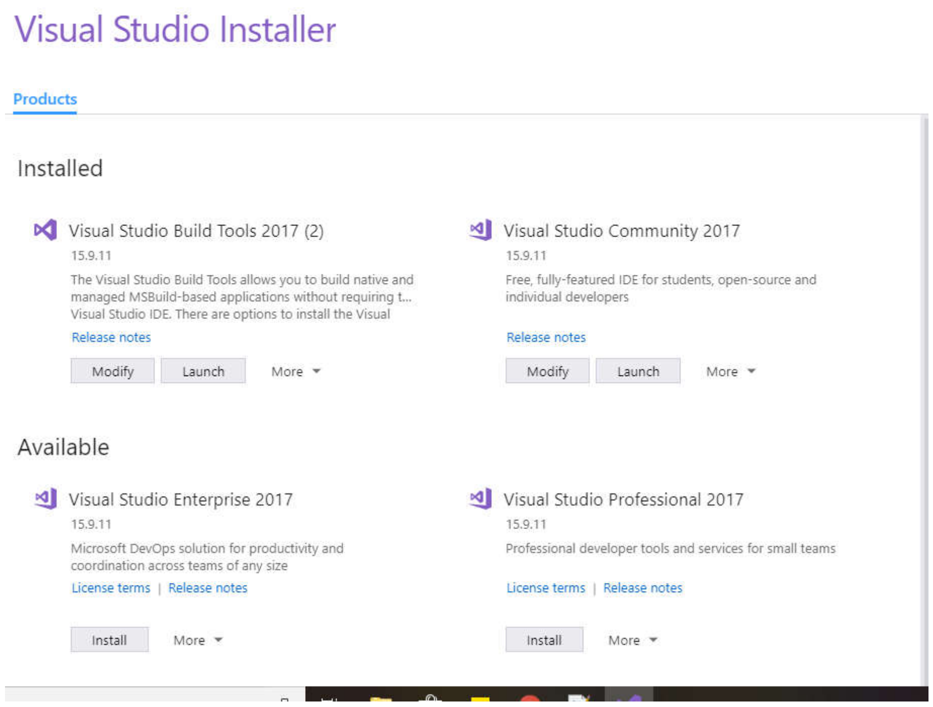
CI/CD for .NET MVC Using Jenkins

by Shivangi Garg · May. 13, 19 · DevOps Zone · Tutorial

Jenkins provides us many ways to set up a CI/CD environment for almost any code language and source code repository. It is an open source automation server which can be used to automate tasks related to building, testing, and delivering or deploying software. It is also an essential tool that aids DevOps teams, which are getting more popular due to the increasing amount of projects we manage at a given period, to ensure their high-quality deliverables. Jenkins has plugins that integrate with third-party source-controllers like Github.

Before we proceed with actual building and deployment, we need to make sure we have build tools installed on the machine.

This can be done through Visual Studio Build Tools, available from the Visual Studio Installer.


Visual Studio Installer

Products

Installed

Visual Studio Build Tools 2017 (2)
15.9.11
The Visual Studio Build Tools allows you to build native and managed MSBuild-based applications without requiring t... Visual Studio IDE. There are options to install the Visual
Release notes
Modify Launch More

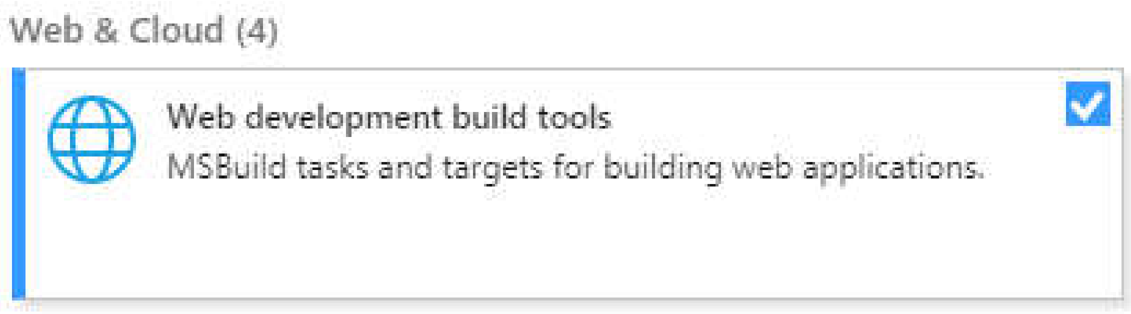
Visual Studio Community 2017
15.9.11
Free, fully-featured IDE for students, open-source and individual developers
Release notes
Modify Launch More

Available

Visual Studio Enterprise 2017
15.9.11
Microsoft DevOps solution for productivity and coordination across teams of any size
License terms | Release notes
Install More

Visual Studio Professional 2017
15.9.11
Professional developer tools and services for small teams
License terms | Release notes
Install More

Then we need to install build tools to build a MVC application. This can be done by clicking the Modify button for Visual Studio Build Tools and then selecting web development build tools.


Web & Cloud (4)

Web development build tools ✓
MSBuild tasks and targets for building web applications.

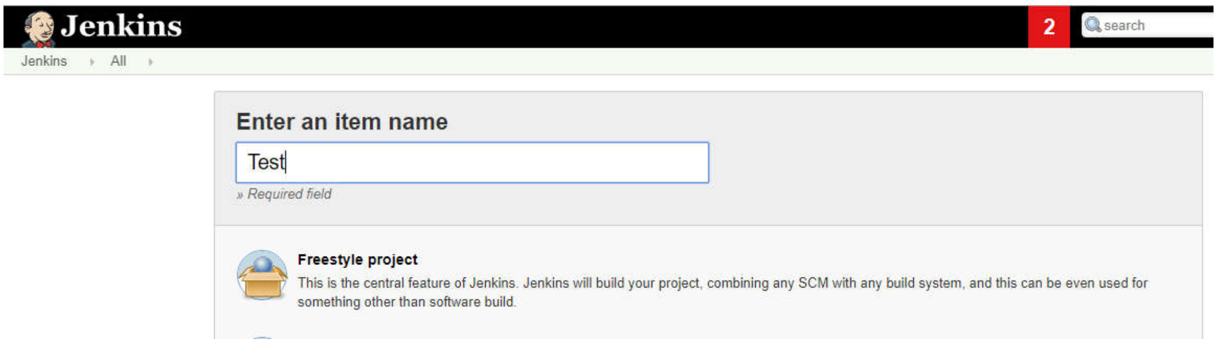
If we didn't install these build tools, the `msbuild` command would only work in the Developer Command Prompt.

We need to install plugins to use in the Jenkins.

- Go to Manage Plugins.

- Find and install the Github Plugin.
- Find and install the MsBuild Plugin.

I have implemented CI/CD using Jenkins. First of all, you need to download Jenkins and there are a number of ways to use it. I am using it as a windows service on the machine. After starting the Jenkins service, you need to add your application and add the configuration. First, click on "New Item." Then, give the application an Item Name and select the "Freestyle Project" option and then click "OK."



Enter an item name

Test

» Required field

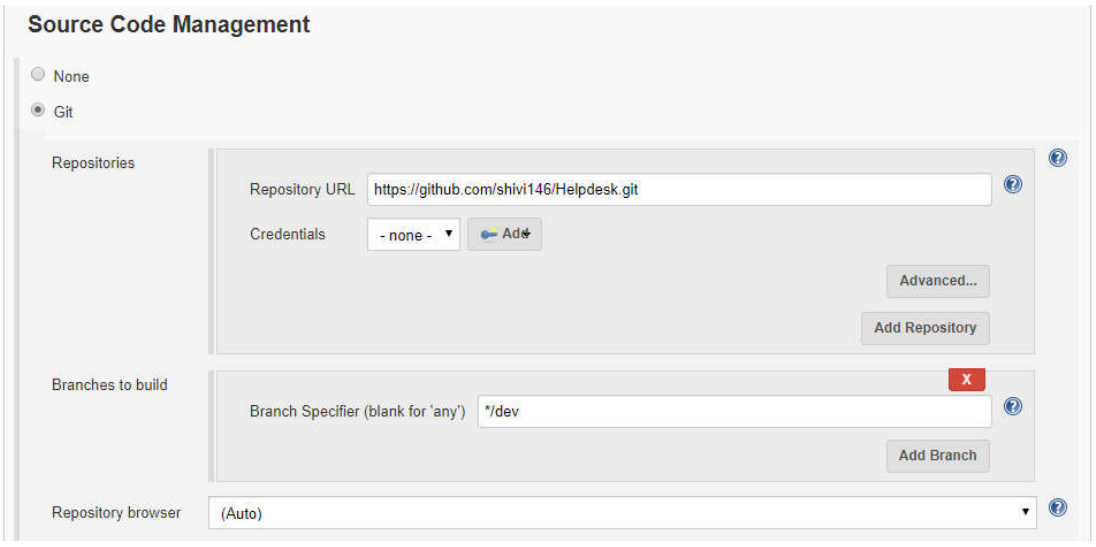
Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Now for the configuration.

