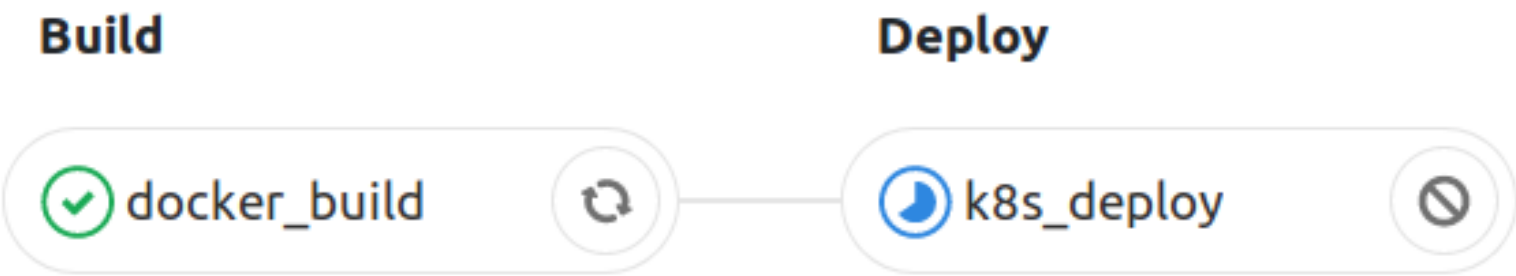
ANTI-YAML:
DSL IS THE NEW BLACK

АНДРЕЙ ЕРМАКОВ, TINKOFF.RU

- Андрей Ермаков
  Ведущий разработчик
- Telegram: dreef3

- >~30 микросервисов
- .NET Core, Scala, Kotlin, Node.js, Python
- Ежедневные релизы

1. YAML на примере GitLab CI
2. DSL – что это за зверь?
3. При чем тут ваш CI-сервер?
4. Личный опыт


Build
Deploy
docker_build
k8s_deploy

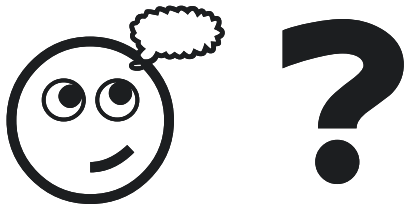
```yaml
stages:
- build
- deploy

build:
# ...
deploy:
# ...
```

```
build:
  image: node:8
  stage: build
  script:
  - npm ci
  - npm build
  - docker build -t my-app:latest .
  - docker push my-app:latest
```

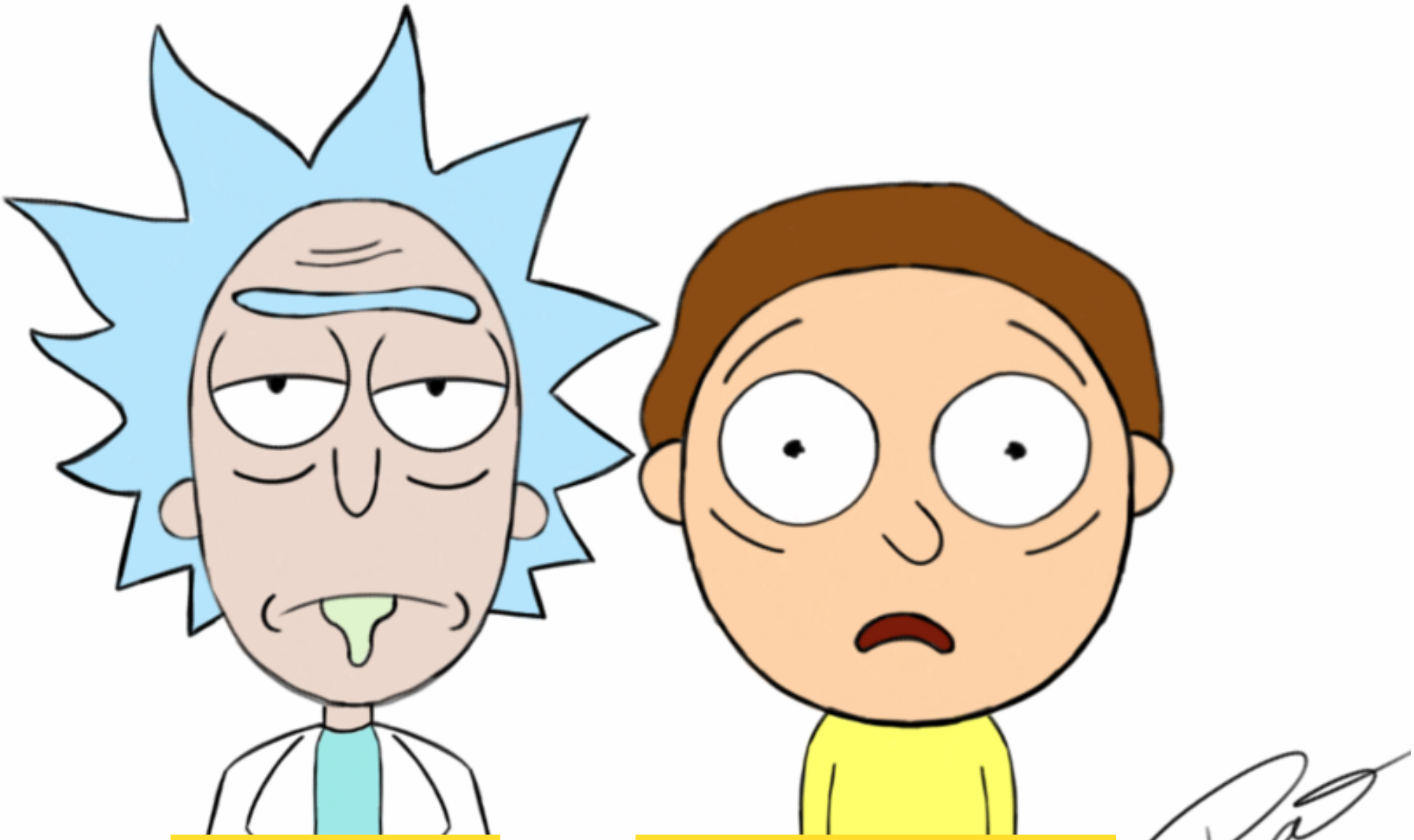
```
deploy:
  image: helm
  stage: deploy
  script:
  - helm upgrade -i my-app ./my-app-chart
```

```
$INCREMENTAL_ROLLOUT_ENABLED

rollout 50%:
  <<: *rollout_template
  variables:
    ROLLOUT_PERCENTAGE: 50
  when: manual
  only:
    refs:
      - master
    kubernetes: active
    variables:
      - $INCREMENTAL_ROLLOUT_ENABLED

rollout 100%:
  <<: *production_template
  when: manual
  allow_failure: false
  only:
    refs:
      - master
    kubernetes: active
    variables:
      - $INCREMENTAL_ROLLOUT_ENABLED
```

- Коварен – легко начать на простых примерах
- Опасно рефакторить, IDE нам не помощник




DSL НЕ ЖЕЛАЕТЕ?


KOTLIN
KOTLIN DSL

1. Интерактивный: описываем, исполняем
2. Декларативный: описываем, конвертируем в X

   X = YAML, XML,..

```
newIngress {
    metadata {
        name = "example-ingress"
    }
    spec {
        backend {
        serviceName = "example-service"
        servicePort = IntOrString(8080)
        }
    }
}
```

```
plugins {
    application
    kotlin("jvm") version "1.2.61"
}

application {
    mainClassName = "samples.HelloWorldKt"
}

dependencies {
    compile(kotlin("stdlib"))
}
```

ПРИ ЧЕМ ТУТ CI/CD?


Groovy

```
pipeline {
    stages {
        stage('Build') {
            // ...
        }
        stage('Deploy') {
            // ...
        }
    }
}
```

```
agent { docker { image 'node:8' } }
steps {
  sh 'npm ci'
  sh 'npm test'
  sh 'npm build'
  script {
    docker.build("my-app:release").push()
  }
}
```

```
agent {
    docker { image 'helm' }
}
steps {
    sh '''
    helm upgrade -i my-app ./my-app-chart
    '''
}
```

```
project {
    buildType {
        id = AbsoluteId("build")
    }

    buildType {
        id = AbsoluteId("deploy")
    }
}
```

```
params {
    param("plugin.docker.imageId", "node:8")
}
steps {
    script {
        scriptContent = """npm ci"""
    }
    // ...
}
```

```
params { param("plugin.docker.imageId", "helm") }
steps {
    script {
        scriptContent = """
        helm upgrade -i my-app ./my-app-chart
        """
    }
}
dependencies {
    dependency(AbsoluteId("build")) { snapshot() }
}
```

BOILERPLATE

vars/npm.groovy

```groovy
def installModules(String name) {
    sh """
    nvm use env.NODE_VERSION
    npm ci
    """
}
```

vars/helmUp.groovy

```
def call(String release, String path) {
    sh """
    helm upgrade $release $path
    """
}
```

Jenkinsfile

```
@Library('common-steps') _

pipeline {
    stages('Build') {
        npm.installModules()
        npm.test()
        npm.build()
        docker.build("my-app:release").push()
    }
}
```

Jenkinsfile

```
@Library('pipelines') _

nodeJsApp()
```

kafkaTopic.kt

```
fun BuildSteps.kafkaTopic(topic: String) {
    script {
        name = "Create kafka topic"
        scriptContent = """
        kafka-topics --create --topic $topic
        """.trimIndent()
    }
}
```

settings.kts

```
import util.kafka.kafkaTopic

project {
    buildType {
        steps {
            kafkaTopic("my-topic")
        }
    }
}
```

- Навыки программирования
- Требуется IDE, инструменты сборки

- Разработчик? **Мы уже умеем писать код**
- Типизация, мощность языка
- Поддержка IDE, tooling

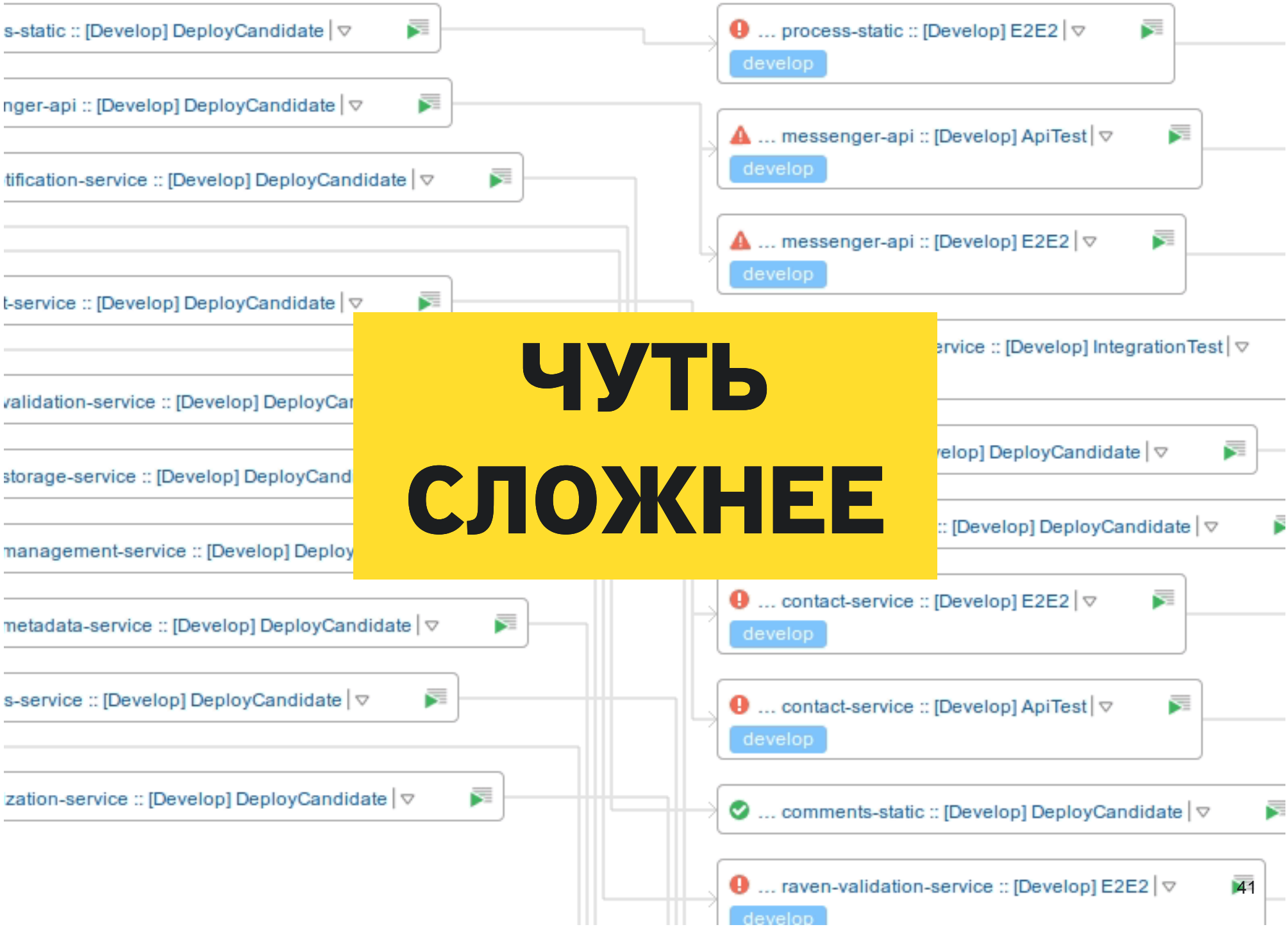
ТЕСТИРОВАНИЕ ИЗМЕНЕНИЙ


КТО ЖЕ НЕ ТЕСТИРУЕТ?
GIT COMMIT; GIT PUSH


КОГДА СЛОМАЛ ВСЕ БИЛДЫ

```
void should_make_kittens_cute() {
  binding.setVariable('AWESOMENESS', '1')

  def script = loadScript("Jenkinsfile")
  script.execute()
}
```

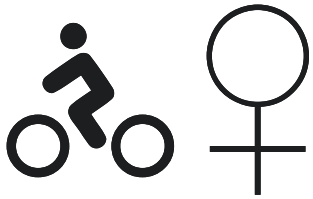
```
@Test
fun `shell scripts have valid shebang`() {
  allSteps.forEach {
    if (it is ScriptBuildStep) {
      assertThat(it.scriptContent,
                 startsWith("#!/bin/bash"))
    }
  }
}
```

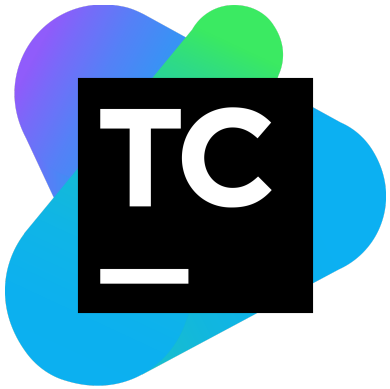

s-static :: [Develop] DeployCandidate
... process-static :: [Develop] E2E2
develop
nger-api :: [Develop] DeployCandidate
... messenger-api :: [Develop] ApiTest
develop
tification-service :: [Develop] DeployCandidate
... messenger-api :: [Develop] E2E2
develop
t-service :: [Develop] DeployCandidate
ЧУТЬ
СЛОЖНЕЕ
ervice :: [Develop] IntegrationTest
validation-service :: [Develop] DeployCa
storage-service :: [Develop] DeployCand
velop] DeployCandidate
management-service :: [Develop] Deploy
:: [Develop] DeployCandidate
... contact-service :: [Develop] E2E2
develop
metadata-service :: [Develop] DeployCandidate
s-service :: [Develop] DeployCandidate
... contact-service :: [Develop] ApiTest
develop
zation-service :: [Develop] DeployCandidate
... comments-static :: [Develop] DeployCandidate
... raven-validation-service :: [Develop] E2E2
develop

1. Build
2. Deploy to DEV
3. Test
4. Promote to QA
5. Deploy to QA
6. Promote to PROD
7. Deploy to PROD
8. Test

- Ctrl + C, Ctrl + V
- Библиотеки
- Тесты

НАШ ОПЫТ





2017.2

1.1

- Сервисы появляются как грибы
- Деплоим все, что есть, на каждый фича-бранч
- Ручки у разработчиков – каждый пишет, как хочет

- Фича-бранч – нужно чистое окружение – БД, очереди
- Много действий, чтобы добавить сервис
- Каждая команда хочет кастомизаций

ПЕРВЫЙ ПОДХОД К СНАРЯДУ

```
object MasterVCS: GitVcsRoot({
    name = "Master branch VCS root"
    url = "git@server:apps/contact-api.git"
    branch = "master"
    branchSpec = "+:refs/heads/*"
    authMethod = defaultPrivateKey {
    }
})
```

```
fun Project.vcs(repository: String) {
  vcsRoot(GitVcsRoot {
      id("Master".toId(this@Project.toString()))
      url = "git@server:$repository.git"
      // ...
  })
}

project {
  vcs("apps/contact-api")
}
```

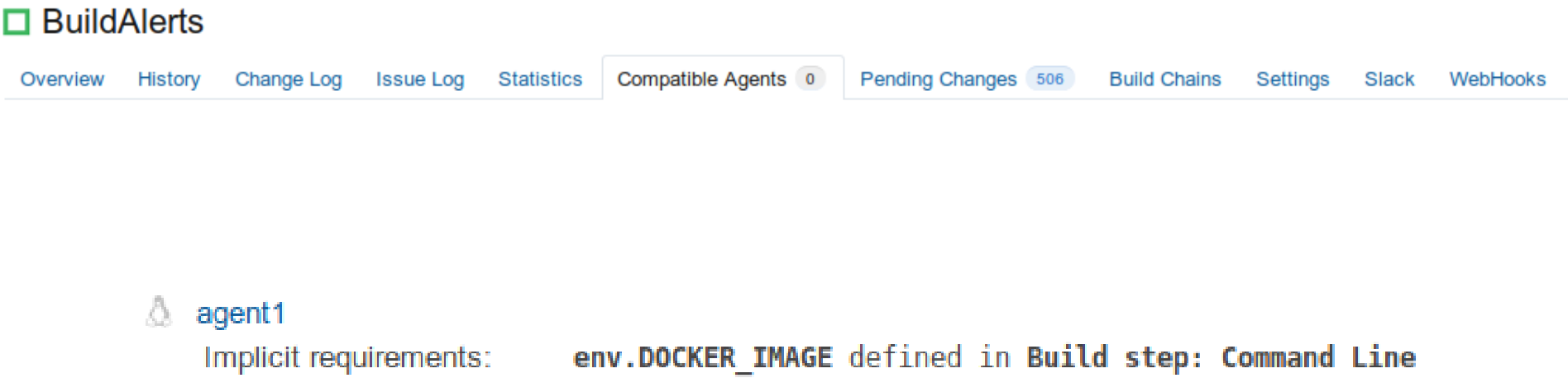
```
object PullRequestTrigger : VcsTrigger({
        triggerRules = """
                       +:**
                       -:comment=^Merge pull request #.*
                   """.trimIndent()
        branchFilter = """
                       +:*
                       -:
                       -:master
                   """.trimIndent()
    })
```

```
buildType {
  triggers {
    PullRequestTrigger()
  }
}
```

```
buildType {
  triggers {
    trigger(PullRequestTrigger())
  }
}
```

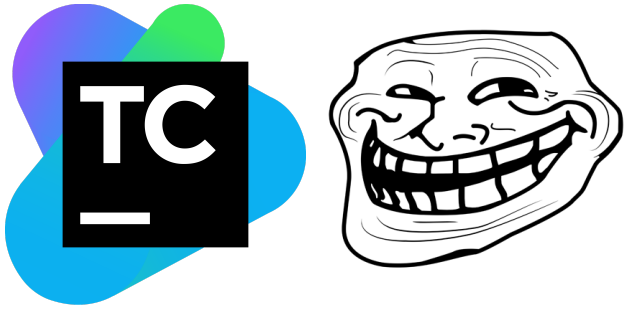
```
buildType {
    script {
        scriptContent = """
        IMAGE=%env.DOCKER_REGISTRY%/%env.DOCKER_IMAGE%

        docker run ${'$'}IMAGE
        """.trimIndent()
    }
}
```

```
mvn test
mvn teamcity-configs:generate
> Changes from VCS are applied to project settings
```

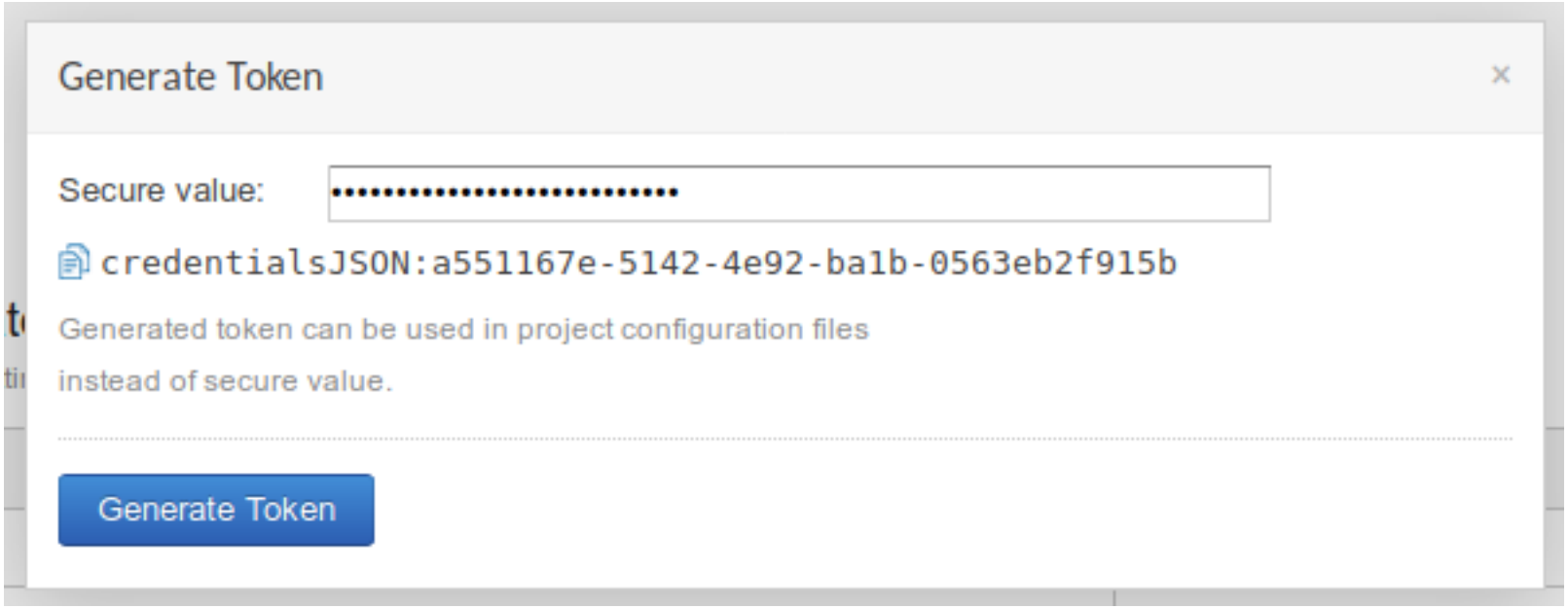

BuildAlerts
Overview
History
Change Log
Issue Log
Statistics
Compatible Agents 0
Pending Changes 506
Build Chains
Settings
Slack
WebHooks
agent1
Implicit requirements: env.DOCKER_IMAGE defined in Build step: Command Line

```
buildType {
  params {
    param(
      "env.DOCKER_IMAGE",
      "%dep.ContactBuild.env.DOCKER_IMAGE%"
    )
  }
}
```

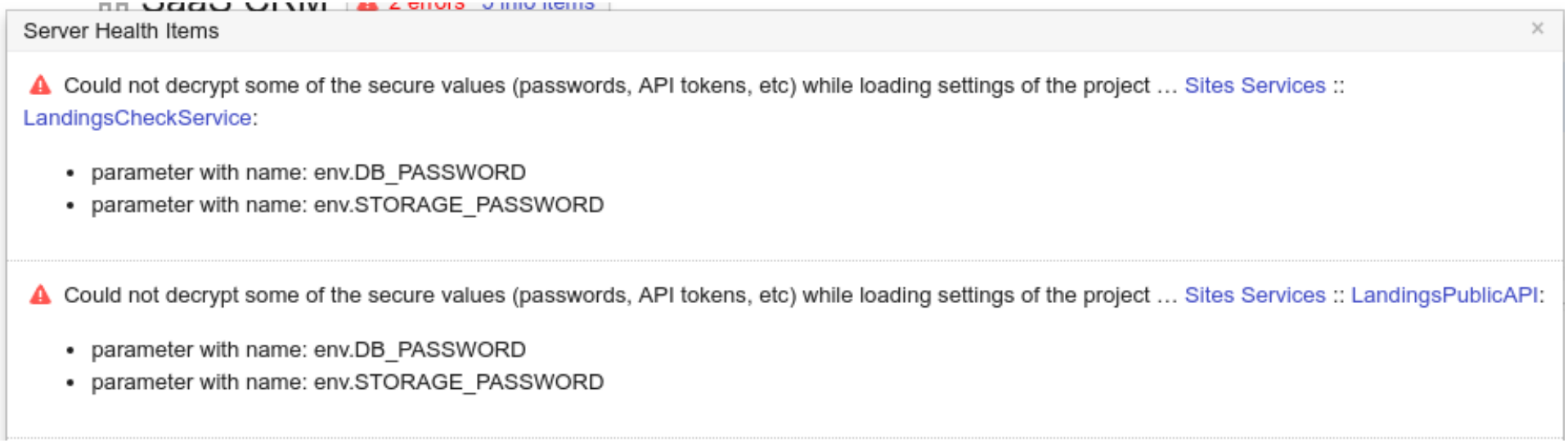
```
> +IMAGE="registry:5050/"
> docker: invalid reference format.
```



```
val vars = injectVars.joinToString("\n") {
  "echo \\\"##teamcity[setParameter name='$it' value='%$it
}

scriptContent = """
#!/bin/bash
echo "$vars" > inject_file.sh
""".trimIndent()
```




```
password(
  name = "env.DB_PASSWORD",
  value = "credentialsJSON:112e157f-76e8-4e7d-b12f-33f3510
)
```


Server Health Items
Could not decrypt some of the secure values (passwords, API tokens, etc) while loading settings of the project … Sites Services :: LandingsCheckService:
parameter with name: env.DB_PASSWORD
parameter with name: env.STORAGE_PASSWORD
Could not decrypt some of the secure values (passwords, API tokens, etc) while loading settings of the project … Sites Services :: LandingsPublicAPI:
parameter with name: env.DB_PASSWORD
parameter with name: env.STORAGE_PASSWORD

```
project {
  subProject {
    params {
      password("env.DB_PASSWORD", "credentialsJSON:...")
    }
  }
  subProject {
    // Nope
  }
}
```

```
project {
  params {
    password("env.DB_PASSWORD", "credentialsJSON:...")
  }
  subProject {
  }
  subProject {
    // OK
  }
}
```

```
project {
  params {
    password("env.DB_PASSWORD", "%vault:qa/db!password%")
  }
}
```


— ДАВАЙ КОЕ-ЧТО УЛУЧШИМ...

```
fun apiBuild(parent: Project, apiTests: Boolean,
             pr: Boolean) = BuildType({
  steps {
      if (pr) {
          getJiraTaskNumber()
      }

      if (apiTests) {
        e2eTest("")
      }
  }
})
```

ВТОРОЙ ПОДХОД К СНАРЯДУ

TeamCity	**Jenkins**
BuildStep	Step
BuildType	Stage
Project	Pipeline

Приложение
Сервис
Алерт

База данных
Окружения: DEV, QA, PROD

Джек:

Написал сервис, ему нужна база данных и еще бы миграции накатить

Thingy:

Билды настроил, базу создал, миграции накатил

- **Продукт**
  CRM, Конструктор Сайтов
- **Сервис**
  contact-api, crm-ui, notification-service
- **Ресурс**
  Postgres БД, Flyway-миграции; Kafka-топик; Swagger; Vault-секреты

```
product("CRM") {
  services {
    backend {
      name = "file-api"

      resources { /* ... */ }
    }
  }
}
```

```
flywayMigrations {
    db {
    name = "file_storage"
    schema = "files"
    }
}
```

```
swagger {
  name = "file-storage-service-swagger"
  dockerFile = "swagger-public.Dockerfile"
}
```

Product, Service :arrow_right: Project, BuildType

```
fun toProject(p: Product) = Project({
    id = AbsoluteId("${p.id}")
    p.services.each {
        buildType {
            // ...
        }
    }
})
```

```
backend {
  name = "contact-api"
  resources {
    postgres {
      name = "%env.CONTACT_API_DB_NAME%"
    }
  }
}
```


ContactApiBuild_%env.CONTACT_API_DB_NAME%
#88
Success

```
class Service(val name: String)
class Postgres(val dbName: String)
```

```kotlin
@DslMarker
annotation class TeamCityDslMarker

@TeamCityDslMarker
class Service(val name: String)

@TeamCityDslMarker
class Postgres(val dbName: String)
```

- Из одного DSL в другой – **сложно**
- Зависим от версии TeamCity
- Иногда приходится дебажить вслепую

- Лаконичный словарь
- Нельзя менять, можно расширять
- Описываем «что», а не «как»

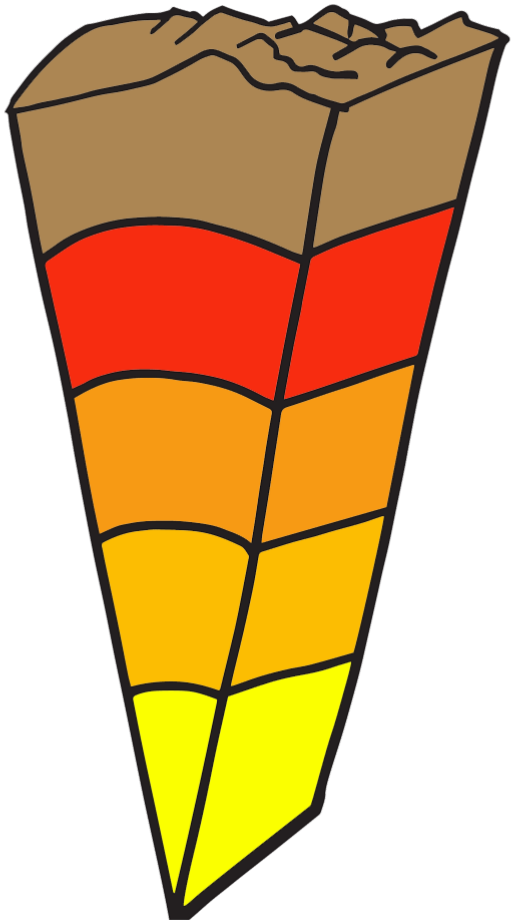
- **Было:**
  отдельная задача, до 5 дней
- **Стало:**
  разработчик сам делает PR с новым сервисом, 1 день

ТРЕТИЙ ПОДХОД К СНАРЯДУ

```
<dependency>
    <groupId>ru.tinkoff.teamcity</groupId>
    <artifactId>kotlin-dsl</artifactId>
    <version>${teamcity.dsl.version}</version>
    <scope>compile</scope>
</dependency>
```

- 2017.1
- 2017.2
- 2018.1

- Rancher DNS
- Consul
- Kubernetes



Custom DSL

Custom DSL Adapter

Helper Steps

TeamCity DSL

```
DSL.register {
  deploy {
    k8s()
  }

  teamcity {
    v2017_2()
  }

  product(CRM)
}
```

ИТОГО

- YAML – коварен
- DSL повсюду
- No silver bullet
- Не стоит прогибаться под ~~изменчивый мир~~ термины вашего CI

?

Андрей Ермаков, Tinkoff.ru

AntiYAML: DSL is the new black