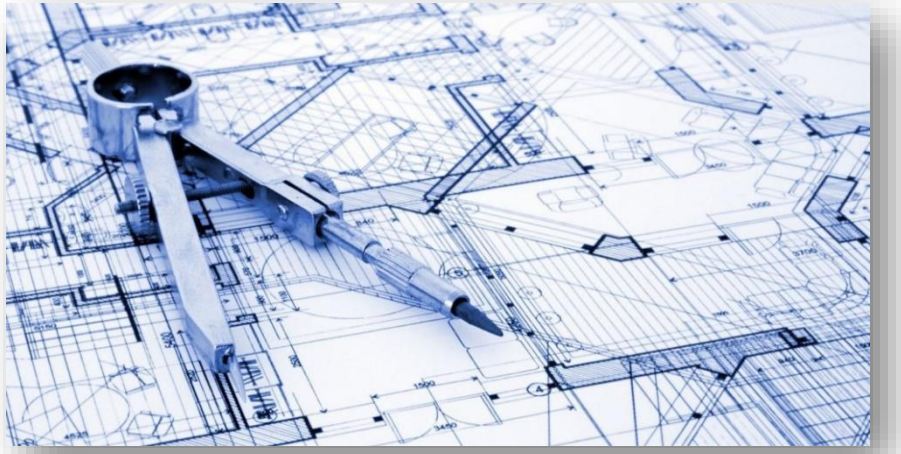
Architectures Logicielles

Concepts de base et solutions



Introduction

Les bases du développement logiciel

Introduction

« Building Software is not just Coding »

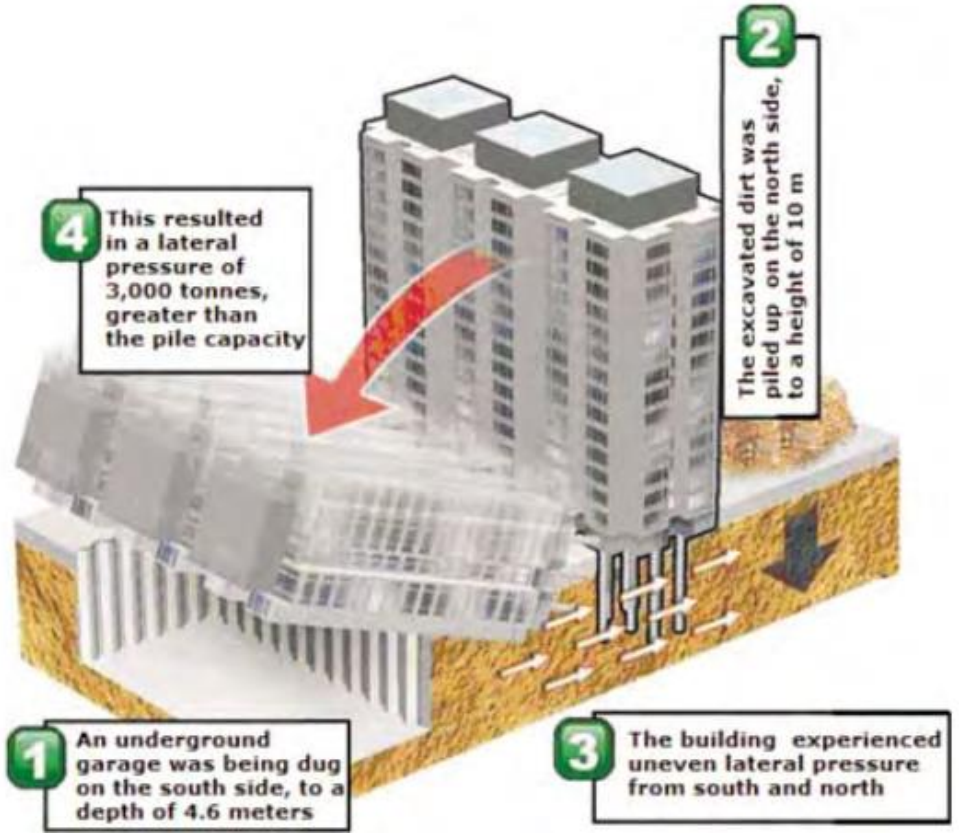
Introduction

« Building Software is not just Coding »

Introduction

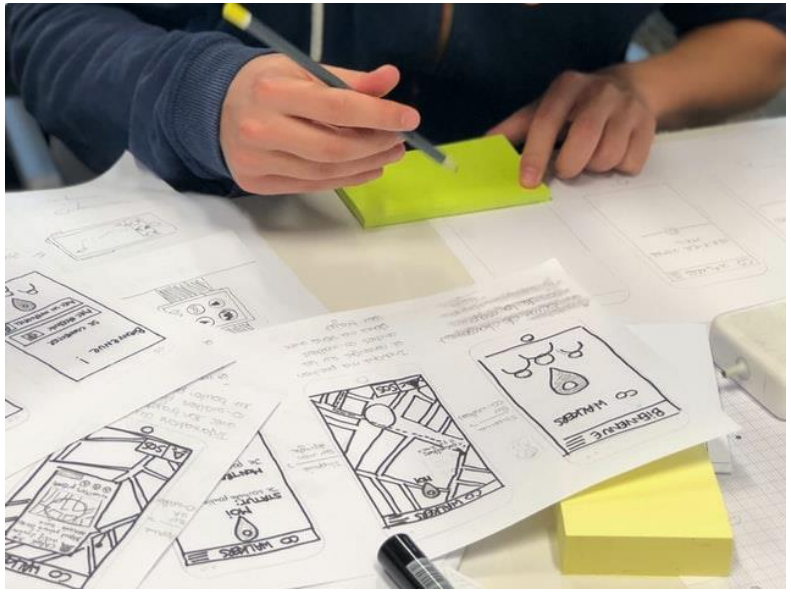
« Building Software is not just Coding »

Figure 2: Cause of failure, showing the condition at site



Introduction

« Building Software is not just Coding »

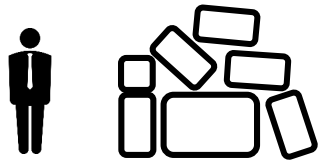


Rappel sur le développement logiciel

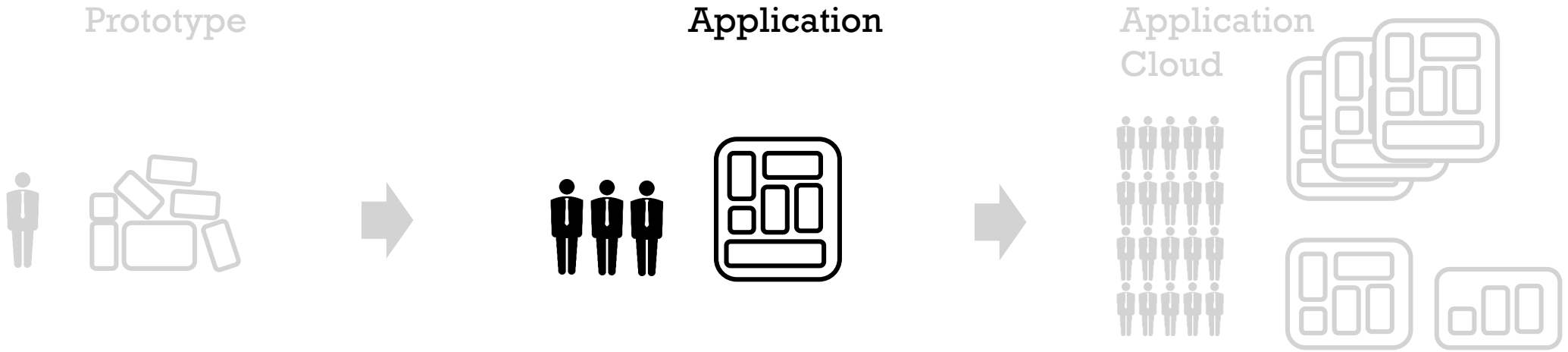
Prototype

Application

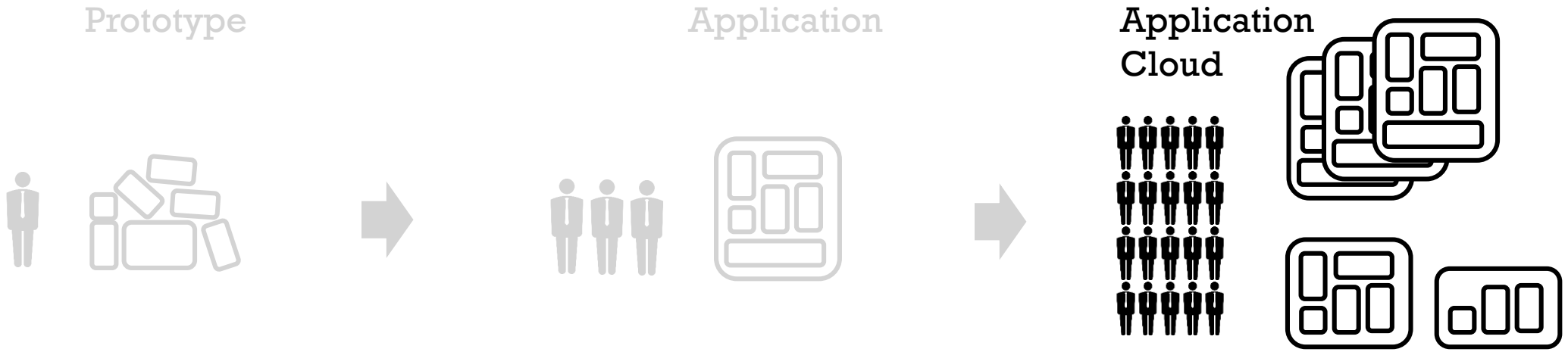
Application Cloud

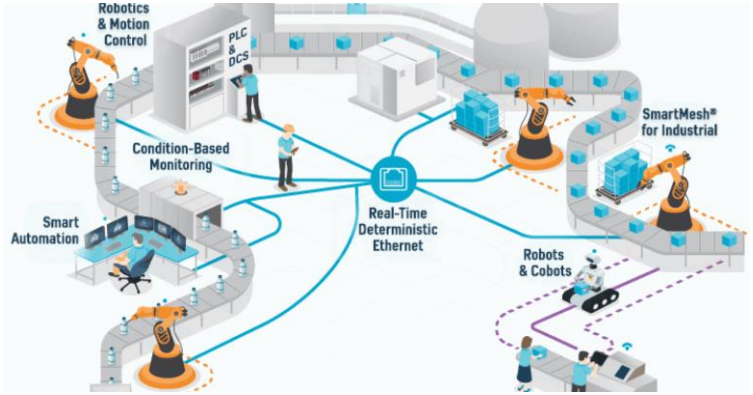


Rappel sur le développement logiciel

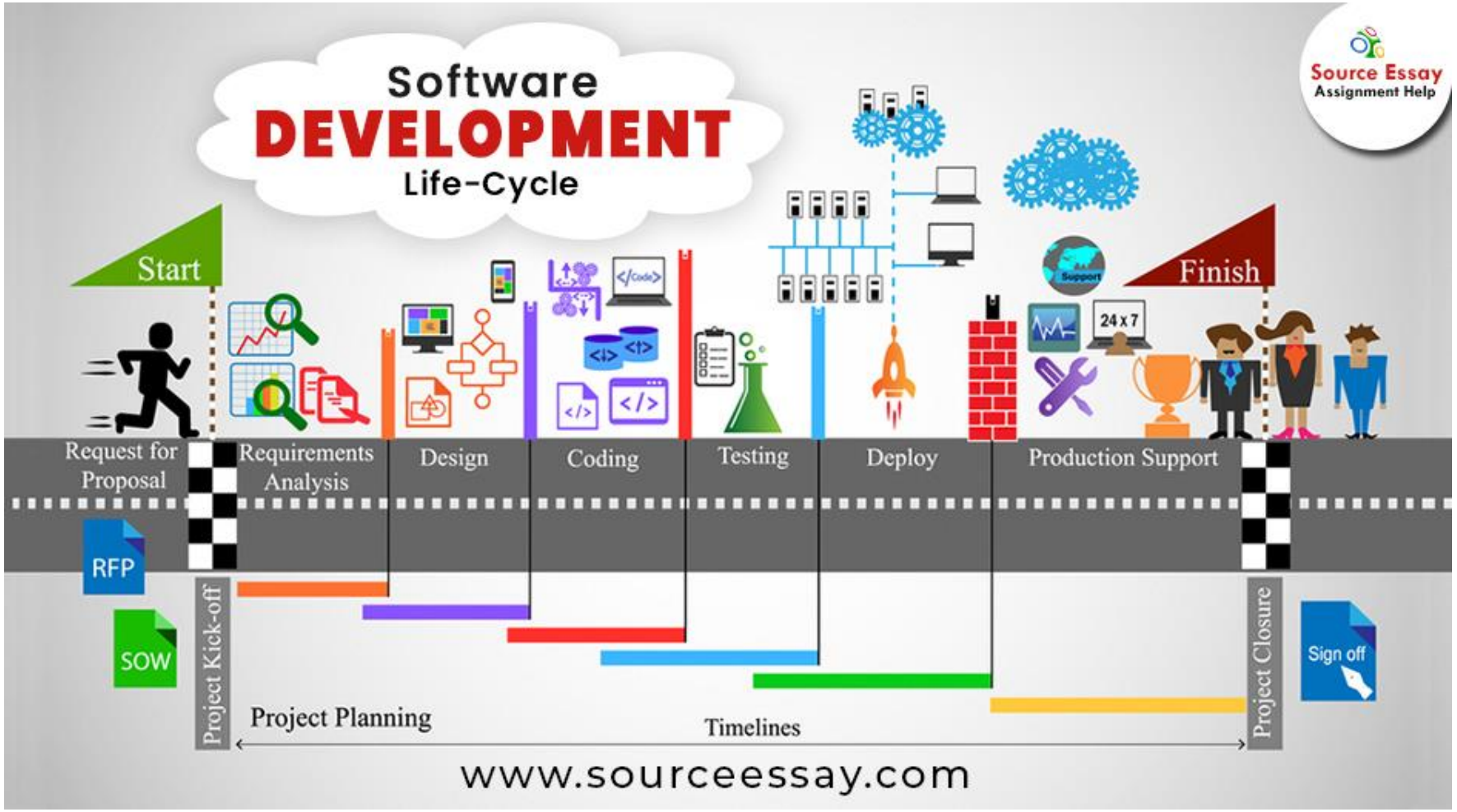


Rappel sur le développement logiciel

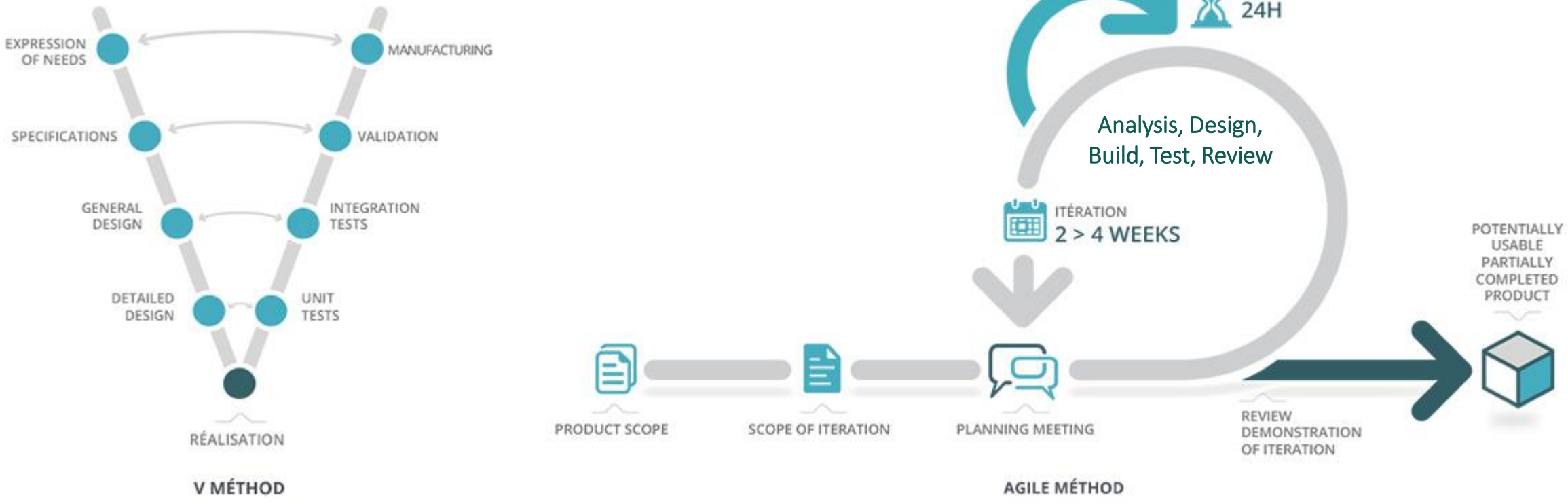




Software life cycle

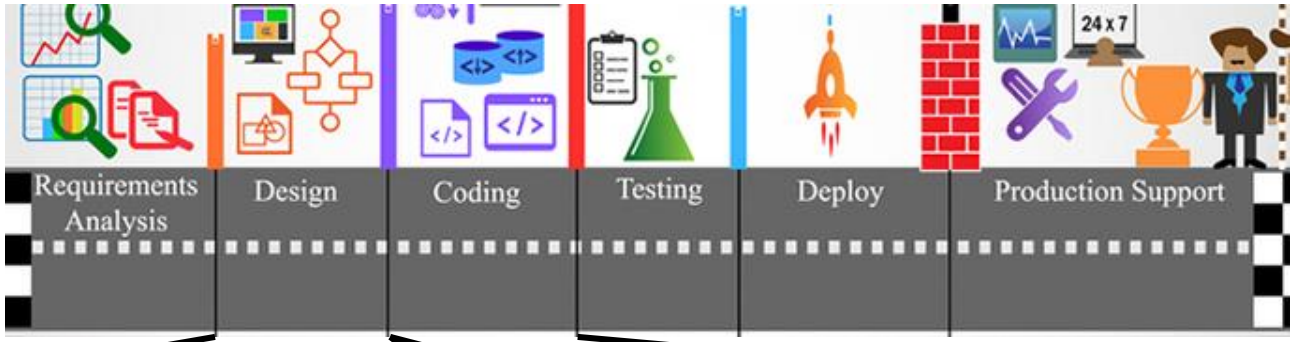


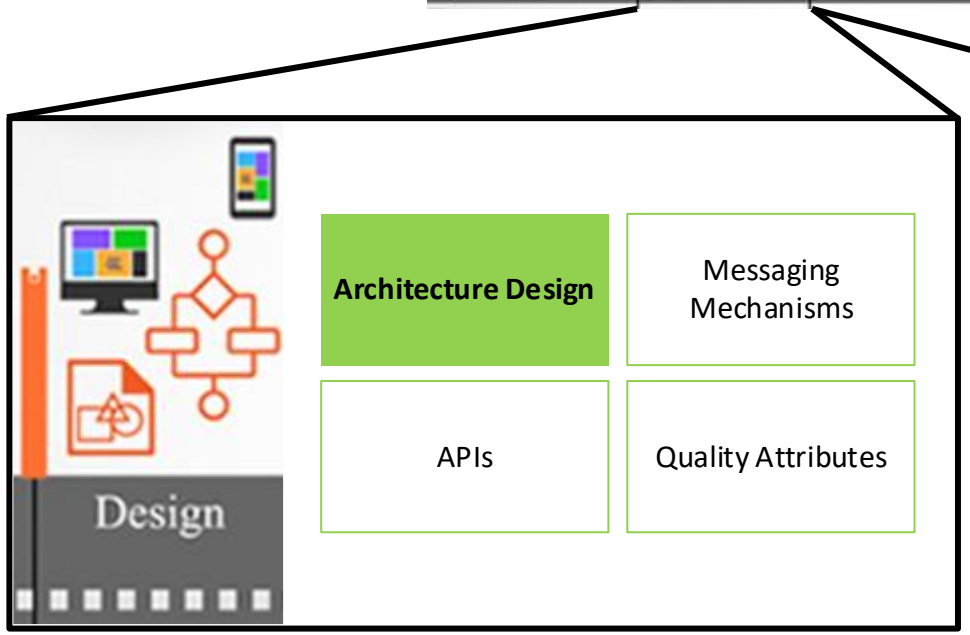
Software life cycle

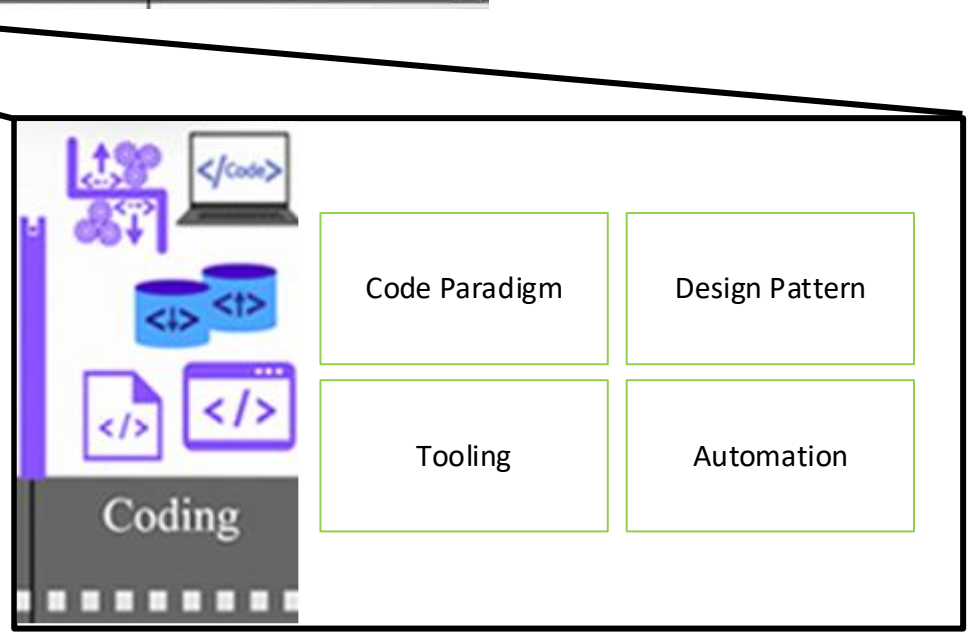


https://www.cegedim-insurance.com/en-EN/solutions-services/our-added-value/Pages/agile-methodology.aspx

Software life cycle



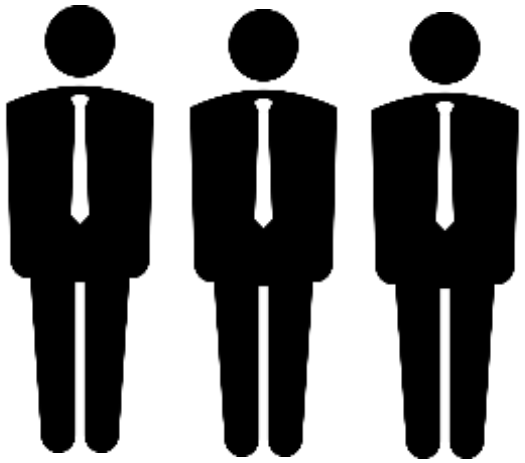


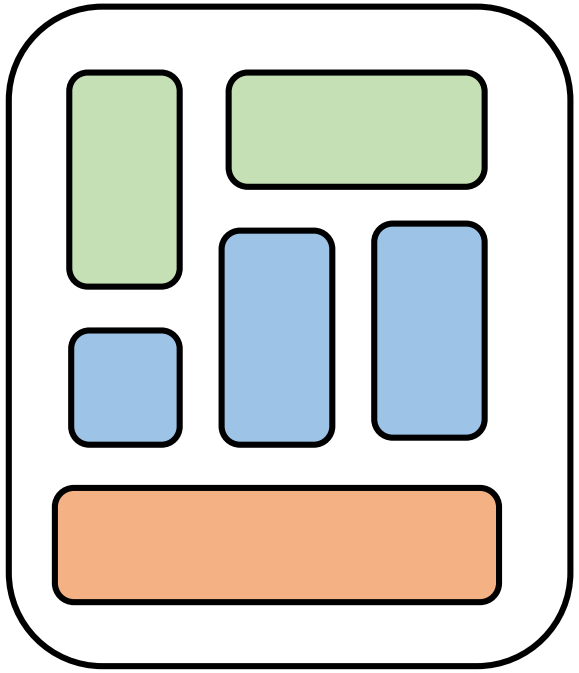


Architecture

Evolution des architectures logicielles

Application





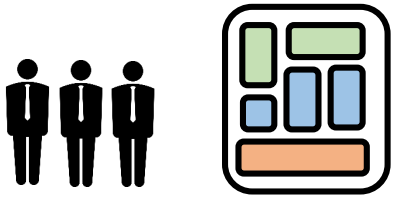
User Interface

Services Métier

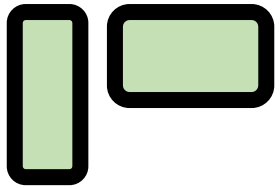
Persistance

Architecture logicielle

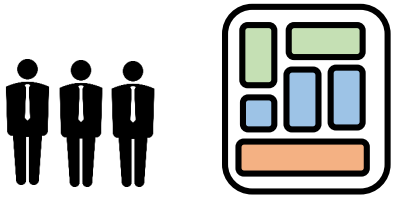
Application



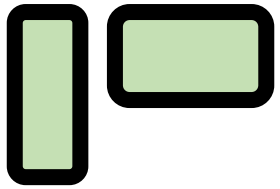
User Interface



Application

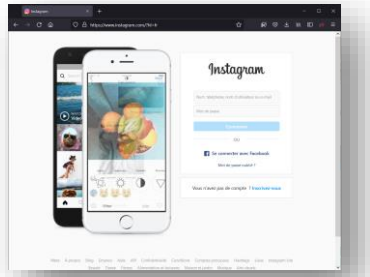


User Interface

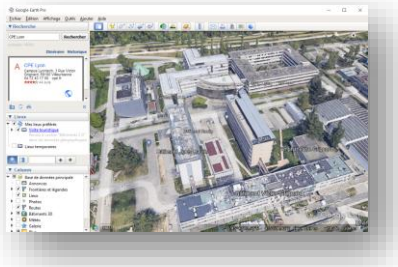


- Objectif:
  - Gestion interaction-utilisateur
  - Présentation des données
  - Traitements liés à la mise en forme des données

- Client Léger
  - Sobre en place et consommation de ressources
  - Un service tiers gère les outils de présentations de l'information
  - Interface standardiser facilitant le développement
  - Lié à une architecture client serveur (le serveur effectuant la plus part des calculs/traitements)

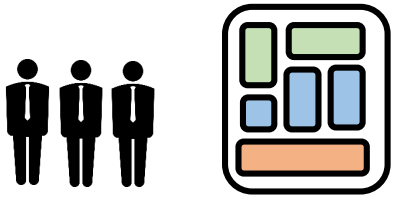


- Client Lourd
  - Autonome (pas nécessité d'avoir un serveur)
  - Peut utiliser toutes les ressources de la machine (carte graphique, interruptions)
  - Création plus complexe et couteuse

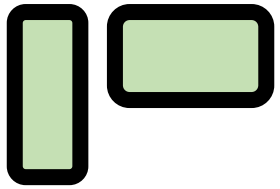


Architecture logicielle

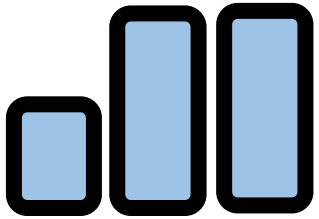
Application

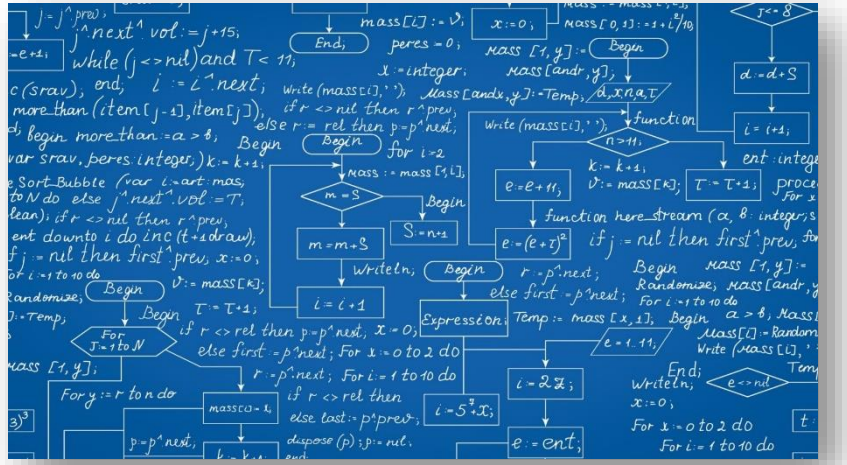


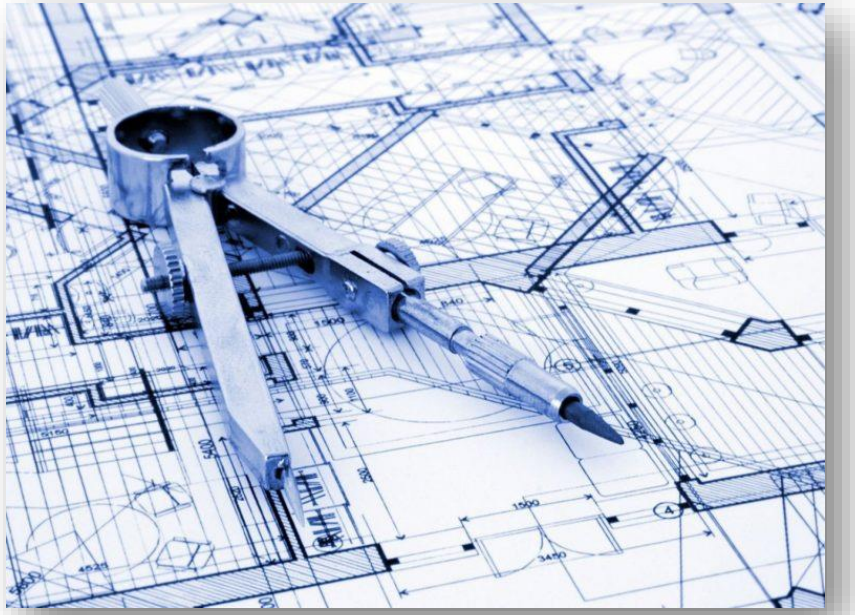
User Interface

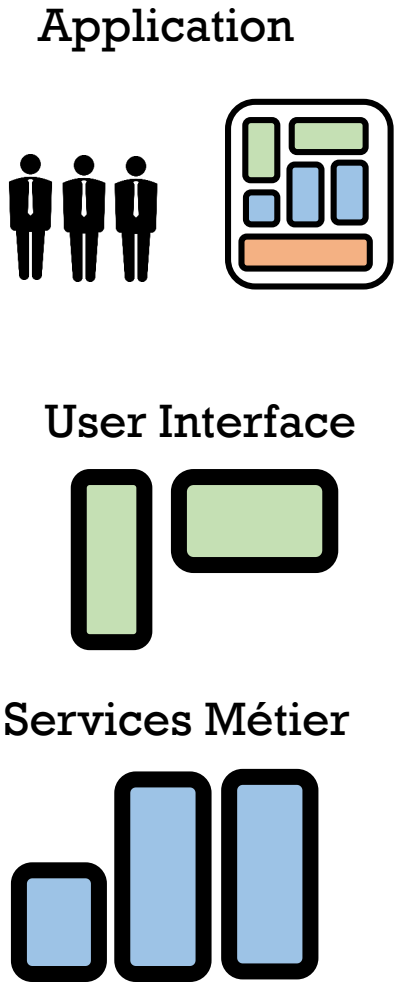


Services Métier





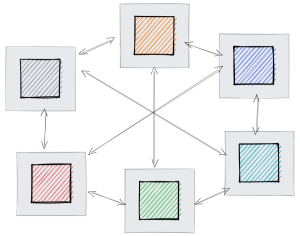




- Objectif:
  - Traitement des informations
  - Transformations des informations
  - Calculs / Algorithmes
  - Communication avec l'User Interface et la base de données

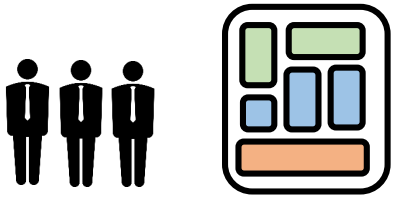
- Centralisée
  - Regroupe les tiers sur une seule et même machine
  - Accès aux ressources et communication immédiates

- Distribuée
  - Les tiers sont exécutés sur des machines différentes
  - Les communications entre les tiers sont indispensables
  - Les tiers peuvent être eux même sous-divisés

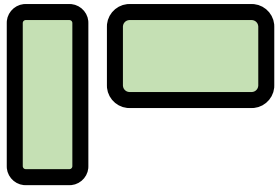


Architecture logicielle

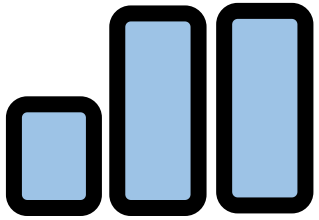
Application



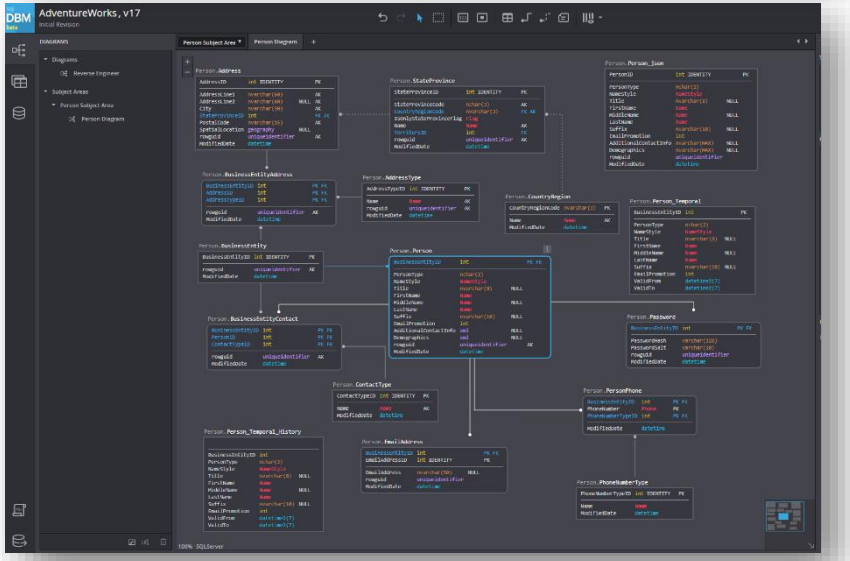
User Interface

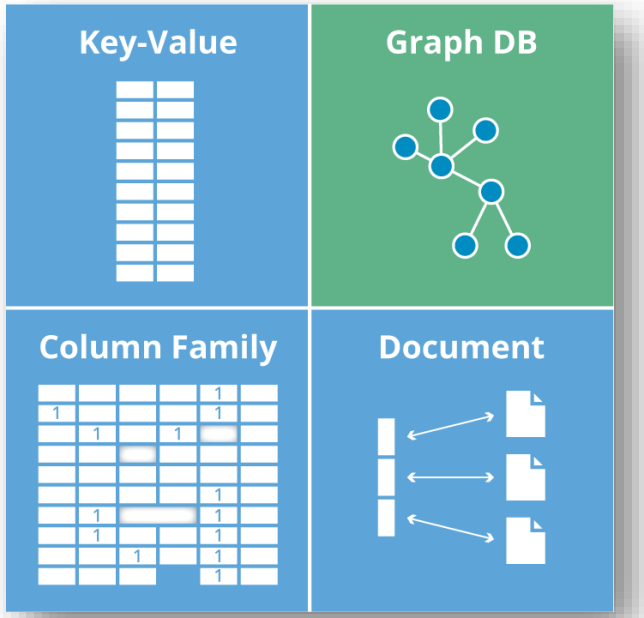


Services Métier

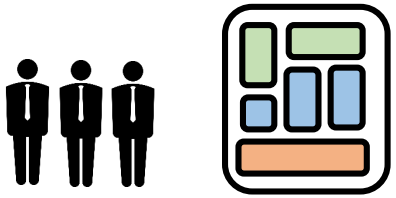


Persistance

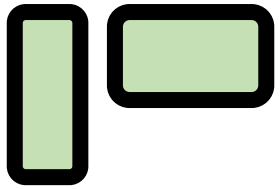




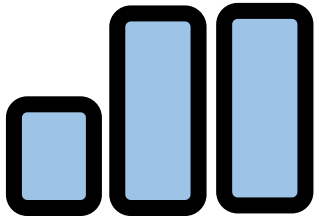
Application



User Interface



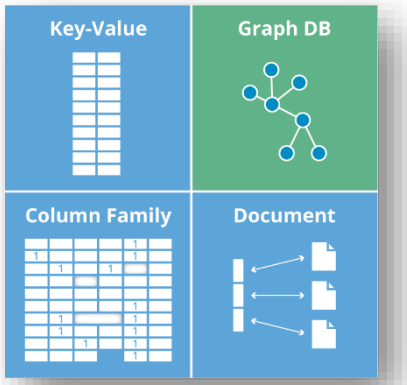
Services Métier



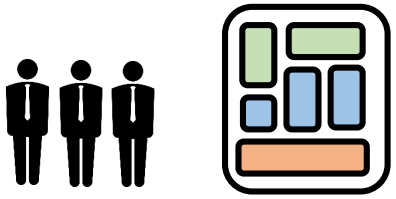
Persistance

- Objectif:
  - Stockage des données
  - Manipulation des données

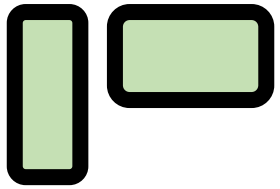
- Propriétés
  - Différents supports (Fichiers binaires/Xml/JSON, bd)
  - Langages spécifiques d'interaction (BD == langage SQL)
  - Outils de connections à distance (RDA,ODBC,JDBC)
    - Gestion des requêtes SQL
    - Gestion des connections à la BD
  - Différentes représentations des données à stocker
    - Dictionnaire
    - Document
    - Schéma relationnel
    - Graphe
    - …



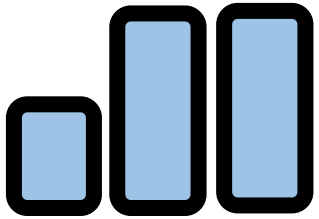
Application



User Interface

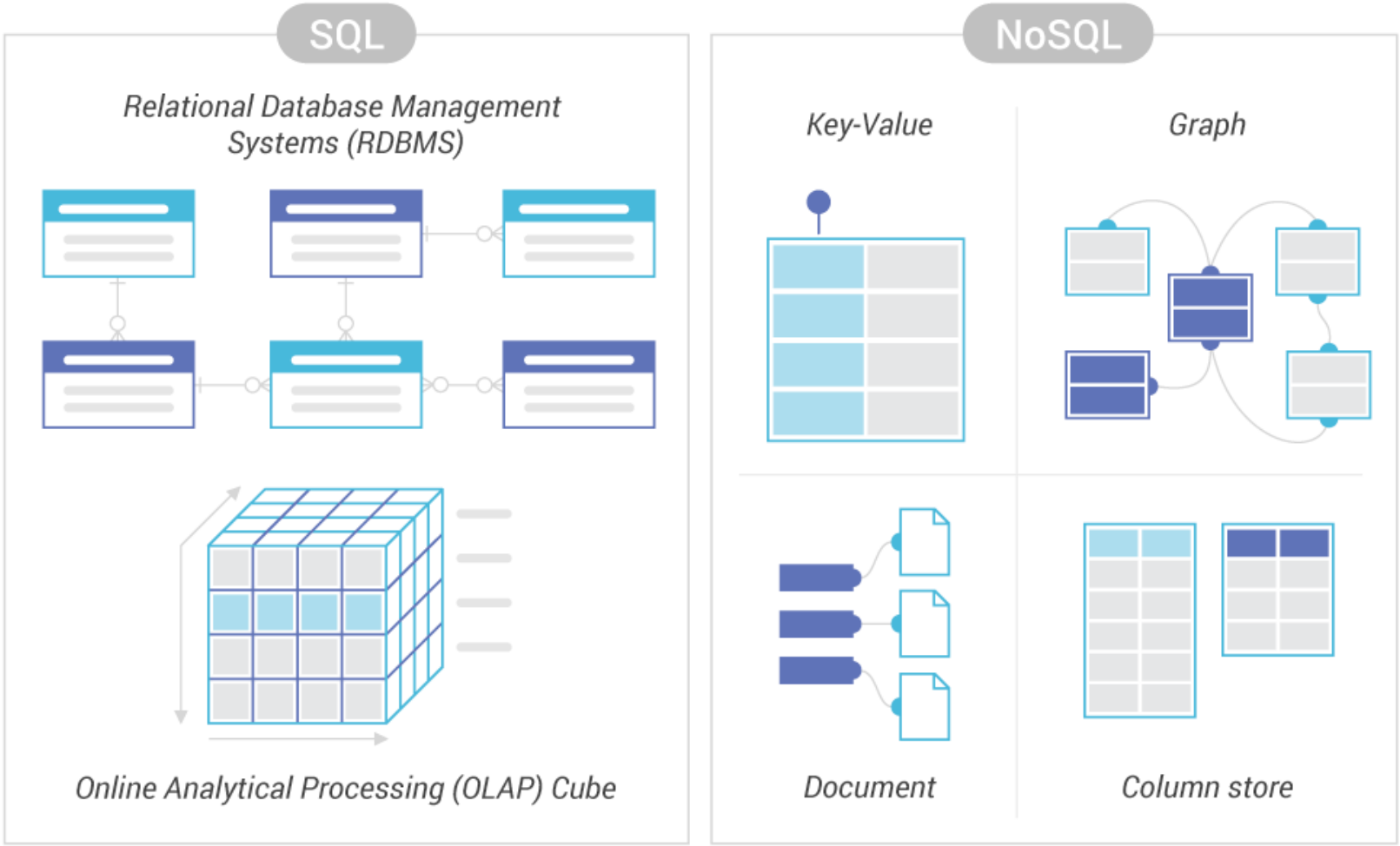


Services Métier



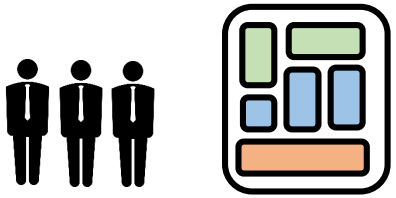
Persistance

Base de données: SQL VS NOSQL

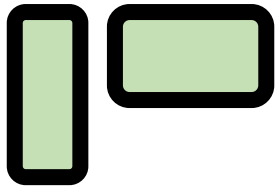


https://altitudetvm.com/fr/komputer/1556-apa-itu-nosql-berikut-ini-pengertian-nosql-beserta-kelebihannya.html

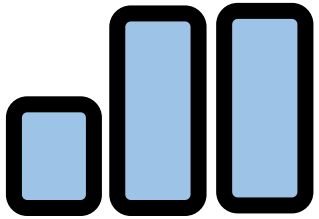
Application



User Interface



Services Métier



Persistance

Base de données: SQL VS NOSQL

	SQL	NoSQL
Définition	**Relationnelle**	**Non relationnelle ou distribué**
Utilisation	Requête pour analyser et récupérer données	Adapté pour une variété de technologie d'application moderne comme les WebApp
Langage de requête	SQL	Plusieurs langage pour les différents bords
Type	Tableau	Document / Graphes / Clés-Valeurs
Schéma	Fixe et prédéfini	Dynamique
Exemple de DMS (Data Management System)	Oracle, PostGres, MySql	MongoDB, Neo4J
Adapté pour	Requêtes complexes et intensives	Grande base de données, Big Data
Année de création	Années 70	Années 2000
Open Source	Mélange open source (PostGres, MySql) et commerciaux (Oracle)	Majorité d'open source
Avantages	Stockage optimisé et stabilité	Facilité et flexibilité du stockage
Inconvénients	Rigidité et besoin d'une certaine expertise	Parfois trop permissif

https://datascientest.com/sql-vs-nosql-differences-utilisations-avantages-et-inconvenients

Build your App!

- Proposer une application à développer !!
- Découper votre application suivant les tiers présentés

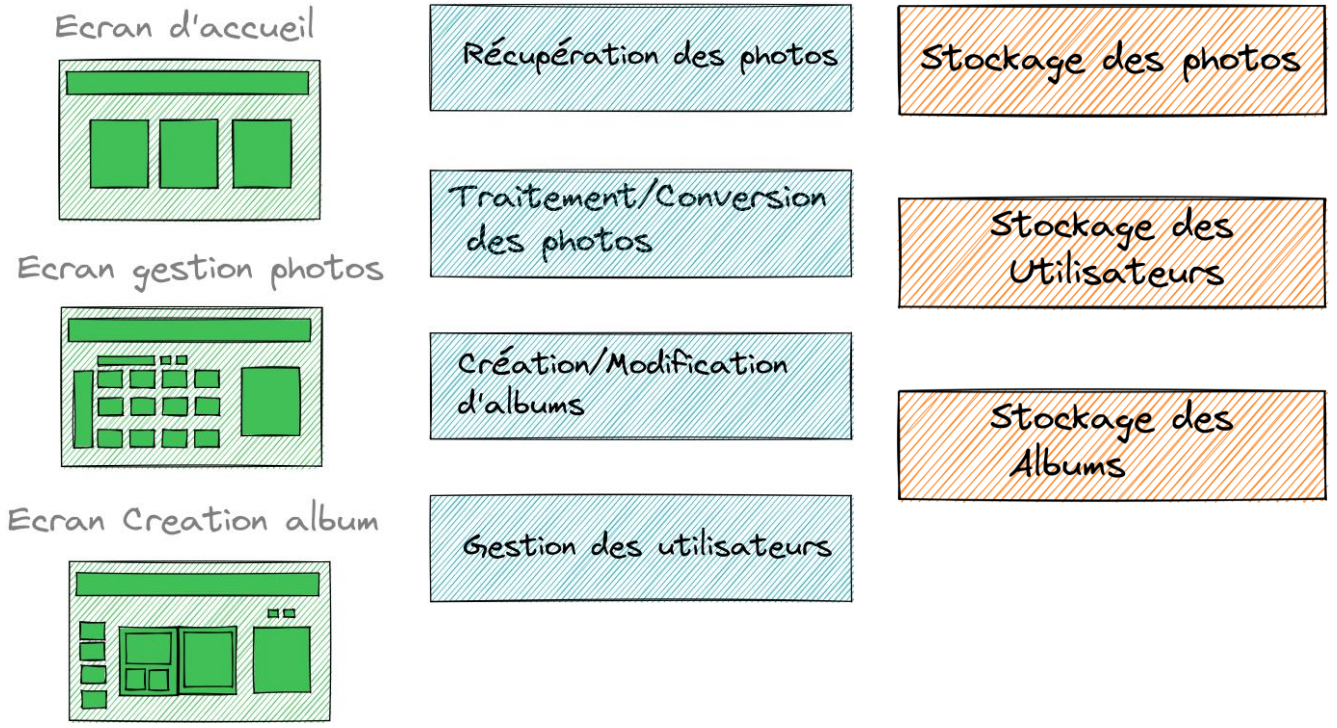
Build your App!

- Proposer une application à développer
- E.g
  - Création d'album de photo de chat
    - Récupération de Photos
    - Modification de Photos
    - Création d'album + Text

© Stocklib / andreykuzmin

Build your App!

- Proposer une application à développer
- E.g
  - Création d'album de photos de chat
    - Récupération de photos
    - Modification de photos
    - Création d'albums + Text



Architecture – organisation des tiers

- Modèle Centralisé (1 tier)
  - Tous les tiers sont sur la même machine
  - Accès local aux ressources nécessaires (Données, Code, périphérique, mémoire)

Architecture – organisation des tiers

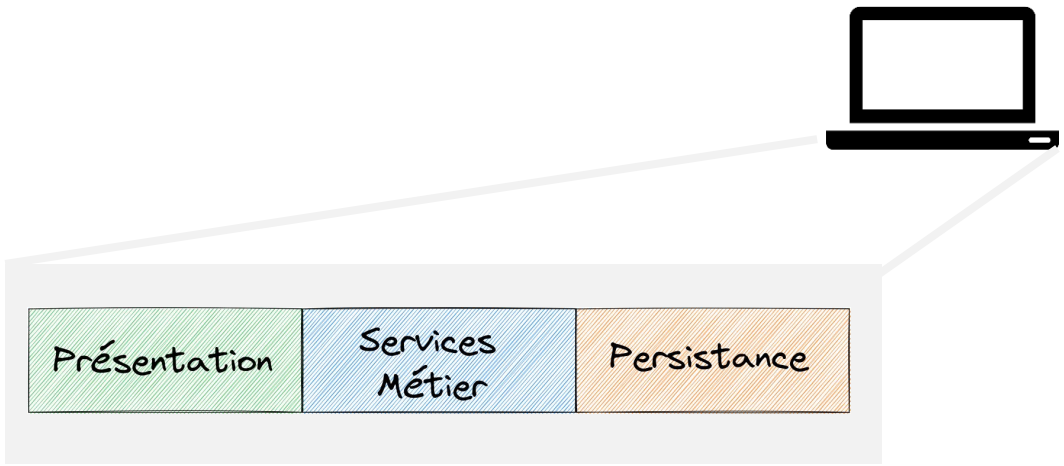
❑ Modèle Centralisé (1 tier)

Avantages ✓

- Partage de données simple
- UI très réactive
- Usage des ressources locales

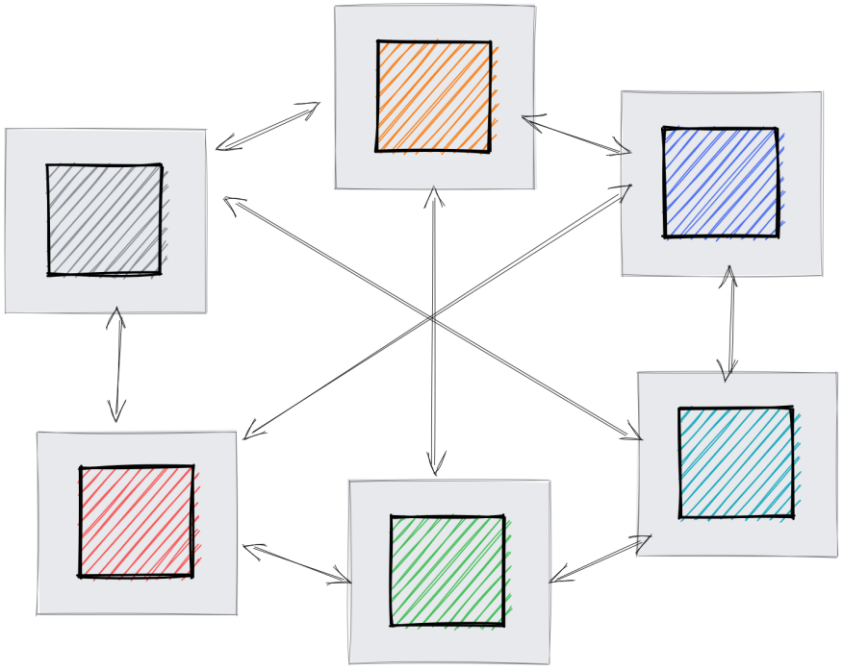
Inconvénient ⊘

- Distribution complexe
- Passage à l'échelle (inexistant)
- Mise à jour complexe
- Fiabilité (très limité)
- Sécurité (complexe)



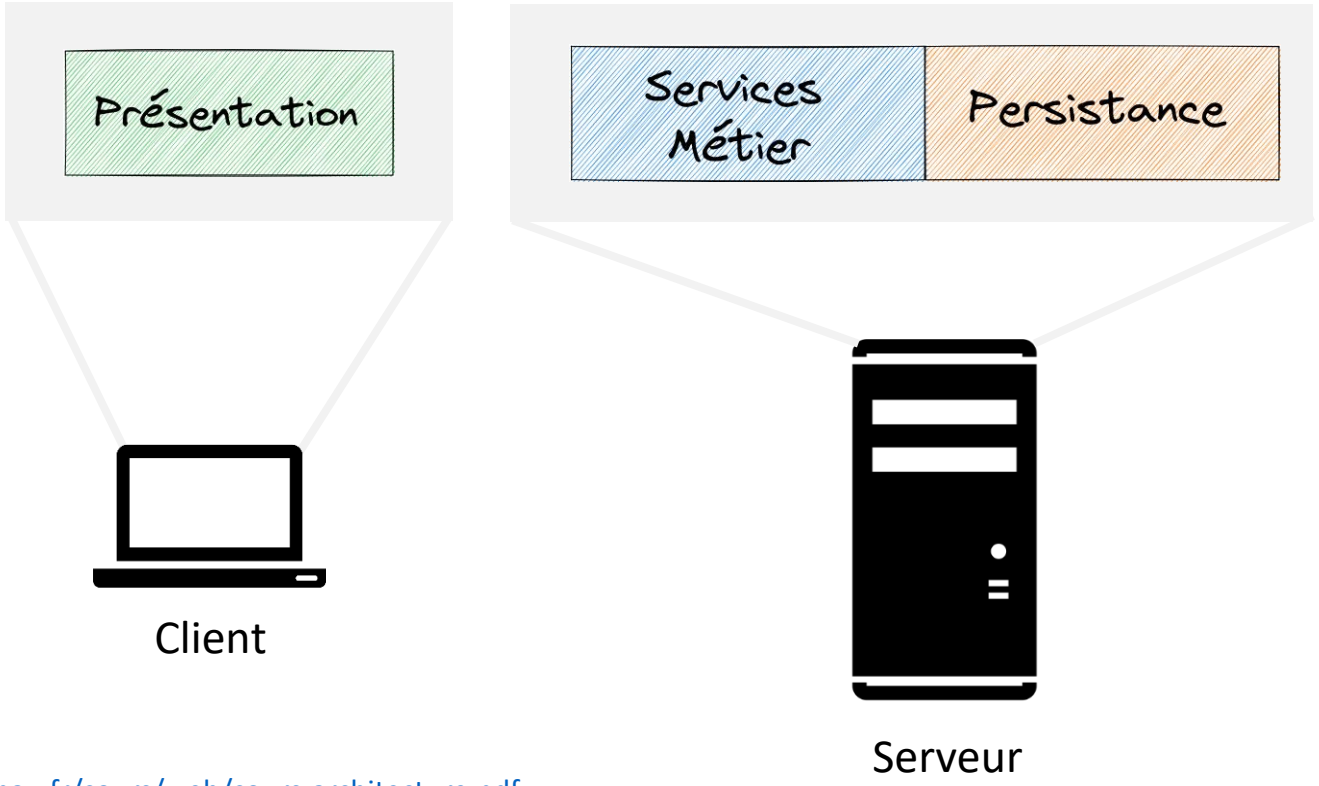
Architecture – organisation des tiers

- Modèle Décentralisé (multi-tiers)
  - Tiers distribués sur différents ordinateurs
  - Architecture invisible pour l'utilisateur (notion de transparence)
  - Nécessité de partager les ressources distantes
  - **Protocol de communication** nécessaire
  - **Identification des ressources** nécessaires
  - **Hétérogénéité des systèmes** et des langages
  - Sécurisation plus complexe
  - Différents modèles d'interaction possibles (Client – serveur, diffusion de messages, mémoire partagée, pair à pair)



Architecture – organisation des tiers

- Modèle Décentralisé
  - 2 Tiers : Client Server
    - Client présentation des données uniquement
    - Serveur
      - Services métiers
      - Persistance



https://ecariou.perso.univ-pau.fr/cours/web/cours-architecture.pdf

Systèmes Distribués, Architecture N-Tiers, Eric Cariou Universtité de Pau et des Pays de l'Adour, Département Informatique

Architecture – organisation des tiers

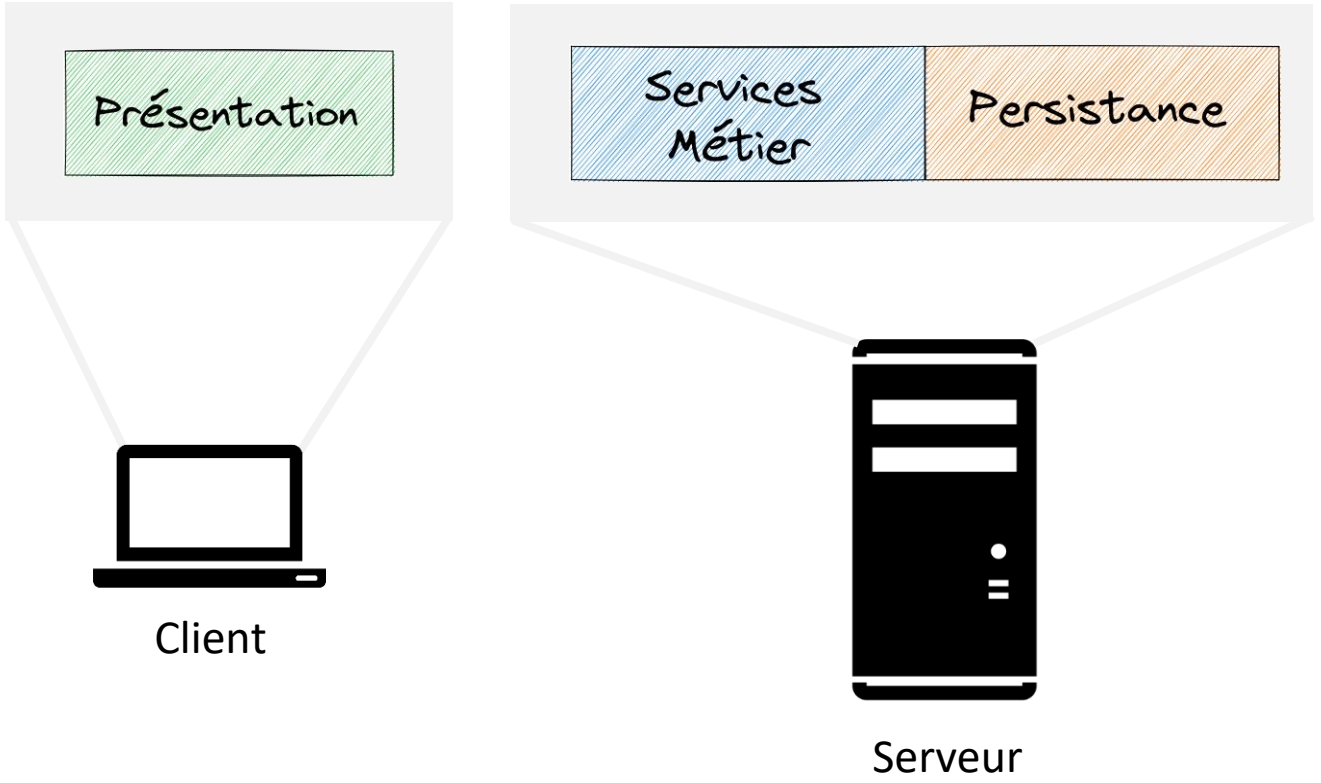
❑ 2 Tiers: Client-Serveur

Avantages

- Distribution plus simple
- Mise à jour plus simple
- Sécurité(limité)

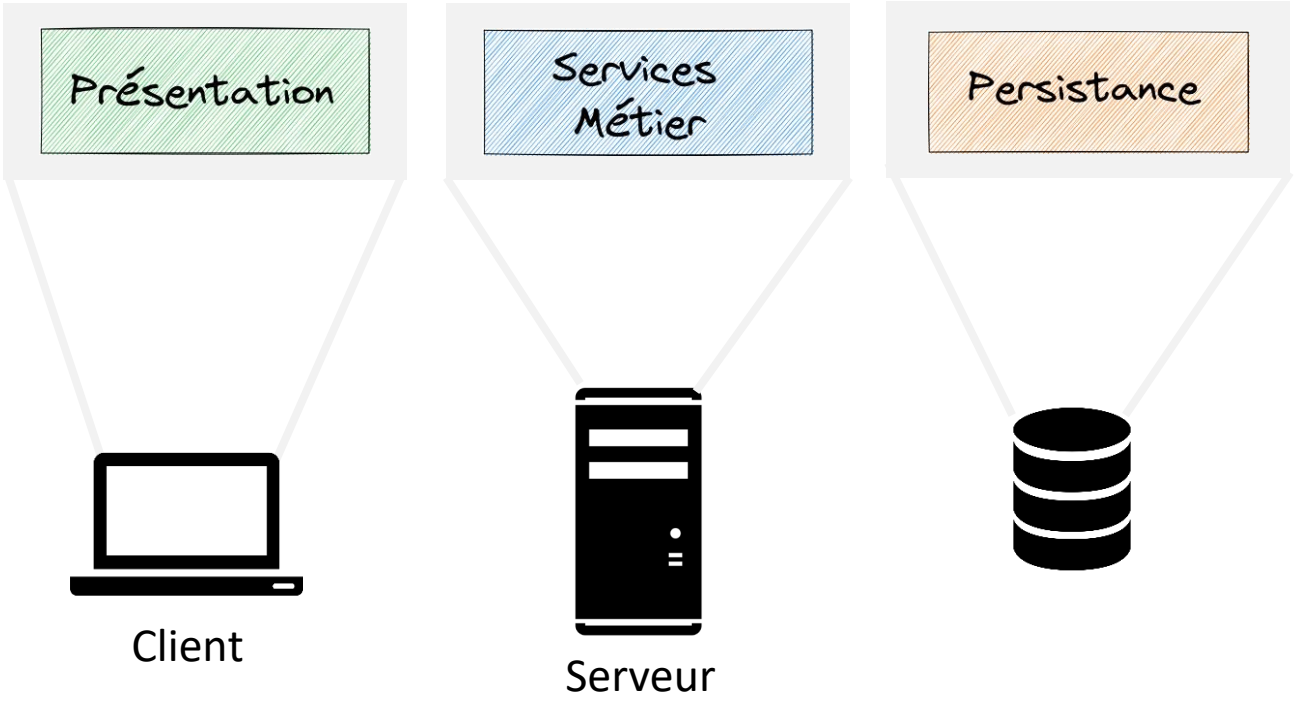
Inconvénient

- Fiabilité (1 serveur unique)
- Passage à l'échelle (limité)
- Gestion des communications nécessaires
- Usage des ressources locales limitées



Architecture – organisation des tiers

- Modèle Décentralisé
  - 3 Tiers :
    - Architecture la plus répandue
    - Séparation persistance/services métiers
    - Meilleur passage à l'échelle et fiabilité
      - Services métiers
      - Persistance



https://ecariou.perso.univ-pau.fr/cours/web/cours-architecture.pdf
Systèmes Distribués, Architecture N-Tiers, Eric Cariou Universtité de Pau et des Pays de l'Adour, Département Informatique

Architecture – organisation des tiers

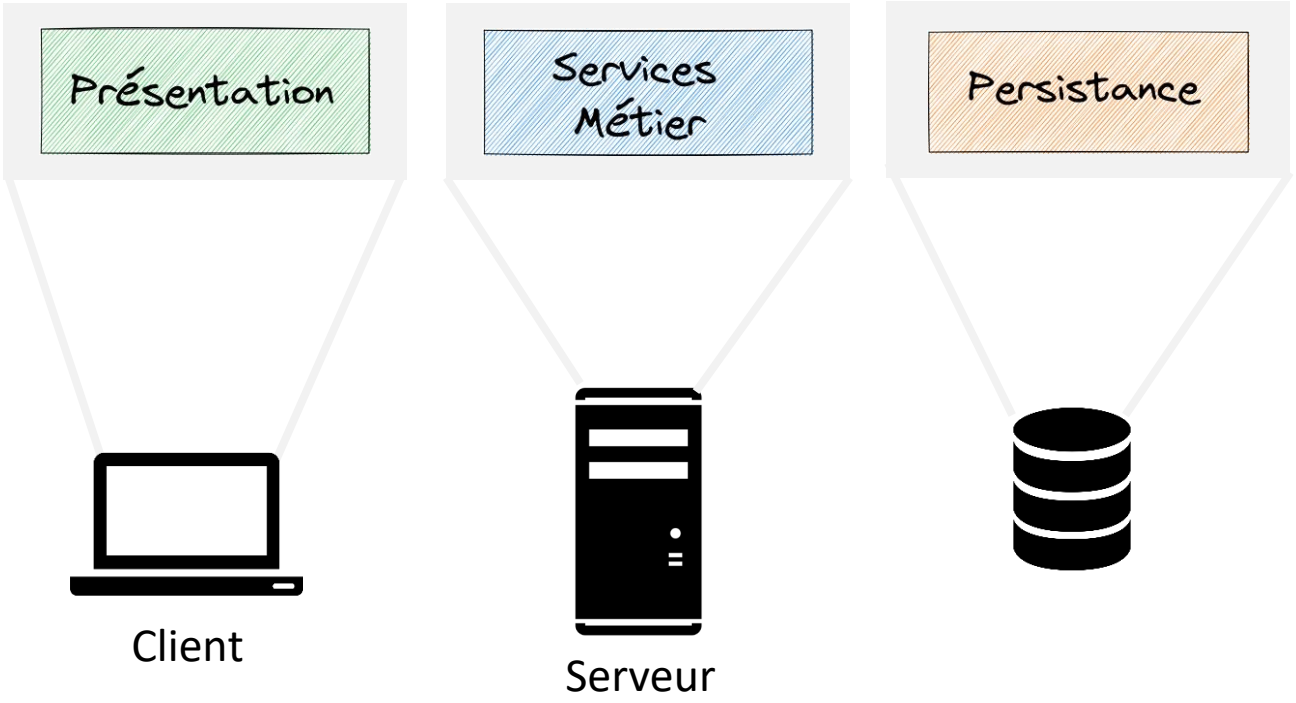
❑ 3 Tiers

Avantages

- Distribution plus simple
- Mise à jour plus simple
- Sécurité (étendue)
- Passage à l'échelle (simple)
- Fiabilité (plus simple)

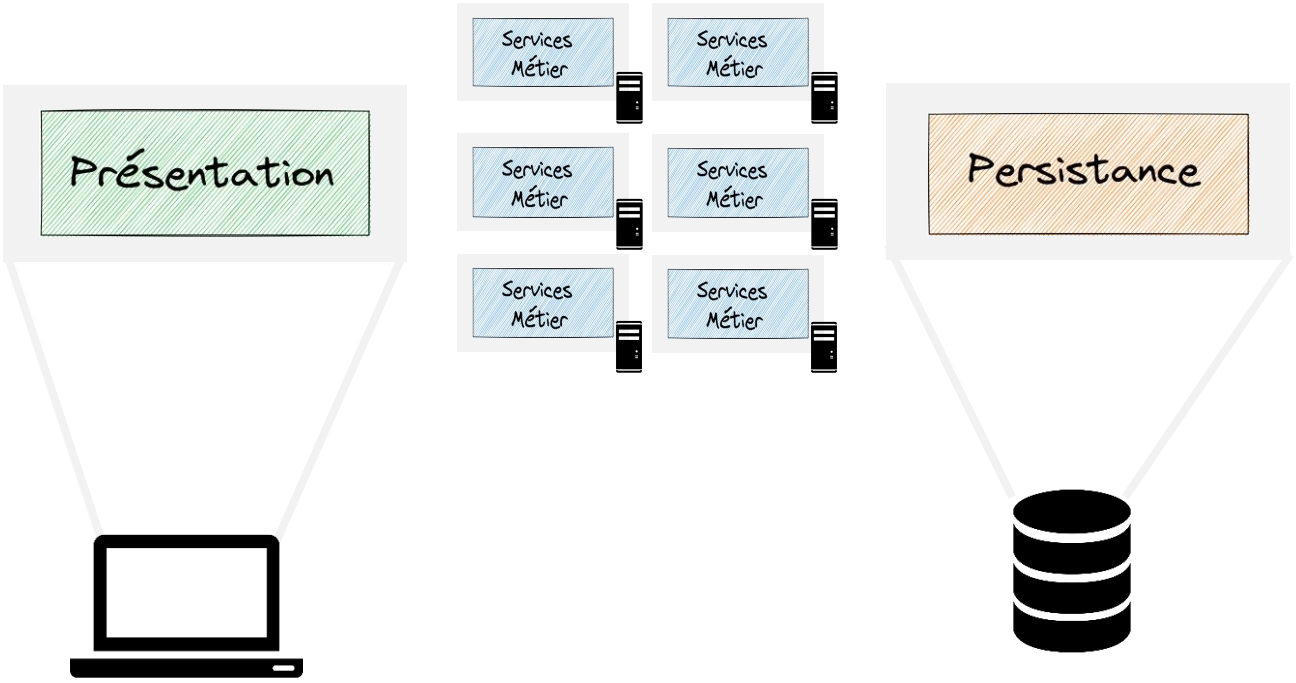
Inconvénient

- Gestion des communications nécessaires
- Usage des ressources locales limitées
- Plus complexe



Architecture – organisation des tiers

- Modèle Décentralisé
  - N Tiers :
    - Ajout de couches supplémentaires (la plupart du temps séparation des services métiers)
    - Découpage de la couche Services Métier (≠ monolithique)
      - Composition horizontale
        - Interaction entre plusieurs services métiers
      - Composition verticale
        - Les services métiers utilisent aussi des services techniques (transaction, sécurité..)



https://ecariou.perso.univ-pau.fr/cours/web/cours-architecture.pdf

Systèmes Distribués, Architecture N-Tiers, Eric Cariou Universtité de Pau et des Pays de l'Adour, Département Informatique

Architecture – organisation des tiers

❑ N Tiers

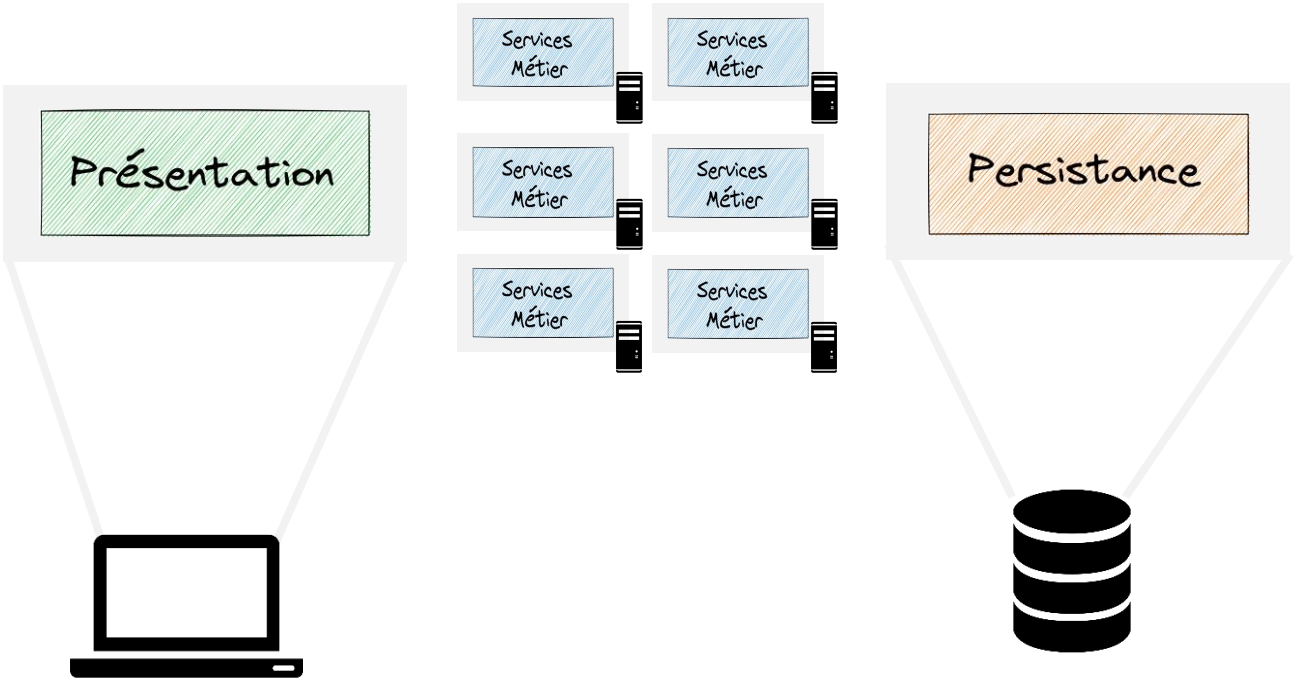
Avantages ✓

- Distribution plus simple
- Mise à jour plus simple
- Passage à l'échelle (étendue)

- Fiabilité (étendue)
- Hétérogénéité plus simple
- Gestion des couts (flexibilité)

Inconvénient ⊘

- Gestion des communications nécessaires (complexe)
- Usage des ressources locales limitées
- Plus complexe
- Sécurité (complexe)
- Découverte de Services

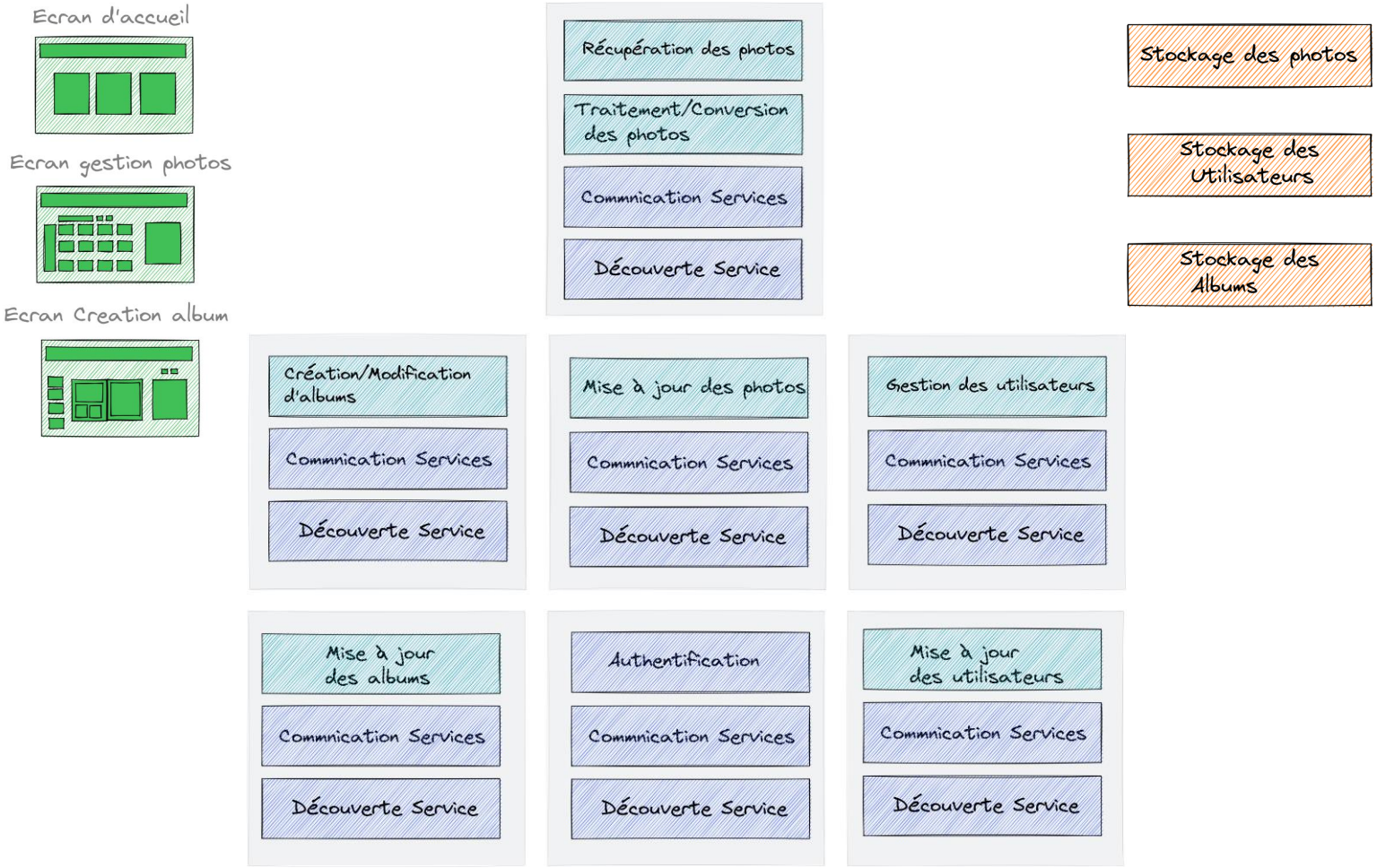


Build your App!

- Proposer un découpage de votre application en architecture N-Tiers

Build your App!

- E.g
  - Création d'album de photos de chat
    - Récupération de photos
    - Modification de photos
    - Création d'albums + Text



Architecture Centrale vs Distribuée

☑ Intérêt des systèmes distribués

- Utiliser et partager des ressources distantes
- Optimiser l'utilisation des ressources disponibles
- Système plus robuste
- Passage à l'échelle plus simple

⛔ Inconvénients des systèmes distribués

- Communication élément critique (si réseau lache, application indisponible)
- Souvent présence d'un élément central (e.g auth, e.g service discovery)
  - Si élément tombe application indisponible
  - Goulot d'étranglement
- Gestion systèmes totalement décentralisés complexe
- Découpages des algorithmes et des applications complexes

https://ecariou.perso.univ-pau.fr/cours/cs-umbb/cours-intro.pdf
Systèmes Distribués, Introduction, Eric Cariou Universtité de Pau et des Pays de l'Adour, Département Informatique

Architecture N Tiers Communication

- Communication par protocole réseau bas niveau (TCP/UDP)
- Communication par protocole de plus haut niveau (HTTP, SOAP)
- Usage de middleware possible (gestion de l'hétérogénéité, services supplémentaires, annuaires, persistance, sécurité)
- Différents Modèles d'interaction:
  - Client/Serveur : Client demande la ressource, le serveurs répond (pull) interaction de type (1-1)
  - Diffusion de message: Emetteur envoie des message à un ou plusieurs récepteur (Broadcast, Multicast) pull et push possible (interaction de type 1-N)
  - Mémoire partagée: pas de lien direct entre les entités interaction uniquement avec la mémoire partagée
  - Pair à pair: pas de distinction entre les participants, pour partager des données, effectuer un calcul commun (algorithme de consensus)

https://ecariou.perso.univ-pau.fr/cours/cs-umbb/cours-intro.pdf
Systèmes Distribués, Introduction, Eric Cariou Universtité de Pau et des Pays de l'Adour, Département Informatique

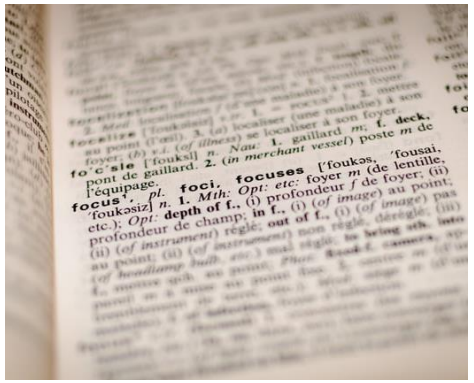
Architectures et cloud computing

Introduction cloud computing, exemple d'applications cloud ready

Cloud Computing

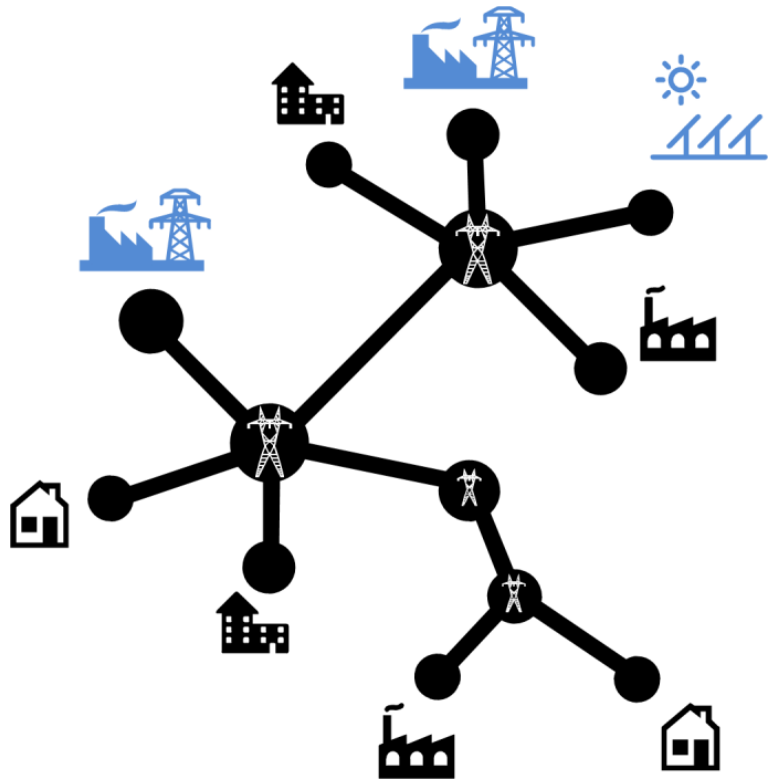
- Un terme Marketing, pour parler d'une évolution plus que d'une révolution

- Principes de bases :
  - On utilise un service
  - L'infrastructure sous jacente est mutualisée
  - On paye en fonction de ce qu'on utilise réellement



Cloud Computing

- Les équivalents hors informatique
  - L'électricité
  - L'eau
- Je paye ce que j'utilise réellement
- L'infrastructure est mutualisée



Cloud Computing

« …le cloud computing est la fourniture de services informatiques (notamment des serveurs, du stockage, des bases de données, la gestion réseau, des logiciels, des outils d'analyse, l'intelligence artificielle) via Internet (le cloud) dans le but d'offrir une innovation plus rapide, des ressources flexibles et des économies d'échelle. En règle générale, vous payez uniquement les services cloud que vous utilisez (réduisant ainsi vos coûts d'exploitation), gérez votre infrastructure plus efficacement et adaptez l'échelle des services en fonction des besoins de votre entreprise. »

Microsoft Azure https://azure.microsoft.com/fr-fr/overview/what-is-cloud-computing/

Les promesses du cloud

Coût : paiement à la consommation

Mise à l'échelle : Augmentation des ressources dynamique en fonction du besoin

Performances: Matériel et infrastructure optimisés (vaste gamme de ressources performantes)

Sécurité: Beaucoup d'outil mis à disposition, leur mise en œuvre reste complexe

Productivité : Plus de gestion interne de salle serveur de montée de version de logicielles et de configuration

Fiabilité : infrastructure mutualisée et spécialisée mettant en œuvre des systèmes permettant des garanties de fonctionnement élevée (Redondance, PLA, duplication,…)

Inspiration Microsoft Azure https://azure.microsoft.com/fr-fr/overview/what-is-cloud-computing/

Les services du cloud

IAAS

Infrastructure As A Services

Location d'infrastructure
(serveur, vm, stockage, réseaux…)

PAAS

Platform As A Services

Outils de développement facilitant la création d'application cloud
(Scalable, services technique à la demande, auto-gestion configuration)

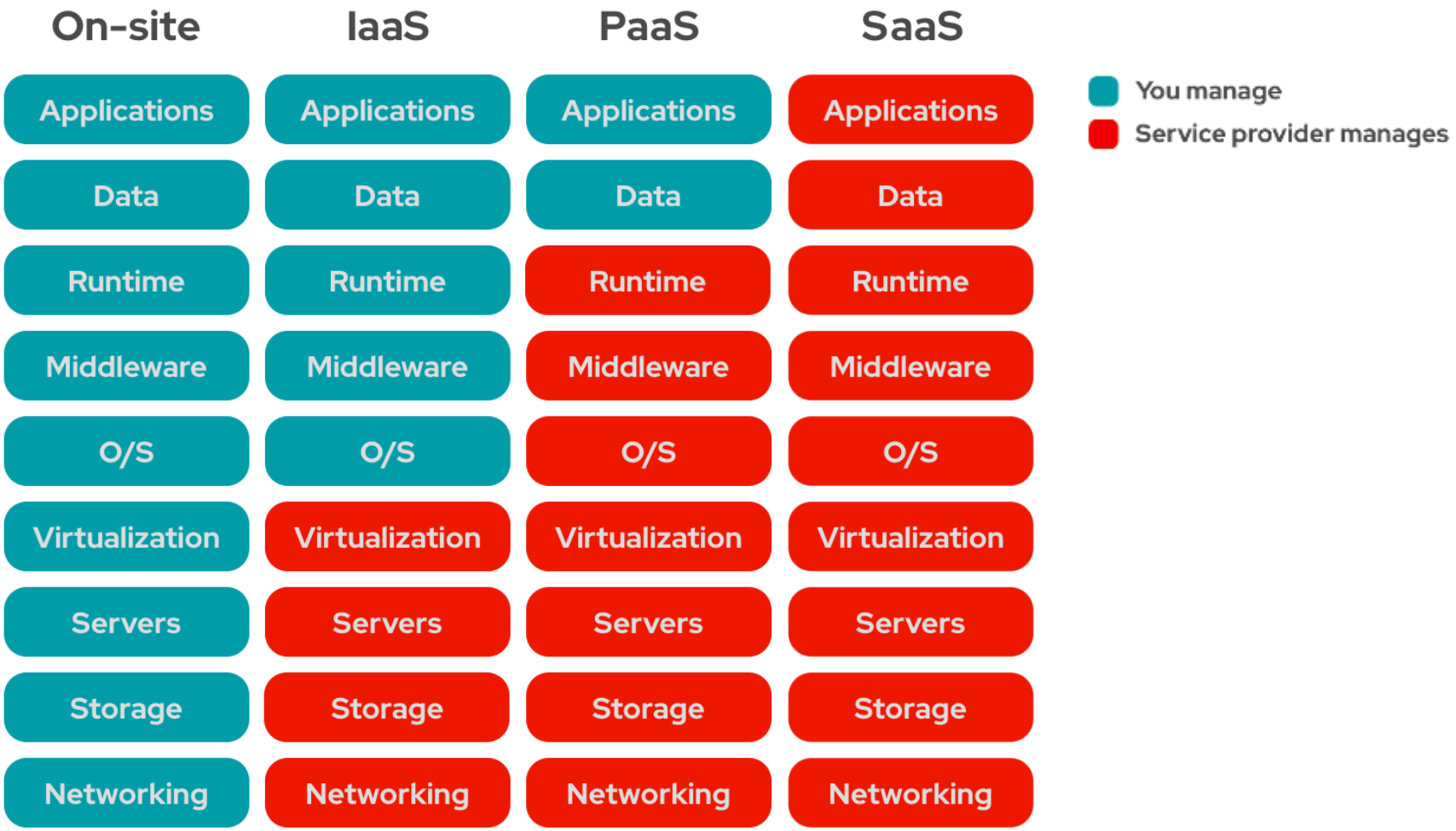
SAAS

Software As A Services

Crédit: www.applixia.com

Méthode de diffusion d'application
(Scalable, par abonnement, premium,…)

Les Services du cloud

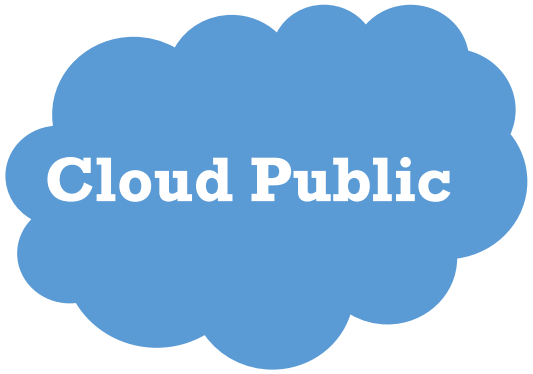


Red Hat : https://www.redhat.com/fr/topics/cloud-computing/iaas-vs-paas-vs-saas

Différents Types de Cloud

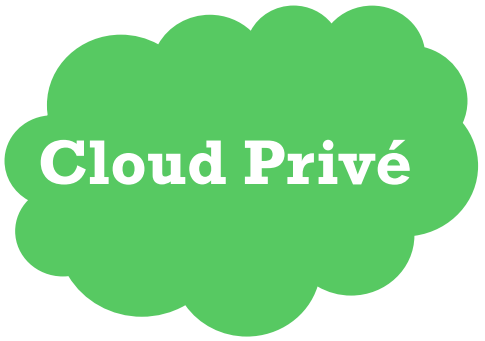
Public

- Exécution chez un prestataire public
- google, amazon, azure
- Public car accessible en théorie par tout le monde …

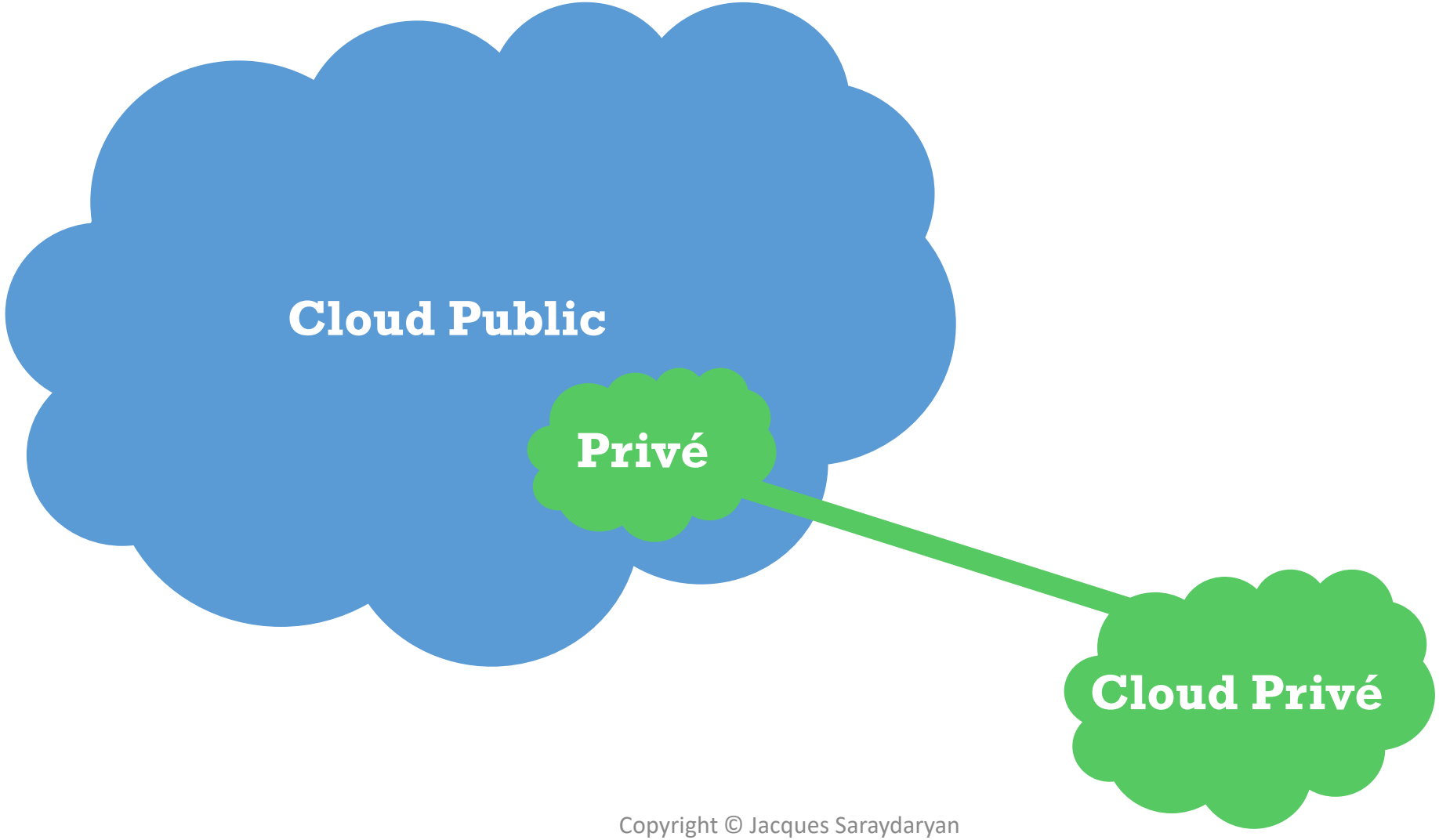


Privé

- Exécution sur les infrastructures d'une entreprise, ou d'un sous groupe de machines
- Associé a plus de sécurité



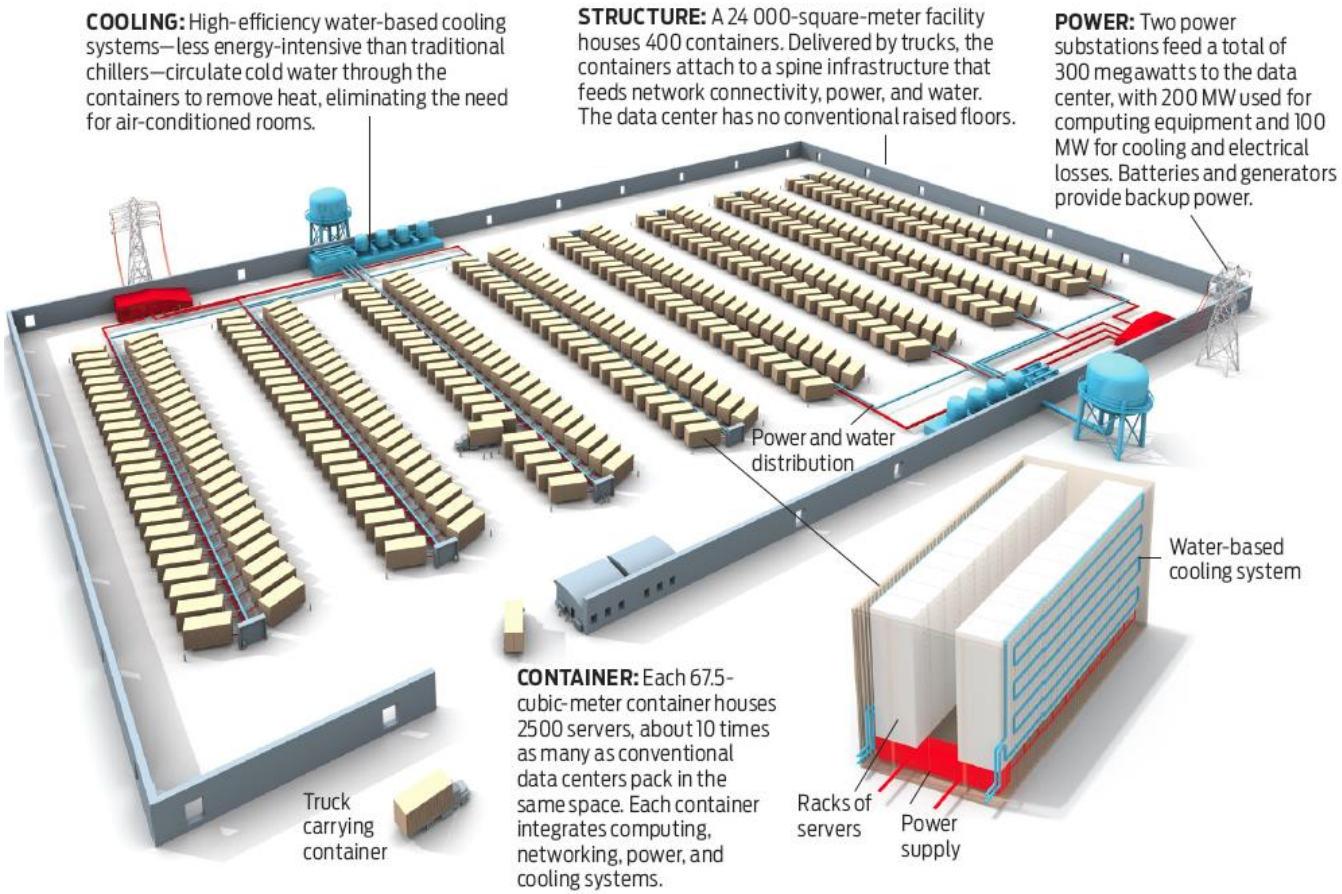
Différents Types de Cloud



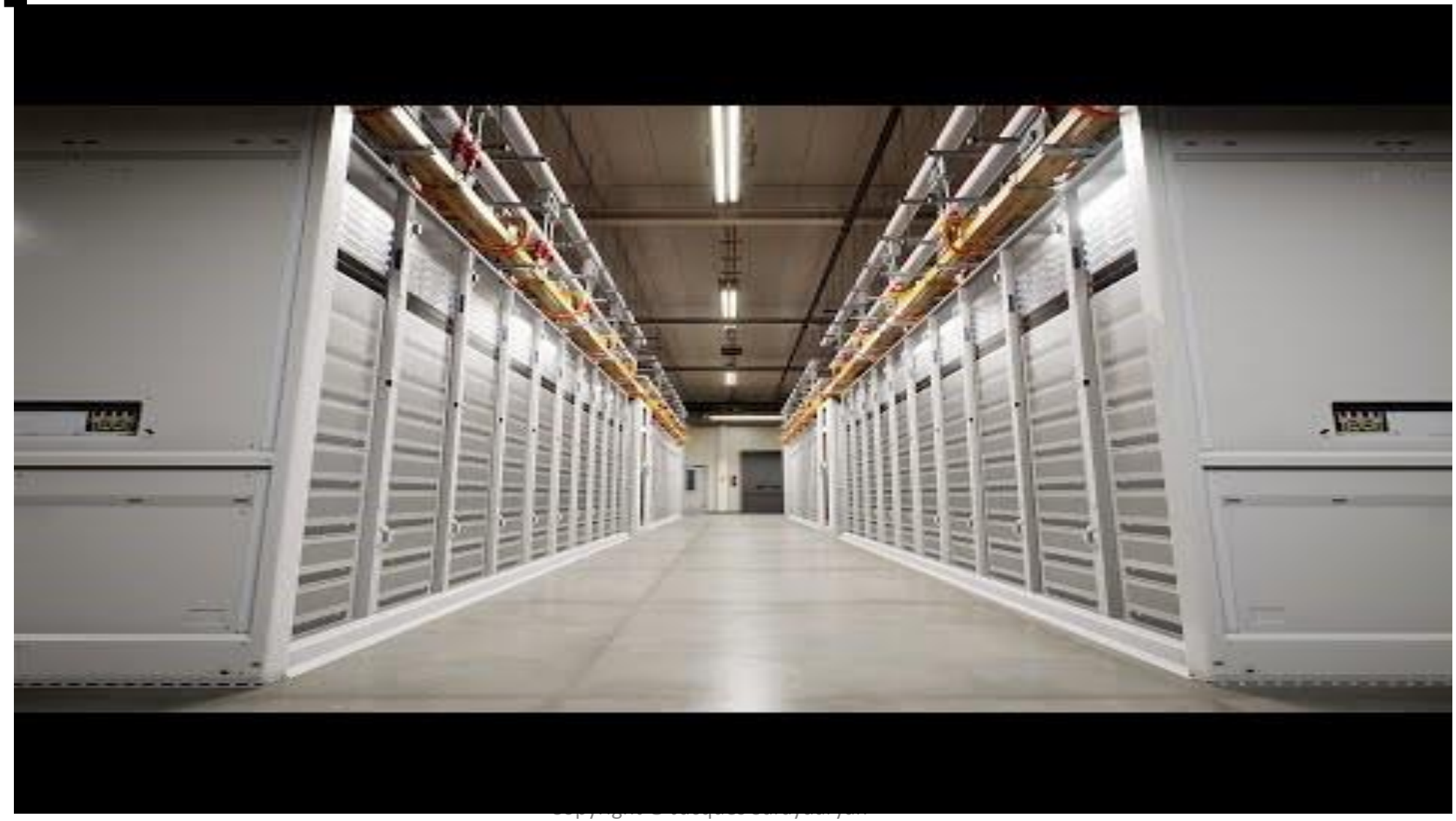
Exemple de datacenter

https://www.google.com/intl/fr/about/datacenters/discover/

Exemple de datacenter

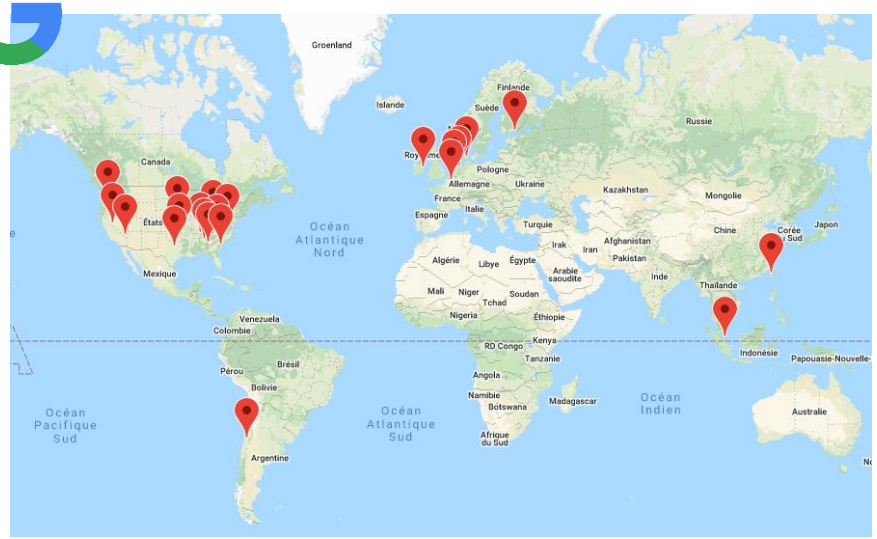


Exemple de datacenter



https://youtu.be/m9WhX-WDGHg

Datacenter Locations



https://www.google.com/about/datacenters/locations/

https://azure.microsoft.com/en-us/global-infrastructure/geographies

https://aws.amazon.com/fr/about-aws/global-infrastructure/

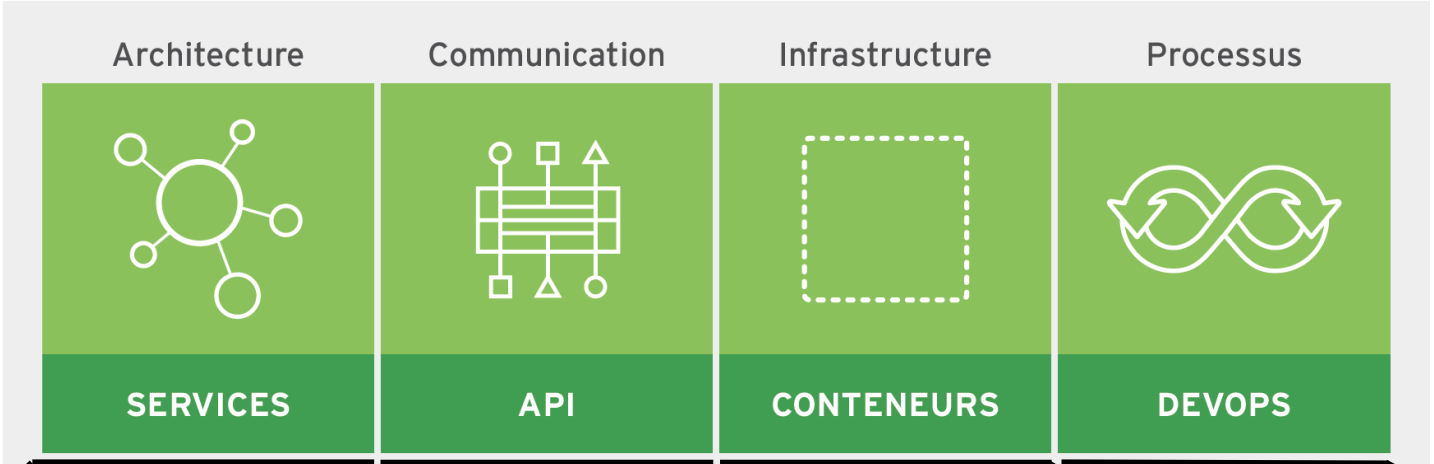
Architecture Cloud Ready/Native

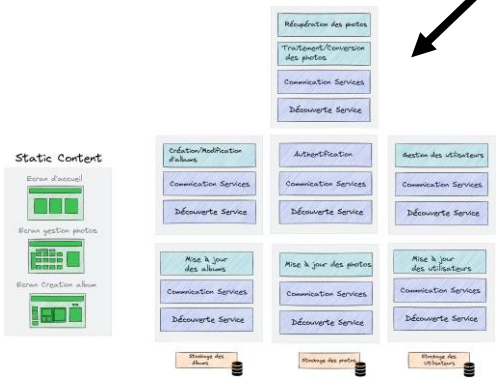
TABLEAU 1 : DÉVELOPPEMENT DES APPLICATIONS TRADITIONNELLES ET DES APPLICATIONS NATIVES POUR LE CLOUD

	TRADITIONNELLES	NATIVES POUR LE CLOUD
AXE	Longévité et stabilité	Commercialisation rapide
MÉTHODE DE DÉVELOPPEMENT	En cascade, semi-agile	Agile, pratiques DevOps
ÉQUIPES	Cloisonnement des équipes de développement, d'exploitation, de contrôle qualité et de sécurité	Collaboration des équipes DevOps
CYCLES DE DISTRIBUTION	Longs	Courts et continus
ARCHITECTURE D'APPLICATIONS	Couplage fort Monolithique	Couplage faible Basée sur des services Communication basée sur des API
INFRASTRUCTURE	Basée sur des serveurs Conçue pour les déploiements sur site Crée une dépendance Mise à l'échelle verticale Capacité maximale préapprovisionnée	Basée sur des conteneurs Conçue pour les déploiements sur site et dans le cloud Portabilité des applications Mise à l'échelle horizontale Capacité à la demande

https://www.redhat.com/fr/engage/cloud-native-application-development-20180622

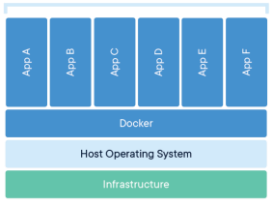
Architecture Cloud Ready/Native

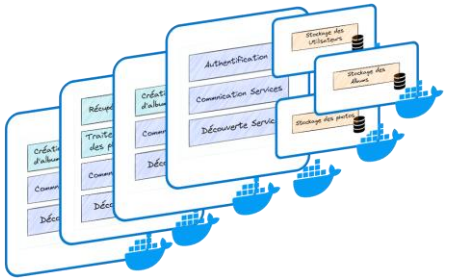


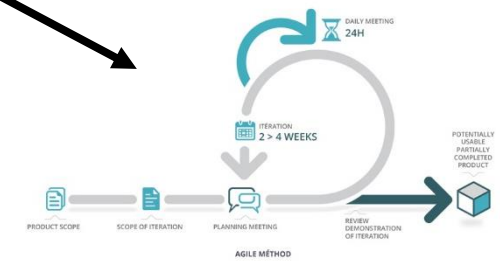


Architecture distribuée , N Tiers, Micro-services

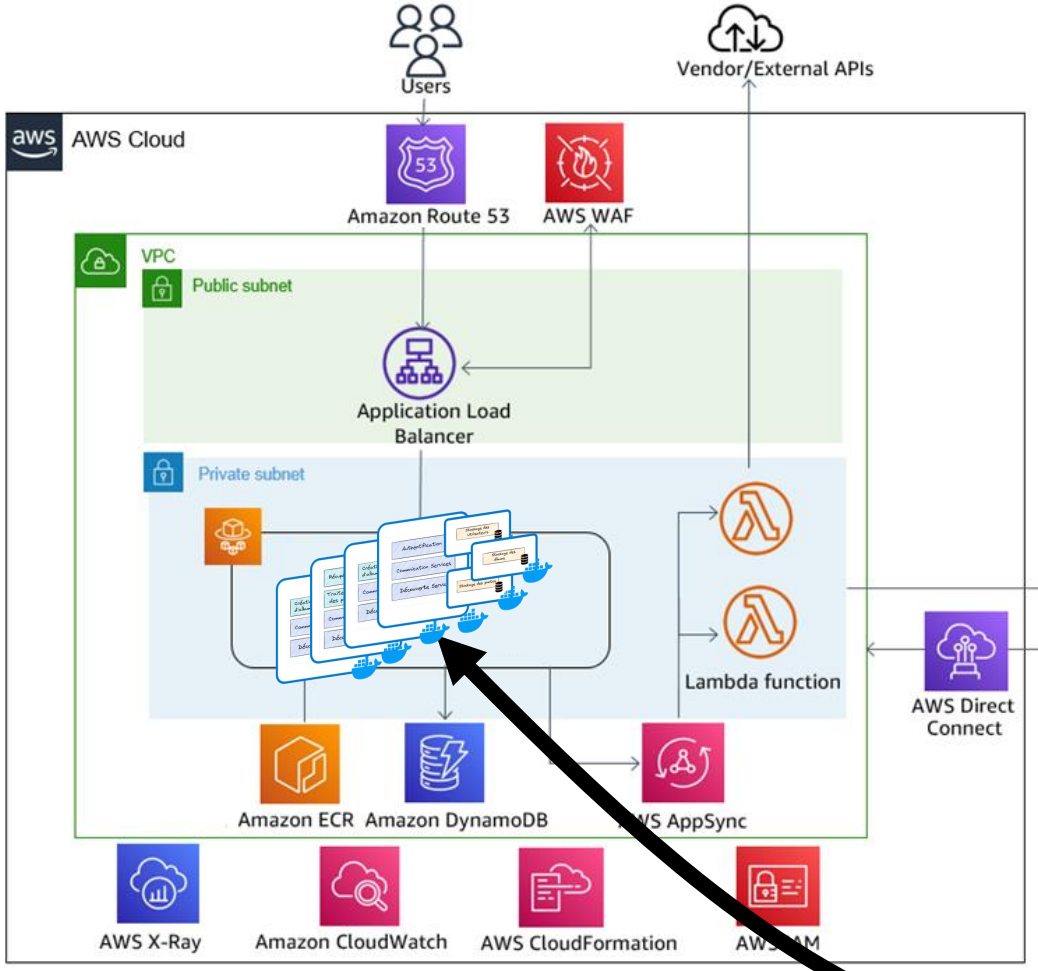
Comm. Client/Serveur (REST), Diffusion de message
(Message Broker)







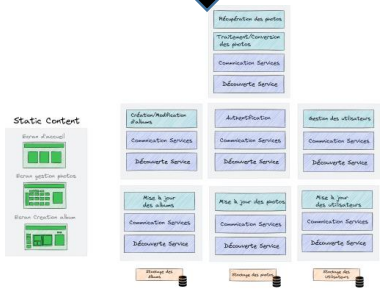
Architecture Cloud Ready/Native



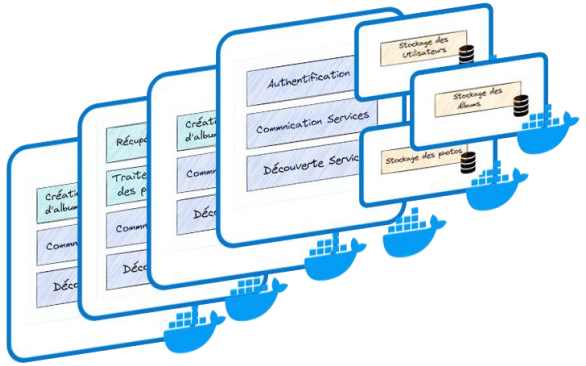
Projet / besoin

© Stocklib / andreykuzmin

Architecture / Dev.



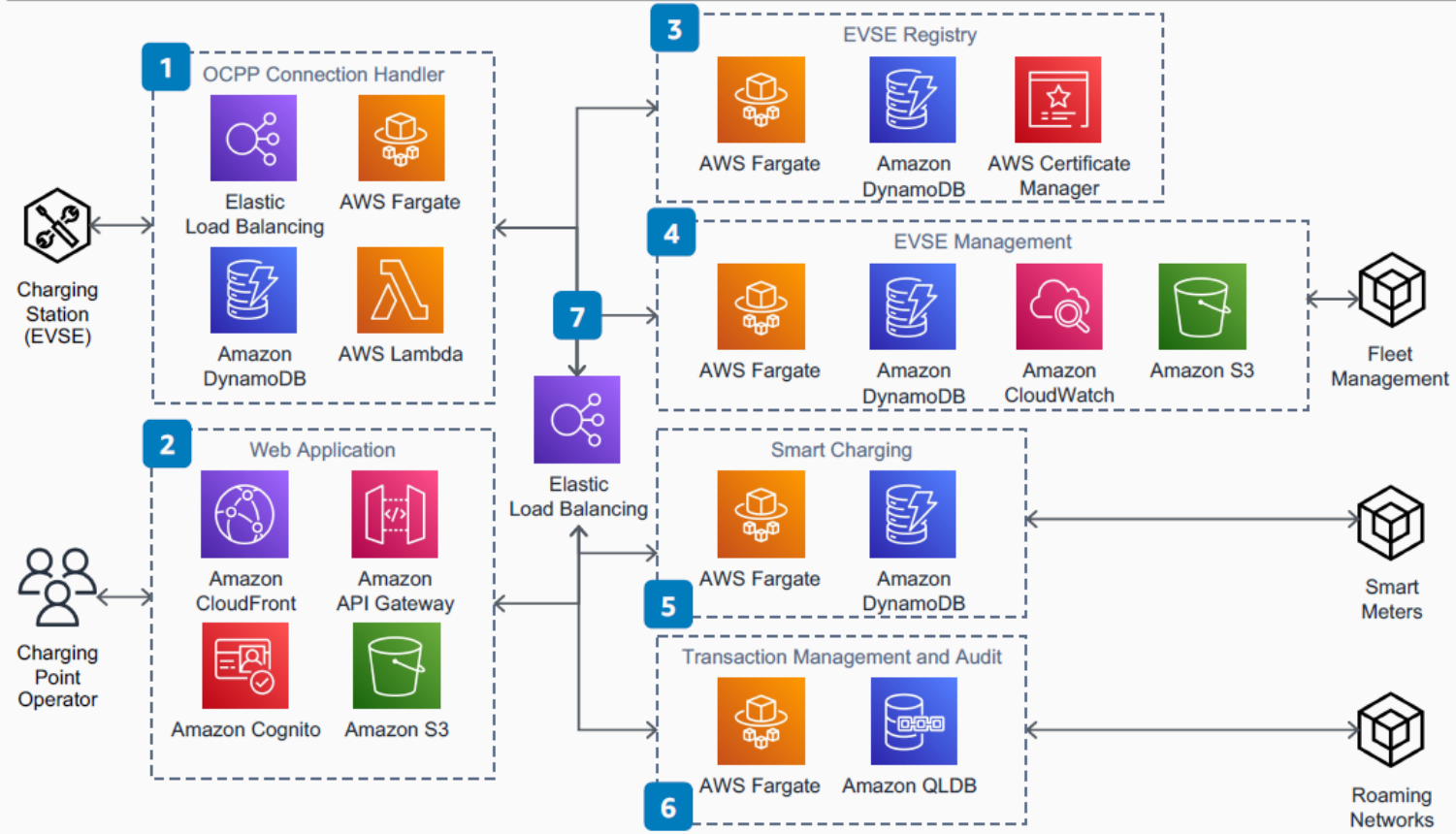
Containerisation / CI-CD



Architecture Cloud Ready/Native

Electric Vehicle Charging Station Management Software

Option 1: Container-based microservices architecture utilizing familiar technologies and concepts, most suitable for organizations embarking on containerization. Whether you're planning a development from scratch or the modernization of existing product, this architecture provides a scalable, reliable, and cost-efficient solution for the critical electric vehicle (EV) infrastructure.



1. Charging stations (electric vehicle supply equipment, or EVSE) are connected to the **Amazon Elastic Container Service (Amazon ECS)** on **AWS Fargate** behind **Network Load Balancer**. **AWS Lambda** routes outbound open charge point protocol (OCPP) messages to EVSEs and **Amazon DynamoDB** keeps active connections.
2. **Amazon CloudFront** serves static EVSE management web application from **Amazon Simple Storage Service (Amazon S3)** bucket. **Amazon API Gateway** exposes backend REST API for the web application. **Amazon Cognito** stores Charging Point Operator's user identities.
3. **Amazon DynamoDB** stores the registry of EVSE with credentials in encrypted form. **AWS Certificate Manager** manages EVSE certificates for authentication.
4. Management microservice delivers EVSE metrics to and from **Amazon CloudWatch**. **Amazon S3** used to store EVSE logs, cold metrics, and firmware files.
5. The configuration of charging sites, balancing algorithms, and power grid capacity stored in **Amazon DynamoDB.**
6. **Amazon Quantum Ledger Database (Amazon QLDB)** stores immutable and cryptographically verifiable log of charging transactions.
7. **Application Load Balancer** exposes the internal APIs of the microservices, routes requests, and balances between multiple tasks on **Amazon ECS** services.

Références

- Systèmes Distribués, Architecture N-Tiers, Eric Cariou Universtité de Pau et des Pays de l'Adour, Département Informatique, https://ecariou.perso.univ-pau.fr/cours/web/cours-architecture.pdf
- Systèmes Distribués, Introduction, Eric Cariou Universtité de Pau et des Pays de l'Adour, Département Informatique, https://ecariou.perso.univ-pau.fr/cours/cs-umbb/cours-intro.pdf
- Microsoft Azure: https://azure.microsoft.com/fr-fr/overview/what-is-cloud-computing/
- AWS: https://aws.amazon.com/fr/what-is-cloud-computing/
- Red Hat : https://www.redhat.com/fr/topics/cloud-computing/iaas-vs-paas-vs-saas
- Red Hat : https://www.redhat.com/fr/engage/cloud-native-application-development-20180622
- Haute disponibilité et datacentre, Blaise DAVID, Consultant Virtualisation, Quadix Technologies
- IBM: https://www.ibm.com/cloud/blog/four-architecture-choices-for-application-development
- Informatique réparties, Cecilia Zanni Merk, Alexandre Pauchet, INSA Rouen Normandie
- https://blog.octo.com/strategie-d-architecture-api/

✉ **Jacques Saraydaryan**
Jacques.saraydaryan@cpe.fr