# **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.

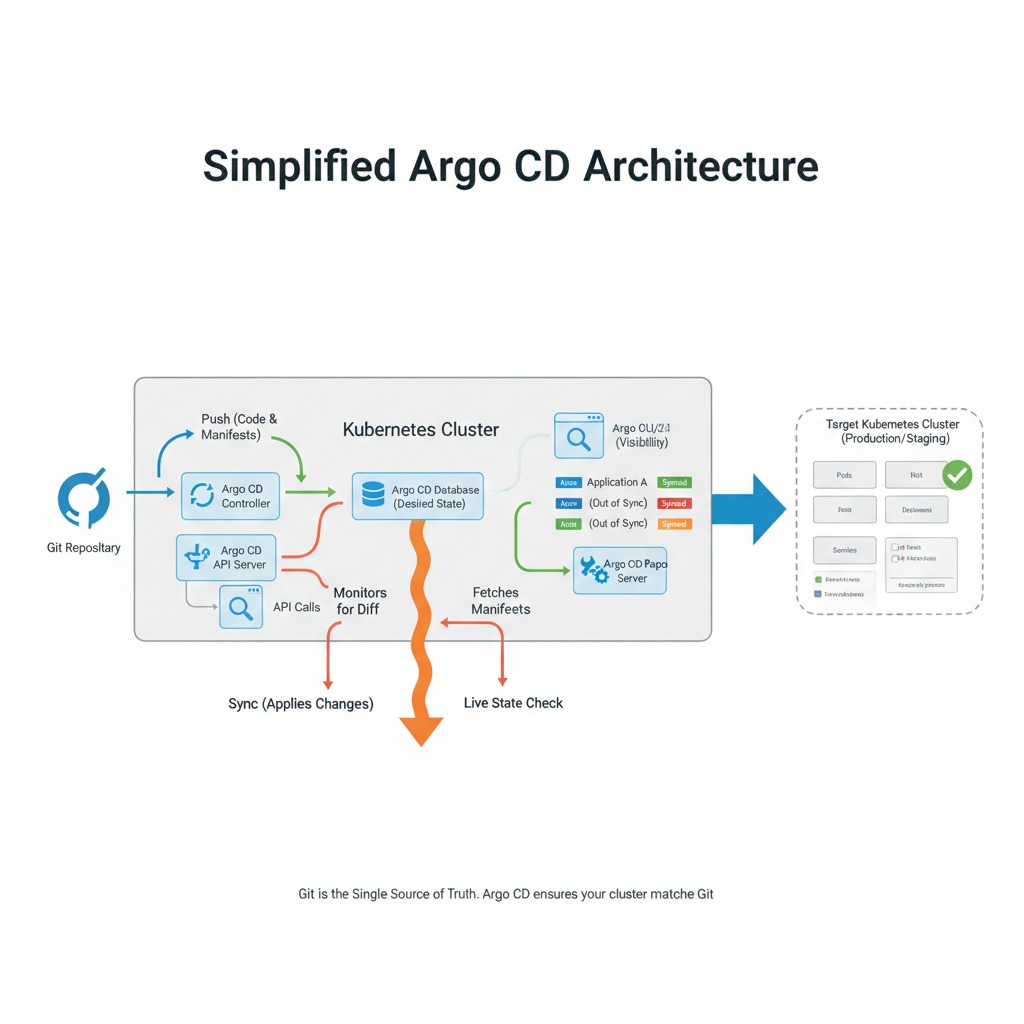
**Why Choose Argo CD?**

* **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.

