# **ARGO CD**

**What is Argo CD ?**

**Argo CD** is a GitOps-based continuous delivery tool designed for Kubernetes. It enables applications to be deployed automatically into Kubernetes clusters by pulling configurations directly from Git repositories.

**In simple terms:**

* **GitOps:** The Git repository acts as the single source of truth for your application’s desired configuration.
* **Declarative:** You define the intended state of your applications in YAML manifests stored in Git.
* **Continuous Delivery:** Argo CD keeps watching both Git and your Kubernetes cluster. If it detects any drift between them, it ensures the cluster is brought back to the state defined in Git.

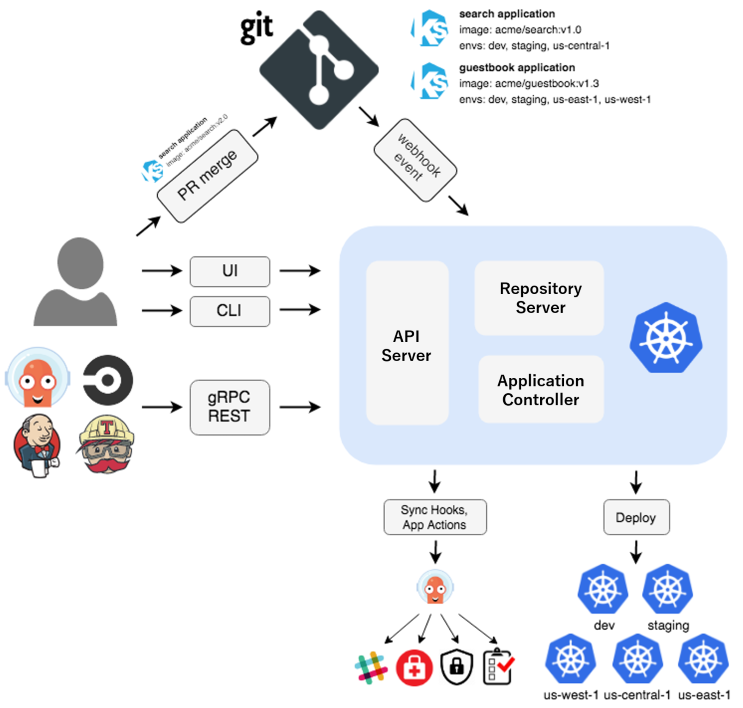
**Why Choose Argo CD?**

* **Automation:** Minimizes the need for manual deployment tasks by handling application delivery automatically.
* **Consistency:** Keeps your Kubernetes clusters aligned with the desired configurations stored in Git, avoiding drift.
* **Accountability:** Since all modifications pass through Git, you gain a complete history of changes.
* **Simple Rollbacks:** Reverting to a stable version is as easy as restoring a previous commit in Git.
* **Clarity:** Offers an intuitive web interface that gives clear insight into the health and status of your applications.

**Core Concepts in Argo CD**

1. **Application:** A collection of Kubernetes resources (like Deployments, Services, or Ingresses) that work together as one unit, usually representing a microservice or workload.
2. **Source:** The Git repository that stores the YAML manifests or Helm charts describing your application.
3. **Destination:** The Kubernetes cluster where the application should be deployed.
4. **Desired State:** The configuration defined in Git that represents how the application should run.
5. **Live State:** The current state of the application that actually exists in the Kubernetes cluster.
6. **Sync:** The action of aligning the live state in the cluster with the desired state from Git. This can be triggered manually or automatically by Argo CD.

