Projet

Le but du projet est de mettre en place un chaine de déploiement automatisée d'une application Web développée dans des technologies cloud-native sur une infrastructure que vous pilotez.

L'application est une application de type doodle like qui offre la possibilité de déployer aussi une intégration avec un service de type etherpad pour la prise de notes de réunions. Nous souhaitons que vous fassiez au minimum le travail pour

- Faciliter son déploiement à l'aide de containers
- Faciliter sa configurabilité au déploiement (serveur smtp à utiliser, connexion à l'etherpad, …)
- Faciliter sa sécurisation au moment du déploiement

Ces tâches là sont appélés le niveau 1 du projet dans le suite de ce texte.

Pour ceux qui veulent aller plus loin, vous pouvez explorer quatre autres pistes/aventures autour du cloud pour cette matière TLC. Ces trois pistes sont indépendantes et une, deux ou les trois de ces pistes peuvent être menées.

- Au choix:
  - **Aventure 1**: Mettre en place un mécanisme de déploiement continue qui permette de déployer la nouvelle version de l'application automatiquement si un nouveau commit est activé sur le back ou le front. L'idée de ce travail est d'explorer les liens entre pratique devops (intégration/déploiement continue) et outil de déploiement en mode cloud native
  - **Aventure 2**: Mettre en place des outils de monitoring avancés de cette application et des dashboard associés à ce monitoring
  - **Aventure 3**: Utiliser K8S comme orchestrateur pour le déploiement de nos containers
  - **Aventure 4**: mettre en place un identity provider et ou une passerelle d'API pour mettre de faire de l'accounting sur l'API de l'application
- Voici une petite vidéo à venir de présentation des fonctionnalités de l'application.
- Voici une petite vidéo à venir de présentation de l'architecture de l'application.
- Voici une petite vidéo à venir de revue de code de l'application.
- Voici une petite vidéo à venir pour aider à la compréhension du projet afin de comprendre le travail à faire sur les premières étapes.

Voici le repo de code source à venir de cette application. Vous pouvez regarder le code source, le *back* est développé en quarkus.io et le front en angular.

Backlog du projet

Niveau 1: minimum à faire

- **Tâche 0**. Créer une machine virtuelle (https://vm.istic.univ-rennes1.fr/ séléctionner **ubuntu20** comme image de base) et demandez l'accès externe vers le port *80* (http) et *443* (https) de votre machine virtuelle par le helpdesk, catégorie ISTIC-ESIR - Tous problèmes informatiques, l'accès au port 22 se fera au travers du VPN. Partager moi l'adresse IP de la machine son nom
- **Tache 1**. Fournir un ou plusieurs docker file(s) et un ou plusieurs docker compose(s) pour cette application permettant de facilement déployer et configurer l'application. (Faites un fork du projet avant.)

 https://nearsoft.com/blog/how-to-synchronize-your-github-fork/

[1] docker compose documentation
[2] docker file documentation

 Vous aurez du mal à avoir un fonctionnement correct à cette étape là. En effet, le code du front va faire ces requêtes *REST* à la même adresse que celui qui lui a fourni le code html, css et js pour éviter les problèmes de CORS. Il est donc nécessaire de se forcer à configurer le serveur nginx qui délivre le front pour faire *proxy_pass* quand il reçoit une requête sur la route */api* ou une sous route de */api*. Ne vous inquiétez pas, on configure cela à l'étape suivante.

- **Tâche 2**,

Il va falloir maintenant configurer le serveur Web du Front pour qu'il soit capable de servir de point d'entrée à l'ensemble des requêtes puis qu'il les *route* vers le bon service de Back. Il est possible de mettre en place un serveur Web spécifique pour gérer ce routing (on le nomme alors la gateway d'API). On peut aussi dans notre cas se servir du fichier nginx du front pour router les requêtes.

Nous fournissons trois examples de fichiers de configuration nginx pour configurer les trois services. Pour plus d'information sur la configuration nginx, RDV sur ce très bon site Web et Proxy nginx explained.

```
server {
    listen       80;
    listen  [::]:80;
    # server name to change based on your own domain name for doodle
    server_name  doodle.tlc.fr;

    location /api {
        proxy_pass http://api:8080/api;
        proxy_set_header Host $http_host;

    }

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html?$args;

    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

```
server {
    listen       80;
    listen  [::]:80;
    # server name to change based on your own domain name for doodle
    server_name  myadmin.tlc.fr;

    location / {
        proxy_pass http://myadmin:80;
        proxy_set_header Host $http_host;

