ARGO-CD

What is argo-cd???

Argo CD is a **declarative, GitOps continuous delivery tool** for Kubernetes. In simpler terms, it helps you **automate the deployment of applications to Kubernetes clusters** using Git as the source of truth.

Here’s a breakdown:

**Key Features:**

1. **GitOps-based Deployment**
   * Your application’s desired state (like deployments, services, config maps) is stored in a **Git repository**.
   * Argo CD continuously monitors Git and your Kubernetes cluster to ensure the live state matches the desired state in Git.
2. **Declarative Approach**
   * You define **what you want**, not how to do it.
   * Any change in Git triggers an update in the cluster automatically (or manually if you choose).
3. **Application Management**
   * Supports multiple environments (dev, staging, production).
   * Allows rollbacks if something goes wrong.
4. **Sync and Health Checks**
   * Can automatically synchronize cluster state with Git.
   * Provides health status of applications (Healthy, Degraded, Unknown).
5. **Web UI and CLI**
   * You can manage apps via a **dashboard**, **CLI**, or **REST API**.
6. **Supports Helm, customize, JSON/YAML**
   * Works with popular tools to define Kubernetes manifests.

Why argo-cd over Jenkins??

**Jenkins**

* **Type:** CI/CD tool (Continuous Integration / Continuous Delivery)
* **Focus:** Automating **builds, tests, and deployments**
* **How it works:**
  1. Code is pushed to Git → Jenkins triggers a **pipeline**.
  2. Pipeline builds the code, runs tests, creates artifacts (Docker images, binaries).
  3. Optionally deploys to environments using kubectl, Helm, or scripts.
* **Strength:** Handles **build/test automation** very well.
* **Weakness:**
  1. Manual scripts needed for deployments.
  2. Harder to ensure the **cluster state matches the Git repo** (no drift detection).

**Argo CD**

* **Type:** GitOps **continuous delivery** tool
* **Focus:** Deploying **applications to Kubernetes clusters** and keeping them in sync with Git
* **How it works:**
  1. You define the desired Kubernetes manifests in Git.
  2. Argo CD continuously **syncs cluster state with Git**.
  3. Detects **drifts** if someone manually changes the cluster and can auto-correct.
* **Strength:**
  1. Declarative, GitOps approach
  2. Visual dashboard with health status
  3. Supports Helm/Kustomize and multi-environment setups
* **Weakness:**
  1. Doesn’t build/test your code (it assumes manifests or images are ready).

Architecture of Argo-cd  
  
**1. High-Level Overview**

Argo CD follows a **client-server architecture** and works inside Kubernetes. The main goal is to **ensure the live state of applications in the cluster matches the desired state in Git**.

**2. Key Components**

**a) Argo CD API Server**

* The **central entry point** for users and tools.
* Exposes:
  + **REST API**
  + **gRPC API**
  + **Web UI**
* Responsibilities:
  + Receives requests (create app, sync, rollback)
  + Communicates with Kubernetes cluster and repository

**b) Argo CD Repository Server**

* Responsible for interacting with Git or Helm repositories.
* **Fetches manifests** (YAML, Helm charts, Kustomize files) from Git.
* Validates and generates Kubernetes manifests for deployment.

