Automating Selenium Tests in Azure Pipelines

Overview

Selenium is a portable open source software-testing framework for web applications. It has the capability to operate on almost every operating system. It supports all modern browsers and multiple languages including .NET (C#), Java.

In this lab, you will learn how to execute selenium test cases on a C# web application, as part of the Azure DevOps Release pipeline.

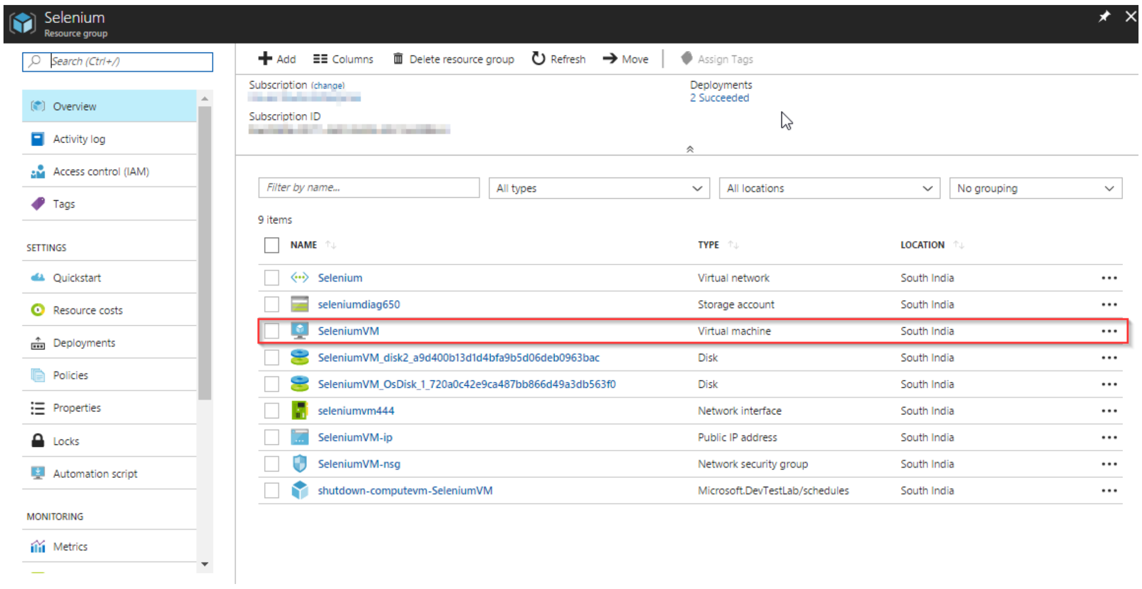
Before you begin

1. Refer to the Getting Started before you follow the below exercises.
2. Click on the deploy to Azure button below, to provision a Windows Server 2016 virtual machine along with SQL Express 2017 and browsers - Chrome and FireFox.

   Deploy to Azure

   https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FMicrosoft%2Falmvm%2Fmaster%2Flabs%2Fvstsextend%2Fselenium%2Farmtemplate%2Fazuredeploy.json

   It should take approximately 20-25 minutes to provision the resources. Once the deployment is successful, you will see the resources as shown.



Subscription (change)
[redacted]
Subscription ID
[redacted]
Deployments
2 Succeeded

9 items

NAME	TYPE	LOCATION
Selenium	Virtual network	South India
seleniumdiag650	Storage account	South India
SeleniumVM	Virtual machine	South India
SeleniumVM_disk2_a9d400b13d1d4bfa9b5d06deb0963bac	Disk	South India
SeleniumVM_OsDisk_1_720a0c42e9ca487bb866d49a3db563f0	Disk	South India
seleniumvm444	Network interface	South India
SeleniumVM-ip	Public IP address	South India
SeleniumVM-nsg	Network security group	South India
shutdown-computevm-SeleniumVM	Microsoft.DevTestLab/schedules	South India

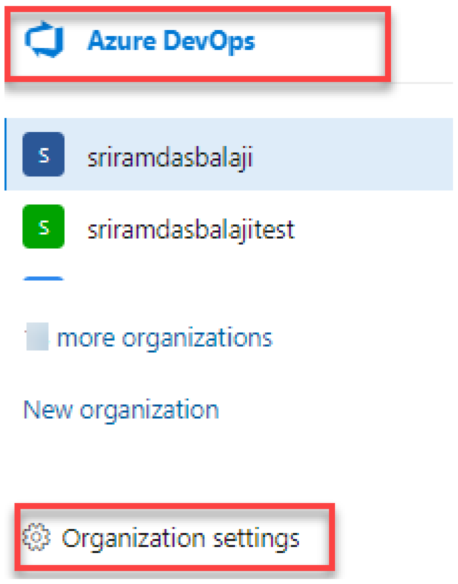
3. Use the Azure DevOps Demo Generator to provision the project on your Azure DevOps Organization. This URL will automatically select Selenium template in the demo generator. If you want to try other projects, use this URL instead - https://azuredevopsdemogenerator.azurewebsites.net/

Follow the simple walkthrough to know how to use the Azure DevOps Demo Generator.

