# ARGO CD

**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 satyam-ns

>> 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

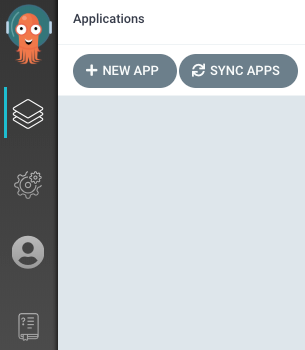
**Deploying Application from a Git Repository**

1. **Go to Settings**

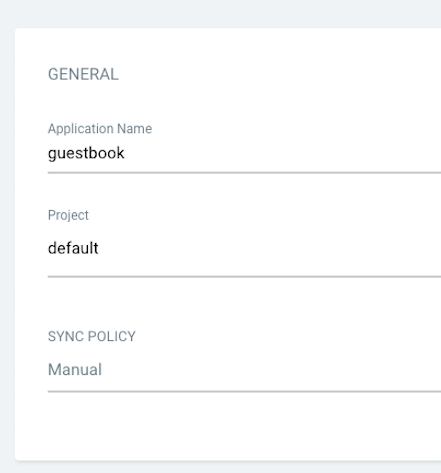
* Select **Repositories**
* Click on **+Connect Repo** > Choose connection: **HTTPS/HTTP**
* Give type: git and provide Name
* Set project as **Default**
* Provide Project URL – git repo URL
* Provide Git username and Password (Provide Token created in git)
* Click on **Connect**

1. **Go to Applications**

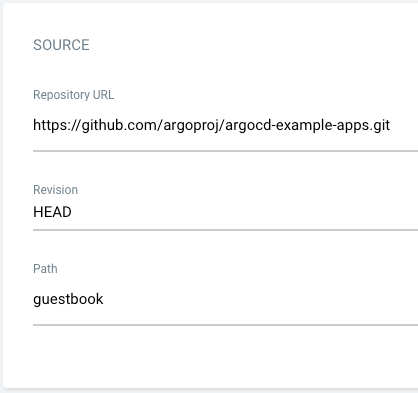
* Click the **+ New App** button as shown below:



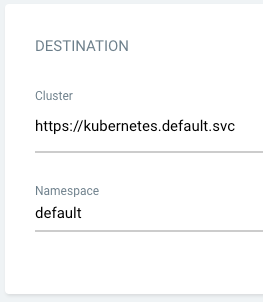
* Give your app the name, select project as default, and give the sync policy as Automatic:



* 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:



ABOUT ARGO CD

Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes. It automates the deployment of applications to Kubernetes clusters directly from Git repositories.

In a nutshell:

**GitOps:** Your Git repository is the single source of truth for your application's desired state.

**Declarative:** You "declare" how your application should look in YAML files in Git.

**Continuous Delivery:** Argo CD constantly monitors your Git repository and your Kubernetes cluster. If there's a difference, it automatically syncs your cluster to match what's in Git.

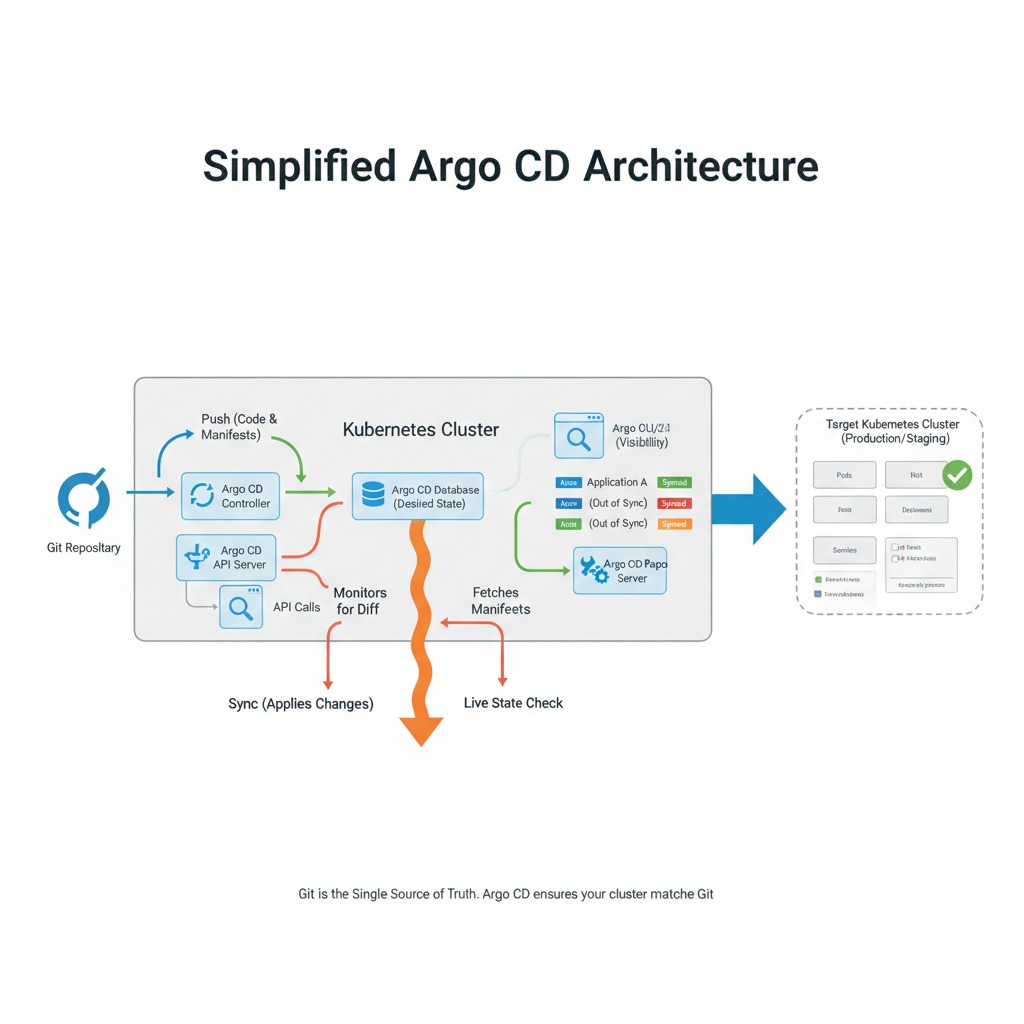
**Why use Argo CD?**

* Automation: Reduces manual effort in deploying and managing applications.
* Reliability: Ensures your cluster state always matches your desired state in Git, preventing configuration drift.
* Traceability: Every change to your application goes through Git, providing a clear audit trail.
* Rollbacks: Easily roll back to a previous working state by reverting changes in Git.
* Visibility: Provides a user-friendly UI to see the status of your applications and deployments.

**Core Concepts:**

1. **Application:** In Argo CD, an "Application" is a grouping of Kubernetes resources that belong together (e.g., a deployment, service, ingress for a single microservice).
2. **Source:** This is your Git repository where your application's YAML manifests are stored.
3. **Destination:** This is the Kubernetes cluster where your application will be deployed.
4. **Desired State:** What you've defined in your Git repository.
5. **Live State:** The actual state of your application in the Kubernetes cluster.
6. **Sync:** The process of making the Live State match the Desired State. Argo CD can do this automatically or manually.

# ARGO CD ARCHITECTURE



1. **Argo CD API Server:**
   * This is the central point of contact.
   * It provides the gRPC/REST API that the UI and CLI interact with.
   * Responsible for application management and status reporting.
2. **Argo CD Controller:**
   * This is the "brain" of Argo CD.
   * It continuously monitors your Git repositories for changes to your application's desired state.
   * It also monitors your Kubernetes clusters for the live state of your applications.
   * When it detects a difference between the desired state (in Git) and the live state (in the cluster), it marks the application as "OutOfSync."
   * If auto-sync is enabled, it triggers the synchronization process.
3. **Argo CD Repo Server:**
   * This component is responsible for cloning application manifests from Git repositories.
   * It caches repository content to improve performance.
   * It renders Kubernetes manifests (e.g., using Kustomize, Helm, or raw YAML) from the Git repository. The Controller then uses these rendered manifests.
4. **Argo CD Application Controller (sometimes referred to as the main controller):**
   * Watches for Application custom resources (CRs) in Kubernetes.
   * Responsible for invoking the Repo Server to get the desired state and comparing it with the live state in the target cluster.
   * Performs the actual kubectl apply operations to synchronize the cluster.
5. **Argo CD Redis:**
   * A caching layer used by the API server and other components to store application state, authentication tokens, and other transient data.
6. **Argo CD Dex (optional, for OIDC integration):**
   * An identity service that supports various identity providers (e.g., Google, GitHub, LDAP) for authentication.
7. **Argo CD UI (Web Interface):**
   * A user-friendly web-based interface that provides a visual representation of your applications, their status, and the sync process.
   * Allows you to manage applications, view logs, perform manual syncs, and more.

Here's what the UI typically looks like, showing application health and sync status:

