Indore 2017.2 EAP3 (build 49972) Release Notes

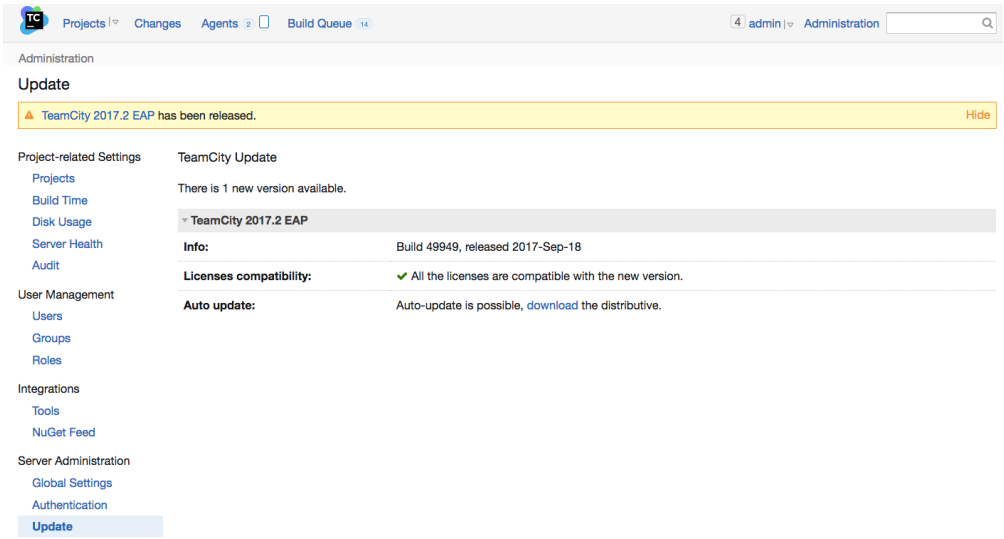
See also
Indore 2017.2 EAP2 (build 49708) Release Notes
Indore 2017.2 EAP1 (build 49391) Release Notes

On this page:

Server Auto Upgrade

Upgrading a TeamCity server is much simpler now. When a new version of TeamCity is detected, the server displays the corresponding health item for system administrators. This health item points to a new Update page in the server administration area, where all the upgrade options are listed. On this page there are notes about licenses compatibility, the new version description and controls to perform auto-upgrade.

If you're running the previous TeamCity EAP, you can wait till your server detects that new version is available and check the A dministration | Update page after that. To start upgrading to the new EAP build click the download link.



Current limitations (some of them will be addressed in future releases):

- auto-upgrade is disabled if the server is deployed from a .war or runs under the official TeamCity docker container
- the Windows uninstaller is not updated during the upgrade, so after several updates, during the uninstallation it may not delete all of the files
- the bundled java is not updated
- with several nodes installation, only the main TeamCity server can be auto-updated, the Running Builds node needs to be updated manually.

Kotlin DSL Improvements

In recent versions of TeamCity, when Kotlin DSL is enabled for a project, the administration part of the TeamCity user interface used to be read-only. Which is fine if you're familiar with Kotlin DSL, but not very user-friendly if you've just started working with Kotlin projects.

But we’re glad to announce that starting with this EAP when you switch a project to Kotlin DSL, project and build configuration settings will be available for editing from the web UI. All changes made in a project via the web user interface will be presented as a patch in the Kotlin format which will be checked in under the project “patches” directory. Moreover, if after switching to the Kotlin DSL you did not change the generated DSL files, UI changes will be applied in place, no patch will be generated at all. So if you ever had an interest in Kotlin DSL, but thought that sacrificing the user interface was too high a price for it, starting with this EAP this is no longer the case.

Note that TeamCity does not enable editing of the project right after the switch to Kotlin format. TeamCity needs to detect its own commit in the repository, and only after that editing will be enabled. Usually it takes a minute or two.

Switching to editable administration UI for existing Kotlin DSL projects

If you already have a project in Kotlin, then for web UI editing to be enabled for your project, the project should start using `v2017_2 API` of the Kotlin DSL. For this, your `.kt` files should be switched to packages from v2017_2 version. To compile this project, you also need to update your pom.xml with the following:

- change the teamcity.dsl.version to: `<teamcity.dsl.version>2017.2-eap3</teamcity.dsl.version>`
- kotlin version to: `<kotlin.version>1.1.4-3</kotlin.version>` and add a dependency on kotlin-runtime:

```
<dependency>
<groupId>org.jetbrains.kotlin</groupId>
<artifactId>kotlin-runtime</artifactId>
<version>${kotlin.version}</version>
<scope>compile</scope>
</dependency>
```

Alternatively, you can invoke the ‘Download settings in Kotlin format’ action in your project and copy the pom.xml from the generated zip archive.

Once you switch the project to new API and check in the changes, TeamCity will detect and apply them and after that web UI editing will be enabled.

Notes on patches

When settings are edited in the UI, TeamCity generates a patch script in a dedicated package ‘`<project external id>.patches.(buildTypes|templates|vcsRoots|projects)`’, script name is `<uuid of the entity>.kts`. The patch script is executed after regular dsl scripts and applies the UI changes to the generated settings if the settings are in the expected state. For example, if you change the build configuration name, then the patch scripts checks that the name produced by the regular script isn’t changed and then updates the name. Once the patch script is committed to the settings repository, you can apply its changes to your settings. After that, the patch script should be removed.

Example of a patch changing name of a build configuration

```
/*
This patch script was generated by TeamCity on settings change in UI.
To apply it, change the buildType with uuid '5dc8e147-11dc-4cb6-83aa-cfdb2595797d'
accordingly and delete the patch script.
*/
changeBuildType("5dc8e147-11dc-4cb6-83aa-cfdb2595797d") {
    check(name == "Old Name") {
        "Unexpected name: '$name'"
    }
    name = "New Name"
}
```

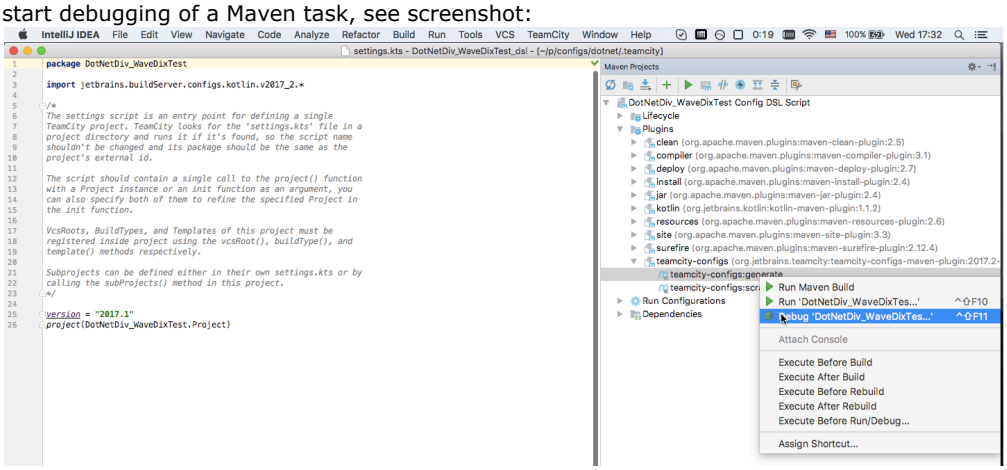
Limitations

Some of the UI actions will still be disabled for projects in Kotlin:

- external id change (for all entities - project, build configuration, template or VCS root)
- move actions
- editing of the versioned settings VCS root
- versioned Settings configuration change in a project

Other improvements

- The pom.xml file provided for a Kotlin project has the `'generate'` task, which can be used to generate TeamCity XML files locally from the Kotlin DSL files. This task now supports debugging. If you're using IntelliJ IDEA, you can easily start debugging of a Maven task, see screenshot:



- Kotlin DSL documentation is now generated in the background reducing the server startup time.

Default Templates

It is now possible to define a default template in a project. If defined, it affects build configurations in all the subtree of this project unless other default templates are defined in subprojects. With default templates it is much easier now to add a specific build feature to all build configurations of a project, or switch all build configurations to some specific checkout mode, or provide a default failure condition.

Composite Builds Improvements

A composite build now supports artifact dependencies. A composite build is not executed on an agent, so artifact dependencies work like a virtual view on artifacts from dependencies. For instance, if there is a build configuration "Dist", publishing an artifact "my_product.exe", and also dependent composite build configuration "Composite", which has an artifact dependency on Dist: my_product.exe, then, once a build of "Dist" publishes my_product.exe, this file will be shown as an artifact of dependent "Composite" build as if it was published there. In reality, though, this is just a view, and artifact is still in the build of "Dist".

Note: exclude rules and different target paths are not supported yet, but we are planning to add support for them in the next EAP build.

Several other improvements in snapshot dependencies:

- now the Builds tab on the Change details page also includes the builds with the "show changes from dependencies" option enabled and affected by the change. The problems from such builds are also listed on the Problems and tests tab.
- email notifications for builds with "Show changes from dependencies" option enabled now list changes from dependencies
- for composite builds, option to limit number of running builds actually limits the number of build chains, so another build chain will not start if maximum number of build chains reached

Perforce Stream Depth Support

Previously the Perforce VCS Roots in TeamCity only supported streams stored one level below the depot name, i.e. the default stream depth of 1: //myStreamDepot/myStream1). Starting from this EAP, TeamCity supports deeper directory structure within the root depot: depots with a depth of //DEPOTNAME/1/2/n can be specified in the Perforce "Stream" field.
See the related request in our tracker.

Docker Support Improvements

- TeamCity docker integration now supports docker on Windows. For instance, it is now possible to use Docker Wrapper o n windows agents if docker is installed there.

- Now TeamCity reports the Docker server host operating system via the `docker.server.version` and `docker.server.osType` configuration parameters on all platforms including Windows.
- You can now opt to perform the "`docker pull`" command before starting a build step with the Docker Wrapper.

Cloud Agents

- Manual termination: when using cloud agents, you can now choose to stop the instance after current build using the corresponding new option on the agent page - the agent will terminate gracefully. It is also possible to force instance termination if required.
- Now in a number of places in the UI TeamCity provides human readable agent name instead of the image name.

Other Improvements

- The Command Line Runner can now log stderr output as error messages. We've added the Format stderr output as opti on with warnings/errors selector (warnings is the default value, as before)
- REST API now allows tests and problems mutes management via .../app/rest/mutes endpoint
- All fixed issues