

INTEROPÉRABILITÉ ET INNOVATION DANS LES SI

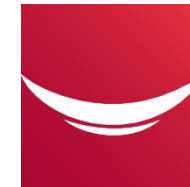
La partie cool du module!



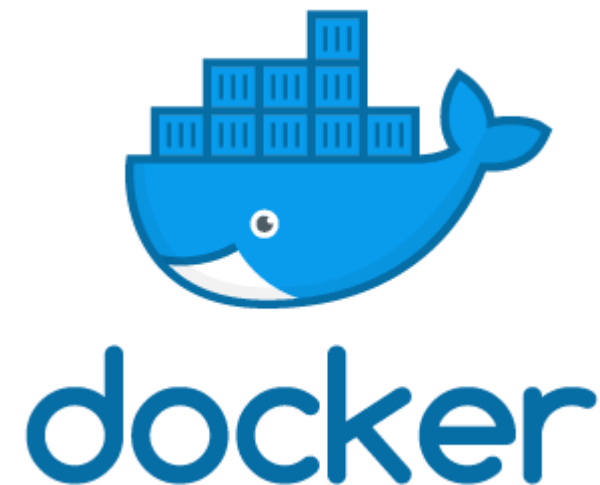
ME, MYSELF & I : NICOLAS FLEURY



- Architecte Senior .NET & Responsable Industrialisation
- MIAGE Promo 2007
- 12 ans de .NET
 - Dev
 - Lead Dev
 - CP
 - Archi
- 6 ans chez Sopra Steria
 - Membre de la cellule AET



AGENDA

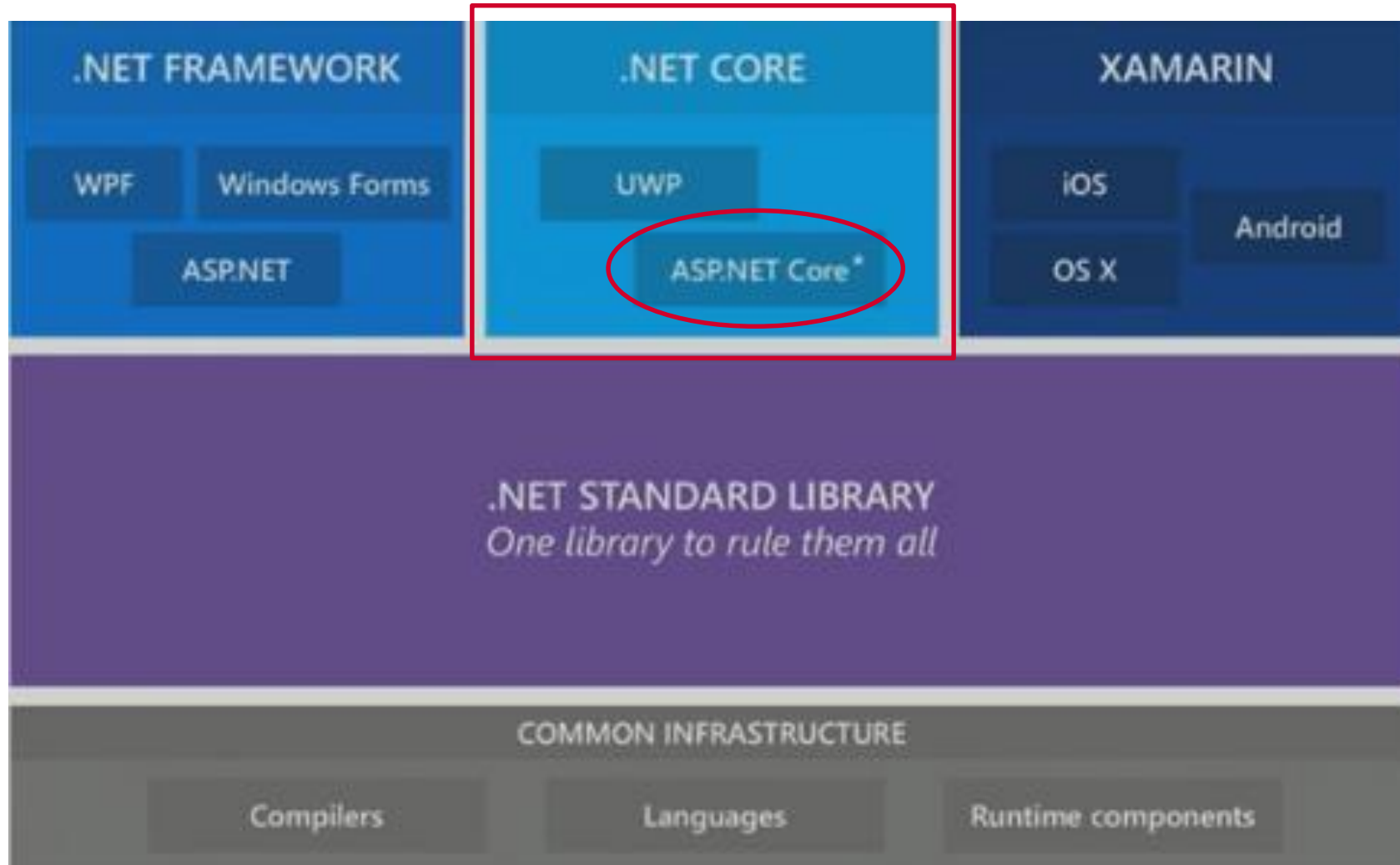


QU'EST-CE QUE LE .NET

- Historiquement
 - C'est la réponse de Microsoft à Java et à J2E
 - 1ère version en 2001
 - Multi langage, mono plateforme
 - A évolué jusqu'à aujourd'hui
 - Framework fermé et propriétaire
- **Mais ça, c'était avant...**

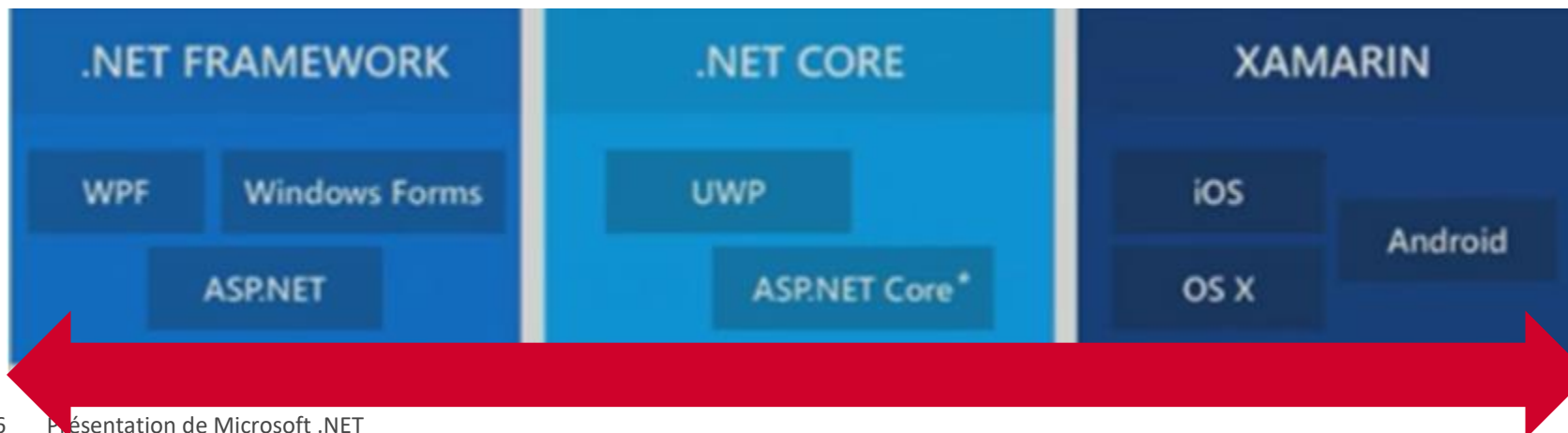


PLUSIEURS IMPLÉMENTATIONS



UN PETIT DÉTOUR

- .NET Standard
 - Spécification officielle des API .NET
 - APIs destinées à être disponibles sur toutes les implémentations de .NET
 - Objectif : Etablir une meilleure uniformité dans l'écosystème .NET
 - Framework socle pour les implémentations communes:
 - Si votre code cible une version de .NET Standard, il peut s'exécuter sur n'importe quelle implémentation de .NET qui prend en charge cette version de .NET Standard.



.NET CORE : C'EST QUOI

- **Implémentation Open Source du .NET**

- Licence MIT & Apache 2
- Projet de la .NET Foundation maintenu par Microsoft et la communauté.NET
- Github : <https://github.com/dotnet/core>



- **Multiplateforme**



- **Cohérence d'architectures**

- Même comportement d'exécution sur les architecture x86, x64 & ARM

- **Souplesse de déploiement**

- Peut être inclus dans votre application ou installé côte à côte à l'échelle d'un utilisateur ou de l'ordinateur. Peut être utilisé avec des conteneurs Docker.

- **Langages supportés**



BEGINNER'S GUIDE

- Comment l'installer ?
 - <https://dotnet.microsoft.com/download/archives>

v2.2.3

Current ⓘ Security patch ⓘ

Released 2019-03-12

[Release notes](#)

Supports C# 7.3

Supports F# 4.5

Supports Visual Basic 15.5

ASP.NET Core IIS Module
12.2.19024.2

SDK 2.2.105

Windows

- .NET Core Installer: [x64](#) | [x86](#)
- .NET Core Binaries: [x64](#) | [x86](#) | [ARM32](#)

macOS

- .NET Core Installer: [x64](#)
- .NET Core Binaries: [x64](#)

Linux

- Package Manager Instructions: [x64](#)
- .NET Core Binaries: [x64](#) | [ARM32](#) | [ARM64](#) | [x64 Alpine](#) | [RHEL 6 x64](#)

Other

- Checksums: [Checksums](#)

.NET Core

.NET Core is a cross-platform version of .NET for building websites, services, and console apps.

- [.NET Core 3.0](#) Preview ⓘ
- [.NET Core 2.2](#) Current ⓘ
- [.NET Core 2.1](#) LTS ⓘ
- [.NET Core 2.0](#) End of life ⓘ
- [.NET Core 1.1](#) LTS ⓘ
- [.NET Core 1.0](#) LTS ⓘ

Runtime 2.2.3

Windows

- ASP.NET Core/.NET Core: [Runtime & Hosting Bundle](#)
- ASP.NET Core Installer: [x64](#) | [x86](#)
- ASP.NET Core Binaries: [x64](#) | [x86](#) | [ARM32](#)
- .NET Core Installer: [x64](#) | [x86](#)
- .NET Core Binaries: [x64](#) | [x86](#) | [ARM32](#)

macOS

- ASP.NET Core Binaries: [x64](#)
- .NET Core Installer: [x64](#)
- .NET Core Binaries: [x64](#)

Linux

- Package Manager Instructions: [x64](#)
- ASP.NET Core Binaries: [x64](#) | [ARM32](#) | [x64 Alpine](#)
- .NET Core Binaries: [x64](#) | [ARM32](#) | [ARM64](#) | [x64 Alpine](#) | [RHEL 6 x64](#)

Other

- Checksums: [Checksums](#)



BEGINNER'S GUIDE

- One command line to rule them all : dotnet

```
sdk-options:
-d|--diagnostics  Activez la sortie des diagnostics.
-h|--help         Affichez l'aide de la ligne de commande.
--info           Affichez les informations sur .NET Core.
--list-runtimes   Affichez les runtimes installés.
--list-sdks       Affichez les SDK installés.
--version        Affichez la version utilisée du kit SDK .NET Core.

Commandes du SDK:
add              Ajoutez un package ou une référence à un projet .NET.
build            Générez un projet .NET.
build-server     Interagissez avec les serveurs démarrés par une build.
clean            Nettoyez les sorties de build d'un projet .NET.
help            Affichez l'aide de la ligne de commande.
list            Listez les références de projet d'un projet .NET.
migrate          Effectuez la migration d'un projet project.json vers un projet MSBuild.
msbuild          Exécutez des commandes MSBuild (Microsoft Build Engine).
new             Créez un fichier ou projet .NET.
nuget            Fournit des commandes NuGet supplémentaires.
pack            Créez un package NuGet.
publish          Publiez un projet .NET à des fins de déploiement.
remove           Supprimez un package ou une référence d'un projet .NET.
restore          Restaurez les dépendances spécifiées dans un projet .NET.
run             Générez et exécutez une sortie de projet .NET.
sln             Modifiez les fichiers solution Visual Studio.
store            Stockez les assemblys spécifiés dans le magasin packages de runtime.
test            Exécutez des tests unitaires à l'aide du programme Test Runner spécifié dans un projet .NET.
tool            Installez ou gérez les outils qui étendent l'expérience .NET.
vstest          Exécutez des commandes VSTest (Microsoft Test Engine).

Commandes supplémentaires d'outils groupés :
dev-certs       Créez et gérez des certificats de développement.
ef              Outils en ligne de commande Entity Framework Core.
sql-cache       Outils en ligne de commande du cache SQL Server.
user-secrets    Gérez les secrets d'utilisateur de développement.
watch           Démarrez un observateur de fichier qui exécute une commande quand les fichiers changent.
```



QUEL IDE UTILISER ?

- **Le plus complet**



- Version actuelle 2017
- Sortie de la version 2019 le 02/19

- **Le plus modulaire**






- Nécessite quelques extensions
 - C#
 - ...

DÉMO : HELLO WORLD



ASP.NET CORE

« ASP.NET Core est un framework multiplateforme à hautes performances et [open source](#) pour créer des applications cloud modernes et connectées à Internet. »

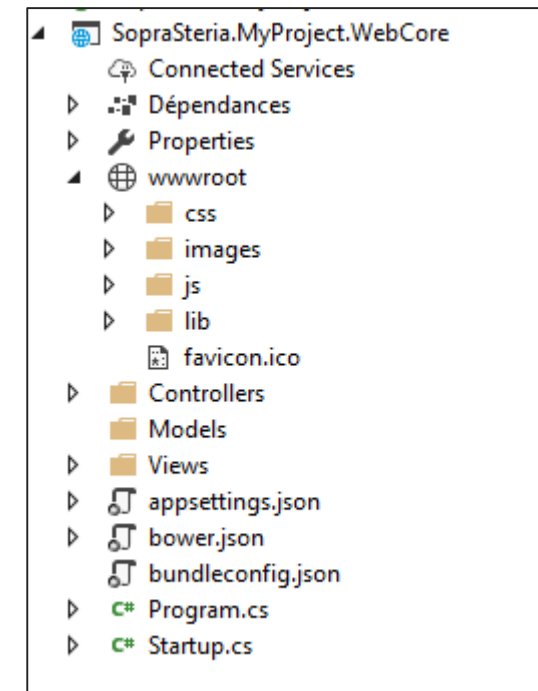
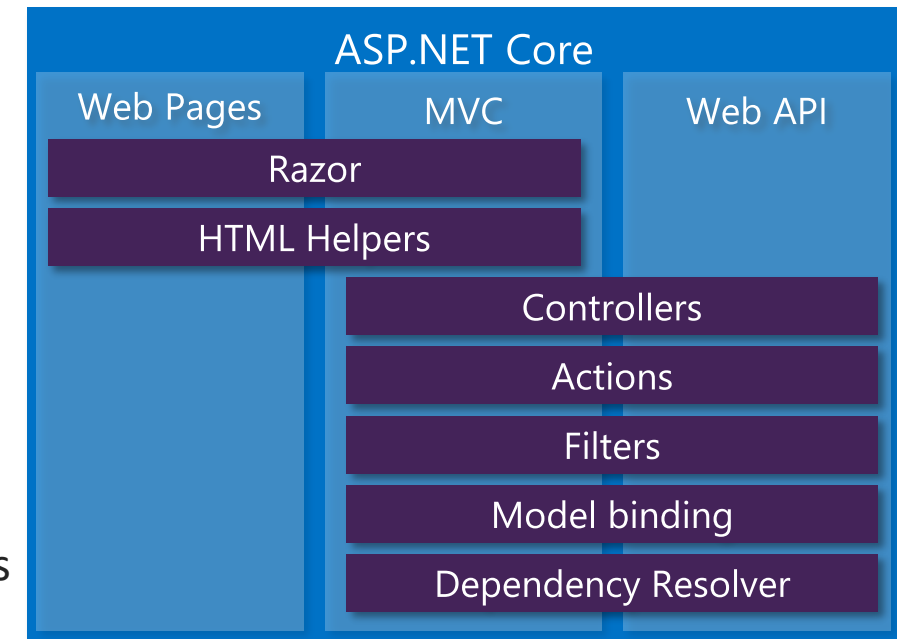
- Permet des applications Web & des API REST
- Multiplateforme :   
- Open source : <https://github.com/aspnet/AspNetCore>
- Capable être héberger par IIS, Apache, Nginx, Docker, etc...
- Capable de s'auto-héberger



ASP.NET CORE

AUTRE INFOS EN VRAC

- Les partie Web et API d'ASP.NET Core partagent le même modèle de développement
- C'est une application console !!
- L'arborescence d'un projet sépare les ressources statiques des ressources de programmation
- La configuration applicative passe par un fichier json
- L'injection de dépendance est intégré
- Les différentes fonctionnalités que l'on ajouter sont géré par des « middleware »
- Nuget toujours utilisé pour la gestion des packages
- Compatible avec tous les Frameworks modernes du moment : Angular, React, etc...



ASP.NET CORE MVC

```
public class PlayerController : Controller
{
    public IActionResult Add()
    {
        PlayerModel model = new PlayerModel();
        return View(model);
    }

    [HttpPost]
    public IActionResult Add(PlayerModel model)
    {
        if (!ModelState.IsValid)
            return View(model);
        players.Add(model);

        return RedirectToAction("Index");
    }
}
```

```
public class PlayerModel
{
    public int Id { get; set; }

    [Required]
    [MaxLength(20)]
    public string Name { get; set; }

    [Required]
    [EmailAddress]
    public string Email { get; set; }
}
```

```
<form asp-action="Edit">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <input type="hidden" asp-for="Id" />
    <div class="form-group">
        <label asp-for="Name" class="control-label"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Email" class="control-label"></label>
        <input asp-for="Email" class="form-control" />
        <span asp-validation-for="Email" class="text-danger"></span>
    </div>
    <div class="form-group">
        <input type="submit" value="Save" class="btn btn-primary" />
    </div>
</form>
```



ASP.NET CORE MVC

QUELQUES COMMANDES PRATIQUES

- `dotnet tool install --global dotnet-aspnet-codegenerator --version 2.2.3`
- `dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design`
- `dotnet aspnet-codegenerator controller [options]`
- `dotnet build`
- `dotnet run`



DÉMO : WEBAPP



ASP.NET CORE WEBAPI

- Même modèle que le MVC mais sans le V
- Spécification de la route pour éviter les collisions avec la partie WebApp (Convention)
- Spécification des verbes HTTP en entête
 - Précision des paramètres de query en paramètre
- Pour générer un controller:
 - `dotnet aspnet-codegenerator controller -name <controller> -m <model> -dc <dataContext> --relativeFolderPath Controllers -api`

```
[Route("api/[controller]")]
[ApiController]
public class PlayerController : ControllerBase
{
    // GET: api/Player
    [HttpGet]
    public async Task<ActionResult<IEnumerable<Player>>> GetPlayer()
    {
        ...
    }

    // GET: api/Player/5
    [HttpGet("{id}")]
    public async Task<ActionResult<Player>> GetPlayer(int id)
    {
        ...
    }

    // PUT: api/Player/5
    [HttpPut("{id}")]
    public IActionResult PutPlayer(int id, Player player)
    {
        ...
    }

    // POST: api/Player
    [HttpPost]
    public ActionResult<Player> PostPlayer(Player player)
    {
        ...
    }

    // DELETE: api/Player/5
    [HttpDelete("{id}")]
    public ActionResult<Player> DeletePlayer(int id)
    {
        ...
    }
}
```

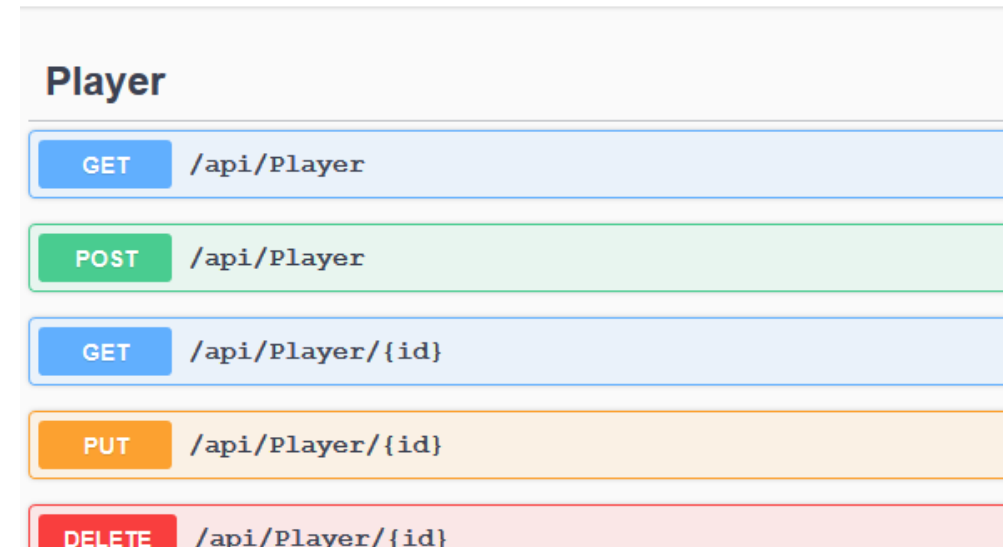
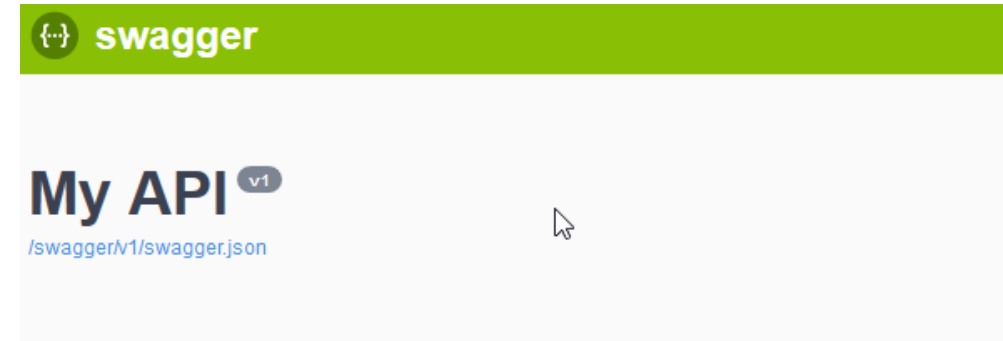
ON N'AVAIT PAS DIT QU'ON PARLERAIS DE SWAGGER?

- Package nuget à installer
 - `dotnet add package Swashbuckle.AspNetCore`
- 2 portions de code à ajouter dans Startup.cs
 - Dans ConfigureServices

```
services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new Info { Title = "My API", Version = "v1"
});
```

- Dans Configure

```
app.UseSwagger();
app.UseSwaggerUI(c =>
{
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");
});
```



DÉMO : WEBAPI



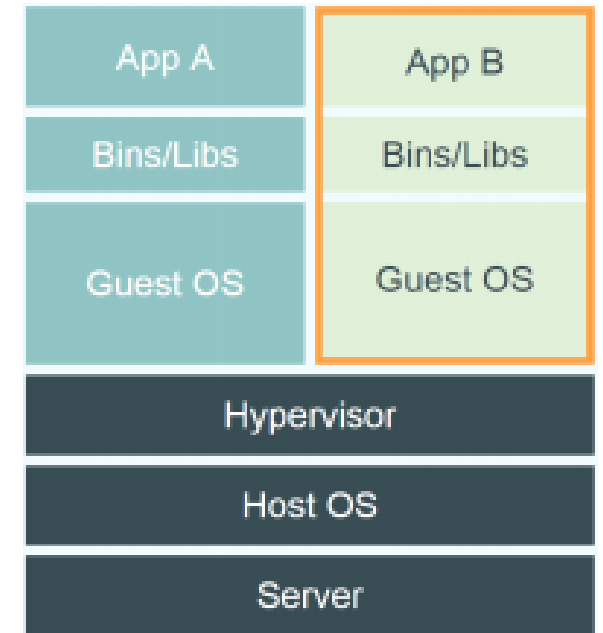
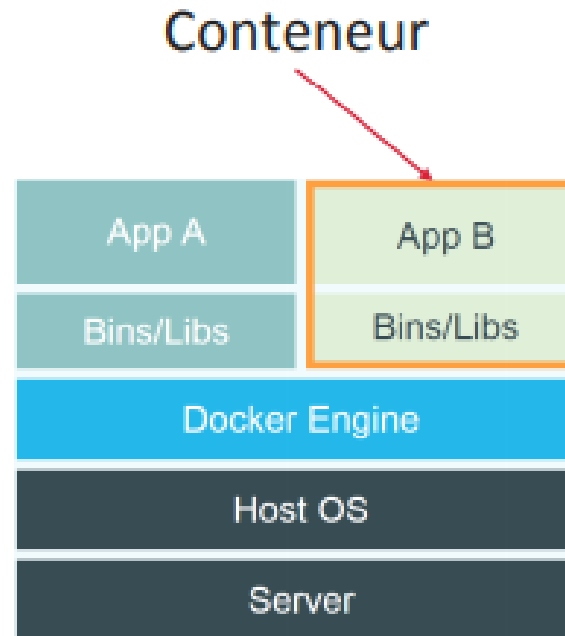
COMMENT HÉBERGER MON APPLICATION

- A l'ancienne
 - Un machine physique avec son OS (Windows ou Linux)
 - Installation des Framework, server webs & co
 - Plusieurs machines physiques avec un LoadBalancer pour absorber la charge
 - Il faut déployer l'application sur chaque machine individuellement
- Avec de la virtualisation
 - Un hyperviseur sur uen machine physique pouvant héberger plusieurs VM
 - Une VM requiert de la RAM et du stockage dédié
 - Nécessite d'installer un OS Guest par VM nécessitant entre 1 et 2 Go de RAM
 - Les applications sont toujours dépendantes de l'OS Guest de la VM
 - Difficile de maintenir et distribuer les images virtualisées



C'EST QUOI DOCKER ?

- Système de virtualisation basée sur des containers utilisant le noyau de l'OS hôte pour s'exécuter
- Un container est complètement isolé de son environnement. Et dispose de :
 - Son système de fichier propre
 - Ses processus
 - Sa mémoire
 - Ses périphériques
 - Son IP et ses ports réseaux
- Un container embarque l'application ET ses dépendances (binaires et librairies)

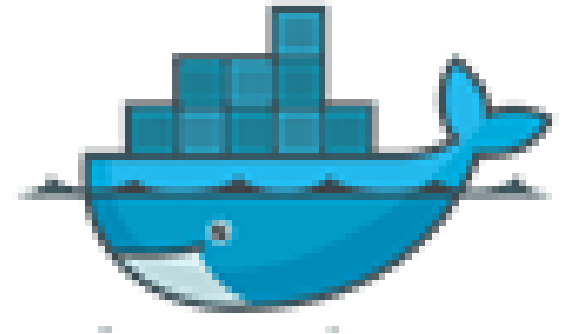


VM



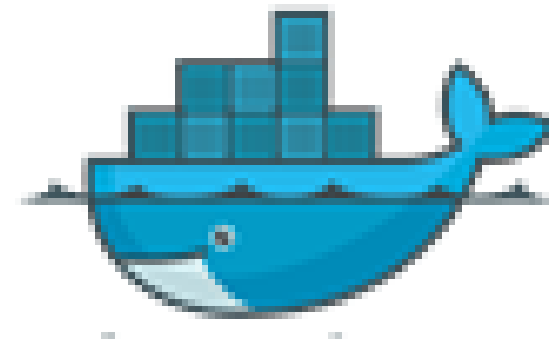
LES AVANTAGES DE DOCKER

- Isolation d'une application au travers d'un container
- Indépendance vis-à-vis de l'OS hôte
 - Le container est donc transportable sur un autre OS sans changement
 - Facilite le passage de la dev à la production
- Démarrage plus rapide qu'une VM car un container n'embarque pas l'OS
 - Scalabilité horizontale plus rapide (tt dépend de l'appli à démarrer)
- Mutualisation accrues des ressources systèmes
- Distribution des images Docker grâce à des repository publics ou privés
- Facilite l'exploitation en production car les applications sont standardisées
- Séparation des responsabilités
 - Les développeurs créent les applications dockerisées
 - Les exploitants les déploient et les gèrent



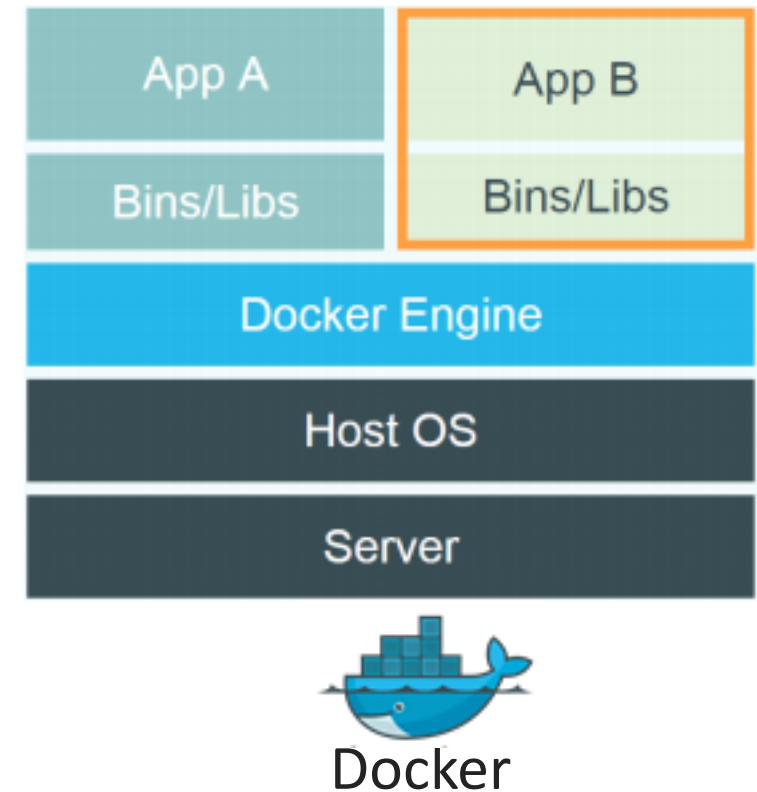
LES AVANTAGES DE DOCKER

- Permet de répondre aux 5 enjeux pour une application cloud-ready
 - Scalabilité horizontale
 - Qualité de service
 - Haute disponibilité
 - Garantie d'intégrité et de sécurité
 - Déploiement facile et fréquent



DOCKER ENGINE

- Appelé communément le daemon Docker
- Chaque machine voulant exécuter des containers doit installer Docker Engine
- C'est lui qui permet d'isoler les containers grâce aux namespaces du noyau Linux et aux Controls Groups (cgroups).
- Docker Engine a aussi comme fonctions de :
 - Fabriquer les images Docker
 - Livrer les images sur un repository
 - Exécuter les containers et gérer leur cycle de vie
 - Faire office d'interface unique et normalisée quelque soit l'OS hôte
 - Être la passerelle entre les commandes du container et l'OS hôte
- Docker Engine s'exécute nativement sur les systèmes Linux
 - Mais aussi installable sur Windows et MacOS
- Existe en version gratuite (Docker CE) et Entreprise (Docker EE)



LES OUTILS DOCKER



- Docker fournit les outils suivants :
 - Docker Engine
 - Docker Client (CLI)
 - Docker Compose
 - Permet l'installation d'application multi-containers
 - Docker Swarm & Kubernetes
 - Permettent de gérer des clusters de machines Docker
 - Docker Hub
 - Docker Hub est le store (ou registry) d'images Docker accessible depuis Internet. Permet de partager des images officielles à la communauté.
 - Docker permet aussi de créer des registry privés. Nécessaire dans un environnement Entreprise
 - Docker Machine
 - Permet d'installer et de gérer Docker sur des hôtes virtualisés. Donc d'installer Docker sur d'autres OS que Linux (Windows et MacOS)



LES CONCEPTS INDISPENSABLES



- Image
 - Résultat compilé immuable qui va permettre l'exécution de containers
 - Une image est construite à partir d'images parentes (empilement ou héritage d'images).
 - Une image est stockée sur un registry (public ou privée). Avec un numéro de version
- Dockerfile
 - Contient le code source pour construire une image
- Container
 - C'est une instance d'une image qui s'exécute
 - Un container étant isolé de son environnement, la destruction d'un container supprime toutes les données qu'il a créé. D'où la notion de volume pour persister les données
- Volume
 - Permet de persister les données créées par un container même après sa destruction ou arrêt.
 - Ex de fichiers persistés : contenu de la bdd, log, fichier de conf



ET ASP.NET CORE DANS TOUT ÇA?

- Il suffit juste d'un fichier Dockerfile et d'une ligne de commande pour créer son conteneur Docker
- Le DockerFile peut être généré
 - Nativement dans VS2017
 - En installant l'extension Docker dans VS Code
- La ligne de commande à exécuter
 - `docker build --rm -f "Dockerfile" -t <app>:<version>`
 - Votre IDE peut le faire pour vous ;)

```
FROM microsoft/dotnet:2.2-aspnetcore-runtime AS base
WORKDIR /app
EXPOSE 80

FROM microsoft/dotnet:2.2-sdk AS build
WORKDIR /src
COPY ["SopraSteria.Exemple.Web.csproj", "."]
RUN dotnet restore "./SopraSteria.Exemple.Web.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "SopraSteria.Exemple.Web.csproj" -c Release -o /app

FROM build AS publish
RUN dotnet publish "SopraSteria.Exemple.Web.csproj" -c Release -o /app

FROM base AS final
WORKDIR /app
COPY --from=publish /app .
ENTRYPOINT ["dotnet", "SopraSteria.Exemple.Web.dll"]
```



DÉMO : DOCKER



C'EST QUOI LE CLOUD COMPUTING?

- A ne pas confondre avec la nomination Cloud utilisé pour le stockage et le partage de vos fichiers
 - DropBox, OneDrive, GoogleDrive, etc...
- Mise à disposition de services informatiques dématérialisées permettant un usage flexible de ressources (calcul, stockage, réseau, etc...) pour des usages principalement professionnels
- Il existe plusieurs types de cloud
 - Privé
 - Public
 - Hybride

Les leaders du cloud public



LES GRANDS PRINCIPES DU CLOUD PUBLIC



**On-demand
self-service**

No human
intervention
needed to get
resources



**Broad network
access**

Access
from
anywhere



**Resource
pooling**

Provider
shares
resources
to
customers



**Rapid
elasticity**

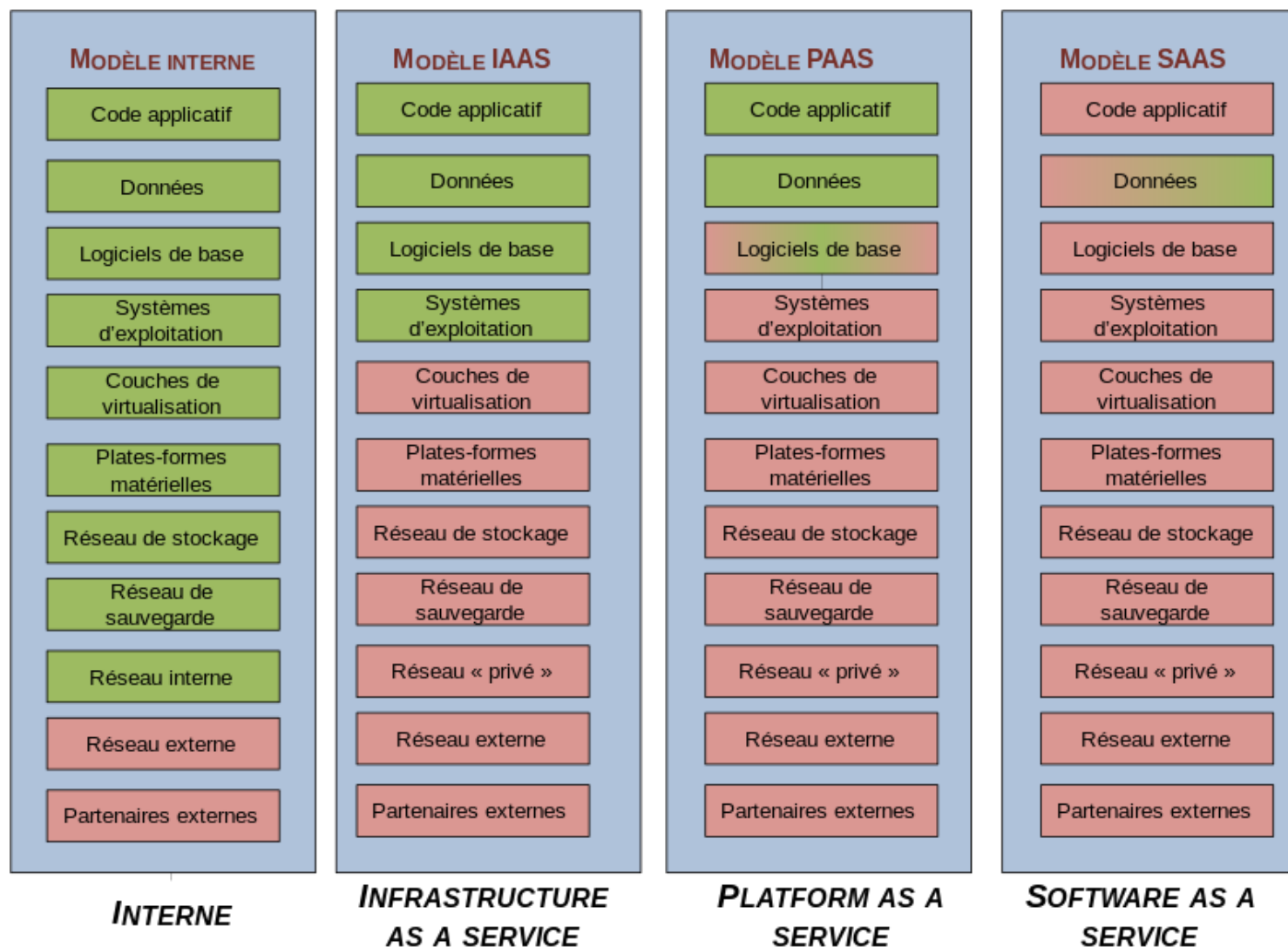
Get more
resources
quickly as
needed



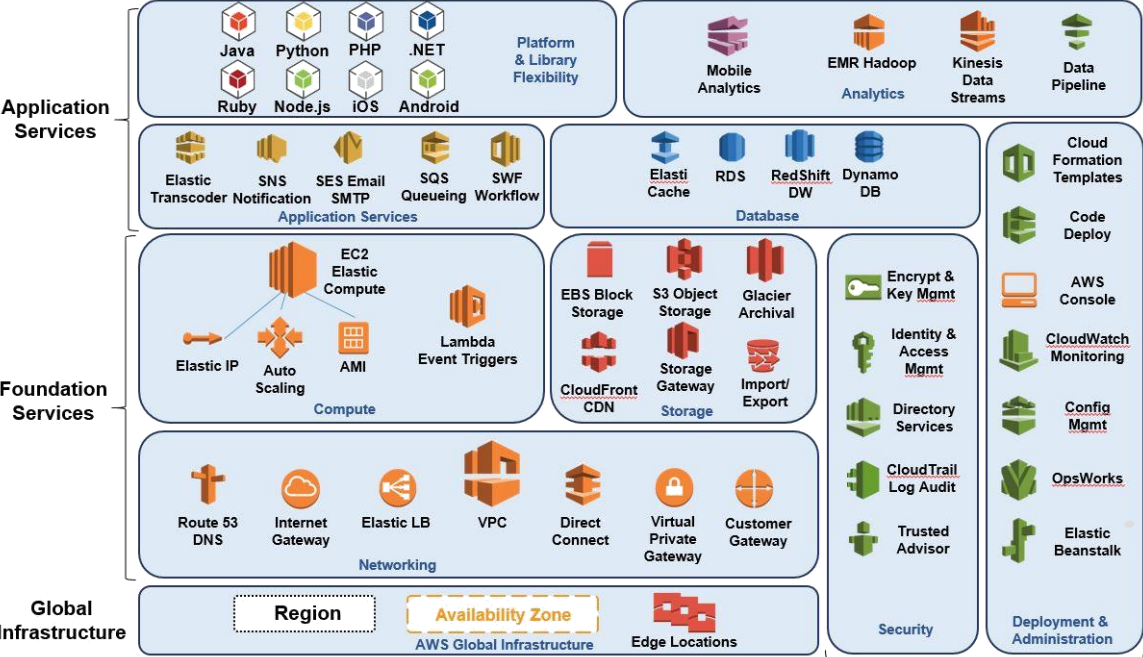
**Measured
service**

Pay only
for what
you
consume

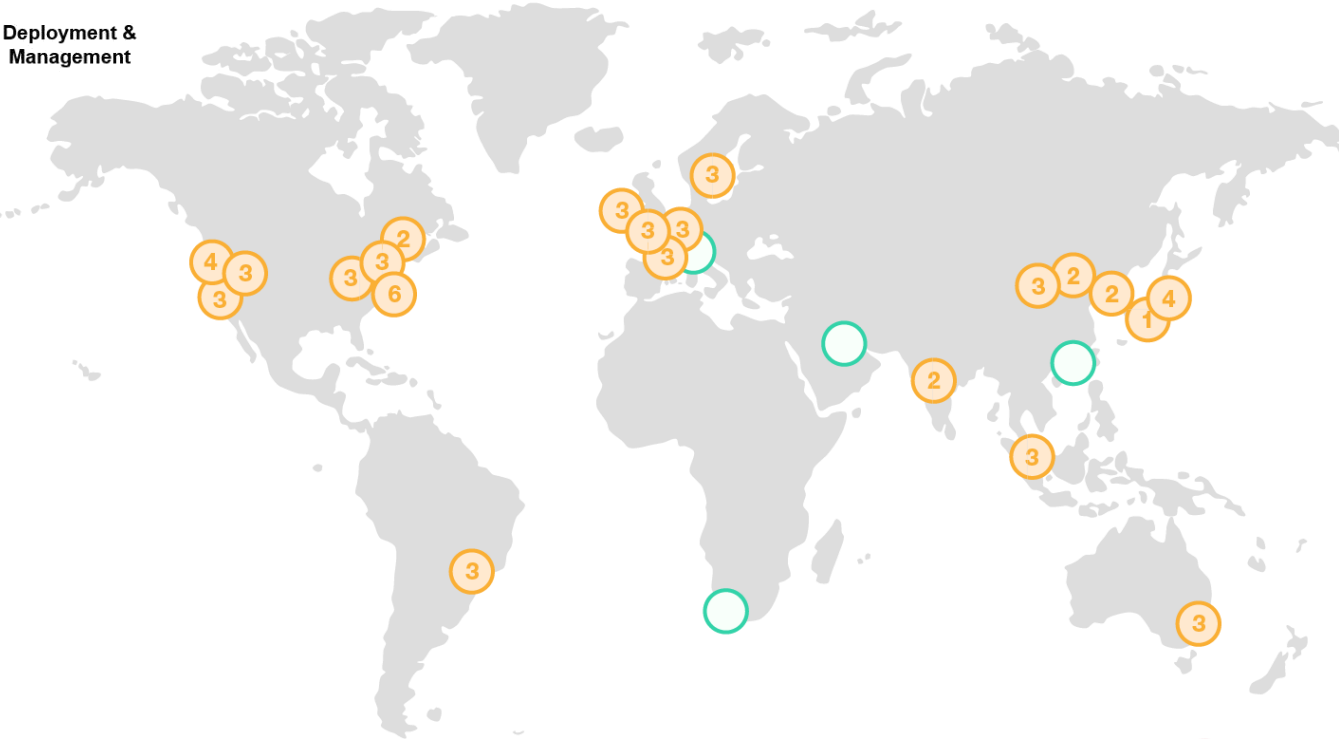
LES MODÈLES DU CLOUD



FOCUS SUR AWS



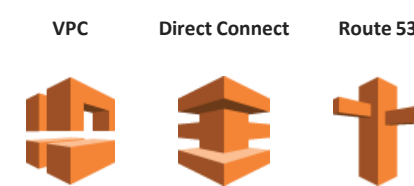
Deployment & Management



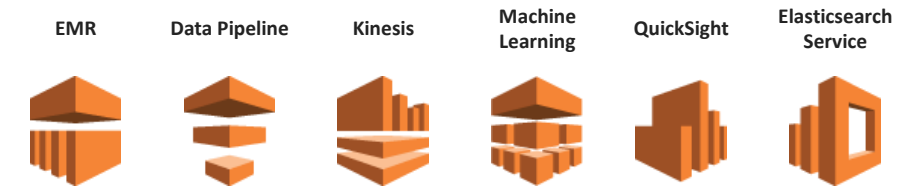
Compute



Networking



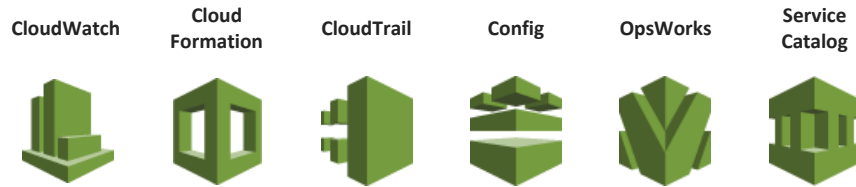
Analytics



Developer Tools



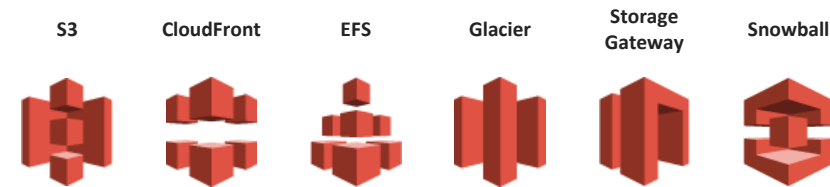
Management Tools



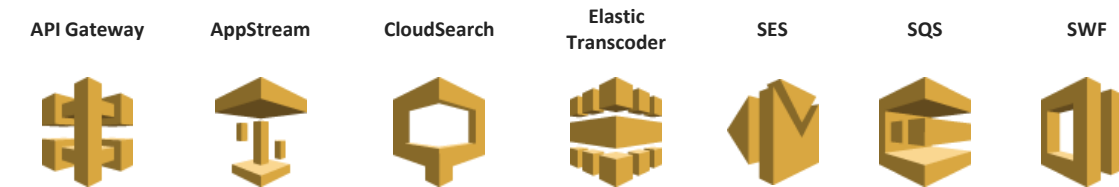
Security & Identity



Storage & Content Delivery



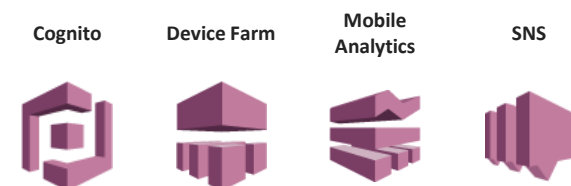
Application Services



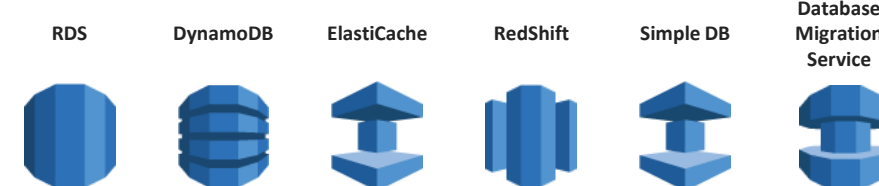
Hubs



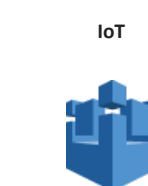
Mobile Services



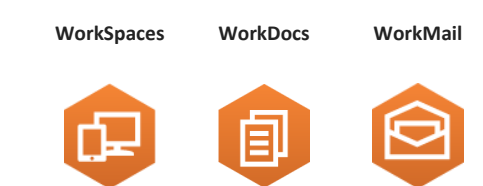
Database



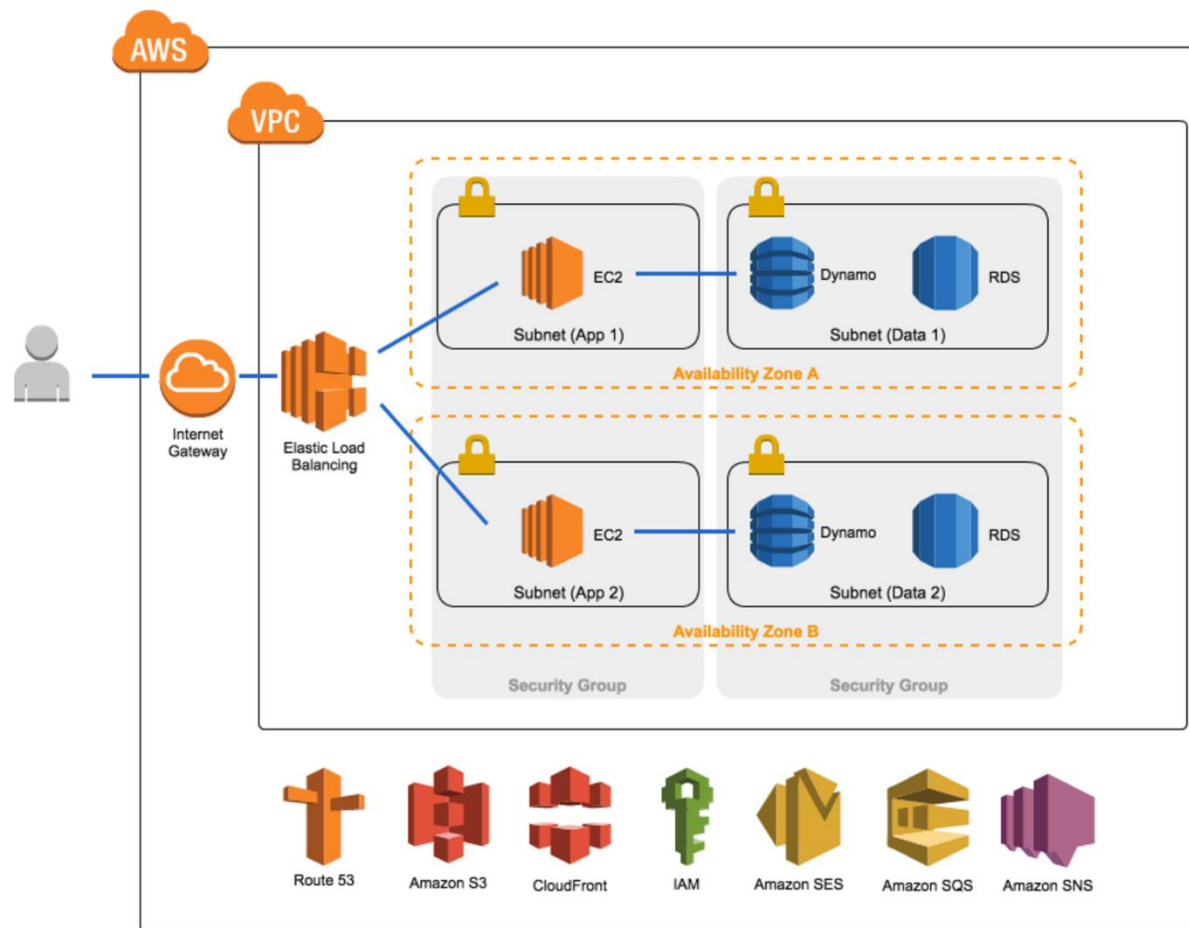
IOT



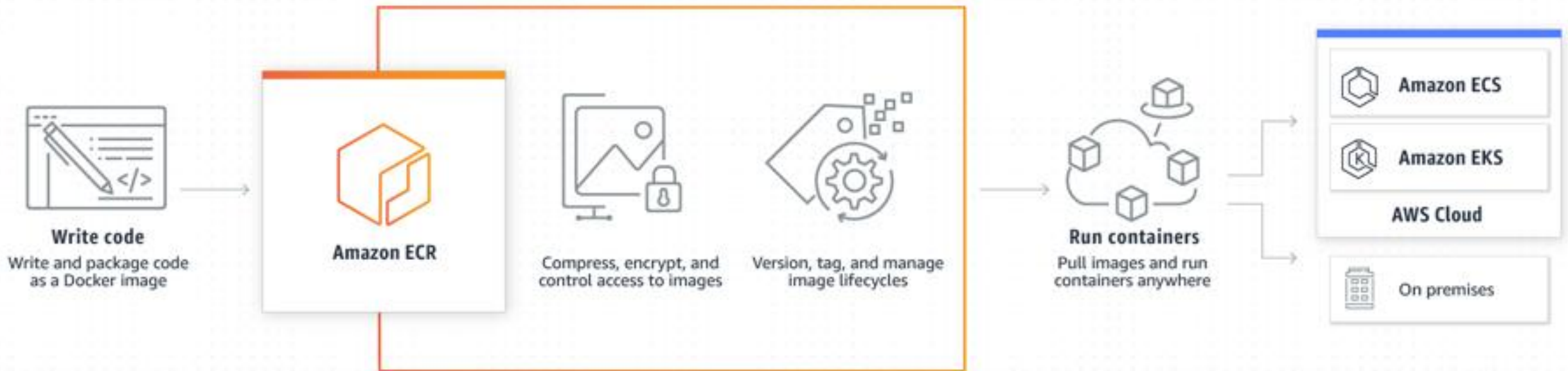
Enterprise Applications



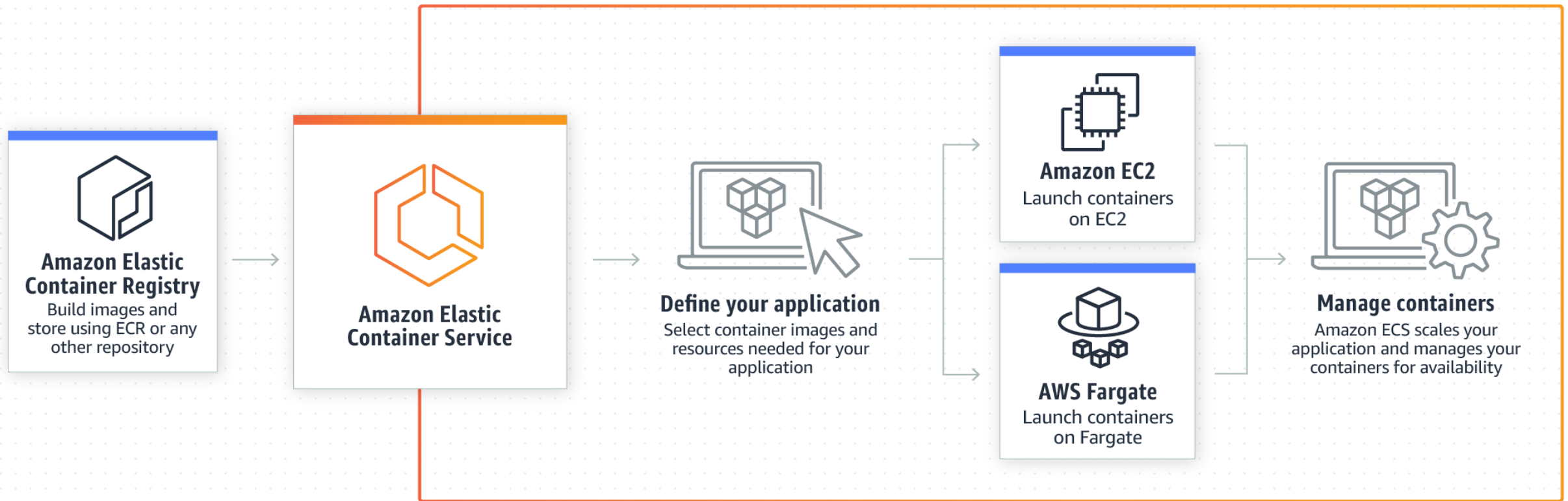
AWS : EXEMPLE D'ARCHITECTURE



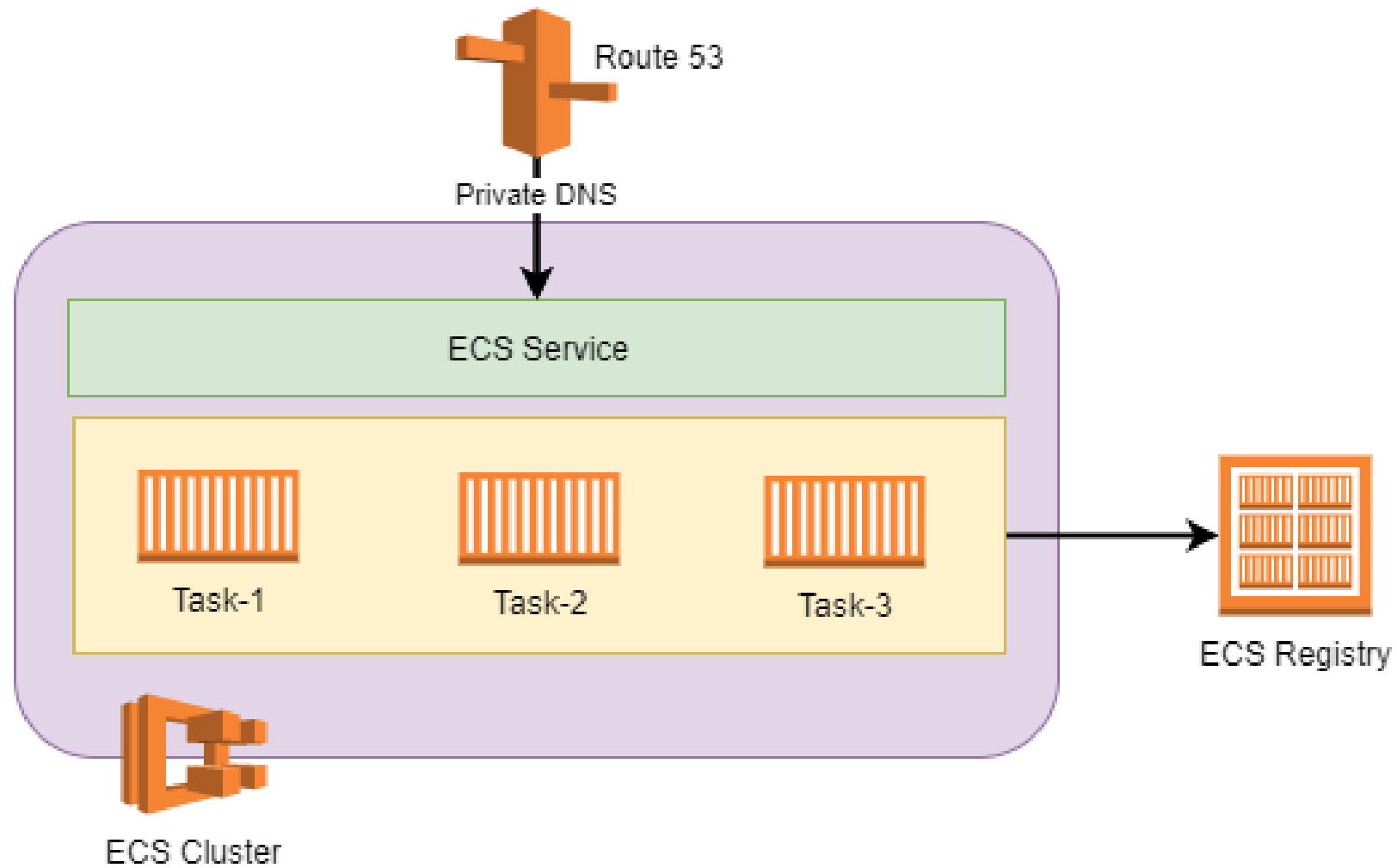
ECR



ECS



AWS : ECS



DÉMO : DOCKER DANS AWS



C'est quoi le "serverless"



Abstraction
des serveurs

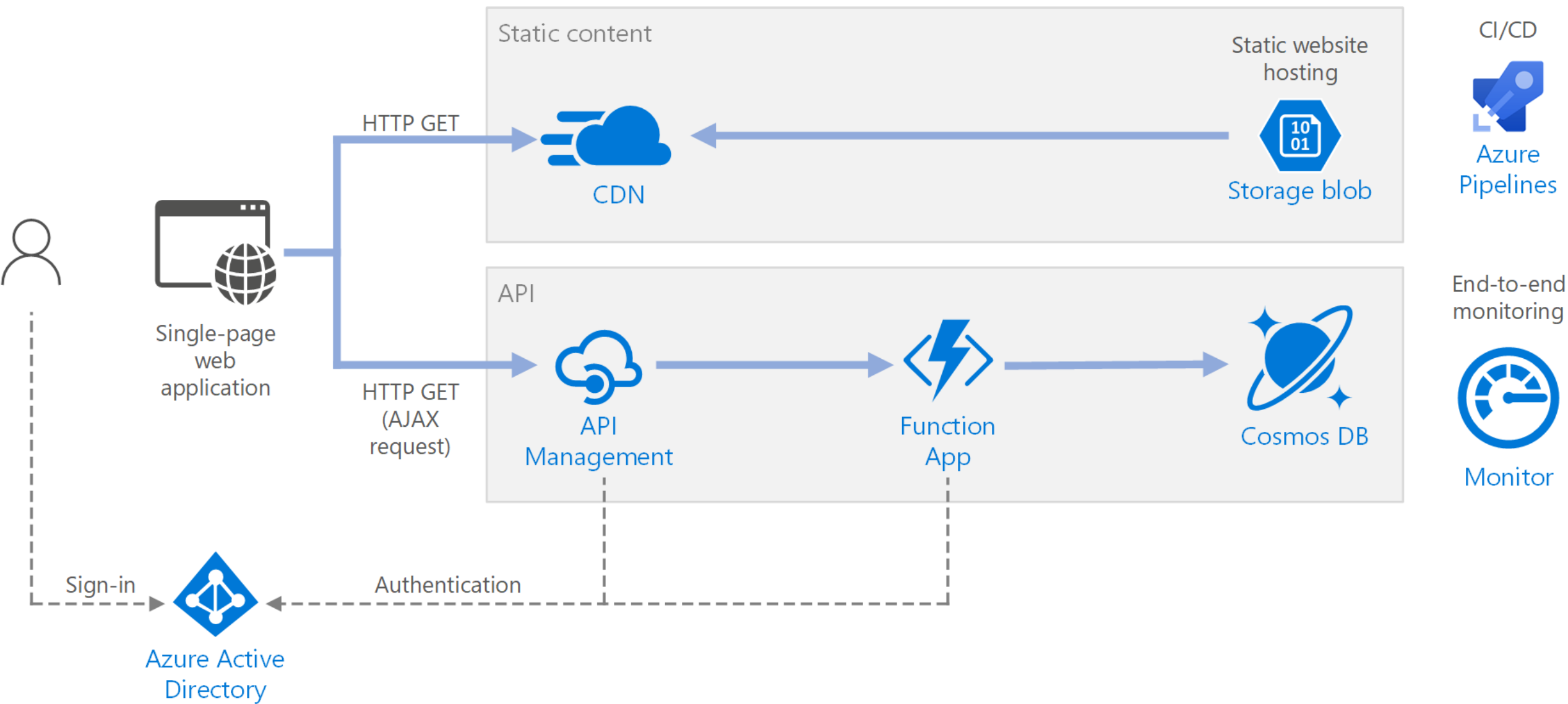


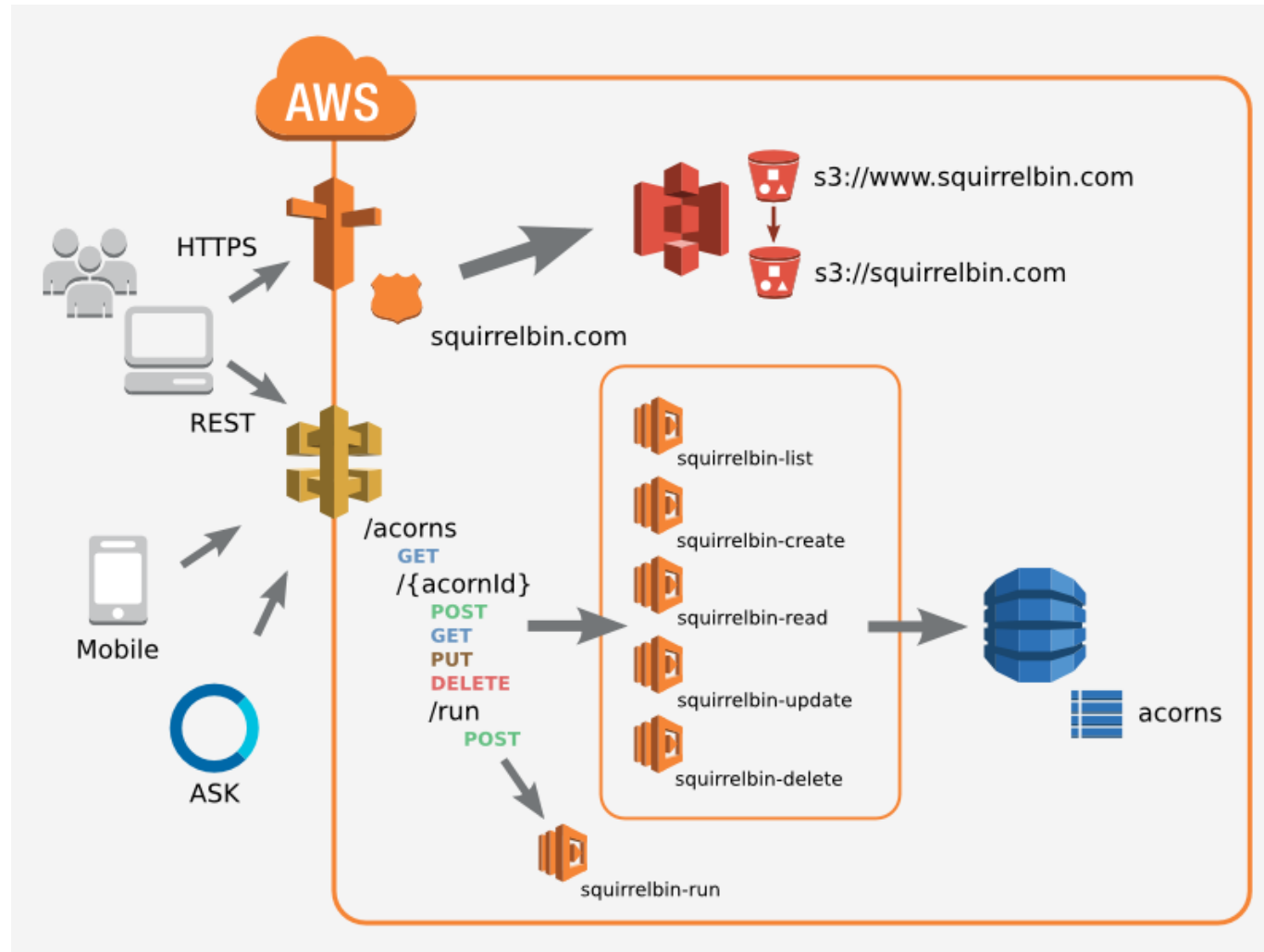
Basé sur des
événements



Paiement à
la demande

EXEMPLE SUR AZURE





Advantages of Serverless Computing



	Bare Metal	VM	Container	Serverless
Boot Time	~20 mins	~2 mins	2 secs	~0.0003 secs
App deployment lifecycle	Deploy in Weeks Live for years	Deploy in minutes Live for weeks	Deploy in Seconds Live for minutes/hours	Deploy in milliseconds Live for seconds
Development Complexity	Need to know: 1. Hardware 2. OS 3. Runtime Environm 4. Application code	Need to know: 1. OS 2. Runtime Environme 3. Application code	Need to know: 1. Runtime Environment 2. Application code	Need to know: 1. Application code
Investment	Buy/rent dedicated server	Rent a dedicated VM, on a shared server	Rent Containers, pay for the actual runtime	Pay for compute resouces used during runtime
Scaling	Takes months Should be approved by a panel of experts	Takes hours Should be approved by adminstators	Takes seconds Policy driven scaling	Takes milliseconds Scaling is event driven

AVANTAGES/INCONVÉNIENTS

- Réduction des coûts
 - Pas de serveur *Always On*
 - Paiement au compute
- Facilité de mise à l'échelle
 - Auto-scaling
- Green IT
 - Mutualisation des ressources
 - Pas de serveur inactif
 - Importance de l'optimisation
- Dépendant du cloud provider
 - Function chez Azure
 - Lambda chez AWS
 - Cloud Function chez GCP
- Verrouillé aux langages fourni
 - .NET/JavaScript/Java/Python chez Azure
 - Java/Node.JS/C#/Python chez AWS
 - JavaScript/Python chez GCP
- Sécurité
 - Pas de verrouillage réseau

