

Cours 6

Evolution et maintenance des services Web

Un cours de Yann Fornier



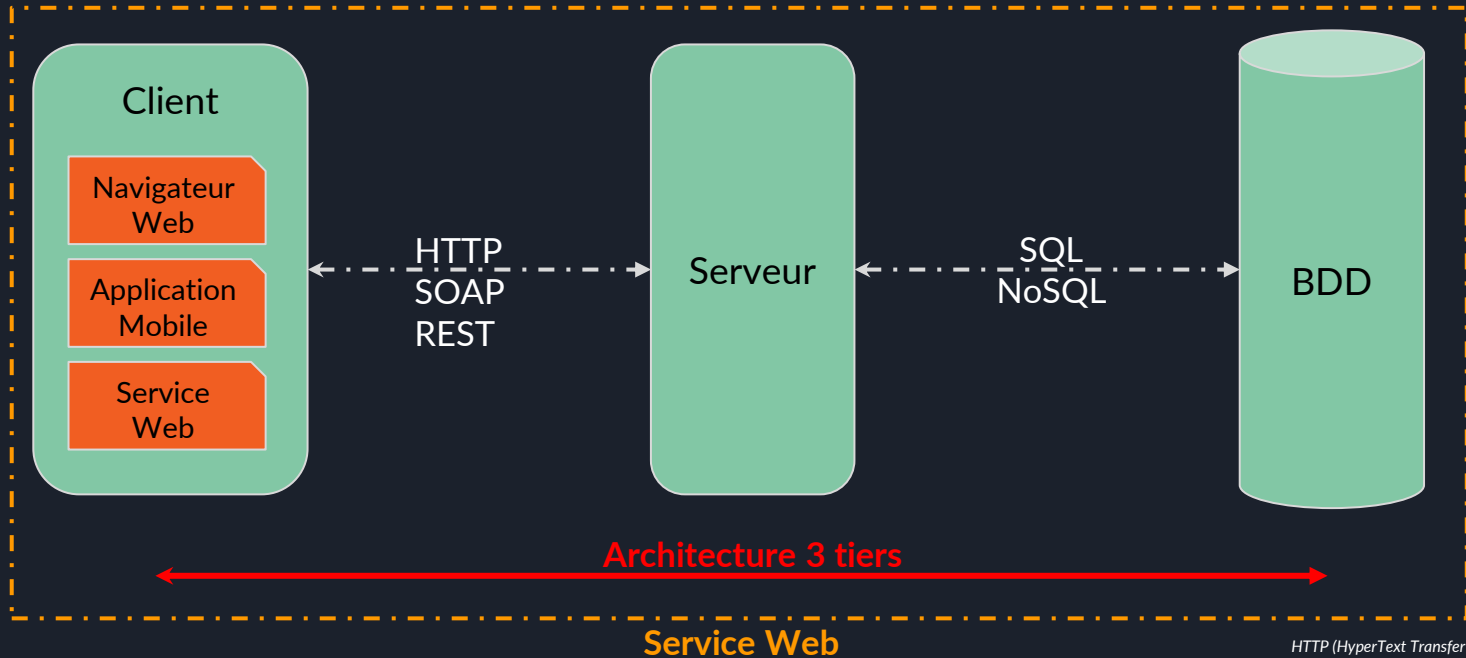
Evolution et maintenance des Services Web

Session 6 : Évolution et maintenance des services web

- Les différentes approches pour évoluer et maintenir une architecture web (versioning, backward compatibility, etc.)
- La gestion des dépendances et des bibliothèques utilisées par les services web
- La mise en place de processus de déploiement continu et de tests automatisés pour assurer la qualité des services web
- **Exercice** : évoluer et maintenir une architecture web en utilisant les différentes techniques vues en cours
- **QCM** : connaissances sur l'évolution et la maintenance des services web

Concepts de base

Architecture des services web



HTTP (HyperText Transfer Protocol)

SOAP (Simple Object Access Protocol)

REST (Representational State Transfer)



Protocoles et normes associés aux services web

Les **protocoles** et **normes** associés aux services web permettent de décrire les services, les messages échangés, les interfaces, et les opérations supportées.

WSDL
(Web Services
Description
Language)

XML
(Extensible
Markup Language)

JSON
(JavaScript Object
Notation)

Le WSDL (Web Services Description Language)

WSDL
(*Web Services
Description
Language*)

Langage de description de service qui permet de décrire les opérations du service, les types de données échangés, et les protocoles de communication utilisés.

WSDL Elements

A WSDL document describes a web service using these major elements:

**Abstract
Definition of
Service**

**Protocol and
physical
locations**

Types

Messages

Port Types

Bindings

Service ports

- Types
 - What data types will be transmitted
- Messages
 - What messages will be transmitted
- Port Types
 - What business operations (functions) will be supported
- Bindings
 - How will the messages be transmitted on the wire?
 - What message protocol (e.g. SOAP) specific details are there?
- Service ports
 - Where is the service located?

Le WSDL (Web Services Description Language)

WSDL (Web Services Description Language)

Langage de description de service qui permet de décrire les opérations du service, les types de données échangés, et les protocoles de communication utilisés.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions name="AktienKurs"
  targetNamespace="http://localhost:8080/AktienKurs"
  xmlns:xsd="http://schemas.xmlsoap.org/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl" >
  <service name="AktienKurs">
    <port name="AktienSoapPort" binding="AktienKursSoap" >
      <soap:address location="http://localhost:8080/AktienKurs/AktienSoapPort" />
    </port>
    <message name="Aktie.HoleWert">
      <part name="body" element="xsd:Tra" />
    </message>
    ...
  </service>
</definitions>
```

WSDL

Le XML (Extensible Markup Language)

XML
(*Extensible
Markup Language*)

XML (Extensible Markup Language) est un langage de balisage qui permet de décrire les données échangées entre le client et le serveur. XML est souvent utilisé pour encapsuler les données dans les messages échangés entre le client et le serveur.

```
<?xml version="1.0"?>
- <birds>
  - <owl id="1201">
    <species>Bubo bubo</species>
    <name>Eagle Owl</name>
    <region>Eurasia</region>
  </owl>
  - <owl id="1202">
    <species>Strix occidentalis</species>
    <name>Spotted Owl</name>
    <region>North America</region>
  </owl>
</birds>
```

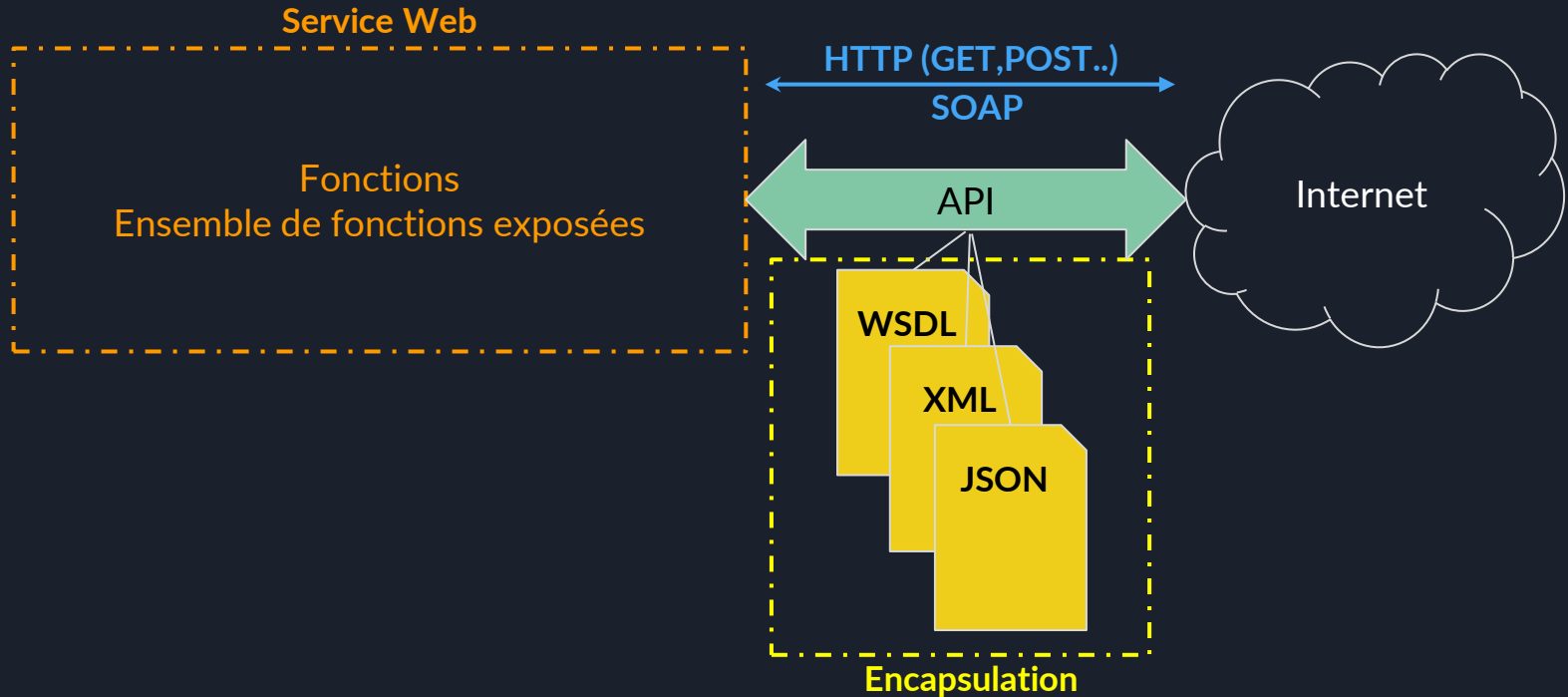
Le JSON (JavaScript Object Notation)

JSON
(JavaScript Object
Notation)

JSON (JavaScript Object Notation) est un format de données léger et facile à lire qui est souvent utilisé pour les services web RESTful. JSON est souvent utilisé pour encapsuler les données dans les messages échangés entre le client et le serveur.

```
{
  hey: "guy",
  anumber: 243,
  - anobject: {
    whoa: "nuts",
    - anarray: [
      1,
      2,
      "thr<h1>ee"
    ],
    more: "stuff"
  },
  awesome: true,
  bogus: false,
  meaning: null,
  japanese: "明日がある。",
  link: http://jsonview.com,
  notLink: "http://jsonview.com is great"
}
```


Notions de service, d'interface, de message et de transport





Différence entre REST et SOAP

REST

(REpresentational State Transfer)

Ensemble de procédés
architecturaux

IoT

SOAP

(Simple Object Access Protocol)

Protocole officiel géré par le W3C

Standards de conformité ACID

Atomicité
Cohérence
Isolement
Durabilité

Conformité & Sécurité



Modèles de développement de services web

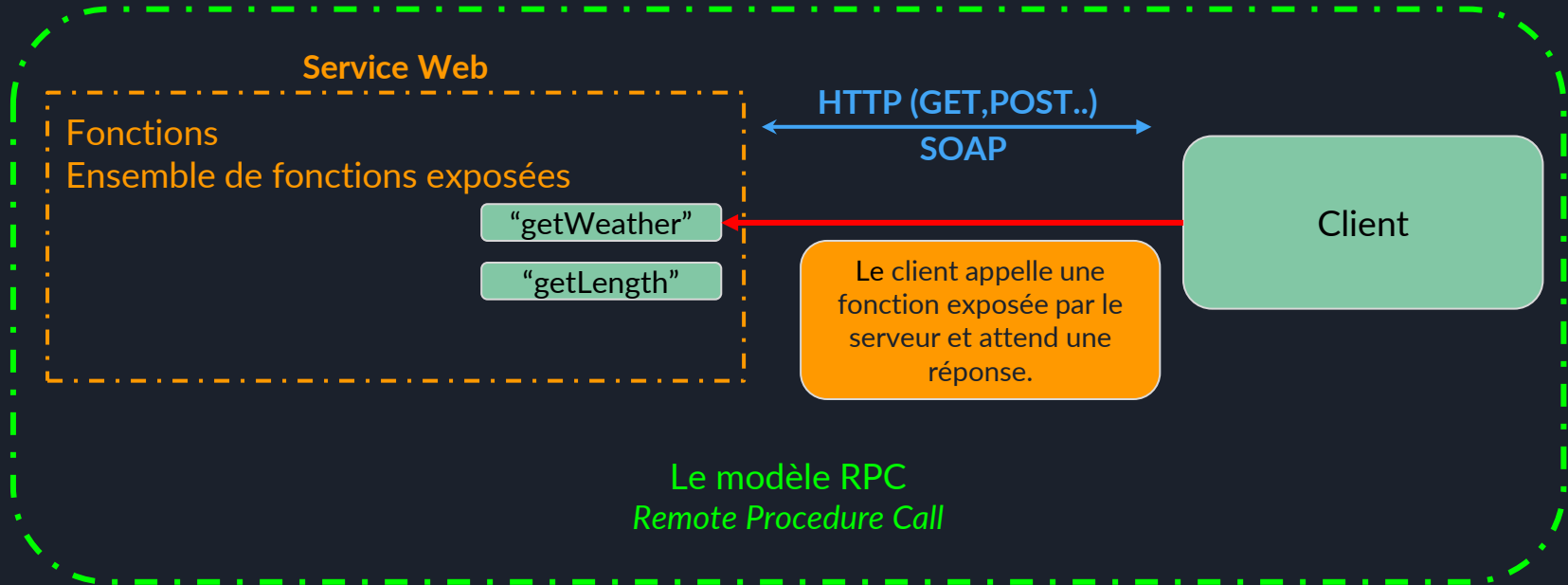
Il existe plusieurs modèles de développement de services web, chacun avec ses propres caractéristiques, avantages et inconvénients.

Le modèle RPC
Remote Procedure Call

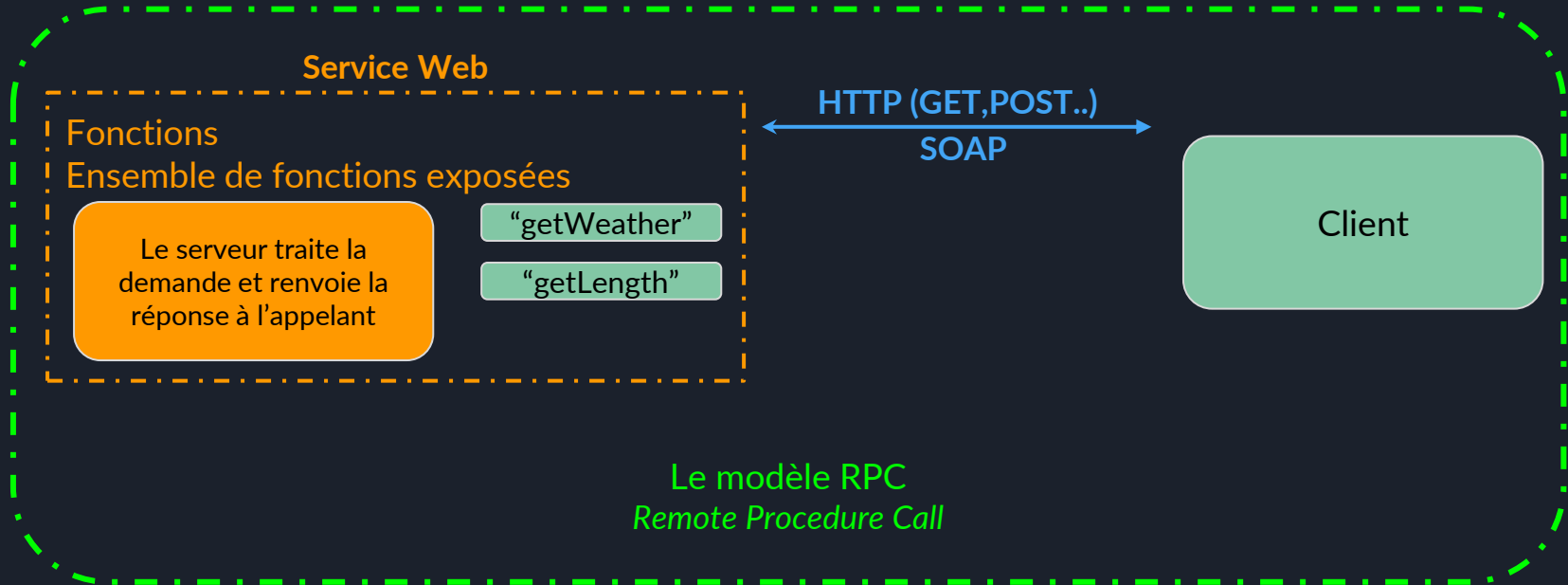
Le modèle
documentaire

Le modèle RESTful

Notions de service, d'interface, de message et de transport

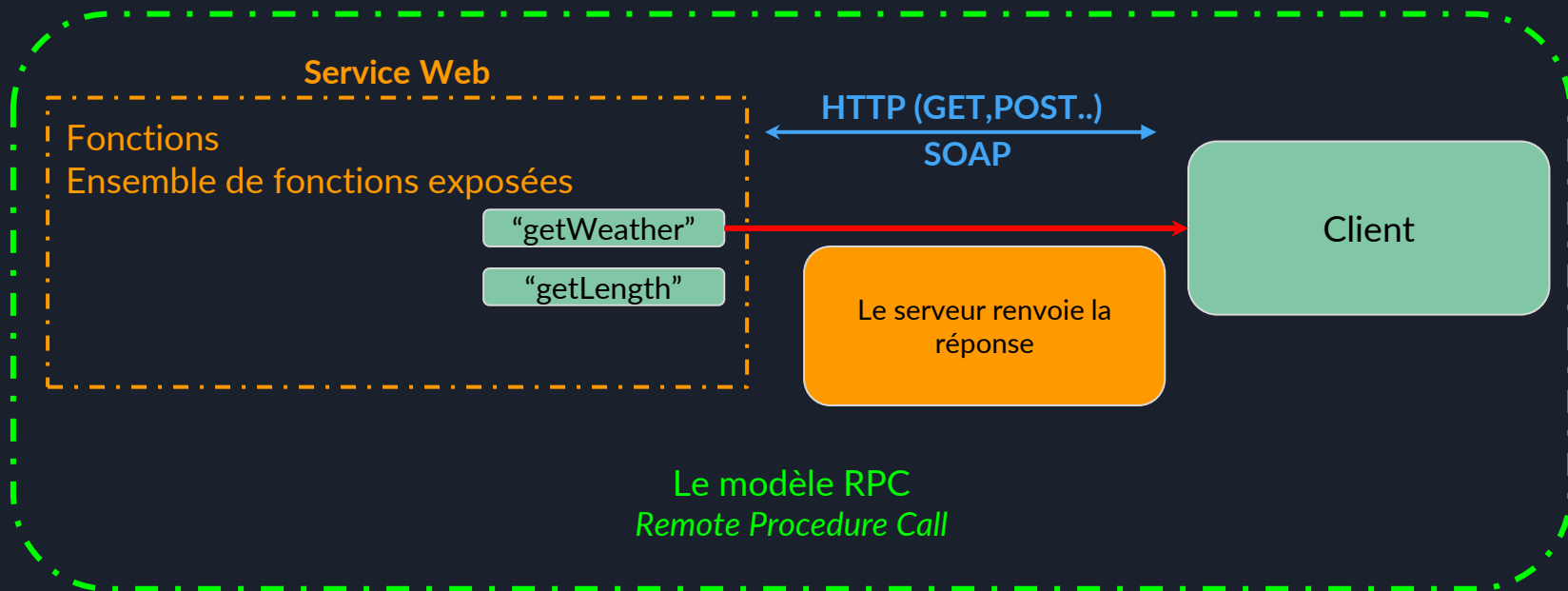


Notions de service, d'interface, de message et de transport

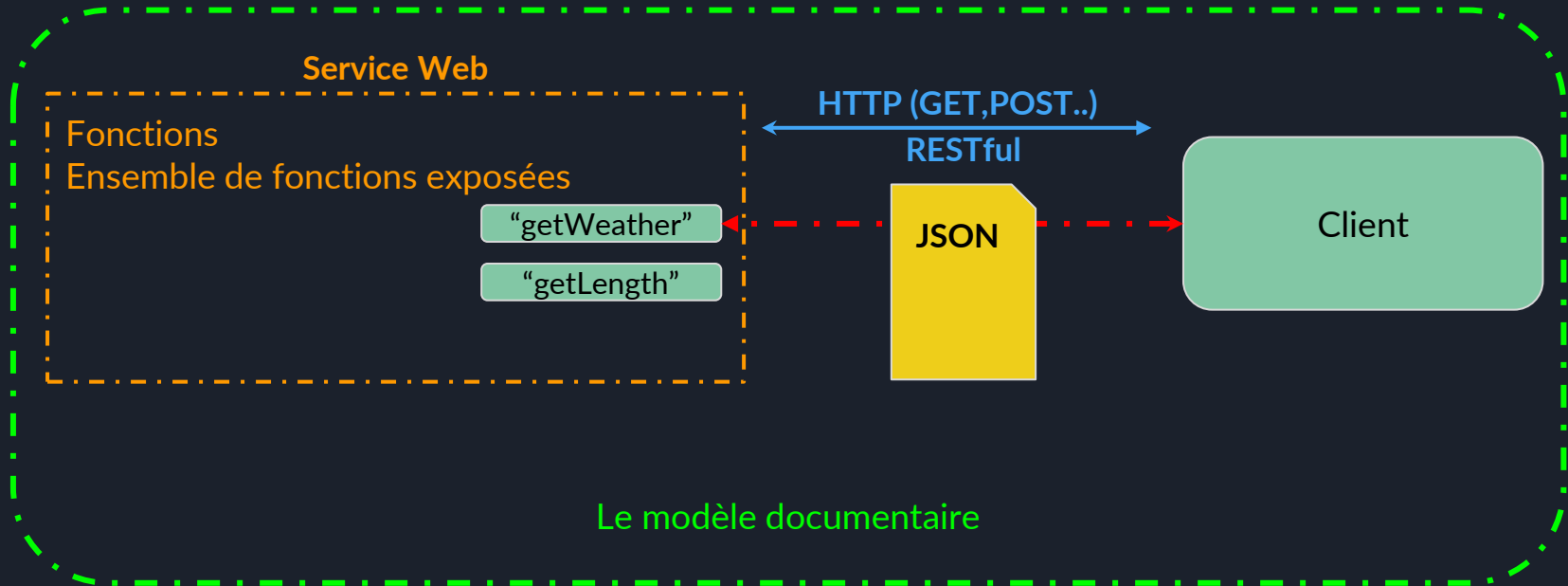


Notions de service, d'interface, de message et de transport

Ce modèle est similaire à l'appel de méthode locale, mais la fonction est exécutée sur le serveur distant. Le modèle RPC est souvent utilisé avec le protocole SOAP.



Notions de service, d'interface, de message et de transport





Le modèle RESTful

Ce modèle est basé sur les principes de l'architecture REST (Representational State Transfer). Le client envoie des requêtes HTTP au serveur pour effectuer des opérations sur des ressources (par exemple, un document XML ou une base de données). Le serveur renvoie une réponse HTTP, généralement au format JSON. Le modèle RESTful est de plus en plus populaire en raison de sa simplicité et de sa flexibilité.



Evolution des services web

Besoins d'évolution et migrations

Les besoins d'évolution des services web sont nombreux et peuvent inclure des changements de besoins des utilisateurs, des changements de la législation ou des normes, des changements de technologie, ou des corrections de bugs. Les besoins d'évolution peuvent être mineurs, comme la correction d'un bug mineur, ou majeurs, comme l'ajout de nouvelles fonctionnalités importantes.



Approches d'évolution

Il existe plusieurs approches pour l'évolution des services web, selon les besoins spécifiques.

L'évolution
compatible

L'évolution
incompatible

Le versioning



Approches d'évolution

L'évolution
compatible

Ajouter de nouvelles fonctionnalités sans affecter les anciennes

L'évolution compatible : Cette approche consiste à ajouter de nouvelles fonctionnalités sans affecter les anciennes. Par exemple, si un service web fournit une opération "getWeather", il peut être étendu pour fournir une nouvelle opération "getHistoricalWeather" sans affecter l'opération existante. Cette approche est souvent utilisée lorsque les clients existants ne doivent pas être modifiés.



Approches d'évolution

L'évolution
incompatible

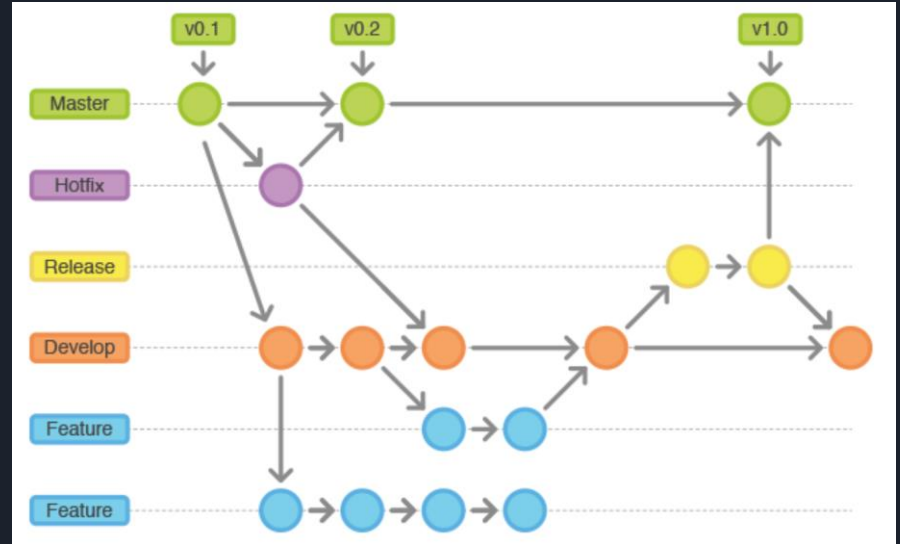
Modifier l'interface du service de manière à ce que les clients existants ne puissent plus l'utiliser sans modification.

Changements majeurs et mise à jour des clients existants

Approches d'évolution

Le Versioning

Il s'agit de donner des numéros de version à votre application ou à ses composants, de manière à pouvoir identifier facilement les différentes versions d'une application. Cela permet de gérer les différentes versions de l'application de manière efficace et de garantir la compatibilité entre les différentes versions. Il existe différents formats de versionnement tels que la norme semantic versioning (SemVer) qui est un format standard pour versioning les logiciels.





Stratégies de migration

La migration d'un service web d'une version à une autre peut être complexe et nécessite une planification soignée.

La migration en douceur

Mettre à jour les clients et le service web en même temps

Changements mineurs

Risques de perturbation faibles

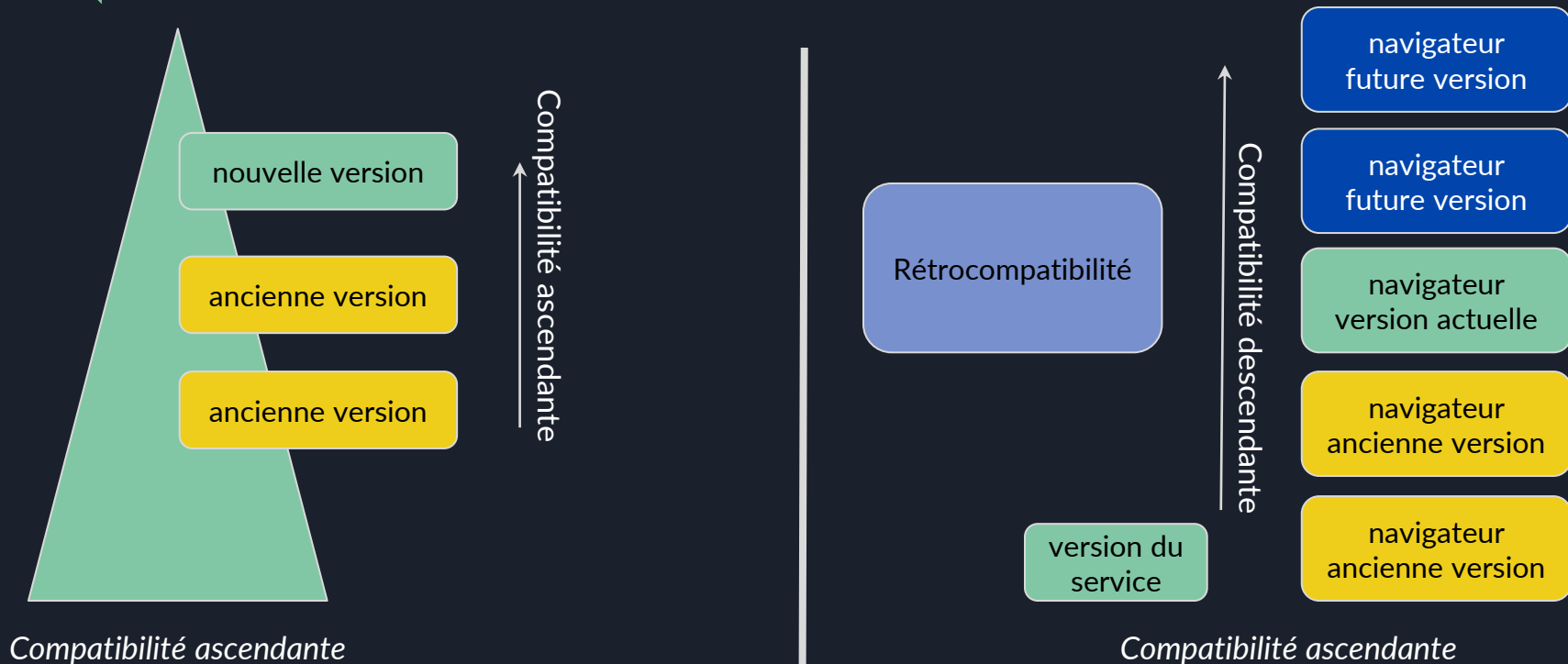
La migration par étapes

Mettre à jour le service web et les clients par étapes successives, en, s'assurant que chaque étape est testée et validée avant de passer à la suivante

Changements majeurs

Risques de perturbation élevés

Pratiques de gestion de la compatibilité ascendante et descendante





Les processus de maintien

Planification des mises à jour

MaJ en dehors des heures de pointe

Communication claire des modifications apportées

Formation pour les utilisateurs sur les nouvelles fonctionnalités

Gestion des dépendances

Compatibilité

Inventaire des dépendances

Vérification des MaJ

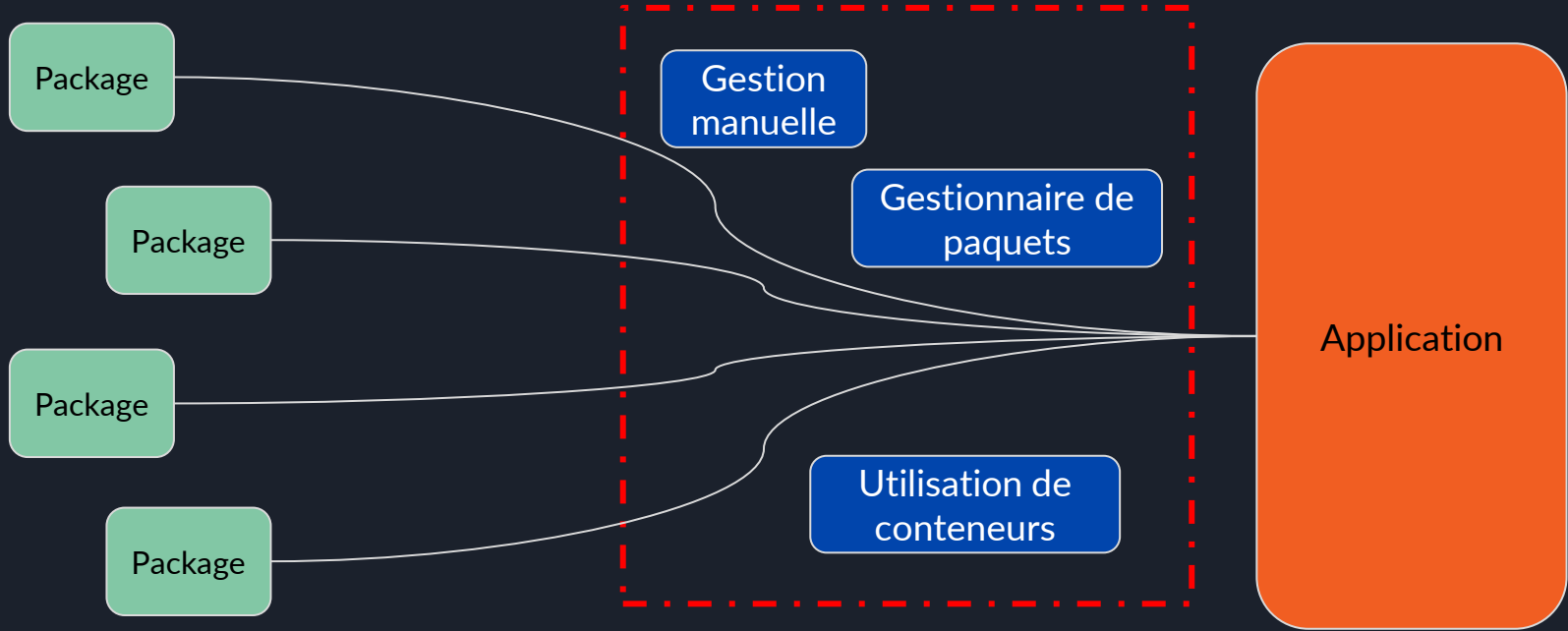
Documenter les modifications


Faire une doc claire et compréhensible

Équipe de développement et processus clés



La gestion des dépendances et des bibliothèques utilisées par les services web



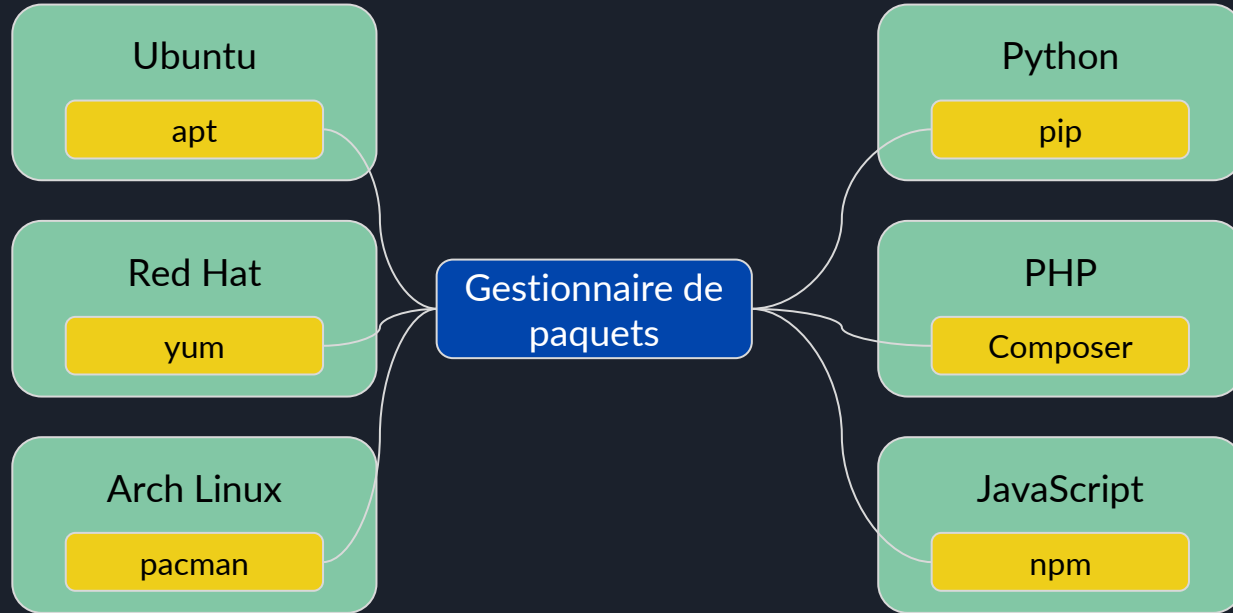


La gestion des dépendances et des bibliothèques utilisées par les services web

Cette approche consiste à gérer manuellement les dépendances de l'application en téléchargeant et en installant les bibliothèques nécessaires sur chaque serveur ou poste de développement. Cette approche peut être fastidieuse et sujette aux erreurs, surtout si l'application utilise de nombreuses dépendances.

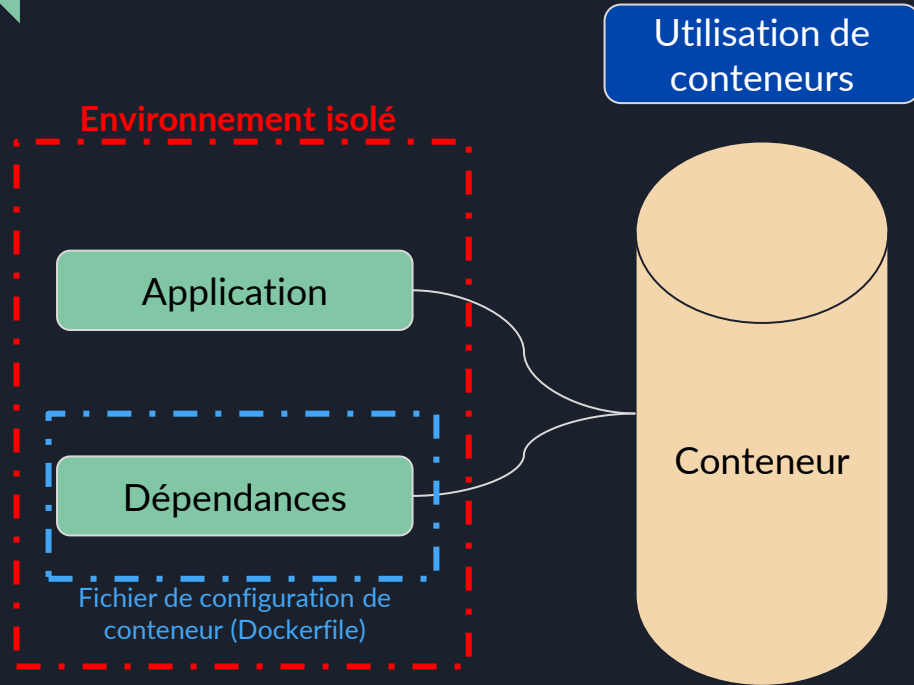
Gestion
manuelle

La gestion des dépendances et des bibliothèques utilisées par les services web



Cette approche est plus efficace que la gestion manuelle, mais il est important de s'assurer que les dépendances sont à jour et sécurisées.


La gestion des dépendances et des bibliothèques utilisées par les services web



La gestion des dépendances et des bibliothèques utilisées par les services web

Il est important de noter que la gestion des dépendances inclut également la mise à jour régulière des dépendances pour garantir la sécurité et la fiabilité de l'application. Il est également important de tenir un inventaire des dépendances utilisées dans l'application pour s'assurer qu'elles sont régulièrement mises à jour. Il est également important de s'assurer que les dépendances utilisées sont licenciées de manière appropriée et ne posent pas de risques de sécurité.





La gestion des dépendances et des bibliothèques utilisées par les services web

