Agro cd

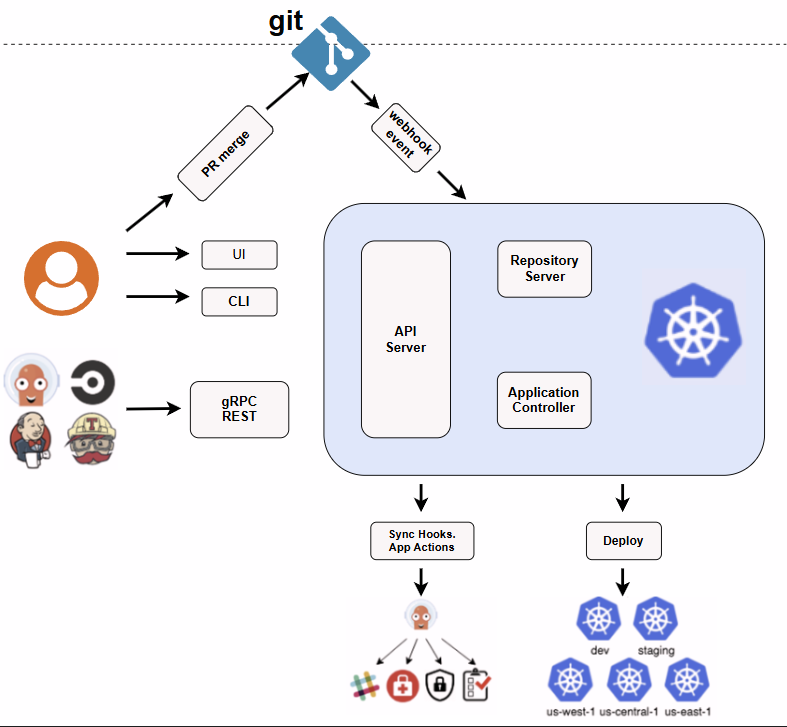
What Is Argo CD?

* **ArgoCD** is a **declarative, GitOps-based continuous delivery tool** for Kubernetes.
* It continuously monitors **Git repositories** and automatically syncs the desired state with the actual state in Kubernetes clusters.

Why Argo CD?

* Application definitions, configurations, and environments should be declarative and version controlled. Application deployment and lifecycle management should be automated, auditable, and easy to understand.

**Architecture**

****

**1. Git Repository**

* All Kubernetes manifests (YAML, Helm charts, Kustomize, etc.) live in Git.
* Developers push changes → merge pull requests.
* Git generates webhook events to notify ArgoCD that something has changed.

**2. Web UI & CLI**

* Webapp: The web interface for managing and visualizing applications.
* CLI: Command-line tool for scripting and automation, interacting with the API server.

**3. Inside kubernet cluster**

**1. API Server**

* Acts as a gRPC/REST server consumed by the Web UI, CLI, and CI/CD systems.
* Responsibilities include:
  + Application management and status reporting
  + Initiating operations like sync, rollback, and custom actions
  + Managing credentials for repos and clusters (stored as Kubernetes secrets)
  + Handling authentication, authorization (including RBAC), and delegation to identity providers
  + Listening for Git webhook events

**2. Repository Server (Repo Server)**

* Maintains a **local cache** of Git repositories holding application manifests.
* Generates and serves the required Kubernetes manifests based on:
  + Repository URL
  + Revision (branch, tag, or commit)
  + Application path
  + Any template-specific settings (like Helm values)

**3. Application Controller**

* A Kubernetes controller that continuously **reconciles** live cluster state with the desired state defined in Git.
* Detects when an application is OutOfSync and triggers corrective actions.
* Manages lifecycle hooks such as PreSync, Sync, and PostSync

**4. Sync Hooks / App Actions**

* Custom logic that runs before/after deployments (e.g., database migrations, approvals).
* Integration with external systems like Slack, notifications, or security checks.

**5.Deployment to Clusters**

* Ensures each cluster matches Git definitions.
* ArgoCD deploys apps into:

Multiple environments (dev, staging, prod).

Multiple clusters (multi-region: us-west-1, us-central-1, us-east-1).

