



ArgoCD

Déployer ses ressources



Programme

Déployer ses ressources

Présentation de ArgoCD

- Architecture d'ArgoCD

Installation et configuration d'ArgoCD

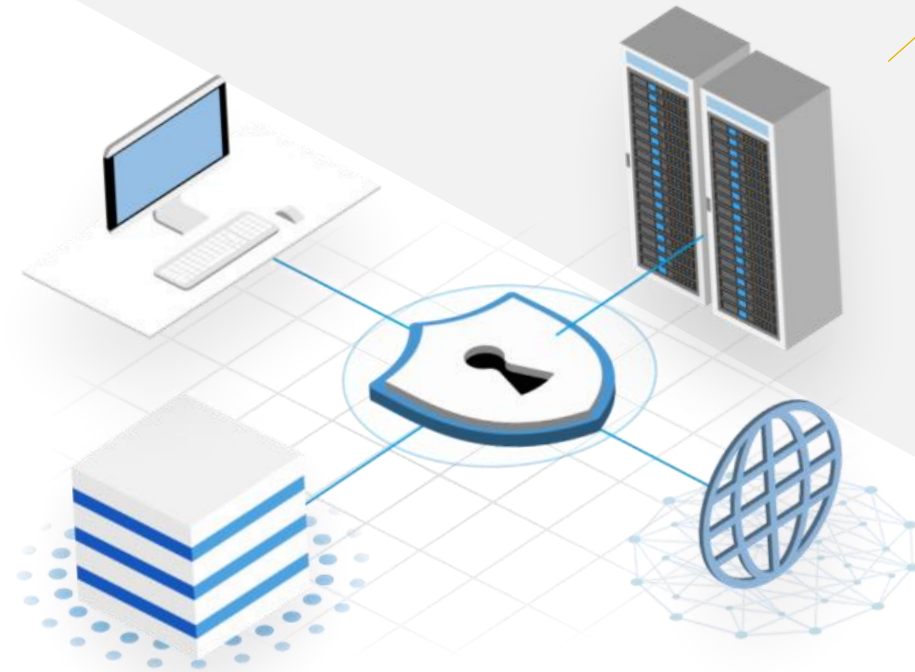
- Installation sur Kubernetes
- Configuration initiale

Applications et projets dans ArgoCD

- Création et gestion des applications
- Gestion des projets

Déploiement d'applications

- Utilisation de manifestes YAML
- Suivi des déploiements



Programme

Déployer ses ressources

Personnalisation et Optimisation

- Paramétrisation des applications

Sécurité et gestion des secrets

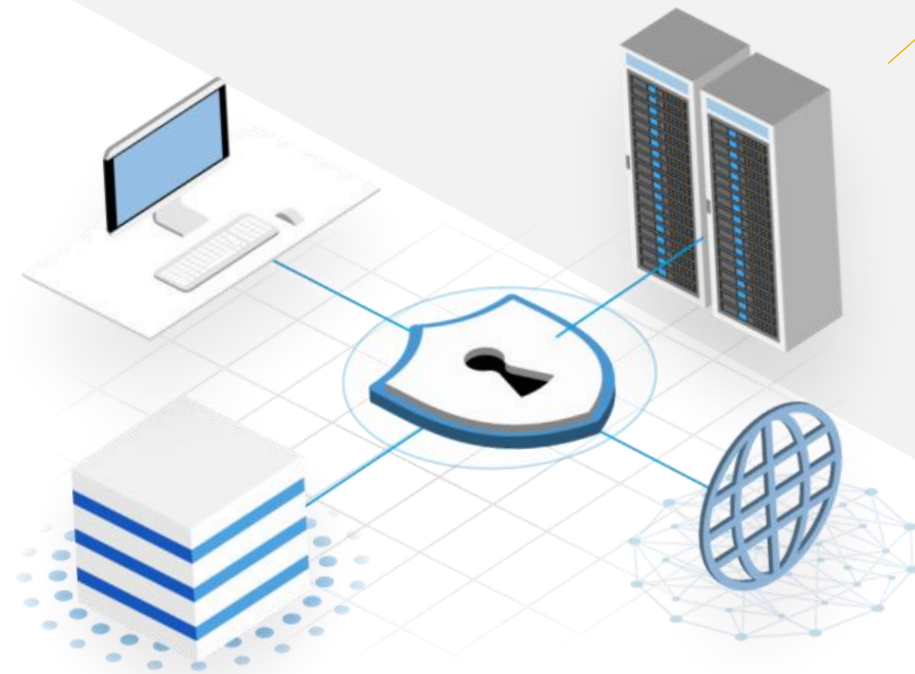
- Sécuriser ArgoCD
- Gestion des secrets dans les déploiements

Bonnes pratiques d'utilisation d'ArgoCD

- Stratégies de déploiement
- Gestion de la configuration à grande échelle

Intégration avec l'écosystème

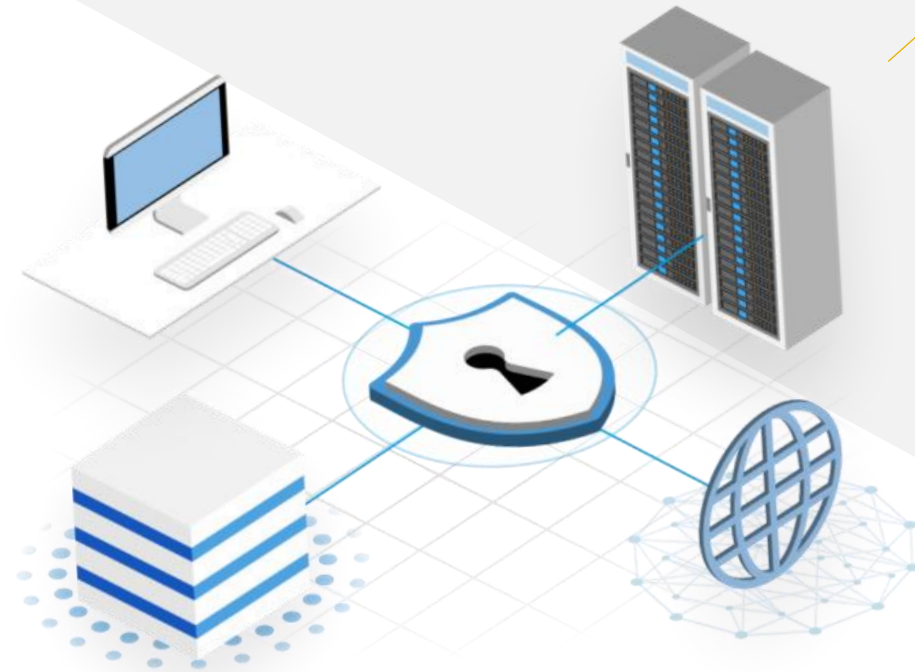
- CI/CD avec ArgoCD



Objectif

Déployer ses ressources

L'objectif de cette formation est d'acquérir une compréhension profonde et pratique d'ArgoCD et des principes GitOps.



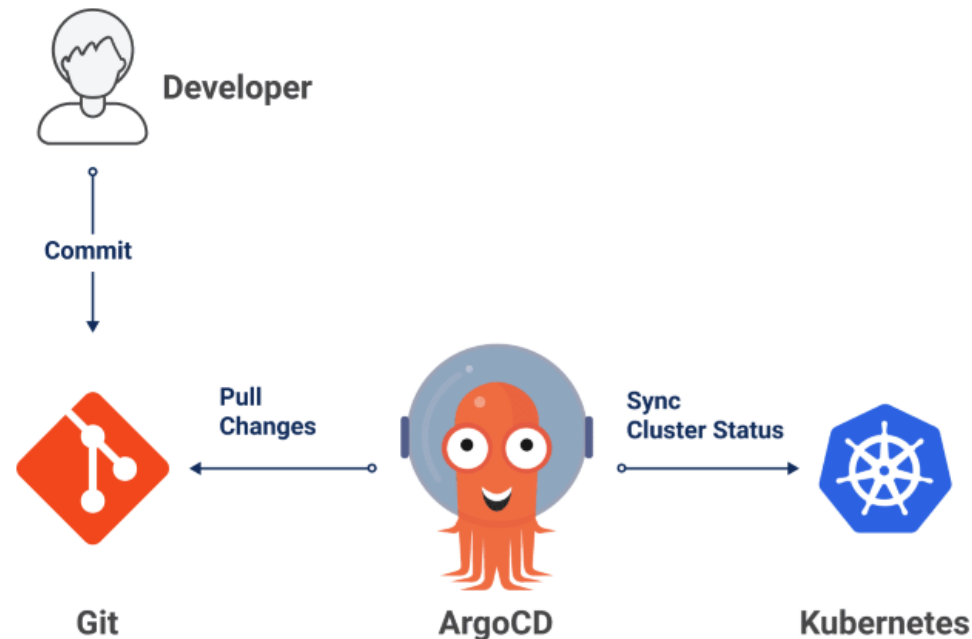
Présentation d'ArgoCD



ArgoCD

Présentation

ArgoCD, une pierre angulaire de l'écosystème GitOps, a été conçu pour répondre aux défis spécifiques du déploiement continu et de la gestion de configuration dans des environnements Kubernetes.



ArgoCD

Les différentes versions :

Argo Workflows : Le premier projet Argo a été Argo Workflows, qui a été créé en 2017 par Applatrix, une start-up spécialisée dans le *cloud computing*. Argo Workflows a été publié en *open source* en janvier 2018.

ArgoCD : En août 2018, Argo a lancé un deuxième projet appelé Argo CD, qui est une solution de déploiement continue (CD) pour Kubernetes. Argo CD a été publié en *open source* en novembre 2018.

