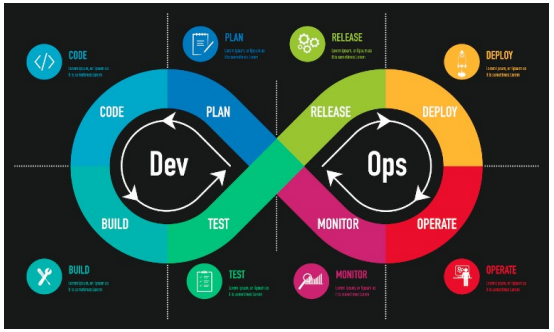
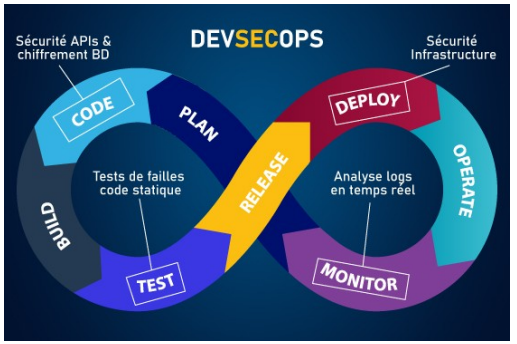
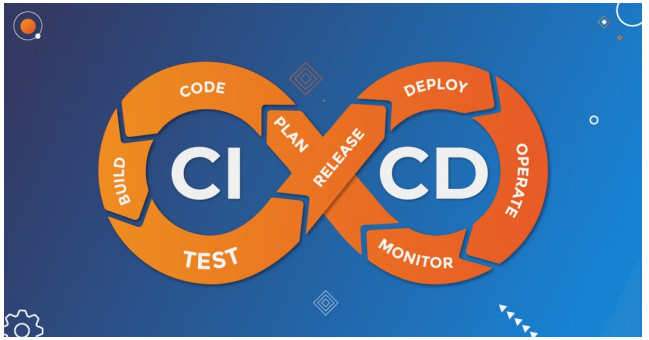


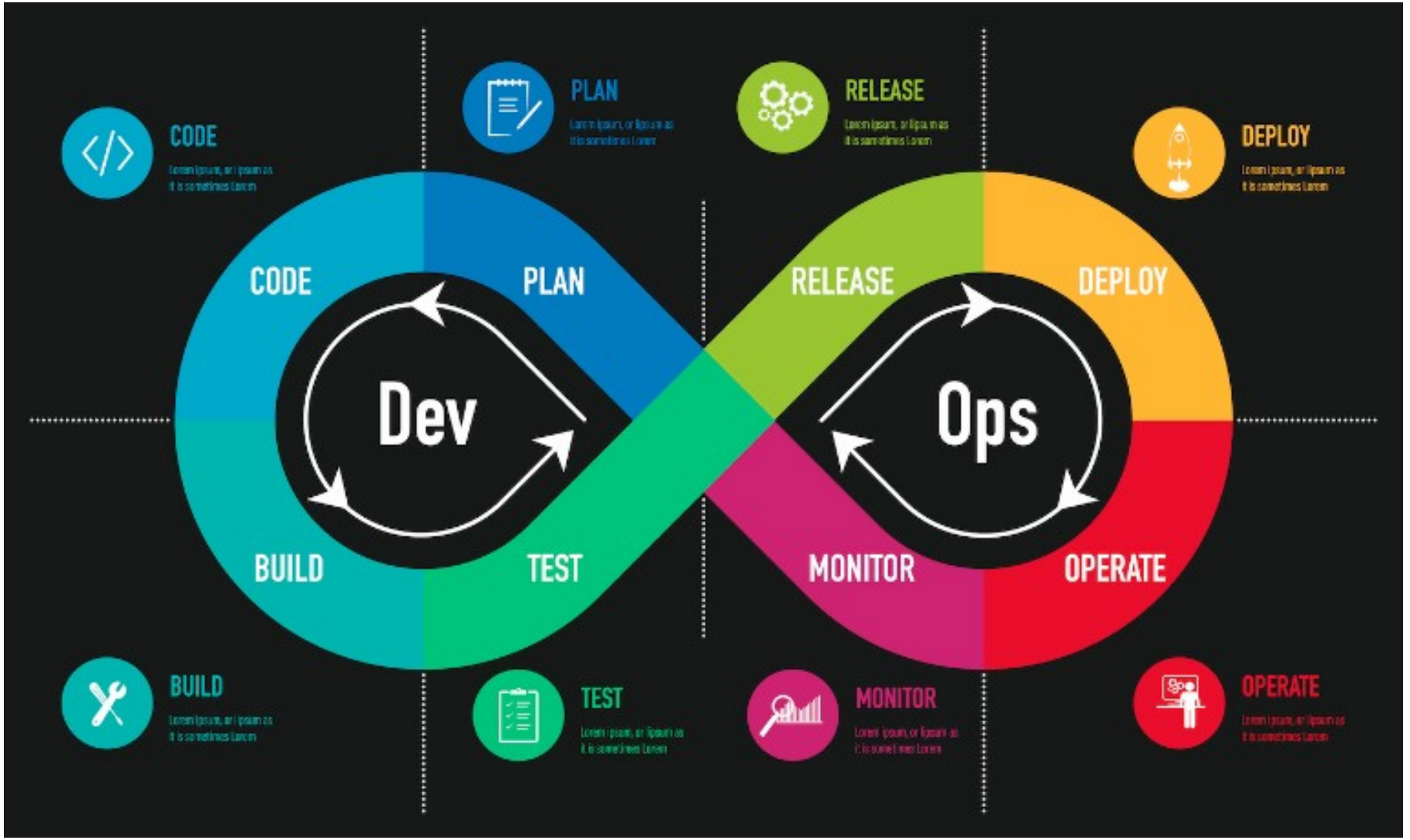




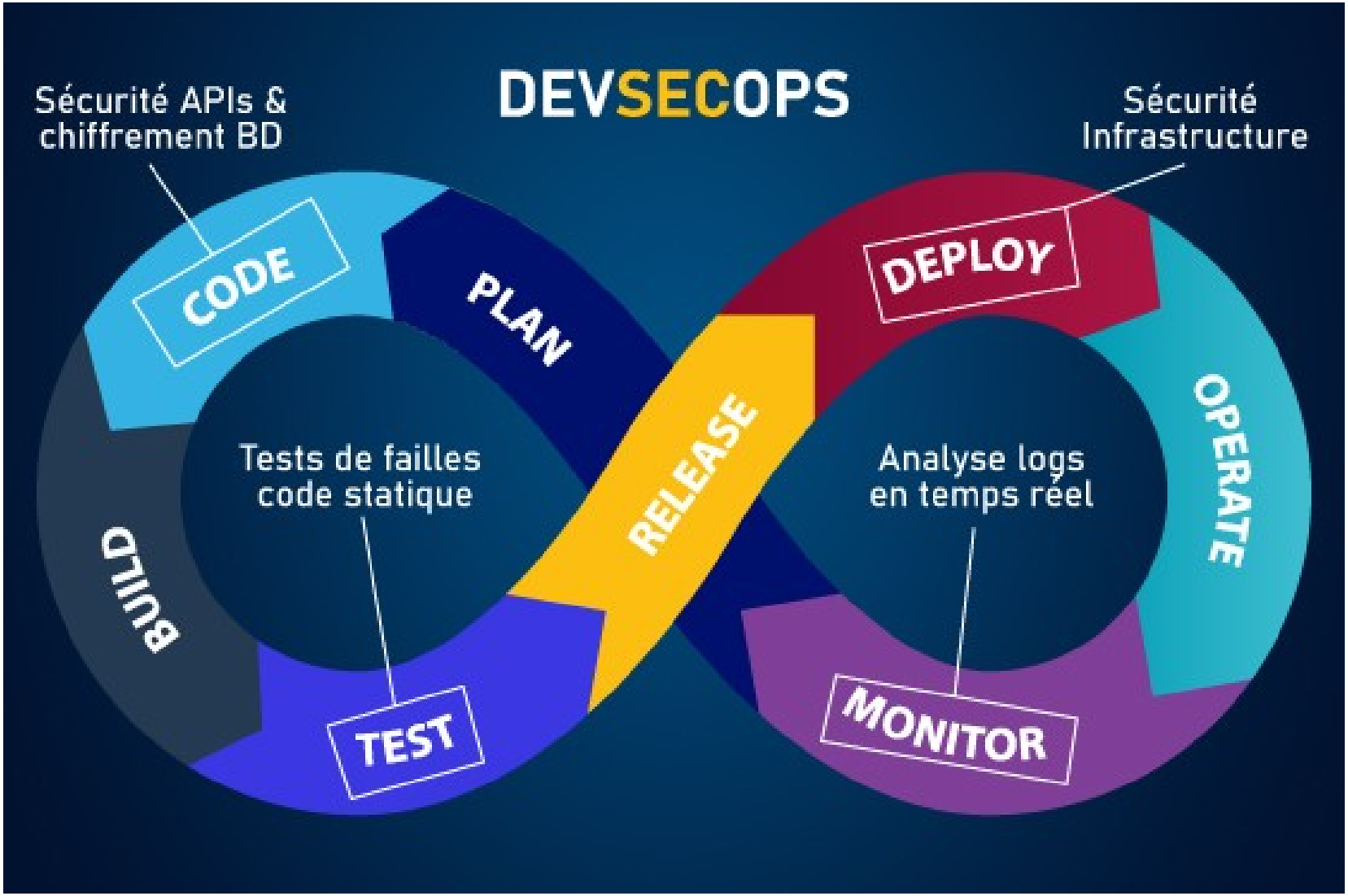
Cours CI/CD

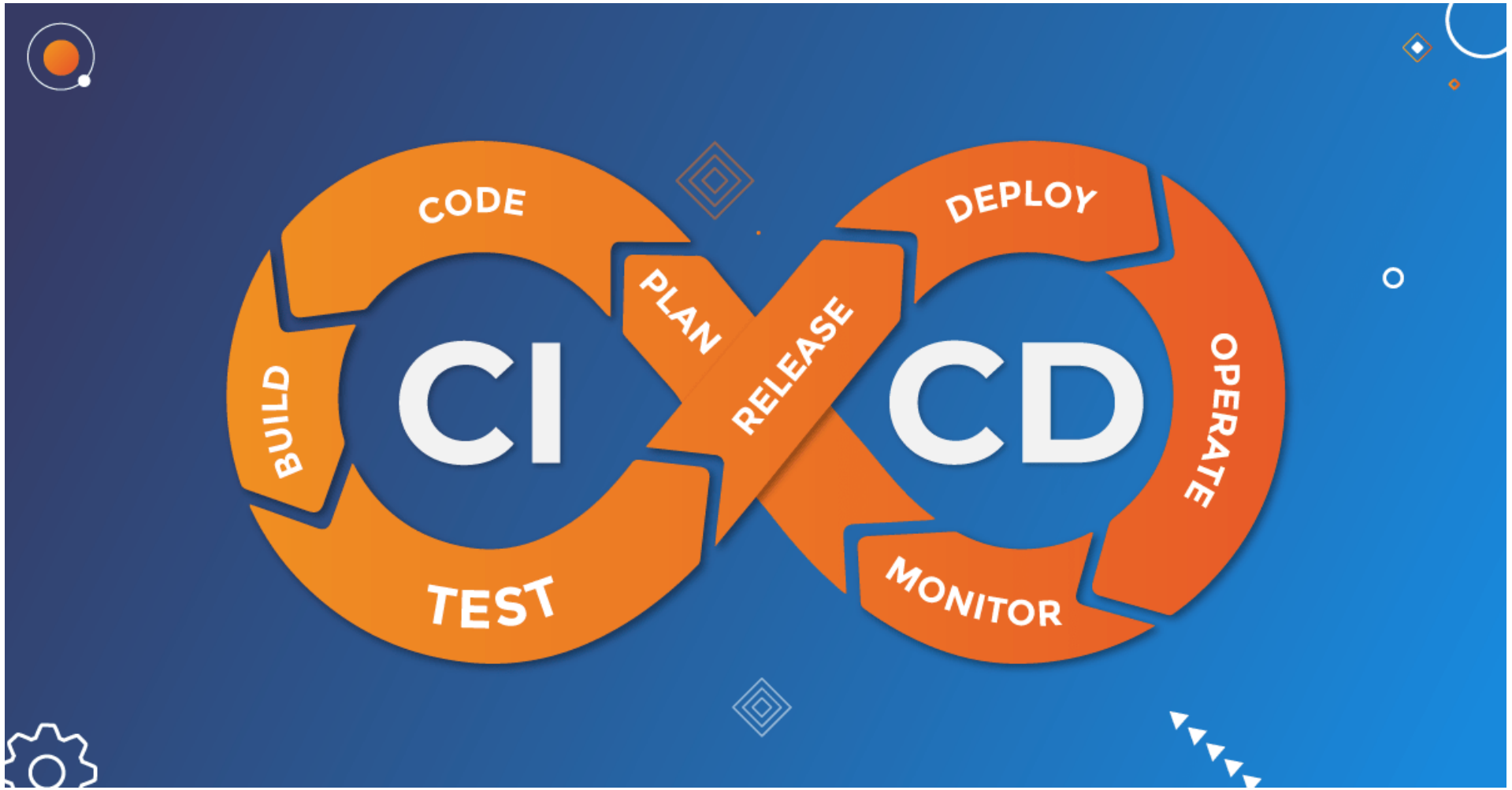
Y. Stroppa - 2023
BUT 3

DevOps Panorama



+Sécurité


DEVSECOPS
Sécurité APIs & chiffrement BD
Sécurité Infrastructure
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR
Tests de failles code statique
Analyse logs en temps réel


CODE
PLAN
BUILD
CI
RELEASE
TEST
DEPLOY
OPERATE
CD
MONITOR

Bibiliographie

Learning DevOps

Second Edition : Mikael Krief

Pratical DevOps

PACKT : Joakim Verona

Starting with DevSecOps

Java Code Geeks

Apprenez Jenkins : ebook

Site :

https://github.com/ahmedamsaleh/Free-DevOps-Books-1/tree/master/book

https://github.com/gladjoe/Free-DevOps-Books-1/tree/master/book

Web

https://devops.com/

https://devops.com/9-pillars-of-continuous-security-best-practices/

https://kubernetes.io

https://redhat.com --> Openshift

Sites d'utilisation en ligne

https://killercoda.com/kubernetes/scenario/playground

https://play-with-docker.com/

https://labs.play-with-k8s.com/

Sources d'informations sur le sujet

- DZone
- https://www.simform.com/
- https://medium.com/
  - Medium daily Digest

Sigles et acronymes

CNCF : Cloud Native Computing Foundation

CNI : Networking for Linux containers

OCI : Open Container Initiative

CRI : Container Runtime Interface

DevOps :

CI/CD : Continuous Integration / Continuous Delivery

SDLC : Software Development Life Cycle

OWASP : Open Web Application Security Project

Sommaire

Rappels :

- Virtualisation
- Conteneurisation (Docker, Podman, Cri-o ...)

Architecture d'orchestrateurs

- Swarm (Docker)
- Kubernertes

Outils de configuration/paramétrage

- Ansible (Continuous Deployment)
- Chef

CI/CD

- Git, Github
- Tests (Junit, Selenium, Playwrith et autres)
- Exemple de CI/CD avec Jenkins
- Exemple de CI/CD avec gitlab

Rappel sur devops

Historique de Devops : 2008 conférence à Toronto

==> mouvement issu de conférences professionnelles

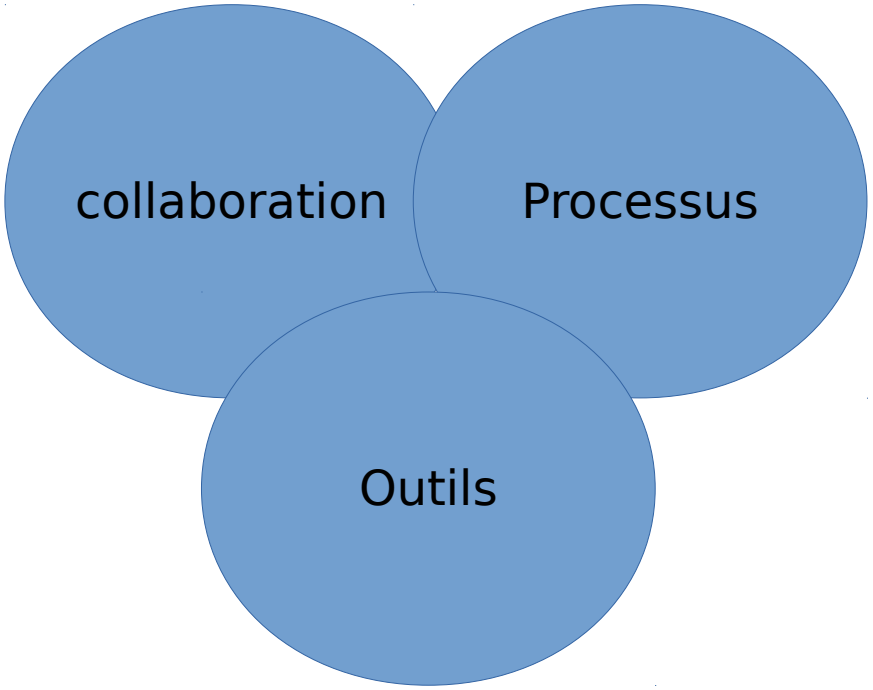
==> première fois en 2008 Patrick Debois et Andrew Shafer

démarche Lean : objectif améliorer la performance d'une organisation en matière de productivité, de qualité, de délais et de coûts.

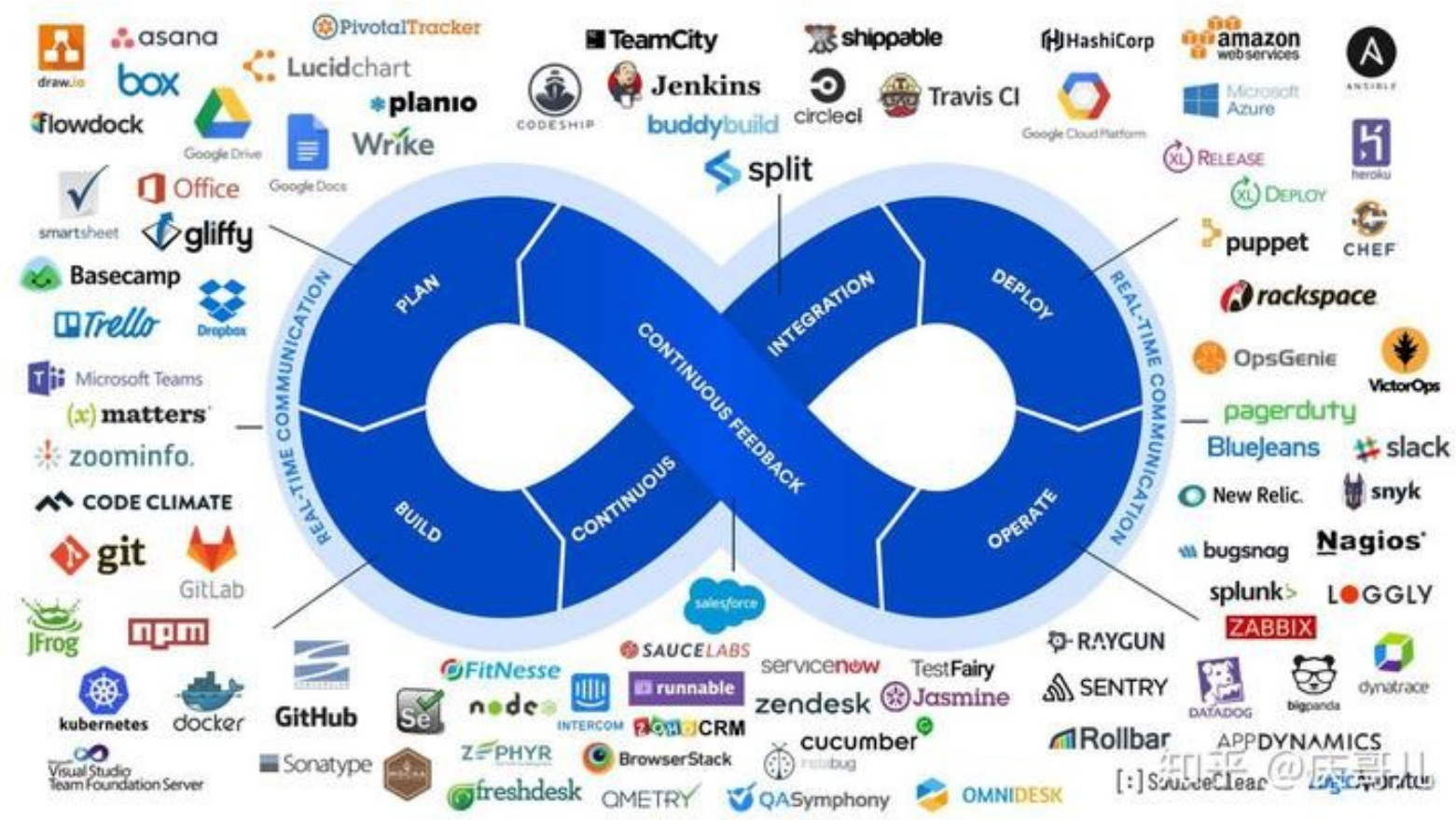
DevOps

Union des cultures :

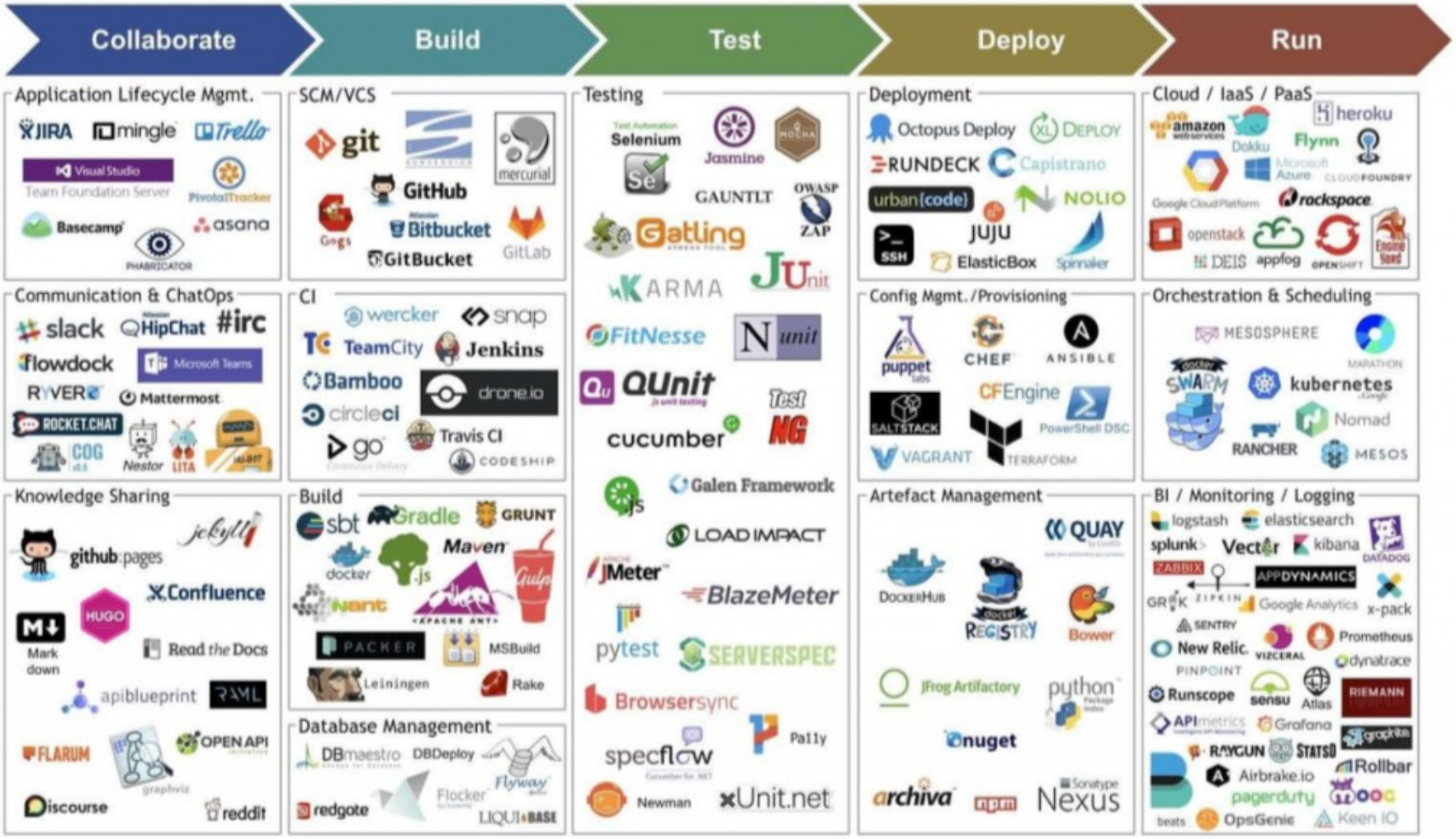
Développement et Opérations



Introduction - panorama des outils

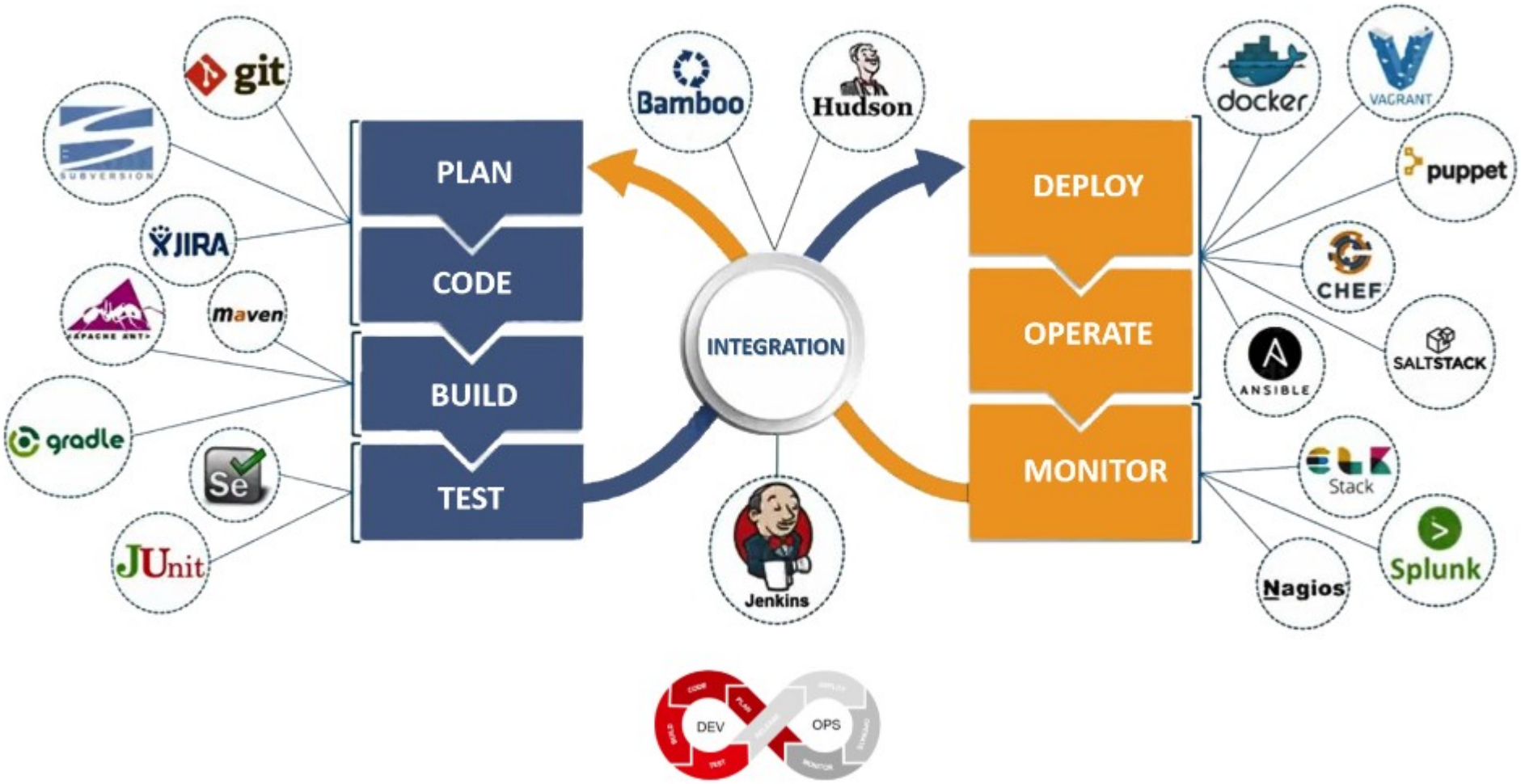


Introduction - panorama des outils



Introduction - panorama

PIPELINE CI/CD



Les outils pour manipuler

Outils en ligne

- https://killercoda.com/
  - Compte gilhub, gitlab, google ....
- https://labs.play-with-docker.com/
  - compte : docker
- https://labs.play-with-k8s.com/
  - compte, github ou docker

Outils à l'IUT

- virtualBox
- Accès aux VMs de proxmox ??

Rappel Virtualisation

Plusieurs intérêts à virtualiser

isoler les éléments par projet

éviter les conflits entre différentes versions

préserver l'intégrité de son operating system

partager des éléments

possibilité de libérer et d'archiver une fois le projet terminé

Plusieurs solutions possibles

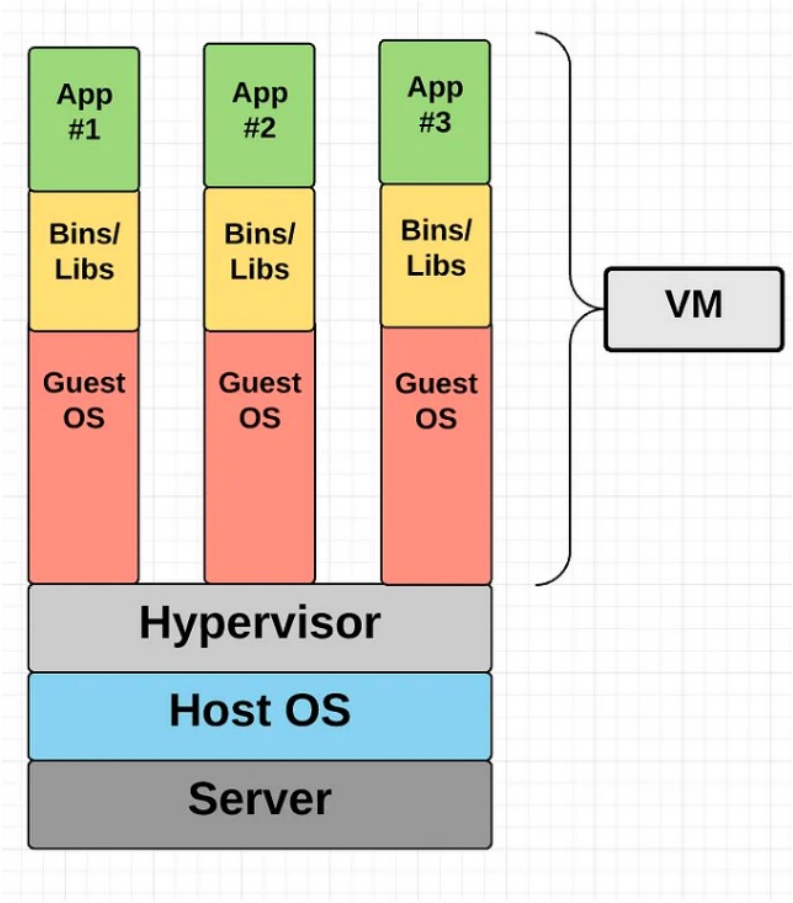
Poste de travail :

VirtualBox, VmWare Workstation, parallels, hyper V, UTM ...

Serveurs :

proxmox, qemu-system ..., kvm, HyperV, etc ..

Rappel Virtualisation



Virtual Machine Diagram

Rôle de hyperviseur : gérer l'accès aux ressources communes pour l'ensemble des VMs et de l'hôte.

Rappel Virtualisation

Caractériser sa VM

taille mémoire, espace disque, cpu, interface réseau

Mode connexion réseau :

NAT, NATNETWORK, BRIDGE, HOST ONLY ....

Communication entre les VMs sous VirtualBox

Attention à définir un NatNetwork et d'accrocher vos VMs sur ce réseau

Attention à l'utilisation du mode bridge : monte la VM au même niveau que l'Hôte pour la rendre accessible dans le même LAN.

Rappel conteneurisation

Conteneurs : liens

docker :

podman :

crio : https://access.redhat.com/documentation/fr-fr/openshift_container_platform/3.11/html/cri-o_runtime/use-crio-engine

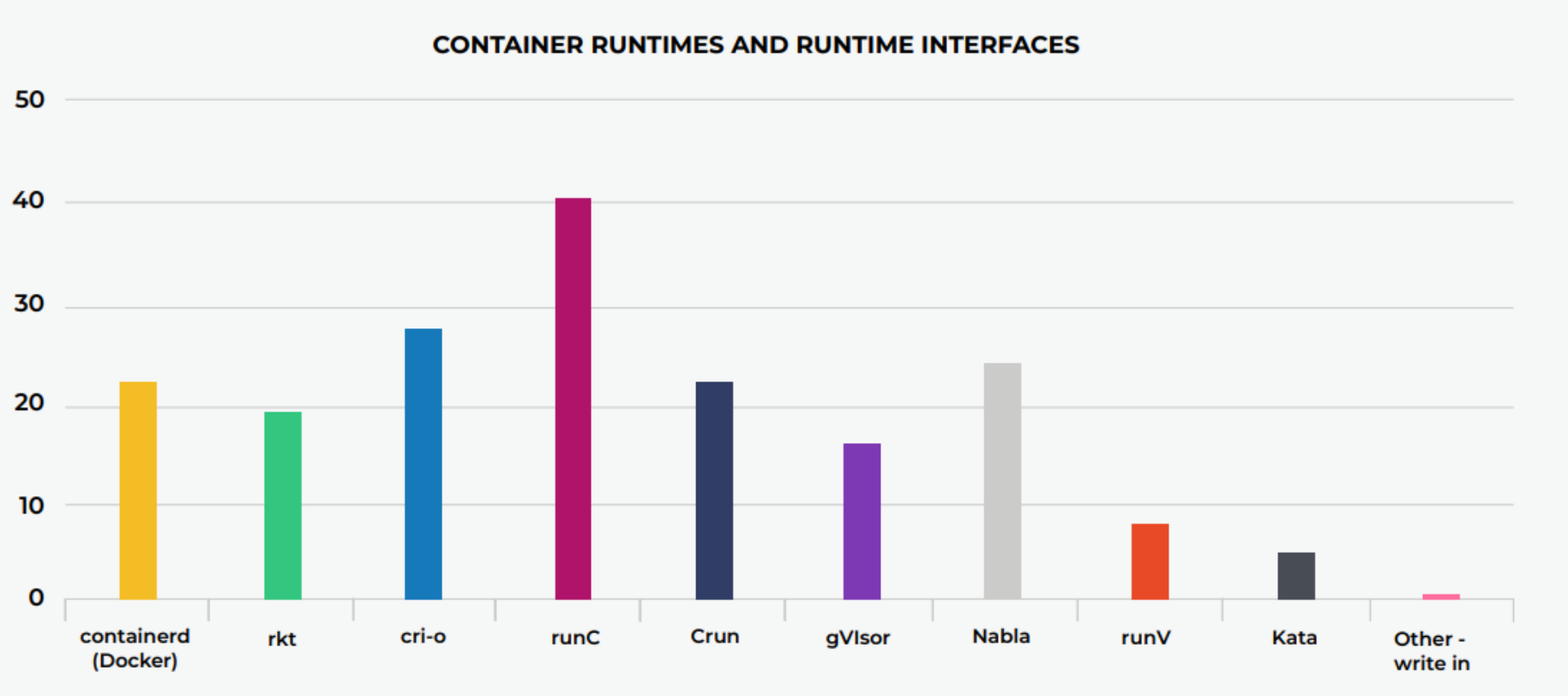
https://cri-o.io/

Kata-container

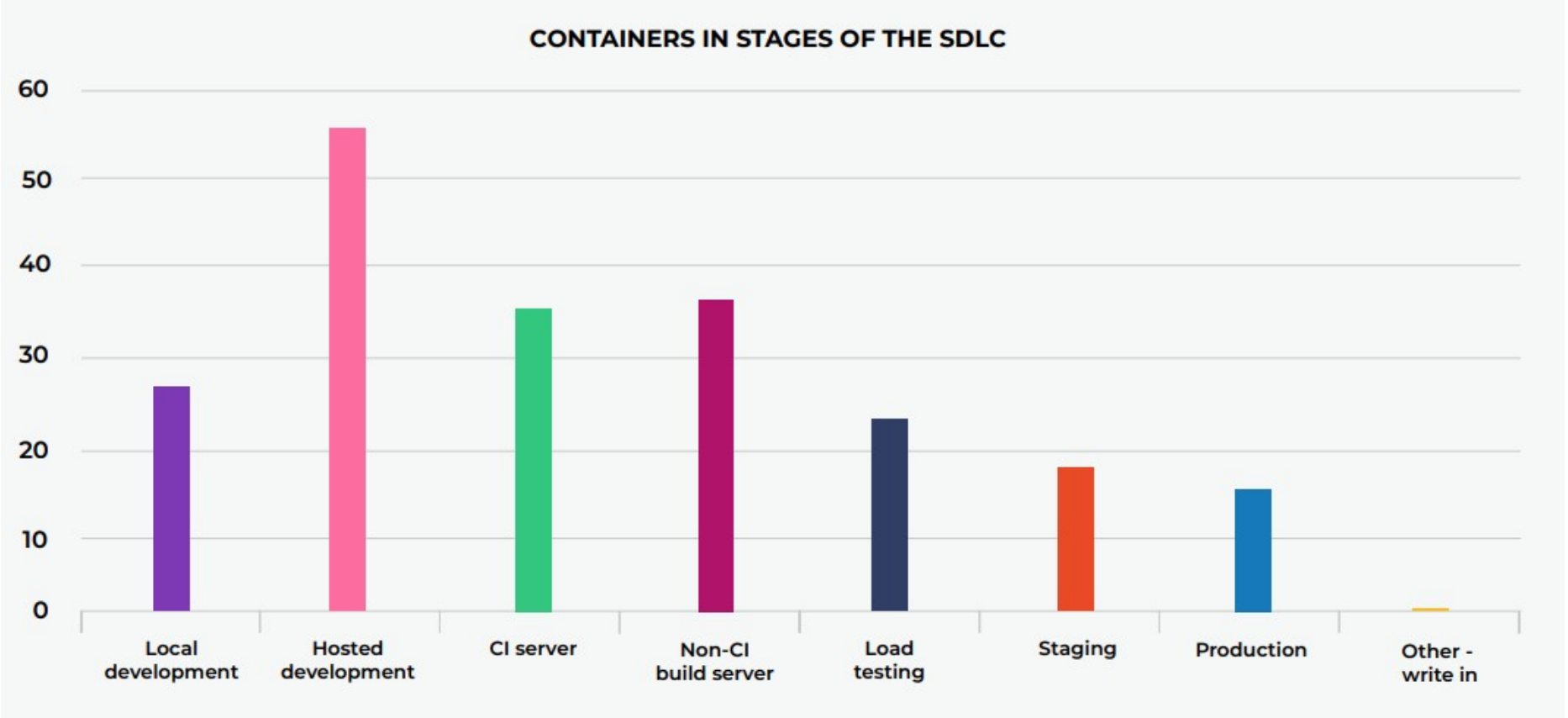
singularity :

Rappel : les conteneurs runtime

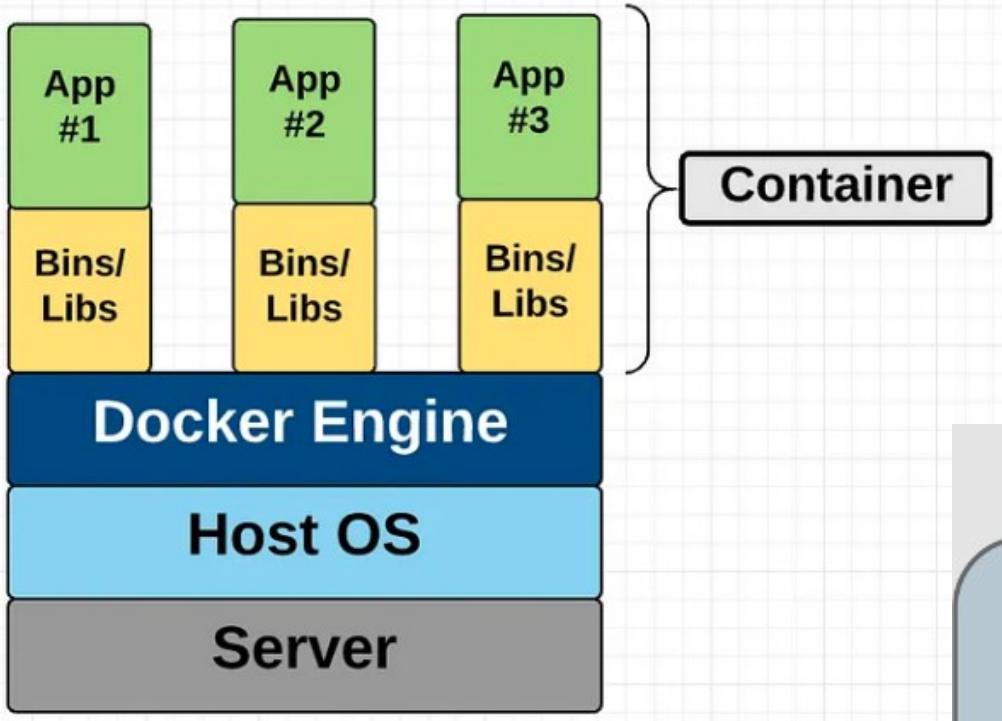
LXC, Docker , Podman, rkt, singularity, crio



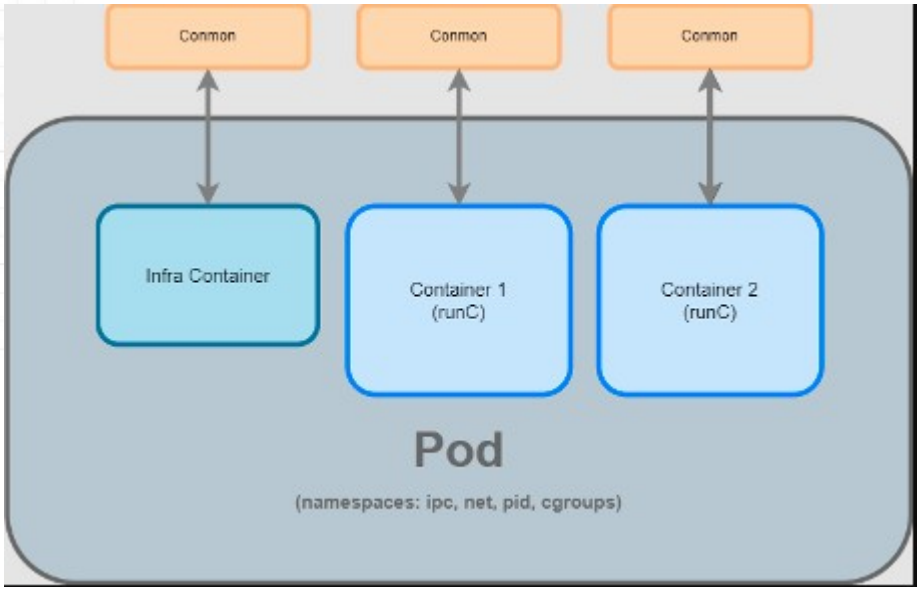
Rappel : les conteneurs



Rappel conteneurisation



Container Diagram



Rappel conteneurisation

Conteneurisation apporte des solutions et des facilités pour le développeur :

- être capable de mettre en place un service sans installation
- de mettre plusieurs mêmes versions ou pas d'un même service sur la même machine
- d'isoler les services les uns des autre : notion de namespaces dans un contexte de ressources cgroups
- rapidité d'exécution comparer à une VM

Notions de namespaces/cgroups

namespace : est un regroupement de processus qui ont une vision commune (hostname, processus, réseau, montage)

différents types de namespaces

pid, user, uts, net, mnt

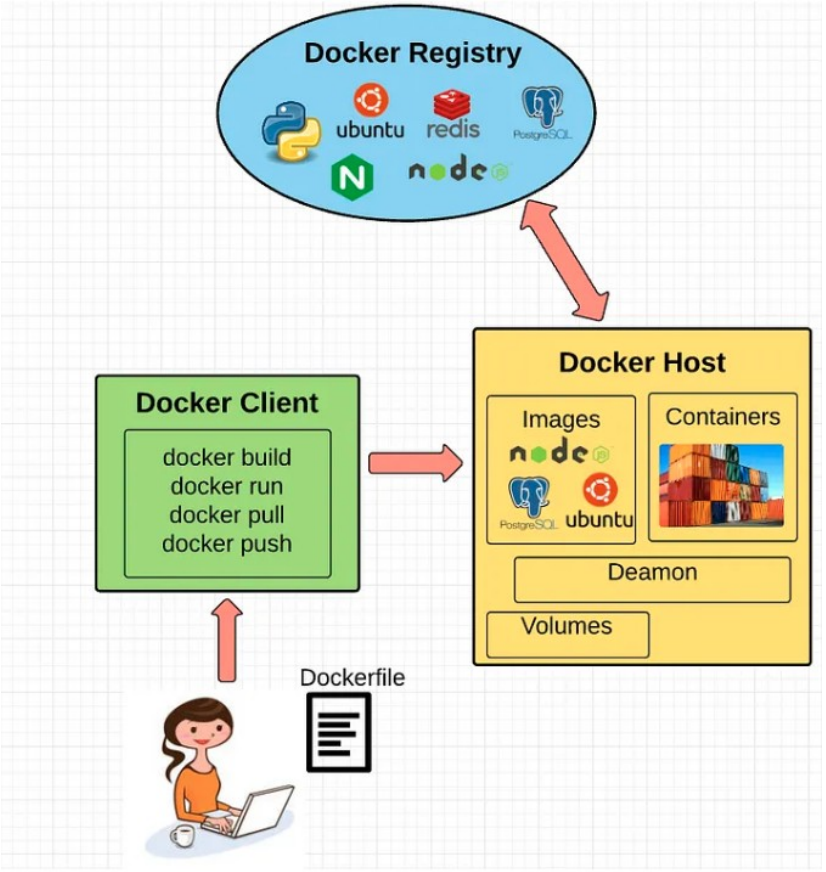
commande : lsns

Cgroup : control group permet de fixer et de limiter la taille des ressources allouées

commandes : systemd-cgls et systemd-cgtop

répertoire : /sys/fs

Docker


Docker Registry
ubuntu
redis
PostgreSQL
node
Docker Client
docker build
docker run
docker pull
docker push
Docker Host
Images
node
PostgreSQL
ubuntu
Containers
Deamon
Volumes
Dockerfile

Docker (rappel)

Point sur docker-compose : 2014

version python 1.x--> version go 2.x

Docker compose a été écrit en python et communiquait avec le démon docker via son API REST.

Visant à gérer des groupes de conteneurs basés sur une définition YAML

Ce YAML a reçu par la suite une spécification formelle --> la specification Compose
https://github.com/compose-spec/compose-spec/blob/master/spec.md

cette spécification définit un langage structuré vous permettant d'exécuter facilement plusieurs conteneurs sur une seule machine.

Alternative à l'exécution de ces conteneurs directement à partir de la ligne de commande.

Voir version en fonction

Podman : redhat

Podman s'exécute en tant qu'utilisateur non privilégié et sans daemon central.

Podman est sorti en 2018, concentrés au début sur la compatibilité avec la ligne de commandes (CLI) de Docker . N'a pas inclut tout de suite la prise en charge de l'API REST .

Un projet communautaire Podman Compose a vu le jour. Podman compose traite la spécification compose et la traduit en commandes CLI Podman.

Podman s'oriente plus vers Kube pour son interfaçage

Podman : exemple de commande

podman run -dt --publics-all --rm docker.io/httpd

activation un processus conmon

```
ystroppa@vmteo:/etc/containers$ podman ps
CONTAINER ID  IMAGE                           COMMAND         CREATED        STATUS
PORTS                   NAMES
f86aba57ad8a  docker.io/library/httpd:latest  httpd-foreground  13 seconds ago  Up 13
seconds ago  0.0.0.0:37785->80/tcp  bold_visvesvaraya
```

```
ystroppa@vmteo:/etc/containers$ podman
port -l
80/tcp -> 0.0.0.0:37785
```

```
ystroppa@vmteo:/etc/containers$ curl http://localhost:37785
<html><body><h1>It works!</h1></body></html>
```

Podman : notion de pod

Un pod est une grappe(groupe de conteneurs) qui vont partager des informations entre eux (ipc, net et uts).

cri-o

Rappel : les conteneurs
Best Practices

Use Smaller Base Image :

les tags sur docker hub indique la version de l'image et de l'OS.

Plus c'est petit, plus en réduit la surface d'attaque, analogie aux OS. Ne pas laisser de package par défaut qui ne sert à au processus que l'on souhaite démarrer.

Always use .dockerignore file

Afin de préciser les fichiers qui ne sont pas nécessaires pour le conteneur par exemple dans un projet le dist, build, Readme ou Dockerfile.

Use Specific Docker Image Version

éviter les FROM python en version latest et préférer FROM python:3.11 afin d'avoir une meilleure maîtrise et possibilité de reproductibilité de l'image.

Do not use the root user

éviter d'utiliser le moins de privilèges possibles. Indiquez dans la fabrique à la fin USER ... sinon par défaut c'est root.

Use multi-stage Builds

afin de ne pargder que l'essentiel à l'exécution du conteneur, comme par exemple avec java ; On n' a pas de besoin de JDK mais d'un JRE. Donc avec la notion mutli-stage on peut lors de la construction utiliser des environnements temporaires afin de produire le livrable et pousser uniquement le jar par exemple avec le jre dans l'image finale.

Sécurité autour des conteneurs

docker scout

https://docs.docker.com/scout/quickstart/

Identifie les CVEs

- Common Vulnerabilities and Exposure
- Exemple : docker scout recommendations httpd

Container Multi-Architecture

Multi-arch Docker image --> liste d'images avec binaires et librairies compilées pour de multi-architectures (ARM, x86, RISC-V,...)

- Performance and Cost Optimization
- Cross Platform Developement
- IoT Devices

Bénéfices d'utiliser des images multi-architectures

- capacité à exécuter une image docker sur plusieurs archi
- capable de choisir son eco-friendly CPU architecture
- moindre effort dans la migration d'une architecture vers une autre
- meilleure performance et cout en utilisant ARM64
- capacité à utiliser plusieurs cores par CPU en utilisant ARM64

Comment construire des images de conteneurs en multi-architectures

- Méthode traditionnelle
- en utilisant docker buildx
- en utilisant buildah

Container Multi-Architecture / manuelle

Dockerfile
FROM nginx
RUN echo"Hello But3" > /usr/share/nginx/html/index.html

sur une machine amd64
docker build -t ystroppa2/custom-nginx:v1-amd64
docker push ystroppa2/custom-nginx:v1-amd64

sur une machine arm64
docker build -t ystroppa2/custom-nginx:v1-arm64
docker push ystroppa2/custom-nginx:v1-arm64

Création du fichier manifest
docker manifest create ystroppa2/custom-nginx:v1 \
   ystroppa2/custom-nginx:v1-amd64 \
   ystroppa2/custom-nginx:v1-arm64
docker manifest push ystroppa2/custom-nginx:v1

Container Multi-Architecture / buildx

Dockerfile
FROM nginx
RUN echo"Hello But3" > /usr/share/nginx/html/index.html

docker buildx build --push --platform linux/amd64,
linux/arm64 -t ystroppa2/custom2-nginx:v1

Attention à la version de docker avec l'option
platform

Avec Buildah

Dockerfile
FROM nginx
RUN echo"Hello But3" > /usr/share/nginx/html/index.html

Au préalable installer
sudo apt install qemu-user-static

buildah manifest create multiarch-test
buildah bud --tag ystroppa2/cut-nginx:v1-amd --manifest multiarch-test --arch amd64 .
buildah bud --tag ystroppa2/cut-nginx:v1-arm --manifest multiarch-test --arch arm64 .
buildah manifest
buildah manifest inspect multiarch-test

Résultat sur docker hub

TAG

v1

Last pushed **a few seconds ago** by ystroppa2

```
docker pull ystroppa2/custom-ng…
```

DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
fd1d74fc6de0	linux/amd64	15 minutes ago	51.23 MB
9ad1e90f98a3	linux/arm64/v8	5 minutes ago	64.07 MB

contenu du manifest

```
(base) ystroppa@ystroppa-Latitude-5510:~/BUT3/multi-arch$ docker manifest inspect  ystroppa2/custom-nginx:v1
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
   "manifests": [
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1777,
         "digest": "sha256:fd1d74fc6de007b532bc3ce955a34ad4103449838632c15ece7b3b1ce049cbcc",
         "platform": {
            "architecture": "amd64",
            "os": "linux"
         }
      },
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1985,
         "digest": "sha256:9ad1e90f98a3beb374c3bd9f4131825ad79cd79bc61463502618cb132e0e1e35",
         "platform": {
            "architecture": "arm64",
            "os": "linux",
            "variant": "v8"
         }
      }
   ]
}
```

Container Multi-Architecture

Deux liens de réfs :

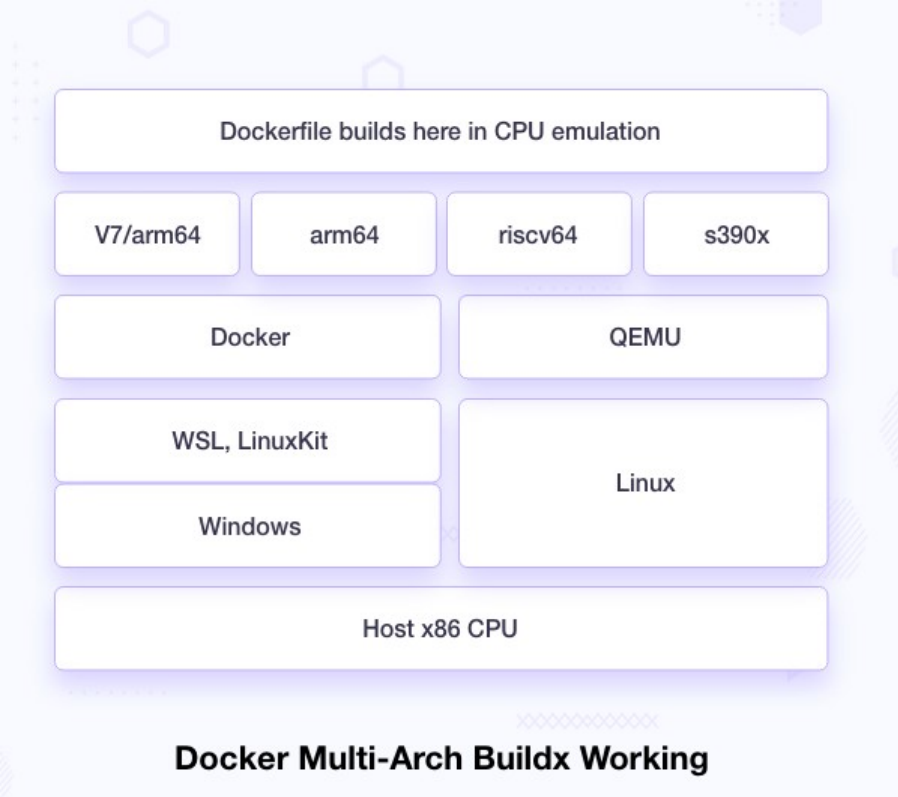
Avec docker :

https://dzone.com/articles/understanding-multi-arch-containers-benefits-and-c

Avec Buildah

https://danmanners.com/posts/2022-01-buildah-multi-arch/

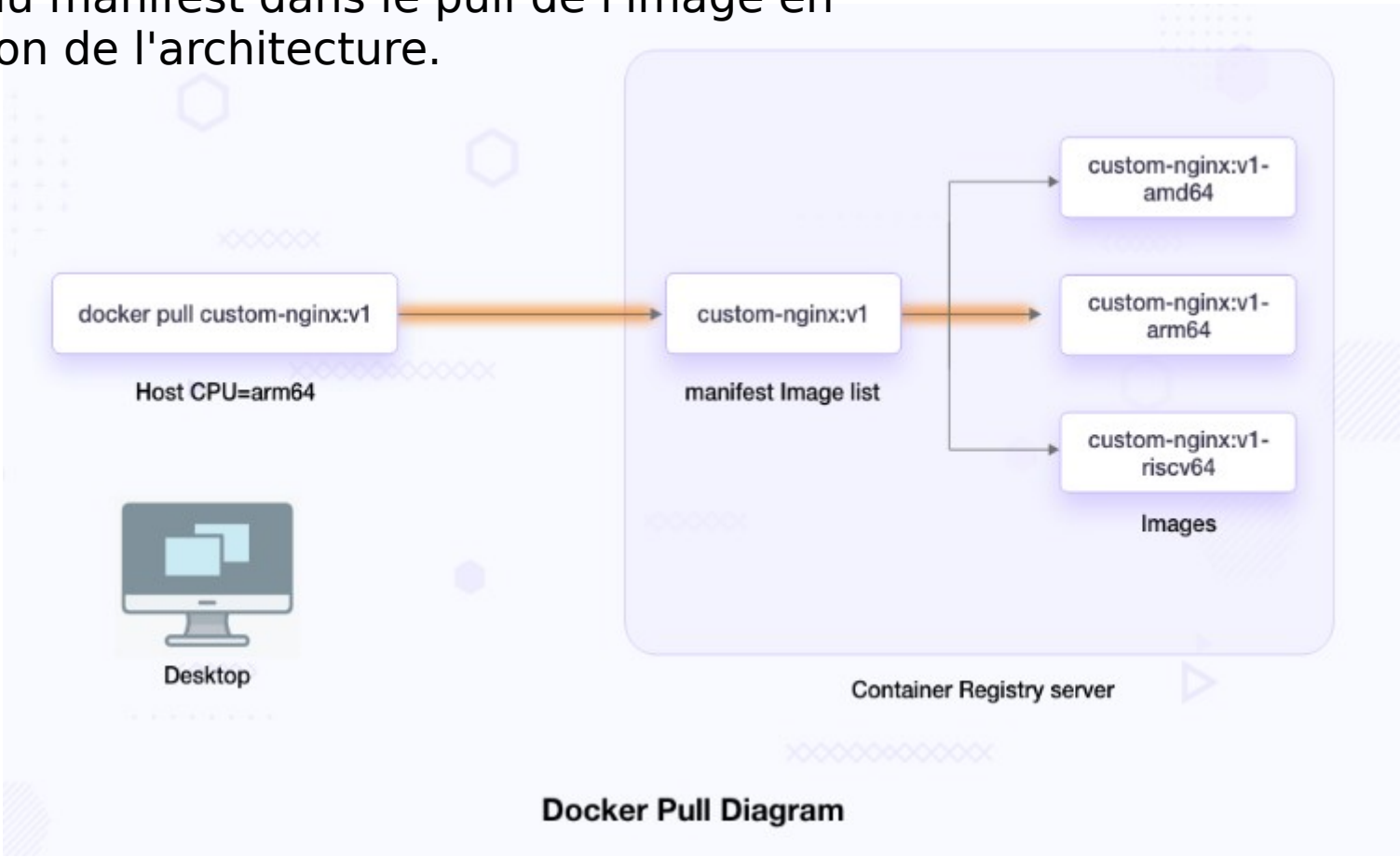
Container Multi-Architecture



Docker Multi-Arch Buildx Working

Container Multi-Architecture

Rôle du manifest dans le pull de l'image en fonction de l'architecture.



Multi-arch : intégration dans Jenkins CI

Description du pipeline de traitement

```
pipeline
{
    agent {
        label 'worker1'
    }
    options{
        timestamps()
        timeout(time: 30, unit: 'MINUTES')
        buildDiscarder(logRotator(numToKeepStr: '10'))
    }
    environment {
        DOCKER_REGISTRY_PATH = "https://registry.example.com"
        DOCKER_TAG = "v1"
    }
```

Multi-arch : intégration dans Jenkins CI

```
    stages
    {
        stage('build-and-push')
        {
        steps{
          script{
            docker.withRegistry(DOCKER_REGISTRY_PATH, ecrcred_dev){
            sh '''
              ######## check multiarch env ############
              export DOCKER_BUILDKIT=1
              if [[ $(docker buildx inspect --bootstrap | head -n 2 | grep Name | awk -F" " '{print $NF}') !=
"multiarch" ]]
              then
                docker buildx rm multiarch | exit 0
                docker buildx create --name multiarch --use
                docker buildx inspect --bootstrap
              fi
              ######## Push multiarch ############
              docker buildx build --push --platform linux/arm64,linux/amd64 -t
"$DOCKER_REGISTRY_PATH"/username/custom-nginx:"$DOCKER_TAG" .
            '''
          }
        }
      }
    }
}
```

Multi-arch intégration dans Github CI

```
name: docker-multi-arch-push
on:
  push:
    branches:
      - 'main'
jobs:
  docker-build-push:
    runs-on: ubuntu-20.04
    steps:
      - name: Checkout Code
        uses: actions/checkout@v3
      - name: Set up QEMU
        uses: docker/setup-qemu-action@v2
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2
```

```
- name: Login to docker.io container registry
  uses: docker/login-action@v2
  with:
    username: $
    password: $
- name: Build and push
  id: docker_build
  uses: docker/build-push-action@v3
  with:
    context: .
    file: ./Dockerfile
    platforms: linux/amd64,linux/arm64
    push: true
    tags: username/custom-nginx:latest
```

docker : sécurité

Installation de scout pour vérifier la vulnérabilité des images

Identification des CVEs 'commun Vulnerability and exposure)

https://cevdetails.com; https://cve.org

Exemples d'utilisations :

docker scout cves alpine

docker scout cves httpd

docker scout recommandations httpd

(informations renvoyées par ces commandes :

C : Critical, H : High, M : Medium, L : Low)

Que-est-ce CI/CD pipeline ?

CI/CD est la pierre angulaire de l'approche DevOps, une série d'étapes orchestrées qui conduisent le développement logiciel par la production de code, l'exécution de tests et le déploiement de la nouvelle application à travers différents environnements.

L'objectif ultime d'un pipeline CI/CD est de minimiser les erreurs humaines et d'avoir une release consistante du logiciel en automatisant les tâches répétitives .

Que-est-ce CI/CD pipeline ?

En automatisant les différentes phases du pipeline, l'équipe de dev peut travailler rapidement et améliorer la qualité logicielle de base sur le pipeline.

- les pipelines logiciels sont généralement personnalisés pour répondre aux besoins particuliers du projet et de l'entreprise.
- différents types d'outils pour la création de binaires, la compilation de code, les tests unitaires, l'analyse de code et la sécurité peuvent être inclus dans le pipeline.
- les environnements conteneurisés automatisent le conditionnement du code dans une image qui peut être déployée dans le cloud
- des pipelines multiples sont souvent nécessaires pour amener le code source en production en fonction de la structure de l'organisation et de l'équipe.
- un pipeline CI/CD peut être déclenché par un événement, comme une modification de code, la présente d'un nouvel artefact dans le repository ou un calendrier défini pour une nouvelle release.

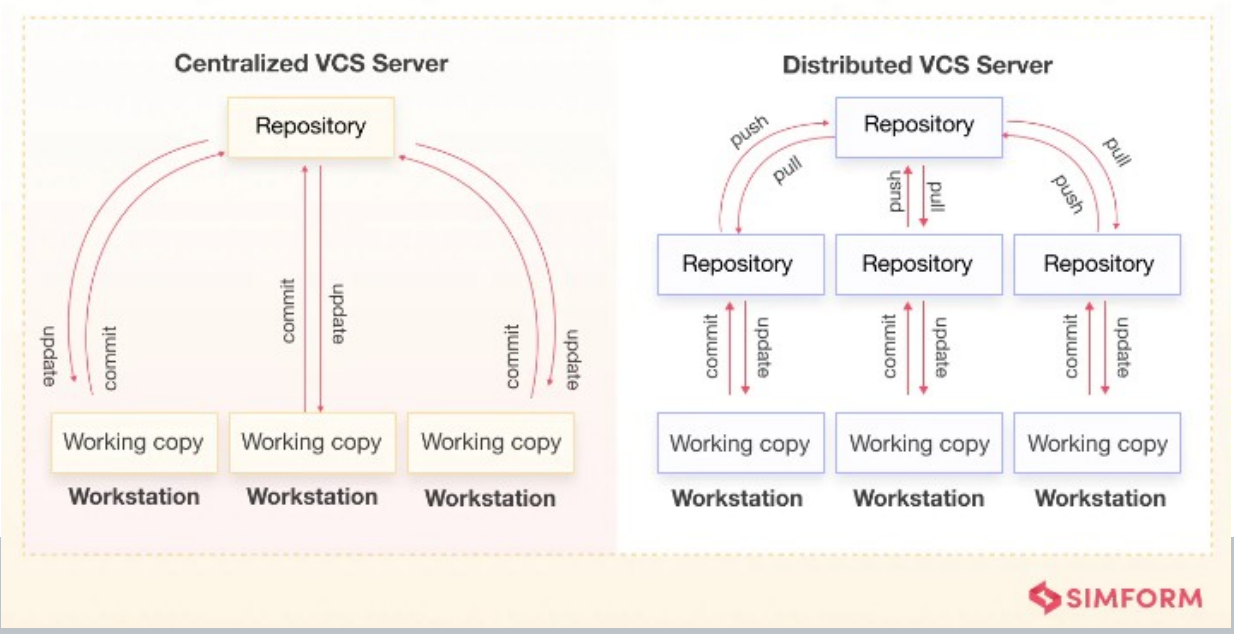
Que-est-ce CI/CD pipeline ?

Un pipeline CI/CD comporte généralement plusieurs scripts qui doivent être exécutés dans un ordre spécifique pour atteindre un objectif commun. le code source au début du pipeline, passe par une série d'étapes à l'intérieur du pipeline, puis est publié en tant que code prêt pour la production.

En règle générale, le pipeline CI/CD comporte 4 étapes importantes (Sources, construction, test et déploiement). Chaque phase sui des normes approrpiées, des processus bien détaillés et des outils spécifiques.

Que-est-ce CI/CD pipeline ?

Source : les développeurs traduisent les exigences en algo fonctionnels, caractéristiques et comportements. Toutes modification d'un indicateur pré configuré dans le référentiel du code ou dans le programme déclenche le pipeline CI/CD. On peut avoir d'autres déclencheurs résultats d'autres pipelines.



mercurial,
subversion SVN
Team foundation version Control

git
AWS codecommit

Que-est-ce CI/CD pipeline ?

Build : cette étape du pipeline CI/CD extrait le code source du référentiel, le lie aux bibliothèques, dépendances et modules divers pour construire le livrable. Outre la construction du code, l'automatisation de la construction implique également l'utilisation d'outils pour garantir la sécurité du code et s'assurer qu'il respecte les meilleures pratiques de CI/CD.

Parallèlement, les outils utilisés dans cette phase génèrent des journaux du processus, analysent le code à la recherche d'erreurs stylistiques et programmatiques et informent les développeurs de l’achèvement du code.

makefile,
ant,
maven,
gradle

jenkins
Travis,
CircleCI
Azure Pipelines
....

Que-est-ce CI/CD pipeline ?

Tests: maintenant que le code source a été soumis à des tests statiques, la construction entre dans une phase de tests dynamiques qui commencent par une série de tests fonctionnels de base. Ensuite la version est soumise à divers tests tels que de conformités , les tests de bout en bout (E2E) etc ... pour valider son fonctionnement.

Cette phase peut durer quelques secondes à quelques heures en fonction de la taille et de la complexité du produit.

Quelques outils :

- Selenium
- Appium
- Jest
- Playwright

et biens d'autres ainsi que des scripts maison qu'il va falloir également gérer et entretenir.

Que-est-ce CI/CD pipeline ?

Deploy : dans un pipeline de livraison continue, la version est d'abord envoyée aux parties prenantes humaines pour approbation, tandis que la version est automatiquement déployée après avoir passé sa suite de tests dans un pipeline de déploiement continu. Dans cette phase, le code approuvé est présenté sous forme d'artefact et déployé dans les environnements appropriés principalement dans un environnement de mise en scène puis dans un environnement d'assurance qualité et enfin dans l'environnement de production.

Cette étape doit être adaptée pour soutenir la stratégie de déploiement appropriée "Blue-Green" aux déploiements "Canari" en passant par les déploiements "sur place". le déploiement implique naturellement des précautions supplémentaires et des périodes de test en direct, y compris des tests A/B, des tests Beta, des tests Blue-Green .... pour le moindre impact possible sur le business et éviter les retours arrières.

L'étape de déploiement peut impliquer le provisionnement de l'infrastructure (Terraform), la configuration (Puppet) et la conteneurisation (Docker, Kubernetes).

D'autres outils :

Chef, Ansible, AWS codeDeploy, AWS Elastic Beanstalk

Azure Pipeline

.....

Pipeline CI/CD

Attributs d'un bon Pipeline CI/CD

Le but ultime de l'utilisation d'un pipeline CI/CD est de donner aux équipes un retour d'information rapide, fiable et complet sur leur production et de leur permettre de déployer des logiciels dans la meilleur qualité possible.

Par conséquent, un bon pipeline doit présenter des caractéristiques de base :

Précision/Accuracy

la responsabilité de l'exécution des flux incombe aux outils Ci/CD. cela permet d'élimer les erreurs manuelles dans les tâches répétitives et permet au pipeline d'avoir une vision précise de l'ensemble du processus de livraison.

Fiabilité / Reliability

Un pipeline fiable fonctionnerait à chaque fois sans générer d'erreurs, éliminant ainsi la frustration des développeurs face à une charge de travail accrue en plus d'une qualité logiciel compromise.

Vitesse /Speed

Un pipeline optimisé s'exécute rapidement et fournit un retour d'information rapide aux développeurs pour un échec ou un succès. Des pipelines plus rapides permettent également plus de déploiements.

Quelques exemples de pipeline

Il existe de multiples façons et bien sûr des outils pour créer un pipeline. Des noms comme Git, Jenkins, Docker, Kubernetes, ...

De plus en plus d'organisations adoptent aujourd'hui les microservices. Cependant les pipelines de microservices sont des pipelines différents des pipelines traditionnels. Et avec le nombre croissant de microservices, la gestion de pipeline CI/CD distincts pour chaque service peut devenir rapidement complexe.

Challenges :

Si une organisation utilise un pipeline CI/CD par service, elle devra supporter des centaines de pipelines, chacun connecté à un dépôt git séparé.

Un microservice peut également avoir plusieurs pipelines pour l'usage interne, la production ou les environnements de tests ce qui conduit à une duplication encore plus importante.

la consolidation des codes avec des bibliothèques et des plugins partagés peut s'avérer assez difficile. les bibliothèques partagées créent généralement des conflits spécifiques aux versions.

Quelques exemples de pipeline

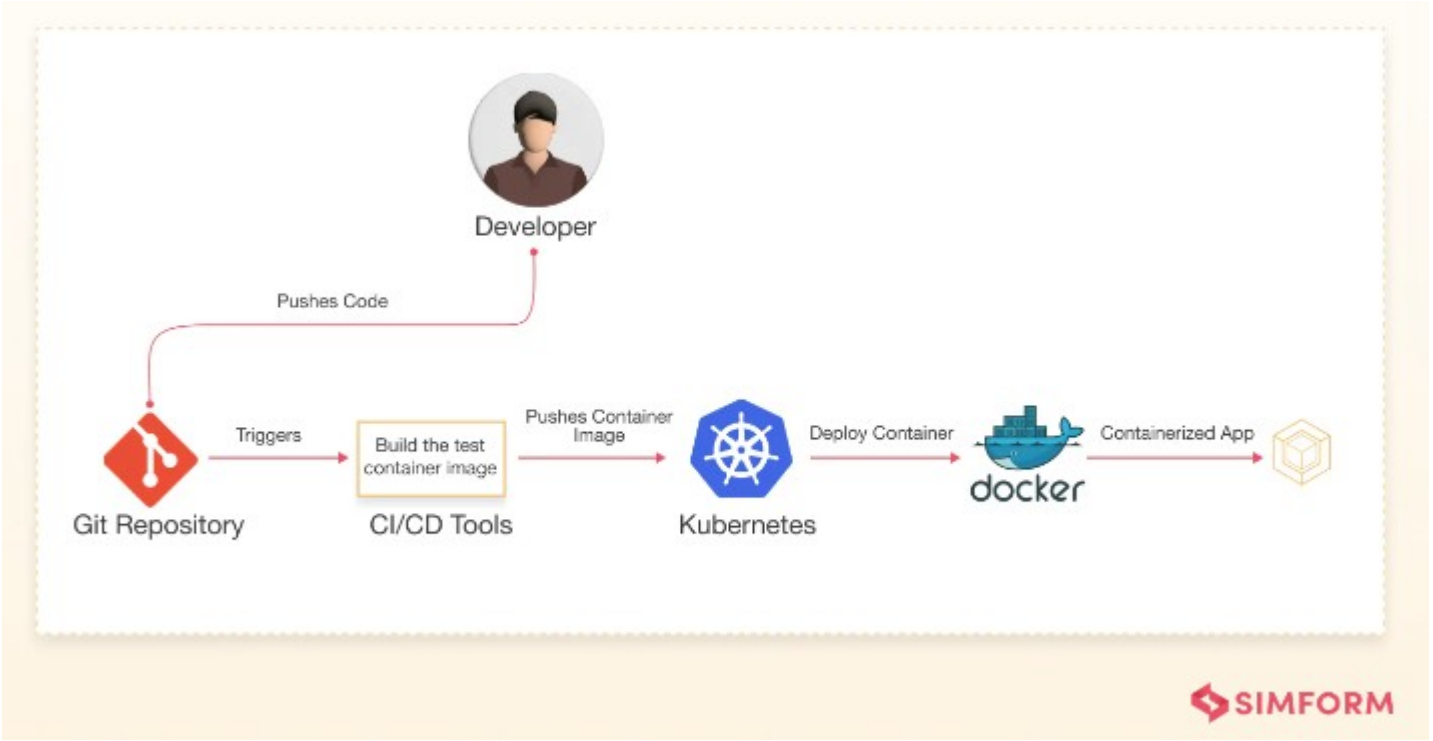
Solutions :

Conteneuriser un pipeline pour lui permettre de fonctionner indépendamment avec des versions différentes de langages.

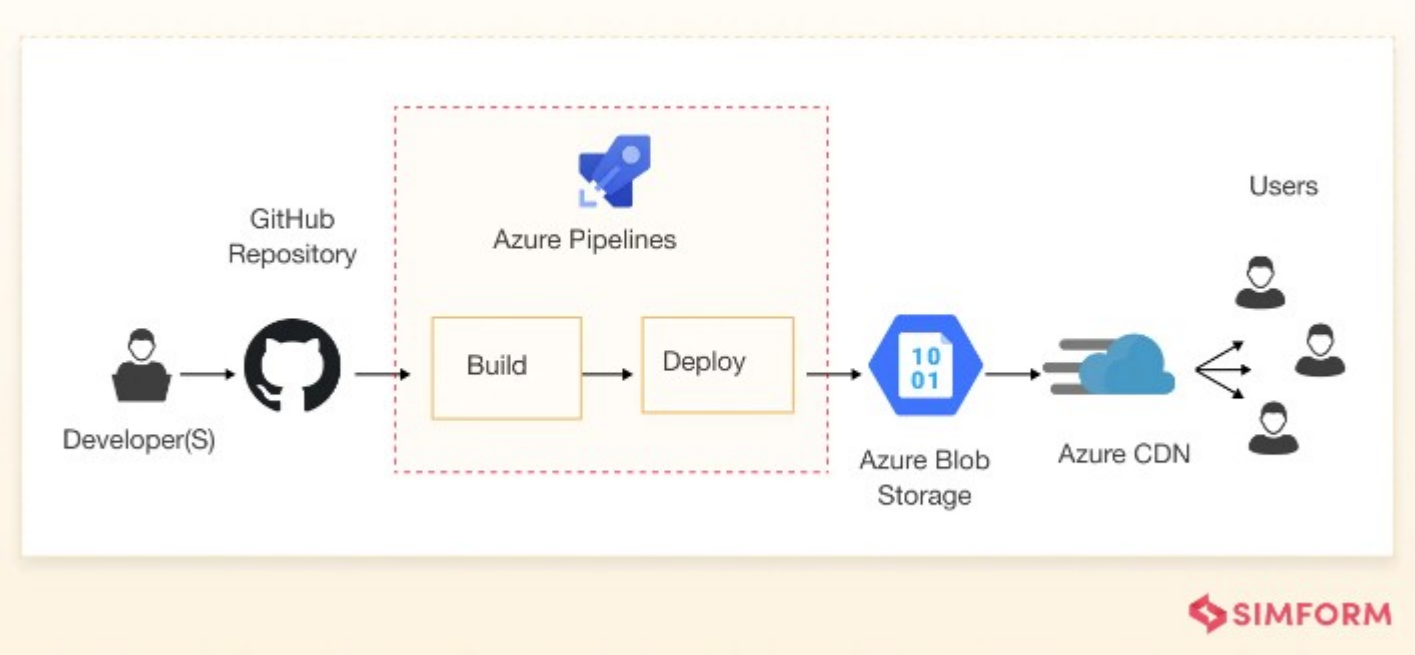
Utilisez une stratégie malléable pour chaque intégration et déploiement . les déclencheurs peuvent contenir des informations telles que des métadonnées ou un contexte, ce qui permet à un pipeline de fonctionner en conséquence

Adapter une stratégie de diffusion Canari qui permet de diffuser et de tester les nouveaux logiciels d'abord auprès d'un sous groupe d'utilisateurs sélectionnés.

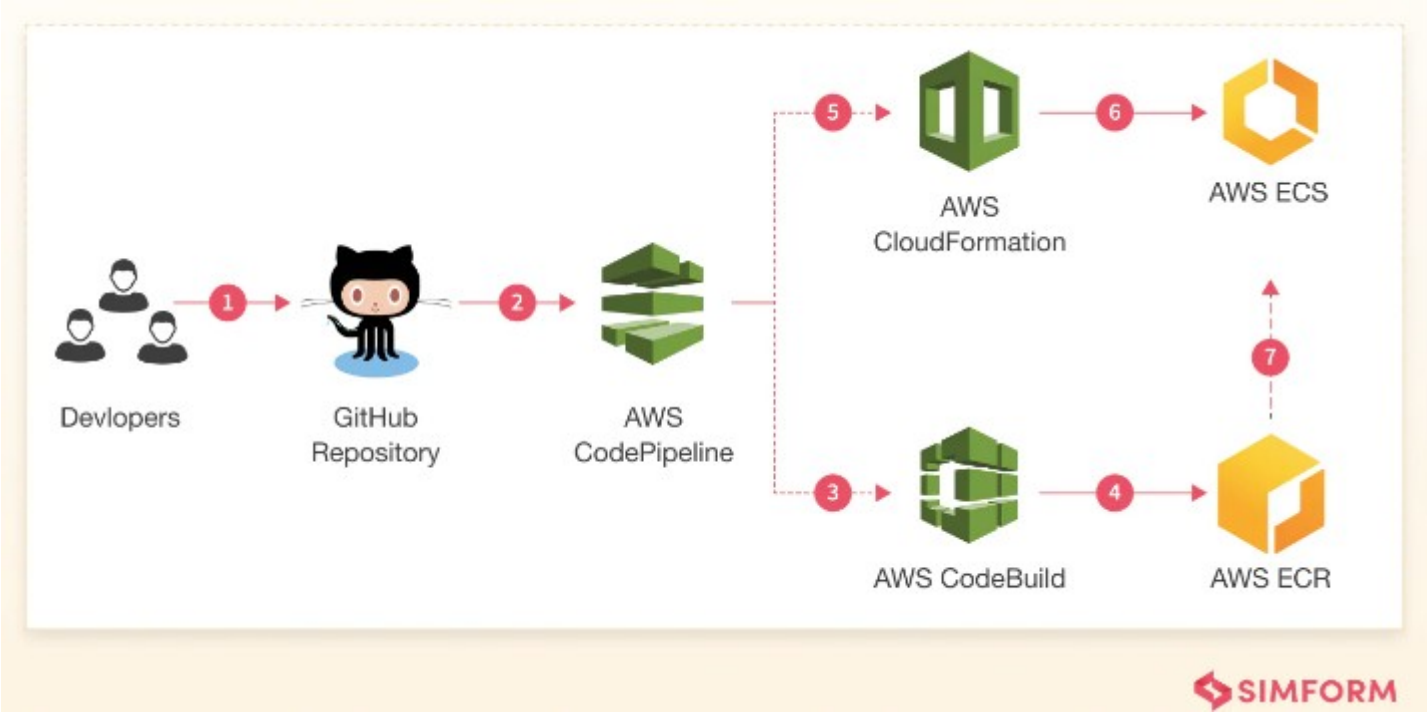
Exemples


Developer
Pushes Code
Triggers
Build the test container image
Pushes Container Image
Deploy Container
Containerized App
Git Repository
CI/CD Tools
Kubernetes
docker
SIMFORM

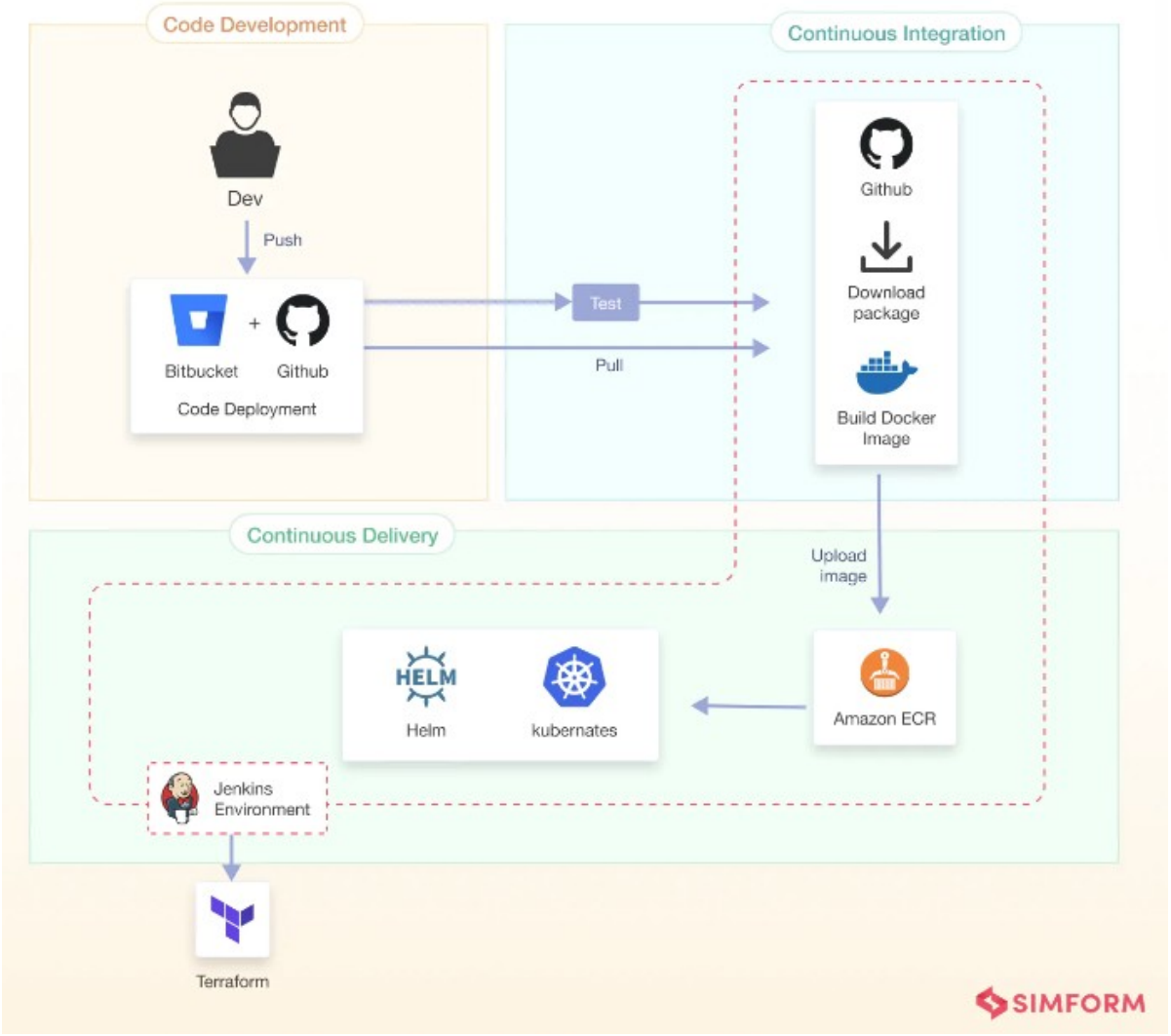
Exemples


GitHub
Repository
Azure Pipelines
Users
Developer(S)
Build
Deploy
10
01
Azure Blob
Storage
Azure CDN
SIMFORM

Exemples


Devlopers
1
GitHub
Repository
2
AWS
CodePipeline
5
AWS
CloudFormation
6
AWS ECS
3
AWS CodeBuild
4
AWS ECR
7
SIMFORM

Exemples


Code Development
Continuous Integration
Dev
Push
Bitbucket
Github
Code Deployment
Test
Pull
Github
Download package
Build Docker Image
Continuous Delivery
Upload image
HELM
Helm
kubernates
Amazon ECR
Jenkins Environment
Terraform
SIMFORM

Deploiements - type Canary

Utiliser une stratégie de déploiement : Canary

est un état de déploiement progressif d'une application qui répartit le trafic entre une version déjà déployée et un e nouvelle version, en la déployant auprès d'un sous-ensemble d'utilisateurs avant de la déployer complètement

On peut avoir différents niveaux de phase Canary

canary 25%

canary 50%

canary 75%

....

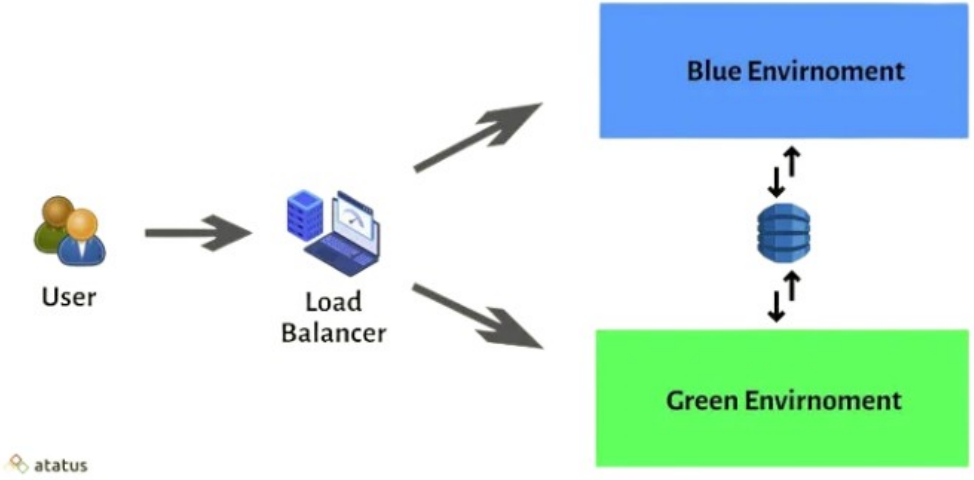
GKE/Anthos

Déploiement progressif (ou Rolling)

Comme le déploiement Canary, il s'agit d'échelonner les modifications. Mais ici, les modifications sont implémentées sur des serveurs ou instances. A l'inverse, avec le déploiement Canary DevOps, les nouvelles fonctionnalités sont directement mises à dispositions de certains utilisateurs.

Déploiement Blue/Green

Deux infrastructures d'hébergement sont programmés.L'une héberge la version de production de l'application (Blue) alors que l'autre est mise en réserve (Green). Cette dernière permet de déployer la nouvelle version et de réaliser les différents tests. lorsque les fonctionnalités sont validées , la mise à jour est déployée au sein de l'infrastructure Blue.



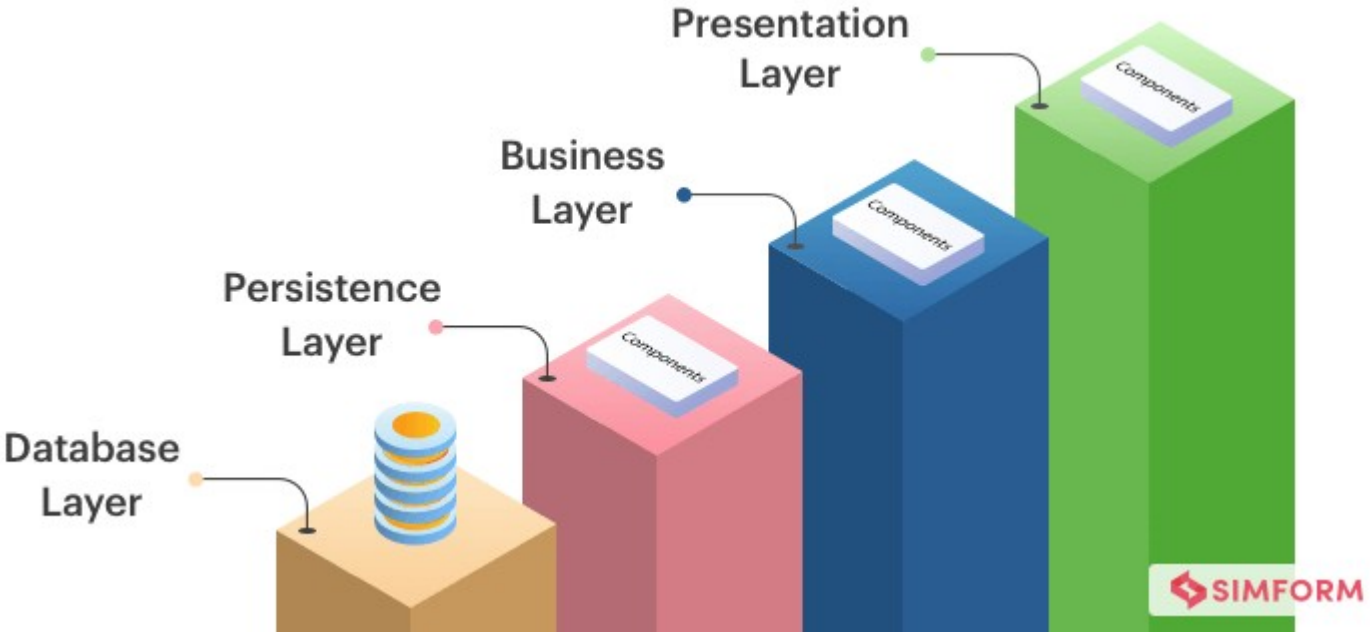
A/B Testing

Pour tester deux versions et sélectionner celle qui est la meilleure.

Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

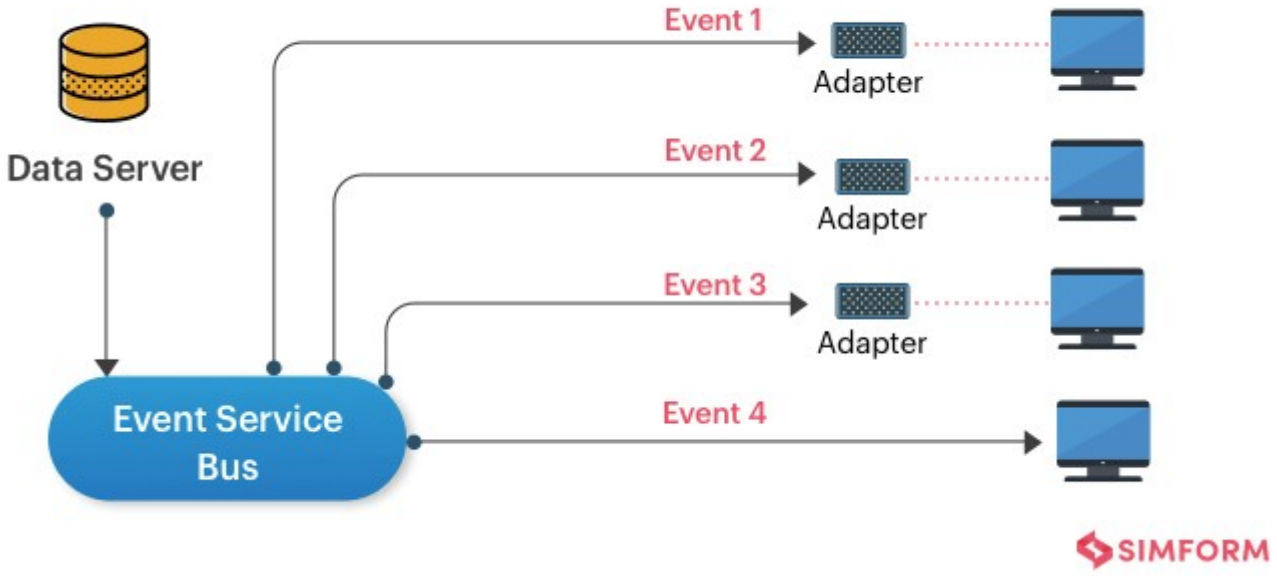
Layered Architecture Pattern



Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

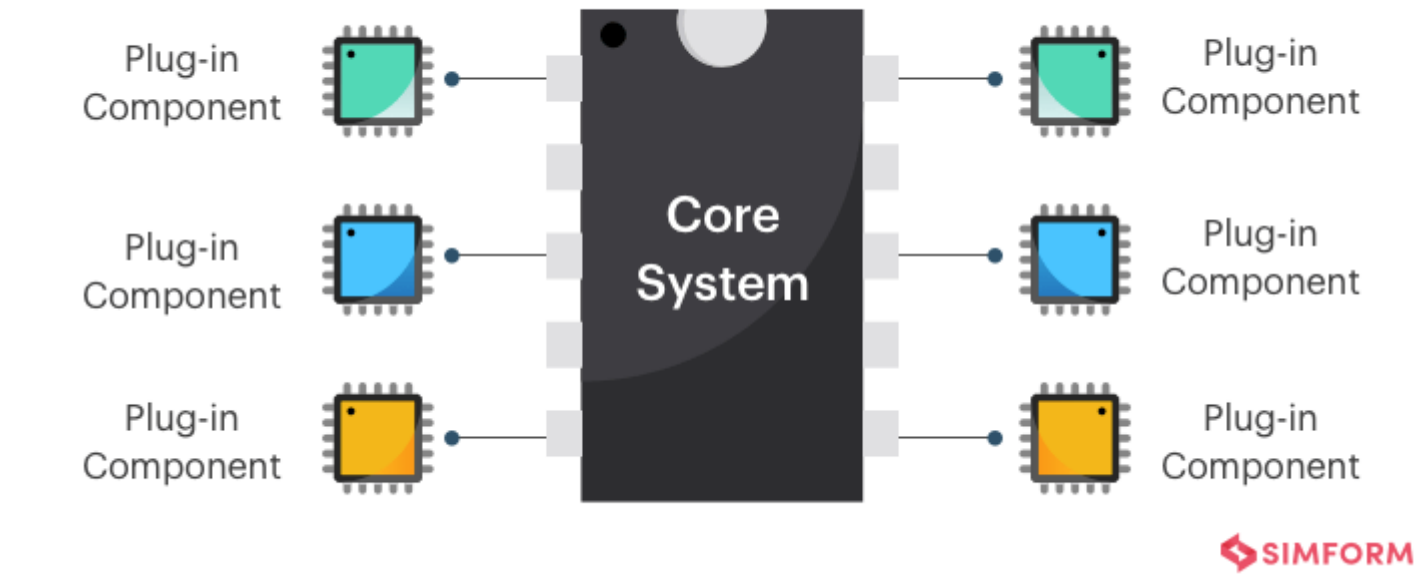
Event-driver Architecture Pattern



Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

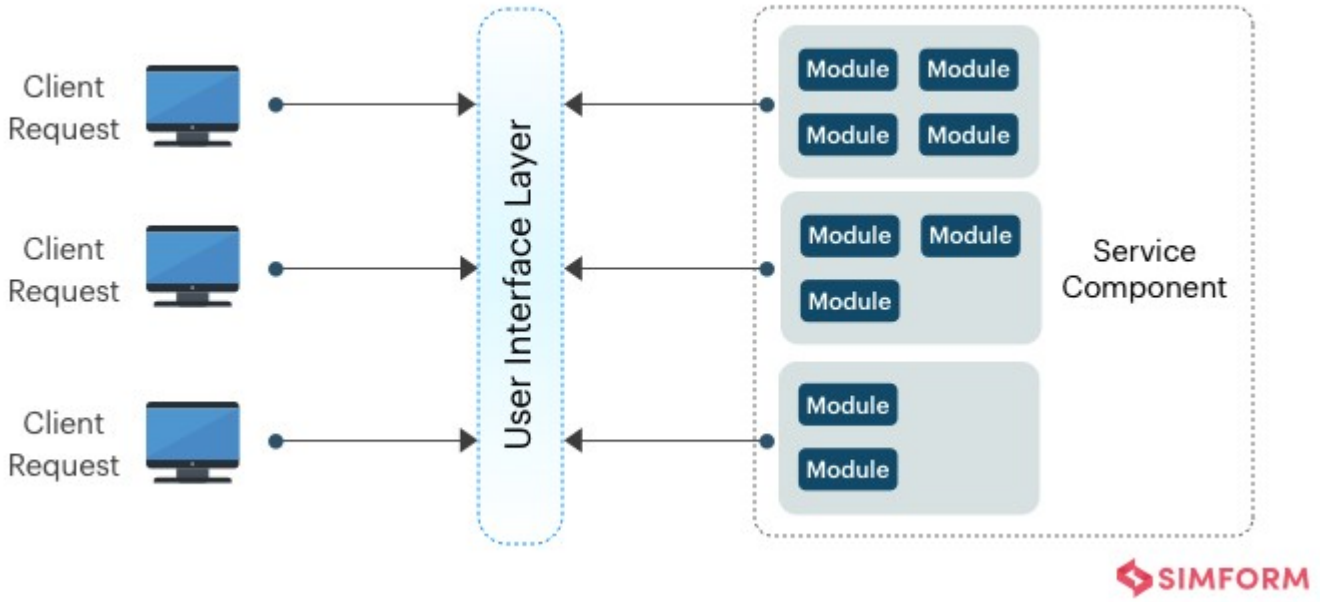
MicroKernel Architecture Pattern



Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

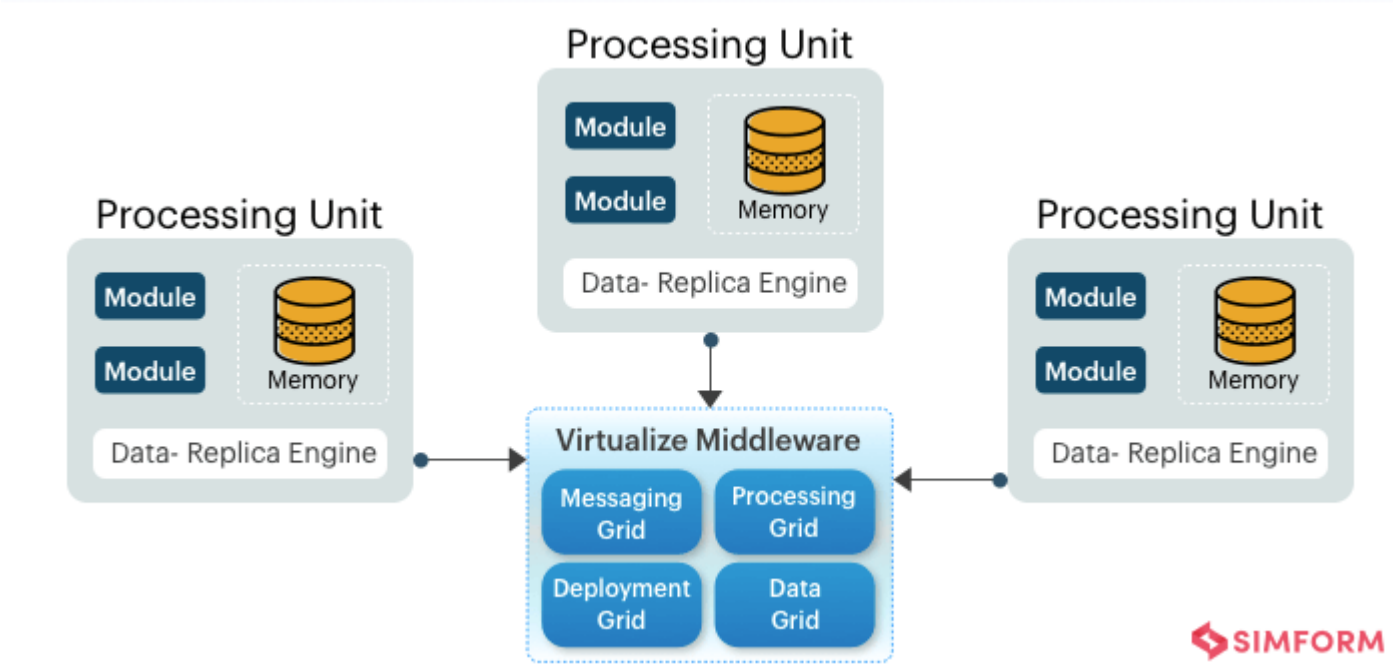
Microservices Architecture Pattern



Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

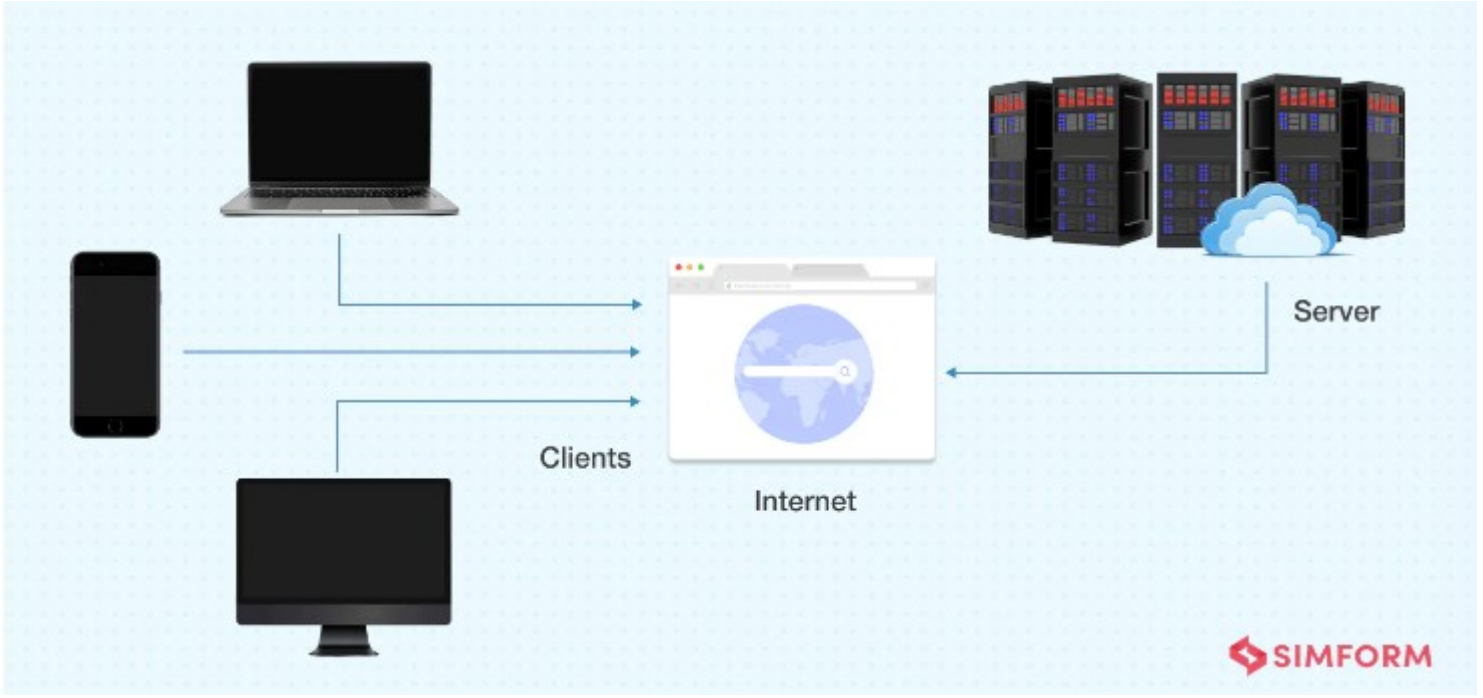
Space-based Architecture Pattern



Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

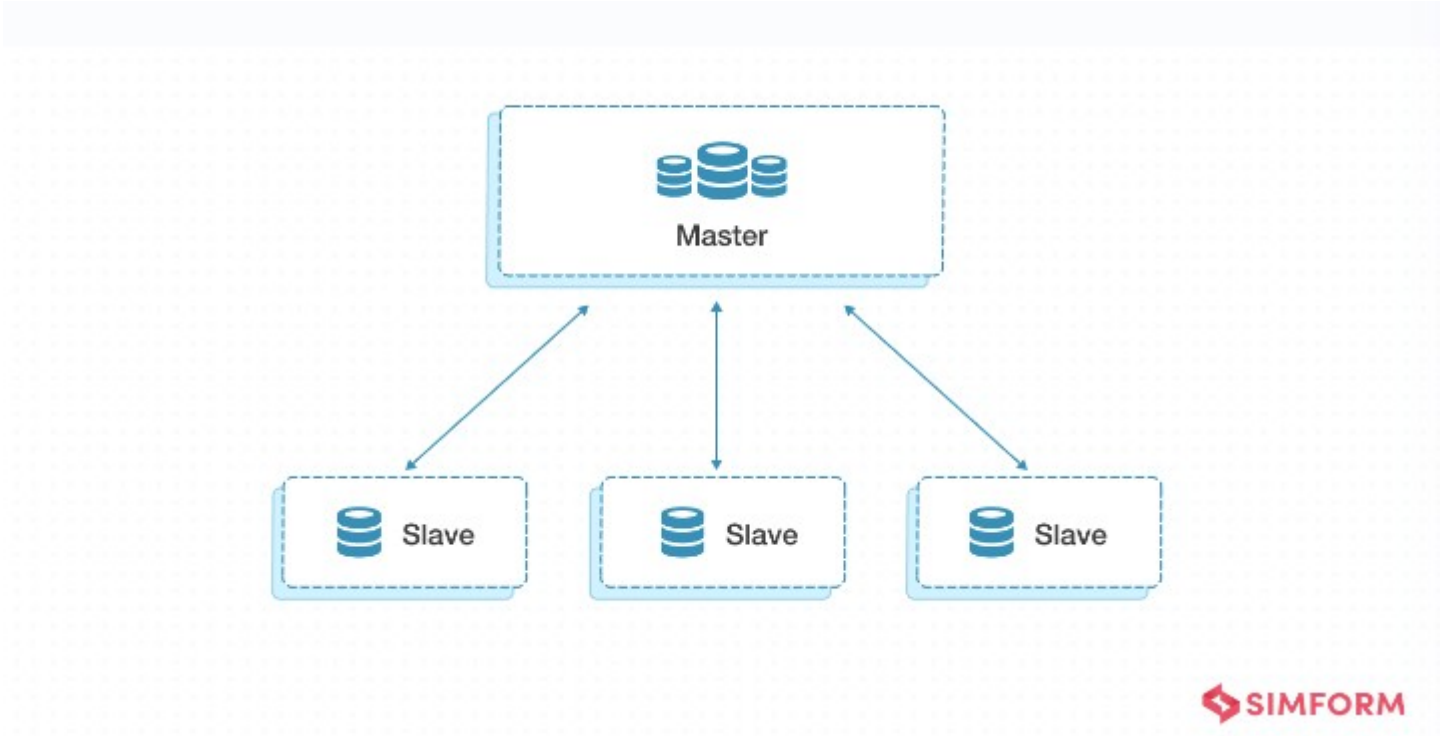
Client-Server Architecture Pattern



Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

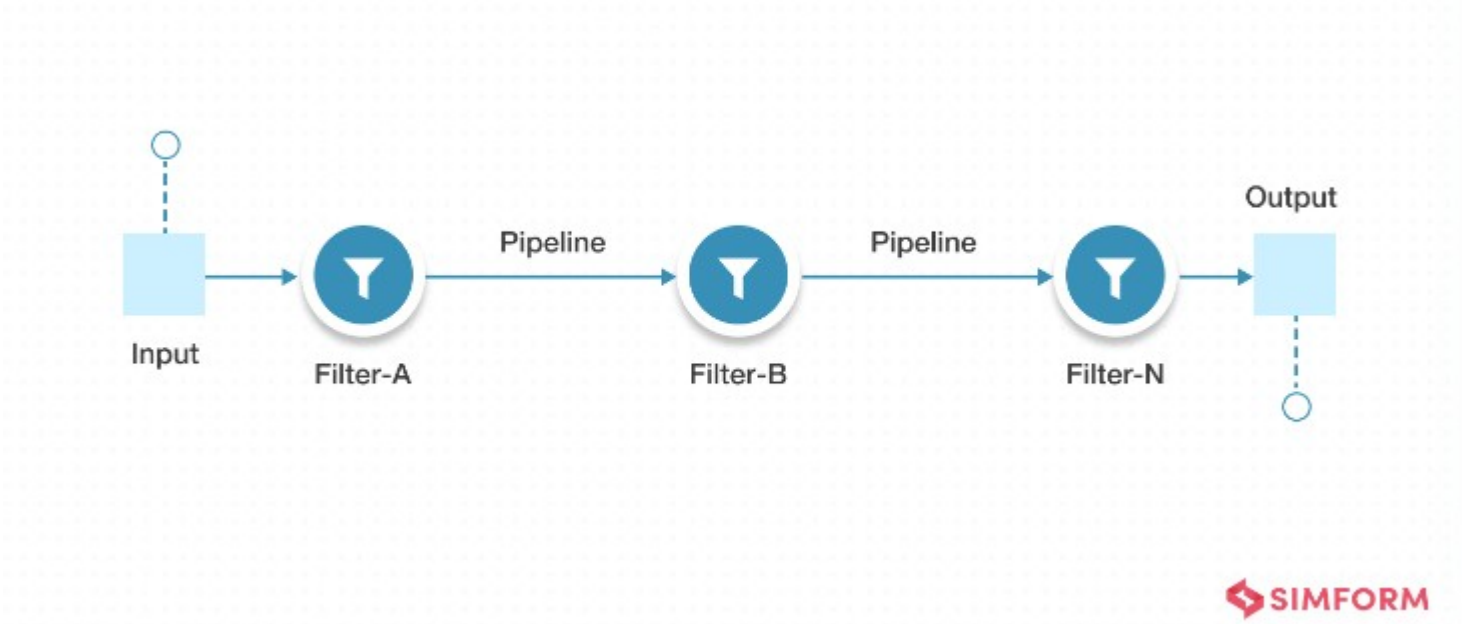
Master-Slaves Architecture Pattern



Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

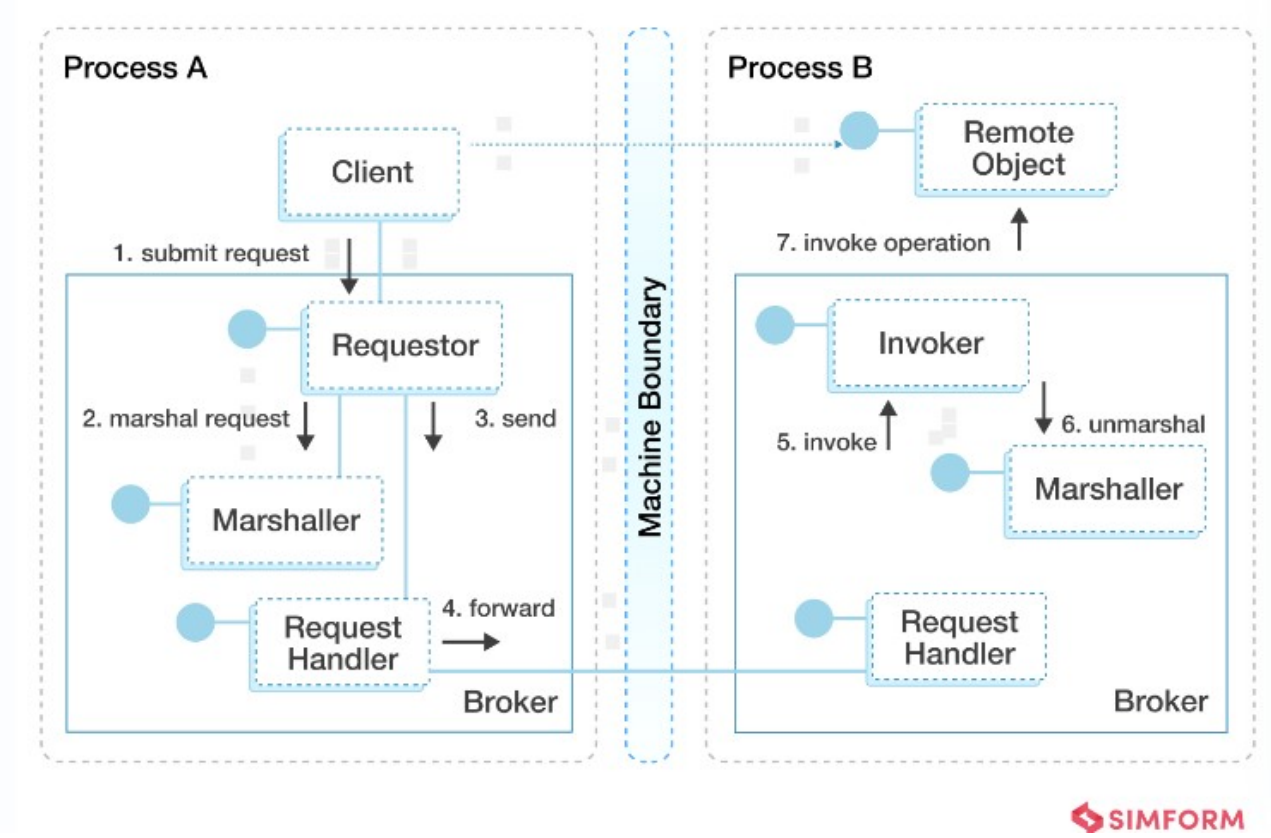
Pipe-filter Architecture Pattern



Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

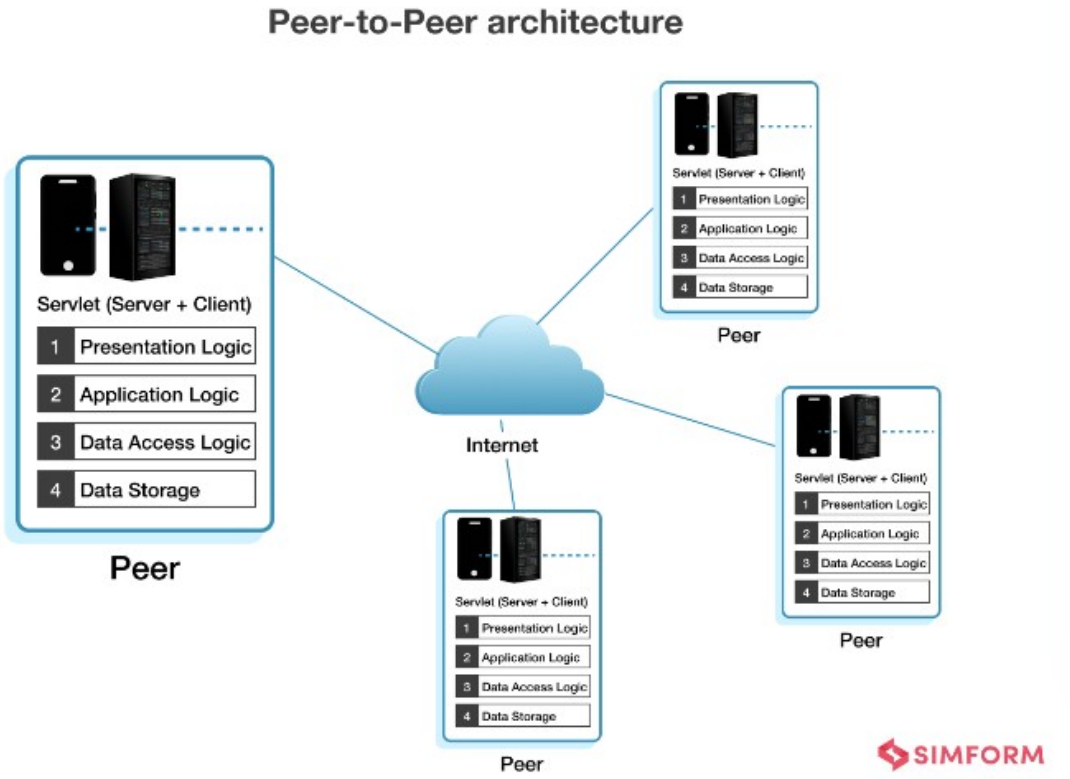
Broker Architecture Pattern



Software architecture patterns

https://www.simform.com/blog/software-architecture-patterns/

Peer-to-peer Architecture Pattern



Design Principles : S.O.L.I.D

Tawfik Nouri
Lead QA / Expert Automatisation des Tests | BDD | CI/CD | Contract Testing

Acronyme inventé par Michel Feathers à partir des principes de programmation orienté objet identifiés par Robert Cecil Martin, principes visant à rendre le code plus lisible, facile à maintenir, extensible, réutilisable et sans répétition.

Single Responsability Principle

Open/Closed Principle

Liskov Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle

Design Principles : S.O.L.I.D

Single Responsability Principle

==> stipule qu'une seule classe ne devrait avoir qu'une seule raison de changer et une seule responsabilité.

Design Principles : S.O.L.I.D

Open/Closed Principle

==> affirme qu'une classe/modules/fonctions doit être ouverte à l'extension mais fermée à la modification. C'est à dire que l'on doit pouvoir ajouter une nouvelle fonctionnalité sans changer le code existant.

==> Héritage et interface

Design Principles : S.O.L.I.D

Liskov Substitution Principle

==> ce principe affirme que les classes dérivées doivent être substituables à leurs classes de base

Design Principles : S.O.L.I.D

Interface Segregation Principle

==> ce principe affirme qu'il vaut mieux plisueirs interfaces spécifiques à une classe qu'une grosse interface générique. Il ne faut pas obliger à implémenter des méthodes que l'on ne veut pas.

Design Principles : S.O.L.I.D

Dependency Inversion Principle

==> ce principe est basé sur les affirmations suivantes :

- les modules de haut-niveau ne devraient pas dépendre des modules de bas niveau, tous deux devraient dépendre d'abstraction
- les abstractions ne devraient pas dépendre des détails, ce sont les détails qui devraient dépendre des abstractions.

Introduction

Aujourd'hui les applications doivent être produites et déployées en continu

- amélioration de la qualité logiciel
- réduire les risques

A l'ère du cloud

- les solutions logicielles doivent être
  - évolutives, disponibles, hyper-performantes et à moindre coût

Devops permet aux équipes de développements et d'infrastructure d'être plus réactives

DevSecOps

Approche de développement de logiciel qui combine développement, sécurité et opérationnel dans un processus unifié et collaboratif.

Intégrer les principes et la pratique sécuritaire dans toutes les étapes d'un projet de développement.

De la conception au déploiement

Comment sécuriser le pipeline CI/CD
(GitGuardian)

Suite à une attaque récente de la partie CI avec circleCI.

Si quelqu'un peut rentrer dans cette aprtie , il peut compromettre toute la chaîne de production des applicatifs.

Rappel du principe

Les Dév. posent leur code vers des hôtes référentiels comme (github, gitlab, Azure DevOps, Bitbucket)

Qui démarrent les processus automatisés de construction de l'application, de tests et enfin de déploiement de ce code en production.

Alors que les pipelines CI/CD permettent aux équipes de travailler plus rapidement et avec moins de chance d'erreurs manuelles, ils présentent une nouvelle surface d'attaque.

L'un des meilleurs moyens de s'assurer que des aspects sécurités est de verrouiller et de contrôler l'accès aux outils et ressources (on pourra également conserver la trace des différents accès par les logs) par mot de passe, clé d'accès et autres mécanimes de contrôle.

Comment sécuriser le pipeline CI/CD
(GitGuardian)

Il est pratique d'accorder l'accès à tous les membres des équipes DevOps : devs et autres ingénieurs qui accèdent à tout.

=> Mauvaise idée

Car un seul compte piraté accède à tout ????

Reommandation : authentification unique ou utiliser RBAC (accès basé sur les rôles)

Ceci est valide également pour les accès machines (services et outils tiers)

Lorsque vous déployez un élément de votre pipeline CI/CD, vous devez savoir exactement quels systèmes et services enverront des demandes et où iront ces réponses.

Lorsque vous utilisez des conteneurs, assurez-vous d'utiliser des authentifications pour vérifier l'identité de la machine.

De plus, tout ce qui n'est plus utile doit être détruit (création temporaire)

Comment sécuriser le pipeline CI/CD
(GitGuardian)

Ne déposez pas de secret dans votre pipeline CI

- souvent solution simple, mais qui doit être complètement proscrite
- Automatiser la rotation des secrets => AWS, Gooclecloud et Azure Devops proposent des services dans ce sens

Surveillez activement les comportements suspects.

Solution de HoneyTokens ou canaryToken pour la détection des intrusions.

Comment sécuriser le pipeline CI/CD
(GitGuardian)

Voir comment sécuriser votre SCM avec GitGuardian Honeytokens

(Actuellement est disponible que des clés AWS d'autres sont en cours de développement.)

les honeykeys sont des pièges numériques configurés pour attirer les cyber-attaquants et dès qu'ils vont les utiliser ils seront détectés.

C'est une sorte de système d'alarme

Dès que l'honeykey est utilisé, il y a une alerte qui est remontée.... On les copie dans les fichiers scripts a des endroits spécifiques que l'on veut surveiller.

OWASP : TOP 10 Ci/CD Security risks

CI/CD SEC-1 : Insufficient Flow Control Management

CI/CD SEC-2 : Inadequate Identity and Access Management

CI/CD SEC-3 : Dependency Chain Abuse

CI/CD SEC-4 : Poisoned Pipeline Execution (PPE)

CI/CD SEC-5 : Insufficient PBAC (Pipeline based Access Controls)

CI/CD SEC-6 : Insufficient Credential Hygiene

CI/CD SEC-7 : Insecure System Configuration

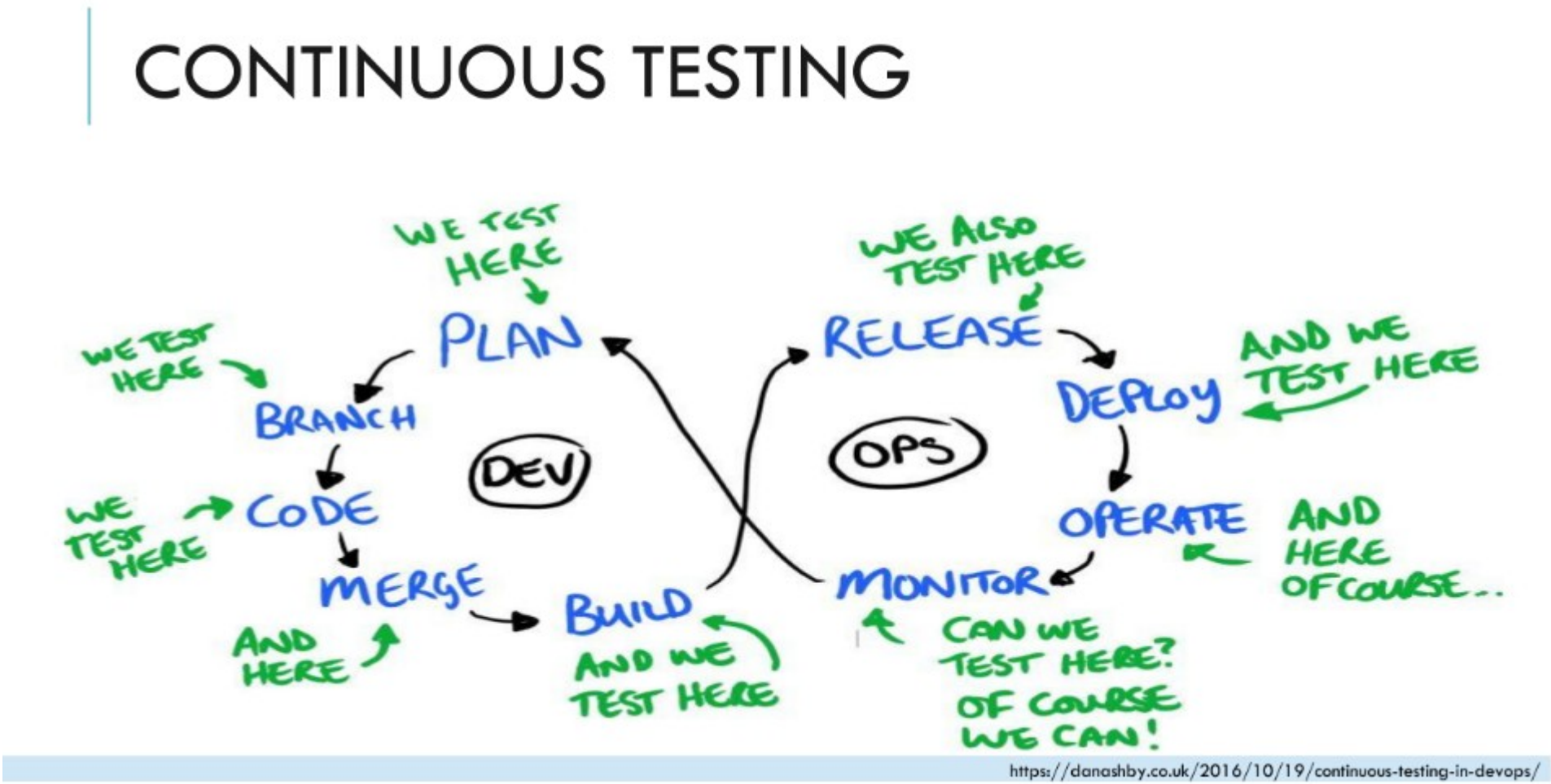
CI/CD SEC-8 : Ungoverned Usage of 3rd Party Services

CI/CD SEC-9 : Improper Artifact Integrity Validation

CI/CD SEC-10 : Insufficient Logging and Visibility

Les tests

https://synapsys-groupe.com/blog/le-quoi-le-devops/



Ingénierie des tests
(http://gayerie.dev/docs/testing)

Notion de tests :

- permet de vérifier l'adéquation d'un système informatique à un fonctionnement attendu
  - Notion défaut : écart entre le constaté et l'attendu
- activité déterminante et incontournable
- activités et spécialités impliquées :
  - la personne en charge de définir les procédures à réaliser
  - développeur en charge de réaliser le ou les programmes d'automatisation des tests
  - les personnes impliques dans l'organisation et la gestion des activités de tests et toutes les personnes impliquées dans la conception et la définition du système.
  - le testeur

L'ingénierie des tests implique donc des aspects fonctionnels, techniques et organisationnels.

Ingénierie des tests
Stratégie de test et types de tests

Les tests sont conditionnés par la stratégie de test que l'on souhaite mettre en place

Stratégie dépendante du budget

Il existe différents types de tests

Tests unitaires, tests d'intégration, tests de sécurité, tests de performances, tests de robustesse et tests d'acceptation.

Cette liste n'est pas exhaustive car des tests complémentaires sont possibles en fonction du défaut à détecter.

Ingénierie des tests

Tests :

tests unitaires : testent une partie

tests d'intégration : testent le système sous une plateforme proche de la plateforme cible ;

tests de performance : testent les performances du système afin de s'assurer qu'il réponde à certains critères de réactivité.

tests de sécurité : testent que l'application ne contient pas de failles de sécurité connues (injection de code, attache XSS ...)

tests de robustesse ou de stress : testent le comportement de l'application aux limites des ressources disponibles (mémoire, CPU..) --> JMeter, Gatling, Dynatrace

tests d'acceptation : testent le système afin de s'assurer qu'il est conforme aux attentes des utilsateurs (test de recette)

On a aussi les tests de non régression :

Ce ne sont pas des tests spécifiques, mais tous les types de tests peuvent servir de tests de non régression.

Ingénierie des tests

Les 7 principes de test formalises par l'ISTQB (International Software Testing Qualification Board)

- les tests montrent la présence d'un défaut mais ne permettent pas de prouver l'absence de défaut. Permettent de réduire la probabilité qu'il subsiste des défauts.

- Les tests exhaustifs sont impossibles

- tester au plus tôt : car la correction tardive sera plus complexe et plus coûteuse

- Regroupement des défauts : les défauts d'un système tendent à être regroupes sur des composants ou des fonctionnalités

- le paradoxe du pesticide : si le tests ne trouvent pas de défaut, cela peut signifier que les tests tendent à mettre en valeur une catégorie particulières de défaut ignorant ainsi les autres (analogie aux pesticides)

- la stratégie de tests dépend du contexte

- l'illusion d'absence d'erreur

Ingénierie des tests

Les tests d'acceptation :

La qualification correspond à toutes les activités réalisées pour vérifier qu'un système logiciel est conforme aux attentes.

s'inscrivent dans une démarche de qualification

description formelle du comportement d'un logiciel par des scénarios de tests

s'inscrivent dans un cycle de développement logiciel

méthodes de gestion de projet

méthodes agile

Méthode de gestion de projet

Déf par l'iso : processus unique qui consiste en un ensemble d'activités coordonnées et maîtrisées, comportant des dates de début et de fin, entrepris dans le but d'atteindre un objectif conforme à des exigences spécifiées telles que les contraintes de délais, de coûts et de ressources.

Notion d'exigence : forme d'expression de besoins.

- fonctionnelles, disponibilités, performance, sécurité ....
- possède certaines caractéristiques
  - non ambiguë
  - elle est délimitée
  - elle doit être testable

Plusieurs phases dans cette méthode :

- phase d'expression de besoins, phase de conception, phase de réalisation, phase de test, phase de livraison

Méthode de gestion de projet

Notion de cas test :

Désigne une interaction avec le système destinée à produire un résultat attendu

Il est décrit :

par un titre

liste d'étapes :

chaque étape doit correspondre à une action de l'utilisateur et s'accompagne obligatoirement d'un résultat attendu (si possible qu'un seul). le résultat doit être explicité clairement et ne doit par être interprétable. Un cas test peu se rapporter à un ou à plusieurs exigences.

Rédaction d'un cas test

cas test : afficher son profil utilisateur

Etape : sélectionner le menu utilisateur

Résultat attendu : le menu s'affiche et contient l'entrée profil

Etape : sélectionner l'entrée profil

Résultat attendu : le détail du profil s'affiche dans une popup avec le nom ($nom), le prénom ($prenom) et la photo, les boutons modifier et fermer sont visibles et cliquables.

....

La façon d'écrire les cas test dépend de la connaissance supposée qu'auront les testeurs de l'application

Méthode de gestion de projet

Scenario de test :

Un scénario de tests est un regroupement de cas tests, il représente la réalisation d'une fonctionnalité complète.

Un ensemble de scénarios permet de s'assurer qu'une fonctionnalité fournie est conforme à l'ensemble des exigences qui la spécifient. Chaque scénario fournit son jeu de données particuliers pour les paramètres décrits par les cas tests. un scénario peut également s'accompagner d'un pré-requis sur l'état nécessaire du système :

Exemple d'un scnéario : modification d'un profil utilisateur

Pré-requis : il existe un profil utilisateur Yvan Stroppa

Paramètres:$nom : Stroppa, $prenom : Yvan, $nouveau_nom : sliman, $nouveau_prenom : Layth

cas de test : affiche profil utilisateur

cas de test : éditer le profil utilisateur

cas de test : sauver le profil utilisateur

....

Méthode de gestion de projet

La matrice de traçabilité des exigences

est un tableau à deux entrées : exigences et scénarios de tests

A l'intersection d'une ligne et d'une colonne nous pouvons indiquer si l'exigence est couverte par de scénario.

Rapport de tests :

La phase de recette est constituée de plusieurs sessions de tests au cours desquels on doit réaliser les tests décrits dans le cahier de tests :

Les résultats des tests est conservé sous forme de rapport de test.

Méthodes agiles

Les plus connues : Scrum et eXtreme Programming (XP)

=> Elles ne reposent pas sur la notion de projet et ne se présentent pas comme une succession de phases situées entre un début et une fin.

=> Elles sont itératives et incrémentales.

Le travail en équipe de dév est découpé en bloc de temps fixé (appelé sprint dans Scrum et Itération dans XP) ne dépassent pas quelques semaines. A la fin de chaque bloc de temps, l'équipe fournit un produit correspondant au produit précédant plus ce qui a été rajouté par le travail de l'équipe

==> nouvelle fonctionnalités (développement incrémental)

==> amélioration de fonctionnalités existantes (développement itératif)

Dans cet approche, l'expression de besoins n'est pas nécessairement entièrement collectée au lancement des dévs. Cette approche insiste sur l capacité à améliorer le produit et à s'adapter au fur tet à mesure des développements.

Méthodes agiles

la notion d'exigence est alors peu applicable et on la remplace entre autres pour l'utilisateur de User Stories. Une User Storie décrit une interaction du point de vue d'un utilisateur. un produit est défini par l'ensemble des User Stories qui évolue au cours des devs. Chaque User Story doit être testable cad fournir des critères d'acceptation.

Dans cette approche, les tests d'acceptation s'inscrivent dans un processus de dév différent. Les test sot rédigés lorsque les développeurs doivent implémenter la partie du système répondant à une User Story. il n'est pas question de vérifier un écart avec une exigence mais de préciser les objectifs de développement.

L'eXtreme Programming a proposé l'approche ATTD (Acceptance Test Driven Developement). Il s'agit de rédiger les tests d'acceptation avant le développement d'une User Story de manière à guider les développements au plus près des attentes des utilisateurs.

Méthodes agiles : BDD

Le Behaviour Driven Development proposé par Dan North est en fait une évolution de la notion de test d'acceptation en agilité. Il se fonde sur le constat qu'il est plus pertinent de raisonner en terme de description de comportement du système qu'en terme de test. Le test n'est qu'un formalisme pour mesurer les écarts entre une expression de besoins et un produit. L'approche BDD oblige à revoir la notion de test et d'expression de besoins. Le BDD s'appuie sur les méthodes agiles et reprend la notion de critères d'acceptation d'une User Story. les critères devraient correspondre à des exemples d'utilisation du système. En utilisant un formalisme pour la rédaction de ces exemples.

Ils deviennent u support à la communication entre les intervenants qu'ils soient dev, utilisateur ou experts fonctionnels . Dan North propose de suivre un formalisme simple : Given, When Then

Given some : initial content	Etant donné un contexte initial
When : an event occurs	Quand : un événement survient
Then : ensure somme outcomes	Alors :on s'assure d'un certain résultat

Plusieurs farmeworks : Fitness, Jbehave et Cucumber.

Testing the Code : tools

Outils de tests dédiés aux différentes environnements et IDE utilisés.

- JUnit (4, 5)
- pytest
- Selenium (IDE, WebDriver, Grid)
- Playwright : npx playwright test --ui (utilisation également en mode commande : npx playwright test sous macOS)
- Cucumber : notion de BDD (Behaviour Driven Development)

Outils de tests JS :

- MochaJS, jasmine, Jest, Cypres.io, puppeteer, chaiJS, Qunit ....

Outils de tests : Selenium
(https://www.selenium.dev/)

Les outils fournis par Selenium

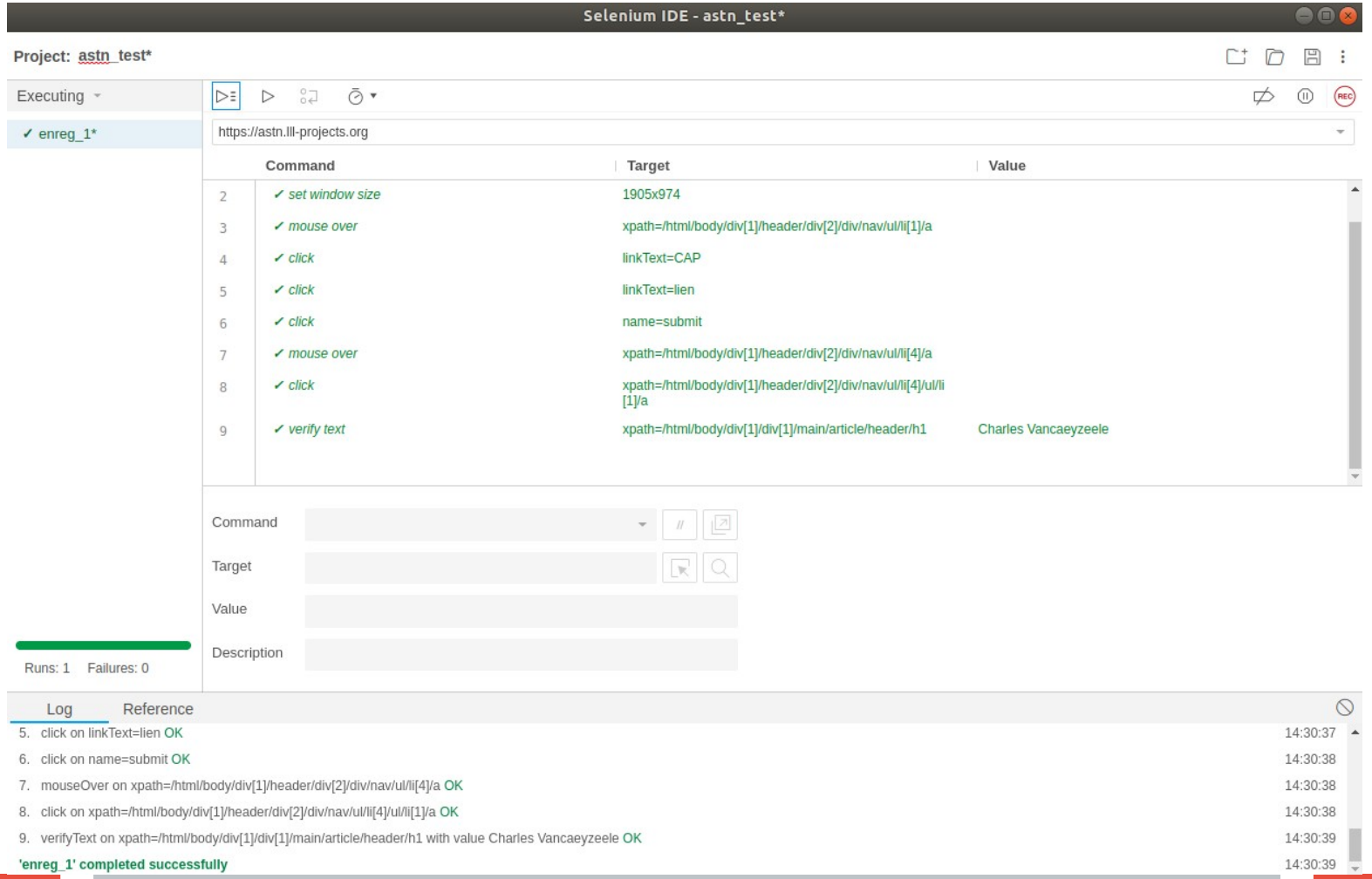
Selenium Ide : se présente comme un addon pour navigateur web ; Il permet d'enregistrer des scenarios de test à partir des actions de l'utilisateur.

Web Driver : est un module qui permet d'interagir par programmation avec un navigateur web. Selenium fournit une API disponible pour, plusieurs langages (java, C#, Python, javascript , Ruby) afin de communiquer avec un WebDriver. Offre une alternative à Selenium IDE.

Grid : permet d'exécuter les test développés à partir de l'API WebDriver sur plusieurs types de navigateurs Web et même sur plusieurs système d'exploitation.

Outils de tests : Selenium
(https://www.selenium.dev/)

Installation de l'addon dans chrome :



Outils de tests : playwright

(https://playwright.dev/)

Outils de tests : pytest

Outils de tests : Junit

Outils de tests : Cucumber
(https://cucumber.io/)

Les orchestrateurs

Swarm : Docker

https://docs.docker.com/engine/swarm/key-concepts/

Kubernetes :

1,27

CNCF

OpenShift Container Platform :

4,12

RedHat --> IBM

....

Orchestrateur : Cluster Swarm

Cluster natif et outils d'orchestration pour les conteneurs Docker. Partie intégrante de l'éco-système de Docker et apporte au moteur Docker la coordination de multiples instances de docker sur plusieurs machines. Comprend des nodes Manager et Worker. Permet de faire fonctionner des conteneurs de haute disponibilité sans la complexité de Kubernetes.

Vous pouvez utiliser Docker Cli, Docker-compose et docker API pour gérer les conteneurs et déployer des applications sur ce cluster.

Orchestrateur : Cluster Swarm

notion de services

docker service create nginx

```
$ docker service create --name my_web \
        --replicas 3 \
        --publish published=8080,target=80 \
        nginx
$ curl localhost:8080

$ docker service create \
  --mode global \
  --publish mode=host,target=80,published=8080 \
  --name=nginx \
  nginx:latest
```

Orchestrateur : Cluster Swarm

Notion de stack : un ensemble de conteneurs que l'on va déployer en même temps sur un même worker.

Orchestrateur : Cluster Swarm

Commandes très simples :

docker swarm init

docker node list ou ls

Avantages :

facile, service discovery, qui assigne automatiquement un nom DNS pour chaque service de SWARM, simplifie les communications inter-services, étend les fonctionnalités de load-balacing. Offre également la distributions des tasks à travers les workers nodes.

Inconvénients :

n'a pas de système de provisionnement de stockage partagé n'a pas le même niveau d'offre que Kubernetes. On peut utiliser GlusterFS

Orchestrateur : Kubernetes

Invention de google, maintenu par le Cloud Native Computing foundation CNCF. Contrairement à SWARM, Kubernetes est mieux équipé pour gérer des charges de travail complexes et une architecture avec des APIs.

Induit plusieurs Nodes, partitionnés entre Worker et Manager Nodes. il introduit plusieurs focntionnalités telles que automates scaling, load balancing, service discovery. Comme SWARM il n'a pas de stockage partagé intégré.

Avantage de Kubernetes

automatiquement scaling et load balancing, il est plus fort que Swarm dans cette gestion. Il a une architecture robuste et sécurisée plus sophistiquée qui nécessitent des protocoles de sécurité rigoureux. On peut implémenter des RBAC, des network policies ...

Inconvénients :

Plus complexe dans l'installation et nécessite une bonne connaissance de ces structures et des lignes de commandes.

Kubernetes services dispo sur le cloud

Cluster en ligne :

- Google Kubernetes Engine (GKE)
  - client gcloud
- Amazon Elastic kubernetes Service (EKS)
  - voir également ECS
- Azure Kubernetes Service (AKS)
- IBM Cloud Kubernetes Service
- DigitalOcean Kubernetes

Clusterless Container service

- AWS fargat
- Azure Container Instances
- Google Cloud Run

Kubernetes services dispo sur le cloud

Kubernetes installers

- kops
- kubespray
- kubeadm
- Rancher Kubern,etes engine
- Puppet Kubernetes Module

Multicloud Kubernetes clusters

- VMWare tanzu
- OpenShift
- Anthos
- version de base

Orchestrateur : Kubernetes

Plusieurs solutions possibles pour installer ce type de solution :

- k0s CNCF
- k3s Rancher
- K8s CNCF
- MiniKube
- MicroK8s

Pour K8S plusieurs configurations possibles avec podman, cri-o ou docker.

Rappel infrastructure Kubernetes

1.2.1 Orchestration efficace des conteneurs

Automatise le déploiement, la mise à l'échelle et la gestion des applications conteneurisées.

Optimise l'utilisation des ressources pour améliorer les performances et réaliser des économies.

Permet l'auto-réparation en remplaçant automatiquement les conteneurs ou nœuds défaillants.

1.2.2 Évolutivité et équilibrage de charge

Prend en charge la mise à l'échelle automatisée pour gérer différents niveaux de demande.

Fournit un équilibrage de charge intégré pour une répartition uniforme du trafic.

1.2.3 Stratégies de déploiement flexibles

Propose diverses stratégies de déploiement pour des mises à jour et des restaurations transparentes.

Permet aux versions Canary de tester les modifications avec un sous-ensemble d'utilisateurs.

1.2.4 Découverte de services et équilibrage de charge

Facilite la découverte de services internes basés sur DNS pour une communication transparente.

Équilibre automatiquement le trafic pour maintenir des performances constantes.

Rappel infrastructure Kubernetes

1.2.5 Gestion déclarative des configurations

Utilise des fichiers YAML déclaratifs pour définir l'état d'application souhaité.

Réduit la dérive de configuration et garantit des déploiements cohérents.

1.2.6 Communauté et écosystème forts

Possède une communauté dynamique et active pour le soutien et la collaboration.

Offre un riche écosystème d'outils, d'extensions et d'intégrations.

1.2.7 Prise en charge multi-environnements

Fonctionne dans divers environnements, y compris sur site et dans le cloud.

Permet un déploiement et une gestion cohérents des applications.

1.2.8 Extensibilité et personnalisation

Hautement extensible avec des API et des plugins pour des fonctionnalités personnalisées.

Permet d'adapter Kubernetes aux besoins organisationnels spécifiques.

Les commandes pour la solution de base

les outils :

- kubeadm
- kubelet
- kubectl

Figure 1: Kubernetes architecture

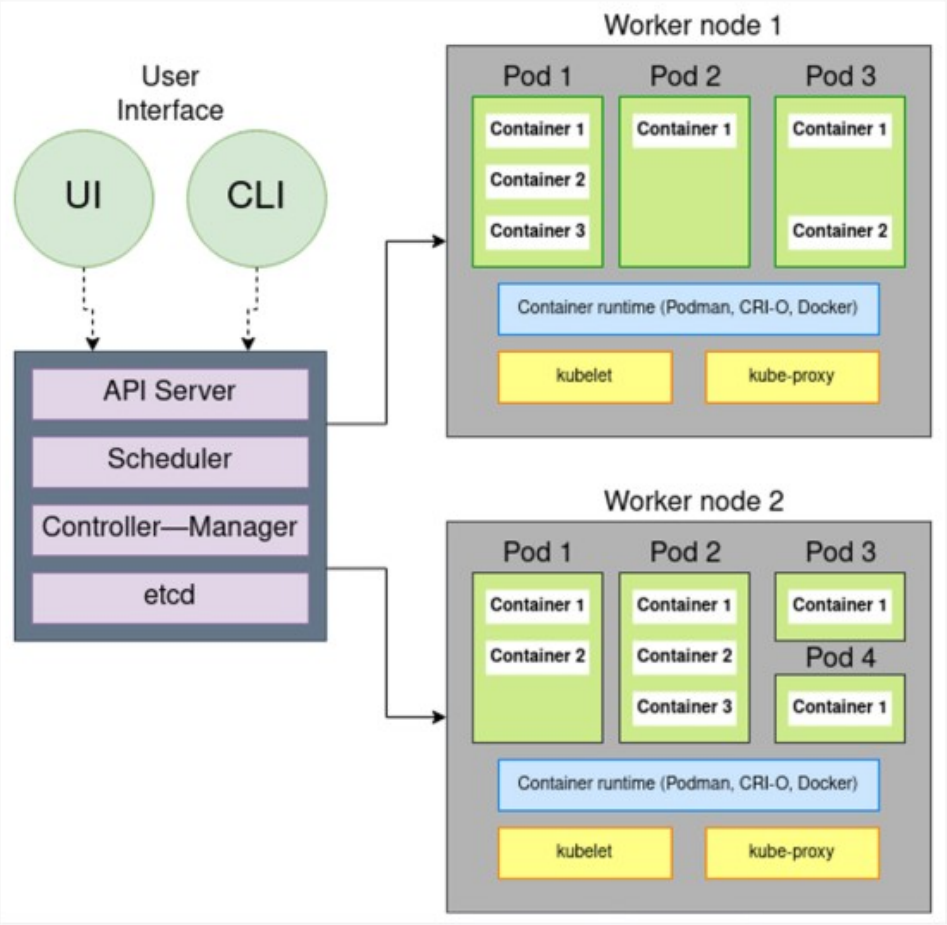


Image source: Nived Velayudhan, CC BY-SA 4.0

Kubernetes :

kube-apiserver : frontal du control-plan --> APi request

etcd : (key-value store) database où K8S stocke toutes ses informations quels noeuds, quelles ressources ...

kube-scheduler : tâche qui décide où sera exécuté le prochain pod.

kube-controller-manager : responsable du contrôle d'exécution des ressources --> telles que les déploiements

cloud-controller-manager : gère les ressources telles que load-balancers et les disk volumes.

codeDNS : DNS-based Kubernetes service discovery.

Kubernetes : Concepts

namespaces

pod

deploy

service

endpoints

secret

configmap

persistent volume

persistent claim volume

Kubernetes démarche

https://medium.com/@bijit211987/kubernetes-ci-cd-pipelines-best-practices-and-tools-ca2158939299

K8S est un des leaders dans ce contexte CI/CD, mais utiliser des commandes kubectl manuellement peut être cause d'erreurs. en utilisant un pipeline CI/CD avec Kubernetes on va automatiser le déploiement des applications dans ce type de contexte.

7 bonnes pratiques pour CI/CD et Kubernetes

- use GitOps
- Scan your container images (Snyk, docker scan command)
- Use Helm to manage Deployments
- Ensure There's Rollbak Mechanism
- Use Immutable Image Tags : évitez les my-app:latest ???
- Follow Kubernetes security Best Practices : secret, RBAC ....
- Try pull-based CI/CD workflows (automatisation)

Kubernetes : déploiement

Helm : voir https://helm.sh/

installation ./get_helm.sh

Système d'installation clés en main à partir de dépôts sous Kubernetes

Fonctionne avec une notion de charts : ensemble de paramétrage spécifique pour le déploiement sous kubernetes.

Ajout de dépôt dans votre environnement :

helm repo add stable https://charts.helm.sh/stable

Exemple : helm install mysqlys bitnami/mysql

à exécuter sur une machine qui accéde au cluster

Kubernetes : déploiement

Helm :

Quelques commandes pratiques :

helm list

helm show chart bitnami/mysql

helm uninstall [nom]

helm install --dry-run --debug nginx-test bitnami/nginx

Attention à l'installation de service nécessitant de la persistance comme mysql, wordpress . Helm n'approvisionne pas votre système c'est à vous de mettre en place les PV qui vont bien pour permettre ensuite aux CLAIMS de les accrocher.

Kubernetes : persistance PV et PVC Applications statefull

Comment conserver une persistance quand un pod est supprimé ou replacé sur un autre worker sachant que l'on ne maîtrise pas où sont exécutés les pods. donc il faut que l'on ait un système de volume qui s'associe au pod.

Plusieurs solutions : local, NFS, CEPH, GlusterFS, EKS,AWS S3 .....

Premier concept PV :

est une façon pour définir un stockage de données tels qu'une Storage Class ou storage implementations.

est une ressource object de Kubernetes (kubectl get pv)

créer un PV est équivalent à créer un objet ressource de stockage avec un plugin spécifique en fonction de la nature de cette ressource. Pour utiliser cette ressource, on doit le demander à l'aide d'un PVC. Un PVC est une requête de stockage, laquelle est utilisée pour monter le PV dans le Pod.

L'administrateur peut mapper différentes classes sur différents niveaux de services et différentes règles.

Kubernetes : persistance PV et PVC Applications statefull

Un PV est décrit à partir d'un fichier Yaml qui permet de décrire la configuration et le plugin utilisé

```
apiVersion : v1
kind : PersistentVolume
metadata :
   name : pv005
spec :
   capacity :
      storage : 5Gi
   volumeMode : Filesystem
   accessMode :
     - ReadWriteOnce
   persistantVolumeReclaimPolicy : Recycle
   storageClassName : slow
   mountOptions :
      - hard
      - nfsvers : 4.1
   nfs:
     path : /tmp
     server : 172.17.0.2
```

Kubernetes : persistance PV et PVC Applications statefull

Les PVC ?

c'est une requête/déclaration de l'usage d'un espace de stockage. Lequel sera monté dans le pod. Niveau d'abstraction, les devs ne sont pas obligés de savoir comment et sur quoi va reposer leur espace de stockage.

Exemple de fichier yaml de déclaration d'un PVC

```
apiVersion : v1
kind : PersistantVolumeClaim
metadata :
   name : pvc004
spec :
   storageClassName : manual
   accessModes :
     - ReadWriteOnce
   ressources :
     requests :
       storage : 3Gi
```

Kubernetes : persistance PV et PVC Applications statefull

Lifecycle de PV et PVC

Provisionning : création du PV

Binding : assignation PV à PVC

Using : Pods utilisent le PV à travers le PVC

Reclaimaing : il est gardé pour une nouvelle utilisation ou destruction

Un volume peut être :

- Available
- Bound
- Released
- failed

Provisioning :

- Static : l'administrateur créer les PVs et lors des demandes ils sont associés aux PVC en fonction des classes éventuellement.
- dynamic : si il n'y a pas de PV, le cluster peut allouer directement des PVs pour répondre aux besoins des PVC.

HELM : package manager for Kubernetes

https://helm.sh

Rook : Storage orchestration platform

https://rook.io

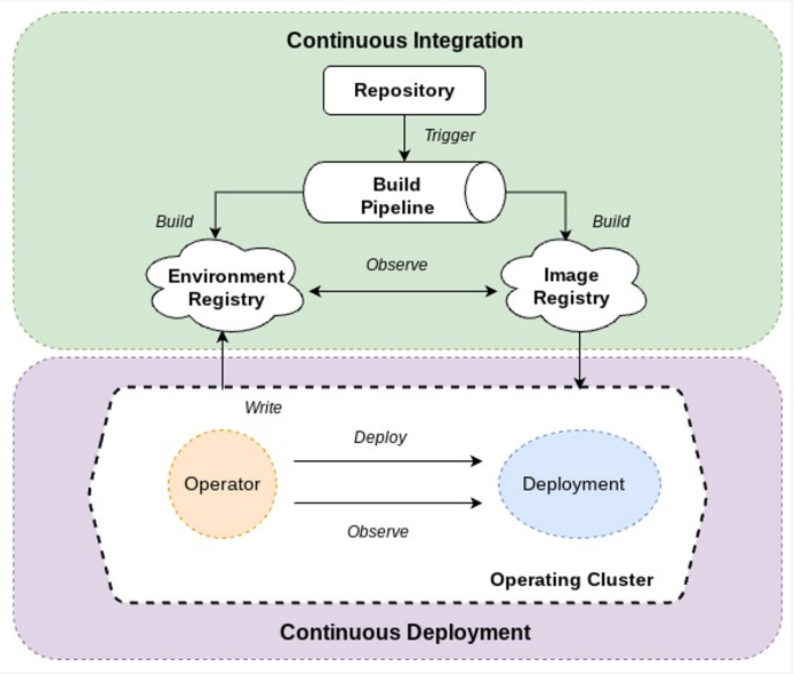
Mise en place de partage distribué sur la base de CEPH.

Kubernetes CI/CD pipelines

K8S supporte deux approches d'implémentation d'un pipeline CI/CD :

PULL-BASED pipeline

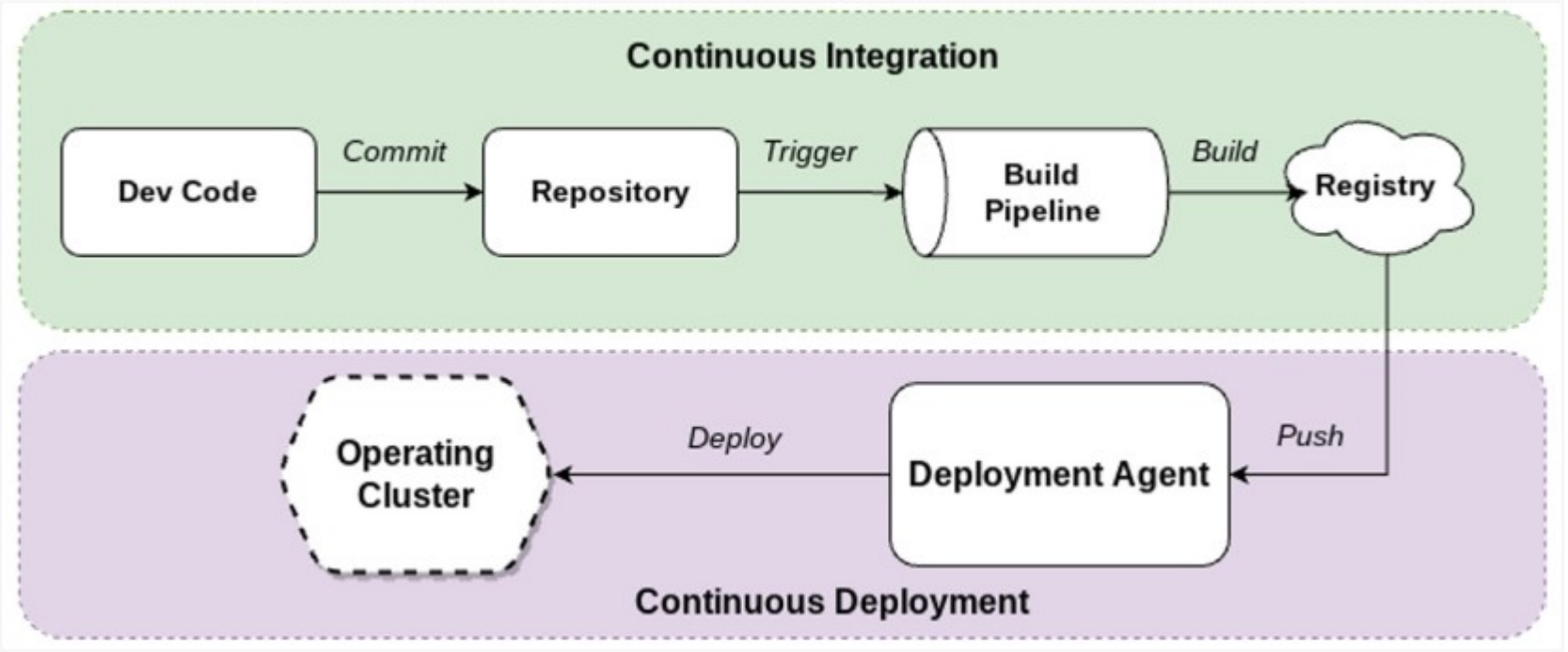
Figure 1: Pull-based CI/CD pipeline



Kubernetes CI/CD pipelines

PUSH-BASED pipeline

Figure 2: Push-based CI/CD pipeline



Kubernetes - CI/CD tools

Table 3

Tool	Description	Advantages	Disadvantages
Spinnaker	A multi-cloud CI/CD platform that relies on JSON to implement and configure software delivery pipelines. The platform implements in-built, production-safe deployment strategies to help deliver composite applications to their target environments without having to write any scripts. By following an API-based modular architecture, Spinnaker allows DevOps teams to integrate external services seamlessly for enhanced security, visibility, and cost efficiency.	• Rapid releases and deployments • Integrated deployment strategies • Rollbacks can be implemented with a single click • Out-of-the-box support for multi-cloud and hybrid models	• Adds operational overhead for orchestration of multiple underlying microservices • Allows only immutable deployments
Jenkins X	Helps accelerate software delivery by creating previews on pull requests while using Git as the single source of truth. As the platform provides an official Kubernetes plugin, it is considered one of the most adopted CI/CD tools for Kubernetes workloads.	• Provides automated feedback on pull requests and issues • Relies on Git as a single source of truth	• Limited to Kubernetes clusters and works purely on GitHub • Inefficient UI and dashboard view
Argo CD	A declarative, Kubernetes-centric continuous delivery tool that fetches commits made in a centralized repository and applies them to the production cluster. The platform relies on an agent that is installed directly within the cluster to implement pull-based GitOps for managing application updates and infrastructure configuration.	• Kubernetes-native deployments • Enforces portability and flexibility • Uses standard, declarative patterns for application delivery	• Being dependent on a pull-based mechanism, it requires a longer implementation cycle • Does not offer continuous integration out of the box

Orchestrateur : OpenShift (Redhat)

Quelques mots :

Ansible

Ansible sous ubuntu peut également s'installer avec une application graphique semaphore : sudo snap install semaphore

snap list

Nom	Version	Révision	Suivi	Éditeur	Notes
semaphore	2.8.89	238	latest/stable	semaphoreui	-

sudo snap stop semaphore

sudo semaphore user add --admin --login ystroppa --name ystroppa --email=yvan.stroppa@gmail.com --password=password

sudo snap start semaphore

Il faut suivre le même procéder qu'en ligne de commandes :

- Inventory
- Repository
- Environnement

Fonctionne en serveur et nodes

Utilise la notion de playbook (équivalent à la notion de recette dans les autres outils)

Ansible

Quelques exemples et vérifications

==> https://www.middlewareinventory.com/blog/ansible-command-examples/

ansible testservers –list-hosts -i ansible_hosts

ansible testservers -m command -a uptime -i ansible_hosts

Get the Uptime of remote servers

attention aux autorisations à fournir pour accéder aux machines

ansible-command-ex1.yml

```
- name: Check the remote host uptime
  hosts: testservers
  tasks:
    - name: Execute the Uptime command over Command module
      register: uptimeoutput
      command: "uptime"

    - debug:
        var: uptimeoutput.stdout_lines
```

ansible-playbook ansible-command-ex1.yml -i ansible_hosts

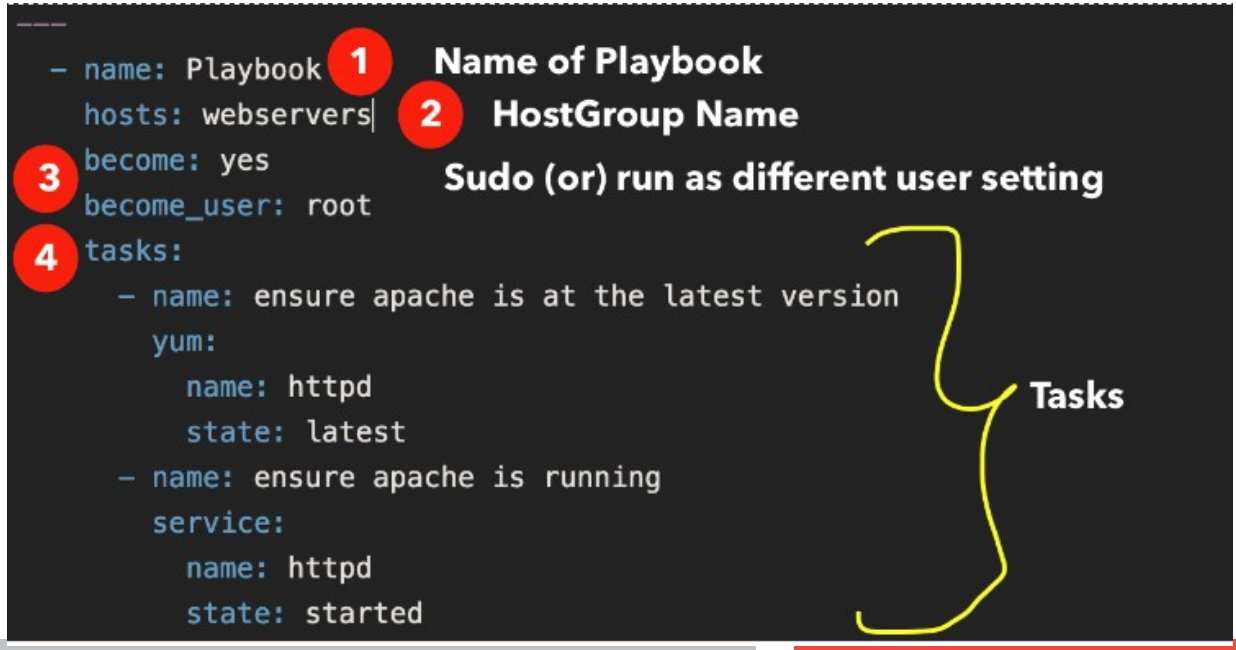
Exemple sous Ansible : playbook

```
- name : install and configure Apache web server
  hosts: webserver
  become: true
  tasks:
    - name: Install Apache package
      apt:
        name: apache2
        state: present
    - name: Enable Apache Service
      service:
        name: apache2
        enabled: true
        state: started
    - name: Copy curtoms index.html file
       copy:
          src: /path/to/index.html
          dest: /var/www/html/index.html
          group: root
          mode: 0644
```

Exemple détaillé

https://www.middlewareinventory.com/blog/ansible-playbook-example/

```
---
 - name: Playbook
   hosts: webservers
   become: yes
   become_user: root
   tasks:
     - name: ensure apache is at the latest version
       yum:
         name: httpd
         state: latest
     - name: ensure apache is running
       service:
         name: httpd
         state: started
```



Ansible

https://www.middlewareinventory.com/blog/ansible-playbook-example/

Exemple de ansible_hosts

cat ansible_hosts
[webservers]
mwivmweb01
mwivmweb02

Exécution en Dry Run
ansible-playbook -C sampleplaybook.yml -i ansible_hosts

Exécution du playbook
ansible-playbook sampleplaybook.yml -i ansible_hosts

Exemple sous Puppets

```
node 'webserver' {
   package { 'apache2' : ensure => installed }
   service {'apache2' :
         ensure => running,
         enable => true,
         require => Package['apache2'],
   }
   file { '/var/ww/html/index.html' :
        ensure => file,
        source => '/path/to/index.html' ,
        owner => 'root',
        group => 'root',
        mode => '0644',
        require => Package['apache1'],
        notify => Servuce["apache2'],
   }
}
```

Exemple sous Chef

```
package 'apache2' do
     action :install
end

service 'apache2' do
    action [:enable,:start]
end

cookbook_file
'/var/www/html/index.html' do
   source 'index.html'
   owner 'root'
   group 'root'
   mode '0644'
   action :create
   notifies:restart, 'service[apache2]'
end
```

CD : Jenkins

```
pipeline {
    agent any
    stages {
       stage('Build') {
          steps {
              sh 'npm install'
          }
       }
       stage('Test') {
         steps {
              sh 'npm install'
          }
      }
      stage('Deploy') {
         steps {
              sh 'npm run deploy'
          }
      }
   }
}
```

Installation et déploiement d'un package Java à l'aide de Maven.

CI/CD Gitlab

Fichier .gitlab-ci.yml

```
stage :
   - build
   - test
   - deploy

build :
   stage : build
   script :
      - npm install

test :
   stage : test
   script :
      - npm test

deploy :
   stage : deploy
   script :
      - npm run deploy
```

CircleCI : CI

```
version : 2.1
jobs:
   build-and-deploy:
    docker:
       - image:node:latest
    steps:
       - checkout
       - run:
            name: Build
            command: npm install
       - run:
            name: Test
            command: npm test
       - run:
            name: Deploy
            command: npm run deploy
```

Démarrage de Gerrit : https://www.gerritcodereview.com/

Gerrit un gestionnaire de version

Réf sous docker hub

https://hub.docker.com/r/gerritcodereview/gerrit

Démarrage sous docker

```
docker run -it -p 8080:8080 -p 29418:29418
gerritcodereview/gerrit
```

Compte pour l'accès : admin/secret

Pour se connecter : http://localhost:8080

Installation Jenkins : voir présentation dédiée

Voir https://github.com/jenkinsci/docker/blob/master/README.md

Attention pour le répertoire jenkins_home faire en sorte que les droits soient affectés au user PID : 1000 pour des problèmes de droits.

docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk11

En fonction des actions demandées il faudra peut-être prévoir de reconstruire l'image avec les éléments complémentaires (docker, ansible)

Installation GitLab-ce: voir présentation dédiée

Monitoring

https://sematext.com/blog/system-monitoring-tools/#what-is-the-best-system-monitoring-tool-for-you

What Is the Best System Monitoring Tool for You?

Here are some of the minimum requirements you should look for when choosing a system monitoring software:

Support for complex environments: Your system monitoring tool must be able to monitor highly distributed applications, ephemeral components, and diverse hosts and devices across multiple public and private clouds and networks.

Centralized management of metrics and insights: You need a reliable and easy-to-parse picture of system health, performance, and integrity. AI-powered analytics and strong visualizations in a consolidated dashboard can give you this.

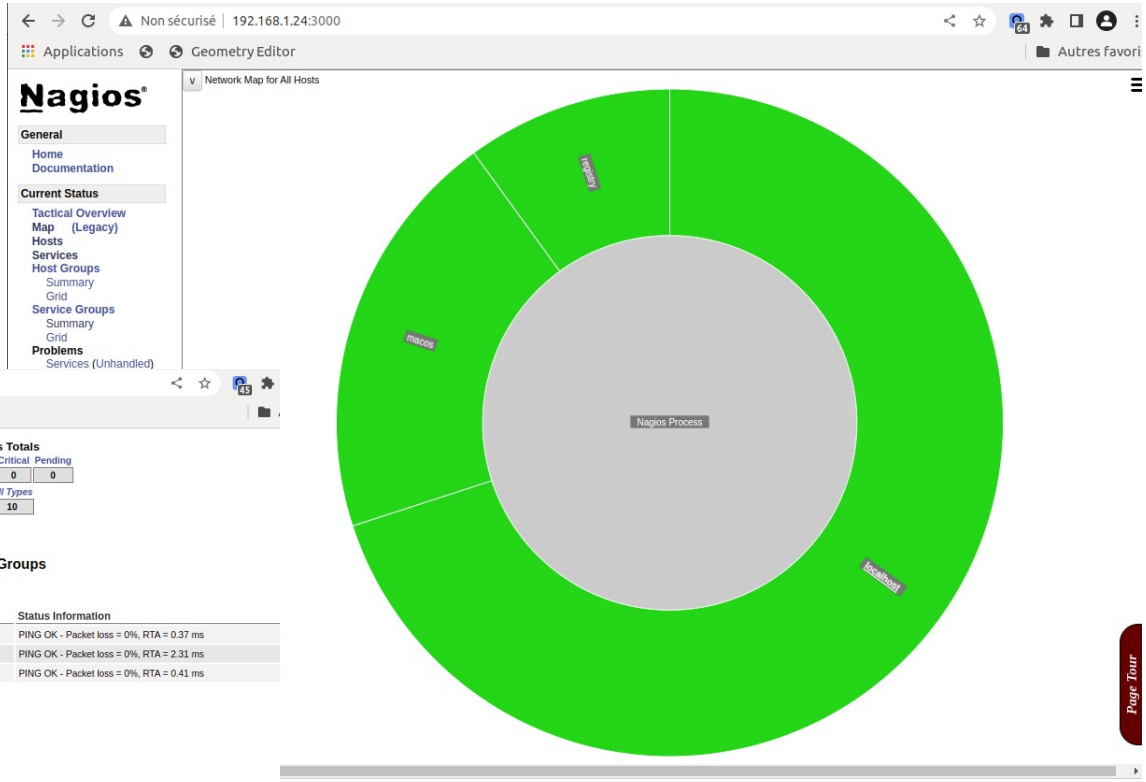
Notifications and alerts: The solution you choose must be able to detect anomalous system behavior anywhere in the stack that is, or could be, detrimental to system performance. It should also be able to notify the right personnel in real time.

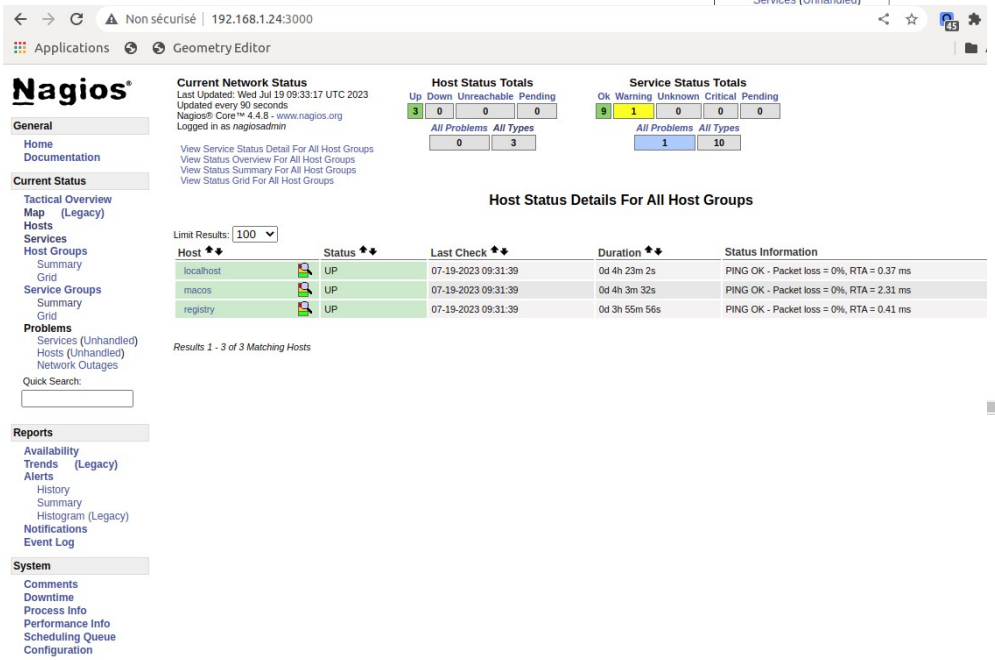
High levels of automation: Your system monitoring tool should be able to automatically open tickets with detailed information on system activities that led to a detected issue, as well as trigger automated workflows that mitigate, or even fix, the problem, such as shutting down a compromised service or deploying a patch. It should also be able to carry out preventive maintenance with little or no human intervention.

Scalability and flexible pricing: Your solution should make it easy for you to scale up or down in response to dynamic system requirements—and to adjust pricing accordingly.

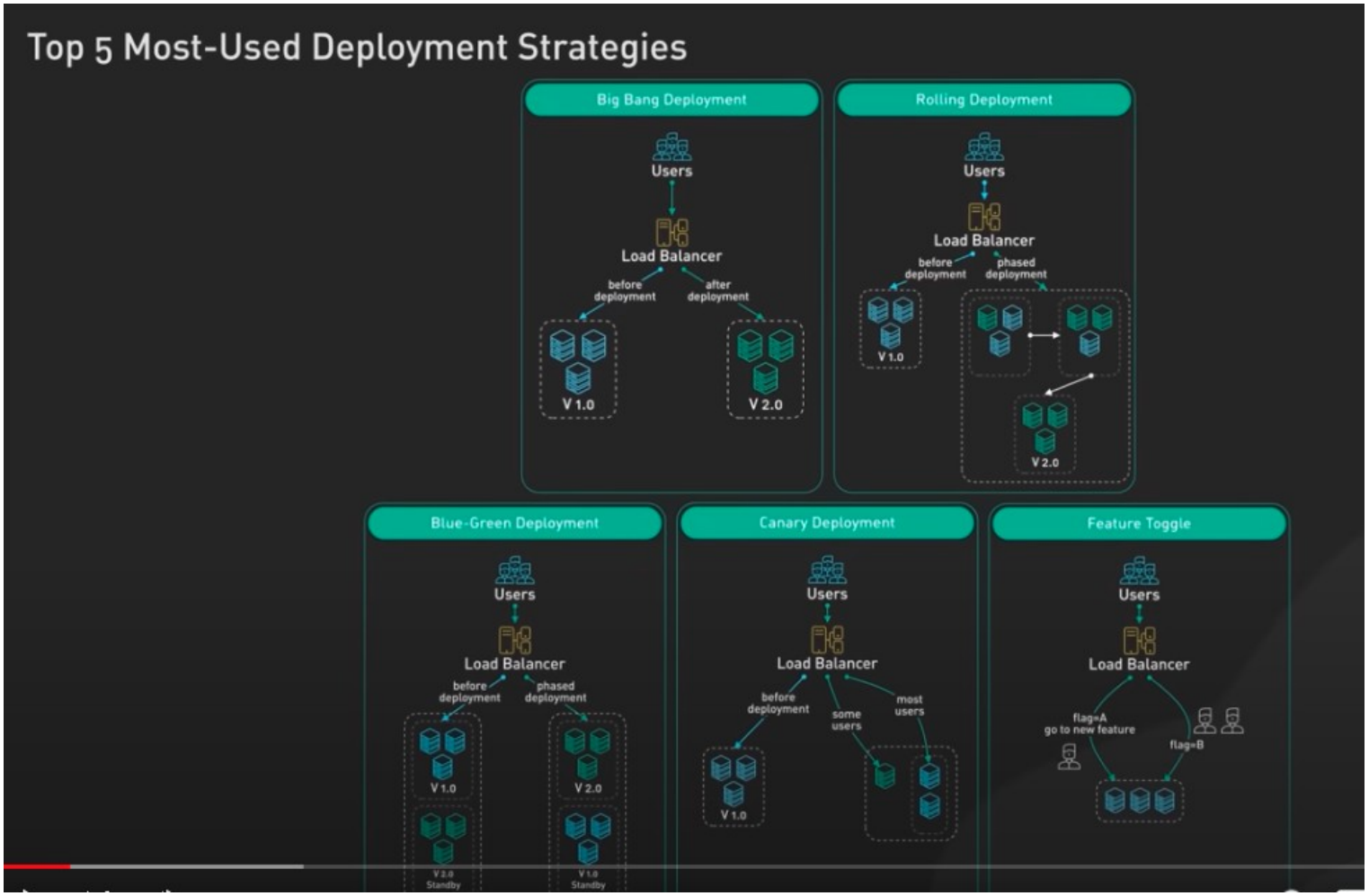
Reporting: To help with budget and capacity planning, as well as compliance auditing, choose a tool that comes with a set of informative historical and real-time reports. Since every system is different, these reports should be customizable.

Nagios


Non sécurisé | 192.168.1.24:3000
Applications
Geometry Editor
Autres favori
Nagios
Network Map for All Hosts
General
Home
Documentation
Current Status
Tactical Overview
Map (Legacy)
Hosts
Services
Host Groups
Summary
Grid
Service Groups
Summary
Grid
Problems
Services (Unhandled)
registry
macos
Nagios Process
localhost
Page Tour


Non sécurisé | 192.168.1.24:3000
Applications
Geometry Editor
Nagios
Current Network Status
Last Updated: Wed Jul 19 09:33:17 UTC 2023
Updated every 90 seconds
Nagios® Core™ 4.4.8 - www.nagios.org
Logged in as nagiosadmin
Host Status Totals
Up Down Unreachable Pending
3 0 0 0
All Problems All Types
0 3
Service Status Totals
Ok Warning Unknown Critical Pending
9 1 0 0 0
All Problems All Types
1 10
View Service Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups
Host Status Details For All Host Groups
Limit Results: 100
Host
Status
Last Check
Duration
Status Information
localhost
UP
07-19-2023 09:31:39
0d 4h 23m 2s
PING OK - Packet loss = 0%, RTA = 0.37 ms
macos
UP
07-19-2023 09:31:39
0d 4h 3m 32s
PING OK - Packet loss = 0%, RTA = 2.31 ms
registry
UP
07-19-2023 09:31:39
0d 3h 55m 56s
PING OK - Packet loss = 0%, RTA = 0.41 ms
Results 1 - 3 of 3 Matching Hosts
General
Home
Documentation
Current Status
Tactical Overview
Map (Legacy)
Hosts
Services
Host Groups
Summary
Grid
Service Groups
Summary
Grid
Problems
Services (Unhandled)
Hosts (Unhandled)
Network Outages
Quick Search:
Reports
Availability
Trends (Legacy)
Alerts
History
Summary
Histogram (Legacy)
Notifications
Event Log
System
Comments
Downtime
Process Info
Performance Info
Scheduling Queue
Configuration

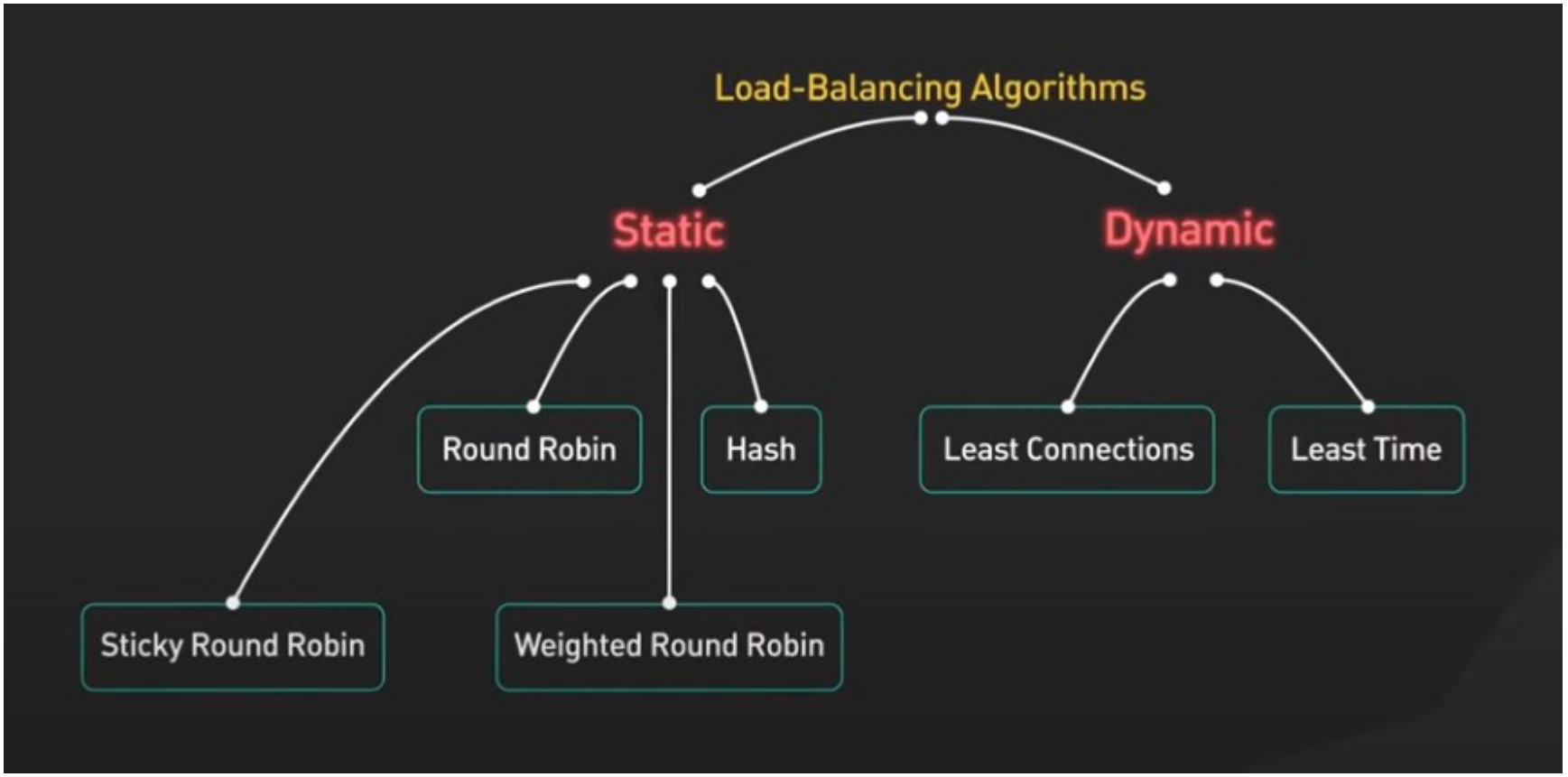
Stratégie de déploiements



Load Balancing

https://www.youtube.com/watch?v=dBmxNsS3BGE

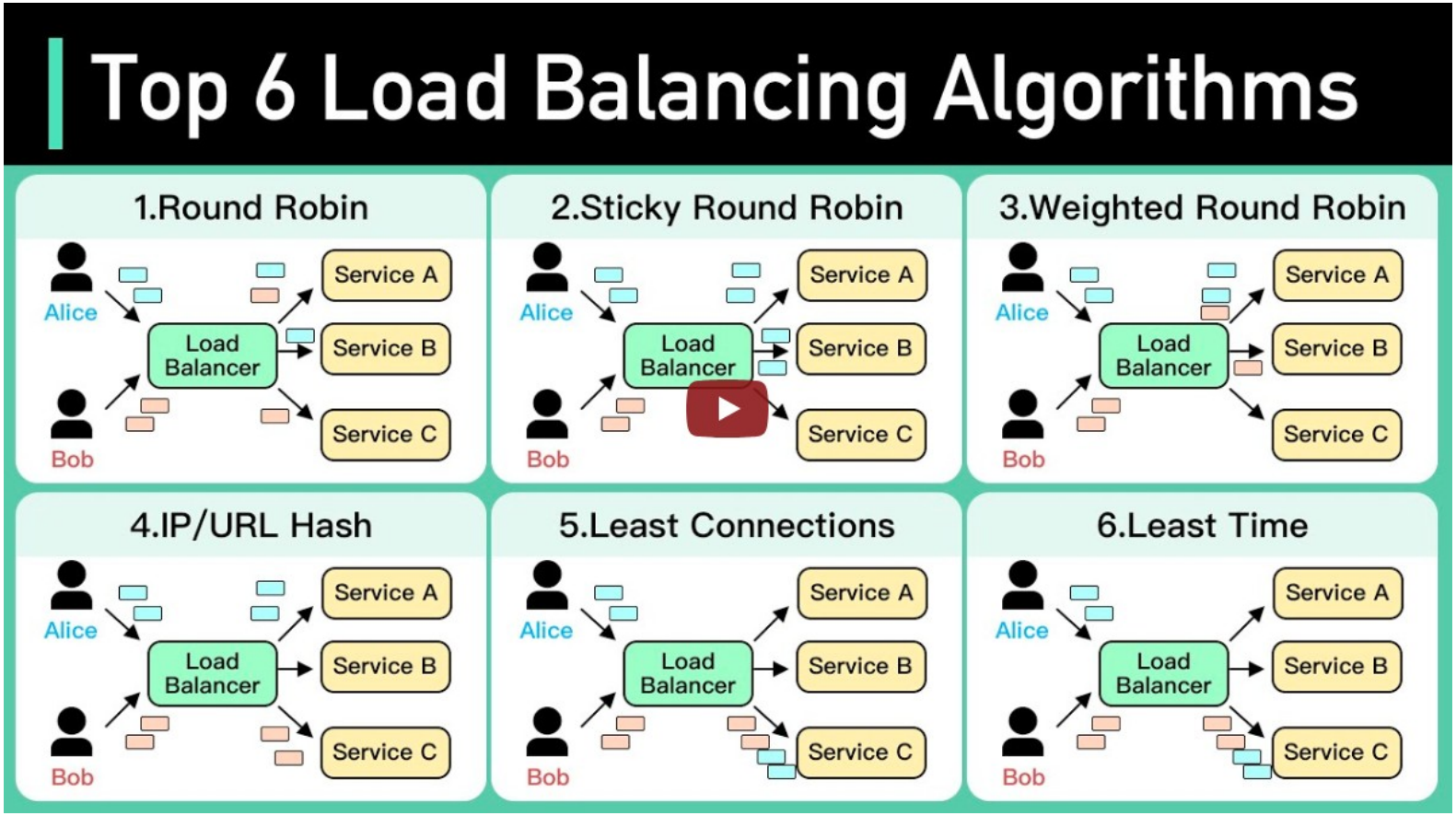
Algorithmes utilisés dans le load balanging



Load Balancing

https://www.youtube.com/watch?v=dBmxNsS3BGE

Algorithmes utilisés dans le load balancing



Idempotency dans les microservices ?

Les micro services divisent les applications en services faiblement couplés et déployables indépendamment, favorisant la flexibilité.

L'idempotence dans les micro services :

Une opération est considérée comme idempotente si son exécution plusieurs fois donne le même résultat que pour une exécution. Cette propriété subtile garantit la cohérence, la fiabilité et la prévisibilité des interactions entre les services distribués. Il garantit que répéter une opération ne modifiera pas le résultat si elle a déjà été exécutée.

Par ex, si une transaction de paiement est idempotente, l'appliquer plusieurs fois donnera le même résultat qu'une seule fois, quelques soient les tentatives ou les anomalies du réseau. dans de tels scénarios, l'idempotent devient un principe essentiel. Sans idempotent, des conséquences inattendues peuvent survenir suite à des opérations répétées, conduisant à des incohérences de données et à des états indésirables.

Idempotency dans les microservices ?

Exemples de fonctions simples :

fonction idempotente

la fonction absolue

a(x)=|x|

fonction non idempotente

Changement de signe

b(x)=-x

Autres exemples : idempotence et REST

operation	GET	HEAD	OPTION	POST	PUT	PATCH	DELETE	TRACE
Idemtpotent	Y	Y	Y	N	Y	N	Y	Y

Put est idempotent : car il fait une mise à jour si la donnée existe ou en créer une nouvelle

Idempotency dans les microservices ?

Quels sont les avantages de l'imempotence dans les micro services

- Cohérence et intégrité des données
- Prévention des effets secondaires involontaires
- Nouvelles tentatives fiables et gestion des erreurs
- Évolutivité et tolérance aux pannes

Quelles sont les applications de l'idempotence ?

- Traitements de commandes
- Gestion de stocks
- Traitements de paiements ...

Quelques références :

- https://www.baeldung.com/cs/idemtpotent-operation
- (quelques illustrations d'échange en C#)

Autre solution : ELK

ElasticSearch , Logtash, Kibana

SDLC : Software development Life Cycle

Les 7 phases :

Etape 1 : Planification du projet

Etape 2 : Collecte des exigences et analyse

Etape 3 : Conception

Etape 4 : Codage et mise en oeuvre

Etape 5 : tests

Etape 6 : déploiement

Etape 7 : entretien/maintenance

SDLC : Modèles

https://www.betsol.com/blog/7-stages-of-sdlc-how-to-keep-development-teams-running/

Modèle de cascade | ce modèle SDLC est considéré comme le plus ancien et le plus direct. Nous terminons par une phase puis commençons par la suivante, avec l'aide de cette méthodologie. Pourquoi ce nom de cascade ? Parce que chacune des phases de ce modèle a son propre mini-plan et chaque étape se transforme en cascade dans la suivante. Un inconvénient qui freine ce modèle est que même les petits détails laissés incomplets peuvent contenir tout un processus.

Modèle Agile | Agile est la nouvelle norme ; Il s'agit de l'un des modèles les plus utilisés, car il aborde le développement de logiciels par cycles incrémentiels mais rapides, communément appelés « sprints ». Avec de nouveaux changements de portée et d'orientation mis en œuvre à chaque sprint, le projet peut être réalisé rapidement avec une plus grande flexibilité. Agile signifie passer moins de temps dans les phases de planification et un projet peut s'écarter des spécifications initiales.

Modèle itératif | ce modèle SDLC met l'accent sur la répétition. Les développeurs créent une version rapidement pour un coût relativement moindre, puis la testent et l'améliorent au fil des versions successives. Un gros inconvénient de ce modèle est que si rien n'est fait, il peut rapidement consommer des ressources.

SDLC : Modèles

https://www.betsol.com/blog/7-stages-of-sdlc-how-to-keep-development-teams-running/

Modèle en forme de V : Ce modèle peut être considéré comme une extension du modèle en cascade, car il inclut des tests à chaque étape de développement. Tout comme dans le cas d'une cascade, ce processus peut se heurter à des obstacles.

Modèle Big Bang : Ce modèle SDLC est considéré comme le meilleur pour les petits projets car il consacre la plupart de ses ressources au développement. Il lui manque l'étape de définition détaillée des exigences par rapport aux autres méthodes.

Modèle en spirale : L'un des modèles SDLC les plus flexibles est le modèle en spirale. Il ressemble au modèle itératif dans la mesure où il met l'accent sur la répétition. Même ce modèle passe encore et encore par les phases de planification, de conception, de construction et de test, avec des améliorations progressives à chaque étape.

Rappel : les conteneurs

LXC, Docker , Podman, rkt, singularity, crio,kata

crio : https://access.redhat.com/documentation/fr-fr/openshift_container_platform/3.11/html/cri-o_runtime/use-crio-engine

https://cri-o.io/

Docker (rappel)

Point sur docker-compose : 2014

version python 1.x--> version go 2.x

Docker compose a été écrit en python et communiquait avec le démon docker via son API REST.

Visant à gérer des groupes de conteneurs basés sur une définition YAML

Ce YAML a reçu par la suite une spécification formelle --> la specification Compose
https://github.com/compose-spec/compose-spec/blob/master/spec.md

cette spécification définit un langage structuré vous permettant d'exécuter facilement plusieurs conteneurs sur une seule machine.

Alternative à l'exécution de ces conteneurs directement à partir de la ligne de commande.

Voir version en fonction

Podman : redhat

Podman s'exécute en tant qu'utilisateur non privilégié et sans daemon central.

Podman est sorti en 2018, concentrés au début sur la compatibilité avec la ligne de commandes (CLI) de Docker . N'a pas inclut tout de suite la prise en charge de l'API REST .

Un projet communautaire Podman Compose a vu le jour. Podman compose traite la spécification compose et la traduit en commandes CLI Podman.

Podman s'oriente plus vers Kube pour son interfaçage

Podman : exemple de commande

podman run -dt --publics-all --rm docker.io/httpd

activation un processus conmon

```
ystroppa@vmteo:/etc/containers$ podman ps
CONTAINER ID  IMAGE                           COMMAND         CREATED        STATUS
PORTS                   NAMES
f86aba57ad8a  docker.io/library/httpd:latest  httpd-foreground  13 seconds ago  Up 13
seconds ago  0.0.0.0:37785->80/tcp  bold_visvesvaraya
```

```
ystroppa@vmteo:/etc/containers$ podman
port -l
80/tcp -> 0.0.0.0:37785
```

```
ystroppa@vmteo:/etc/containers$ curl http://localhost:37785
<html><body><h1>It works!</h1></body></html>
```

Podman : notion de pod

Un pod est une grappe(groupe de conteneurs) qui vont partager des informations entre eux (ipc, net et uts).

cri-o

Rappel : les conteneurs
Best Practices

Use Smaller Base Image :

les tags sur docker hub indique la version de l'image et de l'OS.

Plus c'est petit, plus en réduit la surface d'attaque, analogie aux OS. Ne pas laisser de package par défaut qui ne sert à au processus que l'on souhaite démarrer.

Always use .dockerignore file

Afin de préciser les fichiers qui ne sont pas nécessaires pour le conteneur par exemple dans un projet le dist, build, Readme ou Dockerfile.

Use Specific Docker Image Version

éviter les FROM python en version latest et préférer FROM python:3.11 afin d'avoir une meilleure maîtrise et possibilité de reproductibilité de l'image.

Do not use the root user

éviter d'utiliser le moins de privilèges possibles. Indiquez dans la fabrique à la fin USER ... sinon par défaut c'est root.

Use multi-stage Builds

afin de ne pargder que l'essentiel à l'exécution du conteneur, comme par exemple avec java ; On n' a pas de besoin de JDK mais d'un JRE. Donc avec la notion mutli-stage on peut lors de la construction utiliser des environnements temporaires afin de produire le livrable et pousser uniquement le jar par exemple avec le jre dans l'image finale.

Sécurité autour des conteneurs

docker scout

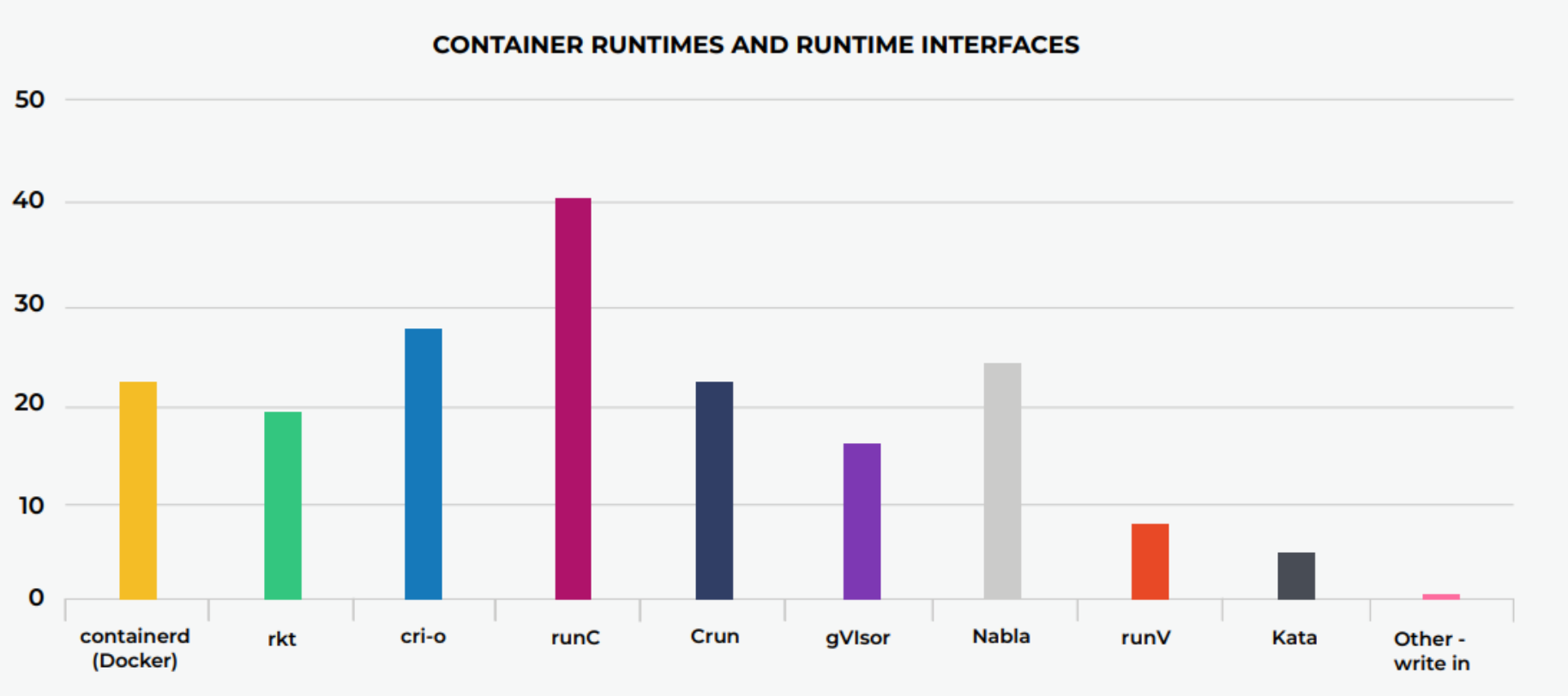
https://docs.docker.com/scout/quickstart/

Identifie les CVEs

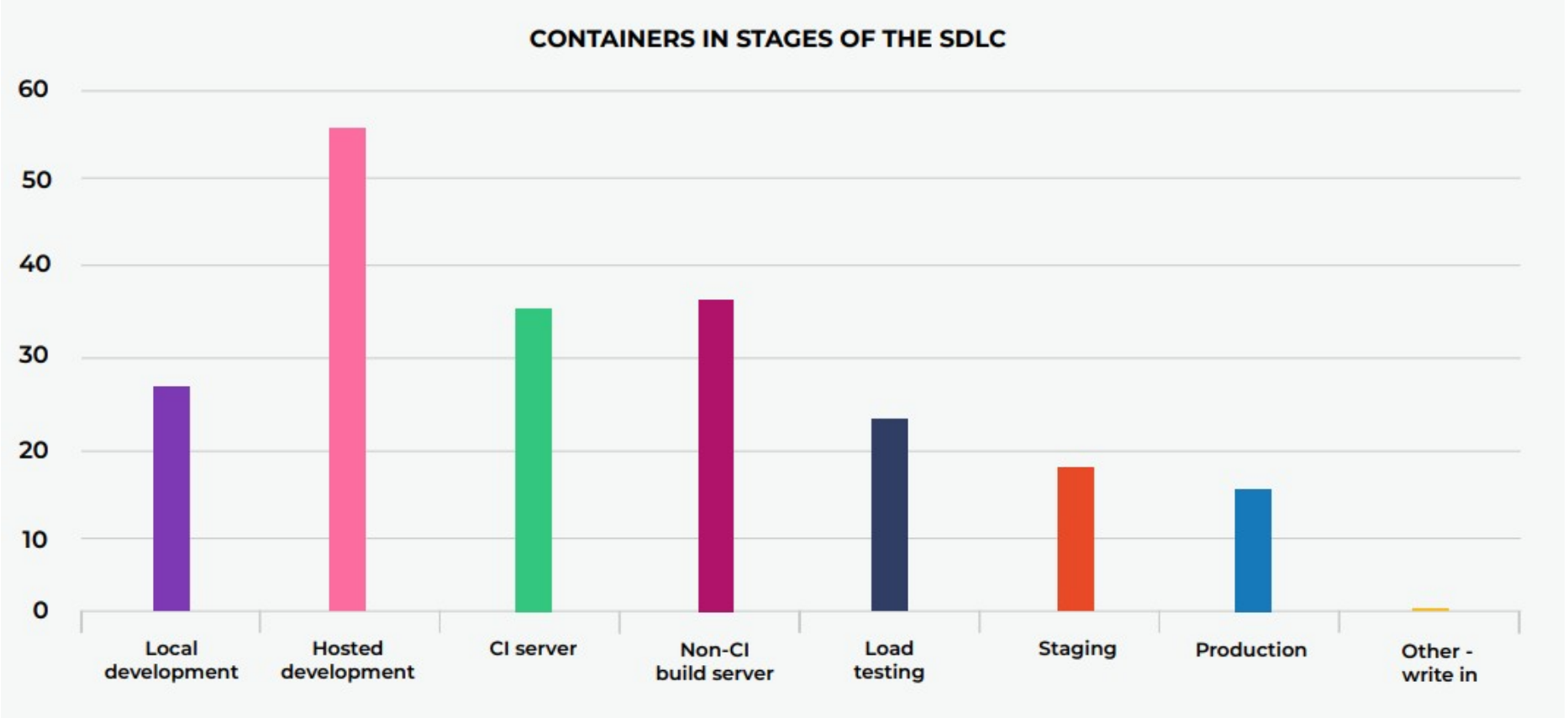
- Common Vulnerabilities and Exposure
- Exemple : docker scout recommendations httpd

Rappel : les conteneurs runtime

LXC, Docker , Podman, rkt, singularity, crio



Rappel : les conteneurs



Container Multi-Architecture

Multi-arch Docker image --> liste d'images avec binaires et librairies compilées pour de multi-architectures (ARM, x86, RISC-V,...)

- Performance and Cost Optimization
- Cross Platform Developement
- IoT Devices

Bénéfices d'utiliser des images multi-architectures

- capacité à exécuter une image docker sur plusieurs archi
- capable de choisir son eco-friendly CPU architecture
- moindre effort dans la migration d'une architecture vers une autre
- meilleure performance et cout en utilisant ARM64
- capacité à utiliser plusieurs cores par CPU en utilisant ARM64

Comment construire des images de conteneurs en multi-architectures

- Méthode traditionnelle
- en utilisant docker buildx
- en utilisant buildah

Container Multi-Architecture / manuelle

Dockerfile
FROM nginx
RUN echo"Hello But3" > /usr/share/nginx/html/index.html

sur une machine amd64
docker build -t ystroppa2/custom-nginx:v1-amd64
docker push ystroppa2/custom-nginx:v1-amd64

sur une machine arm64
docker build -t ystroppa2/custom-nginx:v1-arm64
docker push ystroppa2/custom-nginx:v1-arm64

Création du fichier manifest
docker manifest create ystroppa2/custom-nginx:v1 \
  ystroppa2/custom-nginx:v1-amd64 \
  ystroppa2/custom-nginx:v1-arm64
docker manifest push ystroppa2/custom-nginx:v1

Container Multi-Architecture / buildx

Dockerfile
FROM nginx
RUN echo"Hello But3" > /usr/share/nginx/html/index.html

docker buildx build --push --platform linux/amd64,
linux/arm64 -t ystroppa2/custom2-nginx:v1

Attention à la version de docker avec l'option
platform

Avec Buildah

Dockerfile
FROM nginx
RUN echo"Hello But3" > /usr/share/nginx/html/index.html

Au préalable installer
sudo apt install qemu-user-static

buildah manifest create multiarch-test
buildah bud --tag ystroppa2/cut-nginx:v1-amd --manifest multiarch-test --arch amd64 .
buildah bud --tag ystroppa2/cut-nginx:v1-arm --manifest multiarch-test --arch arm64 .
buildah manifest
buildah manifest inspect multiarch-test

Résultat sur docker hub

TAG

v1

Last pushed **a few seconds ago** by ystroppa2

```
docker pull ystroppa2/custom-ng…
```

DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
fd1d74fc6de0	linux/amd64	15 minutes ago	51.23 MB
9ad1e90f98a3	linux/arm64/v8	5 minutes ago	64.07 MB

contenu du manifest

```
(base) ystroppa@ystroppa-Latitude-5510:~/BUT3/multi-arch$ docker manifest inspect  ystroppa2/custom-nginx:v1
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
   "manifests": [
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1777,
         "digest": "sha256:fd1d74fc6de007b532bc3ce955a34ad4103449838632c15ece7b3b1ce049cbcc",
         "platform": {
            "architecture": "amd64",
            "os": "linux"
         }
      },
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1985,
         "digest": "sha256:9ad1e90f98a3beb374c3bd9f4131825ad79cd79bc61463502618cb132e0e1e35",
         "platform": {
            "architecture": "arm64",
            "os": "linux",
            "variant": "v8"
         }
      }
   ]
}
```

Container Multi-Architecture

Deux liens de réfs :

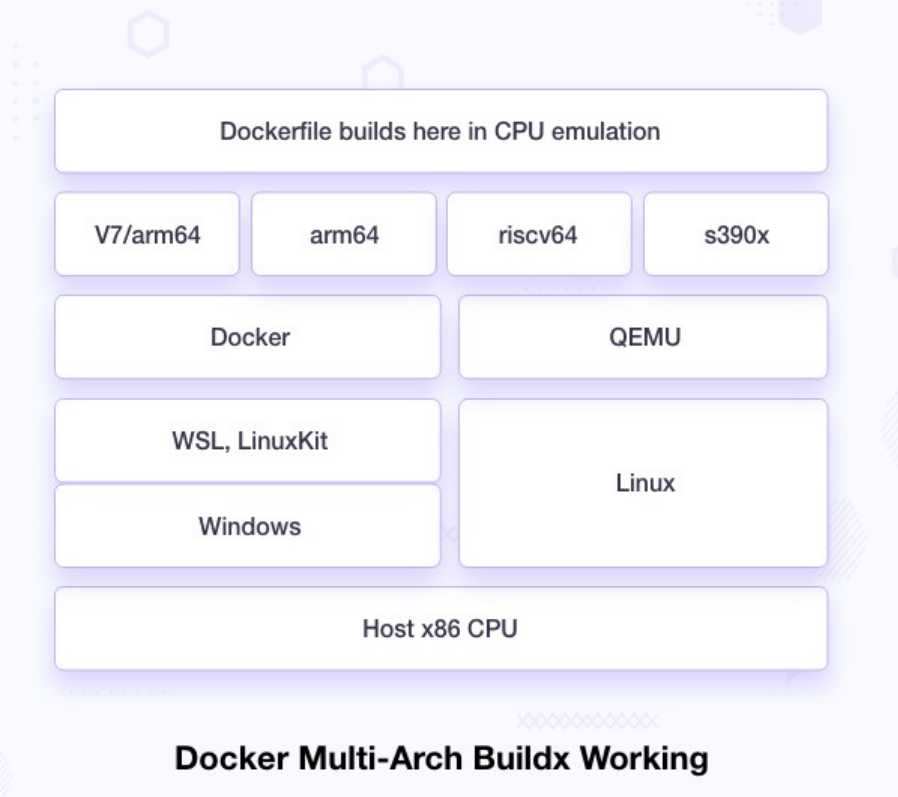
Avec docker :

https://dzone.com/articles/understanding-multi-arch-containers-benefits-and-c

Avec Buildah

https://danmanners.com/posts/2022-01-buildah-multi-arch/

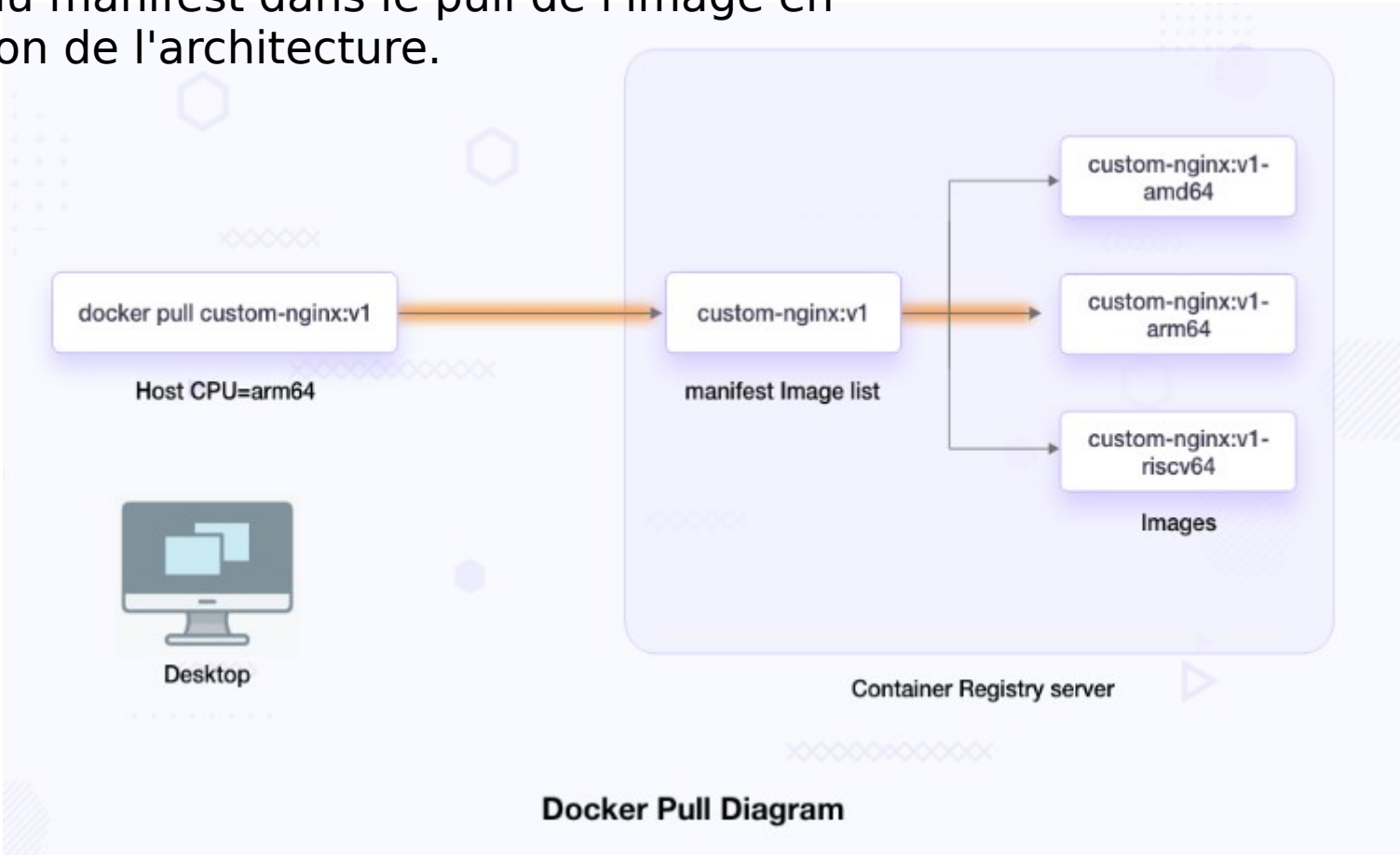
Container Multi-Architecture



Docker Multi-Arch Buildx Working

Container Multi-Architecture

Rôle du manifest dans le pull de l'image en fonction de l'architecture.



Multi-arch : intégration dans Jenkins CI

Description du pipeline de traitement

```
pipeline
{
    agent {
        label 'worker1'
    }
    options{
        timestamps()
        timeout(time: 30, unit: 'MINUTES')
        buildDiscarder(logRotator(numToKeepStr: '10'))
    }
    environment {
        DOCKER_REGISTRY_PATH = "https://registry.example.com"
        DOCKER_TAG = "v1"
    }
```

Multi-arch : intégration dans Jenkins CI

```
   stages
   {
      stage('build-and-push')
      {
      steps{
        script{
          docker.withRegistry(DOCKER_REGISTRY_PATH, ecrcred_dev){
          sh '''
            ######## check multiarch env ############
            export DOCKER_BUILDKIT=1
            if [[ $(docker buildx inspect --bootstrap | head -n 2 | grep Name | awk -F" " '{print $NF}') !=
"multiarch" ]]
            then
              docker buildx rm multiarch | exit 0
              docker buildx create --name multiarch --use
              docker buildx inspect --bootstrap
            fi
            ######## Push multiarch ############
            docker buildx build --push --platform linux/arm64,linux/amd64 -t
"$DOCKER_REGISTRY_PATH"/username/custom-nginx:"$DOCKER_TAG" .
          '''
          }
        }
      }
    }
  }
}
```

Multi-arch intégration dans Github CI

```
name: docker-multi-arch-push
on:
  push:
    branches:
      - 'main'
jobs:
  docker-build-push:
    runs-on: ubuntu-20.04
    steps:
      - name: Checkout Code
        uses: actions/checkout@v3
      - name: Set up QEMU
        uses: docker/setup-qemu-action@v2
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2
```

```
- name: Login to docker.io container registry
  uses: docker/login-action@v2
  with:
    username: $
    password: $
- name: Build and push
  id: docker_build
  uses: docker/build-push-action@v3
  with:
    context: .
    file: ./Dockerfile
    platforms: linux/amd64,linux/arm64
    push: true
    tags: username/custom-nginx:latest
```