**[Microsoft Application Lifecycle Management](https://blogs.msdn.microsoft.com/visualstudioalm)**

Learn to use Visual Studio, Visual Studio Team Services, Application Insights and Team Foundation Server to decrease rework, increase transparency into your application and increase the rate at which you can ship high quality software throughout the application lifecycle

[Microsoft Application Lifecycle Management](https://blogs.msdn.microsoft.com/visualstudioalm/)

**Using Selenium with Cloud-based Load Testing**

★★★★★

★★★★

★★★

★★

★

November 16, 2014 by [RanganathG [MSFT]](https://social.msdn.microsoft.com/profile/RanganathG+%5BMSFT%5D) // [21 Comments](https://blogs.msdn.microsoft.com/visualstudioalm/2014/11/16/using-selenium-with-cloud-based-load-testing/#comments)

* [0](https://blogs.msdn.microsoft.com/visualstudioalm/2014/11/16/using-selenium-with-cloud-based-load-testing/)
* [0](https://blogs.msdn.microsoft.com/visualstudioalm/2014/11/16/using-selenium-with-cloud-based-load-testing/)
* [0](https://blogs.msdn.microsoft.com/visualstudioalm/2014/11/16/using-selenium-with-cloud-based-load-testing/)

This blog is an introduction to how users can execute Selenium tests using [Cloud-based Load Testing](https://www.visualstudio.com/features/vso-cloud-load-testing-vs) from Visual Studio Online.

Introduction

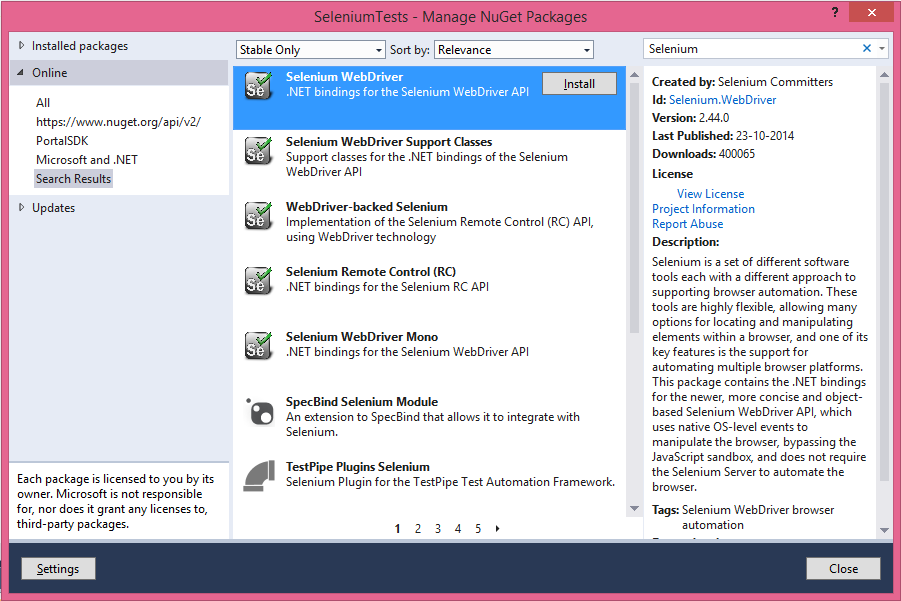
Let’s get familiar with Cloud-based Load Testing (CLT) from Visual Studio Online (VSO) and Selenium.

* Cloud-base Load Testing:
  + A good introduction to CLT with VSO and key feature can be found here – [Introducing Cloud-based Load Testing](http://blogs.msdn.com/b/visualstudioalm/archive/2013/06/03/introducing-cloud-based-load-testing-with-team-foundation-service.aspx)
  + To get started quickly follow the instructions at this link – [Load Testing in cloud](http://www.visualstudio.com/get-started/load-test-your-app-vs)
* Selenium
  + *Selenium automates browsers*Quote: [Selenium Official Website](http://www.seleniumhq.org/)
  + Please visit the official web site to get an overall idea of Selenium
  + For this tutorial we will be specifically using a [Selenium Nuget Package](http://www.nuget.org/packages/selenium.webdriver) and [Phantomjs Nuget Package](http://www.nuget.org/packages/PhantomJS/) in Visual Studio

Let’s get started.

