



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**

Membre de 
HONORIS UNITED UNIVERSITIES

Rapport Projet : Architecture des composants d'entreprise

Réalisé par : YAHYA ABIDA\MOHAMED TAHA EL ZENJARI\ ZAKARIA RAISSI



Introduction du Projet

Bienvenue dans notre application Web innovante de location de voitures, une solution de pointe qui intègre de manière transparente la puissance de Spring Boot, Angular et RabbitMQ pour offrir une expérience utilisateur exceptionnelle dans l'industrie de la location de voitures.

Objectif :

L'objectif principal de notre application est de fournir aux utilisateurs une plateforme conviviale pour naviguer, réserver et gérer leurs locations de voitures de manière pratique. Que les utilisateurs souhaitent activer leur compte, confirmer des réservations ou effectuer des paiements sécurisés, notre application garantit un processus fluide et sécurisé.

Objectifs :

1. **Onboarding Efficace des Utilisateurs** : Rationaliser le processus d'intégration des utilisateurs grâce à l'activation du compte via e-mail, renforçant ainsi la sécurité et la confiance des utilisateurs.
2. **Confirmation de Réservation Sans Faille** : Permettre aux utilisateurs de confirmer facilement leurs réservations, de recevoir des informations pertinentes et de procéder à des transactions de paiement sécurisées.
3. **Intégration de Microservices** : Tirer parti d'une architecture de microservices, améliorant la scalabilité, la maintenabilité et la robustesse globale du système.
4. **Messagerie Fiable avec RabbitMQ** : Utiliser RabbitMQ pour des services de messagerie efficaces, garantissant une livraison rapide des liens d'activation de compte et des confirmations de réservation par e-mail aux utilisateurs.
5. **Front-End Réactif avec Angular** : Mettre en œuvre une interface utilisateur dynamique et réactive à l'aide d'Angular, offrant une expérience moderne et engageante pour les utilisateurs sur tous les appareils.

Caractéristiques Clés :

- **Activation de Compte Utilisateur** : Les utilisateurs reçoivent des liens d'activation par e-mail, assurant une création de compte sécurisée et vérifiée.
- **Confirmation de Réservation** : Confirmez instantanément les réservations et guidez les utilisateurs vers un processus de paiement sans problème.
- **Architecture de Microservices** : Améliorez la scalabilité et la maintenabilité grâce à la mise en œuvre de microservices.

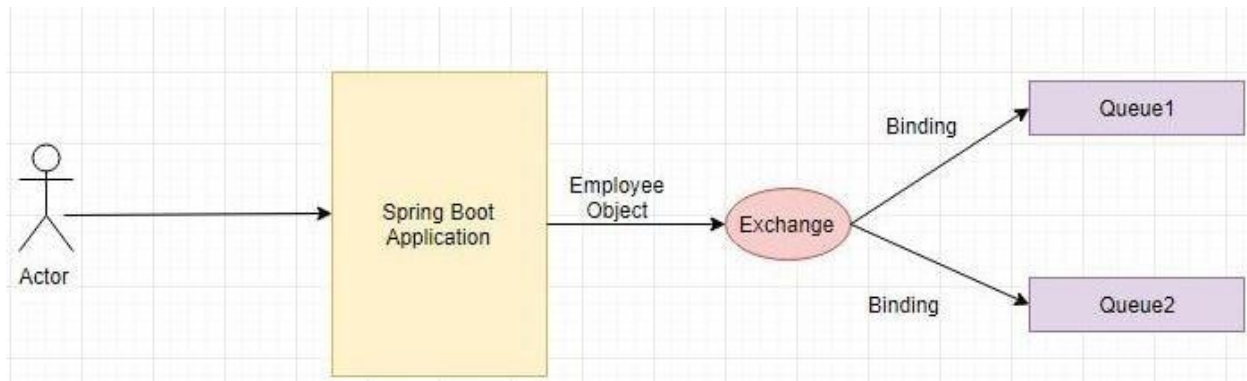
- **File d'Attente de Messages avec RabbitMQ** : Gérez efficacement les services de messagerie, assurant une communication fiable entre les différents composants.
- **Front-End Dynamique avec Angular** : Fournissez une interface utilisateur dynamique et réactive grâce à Angular, améliorant l'engagement des utilisateurs.

Notre application Web de location de voitures est conçue avec la commodité de l'utilisateur et l'efficacité du système à l'esprit. Explorez les fonctionnalités, profitez des avantages et embarquez pour un voyage de location de voitures sans tracas avec nous !

2. Architecture Microservices

Architecture

L'architecture microservices adoptée dans notre projet de location de voitures se caractérise par une approche modulaire et décentralisée. Plutôt que d'avoir une seule application monolithique, nous avons organisé le système en plusieurs microservices autonomes, chacun dédié à des fonctions spécifiques.



Principales Caractéristiques de l'Architecture Microservices :

- **Indépendance des Services** : Chaque microservice est une entité indépendante, déployable et évolutive de manière autonome. Cette indépendance favorise une gestion plus souple et efficace des différentes parties du système.
- **Scalabilité Sélective** : La scalabilité est appliquée de manière sélective à chaque microservice en fonction de ses besoins spécifiques. Par exemple, le service de gestion des réservations peut être mis à l'échelle de manière indépendante du service de gestion des utilisateurs.
- **Déploiement Indépendant** : Les microservices peuvent être déployés indépendamment les uns des autres, facilitant ainsi les mises à jour continues sans perturber l'ensemble de l'application. Cela permet également d'assurer une disponibilité continue du service.



- **Technologies Variées** : Chaque microservice peut être développé en utilisant les technologies les mieux adaptées à sa fonctionnalité. Cela favorise l'utilisation des outils et frameworks les plus appropriés pour résoudre des problèmes spécifiques.

Description des Services

Chaque microservice dans notre architecture joue un rôle crucial dans la réalisation des fonctionnalités globales de l'application de location de voitures. Voici une brève description de certains des principaux services :

1. **Service de Gestion des Utilisateurs** : Responsable de la gestion des comptes utilisateurs, de l'authentification et de l'autorisation. Gère également le processus d'activation du compte par e-mail.
2. **Service de Gestion des Réservations** : Gère toutes les opérations liées aux réservations de voitures, de la confirmation à la gestion des disponibilités et des paiements.
3. **Service de Messagerie avec RabbitMQ** : Facilite la communication asynchrone entre les différents microservices en utilisant RabbitMQ. Gère l'envoi de messages liés à l'activation du compte et à la confirmation de réservation.

Mécanismes de Communication

Dans notre architecture microservices, la communication entre les services est essentielle pour assurer la cohérence et la coordination des actions. Nous utilisons plusieurs mécanismes de communication, dont les principaux sont :

- **Communication HTTP/REST** : Les microservices exposent des API RESTful pour permettre une communication synchrone entre eux. Cela facilite les interactions directes pour des opérations telles que la récupération d'informations utilisateur ou la gestion de réservations.
- **Messagerie Asynchrone avec RabbitMQ** : Pour des communications asynchrones, tels que l'envoi d'e-mails d'activation de compte, nous utilisons RabbitMQ comme mécanisme de messagerie. Cela garantit une communication fiable même lorsque les services opèrent à des vitesses différentes.

Cette architecture favorise une communication souple et efficace entre les microservices, permettant une mise en œuvre modulaire et évolutive de l'ensemble du système de location de voitures.

3. Conception des Microservices

Approche de Conception pour Chaque Service

Chaque microservice dans notre architecture de location de voitures suit une approche de conception spécifique, alignée sur les principes de modularité, d'indépendance, et de maintenabilité. Voici une vue d'ensemble de l'approche de conception adoptée pour certains des principaux services :



1. Service de Gestion des Utilisateurs :

- **Modularité** : Le service de gestion des utilisateurs est conçu de manière modulaire pour faciliter l'ajout de nouvelles fonctionnalités liées aux utilisateurs. Chaque aspect, comme l'authentification, l'autorisation, et l'activation du compte, est traité de manière indépendante.
- **Sécurité** : La sécurité est une priorité, avec l'utilisation de protocoles standards tels que OAuth pour l'authentification. Les communications sensibles sont cryptées, et des mécanismes robustes sont mis en place pour prévenir les vulnérabilités.
- **Gestion des Événements** : Les événements liés aux utilisateurs, tels que l'activation de compte, déclenchent des messages asynchrones via RabbitMQ, assurant ainsi une réactivité rapide et une gestion décentralisée des tâches.

2. Service de Gestion des Réservations :

- **Indépendance de la Logique Métier** : La logique métier liée aux réservations, comme la disponibilité des véhicules et la gestion des paiements, est encapsulée de manière indépendante. Cela permet des modifications sans impacter d'autres parties du système.
- **Évolutivité** : La conception permet une évolutivité horizontale en cas de demande accrue. La gestion des réservations peut être étendue de manière indépendante pour répondre aux pics de trafic.
- **Communication avec d'Autres Services** : Les communications avec d'autres services, tels que le service de gestion des utilisateurs, sont gérées de manière asynchrone pour minimiser les dépendances et assurer une exécution fluide.

3. Service de Messagerie avec RabbitMQ :

- **Intégration de RabbitMQ** : Ce service est spécifiquement conçu pour gérer la communication asynchrone entre les microservices. Il garantit la fiabilité des messages et permet une distribution efficace des tâches.
- **Gestion des Erreurs** : Une attention particulière est portée à la gestion des erreurs pour assurer la robustesse du système. Les messages sont traités avec des mécanismes de reprise en cas d'échec.
- **Séparation des Responsabilités** : Le service de messagerie se concentre sur son rôle de gestion des communications, laissant aux autres microservices la responsabilité des actions spécifiques déclenchées par les messages.

Cette approche de conception pour chaque service vise à garantir une architecture modulaire, souple, et facilement évolutive. Chaque microservice est conçu pour répondre efficacement à ses responsabilités spécifiques tout en contribuant à l'ensemble cohérent de l'application de location de voitures.

4. Conteneurisation avec Docker

Dockerfile for Backend (Java/Spring Boot):

```
1 >> FROM openjdk:17
2
3 WORKDIR /App
4
5 COPY /target/RentaCar-0.0.1-SNAPSHOT.jar .
6
7 EXPOSE 8086
8
9 ENTRYPOINT ["java", "-jar", "RentaCar-0.0.1-SNAPSHOT.jar"]
```

```
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
<modelVersion>4.0.0</modelVersion>
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.0</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.Service</groupId>
<artifactId>RentaCar</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>RentaCar</name>
<description>RentaCar</description>
```

Avantages :

- Utilise une image de base légère (OpenJDK).
- Configure le répertoire de travail.
- Copie le fichier JAR de l'application et expose le port requis.

- Configure le point d'entrée pour exécuter l'application Spring Boot.

Dockerfile for Frontend (Angular):

DockerfileCopy code

```
rentacar\...\Dockerfile × application.properties environment.ts docker-compose.yml
1 # Stage 1: Angular build
2 FROM node:latest as build
3
4 WORKDIR /app
5
6 COPY package.json package-lock.json ./
7
8 RUN npm ci
9
10 COPY . .
11
12 RUN npm run build
13
14 # Stage 2: Serve with Nginx
15 FROM nginx:1.21.6
16
17 COPY --from=build /app/dist /usr/share/nginx/html
18
19 CMD ["nginx", "-g", "daemon off;"]
```

Avantages :

- Utilise des constructions multi-étapes pour une image finale plus petite.
- Configure la phase de construction avec Node pour la compilation de l'application Angular.
- Copie les fichiers nécessaires, installe les dépendances et compile l'application Angular.
- Utilise Nginx pour servir l'application Angular compilée dans l'image finale.



Docker Compose File:

```
version: "3.8"
> > services:
> > rabbitmq:
  image: rabbitmq:3.12.2-management-alpine
  container_name: 'rentacar-rabbitmq'
  command: ["bash", "-c", "chmod 600 /var/lib/rabbitmq/.erlang.cookie && rabbitmq-server"]
  ports:
    - 5672:5672
    - 15672:15672
  volumes:
    - ../docker-runtime-data/rabbitmq/data:/var/lib/rabbitmq/
    - ../docker-runtime-data/rabbitmq/log:/var/log/rabbitmq
  networks:
    - rentacar
  environment:
    - RABBITMQ_DEFAULT_USER=rentacar
    - RABBITMQ_DEFAULT_PASS=rentacar_pass
```

```
db:
  image: mysql:8.0
  command: --default-authentication-plugin=caching_sha2_password
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: Y@hya@bida123
    MYSQL_DATABASE: carrentams
  ports:
    - "3316:3306"
  volumes:
    - db_data:/var/lib/mysql
  networks:
    - rentacar

volumes:
  db_data:
```




Avantages :

- Définit des services pour RabbitMQ, MySQL, Backend et Frontend.
- Configure un réseau pour la communication entre les services.
- Assure la persistance des volumes pour les données MySQL.

Points Clés :

- L'utilisation de constructions multi-étapes minimise la taille de l'image finale.
- Chaque service est encapsulé dans son propre Dockerfile, assurant l'isolation.
- Docker Compose orchestre le déploiement et la communication entre les services.
- L'isolation réseau est mise en œuvre pour une sécurité renforcée.
- Les montages de volume garantissent la persistance des données pour MySQL.

Cette configuration Docker facilite le déploiement, la scalabilité et la gestion aisée de vos microservices de location de voitures.

5. CI/CD avec Jenkins

Etape 1 :

Dashboard > All >

Enter an item name

rapport

» Required field

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK branch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.



Etape 2 :

Jenkins Search (CTRL+K) abida yahya log out

Dashboard > car_microserv > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

General

Enabled ☒

Description

RENTAL CAR SITE

Plain text [Preview](#)

- ☐ Discard old builds
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the controller restarts
- ☒ GitHub project

Project url

Save Apply

☒ GitHub project

Project url

https://github.com/Yahyaab12/Car_microservice/

Advanced ▾

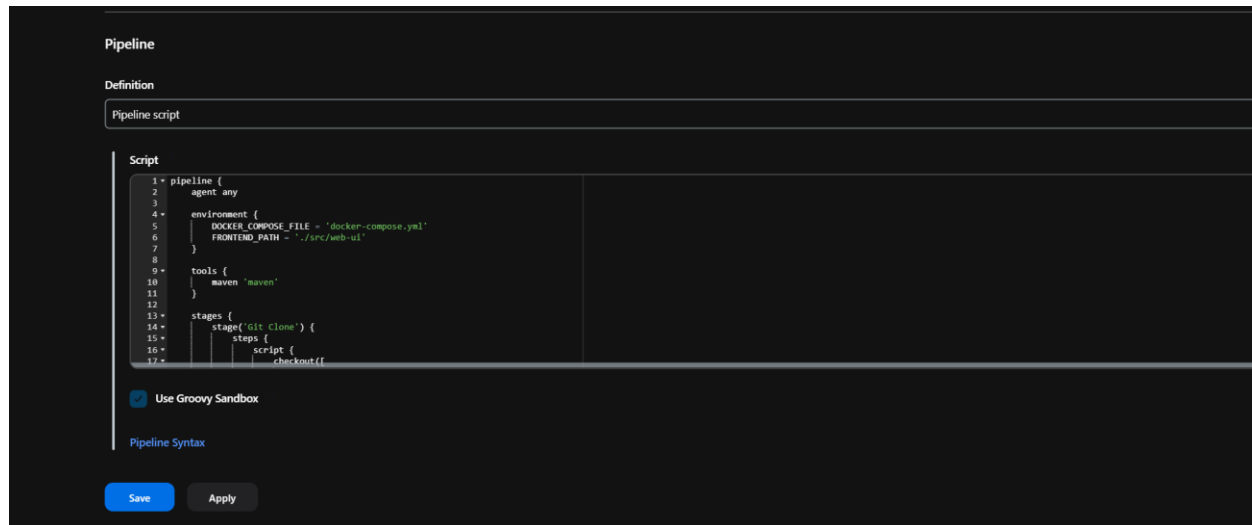
Build Triggers

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub Branches
- ☐ GitHub Pull Requests
- ☒ GitHub hook trigger for GITScm polling
- ☐ Poll SCM
- ☐ Quiet period
- ☐ Trigger builds remotely (e.g., from scripts)



ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR

Membre de
HONORIS UNITED UNIVERSITIES





**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**

Membre de 
HONORIS UNITED UNIVERSITIES

```
branches: [[name: 'main']],

userRemoteConfigs: [[url: 'https://github.com/Yahyaab12/Car_microservice']]

  })
}
}
}

stage('Build') {
  steps {
    script {
      echo 'Building Backend...'

      bat 'mvn clean install -Dmaven.test.skip=true'
    }
  }
}

stage('Build Frontend') {
  steps {
    echo 'Building Frontend...'

    dir(FRONTEND_PATH) {
      bat '''
        npm install
        npm run build
      '''
    }
  }
}
```

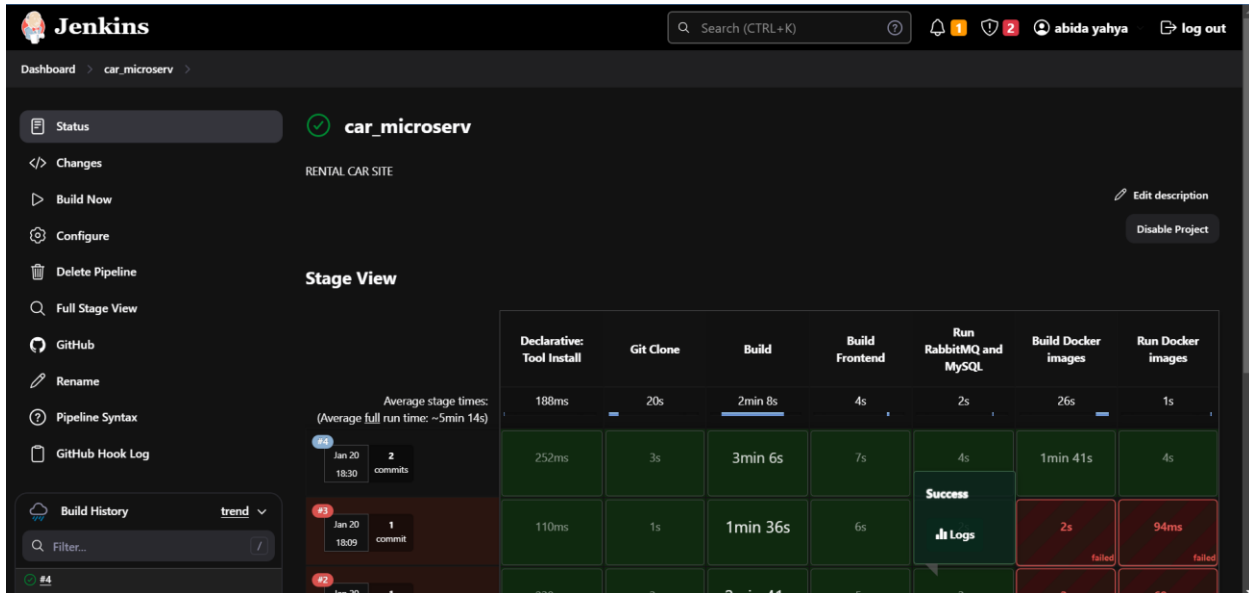


ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR

Membre de
HONORIS UNITED UNIVERSITIES

```
stage('Run RabbitMQ and MySQL') {  
    steps {  
        echo 'Running RabbitMQ and MySQL...'  
        bat 'docker-compose up -d rabbitmq db'  
    }  
}  
  
stage('Build Docker images') {  
    steps {  
        echo 'Building Docker images...'  
        bat 'docker build -t rentacar-backend .'  
        bat 'docker build -t rentacar-frontend ./src/web-ui'  
    }  
}  
  
stage('Run Docker images') {  
    steps {  
        echo 'Running Docker Compose...'  
        bat "docker-compose -f $DOCKER_COMPOSE_FILE up -d"  
    }  
}  
}
```

Etape 3 : application Build



Jenkins Dashboard > car_microserv

car_microserv RENTAL CAR SITE

Stage View

Average stage times: (Average full run time: ~5min 14s)

Stage	Declarative: Tool Install	Git Clone	Build	Build Frontend	Run RabbitMQ and MySQL	Build Docker images	Run Docker images
188ms	20s	2min 8s	4s	2s	26s	1s	
252ms	3s	3min 6s	7s	4s	1min 41s	4s	
110ms	1s	1min 36s	6s	Success	2s	94ms	
320ms	2s	2min 41s	5s	Logs	failed	failed	

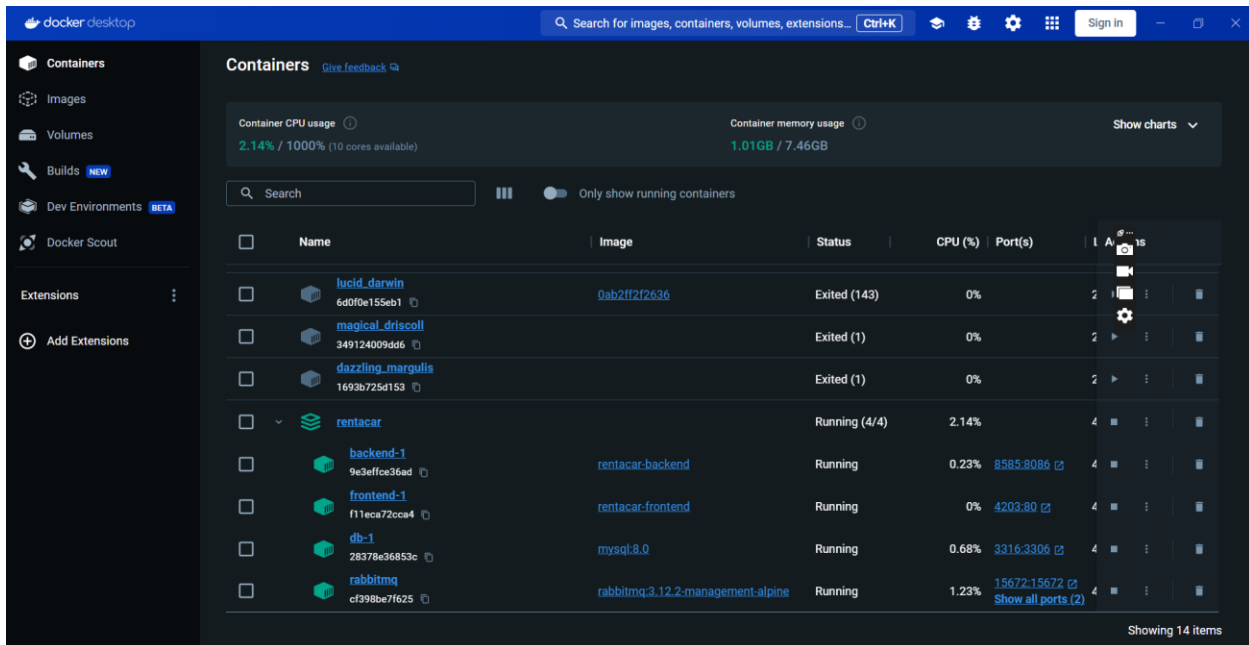
Build History

Filter...

#4 Jan 20 18:30 2 commits

#3 Jan 20 18:09 1 commit

#2 Jan 20 17:30 1 commit



docker desktop Search for images, containers, volumes, extensions... Ctrl+K

Containers Give feedback

Container CPU usage: 2.14% / 1000% (10 cores available)

Container memory usage: 1.01GB / 7.46GB

Search

Only show running containers

Name	Image	Status	CPU (%)	Port(s)
lucid_darwin	0ab2ff2/2636	Exited (143)	0%	2
magical_driscoll	349124009dd6	Exited (1)	0%	2
dazzling_margulis	1692b725d153	Exited (1)	0%	2
rentacar		Running (4/4)	2.14%	4
backend-1	rentacar-backend	Running	0.23%	8585:8086
frontend-1	rentacar-frontend	Running	0%	4203:80
db-1	mysql:8.0	Running	0.68%	3316:3306
rabbitmq	rabbitmq:3.12.2-management-alpine	Running	1.23%	15672:15672

Showing 14 items



ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR

Membre de

HONORIS UNITED UNIVERSITIES

Containers

Images

Volumes

Builds **NEW**

Dev Environments **BETA**

Docker Scout

Extensions

Add Extensions

Search for images, containers, volumes, extensions... **Ctrl+K**

Sign in

rentacar-backend-1

rentacar-backend
9e3efce36ad
8585.8086

STATUS
Running (9 minutes ago)

Logs

Inspect

Bind mounts

Exec

Files

Stats

```
fied":{"Sat, 20 Jan 2024 19:13:12 GMT"},"Pragma":{"no-cache"},"Vary":{"Origin"},"Access-Control-Request-Method","Access-Control-Request-Headers"},"X-Content-Type-Options":["nosniff"],"X-Frame-Options":["DENY"],"X-XSS-Protection":["0"]},"body":{"skipped-"}
2024-01-20T19:27:11.329Z TRACE 1 --- (Rentacar) [nio-8086-exec-4] org.zalando.logbook.Logbook : {"origin":"local","type":"response","correlation":"e6d65f0c24715eef","duration":20,"protocol":"HTTP/1.1","status":200,"headers":{"Accept-Ranges":["bytes"],"Cache-Control":["no-cache, no-store, max-age=0, must-revalidate"],"Connection":["keep-alive"],"Content-Length":["44378"],"Content-Type":["image/webp"],"Date":["Sat, 20 Jan 2024 19:27:10 GMT"],"Expires":["0"],"Keep-Alive":["timeout=60"],"Last-Modified":["Sat, 20 Jan 2024 19:13:12 GMT"],"Pragma":{"no-cache"},"Vary":{"Origin"},"Access-Control-Request-Method","Access-Control-Request-Headers"},"X-Content-Type-Options":["nosniff"],"X-Frame-Options":["DENY"],"X-XSS-Protection":["0"]},"body":{"skipped-"}
2024-01-20T19:27:11.333Z TRACE 1 --- (Rentacar) [nio-8086-exec-3] org.zalando.logbook.Logbook : {"origin":"local","type":"response","correlation":"e1c36ba3a310f97b","duration":26,"protocol":"HTTP/1.1","status":200,"headers":{"Accept-Ranges":["bytes"],"Cache-Control":["no-cache, no-store, max-age=0, must-revalidate"],"Connection":["keep-alive"],"Content-Length":["30180"],"Content-Type":["font/woff2"],"Date":["Sat, 20 Jan 2024 19:27:10 GMT"],"Expires":["0"],"Keep-Alive":["timeout=60"],"Last-Modified":["Sat, 20 Jan 2024 19:13:12 GMT"],"Pragma":{"no-cache"},"Vary":{"Origin"},"Access-Control-Request-Method","Access-Control-Request-Headers"},"X-Content-Type-Options":["nosniff"],"X-Frame-Options":["DENY"],"X-XSS-Protection":["0"]},"body":{"skipped-"}
2024-01-20T19:27:11.345Z TRACE 1 --- (Rentacar) [nio-8086-exec-7] org.zalando.logbook.Logbook : {"origin":"local","type":"response","correlation":"b3463faf401596752","duration":30,"protocol":"HTTP/1.1","status":200,"headers":{"Accept-Ranges":["bytes"],"Cache-Control":["no-cache, no-store, max-age=0, must-revalidate"],"Connection":["keep-alive"],"Content-Length":["168260"],"Content-Type":["font/ttf"],"Date":["Sat, 20 Jan 2024 19:27:10 GMT"],"Expires":["0"],"Keep-Alive":["timeout=60"],"Last-Modified":["Sat, 20 Jan 2024 19:13:12 GMT"],"Pragma":{"no-cache"},"Vary":{"Origin"},"Access-Control-Request-Method","Access-Control-Request-Headers"},"X-Content-Type-Options":["nosniff"],"X-Frame-Options":["DENY"],"X-XSS-Protection":["0"]},"body":{"skipped-"}
2024-01-20T19:27:11.389Z TRACE 1 --- (Rentacar) [nio-8086-exec-8] org.zalando.logbook.Logbook : {"origin":"remote","type":"request","correlation":"b24467efa2c874bc","protocol":"HTTP/1.1","remote":"172.25.0.1","method":"GET","url":"http://localhost:8585/favicon.ico","host":"localhost","path":"/favicon.ico","scheme":"http","port":8585,"headers":{"accept":["image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8"],"accept-encoding":["gzip, deflate, br"],"accept-language":["en-US,en;q=0.9"],"connection":["keep-alive"],"cookie":["_ga-GA1.1.1383884829.1698397418; _ga_5R3HG8EBP-CS1.1.1708078834.8.1.1708082747.0.0.0; Idea-2a3d4c-71a3c6cd-a732-4012-ac16-b2da208cc2cp; jenkins-timestamper-offset=-3600000; JSESSSIONID.8839743d-node03c2uy4y6er93wQg8lkt6ssbq54.node0; screenResolution=1536x864; JSESSSIONID.e23c7198-node081rbvh3b5w4sk1oocjbzbf48p6p6.node0"],"host":["localhost:8585"],"referer":["http://localhost:8585/"],"sec-ch-ua":["\"Not A Brand\";v=\"8\"\",\"Chromium\";v=\"120\"\",\"Google Chrome\";v=\"120\""],"sec-ch-ua-mobile":["0"],"sec-ch-ua-platform":["Windows"],"sec-fetch-dest":["image"],"sec-fetch-mode":["no-cors"],"sec-fetch-site":["same-origin"],"user-agent":["Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"]}}
2024-01-20T19:27:11.399Z TRACE 1 --- (Rentacar) [nio-8086-exec-8] org.zalando.logbook.Logbook : {"origin":"local","type":"response","correlation":"b24467efa2c874bc","duration":10,"protocol":"HTTP/1.1","status":200,"headers":{"Accept-Ranges":["bytes"],"Cache-Control":["no-cache, no-store, max-age=0, must-revalidate"],"Connection":["keep-alive"],"Content-Length":["848"],"Content-Type":["image/x-icon"],"Date":["Sat, 20 Jan 2024 19:27:10 GMT"],"Expires":["0"],"Keep-Alive":["timeout=60"],"Last-Modified":["Sat, 20 Jan 2024 19:13:12 GMT"],"Pragma":{"no-cache"},"Vary":{"Origin"},"Access-Control-Request-Method","Access-Control-Request-Headers"},"X-Content-Type-Options":["nosniff"],"X-Frame-Options":["DENY"],"X-XSS-Protection":["0"]},"body":{"skipped-"}

```

Engine running RAM 6.62 GB CPU 0.04% Not signed in v24.1.2

Containers

Images

Volumes

Builds **NEW**

Dev Environments **BETA**

Docker Scout

Extensions

Add Extensions

Search for images, containers, volumes, extensions... **Ctrl+K**

Sign in

rentacar-frontend-1

rentacar-frontend
f11eca72cc84
4203.80

STATUS
Running (11 minutes ago)

