Microsoft

Practical Exercises: Application Monitoring and Feedback Loops

Prerequisites

- Visual Studio 2017 Enterprise.
- Download the Parts Unlimited project from GitHub.
- Azure Subscription.
- VSTS project.

Scenario 1: APM with Application Insights

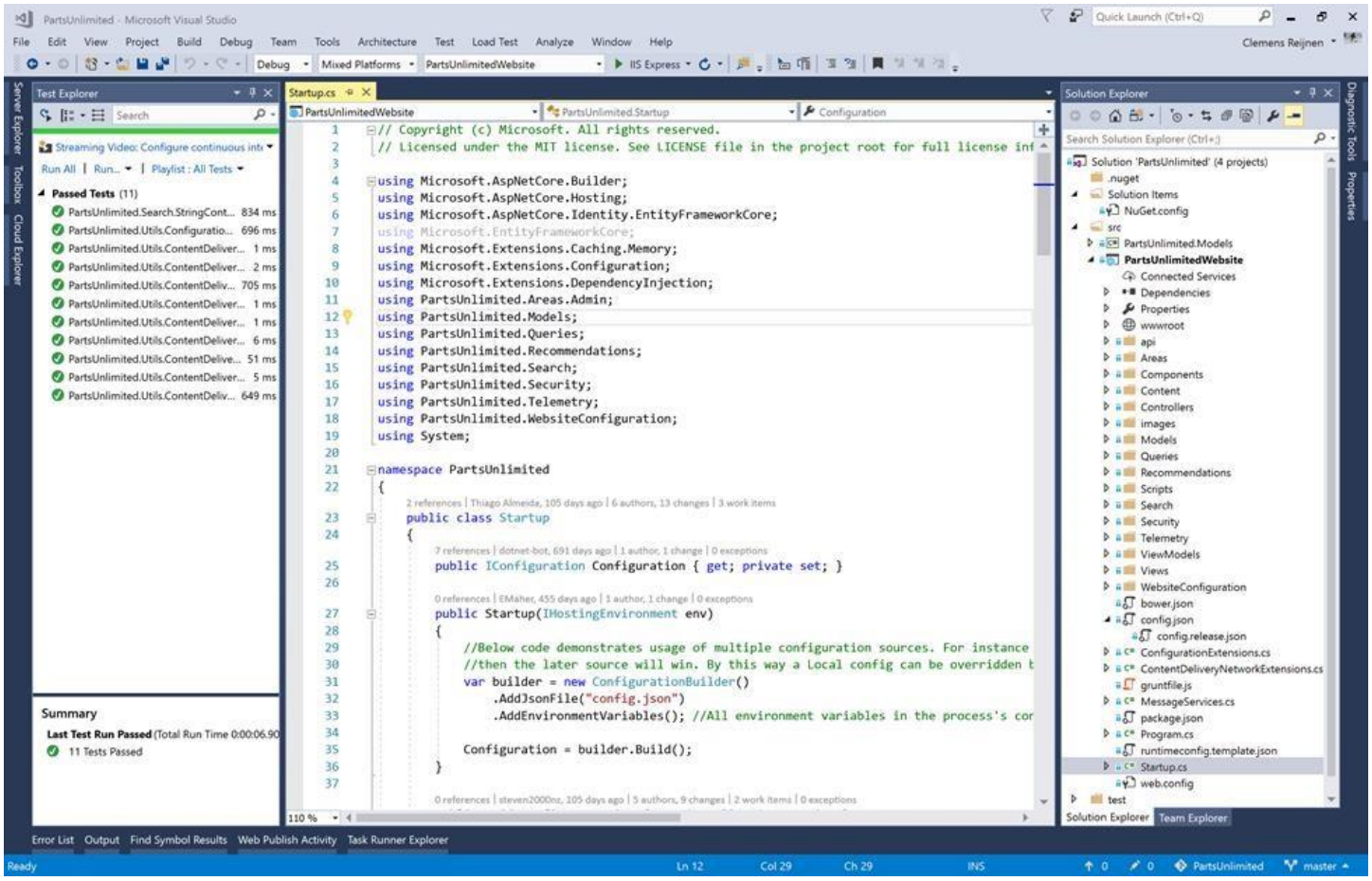
To maximize the availability and time-to-recover of the PartsUnlimited website, the operations team wants to track all requests performed on the website. The good and failing requests for the server and depended calls should be reported.

Implement and configure, as a developer on the PartsUnlimited DevOps team, the monitoring and reporting of telemetry data.

Lab 1: Add Application Insights Telemetry to the Parts Unlimited Website

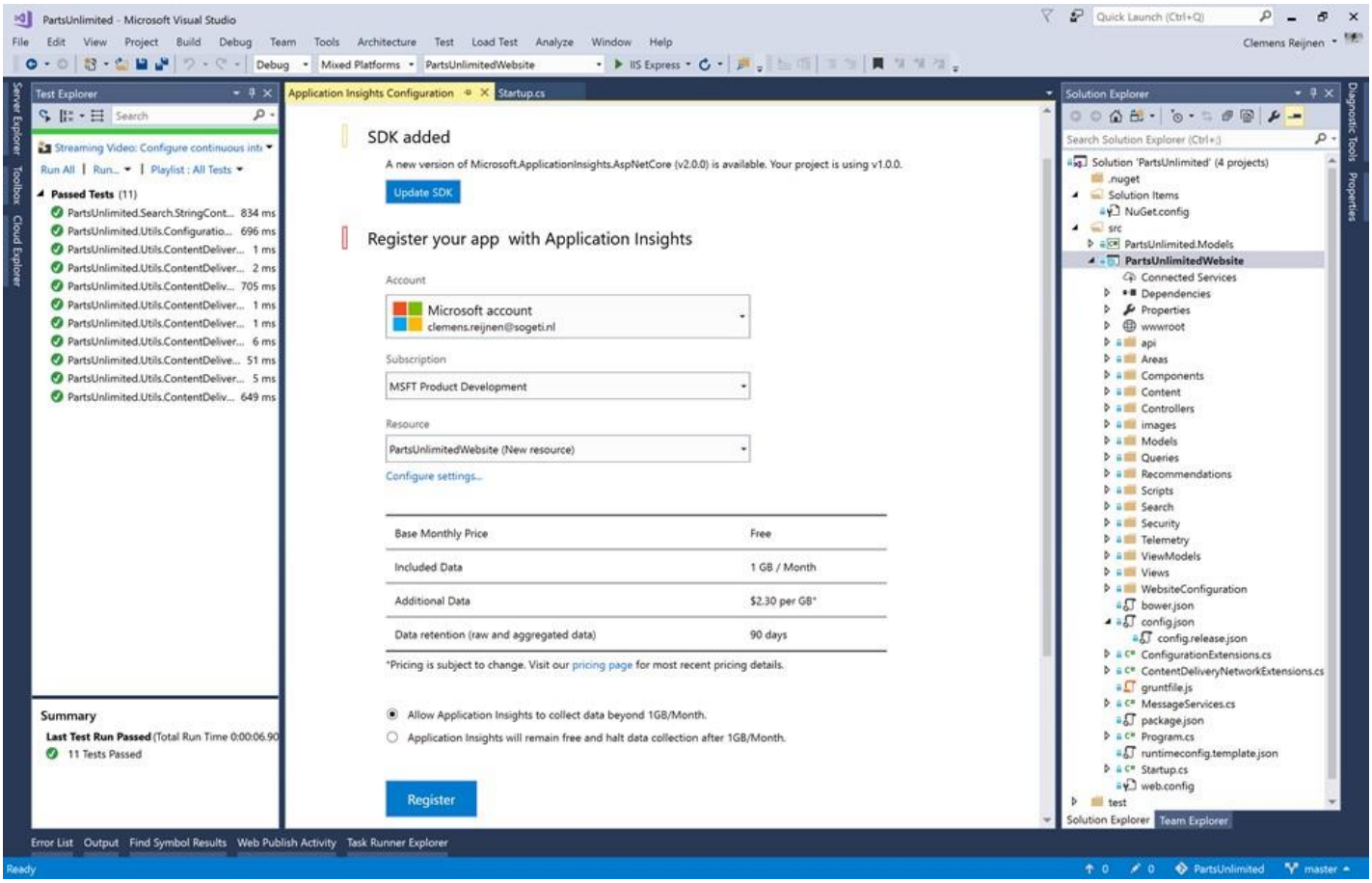
1. Open the PartsUnlimited solution within Visual Studio. Notes:
    - VS2017 may need to migrate the project files.
    - Note: npm and grunt must be installed.
    - **Note**: set the *config.json* and *appsettings.json* properties to **Copy if newer** instead.

    a) Restore Packages
    b) Compile the solution
    c) Make sure the unit tests execute successfully



2. Add Application Insights to the PartsUnlimited Website.
   a) Select the PartsUnlimited Website, right -click, add Application Insights.

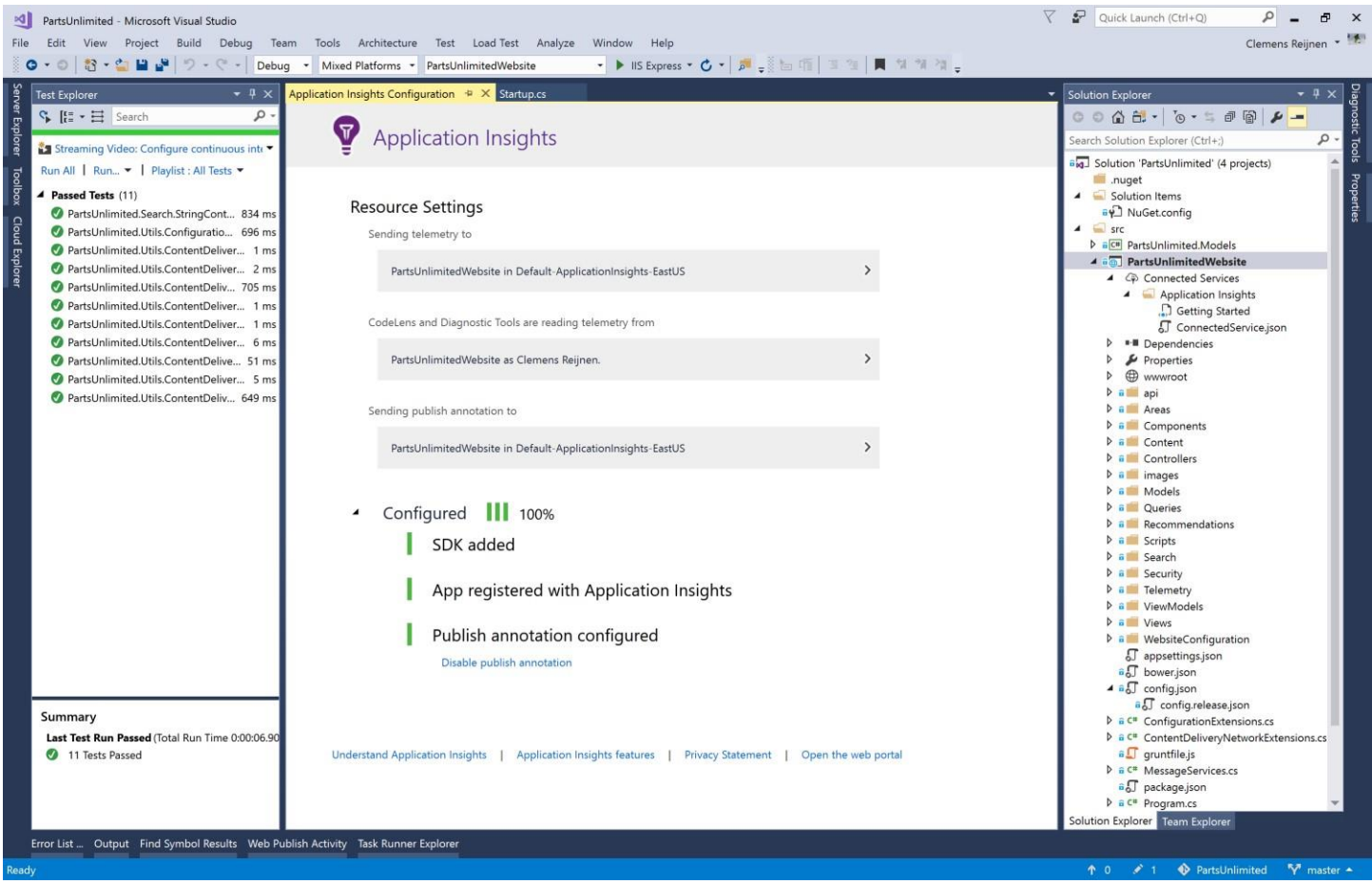
   b) Create new Application Insights resource. Update the SDK and select your Azure subscription.