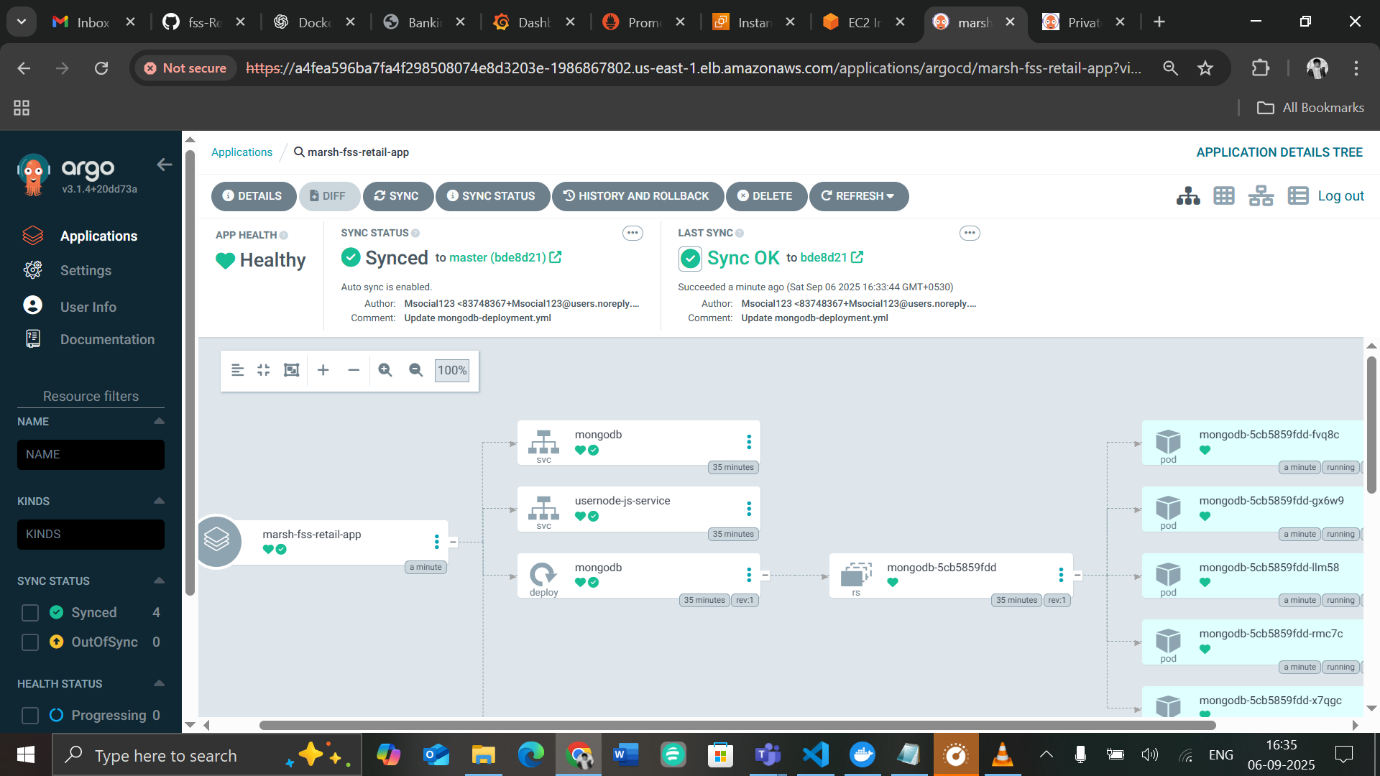
**Argo CD Architecture**



**Argo CD Architecture Components**

1. API Server
   * Acts as the main entry point for all interactions.
   * Exposes gRPC and REST APIs, which both the CLI and the web UI use.
   * Handles application management tasks and provides status updates.
2. Controller
   * Considered the “brain” of Argo CD.
   * Continuously checks Git repositories for changes in the desired state.
   * Observes the Kubernetes cluster to detect the live state of applications.
   * If there’s a mismatch between Git (desired state) and the cluster (live state), it flags the app as *OutOfSync*.
   * If auto-sync is enabled, it automatically triggers synchronization.
3. Repository Server (Repo Server)
   * Fetches and clones manifests from Git repositories.
   * Keeps a cache to boost performance.
   * Processes manifests (using Helm, Kustomize, or plain YAML) before passing them to the Controller.
4. Application Controller
   * Monitors Application Custom Resources (CRs) inside Kubernetes.
   * Requests the Repo Server for the desired state and compares it with the live cluster state.
   * Executes the actual kubectl apply operations to bring the cluster into sync.
5. Redis
   * Provides a caching layer for storing app states, authentication tokens, and other temporary data used by Argo CD components.
6. Dex (Optional)
   * An identity service for Single Sign-On (SSO).
   * Supports authentication with providers like Google, GitHub, and LDAP.
7. Web UI
   * A browser-based interface that offers a clear view of application health and sync status.
   * Lets users manage apps, trigger syncs, check logs, and more, through a user-friendly dashboard.

