
RELEASE MANAGEMENT
WITH GRADLE
alfalab

@tolkv

@lavcraft

alfalab

octoberry

РАЗБОР ПОЛЁТОВ
#RAZBORPOLETOV
SINCE 2011
EST. 5772
ПОДКАСТ
ДОБРЫЙ ДЕНЬ ДОБРЫЙ ВЕЧЕР
ДОБРОЕ УТРО ДОБРОЙ НОЧИ

Чего не будет

Чего не будет

- Релизь чаще — будет счастье!

Чего не будет

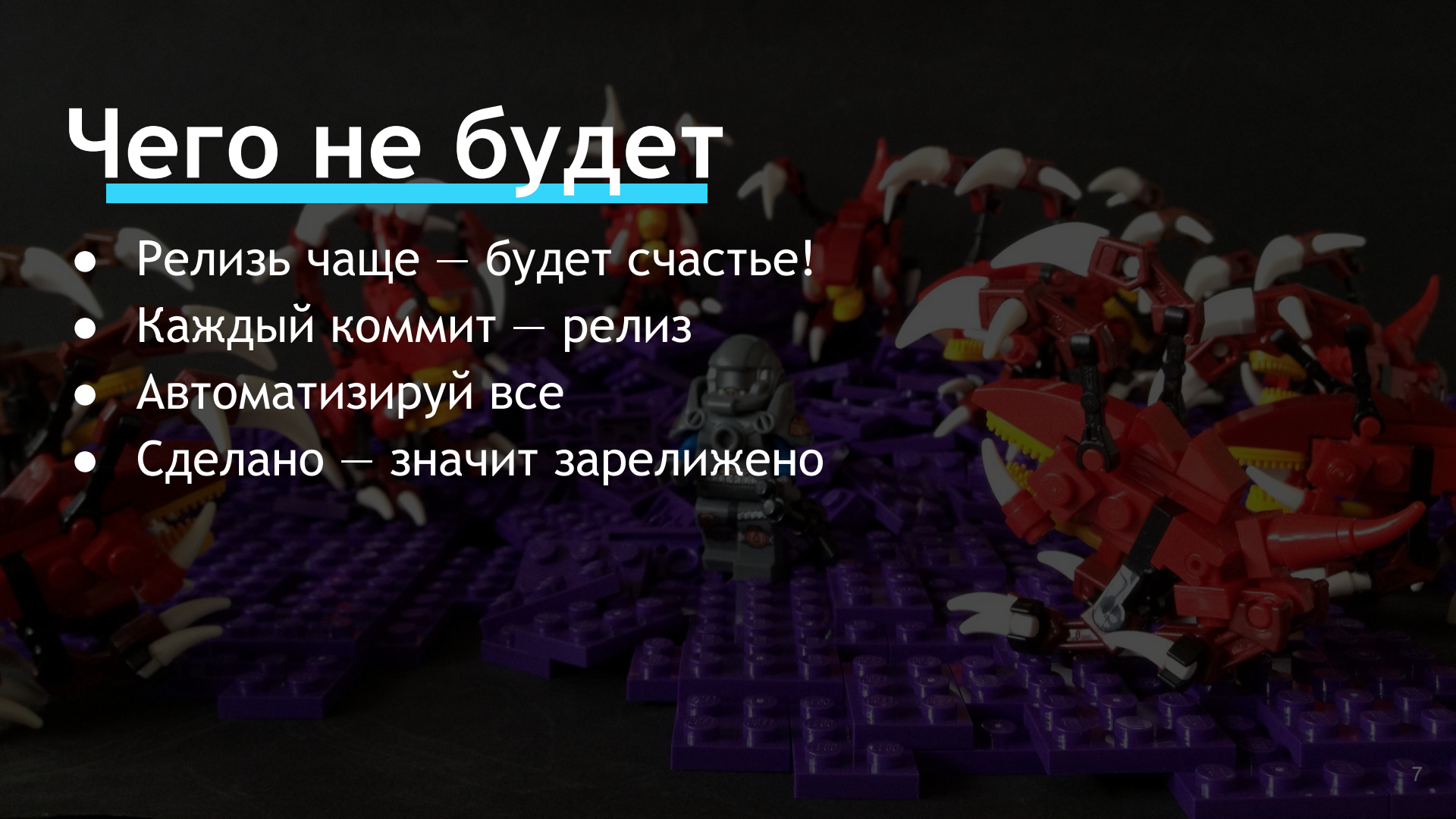
- Релизь чаще — будет счастье!
- Каждый коммит — релиз

Чего не будет

- Релизь чаще — будет счастье!
- Каждый коммит — релиз
- Автоматизируй все

Чего не будет

- Релизь чаще — будет счастье!
- Каждый коммит — релиз
- Автоматизируй все
- Сделано — значит зарелижено



Что будет

Что будет

- Жизненный цикл скриптов сборки

Что будет

- Жизненный цикл скриптов сборки
- Основные проблемы

Что будет

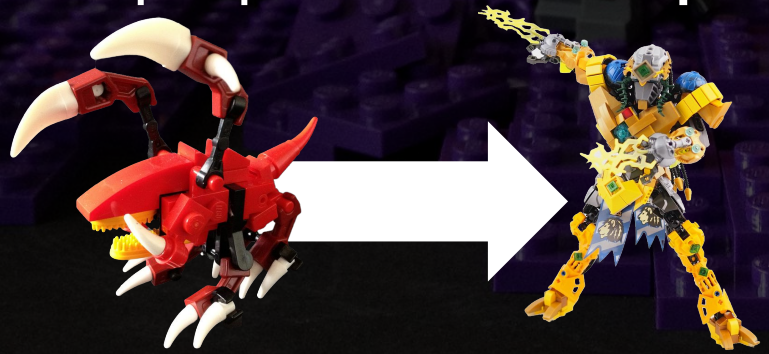
- Жизненный цикл скриптов сборки
- Основные проблемы
- Инструменты для решения

Что будет

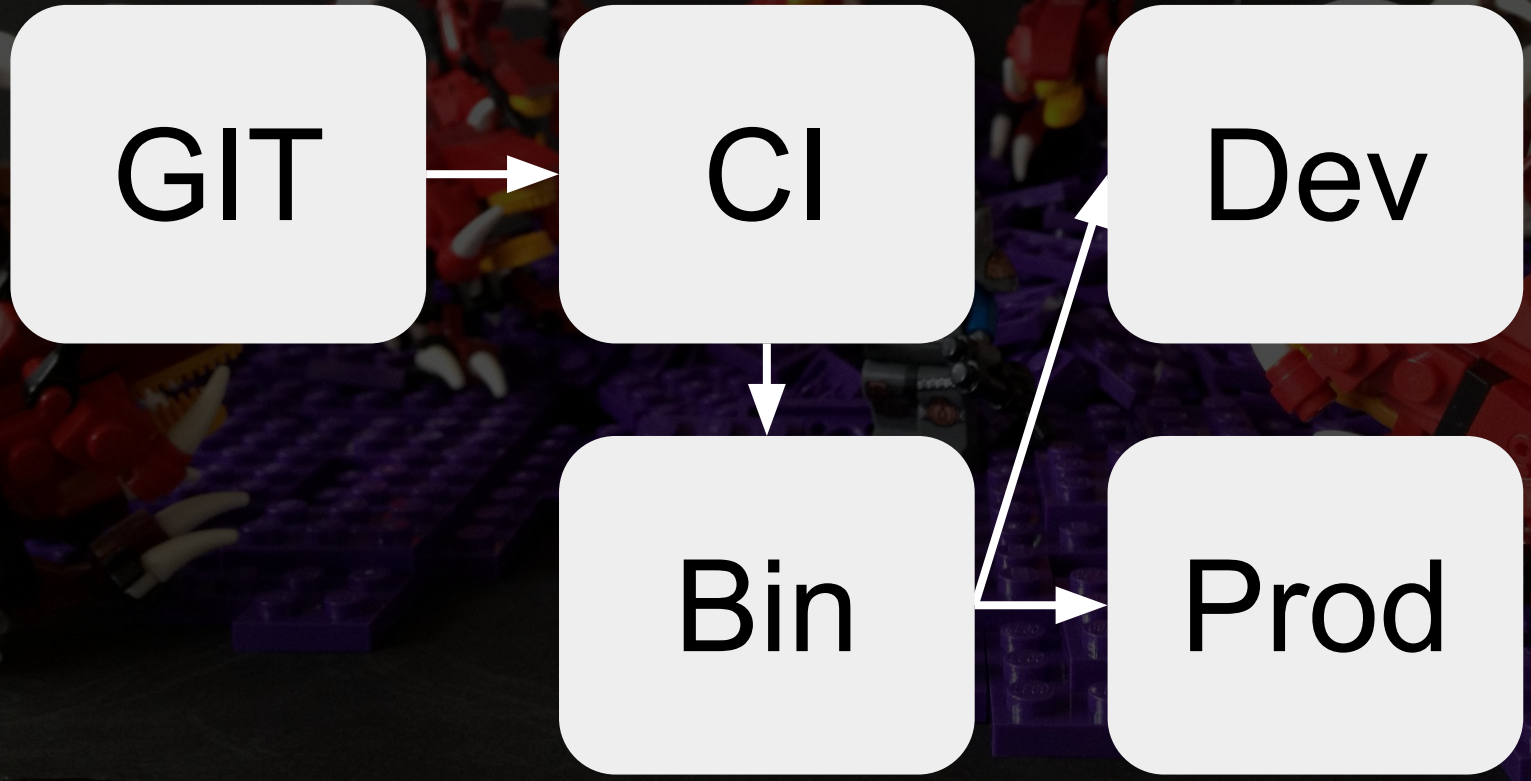
- Жизненный цикл скриптов сборки
- Основные проблемы
- Инструменты для решения
- Императивный vs декларативный Gradle

Что будет

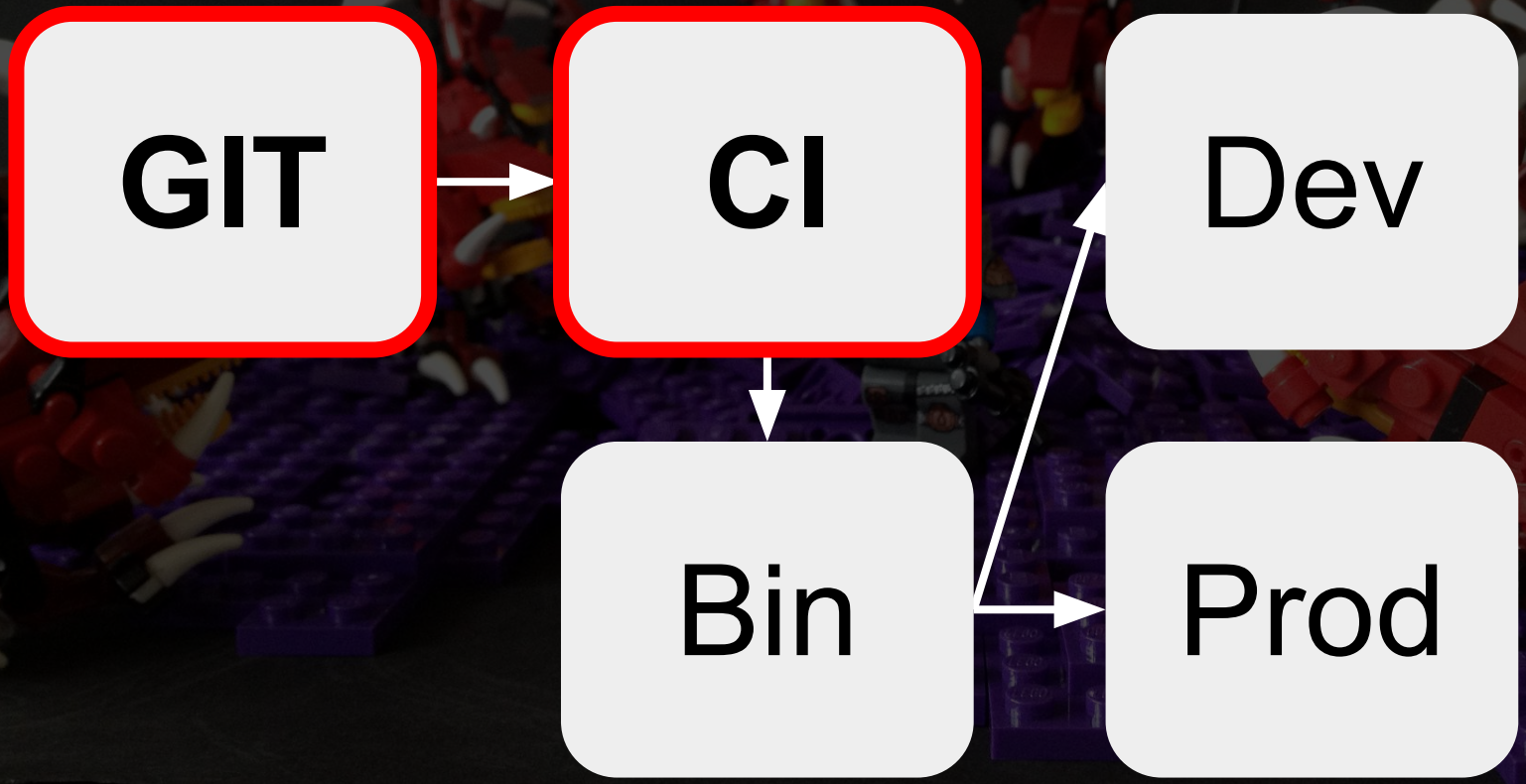
- Жизненный цикл скриптов сборки
- Основные проблемы
- Инструменты для решения
- Императивный vs декларативный Gradle
- Можно ли превратиться из **Зерга** в **Протосса**



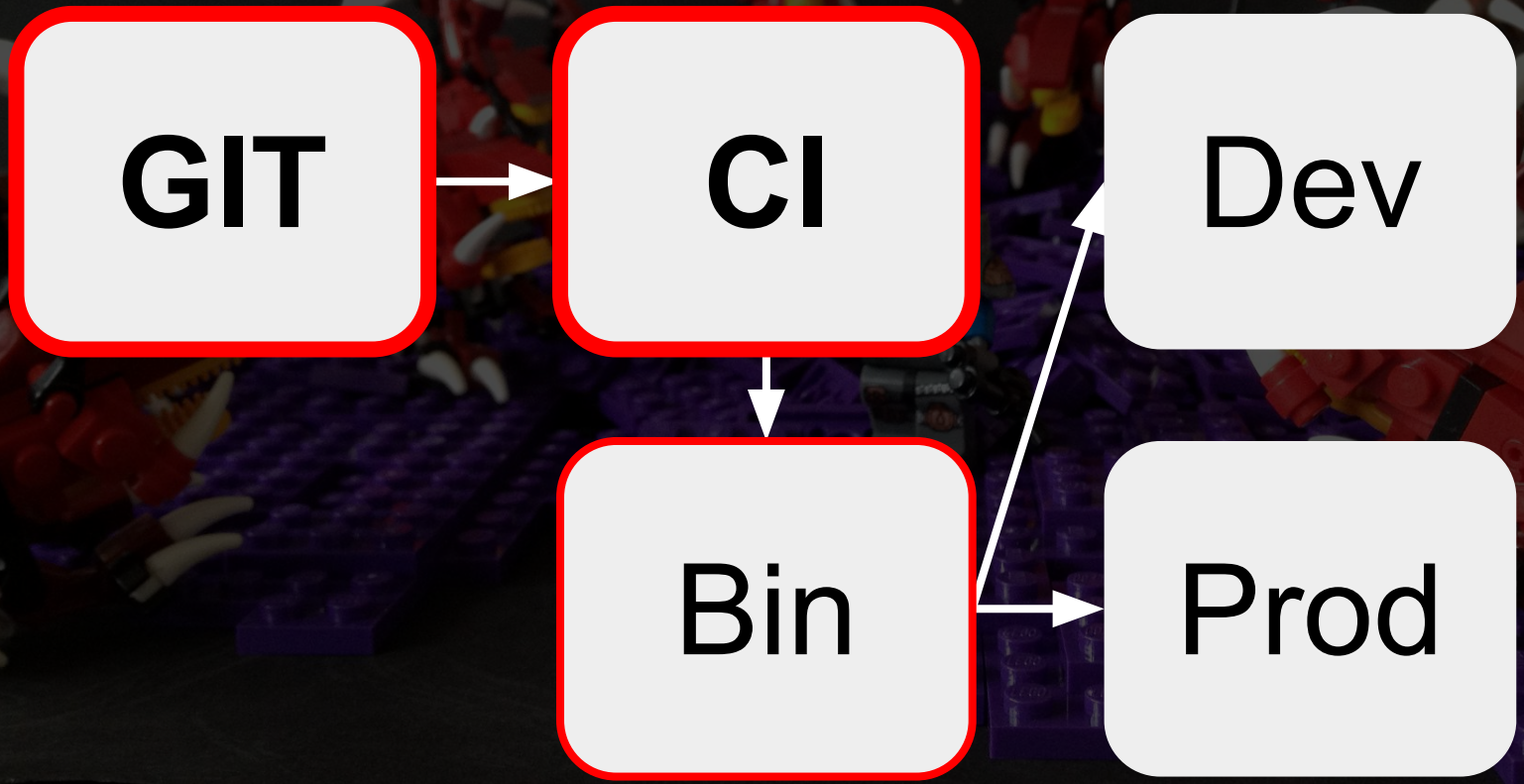
Где мы



Где мы



Где мы



Плохой код

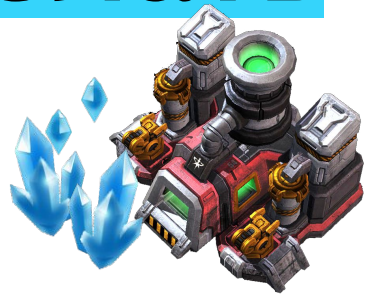
Плохой билдскрипт

Стратегия успеха

Что нужно сделать

Что нужно сделать

- Добыть ресурсов



Что нужно сделать

- Добыть ресурсов
- Подготовить базу

Что нужно сделать

- Добыть ресурсов
- Подготовить базу
- Создать армию

Что нужно сделать

- Добыть ресурсов
- Подготовить базу
- Создать армию
- Укрепить позиции

Что нужно сделать

- Добыть ресурсов
- Подготовить базу
- Создать армию
- Укрепить позиции
- Захватить мир

Все еще собираете?

- Кто писал “скрипты сборки”?

Все еще собираете?

- Кто писал “скрипты сборки” с нуля?

Все еще собираете?

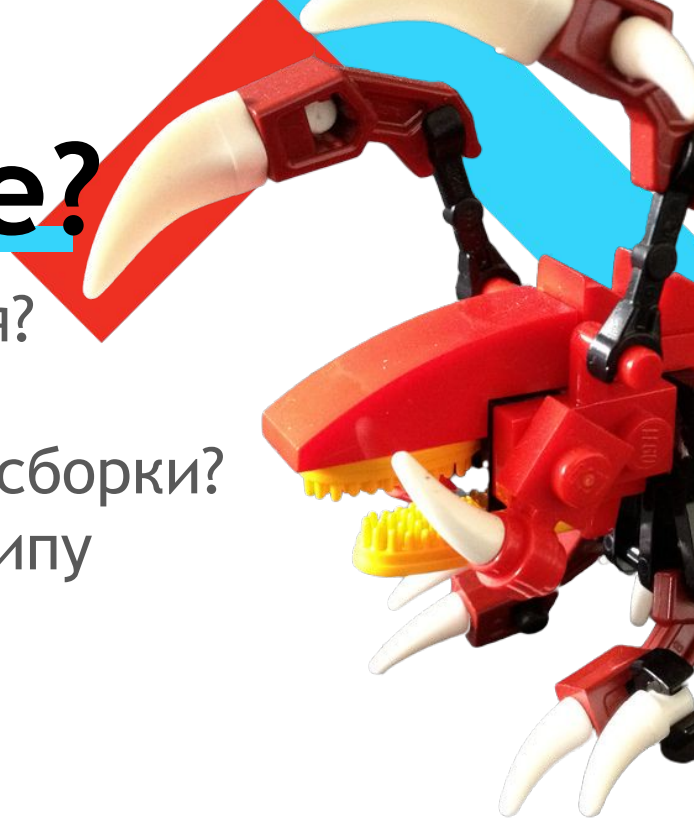
- Кто писал “скрипты сборки” с нуля?
- А кто правил существующие?

Все еще собираете?

- Кто писал “скрипты сборки” с нуля?
- А кто правил существующие?
- По какому принципу пишут логику сборки?

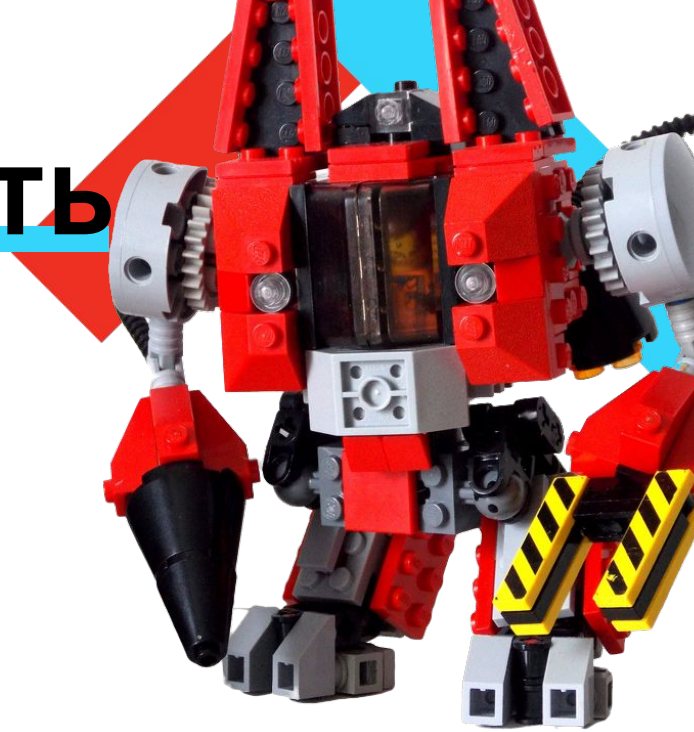
Все еще собираете?

- Кто писал “скрипты сборки” с нуля?
- А кто правил существующие?
- По какому принципу пишут логику сборки?
- Правильно - по остаточному принципу

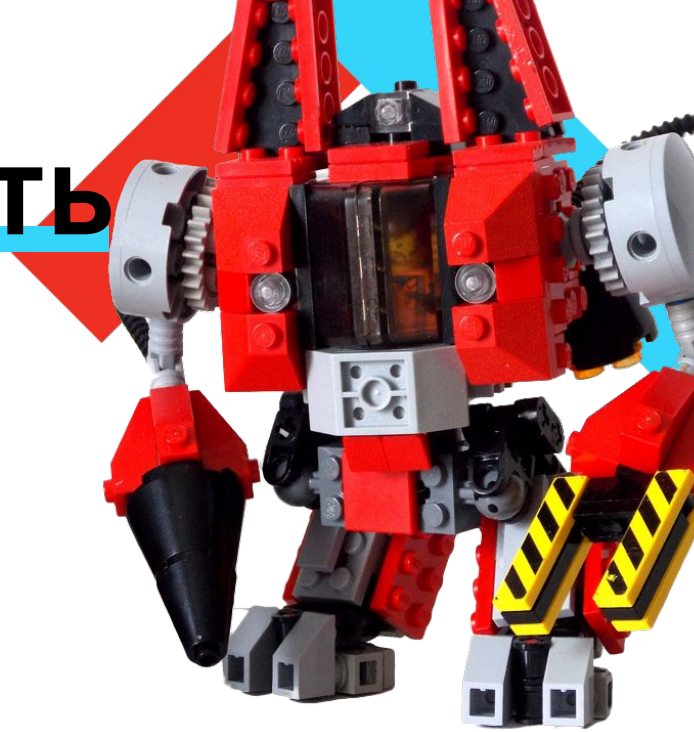


Готовимся собирать

- build.gradle

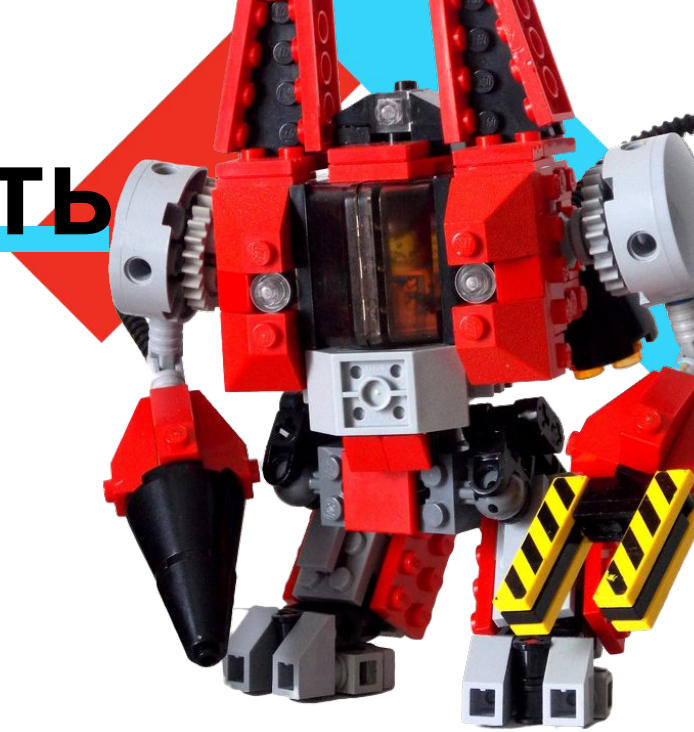


Готовимся собирать



- build.gradle
- settings.gradle

Готовимся собирать



- build.gradle
- settings.gradle
- ./gradlew

Добываем инструменты

```
$ gradle wrapper --gradle-version 3.1
```

Добываем инструменты

```
$ gradle wrapper --gradle-version 3.1

:wrapper
BUILD SUCCESSFUL

Total time: 1.835 secs
```

Добываем инструменты

```
$ ./gradlew init
```

Добываем инструменты

```
$ ./gradlew init
:wrapper
:init

BUILD SUCCESSFUL

Total time: 0.772 secs
```

Добыли

```
$ ls
build.gradle
gradle
gradlew
gradlew.bat
settings.gradle
```

Добыли

```
$ ls
build.gradle
gradle/
gradlew
gradlew.bat
settings.gradle
```

Добыли

```
$ ls
build.gradle
gradle/
gradlew
gradlew.bat
settings.gradle
```

Добыли

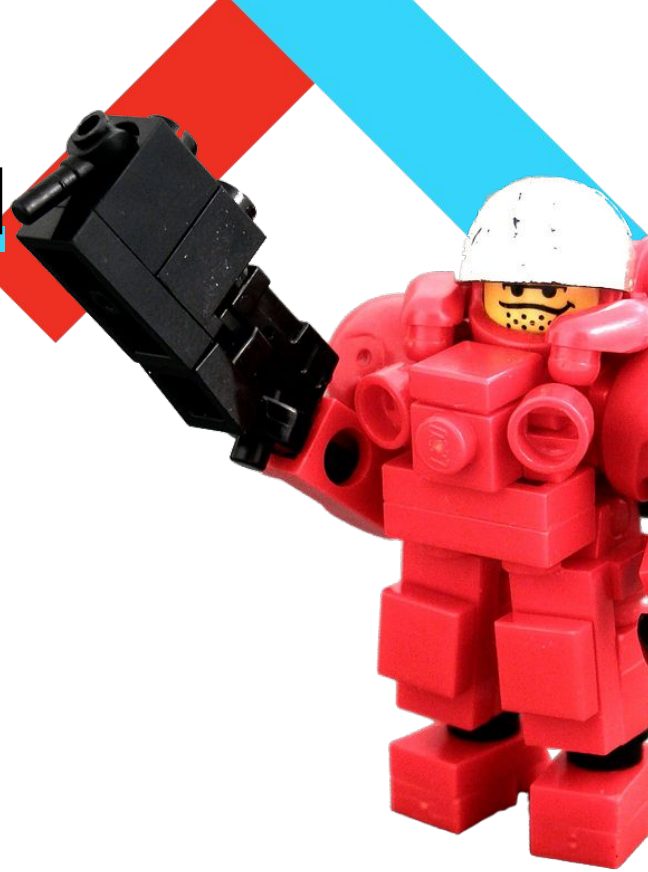
```
$ ls
build.gradle
gradle/
gradlew
gradlew.bat
settings.gradle
```

Добыли

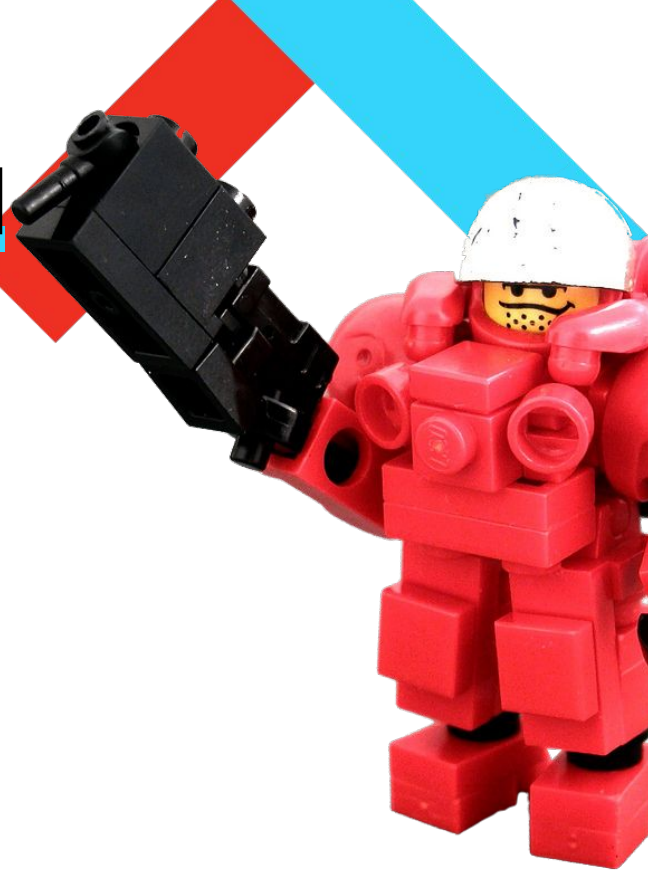
```
$ ls
build.gradle
gradle/
gradlew
gradlew.bat
settings.gradle
```

Добываем ресурсы

- Соберем
  `./gradlew build`

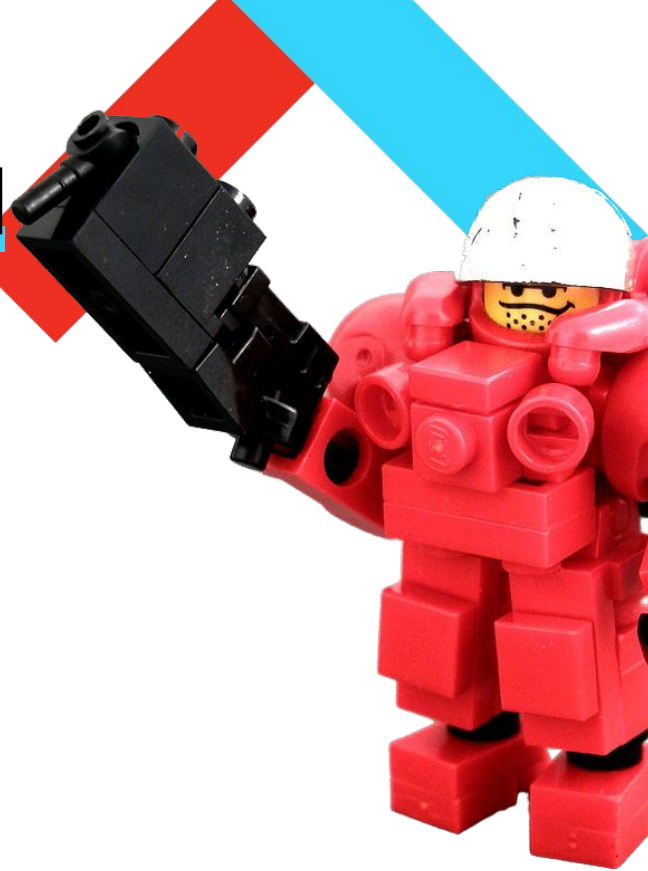


Добываем ресурсы



- Соберем
  `./gradlew build`
- Получили
  `ls ./build/libs/`
  `project_name.jar`

Добываем ресурсы



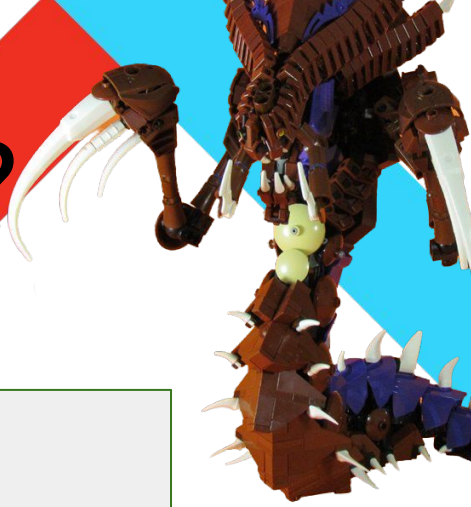
- Соберем
  `./gradlew build`
- Получили
  `ls ./build/libs/`
  `project_name.jar`

Где версия?

Где моя версия, чувак?

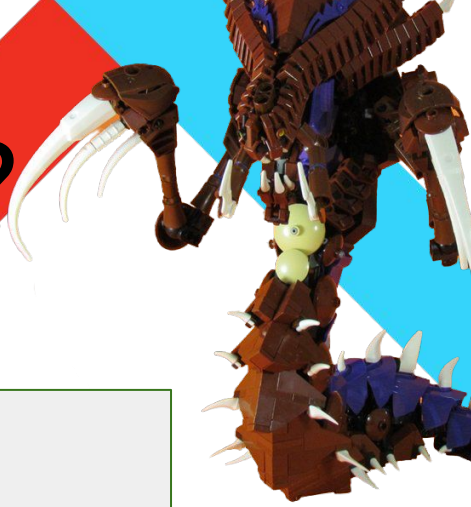
```
$ cat build.gradle | grep version
```

Где моя версия, чувак?



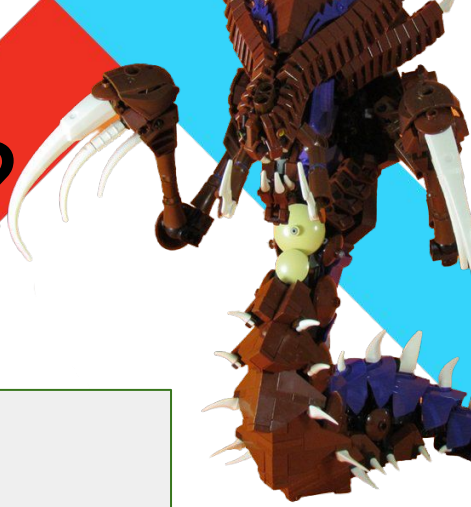
```
$ cat build.gradle | grep version

Ничего!
```

Где моя версия, чувак?



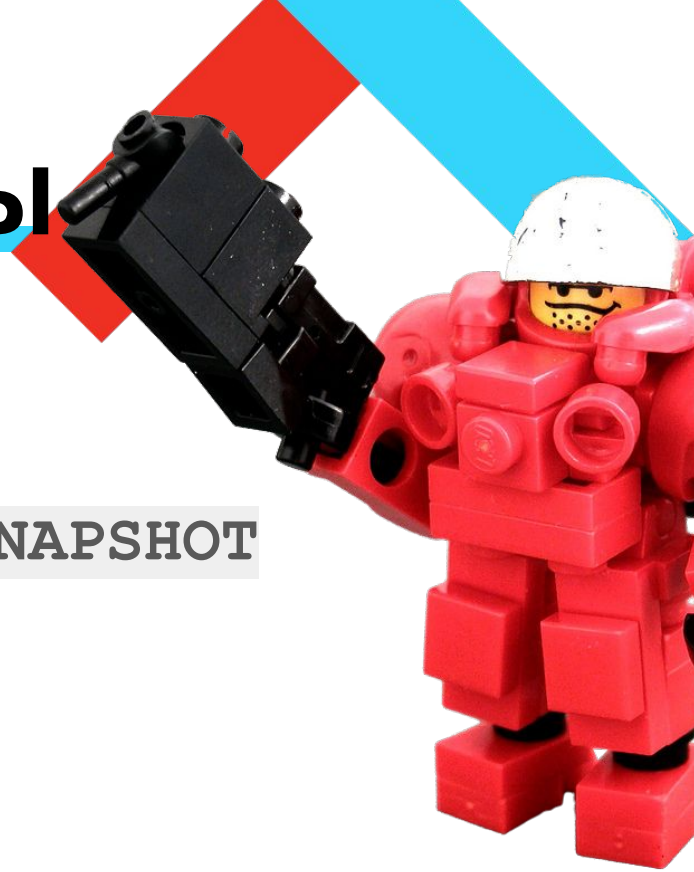
```
$ cat build.gradle | grep version

Ничего!

$ echo "version = '1.0.1'" >> build.gradle
```

Где моя версия, чувак?



```
$ cat build.gradle | grep version

Ничего!

$ echo "version = '1.0.1'" >> build.gradle

$ ./gradlew clean build
$ ls build/libs/

project_name-1.0.1.jar
```

Добываем ресурсы



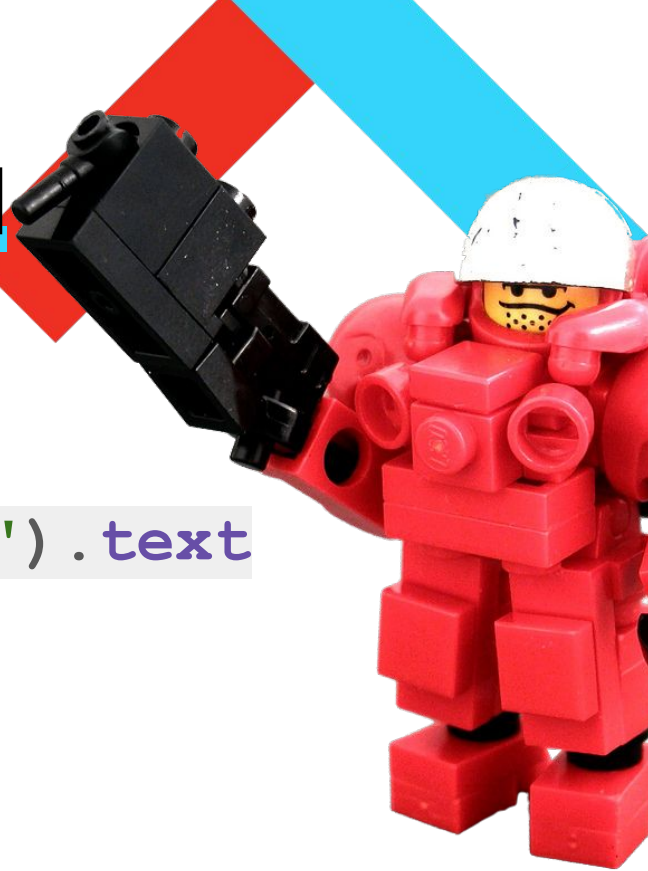
- Указывать вручную

```
./gradlew -Pversion=1.3.4-SNAPSHOT
./gradlew -Pversion=1.3.4-Ы
```

Добываем ресурсы

- Брать из файла

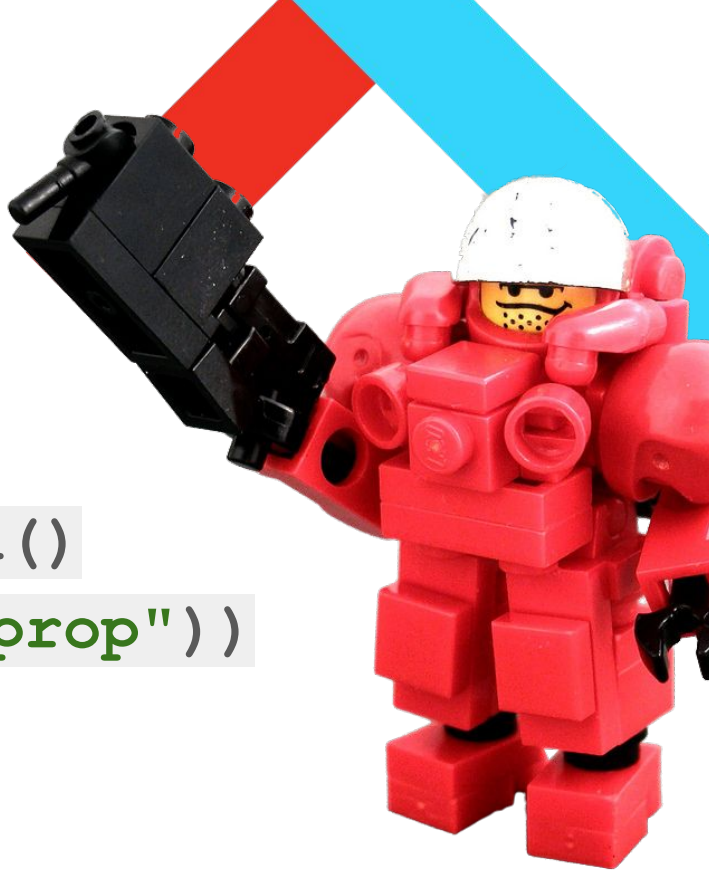
```
version = file("version.prop").text
```



Версия?

- Git/Metadata

```
version = new CustomVersion()
    .loadFrom(file("version.prop"))
    .appVersion()
```



Версия?

- Версия хранится в файле в git

Версия?

- Версия хранится в файле в git
- Синхронизировали версию - коммит

Версия?

- Версия хранится в файле в git
- Синхронизировали версию - коммит
- Осложняется поиск по нужной версии

Будь как зерг

- Zerg Rush?

Будь как зерг

- Zerg Rush?
- google.com: gradle problem_name solution

Будь как зерг

- Zerg Rush?
- google.com: gradle problem_name solution
- Copy/Paste into build.gradle

Будь как зерг

- Zerg Rush?
- google.com: gradle problem_name solution
- Copy/Paste into build.gradle

Будь как зерг

- GrGit

```
version = GrGit.open(file("."))
    .head()
    .abbreviatedId
```

Будь как зерг

- GrGit

```
version = GrGit.open(file("."))
    .head()
    .abbreviatedId
```

- Не закоммитил – нет новой версии

Что делать с Git?

```
1: task release {
2:   doFirst {
3:     grgit.tag.add {
4:       name = version
5:       message = "Release of ${version}"
6:     }
7:   }
8: }
```

Порядок запуска

1. Task definition

Порядок запуска

1. Task definition

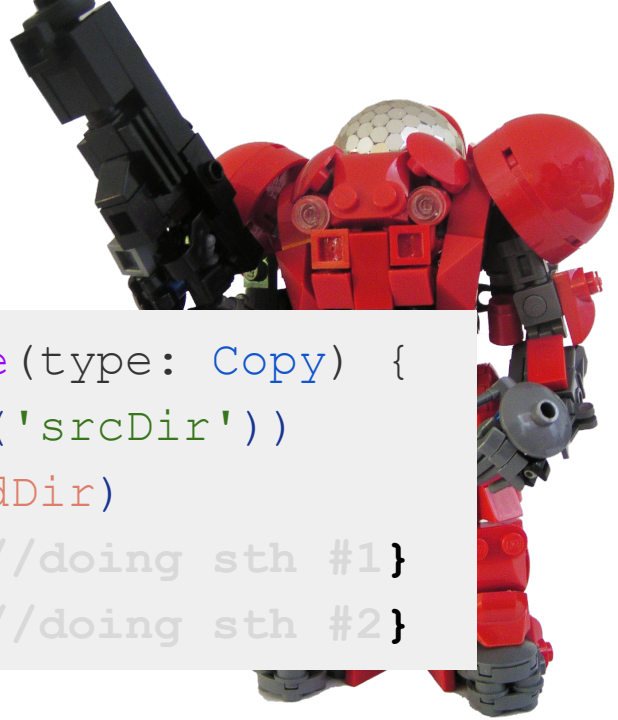
```
1: task release(type: Copy) {
2:   from(file('srcDir'))
3:   into(buildDir)
4: }
```

Порядок запуска

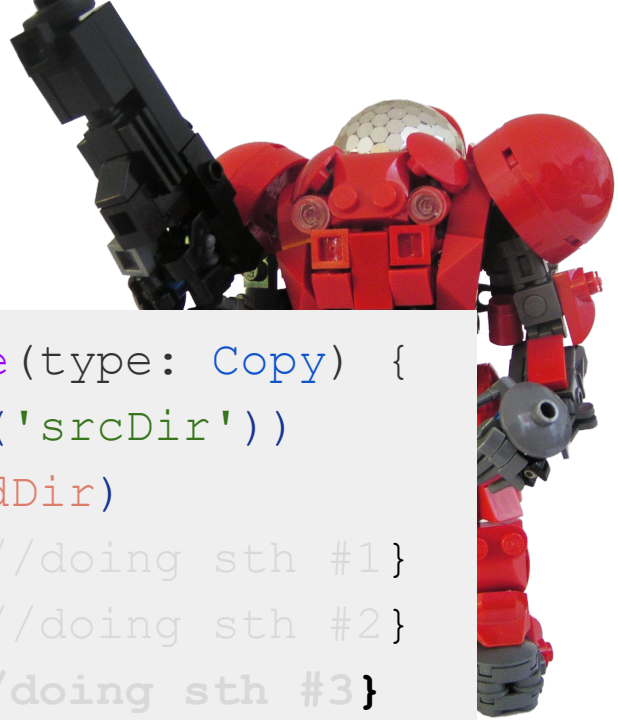
1. Task definition

2. Task doFirst

```
1: task release(type: Copy) {
2:   from(file('srcDir'))
3:   into(buildDir)
4:   doFirst {//doing sth #1}
5:   doFirst {//doing sth #2}
```



Порядок запуска



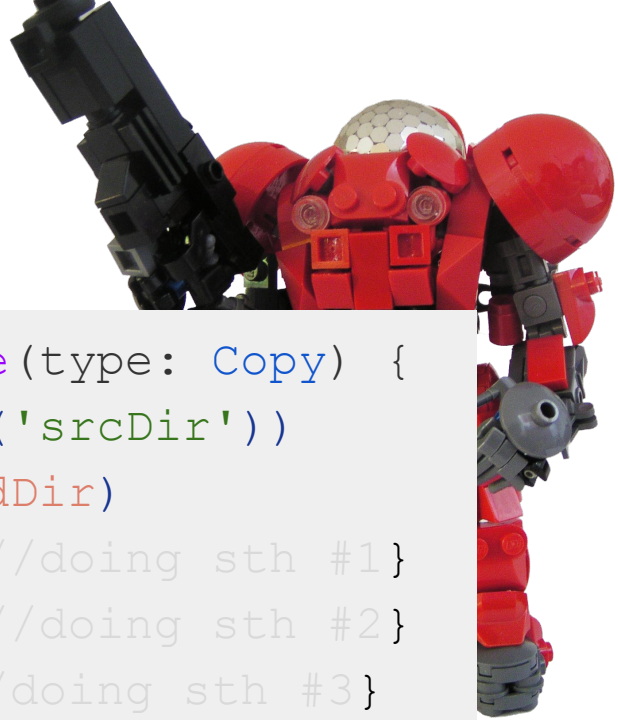
1. Task definition

2. Task doFirst

3. Task doLast

```
1: task release(type: Copy) {
2:     from(file('srcDir'))
3:     into(buildDir)
4:     doFirst {//doing sth #1}
5:     doFirst {//doing sth #2}
7:     doLast {//doing sth #3}
8:     doLast {//doing sth #4}
9: }
```

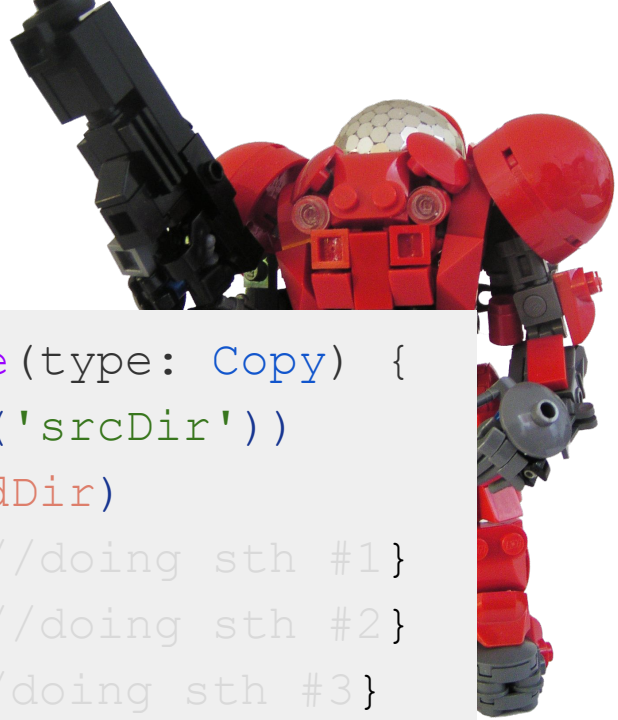
Порядок запуска



1. Task definition
2. Task doFirst
3. Task doLast

```
1: task release(type: Copy) {
2:     from(file('srcDir'))
3:     into(buildDir)
4:     doFirst {//doing sth #1}
5:     doFirst {//doing sth #2}
7:     doLast {//doing sth #3}
8:     doLast {//doing sth #4}
9: }
9: release.doLast {//do...#5}
```

Порядок запуска



1. Task definition

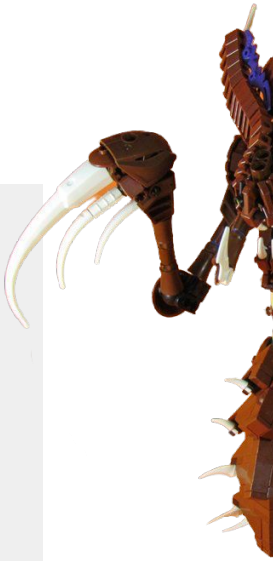
2. Task doFirst

3. Task doLast

```
1:  task release(type: Copy) {
2:    from(file('srcDir'))
3:    into(buildDir)
4:    doFirst {//doing sth #1}
5:    doFirst {//doing sth #2}
7:    doLast {//doing sth #3}
8:    doLast {//doing sth #4}
9:  }
9:  release.doLast {//do...#5}
10: release <<     {//do...#6}
```

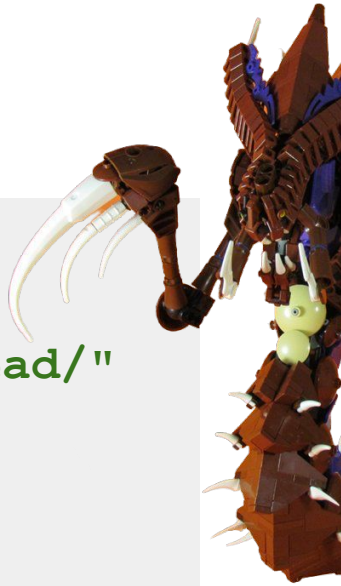
Или с Gradle

```
1: task preRelease {
2:  ...
```



Или с Gradle

```
1: task preRelease {
2:   doFirst {
4:     def br = file(".git/HEAD").text - "ref: refs/head/"
```



Groovy...

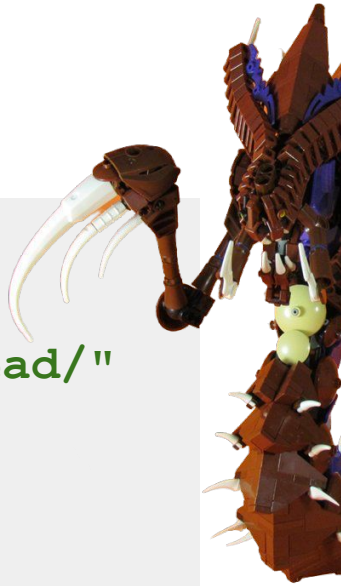
```
1: println "ref: refs/head/master" - "ref: refs/head/"
```

Groovy...

```
println "ref: refs/head/master" - "ref: refs/head/"
> "master "
```

Или с Gradle

```
1:  task preRelease {
2:    doFirst {
4:     def br = file(".git/HEAD").text - "ref: refs/head/"
```



Или с Gradle

```
1:  task preRelease {
2:    doFirst {
4:     def br = file(".git/HEAD").text - "ref: refs/head/"
5:     def cid = file(".git/refs/heads/$br").text"
```

Или с Gradle

```
1: task preRelease {
2:  doFirst {
4:   def br = file(".git/HEAD").text - "ref: refs/head/"
5:   def cid = file(".git/refs/heads/$br").text"
6:   def tag = grgit.tag.list()
7:    .first()
8:    .getName()
```

Или с Gradle

```
1:  task preRelease {
2:    doFirst {
4:     def br = file(".git/HEAD").text - "ref: refs/head/"
5:     def cid = file(".git/refs/heads/$br").text"
6:     def tag = grgit.tag.list()
7:       .first()
8:       .getName()
9:     version = tag + br && cid ? "-$br-$cid":""
10:}
11:tasks.release.dependsOn preRelease
```

Итого

v1.0.1-master-a6f5ac0677421f486bd273ba4016e51b7634afa9

Итого

v1.0.1-master-a6f5ac0677421f486bd273ba4016e51b7634afa9

- uncommitted

Итого

v1.0.1-master-a6f5ac0677421f486bd273ba4016e51b7634afa9

- uncommitted
- release

Итого

v1.0.1-master-a6f5ac0677421f486bd273ba4016e51b7634afa9

- uncommitted
- release
- dependencies

Метаданные

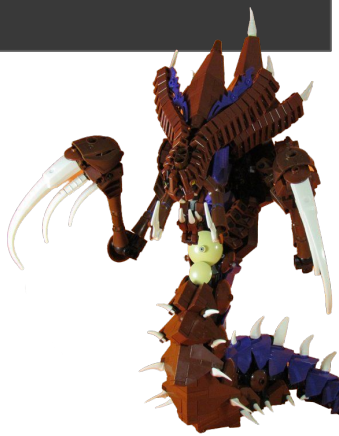
```
jar {
    manifest {
        attributes(
            "Manifest-Version"       : "1.0",
            "Implementation-Version": version,
        )
    }
}
```

Метаданные

```
jar {
    manifest {
        attributes(
            ["Manifest-Version"        :"1.0",
             "Implementation-Version":version,
            ] << System.getenv()
        )
    }
}
```

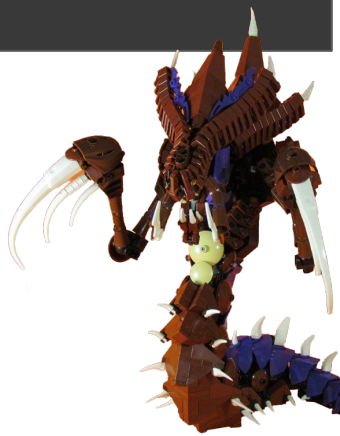
Manifest.mf

```
Manifest-Version: 1.0
Implementation-Title: ru.joker.demo#app;0.3.0-rc.2
Module-Origin: ssh://git@git/joker/demo-app.git
Build-Date: 2016-08-11_07:09:52
Build-Id: LOCAL
Implementation-Version: 0.3.0-rc.2
Build-Number: LOCAL
Built-By: tolkv
Change: 18d2261
Module-Source: /app
Branch: master
Spring-Boot-Version: 1.3.6.RELEASE
Built-Status: integration
```



Manifest.mf

```
Implementation-Version: 0.3.0-rc.2
Build-Number: LOCAL
Built-By: tolkv
Branch: master
Spring-Boot-Version: 1.3.6.RELEASE
Built-Status: integration
Gradle-Version: 2.13
Build-Job: LOCAL
Built-OS: Mac OS X
X-Compile-Target-JDK: 1.8
Build-Host: tolkv-work
Start-Class: ru.sense.user.UserApplication
X-Compile-Source-JDK: 1.8
Created-By: 1.8.0_60-b27 (Oracle Corporation)
Build-Java-Version: 1.8.0_60
```



Еще полезные ресурсы

- информация об окружении сборки **+X LOC**

Еще полезные ресурсы

- информация об окружении сборки **+X LOC**
- автор **+Y LOC**

Еще полезные ресурсы

- информация об окружении сборки **+X LOC**
- автор **+Y LOC**
- git окружение в manifest **+Z LOC**
- publish* **+M LOC**

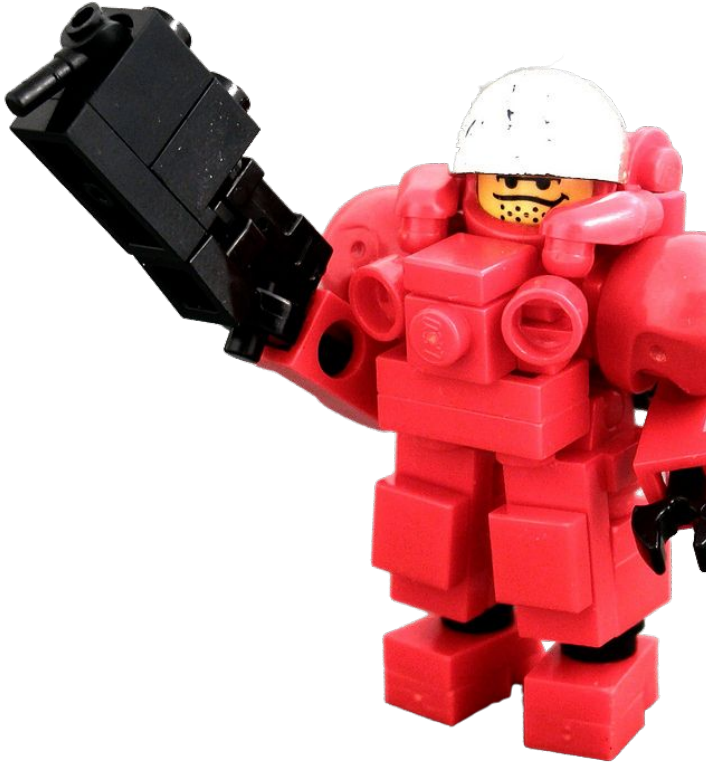
Еще полезные ресурсы

- информация об окружении сборки **+X LOC**
- автор **+Y LOC**
- git окружение в manifest **+Z LOC**
- publish* **+M LOC**

Все это в build.gradle

build.gradle

- от 3х строчек



build.gradle

- от 3х строчек
- до 2000

build.gradle

- от 3х строчек
- до 2000
- за 30 минут

build.gradle

- от 3х строчек
- до 2000
- за 30 минут
- типичная история

Как повторить это все?

Как повторить это все?

- в другой команде

Как повторить это все?

- в другой команде
- на другом проекте

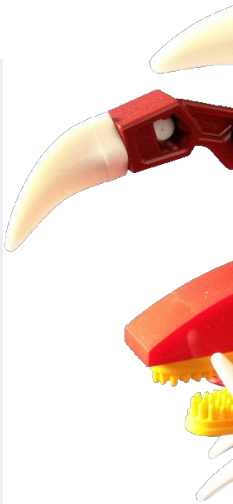
Думай как зерг!

Copy apply подход

```
build.gradle:
    apply from: "http://someorg.com/version.gradle"
```

Copy apply подход

```
build.gradle:
    apply from: "http://someorg.com/version.gradle"
    apply from: "http://someorg.com/maven.gradle"
```



Copy apply подход

```
build.gradle:
    apply from: "http://someorg.com/version.gradle"
    apply from: "http://someorg.com/maven.gradle"
    apply from: "http://someorg.com/check.gradle"
```

Copy apply подход

```
build.gradle:
    apply from: "http://someorg.com/version.gradle"
    apply from: "http://someorg.com/maven.gradle"
    apply from: "http://someorg.com/check.gradle"
    apply from: "http://someorg.com/findbugs.gradle"
    apply from: "http://someorg.com/verify.gradle"
    apply from: "http://someorg.com/publish.gradle"
    apply from: "http://someorg.com/awesome!.gradle"
```

Copy apply подход

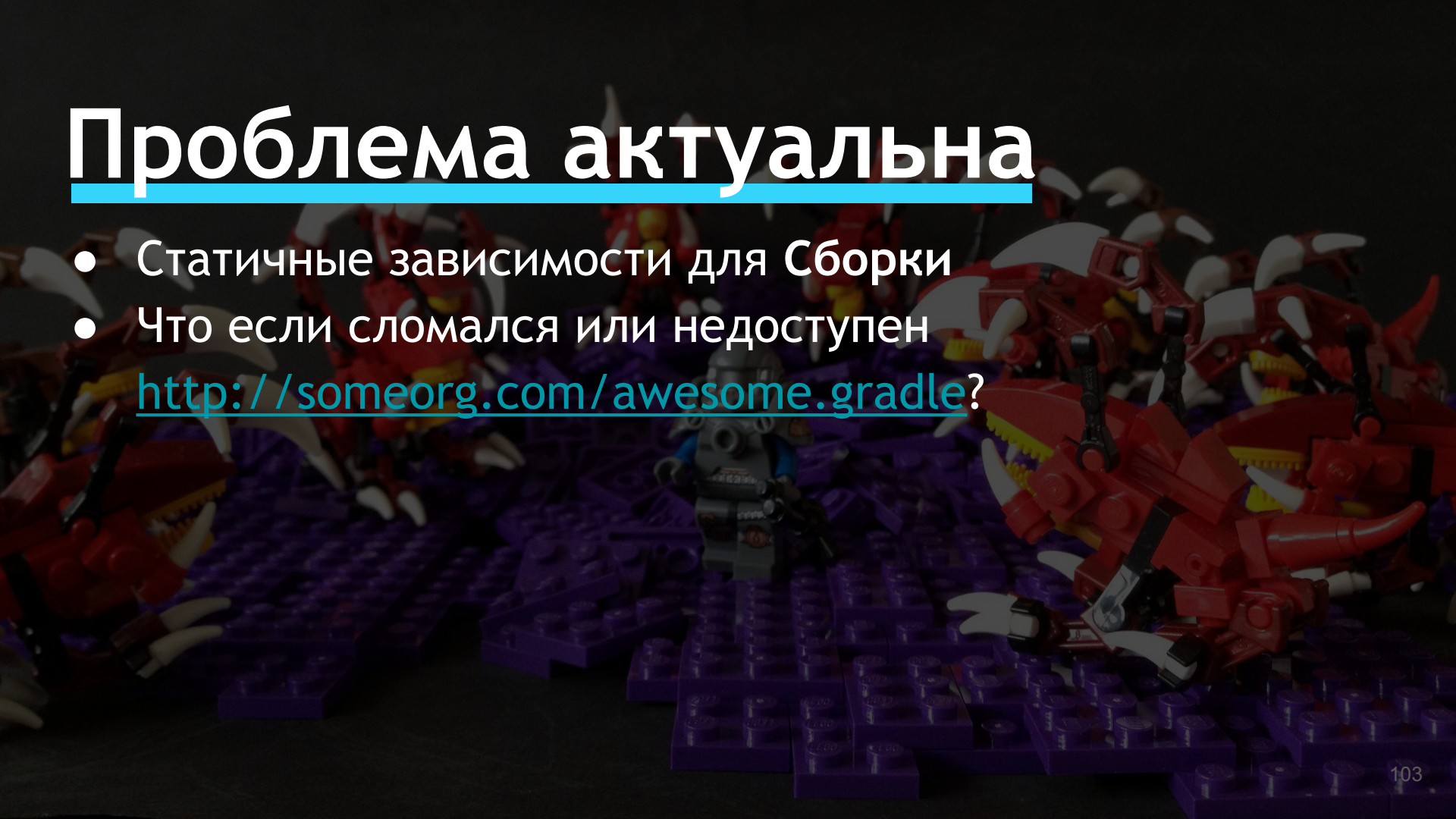
```
build.gradle:
    apply from: "http://someorg.com/version.gradle"
    apply from: "http://someorg.com/maven.gradle"
    apply from: "http://someorg.com/check.gradle"
    apply from: "http://someorg.com/findbugs.gradle"
    apply from: "http://someorg.com/verify.gradle"
    apply from: "http://someorg.com/publish.gradle"
    apply from: "http://someorg.com/awesome!.gradle"
    apply from: "http://someorg.com/v1/awesome!.gradle"
```

Проблема актуальна

- Статичные зависимости для **Сборки**

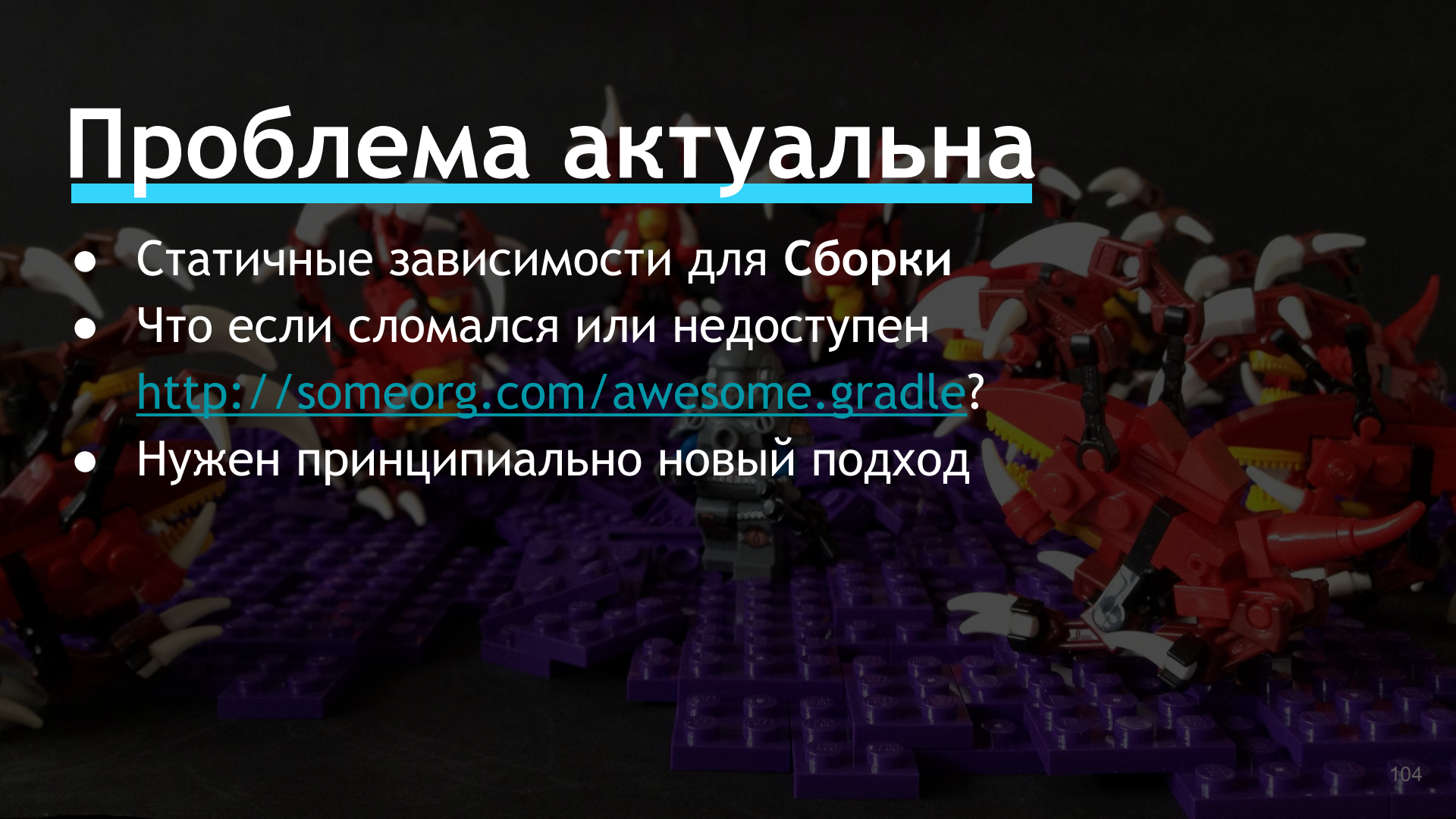
Проблема актуальна

- Статичные зависимости для **Сборки**
- Что если сломался или недоступен http://someorg.com/awesome.gradle?



Проблема актуальна

- Статичные зависимости для **Сборки**
- Что если сломался или недоступен http://someorg.com/awesome.gradle?
- Нужен принципиально новый подход



Новый подход

Императивность vs декларативность

- Статус проекта

Новый подход

Императивность vs декларативность

- Статус проекта
- Сложность проекта

Новый подход

Императивность vs декларативность

- Статус проекта
- Сложность проекта
- Поддержка

Новый подход

Императивность vs декларативность

- Статус проекта
- Сложность проекта
- Поддержка
- Перспективы

Декларативность

- Безопасно

Декларативность

- Безопасно
- Легко поддерживаемо

Декларативность

- Безопасно
- Легко поддерживаемо
- Понятно

Декларативный подход

```
apply from: .../version.gradle"
apply from: .../maven.gradle"
apply from: .../check.gradle"
apply from: .../findbugs.gradle"
apply from: .../verfify.gradle"
apply from: .../publish.gradle"
apply from: .../awesome!.gradle"
```

```
apply plugin: "yourplugin"
```

Декларативный подход

```
build.gradle:
    apply plugin: "version.plugin"
    apply plugin: "maven.plugin"
    apply plugin: "check.plugin"
    apply plugin: "findbugs.plugin"
    apply plugin: "verfify.plugin"
    apply plugin: "publish.plugin"
    apply plugin: "awesome.plugin"
```

Опять на те же грабли

Настраиваем каждый плагин:

- информация об окружении сборки **+X LOC**
- автор **+Y LOC**
- git окружение в manifest **+Z LOC**
- publish* **+M LOC**

Все это в build.gradle

Композиция плагинов

Сила в композиции

```
apply plugin: "your.plugin.all" //2.1.+
```

Сила в композиции

```
class MyPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.plugins.apply( ResolveProjectVersoinPlugin)
    }
}
```

Сила в композиции

```
class MyPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.plugins.apply( ResolveProjectVersoinPlugin)
        project.plugins.apply( AddGitTagPlugin)
    }
}
```

Сила в композиции

```
class MyPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.plugins.apply(ResolveProjectVersoinPlugin)
        project.plugins.apply(AddGitTagPlugin)
        project.plugins.apply(UserInfoPlugin)
    }
}
```

Сила в композиции

```
class MyPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.plugins.apply(ResolveProjectVersoinPlugin)
        project.plugins.apply(AddGitTagPlugin)
        project.plugins.apply(UserInfoPlugin)
        project.plugins.apply(DependencyLockPlugin)

    }
}
```

Сила в композиции

```
class MyPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.plugins.apply(ResolveProjectVersoinPlugin)
        project.plugins.apply(AddGitTagPlugin)
        project.plugins.apply(UserInfoPlugin)
        project.plugins.apply(DependencyLockPlugin)
        project.tasks.withType(Test) { Test testTask ->
            testTask.minHeapSize = '32m'
            testTask.maxHeapSize = '256m'
            testTask.jvmArgs "-XX:MaxPermSize=512m"
            testTask.jacocoTestReport.executionData += files(...)
            testTask.testLogging.exceptionFormat = 'full'
        }
    }
}
```

Сила в композиции

```
apply plugin: "nebula.nebula-release"
```

- управление Git
- управление номером версии
- контроль за состоянием репозитория

Что получили

```
$ ./gradlew snapshot / devSnapshot
$ ./gradlew candidate
$ ./gradlew final
```

```
Inferred project: ws, version:
0.18.0-dev.0.uncommitted+3791c32
```

Что получили

```
Inferred project: joker-demo-project,
version: 0.18.0-dev.0+uncommitted+3791c32
```

Что получили

```
Inferred project: joker-demo-project,
version: 0.18.0-dev.0+uncommitted+3791c32
```

Легко найти негодяя

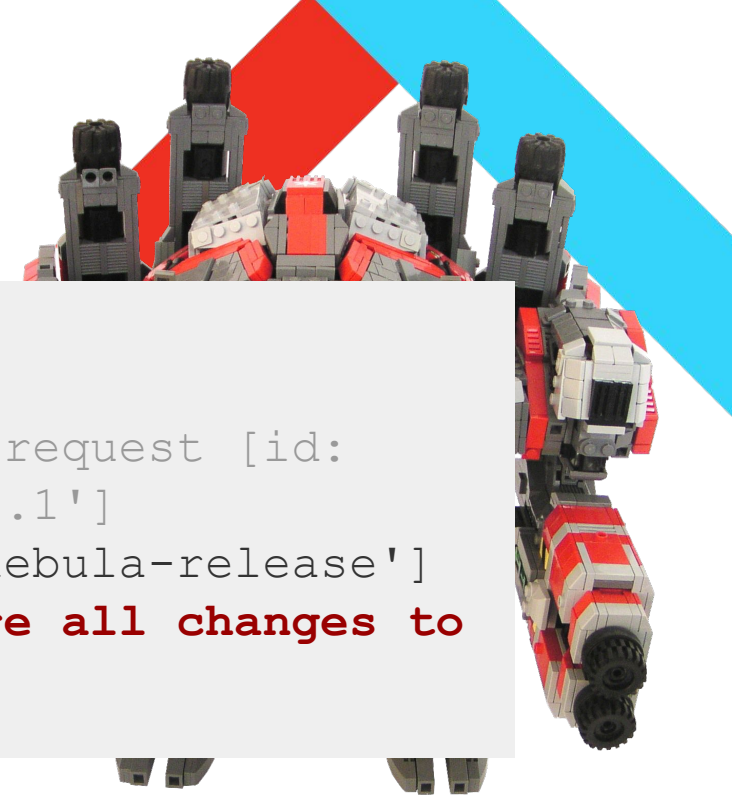
Что получили

```
Inferred project: joker-demo-project,
version: 0.18.0-dev.0+uncommitted+3791c32
```

Легко найти негодяя
Если есть метаинформация

Что получили

```
$ ./gradlew final
* What went wrong:
An exception occurred applying plugin request [id:
'nebula.nebula-release', version: '4.0.1']
> Failed to apply plugin [id 'nebula.nebula-release']
   > Final and candidate builds require all changes to
be committed into Git.
```

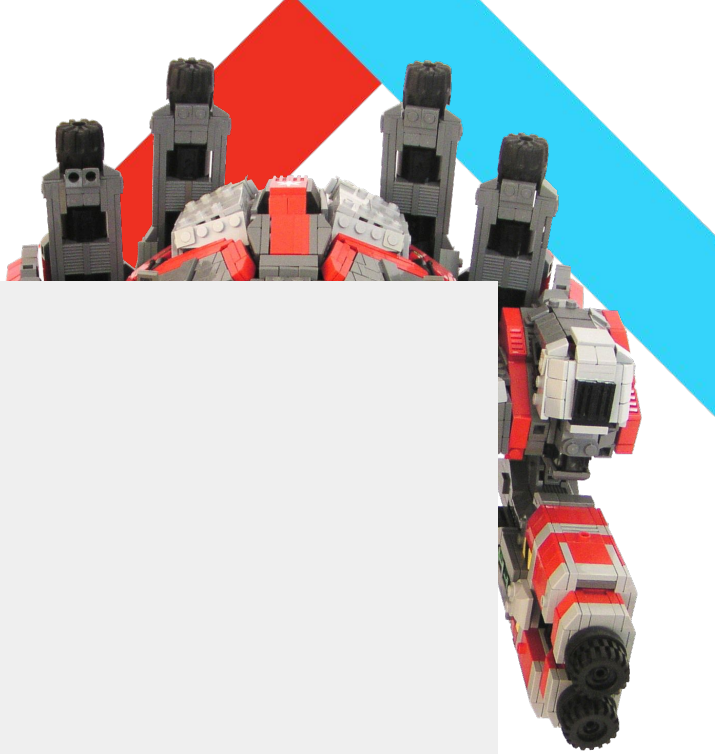


Что получили

```
$ git tag
v0.8.3
v0.9.0
v0.9.1
v0.10.0-rc.1
v0.10.0-rc.2
v0.10.0
v0.14.0
v0.15.0
v0.16.0
v0.17.0
```

Что получили

```
$ ./gradlew final
…
BUILD SUCCESSFUL

$ git tag
…
v0.16.0
v0.17.0
v0.18.0 # <--- new version
```

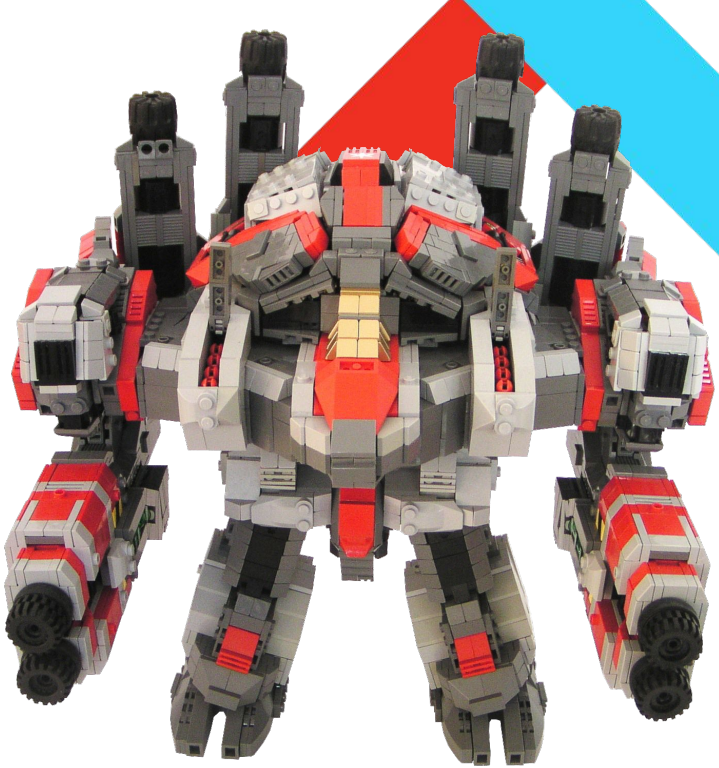


Что получили

```
$ gw final -Prelease.scope=patch
…
BUILD SUCCESSFUL

$ git tag
…
v0.16.0
v0.17.0
v0.18.0
v0.18.1 # <--- new version
```

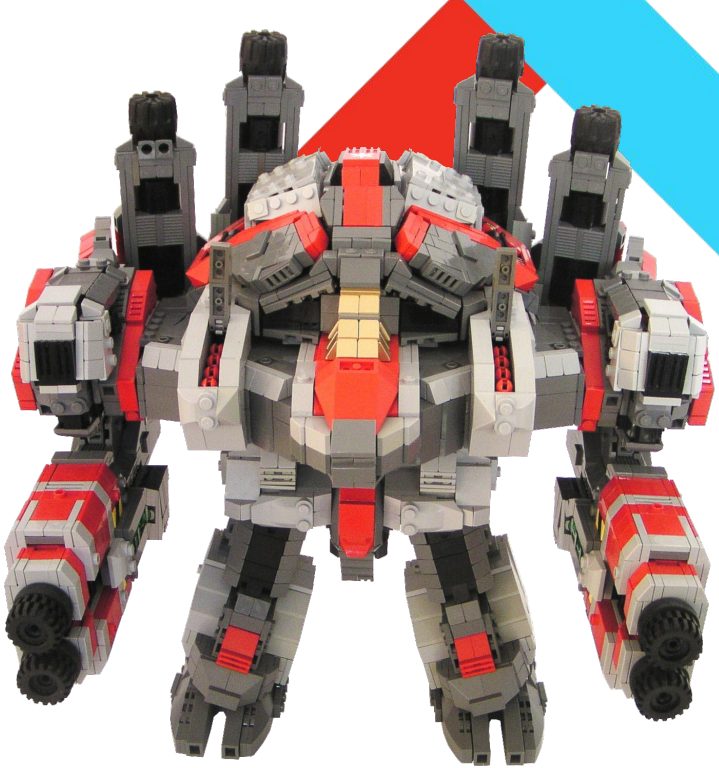
Что еще

- commit all for release check



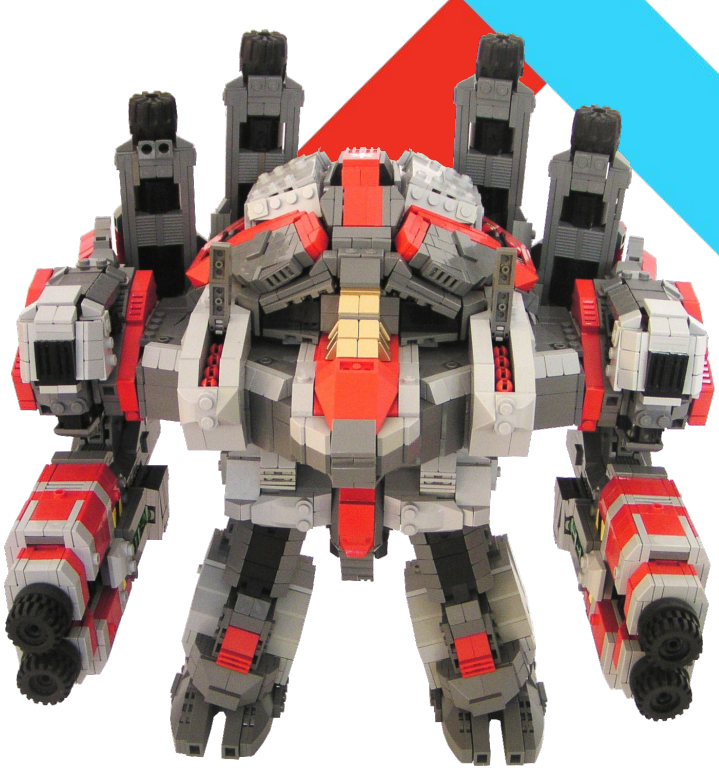
Что еще

- commit all for release check
- push to origin check



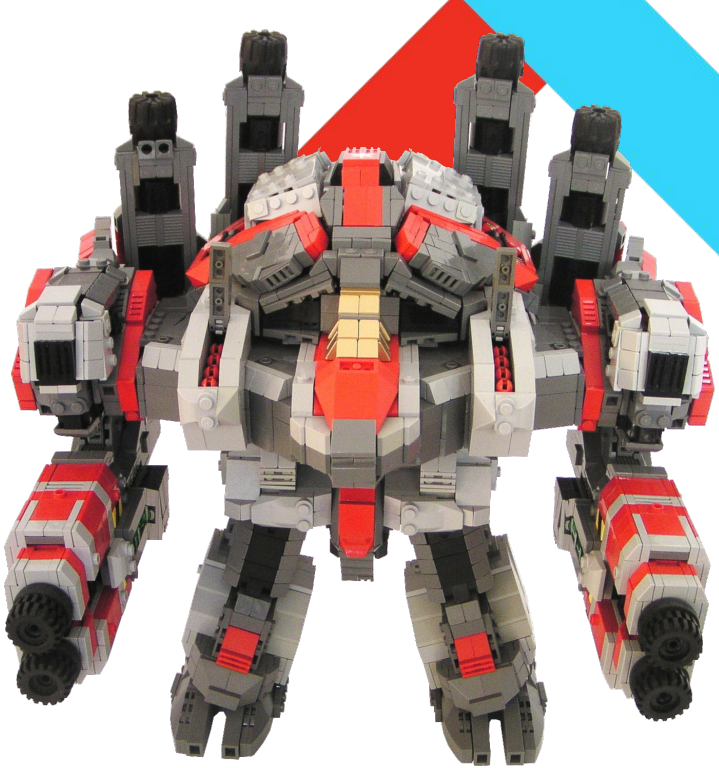
Что еще

- commit all for release check
- push to origin check
- add tags



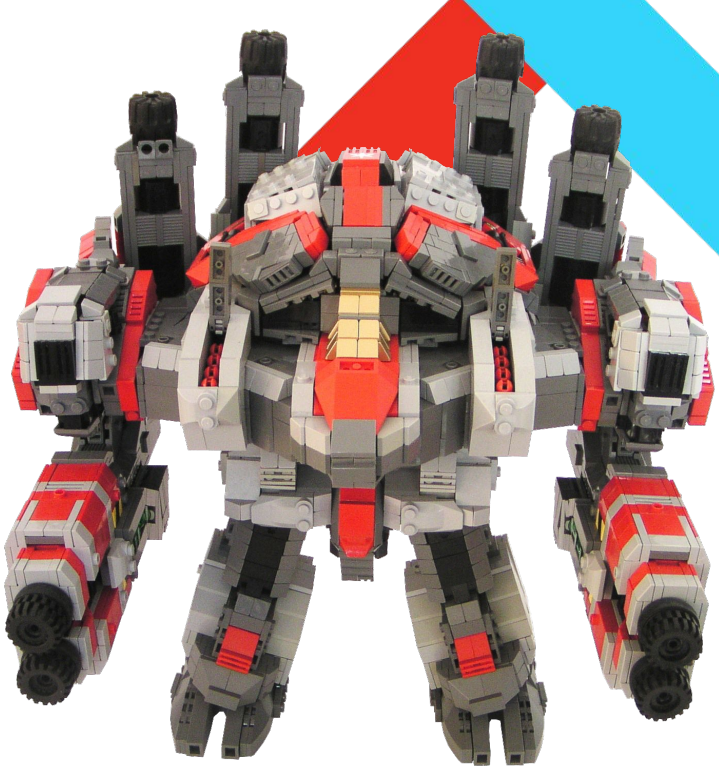
Что еще

- commit all for release check
- push to origin check
- add tags
- resolve version from tags and env



Что еще

- commit all for release check
- push to origin check
- add tags
- resolve version from tags
- tag in CI
- tag in local
- tag push



Доставляем

```
if [ "$TRAVIS_PULL_REQUEST" != "false" ]; then
  ./gradlew build
```

Доставляем

```
if [ "$TRAVIS_PULL_REQUEST" != "false" ]; then
 ./gradlew build
elif [ "$TRAVIS_PULL_REQUEST" == "false" ] && [ "$TRAVIS_TAG" == "" ]; then
 ./gradlew -Prelease.travisci=true snapshot
```

Доставляем

```
if [ "$TRAVIS_PULL_REQUEST" != "false" ]; then
 ./gradlew build
elif [ "$TRAVIS_PULL_REQUEST" == "false" ] && [ "$TRAVIS_TAG" == "" ]; then
 ./gradlew -Prelease.travisci=true snapshot
elif [ "$TRAVIS_PULL_REQUEST" == "false" ] && [ "$TRAVIS_TAG" != "" ]; then
 case "$TRAVIS_TAG" in
 *-rc\.*)
   ./gradlew -Prelease.travisci=true -Prelease.useLastTag=true candidate ;;
 *)
   ./gradlew -Prelease.travisci=true -Prelease.useLastTag=true final
publishPlugins ;;
 esac ...
```

Доставляем

```
if [ "$TRAVIS_PULL_REQUEST" != "false" ]; then
 ./gradlew build
elif [ "$TRAVIS_PULL_REQUEST" == "false" ] && [ "$TRAVIS_TAG" == "" ]; then
 ./gradlew -Prelease.travisci=true snapshot
elif [ "$TRAVIS_PULL_REQUEST" == "false" ] && [ "$TRAVIS_TAG" != "" ]; then
 case "$TRAVIS_TAG" in
 *-rc\.*)
   ./gradlew -Prelease.travisci=true -Prelease.useLastTag=true candidate ;;
 *)
   ./gradlew -Prelease.travisci=true -Prelease.useLastTag=true final
publishPlugins ;;
 esac ...
```

Императивный подход

- Классика Gradle

Императивный подход

- Классика Gradle
- Опасное ружье

Императивный подход

- Классика Gradle
- Опасное ружье
- Сложно контролировать

Контроль

- CheckStyle

Контроль

- CheckStyle
- PMD

Контроль

- CheckStyle
- PMD
- Codenarc

Контроль

- CheckStyle
- PMD
- Codenarc
- Nebula Lint Plugin

Gradle Linter

```
dependencies {
    compile 'org.slf4j:slf4j-api:1.7.21'
    compile  group:'junit', name:'junit', version:'+'
}
```

Gradle Linter

```
plugins {
  id 'nebula.lint' version '5.1.0'
}

gradleLint.rules = ['all-dependency']

dependencies {
    compile 'org.slf4j:slf4j-api:1.7.21'
    compile  group:'junit', name:'junit', version:'+'
}
```

Gradle Linter

```
$ ./gradlew build
…

this dependency is unused and can be removed
warning   unused-dependency
build.gradle:13
compile 'org.slf4j:slf4j-api:1.7.21'

✖ build.gradle: 2 problems (0 errors, 2 warnings)

To apply fixes automatically, run fixGradleLint,
review, and commit the changes.
```

Gradle Linter

```
$ ./gradlew build
…

this dependency is unused and can be removed
warning   unused-dependency
build.gradle:13
compile 'org.slf4j:slf4j-api:1.7.21'

✖ build.gradle: 2 problems (0 errors, 2 warnings)

To apply fixes automatically, run fixGradleLint,
review, and commit the changes.
```

Gradle Linter

```
$ ./gradlew build fixGradleLint
…

Corrected 2 lint problems

BUILD SUCCESSFUL

Total time: 0.714 secs
```

Сделать приятно всем?

Сделать приятно всем?

- приятно

Сделать приятно всем?

- приятно
- потому что можем

Сделать приятно всем?

- приятно
- потому что можем
- gradle позволяет

Сделать приятно всем!

Кто нам поможет?

Сделать приятно всем!

Кто нам поможет?

`$HOME_DIR/.gradle/init.gradle`

Сделать приятно всем!

```
~/.gradle/init.gradle:

allprojects {
  apply plugin: GradleLintPlugin
  gradleLint.rules = [ 'all-dependency',
                       'dependency-parentheses',
                       'dependency-tuple']
}
```

Сделать приятно всем!

```
dependencies {
    compile 'org.slf4j:slf4j-api:1.7.21'
    compile  group:'junit', name:'junit', version:'+'
}
```

Сделать приятно всем!

```
$ ./gradlew build
…

use the shortcut form of the dependency
warning   dependency-tuple   build.gradle:

compile  group:'junit', name:'junit', version:'+'

✖ build.gradle: 1 problems (0 errors, 2 warnings)

To apply fixes automatically, run fixGradleLint,
review, and commit the changes.
```

Сделать приятно всем!

```
$ ./gradlew build
…

use the shortcut form of the dependency
warning   dependency-tuple   build.gradle:

compile  group:'junit', name:'junit', version:'+'

✖ build.gradle: 1 problems (0 errors, 2 warnings)

To apply fixes automatically, run fixGradleLint,
review, and commit the changes.
```

Сделать приятно всем!

```
$ ./gradlew build
…

use the shortcut form of the dependency
warning   dependency-tuple   build.gradle:

compile  group:'junit', name:'junit', version:'+'

✖ build.gradle: 1 problems (0 errors, 2 warnings)

To apply fixes automatically, run fixGradleLint,
review, and commit the changes.
```

:fixGradleLint

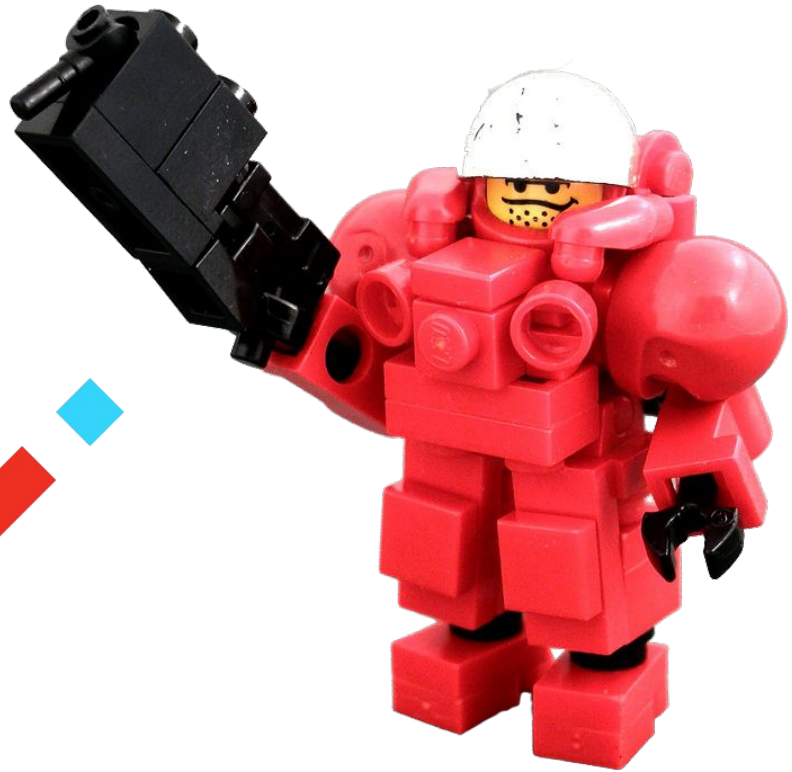
```
dependencies {
    compile 'org.slf4j:slf4j-api:1.7.21'
    compile 'junit:junit:latest.release'
}
```

:fixGradleLint

```
dependencies {
    compile 'org.slf4j:slf4j-api:1.7.21'
    compile 'junit:junit:latest.release'
}
```

version: 'latest.release'

- Много зависимостей
- У каждой своя версия
- Придумали Bom



Альтернативы

- Использовать всегда **последнюю** версию
- не все готовы
- отнимает время в самый **неподходящий момент**

Lock Dependencies

- Создали проблему - решили проблему
- Lock перед каждым релизом
- Апдейт → Тесты → Update Lock

Lock Dependencies

```
compile 'joker.org:demo-lib1: latest.release'
...
compile 'joker.org:demo-libN: latest.release'

$ ./gradlew generateLock saveLock
$ ./gradlew test
$ ./gradlew commitLock
```

Lock Dependencies

```
$ cat dependencies.lock
{
    "compile": {
        "com.google.guava:guava": {
            "locked": "14.0.1",
            "requested": "14.+"
        }
    },
    "default": {
        "com.google.guava:guava": {
            "locked": "14.0.1",
            "requested": "14.+"
        }
    },
    "runtime": {
        "com.google.guava:guava": {
            "locked": "14.0.1",
```

Release больших кусков

- Релиз менеджмент платформы
- Много зависимостей
- Инструмент для контроля

Release больших кусков

- Релиз менеджмент платформы
- Много зависимостей
- Инструмент для контроля (не для зергов)

Dependency Control

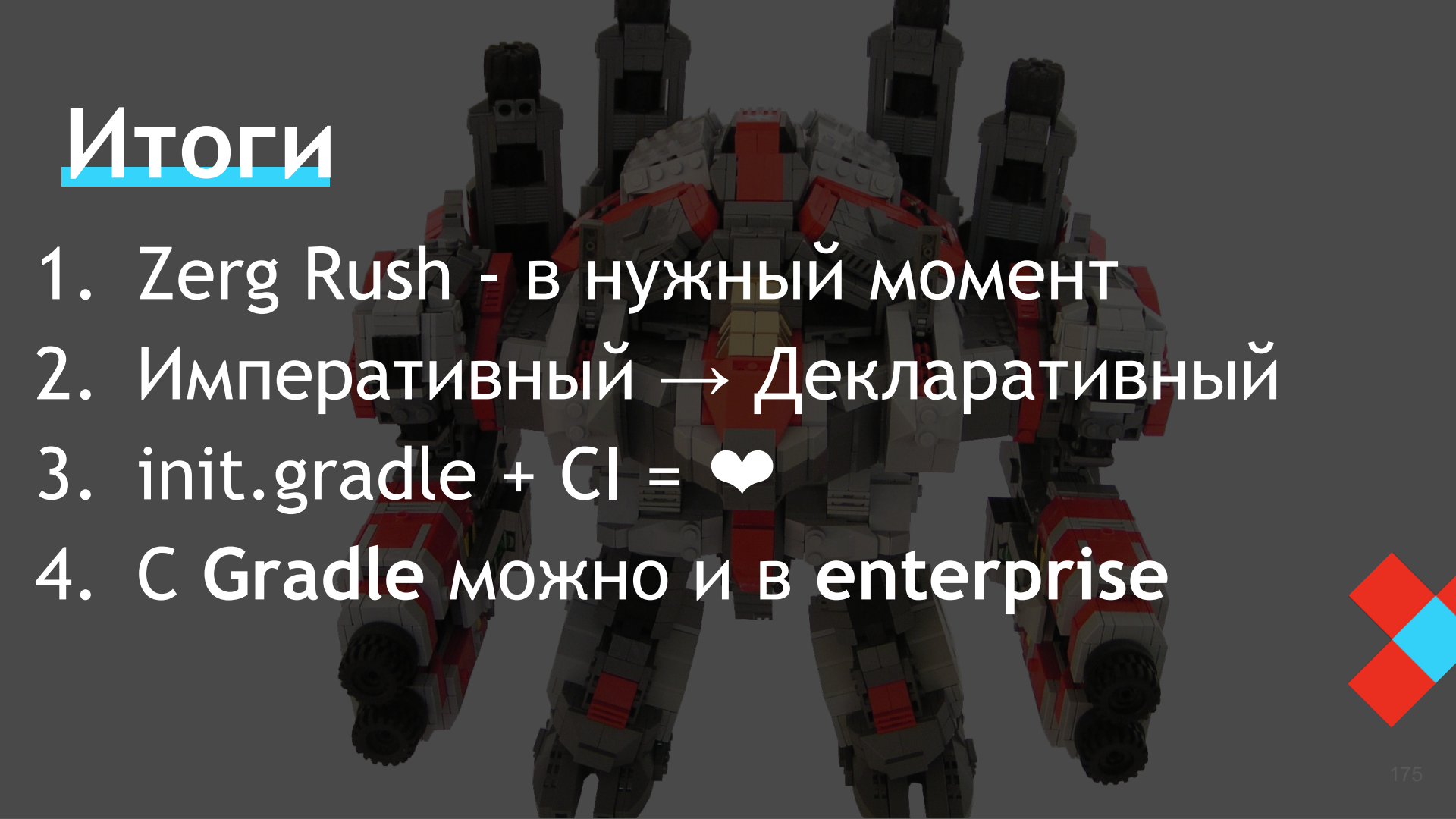
```
apply plugin: 'nebula.dependency-recommender'

dependencyRecommendations {
   mavenBom module: 'netflix:platform: latest.release'
   propertiesFile uri: ' http://yourorg/extlib-bundle.properties ', name: 'prop'
}

dependencyRecommendations {
   map recommendations: ['commons-logging:commons-logging': '1.1']
}

dependencies {
   compile 'commons-logging:commons-logging' // version 1.1 is selected
}
```

Dependency Control

```
dependencyRecommendations {
   mavenBom module: 'netflix:platform: latest.release'
   propertiesFile uri: ' http://yourorg/extlib-bundle.properties ', name: 'prop'
}

dependencyRecommendations {
   map recommendations: ['commons-logging:commons-logging': '1.1']
}

dependencies {
   compile 'commons-logging:commons-logging:1.0' // version 1.0 is selected
}
```

Сила в композиции

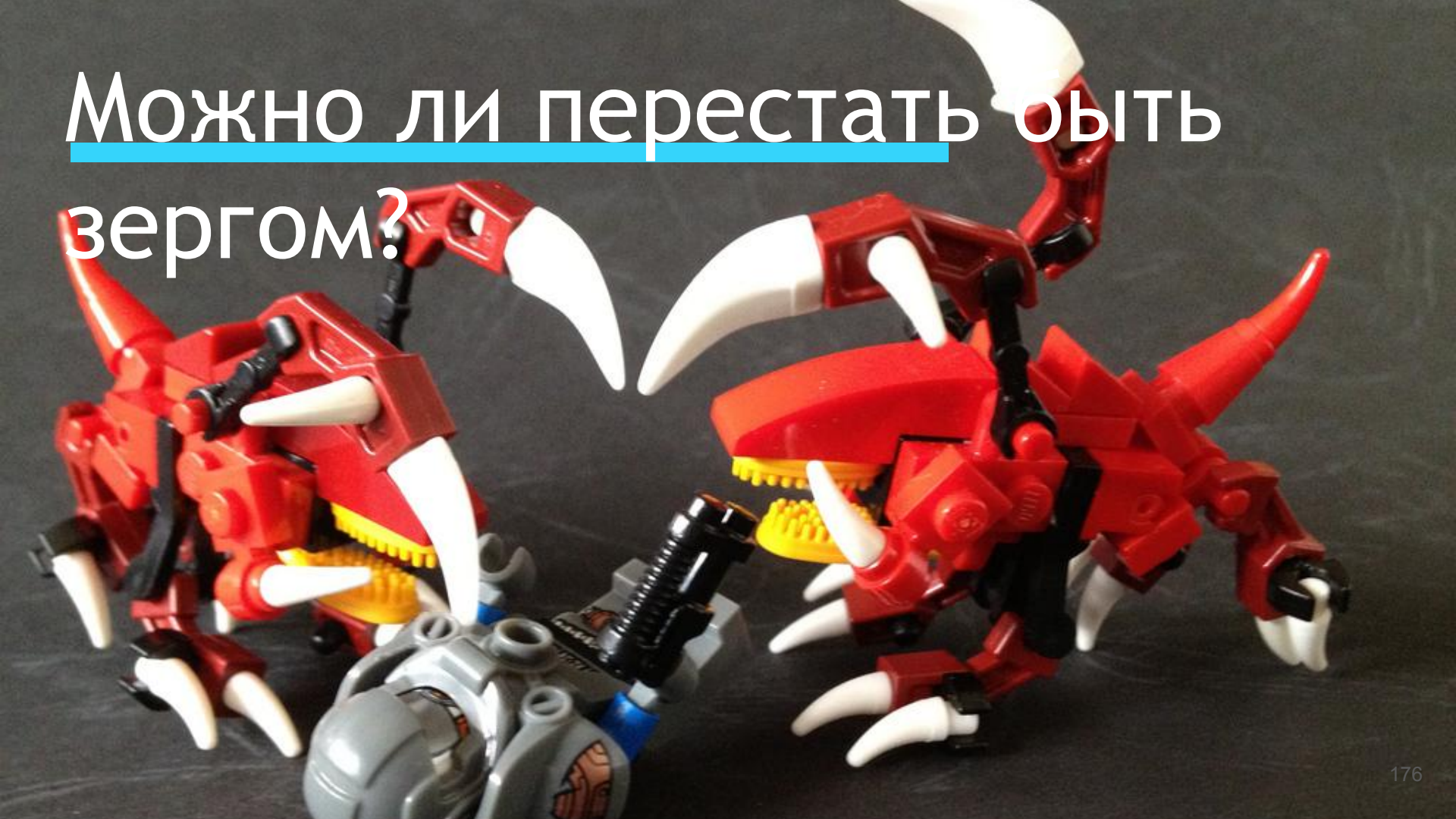
```
Plugin<Project> {
    void apply(Project project) {
        project.plugins.apply( GradleLintPlugin)

        gradleLint.rules = [
             'dependency-parentheses', 'dependency-tuple'
        ]

        project.plugins.apply( JavadocJarPlugin)
        project.plugins.apply( SourceJarPlugin)
        project.plugins.apply( DependencyLockPlugin)
        project.tasks.withType(Test) { Test testTask ->
            testTask.testLogging.exceptionFormat = 'full'
        }
    }
}
```



Итоги

1. Zerg Rush - в нужный момент
2. Императивный ⟶ Декларативный
3. init.gradle + CI = ❤
4. С **Gradle** можно и в **enterprise**

Можно ли перестать быть зергом?



GrGit - https://github.com/ajoberstar/grgit

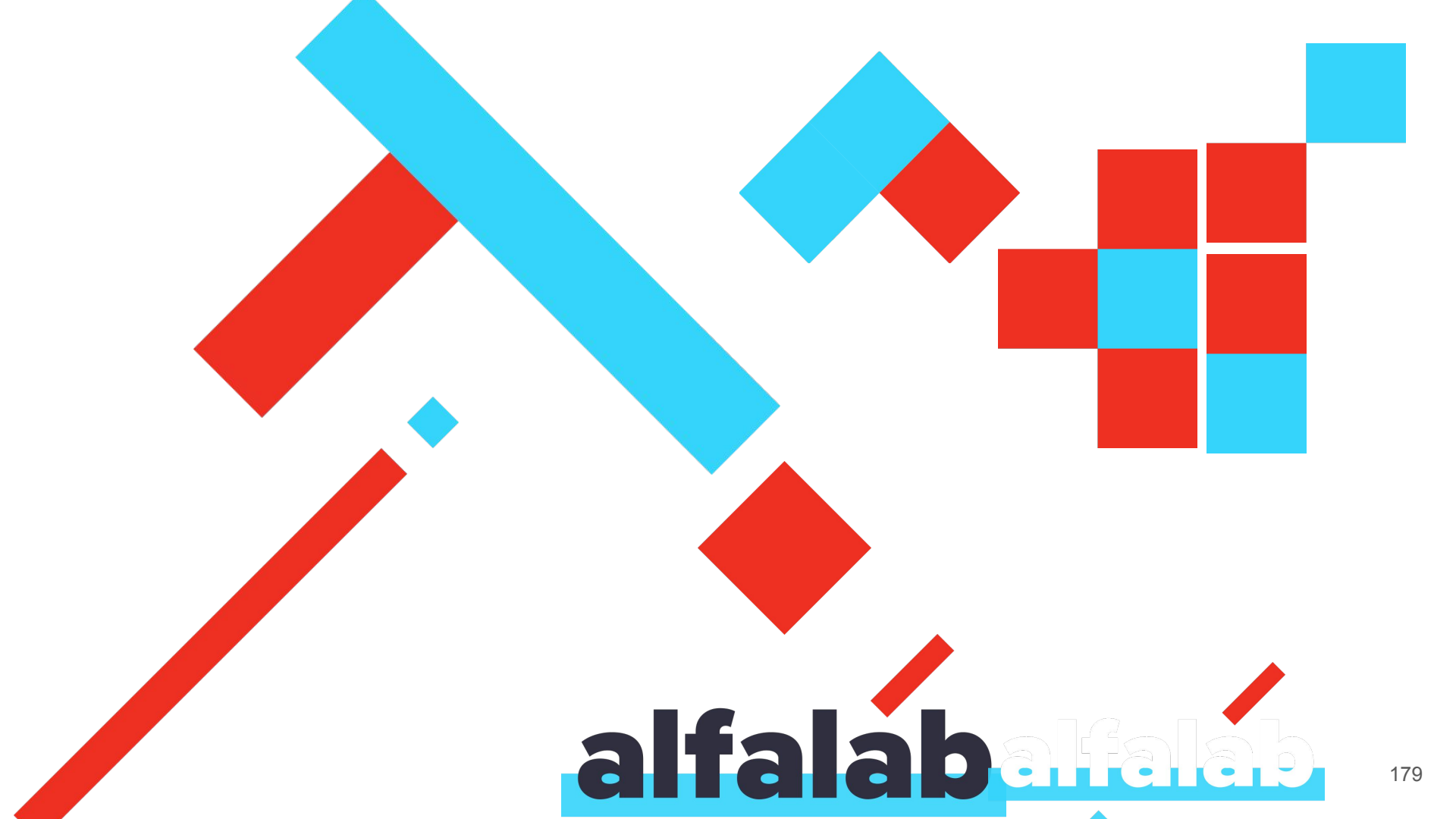
Nebula Release Plugin -

Nebula Lint Plugin - https://github.com/nebula-plugins/gradle-lint-plugin

Nebula Dependency Lock Plugin

Nebula Dependency Recommender Plugin

QA


alfalab
alfalab





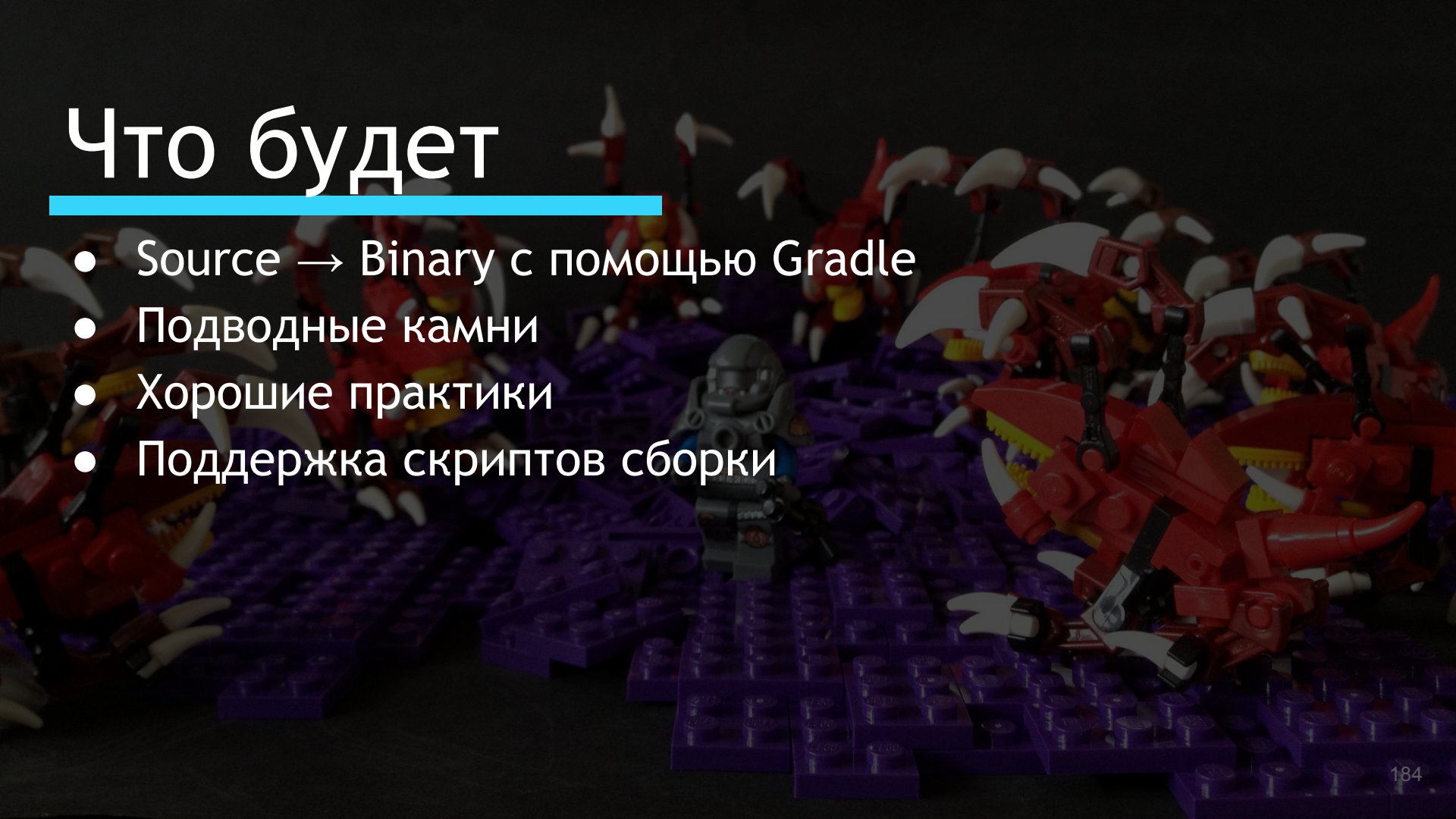


QA


Нельзя просто так взять и
зарелизить

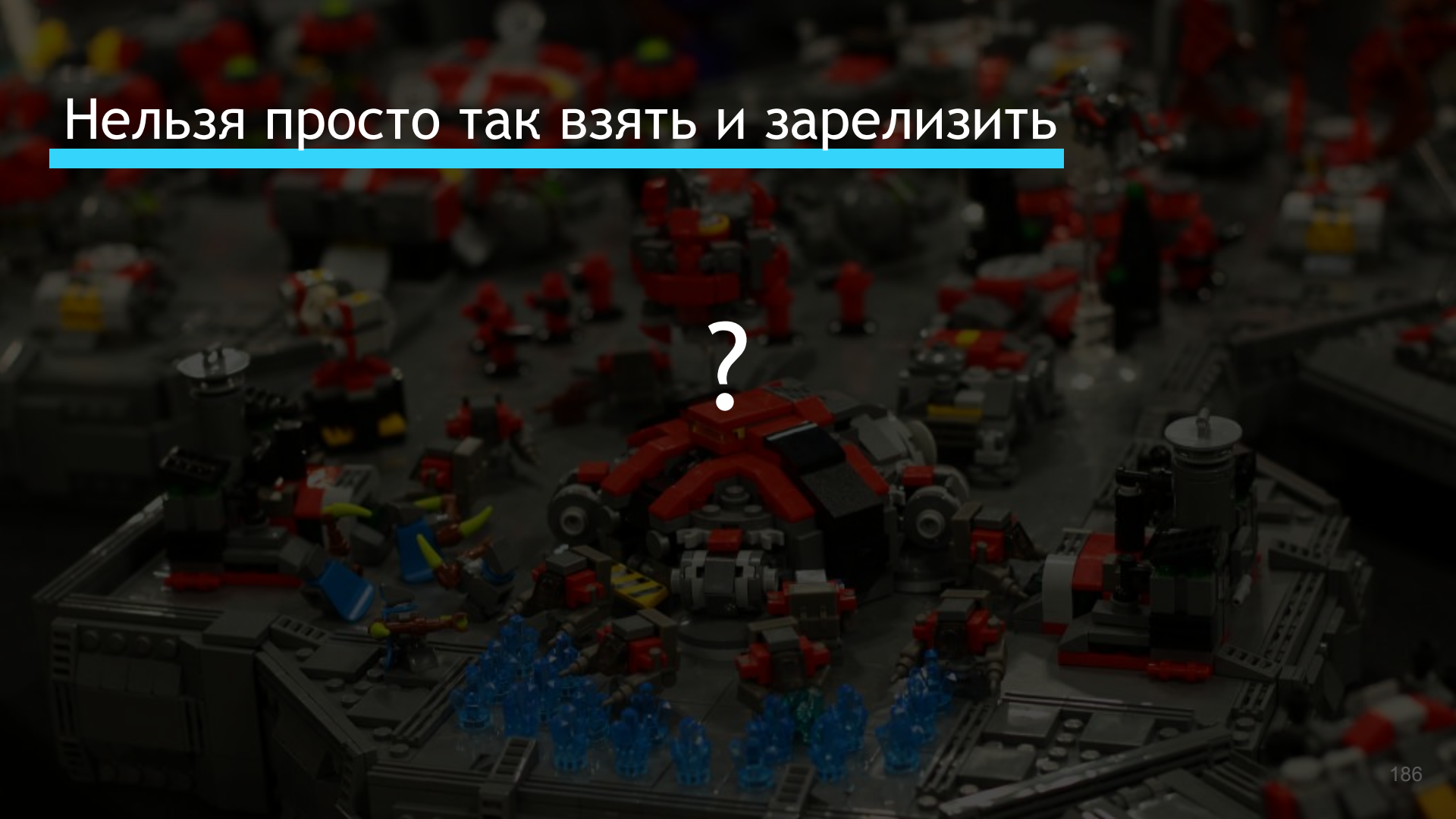
Что будет

- Source → Binary с помощью Gradle
- Подводные камни
- Хорошие практики
- Поддержка скриптов сборки



История проблемы

Нельзя просто так взять и зарелизить



Нельзя просто так взять и зарелизить

VCS 

Нельзя просто так взять и зарелизить

VCS   Conflicts

Нельзя просто так взять и зарелизить

VCS 

 Conflicts

Tools 

Нельзя просто так взять и зарелизить

VCS 

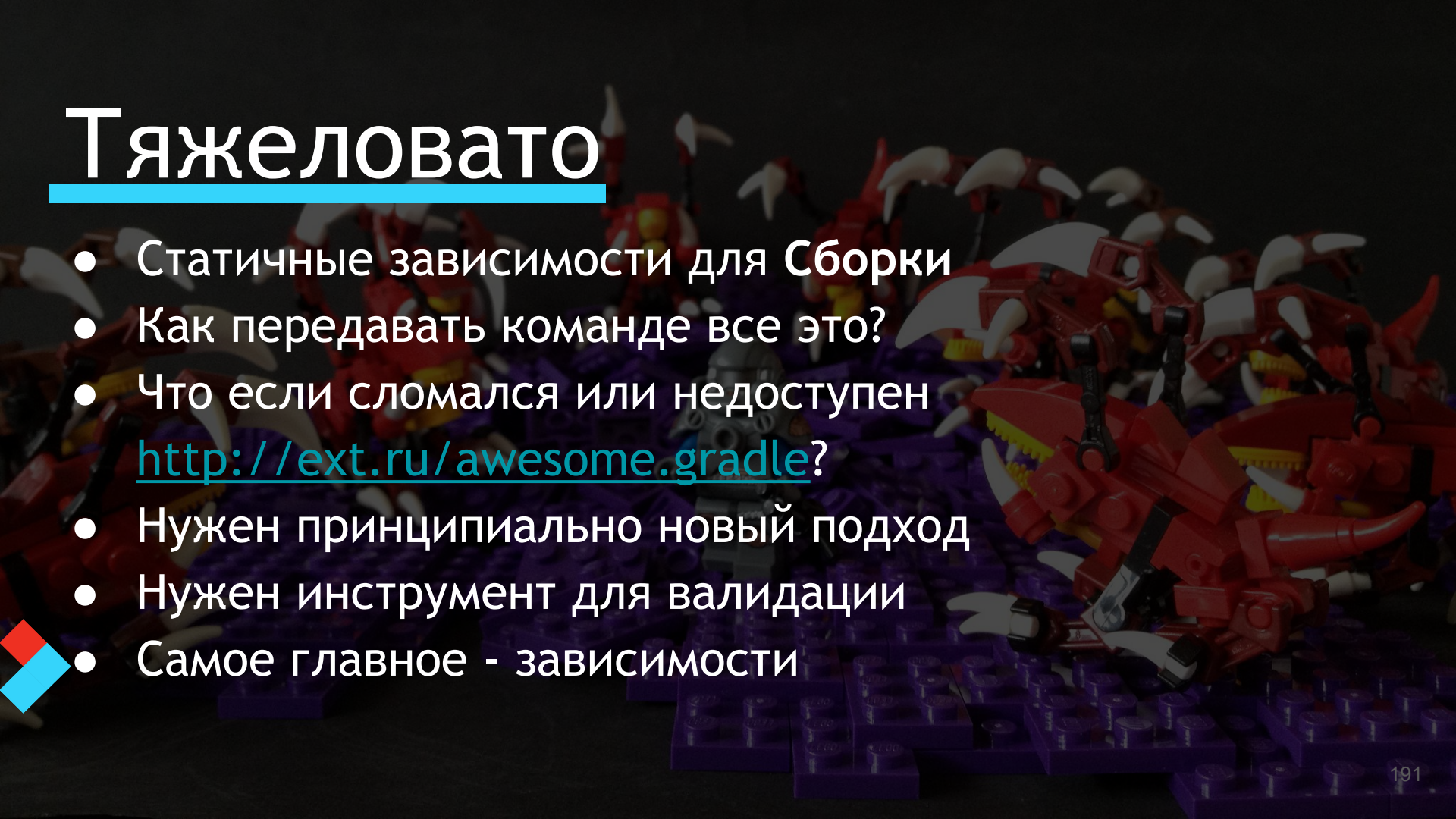
 Conflicts

Tools 

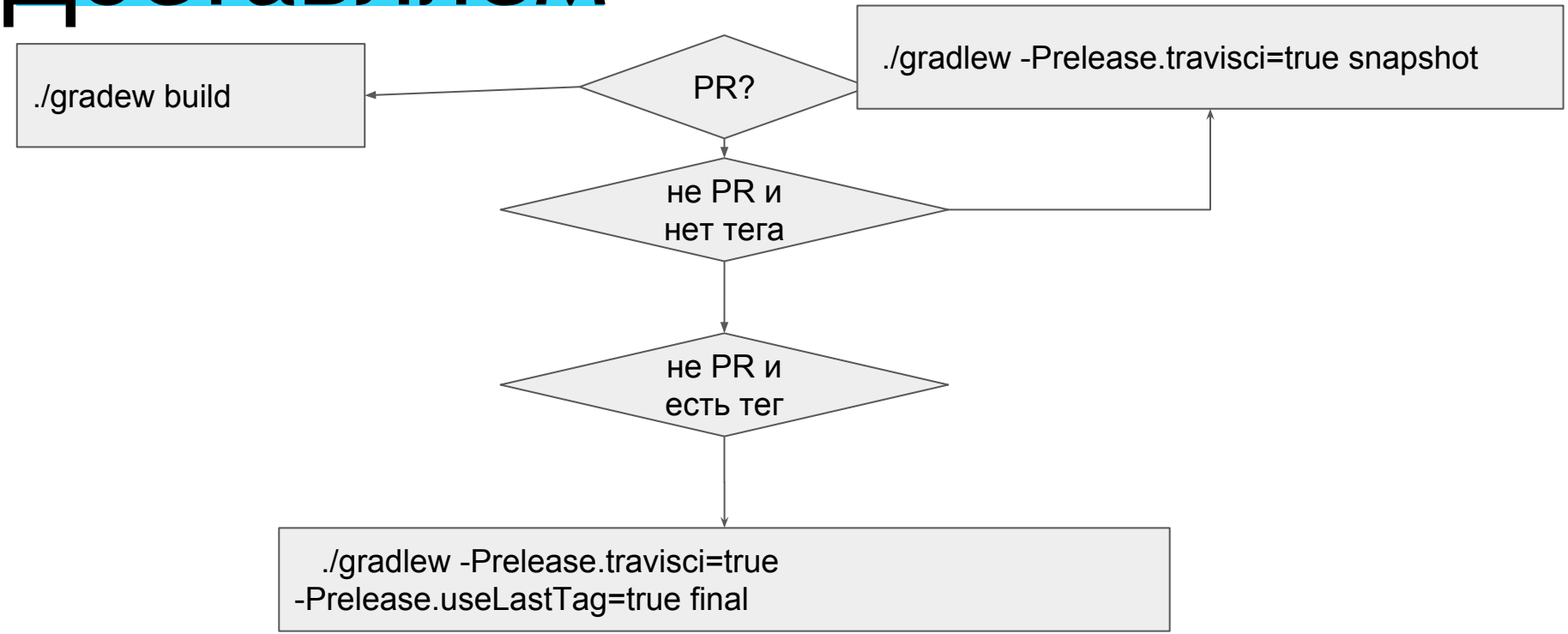
 Delivery

Тяжеловато

- Статичные зависимости для **Сборки**
- Как передавать команде все это?
- Что если сломался или недоступен http://ext.ru/awesome.gradle?
- Нужен принципиально новый подход
- Нужен инструмент для валидации
- Самое главное - зависимости



Доставляем


./gradew build
PR?
./gradlew -Prelease.travisci=true snapshot
не PR и
нет тега
не PR и
есть тег
./gradlew -Prelease.travisci=true
-Prelease.useLastTag=true final