c) Configure resource settings:

i. Telemetry

ii. CodeLens and Diagnostics



3. Add and check the Application Insights resources (see also, https://github.com/Microsoft/ApplicationInsights-aspnetcore/wiki/Getting-Started).

    a) Install NuGet package Microsoft.ApplicationInsights.AspNetCore
    b) Add appsettings.json instrumentation key
    c) Add to startup.cs to the Application Insights instrumentation code

        i. In the startup method: appsettings.json and the isDevelopment statement.

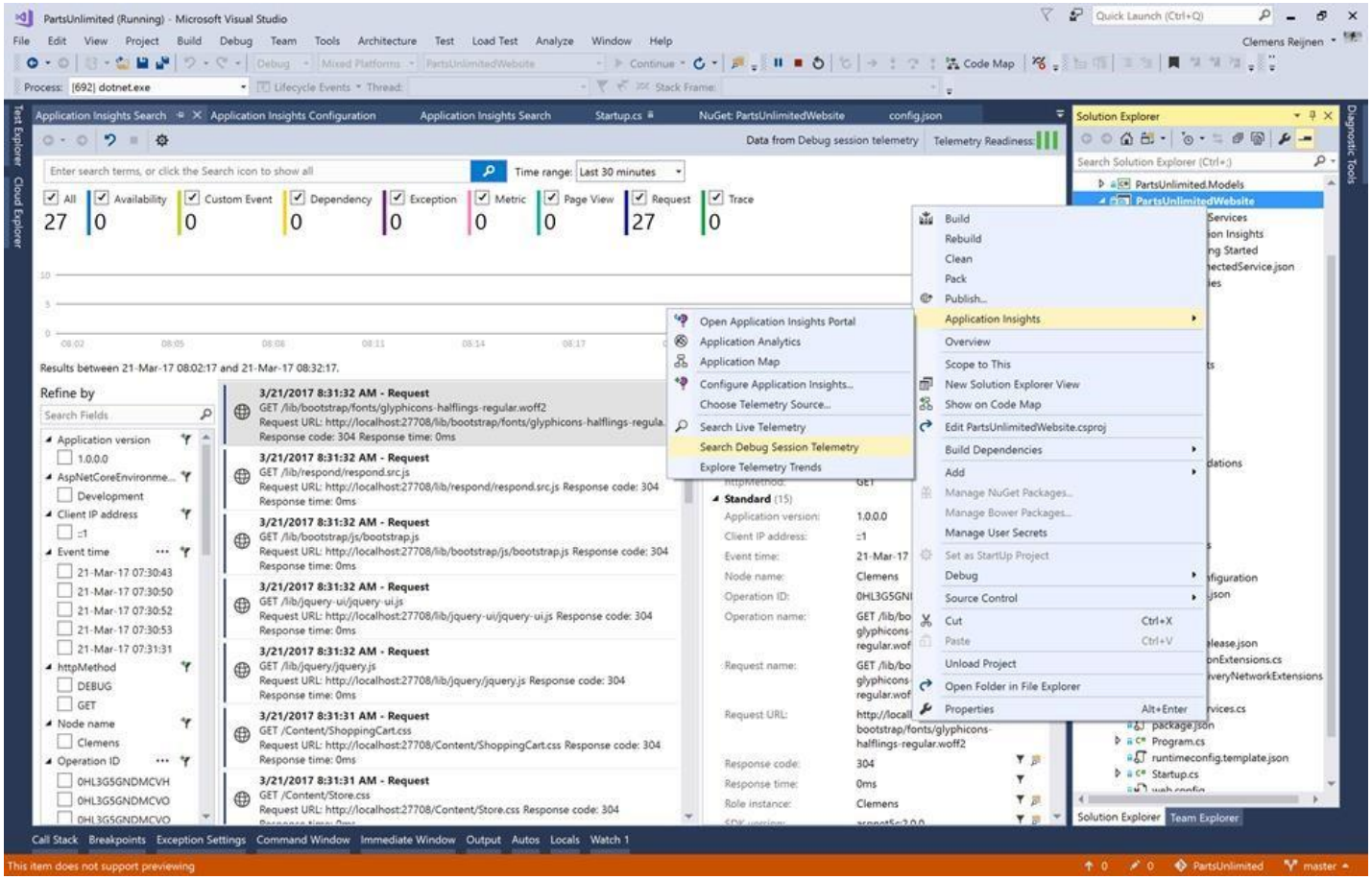
```
var builder = new ConfigurationBuilder()
    .AddJsonFile("config.json")
    .AddJsonFile("appsettings.json")
    .AddEnvironmentVariables(); //All environment variables in the

if (env.IsDevelopment())
{
    builder.AddApplicationInsightsSettings(developerMode: true);
}

Configuration = builder.Build();
```

        ii. In the ConfigureServices method: *services.AddApplicationInsightsTelemetry(Configuration);*

4. Create telemetry data by running the website.

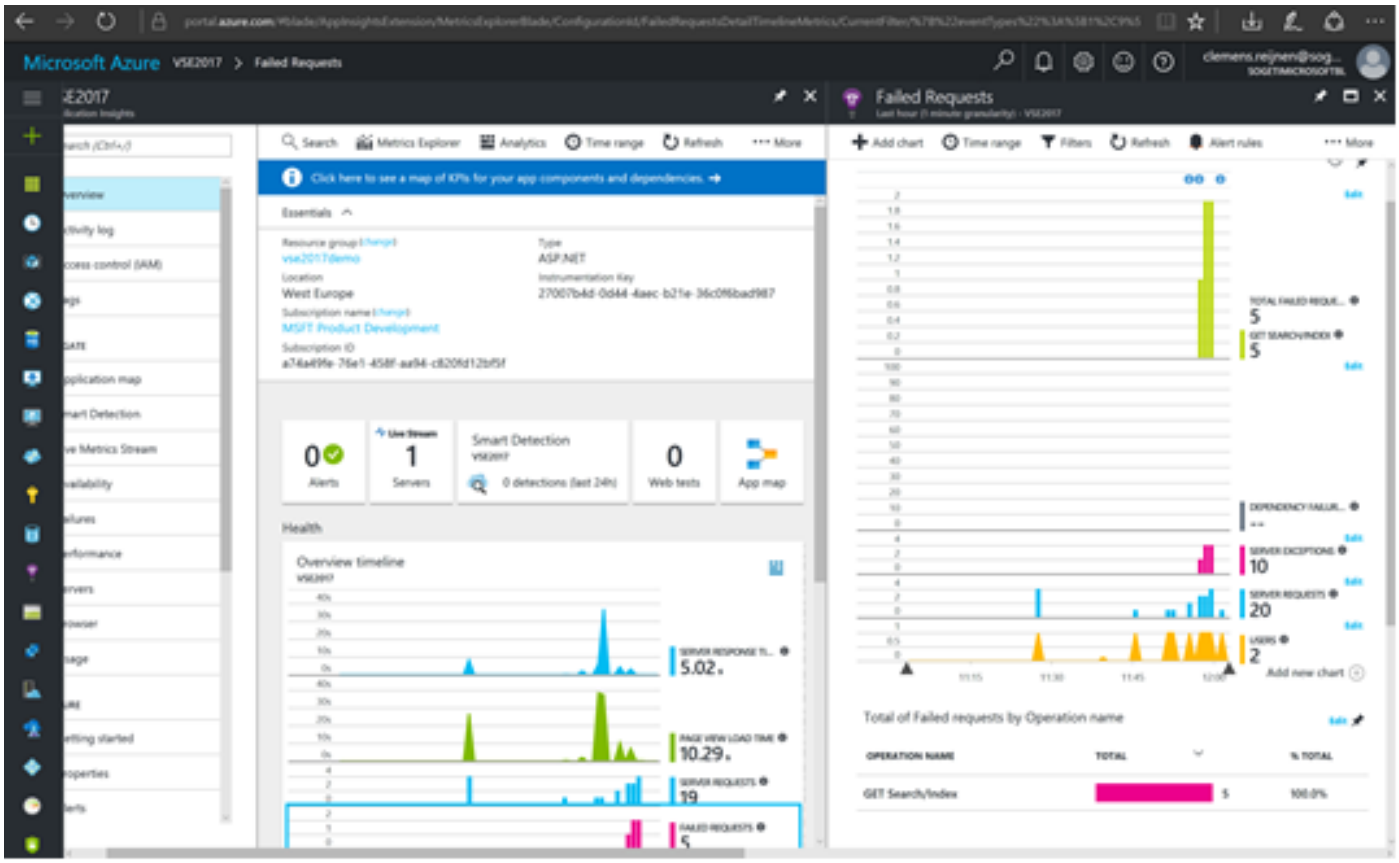
a) Start the application from Visual Studio. Press F5.
b) Browse website and search for products.

5. See your telemetry in Visual Studio.
   a) In Visual Studio, open the Application Insights window. Either click the Application Insights button, or right-click your project in Solution Explorer, select Application Insights, and then click Search Debug Telemetry.



6. Investigate telemetry data.

   a) Open Azure Portal, goto portal.azure.com
   b) Navigate to the just created Application Insight resource.
   c) Investigate reports.



