

Kotlin all the things

So, after all, it seems JetBrains is very serious with Kotlin. And I have to admit it comes with some handy features and good IDE support. But this is not about the Kotlin language, this is about where it can be used.

As we have seen in [Migrating from Gradle to Gradle](#) writing Gradle build scripts using the new Kotlin DSL is supported. So far we have

- our sources in Kotlin
- our build configuration in Kotlin
- but not our Continuous Integration configuration (depending on how far you want to push it your build chain or pipeline as well)

Since TeamCity (the CI server) is from JetBrains as well, it supports storing your build configuration not only over the UI. It supports storing the configuration in your VCS in XML format and (since around version 10 and 2017 in a Kotlin format. The current version 2019.1 comes with even more improvements and simplifications in this section. So throw away your build config yaml file! Kotlin you build config too! Although this seems a bit weird at the beginning, there are some big advantages:

- it real source code
- it compiles
- you can share common pipeline definitions via libraries
- since Kotlin is a typed language, there is a nice support for in your IDE with auto-completion
- you can compile the code prior to pushing it - compared to the try and error cycle that YAML config files come with, I'll argue it is the better method

There is a very nice Blog from JetBrains on this topic that I can highly recommend to read:

- [Configuration as Code, Part 1: Getting Started with Kotlin DSL](#)
- [Configuration as Code, Part 2: Working with Kotlin Scripts](#)
- [Configuration as Code, Part 3: Creating Build Configurations Dynamically](#)
- [Configuration as Code, Part 4: Extending the TeamCity DSL](#)
- [Configuration as Code, Part 5: Using DSL extensions as a library](#)
- [Configuration as Code, Part 6: Testing Configuration Scripts](#)

And while you're at it, maybe watch the webinar on "[Turbocharging TeamCity with Octopus Deploy](#)" as well. [Octopus](#) is an additional commercial service. But distinguishing between continuous integration and deployment seems a good split in responsibilities.

Since I consider you know to be convinced let us test-drive it! We first need a TeamCity Server, a TeamCity Agent, and an example project.

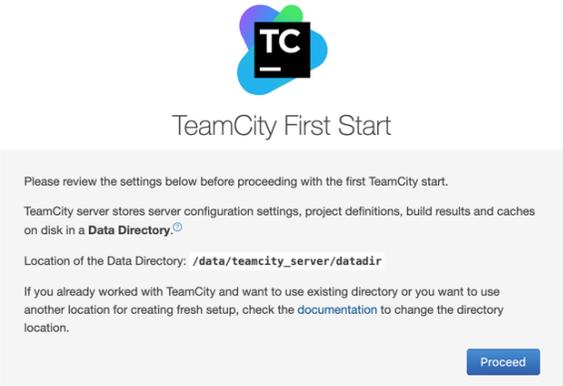
To have it quickly set up to test-drive, I created a docker-compose.yaml file (yes a yaml file, isn't it ironic): <https://github.com/brontofundus/kotlin-all-the-things>

Clone the repository and fire it up:

```
$ docker-compose pull
$ docker-compose up -d
```

Then point a browser to <http://localhost:8111/>

This will bring you to the TeamCity installer. But don't panic, it will only take a few minutes!

Proceed	 <p>TeamCity 2019.1 (build 65998)</p>
---------	---

- Select PostgreSQL
- download the driver
- and use the same settings as used in the docker-compose file:
 - "postgres" as host
 - "teamcity" as username, password and database name

