Argo CD

Config repo: https://gitlab.com/nanuchi/argocd-app-config

Docker repo: https://hub.docker.com/repository/docker/nanajanashia/argocd-app

Install ArgoCD: https://argo-cd.readthedocs.io/en/stable/getting_started/#1-install-argo-cd

ArgoCD Configuration:
https://argo-cd.readthedocs.io/en/stable/operator-manual/declarative-setup/

Notification
https://argocd-notifications.readthedocs.io/en/stable/triggers/

install ArgoCD in k8s

```
kubectl create namespace argocd
kubectl apply -n argocd -f
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/i
nstall.yaml
```

access ArgoCD UI

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

login with admin user and below token (as in documentation):

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

(kubectl get secret argocd-initial-admin-secret -n argocd -o yaml)
we can change and delete init password

Create the application.yml and apply it

Application.yml

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
```

```yaml
 name: myapp-argo-application
 namespace: argocd
spec:
 project: default

 source:
   repoURL: https://gitlab.com/anupam.dutta753/argocd-app-config.git
   targetRevision: HEAD
   path: dev
 destination:
   server: https://kubernetes.default.svc
   namespace: myapp

 syncPolicy:
   syncOptions:
   - CreateNamespace=true

   automated:
     selfHeal: true
     prune: true
```

Deployment.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: myapp
spec:
 selector:
   matchLabels:
     app: myapp
 replicas: 2
 template:
   metadata:
     labels:
       app: myapp
   spec:
     containers:
     - name: myapp
       image: nanajanashia/argocd-app:1.2
       ports:
       - containerPort: 8080
```

Service.yml

```yaml
apiVersion: v1
kind: Service
metadata:
 name: myapp-service
spec:
 selector:
   app: myapp
 ports:
 - port: 8080
   protocol: TCP
   targetPort: 8080
```

This way, Argo CD will look for changes every 3 minutes. We can use webhooks to sync changes as soon as they are made.

Step 1: Configure GitLab Webhook

Go to GitLab repository's settings.
Navigate to `Settings` > `Integrations`.
Add a new webhook with the following details:

- URL: The URL of the Argo CD API server. For example, `http://argocd-server:8080/api/webhook`.
- Secret Token: Optionally, generate a secret token and configure it in both GitLab and Argo CD for added security.

Step 2: Configure Argo CD Application

Update Argo CD Application manifest (`myapp-argo-application.yaml`) to include the `webhook` section under the `syncPolicy`:

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
```

```
name: myapp-argo-application
namespace: argocd
spec:
project: default

source:
repoURL: https://gitlab.com/anupam.dutta753/argocd-app-config.git
targetRevision: HEAD
path: dev
destination:
server: https://kubernetes.default.svc
namespace: myapp

syncPolicy:
automated:
selfHeal: true
prune: true
webhook:
gitLab:
secretRef:
name: argocd-webhook-secret # Name of the secret containing the webhook token
```

Replace `argocd-webhook-secret` with the name of the Kubernetes Secret containing the webhook token.

Step 3: Create Secret with Webhook Token

Create a Kubernetes Secret containing the webhook token:

bash

Copy code