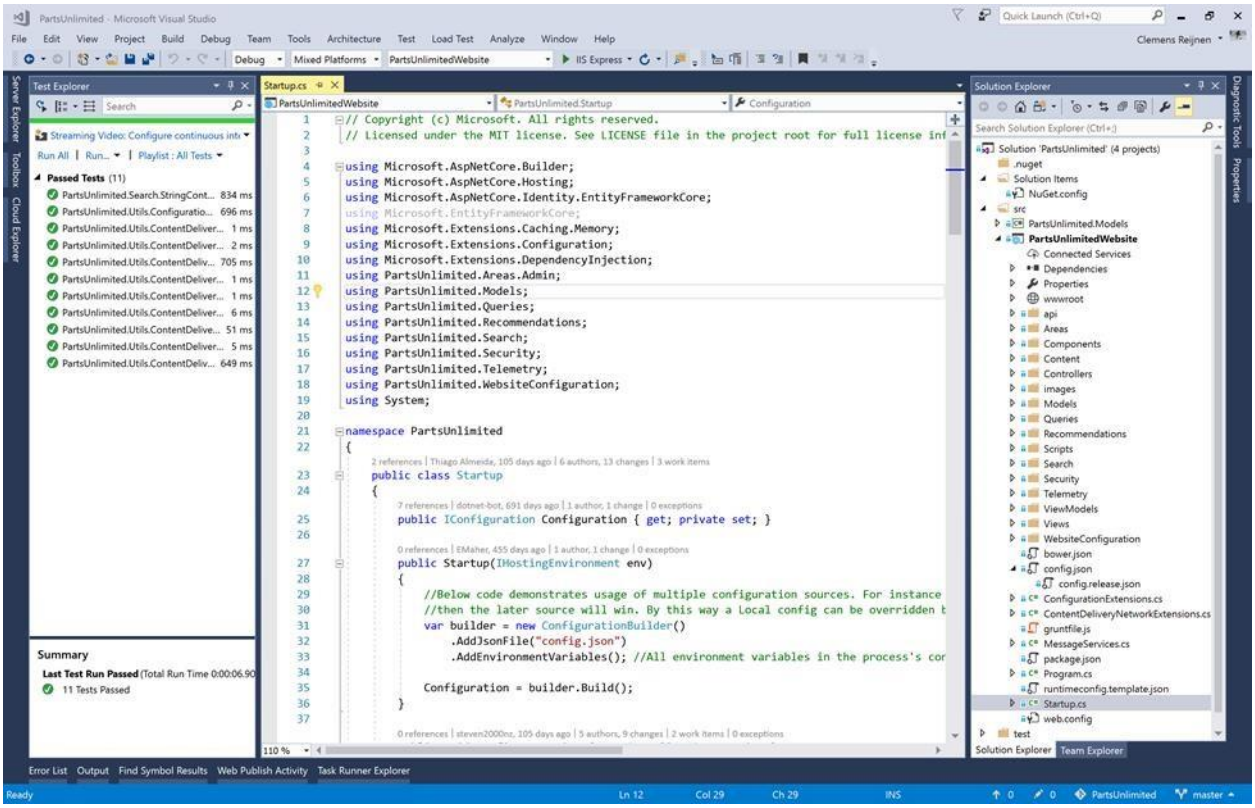
Scenario 2: User Telemetry

As the business owner of PartsUnlimited, you want to track how the Sales site is used so that you know which part of the site tracks the most users and by whom. When a new feature is added or changed, you want to see the impact of the change with a mark on the telemetry data.

Implement and configure, as a developer, the usages data of the website. Show the telemetry data in the Azure portal and separate it by release. Make sure the different stages report to different reports.

Lab 2: Add Usages Telemetry Metering

1. Open PartsUnlimited within Visual Studio from Lab1.
    a) Restore Packages
    b) Compile the solution
    c) make sure the unit tests execute successfully

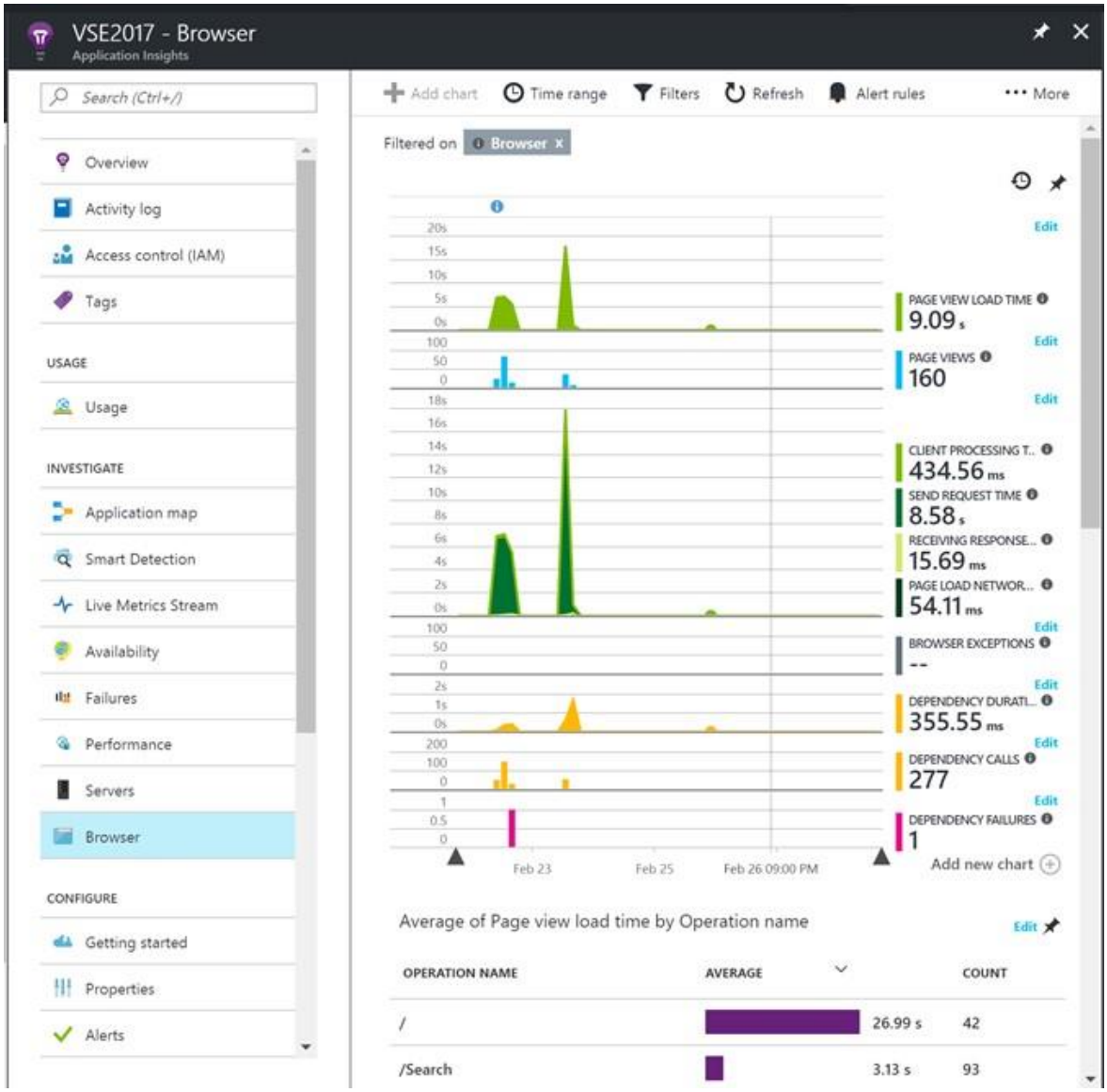


2. Add or validate the web front-end monitoring for user telemetry.
    a) Go to the `_Layout.cshtml file`.
    b) Add `@inject Microsoft.ApplicationInsights.AspNetCore.JavaScriptSnippet JavaScriptSnippet`.
    c) Add before the closing head section `@Html.Raw(JavaScriptSnippet.FullScript)`

Lab 3: Make Telemetry Metering Stage Specific

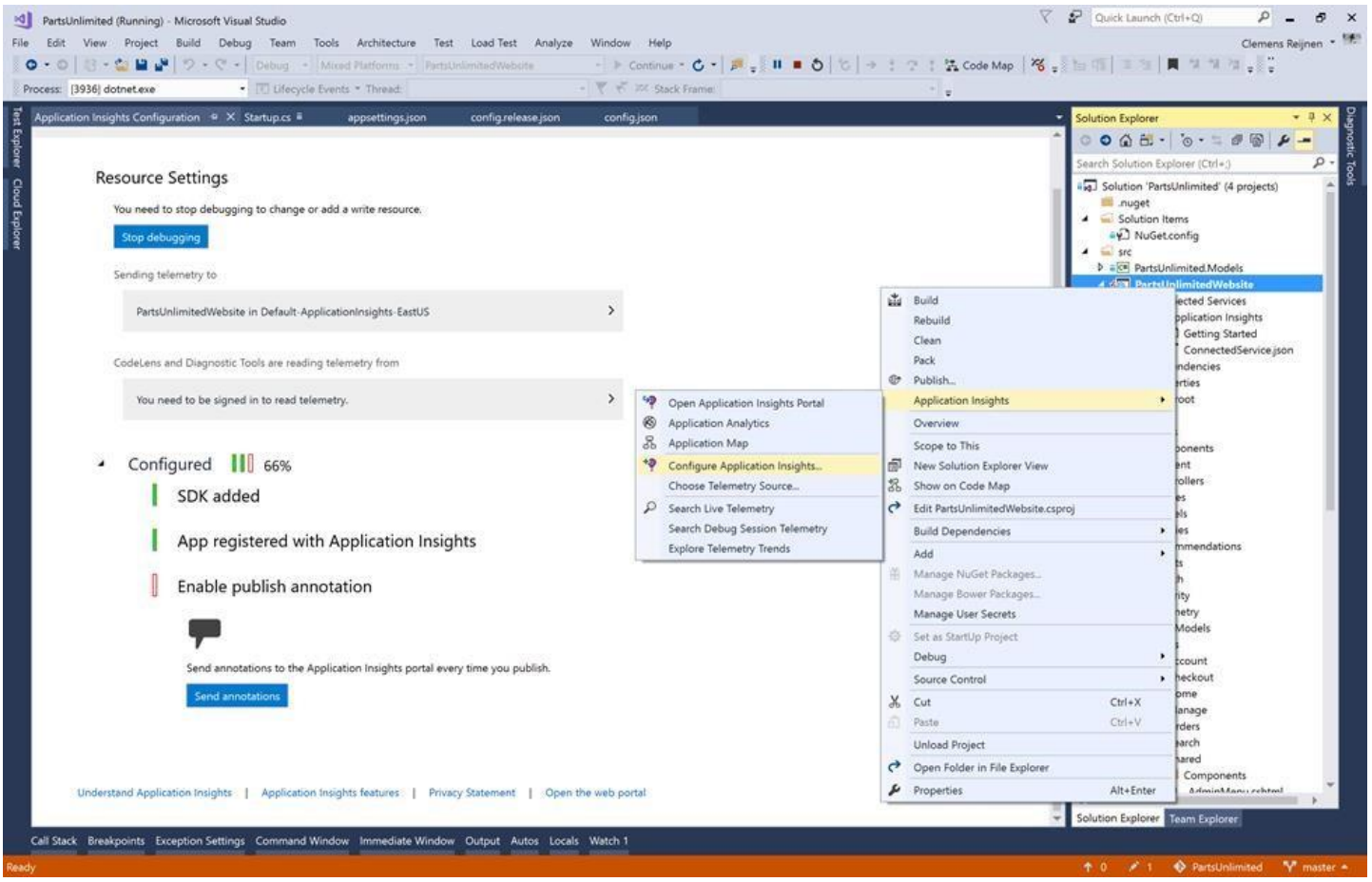
1. Make the instrumentationKey dynamic so it can be set for different stages.
    a) Add an instrumentation key section to config to json and to the config.release.json.
    b) Remove the `.AddJsonFile("appsettings.json")` from the startup.cs.

2. Create telemetry data by running the website and browse the products.
3. Investigate created front-end request and response data.

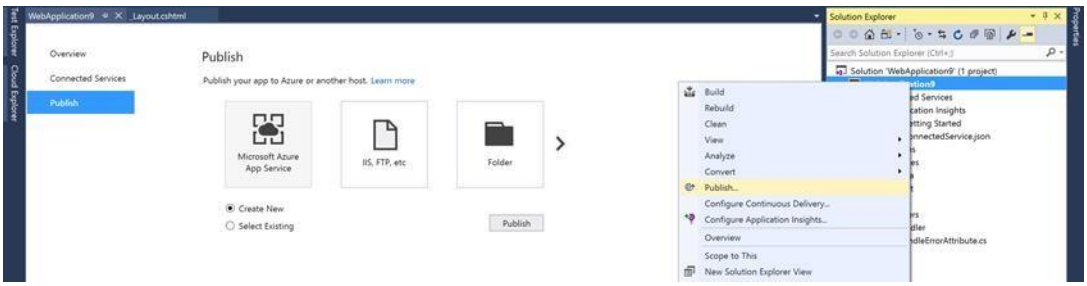


Lab 4: Set Version Publish Annotations to Telemetry Graphs

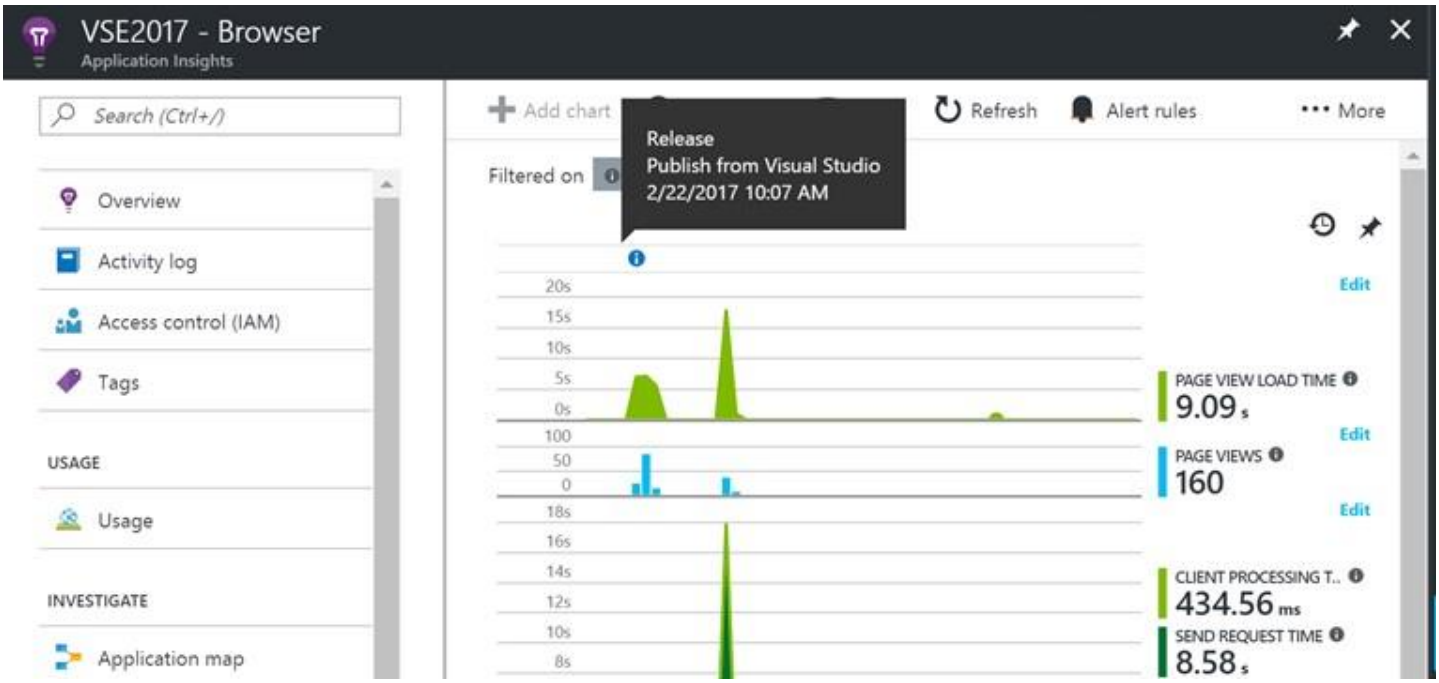
1. Add publish annotations Application Insights configuration.
    a) Stop debugging.
    b) Open the Application Insights Resource Settings.
    c) Configure Publish annotations if not done already.



2. Publish the PartsUnlimited website to Azure WebApps and see the publish annotation in the Application Insights graphs.
    a) Right-click on the PartsUnlimited WebApp and select Publish WebApp.



    b) Create Azure Resource “PartsUnlimited-RG”.
    c) Publish PartsUnlimited website.
    d) Browse the PartsUnlimited website.
    e) Open the Azure portal Application insight resources see the publish annotation



Scenario 3: Health and Performance Monitoring

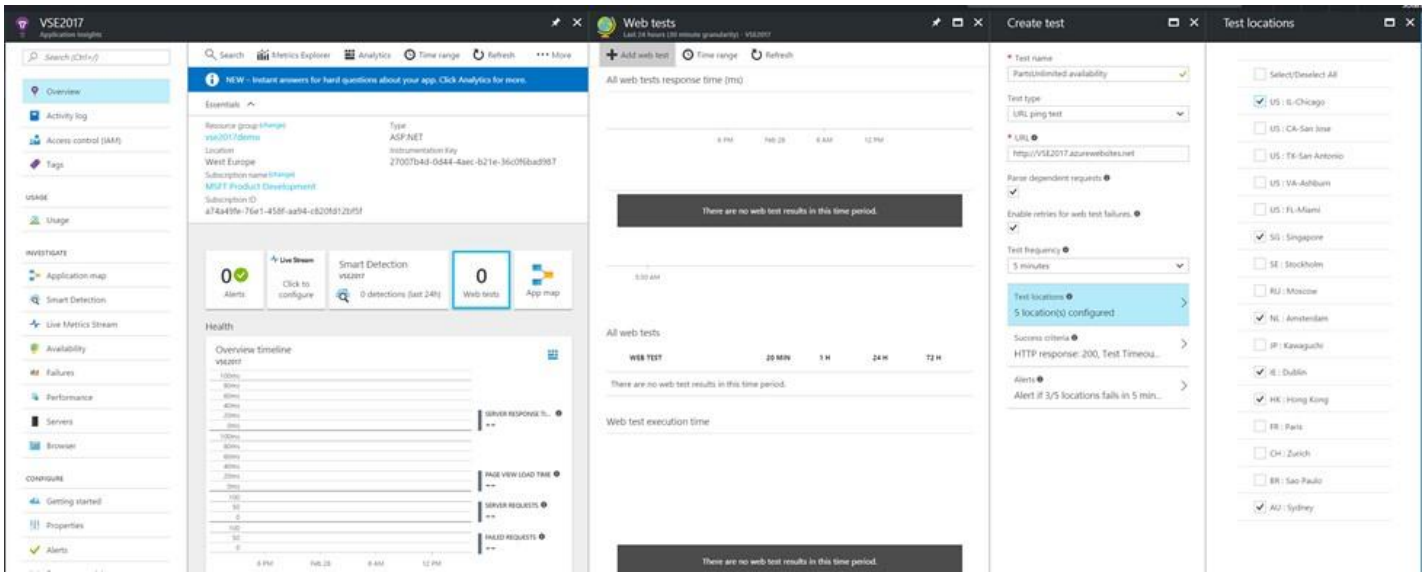
The sales of PartsUnlimited is expanded globally to multiple regions. These regions are Asia, West Europe, and North Europe. From Asia, a lot of traffic is expected on the search part of the website, due to interest in Jumper Leads from that area. The product owners from PartsUnlimited want to monitor the availability of the website from all regions. Next to availability monitoring, they also want to be sure the website stays responsive when an enormous load is taken from Asia on the search part of the site.

Validate Europe is still available when the Asia load is generated.

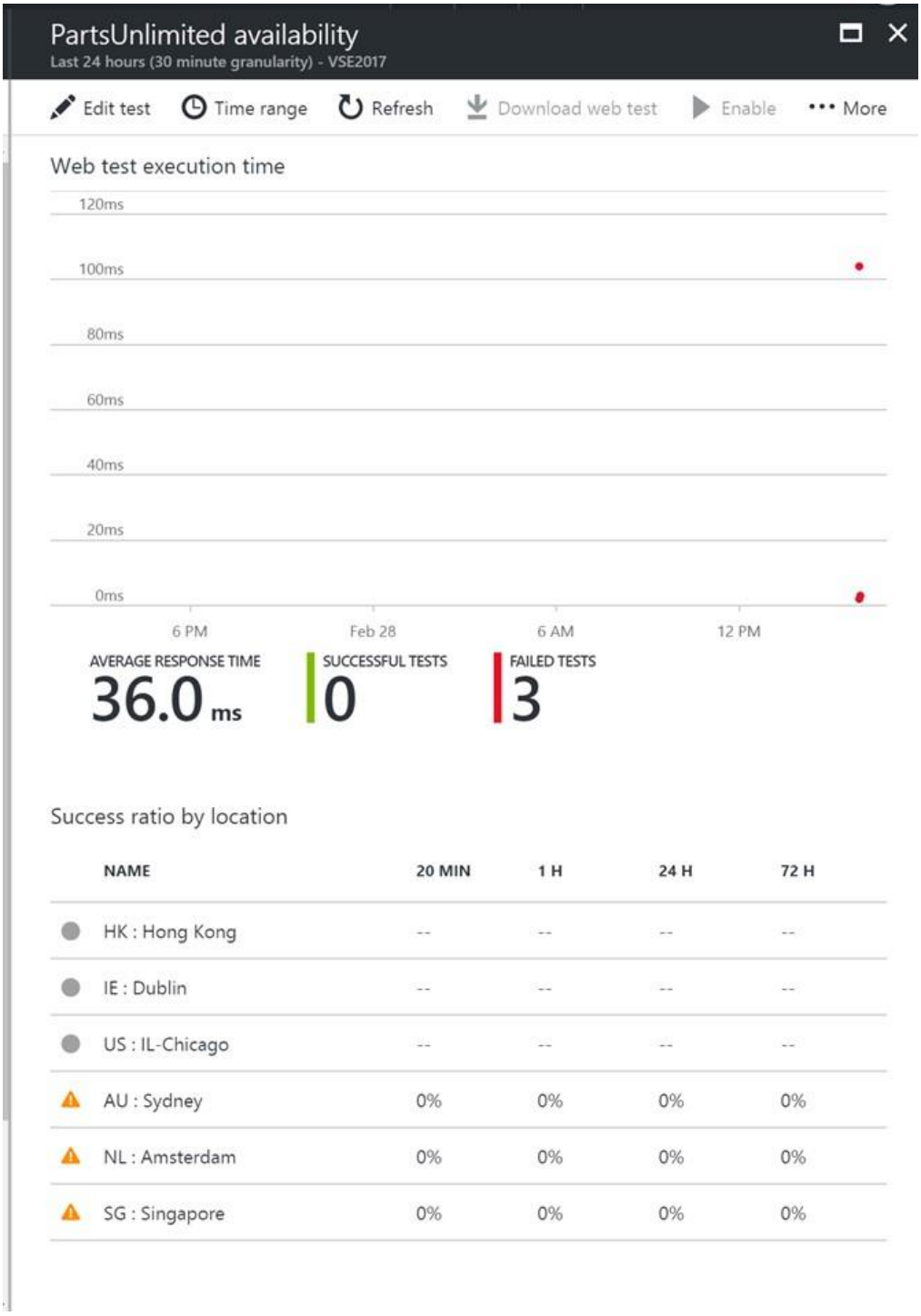
Lab 5: Set Availability Ping Tests for PartsUnlimited Website