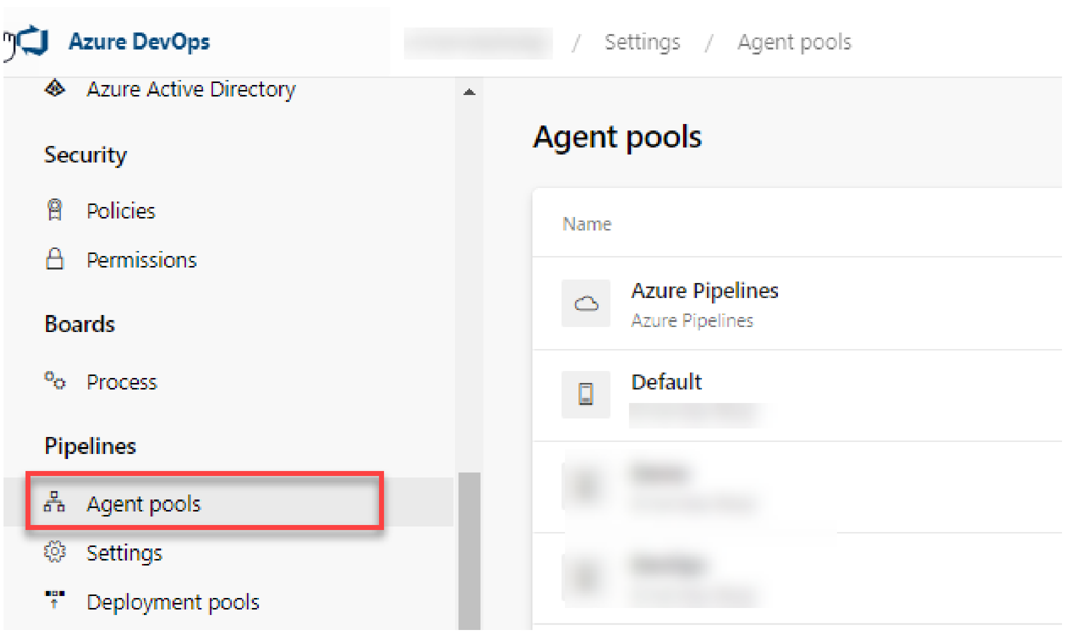
Exercise 1: Configure agent on the VM

Let us configure a ***private*** self-hosted agent on this VM. Selenium requires the agent to be run in **interactive** mode to execute the UI tests.

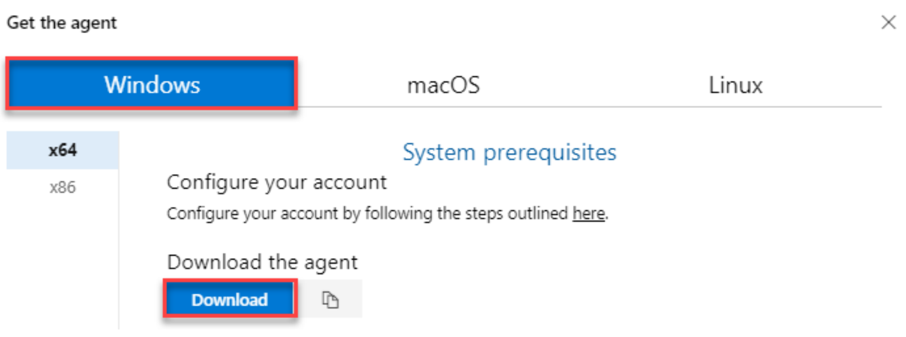
1. Login to the VM using RDP with the following credentials
   - **Username**: vmadmin
   - **Password**: P2ssw0rd@123
2. In the VM open web browser, sign in to your Azure DevOps organization and navigate to the Agent pools tab:
   - Choose **Azure DevOps**, **Organization settings**.



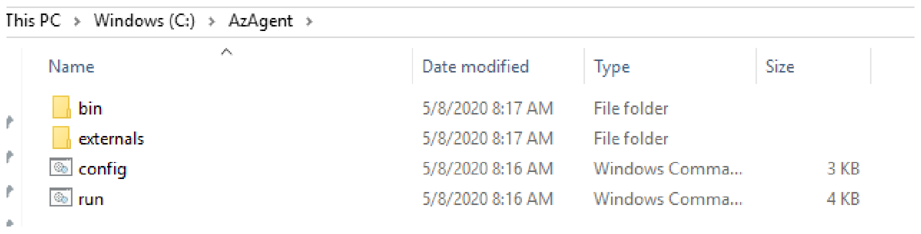
   - Choose **Agent pools**.



- Select the **Default** pool, select the **Agents** tab and choose **New agent**.
- On **Get the agent** dialog box, choose **Windows** and **Download** agent.



3. Make a directory in **C Drive** with the name **AzAgent** and extract the downloaded agent zip file to this directory.



