

Une plate-forme micro-
services
avec configuration
distante

Sommaire

Ce document est constitué des chapitres ci-dessous :

(00.) Introduction

(01.) L'environnement logiciel requis

(02.) L'architecture logicielle de la plate-forme

(03.) Un micro-service simple : « microservice-produit »

(04.) Le dépôt distant

(05.) La communication sécurisée avec le protocole SSH

(06.) Le dépôt local

(07.) Le serveur de configuration

(08.) Le déploiement de la plate-forme micro-services

(09.) Les tests sur la plate-forme micro-services

(00.) Introduction

L'objectif de ce document est la réalisation puis la mise en service d'une plate-forme de micro-services réduite à sa forme minimale : Elle sera constituée des éléments suivants :

- Un micro-service complet (Plus tard, celui-ci servira de modèle pour réaliser d'autres micro-services).
- Un dépôt distant.
- Un serveur de configuration (Celui-ci possédera une architecture de micro-service, mais avec des caractéristiques particulières)

(01.) L'environnement logiciel requis

Les activités proposées dans ce document requièrent :

(01.01.) L'installation sur votre ordinateur de plusieurs logiciels

(01.02.) La création de plusieurs variables d'environnement (dans votre système d'exploitation)

(01.03.) L'enrichissement de la variable d'environnement « path » (dans votre système d'exploitation)

Ces 3 pré-requis sont détaillés dans les paragraphes ci-dessous.

(01.01.) L'installation des logiciels

Les **logiciels que vous devez installer sur votre ordinateur** sont listées ci-dessous :

(01.01.01.) Le kit de développement Java : JDK8

(01.01.02.) L'environnement d'exécution Java : JRE8

(01.01.03.) Le gestionnaire de cycle de vie des projets java : Apache Maven

(01.01.04.) Le serveur d'applications : Apache Tomcat

(01.01.05.) Le client lourd « REST » : Postman

(01.01.06.) Le serveur de bases de données : MariaDB

(01.01.07.) Le gestionnaire de versions : Git

(01.01.08.) L'environnement de développement intégré : STS (Spring-Tools-Suite)

(01.02.) La création des variables d'environnement

Vous devez créer dans votre système d'exploitation les variables d'environnement détaillées ci-dessous :

NOMS :	VALEURS :
JAVA_HOME	[Répertoire d'installation du JDK8]
JRE_HOME	[Répertoire d'installation du JRE8]
CATALINA_HOME	[Répertoire d'installation de APACHE-TOMCAT]

(01.03.) L'enrichissement de la variable d'environnement « path »

Vous devez enrichir la variable d'environnement « path » de votre système d'exploitation avec les chemins de répertoires listés ci-dessous :

NOMS :	VALEURS :
MAVEN_BIN	[Répertoire d'installation de MAVEN]\bin
JDK_BIN	[Répertoire d'installation du JDK8]\bin
JRE_BIN	[Répertoire d'installation du JRE8]\bin
GIT_CMD	[Répertoire d'installation de GIT]\cmd

(02.) L'architecture logicielle de la plate-forme

Ce chapitre fournit une description détaillée de **l'architecture logicielle de la plate-forme de micro-service**. Celle-ci est constituée des éléments suivants :

(02.01.) Un **micro-service complet**

(02.02.) Un **serveur de configuration**

(02.03.) Un **dépôt de configuration distant**.

Ces **3 constituants de la plate-forme** seront détaillés dans des paragraphes dédiés ci-dessous.

(02.01.) Un micro-service complet

Le micro-service que je vous propose de réaliser s'appelle : « **micro-service produit** ».

Ce micro-service est intégré au sein de la plate-forme **par des mécanismes** qui sont présentés dans le paragraphe ci-dessous.

(02.01.01.) Intégration du « micro-service produit » au sein de la plate-forme

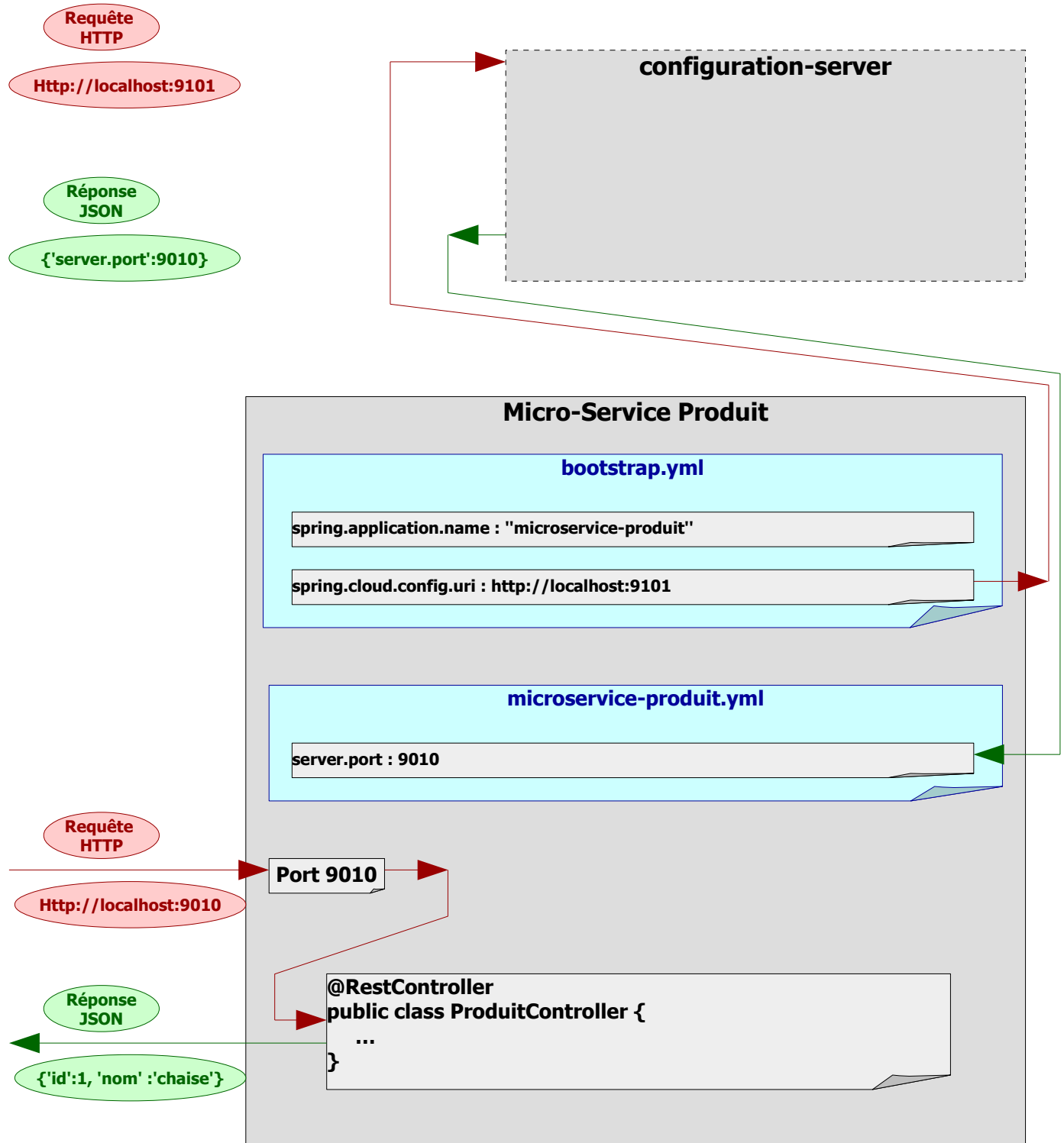
Ces **mécanismes** sont détaillés dans les 2 paragraphes ci-dessous :

(02.01.02.) La phase d'initialisation du « micro-service-produit »

(02.01.03.) La phase d'exploitation du « micro-service produit »

(02.01.01.) Intégration du « micro-service produit » au sein de la plate-forme

L'intégration du « micro-service produit » au sein de la plate-forme : Elle est illustrée par le schéma ci-dessous :



(02.01.02.) La phase d'initialisation du « micro-service produit »

La phase d'initialisation du « micro-service produit » : elle démarre au lancement de celui-ci.

Le fichier de propriétés « bootstrap.yml », est récupéré par le framework « Spring-Boot », qui en charge le contenu dans le contexte de l'application.

En particulier, la propriété « spring.cloud.config.uri », ainsi que sa valeur « http://localhost:9101 », sont récupérées par « Spring-Boot », qui envoie une requête REST à cette URI.

La réponse à cette requête REST, au format JSON, contient une liste de propriétés avec leurs valeurs respectives. En particulier la propriété « server.port », avec sa valeur « 9010 ».

Le « micro-service produit » interprète ce numéro de port comme étant son port d'écoute HTTP.

(02.01.03.) La phase d'exploitation du « micro-service produit »

La phase d'exploitation du « micro-service produit » : elle démarre après que sa phase d'initialisation soit terminée.

Le « micro-service produit » est à l'écoute du port 9010, et à la réception de chaque requête REST, il la redirige vers son contrôleur REST, puis dans ce contrôleur, vers une méthode dédiée à ce type de requête.

Ce contrôleur REST, à chaque requête appropriée, effectuera un traitement et renverra une réponse sous forme d'un bloc de données au format JSON.

(02.02.) Un serveur de configuration

Le **serveur de configuration** que je vous propose de réaliser s'appelle : « **config-server** ».

Le serveur de configuration, par son architecture interne, est lui aussi un micro-service.

Ce serveur de configuration est intégré au sein de la plate-forme **par des mécanismes** qui sont présentés dans le paragraphe ci-dessous.

(02.02.01.) Intégration du serveur de configuration au sein de la plate-forme

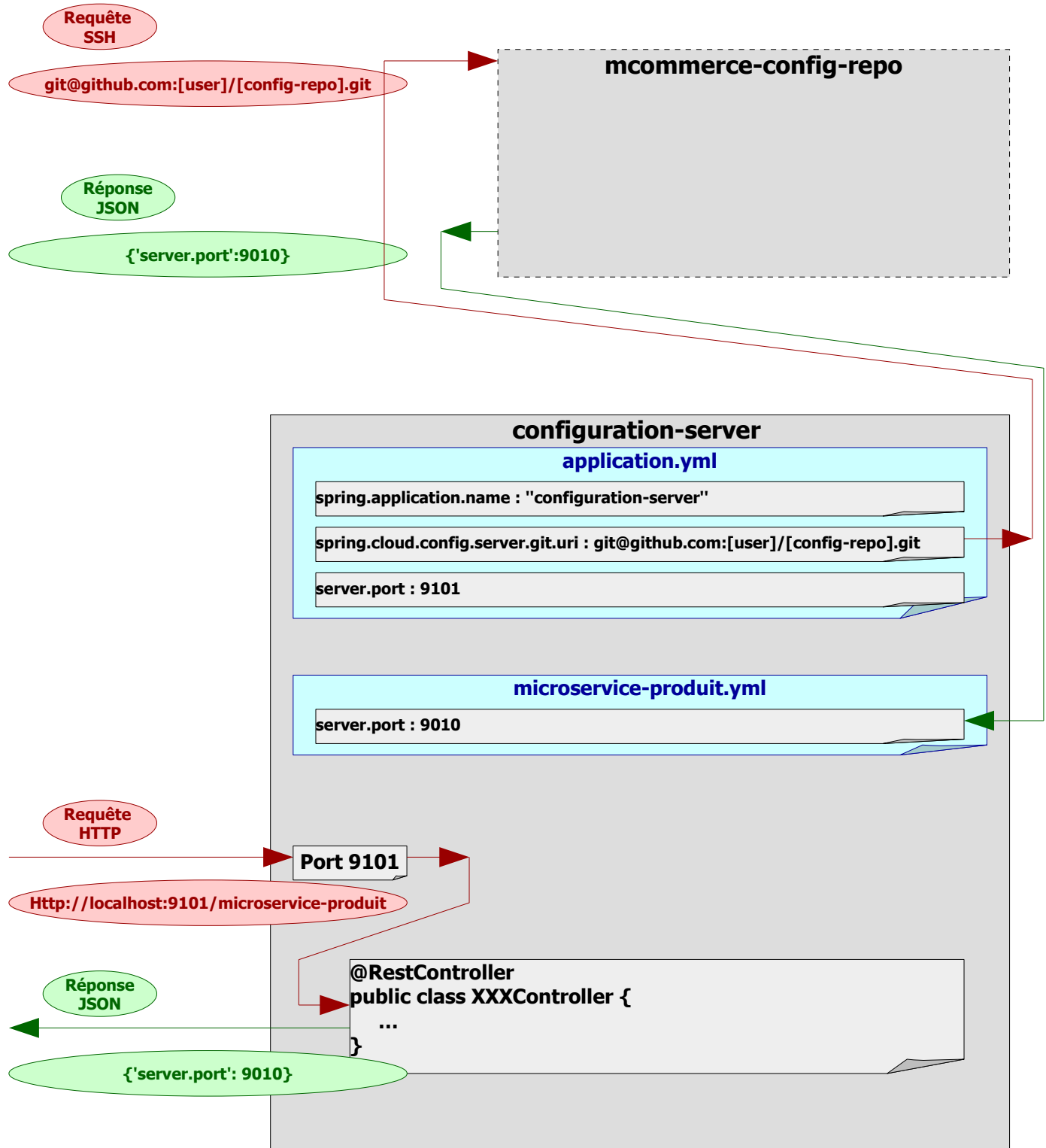
Ces **mécanismes** sont détaillés dans les 2 paragraphes ci-dessous :

(02.02.02.) La phase d'initialisation du serveur de configuration

(02.02.03.) La phase d'exploitation du serveur de configuration

(02.02.01.) Intégration du « serveur de configuration » au sein de la plate-forme

L'intégration du « serveur de configuration » au sein de la plate-forme : Elle est illustrée par le schéma ci-dessous :



(02.02.02.) La phase d'initialisation du serveur de configuration

La **phase d'initialisation du serveur de configuration** : elle démarre au **lancement de celui-ci**.

Le fichier de propriétés « **application.yml** », est récupéré par le **framework « Spring-Boot »**, qui en charge le contenu dans le **contexte de l'application**.

En particulier, la propriété « **spring.cloud.config.server.git.uri** », ainsi que sa valeur « **git@github.com:[user]/I[config-repo].git** », sont récupérées par « **Spring-Boot** », qui envoie une requête REST à cette URI.

La **réponse à cette requête REST**, au format JSON, contient une liste de **propriétés** avec leurs **valeurs respectives**.

(02.02.03.) La phase d'exploitation du serveur de configuration

La **phase d'exploitation du serveur de configuration** : elle démarre **après que sa phase d'initialisation soit terminée**.

Le serveur de configuration est **à l'écoute du port 9101**, et à la réception de chaque requête REST, il la **redirige vers son un contrôleur REST** dédié à ce type de requête.

Ce contrôleur REST, à chaque requête appropriée, **effectuera un traitement** et **renverra une réponse sous forme d'un bloc de données au format JSON**.

(02.03.) Un dépôt de configuration distant.

Le « dépôt de configuration distant » que je vous propose de réaliser s'appelle : « mcommerce-03-config-repo ».

Ce dépôt distant est intégré au sein de la plate-forme par des mécanismes qui sont présentés dans le paragraphe ci-dessous.

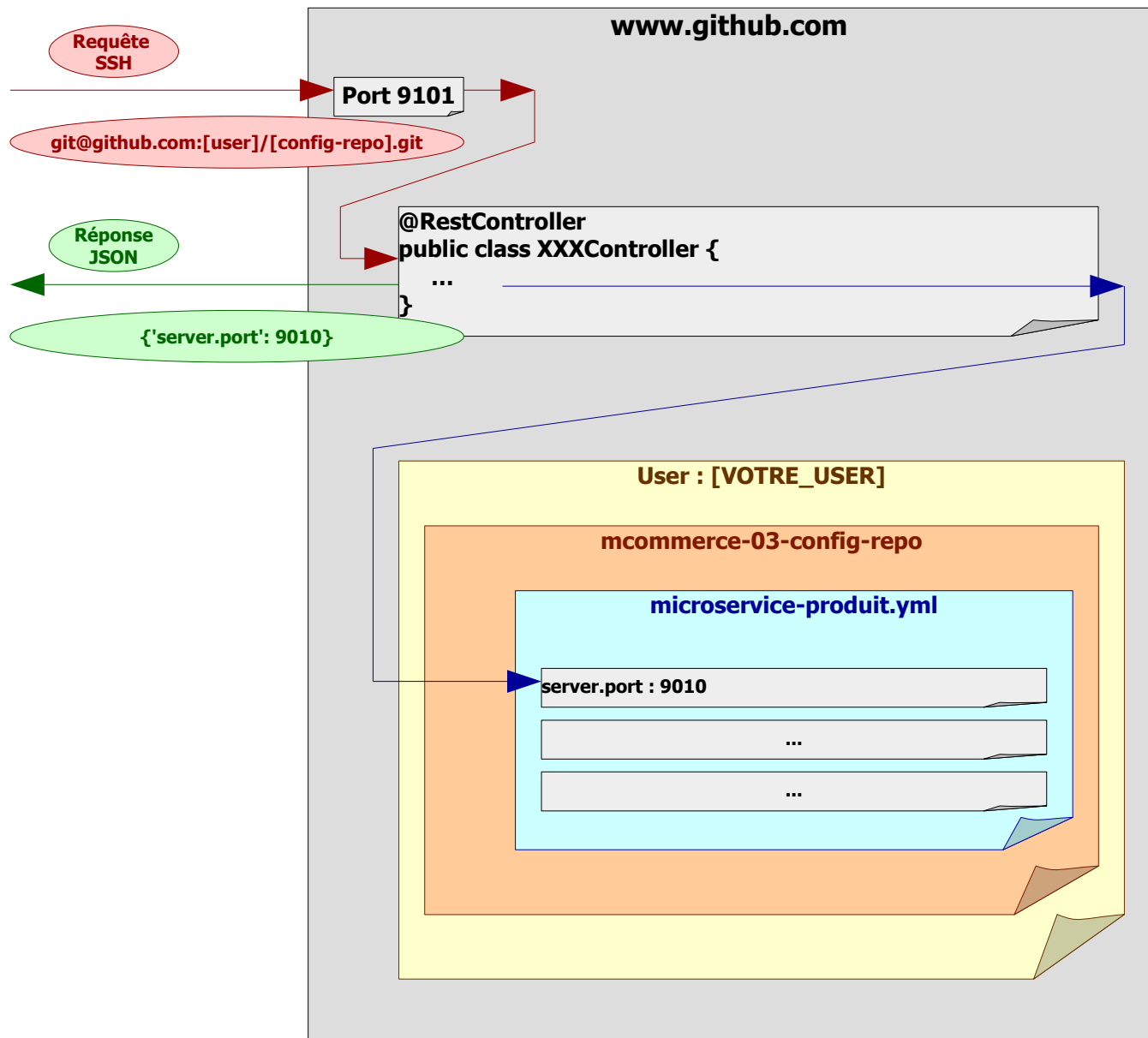
(02.03.01.) L'intégration du dépôt de configuration distant au sein de la plate-forme

Ces mécanismes sont détaillés dans le paragraphe ci-dessous.

(02.03.02.) L'exploitation du dépôt de configuration distant

(02.03.01.) L'intégration du dépôt distant au sein de la plate-forme

L'intégration du dépôt distant au sein de la plate-forme : Elle est illustrée par le schéma ci-dessous :



(02.03.02.) L'exploitation du dépôt distant

- Le déploiement de notre dépôt distant : est effectué sur un serveur d'applications.
- L'interrogation de ce serveur d'applications : est effectué par envoi de requêtes REST à son URI d'entrée.
- L'URI d'entrée du serveur d'applications : c'est « www.github.com ».
- La création de notre dépôt distant : sera effectuée sur votre compte utilisateur sur le serveur « github.com ».
- L'interrogation de notre dépôt distant : sera effectuée en complétant l'URI d'entrée du serveur, par rajout de votre identifiant (propre à votre compte utilisateur sur « github.com »).

(03.) Un micro-service simple : « microservice-produit »

Ce chapitre est constitué des **sous-chapitres listés ci-dessous** :

(03.01.) L'arborescence des répertoires du projet.

(03.02.) Le fichier de configuration « pom.xml » du projet.

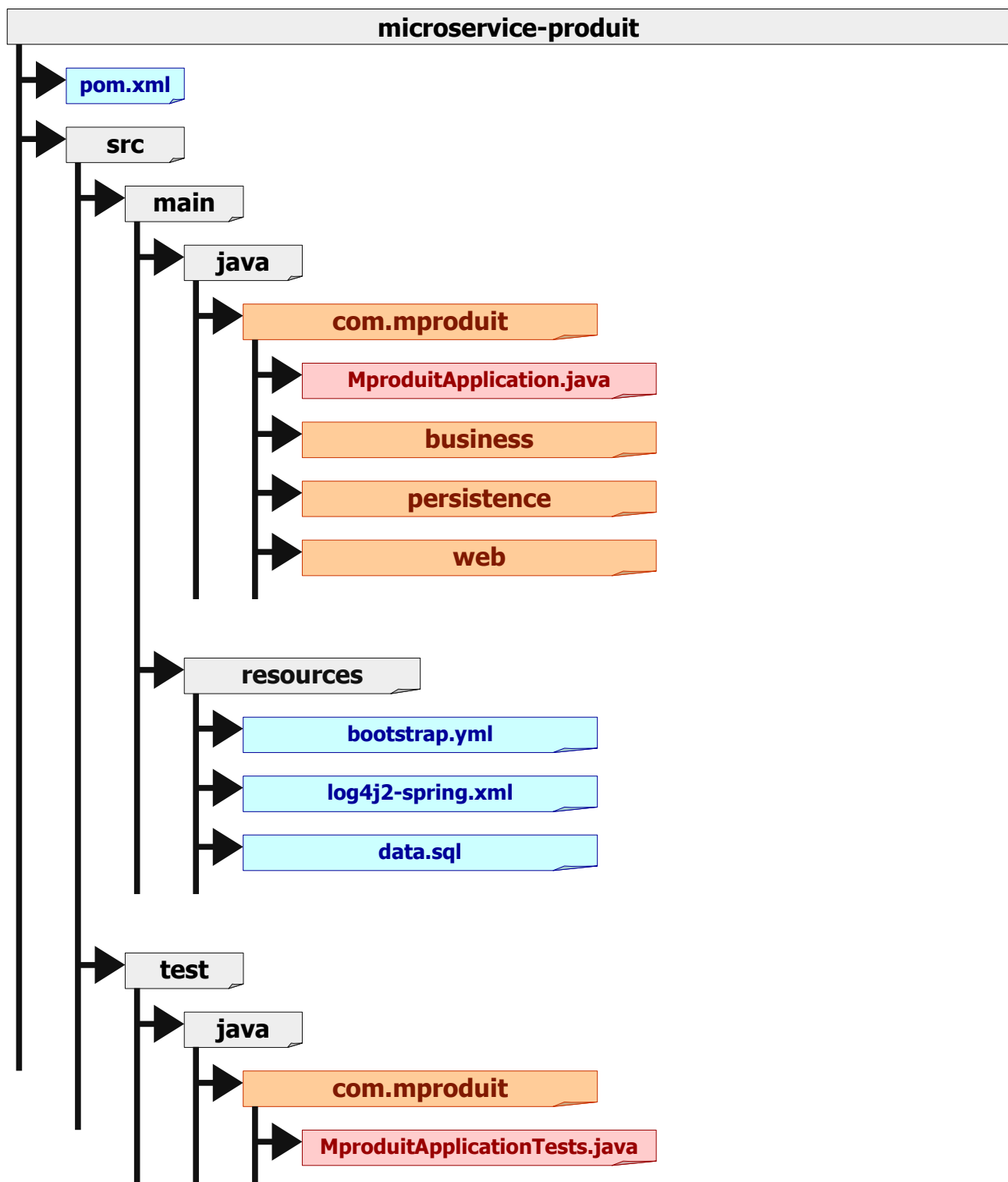
(03.03.) Le fichier de propriétés « bootstrap.yml » de l'application.

(03.04.) Le fichier de configuration des loggers.

(03.05.) La classe principale de l'application.

(03.01.) L'arborescence des répertoires du projet

L'arborescence des répertoires du projet sera celle d'une **application web avec une architecture N-Tiers**.
Veuillez créer cette arborescence conformément au schéma ci-dessous, svp.



(03.02.) Le fichier de configuration « pom.xml » du projet

Lorsque l'on demande à MAVEN de compiler le projet, MAVEN parcourt ce fichier et configure le procédé de compilation en fonction du contenu de ce fichier.

Les informations permettant de trouver ce fichier sont fournies ci-dessous :

Répertoire	La racine du projet
Fichier	pom.xml

Le contenu de ce fichier est détaillé dans les rubriques ci-dessous :

(03.02.01.) La Rubrique : « Informations du projet »

(03.02.02.) La rubrique : « Informations du projet parent »

(03.02.03.) La rubrique : « Propriétés du projet »

(03.02.04.) La rubrique : « Dépendances »

(03.02.05.) La rubrique : « Management des dépendances »

(03.02.06.) La rubrique : « Plugins »

(03.02.07.) La rubrique : « Nom final de l'exécutable »

(03.02.01.) La rubrique : « Informations du projet »

Cette rubrique est constituée du [bloc ci-dessous](#). [Veuillez le récupérer](#) , svp.

```
<modelVersion>4.0.0</modelVersion>

<groupId>com.mproduit</groupId>
<artifactId>mproduit</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>mproduit</name>
<description>Microservice de gestion des produits</description>
```

(03.02.02.) La rubrique : « Informations du projet parent »

Cette rubrique est constituée du [bloc ci-dessous](#). [Veuillez le récupérer](#) , svp.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.0.4.RELEASE</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
```

(03.02.03.) La rubrique : « Propriétés du projet »

Cette rubrique est constituée du [bloc ci-dessous](#). [Veuillez le récupérer](#) , svp.

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <java.version>1.8</java.version>
  <spring-cloud.version>Finchley.SR1</spring-cloud.version>
</properties>
```

(03.02.04.) La rubrique : « Dépendances »

Cette rubrique est constituée du [bloc ci-dessous](#). [Veuillez le récupérer](#) , svp.

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
    <exclusions>
      <exclusion>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-logging</artifactId>
      </exclusion>
    </exclusions>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-log4j2</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-configuration-processor</artifactId>
    <optional>true</optional>
  </dependency>

  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
  </dependency>

  <!-- Connecteur JDBC pour MYSQL -->
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
  </dependency>

  <!-- Driver pour MariaDB -->
  <dependency>
    <groupId>org.mariadb.jdbc</groupId>
    <artifactId>mariadb-java-client</artifactId>
  </dependency>
</dependencies>
```

(03.02.05.) La rubrique : « Management des dépendances »

Cette rubrique est constituée du [bloc ci-dessous](#). [Veuillez le récupérer](#) , svp.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>${spring-cloud.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

(03.02.06.) La rubrique : « Plugins »

Cette rubrique est constituée du [bloc ci-dessous](#). [Veuillez le récupérer](#) , svp.

```
<plugins>
  <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
  </plugin>
</plugins>
```

(03.02.07.) La rubrique : « Nom final de l'exécutable »

Cette rubrique est constituée du [bloc ci-dessous](#). [Veuillez le récupérer](#) , svp.

```
<finalName>mproduit</finalName>
```

(03.03.) Le fichier de propriétés « bootstrap.yml » de l'application

Les informations permettant de trouver ce fichier sont fournies ci-dessous :

Répertoire	src\main\resources
Fichier	bootstrap.yml

Le contenu de ce fichier est détaillé dans le bloc ci-dessous. Veuillez le récupérer, svp.

```
#####  
# ---- (01.) SPRING-CONFIGURATION ----  
#####  
spring:  
  
# ---- (01.01.) SPRING-APPLICATION-CONFIGURATION ----  
application:  
  name: "microservice-produit"  
  
# ---- (01.02.) SPRING-CLOUD-CONFIGURATION ----  
cloud:  
  config:  
    uri: "http://localhost:9101"  
    fail-fast: true
```

Remarques :

- Le contenu de ce fichier : C'est une structure en forme d'arbre (ressemblant à une arborescence de packages, de classes et d'attributs).
- Les branches de cet arbre : Elles se terminent par une déclaration, sous la forme : « [clé] : [valeur] »
- Dans cette déclaration : la clé et la valeur sont séparées par un symbole « : » suivi d'un espace.
- Cette syntaxe déclarative : Elle doit être respectée rigoureusement, svp !!!

(03.04.) Le fichier de configuration des loggers

Les informations permettant de trouver ce fichier sont fournies ci-dessous :

Répertoire	src\main\resources
Fichier	log4j2-spring.xml

Le contenu de ce fichier est détaillé dans les rubriques ci-dessous :

(03.04.01.) La rubrique : « Properties »

(03.04.02.) La rubrique : « Appenders »

(03.04.03.) La rubrique : « Loggers »

(03.04.01.) La rubrique : « Properties »

Cette rubrique est constituée du bloc ci-dessous. Veuillez le récupérer , svp.

```
<Properties>
  <Property name="log-path">./logExecute</Property>
</Properties>
```


(03.04.02.) La rubrique : « Appenders »

Cette rubrique est constituée du [bloc ci-dessous](#). [Veuillez le récupérer](#), svp.

```
<!-- ===== -->
<!-- (01.) Appenders : -->
<!-- - (01.A.) Chaque Appender pointe sur le fichier qui lui est associe -->
<!-- - (01.B.) Chaque Appender ecrit dans ce fichier -->
<!-- ===== -->
<Appenders>
  <Console name="Console-Appender" target="SYSTEM_OUT">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </Console>

  <File name="MProduit-File-Appender" fileName="${log-path}/MProduit.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="Controller-File-Appender" fileName="${log-path}/MProduit-controller.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="Exception-File-Appender" fileName="${log-path}/MProduit-exception.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="Model-File-Appender" fileName="${log-path}/MProduit-model.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="Spring-File-Appender" fileName="${log-path}/MProduit-spring.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="SpringCloud-File-Appender" fileName="${log-path}/MProduit-springCloud.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="SpringBoot-File-Appender" fileName="${log-path}/MProduit-springBoot.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="SpringContext-File-Appender" fileName="${log-path}/MProduit-springContext.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>
</Appenders>
```

(03.04.03.) La rubrique : « Loggers »

Cette rubrique est constituée du **bloc ci-dessous**. **Veillez le récupérer**, svp.

```
<!-- ===== -->
<!-- (02.) Loggers : -->
<!-- - (02.A.) Chaque Logger scanne le package qui lui est associe -->
<!-- - (02.B.) Chaque Logger récupère les logs qu'il trouve dans les classes -->
<!-- ===== -->
<Loggers>
  <Root>
    <AppenderRef ref="Console-Appender" level="all" />
  </Root>

  <Logger name="com.mproduit" level="all" additivity="true">
    <AppenderRef ref="MProduit-File-Appender" level="all" />
  </Logger>

  <Logger name="com.mproduit.web.controller" level="all" additivity="true">
    <AppenderRef ref="Controller-File-Appender" level="all" />
  </Logger>

  <Logger name="com.mproduit.business.exception" level="all" additivity="true">
    <AppenderRef ref="Exception-File-Appender" level="all" />
  </Logger>

  <Logger name="com.mproduit.persistance.model" level="all" additivity="true">
    <AppenderRef ref="Model-File-Appender" level="all" />
  </Logger>

  <Logger name="org.springframework" level="info" additivity="true">
    <AppenderRef ref="Spring-File-Appender" level="all" />
  </Logger>

  <Logger name="org.springframework.cloud" level="info" additivity="true">
    <AppenderRef ref="SpringCloud-File-Appender" level="all" />
  </Logger>

  <Logger name="org.springframework.boot" level="info" additivity="true">
    <AppenderRef ref="SpringBoot-File-Appender" level="all" />
  </Logger>

  <Logger name="org.springframework.context" level="info" additivity="true">
    <AppenderRef ref="SpringContext-File-Appender" level="all" />
  </Logger>
</Loggers>
```

(03.05.) La classe principale de l'application

Les informations permettant de trouver cette classe sont fournies ci-dessous :

Répertoire	src\main\java
Package	com.mproduit
Classe	MProduitApplication

Le contenu de ce fichier est détaillé dans les rubriques ci-dessous :

(03.05.01.) Les annotations sur la classe

(03.05.02.) L'attribut « Logger »

(03.05.03.) La méthode « main »

(03.05.01.) Les annotations sur la classe

Les annotations à poser sur la classe sont indiquées ci-dessous. Veuillez les récupérer, svp.

```
@EnableConfigurationProperties
@SpringBootApplication
public class MProduitApplication {...}
```

(03.05.02.) L'attribut « Logger »

L'attribut « Logger » est déclaré dans le bloc de code ci-dessous. Veuillez le récupérer, svp.

```
/**
 * <b>OBJET QUI POSSEDE LES FONCTIONNALITES SUIVANTES : </b> <br/>
 * <br/>
 * Les fonctionnalites d'ecriture de messages de log dans la console.
 */
private static final Logger LOGGER = LoggerFactory.getLogger(MProduitApplication.class);
```

(03.05.03.) La méthode « main »

La méthode « main » est implémentée dans le [bloc de code ci-dessous](#). [Veuillez le récupérer](#), svp.

```
/**
 * <b>METHODE D'ENTREE DE L'APPLICATION</b><br/>
 *
 * @param args
 */
public static void main(String[] args) {

    LOGGER.info("CLASS : MProduitApplication -- METHOD : main - BEGIN");
    SpringApplication.run(MProduitApplication.class, args);
    LOGGER.info("CLASS : MProduitApplication -- METHOD : main -- END");
}
```

(04.) Le dépôt distant

Ce chapitre est constitué des **sous-chapitres listés ci-dessous** :

(04.01.) Créer un compte sur « github.com »

(04.02.) Ouvrir une session sur votre compte « github.com »

(04.03.) Créer le dépôt distant

(04.04.) Récupérer l'URL « SSH » du dépôt distant

(04.01.) Créer un compte sur « github.com »

Utiliser pour cela les **informations fournies** ci-dessous :

user	chat_roux
mot de passe	xxx

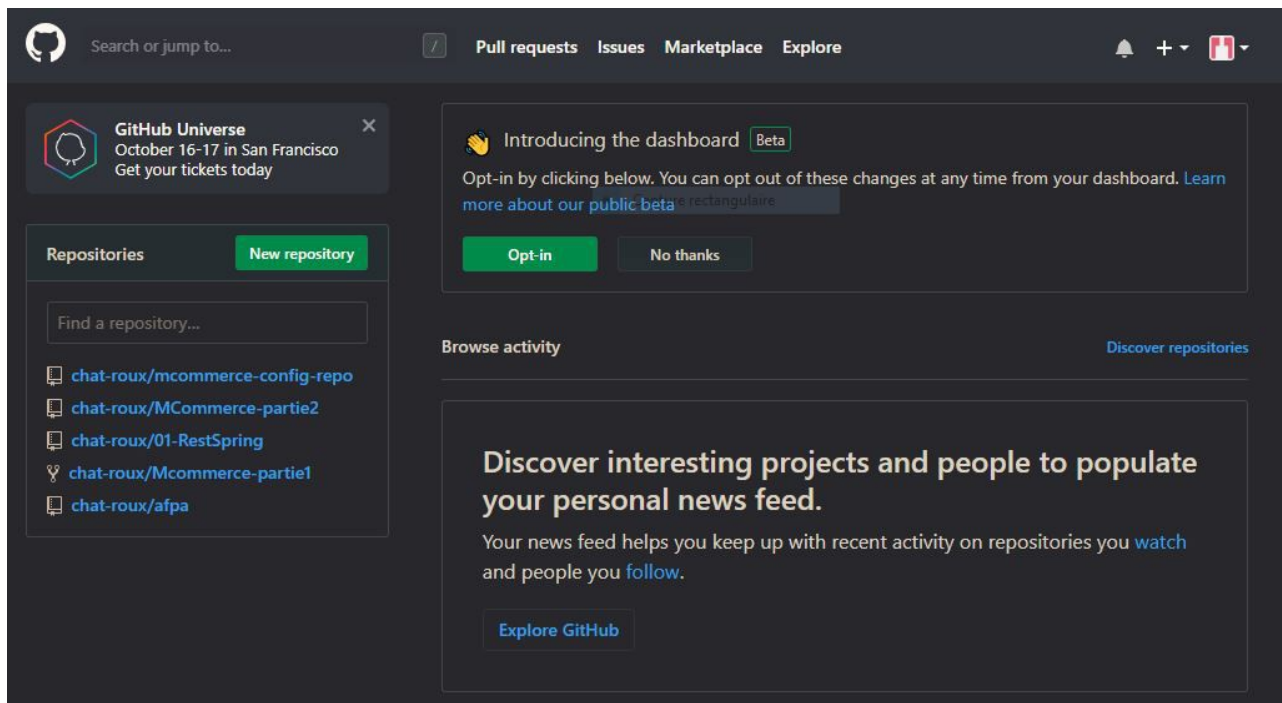
(04.02.) Ouvrir une session sur votre compte « github.com »

Utiliser pour cela les **informations fournies** ci-dessous :

user	chat_roux
mot de passe	xxx

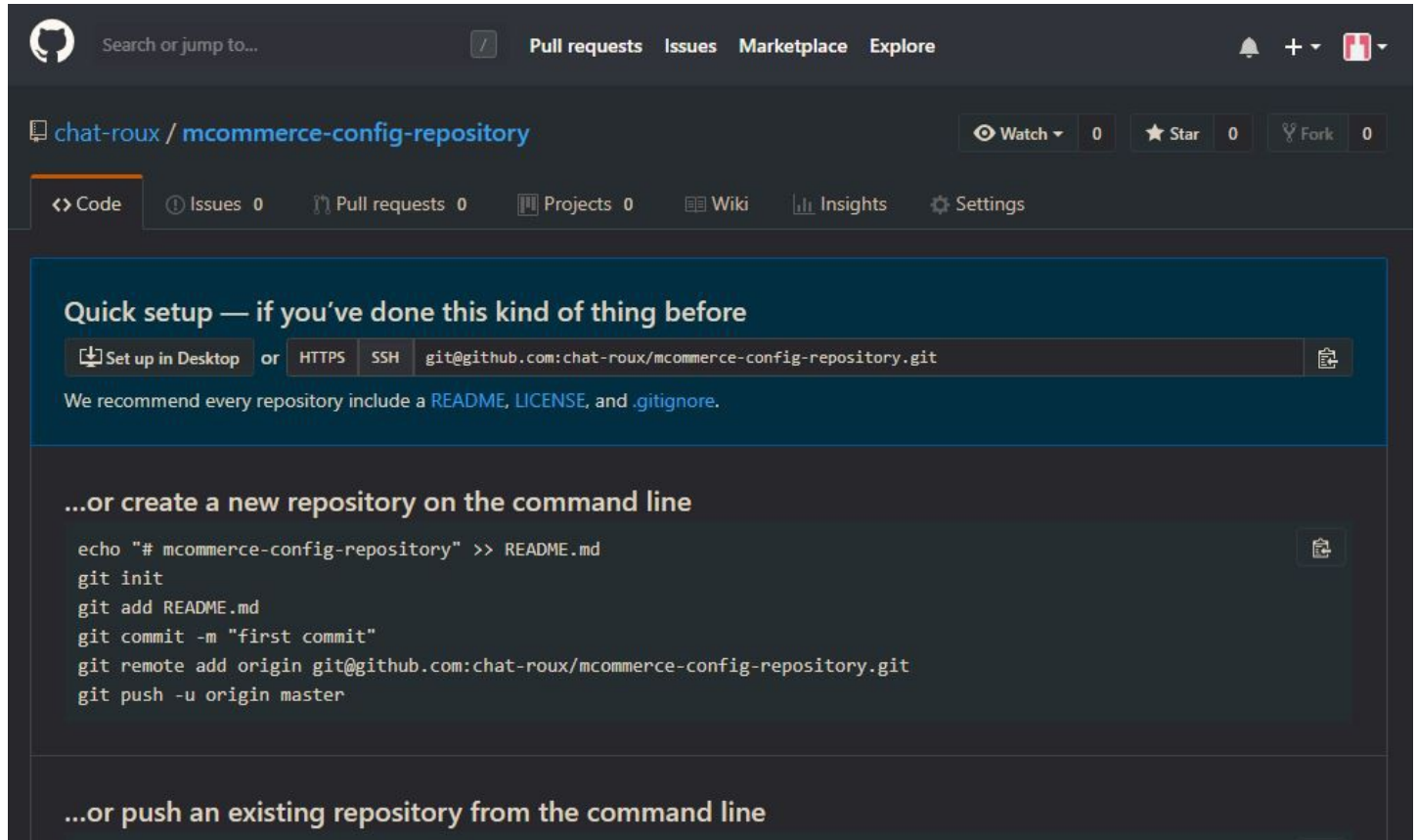
(04.03.) Créer le dépôt distant

- **Bouton** : « **New repository** »
- **Nom** : « **mcommerce-03-config-repo** »



(04.04.) Récupérer l'URI « SSH » du dépôt distant

- **Champ** : « SSH »
- **Sauvegarder sur votre machine locale** : Dans le **bloc-note** (vous en aurez besoin **au chapitre suivant**)



The screenshot shows the GitHub interface for the repository 'chat-roux / mcommerce-config-repository'. The top navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The repository name is 'chat-roux / mcommerce-config-repository'. Below the repository name, there are tabs for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. The 'Code' tab is selected, showing a 'Quick setup' section with options to 'Set up in Desktop', 'HTTPS', or 'SSH'. The SSH URL is 'git@github.com:chat-roux/mcommerce-config-repository.git'. Below this, there is a section titled '...or create a new repository on the command line' with a code block containing the following commands:

```
echo "# mcommerce-config-repository" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:chat-roux/mcommerce-config-repository.git
git push -u origin master
```

At the bottom, there is a section titled '...or push an existing repository from the command line'.

(05.) La communication sécurisée avec le protocole SSH

Ce chapitre est constitué des **sous-chapitres listés ci-dessous** :

(05.01.) Créer une clé SSH

(05.02.) Repérer votre clé SSH privée

(05.03.) Repérer votre clé SSH publique

(05.04.) Enregistrer votre clé SSH publique sur le serveur « github.com »

(05.05.) Demander la clé SSH publique du serveur « github.com »

(05.01.) Créer une clé SSH

- Démarrer une invite de commande « **Git Bash** »
- Lancer la **commande ci-dessous** :

```
#####  
# (01.)GENERATE RSA-SSH-KEY  
#####  
ssh-keygen -t rsa^  
            -b 1024^  
            -C "tcharou.dalgalian@afpa.fr"
```

Remarque :

Durant **l'exécution de la commande SSH** ci-dessus :

- Vous serez invité à **saisir une phrase de passe** (en anglais : « **passphrase** »).
- Veuillez la **sauvegarder dans votre bloc-note** !
- **Elle vous sera demandée** dans les prochains chapitres !!!

(05.02.) Repérer votre clé SSH privée

Votre **clé SSH privée** : Elle se trouve dans le fichier spécifié ci-dessous :

Répertoire	C:\Users\chat_roux\.ssh
Fichier	id_rsa

Remarques :

- Vous pouvez ouvrir ce fichier avec un éditeur de texte.
- Vous pouvez examiner le contenu de ce fichier avec un éditeur de texte.
- Vous aurez à le faire au chapitre (07) !!!

(Editeur de texte : Notepad++)

(05.03.) Repérer votre clé SSH publique

Votre **clé SSH publique** : Elle se trouve dans le fichier spécifié ci-dessous :

Répertoire	C:\Users\chat_roux\.ssh
Fichier	id_rsa.pub

Remarques :

- Vous pouvez ouvrir ce fichier avec un éditeur de texte.
- Vous pouvez examiner le contenu de ce fichier avec un éditeur de texte.
- Vous aurez à le faire au chapitre (07) !!!

(Editeur de texte : Notepad++)

(05.04.) Enregistrer votre clé SSH publique sur le serveur « github.com »

Dans « github.com » :

- Ouvrir le **menu latéral personnalisé**
- Cliquer sur l'entrée : « **Settings** »

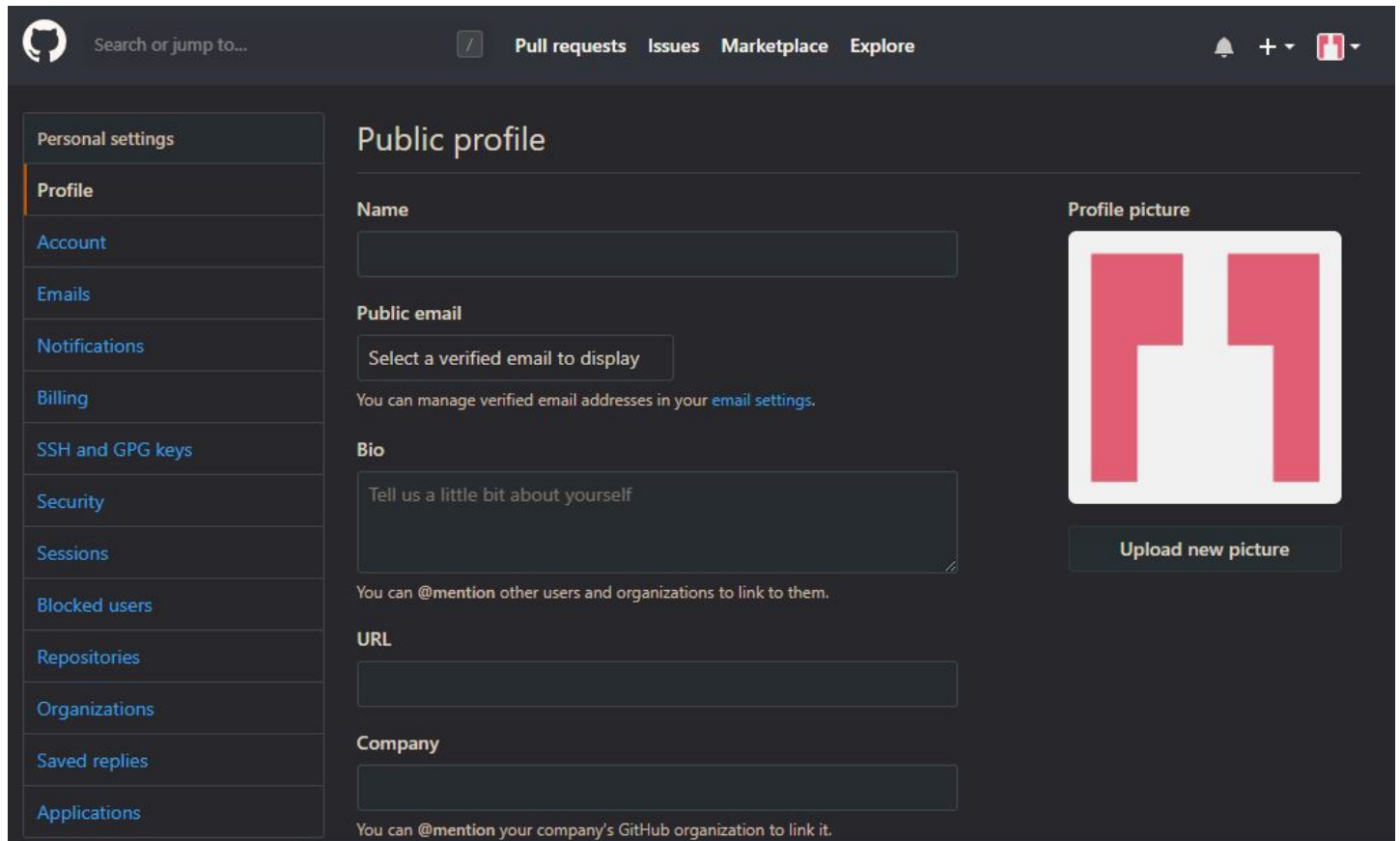
The screenshot shows the GitHub interface for the repository `chat-roux / mcommerce-config-repository`. The user is signed in as `chat-roux`. The repository page includes a navigation bar with links to Pull requests, Issues, Marketplace, and Explore. Below the repository name, there are tabs for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. The main content area displays a 'Quick setup' section with instructions for setting up a new repository on the command line. The instructions include a list of commands to create a new repository, add a README, and push it to GitHub. The commands are:

```
echo "# mcommerce-config-repository" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:chat-roux/mcommerce-config-repository.git
git push -u origin master
```

The page also includes a sidebar on the right with links to the user's profile, repositories, stars, gists, help, settings, and sign out.

Dans « github.com » :

- Dans le menu latéral gauche
- Cliquer sur l'entrée : « SSH and GPG keys »



Search or jump to...

Pull requests Issues Marketplace Explore

Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security

Sessions

Blocked users

Repositories

Organizations

Saved replies

Applications

Public profile

Name

Public email

Select a verified email to display

You can manage verified email addresses in your [email settings](#).

Bio

Tell us a little bit about yourself

You can @mention other users and organizations to link to them.

URL

Company

You can @mention your company's GitHub organization to link it.

Profile picture

Upload new picture

Dans « [github.com](#) » :

- Dans l'écran « [SSH and GPG keys](#) »
- Cliquer sur l'entrée : « [New SSH key](#) »

The screenshot shows the GitHub user interface. On the left is a sidebar with navigation links: Personal settings, Profile, Account, Emails, Notifications, Billing, SSH and GPG keys (highlighted), Security, Sessions, Blocked users, Repositories, Organizations, Saved replies, and Applications. The main content area is titled 'SSH keys' with a 'New SSH key' button. Below the title is a message: 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' A single SSH key is listed for the email 'tcharou.dalgalian@afpa.fr' with a green key icon, a fingerprint, and a 'Delete' button. Below this is a link to a guide on generating SSH keys. The 'GPG keys' section follows, with a 'New GPG key' button and a message stating there are no GPG keys associated with the account, along with a link to learn how to generate one.

Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security

Sessions

Blocked users

Repositories

Organizations


Saved replies

Applications

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 **tcharou.dalgalian@afpa.fr**
Fingerprint: ce:84:5c:0c:7f:59:ed:aa:82:41:0f:f6:9d:00:39:82
Added on 27 Sep 2018
Last used within the last week — Read/write
[Delete](#)

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

GPG keys

New GPG key

There are no GPG keys associated with your account.

Learn how to [generate a GPG key and add it to your account](#).

Dans « github.com » :

- Dans l'écran « SSH key / Add new »
- Coller votre clé SSH publique (générée précédemment).
- **Bouton** : « Add SSH key »

Search or jump to...

Pull requests Issues Marketplace Explore

Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security

Sessions

Blocked users

Repositories

Organizations

Saved replies

Applications

SSH keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

(05.05.) Demander la clé SSH publique du serveur « github.com »

- Démarrer une invite de commande « **Git Bash** »
- Lancer la **commande ci-dessous**.

```
#####  
# (02.)RECUPERER LA CLE SSH PUBLIQUE DU SERVEUR :  
# ->HOTE : 'git@github.com'  
#####  
ssh -T git@github.com
```

- Résultat : La **clé SSH publique du serveur** est enregistrée dans le **fichier spécifié ci-dessous**.
- Vérification : **Repérer cette clé** dans ce fichier (ouvrir le fichier avec un éditeur de texte).

Répertoire	C:\Users\chat_roux\.ssh
Fichier	known_hosts

Remarques :

- **Vous pouvez ouvrir ce fichier** avec un éditeur de texte.
- **Vous pouvez examiner le contenu de ce fichier** avec un éditeur de texte.
- **Vous aurez à le faire** au chapitre (07) !!!

(Editeur de texte : Notepad++)

(06.) Le dépôt local

Ce chapitre est constitué des sous-chapitres listés ci-dessous :

(06.01.) Créer un répertoire local

(06.02.) Initialiser le répertoire local

(06.03.) Déposer un fichier de configuration dans le dépôt local

(06.04.) Valider les modifications sur ce fichier

(06.05.) Entériner la liste des modifications validées du dépôt local

(06.06.) Faire pointer le dépôt local vers le dépôt distant

(06.07.) Sauvegarder le contenu du dépôt local dans le dépôt distant

(06.01.) Créer un répertoire local

Emplacement : Le répertoire « 05-MicroService-TP03 » (qui contient ce document)
Nom du répertoire : « mcommerce-03-config-repo »

(06.02.) Initialiser le répertoire local

- Démarrer une invite de commande « Git Bash »
- Faire pointer l'invite de commande sur le répertoire local créé précédemment
- Lancer la commande ci-dessous :

```
#####
# (01.)INITIALISER UN REPOSITORY LOCAL :
# -->REPOSITORY A INITIALISER : LE REPERTOIRE COURANT.
#####
git init
```

- **Résultat :** Un sous-répertoire « .git » a été créé à la racine du répertoire courant.

(06.03.) Déposer un fichier de configuration dans le dépôt local

- **Fichier à créer** : « `microservice-produit.yml` »
- **Contenu du fichier** : Celui-ci est détaillé dans les rubriques ci-dessous :

(06.03.01.) Rubrique « Server-Port-Configuration »

(06.03.02.) Rubrique « Spring-Configuration »

(06.03.03.) Rubrique « Actuator-Configuration »

(06.03.04.) Rubrique « Logging-Configuration »

(06.03.01.) Rubrique « Server-Port-Configuration »

Elle est fournie dans le bloc ci-dessous. Veuillez le récupérer.

```
#####  
# ---- (01.) SERVER-PORT-CONFIGURATION ----  
#####  
#server:  
#  port: 9010
```

Remarques :

- **Le bloc de données de cette rubrique** : il est **entièrement est désactivé** (vous remarquerez le symbole « # » au début de chaque ligne !).
- **La raison de cette désactivation** : Ce bloc de données a une **utilité uniquement pédagogique**.
- **La propriété « server.port »** : Elle sera **effectivement alimentée avec la valeur « 9010 »**, mais **pas dans ce fichier de propriétés** « `microservice-produit.yml` ».

(06.03.02.) Rubrique « Spring-Configuration »

Elle est fournie dans le **bloc ci-dessous**. Veuillez le récupérer.

```
#####
# ---- (02.) SPRING-CONFIGURATION ----
#####
spring:
# ---- (02.01.) SPRING-DATASOURCE-CONFIGURATION ----
datasource:
  url: "jdbc:mariadb://localhost:3306/produitdb?createDatabaseIfNotExist=true"
  username: "root"
  password: "tcharou"
  #driverClassName: "com.mysql.cj.jdbc.Driver"
  driverClassName: "org.mariadb.jdbc.Driver"
  sql-script-encoding: UTF-8
  initialization-mode: always
  data: "classpath:data.sql"
# ---- (02.02.) SPRING-JPA-CONFIGURATION ----
jpa:
  show-sql: true
  hibernate:
    ddl-auto: "update"
  properties:
    hibernate:
      dialect: "org.hibernate.dialect.MariaDB53Dialect"
```

(06.03.03.) Rubrique « Actuator-Configuration »

Elle est fournie dans le **bloc ci-dessous**. Veuillez le récupérer.

```
#####
# ---- (03.) ACTUATOR-CONFIGURATION ----
#####
management:
  endpoints:
    web:
      exposure:
        include: "*"

```

(06.03.04.) Rubrique « Logging-Configuration »

Elle est fournie dans le **bloc ci-dessous**. Veuillez le récupérer.

```
#####
# ---- (06.) LOGGING-CONFIGURATION ----
#####
logging :
  config: "classpath:log4j2-spring.xml"
```

(06.04.) Valider les modifications sur ce fichier

- Démarrer une invite de commande « **Git Bash** »
- Faire pointer l'invite de commande **sur le répertoire local créé précédemment**
- Lancer la **commande ci-dessous** :

```
#####
# (03.)AJOUTER DES ELEMENTS DANS LE PROCHAIN COMMIT :
#   -->ELEMENTS A AJOUTER : UN FICHIER / REPERTOIRE DU REPERTOIRE COURANT.
#
#   -->SYNTAXE : git add [Nom du fichier]
#####
git add [Nom du fichier]
```

(06.05.) Entériner les modifications validées du dépôt local

- Démarrer une invite de commande « **Git Bash** »
- Faire pointer l'invite de commande **sur le répertoire local créé précédemment**
- Lancer la **commande ci-dessous** :

```
#####
# (04.A.)LANCER LE PROCHAIN COMMIT VERS UN REPOSITORY LOCAL :
#   -->REPOSITORY LOCAL DESTINATAIRE : LE REPOSITORY LOCAL COURANT.
#
#   -->SYNTAXE : git commit -m "[Description du contenu du commit]"
#####
git commit -m "[Description du contenu du commit]"
```

- Vérification : Lancer la **commande de consultation** ci-dessous :

```
#####
# (04.B.)VERIFIER L'ETAT DU REPOSITORY LOCAL :
#   -->REPOSITORY LOCAL : LE REPOSITORY LOCAL COURANT.
#
#   -->SYNTAXE : git status
#####
git status
```

(06.06.) Faire pointer le dépôt local vers le dépôt distant

- Démarrer une invite de commande « **Git Bash** »
- Faire pointer l'invite de commande **sur le répertoire local créé précédemment**
- Lancer la **commande ci-dessous** :

```
#####
# (05.A.)ENREGISTRER / DESENGISTRER UN REPOSITORY DISTANT COMME "CIBLE" D'UN REPOSITORY LOCAL :
# -->REPOSITORY DISTANT A ENREGISTRER / DESENGISTRER : LE REPOSITORY DISTANT FOURNI.
# -->REPOSITORY LOCAL DANS LEQUEL ENREGISTRER / /DESENGISTRER : LE REPOSITORY LOCAL COURANT.
#
# -->SYNTAXE : git remote add [Nom du repository distant] [URL du repository distant]
#####
#git remote add distantHttps https://github.com/chat-roux/mcommerce-03-config-repo.git
git remote add distantSsh git@github.com:chat-roux/mcommerce-03-config-repo.git
```

- **Vérification** : Lancer la **commande de consultation** ci-dessous :

```
#####
# (05.B.)AFFICHER LA LISTE DES REPOSITORIES DISTANTS :
# -->REPOSITORIES DISTANTS : LES REPOSITORIES DISTANTS POINTES PAR LE REPOSITORY LOCAL COURANT.
#
# -->SYNTAXE : git remote -v
#####
git remote -v
```

(06.07.) Sauvegarder le contenu du dépôt local dans le dépôt distant

- Démarrer une invite de commande « **Git Bash** »
- Faire pointer l'invite de commande **sur le répertoire local créé précédemment**
- Lancer les **commandes ci-dessous** :

```
#####
# (07.A.)AFFICHER LA LISTE DES BRANCHES DANS LE REPOSITORY LOCAL :
#####
git branch
```

```
#####
# (07.B.)TRANSFERER LE CONTENU DU REPOSITORY LOCAL DANS LE REPOSITORY DISTANT :
# -->ORIGINE DU TRANSFERT : LE REPERTOIRE COURANT (REPOSITORY LOCAL COURANT).
# -->DESTINATION DU TRANSFERT : LA "CIBLE" PRE-ENREGISTREE (REPOSITORY DISTANT).
# -->TYPE DE TRANSFERT : TRANSFERT AVEC ECRASEMENT.
#
# -->SYNTAXE : git push -u [Nom du repository distant] [Nom de la branche distante]
#####
git push -u distantSsh master
```

(07.) Le serveur de configuration

Ce chapitre est constitué des **sous-chapitres listés ci-dessous** :

(07.01.) L'arborescence des répertoires du projet.

(07.02.) Le fichier de configuration « pom.xml » du projet

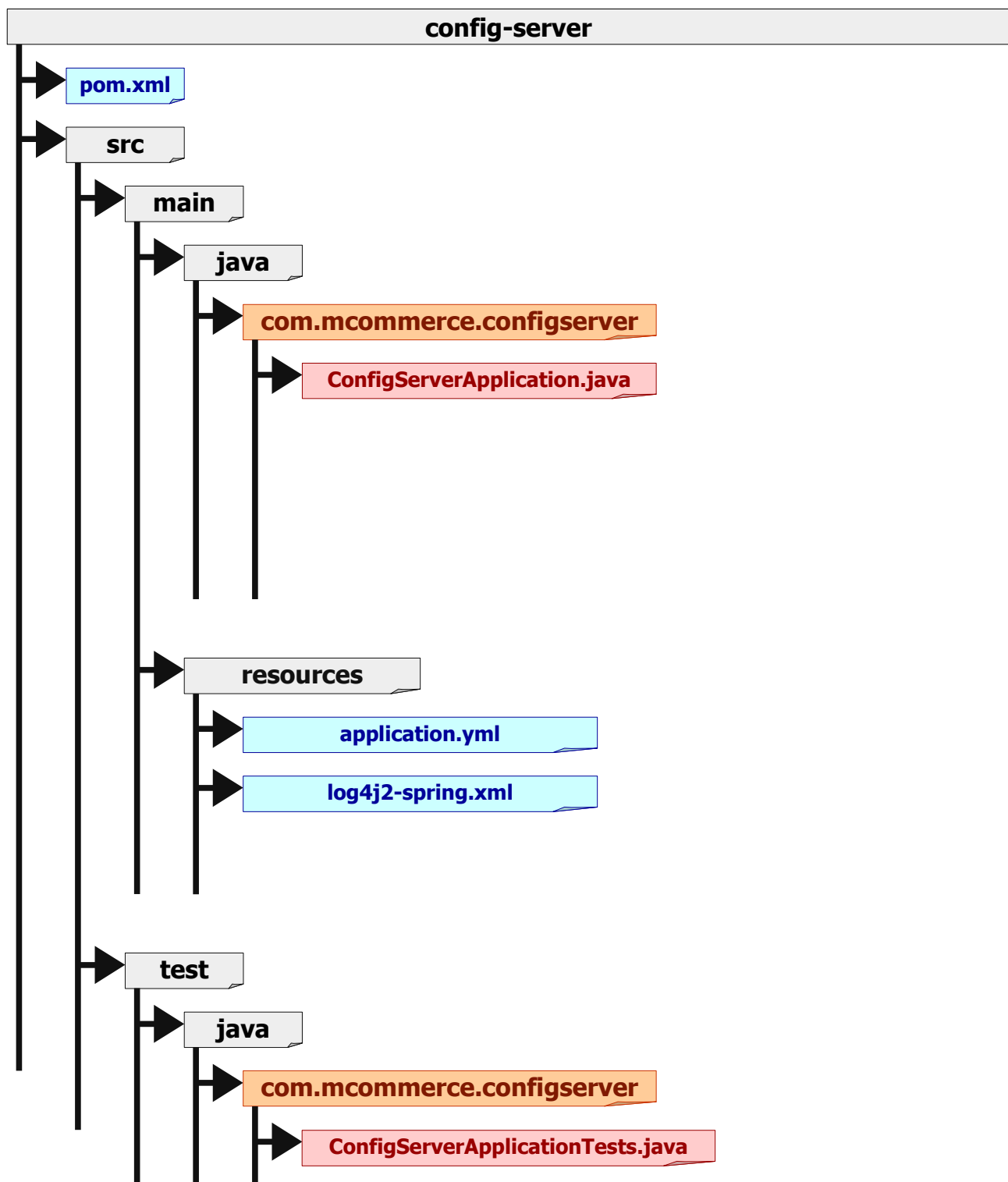
(07.03.) Le fichier de propriétés « application.yml » de l'application

(07.04.) Le fichier de configuration des loggers

(07.05.) La classe principale de l'application

(07.01.) L'arborescence des répertoires du projet

L'arborescence des répertoires du projet sera celle d'une **application web avec une architecture N-Tiers**.
Veillez créer cette arborescence conformément au schéma ci-dessous, svp.



(07.02.) Le fichier de configuration « pom.xml » du projet

Lorsque l'on demande à MAVEN de compiler le projet, MAVEN parcourt ce fichier et configure le procédé de compilation en fonction du contenu de ce fichier.

Les informations permettant de trouver ce fichier sont fournies ci-dessous :

Répertoire	La racine du projet
Fichier	pom.xml

Le contenu de ce fichier est détaillé dans les rubriques ci-dessous :

(07.02.01.) La rubrique : « Informations du projet parent »

(07.02.02.) La rubrique : « Informations du projet »

(07.02.03.) La rubrique : « Propriétés du projet »

(07.02.04.) La rubrique : « Dépendances »

(07.02.05.) La rubrique : « Management des dépendances »

(07.02.06.) La rubrique : « Construction de l'exécutable »

(07.02.01.) La rubrique : « Informations du projet parent »

Elle est fournie dans le bloc de données ci-dessous. Veuillez la récupérer, svp :

```
<!-- ===== -->
<!-- (01.)Informations sur le parent du projet -->
<!-- ===== -->
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.0.4.RELEASE</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
```

(07.02.02.) La rubrique : « Informations du projet »

Elle est fournie dans le bloc de données ci-dessous. Veuillez la récupérer, svp :

```
<!-- ===== -->
<!-- (02.)Informations sur le projet -->
<!-- ===== -->
<!-- ===== (02.01.)Informations d'identification du projet ===== -->
<groupId>com.mcommerce</groupId>
<artifactId>config-server</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<!-- ===== (02.02.)Informations generales ===== -->
<name>config-server</name>
<description>Demo project for Spring Boot</description>
```

(07.02.03.) La rubrique : « Propriétés du projet »

Elle est fournie dans le bloc de données ci-dessous. Veuillez la récupérer, svp :

```
<!-- ===== -->
<!-- (03.)Proprietes du projet -->
<!-- ===== -->
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <java.version>1.8</java.version>
  <spring-cloud.version>Finchley.SR1</spring-cloud.version>
</properties>
```

(07.02.04.) La rubrique : « Dépendances »

Elle est fournie dans le bloc de données ci-dessous. Veuillez la récupérer, svp :

```
<dependencies>
<!-- ===== -->
<!-- ===== (04.01.) Spring-BOOT ===== -->
<!-- ===== -->
<!-- A RENSEIGNER !!! -->

<!-- ===== -->
<!-- ===== (04.02.) Spring-CLOUD ===== -->
<!-- ===== -->
<!-- A RENSEIGNER !!! -->

</dependencies>
```

Le bloc de données ci-dessus est subdivisé en 2 sous-rubriques détaillées ci-dessous :

- [Les «Dépendances Spring-Boot »](#)
- [Les «Dépendances Spring-Cloud »](#)

■ Les «Dépendances Spring-Boot »

```
<!-- ===== -->
<!-- ===== (04.01.) Spring-BOOT ===== -->
<!-- ===== -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter</artifactId>
  <exclusions>
    <exclusion>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-logging</artifactId>
    </exclusion>
  </exclusions>
</dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-log4j2</artifactId>
</dependency>
```

■ Les «Dépendances Spring-Cloud »

```
<!-- ===== -->
<!-- ===== (04.02.) Spring-CLOUD ===== -->
<!-- ===== -->
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-config-server</artifactId>
</dependency>
```


(07.02.05.) La rubrique : « Management des dépendances »

Elle est fournie dans le bloc de données ci-dessous. Veuillez la récupérer, svp :

```
<!-- ===== -->
<!-- (05.)Management des dependances du projet ===== -->
<!-- ===== -->
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>${spring-cloud.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

(07.02.06.) La rubrique : « Construction de l'exécutable »

Elle est fournie dans le bloc de données ci-dessous. Veuillez la récupérer, svp :

```
<!-- ===== -->
<!-- (06.)Construction de l'executable du projet -->
<!-- ===== -->
<build>
  <!-- ===== -->
  <!-- (06.01.)Nom du fichier executable -->
  <!-- ===== -->
  <!-- A RENSEIGNER !!! -->

  <!-- ===== -->
  <!-- (06.02.)Plugins -->
  <!-- ===== -->
  <!-- A RENSEIGNER !!! -->

</build>
```

Le bloc de données ci-dessus est subdivisé en 2 sous-rubriques détaillées ci-dessous :

- Le nom du fichier exécutable
- Les «Plugins »

■ Le nom du fichier exécutable

```
<!-- ===== -->
<!-- (06.01.)Nom du fichier executable -->
<!-- ===== -->
<finalName>configServer</finalName>
```

■ Les «Plugins »

```
<!-- ===== -->
<!-- (06.02.)Plugins -->
<!-- ===== -->
<plugins>
  <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
  </plugin>
</plugins>
```

(07.03.) Le fichier de propriétés « application.yml » de l'application

Le fichier de propriétés de l'application : Il s'appelle « application.yml », et est subdivisé en rubriques. Celle-ci sont détaillées ci-dessous.

(07.03.01.) La rubrique « server-port-configuration »

(07.03.02.) La rubrique « spring-configuration »

(07.03.03.) La rubrique « logging-configuration »

(07.03.01.) La rubrique « server-port-configuration »

Elle est détaillée ci-dessous. Veuillez la récupérer, svp.

```
#####  
# ---- (01.) SERVER-PORT-CONFIGURATION ----  
#####  
server:  
  port: 9101
```

(07.03.02.) La rubrique « spring-configuration »

Elle est détaillée ci-dessous. Veuillez la récupérer, svp.

```
#####  
# ---- (02.) SPRING-CONFIGURATION ----  
#####  
spring:  
  # ---- (02.01.) SPRING-APPLICATION-CONFIGURATION ----  
  application:  
    name: "config-server"  
  # ---- (02.02.) SPRING-CLOUD-CONFIGURATION ----  
  cloud:  
    config:  
      server:  
        git:  
          #uri: "https://github.com/[USER]/[DEPOT_DISTANT].git"  
          uri: "git@github.com:[USER]/[DEPOT_DISTANT].git"  
          clone-on-start: true  
          force-pull: true  
          ignoreLocalSshSettings: true  
          strict-host-key-checking: true  
          hostKey: "[CLE SSH PUBLIQUE DE L'HÔTE DU DEPÔT DISTANT]"  
          hostKeyAlgorithm: "ssh-rsa"  
          passphrase: "[PHRASE DE PASSE DE VOTRE CLE SSH PRIVEE]"  
          privateKey: |  
            [VOTRE CLE SSH PRIVEE]
```

Remarque :

La rubrique ci-dessus déclare une liste de propriétés. Certaines d'entre elles sont liées à l'authentification sur l'hôte du dépôt distant. Elles sont détaillées dans le tableau ci-dessous :

Propriété	Valeur théorique	Description	Obtention	Emplacement / Valeur réelle
<code>spring.cloud.config.server.git.uri</code>	"git@github.com:[USER]/[DEPOT_DISTANT].git"	<u>[USER]</u> : Votre nom d'utilisateur sur « github.com » <u>[DEPOT_DISTANT]</u> : Nom du dépôt distant	Il a été <u>créé</u> (sous-chapitre (04.01.)). Il a été <u>créé</u> (sous-chapitre (04.03.)).	<u>Valeur</u> : mcommerce-03-config-repo
<code>spring.cloud.config.server.git.hostKey</code>	"[CLE SSH PUBLIQUE DE L'HOTE DU DEPOT DISTANT]"	La <u>clé SSH publique</u> de l' <u>hôte du dépôt distant</u>	Elle a été <u>demandée</u> (sous-chapitre 05.05.) puis <u>déposée</u> sur votre ordinateur.	<u>Emplacement</u> : ■ <u>Ordinateur</u> : Votre ordinateur ■ <u>Répertoire</u> : C:\Utilisateurs\[VOTRE NOM D'UTILISATEUR\.ssh] ■ <u>Fichier</u> : known_hosts
<code>spring.cloud.config.server.git.hostKeyAlgorithm</code>	"ssh-rsa"	Nom de l' <u>algorithme de cryptage</u>	Valeur <u>fournie dans ce tableau</u>	Valeur <u>non modifiable</u>
<code>spring.cloud.config.server.git.passphrase</code>	"[PHRASE DE PASSE DE VOTRE CLE SSH PRIVEE]"	La <u>phrase de passe de votre clé SSH privée</u>	Elle a été <u>créée</u> (sous-chapitre (05.01)), puis <u>sauvegardée</u> sur votre ordinateur.	<u>Emplacement</u> : Dans un bloc-note.
<code>spring.cloud.config.server.git.privateKey</code>	"[VOTRE CLE SSH PRIVEE]"	<u>Votre clé SSH privée</u>	Elle a été <u>générée</u> (sous-chapitre (05.01)), puis <u>déposée</u> sur votre ordinateur.	<u>Emplacement</u> : ■ <u>Ordinateur</u> : Votre ordinateur ■ <u>Répertoire</u> : C:\Utilisateurs\[VOTRE NOM D'UTILISATEUR\.ssh] ■ <u>Fichier</u> : id_rsa

(07.03.03.) La rubrique « logging-configuration »

Elle est détaillée ci-dessous. Veuillez la récupérer, svp.

```
#####
# ---- (04.) LOGGING-CONFIGURATION ----
#####
logging :
  config: "classpath:log4j2-spring.xml"
```

(07.04.) Le fichier de configuration des loggers

Les informations permettant de trouver ce fichier sont fournies ci-dessous :

Répertoire	src\main\resources
Fichier	log4j2-spring.xml

Le contenu de ce fichier est détaillé dans les rubriques ci-dessous :

(07.04.01.) La rubrique : « Properties »

(07.04.02.) La rubrique : « Appenders »

(07.04.03.) La rubrique : « Loggers »

(07.04.01.) La rubrique : « Properties »

Cette rubrique est constituée du bloc ci-dessous. Veuillez le récupérer , svp.

```
<Properties>
  <Property name="log-path">./logExecute</Property>
</Properties>
```

(07.04.02.) La rubrique : « Appenders »

Cette rubrique est constituée du [bloc ci-dessous](#). [Veuillez le récupérer](#), svp.

```
<!-- ===== -->
<!-- (01.) Appenders : -->
<!--   - (01.A.) Chaque Appender pointe sur le fichier qui lui est associe -->
<!--   - (01.B.) Chaque Appender ecrit dans ce fichier -->
<!-- ===== -->
<Appenders>
  <Console name="Console-Appender" target="SYSTEM_OUT">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </Console>

  <File name="ConfigServer-File-Appender" fileName="${log-path}/ConfigServer.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="Spring-File-Appender" fileName="${log-path}/ConfigServer-spring.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="SpringCloud-File-Appender" fileName="${log-path}/ConfigServer-springCloud.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="SpringBoot-File-Appender" fileName="${log-path}/ConfigServer-springBoot.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>

  <File name="SpringContext-File-Appender" fileName="${log-path}/ConfigServer-springContext.log">
    <PatternLayout>
      <pattern>[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n</pattern>
    </PatternLayout>
  </File>
</Appenders>
```

(07.04.03.) La rubrique : « Loggers »

Cette rubrique est constituée du **bloc ci-dessous**. **Veillez le récupérer**, svp.

```
<!-- ===== -->
<!-- (02.) Loggers : -->
<!-- - (02.A.) Chaque Logger scanne le package qui lui est associe -->
<!-- - (02.B.) Chaque Logger récupère les logs qu'il trouve dans les classes -->
<!-- ===== -->
<Loggers>
  <Root>
    <AppenderRef ref="Console-Appender" level="all" />
  </Root>

  <Logger name="com.mcommerce.configserver" level="all" additivity="true">
    <AppenderRef ref="ConfigServer-File-Appender" level="all" />
  </Logger>

  <Logger name="org.springframework" level="info" additivity="true">
    <AppenderRef ref="Spring-File-Appender" level="all" />
  </Logger>

  <Logger name="org.springframework.cloud" level="info" additivity="true">
    <AppenderRef ref="SpringCloud-File-Appender" level="all" />
  </Logger>

  <Logger name="org.springframework.boot" level="info" additivity="true">
    <AppenderRef ref="SpringBoot-File-Appender" level="all" />
  </Logger>

  <Logger name="org.springframework.context" level="info" additivity="true">
    <AppenderRef ref="SpringContext-File-Appender" level="all" />
  </Logger>
</Loggers>
```

(07.05.) La classe principale de l'application

Les informations permettant de trouver cette classe sont fournies ci-dessous :

Répertoire	src\main\java
Package	com.mcommerce.configserver
Classe	ConfigServerApplication

Le contenu de ce fichier est détaillé dans les rubriques ci-dessous :

(07.05.01.) Les annotations sur la classe

(07.05.02.) L'attribut « Logger »

(07.05.03.) La méthode « main »

(07.05.01.) Les annotations sur la classe

Les annotations à poser sur la classe sont indiquées ci-dessous. Veuillez les récupérer, svp.

```
@EnableConfigServer
@SpringBootApplication
public class ConfigServerApplication {...}
```

(07.05.02.) L'attribut « Logger »

L'attribut « Logger » est déclaré dans le bloc de code ci-dessous. Veuillez le récupérer, svp.

```
/**
 * <b>OBJET QUI POSSEDE LES FONCTIONNALITES SUIVANTES : </b> <br/>
 * <br/>
 * Les fonctionnalites d'ecriture de messages de log dans la console.
 */
private static final Logger LOGGER = LoggerFactory.getLogger(ConfigServerApplication.class);
```

(07.05.03.) La méthode « main »

La méthode « main » est implémentée dans le [bloc de code ci-dessous](#). [Veuillez le récupérer](#), svp.

```
/**
 * <b>METHODE D'ENTREE DE L'APPLICATION</b><br/>
 *
 * @param args
 */
public static void main(String[] args) {

    LOGGER.info("CLASS : ConfigServerApplication -- METHOD : main - BEGIN");
    SpringApplication.run(ConfigServerApplication.class, args);
    LOGGER.info("CLASS : ConfigServerApplication -- METHOD : main -- END");
}
```


(08.) Le déploiement de la plate-forme micro-services

Pour le déploiement de la plate-forme micro-services : Veuillez suivre les étapes ci-dessous.

(08.01.) Configurer le serveur de BDD « MariaDB »

(08.02.) Démarrer le serveur de BDD « MariaDB »

(08.03.) Compiler le projet « config-server »

(08.04.) Exécuter le projet « config-server »

(08.05.) Compiler le projet « microservice-produit »

(08.06.) Exécuter le projet « microservice-produit »

(08.01.) Configurer le serveur de BDD « MariaDB »

Pour **configurer le serveur de BDD**, veuillez suivre les étapes ci-dessous :

(08.01.01.) Créer le fichier de configuration « my.ini »

(08.01.02.) La rubrique « Default-Time-Zone-Configuration »

(08.01.03.) La rubrique « Base-Directory-Configuration »

(08.01.04.) La rubrique « Data-Directory-Configuration »

(08.01.01.) Créer le fichier de configuration « my.ini »

Ce fichier de configuration est défini ci-dessous :

Répertoire	Répertoire d'installation du serveur de BDD « MariaDB »
Fichier	my.ini

(08.01.02.) La rubrique « Default-Time-Zone-Configuration »

Cette rubrique est fournie ci-dessous. Veuillez la récupérer, svp :

```
#####
# (01.)DEFAULT-TIME-ZONE-CONFIGURATION
#
# set default-time-zone of your server
#####
[mysqld]
default-time-zone='+02:00'
```

(08.01.03.) La rubrique « Base-Directory-Configuration »

Cette rubrique est fournie ci-dessous. Veuillez la récupérer, svp :

```
#####
# (02.)BASE-DIRECTORY-CONFIGURATION
#
# set basedir to your installation path
#####
basedir=C:\\software\\mariadb
```

(08.01.04.) La rubrique « Data-Directory-Configuration »

Ce fichier contient les rubriques ci-dessous :

```
#####  
# (03.)DATA-DIRECTORY-CONFIGURATION  
#  
# set datadir to the location of your data directory  
#####  
datadir=C:\\software\\mariadb\\data
```

(08.02.) Démarrer le serveur de BDD

« MariaDB »

Le démarrage du serveur de BDD « MariaDB » est décrit dans les paragraphes ci-dessous :

(08.02.01.) Le fichier « batch » de lancement du serveur de BDD

(08.02.02.) La commande de lancement du serveur de BDD

(08.02.01.) Le fichier « batch » de lancement du serveur de BDD

Ce fichier « batch » de lancement du serveur de BDD : Il est fourni à l'emplacement défini ci-dessous. Veuillez l'ouvrir , svp. (Ne le lancez pas !)

Répertoire	Reference\03-DEVLOPM\09-DeveloppementBDD
Sous-répertoire	09-01-InstallationConfiguration\02-ServerStartStop
Fichier	01-mysqlServerStart.bat

(08.02.02.) La commande de lancement du serveur de BDD

Cette commande de lancement du serveur de BDD : Elle est fournie à l'emplacement défini ci-dessous. Veuillez la lancer, svp.

```

REM *****
REM * (01.) LANCER LE PROCESSUS SUIVANT :
REM *      ->LE PROCESSUS DE DEMARRAGE DU SERVEUR.
REM *****
REM *      FICHIER DE CONFIGURATION DU SERVEUR :
REM *      ->NOM : 'my.ini'
REM *      ->EMPLACEMENT POUR MySQL : 'C:\ProgramData\MySQL\MySQL Server 5.7'
REM *      ->EMPLACEMENT POUR MariaDB : 'C:\software\mariadb'
REM *****
mysql.exe --defaults-file="C:\ProgramData\MySQL\MySQL Server 5.7\my.ini"
mysql.exe --defaults-file="C:\software\mariadb\my.ini"

```

(08.03.) Compiler le projet « config-server »

Pour compiler le projet « config-server », placez-vous à la racine du projet, puis suivez les étapes suivantes :

(08.03.01.) Créer un sous-répertoire « logCompile ».

(08.03.02.) Lancer la commande de compilation

(08.03.03.) Vérifier le résultat de la compilation

(08.03.01.) Créer un sous-répertoire « logCompile ».

Le sous-répertoire à créer est défini ci-dessous :

Répertoire	Répertoire racine du projet « config-server »
Sous-répertoire	.\logCompile

(08.03.02.) Lancer la commande de compilation

Le fichier « batch » de compilation : Il est fourni à l'emplacement défini ci-dessous. Veuillez l'ouvrir, svp. (Ne le lancez pas !)

Répertoire	Répertoire racine du projet « config-server »
Sous-répertoire	.\maven\02-ProjectCompile
Fichier	mvnProjectCompile--mcommerce-03-ConfigServer.bat

La commande de compilation : Elle est fournie ci-dessous. Veuillez la lancer, svp.

```
REM *****
REM * (01.)EFFECTUER LES OPERATIONS SUIVANTES :
REM *
REM *      ->NETTOYER LE PROJET
REM *      ->COMPILER LE PROJET (GENERER LES CLASSES EN BYTE-CODE) .
REM *      ->GENERER LE FICHIER EXECUTABLE DU PROJET.
REM *      ->COPIER LE FICHIER EXECUTABLE DANS LE DEPOT MAVEN LOCAL.
REM *****
mvn clean install ^
-e -X ^
> .\logCompile\compile--mcommerce-03-ConfigServer.log
```

(08.03.03.) Vérifier le résultat de la compilation

Pour **vérifier le résultat de la compilation** : Veuillez ouvrir le **fichier de log de compilation**, svp. Celui-ci se trouve à l'emplacement ci-dessous :

Répertoire	Répertoire racine du projet « config-server »
Sous-répertoire	logCompile
Fichier	compile--mcommerce-03-ConfigServer.log

(08.04.) Exécuter le projet « config-server »

Pour exécuter le projet « config-server », placez-vous à la racine du projet, puis suivez les étapes suivantes :

(08.04.01.) Créer un sous-répertoire « logExecute ».

(08.04.02.) Lancer la commande d'exécution

(08.04.03.) Vérifier le résultat de l'exécution

(08.04.01.) Créer un sous-répertoire « logExecute ».

Le sous-répertoire à créer est défini ci-dessous :

Répertoire	Répertoire racine du projet « config-server »
Sous-répertoire	.\logExecute

(08.04.02.) Lancer la commande d'exécution

Le fichier « batch » d'exécution : Il est fourni à l'emplacement défini ci-dessous. Veuillez l'ouvrir, svp. (Ne le lancez pas !)

Répertoire	Répertoire racine du projet « config-server »
Sous-répertoire	.\maven\03-ProjectExecute
Fichier	javaProjectExecute--mcommerce-03-ConfigServer.bat

La commande d'exécution : Elle est fournie ci-dessous. Veuillez la lancer, svp.

```
REM *****
REM * (01.)EFFECTUER LES OPERATIONS SUIVANTES :
REM *
REM *      ->LANCER LE FICHIER EXECUTABLE DU PROJET.
REM *****
java -verbose ^
-jar ^
./target/configServer.jar ^
> .\logExecute\execute--mcommerce-03-ConfigServer.log
```

(08.04.03.) Vérifier le résultat de l'exécution

Pour **vérifier le résultat de l'exécution** : Veuillez ouvrir le **fichier de log d'exécution**, svp. Celui-ci se trouve à l'emplacement ci-dessous :

Répertoire	Répertoire racine du projet « config-server »
Sous-répertoire	logExecute
Fichier	execute--mcommerce-03-ConfigServer.log

(08.05.) Compiler le projet « microservice-produit »

Pour **compiler le projet « microservice-produit »**, placez-vous **à la racine du projet**, puis suivez les étapes suivantes :

(08.05.01.) Créer un sous-répertoire « logCompile ».

(08.05.02.) Lancer la commande de compilation

(08.05.03.) Vérifier le résultat de la compilation

(08.05.01.) Créer un sous-répertoire « logCompile ».

Le sous-répertoire à créer est défini ci-dessous :

Répertoire	Répertoire racine du projet « microservice-produit »
Sous-répertoire	.\logCompile

(08.05.02.) Lancer la commande de compilation

Le fichier « batch » de compilation : Il est fourni à l'emplacement défini ci-dessous. Veuillez l'ouvrir, svp. (Ne le lancez pas !)

Répertoire	Répertoire racine du projet « microservice-produit »
Sous-répertoire	.\maven\02-ProjectCompile
Fichier	mvnProjectCompile--mcommerce-03-Produit.bat

La commande de compilation : Elle est fournie ci-dessous. Veuillez la lancer, svp.

```
REM *****
REM * (01.)EFFECTUER LES OPERATIONS SUIVANTES :
REM *
REM *      ->NETTOYER LE PROJET
REM *      ->COMPILER LE PROJET (GENERER LES CLASSES EN BYTE-CODE) .
REM *      ->GENERER LE FICHIER EXECUTABLE DU PROJET.
REM *      ->COPIER LE FICHIER EXECUTABLE DANS LE DEPOT MAVEN LOCAL.
REM *****
mvn clean install ^
-e -X ^
> .\logCompile\compile--mcommerce-03-Produit.log
```

(08.05.03.) Vérifier le résultat de la compilation

Pour **vérifier le résultat de la compilation** : Veuillez ouvrir le **fichier de log de compilation**, svp. Celui-ci se trouve à l'emplacement ci-dessous :

Répertoire	Répertoire racine du projet « microservice-produit »
Sous-répertoire	logCompile
Fichier	compile--mcommerce-03-Produit.log

(08.06.) Exécuter le projet « microservice-produit »

Pour **exécuter le projet « config-server »**, placez-vous **à la racine du projet**, puis suivez les étapes suivantes :

(08.06.01.) Créer un sous-répertoire « logExecute ».

(08.06.02.) Lancer la commande d'exécution

(08.06.03.) Vérifier le résultat de l'exécution

(08.06.01.) Créer un sous-répertoire « logExecute ».

Le sous-répertoire à créer est défini ci-dessous :

Répertoire	Répertoire racine du projet « microservice-produit »
Sous-répertoire	.\logExecute

(08.06.02.) Lancer la commande d'exécution

Le fichier « batch » d'exécution : Il est fourni à l'emplacement défini ci-dessous. Veuillez l'ouvrir, svp. (Ne le lancez pas !)

Répertoire	Répertoire racine du projet « microservice-produit »
Sous-répertoire	.\maven\03-ProjectExecute
Fichier	javaProjectExecute--mcommerce-03-Produit.bat

La commande d'exécution : Elle est fournie ci-dessous. Veuillez la lancer, svp.

```
REM *****
REM * (01.)EFFECTUER LES OPERATIONS SUIVANTES :
REM *
REM *      ->LANCER LE FICHIER EXECUTABLE DU PROJET.
REM *****
java -verbose ^
-Dserver.port=9010 ^
-jar ^
./target/configServer.jar ^
> .\logExecute\execute--mcommerce-03-Produit--Port9010.log
```

(08.06.03.) Vérifier le résultat de l'exécution

Pour **vérifier le résultat de l'exécution** : Veuillez ouvrir le **fichier de log d'exécution**, svp. Celui-ci se trouve à l'emplacement ci-dessous :

Répertoire	Répertoire racine du projet « microservice-produit »
Sous-répertoire	logExecute
Fichier	execute--mcommerce-03-Produit--Port9010.log

(09.) Les tests sur la plate-forme micro-services

On effectuera des **tests sur chacun des composants applicatifs** de la plate-forme micro-services.

Ces tests sont **regroupés par composant applicatif**, et listés ci-dessous :

(09.01.) Les tests sur le « config-server »

(09.02.) Les tests sur le « microservice-produit »

(09.01.) Les tests sur le « config-server »

Les tests sur le « config-server » sont listés ci-dessous :

(09.01.01.) Interroger le « config-server » par défaut

(09.01.02.) Consulter les propriétés du « microservice-produit »

(09.01.01.) Interroger le microservice par défaut

Dans le client lourd « REST » (« Postman »), veuillez lancer la requête fournie ci-dessous.

```
#####  
# (01.)INTERROGER LE « config-server » PAR DEFAULT  
#####  
GET http://localhost:9101/default
```

Réponse attendue :

```
{  
  "name": "config-server",  
  "profiles": [  
    "default"  
  ],  
  "label": null,  
  "version": "5b26963e49f9de2d3dba9c8b8c8a430980a694d3",  
  "state": null,  
  "propertySources": []  
}
```

(09.01.02.) Consulter les propriétés du « microservice-produit »

Dans le client lourd « REST » (« Postman »), veuillez lancer les requêtes listées ci-dessous.

```
#####  
# (02.)CONSULTER LES PROPRIETES DU "micro-service-produit"  
#####  
GET http://localhost:9101/microservice-produit/default
```

Réponse attendue :

```
{  
  "name": "microservice-produit",  
  "profiles": [  
    "default"  
  ],  
  "label": null,  
  "version": "5b26963e49f9de2d3dba9c8b8c8a430980a694d3",  
  "state": null,  
  "propertySources": [  
    {  
      "name": "git@github.com:chat-roux/mcommerce-03-config-repo.git/microservice-produit.yml",  
      "source": {  
        "spring.datasource.url": "jdbc:mariadb://localhost:3306/produitdb?  
createDatabaseIfNotExist=true",  
        "spring.datasource.username": "root",  
        "spring.datasource.password": "tcharou",  
        "spring.datasource.driverClassName": "org.mariadb.jdbc.Driver",  
        "spring.datasource.sql-script-encoding": "UTF-8",  
        "spring.datasource.initialization-mode": "always",  
        "spring.datasource.data": "classpath:data.sql",  
        "spring.jpa.show-sql": true,  
        "spring.jpa.hibernate.ddl-auto": "update",  
        "spring.jpa.properties.hibernate.dialect": "org.hibernate.dialect.MariaDB53Dialect",  
        "logging.config": "classpath:log4j2-spring.xml",  
        "ma-config.produitNombreMax": 2  
      }  
    }  
  ]  
}
```

(09.02.) Les tests sur le « microservice-produit »

Les tests sur le « microservice-produit » sont listés ci-dessous :

(09.02.01.) Interroger le « microservice-produit » par défaut

(09.02.02.) Actualiser les propriétés du « microservice-produit »

(09.02.03.) Consulter la liste des produits

(09.02.01.) Interroger le « microservice-produit » par défaut

Dans le client lourd « REST » (« Postman »), veuillez lancer les requêtes listées ci-dessous.

```
#####  
# (01.)INTERROGER LE « microservice-produit » PAR DEFAULT  
#####  
GET http://localhost:9010/default
```

Réponse attendue :

```
{  
  "timestamp": "2018-10-11T09:57:08.466+0000",  
  "status": 404,  
  "error": "Not Found",  
  "message": "No message available",  
  "path": "/default"  
}
```

(09.02.02.) Actualiser les propriétés du « microservice-produit »

Dans le client lourd « REST » (« Postman »), veuillez lancer les requêtes listées ci-dessous.

```
#####  
# (02.)ACTUALISER LES PROPRIETES DU "microservice-produit"  
#####  
POST http://localhost:9010/actuator/refresh
```

Réponse attendue :

(09.02.03.) Consulter la liste des produits

Dans le client lourd « REST » (« Postman »), veuillez lancer les requêtes listées ci-dessous.

```
#####  
# (03.)CONSULTER LA LISTE DES PRODUITS  
#####  
GET http://localhost:9010/produit
```

Réponse attendue :