1. Continue from creation of Application Insights resources and Azure WebApp resource in previous labs.

2. Configure availability tests and monitor availability from different regions.



a) Open the Azure Portal and go to the Application Insights resource.
b) Select Web Tests in the overview pane.
c) Add WebTest, “PartsUnlimited availability”.
d) Select locations in Asia and Europe.
e) Investigate the graph.

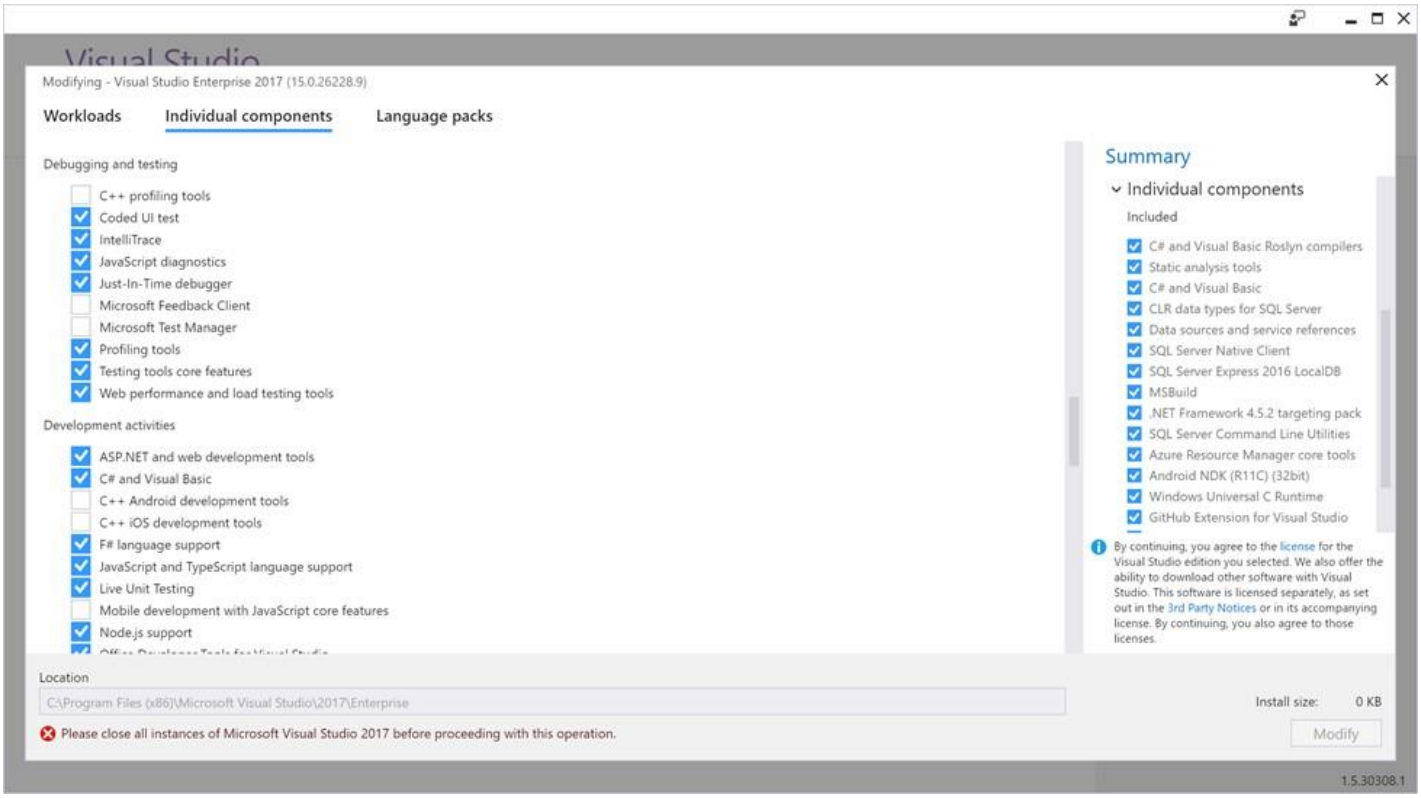


Lab 6: Create Multi-Step Web Test

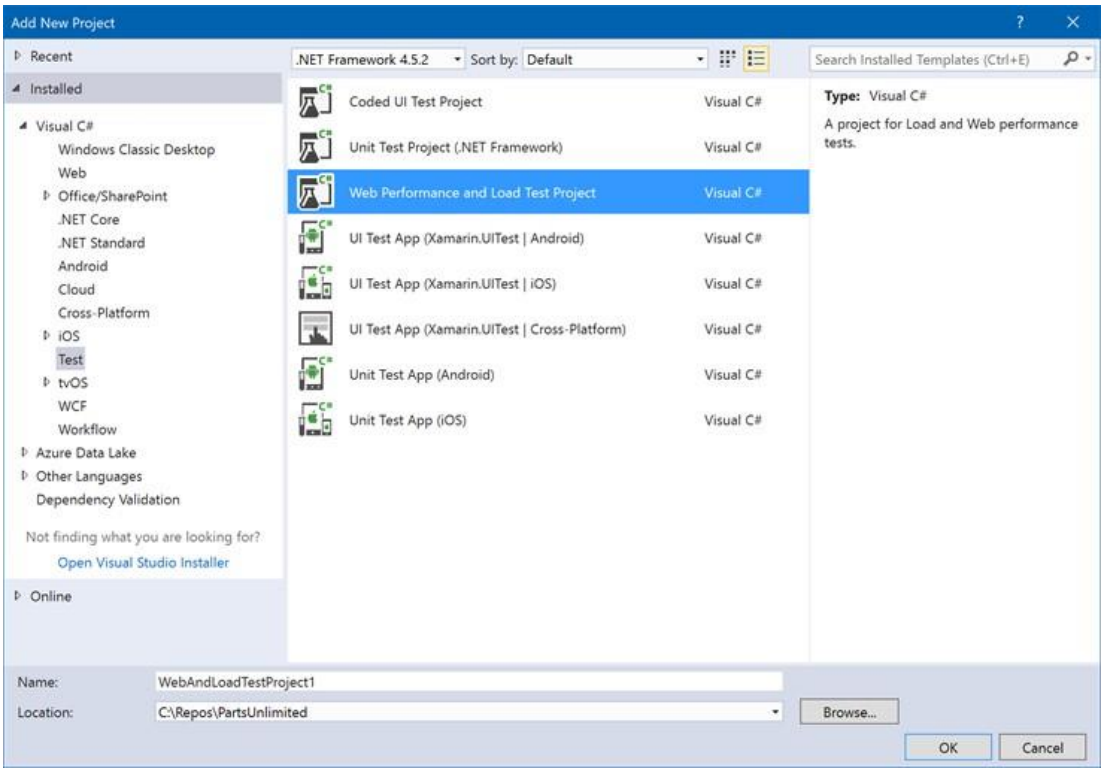
Create add to basket multi step test, with response time requirement in Visual Studio.

1. Open Visual Studio with the PartsUnlimited solution. Add a WebAndLoadTests project to the solution.

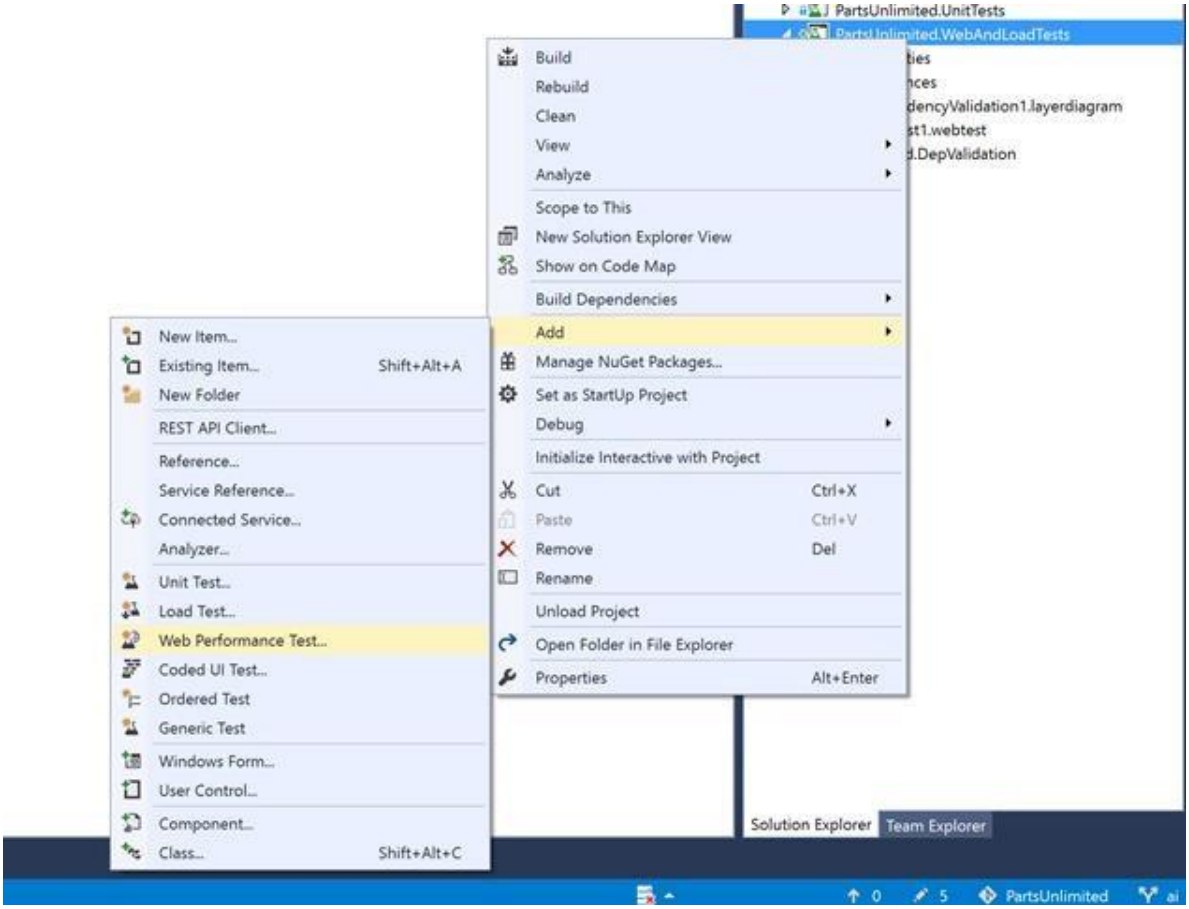
(Note: if the load and performance project isn't available, install it via the Visual Studio Installer.)



