

1. Setup and Configuration

I. Create a Git Repository

I created a GitHub repository for this project called "Argocd-gitops" and pushed our application source code to it.

Repository link : <https://github.com/Raunak0604/Argocd-gitops>

II. Install Argo CD on Your Kubernetes Cluster

I've installed Argo CD following the official documentation

Doc Link : https://argo-cd.readthedocs.io/en/stable/getting_started/

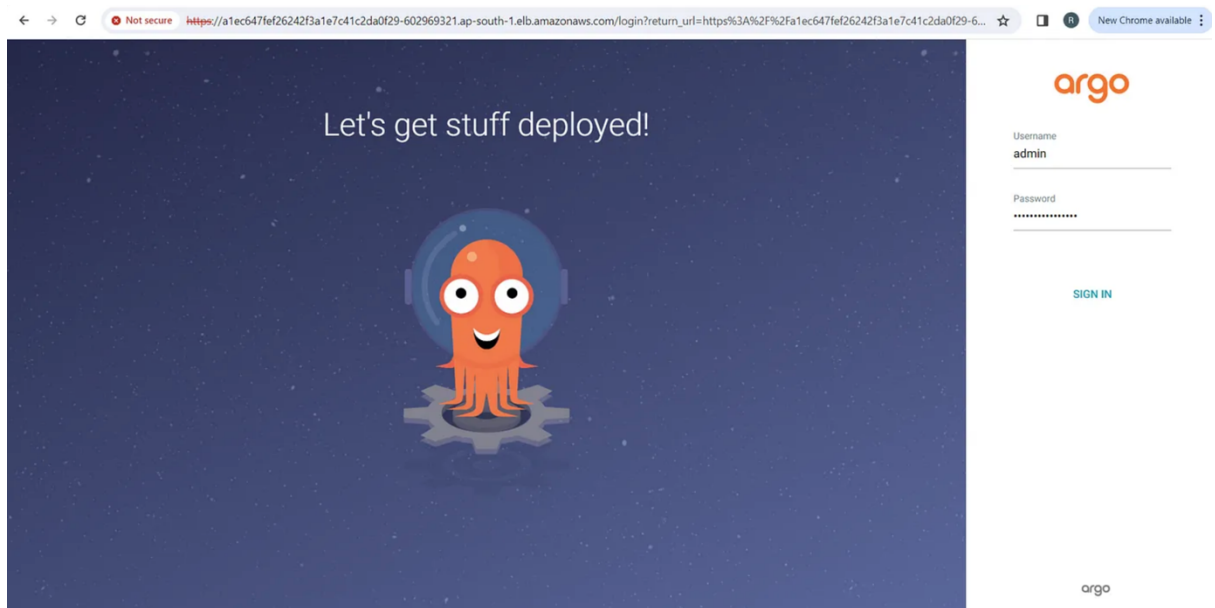
Install Argo CD Command:

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

All resources, such as pods, services, and deployments, are automatically deployed. I then edited the **argocd-server** service from **ClusterIP to LoadBalancer** to enable access to the Argo CD dashboard from outside the cluster

To retrieve the Argo CD login password, locate the **argocd-initial-admin-secret** that is automatically deployed. Describe its contents and decode it to base64 format.

Successfully login now



III. Install Argo Rollouts:

I've installed Argo **Rollouts** following the official documentation

Doc Link : <https://argoproj.github.io/argo-rollouts/installation/#kubectl-plugin-installation>

Install Argo Rollouts Command :

- `kubectl apply -f https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml`

Install Argo Rollouts Kubectl plugin Command :

- `curl -LO https://github.com/argoproj/argo-rollouts/releases/latest/download/kubectl-argo-rollouts-darwin-amd64`
- `chmod +x ./kubectl-argo-rollouts-darwin-amd64`
- `sudo mv ./kubectl-argo-rollouts-darwin-amd64 /usr/local/bin/kubectl-argo-rollouts`

2. Creating the GitOps Pipeline

I. Dockerize the Application :

I wrote a Dockerfile for my application, built the Docker image, and then pushed the image to **Docker Hub**.