Authoring Selenium unit tests in Visual Studio

1. Let’s start by creating a Unit Test Project in Visual Studio. File -> New Project -> Templates -> Visual C# -> Test -> Unit Test Project
2. Once the library is created, install Selenium Nuget Package. Right Click on the project -> Manage Nuget Packages… -> “Search for Selenium” -> Install

[](https://msdnshared.blob.core.windows.net/media/MSDNBlogsFS/prod.evol.blogs.msdn.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/45/92/3426.SeleniumNuget.png)

1. After installing the Nuget package, the references would include WebDriver reference.
2. Download required Web drivers. For the sake of this example let’s use PhantomJs. You can search for Phantomjs in the Nuget Package Manager and install it just as you did for Selenium. Change the property of phantomjs.exe “Copy to Output Directory” as “Copy if Newer”

1. You can create a CS file to get started with writing Unit Tests. Let’s add a small unit test using Selenium

using Microsoft.VisualStudio.TestTools.UnitTesting;  
 using OpenQA.Selenium;  
 using OpenQA.Selenium.PhantomJS;  
 using System;  
 using System.Text;  
   
 namespace SeleniumSample  
 {  
 [TestClass]  
 public class SeleniumTests  
 {  
 [TestMethod]  
 public void TheBingSearchTest()  
 {  
 TestContext.BeginTimer("BingSearchTest\_Navigate");  
 \_driver.Navigate().GoToUrl("http://www.bing.com/");  
 TestContext.EndTimer("BingSearchTest\_Navigate");  
   
 TestContext.BeginTimer("BingSearchTest\_SearchBHarry");  
 \_driver.FindElement(By.Id("sb\_form\_q")).SendKeys("Brian harry blog");  
 \_driver.FindElement(By.Id("sb\_form\_go")).Click();  
 TestContext.EndTimer("BingSearchTest\_SearchBHarry");  
   
 var elementText = \_driver.FindElement(By.XPath("//ol[@id='b\_results']/li/h2/a"));  
 Assert.IsTrue(elementText.Text.Equals("Brian Harry's blog - Site Home - MSDN Blogs"), "Verified title of the blog page");  
 }  
   
 public TestContext TestContext { get; set; }  
   
 #region Additional test attributes  
   
 [TestInitialize]  
 public void SetupTestSuite()  
 {  
 Console.WriteLine("Test init called: {0}");  
 \_driver = new PhantomJSDriver();  
 }  
   
 [TestCleanup]  
 public void CleanupTestSuite()  
 {  
 \_driver.Quit();  
 }  
 #endregion  
 private IWebDriver \_driver;  
 }  
 }

1. Right click on the solution and add a “Test Settings” File. This is important\* in order to run the above code using Test Explorer (Test -> Windows -> Test Explorer)

1. Configure TestSettings to include phantomjs.exe as a deployment item

1. Build the solution.
2. From Test -> Test Settings -> Select Test Settings File, choose the testsettings file that is created in the previous step.
3. Verify this runs on your local machine using Test Explorer (Test> Windows> Test Explorer) to check if things are working fine.

Adding Load Test for the unit tests

1. Add a loadtest to the project (Right click -> Add ->  Load Test …). In the load test configuration wizard, in the “Text Mix”, click add and choose unit test of your interest, for example: TheBingSearchTest
2. Verify the load runs on your local machine.

Running Selenium Tests on CLT On VSO

1. Congratulations you are just one click to run your tests on Microsoft Azure cloud!
2. From Test -> Test Settings -> Select Test Settings File, choose the Local.testsettings file that is created when the load test project is added.
3. Open this file change the option to “Run Tests using Visual Studio Online”

              For more information please visit this blog to get a good understanding of how you can run your tests on Cloud –<http://www.visualstudio.com/get-started/load-test-your-app-vs>

1. Once the run is completed and you have downloaded the report. Click on View report as mentioned in the above link.
2. You would be able get a lot of insights into how the test execution happened. You can get a view of the transactions that happened in your tests (BingSearchTest\_Navigate is a transaction in your test). It would appear like