UEFI

[illegible]

How to configure Argo CD?

1) Deploy ArgoCD into K8s cluster
2) Configure ArgoCD with K8s YAML file ⟹ Extends the K8s API with Custom Resource Definitions

→ Main Resource is "Application"
↳ application.yaml ⟷ which Git repository? / which K8s cluster

```
apiVersion: argoproj.io/v1alpha1
Kind: Application
metadata:
  name: myapp-argo-application
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://gitlab.com/nanuchi/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
```

← This Kubernetes clusters
This could be the cluster where argocd itself is running in but it could also be an external cluster that argocd manages

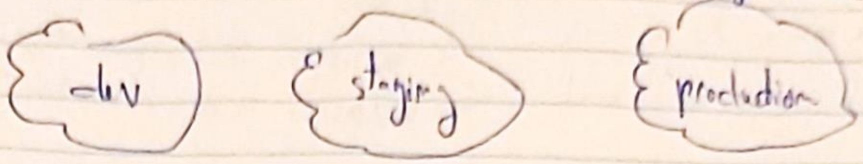
⟹ You can configure multiple such applications for different micro services for example for your cluster and if some applications belong together You can group them in another CRD called AppProject,

↳ Logically group applications
↳ and set restrictions
  ↳ what git repos may be deployed
  ↳ where app may be deployed

ArgoCD & Multi-cluster Setups?

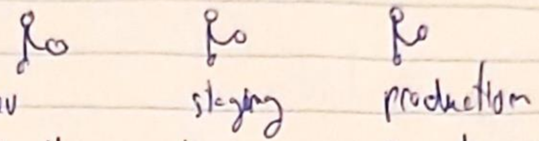
├→ Configure and manage just 1 ArgoCD
└→ Same ArgoCD instance is able to sync a fleet of K8s clusters

Multi-cluster environment : in each cluster there is ArgoCD deployed



1) Git branch for each environment

not the best options →



⇒ The better option would be to use overlays customiz where you have your own context for each environment

```
/myapp-clust
├→ base
│    ├─ deployment.yaml
│    ├→ kustomization.yaml
│    ├→ rbac-yaml
│    └→ service.yaml
└──→ overlays
       ├─ developent
       │    └→ Kustomization.yaml
       ├→ staging
       │    └→ Kustomization.yaml
       └→ production
              └→ Kustomization.yaml
```

⇒ with overlays you can reuse the same base yaml files and then selectively change specific parts in them for different environments

Argo CD → Replacement for
↳ Continuous Delivery
↳ Specifically for Kubernetes

~~stp~~ steps
1) install Argo CD in K8's cluster
2) Configure Argo CD with 'Application' CRD
3) Test our setup by updating Deployment.yaml file

→ Kubectl create namespace argocd
Kubectl apply -n argocd -f https://co- — — yaml

access Argo CD UI
↳ Kubectl get svc -n argocd
↳ argocd-server → accessible on :80/TCP
443/TCP
↳ Kubectl port-forward -n argocd
svc/argocd-server 8080:443

Kubectl get secret argocd-initial-admin-secret
-n argocd -o yaml

echo "password base64" | base64 --decode
↳
→ Configure Argo CD

→

```
application.yaml
└ apiVersion: argoproj.io/v1alpha1
  Kind: Application
  metadata:
    name: myapp-argo-application                    *****
    namespace: argocd
  spec:
   project: default
   source:
     repoURL: https://gitlab.com/name——/git
     targetRevision: HEAD ← last commit ~~the~~ git repo
     path: dev ← lets us specify whether we want to sync or
                 track a specific path in that repository
   destination:
     server: https://kubernetes.default.svc ← internal service name
                                              of Kubernetes api server
     namespace: myapp ← which namespace should
                        argo CD apply the changes
   syncPolicy:
     syncOptions:
      - CreateNamespace=true

     automated: ←———————— (Enable automatic sync
                             any change the git repo
       selfHeal: true  } ← ArgoCD polls Git
       prune: true        repository every 3 minutes
                          └ if you don't want this delay
                            you can configure a git webhook
```

═══

```
Kubectl apply -f application.yaml
```

═══

```
Kubectl edit deployment -n myapp myapp
```