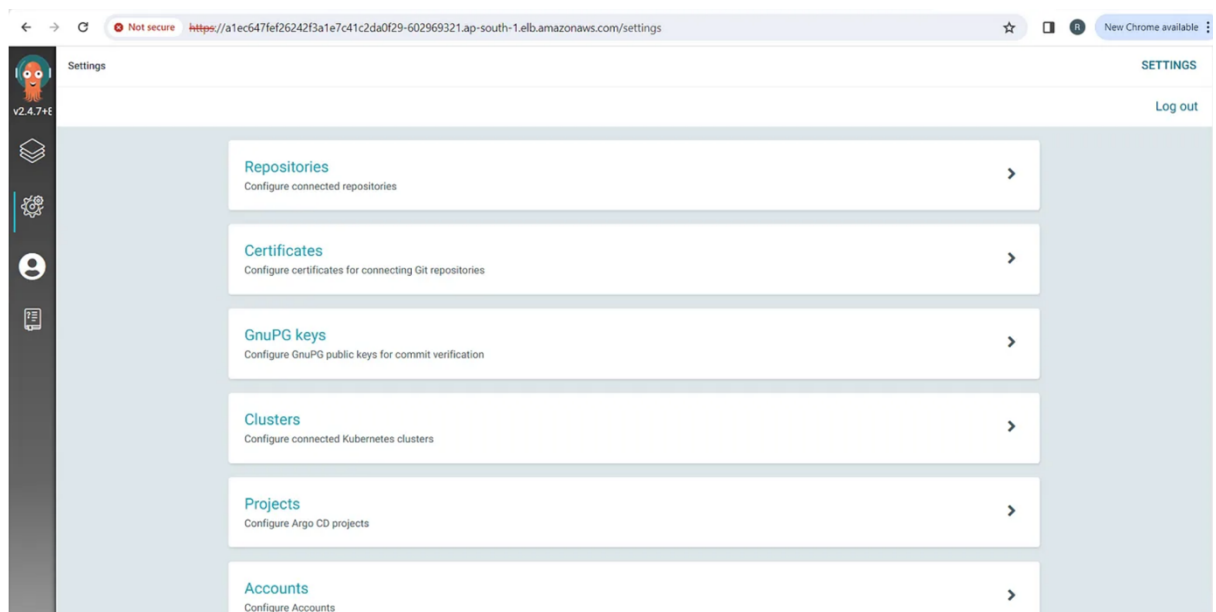
Docker Hub Image link :

<https://hub.docker.com/repository/docker/rabbit0604/app/general>

II. Deploy the Application Using Argo CD

Click on the gear icon in the left panel, then click on “**repository,**” and select “**Connect Repo Using HTTPS**”

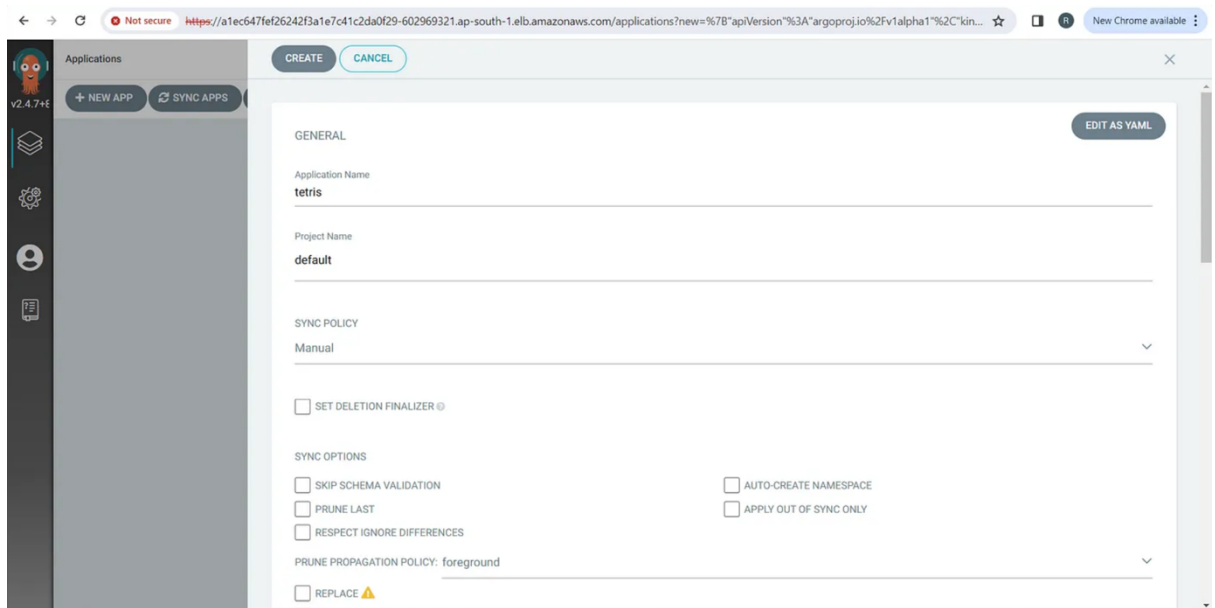
Provide your GitHub details and the repository URL of your project to connect.