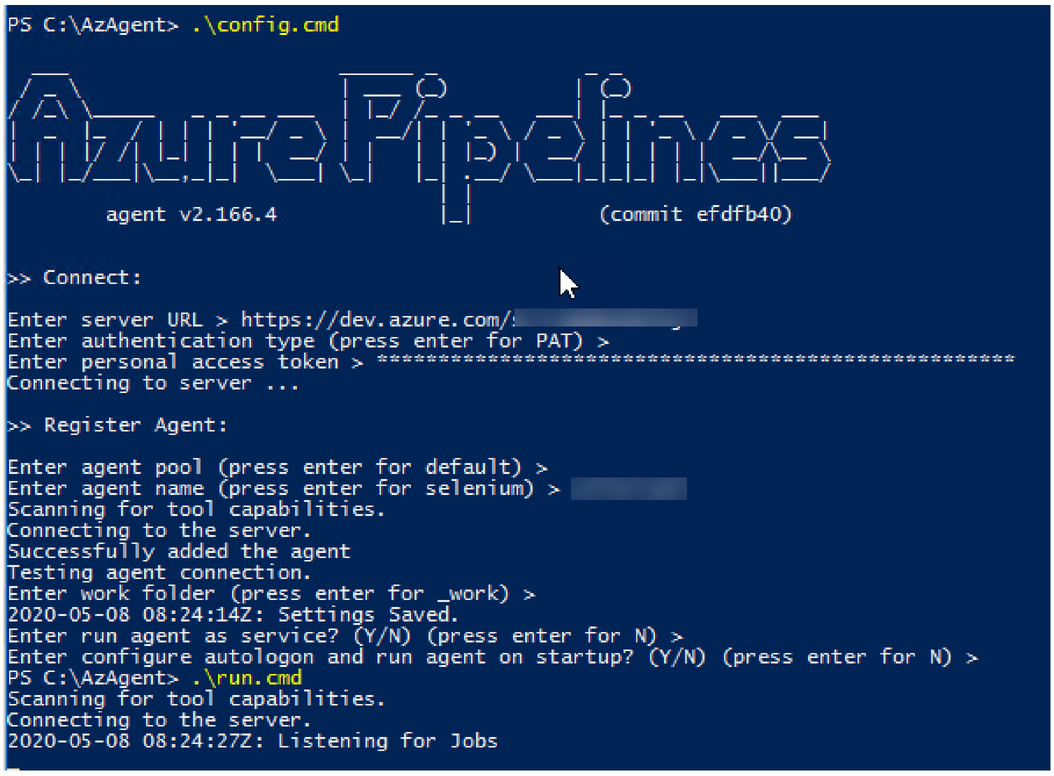
Name	Date modified	Type	Size
bin	5/8/2020 8:17 AM	File folder	
externals	5/8/2020 8:17 AM	File folder	
config	5/8/2020 8:16 AM	Windows Comma...	3 KB
run	5/8/2020 8:16 AM	Windows Comma...	4 KB

4. Open Powershell in **administrator mode**. Change the path to **C:\AzAgent** and type **Config.cmd** and hit **Enter**.
5. Provide the following details:

   - Enter server URL: Your Azure DevOps Organization URL
   - Authentication type: Press the **enter key** for **PAT** as the authentication type and paste the PAT in the next prompt.

- Let us use the default options for the rest of the configuration.
  Press **Enter** for all prompts until the command execution completes.
- Once the agent is registered, type **run.cmd** and hit **Enter** to start the agent.

Click here for more information on how to configure the agent.

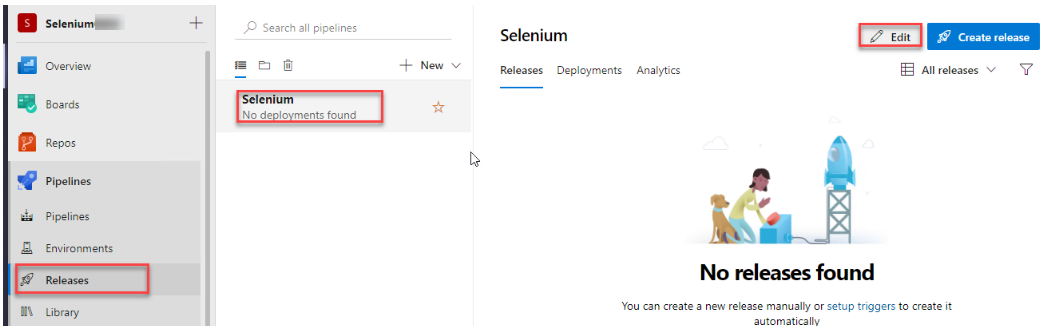


```
PS C:\AzAgent> .\config.cmd

  ___                      ______ _            _ _
 / _ \                     | ___ (_)          | (_)
/ /_\ \_____   _ _ __ ___  | |_/ /_ _ __   ___| |_ _ __   ___  ___
|  _  |_  / | | | '__/ _ \ |  __/| | '_ \ / _ \ | | '_ \ / _ \/ __|
| | | |/ /| |_| | | |  __/ | |   | | |_) |  __/ | | | | |  __/\__ \
\_| |_/___|\__,_|_|  \___| \_|   |_| .__/ \___|_|_|_| |_|\___||___/
                                   | |
        agent v2.166.4             |_|          (commit efdfb40)


>> Connect:

Enter server URL > https://dev.azure.com/[redacted]
Enter authentication type (press enter for PAT) >
Enter personal access token > ****************************************************
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) >
Enter agent name (press enter for selenium) > [redacted]
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2020-05-08 08:24:14Z: Settings Saved.
Enter run agent as service? (Y/N) (press enter for N) >
Enter configure autologon and run agent on startup? (Y/N) (press enter for N) >
PS C:\AzAgent> .\run.cmd
Scanning for tool capabilities.
Connecting to the server.
2020-05-08 08:24:27Z: Listening for Jobs
```

Exercise 2: Configure Release Pipeline

The target machine is having agent configured to deploy the applications and run selenium testcases. The release definition uses **Phases** to deploy to target servers.

