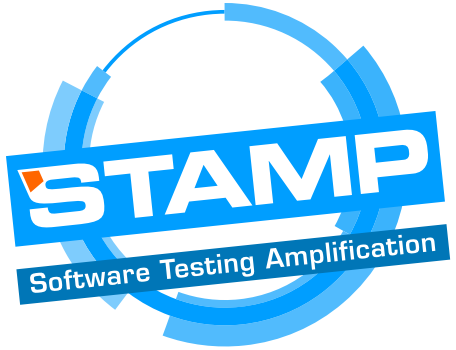
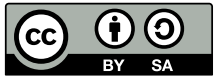


Software Testing AMPlification for the DevOps Team

WP 4 - CI/CD Scenario - Demo

Daniele Gagliardi, Ciro Formisano (ENG)

STAMP Project Final Review
Brussels, 6 February, 2020



The STAMP project received funding from European Union's Horizon 2020 research and innovation programme under grant agreement 731529.

Demo overview

- Scenario overview
  - STAMP in the CI Epic
  - Real projects selection
- STAMP in the CI user stories:
  - Story 1: Assess and amplify unit tests through Descartes and DSpot
  - Story 2: Amplify test configuration and perform system functional tests through CAMP
  - Story 3: Reproduce automatically a bug created in a Jira instance or in GitHub Issues through **Botsing**

Scenario overview: the STAMP CI/CD Epic

Story 1
As a Developer
I want to submit a code change to STAMP CI
In order to have amplified test cases
    In a dedicated branch within a PR
    And have information about how good and
     needed these test case are
      to accept them in the codebase

Scenario overview: the STAMP CI/CD Epic

Story 1
As a Developer
I want to submit a code change to STAMP CI
In order to have amplified test cases
In a dedicated branch within a PR
And have information about how good and
needed these test case
to accept them in the

Story 2
As a Developer
I want to system test my software
on different configurations
In order to identify compatibility issues
and identify
most performing configurations as well

Scenario overview: the STAMP CI/CD Epic

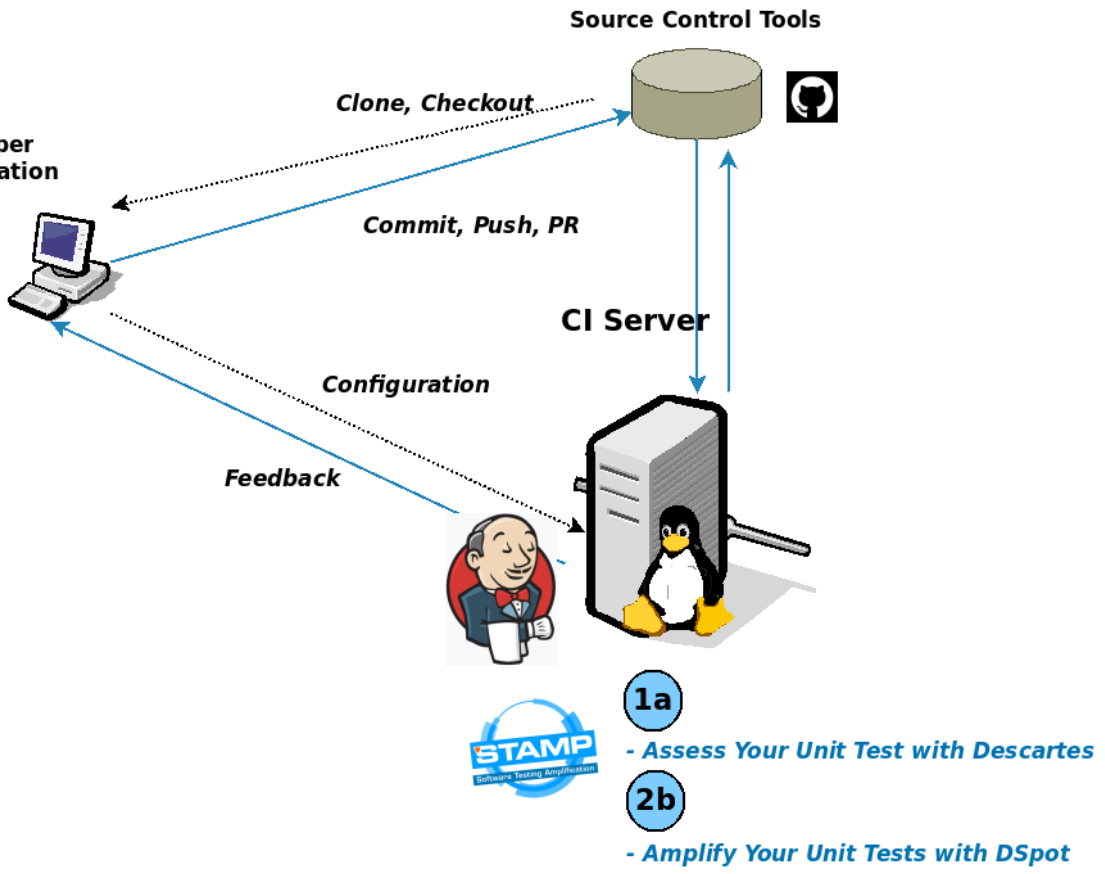
Story 1
As a Developer
I want to submit a code change to STAMP CI
In order to have amplified test cases
In a dedicated branch within a PR
And have information about how good and
needed these test case
to accept them in the

Story 2
As a Developer
I want to system test my software
on different configurations
In order to identify compatibility issues
and identify
most performing

Story 3
As a Developer
I want to speed up my bug fixing process
For runtime bugs

Story 1: unit test amplification with Descartes and DSpot

As a Developer
I want to submit a code change to STAMP CI
In order to have amplified test cases
In a dedicated branch within a PR
And have information about how good and needed these test case are
to accept them in the codebase

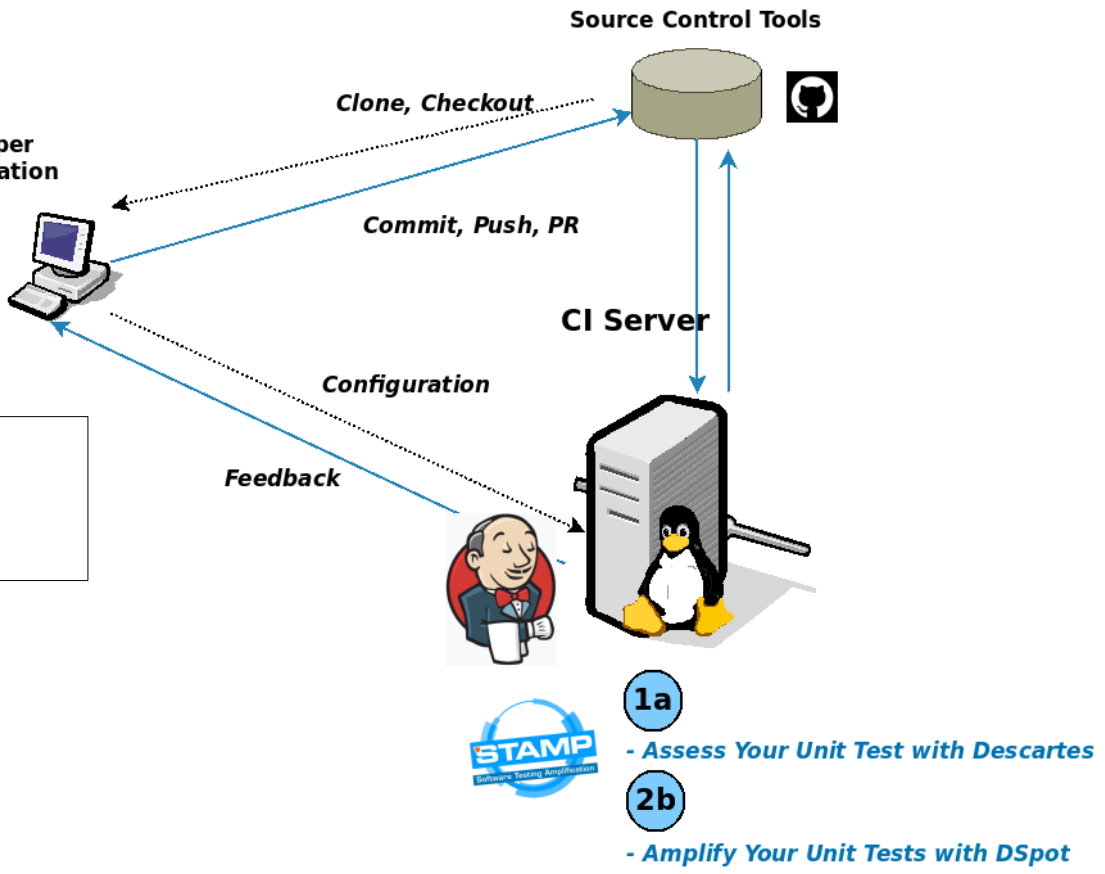


Story 1: unit test amplification with Descartes and DSpot

As a Developer
I want to submit a code change to STAMP CI
In order to have amplified test cases
In a dedicated branch within a PR
And have information about how good and needed these test case are
to accept them in the codebase

