

# Building a Continuous Delivery Pipeline with



and  
**Jenkins**

Benjamin Muschko  
Principal Engineer, Gradleware  
@bmuschko

Releases don't have to be painful

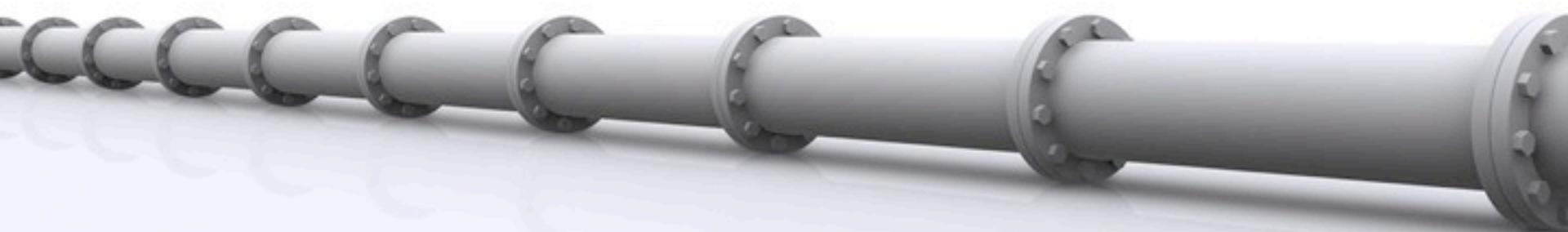


# Continuous Delivery

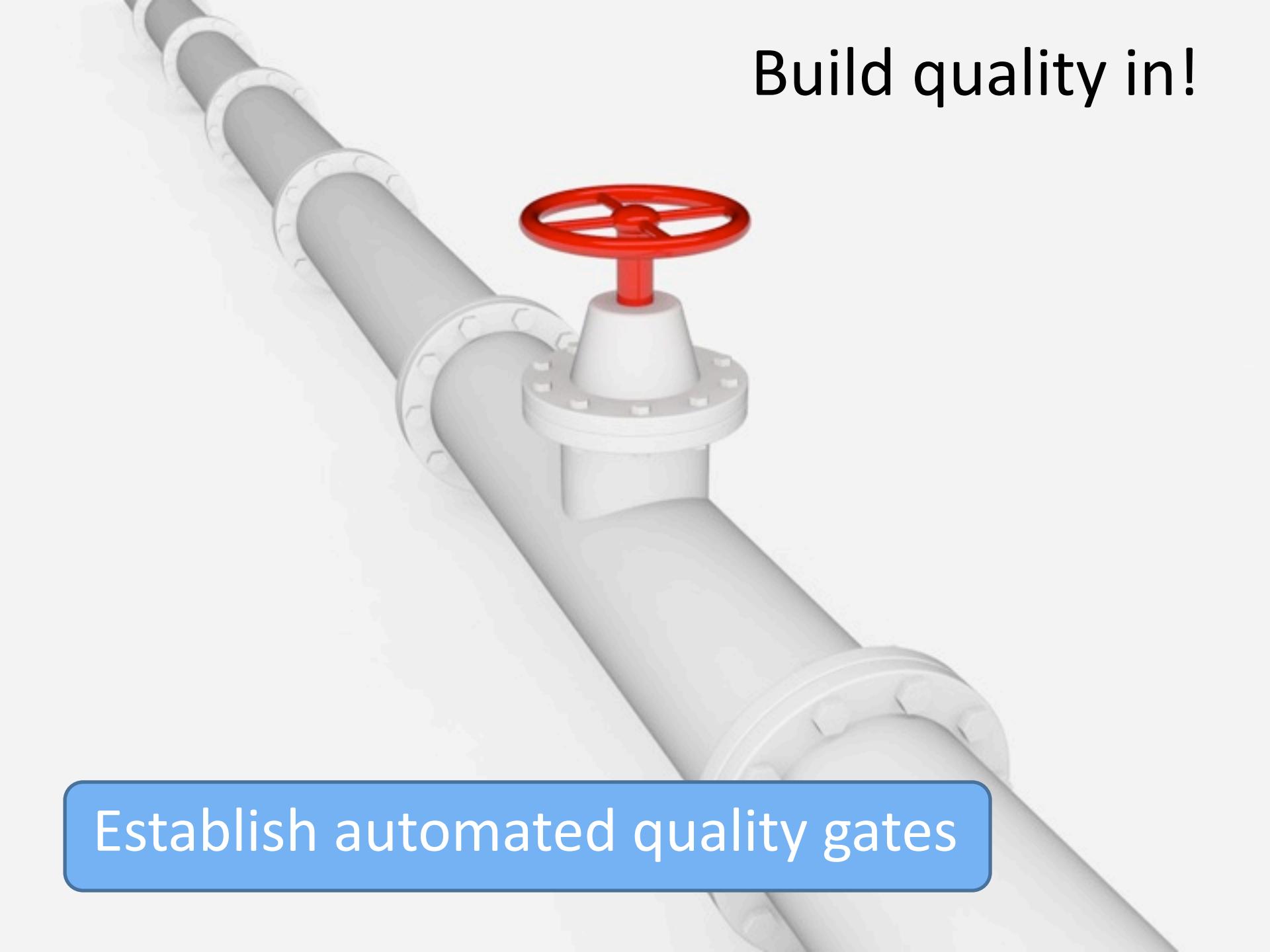


Deliver software fast and frequently

# Build pipeline



Automated manifestation of delivery process



# Build quality in!

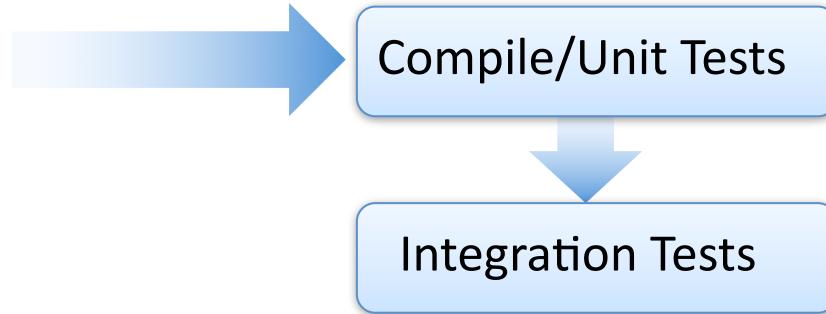
Establish automated quality gates

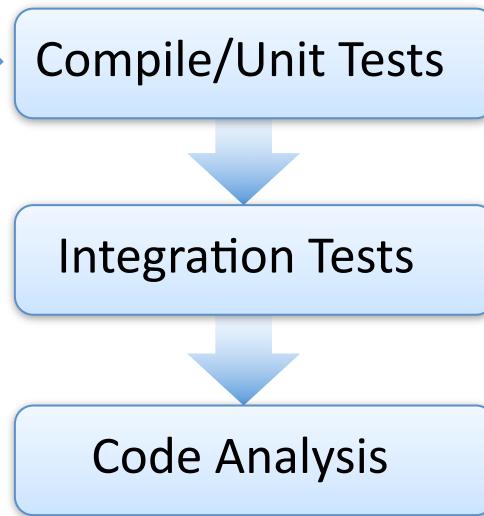


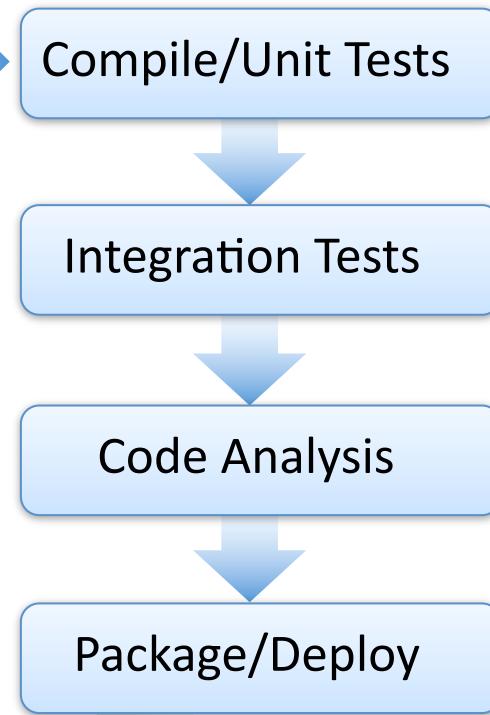
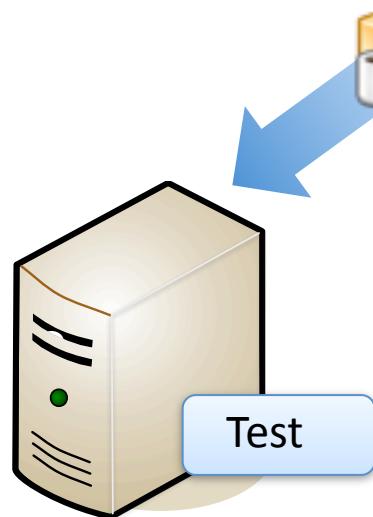
**git**

