Building a Continuous Delivery Pipeline with Gradle and Jenkins

Jenkins

Gary Hale

gradleware

Presentation will be available on
https://speakerdeck.com/ghale

gary.hale@gradleware.com

ghale

Original content by Ben Muschko

Modified and used with permission


Releases don’t have to be painful

Continuous Delivery

Deliver software fast and frequently

#1 Every commit can result in a release

#2 Automate everything!

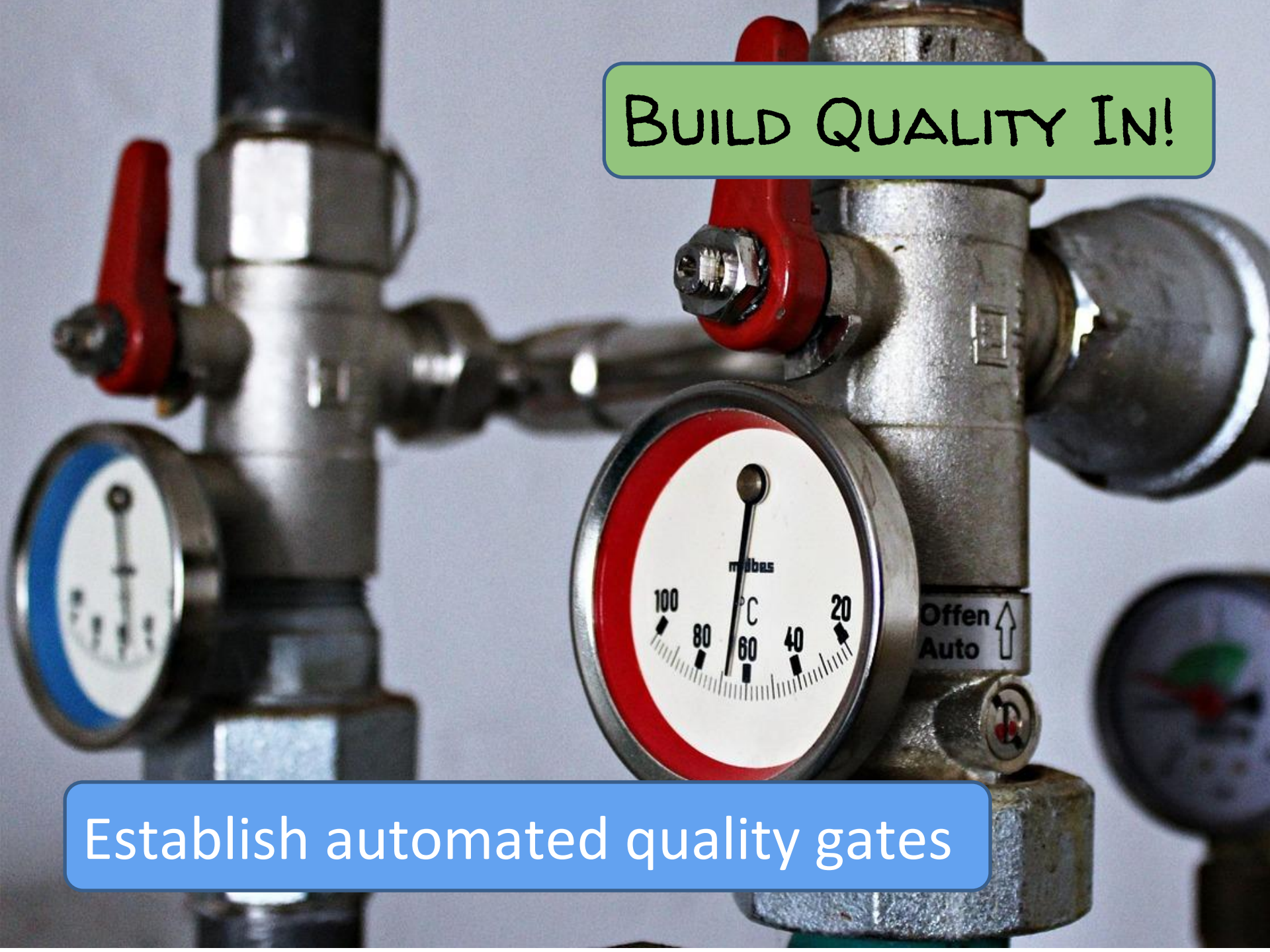
#3 Automated tests are essential

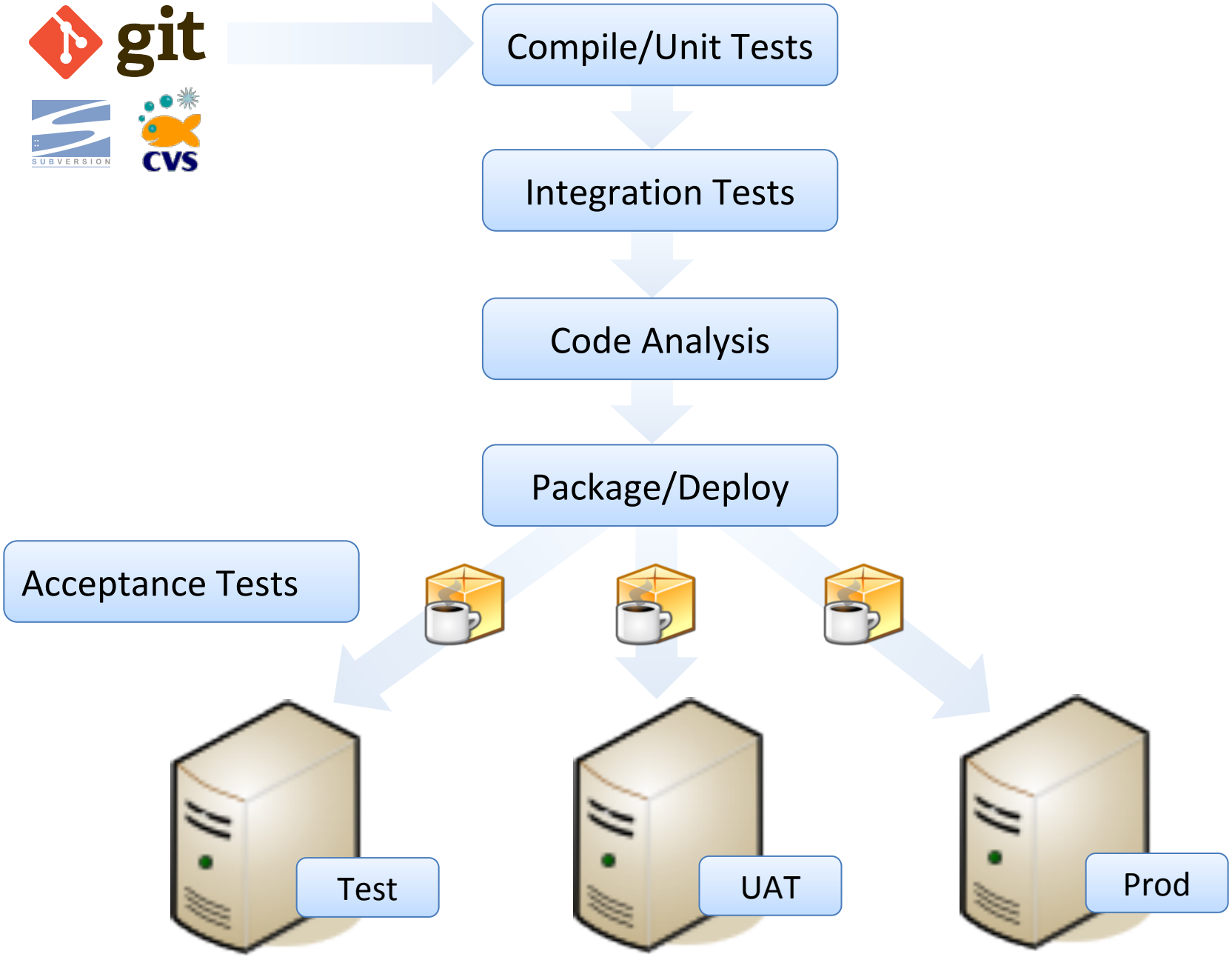
#4 Done means released

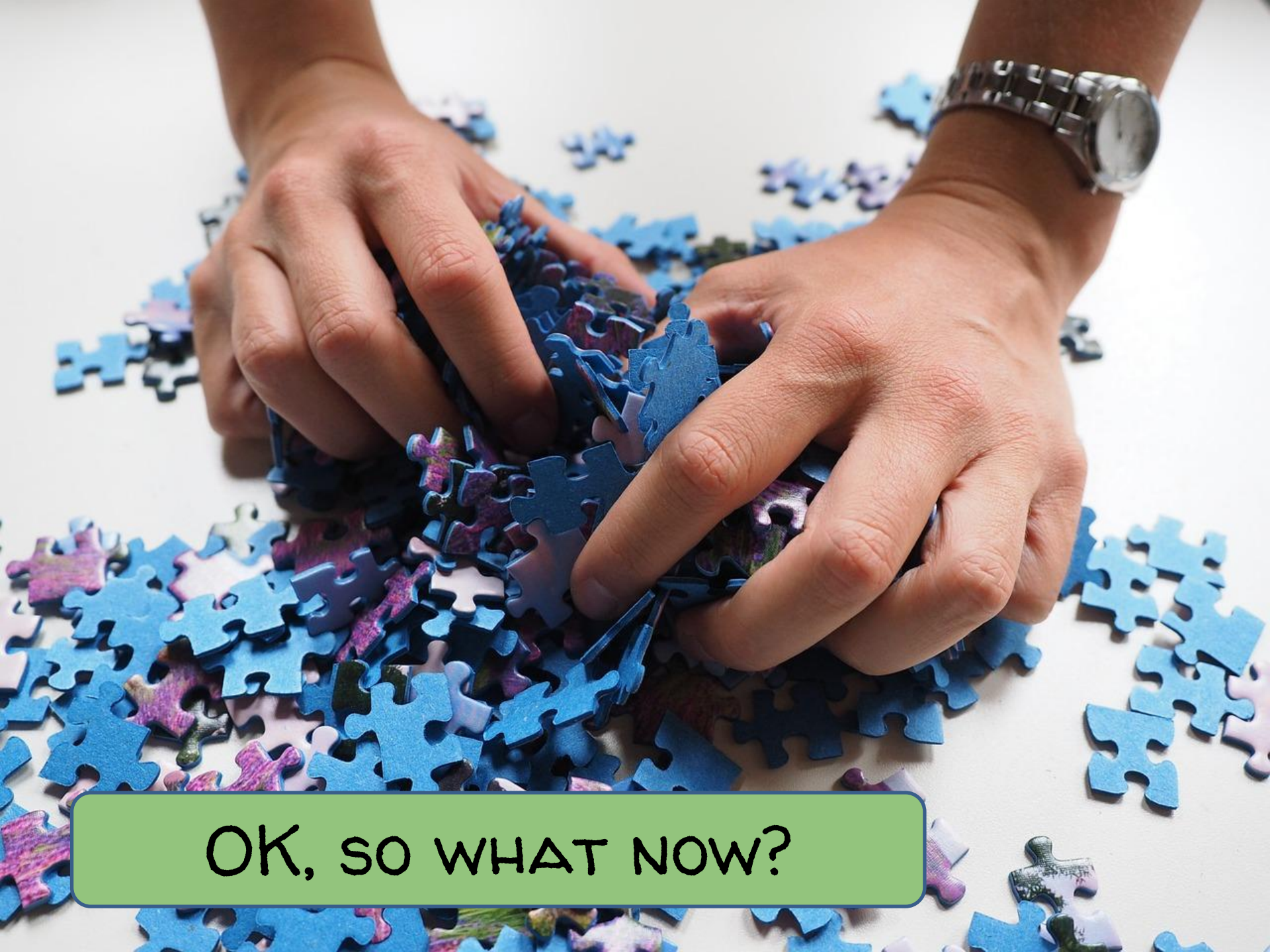
PRINCIPLES

The Build as a Pipeline

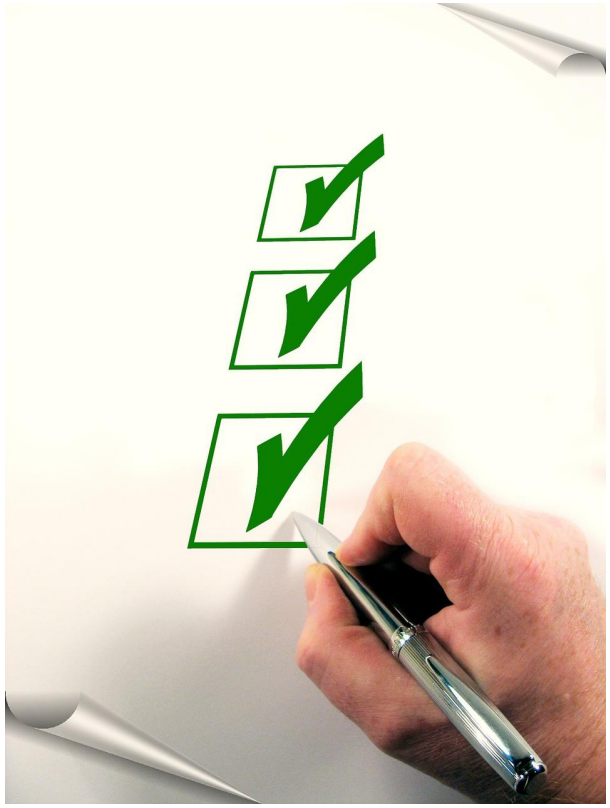
Automated manifestation of delivery process

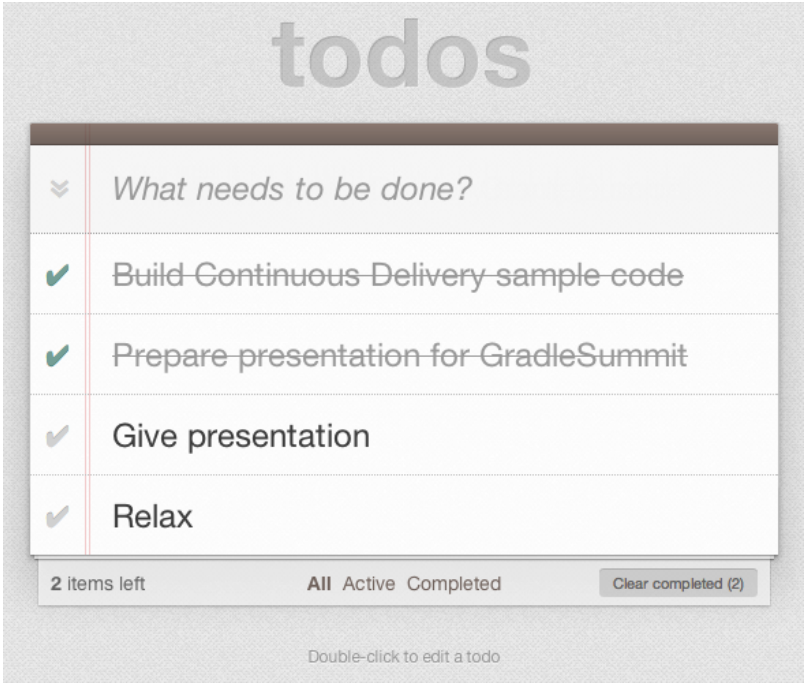

Build Quality In!
100
80
60
40
20
°C
Offen
Auto
Establish automated quality gates

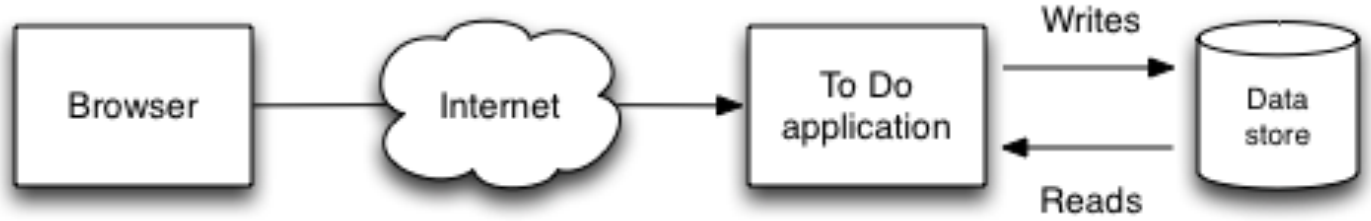

git
SUBVERSION
CVS
Compile/Unit Tests
Integration Tests
Code Analysis
Package/Deploy
Acceptance Tests
Test
UAT
Prod


OK, so what now?

