Fest 2018 Workshop

Building Applications for Relativity and RelativityOne

October 1, 2018 - Version 1.0

© Relativity. All rights reserved.

Contents

1 Course Overview 3

1.1 What You'll Learn 3

1.2 Required Development Software 3

2 Getting Started 4

2.1 Workshop Projects 4

2.2 DevVM Setup 5

3 Visual Studio Templates 9

3.1 Application Set Up 9

3.2 New Event Handler 13

3.3 Publish 18

4 Sample Application Overview 23

4.1 Install Sample Application 23

4.2 Run Sample Application 25

5 Unit Tests 28

5.1 Writing Unit Tests 28

5.2 Running Unit Tests 33

6 Integration Tests 35

6.1 Writing Integration Tests 35

6.2 Running Integration Tests 38

6.2 Running Integration Tests as CI 40

7 Gravity API 45

© Relativity. All rights reserved.

1 Course Overview

As a developer, you want a solid process for building and testing your apps, so you can focus on delivering value quickly. The Application Deployment System (ADS) is the standard way to build applications on the Relativity platform. Relativity Test Helpers is our new, open source testing framework.

In this advanced session, we'll review both of these standard approaches, covering common patterns for setting up effective ADS development solutions and integration tests. Topics will include: setting up your development environment using DevVM's; Visual Studio developer tools including templates, NuGet packages, and the "publish to Relativity" extension; and GitHub open source solutions such as Gravity, Test Helpers, and more. We'll also highlight ways to ensure your applications make a smooth transition from the on-premise world to RelativityOne. You will create tests and learn how to run those test from scripts as well as gain practical experience writing tests and potentially automating tests for your solutions, ensuring your customization will be ready run with the latest release of Relativity and RelativityOne.

1.1 What You'll Learn

- Relativity Templates
- Publish to Relativity
- Unit Tests
- Integration Tests
- Run Integration Tests as CI
- Gravity API

1.2 Required Development Software

This workshop and associated development resources require a development environment that includes the following items:

- Relativity 9.6.134.78 DevVM
- Visual Studio 2017
- Relativity Visual Studio Templates
- .NET Framework 4.6.2
- SQL Server 2016

We have provided the required software for you for this workshop.

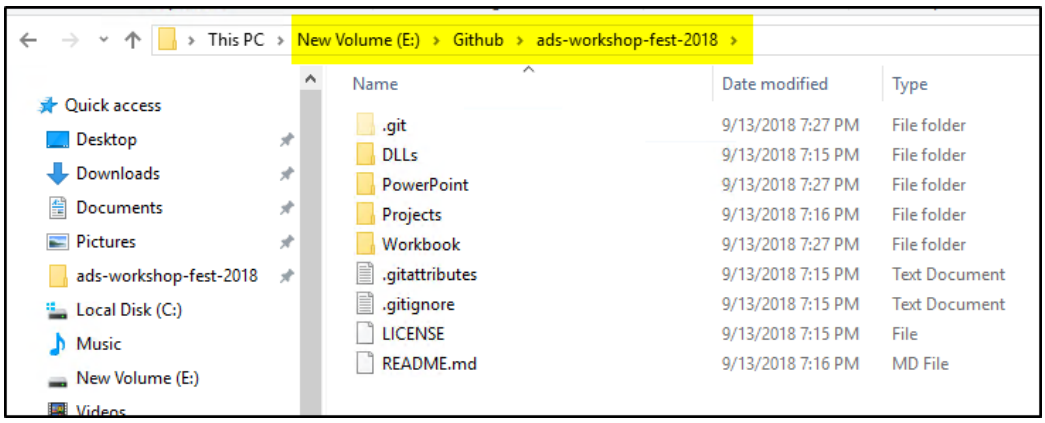
© Relativity. All rights reserved.

2 Getting Started

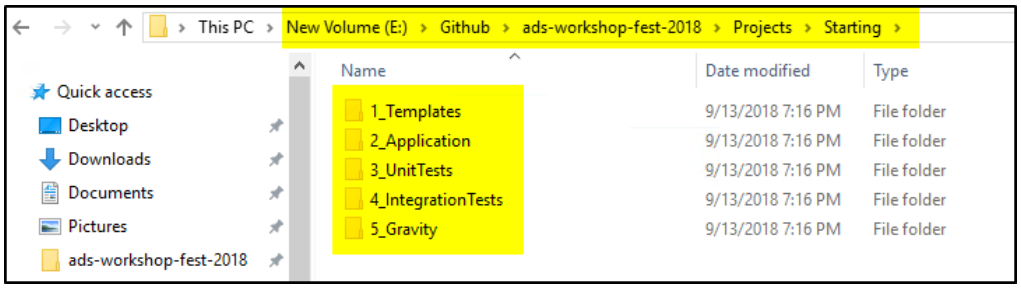
2.1 Workshop Projects

1. Make sure you have a folder shortcut on Desktop with name **ads-workshop-fest-2018**.

2. Double click on the **ads-workshop-fest-2018** folder to open it. Make sure you see the files as shown in the below screenshot.



3. Go to the **Projects/Starting** folder to make sure you see the folders as shown in the below screenshot. These folders contain Visual Studio projects for different sections in the workbook.

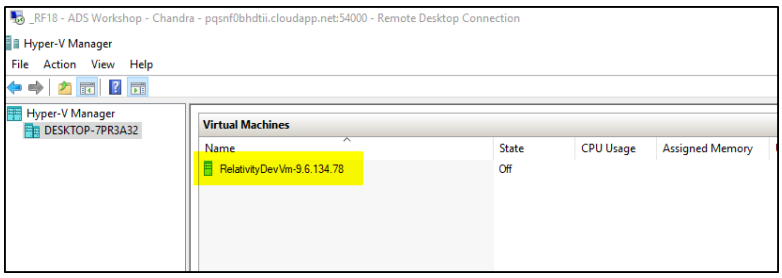


© Relativity. All rights reserved.

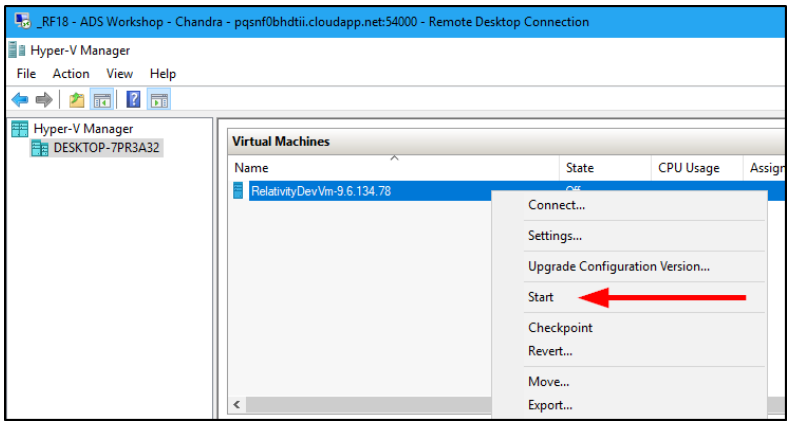
2.2 DevVM Setup

1. Open the **Hyper-V Manager** application by clicking the blue icon in the taskbar.

2. You will see a **Relativity 9.6.134.78 DevVM** listed in the Hyper-V Manager **Virtual Machines** section.

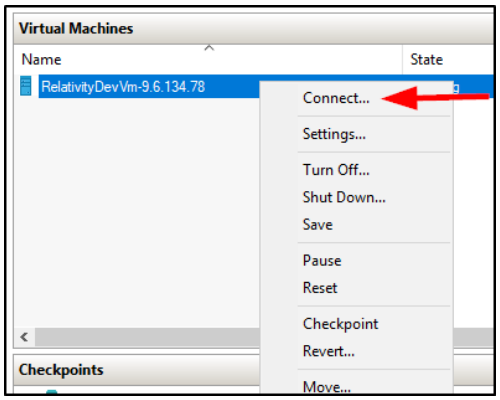


3. Right click on the **DevVM** and select the **Start** option.

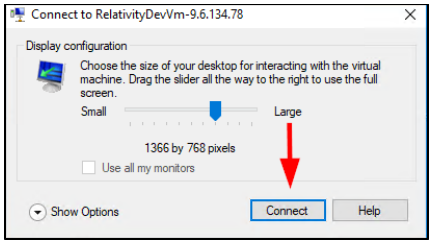


© Relativity. All rights reserved.

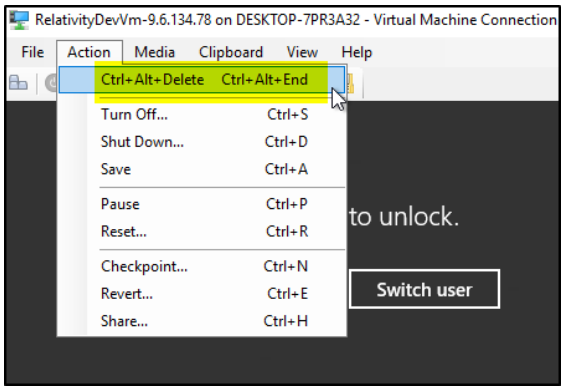
4. Right click on the **DevVM** and select the **Connect** option.

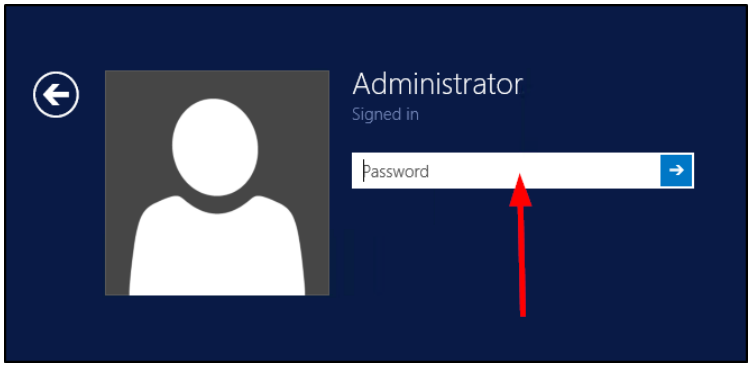


5. If you get a Display configuration pop-up, click the **Connect** button.



6. Select the **Action** Menu item and then select **Ctrl+Alt+Delete** option.





7. Use the below DevVM credentials highlighted in yellow to login to the VM.

© Relativity. All rights reserved.

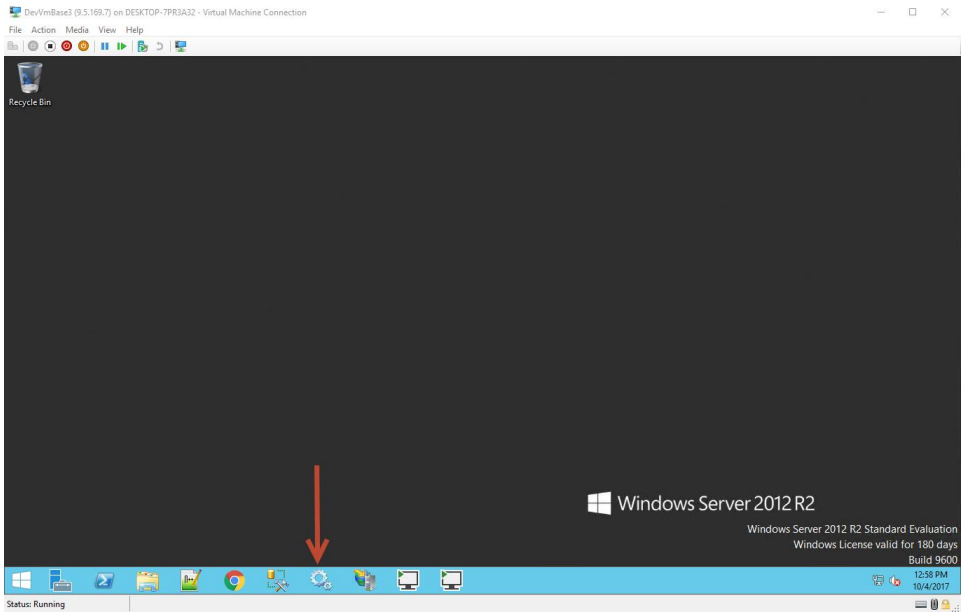
DevVM:
Windows Admin login: Administrator
Windows Admin password: Test1234!

Relativity:
Admin login: relativity.admin@relativity.com
Admin password: Test1234!

SQL:
login: eddsdbo
password: Test1234!

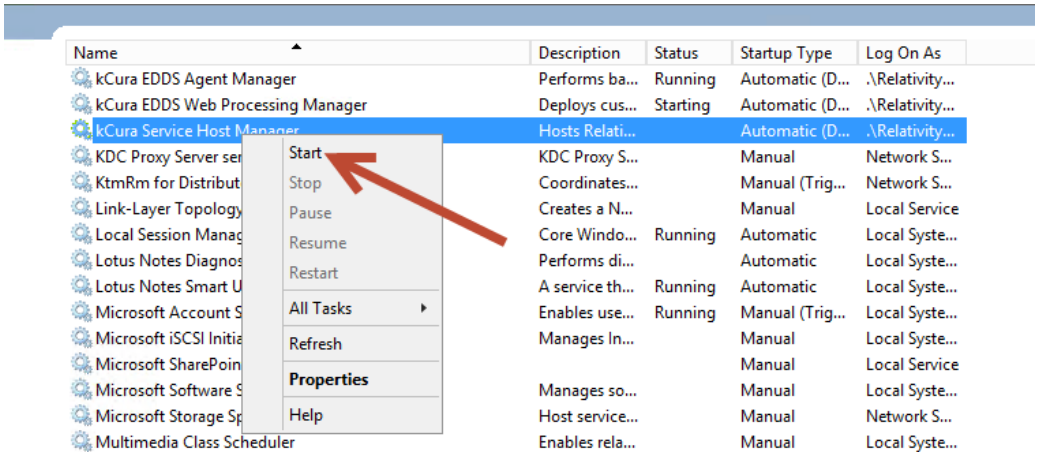
Admin login: sa
Admin password: Test1234!

8. Once you log in to the DevVM, you should see the Desktop with few applications pinned to the taskbar, which you will be using throughout the workshop.
9. Click on the **Services** program icon in the taskbar as shown in the below screenshot.

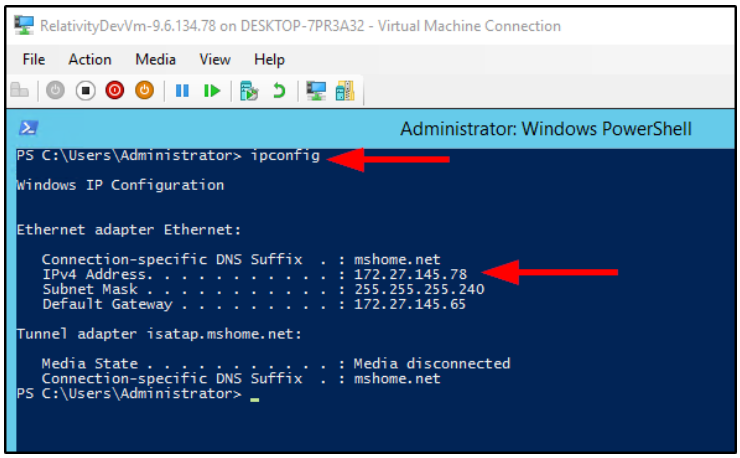


10. Make sure all the **services** listed below are **running**. If they are not already running, right click on the service and select the **Start** option,
    - Service Bus Gateway
    - Service Bus Message Broker
    - Service Bus Resource Provider
    - Service Bus VSS
    - kCura Service Host Manager
    - kCura EDDS Web Processing Manager
    - kCura EDDS Agent Manager

© Relativity. All rights reserved.

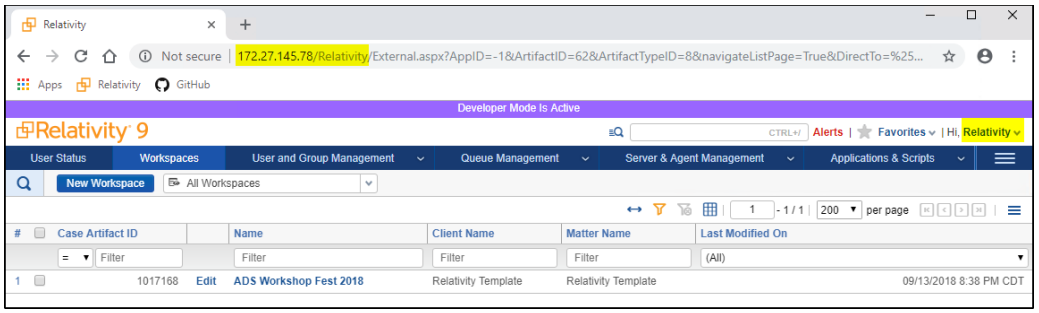


11. Open PowerShell on your DevVM and get the IP address of the VM.



12. Open **Chrome** on your Workshop machine (Not the DevVM) and go to **http://172.27.145.78/Relativity**. Use the below Relativity admin credentials to login. Verify that you can login to Relativity.
    a. If the workspaces tab doesn't load, give it a few mins and then try refreshing the tab.

Relativity:
Admin login: relativity.admin@relativity.com
Admin password: Test1234!



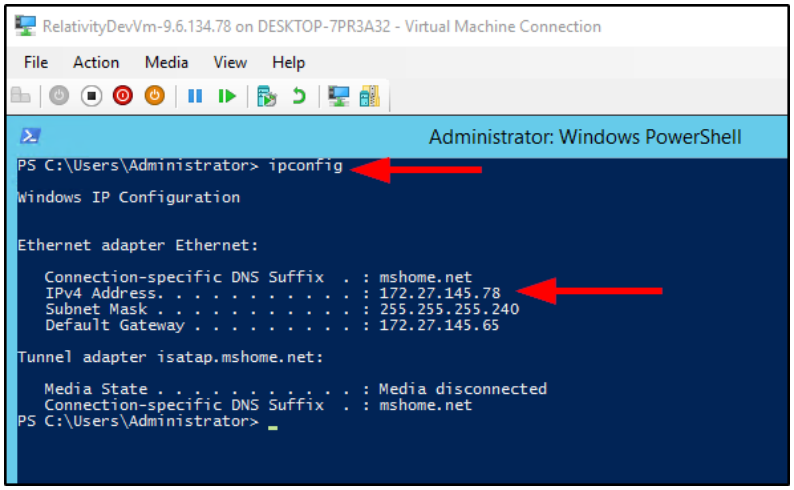
© Relativity. All rights reserved.

3 Visual Studio Templates

Relativity is an extensible platform that allows an administrator/developer to create customizations. Development on the Relativity Platform is made easier with the use of some tools such as Visual Studio Templates and the Publish to Relativity GUI. This section demonstrates utilizing the visual studio templates and Publish to Relativity tool to quickly iterate on application development.

