DÉPLOIEMENT D'APPLICATIONS AVEC KUBERNETES

Préproduction de SI dans l'équipe Info&Sols

Julien Robert (julien.robert@inrae.fr)

4 Juillet 2023

CONTEXTE TECHNIQUE

- Mise en place d'un cluster k8s
- Déploiement d'applications sur ce cluster
    - Utilisation effective pour 2 applications
    - Autonomie sur les outils pour 3 collègues
- Déploiement continu avec gitlab CI/CD sur une application

CONTEXTE HUMAIN

- Équipe
  - Unité Info&Sols (fusion URSOL et INFOSOL depuis janvier)
  - ~55 personnes
  - Dont 20 informaticiens.ennes (Bap E.)
  - Dont 7 développeurs.euses
- Julien Robert
  - Indépendant, expert DevOps & Infrastructure as Code
  - Formateur git, gitlab, docker, kubernetes, ansible
  - En mission dans l'unité Info&Sols entre septembre 2023 et juin 2024

DÉPLOIEMENT D'APPLICATIONS AVEC KUBERNETES - PLAN

1. Le besoin
2. Avant docker
3. Avec docker
4. Avec kubernetes
5. Bilan : les + et les - et les suites à donner
6. Mise en place du cluster

LE BESOIN

- Type de code : SI développés en interne ou par des prestataires
- Mettre du code en préproduction
    - pour béta-tests / démo / ..
    - avec mise à jour automatique et rapide (déploiement continu)
    - permettant de suivre les logs
    - sans impacter les autres développeurs
    - ressemblant à un environnement de production (ressemblant lui même à l'environnement de dev)

EXEMPLE FIL ROUGE

- Architecture
  - Front (ex: VueJS)
  - Back (ex : Springboot)
  - Base de données (ex : Postgres) administrée par ailleurs (en dehors du cluster)
- 1 dépot git pour le back, 1 pour le front (forgemia)
- Un seul accès : https://monappli.inrae.fr transmis par un proxy
  - route / : frontend
  - route /api/ : backend

AVANT DOCKER

- Administration d'une VM
  - Installation et maintenance de l'OS
  - Installation et maintenance des outils (nodejs/java/haproxy/...)
  - Paramétrages (proxy)
- Gestion des certificats
- Lancement et mise à jour
  - Clonage des dépôts
  - Création & lancement de services

AVEC DOCKER

- Administration d'une VM "coquille vide" + Gestion des certificats
- Dépôt "deploiement" contenant un docker-compose.yaml

```
version: '3.7'
services:
  reverse_proxy:
    image: haproxy:vXX
    volumes:
       - haproxy.conf:/etc/haproxy.conf # règles proxy vers back/front
       - cert.pem:/etc/ssl/certs/cert.pem #certificat ssl

    ports:
       - "443:443"
  frontend:
    image: registry.forgemia.inra.fr/monappli/front:v2023-05-14

  backend:
    image: registry.forgemia.inra.fr/monappli/back:v2023-05-14
    environment:
```

- Lancement et mise à jour : `docker compose up`

AVEC DOCKER

- Les +
  - Administration de la VM réduite au minimum
  - Environnement documenté (Dockerfile / docker-compose) **reproductible** , **auditable** (sécurité)
- Les -
  - Couche docker en + (compétences à acquérir)
- Nécessite pour le développeur
  - Concepts docker (image, conteneur, volume, redirection de port)
  - Syntaxe & concepts docker-compose
  - Aisance avec la ligne de commande docker (run, exec, logs, ..)
  - Bonnes pratiques Dockerfile
  - Gestion des images docker (manuel ou via pipelines CI/CD)

AVEC K8S - EN PRATIQUE

- Dépôt "déploiement"
- Contient un "helm chart" (équivalent docker-compose, généré semi-automatiquement) :
    - des templates pour chaque objet k8s
    - un fichier values.yaml avec les paramètres

```
.
└── charts
    ├── back
    │   ├── templates
    │   │   ├── deployment.yaml # responsable du conteneur back
    │   │   ├── ingress.yaml # règles "proxy" back (monappli.inrae.fr/api/)
    │   │   └── service.yaml
    │   └── values.yaml # paramétrage
    └── front
        ├── templates
        │   ├── deployment.yaml # responsable du conteneur front
        │   ├── ingress.yaml # règles "proxy" front (monappli.inrae.fr/)
        │   └── service.yaml
        └── values.yaml
```

AVEC K8S - EN PRATIQUE (VALUES.YAML DU BACK)

```yaml
# Value.yaml du back
replicaCount: 1
image:
  repository: registry.forgemia.inra.fr/monappli/back
  tag: "v2023-05-17"
ingress:
  hosts:
    - host: monappli.inrae.fr
      paths:
        - path: /api/(.*)
```

DÉMONSTRATION

Une fois ce helm chart déposé sur la forge:

- argocd pour l'installer sur notre cluster (démonstration avec dw4-dev)
- synchronisation automatique
- rollback / suivi basique des logs / debuggage k8s / via argocd