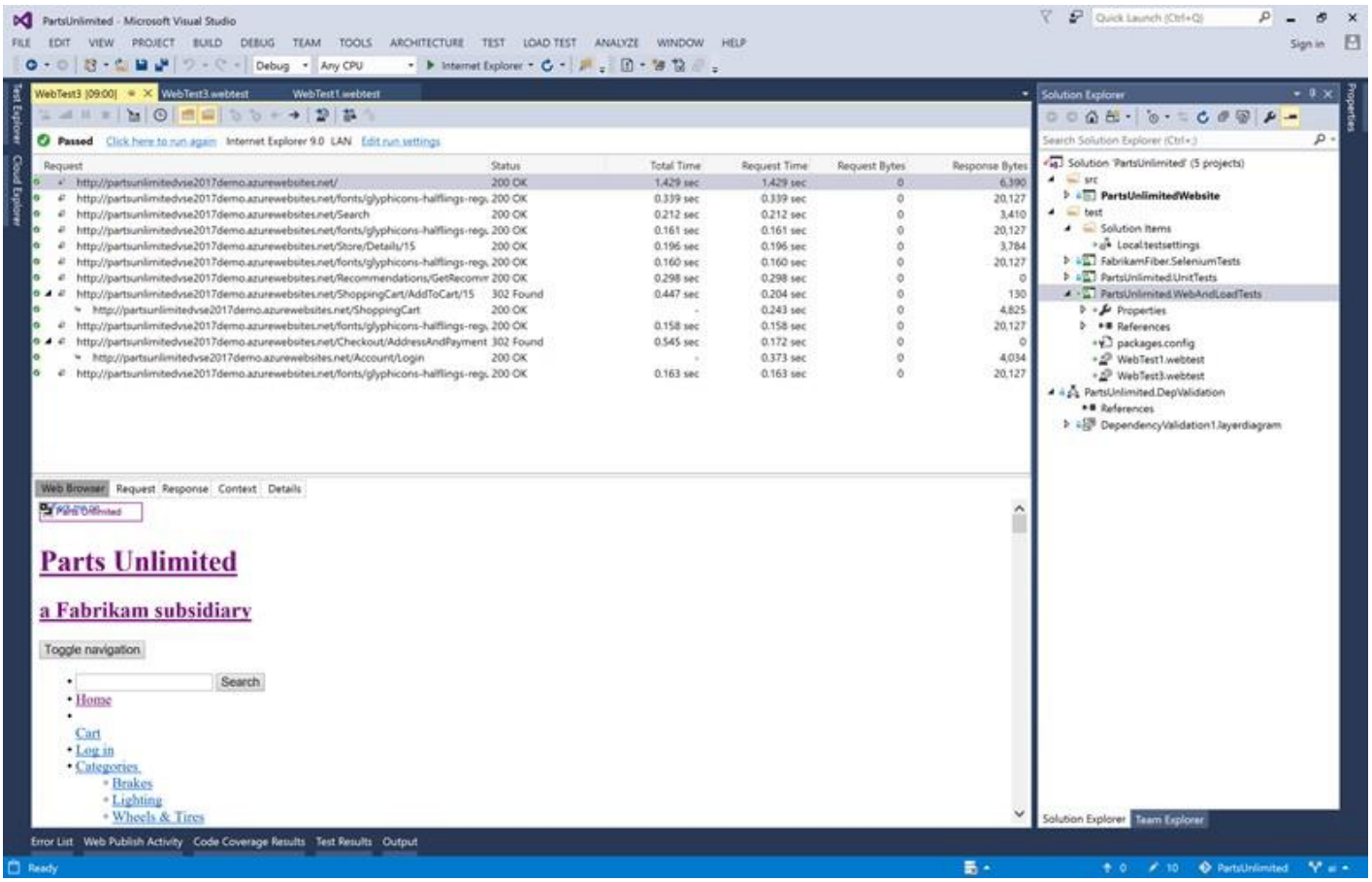
a) Select the solution folder Tests and add a WebAndLoadTests project. Name it: PartsUnlimited.WebAndLoadTests.



2. Add a Web Performance Test to the WebAndLoadTests project.
    a) Right-click on the WebAndLoadTest project and add a Web Performance Test.

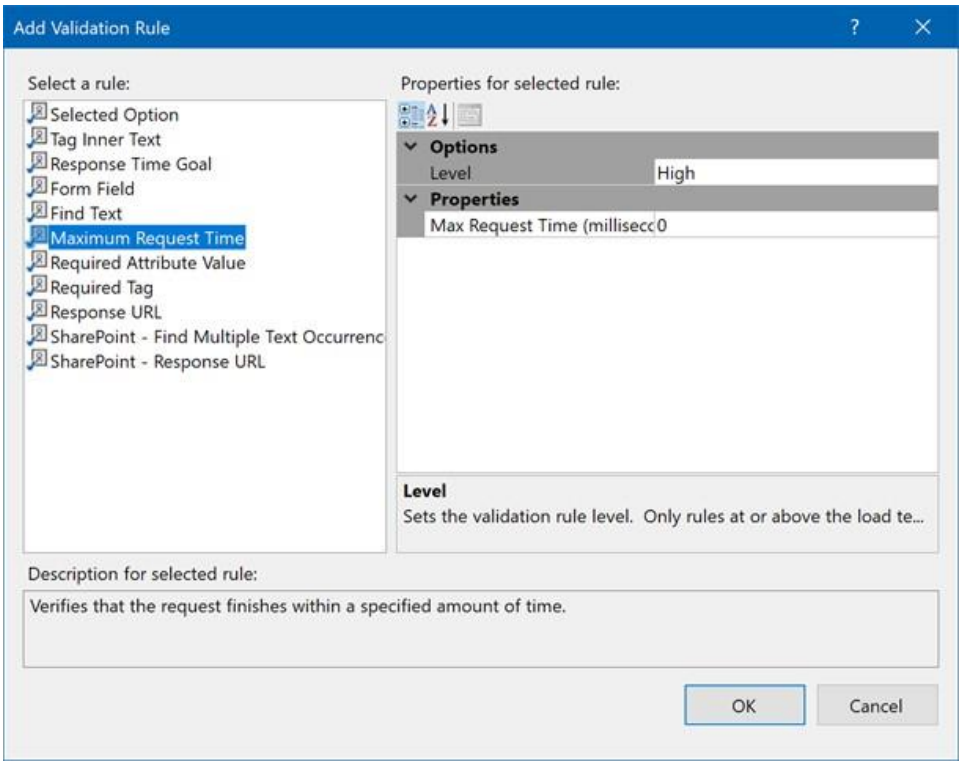


    b) Record the checkout scenario with the web test recorder.
        a) Open the PartsUnlimited website.
        b) Select ‘jumper lead’ product.
        c) Select the product.
        d) Check out.
        e) Stop the recording.
        f) Delete any requests made to any host different than *.azurewebsites.net (e.g., calls to app Insights, like dc.services.visualstudio.com) to make sure the test isn’t testing any external service like Application Insights.



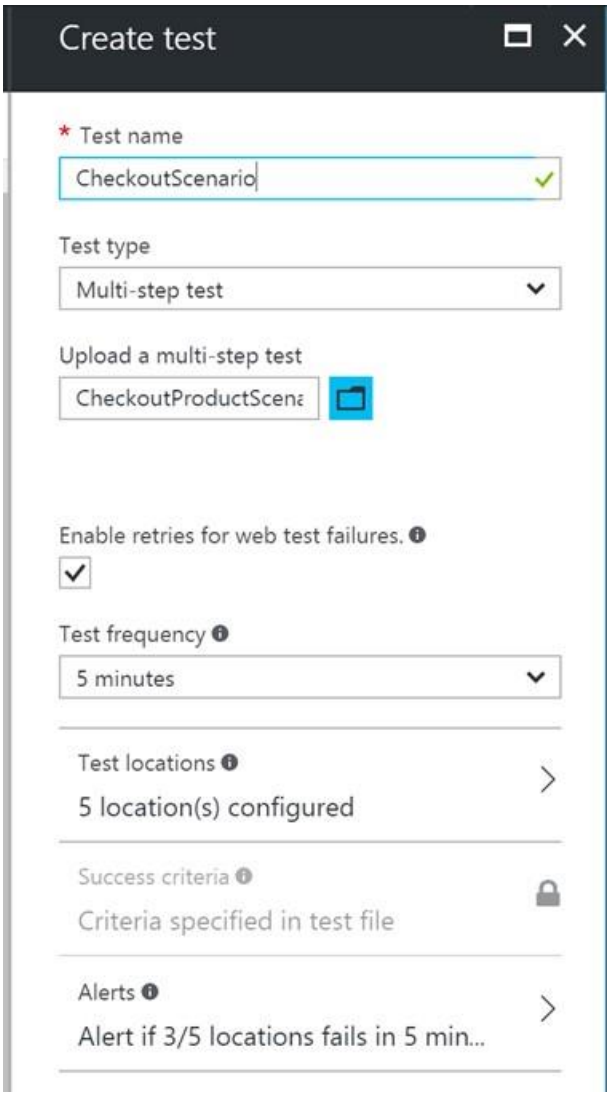
c) Rename the WebTest to ‘CheckoutProductScenario.webtest’.

d) Add a Maximum Request Time Validation to validate requests are processed within 1 second.



