

# Déploiement d'une application cloud native à l'Université de Lorraine

**Journées déploiement  
d'application**

> 12 et 13 juin 2024

> Nancy

# Application cloud native

Conçue pour être évolutive et hautement disponible

- > Basée sur des conteneurs versionnés, sécurisés, avec healthcheck permettant de connaître leur statut
- > Orientée microservices
- > Gérée de manière dynamique pour optimiser l'utilisation des ressources





# Pléiades Relations - ULEP

« Dynamiser l'apprentissage des langues »

**Relations**, l'un des 5 programmes du projet Pléiades

Projet Lorrain d'Environnement numérique pour des Apprentissages Durables

<https://www.univ-lorraine.fr/pleiades/relations/>

**ULEP**, une application pour faciliter les tandems linguistiques

University Language Exchange Programme

- > Connecter les étudiants et personnels de l'université avec ceux des établissements partenaires à travers le monde
- > Les accompagner pédagogiquement
- > Tranche 1 en production restreinte (inscriptions, administration, algorithme d'appariement...)
- > Tranche 2 en développement (chat, appels vidéo, journal d'apprentissage...)

# Acteurs du projet



## UFR LANSAD

Langues pour spécialistes  
d'autres disciplines

Responsables du programme :

Carine Martin

Nicolas Molle



## Direction du Numérique

Sous-direction SIED

Chef de projet : Philippe Brouard



## The Tribe

Agence de développement  
web & mobile



## Ability

Agence de design

# Briques applicatives

## ULEP

- > **Applications web/mobile** Ionic React
- > **Back office** React-admin
- > **APIs** NestJS
- > **Base de données** PostgreSQL

## AUTRES BRIQUES

- > **Keycloak** authentification
- > **Weblate** traductions
- > **MinIO** stockage des médias
- > **Firebase** notifications
- > **Jitsi** visioconférence
- > **SocketIO** chat
- > **GlitchTip** centralisation des erreurs
- > **Connecteur SI** données des étudiants et des personnels de l'université