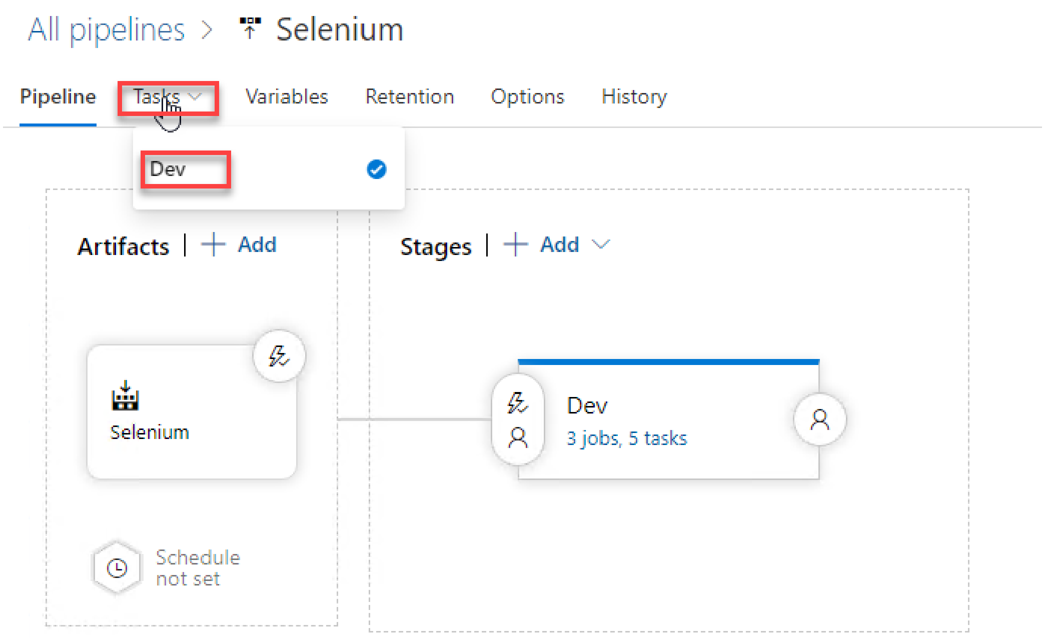
1. Go to **Releases** under **Pipelines** tab. Select **Selenium** release definition and click on **Edit**.

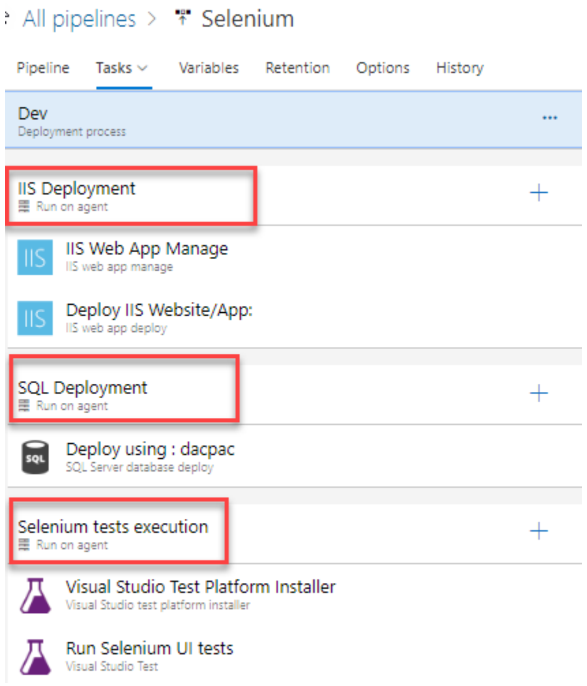


No releases found

You can create a new release manually or setup triggers to create it automatically

2. Open **Dev** environment to see the three deployment phases.



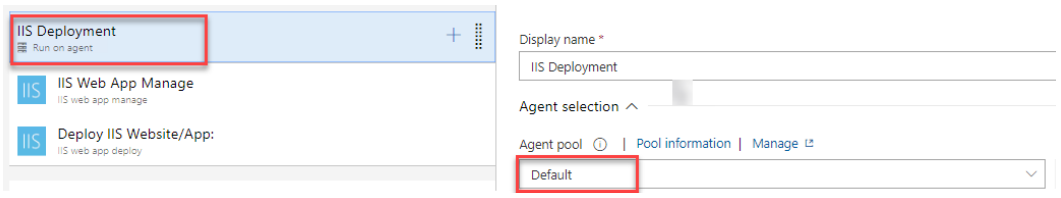


- **IIS Deployment phase**: In this phase, we deploy application to the VM using the following tasks-
    - **IIS Web App Manage**: This task runs on the target machine where we registered agent. It creates a *website* and an *Application Pool* locally with the name **PartsUnlimited** running under the port **82** , **http://localhost:82**
    - **IIS Web App Deploy**: This task deploys the application to the IIS server using **Web Deploy**.
- **Database deploy phase**: In this phase, we use **SQL Server Database Deploy** task to deploy **dacpac** file to the DB server.
- **Selenium tests execution**: Executing **UI testing** as part of the release process is a great way of detecting unexpected changes, and need not be

difficult. Setting up automated browser based testing drives quality in your application, without having to do it manually. In this phase, we will execute Selenium tests on the deployed web application. The below tasks describes using Selenium to test the websites in the release pipeline.