AVEC K8S

- Les +
  - Pas d'administration de VM ; moins de VMs
  - Duplication très simple (beta.appli1.inrae.fr ; alpha.appli1.inrae.fr, ..)
  - Profite de l'écosystème k8s
    - gestion des certificats automatisée
    - configuration proxy simplifiée
    - (montée en charge / haute disponibilité gratuite)
    - rollback
    - sécurisation
    - monitoring & centralisation des logs
- Les -
  - Besoin de formation k8s
  - Administration du cluster & Complexité de la machinerie
- Nécessite pour le développeur
  - Notions de k8s (Docker + pod / deployment/ingress/secrets/configmap)
  - Templating & un peu de Helm

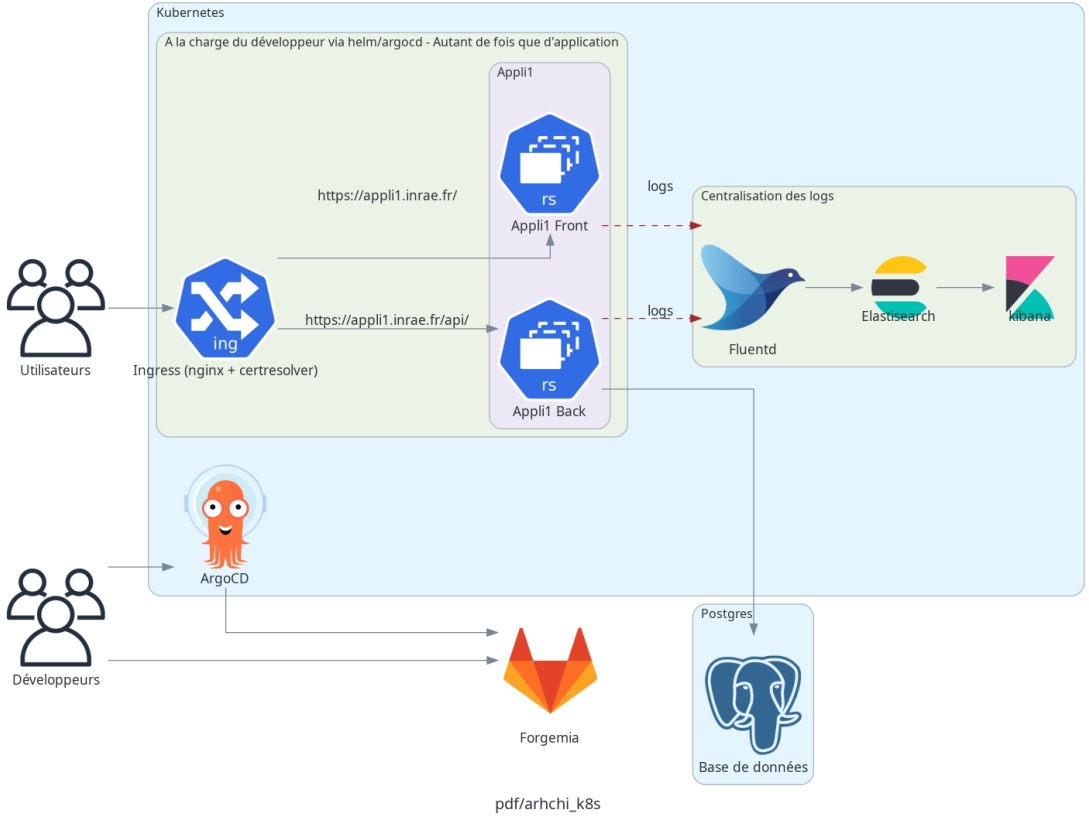
RÉSUMÉ POINT DE VUE DÉVELOPPEUR

- Créer et pousser sur la forge les images docker du front et du back (nécessite aisance avec docker)
- Créer un helm chart (semi-automatisé; nécessite de bonnes bases k8s)
- Déployer/mettre à jour/vérifier que ça marche en clic-bouton via argocd)

BILAN

- Les +
  - Utilisation effective pour 2 applications (dont une dans deux versions)
  - Autonomie de 3 collègues
  - Beaucoup d'outils à faible coût
  - Gestion automatique des certificats appréciée
- Les -
  - Peu de recul sur la maintenance du cluster
  - Beaucoup de concepts, de couches, forte courbe d'apprentissage
- La suite
  - Asseoir le savoir faire : manipuler, manipuler, manipuler...
  - Poursuivre la prise en main avec le reste de l'équipe
  - Profiter des outils de l'écosystème et aller plus loin :
    - Dashboard de logs
    - Monitoring
    - Function as a Service
    - Sécurisation
    - ...

LES ENTRAILLES - VUE D'EN HAUT



LES ENTRAILLES - ET EN DESSOUS

- Au moins 3 machines (VMS) suffisamment "costaudes" (RAM/coeurs/disques)
- Déploiement et maintenance du cluster avec kubespray, un outil basé sur ansible permettant de :
    - déployer etcd (pour des données distribuée dans le cluster), déployer le controle pane (les "maitres" du cluster), déployer les noeuds (les "esclaves"), déployer l'ingress controller
- Déploiement des add-ons essentiels (kubectl apply): calico (communication entre pods), cert-manager, argocd, EFK
- Paramétrage Argocd (configmap)
- Paramétrage EFK (configmap + appels api elasticsearch)
- Nécessite
    - compréhension de l'architecture d'un cluster k8s
    - connaissance de ansible
    - du travail !