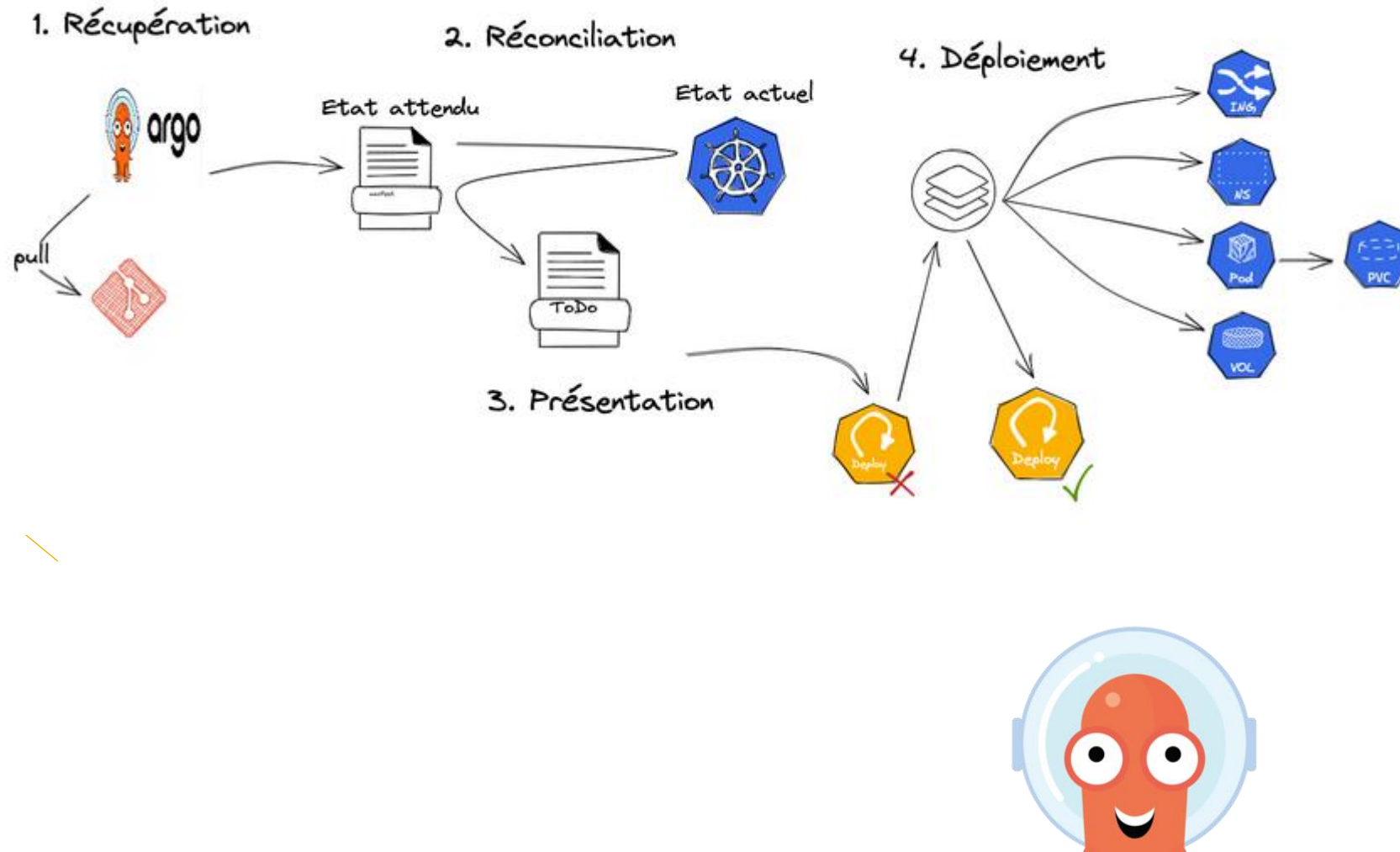
Argo Events : En février 2019, Argo a annoncé Argo Events, un projet qui fournit une infrastructure d'événements pour Kubernetes. Argo Events a été publié en *open source* en mars 2019.

Argo Rollouts : En mai 2019, Argo a lancé Argo Rollouts, une solution de déploiement Canary pour Kubernetes. Argo Rollouts a été publié en *open source* en septembre 2019.



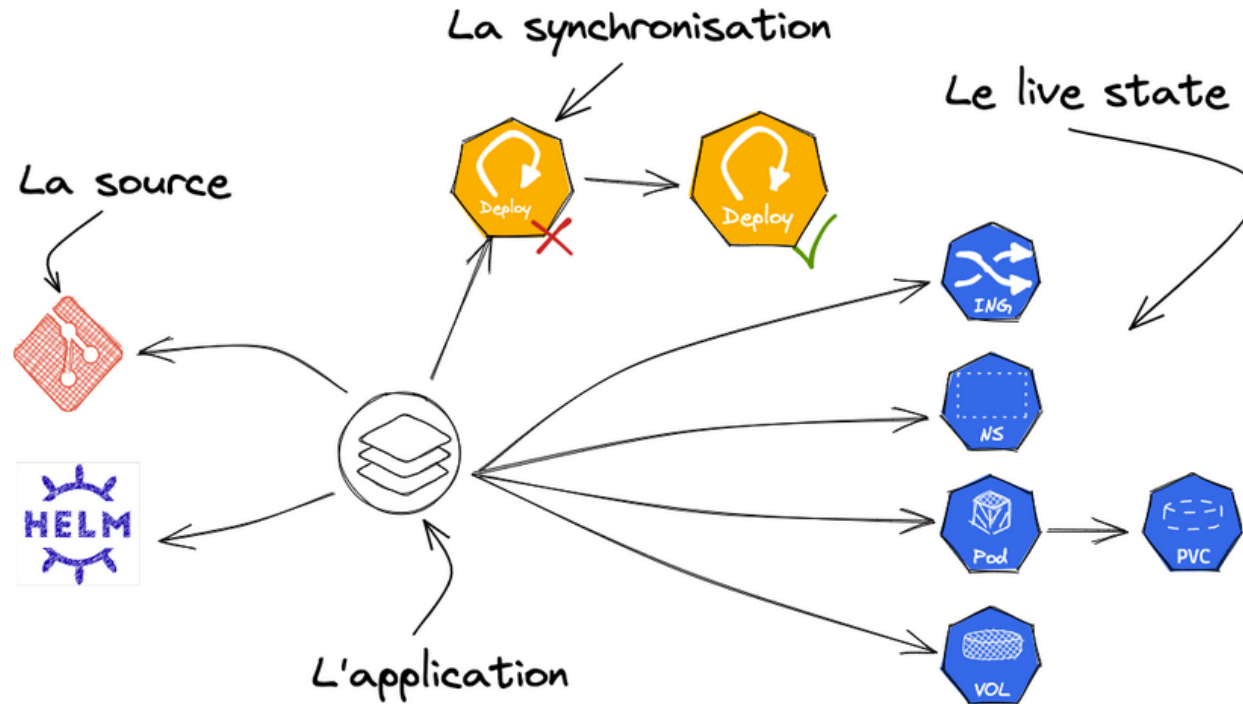
ArgoCD

Son fonctionnement



ArgoCD

Son fonctionnement



ArgoCD

Installation ArgoCD :

Prérequis :

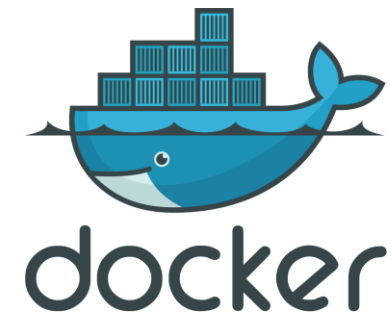
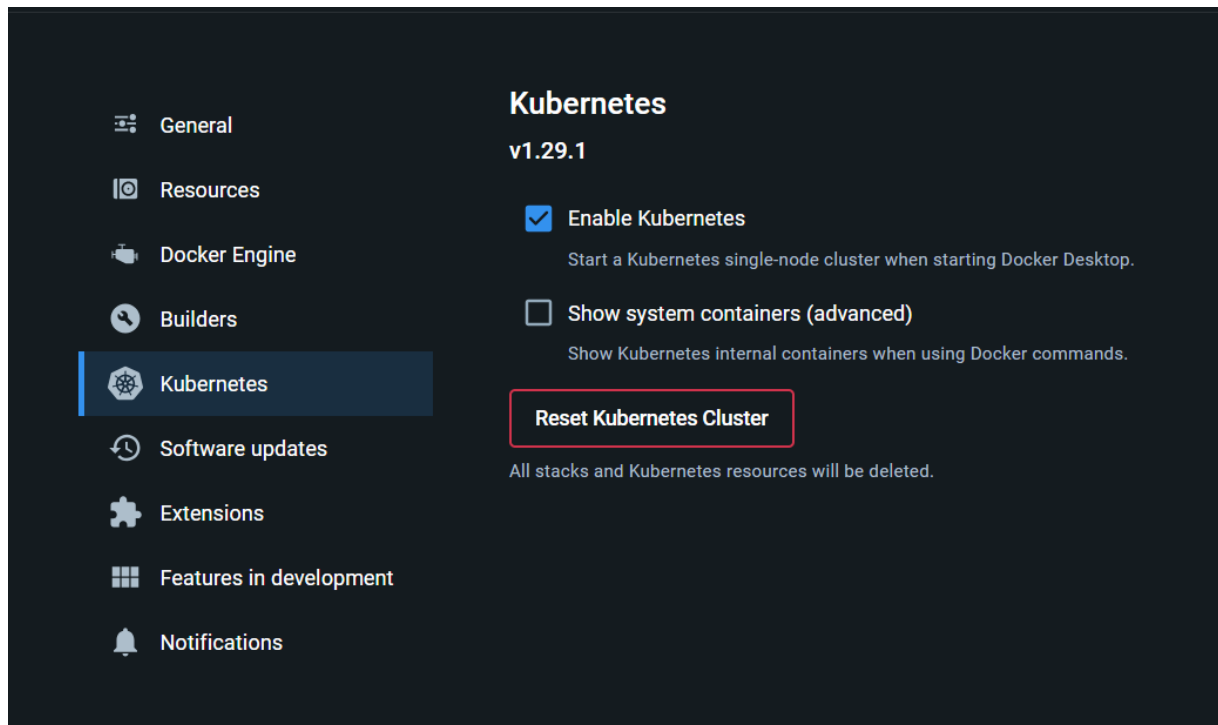
- Un cluster Kubernetes
- Kubectl installé sur le poste
- Un compte Github et un référentiel de code disponible