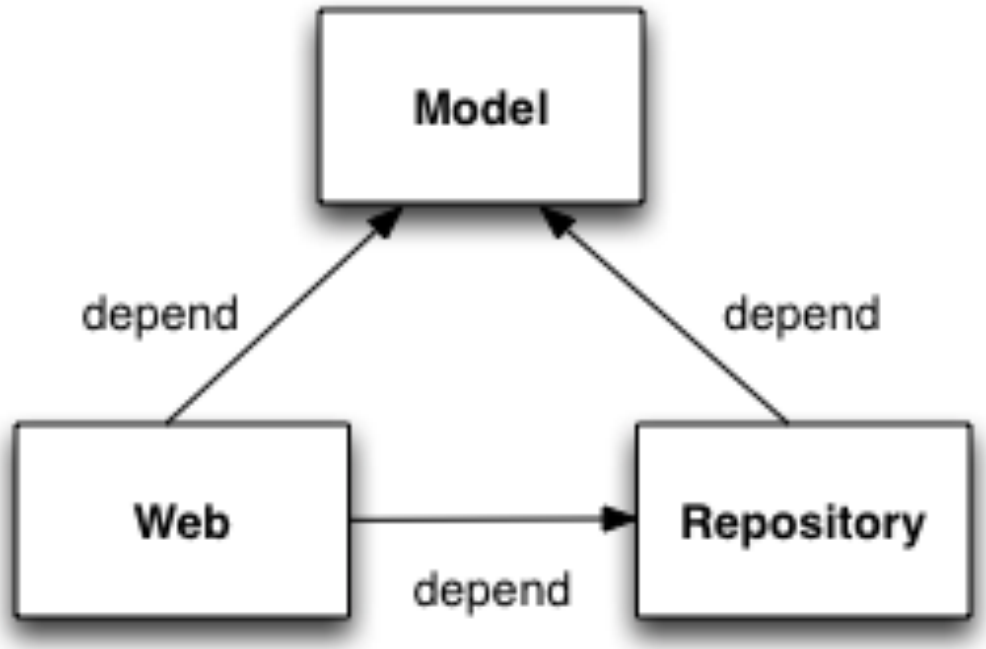
The “revolutionary” sample application



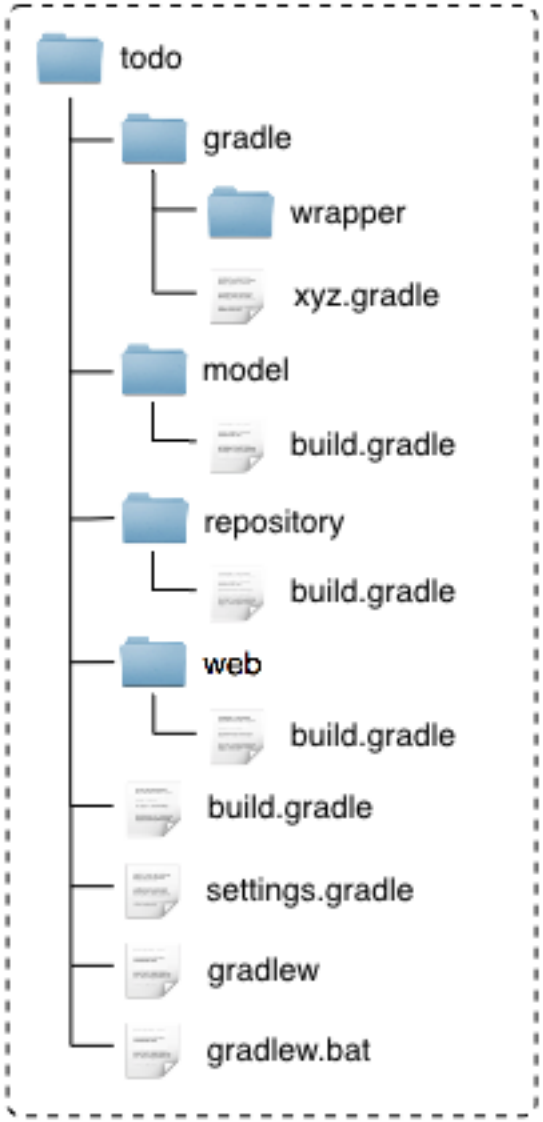




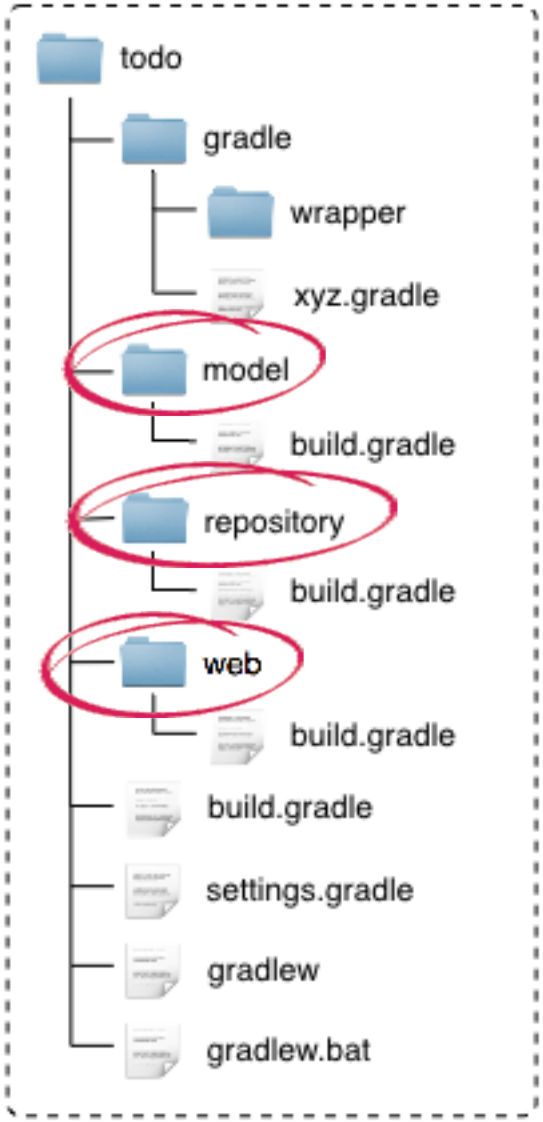
Multi-project dependencies



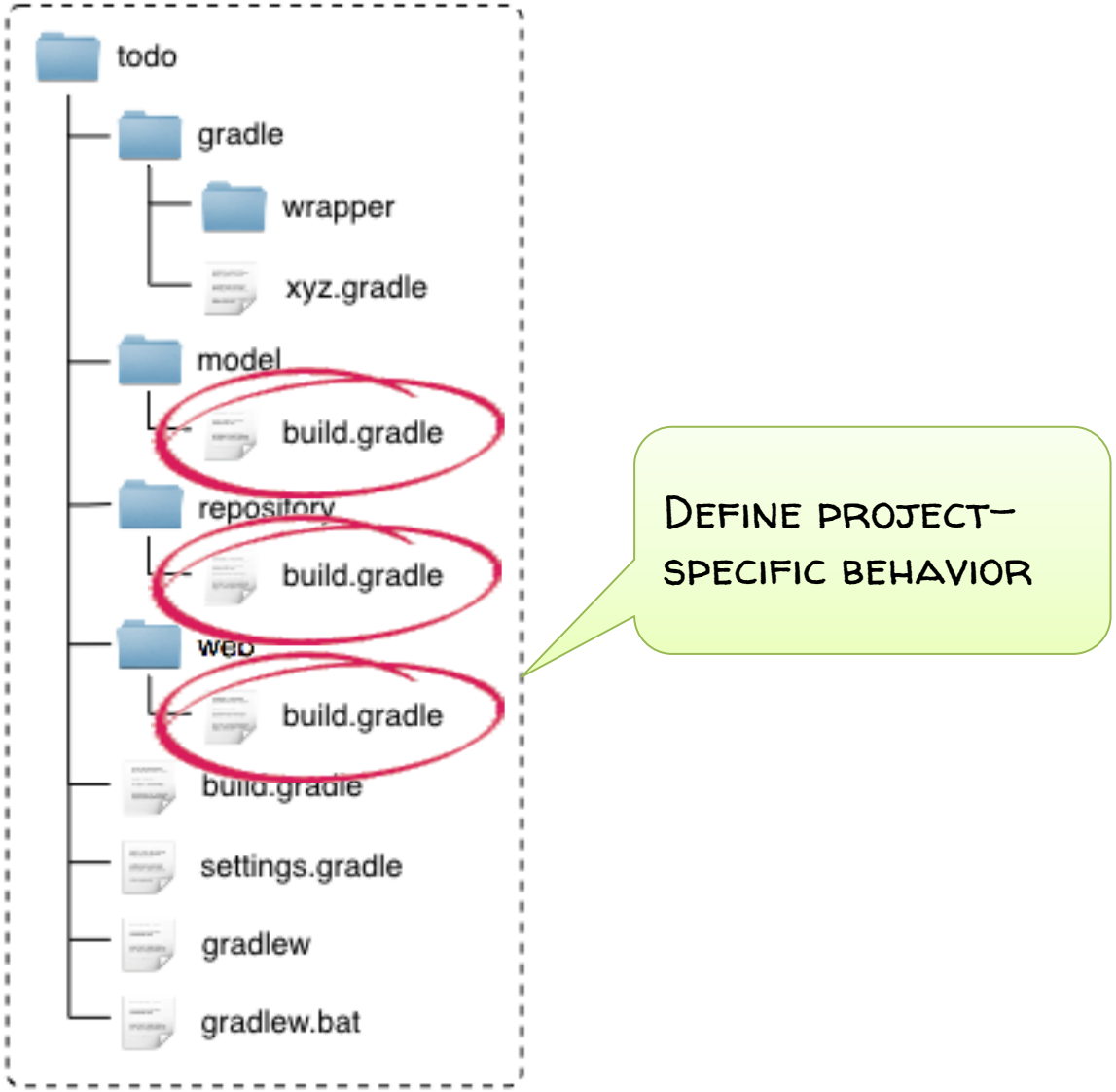
Project hierarchy



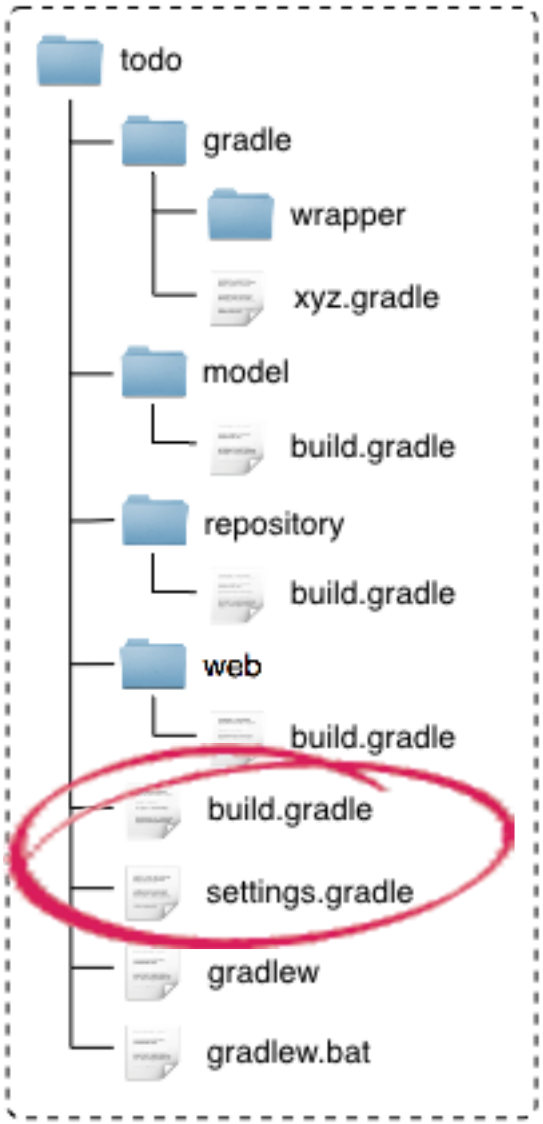
Project hierarchy



Project hierarchy



Project hierarchy

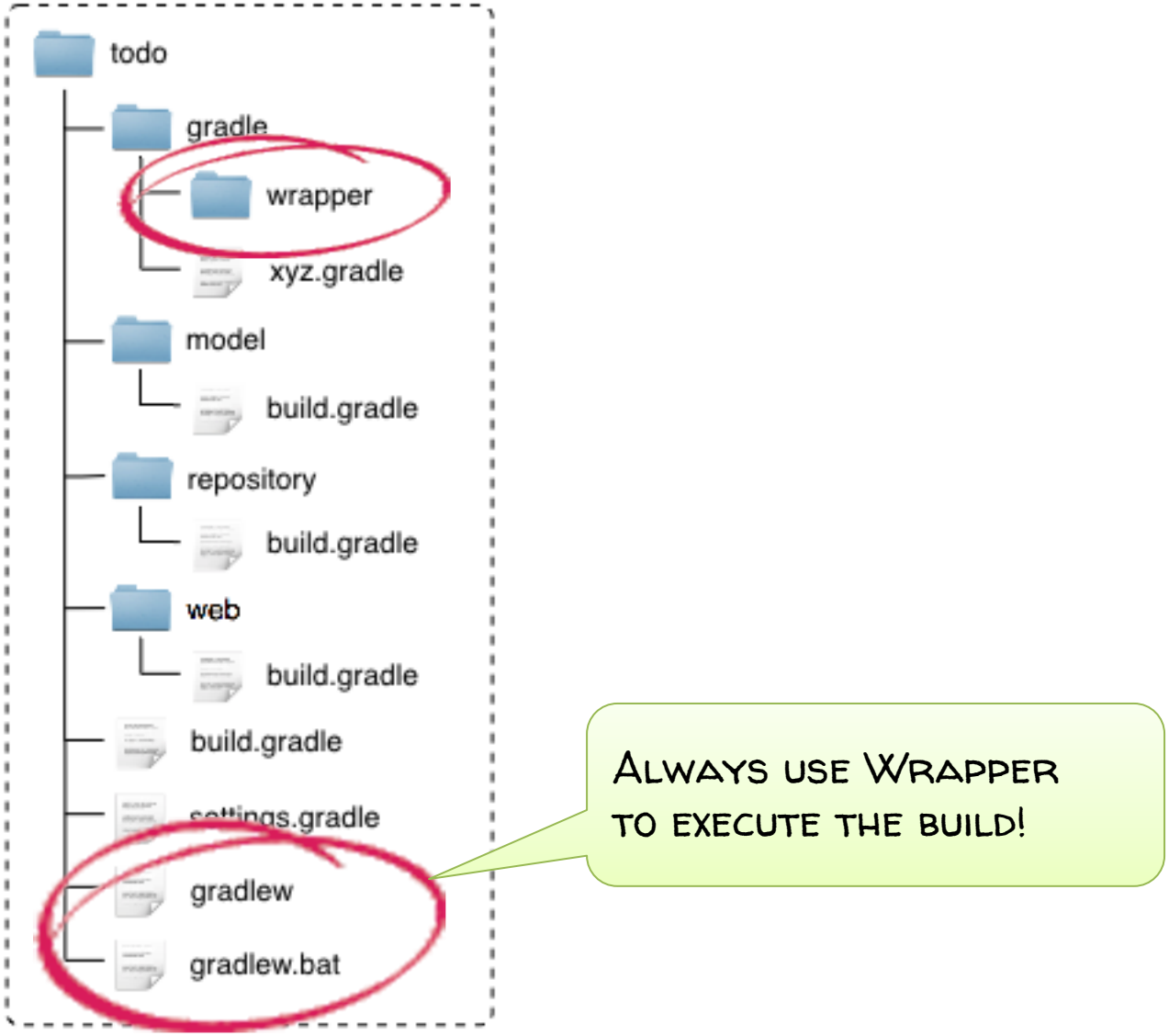


Defines which projects are taking part in the build

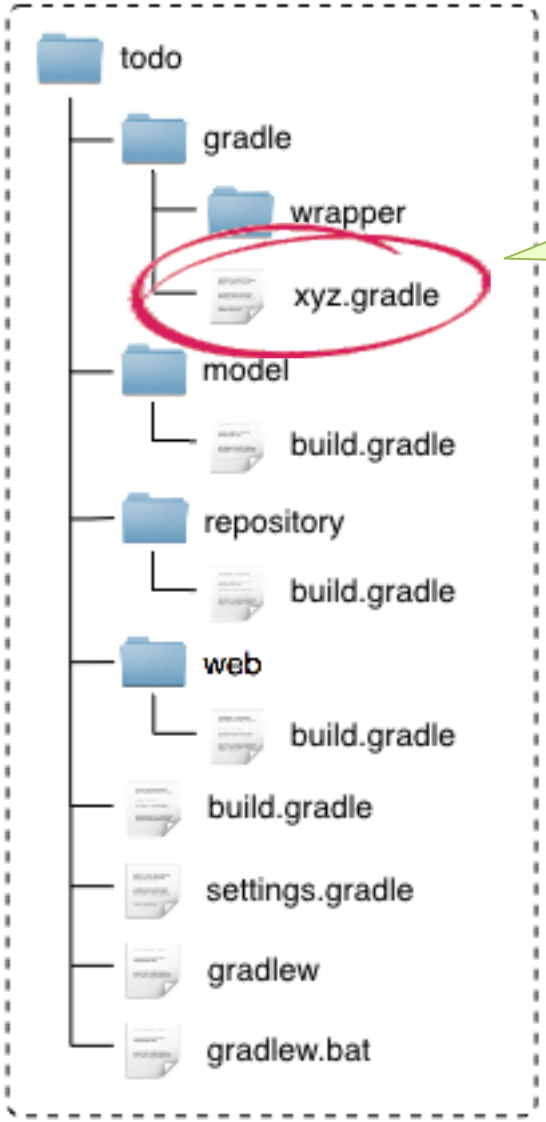
settings.gradle

```
include 'model'
include 'repository'
include 'web'
```

Project hierarchy



Project hierarchy

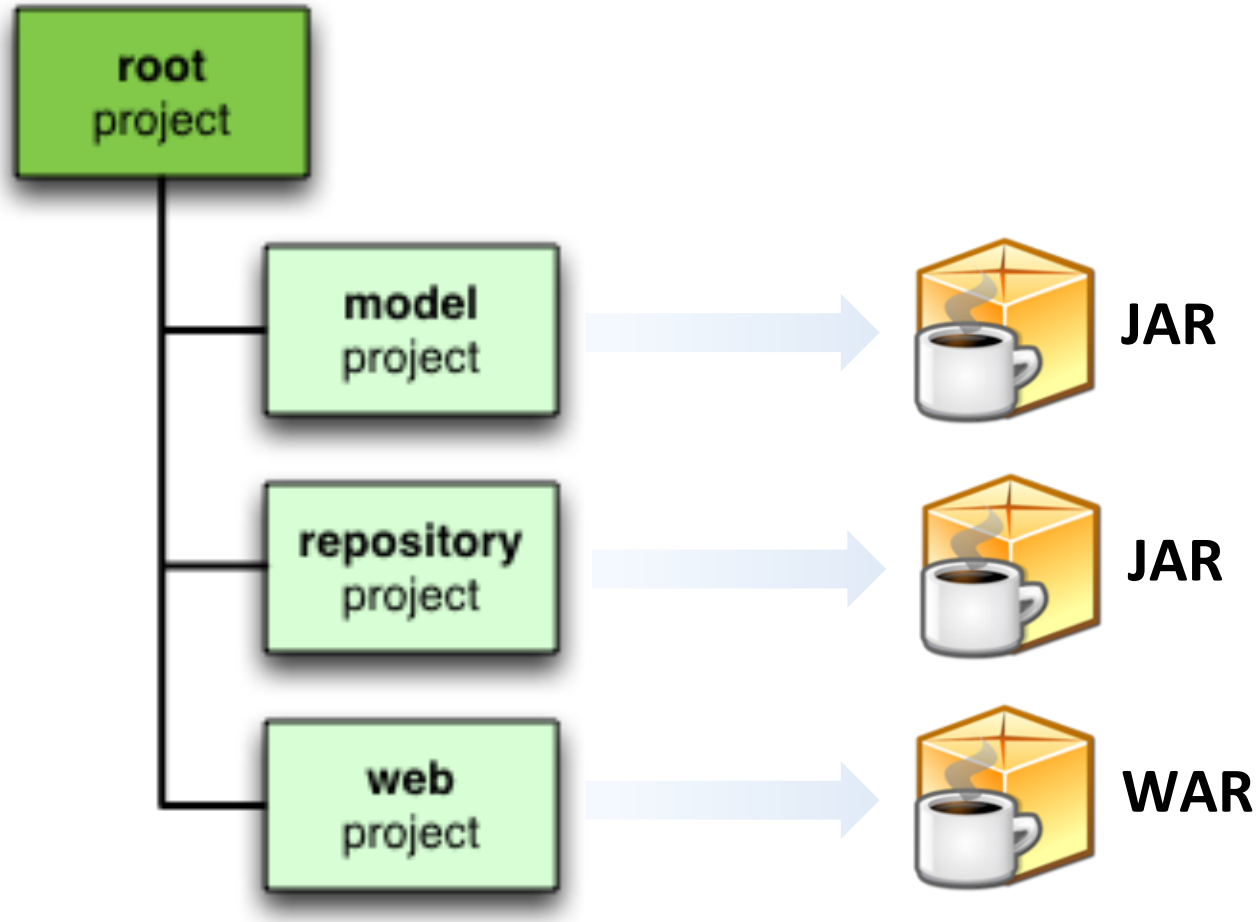


Externalize concerns into script plugins and organize them in a dedicated directory

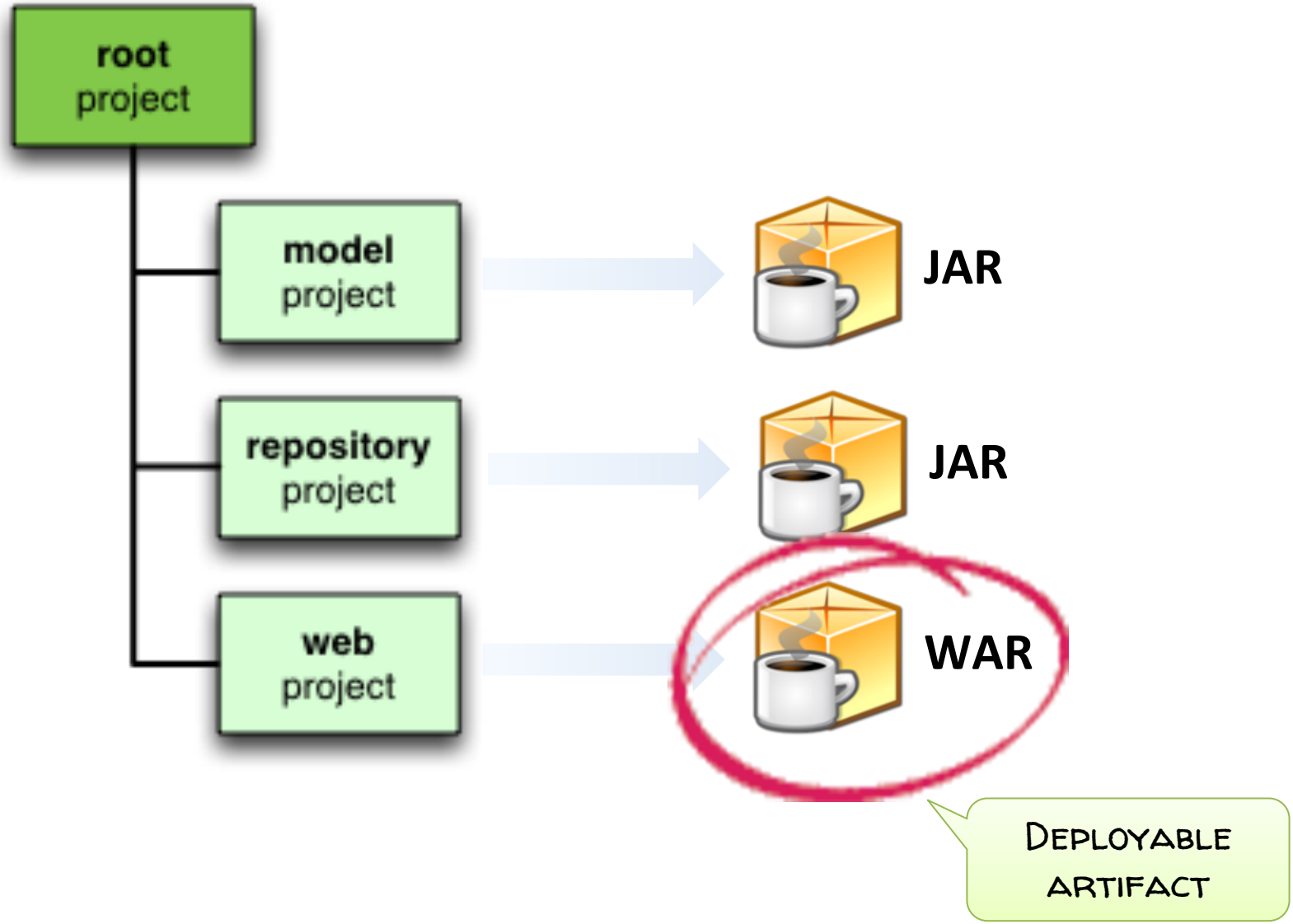
Examples:

- ✓ Versioning strategy
- ✓ Integration and functional test setup
- ✓ Deployment functionality
- ✓ ...

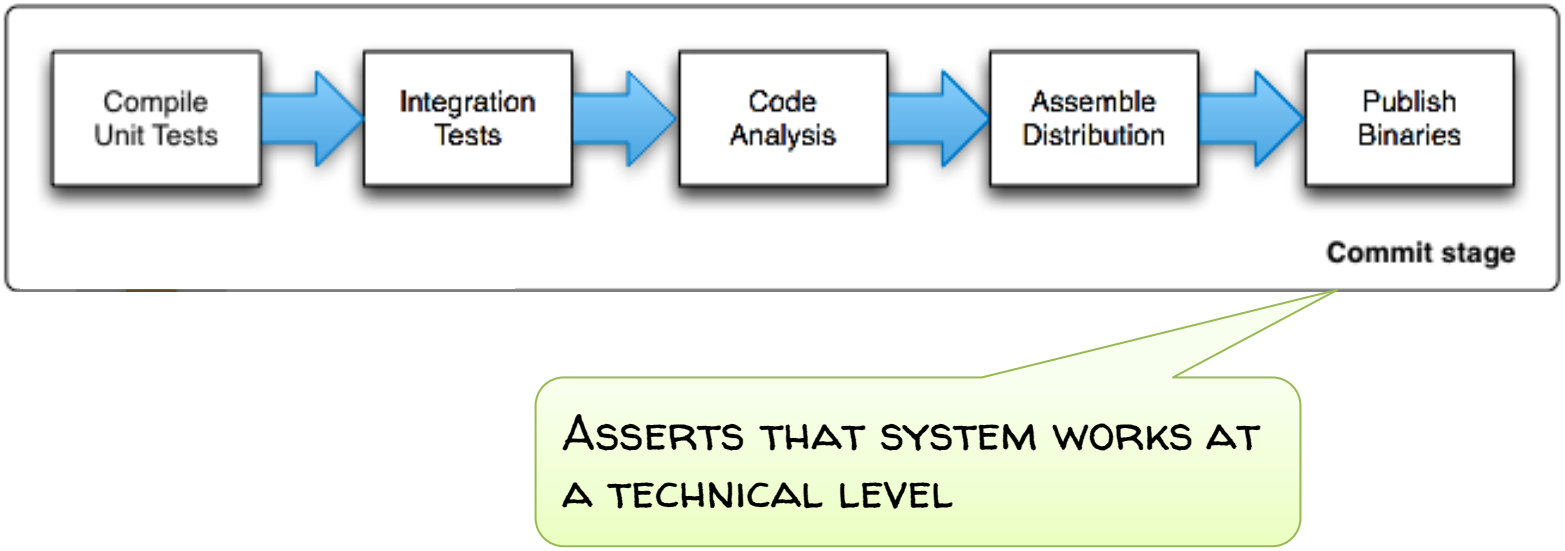
Project artifacts



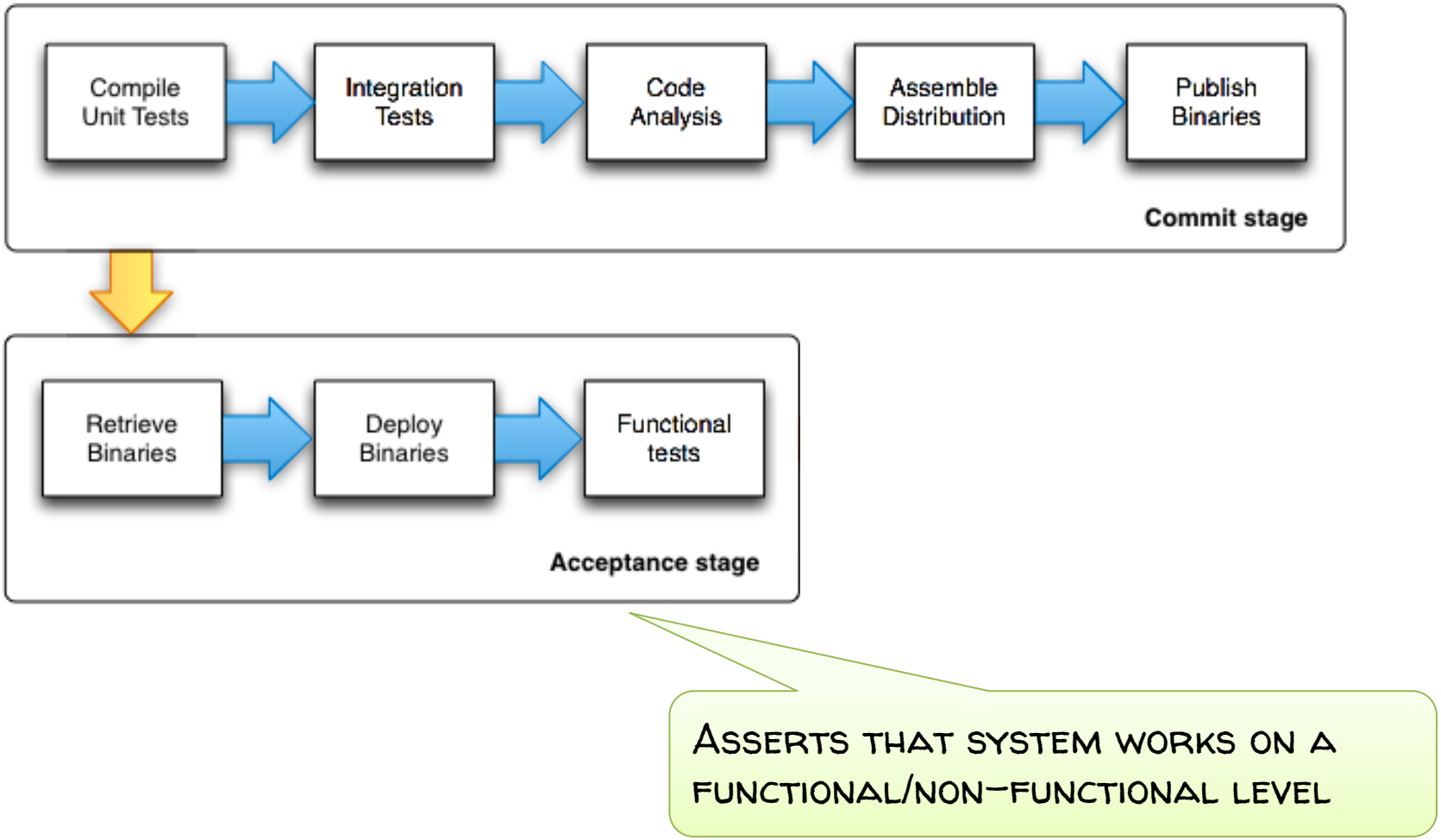
Project artifacts



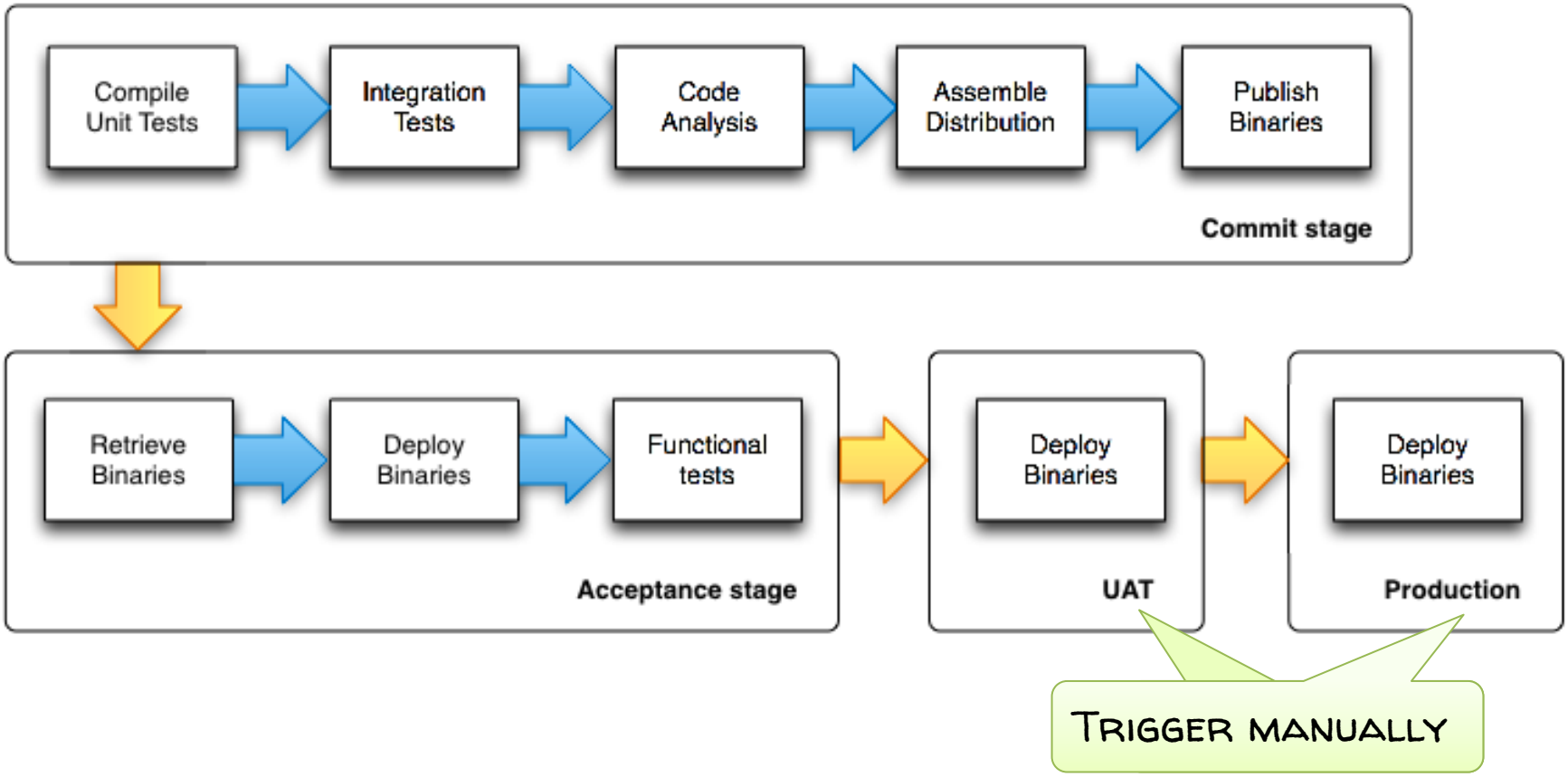
Stages in build pipeline



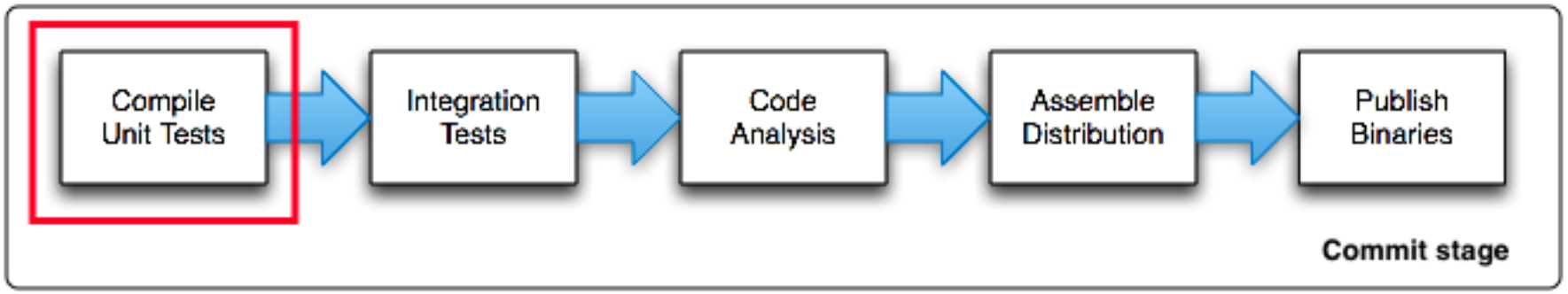
Stages in build pipeline



Stages in build pipeline

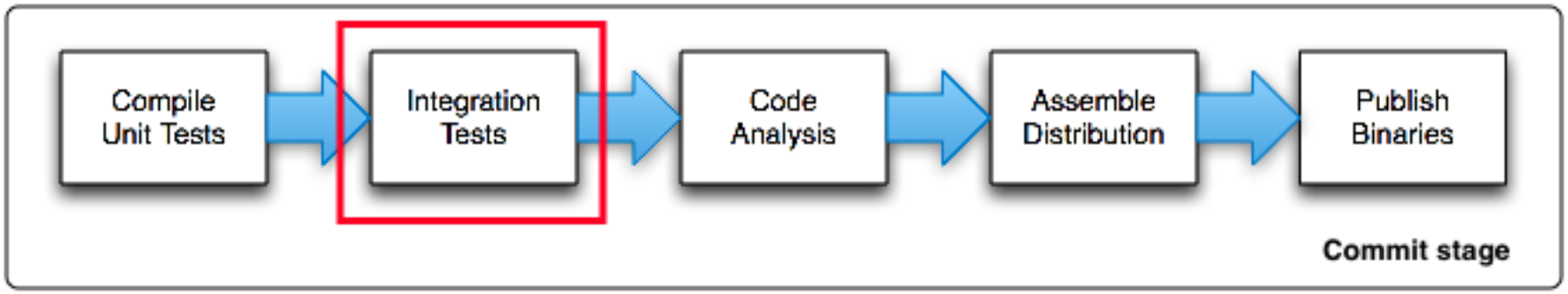


Commit stage: Compile/unit tests



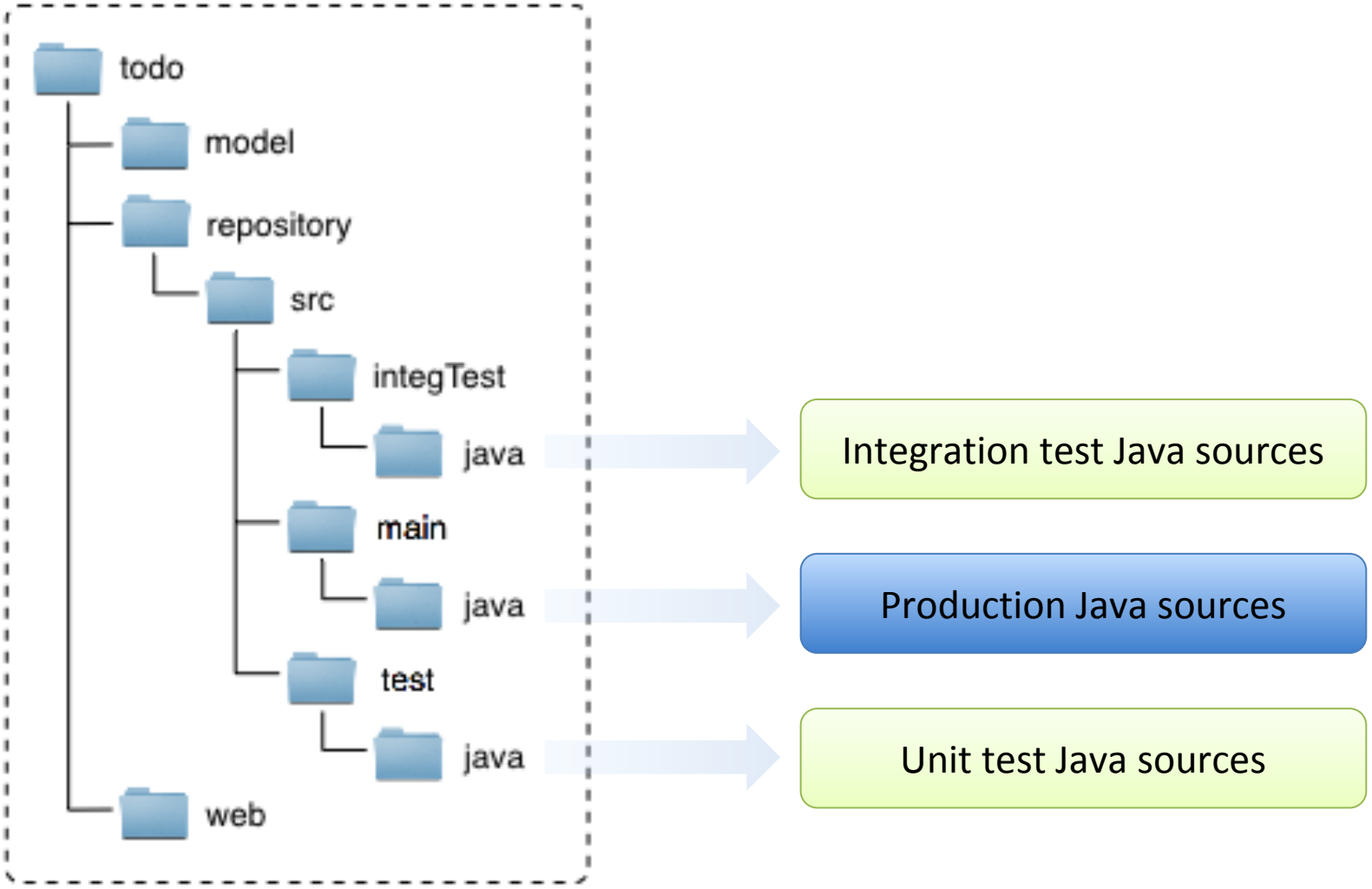
- Rapid feedback (< 5 mins)
- Run on every VCS check-in
- Priority: fix broken build

Commit stage: Integration tests



- Long running tests
- Environment setup
- Maintenance cost

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")
}
```

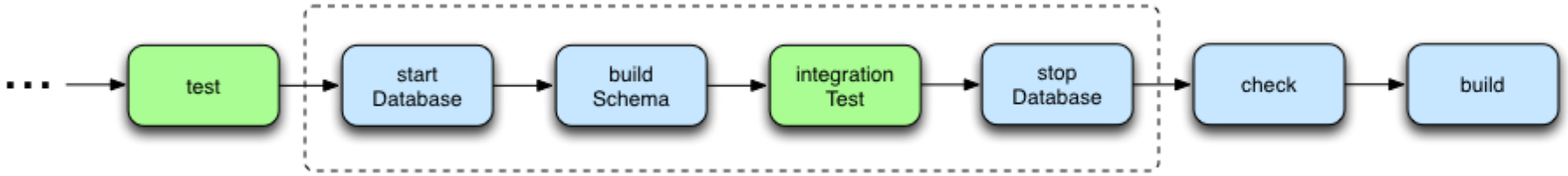
Set source and resources directory

Set compile and runtime classpath

Custom test results Directory

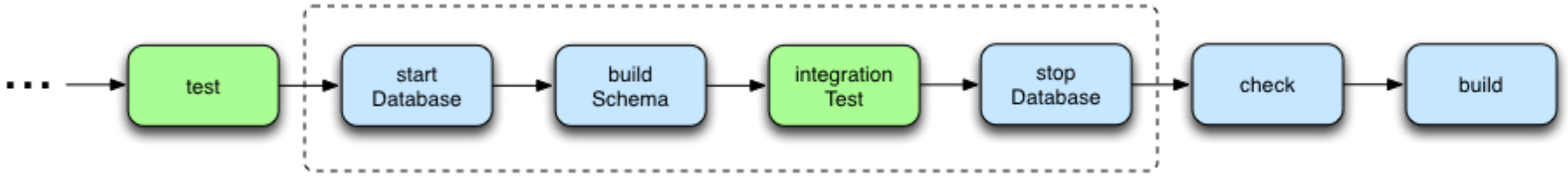
gradlew integrationTest

Database integration tests



LIQUIBASE

Database integration tests



Separate complex setup logic into script plugin

```
apply from: "$rootDir/gradle/databaseSetup.gradle"

integrationTest.dependsOn startAndPrepareDatabase
integrationTest.finalizedBy stopDatabase

check.dependsOn integrationTest
```

Integrate tasks into build lifecycle

Picking the “right” code coverage tool

Cobertura → Offline bytecode instrumentation

→ Offline bytecode instrumentation

→ Source code instrumentation

→ On-the-fly bytecode instrumentation


JACOCO
Java Code Coverage
On-the-fly bytecode instrumentation
No modification to source or bytecode

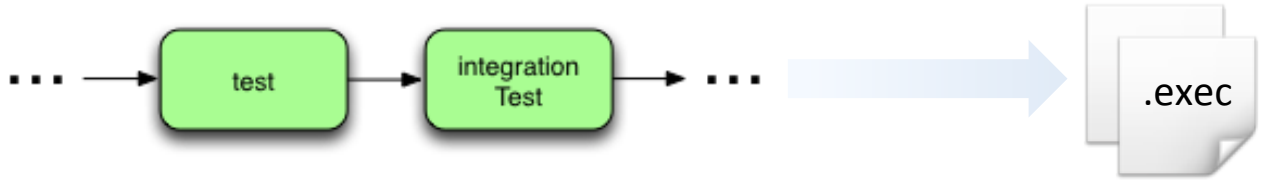
Code coverage with JaCoCo

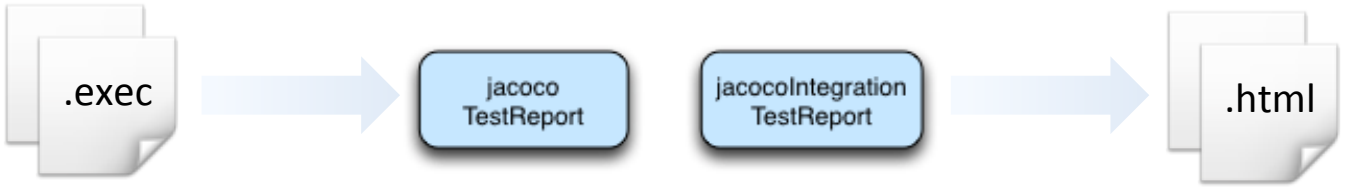
```
apply plugin: "jacoco"

test {
        jacoco {
                append = false
                destinationFile = file "$buildDir/jacoco/jacocoTest.exec"
                classDumpFile = file "$buildDir/jacoco/testDumpFile"
        }
}

integrationTest {
        jacoco {
                append = false
                destinationFile = file "$buildDir/jacoco/jacocoInt.exec"
                classDumpFile = file "$buildDir/jacoco/intDumpFile"
        }
}
```

Jacoco extension is added to all tasks with type Test

Configures instrumentation for integration tests as well

Generating coverage reports





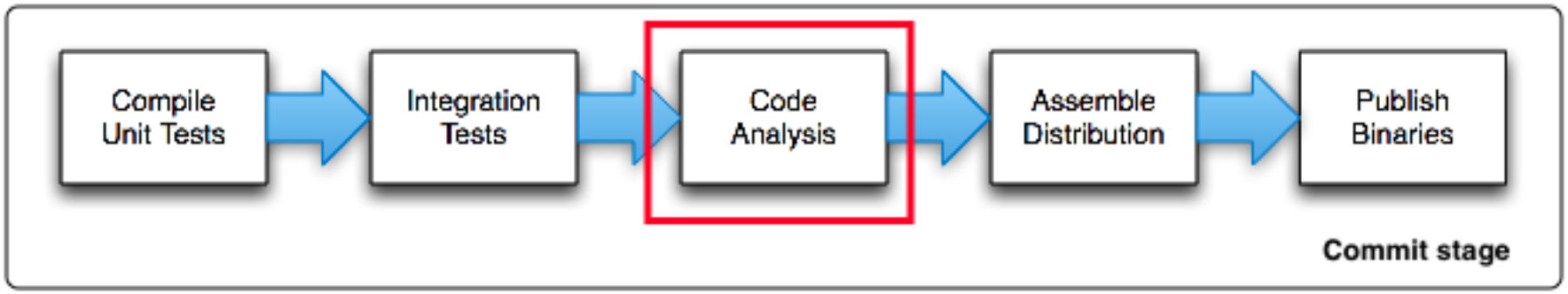
repository > com.manning.gia.todo.repository > InMemoryToDoRepository — Sessions

InMemoryToDoRepository

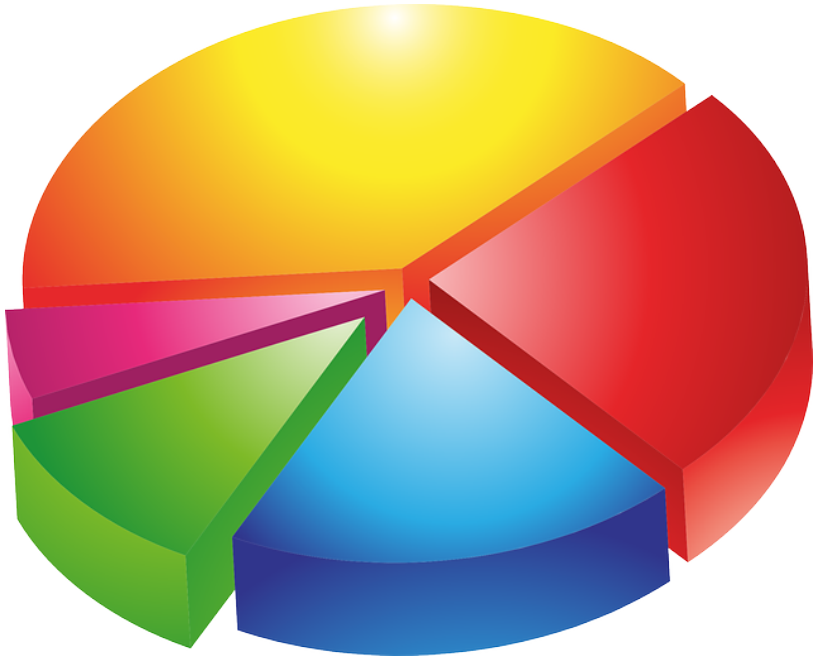
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
findAllActive()		87%		75%	1	3	0	7	0	1
findAllCompleted()		87%		100%	0	3	0	7	0	1
insert(ToDoItem)		100%		n/a	0	1	0	4	0	1
InMemoryToDoRepository()		100%		n/a	0	1	0	3	0	1
findAll()		100%		n/a	0	1	0	3	0	1
update(ToDoItem)		100%		n/a	0	1	0	2	0	1
delete(ToDoItem)		100%		n/a	0	1	0	2	0	1
findById(Long)		100%		n/a	0	1	0	1	0	1
Total	10 of 139	93%	1 of 8	88%	1	12	0	29	0	8

Created with JaCoCo 0.6.2.201302030002

Commit stage: Code analysis



- Perform code health check
- Fail build for low quality
- Record progress over time



Static code analysis tools


```
apply plugin: 'pmd'

pmd {
    ignoreFailures = true
}

tasks.withType(Pmd) {
    reports {
        xml.enabled = false
        html.enabled = true
    }
}
```

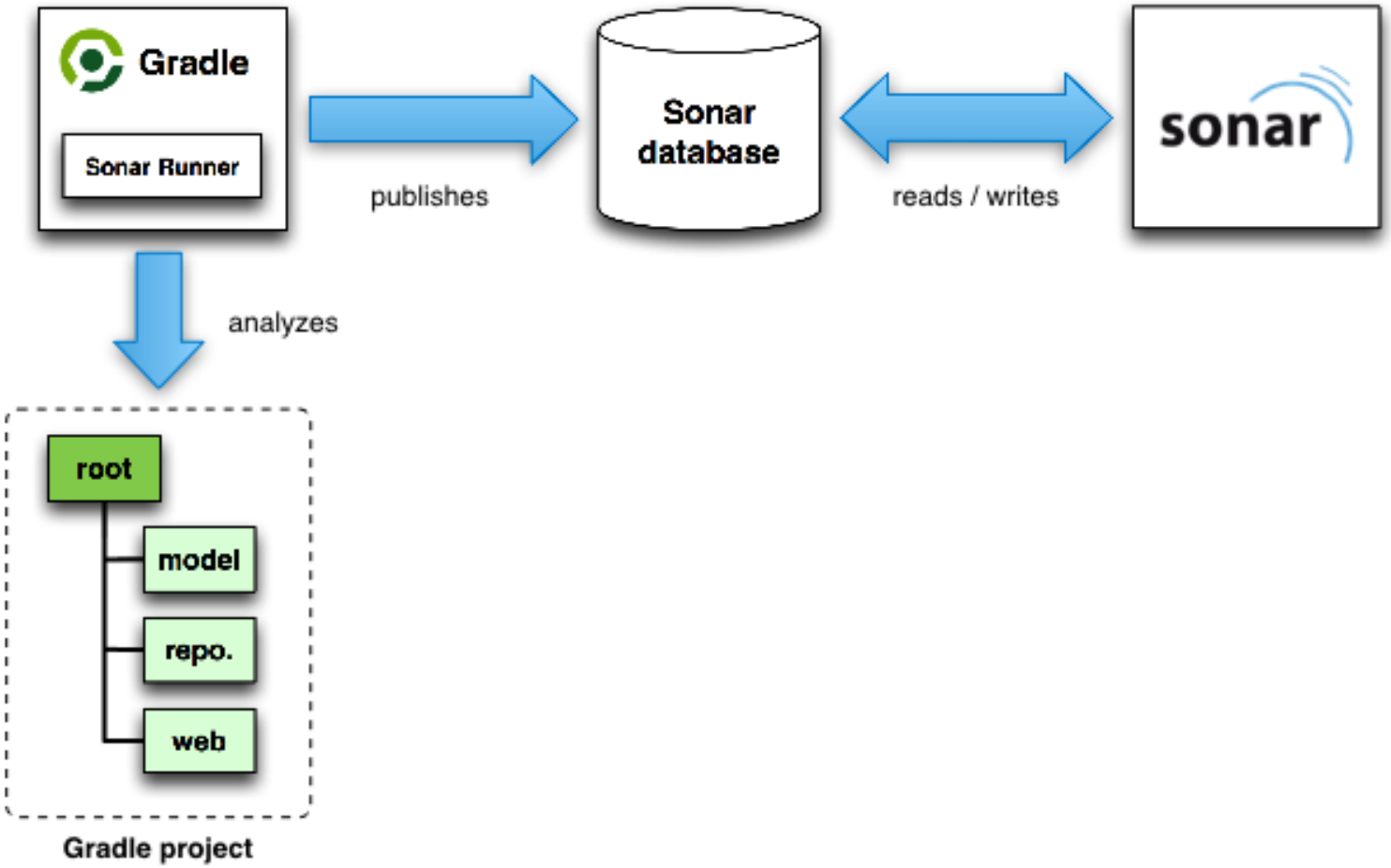
FindBugs

Checkstyle


```
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.projectName', 'todo'
        property 'sonar.projectDescription', 'A task management
application'
    }
}

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

gradlew sonarRunner

Dashboards | Projects | Measures | Issues | Rules | Quality Profiles | Quality Gates | Log in | Search

- Helicopter View
- Languages Panel
- Activity
- SonarQube Ecosystem

TOOLS

- Dependencies
- Compare

sonarqube

Sonar as a Service for your project with CloudBees

ALL PROJECTS

SQALE Rating: A

Technical Debt Ratio: 9.9%

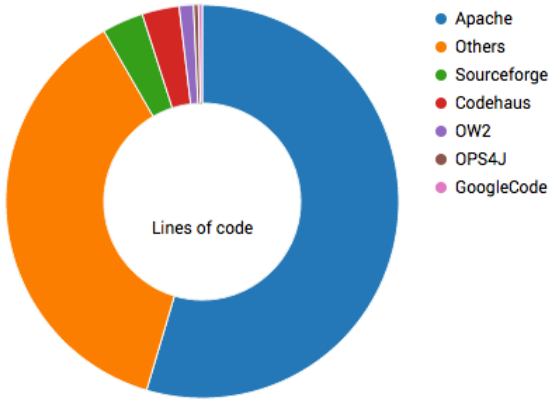
Technical Debt	Lines Of Code
89,887d	14,696K

ALL PROJECTS

Technical Debt	Issues
89,887d	2,800,295

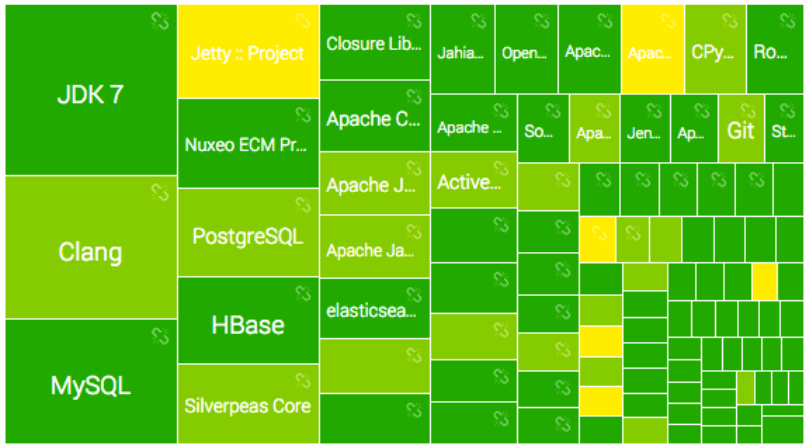
Blocker	63,482
Critical	50,660
Major	2,111,025
Minor	504,329
Info	70,799

FORGES



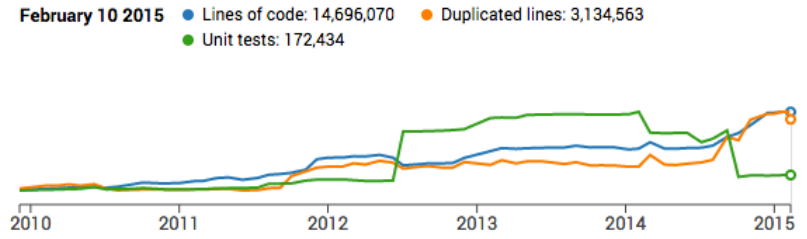
ALL PROJECTS

Size: Lines of code Color: SQALE Rating

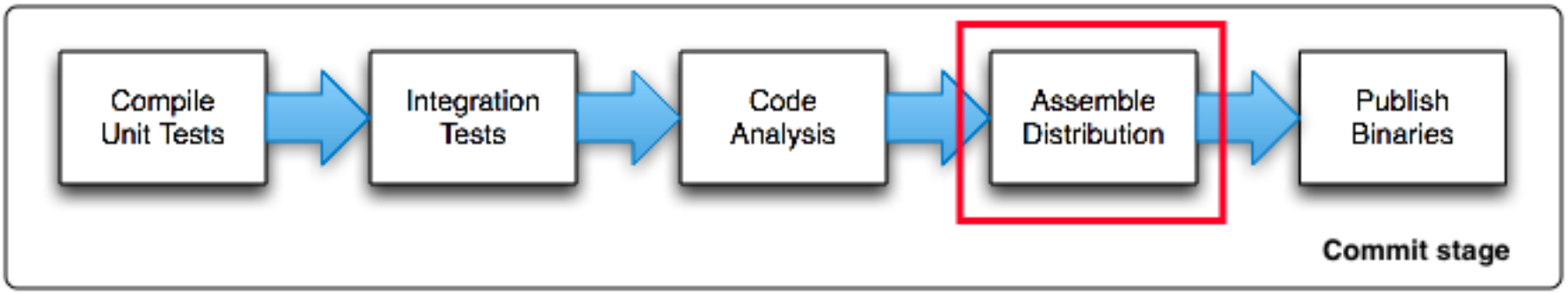


Only the first 100 components are displayed

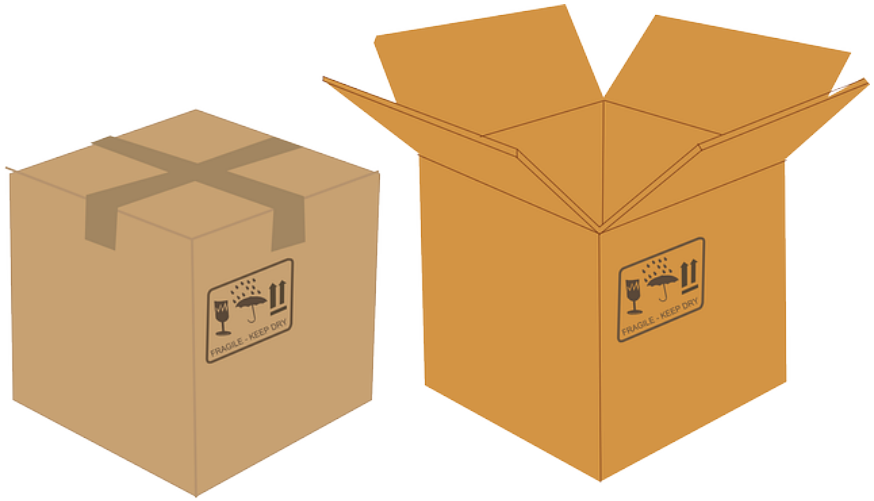
ALL PROJECTS



Commit stage: Assemble distribution

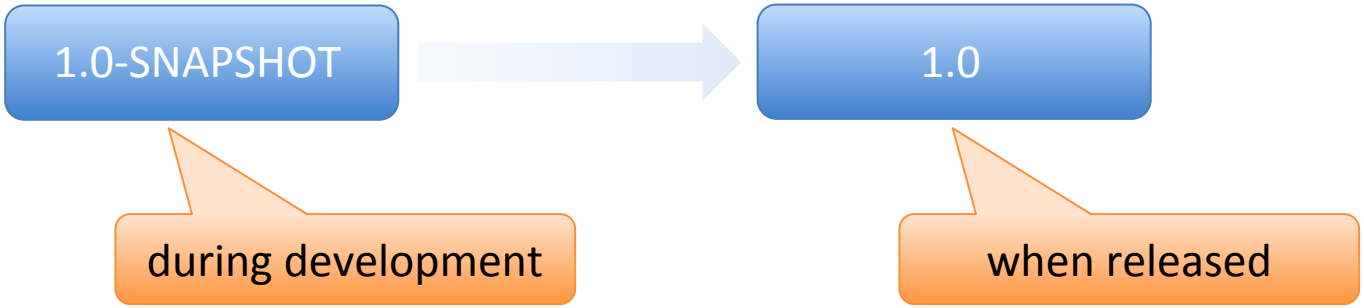


- Exclude env. configuration
- Include build information
- Versioning strategy



Versioning strategy

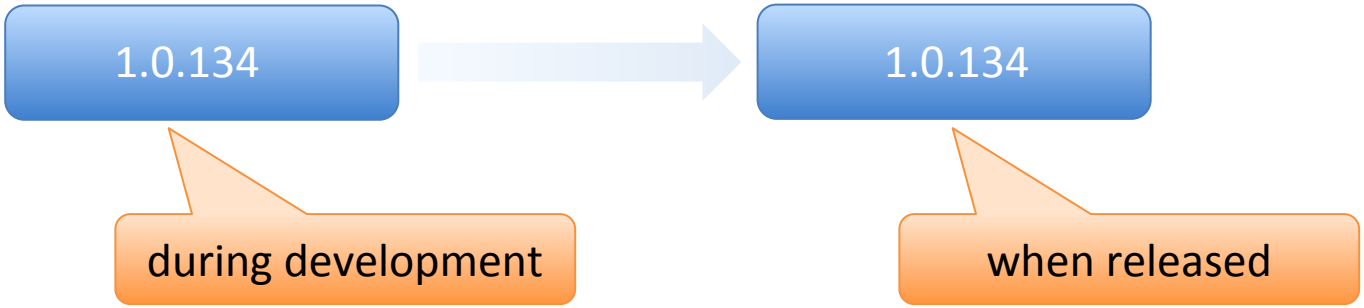
…the Maven way



Change version with Maven Release plugin

Versioning strategy

…the Continuous Delivery way



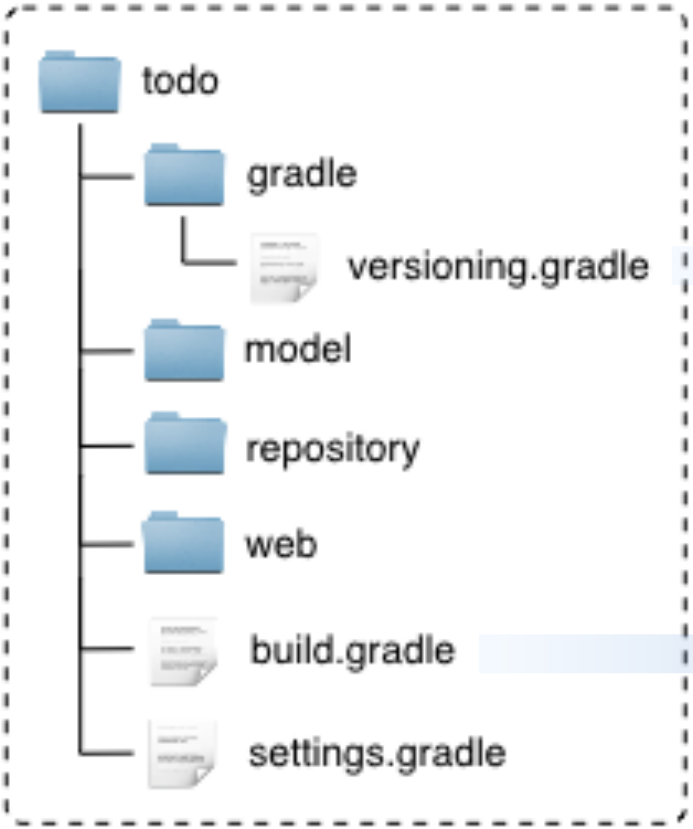
1.0.134

Project version number

Jenkins build number

Versioning strategy

…implemented with Gradle



Contains version implementation

```
allprojects {
    apply from: "$rootDir/gradle/versioning.gradle"
}
```

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

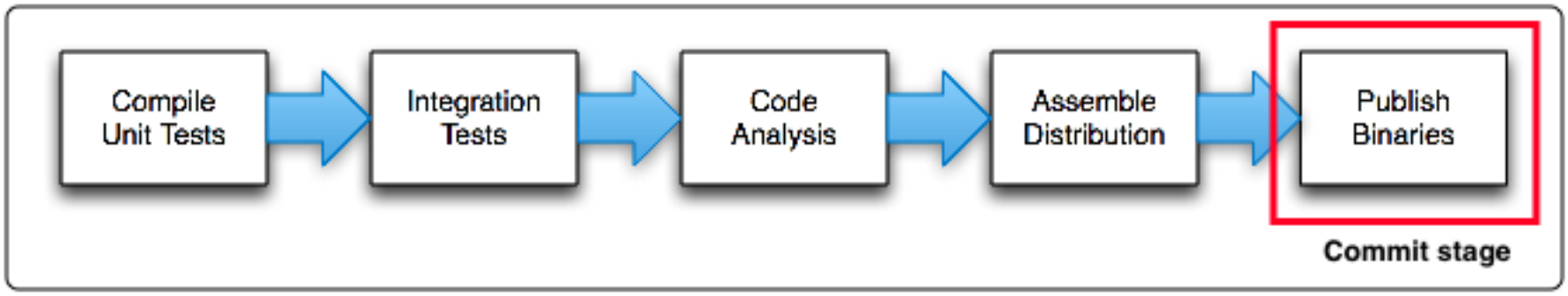
```
project(':web') {
    apply plugin: 'war'

    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')
        }
    }
}
```

Creates file containing build information

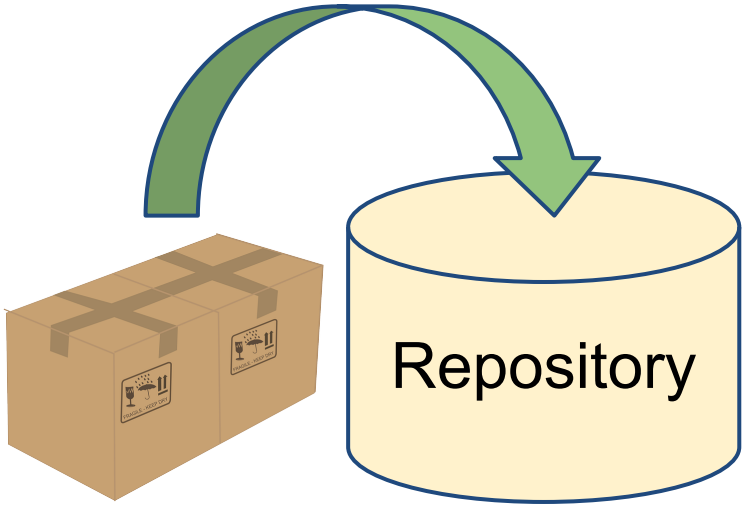
Include build info file Into WAR distribution

gradlew assemble

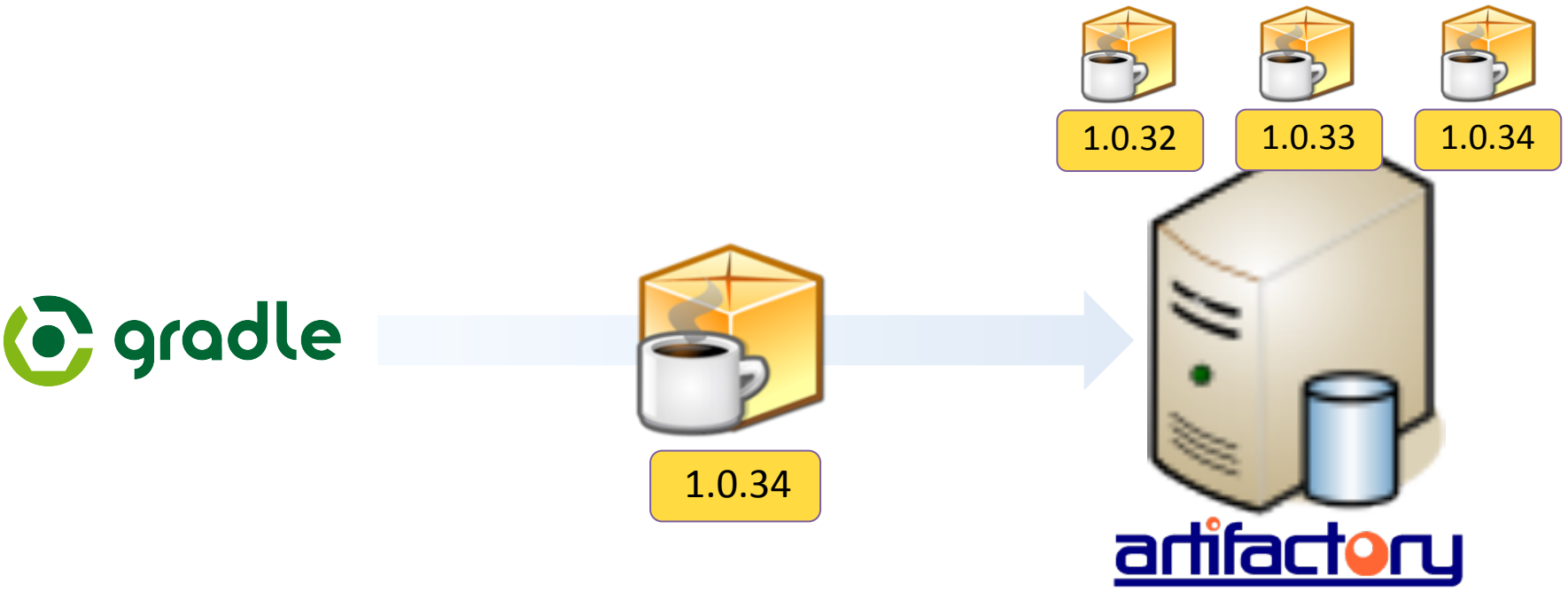
Commit stage: Publish binaries



- Version artifact(s)
- Use binary repository
- Publish once, then reuse



Publishing the deployable artifact



Defining build configuration

Common configuration

Env-specific configuration

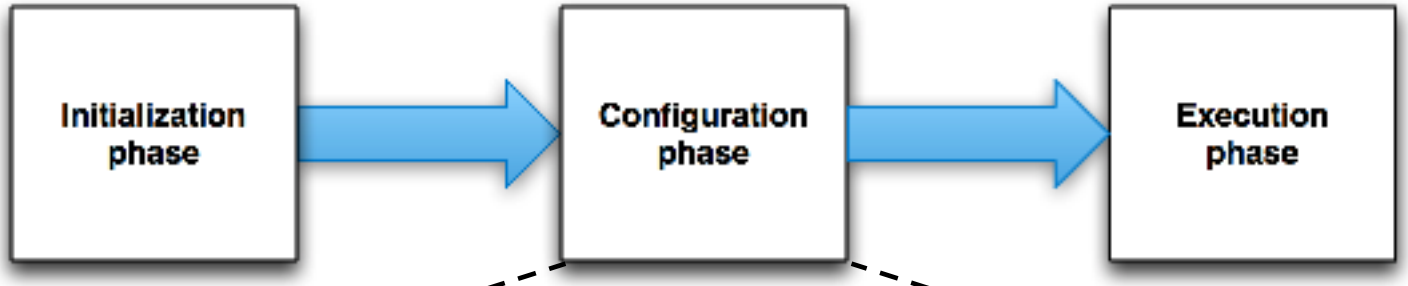
```
binaryRepository {
    url = 'http://mycompany.bin.repo:8081/artifactory'
    username = 'admin'
    password = 'password'
    name = 'libs-release-local'
}

environments {
    test {
        server {
            hostname = 'mycompany.test'
            port = 8099
            context = 'todo'
            username = 'manager'
            password = 'manager'
        }
    }
    uat {
        server {
            hostname = 'mycompany.uat'
            port = 8199
            context = 'todo'
            username = 'manager'
            password = 'manager'
        }
    }
    ...
}
```

Read credentials from gradle.properties

Read credentials from gradle.properties

Read credentials from gradle.properties

Reading build configuration




```
def env = project.hasProperty('env') ? project.getProperty('env') : 'test'
logger.quiet "Loading configuration for environment '$env'."

def configFile = file("$rootDir/gradle/config/buildConfig.groovy")
def parsedConfig = new ConfigSlurper(env).parse(configFile.toURL())

allprojects {
    ext.config = parsedConfig
}
```

Assign configuration to extra property

gradlew –Penv=uat ...

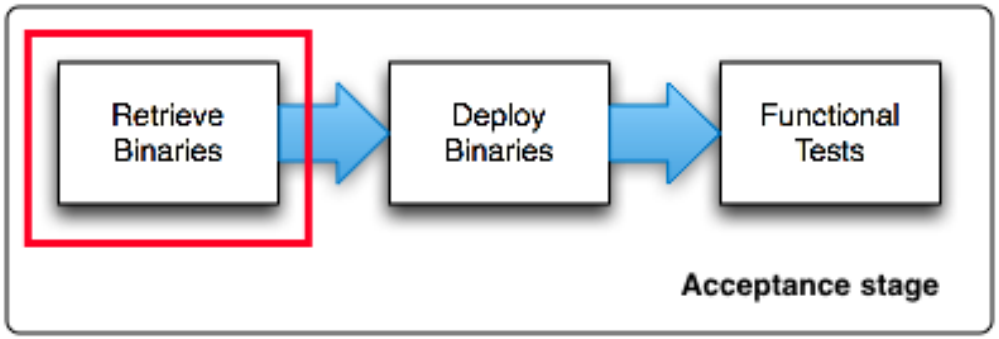
Using the Maven Publishing plugin

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
            }
        }
    }
}
```

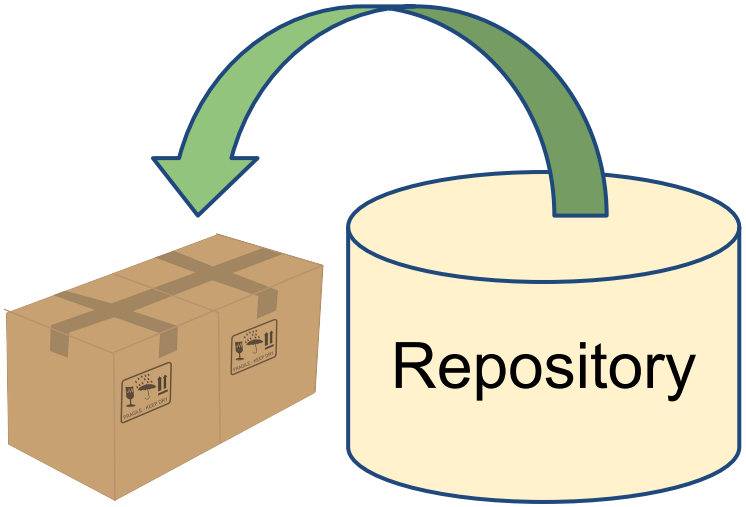
gradlew publish

Acceptance stage: Retrieve binaries

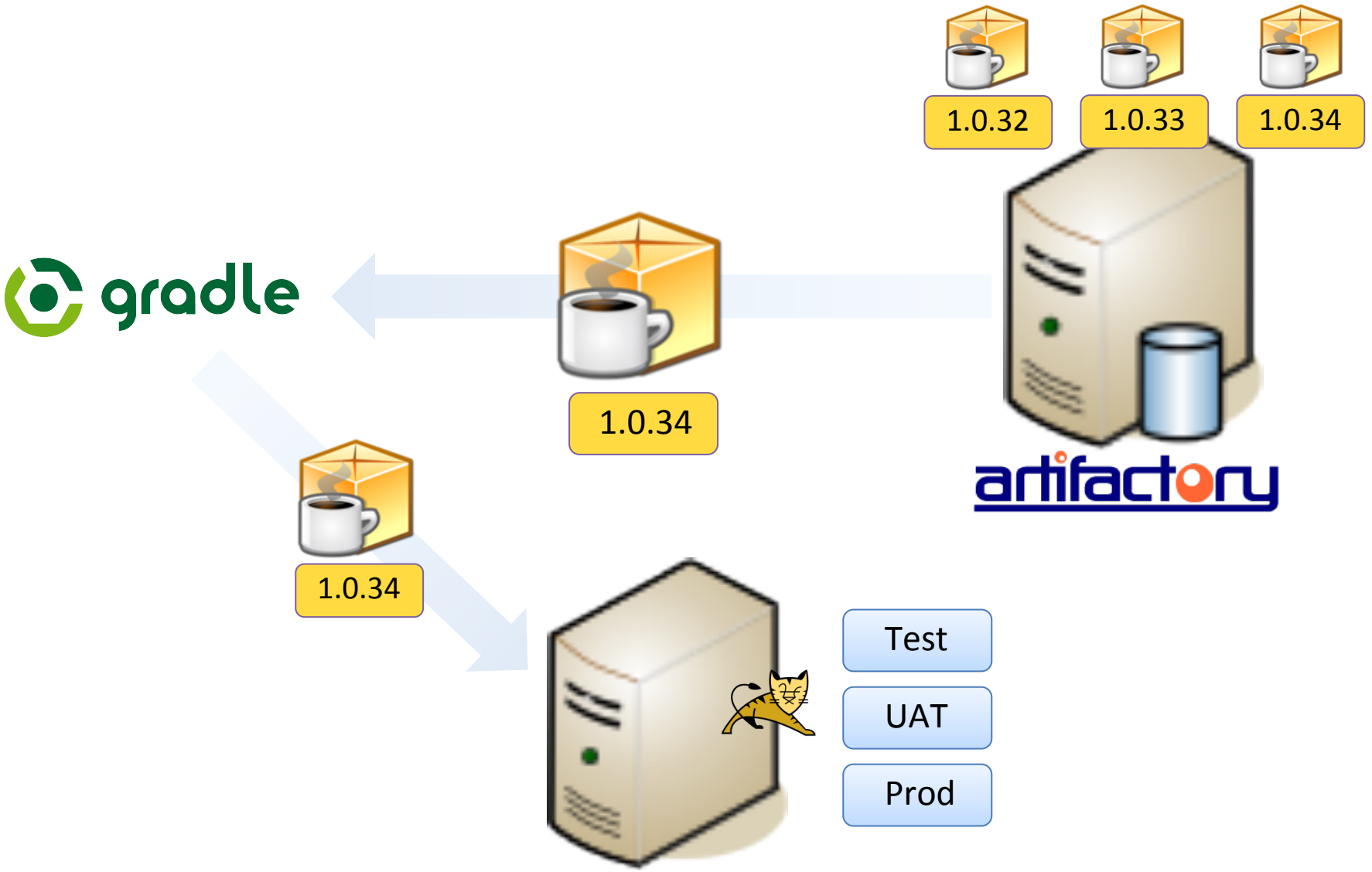


Request versioned artifact

Store in temp. directory

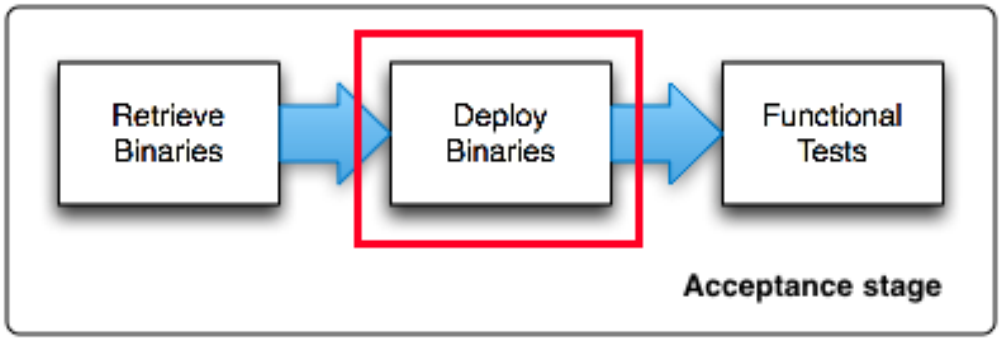


Downloading the deployable artifact



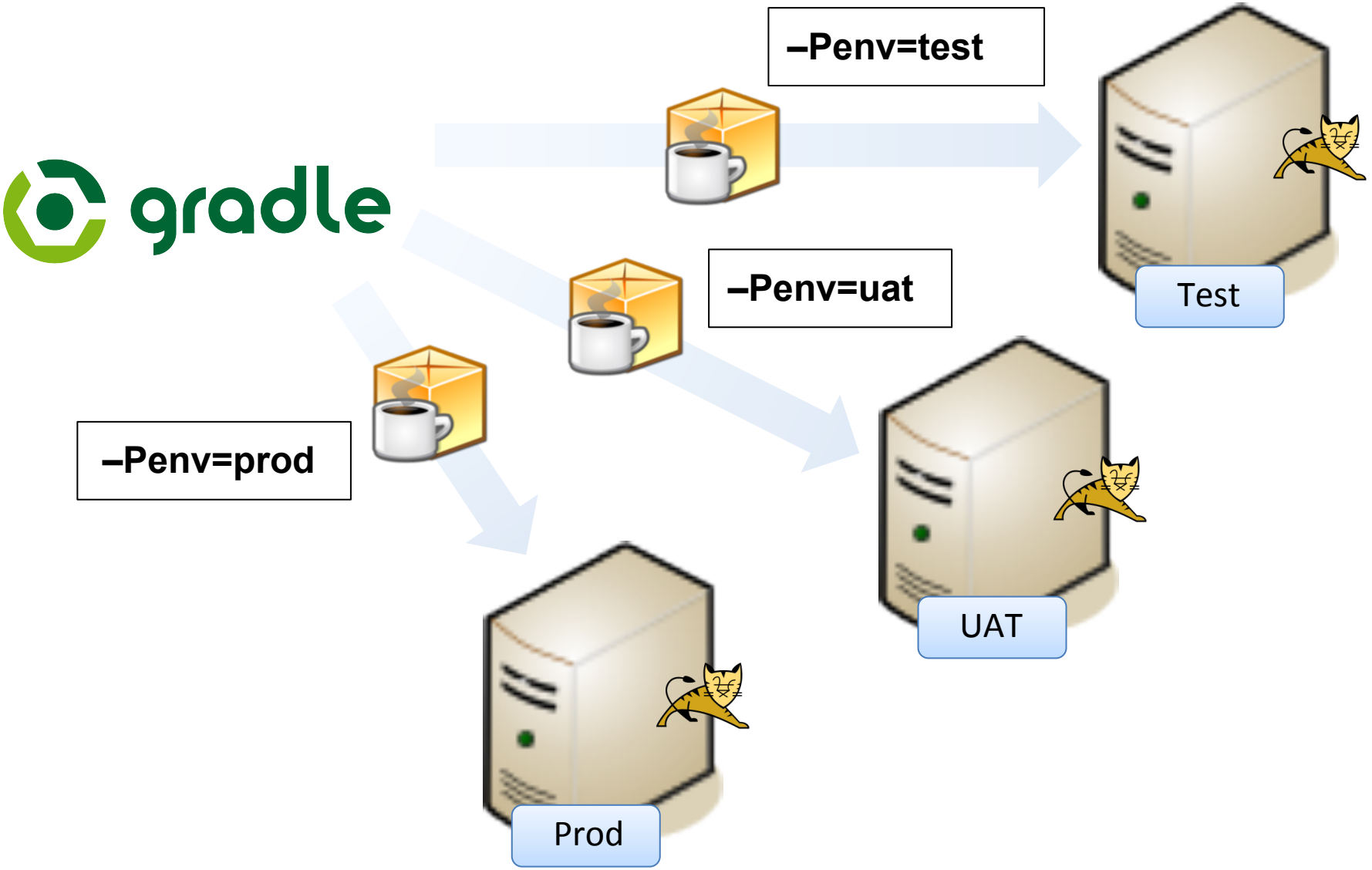
Task for downloading artifact

Acceptance stage: Deploy binaries

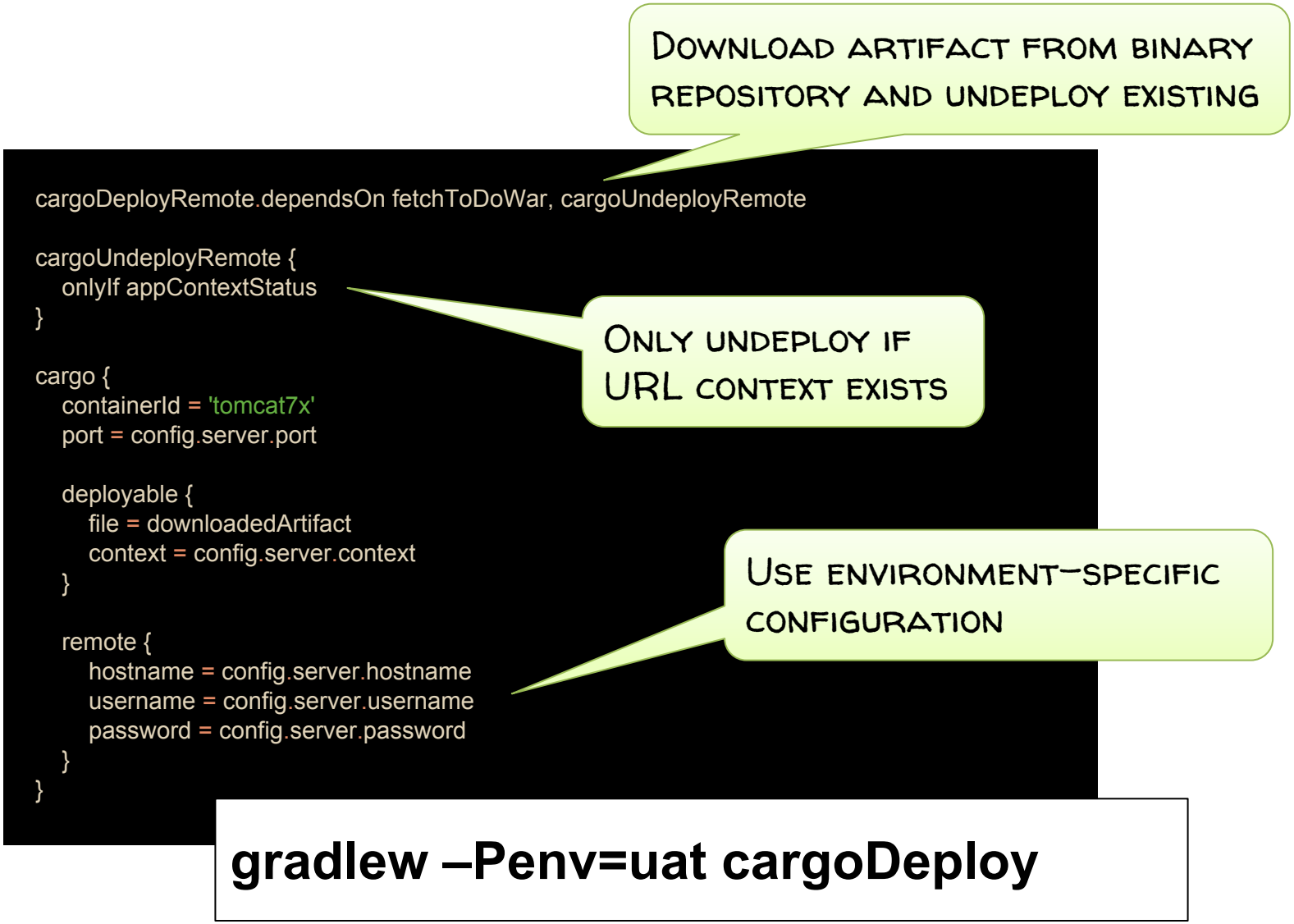


- Deployment on request
- Make it a reliable process
- Use process for all envs.

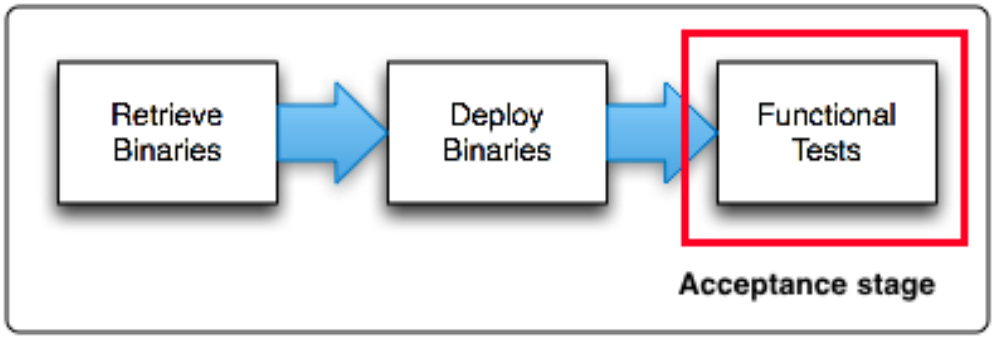
Deploying to multiple environments



Deployment with the Cargo plugin

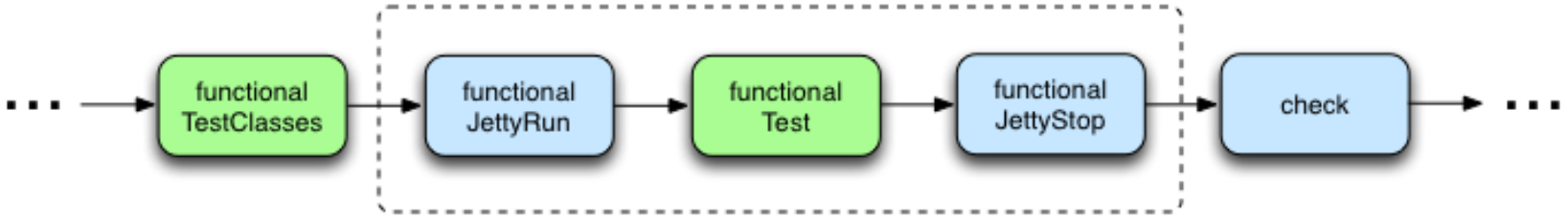


Acceptance stage: Functional tests

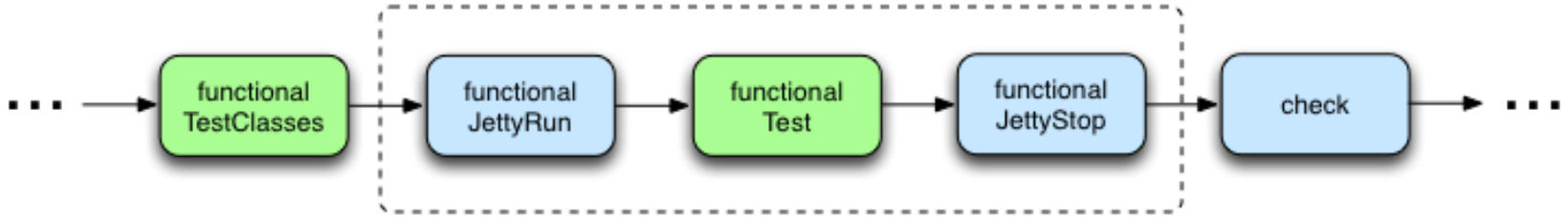


- Test all UI permutations
- Test important use cases
- Run against different envs.

In-container functional tests



Local functional tests




```
task functionalTest(type: Test) {
    ...
}

task functionalJettyRun(type: org.gradle.api.plugins.jetty.JettyRun) {
    httpPort = functionalJettyHttpPort
    stopPort = functionalJettyStopPort
    stopKey = functionalJettyStopKey
    contextPath = functionalJettyContextPath
    daemon = true
}

task functionalJettyStop(type: org.gradle.api.plugins.jetty.JettyStop) {
    stopPort = functionalJettyStopPort
    stopKey = functionalJettyStopKey
}

functionalJettyRun.dependsOn functionalTestClasses
functionalTest.dependsOn functionalJettyRun
functionalTest.finalizedBy functionalJettyStop
```

Functional test task

Custom Jetty Run task

Custom Jetty Stop task

Executing remote functional tests

```
ext {
    functionalTestReportDir = file("$testReportDir/functional")
    functionalTestResultsDir = file("$testResultsDir/functional")
    functionalCommonSystemProperties =
                ['geb.env': 'firefox',
                 'geb.build.reportsDir': reporting.file("$name/geb")]
}

task remoteFunctionalTest(type: Test) {
    testClassesDir = sourceSets.functionalTest.output.classesDir
    classpath = sourceSets.functionalTest.runtimeClasspath
    testReportDir = functionalTestReportDir
    testResultsDir = functionalTestResultsDir
    systemProperties functionalCommonSystemProperties
    systemProperty 'geb.build.baseUrl',
    "http://$config.server.hostname:$config.server.port/$config.server.context/"
}
```

Reuse setup properties

Build URL from env. configuration

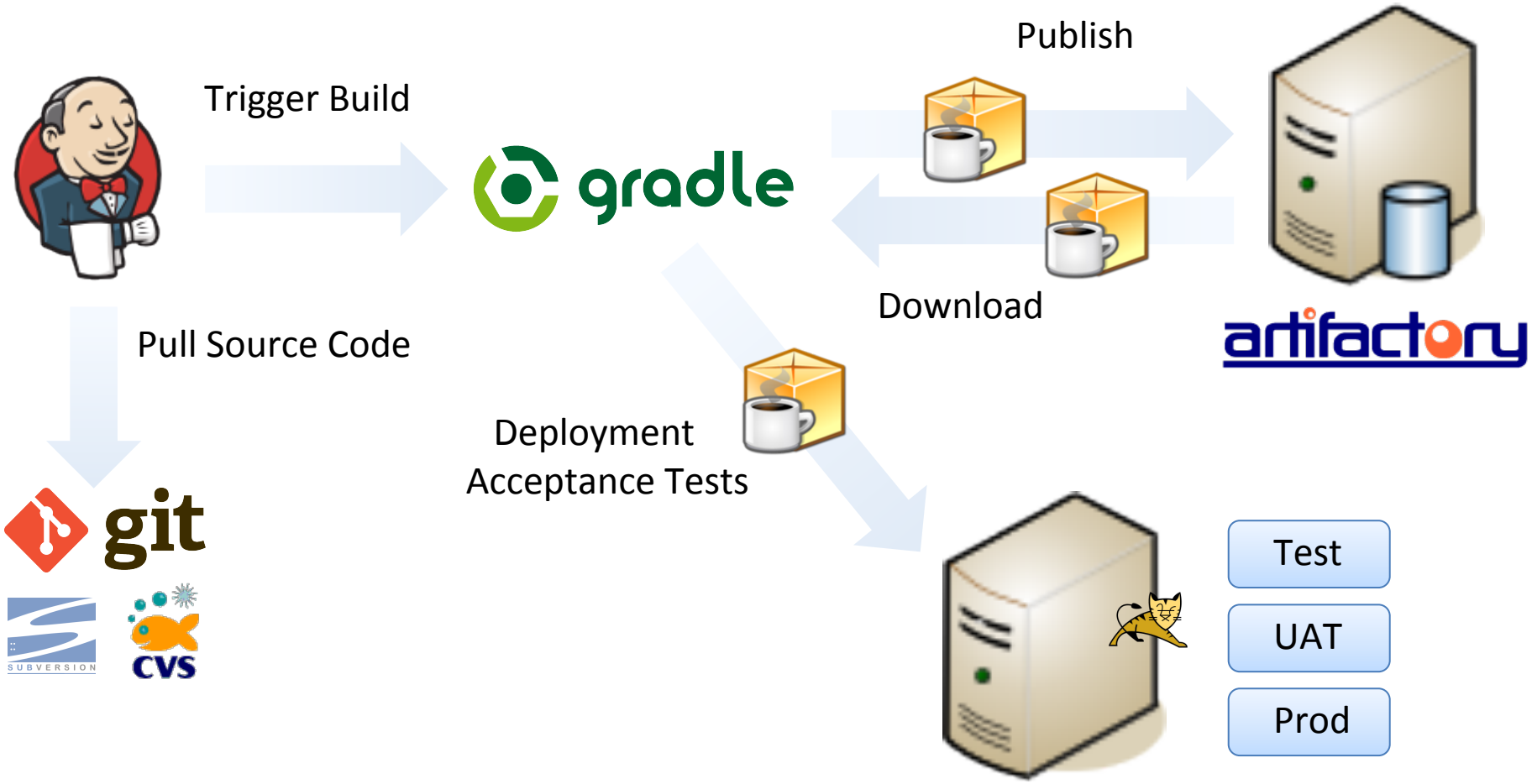
gradlew –Penv=test remoteFunctionalTest

Going further: Capacity testing

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath "com.github.kulya:jmeter-gradle-plugin:1.3.1-2.9"
    }
}

apply plugin: com.github.kulya.gradle.plugins.jmeter.JmeterPlugin

ext.loadTestResources = "$projectDir/src/loadTest/resources"

jmeterRun.configure {
    jmeterTestFiles = [file("$loadTestResources/todo-test-plan.jmx")]
    jmeterPropertyFile = file("$loadTestResources/jmeter.properties")
    jmeterUserProperties = ["hostname=${config.server.hostname},
                            "port=${config.server.port}",
                            "context=${config.server.context}"]
    logging.captureStandardError LogLevel.INFO
}
```

gradlew –Penv=uat jmeterRun

Let’s bring Jenkins into play!



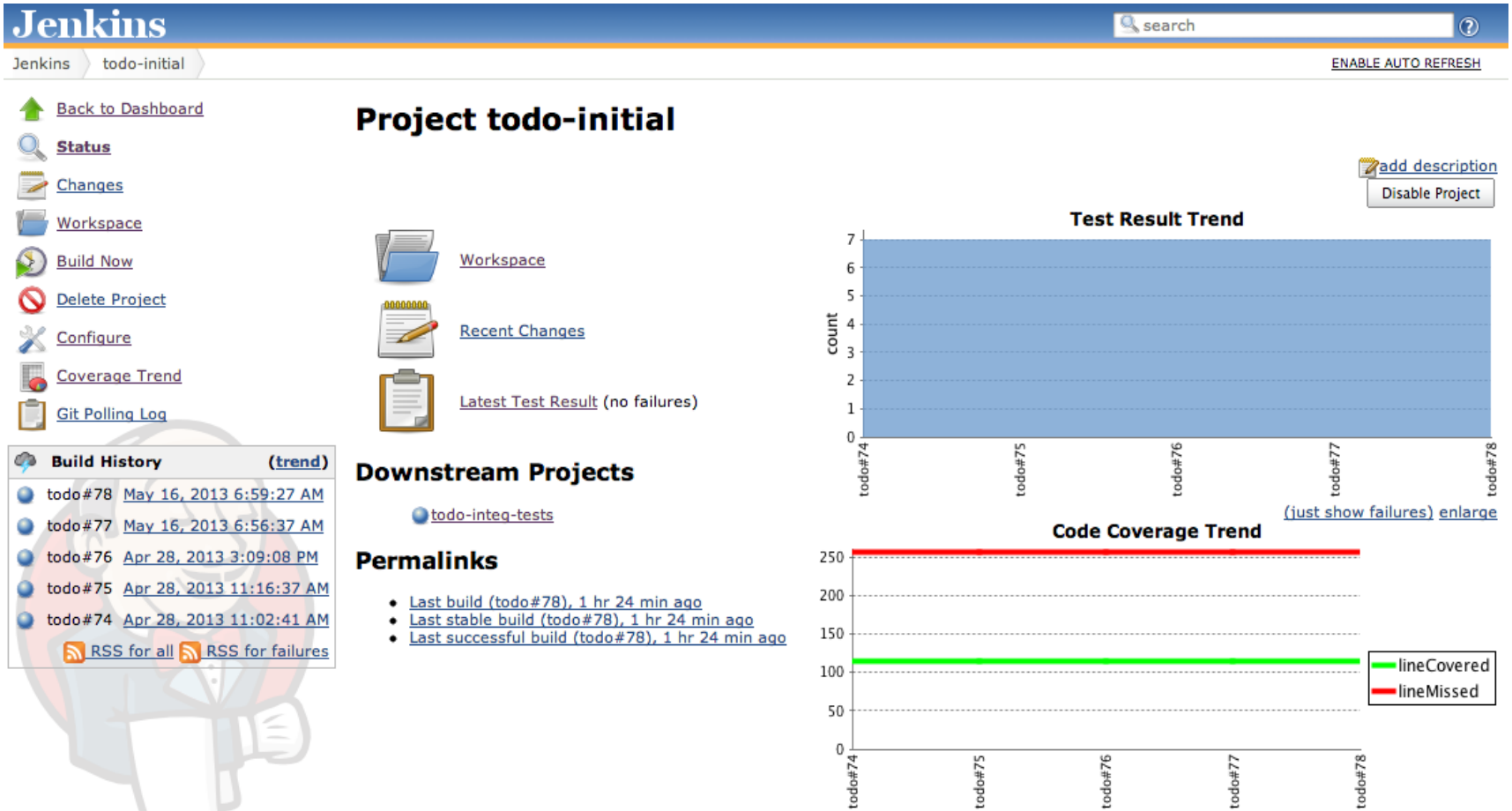
Model pipeline as series of jobs

todo-acceptance-tests	15 days (todo#76)	N/A	1 min 56 sec
todo-code-quality	15 days (todo#76)	N/A	1 min 6 sec
todo-deploy-production	N/A	N/A	N/A
todo-deploy-test	15 days (todo#76)	N/A	51 sec
todo-deploy-uat	20 days (todo#65)	16 days (todo#67)	47 sec
todo-distribution	15 days (todo#76)	29 days (todo#41)	45 sec
todo-initial	15 days (todo#76)	N/A	56 sec
todo-[illegible]nteg-tests	15 days (todo#76)	N/A	46 sec
tod[illegible]formance-tests	15 days (todo#76)	15 days (todo#74)	1 min 28 sec

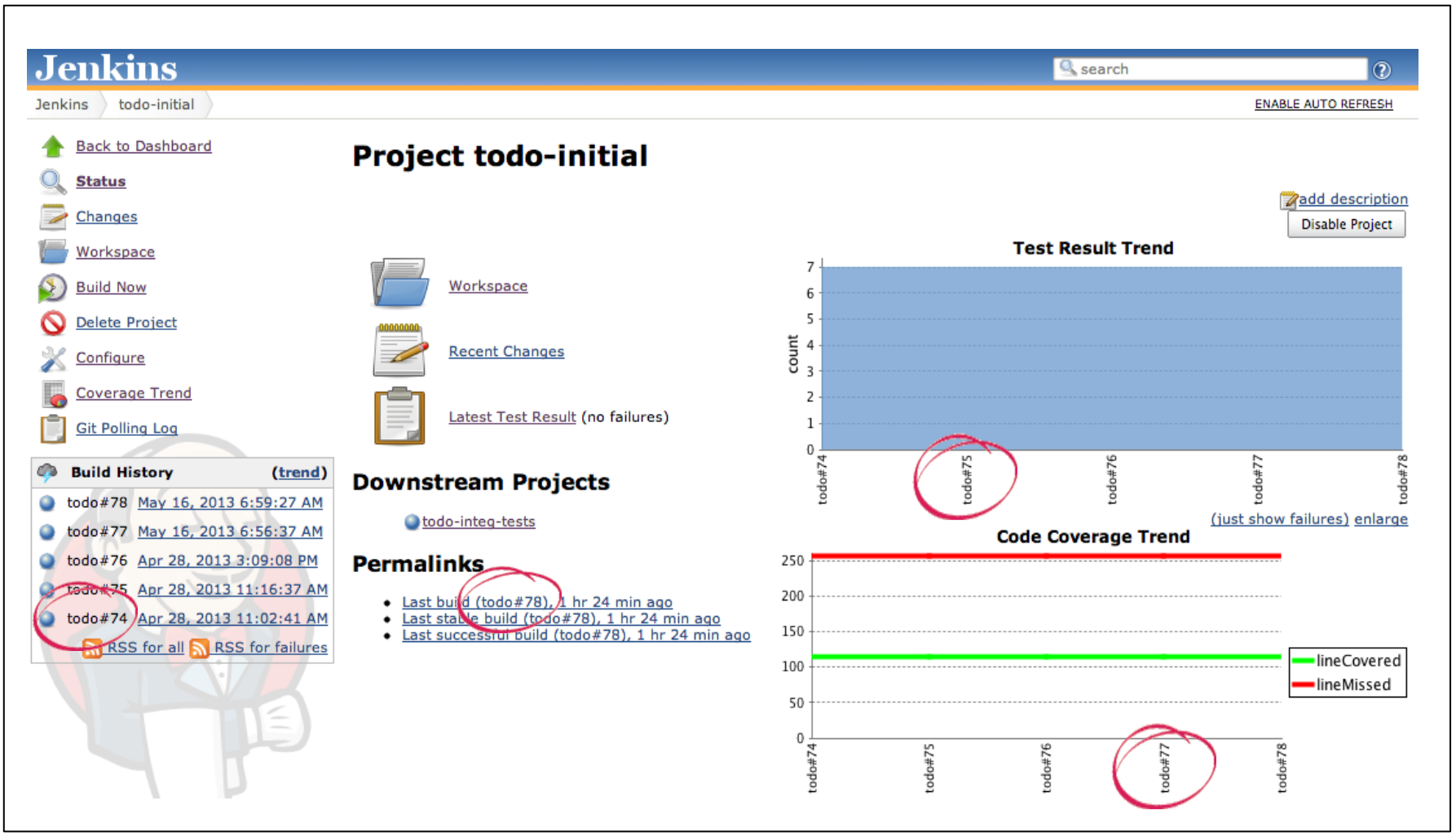
Trigger job when SCM change is detected

Jenkins

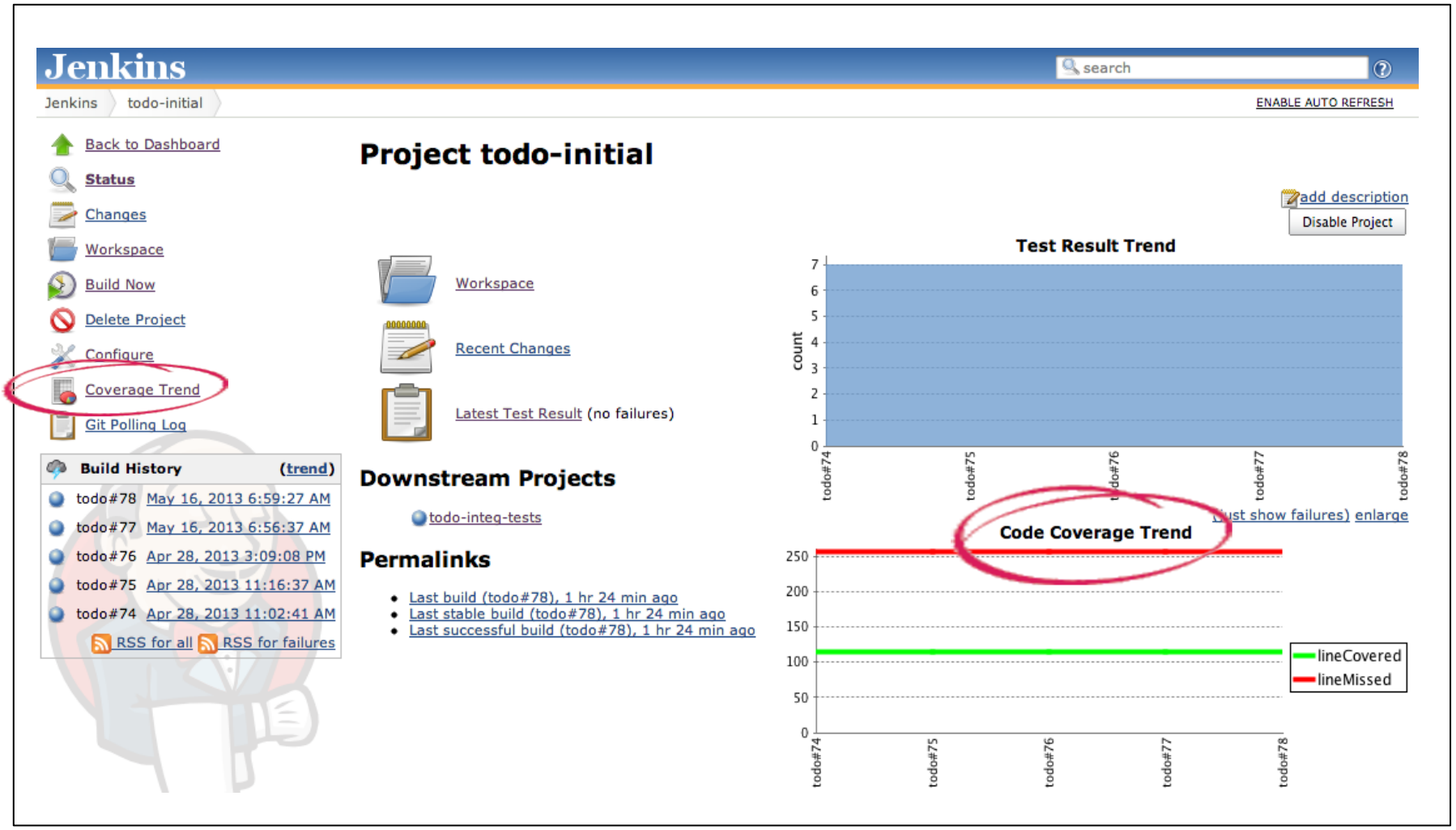
Initial Jenkins Build Job



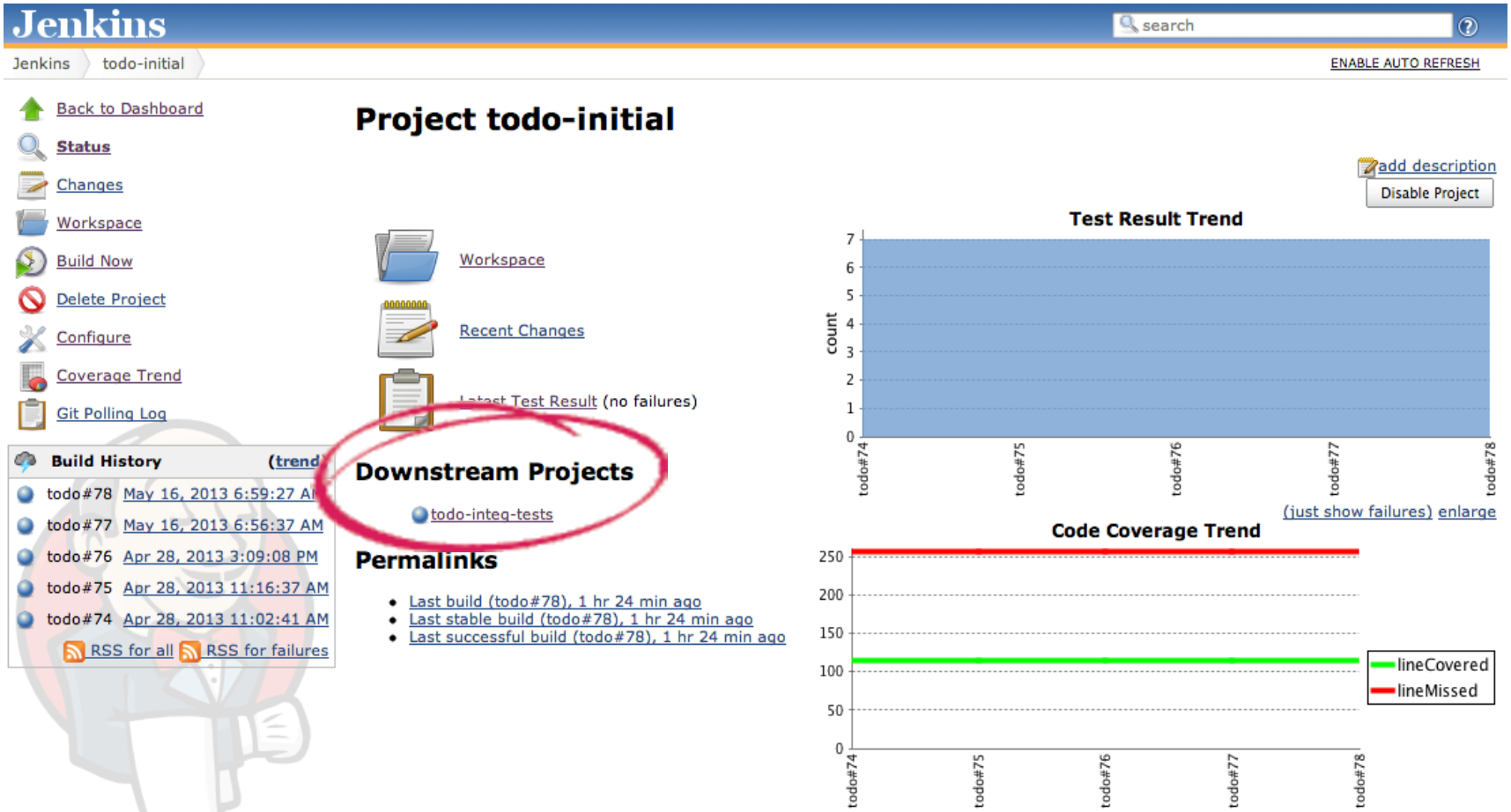
Build Name Setter Plugin



JaCoCo Plugin



Parameterized Trigger Plugin



Build

Invoke Gradle script

- Invoke Gradle
- Use Gradle Wrapper

Make gradlew executable

From Root Build Script Dir

Build step description

Switches

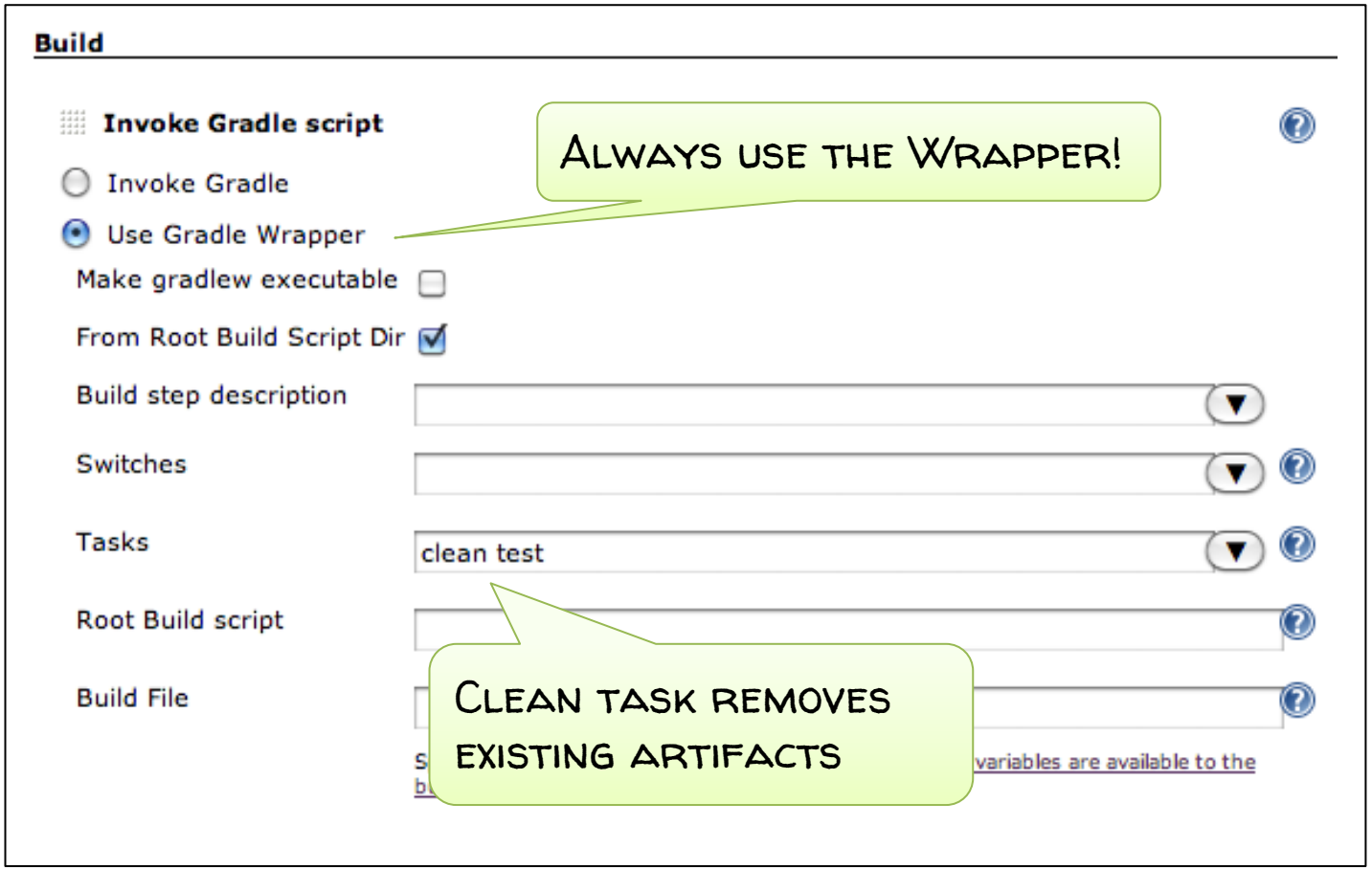
Tasks: clean test

Root Build script

Build File

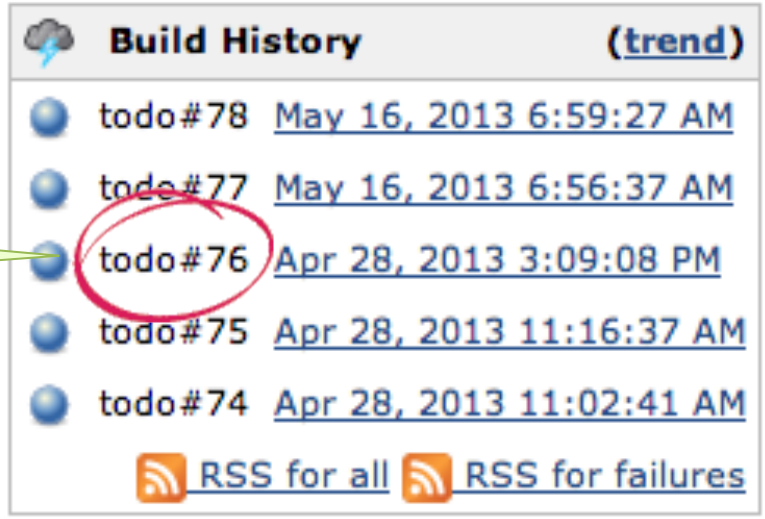
Specify Gradle build file to run. Also, some environment variables are available to the build script

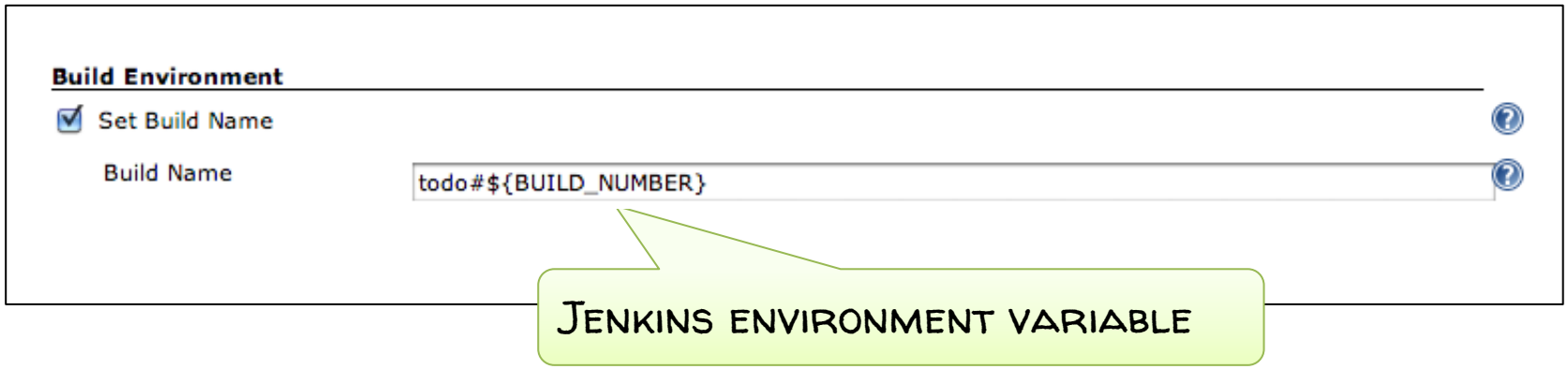
Gradle Plugin



Build Name Setter Plugin

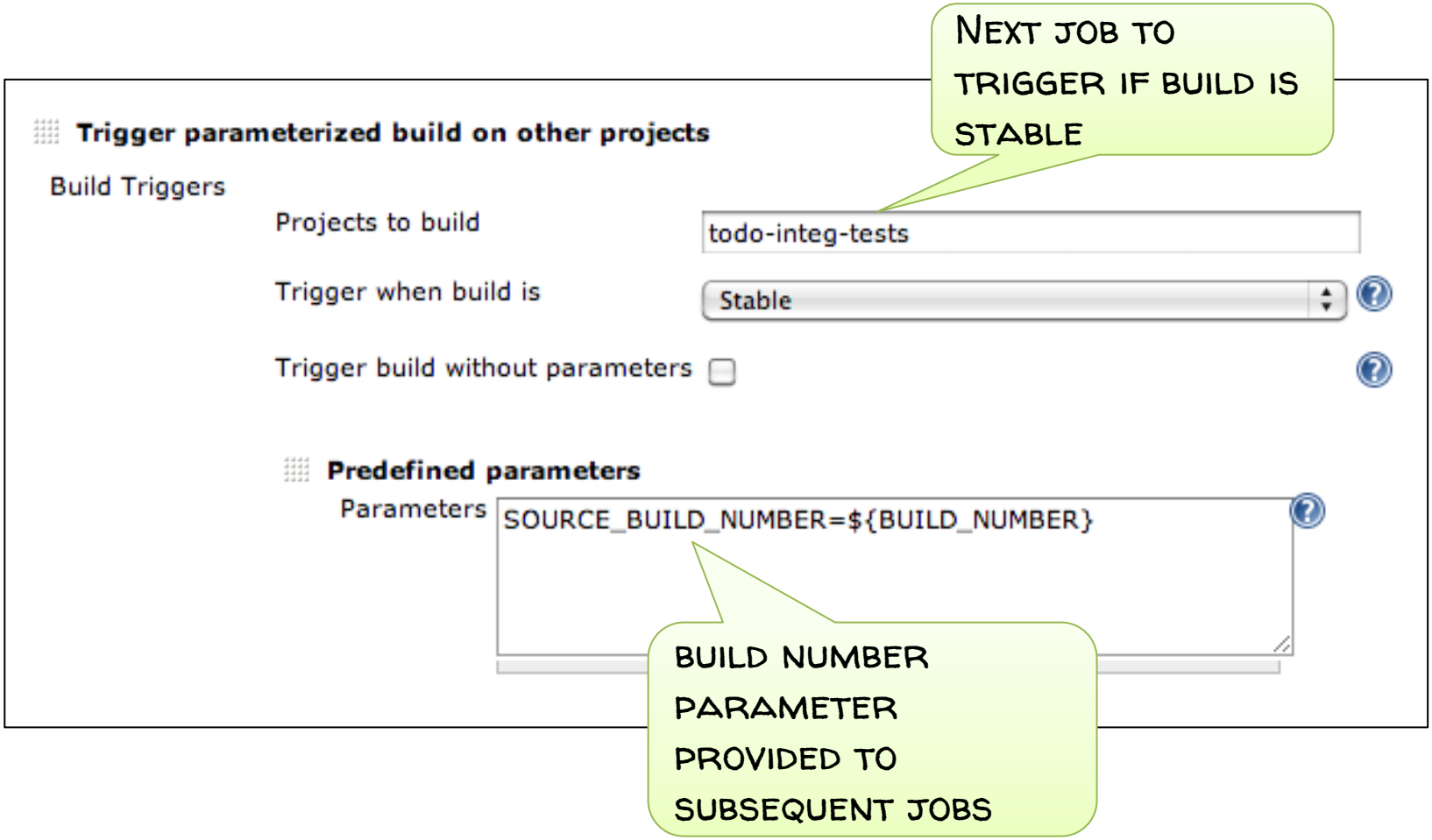
Expressive build name



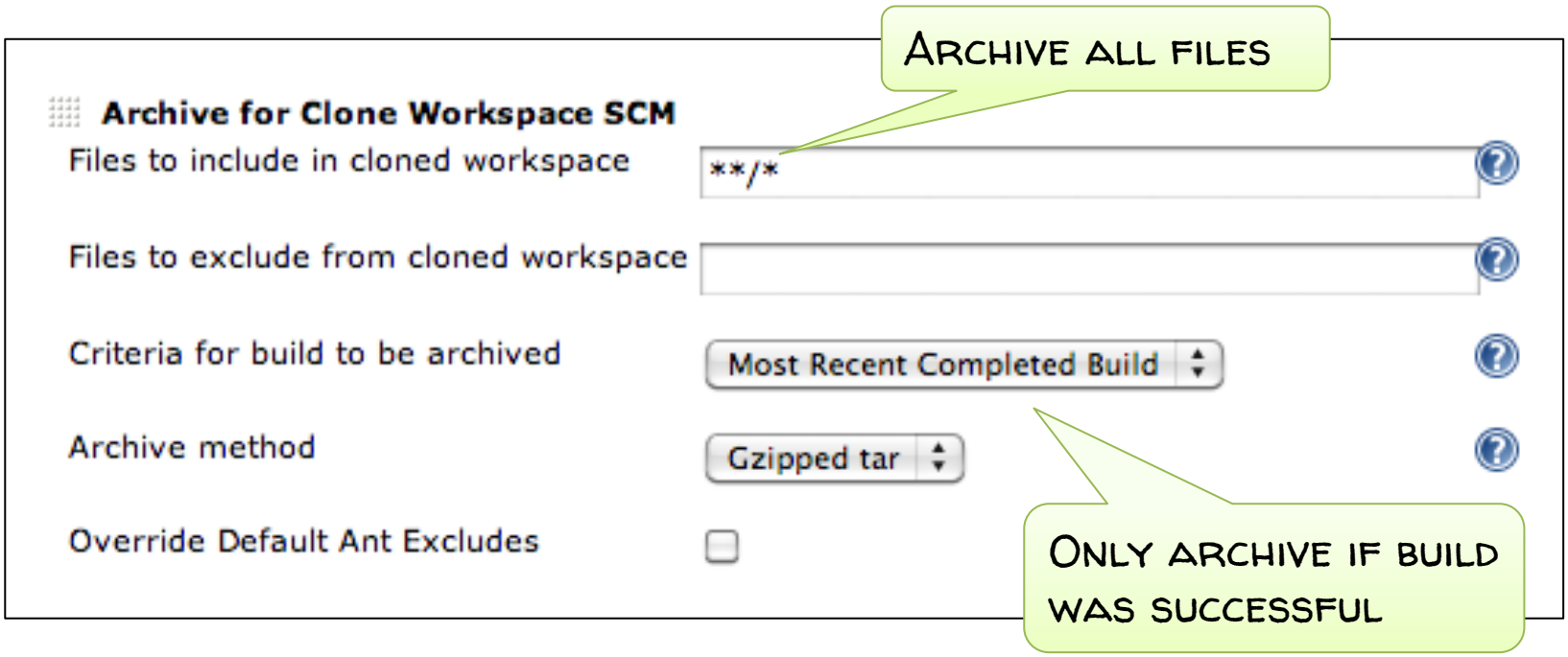


Jenkins environment variable

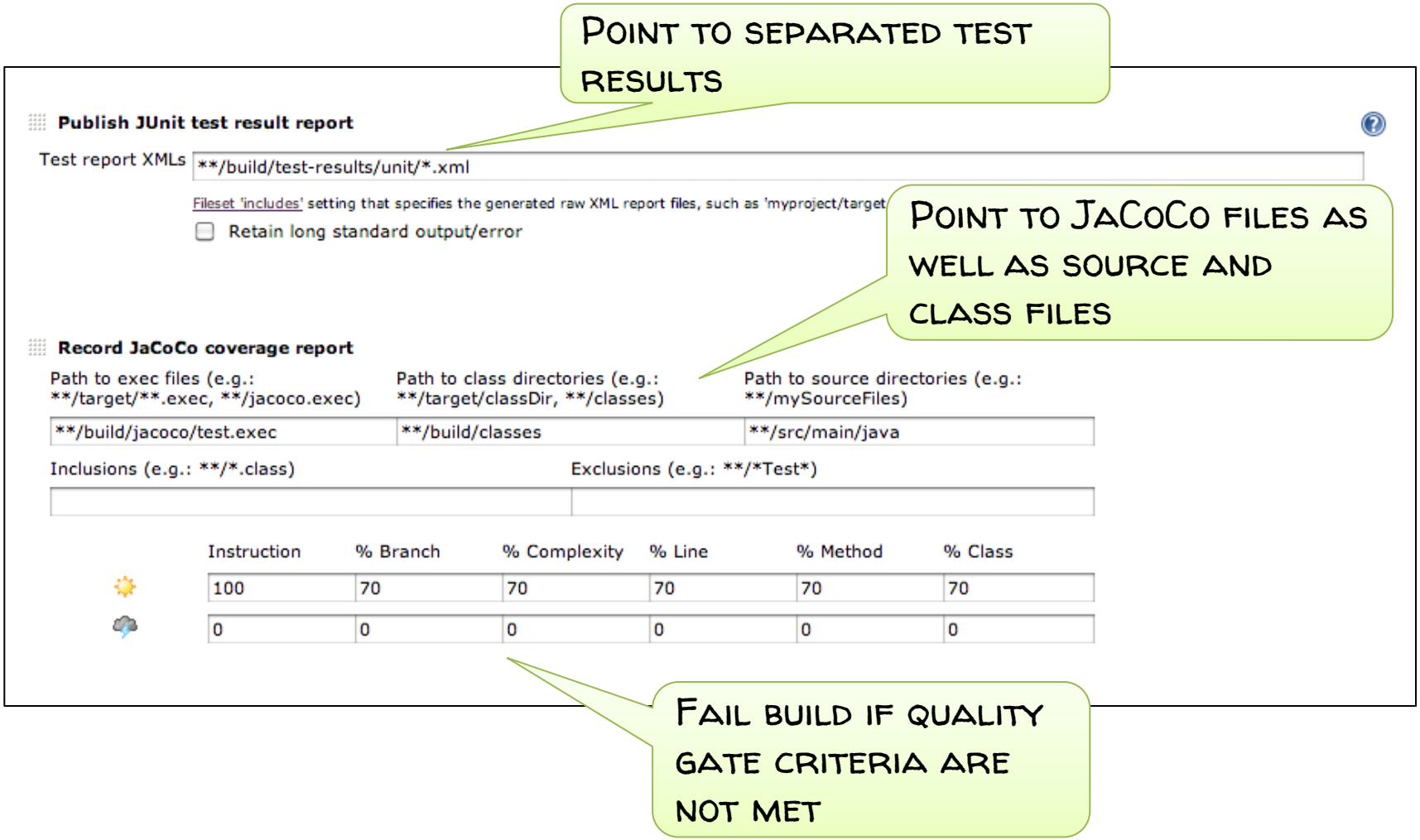
Parameterized Trigger Plugin



Clone Workspace SCM Plugin

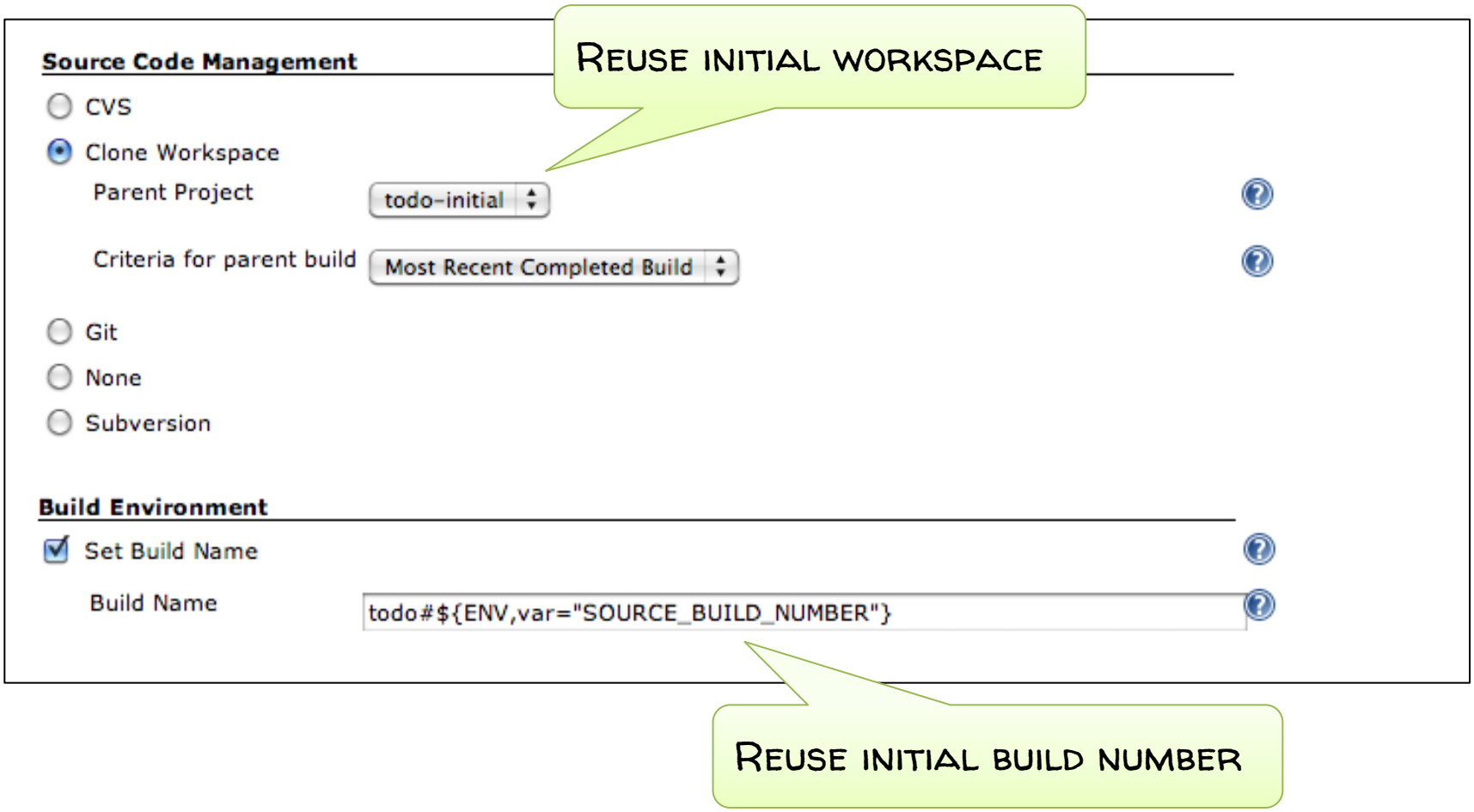


JaCoCo Plugin

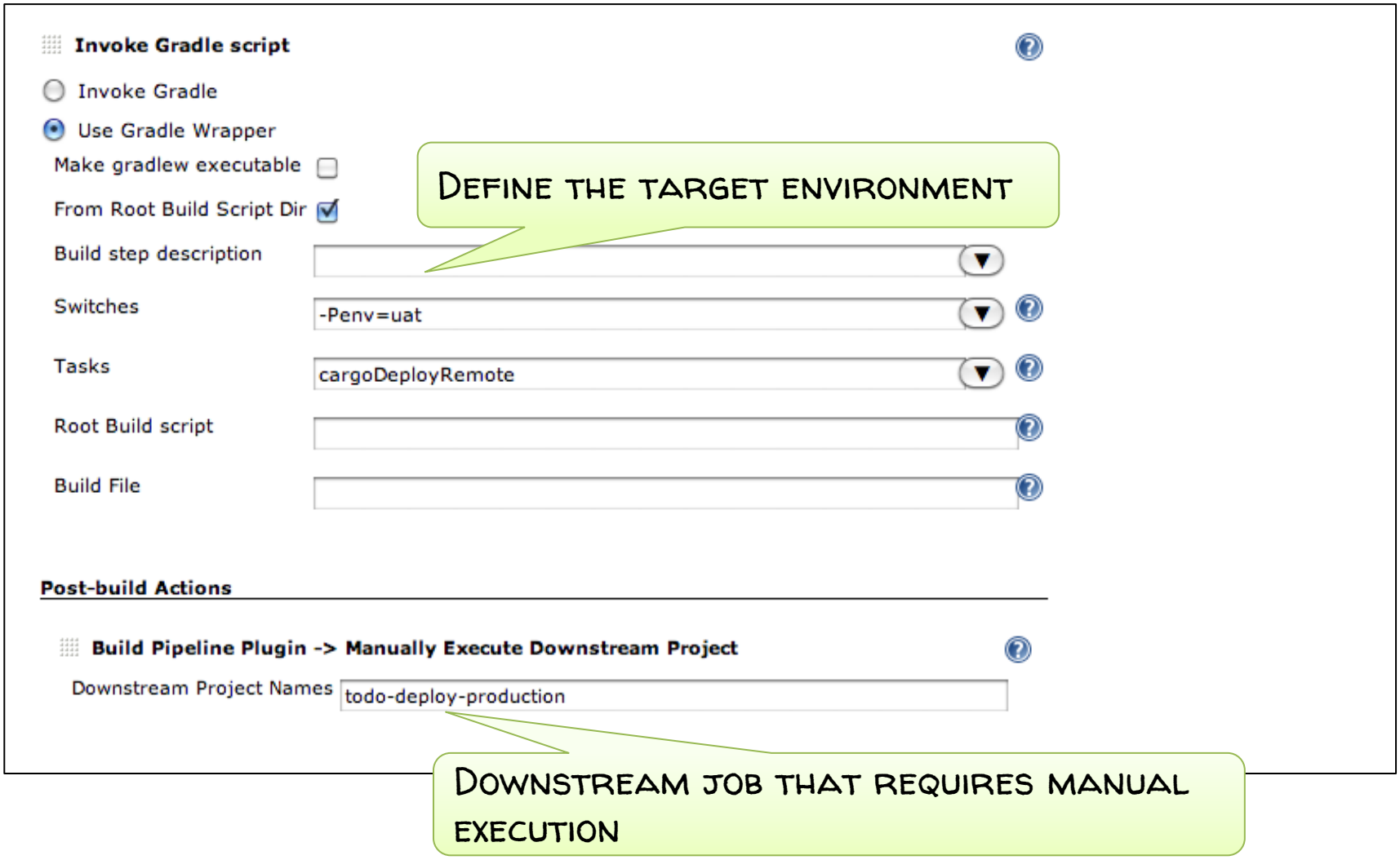


Clone Workspace SCM Plugin

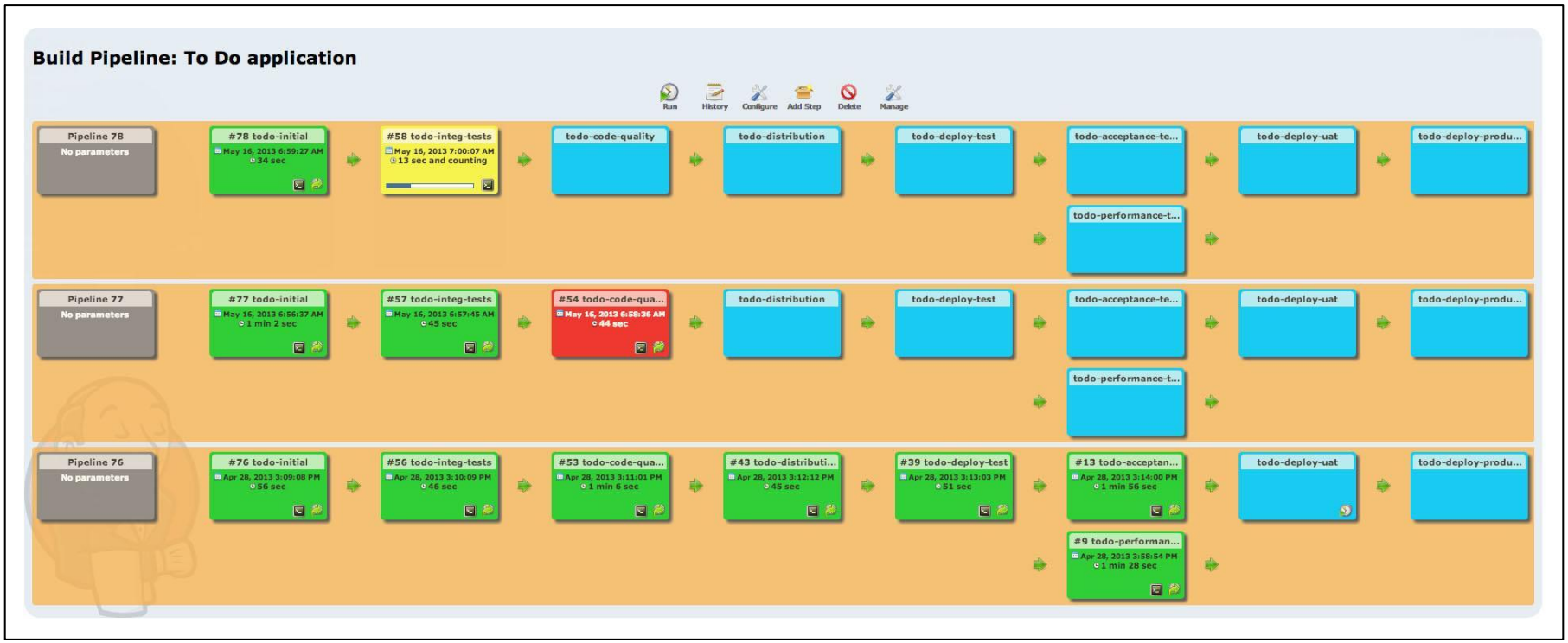
Build Name Setter Plugin



Build Pipeline Plugin



Build Pipeline Plugin



Visualization of chained pipeline jobs

```
> gradle qa
:askQuestions

BUILD SUCCESSFUL

Total time: 300 secs
```