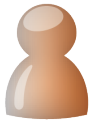



Testing a Large Support Matrix Using Jenkins

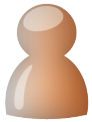
Amir Kibbar
HP

http://hp.com/go/oo



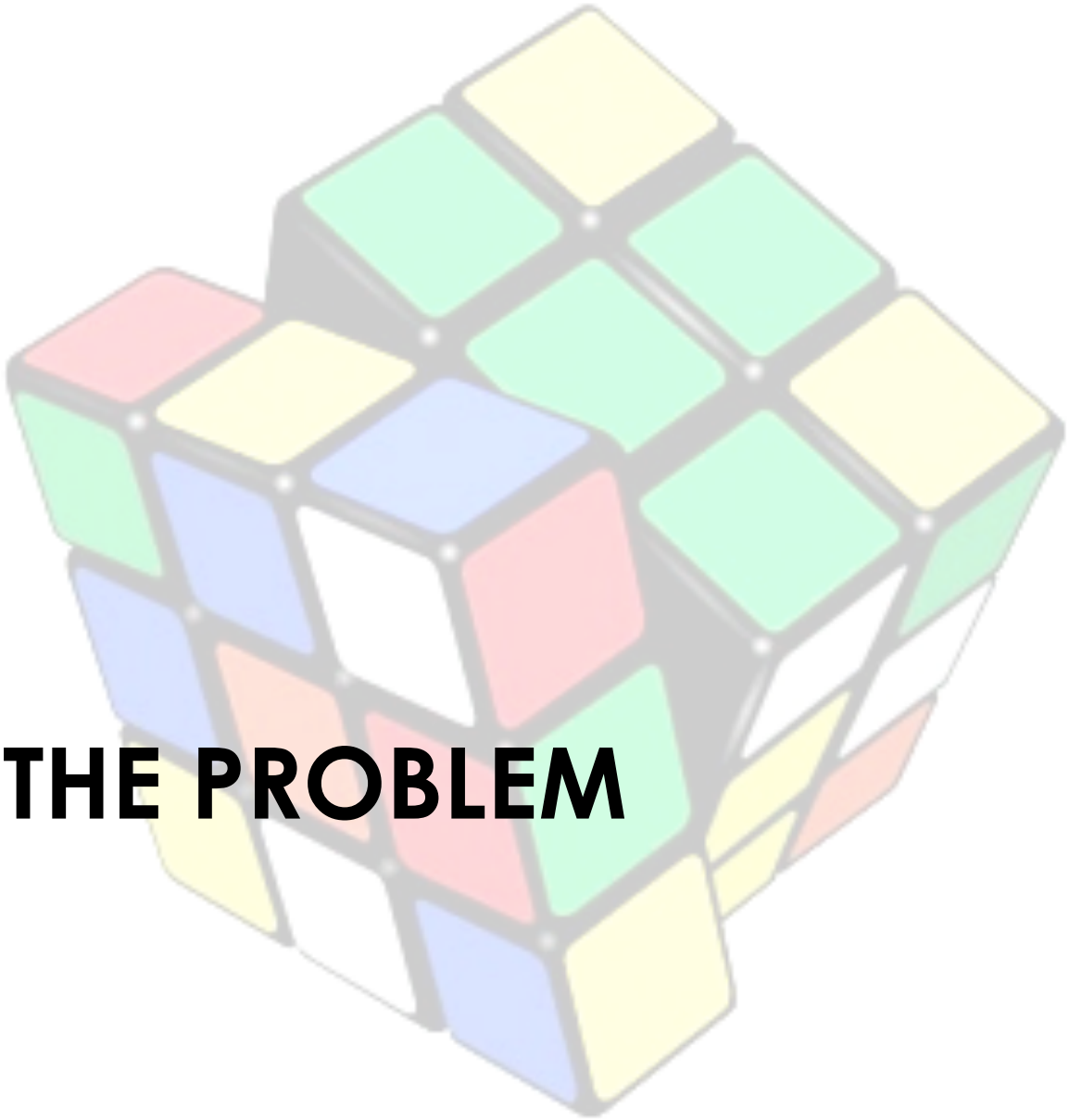
About Me

- 4.5 years with HP
- Almost 3 years System Architect
- Out of which 1.5 HP OO's SA
- Before that a Java consultant for 4 years
- Few start up companies
- IAF

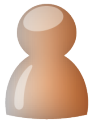


HP Operations Orchestration

- An ITPA (IT Process Automation) tool
- Intuitive automation of IT tasks and processes
- Execute workflows
- Lots of OOTB content – building blocks for workflows
- Extensible, pluggable and embeddable



THE PROBLEM



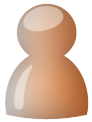
What Types of Bugs Are We Addressing?

