Argo CD

Argo CD is an open-source, declarative, GitOps continuous delivery tool for Kubernetes. It helps automate the deployment and management of applications in Kubernetes clusters by using Git repositories as the source of truth for defining and managing application configurations. Argo CD follows GitOps principles, which emphasize using Git repositories as the single source of truth for infrastructure and application configurations, promoting collaboration, auditability, and reproducibility.

Key Features of Argo CD:

1. **Declarative Configuration**: Argo CD enables users to define application configurations declaratively using YAML manifests. These manifests describe the desired state of applications and their dependencies, such as Kubernetes resources like deployments, services, ingress rules, and more.

2. **Continuous Delivery**: Argo CD continuously monitors Git repositories for changes in application configurations. When changes are detected, it automatically reconciles the desired state defined in the Git repository with the actual state of applications running on Kubernetes clusters.

3. **Automated Synchronization**: Argo CD synchronizes application configurations with Kubernetes clusters, ensuring that the cluster's state matches the desired state defined in the Git repository. It automates the deployment, updating, and deletion of applications and their resources based on changes in Git.

4. **Rollback and Auditing**: Argo CD supports automated rollbacks to previous versions of applications in case of failures or errors during deployment. It provides audit logs and history tracking, allowing users to review changes, rollbacks, and deployments for auditing and troubleshooting purposes.

5. **Multi-Environment Support**: Argo CD facilitates managing applications across multiple environments, such as development, staging, and production. It allows users to define different configurations for each environment, ensuring consistency and reproducibility across environments.

6. **User Interface and CLI**: Argo CD provides a web-based user interface and a command-line interface (CLI) for managing applications, monitoring deployments, and performing administrative tasks. The UI offers visibility into application status, synchronization status, and deployment history.

Example Usage of Argo CD:

Let's consider an example scenario where a team wants to deploy a microservices-based application to a Kubernetes cluster using Argo CD:

1. **Application Configuration**: The team defines the application's configuration declaratively using YAML manifests. These manifests include specifications for various microservices, database connections, ingress rules, and other resources required for the application.

2. **Git Repository Setup**: The application configurations are stored in a Git repository, which serves as the source of truth for the application's desired state. The team collaborates on the application configurations, making changes and updates as needed.

3. **Argo CD Configuration**: The team configures Argo CD to monitor the Git repository for changes in the application configurations. They specify the Git repository URL, branch, and path to the application manifests, allowing Argo CD to sync the desired state with the Kubernetes cluster.

4. **Continuous Deployment**: Whenever the team makes changes to the application configurations and commits them to the Git repository, Argo CD detects the changes and automatically deploys or updates the application in the Kubernetes cluster. It reconciles the desired state with the actual state of the cluster, ensuring that the application is deployed according to the latest configuration.

5. **Rollback and Auditing**: If a deployment fails or encounters errors, Argo CD automatically rolls back to the previous known-good version of the application. It provides audit logs and history tracking, allowing the team to review deployment history, changes, and rollbacks for auditing and troubleshooting purposes.

6. **User Interface and Monitoring**: The team monitors the deployment status, synchronization status, and application health using the Argo CD user interface. They can view application details, review synchronization logs, and perform administrative tasks such as configuring access control and managing synchronization policies.

In this example, Argo CD simplifies and automates the deployment and management of the microservices-based application in the Kubernetes cluster, following GitOps principles for versioning, collaboration, and reproducibility.

Problem Statement:

In modern software development, deploying applications to Kubernetes clusters and managing their configurations across different environments can be challenging. Some common challenges include:

1. **Manual Deployment Processes**: Manually deploying applications to Kubernetes clusters can be error-prone and time-consuming, especially when managing multiple environments.

2. **Lack of Version Control**: Without proper version control, tracking changes to application configurations becomes difficult, leading to inconsistencies and potential issues.

3. **Complexity in Managing Environments**: Managing configurations across different environments (e.g., development, staging, production) requires coordination and can lead to discrepancies between environments.

4. **Limited Rollback Capabilities**: In case of deployment failures or errors, rolling back to a previous known-good state may be challenging without automated processes.

