Kanpur 2019.1 RC (build 65943) Release Notes

See also:

Kanpur 2019.1 EAP3 (build 65740) Release Notes
Kanpur 2019.1 EAP2 (build 65579) Release Notes
Kanpur 2019.1 EAP1 (build 65427) Release Notes

On this page:

Using snapshot dependencies without synchronizing revisions

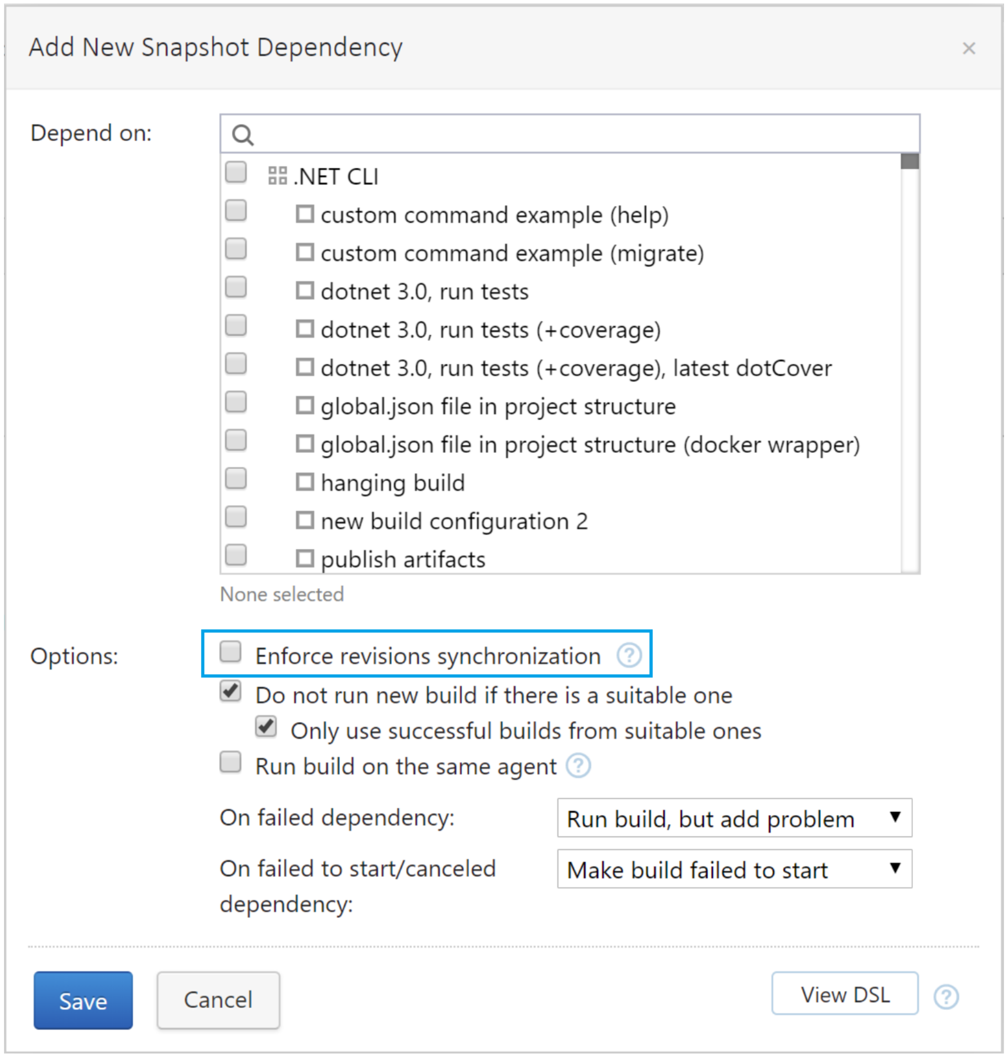
In a TeamCity build chain, all builds linked via snapshot dependencies use the synchronized revision of the source code, which means that the revision of the sources for all builds is the same or taken at the same time (if the VCS roots are different). So if you trigger a dependent build, but there are still pending changes in any of the dependencies (preceding builds in the chain), they will have to be built first for the code revision to synchronize, and only then the dependent build will run.

But in some cases, you may want to run the dependent build without synchronizing its code revision with the preceding builds in the chain.

This can be useful, for example, in deployment build configurations, when you do not want to run the whole chain if pending changes are available, but just deploy artifacts of one of the builds.

In load/acceptance testing, when you store tests in a version control system and often change them to test your system, you do not need to rebuild your application entirely, but just pick up the chain from the testing phase.

In this version we’ve partially addressed these cases: you can now disable revision synchronization in a snapshot dependency when promoting a build. If you uncheck the 'Enforce revision synchronization' box in the Add/Edit Snapshot Dependency dialog ue, you will be able to promote a build (manually or via a trigger) in a dependency chain, and TeamCity will put the target build into the queue without synchronizing its code revision with other build configurations of the chain.


Add New Snapshot Dependency

Depend on:

- [ ] .NET CLI
- [ ] custom command example (help)
- [ ] custom command example (migrate)
- [ ] dotnet 3.0, run tests
- [ ] dotnet 3.0, run tests (+coverage)
- [ ] dotnet 3.0, run tests (+coverage), latest dotCover
- [ ] global.json file in project structure
- [ ] global.json file in project structure (docker wrapper)
- [ ] hanging build
- [ ] new build configuration 2
- [ ] publish artifacts

None selected

Options:

- [ ] Enforce revisions synchronization
- [x] Do not run new build if there is a suitable one
  - [x] Only use successful builds from suitable ones
- [ ] Run build on the same agent

On failed dependency: Run build, but add problem

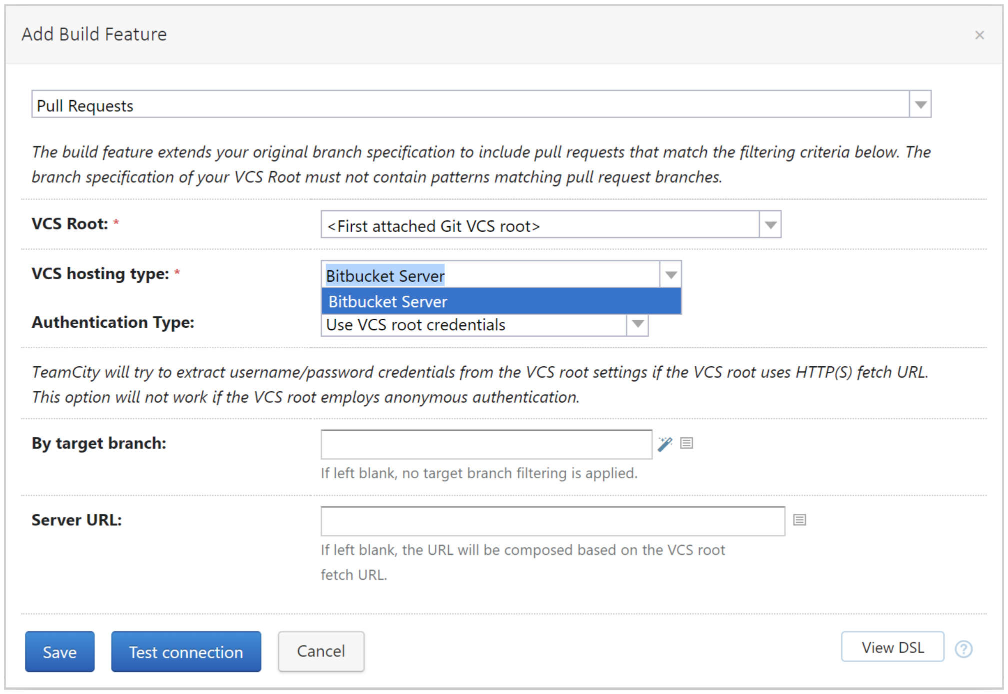
On failed to start/canceled dependency: Make build failed to start

Support of Bitbucket Server pull requests

Since TeamCity 2019.1, the Pull Requests build feature can detect pull requests created in Bitbucket Server.

To add a Bitbucket VCS root, select ‘Bitbucket Server’ as ‘VSC hosting type’ and configure the connection parameters:

- Authentication type: VCS root credentials or username/password
- Filtration by target PR branches
- Bitbucket Server’s base URL


Add Build Feature

Pull Requests

The build feature extends your original branch specification to include pull requests that match the filtering criteria below. The branch specification of your VCS Root must not contain patterns matching pull request branches.

VCS Root: * <First attached Git VCS root>

VCS hosting type: * Bitbucket Server

Authentication Type: Use VCS root credentials

TeamCity will try to extract username/password credentials from the VCS root settings if the VCS root uses HTTP(S) fetch URL. This option will not work if the VCS root employs anonymous authentication.

By target branch:

If left blank, no target branch filtering is applied.

Server URL:

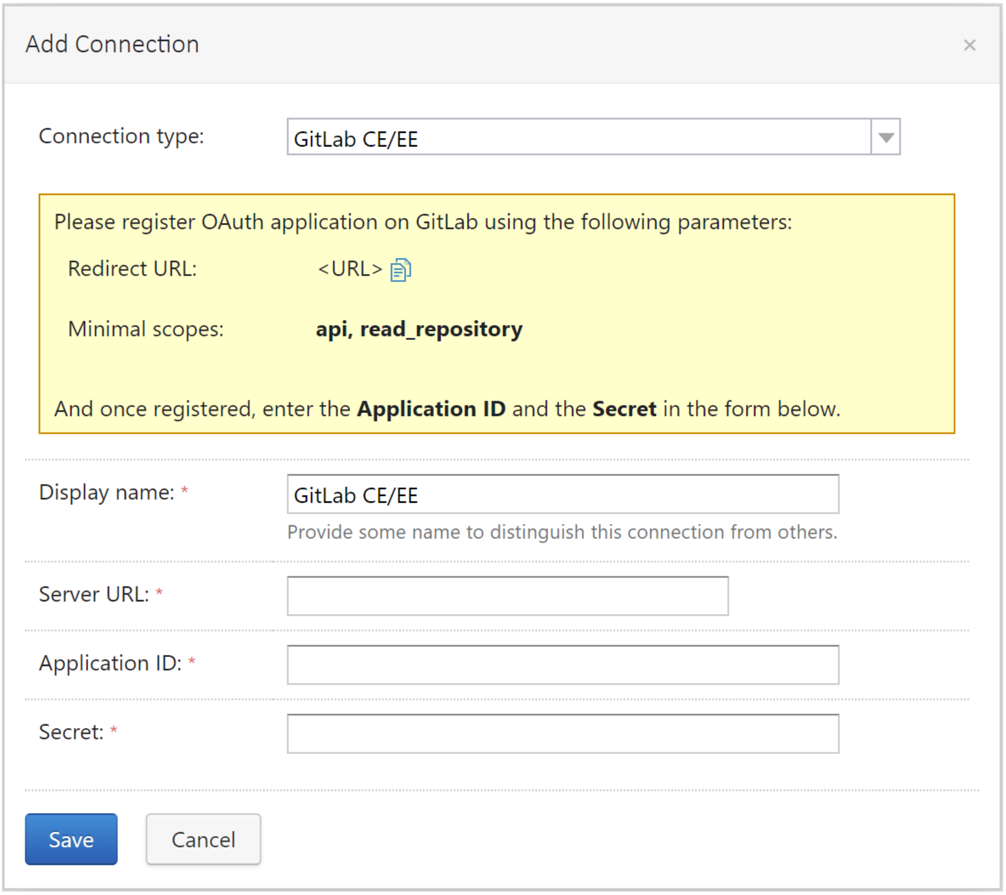
If left blank, the URL will be composed based on the VCS root fetch URL.

GitLab connections

TeamCity now allows creating connections to GitLab.com and GitLab CE/EE so you could easily select a predefined GitLab repository when creating a new project or a build configuration.

To be able to authenticate to GitLab during the connection, register an OAuth application in GitLab with the `api` and `read_repository` scopes and generate a secret and an application ID.

Enter the secret and application ID, along with the GitLab server URL, when adding a new connection.


Add Connection

Connection type:	GitLab CE/EE

Please register OAuth application on GitLab using the following parameters:

Redirect URL:	<URL>
Minimal scopes:	**api, read_repository**

And once registered, enter the **Application ID** and the **Secret** in the form below.

Display name: *	GitLab CE/EE Provide some name to distinguish this connection from others.
Server URL: *	
Application ID: *	
Secret: *	

Faster agents upgrade

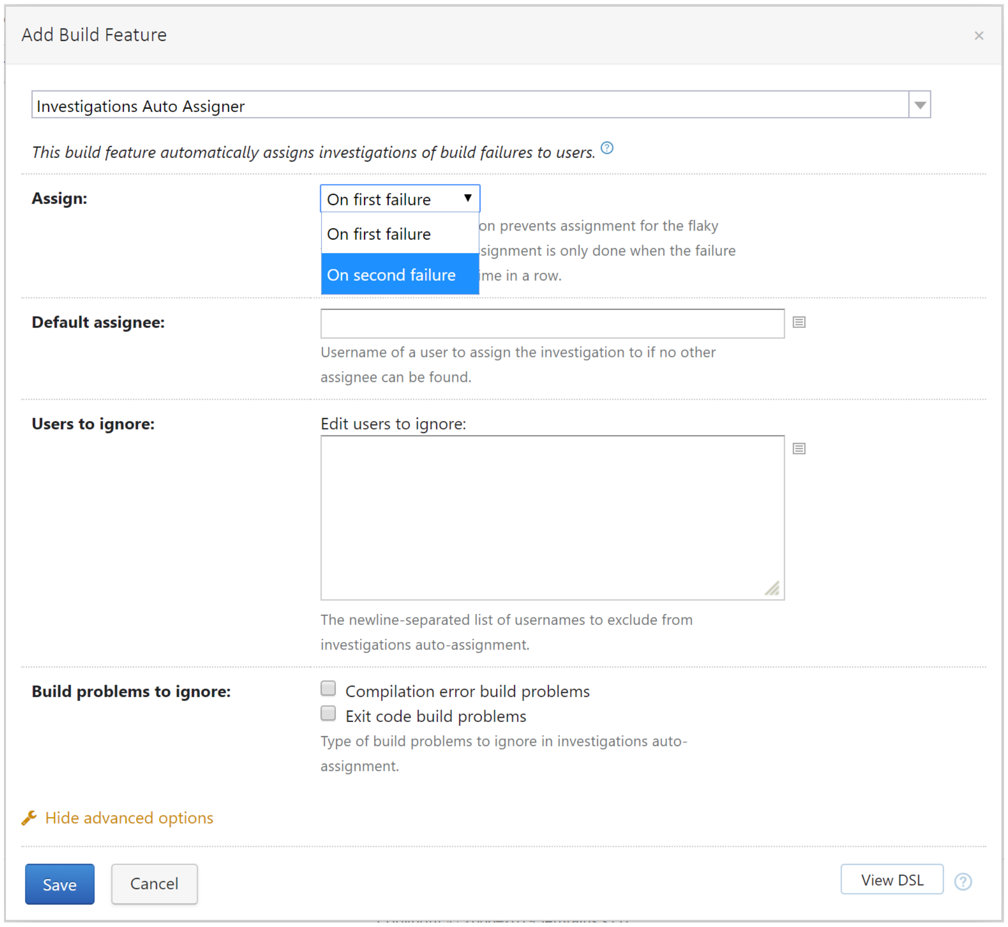
Previously, build agents downloaded all available tools from the server right after the server upgrade. In setups with hundreds of agents, this could significantly load and slow down the server. To reduce the agents' requests impact on the server and speed up the agents' upgrade, we have changed this behavior. Agents now download tools from the server only when starting the first build that requests these tools. The downloaded tools are stored on agents so builds don't spend time on downloading them again.

Delayed auto-assignment of investigations for flaky tests

The Investigation Auto Assigner build feature doesn't assign investigations to users when builds fail due to flaky tests. TeamCity needs time to detect a flaky test after the build finish, and in some cases the Auto Assigner can assign an investigation to a user even if the build fails without any user involvement. When there are many flaky tests in a project, this may be distracting.

To prevent this scenario, TeamCity now can delay the investigation assignment until the test fails repeatedly. This allows covering most situations with failures caused by flaky tests.

You can now choose whether to assign an investigation on the first or second build failure in the Investigation Auto Assigner build feature:


Add Build Feature

Investigations Auto Assigner

This build feature automatically assigns investigations of build failures to users.

Assign: On first failure
On first failure
On second failure
on prevents assignment for the flaky
signment is only done when the failure
me in a row.

Default assignee:
Username of a user to assign the investigation to if no other assignee can be found.

Users to ignore: Edit users to ignore:
The newline-separated list of usernames to exclude from investigations auto-assignment.

Build problems to ignore:
- [ ] Compilation error build problems
- [ ] Exit code build problems

Type of build problems to ignore in investigations auto-assignment.

If you select 'On second failure', the Assigner will wait for the second sequential failure of a test in the current build configuration (in the default branch) before assigning investigations to a user.

Separate Maven artifact repository for all builds on agent

We have changed the local artifact repository settings selection for Maven and added an option to create a separate repository for all builds run by an agent.

Here is what changed:

Old option	New option	Description
-	Per agent (default)	Use a separate repository to store artifacts, produced by all builds run by an agent, under the agent system directory.
Enabled “Use own local repository for this build configuration”	Per build configuration	Use a separate repository to store artifacts, produced by all builds of the current build configuration.
Disabled “Use own local repository for this build configuration”	Maven default	Use the default Maven repository location. The repository is shared between all build configurations and all agents on the machine.

Branch filter for VCS of build configuration

We have added a branch filter to version control settings of a build configuration, similarly to filters in build triggers or test details. In the previous versions of TeamCity, VCS settings allowed disabling builds in the default branch only, but the branch filter offers a more flexible approach. To filter branches, use the syntax described in Configuring branches.

The VCS branch filter is applied before any other branch filter and limits branches shown in the custom build dialog, branches visible to triggers, and changes from snapshot dependencies.

Cached configuration files on nodes

In a multinode setup, the TeamCity data directory is shared among secondary nodes via network. When a secondary node launches, it reads configuration files located in the shared data directory. In large setups with thousands of projects and build configurations, downloading lots of files from network storage can take a lot of time.

In TeamCity 2019.1, we have optimized this operation by creating a cache of configuration files stored under the node local data directory. The files are cached on the first node launch and then updated in runtime which makes the next launches faster.

Other improvements

- Storage of VCS-related data has been optimized to improve the server performance during startup.
- Access tokens now can be used for authentication to the TeamCity server instead of passwords.

Known issues

- If you used access tokens in TeamCity Kanpur 2019.1 EAP3, note that they will be deleted from the database and stop working on upgrading to 2019.1 RC. Make sure to generate new access tokens after upgrading.
- When you use .NET CLI for running a build that tries to pass credentials via some NuGet command, you may receive one of the following errors.
    - If .NET Core SDK 2.x is installed but its version is earlier than 2.1.400:
      "Could not load file or assembly 'System.Runtime, Version=<version>, Culture=neutral, PublicKeyToken=<key> or one of its dependencies. The system cannot find the file specified."
    - If only .NET Core SDK 3.0 is installed on your server:
      "A plugin was not found at path <path> if only dotnet version 3.0 is installed on an agent."

  To fix any of these problems, we recommend installing .NET Core SDK 1.x and/or 2.1.400 or later additionally to your current SDK version.