Proceed

Wait



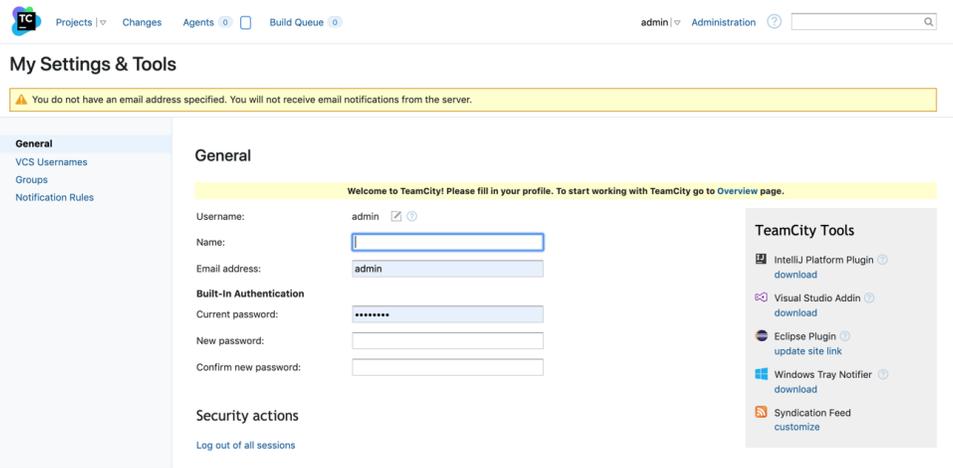
Scroll down and agree to the license (well read it of course - but don't tell me you are not used to selling your soul)

Create an admin account
(for simplicity use "admin" and "password" here)



The screenshot shows the 'Create Administrator Account' page. At the top is the TeamCity logo. Below it is the title 'Create Administrator Account'. The form contains three input fields: 'Username' (with a dropdown arrow), 'Password', and 'Confirm password'. A 'Create Account' button is positioned below the fields. Below the button is a link that says 'Login as Super user'. At the bottom of the page, the version information 'Version 2019.1 (build 65998)' is displayed.

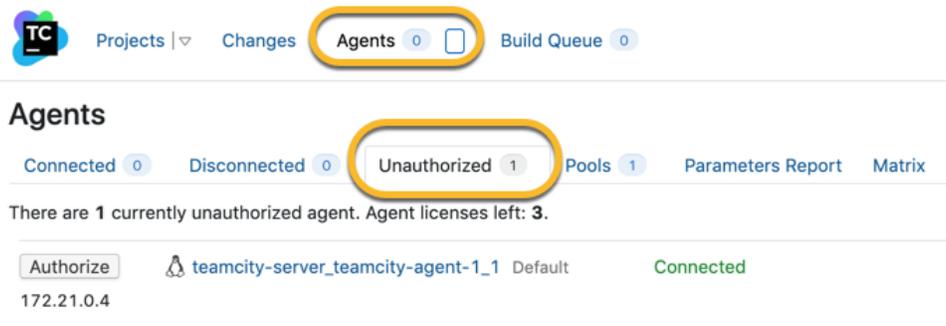
There you are



The screenshot shows the 'My Settings & Tools' page in TeamCity. The top navigation bar includes 'Projects', 'Changes', 'Agents' (with a count of 0), and 'Build Queue' (with a count of 0). The user is logged in as 'admin'. A yellow warning banner at the top states: 'You do not have an email address specified. You will not receive email notifications from the server.' The main content area is divided into sections: 'General' (with a sub-menu for 'VCS Usernames', 'Groups', and 'Notification Rules'), 'Built-In Authentication' (with fields for 'Current password', 'New password', and 'Confirm new password'), and 'Security actions' (with a 'Log out of all sessions' link). On the right side, there is a 'TeamCity Tools' section listing various plugins like 'IntelliJ Platform Plugin', 'Visual Studio Addin', 'Eclipse Plugin', 'Windows Tray Notifier', and 'Syndication Feed', each with a 'download' link.

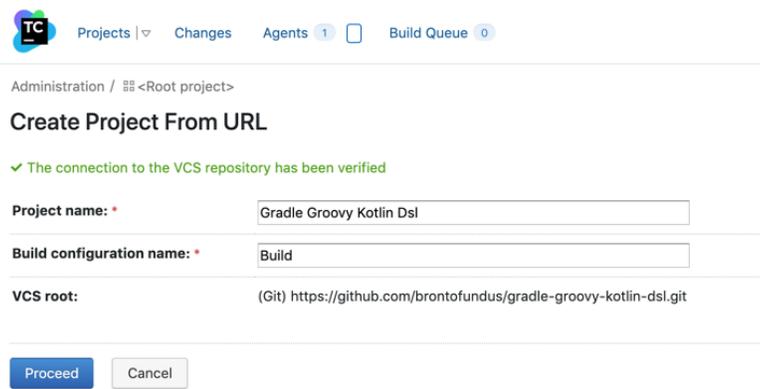
As you may notice on top there are 0 agents. Which is not entirely true.
But to have the one we have enabled it needs to be authorized first

Go to "Agents" and "Unauthorized" and enable the agent

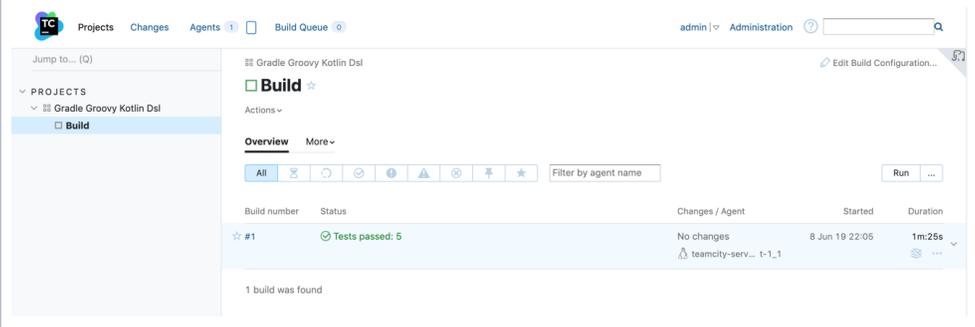


The screenshot shows the 'Agents' page. The top navigation bar is similar to the previous page, but the 'Agents' count is now 1. Below the navigation bar, there are tabs for 'Connected' (0), 'Disconnected' (0), and 'Unauthorized' (1). The 'Unauthorized' tab is selected and highlighted with a yellow circle. Below the tabs, there is a summary: 'There are 1 currently unauthorized agent. Agent licenses left: 3.' Below this, there is a table with one row showing an agent: 'teamcity-server_teamcity-agent-1_1' with the status 'Default' and 'Connected'. An 'Authorize' button is located to the left of the agent name. The IP address '172.21.0.4' is visible below the agent name.

If you now go to the start page (click on the logo on the top left) you will be able to add a project



Builds run already



If you have a close look at the project you will notice it contains a ".teamcity" directory that contains the build configuration: <https://github.com/brontofundus/gradle-groovy-kotlin-dsl/tree/kts/.teamcity>

The new 2019.1 format comes in a "portable" variant. So the number of files to earlier TeamCity version is reduced to only the settings.kts file and the pom.xml

From here on no more clicking in the UI is necessary.

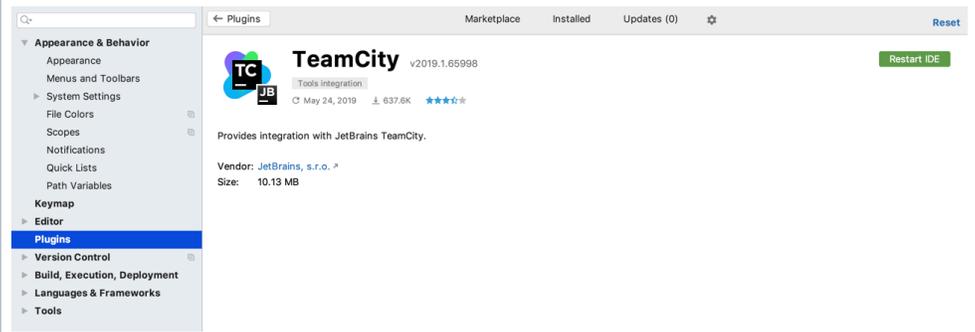
And even more comfort

Since you probably use IntelliJ to develop in Kotlin anyway and you now have a running TeamCity server from the same company, even more, comfort is possible!

Just install the TeamCity plugin in IntelliJ and point it via the new menu entry to your local server.

This will show the build status of your projects, but in addition, also allows you to run your local changes remotely as personal build! This is not a new feature but people tend to forget about it:

Install the plugin in IntelliJ

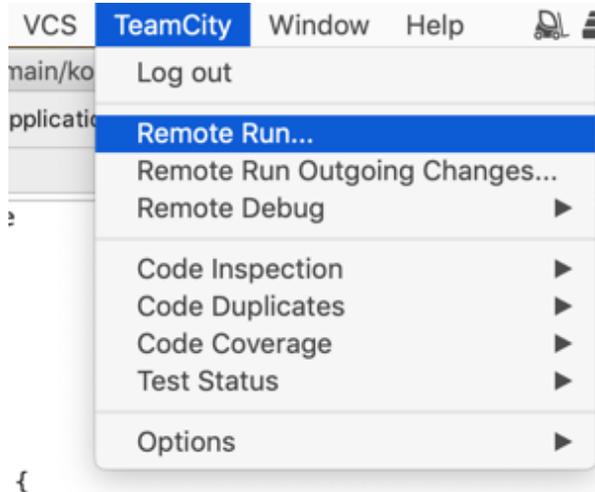


Restart and point it to the local TeamCity server
(we used "password" as password above)

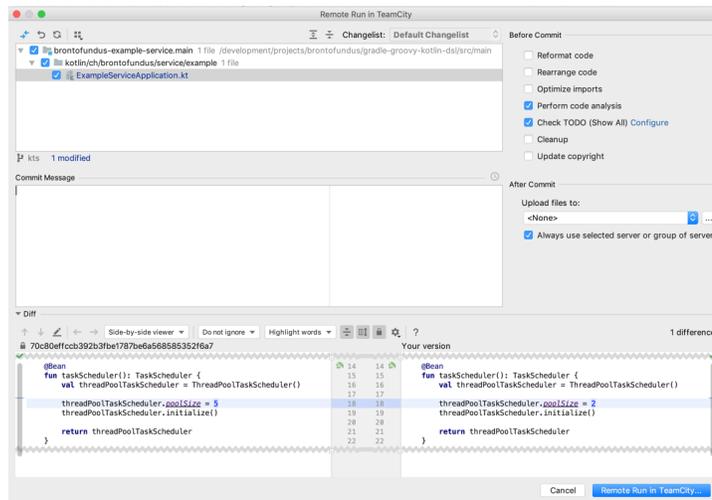
The screenshot shows a login dialog box for TeamCity. It has the following fields and options:

- URL:
- Username:
- Password:
- Use IntelliJ IDEA proxy settings
- Remember me
- Buttons: ? (help), Cancel, OK

And you can now remote run builds with local changes!



Even with not yet committed changes



Personal builds have this nice additional icon to mark personal builds. These are only visible to the user that created them

The screenshot shows the TeamCity interface for a build named 'Build' under the project 'Gradle Groovy Kotlin Dsl'. The build is marked as a personal build with a star icon. The status is 'Tests passed: 5'. A tooltip message reads: 'Your personal build was successful. Personal change: reduce scheduler pool size'. The build is assigned to agent 'admin: 1' on 'teamcity-serv... t-1_1'. A table below shows build history:

Build number	Status	Changes / Agent
☆ #4	Tests passed: 5	admin: 1 teamcity-serv... t-1_1
#3	Tests passed: 5	admin: 1 teamcity-serv... t-1_1

If you create an additional user and re-login you will not see other peoples personal builds

The screenshot shows the TeamCity interface from a different user's perspective. The 'Build Queue' tab is active, showing a list of builds. The 'Build' under 'Gradle Groovy Kotlin Dsl' is highlighted. The user dropdown menu is set to 'user'. The build queue shows:

Build number	Status	Changes / Agent
#3	Tests passed: 5	admin: 1 teamcity-serv... t-1_1
#2	Tests passed: 5; failed to load build settings from VCS (new)	3 changes teamcity-serv... t-1_1

So we have:

- our software in Kotlin
- our Gradle build script in Kotlin
- our CI configuration in Kotlin - able to share and re-use our build chains (read the JetBrains blog above for more details on this)
- and free of charge with this setup: remote runs

What a Kotlin world to live in!