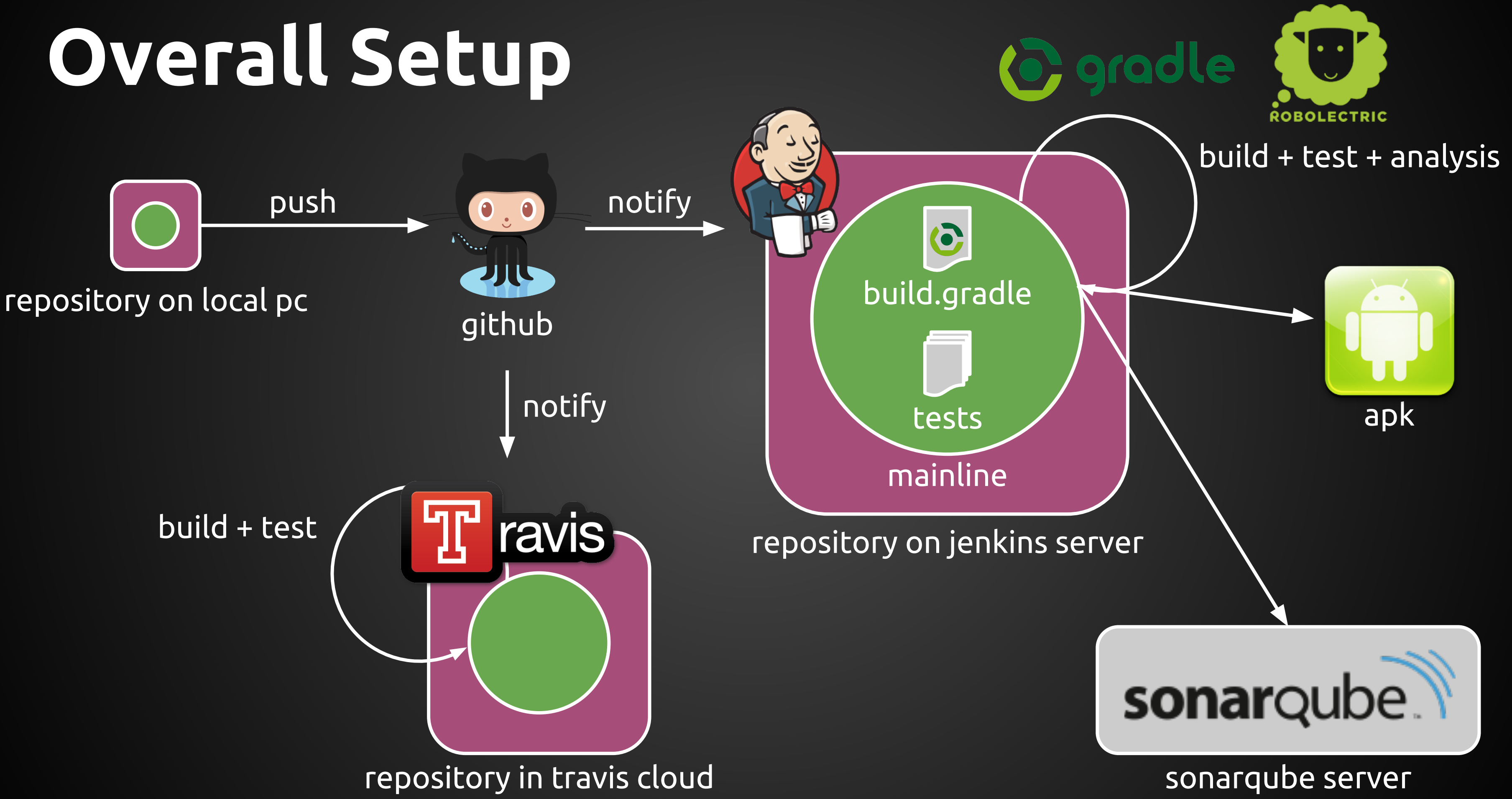
Android

Software Quality & Testing

Disclaimer

In the scope of this presentation, we will show one possible setup with our individual choice of tools. This is by no means exhaustive and there exist many alternative ways to improve software quality in Android projects.

Overall Setup

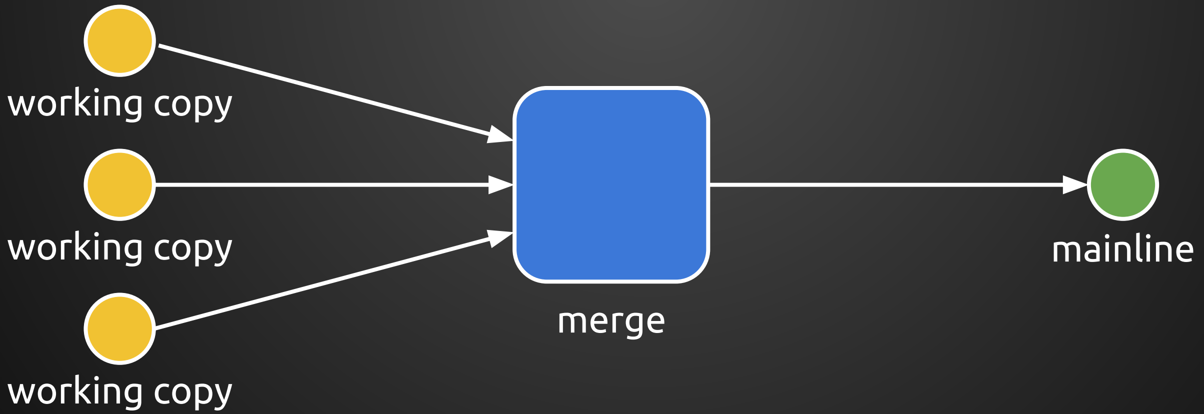


Agenda

Continuous Integration	Definition
Source Control	Git
Unit-Testing	JUnit, Robolectric
Static Code Analysis	SonarQube
Dependency Management	Gradle
Build Automation	Gradle
Continuous Integration Server	Travis, Jenkins

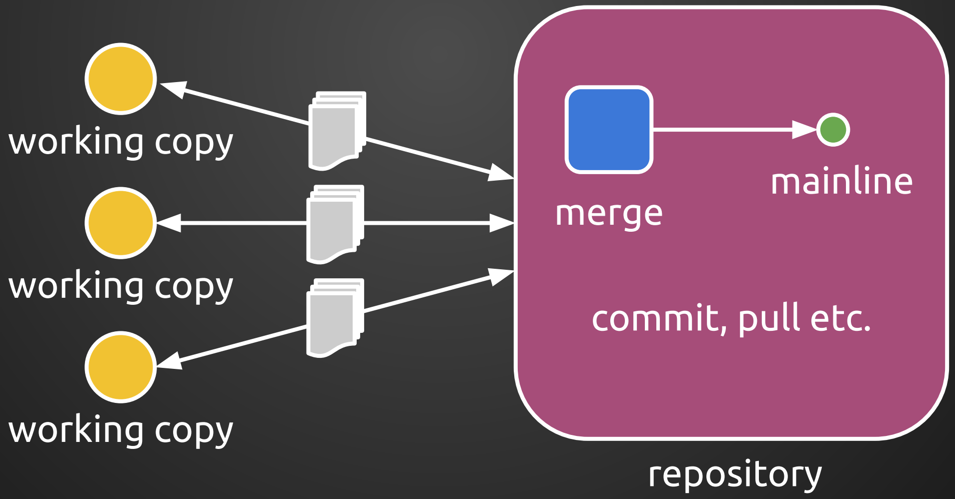
Continuous Integration

Continuous integration (CI) is the practice of continuously merging all developer working copies with a shared mainline. This usually also covers building and testing.



Source Control

Source Control (Revision Control, Version Control) is the management of changes to files.



Source Control

In our case git is the source control system of choice:

Get the code from the remote repository:

```
git clone https://github.com/schreon/matchmaking.git
```

Get the newest code changes:

```
git pull
```

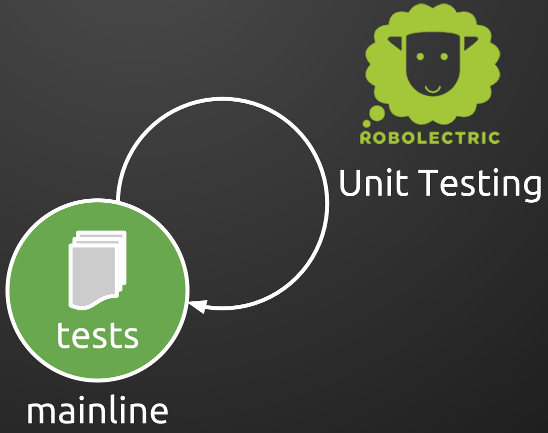
Source Control

Update the repository:

```
git add README.md
git commit -m "readme changed"
git push origin master
```

Unit Testing

Unit Testing is a method by which individual units of source code (classes, functions) and sets of one or more modules are tested to determine if they are fit for use.



Unit Testing

We use JUnit 4 as Testing Framework:

- Tests are source code themselves and are part of the repository

```
@RunWith(JUnit4.class)
public class FooTest {
  @Test
  public void thisAlwaysPasses() {
    assertTrue(true);
  }
}
```

Unit Testing

The Problem:

With the official Android testing framework (JUnit), test cases must run on an emulator or a device to test against the Android API.

→ Slow on developer machines

→ Not viable on an automated build server

Robolectric

- Uses JUnit 4 as underlying Testing Framework
- Unit Test Framework
- de-fangs the Android SDK jar (rewrites Android SDK classes as they're being loaded)
- Tests run inside the JVM and can be started with the IDE
- Simulation of certain events like disabling the wifi option

Robolectric

```java
// Test class for MyActivity
@RunWith(RobolectricTestRunner.class)
public class ActivityTest {

  @Test
  public void clickingButton_shouldChangeResultsViewText() throws Exception {
    Activity activity = Robolectric.buildActivity(MyActivity.class).create().get();

    Button pressMeButton = (Button) activity.findViewById(R.id.press_me_button);
    TextView results = (TextView) activity.findViewById(R.id.results_text_view);

    pressMeButton.performClick();
    String resultsText = results.getText().toString();
    assertThat(resultsText, equalTo("Testing Android Rocks!"));
  }
}
```

Gradle

Gradle automates the building, testing, deployment of our project.

- It load all required dependencies:
  - Dependency Management
- It schedules the following processes:
  - Build the app (generating the apk)
  - Perform Static Code Analysis
  - Generate Test Reports
  - Generate Test Coverage Reports

Dependency Management

Gradle relies on maven repositories to centralize the management of used libraries and third-party packages.

Together with gradle we only have to write

```
gradle compile
```

to get all dependencies of the project.

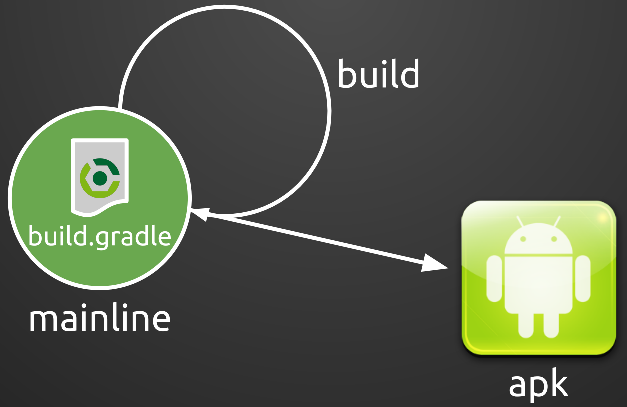
Dependency Management

```
// Configuration of the build script itself
buildscript {
    // Repositories which contain the libraries necessary for the buildscript itself
    repositories {
        mavenCentral()
        maven {
            url 'https://oss.sonatype.org/content/repositories/snapshots/'
        }
    }
    // Libraries which are needed for the build script itself
    dependencies {
        classpath 'com.android.tools.build:gradle:0.6.+'
        classpath 'com.squareup.gradle:gradle-android-test-plugin:0.9.1-SNAPSHOT'
    }
}
```

Build Management

The possible build steps are defined in the build.gradle file. This file is part of the repository.

The final product of the build is an apk file:



Static Code Analysis

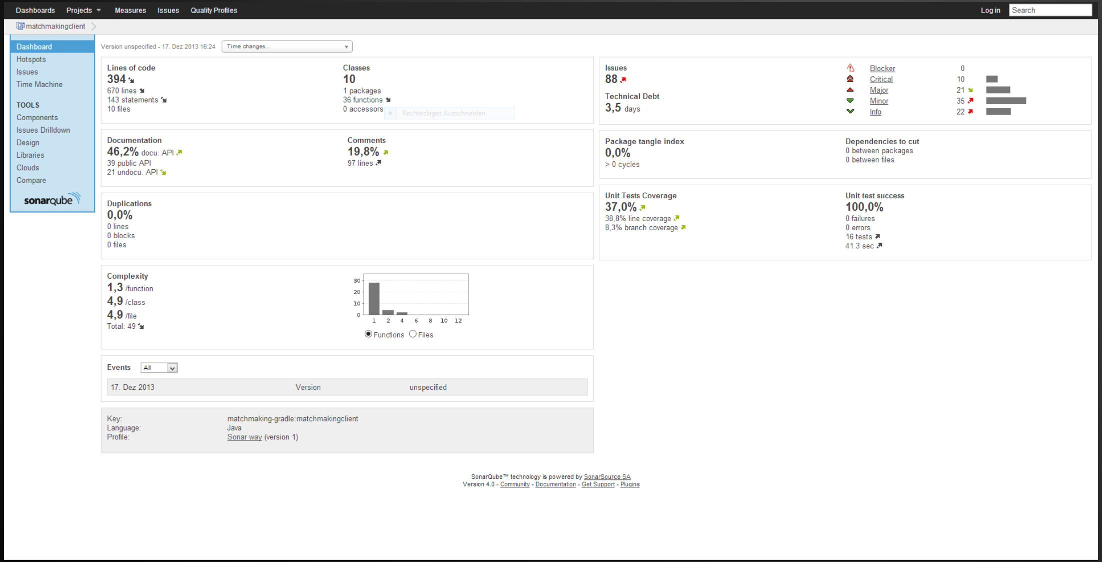
Static program analysis is the analysis of computer software that is performed without actually executing programs.

Common software metrics are:

- Lines of code
- Number of classes
- Amount of documentation
- Amount of duplications
- Complexity

SonarQube

Used to generate Static Code Analysis reports and display Analysis and Test results.



Version unspecified - 17. Dez 2013 16:24

Lines of code
394
670 lines
143 statements
10 files

Classes
10
1 packages
36 functions
0 accessors

Issues
88

Technical Debt
3,5 days

Blocker	0
Critical	10
Major	21
Minor	35
Info	22

Documentation
46,2% docu. API
39 public API
21 undocu. API

Comments
19,8%
97 lines

Package tangle index
0,0%
> 0 cycles

Dependencies to cut
0 between packages
0 between files

Duplications
0,0%
0 lines
0 blocks
0 files

Unit Tests Coverage
37,0%
38,8% line coverage
8,3% branch coverage

Unit test success
100,0%
0 failures
0 errors
16 tests
41.3 sec

Complexity
1,3 /function
4,9 /class
4,9 /file
Total: 49

[✓ Functions] Files

Events All

17. Dez 2013	Version	unspecified

Key:	matchmaking-gradle:matchmakingclient
Language:	Java
Profile:	Sonar way (version 1)

SonarQube™ technology is powered by SonarSource SA
Version 4.0 - Community - Documentation - Get Support - Plugins

SonarQube

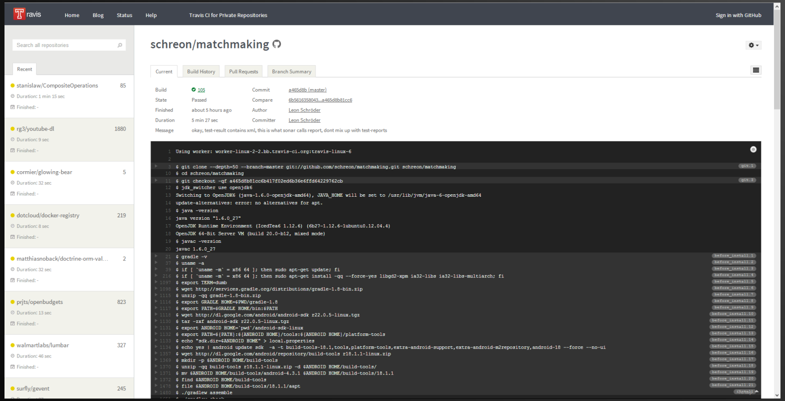
Run the tests and create software quality report:

```
gradle clean sonarRunner
```

Travis

An continuous integration server in the cloud which provides an **easy integration with github**.

build passing



Home Blog Status Help Travis CI for Private Repositories Sign in with GitHub

Search all repositories

Recent

stanislaw/CompositeOperations 85
Duration: 1 min 15 sec
Finished: -

rg3/youtube-dl 1880
Duration: 9 sec
Finished: -

cormier/glowing-bear 5
Duration: 32 sec
Finished: -

dotcloud/docker-registry 219
Duration: 8 sec
Finished: -

matthiasnoback/doctrine-orm-val... 2
Duration: 32 sec
Finished: -

prjts/openbudgets 823
Duration: 13 sec
Finished: -

walmartlabs/lumbar 327
Duration: 46 sec
Finished: -

surfly/gevent 245

schreon/matchmaking

Current | Build History | Pull Requests | Branch Summary

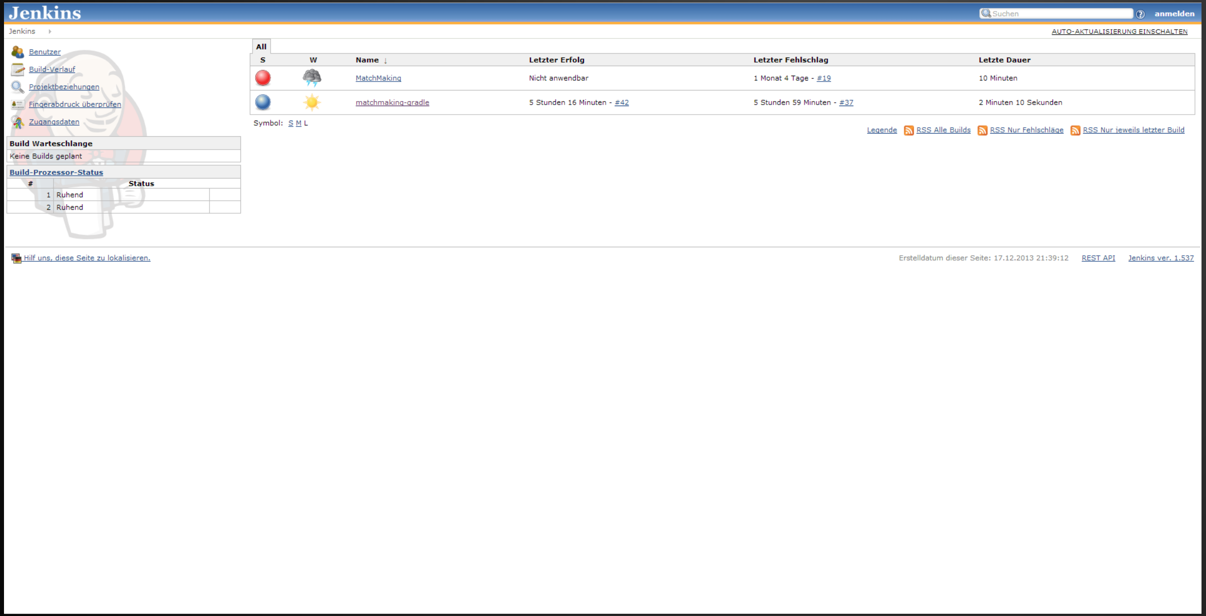
Build	105	Commit	a465d8b (master)
State	Passed	Compare	6b5616358043...a465d8b81cc6
Finished	about 5 hours ago	Author	Leon Schröder
Duration	5 min 27 sec	Committer	Leon Schröder
Message	okay, test-result contains xml, this is what sonar calls report, dont mix up with test-reports		

```
Using worker: worker-linux-2-2.bb.travis-ci.org:travis-linux-6

$ git clone --depth=50 --branch=master git://github.com/schreon/matchmaking.git schreon/matchmaking
$ cd schreon/matchmaking
$ git checkout -qf a465d8b81cc6b417f02ed6b36e6ffd64229762cb
$ jdk_switcher use openjdk6
Switching to OpenJDK6 (java-1.6.0-openjdk-amd64), JAVA_HOME will be set to /usr/lib/jvm/java-6-openjdk-amd64
update-alternatives: error: no alternatives for apt.
$ java -version
java version "1.6.0_27"
OpenJDK Runtime Environment (IcedTea6 1.12.6) (6b27-1.12.6-1ubuntu0.12.04.4)
OpenJDK 64-Bit Server VM (build 20.0-b12, mixed mode)
$ javac -version
javac 1.6.0_27
$ gradle -v
$ uname -a
$ if [ `uname -m` = x86 64 ]; then sudo apt-get update; fi
$ if [ `uname -m` = x86 64 ]; then sudo apt-get install -qq --force-yes libgd2-xpm ia32-libs ia32-libs-multiarch; fi
$ export TERM=dumb
$ wget http://services.gradle.org/distributions/gradle-1.8-bin.zip
$ unzip -qq gradle-1.8-bin.zip
$ export GRADLE HOME=$PWD/gradle-1.8
$ export PATH=$GRADLE HOME/bin:$PATH
$ wget http://dl.google.com/android/android-sdk r22.0.5-linux.tgz
$ tar -zxf android-sdk r22.0.5-linux.tgz
$ export ANDROID HOME=`pwd`/android-sdk-linux
$ export PATH=${PATH}:${ANDROID HOME}/tools:${ANDROID HOME}/platform-tools
$ echo "sdk.dir=$ANDROID HOME" > local.properties
$ echo yes | android update sdk  -a -t build-tools-18.1,tools,platform-tools,extra-android-support,extra-android-m2repository,android-18 --force --no-ui
$ wget http://dl.google.com/android/repository/build-tools r18.1.1-linux.zip
$ mkdir -p $ANDROID HOME/build-tools
$ unzip -qq build-tools r18.1.1-linux.zip -d $ANDROID HOME/build-tools/
$ mv $ANDROID HOME/build-tools/android-4.3.1 $ANDROID HOME/build-tools/18.1.1
$ find $ANDROID HOME/build-tools
$ file $ANDROID HOME/build-tools/18.1.1/aapt
$ ./gradlew assemble
```

Jenkins

An extendable open source continuous integration server.



Jenkins

Suchen | anmelden

Jenkins ›

AUTO-AKTUALISIERUNG EINSCHALTEN

- Benutzer
- Build-Verlauf
- Projektbeziehungen
- Fingerabdruck überprüfen
- Zugangsdaten

Build Warteschlange

Keine Builds geplant

Build-Prozessor-Status

#	Status
1	Ruhend
2	Ruhend

All

S	W	Name ↓	Letzter Erfolg	Letzter Fehlschlag	Letzte Dauer
🔴	⛈	MatchMaking	Nicht anwendbar	1 Monat 4 Tage - #19	10 Minuten
🔵	☀	matchmaking-gradle	5 Stunden 16 Minuten - #42	5 Stunden 59 Minuten - #37	2 Minuten 10 Sekunden

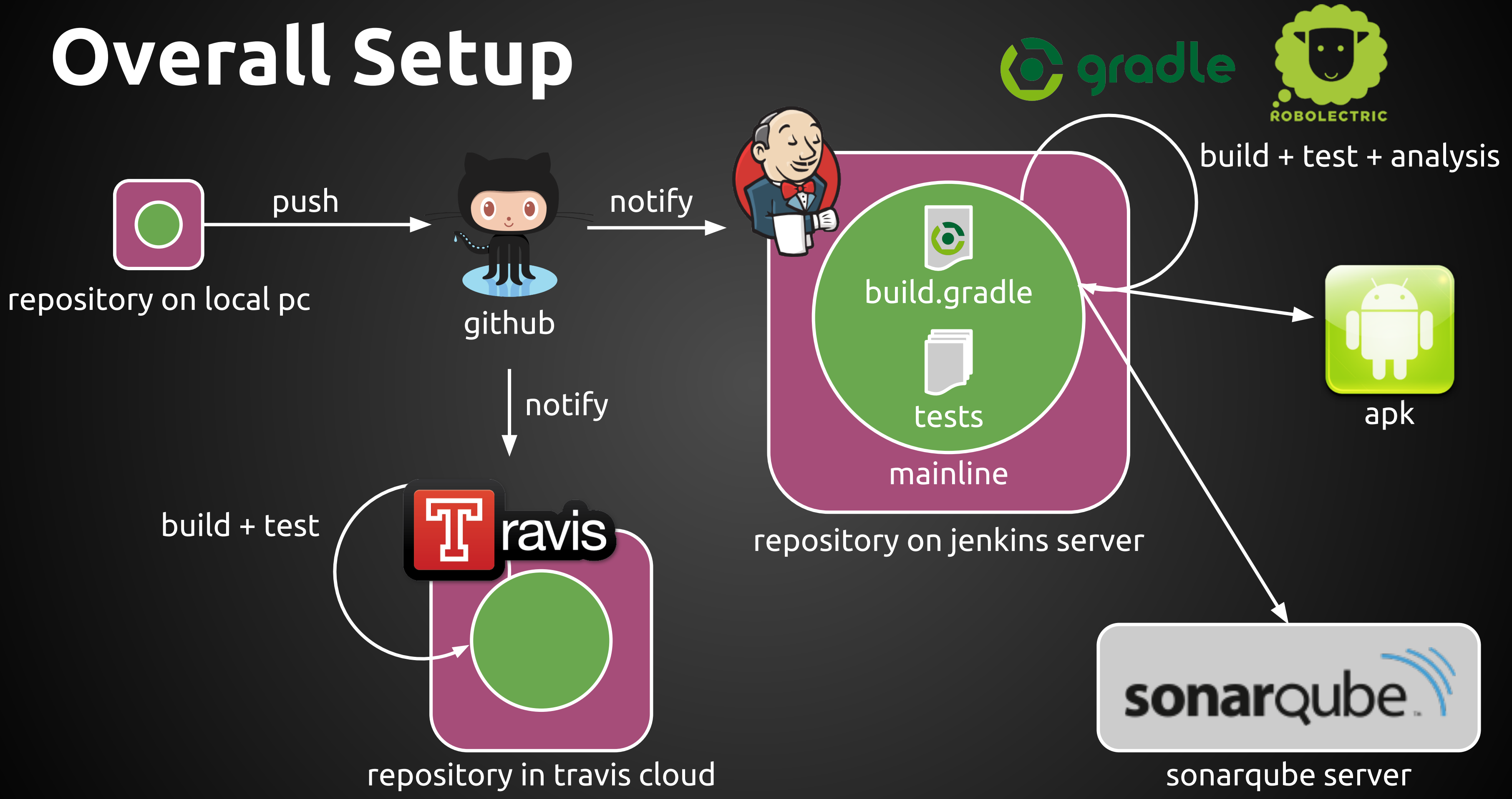
Symbol: S M L

Legende | RSS Alle Builds | RSS Nur Fehlschläge | RSS Nur jeweils letzter Build

Hilf uns, diese Seite zu lokalisieren.

Erstelldatum dieser Seite: 17.12.2013 21:39:12 | REST API | Jenkins ver. 1.537

Overall Setup



Thank you for listening!

Have a look at the following github repository with our sample project and instructions on how to set all this up:

https://github.com/schreon/android-quality-template

(work in progress!)