Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Centre Universitaire de Mila
Institut des Sciences et de la Technologie

Développement Web Avancé

Chapitre 3 : Mise en place d'un site web

Département MI

s.meghzili@centre-univ-mila.dz

Plan

- **Chapitre 3 : Mise en place d'un site web**
  - **Conteneurisation des applications**
  - **Vulnérabilité des applications Web et contre-mesures**
  - **L'hébergement de sites Web et Le référencement Internet**

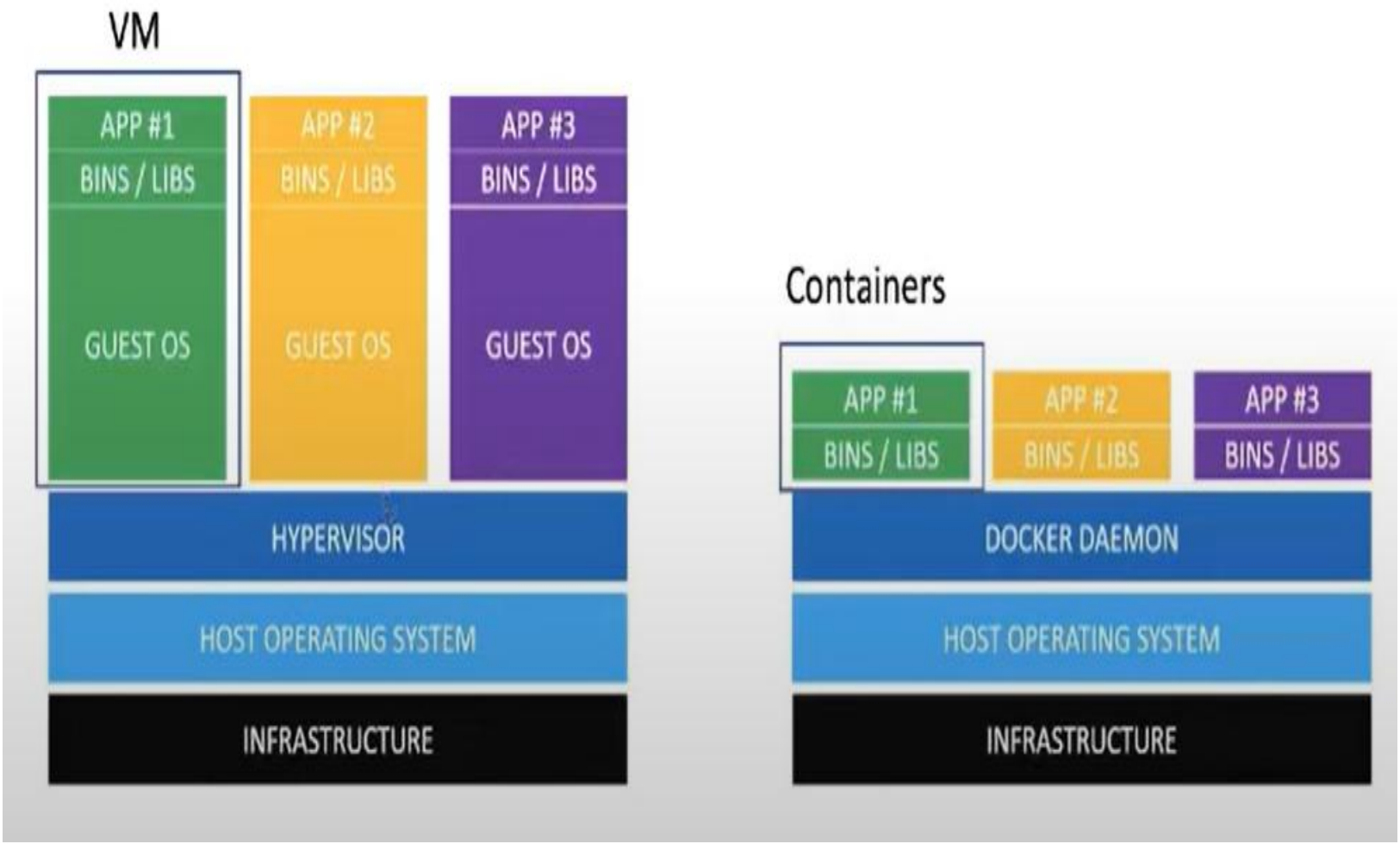
Conteneurisation des applications

- **Notion de Centenaire**
- **Notion d'image**
- **Docker File**
- **Docker Hub**
- **Docker compose**
- **Réseaux Docker**
- **Kubernetes**

Que-ce-qu'un Conteneur ?

- Un conteneur est une enveloppe virtuelle qui permet de distribuer une application avec tous les éléments dont elle a besoin pour fonctionner : fichiers source, environnement d'exécution, librairies, outils et fichiers.
- Ils sont assemblés en un ensemble cohérent et prêt à être déployé sur un serveur et son système d'exploitation (OS).
- Contrairement à la virtualisation de serveurs et à une machine virtuelle, le conteneur n'intègre pas de noyau, il s'appuie directement sur le noyau de l'ordinateur sur lequel il est déployé.
- Le conteneur est plus léger qu'une VM

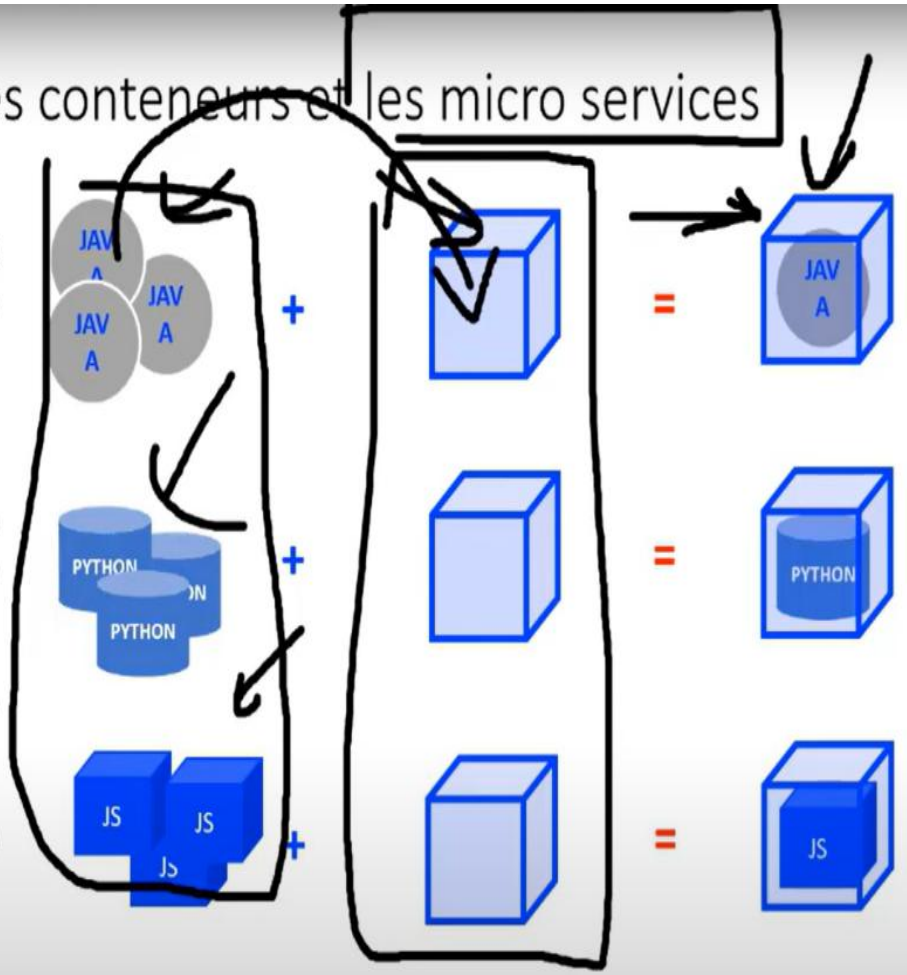
Différence entre Conteneur et VM



Opération de Conteneurisation

La conteneurisation (2/8) : les conteneurs et les micro services

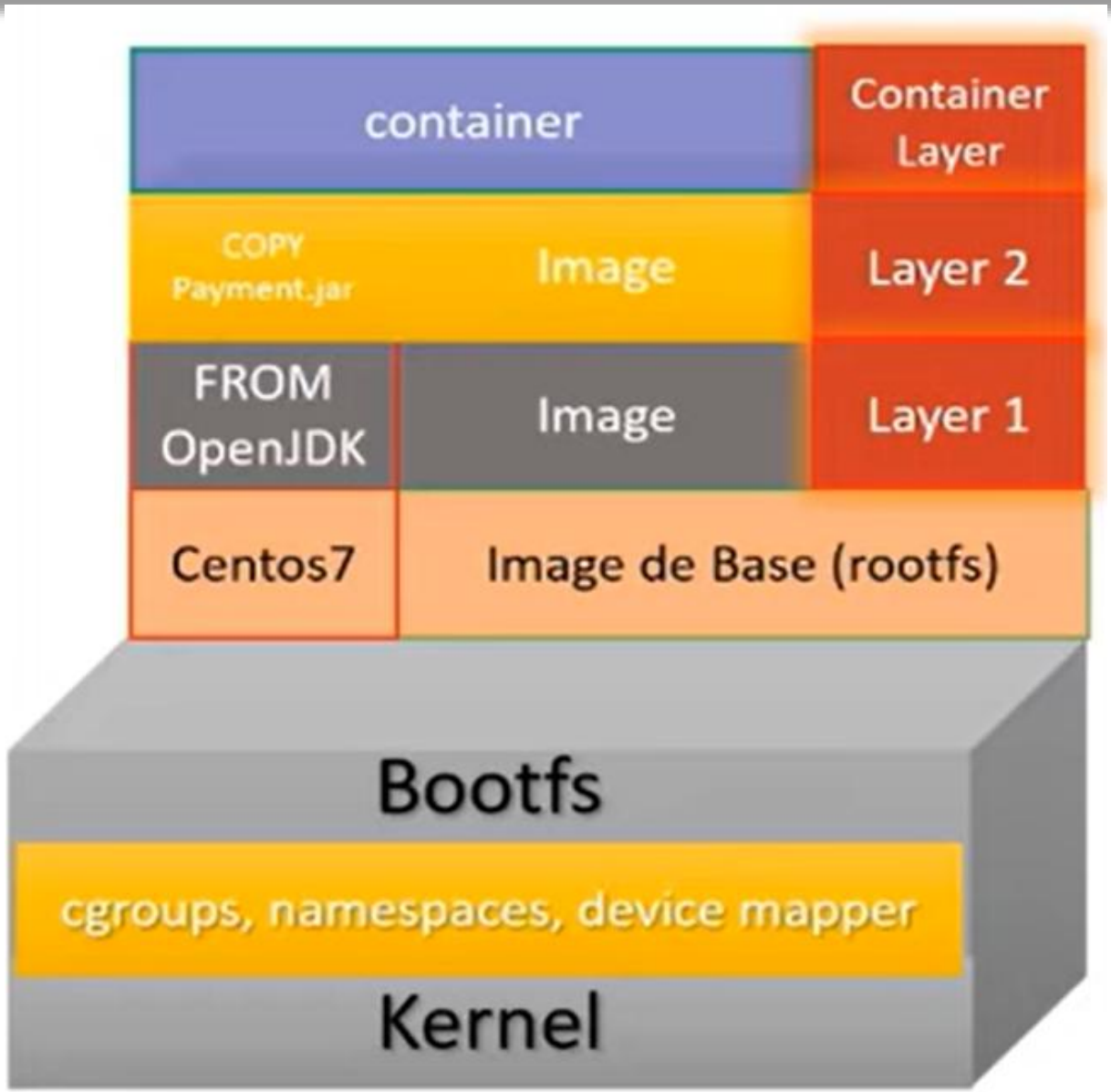
- Un conteneur n'est pas modifiable, c'est un ensemble de processus système cloisonnées
- Un conteneur ne reboot pas
- Un conteneur aura toujours le même comportement quel que soit l'hôte d'exécution.
- Il est portable et agnostique à l'hôte d'exécution
- Un conteneur s'instancie à partir d'une image



Que ce qu'une Image ?