3.1 Application Set Up

1. Open PowerShell on your DevVM and get the IP address of the VM.

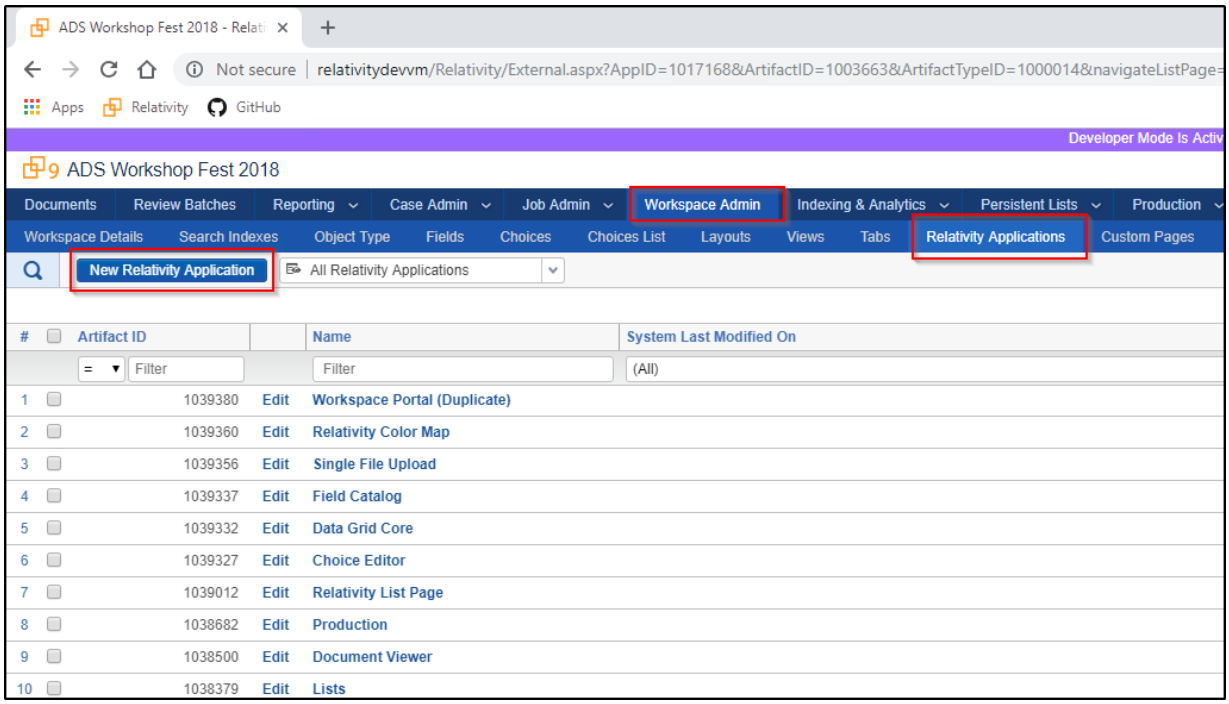


2. Open **Chrome** on your Workshop machine (Not the DevVM) and go to **http://172.27.145.78/Relativity**. Use the below Relativity admin credentials to login.
    a. If the workspaces tab doesn't load, give it a few mins and then try refreshing the tab.

Relativity:
Admin login: relativity.admin@relativity.com
Admin password: Test1234!

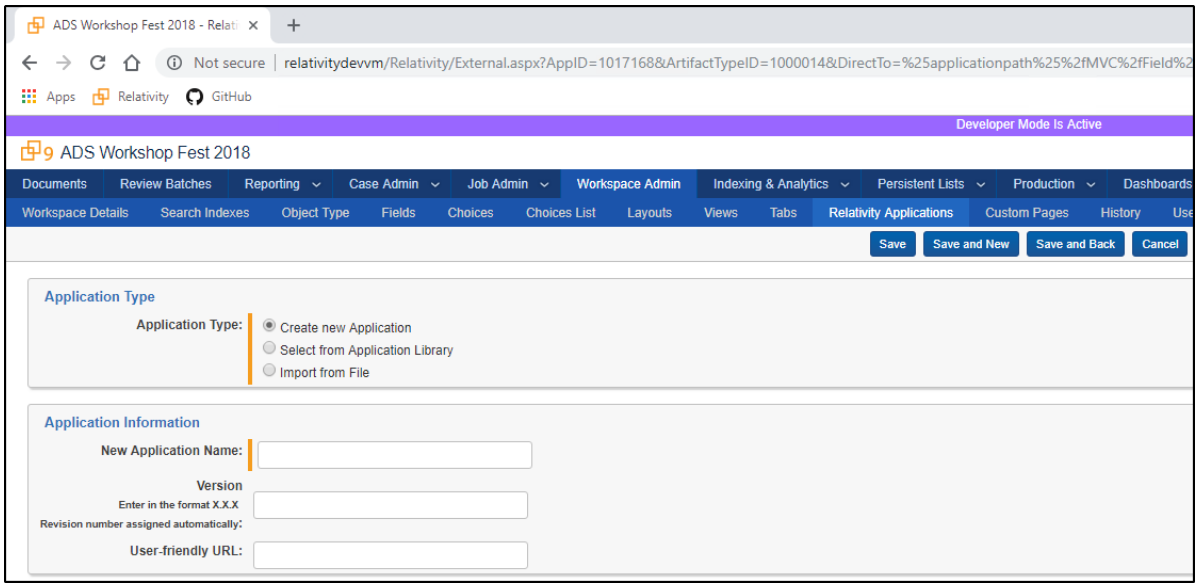
3. Enter **ADS Workshop Fest 2018** and navigate to the **Relativity Applications** Tab.

© Relativity. All rights reserved.

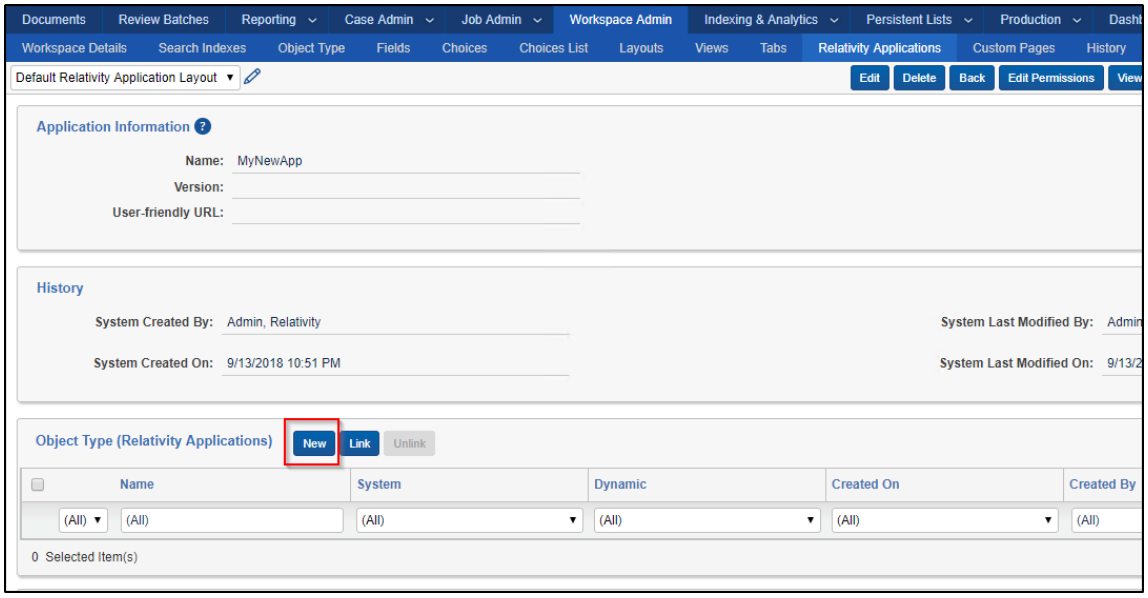


© Relativity. All rights reserved.

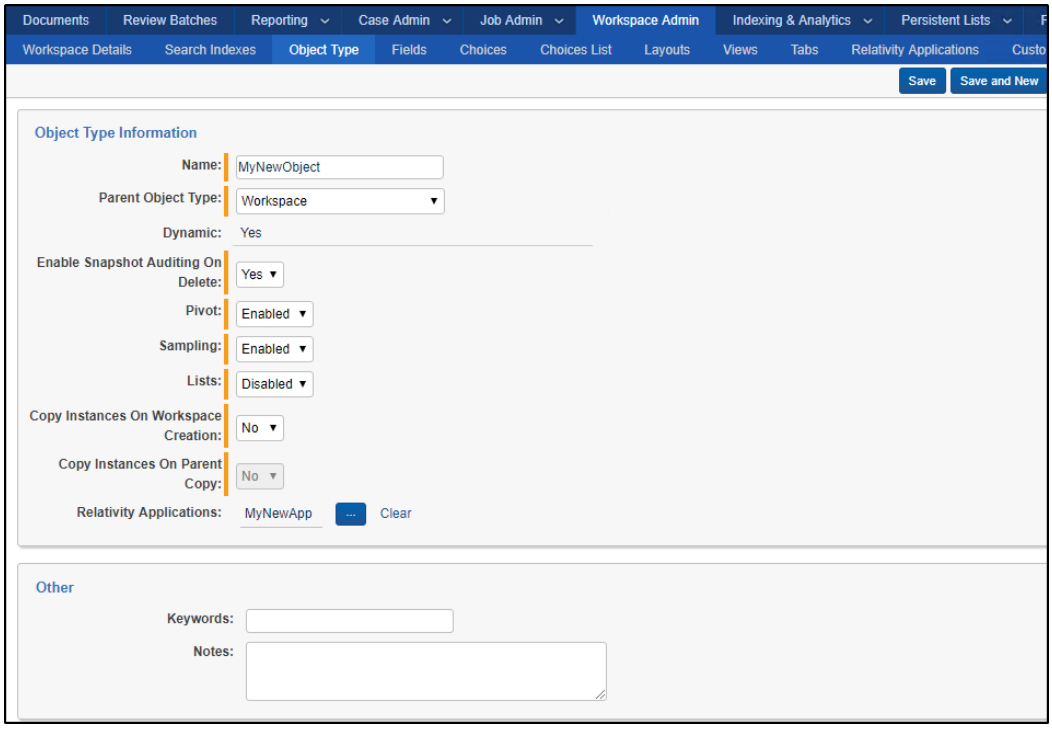
4. Create a new **Relativity Application** with the name of your choice.



5. Create a new **Object Type** to the application from the view page. This auto-associates the objects initial views, fields, and layouts.



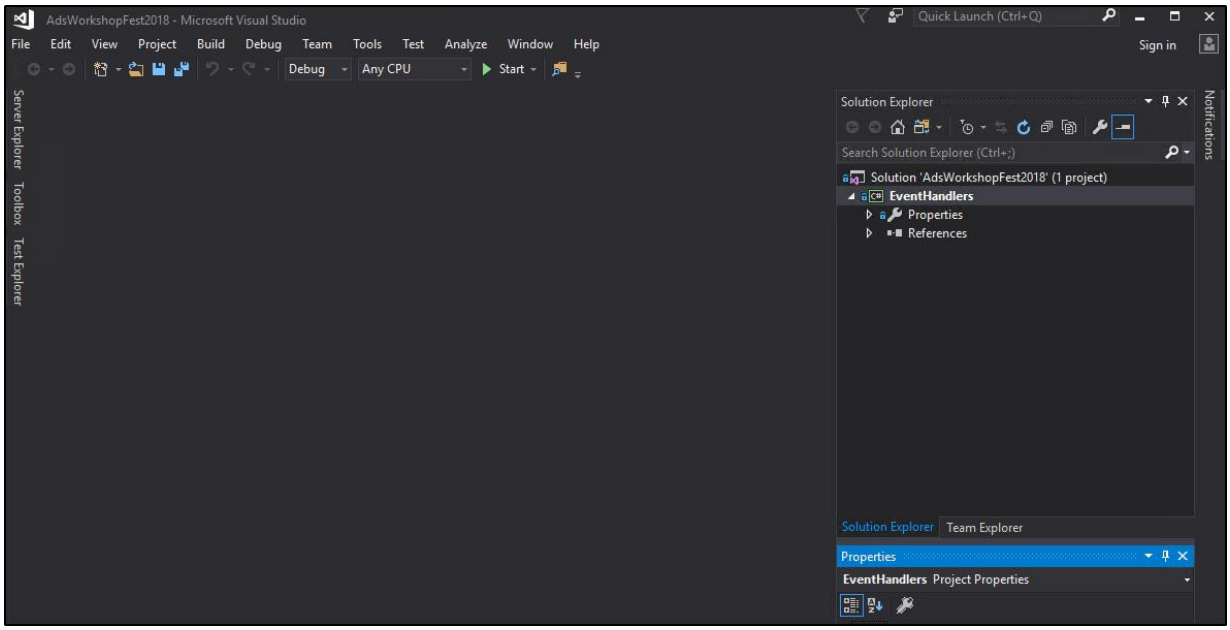
© Relativity. All rights reserved.



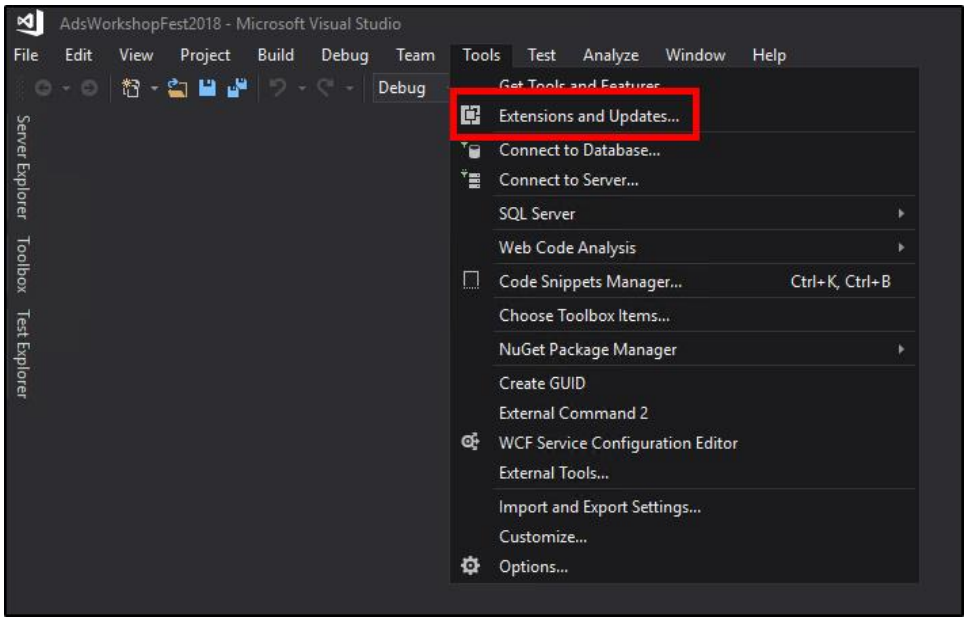
© Relativity. All rights reserved.

3.2 New Event Handler

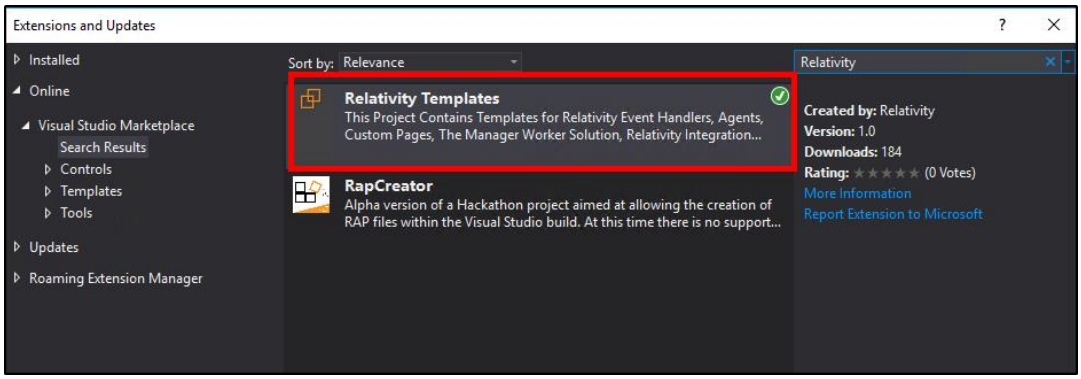
1. Open the Visual Studio solution **AdsWorkshopFest2018.sln** in the **E:\Github\ads-workshop-fest-2018\Projects\Starting\1_Templates\Project** folder.



2. Verify the **Relativity Templates** are installed. You may need to **restart** Visual Studio.
3. **Tools** — **Extensions and Updates** — Search "**Relativity**"

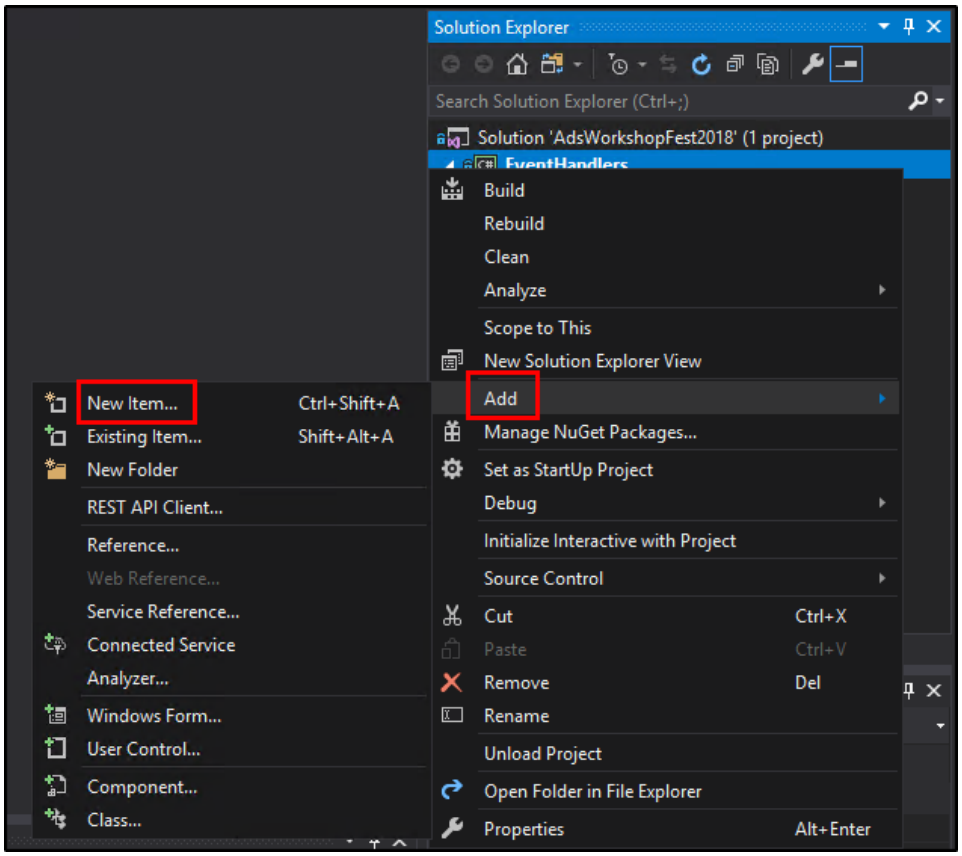


© Relativity. All rights reserved.