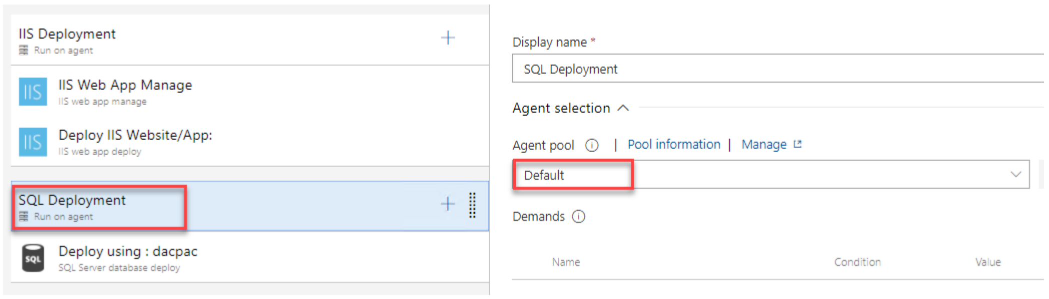
- **Visual Studio Test Platform Installer**: The Visual Studio Test Platform Installer task will acquire the Microsoft test platform from nuget.org or a specified feed, and add it to the tools cache. It satisfies the 'vstest' demand and a subsequent Visual Studio Test task in a build or release pipeline can run without needing a full Visual Studio install on the agent machine.
- **Run Selenium UI tests**: This task uses **vstest.console.exe** to execute the selenium testcases on the agent machines.

3. Click on **IIS Deployment** phase and select the **Default** Agent pool where we registered the agent in **Exercise 1**. In case if you have registered the agent to different agent pool, you need to select that.


IIS Deployment
Run on agent
IIS Web App Manage
IIS web app manage
Deploy IIS Website/App:
IIS web app deploy

Display name * IIS Deployment
Agent selection
Agent pool | Pool information | Manage
Default

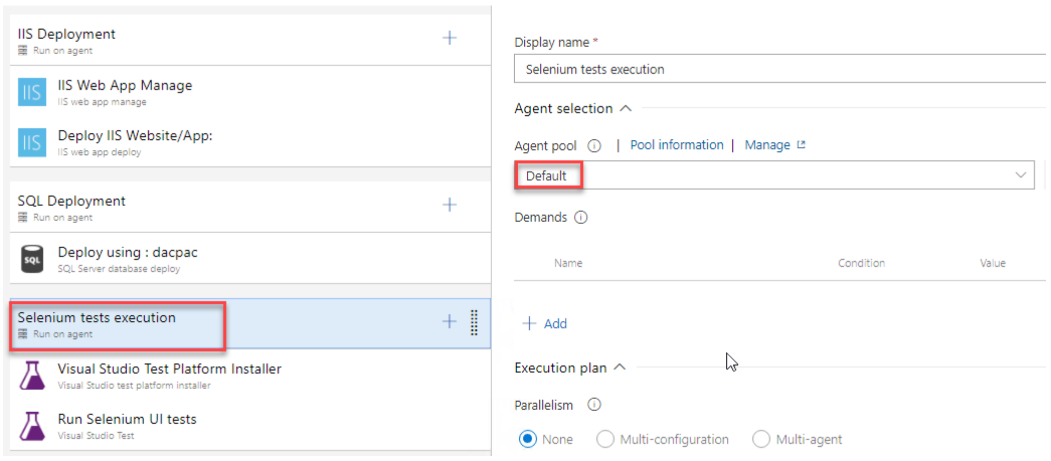
4. Repeat the above step for **SQL Deployment** phase


IIS Deployment
Run on agent
IIS Web App Manage
IIS web app manage
Deploy IIS Website/App:
IIS web app deploy
SQL Deployment
Run on agent
Deploy using : dacpac
SQL Server database deploy

Display name * SQL Deployment
Agent selection
Agent pool | Pool information | Manage
Default
Demands

Name	Condition	Value

5. Click on **Selenium tests execution** phase and set Agent pool to **Default** then **Save** the changes.


IIS Deployment
Run on agent
IIS Web App Manage
IIS web app manage
Deploy IIS Website/App:
IIS web app deploy
SQL Deployment
Run on agent
Deploy using : dacpac
SQL Server database deploy
Selenium tests execution
Run on agent
Visual Studio Test Platform Installer
Visual Studio test platform installer
Run Selenium UI tests
Visual Studio Test

Display name * Selenium tests execution
Agent selection
Agent pool | Pool information | Manage
Default
Demands