**How ArgoCD Works**

Think of 3 main states:

1. **Desired State** → Stored in Git repo (YAML, Helm, Kustomize).
2. **Live State** → Actual running state in the Kubernetes cluster.
3. **Observed State** → ArgoCD continuously compares Desired vs Live.

* If **drift** happens, ArgoCD alerts or automatically syncs back.

**Argo CD Workflow**

* **Git Repository:** Stores all desired (declarative) application state and configurations.
* **Refresh:** Argo CD regularly fetches/appraises the configuration from Git.
* **Sync:** Argo CD can automatically or manually apply changes so the live state matches the desired state from Git (“GitOps”).
* **Drift Detection:** Reports when live state diverges from Git-defined state and helps you resolutely sync.

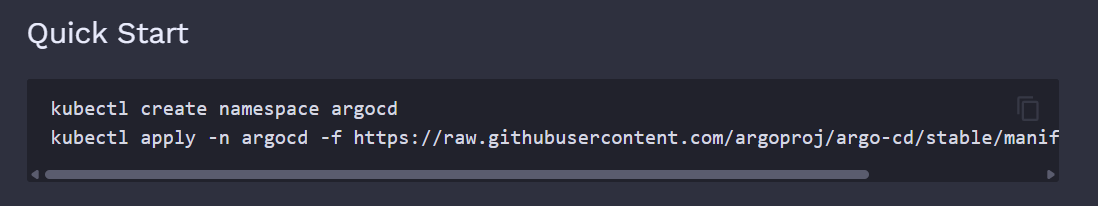
Installation of AgroCD

Step 1:

This will create a new namespace, argocd, where Argo CD services and application resources will live.

kubectl create namespace argocd

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

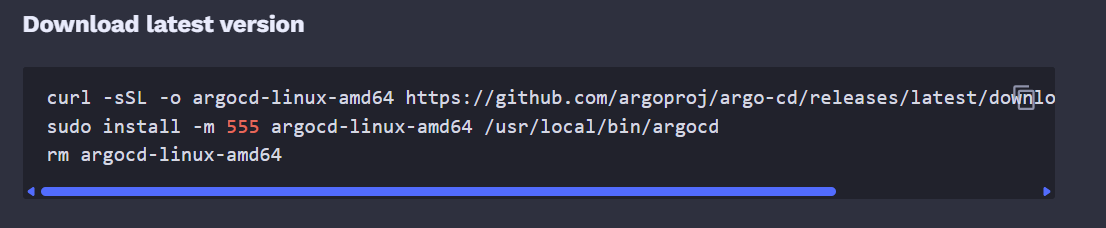


step2:-

Download Argo CD CLI

kubectl create namespace argocd

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



step3:-

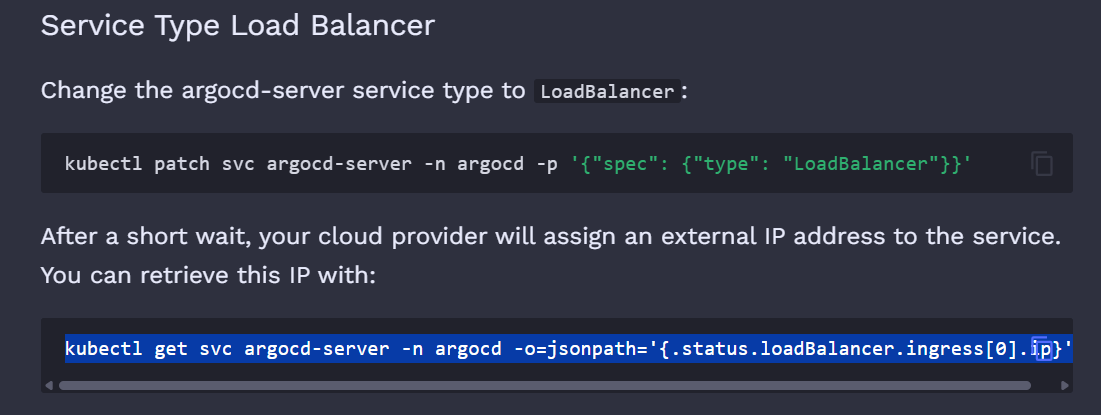
Access The Argo CD API Server

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}'



step 5:-

Get intial password

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d

step 6:-

# Access UI (port-forward)

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