Continuous Testing at Scale

TAPOST Conference October 12th 2016, Riga

dmitry@buzdin.lv

@buzdin

Dmitry Buzdin

Introduction to Continuous Testing

Continuous Delivery

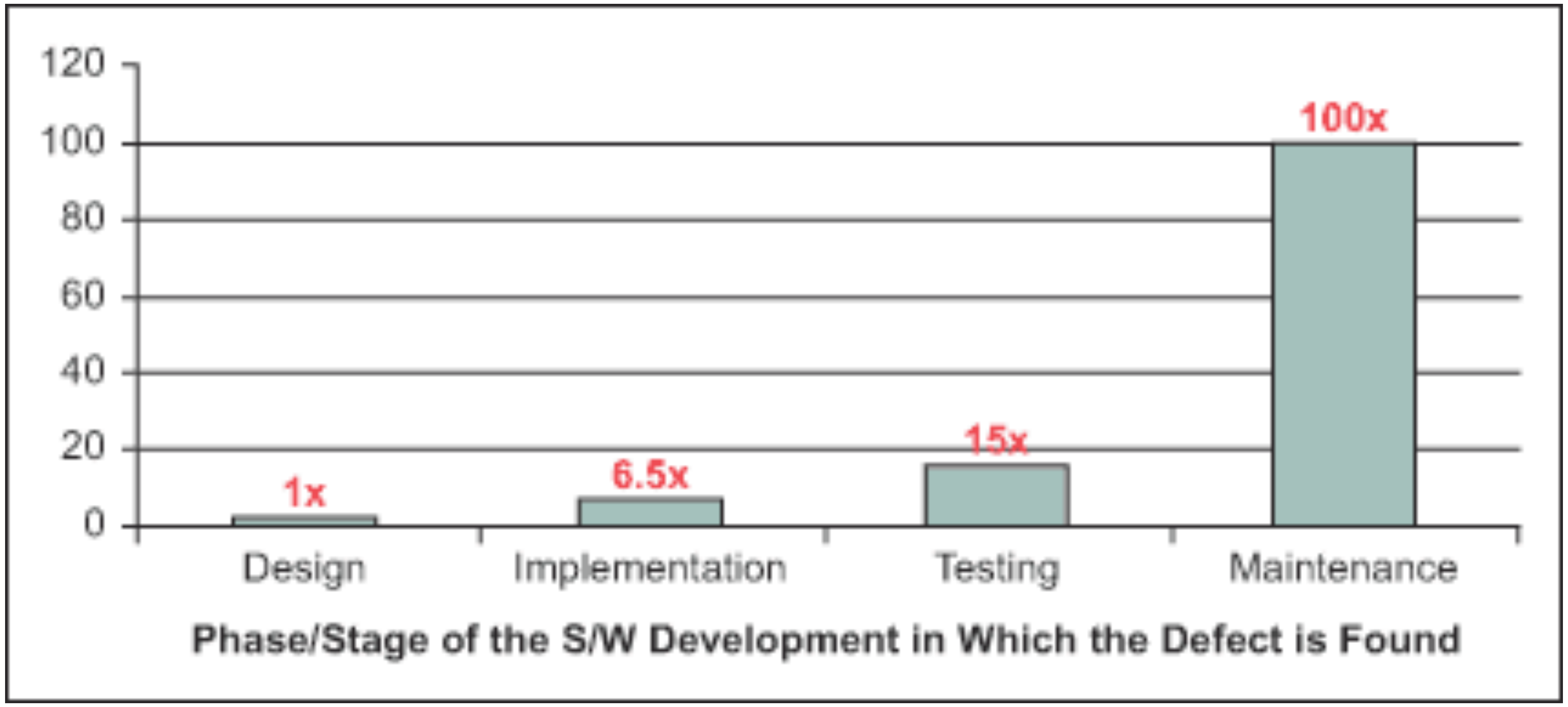
- Get changes to production in fast and efficient way
- Deployment happens often and can be performed on demand
- Code is always in a deployable state

https://continuousdelivery.com

Why CD?

- Frequent release cycle
- Decreased time to market
- Decreased delivery risks
- Commercial product development

Cost of Defect Fixing

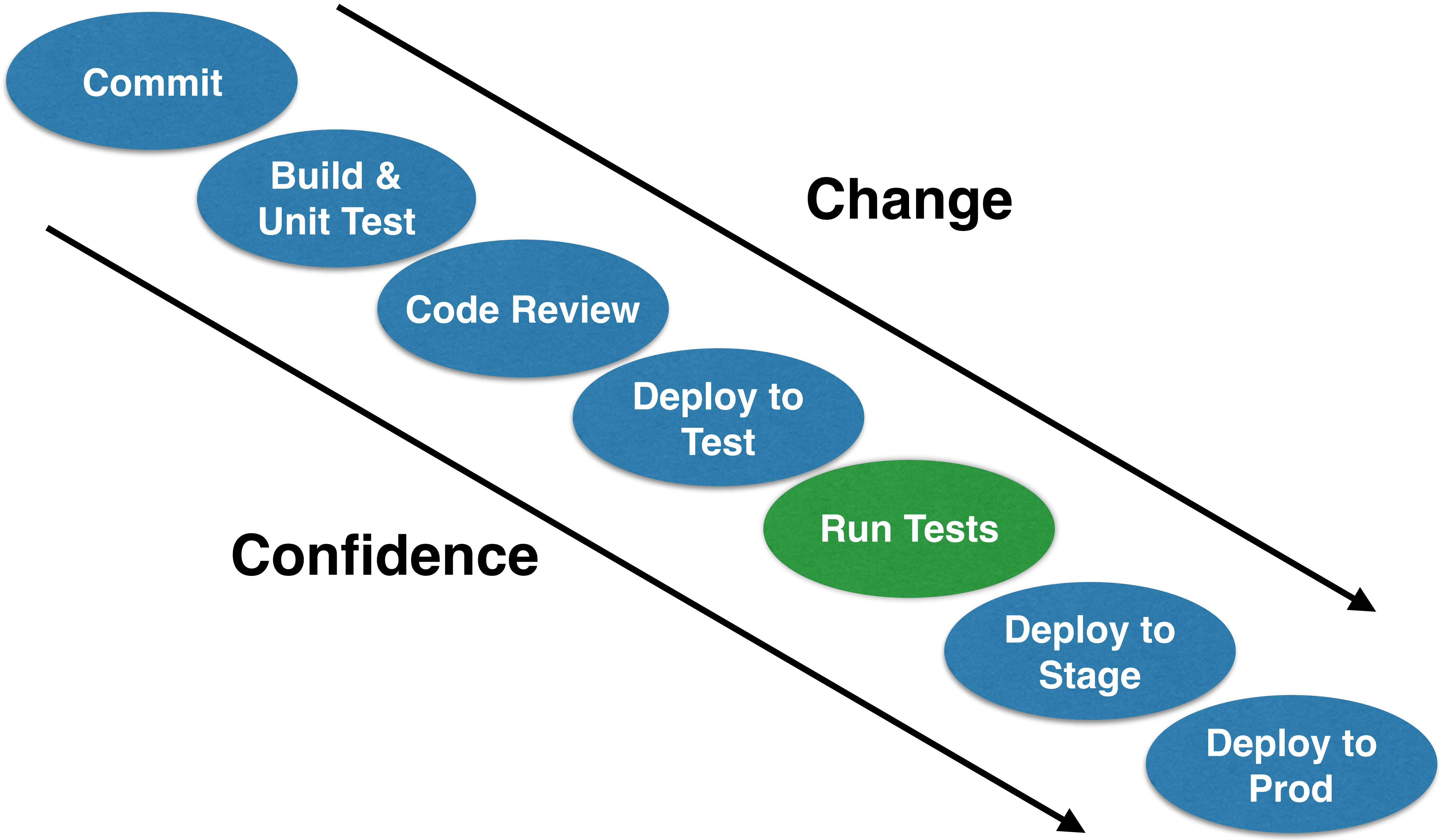


This chart is slightly different in CD
Testing is performed as frequently as possible

Continuous Delivery Pipeline

- Every code change needs to be verified
- Pipeline breaks the delivery process into stages
- Pipeline stages are not standardised

Pipeline Example



What is in “Test” Phase?

- Simplest scenario - Write tests in your favoured test framework and execute!

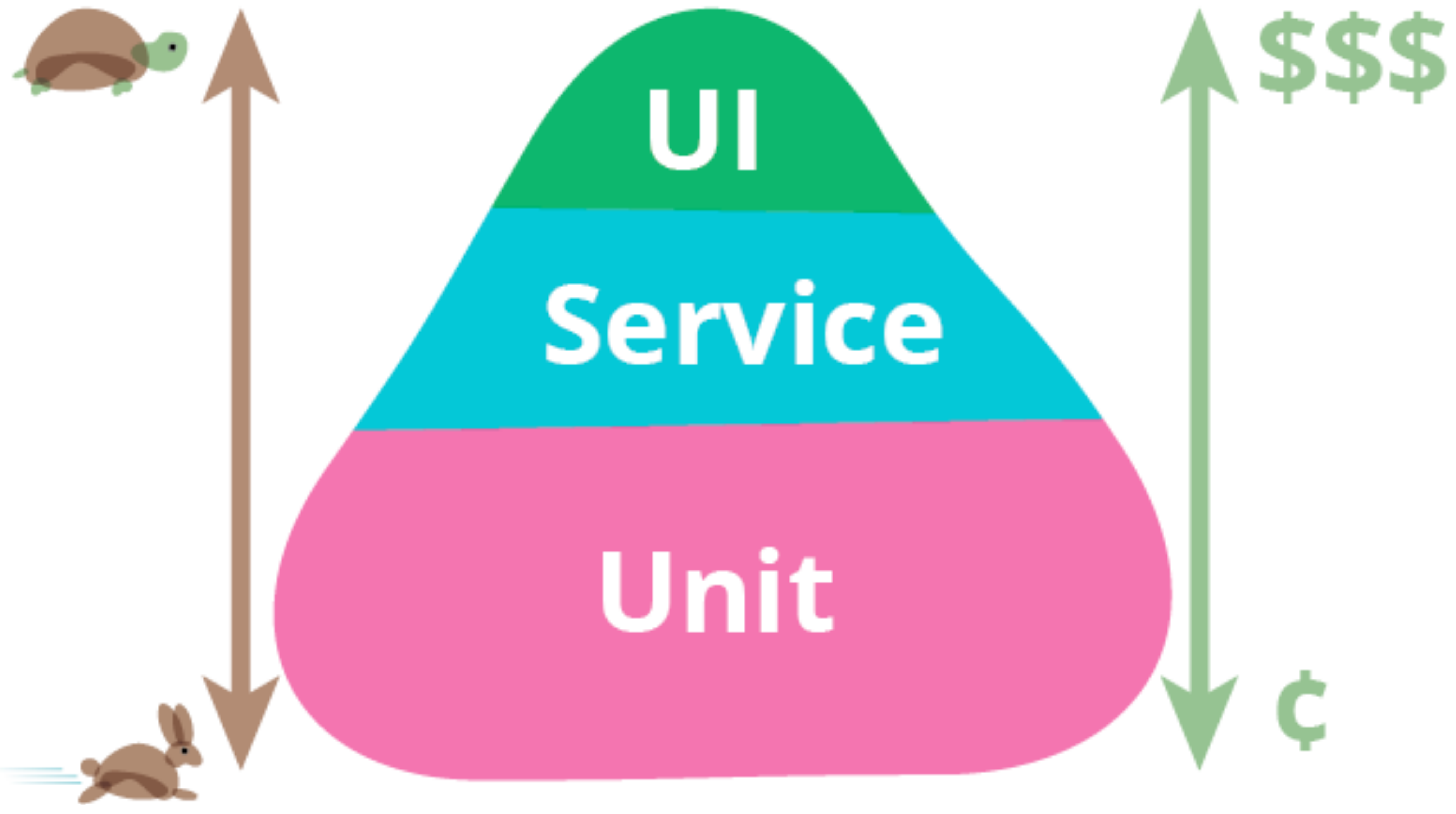
Agile Testing Strategy

"Developers write some integration tests and execute them in the pipeline"

Have Devs Considered?…

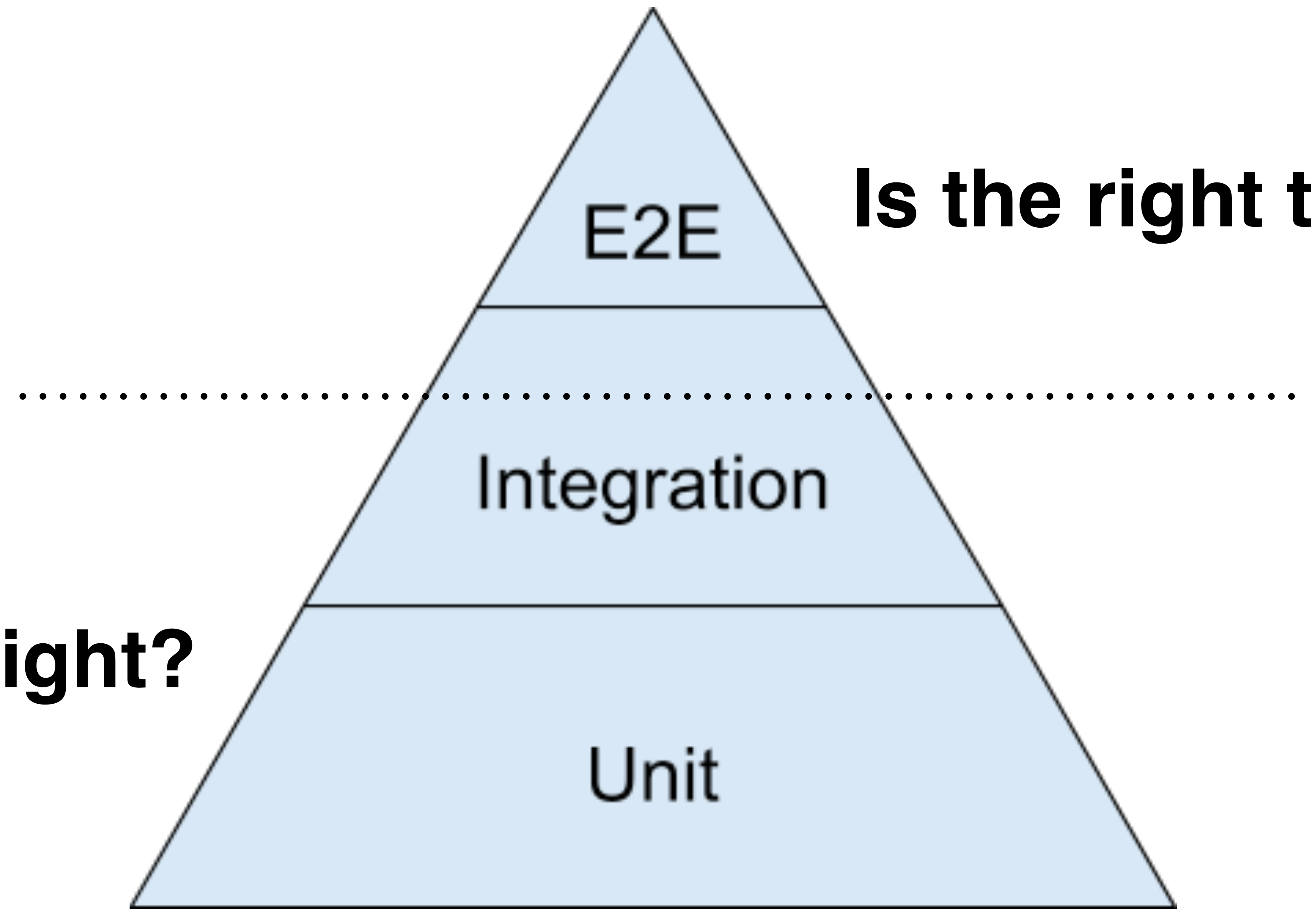
environment live checks
smoke testing
functional testing
security testing
performance testing
fault-tolerance testing
browser compatibility testing
mobile testing
test environment preparation
test planning
test scheduling
test reporting
testing process traceability

Tests Pyramid



http://martinfowler.com/bliki/TestPyramid.html

Another Pyramid



Is the right thing built?

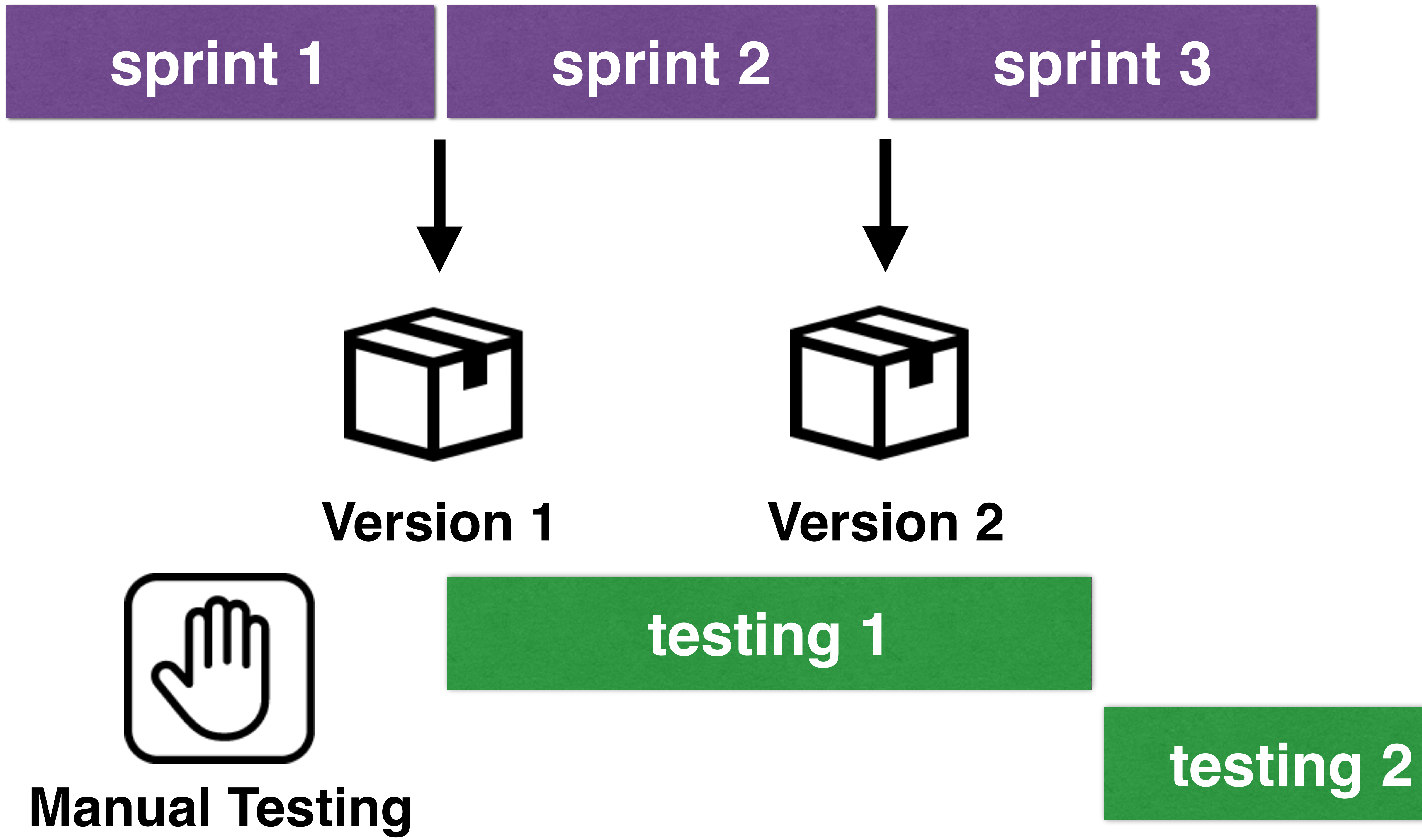
Is it build right?

https://testing.googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html

What tend to happen is

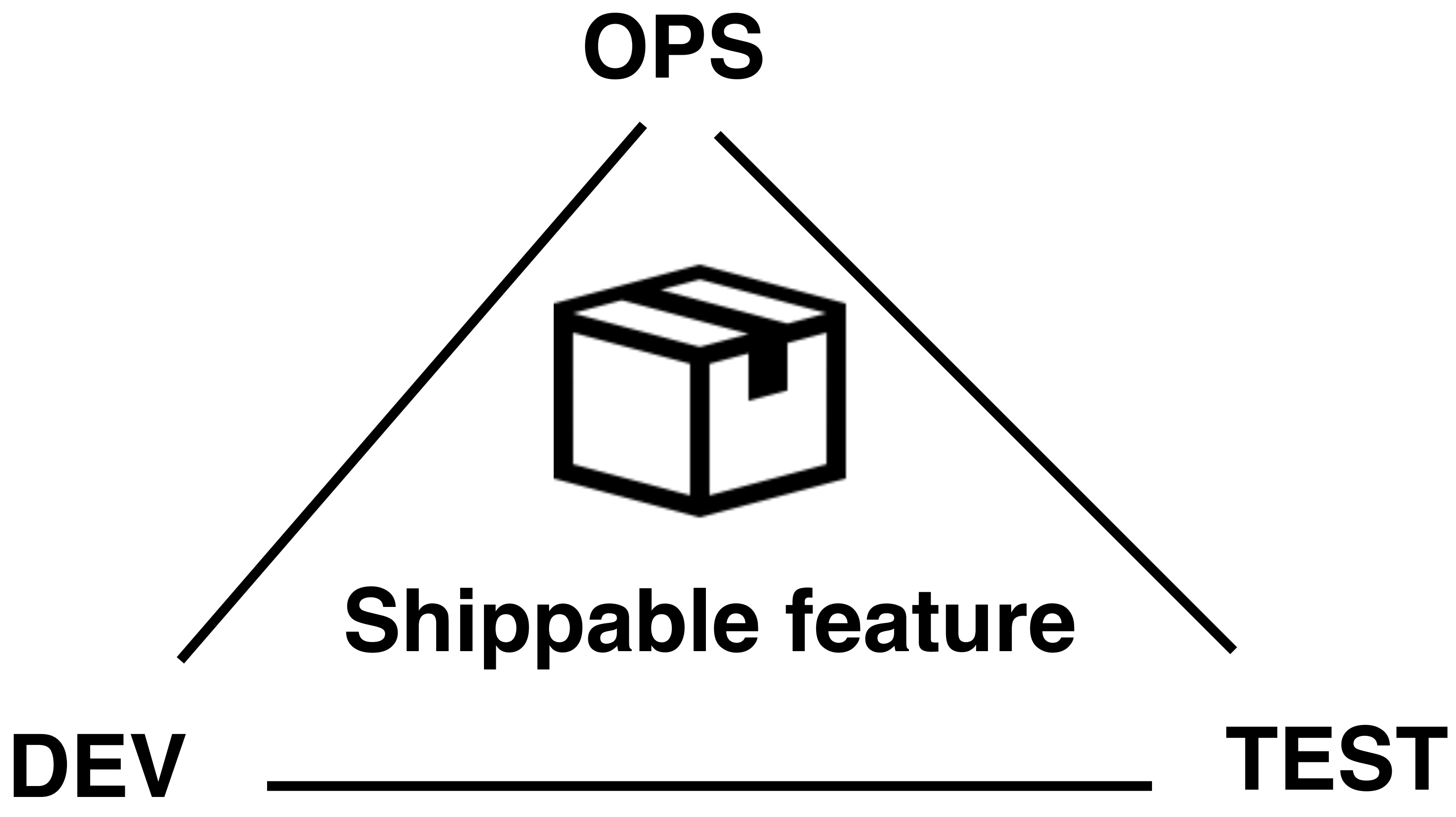
“Development focused CD”

Development focused CD



Testers are lagging behind a continuous delivery development train

“**Continuous Testing** *is the process of executing automated tests as part of the software delivery pipeline to obtain immediate feedback on the business risks associated with a software release candidate.*”



https://en.wikipedia.org/wiki/Continuous_testing

Developers

Apply TDD
No bureaucracy
All automated tests
Ad-hoc decisions
Feature-level testing

How do you align?

Test management
Test plans
Manual testing
Non-functional testing
Structured approach
Standards compliancy

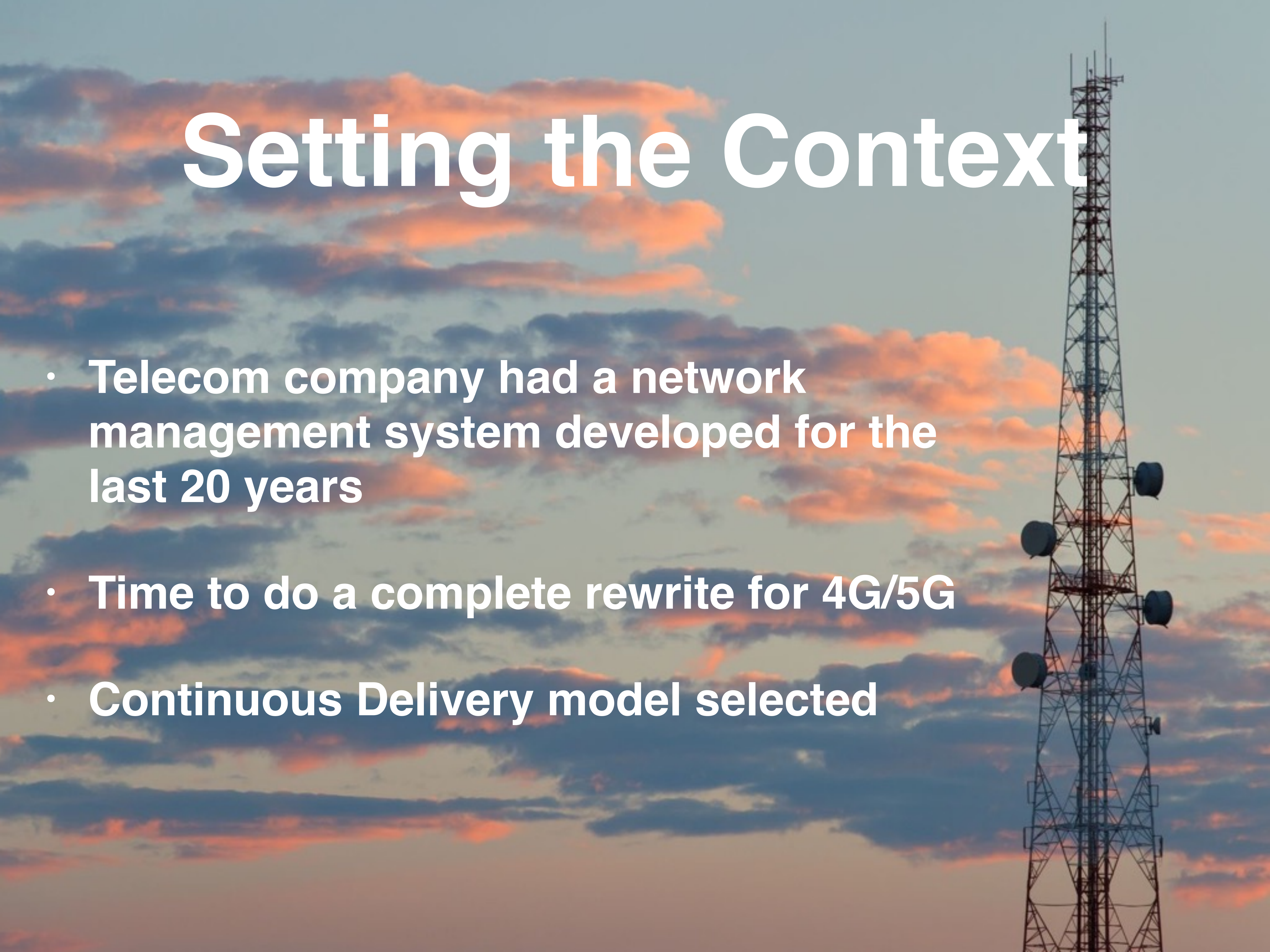
Testers

Is Continuous Testing easy?

Not at all!

Some advices to follow…

Story Begins



Setting the Context

- Telecom company had a network management system developed for the last 20 years
- Time to do a complete rewrite for 4G/5G
- Continuous Delivery model selected

100+
Scrum teams

2000+
Git Repositories

1000+
Components

Everything
Automated

10000+
Functional Tests

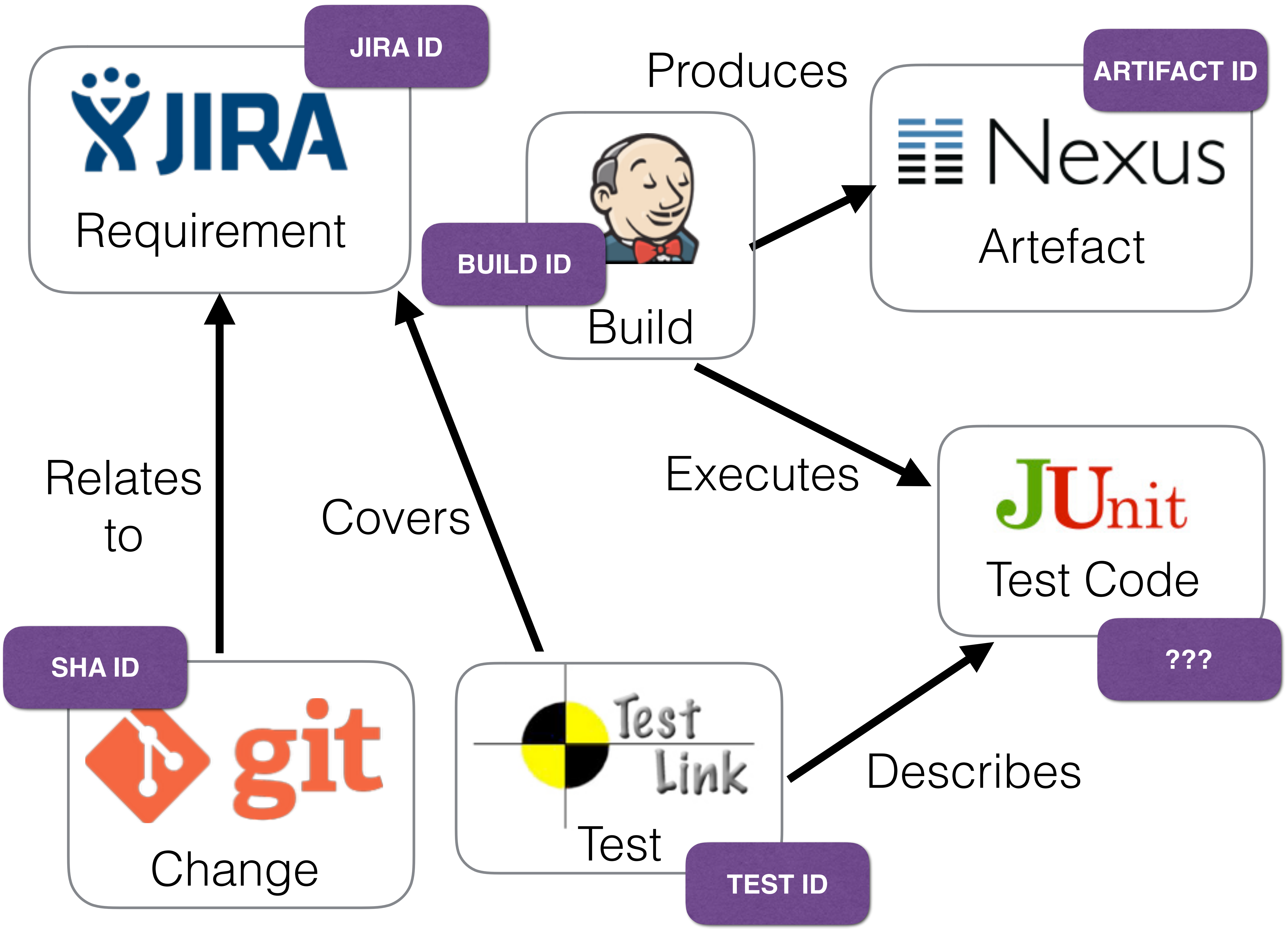
200+
Test Environments

Expectations

- Fast feedback cycle
- System quality statement few times a day
- Testing compliant to quality standards
- Traceability throughout the process
- Agile development model

Traceability is the ability to verify the history, location, or application of an item by means of documented recorded identification.

Boring bureaucratic stuff for non-agile environments only, right?


JIRA ID
Requirement
Produces
ARTIFACT ID
Nexus
Artefact
BUILD ID
Build
Relates to
Covers
Executes
JUnit
Test Code
???
SHA ID
git
Change
Test Link
Test
TEST ID
Describes

Test ID in Code

```
@TestCaseId("TMS-1")
public void testSomething() {
    ...
}
```

Every test should have a unique test id

```
@Features("My Feature")
@Stories({"Story1", "Story2"})
@Test
public void myTest() {
    ...
}
```

Test case related meta-data

Test Case Management

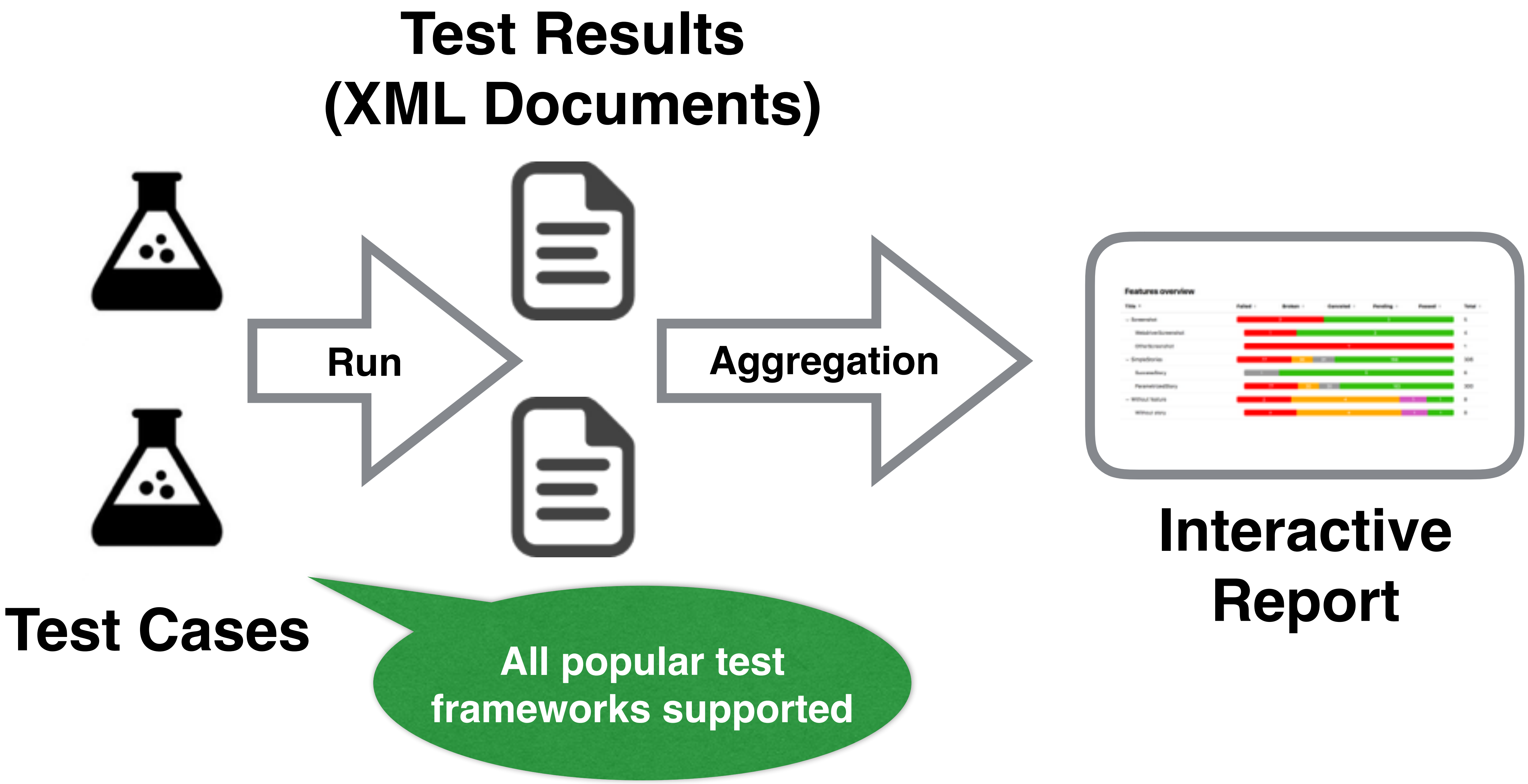
```java
@TestCaseId("TMS-1")
public void testSomething() {
    ...
}
```

Relates to a test description in TMS

```
Test ID: TMS-1
Title: Test database connection
Requirements: JIRA-124, JIRA-234
Priority: CRITICAL
Components: DB
Tags: Regression, DB
Description: ...
```

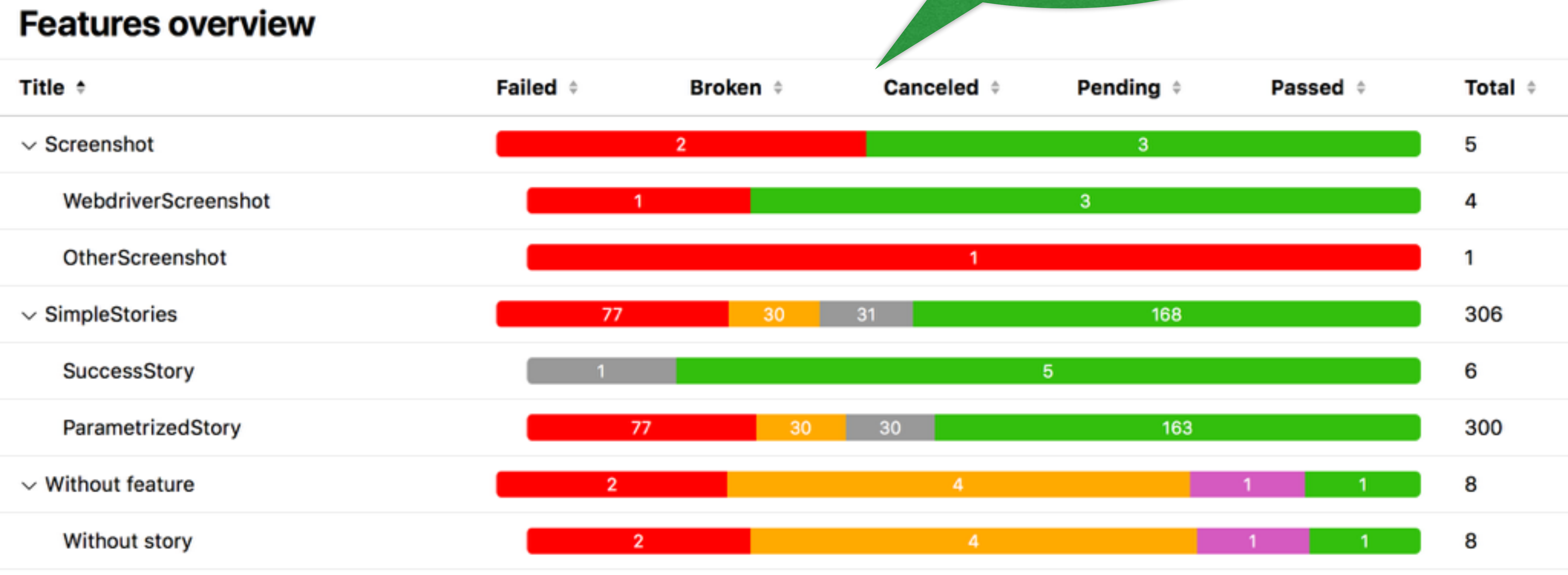
```
Test ID: TMS-2
Title: Test login
Requirements: JIRA-678
Priority: CRITICAL
Components: Authorisation
Tags: Regression, UI
Description: ...
```

Allure Framework



https://github.com/allure-framework/

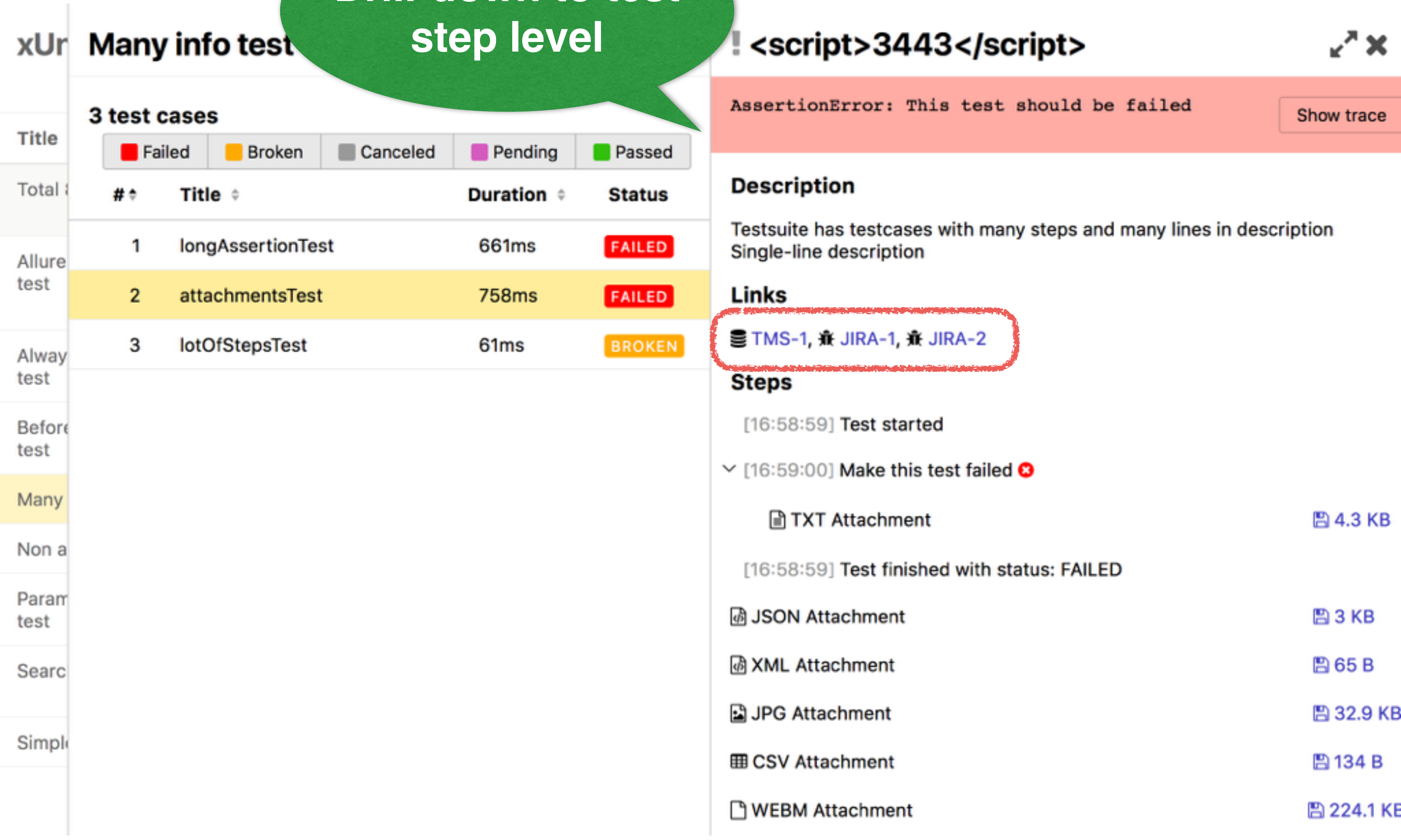
Feature coverage by tests



Features overview

Title	Failed	Broken	Canceled	Pending	Passed	Total
Screenshot	2				3	5
WebdriverScreenshot	1				3	4
OtherScreenshot	1					1
SimpleStories	77	30	31		168	306
SuccessStory			1		5	6
ParametrizedStory	77	30	30		163	300
Without feature	2	4		1	1	8
Without story	2	4		1	1	8

Drill-down to test step level



xUr

Title

Total

Allure test

Alway test

Before test

Many

Non a

Param test

Searc

Simpl

Many info test

3 test cases

🟥 Failed | 🟧 Broken | ■ Canceled | 🟪 Pending | 🟩 Passed

#	Title	Duration	Status
1	longAssertionTest	661ms	FAILED
2	attachmentsTest	758ms	FAILED
3	lotOfStepsTest	61ms	BROKEN

! <script>3443</script>

```
AssertionError: This test should be failed
```

Show trace

Description

Testsuite has testcases with many steps and many lines in description
Single-line description

Links

TMS-1, JIRA-1, JIRA-2

Steps

[16:58:59] Test started

[16:59:00] Make this test failed

TXT Attachment 4.3 KB

[16:58:59] Test finished with status: FAILED

JSON Attachment 3 KB

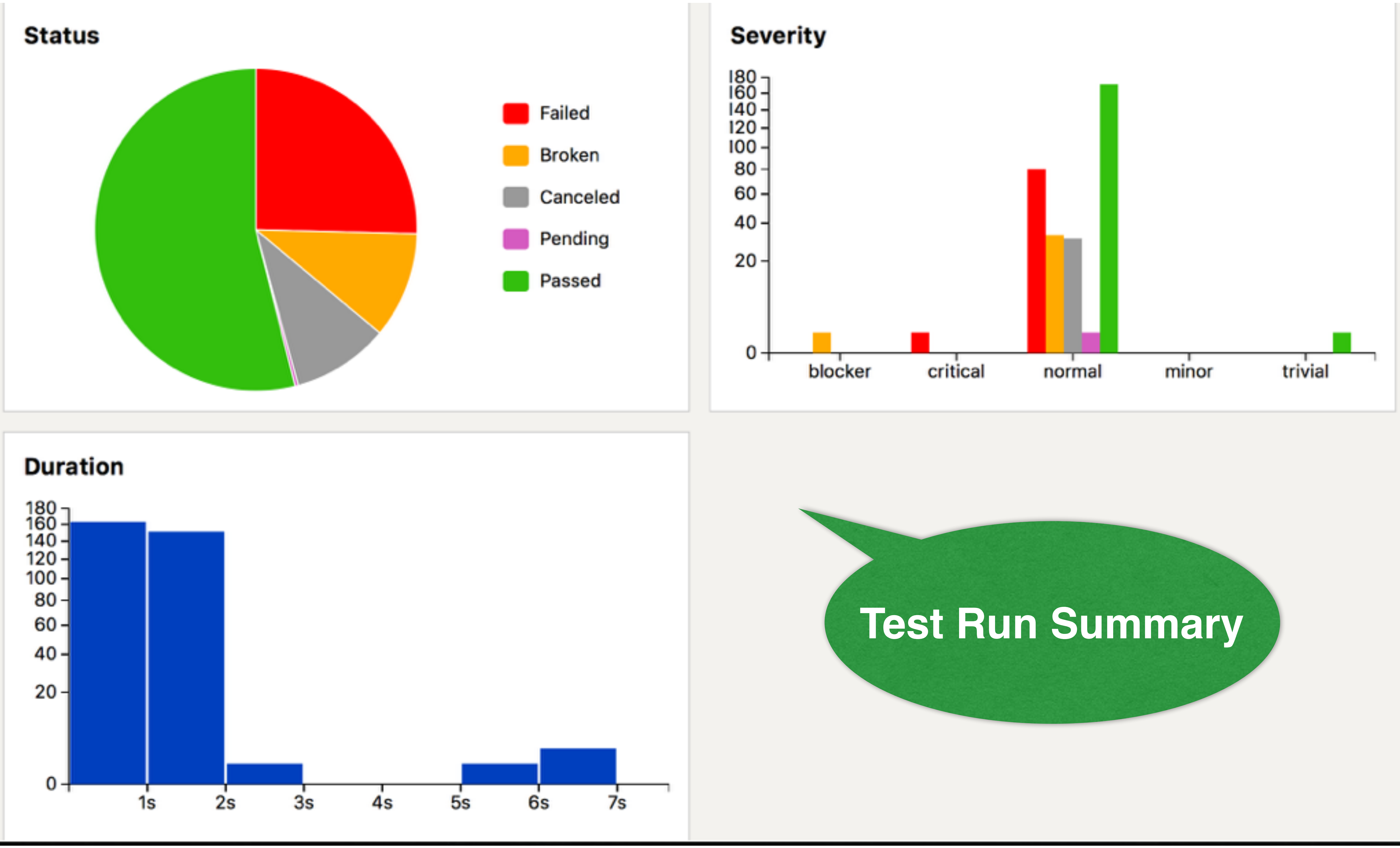
XML Attachment 65 B

JPG Attachment 32.9 KB

CSV Attachment 134 B

WEBM Attachment 224.1 KB

https://wiki.jenkins-ci.org/display/JENKINS/Allure+Plugin

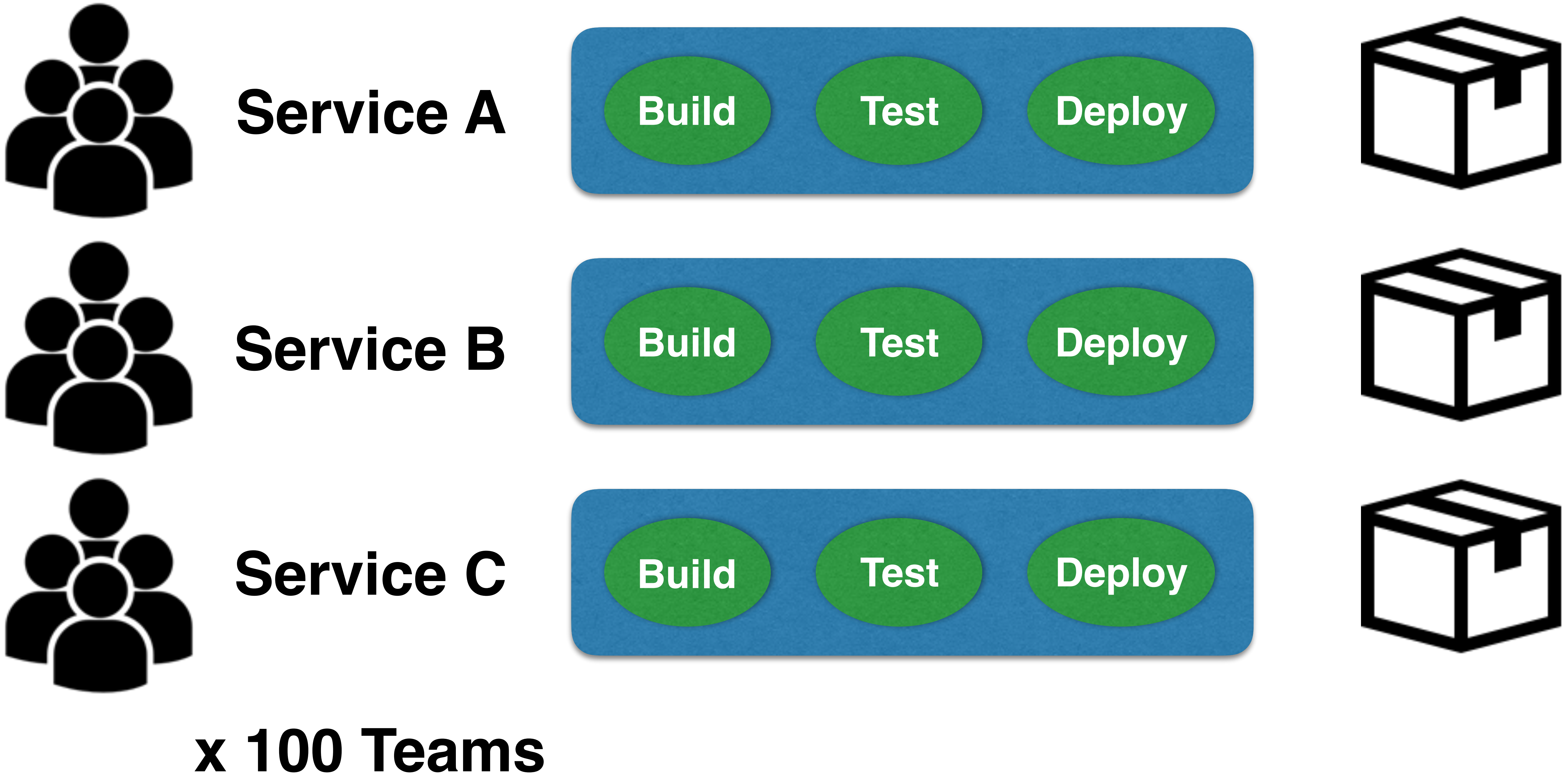


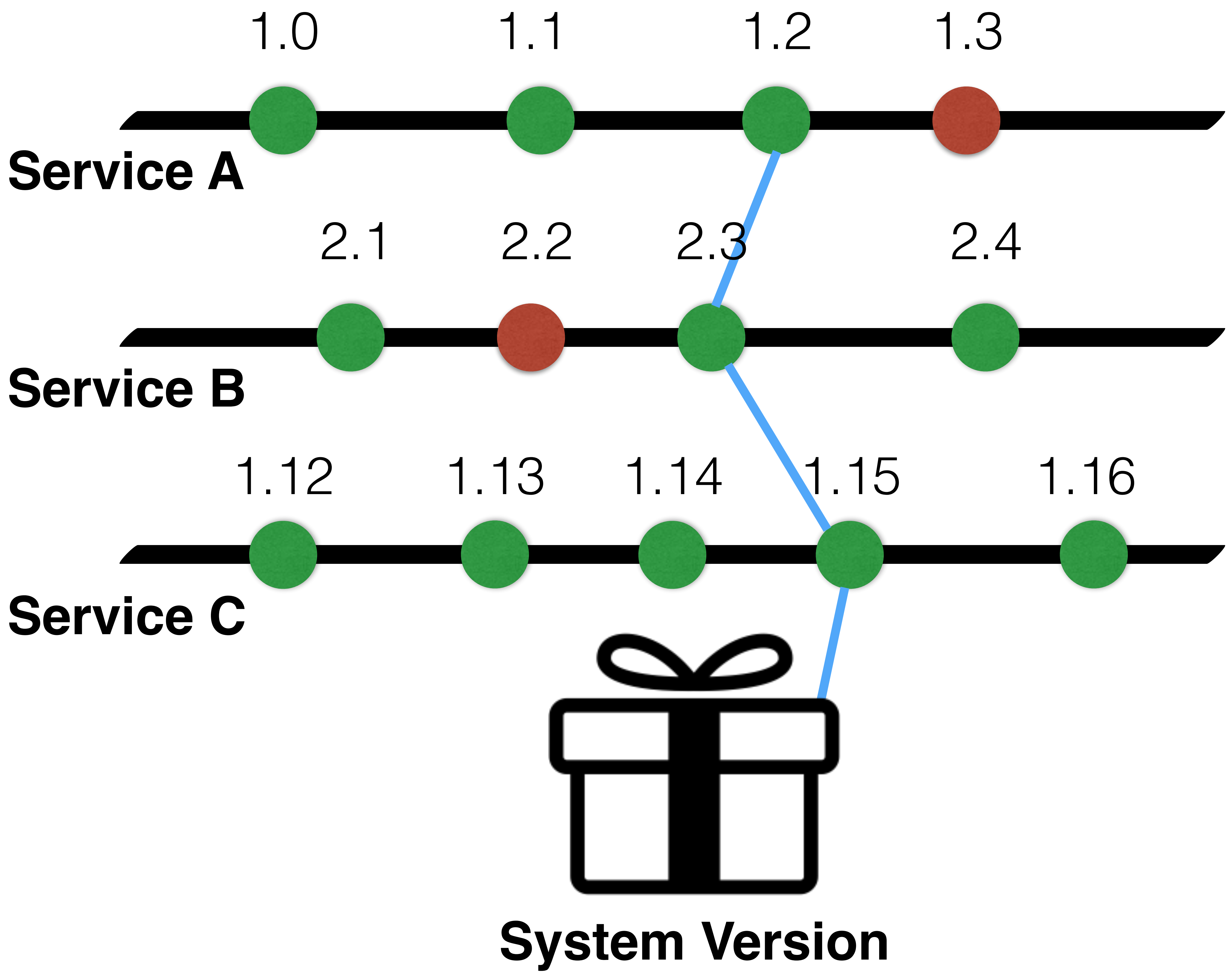
Test Run Summary

Achievements

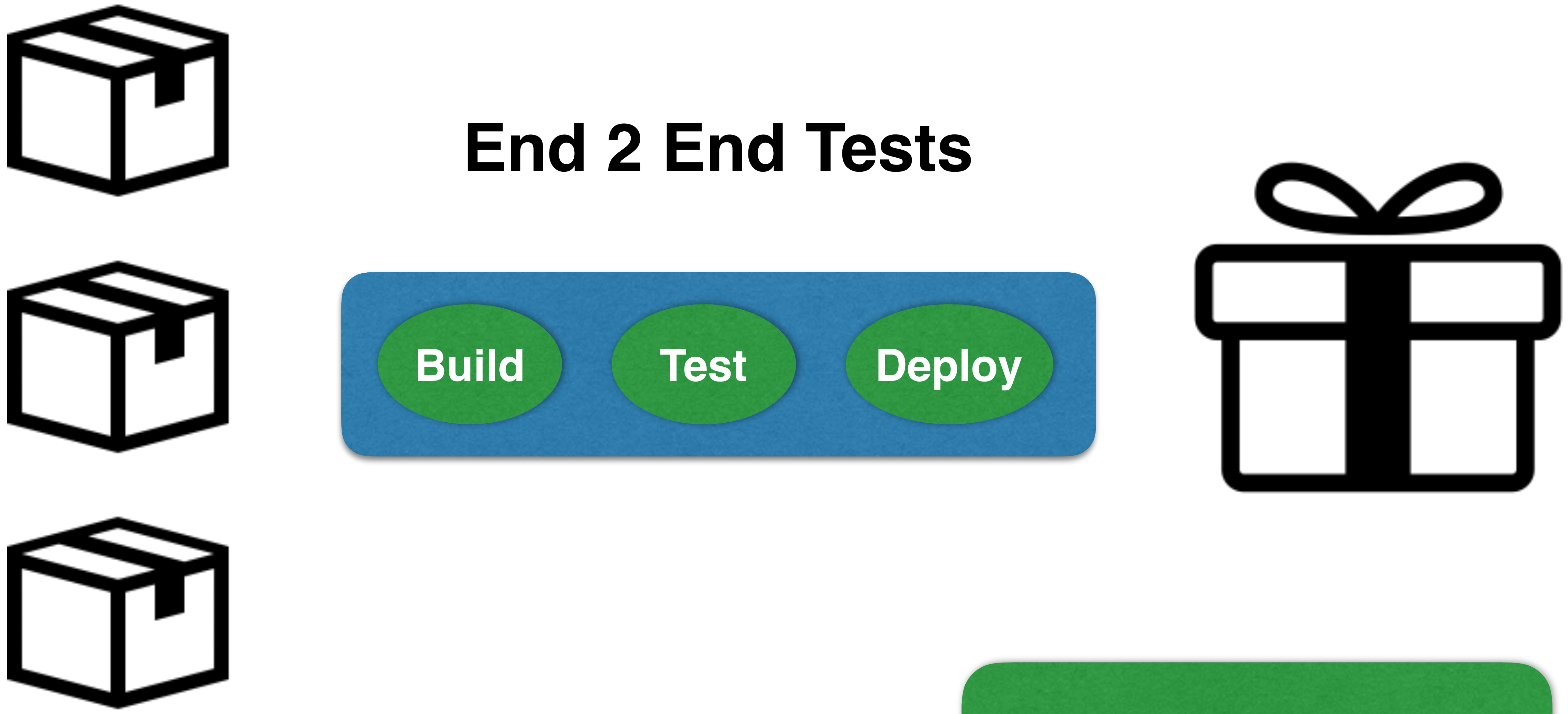
- Achieved a fully automated traceability in a single test run
- Possibility to extract and store test statistics
- Traceability is simple given right tools

CD Meets Microservices



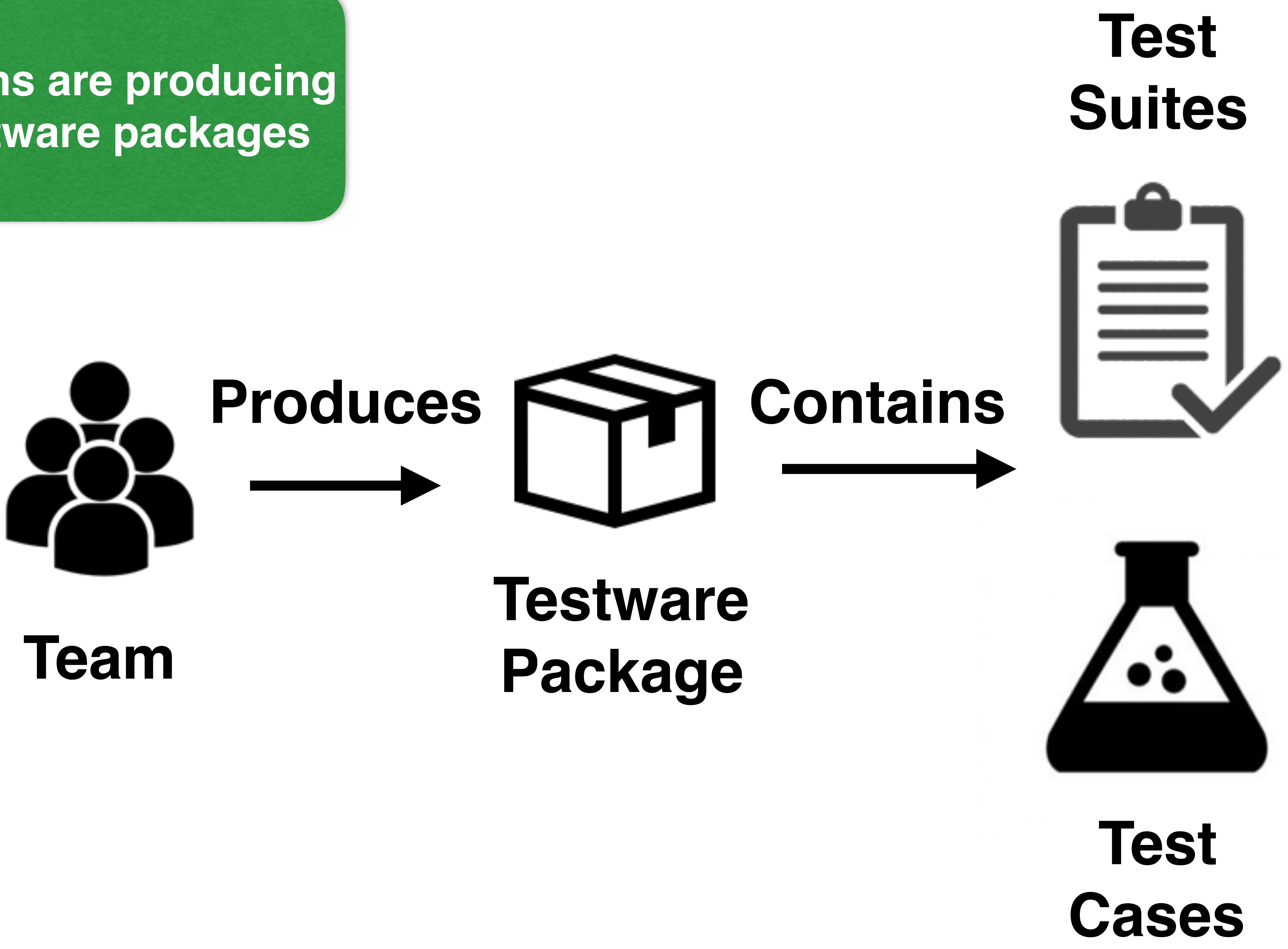

1.0
1.1
1.2
1.3
Service A
2.1
2.2
2.3
2.4
Service B
1.12
1.13
1.14
1.15
1.16
Service C
System Version

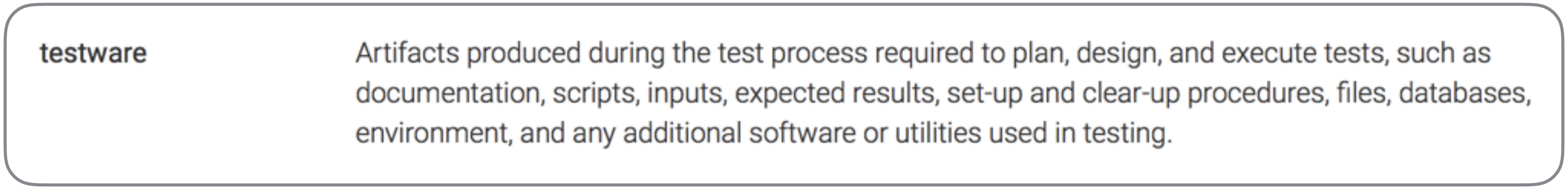
System Level Testing



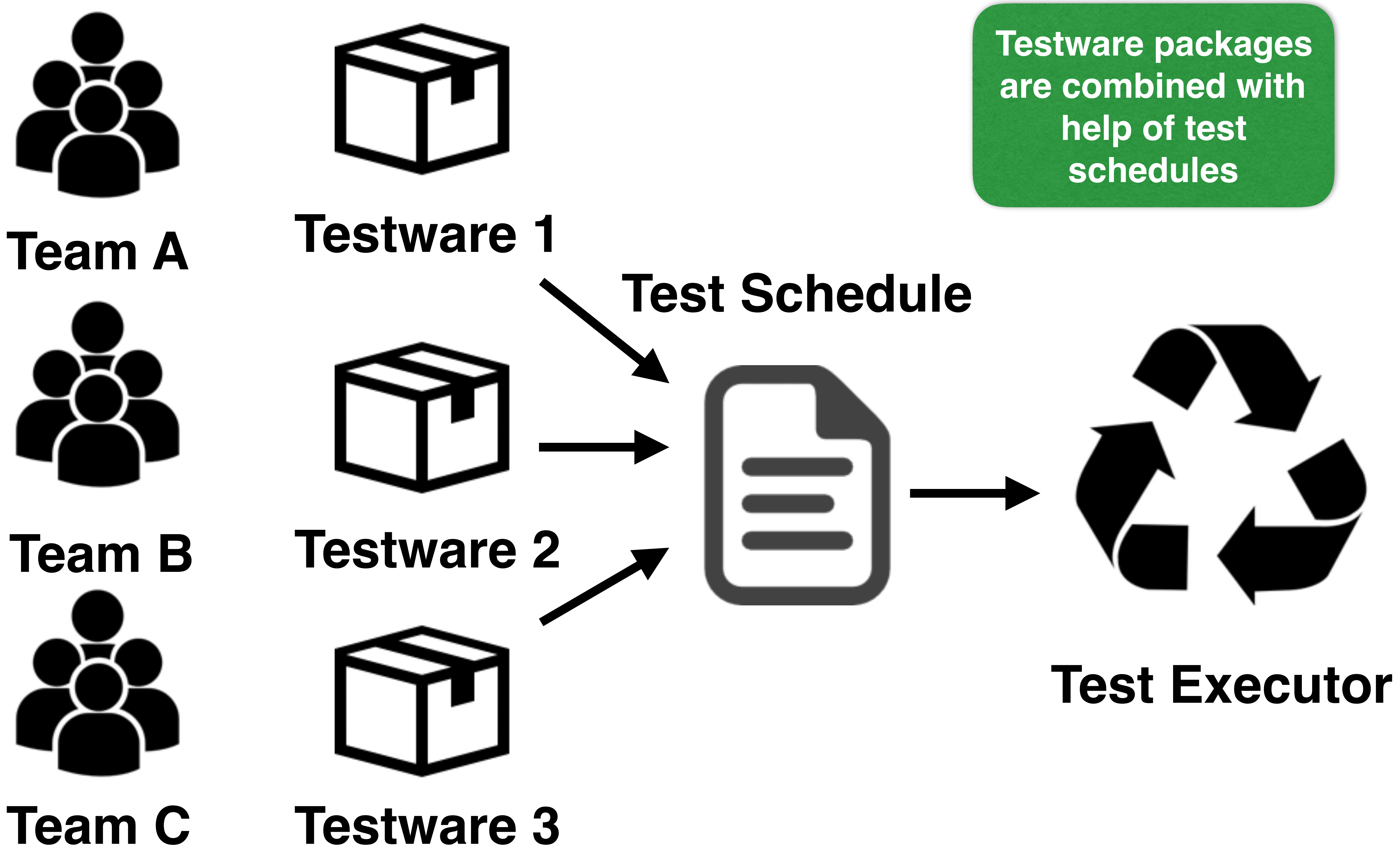
New package releases trigger E2E tests

Teams are producing testware packages



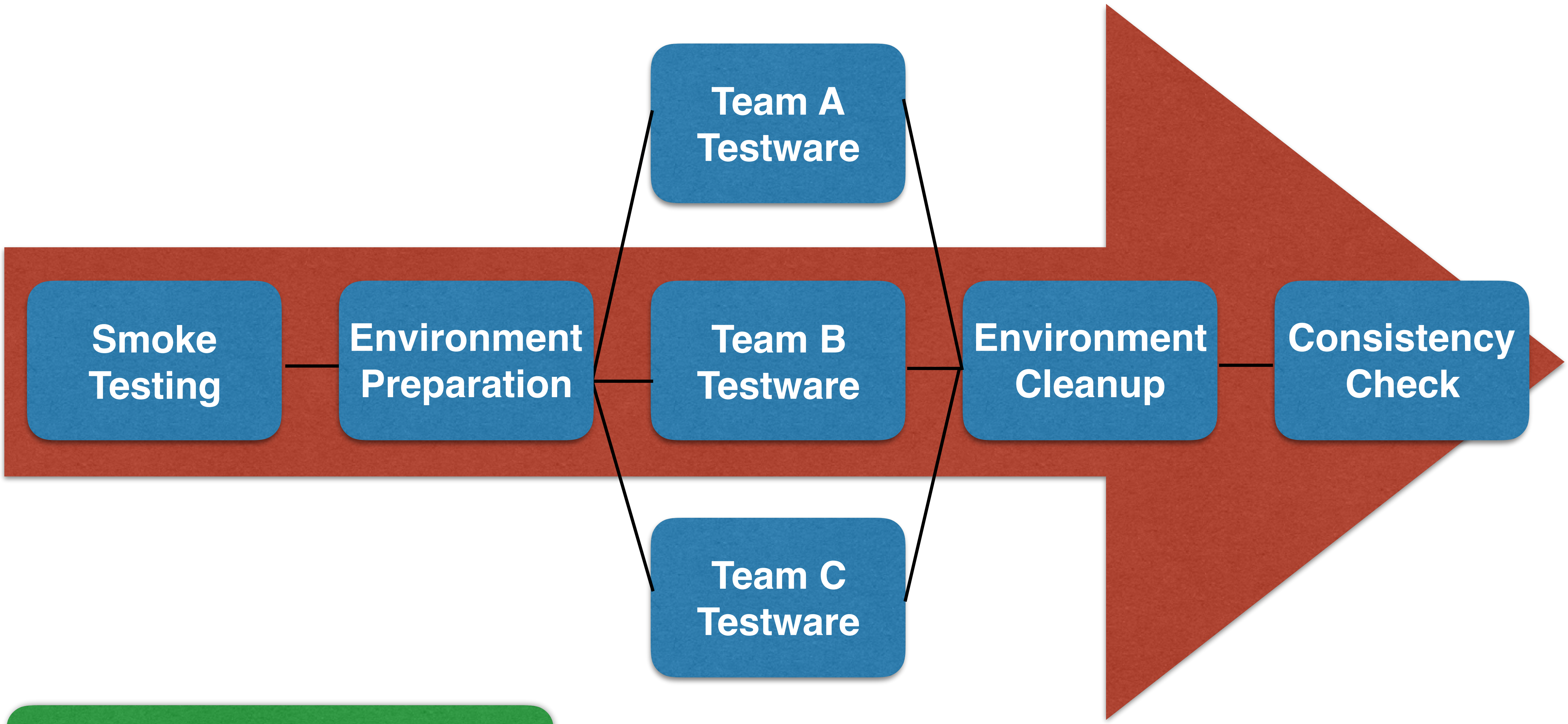


testware	Artifacts produced during the test process required to plan, design, and execute tests, such as documentation, scripts, inputs, expected results, set-up and clear-up procedures, files, databases, environment, and any additional software or utilities used in testing.



test schedule	A list of activities, tasks or events of the test process, identifying their intended start and finish dates and/or times, and interdependencies.

Test Schedule

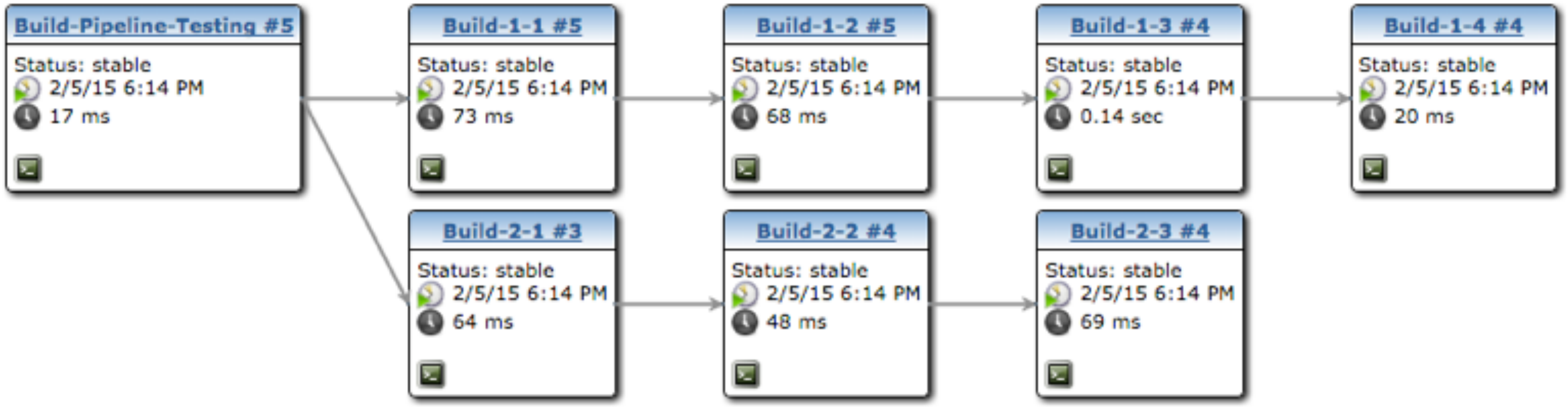


Schedule items are coming from different teams

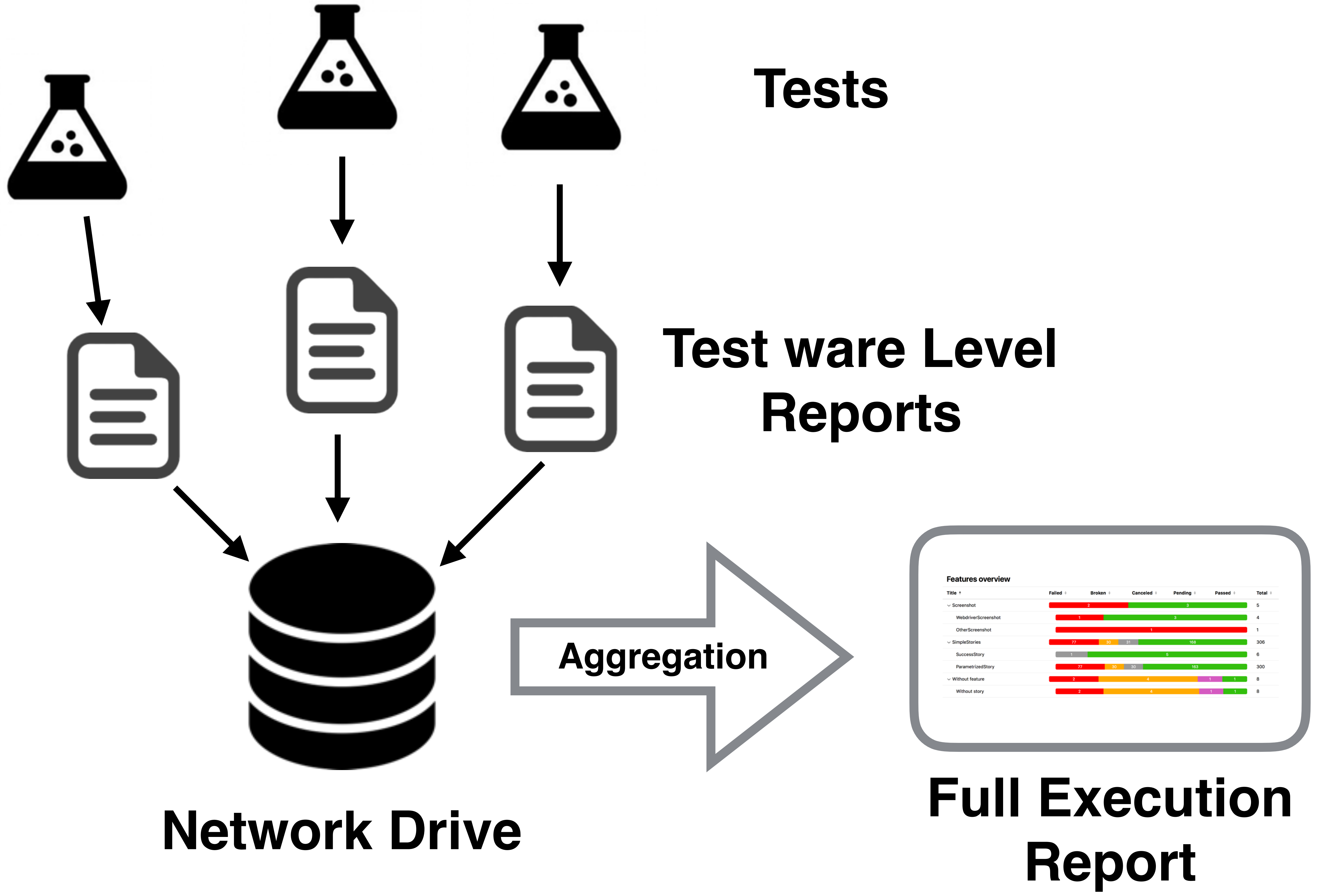
Tests schedule produces build flow definition, which is given to Jenkins build farm

```
parallel ({
        guard {
            build(“run-tests”, “testware1”)
            build(“run-tests”, “testware2”)
        } rescue {
            build(“run-tests”, “cleanup”)
        }
})
```

Visualisation



https://wiki.jenkins-ci.org/display/JENKINS/Build+Flow+Plugin



Features overview

Title	Failed	Broken	Canceled	Pending	Passed	Total
Screenshot	2				3	5
WebdriverScreenshot	1				3	4
OtherScreenshot	1					1
SimpleStories	77	30	31		168	306
SuccessStory			1		5	6
ParametrizedStory	77	30	30		163	300
Without feature	2	4		1	1	8
Without story	2	4		1	1	8

Achievements

- Tests from multiple teams are combined to a single test run
- Aggregated reports for all test cases are available

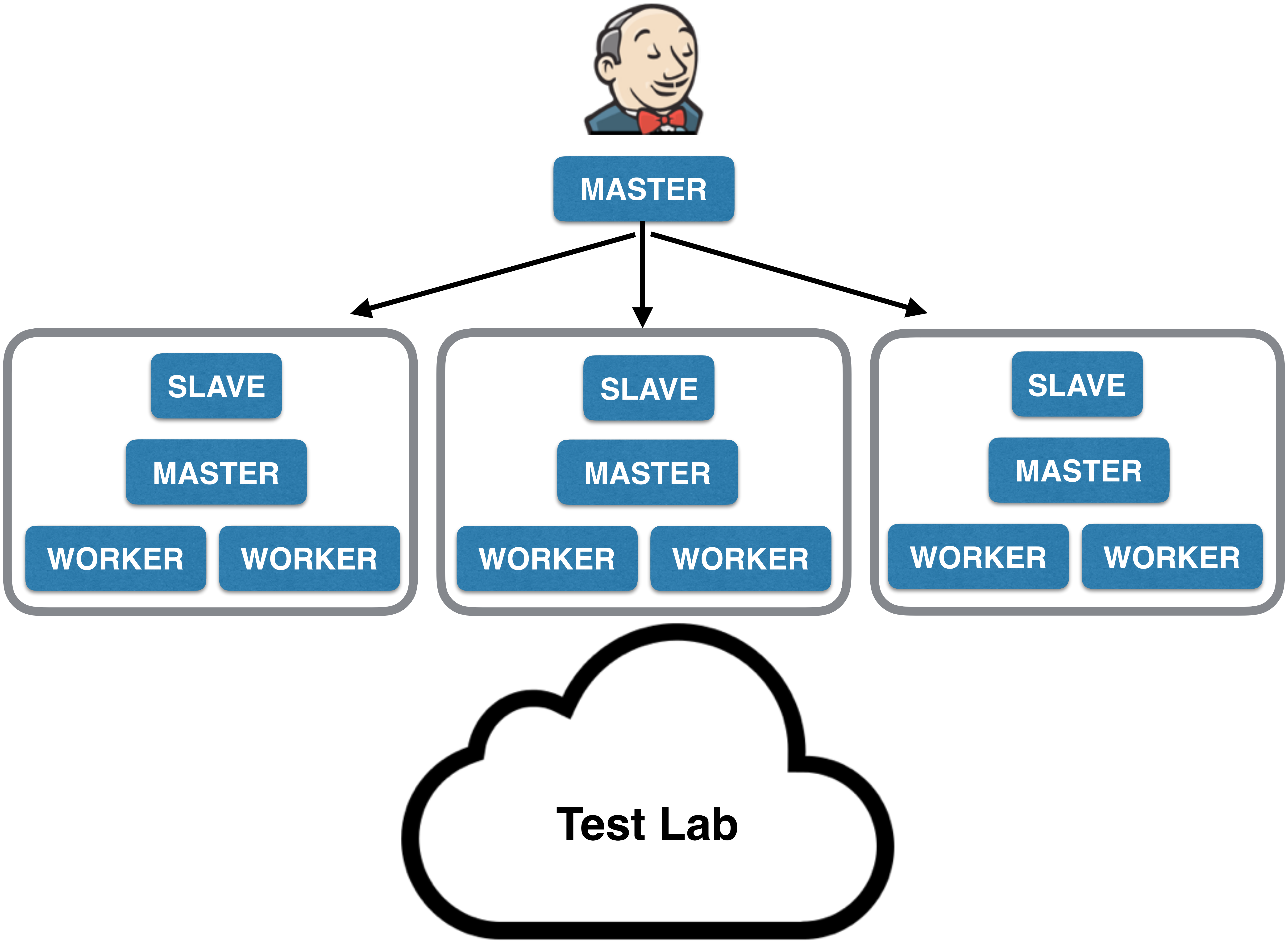
How to run tests as quickly as possible to minimise the feedback loop?

Tests Quality Police

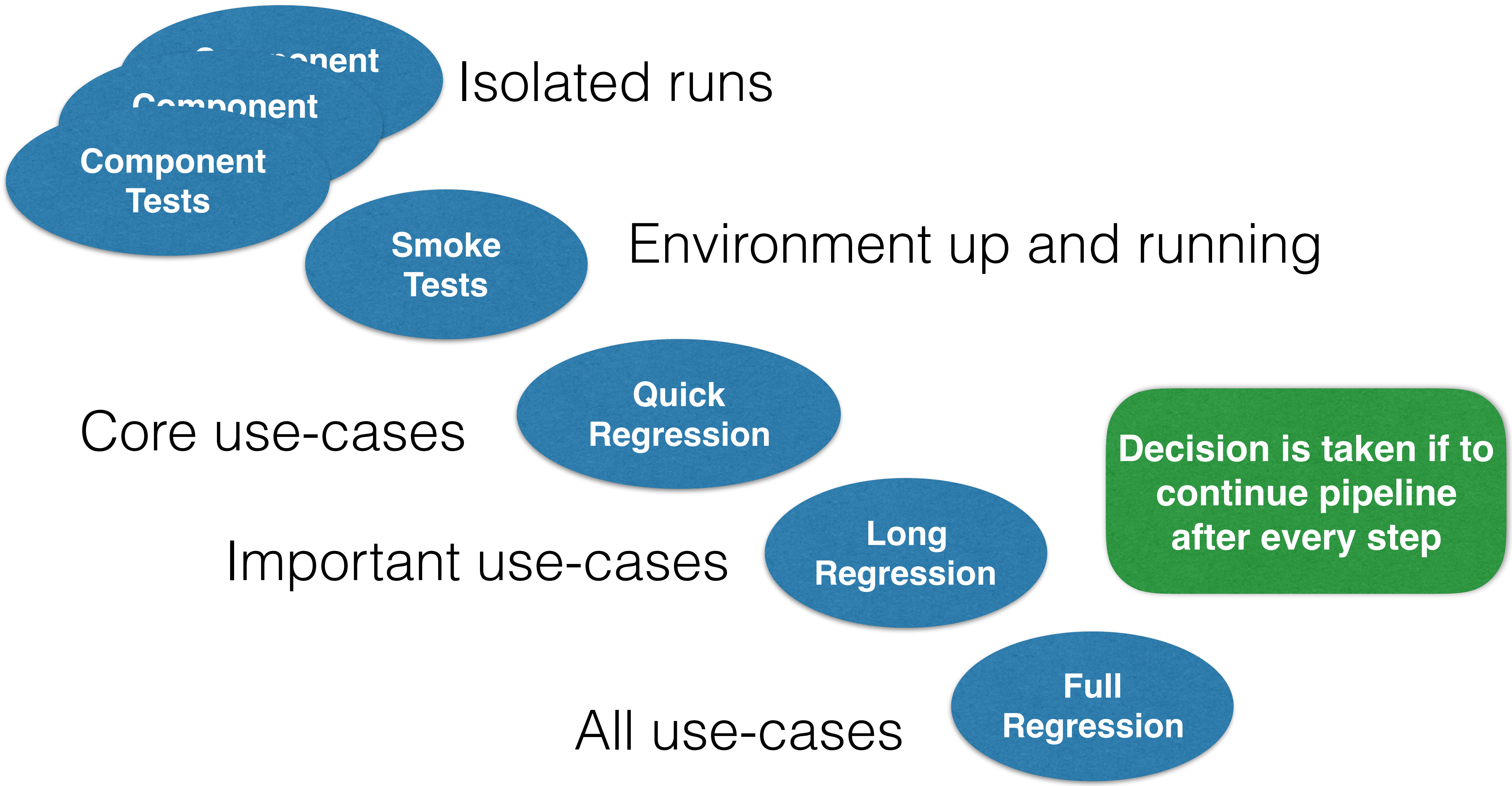
- Ensuring tests are
  - efficient (API vs UI, no waits)
  - can run in parallel
  - data-driven
  - reusing other teams test API

Distributed Test Executor

- Cloud based
- Scale-up/scale-down
- Health checks
- Parallelisation


MASTER
SLAVE
MASTER
WORKER
WORKER
SLAVE
MASTER
WORKER
WORKER
SLAVE
MASTER
WORKER
WORKER
Test Lab

Testing Flow



Achievements

- Tests quality is measured and improved
- Tests execution time is minimised

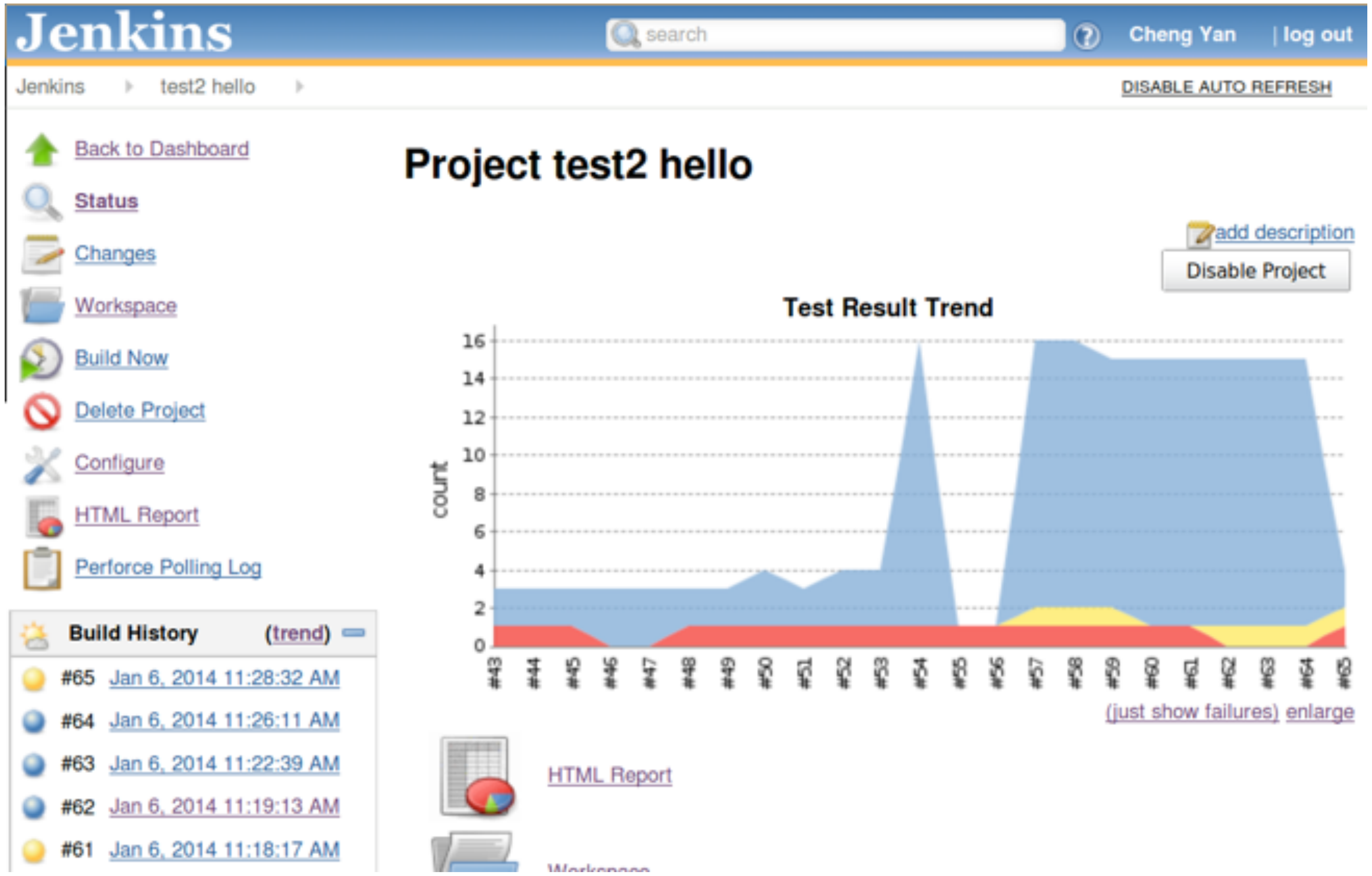
Failed test != 🔴

Sometimes

Interpreting Results

- 100% green tests pipeline all the time - not realistic at this scale
- what is failed test priority?
- is it a new test?
- is it a flaky test?
- was it rerun several times?

Test Result Analytics

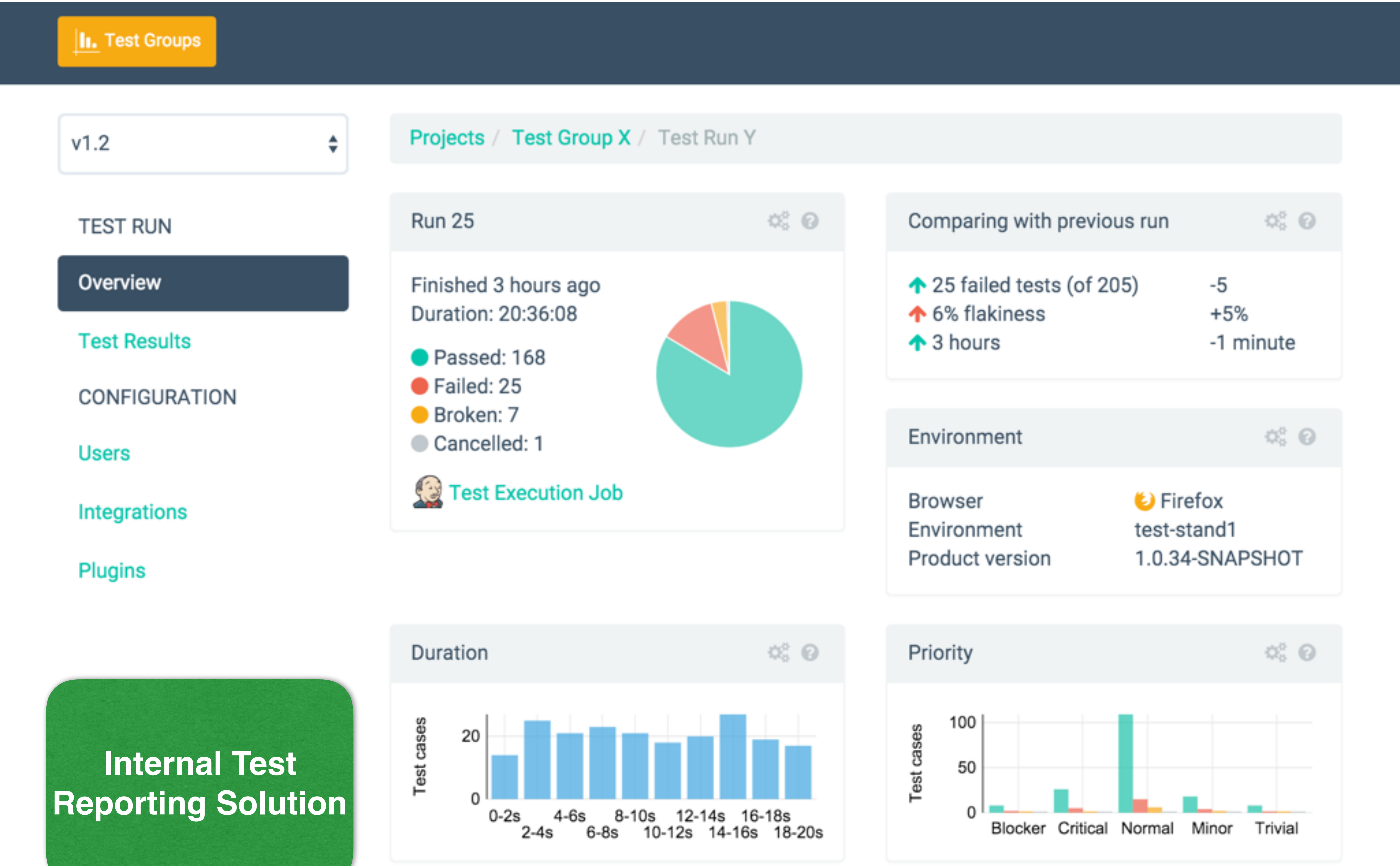


Project test2 hello

Build History (trend)

#65 Jan 6, 2014 11:28:32 AM
#64 Jan 6, 2014 11:26:11 AM
#63 Jan 6, 2014 11:22:39 AM
#62 Jan 6, 2014 11:19:13 AM
#61 Jan 6, 2014 11:18:17 AM

Need something like this, but with more information



Test Groups

v1.2

TEST RUN

Overview

Test Results

CONFIGURATION

Users

Integrations

Plugins

Projects / Test Group X / Test Run Y

Run 25

Finished 3 hours ago
Duration: 20:36:08

● Passed: 168
● Failed: 25
● Broken: 7
● Cancelled: 1

Test Execution Job

Comparing with previous run

↑ 25 failed tests (of 205)	-5
↑ 6% flakiness	+5%
↑ 3 hours	-1 minute

Environment

Browser	Firefox
Environment	test-stand1
Product version	1.0.34-SNAPSHOT

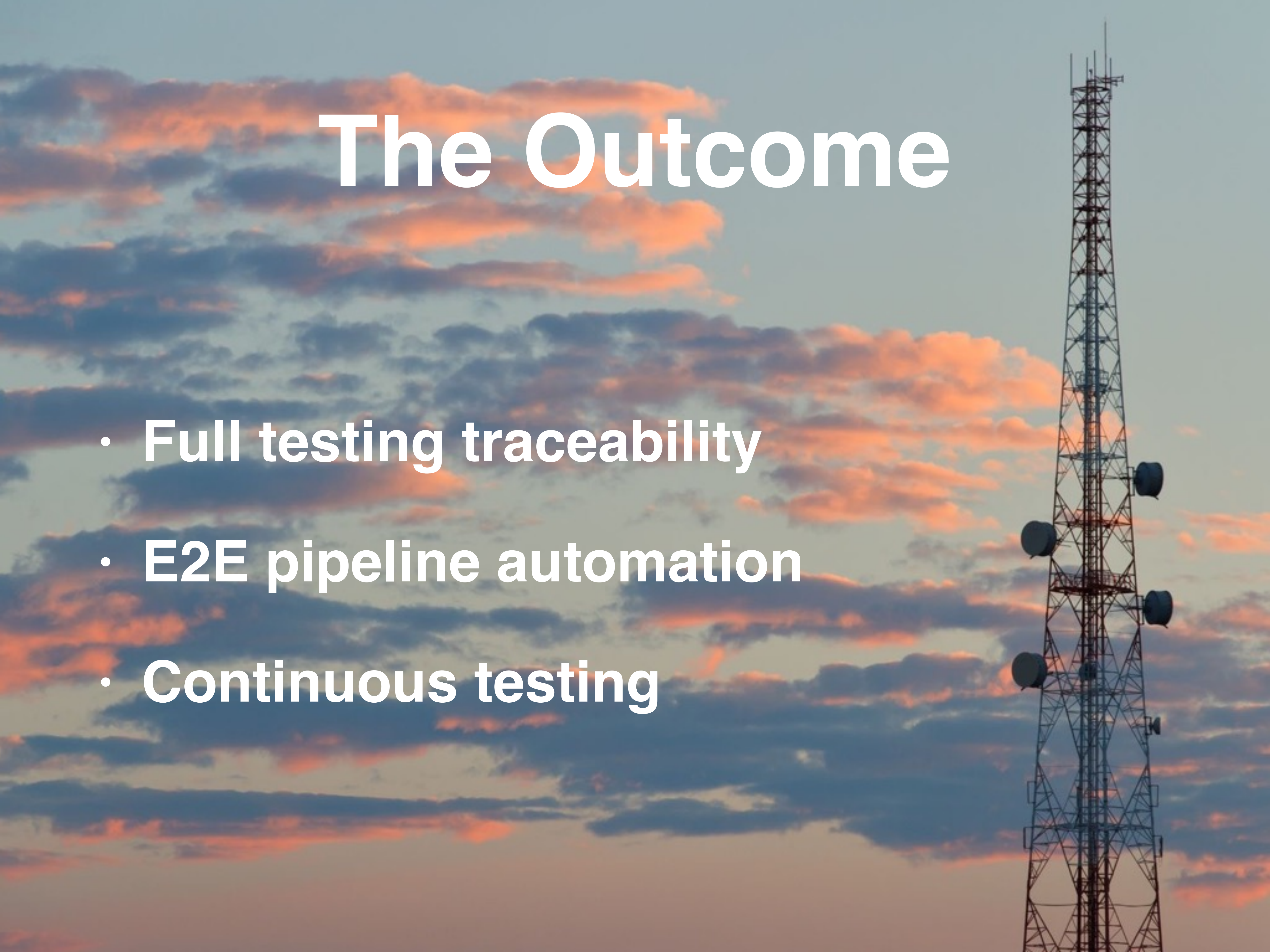
Duration

Priority

Internal Test Reporting Solution

Achievements

- Identifying good enough builds
- Ignoring random faults
- Historical test results for analysis



The Outcome

- Full testing traceability
- E2E pipeline automation
- Continuous testing

Lessons Learned

There are not too many tools out there to support large scale continuous testing

Instead of using monolithic tools - aim for extensible independent services and interoperability standards

CD Standards

- Some well-known standards and platforms:
  - Jenkins - Pipelenes
  - Docker Containers
  - JUnit 5 - Test Runner
  - WebDriver - Selenium HTTP API
  - Allure Reporting - Common for all test tools

Next 5 years are going to be about CD and Testing tools interoperability

This slide was intentionally left blank