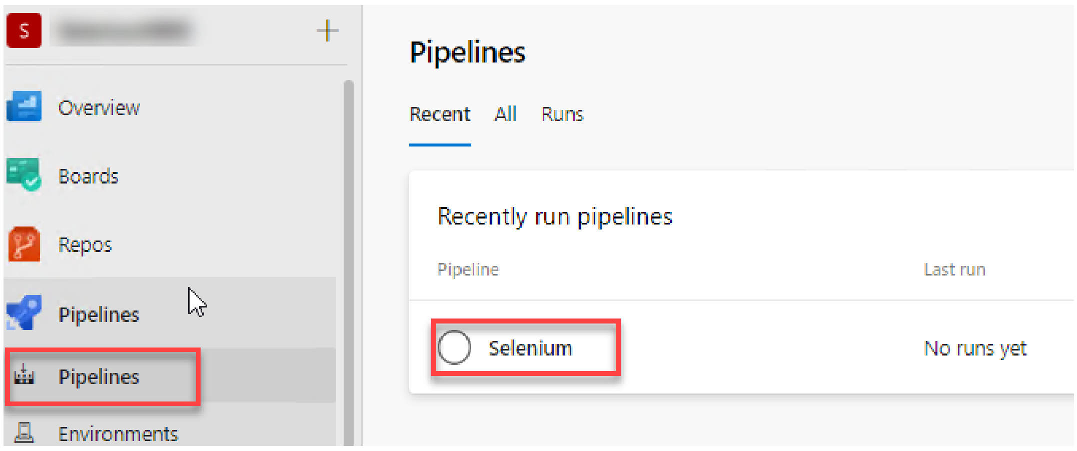
Name	Condition	Value

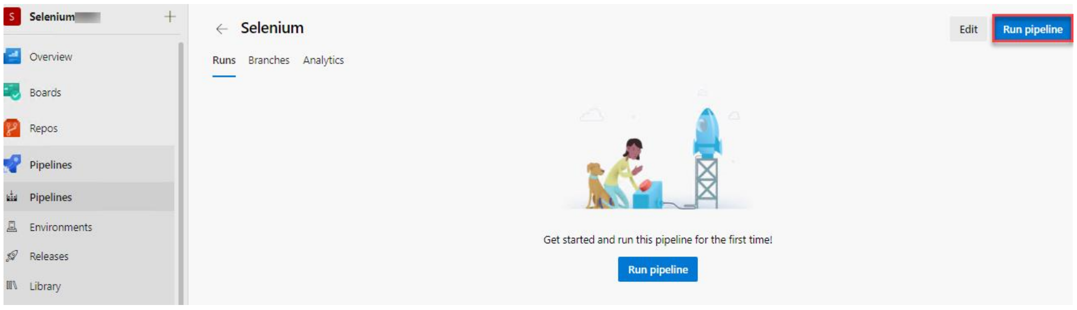
+ Add
Execution plan
Parallelism
[✓ None] Multi-configuration Multi-agent

Exercise 3: Trigger Build and Release

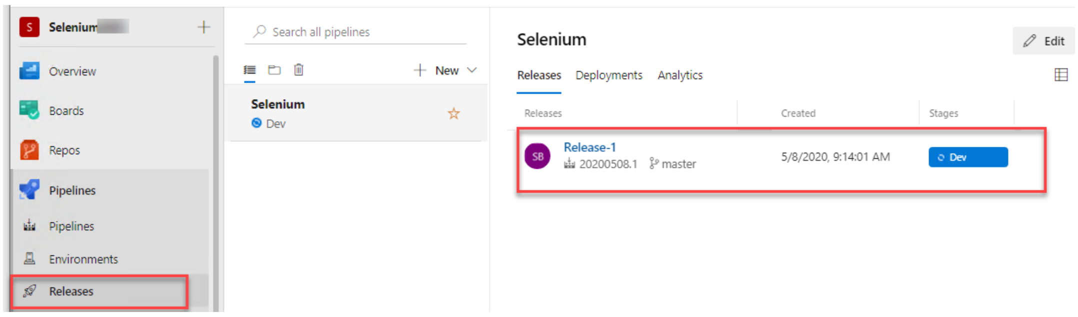
In this exercise, we will trigger the **Build** to compile Selenium C# scripts along with the Web application. The resulting binaries are copied to Azure VM and finally the selenium scripts are executed as part of the automated **Release**.

