Pull

Checkout

**Git**

Local VCS

**GitHub**

Git DVCS

**Visual Studio DevopsCalc** IDE

Push

Commit

Fetch

VCS Trigger

**Teamcity**

CI Server

localhost:81111

**Octopus Deploy**

CD Server

**localhost:80** IIS Server

Development Environment Local Tentacle Agent

Process: Deploy to IIS Server

MSBuild

Email Notification

**someIPAdd** IIS Server

Production Environment

Azure Windows Server VM Tentacle Agent

**someIPAdd** IIS Server

Staging Environment AWS Windows Server VM Tentacle Agent

Octopus Deploy: Promote Release

Octopus Deploy: Deploy Release

Octopus Deploy: Create Release

Octopus Deploy: Push Packages

NUnit

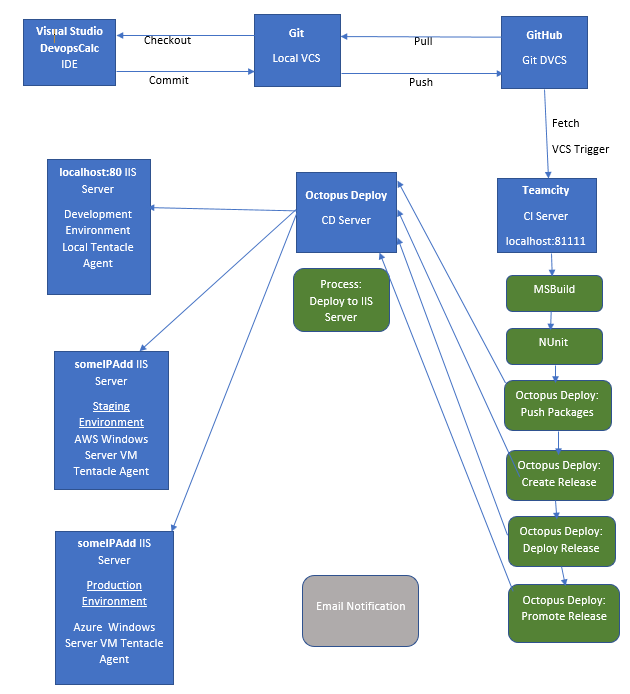


Table of Contents

[1. Software Required: 4](#_Toc508831167)

[2. Calculator ASP Application 5](#_Toc508831168)

[3. Version Control System: Git 6](#_Toc508831169)

[4. Distributed Version Control System: GitHub 9](#_Toc508831170)

[5. CI Server: Teamcity 11](#_Toc508831171)

[5.1 About Teamcity 11](#_Toc508831172)

[5.2 Teamcity logs 11](#_Toc508831173)

[5.3 Build 11](#_Toc508831174)

[5.4 MSBuild 12](#_Toc508831175)

[5.5 Nunit 13](#_Toc508831176)

[5.6 Octopus Deploy: Push packages 13](#_Toc508831177)

[5.7 Octopus Deploy: Create release 13](#_Toc508831178)

[5.8 Octopus Deploy: Deploy release 13](#_Toc508831179)

[5.9 Promote release 13](#_Toc508831180)

[5.10 Email & Visual Studio IDE notifiers 13](#_Toc508831181)

[5.11 Tutorials Used 13](#_Toc508831182)

[6. CD Server: Octopus Deploy 14](#_Toc508831183)

[**Octopus Deploy** 14](#_Toc508831184)

[**Machines Roles** 14](#_Toc508831185)

[**Environments** 14](#_Toc508831186)

[**Deployment Targets** 15](#_Toc508831187)

[**Lifecycle** 16](#_Toc508831188)

[**Channels** 16](#_Toc508831189)

[**Tenants** 17](#_Toc508831190)

[**Tentacle** 17](#_Toc508831191)

[**Variables** 17](#_Toc508831192)

[**Deployment Process/Step** 18](#_Toc508831193)

[**Deploy an IIS website process** 19](#_Toc508831194)

[7. Issues 21](#_Toc508831195)

[References 22](#_Toc508831196)

# Software Required:

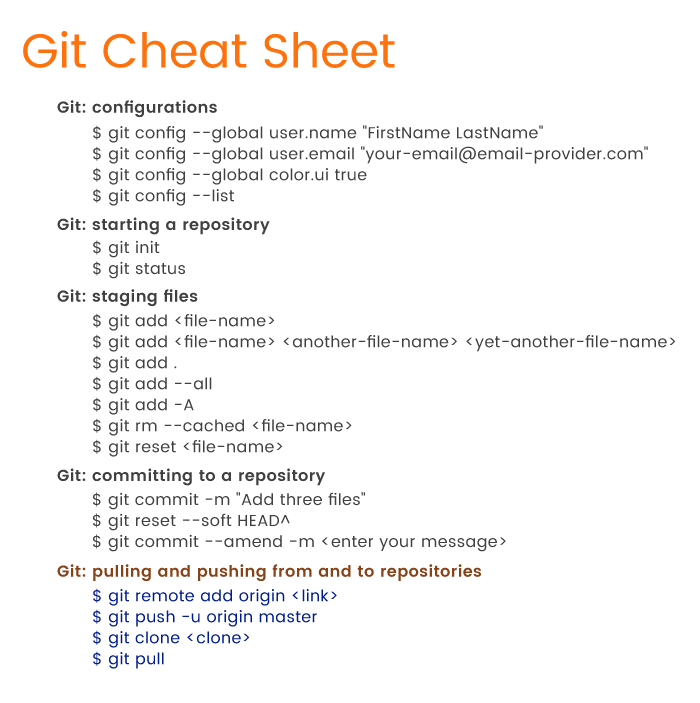
* SQL Server Express
* Octopus Deploy Server Manager
* Octopus Deploy Tentacle Manager
* Teamcity Server
* Teamcity Build Agent
* Octopus Deploy Plugin - install zip via web UI (Teamcity zip file)
* Servers/VMs
  + Windows Server 2016 - AWS & Azure
    - Network Security Groups/Rules / Firewall- Inbound
      * RDP - 3389
      * HTTP - 80
      * HTTPS - 443
      * Listening Tentacle - 10933
      * Polling Tentacle - 10943
* Nuget Package Explorer (If you’re not using Teamcity to pack solutions)
* Visual Studio
* Microsoft Build Tools/MSBuild
* Nuget Packages
  + Octopack
  + NUnit
  + NUnit Console
  + NUnit Console Runner

# Calculator ASP Application

# Version Control System: Git

Git was used as a version control system. Git is a free open source version control system (Git, 2018). Git Bash was used as the CLI for using Git.

