



# Cloud

Optimiser le déploiement de son  
application web grâce au Cloud

# Présentation du cours

# Qui suis-je ?

- EPITA 2015 - ACDC / YAKA / ACU
- Développeur freelance full-stack
- CTO @ Flappy → <https://flappy.tech>
- Prof en 2e, 3e, 4e et 5e année à l'ESGI
  - Filières : AL / MOC / IABD / IW
  - Matières : Flutter / Cloud / Linux / Serveurs Web
- Mail : [kiffer.raphael+prof@gmail.com](mailto:kiffer.raphael+prof@gmail.com)

# Déroulement du cours

- attention aux retards
- soyez attentifs
- mise à disposition du support de cours
- notation
  - CC1 → QCM / questions ouvertes
  - examen → TP noté
- je prends vos feedbacks !

# Prérequis

Concepts requis pour suivre le cours correctement :

- développement web basique
- hébergement linux

Concepts non requis mais qui peuvent bien aider :

- git

# Jusqu'où irons-nous ?

Plusieurs parties du cours :

- introduction au Cloud : 2-3h
- plateformes Cloud : 12h
- architecture *serverless* : 3h

Le but étant d'arriver à la fin avec une vraie connaissance du Cloud et des plateformes à disposition pour optimiser le déploiement de son application web

# Introduction au Cloud

# Cloud ?

Mise à disposition de **ressources numériques**, surtout en terme de **stockage** et de **puissance de calcul**, accessible partout, facilement et rapidement **via Internet**

C'est finalement le terme utilisé pour décrire les **centres de données** accessibles au plus grand nombre **via Internet**



# Points clefs

- Rapidité et facilité de mise en place
- Accessibilité
- Disponibilité continue
- Sécurisation des données
- Évolution des ressources au besoin

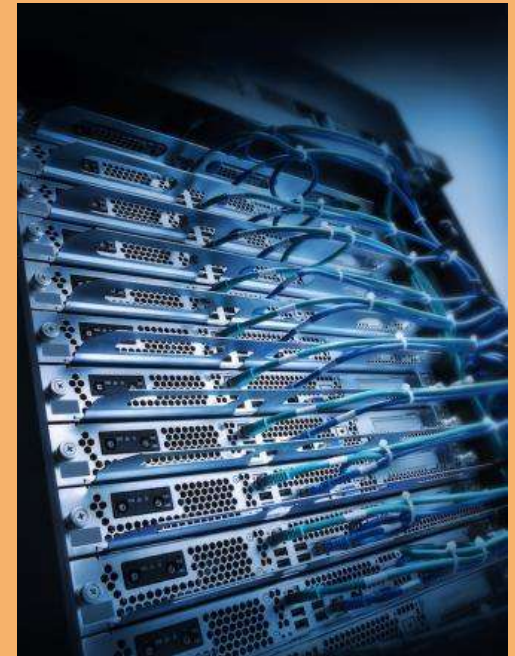
# Chronologie

- **1956** : J. McCarthy, définition de l'intelligence artificielle
- **1960** : J.C.R. Licklider, "Man-Computer Symbiosis"
- **1962** : J.C.R. Licklider, premières idées d'un réseau mondial
- **1966** : Début d'ARPANET (**1969-1990**)
- **1980** : Séparation du réseau militaire et universitaire
- **1980s** : Nécessité d'étaler des tâches sur un réseau
- **1991** : T. Berners-Lee, inventeur de HTTP
- **1997** : R. Chellappa, naissance du mot "Cloud"
- **1999** : Salesforce.com, introduction du concept SaaS
- **2002 > 2006** : Lancement d'AWS