ArgoCD

Rappel : Docker et Kubernetes

Vous pouvez installer Kubernetes à l'intérieur de votre Docker



ArgoCD

Rappel : Installer Kubectl sur Windows

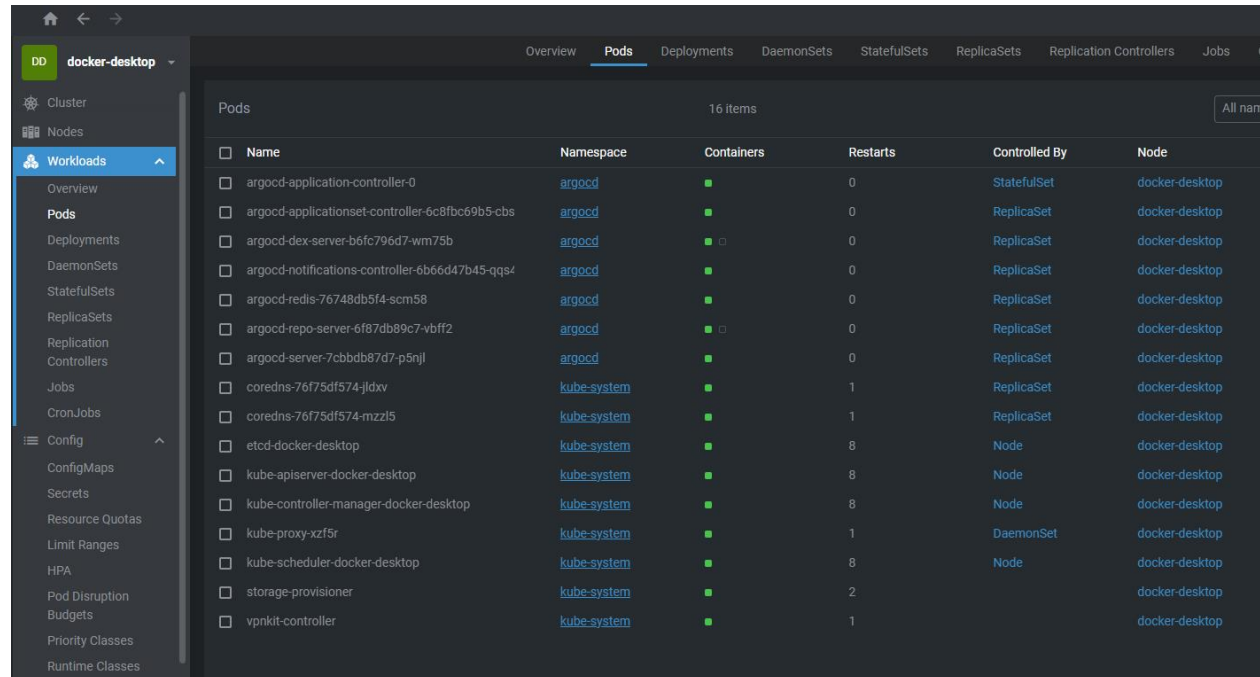
Vous pouvez installer l'exécutable **Kubectl** avec **Scoop** (le paquet manager).



ArgoCD

Rappel : Installer Openlens sur Windows

Vous pouvez installer **Openlens** afin de visualiser vos ressources plus facilement dans votre cluster kubernetes que celui-ci soit en local ou sur le cloud.



Name	Namespace	Containers	Restarts	Controlled By	Node
argocd-application-controller-0	argocd		0	StatefulSet	docker-desktop
argocd-applicationset-controller-6c8fbc69b5-cbs	argocd		0	ReplicaSet	docker-desktop
argocd-dex-server-b6fc796d7-wm75b	argocd		0	ReplicaSet	docker-desktop
argocd-notifications-controller-6b66d47b45-qqs4	argocd		0	ReplicaSet	docker-desktop
argocd-redis-76748db5f4-scm58	argocd		0	ReplicaSet	docker-desktop
argocd-repo-server-6f87db89c7-vbff2	argocd		0	ReplicaSet	docker-desktop
argocd-server-7cbbdb87d7-p5njl	argocd		0	ReplicaSet	docker-desktop
coredns-76f75df574-jldxv	kube-system		1	ReplicaSet	docker-desktop
coredns-76f75df574-mzzl5	kube-system		1	ReplicaSet	docker-desktop
etcd-docker-desktop	kube-system		8	Node	docker-desktop
kube-apiserver-docker-desktop	kube-system		8	Node	docker-desktop
kube-controller-manager-docker-desktop	kube-system		8	Node	docker-desktop
kube-proxy-xzf5r	kube-system		1	DaemonSet	docker-desktop
kube-scheduler-docker-desktop	kube-system		8	Node	docker-desktop
storage-provisioner	kube-system		2		docker-desktop
vpkkit-controller	kube-system		1		docker-desktop

ArgoCD

Installation

En passant par Kubectl :



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

En passant par Helm :



```
helm install my-release oci://registry-1.docker.io/bitnamicharts/argo-cd
```



ArgoCD

Installation : Accéder à l'interface d'ArgoCD

Une fois installé vous aurez accès à l'interface UI d'ArgoCD. Celui-ci peut également s'utiliser à l'aide d'un CLI.

Pour accéder à l'interface UI d'ArgoCD :

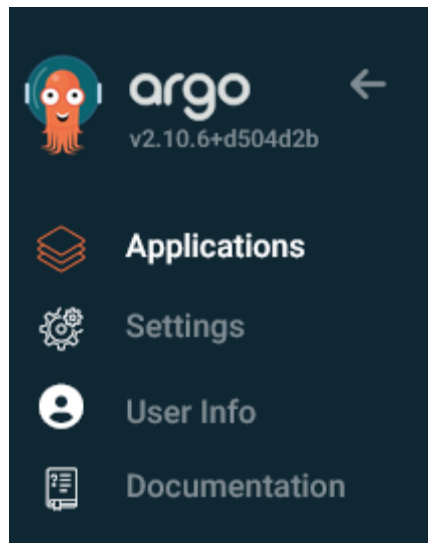
- Faire un port forwarding « argocd-server-xxx »
- Retrouver le mot de passe initial

```
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | %  
{[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($_))}
```

ArgoCD

Installation : Présentation de l'interface d'ArgoCD

Sur la gauche votre menu comprenant les applications ArgoCD ainsi que les différents réglages et documentation



Une application ArgoCD est une définition déclarative qui décrit l'état désiré d'une application déployée via ArgoCD dans un environnement Kubernetes. En gros, c'est une spécification qui indique à ArgoCD comment déployer, mettre à jour, et gérer une application dans un cluster Kubernetes.

ArgoCD

Installation : Présentation de l'interface d'ArgoCD

Sur la partie haute vous retrouverez une bannière comprenant les manipulations essentiels d'ArgoCD.



New App : Ce bouton est utilisé pour créer une nouvelle application dans ArgoCD. En cliquant dessus, vous êtes guidé à travers un formulaire où vous devez spécifier les détails de votre application, tels que le nom de l'application, le dépôt source du code, la révision (branch, tag, commit), le chemin dans le dépôt, le cluster cible, et l'espace de noms (namespace) où l'application sera déployée. C'est l'étape initiale pour déployer une application via ArgoCD.

ArgoCD

Installation : Présentation de l'interface d'ArgoCD

Sur la partie haute vous retrouverez une bannière comprenant les manipulations essentiels d'ArgoCD.



Sync App : Après avoir défini une application et son état désiré dans ArgoCD, il se peut que l'état actuel du cluster ne corresponde pas à cet état désiré (par exemple, après une modification du code source). Le bouton "Sync App" permet de déclencher manuellement la synchronisation de l'application, c'est-à-dire d'appliquer les changements nécessaires pour que l'état actuel du déploiement dans Kubernetes corresponde à l'état désiré spécifié. Cela inclut la création, la mise à jour, ou la suppression de ressources Kubernetes basées sur les différences détectées.

ArgoCD

Installation : Présentation de l'interface d'ArgoCD

Sur la partie haute vous retrouverez une bannière comprenant les manipulations essentiels d'ArgoCD.



Refresh App : Ce bouton permet de rafraîchir l'état d'une application dans l'interface utilisateur d'ArgoCD. Il ne modifie pas l'application elle-même dans le cluster Kubernetes, mais met à jour les informations affichées dans l'UI d'ArgoCD, telles que l'état actuel et les différences potentielles avec l'état désiré. C'est utile pour obtenir les informations les plus récentes, surtout si des modifications ont été appliquées directement sur le cluster ou si vous suspectez que l'UI n'affiche pas l'état actuel à jour.

ArgoCD

Première Application :

Nous allons créer notre première application ArgoCD.

Manipulation :

- Créer un référentiel de code sur github
- Créer un manifest de déploiement pour nginx
- Pousser votre manifest sur votre référentiel github

Rappel : Un **Deployment** en Kubernetes est une ressource qui vous permet de déclarer l'état désiré pour vos applications basées sur des conteneurs. Il s'occupe de gérer le déploiement et la mise à jour de vos instances de conteneurs (pods) en fonction de cette spécification.

ArgoCD

Première Application :

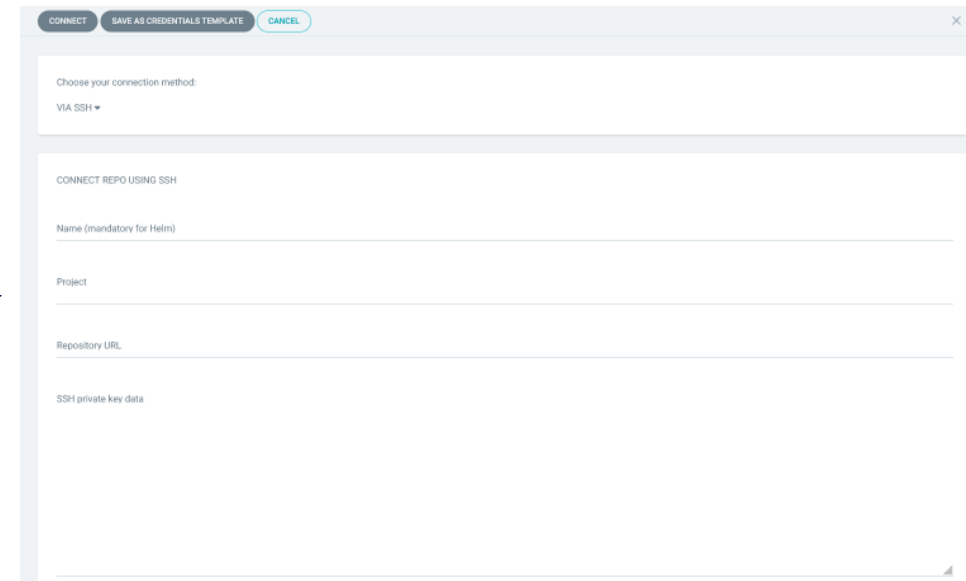
Créer votre **token** et connecter votre référentiel en allant dans **Settings > Repositories > + Connect Repo**

Manipulation :

- Afin qu'ArgoCD accède correctement à votre référentiel de code créer un token qu'il pourra utiliser