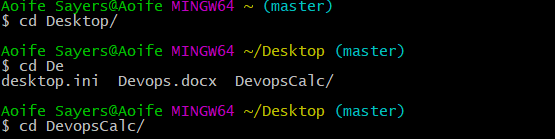
Some of the commands I used were:

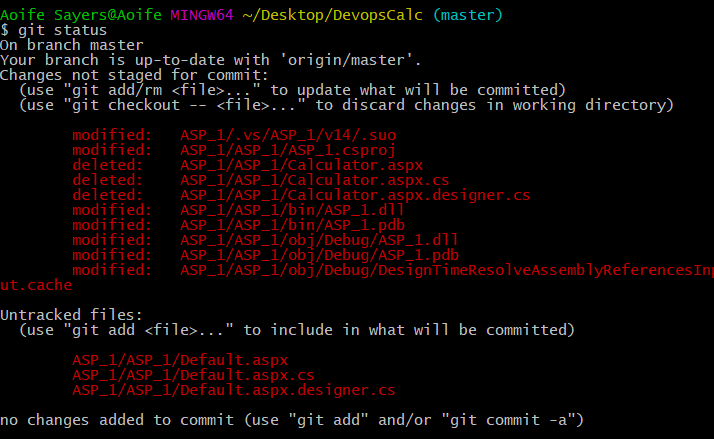


Git Cheat Sheet (A., 2018)

The git repo was automatically initialised by Visual Studio. To create a new repository in git, the developer should change directory to the folder to be initialised as a git repository. The command “git init” should then be issued to initialise a git repository.

To commit to the Visual Studio project I changed directory “cd” in to Desktop/DevopsCalc/

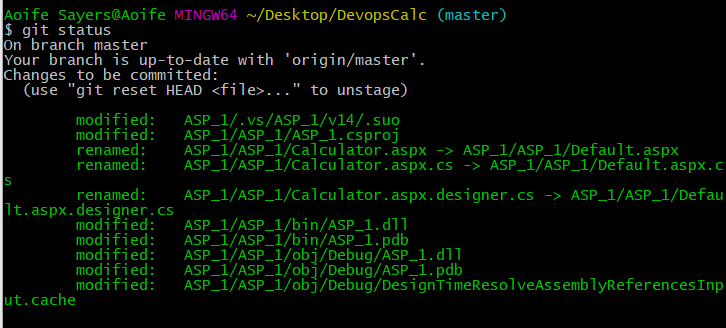


The “git status” command to list the untracked changes of the project since the last commit. Red text denotes unstaged files via Git. Green files denote staged files in Git

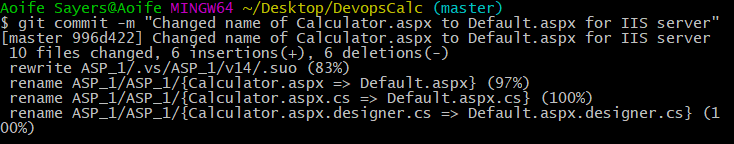
The “git add .” command was used to stage new and modified files via git bash.



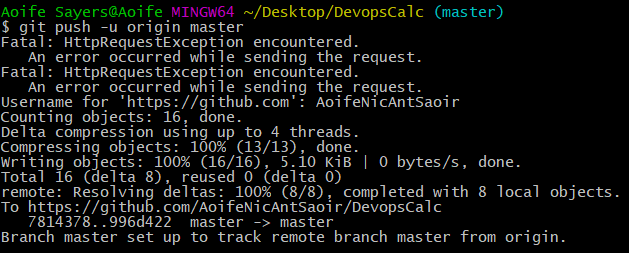
Git status was used again to verify the changes are now staged in Git. The files previously unstaged are now staged as they are coloured green.



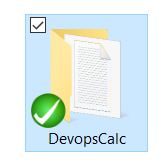
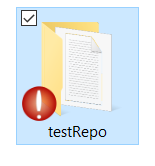
The command “git commit -m “commit message” is now tracked as a revision in source control with a unique ID (SHA or hash) and message of the changes made of the previous revision.



The local commit is now pushed up to a hosted Git repository on Github.com using the “git push -u origin master” command.



Tortoise Git was used to verify in a GUI manner to verify the correct files were staged. Tortise Git denotes all modifications have been committed and tracked in version control using a “tick” as seen below. Alternatively, repositories with modification not committed in version control are denoted by a red “!”.

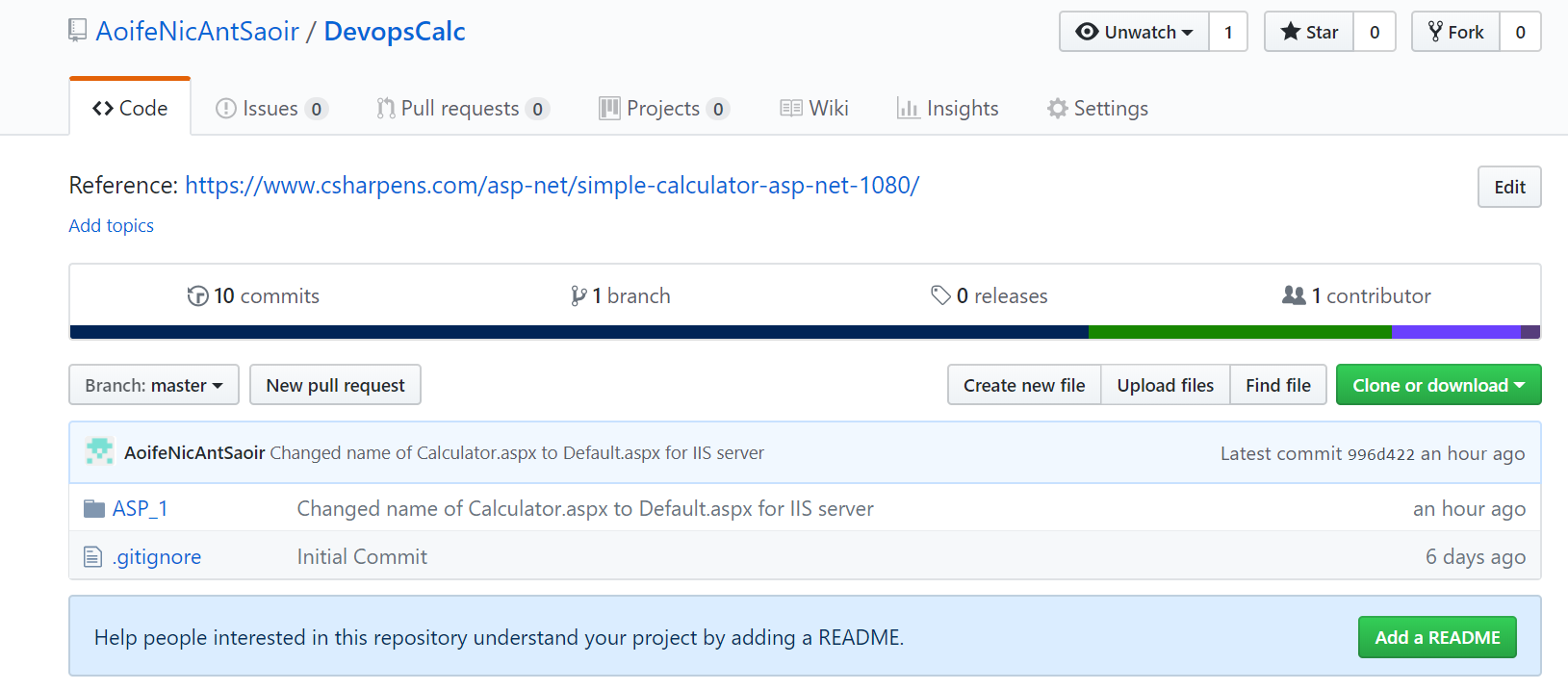
Tutorials Used

I used the Git documentation as a tutorial for Git located here: <https://git-scm.com/docs/gittutorial>

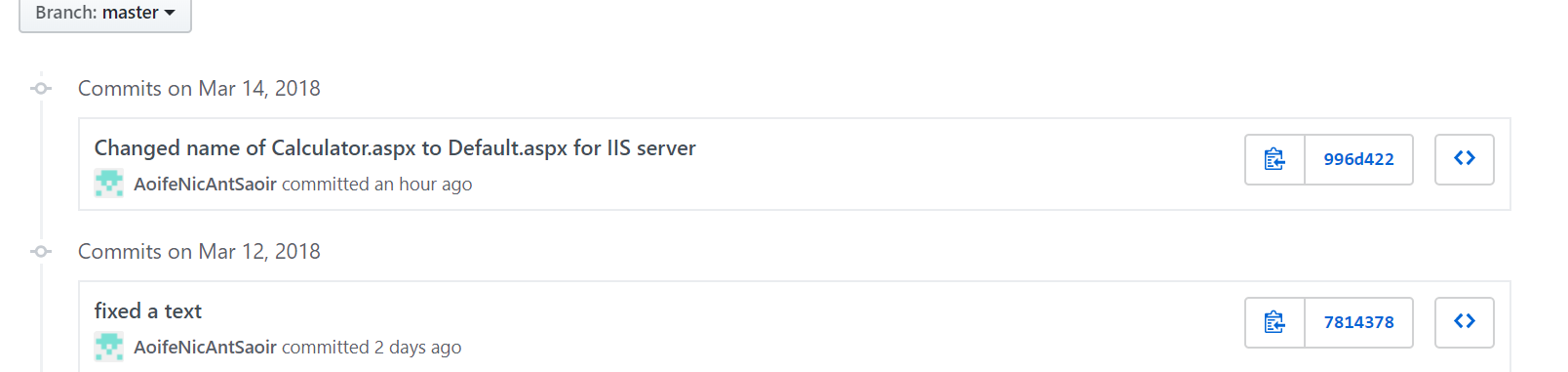
# Distributed Version Control System: GitHub

Github is a distributed and hosted Git version control system. Github encourages the sharing of code and working on building of software with other developers worldwide. Github may be used as a repository in which a CI Server will fetch code from for building and testing etc. applications. Teamcity, Microsoft Team Services and Jenkins use Github to fetch code from for their build process.

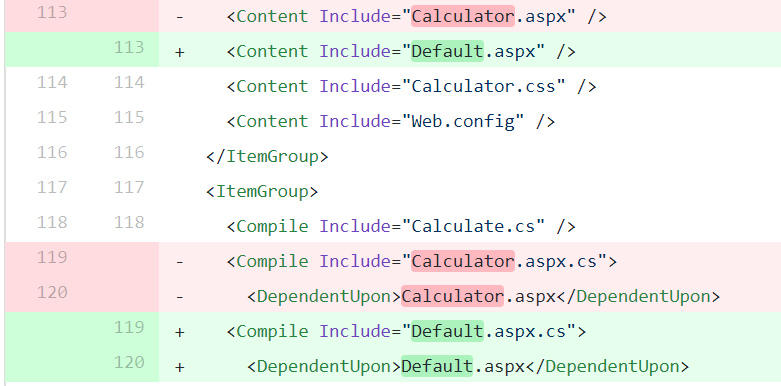
The Github repository for this project is located at the following URL: <https://github.com/AoifeNicAntSaoir/DevopsCalc>



The Commits made to Github can be seen in the image below. The latest commit on March 14 2018 corresponds to the commit made on Git and then pushed to Github as seen below.



The latest commit is compared with the most recent commit when you click on the individual commits. The red denotes the removed or altered code and the green denotes the most recent modifications/additions to the code. For instance in the example below, the .aspx file was renamed from Calculator.aspx to Default.aspx.



# CI Server: Teamcity

## About Teamcity

Teamcity is running on localhost:8111 – port 8111 as it was originally running on port 80 and clashing with Octopus Deploy

Agents – I have one agent running on localhost:9090

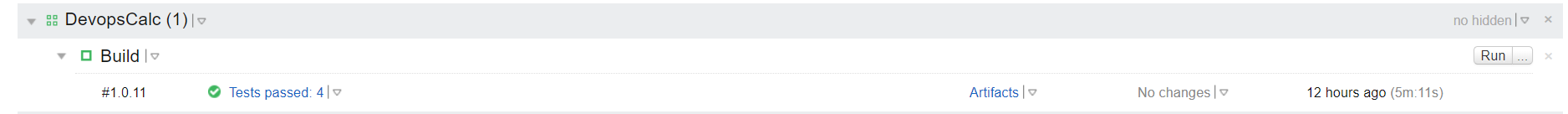
Project created called DevopsCalc(1)

## Teamcity logs

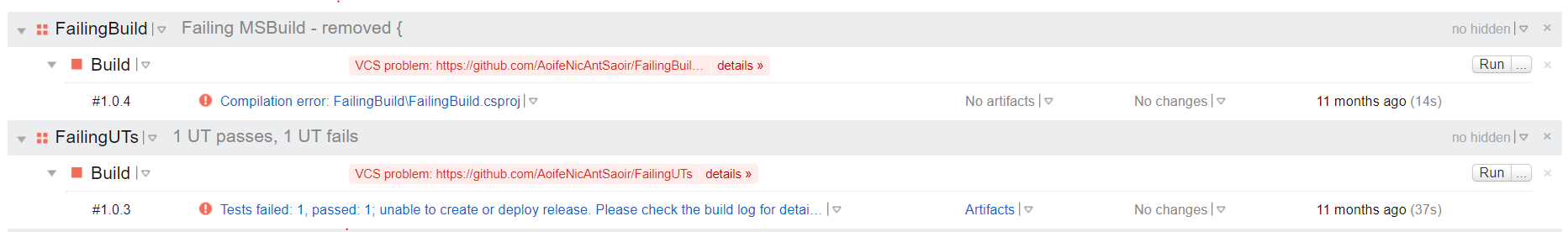
Team City Build Agent

## Build

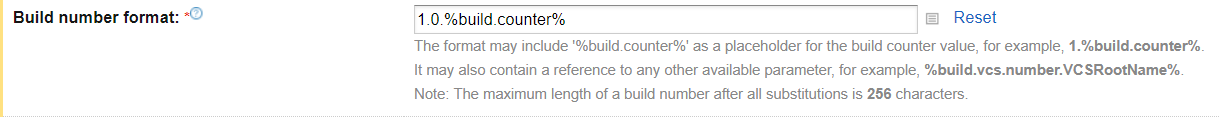
Successful Build

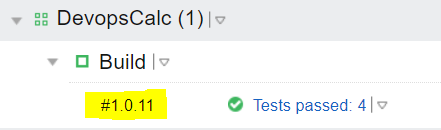


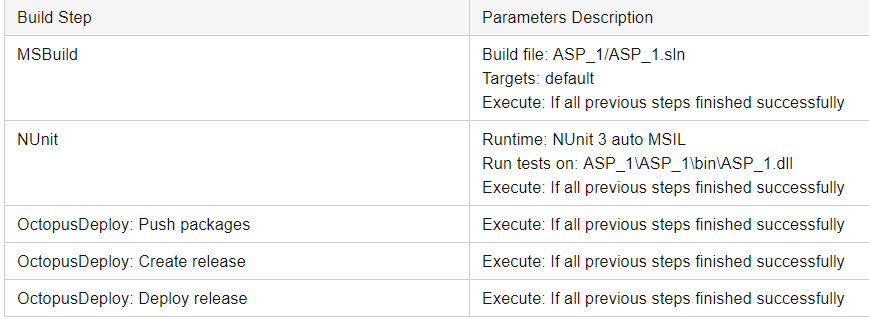
Unsuccessful Build



Build Number in the format of Major.Minor.Patch





Build Steps 

The various build steps in Teamcity were listed above. Each subsequent step executes only if the previous steps finish successfully.