# Centre de données

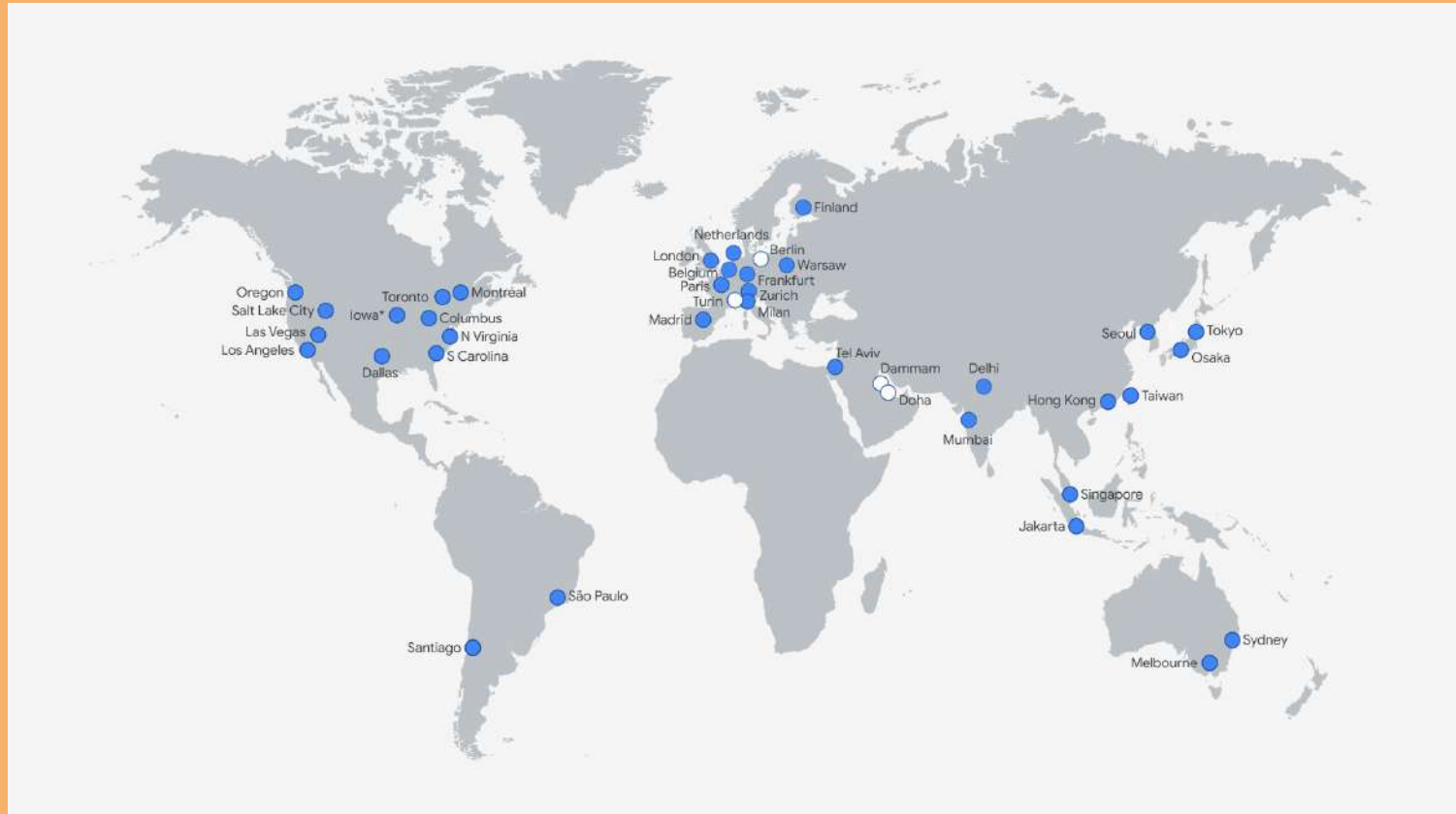


*Centre de données ou Datacenter*



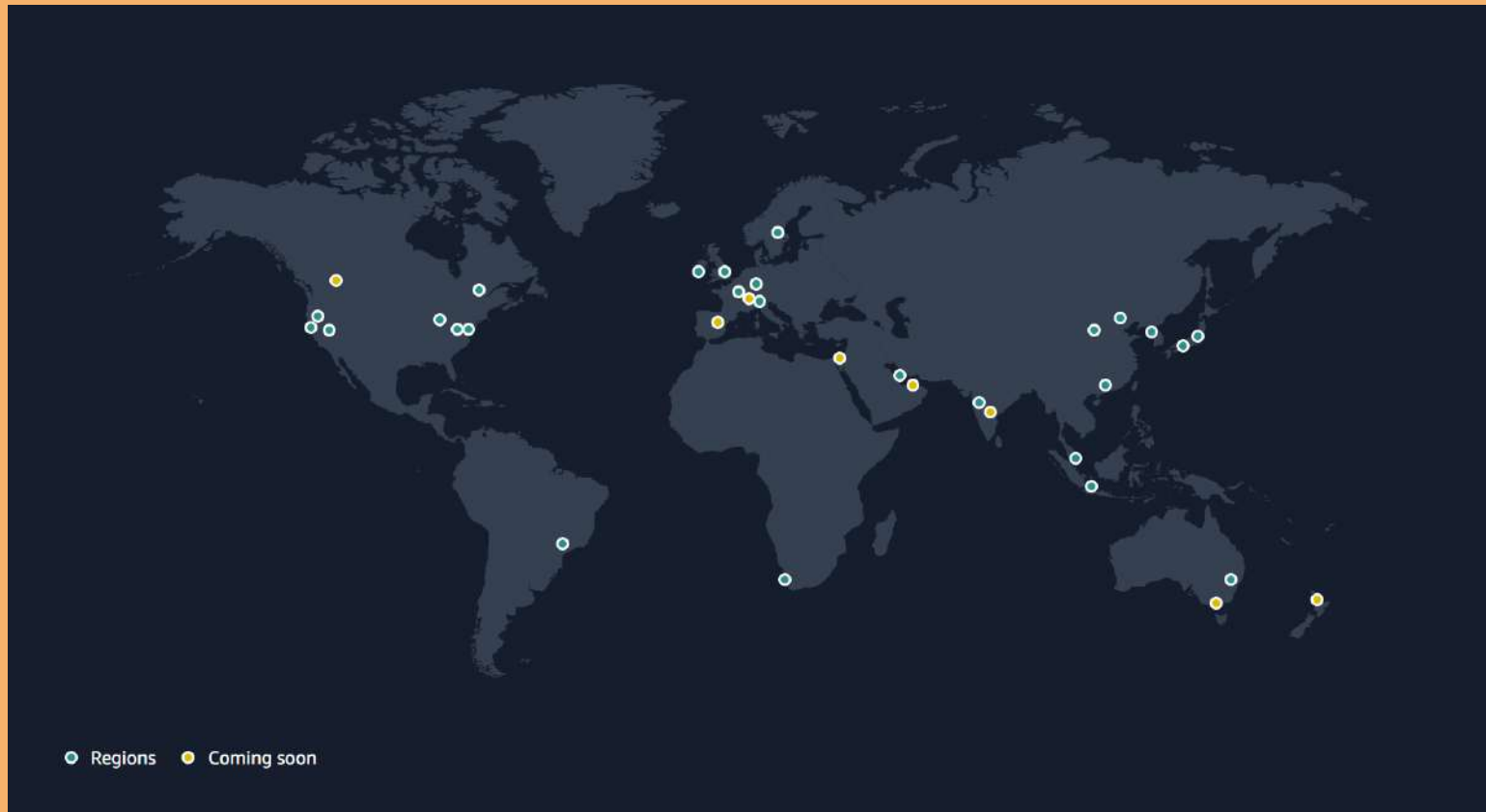
*Serveurs*

# Répartition mondiale



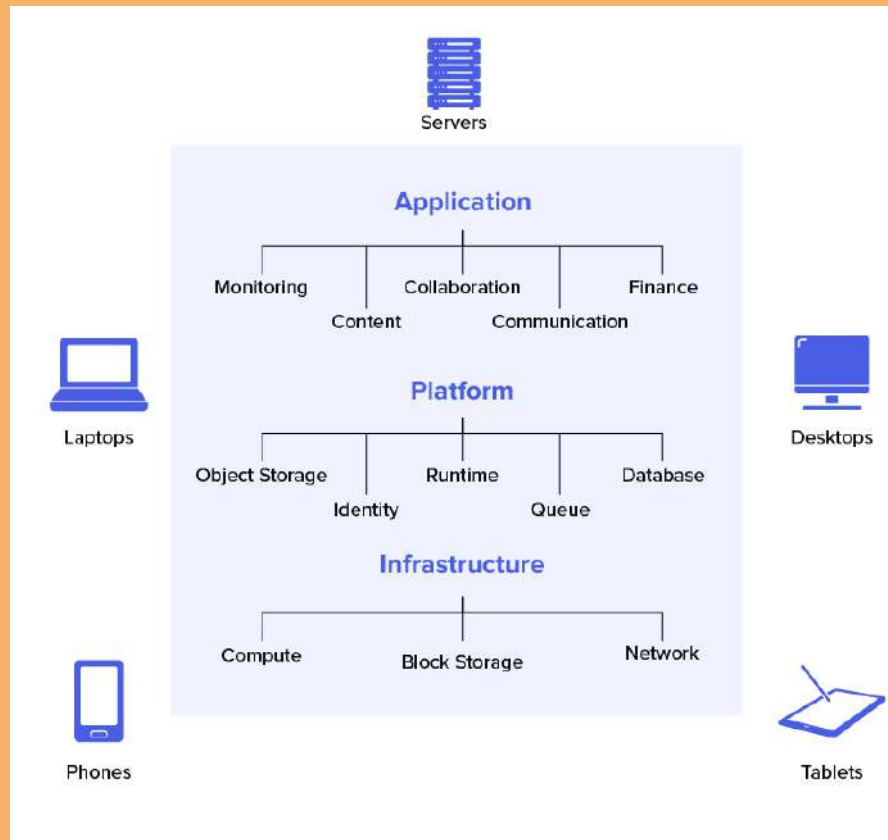
*Datacenters - Google Cloud*

# Répartition mondiale

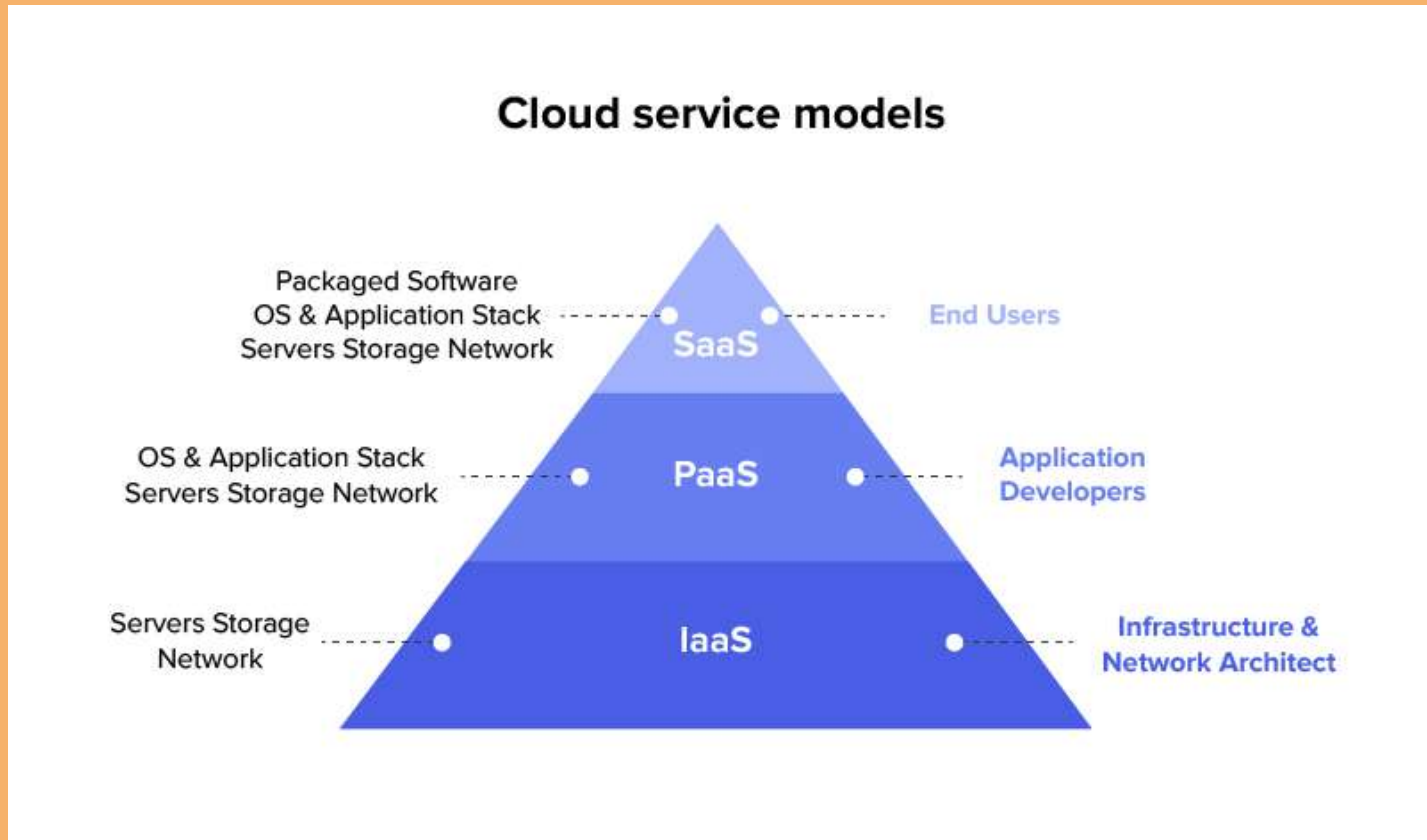


*Datacenters - AWS*

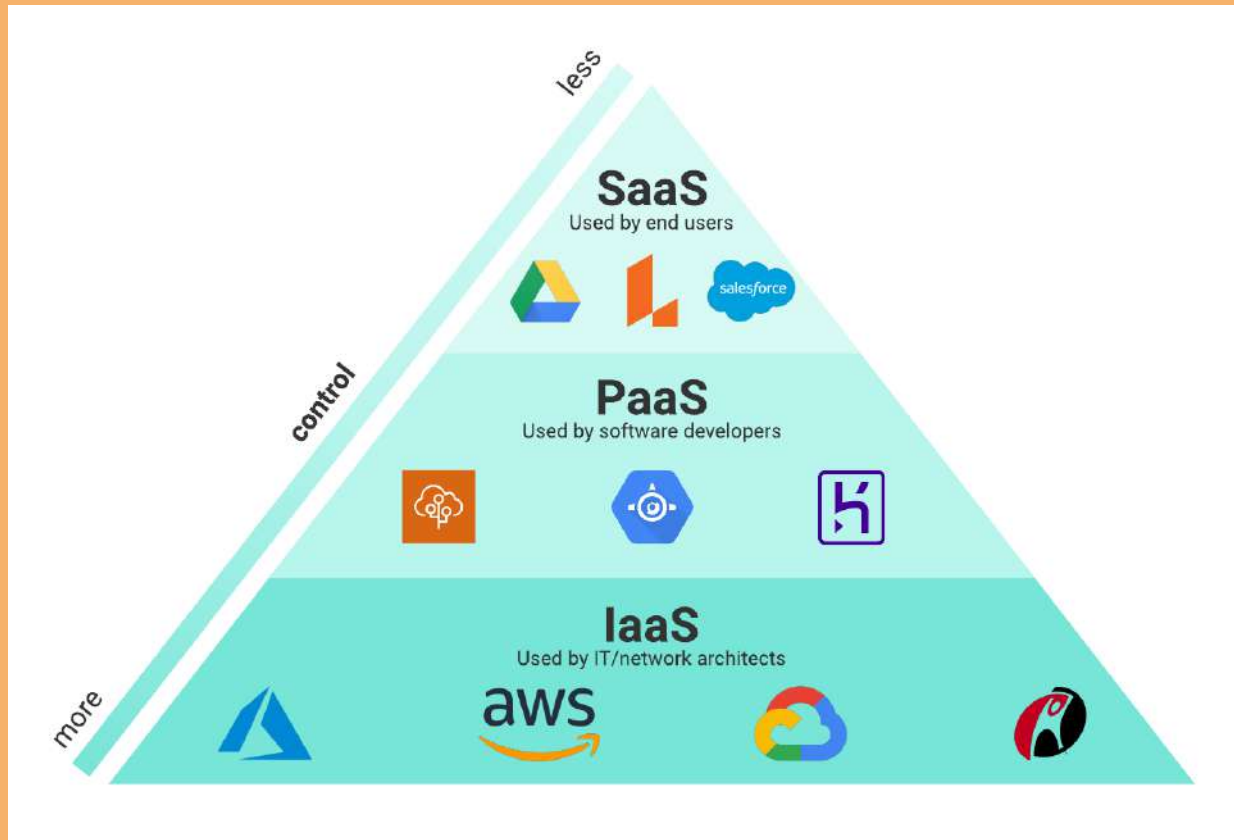
# Différents types de Cloud



# Différents types de Cloud



# Différents types de Cloud





# IaaS

## Infrastructure as a Service

- Mise à disposition d'une partie d'une **infrastructure technique**
- Ressources brutes
- Problèmes **matériels** gérés par le prestataire
- Assez **bas niveau**
- La structure complète reste à faire

# PaaS

## Platform as a Service

- Sorte de **surcouche** des IaaS
- Déploiement et gestion de son application
- Service **semi-automatisé** pour la gestion des ressources
- Possibilité de configurer et de **mettre à l'échelle** rapidement
- Relation de confiance avec la plateforme

# SaaS

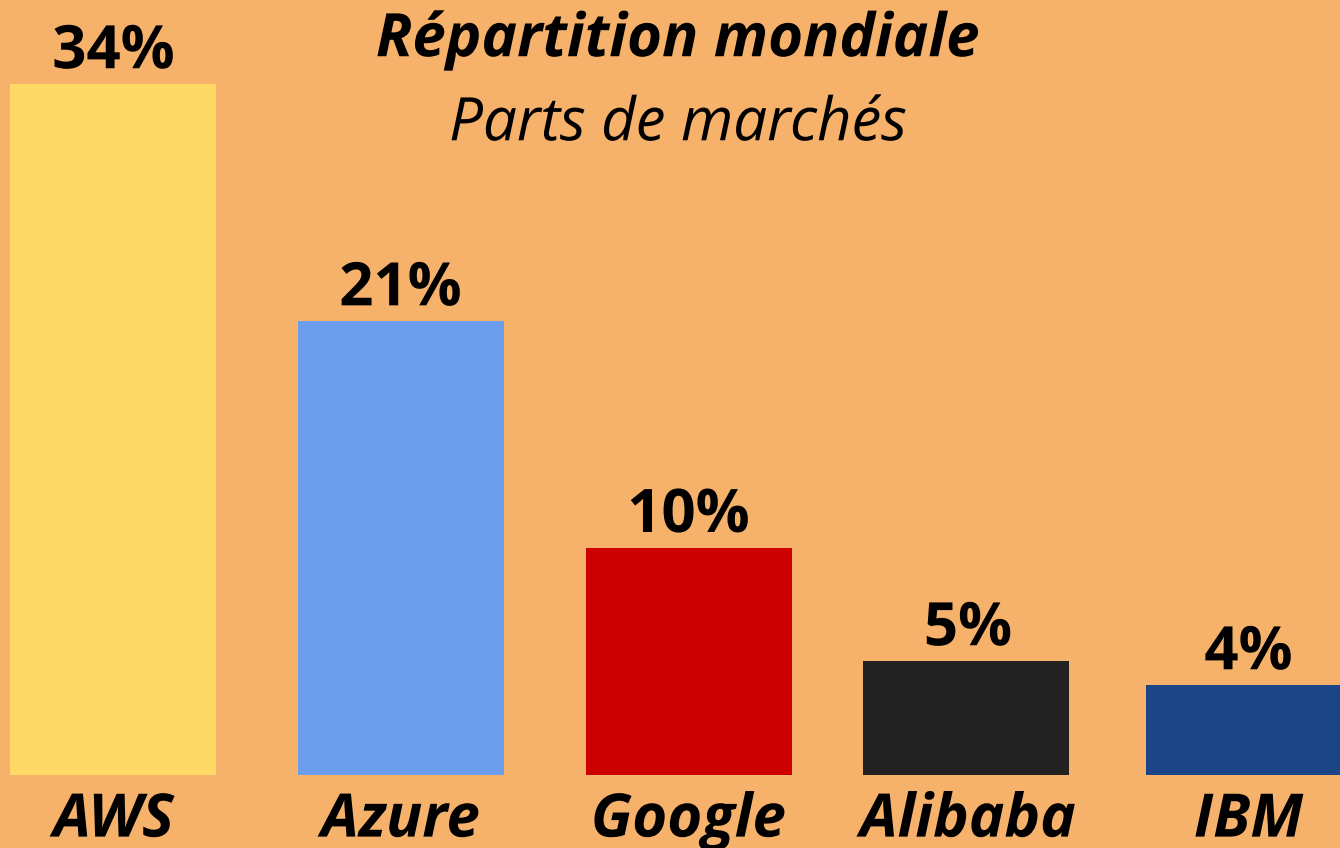
## Software as a Service

- Mise à disposition de **ressources logicielles**
- **Accessible partout** depuis son navigateur
- Souvent via abonnement
- Pas de **version** spécifique
- Mises à jour incluses

# IaaS & PaaS



# IaaS & PaaS



# SaaS

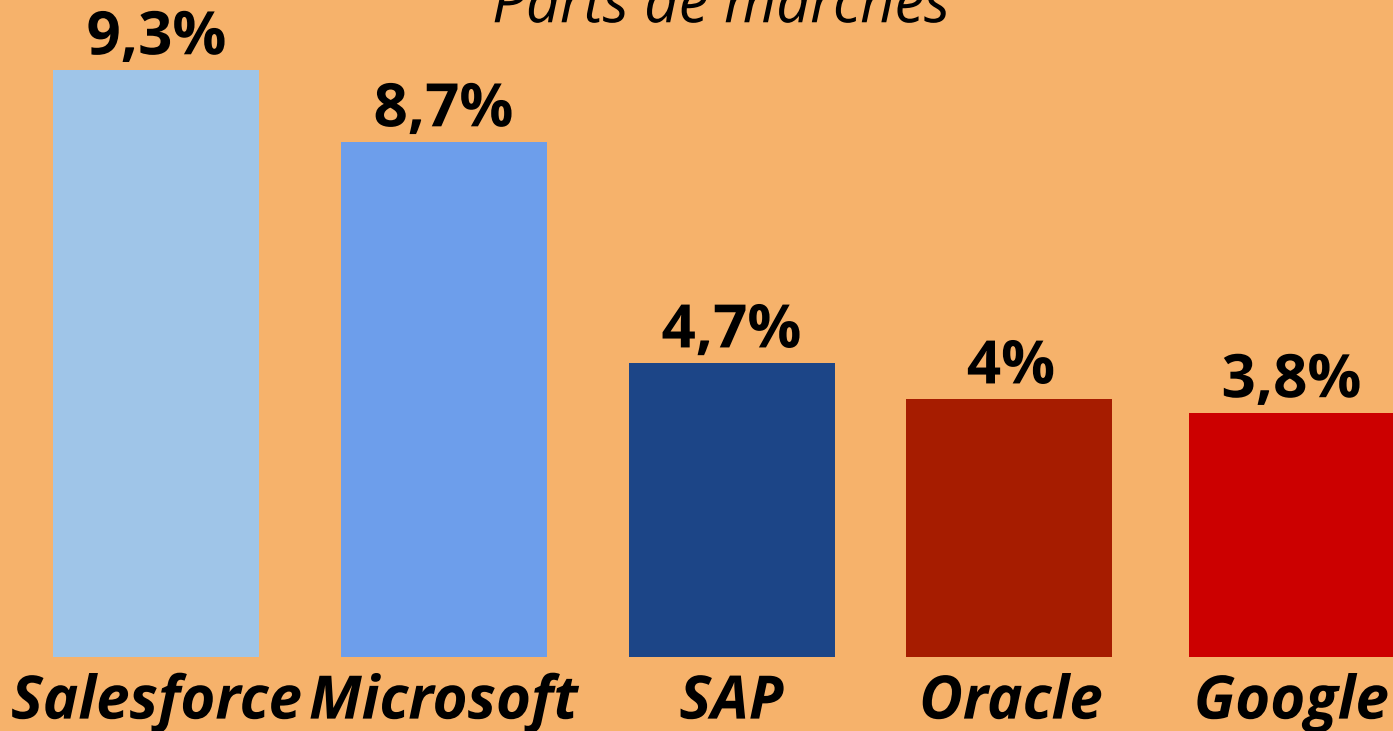
 Suite  Spotify® **NETFLIX**

 **slack**  **Dropbox**

 Jira Software

# SaaS

## *Répartition mondiale* *Parts de marchés*



# En résumé

- Beaucoup d'offres différentes
- Évaluation des coûts
  - Service choisi
  - Durée d'utilisation
  - Capacité requise
  - Région du centre de données



# Plateformes

# Amazon Web Services

- Lancement officiel en **mars 2006**
- Division d'Amazon
- IaaS / PaaS
- **+ de 175** produits et services
- Présent dans le monde entier
- *Première* plateforme Cloud



# Microsoft Azure

- Lancement officiel en **février 2010**
- Division de Microsoft
- IaaS / PaaS
- **+ de 600** produits et services
- Présent dans le monde entier
- *Deuxième* plateforme Cloud



# Google Cloud

- Lancement officiel en **2011**
- Beta depuis **2008**
- IaaS / PaaS
- Grosses infrastructures de Google
- Réponse à AWS et Azure
- Peine à rattraper ses concurrents



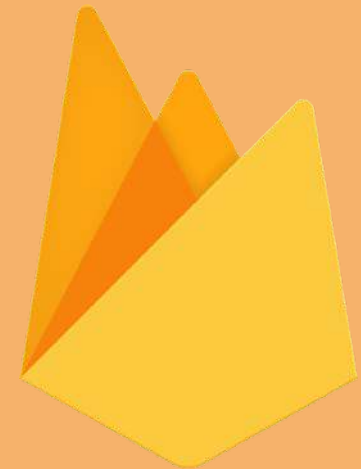
# Heroku

- Lancement officiel en **2007**
- Racheté par Salesforce en **2010**
- Build, Run & Scale
- Ruby puis Java, Node.js, PHP, ...
- Basé sur **AWS**
- Système d'**addons**
- Offre gratuite maintenant payante



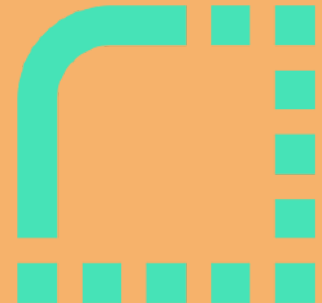
# Firebase

- Lancement officiel en **2011**
- Rachat par **Google** en **2014**
- Outils pour développer sur le web et le mobile
  - Stockage, base de données, ...
  - Analytics, gestion d'erreur, notifications, ...
- Gratuit ou payant à l'usage



# Render

- Lancement officiel en **2019**
- Vainqueur du *TechCrunch Disrupt Battlefield* la même année
- Outils et technologies Cloud modernes PaaS
- Concurrent direct de **Heroku**
- Bonne **offre gratuite**



# Heroku



# Heroku

Heroku est une **PaaS** permettant de **déployer** facilement son application web dans le Cloud, tout en ayant la possibilité de la **mettre à l'échelle** via des **dynos**

Le fonctionnement est simple, il suffit de créer une application sur la plateforme et d'utiliser **Git** pour déployer son code s'il correspond à un **archétype de projet** géré

# Dynos

Les **dynos** sont des conteneurs virtuels légers sur Linux et permettant de faire tourner une application. C'est ce système, qui dans un cas normal, est **facturé au prorata** d'utilisation

Il existe différents types de dynos, plus ou moins chers, en fonction de la **capacité / puissance** choisie

Pour une application, il est possible de **paralléliser** plusieurs dynos afin de la **mettre à l'échelle** de façon **semi-automatique**

# Offre gratuite

L'offre gratuite n'est plus celle qu'elle était il y a 2-3 ans. Suite à de nombreux abus, la seule offre disponible est celle pour les étudiants via **Github Education**

- <https://education.github.com/pack>

Il suffit de s'authentifier avec **l'adresse mail de son école** et d'attendre l'acceptation du programme

Ensuite, il faut s'authentifier sur son compte **Heroku** et **postuler** pour le programme en liant son compte **GitHub**

- <https://www.heroku.com/github-students>

# Offre gratuite

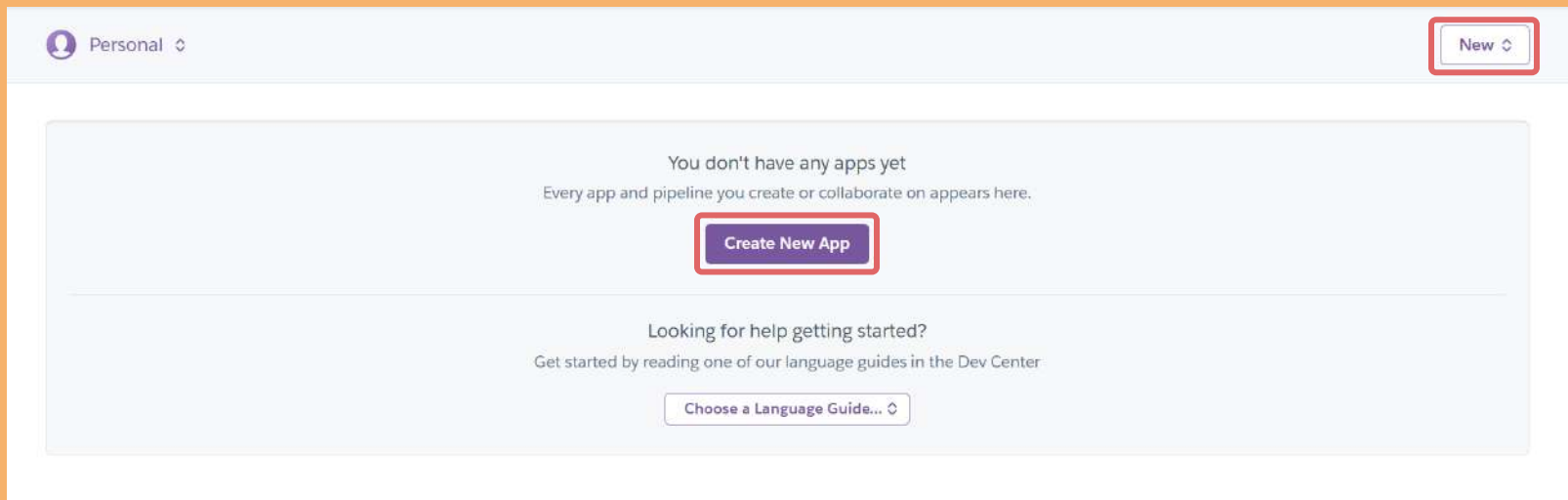
Cette offre permet de débloquent des **crédits** pour la plateforme **renouvelés tous les mois**

**Attention**, dans tous les cas, il est nécessaire d'avoir une **carte de crédit** (qui ne sera pas débitée) pour vérifier son compte

En cas de problème, il existe une offre **Eco** qui permet de reproduire l'usage de l'offre gratuite pour **5€ / mois**

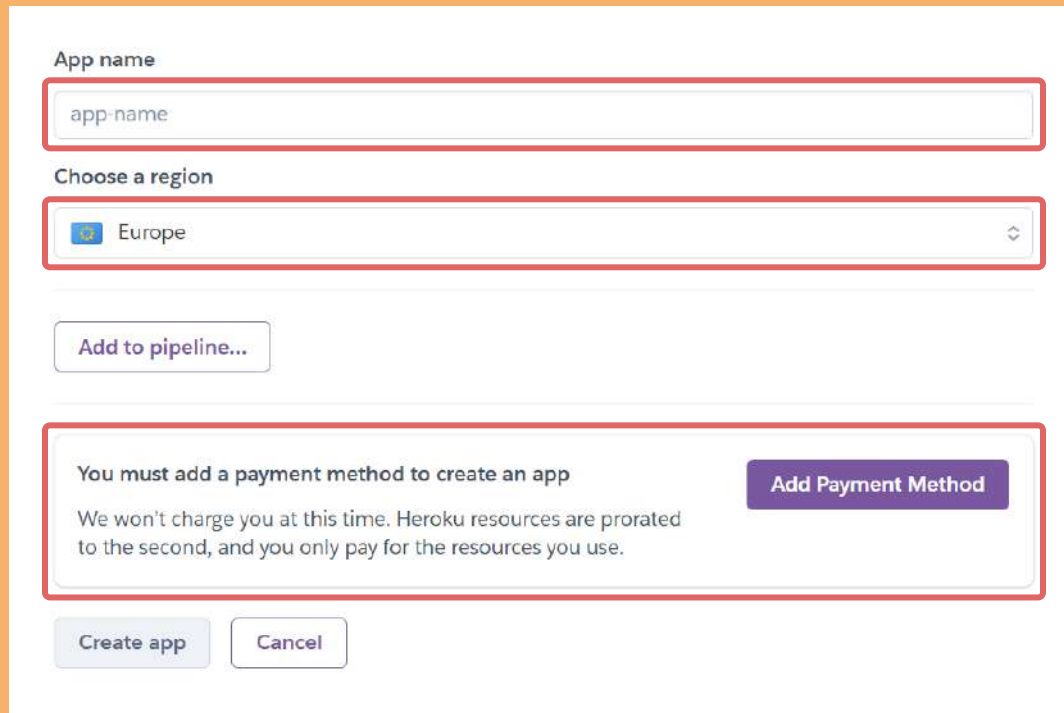
# Créer son application

Une fois connecté, on arrive sur le **dashboard** où l'on peut **créer une nouvelle application**



# Créer son application

Il faut ensuite sélectionner la région appropriée : **Europe**



App name

app-name

Choose a region

Europe

Add to pipeline...

You must add a payment method to create an app

We won't charge you at this time. Heroku resources are prorated to the second, and you only pay for the resources you use.

Add Payment Method

Create app Cancel

# Créer son application

Il n'est pas nécessaire de trouver un **nom d'application**, il sera **généré** automatiquement s'il est vide

Attention, vous devez ajouter une **carte de crédit** dans tous les cas, même si vous ne **payez pas au prorata** d'utilisation

# Heroku CLI

Il existe également une **interface en ligne de commande** qui permet de gérer son application et notamment de facilement la lier avec son **dépôt Git** pour effectuer des déploiements

- <https://devcenter.heroku.com/articles/heroku-cli>

Une fois l'installation terminée, on peut utiliser

```
1 heroku login
2 heroku create --region eu
3 heroku git:remote -a <app-name>
```



# Déploiement

Dans notre projet, il faut impérativement avoir un **dépôt Git**

```
1 git init
```

Une fois que c'est fait, il suffit **d'ajouter** puis de ***commit*** ses fichiers pour enfin ***push*** tout ça sur la ***remote Heroku*** que l'on aura ajouté au préalable

```
1 git add .  
2 git commit -m "Initial commit"  
3 git push heroku master
```

# Buildpacks

Avec **Heroku**, les déploiements sont un peu **magiques** !

C'est dû aux **buildpacks** qui permettent de préconfigurer les **dynos** avec un environnement adapté pour votre projet

Si le projet correspond à un **archétype** supporté officiellement, la détection sera automatique, sinon il faut ajouter le **buildpack** nécessaire

```
1 heroku buildpacks:add <buildpack>
```

# Buildpacks

Il existe beaucoup de **buildpacks** différents, dont la plupart sont fait par la **communauté**

- <https://elements.heroku.com/buildpacks>

En général, quand on utilise autre chose qu'un **buildpack officiellement supporté**, on a besoin de **configurer** son application d'une manière spécifique, souvent décrite dans la **documentation** du buildpack utilisé

# Addons

Heroku permet également d'ajouter des **services tiers** (souvent payant) à son application et de **centraliser la facturation** globale

Ces **services** proviennent en majorité de **fournisseurs différents** mais qui ont un partenariat avec Heroku

- <https://elements.heroku.com/addons>

```
1 heroku addons:attach <addon>
```

# Render

# Render

Render est également une **PaaS** permettant de **déployer** facilement son application web dans le Cloud

Elle propose **plusieurs types de service** en fonction de ce que l'on doit mettre en place. Cela permet aussi bien de **prototyper un projet** que de mettre en place des services en **production**

# Render

Tous les projets ne sont pas de base **déployables** sur la plateforme, il faut respecter certains **langages / frameworks**  
Il existe plusieurs formules, donnant accès à plus ou moins de **fonctionnalités**

- <https://render.com/pricing>

Les services sont dans tous les cas payants au **prorata d'utilisation**

# Offre gratuite

L'offre gratuite de Render est **très complète**, elle permet de déployer facilement certains types de services **sans coût** et **sans rentrer sa carte bleue**

Les services ont tous plus ou moins des **limitations** mais sont largement exploitables

- <https://render.com/docs/free>

Par exemple, les services web gratuits **se mettent en pause** en cas **d'inactivité**



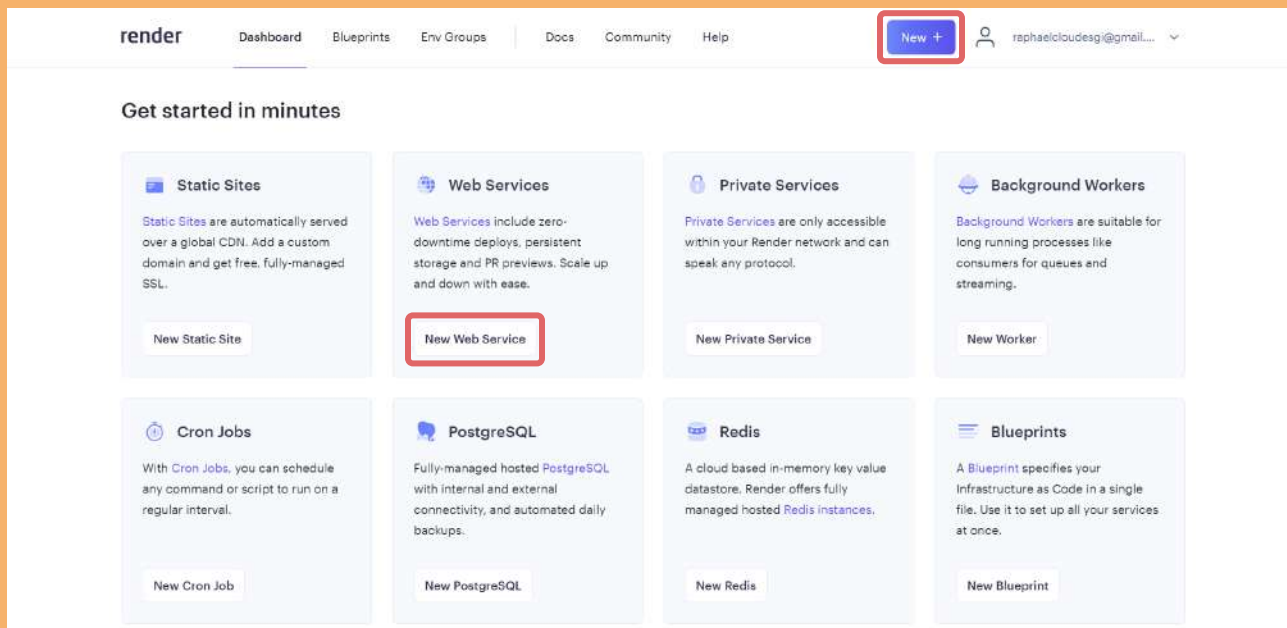
# Application ?

Contrairement à Heroku, ici on ne crée pas une **application** qui va contenir des services mais directement des **services** qui vont potentiellement composer une ou plusieurs applications

Il existe également la possibilité de gérer son ***Infrastructure as Code*** pour rassembler et faire évoluer les services de son application

# Créer son service

Une fois connecté, on arrive sur le **dashboard** où l'on peut **créer un nouveau service**



# *Web Services*

Le service qui s'apparente le plus au déploiement d'une application web est le **Web Service**

Au moment de la création on nous propose d'utiliser un **dépôt Git** ou une **image depuis un registre**

Il est possible avec Git d'utiliser un dépôt **public** ou un dépôt **privé**

- <https://render.com/docs/web-services>

# *Web Services*

Le projet à déployer doit correspondre à des **langages / technologies spécifiques**

- <https://render.com/docs/language-support>

Pour des projets privés sur GitHub, il faut lier son compte

How would you like to deploy your web service?

- ☒ Build and deploy from a Git repository  
Connect a GitHub or GitLab repository.

# *Web Services*

Une fois le dépôt sélectionné, il faut spécifier certaines informations comme le **nom** du service, sa **région**, son **runtime**, les commandes permettant de le **construire** et de le **démarrer**, ...

Il y a une détection automatique de **l'archétype du projet** pour **préremplir** certains champs

La dernière étape consiste à sélectionner le **type d'instance**

**Runtime**

The runtime for your web service.

Node

**Build Command**

This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

\$ npm install

**Start Command**

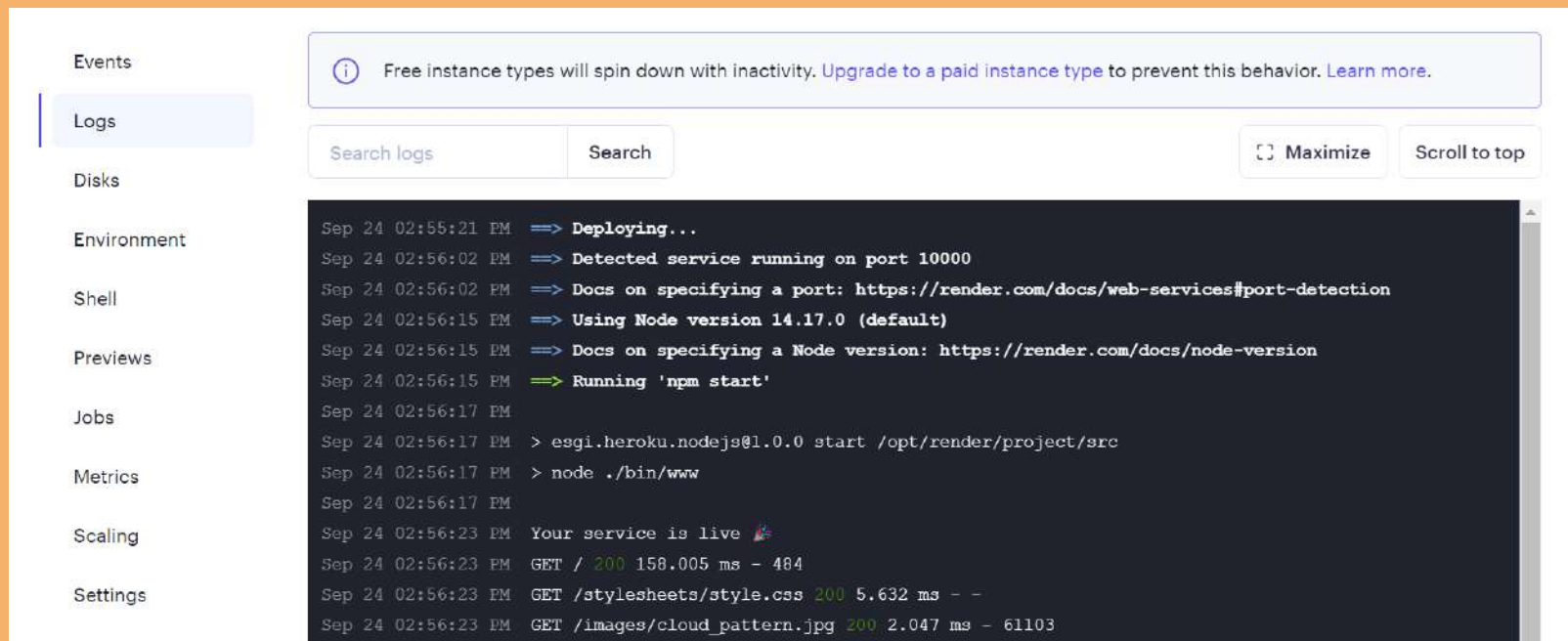
This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.

\$ npm start

*Exemple de configuration pour un projet node.js*

# Web Services

Quand la configuration est terminée, il suffit de créer le service pour être redirigé sur la **console de logs**



The screenshot displays the Render console interface. On the left is a sidebar with navigation links: Events, Logs (selected), Disks, Environment, Shell, Previews, Jobs, Metrics, Scaling, and Settings. The main area features a light blue notification banner at the top stating: "Free instance types will spin down with inactivity. Upgrade to a paid instance type to prevent this behavior. Learn more." Below this is a search bar with the text "Search logs" and a "Search" button, along with "Maximize" and "Scroll to top" buttons. The log output is shown in a dark-themed terminal window with the following content:

```
Sep 24 02:55:21 PM => Deploying...
Sep 24 02:56:02 PM => Detected service running on port 10000
Sep 24 02:56:02 PM => Docs on specifying a port: https://render.com/docs/web-services#port-detection
Sep 24 02:56:15 PM => Using Node version 14.17.0 (default)
Sep 24 02:56:15 PM => Docs on specifying a Node version: https://render.com/docs/node-version
Sep 24 02:56:15 PM ==> Running 'npm start'
Sep 24 02:56:17 PM
Sep 24 02:56:17 PM > esgi.herokuapp@1.0.0 start /opt/render/project/src
Sep 24 02:56:17 PM > node ./bin/www
Sep 24 02:56:17 PM
Sep 24 02:56:23 PM Your service is live 🎉
Sep 24 02:56:23 PM GET / 200 158.005 ms - 484
Sep 24 02:56:23 PM GET /stylesheets/style.css 200 5.632 ms - -
Sep 24 02:56:23 PM GET /images/cloud_pattern.jpg 200 2.047 ms - 61103
```

# PostgreSQL

On peut également créer une base de données **PostgreSQL** pour héberger son modèle de données relationnel

C'est encore une fois **très simple**, il suffit de spécifier un nom et une région pour le service et de potentiellement ajouter **le nom de la base** ainsi que **l'utilisateur** désiré

Une base de donnée **gratuite** sera tout le temps **disponible** mais **ne durera que 90 jours**

- <https://render.com/docs/databases>



# *Static sites*

Enfin une autre possibilité est de **déployer** un site statique, qui va être **distribué** via un **CDN**

Ce site peut être complètement statique ou nécessiter un ***build*** au préalable dans le cas d'un ***framework front***

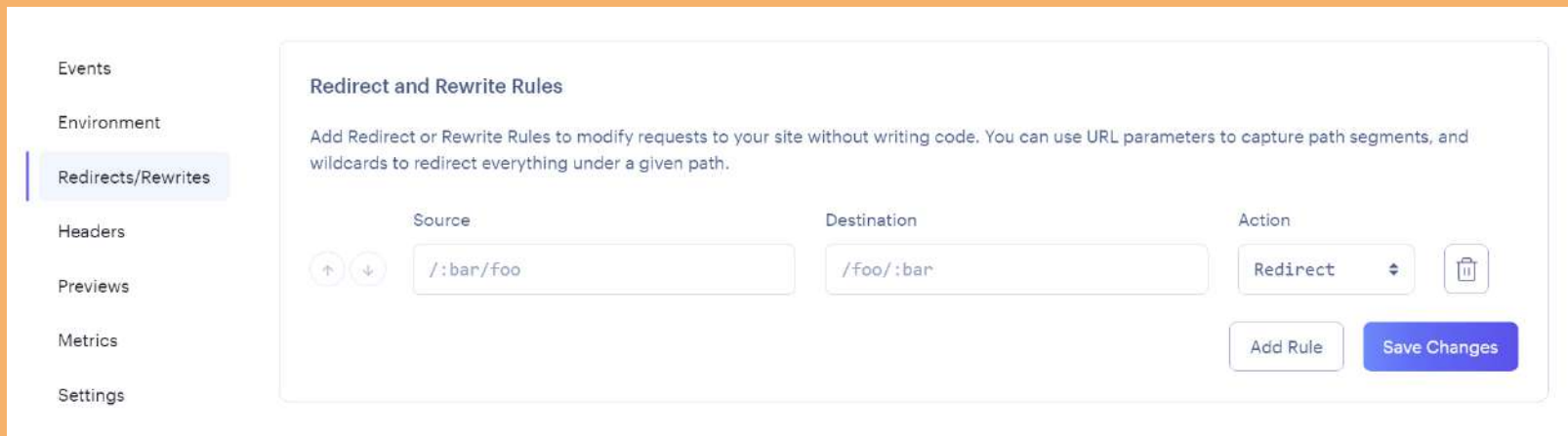
- <https://render.com/docs/static-sites>

Il est possible de spécifier le **dossier racine** ainsi que celui contenant les ***assets*** à publier

# Static sites

On peut également spécifier des **redirections** ou des **réécritures d'URL** pour personnaliser le comportement de son site statique une fois déployé

- <https://render.com/docs/redirects-rewrites>



The screenshot shows the 'Redirects/Rewrites' configuration page in the Render dashboard. On the left is a sidebar with navigation links: Events, Environment, Redirects/Rewrites (highlighted), Headers, Previews, Metrics, and Settings. The main content area is titled 'Redirect and Rewrite Rules' and includes a descriptive text: 'Add Redirect or Rewrite Rules to modify requests to your site without writing code. You can use URL parameters to capture path segments, and wildcards to redirect everything under a given path.' Below this, there is a table with three columns: 'Source', 'Destination', and 'Action'. The 'Source' column has a dropdown menu with up and down arrows and a text input containing '/:bar/foo'. The 'Destination' column has a text input containing '/foo/:bar'. The 'Action' column has a dropdown menu with 'Redirect' selected and a trash icon. At the bottom right of the table are two buttons: 'Add Rule' and 'Save Changes'.

Source	Destination	Action
<div>↑ ↓</div> /:bar/foo	/foo/:bar	Redirect <div>🗑️</div>

Add Rule Save Changes

# Render CLI

Il est possible d'installer **l'interface en ligne de commande** pour **automatiser** le déploiement de vos services

- <https://render.com/docs/cli>

Cependant, elle est actuellement encore en **alpha**, elle n'est donc **pas aussi complète** que peut l'être celle d'Heroku

# ***Blueprints***

Dans le cas d'un projet plus conséquent, plutôt que de définir des services un par un, Render met à disposition un système de ***blueprints*** pour gérer son ***Infrastructure as Code (IaC)***

Cela permet donc de centraliser la définition de tous ses services et de pouvoir ajuster à la volée leur configuration

De plus, l'utilisation de **Git** permet de **versionner** l'évolution de son infrastructure

# Blueprints

Pour gérer les différentes ressources, il suffit de définir un fichier ***render.yaml*** à la racine de son projet

```
1 services:
2   # A Docker web service
3   - type: web
4     name: webdis
5     runtime: docker
6     repo: https://github.com/render-examples/webdis.git # optional
7     region: oregon # optional (defaults to oregon)
8     plan: standard # optional (defaults to starter instance type)
9     branch: master # optional (defaults to master)
10    rootDir: webdis
11    dockerCommand: ./webdis.sh # optional (defaults to Dockerfile command)
12    numInstances: 3 # optional (defaults to 1)
13    healthCheckPath: /
```

# *Blueprints*

Il est possible de définir plusieurs services et bases de données et de lier chaque ressource les unes avec les autres

Les différents types de services disponibles, ainsi que la spécification complète du fichier ***render.yaml*** sont référencés dans cette page de la documentation

- <https://render.com/docs/blueprint-spec>

# *Blueprints*

**Supprimer une ressource** du fichier ne supprime pas la ressource sur Render, cela **évite** donc les suppressions **accidentelles**. Pour ajouter une nouvelle ressource, il suffit de la rajouter dans le fichier ***render.yaml***

Il est possible de supprimer le ***blueprint*** du côté de Render pour stopper la **synchronisation** tout en gardant les instances créées

# Object storage



# *Object storage*

Les **besoins** associés à ***l'object storage*** sont multiples

- stockage de fichiers pour son application
- distribution facile
- déchargement du serveur contenant l'intelligence
- choix de différentes localisations sources
- stockage illimité

# *Object storage*

Le ***file storage***, plus classique, a quant à lui plusieurs différences notables

- utilisation comme un disque dur externe en ligne
- limitation à certains types de fichiers suivant la plateforme
- présence d'un lecteur interne
- possibilité de filtrer plus ou moins les fichiers
- pas vraiment d'API pour téléverser / télécharger
- pas de stockage élastique / adaptatif

# AWS S3

S3 est un **service** d'AWS permettant de mettre en place des **compartiment de stockage** de fichiers

Ces compartiments sont **facturés au prorata** de leur utilisation, par rapport au **poids total** des fichiers et aux **requêtes** de lecture et de modification

C'est un des services les plus utilisés d'AWS et il sert de **standard** global pour les services du genre

Cependant, il est payant (sauf pour le partie ***free tier***)

# Wasabi

Wasabi est une **plateforme** permettant aussi la création d'un compartiment de **stockage** mais avec une **offre gratuite** sans carte bleue qui dure **30 jours**

De plus, il est possible de **s'interfacer** avec ses compartiments de la même façon qu'avec **S3** via un couple de **clefs HMAC** comprenant **clef d'accès & clef secrète**

- <https://wasabi.com/fr>

# Autres plateformes

Il existe également d'autres plateformes permettant de créer des **compartiments de type S3** avec une offre gratuite

- <https://tebi.io>
- <https://www.stackhero.io/en-fr/services/MinIO>

L'interface diffère mais les fonctionnalités sont bien souvent **similaires**, surtout pour la **génération de clefs**

# Intégration

Pour des compartiments de **type S3**, on va chercher à générer notre paire de **clefs HMAC**

En général la clef secrète n'est accessible **qu'une seule fois** pour des raisons de sécurité

Le compartiment doit normalement avoir un **nom unique** sur la plateforme pour **référencer les fichiers** correctement