DC Continuous Delivery Meetup

Testing the build with TestKit

Benjamin Muschko

2016

About to ship code to production…

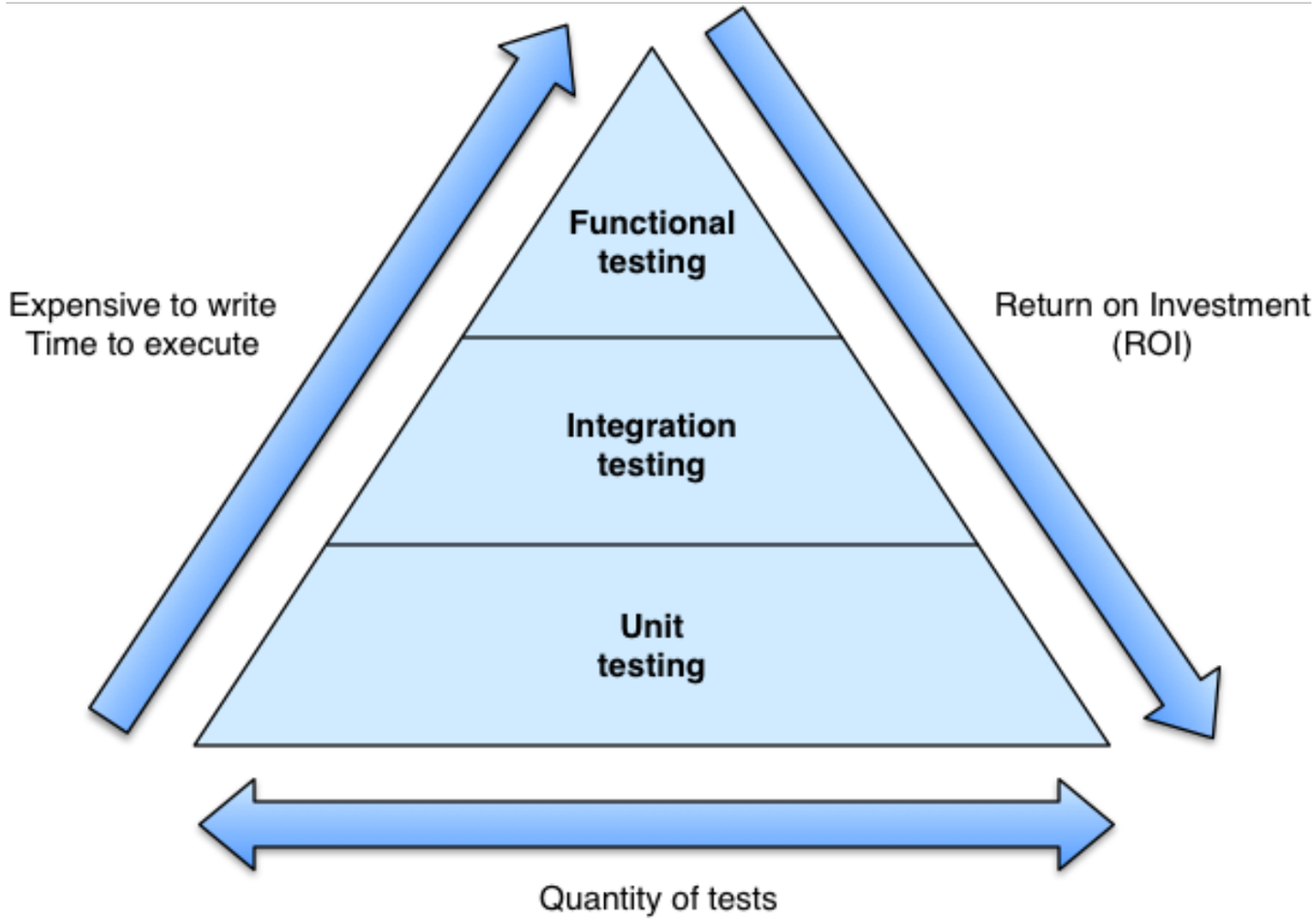
Gradle

…but the build logic contained bugs

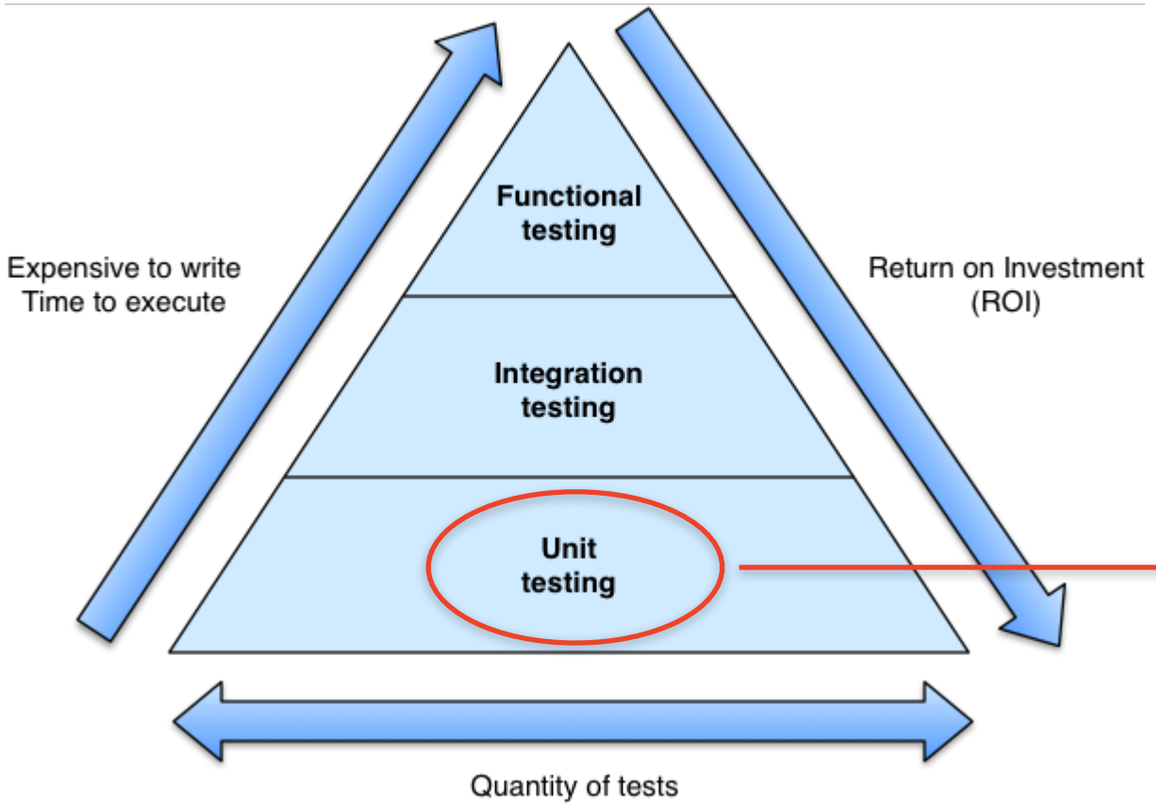
Why not test build logic?

Testing build logic

Build logic needs to be testable on multiple levels.



Writing unit tests in Gradle



Testing a single class

Class under test does not use Gradle API

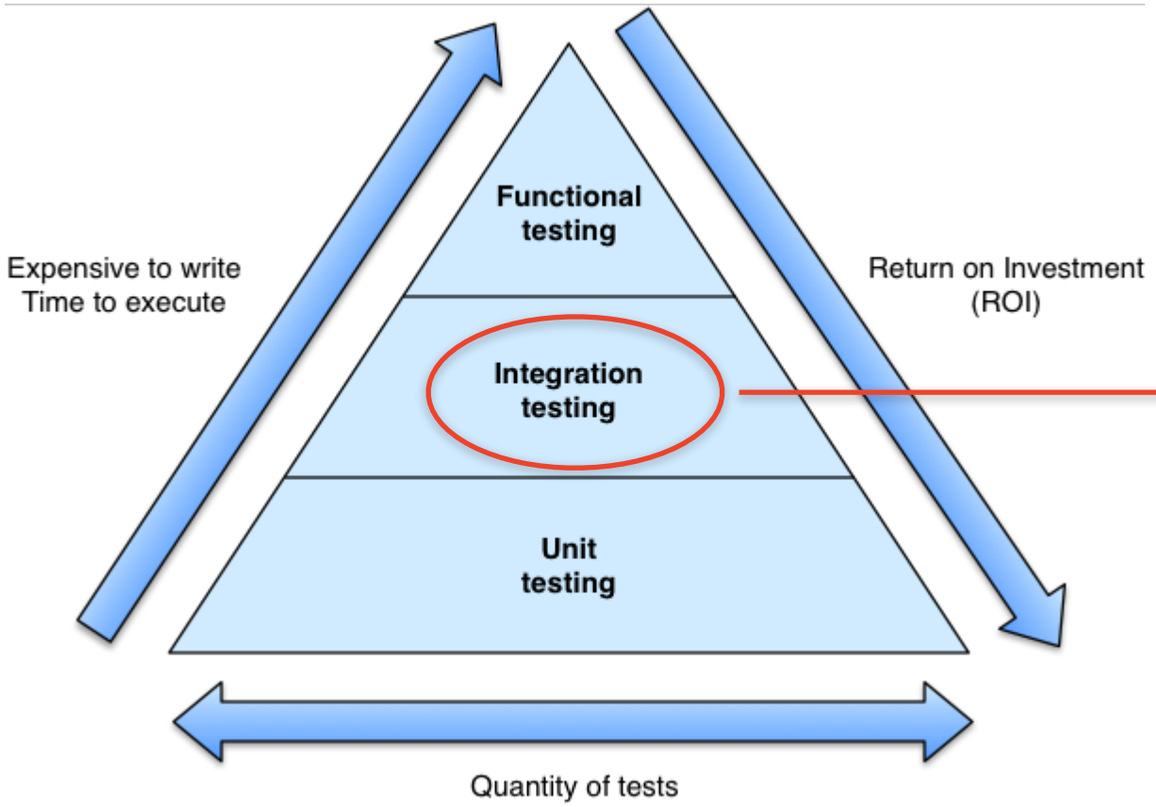
Test logic in isolation

Gradle

Example: Writing a unit test with Spock

```
package com.bmuschko.gradle.docker.tasks.image

import spock.lang.Specification

import static com.bmuschko.gradle.docker.tasks.image.Dockerfile.*

class DockerfileTest extends Specification {
    def "Instruction String representation is built correctly"() {
        expect:
        instructionInstance.keyword == keyword
        instructionInstance.build() == builtInstruction

        where:
        instructionInstance                          | keyword   | builtInstruction
        new FromInstruction('ubuntu:14.04')          | 'FROM'    | 'FROM ubuntu:14.04'
        new FromInstruction({ 'ubuntu:14.04' })      | 'FROM'    | 'FROM ubuntu:14.04'
        ...
    }
}
```