- Une image est un fichier à partir duquel on crée des conteneurs.
- À l'origine de toute image il y a le boot filesystem (bootfs).
- L'image de base (rootfs) est la première couche d'une image.
- Une couche est le résultat d'une commande dans le Dockerfile (fichier docker).
- Les couches de l'image sont immuables (read only)
- Un Layer ne contient que le delta des changements des Layers précédentes (plus bas dans le schéma)
- Le contenu de chaque layer correspond à un répertoire du système hôte (habituellement sous /var/lib/docker/)
- Lorsque docker crée un conteneur à partir de ces layers il ajoute une couche Read/Write au-dessus.

La Structure d'une Image



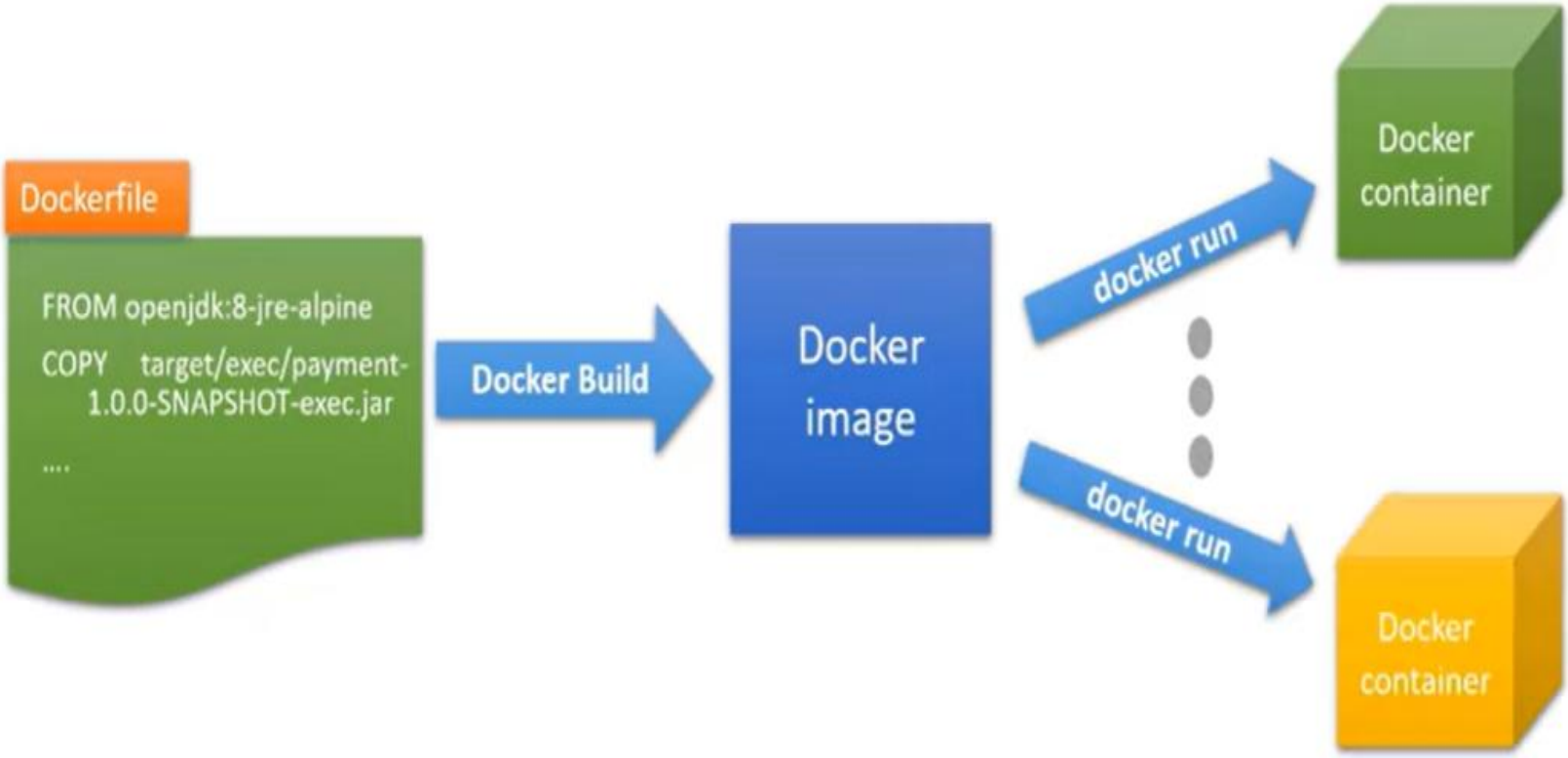
Exemple d'un DockerFile

```
1  FROM openjdk:8-jre-alpine
2  COPY target/exec/payment-1.0.0-SNAPSHOT-exec.jar /usr/local/lib/payment.jar
3  EXPOSE 8080
4  ENTRYPOINT ["java","-jar","/usr/local/lib/payment.jar"]
```

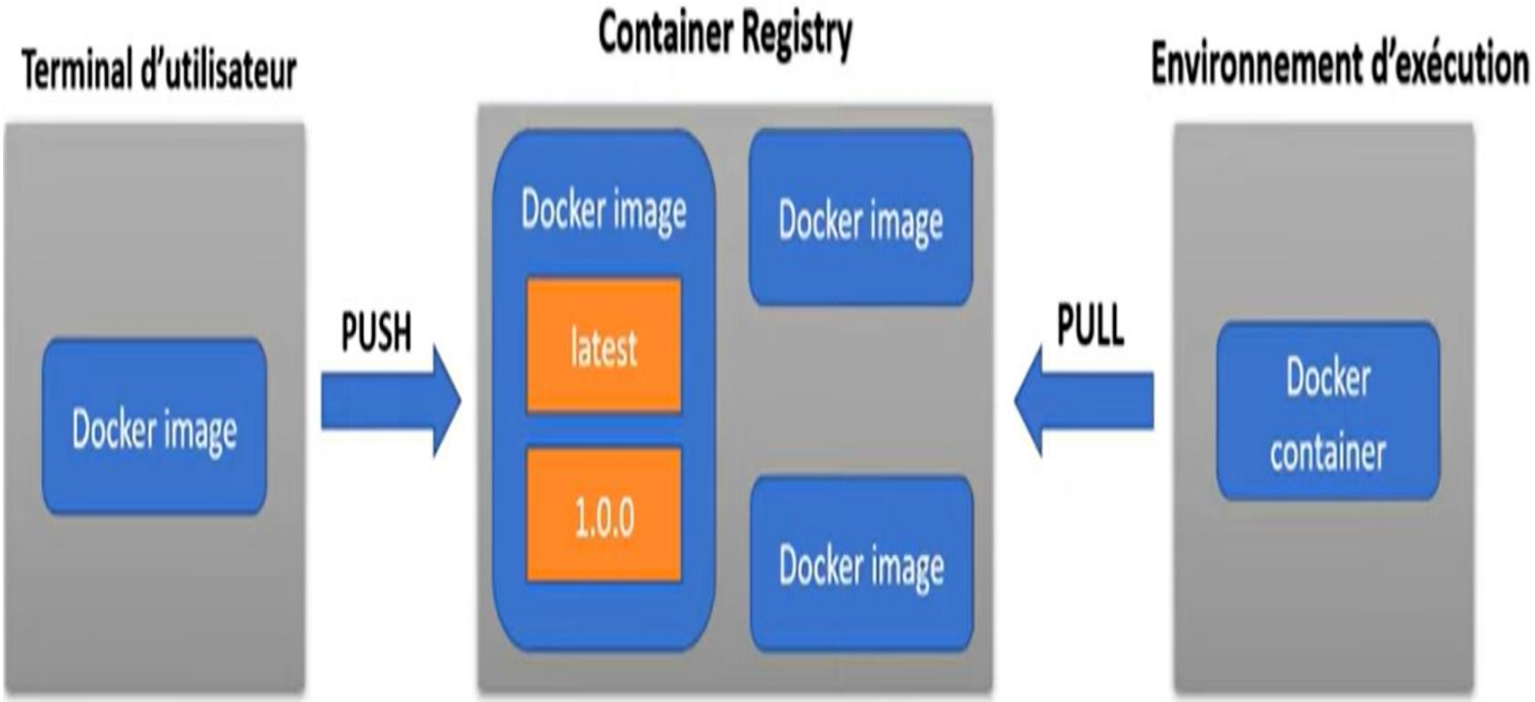
- FROM openjdk:8-jre-alpine
- COPY target/exec/payment-1.0.0-SNAPSHOT-exec.jar /usr/local/lib/payment.jar
- EXPOSE 8080
- ENTRYPOINT ["java","-jar","/usr/local/lib/payment.jar"]

Exemple d'un fichier Docker

La conteneurisation (5/8) : le processus de création d'un conteneur



Docker Hub : le registre d'image

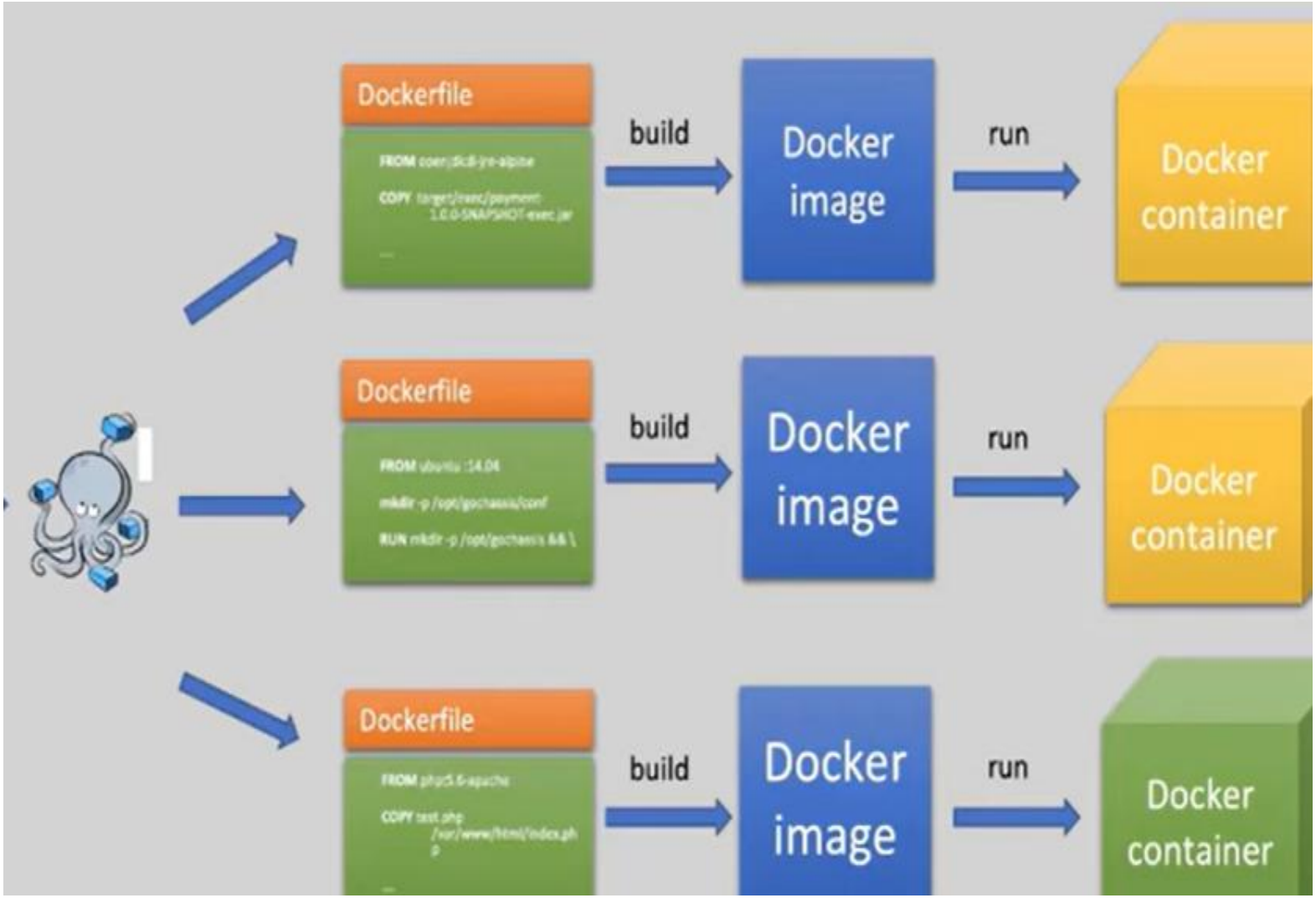


Docker Compose (1)

Docker-compose.yaml

```yaml
version: '3'
services:
  servicecenter:
    image: servicecomb/service-center:1.0.0
  catalog:
    image: catalog:1.0.0
    environment:
      CSE_REGISTRY_ADDR: http://servicecenter:30100
    links:
      - servicecenter
  payment:
    image: payment:1.0.0
    environment:
      SC_HOST: servicecenter
    links:
      - servicecenter
  mesherfrontend:
    image: thanda/mesher:osi
    environment:
      - CSE_REGISTRY_ADDR=http://servicecenter:30100
      - SERVICE_NAME=frontend
      - VERSION=0.0.1
      - APP_ID=OSIConference
    ports:
      - 80:80
    links:
      - servicecenter
  frontend:
    image: frontend:1.0.0
    environment:
      - http_proxy=127.0.0.1:30101
    network_mode: "service:mesherfrontend"
```

Docker Compose (2)



Docker Compose (3)

- **docker-compose up** démarre les services décrits dans mon docker-compose.yml et ne me rend pas la main.
- **docker-compose up -d** fait la même chose mais me rend la main une fois que les services sont démarrés.
- **docker-compose up –build** reconstruit les services avant de les lancer.
- **docker-compose down** stoppe les services.

Réseaux docker (1)

Les **réseaux** Docker fournissent un moyen pour les **conteneurs** de communiquer entre eux, que ce soit sur une même machine **hôte** ou sur des machines **hôtes** différentes dans un environnement **distribué**.

Docker propose plusieurs types de réseaux : **bridge**, **host** et **overlay**.

- Le réseau **bridge** est le réseau par défaut. Il isole chaque container dans son propre espace réseau, mais permet la communication avec l'hôte.
- Le réseau **host** permet au container d'utiliser directement l'interface réseau de l'**hôte**, sans isolation.
- Le réseau **overlay** est utilisé dans des environnements d'**orchestration** comme **Docker Swarm** ou **Kubernetes** et permet de connecter des containers sur différents hôtes.

Réseaux docker (2)

Caractéristique des réseaux Docker :

- **Isolation** : Les conteneurs sont isolés par défaut, ce qui signifie qu'ils ne peuvent pas communiquer entre eux sans configuration supplémentaire.
- **Connectivité** :Ils permettent aux conteneurs de communiquer de manière transparente, comme s'ils étaient sur le même réseau physique.
- **Sécurité** : Ils peuvent être segmentés pour limiter l'accès aux ressources et protéger vos applications contre les attaques.
- **Simplicité** : Ils sont faciles à configurer et à gérer, ce qui vous permet de gagner du temps et de vous concentrer sur vos applications

Réseaux docker (3)

Exemple: Création et gestion des réseaux Docker:

Pour créer un réseau Docker, vous pouvez utiliser la commande:

$ docker network create *reseau1*

```
alphorm@alphorm-as:~$ docker network create reseau1
8e964c3977bf0ea956841ac4152c7db24fcd8d5e7d825c152aa7421f1307637b
alphorm@alphorm-as:~$
```

Pour attacher un réseau à un conteneur, il suffit d'utiliser l'argument `--network` avec **`docker run.`** Exemple:

$ docker run --name alphorm --network reseau1

Vous pouvez utiliser la commande **docker network connect** pour connecter un conteneur à un réseau existant. Par exemple :

$ **docker network connect** *nom_du_réseau nom_id_container*

Kubernetes

Kubernetes est devenu un standard incontournable pour **déployer et gérer des applications conteneurisées**.

Mais sa maîtrise ne concerne pas qu'un seul métier :

administrateurs système et développeurs doivent travailler ensemble pour exploiter pleinement ses capacités.

Cela devrait vous rappeler la culture **DevOps** : la collaboration entre équipes pour améliorer la qualité et l'efficacité des déploiements.

- **Administrateurs système** : Ils assurent la **gestion des clusters**, la **sécurisation des infrastructures** et l'**optimisation des ressources**.
- **Développeurs** : Ils utilisent **Kubernetes** pour **déployer, mettre à jour et scaler** leurs applications sans dépendre d'une infrastructure spécifique.

kubernetes

Kubernetes à quoi sert-il ? (1)

1. Centraliser la gestion des conteneurs

L'objectif principal de **Kubernetes** est de fournir une **plateforme unifiée** pour orchestrer vos applications conteneurisées. Que vous ayez un cluster de trois machines ou une infrastructure mondiale, **Kubernetes** agit comme une couche d'abstraction qui vous permet de :

- **Déployer** vos conteneurs de manière uniforme.
- **Surveiller** leur état en continu.
- **Adapter automatiquement** les ressources en fonction des besoins.

2. Garantir la disponibilité des applications

L'une des promesses fondamentales de **Kubernetes** est de garantir que vos applications restent disponibles, quoi qu'il arrive. Il assure cette **haute disponibilité** grâce à des mécanismes comme :

- **Redondance** : En déployant des répliques de vos applications sur plusieurs nœuds.
- **Self-healing** : En redémarrant automatiquement les conteneurs défaillants.
- **Load balancing** : En répartissant intelligemment le trafic entre vos **pods**.

Kubernetes à quoi sert-il ? (2)

3. Simplifier la scalabilité

Kubernetes est un allié de choix pour faire face aux variations de charge. Son objectif est de permettre une **scalabilité simple et rapide**, que ce soit :

Verticale : En augmentant les ressources (**CPU**, **RAM**) allouées aux conteneurs.

Horizontale : En ajoutant ou supprimant des répliques de vos applications en fonction des besoins.

Par exemple, lors d'un **pic** de trafic, **Kubernetes** peut automatiquement ajouter des **pods** pour absorber la charge, puis les retirer une fois le calme revenu.

4. Automatiser les déploiements et les mises à jour

Avec **Kubernetes**, fini les déploiements manuels laborieux ! Son objectif est de **simplifier et automatiser** les cycles de vie de vos applications grâce à des fonctionnalités comme :

Rolling Updates : Déployer progressivement une nouvelle version sans interruption.
Rollbacks : Revenir rapidement à une version **stable** en cas de problème.
Canary Releases : Tester une nouvelle version sur une fraction des utilisateurs avant un déploiement complet.