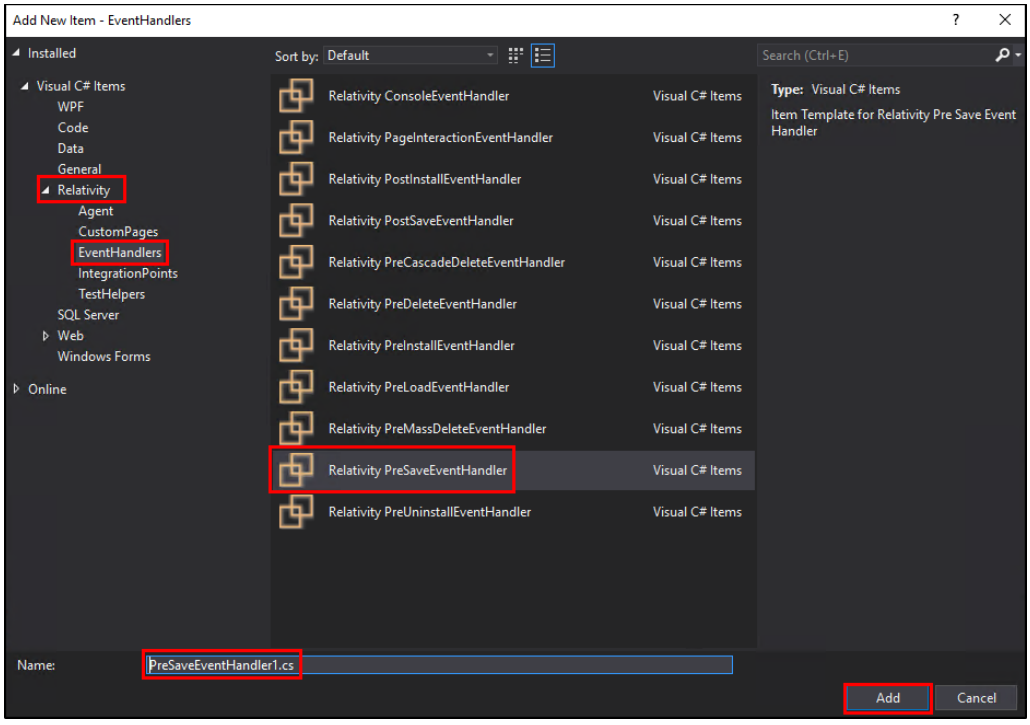


© Relativity. All rights reserved.

4. Right click the **EventHandlers** Project in the solution explorer. Select Add — **New Item**.

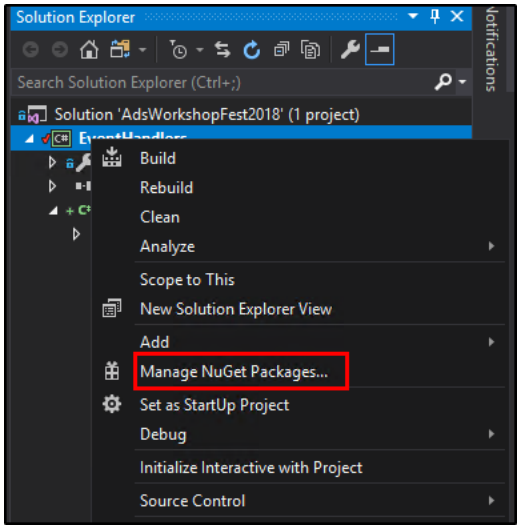


5. Under the Relativity Section, select the **pre-save event handler template.**

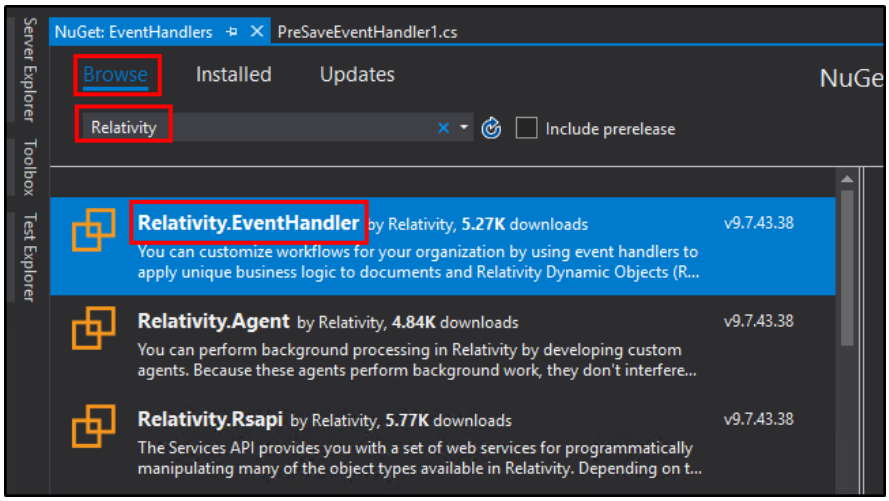


© Relativity. All rights reserved.

6. Click **Add**.
7. Install the 9.6.134.78 **Relativity.EventHandler** NuGet package.
    a. Right click on the Project.
    b. Select **Manage Nuget Packages** option.

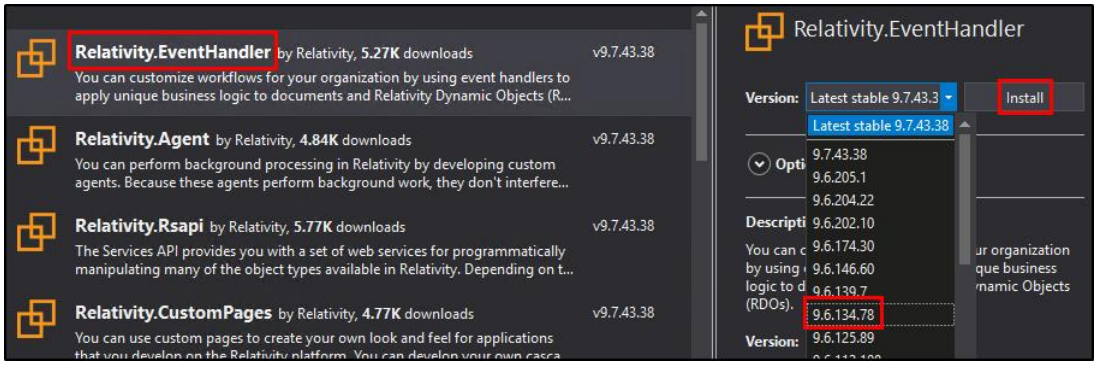


    c. Switch to **Browse** tab.
    d. Search for "**Relativity**".
    e. Select the **Relativity.EventHandler** NuGet package.

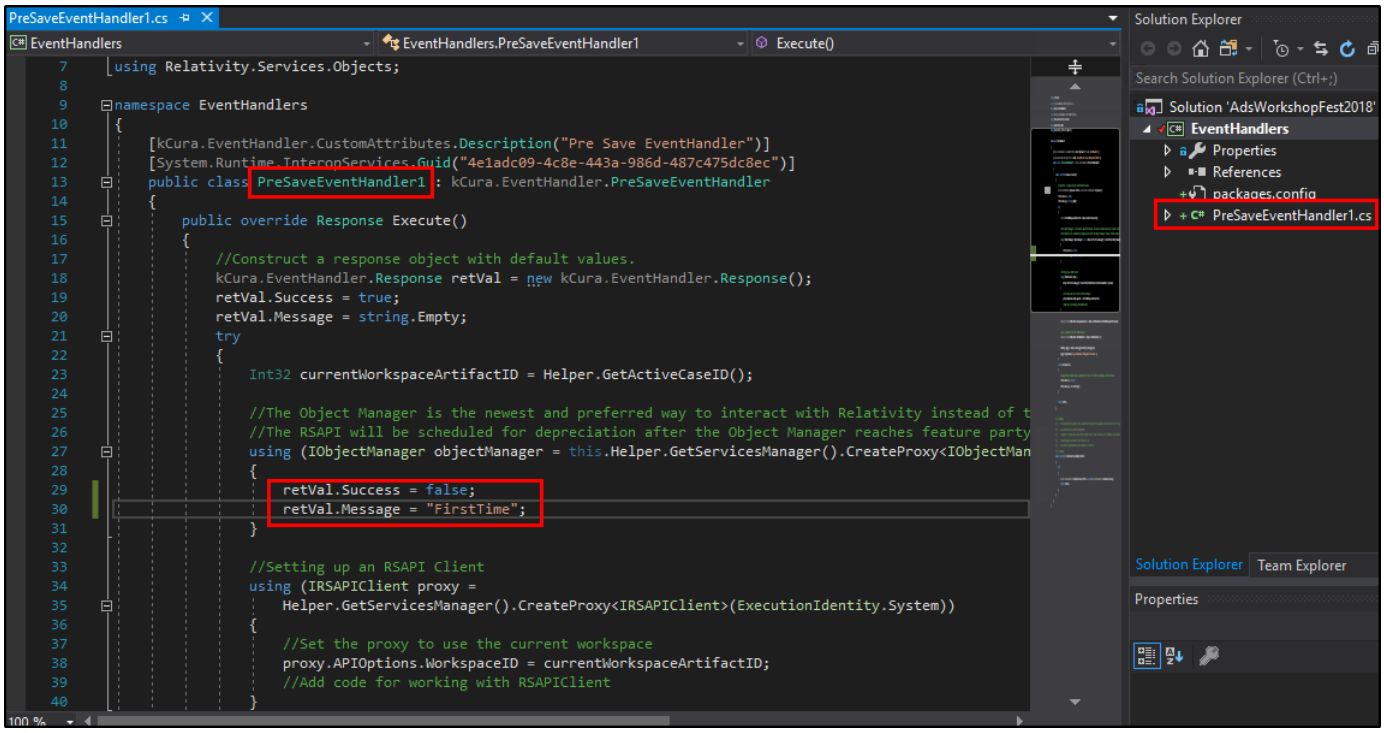


    f. From the **Version** drop-down select the **9.6.134.78** option.
    g. Click **Install**.

© Relativity. All rights reserved.



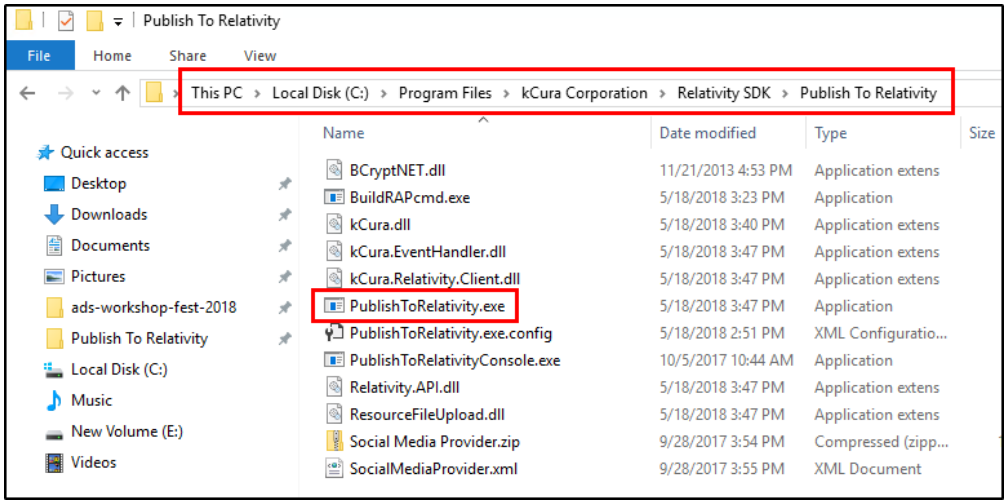
8. Update the event handler response.
    a. *Success* – false
    b. *Message* – Message of your choice



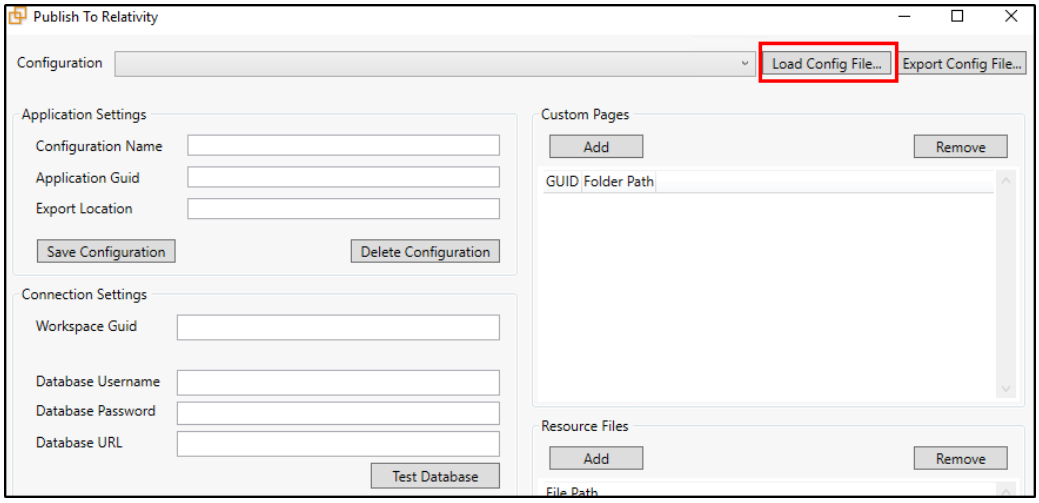
© Relativity. All rights reserved.

3.3 Publish

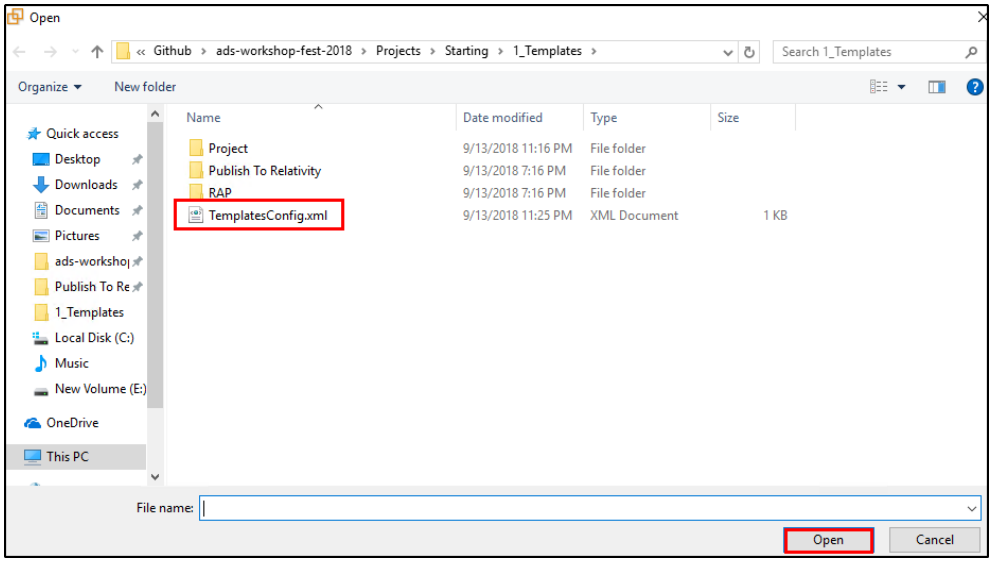
1. Build the solution.
2. **Build — Build Solution**
3. Run the Publish to Relativity executable.



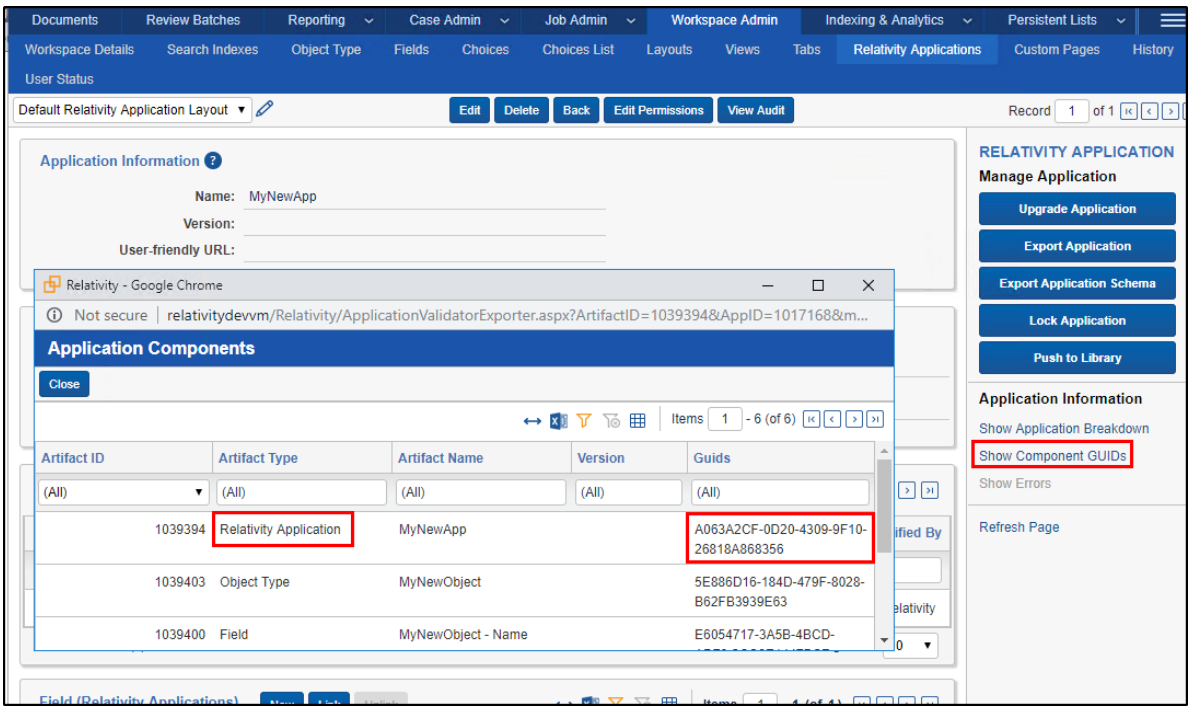
4. Click the **Load Config File** button and select the **TemplatesConfig.xml** file in the **E:\Github\ads-workshop-fest-2018\Projects\Starting\1_Templates** folder.



© Relativity. All rights reserved.