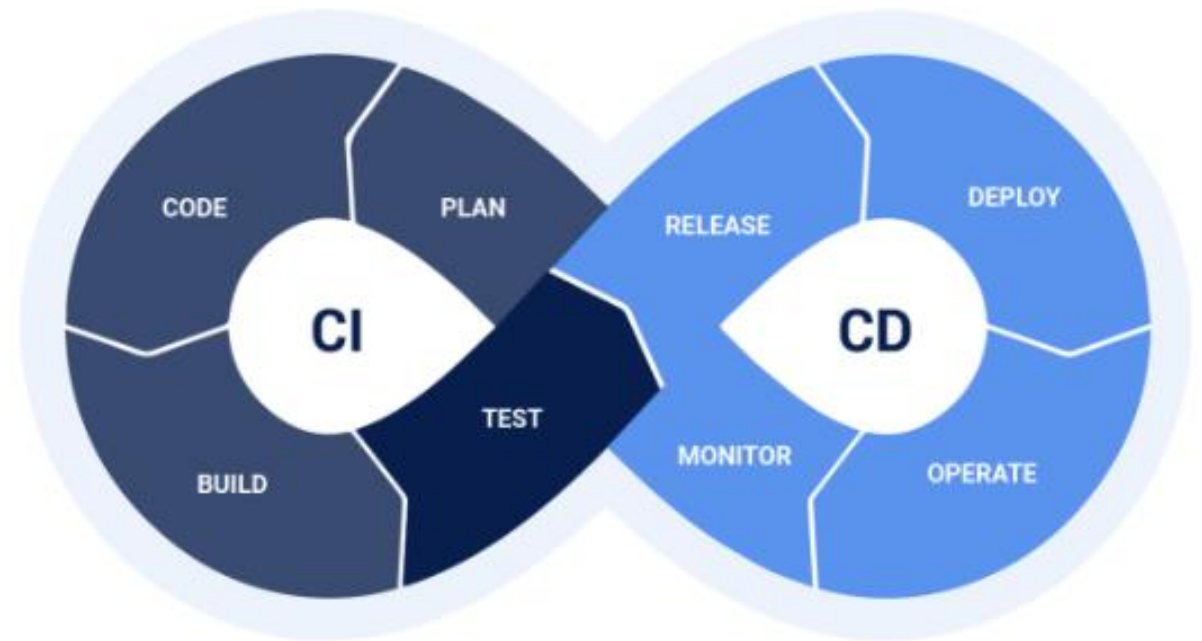
# Déploiement

Intégration continue

Kubernetes

Chart Helm

Déploiement continu



# Intégration continue

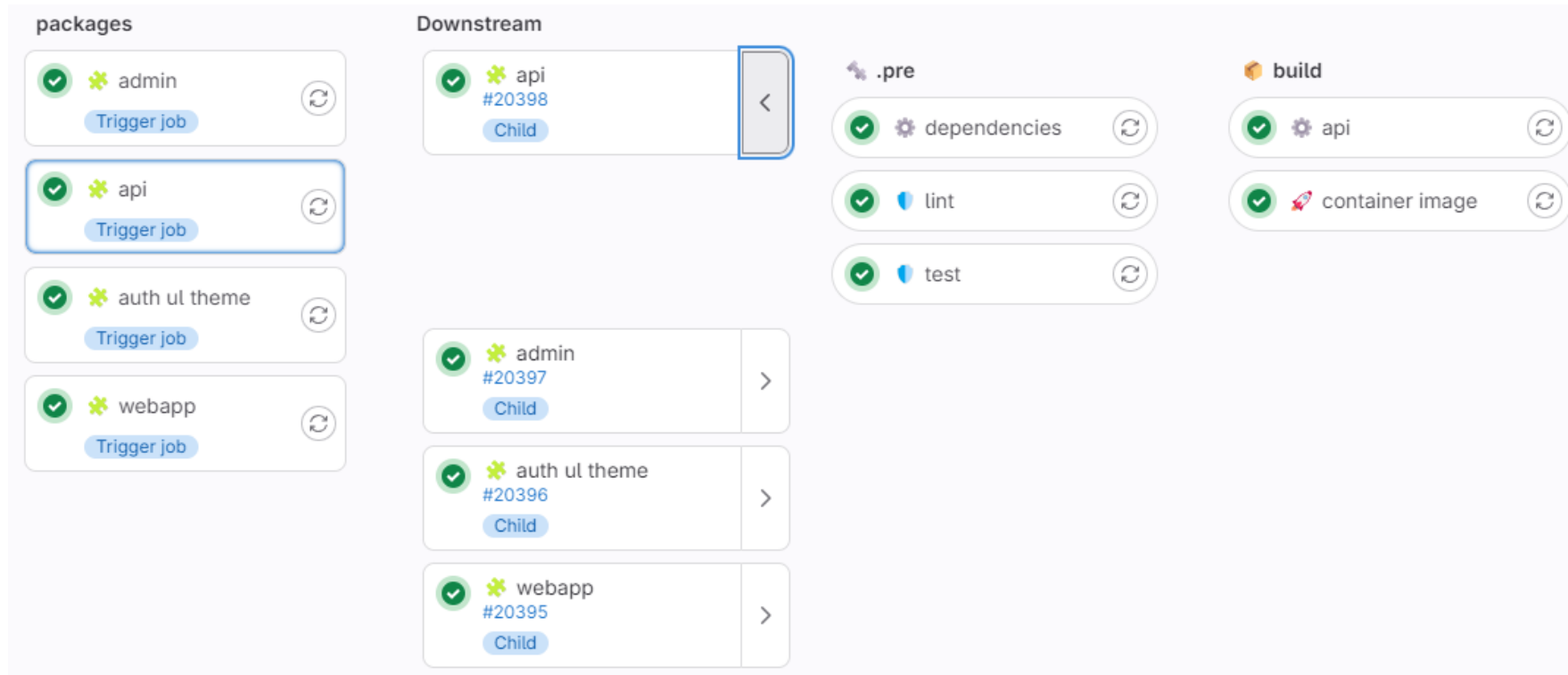
## GitLab CI

- > **.gitlab-ci.yml** configuration de la pipeline (stages, jobs) exécutée automatiquement à chaque changement dans le dépôt
- > **Dockerfile** fichier texte contenant toutes les commandes, dans l'ordre, nécessaires à la construction d'une image
- > **Kaniko** build des images à partir des Dockerfiles et dépôt sur la registry
- > **Harbor** registry d'images et de charts Helm



