Team CrowSoft - Documentation

Team CrowSoft

LYIT - MSc

5/1/2019

Team Members

NAME	L NUMBER
BHARATHI GADHIRAJU	L00144819
MARY WALSH MCGINTY	L00113835
CHARLES AYLWARD	L00131535
COLIN KENNY	L00086074
MATTHEW MC COLGAN	L00083863
MICHAEL MCFADDEN	L00105787
JOJI THOKLA	L00144702
LIAM WHORRISKEY	L00113360

Updated: 02/04/2019
Created by: Charles Aylward

CrowSoft's Processes, Team and Environments (Version 1.1)

Contents

Introduction

As we are starting as a brand new Scrum team, to successfully deliver the new CrowSoft product for developing an online system for business intelligence creating building costs and analysis according to CrowSoft's business requirements, it will be vitally important to define our Agile and DevOps processes, setting up the team (including roles), how we will work together, what tools we need to set up our environment, to collaborate, and finally the architecture and design for our solution.

In this document, you will find an outline of the proposed Scrum process, roles required per process item, artifacts required and DevOps process and environments required

The outcome of our first spring planning meeting was as follows:

- Define overall processes for CI/CD, including what environments is required for the new CrowSoft Solution
- Define and select tools for Continuous Integration
- Define and select tools for Continuous Deployment
- Branching Structure on Github
- ***NOTE: The team was going to add more stories to this list on Jira over the weekend before the first Sprint Kick off on Monday, 4th March 2019.***

Updated: 02/04/2019
Created by: Charles Aylward

Agile Process using Scrum

Sprint Planning

- Hosted by Scrum Master (Alternate each week)
- Select highest priority items on Product Backlog and team turns item into Sprint Backlog
- Estimate Sprint Backlog in hours
- Work breakdown
- Declare sprint goal

Daily Scrum

- Hosted by Scrum Master (Twice a week)
- Roles required
    - PO – Ruth Lennon
    - Scrum Master (alternate)
    - Development Team (As per Slack List for CrowSoft):
        - UX, Design, Development, QA/Testing, Delivery
- 30 Mins Standup – Monday 9:30pm, Friday 9:30am
- Not for problem solving (Let's use Slack for problem solving)
- 3 Question:
    - What did you do?
    - What will you do?
    - What's in your way?

Sprint Reviews

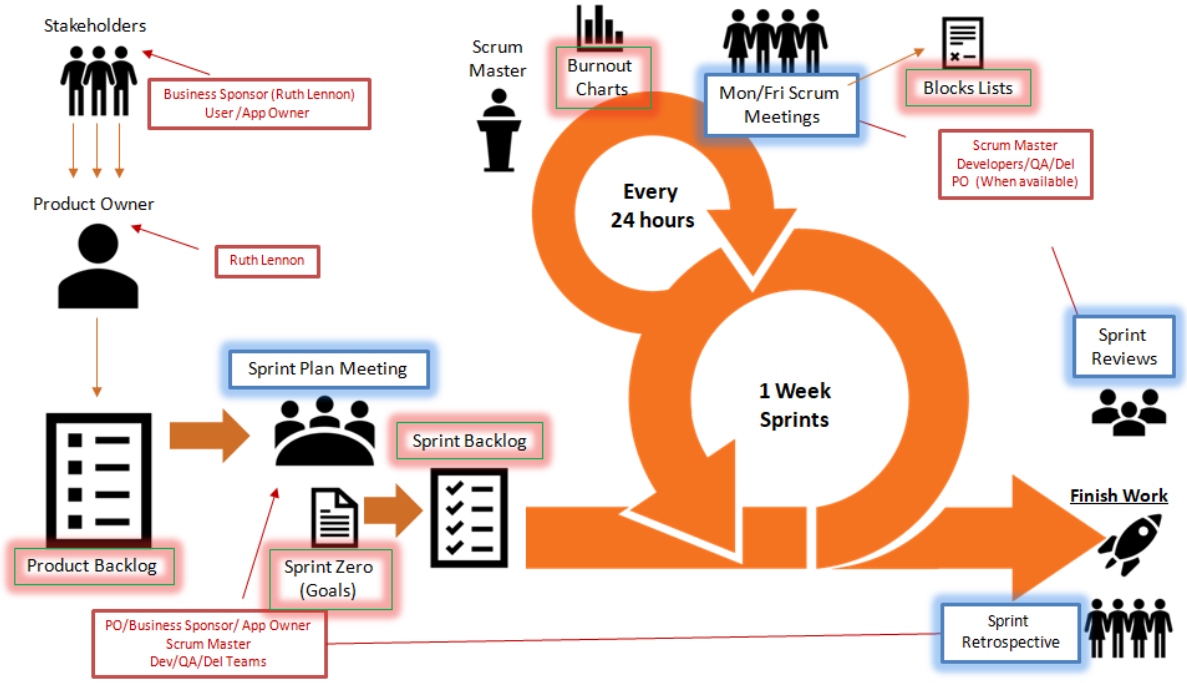
- Hosted by Scrum Master
- After each sprint
- Accomplishments
- Whole Team present
- Demo shippable product/features

Sprint Retrospective

- Hosted by Scrum Master
- Discuss "Start Doing", "Continue Doing", "Stop Doing"

Updated: 02/04/2019
Created by: Charles Aylward

Figure 1 - Scrum Process



Agile (Scrum) Key Artifacts

Product Backlog

- User Stories (Epics in Jira)
- Desired Work
- Prioritized by Product Owner

Sprint Zero (Goal)

- Hight Level context Diagram (CrowSoft Solution)
- Summary of work in sprint
- Declared by Product Owner
- Accepted by Team

Sprint Backlog

- Team select work to do
- Owned / manage by team
- Estimated work remaining updated daily

Blocks List

- List of blocks /unmade decisions
- Owned by Scrum Master
- Updated daily

Burndown Chart

- Effort spend over time
- Stories / features completed

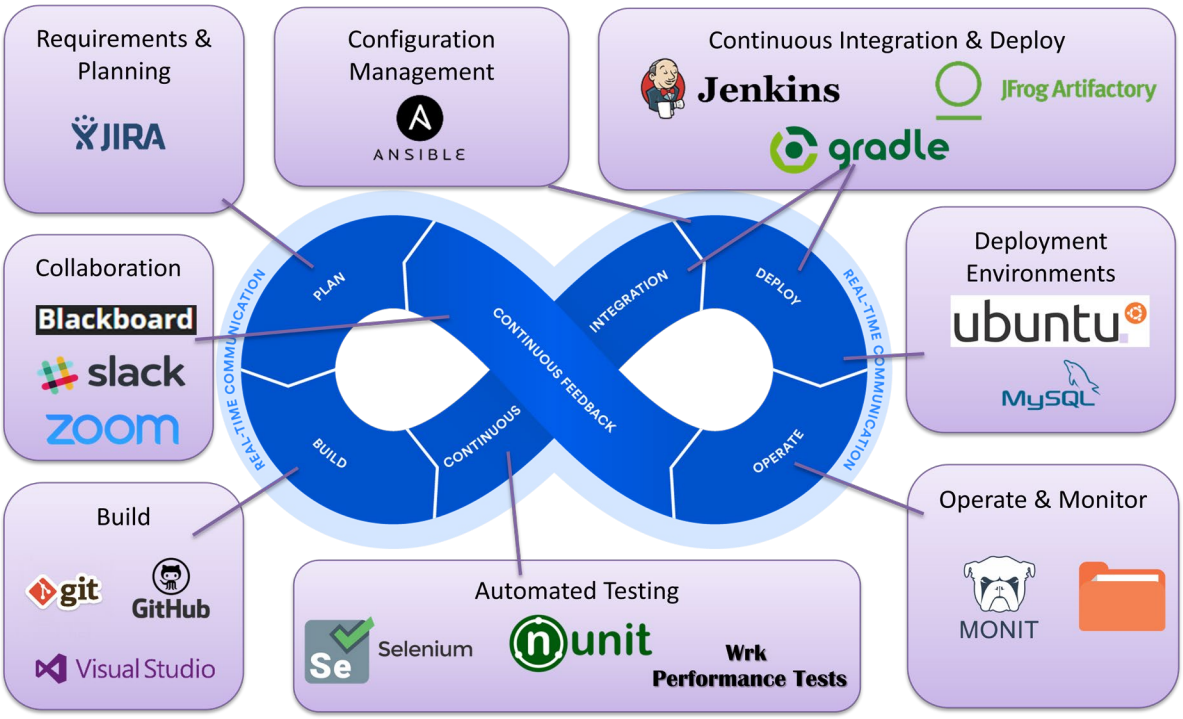
Updated: 02/04/2019
Created by: Charles Aylward

DevOps Process and Environments

DevOps Process

DevOps is a software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops).

Figure 2 - DevOps Process



Source: 1: https://en.wikipedia.org/wiki/DevOps

DevOps Environments

We need to select tools and environments to be configured for the following areas:

- Plan
  - Requirements & Planning (Jira, Slack, Blackboard, Zoom)
  - Architecture & Design (Draw.io)
- Build
  - Language tools (Visual Studio Code or Visual Studio Community 2017)
  - Unit Testing (Nunit)
  - Source Control (Git, GitHub)
  - Automated Testing (Selenium)
- Continuous Integration (Jenkins)
- Deploy
  - Configuration Management (Nuget, Ansible)
  - Artifact Management (jFrog Atrifactory)
  - Deployment Environments (Linux Ubuntu 16.04)
- Operate
  - Operate and Monitor (Monit)

Updated: 02/04/2019
Created by: Charles Aylward

- Continuous Feedback (Slack, Jira, Zoom, Blackboard)

Ansible

Introduction

Ansible is a universal language, unraveling the mystery of how work gets done. Turn tough tasks into repeatable playbooks. Roll out enterprise-wide protocols with the push of a button. Give your team the tools to automate, solve, and share.

WHAT IS ANSIBLE?

Ansible is powerful IT automation that you can learn quickly. It's simple enough for everyone in your IT team to use, yet powerful enough to automate even the most complex deployments. Ansible handles repetitive tasks, giving your team more time to focus on innovation.

Simple. Powerful. Agentless.

With Ansible you can start to do real work in just minutes due to its simple, human-readable language. Altogether its powerful capabilities allow orchestration of your entire application lifecycle regardless of where it's deployed. And Ansible's agentless architecture means it is one less thing to keep secure.

Control. Knowledge. Delegation.

Red Hat® Ansible® Tower expands IT automation to your enterprise adding control, knowledge, and delegation on top of Ansible's automation engine.

Ready to Accelerate?

The answer is clear. We need to stretch beyond the boundaries of manual effort. When you automate, you accelerate. And when you build a culture of automation across your company, anything is possible. Ansible helps smart people do smarter work by having automation do the rest.

Download

The following is a link to an open-source version of Ansible: https://www.ansible.com/

Benefits

Automate: Deploy apps. Manage systems. Crush complexity.

Accelerate: Solve problems once and share the results with everyone.

Collaborate: Break down silos, create a culture of automation.

Integrate: Automate the technologies you already use.

Jenkins

With the Ansible Plugin in Jenkins, this brings the automation integration support into Jenkins. This integration allows the build jobs to deploy to different servers and automate different jobs within the server.

Video: https://www.ansible.com/resources/videos/quick-start-video

@Mary Walsh McGinty

Artifactory

Introduction

Artifactory is a product by JFrog that serves as a binary repository manager. It is a natural extension to the source code repository, as it stores the outcome of the build process called artifacts. This will store individual application components that can later be assembled into a full product, allowing a build to be broken into smaller chunks making more efficient use of resources.

Download

The following is a link to an open-source version of Artifactory from JFrog

https://jfrog.com/open-source/

Benefits

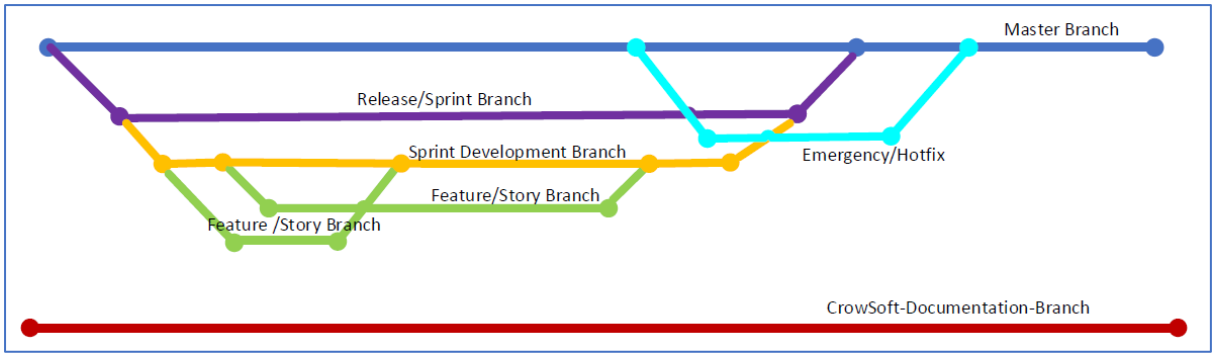
The binary repository allows to host all the artifacts in one central location making the management much simpler for teams. This fits into the DevOps mentality of Build once, Deploy always. After the binary is sent to artifactory it can be deployed into different platforms. Confirming that the code that works in Dev, and then pushed to Prod, will work there also. Artifactory provides an end-to-end automated solution for tracking artifacts from development to production. One of the main benefits of running builds through Artifactory is, fully reproducible builds

Jenkins

With the Artifactory Plugin in Jenkins, this brings the artifactory's build integration support into Jenkins. This integration allows the build jobs to deploy artifacts and resolve dependencies to and from Artifactory and then have them linked to the build job that created them.

@Authors
Colin

CrowSoft Branching Strategy



- Master Branch complete code running on Production (no changes pushed to master)
- Release /Sprint branch will be taken from the Master prefixed by release and sprint identifier
- Each Sprint/Release will have a Development branch branched from the Release.
- Each Feature/Story for a Release will be branched from the Development Branch.
- When a feature is complete, tested and peer reviewed a pull request is made to the Development Branch.
- When all development and testing is complete a pull request will be made to the Release Branch from Development.
- When Release is fully verified as complete a pull request will be made to the Master Branch From the Release Branch.

- Emergency/hotfix branch to be taken from Master with pull request made to Master for hotfix

- **CrowSoft-Documentation-Branch** is for documentation only not to be merged with the Master Branch

@Authors
Mary Walsh McGinty, Liam Whorriskey, Matthew McColgan

Build Server configuration

@ Mary Walsh McGinty

Op System: Ubuntu 16.04

Build_Jenkins_server IP address : (Gold) 172.28.25.136 , (Sandbox) 172.28.25.123

Java

Download Site:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Version to download: jdk-11.02_linux-x64_bin.tar.gz

x64 for 64 bit systems and tar.gz for Ubuntu Op Systems

Steps:

Cd Downloads directory

/Downloads$ sudo tar -xzvf jdk-11.02_linux-x64_bin.tar.gz -C /usr/lib/jvm

X = extract, v= verbose, z= gunzip/unzipping f= extract in filesystem tar= to create and extract tape archives

-C specifies a different directory other than the current working directory

1. Add the file path for java to the /etc/environment file as JAVA_HOME

   cd /etc

   sudo nano environment

   #Edit the file with this line.

   JAVA_HOME=”usr/lib/jvm/java-8-oracle”

2. Next show Ubuntu a link to java

   sudo update-alternatives --install "/usr/lib/jvm" "java" "/usr/lib/jvm/java-8-oracle/bin/java" 1

   Set this version of java as the default

   sudo update-alternatives --set java /usr/lib/jvm/java-8-oracle/bin/java

3. To check that java is install

   java -version This should return:

   java version "1.8.0_201"

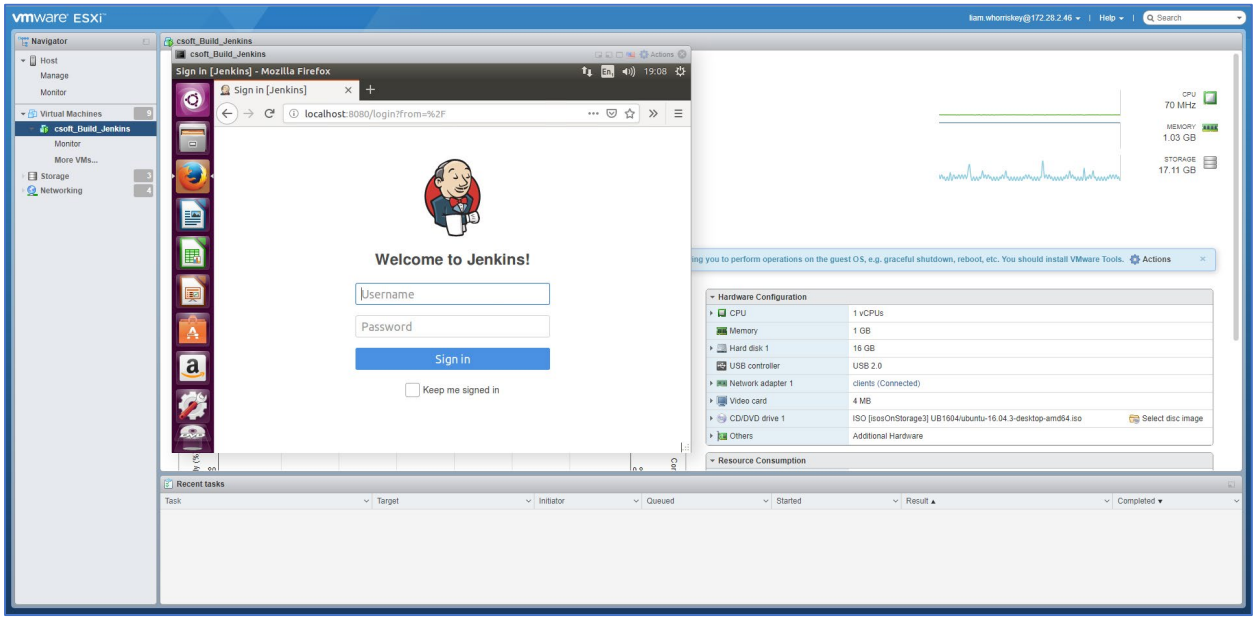
   Java(TM) SE Runtime Environment (build 1.8.0_201-b09)

   Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)

@Authors
Mary Walsh McGinty, Liam Whorriskey, Matthew McColgan

@Liam Whorriskey

Jenkins



Jenkins Setup Complete with suggested plugins.

- Name: Dev Ops
- User: admin
- Pass: ********
- email: l00113360@student.lyit.ie
- Jenkins Url: http://localhost:8080/

Plugins Installed

- Ant Plugin
- Build Timeout
- Email Extension Plugin
- GitHub Branch Source Plugin
- Gradle Plugin
- LDAP Plugin
- Matrix Authorisation Strategy Plugin
- SWASP Markup Formatter Plugin
- PAM Authentication Plugin
- Pipeline
- Pipeline: GitHub Groovy Libraries
- SSH Slaves Plugin
- Subversion plugin
- Timestamper
- Workspace Cleanup Plugin

@Mary Walsh McGinty

- Artifactory plugin
- Ansible

@Authors
Mary Walsh McGinty, Liam Whorriskey, Matthew McColgan

- Audit Trail
- Backup plugin

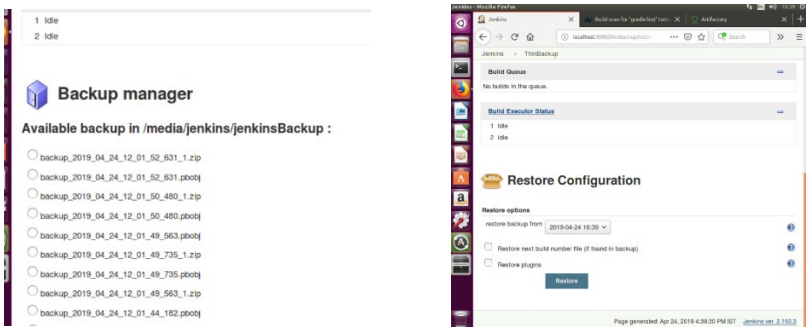


Figure 1 Thin Backup

- CodeSonar
- OWASP Dependency-Check
- Release
- Selenium Builder
- Selenium
- SonarQube Scanner for Jenkins
- Visual Studio Code Metrics

@Mary Walsh McGinty

Jenkins must be running before it can be accessed through LOCALHOST

To start Jenkins: sudo systemctl start jenkins

To check if Jenkins is running: sudo systemctl status jenkins

```
Reading package lists... Done
jenkins@jenkins-virtual-machine:~$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Thu 2019-04-11 21:30:13 IST; 17h ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1810 ExecStart=/etc/init.d/jenkins start (code=exited, status=0/SUCCE

Apr 11 21:30:11 jenkins-virtual-machine systemd[1]: Starting LSB: Start Jenkins
Apr 11 21:30:12 jenkins-virtual-machine jenkins[1810]: Correct java version foun
Apr 11 21:30:12 jenkins-virtual-machine jenkins[1810]:  * Starting Jenkins Autom
Apr 11 21:30:12 jenkins-virtual-machine su[1935]: Successful su for jenkins by r
Apr 11 21:30:12 jenkins-virtual-machine su[1935]: + ??? root:jenkins
Apr 11 21:30:12 jenkins-virtual-machine su[1935]: pam_unix(su:session): session
Apr 11 21:30:13 jenkins-virtual-machine jenkins[1810]:    ...done.
Apr 11 21:30:13 jenkins-virtual-machine systemd[1]: Started LSB: Start Jenkins a
lines 1-14/14 (END)
```

To stop Jenkins: sudo systemctl stop jenkins

Jenkins Security

- Added Users with permissions
- Disabled “Remember Me” option
- Set up Matrix-based security

Artifactory

Prerequisites : Java JDK

Install: JFrog Artifactory-OSS Version 5.8.3. An open source version of JFrog Artifactory

@Authors
Mary Walsh McGinty, Liam Whorriskey, Matthew McColgan

Version to download : jfrog-artifactory-oss-5.8.3.deb

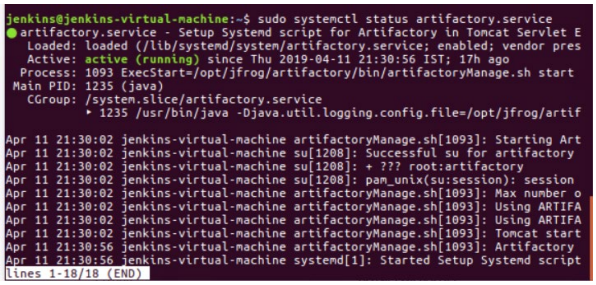
Steps

1. You must be running as root to download
   wget https://bintray.com/artifact/download/jfrog/artifactory-debs/pool/main/j/jfrog-artifactory-oss-deb/jfrog-artifactory-oss-5.8.3.deb
2. Install artifactory
   $ gpg -keyserver pgpkeys.mit.edu -recv-key6B219DCCD7639232
   $ gpg -a -export 6B219DCCD7639232 | sudo apt-key add –
   $ apt-get update
   $ dpkg -i jfrog-artifactory-oss-5.8.3.deb
3. Start the service
   sudo systemctl start artifactory.service

Artifactory must be running before it can be accessed through LOCALHOST

To start Artifactory: sudo systemctl start artifactory.service

To check if Artifactory is running: sudo systemctl status artifactory.service



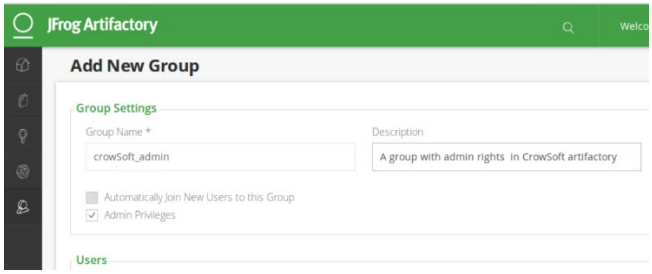
To stop Artifactory: sudo systemctl stop artifactory.service

Configure Artifactory

Artifactory can be accessed through the default Artifactory URL: localhost:8081/artifactory

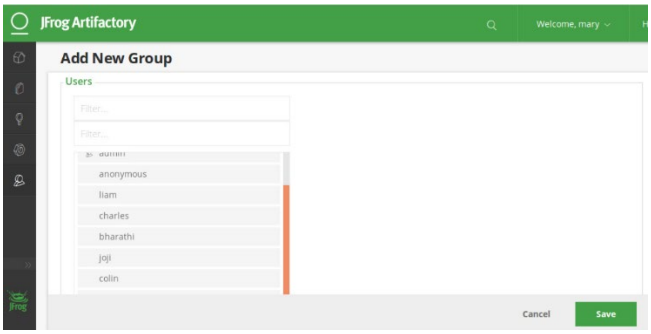
Steps

1. Initial setup activates a setup wizard where basic configuration is completed.
2. Setup administration username and password. Artifactory comes with a default set.
3. Create repositories. Created: Maven, Gradle and Generic Repositories.
4. Create a group; Admin section -> Security -> Groups



Click Save to save the group.

@Authors
Mary Walsh McGinty, Liam Whorriskey, Matthew McColgan



Double click on

5. Set up: Users with username, password.
6. Set permissions for each User.
7. Configured general security

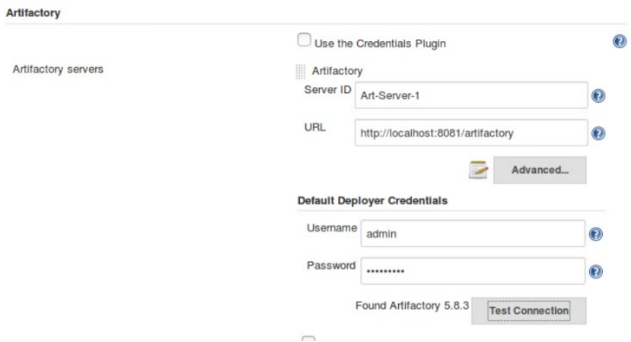
Configuring Artifactory on the Jenkins Server

Steps

1. Log into Jenkins and click on Manage Jenkins
2. Click on Configure System
3. Scroll down the page to the Artifactoy section
4. Add the server details, Url: localhost:8081/artifactory,

@ Matthew McColgan

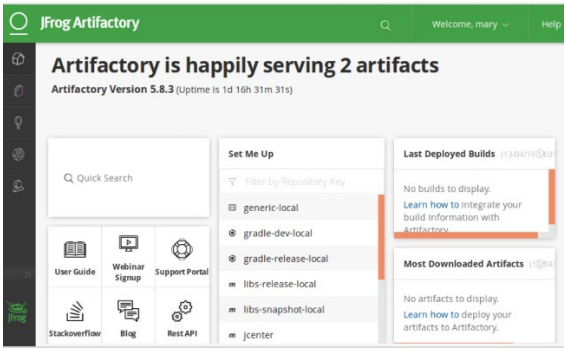
5. Add the Artifactory log in details, username and password, then test the connection



@Mary Walsh McGinty

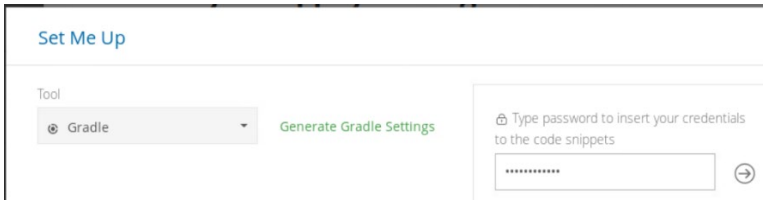
Configuring Jenkins Build

1. Log into Artifactory

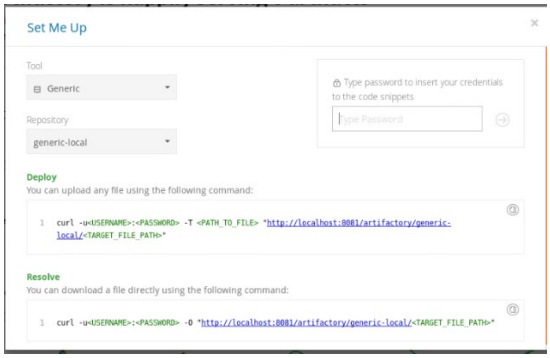


2. Click on repository in Set me up and add your password

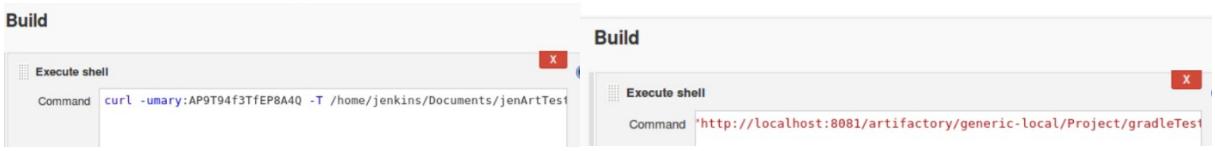
@Authors
Mary Walsh McGinty, Liam Whorriskey, Matthew McColgan



3. A curl command will be returned with a path to the repository you choose



4. Log into Jenkins and in the configuration file for the build go to the build section



Copy the curl command into the Build section as shown, inserting the details relevant to you.

5. Run the Build Now in Jenkins

Gradle

Prerequisites : Java JDK

After Java JDK has been installed install Gradle.

Steps

1 Download the zip Gradle file

cd /tmp

wget https://services.gradle.org/distributions/gradle-4.10.2-bin.zip

This will download the file into the tmp directory
sudo unzip -d /opt/gradle /tmp/gradle-*.zip
This extracts the file into the /opt/gradle/gradle-4.10.2 directory

2. Configure Ubuntu Environment Variables

sudo nano /etc/profile.d/gradle.sh

This creates a new file called gradle.sh in the /etc/profile.d directory. Inside this file the following is added.

export GRADLE_HOME=/opt/gradle/gradle-4.10.2

export PATH=${GRADLE_HOME}/bin:${PATH}

Exit and save the file 'ctrl + x' then y.

When done run the following commands to make the file executable.

sudo chmod +x /etc/profile.d/gradle.sh

source /etc/profile.d/gradle.sh

3. To check if Gradle installed correctly

   gradle -v

   It should return this →

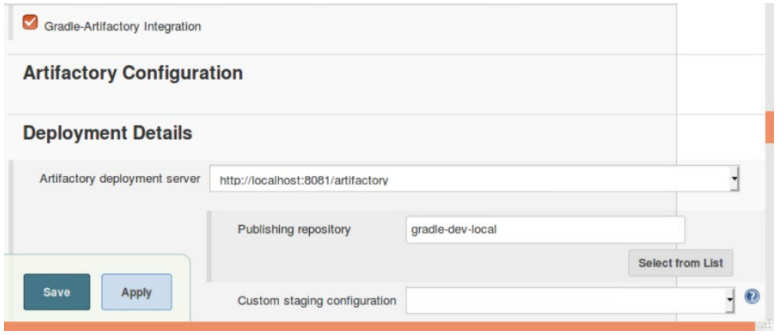
```
jenkins@jenkins-virtual-machine:/tmp$ gradle -v

Welcome to Gradle 4.10.2!

Here are the highlights of this release:
 - Incremental Java compilation by default
 - Periodic Gradle caches cleanup
 - Gradle Kotlin DSL 1.0-RC6
 - Nested included builds
 - SNAPSHOT plugin versions in the `plugins {}` block

For more details see https://docs.gradle.org/4.10.2/release-notes.htm
l

------------------------------------------------------------
Gradle 4.10.2
------------------------------------------------------------

Build time:   2018-09-19 18:10:15 UTC
Revision:     b4d8d5d170bb4ba516e88d7fe5647e2323d791dd

Kotlin DSL:   1.0-rc-6
Kotlin:       1.2.61
Groovy:       2.4.15
Ant:          Apache Ant(TM) version 1.9.11 compiled on March 23 2018
JVM:          1.8.0_201 (Oracle Corporation 25.201-b09)
OS:           Linux 4.10.0-28-generic amd64
```

Gradle Freestyle Example

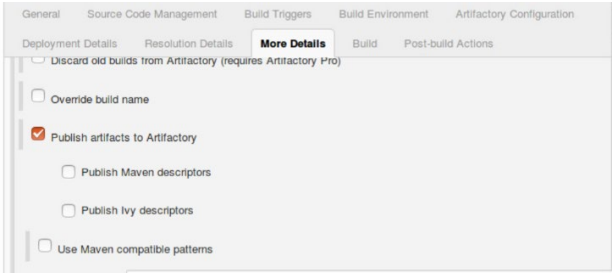
This pulls from Github and builds using Gradle then publishes to Artifactory

Artifactory Configuration

In the artifactory configuration of the Freestyle project. These details were used.



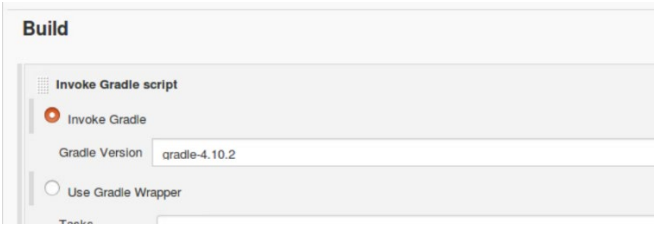
In the More Details check the box to publish the artifacts.



Build the project

In the Build section add the **Invoke Gradle Script**

@Authors
Mary Walsh McGinty, Liam Whorriskey, Matthew McColgan



Here the name of the gradle to use is added.

Example of Gradle script

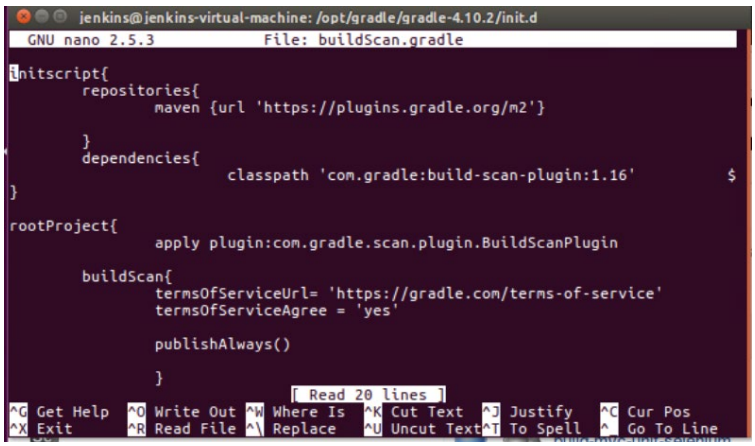
```
//FilePath:  /var/lib/jenkins/workspace/projectName

buildscript{
        repositories{
                jcenter()
                mavenCentral()
                        maven{
                           +url "http://localhost:8081/artifactory/gradle-dev-local"
                          username="admin"
                          password = "Crs0ftArt"
                         }
                }
        dependencies{
                classpath 'com.ullink.gradle:gradle-msbuild-plugin:3.1'
                }
}
apply plugin:'com.ullink.msbuild'

task gradle_msbuild{
        solutionFile = 'crowsoftmvc.csproj'
        projectFile = file('Projects/crowsoftmvc.csproj')
        projectName = 'gradleTest'
        verbosity = 'detailed'
        targets= ['Clean','Rebuld']
}

task artifactoryPublish{
                        skip= false
                        contextUrl = 'http://localhost:8081/artifactory'
                        clientConfig.publisher.repokey="gradle-dev-local"
                        clientConfig.publisher.username="admin"
                        clienConfig.publisher.password="Crs0ftArt"
                }
```

NOTE: The plugin to msBuild through Gradle

This allows for .NET projects to be built in Gradle

Build Scan

This tool allows for a graph representation of the build output. It is a shareable and centralised record of a build. It provides a clear insight into the steps and outcome. This can be published to scans.gradle.com. This is created through a buildScan.gradle file which resides in the init.d directory of gradle.



@Authors
Mary Walsh McGinty, Liam Whorriskey, Matthew McColgan

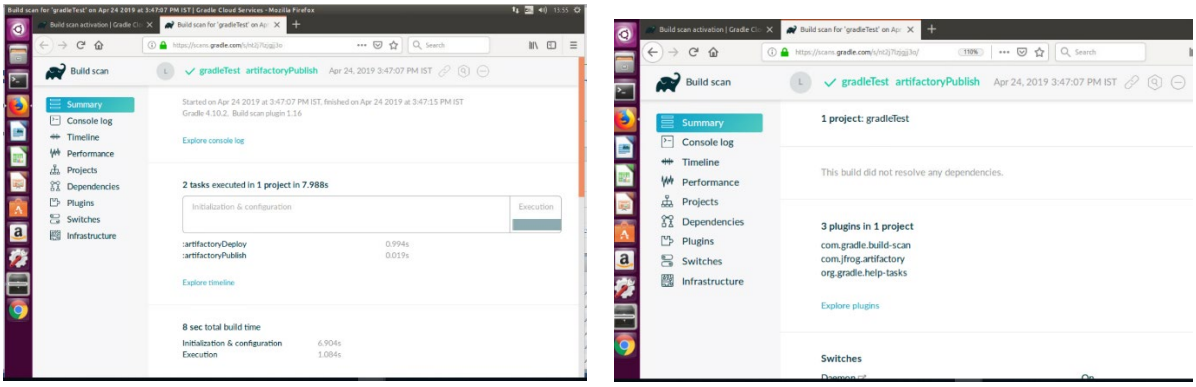


Figure 2 Outcome of the BuildScan

Date Created: 14/04/2019
Created By: Charles Aylward

CrowSoft C# Coding Standards

Contents

Date Created: 14/04/2019
Created By: Charles Aylward

Why Coding Standards?

- To create consistency how the code is developed
- Easy for developers to read and understand the code for maintenance purposes
- Establish a best practise for C# coding and MVC

Naming Conventions

Please note, these naming conventions are based on Microsoft.Net internal standards throughout its development IDE's (Visual Studio IDE), by its intellisense formatter and tooltips.

Classes and Methods

Use PascalCasing for both Classes and Methods. Pascal casing is when the word starts with a Upper case and if more than one word is combined, each word will start with an upper case letter. E.g. **C**ustomer**A**ccount. Make sure the method name is clear and a proper word that can be easily understood.

```
public class Customer
{
    public Customer()
    {
    }

    public void CreateCustomer()
    {
    }
}
```

Local Variables

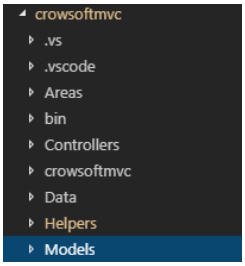
Use camel casing (E.g. camelCasing). Variables passed as parameters will also use camel casing.

```
public void CreateCustomer(int customerId)
{
    var newCustomerId = customerId;
}
```

Namespaces

The application name is in lower case, and any folders use Pascal Casing

Folder example below



Namespace example below.

```
namespace crowsoftmvc.Models
```

Date Created: 14/04/2019
Created By: Charles Aylward

Comments

Commenting your class

Use 3 forward slashes into describe your classes and methods /// (**Note**, Using this method is a standard way to document when using a more advance IDE, such as Visual Studio Community 2017 edition. This will give the developer the ability run automated documentation against its code.)

```
    /// <summary>
    /// This class represent a Customer object
    /// </summary>
    public class Customer
    {
```

Commenting complex code

Use 2 forward slashes, comment delimiter (//) to explain complex code in a short statement.

```
// Get the User Object from the database where email address is found
var user = await mockupDbContext.UserAccount.SingleOrDefaultAsync
      (p => p.EmailAddress == userAccount.EmailAddress);
```

Language Guidelines

Please refer to Microsoft Documentation for language guidelines. C# Coding Conventions (C# Programming Guide) (Microsoft, 2015) https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions

Date Created: 14/04/2019
Created By: Charles Aylward

Best Practice MVC .Net Core Coding

Models

Your Model always represents your database table. In our example, we will use a table called Dummy. It is extremely important to add data annotations to your Model class. This will be used to auto-generate your Controllers and UI validation rules to the Views.

This is the table on MySQL

Table: Dummy

Columns:

PersonID	tinyint(3) UN AI PK
LastName	varchar(255)
FirstName	varchar(255)
Address	varchar(255)
City	varchar(255)

Below find the Model class in our code:

```
public partial class Dummy
{
    [Key]
    5 references
    public byte PersonId { get; set; }

    [Display(Name = "Last Name")]
    [StringLength(60)]
    [Required(ErrorMessage = "{0} is required.")]
    0 references
    public string LastName { get; set; }

    [Display(Name = "First Name")]
    [StringLength(60)]
    [Required(ErrorMessage = "{0} is required.")]
    0 references
    public string FirstName { get; set; }

    [Display(Name = "Address")]
    [StringLength(60)]
    [Required(ErrorMessage = "{0} is required.")]
    0 references
    public string Address { get; set; }

    [Display(Name = "City")]
    [StringLength(60)]
    [Required(ErrorMessage = "{0} is required.")]
    0 references
    public string City { get; set; }
}
```

Note the following key aspects:

- PersonId is the primary key on the database table. This is auto-generated on the database. You need to specify the [Key] data annotation above the column declaration.
- Always use get; set; properties to declare a field in your Model.

```
public byte PersonId { get; set; }
```

- Add the DataAnnotations library to the using statements above the namespace of the model

```
using System.ComponentModel.DataAnnotations;
```

Date Created: 14/04/2019
Created By: Charles Aylward

Note the data annotations below.

- [Display(Name = “”)] is how the name will be displayed on a list or screen on the Views
- StringLength is to limit the input characters on the UI.
- Required will validate that a user entered a value in this field before submitting a form.

```
[Display(Name = "Last Name")]
[StringLength(60)]
[Required(ErrorMessage = "{0} is required.")]
```

For more info on data annotation, go to this:
https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_data_annotations.htm (ASP.NET MVC - Data Annotations)

Controllers

The Controller executes specific actions relating top the Model and called by the View. The controllers will action the CRUD; Create, Read, Update and Delete. The controller normally handles all the incoming requests from a View, using the Model class to return data to the View.

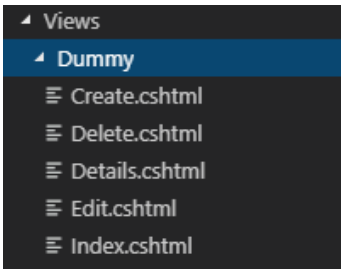
The ApplicationDbContext object needs to be passed as a parameter to the controller to establish a database connection.

```
public class UserAccountsController : Controller
{
    private readonly ApplicationDbContext _context;
    private Counties myCounties = new Counties();

    public UserAccountsController(ApplicationDbContext context)
    {
        _context = context;
    }
```

Views

Views are HTML files generating responses from the user interface. When you use auto generate

We use cshtml files in the new .Net Core MVC web application. It is called Razor Pages which are page focused. Find examples of the CRUD pages created by using the command line, refer to Heading: *Auto Generate Controllers and Views*



Date Created: 14/04/2019
Created By: Charles Aylward

Auto Generate Controllers and Views

You can use the scaffolding tool to auto generate Controllers and razor pages (Views) from your Model.

First, Install the codegenerator using the following command

```
dotnet tool install --global dotnet-aspnet-codegenerator
```

Use the following command line to generate the controller and views automatically.

```
dotnet aspnet-codegenerator --project . controller -name DummyController -m Dummy -dc
ApplicationDbContext
```

ASP.NET Core code generator parameters:

Parameter	Description
-m	The name of the model
-dc	The DbContext class to use.
-udl	Use the default layout
-outDir	The relative output folder path to create the views
--referenceScriptLibraries	Adds _ValidationScriptsPartial to Edit and Create pages

*Reference URL: https://docs.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/model?view=aspnetcore-2.2&tabs=visual-studio-code

Date Created: 14/04/2019
Created By: Charles Aylward

Bibliography

Microsoft. (2015, 07 20). *C# Coding Conventions (C# Programming Guide)*. Retrieved 04 14, 2019, from Microsoft.com: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions

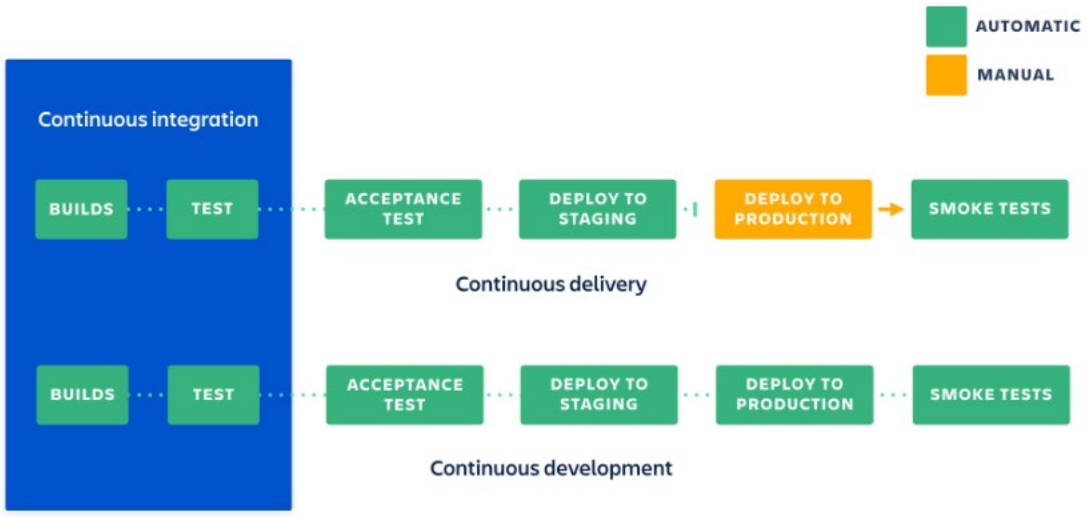
CI/CD

CI and CD are acronyms to represent modern development practices.

CI stands for continuous integration, which is a practice that focuses on developers integrating code regularly into a shared repository. Check-ins are verified by automated builds and testing to ensure bugs are caught often and early.

CD, on the other hand, can either mean continuous delivery or continuous deployment. Both are very similar but have a distinct difference. Continuous delivery is automating the release process by deploying changes on a schedule or at the click of a button. Continuous deployment on the other hand, deploys changes automatically after passing all stages of a production pipeline.

Continuous deployment and continuous delivery are very similar in practice and both are an extension of continuous integration in an automated DevOps environment.



Some tools for CD:

Many CI tools can be doubled up as deployment tools:

Jenkins:
Although Jenkins is primarily a CI server, it can be configured to be used as a deployment tool with the many plugins available to it.

Tools that are designed specifically for automated deployment:

Octopus:
Octopus is an automated deployment and release management server. It is designed to simplify deployment of ASP.NET applications, windows services and databases.

GoCD:
Go is an open-source tool to help in the continuous deployment of software. It supports several version control tools, including Git. Similar to Jenkins, plugins can be installed to extend its features.

These plugins support authentication and authorization software, version control, build tools, notification and chat tools and cloud computing providers.

Recommendation:

Although Octopus and Go are viable options: Octopus as it is the de facto for .NET applications and it integrates with Jenkins and GoCD as it is an open-source tool which can be extended with its plugins. Due to the familiarity with Jenkins, it is already being used as our CI server and with plugins we can extend its features for deployments I recommend we use Jenkins as our CD server.

CI/CD

CI and CD are acronyms to represent modern development practices.

CI stands for continuous integration, which is a practice that focuses on developers integrating code regularly into a shared repository. Check-ins are verified by automated builds and testing to ensure bugs are caught often and early.

CD, on the other hand, can either mean continuous delivery or continuous deployment. Both are very similar but have a distinct difference. Continuous delivery is automating the release process by deploying changes on a schedule or at the click of a button. Continuous deployment on the other hand, deploys changes automatically after passing all stages of a production pipeline.

Continuous deployment and continuous delivery are very similar in practice and both are an extension of continuous integration in an automated DevOps environment.

Some tools for CI:

Jenkins:
Jenkins is an open-source CI tool written in Java. Some of its main features are:

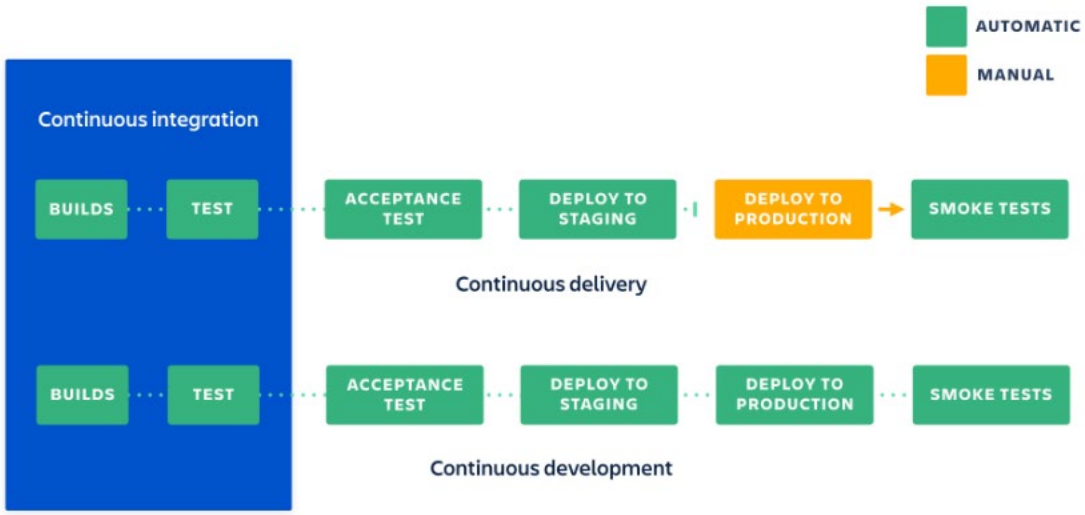
- Compatible with most OS and versions of Linux, Mac OS and Windows.
- Easy installation
- Easily set up and configured with its web interface
- Plugins – there are over 1500 plugins to

TeamCity:
TeamCity, also written in Java, main features include:

- Configure builds in DSL
- Project level cloud profiles
- Remote run and pre-tested commit

Travis CI:
TravisCI is free for open source projects. Its main features include:

- Hosted – not dependant on any platform
- Parallel testing
- Easy setup, no installation



Recommendation:

Due to its flexibility, compatibility, ease of use and as an open-source tool, I recommend we use Jenkins as our continuous integration server.

CrowSoft Technologies

Disaster Recovery Document

Michael Mcfadden

Last reviewed & tested - 26/04/2019

Introduction

This document covers the process and disaster recovery procedures in place at CrowSoft Technologies in case of a disaster. The disaster can be a geographical disaster or any other failure that leads to the Production Environment's downtime. The purpose of this document is to ensure minimal downtime, data integrity and availability, in case of a disaster. This document will try to cover all the aspects that should be taken care in case of a disaster, as well as the safety of people. This document outlines the process and procedures that will help us overcome the disaster with minimal effect on the working of our organization

Personnel

Stake Holders & Responsibilities during a Disaster Recovery

Name	**Job role**	**Contact details**	**DR process owned**
Ruth Lennon	*Product Owner*	*(Phone numbers, email address, normal workplace)*	*Oversees all decisions made as part of DR*
Colin Kenny	*IT sys admin*	*(Phone numbers, email address, normal workplace)*	*Backup and restore Jenkins pipeline and Git Repositories*
Charles Aylward	*IT sys admin*	*Phone numbers, email address, normal workplace)*	*Facilitate in coordination of recovery plan*
Matthew McColgan	*IT sys admin*	*Phone numbers, email address, normal workplace)*	*Backup and restore application code and database*
Mary Walsh McGinty	*IT sys admin*	*Phone numbers, email address, normal workplace)*	*Backup and restore server snapshots from VM*
Liam Whoriskey	*IT sys admin*	*Phone numbers, email address, normal workplace)*	*Backup and restore Documentation*
Bharathi	*IT sys admin*	*Phone numbers, email address, normal workplace)*	*Backup and restore Jenkins pipeline and Git Repositories*
Joji	*IT sys admin*	*Phone numbers, email address, normal workplace)*	*Backup and restore application code and database*

Michael McFadden	*IT sys admin*	*Phone numbers, email address, normal workplace)*	*Facilitate in coordination of recovery plan*

This DR Policy

The major goals of this policy are as follows:

1. Ensure Personnel safety is number 1 priority
2. To minimize interruptions to the normal operations.
3. To limit the extent of disruption and damage.
4. To minimize the economic impact of the interruption.
5. To establish alternative means of operation in advance.
6. To train personnel with emergency procedures.
7. To provide for smooth and rapid restoration of service.

Discovery - Policy Initiation

1. Notify Stakeholders
2. Contact and set up disaster recovery team
3. Set a disaster recovery team primary point of contact and incident center of command
4. Determine degree of disaster
5. Determine response first steps
6. Determine estimated time to recovery complete
7. Update senior management on initial findings
8. Select appropriate disaster recovery procedure based on degree of disaster

Response - Policy Response

1. Notify users of the disruption of service
2. Implement proper application recovery procedure dependent on extent of disaster
3. Monitor progress on recovery policy implementation through DR recovery team primary point of contact
4. Contact backup site and establish a point of contact at that location

5. Contact all other necessary personnel–both user and data processing–required for success of the recovery policy
6. Contact vendors–both hardware and software–related to the disasters affected equipment

Recovery - Recover from disaster

1. Determine applications to be run and in what sequence
2. Ensure that all personnel involved know their tasks
3. Make sure that the DR team at the disaster site has the necessary information to begin restoring
4. Recover affected equipment from disk or tape media
5. Verify recovered equipment is functioning properly
6. Begin normal operations and notify staff of completion of recovery
7. Review disaster process and document good/bad sections of the procedure

Backups

- Server Snapshots – can be restored via recent snapshots
- Code Backups – Back up on Git repository
- Jenkins Pipelines Backups – Scheduled backups stored on server
- Database backups – Adhoc backups saved locally
- Documentation Backups – Adhoc backup saved to shared one drive

Backups should be accessible in a timely manner in order to restore them to your recovery servers. The best solution is to store your data backups in multiple offsite locations. Another solution is to store backups or data on a cloud platform that guarantees the five 9's, 99.999% availability. Due to the restriction of the college project, the above backup policies would be carried out as best practice.

Restore Procedures

- Manual Restore from backups if we lose a server or source code
- Rebuild from scratch - Refer to Flipping book Binder link below for all set up procedures

Supporting Documemtation

- Flipping book Binder: https://online.flippingbook.com/view/22284/14/
- OneDrive videos: https://studentlyit-my.sharepoint.com/personal/l00113360_student_lyit_ie/_layouts/15/onedrive.aspx
- Videos: https://www.youtube.com/playlist?list=PLQYTpRHapl6FHiupp7bP0LyOQzR1bf8L2

Risk Management

Risk management involves the identification of risks and their causes;

Risk factors can be categorized as (Gupta 2008):
- Technical
- Environmental
- Managerial
- Organisational

Balancing disaster recovery planning with risk management will save money in the long run while offering adequate protection from the most likely disasters. Ideally, a disaster recovery plan will protect your company from every foreseeable disaster and return your company to full operations in the shortest possible amount of time. Unfortunately, this is cost prohibitive. You simply cannot afford to protect your organization against every possible disaster. Selecting which scenarios and how to protect your company against them is called risk management.

Disaster risk management approach

Risk prevention

Risk analyses to identify the hazards and how exposed the business is and can reveal what can be done to prevent or minimize damage, steps can then be taken to strengthen the defences

Addressing residual risk

Even when prevention measures are put in place, every scenario can't be accounted for due to often the cost and effort to cover every scenario. .Ex-ante financing such as specific insurance policies to certain disasters, which may be too costly to attempt to prevent ,can be obtained to help finance such a recovery plan .Ex-post financing will severely affect the business as all costs will have to be met by the company

Preparing

Prepare for a disaster by ensuring a disaster recovery plan is put in place, this plan if well thought out and tested will ensure a speedy and cost-efficient recovery.

Responding

Following a disaster, a quick response will ensure further damage to the business and lower the cost of recovery. The DRD will come into play here again with everyone knowing their roles and the Ex-ante financing that was put in place will aid and make the recovery more successful.

Recovery

The recovery process can now begin with the help of the ex-ante financing to aid the recovery and lessons learnt from the disaster can be used as opportunities to improve the rebuilding of the business. These will help in preventing the same disaster from occurring again.

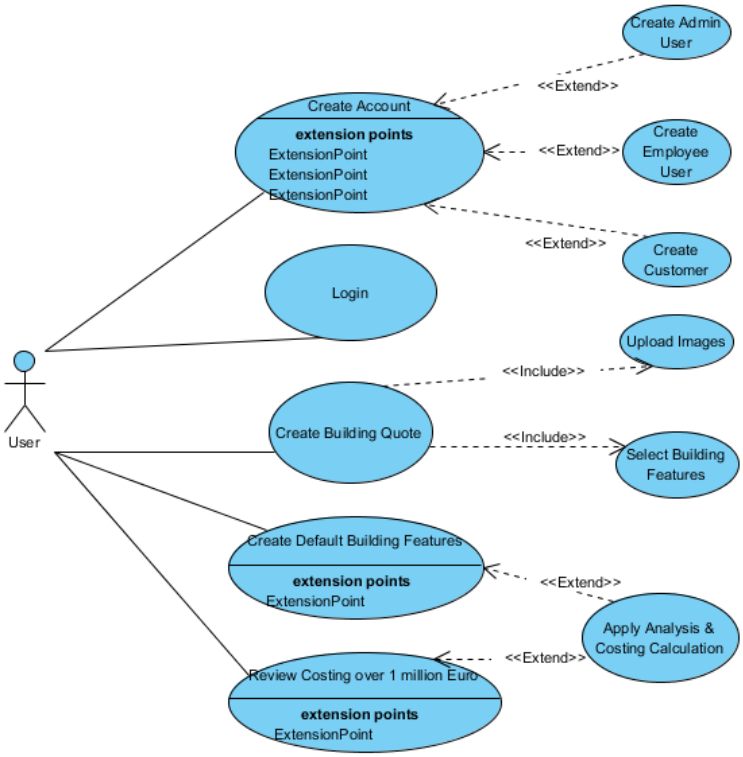
Date Created: 20/03/2019 (Charles Aylward)
Date Updated: 27/04/2019

CrowSoft Database Design

CrowSoft Use Case

The use case below is base on CrowSoft functional requirements.

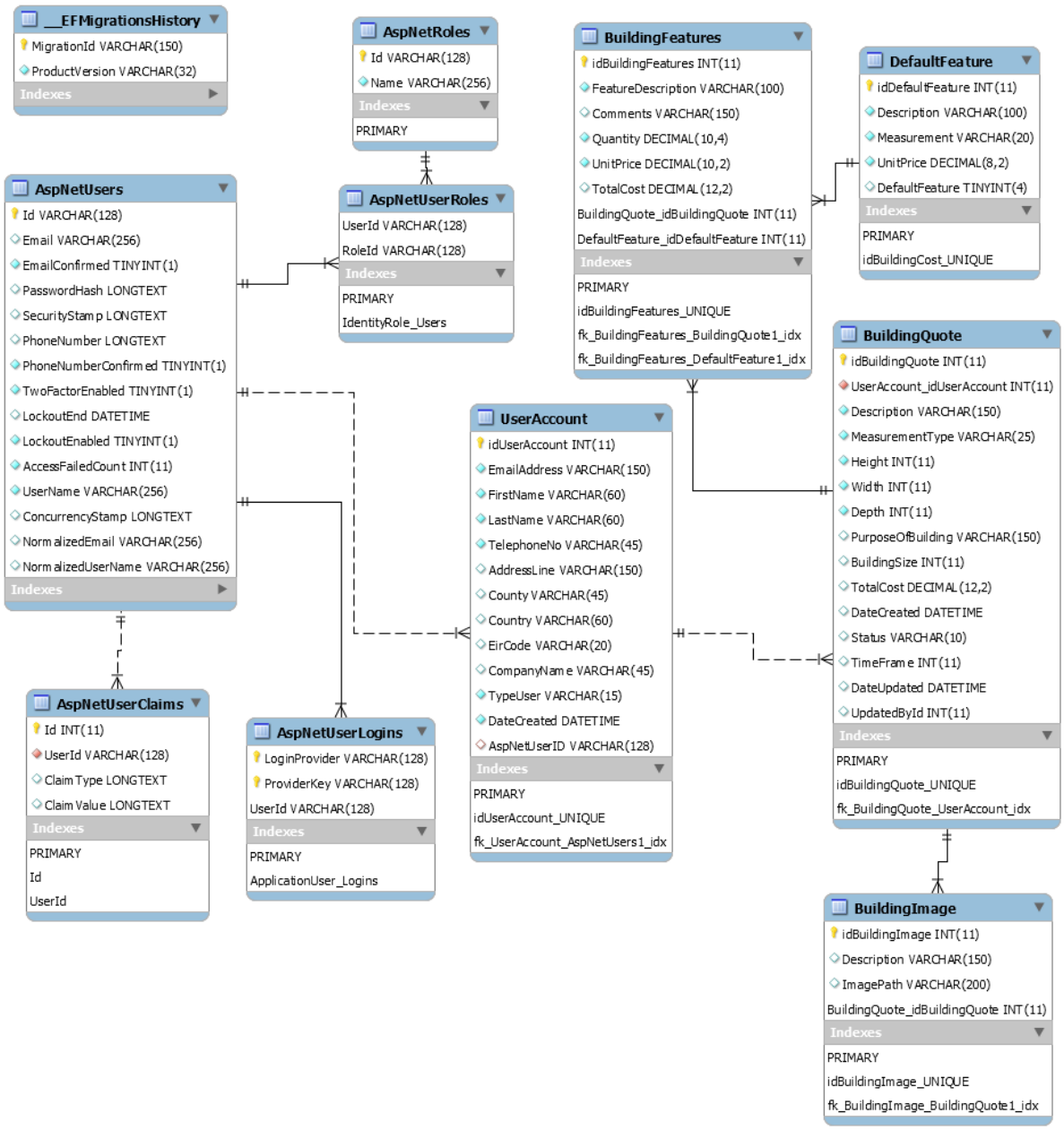
- Users require creating an account, and can either be an Admin, Employee or Customer user.
- User requires to login into the system and the system needs to validate what role the user belongs to
- A customer/client user can create a building quote for analysis and costing. The customer can upload multiple images and also select custom features to be added to the costs.
- The Admin user can add default building features for analysis and costing
- An admin user can review the building analysis and costing, update changes to the costs/features and apply the costing again, or approve the costs.



Date Created: 20/03/2019 (Charles Aylward)
Date Updated: 27/04/2019

CrowSoft Database Table Structures

CrowSoft ERD



Date Created: 20/03/2019 (Charles Aylward)
Date Updated: 27/04/2019

Asp.Net Identity Security Tables

Microsoft introduced ASP.Net Identify Security for use by an ASP.Net Core MVC application. This security identity enables a role-based security, hidden/hash passwords and also prevents cross site scripting, SQL injections, and much more. The reason for choosing this method is ensure users are allocated to a specific role.

The following table are required for ASP.Net Core Identity Manager:

- AspNetUsers
  - This table stores the main user when registered on the web site
  - It also store passwords using Microsoft's internal hashing algorithm. The following hashing format is used: *PBKDF2 with HMAC-SHA256, 128-bit salt, 256-bit subkey, 10000 iterations*
  - A sign-in cookie is used when a user login
- AspNetRoles
  - To store roles in the database assigned to users
  - Roles: Client, Admin and User
- AspNetUserRoles
  - This table is the reference table of what users belong to which roles.
  - Many-to-many between AspNetUsers and AspNetRoles
- AspNetUserLogins (Not in use)
  - This table is used when using OAuth for storing other provider information such as Google, Facebook or Microsoft.
- AspNetUserClaim (Not in use)
  - This is normally use when a user have claims in more than one application. We do not need to use this table.

Date Created: 20/03/2019 (Charles Aylward)
Date Updated: 27/04/2019

User Account Table

This table will be used for multipurpose users; for employees, admin users and customers. This will use a role-base security at three levels, admin (all access), employee user (access to view costing and building quotes), customer (allow to create building quote including features). Features on the UI will be visible according to the role of the user.

Column Name	Description	Mandatory	Sample Data	Validation	type
idUserAccount	Primary Key - Auto Increment	Y	1		int
EmailAddress	Email Address and Username	Y	joe@email.com	Valid Email Address	string
Password	Hash Password using Cryptography in ASP.Net Core before saving to DB	Y	1O3JMY.Tw$AdLn	Validate Login	string
FirstName	First Name	Y	Joe	Not null	string
LastName	Last Name	Y	Blogg	Not null	string
TelephoneNo	Telephone No	Y	555-555-5555	Not null	string
AddressLine1	Address Line 1	N	123 Street		string
AddressLine2	Address Line 2	N	My Area		string
County	County	N	Donegal		string
Country	Country	N	Ireland		string
EirCode	Postal Code	N	F90 A1A1		string
CompanyName	CompanyName	Y	CrowSoft	Not null	string
TypeUser	Drop Down Box (Admin, Employee, Customer)	Y	Admin	Not null	string
DateCreated	Auto Generated by MySQL (CURRENT_TIMESTAMP`)	Y	2019-02-31 20:00:00'	Not null	DateTime
AspNetUserID	Foreign Key column for AspNetUsers table	Y	GUID String 142331f4-23d2-4835-9b77-b6a3d38223a2	Not null	string

Date Created: 20/03/2019 (Charles Aylward)
Date Updated: 27/04/2019

Default Feature Table

The default feature table is a configuration table for both default features to be added to each and every building quote, but also where a customer can select multiple features to be added to the quote and to be calculated.

Column Name	Description	Mandatory	Sample Data	Validation	type
idDefaultFeature	Primary Key - Auto Increment	Y	1		int
Description	Description of the Feature	Y	E.g. Concrete Walls, Kitchen, Tiled Roof	Not null	string
Measurement	Type of measurement used to calculate unit price	Y	E.g. Qty, Time, Sqr Meters, Sqr Foot	Not null	string
UnitPrice	Unit Price of feature	Y	€100.00	Not null	int
DefaultFeature	This is a Boolean field. If this field = 1, then this feature will be automatically be added to the total cost, if 0, then it will display as an option for the customer to select.	Y	1 or 0	Not null	tinyint

Date Created: 20/03/2019 (Charles Aylward)
Date Updated: 27/04/2019

Building Quote Table

This is the header table when a customer creates a new quote for costing. A user/customer has to register for an account before they can create a quote for costing. A separate process will automatically run when a customer click confirms. This calculation will add up all default and customer features required and add it to the total cost.

Column Name	Description	Mandatory	Sample Data	Validation	type
idBuildingQuote	Primary Key - Auto Increment	Y	1		int
UserAccount_idUserAccount	Many to One Foreign Key to UserAccess Table	Y	1	User has to be logged in	int
Description	Description of the building the customer requires	Y	Offices for Joe Blogg	Not null	string
MeasurementType	Drop Down for Metrics or Imperial	Y	Metric	Not null	string
Height	Height	Y	50	Not null	int
Width	Width	Y	200	Not null	int
Depth	Depth	Y	200	Not null	int
PurposeOfBuilding	Customer explains purpose of the building	N	Office Block		string
BuildingSize	Calculated from Height, Width and Depth	N		Auto calculation	int
TotalCost	System calculates total cost of all features	N	€130,000.00	If cost is over €1 mill, change status to Withhold	decimal
DateCreated	Auto Generated by MySQL (CURRENT_TIMESTAMP)	Y	2019-02-31 20:00:00'	Not null	datetime
Status	Status of the quote: Confirmed, Approved, Withhold, Cancelled	Y	Approved	Not null	string
TimeFrame	This is the time in months. Calculated from labour cost.	N	6	Auto calculation	decimal
DateUpdated	Updated when record is updated by Admin	N	2019-02-31 20:00:00'		datetime
UpdatedBy	Admin Id who updated the qoute	N	1		int

Date Created: 20/03/2019 (Charles Aylward)
Date Updated: 27/04/2019

Building Features Table

This table is to save the customer selected features to be added to the final costing.

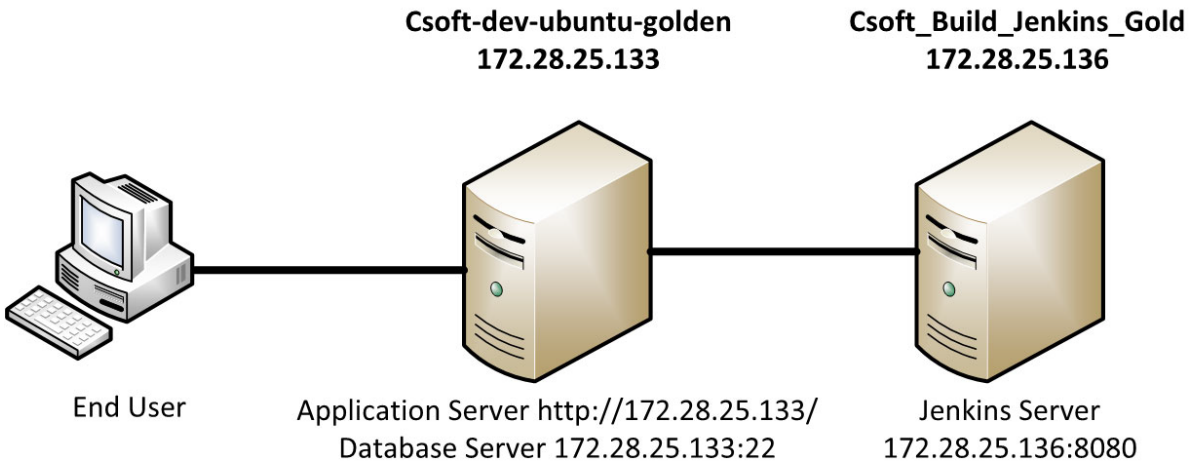
Column Name	Description	Mandatory	Sample Data	Validation	type
idBuildingFeatures	Primary Key - Auto Increment	Y	1		int
Comments	Comments of whats required	N	Colour of the walls white		string
FeatureDescription	Copied from default feature select		Concrete	Auto generate from default features selected	string
Quantity	Qty required	Y	150	Not null	int
UnitPrice		Y	€50	Not null	decimal
TotalCost	Total cost calculated	N	€7,500	Auto calculate	decimal
BuildingQuote_idBuildingQuote	Foreign key for building quote	Y	1	Not null	int
DefaultFeature_idDefaultFeature	Foreign key for Default Feature	Y	1	Not null	int

Building Image Table

The customer will be able to upload multiple images and this is where the image path will be stored. This has a many to one relationship with the Building Quote header table.

Column Name	Description	Mandatory	Sample Data	Validation	type
idBuildingImage	Primary Key - Auto Increment	Y	1		int
Description	Image Description	Y	Building Plans		string
ImagePath	Path of the folder where images is stored	Y	d:/crowsoft/images	Auto generate filename	string
BuildingQuote_idBuildingQuote	Foreign key for building qoute	Y	1	Not null	int

CrowSoft Dev Environment



Software/apps installed
Jenkins
Java
Open ssh client
Open ssh server
Putty

Networking
Ssh
https

Security Tools for the DevOps Pipeline

Contents

Introduction

These are some of the available Open Source security tools for the DevOps pipeline. I have only included the tools that are going to be relevant to the project. I have included both Python and Java for an example. https://mattboegner.com/secure_cicd_pipeline_2/ This is an interesting article on securing the DevOps pipeline if anybody is interested in reading it.

Security Development Tools

Mittn: For Python development, allows the developer to create checks based on use cases. Developers can also code check to catch mistakes that they previously made. A security testing tool for CI. https://github.com/F-Secure/mittn

BDD-Security: For Java development, works the same way as Mittn. A testing framework used for functional security, infrastructure security and application security testing.

Jasmine: JavaScript

QUnit: JavaScript

Static and Dynamic code Analysis

SonarQube

OWASP Zap & OWASP Zapper (Jenkins plugin): Allow automating attack proxy to test for possible attacks

Pre-commit Version control Security

Git-hound: A free security tool to provide automated checks to ensure that no sensitive data is committed into code.

OWASP Threat Dragon: Building security into the design of the application. A threat modeling web application, including system diagramming.

SonarLint: An IDE extension (IntelliJ, Eclipse, Visual Studio). Helps to detect and fix quality issues as the code is written. Bug detection issues are detected and reported, pinpointing the problem, and gives corrective recommendations.

Puma Scan: An IDE extension (Visual Studio) that provides real time continuous source code analysis for C#, .NET. Vulnerabilities are displayed immediately inside the development environment.

Continuous Integration Security

ESLint: A static code analysis tool. Debugging is done by examining the port without executing the port. A linter tool for identifying and reporting on patterns in JavaScript. Helps to maintain code quality with ease.

Mocha: A JavaScript test framework running on NodeJS and in the browser making asynchronous testing simple and easy. Runs serially allowing for flexible and accurate reporting while mapping uncaught exceptions to the correct test cases.

OWASP Dependency Check: A utility that identifies project dependencies and checks if there are any known publicly disclosed vulnerabilities . A security audit tool with plugins for build-tools, Maven, Jenkins, gradle and Ant-task. It automatically updates itself.

Docker Bench: A script that checks for dozens of common best practices around deploying docker containers in production. Currently supports multiple versions of docker and docker-bench will determine the test set to run based on the docker version on the host machine. It scans the docker environments, start the host level and inspect all the aspects of the host, the docker daemon and its configuration.

Continuous Delivery Security. (Before, during and after Deployment)

SSLLabs-scan: A command-line scanning tool that doubles as the reference API client of SSL Labs. Security smoke tests. Scans web PKIs. Designed for automated or bulk testing. SSL Labs API exposes the complete SSL/ TLS server testing functionality in a programmatic fashion allowing for scheduled and bulk assessments.

OSQuery: Uses basic SQL commands to leverage a relational data-model to describe a device. Safety checks. When attackers leave a malicious process running but delete the original binary on the disk. This query returns any process whose original binary has been deleted, which could be and indicator of a suspicious process. Under Apache license.

Ansible vault: A feature of Ansible that allows keeping sensitive data in encrypted files. These can then be placed in source code.

Samhain: A host-based intrusion detection system. It provides file integrity checking and log file monitoring/analysis as well as rootkit detection, port monitoring, detection of rogue SUID executables and hidden process. Designed to monitor multiple hosts with potentially different operating systems. It provides centralized logging and maintenance and can be used as a standalone application on a single host.

Operations Security (Continuous security, monitoring, testing, audit and compliance checks)

Chaos Monkey: It injects faults to randomly terminate virtual machine and container instances that run inside a production environment. Exposing engineers to failures more frequently in order to help them build more resilient services.

Open SCAP: An ecosystem which provides multiple tools to assist administrators and auditors with assessment, measurement and enforcement of security baselines. Provides a wide variety of hardening guides and configuration baseline. Enabling you to choose a security policy that best suits the needs of your organisation.

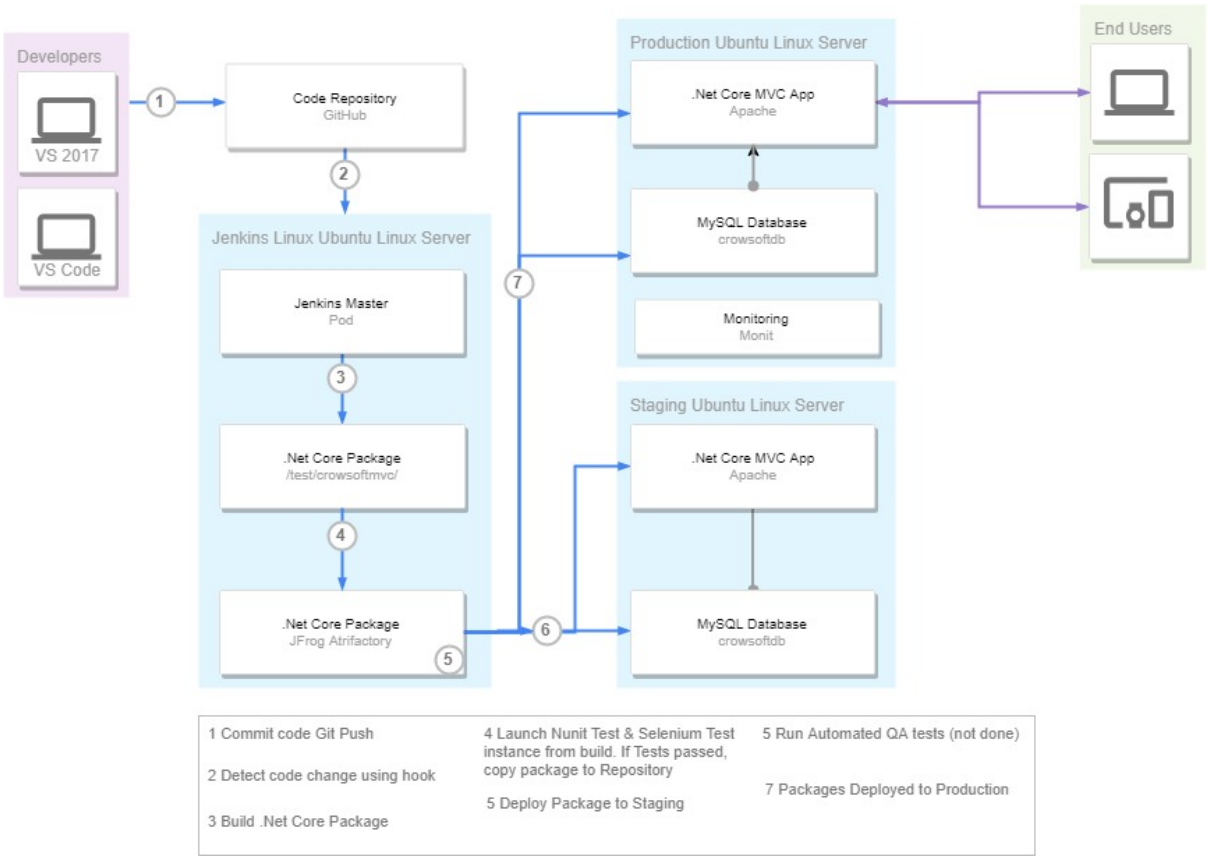
Grafana: A metric analytics and visualization suite for continuous monitoring. Used for time series data for infrastructure and application analytics. It allows to query, visualise, alert and understand the matrices no matter where they are stored. It creates an explore and share dashboard with your schemas.

ElastAlert: A tool for continuous monitoring. A simple framework for alerting on anomalies , inconsistencies, spikes or other patterns of interest from data in Elasticsearch. It works by combining Elasticsearch with two types of components, rule types and alerts. Elasticsearch is periodically queried and the data is passed to the rule type, which determines when a match is found. When a match occurs, it is given to one or more alerts, which take action based on the match. This is configured by a set of rules, each of which defines a query, a rule type and a set of alerts.

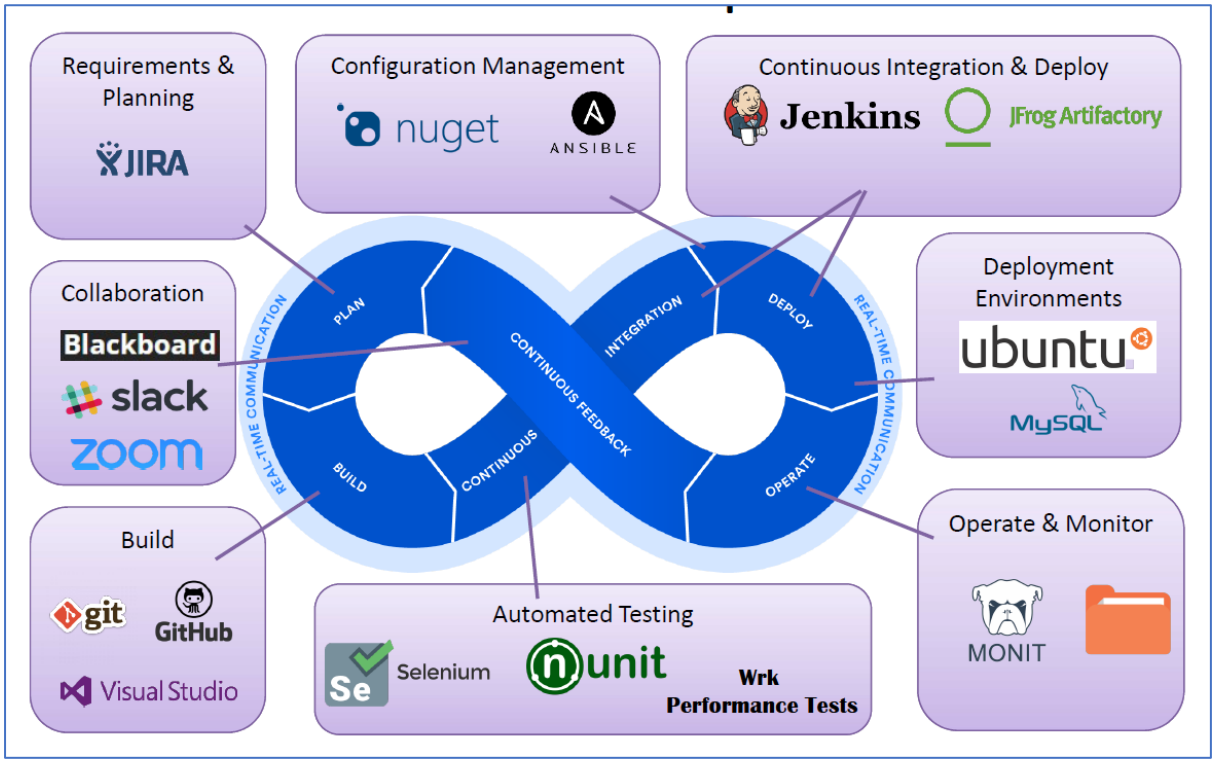
Created by Charles Aylward (Updated: 02/05/2019)

DevOps Pipeline

Diagram updated.



CrowSoft DevOps Tools



CrowSoft

The customer would like an on-line system to take in building details for analysis. CrowSoft is an engineering and tech company. The product that they wish to market is BusIntelligence. The system must take details for buildings and to provide easy to use analysis features. For example, they may wish to have a review all buildings which cost more than €1 million to build and have special features as this may indicate future maintenance projects that may be exploited. Keep the analysis simple. The analysis system should be clean and simple. The system needs to take into account the usual details and present simplified graphics. It must be possible to upload files or images. The administrator should be able to access detailed information and edit as appropriate. Once the client enters details it should not be able to be changed by the client.

1. Functional Requirements

Client/User Requirements

1. Sign-In/Sign-out

1.1 The Client/User shall enter sign-in credentials (username and password).

1.1.1 On successful sign-in the user shall be brought to the signed-in home page.

1.1.2 On unsuccessful sign-in the user shall be displayed an error and be brought back to sign-in page.

1.2 The Client/User shall sign-out and confirmation page/home page is displayed.

2. Create Account (Sign-up)

2.1 Client/User shall create an account via the sign-in page.

2.2 Client/User shall be able to enter in following sign-up credentials:

- First name
- Last name
- Telephone/Mobile
- Address line 1
- Address line 2
- County
- Country
- Eircode
- Email
- Company name

2.2.1 On successful sign-up the Client/User shall be redirected to the sign-in page.

2.2.2 On unsuccessful sign-up the Client/User shall be displayed an appropriate error message.

3. Enter Details of Buildings

3.1 Client/User can create a new entry

3.2 Client/User must be able to upload images from file

3.2.1 Client/User must be able to upload images from mobile device
3.3 Client/User selects measurement types from drop down boxes
3.3.1.1 Select imperial or metric standard
3.3.1.2 Select height, width, depth
3.3.1.3 Select material type of building
3.3.1.4 Select purpose of building
3.4 Client/User will view selected items and confirm/cancel
3.5 Client/User can select various analysis features (cost per sqm, buildings for purpose, age of building etc.)
3.6 Client/User will receive confirmation email when required analysis feature is activated, viewed and confirmed.
3.7 Client/User cannot edit details once confirmed

Admin Requirements

1. Sign-In/Sign-out

1.1 The admin shall enter sign-in credentials (username and password).
1.1.1 On successful sign-in the admin shall be brought to the signed-in home page.
1.1.2 On unsuccessful sign-in the admin shall be displayed an error and be brought back to sign-in page.
1.2 The admin shall sign-out and confirmation page/home page is displayed.

2. Add new client/user

2.1 The admin can create an account for new client/user
2.2 The admin shall be able to enters the following client/user details:

- client/user ID
- First name
- Last name
- Contact number
- Address
- Email

2.2.1 If client/user account is created successfully the admin shall be redirected to the admin homepage.
2.2.2 If employee account is unsuccessful an error message shall appear.

3. Edit client/user

3.1 The admin shall enter unique Employee ID.
3.2 Employee details page will appear, edit details and save any changes.
3.2.1 A successful save will redirect the admin to the admin homepage.
3.2.2 An unsuccessful save will prompt an error.

4. Edit client/user building details

4.1 The admin shall enter unique Client/User ID.
4.2 Client/user details page will appear, edit details and save any changes.
4.2.1 A successful save will redirect the admin to the admin homepage.
4.2.2 An unsuccessful save will prompt an error.

5. Remove client/user

5.1 The admin shall enter a client/user ID which they wish to remove.
5.2 The admin shall select a Remove client/user button.
5.3 A message prompt will ask to confirm, select yes.

5.3.1 A successful remove will redirect the admin to the admin homepage.
5.3.2 An unsuccessful remove will prompt an error.

2. Non-Functional Requirements

Availability

1. Application must be available 24 hours a day.
2. Application must be available 365 days a year.

Reliability

1. Application must always present the correct information for each user.
2. Application must analyse the correct information and display to screen.
3. Invalid inputs must be properly reported as such.
4. Application must always maintain data integrity
   4.1 Table relationships must always be consistent.
   4.2 Foreign key fields must agree with the primary key that is referenced by the foreign key.
   4.3 Ensuring that changes made to the database by authorized users do not result in the loss or confliction of data consistency.
5. The application failure rate shall be less than 1 failure per 1000 hours of operation.

Performance

1. Application must not exceed given amount of memory.
2. Web application must be performed within milliseconds.
3. Application must not lag or freeze when in use.

Usability

1. Layout should be simple and clean.
2. Buttons must be of equal size and shape.
3. Application interface must be visually appealing and easy to use.
4. Use at least font size 17 with contrasting backgrounds.

Scalability

1. Application must be able to work with different languages.

Compatibility

1. The application must be compatible on every browser i.e Chrome, Firefox, IE, etc.

Security

1. User application must lock login after 5 incorrect password attempts.

- user must contact administrator to reactivate log in.

2.User data must be encrypted during application synchronization with back end database to provide secure transfer of sensitive information such as personal and financial details.

Jenkins

Introduction

Continuous Integration is the most important part of DevOps that is used to integrate various DevOps stages. Jenkins is the most famous Continuous Integration tool.

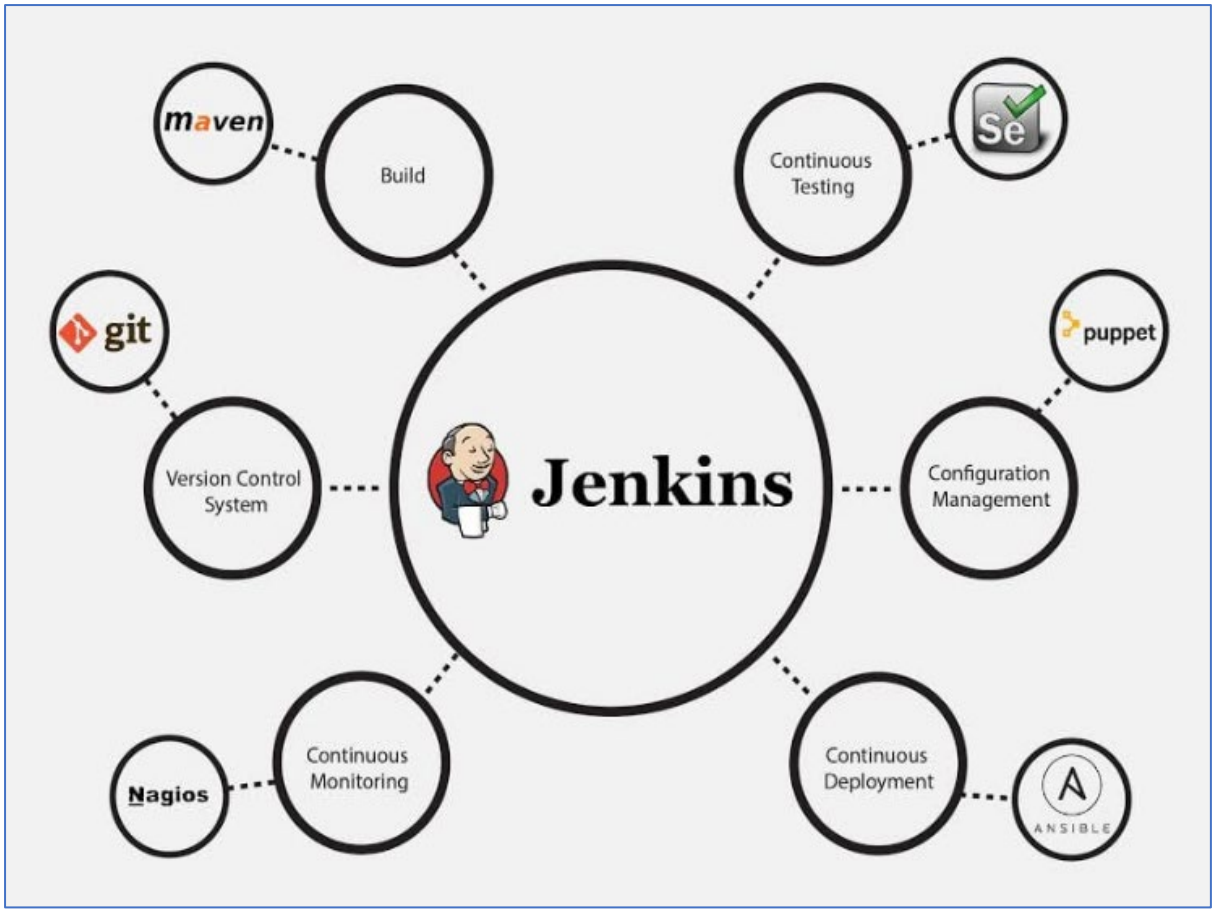
WHAT IS Jenkins?

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with many testing and deployment technologies.

With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis and much more.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allows the integration of Various DevOps stages. If you want to integrate a tool, you need to install the plugins for that tool. For example: Git, Maven 2 project, Amazon EC2, HTML publisher etc.

The image below depicts that Jenkins is integrating various DevOps stages:



Advantages of Jenkins include:

- It is an open source tool with great community support.
- It is easy to install.
- It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share with the community.
- It is free of cost.
- It is built with Java and hence, it is portable to all the major platforms.

There are certain things about Jenkins that separates it from other the Continuous Integration tool.

Jenkins Key Metrics

Following are some facts about Jenkins that makes it better than other Continuous Integration tools:

- Adoption: Jenkins is widespread, with more than 147,000 active installations and over 1 million users around the world.
- Plugins: Jenkins is interconnected with well over 1,000 plugins that allow it to integrate with most of the development, testing and deployment tools.

It is evident from the above points that Jenkins has a very high demand globally. Before we dive into Jenkins it is important to know what is Continuous Integration and why it was introduced.

What is Continuous Integration?

Continuous Integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently. Every commit made in the repository is then built. This allows the teams to detect the problems early. Apart from this, depending on the Continuous Integration tool, there are several other functions like deploying the build application on the test server, providing the concerned teams with the build and test results etc.

Let us understand its importance with a use-case.

Continuous Integration in Nokia

I am pretty sure you all have used Nokia phones at some point in your life. In a software product development project at Nokia there was a process called Nightly builds. Nightly builds can be thought of as a predecessor to Continuous Integration. It means that every night an automated system pulls the code added to the shared repository throughout the day and builds that code. The idea is quite similar to Continuous Integration, but since the code that was built at night was quite large, locating and fixing of bugs was a real pain. Due to this, Nokia adopted Continuous Integration (CI). As a result, every commit made to the source code in the repository was built. If the build result shows that there is a bug in the code, then the developers only need to check that particular commit. This significantly reduced the time required to release new software.

Continuous Integration With Jenkins

Let us imagine a scenario where the complete source code of the application was built and then deployed on test server for testing. It sounds like a perfect way to develop a software, but, this process has many flaws. I will try to explain them one by one:

- Developers have to wait till the complete software is developed for the test results.
- There is a high possibility that the test results might show multiple bugs. It was tough for developers to locate those bugs because they have to check the entire source code of the application.
- It slows the software delivery process.
- Continuous feedback pertaining to things like coding or architectural issues, build failures, test status and file release uploads was missing due to which the quality of software can go down.
- The whole process was manual which increases the risk of frequent failure.

It is evident from the above stated problems that not only the software delivery process became slow but the quality of software also went down. This leads to customer dissatisfaction. So to overcome such a chaos there was a dire need for a system to exist where developers can continuously trigger a build and test for every change made in the source code. This is what CI is all about. Jenkins is the most mature CI tool available so let us see how Continuous Integration with Jenkins overcame the above shortcomings.

I will first explain you a generic flow diagram of Continuous Integration with Jenkins so that it becomes self explanatory, how Jenkins overcomes the above shortcomings:

Download

The following is a link to the open-source version of Jenkins: https://jenkins.io/

Benefits

CI/CD

As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

installation

Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Mac OS X and other Unix-like operating systems.

Configuration

Jenkins can be easily set up and configured via its web interface, which includes on-the-fly error checks and built-in help.

Plugins

With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain.

Extensible

Jenkins can be extended via its plugin architecture, providing nearly infinite possibilities for what Jenkins can do.

Distributed

Jenkins can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.

Jenkins

Video: https://www.youtube.com/watch?v=WWcijE7ifcA

Source: https://www.edureka.co/blog/what-is-jenkins/
https://jenkins.io/

Marking Scheme:

An individual report is required from each learner to describe the project in terms of each of the areas below. Each team member must be familiar with each area regardless of who creates the original implementation. The project is small so that you can focus on Why more than How. Refer to the following draft marking scheme:

Desc	Pts	Desc	Pts
Performance Management Tool(s)	5	Agile dev/team collaboration	5
Unit Testing – automated Sonarqube or similar	5	Scrum	10
Jira	5	GIT & GitHub (pull – push)	5
Security Tools (Greenlight)	5	Staging/Production Environments	10
Artifactory	5	Conclusions	20
Automated documentation	5		
Code Created (include consideration for security, performance, etc.)	10		
Jenkins Pipeline	10		
Total	**50**		**50**

Monit

Monit is a utility for managing and monitoring processes, programs, files, directories and filesystems on a Unix system. Monit conducts automatic maintenance and repair and can execute meaningful causal actions in error situations. E.g. Monit can start a process if it does not run, restart a process if it does not respond and stop a process if it uses too much resources. You can use Monit to monitor files, directories and filesystems for changes, such as timestamps changes, checksum changes or size changes.

Monit is controlled via an easy to configure control file based on a free-format, token-oriented syntax. Monit logs to syslog or to its own log file and notifies you about error conditions via customisable alert messages. Monit can perform various TCP/IP network checks, protocol checks and can utilise SSL for such checks. Monit provides a HTTP(S) interface and you may use a browser to access the Monit program.