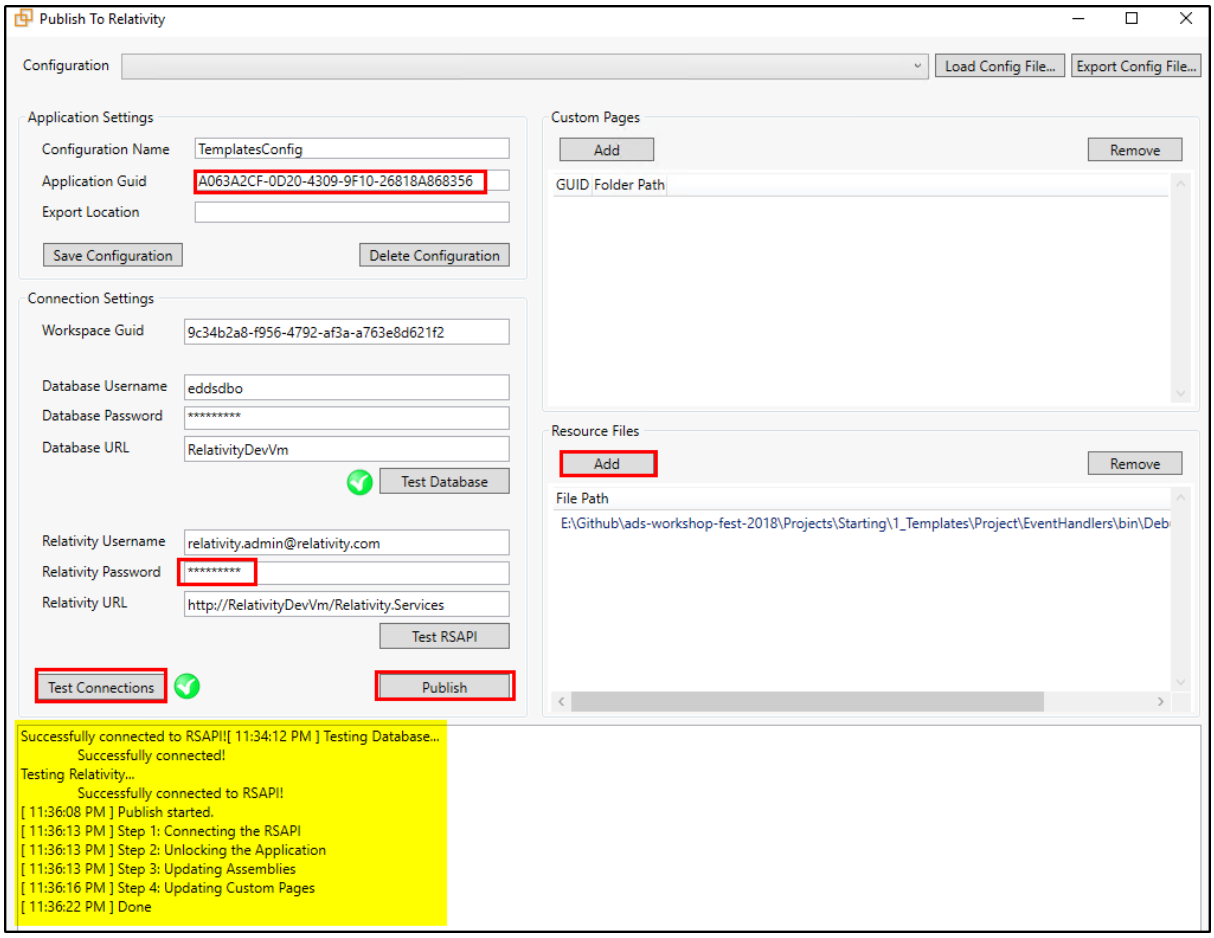
5. Enter the **Relativity Password** (Test1234!).
6. Click on **Test Connections** to validate the connections.
7. Replace the **Application GUID** with your applications GUID.
    a. To look up your applications GUID, navigate back to Relativity Application you created in the workspace. Then select **Show Component GUIDs**.
    b. **Developer Mode** must be active.



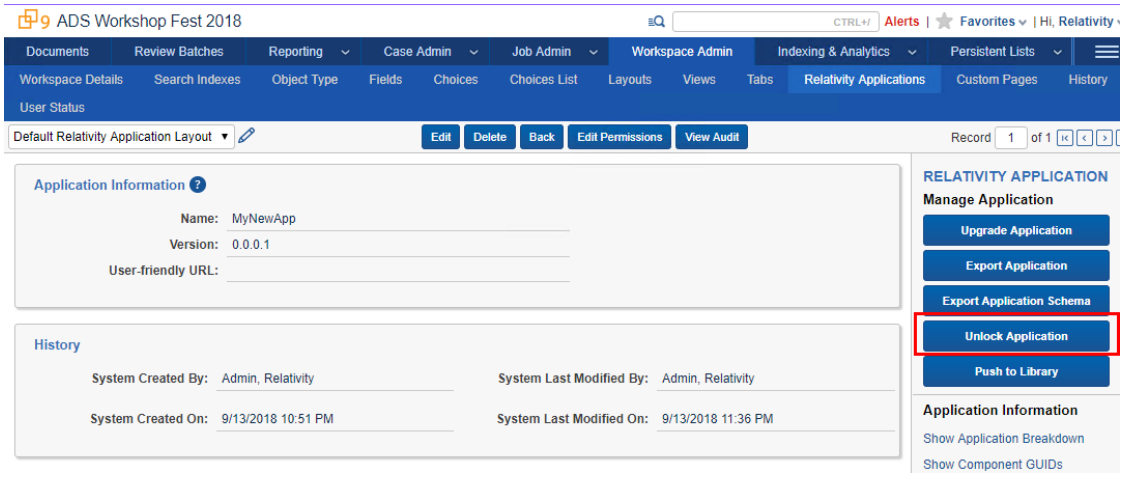
8. In the **Resource Files** Section click Add.
9. Navigate to **E:\Github\ads-workshop-fest-2018\Projects\Starting\1_Templates\Project\EventHandlers\bin\Debug** and select **EventHandlers.dll** file.

© Relativity. All rights reserved.

10. Click **Publish**.

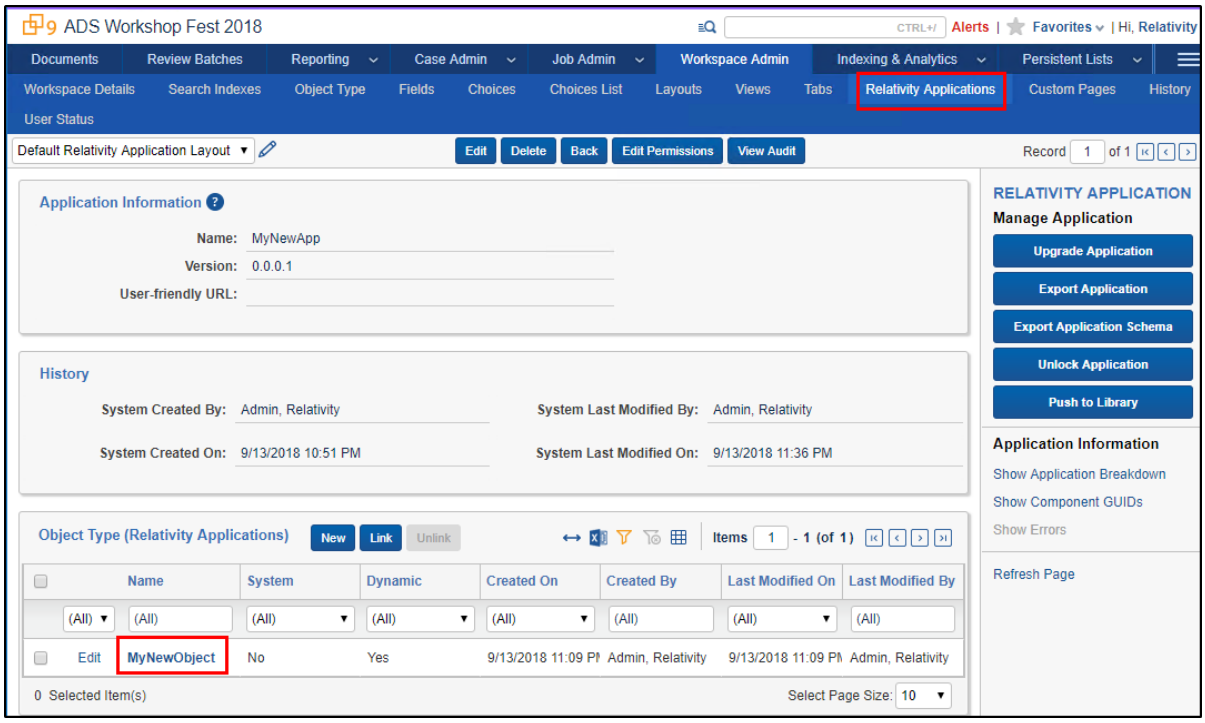


11. Open Relativity and navigate to your application created in section 3.1.
12. **Unlock** the application.

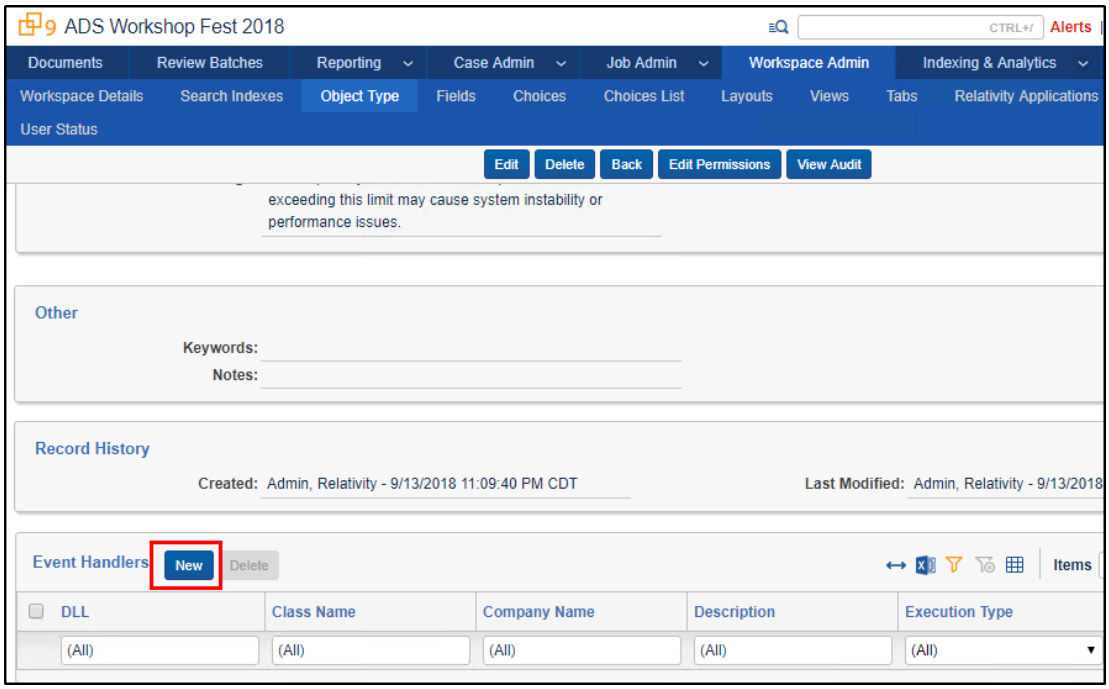


© Relativity. All rights reserved.

13. Navigate to the view page of the object type created in section 3.1.

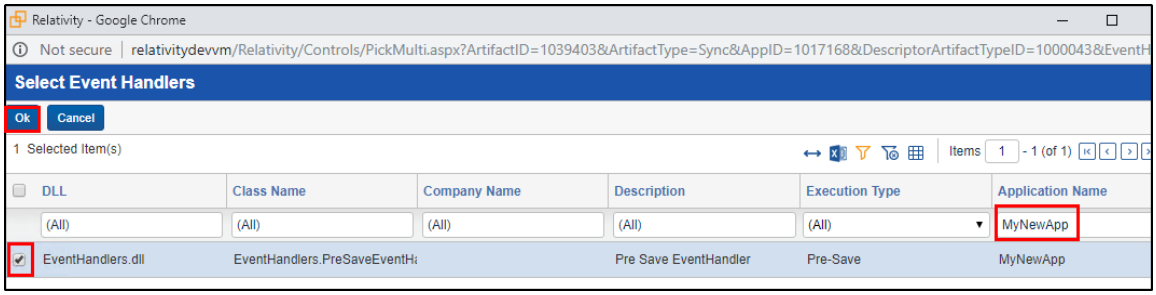


14. Click **New** on the Event handler associated item list.

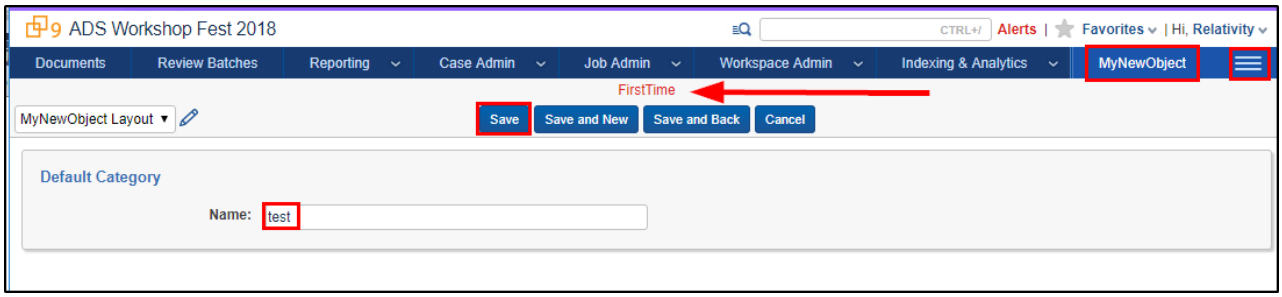


15. Find the newly created Event Handler associated to your application.
    a. Filter by Application Name.
    b. Select it from the list and click **Ok**.

© Relativity. All rights reserved.



16. Test your event handler by creating a new instance of your object type.



© Relativity. All rights reserved.

4 Sample Application Overview

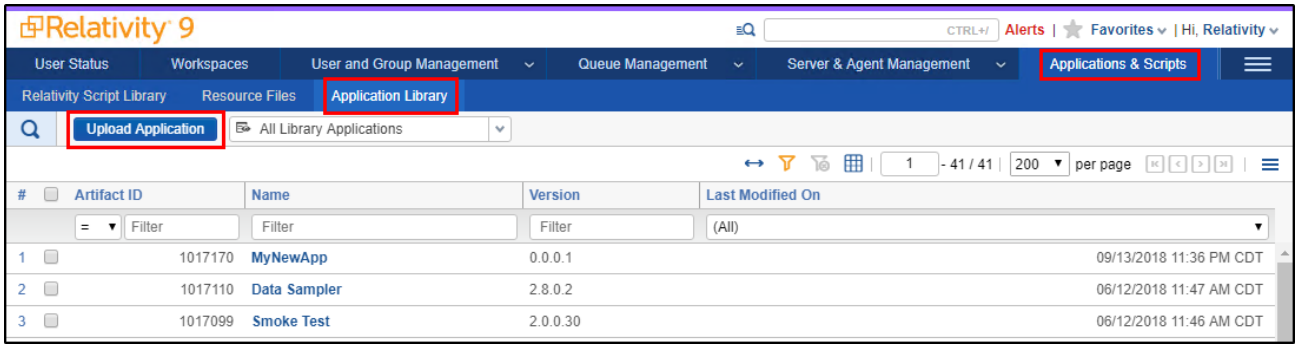
The functionality of the Sample Relativity application (ADS Workshop Fest 2018) is to calculate the selected Instance metrics and write them to the Instance Metrics Job.

The ADS Workshop Fest 2018 Relativity application consists of the following:

- **Custom Object**
  - Instance Metrics Job
    - Fields
      - Metrics
      - Status
      - Workspaces Count
      - Users Count
      - Groups Count
      - Errors
- **Event Handler**
  - PreSaveStatusUpdate
- **Agent**
  - Instance Metrics Calculator Agent

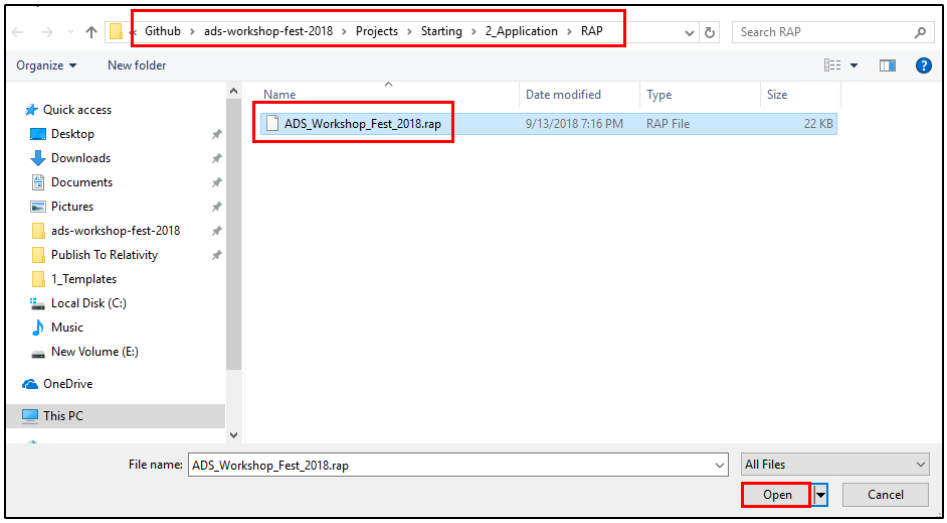
4.1 Install Sample Application

1. Go to the **Application Library** tab.
2. Click on the **Upload Application** button

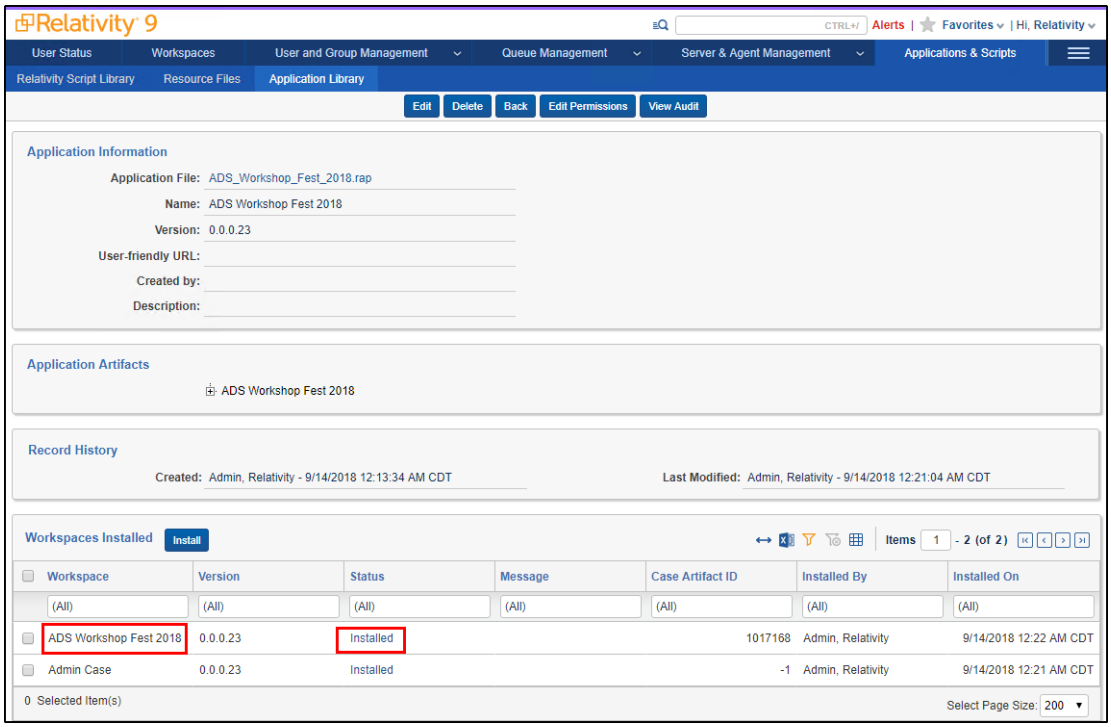


3. Click on **Choose File** button.
4. Navigate to **E:\Github\ads-workshop-fest-2018\Projects\Starting\2_Application\RAP** folder and select the **ADS_Workshop_Fest_2018.rap** file.

© Relativity. All rights reserved.