# Intégration continue

## Pipeline



# Intégration continue

## Focus sur Kaniko

### > Adapté à la CI

pas de dépendance au daemon Docker  
pas besoin de privilèges élevés

### > Distribué sous forme d'image

### > Image debug recommandée

possède un shell, et un shell est nécessaire pour qu'une image puisse être utilisée avec GitLab CI

[https://docs.gitlab.com/ee/ci/docker/using\\_kaniko.html](https://docs.gitlab.com/ee/ci/docker/using_kaniko.html)

```
# Build de l'image
container image:
stage: build
needs:
- job: lint
  optional: true
- job: test
  optional: true
- job: admin
  optional: true
- job: api
  optional: true
image:
name: gcr.io/kaniko-project/executor:debug
entrypoint: [""]
before_script: # Informations de connexion au registry Harbor dans un fichier de conf Kaniko
- mkdir -p /kaniko/.docker
- echo "{\"auths\":{\"${CI_REGISTRY}\":{\"auth\":\"$(printf \"%s:%s\" \"${CI_REGISTRY_USER}\" \"${CI_REGISTRY_PASSWORD}\" | base64 | tr -d '\\n')\"}}}" > /
script:
# Ajout des options de target si elle est définie
- >
  if [ $DOCKER_TARGET ]; then
    TARGET_OPTIONS="--target $DOCKER_TARGET --skip-unused-stages=true"
  fi
# Tag de l'image selon la version de l'app (cf package.json), si elle n'est pas fournie
- >
  if [ -z $TAG ]; then
    TAG=$(cat $CI_PROJECT_DIR/$WORKING_DIR/package.json | grep version | head -1 | awk -F= '{ print $2 }' | sed 's/[version:,\\,]//g' | tr -d
  fi
# NODE_ENV_CI en "development" et suffixe avec le nom de la branche, si la branche n'est pas main
- >
  if [ $CI_COMMIT_BRANCH != "main" ]; then
    NODE_ENV_CI_BUILD_ARGS="--build-arg NODE_ENV_CI=development"
    TAG=$TAG-$CI_COMMIT_BRANCH
  fi
# Build de l'image docker et push sur harbor
- /kaniko/executor
--context $CI_PROJECT_DIR/$WORKING_DIR
--build-arg NODE_VERSION=$NODE_VERSION $NODE_ENV_CI_BUILD_ARGS
--dockerfile $CI_PROJECT_DIR/$DOCKERFILE_PATH
$TARGET_OPTIONS
--destination $CI_REGISTRY_IMAGE/$IMAGE_NAME:$TAG
```

# Kubernetes

## Cluster

- > Ensemble de nœuds (machines) qui permettent d'exécuter des applications conteneurisées
- > Se trouve dans un état souhaité : applications à exécuter, images à utiliser, ressources allouées...
- > API qui permet de définir l'état souhaité à l'aide d'objets

## Objet

- > Peut représenter un volume, une instance de l'application, un service réseau permettant de communiquer avec l'application...
- > Sous forme de manifeste YAML

## Application

- > Composée de nombreux objets/manifestes



# Kubernetes

## Exemple d'objets

- > **ConfigMap** éléments de configuration (paramètres non secrets) de l'application
- > **Secret** paramètres secrets de l'application
- > **Service** accès à l'application depuis l'intérieur du cluster
- > **IngressRoute** accès depuis l'extérieur du cluster (objet pour Traefik)
- > **Certificate** accès en https à l'application
- > **Deployment** création et mise à jour d'un ensemble de pods identiques (replicaset)
- > **HorizontalPodAutoscaler** scaling horizontal des pods



# Chart Helm

## Helm

Gestionnaire de paquets pour Kubernetes

- > Définition des objets sous forme de templates avec des paramètres
- > Génération des manifestes Kubernetes par substitution des paramètres avec les valeurs fournies

## Chart

Package Helm dossier ou archive tgz

- > **Chart.yaml** fichier de métadonnées du chart (nom, version, dépendances...)
- > **values.yaml** valeurs par défaut des paramètres
- > **templates/** objets kubernetes sous forme de templates
- > **charts/** éventuelles dépendances du Chart



# Chart Helm

## Extraits d'un fichier deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "{{ include "etandem.fullname" . }}-{{ .Values.admin.name }}"
  labels:
    {{- include "etandem.labels" . | nindent 4 }}
spec:
  {{- if not .Values.admin.autoscaling.enabled }}
  replicas: {{ .Values.admin.replicaCount }}
  {{- end }}
  selector:
    matchLabels:
      app.kubernetes.io/name: "{{ include "etandem.name" . }}-{{ .Values.admin.name }}"
      app.kubernetes.io/instance: "{{ .Release.Name }}"
  template:
    metadata:
      {{- with .Values.podAnnotations }}
      annotations:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      labels:
        app.kubernetes.io/name: "{{ include "etandem.name" . }}-{{ .Values.admin.name }}"
        app.kubernetes.io/instance: "{{ .Release.Name }}"
    spec:
      {{- with .Values.imagePullSecrets }}
      imagePullSecrets:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      serviceAccountName: {{ include "etandem.serviceAccountName" . }}
      securityContext:
        {{- toYaml .Values.podSecurityContext | nindent 8 }}
      containers:
        - name: {{ .Chart.Name }}
          securityContext:
```

```
livenessProbe:
  httpGet:
    path: /
    port: http
  initialDelaySeconds: {{ .Values.admin.probes.liveness.initialDelaySeconds }}
  periodSeconds: {{ .Values.admin.probes.liveness.periodSeconds }}
  failureThreshold: {{ .Values.admin.probes.liveness.failureThreshold }}
resources:
  {{- toYaml .Values.resources | nindent 12 }}
env:
  - name: REACT_APP_API_URL
    value: "https://{{ .Values.apiUrl }}.{{ .Values.domain }}"
  - name: REACT_APP_SENTRY_DSN
    value: "{{ .Values.admin.env.sentryDsn }}" 1
{{- with .Values.nodeSelector }}
nodeSelector:
  {{- toYaml . | nindent 8 }}
{{- end }}
{{- with .Values.affinity }}
affinity:
  {{- toYaml . | nindent 8 }}
{{- end }}
{{- with .Values.tolerations }}
tolerations:
  {{- toYaml . | nindent 8 }}
{{- end }}
```