Verify the connection status as **“Successful”**

Click on **“Manage Your application”**

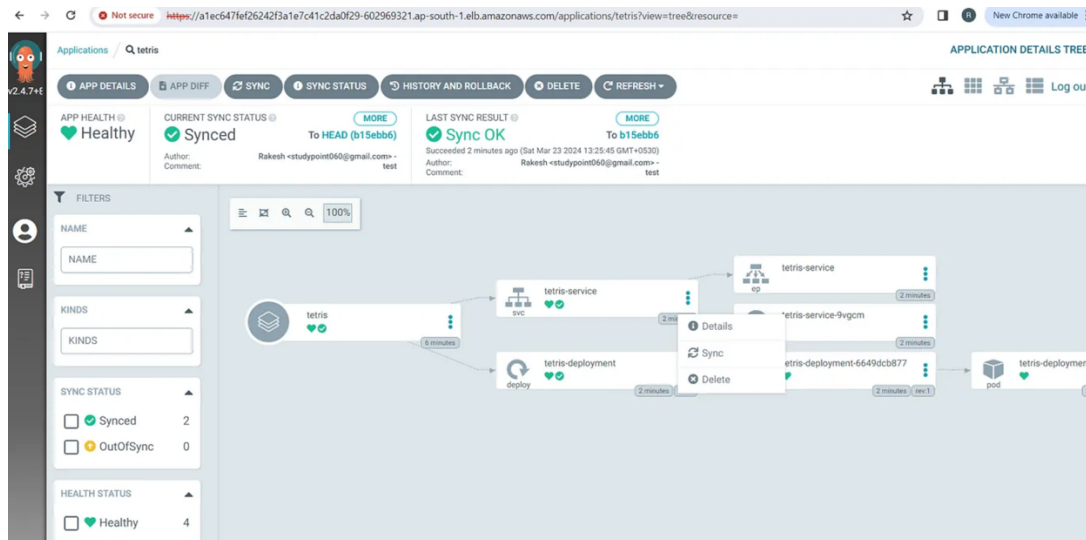
Click on **“New App”** and provide the necessary details.



The screenshot shows the 'New App' form in the AWS IAM console. The form is titled 'Applications' and has a 'CREATE' button. The 'Application Name' is 'tetris' and the 'Project Name' is 'default'. The 'SYNC POLICY' is set to 'Manual'. There are several checkboxes for 'SYNC OPTIONS': 'SET DELETION FINALIZER' (unchecked), 'SKIP SCHEMA VALIDATION' (unchecked), 'PRUNE LAST' (unchecked), 'RESPECT IGNORE DIFFERENCES' (unchecked), 'AUTO-CREATE NAMESPACE' (unchecked), and 'APPLY OUT OF SYNC ONLY' (unchecked). The 'PRUNE PROPAGATION POLICY' is set to 'foreground'. There is a 'REPLACE' checkbox with a warning icon. The form also has an 'EDIT AS YAML' button.

Put the details and Click on **“Create”**.

Access the deployed application by copying the load balancer address and pasting it into your browser.



3. Implementing a Canary Release with Argo Rollouts

write a yaml code for Argo Rollout with canary strategy

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: tetris-rollout
spec:
  replicas: 6
  strategy:
    canary:
      steps:
        - setWeight: 20
        - pause: {}
        - setWeight: 40
        - pause: {duration: 10}
        - setWeight: 60
        - pause: {duration: 10}
        - setWeight: 80
        - pause: {duration: 10}
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: tetris
  template:
```

```
metadata:
  labels:
    app: tetris
spec:
  containers:
    - name: tetris
      image: rabbit0604/app
      ports:
        - containerPort: 8000
```

Code link : <https://github.com/Raunak0604/Argocd-gitops/tree/main/K8s%20manifest>