



\* Le monde est tel que nous le façonnons.

The world is how we shape it\*

 **Urssaf**  
Caisse nationale

sopra  steria

# L'industrialisation

- └ L'industrialisation informatique désigne les étapes successives qui mènent à une gestion optimisée des ressources dans un contexte spécifique. Les améliorations de la performance s'effectuent suivant 4 axes distincts que sont:
  - La performance financière
  - Le gain en efficience pour les processus internes
  - La performance des métiers au sein de l'entreprise
  - La capacité à mieux maîtriser les risques et à assurer une amélioration continue

# DEP OVERVIEW

Digital Enablement Platform (Plateforme de mise en œuvre du numérique)

# Sommaire

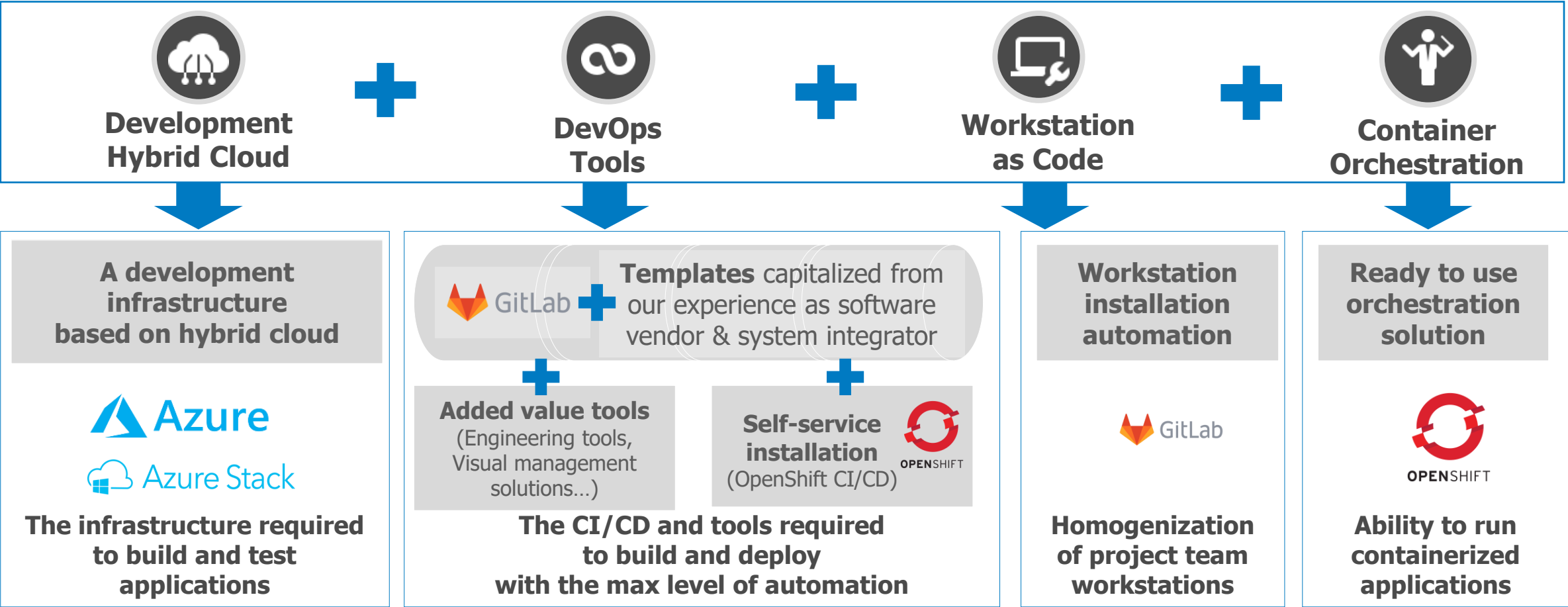
- 01** Introduction aux principes de la DEP
- 02** Présentation des outils
- 03** Pour aller plus loin
- 04** Conclusion

# 01

## Introduction aux principes de la DEP

# development environment Overview

## WHAT DOES DEP PROVIDE FOR YOUR DAY TO DAY DEVELOPMENTS?





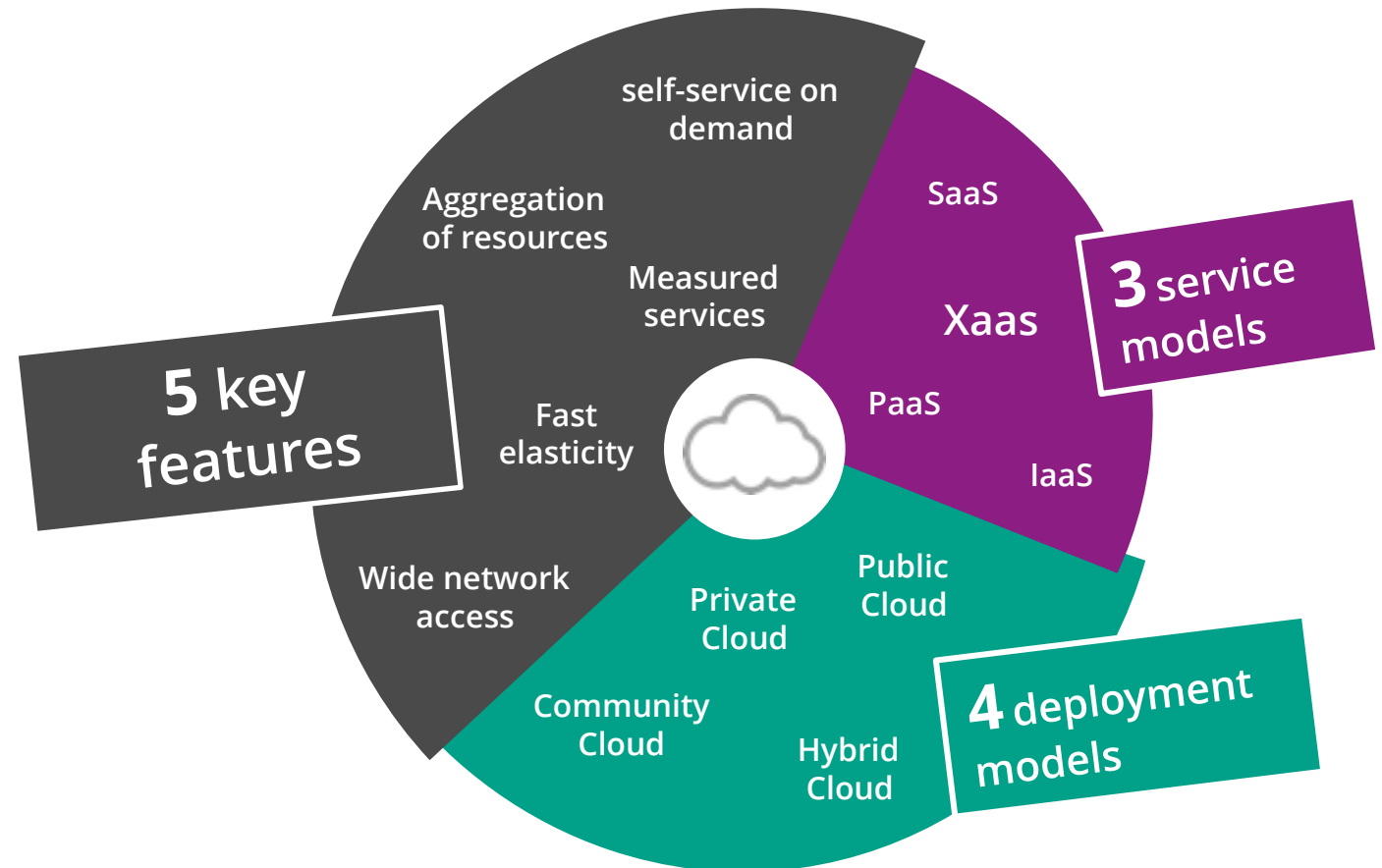
# What is the Cloud ?

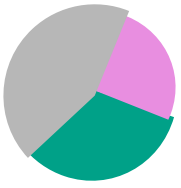
## DEFINITION OF THE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

Cloud computing is a way to deliver IT services

Resources are shared and accessible as services that can be directly used by customers :

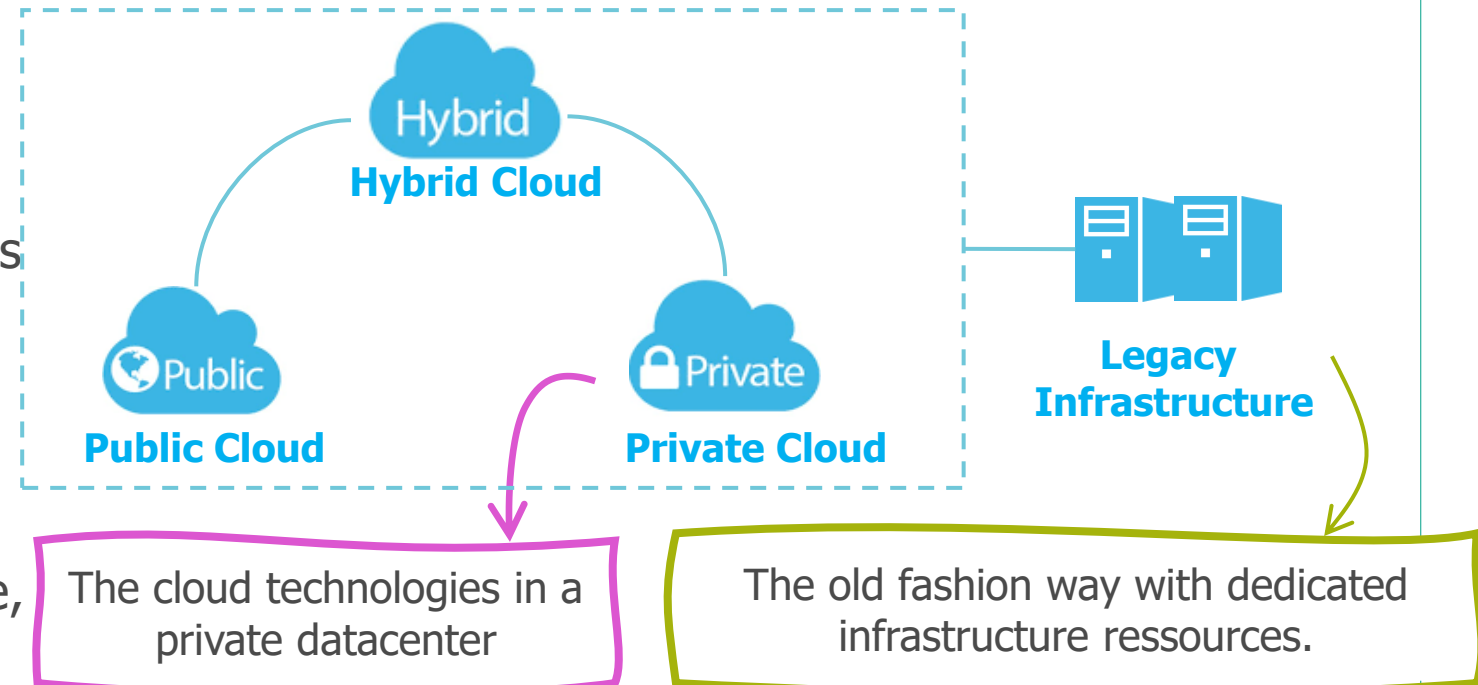
*payment methods, CRM, operating systems, bases "as-a-service" databases*





# Hybrid cloud: a combination between Public cloud and Private Cloud

- The Public Cloud offers:
  - The full power and elasticity of the Cloud
  - An access to a wide range of services
- The Private Cloud offers:
  - The control of the location, and thus data sovereignty
- Hybridation allows to mix both, with a possible access to legacy infrastructure, for example a database



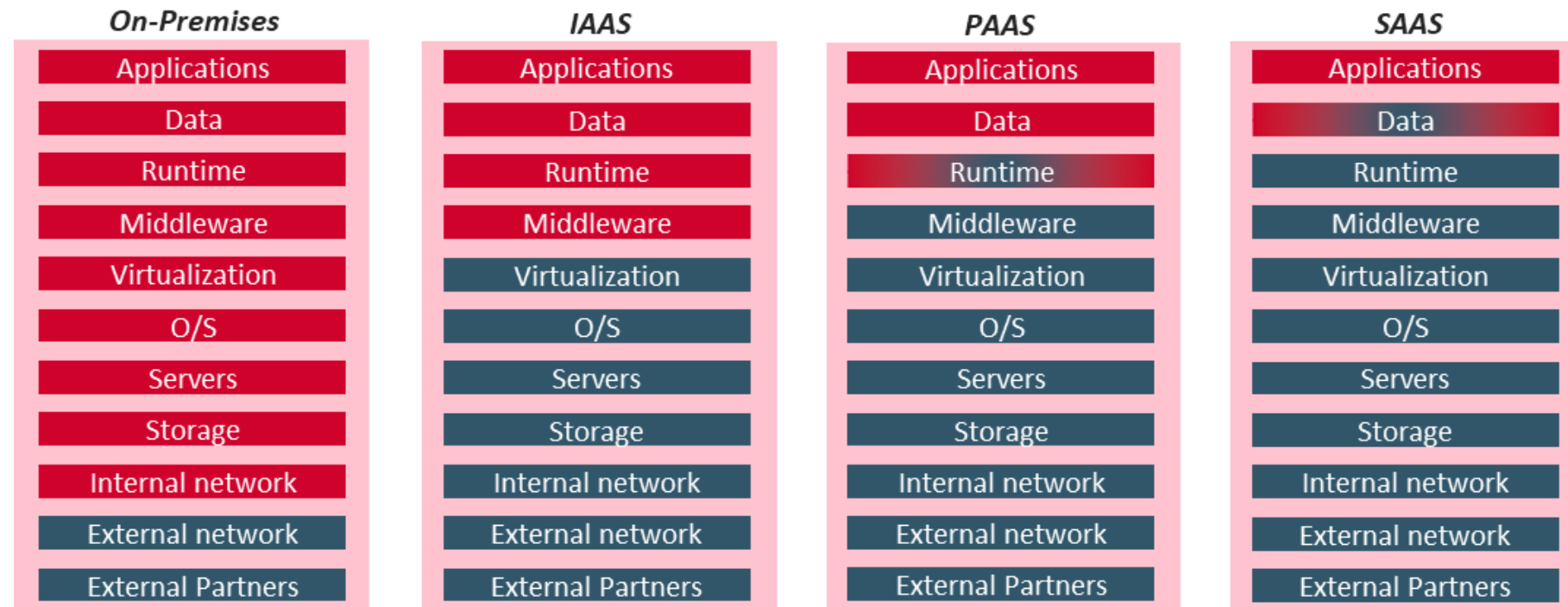




## Service models



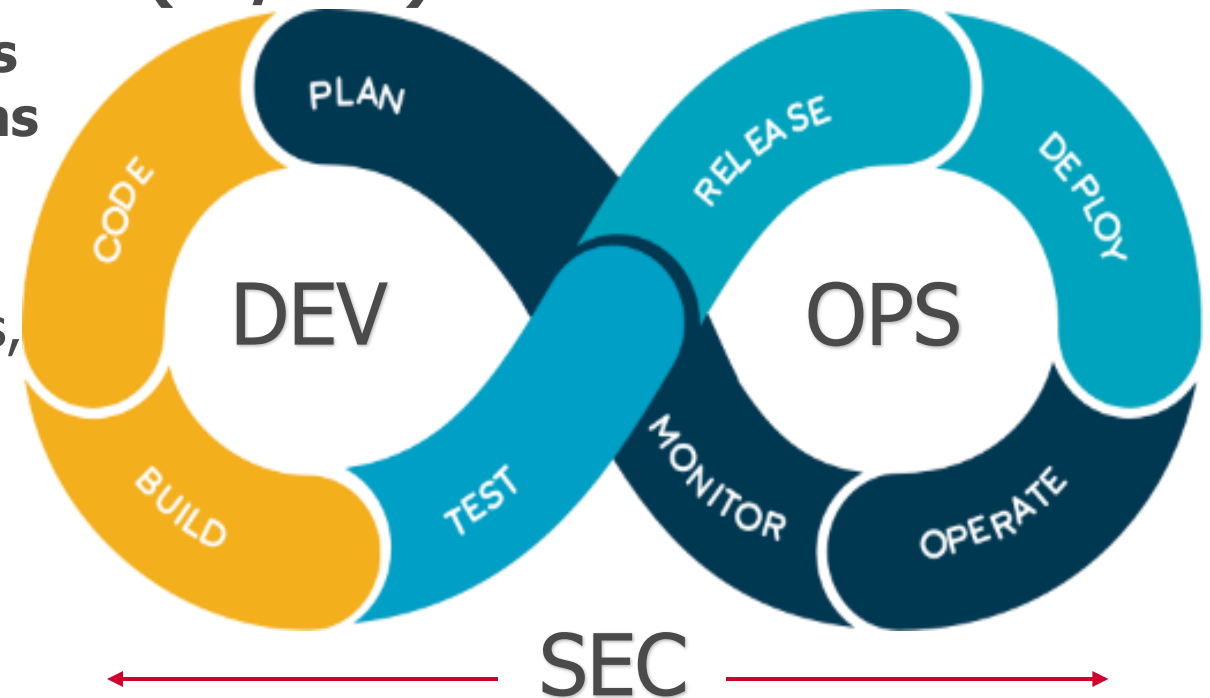
Cloud & DevOps :  
a technological  
disruption





### Continuous Integration/Continuous Delivery/Continuous Deployment (CI/CD<sup>2</sup>)

- Software developed in **successive increments within short cycles** and **deployed as soon as ready**
- **Agile and DevOps cultures and practices**, strengthening **collaboration** between business, developers and operations
- **Automation mandatory** to ensure quality
- Enriched with a **permanent attention to Security**: DevOps → DevSecOps



# 02

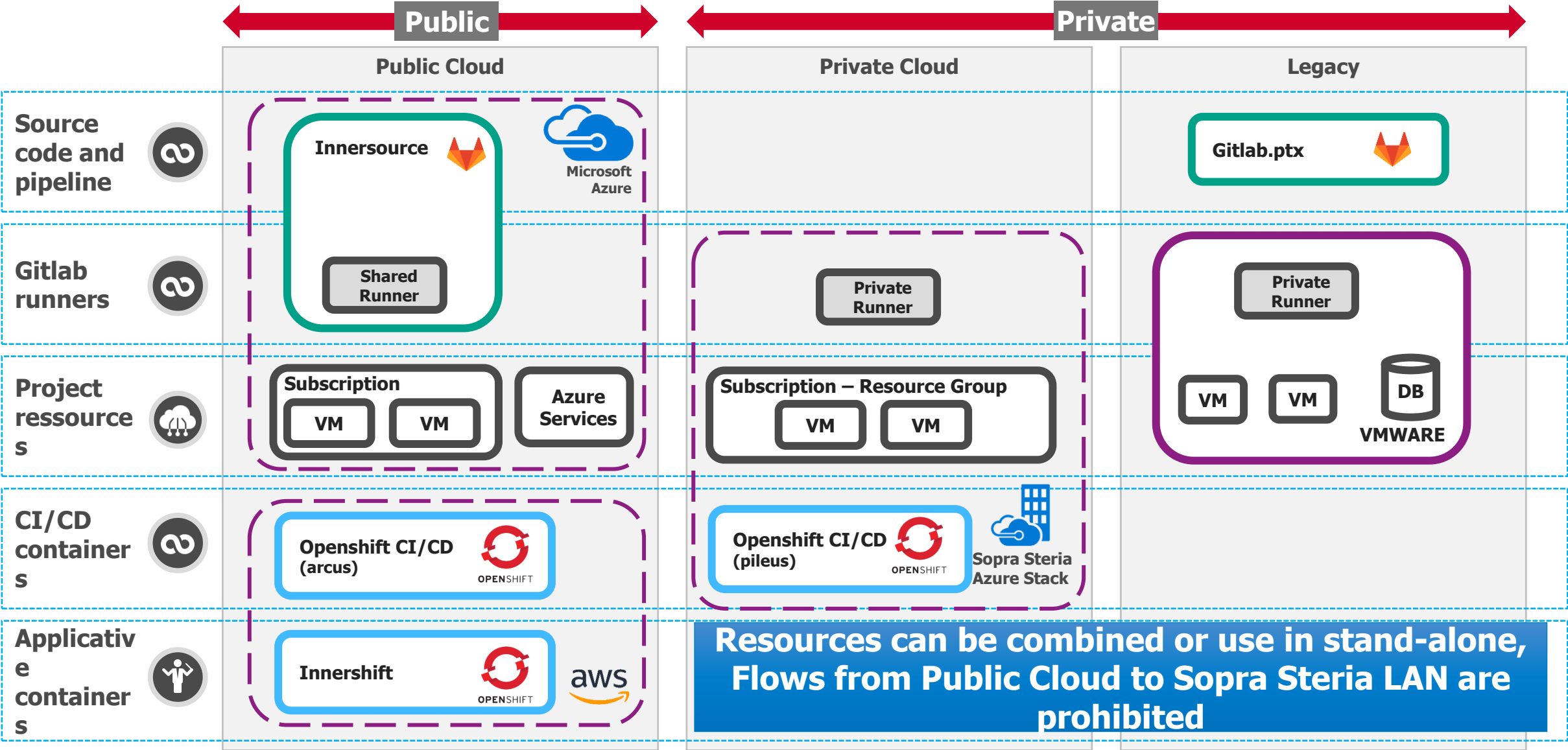
## Présentation des outils

# Development environment Overview

## WHERE ARE DEP RESOURCES FOR DEV ENVIRONMENT LOCATED?



Development environment overview



# Hosting Project resources

## NETWORK ACCESS, FLOW AND FIREWALL POSITIONNING



Development  
environment: hosting  
project resources

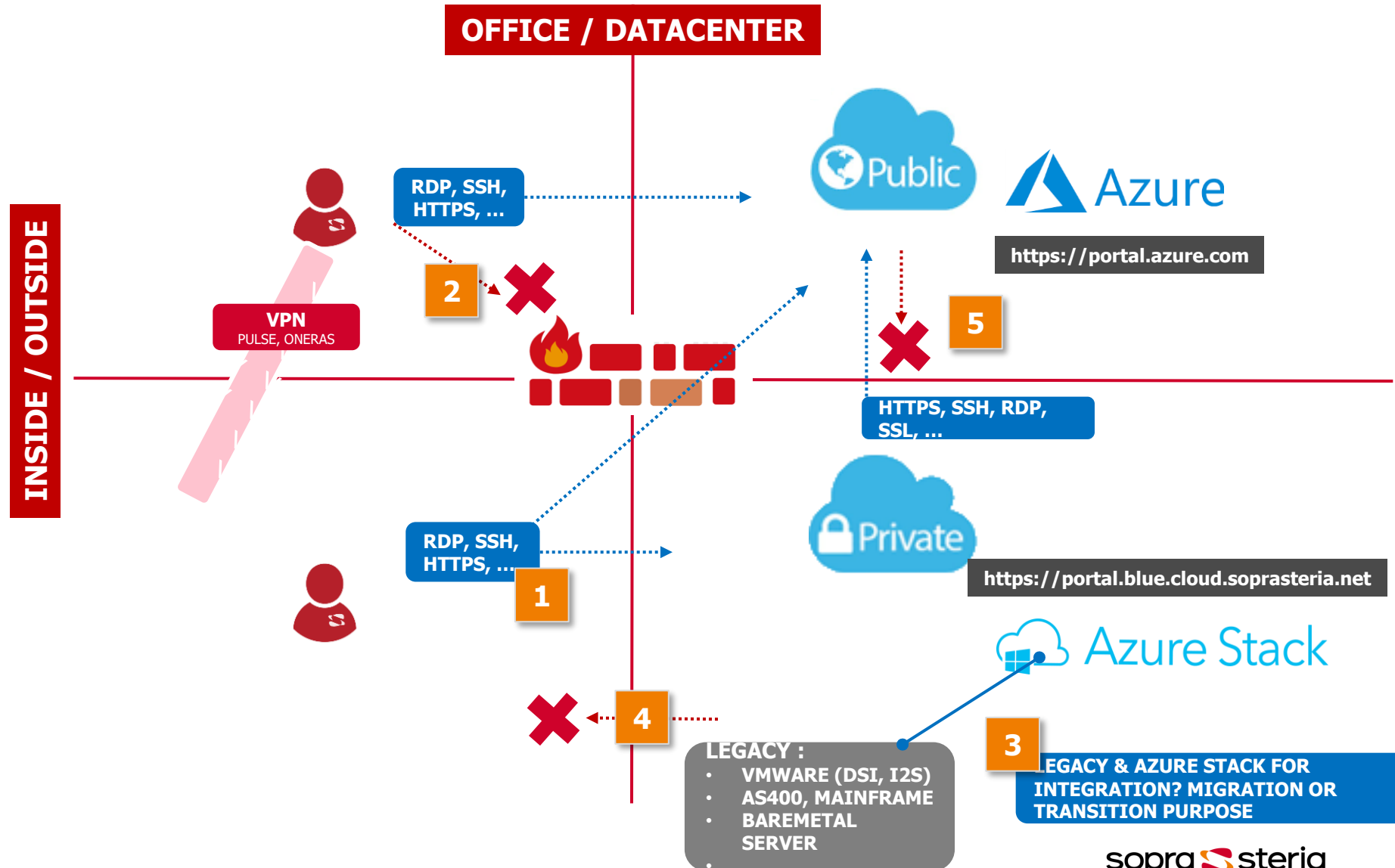
1/ Admin access to  
Azure/Azure Stack  
workload

2/ Can't access Azure Stack  
from Internet, need to  
make VPN with Office  
Network

3/ Azure Stack ↔ Legacy  
(Servers) possible

4/ Azure Stack =>  
Workstation, Non Server  
workload, ... not authorized

5/ Hybrid scenario where  
Azure workload contact  
Azure Stack in 2020  
roadmap




# DevOps Tools

## CREATE A NEW PROJECT FROM TEMPLATE



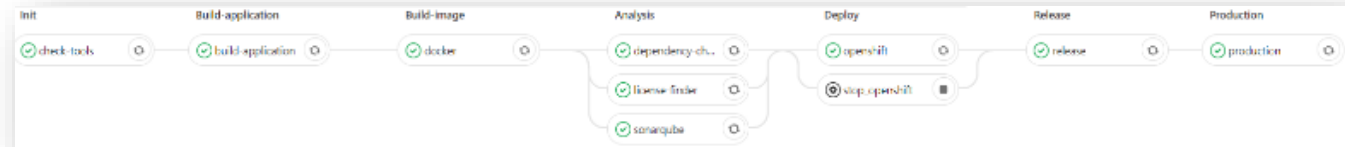
Using the  
templates

### Template content

 **java-gradle-template**  
Sample project using gitlab-ci templates based on Gradle and Spring boot starter

[Preview](#) [Use template](#)

- ❑ A Gitlab CI/CD pipeline



- ❑ A simple Springboot app, built with Gradle, package with Docker

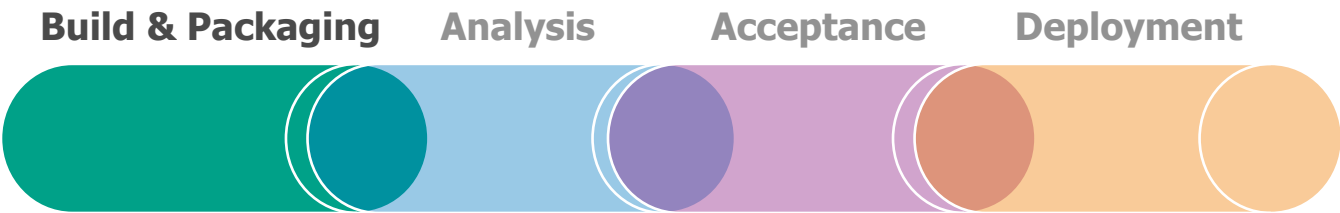


- ❑ A deployment template to deploy the application in Innershift

- ❑ Pre-configured files for Sonarqube analysis, license finder, Dep. Check, Intools as code

# DevOps tools

## PIPELINE & TOOLS: BUILD & PACKAGING



Automatically build the application and run the unit tests

Compilation

 maven

 Gradle

 nuget

 npm

 php

Automated Unit Testing

 JUnit

 nunit

 PHPUnit

Code Review

 GitLab

Packaging

 GitLab

 docker

 Nexus

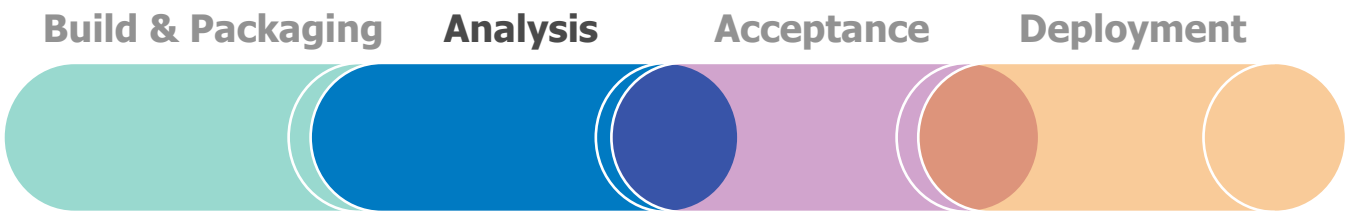
DevOps mindset:  
All repetitive activities must be automated.

It's mandatory when you want to be able to deliver often and rapidly



# DevOps tools

## PIPELINE & TOOLS: ANALYSIS



Ensure the code quality and security

- Static Code Analysis
- Static Application Security Testing
- Static Containers Security Testing
- Technical Debt Management



LicenceFinder



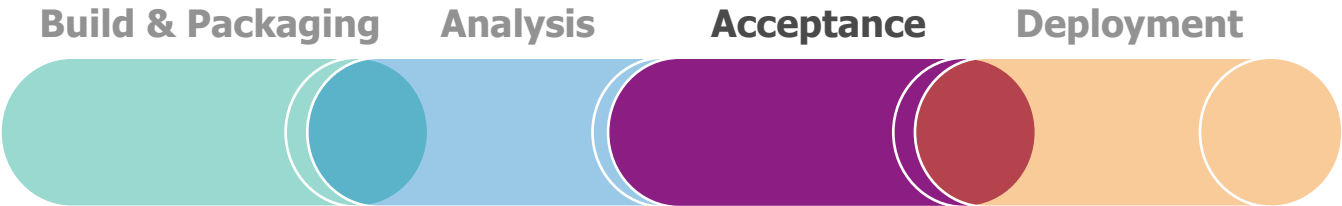
Remember our target:

Being highly professional and combine TTM, Security and Quality

Automating the analysis is fine, but checking the results is essential!

# DevOps tools

## PIPELINE & TOOLS: ACCEPTANCE



Perform automated tests

Non Regression Testing



Performance Testing



Dynamic Application Security Testing



Traceability



Automate all the type of tests but do it with a strategy!

Automating is key, automating without focus can be a loss of time

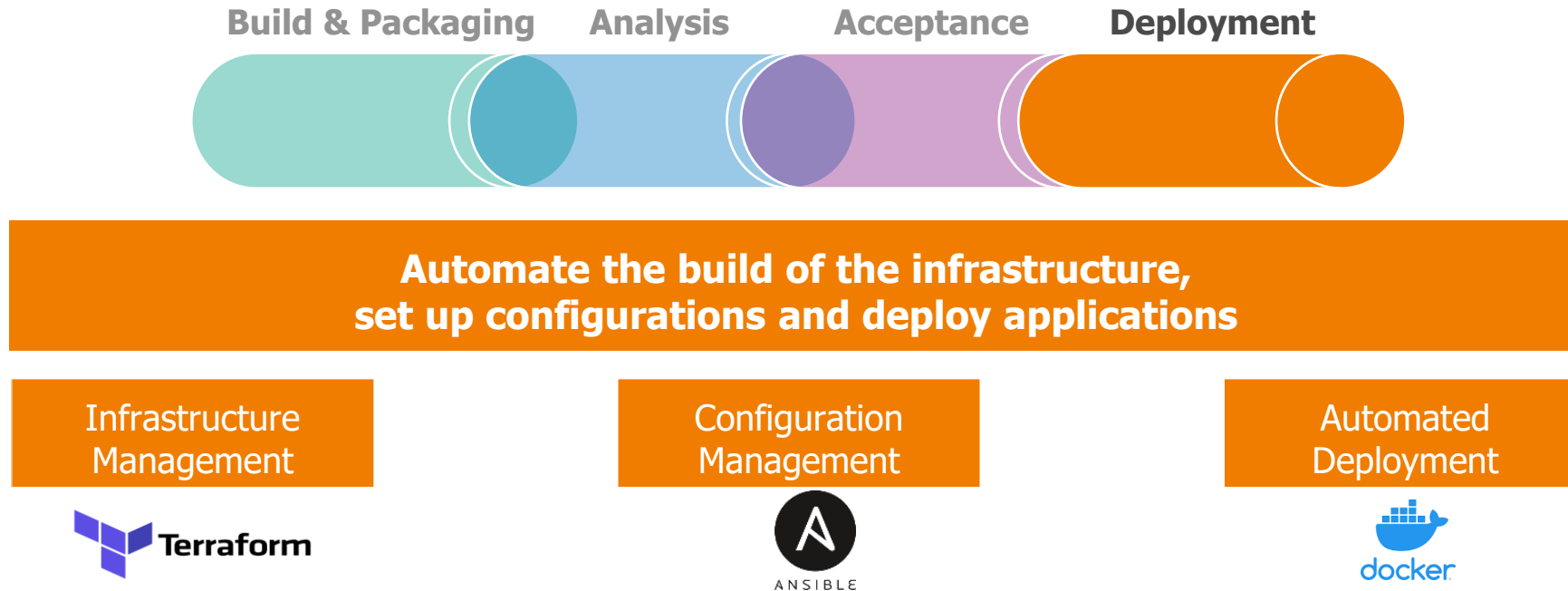
Go progressively:

Automate first the non regression tests you realize each time, and the most critical. Focus on the tests that are not modified everytime

It requires functional and technical skills

# DevOps tools

## PIPELINE & TOOLS: DEPLOYMENT



**These steps will be required several times:**

- Deployment in our testing environment to be able to run Acceptance activities
- Deployment in integration environment, performance tests environment, pre-production...
- Finally deployment in production environment, if you go full DevOps

**Automate the deployment to be sure you can deploy with serenity whenever you need it, including hot fixes**

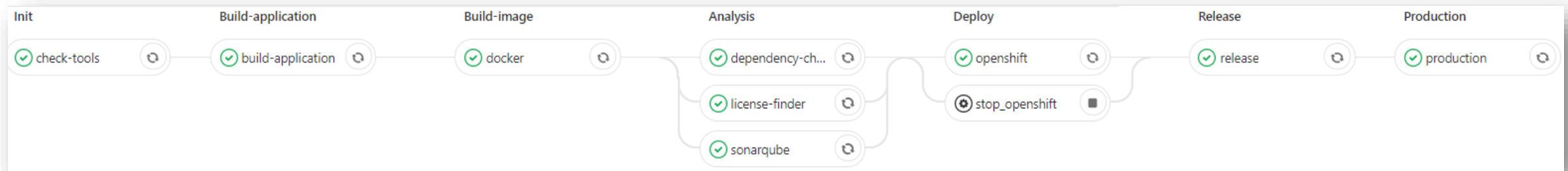
**Automation is key!  
Infrastructure as code  
is  
part of the answer**

# Gitlab concepts

## GITLAB-CI – PIPELINE TEMPLATE



Using the  
templates



Share common stages and jobs

Simple way to override or enhance existing jobs

gitlab-ci.yml available as file template too

Open for contributions (ci library is a gitlab-project)

```
include:  
  - "/configuration/common.yml"  
  
  - "/stages/init.yml"  
  - "/stages/build.yml"  
  - "/stages/analysis.yml"  
  - "/stages/deploy.yml"  
  - "/stages/acceptance.yml"  
  - "/stages/release.yml"  
  - "/stages/production.yml"
```

# Gitlab concepts

## GITLAB-CI - EXAMPLES

```
53 snapshot:
54   except:
55     - master
56     - tags
57   stage: Build
58   image: docker
59   services:
60     - name: docker:dind
61       command: ["--insecure-registry=gitlab"]
62   variables:
63     DOCKER_HOST: tcp://docker:2375
64   before_script:
65     - docker login -u gitlab-ci-token -p
66   script:
67     - version=$(docker run --rm -t -v "${CI_PROJECT_PATH}:/project"
68     - tagname=${CI_COMMIT_REF_SLUG}
69     - imagename="$GITLAB_REGISTRY/${CI_PROJECT_PATH}:${version}-${tagname}"
70     - docker build -t "$imagename" .
71     - docker push "$imagename"
```

```
1  # https://hub.docker.com/r/ciricihq/gitlab-sonar-scanner/
2  sonarqube:
3    stage: analysis
4    image: ciricihq/gitlab-sonar-scanner
5    variables:
6      SONAR_ANALYSIS_MODE: publish
7    script:
8      - gitlab-sonar-scanner -Dsonar.login=$SONAR_TOKEN
9      # - gitlab-sonar-scanner -Dsonar.login=$SONAR_TOKEN -Dsonar.analysis.mode=preview -Dsonar.gitlab.project_id=$CI_PROJECT_PATH -Dsonar.gitlab.token=$SONAR_TOKEN
10   artifacts:
11     name: "${CI_JOB_ID}_${CI_JOB_NAME}"
12     expire_in: 1 week
13     paths:
14       - codeclimate.json
15   only:
16     variables:
17       - $SONAR_URL
18       - $SONAR_TOKEN
19     refs:
20       - master
21
22   license-finder:
23     stage: analysis
24     image:
25       name: licensefinder/license_finder
26     before_script:
27       - source /etc/profile.d/rvm.sh
28     script:
29       - license_finder report --format html --decisions_file license_finder_decisions.yml --save license_finder_report.html
30   artifacts:
31     paths:
32       - "license_finder_report.html"
```

# 03

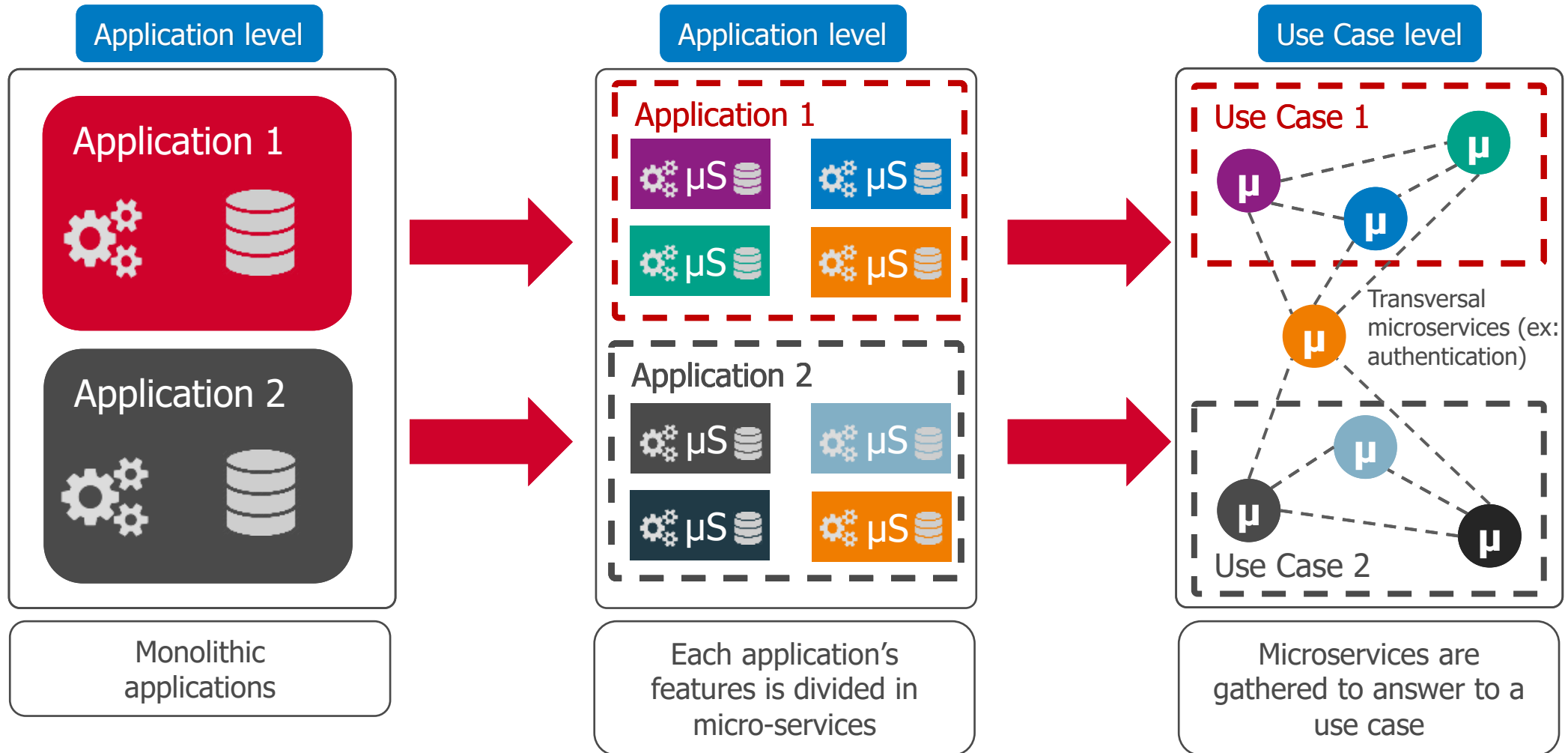
**Pour aller plus loin**

# Cloud native approach

## MONOLITHIC VS MICROSERVICES ?



Cloud Native Approach





# cloud native approach

## THE LEADING CLOUD NATIVE ARCHITECTURE



Cloud Native Approach



### Packaged in containers

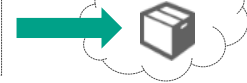
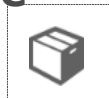
- Containers provide **isolation context**

- Containers are **Fast to create**



- Easily portable**

from one environment to the other



- Scalable**

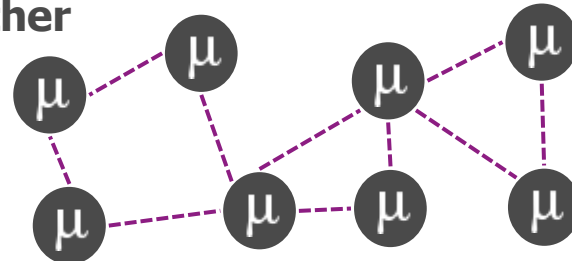


### Microservices architecture

- Each application is a **collection of small services** that can be :

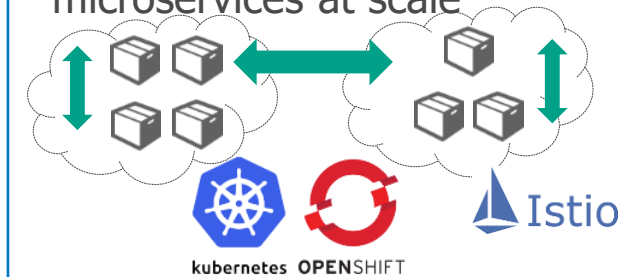
- Built**
- Deployed**
- Operated**

**Independently from each other**



### Dynamically Managed in Cloud

- Containers are orchestrated to use the power of Cloud computing:
  - Scalability**
  - Resilience**
  - Resources optimization**
- Service Mesh allow to handle microservices at scale



These principles are not the only one allowing to build a Cloud Native architecture,  
but they are leading the way

# 04

## Conclusion

# What about your project context ?

## PROJECT ASSESSMENT

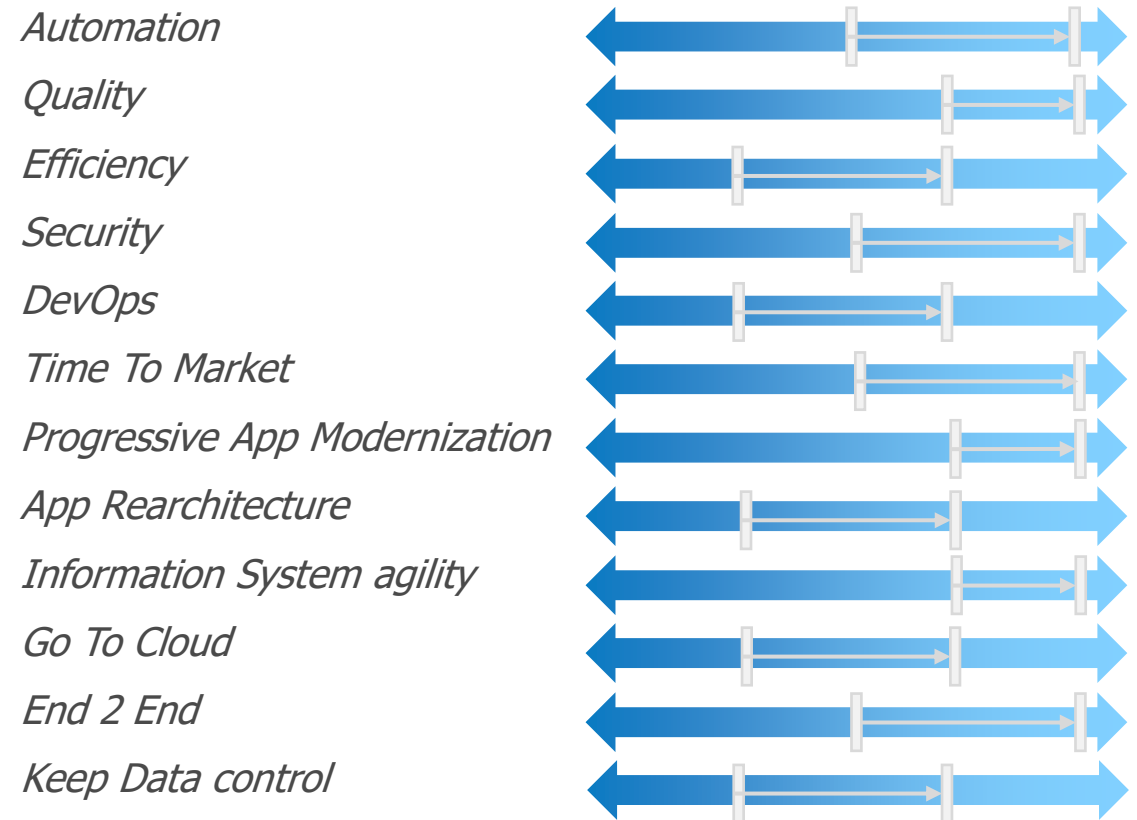


What about your  
project  
context ?

What makes sense and provides  
benefits in my context?

How I am going to use DEP?

### Assess added value



# What about your project context ?

## PROJECT ASSESSMENT



What about your  
project  
context ?

	Development Hybrid Cloud	Production Hybrid Cloud	DevOps Tools	Code Marketplace	Containers orchestrator	Micro-services stack
<i>Improvement areas</i>						
<i>Automation</i>	●		●		●	
<i>Quality</i>			●			
<i>Productivity</i>	●		●	●		
<i>Re-use</i>				●		●
<i>Security</i>	●	●	●		●	
<i>DevOps</i>			●			
<i>Time To Market</i>	●		●	●	●	●
<i>Progressive App Modernization</i>					●	●
<i>App Refactoring</i>					●	●
<i>Information System agility</i>					●	●
<i>Go To Cloud</i>	●	●	●		●	●
<i>End 2 End</i>	●	●	●			
<i>Keep Data control</i>	●	●				



# Des questions ?

N'hésitez pas à me contacter:



# Merci.

---

