**Argocd**

**What is Argocd**

ArgoCD (**Argo Continuous Delivery**) is a **GitOps-based (track)** continuous delivery tool for Kubernetes.

Instead of manually deploying Kubernetes manifests, you store them in Git. ArgoCD automatically syncs the live cluster state with the desired state in Git.

**Example**: In Git (desired state): student-backend should run **3 replicas**.In Cluster (live state): someone manually scaled it to **5 replicas**.  
ArgoCD notices the difference and scales it back to **3 replicas** (syncs them).

Just like your **email app automatically syncs with the mail server**

If there’s a new email, it downloads it. If you delete it locally, it re-syncs to match the server.

**ArgoCD works the same way with Kubernetes**

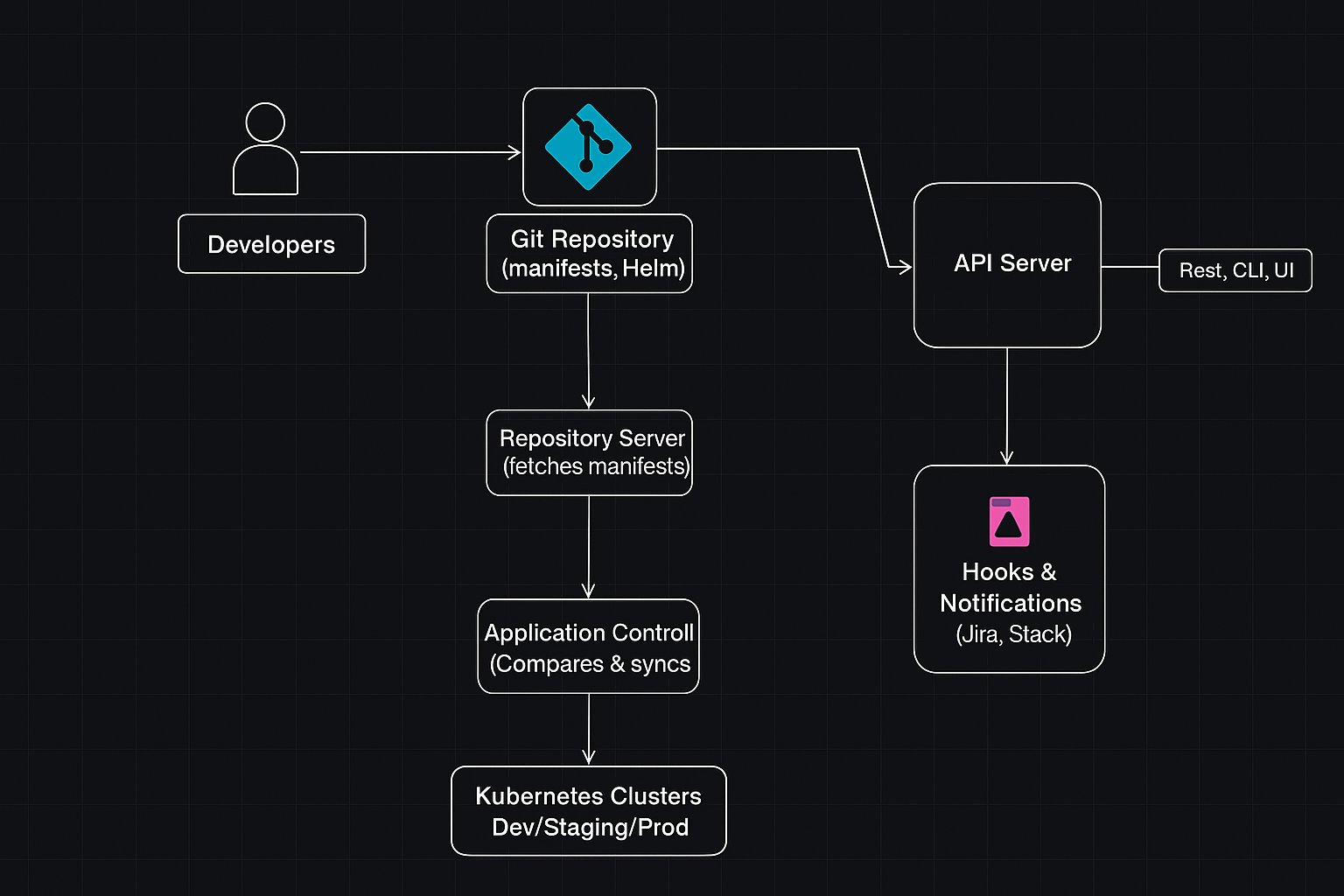
Git repo = **mail server (source of truth).** Kubernetes cluster = **email inbox (actual state)**ArgoCD = **sync engine** that keeps both the same automatically.

## **Why not just Jenkins for CD?**

Jenkins can deploy to Kubernetes (kubectl apply -f in pipeline), **but**:

1. **No GitOps**: State is in Jenkins jobs, not Git. Harder to track/rollback.
2. **No self-healing**: If someone changes a resource manually, Jenkins won’t detect it. ArgoCD will.
3. **No UI for live sync**: Jenkins doesn’t show cluster vs Git drift(slowly). ArgoCD has a nice UI for this.
4. **Multi-cluster pain**: Jenkins needs kubeconfigs for all clusters. ArgoCD handles multi-cluster easily.
5. **Auditing**: ArgoCD logs every sync to Git commit hash.