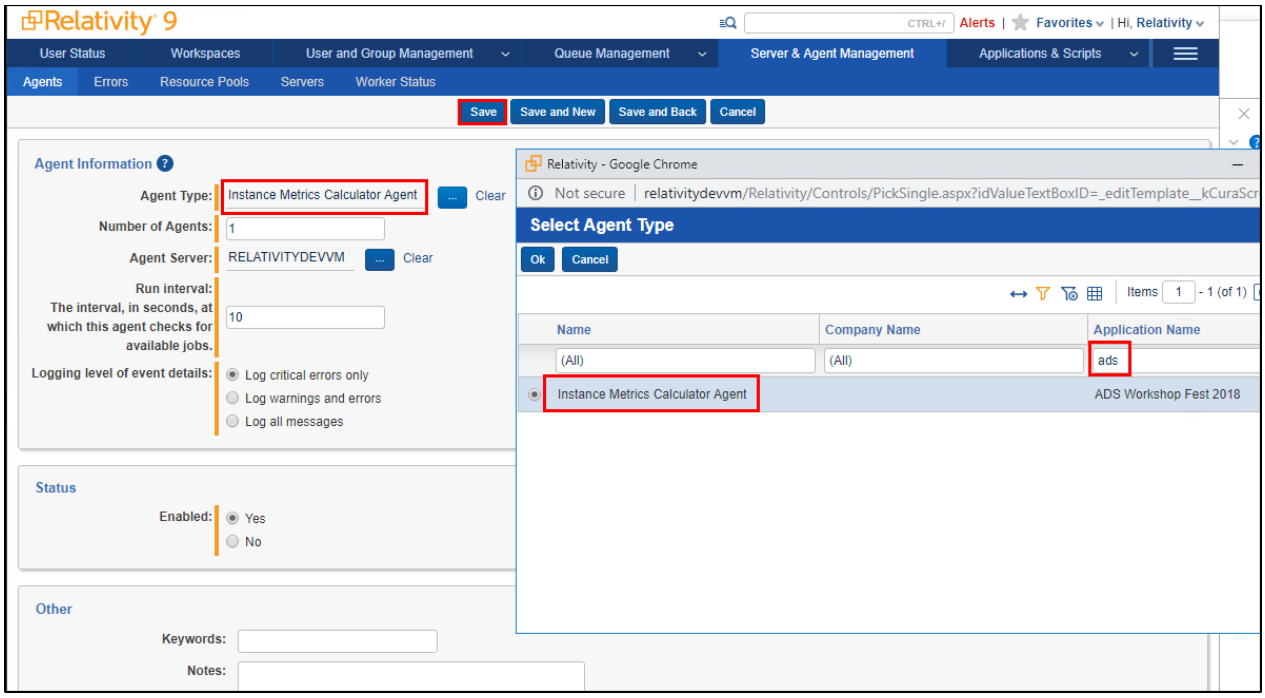
5. Click the **Save** button.
6. Under the **Workspaces Installed** section, click the **Install** button.
7. Select the **ADS Workshop Fest 2018** workspace and click the **Save** button.

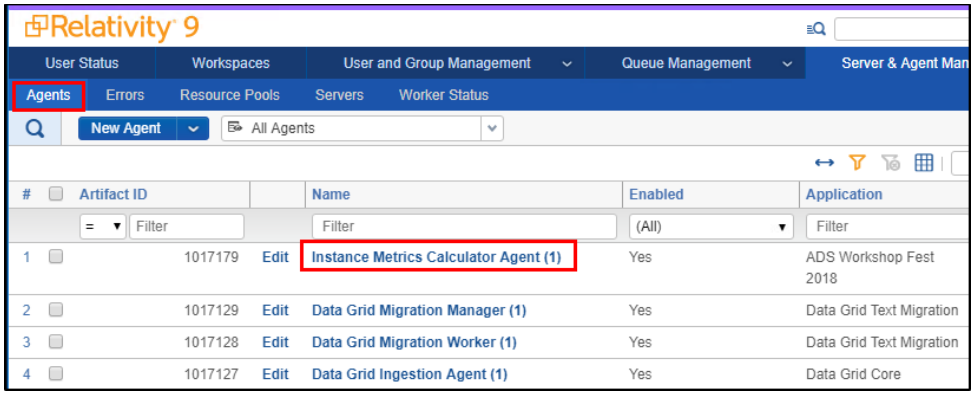


© Relativity. All rights reserved.

4.2 Run Sample Application

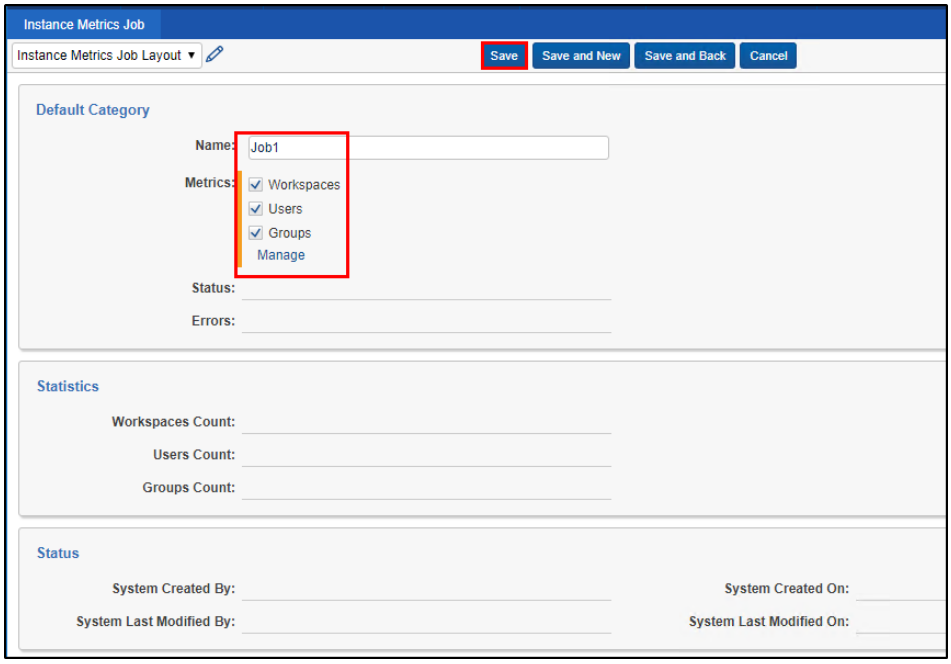
1. Once the application is installed, navigate to the **Agents** tab and create an **Instance Metrics Calculator Agent** agent.

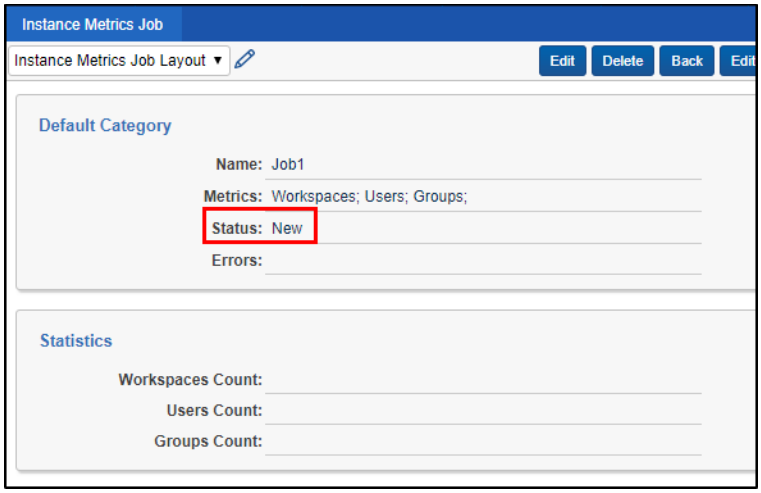




2. Now navigate the **Instance Metrics Job** tab in **ADS Workshop Fest 2018** workspace.
3. Create a new **Instance Metrics Job** and click on the **Save** button.

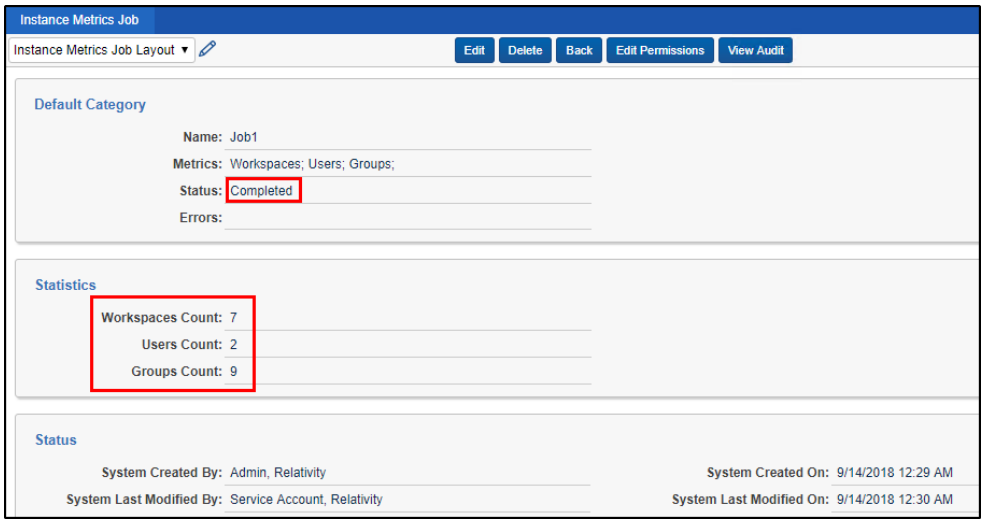
© Relativity. All rights reserved.





4. The **Instance Metrics Calculator Agent** which is running every 10 seconds will pick up the job and calculate the metrics and save it to the job.
5. Keeping refreshing the windows every few seconds until the job status is set to **Completed**.

© Relativity. All rights reserved.

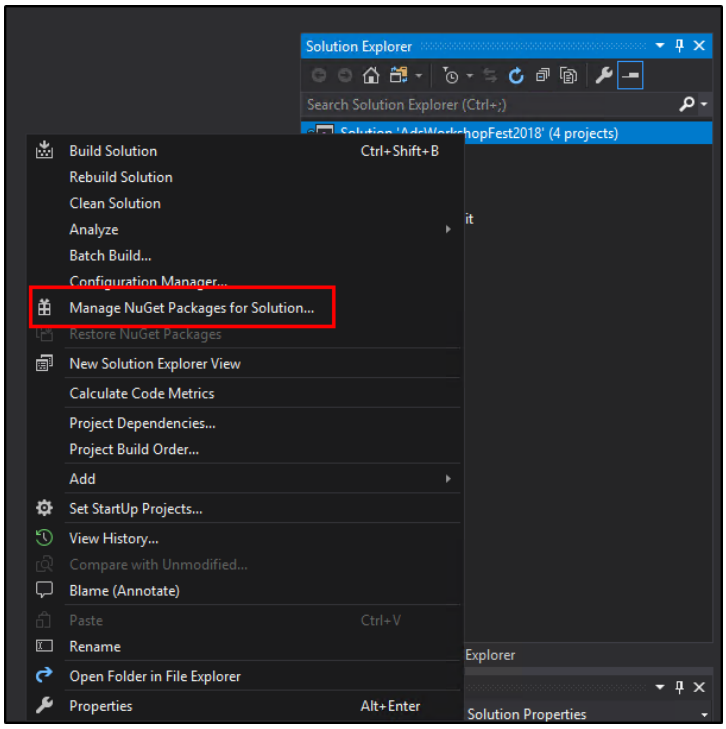


© Relativity. All rights reserved.

5 Unit Tests

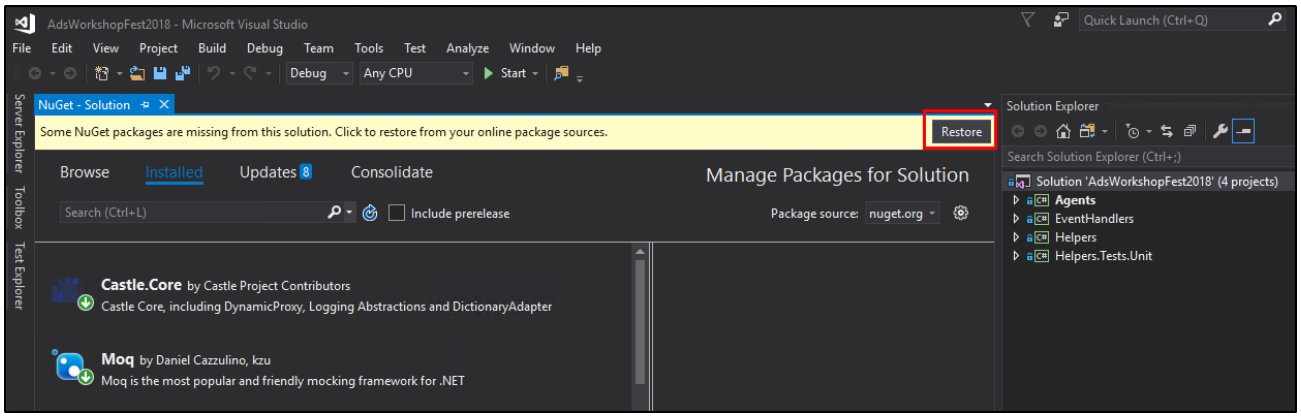
5.1 Writing Unit Tests

1. Navigate to the **E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\Project** folder and open **AdsWorkshopFest2018.sln** file.
2. In this solution, you will find 4 projects.
    a. Agents
    b. EventHandlers
    c. Helpers
    d. Helpers.Tests.Unit
3. In this section we will be working in the **Helpers.Tests.Unit** project.
4. In **Helpers** project, there is an **RsapiHelper.cs** file which contains RSAPI calls used in the sample application.
5. Our goal for this section is to write couple unit tests for the **RetrieveJobsInWorkspaceWithStatus** in **RsapiHelper** class.
6. Right click on the solution and select **Manage NuGet Packages for Solution** option.

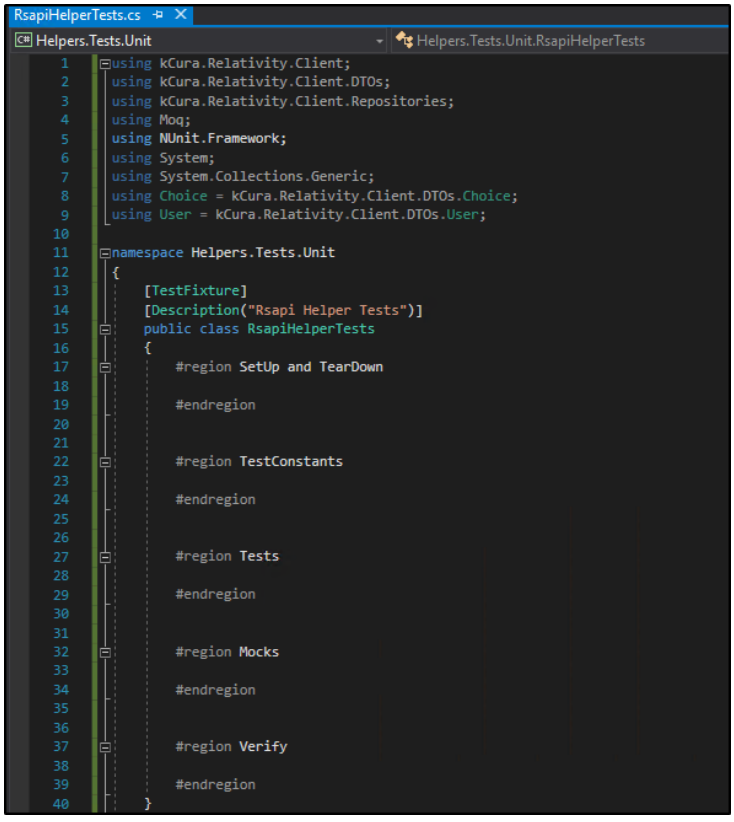


7. Next click the **Restore** button.

© Relativity. All rights reserved.



8. Now all the NuGet packages in the solution should be restored.
9. Right click on the **Helpers.Tests.Unit** project and add a new class named **RsapiHelperTests.cs**.
10. Overwrite the default class structure with the Unit Test structure we will be using.
11. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\1_Structure.txt"** file replacing the entire **RsapiHelperTests.cs** class.



12. Add the RsapiHelper class reference which we will be our System under Test (SUT) for which we are writing unit tests.
13. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\2_System_Under_Test.txt"** file at the beginning of the **RsapiHelperTests.cs** class.

© Relativity. All rights reserved.

```
[TestFixture]
[Description("Rsapi Helper Tests")]
public class RsapiHelperTests
{
    public RsapiHelper Sut { get; set; }
```

14. Next add the Mocks for various RDO repositories which are used in the Relativity application.
15. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\3_Mocks.txt"** file at the beginning of the **RsapiHelperTests.cs** class.

```
[TestFixture]
[Description("Rsapi Helper Tests")]
public class RsapiHelperTests
{
    public RsapiHelper Sut { get; set; }
    public Mock<IGenericRepository<RDO>> MockRdoRepository { get; set; }
    public Mock<IGenericRepository<Choice>> MockChoiceRepository { get; set; }
    public Mock<IGenericRepository<Workspace>> MockWorkspaceRepository { get; set; }
    public Mock<IGenericRepository<User>> MockUserRepository { get; set; }
    public Mock<IGenericRepository<Group>> MockGroupRepository { get; set; }
```

16. Next add the SetUp and TearDown methods which run once before and after all the tests execution.
17. The SetUp method is utilized to inititalize any data needed by tests.
18. The TearDown method is utilized to clean up the data used by the tests.
19. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\4_SetUp_And_TearDown.txt"** file under the **SetUp and TearDown** region in the **RsapiHelperTests.cs** class.

```
#region SetUp and TearDown

[SetUp]
public void SetUp()
{
    MockRdoRepository = new Mock<IGenericRepository<RDO>>();
    MockChoiceRepository = new Mock<IGenericRepository<Choice>>();
    MockWorkspaceRepository = new Mock<IGenericRepository<Workspace>>();
    MockUserRepository = new Mock<IGenericRepository<User>>();
    MockGroupRepository = new Mock<IGenericRepository<Group>>();
    APIOptions rsapiApiOptions = new APIOptions
    {
        WorkspaceID = -1
    };

    Sut = new RsapiHelper(
        rsapiApiOptions: rsapiApiOptions,
        rdoRepository: MockRdoRepository.Object,
        choiceRepository: MockChoiceRepository.Object,
        workspaceRepository: MockWorkspaceRepository.Object,
        userRepository: MockUserRepository.Object,
        groupRepository: MockGroupRepository.Object);
}

[TearDown]
public void TearDown()
{
    MockRdoRepository = null;
    MockChoiceRepository = null;
    MockWorkspaceRepository = null;
    MockUserRepository = null;
    MockGroupRepository = null;
    Sut = null;
}

#endregion
```


© Relativity. All rights reserved.

20. Next add the Test constants used in the Tests.
21. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\5_Test_Constants.txt"** file under the **TestConstants** region in the **RsapiHelperTests.cs** class.

```
#region TestConstants

private const int TestWorkspaceArtifactId = 123;
private const string TestStatus = "New";

#endregion
```

22. Next add the Mock Stubs for an RSAPI call used in the Tests. Here we will be stubbing out the Query method on the RDO repository in RSAPI.
23. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\6_Mock_Stub_Golden_Flow.txt"** file under the **Mocks** region in the **RsapiHelperTests.cs** class.

```
#region Mocks

private void Mock_RdoRepository_Query_Works(int rdoCount)
{
    List<Result<RDO>> results = new List<Result<RDO>>();
    for (int i = 1; i <= rdoCount; i++)
    {
        Result<RDO> newResult = new Result<RDO>
        {
            Artifact = new RDO(i),
            Message = string.Empty,
            Success = true
        };
        results.Add(newResult);
    }
    QueryResultSet<RDO> rdoQueryResultSet = new QueryResultSet<RDO>
    {
        Success = true,
        Results = results
    };

    MockRdoRepository
        .Setup(x => x.Query(It.IsAny<Query<RDO>>(), It.IsAny<int>()))
        .Returns(rdoQueryResultSet);
}
```

24. Next add the Verify method for the previously added Mock Stubs. Use this method we can verify if our stub was called during our test execution.
25. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\7_Mock_Stub_Golden_Flow_Verify.txt"** file under the **Verify** region in the **RsapiHelperTests.cs** class.

```
#region Verify

private void Verify_RdoRepository_Query_Works_Was_Called(int timesCalled)
{
    MockRdoRepository
        .Verify(x => x.Query(It.IsAny<Query<RDO>>(), It.IsAny<int>())
        , Times.Exactly(timesCalled));
}

#endregion
```

26. Next add the Golden flow test.

© Relativity. All rights reserved.

27. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\8_UnitTest_Golden_Flow.txt"** file under the **Tests** region in the **RsapiHelperTests.cs** class.

```
#region Tests

[Test]
public void RetrieveJobsInWorkspaceWithStatus_GoldenFlow()
{
    //Arrange
    int rdoCount = 5;
    Mock_RdoRepository_Query_Works(rdoCount);

    //Act
    List<int> jobsList = Sut.RetrieveJobsInWorkspaceWithStatus(TestWorkspaceArtifactId, TestStatus);

    //Assert
    Verify_RdoRepository_Query_Works_Was_Called(1);
    Assert.That(jobsList.Count, Is.EqualTo(rdoCount));
}
```

28. Next add the Mock Stubs for an RSAPI call to simulate an exception in the RSAPI call.
29. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\9_Mock_Stub_Rsapi_Fails.txt"** file under the **Mocks** region in the **RsapiHelperTests.cs** class.

```
private void Mock_RdoRepository_Query_Throws_Exception()
{
    MockRdoRepository
        .Setup(x => x.Query(It.IsAny<Query<RDO>>(), It.IsAny<int>()))
        .Throws<Exception>();
}

#endregion
```

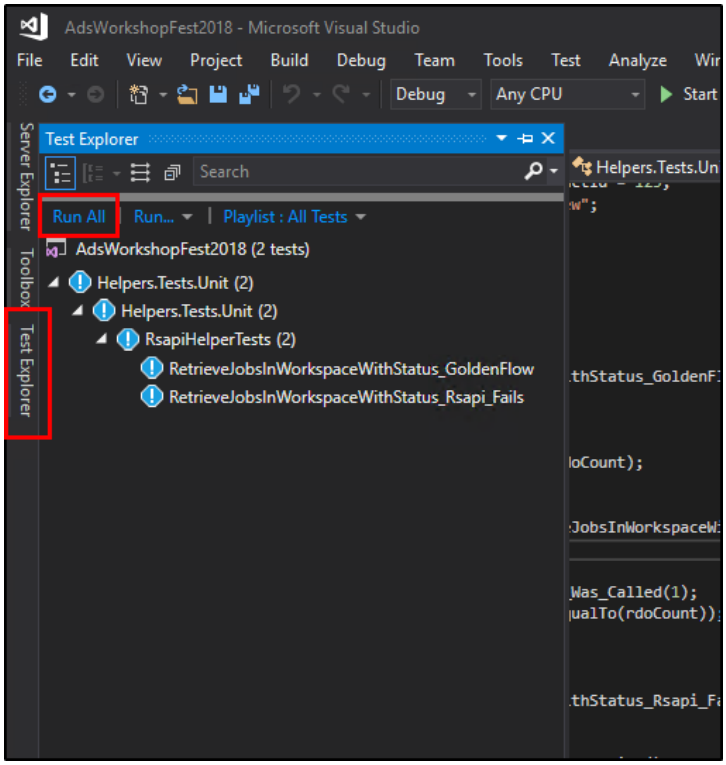
30. Next add the RSAPI Failure test.
31. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\10_UnitTest_Rsapi_Fails.txt"** file under the **Tests** region in the **RsapiHelperTests.cs** class.

```
[Test]
public void RetrieveJobsInWorkspaceWithStatus_Rsapi_Fails()
{
    //Arrange
    Mock_RdoRepository_Query_Throws_Exception();

    //Act
    Exception exception = Assert.Throws<Exception>(() => Sut.RetrieveJobsInWorkspaceWithStatus(TestWorkspaceArtifactId, TestStatus));

    //Assert
    StringAssert.Contains(Constants.ErrorMessages.QUERY_APPLICATION_JOBS_ERROR, exception.ToString());
    Verify_RdoRepository_Query_Works_Was_Called(1);
}

#endregion
```

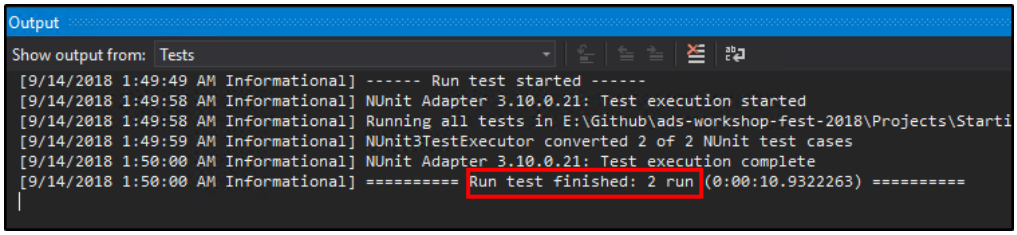

© Relativity. All rights reserved.

5.2 Running Unit Tests

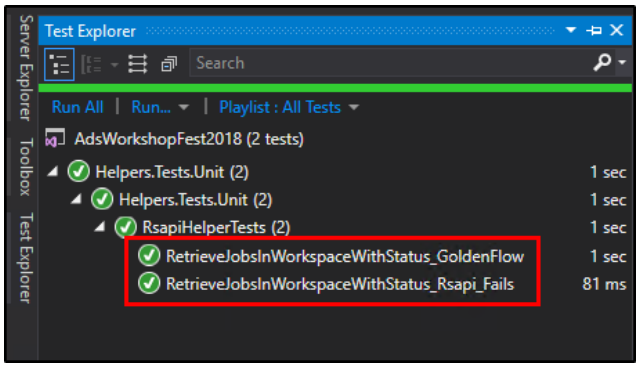
1. Build the solution to successfully build all the projects.
2. Open the **Test Explorer** by click on the **Test Explorer** tab on the left side bar of Visual Studio.
3. Click on the **Run All** link to run the 2 unit tests we just created.



4. On successful run, click on the **Test Explorer** to see the status of the tests. Both the tests should be green.



© Relativity. All rights reserved.

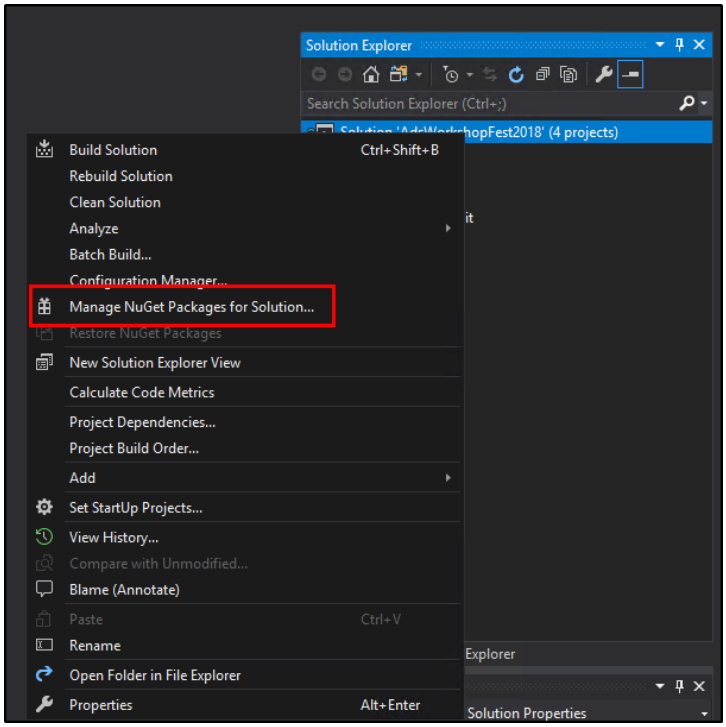


© Relativity. All rights reserved.

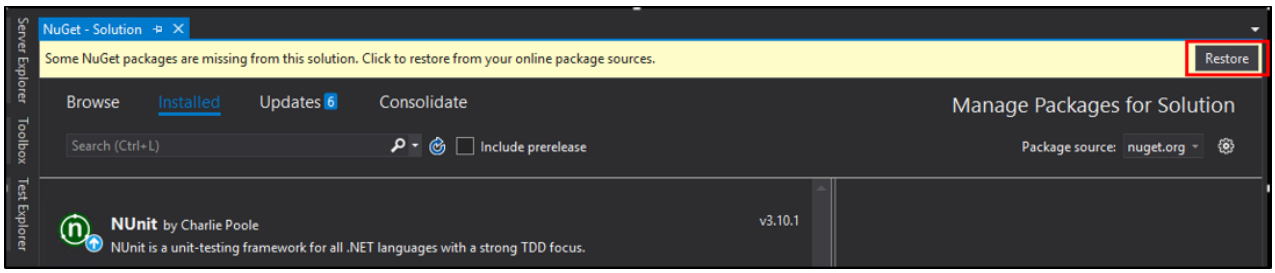
6 Integration Tests

6.1 Writing Integration Tests

1. Navigate to the **E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project** folder and open **AdsWorkshopFest2018.sln** file.
2. In this solution, you will find 4 projects.
    a. Agents
    b. Agents.Tests.Integration
    c. EventHandlers
    d. Helpers
3. In this section we will be working in the **Agents.Tests.Integration** project.
4. Right click on the solution and select **Manage NuGet Packages for Solution** option.



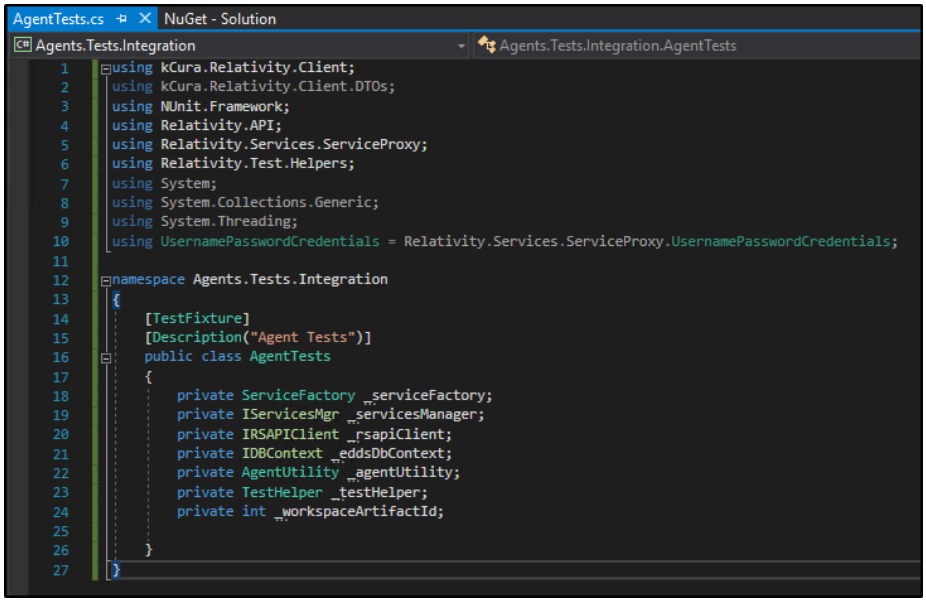
5. Next click the **Restore** button.



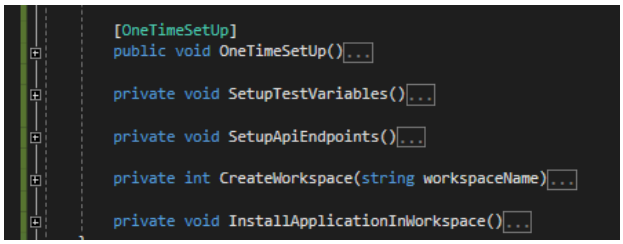
6. Now all the NuGet packages in the solution should be restored.
7. Right click on the **Agents.Tests.Integration** project and add a new class named **AgentsTests.cs**.

© Relativity. All rights reserved.

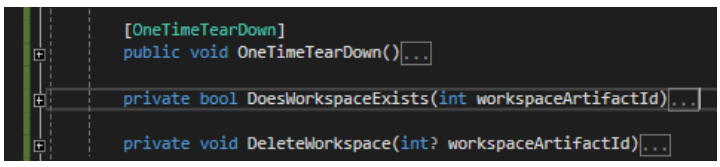
8. Overwrite the default class structure with the Integration Test structure we will be using.
9. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\1_Structure.txt"** file replacing the entire **AgentsTests.cs** class.



10. Add the **OneTimeSetUp** method and its dependent methods to the **AgentsTests** class.
11. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\2_OneTimeSetUp.txt"** to the **AgentsTests** class.



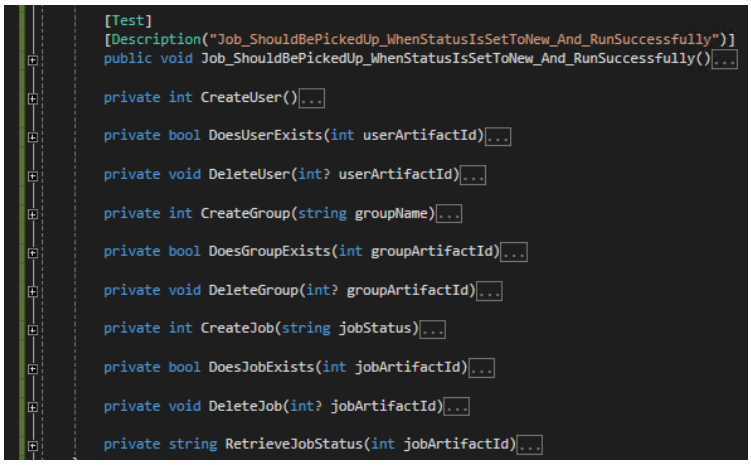
12. Add the **OneTimeTearDown** method and its dependent methods to the **AgentsTests** class.
13. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\3_OneTimeTearDown.txt"** to the **AgentsTests** class.



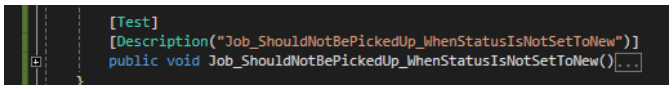
14. Add the **Job_ShouldBePickedUp_WhenStatusIsSetToNew_And_RunSuccessfully** integration test method and its dependent methods to the **AgentsTests** class.

© Relativity. All rights reserved.

15. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\4_Job_ShouldBePickedUp_WhenStatusIsSetToNew_And_RunSuccessfully.txt"** to the **AgentsTests** class.



16. Add the **Job_ShouldNotBePickedUp_WhenStatusIsNotSetToNew** integration test method and its dependent methods to the **AgentsTests** class.
17. Copy and paste code from **"E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\5_Job_ShouldNotBePickedUp_WhenStatusIsNotSetToNew.txt"** to the **AgentsTests** class.



© Relativity. All rights reserved.

6.2 Running Integration Tests

1. Open **TestConstants.cs** file in **Agents.Tests.Integration** project.
2. Replace the **192.168.137.95** IP address with the DevVM IP address.

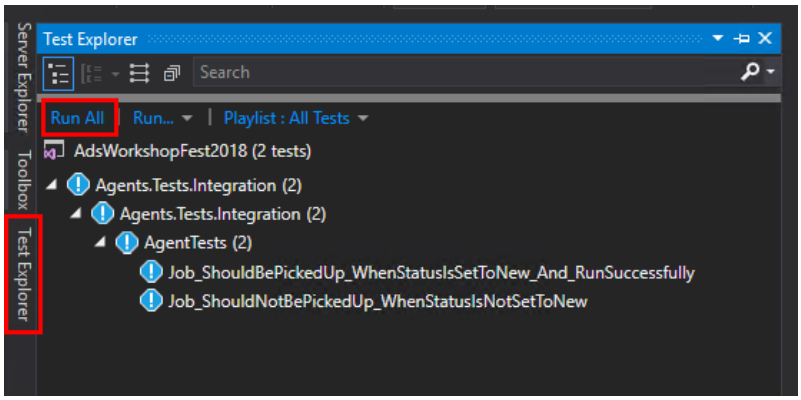
```
public class TestConstants
{
    17 references | Chandra Kaushik Alimeti, 17 days ago | 1 author, 1 change
    public class InstanceDetails
    {
        public const string INSTANCE_NAME = "172.27.145.78";
        public const string PROTOCOL = "http";
        public const string RELATIVITY_ADMIN_USERNAME = "relativity.admin@relativity.com";
        public const string RELATIVITY_ADMIN_PASSWORD = "Test1234!";
        public const string TEST_WORKSPACE_TEMPLATE_NAME = "Relativity Starter Template";
        public const string SQL_SERVER_NAME = "172.27.145.78";
        public const string SQL_DATABASE_NAME = "EDDS";
        public const string SQL_USERNAME = "eddsdbo";
        public const string SQL_PASSWORD = "Test1234!";
        public static readonly Uri RelativityServicesUri = new Uri($"{PROTOCOL}://{INSTANCE_NAME}/Relativity.Services");
        public static readonly Uri RelativityRestUri = new Uri($"{PROTOCOL}://{INSTANCE_NAME}/Relativity.Rest/Api");
    }

    public const string AGENT_OFF_HOURS_SECTION_NAME = "kCura.EDDS.Agents";
    public const string AGENT_OFF_HOUR_START_TIME_NAME = "AgentOffHourStartTime";
    public const string AGENT_OFF_HOUR_END_TIME_NAME = "AgentOffHourEndTime";
    public const bool ENABLE_AGENT = true;
    public const int AGENT_INTERVAL = 10;
    public const Agent.LoggingLevelEnum AGENT_LOGGING_LEVEL = Agent.LoggingLevelEnum.All;
    public const int WORKSPACE_CREATION_RETRY = 3;
    public static readonly string ApplicationRapFilePath = @"S:\SourceCode\GitHub\ads-workshop-fest-2018\Projects\IntegrationTests\RAP
     \ADS_Workshop_Fest_2018.rap";
    public static readonly string WorkspaceName = "ADS" + "-" + Guid.NewGuid();
}
}
```

3. Open **App.config** file in **Agents.Tests.Integration** project.
4. Replace the **192.168.137.95** IP address with the DevVM IP address.