1. Navigate to **Pipelines** under **Pipelines**. Select **Selenium** build pipeline and click **Run pipeline**.



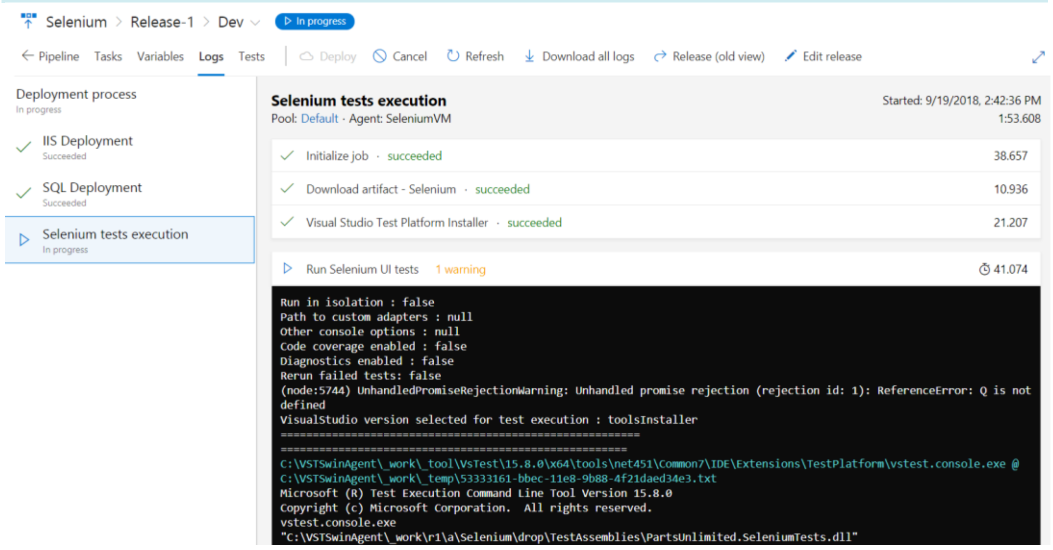


2. This build will publish the test artifacts to Azure DevOps, which will be used in release.
3. Once the build is successful, release will be triggered. Navigate to **Releases** tab to see the deployment in-progress.



4. When **Selenium test execution** phase starts, connect back to the VM provisioned earlier to see UI tests execution.

Note: It takes approximately 8 minutes to deploy test agent for the first time on VM. Once this task is complete, you can connect to the VM to see actual test execution.



Deployment process
In progress

IIS Deployment
Succeeded

SQL Deployment
Succeeded

Selenium tests execution
In progress

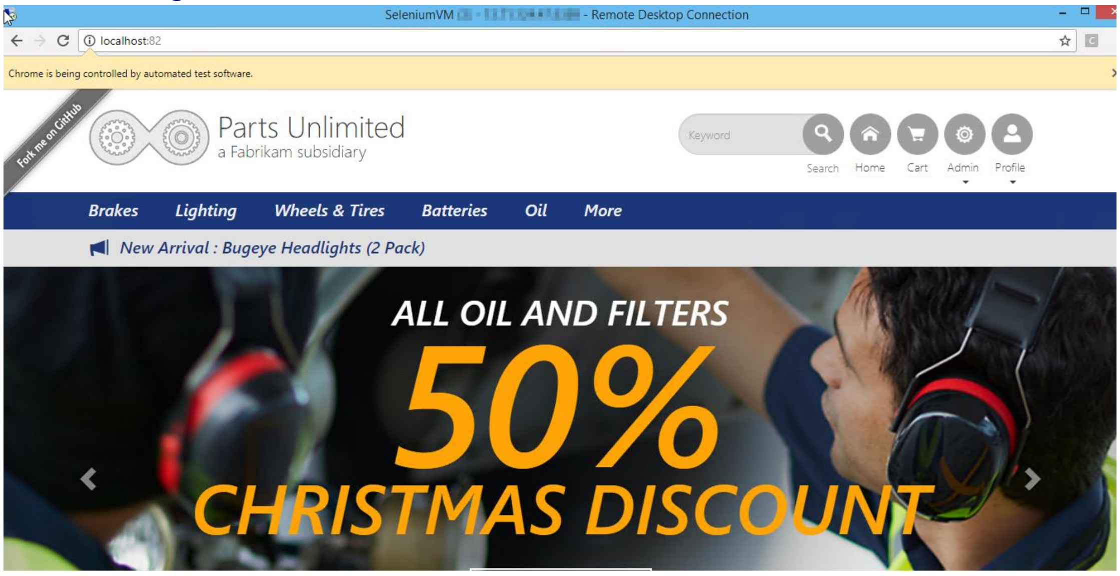
Selenium tests execution
Pool: Default · Agent: SeleniumVM
Started: 9/19/2018, 2:42:36 PM
1:53.608

✓	Initialize job · succeeded	38.657
✓	Download artifact - Selenium · succeeded	10.936
✓	Visual Studio Test Platform Installer · succeeded	21.207
▷	Run Selenium UI tests 1 warning	41.074

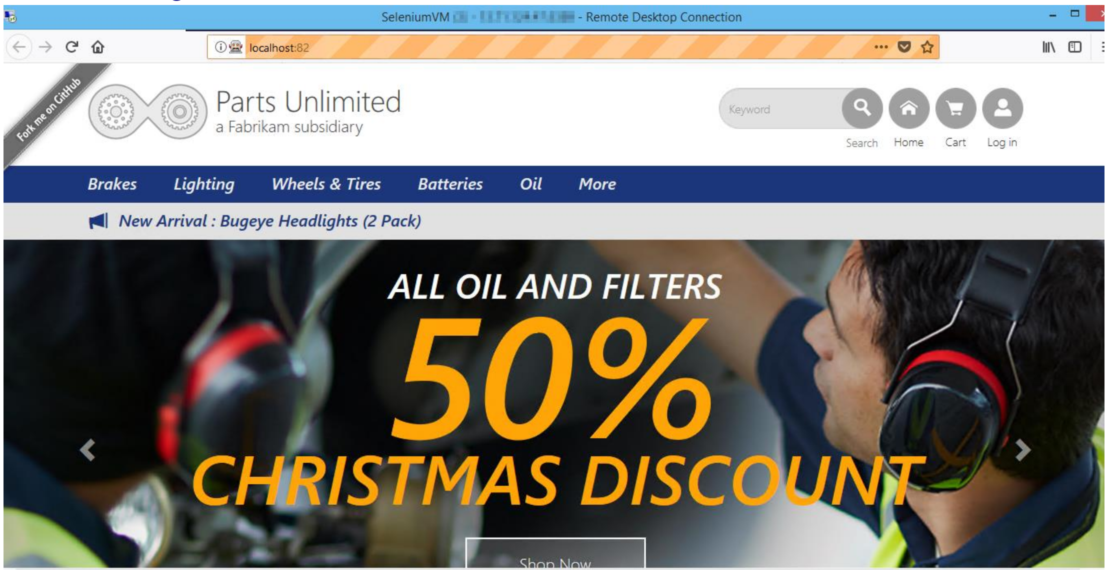
```
Run in isolation : false
Path to custom adapters : null
Other console options : null
Code coverage enabled : false
Diagnostics enabled : false
Rerun failed tests: false
(node:5744) UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id: 1): ReferenceError: Q is not
defined
VisualStudio version selected for test execution : toolsInstaller
========================================================
======================================================
C:\VSTSwinAgent\_work\_tool\VsTest\15.8.0\x64\tools\net451\Common7\IDE\Extensions\TestPlatform\vstest.console.exe @
C:\VSTSwinAgent\_work\_temp\53333161-bbec-11e8-9b88-4f21daed34e3.txt
Microsoft (R) Test Execution Command Line Tool Version 15.8.0
Copyright (c) Microsoft Corporation.  All rights reserved.
vstest.console.exe
"C:\VSTSwinAgent\_work\r1\a\Selenium\drop\TestAssemblies\PartsUnlimited.SeleniumTests.dll"
```