Gestion du cycle de vie de dépendance



La gestion des dépendances et des bibliothèques utilisées par les services web

Gestion des vulnérabilités

Scanner les dépendances et détecter les vulnérabilités



Etude de cas

Salle 1



Salle 2



Salle 3



Salle 4



Salle 5

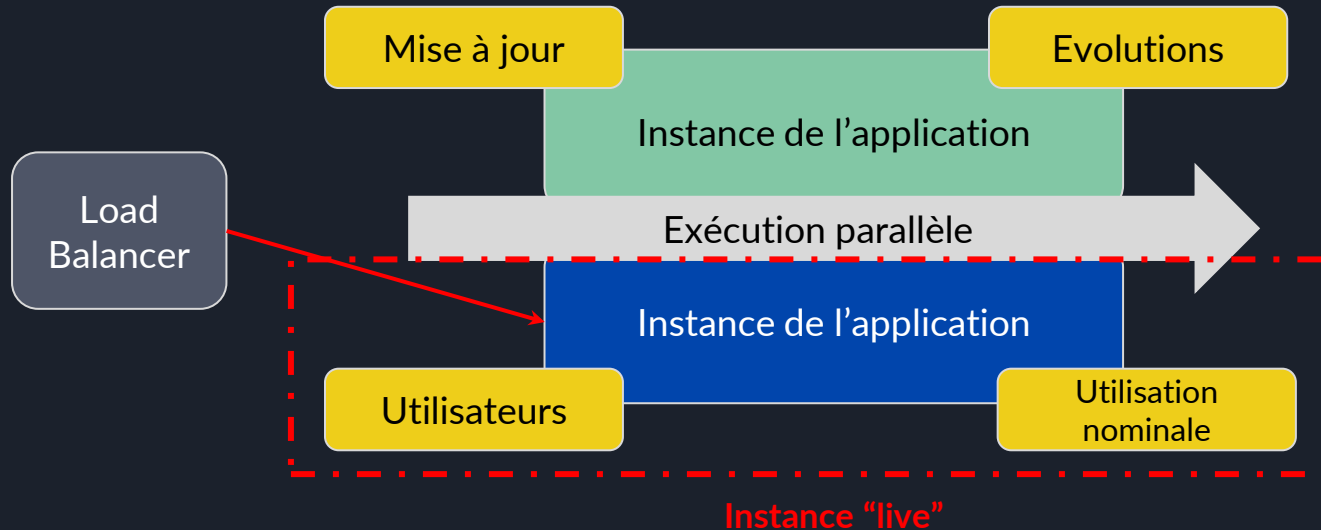


Salle 6



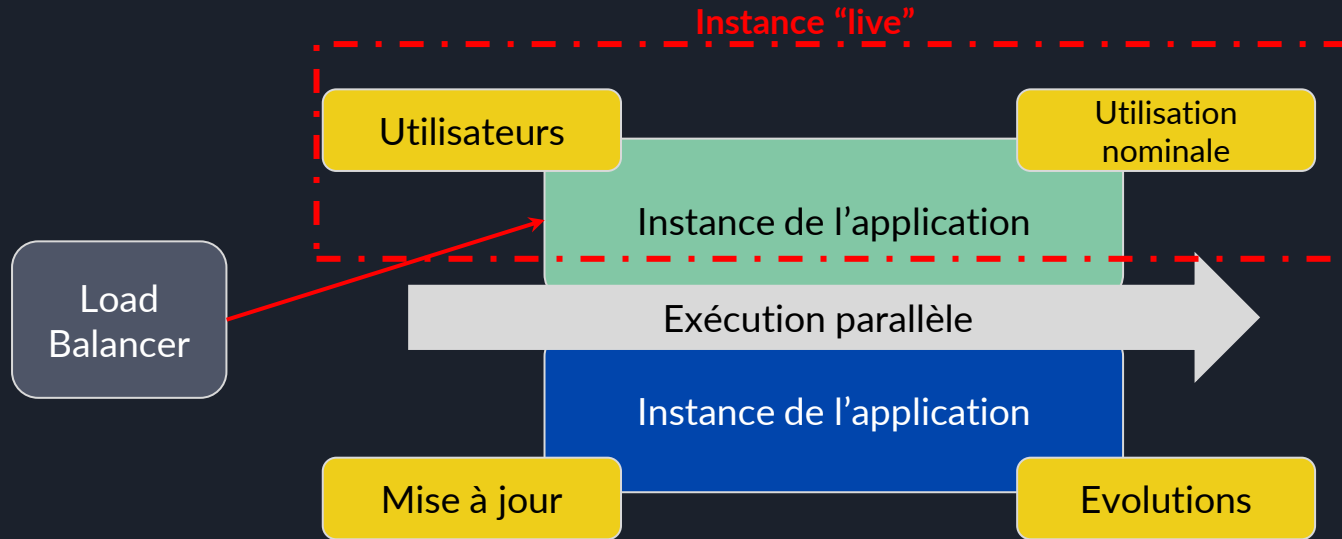
Les différentes techniques de déploiements

Déploiement blue-green



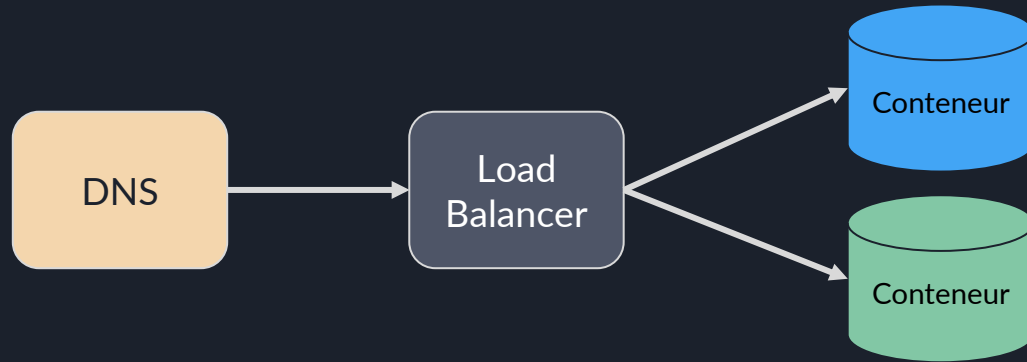
Les différentes techniques de déploiements

Déploiement blue-green



Les différentes techniques de déploiements

Le déploiement blue-green est souvent utilisé en conjonction avec des outils de gestion de conteneurs pour automatiser les tâches de déploiement et de gestion des instances. Il est également utilisé pour les déploiements sur des plateformes cloud, qui offrent des fonctionnalités pour automatiser les tâches de déploiement, de scalabilité et de gestion des instances.



Pratique de gestion de la qualité de service (QoS)

La qualité de service (QoS) est essentielle pour assurer le bon fonctionnement des services web.

La surveillance des performances

La gestion de la capacité

La gestion des incidents

Dashboard

Grafana

DataDog

SAP ME

QCM en ligne