Autodetected Build Step for Visual Studio would not build – due to Visual Studio being in Debug/Release Mode so MsBuild was used as a result.

## MSBuild

Command Line Parameters

/p:Configuration=Release

/p:Platform="Any CPU"

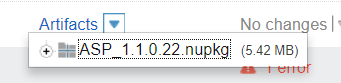
/p:OctoPackPublishPackagesToTeamCity=true \*

\*Need to enable the Nuget feed under Administration > Integrations > Nuget Feed

teamcity public guest feed & Teamcity private feed

http://localhost/guestAuth/app/nuget/v1/FeedService.svc/

Octopack: creates nuget package of the project as an arifact with files required. Discards .sln, .csproj, web.release.config, web.debug.config, obj and Properties folders. Keeps web.config/app.config and bin folder with .dll. Ready for deployment



## Nunit

## Octopus Deploy: Push packages

## Octopus Deploy: Create release

## Octopus Deploy: Deploy release

## Promote release

## Email & Visual Studio IDE notifiers

## Tutorials Used

<https://confluence.jetbrains.com/display/TCD10/TeamCity+Documentation>

To fix team city build agent disconnected issue <https://stackoverflow.com/questions/12279081/teamcity-build-agent-disconnected> - go through log files

MSBuild command <https://msdn.microsoft.com/en-us/library/ms164311.aspx>

MsBuild missing parameters on Teamcity http://blog.chudinov.net/how-to-build-and-deploy-web-deployment-package-using-msbuild/

NUnit <https://www.codeproject.com/Articles/178635/Unit-Testing-Using-NUnit>

# CD Server: Octopus Deploy

## **Octopus Deploy**

* Distributing applications to all the remote machines, securely
* Environment-specific configuration, like connection strings
* Configuring IIS sites and installing Windows Services
* Doing all of the above across many machines in parallel
* Popular .NET deployment tool with integration for CI servers Teamcity (Jetbrains), Team Foundation Server (Microsoft)

45 day free trial

Community Version – Free discontinued from February 15 – allow up to 5 projects

Professional -  20 users, 20 projects and 20 target machines

Team - 60 users, 60 projects and 60 target machines.

Enterprise Unlimited users, projects and target machines.

High Availability - Unlimited users, projects, target machines and multiple nodes

Command line - octo.exe

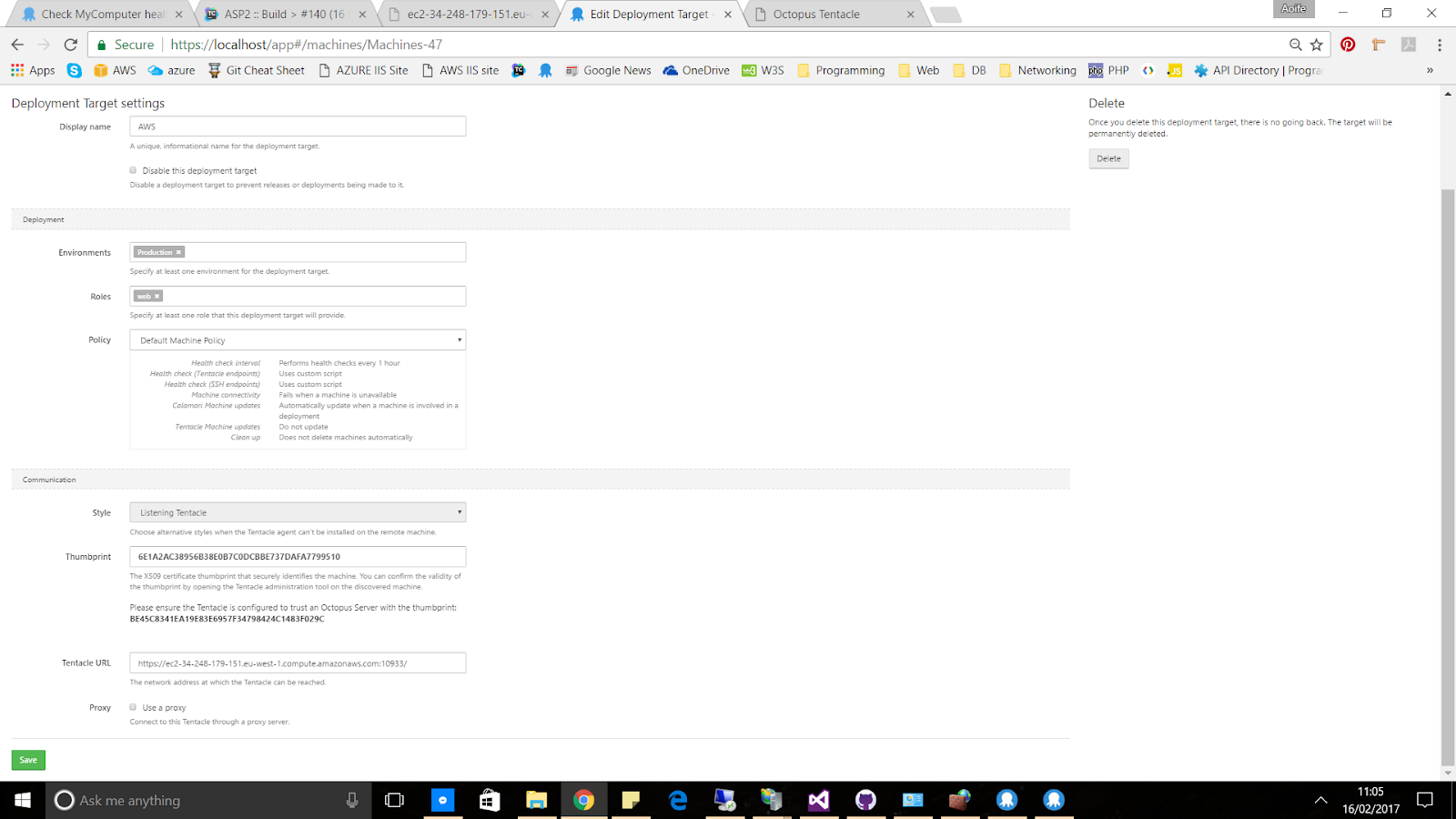
### **Machines Roles**

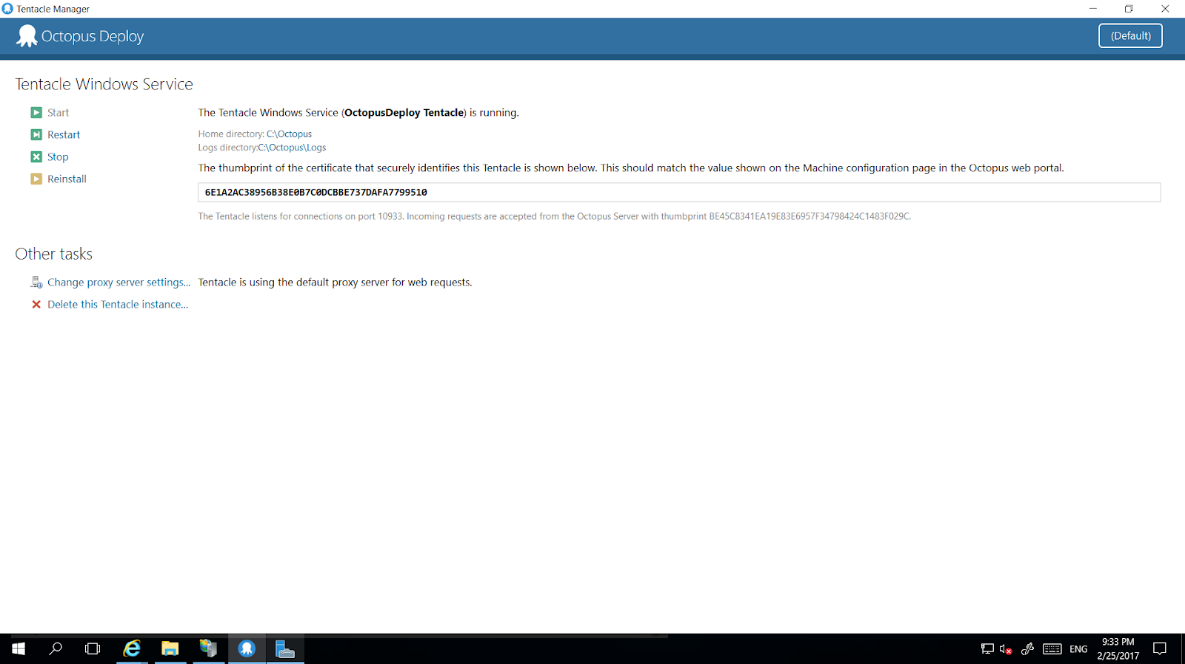
* Each of these servers, whether physical or virtual machines, would be a machine within Octopus.

### **Environments**

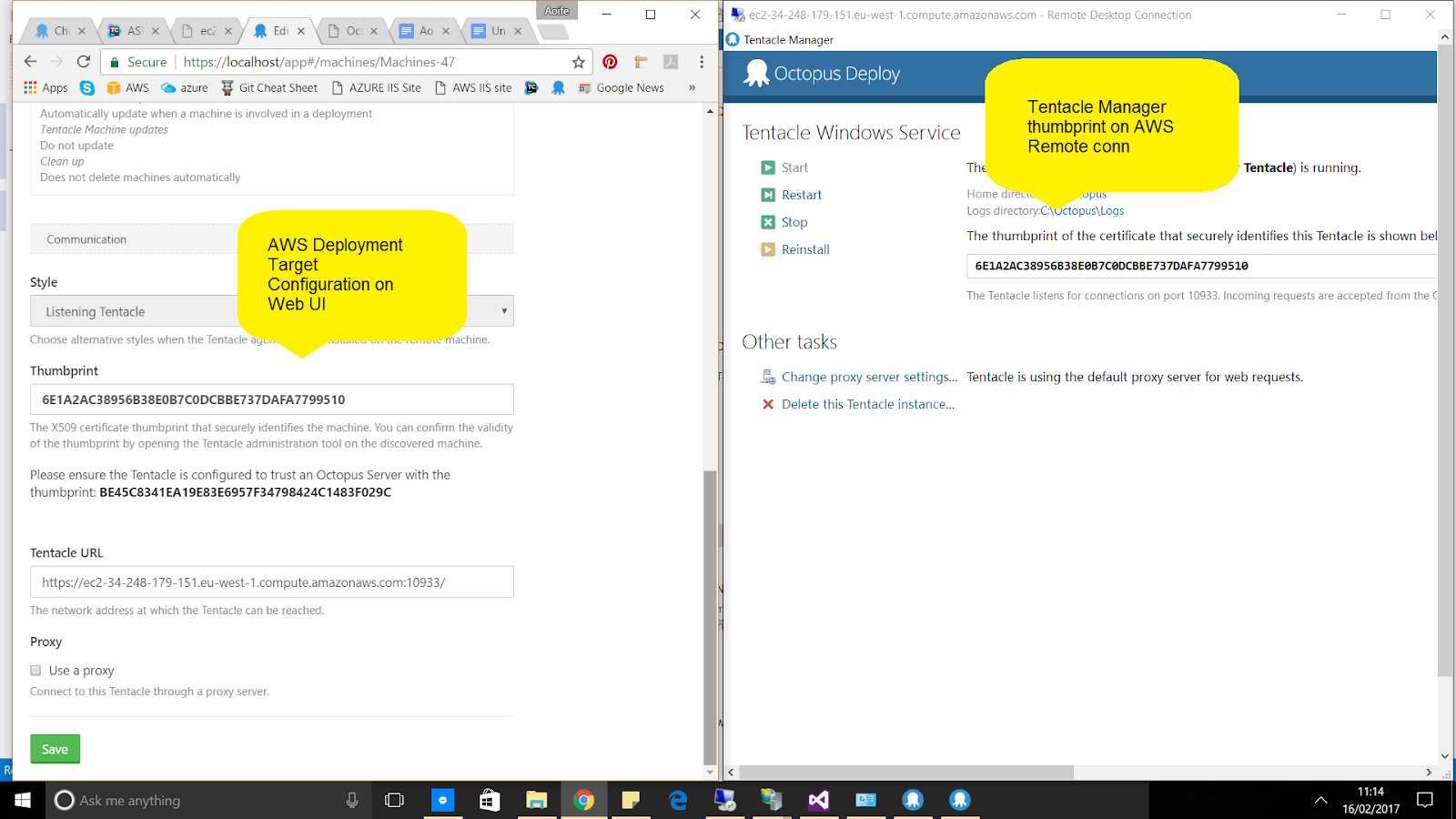
* An environment is made up of multiple machines, and each machine is tagged with a set of roles
* A group of these machines that are deployed to at the same time is called an environment.
* Deployment Targets

### **Deployment Targets**

****

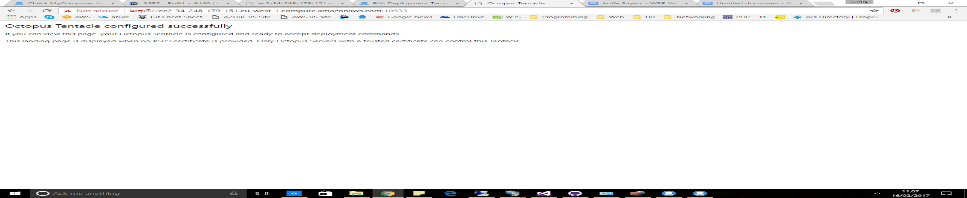
****

Thumbprint:



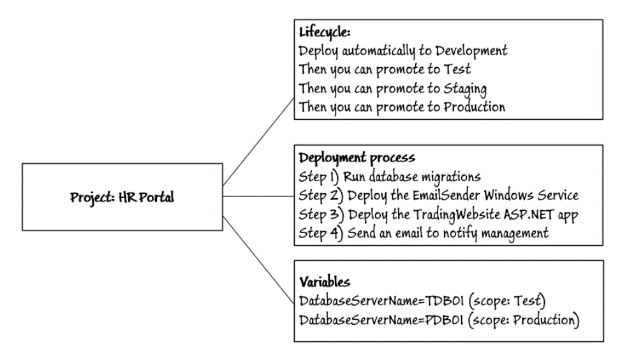
Verify the deploy target is configured successfully

or http://localhost:10933

****

### **Lifecycle**

* List of different environment involved in the deployment process



### **Channels**

Experimental branch of code, perhaps to introduce a new feature, or an entirely new version of your software.

### **Tenants**

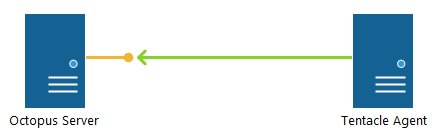
Over time your software may become so successful that you on-sell it to some external customers, and due to the way the software is architected, you need to deploy a copy of the software once per customer. You could achieve this in Octopus by creating an environment-per-customer, or even a project-per-customer, but this leads to duplication and unnecessary complexity. In Octopus 3.4 we introduced the concept of [tenants](https://octopus.com/docs/key-concepts/tenants) that you can manage alongside your existing projects and environments.

### **Tentacle**

Run as Windows services - Tentacle manager installed on deployment targets

Lightweight agent service/job runner

* installed onto target deployment machines.
* Connection secured by public & private key  - no passwords
* Listening Tentacles
  + 
  + Tentacle plays the role of server and Octopus as the client:
  + Octopus establishes the HTTPS connection with the Tentacle
  + The Tentacle presents its certificate as the server certificate allowing Octopus to verify the identity of the Tentacle
  + Octopus presents its certificate as a client certificate so the Tentacle can verify the identity of Octopus
  + Once the identity of the Octopus and Tentacle have been established the connection is held open and Octopus will start issuing commands to the Tentacle
* Polling Tentacles



Octopus plays the role of server and Tentacle as the client:

1. The Tentacle establishes the HTTPS connection with Octopus
2. Octopus presents its certificate as the server certificate allowing the Tentacle to verify the identity of Octopus
3. The Tentacle presents its certificate as a client certificate so Octopus can verify the identity of the Tentacle
4. Once the identity of the Octopus and Tentacle have been established the connection is held open and Octopus will start issuing commands to the Tentacle

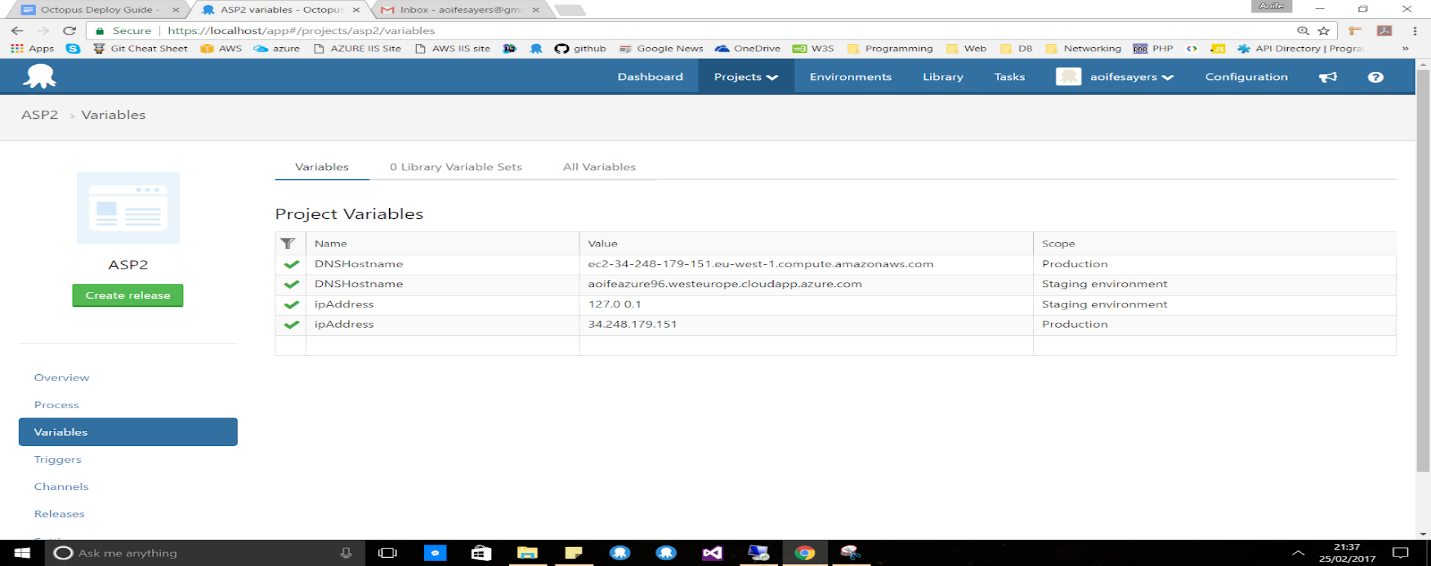
### **Variables**

[Use variables to scope different settings to different environments](https://octopus.com/docs/deploying-applications/variables)

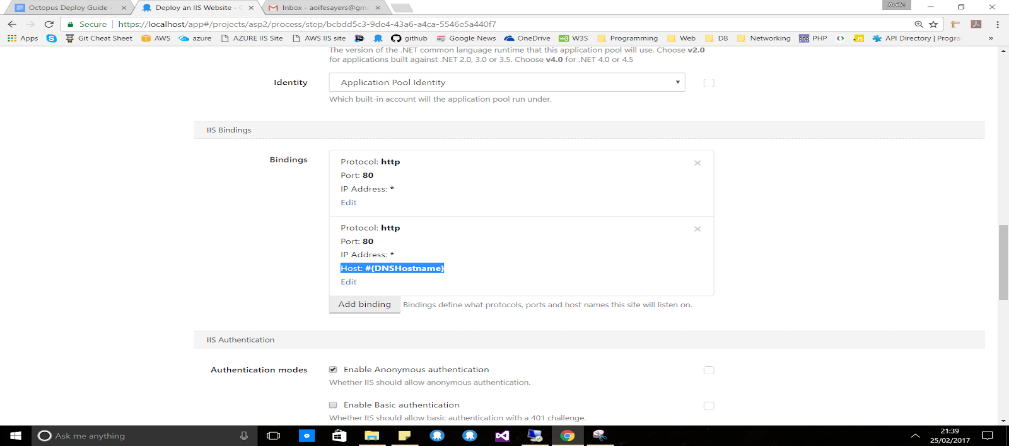
Octostache format #{myVar}

Octopus automatically takes care of:

* Replacing application settings
* Swapping connection strings
* Running environment-specific configuration transforms
* Configuring IIS application pools and web sites
* Installing and updating Windows Services



Using variables in a step

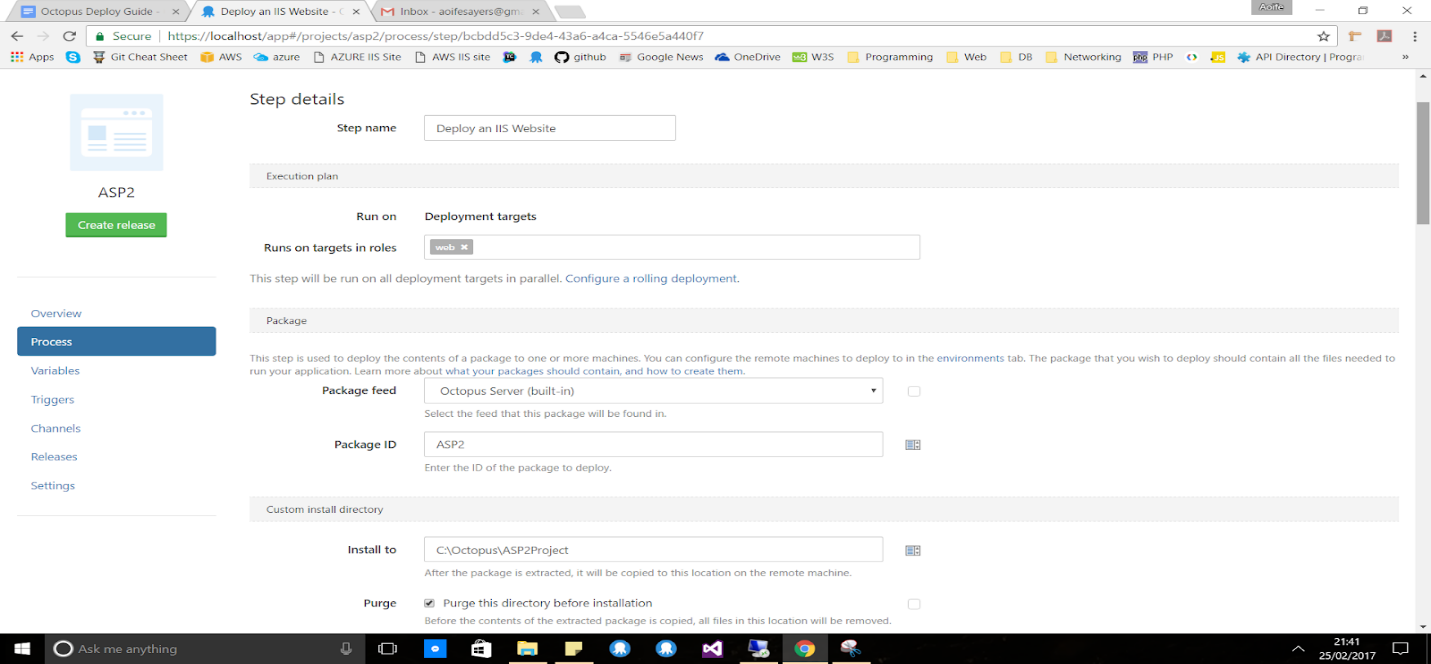


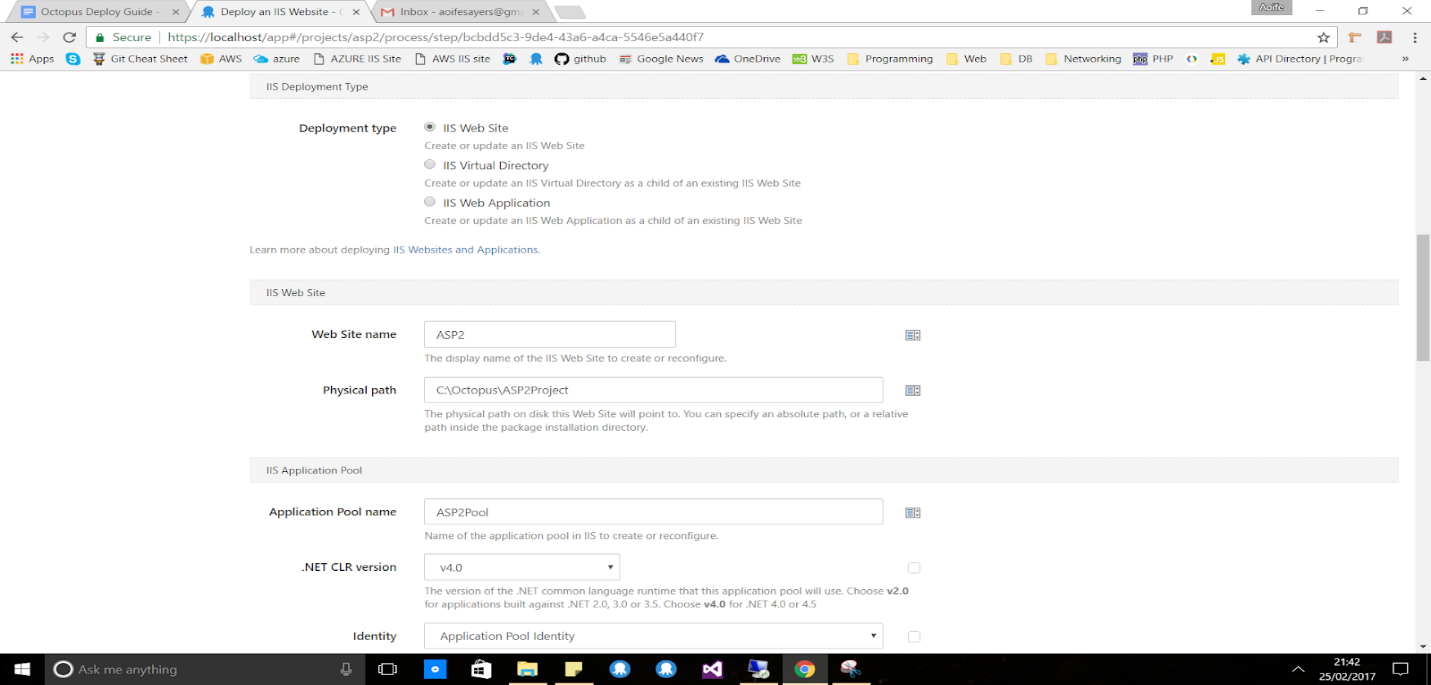
Also Octopus Deploy is able to configure <connectionStrings> with variables & transform the files during the deployment process

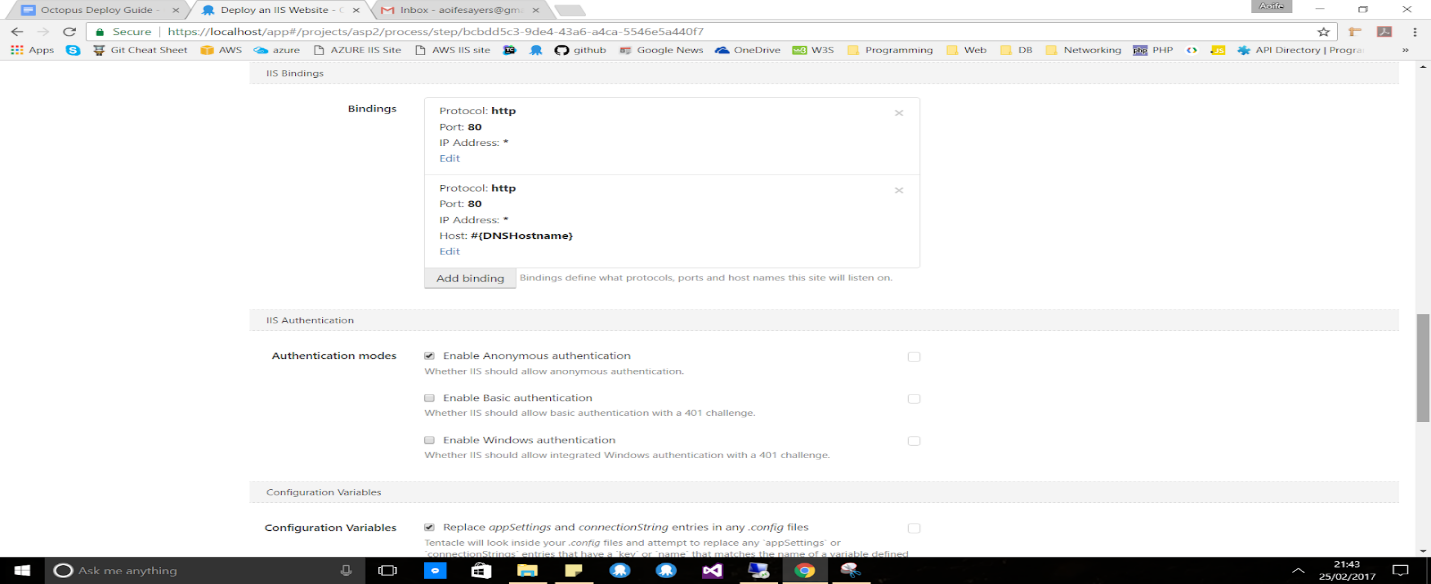
### **Deployment Process/Step**

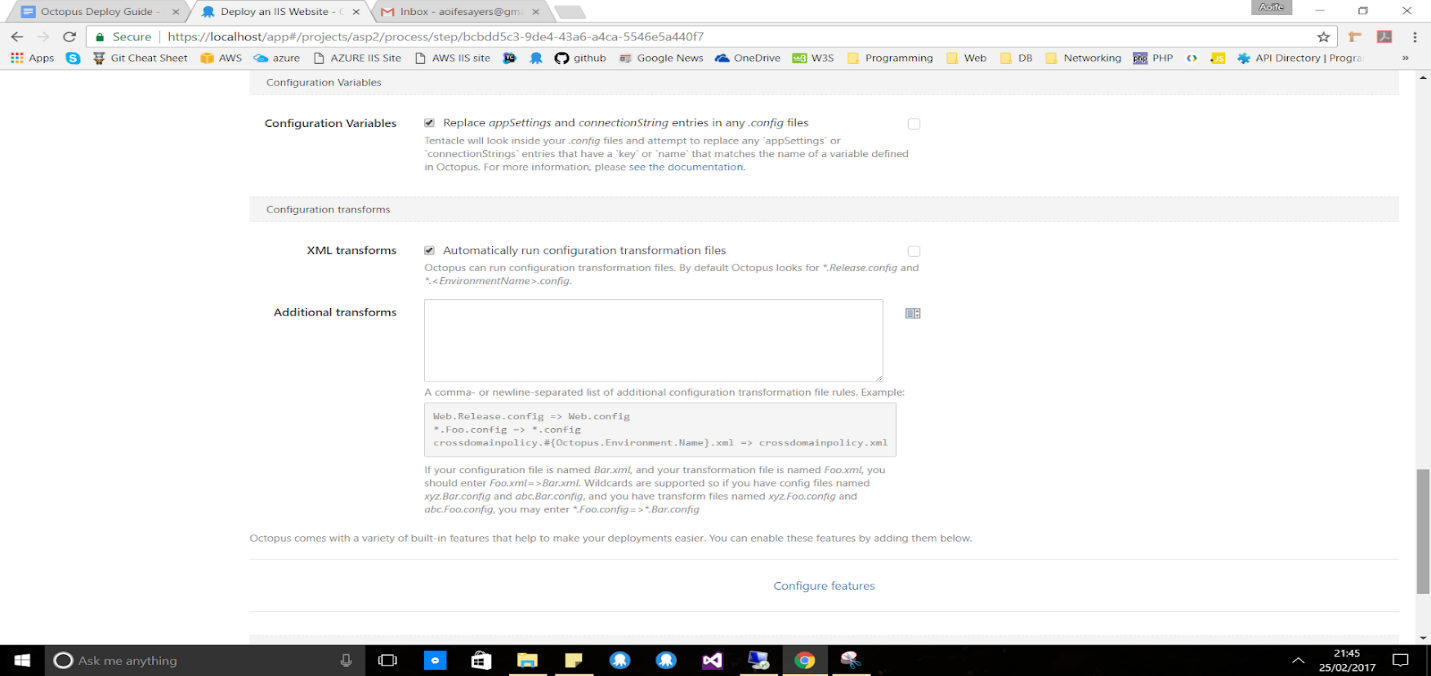
* Specifies the steps that need to happen in a given order during the deployment
* Manual Intervention / Guided Failure Mode
  + Pause before failing
  + Ignore, stop, retry - add comments
* Installed Step Templates
  + IIS Website
  + Windows Service
  + Package
  + Run Powershell, C#, F# or bash scripts
  + Deploy Azure
    - Web App
    - Services
    - Powershell scripts
    - Resource Group
  + Send email - SMTP
* Community Step Templates
  + Run NUnit Tests from assemblies
  + AWS
  + Git
  + Splunk
  + SQL Server
* Install own step template

### **Deploy an IIS website process**











# Issues

Clashing ports – Teamcity Build agent & Octopus Deploy Server running on the same ports

Disconnected build agent – read through log files

Visual Studio & MsBuild

MsBuild missing parameters on Teamcity http://blog.chudinov.net/how-to-build-and-deploy-web-deployment-package-using-msbuild/

# References

A., S., 2018. *Most Basic Git Commands with Examples – a Welcome Introduction to Git.* [Online]   
Available at: https://rubygarage.org/blog/most-basic-git-commands-with-examples  
[Accessed 14 March 2018].

Git, 2018. *Git.* [Online]   
Available at: https://git-scm.com/  
[Accessed 14 March 2018].