Part 3: Build your Web App with Continua

This is part 3 of our Deploying Websites with Continua CI tutorial. It is highly recommended that you complete both Part 1: Create a Web Application and Part 2: Transform Web.Configs before continuing with this tutorial.

So far in this tutorial, we have created a test web application and dynamically transformed our web.configs. In part 3, we will add our web application to a Version Control System (VCS) and build the project in Continua CI using MSBuild.

Adding our Web Application to a Version Control System (VCS)

When using Continua CI, it is best practice to access any build process elements through a Version Control System (VCS). While we can access our Deploy Web Application through the file system, it is highly recommended that the Web Application is added to a VCS of your choosing.

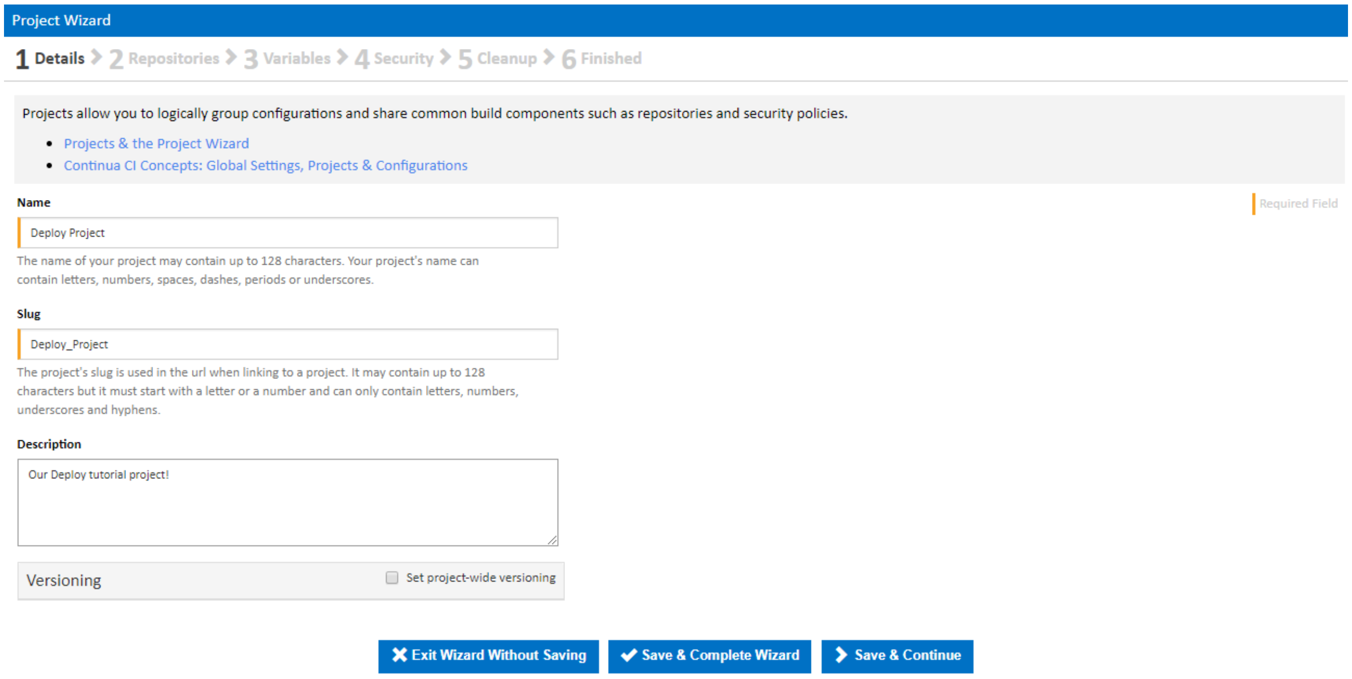
While you can use any VCS of your choosing, for this tutorial we will be using Mercurial as our VCS. Continua interacts with all VCSs the same way so you can use any VCS for this tutorial.

Go ahead and add your Web Application to a VCS of your choosing.

Create a Deploy Project and Configuration in Continua CI

For this tutorial we will create a new project and configuration in Continua CI. Lets call our project **Deploy Project** and then create a configuration under **Deploy Project** called **Deploy Configuration** (as shown below).

If you need additional help with creating your project or configuration then check out our getting started tutorials, in particular Part 1: Create your First Project and Part 2: Create your First Configuration.



Projects allow you to logically group configurations and share common build components such as repositories and security policies.

- Projects & the Project Wizard
- Continua CI Concepts: Global Settings, Projects & Configurations

Required Field

Name

Deploy Project

The name of your project may contain up to 128 characters. Your project's name can contain letters, numbers, spaces, dashes, periods or underscores.

Slug

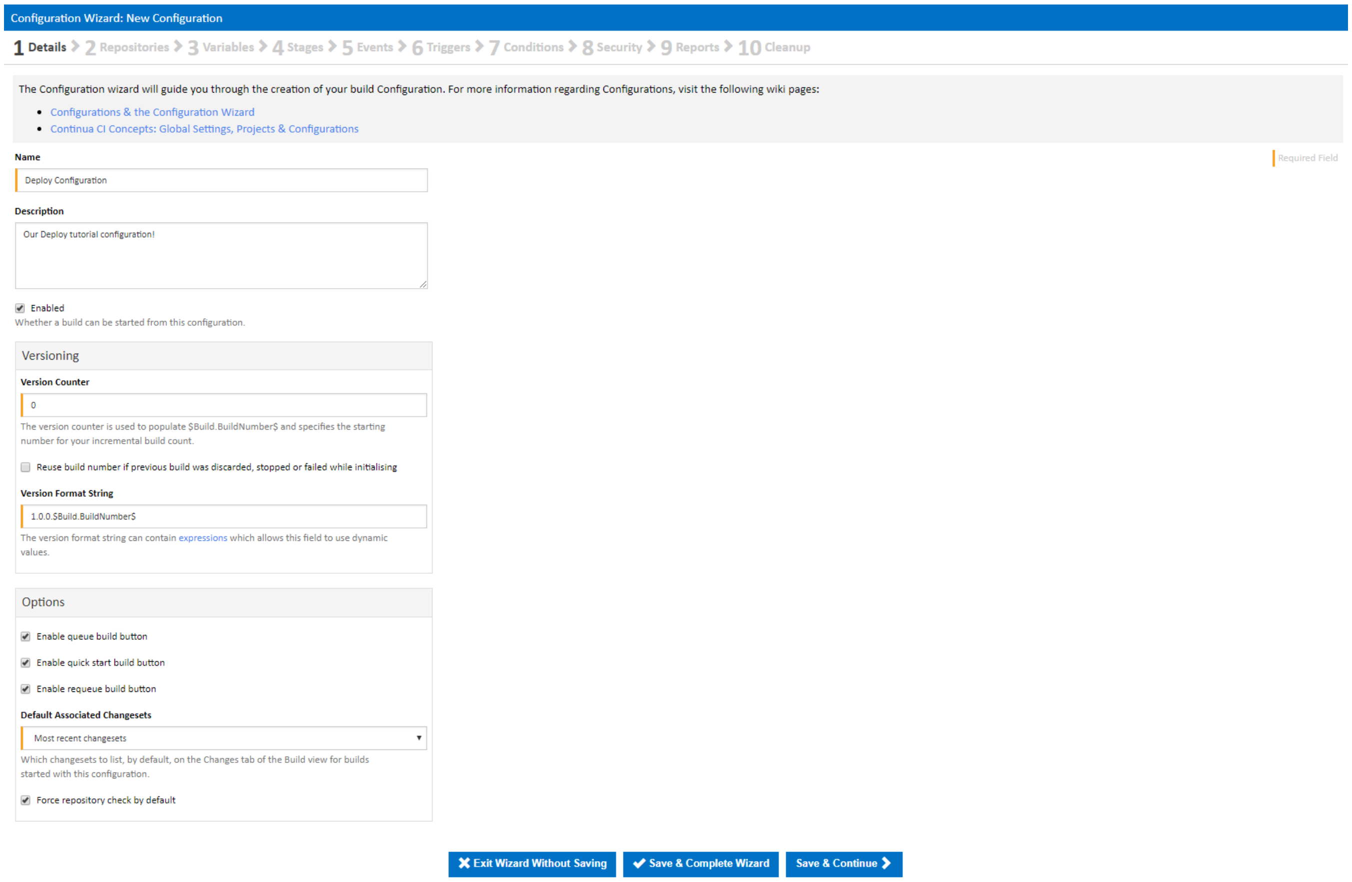
Deploy_Project

The project's slug is used in the url when linking to a project. It may contain up to 128 characters but it must start with a letter or a number and can only contain letters, numbers, underscores and hyphens.

Description

Our Deploy tutorial project!

Versioning ☐ Set project-wide versioning


Configuration Wizard: New Configuration

1 Details > 2 Repositories > 3 Variables > 4 Stages > 5 Events > 6 Triggers > 7 Conditions > 8 Security > 9 Reports > 10 Cleanup

The Configuration wizard will guide you through the creation of your build Configuration. For more information regarding Configurations, visit the following wiki pages:

- Configurations & the Configuration Wizard
- Continua CI Concepts: Global Settings, Projects & Configurations

Required Field

Name
Deploy Configuration

Description
Our Deploy tutorial configuration!

☑ Enabled
Whether a build can be started from this configuration.

Versioning

Version Counter
0
The version counter is used to populate $Build.BuildNumber$ and specifies the starting number for your incremental build count.

☐ Reuse build number if previous build was discarded, stopped or failed while initialising

Version Format String
1.0.0.$Build.BuildNumber$
The version format string can contain expressions which allows this field to use dynamic values.

Options

☑ Enable queue build button
☑ Enable quick start build button
☑ Enable requeue build button

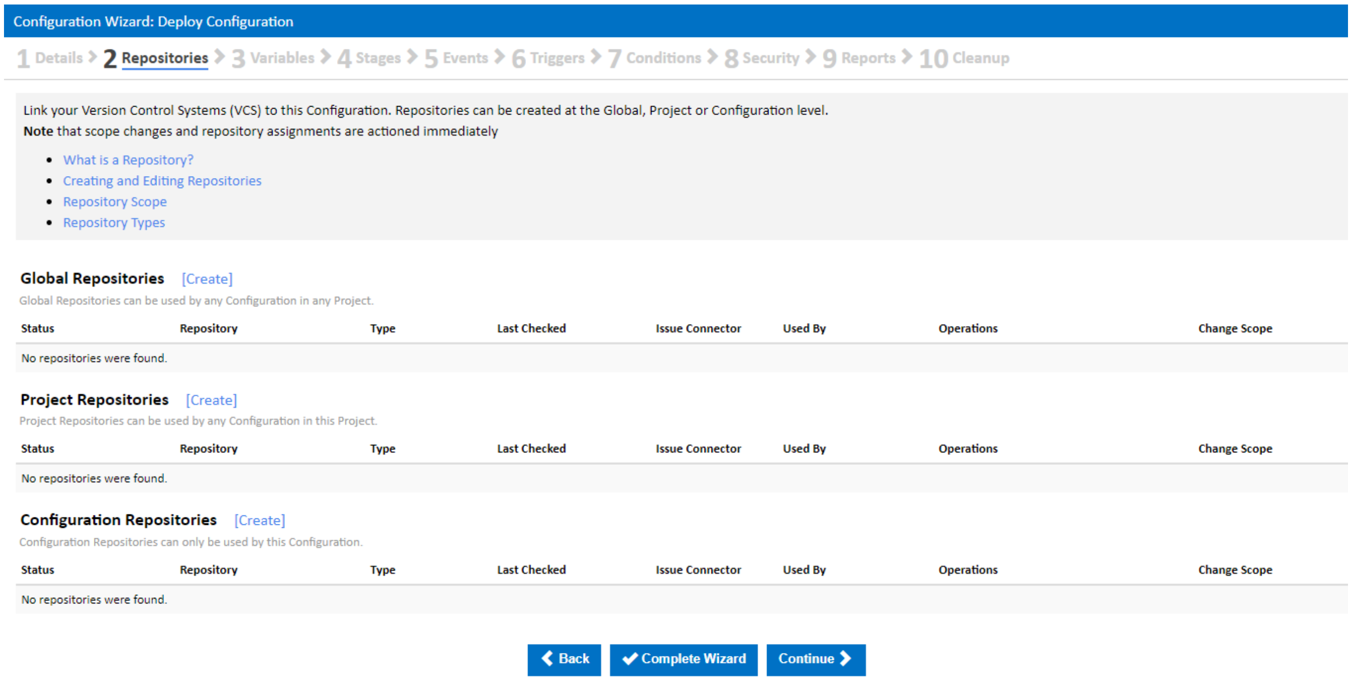
Default Associated Changesets
Most recent changesets
Which changesets to list, by default, on the Changes tab of the Build view for builds started with this configuration.

☑ Force repository check by default

Link Our Configuration to Our VCS

Now that we have our Deploy project and Deploy Configuration, lets link our **Web Application** to our **Deploy Configuration** via our VCS.

Navigate to the **Repositories page** under the **Configuration Wizard** for our **Deploy Configuration**. On this page you can link your VCS to Continua CI at either the Global, Project or Configuration level (learn more about Repository Scope). For this tutorial, lets link our repository to the Configuration by clicking the **[Create]** link next to the **Configuration Repositories** header.



Configuration Wizard: Deploy Configuration

1 Details > 2 Repositories > 3 Variables > 4 Stages > 5 Events > 6 Triggers > 7 Conditions > 8 Security > 9 Reports > 10 Cleanup

Link your Version Control Systems (VCS) to this Configuration. Repositories can be created at the Global, Project or Configuration level.
Note that scope changes and repository assignments are actioned immediately

- What is a Repository?
- Creating and Editing Repositories
- Repository Scope
- Repository Types

Global Repositories [Create]
Global Repositories can be used by any Configuration in any Project.

Status	Repository	Type	Last Checked	Issue Connector	Used By	Operations	Change Scope
No repositories were found.							

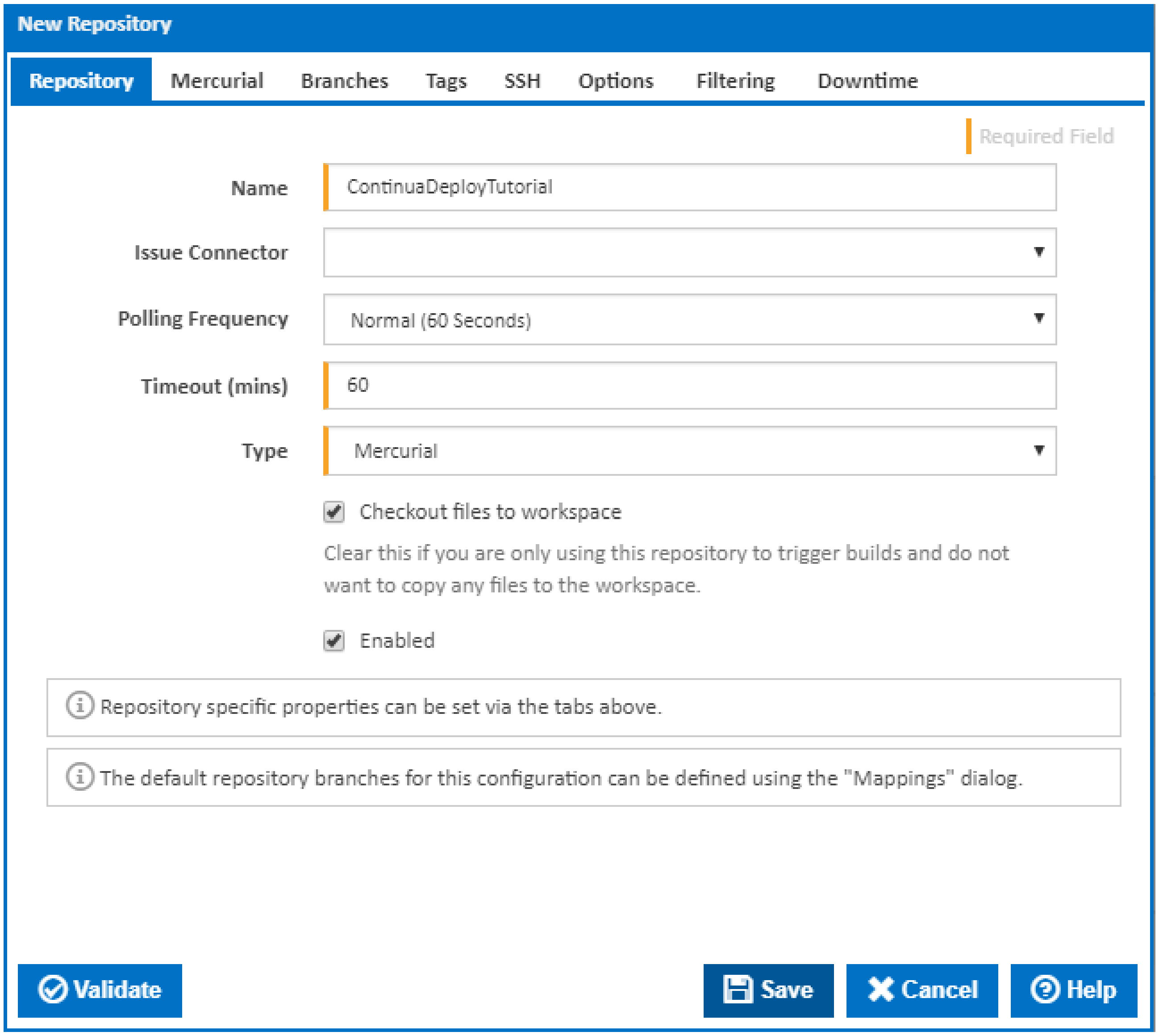
Project Repositories [Create]
Project Repositories can be used by any Configuration in this Project.

Status	Repository	Type	Last Checked	Issue Connector	Used By	Operations	Change Scope
No repositories were found.							

Configuration Repositories [Create]
Configuration Repositories can only be used by this Configuration.

Status	Repository	Type	Last Checked	Issue Connector	Used By	Operations	Change Scope
No repositories were found.							

Clicking the **[Create]** link will bring up the Create Repositories dialog. Using this dialog, enter any VCS specific information that Continua will need to successfully link to your repository. Lets call our repository **ContinuaDeployTutorial**. For this tutorial I have created a Mecurial repository for our Web Application (as shown below). Once you have entered all your VCS settings, you can click Validate which will determine whether your repository is valid and has successfully been linked to Continua.


New Repository

Repository | Mercurial | Branches | Tags | SSH | Options | Filtering | Downtime

Required Field

Name: ContinuaDeployTutorial

Issue Connector:

Polling Frequency: Normal (60 Seconds)

Timeout (mins): 60

Type: Mercurial

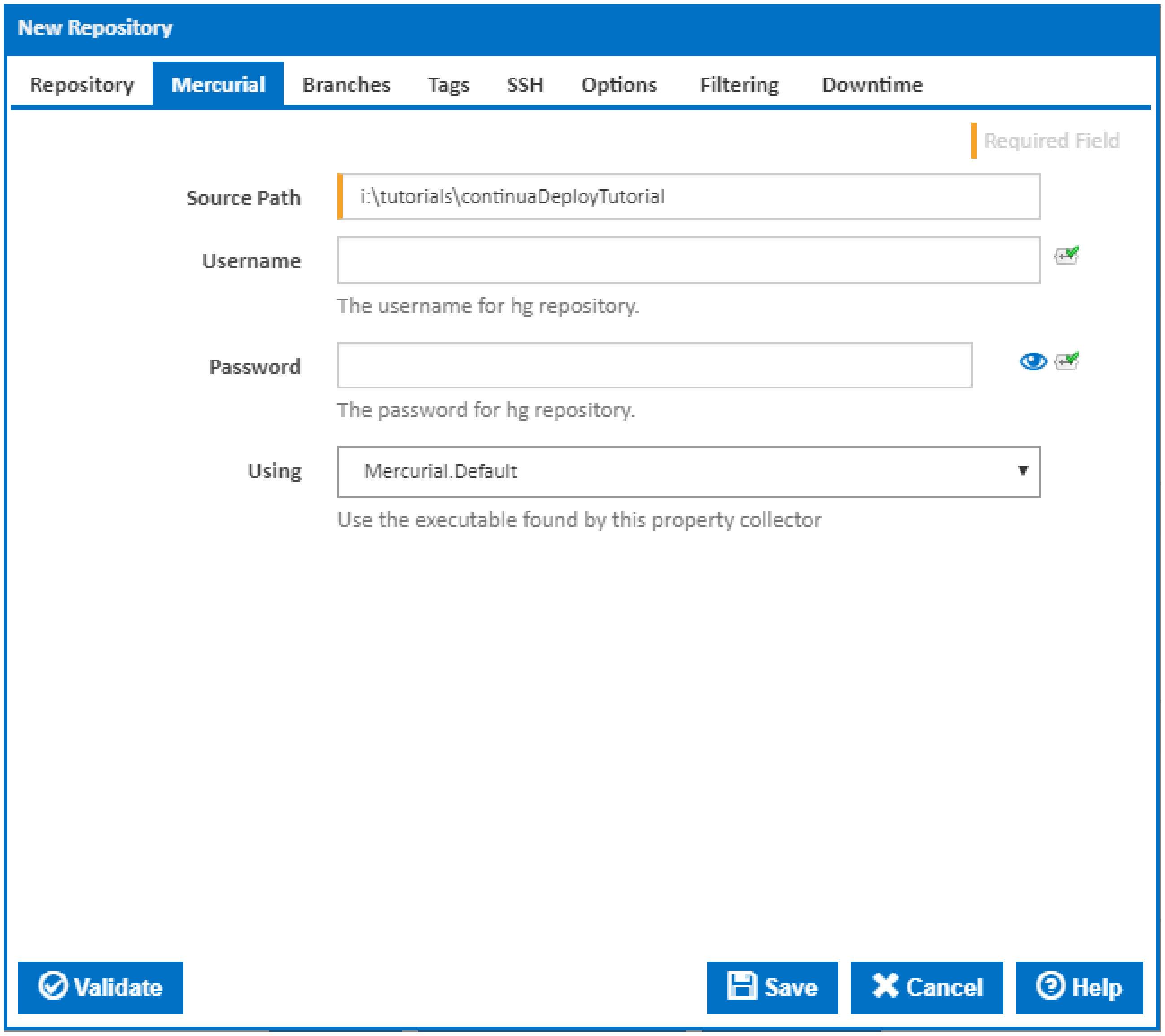
[✓] Checkout files to workspace

Clear this if you are only using this repository to trigger builds and do not want to copy any files to the workspace.

[✓] Enabled

Repository specific properties can be set via the tabs above.

The default repository branches for this configuration can be defined using the "Mappings" dialog.



New Repository

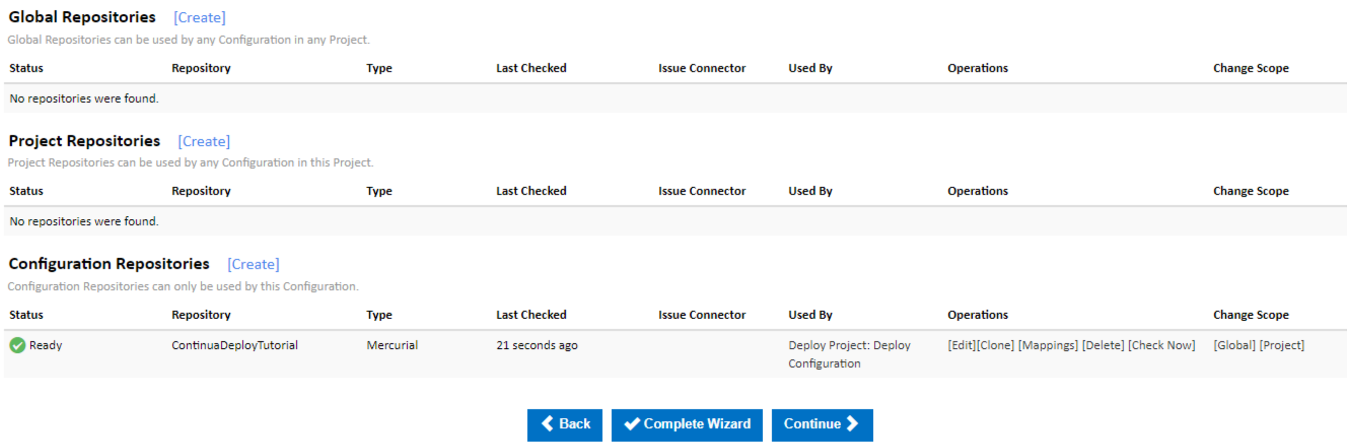
Repository | Mercurial | Branches | Tags | SSH | Options | Filtering | Downtime

Required Field

Source Path	i:\tutorials\continuaDeployTutorial
Username	
	The username for hg repository.
Password	
	The password for hg repository.
Using	Mercurial.Default
	Use the executable found by this property collector

Validate | Save | Cancel | Help

Once you have saved your repository, you should then see it appear under Configuration Repositories, as shown below.



Global Repositories [Create]

Global Repositories can be used by any Configuration in any Project.

Status	Repository	Type	Last Checked	Issue Connector	Used By	Operations	Change Scope
No repositories were found.							

Project Repositories [Create]

Project Repositories can be used by any Configuration in this Project.

Status	Repository	Type	Last Checked	Issue Connector	Used By	Operations	Change Scope
No repositories were found.							

Configuration Repositories [Create]

Configuration Repositories can only be used by this Configuration.

Status	Repository	Type	Last Checked	Issue Connector	Used By	Operations	Change Scope
✓ Ready	ContinuaDeployTutorial	Mercurial	21 seconds ago		Deploy Project: Deploy Configuration	[Edit][Clone] [Mappings] [Delete] [Check Now]	[Global] [Project]

Back | Complete Wizard | Continue

Create a Build Process

Now that we have linked our Web Application to Continua CI, we will need to create an MSBuild action to build our Web Application Solution. For this tutorial, we will run **MSBuild over our project twice**. Basically before we can package our Web Application, we need to Compile the entire solution, which may include tests, services, etc. Once the solution has compiled correctly, we can then run MSBuild over our website project file. With this in mind, we will need a build process that looks something like this:

- **Create an output directory** for our solution in the Agent's workspace.
- **Build our solution** (i.e. the .sln file) using MSBuild.
- **Create an output directory** for our website in the Agent's workspace.
- **Build and Package our Website project** (i.e. the .csproj file) using MSBuild.

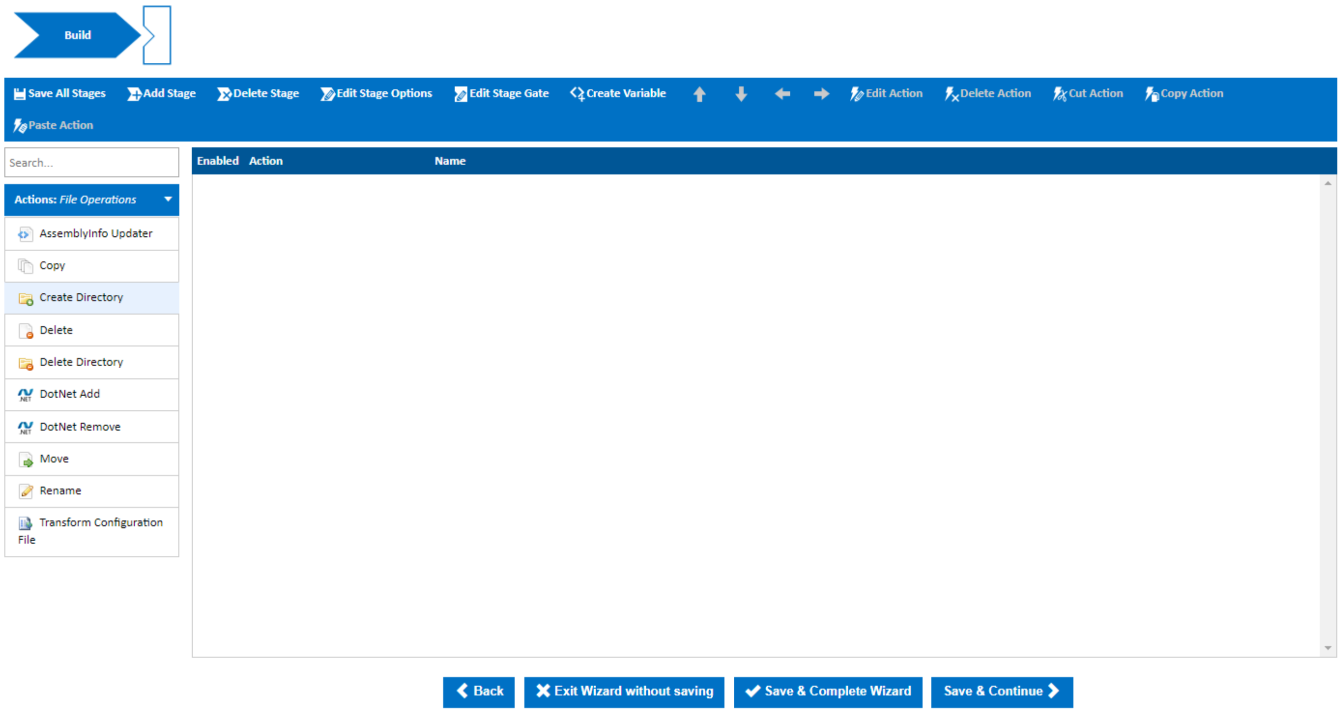
While in the **Configuration Wizard**, navigate to the **Stages** page. The stages page is where you add your individual build actions to the configuration. It is also where you control the flow of various actions by using if, else, etc. actions.

Once you have navigated to the Stages page, you should see that a **Build stage has already been created for you**. This stage is created for new configurations by default, however it can be renamed.

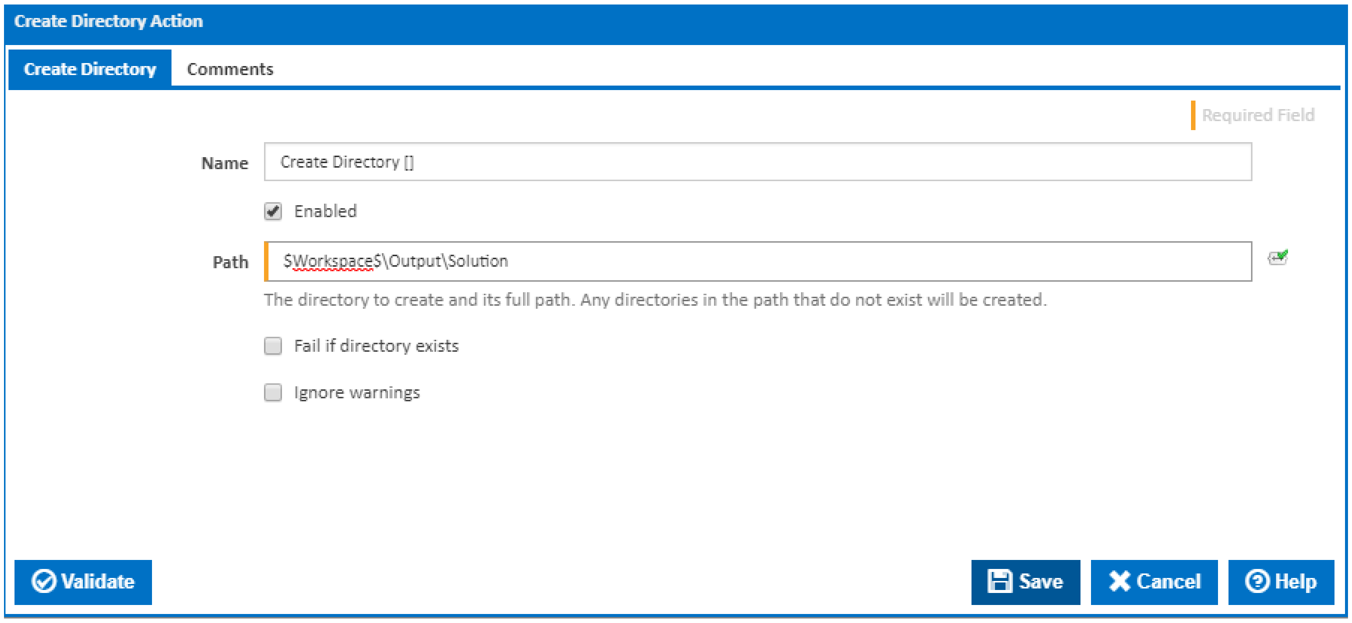
Create our Solution Output Directory

Before we can create our MSBuild action, we will need to create an output directory in the Agents workspace. This directory will be where our built ContinuaDeployTutorial solution will be stored. In Continua, it is best practice to store all information relating to a build within the Agent's workspace. This means that all information regarding an individual build is contained within it's own build folder.

So lets create a Create Directory Action by clicking the **File Operations Category** and selecting the **Create Directory Action** (As shown below).



This will bring up the **Create Directory Action** dialog. The only property that we need to set is the **Path property**, which should be set to $Workspace$\Output\Solution. This is telling Continua to **create the Output/Solution directories in the Agent Workspace**. This directory will be where we store our Web App once we have built our project.


Create Directory Action

Create Directory | Comments

Required Field

Name: Create Directory []

[✓] Enabled

Path: $Workspace$\Output\Solution

The directory to create and its full path. Any directories in the path that do not exist will be created.

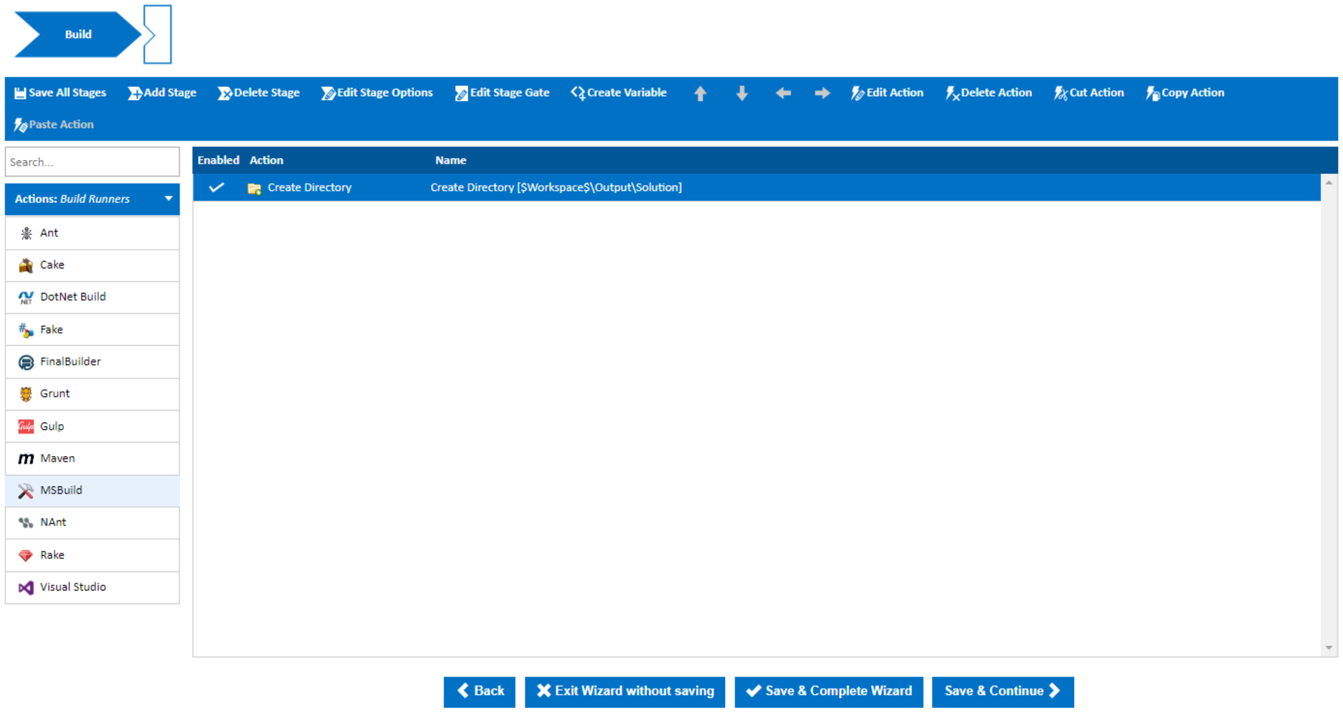
[ ] Fail if directory exists

[ ] Ignore warnings

Once you have saved the action, you will see that your Create Directory Action has been added to the Workflow window in the Build Stage.

Build our Solution using MSBuild

Lets go ahead and **Create a MSBuild action on the Build stage** by clicking the **Build Runners category** and selecting the **MSBuild action**. This will add the MSBuild action below the Create Directory action that we have already created.

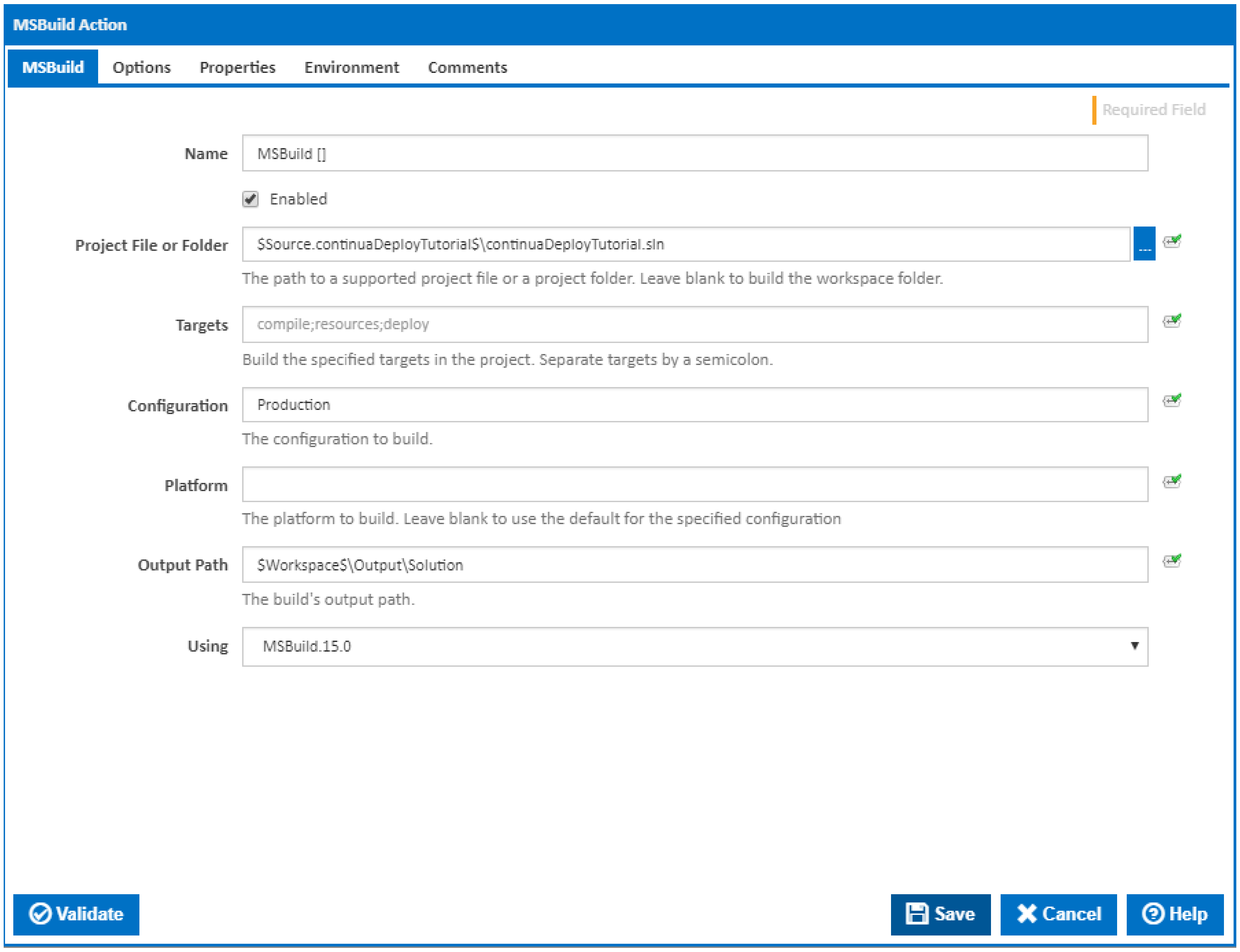


Enabled	Action	Name
✓	Create Directory	Create Directory [$Workspace$\Output\Solution]

Selecting the MSBuild action will bring up the Create Action Dialog which allows you to set the parameters that will be used when the MSBuild action is executed. So lets set the following parameters:

- Firstly, lets point our action to our **Web Application Solution file** in our newly created **continuaDeployTutorial repository**. To point an action to a repository we use expressions to reference our repository. So to point our MSBuild action to our Solution file, we would use the following syntax: $Source.<repository_name>$/<file_path>. So lets set the **Project File property** to **$Source. continuaDeployTutorial$\ContinuaDeployTutorial.sln**.

- We will also need to provide an **Output Path** for the MSBuild action. This should be **set to a folder where our built project should be stored**. Again, we can use expressions to point this property to the Agents workspace directory using the $Workspace$\<subdirectory> syntax. So lets set MSBuild's Output Path property to **$Workspace$\Output**.
- We need to set the **Configuration property** which tells MSBuild which Visual Studio Solution Configuration should be used when building our solution. Now that we have our Web.Config transforms configured correctly, lets set the **Configuration property** to **Production**.
- Finally, lets set the **Using property** to **MSBuild 15.0**.



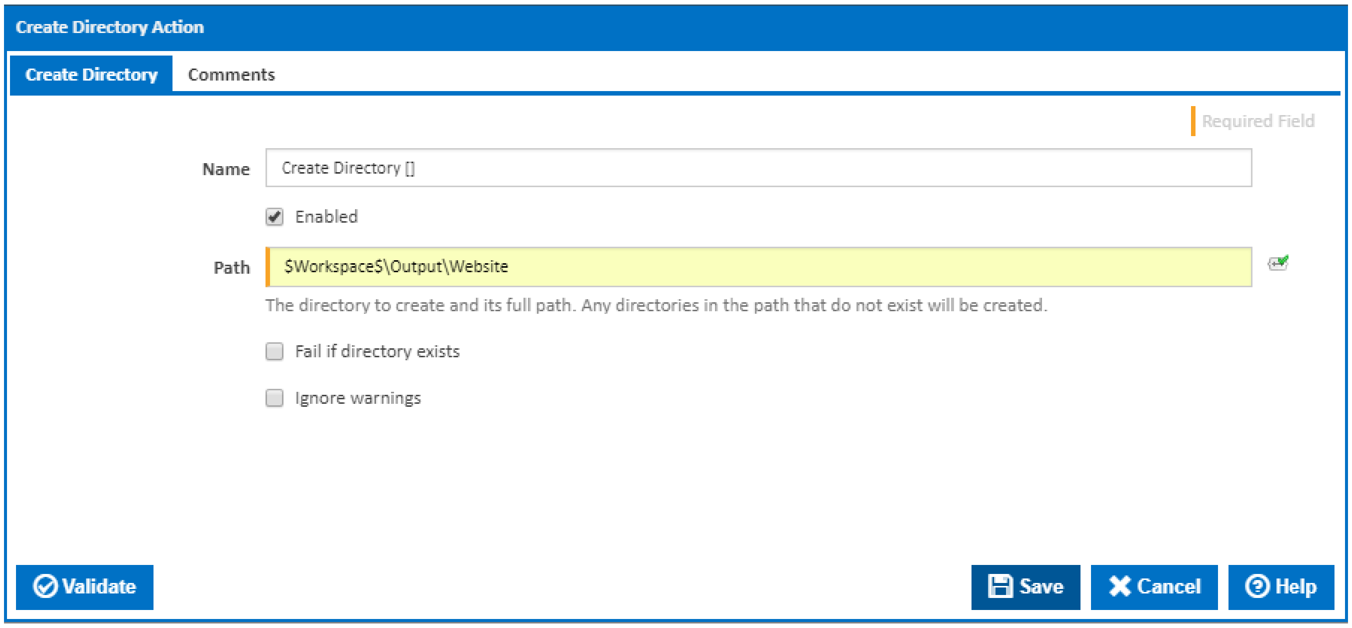
Name	MSBuild []
	✓ Enabled
Project File or Folder	$Source.continuaDeployTutorial$\continuaDeployTutorial.sln The path to a supported project file or a project folder. Leave blank to build the workspace folder.
Targets	compile;resources;deploy Build the specified targets in the project. Separate targets by a semicolon.
Configuration	Production The configuration to build.
Platform	 The platform to build. Leave blank to use the default for the specified configuration
Output Path	$Workspace$\Output\Solution The build's output path.
Using	MSBuild.15.0

Once all these properties have been set, **save the action** and you should see it appear below our Create Directory Action.

Create our Website Output Directory

Now that we have created our actions that will build our solution, we need to add another **Create Directory action** and another **MSBuild action** that will build our Web Application project.

So lets add another Create Directory action, but this time we want to create the directory **$Workspace$\Output\Website** (as shown below).


Create Directory Action

Create Directory | Comments

Required Field

Name: Create Directory []

☑ Enabled

Path: $Workspace$\Output\Website

The directory to create and its full path. Any directories in the path that do not exist will be created.

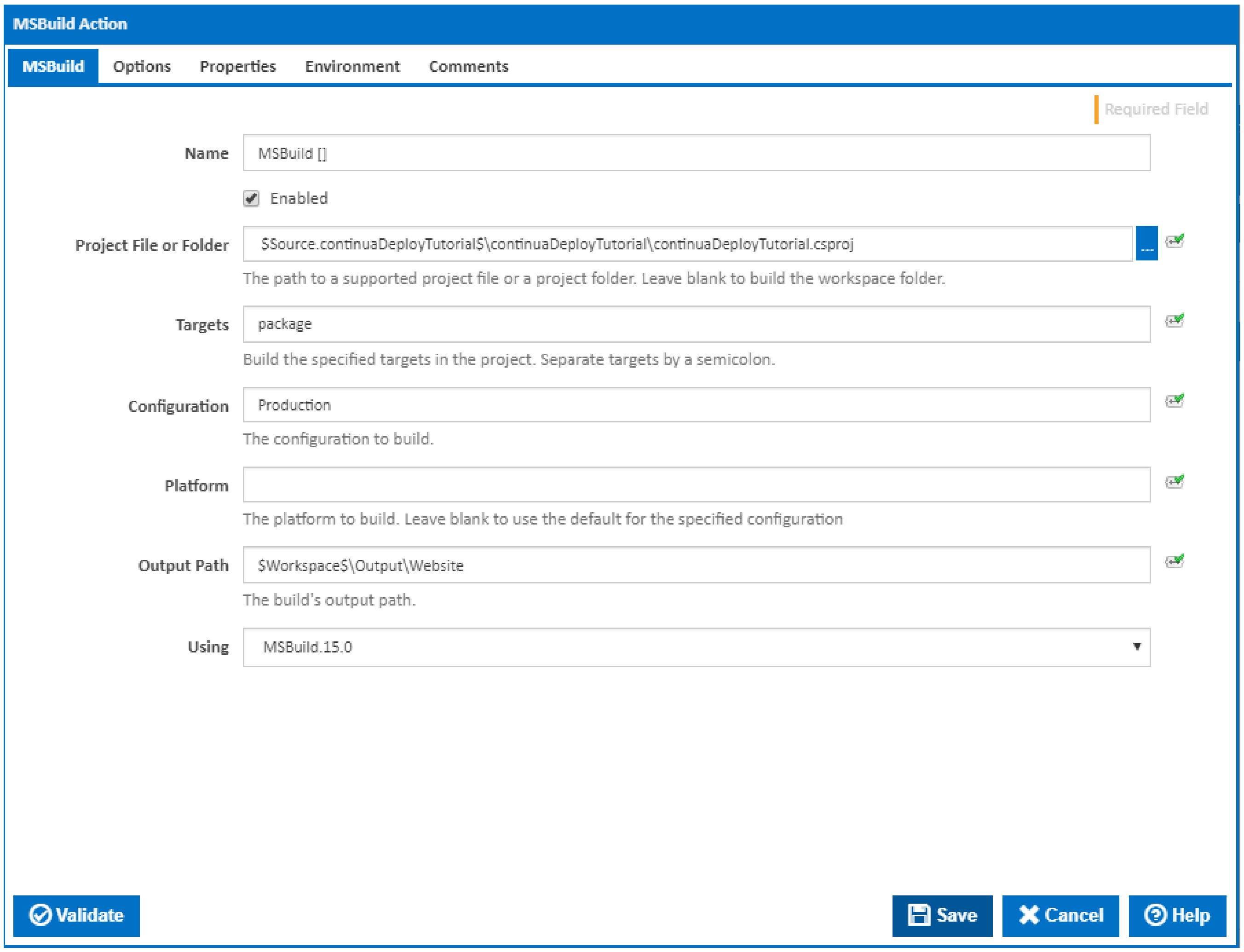
☐ Fail if directory exists

☐ Ignore warnings

Build and Package our Web Application Project using MSBuild

Now we can finally build and package our Web Application so that it is ready to be deployed. Create anther MSBuild action and give it the following properties:

- Set the project file to **the project file of the Web Application**. This will build only the Website and no other components in your solution. For our example, we have given the project file the following value: **$Source.continuaDeployTutorial$\continuaDeployTutorial\continuaDeployTutorial.csproj**
- Set **Targets** to **package**. This tells MSBuild that it should also package up our website into its own .zip folder so that we can then deploy our site to our production server.
- Set **Configuration** to **Production** so that our Production web.config will be used.
- Set the **Output Path** to our Website output folder, **$Workspace$\Output\Website**.
- Set the **Using** property to **MSBuild 15.0**.



Name: MSBuild []

[✓] Enabled

Project File or Folder: $Source.continuaDeployTutorial$\continuaDeployTutorial\continuaDeployTutorial.csproj
The path to a supported project file or a project folder. Leave blank to build the workspace folder.

Targets: package
Build the specified targets in the project. Separate targets by a semicolon.

Configuration: Production
The configuration to build.

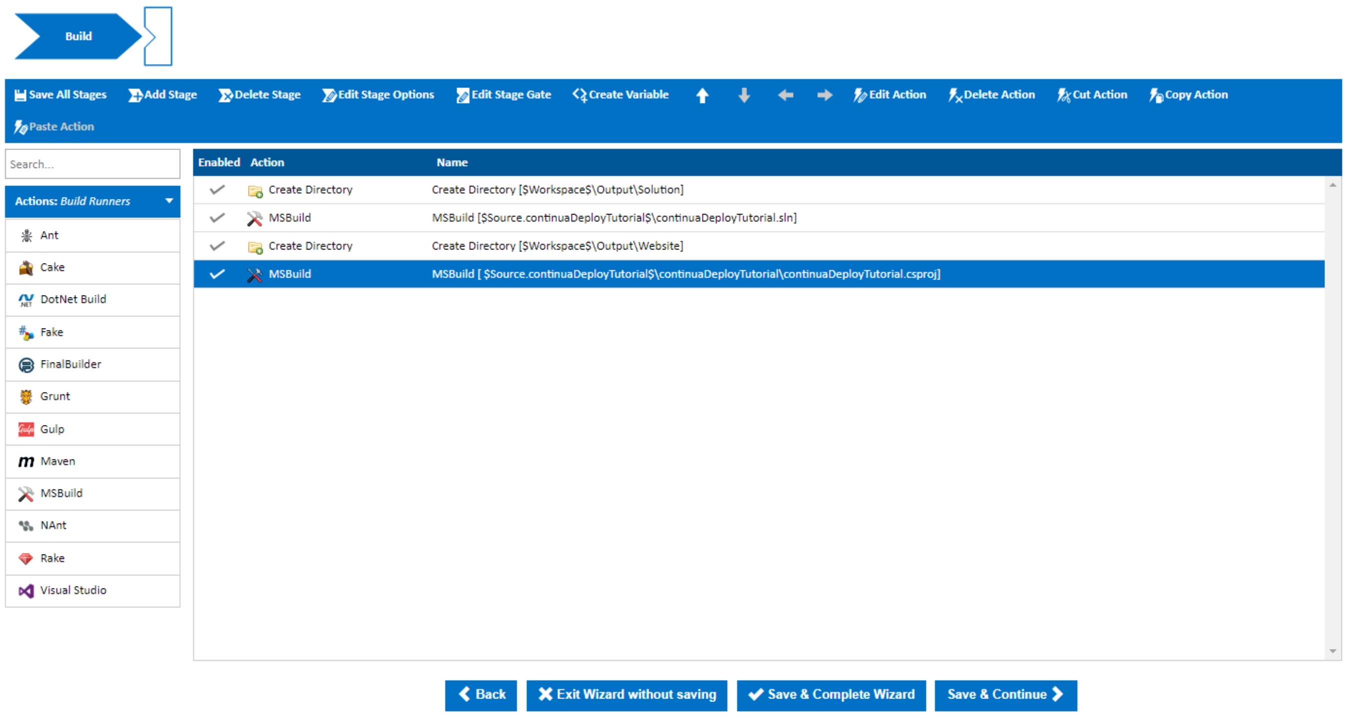
Platform:
The platform to build. Leave blank to use the default for the specified configuration

Output Path: $Workspace$\Output\Website
The build's output path.

Using: MSBuild.15.0

Reviewing your Build Process

Once we have created all 4 actions, our build process should look something like this:

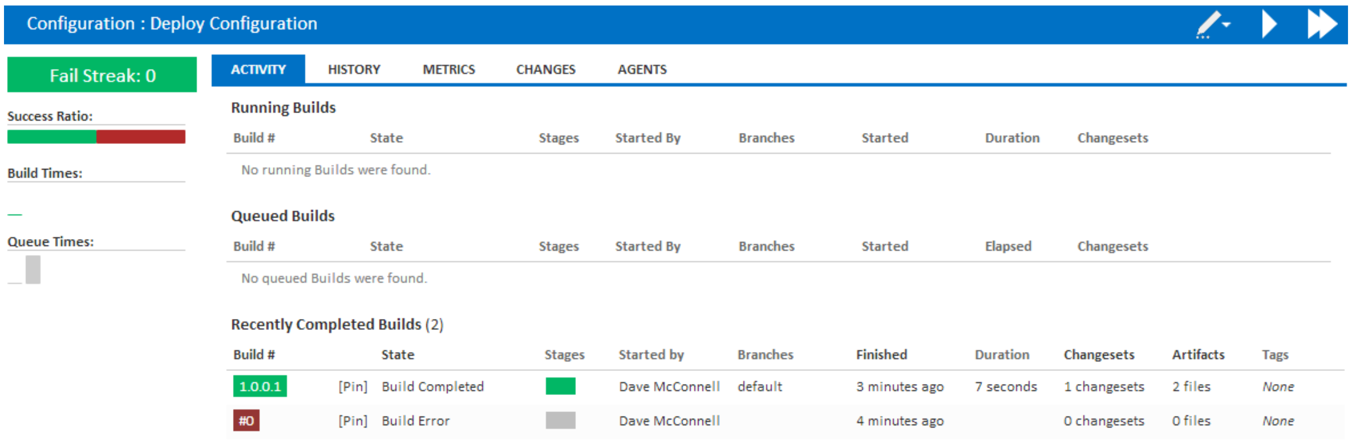


Enabled	Action	Name
✓	Create Directory	Create Directory [$Workspace$\Output\Solution]
✓	MSBuild	MSBuild [$Source.continuaDeployTutorial$\continuaDeployTutorial.sln]
✓	Create Directory	Create Directory [$Workspace$\Output\Website]
✓	MSBuild	MSBuild [$Source.continuaDeployTutorial$\continuaDeployTutorial\continuaDeployTutorial.csproj]

Once you are happy with your build process, click the **Save & Complete Wizard** button.

Build our Deploy Project

Now that we have created our actions, lets run the build. Navigate to the Configuration page and click the **Fast Run button (the fast forward button)**. Once the build has finished running, it should turn green if it completed successfully.



Running Builds

Build #	State	Stages	Started By	Branches	Started	Duration	Changesets
No running Builds were found.							

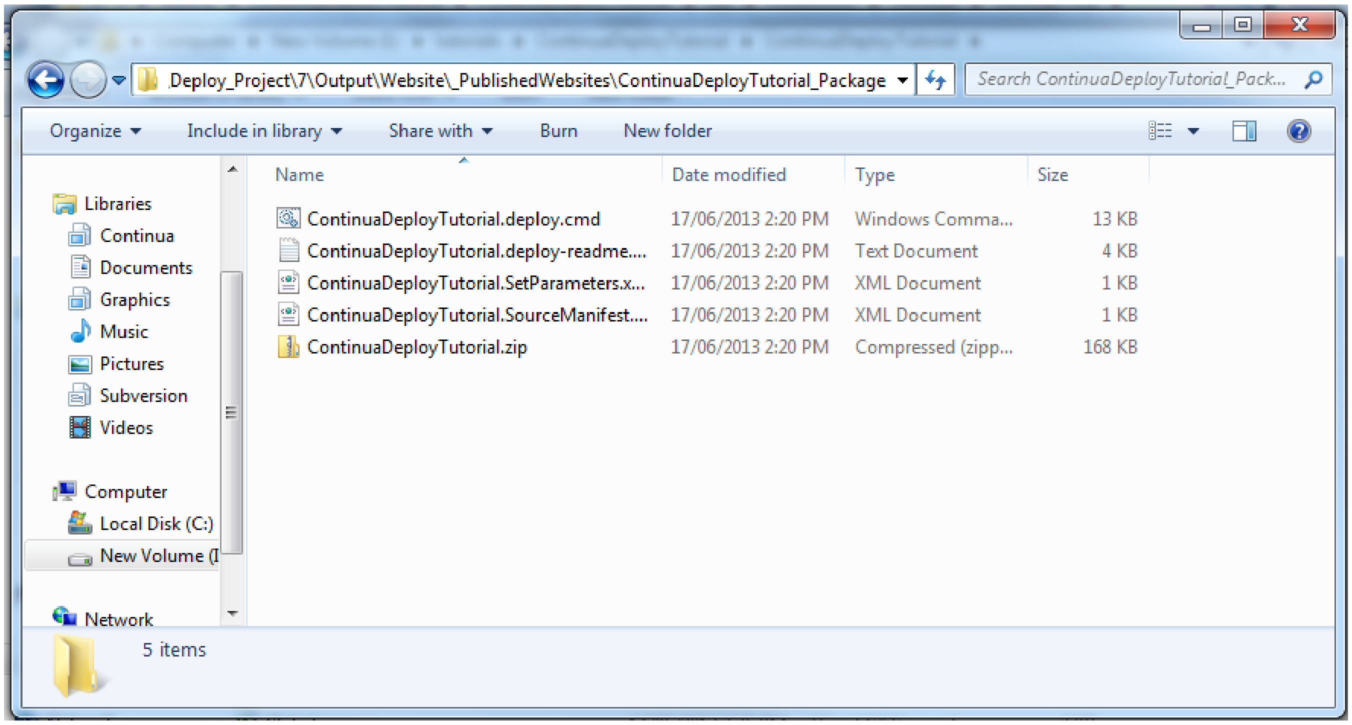
Queued Builds

Build #	State	Stages	Started By	Branches	Started	Elapsed	Changesets
No queued Builds were found.							

Recently Completed Builds (2)

Build #		State	Stages	Started by	Branches	Finished	Duration	Changesets	Artifacts	Tags
1.0.0.1	[Pin]	Build Completed	🟩	Dave McConnell	default	3 minutes ago	7 seconds	1 changesets	2 files	None
#0	[Pin]	Build Error	▨	Dave McConnell		4 minutes ago		0 changesets	0 files	None

Once your build has completed successfully, you can see the end result of our build by navigating to your Continua server workspace and finding your build folder. Your build can be found on your Continua server in the following directory: **<your_continua_share>\Ws\<project_name>\<build_number>\Output**. Navigate to this directory and you should see your 2 directories, **Solution** and **Website**. The Solution directory should now contain your built solution while the Website directory should contain your built project and your packaged project. Your packaged project should look similar to the packaged project shown below. It is this package that we now want to deploy to our production server.


Deploy_Project\7\Output\Website_PublishedWebsites\ContinuaDeployTutorial_Package

Name	Date modified	Type	Size
ContinuaDeployTutorial.deploy.cmd	17/06/2013 2:20 PM	Windows Comma...	13 KB
ContinuaDeployTutorial.deploy-readme....	17/06/2013 2:20 PM	Text Document	4 KB
ContinuaDeployTutorial.SetParameters.x...	17/06/2013 2:20 PM	XML Document	1 KB
ContinuaDeployTutorial.SourceManifest....	17/06/2013 2:20 PM	XML Document	1 KB
ContinuaDeployTutorial.zip	17/06/2013 2:20 PM	Compressed (zipp...	168 KB

5 items

Deploying our Build

Now that we can successfully build our configuration, lets move on the Part 4: Deploy Your Web App where you can learn about deploying your Web Application to your production server.

Part 4: Deploy Your Web App