Repositories

Configure connected repositories

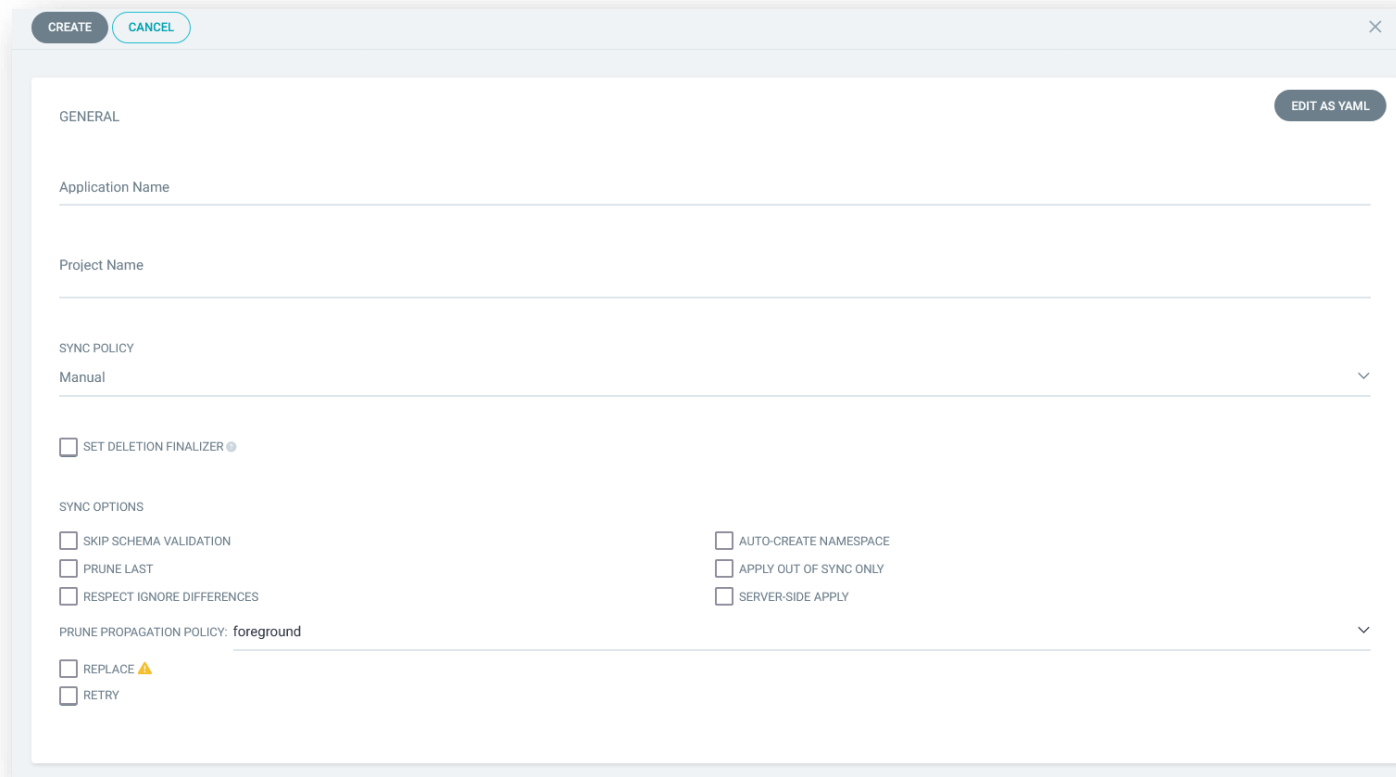


The screenshot shows a modal dialog titled 'CONNECT' with buttons for 'CONNECT', 'SAVE AS CREDENTIALS TEMPLATE', and 'CANCEL'. It prompts the user to 'Choose your connection method:' with a dropdown menu currently set to 'VIA SSH'. Below this, the 'CONNECT REPO USING SSH' section contains four input fields: 'Name (mandatory for Helm)', 'Project', 'Repository URL', and 'SSH private key data'.

ArgoCD

Première Application :

Vous pouvez alors créer votre première application ArgoCD



CREATE CANCEL

GENERAL EDIT AS YAML

Application Name

Project Name

SYNC POLICY

Manual

☐ SET DELETION FINALIZER

SYNC OPTIONS

☐ SKIP SCHEMA VALIDATION ☐ AUTO-CREATE NAMESPACE

☐ PRUNE LAST ☐ APPLY OUT OF SYNC ONLY

☐ RESPECT IGNORE DIFFERENCES ☐ SERVER-SIDE APPLY

PRUNE PROPAGATION POLICY: foreground

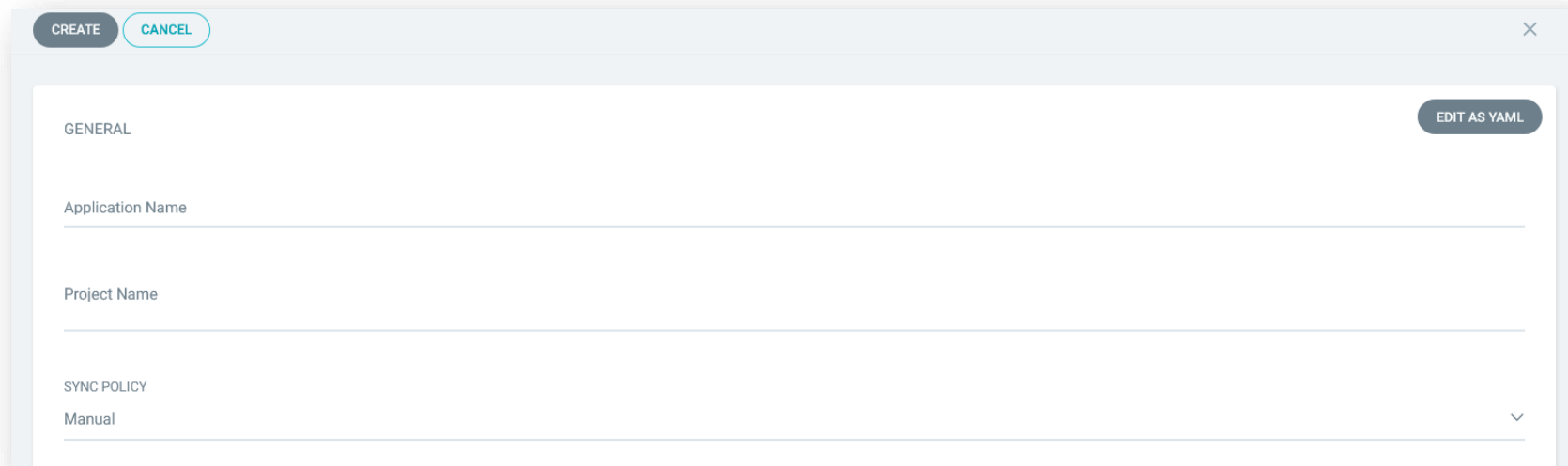
☐ REPLACE ☐ RETRY

ArgoCD

Première Application : Les différentes options

GENERAL :

- **Application Name:** Le nom unique de l'application dans Argo CD.
- **Project Name:** Le projet Argo CD auquel l'application appartient.
- **Sync Policy:** Détermine comment l'application est synchronisée par rapport à son état désiré dans le dépôt Git.
 - **Manual:** La synchronisation doit être déclenchée manuellement.
 - **Automatique :** La synchronisation se fait automatiquement.



The screenshot shows the 'CREATE' dialog for a new ArgoCD application. At the top, there are 'CREATE' and 'CANCEL' buttons. The form is titled 'GENERAL' and has an 'EDIT AS YAML' button in the top right corner. It contains two input fields: 'Application Name' and 'Project Name'. Below these, there is a 'SYNC POLICY' section with a dropdown menu currently set to 'Manual'.

ArgoCD

Première Application : Les différentes options

- **Sync Options:** Options de configuration pour la synchronisation, incluant :
 - **Skip Schema Validation:** Ignore la validation de schéma Kubernetes lors de la synchronisation.
 - **Auto-Create Namespace:** Crée automatiquement l'espace de noms dans le cluster
 - **Prune Last:** Supprime les ressources non désirées à la fin du processus de synchronisation.
 - **Apply Out of Sync Only:** Applique uniquement les changements pour les ressources qui ne sont pas déjà synchronisées.
 - **Respect Ignore Differences:** Ignore les différences spécifiées dans la configuration lors de la comparaison des ressources.
- **Server-Side Apply:** Utilise l'application côté serveur Kubernetes pour la synchronisation.
- **Prune Propagation Policy:** Politique de propagation pour la suppression des ressources, foreground signifie que la suppression est effectuée avant de supprimer la ressource parent.
- **Replace:** Option pour remplacer les ressources au lieu de les mettre à jour.
- **Retry:** Politique de réessai en cas d'échec de synchronisation.

ArgoCD

Première Application : Les différentes options

SOURCE

- **Repository URL:** L'URL du dépôt Git contenant le code source et/ou les configurations de l'application.

GIT

- **Revision:** La révision spécifique du dépôt à utiliser (commit SHA, tag, ou branche).
- **Branches:** Spécifie la branche du dépôt à utiliser.
- **Path:** Le chemin dans le dépôt où se trouvent les configurations spécifiques à l'application.

DESTINATION

- **Cluster URL:** L'URL du cluster Kubernetes cible pour le déploiement.
 - **URL:** Alternative à Cluster URL pour spécifier le cluster cible.
- **Namespace:** L'espace de noms dans le cluster où l'application sera déployée.