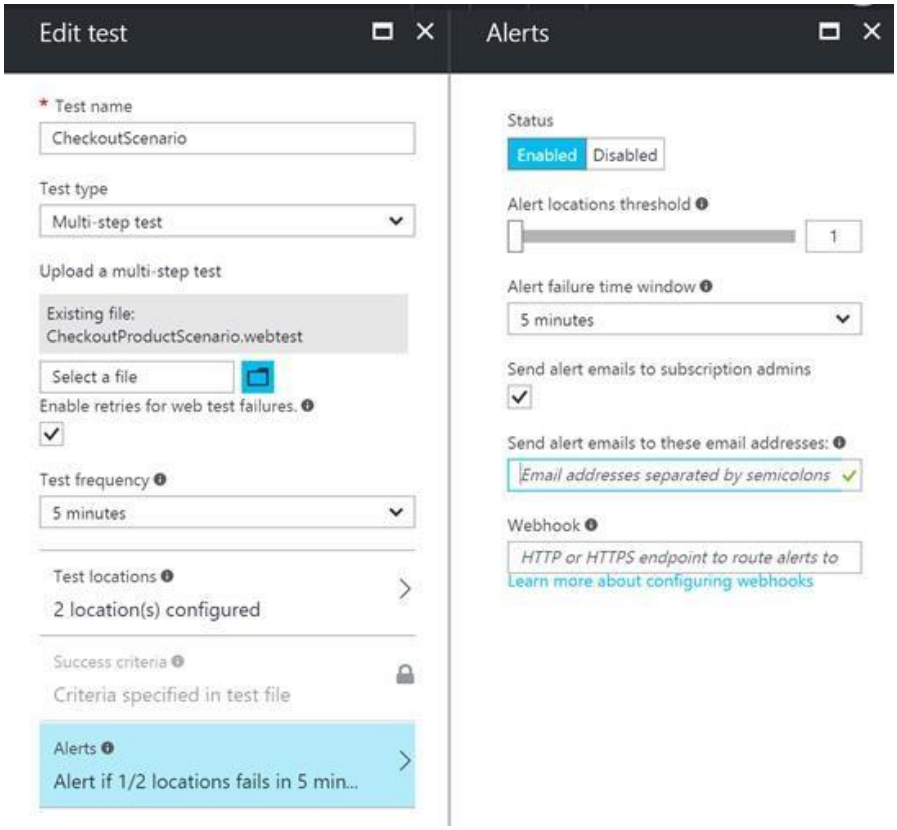
e) Run the web test scenario and validate all requests are responding within 1 second. *Note: when a test step returns a 404 on a jquery request, set the property 'parse dependent requests' to false.*

3. Set the multi-step test as an availability test for Europe regions.
    a) Open the Azure portal and go to the Application Insights resource.
    b) Select availability tests.
    c) Add the CheckoutProductScenario.webtest multi-step tests.

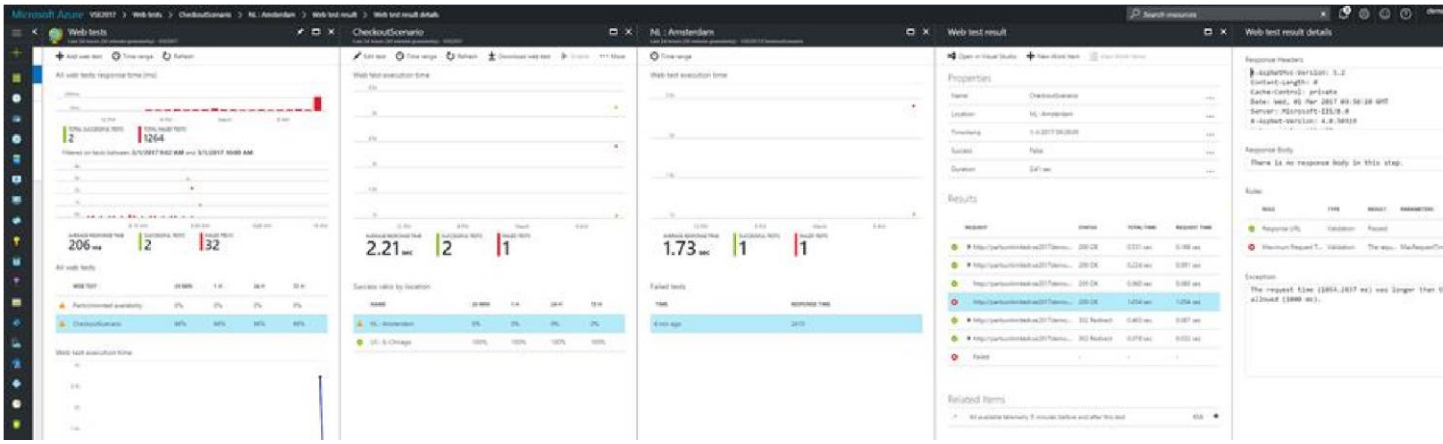


d) Set the locations to run the test from to Europe and US locations.

e) Configure the WebTest to send alerts to your email address.



f) Wait 10 minutes and investigate the results in the Azure portal.



g) Select the Checkout web test and investigate the results.

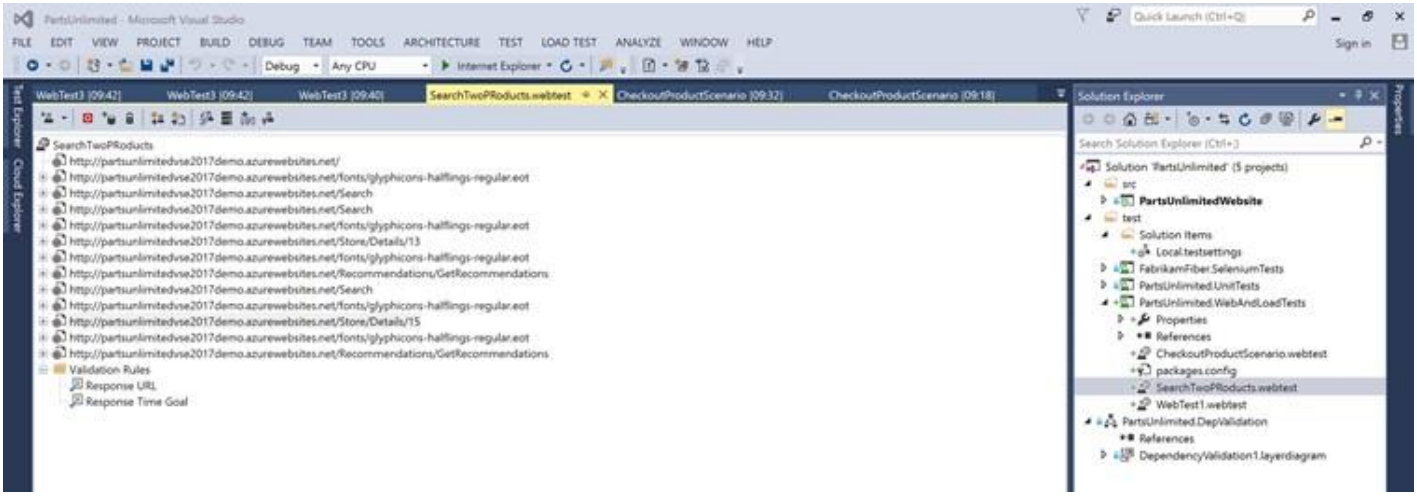
Lab 7: Create Cloud Load Test

1. Create Cloud Load test for Asia search load.
   a) Add performance test to the Visual Studio project.
   b) Record the scenario for searching for jumper leads and batteries.
      i. Open website PartsUnlimited.
      ii. Search for Batteries.

iii. Select Battery product.

iv. Search for Jumper Lead.

v. Select Jumper Lead product.



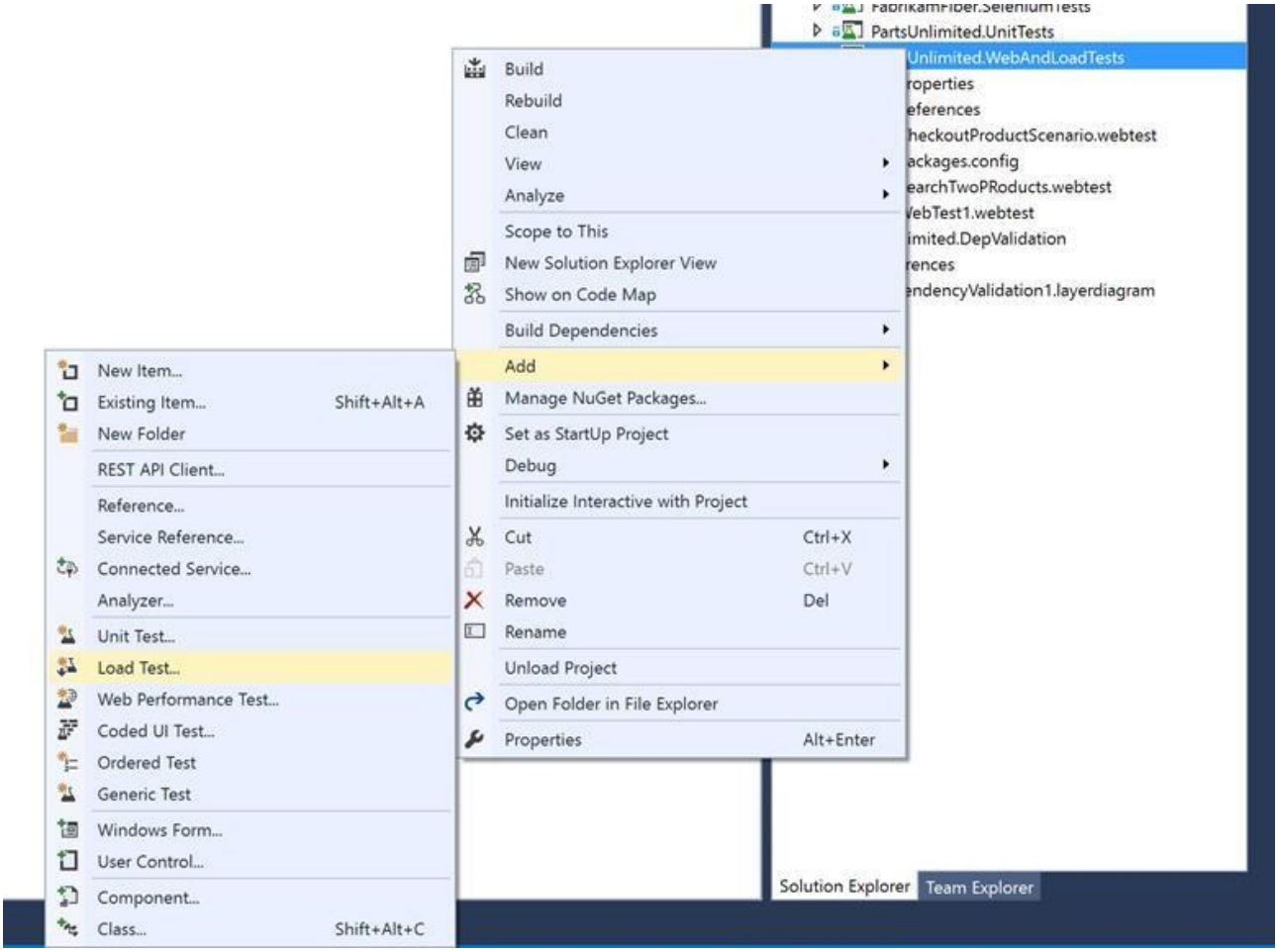
c) Delete any requests made to any host different than *.azurewebsites.net (e.g., calls to App Insights like dc.services.visualstudio.com) to make sure the test isn't testing any external service like Application Insights.

d) Run the performance test to validate the run successes.