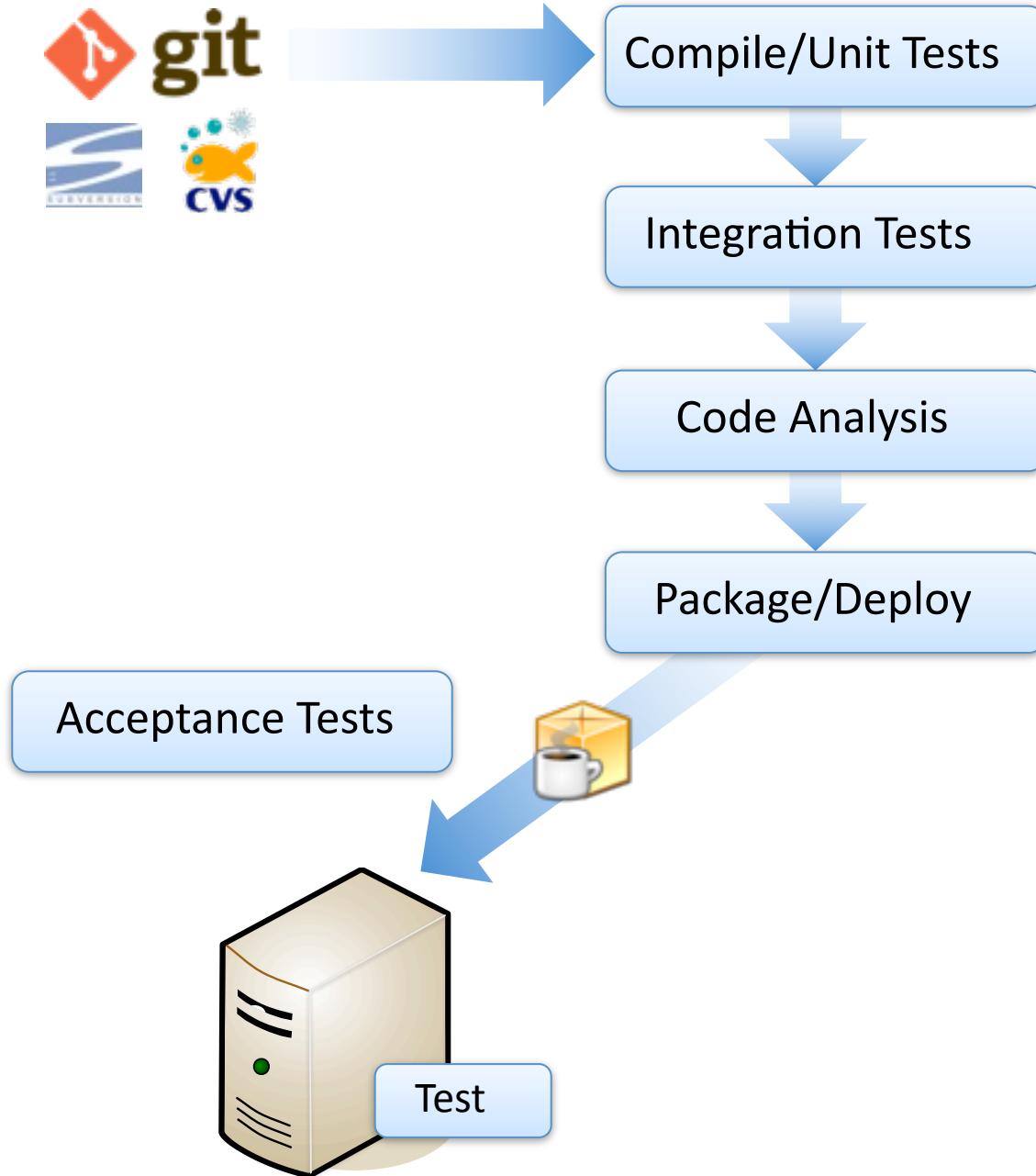


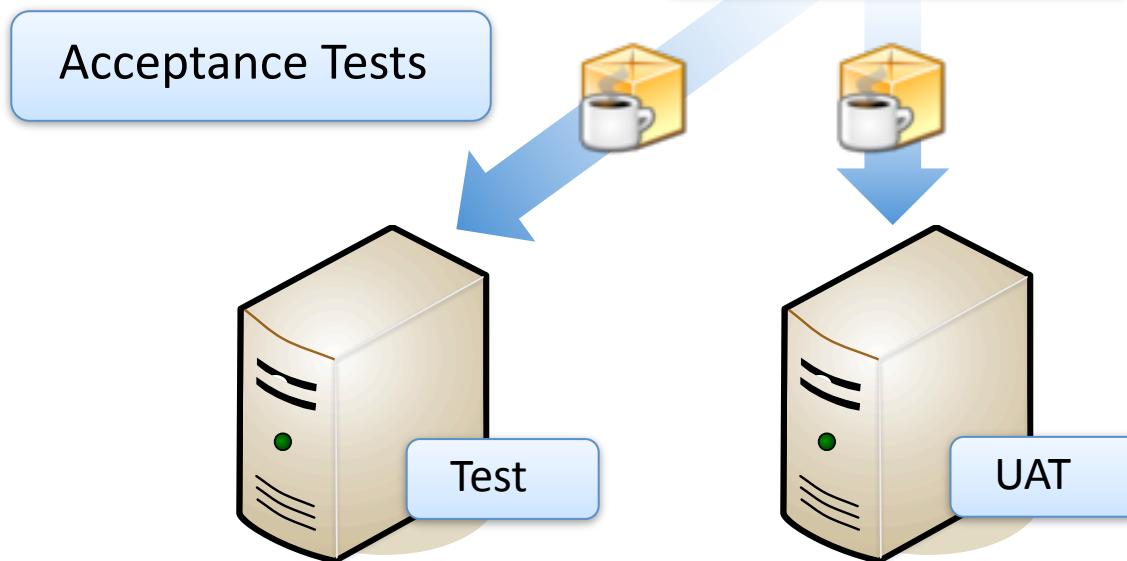
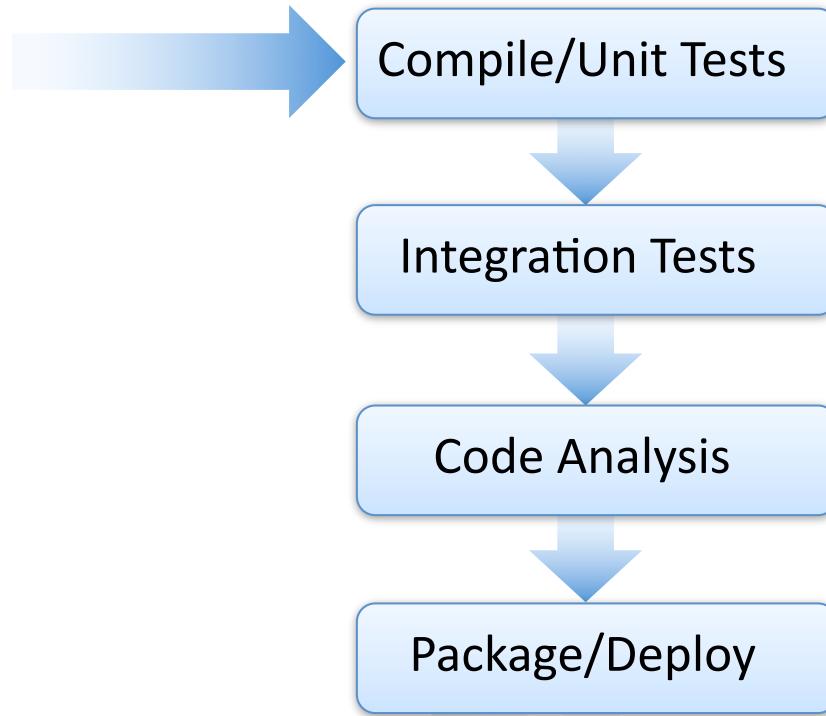
Compile/Unit Tests

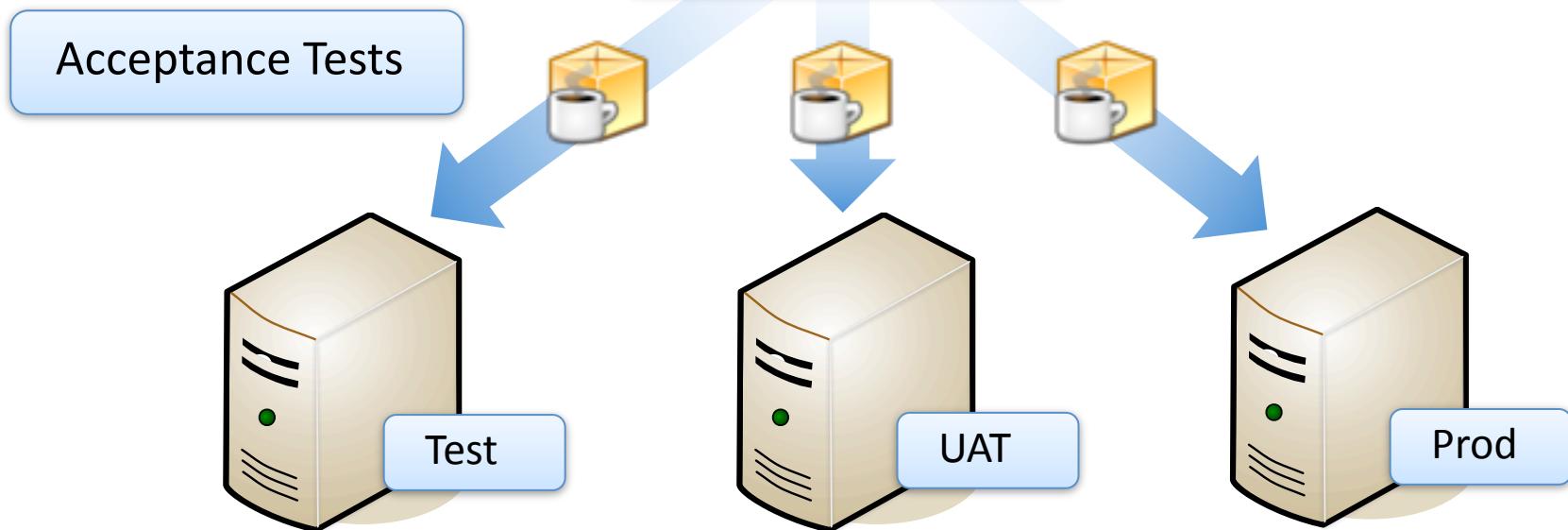
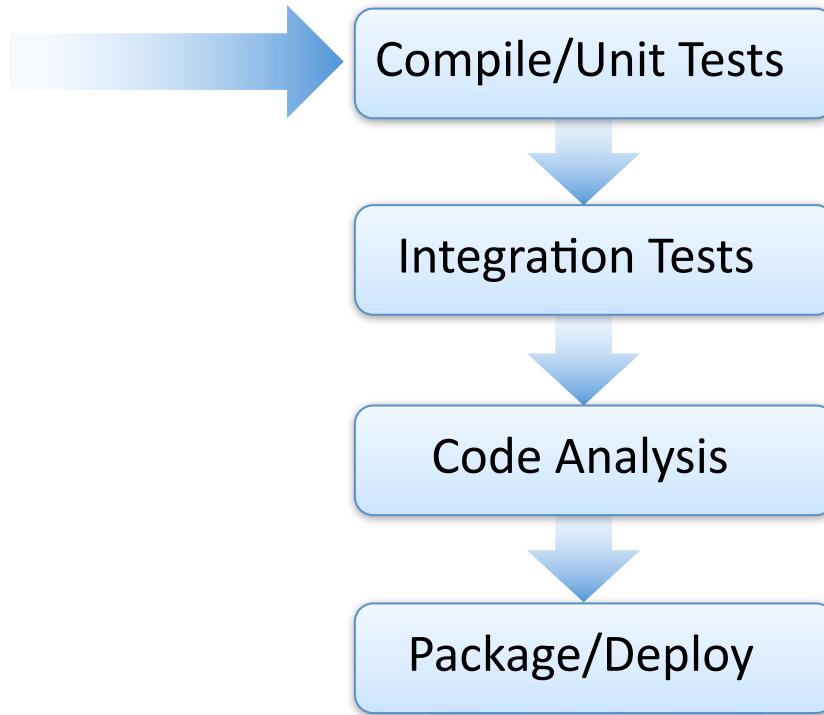




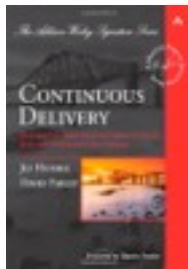












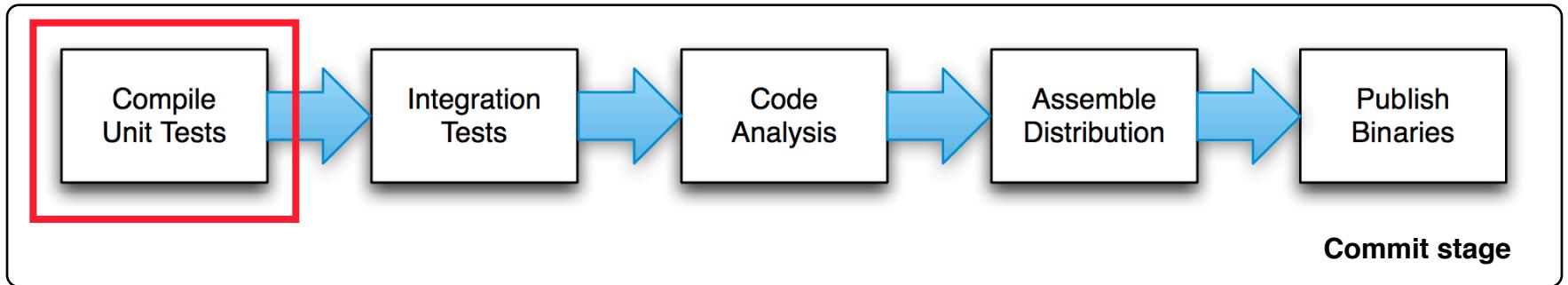
!

But  
how?



Why  gradle ?

# Commit stage: Compile/unit tests



Commit stage

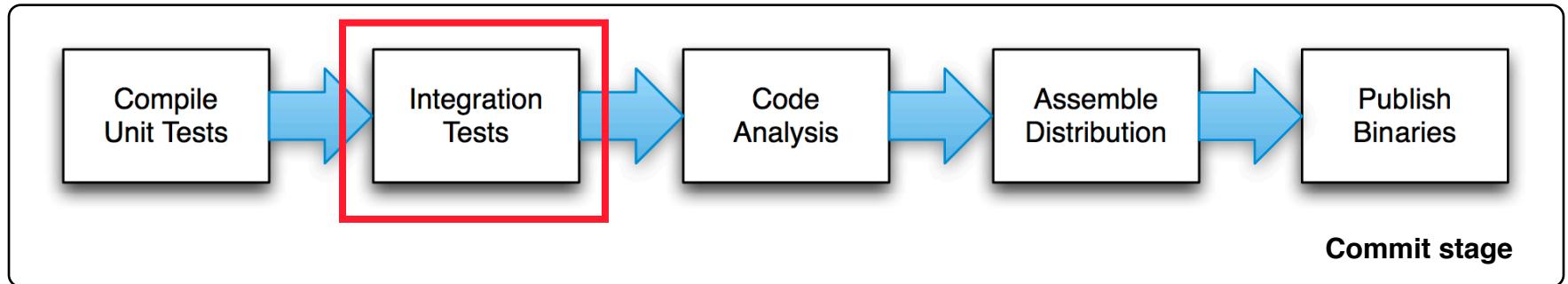
Rapid feedback (< 5 mins)

Run on every VCS check-in

Priority: fix broken build

```
String sql = "select * from stores";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(sql);
if (resultSet.next()) {
    result = true;
    setstoreId(resultSet.getInt("storeId"));
    storeDescription = resultSet.getString("storeDescription");
    storeTypeId = resultSet.getInt("storeTypeId");
}
```

# Commit stage: Integration tests



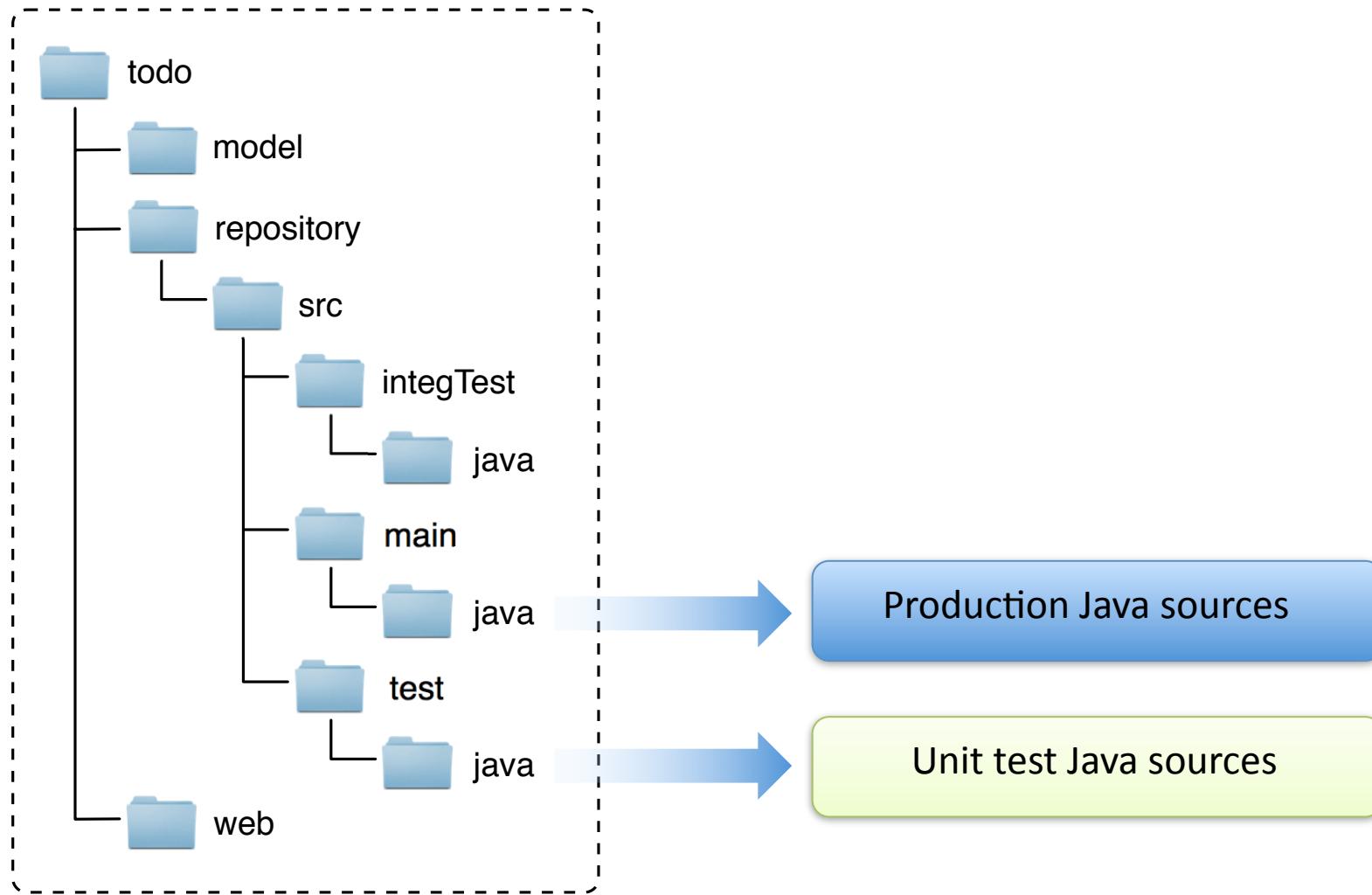
Long running tests

Require environment setup

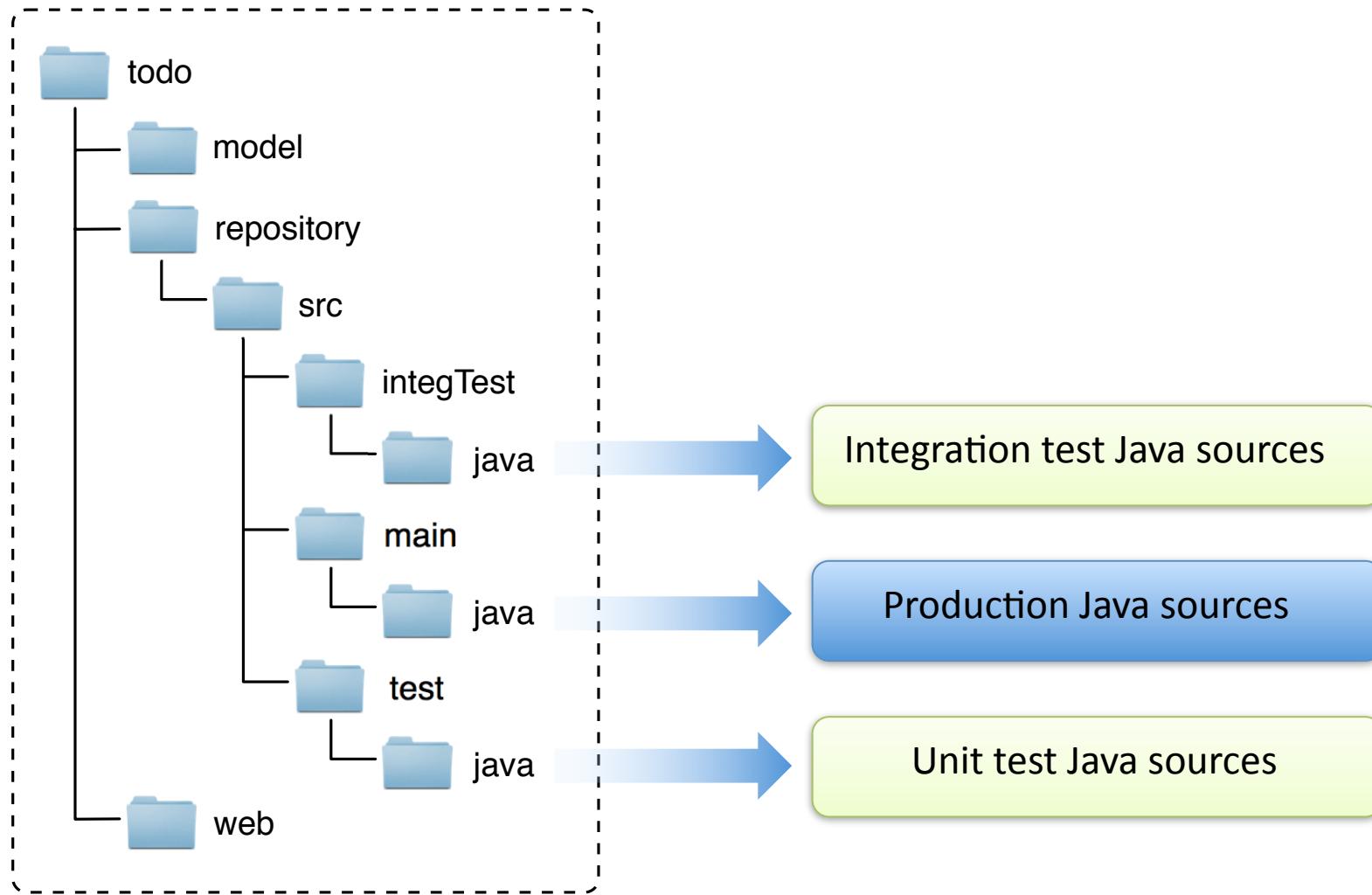
Hard to maintain



# Separate tests in project layout



# Separate tests in project layout



# Separate tests with SourceSets

```
sourceSets {  
    integrationTest {  
        java.srcDir file('src/integTest/java')  
        resources.srcDir file('src/integTest/resources')  
        compileClasspath = sourceSets.main.output + configurations.testRuntime  
        runtimeClasspath = output + compileClasspath  
    }  
}  
  
task integrationTest(type: Test) {  
    description = 'Runs the integration tests.'  
    group = 'verification'  
    testClassesDir = sourceSets.integrationTest.output.classesDir  
    classpath = sourceSets.integrationTest.runtimeClasspath  
    testResultsDir = file("$testResultsDir/integration")  
}
```

# Separate tests with SourceSets

```
sourceSets {  
    integrationTest {  
        java.srcDir file('src/integTest/java')  
        resources.srcDir file('src/integTest/resources')  
        compileClasspath = sourceSets.main.output + configurations.testRuntime  
        runtimeClasspath = output + compileClasspath  
    }  
}  
  
task integrationTest(type: Test) {  
    description = 'Runs the integration tests.'  
    group = 'verification'  
    testClassesDir = sourceSets.integrationTest.output.classesDir  
    classpath = sourceSets.integrationTest.runtimeClasspath  
    testResultsDir = file("$testResultsDir/integration")  
}
```

Set source and resources directory

# Separate tests with SourceSets

```
sourceSets {  
    integrationTest {  
        java.srcDir file('src/integTest/java')  
        resources.srcDir file('src/integTest/resources')  
        compileClasspath = sourceSets.main.output + configurations.testRuntime  
        runtimeClasspath = output + compileClasspath  
    }  
}  
  
task integrationTest(type: Test) {  
    description = 'Runs the integration tests.'  
    group = 'verification'  
    testClassesDir = sourceSets.integrationTest.output.classesDir  
    classpath = sourceSets.integrationTest.runtimeClasspath  
    testResultsDir = file("$testResultsDir/integration")  
}
```

Set source and resources directory

Set compile and runtime classpath

# Separate tests with SourceSets

```
sourceSets {  
    integrationTest {  
        java.srcDir file('src/integTest/java')  
        resources.srcDir file('src/integTest/resources')  
        compileClasspath = sourceSets.main.output + configurations.testRuntime  
        runtimeClasspath = output + compileClasspath  
    }  
}  
  
task integrationTest(type: Test) {  
    description = 'Runs the integration tests.'  
    group = 'verification'  
    testClassesDir = sourceSets.integrationTest.output.classesDir  
    classpath = sourceSets.integrationTest.runtimeClasspath  
    testResultsDir = file("$testResultsDir/integration")  
}
```

Set source and resources directory

Set compile and runtime classpath

Custom test results directory

# Separate tests with SourceSets

```
sourceSets {  
    integrationTest {  
        java.srcDir file('src/integTest/java')  
        resources.srcDir file('src/integTest/resources')  
        compileClasspath = sourceSets.main.output + configurations.testRuntime  
        runtimeClasspath = output + compileClasspath  
    }  
}  
  
task integrationTest(type: Test) {  
    description = 'Runs the integration tests.'  
    group = 'verification'  
    testClassesDir = sourceSets.integrationTest.output.classesDir  
    classpath = sourceSets.integrationTest.runtimeClasspath  
    testResultsDir = file("$testResultsDir/integration")  
}
```

Set source and resources directory

Set compile and runtime classpath

Custom test results directory

gradlew integrationTest



On-the-fly bytecode instrumentation

No modification to source or bytecode

# Code coverage with JaCoCo

```
apply plugin: 'jacoco'

task jacocoIntegrationTestReport(type: JacocoReport) {
    sourceSets sourceSets.main
    executionData integTest
}
```

# Code coverage with JaCoCo

Apply JaCoCo plugin

```
apply plugin: 'jacoco'

task jacocoIntegrationTestReport(type: JacocoReport) {
    sourceSets sourceSets.main
    executionData integTest
}
```

# Code coverage with JaCoCo

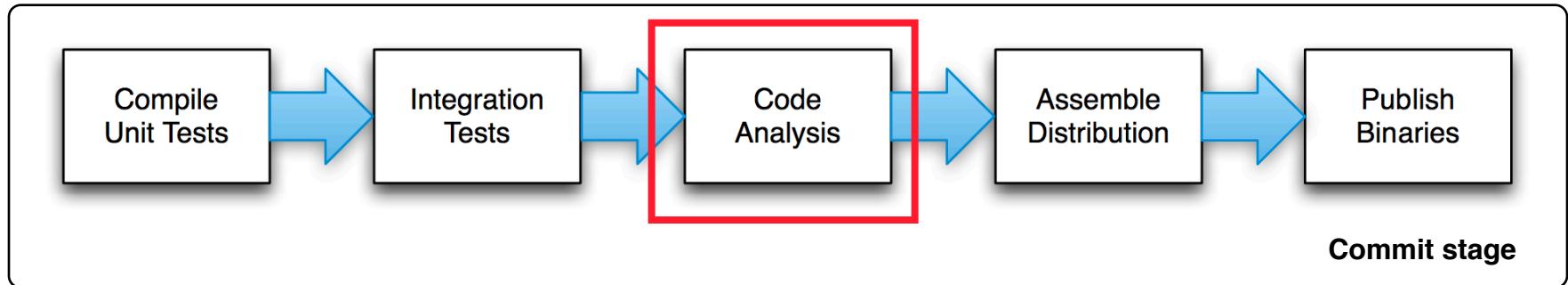
Apply JaCoCo plugin

```
apply plugin: 'jacoco'

task jacocoIntegrationTestReport(type: JacocoReport) {
    sourceSets sourceSets.main
    executionData integTest
}
```

Also report code coverage  
for integration tests

# Commit stage: Code analysis



Perform code health check

Fail build for low quality

Record progress over time



# Static code analysis tools



Checkstyle



FindBugs

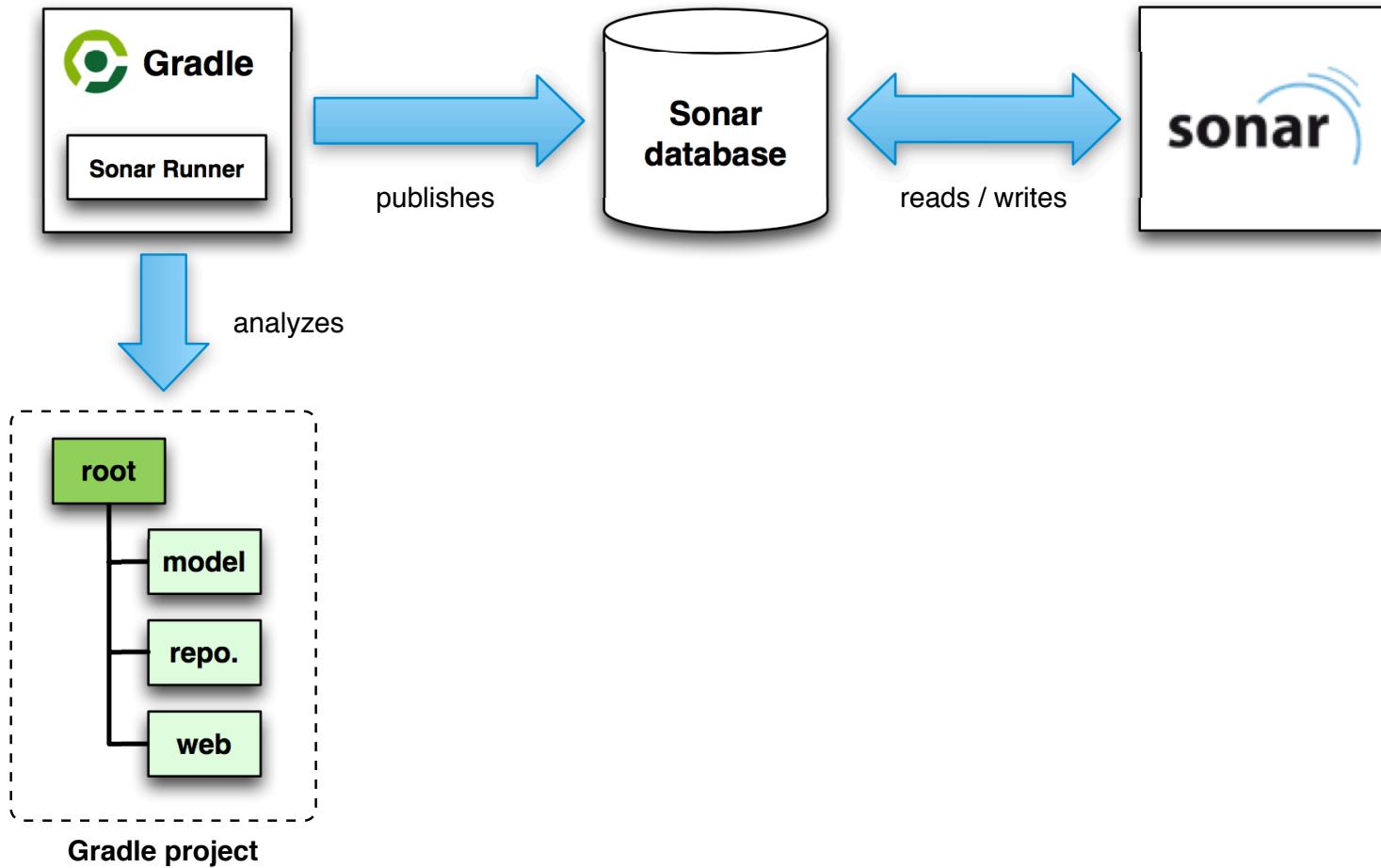
```
apply plugin: 'pmd'  
  
pmd {  
    ignoreFailures = true  
}  
  
tasks.withType(Pmd) {  
    reports {  
        xml.enabled = false  
        html.enabled = true  
    }  
}
```



```
apply plugin: 'jdepend'  
  
jdepend {  
    toolVersion = '2.9.1'  
    ignoreFailures = true  
}
```

gradlew check

# Measure quality over time with Sonar



# Applying the Sonar Runner plugin

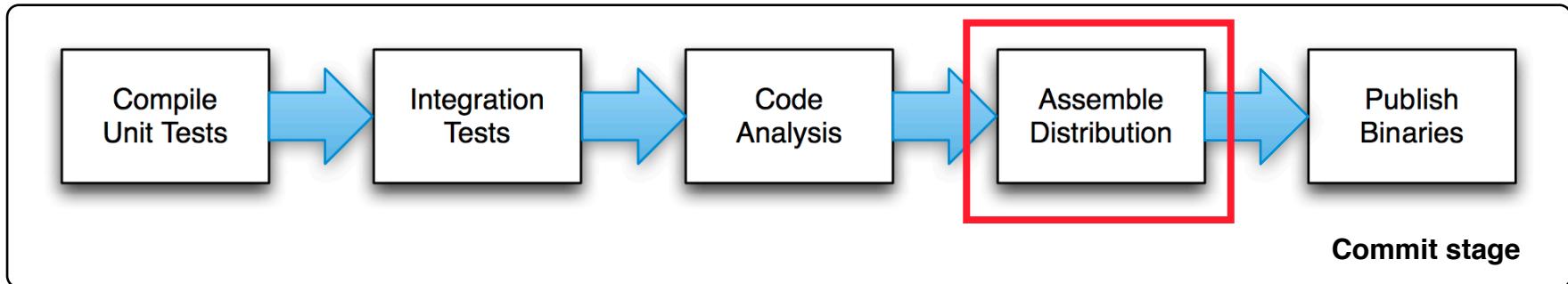
```
apply plugin: "sonar-runner"

sonarRunner {
    sonarProperties {
        property "sonar.host.url", "http://my.server.com"
        property "sonar.jdbc.url", "jdbc:mysql://my.server.com/sonar"
        property "sonar.jdbc.driverClassName", "com.mysql.jdbc.Driver"
        property "sonar.jdbc.username", "Fred Flintstone"
        property "sonar.jdbc.password", "very clever"
    }
}

subprojects {
    sonarRunner {
        sonarProperties {
            property "sonar.sourceEncoding", "UTF-8"
        }
    }
}
```

gradlew sonarRunner

# Commit stage: Assemble distribution



Exclude env. configuration

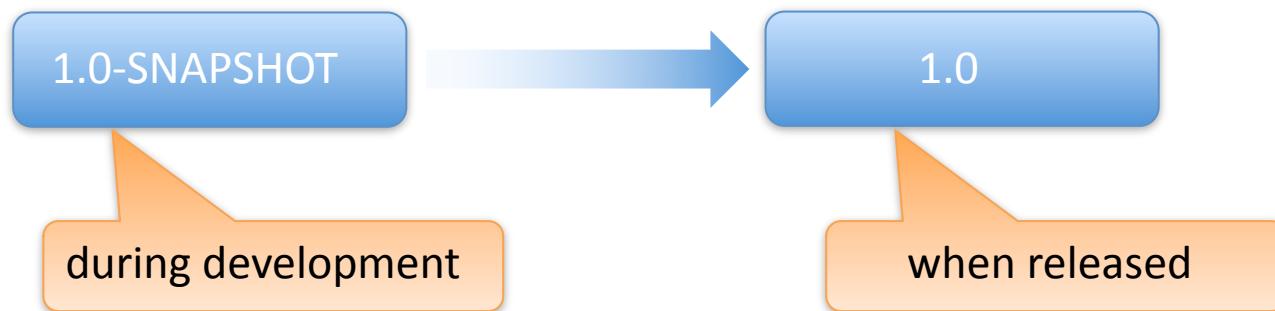
Include build information

Choose versioning strategy



# Versioning strategy

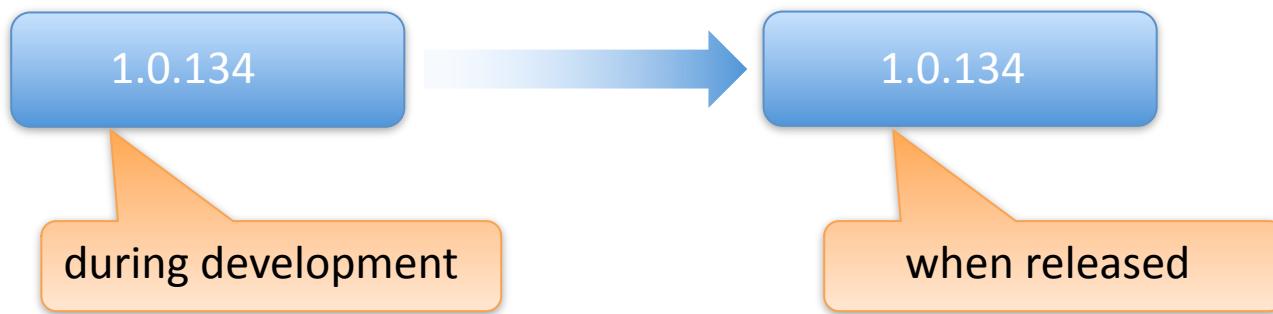
...the Maven way



Change version with Maven Release plugin

# Versioning strategy

...the Continuous Delivery way



1.0.134

Project version number

Jenkins build number



# Versioning strategy

...implemented with Gradle

```
ext.buildTimestamp = new Date().format('yyyy-MM-dd HH:mm:ss')

version = new ProjectVersion(1, 0, System.env.SOURCE_BUILD_NUMBER)

class ProjectVersion {
    Integer major
    Integer minor
    String build

    ProjectVersion(Integer major, Integer minor, String build) {
        this.major = major
        this.minor = minor
        this.build = build
    }

    @Override
    String toString() {
        String fullVersion = "$major.$minor"

        if(build) {
            fullVersion += ".$build"
        }

        fullVersion
    }
}
```

# Versioning strategy

...implemented with Gradle

```
ext.buildTimestamp = new Date().format('yyyy-MM-dd HH:mm:ss')

version = new ProjectVersion(1, 0, System.env.SOURCE_BUILD_NUMBER)

class ProjectVersion {
    Integer major
    Integer minor
    String build

    ProjectVersion(Integer major, Integer minor, String build) {
        this.major = major
        this.minor = minor
        this.build = build
    }

    @Override
    String toString() {
        String fullVersion = "$major.$minor"

        if(build) {
            fullVersion += ".$build"
        }

        fullVersion
    }
}
```

Jenkins Build Number

# Versioning strategy

...implemented with Gradle

```
ext.buildTimestamp = new Date().format('yyyy-MM-dd HH:mm:ss')

version = new ProjectVersion(1, 0, System.env.SOURCE_BUILD_NUMBER)

class ProjectVersion {
    Integer major
    Integer minor
    String build

    ProjectVersion(Integer major, Integer minor, String build) {
        this.major = major
        this.minor = minor
        this.build = build
    }

    @Override
    String toString() {
        String fullVersion = "$major.$minor"

        if(build) {
            fullVersion += ".$build"
        }

        fullVersion
    }
}
```

Jenkins Build Number

Builds version  
String representation

# Packaging the deployable artifact



```
project(':web') {
    apply plugin: 'war'

    task createBuildInfoFile << {
        def buildInfoFile = new File("$buildDir/build-info.properties")
        Properties props = new Properties()
        props.setProperty('version', project.version.toString())
        props.setProperty('timestamp', project.buildTimestamp)
        props.store(buildInfoFile.newWriter(), null)
    }

    war {
        dependsOn createBuildInfoFile
        baseName = 'todo'

        from(buildDir) {
            include 'build-info.properties'
            into('WEB-INF/classes')
        }
    }
}
```

# Packaging the deployable artifact



```
project(':web') {  
    apply plugin: 'war'  
  
    task createBuildInfoFile << {  
        def buildInfoFile = new File("$buildDir/build-info.properties")  
        Properties props = new Properties()  
        props.setProperty('version', project.version.toString())  
        props.setProperty('timestamp', project.buildTimestamp)  
        props.store(buildInfoFile.newWriter(), null)  
    }  
  
    war {  
        dependsOn createBuildInfoFile  
        baseName = 'todo'  
  
        from(buildDir) {  
            include 'build-info.properties'  
            into('WEB-INF/classes')  
        }  
    }  
}
```

Creates file containing  
build information

# Packaging the deployable artifact



```
project(':web') {  
    apply plugin: 'war'  
  
    task createBuildInfoFile << {  
        def buildInfoFile = new File("$buildDir/build-info.properties")  
        Properties props = new Properties()  
        props.setProperty('version', project.version.toString())  
        props.setProperty('timestamp', project.buildTimestamp)  
        props.store(buildInfoFile.newWriter(), null)  
    }  
  
    war {  
        dependsOn createBuildInfoFile  
        baseName = 'todo'  
  
        from(buildDir) {  
            include 'build-info.properties'  
            into('WEB-INF/classes')  
        }  
    }  
}
```

Creates file containing build information

Include build info file Into WAR distribution

# Packaging the deployable artifact



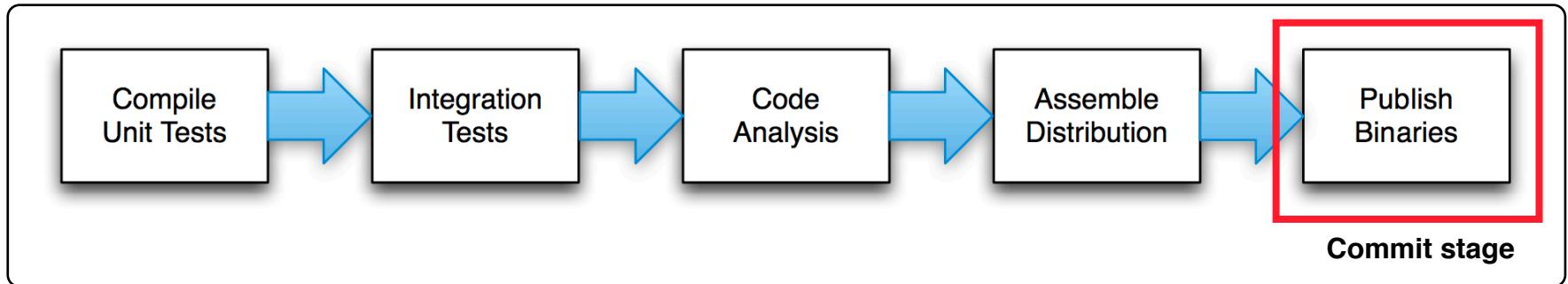
```
project(':web') {  
    apply plugin: 'war'  
  
    task createBuildInfoFile << {  
        def buildInfoFile = new File("$buildDir/build-info.properties")  
        Properties props = new Properties()  
        props.setProperty('version', project.version.toString())  
        props.setProperty('timestamp', project.buildTimestamp)  
        props.store(buildInfoFile.newWriter(), null)  
    }  
  
    war {  
        dependsOn createBuildInfoFile  
        baseName = 'todo'  
  
        from(buildDir) {  
            include 'build-info.properties'  
            into('WEB-INF/classes')  
        }  
    }  
}
```

Creates file containing  
build information

Include build info file  
Into WAR distribution

gradlew assemble

# Commit stage: Publish binaries



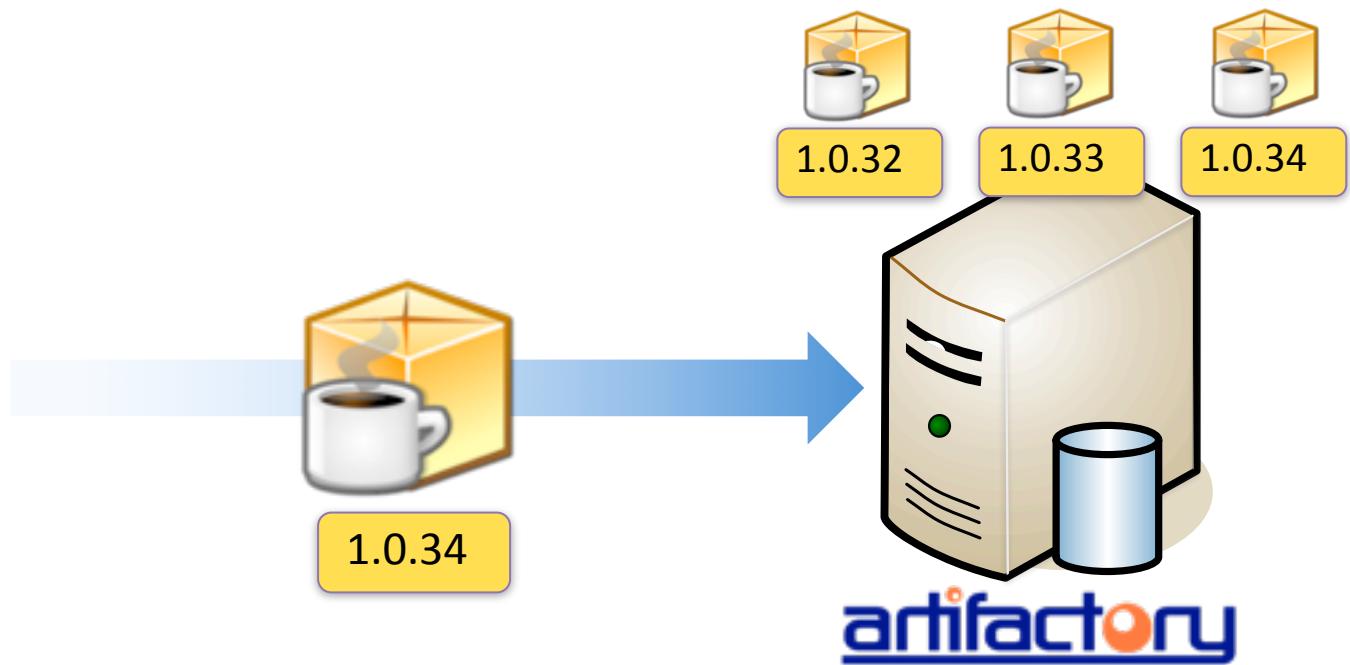
Version artifact(s)

Use binary repository

Publish once, then reuse



# Publishing the deployable artifact



# Using the Maven Publishing plugin

```
apply plugin: 'maven-publish'

ext.fullRepoUrl = "$config.binaryRepository.url/$config.binaryRepository.name"

publishing {
    publications {
        webApp(MavenPublication) {
            from components.web
        }
    }

    repositories {
        maven {
            url fullRepoUrl

            credentials {
                username = config.binaryRepository.username
                password = config.binaryRepository.password
            }
        }
    }
}
```

# Using the Maven Publishing plugin

Build repository URL  
from configuration

```
apply plugin: 'maven-publish'

ext.fullRepoUrl = "$config.binaryRepository.url/$config.binaryRepository.name"

publishing {
    publications {
        webApp(MavenPublication) {
            from components.web
        }
    }

    repositories {
        maven {
            url fullRepoUrl

            credentials {
                username = config.binaryRepository.username
                password = config.binaryRepository.password
            }
        }
    }
}
```

# Using the Maven Publishing plugin

```
apply plugin: 'maven-publish'

ext.fullRepoUrl = "$config.binaryRepository.url/$config.binaryRepository.name"

publishing {
    publications {
        webApp(MavenPublication) {
            from components.web
        }
    }
}

repositories {
    maven {
        url fullRepoUrl
        credentials {
            username = config.binaryRepository.username
            password = config.binaryRepository.password
        }
    }
}
```

Build repository URL  
from configuration

Assign publication name  
and component

# Using the Maven Publishing plugin

```
apply plugin: 'maven-publish'

ext.fullRepoUrl = "$config.binaryRepository.url/$config.binaryRepository.name"

publishing {
    publications {
        webApp(MavenPublication) {
            from components.web
        }
    }
}

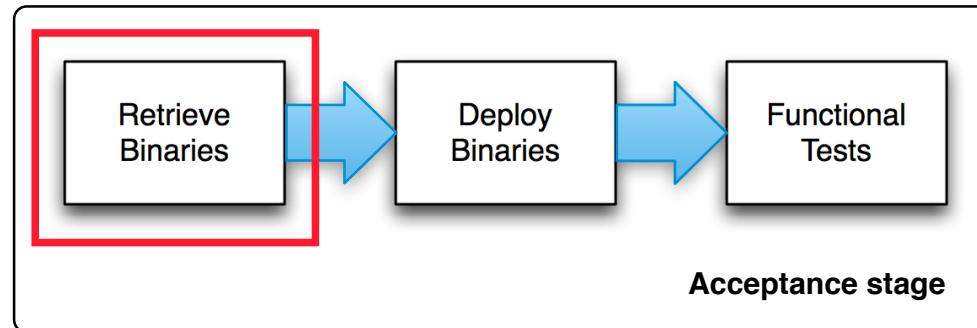
repositories {
    maven {
        url fullRepoUrl
    }
}
```

Build repository URL  
from configuration

Assign publication name  
and component

gradlew publish

# Acceptance stage: Retrieve binaries

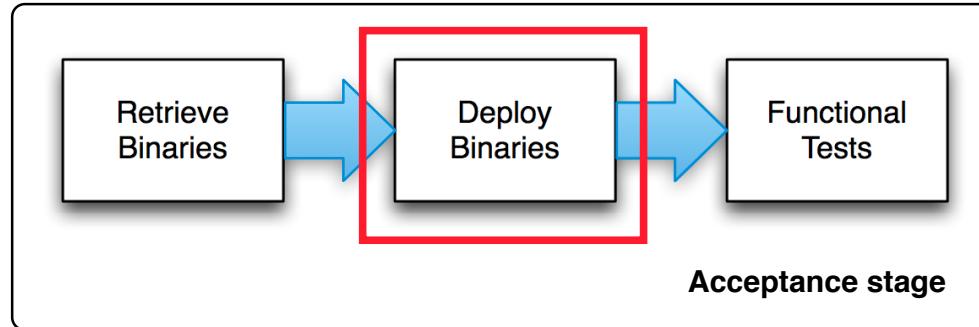


Request versioned artifact

Store in temp. directory



# Acceptance stage: Deploy binaries



Deployment on request

Make it a reliable process

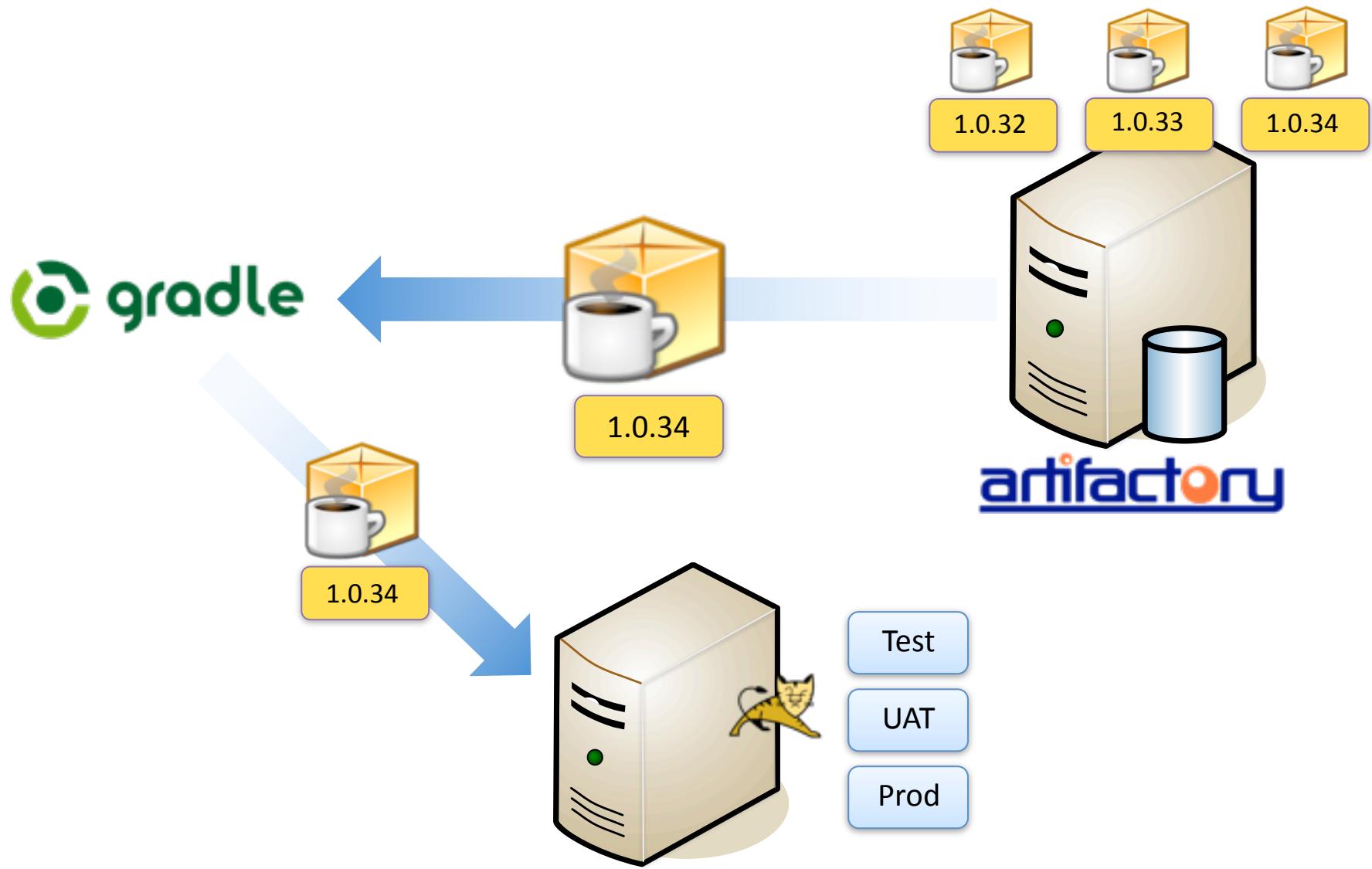
Use process for all envs



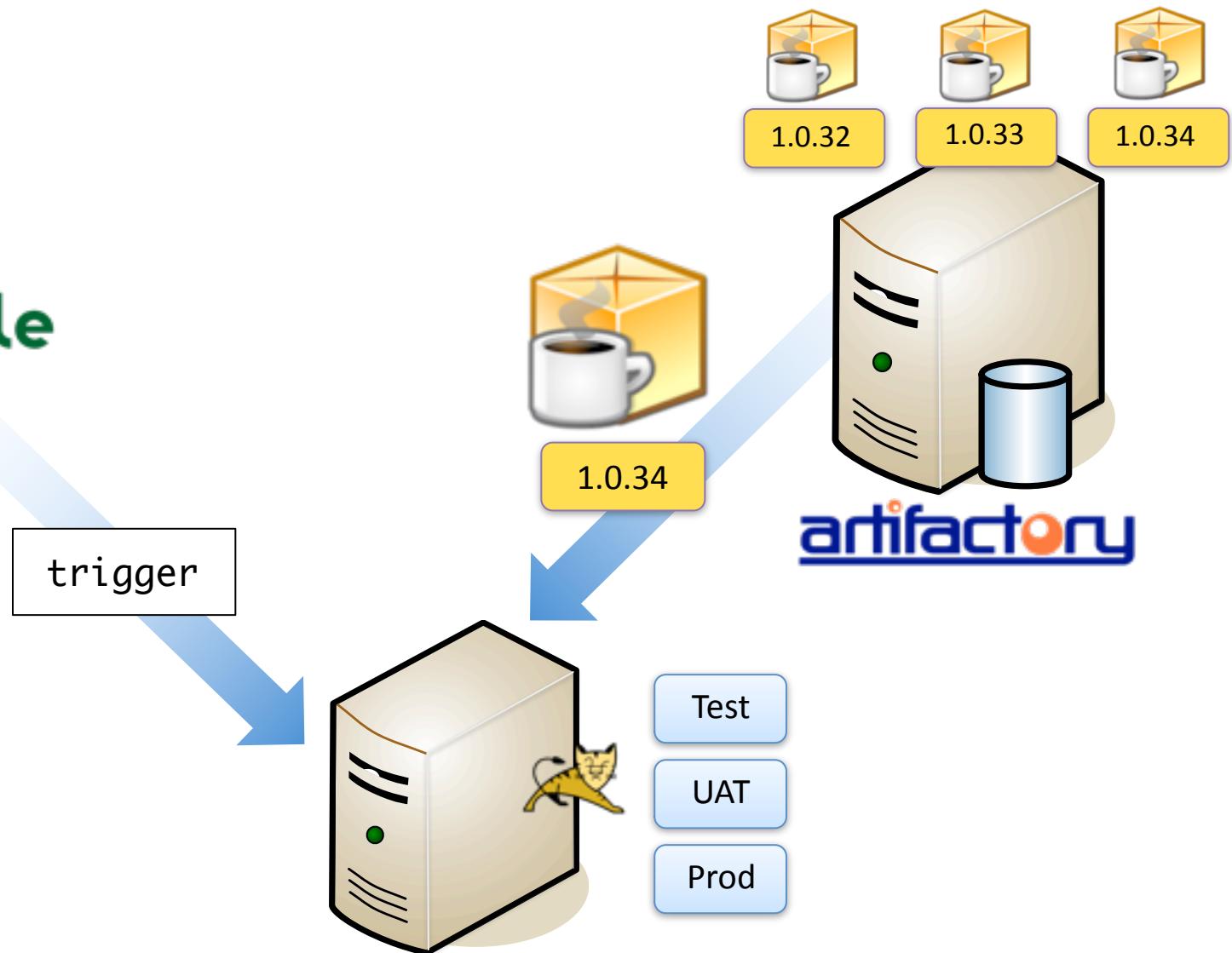
# Deployment is often more than just copying a file



# Push deployment



# Pull deployment



# Remote command execution with plugin



ssh

```
apply plugin: 'ssh'

remotes {
    localhost {
        host = 'localhost'
        user = System.properties['user.name']
        identity = file("${System.properties['user.home']}/.ssh/id_rsa")
    }
}

task deploy(type: SshTask) {
    session(remotes.localhost) {
        execute('sudo -u tomcat $tomcatRemoteDir/bin/shutdown.sh')
        execute('sudo -u tomcat rm -rf $tomcatRemoteDir/webapps/myapp')
        ...
    }
}
```

# Remote command execution with plugin

Define hosts  
configuration



ssh

```
apply plugin: 'ssh'

remotes {
    localhost {
        host = 'localhost'
        user = System.properties['user.name']
        identity = file("${System.properties['user.home']}/.ssh/id_rsa")
    }
}

task deploy(type: SshTask) {
    session(remotes.localhost) {
        execute('sudo -u tomcat $tomcatRemoteDir/bin/shutdown.sh')
        execute('sudo -u tomcat rm -rf $tomcatRemoteDir/webapps/myapp')
        ...
    }
}
```

# Remote command execution with plugin

```
apply plugin: 'ssh'

remotes {
    localhost {
        host = 'localhost'
        user = System.properties['user.name']
        identity = file("${System.properties['user.home']}/.ssh/id_rsa")
    }
}

task deploy(type: SshTask) {
    session(remotes.localhost) {
        execute('sudo -u tomcat $tomcatRemoteDir/bin/shutdown.sh')
        execute('sudo -u tomcat rm -rf $tomcatRemoteDir/webapps/myapp')
        ...
    }
}
```

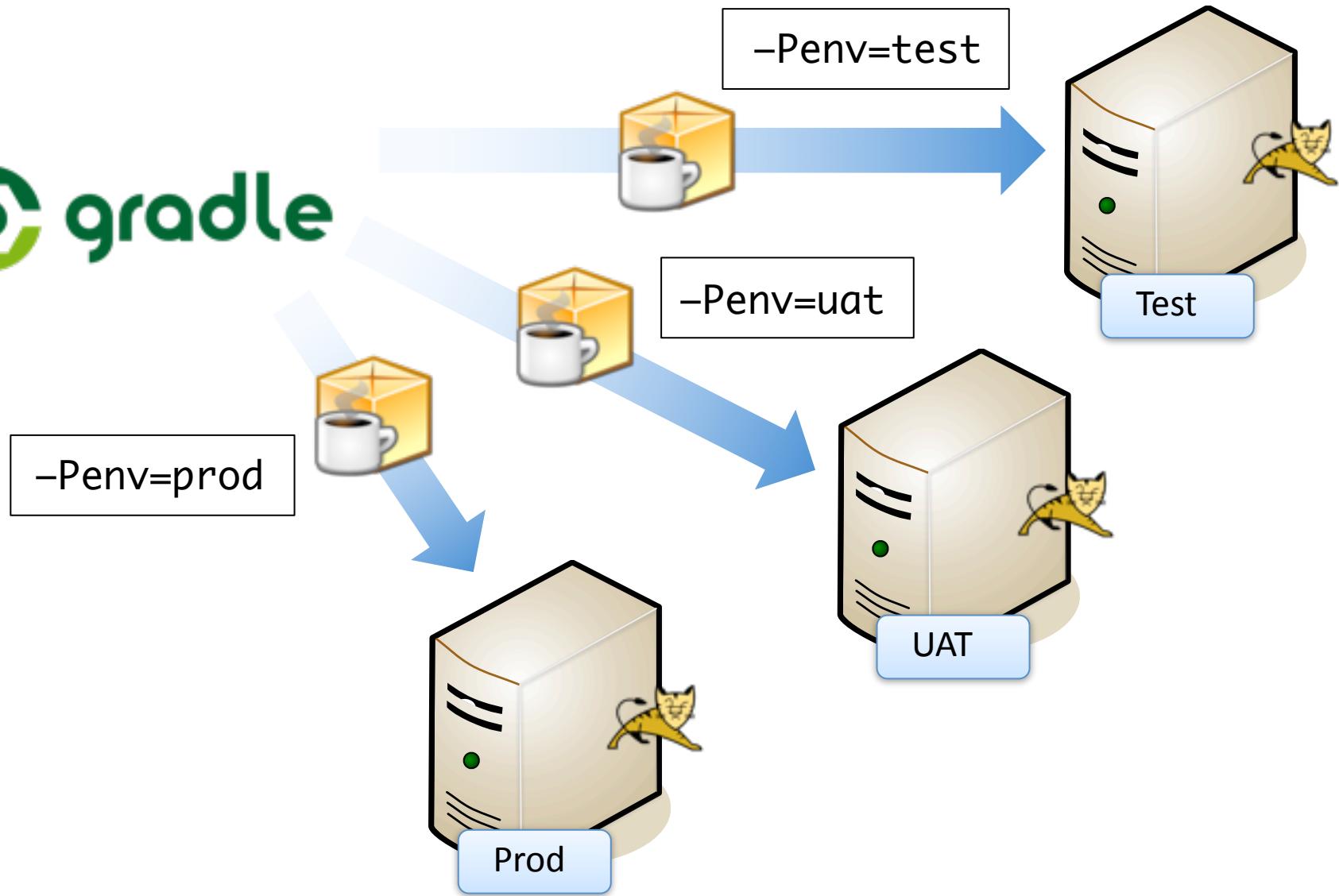
Define hosts configuration



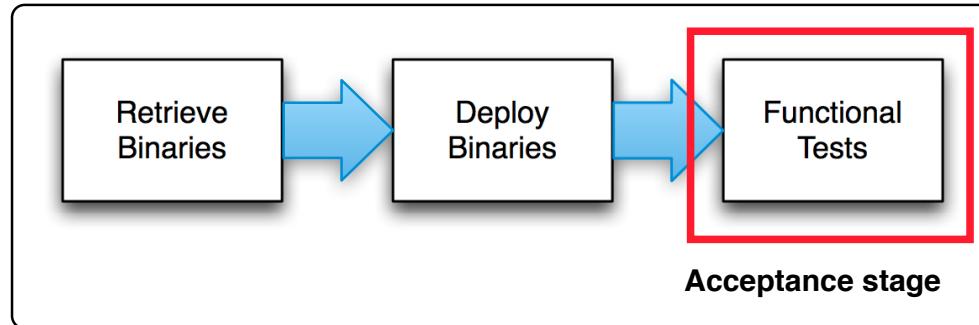
ssh

Execute SSH commands  
against remote

# Deploying to multiple environments



# Acceptance stage: Functional tests



Test all UI permutations

Test important use cases

Run against different envs



# Automate most important use cases as functional tests

```
task functionalTest(type: Test) {  
    testClassesDir = sourceSets.functionalTest.output.classesDir  
    classpath = sourceSets.functionalTest.runtimeClasspath  
    reports.html.destination = file("$reports.html.destination/functional")  
    reports.junitXml.destination = file("$reports.junitXml.destination/functional")  
    systemProperty 'geb.env', 'firefox'  
    systemProperty 'geb.build.reportsDir', reporting.file("$name/geb")  
    mustRunAfter deploy  
    dependsOn smokeTest  
}
```

gradlew functionalTest

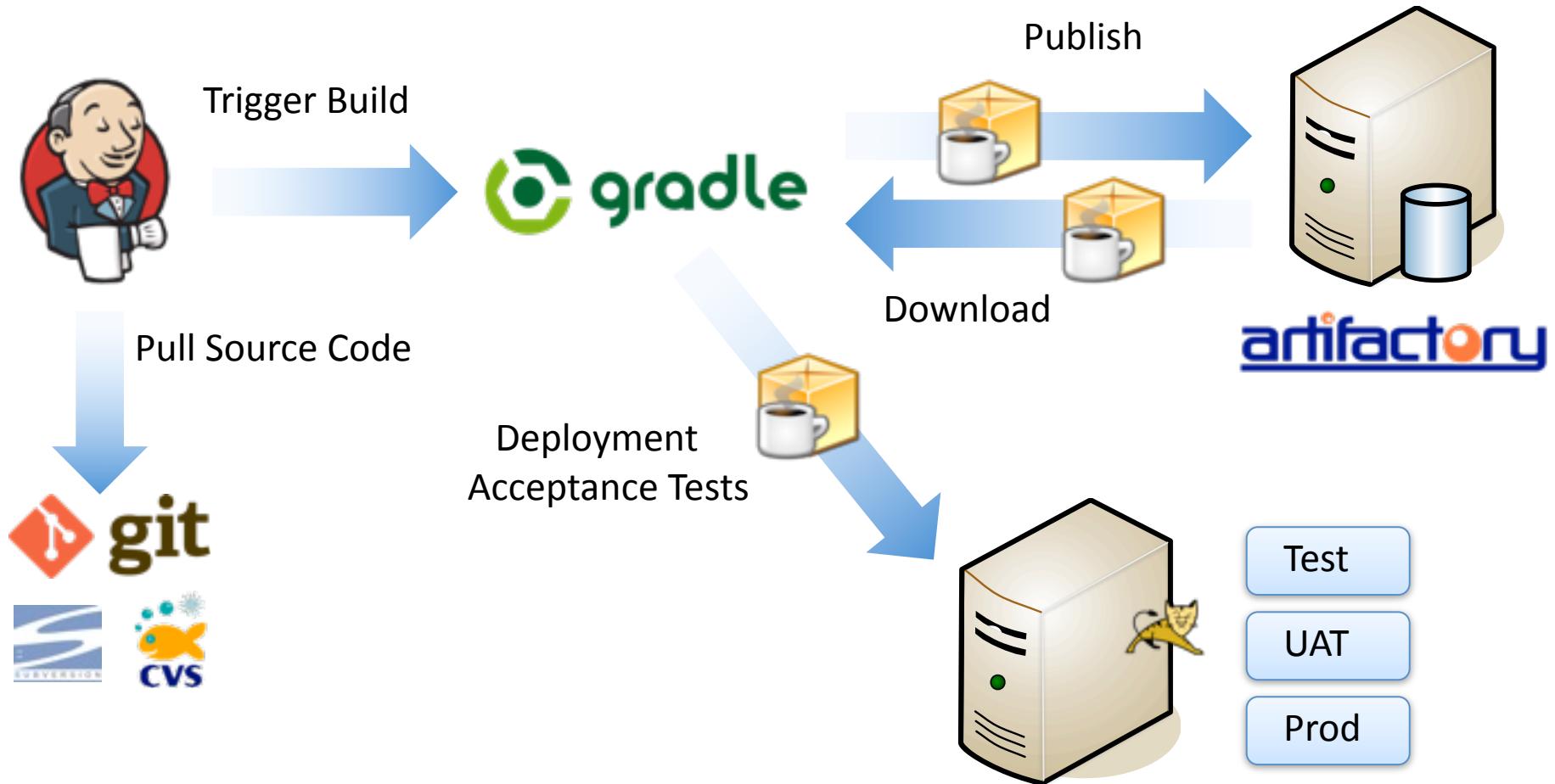
# Automate most important use cases as functional tests

```
task functionalTest(type: Test) {  
    testClassesDir = sourceSets.functionalTest.output.classesDir  
    classpath = sourceSets.functionalTest.runtimeClasspath  
    reports.html.destination = file("$reports.html.destination/functional")  
    reports.junitXml.destination = file("$reports.junitXml.destination/functional")  
    systemProperty 'geb.env', 'firefox'  
    systemProperty 'geb.build.reportsDir', reporting.file("$name/geb")  
    mustRunAfter deploy  
    dependsOn smokeTest  
}
```

gradlew functionalTest

Only run tests after environment  
is deemed "functional"

# Let's bring Jenkins into play!



# Model pipeline as series of jobs

		<a href="#">todo-acceptance-tests</a>	15 days ( <a href="#">todo#76</a> )	N/A	1 min 56 sec	
		<a href="#">todo-code-quality</a>	15 days ( <a href="#">todo#76</a> )	N/A	1 min 6 sec	
		<a href="#">todo-deploy-production</a>	N/A	N/A	N/A	
		<a href="#">todo-deploy-test</a>	15 days ( <a href="#">todo#76</a> )	N/A	51 sec	
		<a href="#">todo-deploy-uat</a>	20 days ( <a href="#">todo#65</a> )	16 days ( <a href="#">todo#67</a> )	47 sec	
		<a href="#">todo-distribution</a>	15 days ( <a href="#">todo#76</a> )	29 days ( <a href="#">todo#41</a> )	45 sec	
		<a href="#">todo-initial</a>	15 days ( <a href="#">todo#76</a> )	N/A	56 sec	
		<a href="#">todo-inteq-tests</a>	15 days ( <a href="#">todo#76</a> )	N/A	46 sec	
		<a href="#">todo-performance-tests</a>	15 days ( <a href="#">todo#76</a> )	15 days ( <a href="#">todo#74</a> )	1 min 28 sec	



# Jenkins

# Model pipeline as series of jobs

		<a href="#">todo-acceptance-tests</a>	15 days (todo#76)	N/A	1 min 56 sec	
		<a href="#">todo-code-quality</a>	15 days (todo#76)	N/A	1 min 6 sec	
		<a href="#">todo-deploy-production</a>	N/A	N/A	N/A	
		<a href="#">todo-deploy-test</a>	15 days (todo#76)	N/A	51 sec	
		<a href="#">todo-deploy-uat</a>	20 days (todo#65)	16 days (todo#67)	47 sec	
		<a href="#">todo-distribution</a>	15 days (todo#76)	29 days (todo#41)	45 sec	
		<a href="#">todo-initial</a>	15 days (todo#76)	N/A	56 sec	
		<a href="#">todo-integ-tests</a>	15 days (todo#76)	N/A	46 sec	
		<a href="#">todo-performance-tests</a>	15 days (todo#76)	15 days (todo#74)	1 min 28 sec	

Triggered job when change  
to SCM is detected



# Jenkins

# Initial Jenkins Build Job

Jenkins / todo-initial

ENABLE AUTO REFRESH

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Coverage Trend](#)

[Git Polling Log](#)

**Project todo-initial**

[Workspace](#)

[Recent Changes](#)

[Latest Test Result \(no failures\)](#)

**Downstream Projects**

[todo-integ-tests](#)

**Permalinks**

- Last build (todo#78), 1 hr 24 min ago
- Last stable build (todo#78), 1 hr 24 min ago
- Last successful build (todo#78), 1 hr 24 min ago

[RSS for all](#) [RSS for failures](#)

**Test Result Trend**

Build	Count
todo#74	7
todo#75	7
todo#76	7
todo#77	7
todo#78	7

[just show failures](#) [enlarge](#)

**Code Coverage Trend**

Build	lineCovered	lineMissed
todo#74	120	250
todo#75	120	250
todo#76	120	250
todo#77	120	250
todo#78	120	250

[lineCovered](#) [lineMissed](#)



# Build Name Setter Plugin

Jenkins search ENABLE AUTO REFRESH

[Back to Dashboard](#) [Status](#) [Changes](#) [Workspace](#) [Build Now](#) [Delete Project](#) [Configure](#) [Coverage Trend](#) [Git Polling Log](#)

**Project todo-initial**

[Workspace](#) [Recent Changes](#) [Latest Test Result \(no failures\)](#)

**Downstream Projects**

[todo-integ-tests](#)

**Permalinks**

- Last build (todo#78), 1 hr 24 min ago
- Last stable build (todo#78), 1 hr 24 min ago
- Last successful build (todo#78), 1 hr 24 min ago

[RSS for all](#) [RSS for failures](#)

**Test Result Trend**

Build Number	Count
todo#74	7
todo#75	0
todo#76	7
todo#77	7
todo#78	7

[add description](#) [Disable Project](#)

**Code Coverage Trend**

Build Number	lineCovered	lineMissed
todo#74	115	250
todo#75	115	250
todo#76	115	250
todo#77	115	250
todo#78	115	250

[just show failures](#) [enlarge](#)



# JaCoCo Plugin

Jenkins / todo-initial

ENABLE AUTO REFRESH

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Coverage Trend](#)

[Git Polling Log](#)

**Project todo-initial**

[Workspace](#)

[Recent Changes](#)

[Latest Test Result \(no failures\)](#)

**Downstream Projects**

[todo-integ-tests](#)

**Permalinks**

- [Last build \(todo#78\), 1 hr 24 min ago](#)
- [Last stable build \(todo#78\), 1 hr 24 min ago](#)
- [Last successful build \(todo#78\), 1 hr 24 min ago](#)

[RSS for all](#) [RSS for failures](#)

**Test Result Trend**

Build	Count
todo#74	7
todo#75	6
todo#76	5
todo#77	4
todo#78	7

**Code Coverage Trend**

Build	lineCovered	lineMissed
todo#74	100	250
todo#75	100	250
todo#76	100	250
todo#77	100	250
todo#78	100	250

Just show failures) enlarge

green lineCovered  
red lineMissed



# Parameterized Trigger Plugin

Jenkins / todo-initial

ENABLE AUTO REFRESH

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Coverage Trend](#)

[Git Polling Log](#)

**Project todo-initial**

[Workspace](#)

[Recent Changes](#)

[Test Result \(no failures\)](#)

**Downstream Projects**

[todo-integ-tests](#)

**Permalinks**

- Last build (todo#78), 1 hr 24 min ago
- Last stable build (todo#78), 1 hr 24 min ago
- Last successful build (todo#78), 1 hr 24 min ago

[RSS for all](#) [RSS for failures](#)

**Test Result Trend**

Count

Build	Count
todo#74	7
todo#75	7
todo#76	7
todo#77	7
todo#78	7

[add description](#) [Disable Project](#)

**Code Coverage Trend**

lineCovered lineMissed

Build	lineCovered	lineMissed
todo#74	120	250
todo#75	120	250
todo#76	120	250
todo#77	120	250
todo#78	120	250



## Gradle Plugin

### Build

#### Invoke Gradle script



Invoke Gradle

Use Gradle Wrapper

Make gradlew executable

From Root Build Script Dir

Build step description



Switches



Tasks



Root Build script



Build File



Specify Gradle build file to run. Also, [some environment variables are available to the build script](#)



## Gradle Plugin

### Build

#### Invoke Gradle script



Invoke Gradle

Use Gradle Wrapper

Make gradlew executable

From Root Build Script Dir

Build step description



Switches



Tasks



Root Build script



Build File



Run clean task to remove  
existing artifacts

variables are available to the



## Build Pipeline Plugin

### Build Pipeline: To Do application



Visualization of chained pipeline jobs

```
> gradle askQuestions  
:askQuestions
```

BUILD SUCCESSFUL

Total time: 300 secs