Logs

Inspect

Bind mounts

Exec

Files

Stats

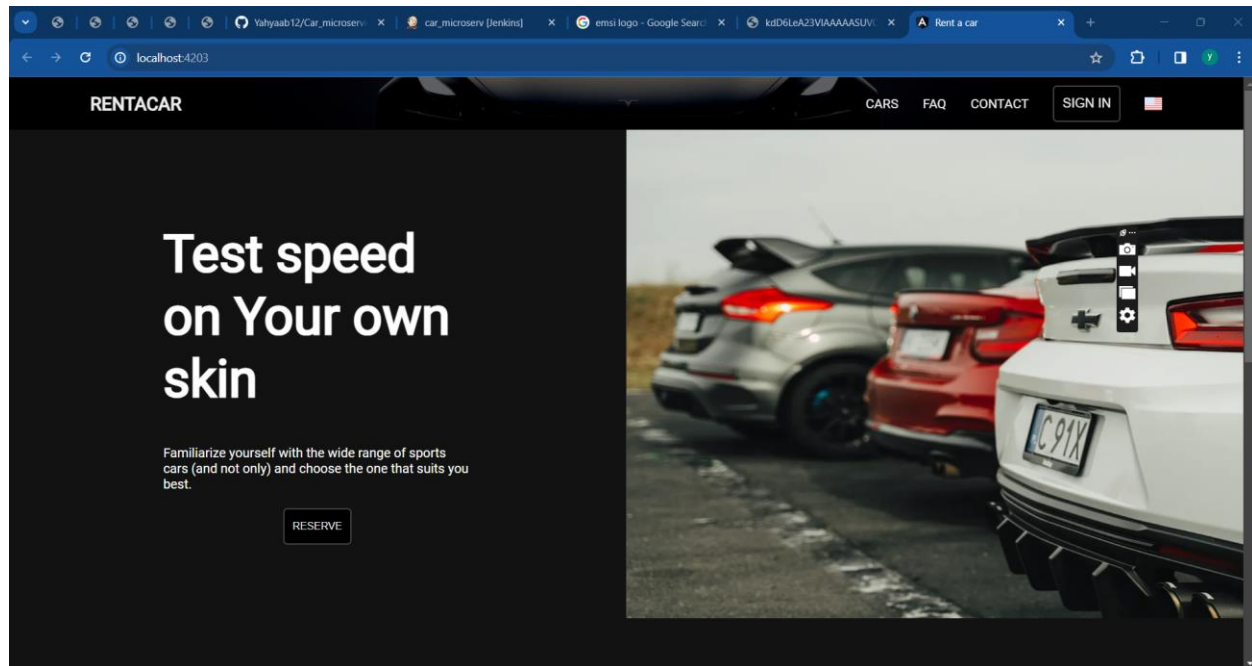
```
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /runtime.36c4ac3f543646009.js HTTP/1.1" 200 2905 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /polyfills.f661899c44cad771.js HTTP/1.1" 200 33815 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /main.529ced9938d8a568.js HTTP/1.1" 200 1517884 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /styles.7d9abc1e4531afe7.css HTTP/1.1" 200 213130 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /Fotolia_94386317_Subscription_Monthly_M.c83f1b2a81fd7e85.jpg HTTP/1.1" 200 85331 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /Roboto-Regular.d0bc87a819730d23.ttf HTTP/1.1" 200 168260 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /assets/118/en.json HTTP/1.1" 200 17017 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /assets/images/banner.webp HTTP/1.1" 200 44378 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /us.4f2d45574b8cd4508.svg HTTP/1.1" 200 651 "http://localhost:4203/styles.7d9abc1e4531afe7.css" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /prismelcons.ba3f9f16dfb4be8c.woff2 HTTP/1.1" 200 30180 "http://localhost:4203/styles.7d9abc1e4531afe7.css" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /assets/118/en.json HTTP/1.1" 200 18646 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET / HTTP/1.1" 304 0 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /fr.4ecb694b2fcc7c6f.svg HTTP/1.1" 200 231 "http://localhost:4203/styles.7d9abc1e4531afe7.css" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET /favicon.ico HTTP/1.1" 200 948 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:09 +0000] "GET / HTTP/1.1" 304 0 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.25.0.1 - - [20/Jan/2024:19:28:30 +0000] "GET / HTTP/1.1" 304 0 "http://localhost:4203/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"