ArgoCD

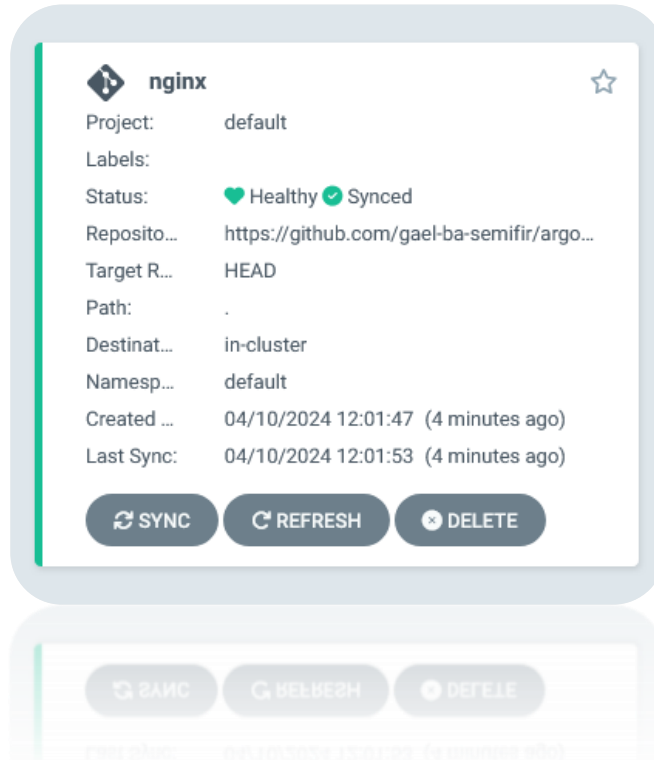
Première Application : Les différentes options

DIRECTORY

- **DIRECTORY RECURSE:** Spécifie si Argo CD doit chercher récursivement les configurations dans les sous-dossiers du chemin spécifié.
- **TOP-LEVEL ARGUMENTS:** Arguments supplémentaires au niveau supérieur pour la commande de déploiement.
- **EXTERNAL VARIABLES:** Variables externes qui peuvent être passées à la configuration.
- **INCLUDE/EXCLUDE:** Permet d'inclure ou d'exclure spécifiquement des fichiers ou des dossiers dans le processus de synchronisation.

ArgoCD

Résultat



nginx

Project: default

Labels:

Status: ♥ Healthy ✓ Synced

Repository: <https://github.com/gael-ba-semifir/argo...>

Target R...: HEAD

Path: .

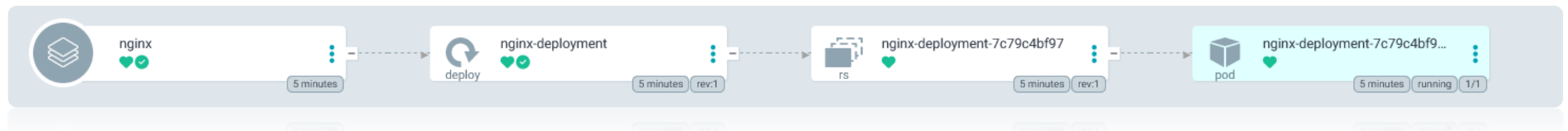
Destination: in-cluster

Namespace: default

Created ...: 04/10/2024 12:01:47 (4 minutes ago)

Last Sync: 04/10/2024 12:01:53 (4 minutes ago)

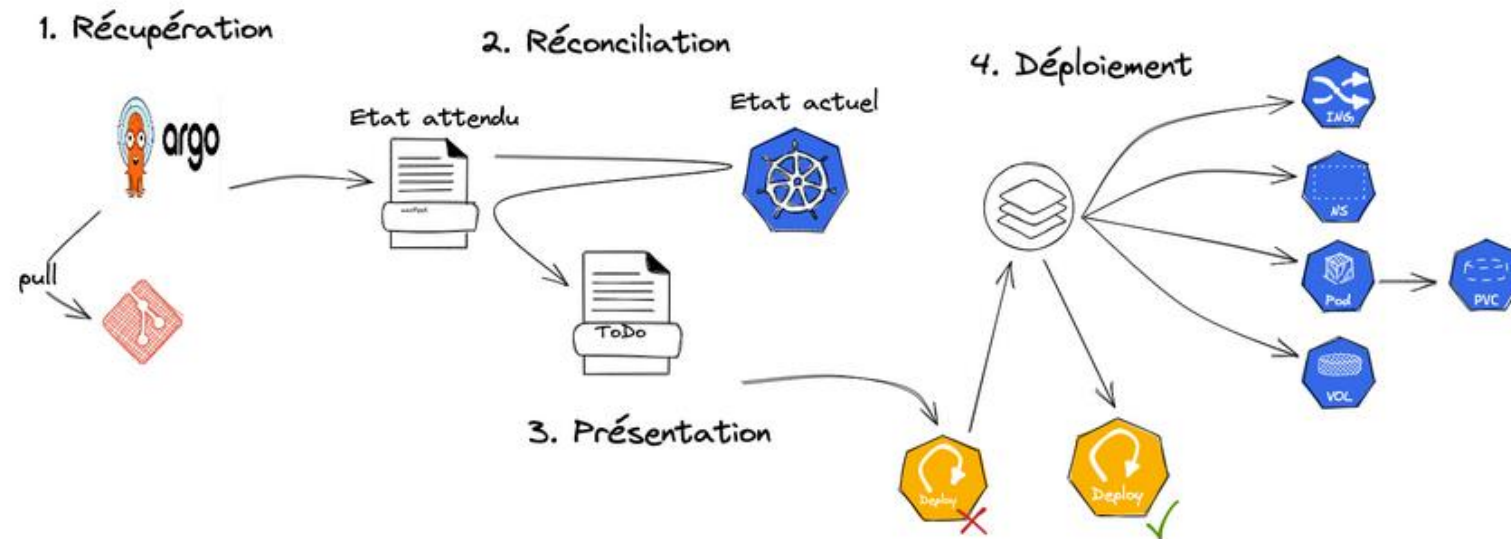
↻ SYNC ↻ REFRESH ✕ DELETE



ArgoCD

Résultat

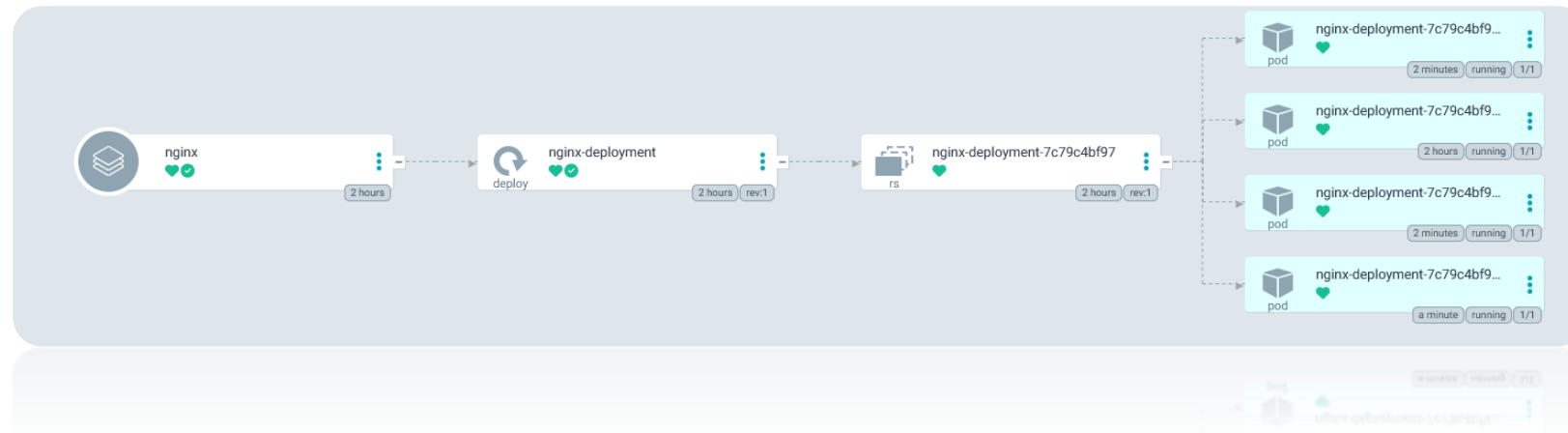
Nous avons raccordé notre ArgoCD à notre référentiel, il s'est occupé de déployer automatiquement notre application Nginx.



ArgoCD

Résultat

Changer votre nombre de réplikas dans votre manifeste de déploiement et pousser vos modifications sur Github.



Attention il faut cliquer sur le bouton : « **Refresh** » afin de mettre à jour l'UI de ArgoCD et de pouvoir observer les changements.

ArgoCD

Exercice :



Manipulation :

- Supprimer votre application ArgoCD et observer les changements

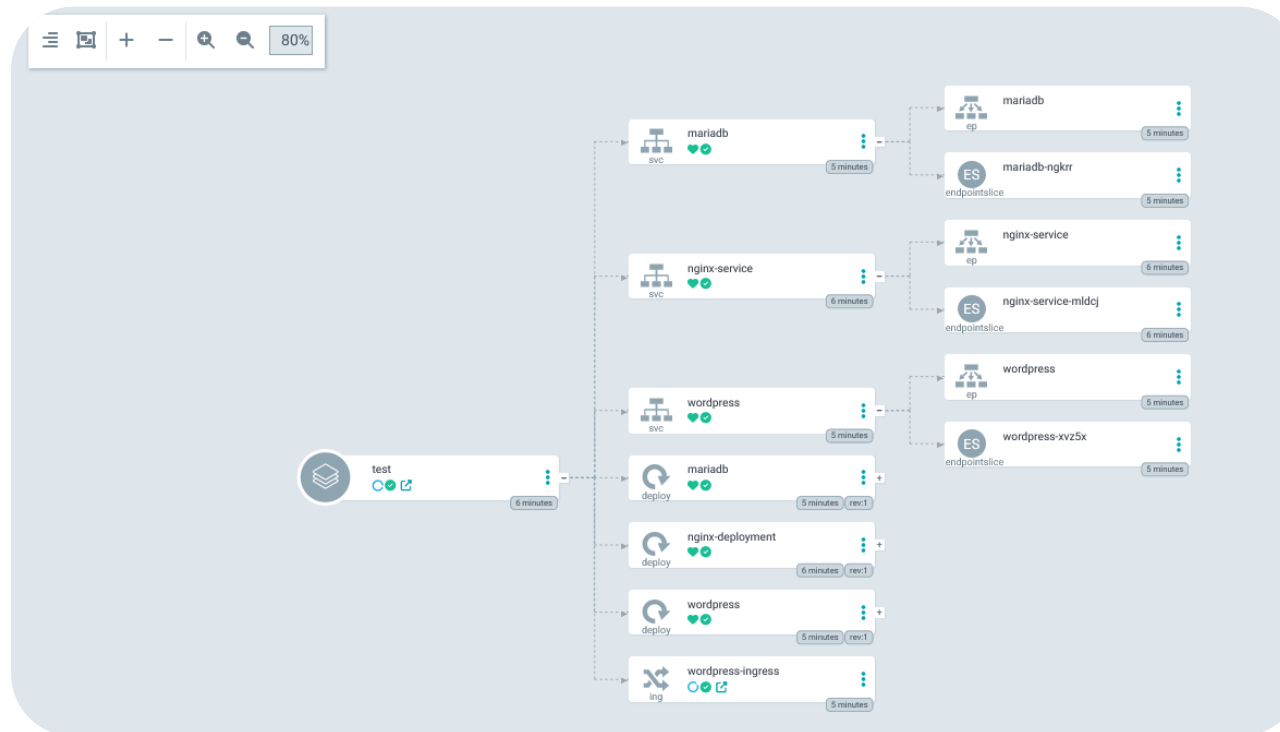
Exercice :

- Supprimer votre fichier nginx dans votre référentiel
- Créer une application ArgoCD qui déploiera à partir d'un même référentiel un ensemble d'application

ArgoCD

Correction :

Dans l'exemple de correction nous déployons un ensemble de manifestes différents dans notre cluster. ArgoCD peut déployer n'importe quel type de manifeste.



ArgoCD

Exercice:

Nous allons maintenant construire un exemple de pipeline CI/CD entier.

Manipulation :

- Créer une Application web
- Créer une pipeline CI qui buildera et poussera celle-ci sur un registry
- Créer une pipeline CD qui déploiera automatique à l'aide d'ArgoCD dans votre cluster la dernière image

Indice :

- Vous pouvez faire les deux dans un meme workflows
- Vous pouvez passer par le registry docker pour des raison de simplicités

ArgoCD

Correction :

- Création d'une application avec **vite** et **pnpm react**
- Création d'un Dockerfile de production
- Envois des modifications sur notre référentiel de code
- Mise en place de la partie CI sur notre docker registry



GitHub Actions

ArgoCD

Correction : Exemple de build vers Docker

```
name: Docker Image CI

on:
  push:
    branches: [ "main" ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3

      - name: Build the Docker image
        run: docker build ./cicd/reactCICD/ -t ${ secrets.DOCKER_USERNAME }}/reactcid:${ github.sha }

      - name: Log in to Docker Registry
        run: echo "${ secrets.DOCKER_TOKEN }}" | docker login docker.io -u ${ secrets.DOCKER_USERNAME } --password-stdin

      - name: Push the Docker image
        run: docker push ${ secrets.DOCKER_USERNAME }}/reactcid:${ github.sha }
```

Attention ne pas oublier de créer des secrets !

ArgoCD

Correction : Exemple de build vers Docker

- Création d'un dossier et d'un fichier de déploiement

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: <NOM_APP>
spec:
  replicas: 3
  selector:
    matchLabels:
      app: <NOM_APP>
  template:
    metadata:
      labels:
        app: <NOM_APP>
    spec:
      containers:
        - name: reactcid
          image: <NOM_DOCKER>/<NOM_IMAGE>:<TAG>
          ports:
            - containerPort: 80
```

ArgoCD

Correction : Exemple de build vers Docker

- Création de la partie CD et mise à jour du pipeline

```
- name: Checkout code for the update
  uses: actions/checkout@v3
  with:
    ref: main

- name: Update Deployment Image Tag
  run: |
    sed -i "s|${{ secrets.DOCKER_USERNAME }}/<NOM_APP>:.*|${{ secrets.DOCKER_USERNAME }}/
    <NOM_APP>:${{ github.sha }}|g" ./cicd/reactCICD/deployment/deployment.yaml
    git config --global user.email "action@github.com"
    git config --global user.name "GitHub Action"
    git add .
    git commit -am "Update image tag to ${{ github.sha }}" || echo "No changes to commit"
    git push origin main
```

Nous configurons ici, notre action pour mettre à jour automatiquement le tag de déploiement. N'oubliez pas d'accorder les droits aux actions !

ArgoCD

Correction : Exemple de build vers Docker

Il nous reste plus qu'à créer notre application ArgoCD, qui viendra observer notre fichier de déploiement !



ArgoCD

Les projets ArgoCD

Dans Argo CD, un **projet** est une entité fondamentale qui sert à regrouper et à gérer des ensembles d'applications déployées via Argo CD.

Les projets Argo CD sont conçus pour aider à organiser, sécuriser et gérer les déploiements de manière plus efficace.



ArgoCD

Les projets ArgoCD

Les projets fournissent les fonctionnalités suivantes :

- Restreindre ce qui peut être déployé (dépôts Git source de confiance)
- Restreindre où les applications peuvent être déployées (clusters et espaces de noms de destination)
- Restreindre quels types d'objets peuvent ou ne peuvent pas être déployés (par exemple, RBAC, CRDs, DaemonSets, NetworkPolicy, etc.)
- Définir des rôles de projet pour fournir un RBAC d'application (lié à des groupes OIDC et/ou des jetons JWT)

ArgoCD

Création d'un projet ArgoCD



Projects



Configure Argo CD projects



[Settings](#) / [Projects](#)

PROJECTS

[+ NEW PROJECT](#) [Log out](#)

NAME	DESCRIPTION
 default	
 online	

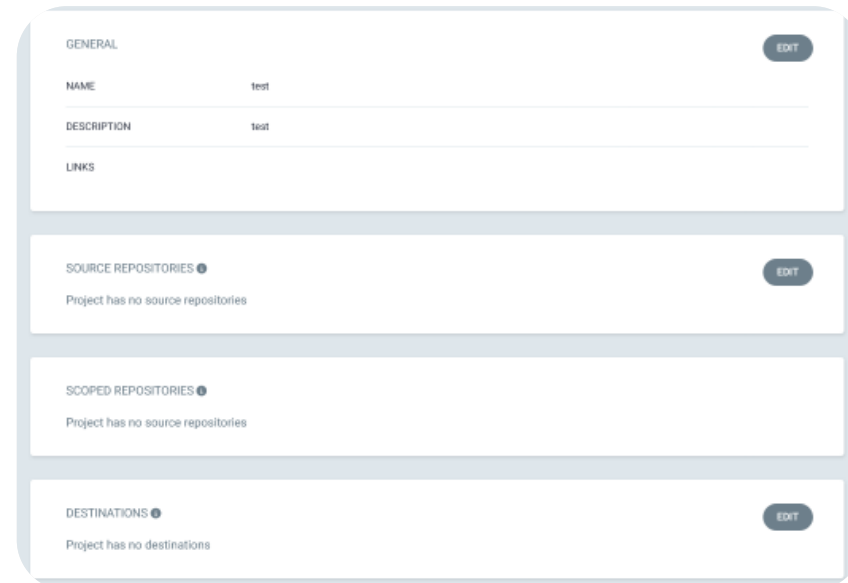


ArgoCD

Création d'un projet ArgoCD

GENERAL :

- **NAME:** Le nom unique attribué au projet Argo CD. C'est l'identifiant principal du projet.
- **DESCRIPTION:** Une description du projet, permettant de donner plus de contexte sur son objectif, son périmètre, ou les équipes impliquées.
- **LINKS:** Permet d'ajouter des liens externes associés au projet, comme des références à la documentation, des tableaux de bord, ou des outils de suivi des problèmes.



The screenshot displays the ArgoCD project configuration page. It features four main sections, each with an 'EDIT' button in the top right corner:

- GENERAL:** Contains fields for 'NAME' (value: test), 'DESCRIPTION' (value: test), and 'LINKS'.
- SOURCE REPOSITORIES:** Shows the message 'Project has no source repositories'.
- SCOPED REPOSITORIES:** Shows the message 'Project has no source repositories'.
- DESTINATIONS:** Shows the message 'Project has no destinations'.

ArgoCD

Création d'un projet ArgoCD

SOURCE REPOSITORIES

- Les dépôts sources sont les emplacements (URLs) des dépôts Git contenant les définitions des ressources Kubernetes à déployer.

SCOPED REPOSITORIES

- Cette option permet de définir des dépôts Git spécifiques autorisés uniquement pour ce projet, limitant ainsi l'accès aux ressources basées sur le projet.

DESTINATIONS

- Les destinations définissent où les applications peuvent être déployées, typiquement spécifiant des clusters Kubernetes et des espaces de noms.



ArgoCD

Création d'un projet ArgoCD

SCOPED CLUSTERS

- Spécifie les clusters Kubernetes et les espaces de noms qui sont explicitement autorisés ou restreints pour ce projet.

CLUSTER RESOURCE ALLOW LIST

- Cette liste permet de spécifier les types de ressources Kubernetes qui sont autorisés à être déployés par les applications de ce projet au niveau du cluster (par exemple, des CRDs ou des secrets au niveau du cluster).

CLUSTER RESOURCE DENY LIST

- Inverse de la liste d'autorisation, cette liste spécifie les types de ressources Kubernetes qui ne sont pas autorisés à être déployés par les applications de ce projet au niveau du cluster.



ArgoCD

Création d'un projet ArgoCD



Rôles de Projet

Cet onglet permet de définir des rôles spécifiques au projet, offrant ainsi une manière flexible et puissante de gérer les droits d'accès pour différentes équipes ou utilisateurs en fonction de leurs besoins.

Un "rôle" dans ce contexte est un ensemble de permissions qui déterminent ce qu'un utilisateur ou un groupe d'utilisateurs peut faire au sein du projet.

ArgoCD

Création d'un projet ArgoCD

SUMMARY

ROLES

WINDOWS

EVENTS

Politiques: Les politiques définissent les permissions spécifiques accordées par le rôle. Elles sont exprimées sous forme de règles qui spécifient ce qu'un utilisateur peut faire (par exemple, créer, mettre à jour, supprimer des applications) et sur quels objets ou ressources ces actions peuvent être effectuées.

Groupes OIDC/JWT: Permet d'associer le rôle à des groupes spécifiques d'utilisateurs authentifiés via OpenID Connect (OIDC) ou des jetons JSON Web Token (JWT). Cela signifie que les permissions définies dans le rôle s'appliqueront automatiquement à tous les utilisateurs membres de ces groupes.

ArgoCD

Comment fonctionne les rôles

SUMMARY

ROLES

WINDOWS

EVENTS

Actions: get, create, update, delete, sync, override, action/<group/kind/action-name>

Resources: clusters, projects, applications, applicationsets, repositories, certificates, accounts, gpgkeys, logs, exec, extensions



```
action : sur quoi : autoriser/ne pas autoriser
```

ArgoCD

Création d'un projet ArgoCD

SUMMARY

ROLES

WINDOWS

EVENTS

L'onglet **Windows** dans la configuration d'un projet Argo CD se réfère à des **fenêtres de synchronisation** (Sync Windows), qui sont des règles définissant quand les synchronisations d'applications peuvent ou ne peuvent pas se produire au sein d'un projet.

ArgoCD

Création d'un projet ArgoCD

SUMMARY

ROLES

WINDOWS

EVENTS

Exemple de Configuration

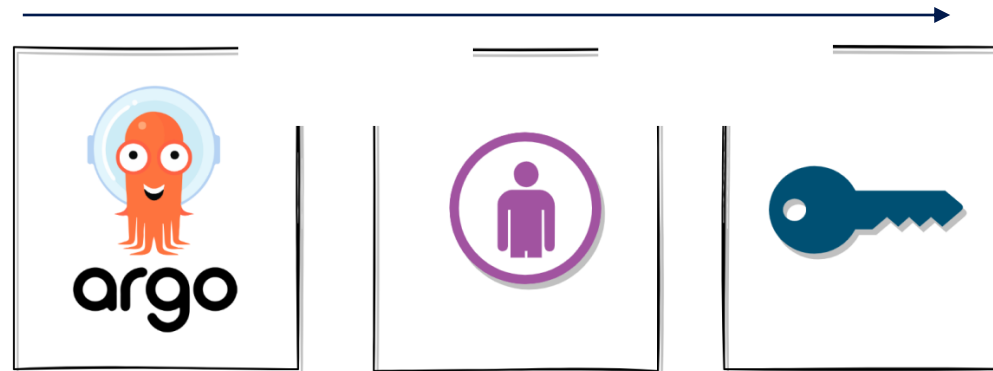
Supposons que vous vouliez éviter les déploiements automatiques pendant les heures de bureau pour réduire le risque d'interruptions. Vous pouvez configurer une fenêtre de synchronisation qui autorise les déploiements automatiques uniquement en dehors des heures de bureau, par exemple de 19h à 7h en semaine, et toute la journée pendant les week-ends.

ArgoCD

Démonstration d'un projet ArgoCD

Nous allons :

- Créer un projet ArgoCD qui s'appellera « pipelinecid »
- Créer deux rôles pour ce projet :
 - Un rôle « developpeur » qui n'aura des accès qu'en lecture seule
 - Un rôle « operationnel » qui aura tous les accès
- Créer deux utilisateurs avec mot de passe pour les deux rôles dans ArgoCD



ArgoCD

Démonstration d'un projet ArgoCD

GENERAL

Role Name
developpeur

Role Description
developpeur role for reading access

POLICY RULES

Manage this role's permissions to applications

ACTION	APPLICATION	PERMISSION	
get	pipelinecd/*	allow	×

ADD POLICY

GENERAL

Role Name
operationnel

Role Description
operationnel role for managing application

POLICY RULES

Manage this role's permissions to applications

ACTION	APPLICATION	PERMISSION	
*	pipelinecd/*	allow	×

ADD POLICY

ArgoCD

Démonstration d'un projet ArgoCD

Vous devriez avoir maintenant deux rôles présents dans votre projet !

+ ADD SYNC WINDOW

DELETE

SUMMARY

ROLES

WINDOWS

EVENTS

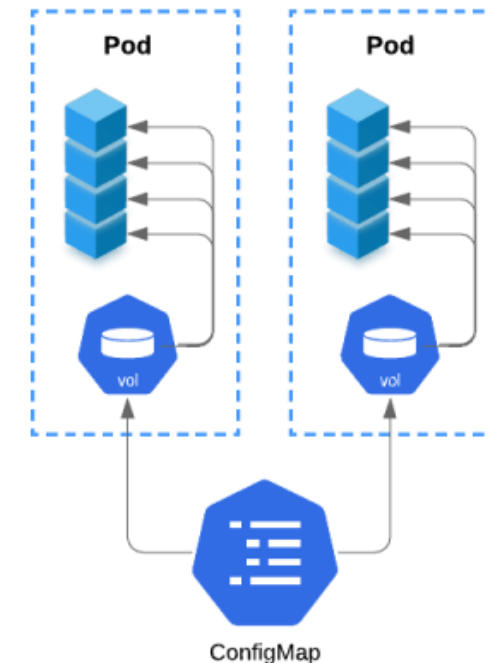
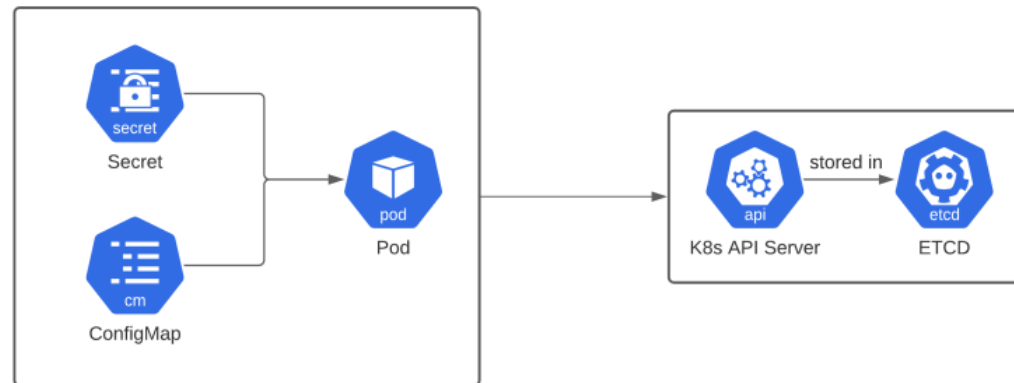
NAME	DESCRIPTION
developpeur	developpeur role for reading access
operationnel	operationnel role for managing application

ArgoCD

Démonstration d'un projet ArgoCD

Nous allons :

- Création de deux utilisateurs correspondant aux rôles
 - Pour ajouter des utilisateurs nous allons devoir configurer le « configMap » d'ArgoCD



Un ConfigMap dans Kubernetes est une ressource qui permet de stocker des données de configuration non-confidentielles sous forme de paires clé-valeur.

ArgoCD

Démonstration d'un projet ArgoCD

Nous allons :

- Ouvrir voter Openlens, se rendre dans l'onglet confiMap dans la partie

Config Maps		8 items
<input type="checkbox"/> Name	Namespace	
<input type="checkbox"/> argocd-cm	argocd	
<input type="checkbox"/> argocd-cmd-params-cm	argocd	
<input type="checkbox"/> argocd-gpg-keys-cm	argocd	
<input type="checkbox"/> argocd-notifications-cm	argocd	
<input type="checkbox"/> argocd-rbac-cm	argocd	
<input type="checkbox"/> argocd-ssh-known-hosts-cm	argocd	
<input type="checkbox"/> argocd-tls-certs-cm	argocd	
<input type="checkbox"/> kube-root-ca.crt	argocd	

ArgoCD

Démonstration d'un projet ArgoCD

Nous allons :

- Modifier le configMap en ajoutant ceci à la fin du fichier

```
data:
  accounts.developpeur: login
  accounts.developpeur.enabled: "true"
  accounts.operationnel: apiKey,login
  accounts.operationnel.enabled: "true"
```

Nous créons nos deux utilisateurs en spécifiant leurs attributions ! Vous devriez les retrouver dans :

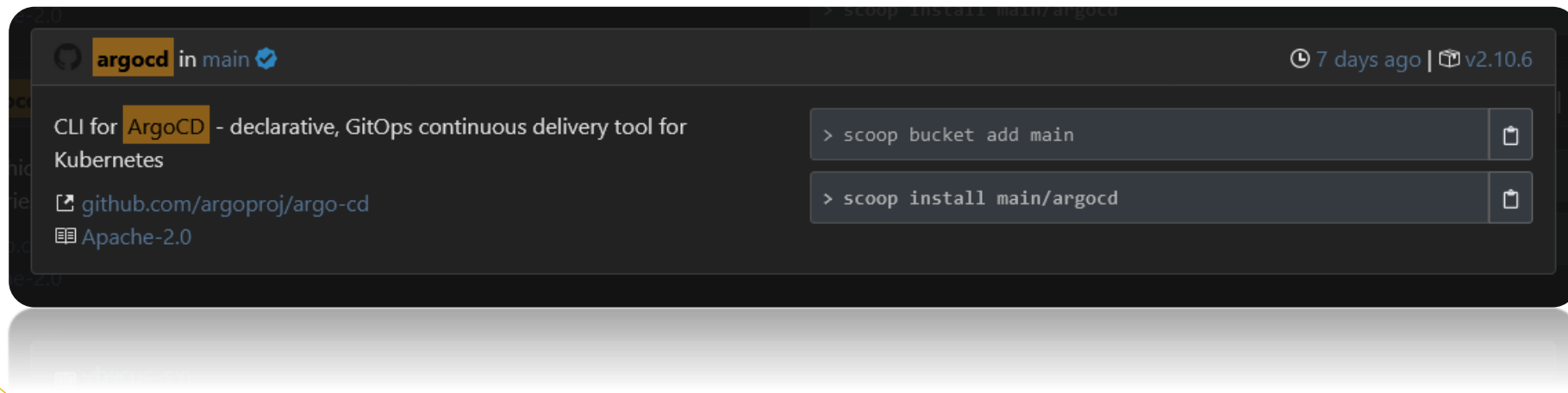
- Settings > Accounts

ArgoCD

ArgoCLI

Afin de procéder aux modifications plus facilement nous allons utiliser le CLI mise à disposition par ArgoCD.

Pour l'installer :



ArgoCD-CLI est un outil en ligne de commande qui permet de gérer ArgoCD

ArgoCD

ArgoCLI

Il faudra vous identifier à travers le portforwarding que nous avons réalisé pour accéder au serveur ArgoCD

A dark-themed terminal window with three colored window control buttons (red, yellow, green) in the top left corner. The text "argocd login localhost:<Port>" is displayed in a light gray monospace font.

```
argocd login localhost:<Port>
```

Une fois connecté taper la commande : `argocd -h` afin d'en apprendre plus sur les commandes disponibles et vérifier que tout fonctionne correctement.

ArgoCD

Démonstration d'un projet ArgoCD

Nous allons maintenant modifier les mots de passes de nos deux utilisateurs afin de se connecter avec eux.

```
→ argocd account update-password --account developpeur  
*** Enter password of currently logged in user (admin):  
*** Enter new password for user developpeur:  
*** Confirm new password for user developpeur:  
Password updated
```

Vous pouvez maintenant vous connecter avec vos utilisateurs. Attention nous avons créé nos utilisateurs mais ceux-ci n'ont pas encore leur rôle attribué.

ArgoCD

ArgoCLI

Nous allons maintenant modifier les comptes utilisateurs afin de leur attribuer leur rôle.

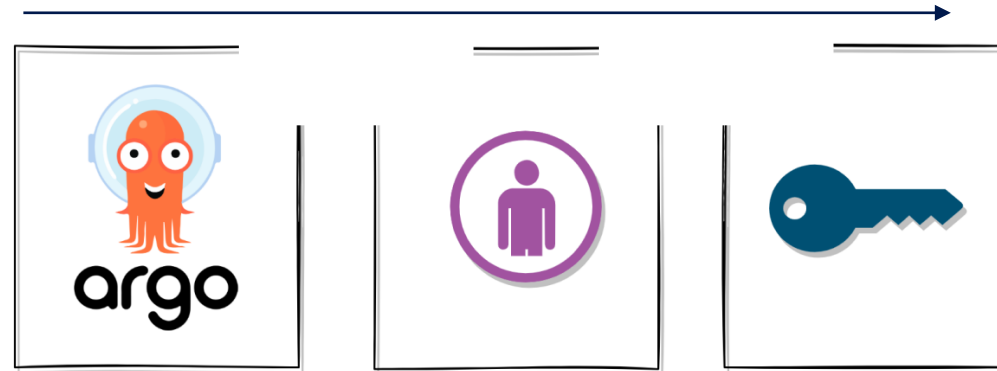


```
argocd proj role add-group pipelinecd developpeur developpeur  
argocd proj role add-group pipelinecd operationnel operationnel
```

ArgoCD

ArgoCLI

Nous avons créé notre premier projet ArgoCD et nous l'avons configuré pour deux utilisateurs.



ArgoCD

ArgoCLI



Manipulation:

- Supprimer vos applications précédemment déployées
- Reconfigurer vos applications dans un projet présentant une division net des rôles et autorisations

ArgoCD

Bonnes pratiques

Séparation des dépôts de configuration et de code source

Il est fortement recommandé d'utiliser un dépôt Git séparé pour conserver vos manifestes Kubernetes, en gardant la configuration distincte du code source de votre application, pour les raisons suivantes :

- **Séparation claire du code de l'application et de sa configuration.**
- **Journal d'audit plus clair.**
- **Votre application peut être composée de services construits à partir de plusieurs dépôts Git, mais déployée comme une unité unique**
- **Séparation des accès.**
- **Si vous automatisez votre pipeline CI, pousser des modifications de manifestes dans le même dépôt Git peut déclencher une boucle infinie de travaux de construction et de déclencheurs de commit Git.**

ArgoCD

Bonnes pratiques

Laisser de la place pour l'impératif

Il peut être souhaitable de laisser de la place à un certain niveau d'impérativité ou d'automatisation, et de ne pas tout définir dans vos manifestes Git. Par exemple, si vous souhaitez que le nombre de réplicas de votre déploiement soit géré par un Autoscaler Horizontal de Pod, alors vous ne voudriez pas suivre les réplicas dans Git.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  # ne pas inclure les réplicas dans les manifestes si vous
  # voulez que les réplicas soient contrôlés par HPA
  # replicas: 1
  template:
    spec:
      containers:
      - image: nginx:1.7.9
        name: nginx
        ports:
        - containerPort: 80
```

ArgoCD


Bonnes pratiques

Assurer que les manifestes aux révisions Git sont réellement immuables

Lors de l'utilisation d'outils de templating tels que Helm ou Kustomize, il est possible que les manifestes changent de sens d'un jour à l'autre.

Cela est généralement causé par des modifications apportées à un dépôt Helm amont ou à une base Kustomize.

```
resources:  
- github.com/argoproj/argo-cd//manifests/cluster-install
```



```
bases:  
- github.com/argoproj/argo-cd//manifests/cluster-install?ref=v0.11.1
```

ArgoCD

Argo Autopilot

Argo Autopilot est une extension de Argo CD, conçue pour simplifier et automatiser le déploiement initial et la gestion de Argo CD lui-même, ainsi que pour faciliter la configuration de nouveaux projets et applications sous Argo CD.



ArgoCD

Argo Autopilot

Argo CD Autopilot fait gagner du temps aux opérateurs en :

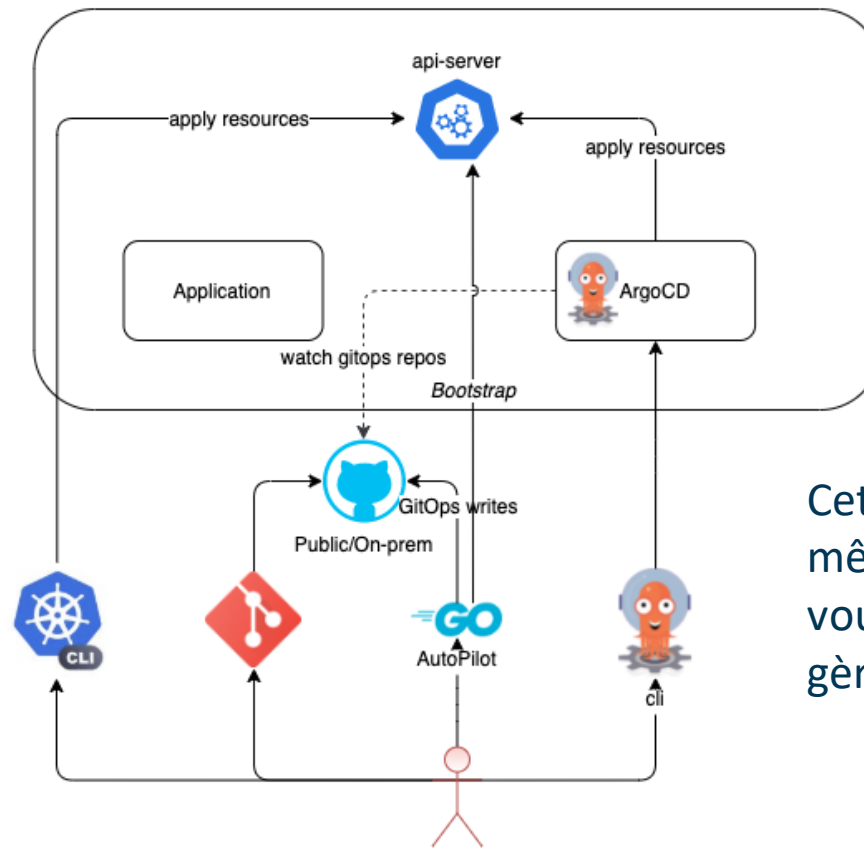
- Installant et gérant l'application Argo CD en utilisant GitOps.
 - Fournissant une structure claire pour l'ajout et la mise à jour des applications, le tout à partir de Git.
 - Créant un modèle simple pour mettre à jour les applications et promouvoir ces changements à travers différents environnements.
-
- Permettant une meilleure récupération après sinistre en étant capable d'initialiser de nouveaux clusters avec toutes les applications précédemment installées.
 - Gérant les secrets pour Argo CD afin de les empêcher de se retrouver en texte clair dans Git. (À venir prochainement)

ArgoCD

Argo Autopilot

Comment ça fonctionne

La commande de démarrage automatique (autopilot bootstrap) déploiera un manifeste Argo CD sur un cluster Kubernetes cible et engagera un manifeste d'application Argo CD sous un répertoire spécifique dans votre dépôt GitOps.



Cette Application gèrera l'installation d'Argo CD elle-même - donc après avoir exécuté cette commande, vous disposerez d'un déploiement Argo CD qui se gère lui-même via GitOps.

ArgoCD

Argo Autopilot

Installation :

```
scoop install main/argocd-autopilot
```

```
$env:GIT_REPO="https://votrerepos.com"  
$env:GIT_TOKEN="votre_token"
```

```
argocd-autopilot repo bootstrap
```

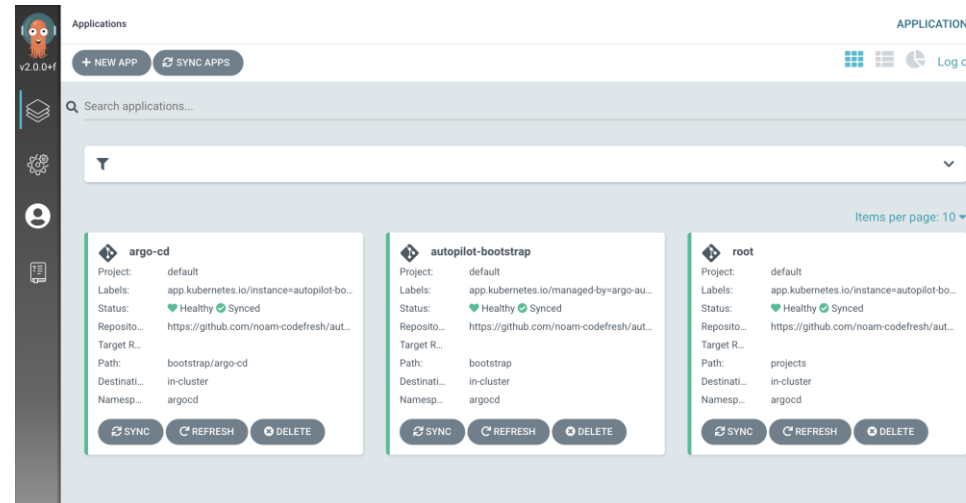
Une fois déployé : Vous trouverez le mot de passe du compte administrateur dans le terminal ainsi que votre référentiel de code contenant les plusieurs dossiers !

ArgoCD

Argo Autopilot

Votre installation initiale d'Argo CD devrait inclure les applications suivantes :

- **autopilot-bootstrap** - Fait référence au répertoire de démarrage dans le dépôt GitOps et gère les deux autres applications.
- **argo-cd** - Fait référence au dossier bootstrap/argo-cd et gère le déploiement d'Argo CD lui-même (y compris l'ensemble d'applications Argo CD).
- **root** - Fait référence au répertoire des projets dans le dépôt. Ce dossier contient uniquement un fichier DUMMY vide après la commande de démarrage, donc aucun projet ne sera créé.



ArgoCD

Argo Autopilot

Nous pouvons faciliter dorénavant la création de projet et d'application directement dans notre terminal :



Etape 1 :

```
- argocd-autopilot project create testing
```

Etape 2 :

```
- argocd-autopilot app create hello-world --app github.com/argoproj-labs/argocd-autopilot/examples/demo-app/ -p testing --wait-timeout 2m
```

ArgoCD

Argo Autopilot

Nous pouvons observer :

- Les changements directement sur notre dépôt.
- Le projet et l'application sont ajoutés et créés automatiquement dans notre ArgoCD.
- Tous les fichiers de configuration peuvent être trouvés et modifiés via notre référentiel.