Solution by ArgoCD:

Argo CD addresses these challenges by providing a GitOps-based continuous delivery solution for Kubernetes:

1. **Declarative Configuration**: Argo CD allows users to define application configurations declaratively using YAML manifests. These manifests describe the desired state of applications and their resources.

2. **Git Repository as Source of Truth**: Argo CD uses Git repositories as the source of truth for application configurations. Users store configuration manifests in Git, enabling version control, collaboration, and auditability.

3. **Continuous Deployment**: Argo CD continuously monitors Git repositories for changes in application configurations. When changes are detected, it automatically deploys or updates applications in Kubernetes clusters to match the desired state.

4. **Automated Rollbacks**: Argo CD supports automated rollbacks to previous versions of applications in case of deployment failures or errors. This ensures reliability and stability of deployments.

5. **Multi-Environment Support**: Argo CD supports managing applications across multiple environments, allowing users to define different configurations for each environment. This ensures consistency and reproducibility across environments.

ArgoCD Workflow:

The workflow of Argo CD involves the following steps:

1. **Define Application Configuration**: Users define application configurations declaratively using YAML manifests. These manifests include specifications for Kubernetes resources such as deployments, services, ingress rules, etc.

2. **Store Configuration in Git**: Application configurations are stored in Git repositories, serving as the source of truth. Changes to configurations are versioned and auditable.

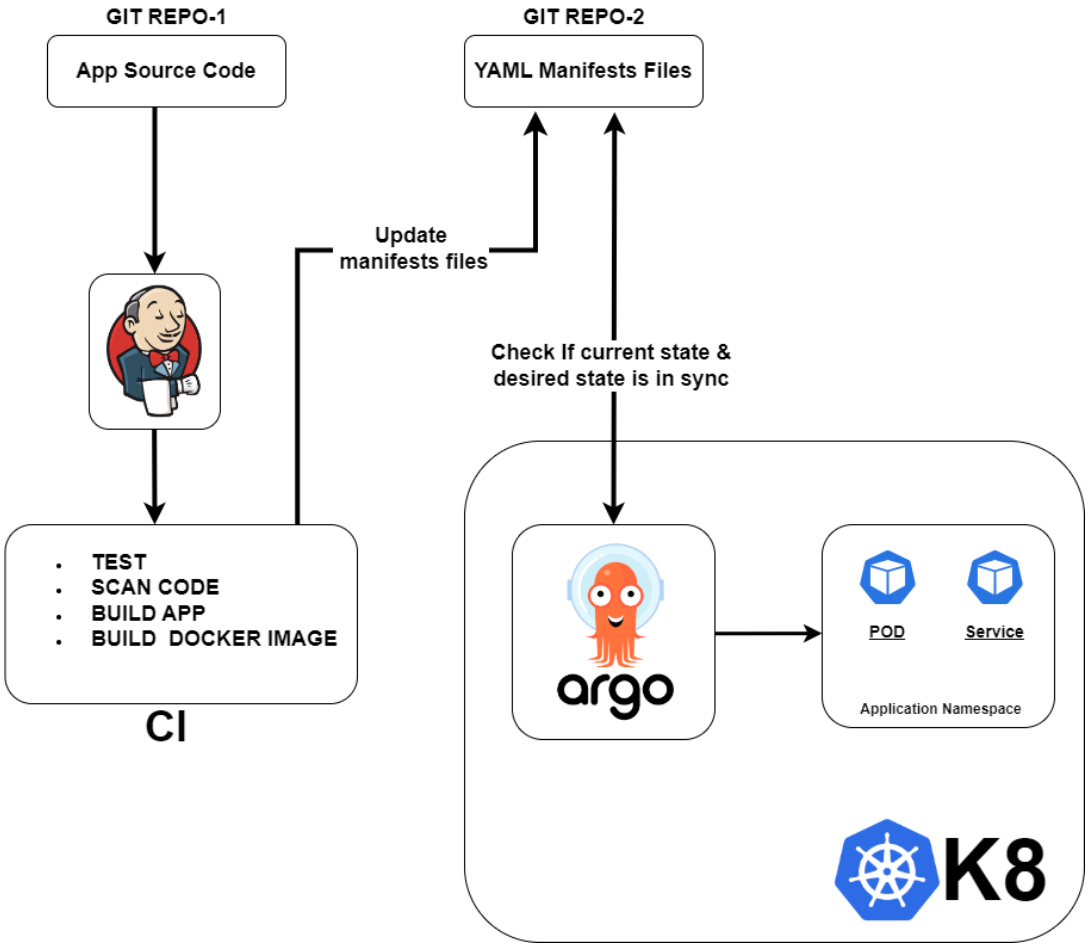
3. **Continuous Monitoring**: Argo CD continuously monitors Git repositories for changes in application configurations.