Installing Monit on Ubuntu

Step 1: Install Monit Package

sudo apt update

sudo apt install monit

After installing Monit, the commands below can be used to stop, start and enable Monit service….

sudo systemctl stop monit.service

sudo systemctl start monit.service

sudo systemctl enable monit.service

Step 2: Configure Monit Service

Open Monit main config file and make the highlighted changes below, then save the file..

sudo nano /etc/monit/monitrc

The highlighted changes will allow HTTP access to Monit web interface…

Monit has an embedded HTTP interface which can be used to view status of

services monitored and manage services from a web interface. The HTTP

interface is also required if you want to issue Monit commands from the

command line, such as 'monit status' or 'monit restart service' The reason

for this is that the Monit client uses the HTTP interface to send these

```
## commands to a running Monit daemon. See the Monit Wiki if you want to
## enable SSL for the HTTP interface.
#
 set httpd port 2812 and
     use address localhost  # only accept connection from localhost
     allow localhost        # allow localhost to connect to the server and
     allow admin:monit      # require user 'admin' with password 'monit'
#    #with ssl {            # enable SSL/TLS and set path to server certificate
#    #    pemfile: /etc/ssl/certs/monit.pem
#    #}
```

Restart Monit service by running the commands below:

```
sudo systemctl restart monit.service
```

Configuring The System Services

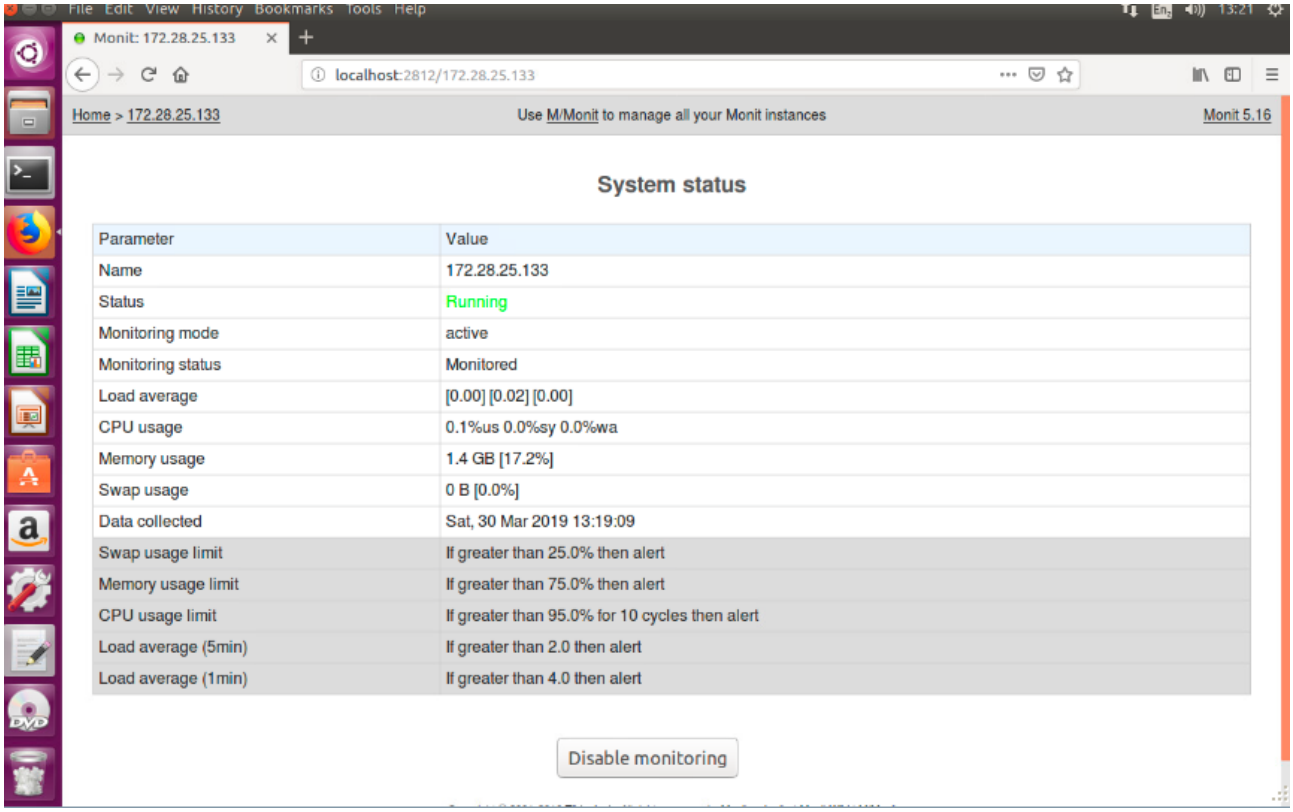
The System

The example below, demonstrate how to test general key performance numbers on your host, such as load average, memory usage and CPU usage. The CPU usage parts, user, system and wait, can be tested individually. The $HOST variable is expanded by Monit to the host's DNS name. If your host does not have a DNS name, just write a string, naming your host and this name will be used as the host-name in alerts and in Monit's UI.

```
check system $HOST
    if loadavg (5min) > 3 then alert
    if loadavg (15min) > 1 then alert
    if memory usage > 80% for 4 cycles then alert
    if swap usage > 20% for 4 cycles then alert
    # Test the user part of CPU usage
    if cpu usage (user) > 80% for 2 cycles then alert
    # Test the system part of CPU usage
    if cpu usage (system) > 20% for 2 cycles then alert
    # Test the i/o wait part of CPU usage
    if cpu usage (wait) > 80% for 2 cycles then alert
    # Test CPU usage including user, system and wait. Note that
    # multi-core systems can generate 100% per core
    # so total CPU usage can be more than 100%
```

if cpu usage > 200% for 4 cycles then alert

By following for the above process for crowsoft , below was the result on monitoring the application



Date Created: 18/04/2019
Created By: Charles Aylward

MVC App Security and Roles

Contents

Date Created: 18/04/2019
Created By: Charles Aylward

ASP.Net Security & Roles

In this document, you will find how I added roles to the database, and how I link users to a specific role, and how I added a policy based authentication method. I will also explain in this document how to limit a user not have access to specific views, controllers or functions.

MySQL Database Changes

The following database changes were required for the Policy Based authentication to work:

- Added two new columns to AspNetRoles table
  - ConcurrencyStamp : longtext
  - NormalizedName : varchar(256)
- The following records where added manually on the AspNetRoles MySQL table
  - Admin, Client and User

Id	Name	ConcurrencyStamp	NormalizedName
1	Admin	NULL	ADMIN
2	Client	NULL	CLIENT
3	User	NULL	USER

- For the security to work, I need to add a new table, called AspNetRoleClaims (Note: This table is not in use, but the application will not work due to the ASP.Net Identity manager searching for this table when adding users to roles in code. Table script is checked into the CrowSoftSQL Script sql file.)

Code Changes on crowsoftmvc

There was several code changes and new classes and I will break it down in different sections.

Changes to Startup.cs

To use the out of the box identity role components, it was necessary to add AddRoles<IdentityRole> to the this code, as seen highlited in brown below.

```
// This is to add the Identity components to the application, AddRoles adds the role components
services.AddDefaultIdentity<crowsoftmvcUser>()
        .AddRoles<IdentityRole>()
        .AddEntityFrameworkStores<ApplicationDbContext>();
```

The following code shows you how I added a build command below to AddMvc to add policy components to the application. This is to authorize a use role into a specific policy.

```
services.AddMvc(obj =>
{
    // This code adds Policy components to the application, to be able to the Authorize users against specific
policies
    var policy = new AuthorizationPolicyBuilder()
                .RequireAuthenticatedUser()
                .Build();
 }).SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
```

Date Created: 18/04/2019
Created By: Charles Aylward

Policies were added to control who can access the controllers:

- RequireAdminOnly = User with Admin Roles Only
- AllUsers = All user roles, Admin, Client and User
- RequireUserandAdminOnly = CrowSoft Users only, Admin and User

```
services.AddAuthorization(options =>
{
        options.AddPolicy("RequireAdminOnly", policy =>
                policy.RequireRole("Admin"));
        options.AddPolicy("AllUsers", policy =>
                policy.RequireRole("Admin", "Client", "User"));
        options.AddPolicy("RequireUserandAdminOnly", policy =>
                policy.RequireRole("Admin", "User"));
});
```

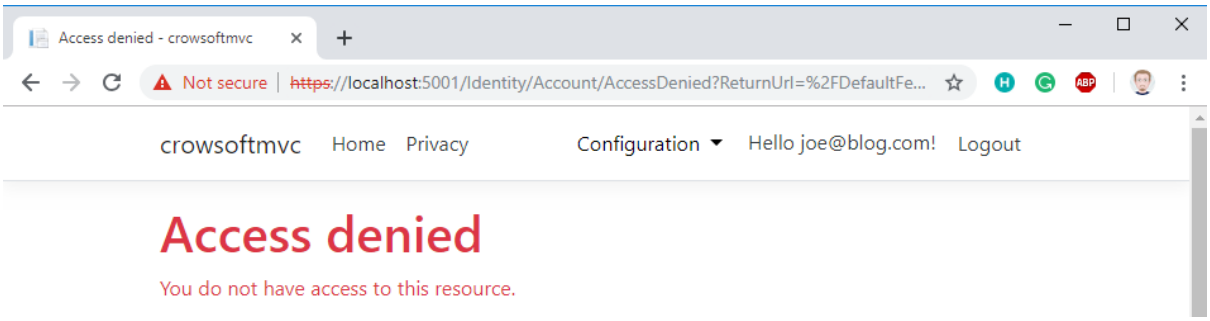
Limit Access To Controllers

The following code example is needed on the top of the each controller class for the policy to be active on a specific controller. E.g. [Authorize(Policy = "RequireAdminOnly")]

Example of the Authorize Controller:

```
[Authorize(Policy = "RequireAdminOnly")]
public class DefaultFeatureController : Controller
{
```

Below is what a user will see when they try to access this controller's view:



Administrator Tasks

We require the administrator to be able to view, edit and delete users. The following few headings will go in more detail how this was done.

Update Users Model

To be able to create a Controller and View from the Identify model, called CrowsoftUser in our project, needed to override the columns to add data annotations for the views to reflect these annotations.

Date Created: 18/04/2019
Created By: Charles Aylward

Here is the update to the CrowSoft Class:

```
    // Add profile data for application users by adding properties to the CrowsoftUser class
    public class CrowsoftUser : IdentityUser
    {
        [Key]
        public override string Id { get => base.Id; set => base.Id = value; }

        [Display(Name = "User Name")]
        public override string UserName { get => base.UserName; set => base.UserName = value; }

        [HiddenInput(DisplayValue = false)]
        public override string NormalizedUserName { get => base.NormalizedUserName; set => base.NormalizedUserName =
value; }

        [Display(Name = "Email Address")]
        [EmailAddress(ErrorMessage = "Invalid Email Address")]
        [Required(ErrorMessage = "{0} is required.")]
        public override string Email { get => base.Email; set => base.Email = value; }

        [HiddenInput(DisplayValue = false)]
        public override string NormalizedEmail { get => base.NormalizedEmail; set => base.NormalizedEmail = value; }

        [Display(Name = "Confirm Email Address")]
        [EmailAddress(ErrorMessage = "Invalid Email Address")]
        [Required(ErrorMessage = "{0} is required.")]
        public override bool EmailConfirmed { get => base.EmailConfirmed; set => base.EmailConfirmed = value; }

        [HiddenInput(DisplayValue = false)]
        public override string PasswordHash { get => base.PasswordHash; set => base.PasswordHash = value; }

        [HiddenInput(DisplayValue = false)]
        public override string SecurityStamp { get => base.SecurityStamp; set => base.SecurityStamp = value; }

        [HiddenInput(DisplayValue = false)]
        public override string ConcurrencyStamp { get => base.ConcurrencyStamp; set => base.ConcurrencyStamp = value;
}

        [Display(Name = "Phone Number")]
        public override string PhoneNumber { get => base.PhoneNumber; set => base.PhoneNumber = value; }

        [HiddenInput(DisplayValue = false)]
        public override bool PhoneNumberConfirmed { get => base.PhoneNumberConfirmed; set =>
base.PhoneNumberConfirmed = value; }

        [HiddenInput(DisplayValue = false)]
        public override bool TwoFactorEnabled { get => base.TwoFactorEnabled; set => base.TwoFactorEnabled = value; }

        [HiddenInput(DisplayValue = false)]
        public override DateTimeOffset? LockoutEnd { get => base.LockoutEnd; set => base.LockoutEnd = value; }

        [HiddenInput(DisplayValue = false)]
        public override bool LockoutEnabled { get => base.LockoutEnabled; set => base.LockoutEnabled = value; }

        [HiddenInput(DisplayValue = false)]
        public override int AccessFailedCount { get => base.AccessFailedCount; set => base.AccessFailedCount = value;
}
    }
```

The following command was executed in the terminal to auto-genrate the Controllers and Views:

```
dotnet aspnet-codegenerator --project . controller -name CrowsoftUserController -m
CrowsoftUser -dc ApplicationDbContext --useDefaultLayout --referenceScriptLibraries
```

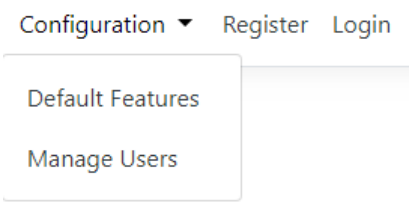
Date Created: 18/04/2019
Created By: Charles Aylward

Changes to add new Menu

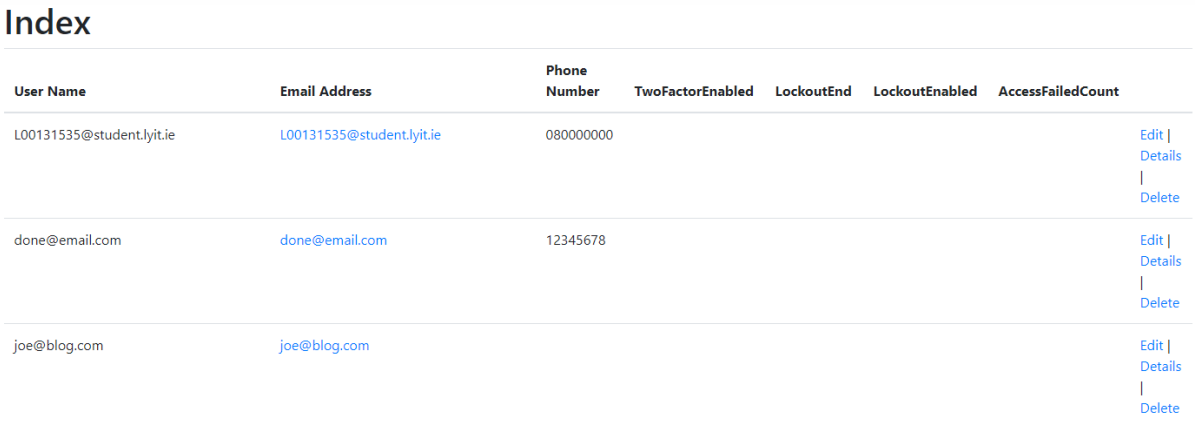
I made the following changes in the _Layout.cshtml to add a menu for Configuration::

```
                <div class="dropdown">
                    <button class="btn btn-primary dropdown-toggle" data-toggle="dropdown"
                            style="background-color:white;color:black;border:none !important;outline:none
!important">
                        Configuration
                    </button>
                    <div class="dropdown-menu">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-controller="DefaultFeature" asp-action="Index">Default
Features</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-controller="CrowsoftUser" asp-action="Index">Manage
Users</a>
                        </li>
                        <li class="nav-item">
                            <a class="dropdown-item" href="#">Link 3</a>
                        </li>
                    </div>
                </div>
```

This is how the menu looks like:



Below find the view to Manage Users:



Index

User Name	Email Address	Phone Number	TwoFactorEnabled	LockoutEnd	LockoutEnabled	AccessFailedCount	
L00131535@student.lyit.ie	L00131535@student.lyit.ie	080000000					Edit | Details | Delete
done@email.com	done@email.com	12345678					Edit | Details | Delete
joe@blog.com	joe@blog.com						Edit | Details | Delete

Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

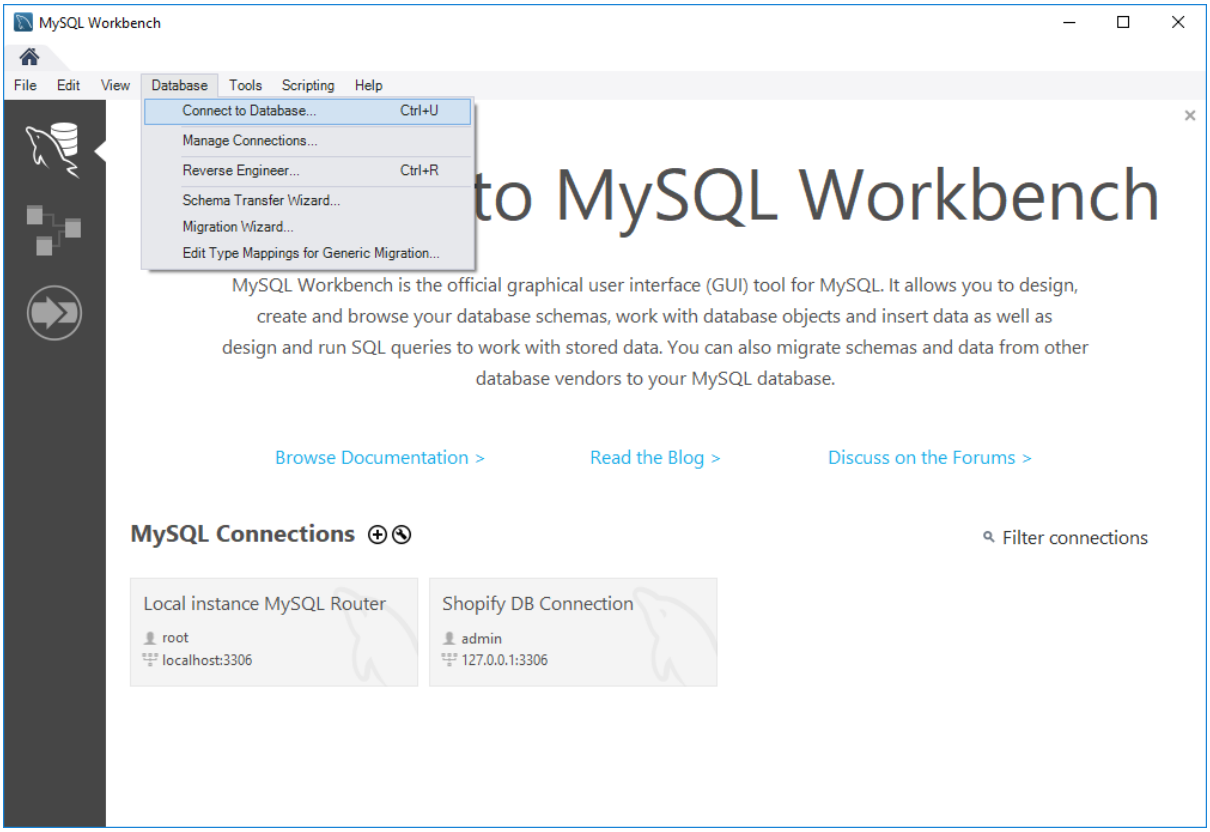
CrowSoft MySQL Script & Admin User

Contents

Log into MySQL Database Remotely

You will need MySQL Workbench if you would like to access the MySQL database from you desktop / home pc. Here is the link (It's open source): https://dev.mysql.com/downloads/workbench/

Open MySQL Workbench, and go to Database, Connect to Database



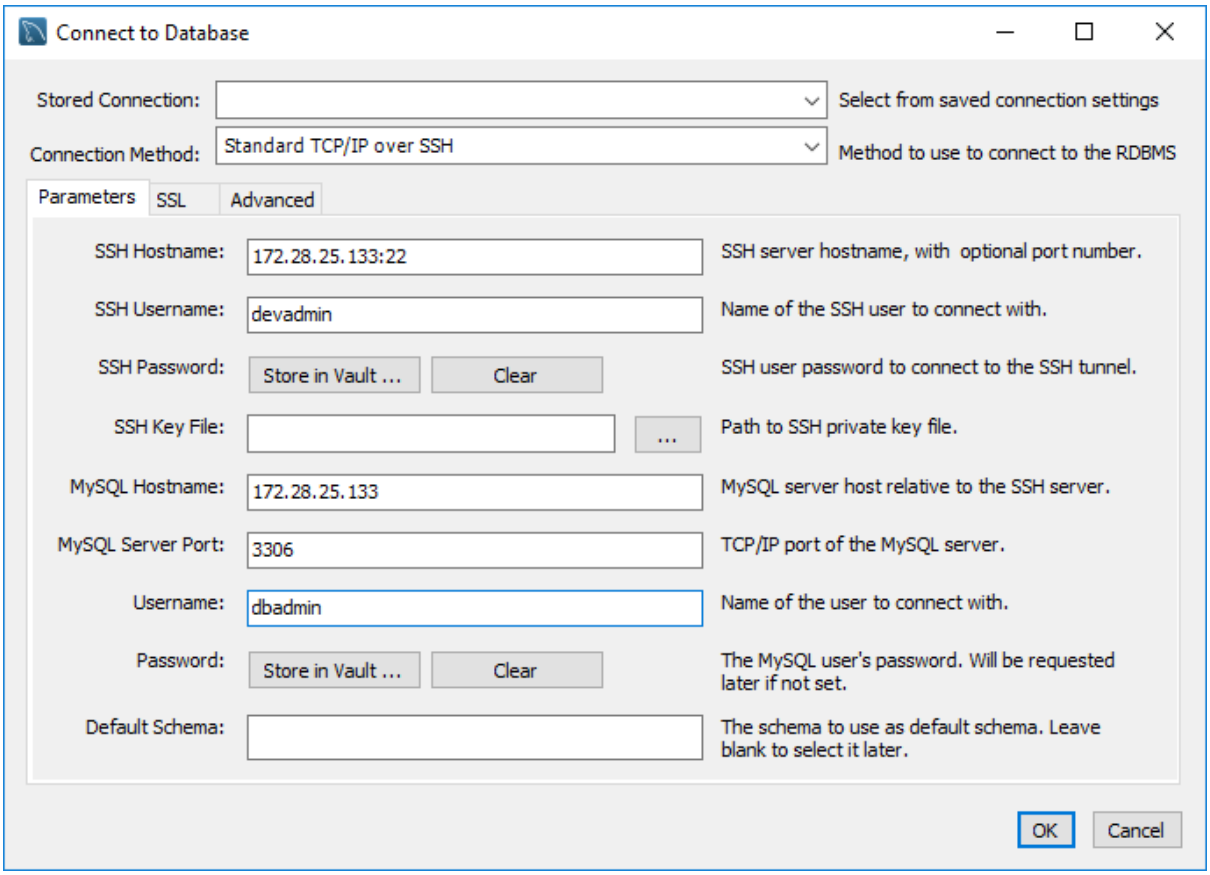
Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

Click on Connection Method and select: Standard TCP/IP over SSH

Change the following settings

- SSH Hostname: 172.28.25.133:22
- SSH Username: devadmin
- SSH Password: refer to dev server password posted on Slack (You can store the password in Vault)
- MySQL Hostname: 172.28.25.133
- MySQL Server port: 3306
- Username: dbadmin
- Password: refer to dev server password posted on Slack (You can store the password in Vault)

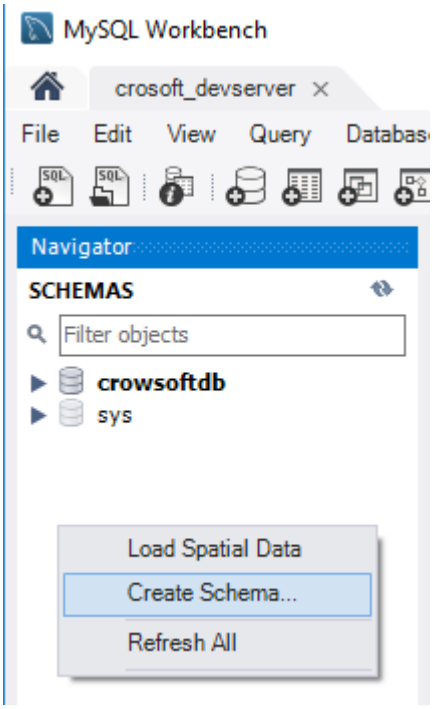
Then click on OK



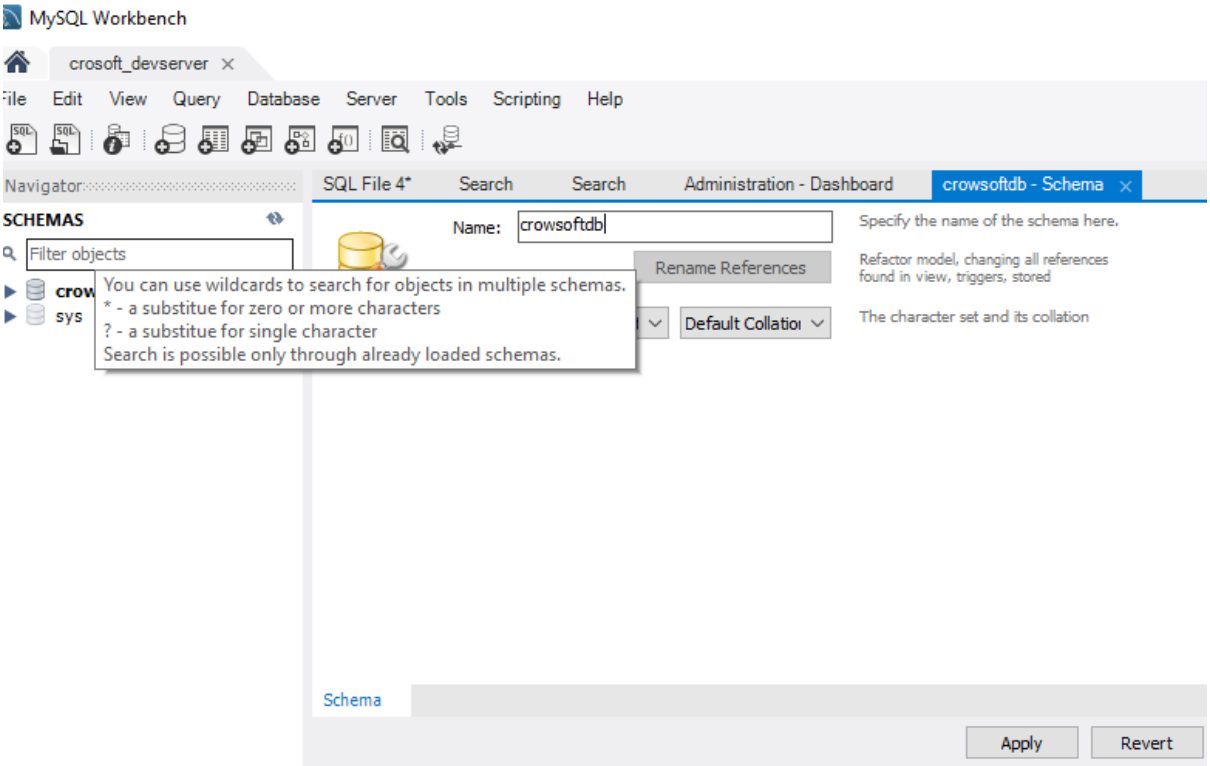
Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

Create Database Schema

Go to the Schemas tab, and right click, then click on Create Schema.



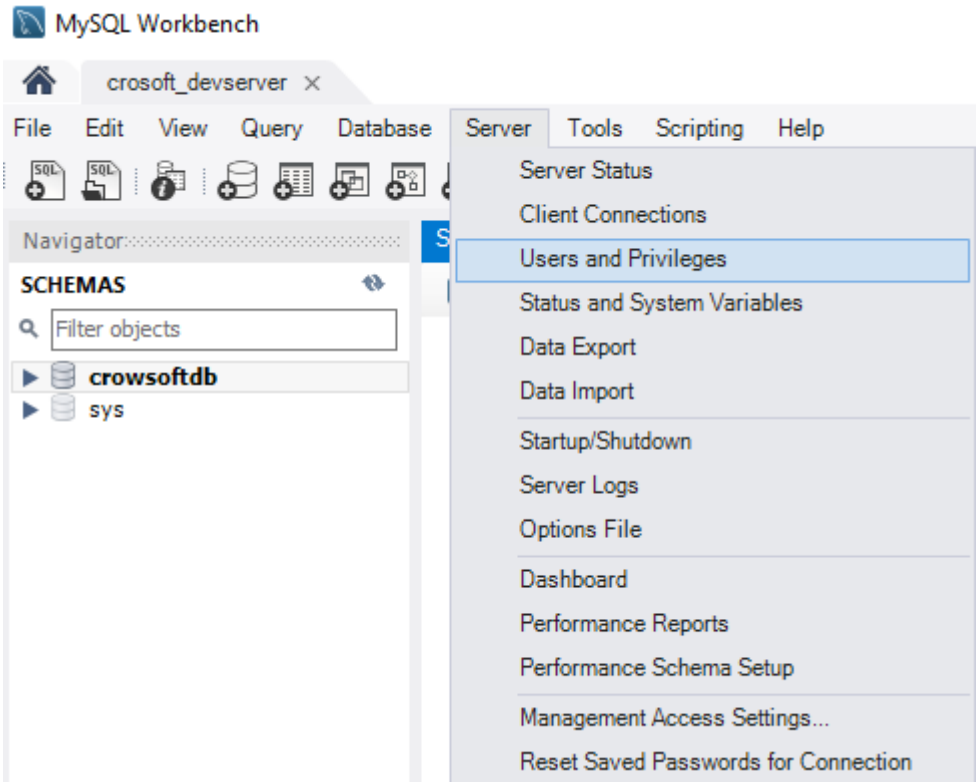
Type in database name in Name column and apply.



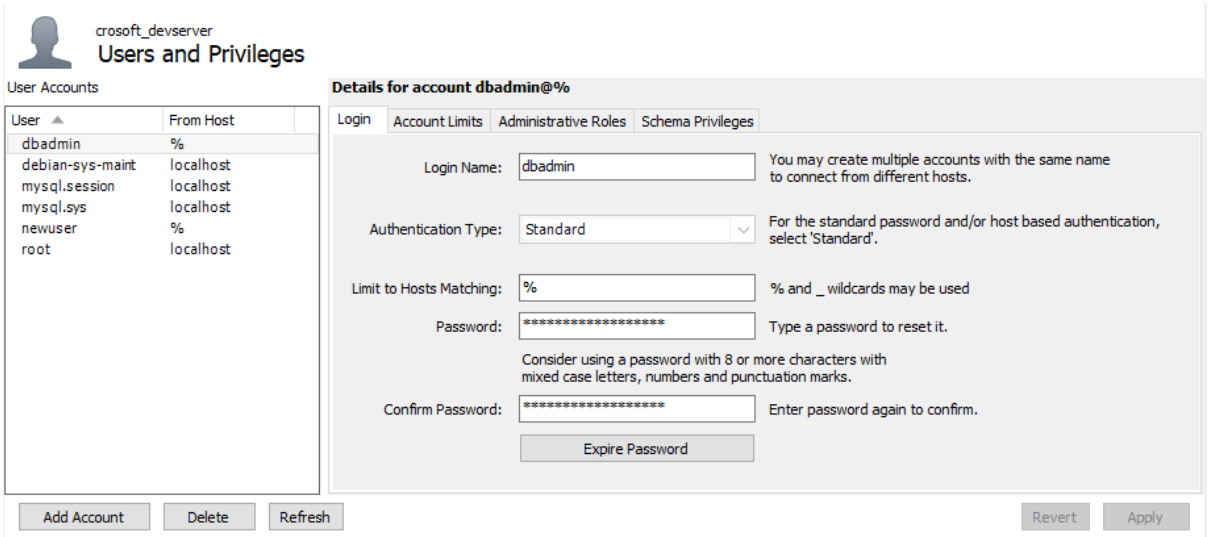
Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

Adding a admin user

Go to Server and click on Users and Privileges

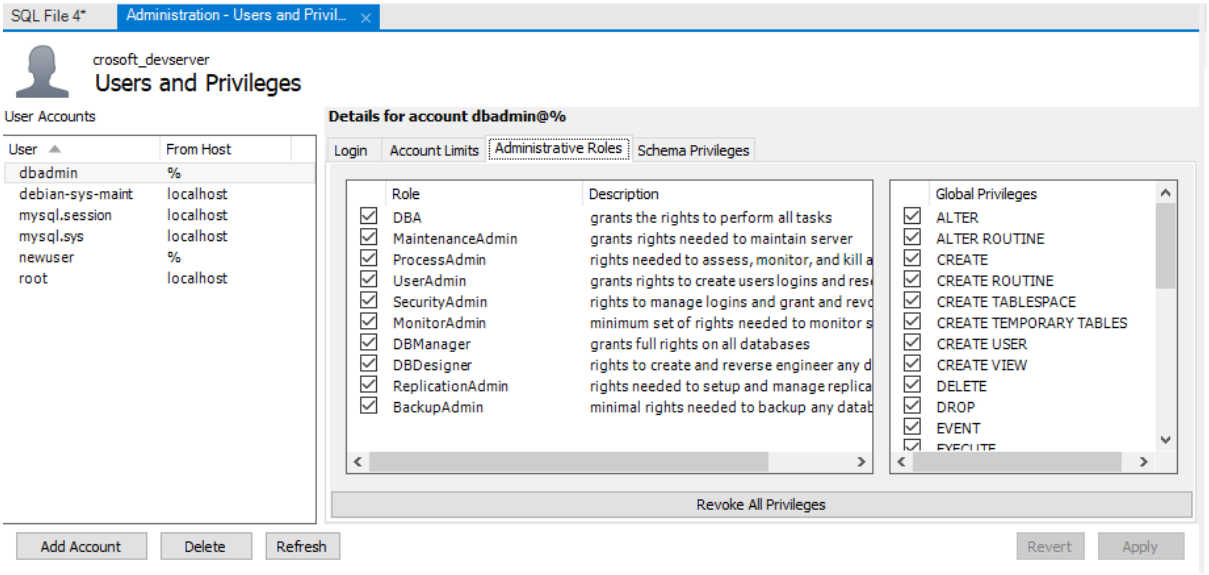


Click on Add Account, enter username and password.