5. In this lab, we are executing **four** UI test scenarios configured to run on **Chrome** and **Firefox** browsers.

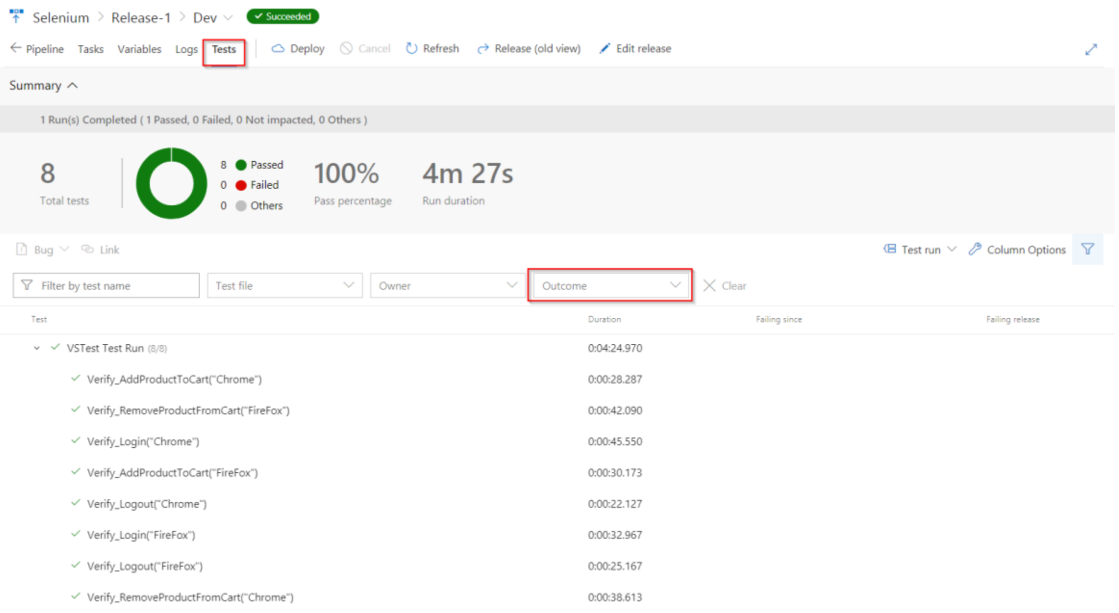
Tests running in Chrome



Tests running in Firefox



Once the release succeeds, click on the **Tests** tab to analyze the test results. Select the required filters from the dropdown in **Outcome** section to view the tests and their status.


Selenium > Release-1 > Dev ✓ Succeeded
Pipeline Tasks Variables Logs Tests
Deploy Cancel Refresh Release (old view) Edit release
Summary
Bug Link
Test run Column Options
Filter by test name
Test file
Owner
Outcome
Clear

1 Run(s) Completed (1 Passed, 0 Failed, 0 Not impacted, 0 Others)

8 Total tests

8 ● Passed
0 ● Failed
0 ● Others

100% Pass percentage

4m 27s Run duration

Test	Duration	Failing since	Failing release
✓ VSTest Test Run (8/8)	0:04:24.970		
✓ Verify_AddProductToCart("Chrome")	0:00:28.287		
✓ Verify_RemoveProductFromCart("FireFox")	0:00:42.090		
✓ Verify_Login("Chrome")	0:00:45.550		
✓ Verify_AddProductToCart("FireFox")	0:00:30.173		
✓ Verify_Logout("Chrome")	0:00:22.127		
✓ Verify_Login("FireFox")	0:00:32.967		
✓ Verify_Logout("FireFox")	0:00:25.167		
✓ Verify_RemoveProductFromCart("Chrome")	0:00:38.613		

Summary

In this lab, you have learnt how to automate and execute Selenium UI test cases on different browsers on an Azure VM with Azure Pipelines.