaml
ls p3/scripts/    # → install_k3d_argocd.sh
```

What to say per file:

File	Your explanation
argocd-app.yaml	"Declares the Argo CD Application — tells it which GitHub repo, which path, and which namespace to deploy into"
deployment.yaml	"Kubernetes Deployment for wil42/playground:v1 in the dev namespace — this is what Argo CD syncs"
service.yaml	"ClusterIP service exposing the app on port 8888 inside the cluster"
install_k3d_argocd.sh	"Installs Docker, kubectl, k3d, creates the cluster, installs Argo CD, applies the Application manifest"

☑ 2 namespaces: `argocd` and `dev`

```
kubectl get ns
# Must show both:
# argocd   Active
# dev      Active
```

Status: Both created in your script

☑ At least 1 pod in `dev` namespace

```
kubectl get pods -n dev
# wil-playground-xxxx   1/1   Running
```

Status: Deployed by argocd-app.yaml via deployment.yaml

☑ Namespace vs Pod — know the answer

What to say: "*A namespace is a logical boundary that isolates resources inside the cluster — like a folder. A pod is the smallest deployable unit in Kubernetes — it wraps one or more containers and runs the actual application.*"

☑ All 7 Argo CD pods Running

```
kubectl get pods -n argocd
# Expected 7 pods all Running:
# argocd-application-controller-0
# argocd-applicationset-controller-xxx
# argocd-dex-server-xxx
# argocd-notifications-controller-xxx
# argocd-redis-xxx
# argocd-repo-server-xxx
# argocd-server-xxx
```

Status: ⚠ Fixed version of script waits for all — apply the update

☑ Argo CD accessible in browser with credentials

```
# If not already running:
kubectl -n argocd port-forward svc/argocd-server 8080:443 &

# Get password:
kubectl -n argocd get secret argocd-initial-admin-secret \
  -o jsonpath='{.data.password}' | base64 -d
```

Open `https://localhost:8080` → login `admin` / `<password>`

Status: Fixed script auto-starts port-forward and prints credentials

☑ GitHub repo name includes login

```
# Your repoURL in argocd-app.yaml:
# https://github.com/usrali2026/Inception_of_Things.git
# "usrali2026" is in the name
```

Status: — evalsheet literally lists `usrali2026/Inception_of_Things` as an example[1]

☑ Docker image + both v1 and v2 tags on Docker Hub

```
# Verify right now before defense:
docker pull wil42/playground:v1   # must succeed
docker pull wil42/playground:v2   # must succeed
```

Check live at: `hub.docker.com/r/wil42/playground/tags`

Status: Using pre-made `wil42/playground` — no login requirement needed since it's Wil's image

☑ Argo CD Application correctly configured

```
kubectl get application dev-app -n argocd -o yaml | grep -E "repoURL|path|namespace|automated"
```

Expected output from your `argocd-app.yaml`:

```
repoURL: https://github.com/usrali2026/Inception_of_Things.git
path: p3/dev-app
namespace: dev
automated:
  prune: true
```

```
  selfHeal: true
```

Usage Checks — The GitOps Flow

☑ Navigate Argo CD UI + explain it

Open the browser → show the evaluator:

1. The `dev-app` application card
2. The resource tree (Deployment → ReplicaSet → Pod)
3. The sync status + repo URL

What to say: "*Argo CD watches this GitHub repo at path `p3/dev-app`. When the manifests in Git differ from what's running in the cluster, it marks the app OutOfSync and automatically reconciles — pulling the new state from Git and applying it.*"

☑ Verify app status

```
kubectl get application -n argocd
# NAME      SYNC STATUS   HEALTH STATUS
# dev-app   Synced        Healthy
```

☑ Verify v1 is running

```
kubectl get deployment -n dev \
  -o jsonpath='{.items[^0].spec.template.spec.containers[^0].image}'
# → wil42/playground:v1
```

☑ Verify Docker Hub is used

```
# Show image clearly comes from Docker Hub
kubectl get pods -n dev -o yaml | grep image
# → image: wil42/playground:v1
```

```
# → imageID: docker.io/wil42/playground@sha256:...
```

☑ Live v1 → v2 demo (the critical one)

```
# Step 1 — edit the file Argo CD watches
sed -i 's/wil42\/playground:v1/wil42\/playground:v2/' p3/dev-app/deployment.yaml

# Step 2 — push to GitHub
git add p3/dev-app/deployment.yaml
git commit -m "upgrade to v2"
git push

# Step 3 — watch sync (open second terminal)
watch kubectl get pods -n dev

# Step 4 — if auto-sync doesn't trigger in ~30s, force it:
argocd app sync dev-app

# Step 5 — verify
kubectl get application dev-app -n argocd
# → Synced   Healthy

kubectl get deployment -n dev \
  -o jsonpath='{.items[^0].spec.template.spec.containers[^0].image}'
# → wil42/playground:v2

kubectl get pod -n dev \
  -o jsonpath='{.items[^0].spec.containers[^0].image}'
# → wil42/playground:v2
```

☑ Optional — Rollback to v1 (do this, it's impressive)

```
sed -i 's/wil42\/playground:v2/wil42\/playground:v1/' p3/dev-app/deployment.yaml
git add p3/dev-app/deployment.yaml
git commit -m "rollback to v1"
git push
# Argo CD syncs back → pod returns to v1
```

Final Readiness Score

Category	Checks	Status

Folder structure	4/4 files correct	
`argocd-app.yaml`	All fields correct	
`deployment.yaml`	Correct image + namespace	
`service.yaml`	Correct selector + port	
`install_k3d_argocd.sh`	Docker group + all pod waits	⚠ Apply fix
Docker Hub tags v1+v2	`wil42/playground`	Verify online
GitOps demo commands	All ready	
Verbal explanations	Namespace/pod/GitOps	Prepare above

The evalsheet has one hard stop: "*If explanations are confused → evaluation stops now.*" The verbal explanation of the GitOps flow is **as important as the commands**. Memorize the flow: **Git push → Argo CD detects drift → reconciles → pod rolls over.**[1]