Kubernetes à quoi sert-il ? (3)

5. Optimiser les ressources

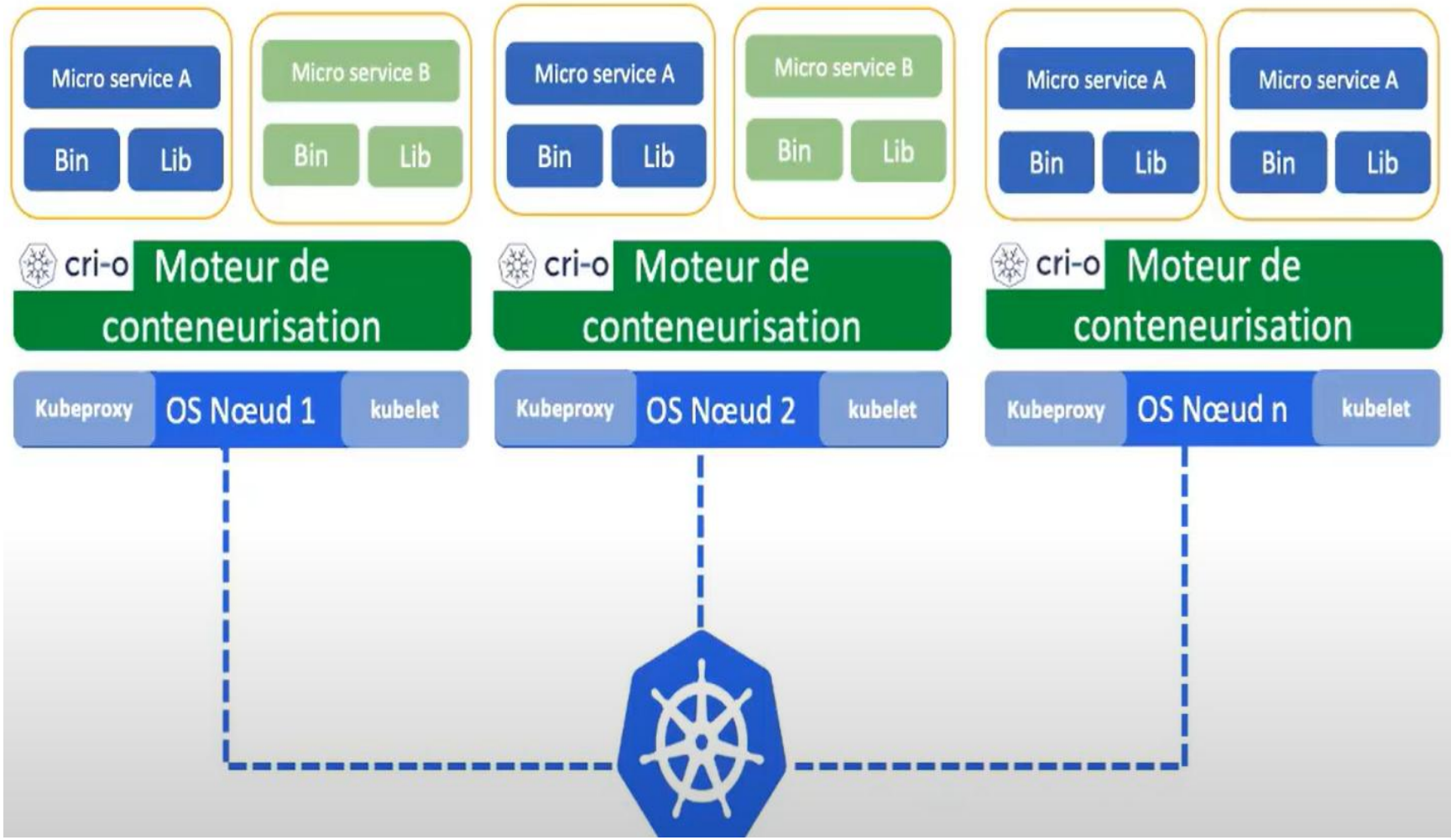
Dans un monde où les infrastructures **cloud** coûtent cher, **Kubernetes** aide à **optimiser l'utilisation des ressources**. Il attribue automatiquement les **conteneurs** aux nœuds disponibles en tenant compte de leurs ressources (**CPU**, **mémoire**) pour éviter les gaspillages.

6. Intégrer la sécurité dans les processus (DevSecOps)

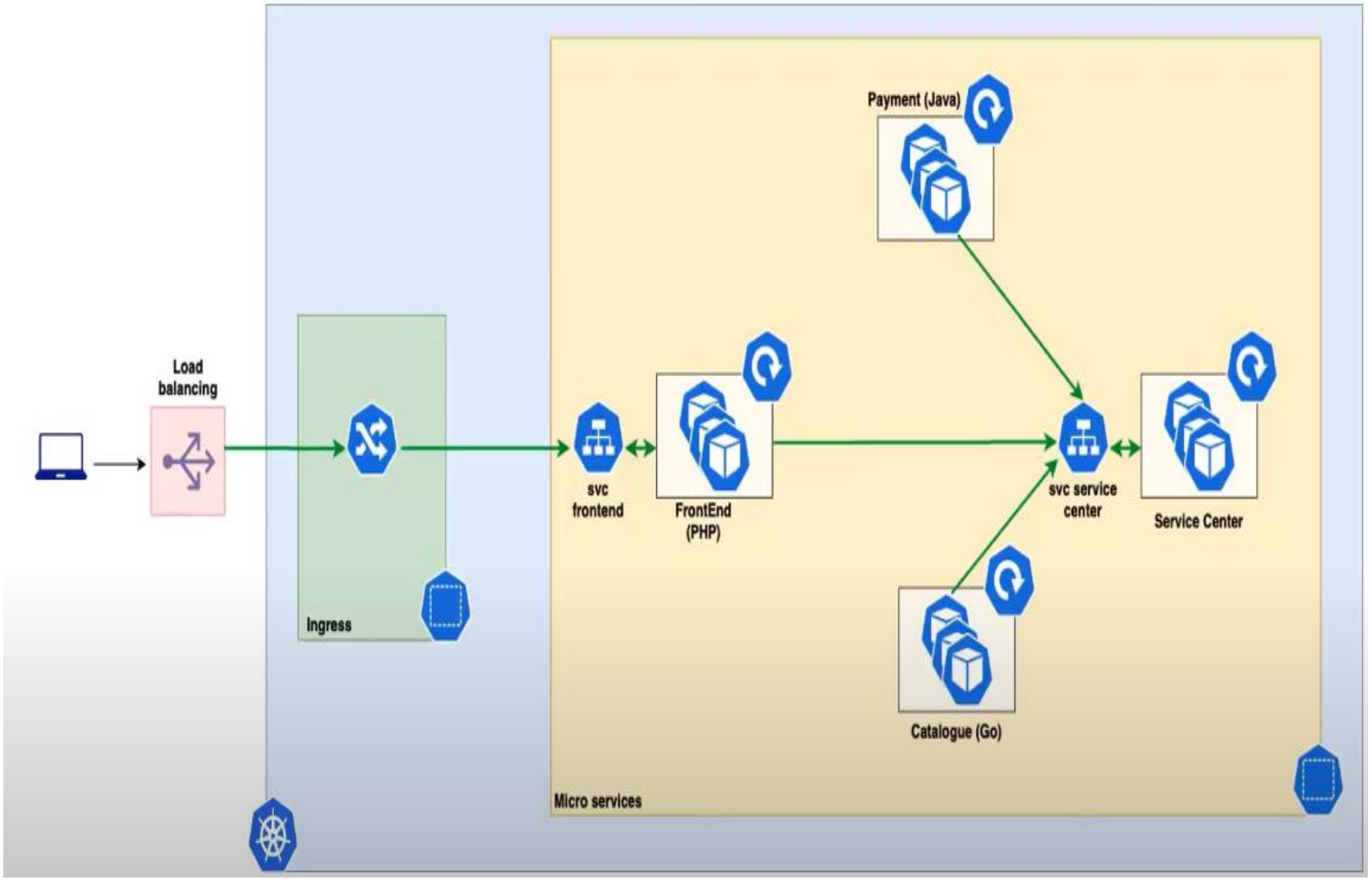
Enfin, **Kubernetes** vise à intégrer la **sécurité** dans vos processus. En tant que plateforme clé pour le **DevSecOps**, il offre :

Isolation des workloads grâce aux **namespaces**.
Gestion fine des accès avec les **RBAC**. (**Le contrôle d'accès basé sur le rôle**)
Chiffrement des communications entre les composants du cluster.

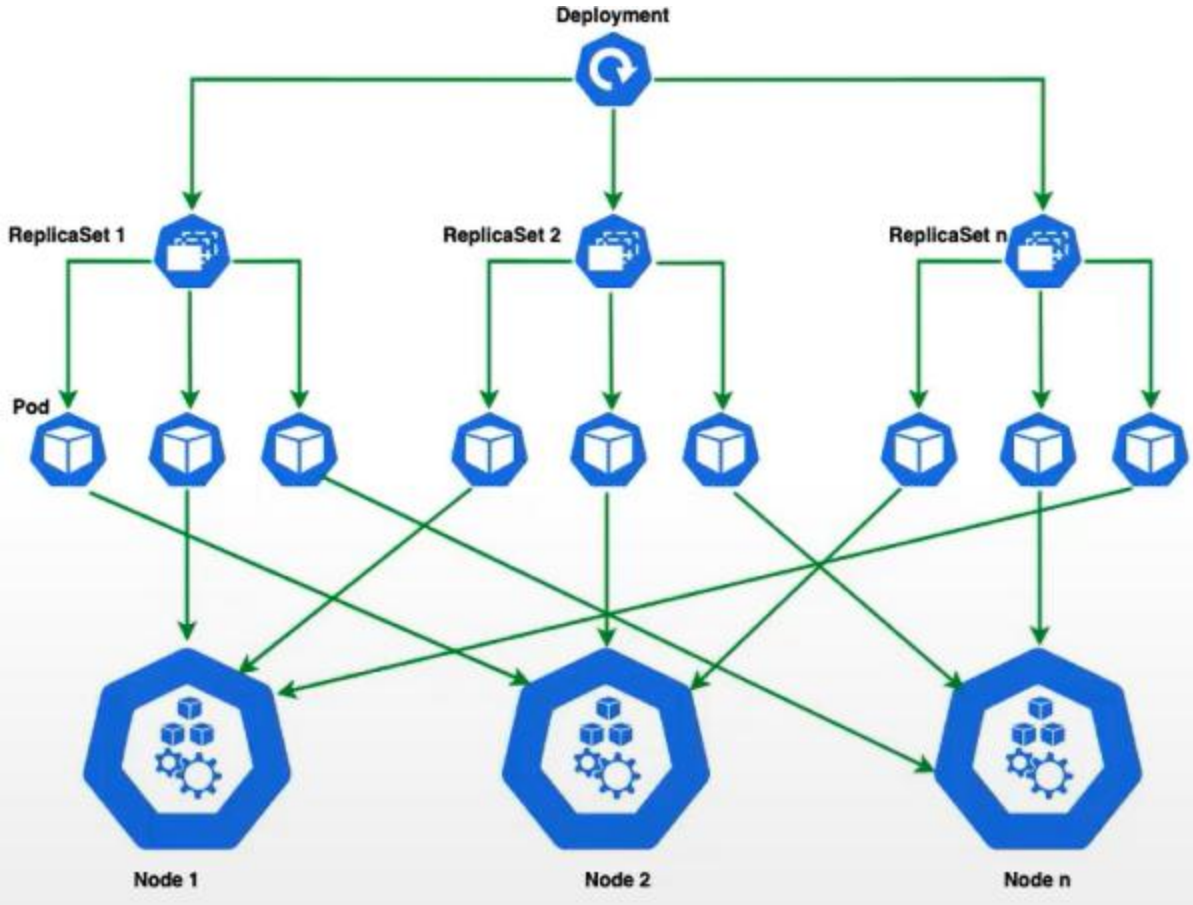
Exemple Kubernetes



Exemple Kubernetes



Exemple Kubernetes



Principales Attaques:

- **SQL Injection**
  - La mauvaise vérification des **entrées** permet une **requête SQL** malveillante
  - **=>** Les défenses connues **résolvent** efficacement le problème
- **XSS (Cross-site Scripting):** script inter-sites
  - Le problème provient de l'**écho** d'une **entrée** non fiable
  - **=> Difficile à prévenir** : nécessite des soins, des tests, des outils, …
- **CSRF (Cross-site Request Forgery):** falsification de requête intersite
  - Demande **falsifiée** tirant parti de **la session en cours**
  - **=>** Peut être évité (si les problèmes **XSS** sont résolus)
- **XEE (XML External Entity)**
- **Denial of Service (Dos)**

SQL Injection

- La faille **SQLi**, abréviation de "**SQL Injection**", soit "injection SQL" en français, est un groupe de méthodes d'exploitation de faille de sécurité d'une **application** interagissant avec une **base de données**.

- Elle permet d'**injecter** dans la **requête SQL** en cours un **morceau de requête non prévu** par le système et pouvant en compromettre la sécurité.
sécurité.

- **Cause** :La mauvaise vérification des **entrées** permet une **requête SQL** malveillante

Exemple d'Injection SQL: Entré Malicieux

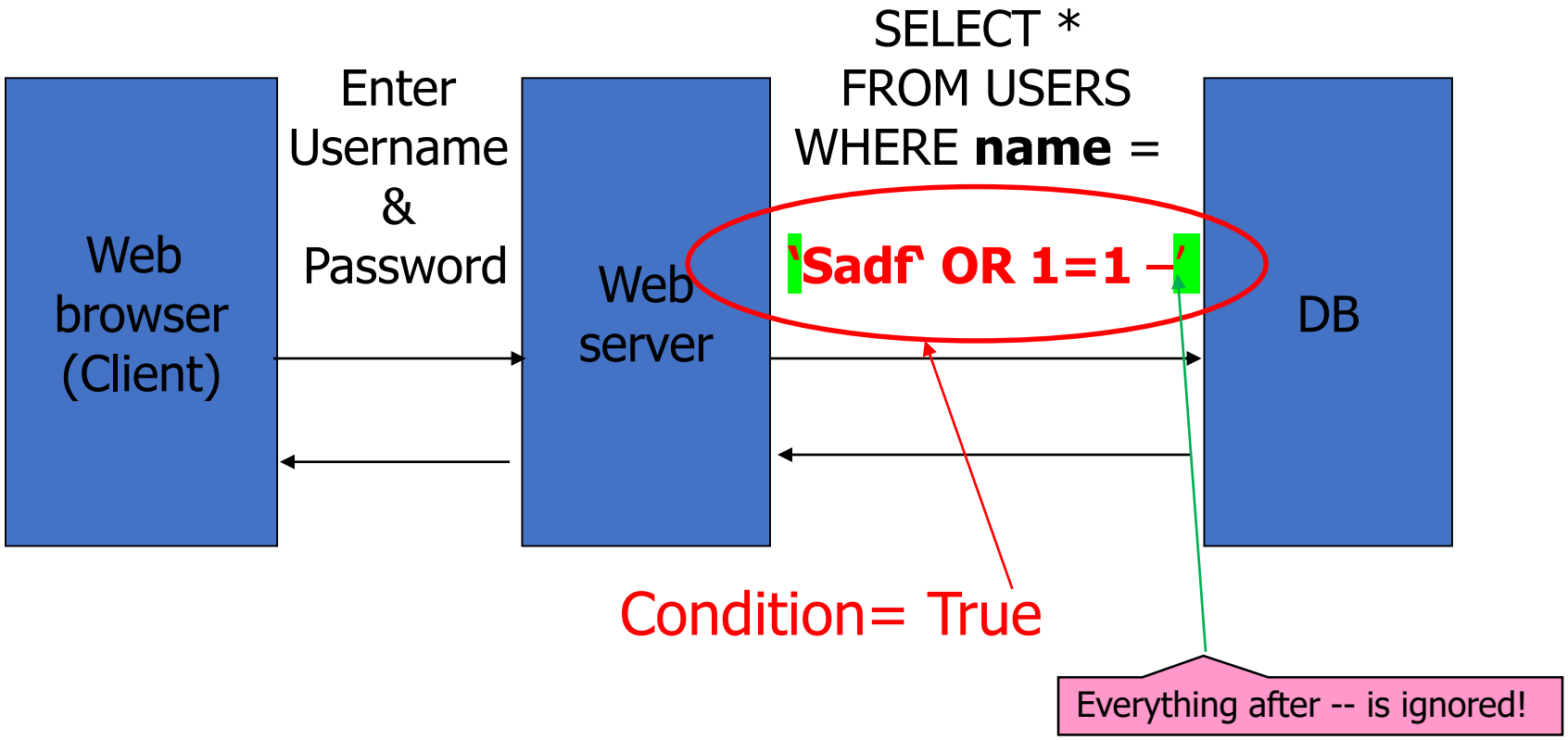
Table users

Id	name	pass	email
1	Sadf	****	s@univ.dz
2	Nassim	****	n@univ.dz

Logon page that shows an SQL injection string

- **Requête: SELECT * From USERS WHERE name=' ' and pass=' '**

Exemple d'Injection SQL: Entré Malicieux



Autres SQL Injection

- Créer un nouveau utilisateur:

  ’; INSERT INTO USERS (‘name’, ‘pass’)

  VALUES (‘hacker’,‘38a74f’);

- Pirater le password:

  ’; UPDATE USERS SET email=hcker@root.org WHERE email=victim@yahoo.com

- Eliminer tous les comptes utilisateurs :

’; **DROP TABLE USERS;** -- ’

Définition XSS(1)

- Le **C**ross **S**ite **S**cripting (**XSS**) est l'une des ***attaques*** au niveau des applications les plus courantes que les ***pirates*** utilisent pour ***se faufiler*** dans les applications Web aujourd'hui.

- **XSS** est une attaque contre la ***vie privée*** des clients d'un site Web particulier qui peut conduire à une ***violation*** totale de la sécurité lorsque les ***détails des clients*** sont ***volés ou manipulés***.

- Contrairement à la plupart des attaques, qui impliquent deux parties : l'attaquant et le ***site Web***, ou l'attaquant et le ***client victime***,

l'attaque ***XSS*** implique trois parties :

- l'***attaquant***,

- un ***client (victime)***

- et le ***site Web***.

Définition XSS(2)

- Le but de l'attaque CSS est de **voler les cookies** du client, ou toute autre information sensible, qui peuvent **identifier** le client avec le site Web.

- Avec le ***jeton*** de l'utilisateur légitime à portée de main, l'**attaquant** peut agir en tant qu'**utilisateur** dans son interaction avec le site - en particulier, **usurper** l'identité de l'utilisateur.

Ecouter les entrées d'un utilisateur

- **Erreur** classique dans le **Coté Serveur** de l' application:

http://naive.com/**search.php**?**term**="**Britney Spears**"

search.php répond avec :

<html> <title>**Search results**</title>

<body>**You have searched for**

<?php echo $_GET[term] ?>… </body>

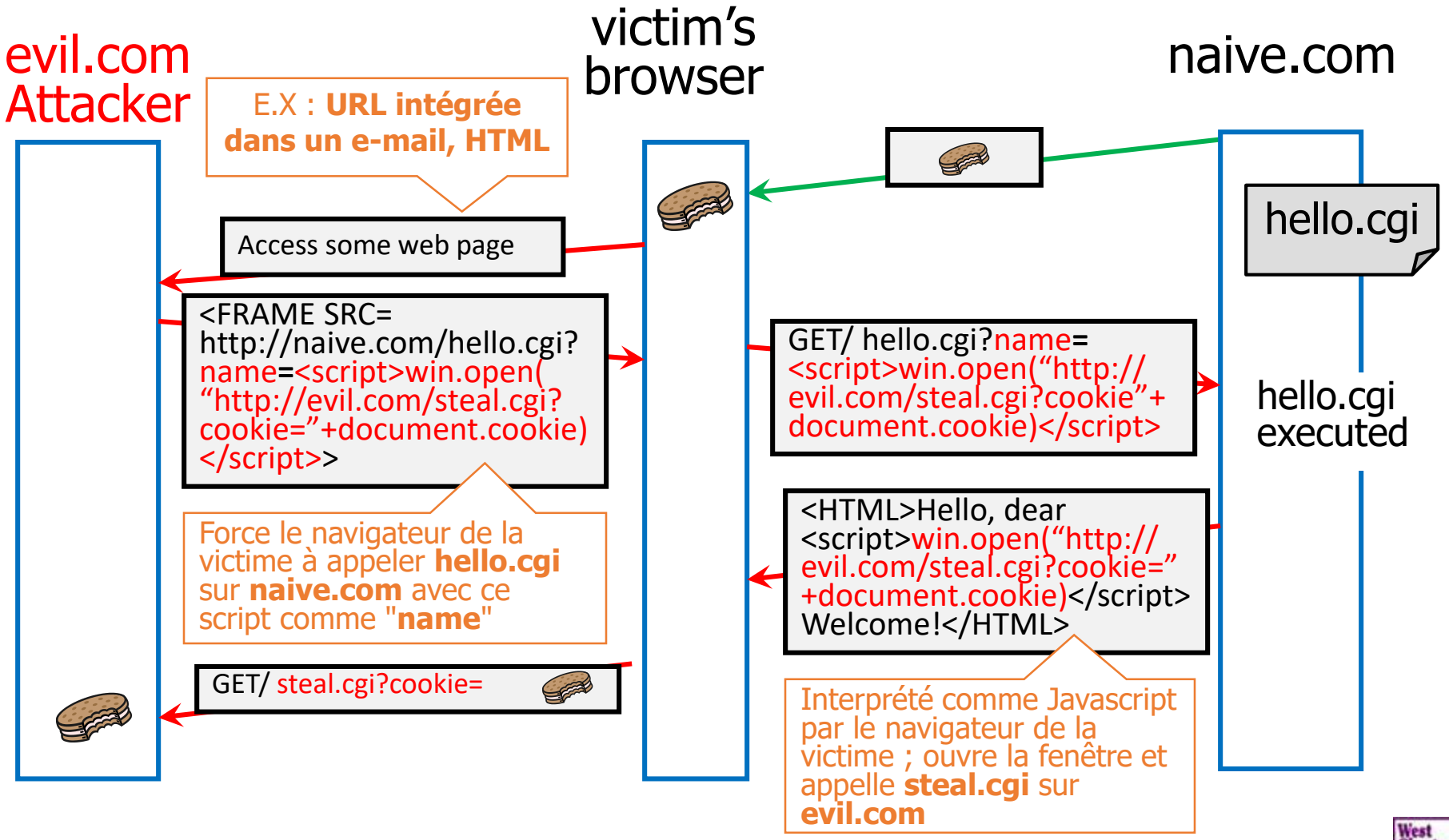
OU

GET/ hello.cgi?name=**Bob**

hello.cgi répond avec:

<html>Welcome, dear **Bob**</html>

XSS: exemple 1



CSRF: Cross-Site Request Forgery

- CSRF : C'est une attaque ou le navigateur de la **victime** génère une **requête** vers une application Web **vulnérable**
- Cette vulnérabilité est causée par la **capacité** que les navigateurs ont d' envoyer automatiquement des données d'authentification (***session ID***, ***IP adresse***, ..) dans **chaque requête**.
- **Imaginez** :
  - Que se passerait-il si un **attaquant** pouvait utiliser votre **souris** pour effectuer des clicks sur votre ***site de banque*** en ligne a votre place.
  - Que pourrait-il **faire** ?
- **Impact :**
  - Initiation de transactions (**transfert de fonds**, logoff, modification de données, …)
  - Accès à des **données sensibles**
  - Changement des **mots de passes/identifiants**

CSRF: exemple 1 – Transfer d'argent (1)

- Les utilisateurs se connectent à **bank.com**, oublient de se **déconnecter !**
  - Le **cookie** de session reste dans l'état du **navigateur**

- L'utilisateur visite alors un site Web **malveillant** contenant :

  <form name=BillPayForm action=http://bank.com/transfer.php>

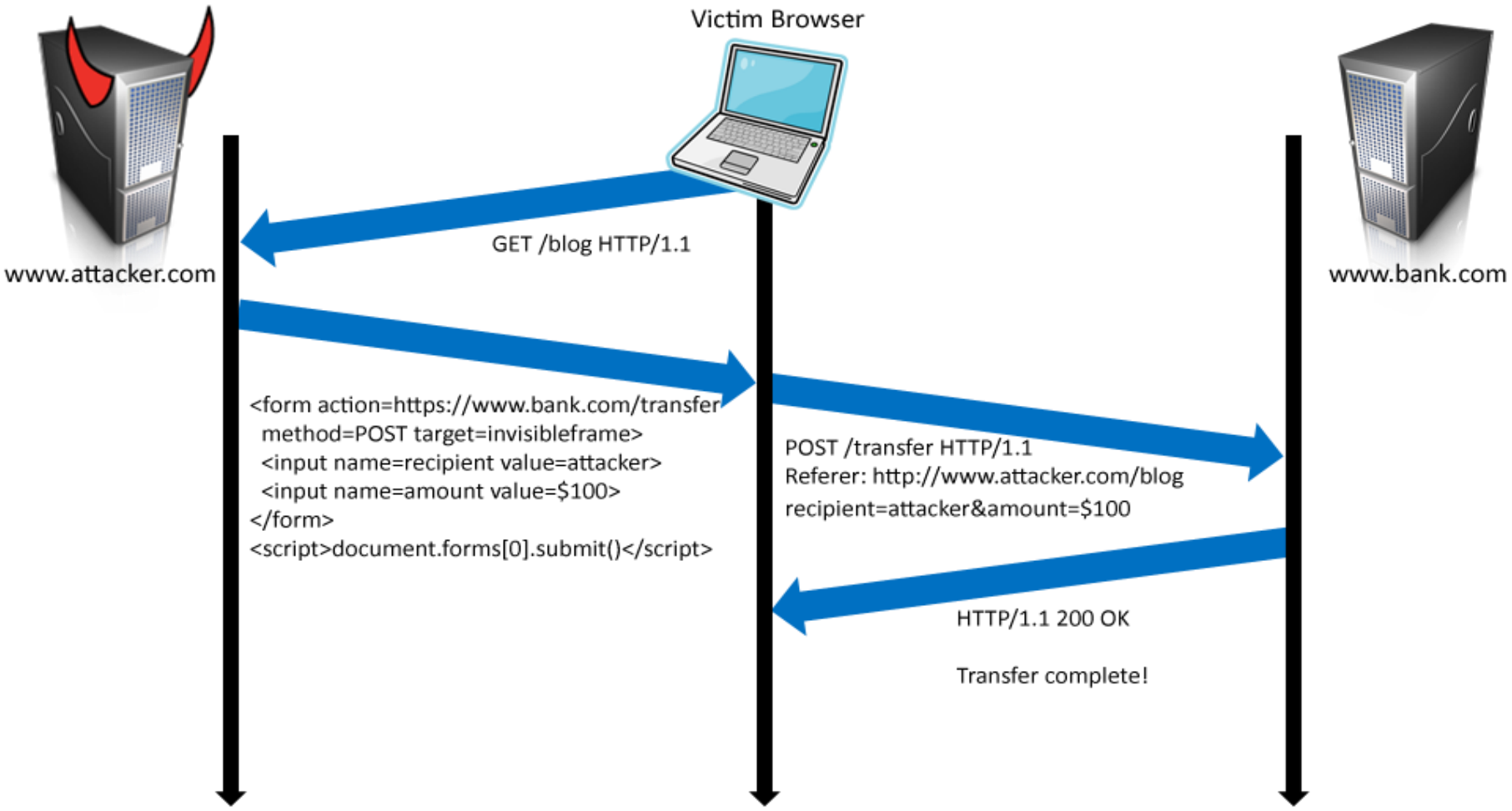
  <input name=**recipient** value=Attacker>

  <input name=**amount** value=1000$> …

  <script> *document.BillPayForm.submit();* **</script>**

- Le navigateur envoie un **cookie**, la demande de paiement est satisfaite !

CSRF: exemple 1: – Transfer d'argent (2)

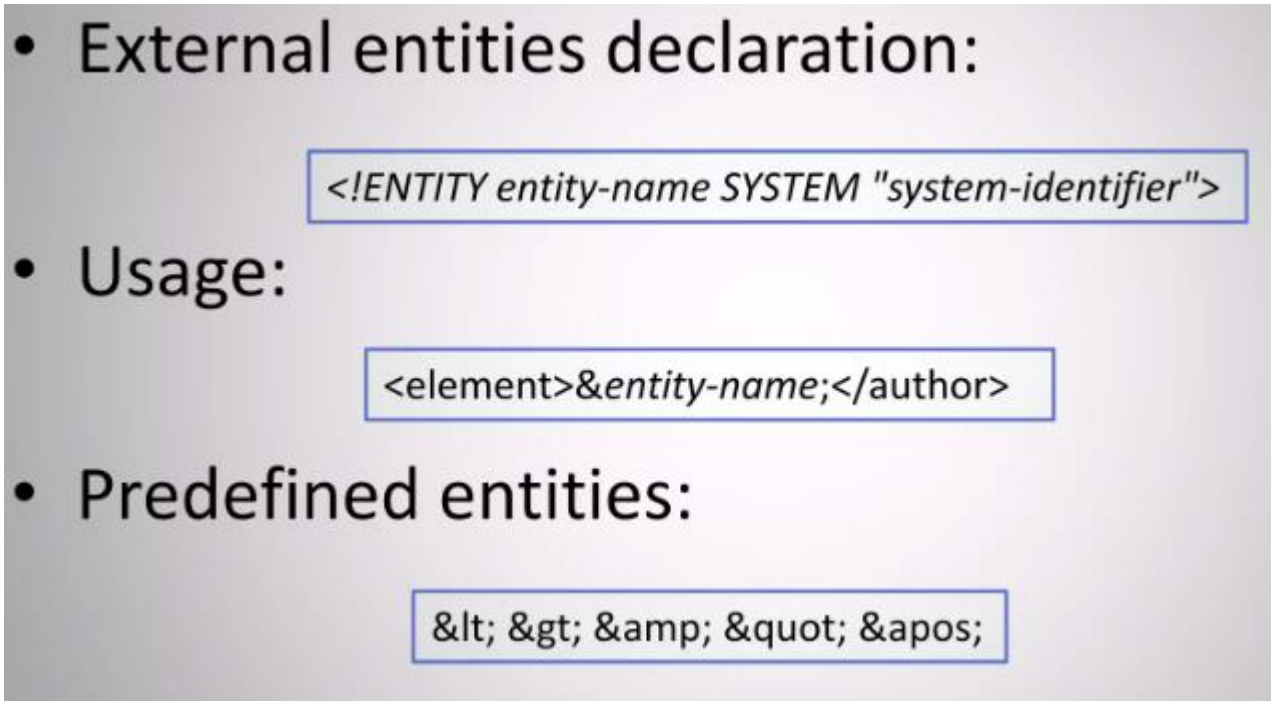


XML External Entity (XEE)

- **XML** a été conçu pour **stocker** et **transporter** les données
- Une attaque **XEE** est un type d'attaque contre une application qui analyse l'**entrée XML**.
- Elle se produit lorsque **l'entrée XML** contenant une référence à une **entité externe** est traitée par un analyseur **XML faiblement configuré**.
- Elle peut entraîner la **divulgation** de données **confidentielles** où se trouve l'analyseur **XML** et d'autres impacts du système.

XEE: Qu'est-ce que l'entité xml ? (1)

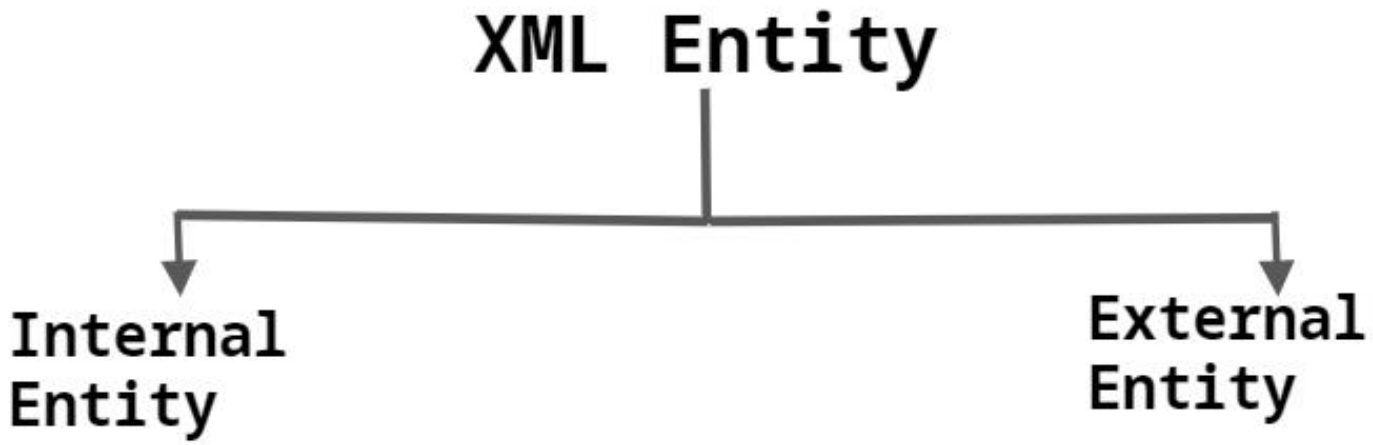
Entité signifie: déclarer un groupe d'éléments sous un **nom** afin de ne pas avoir **à réécrire** ces derniers plusieurs fois dans la DTD s'ils se répètent.



Pourquoi l'**entité** XML est dangereuse?

- Un attaquant peut inclure du contenu ***hostile*** dans un document XML.
- Peut être utilisé pour exécuter différentes ***attaques***.

XEE: Qu'est-ce que l'entité? (2)




```
<!Doctype foo[
        <!ELEMENT foo any>
        <!Entity author "john">                    <!-- internal entity -->
        <!Entity ext SYSTEM "external.dtd">        <!-- external entity -->
        <!Entity file SYSTEM "file:///etc/passwd"> <!-- external entity -->
]>
<foo>&author;&ext;</foo>
```

XEE: Que se passe-t-il pendant l'analyse du fichier?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<xmlroot><xmlEntry>&xxe;3</xmlEntry></xmlroot>
```

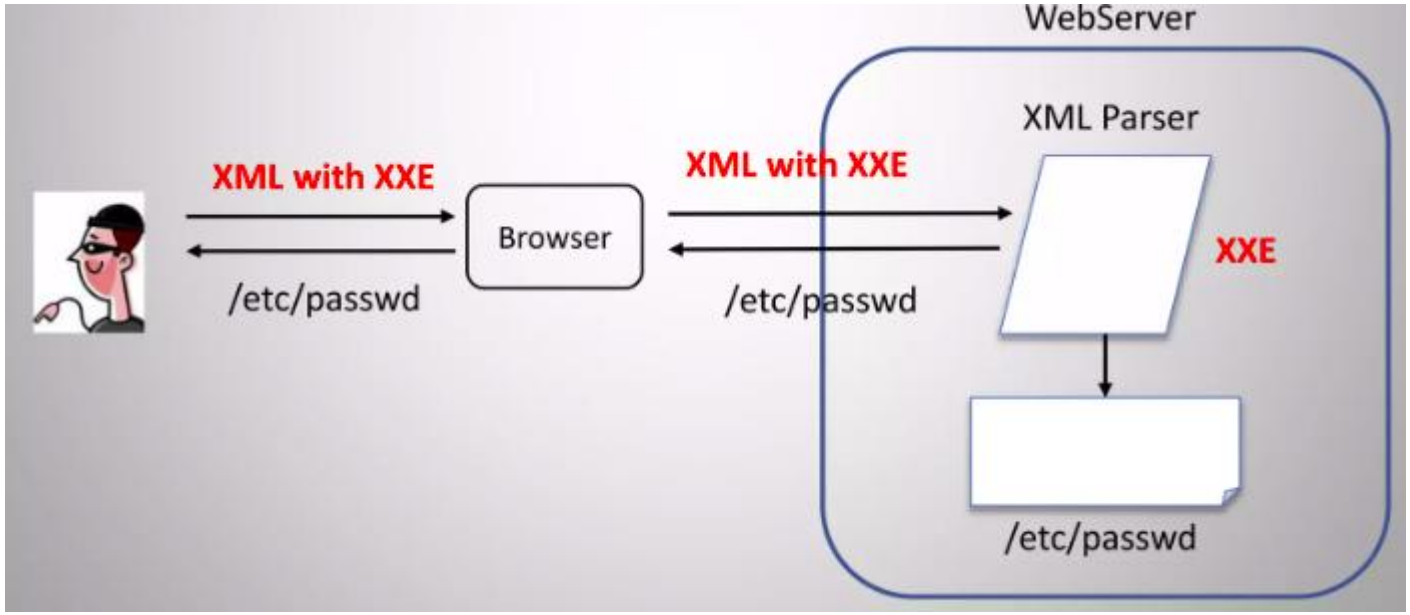
L'analyseur lit le fichier local.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "http://api.geonames.org/timezoneJSON" >]>
<xmlroot><xmlEntry>&xxe;3</xmlEntry></xmlroot>
```

L'analyseur exécute l'appel HTTP distant.

XEE: exemple de l'Attaque (1)

- L'attaque du vecteur: une application Web qui accepte l'**entrée XML** et l'analyse.
- L'attaque permet de lire un **fichier local** et d'envoyer son contenu à un **attaquant!**



Impact de l'attaque XEE

- L'impact d'une **XXE** peut être **dévastateur** : étant donné qu'elle permet d'extraire les fichiers du serveur web, elle rend possible :

- L'extraction du **code source** de l'application.

- Le vol de **clés** d'API, de **mots de passe** stockés en dur, etc.

- Le vol de **base de données.**

- Le vol de **fichiers** générés ou uploadés par d'autres **utilisateurs** et stockés sur le serveur. Cela peut être [illegible] gênant si ceux-ci contiennent des

Attaque Déni de service (DOS)

- l'intention de l'attaque **DOS** n'est pas de **compromettre** un site Web l'**intention** est simplement de le rendre **indisponible** pour les autres **utilisateurs.**
- Cela est réalisé en **inondant** le site avec le **trafic entrant**, de sorte que toutes les **ressources** du **serveur** sont **épuisées**.
- **Types de ressources**: *Bande passante*, connexions à la base de données, stockage sur disque, processeur, mémoire, threads ou ressources spécifiques à l'application
- **Ressources au niveau de l'application**:
  - Allocation/récupération d'objets **lourds**
  - Utilisation excessive de la journalisation
  - Exceptions non gérées
  - Dépendances non résolues sur d'autres systèmes
    - Services Web
    - Bases de données

Exemple d’attaques DOS

- L'attaque ***Slowloris*** ouvre de **nombreuses** connexions **HTTP** à un serveur et **maintient** ces connexions **ouvertes** en envoyer des requêtes **HTTP** partielles, **épuisant** ainsi le serveur de connexion.
- Le R-U-Dead-Yet ? **(RUDY**) : attaque envoie des requêtes **POST** sans fin à un serveur, avec des valeurs d'en-tête ***Content-Length*** arbitrairement ***longues***, pour garder le serveur **occupé** à lire des données sans signification.
- Mettre les serveurs Web **hors ligne** en exploitant des points de terminaison **HTTP** particuliers:
  - Le téléchargement de ***bombes zip*** de fichiers d'**archives corrompus** dont la taille augmente de manière exponentielle lorsqu'ils sont étendus à une fonction de téléchargement de fichiers peut ***épuiser*** l'espace disque disponible du serveur.
  - Toute URL qui effectue la **désérialisation** en convertissant le contenu des requêtes HTTP en ***objets*** de code en mémoire est également potentiellement vulnérable.

Spring sécurité ?

- **Gestion de l'authentification et des autorisations**
- **Protection contre les attaques de type:**
  - **XSS :**
  - **CSRF :**

La librairie « Spring security» propose des mécanismes d'authentification et d'autorisations

Gestion de l'authentification et des autorisations

- Le plus simple pour démarrer un projet Spring Security consiste à le charger dans une configuration Spring Boot.
  - Commencez par créer un projet Spring Boot.
  - Voici la dépendance Maven (le starter Spring Boot) à rajouter dans votre fichier pom.xml pour y injecter Spring Security.

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

Introduction Spring Security

- Il existe plusieurs types d'attaques pouvant compromettre la sécurité d'une application Web. Citons notamment :
  - Les attaques par injection SQL : on injecte dans une requête SQL un bout de code non prévu pouvant compromettre la sécurité.
  - Les attaques de type XSS (Cross-Site Scripting)
  - Les attaques de type CSRF (Cross-Site Request Forgery)
- Il est important de se prémunir de ces types d'attaques.
  - La librairie « Spring Security » propose aussi des outils qui vont dans ce sens.
  - **ATTENTION** : Spring Security ne prend pas en charge les problèmes d'injection SQL. Pour ces sujets, il faut impérativement utiliser des requêtes JPQL à base de « named parameters ».

- **L’hébergement de sites Web et Le référencement Internet**

Introduction : cycle de vie d'un site web

1) **Elaboration**, **conception** , **préparation au référencement**

2) **Hébergement** : Une fois site « **terminé** », il faut trouver un **hébergeur** afin de rendre le site **visible** sur le web.
   1) L'hébergeur **stocke** le site sur internet et attribue au propriétaire une **adresse internet** (ou **URL**).

3) **Référencement :** A cette étape, le **webmaster** s'engage à **référencier** le site et à améliorer son **positionnement** dans les **moteurs de recherche**.
   1) **Par exemple: Indexer** le site auprès des **moteurs de recherche** : suggérer le site à **Google**

4) **Mettre à jour** , **rafraîchir le site**…., **faire vivre le site web**

L'Hébergeur Web

L'hébergeur peut s'occuper d'un site à ***divers niveaux***:

- **Gestion du (des) domaine(s) (DNS)**
  - Entrées DNS à configurer
  - Sous-domaines (par ex. « sous.domaine.tn »)
- **Gestion des emails du domaine**
  - Stockage des emails
  - Redirection des emails
  - Interfaces d'accès aux emails
  - Protocoles de consultation (POP, IMAP)
- **Mise à disposition d'un serveur**
  - Espace de stockage de fichiers
  - Architecture technique (scripts, frameworks, logiciels)

Pourquoi l'hébergement Web ?

Les raisons de l'hébergement Internet :

- la nécessité de **sécuriser** le service hébergé,
- la mise à disposition par le prestataire de **ressources conséquentes** (bande passante en téléchargement,),
- le **conseil** et les **services** de support associés.

Comment héberger un site ? (1)

Pour héberger un site, il faut :

- Un **nom de domaine**
- Un serveur
- Du contenu

Nom de domaine :

Permet d'associer un **nom** facile à retenir à une **adresse IP**.

MYaPPweb.dz → **5.135.181.163** (IPv4)

MYaPPweb.dz → **2001:41d0:8:bca3::1** (IPv6)

Comment héberger un site ? (2)

Acheter un nom de domaine

- Il faut passer par un **bureau d'enregistrement** (registrar) **agréé**.
- Le **paiement** est annuel.
- Le tarif dépend du domaine de premier niveau choisi (.fr, .com,
- Certains domaines sont réservés pour des usages précis.

Choisir son nom de domaine

- Domaines **génériques** : .com, .net, .org, .pro, .edu, .name
- Domaines **géographiques** : .fr, .de, .ru, .alsace, .bzh
- Domaines **professionnels** : .aero, .archi, .xxx

Annuaire des bureaux d'enregistrement : http://s.strasweb.fr/e

Comment héberger un site ? (3)

Les hébergeurs commerciaux:

- **OVH** (ovh.com)
- **Gandi** (gandi.net)
- **1and1** (1and1.com)
- **PHPNET** (phpnet.org)
- **Dedibox** (online.net)
- **SDV Plurimédia** (sdv.fr)

Et plein d'autres...

Activité de l’hébergeur Web

La principale **activité** de l'hébergeur web consiste à :

- **Installer** les serveurs et les **sécuriser** (par une alimentation électrique **ondulée**, secourue par un **groupe électrogène**, une **salle climatisée** équipée de dispositifs anti-incendie),
- Tenir les serveur **à jour** en installant les **mises à jour** de sécurité pour éviter les at**taques malveillantes**.
- **Réparer** les serveurs en **cas de panne**,
- Installer les **technologies logicielles** souhaitées par les **clients** ou qu'il souhaite leur offrir (comme les langages de programmation et les modules supplémentaires).

Types d'hébergement

Dédié (dedicated)

- Une machine entière à disposition
- Plus rapide, plus disponible, plus de stockage
- Plus cher

Mutualisé (mutualized)

- Une « portion » de machine à disposition
- Moins rapide, plus de risques de sécurité
- Beaucoup moins cher

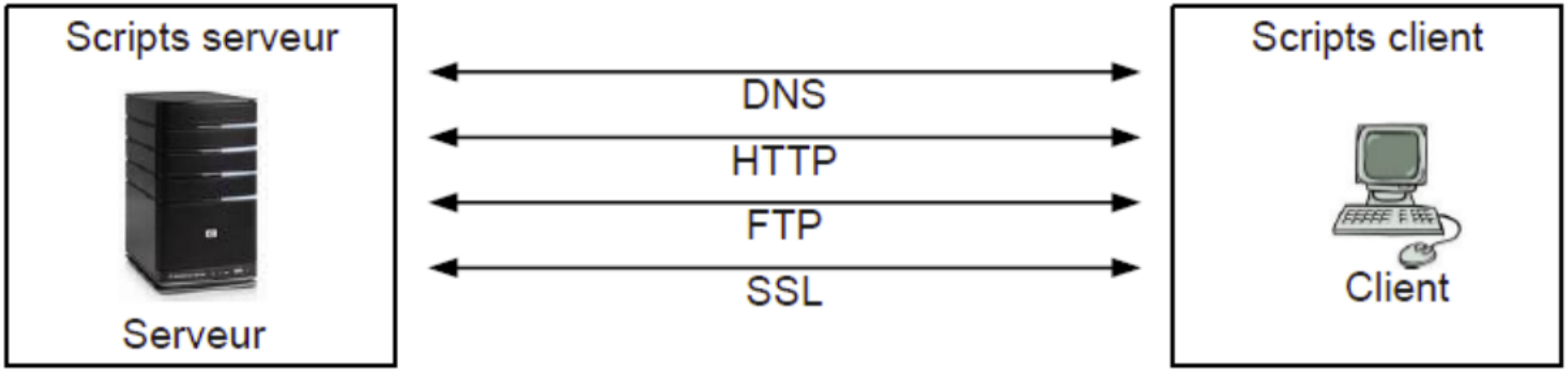
Colocation

- La machine appartient au client
- L'hébergeur fournit l'infrastructure, le réseau, la maintenance

Le serveur Web

Serveur, propose des services « orientés web »

- **DNS** : noms de domaine
- **HTTP** : pages web (HTML, XML, JSON, etc.)
- **FTP** : serveur de fichiers
- **SSL** : transactions cryptées
- Principe **client / serveur**



- *Comment font les **internautes** pour **trouver** le site web?*

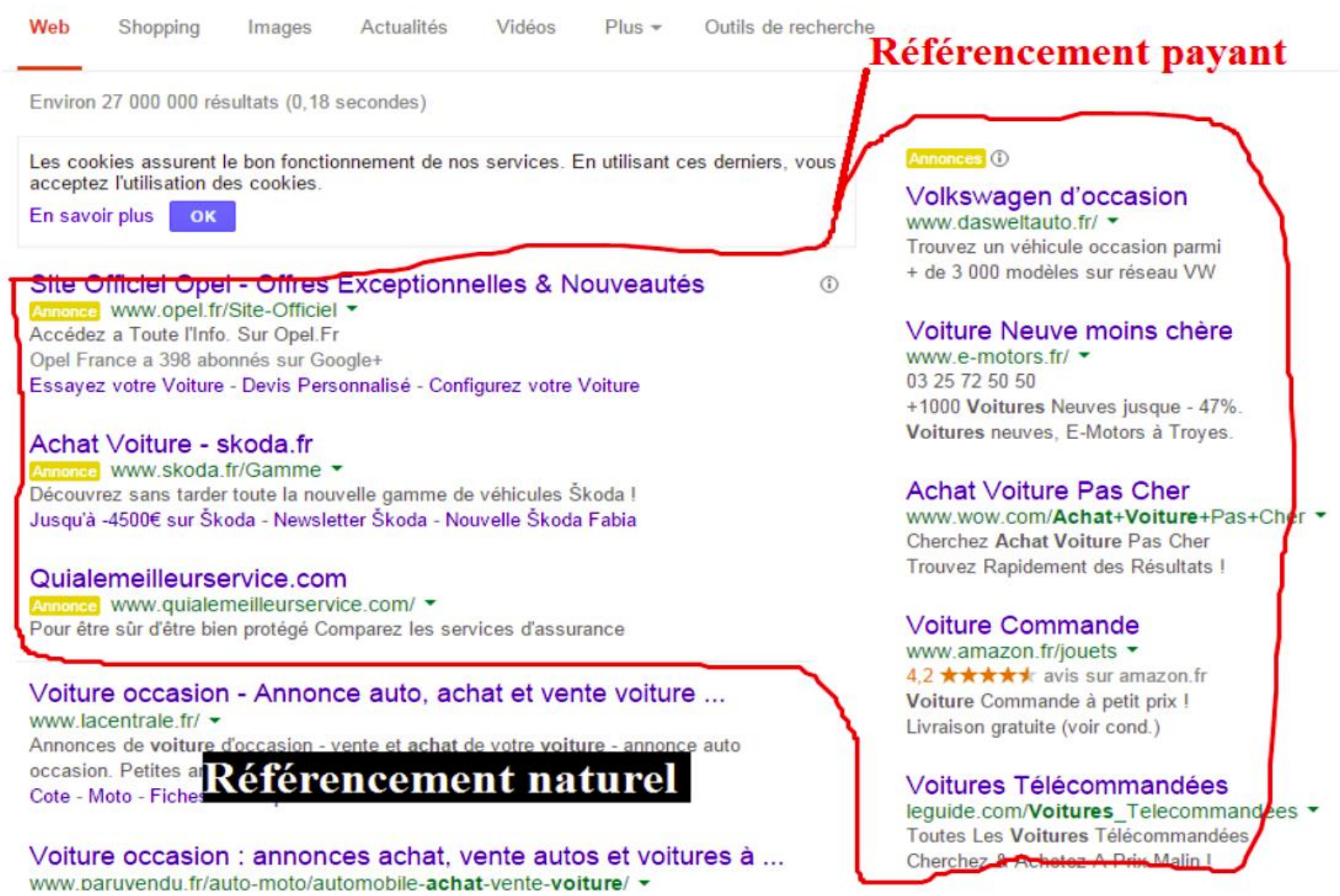
- *Quelle est l'importance du **positionnement** du site web?*

Définition:

- Le référencement est le fait d'**indexer** un **site web** dans les **moteurs de recherche** ou dans les annuaires.

- Par extension: l'ensemble des actions et techniques visant **à améliorer la position du site internet** dans ces **résultats de recherche** (positionnement) et à en optimiser la **visibilité**.

Types de Référencement



Types de Référencement

Référencement naturel

• Toutes les **techniques** utilisées pour améliorer le **positionnement** du site dans les **moteurs de recherche** et accroitre sa popularité, c'est un effort de **conception et de développement** du site

Référencement payant

• L'utilisation de **publicités** comme les campagnes **GoogleAdwords**, achat de bannières

Domaines de Référencement

- **SEA (Search Engine Advertising)** : achat de liens ***sponsorisés*** afin de se positionner sur des mots clés spécifiques.

- **SEM (Search Engine Marketing)** : mise en place de liens sponsorisés, notamment sur les moteurs de recherche, afin de gagner en **visibilité** tout en augmentant son trafic.

- **SEO (Search Engine Optimisation)** : optimisation d'une page ou d'un site pour les moteurs de recherche destinée à faire remonter la page ou le site dans les pages de résultats.

Conclusion Référencement

- L'**hébergement** du site web doit s'accompagner par un effort de **référencement** pour augmenter la **visibilité** du site.

- Le référencement web est tout une **stratégie** qui doit être suivie dès la **conception** du site web.

- Les administrateurs des sites s'engagent à assurer la **mise à jour** régulière et la meilleure qualité de leurs ressources en ligne: Impératifs de qualité

Bibliographie (1)

1- Web Application Security: Exploitation and Counter measures for Modern Web Applications
1rst edition, Andrew Hoffman ,Oreilly editions , 2020.

2- Web Security for Developers: Real Threats, Practical Defense, Malcolm McDonald , No Starch Press ,2020.

3-https://www.bezkoder.com/spring-boot-security-postgresql-jwt-authentication/

4- https://www.bezkoder.com/angular-spring-boot-jwt-auth/

5- https://owasp.org/ : Open Web Application Security Project,

Bibliographie (2)

6- INJECTION SQL ÉNS Vieux Kouba Département d'Info, Présenté par : -Bentami Billel - Mesbaiah Ahmed Dhia Eddine Avril 2017

7- Cours CSS, Moutasem Hamour, New York Institute of Technology (NYIT)-Amman.

8- Security of Web Applications, Vitaly Shmatikov

9- https://slideplayer.fr/slide/14876272/

10- Cours Emna ABIDI, conception d'un site web marchant, url: https://www.nassimbahri.ovh/docs/conception/Chapitre-4.pdf

11- https://programmingtechie.com/2020/10/10/deploy-spring-boot-and-angular-application-to-heroku/

12- https://blog.alphorm.com/reseaux-docker-guide-complet

13- https://blog.stephane-robert.info/docs/conteneurs/orchestrateurs/kubernetes/