*Below is an example of the UI, showing application health and sync details:*



**Argo CD Installation**

Requirements

* kubectl installed and configured with a kubeconfig (~/.kube/config)
* A running Kubernetes cluster
* CoreDNS enabled (for MicroK8s):

microk8s enable dns && microk8s stop && microk8s start

Step 1: Login to your Cluster and Create Namespaces

1. Create a namespace for your applications:

kubectl create namespace sunmarsh

kubectl get ns

1. Create a namespace for Argo CD:

kubectl create namespace argocd

Step 2: Install Argo CD

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

Check that Argo CD pods are running:

kubectl get pods -n argocd

Step 3: Download and Install 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

Test CLI:

argocd version

Step 4: Access the Argo CD API Server

By default, Argo CD server is a ClusterIP service. You can access it in two ways:

Option 1: Port Forward (Local/Minikube/MicroK8s)

kubectl port-forward svc/argocd-server -n argocd 8080:443

Access in browser:

<https://localhost:8080>

Option 2: Expose via LoadBalancer (Cloud)

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

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

Then access in browser:

https://<EXTERNAL\_IP>

Note: On local clusters, LoadBalancer may not work. Use port-forward instead.

Step 5: Login to Argo CD

1. Retrieve initial admin password:

argocd admin initial-password -n argocd

1. Login using CLI:

argocd login <ARGOCD\_SERVER> --username admin --password <PASSWORD> --insecure

1. Change password:

argocd account update-password

Step 6: Access the Argo CD UI

1. Open browser and go to the URL (port-forward or LoadBalancer IP)
2. Ignore certificate warning (self-signed) → Click Advanced > Proceed
3. Login using admin username and password

You can now start creating Applications and manage your deployments from Git repos.

**Deploying an Application from a Git Repository:**

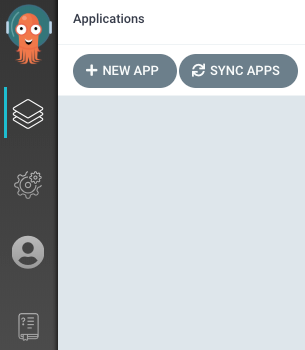
1. Go to Settings
   * Navigate to the Settings section of your deployment platform.
2. Select Repositories
   * Find and click on Repositories to manage your source connections.
3. Connect a New Repository
   * Click on + Connect Repo.
   * Choose the connection type: HTTPS/HTTP.
4. Provide Repository Details
   * Type: Select git.
   * Name: Give a name to your repository connection (for identification).
   * Set Project as Default: Enable if this repository will be the default for your project.
   * Project URL: Enter the Git repository URL

(e.g., https://github.com/username/repo.git).

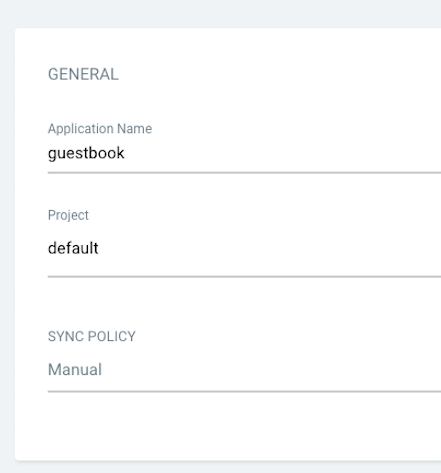
* + Authentication: Provide your Git username and password/token.
    - Tip: Use a personal access token if your Git provider requires it instead of a password.

1. Connect
   * Click Connect to finalize the repository connection.

**Go to Applications and click the + New App button:**

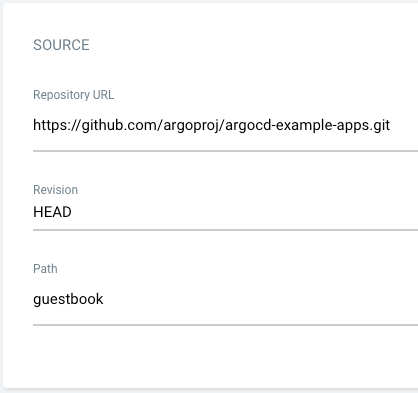


Enter a name for your application, select **default** for the project, and set the **sync policy** to **Automatic**.

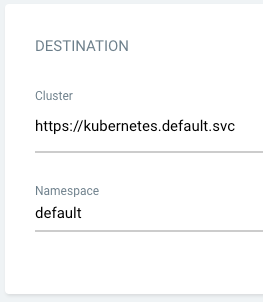


Connect your Git repository to Argo CD by specifying:

* **Repository URL**: the URL of your GitHub repo
* **Revision**: your branch name
* **Path**: the folder containing your manifest files



For the **Destination**, set the **cluster URL** to https://kubernetes.default.svc and provide the desired **namespace**:



Once all fields are completed, click Create at the top of the UI to create your Guestbook application:

