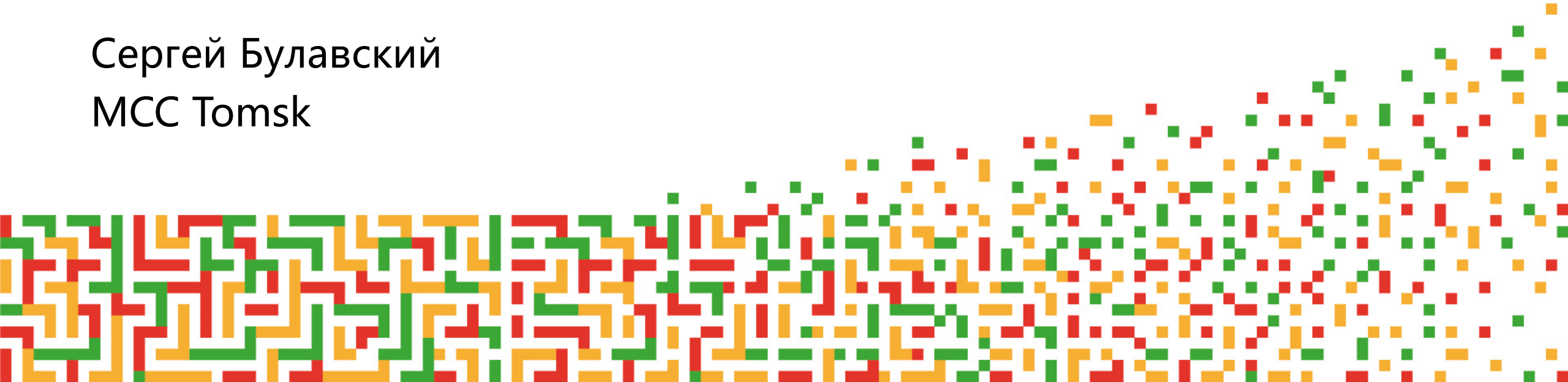
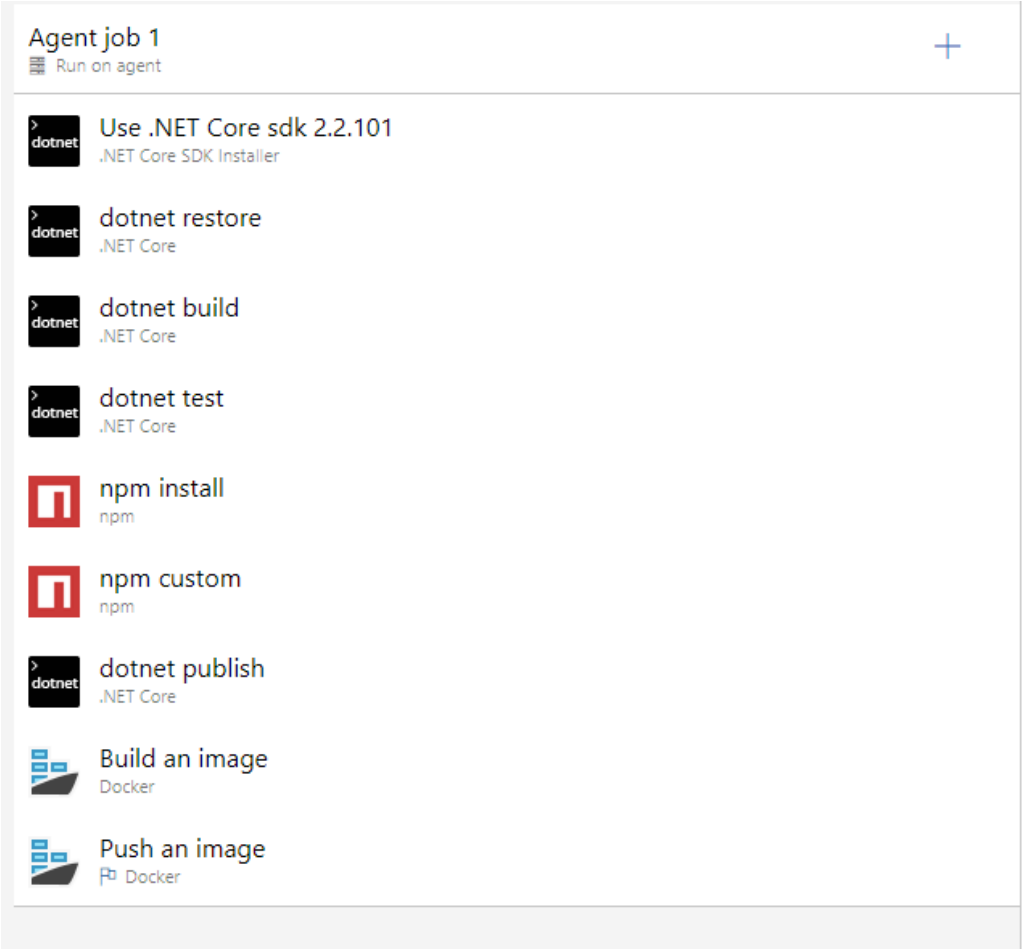
Build definition as a code with F#

Сергей Булавский
MCC Tomsk



Build definition in tfs



Build definition teamcity

Based on npm packages / npmPackageWithTest (detach)

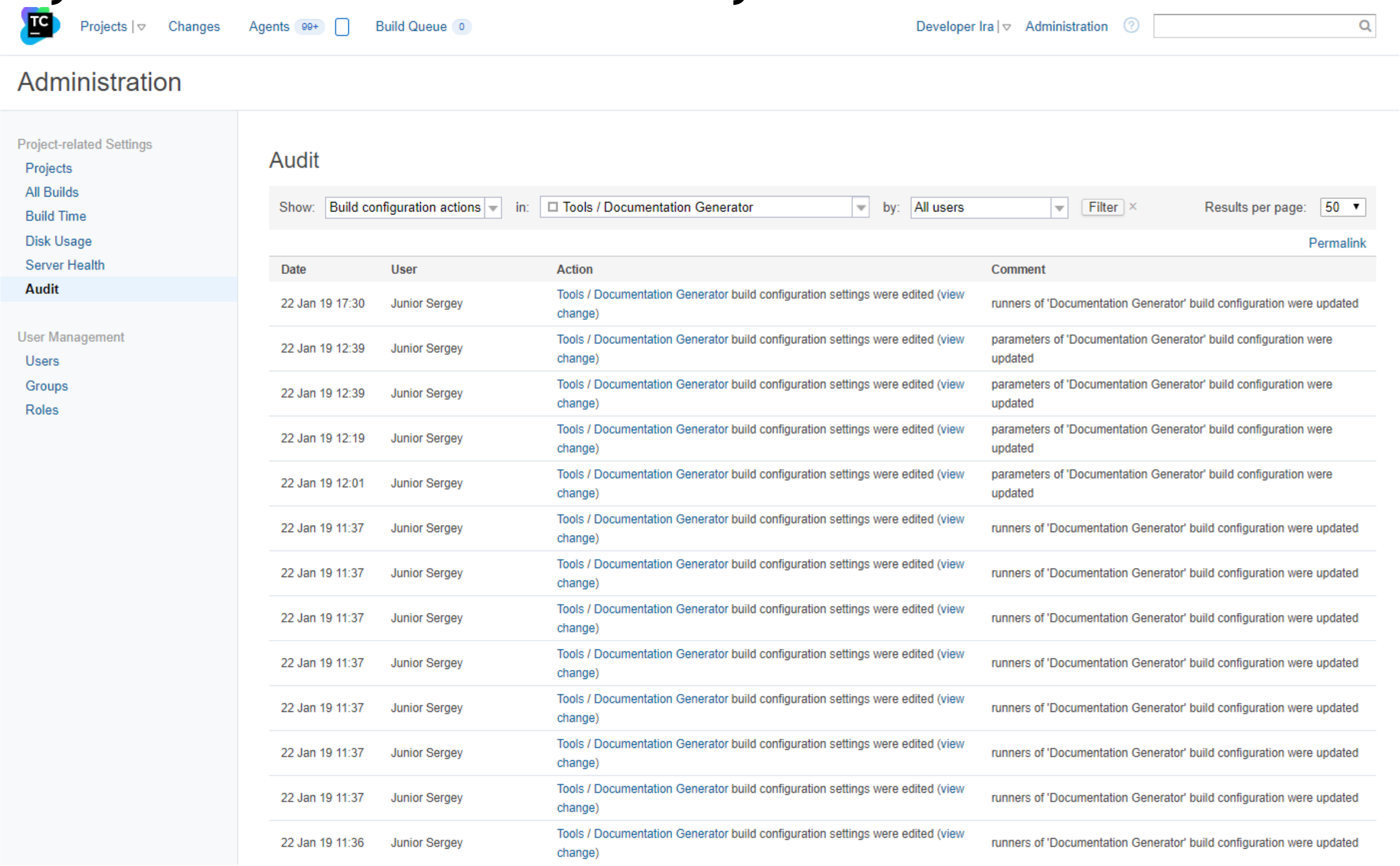
Build Steps

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 | Auto-detect build steps

Build Step	Parameters Description	
1. run npm version *(inherited)*	Command Line Custom script: npm version %BuildNumber% --force Execute: If all previous steps finished successfully	Edit
2. run npm install *(inherited)*	Command Line Custom script: npm install Execute: If all previous steps finished successfully	Edit
3. run npm test *(inherited)*	Command Line Custom script: npm run teamcitytest Execute: If all previous steps finished successfully	Edit
4. run npm publish *(inherited)*	Command Line Custom script: #!/bin/bash (and 6 more lines) Execute: If all previous steps finished successfully	Edit

Teamcity build definition history



Projects | Changes Agents 99+ Build Queue 0 Developer Ira | Administration

Administration

Project-related Settings

- Projects
- All Builds
- Build Time
- Disk Usage
- Server Health
- **Audit**

User Management

- Users
- Groups
- Roles

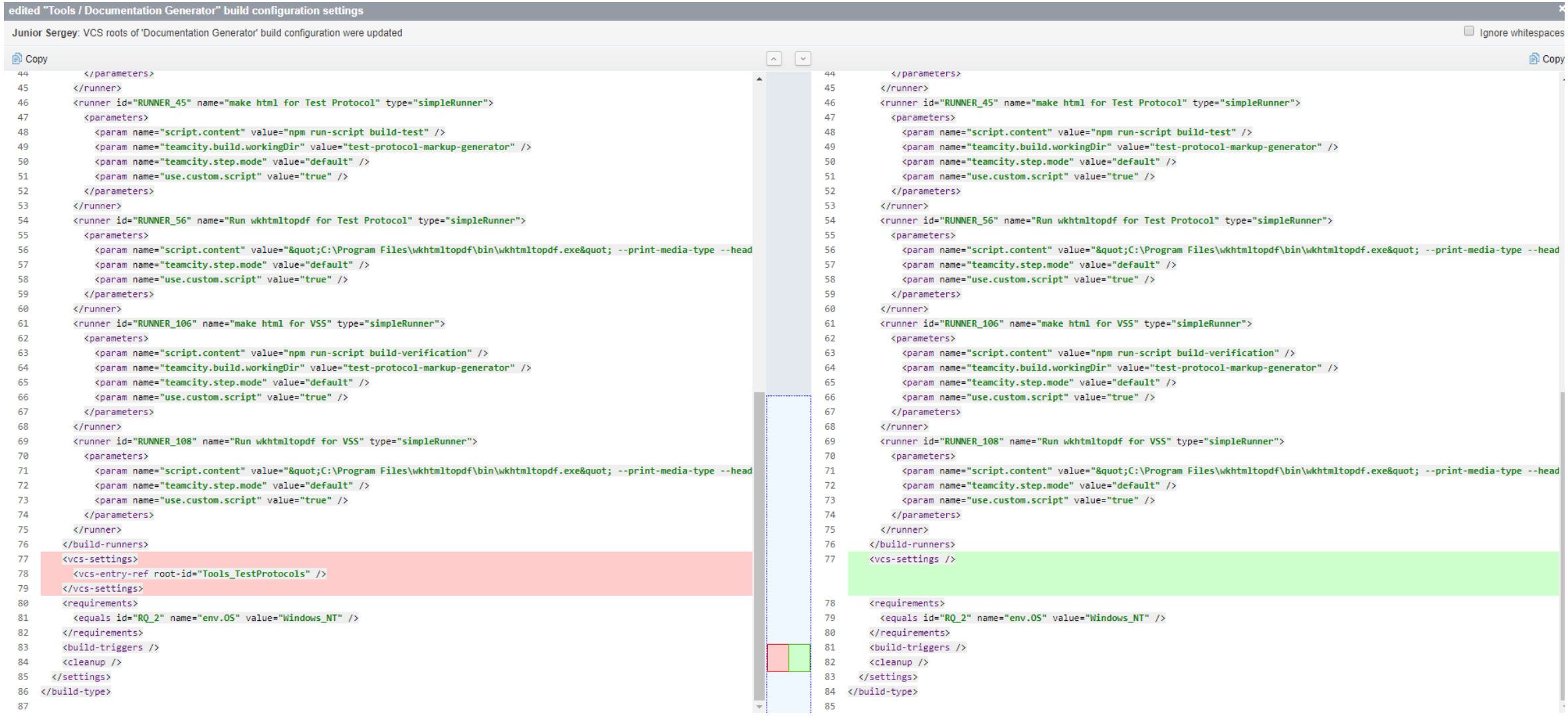
Audit

Show: Build configuration actions in: Tools / Documentation Generator by: All users Filter Results per page: 50

Permalink

Date	User	Action	Comment
22 Jan 19 17:30	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	runners of 'Documentation Generator' build configuration were updated
22 Jan 19 12:39	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	parameters of 'Documentation Generator' build configuration were updated
22 Jan 19 12:39	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	parameters of 'Documentation Generator' build configuration were updated
22 Jan 19 12:19	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	parameters of 'Documentation Generator' build configuration were updated
22 Jan 19 12:01	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	parameters of 'Documentation Generator' build configuration were updated
22 Jan 19 11:37	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	runners of 'Documentation Generator' build configuration were updated
22 Jan 19 11:37	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	runners of 'Documentation Generator' build configuration were updated
22 Jan 19 11:37	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	runners of 'Documentation Generator' build configuration were updated
22 Jan 19 11:37	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	runners of 'Documentation Generator' build configuration were updated
22 Jan 19 11:37	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	runners of 'Documentation Generator' build configuration were updated
22 Jan 19 11:37	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	runners of 'Documentation Generator' build configuration were updated
22 Jan 19 11:37	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	runners of 'Documentation Generator' build configuration were updated
22 Jan 19 11:36	Junior Sergey	Tools / Documentation Generator build configuration settings were edited (view change)	runners of 'Documentation Generator' build configuration were updated

Teamcity build definition history



Build definition in teamcity (kinda source)


```
13      </parameters>
14      <build-runners>
15        <runner id="RUNNER_118" name="run npm version" type="simpleRunner">
16          <parameters>
17            <param name="script.content" value="npm version %BuildNumber% --force" />
18            <param name="teamcity.step.mode" value="default" />
19            <param name="use.custom.script" value="true" />
20          </parameters>
21        </runner>
22        <runner id="RUNNER_116" name="run npm install" type="simpleRunner">
23          <parameters>
24            <param name="script.content" value="npm install" />
25            <param name="teamcity.step.mode" value="default" />
26            <param name="use.custom.script" value="true" />
27          </parameters>
28        </runner>
29        <runner id="RUNNER_117" name="run npm test" type="simpleRunner">
30          <parameters>
31            <param name="script.content" value="npm run teamcitytest" />
32            <param name="teamcity.step.mode" value="default" />
33            <param name="use.custom.script" value="true" />
34          </parameters>
35        </runner>
36        <runner id="RUNNER_119" name="run npm publish" type="simpleRunner">
37          <parameters>
38            <param name="script.content"><![CDATA[#!/bin/bash
39  branch='%teamcity.build.branch%'
40  echo 'branch is '$branch
41  if [[ $branch == *"pull"* ]]; then
42    exit 0
43  fi
44  npm publish --registry %npm_FeedAddress%]]></param>
45            <param name="teamcity.step.mode" value="default" />
46            <param name="use.custom.script" value="true" />
47          </parameters>
48        </runner>
49      </build-runners>
```

Build definition as a code with F#

Story time!

DevOps Vasya

- Powershell pro
- Many years of Windows
- Currently working with teamcity

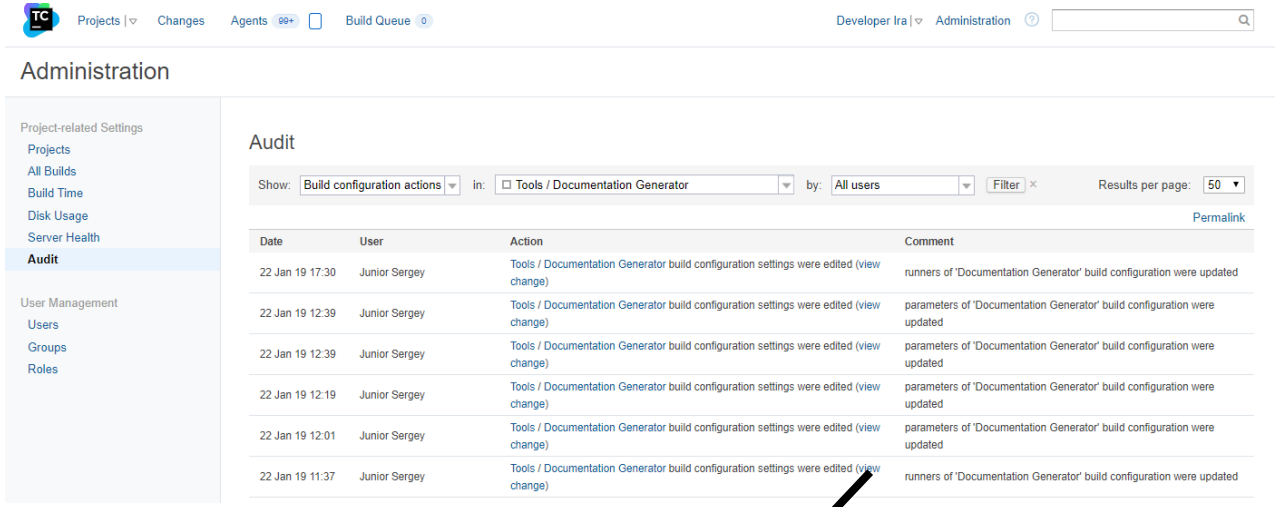
Developer Ira

- Confident middle software developer
- C#/TypeScript pro
- Barely knows PS

Junior Sergey

- Very curious
- Doing before thinking
- What is powershell?
- Git push before
  leaving on friday evening

Working with history in teamcity




```
    </runner>
    <runner id="RUNNER_106" name="make html for VSS" type="simpleRunner">
      <parameters>
        <param name="script.content" value="npm run-script build-verification" />
        <param name="teamcity.build.workingDir" value="test-protocol-markup-generator" />
        <param name="teamcity.step.mode" value="default" />
        <param name="use.custom.script" value="true" />
      </parameters>
    </runner>
    <runner id="RUNNER_108" name="Run wkhtmltopdf for VSS" type="simpleRunner">
      <parameters>
        <param name="script.content" value="&quot;C:\Program Files\wkhtmltopdf\bin\wkhtmltopdf.exe&quot; --print-media-type --head
        <param name="teamcity.step.mode" value="default" />
        <param name="use.custom.script" value="true" />
      </parameters>
    </runner>
  </build-runners>
  <vcs-settings>
    <vcs-entry-ref root-id="Tools_TestProtocols" />
  </vcs-settings>
  <requirements>
    <equals id="RQ_2" name="env.OS" value="Windows_NT" />
  </requirements>
  <build-triggers />
  <cleanup />
</settings>
</build-type>
```

```
    </runner>
    <runner id="RUNNER_106" name="make html for VSS" type="simpleRunner">
      <parameters>
        <param name="script.content" value="npm run-script build-verification" />
        <param name="teamcity.build.workingDir" value="test-protocol-markup-generator" />
        <param name="teamcity.step.mode" value="default" />
        <param name="use.custom.script" value="true" />
      </parameters>
    </runner>
    <runner id="RUNNER_108" name="Run wkhtmltopdf for VSS" type="simpleRunner">
      <parameters>
        <param name="script.content" value="&quot;C:\Program Files\wkhtmltopdf\bin\wkhtmltopdf.exe&quot; --print-media-type --head
        <param name="teamcity.step.mode" value="default" />
        <param name="use.custom.script" value="true" />
      </parameters>
    </runner>
  </build-runners>
  <vcs-settings />
  <requirements>
    <equals id="RQ_2" name="env.OS" value="Windows_NT" />
  </requirements>
  <build-triggers />
  <cleanup />
</settings>
</build-type>
```

Why build scripts

- Versioning (branching, rolling back) and
- Change tracking (Pull requests etc)
- Local run/debug
- Almost no build server configuration.

- Fake (F#)
- Cake (C#)
- PSAKE (Powershell)

What is F#?

- Open source
- Runs on Linux, Mac OS X, Android, IOS, GPU's (http://fsharp/use/gpu) and browsers (can generate JavaScript).
- .NET Compatible (Originated as Ocaml .NET Framework implementation)
- Functional-first
- Strongly typed

What is F#?


```
public class SomeClass
{
    public SomeClass(string firstField, int secondField)
    {
        FirstField = firstField;
        SecondField = secondField;
    }

    public string FirstField { get; private set; }
    public int SecondField { get; private set; }
}

public IEnumerable<TSource> Where<TSource>(
    IEnumerable<TSource> source,
    Func<TSource, bool> predicate
    )
{
    // standard LINQ implementation
    return source.Where(predicate);
}

public IEnumerable<IGrouping<Tkey, TSource>> GroupBy<TSource, TKey>(
    IEnumerable<TSource> source,
    Func<TSource, TKey> keySelector
    )
{
    // standard LINQ implementation
    return source.GroupBy(keySelector);
}
```

C#

Build definition as a code with F#

What is F#?


```csharp
public class SomeClass
{
    public SomeClass(string firstField, int secondField)
    {
        FirstField = firstField;
        SecondField = secondField;
    }

    public string FirstField { get; private set; }
    public int SecondField { get; private set; }
}

public IEnumerable<TSource> Where<TSource>(
    IEnumerable<TSource> source,
    Func<TSource, bool> predicate
    )
{
    // standard LINQ implementation
    return source.Where(predicate);
}

public IEnumerable<IGrouping<Tkey, TSource>> GroupBy<TSource, TKey>(
    IEnumerable<TSource> source,
    Func<TSource, TKey> keySelector
    )
{
    // standard LINQ implementation
    return source.GroupBy(keySelector);
}
```

C#

```fsharp
type SomeClass(firstField:string, secondField:int) =
    member this.FirstField = firstField
    member this.SecondField = secondField

let Where source predicate =
    Seq.filter predicate source

let GroupBy source keySelector =
    Seq.groupBy keySelector source
```

F#

```fsharp
type SomeClass(firstField:string, secondField:int) =
    member this.FirstField = firstField // string
    member this.SecondField = secondField // int

let Where source predicate =  // seq<'a> -> ('a -> bool) -> seq<'a>
    Seq.filter predicate source

let GroupBy source keySelector =  // seq<'a> -> ('a -> 'b) -> seq<'b * seq<'a>>
    Seq.groupBy keySelector source
```

Build definition as a code with F#

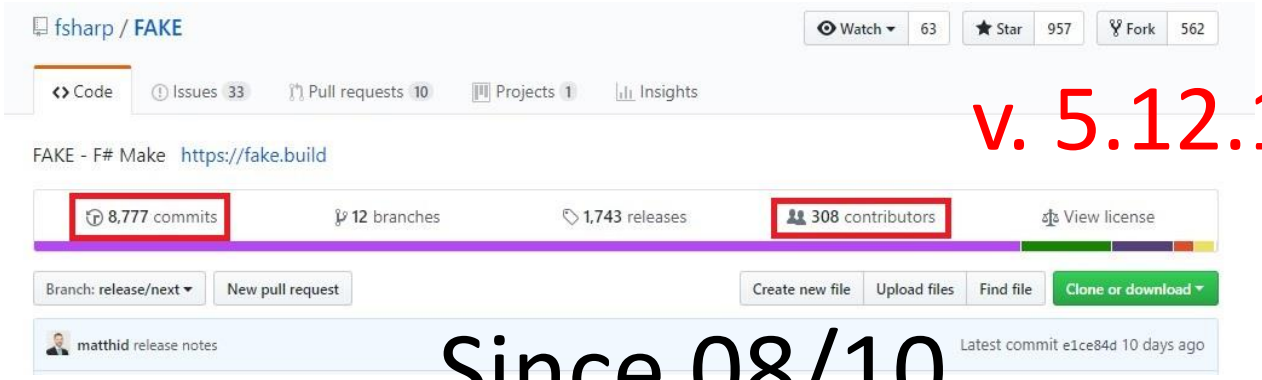
What is FAKE?

Fake (F# Make) – Open source DSL (based on F#) for building, copying, running tests, executing applications, and more.

Versions:
FAKE 4
FAKE 5

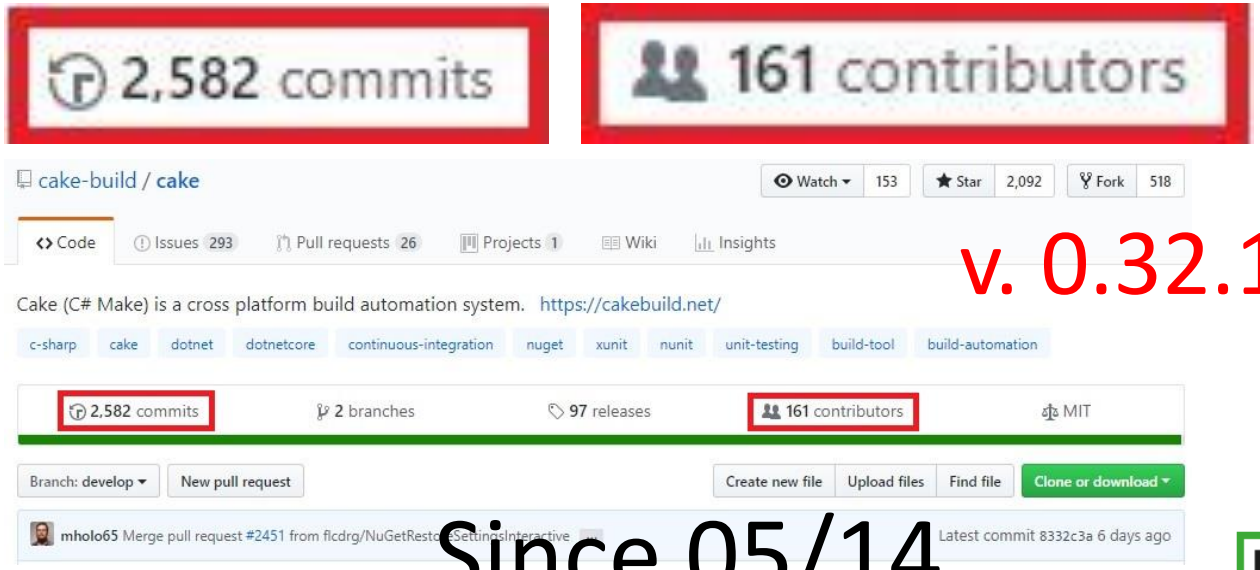
Why FAKE?

- Open source
- Community (Mainly C# developers)
- .NET/.NET Core
- Easy to use even without F# experience
- If Tools is not enough for you – write your own.



v. 5.12.1

Since 08/10



v. 0.32.1

Since 05/14

Syntax of .fsx


```
#load ".fake/build.fsx/intellisense.fsx” // Loads intellisense for current dependencies.

open Fake.Core                          // Opens specific namespace. (usings)
open Fake.DotNet
open Fake.IO
open Fake.IO.Globbing.Operators

Target.create "Clean" (fun _ ->
    !! "src/**/bin"                     // Find all matching pattern.
    ++ "src/**/obj"                     // Include all matching additional pattern.
    -- "src/**/do-not-delete"           // Exclude everything found by this pattern.
    |> Shell.cleanDirs                  // Clean all directories that will be found.
)

Target.create "Build" (fun _ ->
    !! "src/**/*.*proj"
    |> Seq.iter (DotNet.build id)
)

Target.create "All" ignore

"Clean"
  ==> "Build"                           // Can also be "Clean" ==> "Build" ==> "All"
  ==> "All"

Target.runOrDefault "All"               // What to run by default when script starts.
```

Syntax of .fsx - Dependency graph


```
"Build"
      ==> "NUnitTest"
      =?> ("xUnitTest", hasBuildParam "xUnitTest")
      ==> "Deploy"
```

Build dependencies and their relations


```
"CleanDotNetProjects"
      ==> "BuildDotNetProjects"
      ==> "UnitTests"
      ==> "NpmCleanInstall"
      ==> "NpmBuild"
      ==> "RunSeleniumTests"
      ==> "RunIntegrationTests"
      ==> "CheckNpmPackages"
      ==> "CI"
      ==> "PublishArtifacts"
```

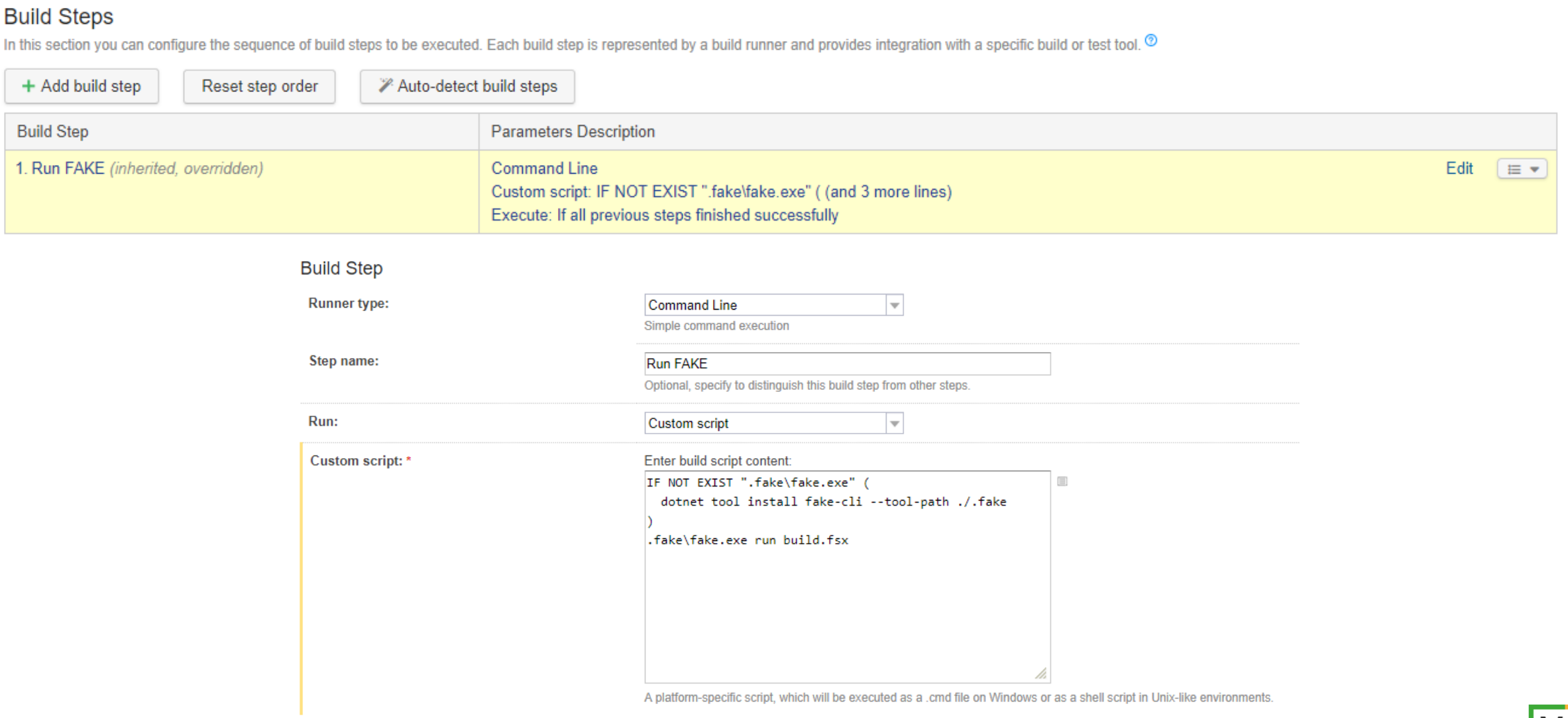
```
Shortened DependencyGraph for Target PublishArtifacts:
<== PublishArtifacts
   <== CI
      <== CheckNpmPackages
         <== RunIntegrationTests
            <== RunSeleniumTests
               <== NpmBuild
                  <== NpmCleanInstall
                     <== UnitTests
                        <== BuildDotNetProjects
                           <== CleanDotNetProjects
The running order is:
Group - 1
  - CleanDotNetProjects
Group - 2
  - BuildDotNetProjects
Group - 3
  - UnitTests
Group - 4
  - NpmCleanInstall
Group - 5
  - NpmBuild
Group - 6
  - RunSeleniumTests
Group - 7
  - RunIntegrationTests
Group - 8
  - CheckNpmPackages
Group - 9
  - CI
Group - 10
  - PublishArtifacts
```

Build dependencies and their relations (parallel)


```
"CleanDotNetProjects"
  ==> "BuildDotNetProjects"
  ==> "UnitTests"

"NpmCleanInstall"
  ==> "NpmBuild"

"UnitTests"
  ==> "RunIntegrationTests"
  ==> "CI“

"NpmBuild"
  ==> "RunSeleniumTests"
  ==> "CI"

"CheckNpmPackages" ==> "CI"
"CI" ==> "PublishArtifacts"
```

```
Shortened DependencyGraph for Target PublishArtifacts:
<== PublishArtifacts
   <== CI
      <== RunSeleniumTests
         <== NpmBuild
            <== NpmCleanInstall
      <== RunIntegrationTests
         <== UnitTests
            <== BuildDotNetProjects
               <== CleanDotNetProjects
      <== CheckNpmPackages

The running order is:
Group - 1
  - NpmCleanInstall
  - CleanDotNetProjects
  - CheckNpmPackages
Group - 2
  - NpmBuild
  - BuildDotNetProjects
Group - 3
  - RunSeleniumTests
  - UnitTests
Group - 4
  - RunIntegrationTests
Group - 5
  - CI
Group - 6
  - PublishArtifacts
```

Super complicated build server config? nope



Debugging? A bit tricky

Ionide-vscode-fake extension + Visual Studio Code

Visual Studio for windows

So what about next Friday evening?

Sergey could've:

- Experiment with build locally, in a branch.
- Run everything locally.
- Leave progress in his own git branch.

Ira could've:

- Easily roll back to last working version in git.
- Track changes that was made by sergey.
- Debug everything locally without taking any build agents.

Сергей Булавский
MCC Tomsk

Спасибо. Вопросы?

F# Make - https://fake.build/

F# Foundation - https://fsharp.org/

F# Foundation Slack - https://fsharp.slack.com

До встречи 21 марта на TomskDotNet #3!

Точка Кипения

21 марта, 18:30