4. **Sync with Kubernetes Cluster**: When changes are detected, Argo CD syncs the desired state defined in the Git repository with the actual state of applications running on Kubernetes clusters.

5. **Deployment and Rollback**: Argo CD automates the deployment of applications based on the desired state. In case of failures, it supports automated rollbacks to previous known-good versions.

6. **User Interface and Monitoring**: Users can monitor deployment status and manage applications through the Argo CD user interface. They can also perform administrative tasks such as configuring access control and managing synchronization policies.

Diagram:



In this diagram:

- Application configurations are stored in a Git repository.
- Argo CD continuously monitors the Git repository for changes.
- When changes are detected, Argo CD synchronizes the desired state with the actual state of applications running on Kubernetes clusters.
- Argo CD automates the deployment and rollback processes based on the desired state.
- Users can monitor deployment status and manage applications through the Argo CD user interface.

HANDS-ON:

Create EKS Cluster From UI

1. **Create Role for EKS Cluster**:
    - Go to AWS Management Console.
    - Navigate to IAM (Identity and Access Management).
    - Click on "Roles" and then click on "Create role".
    - Choose "AWS Service" as the trusted entity.
    - Choose "EKS-cluster" as the use case.
    - Click "Next" and provide a name for the role.
2. **Create Role for EC2 Instances**:
    - Go to AWS Management Console.
    - Navigate to IAM (Identity and Access Management).
    - Click on "Roles" and then click on "Create role".
    - Choose "AWS Service" as the trusted entity.
    - Choose "EC2" as the use case.
    - Click "Next".
    - Add policies: [AmazonEC2ContainerRegistryReadOnly, AmazonEKS_CNI_Policy, AmazonEBSCSIDriverPolicy, AmazonEKSWorkerNodePolicy].
    - Provide a name for the role, e.g., "myNodeGroupPolicy".
3. **Create EKS Cluster**:
    - Go to AWS Management Console.
    - Navigate to Amazon EKS service.
    - Click on "Create cluster".
    - Enter the desired name, select version, and specify the role created in step 1.
    - Configure Security Group, Cluster Endpoint, etc.
    - Click "Next" and proceed to create the cluster.
4. **Create Compute Resources**:
    - Go to AWS Management Console.
    - Navigate to Amazon EKS service.
    - Click on "Compute" or "Node groups".

- Provide a name for the compute resource.
- Select the role created in step 2.
- Select Node Type & Size.
- Click "Next" and proceed to create the compute resource.

5. **Configure Cloud Shell**:
   - Open AWS Cloud Shell or AWS CLI.
   - Execute the command:
   - `aws eks update-kubeconfig --name shack-eks --region ap-south-1`
   - Replace "shack-eks" with the name of your EKS cluster and "ap-south-1" with the appropriate region if different.

These steps should help you in setting up your EKS cluster along with necessary roles and compute resources.

Install ArgoCD

Here are the steps to install ArgoCD and retrieve the admin password:

1. **Create Namespace for ArgoCD**:

   ```
   kubectl create namespace argocd
   ```

2. **Apply ArgoCD Manifests**:

   ```
   kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.7/manifests/install.yaml
   ```

3. **Patch Service Type to LoadBalancer**:

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

4. **Retrieve Admin Password**:

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

These commands will install ArgoCD into the specified namespace, set up the service as a LoadBalancer, and retrieve the admin password for you to access the ArgoCD UI.