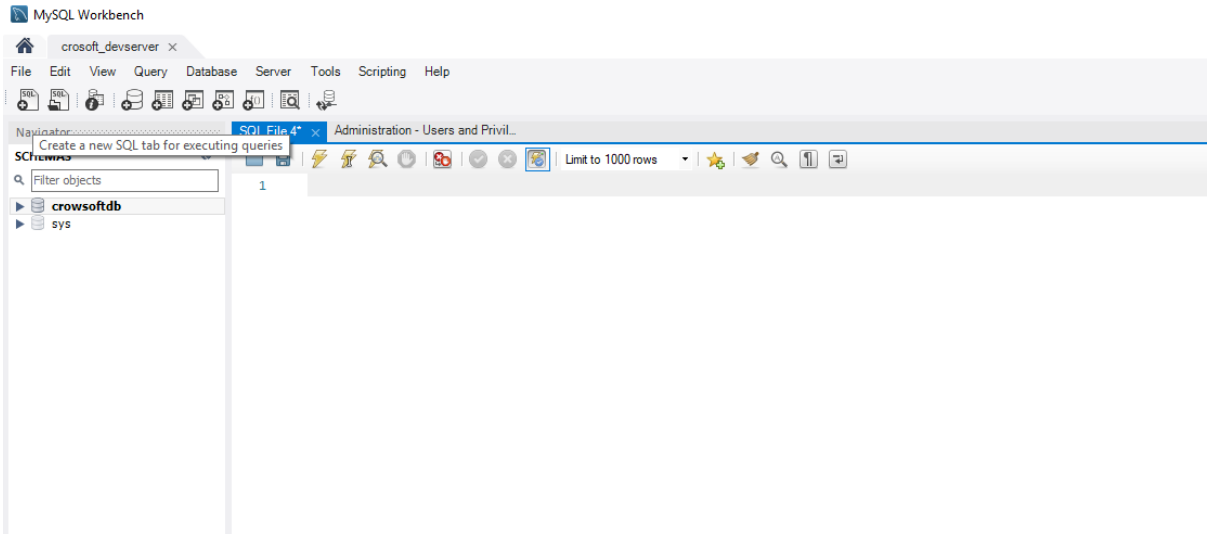
Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

Click on Administrative Roles and all roles and all Global Privileges



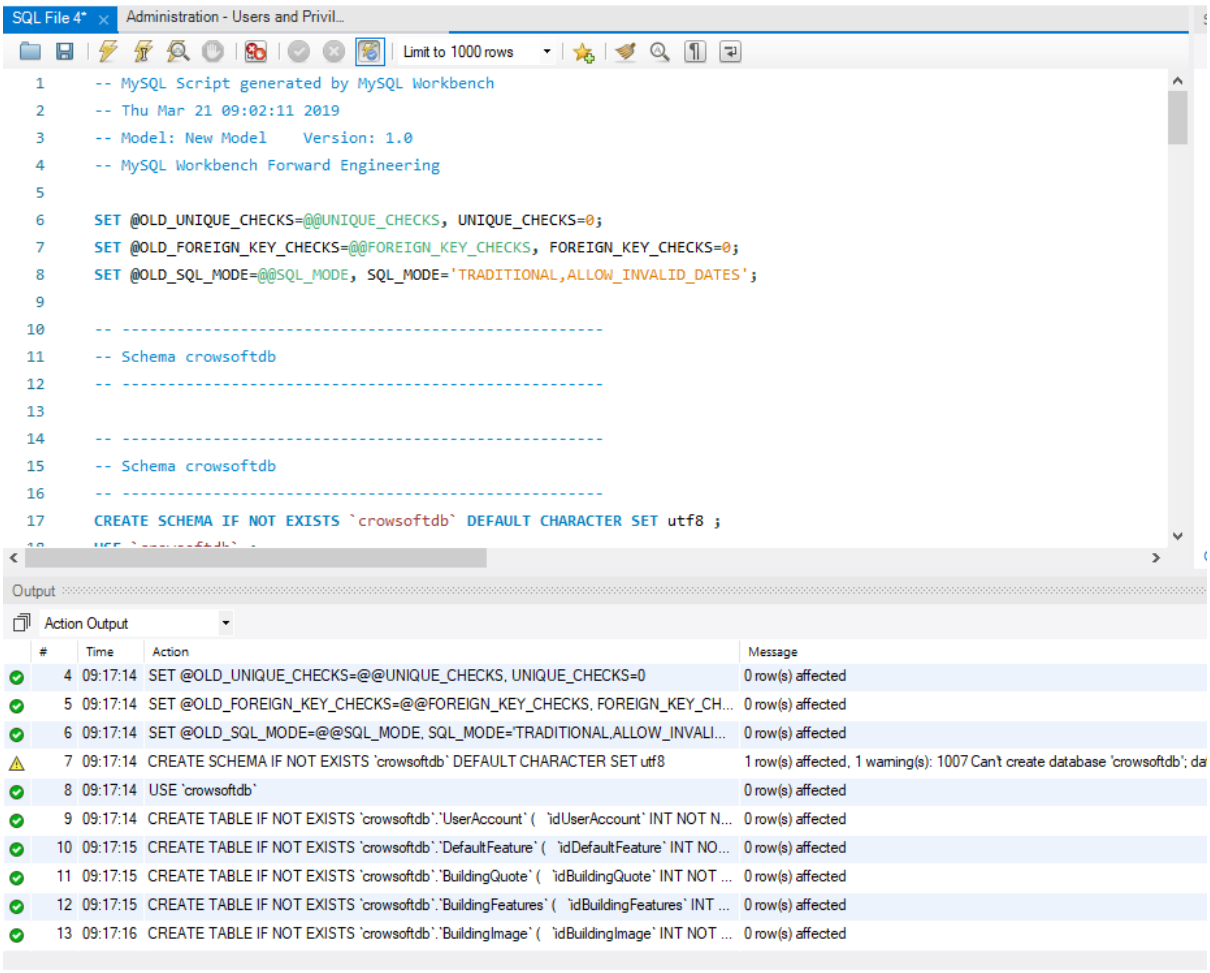
Run SQL script

Click on Create a new SQL tab for executing queries Button.



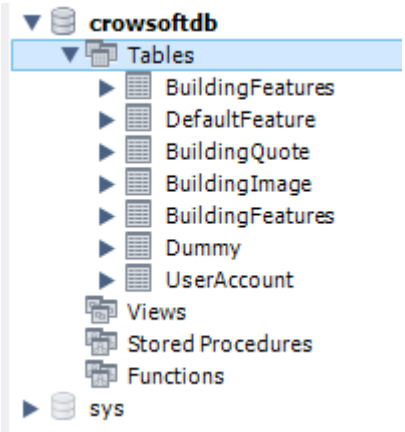
Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

Copy and Paste SQL script (Find script in appendix) and click Execute Icon 



Right click on crowsoftdb and click Refresh All.

Open Tables and view tables.



Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

Appendix

```
-- MySQL Script generated by MySQL Workbench
-- Wed Apr 17 09:10:01 2019
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema crowsoftdb
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `crowsoftdb` ;

-- -----------------------------------------------------
-- Schema crowsoftdb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `crowsoftdb` DEFAULT CHARACTER SET latin1 ;
USE `crowsoftdb` ;

-- -----------------------------------------------------
-- Table `crowsoftdb`.`AspNetRoles`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`AspNetRoles` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`AspNetRoles` (
  `Id` VARCHAR(128) NOT NULL,
  `Name` VARCHAR(256) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`AspNetUsers`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`AspNetUsers` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`AspNetUsers` (
  `Id` VARCHAR(128) NOT NULL,
  `Email` VARCHAR(256) NULL DEFAULT NULL,
  `EmailConfirmed` TINYINT(1) NOT NULL,
  `PasswordHash` LONGTEXT NULL DEFAULT NULL,
  `SecurityStamp` LONGTEXT NULL DEFAULT NULL,
  `PhoneNumber` LONGTEXT NULL DEFAULT NULL,
  `PhoneNumberConfirmed` TINYINT(1) NOT NULL,
  `TwoFactorEnabled` TINYINT(1) NOT NULL,
  `LockoutEnd` DATETIME NULL DEFAULT NULL,
  `LockoutEnabled` TINYINT(1) NOT NULL,
  `AccessFailedCount` INT(11) NOT NULL,
  `UserName` VARCHAR(256) NOT NULL,
  `ConcurrencyStamp` LONGTEXT NULL DEFAULT NULL,
  `NormalizedEmail` VARCHAR(256) NULL DEFAULT NULL,
  `NormalizedUserName` VARCHAR(256) NULL DEFAULT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`AspNetUserClaims`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`AspNetUserClaims` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`AspNetUserClaims` (
  `Id` INT(11) NOT NULL AUTO_INCREMENT,
  `UserId` VARCHAR(128) NOT NULL,
  `ClaimType` LONGTEXT NULL DEFAULT NULL,
  `ClaimValue` LONGTEXT NULL DEFAULT NULL,
  PRIMARY KEY (`Id`),
  UNIQUE INDEX `Id` (`Id` ASC),
  INDEX `UserId` (`UserId` ASC),
  CONSTRAINT `ApplicationUser_Claims`
    FOREIGN KEY (`UserId`)
    REFERENCES `crowsoftdb`.`AspNetUsers` (`Id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

```
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`AspNetUserLogins`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`AspNetUserLogins` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`AspNetUserLogins` (
  `LoginProvider` VARCHAR(128) NOT NULL,
  `ProviderKey` VARCHAR(128) NOT NULL,
  `UserId` VARCHAR(128) NOT NULL,
  PRIMARY KEY (`LoginProvider`, `ProviderKey`, `UserId`),
  INDEX `ApplicationUser_Logins` (`UserId` ASC),
  CONSTRAINT `ApplicationUser_Logins`
    FOREIGN KEY (`UserId`)
    REFERENCES `crowsoftdb`.`AspNetUsers` (`Id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`AspNetUserRoles`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`AspNetUserRoles` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`AspNetUserRoles` (
  `UserId` VARCHAR(128) NOT NULL,
  `RoleId` VARCHAR(128) NOT NULL,
  PRIMARY KEY (`UserId`, `RoleId`),
  INDEX `IdentityRole_Users` (`RoleId` ASC),
  CONSTRAINT `ApplicationUser_Roles`
    FOREIGN KEY (`UserId`)
    REFERENCES `crowsoftdb`.`AspNetUsers` (`Id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `IdentityRole_Users`
    FOREIGN KEY (`RoleId`)
    REFERENCES `crowsoftdb`.`AspNetRoles` (`Id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`UserAccount`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`UserAccount` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`UserAccount` (
  `idUserAccount` INT(11) NOT NULL AUTO_INCREMENT,
  `EmailAddress` VARCHAR(150) NOT NULL COMMENT 'User email address used as username as well. ',
  `FirstName` VARCHAR(60) NOT NULL,
  `LastName` VARCHAR(60) NOT NULL,
  `TelephoneNo` VARCHAR(45) NOT NULL,
  `AddressLine` VARCHAR(150) NULL DEFAULT NULL,
  `County` VARCHAR(45) NULL DEFAULT NULL,
  `Country` VARCHAR(60) NULL DEFAULT NULL,
  `EirCode` VARCHAR(20) NULL DEFAULT NULL,
  `CompanyName` VARCHAR(45) NULL DEFAULT NULL,
  `TypeUser` VARCHAR(15) NOT NULL,
  `DateCreated` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `AspNetUserID` VARCHAR(128) NULL DEFAULT NULL,
  PRIMARY KEY (`idUserAccount`),
  UNIQUE INDEX `idUserAccount_UNIQUE` (`idUserAccount` ASC),
  INDEX `fk_UserAccount_AspNetUsers1_idx` (`AspNetUserID` ASC),
  CONSTRAINT `fk_UserAccount_AspNetUsers1`
    FOREIGN KEY (`AspNetUserID`)
    REFERENCES `crowsoftdb`.`AspNetUsers` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 25
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`BuildingQuote`
-- -----------------------------------------------------
```

Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

```
DROP TABLE IF EXISTS `crowsoftdb`.`BuildingQuote` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`BuildingQuote` (
  `idBuildingQuote` INT(11) NOT NULL AUTO_INCREMENT,
  `UserAccount_idUserAccount` INT(11) NOT NULL,
  `Description` VARCHAR(150) NOT NULL,
  `MeasurementType` VARCHAR(25) NOT NULL,
  `Height` INT(11) NOT NULL,
  `Width` INT(11) NOT NULL,
  `Depth` INT(11) NOT NULL,
  `PurposeOfBuilding` VARCHAR(150) NULL DEFAULT NULL,
  `BuildingSize` INT(11) NULL DEFAULT NULL,
  `TotalCost` DECIMAL(12,2) NULL DEFAULT NULL,
  `DateCreated` DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  `Status` VARCHAR(10) NULL DEFAULT NULL,
  `TimeFrame` INT(11) NULL DEFAULT NULL,
  `DateUpdated` DATETIME NULL DEFAULT NULL,
  `UpdatedById` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`idBuildingQuote`),
  UNIQUE INDEX `idBuildingQuote_UNIQUE` (`idBuildingQuote` ASC),
  INDEX `fk_BuildingQuote_UserAccount_idx` (`UserAccount_idUserAccount` ASC),
  CONSTRAINT `fk_BuildingQuote_UserAccount`
    FOREIGN KEY (`UserAccount_idUserAccount`)
    REFERENCES `crowsoftdb`.`UserAccount` (`idUserAccount`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`DefaultFeature`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`DefaultFeature` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`DefaultFeature` (
  `idDefaultFeature` INT(11) NOT NULL AUTO_INCREMENT,
  `Description` VARCHAR(100) NOT NULL,
  `Measurement` VARCHAR(20) NOT NULL,
  `UnitPrice` DECIMAL(8,2) NOT NULL,
  `DefaultFeature` TINYINT(4) NULL DEFAULT NULL,
  PRIMARY KEY (`idDefaultFeature`),
  UNIQUE INDEX `idBuildingCost_UNIQUE` (`idDefaultFeature` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`BuildingFeatures`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`BuildingFeatures` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`BuildingFeatures` (
  `idBuildingFeatures` INT(11) NOT NULL AUTO_INCREMENT,
  `FeatureDescription` VARCHAR(100) NOT NULL,
  `Comments` VARCHAR(150) NULL DEFAULT NULL,
  `Quantity` DECIMAL(10,4) NOT NULL,
  `UnitPrice` DECIMAL(10,2) NOT NULL,
  `TotalCost` DECIMAL(12,2) NULL DEFAULT NULL,
  `BuildingQuote_idBuildingQuote` INT(11) NOT NULL,
  `DefaultFeature_idDefaultFeature` INT(11) NOT NULL,
  PRIMARY KEY (`idBuildingFeatures`, `BuildingQuote_idBuildingQuote`, `DefaultFeature_idDefaultFeature`),
  UNIQUE INDEX `idBuildingFeatures_UNIQUE` (`idBuildingFeatures` ASC),
  INDEX `fk_BuildingFeatures_BuildingQuote1_idx` (`BuildingQuote_idBuildingQuote` ASC),
  INDEX `fk_BuildingFeatures_DefaultFeature1_idx` (`DefaultFeature_idDefaultFeature` ASC),
  CONSTRAINT `fk_BuildingFeatures_BuildingQuote1`
    FOREIGN KEY (`BuildingQuote_idBuildingQuote`)
    REFERENCES `crowsoftdb`.`BuildingQuote` (`idBuildingQuote`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_BuildingFeatures_DefaultFeature1`
    FOREIGN KEY (`DefaultFeature_idDefaultFeature`)
    REFERENCES `crowsoftdb`.`DefaultFeature` (`idDefaultFeature`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`BuildingImage`
-- -----------------------------------------------------
```

Date: 26/03/2019
Created by: Charles Aylward (Updated: 17/04/2019)

```
DROP TABLE IF EXISTS `crowsoftdb`.`BuildingImage` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`BuildingImage` (
  `idBuildingImage` INT(11) NOT NULL AUTO_INCREMENT,
  `Description` VARCHAR(150) NULL DEFAULT NULL,
  `ImagePath` VARCHAR(200) NULL DEFAULT NULL,
  `BuildingQuote_idBuildingQuote` INT(11) NOT NULL,
  PRIMARY KEY (`idBuildingImage`, `BuildingQuote_idBuildingQuote`),
  UNIQUE INDEX `idBuildingImage_UNIQUE` (`idBuildingImage` ASC),
  INDEX `fk_BuildingImage_BuildingQuote1_idx` (`BuildingQuote_idBuildingQuote` ASC),
  CONSTRAINT `fk_BuildingImage_BuildingQuote1`
    FOREIGN KEY (`BuildingQuote_idBuildingQuote`)
    REFERENCES `crowsoftdb`.`BuildingQuote` (`idBuildingQuote`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`Dummy`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`Dummy` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`Dummy` (
  `PersonID` TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `LastName` VARCHAR(255) NULL DEFAULT NULL,
  `FirstName` VARCHAR(255) NULL DEFAULT NULL,
  `Address` VARCHAR(255) NULL DEFAULT NULL,
  `City` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`PersonID`))
ENGINE = InnoDB
AUTO_INCREMENT = 2
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `crowsoftdb`.`__EFMigrationsHistory`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `crowsoftdb`.`__EFMigrationsHistory` ;

CREATE TABLE IF NOT EXISTS `crowsoftdb`.`__EFMigrationsHistory` (
  `MigrationId` VARCHAR(150) CHARACTER SET 'utf8' NOT NULL,
  `ProductVersion` VARCHAR(32) CHARACTER SET 'utf8' NOT NULL,
  PRIMARY KEY (`MigrationId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Step 1 — Installing MySQL

Install the MySQL server by using the Ubuntu package manager:

sudo apt-get update

sudo apt-get install mysql-server

The installer installs MySQL and all dependencies.

Set the root password username : csoftsql

Step 2 — Testing MySQL

checked the status.

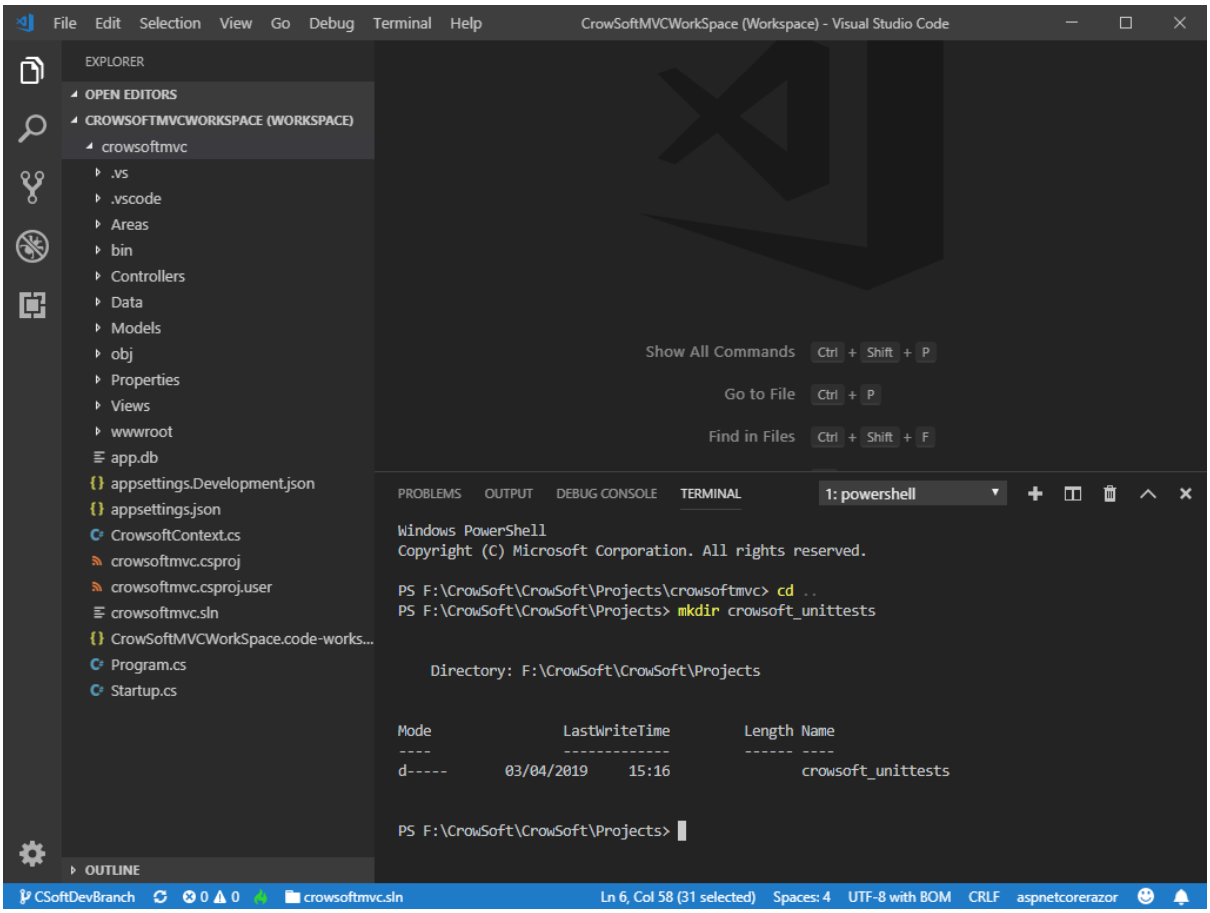
```
systemctl status mysql.service
```

Created by: Charles Aylward
Date Created: 03/04/2019 Updated: 07/04/2019

Nunit Tests on Visual Studio Code

Open Visual Studio Code, and go to your crowsoftmvcworkspace.

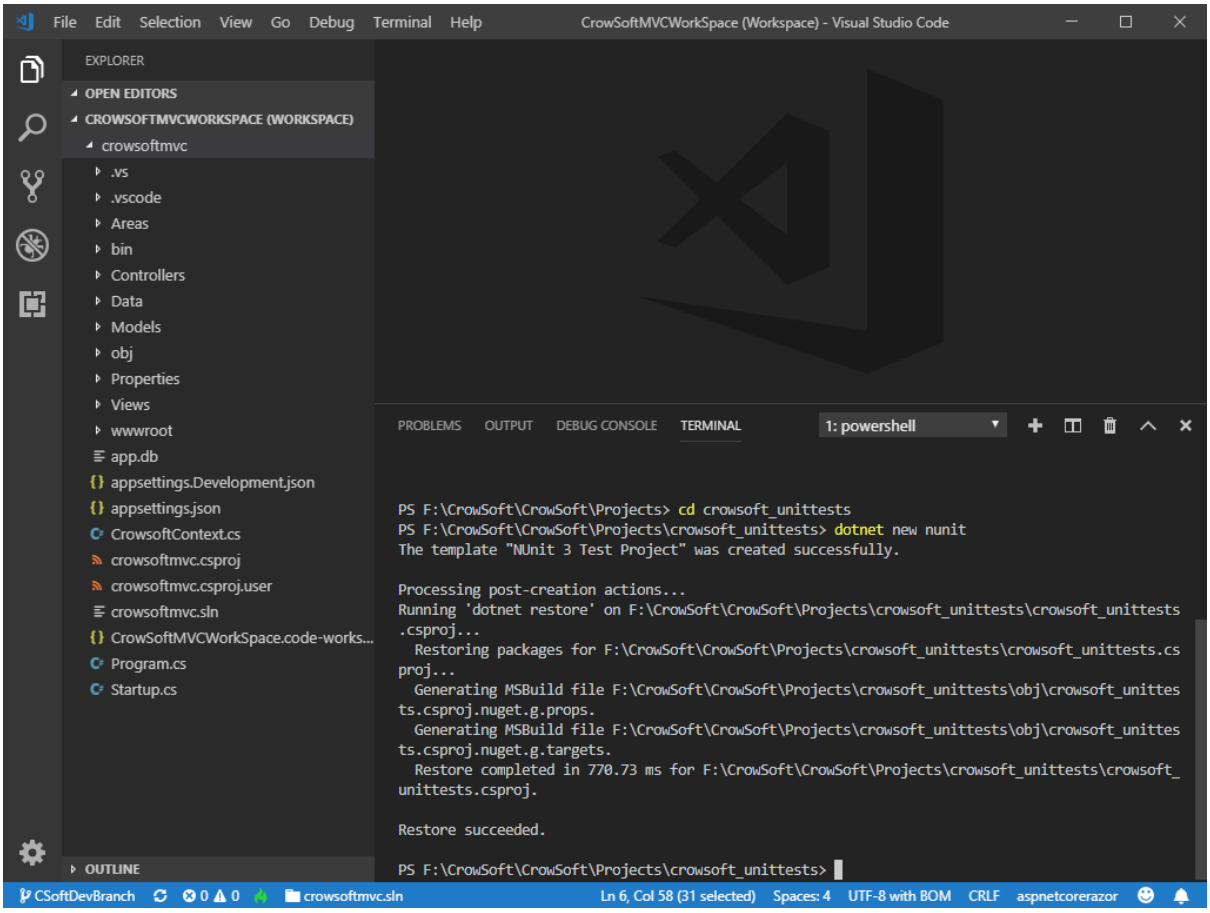
Go to terminal, go back one folder to the Project root, then make a new directory, mkdir crowsoft_unittest



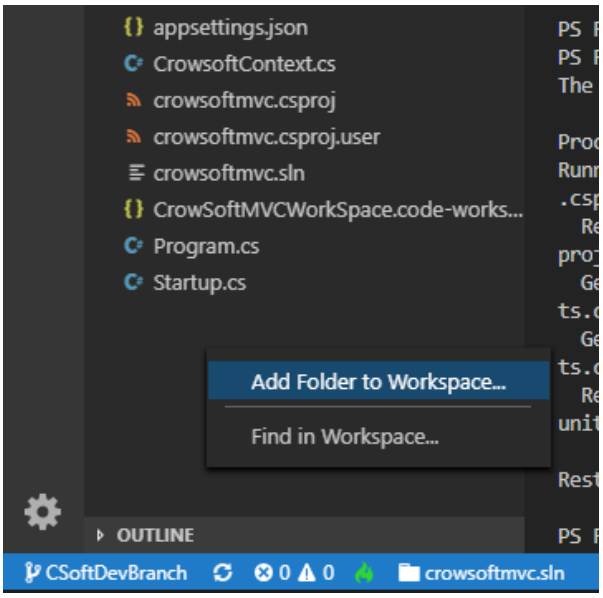
Created by: Charles Aylward
Date Created: 03/04/2019 Updated: 07/04/2019

Go into the new directory by using cd

Type ***dotnet new nunit***, and press enter. Note: This will create the project structure for your nunit tests.

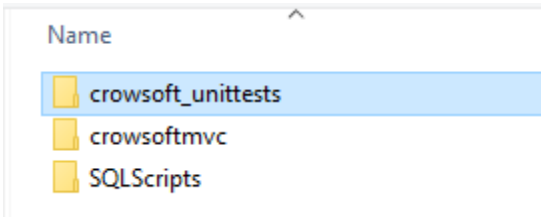


Right click on the open space below the code, then click Add Folder to Workspace.

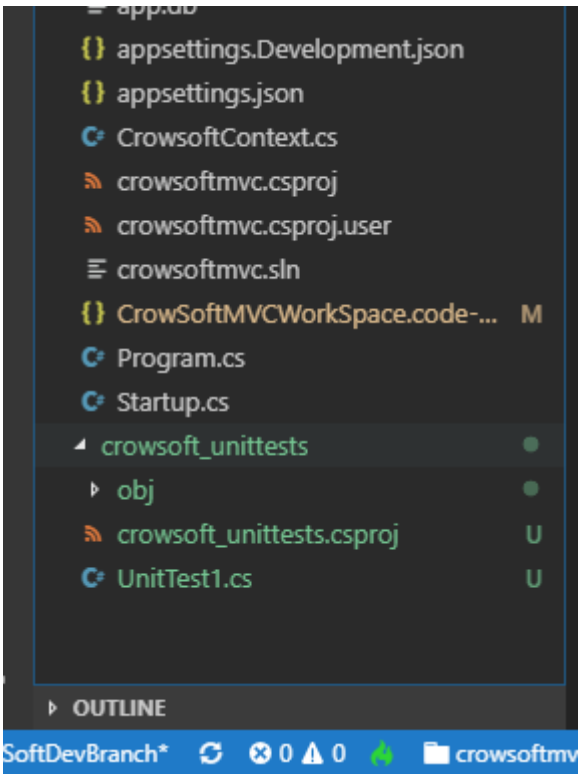


Created by: Charles Aylward
Date Created: 03/04/2019 Updated: 07/04/2019

Select your folder, crowsoft_unittests, and click Add

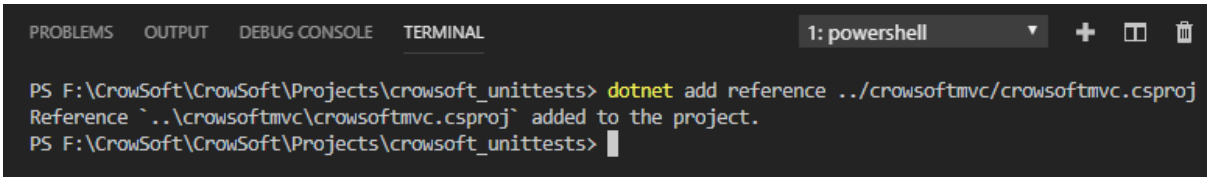


You can see below, crowsoft_unittests folder added to your workspace.



Now we need to reference of our main project to the unit test project.

Open Terminal, and type in the following code and press enter: dotnet add reference ../crowsoftmvc/crowsoftmvc.csproj

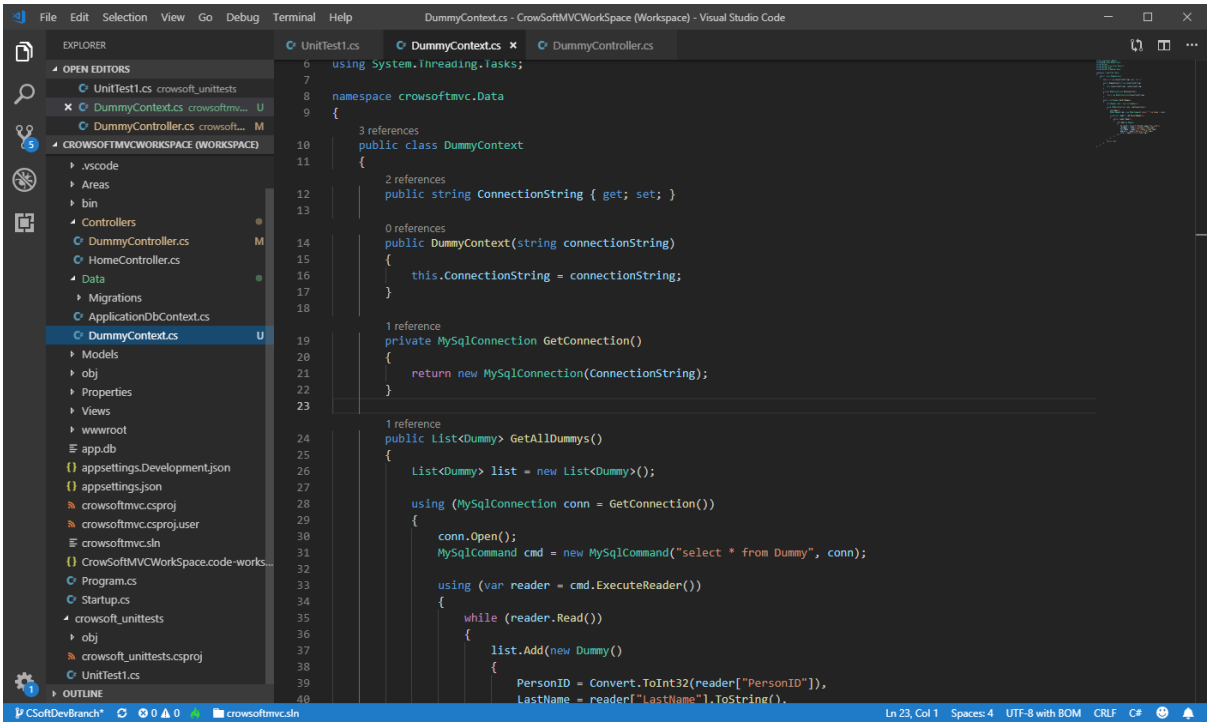


Now the crowsoftmvc.proj is added to the unittests project.

Created by: Charles Aylward
Date Created: 03/04/2019 Updated: 07/04/2019

For the unit test to work, I moved and renamed CrowsoftContext to the Data folder and called it DummyContext. The reason for that is because the context is specific to the Dummy table.

Here is the change below:

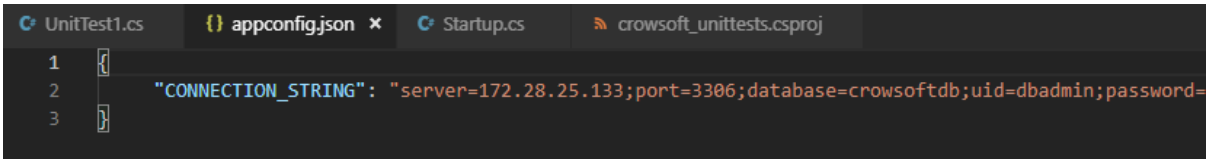


Include the following Package References to your crowsoft_unittests.csproj file.

```
<PackageReference Include="Microsoft.AspNetCore.Mvc.Core" Version="2.2.2" />
    <PackageReference Include="Microsoft.AspNetCore.Razor.Design" Version="2.2.0" />
    <PackageReference Include="Microsoft.AspNetCore.TestHost" Version="2.2.0" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite" Version="2.2.3" />
    <PackageReference Include="Microsoft.Extensions.Configuration" Version="2.2.0" />
    <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="2.2.0" />
    <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="2.2.3" />
    <PackageReference Include="Moq" Version="4.10.1" />
    <PackageReference Include="MySql.Data" Version="8.0.15" />
```

Created by: Charles Aylward
Date Created: 03/04/2019 Updated: 07/04/2019

Add a appconfig.json file. This is to reference the connection string for MySQL.



Open UnitTest1.cs, and added connection string code to setup a connection string for all tests.

First, read the appconfig.json file into a variable called config.

```
var config = new ConfigurationBuilder()
                .AddJsonFile("appconfig.json")
                .Build();
```

Then, read the connectionstring into a local variable, called connection_string.

```
using System.Collections.Generic;
using crowsoftmvc.Data;
using Microsoft.Extensions.Configuration.Json;
using Microsoft.Extensions.Configuration;

namespace Tests
{
    0 references
    public class Tests
    {
        2 references
        private string connection_string;

        // This sets up the test and reads the connectionstring from the appconfig.json file
        [SetUp]
        0 references
        public void Setup()
        {
            var config = new ConfigurationBuilder()
                .AddJsonFile("appconfig.json")
                .Build();

            connection_string = config["CONNECTION_STRING"];
        }
```

Make sure the following using statements are added:

```
using System.Collections.Generic;
using crowsoftmvc.Data;
using Microsoft.Extensions.Configuration.Json;
using Microsoft.Extensions.Configuration;
```

Created by: Charles Aylward
Date Created: 03/04/2019 Updated: 07/04/2019

Add a new method to test if Dummy records are returned. It should return 2 record, which are Greater than 0.

```
// This is a example test, that test if Dummy records are available in the MySQL Database
        [Test]
        public void Test_GetDummyList()
        {
            DummyContext context = new DummyContext(connection_string);
            List<crowsoftmvc.Models.Dummy> myDummyList = context.GetAllDummys();

            Assert.Greater(myDummyList.Count, 0, "Error No Dummy Records Returned");
        }
```

Go to crowsoft_unittests.csproj and add the following into the ItemGroup:

```
<None Update="appconfig.json">
          <CopyToOutputDirectory>Always</CopyToOutputDirectory>
      </None>
```

Example:

```
<ItemGroup>
  <PackageReference Include="Microsoft.AspNetCore.Mvc.Core" Version="2.2.2" />
  <PackageReference Include="Microsoft.AspNetCore.Razor.Design" Version="2.2.0" />
  <PackageReference Include="Microsoft.AspNetCore.TestHost" Version="2.2.0" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite" Version="2.2.3" />
  <PackageReference Include="Microsoft.Extensions.Configuration" Version="2.2.0" />
  <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="2.2.0" />
  <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="2.2.3" />
  <PackageReference Include="Moq" Version="4.10.1" />
  <PackageReference Include="MySql.Data" Version="8.0.15" />
  <PackageReference Include="nunit" Version="3.11.0" />
  <PackageReference Include="NUnit3TestAdapter" Version="3.11.0" />
  <PackageReference Include="Microsoft.NET.Test.Sdk" Version="15.9.0" />
  <None Update="appconfig.json">
          <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </None>
</ItemGroup>
```

Open Terminal for crowsoft_unittests, and type in dotnet test

You should get the following results:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                  1: powershell

Test execution time: 5.2851 Seconds
PS F:\CrowSoft\CrowSoft\Projects\crowsoft_unittests> dotnet test
Build started, please wait...
Build completed.

Test run for F:\CrowSoft\CrowSoft\Projects\crowsoft_unittests\bin\Debug\netcoreapp2.2\crowsoft_unittests.dll(.NETCoreApp,Version=
Microsoft (R) Test Execution Command Line Tool Version 15.9.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

Total tests: 1. Passed: 1. Failed: 0. Skipped: 0.
Test Run Successful.
Test execution time: 1.6014 Seconds
PS F:\CrowSoft\CrowSoft\Projects\crowsoft_unittests>
```

Finish..

Pokerbot

#example how to configure for poker session

/deal config options 1,2,3,5,8

#example how to play a session Adding user @<persons_Slack_name>

/deal @Mary Walsh McGinty @MatthewMcC @Colin @Charles @Michael @JOJI please enter 1, 3, 5, or 8

@Bharathi

@Liam

Scrum Master roles & responsibilities.

Author	Description	Version	Date
Michael McFadden	Scrum Master Roles	1	03/25/2019

Definition:

Scrum is about continuous learning and continuous improvement

The scrum master is the team role responsible for ensuring the team lives agile values and principles and follows the processes and practices that the team agreed they would use.

The responsibilities of this role include:

- Clearing obstacles
- Establishing an environment where the team can be effective
- Addressing team dynamics
- Ensuring a good relationship between the team and product owner as well as others outside the team
- Protecting the team from outside interruptions and distractions.

Roles & responsibilities for Team Crowsoft

- Never Commit the Team to Anything Without Consulting Them First
- Help the team get passed any impediments
- Ensure the PO & all team members are available for Sprint Planning. (no team member will be assigned a story if absent, Scrum Master needs to approve late stories into Sprint)
- Ensure Team update their status daily on csoft_daily_scrum
- Ensure all team members contribute to retrospective, regardless of work done/not done in that Sprint
- Facilitate the meetings and ensure that all relevant parties are in attendance
- Encourage the Team to be self-organising
- Listen to team members and help your team to learn how to solve problems on their own.
- Analyse the Jira dashboard and reporting tools for KPI's for continuous improvement
  Process of how the team works can and should be adapted according to previous Sprint
- Keep morale high amongst the team
- Work with Team to assign story points to backlog items **prior** to Sprint planning

Selenium Testing Setup

Version	Date	Affected Section	Author
1.0	04/04/2019	Initial Draft	Matthew Mc Colgan

Contents

Introduction

Selenium WebDriver is an opensource web automation framework to execute tests against different browsers. You can write your tests in multiple languages including C#, Java, PHP etc. It can operate across multiple OS's and can be integrated with many frameworks including NUnit and other applications like Jenkins.

Prerequisites

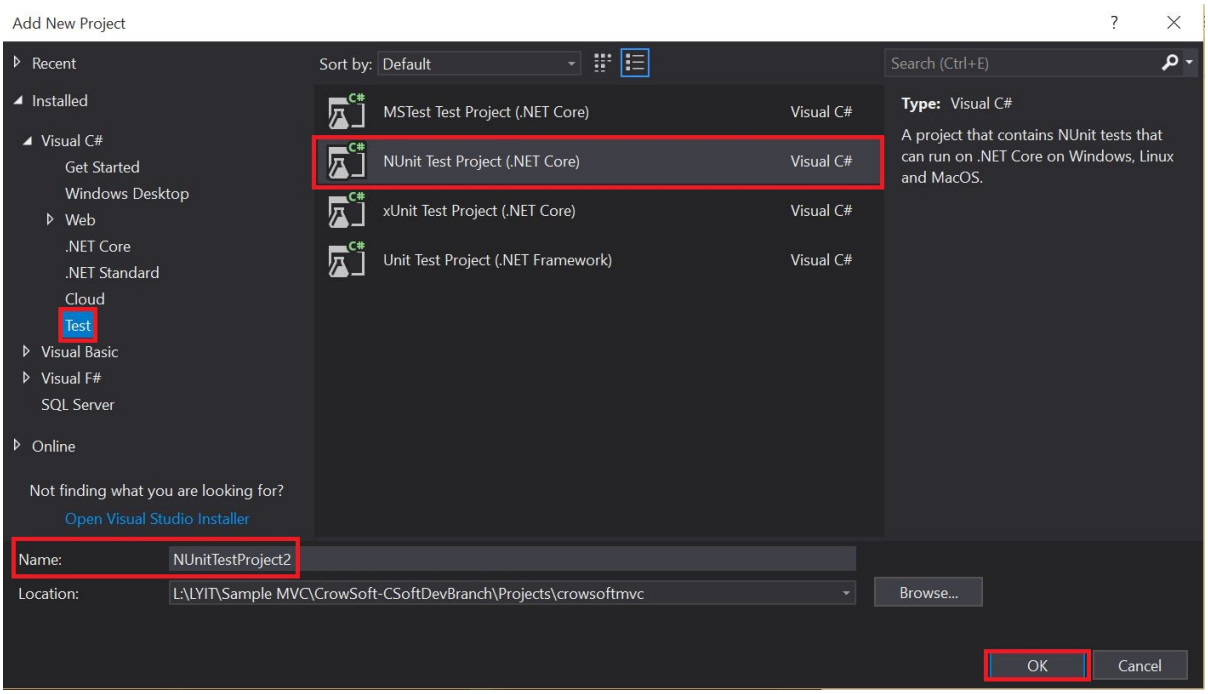
Visual Studio Community Edition

Sample MVC app created by Charles. Can be downloaded/cloned from GitHub @ https://github.com/rlennon/CrowSoft/tree/CSoftDevBranch/Projects/crowsoftmvc

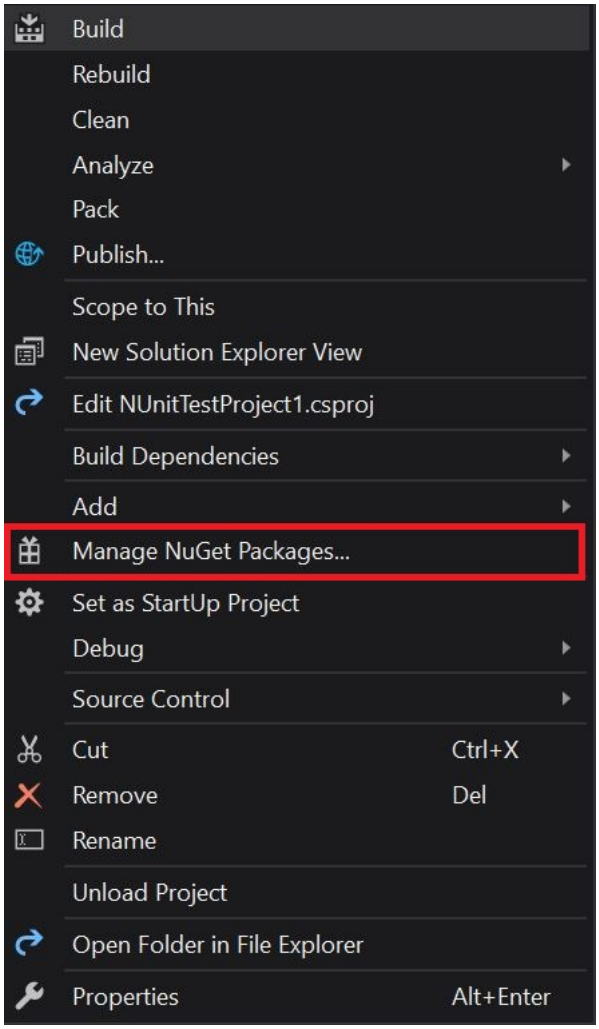
Steps

1. Open the sample MVC app in VS Community
2. Right click on the Solution

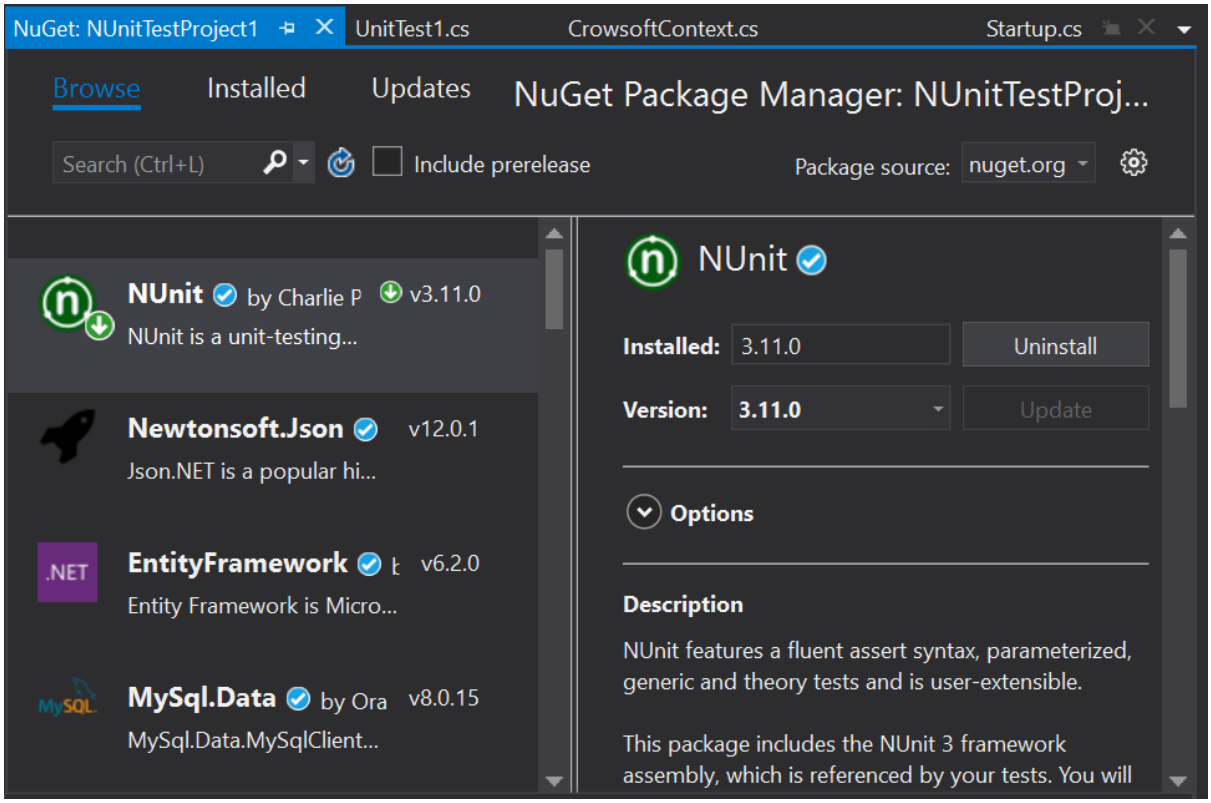
3. Select "Add" and choose "New Project".
4. Select "Test" -> "NUnit Test Project (.NET Core)" and give it a name. Click "OK" when done.



5. Right click on the newly created project and select "Manage NuGet Packages"



6. You can browse for the required NuGet packages using the Browse tab. Select the required package and click install on the pane in the right.

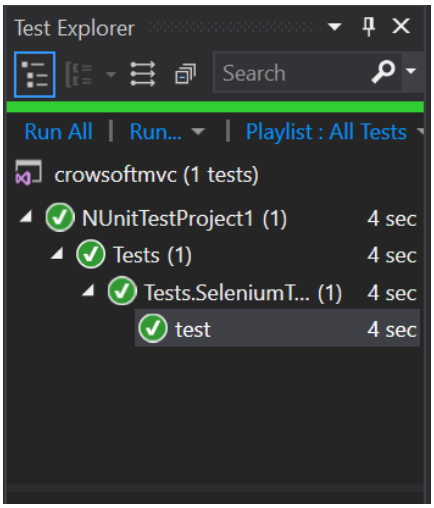


7. Install all of the following NuGet packages:
    a. Microsoft.NET.Test.SDK
    b. Microsoft.NETCore.TestHost
    c. Selenium.Chrome.WebDriver
    d. Selenium.WebDriver
8. Download chromedriver.exe from the following link http://chromedriver.chromium.org/downloads. Make sure to download the driver that corresponds to the version of chrome you are using.
9. Place the chromedriver.exe in your project folder (see code below)**
10. Open the UnitTest1.cs file that was automatically generated when creating the project and paste the following code:

```
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace Tests
{
    public class Tests
    {
        public class SeleniumTests
        {
            IWebDriver driver;

            [SetUp]
            public void startBrowser()
            {
                driver = new ChromeDriver("L:/LYIT/Sample MVC/CrowSoft-
CSoftDevBranch/Projects/NUnitTestProject1/bin/Debug/netcoreapp2.2");
            } **Change path to suit where your chromedriver is installed**
```

```
            [Test]
            public void test()
            {
                driver.Url = "https://localhost:44380";
            }

            [TearDown]
            public void closeBrowser()
            {
                driver.Close();
            }
        }
    }
}
```

11. Open the Test Explorer and run the test.



References

http://chromedriver.chromium.org/downloads

https://www.lambdatest.com/blog/13-reasons-why-selenium-webdriver-should-be-your-first-choice-for-automation-testing/

https://www.guru99.com/selenium-csharp-tutorial.html

Service Level Agreement (SLA)
for *Ruth Lennon*
by
CrowSoft Technologies

Effective Date: 8-04-2019
Start Date: 3-05-2019
End Date: 2-05-2020

Document Owner:	CrowSoft Technologies

Version

Version	Date	Description	Author
1.0	7-04-2019	Service Level Agreement	Charles Aylward

Approval

(By signing below, all Approvers agree to all terms and conditions outlined in this Agreement.)

Approvers	Role	Signed	Approval Date
CrowSoft Director	Service Provider		
Customer	Customer		

*source of this template: (Author, NA) http://www.slatemplate.com/

Table of Contents

1. Agreement Overview

This Agreement represents a Service Level Agreement ("SLA" or "Agreement") between **CrowSoft Technologies**, called the provider and **Ruth Lennon**, called the customer for the provisioning of IT services required to support and sustain the CrowSoft Building Analysis & Business Intelligence Solution.

This Agreement remains valid until superseded by a revised agreement mutually endorsed by the stakeholders.
This Agreement outlines the parameters of all IT services covered as they are mutually understood by the primary stakeholders. This Agreement does not supersede current processes and procedures unless explicitly stated herein.

2. Goals & Objectives

The **purpose** of this Agreement is to ensure that the proper elements and commitments are in place to provide consistent IT service support and delivery to the Customer(s) by the Service Provider(s).

The **goal** of this Agreement is to obtain mutual agreement for IT service provision between the Service Provider(s) and Customer(s).

The **objectives** of this Agreement are to:

- Provide clear reference to service ownership, accountability, roles and/or responsibilities.
- Present a clear, concise and measurable description of service provision to the customer.
- Match perceptions of expected service provision with actual service support & delivery.

3. Stakeholders

The following Service Provider(s) and Customer(s) will be used as the basis of the Agreement and represent the **primary stakeholders** associated with this SLA:

IT Service Provider(s): CrowSoft Technologies. ("Provider")
IT Customer(s): Ruth Lennon ("Customer")

4. Periodic Review

This Agreement is valid from the **Effective Date** outlined herein and is valid until further notice. This Agreement should be reviewed at a minimum once per fiscal year; however, in lieu of a review during any period specified, the current Agreement will remain in effect.

The **Business Relationship Manager** ("Document Owner") is responsible for facilitating regular reviews of this document. Contents of this document may be amended as required, provided mutual agreement is obtained from the primary stakeholders and communicated to all affected parties. The Document Owner will incorporate all subsequent revisions and obtain mutual agreements / approvals as required.

Business Relationship Manager: CrowSoft Technologies
Review Period: Bi-Yearly (6 months)
Previous Review Date: 08-04-2019 (TBD)
Next Review Date: 08-10-2019

5. Service Agreement

The following detailed service parameters are the responsibility of the Service Provider in the ongoing support of this Agreement.

5.1. Service Scope

The following Services are covered by this Agreement;

- Manned telephone support
- Monitored email support
- Remote assistance using Remote Desktop and a Virtual Private Network where available
- Planned or Emergency Onsite assistance (extra costs apply)
- Monthly system health check

5.2. Customer Requirements

Customer responsibilities and/or requirements in support of this Agreement include:

- Payment for all support costs at the agreed interval.
- Reasonable availability of customer representative(s) when resolving a service related incident or request.

5.3. Service Provider Requirements

Service Provider responsibilities and/or requirements in support of this Agreement include:

- Meeting response times associated with service related incidents.
- Appropriate notification to Customer for all scheduled maintenance.

5.4. Service Assumptions

Assumptions related to in-scope services and/or components include:

- Changes to services will be communicated and documented to all stakeholders.

6. Service Management

Effective support of in-scope services is a result of maintaining consistent service levels. The following sections provide relevant details on service availability, monitoring of in-scope services and related components.

6.1. Service Availability

Coverage parameters specific to the service(s) covered in this Agreement are as follows:

- Telephone support : 9:00 A.M. to 5:00 P.M. Monday – Friday
    - Calls received out of office hours will be forwarded to a mobile phone and best efforts will be made to answer / action the call, however there will be a backup answer phone service
- Email support: Monitored 9:00 A.M. to 5:00 P.M. Monday – Friday
    - Emails received outside of office hours will be collected, however no action can be guaranteed until the next working day
- Onsite assistance guaranteed within 72 hours during the business week

6.2. Service Requests

In support of services outlined in this Agreement, the Service Provider will respond to service related incidents and/or requests submitted by the Customer within the following time frames:

- 0-8 hours (during business hours) for issues classified as **High** priority.
- Within 48 hours for issues classified as **Medium** priority.
- Within 5 working days for issues classified as **Low** priority.

Remote assistance will be provided in-line with the above timescales dependent on the priority of the support request.

CrowSoft Social Contract

A Social Contract is a set of agreements that a team makes within itself to describe how the team members will behave and work together

Meetings

- Stand-ups will occur straight after class on Monday night & before class on a Friday morning.
- If you attend a meeting, let the team know on the crowsoftmsc slack channel.
- At stand up we will use the "pass the ball" method for given updates.
- Updates will be in the form: What I've done, What I plan to do, Impediments
- Sprint planning will occur after class every Monday from 9pm to 10pm.
- Sprint retro will occur every week and every member to give their retrospective on the csoft_retrospective channel only.
- Scrum Master to create a Sprint & Retrospective report and uploaded to Git and mailed to group.
- Be on time for Stand Ups and meetings.
- Scrum Master to run meetings and team to use "raised hand "to eliminate multiple voices
- Mic's to be on mute during meetings when others speak to eliminate background noise and make conversation as clear as possible for everyone on meeting.
- Mobile phones on silent
- Everyone has equal voice and valuable contribution.
- Always Keep your language and tone professional.
- Be honest.

Communication

- Channels of communication Slack, Zoom & Mail

  Slack
  - crowsoftmsc – For all CrowSoft project conversation
  - csoft_daily_scrum – For daily stand up comments only (every day regardless of status)
  - csoft_retrospective – For Sprint retrospective comments only (once a week)

  Zoom – For screen sharing and recording sessions

  Mail – For larger statements and attaching files or attachments

- No Slack communications between 10pm and 8am GMT.
- Raise a problem as soon as you see it.
- Respect each other and understand differences in knowledge.
- All team documents are to be shared on the GitHub document branch in the appropriate folder.
- There are no silly questions, if you don't understand, ask.
- Share success stories.
- Focus on the positives.
- Don't make assumptions.
- Zero tolerance for bullying.

DevOps way of working

- If are assigned a job, take ownership of it and keep it up to date.
- Keep JIRA board updated at all times.
- Update your Jira stories as you progress the story in the comments of the story
- Don't be afraid to ask for help.
- Don't be afraid to give constructive criticism, as long as it is constructive.
- Solve roadblocks within the team. If the impediment can't be solved within the team then give it to the Scrum Master.
- Sprints will start Tuesday's and run for 1 week
- The Scrum Master role rotates each week
- Scrum Master's main job is to look after teams interest first and if capacity allows can take on stories.
- Always include a story for the Scrum Master role in every Sprint.
- Each member of the team will work 6 hours per week, unless they are on vacation.

Nothing should go into the Social Contract unless it has complete agreement from all team members.

Only check the box when document is complete, and everyone is in agreement.

Team Members

- Michael McFadden ☐
- Colin Kenny ☐
- Matthew McColgan ☐
- Bharathi Gadhiraju ☐
- Charles Aylward ☐
- Joji Pradhan Thokala ☐
- Mary Walsh McGinty ☐
- Liam Whorriskey ☐

SonarQube

Overview

SonarQube is an automatic code review tool to detect bugs, vulnerabilities and code smells in your code. It can integrate with your existing workflow to enable continuous code inspection across your project branches and pull requests.

The SonarQube Platform is made of 4 components:

1.One SonarQube Server starting 3 main processes:

•Web Server for developers, managers to browse quality snapshots and configure the SonarQube instance

•Search Server based on Elasticsearch to back searches from the UI

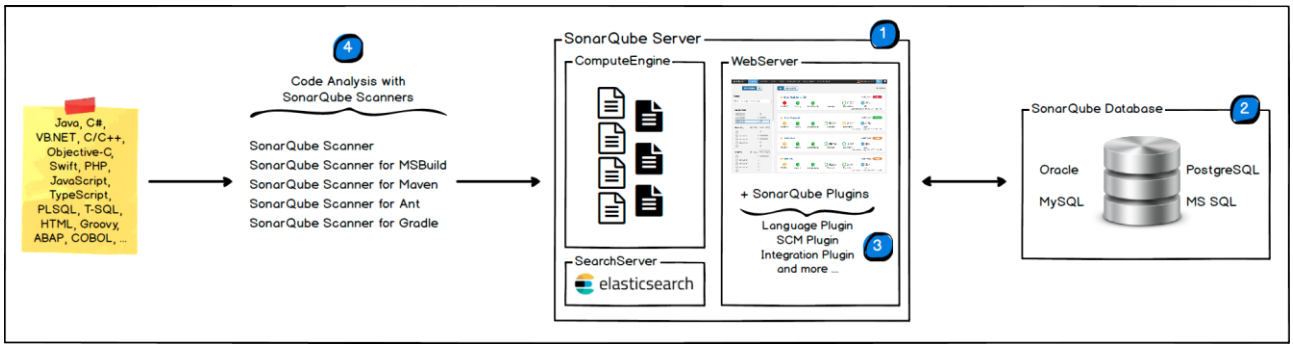
•Compute Engine Server in charge of processing code analysis reports and saving them in the SonarQube Database

2.One SonarQube Database to store:

•the configuration of the SonarQube instance (security, plugins settings, etc.)

•the quality snapshots of projects, views, etc.

3.Multiple SonarQube Plugins installed on the server, possibly including language, SCM, integration, authentication, and governance plugins



4.One or more SonarScanners running on your Build / Continuous Integration Servers to analyze projects

SonarQube SetUp for Crowsoft

Installation

Follow the below commands to install sonarqube on the given OS (ubuntu 16.04).

Step 1: Perform a system update

Before installing any packages on the Ubuntu server instance, it is recommended to update the system. Log in using the sudo user and run the following commands to update the system.

```
sudo apt-get update

sudo apt-get -y upgrade
```

Step 2: Install JDK

Add the Oracle Java repository on the server by running.

```
sudo add-apt-repository ppa:webupd8team/java
```

```
sudo apt-get update
```

Install Oracle JDK by typing:

```
sudo apt install oracle-java8-installer
```

You can now check the version of Java by typing:

```
java -version
```

Step 3: Install and configure PostgreSQL

Install the PostgreSQL repository.

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" >> /etc/apt/sources.list.d/pgdg.list'

wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - | sudo apt-key add -
```

Install the PostgreSQL database server by running:

```
sudo apt-get -y install postgresql postgresql-contrib
```

Start PostgreSQL server and enable it to start automatically at boot time by running:

```
sudo systemctl start postgresql

sudo systemctl enable postgresql
```

Change the password for the default PostgreSQL user.

```
sudo passwd postgres
```

Switch to the postgres user.

```
su - postgres
```

Enter password for the user postgres (used password : csoftsonar)

Create a new user by typing:

```
createuser sonar
```

password for sonar used: sonarcsoft

Switch to the PostgreSQL shell.

```
psql
```

Set a password for the newly created user for SonarQube database.

```
ALTER USER sonar WITH ENCRYPTED password 'sonarcsoft';
```

Create a new database for PostgreSQL database by running:

```
CREATE DATABASE sonar OWNER sonar;
```

Exit from the psql shell:

```
\q
```

Switch back to the sudo user by running the exit command.

Step 4: Download and configure SonarQube

Download the SonarQube installer files archive.

```
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.7.zip
```

You can always look for the link to the latest version of the application on the SonarQube download page.

Install unzip by running:

```
apt-get -y install unzip
```

Unzip the archive using the following command.

```
sudo unzip sonarqube-7.7.zip -d /opt
```

Rename the directory:

```
sudo mv /opt/sonarqube-7.7 /opt/sonarqube
```

Open the SonarQube configuration file using your favorite text editor.

```
sudo nano /opt/sonarqube/conf/sonar.properties
```

Find the following lines.

```
#sonar.jdbc.username=
```

```
#sonar.jdbc.password=
```

Uncomment and provide the PostgreSQL username and password of the database that we have created earlier. It should look like:

```
sonar.jdbc.username=sonar

sonar.jdbc.password= sonarcsoft
```

Next, find:

```
#sonar.jdbc.url=jdbc:postgresql://localhost/sonar
```

Uncomment the line, save the file and exit from the editor.

Step 5: Configure Systemd service

SonarQube can be started directly using the startup script provided in the installer package. As a matter of convenience, you should setup a Systemd unit file for SonarQube.

```
nano /etc/systemd/system/sonar.service
```

Populate the file with:

```
[Unit]

Description=SonarQube service

After=syslog.target network.target



[Service]

Type=forking



ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start

ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop



User=sonar
```

```
Group=sonar

Restart=always



[Install]

WantedBy=multi-user.target
```

Start the application by running:

```
sudo systemctl start sonar
```

Enable the SonarQube service to automatically start at boot time.

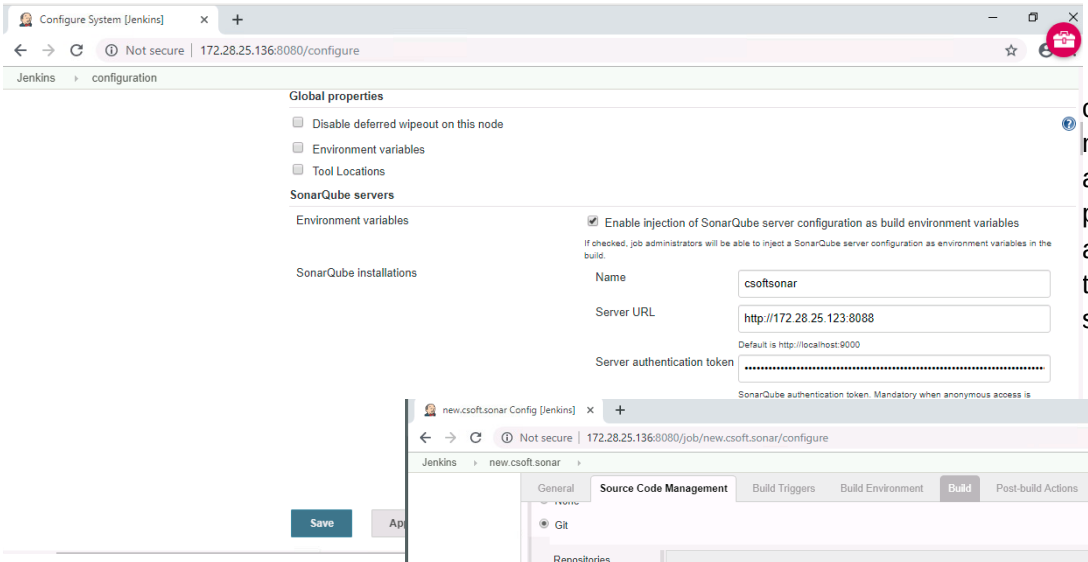
```
sudo systemctl enable sonar
```

To check if the service is running, run:

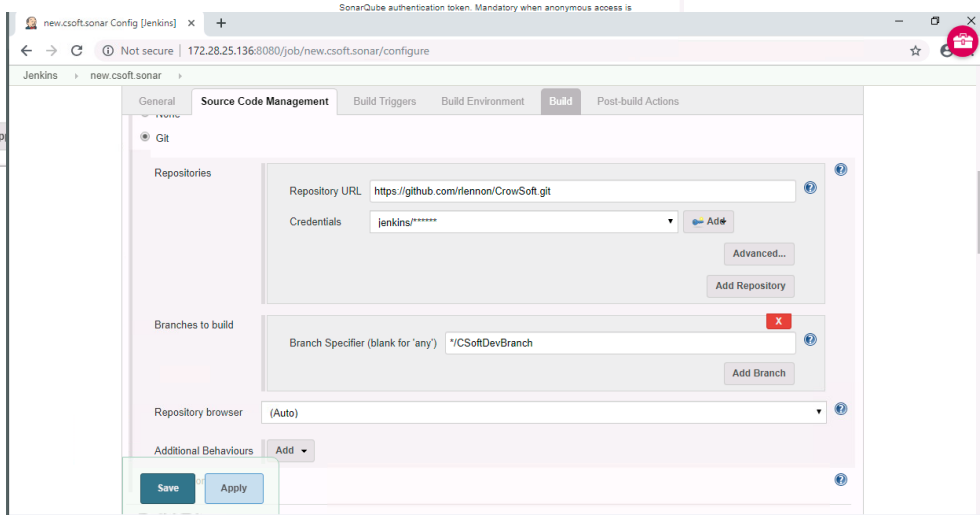
```
sudo systemctl status sonar
```

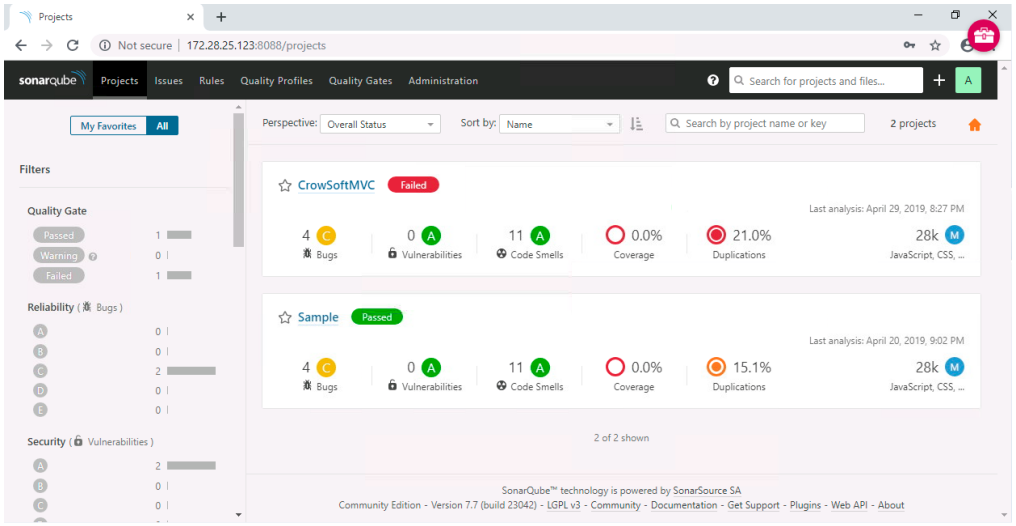
SonarQube with Jenkins Integration

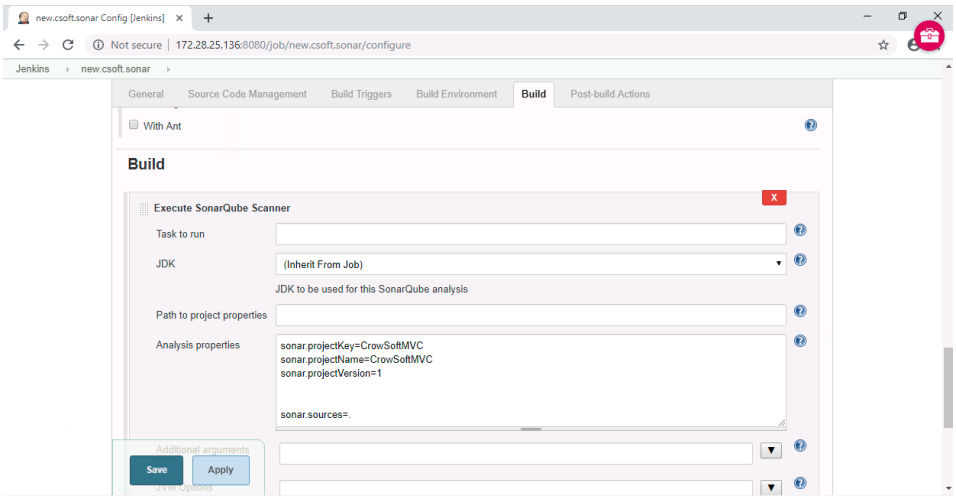
Under Configure System add sonar server details as shown in the screenshot



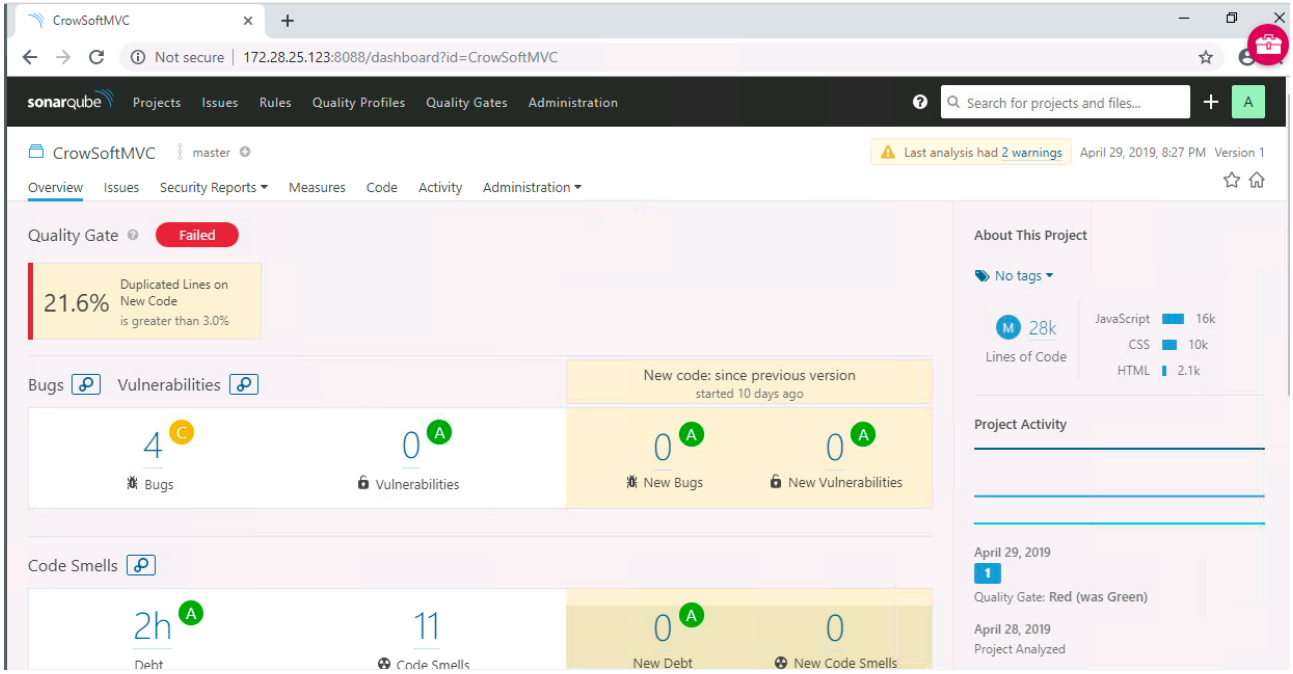
create a new job and add the properties as shown in the screenshots







After doing build now from the job , we can see the analysis results at sonarqube as below


CrowSoftMVC
Not secure | 172.28.25.123:8088/dashboard?id=CrowSoftMVC
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects and files...
CrowSoftMVC
master
Last analysis had 2 warnings
April 29, 2019, 8:27 PM Version 1
Overview
Issues
Security Reports
Measures
Code
Activity
Administration
Quality Gate
Failed
21.6%
Duplicated Lines on New Code
is greater than 3.0%
Bugs
Vulnerabilities
New code: since previous version
started 10 days ago
4
Bugs
0
Vulnerabilities
0
New Bugs
0
New Vulnerabilities
Code Smells
2h
Debt
11
Code Smells
0
New Debt
0
New Code Smells
About This Project
No tags
28k
Lines of Code
JavaScript 16k
CSS 10k
HTML 2.1k
Project Activity
April 29, 2019
1
Quality Gate: Red (was Green)
April 28, 2019
Project Analyzed

---------- End ----------

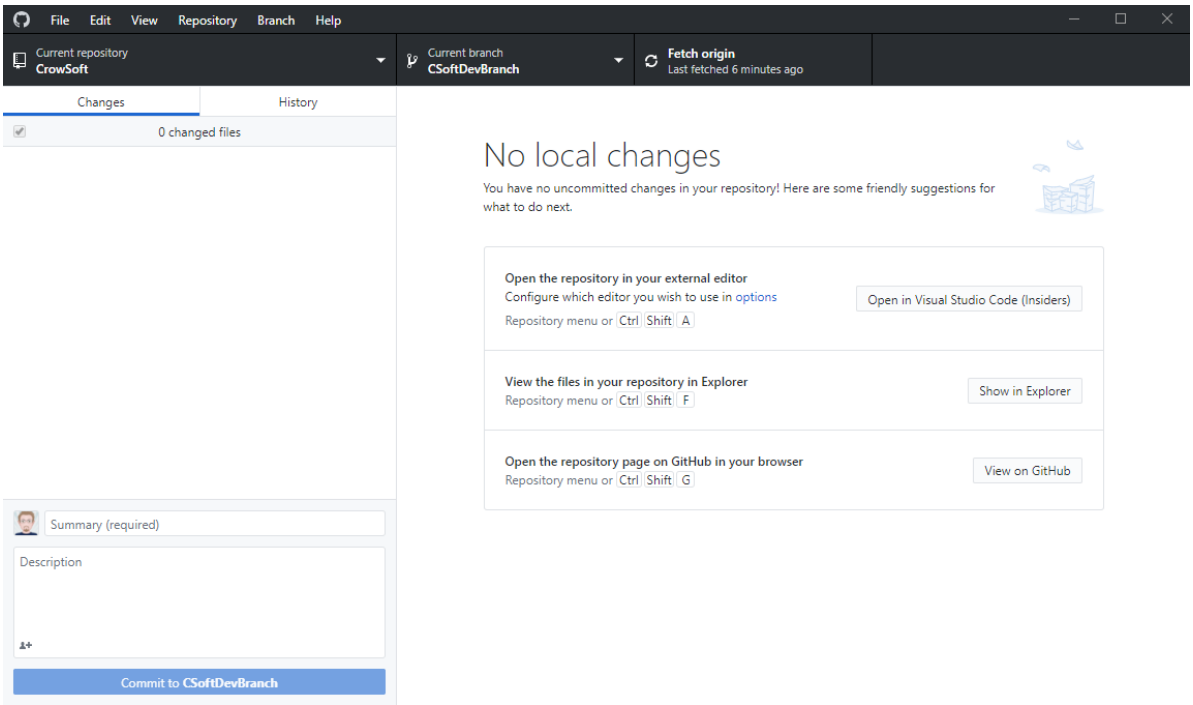
Created by: Charles Aylward
Updated: 03/04/2019

Source Control on Visual Studio Code and Github

I use Github Desktop for windows.

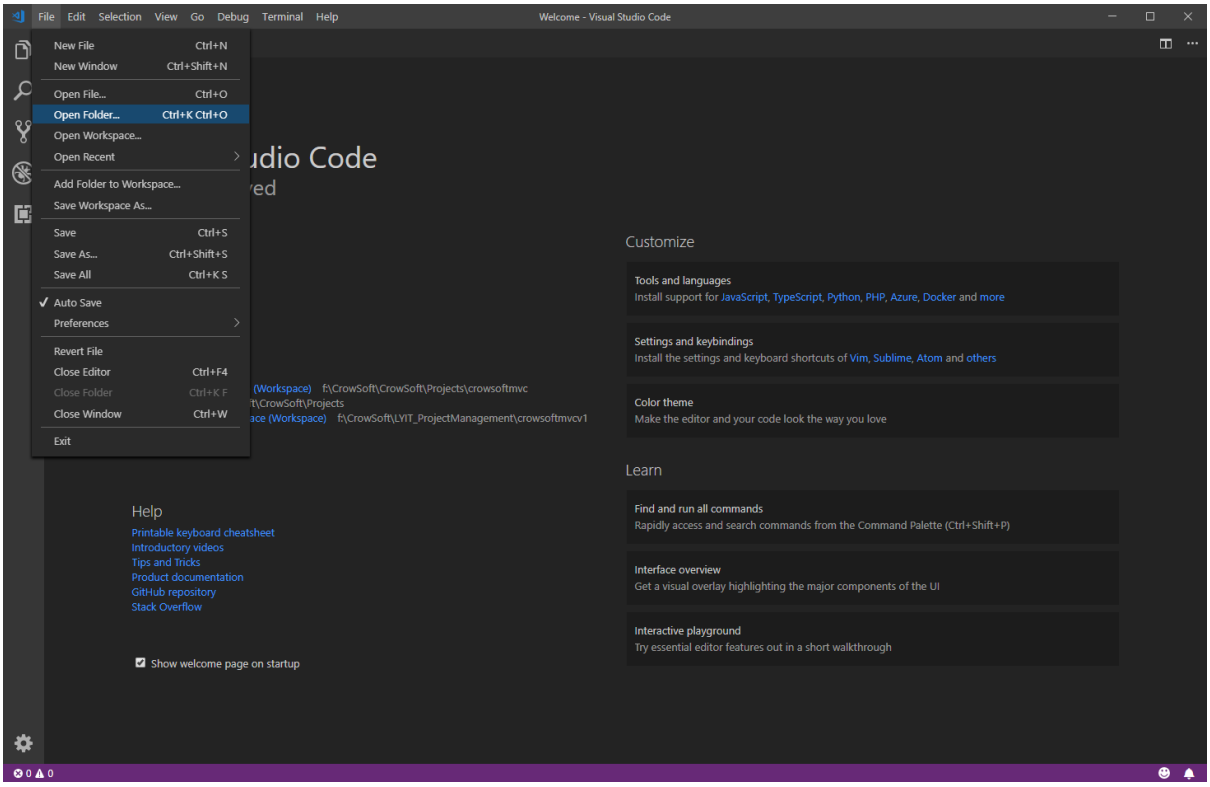
Select your CSoftDevBranch, and then fetch origin.

```
On linux, use: git-pull - Fetch from and integrate with local branch
```

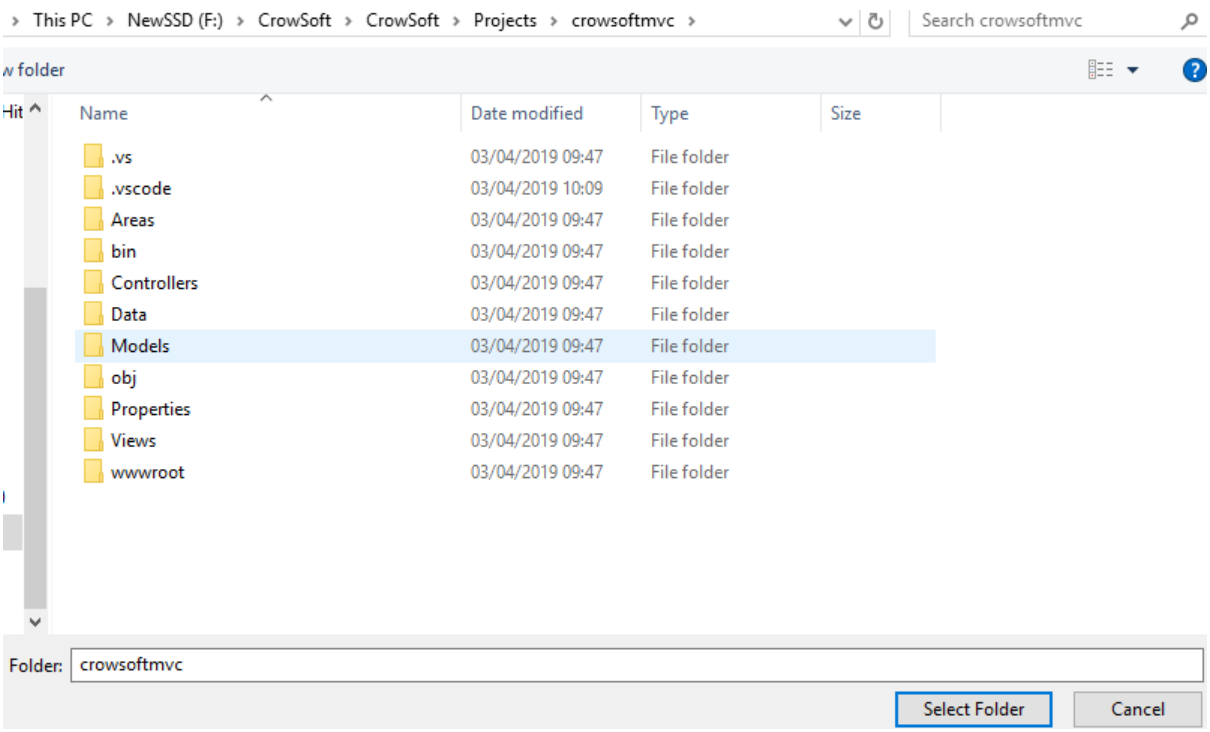


Created by: Charles Aylward
Updated: 03/04/2019

Open Visual Studio Code. Go to File, Open folder.

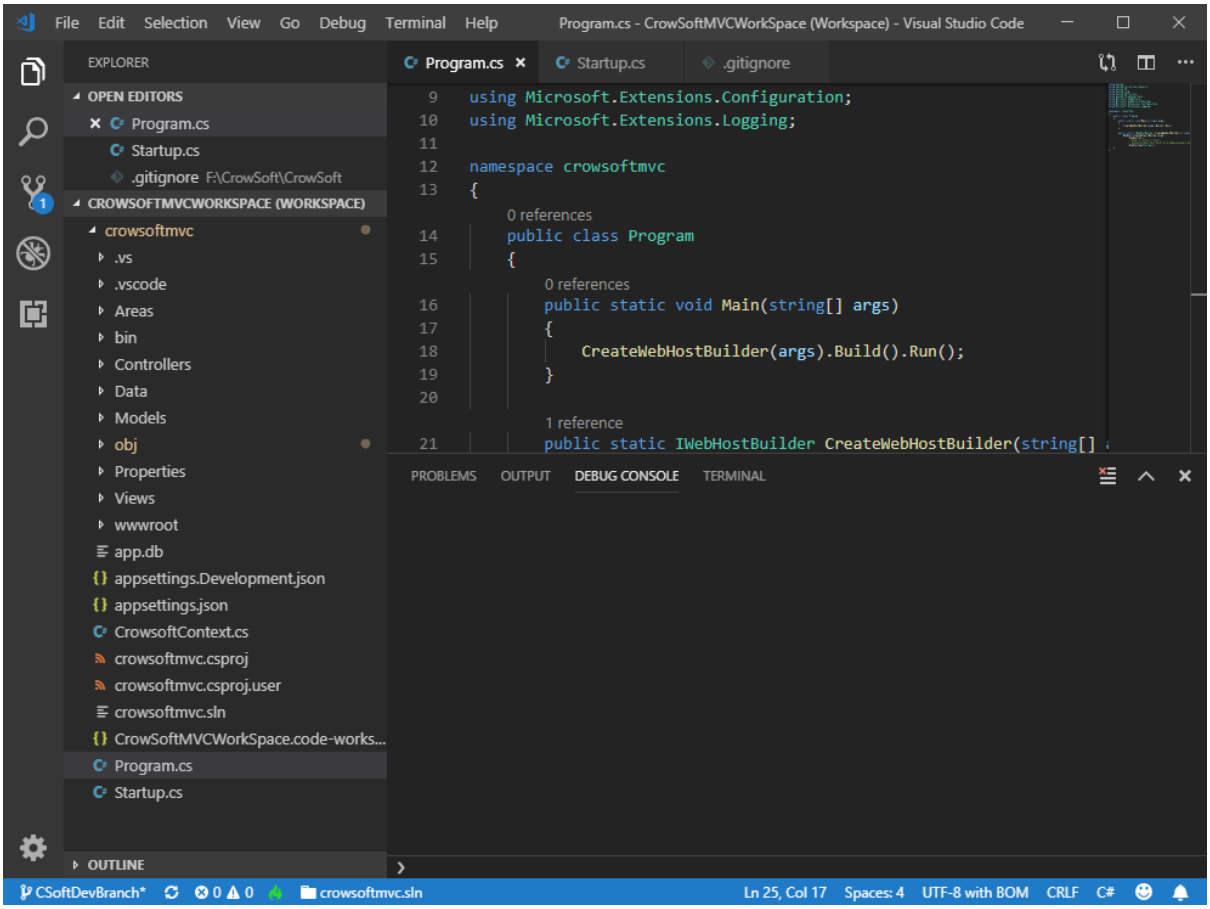


Select the folder on your local drive. **NOTE:** If you see the **CrowSoftMVCWorkSpace.code-workspace** file, then go back, and open the workspace file.

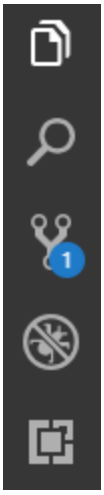


Created by: Charles Aylward
Updated: 03/04/2019

Note  on bottom left of the IDE. This will automatically link to your Github branch if you select the local git folder.



Below is where you can find Source Control.



File Explorer

Search

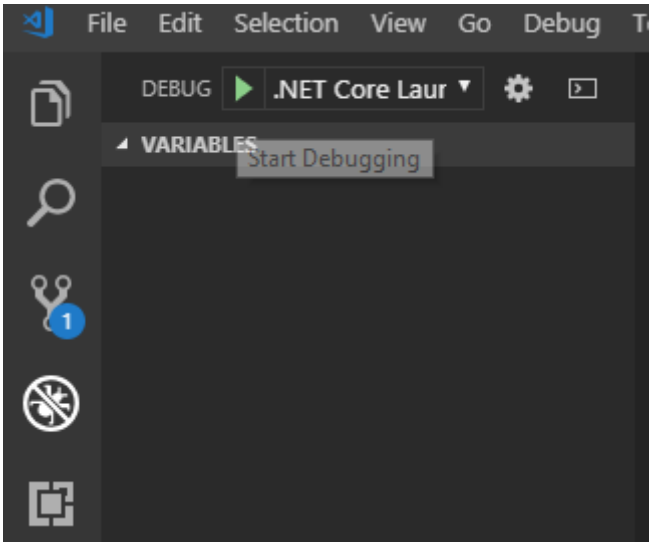
Source Control

Debug

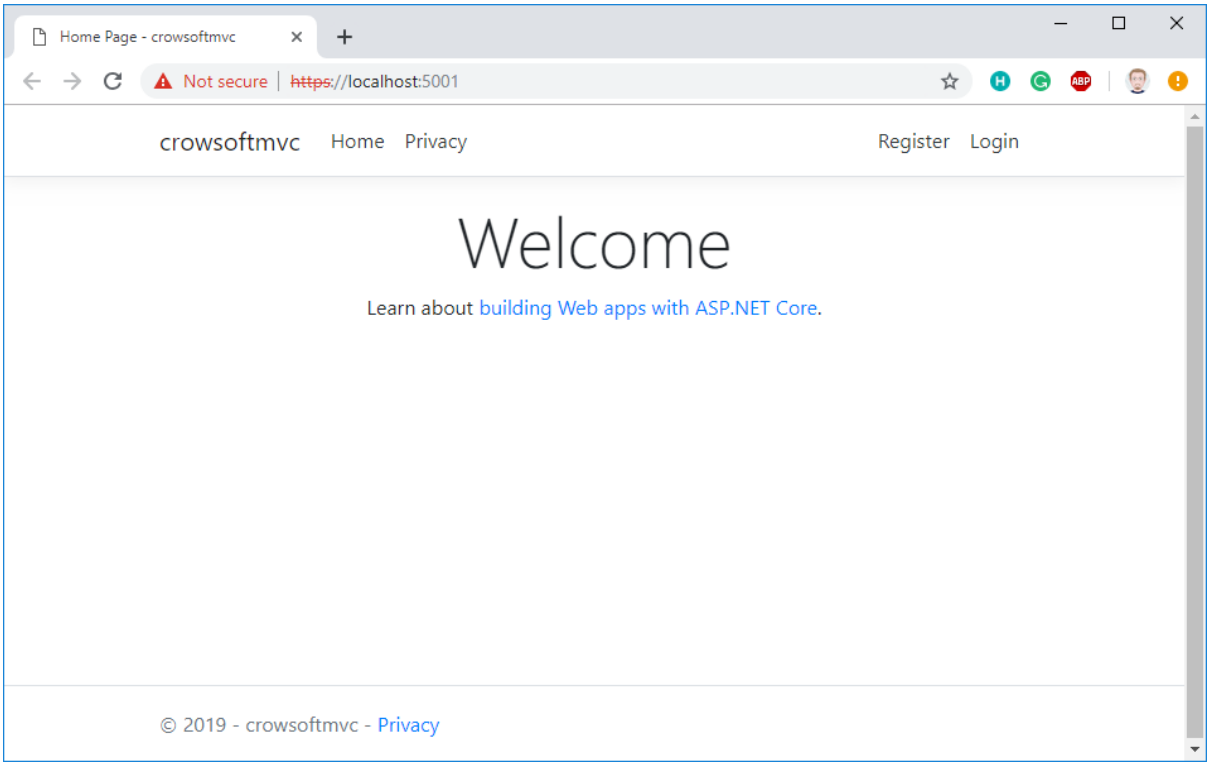
Extensions (To add new extensions)

Created by: Charles Aylward
Updated: 03/04/2019

Let's debug and run the app. Click on the Debug Icon, then click 

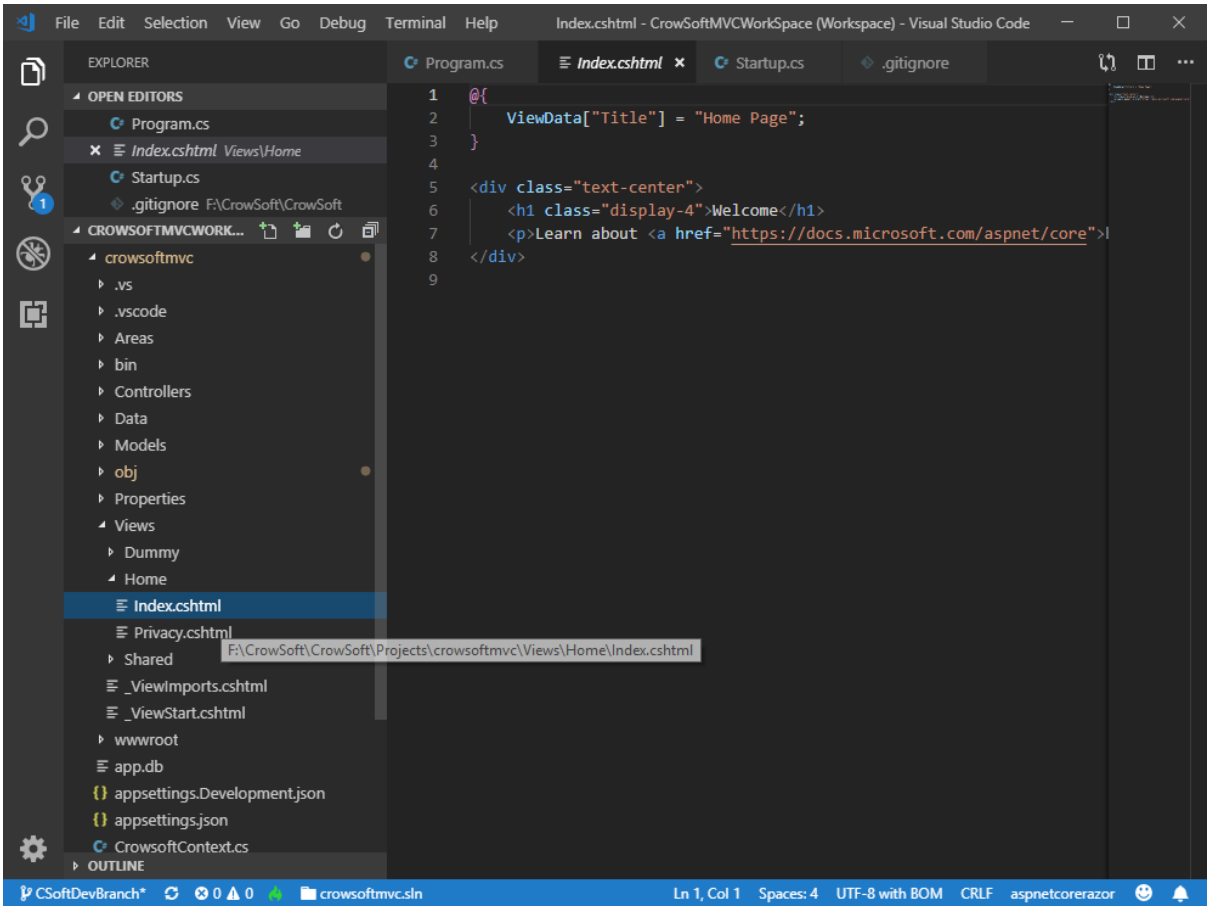


This should open your default browser and run the web application as shown below.

