

Projet - Programmer le Cloud

Ayrwan GUILLERMO - ENSTA Bretagne

Lien vers le répo : (https://github.com/Ays-s/Projet_Cloud)

TD 1 : une application Node.js

2

Le fichier `package.json` est le fichier de gestion du package qui gère les dépendances, les scripts de run et le point d'entrer du package.

Le fichier `package-lock.json` permet d'indexer les packets installé avec la version précise de ceux-ci, afin d'être sûr d'avoir les mêmes versions de modules.

3

Cela ajoute une dépendance à `systeminformation`. `devDependencies` permet d'avoir des dépendances uniquement sur environnement de développement et non celui de production.

5

On utilise ce formalisme pour pouvoir avoir un point d'entrer spécifique à l'API puis à chaque version de celle-ci.

6

On écrit ce jeu de test pour pouvoir mettre en place l'intégration continue. Afin que celle-ci soit faite il faut qu'un test soit réussi.

TD2 : conteneurisation avec Docker

4

Le flag `-p` permet une redirection de port entre le docker et la machine qui exécute.

Le flag `-m` permet de définir la mémoire allouée au docker.

Le flag `--cpus` indique le nombre de cpu alloué à l'exécution du docker.

5

Layers		
Cmp	Size	Command
	5.6 MB	FROM 183f28bea52d7e2
	0 B	#(nop) WORKDIR /app
	16 MB	apk add git
	48 MB	apk add --update nodejs>=16 npm>=8
	548 kB	git clone https://github.com/Ays-s/Projet_Cloud.git
	156 MB	npm install
	24 kB	npm run build
Layer Details		
Tags: (unavailable)		
Id: bdc685a1af33ba82108707f756331261ace8fe15b290855f97441698f6bb8bf		
Digest: sha256:5a84759ee83d913ac4d676ff373c59ae882701f9b1b3924bc1dc9d7d1813d787		
Command: npm run build		
Image Details		
Image name: sysinfo-api:0.0.1		
Total Image size: 226 MB		
Potential wasted space: 1.4 MB		
Image efficiency score: 99 %		
Count	Total Space	Path
2	811 kB	/app/Projet_Cloud/package-lock.json
2	417 kB	/etc/ssl/certs/ca-certificates.crt
3	173 kB	/lib/apk/db/installed
3	39 kB	/lib/apk/db/scripts.tar
3	500 B	/lib/apk/db/triggers
3	196 B	/etc/apk/world
3	0 B	/lib/apk/db/lock
3	0 B	/var/cache/misc

La taille du container est très grande : Total Image size: 226 MB.

Elle contient des utilitaires ou des parties de codes utiles uniquement pour le build. On pourrait réduire la taille de l'image en enlevant ces utilitaires.

6

Layers		
Cmp	Size	Command
	5.6 MB	FROM 183f28bea52d7e2
	0 B	#(nop) WORKDIR /app
	5.7 MB	apk update && apk upgrade
	43 MB	apk add --update nodejs>=16
	4.7 kB	addgroup -S node && adduser -S node -G node
	646 kB	#(nop) COPY --chown=node:nodedir:31eddc618f447a8c4769b57f8840c7edb5e3bfc0eed6107f43c9f13d
	25 kB	#(nop) COPY --chown=node:nodedir:93fb15d8157cdba92fdb5cd86ab81e389f9646508273a9f030143126
Layer Details		
Tags: (unavailable)		
Id: 183f28bea52d7e2fc14055b6af34dd4860f8872a7e68e3d6a89221c80f0f4d20		
Digest: sha256:8d3ac3489996423f53d6087c81180006263b79f206d3fdec9e66f0e27ceb8759		
Command:		
#(nop) ADD file:9233f6f2237d79659a9521f7e390df217cec49f1a8aa3a12147bbca1956acdb9 in /		
Image Details		
Image name: sysinfo-api:0.0.2		
Total Image size: 55 MB		
Potential wasted space: 7.1 MB		
Image efficiency score: 93 %		
Count	Total Space	Path
2	5.2 MB	/lib/libcrypto.so.1.1
2	1.0 MB	/lib/libssl.so.1.1
3	620 kB	/etc/ssl/certs/ca-certificates.crt
3	56 kB	/lib/apk/db/installed
2	53 kB	/usr/lib/engines-1.1/padlock.so
2	46 kB	/usr/lib/engines-1.1/afalg.so
3	37 kB	/lib/apk/db/scripts.tar
2	28 kB	/usr/lib/engines-1.1/capi.so
2	22 kB	/etc/ssl1.1/openssl.cnf
2	22 kB	/etc/ssl1.1/openssl.cnf.dist
2	2.4 kB	/etc/passwd
2	1.4 kB	/etc/group
2	870 B	/etc/shadow
2	824 B	/etc/ssl1.1/ct_log_list.cnf.dist
3	364 B	/lib/apk/db/triggers
3	184 B	/etc/apk/world
2	0 B	/usr/lib/libcrypto.so.1.1
2	0 B	/usr/lib/libssl.so.1.1
2	0 B	/etc/ssl1.1/ct_log_list.cnf
2	0 B	/etc/ssl1.1/certs
2	0 B	/etc/ssl1.1/cert.pem
2	0 B	/var/cache/misc
2	0 B	/etc/ssl/cert.pem
3	0 B	/lib/apk/db/lock

La taille du container est beaucoup plus petite : Total Image size: 55 MB

Le delta est de 171MB soit une baisse de 75%.

8

La commande est :

```
docker login
docker pull ayss/sysinfo-api:0.0.2
```

TD3 : CI/CD avec GitHub

3

On test sont fonctionnement directement sir les `github actions` . On voit dans l'onglet `actions` le build de notre application et les erreurs s'il y en a.

4

Il aurait été plus simple de découper la partie server du reste de l'application. Les tests auraient été plus simples à réaliser.

TD4 : déploiement sur PaaS avec Heroku

3

Le *Process type* est le type d'application qui va être déployé. On utilise donc `web` car on déploie une application web.

4

Bien que l'application build, on ne peut pas accéder à son url. Le port est assigné de façon dynamique et une variable d'environnement l'indique. Il faut donc se servir de cette variable pour indiquer le port de notre application.

5

On peut l'associer au point 7 : Associations de ports.

6

Avec notre application on retrouve les informations de la machine.

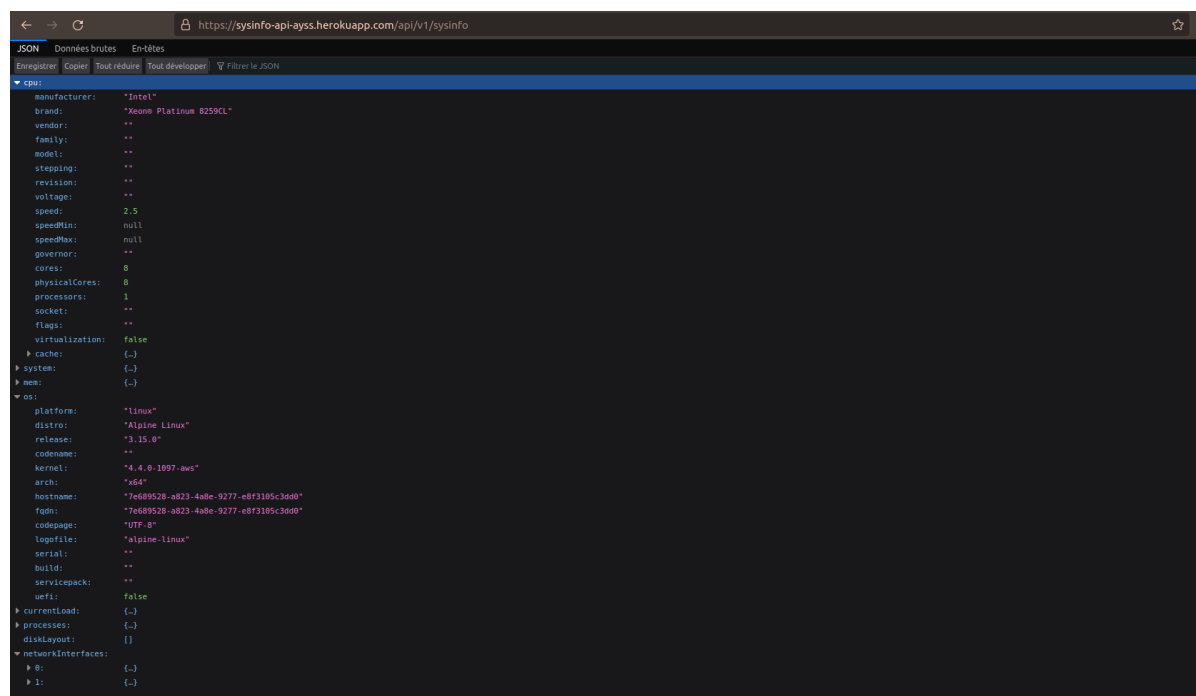
C'est une machine très puissante notamment avec un processeur Intel Xeon Platinum 8259CL.

On remarque que notre application n'est pas exécutée dans une VM car sinon seules les infos de la VM seraient visibles.

7

Le déploiement automatique est disponible à : (<https://dashboard.heroku.com/apps/sysinfo-api-ayss>)

Résultat final :



```
JSON  Données brutes  En-têtes
Enregistrer Copier Tout réduire Tout développer Filtre le JSON

▼ cpu:
  manufacturer: "Intel"
  brand: "Xeon Platinum 8259CL"
  vendor: ""
  family: ""
  model: ""
  stepping: ""
  revision: ""
  voltage: ""
  speed: 2.5
  speedMin: null
  speedMax: null
  governor: ""
  cores: 8
  physicalCores: 8
  processors: 1
  socket: ""
  flags: ""
  virtualization: false
  ▶ cache: (-)
  ▶ system: (-)
  ▶ mem: (-)
▼ os:
  platform: "linux"
  distro: "Alpine Linux"
  release: "3.15.0"
  codename: ""
  kernel: "4.4.0-1097-aws"
  arch: "x86_64"
  hostname: "7ed89528-a823-4a8e-9277-e8f3105c3d08"
  fqdn: "7ed89528-a823-4a8e-9277-e8f3105c3d08"
  codepage: "UTF-8"
  logfile: "alpine-linux"
  serial: ""
  build: ""
  servicepack: ""
  uefi: false
  ▶ currentLoad: (-)
  ▶ processes: (-)
  ▶ diskLayout: (-)
▼ networkInterfaces:
  ▶ 0: (-)
  ▶ 1: (-)
```

