ТЕОРЕТИЧНА ЧАСТИНА

ArgoCD - це відкрите програмне забезпечення для автоматизації впровадження та управління додатками в середовищі Kubernetes. Це інструмент для CI/CD, який дозволяє розробникам та операторам спрощувати і контролювати процеси розгортання та управління конфігураціями у кластерах Kubernetes. У цій лабораторній роботі ми розглянемо важливі аспекти Argocd, його функціональність, архітектуру, основні переваги та способи використання.

ArgoCD - це інструмент управління розгортанням додатків в середовищі Kubernetes. Він розроблений з урахуванням принципів "GitOps", де стан системи описується у конфігураційних файлах і зберігається в системі контролю версій, такій як Git. Argocd забезпечує автоматичне розгортання додатків відповідно до збереженої конфігурації, і, якщо конфігурація змінюється, автоматично оновлює стан ресурсів Kubernetes.

GitOps: Основні принципи

GitOps - це методологія управління конфігурацією та розгортанням, де основним джерелом правди є Git-репозиторій. Основні принципи GitOps включають:

1. Декларативна конфігурація: Вся конфігурація і описи ресурсів зберігаються у декларативному форматі. Описувані ресурси Kubernetes також зберігаються у цьому форматі.
2. Інструменти CI/CD: Для автоматизації розгортання та оновлення конфігурації використовуються інструменти CI/CD, які слідкують за змінами у Git-репозиторіях та автоматично застосовують їх до Kubernetes.
3. Система контролю версій Git: Git-репозиторій використовується як джерело правди для стану системи та конфігурації. Вся історія змін легко доступна і ревізії репозиторія можуть бути використані для відкату змін.
4. Інверсія керування: Замість того, щоб розгортати додатки напряму, GitOps передбачає, що система сама реагує на зміни у репозиторіях та відповідно до них оновлює стан.

Основні функції Argocd

ArgoCD надає ряд функцій, які роблять його потужним інструментом для GitOps в Kubernetes:

1. ArgoCD підтримує Helm-чарти, що дозволяє розгортати та керувати додатками, які описані у вигляді Helm-пакетів.
2. Синхронізація з Git: Він постійно слідкує за змінами у Git-репозиторіях та автоматично синхронізує стан ресурсів з конфігурацією.
3. Спрощений інтерфейс: Argocd надає веб-інтерфейс та командний рядок для управління ресурсами та перевірки стану додатків.
4. Декларативна конфігурація за замовчуванням: Всі налаштування для додатків та ресурсів вказуються у конфігураційних файлах, що полегшує управління.
5. Спрощений рольовий доступ: Argocd підтримує ролі та дозволи для різних користувачів та команд.
6. Моніторинг та аудит: Записи про операції та події ведуться для аудиту та відлагодження.
7. Підтримка множинних середовищ: Argocd може управляти додатками в різних кластерах Kubernetes або середовищах.
8. Розширюваність та інтеграція: Існує можливість розширення функціональності Argocd за допомогою плагінів та інтеграції з іншими інструментами.

Архітектура Argocd

Архітектура Argocd складається з кількох ключових компонентів, які спільно працюють для забезпечення автоматизованого розгортання та управління додатками в Kubernetes.

Компоненти Argocd

1. ArgoCD API сервер: Це головний компонент Argocd, який надає API для взаємодії з системою. Він служить для управління додатками та ресурсами Kubernetes, включаючи їх розгортання та оновлення.
2. ArgoCD UI (веб-інтерфейс): Веб-інтерфейс Argocd надає графічний інтерфейс для управління та відслідковуванням стану додатків. Це спрощує візуалізацію конфігурацій та стану.

3. Сховище конфігурації (Git-репозиторії): Git-репозиторії є джерелом конфігурації для Argocd. Вони містять декларативні описи додатків та ресурсів Kubernetes.
4. Служби репозиторіїв (Repo Servers): Repo Servers відповідають за взаємодію з Git-репозиторіями, витягуючи конфігурацію та сповіщаючи Argocd про зміни.
5. Декларативні додатки та ресурси: Додатки та ресурси Kubernetes описуються у декларативному форматі та зберігаються у Git-репозиторіях.
6. Служба синхронізації (Sync Service): Служба синхронізації відповідає за автоматичне розгортання та оновлення ресурсів Kubernetes на основі конфігурації Git-репозиторіїв.

Взаємодія компонентів

Під час роботи Argocd відбувається наступна послідовність подій:

1. Розробник оновлює конфігурацію додатка та зберігає її в Git-репозиторії.
2. Repo Server витягує зміни з Git-репозиторію та інформує Argocd про зміни.
3. Argocd API сервер аналізує зміни та визначає, які ресурси Kubernetes потребують оновлення.
4. Служба синхронізації розгортає та оновлює ресурси відповідно до зміненої конфігурації.
5. Argocd API сервер відображає стан додатків та ресурсів у веб-інтерфейсі.

Helm

Helm - це інструмент для керування пакунками та шаблонами для контейнеризованих додатків. Він створений з метою полегшення розгортання, оновлення та керування додатками, які працюють на Kubernetes - одній із найпопулярніших платформ для оркестрації контейнерів.

Перед тим як детальніше розглянути Helm, розглянемо деякі основні поняття, які використовуються в цьому контексті:

- Чарт (Chart): Це основна концепція Helm. Чарт - це пакунок, який містить всі необхідні ресурси та конфігурацію для розгортання контейнеризованого додатку на Kubernetes. Він включає в себе шаблони для ресурсів Kubernetes, значення конфігурації та інші компоненти.
- Helm Release: Реліз в Helm - це конкретна інстанція чарта, яка розгортана на Kubernetes кластері. Кожен реліз може мати свої унікальні налаштування та значення конфігурації.

Компоненти Helm

Helm складається з кількох основних компонентів, кожен з яких відповідає за певний аспект керування контейнеризованими додатками. Основні компоненти Helm включають в себе:

- Helm CLI (Command Line Interface): Helm CLI - це інтерфейс командного рядка, який дозволяє розробникам та адміністраторам взаємодіяти з Helm та керувати релізами чартів. Цей інтерфейс надає можливість створювати нові релізи, оновлювати існуючі та видаляти їх.
- Tiller: Tiller був компонентом Helm, який використовувався для керування релізами на Kubernetes. Проте на більш пізніх версіях Helm (починаючи з Helm 3) Tiller був припинений і більше не використовується. Це стало можливим завдяки усуненню проблем безпеки та покращенню архітектури Helm.
- Чарти (Charts): Ці пакунки містять всю необхідну інформацію для розгортання контейнеризованого додатку. Чарти включають в себе файли шаблонів, значення конфігурації, метадані та інші ресурси, які можуть бути розгорнуті на Kubernetes кластері.
- Значення конфігурації (Values): Це файли або параметри, які можна налаштувати під час розгортання чарту. Значення конфігурації дозволяють змінювати параметри додатків, наприклад, розмір ресурсів, порти, змінні середовища та інше.
- Залежності (Dependencies): Чарти можуть мати залежності від інших чартів. Це дозволяє підключати та використовувати готові рішення, які можна інтегрувати в свій власний чарт.

Робочий процес з Helm

Робочий процес Helm може бути описаний наступним чином:

1. Створення Чарта: Розробник створює чарт, який містить усі необхідні файли та шаблони для додатку. Це включає в себе опис ресурсів Kubernetes, значення конфігурації та інші компоненти.
2. Конфігурація Значень (Values Configuration): Розробник налаштовує значення конфігурації, які можуть бути використані під час розгортання чарта. Це дозволяє змінювати параметри додатку для кожного релізу.
3. Розгортання Чарта: За допомогою Helm CLI або інших інструментів (наприклад ArgoCD) розробник розгортає чарт на Kubernetes кластері, створюючи новий реліз. Helm автоматично створює ресурси Kubernetes та надає конфігурацію згідно зі значеннями конфігурації.
4. Оновлення та Управління Релізами: Helm дозволяє легко оновлювати та керувати релізами. Розробник може виконати оновлення чарта або змінити значення конфігурації для існуючого релізу.
5. Видалення Релізів: Після закінчення використання релізу, його можна легко видалити за допомогою Helm CLI. Це включає в себе видалення всіх створених ресурсів та очищення кластера.

Переваги Helm

Helm має кілька важливих переваг, які роблять його популярним інструментом для керування контейнеризованими додатками:

- Спрощення розгортання: Helm робить розгортання додатків на Kubernetes швидким і простим завдяки шаблонам та значенням конфігурації.
- Скальованість: Helm дозволяє легко масштабувати додатки та робити оновлення на багатьох серверах.
- Модульність та Залежності: Helm дозволяє розробникам використовувати готові чарти та інтегрувати їх в свої власні проекти, що робить розробку більш швидкою та ефективною.
- Історія та Управління Релізами: Helm зберігає історію релізів, що дозволяє легко відновити попередні версії додатків та здійснювати відкочування змін.

- Спільнота та Екосистема: Helm має активну спільноту розробників і багато готових чартів, які можуть бути використані для різних потреб.

Детільніше про helm можна подивитись в офіційній документації: https://helm.sh/docs/

Встановлення та конфігурація Argocd

Детальні інструкції та вимоги до встановлення ArgoCD описані в документації: https://argo-cd.readthedocs.io/en/stable/

ПРАКТИЧНА ЧАСТИНА

Встановлення ArgoCD та підготовка

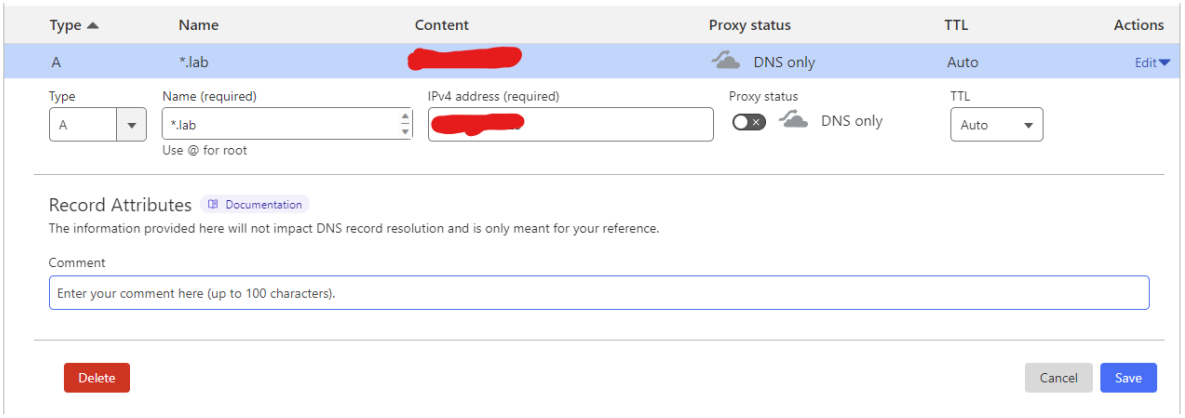
1. Встановіть додаткові інструменти для роботи:
    a. nginx ingress (https://kubernetes.github.io/ingress-nginx/deploy/)
    b. cert-manager (https://cert-manager.io/docs/installation/helm/)

```
$ helm upgrade --install ingress-nginx ingress-nginx --repo
$ https://kubernetes.github.io/ingress-nginx --namespace ingress-nginx
--create-namespace
$ kubectl apply -f
https://github.com/cert-manager/cert-manager/releases/download/v1.13.1/cert-manager.crds.yaml
$ helm install cert-manager jetstack/cert-manager --namespace cert-manager
--create-namespace
```

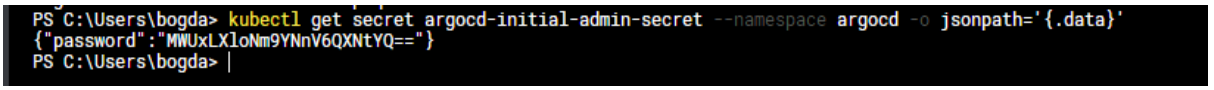
2. Встановіть ArgoCD

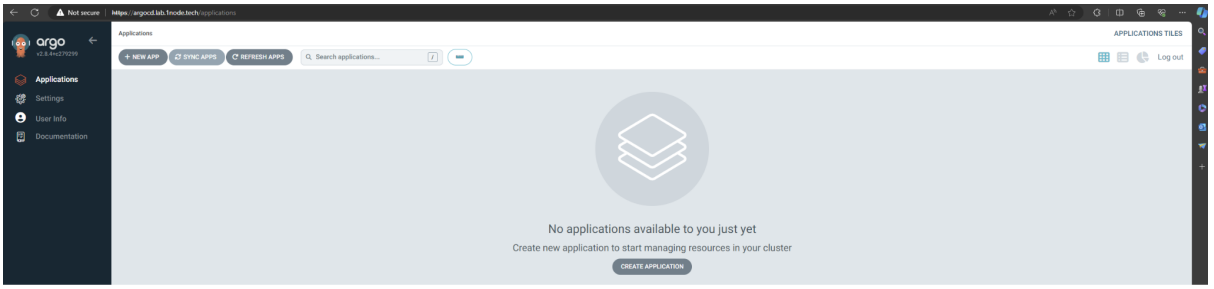
```
$ kubectl create namespace argocd
$ kubectl apply -n argocd -f
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

3. Налаштуйте доменну зону та створіть необхідні DNS записи (A або CNAME) (опціонально).



4. Застосуйте k8s маніфести для випуску сертифікатів letsencrypt та створення ingress ресурсу для argocd. Зверніть увагу, цього не потрібно робити якщо ви не плануєте налаштовувати домени для ваших сервісів. Також перевірте вміст маніфестів, можливо знадобиться оновити їх виходячи з контексту вашої інсталяції.
   $ kubectl apply -f lab_argo/01_installation/le-issuer.yml
   $ kubectl apply -f lab_argo/01_installation/ingress.yml
5. Отримайте пароль для входу в адмін інтерфейс ArgoCD та декодуйте його значення:
   $ kubectl get secret argocd-initial-admin-secret --namespace argocd -o jsonpath='{.data}'
   $ echo "MWUxLXloNm9YNnV6QXNtYQ==" | base64 -d

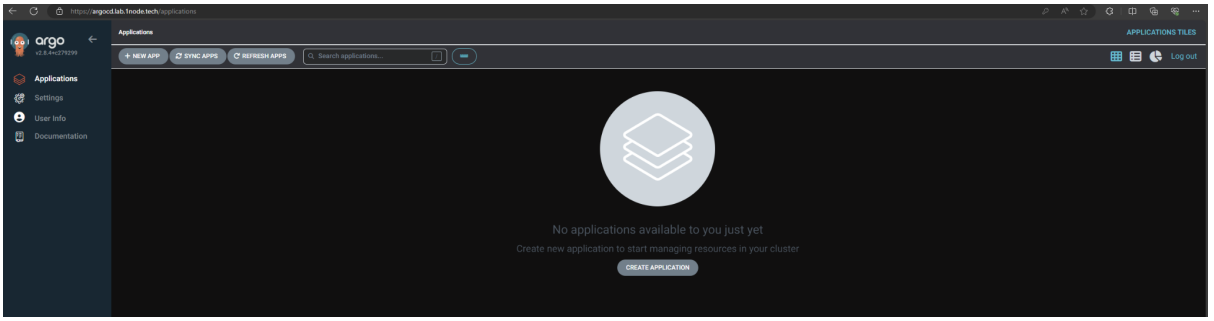


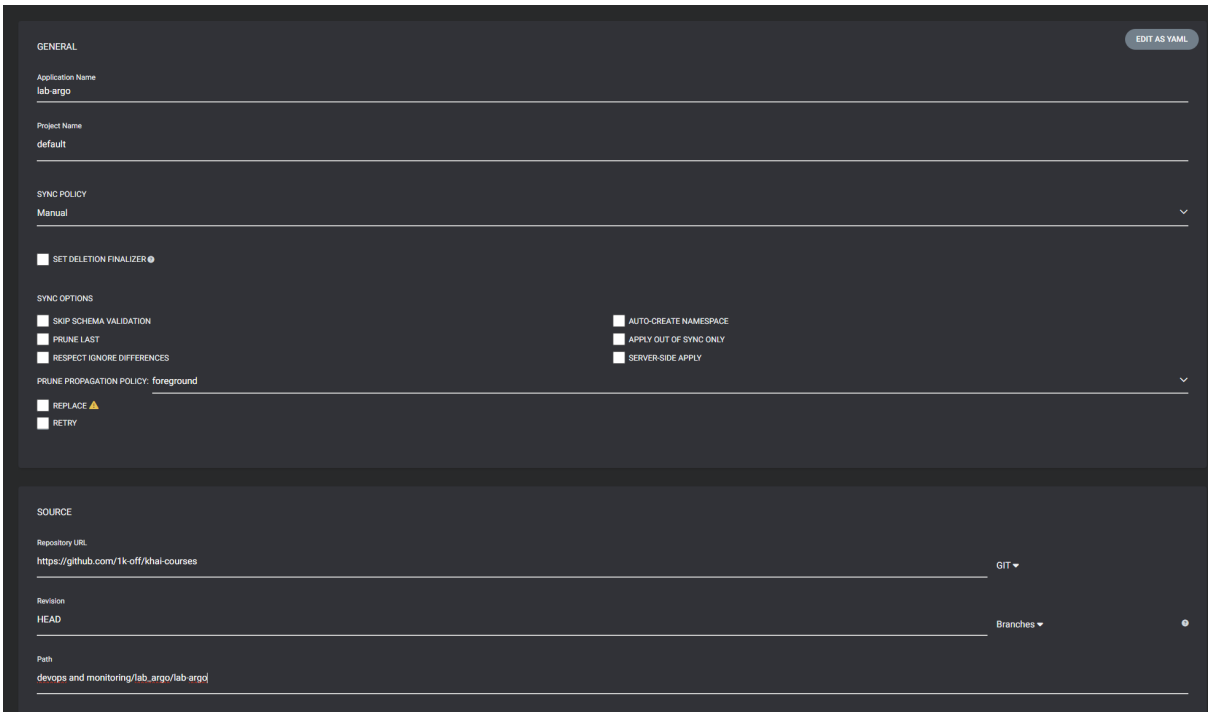


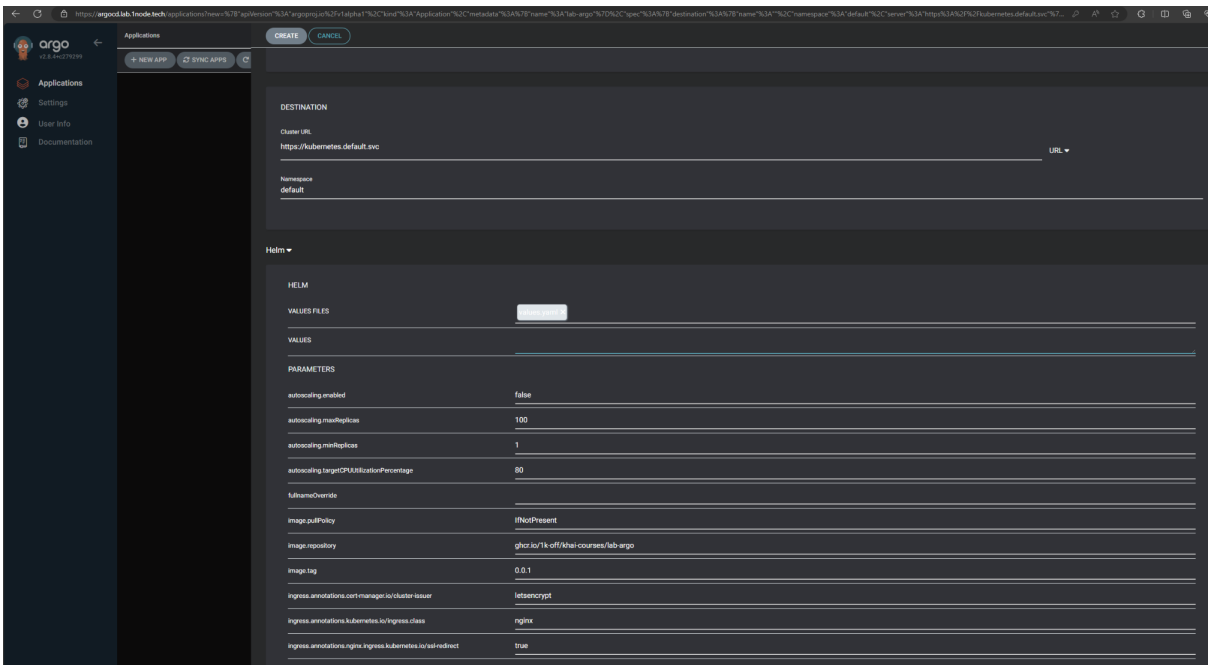
6. Увімкніть темну тему :) (Опціонально, але це +10 до скілу).

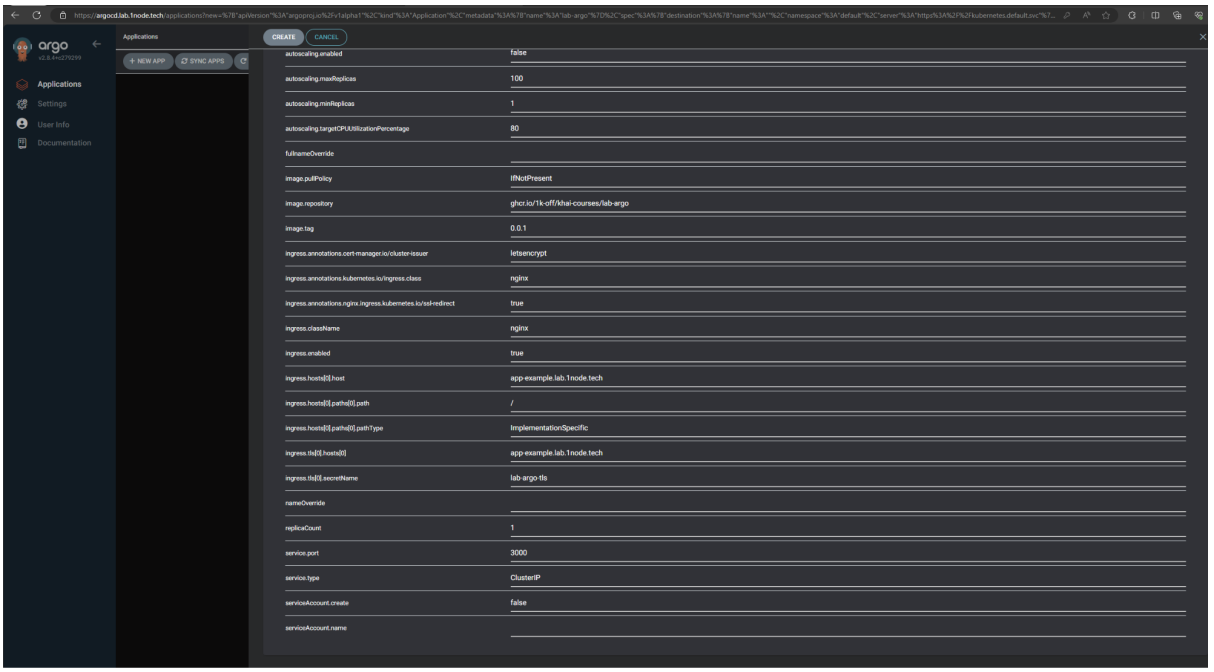
Створення додатку

1. Підготуйте вихідний код вашого застосунку, створіть Dockerfile, створіть helm чарт, зберіть та запуште docker image до docker registry.
2. Створіть додаток в інтерфейсі ArgoCD та додайте необхідну інформацію.

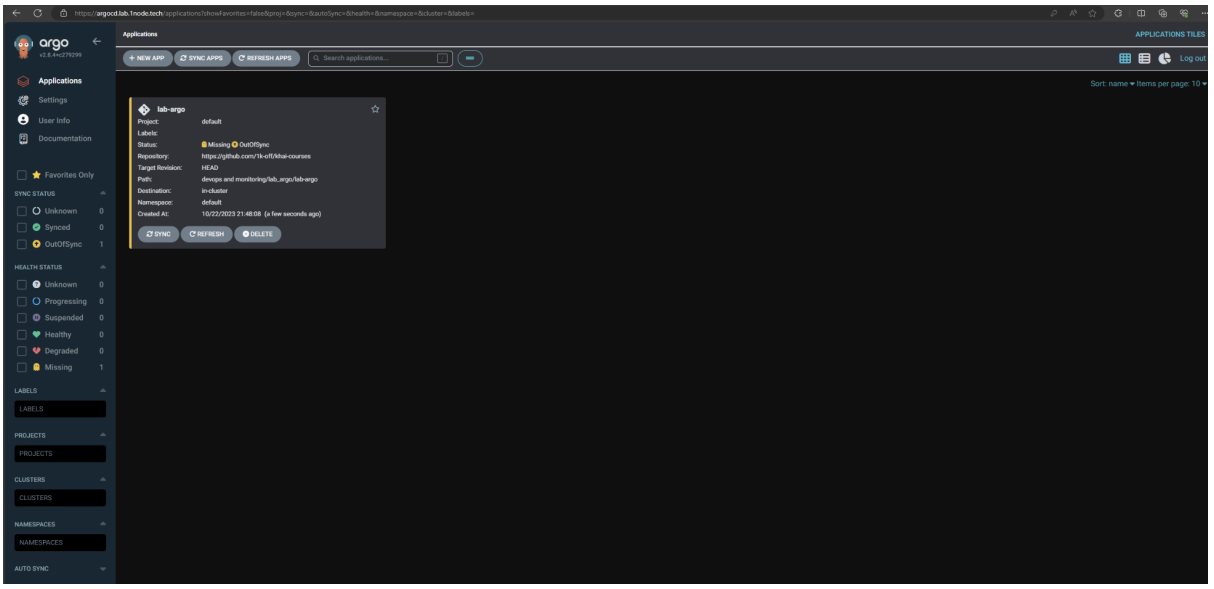

argo
Applications
Settings
User Info
Documentation
Applications
APPLICATIONS TILES
+ NEW APP
SYNC APPS
REFRESH APPS
Search applications...
Log out
No applications available to you just yet
Create new application to start managing resources in your cluster
CREATE APPLICATION

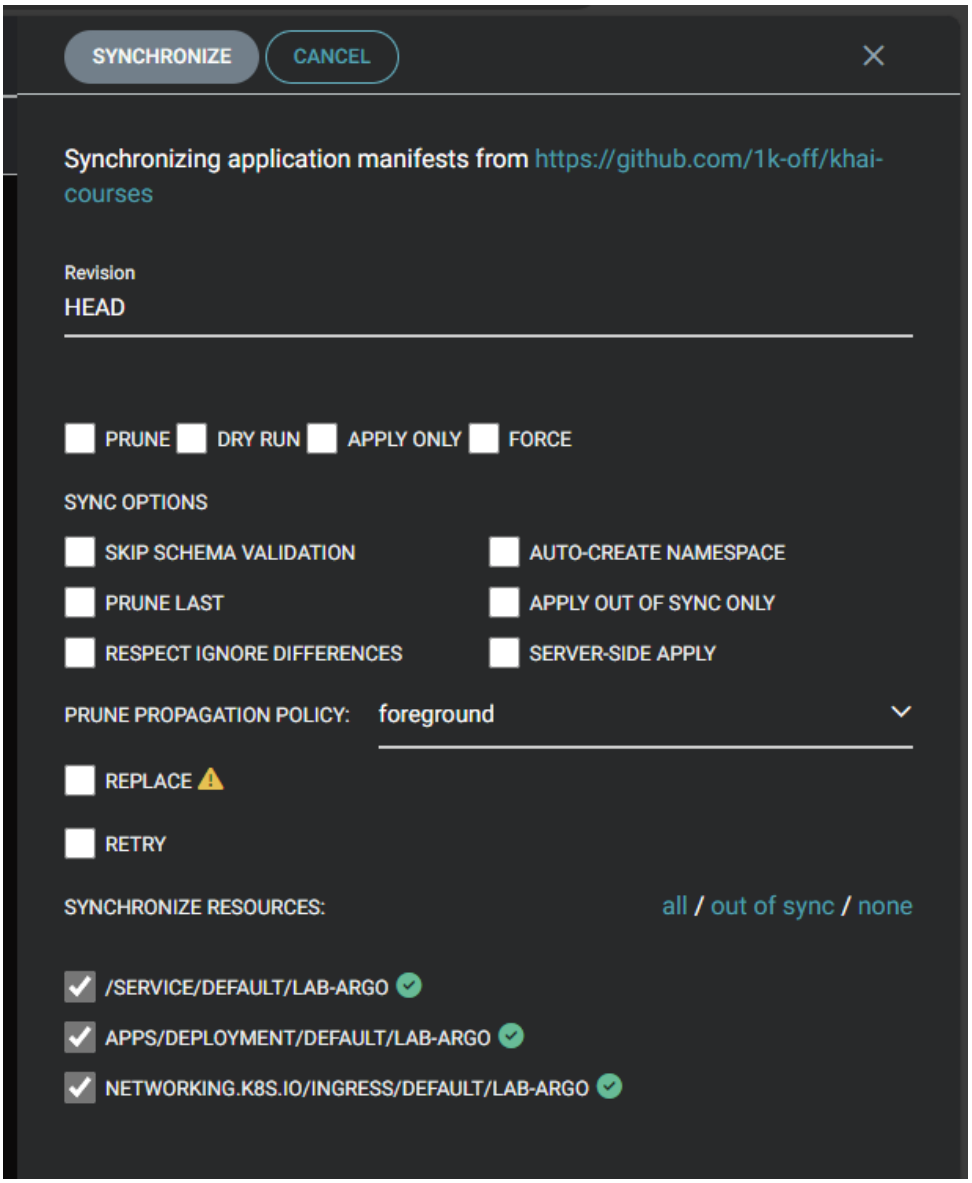

GENERAL
EDIT AS YAML
Application Name
lab-argo
Project Name
default
SYNC POLICY
Manual
SET DELETION FINALIZER
SYNC OPTIONS
SKIP SCHEMA VALIDATION
PRUNE LAST
RESPECT IGNORE DIFFERENCES
AUTO-CREATE NAMESPACE
APPLY OUT OF SYNC ONLY
SERVER-SIDE APPLY
PRUNE PROPAGATION POLICY: foreground
REPLACE
RETRY
SOURCE
Repository URL
https://github.com/1k-off/khai-courses
GIT
Revision
HEAD
Branches
Path
devops and monitoring/lab_argo/lab-argo


argocd.lab.1node.tech/applications?new=%7B"apiVersion"%3A"argoproj.io%2Fv1alpha1"%2C"kind"%3A"Application"%2C"metadata"%3A%7B"name"%3A"lab-argo"%7D%2C"spec"%3A%7B"destination"%3A%7B"name"%3A""%2C"namespace"%3A"default"%2C"server"%3A"https%3A%2F%2Fkubernetes.default.svc"%7...
argo
Applications
+ NEW APP
SYNC APPS
CREATE
CANCEL
Applications
Settings
User Info
Documentation
DESTINATION
Cluster URL
https://kubernetes.default.svc
URL
Namespace
default
Helm
HELM
VALUES FILES
VALUES
PARAMETERS
autoscaling.enabled
false
autoscaling.maxReplicas
100
autoscaling.minReplicas
1
autoscaling.targetCPUUtilizationPercentage
80
fullnameOverride
image.pullPolicy
IfNotPresent
image.repository
ghcr.io/1k-off/khai-courses/lab-argo
image.tag
0.0.1
ingress.annotations.cert-manager.io/cluster-issuer
letsencrypt
ingress.annotations.kubernetes.io/ingress.class
nginx
ingress.annotations.nginx.ingress.kubernetes.io/ssl-redirect
true

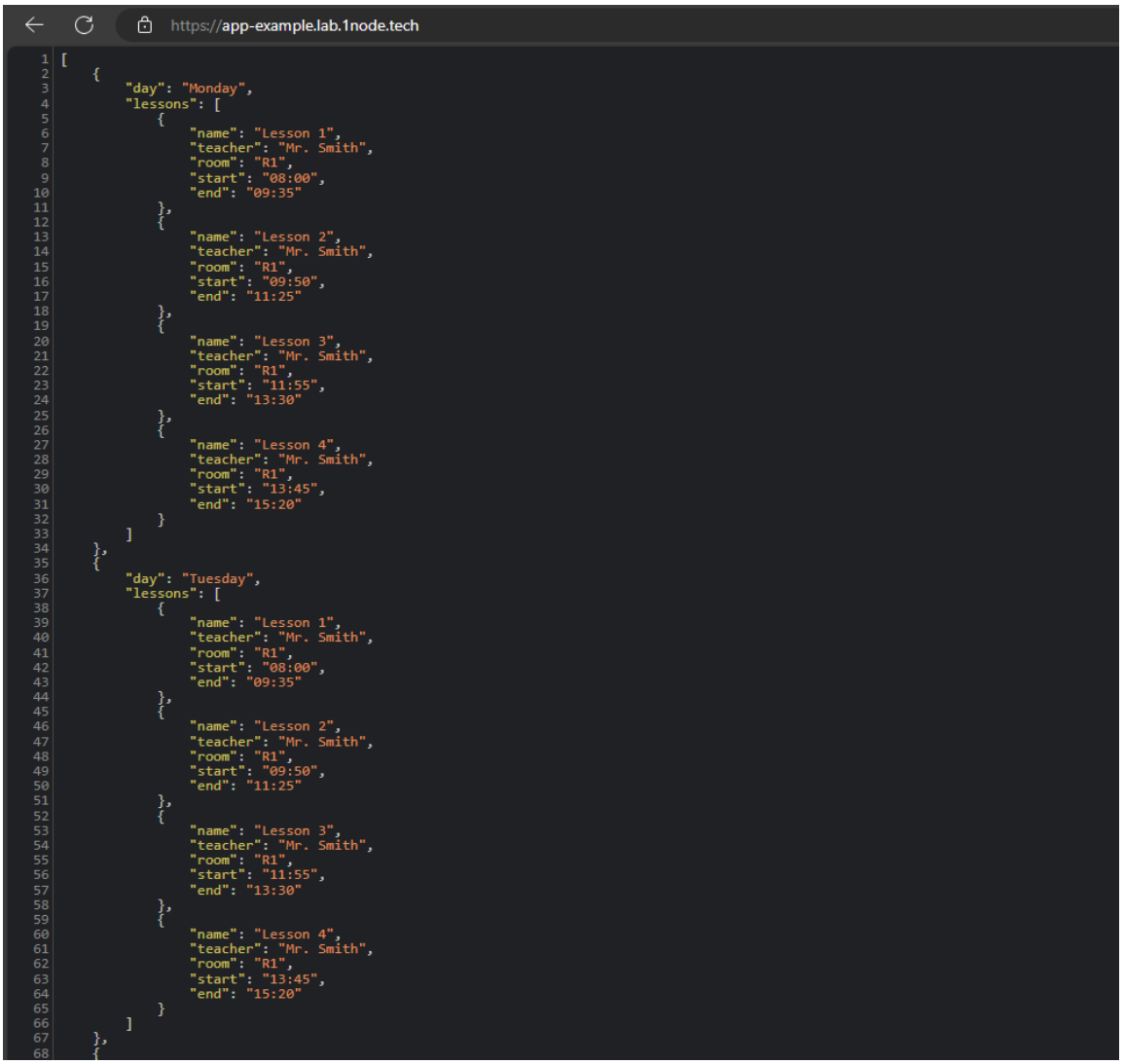

argocd.lab.1node.tech/applications?new=%7B"apiVersion"%3A"argoproj.io%2Fv1alpha1"%2C"kind"%3A"Application"%2C"metadata"%3A%7B"name"%3A"lab-argo"%7D%2C"spec"%3A%7B"destination"%3A%7B"name"%3A""%2C"namespace"%3A"default"%2C"server"%3A"https%3A%2F%2Fkubernetes.default.svc"%7...
argo
Applications
+ NEW APP
SYNC APPS
CREATE
CANCEL
Applications
Settings
User Info
Documentation
autoscaling.enabled
false
autoscaling.maxReplicas
100
autoscaling.minReplicas
1
autoscaling.targetCPUUtilizationPercentage
80
fullnameOverride
image.pullPolicy
IfNotPresent
image.repository
ghcr.io/1k-off/khai-courses/lab-argo
image.tag
0.0.1
ingress.annotations.cert-manager.io/cluster-issuer
letsencrypt
ingress.annotations.kubernetes.io/ingress.class
nginx
ingress.annotations.nginx.ingress.kubernetes.io/ssl-redirect
true
ingress.className
nginx
ingress.enabled
true
ingress.hosts[0].host
app-example.lab.1node.tech
ingress.hosts[0].paths[0].path
/
ingress.hosts[0].paths[0].pathType
ImplementationSpecific
ingress.tls[0].hosts[0]
app-example.lab.1node.tech
ingress.tls[0].secretName
lab-argo-tls
nameOverride
replicaCount
1
service.port
3000
service.type
ClusterIP
serviceAccount.create
false
serviceAccount.name

3. Синхронізуйте стан ArgoCD зі станом репозиторія.



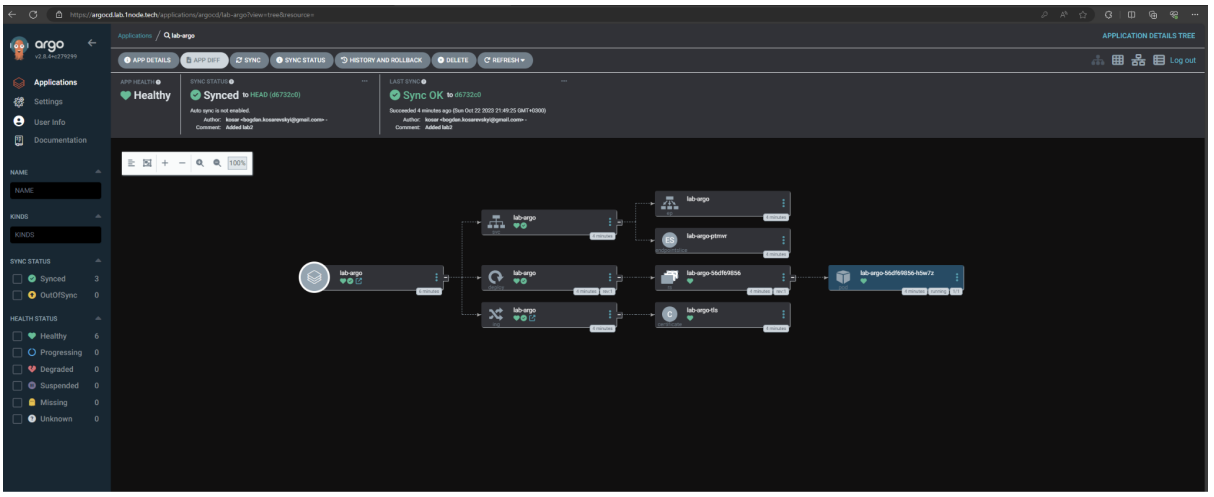


4. Перевірте стан k8s та вашого застосунку. Переконайтеся що додаток запустився коректно, що створились усі необхідні ресурси (ingress, ssl скртифікат якщо це потрібно). Зайдіть на домен що ви зареєстрували, або на сторінку сервісу в kubernetes за допомогою kubectl port forward.



5. Перевірте стан додатку в ArgoCD. Зробіть декілька кліків по ресурсам додатку, поясніть що це за ресурси та яку функцію вони

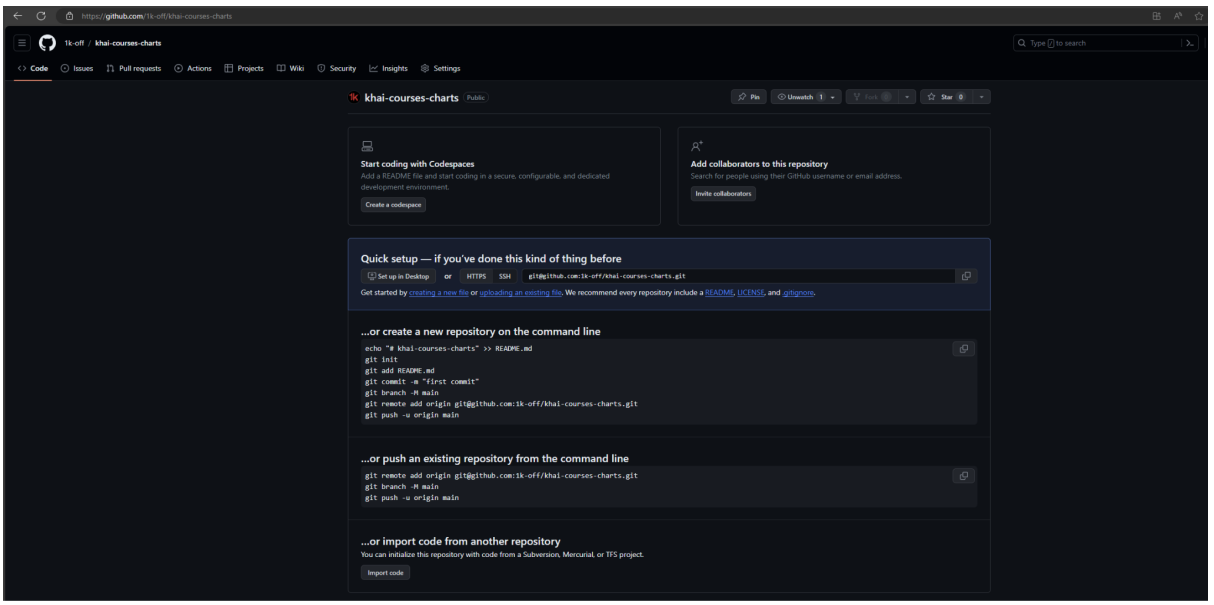
виконують.



Автоматизація оновлень додатку

Прочитайте документацію про Helm Chart releaser. https://github.com/helm/chart-releaser-action.

1. Створіть окремий репозиторій для ваших helm чартів.
2. Встановіть версію image tag як latest для вашого чарту. Вона не має використовуватись для запуску, але буде виконувати роль шаблону в даному випадку. Перенесіть ваш чарт в його репозиторій та видаліть його з репозиторію з кодом.



3. За допомогою GitHub Actions створіть CI пайплайн для вашого застосунку. Переконайтеся що пайплайн може коректно версіонувати ваші docker images та оновлювати ваш чарт.

4. Оновіть додаток в ArgoCD та встановіть посилання на репозиторій з чартом. Увімкніть автоматичну синхронізацію.
5. Оновіть код вашого застосунку. Переконайтесь що CI пайплайн відпрацював коректно, а саме: зібрав docker image, проверсіонував його, запушив до docker registry, оновив значення в helm чарті.
6. Переконайтесь що після оновлення значень в helm чарті ArgoCD оновив версію вашого застосунку (може знадобитись до 5 хвилин). Якщо цього не сталося - спробуйте оновити застосунок мануально, розібратись чому не спрацювало автоматичне оновлення та повторити п. 5-6.