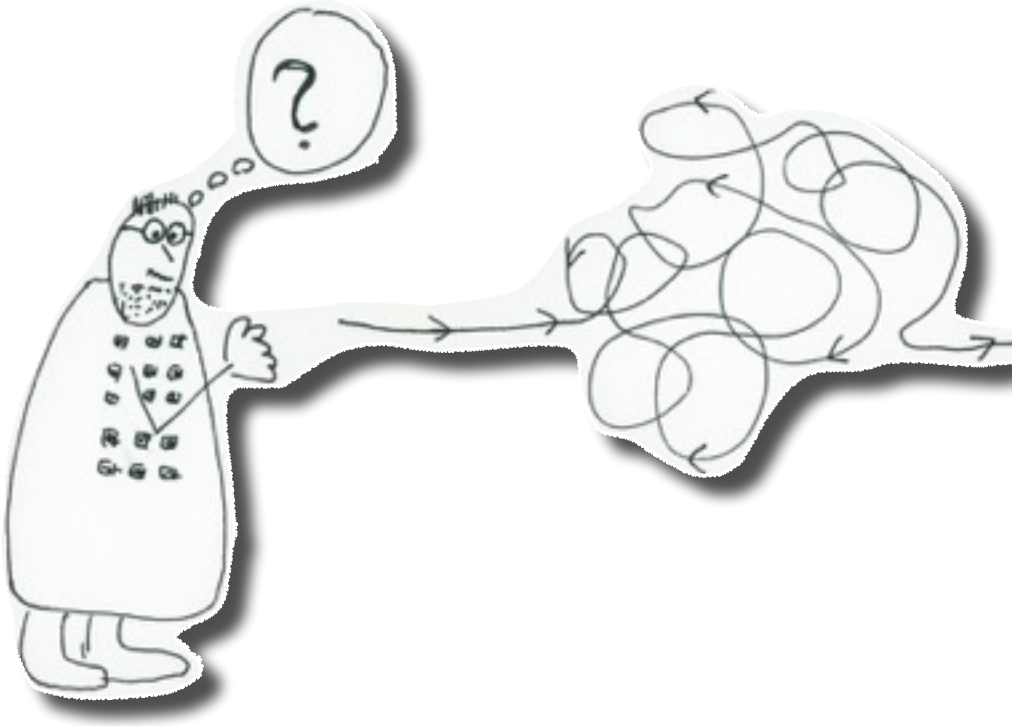
Migrating from Grails 2 to Grails 3

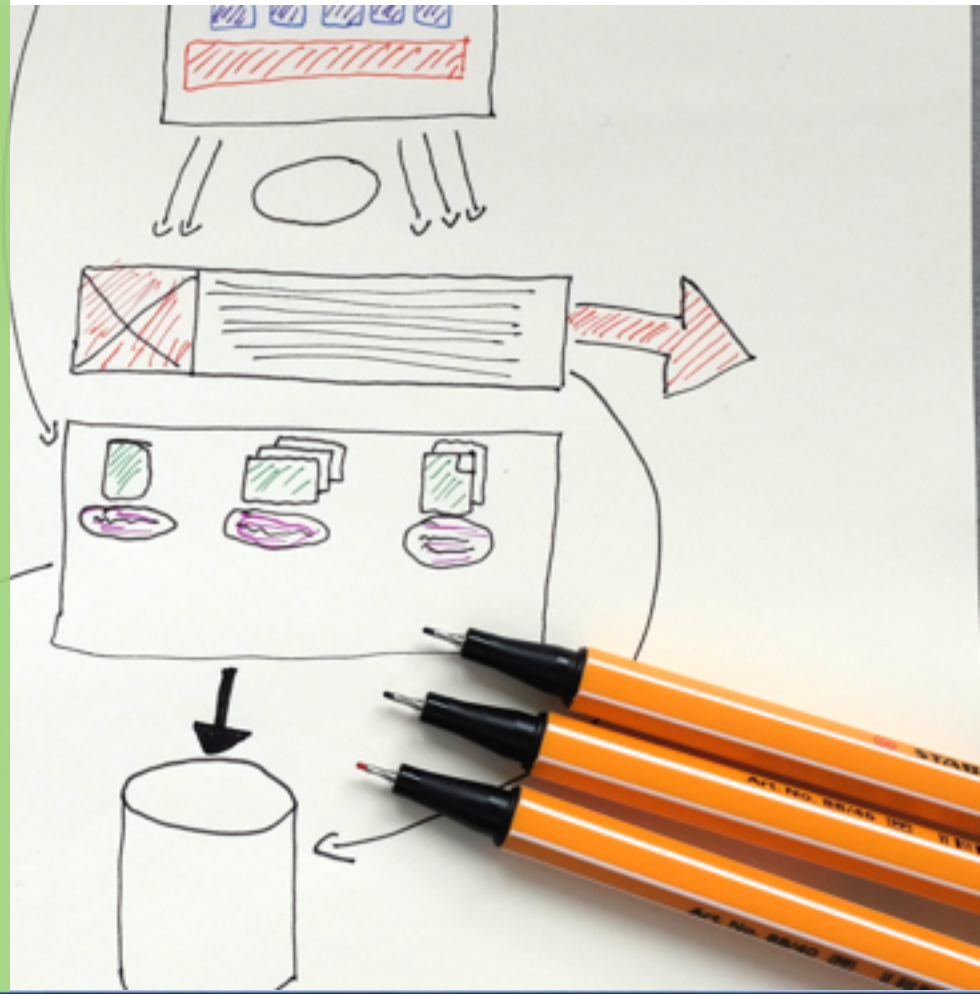
Michael Plöd - innoQ

Grails 3 is the most major and radical change in the history of Grails

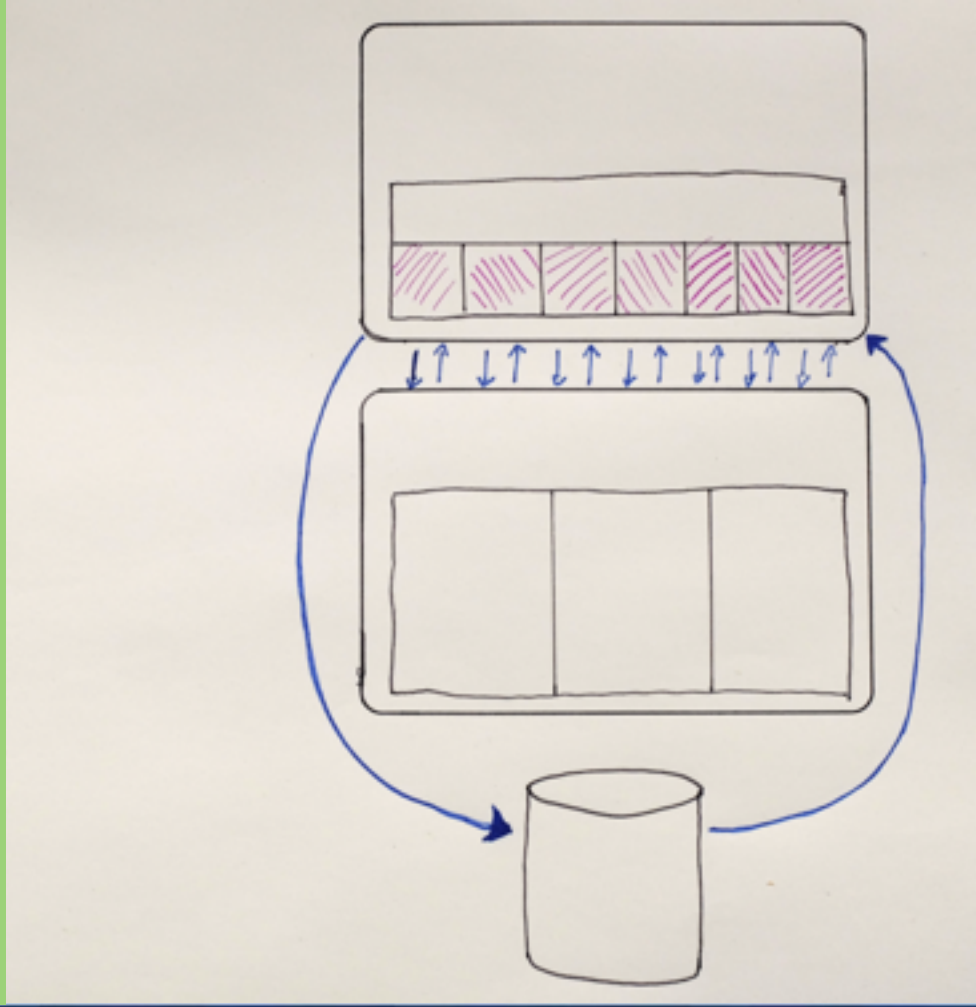
HOW TO MIGRATE



A Grails application usually consists of the application and various plugins



An application usually integrates with many standard and some self written plugins



Don’t add complexity. Wait until your major external plugins have been migrated

Grails 3 has different file / directory locations

Grails 2	Grails 3
grails-app/conf/BuildConfig.groovy	*build.gradle*
grails-app/conf/Config.groovy	*grails-app/conf/application.groovy*
grails-app/conf/UrlMappings.groovy	*grails-app/controllers/UrlMappings.groovy*
grails-app/conf/BootStrap.groovy	*grails-app/init/BootStrap.groovy*
scripts	*src/main/scripts*
src/groovy	*src/main/groovy*
src/java	*src/main/groovy*
test/unit	*src/test/groovy*
test/integration	*src/integration-test/groovy*
web-app	*src/main/webapp or src/main/resources/*
**GrailsPlugin.groovy*	*src/main/groovy*

Files that are not present in Grails 2.x

New File	Explanation
build.gradle	*Gradle build script*
gradle.properties	*Gradle build properties*
grails-app/conf/logback.groovy	*Logging has been extracted from Config.groovy and is now being defined via Logback*
grails-app/conf/application.yml	*You can now also configure your application with YAML*
grails-app/init/PACKAGE_PATH/Application.groovy	*The Application class that is used by Spring Boot to start the Grails 3 application*

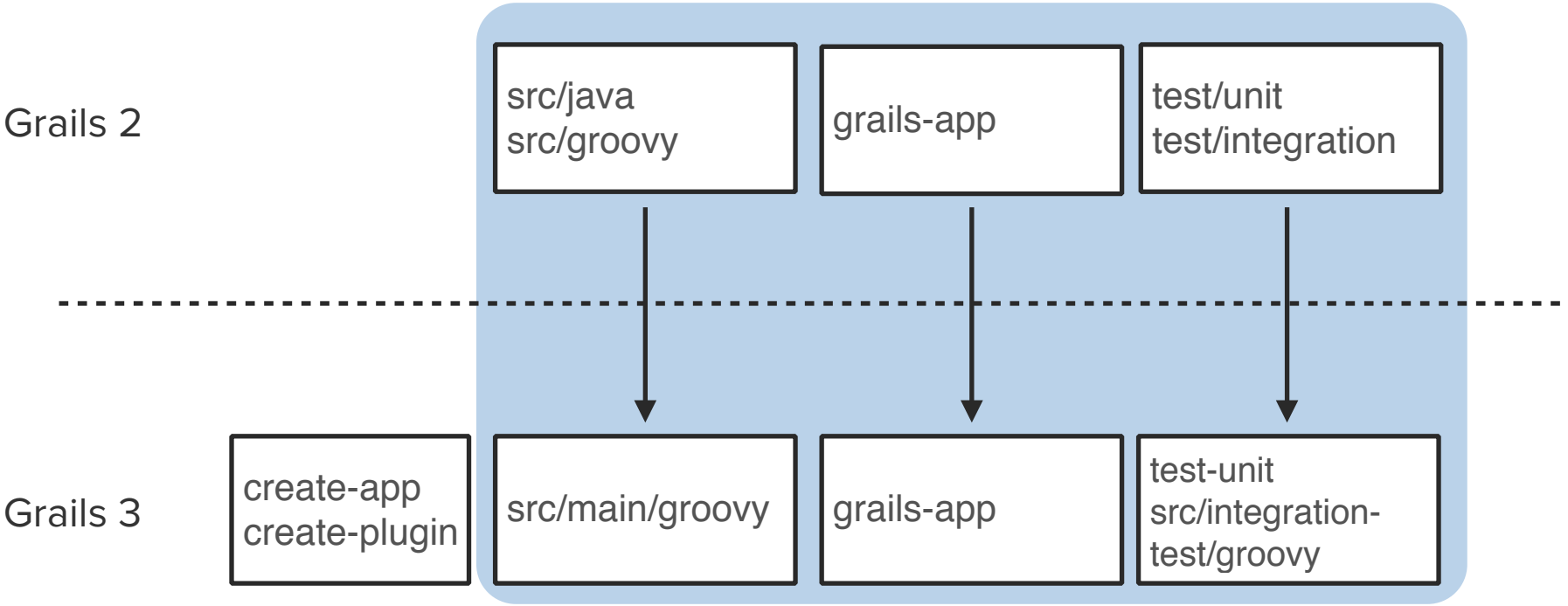
Unneeded files after a successful migration

Obsolete File	Explanation
application.properties	*Moved to build.gradle (for application name and version)*
grails-app/conf/DataSource.groovy	*Merged to application.yml*
lib	*Dependency resolution should be used to resolve JAR files*
web-app/WEB-INF/applicationContext.xml	*Removed, beans should now be defined in grails-app/conf/spring/resources.groovy*
src/templates/war/web.xml	*No longer needed for Grails 3. Do customization via Spring*
web-app/WEB-INF/sitemesh.xml	*Removed, sitemesh filter is no longer present*
web-app/WEB-INF/tld	*Removed, can be restored in src/main/webapp or src/main/resources/WEB-INF*

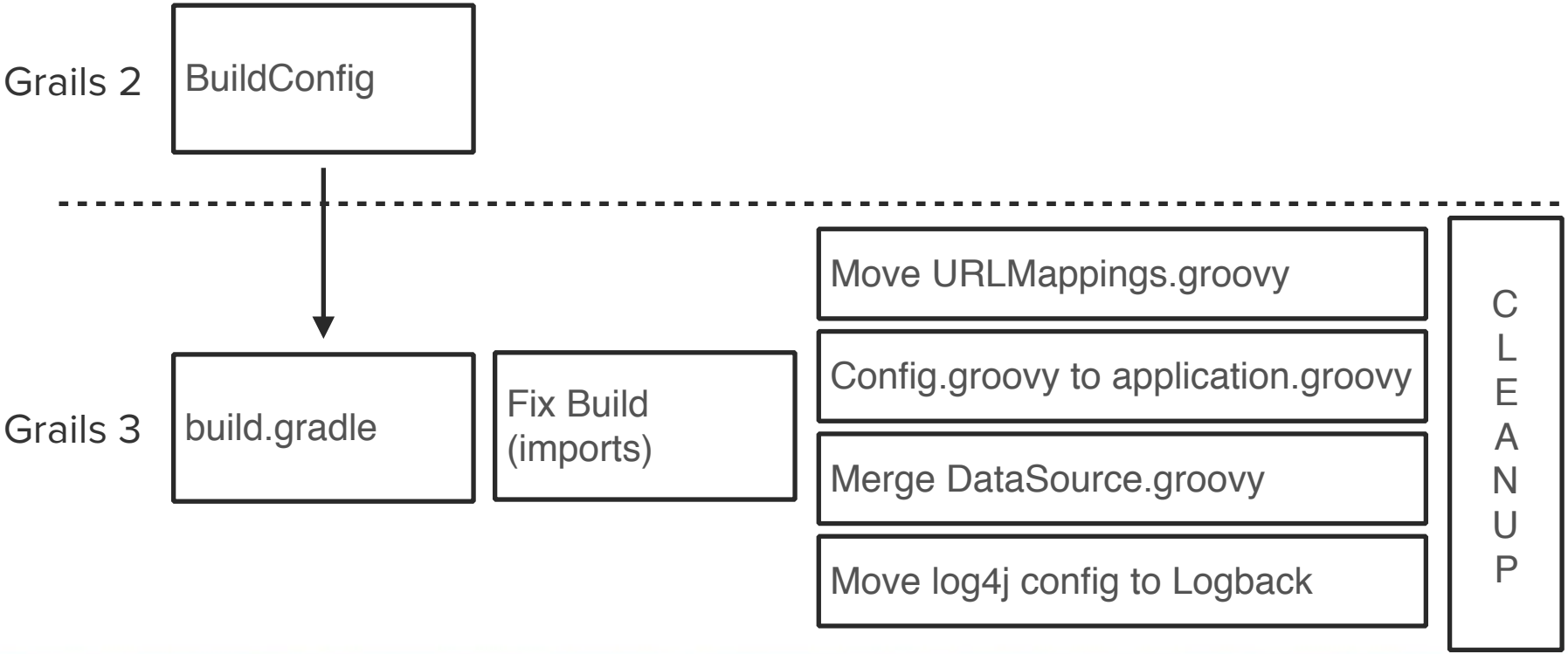
Before and after interceptors have been removed

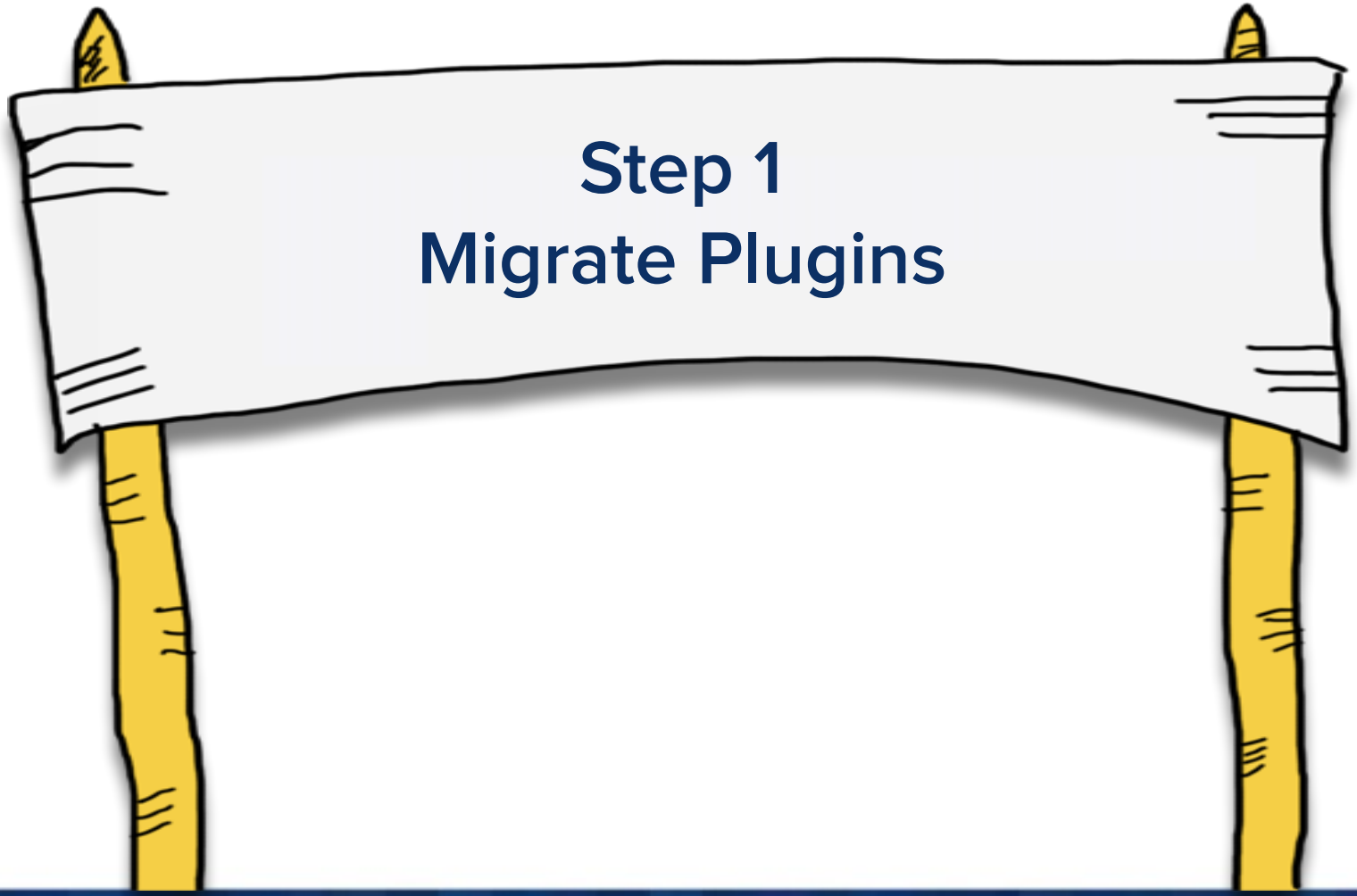
They need to be replaced by standalone interceptors

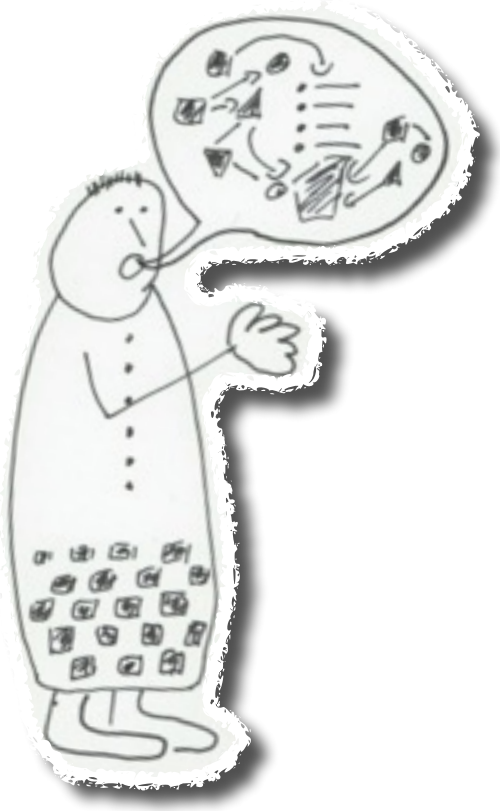
General migration steps 1/2



General migration steps 2/2




Step 1
Migrate Plugins



Mind the Plugin directory on the Grails Website.

It focusses on Grails 1.x - 2.x plugins

Grails 3 Plugins are on Bintray

Steps to take for plugin migration

There are several steps you have to take for migrating a plugin

- Create a new Grails 3 plugin
- Copy Sources
- **Handle the plugin descriptor**
- Add dependencies to the build
- Modify package imports
- Migrate configuration
- **Register ArtefactHandler definitions**
- **Migrate code generation scripts**
- Delete unnecessary files

Handle Plugin Descriptor

You must copy the Plugin descriptor from the Grails2 application to src/main/groovy/grails/plugins/[pluginname]

After that you have to add the correct package declaration to the plugin descriptor

```
package grails.plugins.recaptcha

class RecaptchaGrailsPlugin {
    …
}
```

Register Artefact Handler Definitions

If you have ArtefactHandler definitions **written in Java** you have to declare them in src/main/resources/META-INF/grails.factories

This step can be ignored for Groovy based ArtefactHandlers, Grails detects them.

```
grails.core.ArtefactHandler=grails.plugins.quartz.JobArtefactHandler
```

Migrate code generation scripts

Old Gant code generation scripts have to be replaced by Code Generation Scripts or Gradle tasks.

Simple code generation can easily be migrated to the new code generation API

More complex tasks are better of with a migration to Gradle tasks

Simple code generation example

```
includeTargets << grailsScript("_GrailsCreateArtifacts")
target(createJob: "Creates a new Quartz scheduled job") {
    depends(checkVersion, parseArguments)
    def type = "Job"
    promptForName(type: type)

    for (name in argsMap.params) {
        name = purgeRedundantArtifactSuffix(name, type)
        createArtifact(name: name, suffix: type, type: type, path: "grails-app/jobs")
        createUnitTest(name: name, suffix: type)
    }
}

setDefaultTarget 'createJob'
```

Gradle Tasks for complex generation scripts

> grails create-command run-query

```
import …
class RunQueryCommand implements ApplicationCommand {
  @Autowired
  DataSource dataSource

  boolean handle(ExecutionContext ctx) {
      def sql = new Sql(dataSource)
      println sql.executeQuery("select * from foo")
      return true
  }
}
```

Command can be added to classpath in build.gradle

```
buildscript {
  …
  dependencies {
    classpath "org.grails.plugins:myplugin:0.1-SNAPSHOT"
  }
}
…
dependencies {
  runtime "org.grails.plugins:myplugin:0.1-SNAPSHOT"
}
```

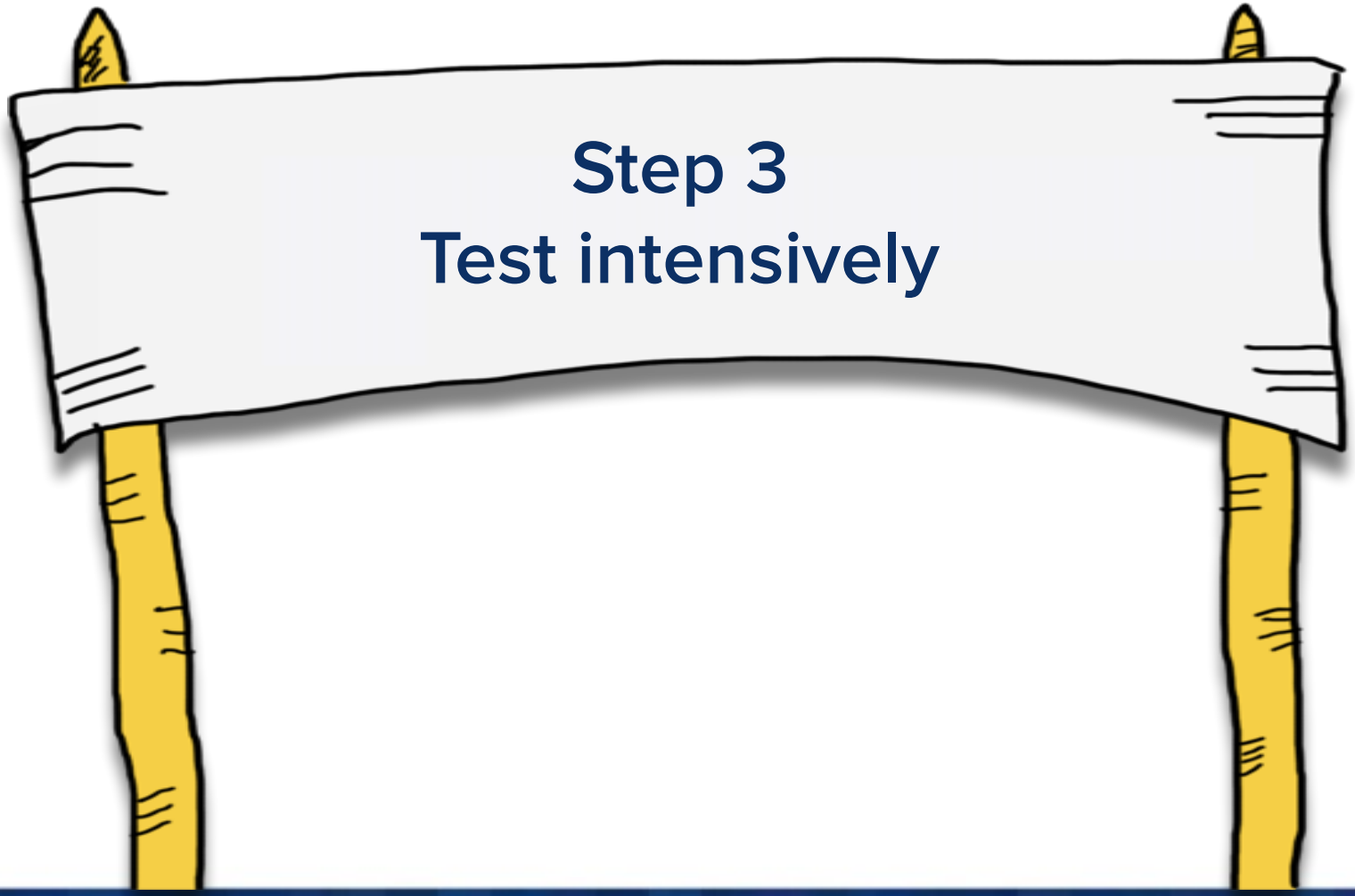
> grails run-query


Step 2
Migrate Application

Steps to take for application migration

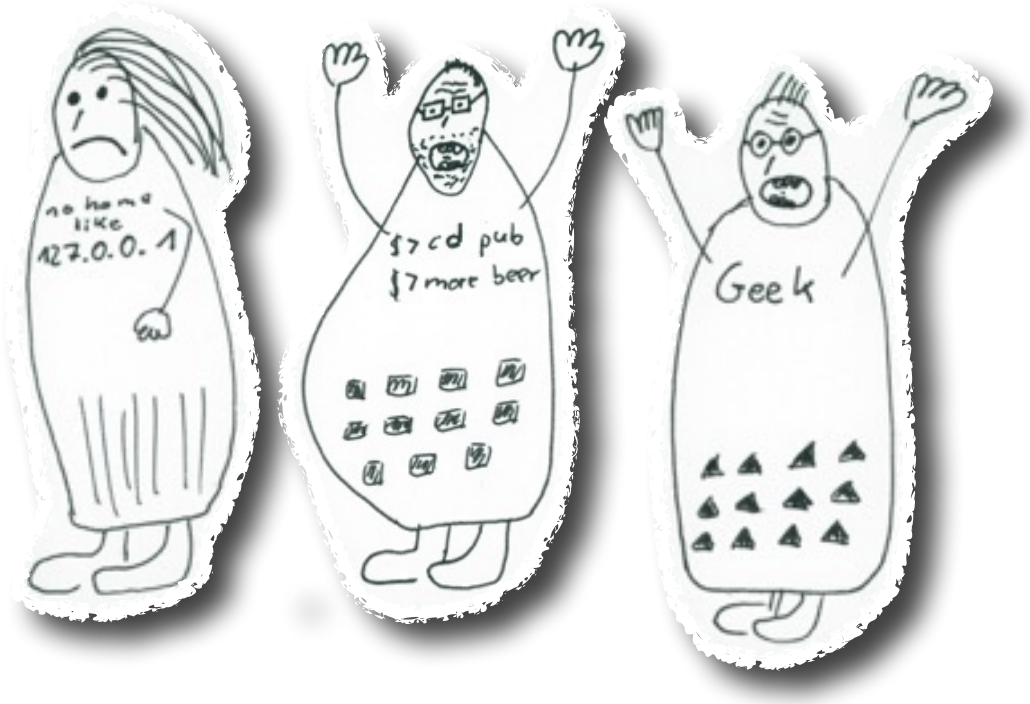
There are several steps you have to take for migrating a application

- Create a new Grails 3 application
- Copy Sources
- Add dependencies to the build
- Migrate Configuration
- Migrate web.xml Modifications to Spring
- Migrate static assets not handled by Asset pipeline
- Migrate Tests
- Delete unnecessary files


Step 3
Test intensively

Let’s migrate an application

ALLSCHOOLS



Recap - Migration Steps

THANK YOU!

Michael Plöd - innoQ
Follow me on Twitter: @bitboss