    }
```

```
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

```
server {
    listen       80;
    listen  [::]:80;
    # server name to change based on your own domain name for doodle
    server_name  pad.tlc.fr;

    location / {
        proxy_pass http://etherpad:9001;
        proxy_set_header Host $http_host;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header X-Real-IP  $remote_addr;
        proxy_set_header X-Forwarded-For $remote_addr;
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

- **Tâche 3**, déployer correctement une première fois votre application en configurant convenablement la partie DNS pour le reverse proxy, letsencrypt pour le certificat côté serveur et ufw pour le firewall sur votre machine virtuelle.

En gros vous allez prendre votre fichier docker-compose, votre fichier de configuration nginx, votre front et mettre cela sur votre vm. Mettre les bonnes variables de configuration dans ces deux fichiers. Vous aurez besoin soit de builder les images sur la vm soit de pushé vos images sur le docker*hub afin de pouvoir les _puller* depuis votre VM.

- **Tâche 4**, Documenter, à l'aide d'un diagramme de déploiement UML ou autre notation, le déploiement réalisé pour le moment sur votre machine virtuelle.
- Voici une petite vidéo à venir pour aider à la compréhension de la première étape du projet, à savoir la construction d'un *DockerFile* pour le front.
- Voici une petite vidéo à venir pour aider à la compréhension de la première étape du projet, à savoir la construction d'un *DockerFile* pour le *back*.
- Voici une petite vidéo à venir pour aider à la compréhension de la deuxième étape du projet, à savoir la défnition d'un *docker-compose* pour le déploiement de l'application.
- Voici une petite vidéo à venir pour aider à la compréhension de l'étape sur le déploiement de l'application avec K8S. note de présentation accessible à venir

Vous pouvez maintenant choisir selon vos envies une, deux trois ou quatre de ces quatres aventures au choix.

Aventure 1: Decploiement continue à l'aide de gitlabCI et déploiement à l'aide de Docker.

- **Aventure 1**, mettre en place un mécanisme de déploiement continue qui permette de déployer la nouvelle version de l'application automatiquement si un nouveau commit est activé sur le back ou le front. Vous pourrez mettre en place votre outil d'intégration continue sur la même VM même si c'est une pratique de nous ne suivrions pas pour un déploiement réel. Vous avez le choix pour l'outil d'intégration continue.

Vous pouvez utiliser gitlabCI de l'ISTIC si vous avez mis le code dans un repo du gitlab de l'ISTIC. Vous pouvez alors créer votre propre runner gitlab ci (https://docs.gitlab.com/runner/) ou utiliser travis sur github ou les runners de gitlab.com. Vous pouvez aussi déployer un jenkins sur votre machine si vous préférez.

Aventure 2: Chaîne de monitoring de l'application en production.

- **Aventure 2** mettre en place la chaine de monitoring de l'application à l'aide de Promotheus et grafana comme front http://www.mastertheboss.com/soa-cloud/quarkus/monitoring-quarkus-with-prometheus.
  Mettre en place le sous domaine et le certificat letsencrypt pour grafana.
  Mettre en place munin pour la surveillance de votre machine virtuelle

Aventure 3: Utilisation de K8S comme orchestrateur d'un petit cluster

- **Aventure 3**, On souhaite maintenant utiliser kubernetes pour déployer l'ensemble des micro-services. Le but est de pouvoir comparer et comprendre les abstractions

supplémentaires fournies par kubernetes par rapport à un seimple déploiement d'une application sur un seul noeud à l'aide de docker et docker-compose. En particulier, on souhaite ajouter de la redondance pour la partie back de l'application (redondance uniquement sur ce micro-service là. Surout pas sur la BD relationnelle, c'est plus compliqué et pas complètement transparent). Déployer donc microk8s sur votre VM.
- Utiliser kubernetes pour mettre en place un minimum de redondance pour le back de l'application (on ne clusterisera pas la BD dans un premier temps).

 Nous utiliserons https://microk8s.io/

Vous pouvez trouver quelques notes autour de microk8s ici

Vous pouvez utiliser le promotheus et les outils de monitoring de microk8s

Aventure 4: Mise en place d'un outils d'autentification et d'une passerelle d'API

- **Aventure 4**, mettre en place keycloak pour un accès sécurisé à vos ressources Web et/ou une passerelle d'API pour mettre de faire de l'accounting sur l'API de l'application. Je vous propose l'utilisation d'ambassador
  - (https://blog.getambassador.io/explore-the-ambassador-api-gateway-with-microk8s-f75a7a295113)

 Le projet est très/trop ambitieux. Allez y à votre rythme. Le strict minimum à atteindre est la tâche 4, j'aimerai que vous atteignez la tâche 6 pour un maximum de personnes. Je serai ravi que certains aillent jusque la tâche 7. Ceux qui vont jusque la tâche 8 auront évidemment mon respect eternel et une recommandation de ma part sur linkedin ;).

 Il va falloir lire beaucoup de documentation, posez beaucoup de questions, faire des schémas pour comprendre ce que vous êtes en train de faire. Utilisez les tickets github dès que quelque chose ne marche pas comme vous le souhaitez.

Evaluation

Pour le rendu je propose d'utiliser le formulaire suivant. https://forms.gle/NsBtwLK81CtYBDsN7 . Comme d'habitude un petit readme à la racine des différents repos.

J'attends un rendu sur le TP Docker, un petit rendu sur la partie ansible et un rendu sur la partie projet. Cela fera trois notes séparées.

Date de rendu 27/03 23h59 ferme

Je sais que cela correspond pour certains non apprenti à une période où vous êtes en stage donc finissez avant le départ en stage. Je corrigerai dans la semaine car les notes sont à renvoyer à Brigitte pour le 1/04 donc pas possible de décaler plus.

Other relevant videos

- What are cloud native apps
- Cloud native applications
- Cloud Native DevOps Explained
- What are Microservices?
- Advantages of Using OpenAPI to design APIs
- What is DevOps?
- What is Continuous Integration?
- What is Continuous Delivery?
- What is Infrastructure as Code?
- Cloud-native Java for this decade with Quarkus
- Quarkus.io
- Principles of Antifragile Software
- Mastering Chaos - A Netflix Guide to Microservices
- Getting started with Chaos Engineering

References

[1] BALALAIE, Armin, HEYDARNOORI, Abbas, et JAMSHIDI, Pooyan. Microservices architecture enables devops: Migration to a cloud-native architecture. Ieee Software, 2016, vol. 33, no 3, p. 42-52.

[2] https://www.redhat.com/en/resources/eight-steps-cloud-native-application-development-brief

[3] D. S. Linthicum, “Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points Between,” in IEEE Cloud Computing, vol. 4, no. 5, pp. 12-14, September/October 2017, doi: 10.1109/MCC.2017.4250932.

Questions réponses

- Mettre en place un serveur Web
- Configurer nginx
- Nginx: Mettre en place letsencrypt
- Nginx : Se protéger des attaques Flood
- Fail2ban c’est quoi
- un firewall c’est quoi: configuration de ufw, vidéo
- une autre présentation de docker
- Pas mal de tutos assez bien fait
- POC en local avec Docker et Traefik
- Mise en place et debug d’Influxdb avec Collectd et Grafana

Appendix

-  **Tâche 2 bis (NON Obligatoire, UNIQUEMENT pour ceux que cela intéressent)**,

Pour servir d’API Gwateway, il y a un super projet que j’adore qui s’appelle **bunkerweb**. C’est un container docker avec nginx qui se configure au travers de variable d’environement et qui inclus de base plein de mécanisme de sécurité pour nginx, la génération des certificats … mais il entraine pas mal de doc à lire.

 **UNIQUEMENT pour ceux que cela intéressent (j’insite)**

configurer convenablement https://github.com/bunkerity/bunkerweb au sein de votre docker-compose pour sécuriser votre application. Tester localement le déploiement de l’application.

nginx bunkerity/bunkerweb va jouer le rôle de passerelle d’API, c’est-à-dire que toutes les requêtes passent par lui et il a la charge de fournir les fichiers html, css et js du front, mais aussi de servir de proxy vers les autres services.

Pour cela, il est nécessaire de configurer finement ce serveur nginx qui tourne dans ce container. Il faudra *monter* ce fichier de configuration au moment du lancement de ce container. (voir https://github.com/bunkerity/bunkerized-nginx/blob/700dfc0184f8d12e15f3ca3d6e5ca08befccd561/examples/wordpress/docker-compose.yml#L14)

Vous trouverez ci-dessous un exemple de fichier de configuration nginx (fichier api.conf par exemple dans le répertoire *server-confs* là où se trouve votre docker-compose.yml).