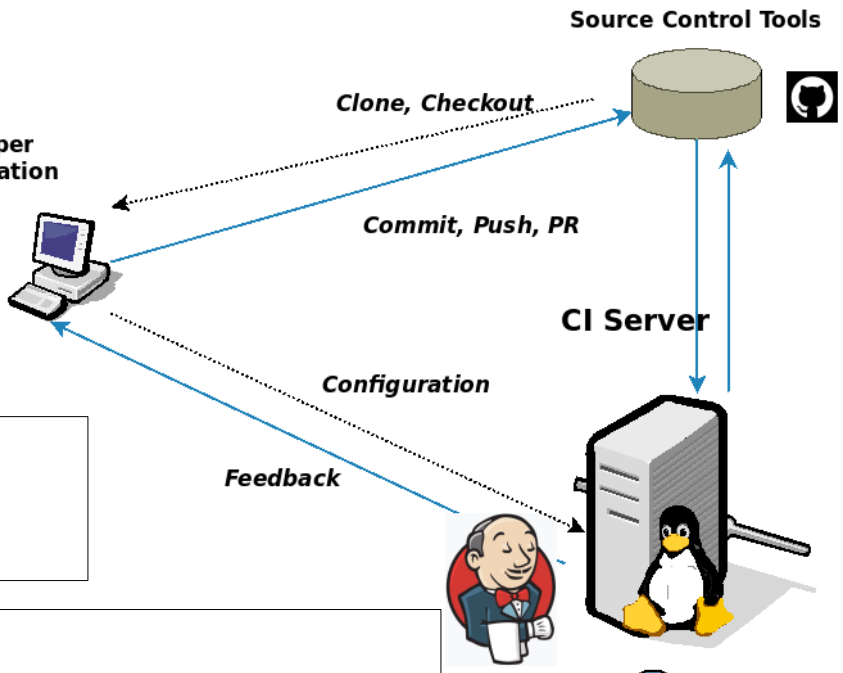
Story implementation

- Test your tests with Descartes in the CI
- Amplify your tests with DSpot in the CI



Story 1: unit test amplification with Descartes and DSpot

As a Developer
I want to submit a code change to STAMP CI
In order to have amplified test cases
In a dedicated branch within a PR
And have information about how good and needed these test case are
to accept them in the codebase



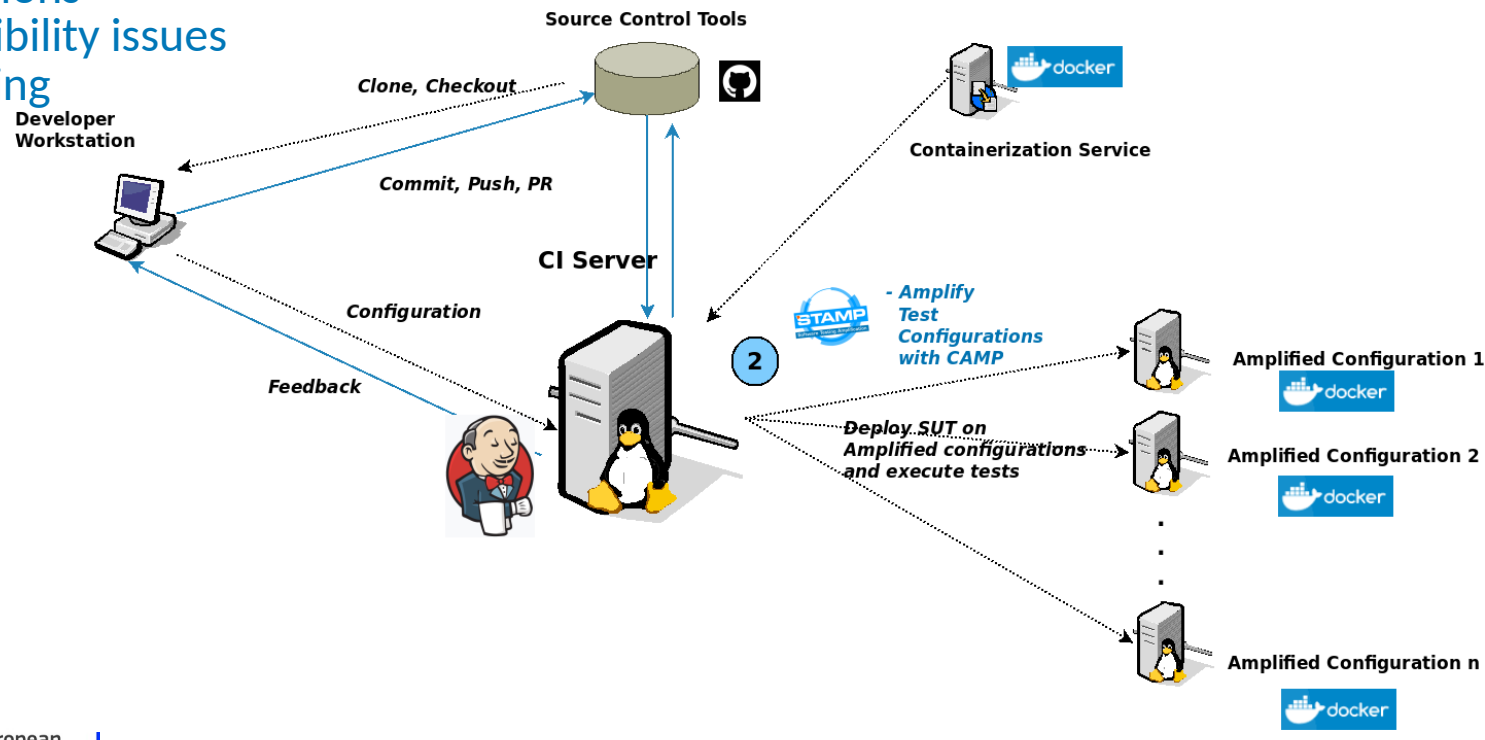
Story implementation

- Test your tests with Descartes in the CI
- Amplify your tests with DSpot in the CI

Outcome:
Codechange => amplified test cases =>
PR automatically opened =>
mutation report to provide developer with info to accept or reject the PR

Story 2: test configuration amplification with CAMP

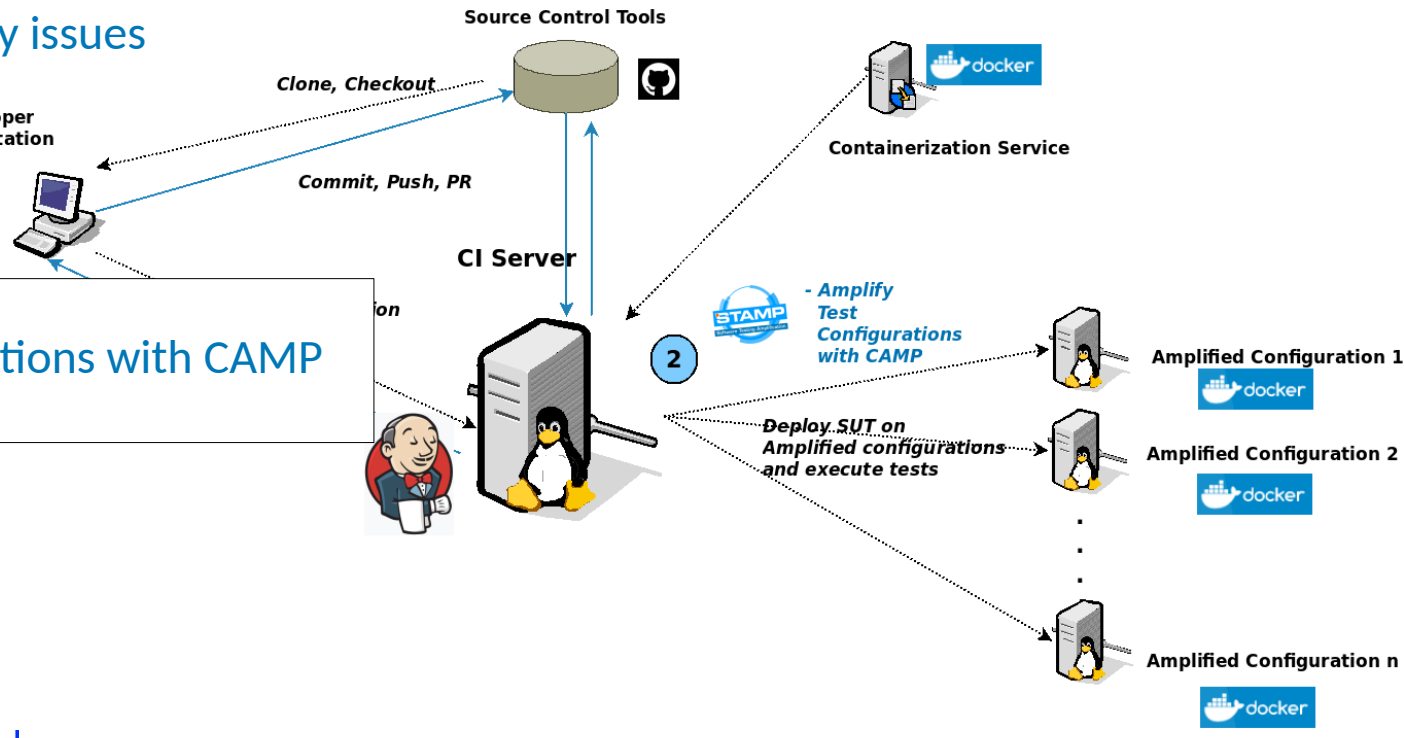
As a Developer
I want to system test my software
on different configurations
In order to identify compatibility issues
and identify most performing
configurations as well



European Commission

Story 2: test configuration amplification with CAMP

As a Developer
I want to system test my software
on different configurations
In order to identify compatibility issues
and identify most performing
configurations as well



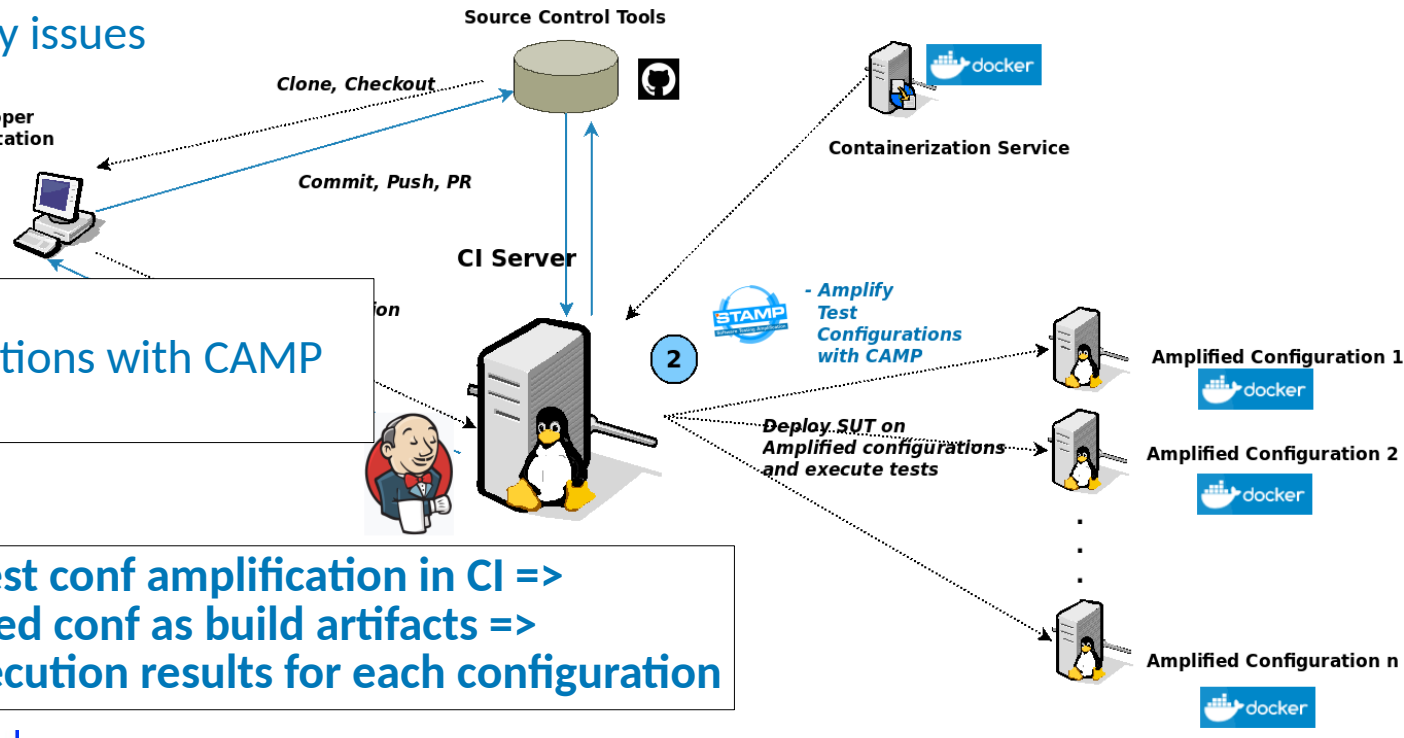
Story implementation

- Amplify your test configurations with CAMP in the CI

European Commission

Story 2: test configuration amplification with CAMP

As a Developer
I want to system test my software
on different configurations
In order to identify compatibility issues
and identify most performing
configurations as well



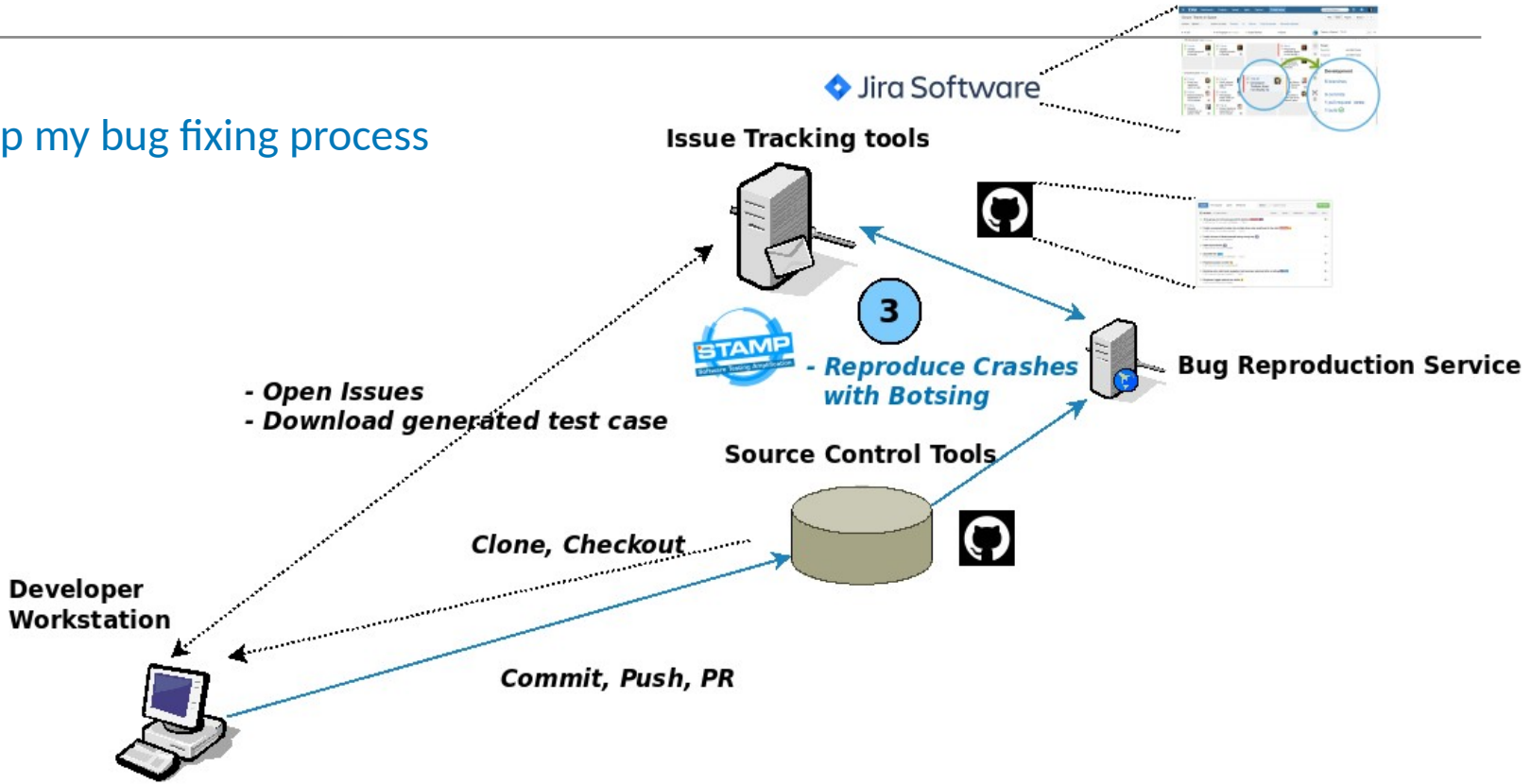
Story implementation

- Amplify your test configurations with CAMP in the CI

**Scheduled test conf amplification in CI =>
generated conf as build artifacts =>
execution results for each configuration**

Story 3: online test amplification with Botsing

As a Developer
I want to speed up my bug fixing process
For runtime bugs

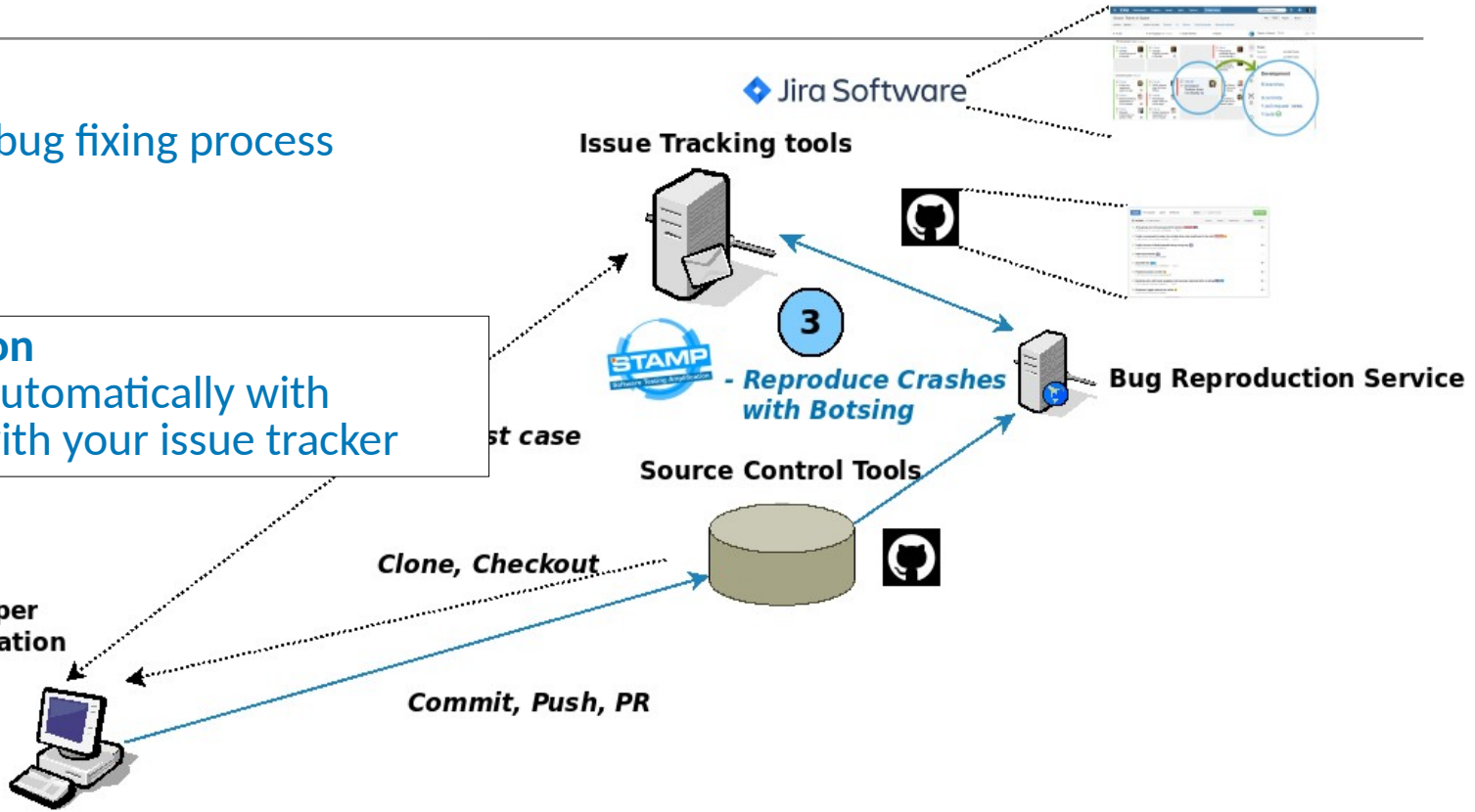


Story 3: online test amplification with Botsing

As a Developer
I want to speed up my bug fixing process
For runtime bugs

Story implementation

- Reproduce crashes automatically with Botsing integrated with your issue tracker



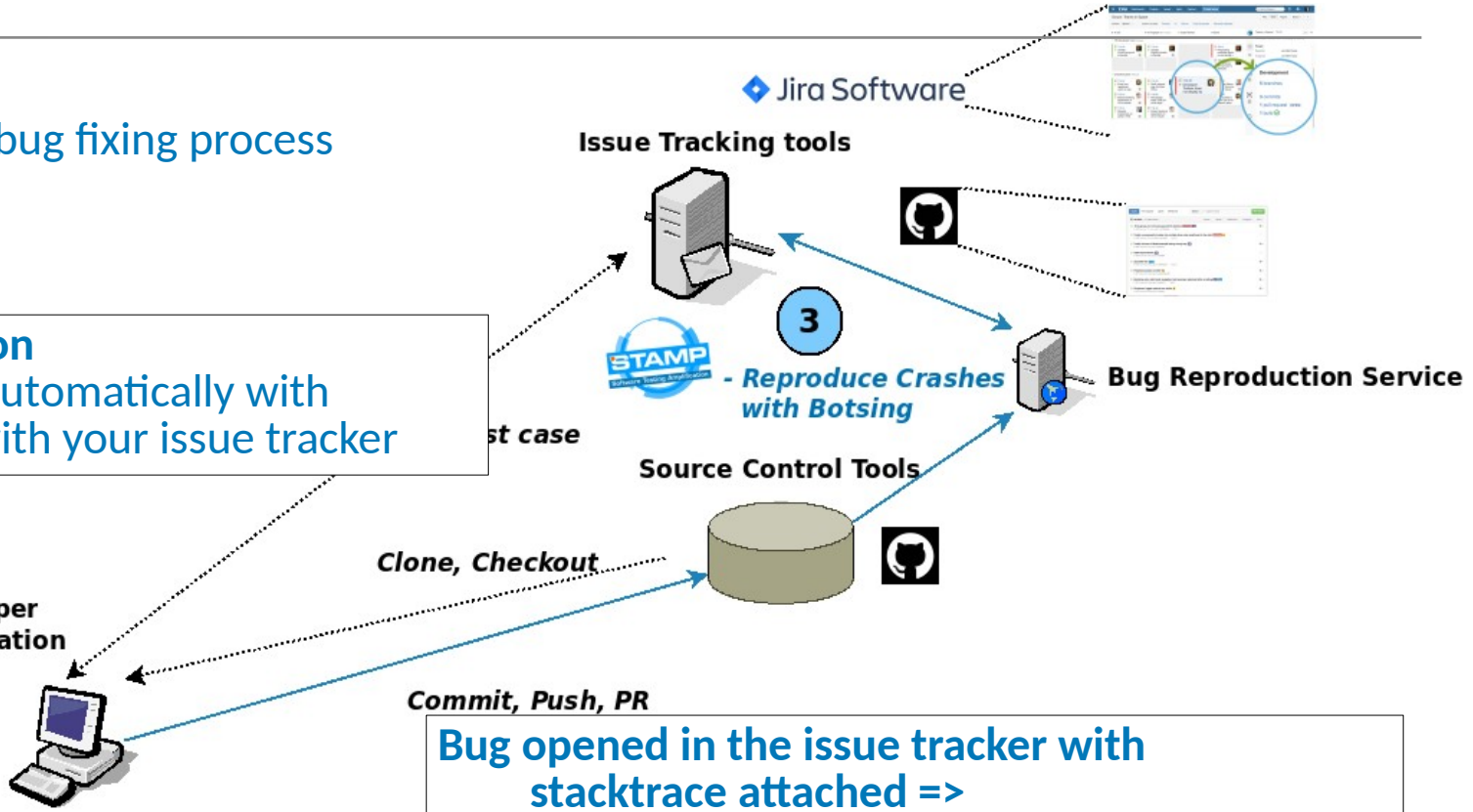
European Commission

Story 3: online test amplification with Botsing

As a Developer
I want to speed up my bug fixing process
For runtime bugs

Story implementation

- Reproduce crashes automatically with Botsing integrated with your issue tracker



Bug opened in the issue tracker with stacktrace attached => test case automatically generated able to reproduce the bug => made available in the issue tracker

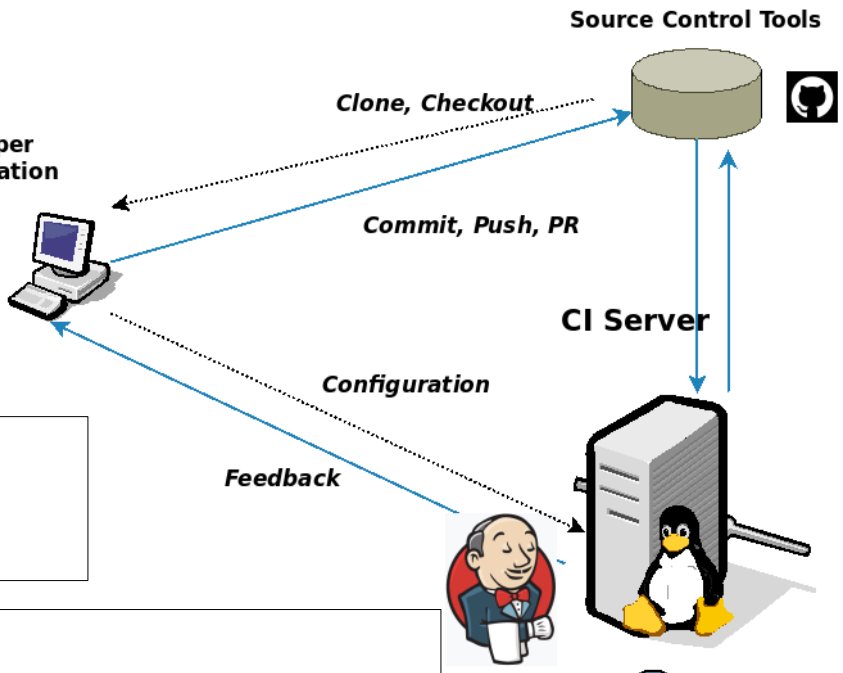
European Commission

Scenario overview: target projects

- **Joram**:
  - Open source Java implementation of JMS API spec
  - Selected to demonstrate workflow 1 (unit test assessment & amplification in CI)
- **Lutece:**
  - Open source CMS
  - Selected to demonstrate workflow 2 (test configuration amplification in CD)
- **Authzforce:**
- Open source attribute based access control framework
- Selected to demonstrate workflow 3 (online test amplification with issue trackers)

Story 1: unit test amplification with Descartes and DSpot

As a Developer
I want to submit a code change to STAMP CI
In order to have amplified test cases
In a dedicated branch within a PR
And have information about how good and needed these test case are
to accept them in the codebase



Story implementation

- Test your tests with Descartes in the CI
- Amplify your tests with DSpot in the CI

Outcome:
Codechange => amplified test cases =>
PR automatically opened =>
mutation report to provide developer with info to accept or reject the PR

Story 1: unit test amplification with Descartes and DSpot

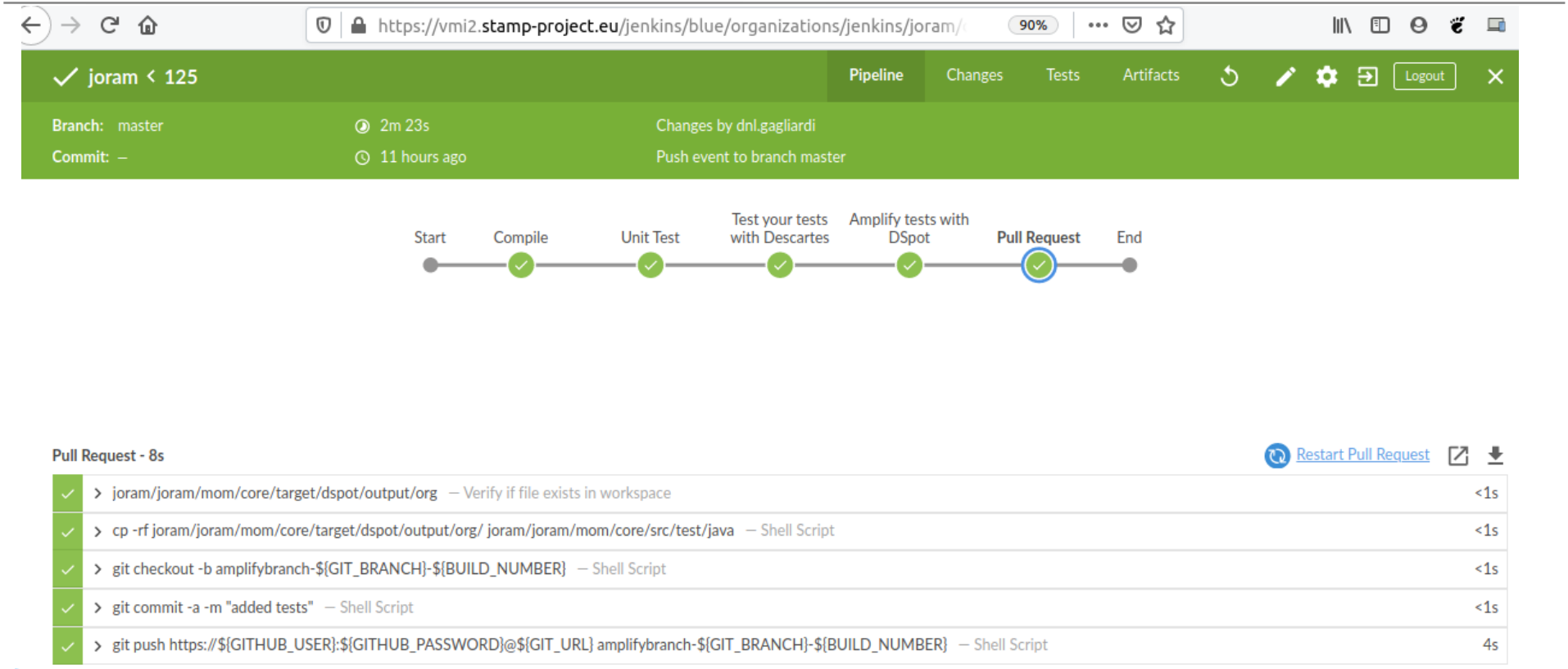
Objectives:

- Enhance continuous integration by amplifying unit tests
  - On test case changes, assess them with PitMP/Descartes
  - On code changes, amplify unit tests
- Make available new test cases on a separate branch and open a pull request

Configuration:

- Define a pipeline with PitMP/Descartes and DSpot steps
  - Use Jenkins changesets to trigger the execution
  - Add a step to open a Pull Request on generated test cases

Story 1: unit test amplification with Descartes and DSpot



Story 1: unit test amplification with Descartes and DSpot

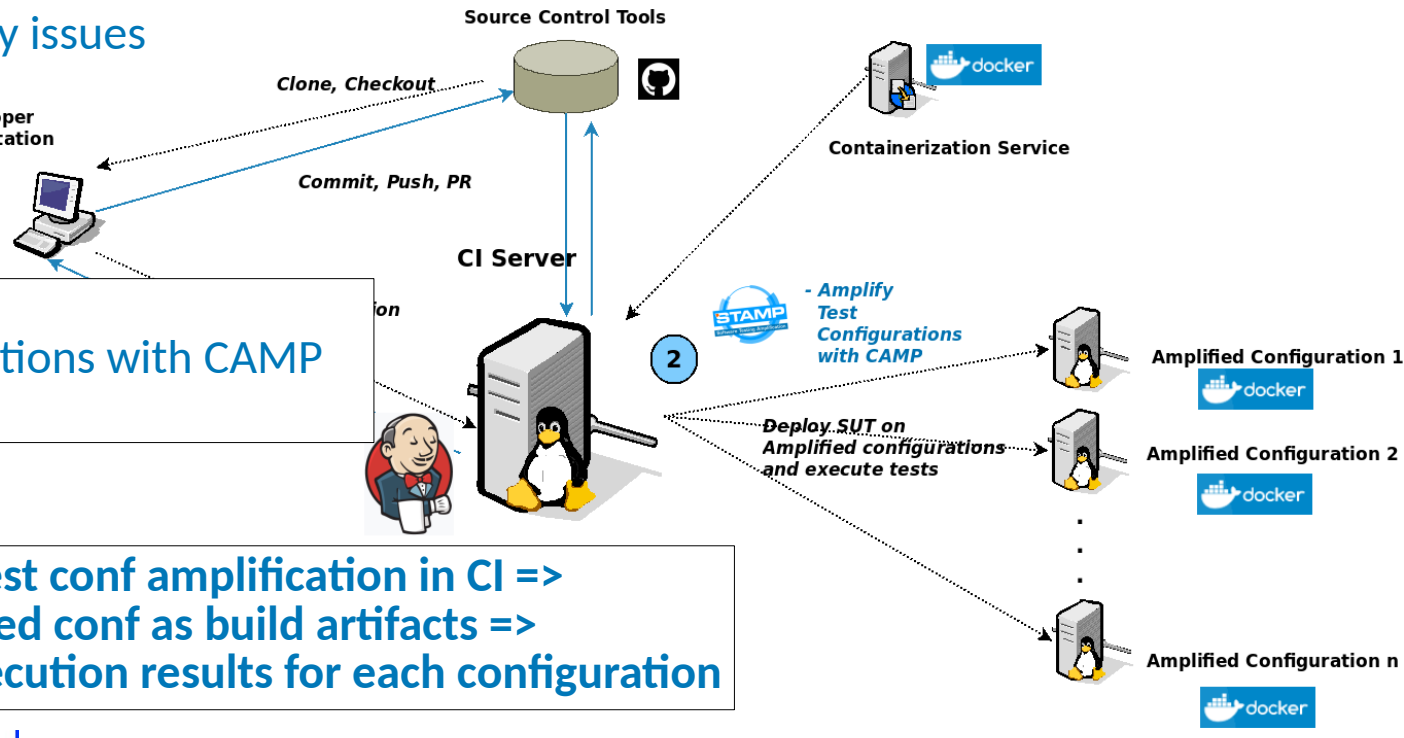
Demo time

Story 1: behind the scenes

1. Ordinary pipeline with usual build/unit test stages
2. A change in unit tests code injects test assessment, triggering the execution of Descartes/PitMP
3. A change in code injects test assessment and test amplification:
    1. Triggering execution of Descartes/PitMP
    2. Triggering execution of DSpot
    3. Creating a new branch containing amplified tests
    4. Opening a Pull Request on master branch
4. Any other change is not considered for test amplification

Story 2: test configuration amplification with CAMP

As a Developer
I want to system test my software
on different configurations
In order to identify compatibility issues
and identify most performing
configurations as well



Story implementation

- Amplify your test configurations with CAMP in the CI

**Scheduled test conf amplification in CI =>
generated conf as build artifacts =>
execution results for each configuration**

Story 2: test configuration amplification with CAMP

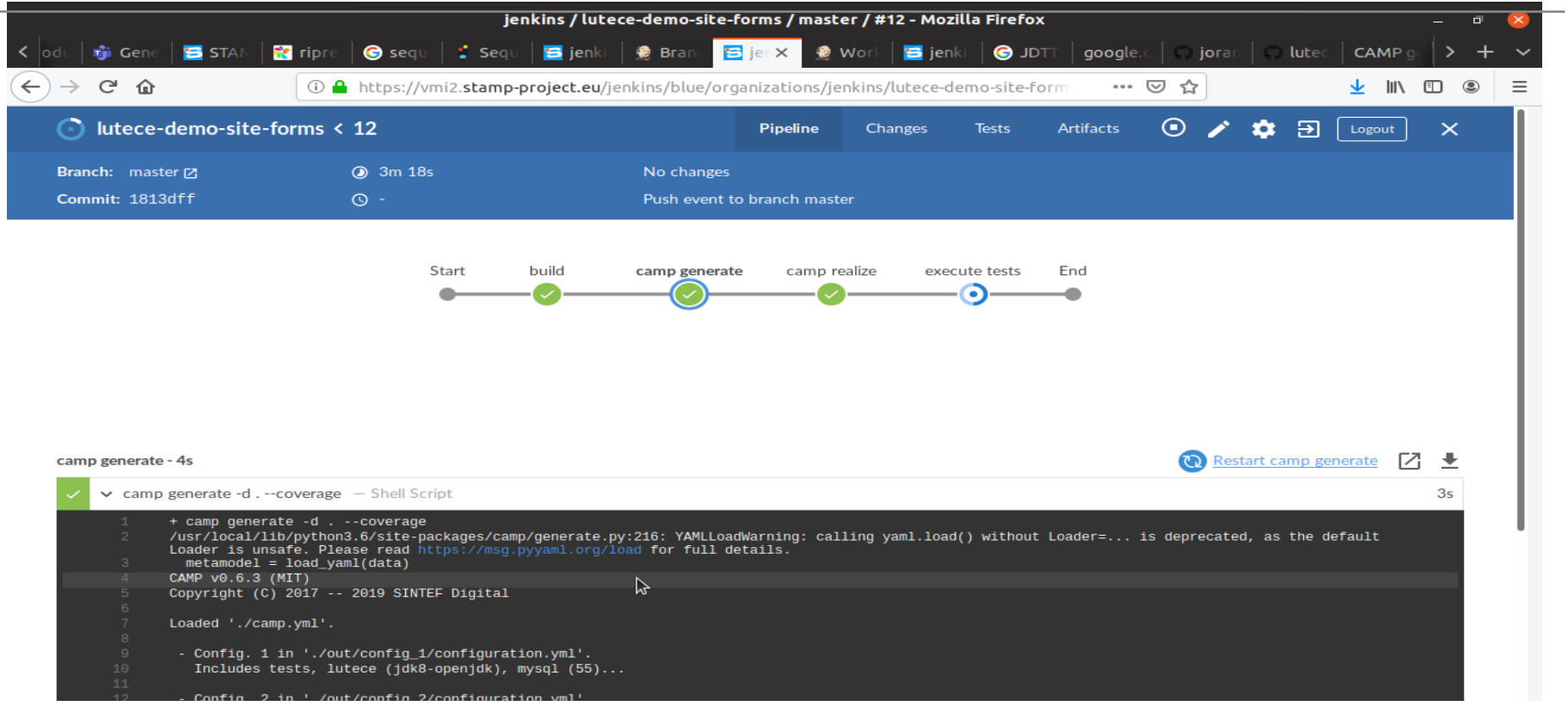
Objective:

- Enhance continuous delivery with test configuration amplification
  - On configuration changes/code changes/**according to a schedule**, launch CAMP
  - CAMP will:
    - Generate new test configurations
    - Execute integration/system tests against generated configurations

Configuration:

- Define a pipeline with CAMP actions steps
- Use Jenkins changesets to trigger the execution only on configuration changes

Story 2: test configuration amplification with CAMP



European Commission

Story 2: test configuration amplification with CAMP

- Test level: System
- Test Type: Functional

Demo time

Story 2: test configuration amplification with CAMP

- Test level: System
- Test Type: Performance

Demo time

Story 2: behind the scenes

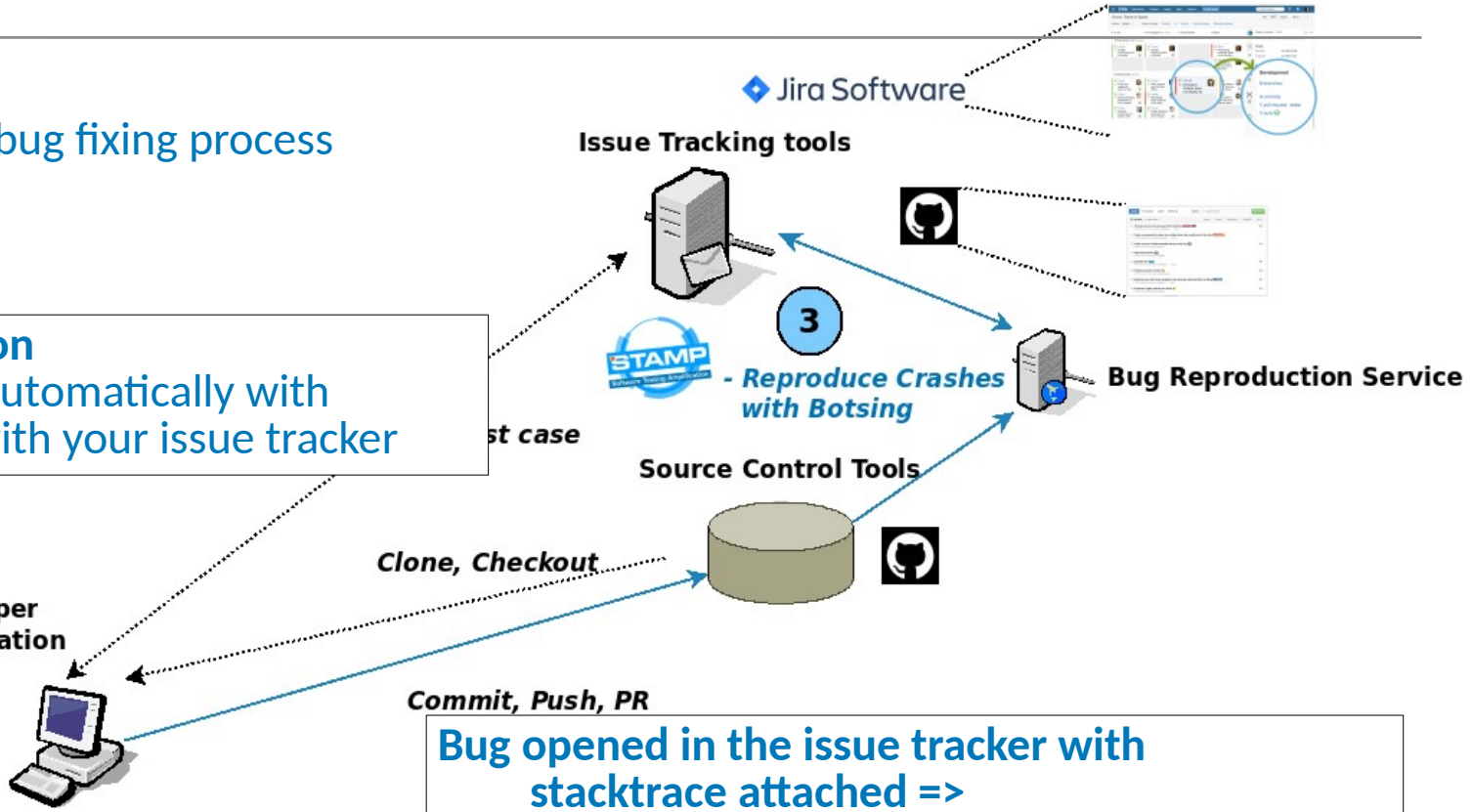
1. A change on available configurations/a code change /a scheduled run triggers configuration amplification with CAMP
2. CAMP generate new configurations:
    1. The parameter ‘--all’ enables the generation of all possible configurations
    2. The parameter ‘--coverage’ allows the generation of the smallest number of possible configurations covering all the features
3. Generated configurations are stored as current build (zipped) artifacts
4. CAMP execute executes integration/system tests against generated configurations

Story 3: online test amplification with Botsing

As a Developer
I want to speed up my bug fixing process
For runtime bugs

Story implementation

- Reproduce crashes automatically with Botsing integrated with your issue tracker



Bug opened in the issue tracker with stacktrace attached => test case automatically generated able to reproduce the bug => made available in the issue tracker

European Commission

Story 3: online test amplification with Botsing

Objectives:

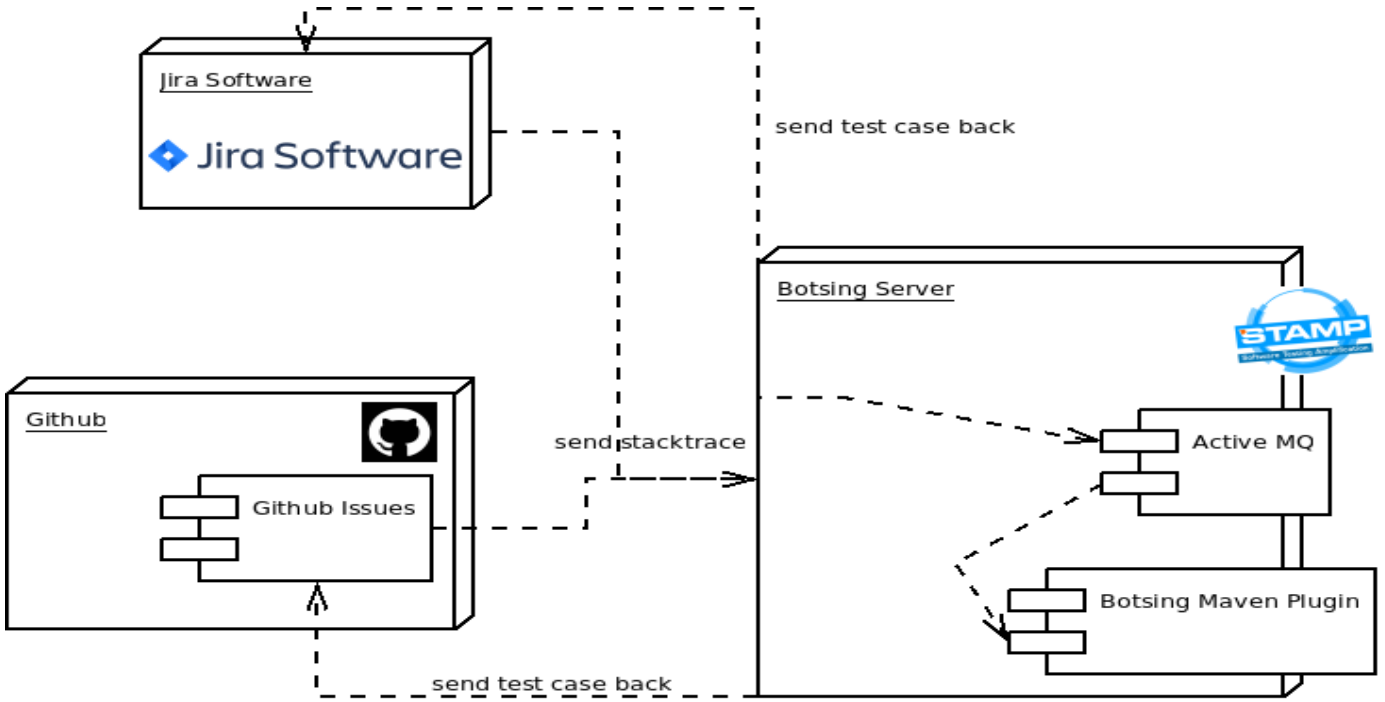
- Automatically reproduce crashes from stacktraces
- Only a stacktrace needed as input
- Stacktraces made available by issue trackers. Currently the supported issue trackers are:
  - Jira Software
  - Github Issues
- Generated test cases made available within the issue tracker

Configuration:

- Jira (Jira Software with a perpetual license for ten users (10$)):
  - Configure the Jira Botsing plugin (GAV + Botsing parameters)
- Github Issues:
  - Add a .botsing configuration file in the repo (GAV + Botsing parameters)
- Configure Botsing Server (Github account, Jira account)

Story 3: online test amplification with Botsing

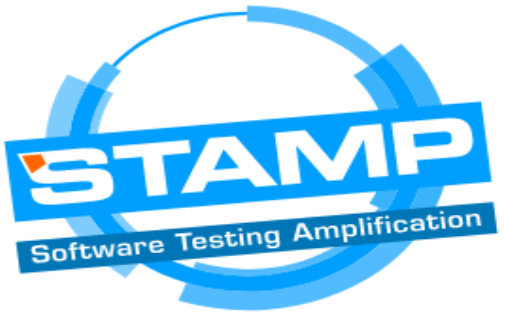
- How does it work?



European Commission

Story 3: online test amplification with Botsing

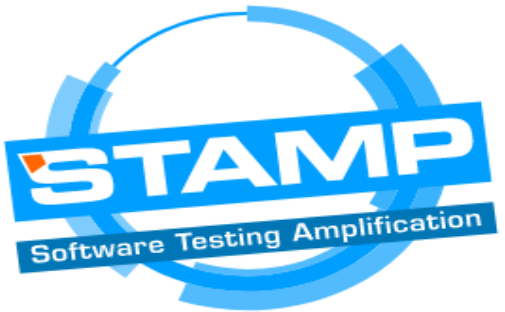
- Jira Integration



Demo time

Story 3: online test amplification with Botsing

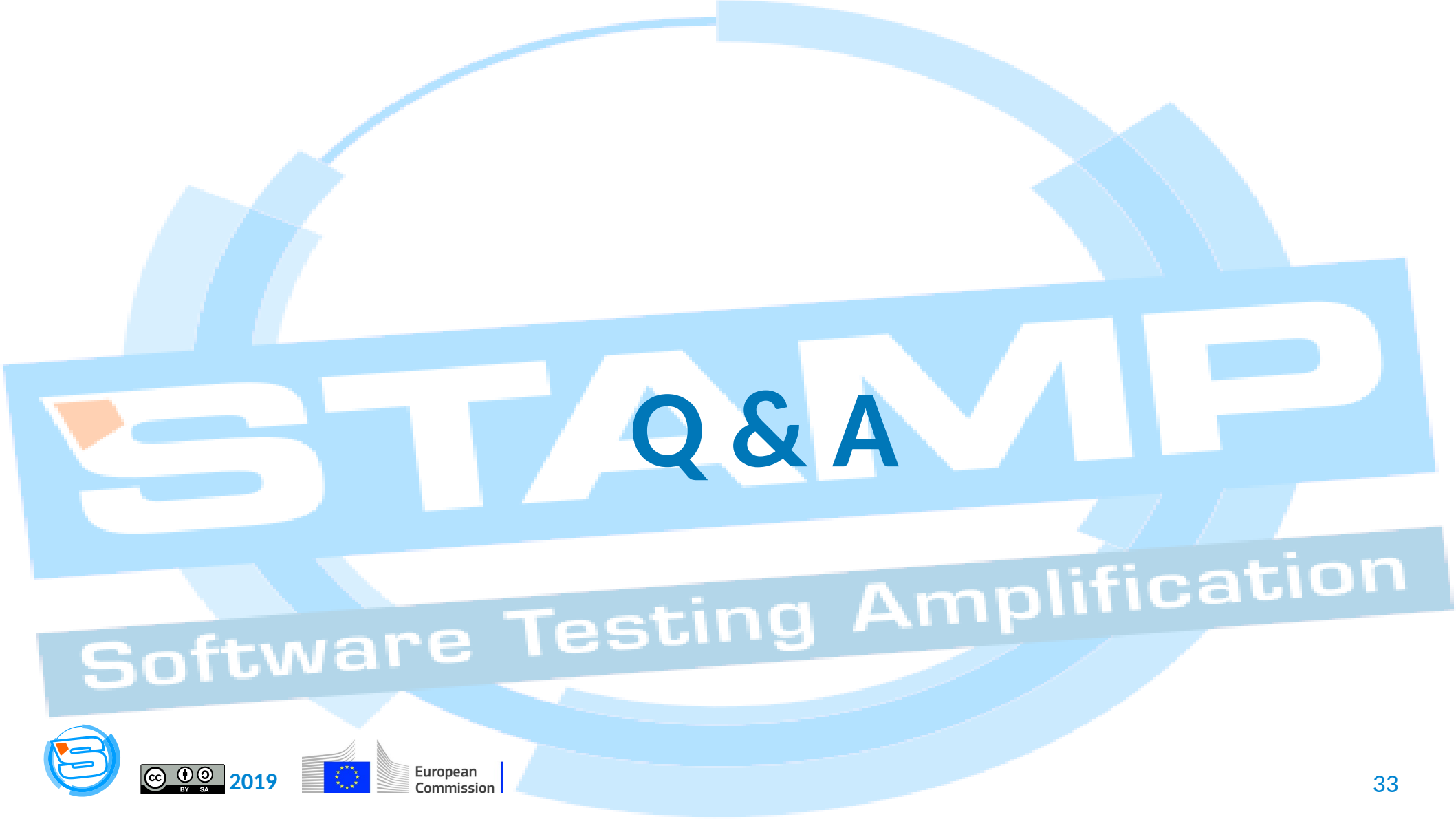
- GitHub Issues Integration



Demo time

Story 3: behind the scenes

1. Botsing Server: Spring Boot microservice + ActiveMQ to manage crash reproduction queues
2. Jira:
    1. When issue labeled as “STAMP”, crash reproduction request is sent to Botsing Server message queue
    2. Botsing server manages each request as follows:
        1. Botsing Preprocessor “*cleans*” the input stacktrace
        2. Botsing checks the classpath libraries/dependencies/binaries (from Maven Central/from local pom.xml/from local folder)
        3. Botsing execution starts from highest frame level until a valid test case is found
        4. Generated test case is sent back to Jira as attachment
        5. In case of failure, a comment is added to the Jira issue containing information on the failure
3. GitHub Issue Tracker
    1. When issue labeled as “botsing”, crash reproduction request is sent to Botsing Server message queue
    2. Botsing server manages each request as shown previously
    3. Send generated test case back to GitHub issue tracker in form of comments



Q&A

European Commission

Thank you!

The STAMP project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731529.