**c) Argo CD Application Controller**

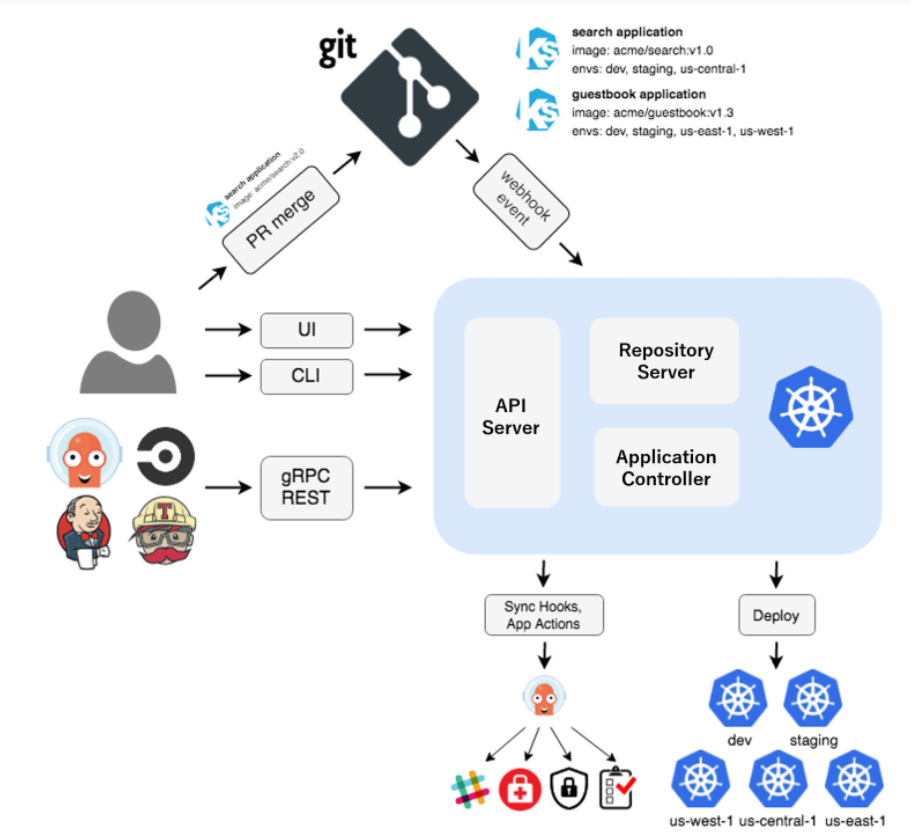
* **Core of Argo CD**.
* Watches the **Git repo and Kubernetes cluster continuously**.
* Detects **drift** between the desired state in Git and the live state in the cluster.
* Syncs changes automatically (or manually if configured).
* Manages deployment lifecycle (sync, rollback, prune).

**d) Argo CD Dex (Optional)**

* Optional component for **SSO / authentication**.
* Supports OAuth2, LDAP, SAML.
* Integrates with the API server to manage users and permissions.

**e) Kubernetes Cluster (Managed Clusters)**

* Argo CD can manage **one or multiple clusters**.
* Uses **Kubernetes RBAC** and service accounts to apply resources.

Flow of argocd-----  
  
Git Repository (desired state)

|

v

Argo CD Repo Server

|

v

Application Controller

|

v

Kubernetes Cluster (live state)

^

|

Drift Detected?

|

Sync/Rollback if needed

How to setup argocd?

**How to install Argo CD**

**Requirements**

* Install kubectl.
* Have a kube config file.

**Install Argo CD**

kubectl create namespace argocd

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

(It creates the namespace named argocd.)

**Download the Argo CD CLI**

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

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

rm argocd-linux-amd64

**Access the API server of Argo CD**

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

kubectl get svc -n argocd

* Get the **external IP** and use it to access the app in browser.

**Login into Argo CD**

argocd admin initial-password -n argocd

argocd login <external-ip>

argocd account update-password

**Creating app in Argo CD**

* Click on **Settings**.
* Choose **Repositories → Connect Repo**.
* Choose connection method as **HTTP/HTTPS**, provide:
  + Type: **Git**
  + Name (optional)
  + Project: **default**
  + Repository URL
  + Username: your GitHub username
  + Password: GitHub token
* Click on **Connect**.
* Go to **Settings → New App**.
* Provide:
  + Application name
  + Project: **default**
  + Sync policy: **Automatic**
  + Repository URL
  + Branch
  + Path of the manifest files
  + Cluster URL
  + Namespace
* Click on **Create → Sync → Synchronize**.