Functional regressions

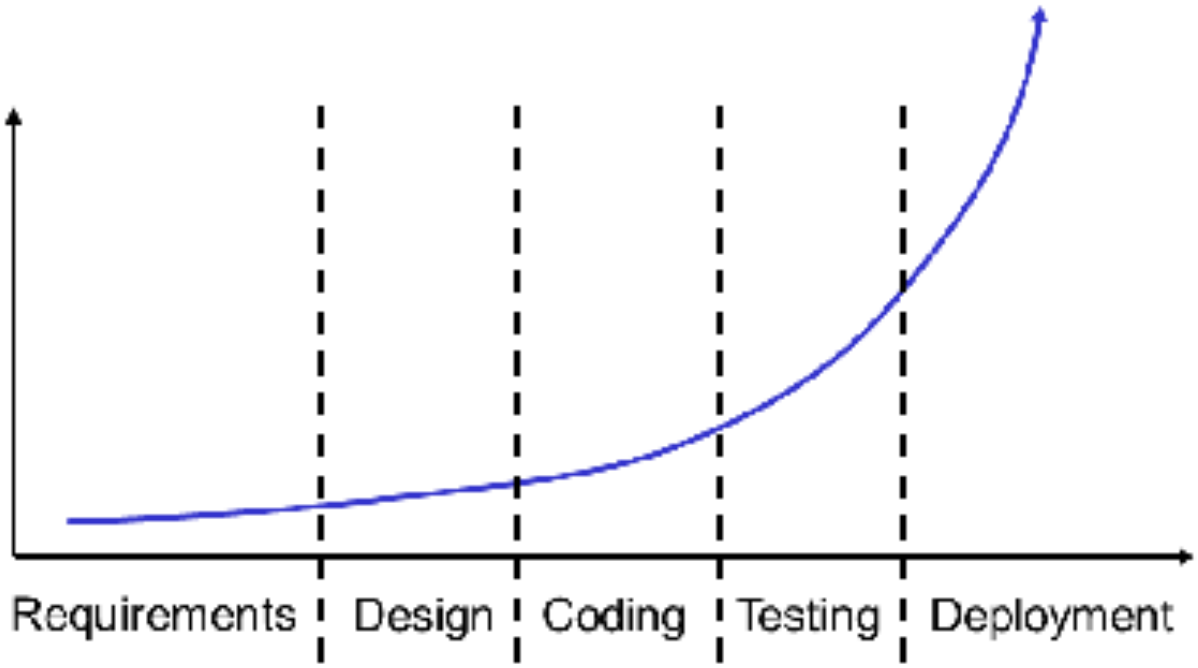
Database incompatibility

Operating system incompatibility

Performance degradation



Late Discovery = $$$$



A Large Support Matrix

7 Server Operating Systems	RHEL 5
	RHEL 6
	OEL 5
	OEL 6
	Win 2008
	Win 2008 Japanese
	Win 2008R2
6 Client Operating Systems	Win 7 32 bit
	Win 7 64 bit
	Win Vista 32 bit
	Win Vista 64 bit
	Win 2008 64 bit
	Win 2008 R2 64 bit
6 Databases	SQLServer 2008
	SQLServer 2008r2
	SQLServer 2012
	MySQL 5.5.x
	PostgreSQL 9.1.x
	Oracle 11gR2
5 Browsers	IE 8
	IE 9
	FF
	Chrome
	Safari
5 Application Servers	Tomcat 7.x
	JBoss 6.x
	JBoss 7.x
	Jetty 7.x
	Glassfish 3.1

7 Server OSs x
6 Client OSs x
6 DBs x
5 Browsers x
5 Application Server

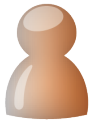
= **6300** combinations

7 Server OSs x 6 DBs +
(5 Browsers x 6 Server OSs x 6 Client OSs) +
(4 Application Server x 2 OSs x 2 DBs)

= **259** combinations

7 Server OSs x 6 DBs +
(5 Browsers x 6 Client OSs) +
(4 Application Servers x 2 OSs x 2 DBs)

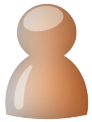
= **88** combinations



But We Also Want

- A continuous process
- Standalone build (can be built without any external services such as a database, application server, etc.)
- Measureable test coverage
- Measureable performance metrics

UNDERSTANDING OO



Our Technologies

Java 1.7

Any servlet 2.5 application server

- Tomcat 7.x
- JBoss 6.x, 7.x
- Glassfish 3.x
- Jetty 7.x

Spring-*

- Spring web MVC
- Spring core
- Spring data JPA
- Spring integration

Hibernate

- 4.1.x
- JPA (Entity Manager)

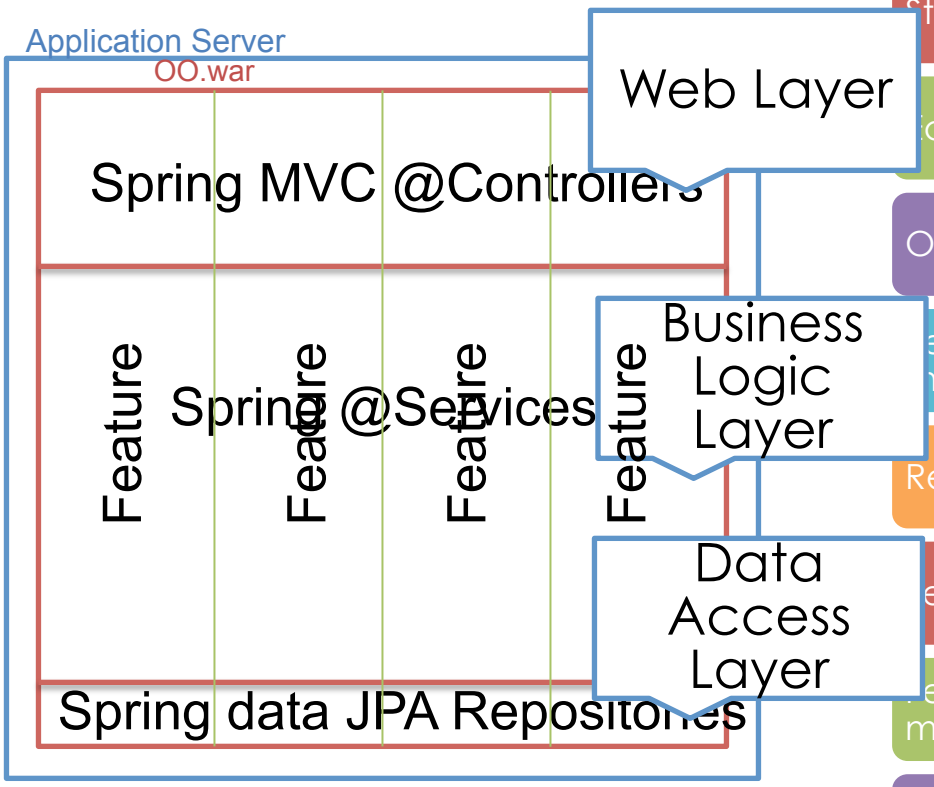
GWT

- Client only – no GWT-RPC

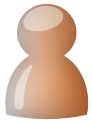
RESTful APIs

- REST everything

(Very) High Level Architecture



- Standard layered architecture
- Each layer talks to the one below it
- One controller cannot call another
- Services can activate one another, but only through interfaces (auto-wired)
- Repositories are interfaces only
- Feature-based architecture
- Features are separated jars. Each feature may have a client, API and impl jar
- All access via REST



Unit Tests – Mocks!!!!

No direct tests to repositories

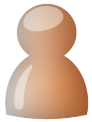
When testing a service we mock all other services

When testing a service we sometimes mock the repositories, but sometimes use in-memory H2

When testing controllers we mock the services

Controllers tests try to simulate even the unlikely scenarios

Client (GWT) tests test the “client logic”



Unit Tests Technologies

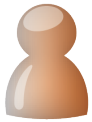
JUnit

Spring test

Mockito

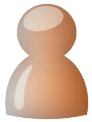
Jukito

H2 (in-memory)



OK, So?

- Unit tests are fast
- Targeted – thanks to the feature-based architecture
- Contribute to regression verification
- Help reduce refactoring risks

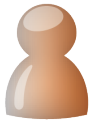


System (Integration) Tests – The Real Deal

Black box tests

API

UI



System Tests Mechanics

In-build tomcat

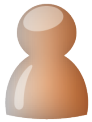
Tests cannot access the DB directly

Tests cannot access the server filesystem

Tests can only do what a potential customer can

API tests use only the RESTful APIs

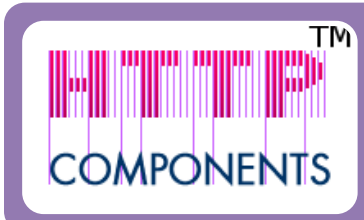
UI tests use the RESTful APIs to stage the UI test and perform as little UI actions as possible



System Tests Technologies

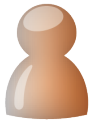
maven-t7-plugin (Tomcat)

JUnit



Apache HttpClient

Selenium

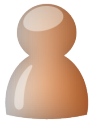


Ok, So?

Real scenarios

Real DB (still in-memory H2 though)

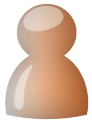
Continuous process



What's Missing?

The support matrix

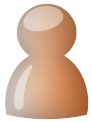
THE SOLUTION



Target Support Matrix - Reminder

$$(7\downarrow OS \times 6\downarrow DB) + (5\downarrow browser \times 6\downarrow OS) + (4\downarrow Application\ Server \times 2\downarrow OS \times 2\downarrow DB) = 88\downarrow co$$

Server-side Client-side Embedded

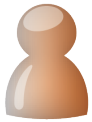


How Does it Work?

The installers build	Compile only – create the installers
	~10 minutes
The deployment matrix	Install the application
	6 OS x 5 DB = 30 instances
	6 VMs
	~10 minutes
The test suite	Run all system tests including Selenium tests which use a browser
	7 VMs with different browsers installed on them
	~25 minutes

$$(6\downarrow OS \times 5\downarrow DB) + (2\downarrow OS \times 4\downarrow browser) = 38\downarrow combinations$$

$$\approx 43\% \ support\ matrix\ coverage$$

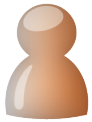


The Installers Build

Creates installation package

Platform specific

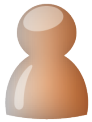
Zip



The Deployment Matrix

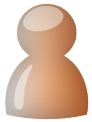
- A Jenkins configuration job
- Slaves determine the OS
- A parameter determines the DB
- Runs on 6 slaves, each one with 6 instances of the application
- Ant script – uninstall, download latest, unzip, post install (create DB properties)

Configuration Matrix	oracle11r2	postgres	mysql	mssql2008	mssql2008r2
mtx-linux-OEL5-64					
mtx-linux-OEL6-64					
mtx-linux-RHEL5-64					
mtx-linux-RHEL6-64					
mtx-windows-2008-64					
mtx-windows-2008-64-JPN					

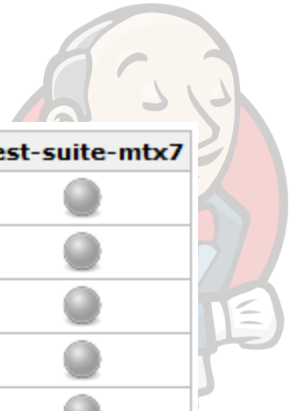


The Test Suite

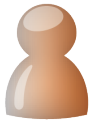
- A Jenkins configuration job
- Slaves determine the OS
- The browser is pre-determined
- A parameter determines the target server instance
- mvn install on the system tests project
- The system tests are “out-of-build” – outside the continuous build



The Test Suite

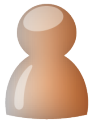


Configuration Matrix	test-suite-mtx1	test-suite-mtx2	test-suite-mtx4	test-suite-mtx5	test-suite-mtx6	test-suite-mtx7
w200864my						
w200864ms8r2						
w200864-2ms8						
w200864-2o112						
rhel564my						
rhel564ms8r2						
rhel564-2p						
rhel564-2ms8						
rhel664my						
rhel664ms8r2						
rhel664-2p						
rhel664-2o112						
oel564my						
oel564ms8r2						
oel564-2p						
oel564-2o112						
oel664ms8r2						
oel664ms8						
oel664-2p						
oel664-2o112						



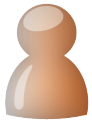
In-Build vs. Out-of-Build System Tests

Item	In-Build	Out-of-Build
Application server	Tomcat	Determined by server instance
Location of AS	Always localhost	Some server
how does the client locate the server?	Always localhost	Resolves the hostname using a mapping file according to a parameter
Who starts the AS	Maven (t7mp)	Service/daemon
Dependencies	The web application (war)	None



What's Left?

- Continuous process
- Standalone build
- Operating systems
- Databases
- Browsers
- Functional regressions
- Application servers
- Measureable performance
- Measureable test coverage



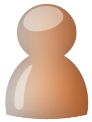
Secondary Matrices

Application Servers

- Deployment matrix
- Test suite

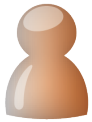
Performance

- Deployment matrix
- Test suite



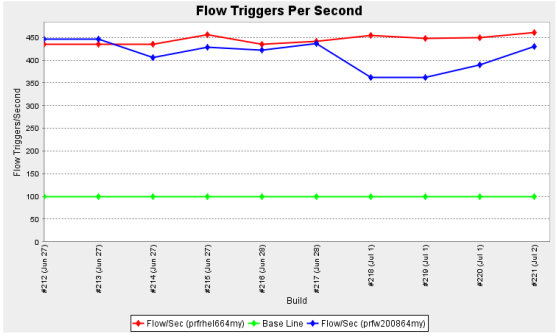
The Application Server Matrix

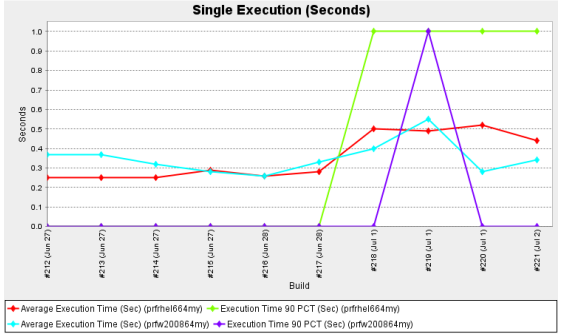
- Identical to the main matrix pair: deployment + test suite
- The deployment script re-deploys the application into an existing AS, does not re-install product
- A parameter determines the AS type (glassfish, JBoss or Jetty)
- Less operating systems x DB combinations – according to the host products

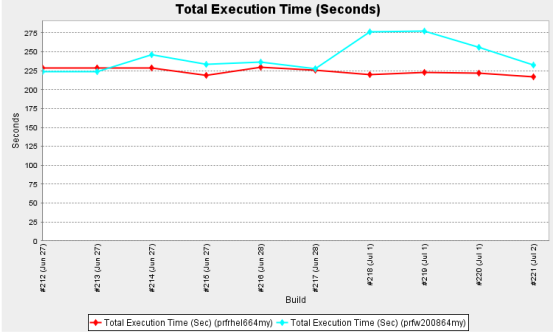


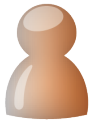
The Performance Matrix

- 3 jobs: deployment + test suite + summary
- The deployment and test suite are identical to the main matrix
- Runs different tests – performance specific tests which produce KPIs in a CSV
- The summary job plots the KPIs





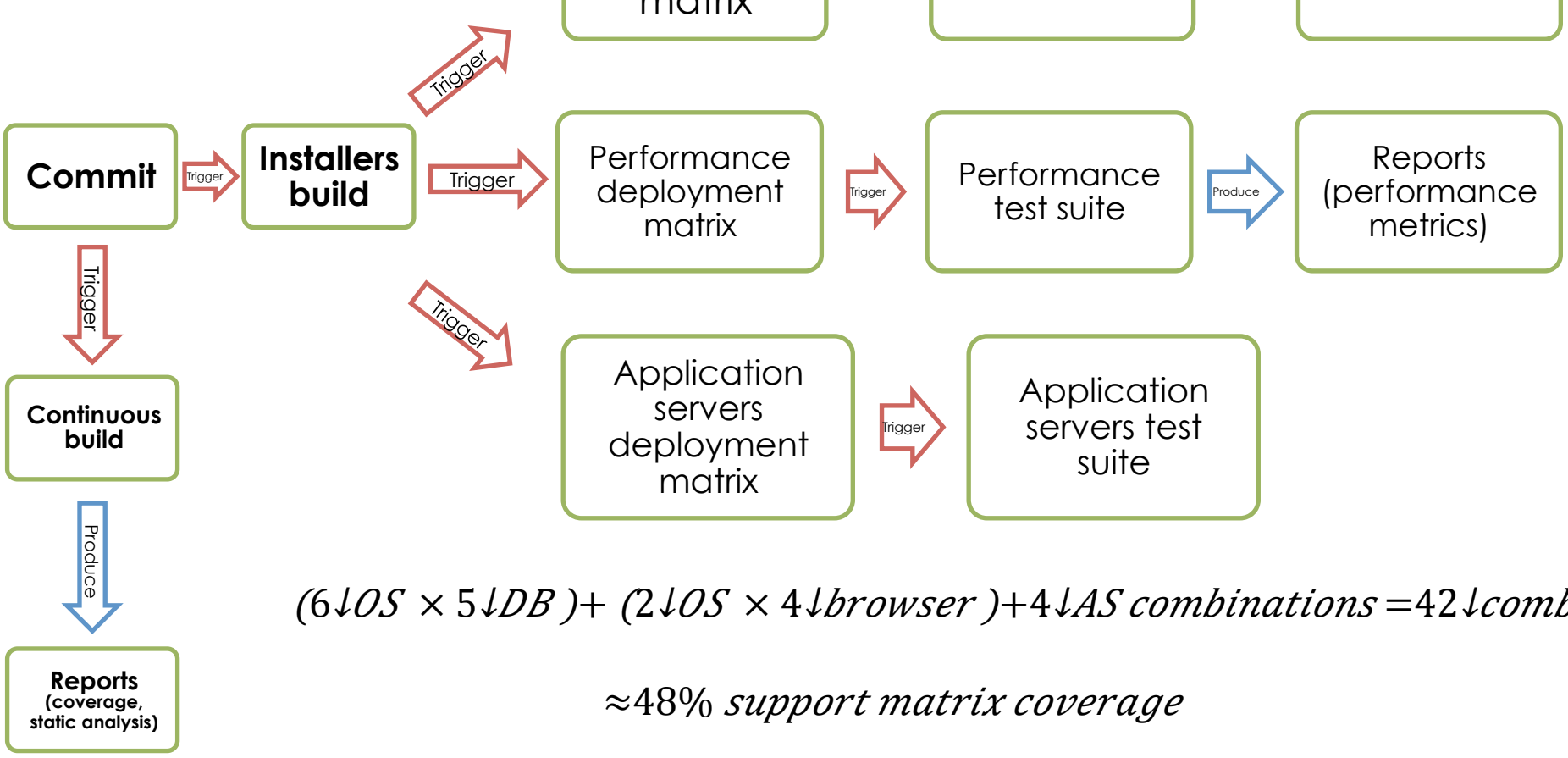




- Cobertura
- Calculated in the continuous build
- Single number – unit tests + system tests
- Classes that are unreachable in any test are filtered from the 100% (e.g.: view related classes)

ALL TOGETHER NOW

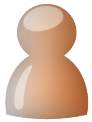
The Entire Process



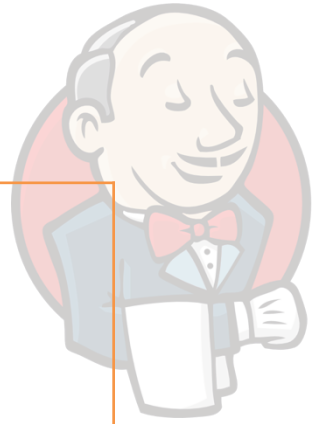
$$(6\downarrow OS \times 5\downarrow DB) + (2\downarrow OS \times 4\downarrow browser) + 4\downarrow AS\ combinations = 42\downarrow comb$$

$$\approx 48\%\ support\ matrix\ coverage$$

QUESTIONS



Thank You To Our Sponsors



Platinum Sponsors	CloudBees JFrog
Silver Sponsor	alphacsp