```
kubectl create secret generic argocd-webhook-secret
--from-literal=token=<webhook-token>
```

Replace `<webhook-token>` with the actual token generated in GitLab.

Step 4: Verify Webhook Configuration

After configuring the webhook in both GitLab and Argo CD, verify that it works correctly. Make changes to wer GitLab repository and observe whether Argo CD automatically syncs the application.

By following these steps, we can set up automatic synchronization in Argo CD using GitLab webhooks. This ensures that wer application is synchronized as soon as changes are pushed to the Git repository.

Notifications:

Prerequisites:

- Cluster Installed
- An ArgoCD cluster
- A Slack workspace

Step #1:Create a Slack App

Go to the Slack API page and create a new app.

Choose a name for wer app (e.g., “**ArgoCD Notifications**“) and select wer Slack workspace.

Under **OAuth & Permissions**

add the following scopes:

`chat:write`

`chat:write.`**customize**

Install the app to wer workspace

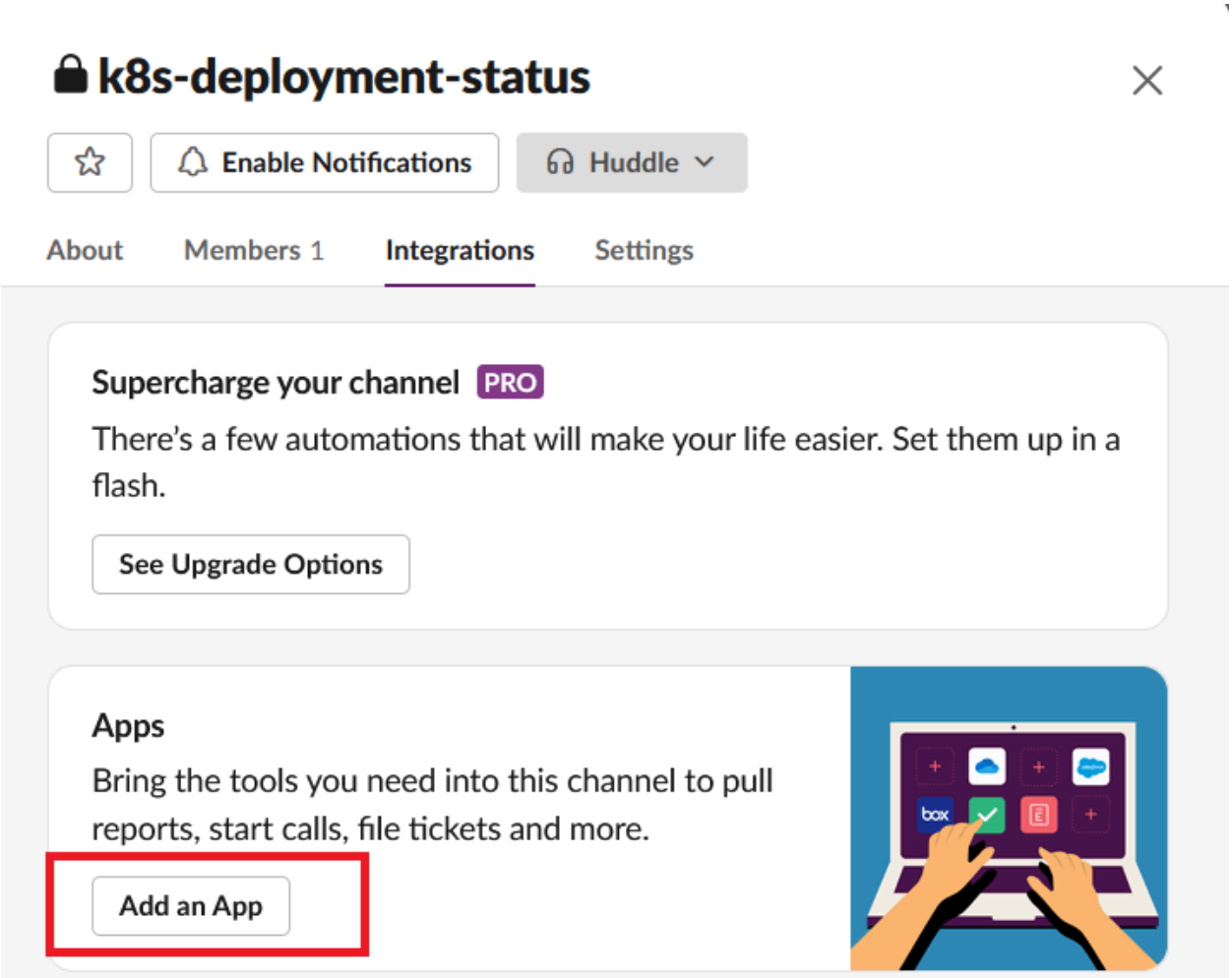
copy the generated Bot User OAuth Token.

Step #2:Create New Slack Channel

Create a New slack Channel For Ex. “k8s-deployment-status“

Click on Add an App to add above “ArgoCD Notifications” app into slack channel (Integrations)

Select the App and Click on “Add“.



Step #3:Install ArgoCD Notifications (if not already installed):

we can install ArgoCD Notifications using the provided installation method based on wer deployment (manifest, operator, or Helm) from ArgoCD Notifications official page.

Step #4:Configure ArgoCD Notifications with Slack

Edit the `argocd-notifications` service manifest and provide the following details or we can create new secret as shown below

Slack token: The Bot User OAuth Token from step 1.

Slack channel: The name of the channel where we want to receive notifications (e.g., "#k8s-deployment-status").

`sudo vi argocd-notifications-secret.yaml`

paste the below code, add slack token from **step 1**