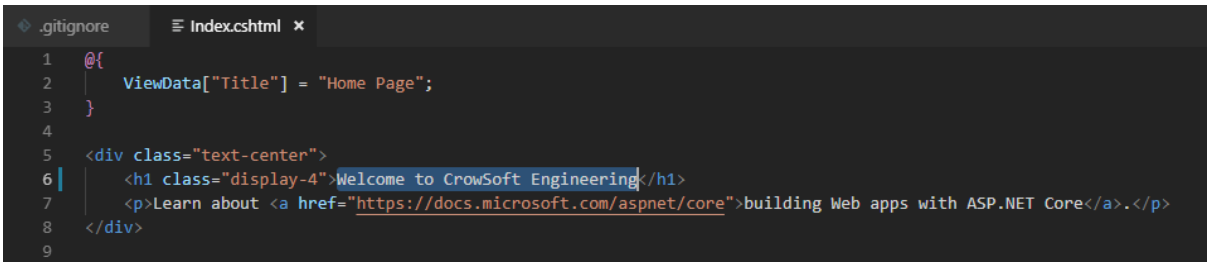


Created by: Charles Aylward
Updated: 03/04/2019

Let's make a change to the code and check it in. Go to Views/Home/Index.cshtml



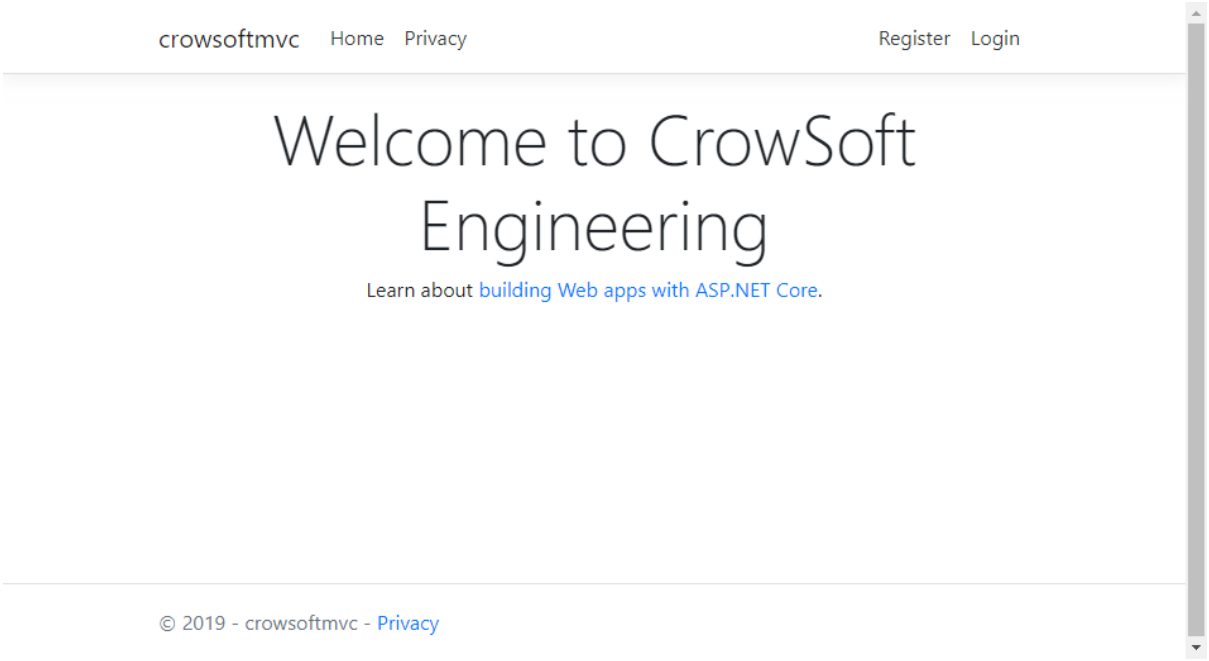
Change the display-4 message to: **Welcome to CrowSoft Engineering**



Click File / Save

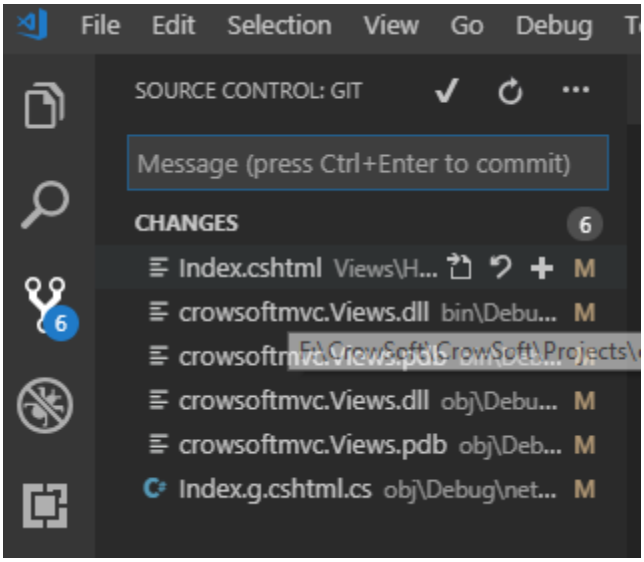
Created by: Charles Aylward
Updated: 03/04/2019

Then Debug and click  again. You can see then change on the web app below.



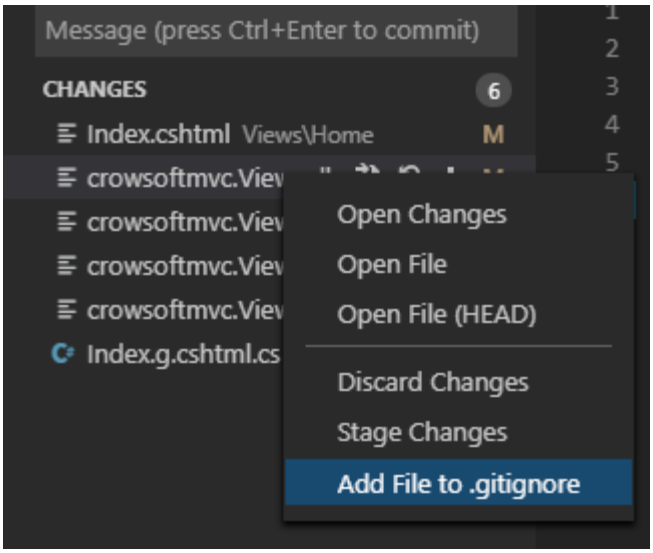
Now let's check in our changes.

Click on Source Control button. You will see a few CHANGES on the list.

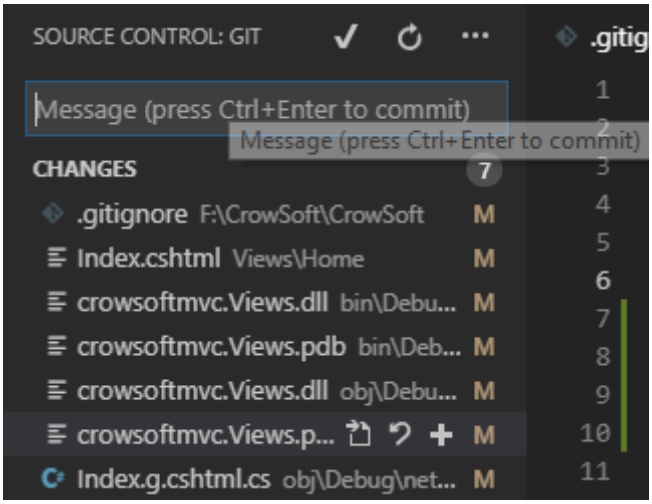


Created by: Charles Aylward
Updated: 03/04/2019

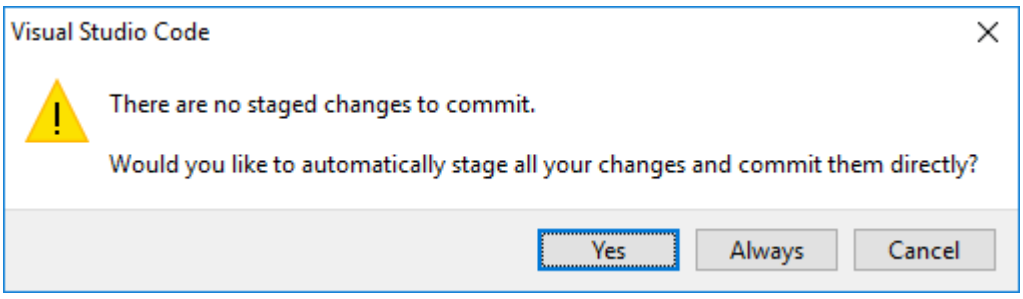
Any dll's and pdb files, must be ignored and not checked in. You can right click on those, and click Add File to .gitignore.



Enter your message below field, and click the commit button or press Ctrl-Enter.

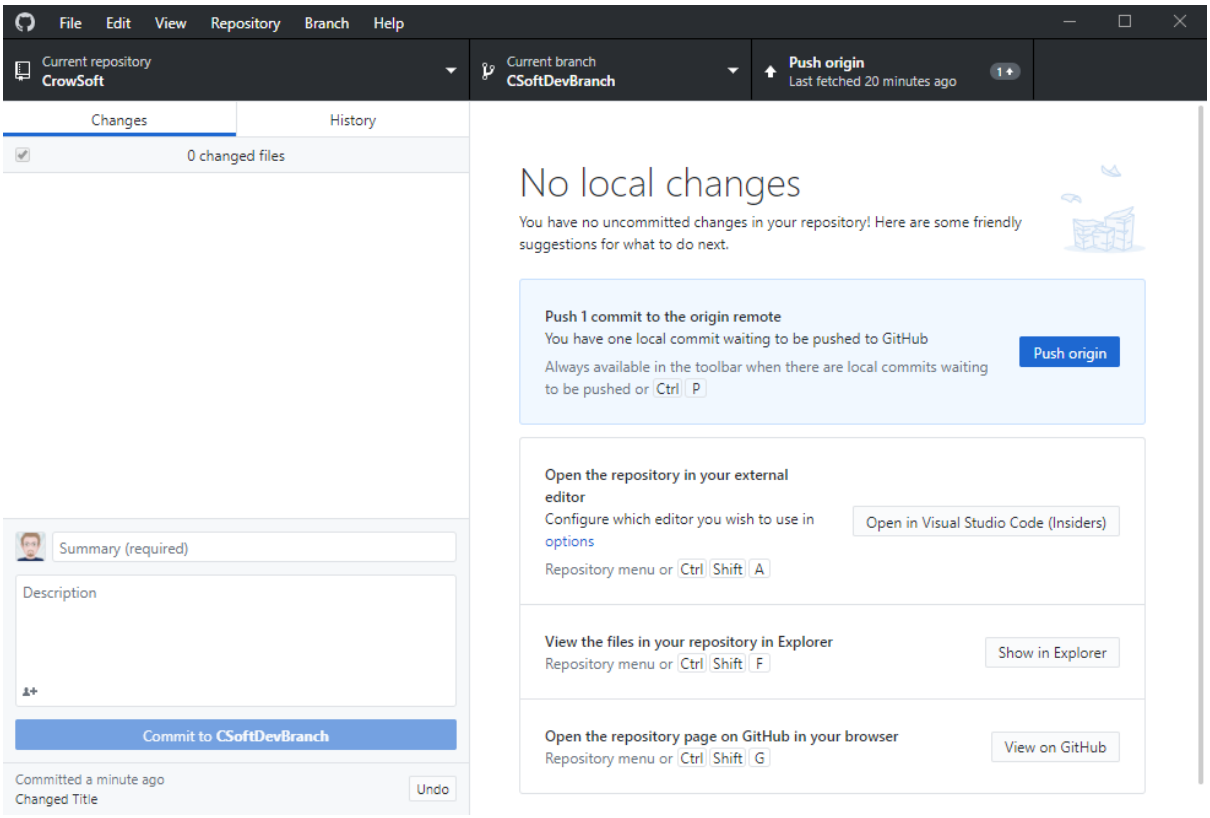


The following screen will appear. Click Yes. **NOTE:** This will only commit code locally. You still need to Push it to Github.

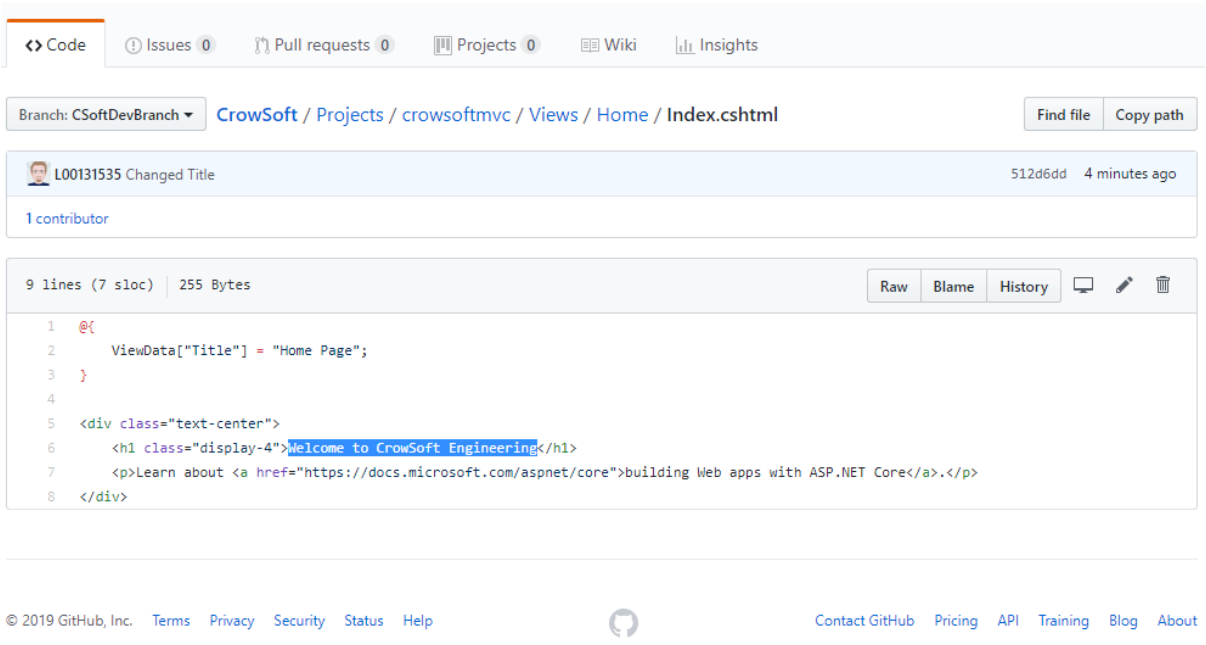


Created by: Charles Aylward
Updated: 03/04/2019

To push changes to Github, open Github Desktop, then Click Push origin or for Linux or command line users: ***git push origin CSoftDevBranch***



Go to Github.com, then open file in this location:
CrowSoft/Projects/crowsoftmvc/Views/Home/Index.cshtml



Make sure the change is showing as seen above.

Bobs your uncle! ☺ Finish!

CrowSoft Technologies

Project Management

Jira:
CrowSoft project has been created
Project tasks to be discussed, estimated and tracked.

Environment and automation tools

These technologies are being proposed based on current course work.
To be discussed and agreed for inclusion on the project
Importance and sequence to be decided.

- VMs – Ubuntu 16.04
- Application Language – C#
- MySQL DB
- Ansible – run ssh connections to run configuration tasks – Python based
- Jenkins – create VM install OS with Jenkins instance
- Github account has been created –
- Artifactory jfrog – repository to store application artifacts executables
  Installing jfrog Jenkins plugin – seting up trial jfrog account
- Testing – Nunit / Selenium
- Application logging -- log4net
- Security implementation for user/information stored

Functional Requirements:

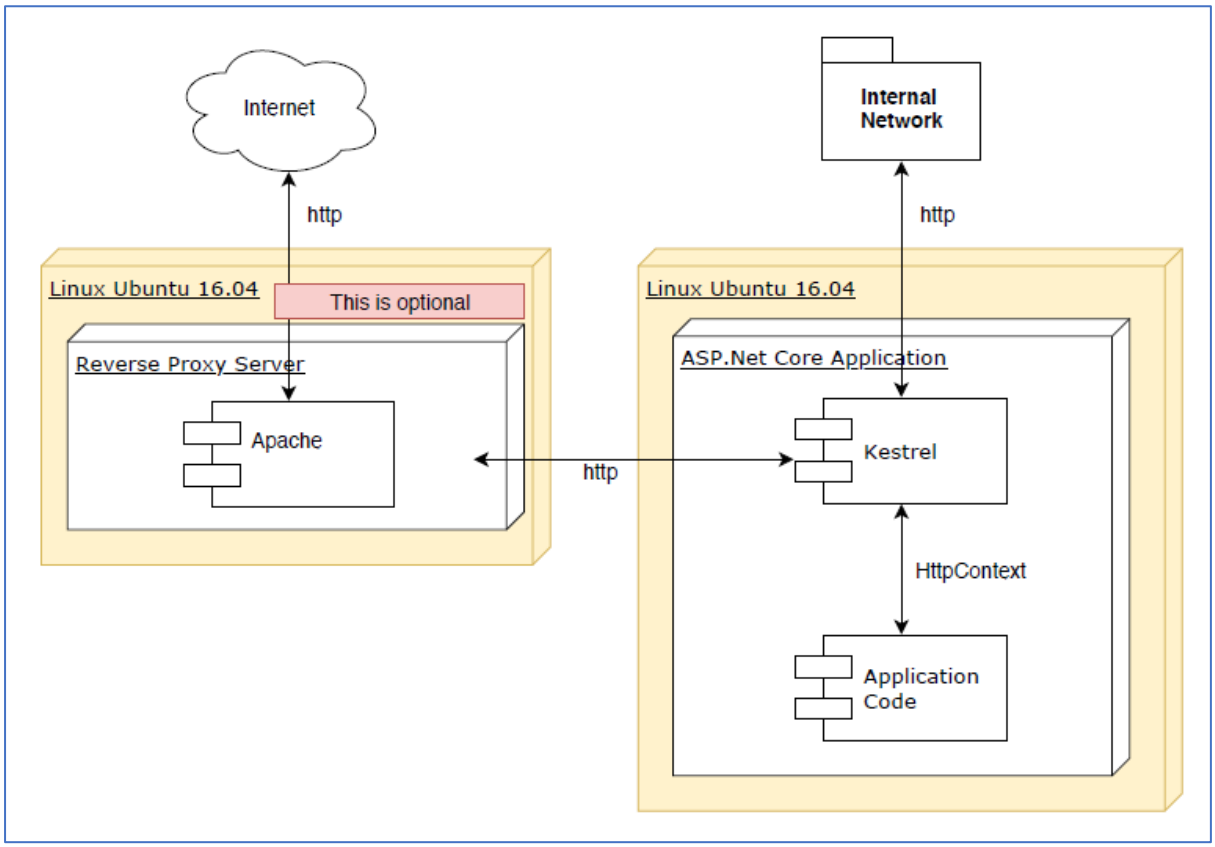
Confirm requirements with product owner

- User logs in
- User adds files
- User adds images
- User queries the system (analysis features)
- Validation on information provided
- Security on information provided

Links:

https://studentjira.lyit.ie/projects/CSOFT/issues/CSOFT-2?filter=allopenissues
https://github.com/rlennon/CrowSoft
https://git-scm.com/docs -- useful reference link
https://www.owasp.org/index.php/OWASP_Proactive_Controls - security info

Technology Stack



Date: 14/04/2019
Created by: Charles Aylward

CroSoft Core Video Links (Standards)

Contents

AspNetUser Security Video

- Description: This video explain how I implement AspNetUser Identity Manager to the CrowsSoft MVC web application.
- Video Name: **AspNetUser_Security.mp4**
- **File**: Download
- **YouTube**: Play Now

Explaining Models in MVC Video

- Description: In this video I explain how the Model needs to be set up in code to be able to automatically generate the Controller and Views.
- Video Name: **ExplainingModelsV1.mp4**
- **File**: Download
- **YouTube**: Play Now

MVC Auto generate classes Video

- Description: In this video I explain how to auto generate the Controller and views the Model needs to be set up in code to be able to automatically generate the Controller and Views.
- Video Name: **MVC_Autogenerate.mp4**
- **File**: Download
- **YouTube**: Play Now

Date: 14/04/2019
Created by: Charles Aylward

MVC Auto generate classes Video

- Description: In this video I explain how I linked the UserAccount table to the AspNetUser and what code I needed to change to make this link work.
- Video Name: **LinkUserAccountToAspNetUsers.mp4**
- **File**: Download
- **YouTube**: Play Now

NUnit Test the Controller

- Description: In this video I explain to create Unit tests to test actions within a Controller using NUnit. I also created Helpers classes to set up mock data for testing.
- Video Name: **NUnit_Controllers.mp4**
- **File**: Download
- **YouTube**: Play Now

dotNet Install

- Description: In this video I explain to install dotNet on Linux
- Video Name: **NUnit_Controllers.mp4**
- **File**: Download
- **YouTube**: Play Now

Install Visual Studio on Linux

- Description: In this video I explain how to install Visual Studio on Linux
- Video Name: **NUnit_Controllers.mp4**
- **File**: Download
- **YouTube**: Play Now

MySQL & dotNet

- Description: In this video I explain MySQL and dotNet
- Video Name: **MySQLAndDotnet.mp4**
- **File**: Download
- **YouTube**: Play Now

Date Created: 20/03/2019
Created By: Charles Aylward

CrowSoft WebApp Copy and Publish to Dev Server

Contents

Date Created: 20/03/2019
Created By: Charles Aylward

The web application service

A service is required to be able to run the application without having to manually start the application in the console.

Service Configuration

The service name is: **crowsoft-webapp.service**

The config file of the service: located at /etc/system/system/crowsoft-webapp.service

```
devadmin@csoft-dev-ubuntu-golden: /home
  GNU nano 2.5.3                File: /etc/systemd/system/crowsoft-webapp.service

[Unit]
Description=Crowsoft .NET MVC Web API App running on Ubuntu

[Service]
WorkingDirectory=/var/www/crowsoftwww
ExecStart=/usr/bin/dotnet /var/www/crowsoftwww/crowsoftmvc.dll
Restart=always
# Restart service after 10 seconds if the dotnet service crashes:
RestartSec=10
KillSignal=SIGINT
SyslogIdentifier=dotnet-example
User=www-data
Environment=ASPNETCORE_ENVIRONMENT=Development

[Install]
WantedBy=multi-user.target
```

How to Check the Service Status

To check the status of the service: *sudo systemctl status crowsoft-webapp.service*
NB: Make sure the status are: **active (running)**

```
devadmin@csoft-dev-ubuntu-golden:/home$ sudo systemctl status crowsoft-webapp.service
● crowsoft-webapp.service - Crowsoft .NET MVC Web API App running on Ubuntu
   Loaded: loaded (/etc/systemd/system/crowsoft-webapp.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2019-03-26 17:37:04 GMT; 1 weeks 3 days ago
 Main PID: 864 (dotnet)
   CGroup: /system.slice/crowsoft-webapp.service
           └─864 /usr/bin/dotnet /var/www/crowsoftwww/crowsoftmvc.dll

Mar 26 17:37:12 csoft-dev-ubuntu-golden dotnet-example[864]: warn: Microsoft.AspNetCore.DataProtection.KeyManagemen
Mar 26 17:37:12 csoft-dev-ubuntu-golden dotnet-example[864]:       Neither user profile nor HKLM registry available
Mar 26 17:37:12 csoft-dev-ubuntu-golden dotnet-example[864]: info: Microsoft.AspNetCore.DataProtection.KeyManagemen
Mar 26 17:37:12 csoft-dev-ubuntu-golden dotnet-example[864]:       Creating key {9eb7f333-1a9d-425f-bf3b-a5c204c921
Mar 26 17:37:12 csoft-dev-ubuntu-golden dotnet-example[864]: warn: Microsoft.AspNetCore.DataProtection.KeyManagemen
Mar 26 17:37:12 csoft-dev-ubuntu-golden dotnet-example[864]:       No XML encryptor configured. Key {9eb7f333-1a9d-
Mar 26 17:37:14 csoft-dev-ubuntu-golden dotnet-example[864]: Hosting environment: Development
Mar 26 17:37:14 csoft-dev-ubuntu-golden dotnet-example[864]: Content root path: /var/www/crowsoftwww
Mar 26 17:37:14 csoft-dev-ubuntu-golden dotnet-example[864]: Now listening on: http://[::]:5123
Mar 26 17:37:14 csoft-dev-ubuntu-golden dotnet-example[864]: Application started. Press Ctrl+C to shut down.
lines 1-17/17 (END)
```

How to stop the service

To stop the service: *sudo systemctl stop crowsoft-webapp.service*

```
devadmin@csoft-dev-ubuntu-golden:/home$ sudo systemctl stop crowsoft-webapp.service
```

How to start the service

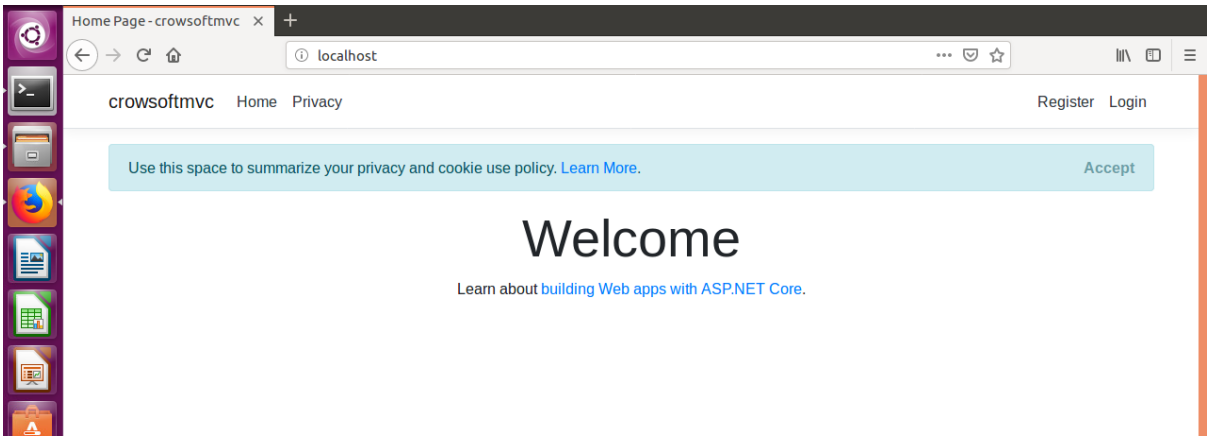
To start the service: *sudo systemctl start crowsoft-webapp.service*

```
devadmin@csoft-dev-ubuntu-golden:/home$ sudo systemctl start crowsoft-webapp.service
```

Date Created: 20/03/2019
Created By: Charles Aylward

Check if the web application is working

After the service is running, test the web app, open FireFox, and go enter localhost



Copy Web App Project Files to Dev Server

On the Dev Server, the folder for the project files should copy into the following folder:

```
devadmin@csoft-dev-ubuntu-golden:~/dev/crowsoftmvcv1$ ls
app.db                      Controllers             Data          Properties
appsettings.Development.json  CrowsoftContext.cs      Models        Startup.cs
appsettings.json            crowsoftmvc.csproj      obj           Views
Areas                       crowsoftmvc.csproj.user  Program.cs    wwwroot
bin                         crowsoftmvc.sln         Program.cs.save
devadmin@csoft-dev-ubuntu-golden:~/dev/crowsoftmvcv1$
```

NOTE: The Staging and Production server will not have this folder. We should publish from either Jenkins or Artifactory for the final build.

Date Created: 20/03/2019
Created By: Charles Aylward

To publish the Web Application

Steps to publish new code:

1. Make sure the service is stopped. Command to stop: ***sudo systemctl stop crowsoft-webapp.service***
2. Run the following command Command: ***sudo dotnet publish ~/dev/crowsoftmvcv1/crowsoftmvc.csproj –c release –o /var/www/crowsoftwww***

```
devadmin@csoft-dev-ubuntu-golden:~$ sudo dotnet publish ~/dev/crowsoftmvcv1/crowsoftmvc.csproj -c release -o /var/www/crowsoftwww
Microsoft (R) Build Engine version 15.9.20+g88f5fadfbe for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 79.62 ms for /home/devadmin/dev/crowsoftmvcv1/crowsoftmvc.csproj.
Models/Dummy.cs(10,33): warning CS0169: The field 'Dummy.context' is never used [/home/devadmin/dev/crowsoftmvcv1/crowsoftmvc.csproj
]
  crowsoftmvc -> /home/devadmin/dev/crowsoftmvcv1/bin/release/netcoreapp2.2/crowsoftmvc.dll
  crowsoftmvc -> /home/devadmin/dev/crowsoftmvcv1/bin/release/netcoreapp2.2/crowsoftmvc.Views.dll
  crowsoftmvc -> /var/www/crowsoftwww/
devadmin@csoft-dev-ubuntu-golden:~$
```

3. Now start the service. Command to start: ***sudo systemctl start crowsoft-webapp.service***
4. Optional: Check status of service: ***sudo systemctl status crowsoft-webapp.service***
5. This is how the folder content should look like.

```
root@csoft-dev-ubuntu-golden:/var/www/crowsoftwww# ls
app.db
appsettings.Development.json
appsettings.json
crowsoftmvc.deps.json
crowsoftmvc.dll
crowsoftmvc.pdb
crowsoftmvc.runtimeconfig.json
crowsoftmvc.Views.dll
crowsoftmvc.Views.pdb
dotnet-aspnet-codegenerator-design.dll
Google.Protobuf.dll
Microsoft.CodeAnalysis.CSharp.Workspaces.dll
Microsoft.CodeAnalysis.Workspaces.dll
Microsoft.Data.Sqlite.dll
Microsoft.EntityFrameworkCore.Abstractions.dll
Microsoft.EntityFrameworkCore.dll
Microsoft.EntityFrameworkCore.Relational.dll
Microsoft.EntityFrameworkCore.Sqlite.dll
Microsoft.VisualStudio.Web.CodeGeneration.Contracts.dll
Microsoft.VisualStudio.Web.CodeGeneration.Core.dll
Microsoft.VisualStudio.Web.CodeGeneration.dll
Microsoft.VisualStudio.Web.CodeGeneration.EntityFrameworkCore.dll
Microsoft.VisualStudio.Web.CodeGeneration.Templating.dll
Microsoft.VisualStudio.Web.CodeGeneration.Utils.dll
Microsoft.VisualStudio.Web.CodeGenerators.Mvc.dll
MySql.Data.dll
NuGet.Frameworks.dll
Remotion.Linq.dll
runtimes
SQLitePCLRaw.batteries_green.dll
SQLitePCLRaw.batteries_v2.dll
SQLitePCLRaw.core.dll
SQLitePCLRaw.provider.e_sqlite3.dll
System.Composition.AttributedModel.dll
System.Composition.Convention.dll
System.Composition.Hosting.dll
System.Composition.Runtime.dll
System.Composition.TypedParts.dll
```

Install wrk

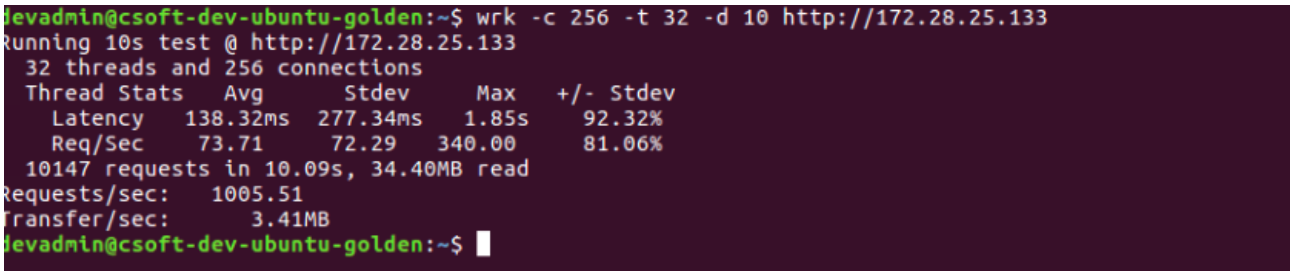
Installing wrk package on Ubuntu

command on terminal:

```
sudo apt-get update

sudo apt-get install wrk
```

To run wrk on the application , issue the below command

$ wrk -c 256 -t 32 -d 10 http://172.28.25.133



Above is the application performance