```
admin:
env:
sentryDsn: https://xxx@ulep-glitchtip.univ.fr/2
```

1 - Extrait du fichier values.yaml



# Chart Helm

## Quelques commandes

[https://helm.sh/fr/docs/intro/using\\_helm/](https://helm.sh/fr/docs/intro/using_helm/)

- > **helm create ulep-user-provider** création d'un chart « ulep-user-provider »
- > **helm lint** analyse statique du chart
- > **helm install ulep-user-provider --dry-run --debug ulep-user-provider** génération des différents fichiers manifests kub pour ulep-user-provider sans déploiement dans le cluster
- > **helm install ul-provider ulep-user-provider -f .\ulep-user-provider\values.yaml -f .\ulep-user-provider\values-custom.yaml** installation d'ulep-user-provider dans le cluster sous le nom de release ul-provider, avec application de valeurs personnalisées
- > **helm uninstall ul-provider** suppression de la release ul-provider du cluster

Déploiement en local : kubectl & Rancher Desktop

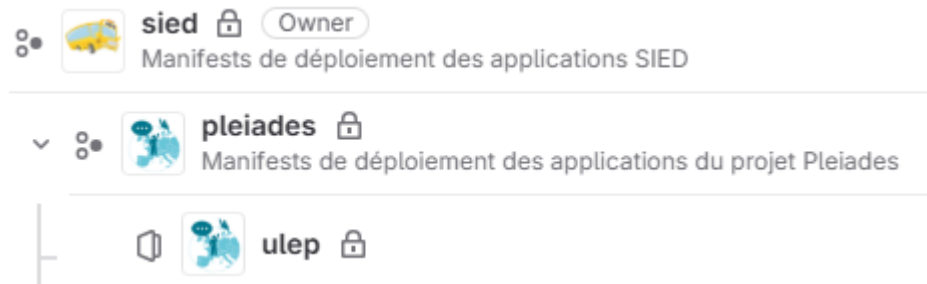


# Déploiement continu

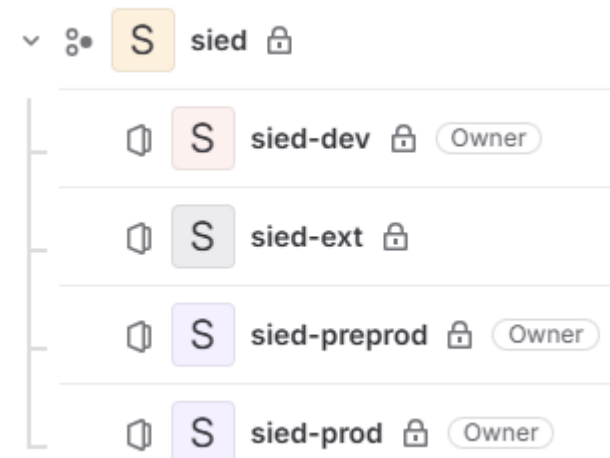
## GitOps

- > Utilisation de référentiels Git comme unique source de vérité (déclaratif)
- > Surveillance de celle-ci par un agent qui s'assure de la cohérence avec la plateforme

### Dépôts Git « charts »



### Dépôts Git « clusters »



# Déploiement continu

## Argo CD

Outil déclaratif de livraison continue  
GitOps pour Kubernetes

> Applications déclarées dans le Git du cluster (un dossier par application)

> **app.yaml** application Argo CD

> **values.yaml** valeurs spécifiques au cluster cible pour l'application déployée

*Note : les secrets sont stockés dans Vault. Ils sont insérés sous cette forme dans le fichier values.yaml :*  
<path:kv/data/sied/ulep#keycloakClientSecret>



# Déploiement continu

## Argo CD – Exemple d'application

### app.yaml

```
---
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: ulep
  namespace: argocd
  finalizers:
    - resources-finalizer.argocd.argoproj.io
spec:
  destination:
    namespace: sied-ulep
    server: https://kubernetes.default.svc
  project: sied
  source:
    repoURL: https://gitlab.univ-lorraine.fr/paas/applications/sied/pleiades/ulep.git
    path: .
    targetRevision: HEAD

  syncPolicy:
    syncOptions:
      - CreateNamespace=true
```

### values.yaml (extrait)

```
api:
  replicaCount: 1
  image:
    repository: harbor.paas.univ-lorraine.fr/sied/pleiades/ulep/api
    tag: "0.0.20"
    pullPolicy: Always
  initImage:
    repository: harbor.paas.univ-lorraine.fr/sied/pleiades/ulep/api-init
    tag: "0.0.20"
    pullPolicy: Always
  env:
    adminUrl: *adminUrl
    appUrl: *webappUrl
    logLevel: warn
    keycloakClientSecret: *keycloakClientSecret
  autoscaling:
    enabled: true
    minReplicas: 2
    maxReplicas: 10
    targetCPUUtilizationPercentage: 80
    # targetMemoryUtilizationPercentage: 80
  resources:
    requests:
      cpu: 200m
      memory: 256Mi
    limits:
      cpu: 1000m
      memory: 512Mi
```





# Pour aller plus loin

## Helm

- > <https://blog.wescale.fr/simplifiez-le-d%C3%A9ploiement-de-vos-applications-avec-le-templating-helm>
- > <https://blog.wescale.fr/helm-pour-les-nuls-et-moins-nuls>

## Argo CD

- > <https://blog.wescale.fr/flux-et-argocd-deux-visions-du-gitops-sur-kubernetes>





# Merci de votre attention

Cédric Champmartin [cedric.champmartin@univ-lorraine.fr](mailto:cedric.champmartin@univ-lorraine.fr)  
Direction du Numérique – Université de Lorraine