In the General section, you can enter the description of the item and discard the old builds, it means it will automatically discard builds.

In the Source Code section, I used a Git repository. Here we have to provide the URL of the repository and the branch name from which you want to deploy the code.



Source Code Management

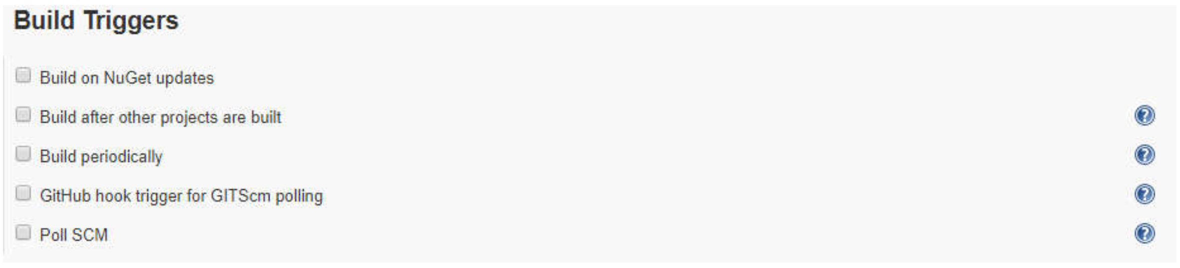
None [✓ Git]

Repositories
Repository URL: https://github.com/shivi146/Helpdesk.git
Credentials: - none -

Branches to build
Branch Specifier (blank for 'any'): */dev

Repository browser: (Auto)

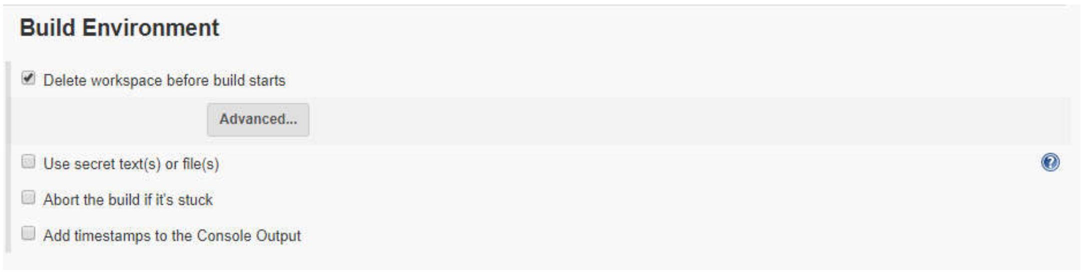
Build triggers can be added if you want the build to be triggered automatically.



Build Triggers

- ☐ Build on NuGet updates
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

In the build environment, we want to make sure we choose to clean the Jenkins workspace before the build starts.

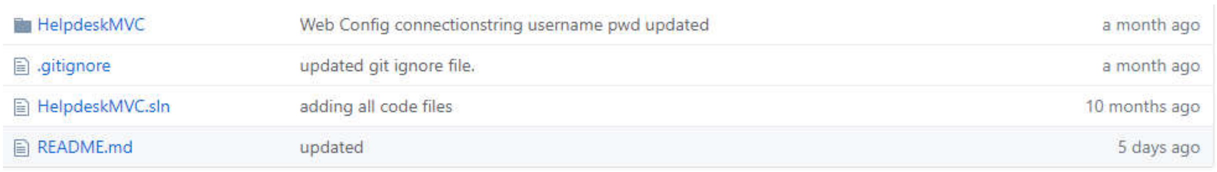

Build Environment

- [✓] Delete workspace before build starts
- [ ] Use secret text(s) or file(s)
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output

Nuget Resrore:

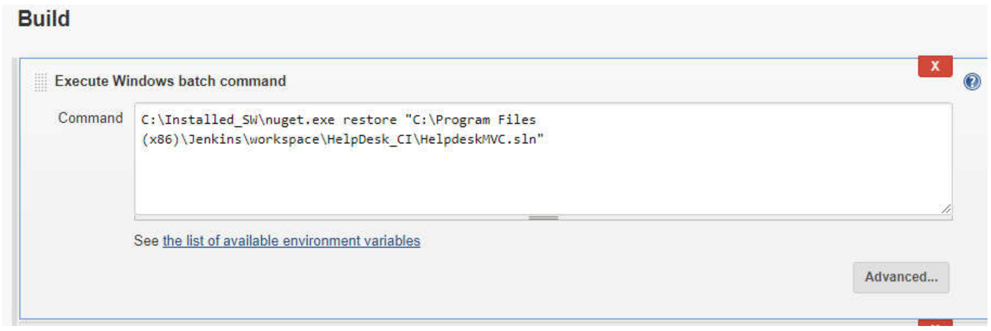
When we commit the code in the repository we don't commit the packages, so first we have to restore the NuGet packages.

As you can see, the code is committed without NuGet dependencies.



HelpdeskMVC	Web Config connectionstring username pwd updated	a month ago
.gitignore	updated git ignore file.	a month ago
HelpdeskMVC.sln	adding all code files	10 months ago
README.md	updated	5 days ago

The restore command from the NuGet CLI downloads and installs any packages missing from the packages folder.


Build

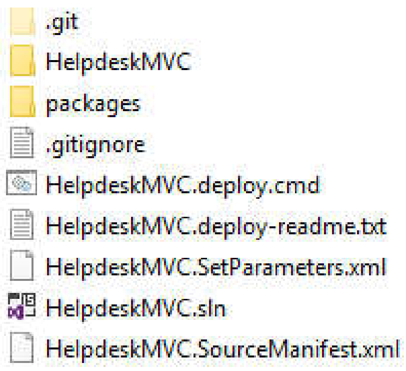
Execute Windows batch command

Command

```
C:\Installed_SW\nuget.exe restore "C:\Program Files
(x86)\Jenkins\workspace\HelpDesk_CI\HelpdeskMVC.sln"
```

See the list of available environment variables

Once the command is entered, you can see packages folder created with all NuGet dependencies.

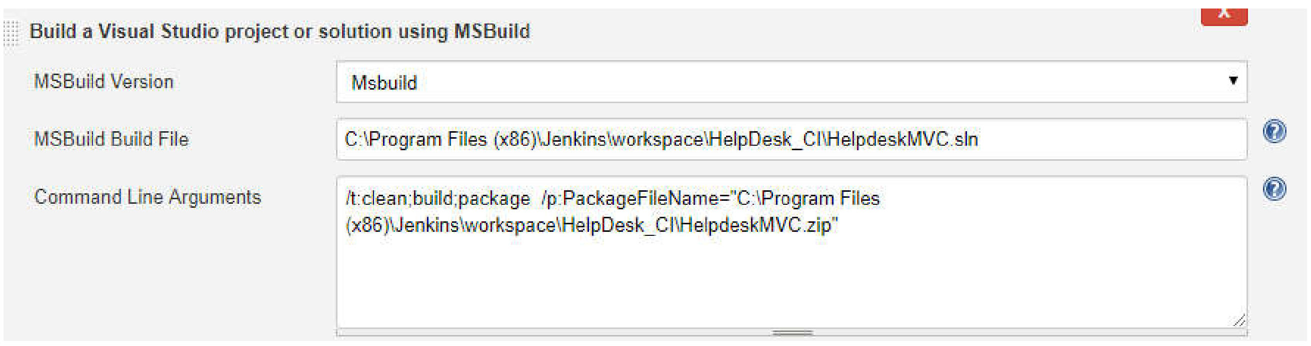


```
.git
HelpdeskMVC
packages
.gitignore
HelpdeskMVC.deploy.cmd
HelpdeskMVC.deploy-readme.txt
HelpdeskMVC.SetParameters.xml
HelpdeskMVC.sln
HelpdeskMVC.SourceManifest.xml
```

MSBuild:

Now we can actually execute MSBuild command.

This command requires that the MSBuild plugin be installed in Jenkins, path to the sln file and optionally we can also provide the `PackageFileName` attribute in the command line with path and package name.


Build a Visual Studio project or solution using MSBuild

MSBuild Version: Msbuild

MSBuild Build File: C:\Program Files (x86)\Jenkins\workspace\HelpDesk_CI\HelpdeskMVC.sln

Command Line Arguments:

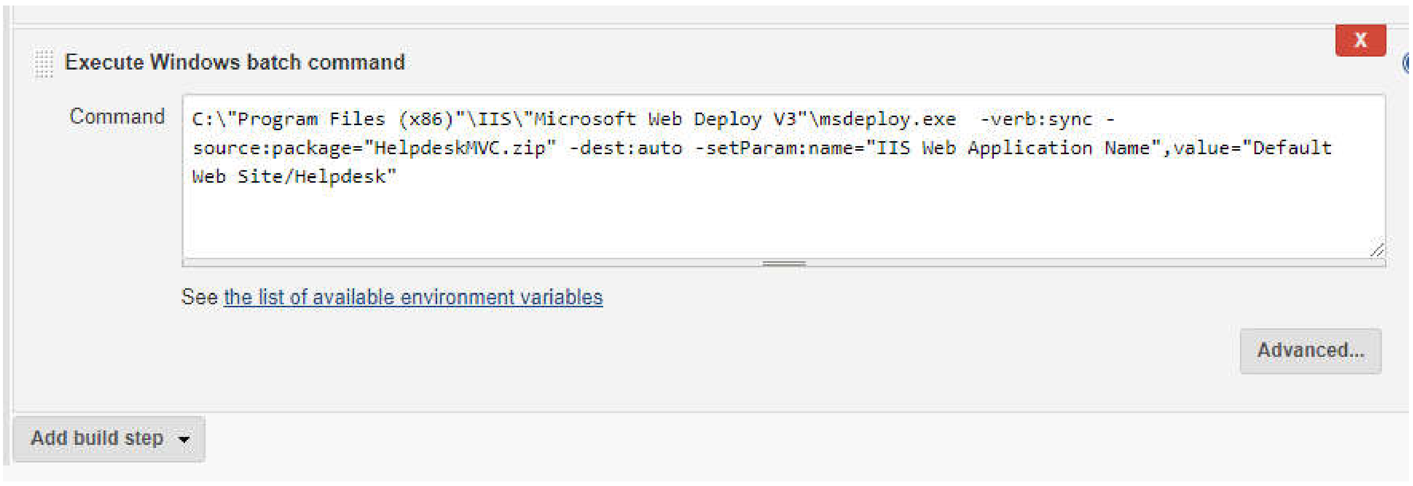
```
/t:clean;build;package  /p:PackageFileName="C:\Program Files
(x86)\Jenkins\workspace\HelpDesk_CI\HelpdeskMVC.zip"
```

Command:

/t:clean;build;package /p:PackageFileName="C:\Program Files (x86)\Jenkins\workspace\HelpDesk_CI\HelpdeskMVC.zip"

MSDeploy:

Msdeploy commad can be used to deploy zip created in previous step to be deployed to IIS.


Execute Windows batch command

Command:

```
C:\"Program Files (x86)"\IIS\"Microsoft Web Deploy V3"\msdeploy.exe  -verb:sync -
source:package="HelpdeskMVC.zip" -dest:auto -setParam:name="IIS Web Application Name",value="Default
Web Site/Helpdesk"
```

See the list of available environment variables

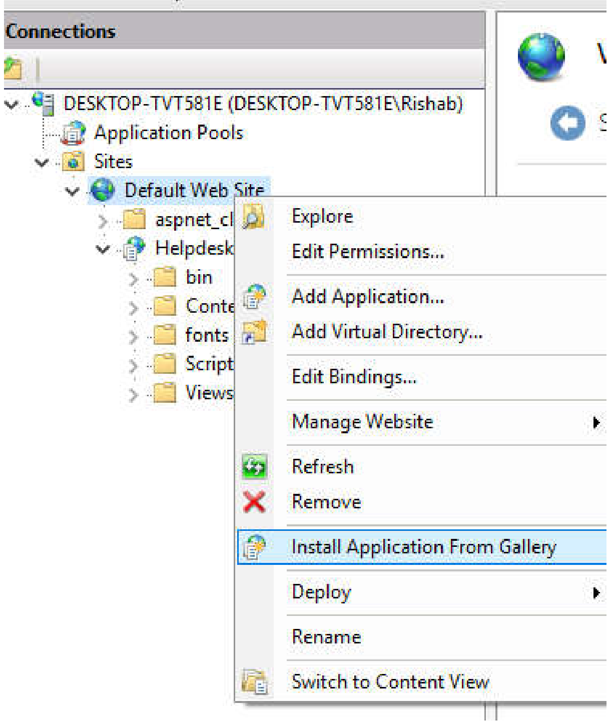
Command:

C:\"Program Files (x86)"\IIS\"Microsoft Web Deploy V3"\msdeploy.exe -verb:sync -source:package="HelpdeskMVC.zip" -dest:auto -setParam:name="IIS Web Application Name",value="Default Web Site/Helpdesk"

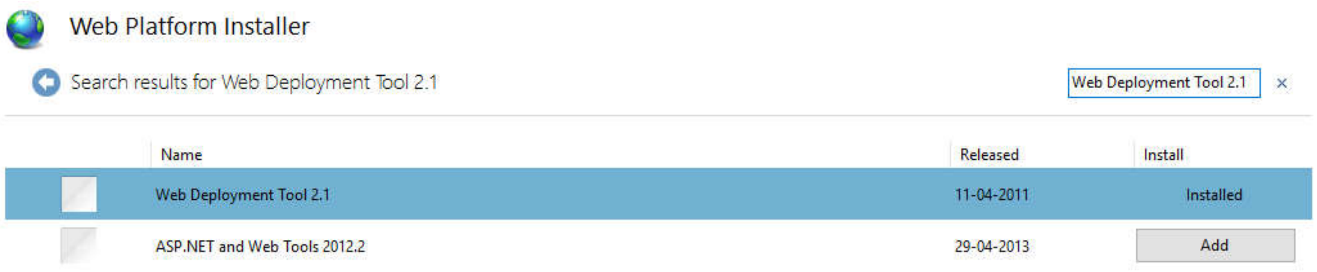
Note:

In order to deploy zip you need to install "**Web Deployment Tool 2.1**".

You can install this by right clicking on "**Default Web Site**" and then going to "**Install Application from Gallery**".



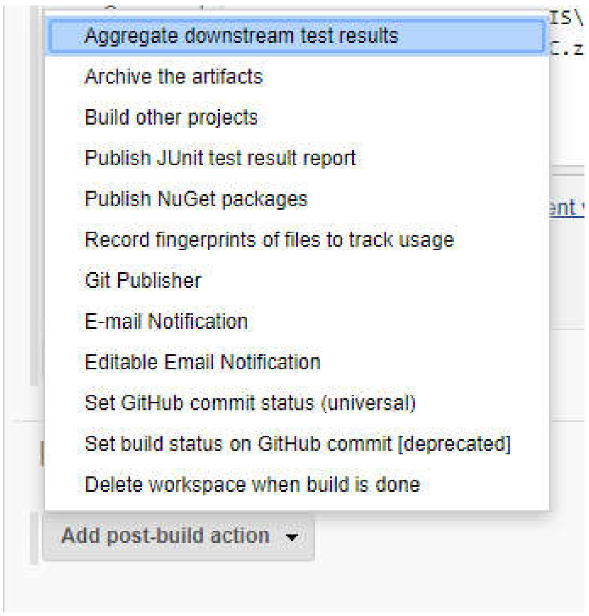
Then search for "Web Deployment Tool 2.1" in search box. In my case it is already installed.



	Name	Released	Install
	Web Deployment Tool 2.1	11-04-2011	Installed
	ASP.NET and Web Tools 2012.2	29-04-2013	Add

Post Build/Deploy actions:

You can add various post build actions like sending email notifications, archiveing the deloyale artifact etc


Aggregate downstream test results
Archive the artifacts
Build other projects
Publish JUnit test result report
Publish NuGet packages
Record fingerprints of files to track usage
Git Publisher
E-mail Notification
Editable Email Notification
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Delete workspace when build is done
Add post-build action

Conclusion:

I created a very basic CI/CD for my .Net MVC application. Jenkins gives us flexibility for adding more complex builds.