Writing integration tests in Gradle



Class(es) under test use Gradle API

Usually interacts with `Project` instance

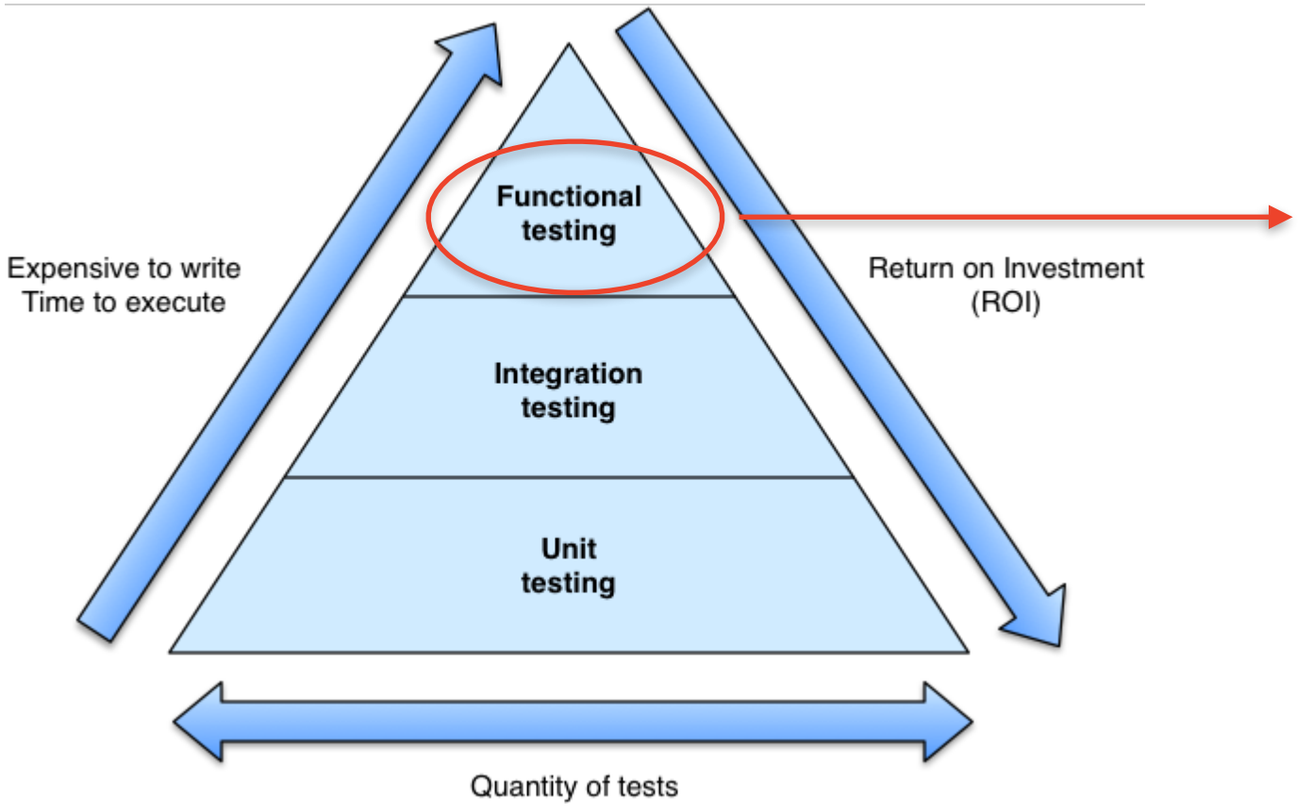
Does not execute a full build script

Gradle

Example: Writing an integration test using ProjectBuilder

```
import org.gradle.api.Project
import org.gradle.testfixtures.ProjectBuilder

class DockerJavaApplicationPluginIntegrationTest extends Specification {
    @Rule TemporaryFolder temporaryFolder = new TemporaryFolder()
    Project project

    def setup() {
        project = ProjectBuilder.builder().withProjectDir(temporaryFolder.root).build()
    }

    def "Creates tasks out-of-the-box when application plugin is applied"() {
        when:
        project.apply(plugin: DockerJavaApplicationPlugin)
        project.apply(plugin: 'application')

        then:
        project.tasks.findByName(DockerJavaApplicationPlugin
                                    .COPY_DIST_RESOURCES_TASK_NAME)
        project.tasks.findByName(DockerJavaApplicationPlugin.DOCKERFILE_TASK_NAME)
        project.tasks.findByName(DockerJavaApplicationPlugin.BUILD_IMAGE_TASK_NAME)
        project.tasks.findByName(DockerJavaApplicationPlugin.PUSH_IMAGE_TASK_NAME)
    }
}
```

Writing functional tests in Gradle



Executes build script similar to end user

Examines build outcome and output

Isolated test environment

Example: Writing a functional test with TestKit

```
def "can successfully create Dockerfile"() {
    given:
    buildFile << """
        import com.bmuschko.gradle.docker.tasks.image.Dockerfile

        task dockerfile(type: Dockerfile) {
            from 'ubuntu:14.04'
            maintainer 'Benjamin Muschko "benjamin.muschko@gmail.com"'
        }
    """

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

    then:
    result.task(":dockerfile").outcome == SUCCESS
    testProjectDir.file('Dockerfile').exists()
}
```

What's the Gradle TestKit?

Functional testing support in Gradle core.

- Agnostic of test framework
- Assertions made based on build output, build logging or executed tasks + their result
- Uses Tooling API as test execution harness

Gradle 2.6

TestKit usage

Declaring TestKit dependency

```
dependencies {
    testCompile gradleTestKit()
}
```

Declaring dependency on test framework

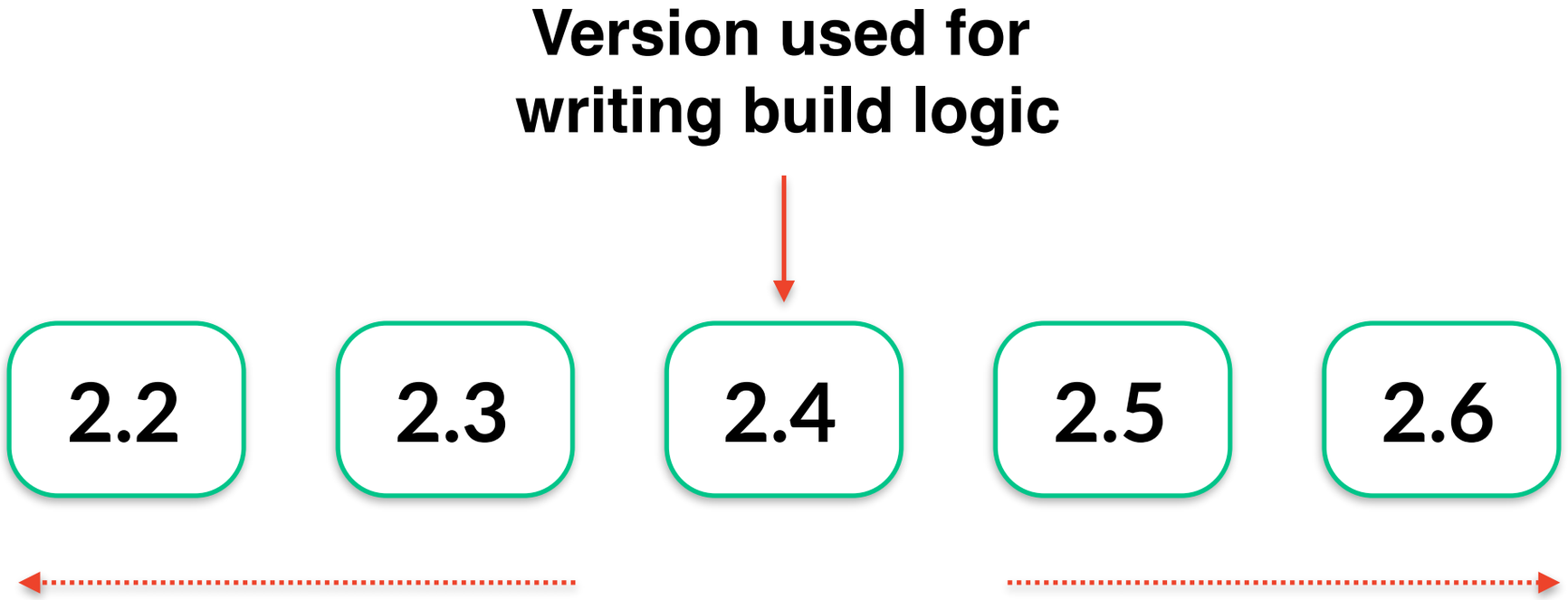
```
dependencies {
    testCompile 'org.spockframework:spock-core:1.0-groovy-2.4'
}
```

DEMO

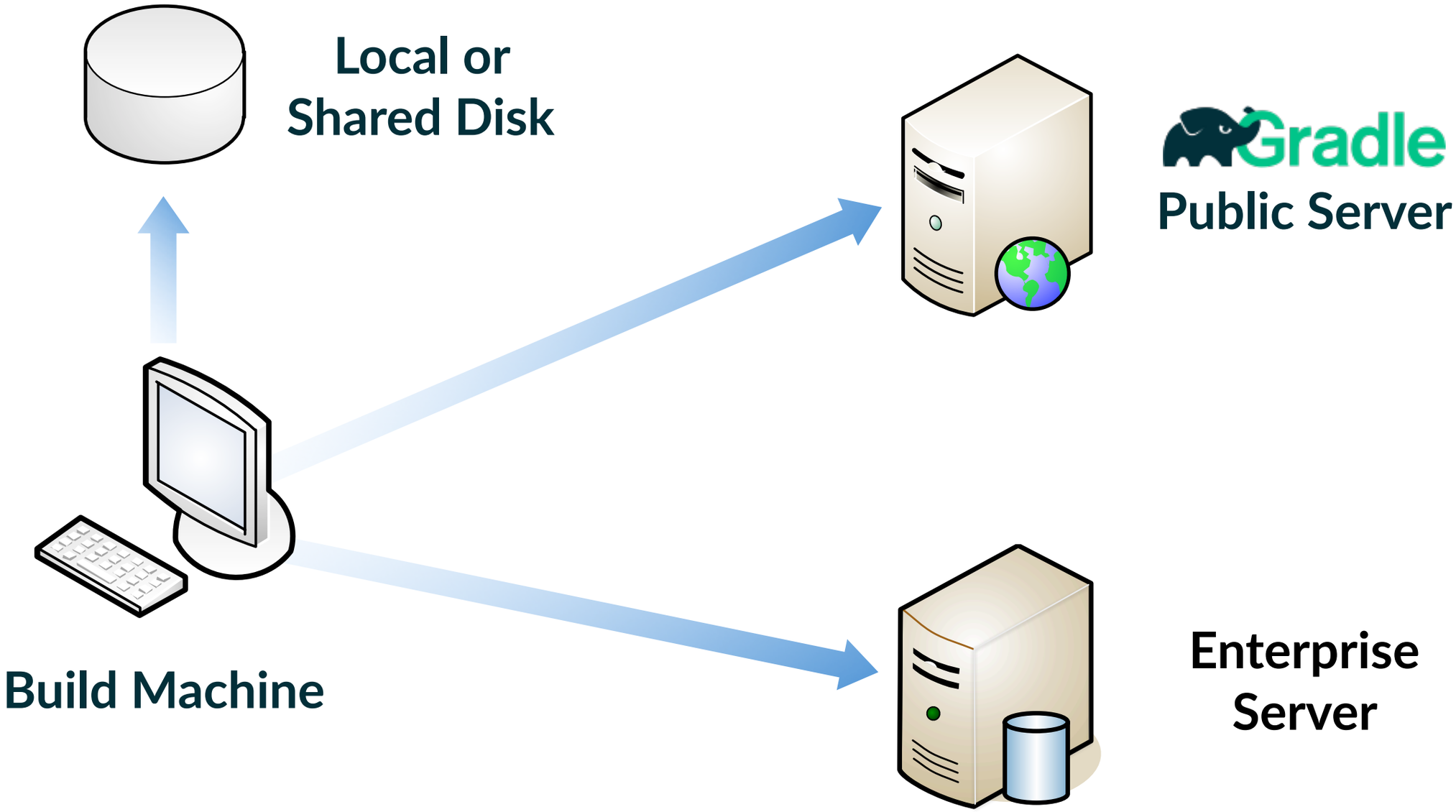
Functional testing of a build script

Cross-version compatibility tests

Forward and backward compatibility independent of Gradle version used to build logic.



Retrieving Gradle distributions



Specifying a Gradle version

Using a Gradle version available on server

```
GradleRunner.withGradleVersion(String)
```

Using a Gradle installation on disk

```
GradleRunner.withGradleInstallation(File)
```

Using a Gradle distribution via URL

```
GradleRunner.withGradleDistribution(URI)
```

DEMO

Cross-version compatibility testing

Testing a plugin

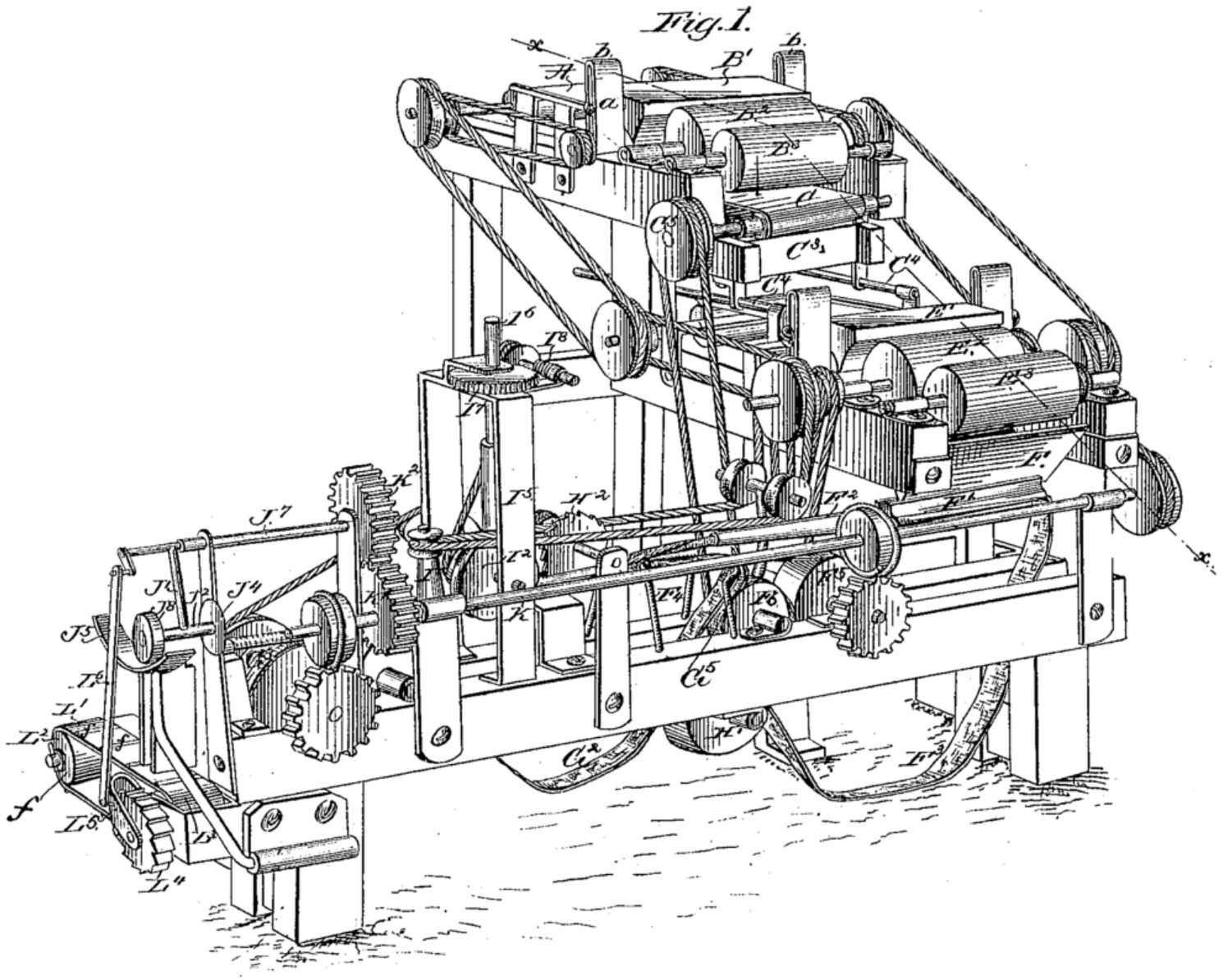
Requires additional work!

- Tooling API runs in separate process
- Does not share classpath and classloaders as test process
- Requires injection of code under test

DEMO

Manually injecting the plugin classpath

That’s some complex machinery!



Simplifying setup with plugin development plugin

Apply sensible default and conventions…

- Declaration of `gradleApi()` and `gradleTestKit()`
- Generates plugin classpath manifest file
- Plugin classpath injection needs to be declared for `GradleRunner` instance explicitly
- Requires use of plugin DSL

Gradle 2.13

Automatic injection of plugin code

Applying the Java Gradle plugin dev. plugin

```
apply plugin: 'java-gradle-plugin'
```

Injecting classpath into `GradleRunner`

```
GradleRunner.create()
    .withPluginClasspath()
    .build()
```

DEMO

Testing a plugin with TestKit

Configurable conventions

Source set containing code under test

`sourceSets.main`

Source set used for injecting the plugin classpath

`sourceSets.test`

Reconfigurable with the help of the class `GradlePluginDevelopmentExtension`

DEMO

Configuring a dedicated test source set

Debugging test execution

Setting system property for ad-hoc testing

```
-Dorg.gradle.testkit.debug=true
```

Enabling debugging programmatically

```
GradleRunner.withDebug(true)
```

DEMO

Debugging tests from the IDE

Feature limitations

Support for features determined by version of Tooling API used to execute test.

More info:

https://docs.gradle.org/current/userguide/test_kit.html

Future enhancements for TestKit

- Convenience test fixtures
- Hooking into IDE plugins
- Integration with JaCoCo plugin

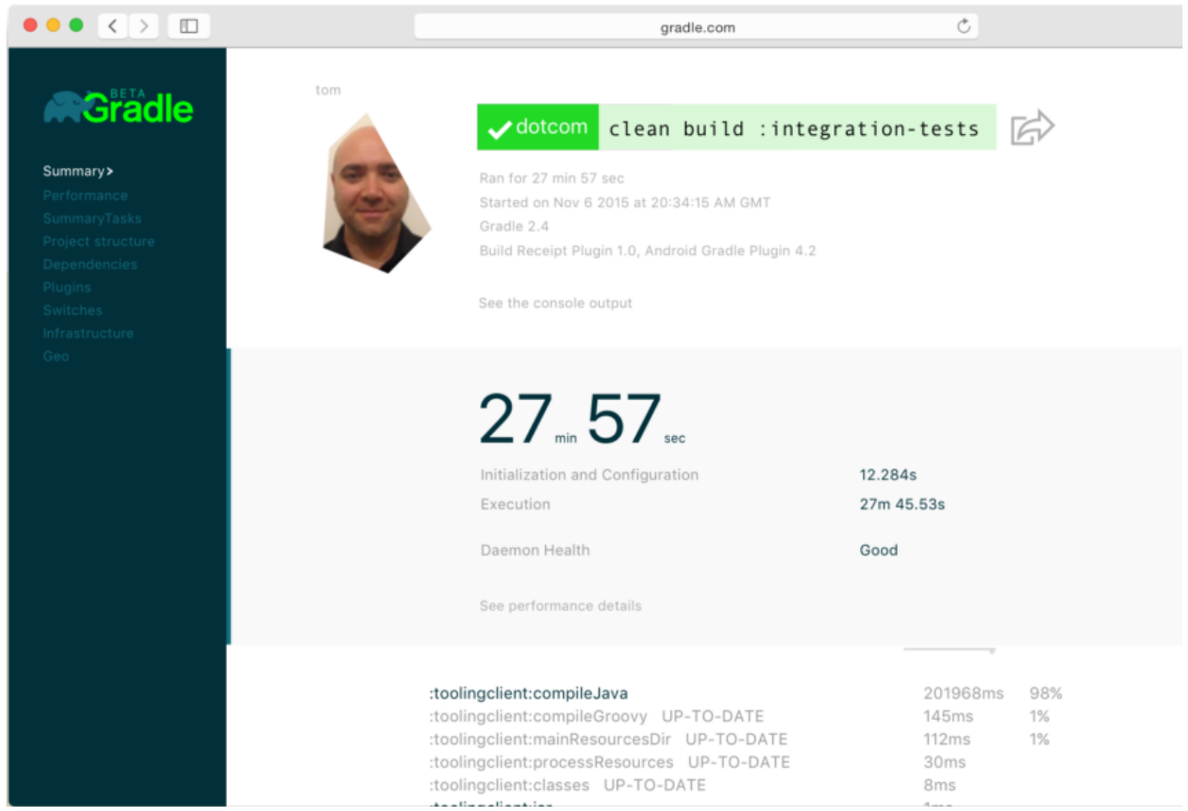
More info:

https://github.com/gradle/gradle/blob/master/design-docs/testing-toolkit.md

Introducing Gradle Cloud Services

The first service, **Gradle Build Scans**, is now available.

- Insights into your build
- View and share via URL
- Debug, optimize and refine
- Analyze *all* of your builds



Gradle

Who Are We

Motto: Build Happiness

Mission: To revolutionize the way software is built and shipped. We exist to end once-and-for-all the worst things about big software and restore the reason you got into coding in the first place.

We're Hiring: Gradle is hiring front-end, back-end, and core software engineers. Visit **gradle.org/jobs** to apply

Gradle

Thank You!

Please ask questions…

https://www.github.com/bmuschko

@bmuschko

Gradle

Learn more at www.gradle.org