Software technology

06 – DevOps Toolchain

System Architecture

Lecture: Zoltán GERA / Practice: László GRAD-GYENGE

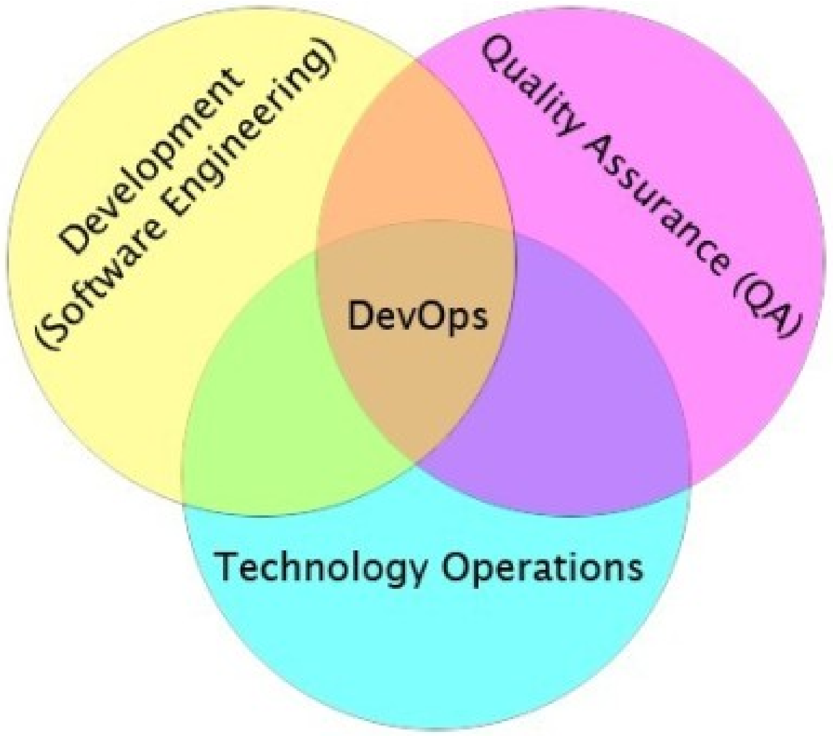
Editor & Presenter: Dr. Attila GLUDOVÁTZ

Online catalog – every week

- https://catalog.inf.elte.hu/
- Log in
- Username: yourUsername (@inf.elte.hu)
- Password: your email password
- Captcha: I generate a number for you…
- Lecture attendance is ***not*** optional! Max 3 misses and you are out

DevOps
Development & Operations

- *Collaboration of developers and other IT professionals to automate software delivery and infrastructure changes*
- Traditionally:
  - Developers change
  - Testers reduce risk
  - Operations stabilize processes
- Contradicting goals + Agile methodologies → DevOps is **cultural change**

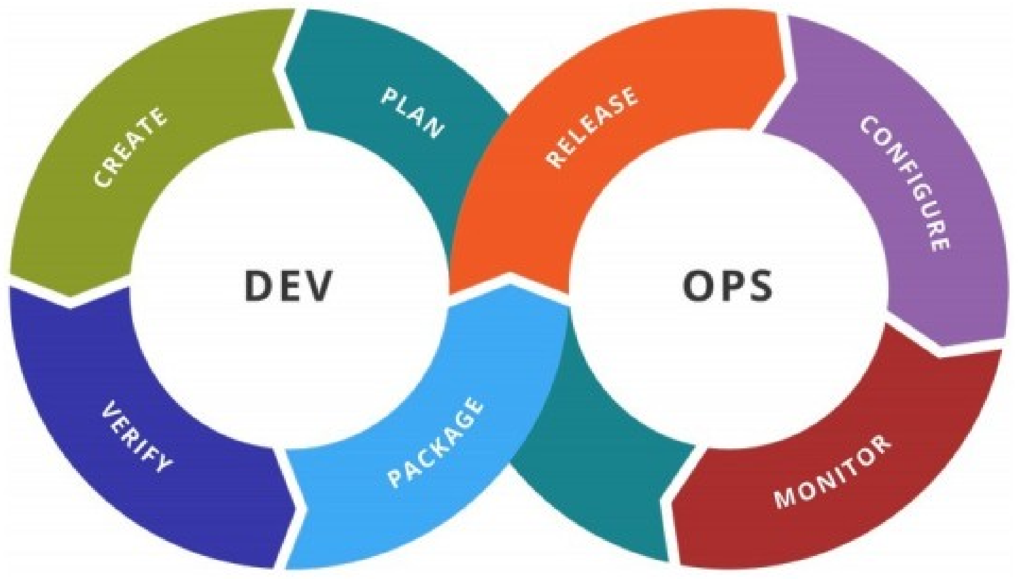


DevOps Goals

- Improve:
  - Time-to-Market
  - Feedback loop delay
  - Commit-to-Deploy (bugfix, new feature)
  - Quality
  - Efficiency
- Very frequent releases
- Fully automated release and deployment pipeline
- Continuous Integration (CI)
- Continuous Delivery (& Continuous Deployment)

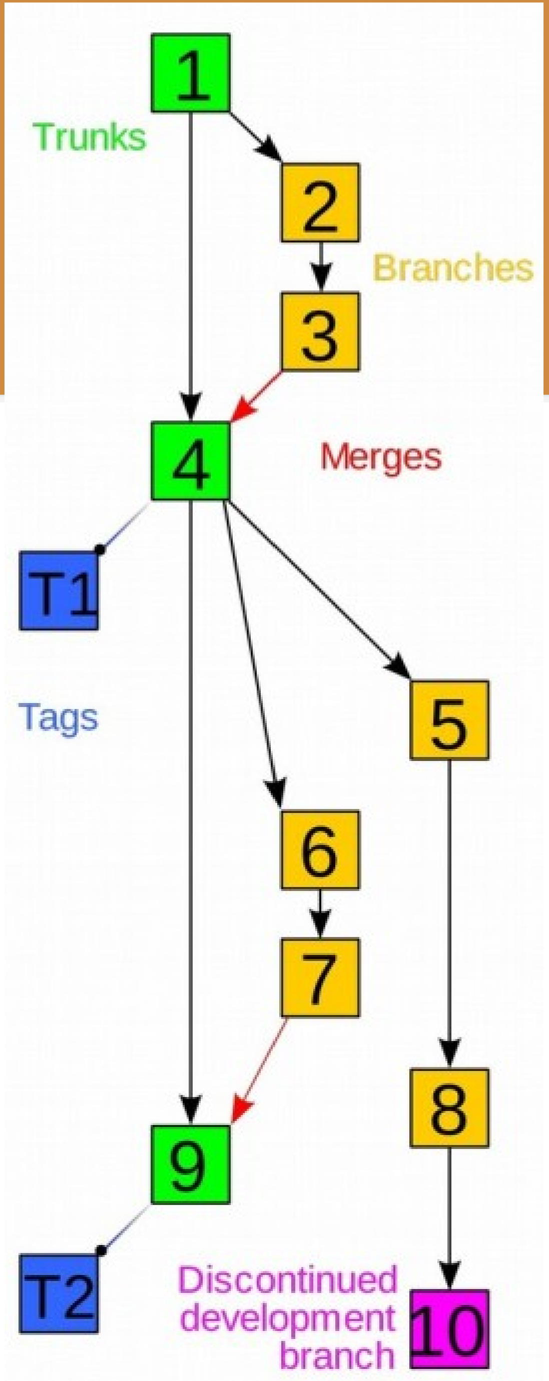
DevOps Pipeline

1. **Code & Review**
2. Build (CI & status)
3. Test
4. Package (Artifact Repository, Staging environment)
5. Release
6. Configure (Infrastructure as Code)
7. Monitor (Errors, Performance, Statistics, UX)

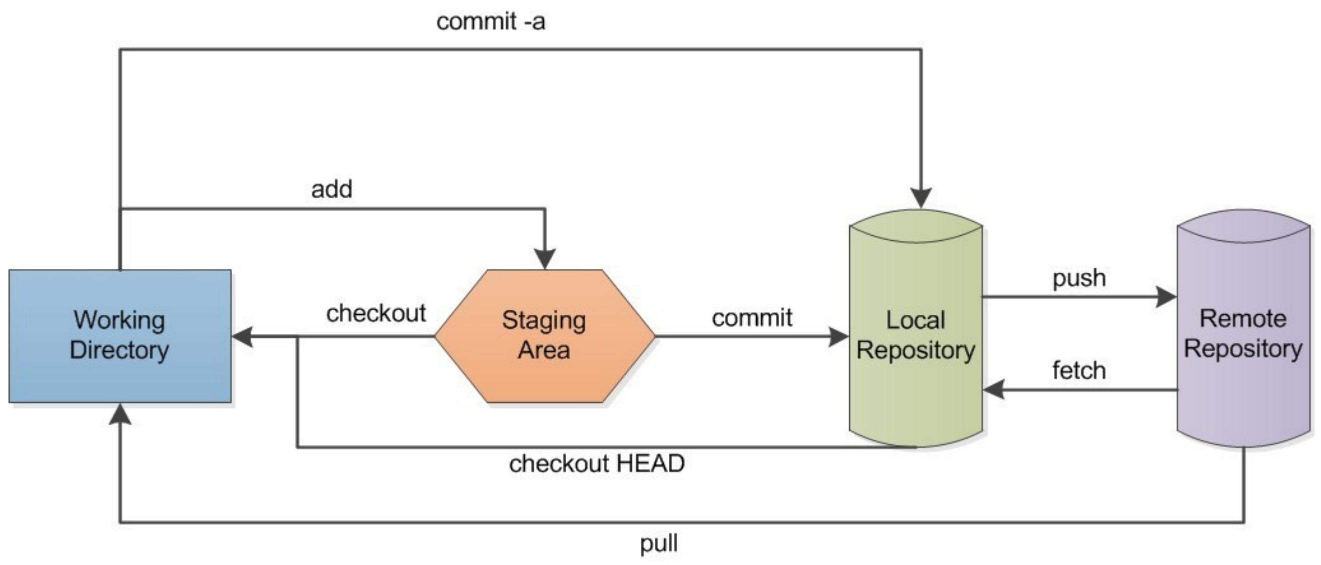


Code & Review

- Version Control Systems (VCS)
- Organization:
  - Centralized (Subversion (SVN))
  - Distributed (Git, Mercurial, Bazaar)
- Workflow
  - Branching
  - Merging (Integrating)
  - Tags (Releases)

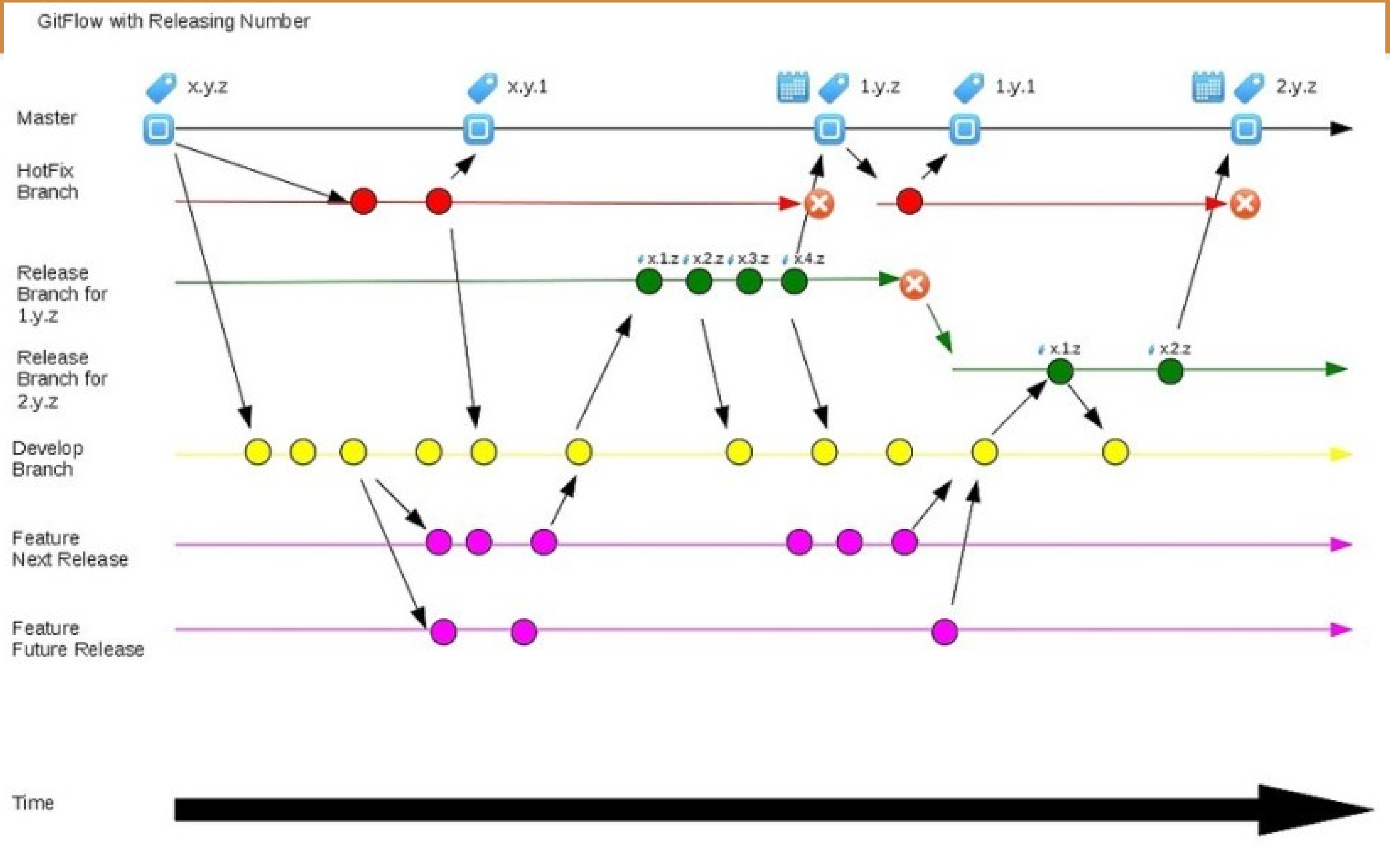


Code & Review
Distributed VCS (Git) operations

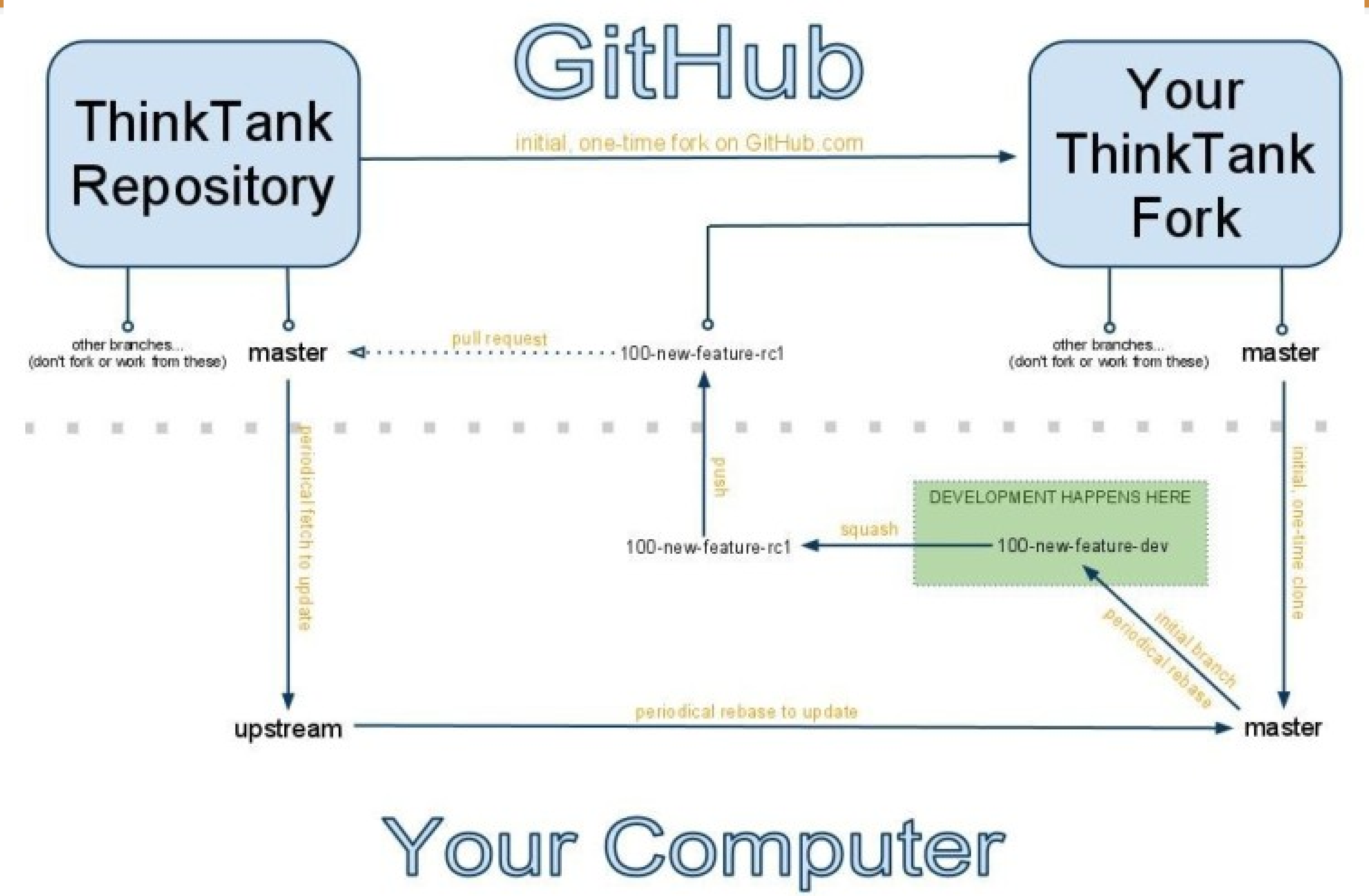


commit -a: Directly commit modified and deleted files into the local repository (*no new files!*)
add: Add a file to the staging area.
checkout: Get a file from the staging area.
checkout HEAD: Get a file from the local repository
commit: Commit files from the staging area to the local repository
push: Send files to the remote repository
fetch: Get files from the remote repository
pull: Get files from the remote repository and put a copy in the working directory

Code & Review
VCS Branching

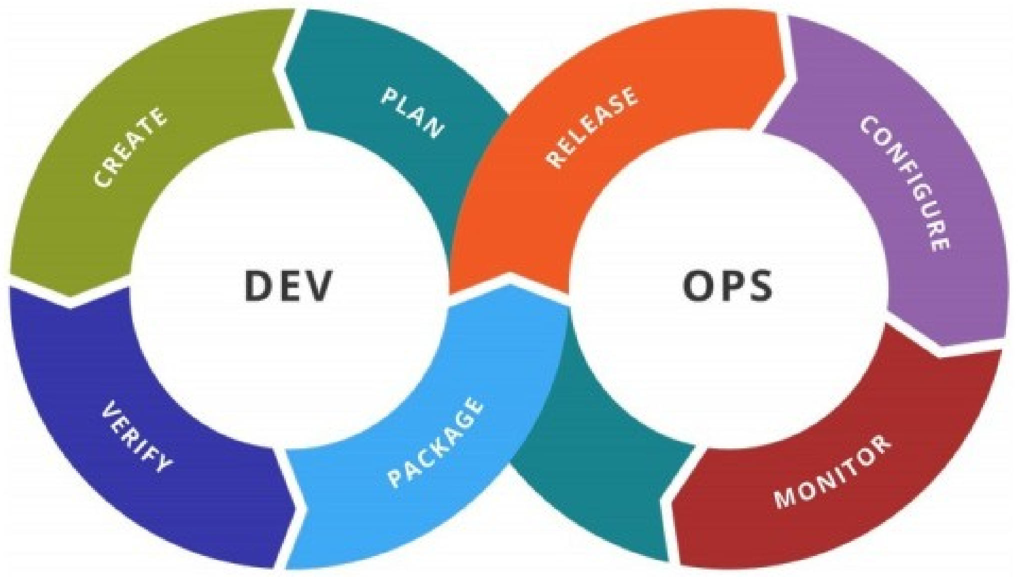


Code & Review
Merging tools (Github Pull Requests)



DevOps Pipeline

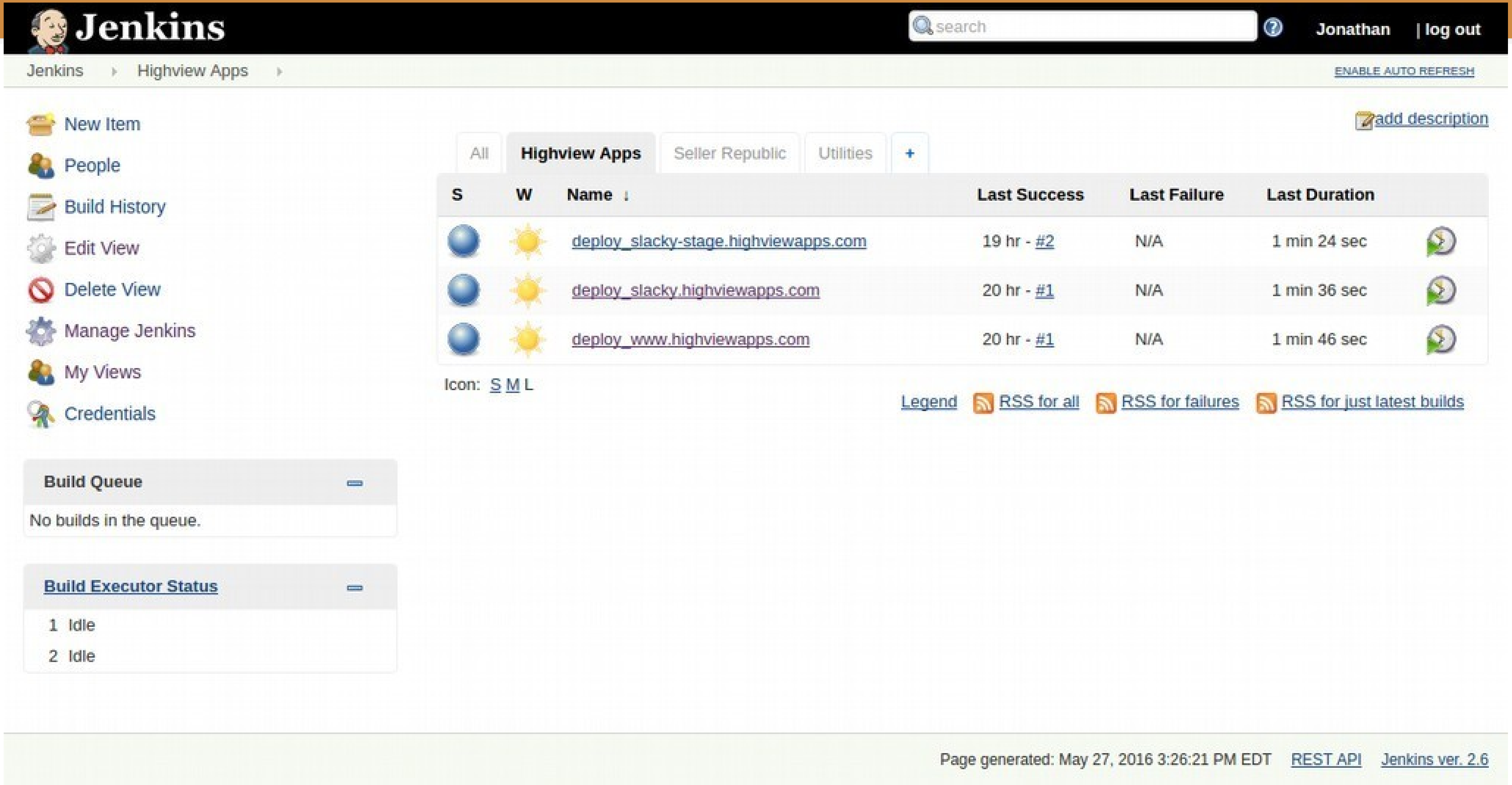
1. Code & Review
2. **Build (CI & status)**
3. **Test**
4. **Package (Artifact Repository, Staging environment)**
5. Release
6. Configure (Infrastructure as Code)
7. Monitor (Errors, Performance, Statistics, UX)



Build, Test, Package

- CI Practices
  - Central Code Repository (can be multiple)
  - Automated Build
  - Automated Tests
  - Almost like production testing (Staging or Preprod)
  - Not much branches (everyone is close to trunk / master → short round-trip)
  - Every commit is Built and Tested
  - Automate deployment into Artifact Repository
  - Results: Build Dashboard

Build, Test, Package
CI Tools (Jenkins)



Jenkins | search | Jonathan | log out

Jenkins › Highview Apps ›   ENABLE AUTO REFRESH

- New Item
- People
- Build History
- Edit View
- Delete View
- Manage Jenkins
- My Views
- Credentials

add description

All | **Highview Apps** | Seller Republic | Utilities | +

S	W	Name ↓	Last Success	Last Failure	Last Duration	
●	☀	deploy_slacky-stage.highviewapps.com	19 hr - #2	N/A	1 min 24 sec	
●	☀	deploy_slacky.highviewapps.com	20 hr - #1	N/A	1 min 36 sec	
●	☀	deploy_www.highviewapps.com	20 hr - #1	N/A	1 min 46 sec	

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Page generated: May 27, 2016 3:26:21 PM EDT REST API Jenkins ver. 2.6

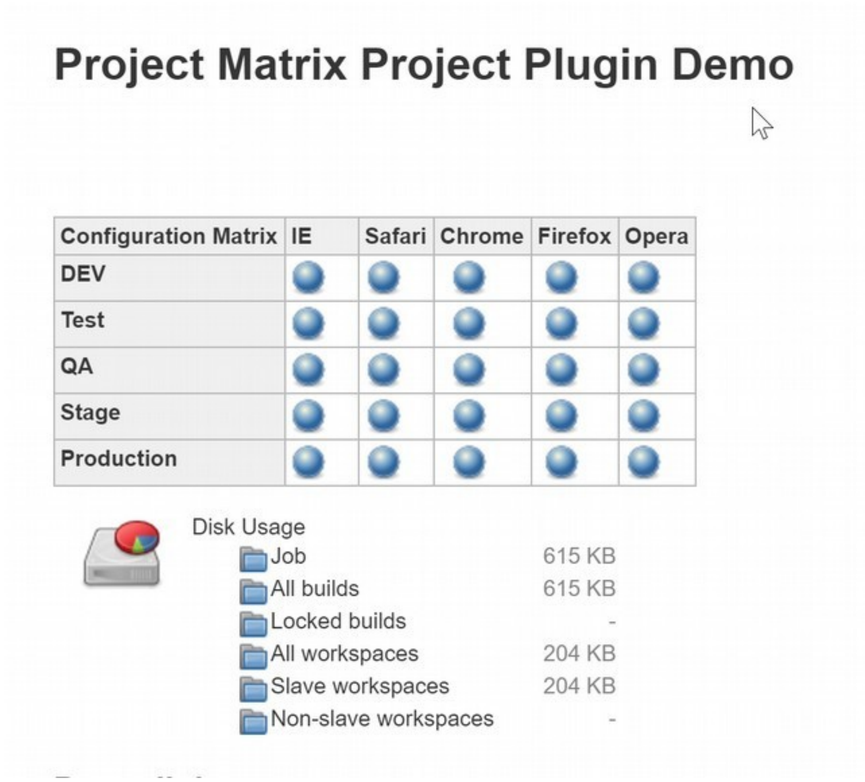
Build, Test, Package

- *“Earlier it is caught, cheaper to fix”*

- Testing frameworks (Junit, Mockito, Gtest…)
- Build tools (command line)
  - Ant, Maven, Gradle (Ivy, Nexus = Artifact or Binary Repositories)
  - Make (autotools), Ninja, CMake (Cross IDE, Cross Platform)

Build, Test, Package

- Matrix Builds = lot of build artifacts according to different categories
  - Debug / ReleaseWithDebug / Release (+Obfuscation)
  - Free / Commercial / With-extra-feature
  - Release per branch
  - Per platform builds
  - Special builds
    - Coverage
    - Memory checking
    - Thread checking



Project Matrix Project Plugin Demo

Configuration Matrix	IE	Safari	Chrome	Firefox	Opera
DEV	●	●	●	●	●
Test	●	●	●	●	●
QA	●	●	●	●	●
Stage	●	●	●	●	●
Production	●	●	●	●	●

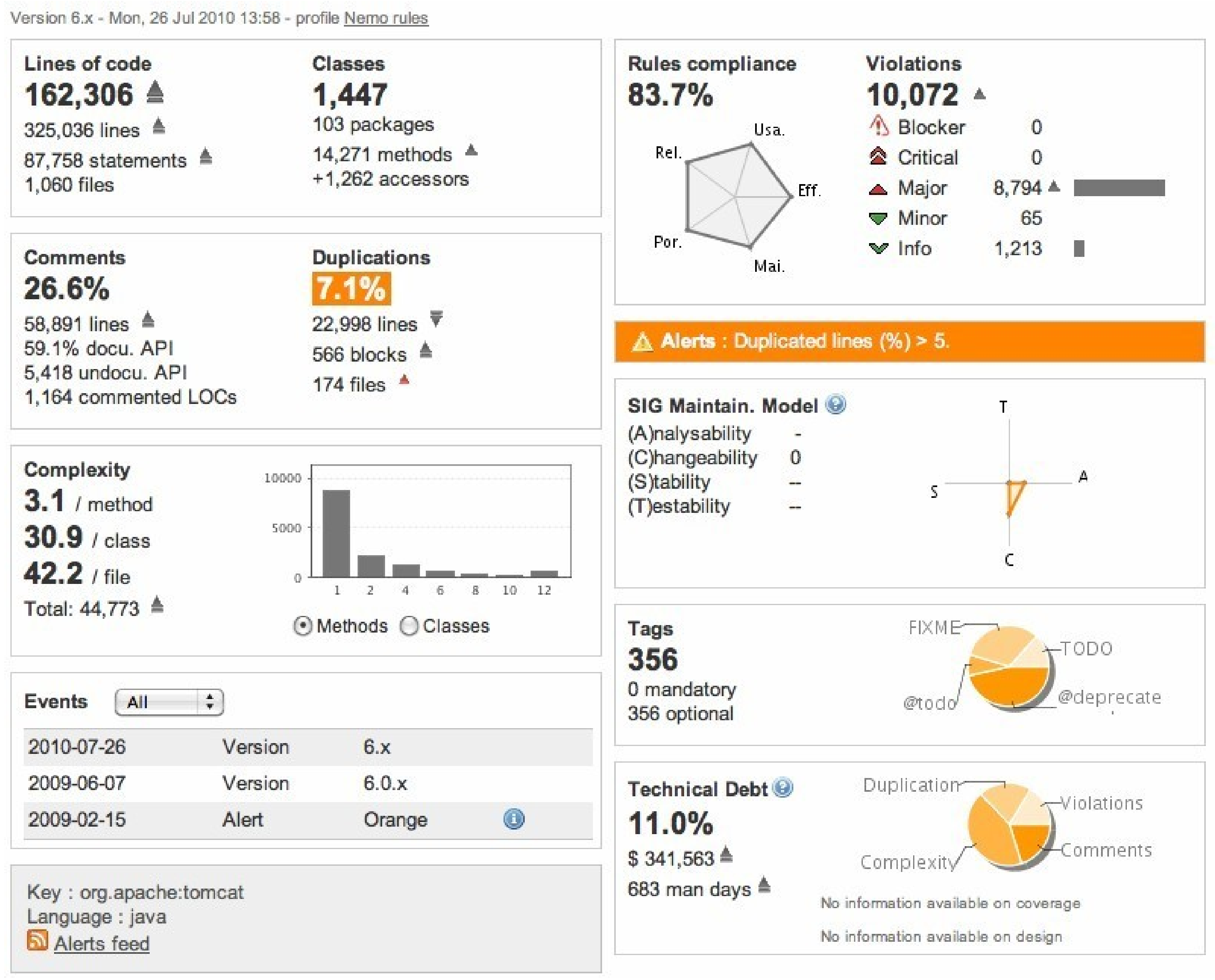
Disk Usage

Job	615 KB
All builds	615 KB
Locked builds	-
All workspaces	204 KB
Slave workspaces	204 KB
Non-slave workspaces	-

Build, Test, Package

- *“Earlier it is caught, cheaper to fix”*
- Compiler Errors, Warnings, Warning Levels
- Other verification
  - Coding Convention Enforcement (clang-format, pep8)
  - Code Metrics
  - Static Code Analysis
    - example: SonarQube, Lint, clang-format, clang-tidy, clang, Eclipse, FindBugs, PMD, pep8, Pylint, PyCharm
- Coverage
- Profiling (Profiling or Sampling)

Build, Test, Package



Version 6.x - Mon, 26 Jul 2010 13:58 - profile Nemo rules

Lines of code
162,306 ▲
325,036 lines ▲
87,758 statements ▲
1,060 files

Classes
1,447
103 packages
14,271 methods ▲
+1,262 accessors

Rules compliance
83.7%
Usa. Eff. Mai. Por. Rel.

Violations
10,072 ▲

Blocker	0
Critical	0
Major	8,794 ▲
Minor	65
Info	1,213

Comments
26.6%
58,891 lines ▲
59.1% docu. API
5,418 undocu. API
1,164 commented LOCs

Duplications
7.1%
22,998 lines ▼
566 blocks ▲
174 files ▲

⚠ Alerts : Duplicated lines (%) > 5.

SIG Maintain. Model

(A)nalysability	-
(C)hangeability	0
(S)tability	--
(T)estability	--

Complexity
3.1 / method
30.9 / class
42.2 / file
Total: 44,773 ▲

(● Methods ○ Classes)

Tags
356
0 mandatory
356 optional

Events All

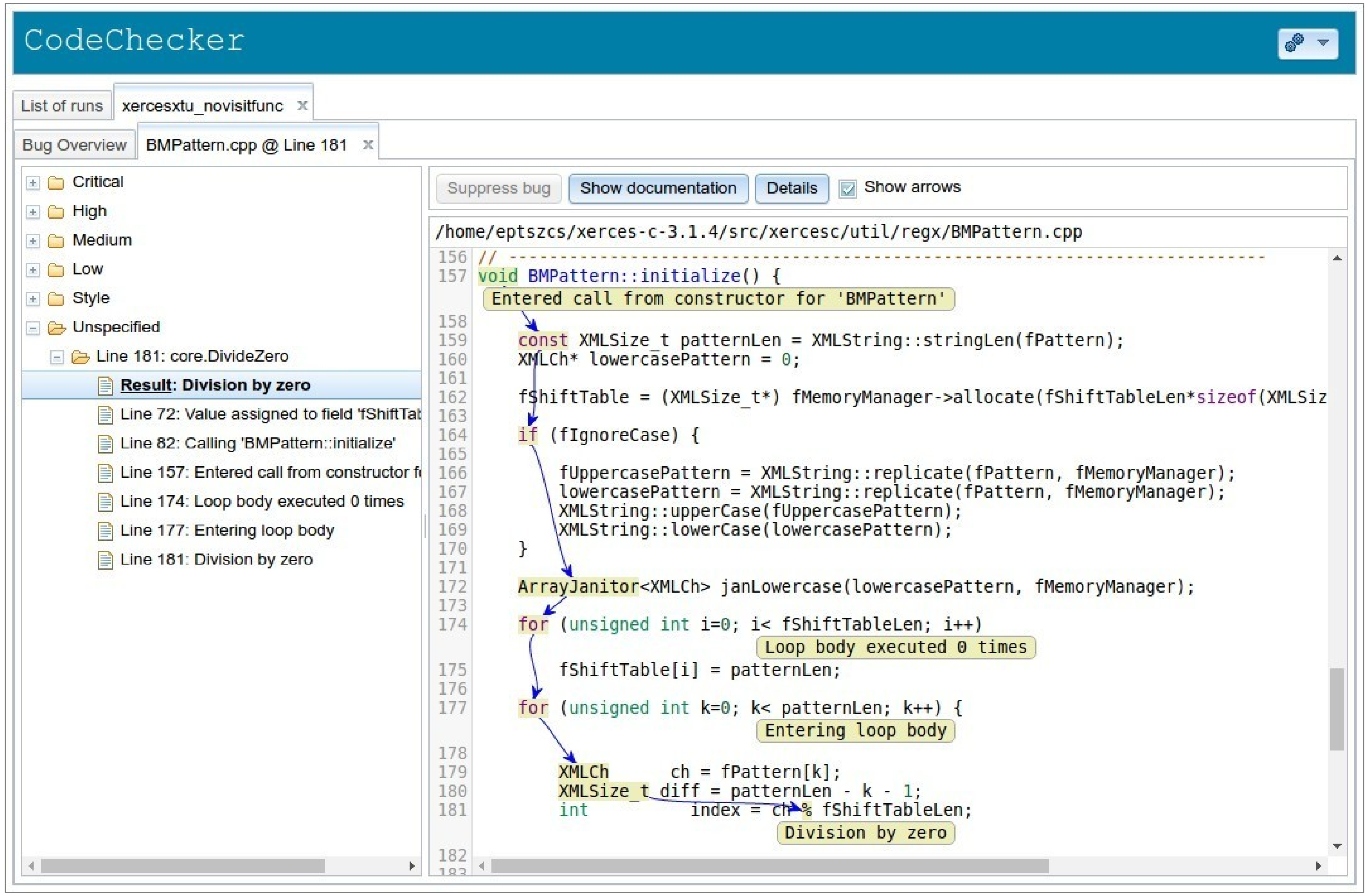
2010-07-26	Version	6.x
2009-06-07	Version	6.0.x
2009-02-15	Alert	Orange

Technical Debt
11.0%
$ 341,563 ▲
683 man days ▲

No information available on coverage
No information available on design

Key : org.apache:tomcat
Language : java
Alerts feed

Build, Test, Package



CodeChecker

List of runs | xercesxtu_novisitfunc

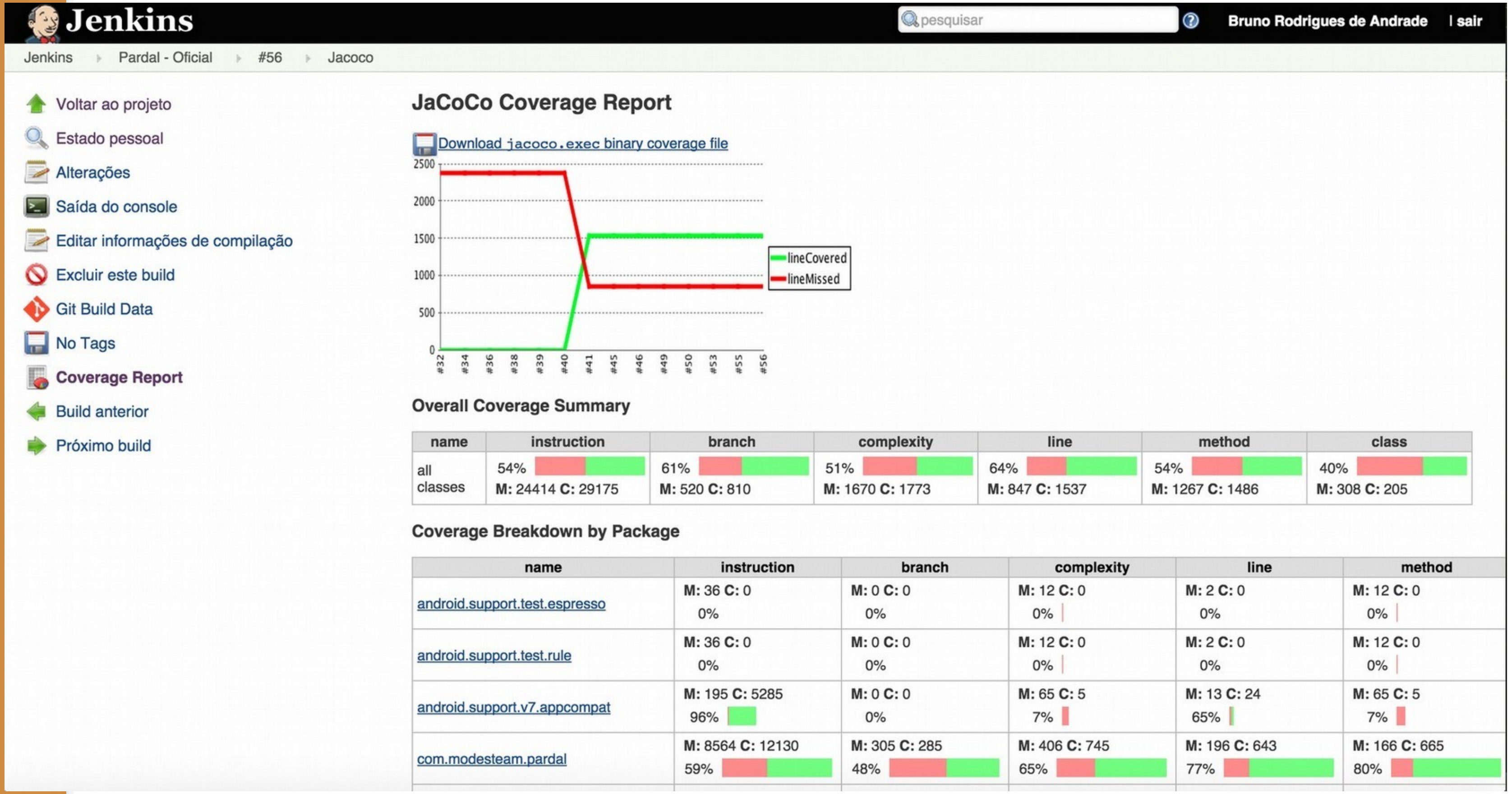
Bug Overview | BMPattern.cpp @ Line 181

- Critical
- High
- Medium
- Low
- Style
- Unspecified
  - Line 181: core.DivideZero
    - **Result: Division by zero**
    - Line 72: Value assigned to field 'fShiftTab
    - Line 82: Calling 'BMPattern::initialize'
    - Line 157: Entered call from constructor f
    - Line 174: Loop body executed 0 times
    - Line 177: Entering loop body
    - Line 181: Division by zero

/home/eptszcs/xerces-c-3.1.4/src/xercesc/util/regx/BMPattern.cpp

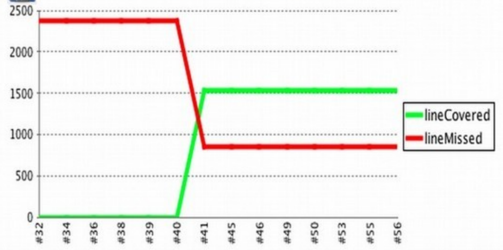
```
156 // ---------------------------------------------------------------------------
157 void BMPattern::initialize() {
        Entered call from constructor for 'BMPattern'
158
159     const XMLSize_t patternLen = XMLString::stringLen(fPattern);
160     XMLCh* lowercasePattern = 0;
161
162     fShiftTable = (XMLSize_t*) fMemoryManager->allocate(fShiftTableLen*sizeof(XMLSiz
163
164     if (fIgnoreCase) {
165
166         fUppercasePattern = XMLString::replicate(fPattern, fMemoryManager);
167         lowercasePattern = XMLString::replicate(fPattern, fMemoryManager);
168         XMLString::upperCase(fUppercasePattern);
169         XMLString::lowerCase(lowercasePattern);
170     }
171
172     ArrayJanitor<XMLCh> janLowercase(lowercasePattern, fMemoryManager);
173
174     for (unsigned int i=0; i< fShiftTableLen; i++)
                               Loop body executed 0 times
175         fShiftTable[i] = patternLen;
176
177     for (unsigned int k=0; k< patternLen; k++) {
                               Entering loop body
178
179         XMLCh      ch = fPattern[k];
180         XMLSize_t diff = patternLen - k - 1;
181         int        index = ch % fShiftTableLen;
                                 Division by zero
182
```

Build, Test, Package



JaCoCo Coverage Report

Download jacoco.exec binary coverage file



Overall Coverage Summary

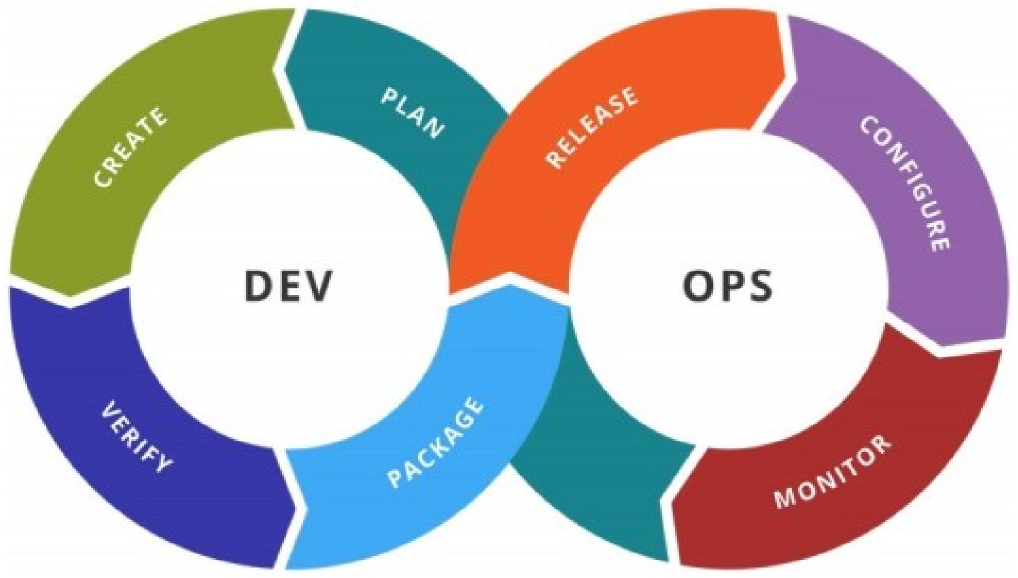
name	instruction	branch	complexity	line	method	class
all classes	54% 🟥🟩 **M:** 24414 **C:** 29175	61% 🟥🟩 **M:** 520 **C:** 810	51% 🟥🟩 **M:** 1670 **C:** 1773	64% 🟥🟩 **M:** 847 **C:** 1537	54% 🟥🟩 **M:** 1267 **C:** 1486	40% 🟥🟩 **M:** 308 **C:** 205

Coverage Breakdown by Package

name	instruction	branch	complexity	line	method
android.support.test.espresso	**M:** 36 **C:** 0 0%	**M:** 0 **C:** 0 0%	**M:** 12 **C:** 0 0% 🟥	**M:** 2 **C:** 0 0%	**M:** 12 **C:** 0 0% 🟥
android.support.test.rule	**M:** 36 **C:** 0 0%	**M:** 0 **C:** 0 0%	**M:** 12 **C:** 0 0% 🟥	**M:** 2 **C:** 0 0%	**M:** 12 **C:** 0 0% 🟥
android.support.v7.appcompat	**M:** 195 **C:** 5285 96% 🟩	**M:** 0 **C:** 0 0%	**M:** 65 **C:** 5 7% 🟥	**M:** 13 **C:** 24 65% 🟩	**M:** 65 **C:** 5 7% 🟥
com.modesteam.pardal	**M:** 8564 **C:** 12130 59% 🟥🟩	**M:** 305 **C:** 285 48% 🟥🟩	**M:** 406 **C:** 745 65% 🟥🟩	**M:** 196 **C:** 643 77% 🟥🟩	**M:** 166 **C:** 665 80% 🟥🟩

DevOps Pipeline

1. Code & Review
2. Build (CI & status)
3. Test
4. Package (Artifact Repository, Staging environment)
5. **Release**
6. **Configure (Infrastructure as Code)**
7. Monitor (Errors, Performance, Statistics, UX)

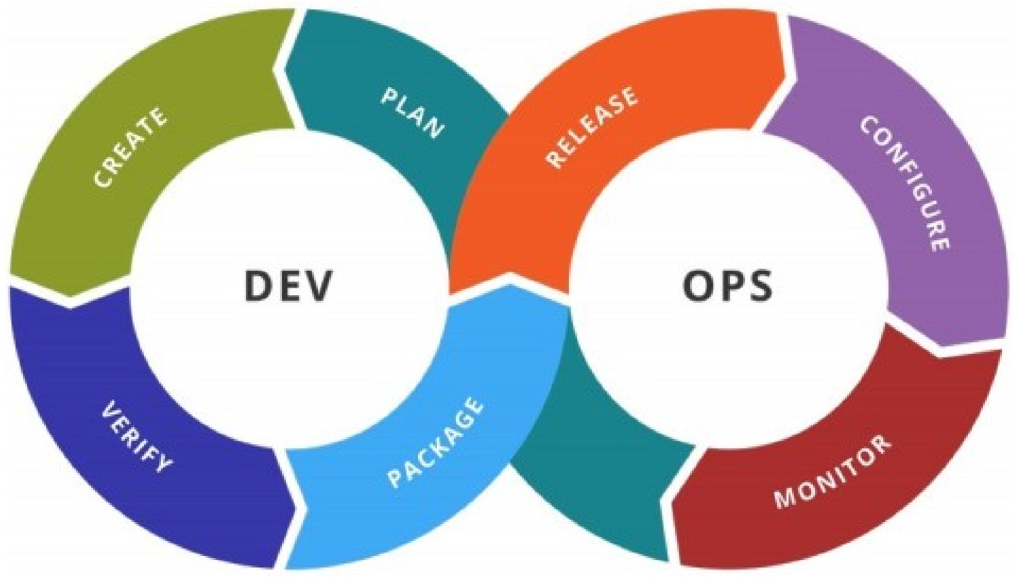


Release & Configure

- Deployment scalability → Virtualization
  - Full (KVM, Xen, QEMU, VirtualBox)
  - OS-Level (Docker, LXC / LXD, OpenVZ)
- Infrastructure as Code
  - Declarative (functional) vs Imperative (procedural)
  - Push or Pull (towards controller server)
  - Continuous Configuration Automation (CCA) (Chef, Puppet, Vagrant)

DevOps Pipeline

1. Code & Review
2. Build (CI & status)
3. Test
4. Package (Artifact Repository, Staging environment)
5. Release
6. Configure (Infrastructure as Code)
7. **Monitor (Errors, Performance, Statistics, UX)**



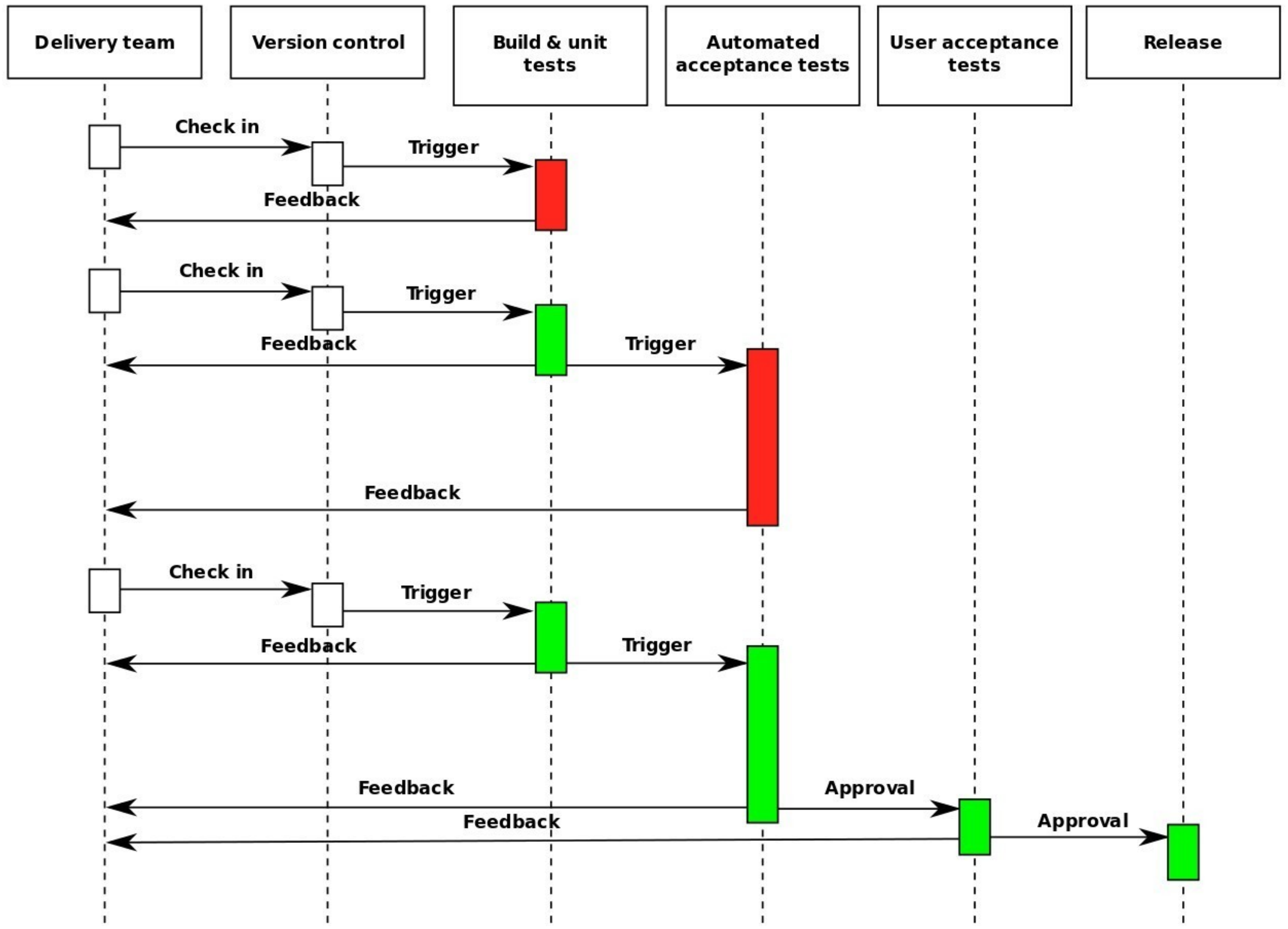
Monitor
Application Performance Monitoring

- Not Profiling
- General usage statistics
- UX Monitoring
  - Command chain analysis
  - Time measurement
- System component monitoring
  - Measure critical times (loading, waiting for network…)
  - Micro measure time spent in subsystems

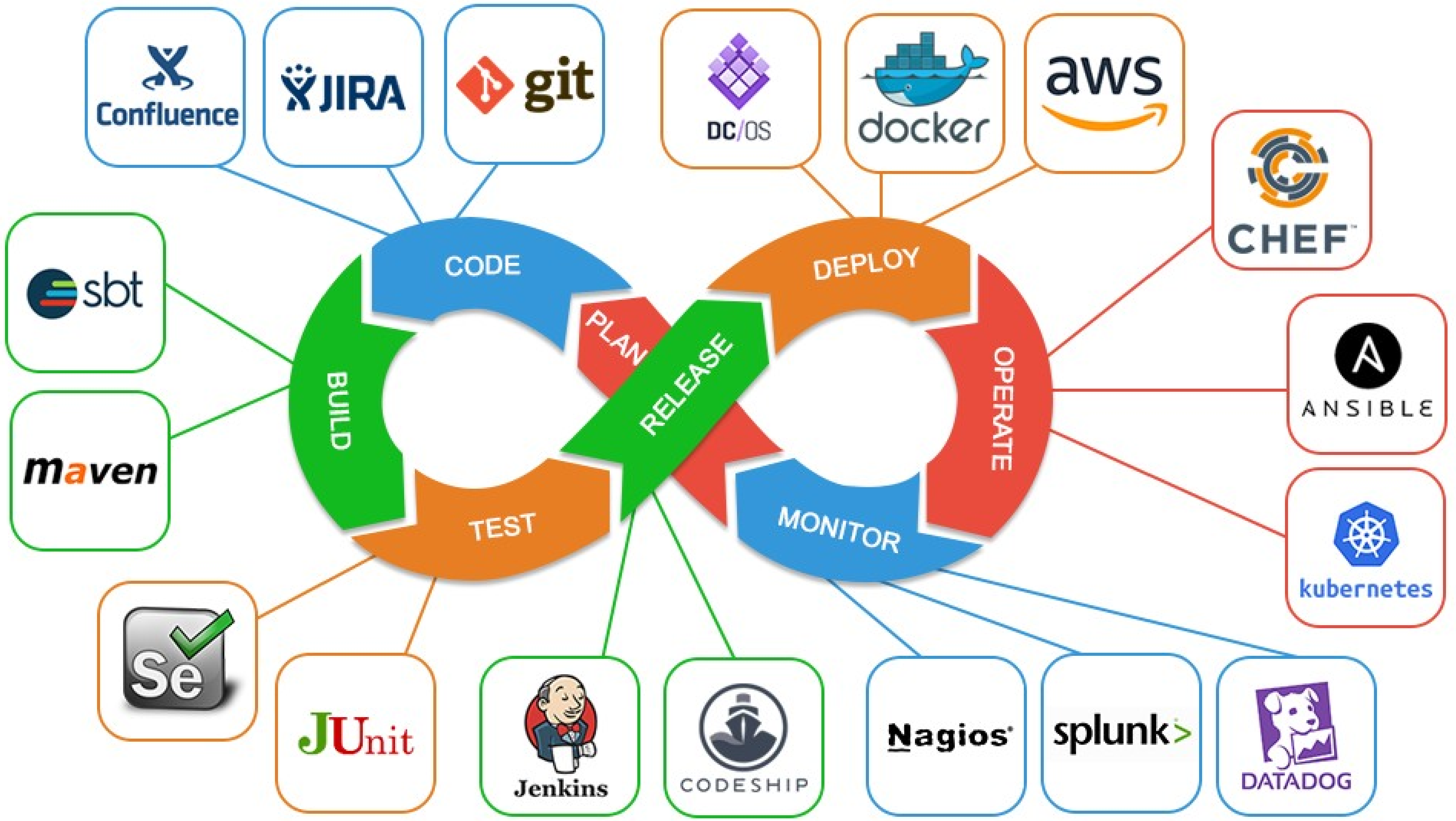
Monitor
Application Performance Monitoring

- Extensive Logging
  - Log levels
  - Log types
  - Log modules
- Structured Logging
- Live / Real-time Dashboards

Continuous Delivery



DevOps Pipeline with tools



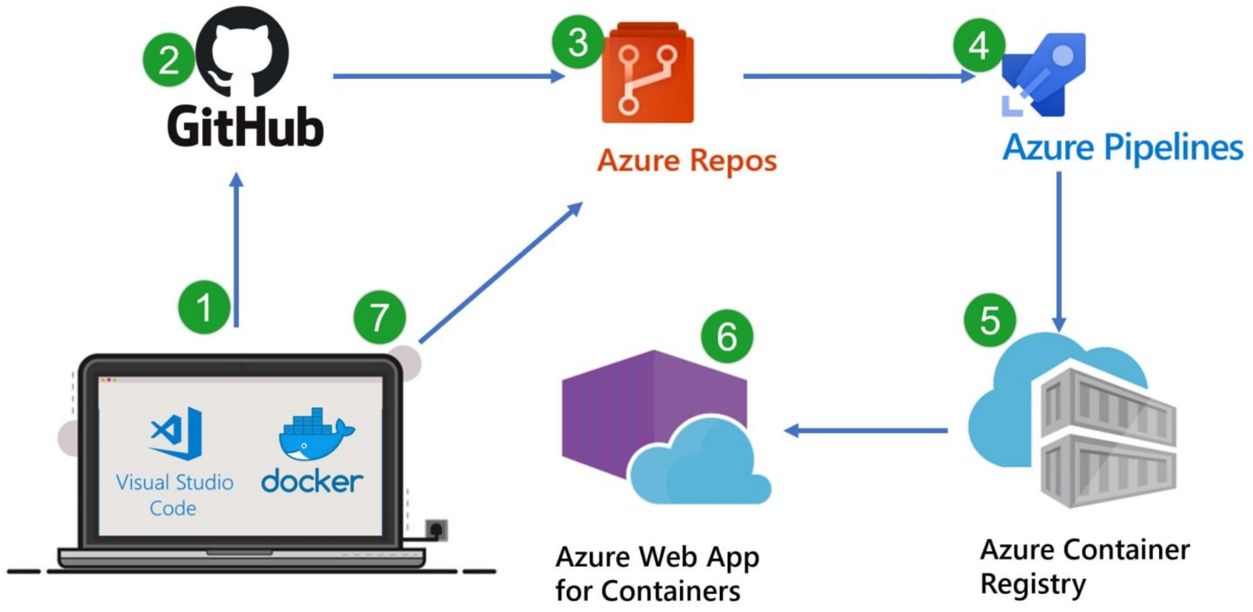
Other weblinks

- A beginner's guide to building DevOps pipelines with open-source tools
  - https://opensource.com/article/19/4/devops-pipeline

- What is DevOps Pipeline & How to Build One
  - https://phoenixnap.com/blog/devops-pipeline

Continuous Integration (CI) with Azure Pipelines and .NET Core Step-by-step tutorial

https://cloudskills.io/blog/ci-dotnet-core

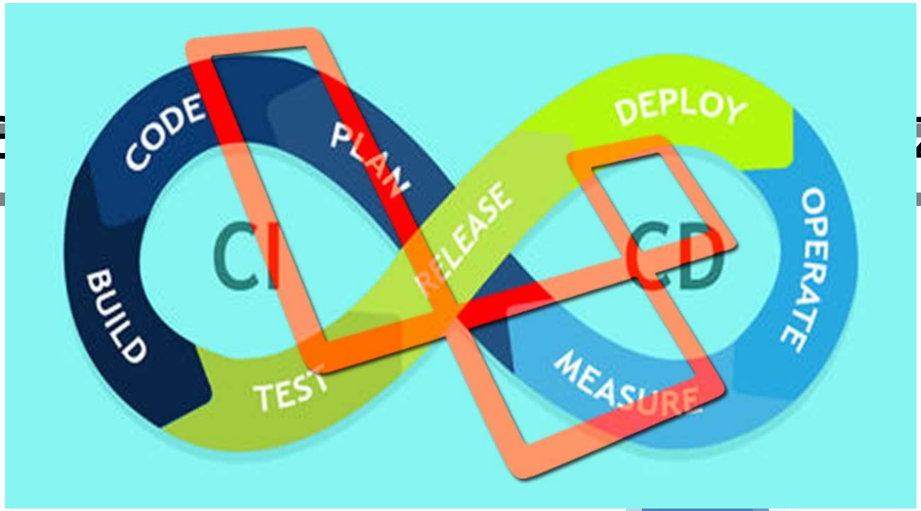


Configure Laravel properly for CI/CD
https://medium.com/faun/configure-laravel-properly-for-ci-cd-6f9965034108

How to setup and run Laravel in Docker Container

https://morioh.com/p/46ef037a07c5

Kép be z
ikonra

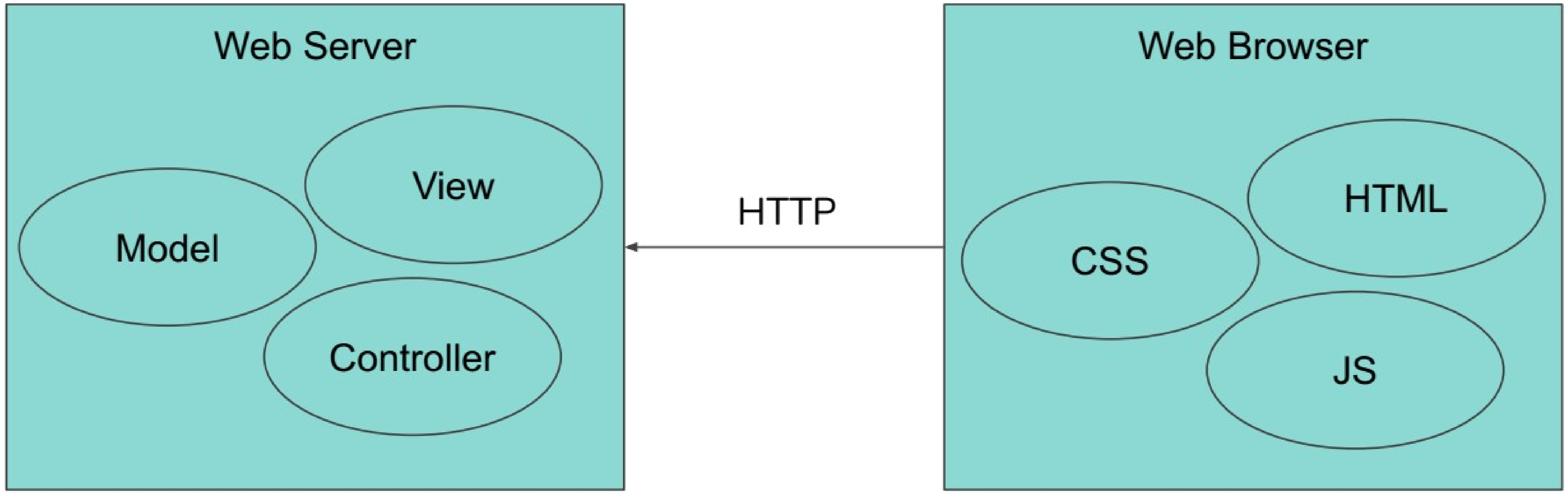


Questions?

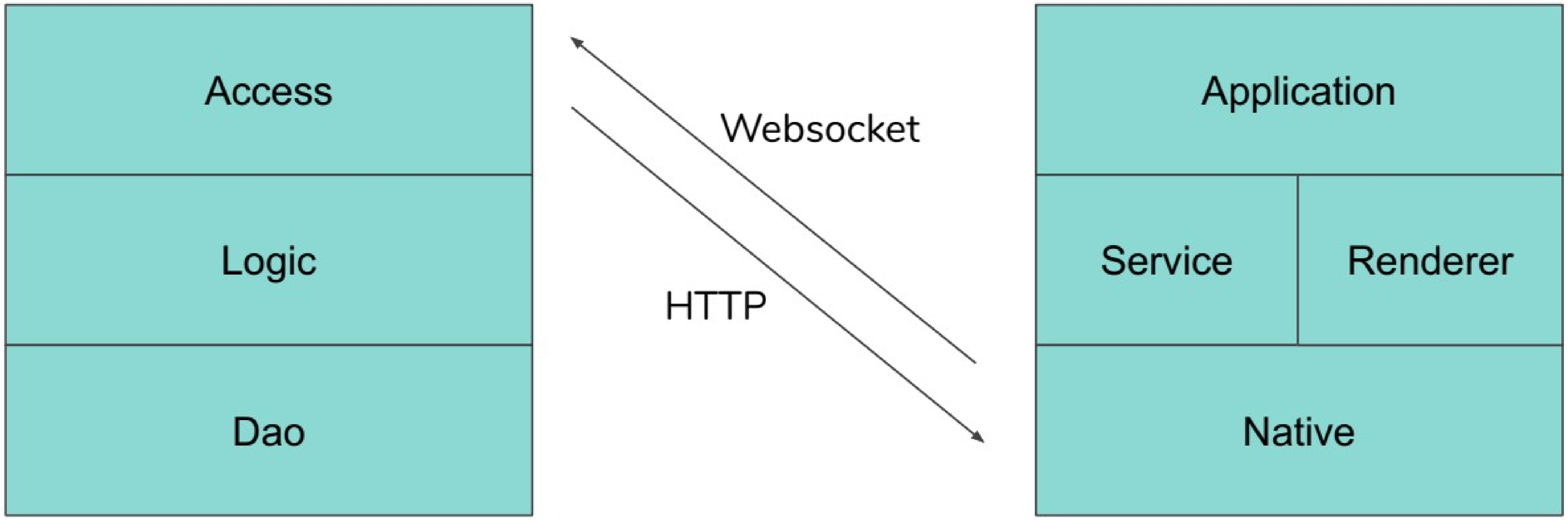
- ...
- Or write me an email to gla@inf.elte.hu

System Architecture

Traditional Architecture



API based Architecture



Advantages

- The business logic is to be implemented only once. Porting your software to a new platform does not mean the reimplementation of the whole business logic
- There is a strict border between the user interface and the business logic development
  - The UI developer has no chance to reach the lower layers → The architecture is cleaner
  - The roles are better defined. The tasks are separated. The frontend and the backend development needs a different attitude
  - As there is a defined communication interface, an API, the responsibilities are better separated

Service Oriented Architecture

- Technology and vendor independent
- A service-oriented architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network
- Service-oriented architecture is less about how to modularize an application, and more about how to compose an application by integration of distributed, separately-maintained and deployed software components
- See: https://en.wikipedia.org/wiki/Service-oriented_architecture

SOA Principles

- A service has four properties according to one of many definitions of SOA
  - It logically represents a business activity with a specified outcome
  - It is self-contained
  - It is a black box for its consumers
  - It may consist of other underlying services

Applying SOA

- We do not develop an enterprise grade architecture software during the course
- The SOA principles are to be applied
- Helpful if you have to extend the service later or integrate with another service
- The business logic can be separated from the user interface logic

Recommended Protocol

- JSON over HTTP
  - Easy to implement
  - General regarding various data structures
  - Fits most network infrastructures because of the HTTP
- Utilize the HTTP status
  - 200 - Ok
  - 500 - Server error

Protocol Example

- The endpoint is: http://www.example.com/service/userLogin/login
- POST the following JSON to the HTTP end point

  {”email”:”laszlo.grad-gyenge@inf.elte.hu”,”password”:”secret”}
- Reply examples

  {true} / {false}

  {

  “errorClass” : “DatabaseException”,

  “errorMessage” : “Unable to connect to the database”

  }

HTTP to Method Mapping

- The system consists of several class instances
- Some of the classes are to be published (access layer). To be more exact, some of the methods of these classes are to be published
- I recommend you to use a mapping like:

  <API endpoint>/<class name>/<method name>
- You may use the REST API paradigm. See:
  - https://en.wikipedia.org/wiki/Representational_state_transfer

Questions?

- …
- Or write me an email to gla@inf.elte.hu