**Argocd Architecture**

****

Developers push code to Git (manifests, Helm, Kustomize).  
**ArgoCD API Server** gives access via UI, CLI, or REST/gRPC.  
**Repository Server** connects to Git and fetches application manifests.  
**Application Controller** compares Git (desired state) with Kubernetes (live state).  
If there’s a difference, it syncs and deploys to the right cluster (dev, staging, prod, etc.).  
**Sync hooks and notifications** can trigger extra actions (like alerts in Slack).

**Example:**

Gmail syncs emails between the server (desired state) and your inbox (live state) using a sync engine and cache. ArgoCD syncs Git (desired state) and Kubernetes cluster (live state) using API Server, Repo Server, Application Controller, and Redis.

**Installation:**

**Requirements**

* Installed [kubectl](https://kubernetes.io/docs/tasks/tools/install-kubectl/) command-line tool.
* Have a [kubeconfig](https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/) file (default location is ~/.kube/config).
* CoreDNS. Can be enabled for microk8s by microk8s enable dns && microk8s stop && microk8s start

**1.** **Login into your cluster**

· Create a namespace

>> kubectl create namespace bhavana

>> kubectl get ns

**2.** **Install Argo CD**

>> kubectl create namespace argocd

>> kubectl apply -n argocd -f<https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml>

**3.** **Download Argo CD CLI**

>> curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argocd/releases/latest/download/argocd-linux-amd64

>> sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd

>> rm argocd-linux-amd64

**4.** **Access The Argo CD API Server**

By default, the Argo CD API server is not exposed with an external IP. To access the API server, choose one of the following techniques to expose the Argo CD API server:

**Change the argocd-server service type to LoadBalancer:**

>> kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'

**After a short wait, your cloud provider will assign an external IP address to the service. You can retrieve this IP with:**

>> kubectl get svc argocd-server -n argocd -o=jsonpath='{.status.loadBalancer.ingress[0].ip}'

**5.** **Login Using The CLI**

The initial password for the admin account is auto-generated and stored as clear text in the field password in a secret named argocd-initial-admin-secret in your Argo CD installation namespace. You can simply retrieve this password using the argocd CLI:

>> argocd admin initial-password -n argocd

Using the username admin and the password from above, login to Argo CD's IP or hostname:

>> argocd login <ARGOCD\_SERVER>

Change the password using the command:

>> argocd account update-password

**6.** **Access Argo Cd Application**

>> kubectl get pods -A

· Copy external IP address of Load balancer pod.

· Paste it in browser ‘External IP address:80’

· Click on Advanced > Proceed

· Provide Username & Password > Sign in

**Deploying Application from a Git Repository**

**1.** **Go to Settings**

· Select **Repositories**

· Click on **+Connect Repo** > Choose connection: **HTTPS/HTTP**

· Give type: git and provide Name

· Set project as **Default**

· Provide Project URL – git repo URL

· Provide Git username and Password (Provide Token created in git)

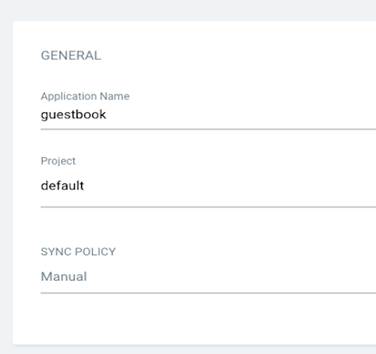
· Click on **Connect**

**2.** **Go to Applications**

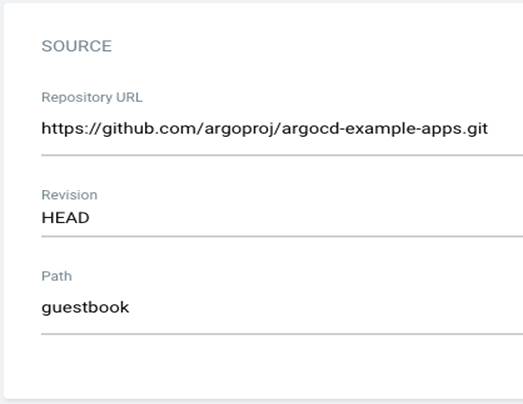
* Click the **+ New App** button as shown below:



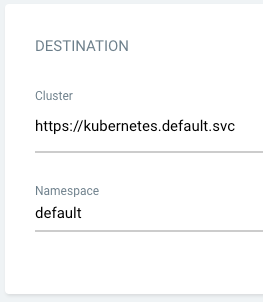
* Give your app the name, select project as default, and give the sync policy as Automatic:



* Connect the git repo to Argo CD by setting repository url to the github repo url, Revision – Your Branch name , and set the path to Folder where Manifest files exists:



* For Destination, set cluster URL to **https://kubernetes.default.svc** and give your namespace:



* After filling out the information above, click Create at the top of the UI to create the guestbook application:

