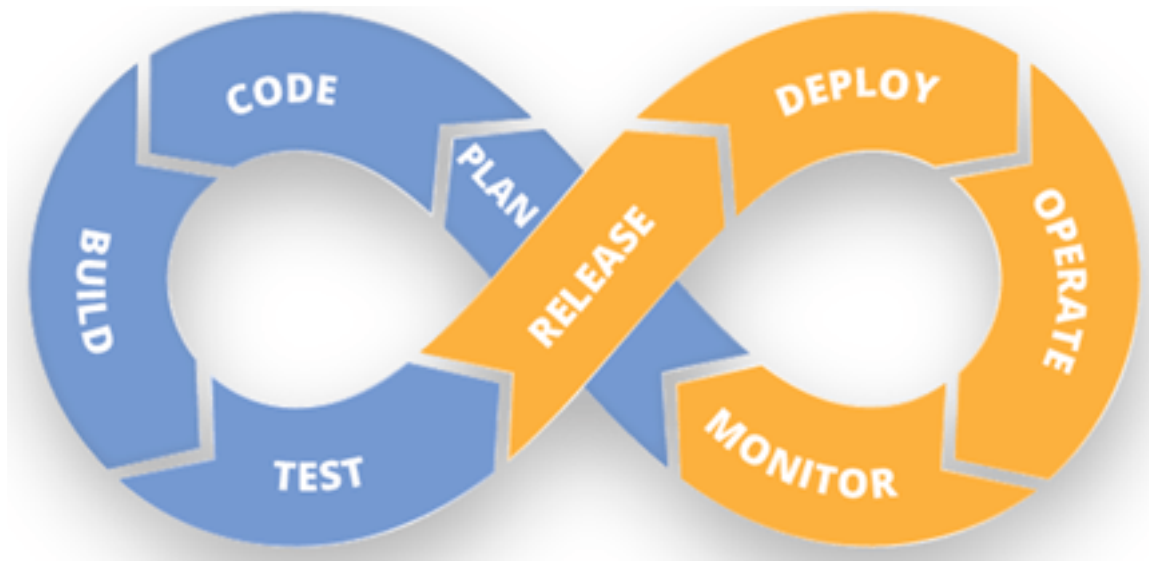

Rapport du projet de Devops

Table of Contents

<u>I.</u>	<u>QU'EST-CE QUE DEVOPS ?</u>	<u>2</u>
<u>II.</u>	<u>OBJECTIF</u>	<u>2</u>
<u>III.</u>	<u>MISE EN PLACE DE LA STRUCTURE</u>	<u>3</u>
<u>IV.</u>	<u>FONCTIONNALITES DE L'APPLICATIONS.....</u>	<u>3</u>
<u>IV.</u>	<u>RESULTAT</u>	<u>4</u>
<u>V.</u>	<u>MISE EN PLACE DE LA STRUCTURE</u>	<u>13</u>
<u>VI.</u>	<u>LIENS DU PROJET ET INTEGRATION.....</u>	<u>25</u>

I. Qu'est-ce que DevOps ?

DevOps est un mouvement en ingénierie informatique et une pratique technique visant à l'unification du développement logiciel (dev) et de l'administration des infrastructures informatiques (ops), notamment l'administration système.



DevOps, c'est en effet :

- Un développement constant : le projet est développé en respectant le cycle PDCA (Planifier, Faire, Vérifier et Ajuster)
- Des tests constants : chaque fonctionnalité de l'application est testée soit en utilisant le TDD ou le test de parité ou les deux à la fois.
- Une intégration constante : le code est envoyé en permanence à un gestionnaire de code source comme GitHub pour être livré ou partagé avec les autres membres de l'équipe
- Une mise en œuvre constante : Les tâches sont définies et assignées aux membres de l'équipe. Elles sont aussi organisées par itération ou sprint.
- Une surveillance constante : Chaque tâche développée et approuvée est ensuite analysée et déployée sur le serveur de production.

II. Objectif

Il était question pour la spécialisation Devops de mettre en place un système Devops utilisant les éléments suivants :

- Un code source managé avec git
- Un outil de planification et de la gestion des builds, du code source, des projets et l'analyse des performances à l'instar d'azure Devops
- Un registre docker pour la sauvegarde des images
- Un serveur de déploiement pour la livraison du projet.

III. Mise en place de la structure

Pour la réussite de ce projet, nous avons mis en place les éléments suivants :

- Un projet fullstack managé en local avec git
- Azure devops: qui nous servira comme outil d'intégration continu, de test d'intégration, gestion du code et de surveillance.
- LWS: qui sera le serveur sur lequel sera déployé la nouvelle image docker gérée
- Docker hub: qui va permettre de stocker nos images

IV. Fonctionnalités de l'applications

- L'application est déployée et configurée de façon automatique sur le serveur de production
- Le backend implémente des services **CRUD** et fait un enregistrement dans une base données MongoDB cloud.
- Le backend utilise jest pour faire des tests unitaires des services et une base de données in-memory basé sur MongoDB cloud.
- Le frontend fait appelle au backend pour enregistrer, modifier, supprimer et lister les post.
- Chaque projet contient un fichier **Dockerfile** qui permet de manager l'application
- Le fichier **compose** est créé pour faciliter la gestion.
- Le projet est envoyé en permanence sur azuredevops (repos) plus précisément la branche danick à travers git

- Les tâches sont configurées sur azuredevops (boards) pour faire la liaison en tant qu'opération avec les tâches du développement
- Un fichier de configuration yaml (ansible) est configuré pour :
 - Builder et déployer les images docker sur le hub
 - Créer un artifact basé sur le projet, faire les tests unitaires des deux projets, vérifier la syntaxe et enfin faire une couverture de tous les tests unitaires.
 - Un artifact est créé pour le pré-déploiement et contient le code du frontend et du backend archivé.
 - Un thread est créé pour copier le backend et le frontend sur le serveur de productions en utilisant ssh
 - Un autre thread est créé pour normaliser les fichiers (dé zipper, supprimer les répertoires non utiles, copier les répertoires et fichiers sur le bon emplacement et démarrer les services)
- Le sous domaine est configuré sur le serveur pour héberger le frontend et le backend
- Pm2 est exécuté le backend sur le port local 3010 et un fichier vhost est configuré pour mapper le sous domaine api-danick-devops.be avec localhost :3010

IV. Résultat

Backend : le backend est disponible via le lien <https://api-danick-devops.datnek.be/api/posts>

```

<  →  ↻  🏠  🔒  api-danick-devops.datnek.be/api/posts  📖  ☆
🌐 Apps  📁 Supinfo  📁 Fastdiscarry  📁 Neerlandais  📁 2020  📁 AJAXNIV  📁 musique  📁 site interessant  📁 site gratuit  |

{
  {
    "_id": "5f08939fe7fc9931cbf0f6e7",
    "title": "Authentification jwt update",
    "description": "Mise en place d'un code d'authentification avec python et jwt",
    "createdAt": "2020-07-10T16:13:19.919Z",
    "updatedAt": "2020-07-10T16:14:00.143Z",
    "__v": 0
  },
  {
    "_id": "5f08c4a45ea4800ed2609364",
    "title": "deployment devops",
    "description": "exemple de déploiement",
    "createdAt": "2020-07-10T19:42:28.687Z",
    "updatedAt": "2020-07-10T19:42:28.687Z",
    "__v": 0
  },
  {
    "_id": "5f0b532d581c9f71650f181e",
    "title": "tutoriel Specialisation devops",
    "description": "Ceci est un excellent tutoriel",
    "createdAt": "2020-07-12T18:15:09.480Z",
    "updatedAt": "2020-07-12T18:15:09.480Z",
    "__v": 0
  }
}

```

- Frontend : le frontend est disponible via le lien <https://danick-devops.datnek.be>

danick-devops.datnek.be

Gestion des produits Accueil

Enregistrement d'un post

Titre

Description

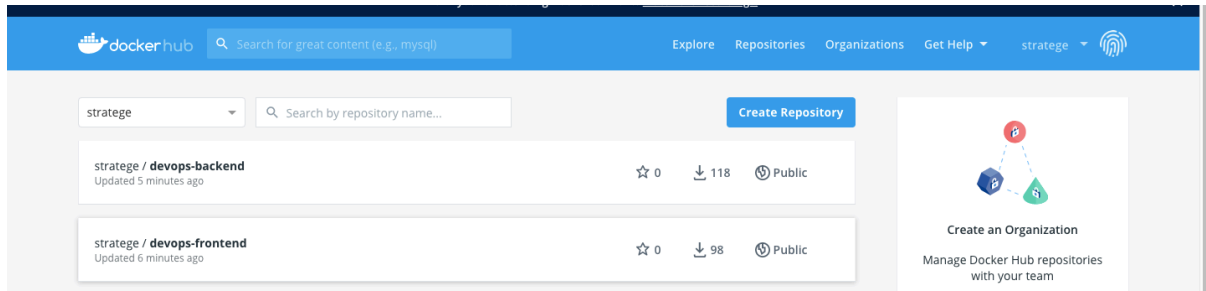
Enregistrer

Nombre de post
3

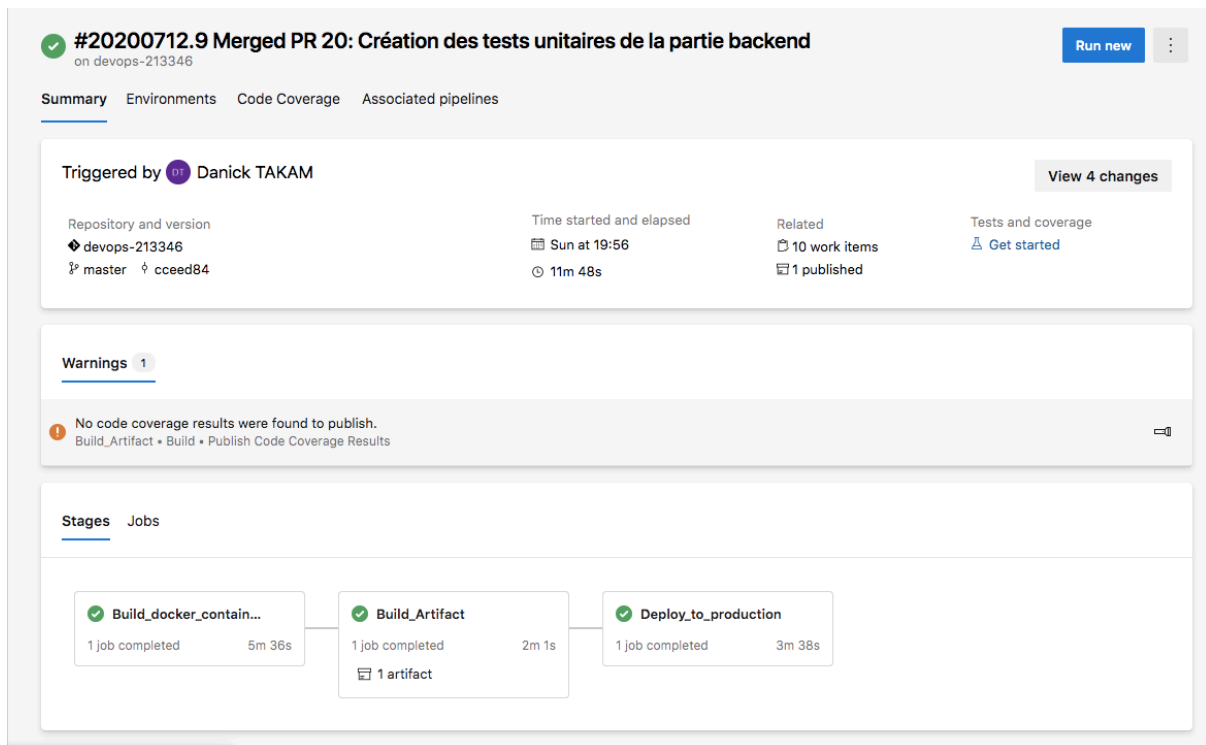
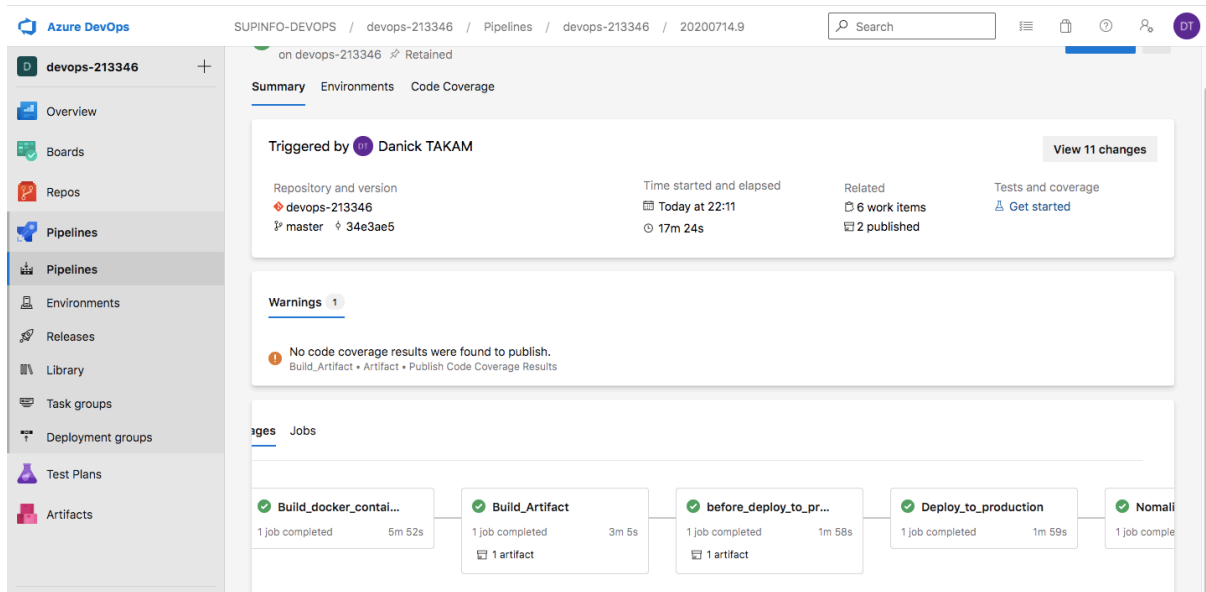
Liste des postes