```
apiVersion: v1

kind: Secret

metadata:

 name: argocd-notifications-secret
```

```
 namespace: argocd

stringData:

 slack-token: "xoxb-"
```

Apply the above secret yaml

```
Unset
kubectl apply -f argocd-notifications-secret.yaml
```

use the OAuth token to configure the Slack integration in the `argocd-notifications-secret` secret in configmap

```
Unset

sudo vi argocd-notifications-cm.yaml
```

Paste the below code.

```
apiVersion: v1

kind: ConfigMap

metadata:

 name: argocd-notifications-cm

data:
```

```
 service.slack: |

   token: $slack-token  # use as it is

 defaultTriggers: |

   - on-deployed

 trigger.on-deployed: |

   - description: Application is synced and healthy. Triggered
once per commit.

     oncePer: app.status.operationState.syncResult.revision

     send:

     - app-deployed

     when: app.status.operationState.phase in ['Succeeded'] and
app.status.health.status == 'Healthy' and app.status.sync.status
== 'Synced'

 template.app-deployed: |

   message: |

     {{if eq .serviceType "slack"}}:white_check_mark:{{end}}
Application {{.app.metadata.name}} is now running new version of
deployments manifests.

   slack:

     attachments: |

       [{

         "title": "{{ .app.metadata.name}}",
```

```
"title_link":"{{.context.argocdUrl}}/applications/{{.app.metadat
a.name}}",

        "color": "#18be52",

        "fields": [

        {

          "title": "Sync Status",

          "value": "{{.app.status.sync.status}}",

          "short": true

        },

        {

          "title": "Repository",

          "value": "{{.app.spec.source.repoURL}}",

          "short": true

        },

        {

          "title": "Revision",

          "value": "{{.app.status.sync.revision}}",

          "short": true

        }

        {{range $index, $c := .app.status.conditions}}
```

```
        {{if not $index}},{{end}}

        {{if $index}},{{end}}

        {

          "title": "{{$c.type}}",

          "value": "{{$c.message}}",

          "short": true

        }

        {{end}}

        ]

      }]
```

apply the above yaml

```
Unset

kubectl apply -f argocd-notifications-cm.yaml
```

Step #4:Configure Application Notifications (optional):

Annotations can be added to individual applications within ArgoCD to specify notification preferences:

`argocd-notifications.argoproj.io/channels`: Comma-separated list of Slack channels to notify.

Other annotations like `argocd-notifications.argoproj.io/kinds` and `argocd-notifications.argoproj.io/resources` can be used to filter notifications based on resource types and kinds.

Create Application in ArgoCD and add slack channel to test argocd notifications with slack channel

```
apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

 name: go-app

 namespace: argocd

 annotations:

   notifications.argoproj.io/subscribe.on-deployed.slack:
k8s-deployment-status

   notifications.argoproj.io/subscribe.on-sync-failed.slack:
k8s-deployment-status

   notifications.argoproj.io/subscribe.on-sync-succeeded.slack:
k8s-deployment-status

 finalizers:

   - resources-finalizer.argocd.argoproj.io

spec:
```

```
 destination:

   name: 'in-cluster'

   namespace: 'default'

 source:

   path: 'chart'

   repoURL:
'https://github.com/devopshint/gitops-workflow-deploy-goapp-mini
kube-using-argocd'

   targetRevision: HEAD

   helm:

     valueFiles:

     - values.yaml

 project: 'default'

 syncPolicy:

   syncOptions:

     - CreateNamespace=false
```

Apply the above Application in ArgoCD, once application is deployed, we will receive deployment status in slack channel