```

Engine running RAM 6.67 GB CPU 0.12% Not signed in v24.1.2

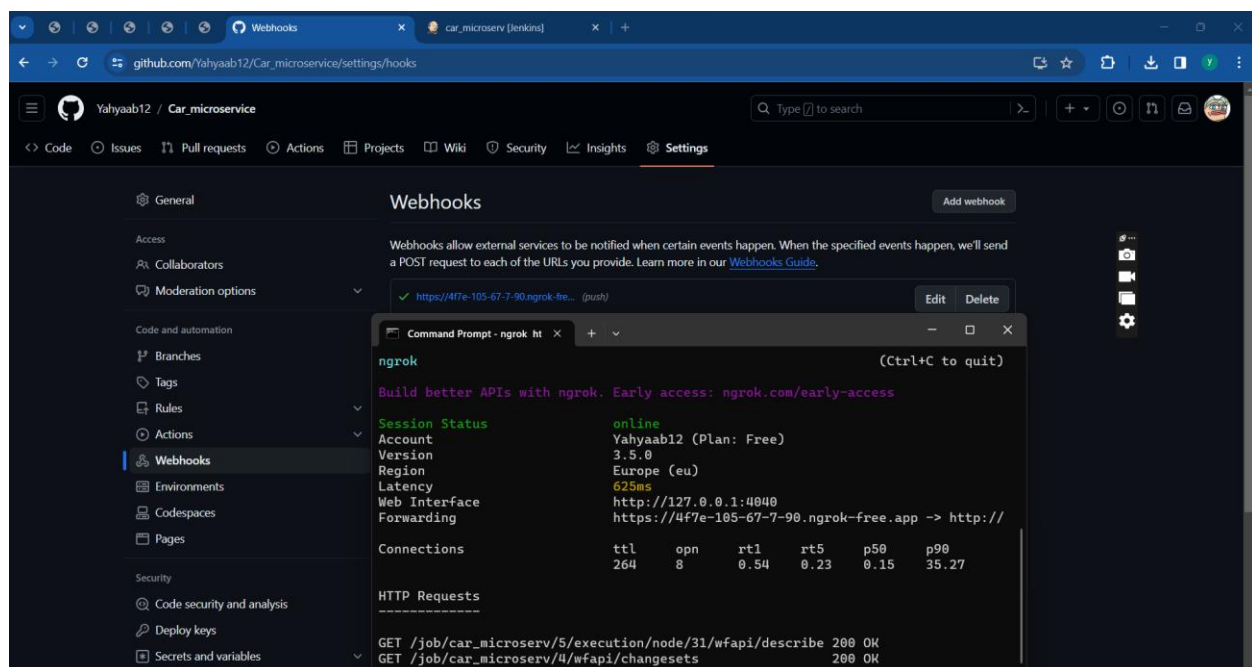


ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR
Membre de
HONORIS UNITED UNIVERSITIES



6. Déploiement Automatique

Le déploiement automatique est un processus crucial dans notre projet de location de voitures, assurant une livraison rapide et efficace des mises à jour du logiciel. Nous avons mis en place un mécanisme automatisé en utilisant Jenkins et ngrok pour simplifier le déploiement. Voici comment le processus fonctionne:





1. Configuration de Jenkins :

- Jenkins est utilisé comme outil d'intégration continue (CI) pour automatiser le processus de construction et de déploiement.
- Les étapes de construction sont définies dans des fichiers de configuration Jenkins, spécifiant les actions à effectuer à chaque modification de code source.

2. Exposition de Jenkins avec ngrok :

- Pour permettre à des services externes d'accéder à Jenkins, nous utilisons ngrok.
- La commande **ngrok http 8080** expose le serveur Jenkins localement sur un URL public, permettant aux services cloud, comme GitHub, de déclencher des opérations sur Jenkins.

3. Configuration des Webhooks :

- Nous utilisons des webhooks pour déclencher automatiquement Jenkins lorsqu'un événement spécifique se produit, tel qu'un nouveau commit sur GitHub.
- Ces webhooks sont configurés dans les services hébergeant notre code source, créant ainsi un lien direct avec Jenkins.

4. Déclenchement Automatique du Processus :

- Lorsqu'un développeur effectue un commit ou une fusion sur le référentiel de code source, le webhook associé déclenche automatiquement Jenkins via ngrok.
- Jenkins récupère les dernières modifications du code, exécute les étapes de construction définies, puis déploie automatiquement les mises à jour sur les environnements appropriés.

Ce processus de déploiement automatique garantit une réactivité rapide aux changements de code, minimise les erreurs humaines et assure une cohérence dans le déploiement sur diverses instances de microservices. L'utilisation de ngrok facilite la connexion sécurisée entre Jenkins et les services externes, contribuant ainsi à un processus de développement fluide et automatisé.

PS : Nous avons réalisé une vidéo sur le déploiement automatique la vidéo dans le README.

7. Conclusion



Résumé des Accomplissements

Au terme de ce projet de location de voitures basé sur une architecture microservices, plusieurs accomplissements significatifs ont été réalisés. Voici un résumé des principales réalisations :

1. Architecture Microservices :

- Mise en place d'une architecture modulaire favorisant la flexibilité et l'évolutivité.
- Utilisation efficace de RabbitMQ pour la communication asynchrone entre les microservices.

2. Déploiement Docker :

- Utilisation de conteneurs Docker pour l'encapsulation et le déploiement indépendant de chaque microservice.
- Mise en œuvre de constructions multi-étapes pour des images finales optimisées.

3. Intégration Continue (CI) avec Jenkins :

- Configuration d'un processus CI/CD automatisé pour assurer des déploiements rapides et fiables.
- Utilisation de webhooks et de ngrok pour déclencher automatiquement les pipelines Jenkins.

4. Gestion de la Sécurité :

- Implémentation de mécanismes de sécurité, y compris l'authentification JWT et la gestion des utilisateurs.
- Intégration de l'authentification à deux facteurs (2FA) pour renforcer la sécurité.

5. Gestion des Erreurs et des Messages :

- Mise en place de mécanismes robustes pour gérer les erreurs et garantir la fiabilité des communications.

Perspectives Futures

Pour l'avenir, plusieurs axes d'amélioration et d'extension peuvent être envisagés :

1. Évolutivité et Performance :

- Explorer des stratégies d'évolutivité horizontale pour faire face à une augmentation de la charge.
- Optimiser les performances des microservices pour garantir une réponse rapide aux requêtes.

2. Intégration de Nouvelles Fonctionnalités :



- Ajouter de nouvelles fonctionnalités pour améliorer l'expérience utilisateur, comme la gestion des commentaires et des évaluations.

3. Monitoring et Analyse :

- Mettre en place des outils de surveillance pour suivre les performances et détecter les problèmes proactivement.
- Utiliser des analyses pour comprendre les tendances d'utilisation et planifier les améliorations futures.

4. Améliorations de Sécurité :

- Effectuer des audits de sécurité réguliers et mettre en œuvre des correctifs en cas de nouvelles vulnérabilités.
- Explorer des méthodes avancées de gestion de la sécurité, telles que la détection d'intrusion.

En résumé, ce projet représente un jalon significatif dans le domaine des applications de location de voitures basées sur des microservices. Les accomplissements actuels serviront de fondement solide pour les évolutions futures, permettant ainsi de maintenir une application robuste, sécurisée et adaptée aux besoins changeants de l'utilisateur.