# How to Build Applications with TeamCity

*On how to build applications with TeamCity.*

## Introduction

TeamCity is used for compiling and building projects into NuGet packages. Octopus Deploy is then used for deploying these NuGet packages to various environments such as test, UAT, client test, partner test and production.

## Projects Page

Login to TeamCity. A login can be obtained by contacting other developers in the team. The ‘Getting started with TeamCity’ page will be the first visible page with information about the number of projects and build configurations.

From this page, the relevant project should be chosen by clicking on the small down arrow next to the ‘Projects’ link at the top left. Once the list of main projects is displayed, the relevant main project should be selected by clicking on it, e.g. ‘xxx’.

The TeamCity main projects correspond to the various development areas within Illion, e.g.:

xxx
xxx
xxx

Once one of the main projects is selected, it will navigate to the page for that main project and the title of the main project will be displayed on the top of the page. This page will display areas of development within this main project.

Various Visual Studio solutions will be listed under these areas. For example, under the ‘xxx’ main project, the following areas of development are listed with the description of each on the right:

xxx
xxx
xxx

These areas can be expanded by clicking on the small right arrows next to them. Expanding them will show the build projects that are listed under them, each usually corresponding to one or more Visual Studio solutions.

For example, under ‘xxx’, the ‘xxx’ build project refers to the following solution in Stash:

xxx.sln

Under ‘xxx’, the entry ‘xxx’ build project refers to the following solution in Stash:

xxx.sln

However, a single build project can refer to and build multiple solutions.

Clicking on the small right arrow next to ‘xxx’ will expand the entry and another entry ‘xxx’ will be shown. Clicking on this link will open the page related to this build project. Similarly, clicking on the small right arrow next to the ‘xxx’ entry will expand that entry and another entry ‘xxx’ will be shown. Clicking on this link will open the page related to this build project.

# Build Project Page

The build project page is divided into tabs such as Branches, Overview, History, etc. If there are any pending changes, then the ‘Branches’ tab will be displayed by default, otherwise the ‘Overview’ tab will be displayed first.

The ‘Overview’ tab will show the master and other branches such as feature branches related to this project. Production code is in the master branch and development is done in feature branches. The master branch will be shown as:

refs/heads/master

Feature branches will be shown by their names, e.g.:

feature/xxx

There will be build numbers to the left of branch names showing the builds of that branch that have been done so far, with the top most number being the latest build number for that branch, e.g. #397.

The ‘Branches’ tab will show just the active branches for the project. Active branches are branches with pending changes. This means that Stash commits have been done to these branches that have not been built in TeamCity yet.

To the right of the branch name entries, there will be links with the text ‘Pending (xx)’, showing the number of new commits that have been done in Stash, not yet built in TeamCity, with the xx representing the exact number of new changes or commits in Stash.

Clicking on the small down arrow next to this link will show the new Stash commits that are pending along with information on the files that were changed to the right with a link with the text ‘xx Files’ where xx denotes the number of files that were changed in each of the new commits.

Clicking on the small down arrow next to this link will list the actual files that were changed and clicking on these file path links will show the code changes that were made to these files.

There will be a build number below the branch name showing the last successful build of that branch, e.g. #396. There will be a link to the right of this build number with the text ‘Success’. Clicking on the small down arrow next to this link will show a menu with entries such as ‘Build log’, ‘Parameters’ and ‘NuGet Packages’.

The ‘Build log’ will show the log of that build, ‘Parameters’ will show TeamCity parameters and other configuration information for that build and ‘NuGet Packages’ will show the NuGet package related information for that build.

# Building a Project Branch

The master and feature branches that have pending changes can be built from the ‘Branches’ tab. Branches with pending changes will have a ‘Run …’ button to the right of them. Click on the ‘Run …’ button to build the pending changes shown. When this is done, it will create a new build number below the branch entry (e.g. #397) and start building that branch.

While it is building it will show a ‘Running’ link to the right of the build number as well as an estimate of how long the build would take to complete further to the right. Clicking on the small down arrow next to this link will show a menu with entries such as ‘Build log’ and ‘Parameters’. The ‘Build log’ will show the log of that build and ‘Parameters’ will show TeamCity parameters and other configuration information for that build.

The build will produce NuGet package for the project with the same build number as above, that is usable in the various environments such as test, UAT, client test, partner test and production. This is done and made available to Octopus Deploy, which is what is used for the actual deployment of the package to the various environments.

Once the build finishes, the ‘Running’ link changes to ‘Success’, which means that the build completed successfully. Now this build will appear in Octopus Deploy with the same build number 1.0.**397**.0.

# Build Log

On examining the build log for the ‘xxx’ build project in ‘xxx’, it becomes clear that TeamCity builds multiple solutions for it:

```
...
Step 5/17: Build Solution - xxx (Visual Studio (sln)) (19s)
Step 6/17: Build Solution - xxx (Visual Studio (sln)) (11s)
Step 7/17: Build Solution - xxx (Visual Studio (sln)) (14s)
...
```

TeamCity then creates NuGet packages from the built solutions:

```
...
Step 8/17: Create NuGet Packages (NuGet Pack) (3s)
   [Step 8/17] pack: Create NuGet package from xxx.nuspec (1s)
   [Step 8/17] pack: Create NuGet package from xxx.nuspec (1s)
   [Step 8/17] pack: Create NuGet package from xxx.nuspec (1s)
...
```

TeamCity then deploys the built NuGet packages to Octopus Deploy:

```
...
Step 10/17: Create xxx Octopus release and deploy (OctopusDeploy: Create release) (32s)
   [Step 10/17] Octopus Deploy (31s)
      [Octopus Deploy] Running command: octo.exe create-release --server https://xxx/ --
apikey xxx --project xxx --enableservicemessages --version 1.0.397.0 --deployto xxx --progress --
packageversion 1.0.397.0
      [Octopus Deploy] Creating Octopus Deploy release
      [Octopus Deploy] Octopus Deploy Command Line Tool, version 2.6.3.60
      [Octopus Deploy] Handshaking with Octopus server: https://xxx/
      [Octopus Deploy] Handshake successful. Octopus version: 2018.3.13; API version: 3.0.0
      [Octopus Deploy] Finding project: xxx
      [Octopus Deploy] Finding deployment process for project: xxx
      [Octopus Deploy] Finding release template...
      [Octopus Deploy] Using version number provided on command-line.
      [Octopus Deploy] Release plan for release: 1.0.397.0
      [Octopus Deploy] Steps:
      [Octopus Deploy] # Name Version Source
      [Octopus Deploy] --- ---------------------------- -------------- ----------------------------------
      [Octopus Deploy] 1 Copy xxx NuGet package 1.0.397.0 User specified
      [Octopus Deploy] Creating release...
      [Octopus Deploy] Release 1.0.397.0 created successfully!
      [Octopus Deploy] Deploying xxx 1.0.397.0 to: xxx (Guided Failure: Not Enabled)
      [Octopus Deploy] Waiting for 1 deployment(s) to complete....
      [Octopus Deploy] Deploy xxx release 1.0.397.0 to xxx: Success
      [Octopus Deploy] Done!
      [Octopus Deploy] Octo.exe exit code: 0
...
Step 13/17: Create xxx Octopus release and deploy (OctopusDeploy: Create release) (31s)
...
[Octopus Deploy] # Name Version Source
[Octopus Deploy] --- ---------------------------- -------------- ----------------------------------
[Octopus Deploy] 1 Deploy xxx NuGet package 1.0.397.0 User specified
```

```
...
Step 16/17: Create xxx release and deploy (OctopusDeploy: Create release) (21s)
...
[Octopus Deploy] # Name Version Source
[Octopus Deploy] --- ------------------------------ -------------- ----------------------------------
[Octopus Deploy] 1 Deploy xxx NuGet package 1.0.397.0 User specified
...
```

Notice how the NuGet packages have the same version and build numbers.

After these build and deployment steps finish, the Success link is shown. Now the deployed NuGet packages can be deployed to the various environments such as test, UAT, client test, partner test and production using Octopus Deploy.

# Meeting Recording

The was a meeting held on the 03rd of June 2022 and this document refers to that meeting. The recording of this meeting is available at this network location:

xxx

It is in the folder named "2022-06-03 CCB NZ B2B Web Service Build, Deployment to Test and Debugging" at the above network location.