```
proxy_set_header Host $host;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

# L'idée ici est que toute requête qui arrive sur /api ou une sous
# route d'API passe par ici et en gros on fait un if selon que cette
# requête est arrivé sur l'adresse IP en demandant à parler à
# doodle.diverse-team.fr ou à  phpmyadmin.diverse-team.fr ou à
# pad.diverse-team.fr
location /api {
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
        if ($host = doodle.diverse-team.fr) {
                proxy_pass http://doodleback:8080$request_uri;
        }
        if ($host = pad.diverse-team.fr) {

                proxy_pass http://etherpad:9001$request_uri;
        }
        if ($host = phpmyadmin.diverse-team.fr) {
                proxy_pass http://myadmin$request_uri;
        }
}

# L'idée ici est que toute requête qui arrive sur / ou une sous
# route de / qui n'a pas été traité avant passe par ici et en gros
# on fait un if selon que cette requête est arrivé sur l'adresse IP
# en demandant à parler à  phpmyadmin.diverse-team.fr ou à
# pad.diverse-team.fr et pour tout le reste on fourni les ressources
# statiques du front
```

```
location / {
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_redirect off;
                proxy_set_header X-Real-IP  $remote_addr;
                proxy_set_header X-Forwarded-For $remote_addr;

                if ($host = pad.diverse-team.fr) {

                proxy_pass http://etherpad:9001$request_uri;
                }
                if ($host = phpmyadmin.diverse-team.fr) {
                proxy_pass http://myadmin$request_uri;
                }
                try_files $uri $uri/ /index.html;

}
```

> A compléter ...