Name	Description	
Authentification jwt update	Mise en place d'un code d'authentification avec python et jwt	<div>modifier</div> <div>supprimer</div>
deployment devops	exemple de déploiement	<div>modifier</div> <div>supprimer</div>
tutoriel Specialisation devops	Ceci est un excellent tutoriel	<div>modifier</div> <div>supprimer</div>

- Docker hub



- Build et déploiement: il y a 5 étapes



SummaryEnvironmentsCode CoverageAssociated pipelines

Repository and version
devops-213346
master d05e3f9

Time started and elapsed
Sun at 00:24
6m 11s

Related
6 work items
1 published

Tests and coverage
Get started

Errors 2

[PM2][ERROR] Process or Namespace danick-devops not found
Deploy_to_production • Production • Config server

Command failed with errors on remote machine.
Deploy_to_production • Production • Config server

Troubleshooting failed runs

StagesJobs

Build_docker_contain...
1 job completed 5m 26s
1 artifact

Deploy_to_production
Failed 24s
Production 22s
Rerun failed jobs

#20200710.10 Merged PR 10: modification de la chaine de connexion
on devops-213346

Run new

SummaryCode Coverage

Triggered by Danick TAKAM

Repository and version
devops-213346
master 7f538e6

Time started and elapsed
Fri at 23:46
6m 20s

Related
6 work items
0 artifacts

Tests and coverage
Get started

Warnings 2

No code coverage results were found to publish.
Publish Code Coverage Results

Directory '/home/vsts/work/1/a' is empty. Nothing will be added to build artifact 'devopsfordanick-artifact'.
Publish Artifacts

Jobs

Name	Status	Duration
Build	Success	6m 16s

On constate que plus les étapes sont longues plus le temps de déploiement est long. Ce qui montre ici une mauvaise performance du déploiement avec plusieurs étapes caractérisées par plusieurs connexion et déconnexion. En outre, on constate que plus les tâches sont séparées, les étapes ressortent et on a une bonne maîtrise des environnements et des séquences.

Runs	Branches	Analytics	
#20200714.12 Merged PR 24: modification des fichiers dockerfile	Individual CI for master 6600bac	✓-✓-✓-✓-✓	22m ago 18m 11s
#20200714.9 Update azure-pipelines.yml for Azure Pipelines	Individual CI for master 34e3ae5	✓-✓-✓-✓-✓	1h ago 17m 24s
#20200712.9 Merged PR 20: Création des tests unitaires de la partie backend	Individual CI for master cceed84	✓-✓-✓	Sunday 11m 48s
#20200712.7 Merged PR 19: correction du problème avec les header	Individual CI for master 719fe85	✓-✓-✓	Sunday 14m 14s
#20200712.6 Merged PR 18: correction du problème avec le background	Individual CI for master 73b5bc4	✓-✓-✓	Sunday 11m 15s
#20200712.5 Merged PR 17: modification de la barre de menu	Individual CI for master 36e064f	✓-✓-✓	Sunday 9m 12s
#20200712.4 Update azure-pipelines.yml for Azure Pipelines	Individual CI for master 26848af	✓-✓-✓	Sunday 13m 45s
#20200711.17 Update azure-pipelines.yml for Azure Pipelines	Individual CI for master 34c30e7	✓-✓-✗	Sunday 13m 1s
#20200711.15 Update azure-pipelines.yml for Azure Pipelines	Individual CI for master d05e3f9	✓-✗	Sunday 6m 11s
#20200710.10 Merged PR 10: modification de la chaine de connexion	Individual CI for master 7f538e6	✓	Friday 6m 20s

- Artifacts : le livrable plus léger contient le zip du frontend et le backend. C'est ce livrable que nous utiliserons

← Artifacts	
Published	
Name	Size
> docker-compose	880 MB
▼ docker-compose-zip	50 MB
> apizip	48 MB
> clientzip	2 MB

- Tâches : le projet développé utilise agile, j'ai planifié quelques tâches et fait une liaison des opérations avec le développement. Une bonne démonstration peut être perçu dans la vidéo.

PROJET DE DEVOPS

devops-213346 Team

No iteration dates
Set dates

Taskboard Backlog Capacity Analytics

+ New Work Item

Column Options

...

Iteration 1

...

Order	Title	State	Ass
1	initialisation de l'application <ul style="list-style-type: none">Initialisation du front endConfiguration du fichier kamar.confCréation et configuration du fichier dockerfileConfiguration du fichier .gitignore	Closed	
2	Création d'un projet nodejs <ul style="list-style-type: none">Initialisation du projetCréation du fichier dockerfileCréation d'un fichier .dockerignoreAjout du fichier serveur + CR de post	Closed	
3	Post <ul style="list-style-type: none">Ajout d'un postListe des postesAjout du fichier docker-compose	Closed	
4	Edit header <ul style="list-style-type: none">Modifier l'enteteModifier la couleur du nombre de post	Closed	

Planning

Drag and drop work items to include them in a sprint.

devops-213346 Team Backlog

Iteration 1
Planned Effort: -
6 15
Current

Iteration 2
No work scheduled yet

Iteration 3
No work scheduled yet

+ New Sprint

devops-213346 Team

No iteration dates
Set dates

Taskboard Backlog Capacity Analytics

+ New Work Item

Column Options

...

Iteration 1

...

Order	Title	State	Ass
	Création et configuration du fichier dockerfile	Closed	
	Configuration du fichier .gitignore	Closed	
2	Création d'un projet nodejs <ul style="list-style-type: none">Initialisation du projetCréation du fichier dockerfileCréation d'un fichier .dockerignoreAjout du fichier serveur + CR de post	Closed	
3	Post <ul style="list-style-type: none">Ajout d'un postListe des postesAjout du fichier docker-compose	Closed	
4	Edit header <ul style="list-style-type: none">Modifier l'enteteModifier la couleur du nombre de postAjout du readme.md	Closed	
5	frontend post	Closed	
+ 6	backend post <ul style="list-style-type: none">Création des tests unitaires du controleur post	Closed	

Planning

Drag and drop work items to include them in a sprint.

devops-213346 Team Backlog

Iteration 1
Planned Effort: -
6 15
Current

Iteration 2
No work scheduled yet

Iteration 3
No work scheduled yet

+ New Sprint

PROJET DE DEVOPS

devops-213346 Team

No iteration dates [Set dates](#)

Taskboard Backlog Capacity Analytics [+ New Work Item](#) [Column Options](#) Iteration 1 Person: All

Expand all New Active Resolved Closed

- initialisation de l'application
- Création d'un projet nodejs
- Post
- Edit header
- frontend post
- backend post

Taskboard Backlog Capacity Analytics [+ New Work Item](#) [Column Options](#) Iteration 1 Person: All

Collapse all New Active Resolved Closed

- Création d'un projet nodejs
 - 13 Post
 - Danick TAKAM
 - State
- Edit header
- frontend post
- 24 backend post
 - Danick TAKAM
 - State

- 14 Ajout d'un post
 - Danick TAKAM
 - State
- 15 Liste des postes
 - Danick TAKAM
 - State
- 16 Ajout du fichier docker-compose
 - Danick TAKAM
 - State
- 25 Création des tests unitaires du contrôleur post
 - Danick TAKAM
 - State

- Rapport des commits, pushes et tâches développées

devops-213346 Private [Invite](#)

About this project

Like

Projet de spécialisation devops 2020

Languages

Project stats

Last 7 days

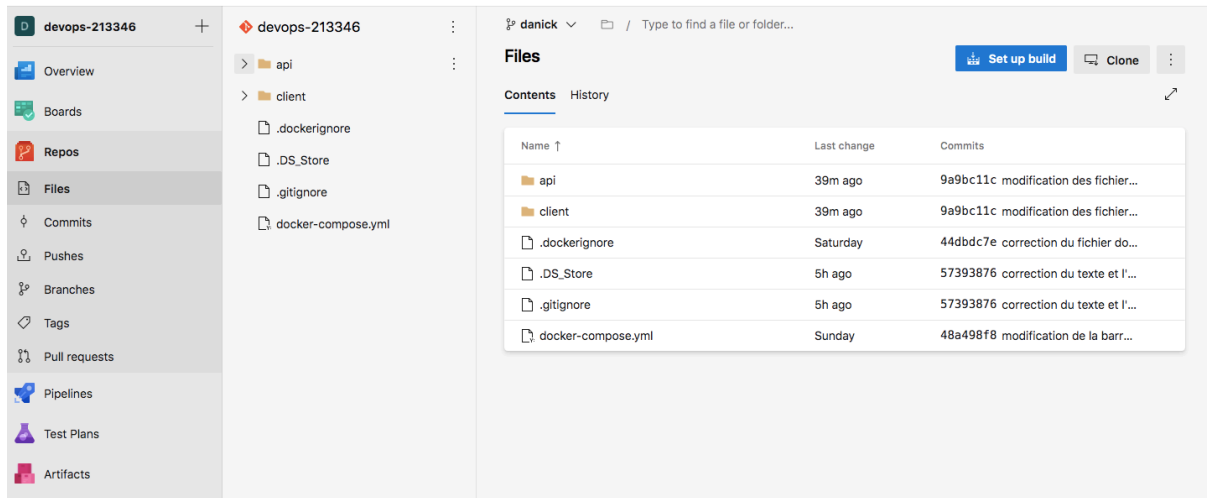
Boards

25 Work items created 25 Work items completed

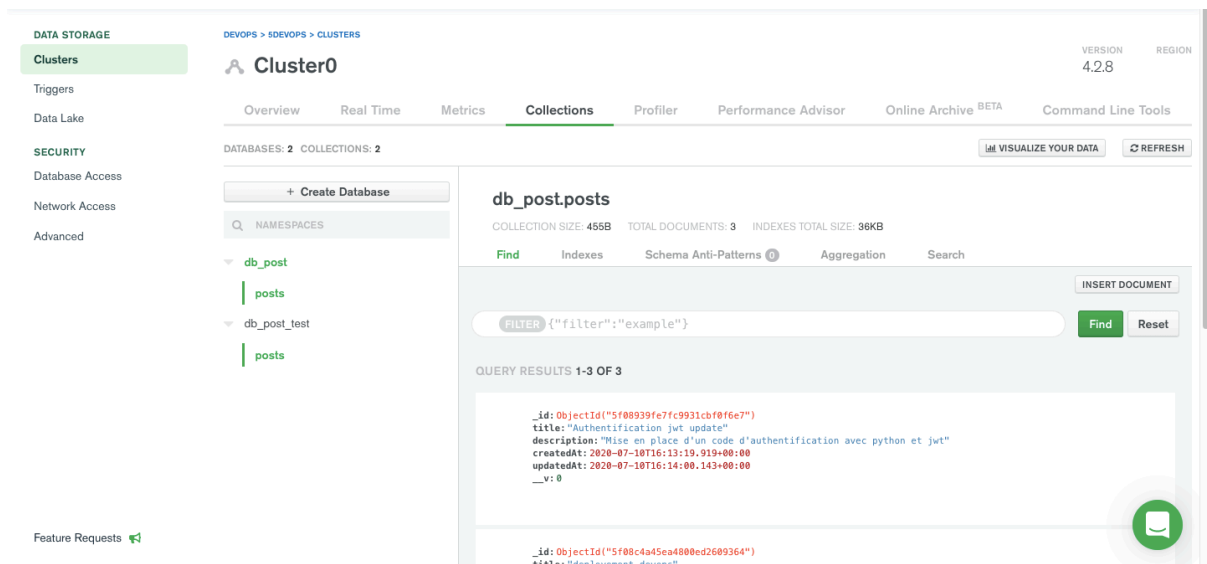
Repos

21 Pull requests opened 127 Commits by 1 authors

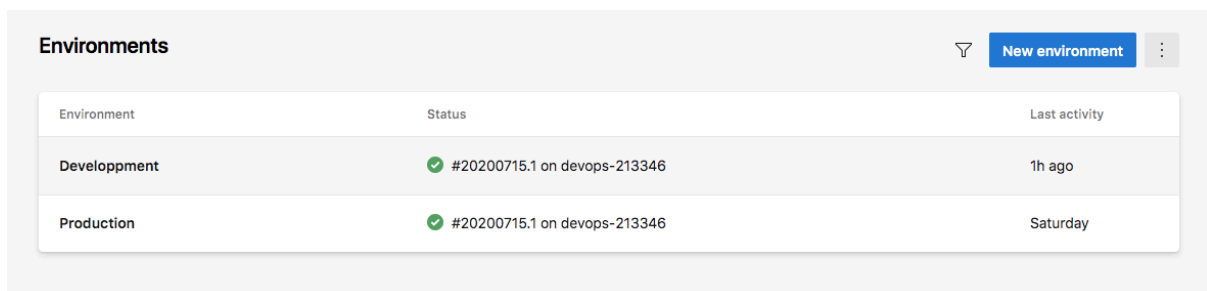
- Structure du projet



- Base de données (Mongo db)



- Environnement



- Test unitaires

Backend

Unit test backend

1 Starting: Unit test backend

2 =====

3 Task : Command Line

4 Description : Run a command line script using Bash on Linux and macOS and cmd.exe on Windows

5 Version : 2.164.2

6 Author : Microsoft Corporation

7 Help : <https://docs.microsoft.com/azure/devops/pipelines/tasks/utility/command-line>

8 =====

9 Generating script.

10 Script contents:

11 npm test

12 ===== Starting Command Output =====

13 /bin/bash --noprofile --norc /home/vsts/work/_temp/7b144b0b-65cf-426b-8681-d0f54b57adb5.sh

14

15 > api@1.0.0 test /home/vsts/work/_temp/7b144b0b-65cf-426b-8681-d0f54b57adb5.sh

16 > jest --runInBand --detectOpenHandles --silent --forceExit ./test

17

18 (node:3309) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future release. Please use the MongoClient class.

19 (node:3309) DeprecationWarning: mongoose: 'findOneAndUpdate()' and 'findOneAndDelete()' without the 'useFindAndModify' option is deprecated.

20 PASS tests/posts.test.js

21

22 Test Suites: 1 passed, 1 total

23 Tests: 5 passed, 5 total

24 Snapshots: 0 total

25 Time: 1.907 s

26

27 Finishing: Unit test backend

Frontend

Unit test frontend

28 Compiling @angular/platform-browser/animations : es2015 as esm2015

29 Compiling @angular/core/testing : es2015 as esm2015

30 Compiling @angular/platform-browser-dynamic : es2015 as esm2015

31 Compiling @angular/platform-browser/testing : es2015 as esm2015

32 Compiling @angular/compiler/testing : es2015 as esm2015

33 Compiling @angular/platform-browser-dynamic/testing : es2015 as esm2015

34 Compiling @angular/common/testing : es2015 as esm2015

35 Compiling @angular/router/testing : es2015 as esm2015

36 Compiling @angular/router/testing : es2015 as esm2015

37 15 07 2020 08:36:47.893:INFO [karma-server]: Karma v5.0.9 server started at http://0.0.0.0:9876/

38 15 07 2020 08:36:47.896:INFO [launcher]: Launching browsers ChromeHeadless with concurrency unlimited

39 15 07 2020 08:36:47.903:INFO [launcher]: Starting browser ChromeHeadless

40 15 07 2020 08:36:51.886:INFO [Chrome Headless 83.0.4103.0 (Linux x86_64)]: Connected on socket _F5eRtDBUanw...

41 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 0 of 4 SUCCESS (0 secs / 0 secs)

42 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 1 of 4 SUCCESS (0 secs / 0.015 secs)

43 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 2 of 4 SUCCESS (0 secs / 0.015 secs)

44 ERROR: 'Can't bind to 'ngModel' since it isn't a known property of 'input'.'

45 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 2 of 4 SUCCESS (0 secs / 0.015 secs)

46 ERROR: 'Can't bind to 'ngModel' since it isn't a known property of 'input'.'

47 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 2 of 4 SUCCESS (0 secs / 0.015 secs)

48 ERROR: 'Can't bind to 'ngModel' since it isn't a known property of 'input'.'

49 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 2 of 4 SUCCESS (0 secs / 0.015 secs)

50 ERROR: 'Can't bind to 'ngModel' since it isn't a known property of 'input'.'

51 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 2 of 4 SUCCESS (0 secs / 0.015 secs)

52 ERROR: 'Can't bind to 'ngModel' since it isn't a known property of 'input'.'

53 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 3 of 4 SUCCESS (0 secs / 0.076 secs)

54 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 4 of 4 SUCCESS (0 secs / 0.097 secs)

55 Chrome Headless 83.0.4103.0 (Linux x86_64): Executed 4 of 4 SUCCESS (0.18 secs / 0.097 secs)

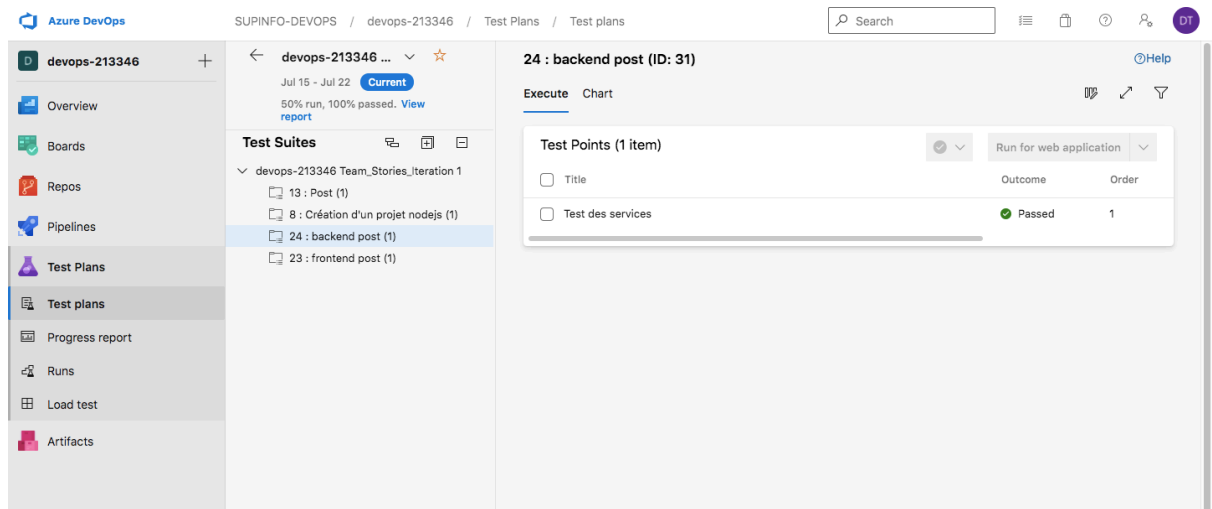
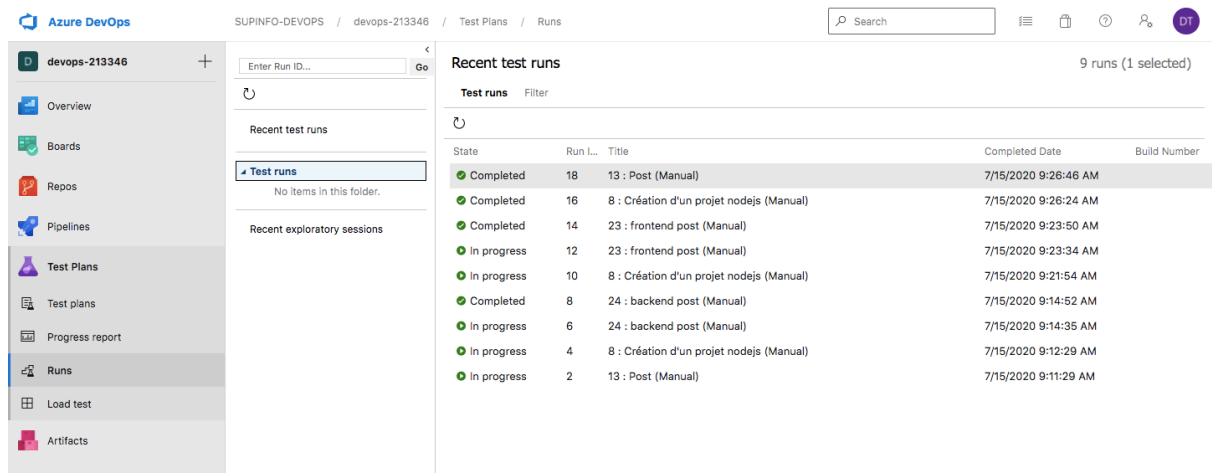
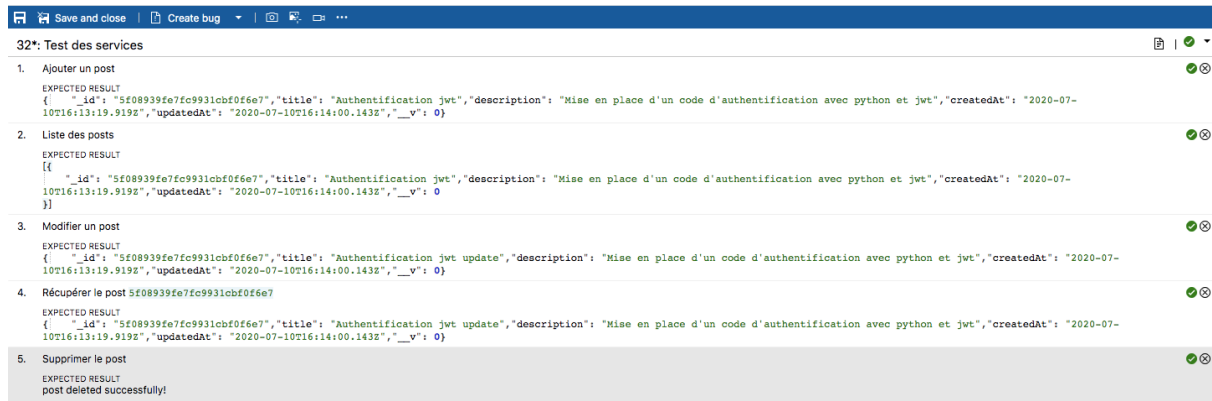
56 TOTAL: 4 SUCCESS

57 TOTAL: 4 SUCCESS

58 TOTAL: 4 SUCCESS

59

• Test d'intégration



V. Mise en place de la structure

1. Outils et technologies

- Le Framework JavaScript angular pour la réalisation du projet

- L'api nodejs pour le backend
- visual studio code pour le développement
- Git pour le versionning de l'application
- Azuredevops pour la planification et la gestion du projet
- Docker pour la gestion des images
- Mongo pour la base de données
- Jasmine pour les tests unitaire sur le frontend
- Jest pour les tests unitaires sur le backend
- Npm pour la gestion des dépendances
- Pm2 pour la gestion de nodejs

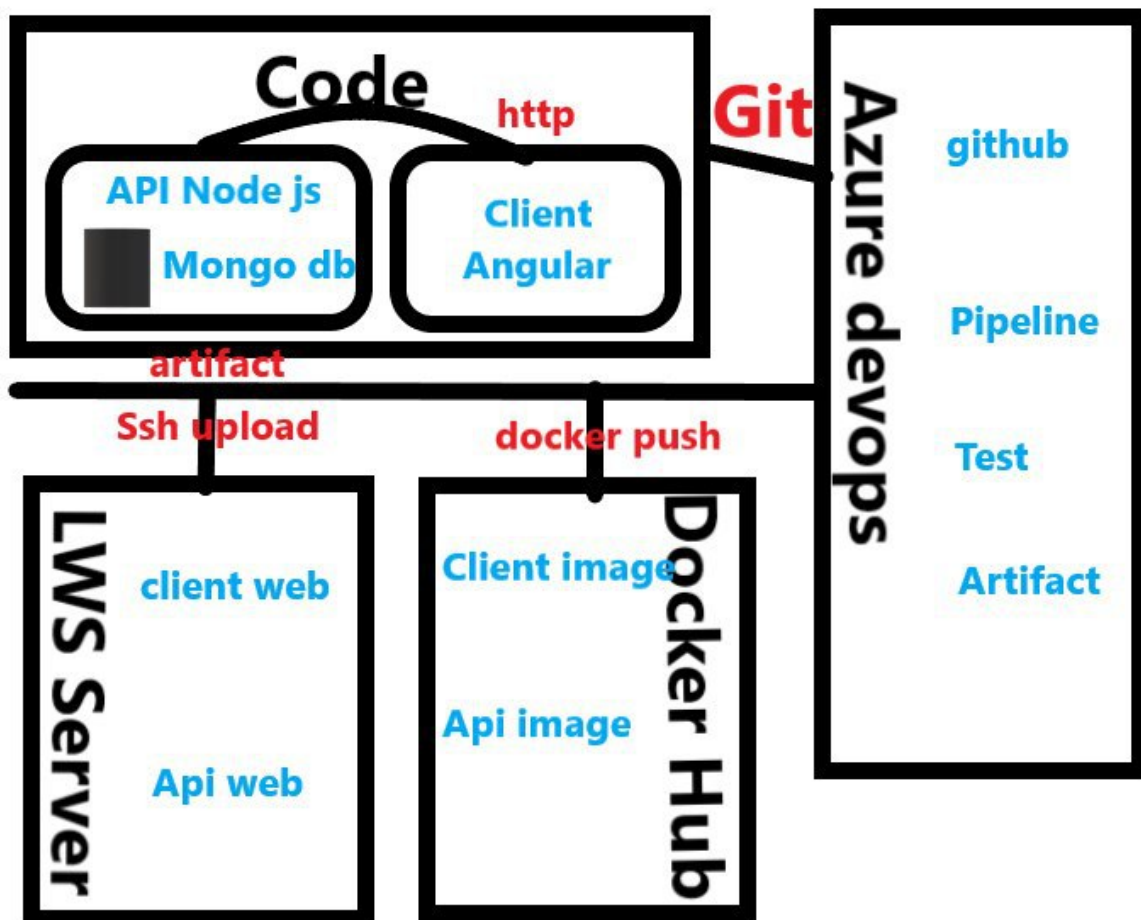


2. Comment ça marche

- Le code en local doit être envoyé en permanence à azuredevops après chaque changement

- Azuredevops se chargera d'exécuter toutes les intructions disponible dans Dockerfile (installation des dépendances, compilation du projet, création d'une nouvelle image docker basé sur le code, test unitaire)
- Docker image task du pupile Azuredevops enverra par la suite la nouvelle image générée sur le registre docker
- La tâche **publishcodecoverageresult** d'azure pipeline permettra de créer une couverture de test unitaire
- Un artifact sera crée en se basant sur le code source de l'api et le serveur
- Un autre artifiat est créé en utilisant uniquement les éléments à déployé. L'artificat actuelle contient le zip du build du frontend et le zip du backend
- Azuredevops se chargera par la suite de déployé l'artificat crée sur lws

Cette figure illsture le mécanisme clair du fonctionnement sans toute fois entré en détail.



Voici. Le fichier yaml qui montre chaque étape.

- Etape 1 : build et déploiement sur docker hub

```
#Déclancher l'événement sur la branche master automatiquement après
chaque
# push ou pull request
trigger:
- master
#Choisir la ressource: pour notre cas, la ressource se trouve sur notre
serveur
resources:
- repo: self
#Les variables à utiliser : ici le tag qui correspond à l'id courant du
build
variables:
  tag: '$(Build.BuildId)'

#Etape 1: Build et déployer les conteneur docker en production
# Noter que j'ai crée un service docker hub 'devopsdanick' en
renseignant mes informations
# d'authentification. Ceci dans l'optique de ne pas mettre mes
informations
# à la volé. Je pouvais aussi décidé de créer des variables
stages:
- stage: Build_docker_containers
  jobs:
  - job: Build
    pool:
      vmImage: 'ubuntu-latest'
    continueOnError: true
    steps:
      #Build et déploiement du frontend sur docker register
      # J'utilise dockerfile se trouvant dans le repertoire client.
      # Noter que je pouvais utiliser aussi docker compose.
      - task: Docker@2
        inputs:
          containerRegistry: 'devopsdanick'
          repository: 'strategie/devops-frontend'
          command: 'buildAndPush'
          Dockerfile: '$(Build.SourcesDirectory)/client/Dockerfile'
          displayName: 'Build and publish frontend'
```



```
# Build et déploiement du serveur sur docker
- task: Docker@2
  inputs:
    containerRegistry: 'devopsdanick'
    repository: 'strategie/devops-backend'
    command: 'buildAndPush'
    Dockerfile: '${Build.SourcesDirectory}/api/Dockerfile'
    displayName: 'Build and publish backend'
```

- Etape 2 : Installation des dépendances, tests unitaires, vérification de la syntaxe création d'un artefact

```
# Création d'un livrable intermédiaire en me basant sur mon code
source.
# J'e pouvais bien utiliser mes images docker ici. Mais cela sera
plus loud et
# prendre plus de temps. Les images docker pourront me servir plutart
si je
# prevoir hoster plusieurs versions de l'application et laisser les
utilisateurs
# choisir de migrer vers la version la plus recente ou pas.
- stage: 'Build_Artifact'
  jobs:
    - job: Artifact
      pool:
        vmImage: 'ubuntu-latest'
        continueOnError: true
      steps:
        - task: NodeTool@0
          inputs:
            versionSpec: '10.16.3'
            displayName: 'Install Node.js'

        - script: |
            npm install -g @angular/cli@10.0.1
            displayName: 'Install angular cli'

        - task: Npm@1
          displayName: 'Install dependancies frontend'
          inputs:
            workingDir: '${Build.SourcesDirectory}/client'
            command: 'install'
        - task: Npm@1
```

```

    displayName: 'Install dependancies backend'
    inputs:
      workingDir: '$(Build.SourcesDirectory)/api'
      command: 'install'

# Test du frontend
- script: |
    npm run test
    displayName: 'Unit test frontend'
    workingDirectory: '$(Build.SourcesDirectory)/client'

# Test du backend
- script: |
    npm test
    displayName: 'Unit test backend'
    workingDirectory: '$(Build.SourcesDirectory)/api'

- script: |
    npm run lint
    displayName: 'Code Analysis'
    workingDirectory: '$(Build.SourcesDirectory)/client'

- script: |
    npm run build --prod --output-path=dist
    displayName: 'Build project'
    workingDirectory: '$(Build.SourcesDirectory)/client'

- task: PublishCodeCoverageResults@1
  condition: succeededOrFailed()
  inputs:
    pathToSources: '$(Build.SourcesDirectory)/client'
    codeCoverageTool: 'Cobertura'
    summaryFileLocation:
      '$(Build.SourcesDirectory)/client/coverage/cobertura-coverage.xml'
    reportDirectory:
      '$(Build.SourcesDirectory)/client/coverage'
    displayName: 'Publish Code Coverage Results'

- task: PublishPipelineArtifact@1
  inputs:
    targetPath: '$(Pipeline.Workspace)'
    artifact: 'docker-compose'
    publishLocation: 'pipeline'
    displayName: 'Publish Artifacts'

```

- Etape 3 : Zip des fichiers important et création d'un artifact pour le pré-déploiement

```
# Prédéploiement: l'idée est de créer un artifact qui doit être
# facilement sur le serveur de production.
# Je vais ziper les fichiers front et le backend.
- stage: 'before_deploy_to_production'
  jobs:
  - deployment: Production
    pool:
      vmImage: 'ubuntu-latest'
    environment: 'Production'
    strategy:
      runOnce:
        deploy:
          steps:
            - task: ArchiveFiles@2
              inputs:
                rootFolderOrFile: '$(Pipeline.Workspace)/docker-
compose/s/api'
                includeRootFolder: true
                archiveType: 'zip'
                archiveFile: '$(Pipeline.Workspace)/docker-compose-
zip/apizip/api.zip'
                replaceExistingArchive: true
            - task: ArchiveFiles@2
              inputs:
                rootFolderOrFile: '$(Pipeline.Workspace)/docker-
compose/s/client/dist'
                includeRootFolder: true
                archiveType: 'zip'
                archiveFile: '$(Pipeline.Workspace)/docker-compose-
zip/clientzip/client.zip'
                replaceExistingArchive: true
```

```

- task: PublishPipelineArtifact@1
  inputs:
    targetPath: '$(Pipeline.Workspace)/docker-compose-zip/'
    artifact: 'docker-compose-zip'
    publishLocation: 'pipeline'
    displayName: 'Publish Artifacts'

```

- Etape 4 : Copie des fichiers zip sur le serveur

Déployer les archives sur le serveur de production.
 # J'ai crée une connexion ssh 'devopsssh' sur mon serveur distant
 lws.

Je ferai dans cette étape une copie des fichiers en ssh sur le
 serveur de

production. Je pouvais aussi utiliser le service ftp sur cette
 partie.

```

- stage: 'Deploy_to_production'
  jobs:
    - deployment: Production
      pool:
        vmImage: 'ubuntu-latest'
      environment: 'Production'
      strategy:
        runOnce:
          deploy:
            steps:
              - task: CopyFilesOverSSH@0
                inputs:
                  sshEndpoint: 'devopsssh'
                  sourceFolder: '$(Pipeline.Workspace)/docker-compose-
zip/apizip'
                  contents: '**'
                  #contents: |
                  #  docker-compose.yml
                  #  .env
                  targetFolder: '/var/www/clients/client0/web37/upload'
                  #cleanTargetFolder: true
                  readyTimeout: '20000'
                  overwrite: true
                  displayName: 'Download server on server'

              - task: CopyFilesOverSSH@0

```

```

      inputs:
        sshEndpoint: 'devopsssh'
        sourceFolder: '${Pipeline.Workspace}/docker-compose-
zip/clientzip'
        contents: '**'
        #contents: |
        #  docker-compose.yaml
        #  .env
        targetFolder: '/var/www/clients/client0/web36/upload'
        cleanTargetFolder: true
        #overwrite: true
        readyTimeout: '20000'

      displayName: 'Download client on server'

```

- Etape 5 : Normalisation des fichiers sur le serveur

```

#Normlization des fichiers sur le serveur de production.
# Cette partie consiste à dézipper les fichiers et les copier dans les
repertoires
# spécifiques puis redemarrer les services et supprimer les dossiers
inutiles.
- stage: 'Nomalize_file_in_production'
  jobs:
    - deployment: Production
      pool:
        vmImage: 'ubuntu-latest'
      environment: 'Production'
      strategy:
        runOnce:
          deploy:
            steps:
              - task: SSH@0
                inputs:
                  sshEndpoint: 'devopsssh'
                  runOptions: 'inline'
                  inline: |
                    rm -rf /var/www/clients/client0/web36/web/*
                    unzip
/var/www/clients/client0/web36/upload/client.zip -d
/var/www/clients/client0/web36/web/

```

```

        mv -f /var/www/clients/client0/web36/web/dist/*
/var/www/clients/client0/web36/web/
        rm -rf /var/www/clients/client0/web36/web/dist/
    displayName: 'Config client'
-   task: SSH@0
    inputs:
        sshEndpoint: 'devopsssh'
        runOptions: 'inline'
        inline: |
            pm2 stop danick-devops
            rm -rf /var/www/clients/client0/web37/web/config
/var/www/clients/client0/web37/web/app
/var/www/clients/client0/web37/web/tests
/var/www/clients/client0/web37/web/node_modules
            unzip /var/www/clients/client0/web37/upload/api.zip -
d /var/www/clients/client0/web37/web/
            mv -u /var/www/clients/client0/web37/web/api/*
/var/www/clients/client0/web37/web/
            rm -rf /var/www/clients/client0/web37/web/api/
            pm2 restart danick-devops
    displayName: 'Config server'

```

Les fichiers docker.

- Fichier dockerfile sur le backend

```

#Installation de nojs
FROM node:latest
#Changer le repertoire racine
WORKDIR /var/api
#Copier le code source vers le repertoire /var/api
COPY . .
#Installer npm
RUN npm install

###
# Test unitaires
###
# Exécuter les tests en désactivant le mode d'écoute
RUN npm test

#Exposer l'application sur le port 3010
EXPOSE 3010
#Exécuté npm run dev pour demarrer le projet en local
CMD ["npm", "run", "dev"]

```

- Fichier dockerfile sur le frontend

```
#####
## Fichier docker file
#####

# Image de base
FROM node:latest as node

# Installer chrome pour les tests
#RUN wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub | apt-key
add -
#RUN sh -c 'echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable
main" >> /etc/apt/sources.list.d/google.list'
#RUN apt-get update && apt-get install -yq google-chrome-stable

# Définir le repertoire de travail
WORKDIR /var/client

# copier les fichiers dans le dossier app
COPY . .

# Installer npm
RUN npm install

# Installer angular
RUN npm install -g @angular/cli@10.0.1

#Vérifier la syntaxe de l'application
#RUN npm run lint

###
# Test unitaires
###
# Exécuter les tests en désactivant le mode d'écoute
RUN npm run test

# Exécuter les tests end to end
#RUN npm run e2e --port 4202

# Générer les fichiers de mise en production dans le repertoire dist
RUN npm run build --prod --output-path=dist

#Exposer l'application
EXPOSE 4001

####
# Déployer l'application sur un serveur apache en développement
####
```

```
# server
#FROM httpd:2.4
#Copier l'artefact vers de l'environnement de bluid
#COPY --from=node /var/client/dist /usr/local/apache2/htdocs/
```

```
CMD [ "npm" , "start" ]
```

- Fichier docker-compose : Ce fichier facilite la gestions des deux projets en exécutant les fichiers dockerfile du backend et du frontend.

```
version: '3'
services:
  backend:
    #restart: always
    build:
      context: ./api
      dockerfile: Dockerfile
    ports:
      - "3010:3010"
    container_name: backend-devops
    volumes:
      - ./api:/var/api
      - /var/api/node_modules
  frontend:
    #restart: always
    build:
      context: ./client
      dockerfile: Dockerfile
    ports:
      - "4001:4001"
    container_name: frontend-devops
    volumes:
      - ./client:/var/client
      - /var/client/node_modules
```

Le vhost du backend sur le serveur de déploiement : j'ai automatiquement redirigé le http vers le https. J'ai ensuite fait une liaison avec le projet déployé en local sur le port 3010.

```
<VirtualHost *:80>
  ServerName api-danick-devops.datnek.be
  ServerAlias www.api-danick-devops.datnek.be
  ServerAdmin webmaster@api-danick-devops.datnek.be
  Redirect permanent / https://api-danick-devops.datnek.be/
</VirtualHost>

<VirtualHost *:443>
```



```

ServerName api-danick-devops.datnek.be
ServerAlias www.api-danick-devops.datnek.be
ServerAdmin webmaster@api-danick-devops.datnek.be

# nos paramètres de bases
Header always set Access-Control-Allow-Origin "https://danick-
devops.datnek.be/, http://danick-devops.datnek.be/"
Header always set Access-Control-Allow-Methods "GET, POST, OPTIONS, PUT, PATCH,
DELETE"
Header always set Access-Control-Max-Age 86400
Header always set Access-Control-Allow-Headers "Origin, X-Requested-With,
RewriteCond %{REQUEST_METHOD} OPTIONS
RewriteRule ^(.*)$ $1 [R=204,L]

ErrorLog /var/log/ispconfig/httpd/api-danick-devops.datnek.be/error.log

SSLEngine on
SSLProtocol All -SSLv2 -SSLv3

SSLHonorCipherOrder      on

SSLCertificateFile /var/www/clients/client0/web37/ssl/api-danick-
devops.datnek.be-le.crt
SSLCertificateKeyFile /var/www/clients/client0/web37/ssl/api-danick-
devops.datnek.be-le.key
SSLUseStapling on
SSLStaplingResponderTimeout 5
SSLStaplingReturnResponderErrors off

# on transmet ensuite les requêtes au backend sur le port local 3001
ProxyPass "/" "http://localhost:3001/"
</VirtualHost>

<IfModule mod_ssl.c>
    SSLStaplingCache shmcb:/var/run/ocsp(128000)
</IfModule>

```

Le fichier vhost est pratiquement le même à la seule différence qu'il faut ajouter le documentroot et le chemin racine du projet. Il faudra ensuite créer un DNS pour les deux sous domaines et aussi se rassurer que le nom de domaine parent soit disponible.

VI. Liens du projet et intégration

Pour avoir le projet en local,

- git clone https://SUPINFO-DEVOPS@dev.azure.com/SUPINFO-DEVOPS/devops-213346/_git/devops-213346
- username: 213346

- mot de passe : ktld22jckzlfjq473z6wacun4sw6wf24shjooc6lpepsdhga4m4q