Gradle 2.0 and beyond

latest efforts, current status & roadmap

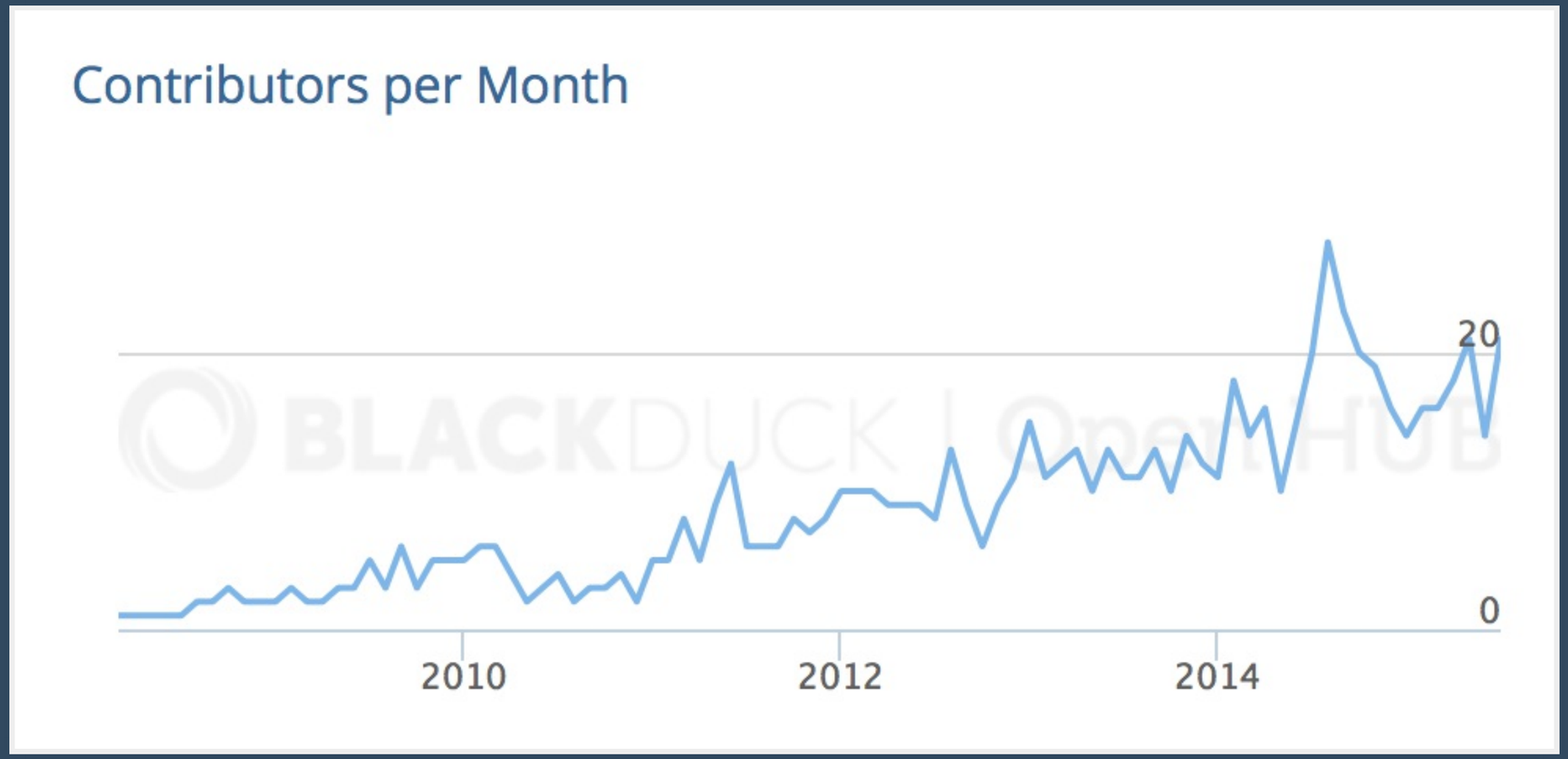
René Gröschke

- Principal Engineer @ Gradle Inc.
- @breskeby
- rene@gradle.com
- breskeby

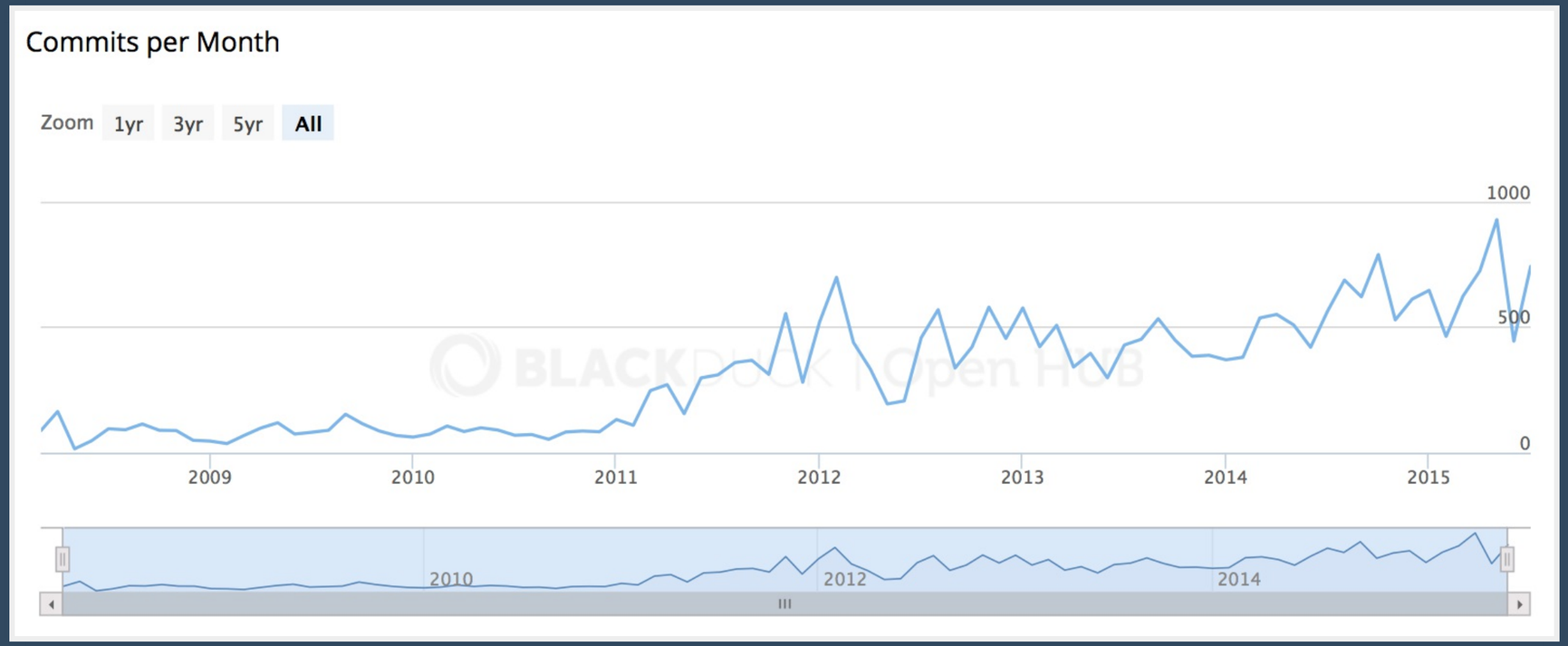
Gradle in a nutshell

- completely open source
- apache2 licensed
- driven by Gradle Inc.

Gradle in a nutshell



Gradle in a nutshell



Gradle in a nutshell

Language Breakdown

Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines	Total Percentage
Groovy	267,462	53,482	16.7%	56,857	377,801	44.2%
Java	225,043	119,744	34.7%	55,808	400,595	46.9%
XML	35,579	3,833	9.7%	929	40,341	4.7%
C++	14,409	5,377	27.2%	2,276	22,062	2.6%
JavaScript	4,804	283	5.6%	187	5,274	0.6%
CSS	1,732	137	7.3%	243	2,112	0.2%
C	822	1,705	67.5%	302	2,829	0.3%
HTML	717	27	3.6%	67	811	0.1%
Scala	500	238	32.2%	169	907	0.1%
XSL Transformation	414	123	22.9%	70	607	0.1%
Python	398	63	13.7%	221	682	0.1%
shell script	256	45	15.0%	43	344	0.0%
DOS batch script	50	16	24.2%	24	90	0.0%
Objective-C	49	0	0.0%	25	74	0.0%
CoffeeScript	20	8	28.6%	8	36	0.0%
Assembly	17	0	0.0%	1	18	0.0%
Structured Basic	6	0	0.0%	0	6	0.0%
Totals	552,278	185,081		117,230	854,589	

Gradle in a nutshell

A simple java project

```
apply plugin:"java"

version = file("version.txt").text

repositories {
    jcenter()
}

dependencies {
    testCompile "junit:junit:4.+"
}

task printVersion << { println "We're using - version '$version'!" }
```

Gradle 2.0

Released 1st July 2014

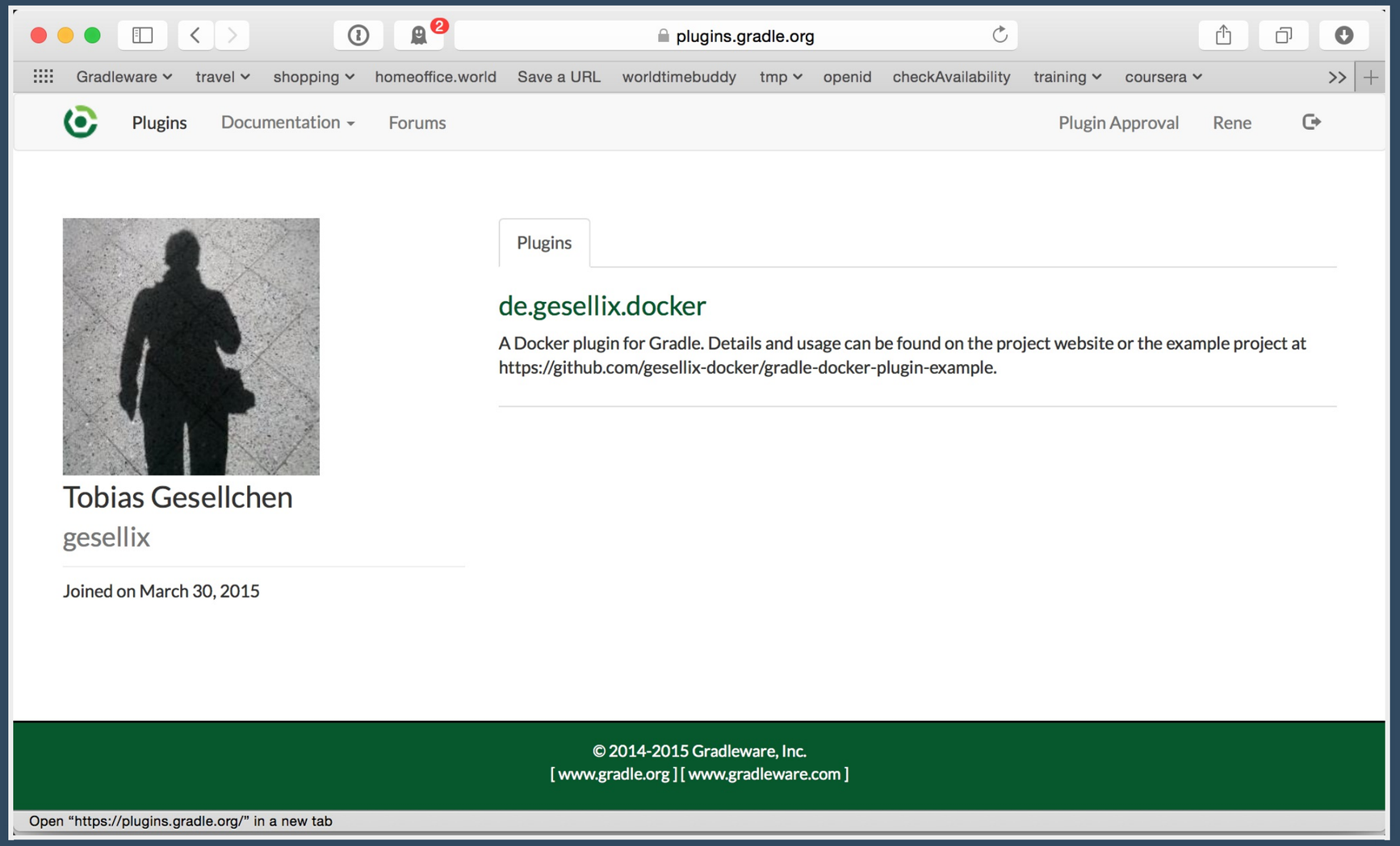
Gradle 2.7

Released 14th September 2015

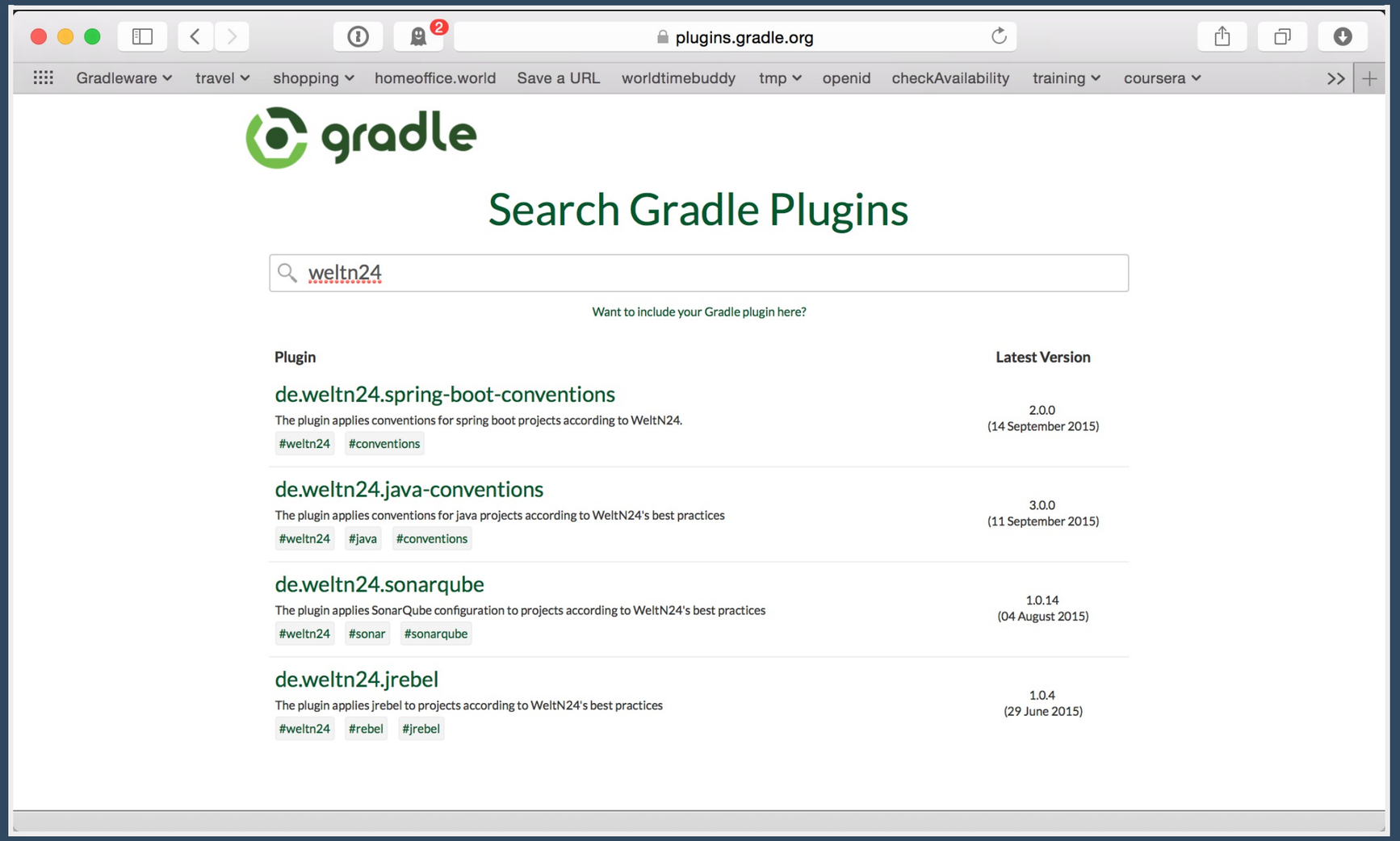
Let’s take a closer look on

- Plugin Portal
- Play Support
- Gradle TestKit
- Even better Dependency Management
- Native Build Support improvements
- Enhanced Tooling API

Plugin Portal



Plugin Portal II



Play Support

DEMO

Continous Mode

```
> gralde build -t
```

Gradle TestKit

Functional testing of your build logic

```
def setup() {
    buildFile = testProjectDir.newFile('build.gradle')
}

def "hello world task prints hello world"() {
    given:
    buildFile << """
        task helloWorld {
            doLast {
                println 'Hello world!'
            }
        }
    """

    when:
    def result = GradleRunner.create()
        .withProjectDir(testProjectDir.root)
        .withArguments('helloWorld')
        .build()

    then:
    result.standardOutput.contains('Hello world!')
```

Dependency Management

Dependency Resolve Rules

Forcing consistent version for a group of libraries

```
configurations.all {
    resolutionStrategy.eachDependency { DependencyResolveDetails details
        if (details.requested.group == 'org.gradle') {
            details.useVersion '2.7'
        }
    }
}
```

Dependency Resolve Rules

Using a custom versioning scheme

```
configurations.all {
    resolutionStrategy {
        eachDependency { DependencyResolveDetails d ->
            if (d.requested.version == 'default') {
                def version = findDefaultVersion(d.requested.group,
                                                 d.requested.name)
                d.useVersion version
            }
        }
    }
}

Object findDefaultVersion(String group, String name) {
    // some custom logic that resolves the default
    // version into a specific version
    "1.0"
}
```

Dependency Resolve Rules

Changing dependency group and/or name at the resolution

```
configurations.all {
    resolutionStrategy {
        eachDependency { DependencyResolveDetails details ->
            if (details.requested.name == 'groovy-all') {
                //prefer 'groovy' over 'groovy-all':
                 details.useTarget(group: details.requested.group,
                                   name: 'groovy',
                                   version: details.requested.version)
            }
            if (details.requested.name == 'log4j') {
                //prefer 'log4j-over-slf4j' over 'log4j',
                details.useTarget "org.slf4j:log4j-over-slf4j:1.7.10"
            }
        }
    }
}
```

Component Selection Rules

```
dependencies {
    compile 'org.slf4j:slf4j-api:+'
    testCompile 'junit:junit:4.11'
}

configurations {
    all {
        resolutionStrategy {
            componentSelection {
                withModule("org.slf4j:slf4j-api") { selection ->
                    if(selection.candidate.version == "1.7.10") {
                        selection.reject("known buggy version")
                    }
                }
            }
        }
    }
}
```

Artifact Query Api

```
task resolveMavenPomFiles << {
    def componentIds = configurations.compile.incoming.resolutionResult.c

    def result = dependencies.createArtifactResolutionQuery()
         .forComponents(componentIds)
         .withArtifacts(MavenModule, MavenPomArtifact)
         .execute()

    for(component in result.resolvedComponents) {
        component.getArtifacts(MavenPomArtifact).each {
            def pom = new XmlSlurper().parse(it.file)
            println pom.url
        }
    }
}
```

Dependency Substitution

Allows *elastic* dependencies

```
configurations.all {
    resolutionStrategy.dependencySubstitution {
        substitute project(":api") with module("org.utils:api:1.3")
    }
}
```

Buildship

- Eclipse plugin developed from scratch by Gradle Inc.
- Part of the eclipse foundation
- We just left incubator status last week
- Shipped as part of the mars.1 release (25.09.2015)

Buildship

Demo

Current focus

- New Gradle model
- Dependency management
- Better domain modelling

Dependency management

Dependency management

- **to deal with dependencies we have:**
  - group, name, version
  - classifier, custom ivy configurations

Dependency management

- **to deal with dependencies we have:**
  - group, name, version
  - classifier, custom ivy configurations
- **but we need to deal with:**
  - java, groovy, scala versions
  - android, native target platforms, all kind of javascript

Dependency management

Dependency management

Allow variant aware dependency management

Dependency management

Allow variant aware dependency management

Support arbitrary dimensions + custom metadata

Better domain modelling

Domain modelling is Gradle's strength.
We want it to be even better.

Better domain modelling

Domain modelling is Gradle's strength.
We want it to be even better.

Stronger modeling

↪ The JAR is not the task that creates it.

Cleaner modeling

↪ Avoid mixing execution concerns into the data model.

Collaborative modeling

↪ I know how to do something to JARs.

Comprehensible models

↪ Who is contributing to the contents of this JAR?

A new Gradle model

The current model

configuration → execution

The current model

configuration ➞ execution

- **configuration:**
  - input = build logic
  - output = build model

The current model

configuration ➞ execution

- **configuration:**
  - input = build logic
  - output = build model
- **execution:**
  - input = build model
  - output = build artifacts

Limitations of the current model

- implementation of declarative build api is hard
  - done in the imperative way
- eagerness
- lazyness
- hooks
- scaling

Too hard

For build engineers and build users.

We can do better.

The new Gradle model

A new approach to the configuration phase.

Really, the same solution for the "execution phase" applied to configuration.

A graph of dependent functions

An interpretable data model

The new Gradle model I

Enter RuleSource

```
class PersonRules extends RuleSource {
  @Model void person(Person p) {}

  @Mutate void setFirstName(Person p) {
    p.firstName = "John"
  }

  @Mutate void createHelloTask(ModelMap<Task> tasks, Person p) {
    tasks.create("hello") {
      doLast {
        println "Hello $p.firstName $p.lastName!"
      }
    }
  }
}
```

The new Gradle model II

the build script

```
apply plugin: PersonRules

model {
  person {
    lastName = "Smith"
  }
}
```

The new Gradle model III

Android experimental plugin

```
model {
    android {
        compileSdkVersion = 22
        buildToolsVersion = "22.0.1"

        defaultConfig.with {
            applicationId =  "com.example.user.myapplication"
            minSdkVersion.apiLevel = 15
            targetSdkVersion.apiLevel = 22
            versionCode = 1
            versionName = "1.0"
        }
    }
}
```

The new Gradle model IV

as an enabler for

- build much faster and more memory efficient
- just configure what is required
- allow fundamental parallization
- provide better diagnostics
- reuse cached configuration
- …

Gradle 3.0

other future plans

- jigsaw support
- shared distributed cache
- next level native build support
- more daemon utilisation
- continued tooling improvements

Links and pointers

- https://docs.gradle.org/current/userguide/new_model.html
- http://gradle.org/roadmap
- http://discuss.gradle.org/c/roadmap

Q & A

thanks!

TODO

- pdf export → decktape.js
- header/footer
- make asciidoctor offline available