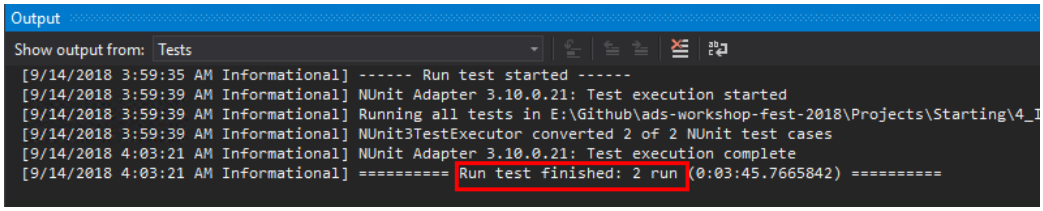
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings file="appSettings.config">
    <add key="WorkspaceID" value="" />
    <add key="RSAPIServerAddress" value="172.27.145.78" />
    <add key="RESTServerAddress" value="172.27.145.78" />
    <add key="AdminUsername" value="relativity.admin@relativity.com" />
    <add key="AdminPassword" value="Test1234!" />
    <add key="SQLServerAddress" value="172.27.145.78" />
    <add key="SQLUsername" value="eddsdbo" />
    <add key="SQLPassword" value="Test1234!" />
    <add key="TestWorkspaceName" value="" />
    <add key="ServerBindingType" value="http" />
    <add key="TestWorkspaceTemplateName" value="Relativity Starter Template" />
    <add key="ClientSettingsProvider.ServiceUri" value="" />
  </appSettings>
</configuration>
```

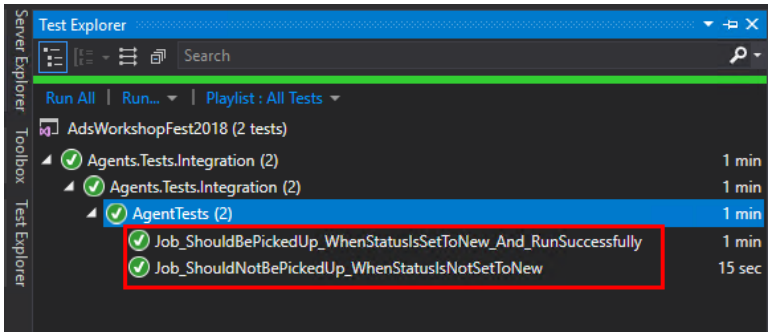
5. Build the solution to successfully build all the projects.
6. Open the **Test Explorer** by click on the **Test Explorer** tab on the left side bar of Visual Studio.
7. Click on the **Run All** link to run the 2 unit tests we just created.

© Relativity. All rights reserved.



8. On successful run, click on the **Test Explorer** to see the status of the tests. Both the tests should be green.

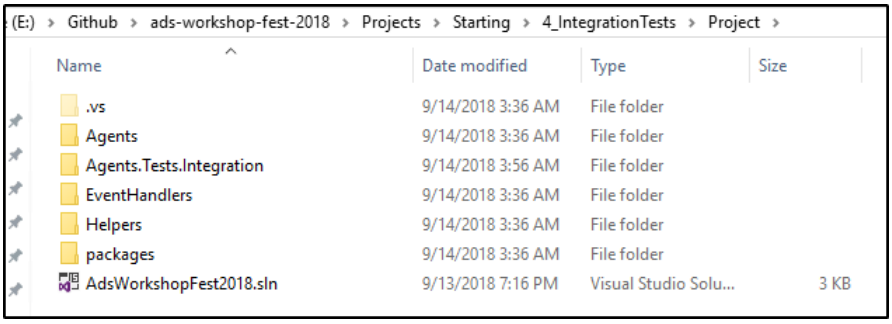


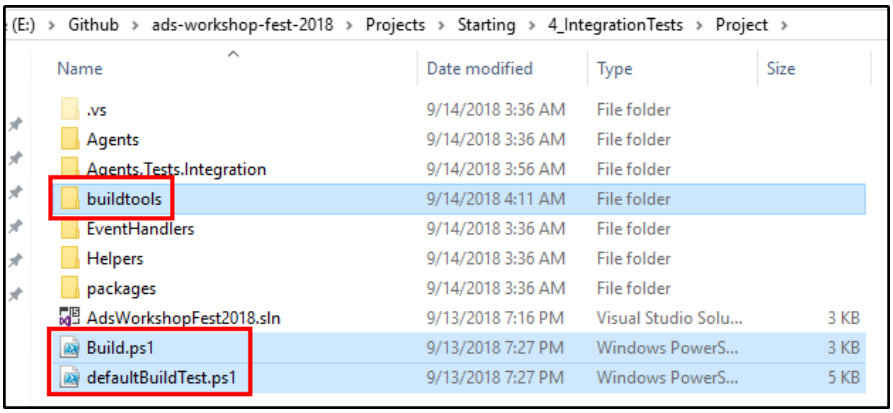


© Relativity. All rights reserved.

6.2 Running Integration Tests as CI

1. Open the **E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\CI_Scripts** folder
2. Here you will find 3 files and 1 folder which we will be using to run Integration Tests as a CI (Continuous Integration) process.
3. Copy all the files and folders from the CI_Scripts folder to **E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project** folder.





4. Open the **Build.ps1** script and update the **NuGet_URL** to use the one which is compatible with Visual Studio 2017.

```
#set up variables
$BASE_DIR = Resolve-Path .
Write-Verbose "BASE_DIR resolves to: $BASE_DIR"
#VS 2017
#$NUGET_URL = "https://dist.nuget.org/win-x86-commandline/latest/nuget.exe"
#VS2015
$NUGET_URL = "https://dist.nuget.org/win-x86-commandline/v3.3.0/nuget.exe"
Write-Verbose "nuget URL: $NUGET_URL"
```

```
#set up variables
$BASE_DIR = Resolve-Path .
Write-Verbose "BASE_DIR resolves to: $BASE_DIR"
#VS 2017
$NUGET_URL = "https://dist.nuget.org/win-x86-commandline/latest/nuget.exe"
#VS2015
#$NUGET_URL = "https://dist.nuget.org/win-x86-commandline/v3.3.0/nuget.exe"
Write-Verbose "nuget URL: $NUGET_URL"
```


© Relativity. All rights reserved.

5. Open the **defaultBuildTest.ps1** script and update the **project paths.**
   a. Replace line 10 with the following code.

Code:
$solution = Join-Path $root "..\AdsWorkshopFest2018.sln"

Before:

```
10    $solution = Join-Path $root "..\RelativityAgent1\RelativityAgent.sln"
```

After:

```
10    $solution = Join-Path $root "..\AdsWorkshopFest2018.sln"
```

   b. Replace line 15 with the following code.

Code:
$testAssembly = $testAssembly = Join-Path $root
"..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll"

Before:

```
15    $testAssembly =  $testAssembly = Join-Path $root
      "..\RelativityAgent1\AgentNunitIntegrationTest\bin\Debug\AgentNunitIntegrationTest.dll"
```

After:

```
15    $testAssembly =  $testAssembly = Join-Path $root
      "..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll"
```

   c. Comment lines 64 – 66.

```
64  # task UnitTest -Alias Test -Depends TestInitialize -Description "Run NUnit unit tests" {
65      # exec { & $nunit_exe $solution --where "class=~/^.+\.UnitTests\..+$/"
        --result="$test_logs\UnitTests.xml;format=nunit2" } -errorMessage "Unit tests failed!"
66  # }
```

   d. Replace lines 69 - 74 with the following code.

Code:
$testDir = Join-Path $root "..\Agents.Tests.Integration"
$testDir = Join-Path $root "..\Agents.Tests.Integration"
$configSource = "..\Project\Agents.Tests.Integration\App.config"
Write-Host "configSource is : $configSource"
$configDestination = Join-Path $root
"..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll.config"
Write-Host "Test assembly : $testAssembly"
Copy-Item $configSource $configDestination -Verbose:$VerbosePreference

Before:

```
69    #$testDir = Join-Path $root "..\RelativityAgent1\AgentNunitIntegrationTest"
70  # $configSource = "..\RelativityAgent1\AgentNunitIntegrationTest\app.config"
71    #Write-Host "configSource is : $configSource"
72    #$configDestination = Join-Path $root
      "..\RelativityAgent1\AgentNunitIntegrationTest\bin\Debug\AgentNunitIntegrationTest.dll.config"
73    #Write-Host "Test assembly : $testAssembly"
74    #Copy-Item $configSource $configDestination -Verbose:$VerbosePreference
```


© Relativity. All rights reserved.

After:

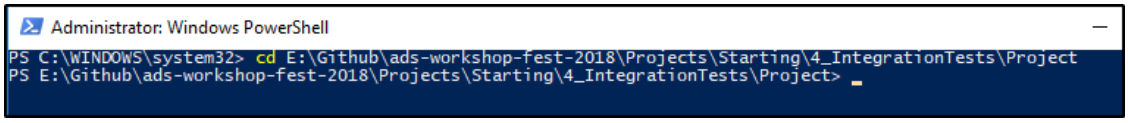
```
69      $testDir = Join-Path $root "..\Agents.Tests.Integration"
70      $configSource = "..\Project\Agents.Tests.Integration\App.config"
71      Write-Host "configSource is : $configSource"
72      $configDestination = Join-Path $root
        "..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll.config"
73      Write-Host "Test assembly : $testAssembly"
74      Copy-Item $configSource $configDestination -Verbose:$VerbosePreference
```

6. Open PowerShell and run the following command.

   Set-ExecutionPolicy Unrestricted

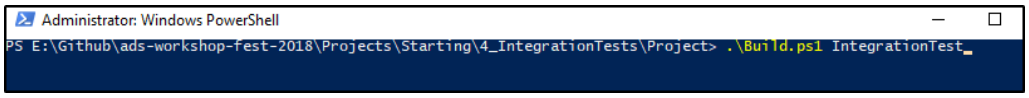
7. When prompted type **A** and press **Enter**.
8. Close the current PowerShell window and open a new PowerShell window.
9. Navigate to **"E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project"** folder by running the following command.

   cd E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project



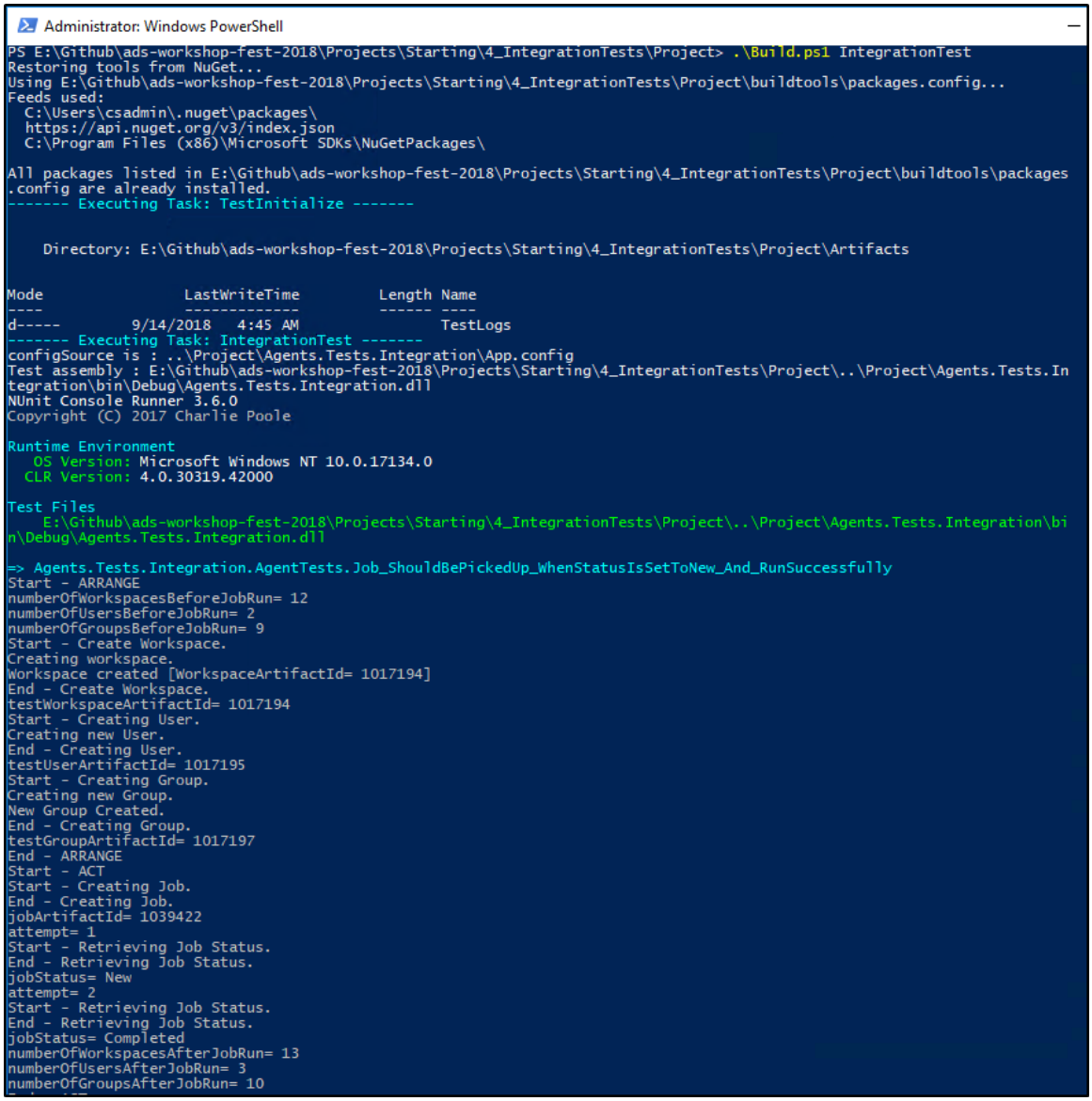
10. Run the following command to run the Integration Tests.

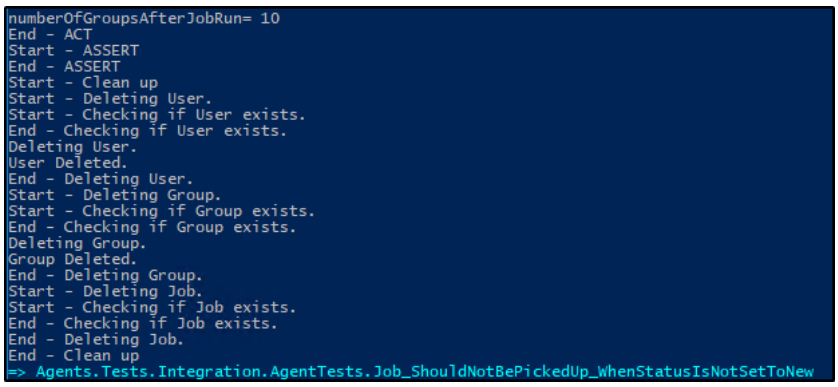
    .\Build.ps1 IntegrationTest



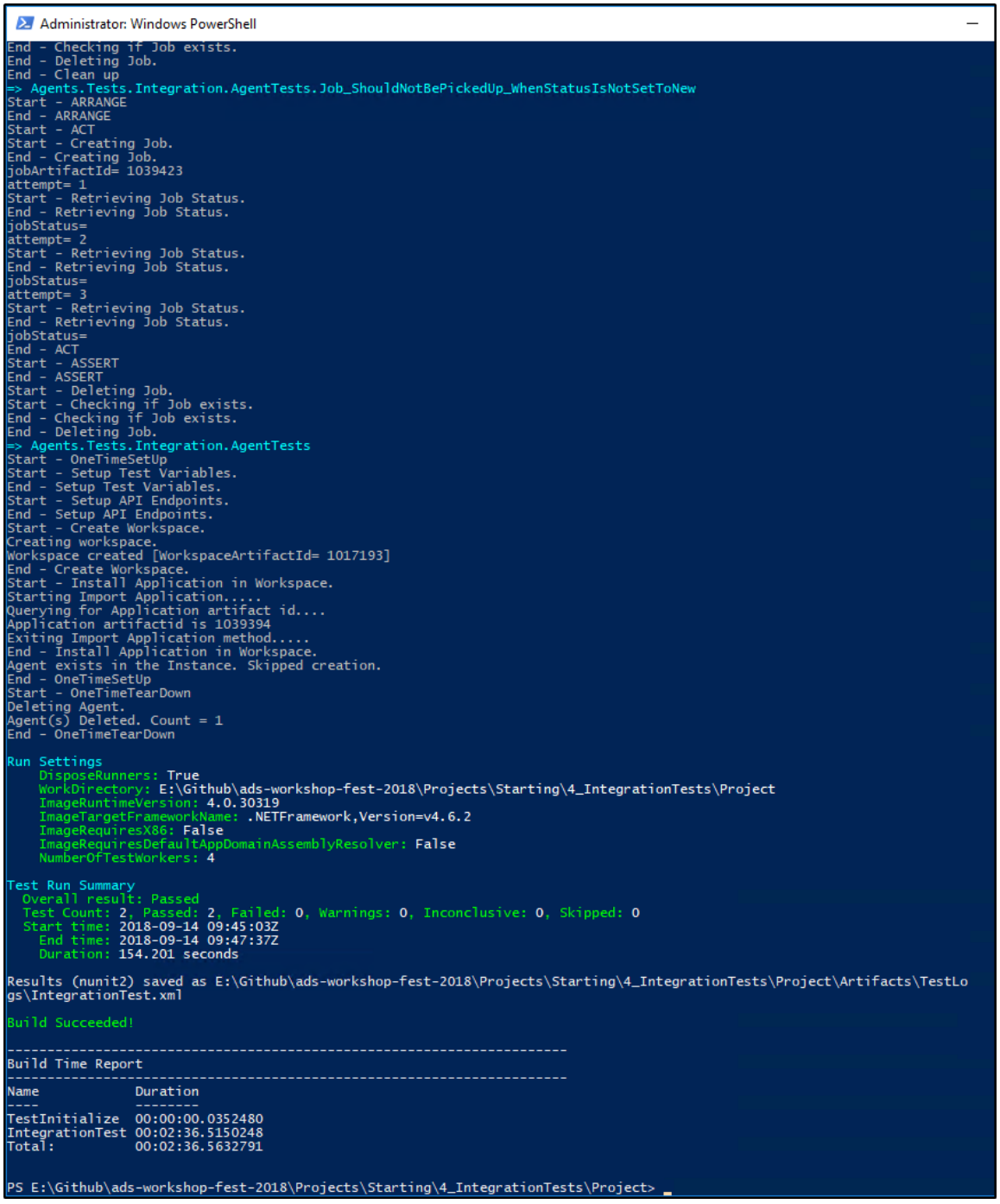
11. You will see the following output when the Integration tests are run successfully.

© Relativity. All rights reserved.





© Relativity. All rights reserved.

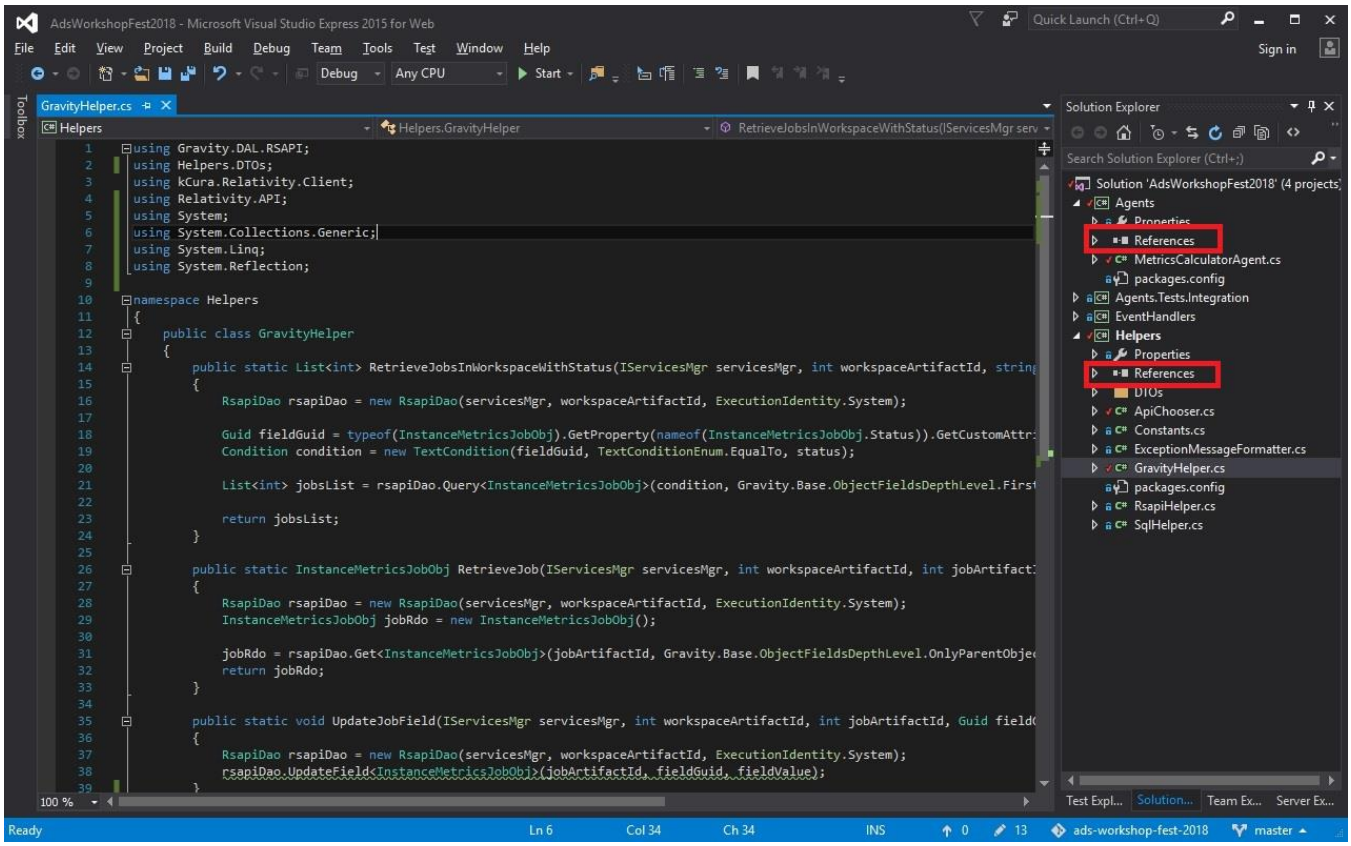


© Relativity. All rights reserved.

7 Gravity API

1 In this section, we will be using the Gravity open source API instead of RSAPI to read and update RDO values. You can read more about the Gravity API at this link: **https://github.com/relativitydev/Gravity**

2 To use the **Gravity** API, first go to Visual Studio and add Gravity.dll as new reference for **Helpers** project and **Agents** project. You can find Gravity.dll in **ads-workshop-fest-2018\DLLs\Gravity**



3 To use the **Gravity** API, first go to Visual Studio and create enum MetricsChoices under folder DTOs in Helpers project. This enum represents the Choices for Metrics field.

*Note: To create the enum right-click on the folder **DTOs** and select **Add -> New Item**. On the new window select **Visual C# Items -> Class**. Name the file MetricsChoices.cs.*