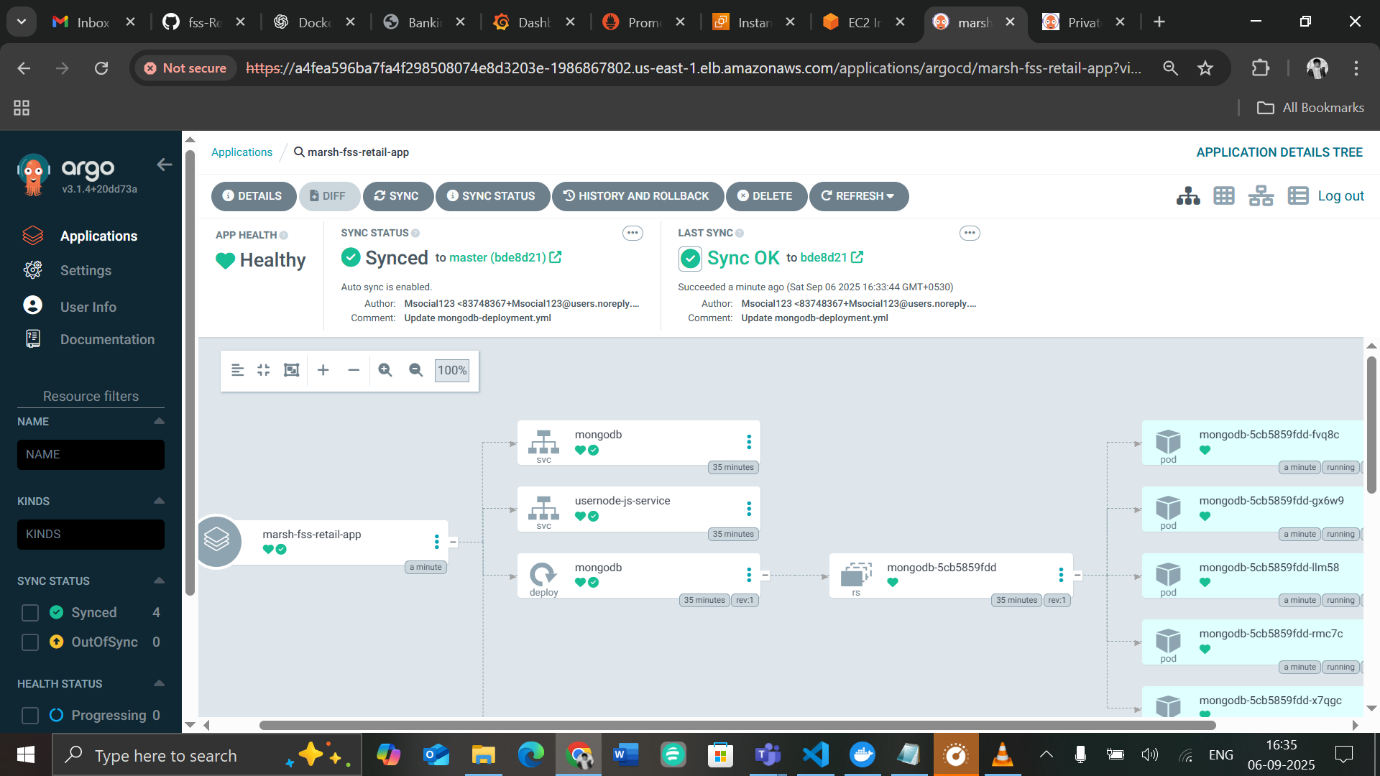
**Argo CD Architecture**



**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:*



**Installation of Argo CD:**

**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 sunmarsh

>> kubectl get ns

1. **Install Argo CD**

>> kubectl create namespace argocd

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

1. **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

1. **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}'

1. **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

1. **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 an Application from a Git Repository:**

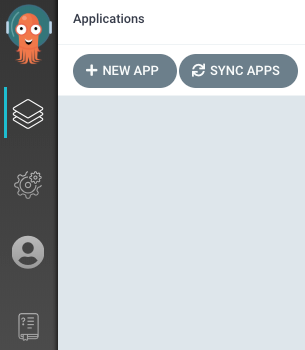
1. Go to Settings
   * Navigate to the Settings section of your deployment platform.
2. Select Repositories
   * Find and click on Repositories to manage your source connections.
3. Connect a New Repository
   * Click on + Connect Repo.
   * Choose the connection type: HTTPS/HTTP.
4. Provide Repository 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).

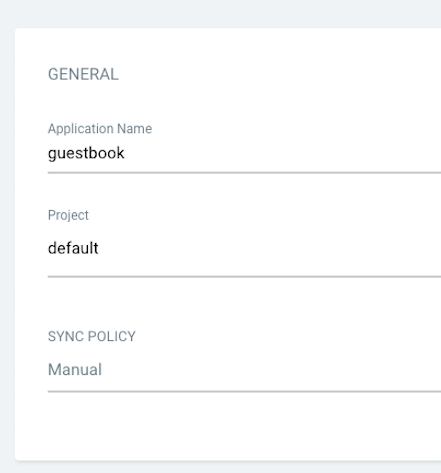
* + Authentication: Provide your Git username and password/token.
    - Tip: Use a personal access token if your Git provider requires it instead of a password.

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

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

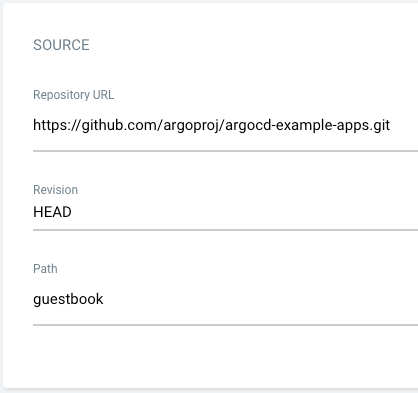


Enter a name for your application, select **default** for the project, and set the **sync policy** to **Automatic**.

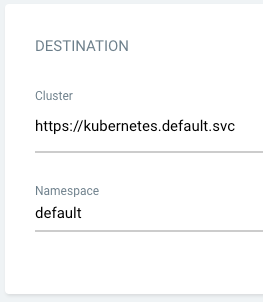


Connect your Git repository to Argo CD by specifying:

* **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:

