

# Configuration as code: Teamcity. Первое знакомство

## Обо мне



test

Меня зовут Алексей.  
Последние 5 лет в ролях  
тестировщика-автоматизатора,  
фронтенд разработчика и  
управленца (тех.лид, тимлид).  
В рамках разных проектов, но  
одной крупной компании.  
От стажёра до ведущего  
инженера.

## Teamcity vs Другие CI

Исторически так сложилось, что Teamcity в отличии от TravisCI и ему подобных не позволял работать с конфигурацией без UI.

Но шло время, сборки становились всё более изощрёнными:

- шаблонные сборки
- мета-раннеры
- pipeline с реверсивными зависимостями

Начиная с Teamcity 10 версии появился программный способ настройки конфигурации.

DSL построенный на основе продвигаемого JetBrains языка Kotlin, а

## Форматы версионирования настроек Teamcity

Доступные на момент эксплуатации версии Teamcity 2021.2

Стоит отметить, что

1. для работы с настройками в формате xml возможно добавить расширенные возможности. Это например проект Terraform Provider TeamCity. Но как пишет сам автор в своём мотивационном посте проект был сделан за долго до полноценной реализации конфигурирования через Kotlin DSL.
2. сравнение возможно немного не честное, т.к. в случае с Kotlin DSL мы пишем инструкции, а в случае с XML готовый артефакт конфигурации. Более правильно было-бы сравнивать Terraform



## Kotlin DSL

Как будет выглядеть сборка в виде DSL можно посмотреть прямо в интерфейсе.

## Формат Kotlin DSL

**Build Steps**

In this section you can configure the sequence of build steps to be executed. Each build step is represented by a build name and provides integration with a specific build or test tool.

#	Name	Description	Build Tool
1	Recorder build steps	Recorder build steps	Auto detect build steps
2	Set parameters (inherited)	Parameters Description Maven Path to POM: c:\pom.xml Goal: clean Executor: If all previous steps finished successfully	Build Tool
3	Check target folder (inherited)	PowerShell PowerShell -city \$Bitness -script <script> Executor: If all previous steps finished successfully	Build Tool
4	Copy standard import setup (inherited)	Standard Import Setup Command Line Custom script: who Remove import files if Java project... (and I know less) Scripter: If all previous steps finished successfully	Build Tool
5	Configure import setup (inherited)	PowerShell PowerShell -city \$Bitness -script <script> Executor: If all previous steps finished successfully	Build Tool
6	Check targets and web folders exist, version is correct (inherited)	PowerShell PowerShell -city \$Bitness -script <script> Executor: If all previous steps finished successfully	Build Tool

```

1 # Maven
2 {
3   name = "maven"
4   id = "RUNNER_220"
5   goals = ["clean"]
6   pomLocation = "core/pom.xml"
7   runnerArgs = "-DskipTests=true -X"
8   userSettingsSelection = "artifactory-settings.xml"
9   localRepoScope = MavenMultiStep.RepositoryScope.HAVEN_DEFAULT
10  jdkHome = "$env:JDK_18_x64%"
11  param("org.jfrog.artifactory.selectedDeployableServer.defaultModuleVersionConfiguration",
12        "GLOBAL")
13  stepsOrder = arrayOf(
14    "RUNNER_220", "RUNNER_12366", "RUNNER_18091", "RUNNER_11982",
15    "RUNNER_11983", "RUNNER_12127", "RUNNER_33474", "RUNNER_33475", "RUNNER_148", "RUNNER_28965",
16    "RUNNER_33473", "RUNNER_18087", "RUNNER_34071", "RUNNER_6593")
17 }
  
```

## Структура версионизируемого проекта

```
.teamcity
[
--| pluginData
----| _Self
-----| metaRunners
-----| matarunner.xml
]
[
--| patches
----| entities
]
--| pom.xml
[--| Readme.md]
--| settings.kts
```

обязательные:

## Типовые сущности и связи сущностей в представлении Teamcity DSL


1. Шаблонная сборка
2. pipeline из нескольких сборок
3. метараннеры
4. параметры сборки в DSL



# Шаблонная сборка. Teamcity DSL vs UI

## Шаблонная сборка

```
1 object BuildBasedOnTemplate : BuildType({
2     // ссылка на шаблон
3     templates(TemplateSourceBuild)
4     name = "BuildBasedOnTemplate"
5     steps {
6         script {
7             scriptContent = """
8             echo another
9             echo another
10            """
11            trimIndent()
12        }
13    }
14    // ссылки на отключаемые ID настроек
15    disableSettings("RUNNER_36043")
16 })
17
18 object TemplateSourceBuild : Template({
19     name = "TemplateSourceBuild"
20
21     steps {
22         script {
23             id = "RUNNER_36043"
24             scriptContent = """
25             echo template
26             echo template
27             """
28             trimIndent()
29         }
30     }
31 })
32 // ...
```



CNC / Sandbox / shcherbakov-test / TemplateSourceBuild (detach)

Build steps

In this section you can configure the sequence of build steps to be executed. Each build step is represented by a build runner and provides integration with a specific build or test tool.

+ Add build step

Reorder build steps

Build Step	Parameters Description
1. Command Line (inherited, disabled)	Custom script: echo template (and 1 more line) Execute: if all previous steps finished successfully
2. Command Line	Custom script: echo another (and 1 more line) Execute: if all previous steps finished successfully



## Шаблонная сборка. Регистрация шаблона в DSL

### Шаблонная сборка

Регистрация шаблона

```
1 project {  
2     template(TemplateSourceBuild)  
3     // ...  
4 }
```

регистрация шаблона в settings.kts

<https://tech-billing.ru/projects/alexey-chcherbakov/repos/teamcity/>

## Шаблонная сборка. Конфигурация шаблона

### Шаблонная сборка

Конфигурация шаблона

```
1  object TemplateSourceBuild : Template({  
2      name = "TemplateSourceBuild"  
3  
4      steps {  
5          script {  
6              id = "RUNNER_36043"  
7              scriptContent = ""  
8                  echo template  
9                  echo template  
10                 ""'.trimIndent()  
11            }  
12        }  
13    })
```

nexign

10

имплементация

<https://tech-billing.ru/projects/alexey-shcherbakov/repos/teamcity/>

## Шаблонная сборка. Использование шаблона

### Шаблонная сборка

Сборка на основе шаблонной

```
1 object BuildBasedOnTemplate : BuildType({
2     // ссылка на шаблон
3     templates(TemplateSourceBuild)
4     name = "BuildBasedOnTemplate"
5     steps {
6         script {
7             scriptContent = ""
8             echo another
9             echo another
10            """.trimIndent()
11        }
12    }
13    // ссылки на отключённые ID настроек
14    disableSettings("RUNNER_36043")
15 })
```

## ИСПОЛЬЗОВАНИЕ

## Pipeline через зависимость артефактов



## Pipeline через зависимость артефактов

Регистрация

```
1 project {  
2     buildType(ChainFirstBuild)  
3     buildType(ChainSecondBuild)  
4     // ...  
5 }
```

## pipeline через зависимость артефактов. Конфигурация первой сборки

### Pipeline через зависимость артефактов

Конфигурация первой сборки

```
1 object ChainFirstBuild : BuildType ({
2     name = "ChainFirstBuild"
3     steps {
4         script {
5             scriptContent = ""
6             echo ChainFirstBuild => ChainFirstBuild.txt
7             """.trimIndent()
8         }
9     }
10    artifactRules = "ChainFirstBuild.txt"
11    requirements {
12        contains("system.agent.name", "RHEL7")
13    }
14 })
```

### Первая сборка

pipeline через зависимость артефактов. Конфигурация второй сборки

## Pipeline через зависимость артефактов

Конфигурация второй сборки

```
1  object ChainSecondBuild : BuildType ({
2      name = "ChainSecondBuild"
3      steps {
4          script {
5              scriptContent = ""
6              ls -a
7              """.trimIndent()
8          }
9      }
10     requirements {
11         contains("system.agent.name", "RHEL7")
12     }
13     dependencies {
14         artifacts(ChainFirstBuild) {
15             artifactRules = "ChainFirstBuild.txt"
16         }
17     }
18 })
```

nexign

использование зависимости по артефакту от первой сборки во

## метараннеры в структуре проекта

```
.teamcity
--| pluginData
----| _Self
-----| metaRunners
-----| matarunner.xml
```

metarunner.xml в файле settings.kts в project явно никак не регистрируется

регистрация в проекте происходит за счёт размещения xml файла  
.teamcity/pluginData/\_Self/metaRunners/<runner\_id>.xml

можно править существующий в проекте (добавленный из UI),  
но не создавать новый в проекте. (по крайней мере у меня не





## использование метараннера в DSL

### Метараннеры

#### использование

```
1 object BuildWithMetarunner : BuildType({
2     name = "Build with local metarunner"
3
4     steps {
5         step {
6             type = "Cnc_Sandbox_ShcherbakovTest_MetaRunner"
7         }
8     }
9 })
```

на строке 6 указан type = "<runner\_id>", T.e.

# метараннер код

## Метараннеры

код метараннера

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <meta-runner name="meta-runner">
3   <description>meta runner</description>
4   <settings>
5     <parameters>
6       <param name="Password" value="credentials:50N:cd2ceb46-ba73-4ce9-87d4-94a2f4d3b9fe" spec="password display="hidden" />
7       <param name="Username" value="cloud.ef.rable" spec="text validationMode="any" display="hidden" />
8       <param name="cnc_name" value="CNC" spec="text display="hidden" readOnly="true" validationMode="any" />
9       <param name="cnc_old_name" value="Cnc" spec="text display="hidden" readOnly="true" validationMode="any" />
10      <param name="last_version" value="7.13.0" spec="text display="hidden" validationMode="any" />
11      <param name="repo_dev" value="Dev" spec="text display="hidden" readOnly="true" validationMode="any" />
12      <param name="repo_ga" value="generic-ga-local" spec="text display="hidden" readOnly="true" validationMode="any" />
13      <param name="repo_release" value="generic-release-local" spec="text display="hidden" readOnly="true" validationMode="any" />
14      <param name="tech_stash.password" value="credentials:50N:97a602e-9890-4e02-85e9-e21e7c611348" spec="password display="hidden" readOnly="true" />
15      <param name="tech_stash.user" value="rnd_infl_cnc" spec="text display="hidden" readOnly="true" validationMode="any" />
16    </parameters>
17    <build-runners>
18      <runner name="" type="simplerunner">
19        <parameters>
20          <param name="org.jfrog.artifactory.selectedDeployableServer.downloadSpecSource" value="Job configuration" />
21          <param name="org.jfrog.artifactory.selectedDeployableServer.uploadSpecSource" value="Job configuration" />
22          <param name="org.jfrog.artifactory.selectedDeployableServer.useSpec" value="false" />
23          <param name="script.content" value="echo test" />
24          <param name="teancity.step.mode" value="default" />
25          <param name="use.custom.script" value="true" />
26        </parameters>
27      </runner>
28    </build-runners>
29    <requirements>
30      <contains id="BQ_14097" name="system.agent.name" value="BHL7" />
31    </requirements>
32  </settings>
33 </meta-runner>
34
35
```

# ИСХОДНЫЙ КОД



параметры сборки в DSL

Параметры сборки (участвующие в диалогах запуска)

```
1 params {
2   text("reverse.dep.*.version_previous", "7.12.0", label = "Преды
3   дущая версия", display =
4   ParameterDisplay.PROMPT,
5   regex = ""^([1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)$""", validationMessage = "Некорректный номер версии")
6
7   select("reverse.dep.*.project_name", "CNC
8   ", label = "Компонент", display = ParameterDisplay.PROMPT,
9   options = listOf("CNC", "CATALOG", "CORE", "CATALOG-UI
10  "))
11  text("reverse.dep.*.version_current", "7.13.0", label = "Текуща
12  я версия", display =
13  ParameterDisplay.PROMPT,
14  regex = ""^([1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)$""", validationMessage = "Некорректный номер версии")
15
16  param("reverse.dep.*.product_name", "CNC")
17 }
```

reverse.dep.*.product_name	CNC
reverse.dep.*.project_name	CNC
reverse.dep.*.version_current	7.13.0
reverse.dep.*.version_previous	7.12.0



КОД:

## Варианты использования версионирования настроек. UI first

1. создание или изменение чего-то из интерфейса пользователем
2. teamcity делает коммит от пользователя на изменение сборки в проекте

## Варианты использования версионирования настроек. git first

1. коммит на изменение сборки от пользователя
2. применение изменений в teamcity

## Варианты использования версионирования настроек. Mixed (Git + UI)

1. коммит на изменение сборки от пользователя
2. применение изменений в teamcity
3. создание или изменение чего-то из интерфейса пользователем
4. teamcity делает коммит от пользователя с патчем на изменение сборки в проекте
5. применение патча от code owner репозитория или от пользователя как инициатора когда-нибудь

Варианты использования версионирования настроек

изменение / подход	git first	ui first	mixed (git + ui)
ui			
code			

## Как придётся трансформировать работу по конфигурированию

### трансформация работы по конфигурированию CI



отправить в шредер бэкапы настроек  
сборок



разработка конфигурации – тоже  
разработка



немного изучить Kotlin



меньше коммуникаций

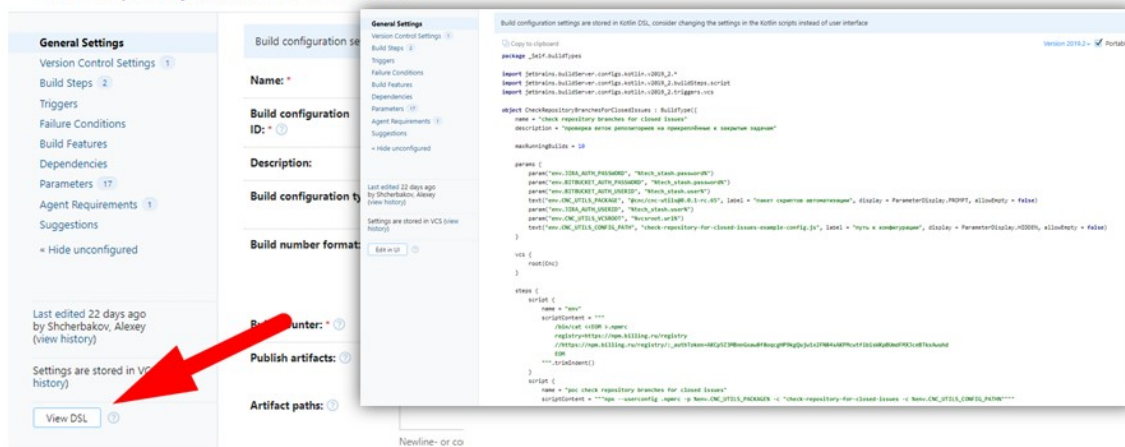
в варианте UI first практически никак.



## Как попробовать фицу представления настроек в DSL

## Как попробовать фичу представления сборок в виде DSL

- ☐ check repository branches for closed issues



23

**nexign**

Уже сейчас достаточно зайти в любую сборку и нажать на кнопку

## Как попробовать фичу версионирования настроек

1. Создать свой, не пустой репозиторий
2. Создать Teamcity проект/подпроект
3. Включить версионирование проекта
4. Дождаться initial commit от teamcity в гит

## Как работать с Kotlin DSL в IntelliJ IDEA

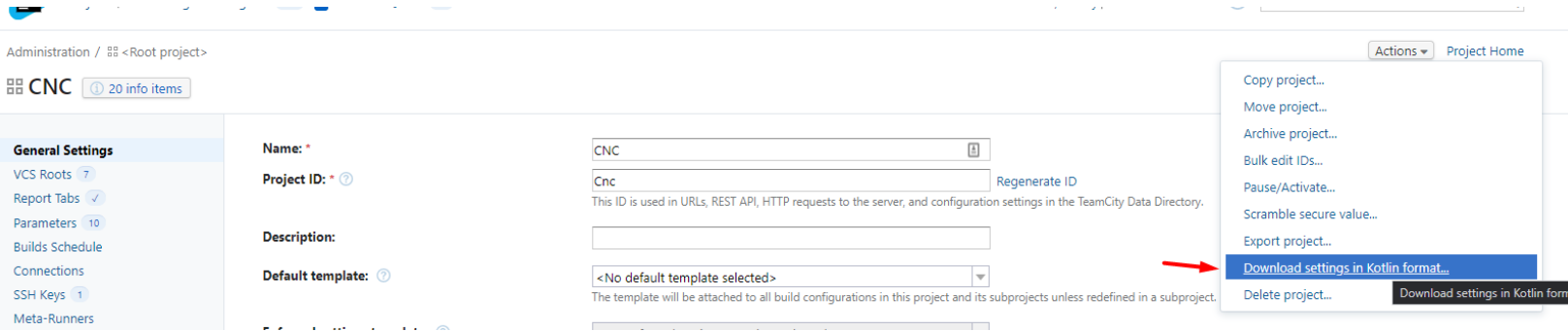
1. Включены плагины maven, kotlin
2. Добавить проект в maven (ПКМ на pom.xml)
3. Выполнить install

если у вас в организации используется локальное зеркало maven,  
то надо добавить исключение для репозитория dsl

делается это путём внесения изменений в ваш конфигурационный файл для maven. Для пользователей windows путь примерно следующий:

```
C:\Users\alexey.shcherbakov\.m2\settings.xml
```

# Начать переводить проект на версионирование настроек безопасно



скачать архив с настройками проекта через Actions > Download settings in Kotlin format

Начать переводить проект на версионирование настроек безопасно. Init проекта

уложить в репозиторий

```
1  ```
2  .teamcity
3  --| pom.xml
4  --| settings.kts
5  ```
```

поместить содержимое скаченного архива в папку .teamcity  
репозитория

**Важно!**

на всемя перехода чтобы не зааффектить существующие сборки рекомендуется выключить триггеры, билд фичи


regexр

# создать проект teamcity

## создать проект на основе версионизируемых настроек

Create Project

From a repository URL

 Manually

Parent project: \*

CNC / Sandbox / shcherbakov-test

Repository URL: \*

ssh://git@stash.billing.ru:2222/~alexey.shcherbakov/teamcity-dsl-first-look.git

A VCS repository URL. Supported formats: `http(s)://`, `svn://`, `git://`, etc. as well as URLs in Maven format.

Username:

Provide username if access to repository requires authentication.

Password:

Provide password if access to repository requires authentication.

Proceed

создать проект на основе версионизуемых настроек

создать проект на основе версионизуемых настроек

Administration / <Root project> / CNC / Sandbox / shcherbakov-test

Create Project From URL

✓ The connection to the VCS repository has been verified

⚠ The VCS repository contains `.teamcity/settings.kts` file with settings of some project. Would you like to import these settings?

☒ Import settings from `.teamcity/settings.kts`

☐ Do not import settings, create project from scratch

Project name: \* Teamcity Dsl First Look

VCS root: (Git) ssh://git@stash.billing.ru:2222/~alexey.shcherbakov/teamcity-dsl-first-look.git

Proceed Cancel

Teamcity увидит ваш файл settings.kts



## создать подпроект на основе версионизируемых настроек

Если вы создаёте подпроект в версионизируемом проекте (как в моём случае), то будет необходимо "отчудить" новый подпроект от основного

после создания проекта заходим в подпроект

- там будет стоять Use settings from a parent project
- выбрать Synchronization enabled
- далее импортируем настройки

# Применение настроек из VCS

импорт может занять продолжительное время в случае  
большого проекта

Teamcity Dsl First Look 2 info items

General Settings

VCS Roots 3

Report Tabs ✓

Parameters 10

Connections

Shared Resources

Allure Servers

WebHooks

SSH Keys

Meta-Runners

Maven Settings

Issue Trackers

Clean-up Rules

Versioned Settings ✓

Artifacts Storage

NuGet Feed

Suggestions

« Hide unconfigured

Project settings are stored in Kotlin DSL, consider changing the settings in the Kotlin scripts instead of user interface

Name: \*

Teamcity Dsl First Look ⓘ

Project ID: \* ⓘ

Cnc\_Sandbox\_ShcherbakovTest\_TeamcityDslFirstLook

This ID is used in URLs, REST API, HTTP requests to the server, and configuration settings in the TeamCity Data Directory.

Description:

Default template: ⓘ

<No default template selected> ▼

The template will be attached to all build configurations in this project and its subprojects unless redefined in a subproject.

Enforced settings template: ⓘ

<No enforced settings template selected> ▼

The enforced settings template imposes its settings on all build configurations in the project and its subprojects without ability to disable or redefine them.

✂ Hide advanced options

Save

Cancel

Build Configurations

Build configurations define how to retrieve and build sources of a project. ⓘ

+ Create build configuration

sorted alphabetically

Reorder

Name	Build Steps
------	-------------

1. Знакомство с конфигурацией
2. Откатов состояния конфигурации всего проекта до последнего рабочего
3. Анализ существующих сборок
4. Перенос сборки из одного проекта в другой методом копирования