INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Le DevOps qu'est-ce c'est ? (Ça se mange ?)

Une introduction au DevOps et aux éléments qui permettent sa mise en œuvre

Quentin Perez
quentin.perez@insa-rennes.fr
INFRES 1A - IMT Mines Alès
2024-2025

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Avant de commencer : mais qui suis-je… ?

- Enseignant-Chercheur de l'INSA Rennes en Informatique - Génie Logiciel
- Équipe de recherche DiverSE - Laboratoire IRISA
- Mes domaines de spécialité :
  - **Métriques logicielles**
  - Génie Logiciel empirique
  - **Utilisation d'IA appliquée à la qualité**
  - **DevOps**
  - GreenIT (depuis Janvier 2023)
- Thème de recherche : **consommation énergétique en DevOps** 🌿🌱
- Diplômé docteur en informatique de l'IMT Mines Alès en 2021 (dirs. de thèse : Christelle Urtado et Sylvain Vauttier)
- Geek (nerd ? 🙈) passionné !

DiverSE Diversity-Centric Software Engineering

UMR IRISA

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Quelques informations

- Cette conférence va durée 3h (oui je sais c'est long...) mais elles seront ponctuées par des pauses, des démos et des quiz 😉
- Les slides sont en anglais ... (J'entends déjà certain.e.s râler au fond) mais la présentation est en français 🐓
- **Objectifs:**
  - **Comprendre les concepts du DevOps**
  - **Comprendre pourquoi on est arrivé au DevOps**
  - **Quelles sont les grands pricipes mis en oeuvre**
  - **Pourquoi c'est cool !**

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Conf DevOps | INFRES 1A | Quentin Perez | 2024-2025

Why doing software quality? Some emblematic cases

Inaugural flight of Ariane 5, the VA-501 flight.

370 million dollars (for fireworks...)

Why doing software quality? Some emblematic cases

Therac-25: a radiation therapy machine.

5 official deaths due to acute irradiation syndrome, several people seriously injured due to massive irradiation.

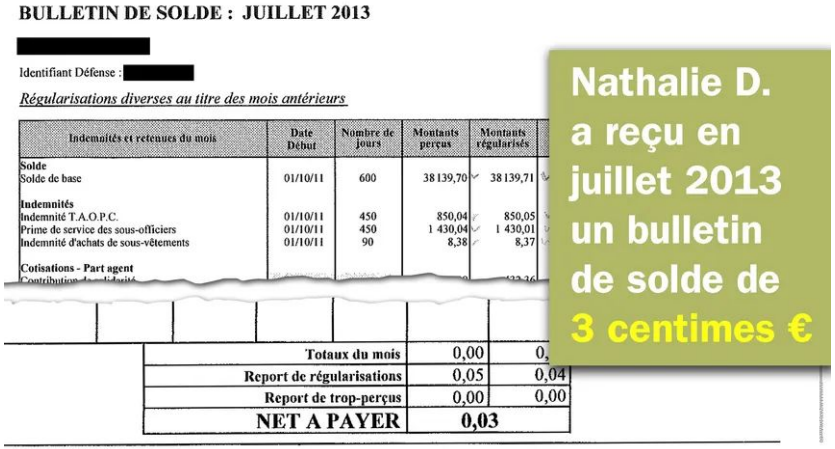

```
Magnet:
    Set bending magnet flag
    repeat
        Set next magnet
        Call Ptime
        if mode/energy has changed, then exit
    until all magnets are set
    return
```

Illustration provenant du site hackaday.com

Why doing software quality? Some emblematic cases

Louvois, the French army's payroll management system.

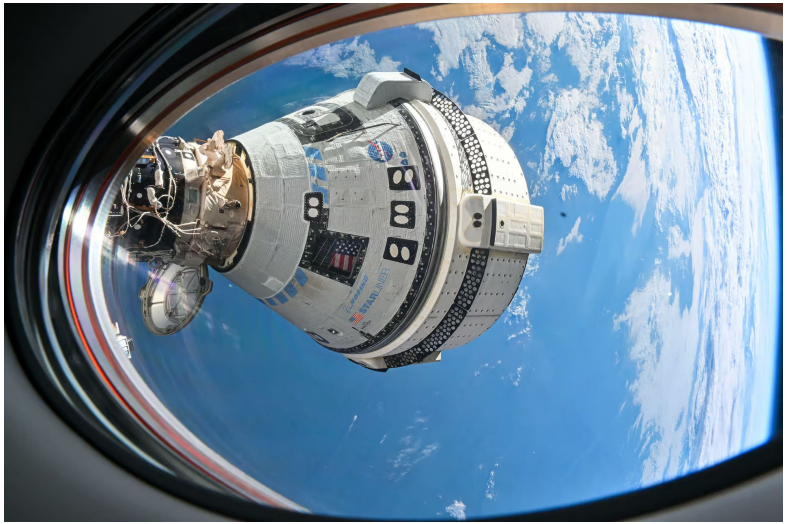
Thousands of salaries not paid or paid incorrectly. Various financial and social repercussions for thousands of people



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

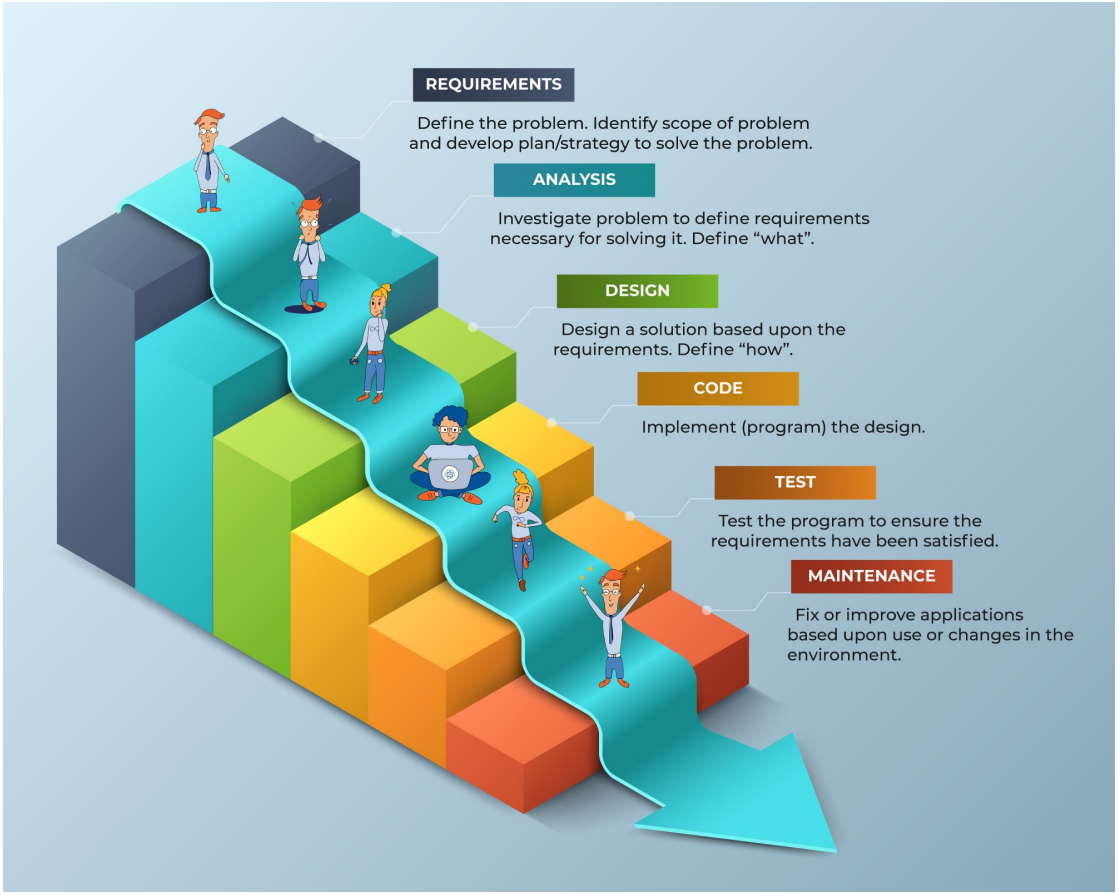
Why doing software quality? Some emblematic cases

Boeing Crew Space Transportation-100 Starliner: a spacecraft designed to transport crew to and from the International Space Station (ISS) and other low-Earth-orbit destinations.
Multiple design issues and the software cannot be maintained outside the module...

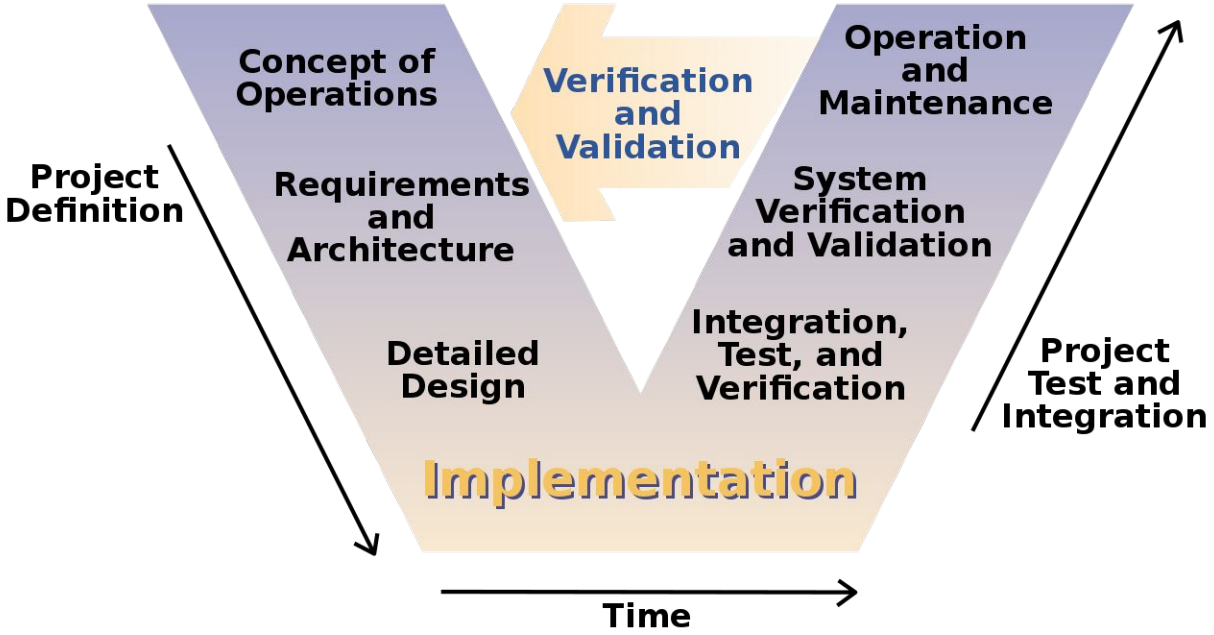


INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

But before DevOps and agile dev'? Waterfall development



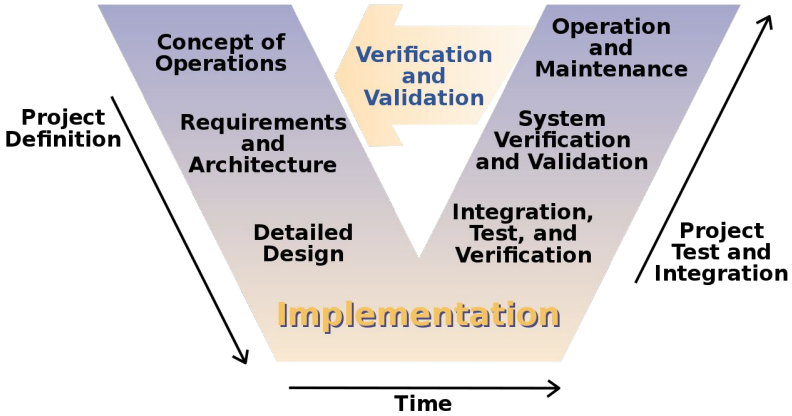
But before DevOps agile dev'? The famous V-Model (FR: Cycle en V)



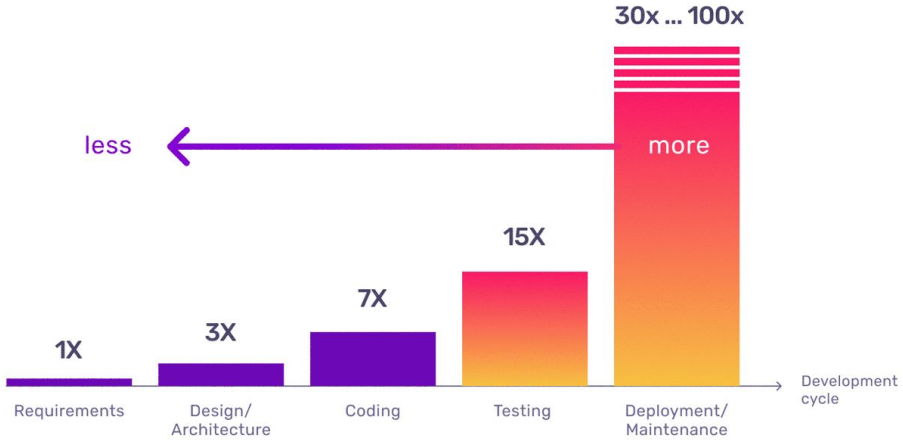
https://fr.wikipedia.org/wiki/Fichier:Devops-toolchain.svg

But some issues appear with “linear method”

1. **Cost of defects** and the late defect detection



Cost of Defects

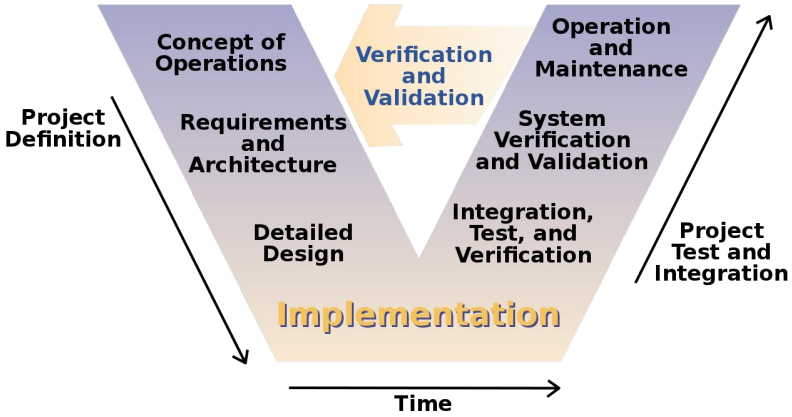


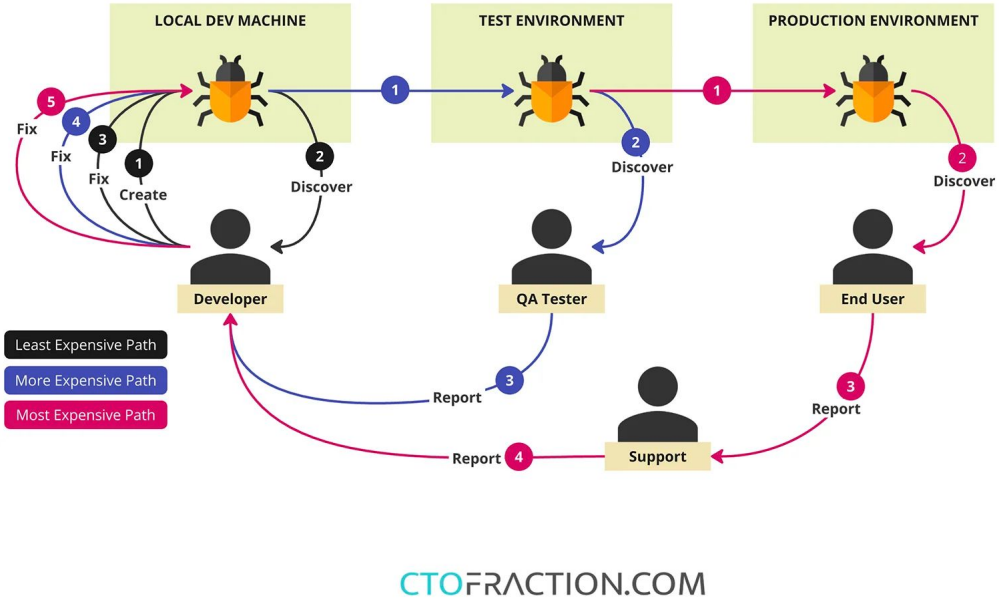
The more time we save your team, the more time they have to find bugs sooner.

That Saves Money

But some issues appear with “linear method”

1. **Cost of defects** and the late defect detection





INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps - Test, principles and pitfalls

1. **Cost of defects** and the late defect detection
2. **Tunnel effect** : stakeholders have no vision or information about the development

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

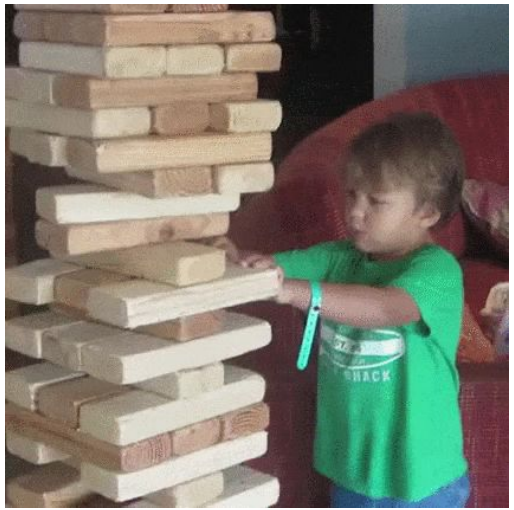
DevOps - Test, principles and pitfalls

1. **Cost of defects** and the late defect detection
2. **Tunnel effect** : stakeholders have no vision or information about the development ⇒ **Expectation VS Reality effect**

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps - Test, principles and pitfalls

1. **Cost of defects** and the late defect detection
2. **Tunnel effect**: stakeholders have no vision or information about the development
3. **Lack of flexibility** in features development



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

2001 Agile Manifesto

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

https://agilemanifesto.org/

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Agile model (SCRUM) Versus V-Model

	V-Model	SCRUM
Lifecycle	Sequential	Iterative and incremental
Delivery	Only at the end of the V ⇒ Late delivery	Regular, each version is a functional and partial version of the final software
Quality Control	Only after the development phase ⇒ Tunnel effect	For each partial version all along the incremental development
Specification	Modifications can only be achieved by going back over all the phases upstream of the specifications ⇒ lack of agility, more costly	More agile, specifications can change during sprints and new features can be added to the next version
Planning	Detailed plans for the whole V-Model, each phase is time planned before the development	Adaptive planning according to the requirements and features to implement

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

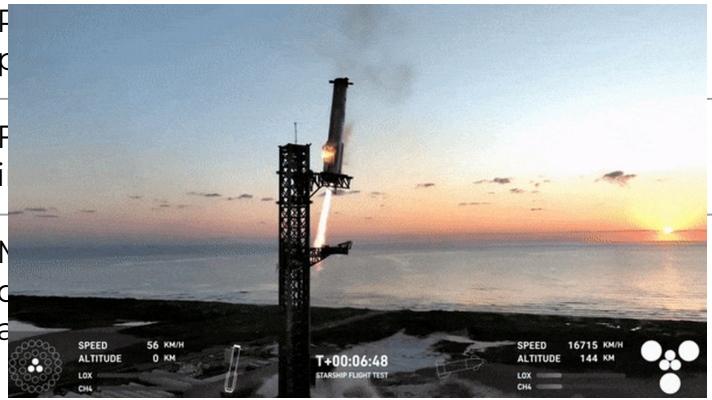
Agile model (SCRUM) Versus V-Model

	V-Model	SCRUM
Lifecycle	Sequential	Iterative and incremental
Delivery		R p
Quality Control		F ir
Specification		M d a
Planning	before the development	A requirements and features to implement

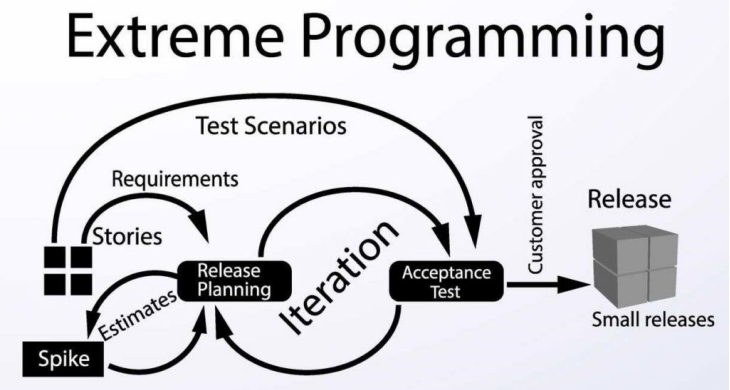
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

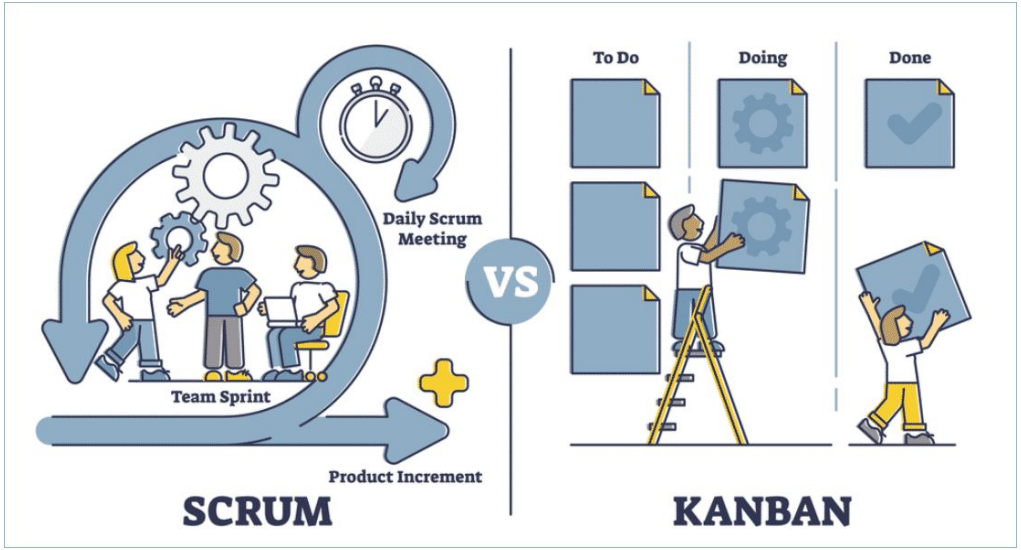
Agile model (SCRUM) Versus V-Model

	V-Model	SCRUM
Lifecycle	Sequential	Iterative and incremental
Delivery		
Quality Control		
Specification		
Planning	before the development	Adaptive planning according to the requirements and features to implement



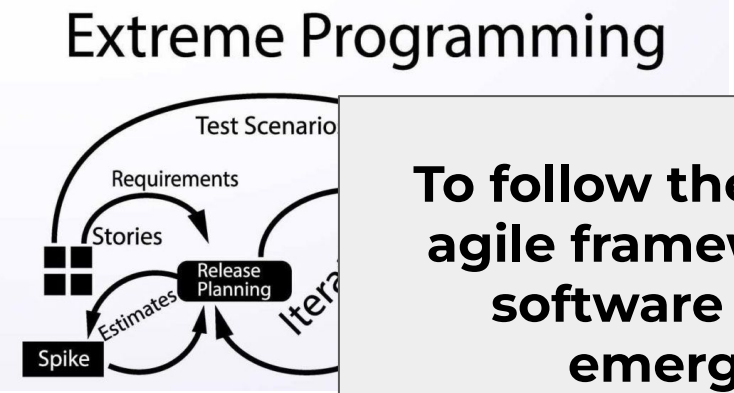
Agiles methods (not exhaustive list)



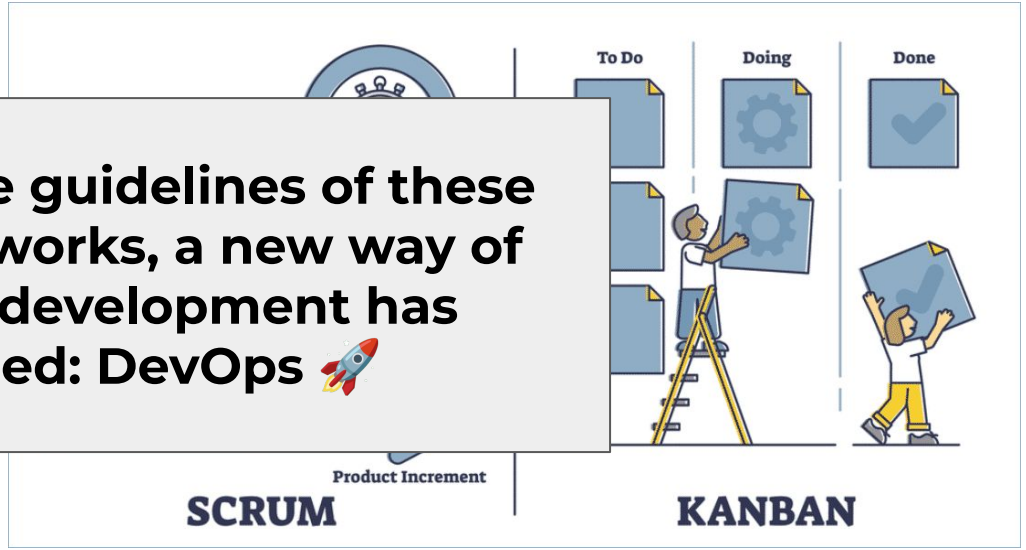


INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Agile methods (not exhaustive list)

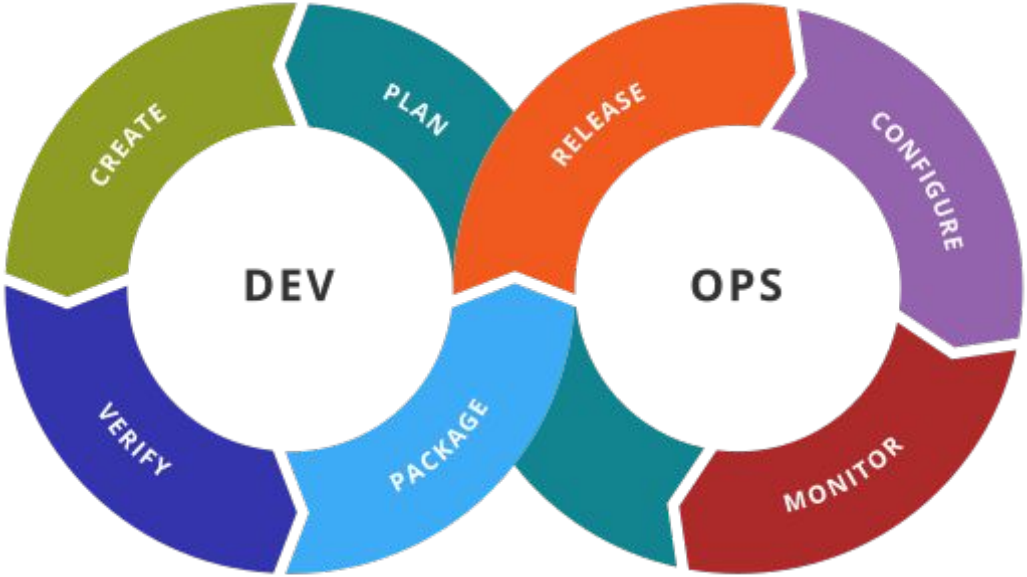


To follow the guidelines of these agile frameworks, a new way of software development has emerged: DevOps 🚀



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps? Tell me more...



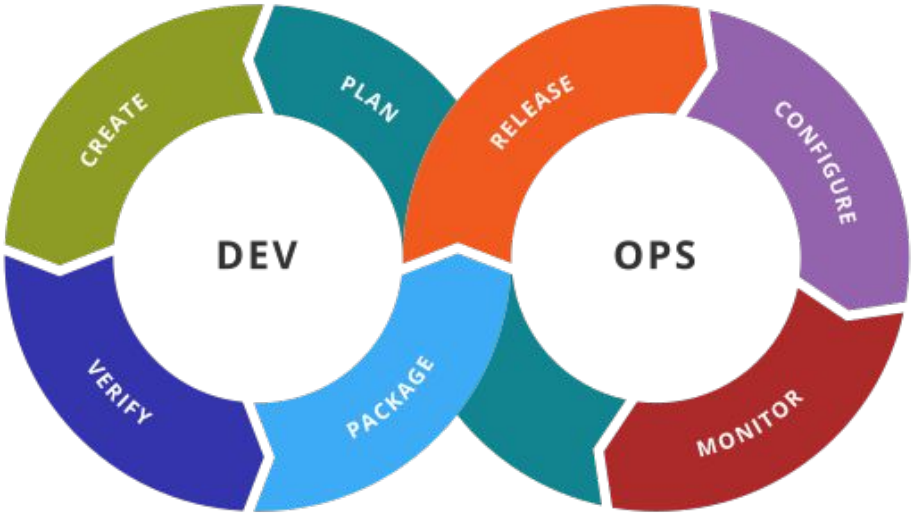
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps? Tell me more...

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps? Tell me more...

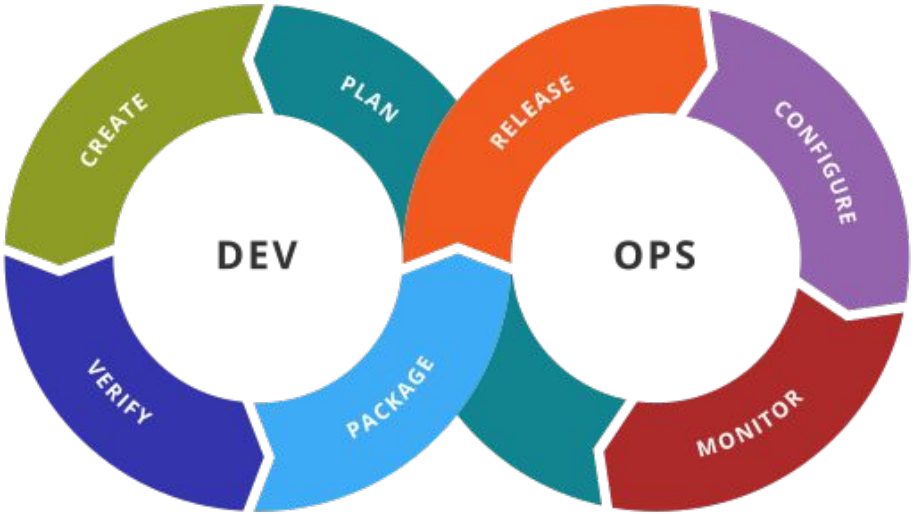
- DevOps is both a methodology of software development and a set of practices and tools



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps? Tell me more...

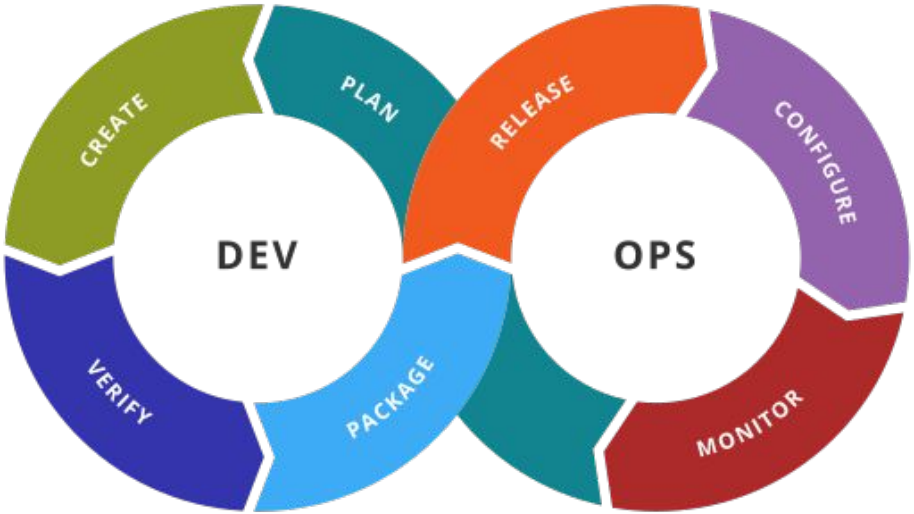
- DevOps is both a methodology of software development and a set of practices and tools
- **Goals:**
  - **Better Quality**
  - **Shorter release cycles**
  - **Reconnect Ops to Dev**
  - **Automation**



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps? Tell me more...

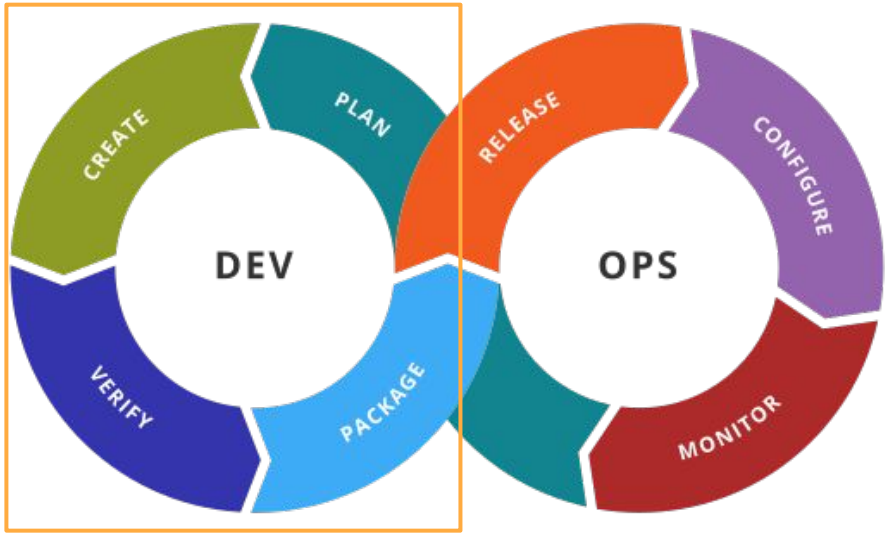
- **Goals:**
  - **Better quality**
  - **Shorter release cycles**
  - **Reconnect Ops to Dev**
  - **Automation**
- Promotes the **automation** and monitoring **all along the cycle**
- Uses **monitoring feedback** to **improve the upstream phases and deliverables**



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps? Tell me more...

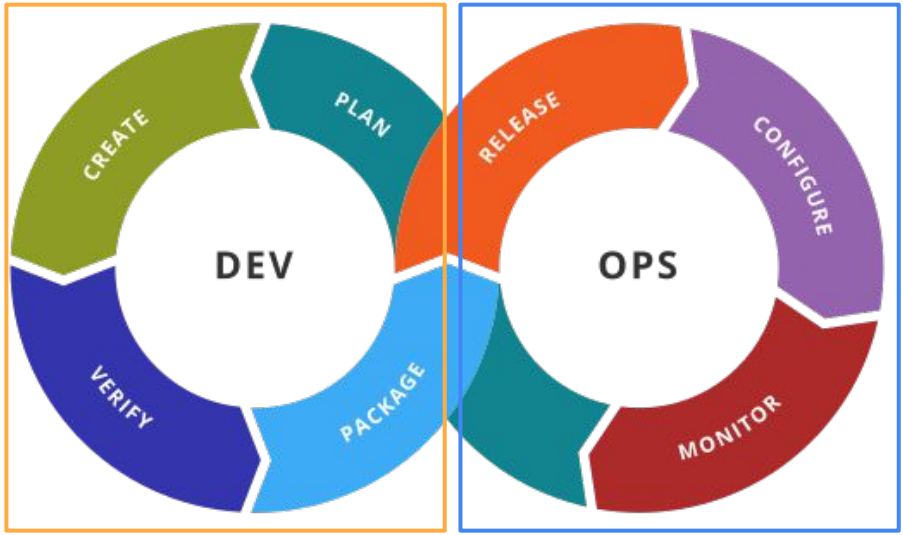
- **Used in support of agility development models** (SCRUM, XP, etc.)
- 2 phases:
  - **DEV ⇒ Software Development**



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

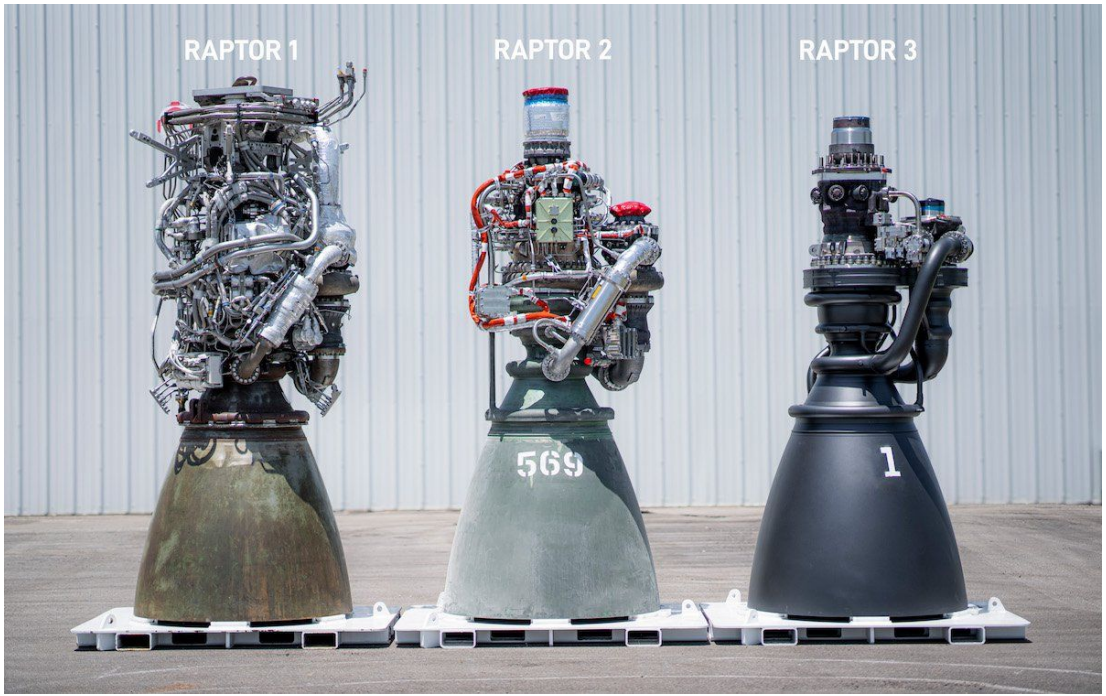
DevOps? Tell me more...

- **Used in support of agility development models** (SCRUM, XP, etc.)
- 2 phases:
  - **DEV ⇒ Software Development**
  - **OPS ⇒ Operations (FR : exploitation)**



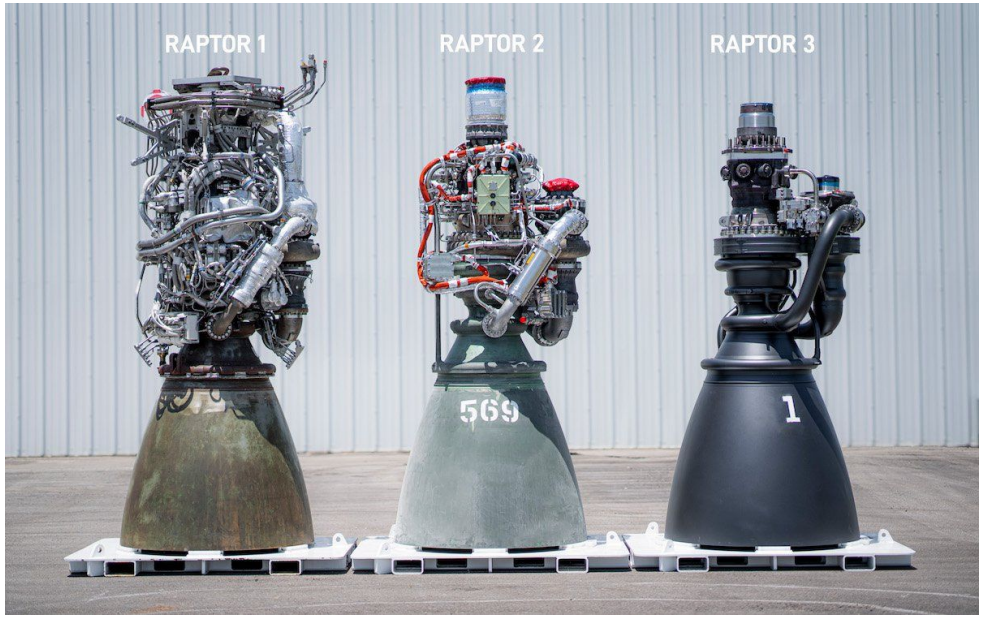
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Agility in rocket propulsion development: Raptor (SpaceX) ⇒ KISS principle



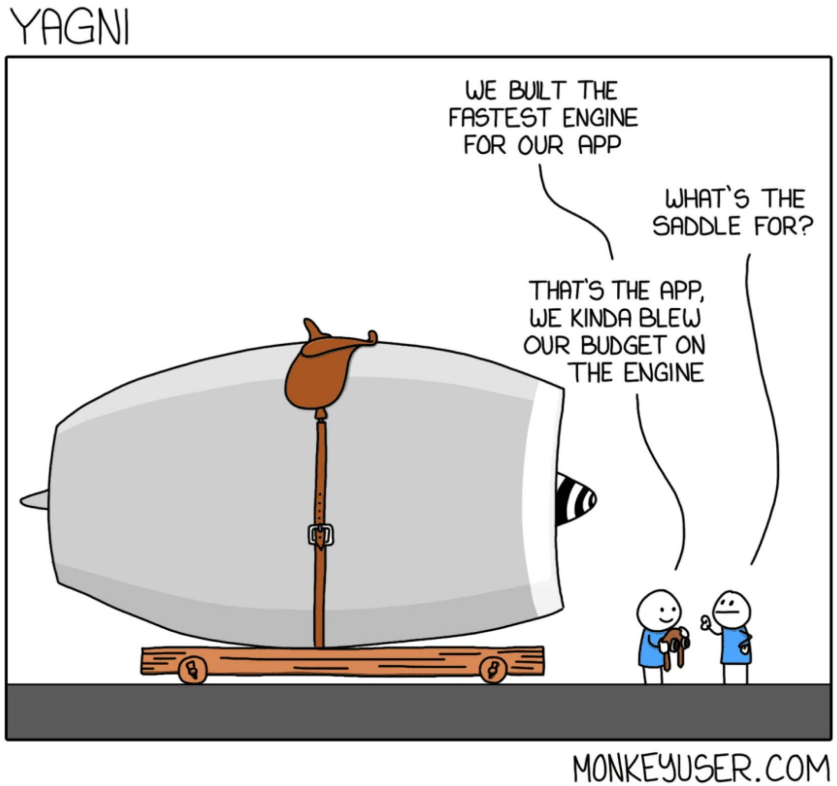
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Agility in rocket propulsion development: Raptor (SpaceX) ⇒ KISS principle



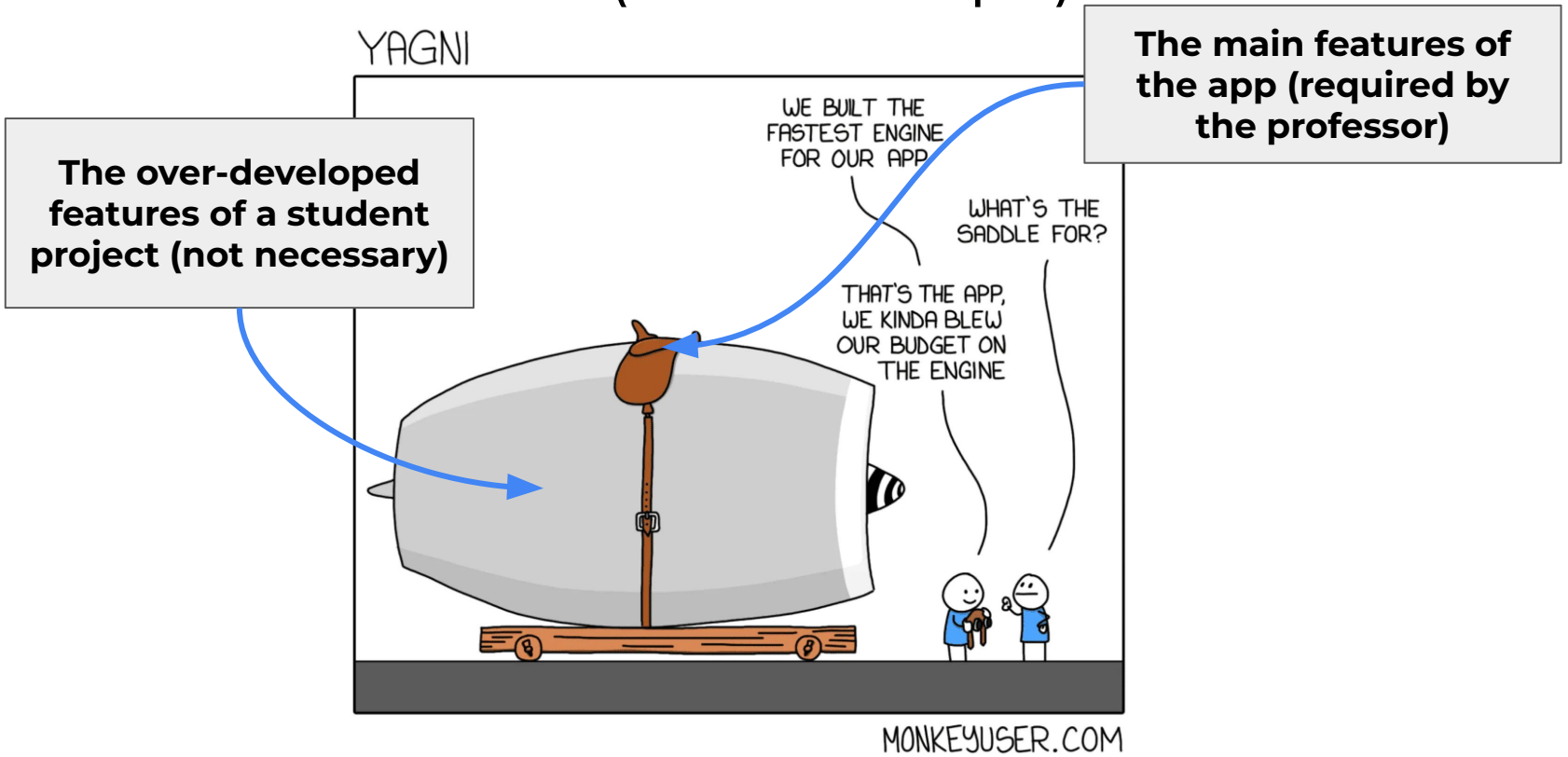
Keep It Simple, Stupid! (KISS)

You Aren't Gonna Need It (YAGNI Principle)



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

You Aren't Gonna Need It (YAGNI Principle)

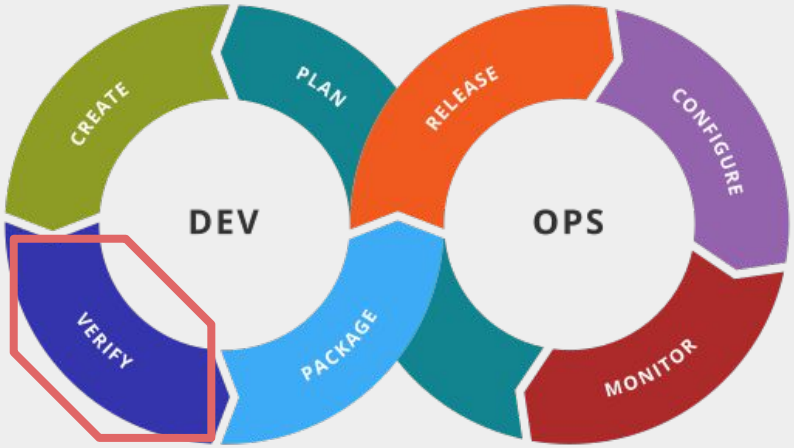


The main features of the app (required by the professor)

The over-developed features of a student project (not necessary)

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

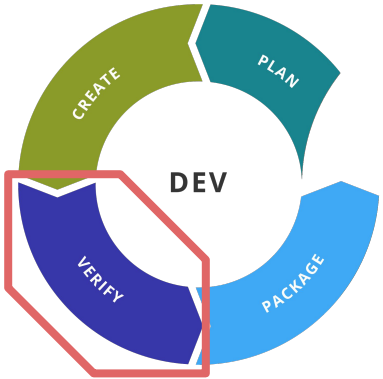
Software testing



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps - Test

(Allegory of version deployed on production without tests)



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps - Test

FR : “Tester c’est douter !”.... ou pas !

Testing provides **confidence** in the software system deployed.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps - Test

FR : “Tester c’est douter !”…. ou pas !

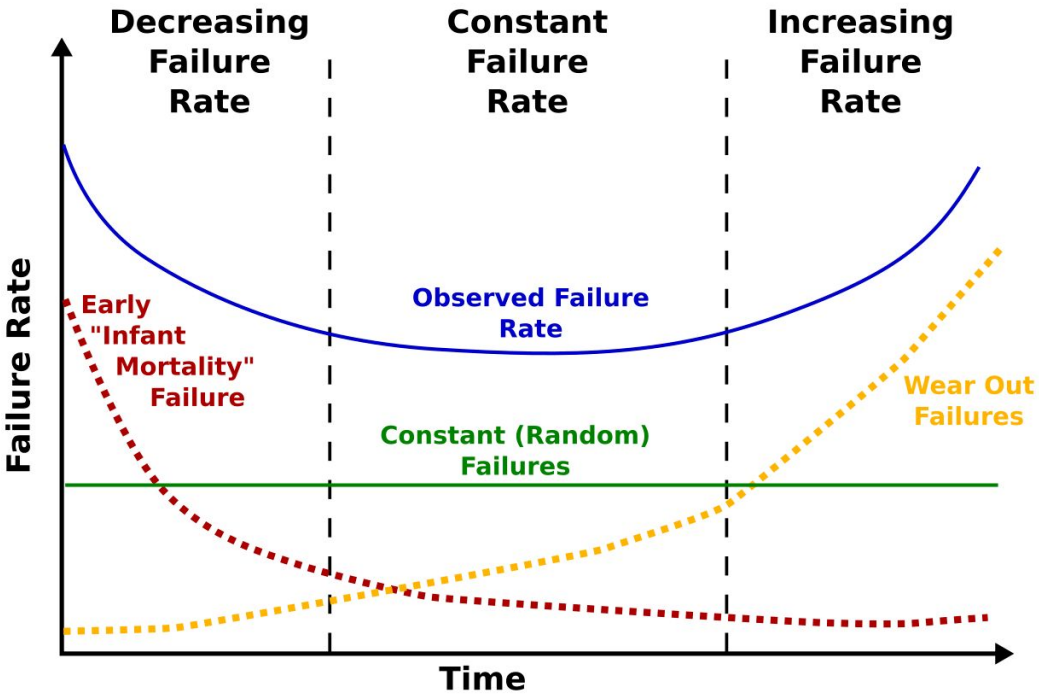
Testing provides **confidence** in the software system deployed.

Improving **confidence** in software system = **improving tests suit**.

But be careful…

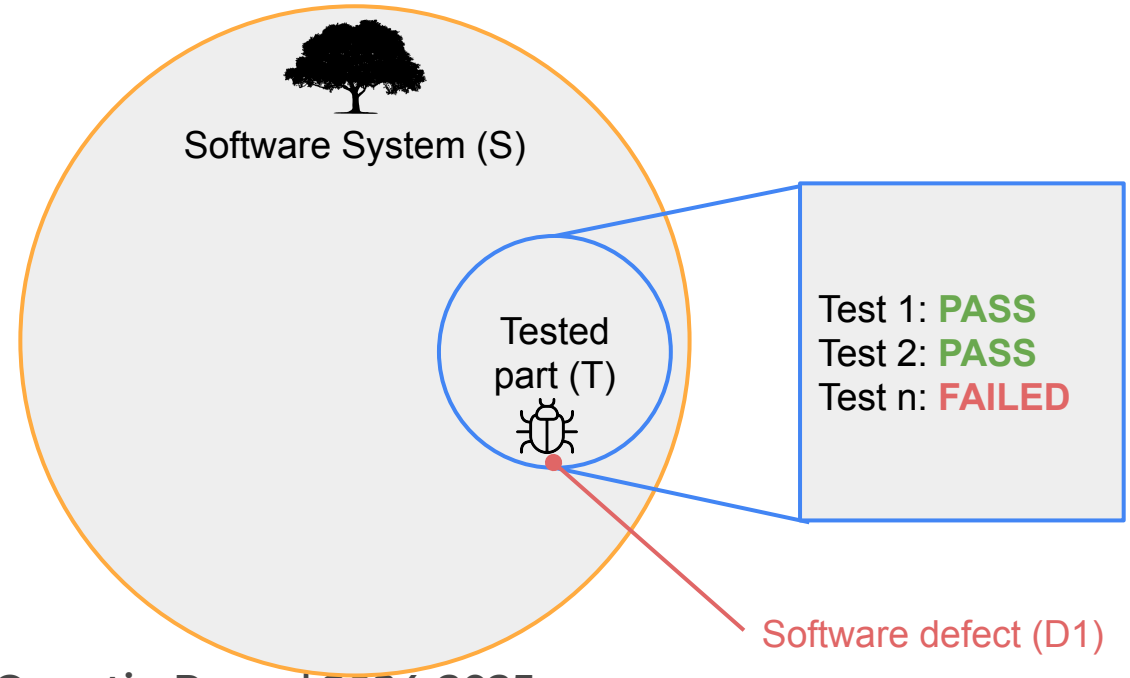
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

As a reminder: bathtub curve' hazard function



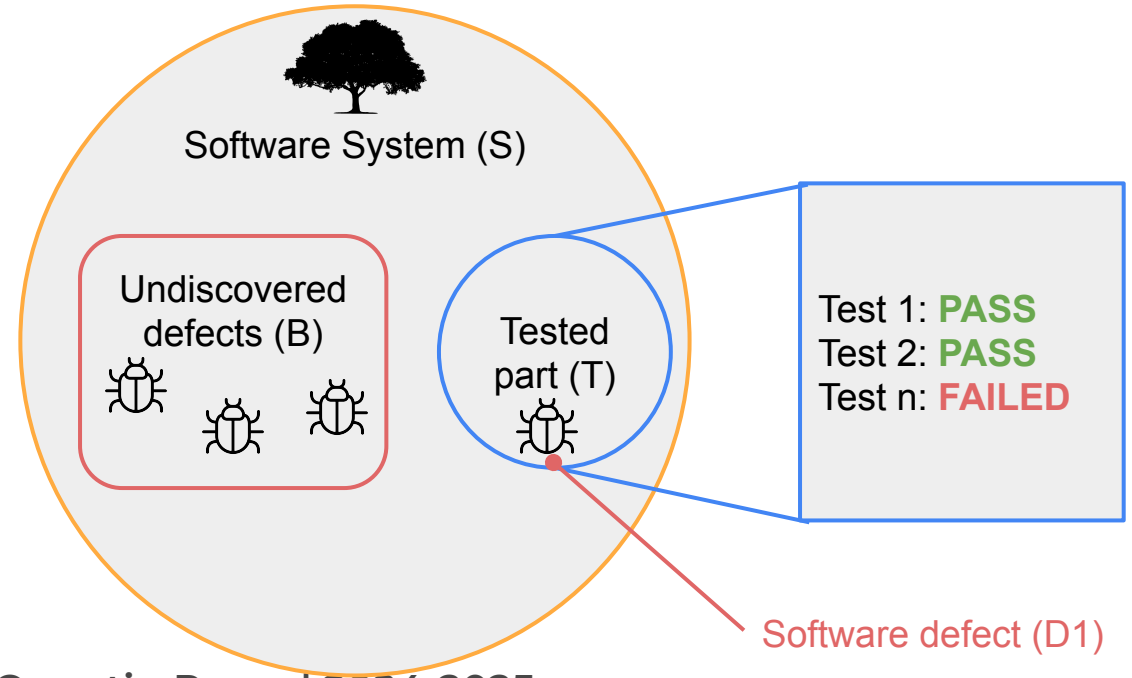
DevOps - Test, principles and pitfalls

1. **Tests show the presence of defect(s) not their absence(s)**



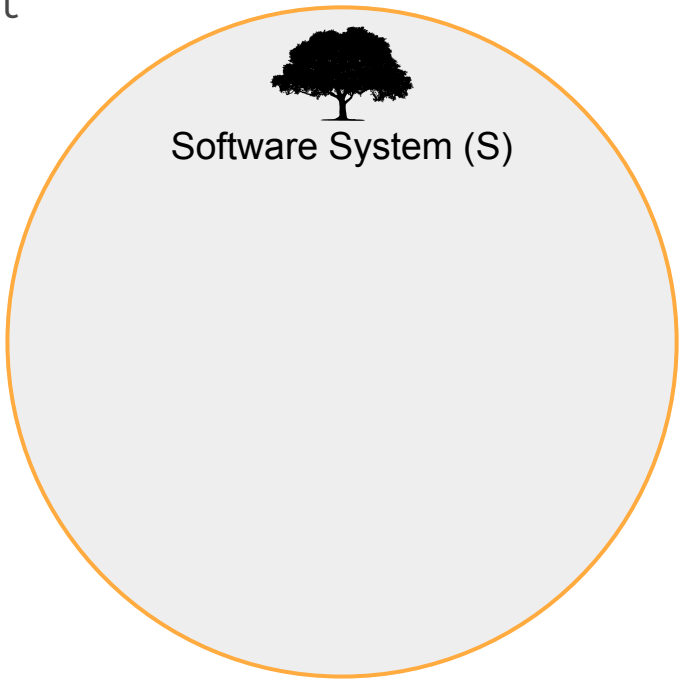
DevOps - Test, principles and pitfalls

1. **Tests show the presence of defect(s) not their absence(s)**



DevOps - Test, principles and pitfalls

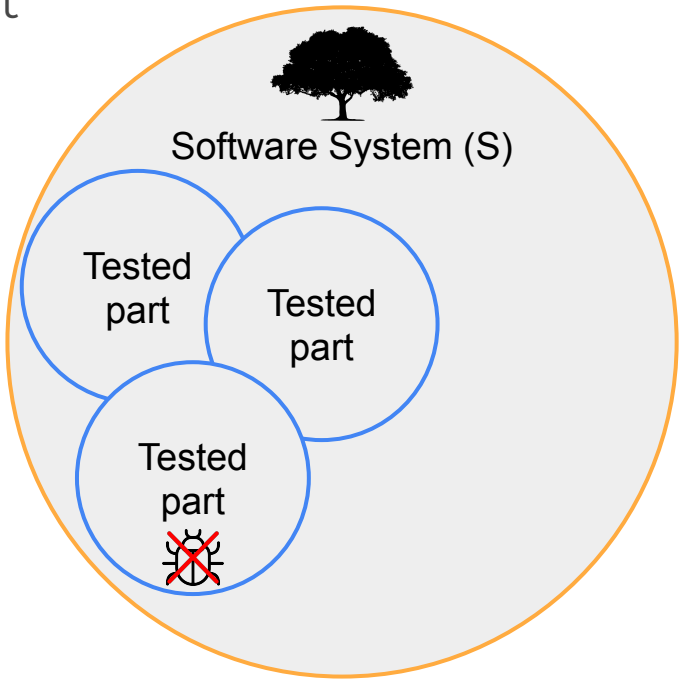
1. Tests show the presence of defect(s) not their absence(s)
2. **Exhaustive testing is impossible**



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

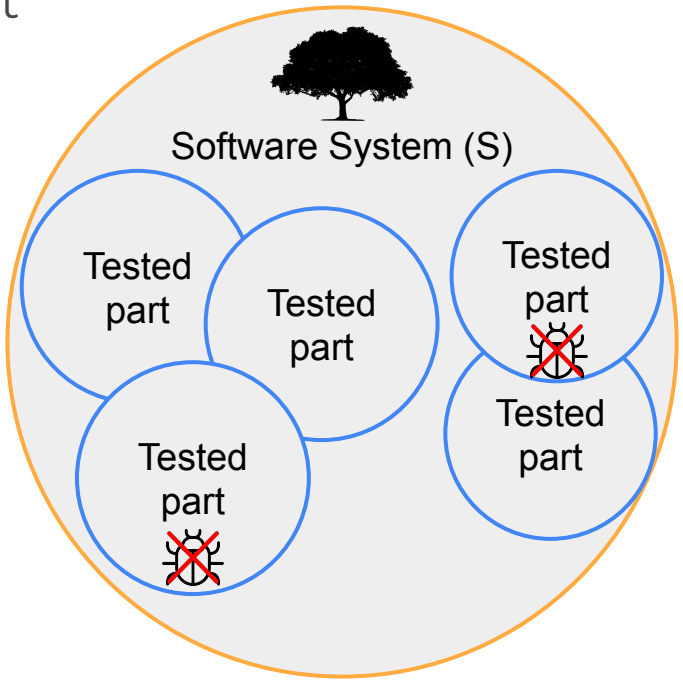
DevOps - Test, principles and pitfalls

1. Tests show the presence of defect(s) not their absence(s)
2. **Exhaustive testing is impossible**



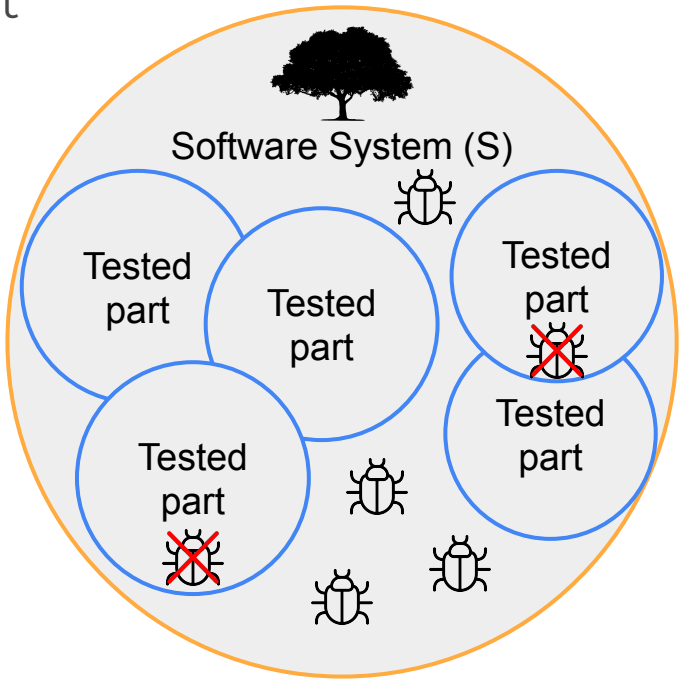
DevOps - Test, principles and pitfalls

1. Tests show the presence of defect(s) not their absence(s)
2. **Exhaustive testing is impossible**



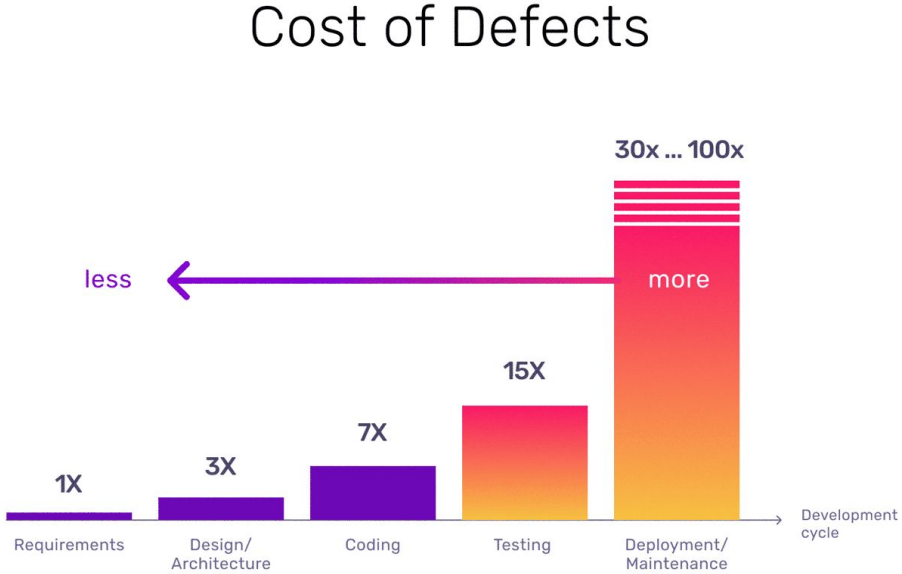
DevOps - Test, principles and pitfalls

1. Tests show the presence of defect(s) not their absence(s)
2. **Exhaustive testing is impossible**



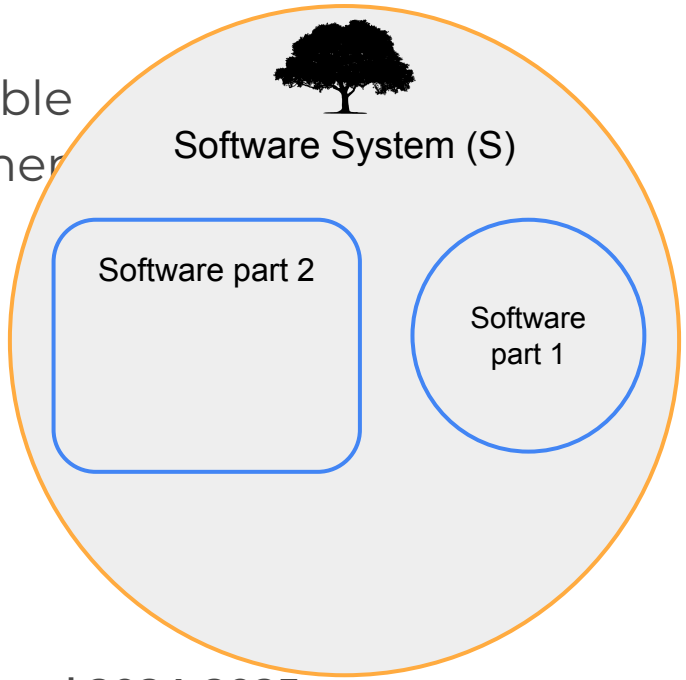
DevOps - Test, principles and pitfalls

1. Tests show the presence of defect(s) not their absence(s)
2. Exhaustive testing is impossible
3. **Testing early in the development**
    - Minimization of correction and refactoring costs



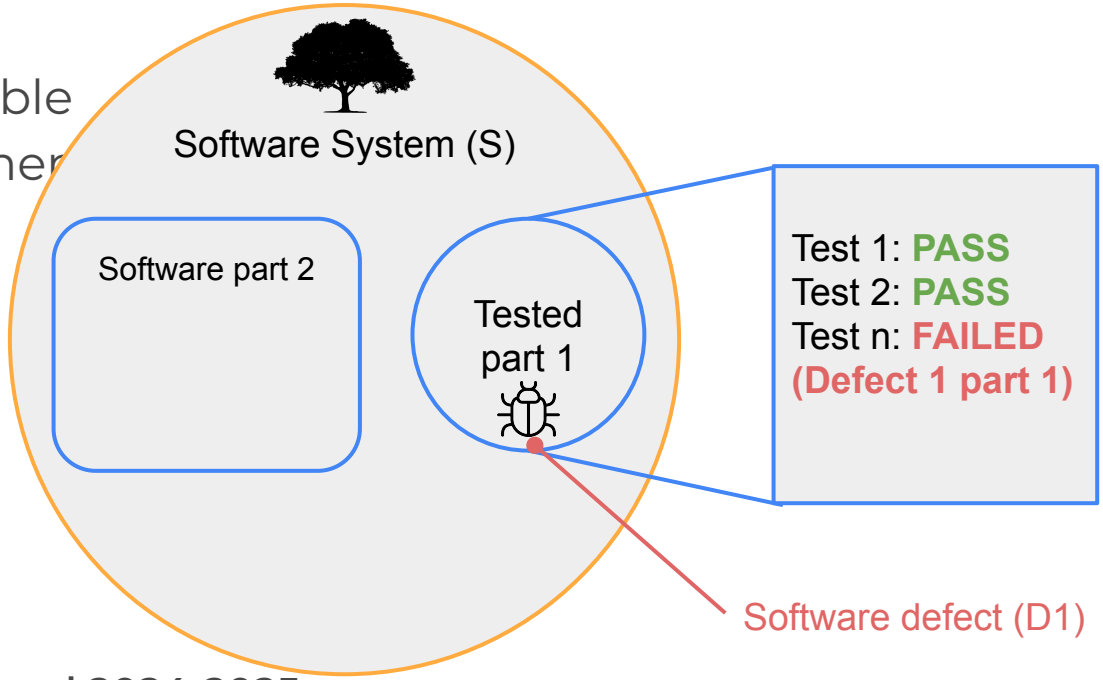
DevOps - Test, principles and pitfalls

1. Tests show the presence of defect(s) not their absence(s)
2. Exhaustive testing is impossible
3. Testing early in the developmer
4. **Defect clustering**



DevOps - Test, principles and pitfalls

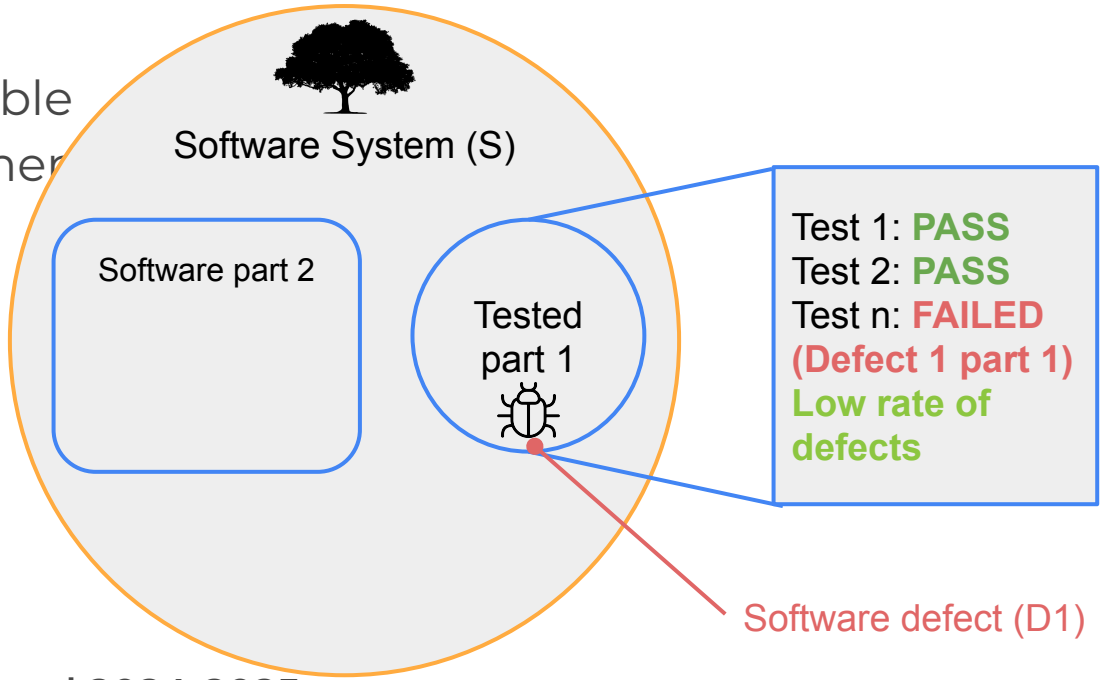
1. Tests show the presence of defect(s) not their absence(s)
2. Exhaustive testing is impossible
3. Testing early in the developmer
4. **Defect clustering**



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

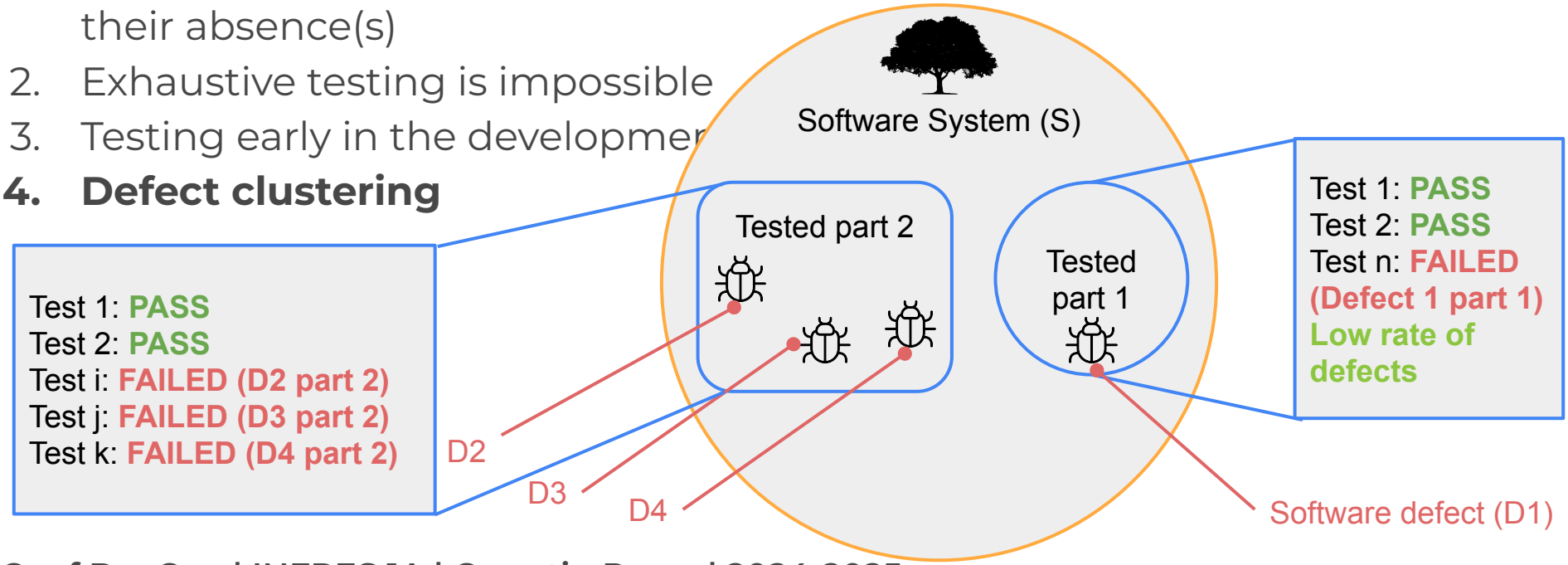
DevOps - Test, principles and pitfalls

1. Tests show the presence of defect(s) not their absence(s)
2. Exhaustive testing is impossible
3. Testing early in the developmer
4. **Defect clustering**



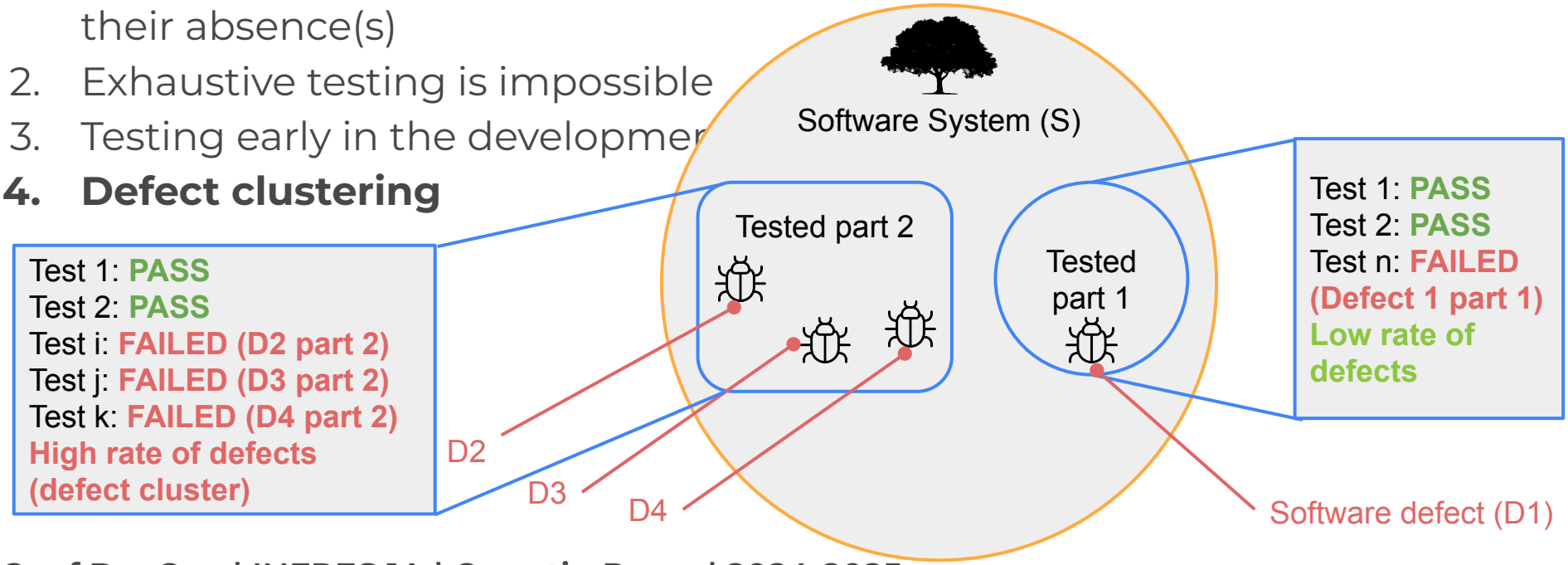
DevOps - Test, principles and pitfalls

1. Tests show the presence of defect(s) not their absence(s)
2. Exhaustive testing is impossible
3. Testing early in the developmer
4. **Defect clustering**



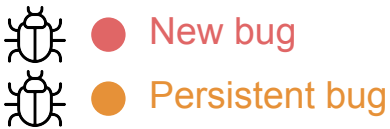
DevOps - Test, principles and pitfalls

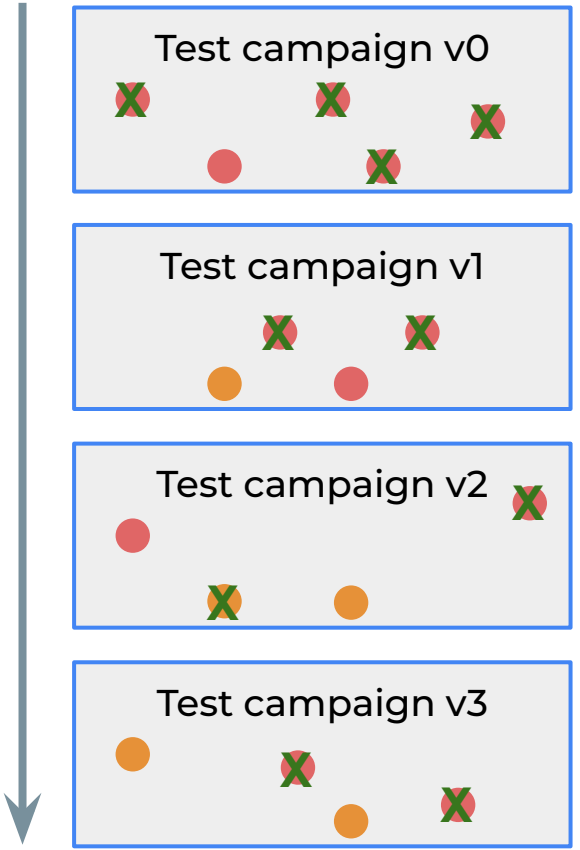
1. Tests show the presence of defect(s) not their absence(s)
2. Exhaustive testing is impossible
3. Testing early in the developmer
4. **Defect clustering**



DevOps - Test, principles and pitfalls

1. Tests show the presence of defect(s) not their absence(s)
2. Exhaustive testing is impossible
3. Testing early in the development
4. Defect clustering
5. **Pesticide paradox**





INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps - Test, principles and pitfalls

1. Tests show the presence of defect(s) not their absence(s)
2. Exhaustive testing is impossible
3. Testing early in the development
4. Defect clustering
5. Pesticide paradox
6. **Software testing is context dependent**
    - Searching for defects in a banking software system is not the same as searching for defects in a developer's personal website in the deepest of Nebraska.
    - You can't test a cloud server with a single 1kb file

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps - Test, principles and pitfalls

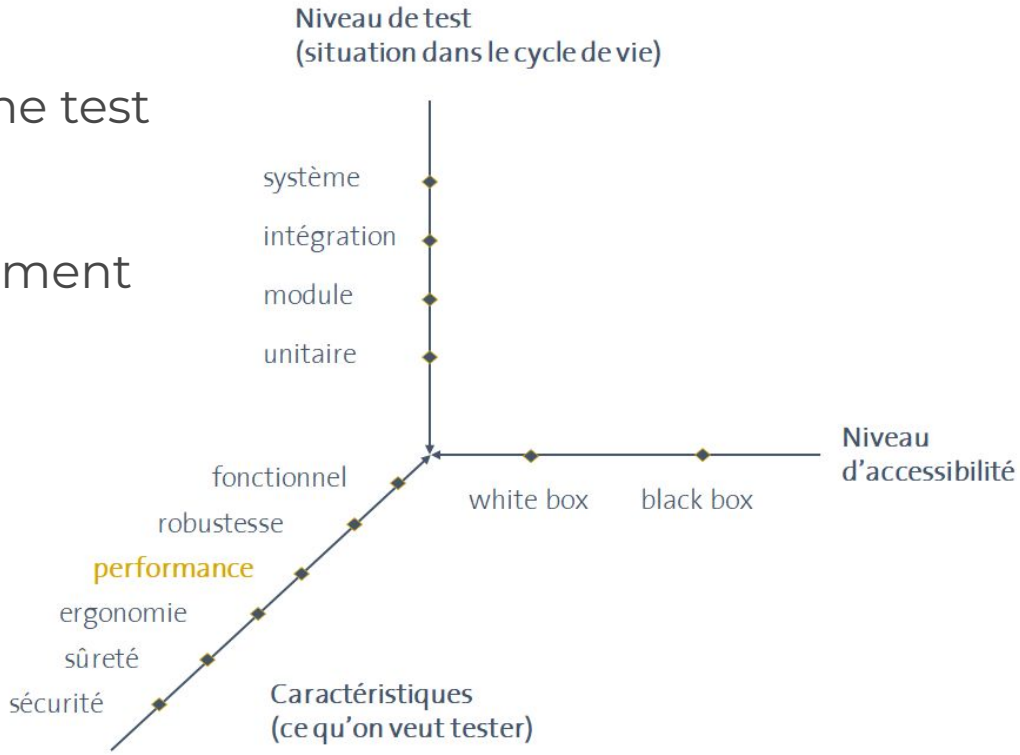
1. Tests show the presence of defect(s) not their absence(s)
2. Exhaustive testing is impossible
3. Testing early in the development
4. Defect clustering
5. Pesticide paradox
6. Software testing is context dependent
7. **The illusion of being flawless**
    - the application may be perfect from a quality point of view, but it may still be unusable for the end user

Artist: Katerina Kamprani

DevOps - Test types and levels

Level of testing: characterizes the test granularity

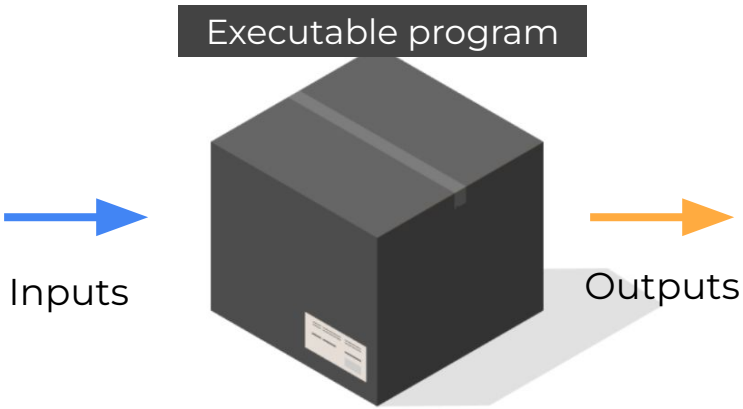
Type of test: defines the requirement to be tested



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps - Test types and levels

Functional testing (black box testing)

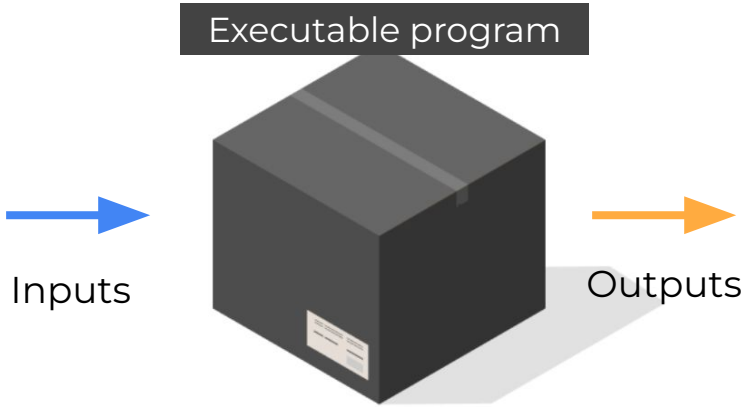


Uses the feature descriptions and documentation

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

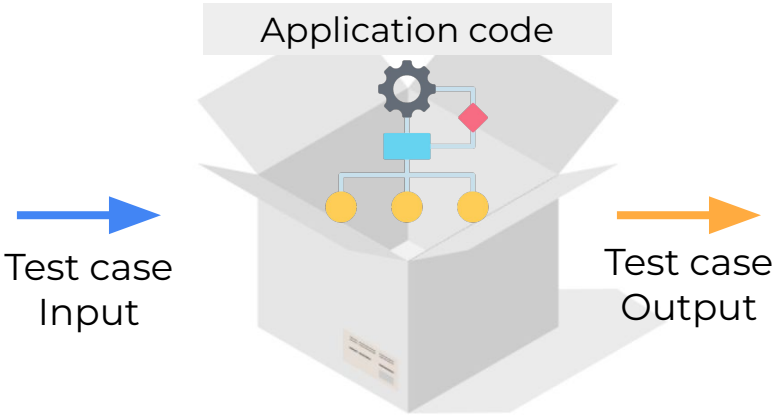
DevOps - Test types and levels

Functional testing (black box testing)



Uses the feature descriptions and documentation

Structural testing (white box testing)



Uses the internal structure of the program

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

DevOps - Test, some examples of frameworks/libs

- Performance : JMeter
- Acceptance / End-to-end (E2E): Cucumber
- GUI Testing: Selenium, Karma, Cypress
- Unit testing: JUnit, Mockito, Hamcrest, etc...

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

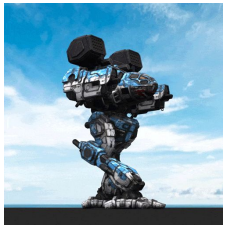
Create fake objects for unit testing: the mocking

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Demo time for unit testing based on Mech robot

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Mocks: how to fake objects?

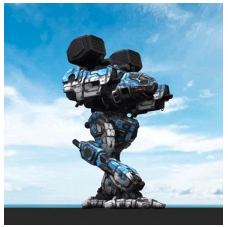


```java
public class Mech {
    2 usages
    private double valeurBlindage;
    1 usage
    private int puissanceBase;
    3 usages
    private CanonLaser canonLaser;
    no usages
    public void setCanonLaser(CanonLaser canonLaser){
        this.canonLaser = canonLaser;
    }
    /**
     * @return Puissance total du mech (chassis + armement)
     */
    no usages
    public int getPuissanceTotalMech(){
        return puissanceBase + canonLaser.getPuissance();
    }
```

```java
public class TestMech {
    3 usages
    Mech mech;
    3 usages
    CanonLaser canonLaser;

    @BeforeEach
    public void setUp() {
        mech = new Mech( valeurBlindage: 85.5,  puissanceBase: 100);
        canonLaser = Mockito.mock(CanonLaser.class);
        mech.setCanonLaser(canonLaser);
    }

    @Test
    public void testGetPuissanceTotale() {
        Mockito.when(canonLaser.getPuissance()).thenReturn( t: 50);
        assertEquals( expected: 150, mech.getPuissanceTotaleMech());
    }
}
```

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Mocks: how to fake objects?



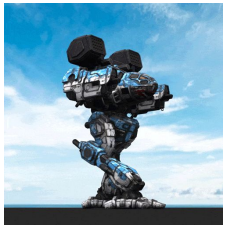
Mock: an object controlled by a framework (Mockito here)

By default **a mock does nothing.** Then, can **fake method calls.**

```java
public class TestMech {
    3 usages
    Mech mech;
    3 usages
    CanonLaser canonLaser;

    @BeforeEach
    public void setUp() {
        mech = new Mech( valeurBlindage: 85.5,  puissanceBase: 100);
        canonLaser = Mockito.mock(CanonLaser.class);
        mech.setCanonLaser(canonLaser);
    }

    @Test
    public void testGetPuissanceTotale() {
        Mockito.when(canonLaser.getPuissance()).thenReturn( t: 50);
        assertEquals( expected: 150, mech.getPuissanceTotaleMech());
    }
}
```

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Mocks: how to fake objects?



Mock: an object controlled by Mockito

By default **a mock does nothing.** Then, can **fake method calls.**

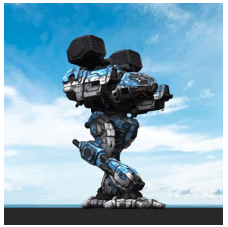
Faking the object CanonLaser

- Mockito creates an empty shell to simulate a “CanonLaser” object
- Breaks cyclic dependency
- No “CanonLaser” initialization

```
public class TestMech {
    3 usages
    Mech mech;
    3 usages
    CanonLaser canonLaser;

    @BeforeEach
    public void setUp() {
        mech = new Mech( valeurBlindage: 85.5,  puissanceBase: 100);
        canonLaser = Mockito.mock(CanonLaser.class);
        mech.setCanonLaser(canonLaser);
    }

    @Test
    public void testGetPuissanceTotale() {
        Mockito.when(canonLaser.getPuissance()).thenReturn( t: 50);
        assertEquals( expected: 150, mech.getPuissanceTotaleMech());
    }
}
```

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Mocks: how to fake objects?



Mock: an object controlled by Mockito

By default **a mock does nothing.** Then, can **fake method calls.**

Faking the object CanonLaser

- Mockito creates an empty shell to simulate a “CanonLaser” object
- Breaks cyclic dependency
- No “CanonLaser” initialization

Behaviour definition of “getPuissance()”

```
public class TestMech {
    3 usages
    Mech mech;
    3 usages
    CanonLaser canonLaser;

    @BeforeEach
    public void setUp() {
        mech = new Mech( valeurBlindage: 85.5,  puissanceBase: 100);
        canonLaser = Mockito.mock(CanonLaser.class);
        mech.setCanonLaser(canonLaser);
    }

    @Test
    public void testGetPuissanceTotale() {
        Mockito.when(canonLaser.getPuissance()).thenReturn( t: 50);
        assertEquals( expected: 150, mech.getPuissanceTotaleMech());
    }
}
```

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Comment participer ?

Copier le lien de participation

1 Allez sur wooclap.com

2 Entrez le code d'événement dans le bandeau supérieur

Code d'événement
ZZUNEA

Activer les réponses par SMS

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

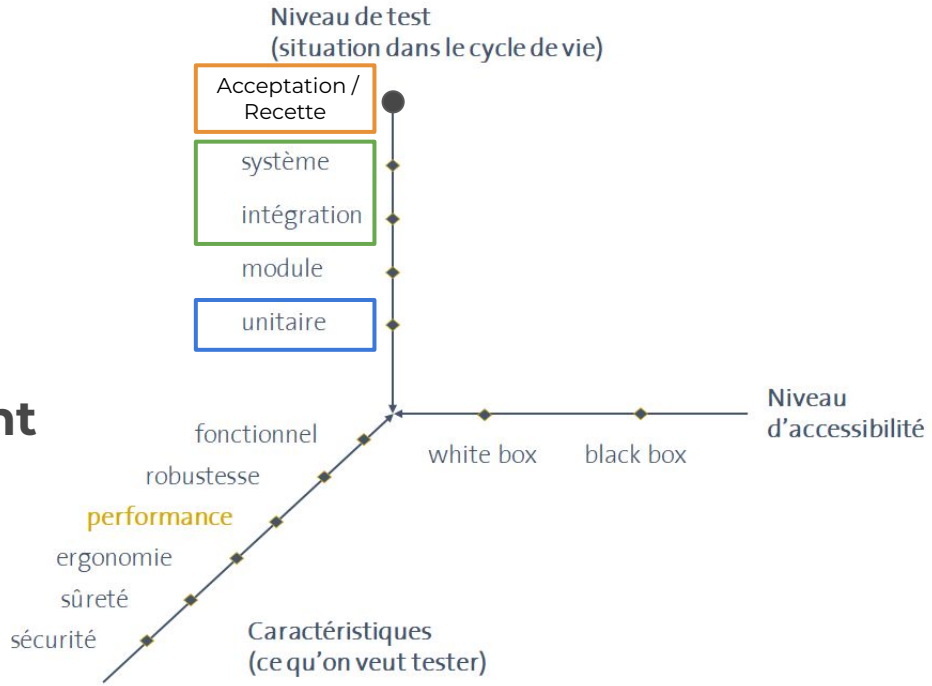
Test Strategies

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Test Strategies

Different software testing strategies are employed depending on the testing levels being worked on.

- **TDD: Test Driven Development → Unit Testing**
- **BDD: Behavior Driven Development → Integration/System testing**
- **ATDD: Acceptance Test Driven Development → Requirement Acceptance Testing**



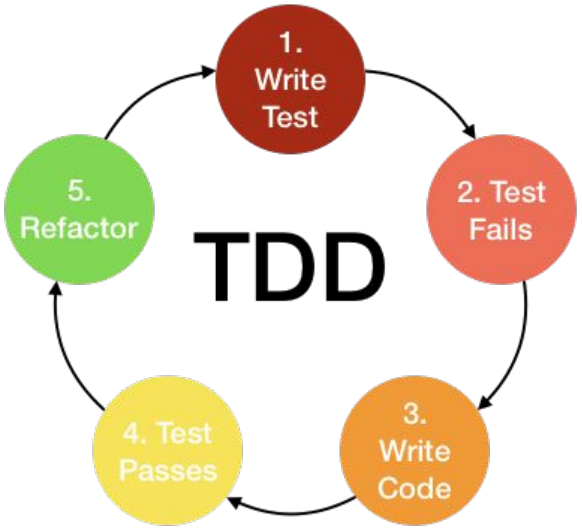
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Test Driven Development

TDD => Unit Testing: Unit testing involves testing a very small part of the software to ensure it functions correctly.

Principe: Writing the test => Writing the code => Test => (Refactor if necessary)

Problem: very clear for developers but not really interpretable by stakeholders (Clients, PO, etc.)



Unit tests, see: Course Software Testing #2 - DevOps

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Behavior Driven Development

BDD => An approach invented by Dan North and first described in an article in Better Software in 2006. **It doesn't focus on an individual unit of the program like Test-Driven Development (TDD), but instead takes a more global and functional approach.**

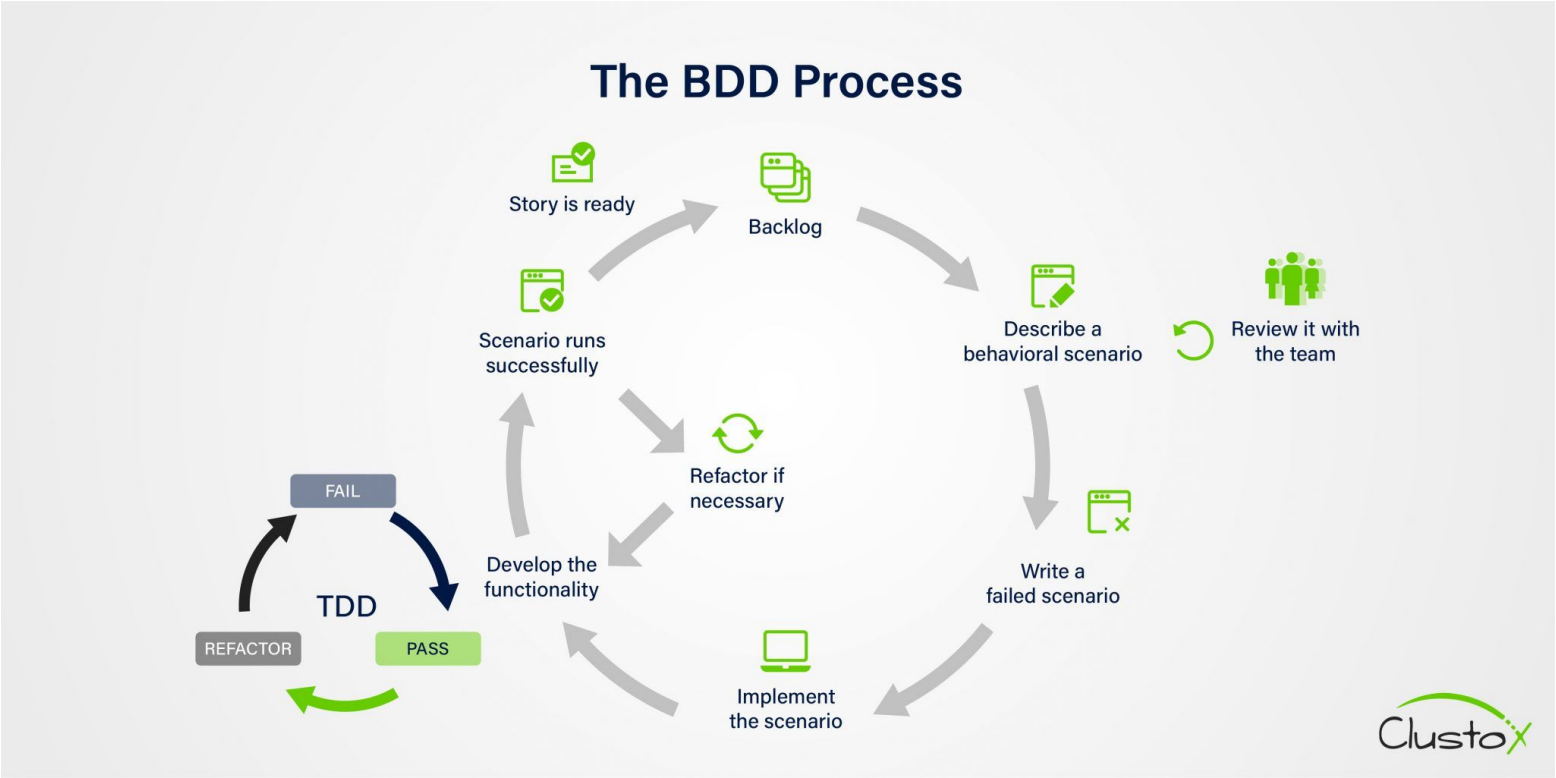
It aims to achieve expressiveness of the requirement through the description of:

- Usage contexts
- Actions to be performed on the system
- Acceptance criteria for the system

It seeks collaboration, understanding, and validation of the system by various stakeholders (Clients, Product Owners, developers, quality analysis teams, etc.).

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Behavior Driven Development In Agile Cycle

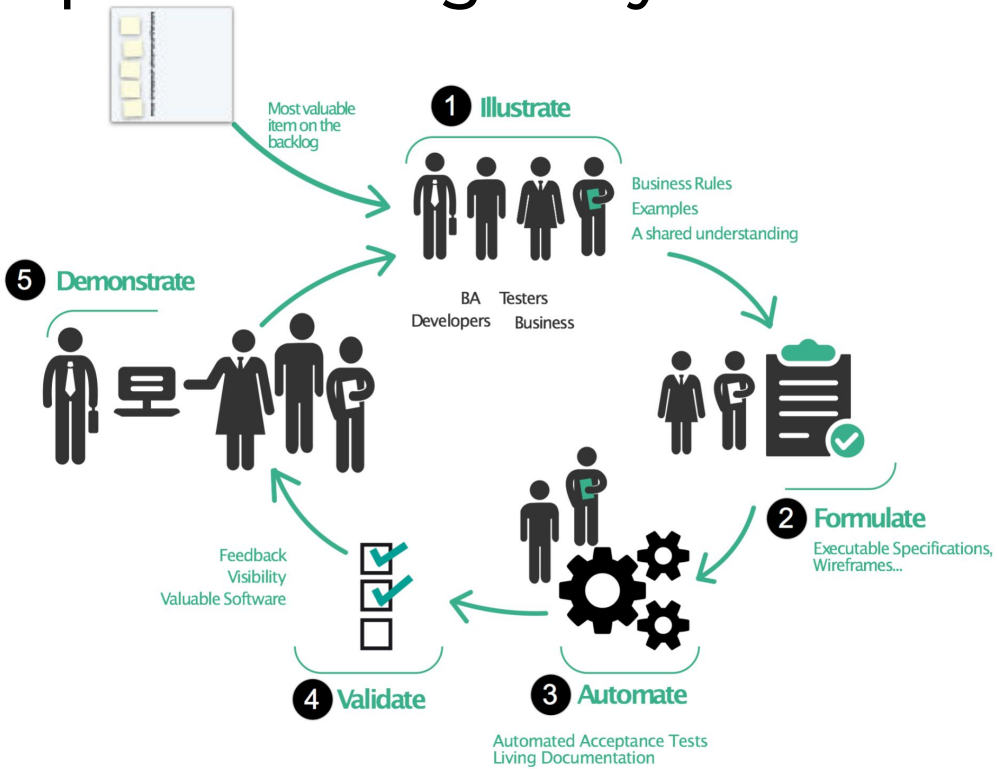


INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Behavior Driven Development In Agile Cycle

BDD has several advantages compared to TDD:

- Provides a "living" project documentation.
- Facilitates natural language communication with stakeholders.
- Clearly outlines the application's functionalities.
- Clearly defines the expected outcomes of testing and the functionality.
- Integrates seamlessly into an agile process through the creation of user stories, similar to a backlog.



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

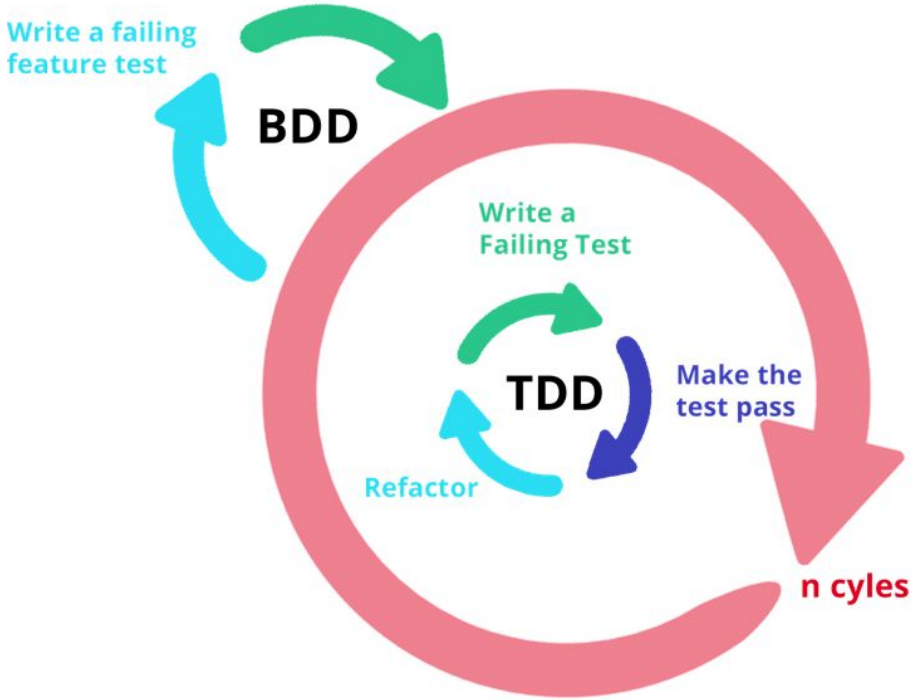
BDD Vs TDD

	BDD	TDD
Stakeholders	Developers, **QA Team, Clients, PO**, SM, etc.	Developers, Scrum Master
Language	**Simplification (subset) of natural language**	Programming language(s) associated with the test technology(ies)
Implementation level	**High Level (Behavior)**	Low-Level (purrely code implementation)
Development steps for testing	**Discussion of features/scenarios,** implementation, testing and refactoring	Implementation / test / refactoring
Documentation used	System requirements, **acceptance criteria**	Implementation requirements

However, BDD and TDD are complementary

BDD : specify and test system behavior

TDD : test the implementation of the unitary parts of the system that form the behavior



However, BDD and TDD are complementary

BDD : specify and test system behavior

TDD : test the implementation of the unitary parts of the system that form the behavior

TDD + BDD allows to:

Avoiding the illusion of perfection

Some Tools for BDD

Quantum Automation

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Some Tools for BDD - Focus on Cucumber

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Formalization of behavior in BDD using Gherkin and Cucumber

Gherkin: A simplified natural language syntax used to describe the behavior of a system and its context.

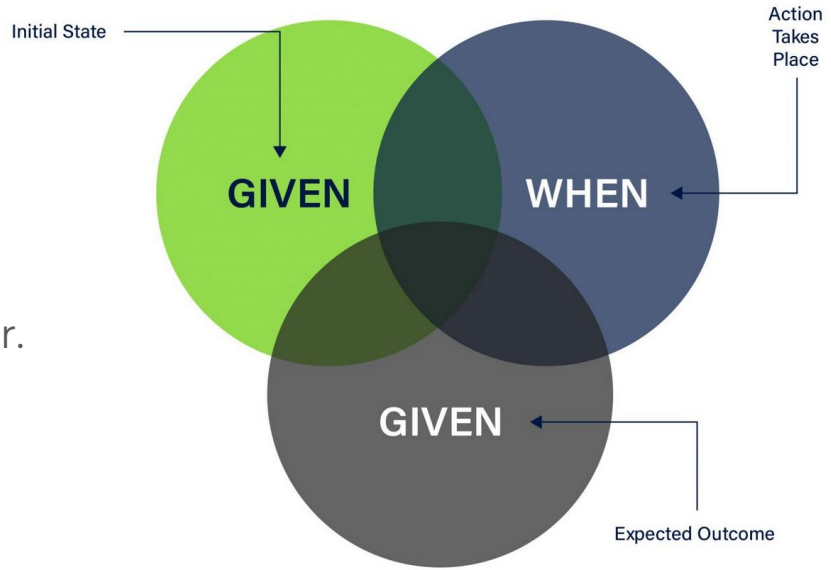
The implementation of a test with Cucumber is based on three textual elements for describing behavior:

Given: Describes an initial context.

When: Specifies events or actions that occur.

Then: Defines the expected outcome.

This formalization is written in **.feature files** in Cucumber



Example of behavioral test using Gherkin syntax and Cucumber

Gherkin: A simplified natural language syntax used to describe the behavior of a system and its context.

The implementation of a test with Cucumber is based on three textual elements for describing behavior:

Given: Describes an initial context.

When: Specifies events or actions that occur.

Then: Defines the expected outcome.

This formalization is written in **.feature files** in Cucumber

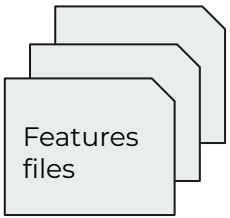
Le fichier PlayHangman.feature

```
Feature: Suggesting a letter

Scenario: Knowing the occurence of one letter in the word

  Given the word to be guessed is cucumber
  When the player suggests the letter u
  Then the letter is found 2 times

Scenario: Knowing the position of all occurences in the word

  Given the word to be guessed is cucumber
  When the player suggests the letter u
  Then the word is -u-u----
```

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Example of behavioral test using Gherkin syntax and Cucumber

1) Writing test scenario in Cucumber files

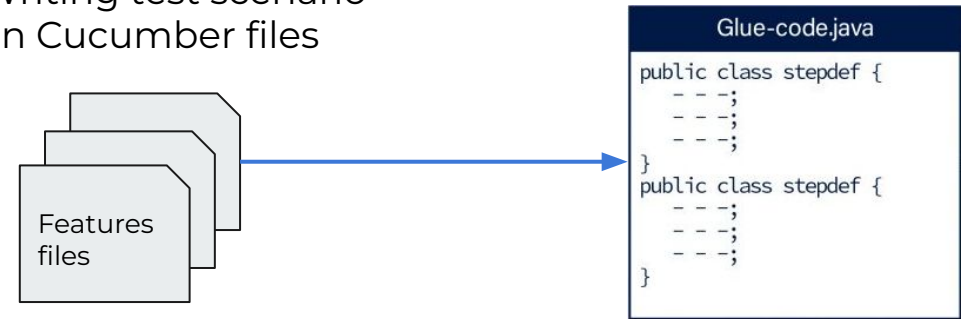



```
Feature: Make a coffee with a complete coffee machine
  A user want a coffee
  Scenario: A user plug the coffee machine and make a coffee Arabica
    Given a coffee machine with 0.10 l of min capacity, 3.0 l of max capacity,
    And a "mug" with a capacity of 0.25
    When I plug the machine to electricity
    And I add 1 liter of water in the water tank
    And I add 0.5 liter of "ARABICA" in the bean tank
    And I made a coffee "ARABICA"
    Then the coffee machine return a coffee mug not empty
    And a coffee volume equals to 0.25
    And a coffee "mug" containing a coffee type "ARABICA"
```

Example of behavioral test using Gherkin syntax and Cucumber

1) Writing test scenario in Cucumber files

2) Writing test steps for each scenario (Glue)



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Example of behavioral test using Gherkin syntax and Cucumber

1) Writing test scenario in Cucumber files

```
Feature: Make a coffee with a complete coffee machine
  A user want a coffee
  Scenario: A user plug the coffee machine and make a coff
    Given a coffee machine with 0.10 l of min capacity, 3.
    And a "mug" with a capacity of 0.25
    When I plug the machine to electricity
    And I add 1 liter of water in the water tank
    And I add 0.5 liter of "ARABICA" in the bean tank
    And I made a coffee "ARABICA"
    Then the coffee machine return a coffee mug not empty
    And a coffee volume equals to 0.25
    And a coffee "mug" containing a coffee type "ARABICA"
```

2) Writing test steps for each scenario (Glue)

```
2 usages  qperez
@Given("a coffee machine with {double} l of min capacity, {double} l of max capacity, {double} l pe
public void givenACoffeeMachine(double minimalWaterCapacity, double maximalWaterCapacity, double pu
    coffeeMachine = new CoffeeMachine(minimalWaterCapacity, maximalWaterCapacity, minimalWaterCapac
}

2 usages  qperez
@And("a {string} with a capacity of {double}")
public void aWithACapacityOf(String containerType, double containerCapacity) {
    if ("mug".equals(containerType))
        mug = new Mug(containerCapacity);
    if ("cup".equals(containerType))
        cup = new Cup(containerCapacity);
}

2 usages  qperez
@When("I plug the machine to electricity")
public void iPlugTheMachineToElectricity() { coffeeMachine.plugToElectricalPlug(); }

2 usages  qperez
@And("I add {double} liter of water in the water tank")
public void iAddLitersOfWater(double waterVolume) { coffeeMachine.addWaterInTank(waterVolume); }
```

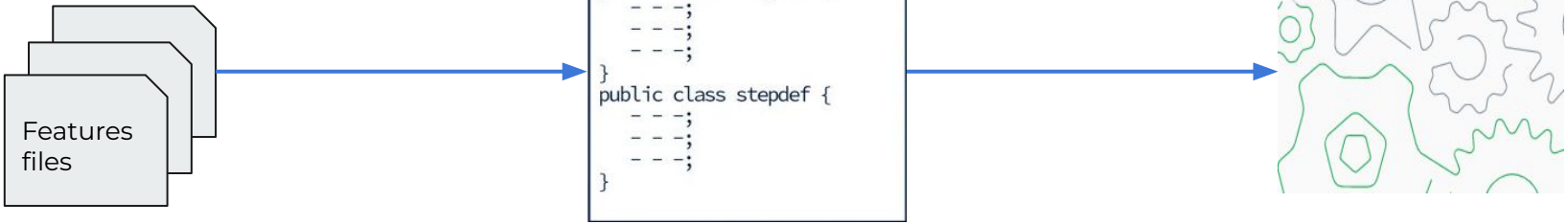
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Example of behavioral test using Gherkin syntax and Cucumber

1) Writing test scenario in Cucumber files

2) Writing test steps for each scenario (Glue)

3) Running Cucumber tests using Cucumber Test Runner




```
Feature: Make a coffee with a complete coffee machine
  A user want a coffee
  Scenario: A user plug the coffee machine and make a coffee Arabi
    Given a coffee machine with 0.10 l of min capacity, 3.0 l of m
    And a "mug" with a capacity of 0.25
    When I plug the machine to electricity
    And I add 1 liter of water in the water tank
    And I add 0.5 liter of "ARABICA" in the bean tank
    And I made a coffee "ARABICA"
    Then the coffee machine return a coffee mug not empty
    And a coffee volume equals to 0.25
    And a coffee "mug" containing a coffee type "ARABICA"
```

```
2 usages  qperez
@Given("a coffee machine with {double} l of min capacity, {double} l of max capacity, {double} l pe
public void givenACoffeeMachine(double minimalWaterCapacity, double maximalWaterCapacity, double pu
    coffeeMachine = new CoffeeMachine(minimalWaterCapacity, maximalWaterCapacity, minimalWaterCapac
}

2 usages  qperez
@And("a {string} with a capacity of {double}")
public void aWithACapacityOf(String containerType, double containerCapacity) {
    if ("mug".equals(containerType))
        mug = new Mug(containerCapacity);
    if ("cup".equals(containerType))
        cup = new Cup(containerCapacity);
}

2 usages  qperez
@When("I plug the machine to electricity")
public void iPlugTheMachineToElectricity() { coffeeMachine.plugToElectricalPlug(); }

2 usages  qperez
@And("I add {double} liter of water in the water tank")
public void iAddLitersOfWater(double waterVolume) { coffeeMachine.addWaterInTank(waterVolume); }
```

```
@RunWith(Cucumber.class)
@CucumberOptions(
        features = {"classpath:functional/features/"},
        glue = "fr.imt.coffee.machine.cucumber.steps"
)
//Permet d'ignorer les tests fonctionnels de Cucumber
//Ne lance pas la class CoffeeMachineFunctionalTest
public class CoffeeMachineCucumberFunctionalTest {

}
```

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Example of behavioral test using Gherkin syntax and Cucumber

```
Feature: Make a coffee with a complete coffee machine
  A user want a coffee
  Scenario: A user plug the coffee machine and make a coffee Arabica
    Given a coffee machine with 0.10 l of min capacity, 5.0 l of max capacity,
    And a "mug" with a capacity of 0.25
    When I plug the machine to electricity
    And I add 1 liter of water in the water tank
    And I add 0.5 liter of "ARABICA" in the bean tank
    And I made a coffee "ARABICA"
    Then the coffee machine return a coffee mug not empty
    And a coffee volume equals to 0.25
    And a coffee "mug" containing a coffee type "ARABICA"
```

Each line in the .feature file is a step definition

Each type of element is mapped in the step definition

```
2 usages  qperez
@Given("a coffee machine with {double} l of min capacity, {double} l of max capacity, {double} l pe
public void givenACoffeeMachine(double minimalWaterCapacity, double maximalWaterCapacity, double pu
    coffeeMachine = new CoffeeMachine(minimalWaterCapacity, maximalWaterCapacity, minimalWaterCapac
}

2 usages  qperez
@And("a {string} with a capacity of {double}")
public void aWithACapacityOf(String containerType, double containerCapacity) {
    if ("mug".equals(containerType))
        mug = new Mug(containerCapacity);
    if ("cup".equals(containerType))
        cup = new Cup(containerCapacity);
}

2 usages  qperez
@When("I plug the machine to electricity")
public void iPlugTheMachineToElectricity() { coffeeMachine.plugToElectricalPlug(); }

2 usages  qperez
@And("I add {double} liter of water in the water tank")
public void iAddLitersOfWater(double waterVolume) { coffeeMachine.addWaterInTank(waterVolume); }
```

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Acceptance Test Driven Development

ATDD => An of testing based on client requirements. High level driving the application development. Like BDD, it aims to achieve expressiveness of the requirement through the description of:

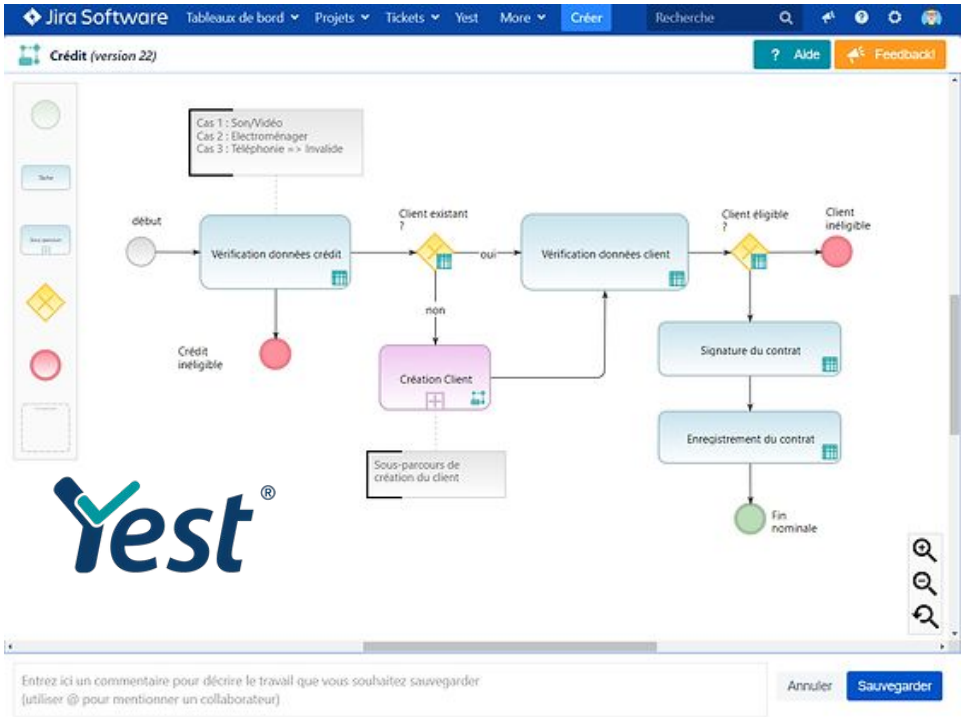
- Usage contexts
- Actions to be performed on the system
- Acceptance criteria for the system

It seeks collaboration, understanding, and validation of the system by various stakeholders (Clients, Product Owners, developers, quality analysis teams, etc.).

Simone Scardapane

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

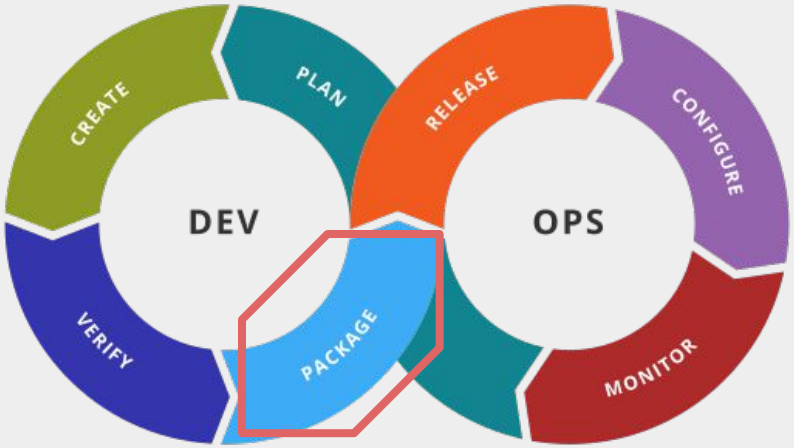
Acceptance Test Driven Development—Technologies



gauge

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Packaging

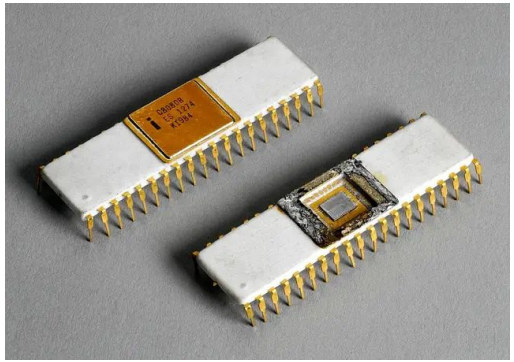


INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

A brief history of software packaging

From 70's to the end of 80's : essentially a binary writing in assembly for a specific kind of processor / platform ⇒ **hard to deliver and platform specific**

```
0000 0e 09              mvi c, 09h
0002 11 09 01           lxi d, 0109h
0005 cd 05 00           call 0005h
0008 c9                 ret
0009 47                 mov b, a
000a 72                 mov m, d
000b 65                 mov h, l
000c 65                 mov h, l
000d 74                 mov m, h
000e 69                 mov l, c
000f 6e                 mov l, m
0010 67                 mov h, a
0011 73                 mov m, e
0012 20                 nop
```



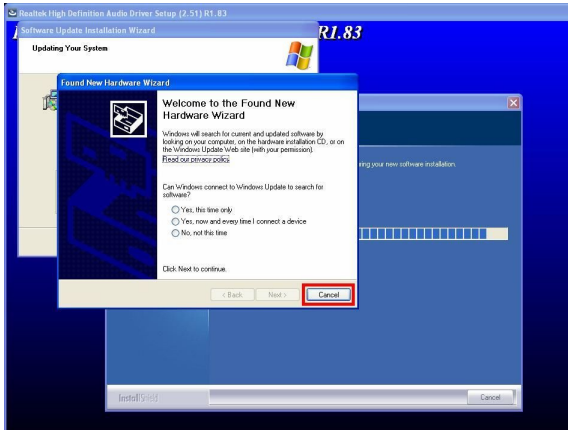
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

A brief history of software packaging

In 90's: emergence of:

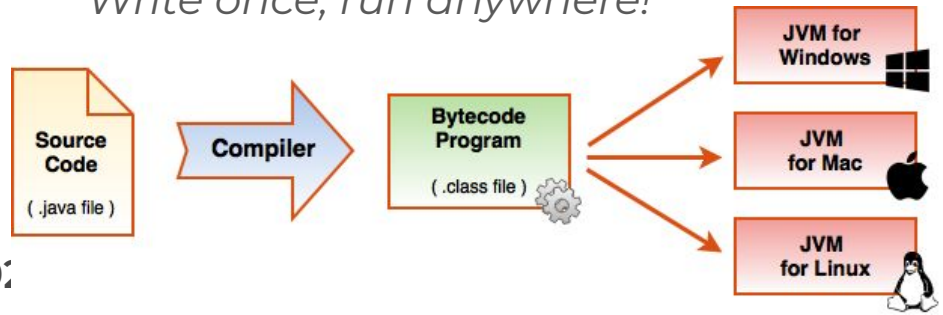
- installers for software programs (InstallShield, etc.)
- evolution of compilers for binaries creation
- invention of the JVM (1994)

⇒ **packaging less specific platform but still often machine dependent**
⇒ .exe with libraries (DLL), InstallShield Wizard, Bytecode (.class and .jar), etc



JAVA

"Write once, run anywhere!"

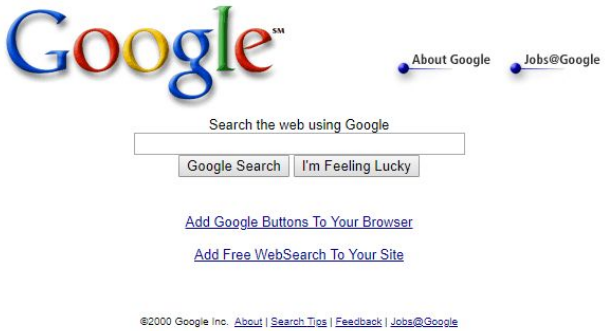


INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

A brief history of software packaging

At the end of 90s and beginning of 2000s:

- emerging of internet and web technologies
- a service could be share to multiple users with only one machine
- abstraction from the machine's execution context, only a browser + an Internet connection are required



⇒ **“packaging” are mainly using web files via FTP on Apache web servers** (.html, .css, .js) + **Databases**

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

A brief history of software packaging

In the 2010's:

- invention of containerization technics (LinuX Containers, Docker, etc.)

⇒ "packaging" is an **all-in-one container** which are **runnable "anywhere"**

A brief history of software packaging

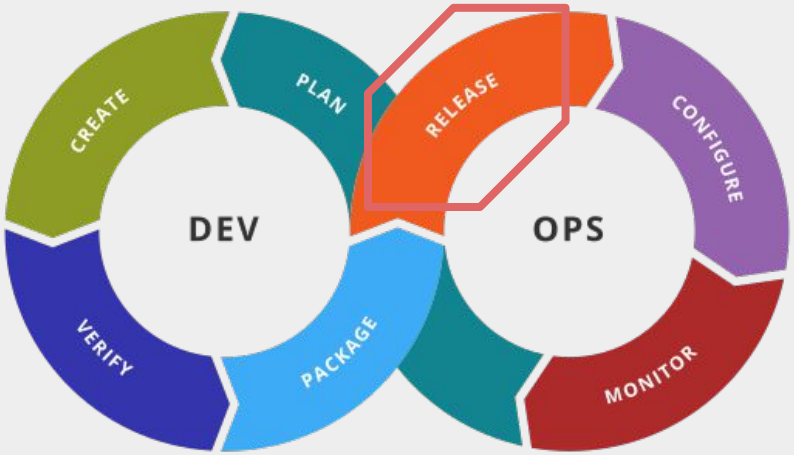
Now multiple ways can be used:

- Containerization technics: LinuX Containers, Docker, etc.
- “Classical packaging”: JAR, WAR, .exe, package (for package managers)
-

⇒ “packaging” is an **all-in-one container** which are **runnable “anywhere”**

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Release



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

A brief history of software releasing

From 60's to end of 90's : software were principally released on removable devices like :

- punched cards
- Magnetic bands
- Floppy disks
- CD-ROMs
- DVD-ROMs

One to one principle ⇒ **one delivery for one user/service**

Required specific hardware systems/supply chains for the removable devices production and distribution ⇒ **complex and costly**

The version upgrade is challenging with removable devices!

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

A brief history of software releasing

At the end of 90s internet has been a game changer!

- Services and software initially released for PCs through removable devices are now accessible via websites
- From “One to one logic” to “one to many” ⇒ one machine to serve many users!

Releasing on one or more servers to give access to the software but… Warning, they are users behind!

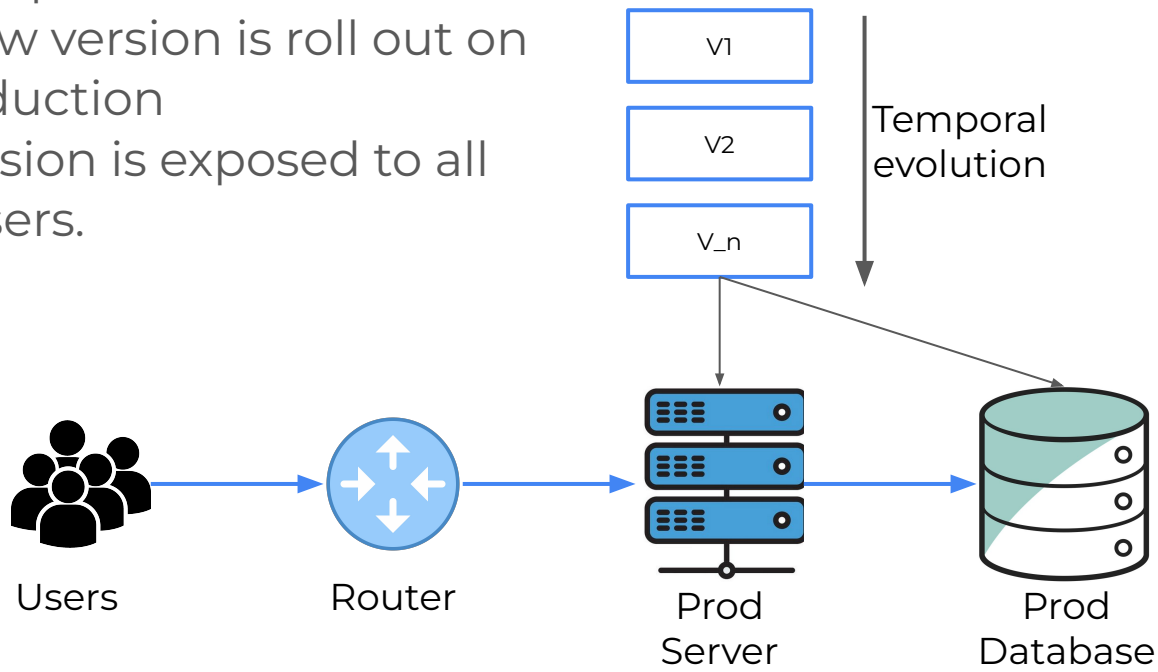
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

But in practice how new releasing/deployment work?

En espérant que l'appli releasée ne finisse pas comme le titanic...

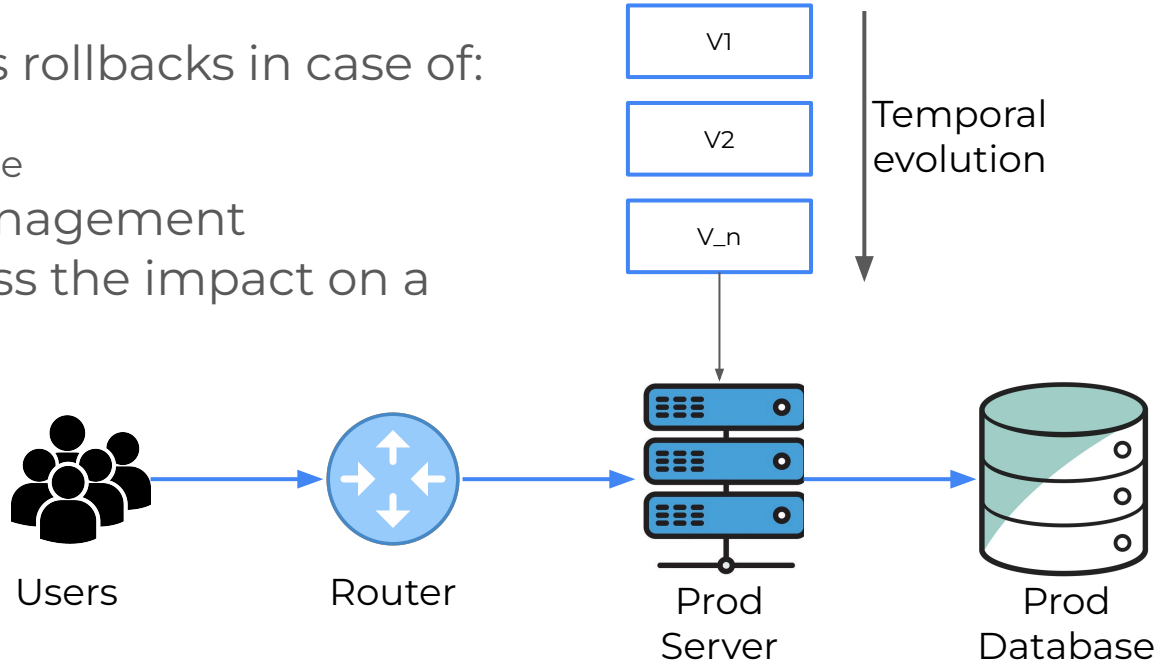
At the beginning, classical deployment (“grandpa deployment”)

- “Unique” production server/DB
- Each new version is roll out on the production
- New version is exposed to all same users.



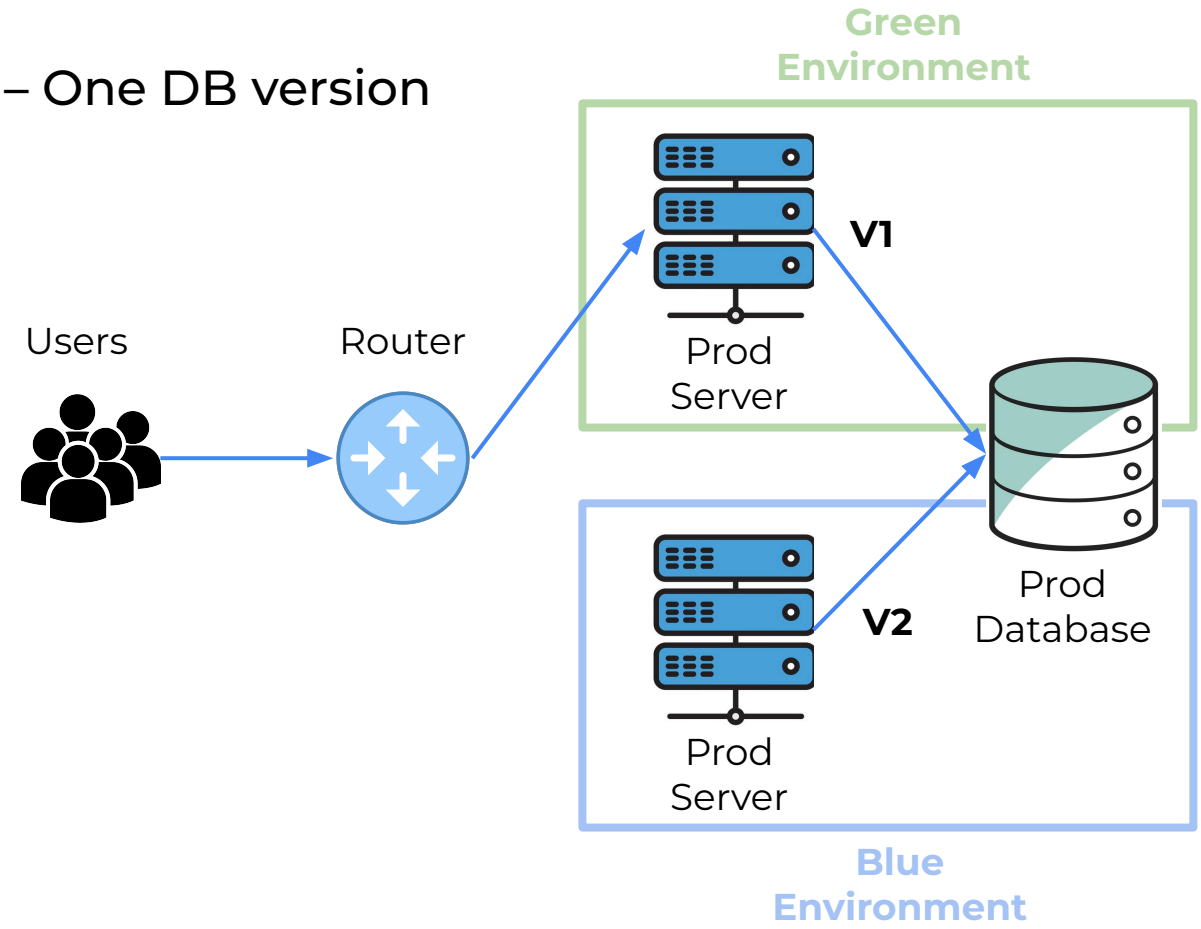
At the beginning, classical deployment (“grandpa deployment”)

- Not easy to performs rollbacks in case of:
  - Bugs
  - Non-users acceptance
- Complicated DB Management
- Impossibility to assess the impact on a subset of users.



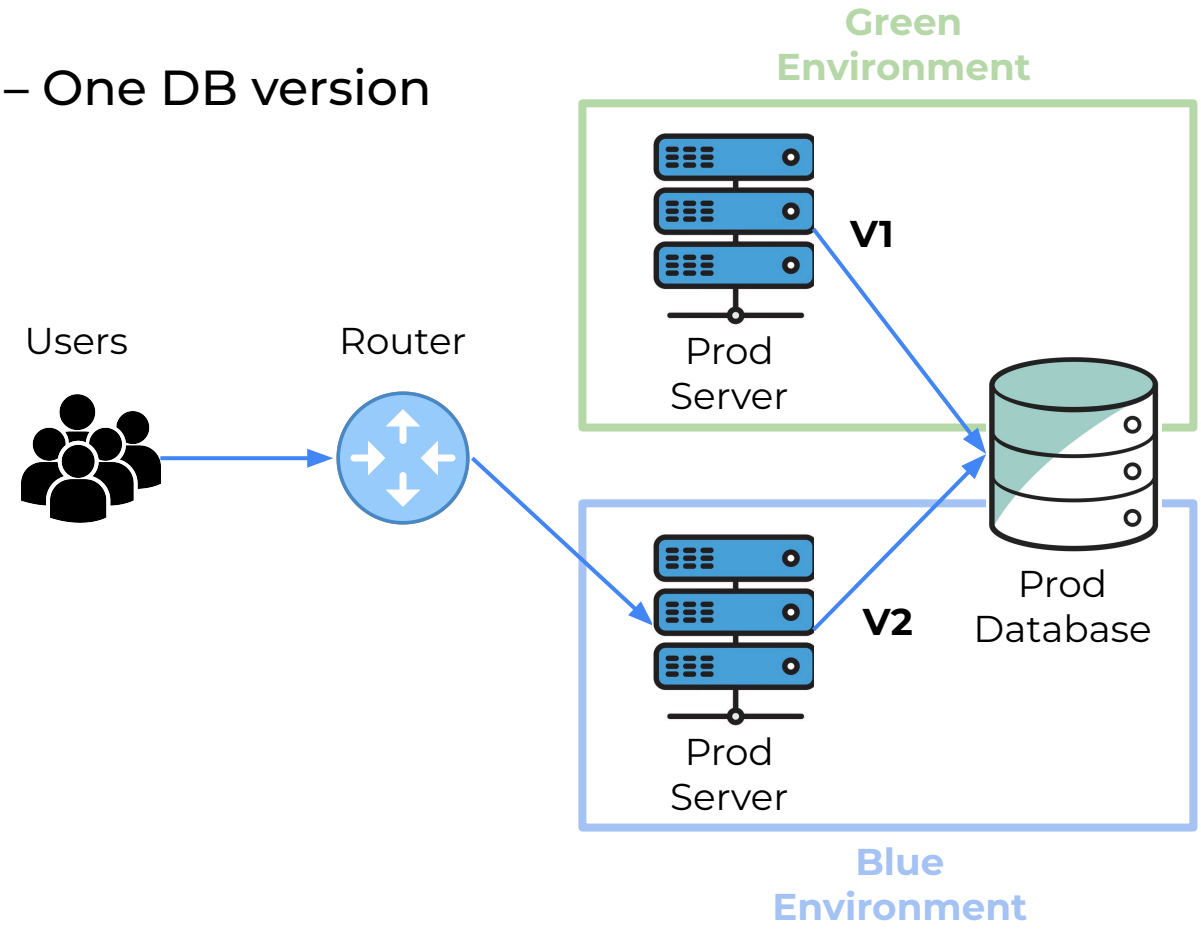
Blue-Green deployment – One DB version

- Each version is deployed alternately on a blue or green environment
- Router configuration switching to redirect users to the new/rollback version
- Easy to rollback
- 2 Versions:
  - Same database for Blue and Green env
  - One for each
- Roll-out on infrastructure copies



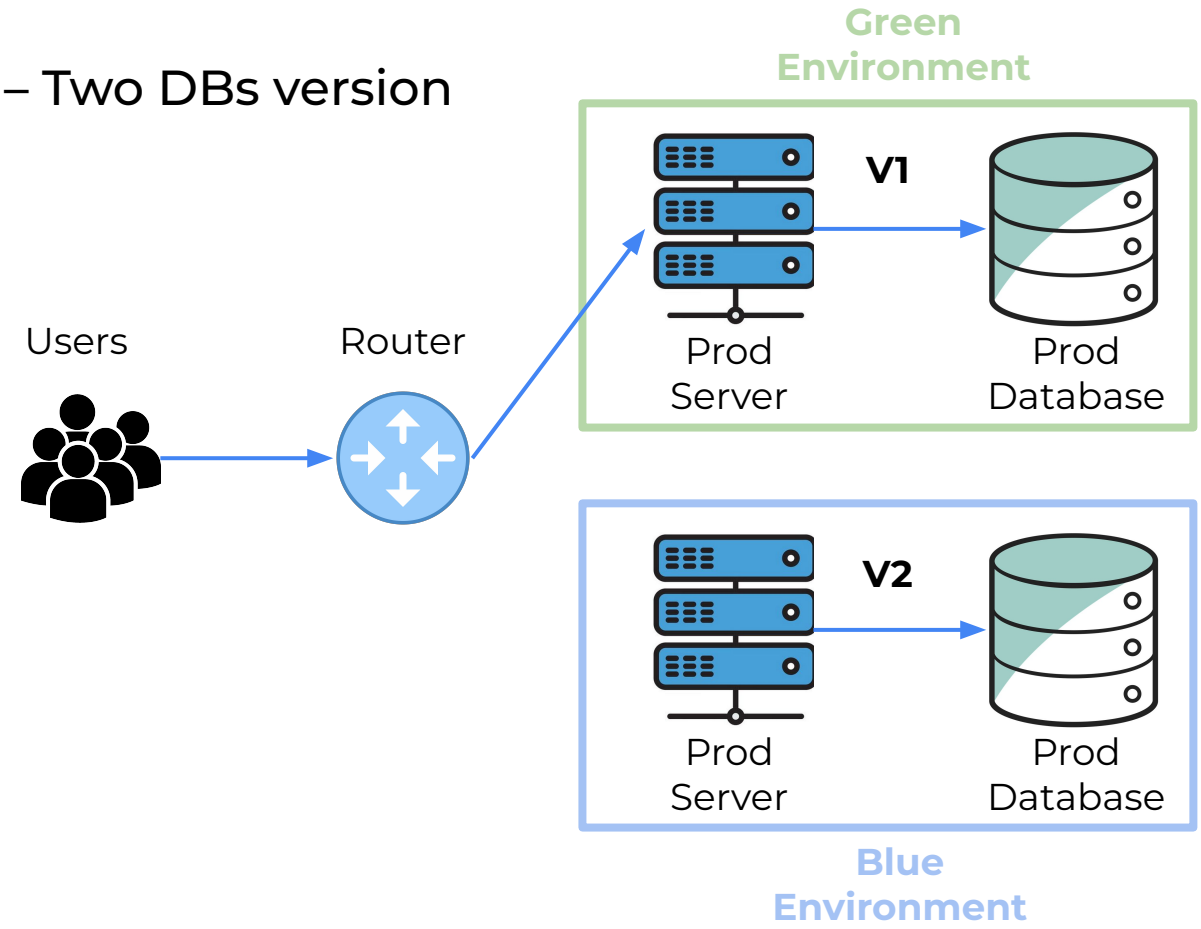
Blue-Green deployment – One DB version

- Each version is deployed alternately on a blue or green environment
- Router configuration switching to redirect users to the new/rollback version
- Easy to rollback
- 2 Versions:
  - Same database for Blue and Green env
  - One for each
- Roll-out on infrastructure copies



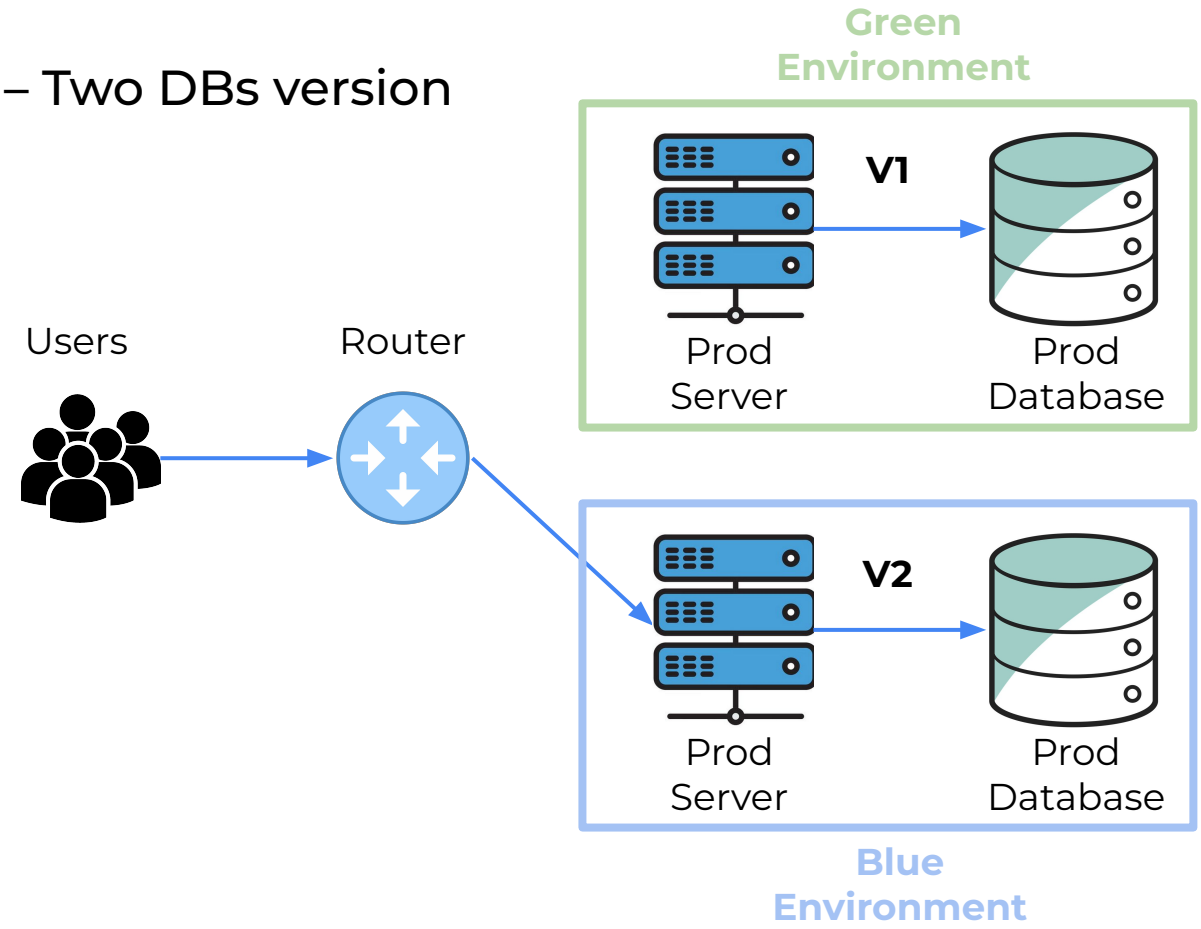
Blue-Green deployment – Two DBs version

- Each version is deployed alternately on a blue or green environment
- Router configuration switching to redirect users to the new/rollback version
- Easy to rollback
- 2 Versions:
  - Same database for Blue and Green env
  - One for each
- Roll-out on infrastructure copies



Blue-Green deployment – Two DBs version

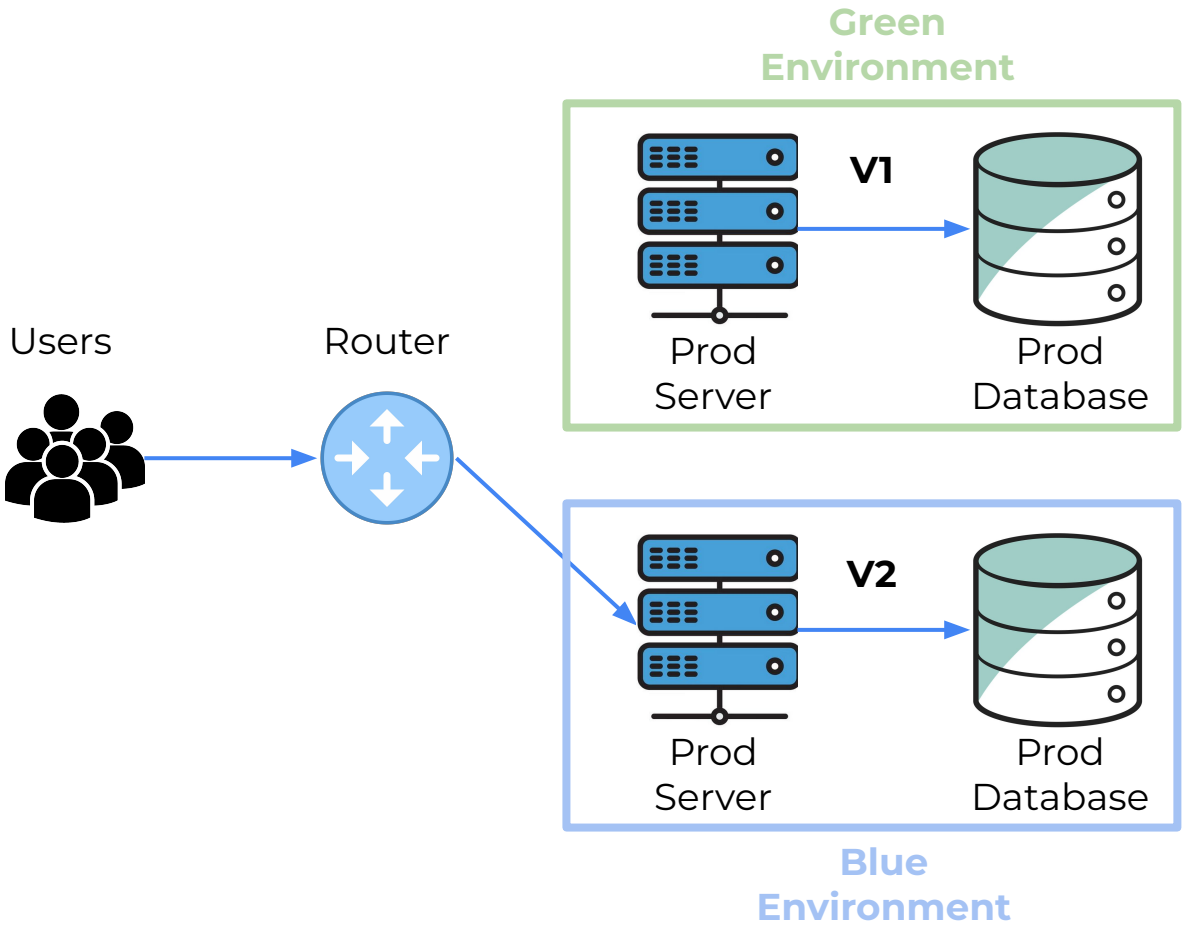
- Each version is deployed alternately on a blue or green environment
- Router configuration switching to redirect users to the new/rollback version
- Easy to rollback
- 2 Versions:
  - Same database for Blue and Green env
  - One for each
- Roll-out on infrastructure copies



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

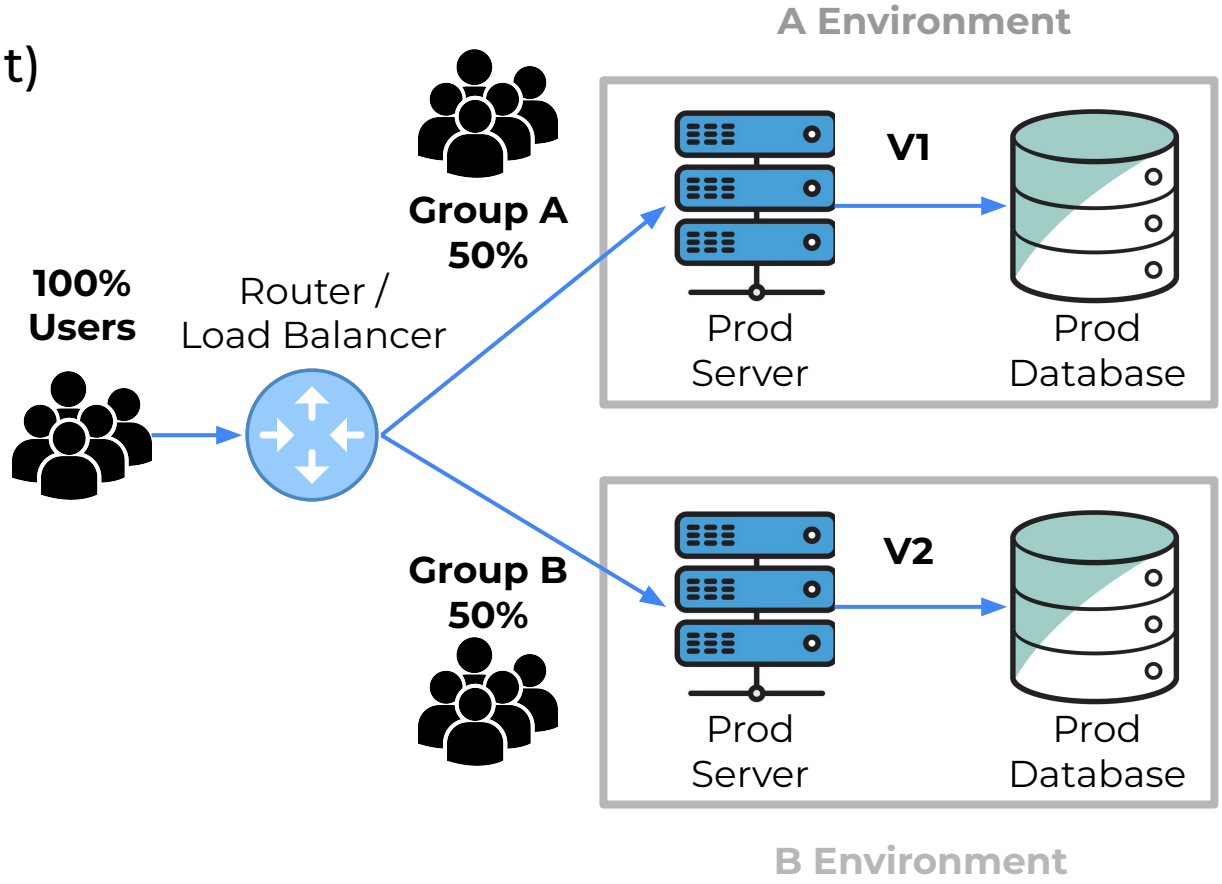
Blue-Green deployment

- All users see the same version
- Impossibility to assess the impact on a subset of users.



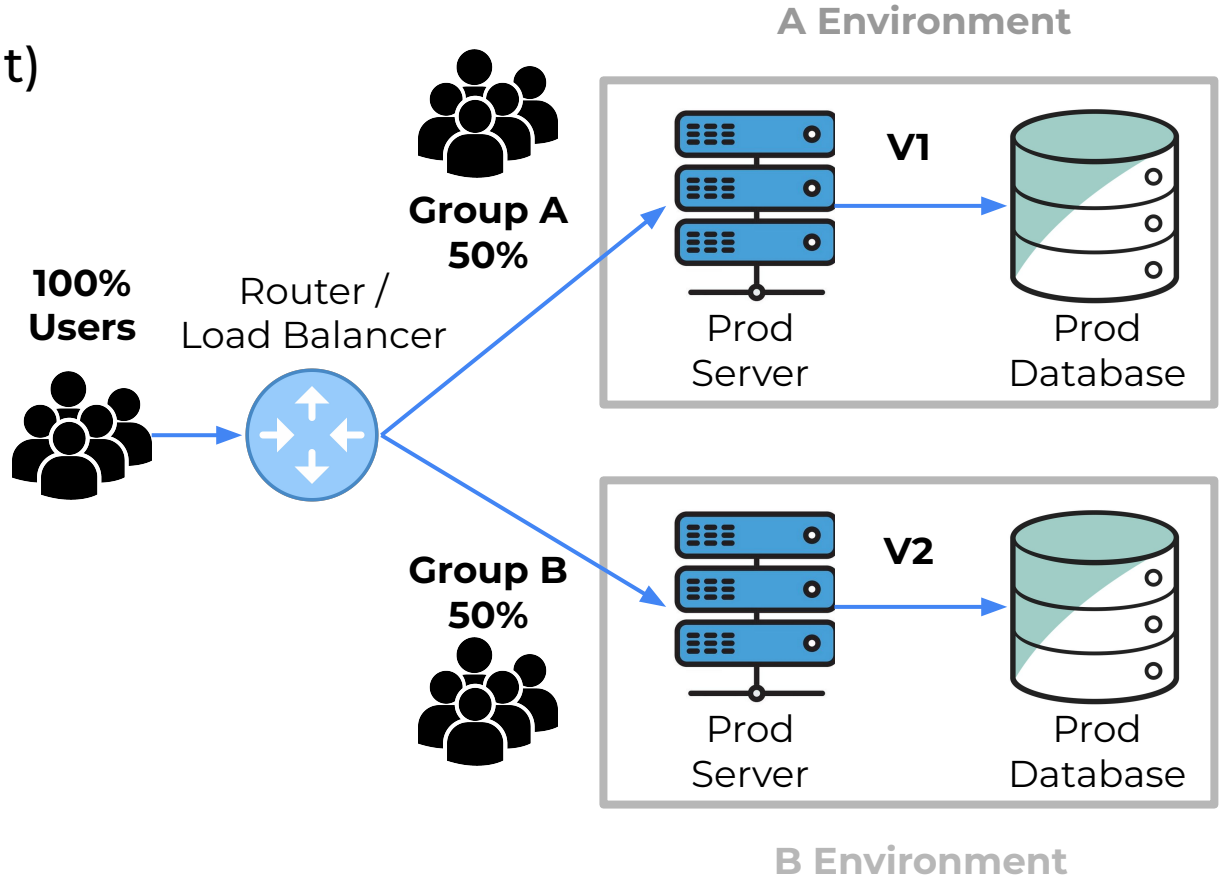
A/B testing (deployment)

- Specific router/load balancer configuration to redirect **50% of users on V1** and **50% on V2**
- Deployment: on a copy of the infrastructure or a different infrastructure.
- Frequently used for evaluating or deploying a new version (can also be used for feature evaluation).
- Allows for assessing the new version while minimizing the impact on all users.



A/B testing (deployment)

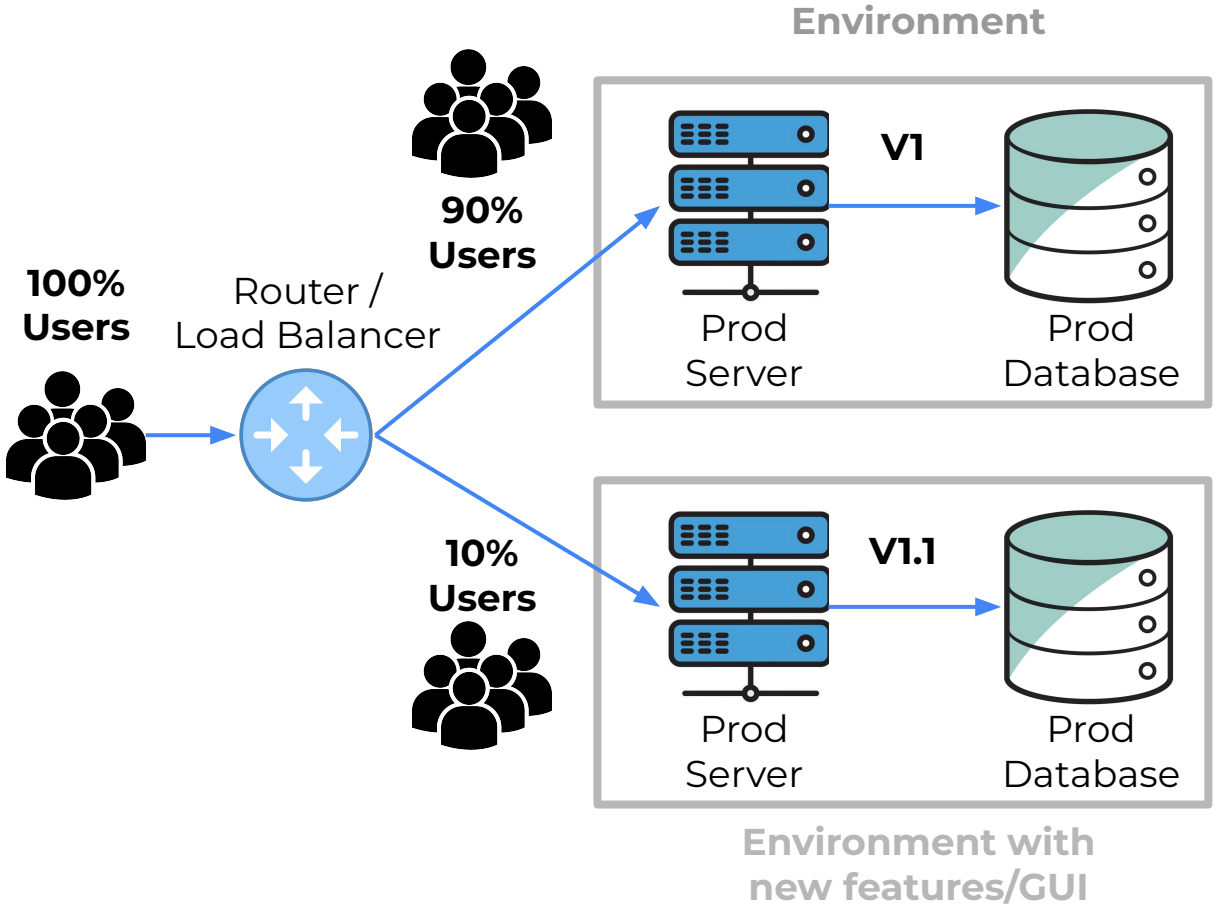
- Controlled experiments “new treatment” ⇒ 50%, “control group” ⇒ 50%
- Allows to collect various metrics on each version and comparing them
- **Impacts can be evaluated using statistical tests (e.g. Fischer Test)**



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

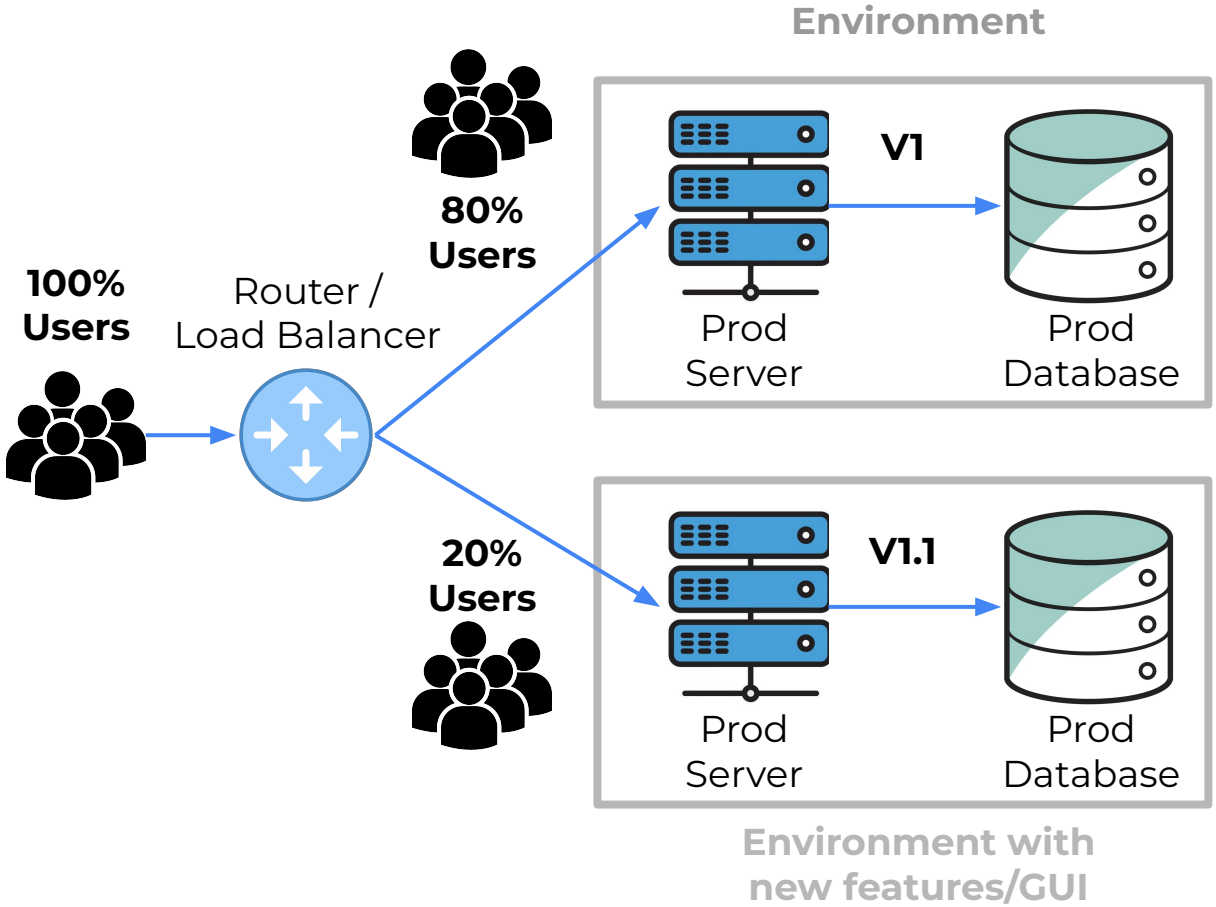
Canary testing

- Specific router/load balancer configuration to redirect **90% of users on V1** and **10% on V1.1 (variable proportions)**
- Deployment: on a copy of the infrastructure or a different infrastructure.
- Reduces the risk of introducing a new software version in production by slowly rolling out the change to a small subset of users
- Incremental roll out



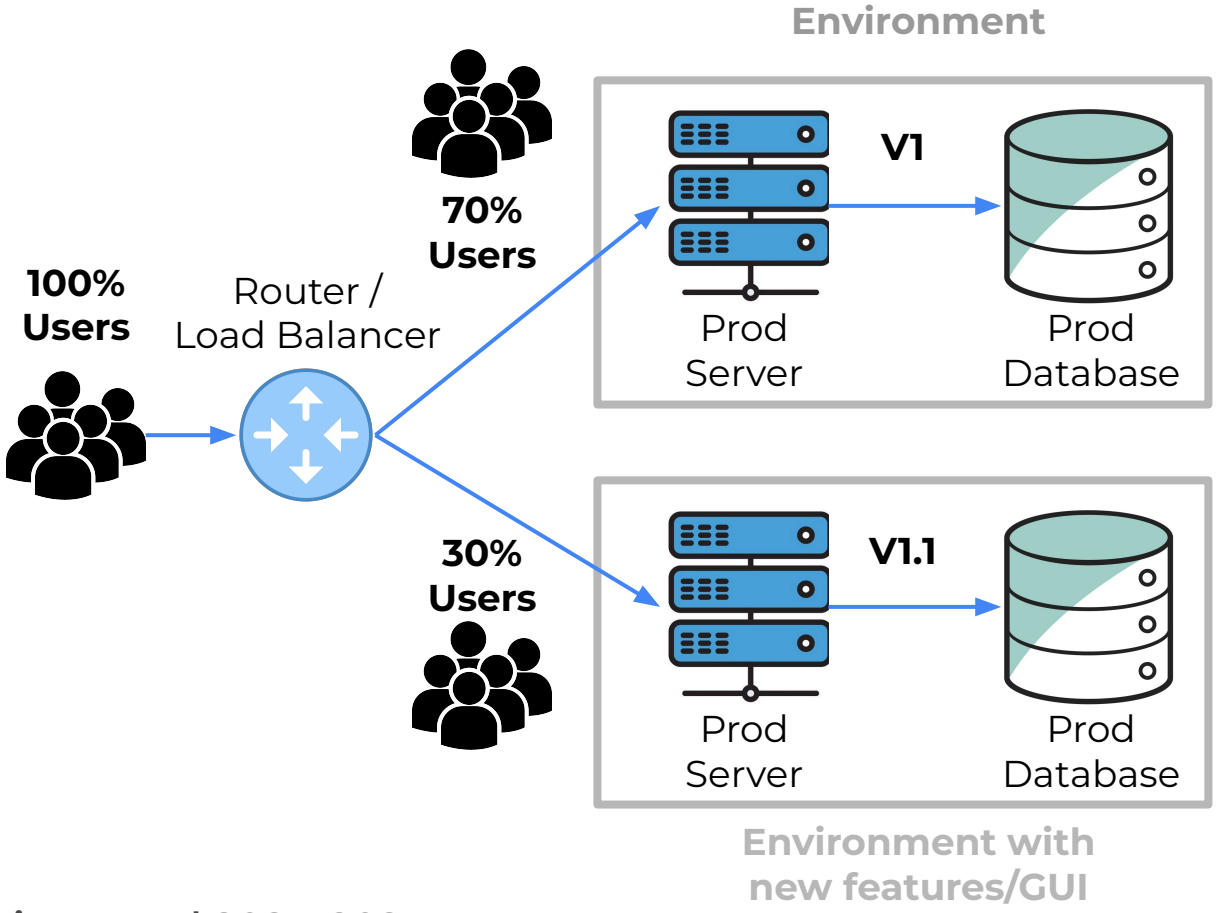
Canary testing

- Specific router/load balancer configuration to redirect **80% of users on V1** and **20% on V1.1 (variable proportions)**
- Deployment: on a copy of the infrastructure or a different infrastructure.
- Reduces the risk of introducing a new software version in production by slowly rolling out the change to a small subset of users
- Incremental roll out



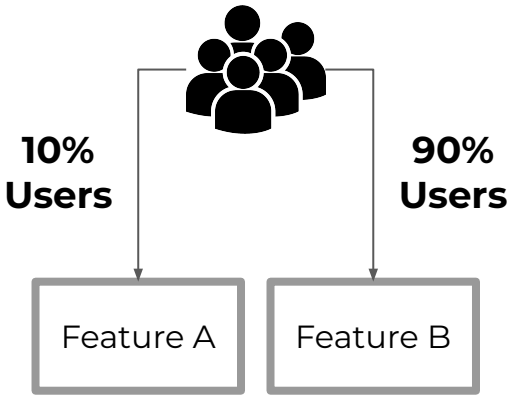
Canary testing

- Specific router/load balancer configuration to redirect **70% of users on V1** and **30% on V1.1 (variable proportions)**
- Deployment: on a copy of the infrastructure or a different infrastructure.
- Reduces the risk of introducing a new software version in production by slowly rolling out the change to a small subset of users
- Incremental roll out

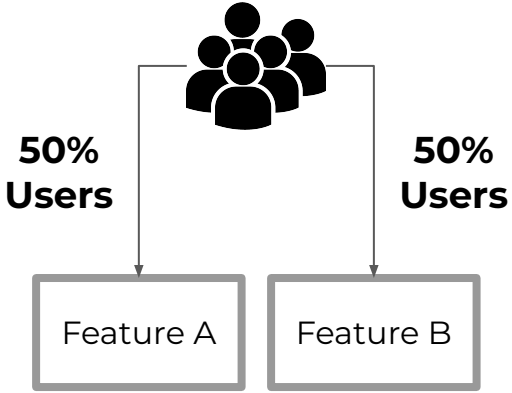


INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

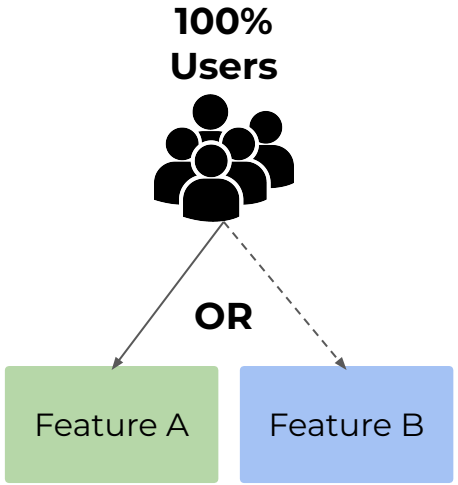
Canary testing VS AB/testing VS Blue-Green deployment



Canary Testing



A/B Testing



Blue/Green Testing

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Canary testing VS AB/testing VS Blue-Green deployment

INSA | INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

How can you automate compilation, testing, packaging and releasing?

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

By using solutions of automation

Build Managers

Continuous Integration / Delivery

GitHub Actions

Analyzers/Coverage

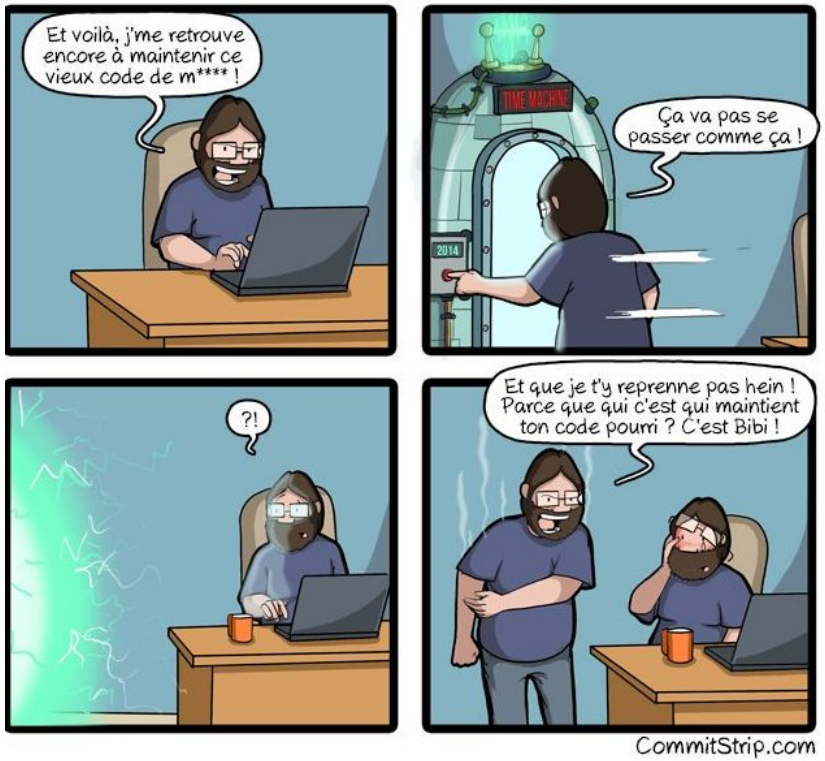
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Automatic Code Analyzers

- **Checkstyle:**
  - https://checkstyle.sourceforge.io/checks/metrics/index.html
  - Code convention
  - Maven, IntelliJ integration
- **SpotBugs**
  - https://spotbugs.github.io/
  - Static code analysis
  - Maven integration
- **PMD**
  - https://docs.pmd-code.org
  - Cross-language static code analyzer
  - Maven integration with plugin
  - Allows to create rules for Maven

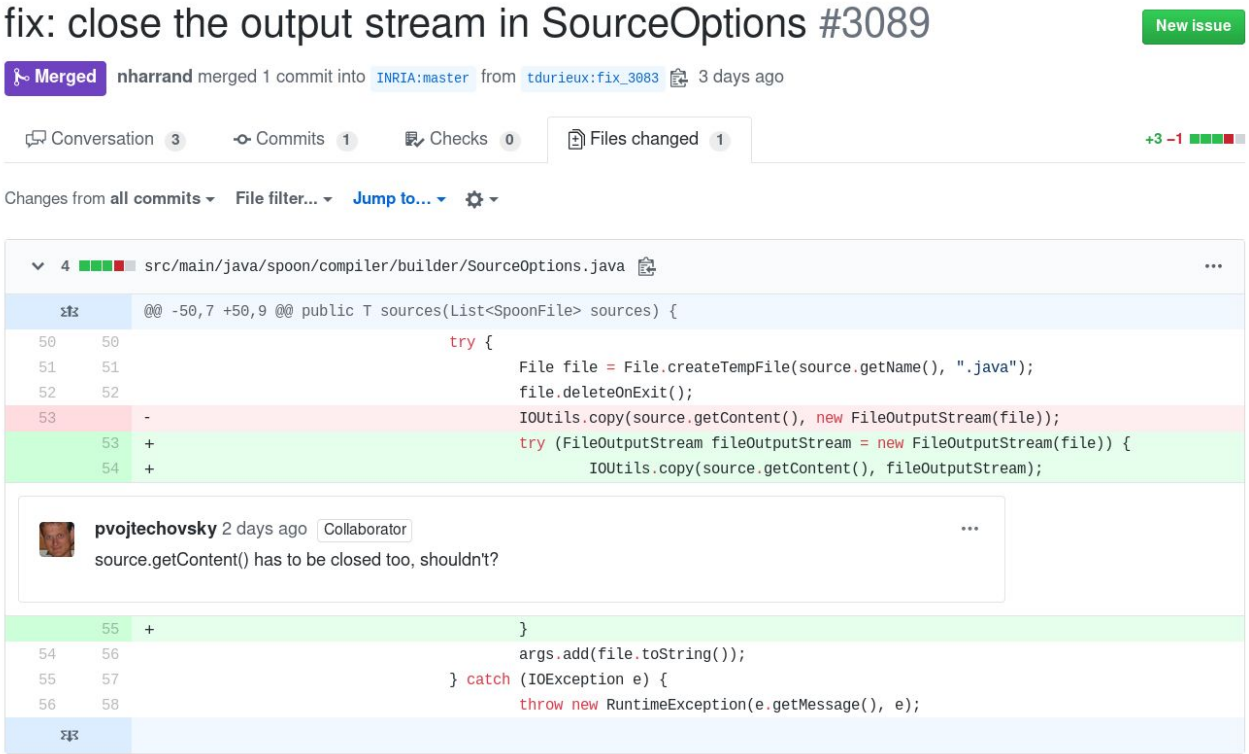
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Manual Code Analysis - Code Inspection



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Manual Code Analysis - Code Inspection



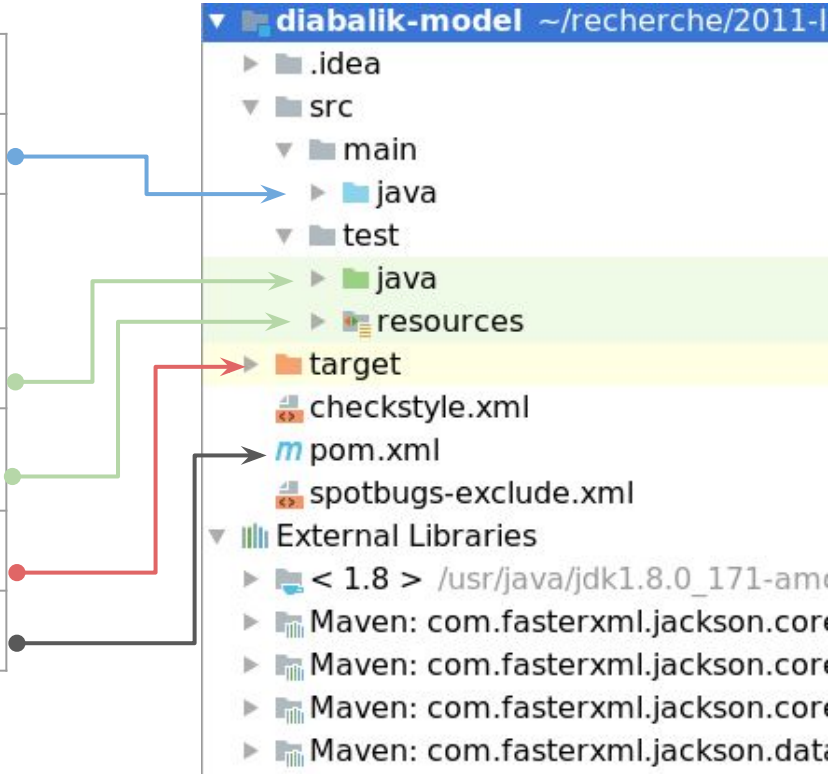
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Build Managers

- **Manage dependencies**
- **Manage “supply chain”**
  - Compilation (JDK, compiler, etc.)
  - Quality Analysis (static analyzer, coverage, etc.)
  - Test (JUnit, Mockito, Cucumber)
  - Packaging (JAR, WAR, Docker Image, etc.)
  - Execution (Shell, Docker, etc.)
  - Deployment (Docker registry, Apache Server, FTP server, etc.)
  - Versionning (plugins Maven/Git)
  - …
- Can be used to **divide an application into modules**

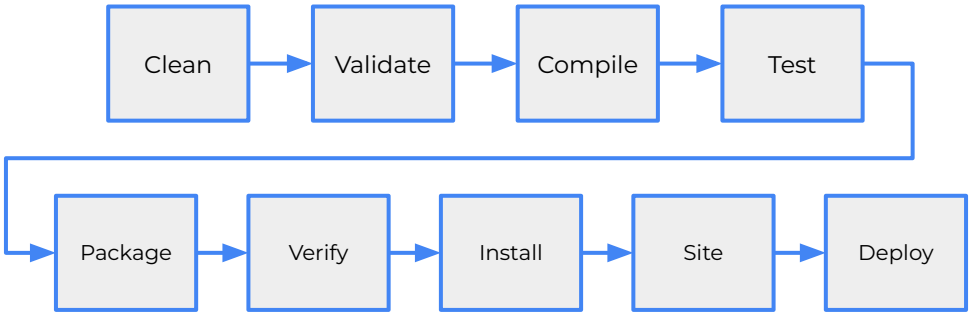
Build Managers - Maven in practices - Structure

Element	Description
`/src/main/java`	Sources and java packages
`/src/main/resources`	Resources for the project (configuration files, images, web pages, etc.)
`/src/main/test`	Test sources and test packages
`/src/main/resources`	Resources needed for the test (files, data, etc.)
`target`	The output Maven folder
`pom.xml`	The Maven script of the project



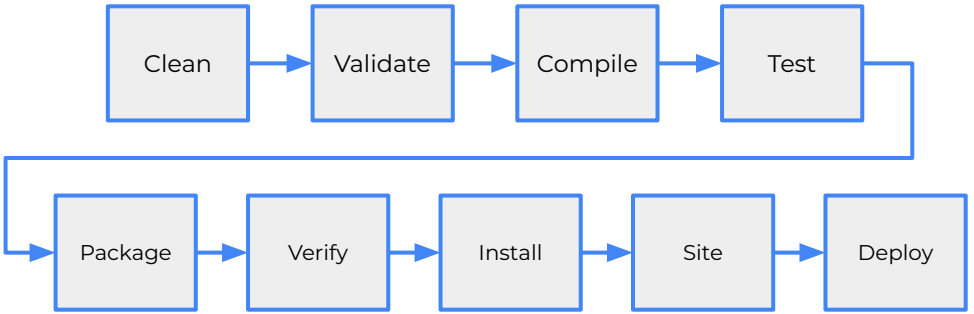
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Build Managers - Maven in practices - Lifecycle



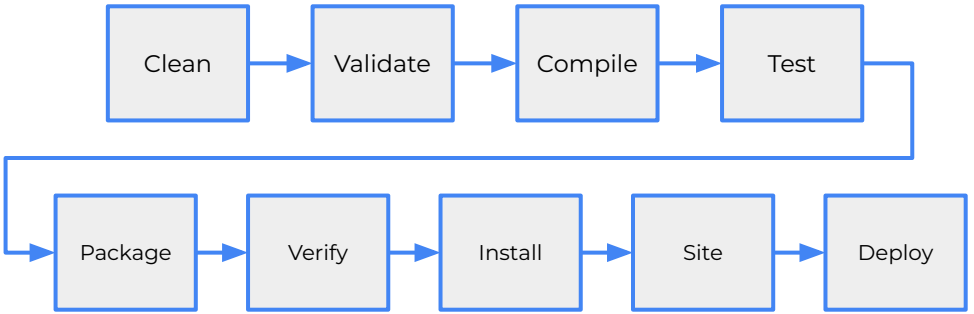
Build Managers - Maven in practices - Lifecycle

- **Clean**: cleans up the project by deleting elements from the previous build (target)
- **Validate**: checks that the project configuration is correct (POM, dependencies, etc.)
- **Compile**: compiles project sources
- **Test**: runs tests
- **Package**: packages project elements according to the defined configuration



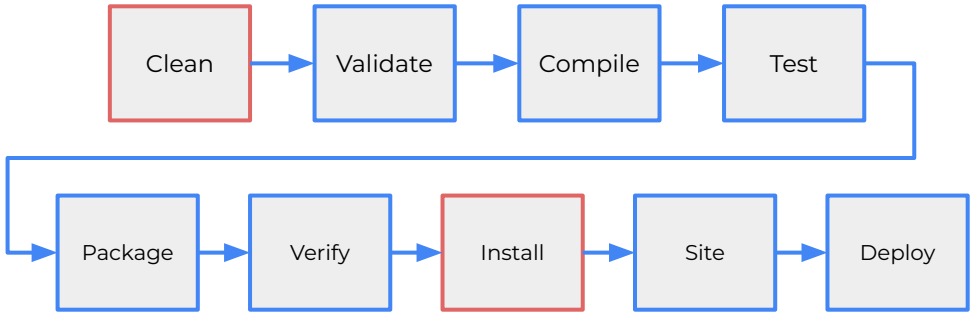
Build Managers - Maven in practices - Lifecycle

- **Verify**: executes compiled code with integration tests
- **Install**: installs the package in the local repository / (run application)
- **Site**: generates the site and the documentation for the project
- **Deploy**: deploy package on the target



Build Managers - Maven in practices - Lifecycle

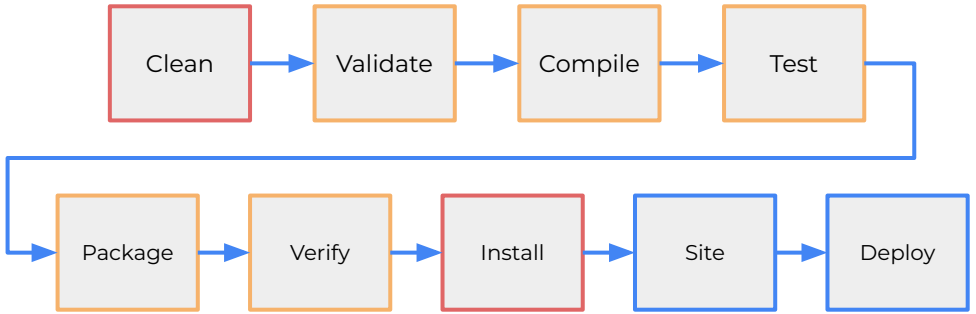
- **Exemple :**
  - mvn clean install



Build Managers - Maven in practices - Lifecycle

- **Exemple :**
  - mvn clean install

Executes all phases between “Clean” and “Install” (including them)



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Build Managers - Maven in practices - Repositories

Maven has two kinds of repository:

- **local:** contains dependencies retrieved from remote repositories (local copies)
  - **stored by default in $HOME/.m2**
- **remote:** mvnrepository / nexus (private or public)

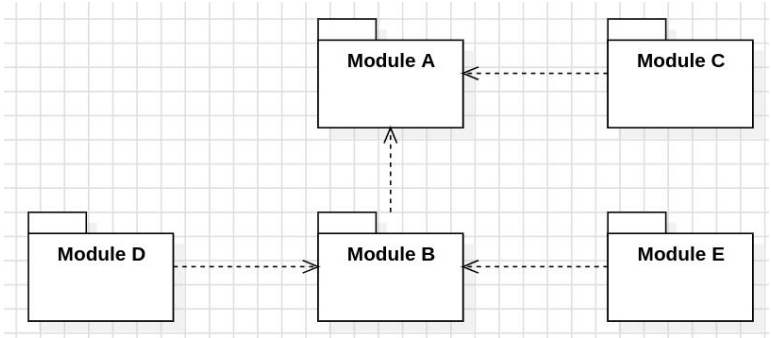
⚠️Sometimes (often in not open-source context) dependencies are not retrieved from the public MVNRepository but from a private Nexus (internal to the compagny) ⇒ need a specific configuration in .m2/settings.xml

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Build Managers - Maven in practices - Modularization

Maven allows modularization using “subproject modules”

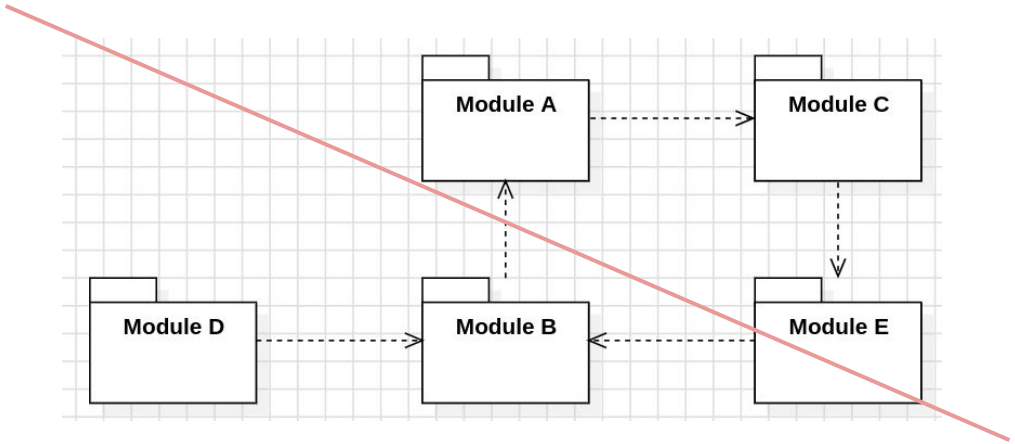
- “composability of projects” using different maven modules
- “Inheritance” of pom.xml configurations



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Build Managers - Maven in practices - Modularization

⚠️ Maven forbids cycle dependency. It’s a sign of bad project architecture.



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

CI/CD - automation server

Not seriously: a CI/CD server is a kind of software “qui fait pouet pouet, qui clignote et qui peut être tout rouge !”

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

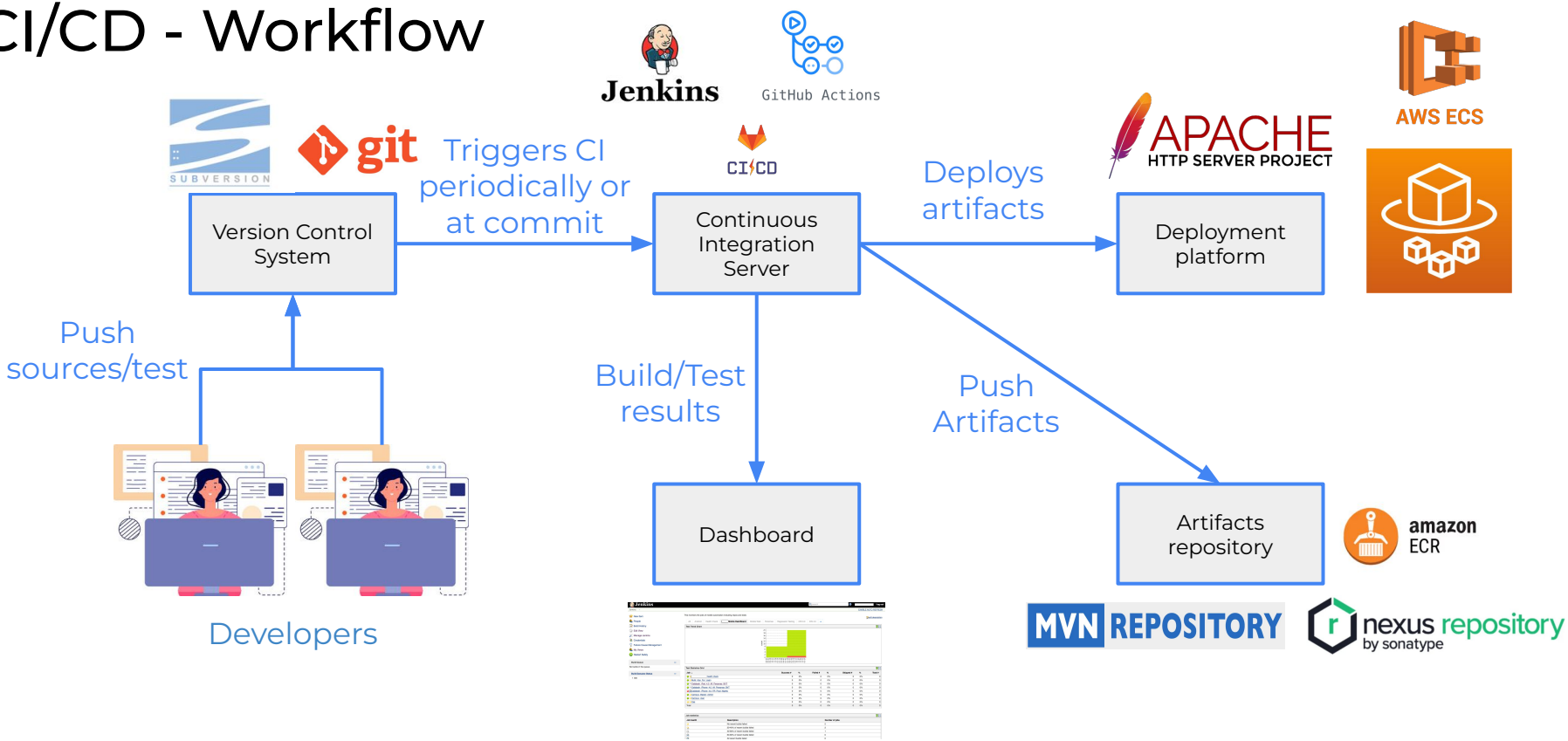
CI/CD - automation server

More seriously, a CI/CD allows to:

- Standardize the building environment (OS, JDK, Environment variables, etc.)
- Automate the build using the build manager (Maven, Gradle, Cargo, NPM, etc.)
  - Compilation
  - Test
  - Packaging
  - Etc.
- Collect information on quality (metrics)
- Produce build reports
- Manage the delivery
- Make coffee, create unicorns, etc....

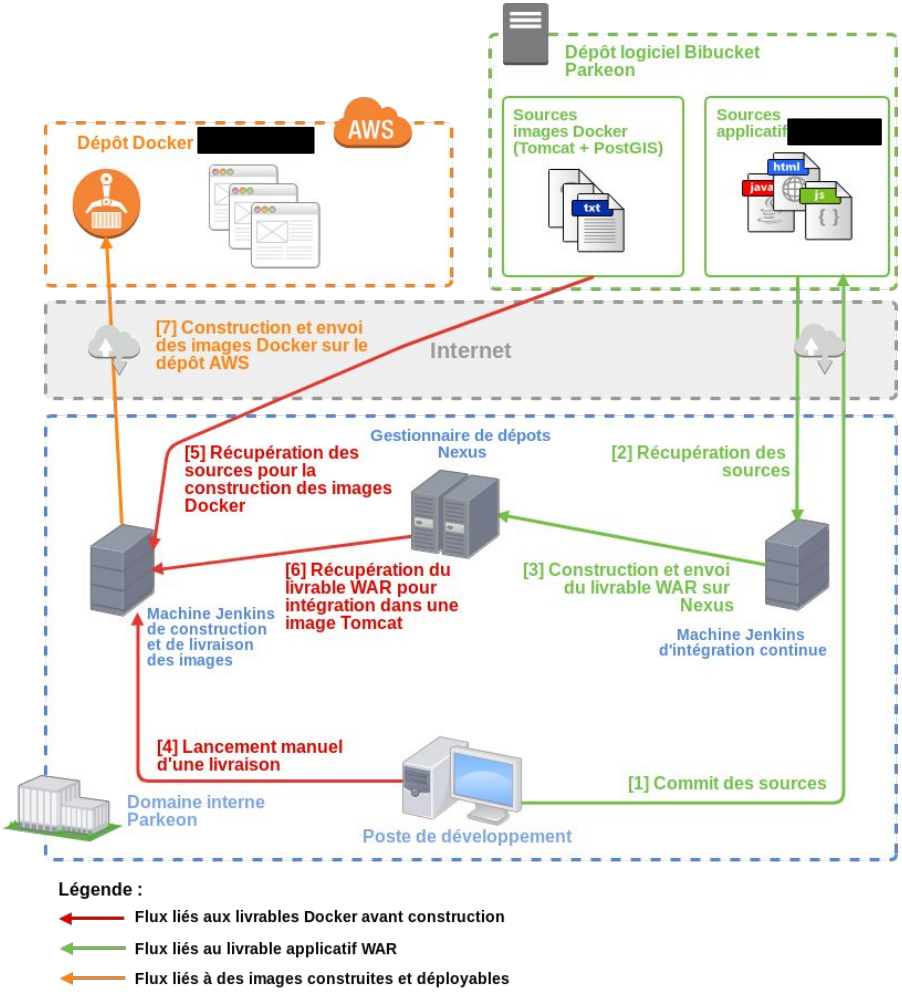
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

CI/CD - Workflow



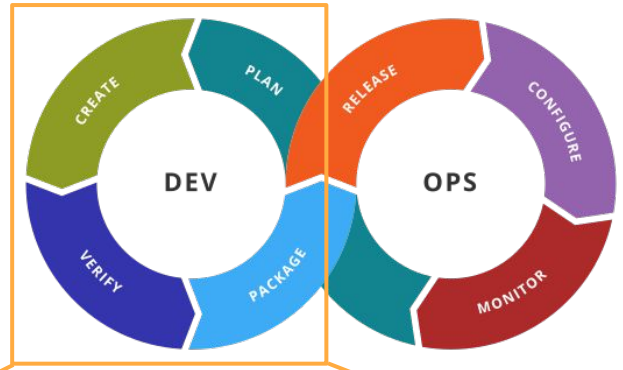
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

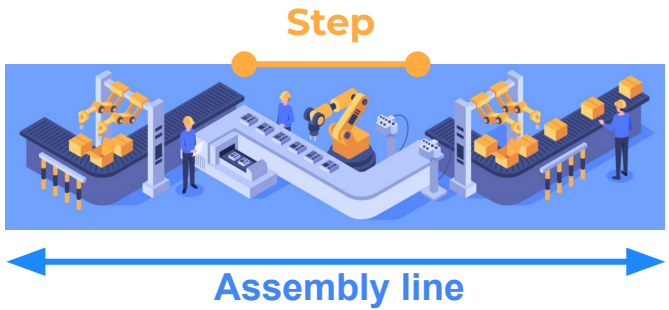
CI/CD - Workflow - Example of real workflow in an IT company



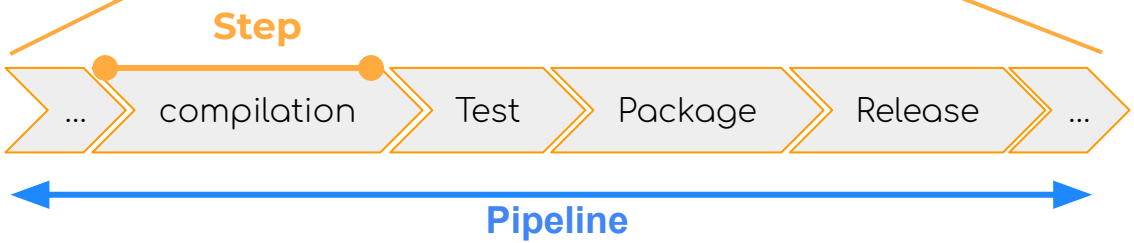
What is a pipeline in DevOps and how to automate?

- **Pipeline** can be seen as an **assembly line** in a factory.
- A **pipeline** is a set of **steps** producing an executable/deployable software (a product) from source code (raw materials).
- **Pipelines** are executed on Continuous Integration and Continuous Delivery Platforms (CI/CD).
- **Pipelines** are file descriptors of **steps** and **tools** to used.



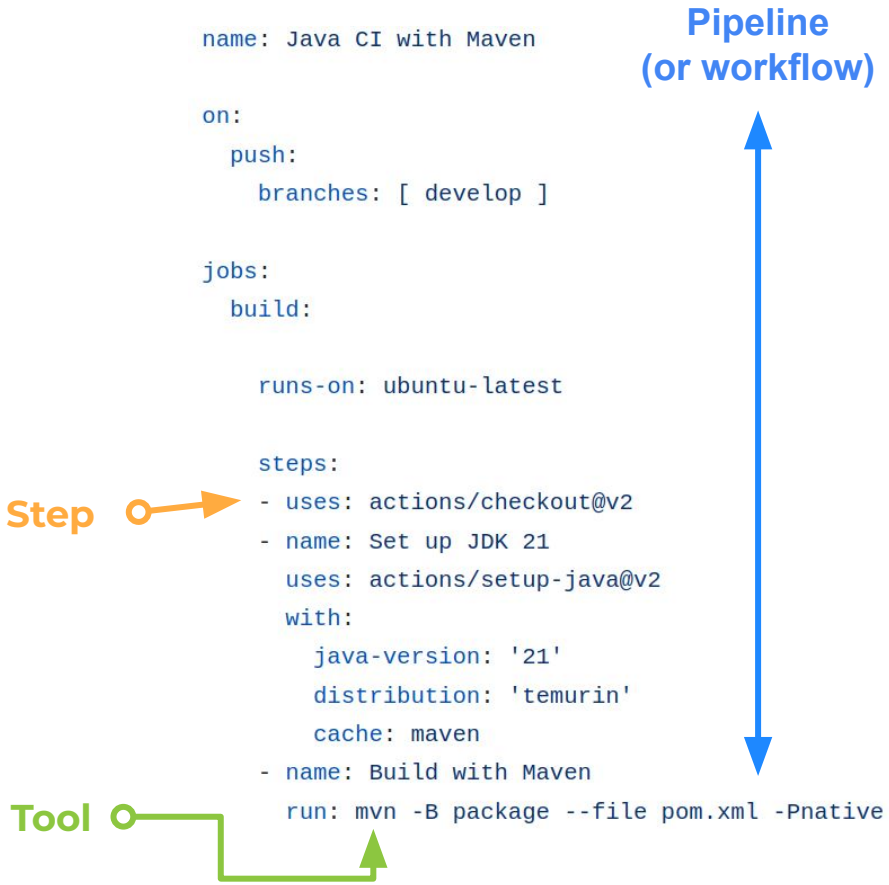


≃



What is a pipeline in DevOps and how to automate?

- **Pipeline (or workflow)** can be seen as an **assembly line** in a factory.
- A **pipeline** is a set of **steps** producing an executable/deployable software (a product) from source code (raw materials).
- **Pipelines** are executed on Continuous Integration and Continuous Delivery Platforms (CI/CD).
- **Pipelines** are file descriptors of **steps** and **tools** to used.



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

How can you automate m y infrastructure deployment?

With some slides of Olivier Barais from the University of Rennes (thanks!)

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

What is IaC?

Infrastructure as Code = using a descriptive language (Domain Specific Language) to code deployment mechanisms

More versatile and more adaptative than “classical” bash/shell scripts!

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

IaC for what?

- With these tools you can **provide and deploy** servers and global infrastructure
- With IaC you can:
  - Create specific deploiement steps for one or machine/VMs/Containers
  - Manage the IaC code and in fine managing the deploiement versions
  - Create some main “basic blocks” easily reusable
  - Test your deployed infrastructure

A taxonomy of the Infrastructure as Code

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

IaC tools

ANSIBLE · CFEngine · CHEF · juju · puppet · SALTSTACK · Terraform · NixOS · PalletOps · CloudFormation · Deployment Manager · Resource Manager · Docker Compose · heat (orchestrating multiple cloud applications) · HELM

Infrastructure mgt software

amazon web services · Google Cloud Platform · Microsoft Azure · openstack CLOUD SOFTWARE · kubernetes

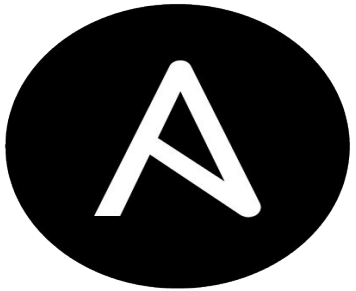
System level software

Vagrant · npm · KVM · docker · OvS Open vSwitch

Introduction - What is Ansible?

"Simple, agent-less and powerful open source IT automation Tool"

- Provisioning
- Configuration Management
- Application Deployment
- Continuous Delivery
- Security & Compliance
- Orchestration



ANSIBLE

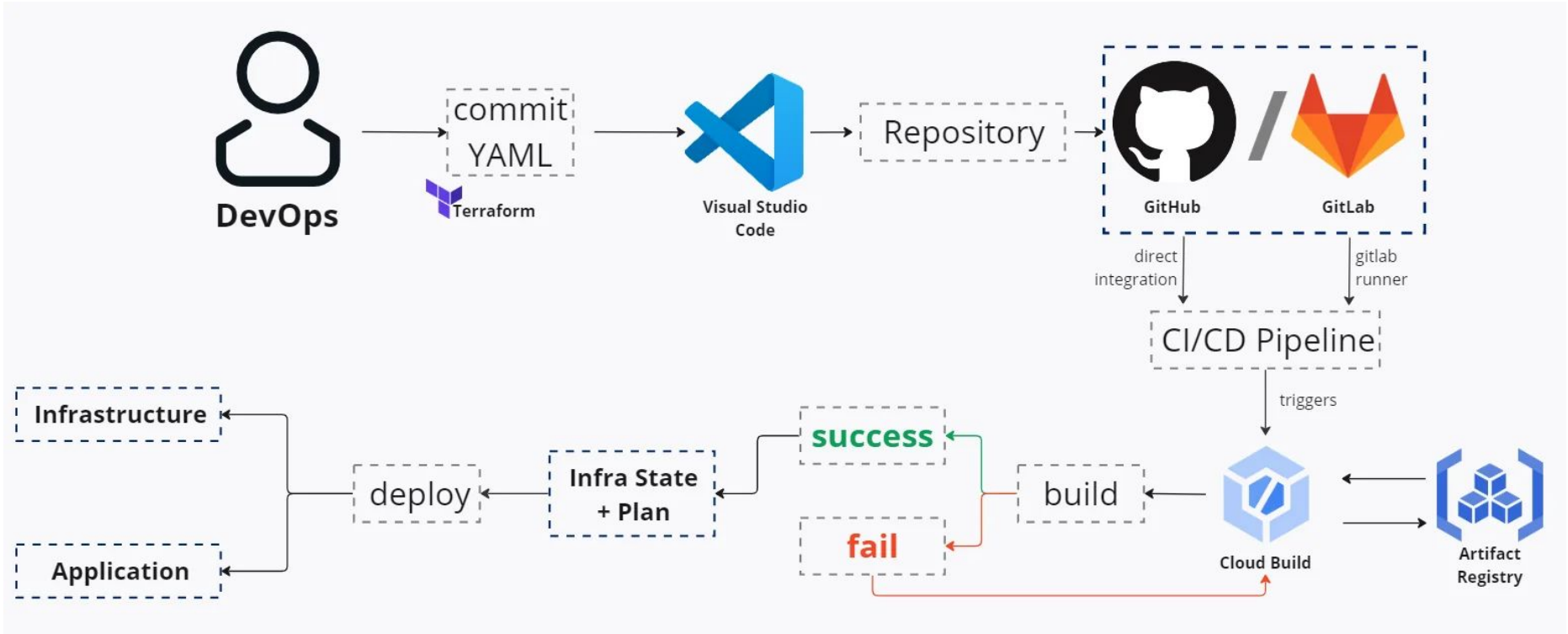
Products:

- Ansible CLI
- Ansible Tower

Red Hat

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

IAC Workflow



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Demo with Ansible for IoT environment deployment (Digital Twin for home automation)