PRÉPARER UN ENVIRONNEMENT DE TEST

Titre Professionnel : Administrateur Système DevOps (niveau 6)

INSTALLER ET CONFIGURER SON ENVIRONNEMENT DE TRAVAIL EN FONCTION DU PROJET

Titre Professionnel : Concepteur Développeur d'Applications (niveau 6)

INSTALLER ET CONFIGURER SON ENVIRONNEMENT DE TRAVAIL EN FONCTION DU PROJET WEB OU WEB MOBILE

Titre Professionnel : Développeur Web et Web Mobile (niveau 5)

GESTION DE VERSIONS

Version	Date	Rédacteur	Description
1.0	01/08/2023	Lesly	Installer et configurer son environnement de travail

ARCHITECTURE CLIENT SERVEUR

GÉNÉRALITÉS

- Contrairement à une **application classique** qui fonctionne sur un modèle d'architecture dit **"client lourd"**, une **application web** fonctionne sur le modèle **"client/serveur"**.
- Cela veux dire qu'une partie de l'application s'exécute sur l'ordinateur **client,** et une autre partie sur l'ordinateur **serveur.**
- Internet et toutes ses applications fonctionnent suivant ce modèle, sinon on devrait stocker tout Internet sur son disque dur.

GÉNÉRALITÉS

- La partie la plus lourde de l'application est prise en charge par le **serveur** ou **"back-end"**.
- Le **client** ou **"front-end"** se charge de recevoir les données déjà traitées par le serveur et d'afficher une interface pour les présenter à l'utilisateur.
- Le client ne stockant pas toute l'application, est donc **"léger"**, contrairement aux applications **"client lourd"** telles que Photoshop ou Excel.

GÉNÉRALITÉS

- En tant que développeur d'applications web il est très important de comprendre qu'une application s'exécute dans **deux environnements distincts** et **séparés** (physiquement parlant).
- Ainsi, Ryan (le client) peut habiter à Toulon et le serveur peut être installé à Paris. Pourtant Ryan est bien sur la même application !

GÉNÉRALITÉS

Exemples d'applications "client lourd" :

- Jeux vidéo 3D
- Navigateur web
- Logiciel de traitement d'image
- Logiciel bureautique

Exemple d'applications "client serveur" :

- Site web
- Site e-commerce
- Moteur de recherche web
- Réseau social
- Plateforme de streaming

EN RÉSUMÉ…

- Le client **envoie une requête** au serveur (exemple : envoie-moi le contenu de la page https://www.google.fr/)
- Le serveur effectue une série de traitements puis **renvoie la page demandée** au client.
- Le client **réceptionne la page** et effectue à son tour une **série de traitements** afin de l'afficher correctement à l'utilisateur.

CÔTÉ CLIENT : LE FRONT-END

CÔTÉ CLIENT : LE FRONT-END

- En développement web, le client est généralement le **navigateur web**.
- C'est l'utilisateur qui contrôle le navigateur, mais le logiciel qui émet la requête vers le serveur, c'est bien le navigateur web.
- Dans ce cas, quand on parle de développement côté client, on parle de **front-end**.
- On peux alors diviser cette première partie en **trois grandes catégories de langages** :
  - le HTML,
  - le CSS et
  - le JavaScript.

LE LANGAGE HTML

- En navigant sur Internet, le navigateur émet une **requête** et reçoit une ou plusieurs **réponses** : par exemple une page **HTML.**
- HTML veut dire **HyperText Markup Language** (langage de balises hyper texte) **:** c'est un langage de **balisage.**
- Le navigateur dispose d'outils (comme un **moteur de rendu**) qui transforme le code HTML en un document facile à lire pour l'utilisateur et communément appelé **page web** !

LE LANGAGE HTML

- Pour voir à quoi ressemble le contenu d'une page web, on peut inspecter son **code source** et découvrir sa version HTML :
  - Appuyer sur la touche **F12** ou bien
  - **Clic droit ➔ Inspecter**

```
<HEAD>
  <TITLE>Ma Page HTML avec du CSS</TITLE>
  <!-- On écrit dans l'en-tête de la page les styles à appliquer à cette page (uniquement) --
  <Style type="text/css">
      body {
          background-color : black ;
      }
      h1 {
          color : yellow ;
          text-align : center ;
      }
  </Style>
  <!-- Fin des styles -->
</HEAD>
<BODY>
  <h1>Ma Page HTML avec du CSS</h1>
</BODY>
```

LE LANGAGE HTML

- HTML n'est pas un **langage de programmation** à proprement parler car il ne permet pas de faire **d'algorithmie.**
- Il permet uniquement de **structurer des documents** (HTML et XHTML) et des **données** (XML).
- Sa version actuelle est la **version 5,** on parle alors de HTML5.
- Cette version a apporté de nombreuses avancées et possibilités par rapport aux anciennes versions du HTML :
  - Balises sémantiques,
  - Nouveaux contrôles de formulaire,
  - API...

LE LANGAGE CSS

- Un autre aspect important à considérer côté client est la **gestion du style** ou mise en forme.
- Pour cela on utilise un second langage : le langage **CSS**.
- CSS veut dire **Cascading Style Sheets** (pour feuilles de style en cascade).
- HTML permet de structurer les informations, tandis que CSS va permettre de lui donner un style graphique.
- Par exemple :
  - la gestion des couleurs,
  - la taille des polices ou images,
  - ou bien encore la forme des menus de l'application.

LE LANGAGE CSS

La logique du langage CSS est assez simple, il utilise :

- des **sélecteurs,** qui permettent de cibler une partie du code HTML (toutes les balises **<p>**, par exemple)
- des **propriétés,** qui vont permettre de cibler le type de modifications que l'on souhaite appliquer (**border** pour modifier la couleur et la taille d'une bordure, par exemple)
- des **valeurs,** qui vont permettre de changer les valeurs par défaut (**border:1px black solid** passe la couleur d'une bordure en noir et met sa taille à 1 pixel, par exemple)

LE LANGAGE CSS

- CSS n'est pas un langage de programmation mais plutôt un langage de **présentation.**

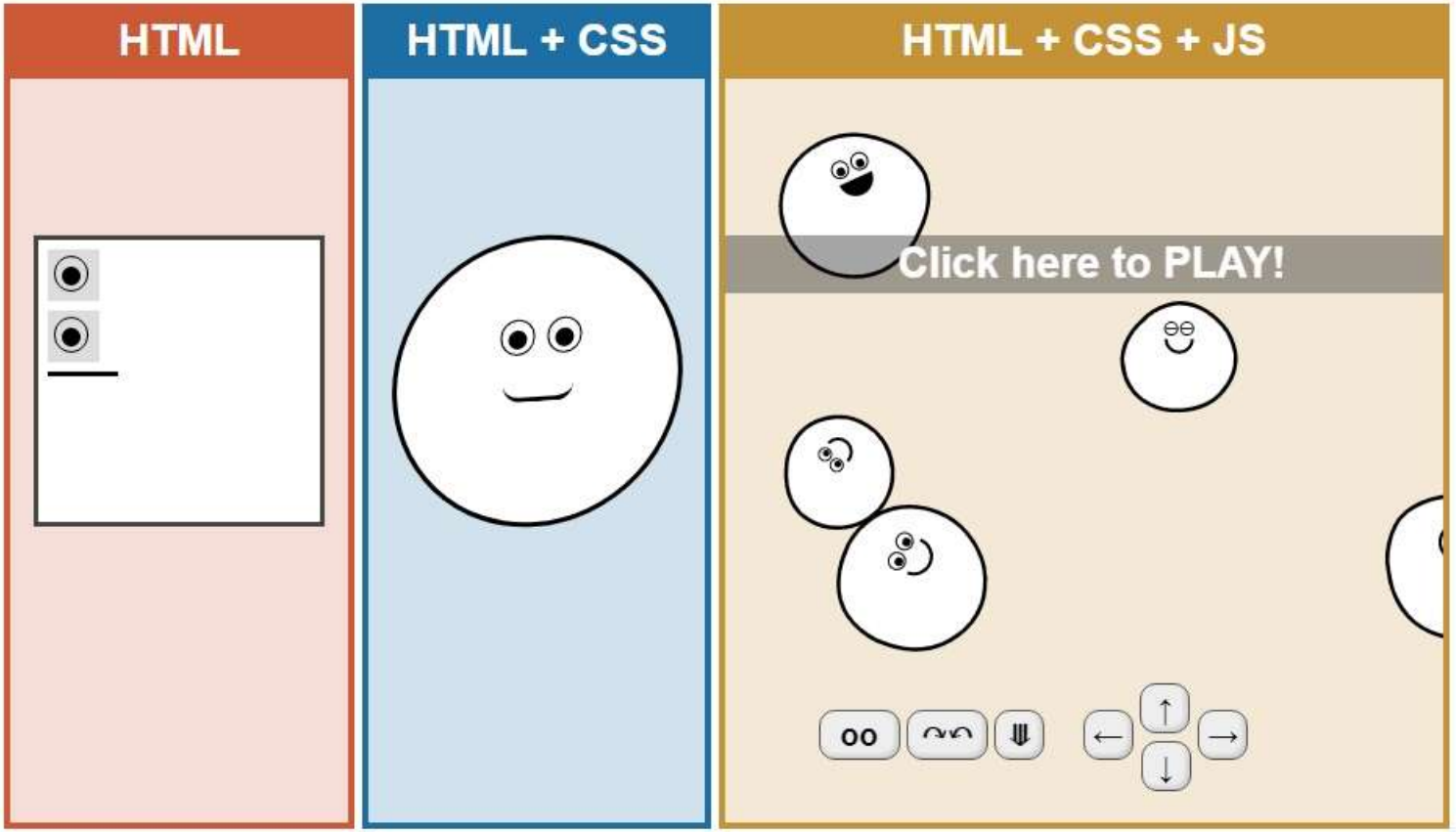
LE LANGAGE JAVASCRIPT

- **JavaScript** (JS pour les intimes) est le seul **langage de programmation** utilisable côté client.
- Il permet d'écrire des **algorithmes** qui seront exécutés par le navigateur.
- À son apparition en 1995, c'était un **petit langage** "utilitaire" qui permettait de faire un peu de programmation dans les pages web.
- Depuis 2005, c'est devenu un langages **incontournable** dans le développement web (y compris côté back).

LE LANGAGE JAVASCRIPT

- JavaScript est devenu **puissant** et permet de faire absolument tout ce que l'on souhaite au sein d'une page web :
  - Animations visuelles,
  - Glisser et déposer,
  - Géolocalisation,
  - Contrôles de saisies de l'utilisateur dans un formulaire,
  - Stockage local,
  - Communication en temps réel avec le serveur...
- C'est donc JavaScript qui va **dynamiser les pages web** côté client.

LE LANGAGE JAVASCRIPT



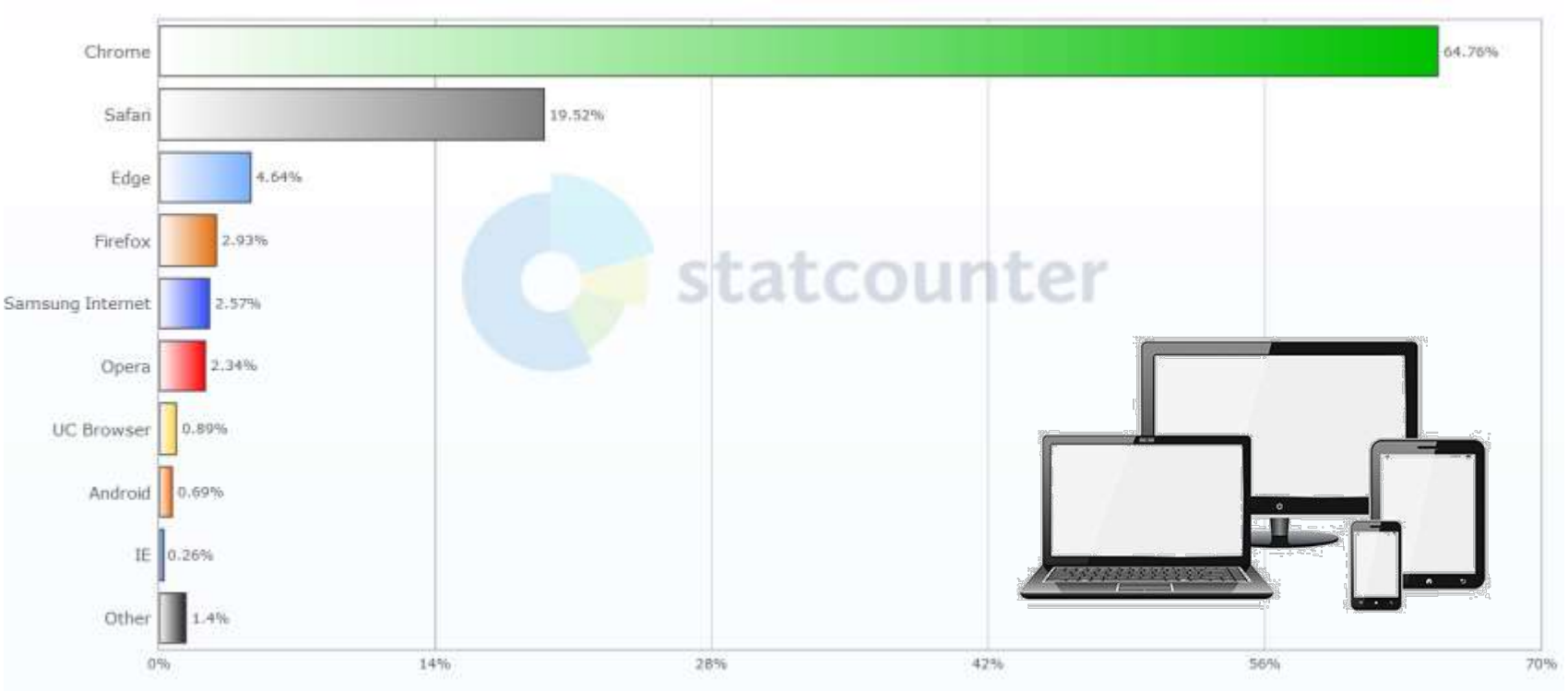
LES DIFFÉRENTS SUPPORTS

- Une première particularité du développement web, c'est que la même application doit fonctionner sur des **supports différents.**
- En effet on distingue plusieurs navigateurs :
  - **Google Chrome :** 64,8 % de parts de marché
  - **Safari :** 19,5 % de parts de marché
  - **Microsoft Edge :** 4,6 % de parts de marché
  - **Mozilla Firefox :** 2,9 % de parts de marché
  - **Samsung Internet :** 2,6 % de parts de marché
- Ils fonctionnent tous de la même manière malgré **quelques différences.**
- C'est au développeur de faire en sorte que l'application se comporte bien sur ces différents navigateurs.

LES DIFFÉRENTS SUPPORTS

- Une seconde particularité du développement web concerne les **différents périphériques.**
- Il y a encore quelques années, seul un **ordinateur de bureau** (desktop) permettait de naviguer sur le web.
- Désormais, avec l'apparition des **smartphones** et autres **tablettes,** une application doit fonctionner aussi bien sur un ordinateur de bureau à haute résolution que sur un petit écran de smartphone bon marché.
- Ce sont les utilisateurs qui décident et les développeurs qui s'adaptent.
- C'est ce que l'on appelle le **Responsive Web Design** (ou RWD) : une application aura un rendu différent selon le périphérique qui l'exécute.

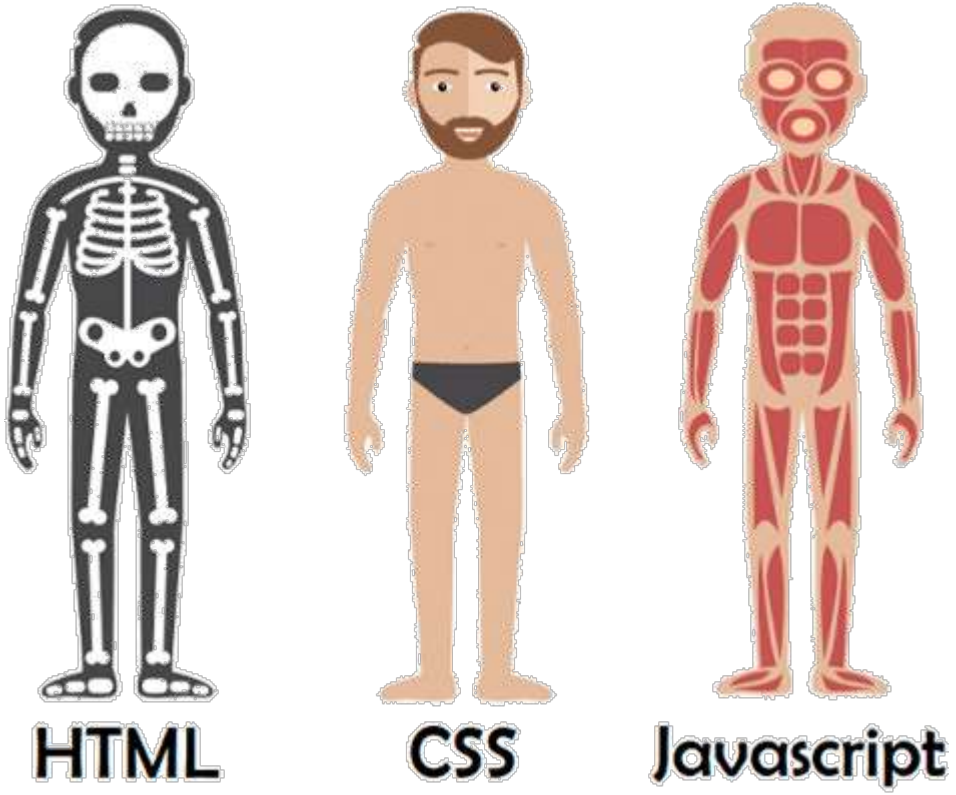
LES DIFFÉRENTS SUPPORTS



Source : https://gs.statcounter.com/

EN RÉSUMÉ…

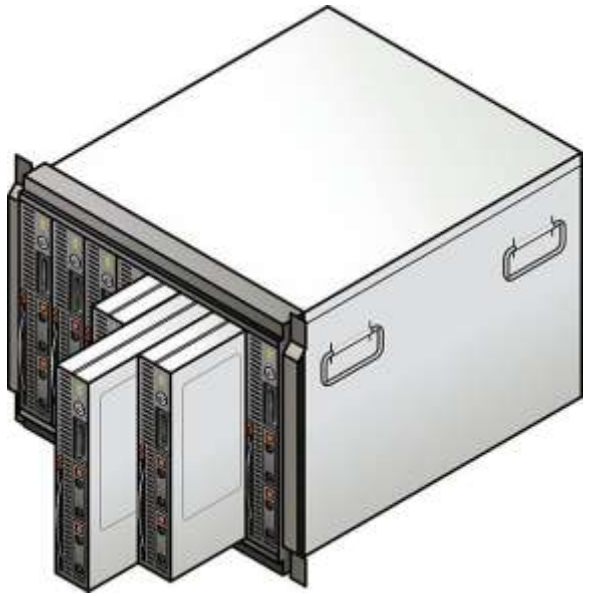
- **HTML** : structuration
- **CSS** : style
- **JavaScript** : programmation



CÔTÉ SERVEUR : LE BACK-END

CÔTÉ SERVEUR : LE BACK-END

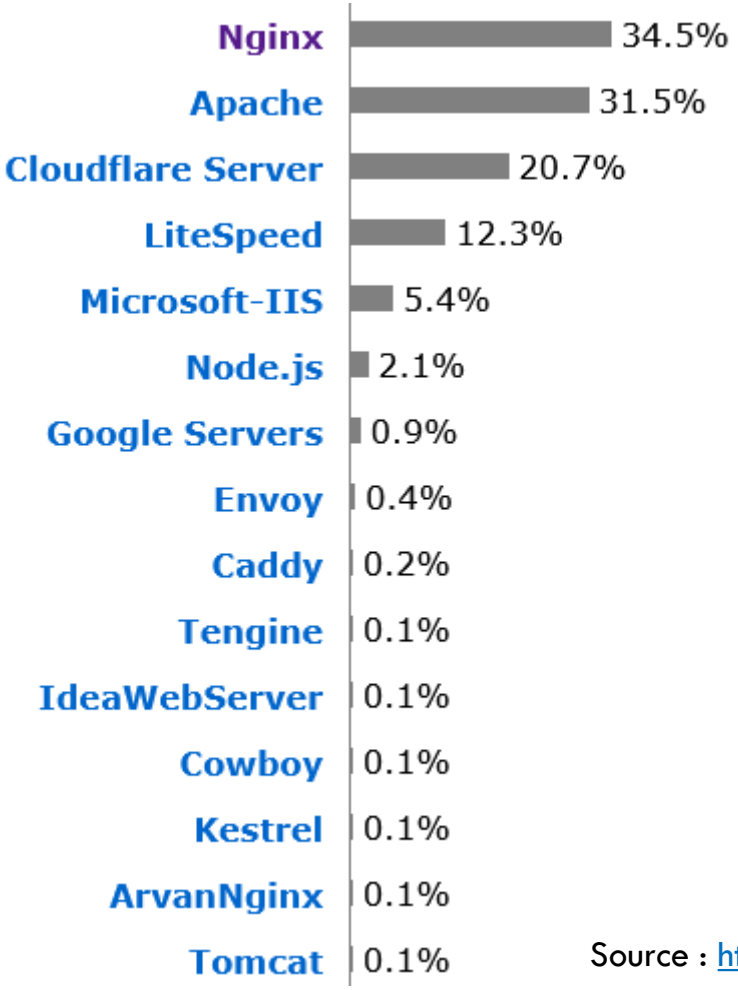
- Pour rappel, un **serveur** n'est rien d'autre qu'un ordinateur qui traite des demandes provenant de différents clients et qui leur renvoie une réponse.
- Techniquement, n'importe quel ordinateur peux être un serveur : le vôtre par exemple !
- La différence réside dans le **rôle** qu'il va remplir et dans les **logiciels** qui y sont installés.
- En pratique, un serveur est un des nombreux **ordinateurs empilés** dans les **datacenters** qui traitent toutes les demandes des clients via une connexion Internet.



CÔTÉ SERVEUR : LE BACK-END

- Une fois que le réseau a transmis la requête du client au bon serveur, via l'adresse IP et les DNS, le **serveur transfère la requête au logiciel** prévu à cet effet.
- **Attention !** Par abus de langage, quand on parle de **serveur web**, on parle à la fois de la machine physique et du logiciel qui traite les requêtes.
- Voici quelques logiciels populaires remplissant le rôle de serveurs web :
  - **Nginx :** 34.5%
  - **Apache :** 31.5%
  - **Cloudflare Server :** 20.7%
  - **LiteSpeed :** 12.3%
  - **Microsoft-IIS :** 5.4%

CÔTÉ SERVEUR : LE BACK-END



Source : https://w3techs.com/

CÔTÉ SERVEUR : LE BACK-END

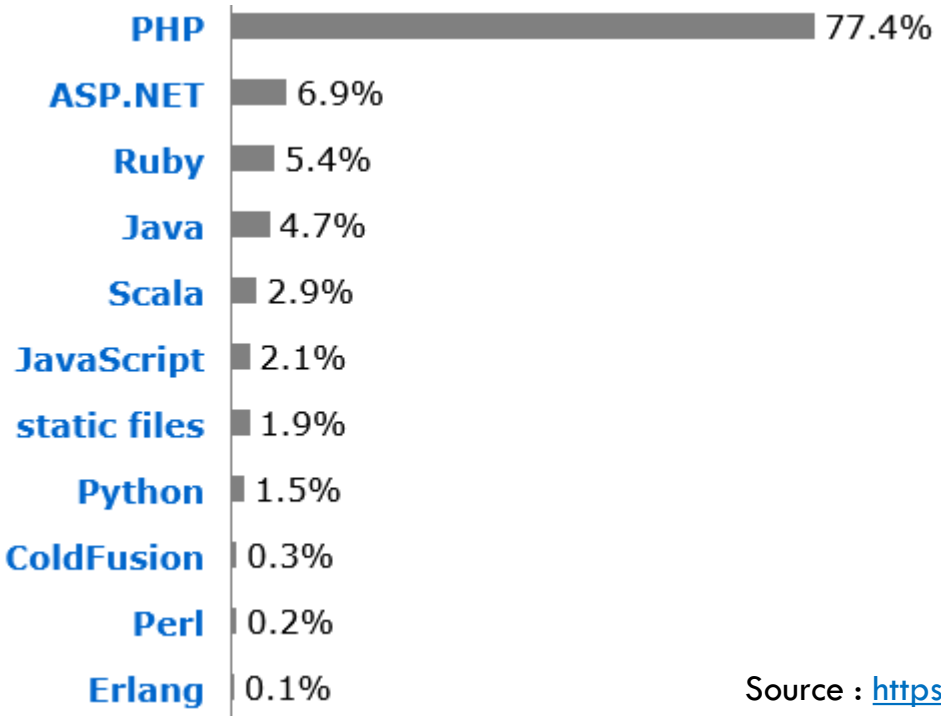
- Le serveur web est un **point d'entrée** de l'information.
- Suivant la manière dont il a été configuré, une **série de traitements** va être appliquée à la requête **via l'application** écrite par le développeur.
- Le serveur retournera enfin une réponse en fonction : c'est ce que l'on appelle le **"back-end"** en anglais.
- Le back-end comporte de multiples facettes :
  - Administration serveur,
  - Développement logiciel,
  - Gestion et exploitation d'une base de données,
  - Etc...

CÔTÉ SERVEUR : LE BACK-END

- Si on se focalise sur le développement pur, il existe de très nombreux **langages de programmation** permettant de réaliser la partie back-end d'une application :
  - **PHP :** 77.4%
  - **ASP.NET :** 6.9%
  - **Ruby :** 5.4%
  - **Java :** 4.7%
  - **Scala :** 2.9%



CÔTÉ SERVEUR : LE BACK-END



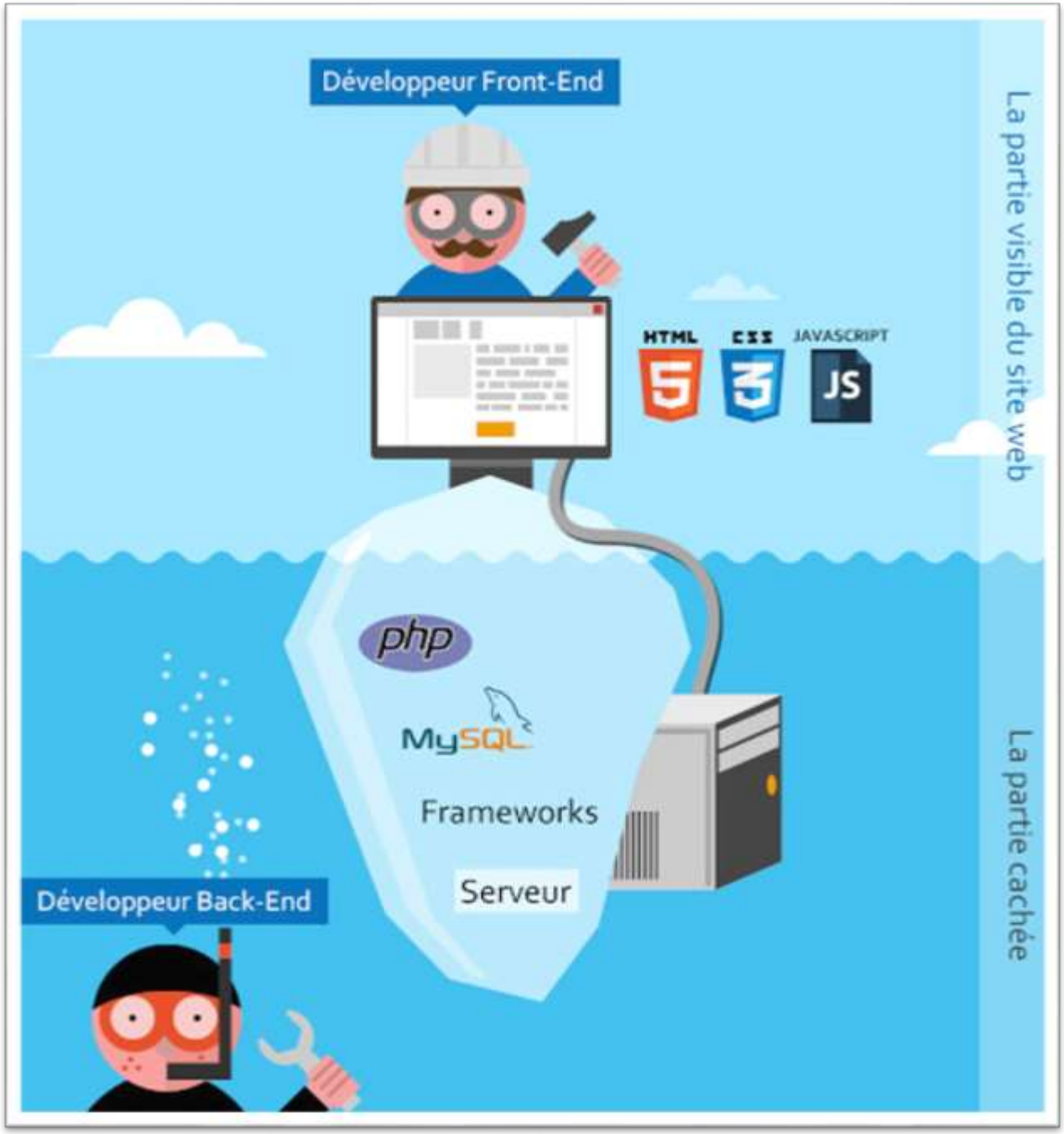
Source : https://w3techs.com/

CÔTÉ SERVEUR : LE BACK-END

- Il n'y a pas de **pire ou meilleur langage** : ils ont tous leurs avantages et leurs inconvénients.
- Ils sont tous répandus et appréciés selon divers **secteurs d'activités** et selon la **nature du projet** à réaliser.
- Par exemple on n'utilisera pas :
  - Les mêmes technos si on travaille sur le projet d'une banque ou sur le site web d'une petite association locale.
  - Les mêmes technos si on a 50 visiteurs par jour ou 100000.
  - Les mêmes technos pour un service de streaming ou un site e-commerce.
- Enfin, ce ne sont **pas des choix purement techniques** ou liés à la performance des différents langages qui déterminent la décision finale !

EN RÉSUMÉ...

- Si le front-end est plus axé sur l'apparence et l'interface utilisateur, le back-end lui est plutôt axé sur la **mécanique** et le **comportement** de l'application.
- C'est dans le back-end que l'on exploite la **base de données** et que l'on définit la **logique métier,** en POO (Programmation Orientée Objet) par exemple.
- Enfin, il n'y a **pas d'affichage** à proprement parler : c'est pour cela qu'il n'y a pas de langages de balisage ou de présentation côté back-end.


Développeur Front-End
HTML
CSS
JAVASCRIPT
La partie visible du site web
php
MySQL
Frameworks
Serveur
La partie cachée
Développeur Back-End

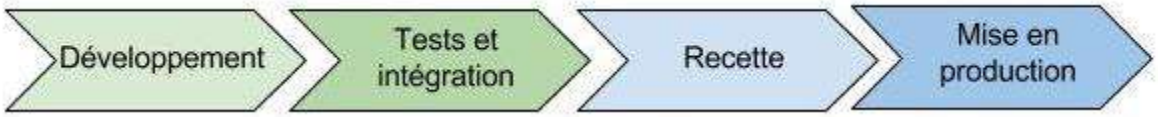
ENVIRONNEMENT DE DÉVELOPPEMENT ET EDI

QU'EST-CE QU'UN ENVIRONNEMENT DE DÉVELOPPEMENT ?

- C'est **l'espace de travail** permettant aux développeurs de construire son application ou d'y apporter des modifications sans affecter la version active du produit logiciel.
- Les **modifications** peuvent inclure :
  - la maintenance (mise à jour des versions logicielles),
  - le débogage (chasse aux erreurs) et
  - les correctifs.
- Chaque site web ou application est hébergé sur un serveur et chaque serveur a des **dépendances** matérielles et logicielles bien précises.
- Il peut donc exister des **centaines de combinaisons** différentes pour **configurer un serveur** et l'adapter aux spécificités d'un site ou d'une application.

QU'EST-CE QU'UN ENVIRONNEMENT DE DÉVELOPPEMENT ?

- Lorsqu'un serveur est paramétré correctement et que le site ou l'application fonctionne parfaitement, on le définit comme **"environnement"**.
- Il peut donc exister, dans la vie d'un site ou d'une application, **plusieurs environnements** en fonction des besoins spécifiques.
- On parle alors d'environnement **local**, de **développement**, de **testing**, de **staging** et enfin de **production**.
- Tous ces environnements sont complètement **indépendants** les uns des autres.
- Bien entendu, chaque projet ne nécessite pas autant d'environnements et le minimum serait d'avoir un environnement local, un staging et une production.



DIFFÉRENTS TYPES D'ENVIRONNEMENT DE DÉVELOPPEMENT

Environnement local :

- Généralement installé **sur l'ordinateur du développeur,** il contient toutes les dernières itérations du code sur lequel on travaille.
- Les nouvelles fonctionnalités sont testées dans un premier temps à ce niveau.

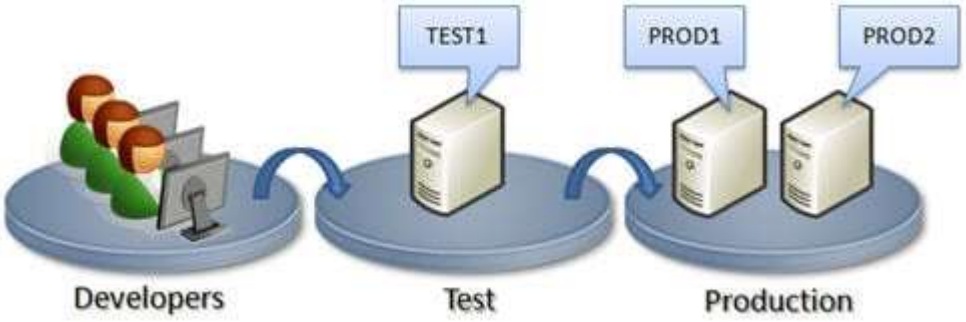
Environnement de développement :

- Les développements locaux sont intégrés sur le serveur et sont testés dans un environnement serveur et **en ligne.**
- Le seul but est de valider le passage d'une technologie sur serveur en ligne.
- Les bugs inhérents à cette étape seront corrigés et ne seront pas propagés aux autres environnements.

DIFFÉRENTS TYPES D'ENVIRONNEMENT DE DÉVELOPPEMENT

Environnement de testing :

- Permet de **tester les fonctionnalités** qui peuvent être développées en parallèle sur ce que l'on appelle les différentes "branches".
- On peut donc **passer d'une branche à une autre** pour tester les différentes versions de code ou les différentes fonctionnalités.



DIFFÉRENTS TYPES D'ENVIRONNEMENT DE DÉVELOPPEMENT

Environnement de staging :

- C'est la version juste avant la livraison : l'interface "officielle" de validation qui sert de pont entre les développements et le site en production.
- À ce stade, aucune erreur ne doit subsister et les modifications doivent être prêtes à être déployées vers la production.
- Cet environnement doit ressembler en tout point à la production : on l'appelle aussi **"pré-prod"**.

Environnement de production :

- Comprend la **version finalisée et validée**.
- Cette version sans bugs offre une expérience-utilisateur optimale.

AVANTAGES D'UN ENVIRONNEMENT DE DÉVELOPPEMENT

Rationaliser le flux de travail :

- Aide les développeurs à configurer facilement chaque **outil de développement** web sans avoir à le paramétrer séparément :
  - Outils populaires : GitHub, Chrome Developer Tools, Sublime Text, Visual Studio Code, Gestionnaire de paquets Node (npm), Sass, Bootstrap, Grunt, Ruby on Rails
  - Outils avancés : Postman, Docker, Kubernetes, NGINX, Flutter, ReactJS, Angular, Vue.js, Symfony

Minimiser les erreurs potentielles :

- Grâce à **l'environnement de test,** les développeurs peuvent **tester le code source,** ce qui facilite la détection et la correction des erreurs.

AVANTAGES D'UN ENVIRONNEMENT DE DÉVELOPPEMENT

Améliorer la productivité :

- Le processus de développement est plus simple, ce qui permet aux développeurs d'effectuer **plusieurs tâches** plus rapidement et plus efficacement.
- Par exemple, ils peuvent analyser le code et la syntaxe pendant l'édition.

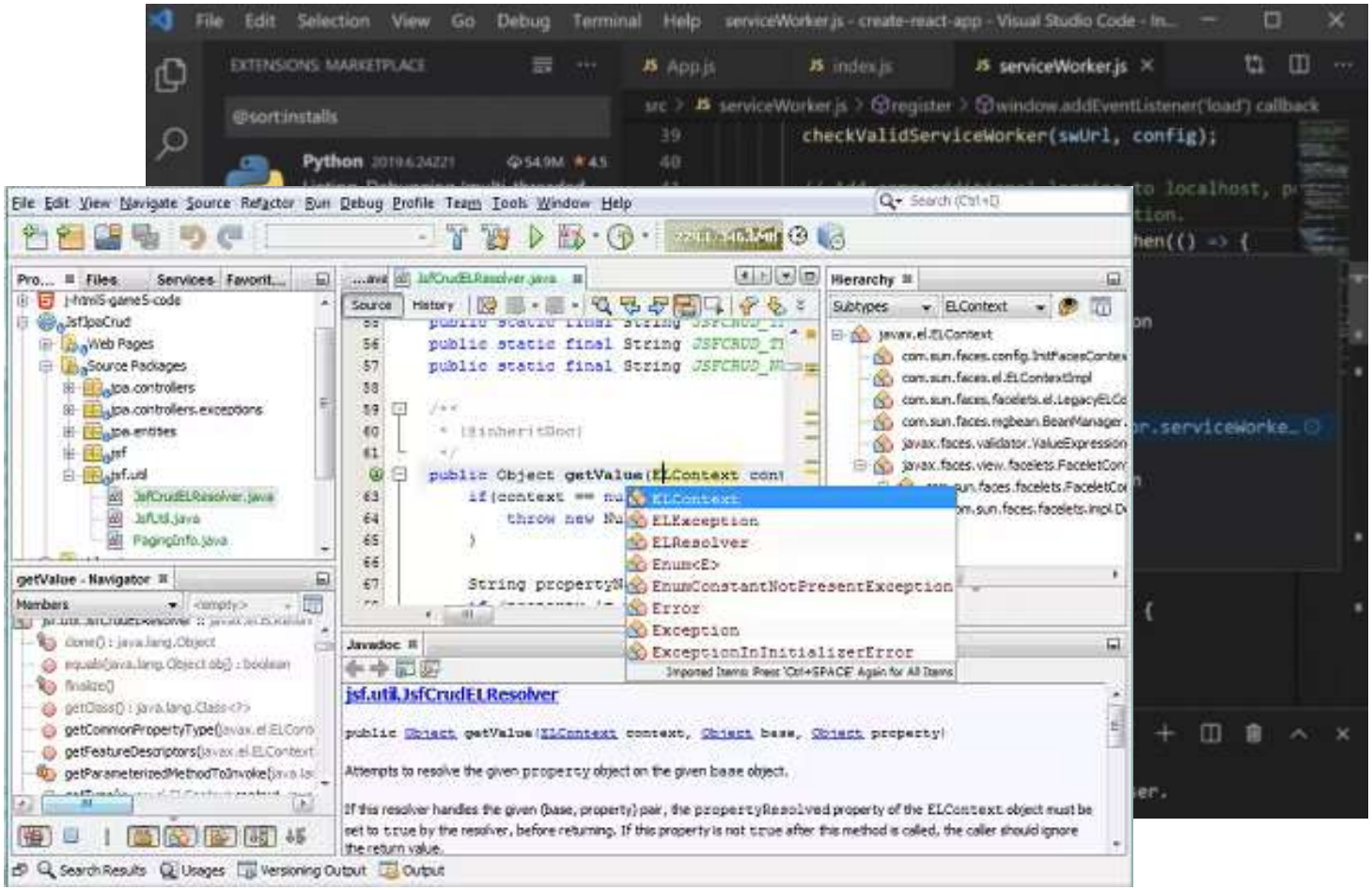
Normaliser le processus de développement :

- L'utilisation d'une interface d'environnement de développement permet à plusieurs développeurs de **collaborer** et de gagner du temps.

QU'EST-CE QU'UN ENVIRONNEMENT DE DÉVELOPPEMENT INTÉGRÉ (EDI) ?

- C'est une **suite logicielle** qui combine tous les outils de développement en une seule **interface utilisateur graphique** (GUI).
- Il rend le processus de développement plus **efficace** et plus **rapide**.
- On peut citer quelques exemples d'environnements de développement intégrés populaires tels que NetBeans, Microsoft Visual Studio ou Eclipse.

QU'EST-CE QU'UN ENVIRONNEMENT DE DÉVELOPPEMENT INTÉGRÉ (EDI) ?



PRINCIPALES CARACTÉRISTIQUES D'UN EDI

Éditeur de code :

- Principalement utilisés pour **écrire et modifier le code source,** de nombreux EDI intègrent également un éditeur de texte mettant en évidence les mots-clés et les erreurs de syntaxe.

Complétion de code :

- Fonction utilisée pour analyser l'ensemble du code afin d'identifier et **d'insérer les composants de code manquants** : utilisation permet de gagner du temps à l'écriture du code et de minimiser les erreurs de syntaxe.

Compilateur :

- Traduit le **langage de programmation** comme JavaScript ou Python **en code machine** afin qu'un ordinateur puisse le traiter.

PRINCIPALES CARACTÉRISTIQUES D'UN EDI

Débogueur :

- Aide les développeurs à trouver et à **corriger les codes d'erreur** dans une application ou un site web pendant la phase de test.

Outils d'automatisation de la création :

- Utilisés pour **automatiser les processus** de création et de développement de logiciels, tels que la compilation du code source en code machine, le packaging du code binaire et l'exécution de tests automatisés.

Contrôle de version :

- Permet aux développeurs de **suivre les modifications** apportées au code source et connecte l'EDI au **dépôt de sources** qu'ils utilisent.

DIFFÉRENTS TYPES D'EDI

- Un projet peut nécessiter **plusieurs EDI**, il est donc important de prendre en compte le langage de programmation, la facilité d'utilisation, la fiabilité et le type.
- De nombreux types d'EDI offrent **différentes fonctionnalités** pour générer efficacement une application de haute qualité.

Multi langage :

- prend en charge les programmes qui utilisent **plusieurs langages de programmation**, ce qui peut être bénéfique aux débutants pour améliorer leurs compétences.
- Par exemple, Visual Studio offre des fonctionnalités solides et prend en charge une configuration facile pour les extensions et les mises à niveau.

DIFFÉRENTS TYPES D'EDI

Développement mobile :

- Spécialement conçu pour le **développement d'applications mobile,** par exemple AppCode et Android Studio.

Spécifique au langage :

- Conçu pour les développeurs de logiciels travaillant avec **un seul langage,** par exemple Jikes et Jcreator conçus pour Java ou Idle pour Python.

Basé sur le cloud :

- Permet aux développeurs de créer des logiciels avec seulement un navigateur, leur permettant d'accéder au code à distance.

EN RÉSUMÉ…

- Le développement et le test des applications sont des tâches qui demandent **beaucoup de temps.**
- L'utilisation d'un **environnement de développement** simplifie alors le processus.
- Il fournit un **ensemble d'outils** et de **procédures de développement** permettant de créer des logiciels sans affecter l'application originale.

ENVIRONNEMENT DE DÉVELOPPEMENT LOCAL

GÉNÉRALITÉS

- Lorsqu'on débute dans le développement web, installer un **environnement de développement local** est la premières chose à laquelle on est confronté.
- Pour coder avec des langages web et divers frameworks, il faut absolument disposer d'un **serveur web local**.
- Le principal avantage, c'est que **même hors-ligne** on peut continuer à développer son site web ou son application.
- C'est pratique pour développer et tester son application **avant de la déployer** sur un serveur de production et la rendre accessible à tous.

GÉNÉRALITÉS

- Avec peu de connaissance, on peut facilement installer un **environnement web complètement fonctionnel** sur un PC ou un Mac, comme si on était sur un **vrai serveur** hébergé dans un datacenter.
- Il comprend généralement un **server stack** composé de :
  - Apache : serveur Web open source qui gère les requêtes HTTP et sert les pages Web en conséquence.
  - MySQL : serveur de base de données open source proposé par Oracle (on peut aussi lui substituer MariaDB).
  - PHP (abréviation de Hypertext Preprocessor) : langage de programmation orienté Web.
- Selon le système d'exploitation sur lequel le server stack est installé, on parle d'environnement **WAMP** pour windows, **MAMP** pour Mac, **LAMP** pour Linux et **XAMPP** disponible pour tous.

GÉNÉRALITÉS

- Il existe de **nombreuses solutions** sur le marché !
- Depuis mes premiers sites web en 1994, j'ai eu l'occasion d'en utiliser suffisamment pour pouvoir affirmer qu'il n'y a **pas de solution qui fasse l'unanimité.**
- On peut néanmoins donner une liste des solutions les **plus utlisées** et les **plus populaires…**

WAMPSERVER

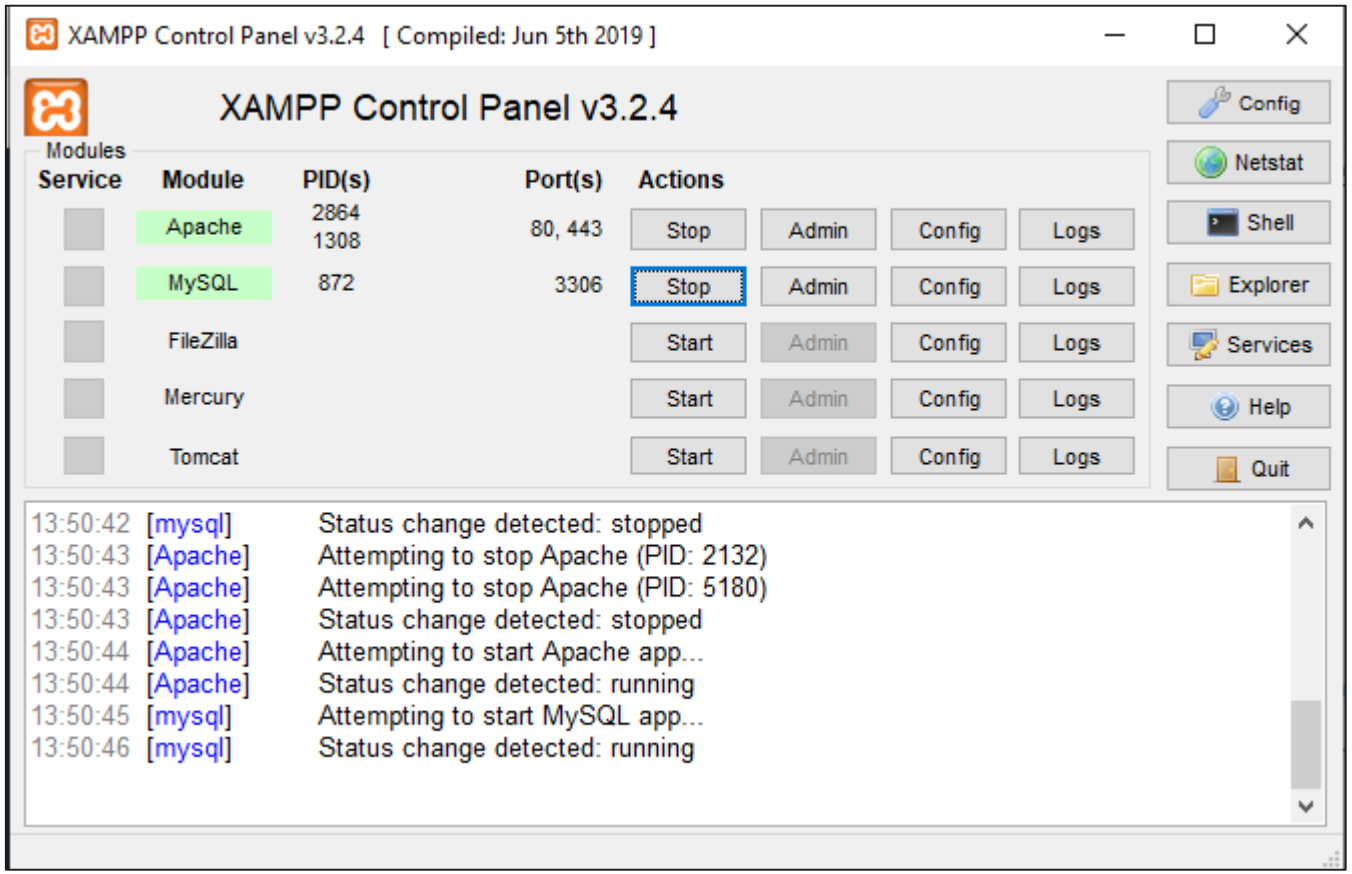
- Le plus utilisé sur Windows !

EASYPHP

- Un serveur local avec une belle popularité.

XAMPP

- Un autre serveur local très complet et multi OS.

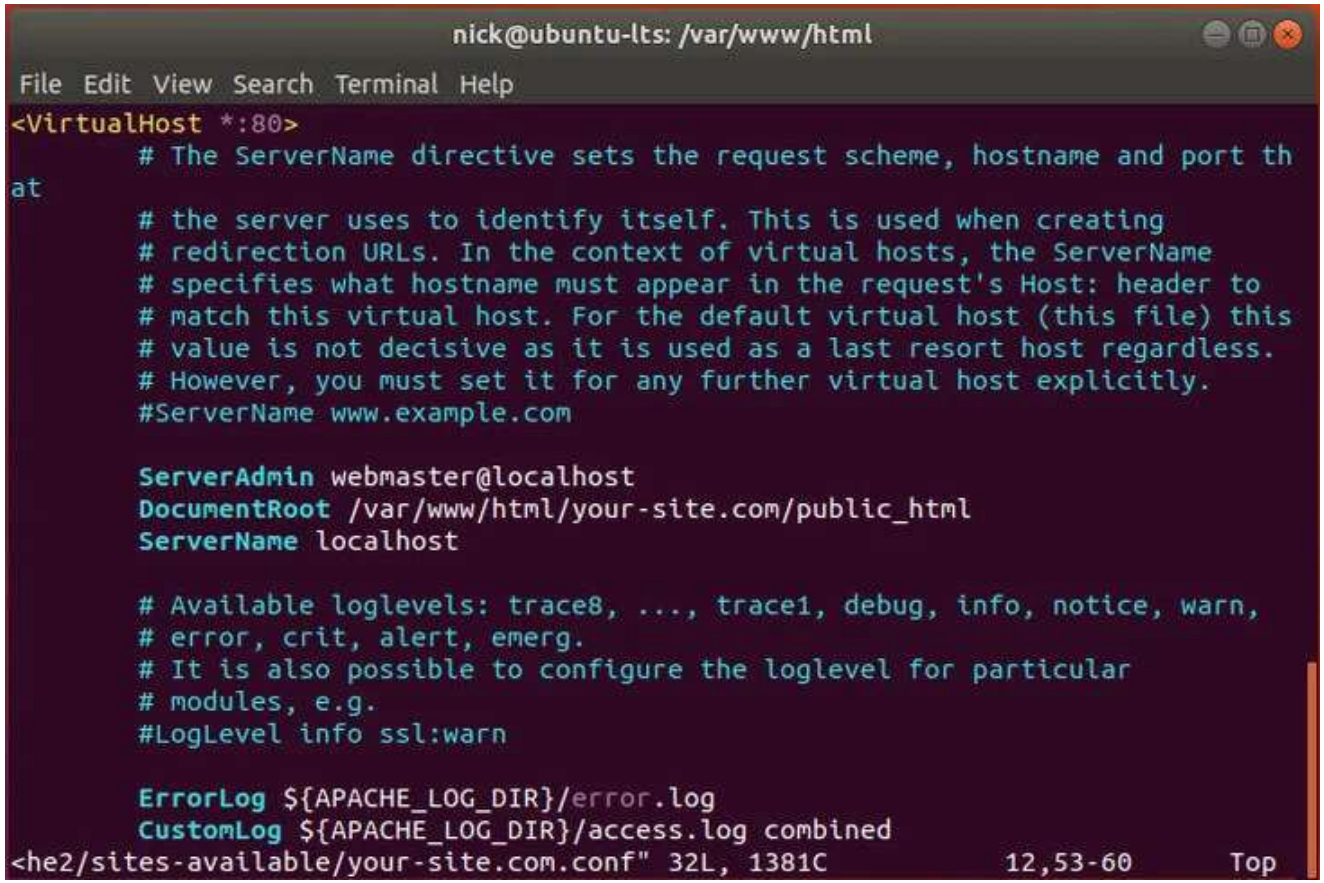


MAMP

- Un serveur web pour les addicts du Mac.

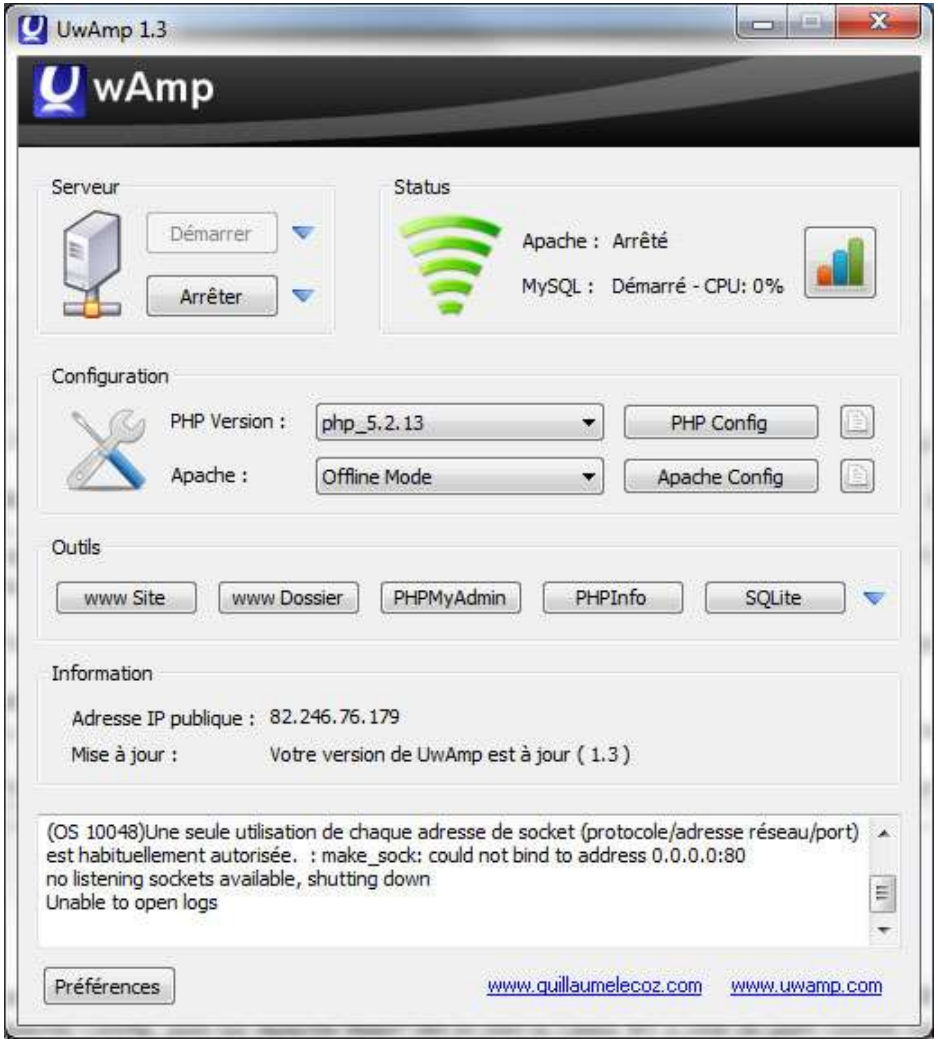
LAMP

- Un serveur web local pour les "barbus" Linux.



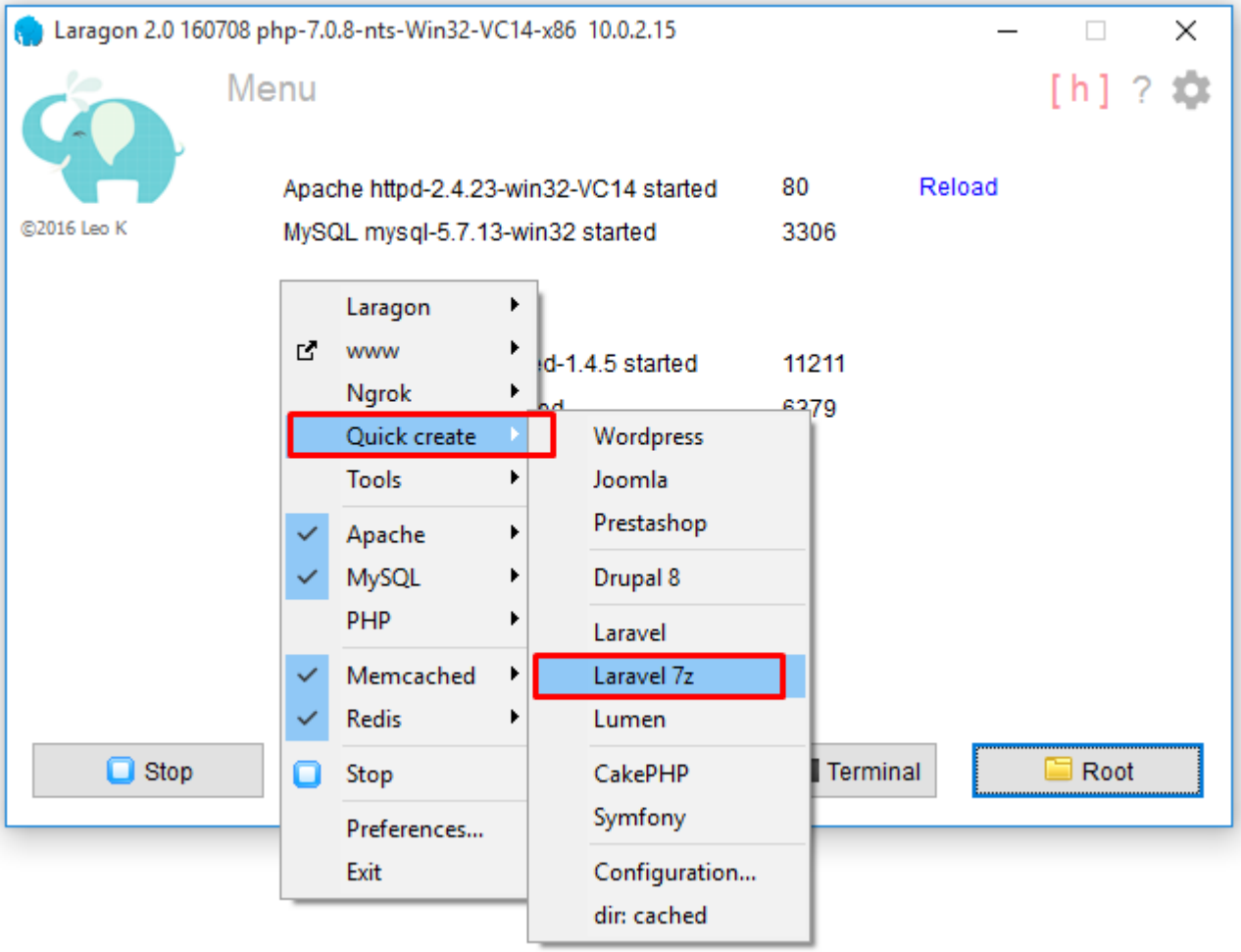
UWAMP

- Un très bon équivalent portable de WampServer.



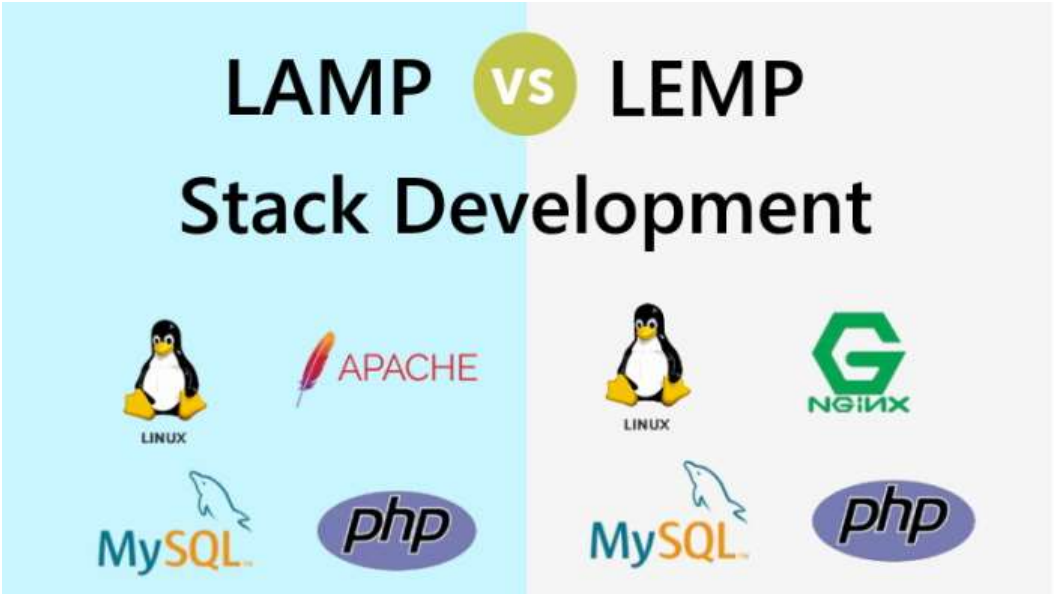
LARAGON

- Performant en version installable ou portable.



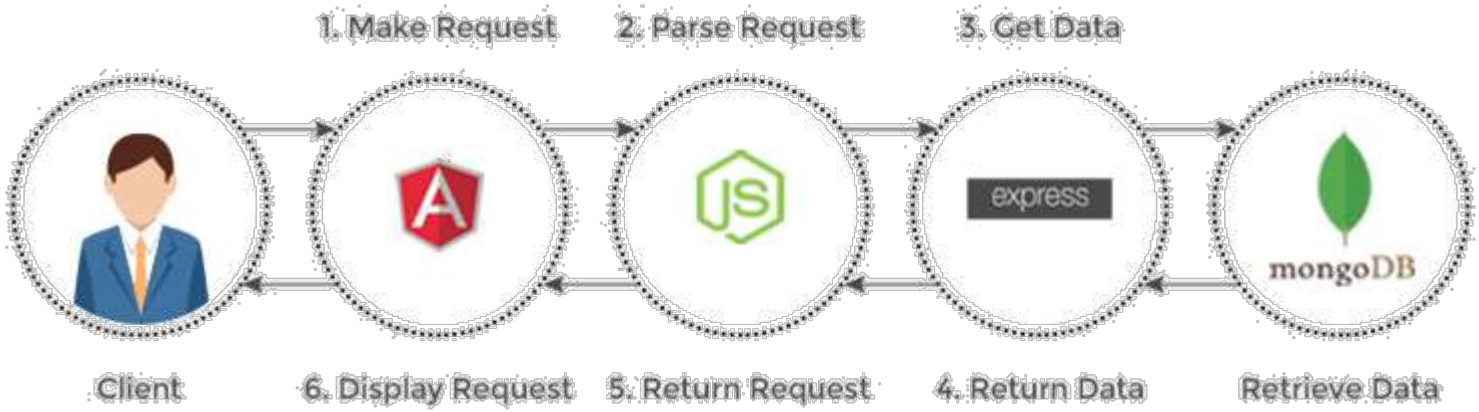
ENVIRONNEMENT LAMP VS LEMP

- Un serveur **LEMP** est différent d'un serveur **LAMP** seulement car il utilise **NginX** en serveur web plutôt **qu'Apache.**
- NginX est un serveur web de plus en plus utilisé notamment pour ses **performances** par rapport à Apache 2.



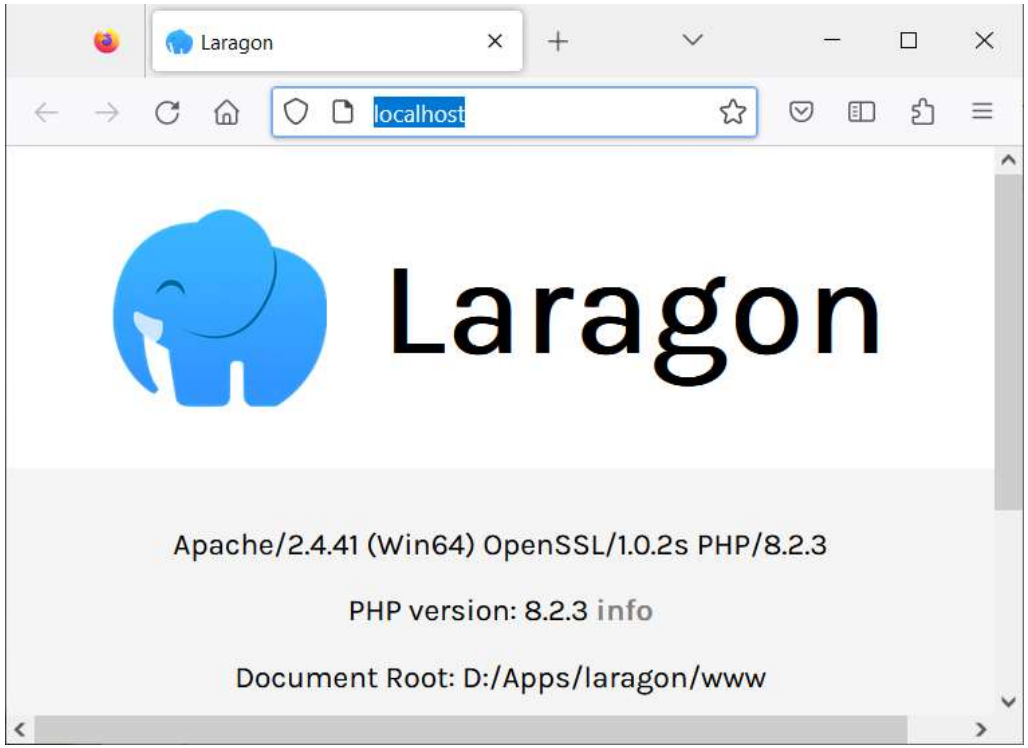
ENVIRONNEMENT MEAN

- Suite de composants **Open Source** fournissant un framework pour la création d'applications web dynamiques et composée de 4 programmes libres :
  - **MongoDB** comme base de données de documents pour l'application back-end.
  - **Express** pour l'application web back-end.
  - **Angular** pour l'application web front-end.
  - **Node.js** pour l'environnement d'exécution.



EN RÉSUMÉ…

- Une fois l'environnement local choisi et installé, on peut enfin accéder à la **racine du serveur** via un navigateur web, en mentionnant :
  - Son nom : **"localhost"**
  - Ou son adresse IP : **"127.0.0.1"**

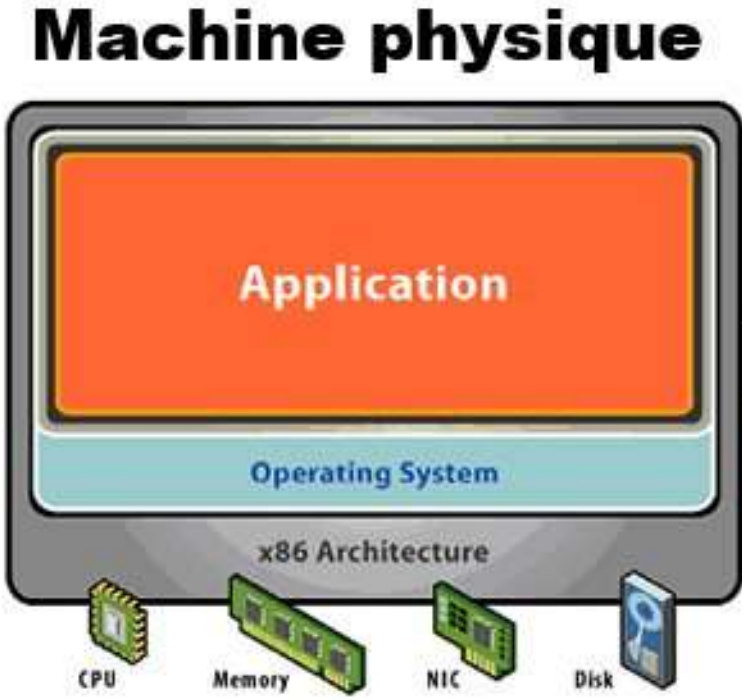


ENVIRONNEMENT DE DÉVELOPPEMENT VIRTUEL

GÉNÉRALITÉS

- Une **Machine Virtuelle**, ou VM pour Virtual Machine, est **"le client"** créé dans un environnement informatique physique, **"l'hôte"**.
- Plusieurs machines virtuelles peuvent coexister sur un seul hôte.
- Les principaux fichiers qui constituent une machine virtuelle sont :
  - un fichier **journal**,
  - un fichier de **paramètres** de RAM non volatile,
  - un fichier de **disque virtuel**
  - et un fichier de **configuration**.

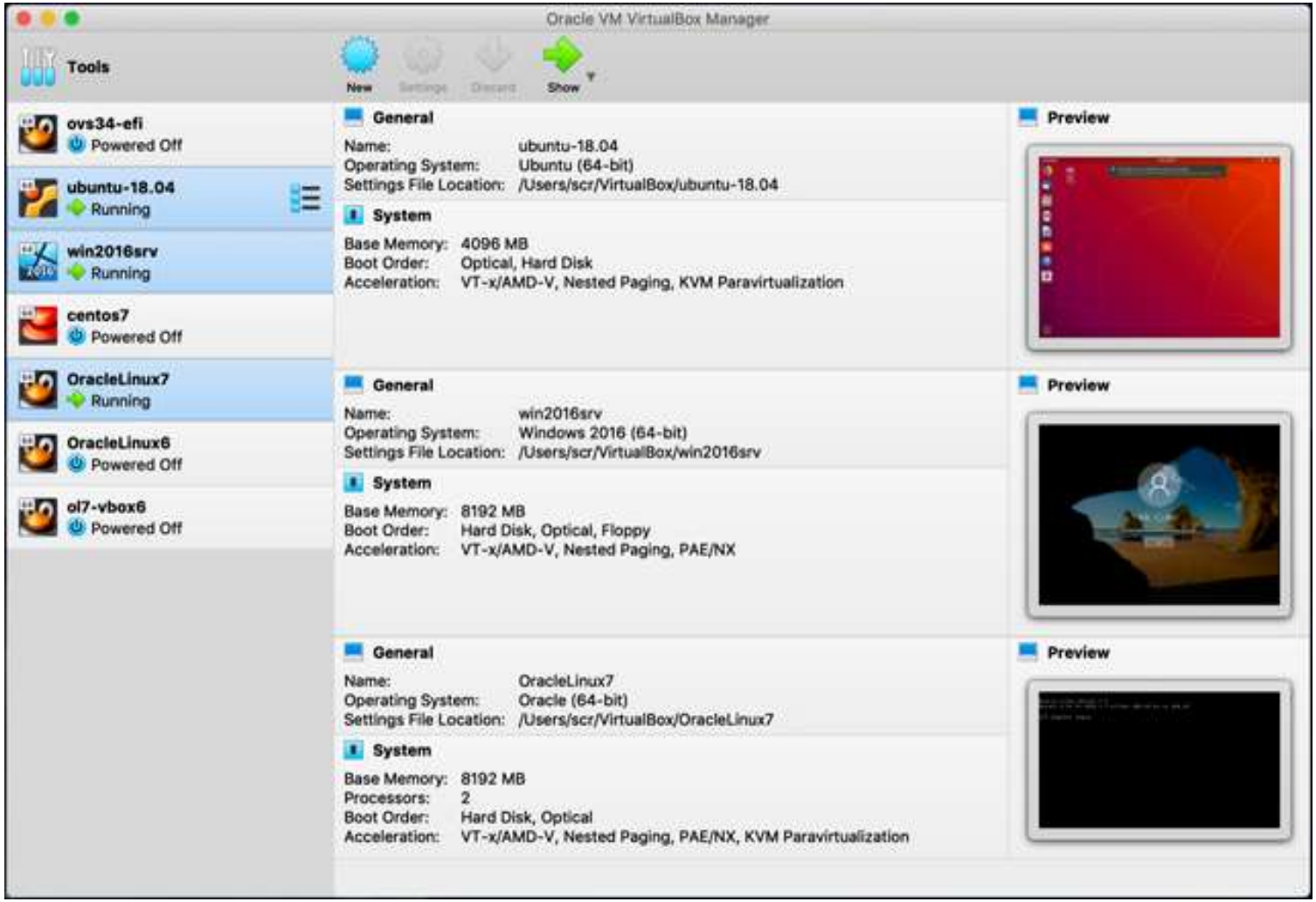
GÉNÉRALITÉS



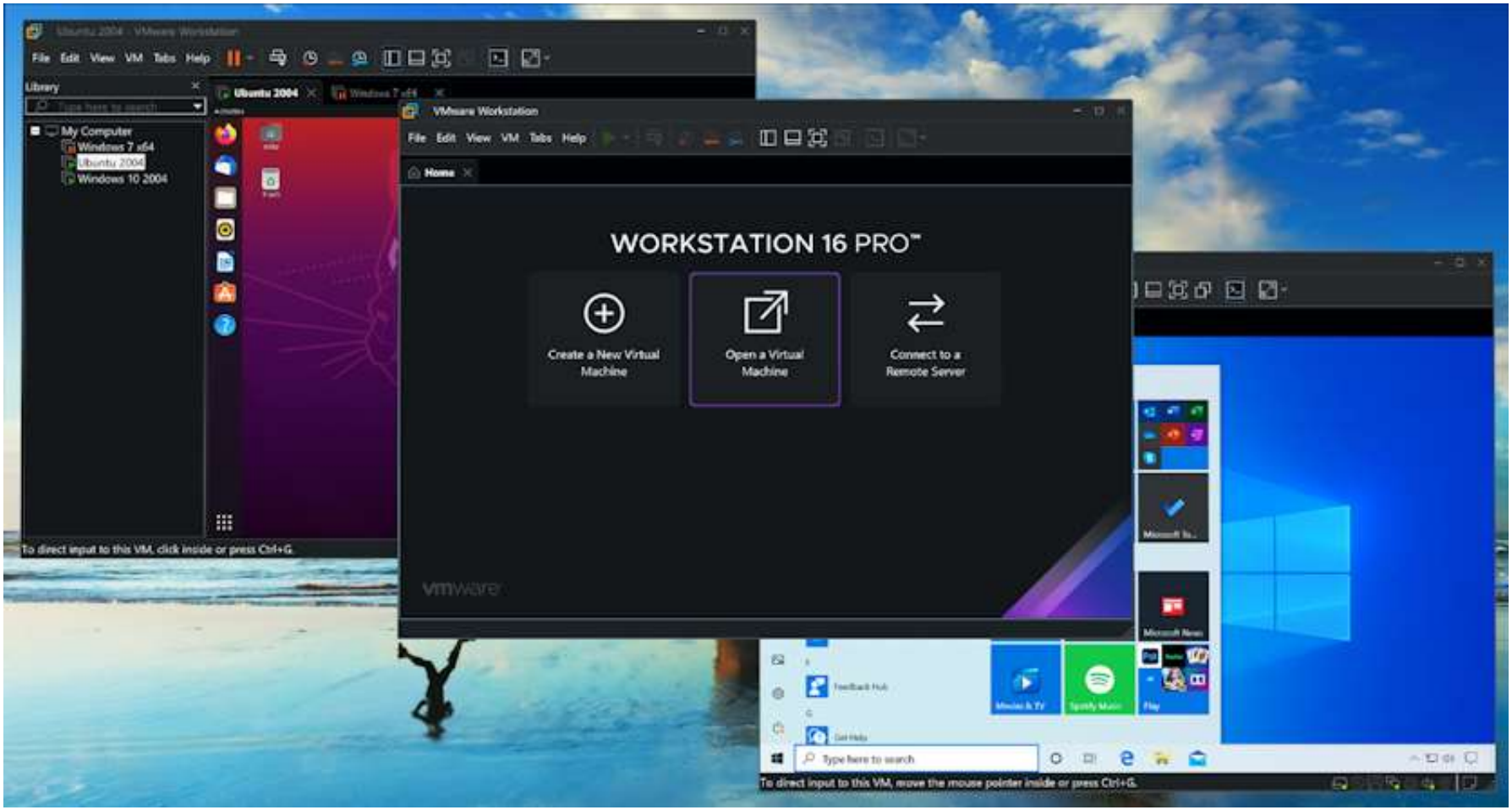
GÉNÉRALITÉS

- Un logiciel de virtualisation est un outil permettant de **simuler des systèmes informatiques en ligne.**
- Chaque logiciel présente ses **avantages et inconvénients** qui peuvent orienter dans le choix de celui qui conviendra le mieux à notre projet.
- Parmi les logiciels de virtualisation disponibles sur le marché, on peut noter les **plus utilisés** et les **plus populaires** d'entre eux.

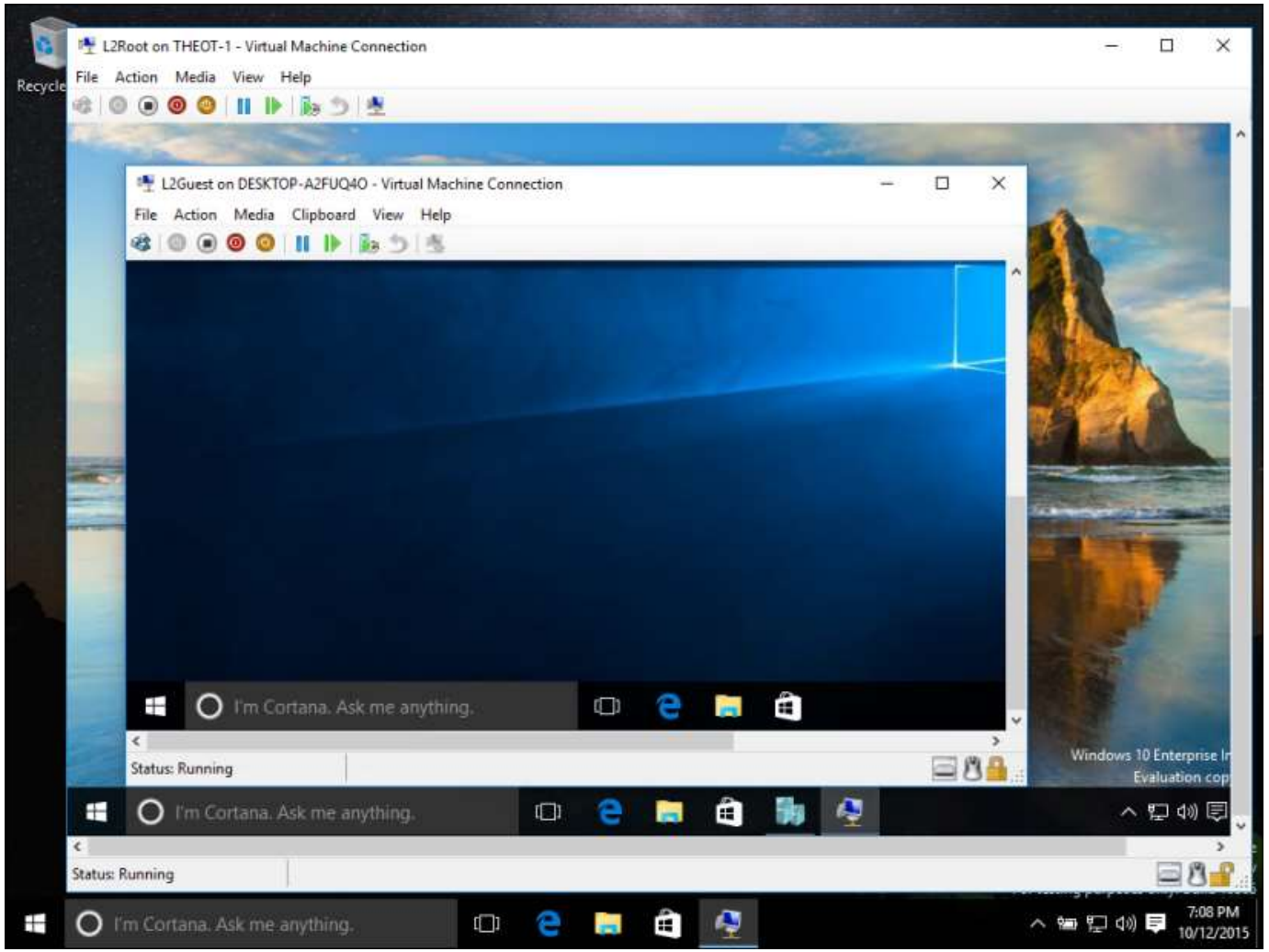
ORACLE VIRTUALBOX



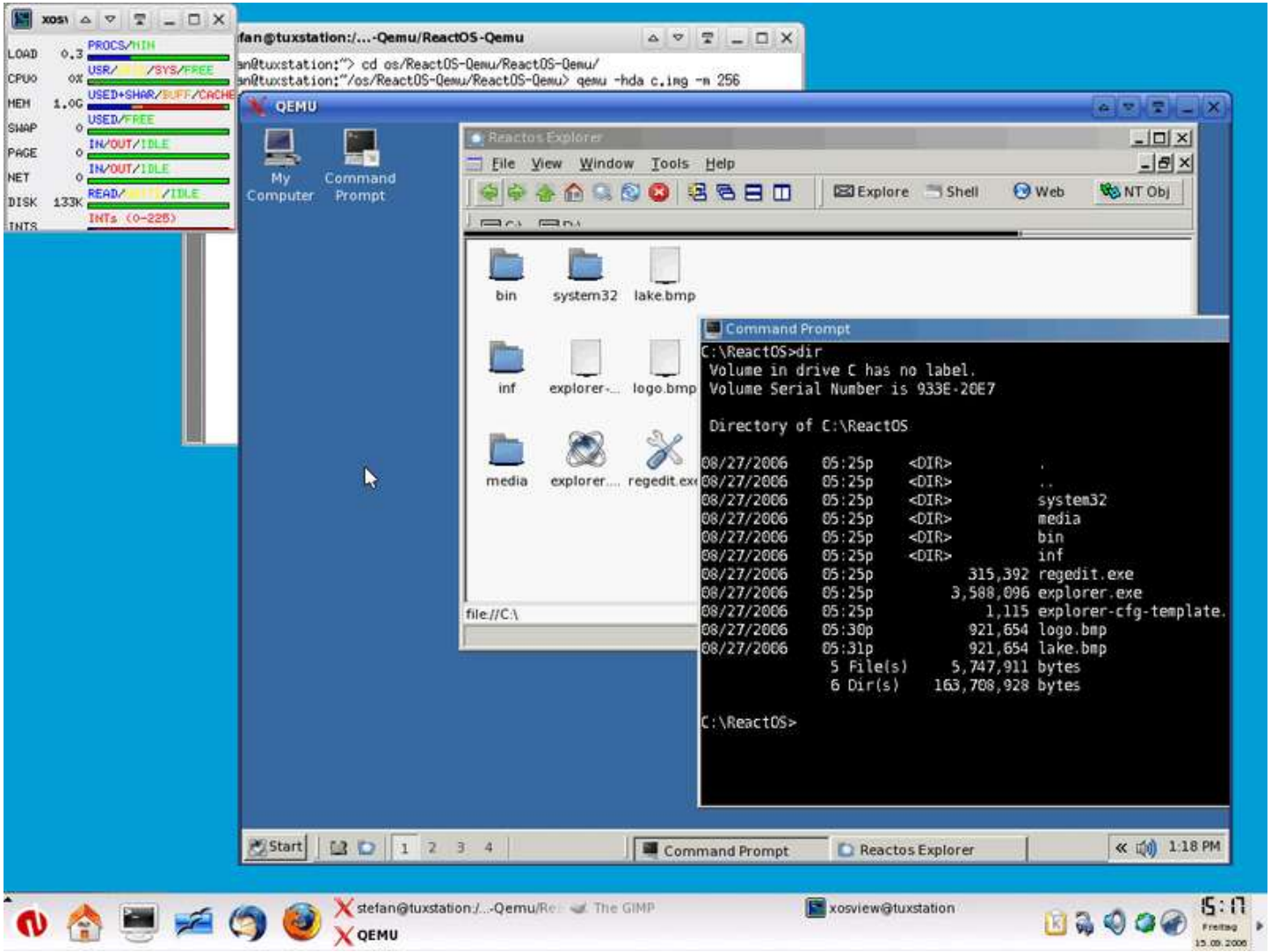
VMWARE WORKSTATION



MICROSOFT HYPER-V



QEMU


QEMU
My Computer
Command Prompt
ReactOS Explorer
File View Window Tools Help
Explore Shell Web NT Obj
bin
system32
lake.bmp
inf
explorer-...
logo.bmp
media
explorer....
regedit.ex
file://C:\
Command Prompt
C:\ReactOS>dir
Volume in drive C has no label.
Volume Serial Number is 933E-20E7
Directory of C:\ReactOS
08/27/2006 05:25p <DIR> .
08/27/2006 05:25p <DIR> ..
08/27/2006 05:25p <DIR> system32
08/27/2006 05:25p <DIR> media
08/27/2006 05:25p <DIR> bin
08/27/2006 05:25p <DIR> inf
08/27/2006 05:25p 315,392 regedit.exe
08/27/2006 05:25p 3,588,096 explorer.exe
08/27/2006 05:25p 1,115 explorer-cfg-template.
08/27/2006 05:30p 921,654 logo.bmp
08/27/2006 05:31p 921,654 lake.bmp
5 File(s) 5,747,911 bytes
6 Dir(s) 163,708,928 bytes
C:\ReactOS>
Start
Command Prompt
Reactos Explorer
1:18 PM
stefan@tuxstation:/...-Qemu/ReactOS-Qemu
QEMU
xosview@tuxstation

KVM (KERNEL-BASED VIRTUAL MACHINE)

POURQUOI CRÉER UNE MACHINE VIRTUELLE ?

- Installer une VM permet d'avoir accès aux **mêmes fonctionnalités** que des ordinateurs physiques, mais via des logiciels.
- Comme les ordinateurs physiques, elles exécutent des **applications** et un **système d'exploitation**.
- Toutefois, elles ne sont que des fichiers informatiques qui sont exécutés sur un ordinateur physique et se comportent comme un ordinateur physique.
- Autrement dit, les machines virtuelles fonctionnent comme des **systèmes informatiques distincts**.

POURQUOI UTILISER UNE MACHINE VIRTUELLE ?

- Créer une machine virtuelle est utile pour **exécuter des tâches spécifiques,** trop risquées pour être exécutées dans l'environnement de production.
- Par exemple accéder à des données infectées par des virus et tester des systèmes d'exploitation !
- Comme la machine virtuelle est **isolée du reste du système,** les logiciels qui y résident ne peuvent pas intervenir sur l'ordinateur hôte.
- Les machines virtuelles peuvent également avoir d'autres usages, tels que la **virtualisation des serveurs.**

AVANTAGES DES MACHINES VIRTUELLES

- Options de **reprise d'activité** (suite à une catastrophe naturelle ou une cyber-attaque) et de provisionnement des applications.
- Les machines virtuelles sont **simples à gérer et à entretenir** et sont disponibles partout.
- **Plusieurs environnements** de systèmes d'exploitation peuvent être exécutés sur un même ordinateur physique.

INCONVÉNIENTS DES MACHINES VIRTUELLES

- L'exécution de plusieurs machines virtuelles sur une seule machine physique peut être à l'origine de **performances instables.**
- Les machines virtuelles sont **moins efficaces** et **plus lentes** qu'un ordinateur physique.

EN RÉSUMÉ…

- Au sens large, la **virtualisation** consiste à simuler l'existence de plusieurs machines informatiques en n'utilisant qu'une seule machine hôte.
- Elle permet de **diminuer les coûts** d'achat de matériel informatique et de **rentabiliser leur utilisation**.
- Elle permet également de **gagner du temps** : une simple modification de la configuration remplace une longue procédure d'acquisition de matériel informatique.
- Enfin elle offre au développeur la possibilité de tester ses applications sur **différents systèmes d'exploitation** et **différentes versions**.