```
public enum MetricsChoices
{
    [RelativityObject("9715EB01-97F0-496A-9640-2494AD7CAA35")]
    Workspaces = 1,

    [RelativityObject("49BE6FCC-DB19-4BA3-A849-712DD2A72650")]
    Users = 2,

    [RelativityObject("E9BCE5CE-EC87-4A46-AB61-E8157DC5BA57")]
    Groups = 3
}
```


© Relativity. All rights reserved.

Code to add:

```
public enum MetricsChoices
{
        [RelativityObject("9715EB01-97F0-496A-9640-2494AD7CAA35")]
        Workspaces = 1,

        [RelativityObject("49BE6FCC-DB19-4BA3-A849-712DD2A72650")]
        Users = 2,

        [RelativityObject("E9BCE5CE-EC87-4A46-AB61-E8157DC5BA57")]
        Groups = 3
}
```

4 Then create class for the RDO with its fields as properties under folder DTOs in Helpers project.
*Note: To create the class right-click on the folder **DTOs** and select **Add -> New Item**. On the new window select **Visual C# Items -> Class**. Name the file InstanceMetricsJobObj.cs.*

```
[Serializable]
[RelativityObject("07FCE2E4-3318-4A00-9EF4-566FFCD7C198")]
public class InstanceMetricsJobObj : BaseDto
{
    [RelativityObjectField("7D1DFEDD-36A2-41A2-97D3-C1537DCD0598", RdoFieldType.FixedLengthText)]
    public override string Name { get; set; }

    [RelativityObjectField("065F1211-5A65-4DAC-AA26-EEED9007DCA9", RdoFieldType.LongText)]
    public string Status { get; set; }

    [RelativityObjectField("70401A4A-94CC-45BB-A6CA-808F6754F114", RdoFieldType.MultipleChoice)]
    public IList<MetricsChoices> Metrics { get; set; }

    [RelativityObjectField("8435115F-894F-43C3-978E-8E9CF42AB2DB", RdoFieldType.LongText)]
    public string WorkspacesCount { get; set; }

    [RelativityObjectField("45D186F4-A9BB-4427-A6DA-DAA4AD639024", RdoFieldType.LongText)]
    public string UsersCount { get; set; }

    [RelativityObjectField("9B56A63F-5F42-4CBD-BE53-761E8AD32CA0", RdoFieldType.LongText)]
    public string GroupsCount { get; set; }

    [RelativityObjectField("B9A22B34-0D87-4527-AD7F-B070AF4470AE", RdoFieldType.LongText)]
    public string Errors { get; set; }
}
```

Code to add:

```
using Gravity.Base;
using System;
using System.Collections.Generic;

namespace Helpers.DTOs
{
        [Serializable]
```


© Relativity. All rights reserved.

```
        [RelativityObject("07FCE2E4-3318-4A00-9EF4-566FFCD7C198")]
        public class InstanceMetricsJobObj : BaseDto
        {
                [RelativityObjectField("7D1DFEDD-36A2-41A2-97D3-C1537DCD0598",
RdoFieldType.FixedLengthText)]
                public override string Name { get; set; }

                [RelativityObjectField("065F1211-5A65-4DAC-AA26-EEED9007DCA9",
RdoFieldType.LongText)]
                public string Status { get; set; }

                [RelativityObjectField("70401A4A-94CC-45BB-A6CA-808F6754F114",
RdoFieldType.MultipleChoice)]
                public IList<MetricsChoices> Metrics { get; set; }

                [RelativityObjectField("8435115F-894F-43C3-978E-8E9CF42AB2DB",
RdoFieldType.LongText)]
                public string WorkspacesCount { get; set; }

                [RelativityObjectField("45D186F4-A9BB-4427-A6DA-DAA4AD639024",
RdoFieldType.LongText)]
                public string UsersCount { get; set; }

                [RelativityObjectField("9B56A63F-5F42-4CBD-BE53-761E8AD32CA0",
RdoFieldType.LongText)]
                public string GroupsCount { get; set; }

                [RelativityObjectField("B9A22B34-0D87-4527-AD7F-B070AF4470AE",
RdoFieldType.LongText)]
                public string Errors { get; set; }
        }
}
```

5 To use Gravity, you have to create RsapiDao. You will see this for the methods below.

```
RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId, ExecutionIdentity.System);
```

6 When you have created the RsapiDao, you can use it for various CRUDQ operations.

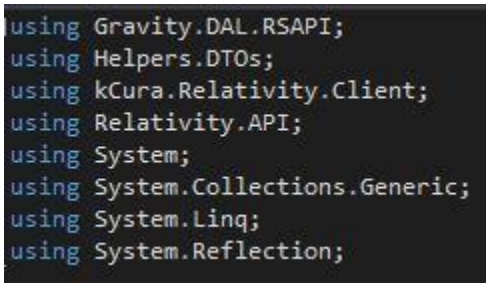
```
List<int> jobsList = rsapiDao.Query<InstanceMetricsJobObj>(condition, Gravity.Base.ObjectFieldsDepthLevel.FirstLevelOnly).Select(x => x.ArtifactId).ToList();
```

7 If you want to retrieve the GUID for a field, you can do that by using Reflection

```
Guid fieldGuid = typeof(InstanceMetricsJobObj).GetProperty(nameof(InstanceMetricsJobObj.Status)).GetCustomAttribute<RelativityObjectFieldAttribute>().FieldGuid;
```


© Relativity. All rights reserved.

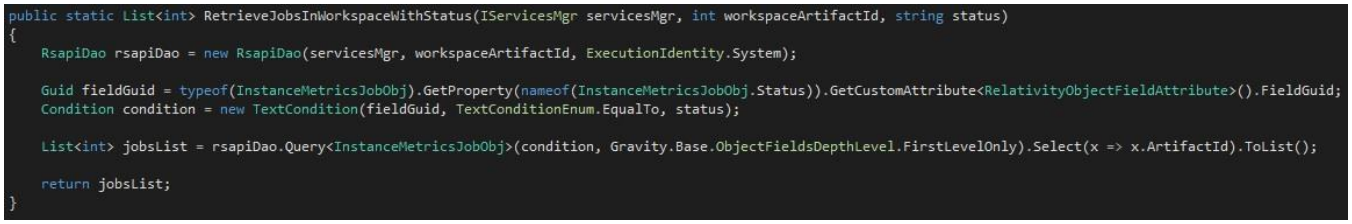
8 Open the **GravityHelper.cs** file in **Helpers** project and add the following **using** statements



Code to add:

```
using Gravity.DAL.RSAPI;
using Helpers.DTOs;
using kCura.Relativity.Client;
using Relativity.API;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
```

9 Next declare method to retrieve all jobs in a workspace for specific status



Code to add:

```
public static List<int> RetrieveJobsInWorkspaceWithStatus(IServicesMgr servicesMgr, int workspaceArtifactId, string status)
        {
                RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId, ExecutionIdentity.System);

                Guid fieldGuid = typeof(InstanceMetricsJobObj).GetProperty(nameof(InstanceMetricsJobObj.Status)).GetCustomAttribute<RelativityObjectFieldAttribute>().FieldGuid;
                Condition condition = new TextCondition(fieldGuid, TextConditionEnum.EqualTo, status);

                List<int> jobsList = rsapiDao.Query<InstanceMetricsJobObj>(condition, Gravity.Base.ObjectFieldsDepthLevel.FirstLevelOnly).Select(x => x.ArtifactId).ToList();
```


© Relativity. All rights reserved.

```
        return jobsList;
    }
```

10 In the same **GravitiyHelper.cs**, declare method to retrieve one specific job

```
public static InstanceMetricsJobObj RetrieveJob(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
    RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId, ExecutionIdentity.System);
    InstanceMetricsJobObj jobRdo = new InstanceMetricsJobObj();

    jobRdo = rsapiDao.Get<InstanceMetricsJobObj>(jobArtifactId, Gravity.Base.ObjectFieldsDepthLevel.OnlyParentObject);
    return jobRdo;
}
```

Code to add:

```
public static InstanceMetricsJobObj RetrieveJob(IServicesMgr servicesMgr, int workspaceArtifactId,
int jobArtifactId)
        {
            RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId,
ExecutionIdentity.System);
            InstanceMetricsJobObj jobRdo = new InstanceMetricsJobObj();

            jobRdo = rsapiDao.Get<InstanceMetricsJobObj>(jobArtifactId,
Gravity.Base.ObjectFieldsDepthLevel.OnlyParentObject);
            return jobRdo;
        }
```

11 In the same **GravityHelper.cs**, declare method to update specific field

```
public static InstanceMetricsJobObj RetrieveJob(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
    RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId, ExecutionIdentity.System);
    InstanceMetricsJobObj jobRdo = new InstanceMetricsJobObj();

    jobRdo = rsapiDao.Get<InstanceMetricsJobObj>(jobArtifactId, Gravity.Base.ObjectFieldsDepthLevel.OnlyParentObject);
    return jobRdo;
}
```

Code to add:

```
public static void UpdateJobField(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId,
Guid fieldGuid, object fieldValue)
        {
            RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId,
ExecutionIdentity.System);
            rsapiDao.UpdateField<InstanceMetricsJobObj>(jobArtifactId, fieldGuid, fieldValue);
        }
```


© Relativity. All rights reserved.

12 Open the **ApiChooser.cs** file in **Helpers** project and invoke the method **GravityHelper.RetrieveJobsInWorkspaceWithStatus** (declared in step 4) in method public **List<int> RetrieveJobsInWorkspaceWithStatus(IServicesMgr servicesMgr, int workspaceArtifactId, string status)**

```
public List<int> RetrieveJobsInWorkspaceWithStatus(IServicesMgr servicesMgr, int workspaceArtifactId, string status)
{
    List<int> jobsList;

    if (_apiType.Equals(Constants.ApiType.Rsapi))
    {
        jobsList = RsapiHelper.RetrieveJobsInWorkspaceWithStatus(servicesMgr, workspaceArtifactId, status);
    }
    else if (_apiType.Equals(Constants.ApiType.Gravity))
    {
        jobsList = GravityHelper.RetrieveJobsInWorkspaceWithStatus(servicesMgr, workspaceArtifactId, status);
    }
    else
    {
        throw new Exception(Constants.ErrorMessages.INVALID_API_TYPE_ERROR);
    }

    return jobsList;
}
```

Code to comment:

jobsList = new List<int>();

Code to add:

jobsList = GravityHelper.RetrieveJobsInWorkspaceWithStatus(servicesMgr, workspaceArtifactId, status);

13 Next, create a method to invoke **GravityHelper.RetrieveJob**

```
public InstanceMetricsJobObj RetrieveJobWithGravity(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
    InstanceMetricsJobObj jobRdo = null;

    jobRdo = GravityHelper.RetrieveJob(servicesMgr, workspaceArtifactId, jobArtifactId);

    return jobRdo;
}
```

Code to add:

public InstanceMetricsJobObj RetrieveJobWithGravity(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
InstanceMetricsJobObj jobRdo = null;

© Relativity. All rights reserved.

```
        jobRdo = GravityHelper.RetrieveJob(servicesMgr, workspaceArtifactId, jobArtifactId);

        return jobRdo;
    }
```

14 Now invoke **GravityHelper.UpdateJobField** in method **public void UpdateJobField(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId, Guid fieldGuid, object fieldValue)**

```
public void UpdateJobField(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId, Guid fieldGuid, object fieldValue)
{
    if (_apiType.Equals(Constants.ApiType.Rsapi))
    {
        RsapiHelper.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, fieldGuid, fieldValue);
    }
    else if (_apiType.Equals(Constants.ApiType.Gravity))
    {
        GravityHelper.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, fieldGuid, fieldValue);
    }
    else
    {
        throw new Exception(Constants.ErrorMessages.INVALID_API_TYPE_ERROR);
    }
}
```

Code to add:

GravityHelper.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, fieldGuid, fieldValue);

15 Now you are ready to use the ApiChooser methods in the Agents project. Go to Agents project and open **MetricsCalculatorAgent.cs**. Create overload method **ProcessAllMetrics(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId, InstanceMetricsJobObj jobRdo)**.

```
private void ProcessAllMetrics(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId, InstanceMetricsJobObj jobRdo)
{
    try
    {
        foreach (MetricsChoices metric in jobRdo.Metrics)
        {
            Guid metricGuid = metric.GetRelativityObjectAttributeGuidValue();

            ProcessSingleMetric(servicesMgr, workspaceArtifactId, jobArtifactId, metricGuid);
        }
    }
    catch (Exception ex)
    {
        throw new Exception(Constants.ErrorMessages.PROCESS_ALL_JOB_METRICS_ERROR, ex);
    }
}
```

Code to add:

```
private void ProcessAllMetrics(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId,
InstanceMetricsJobObj jobRdo)
        {
                try
                {
                        foreach (MetricsChoices metric in jobRdo.Metrics)
```


© Relativity. All rights reserved.

```
                {
                        Guid metricGuid = metric.GetRelativityObjectAttributeGuidValue();

                        ProcessSingleMetric(servicesMgr, workspaceArtifactId, jobArtifactId,
metricGuid);
                }
            }
            catch (Exception ex)
            {
                throw new
Exception(Constants.ErrorMessages.PROCESS_ALL_JOB_METRICS_ERROR, ex);
            }
        }
```

16 Use RetrieveJobWithGravity to retrieve and process the job in **private void ProcessJob(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)**

```
private void ProcessJob(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
    try
    {
        //Update job status to In Progress
        _apiChooser.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, Constants.Guids.Fields.InstanceMetricsJob.Status_LongText, Constants.JobStatus.IN_PROGRESS);

        //Update job metrics
        if(selectedApiType == Helpers.Constants.ApiType.Rsapi)
        {
            RDO jobRdo = _apiChooser.RetrieveJob(servicesMgr, workspaceArtifactId, jobArtifactId);
            RaiseMessage("Calculating metrics for the job", 10);
            ProcessAllMetrics(servicesMgr, workspaceArtifactId, jobArtifactId, jobRdo);
            RaiseMessage("Calculated metrics for the job", 10);
        }
        else if(selectedApiType == Helpers.Constants.ApiType.Gravity)
        {
            InstanceMetricsJobObj jobRdo = _apiChooser.RetrieveJobWithGravity(servicesMgr, workspaceArtifactId, jobArtifactId);
            RaiseMessage("Calculating metrics for the job", 10);
            ProcessAllMetrics(servicesMgr, workspaceArtifactId, jobArtifactId, jobRdo);
            RaiseMessage("Calculated metrics for the job", 10);
        }

        //Update job status to Completed
        _apiChooser.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, Constants.Guids.Fields.InstanceMetricsJob.Status_LongText, Constants.JobStatus.COMPLETED);
    }
    catch (Exception ex)
    {
        //Update job status to Error
        string errorMessage = ExceptionMessageFormatter.GetInnerMostExceptionMessage(ex);
        _apiChooser.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, Constants.Guids.Fields.InstanceMetricsJob.Status_LongText, Constants.JobStatus.COMPLETED);
        _apiChooser.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, Constants.Guids.Fields.InstanceMetricsJob.Errors_LongText, errorMessage);
    }
}
```

Code to add:

InstanceMetricsJobObj jobRdo = _apiChooser.RetrieveJobWithGravity(servicesMgr, workspaceArtifactId, jobArtifactId);

RaiseMessage("Calculating metrics for the job", 10);

ProcessAllMetrics(servicesMgr, workspaceArtifactId, jobArtifactId, jobRdo);

RaiseMessage("Calculated metrics for the job", 10);

© Relativity. All rights reserved.

17 To switch from RSAPI to Gravity change the field **private static readonly Constants.ApiType selectedApiType** in **MetricsCalculatorAgent.cs**

```
[kCura.Agent.CustomAttributes.Name(Constants.Names.AGENT_INSTANCE_METRICS_CALCULATOR)]
[System.Runtime.InteropServices.Guid("20530D16-D825-4FA5-9A7E-6760579EB07B")]
public class MetricsCalculatorAgent : kCura.Agent.AgentBase
{
    private IAPILog _logger;

    public override string Name => Constants.Names.AGENT_INSTANCE_METRICS_CALCULATOR;
    private static readonly Constants.ApiType selectedApiType = Constants.ApiType.Gravity;
    private readonly ApiChooser _apiChooser = new ApiChooser(selectedApiType);

    public override void Execute()
    {
        RaiseMessage("Enter Agent", 10);

        try
        {
            _logger = Helper.GetLoggerFactory().GetLogger();
            IServicesMgr servicesMgr = Helper.GetServicesManager();
            IDBContext eddsDbContext = Helper.GetDBContext(-1);
```


© Relativity. All rights reserved.