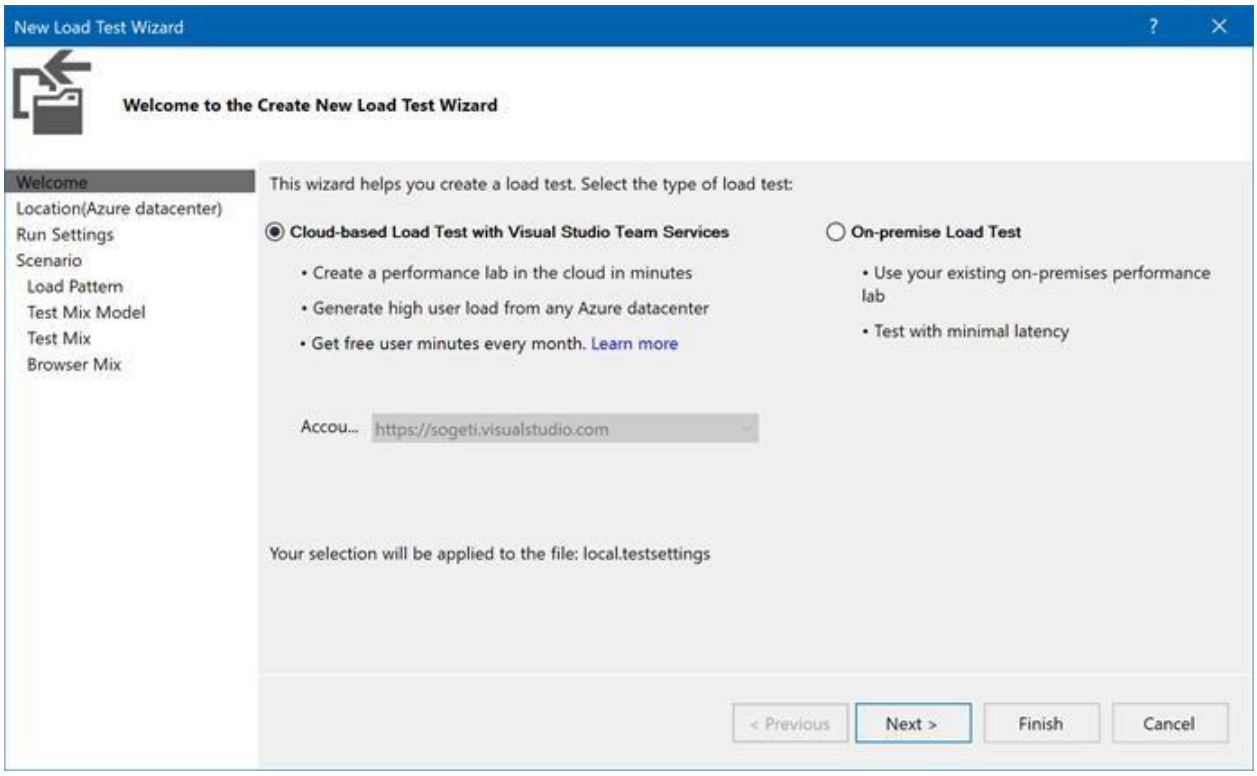
Note: when a test step returns a 404 on a jquery request, set the property 'parse dependent requests' to false.

e) Save the WebPerformance test as SearchProducts.webtest.

f) Add a Load Test to the WebAndLoadTests project.

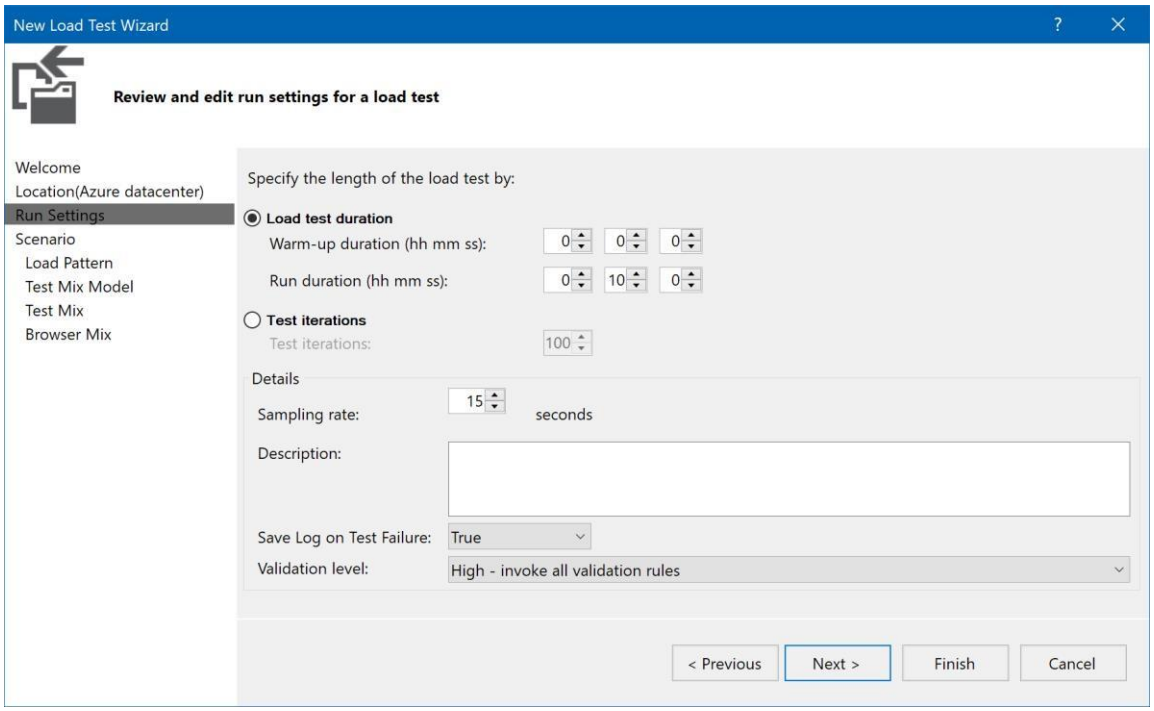


g) Select Cloud Load Test.

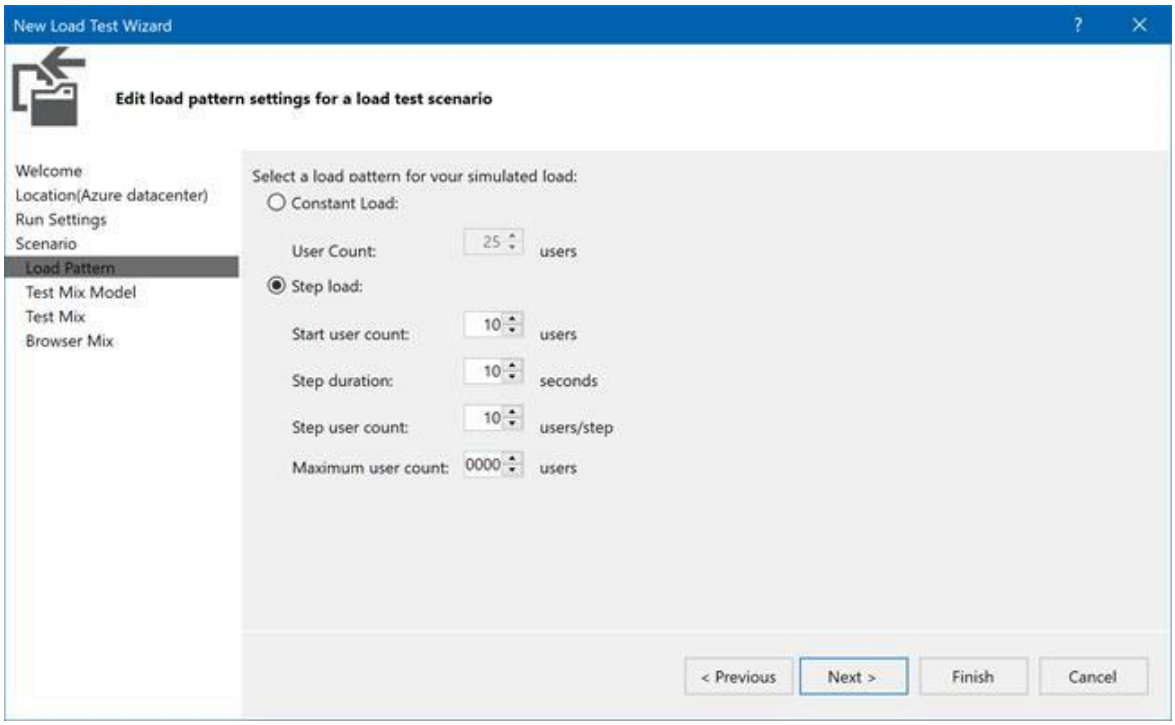


h) Create load test for 10.000 users for 10 minutes. Keep all settings in the New Load Test Wizard to default except:

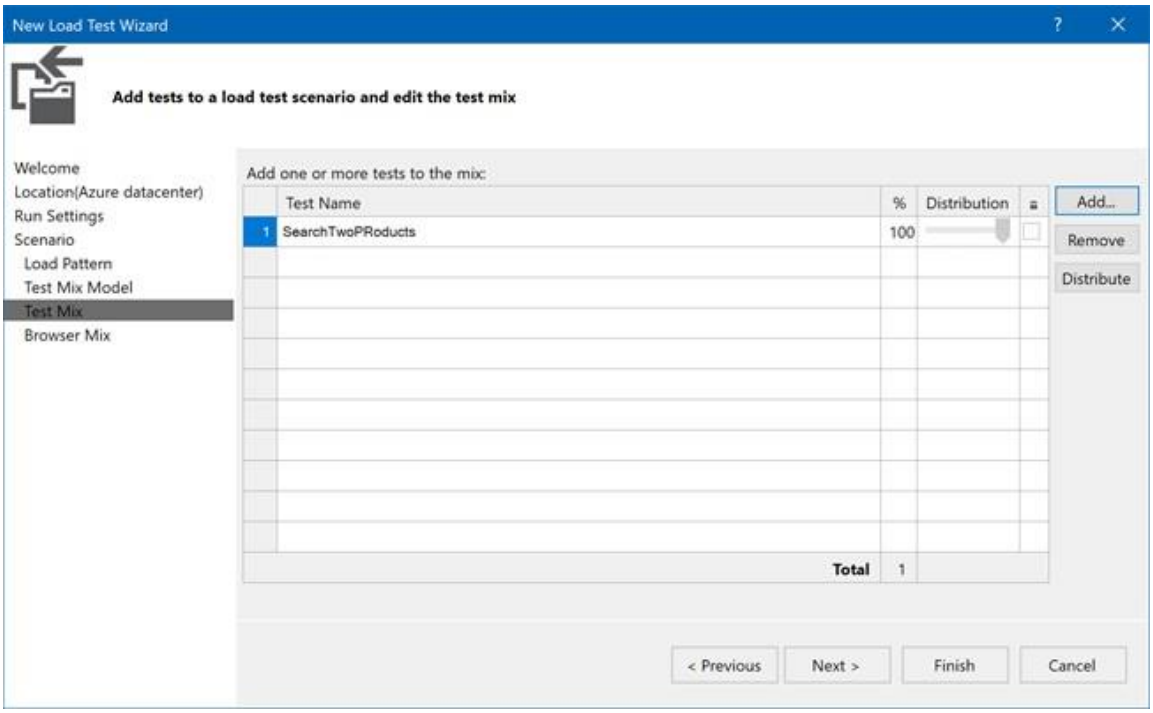
i. Set load test run settings to 10 minutes.



ii. Set Load Pattern to ‘Step Load’ maximum user count 10.000

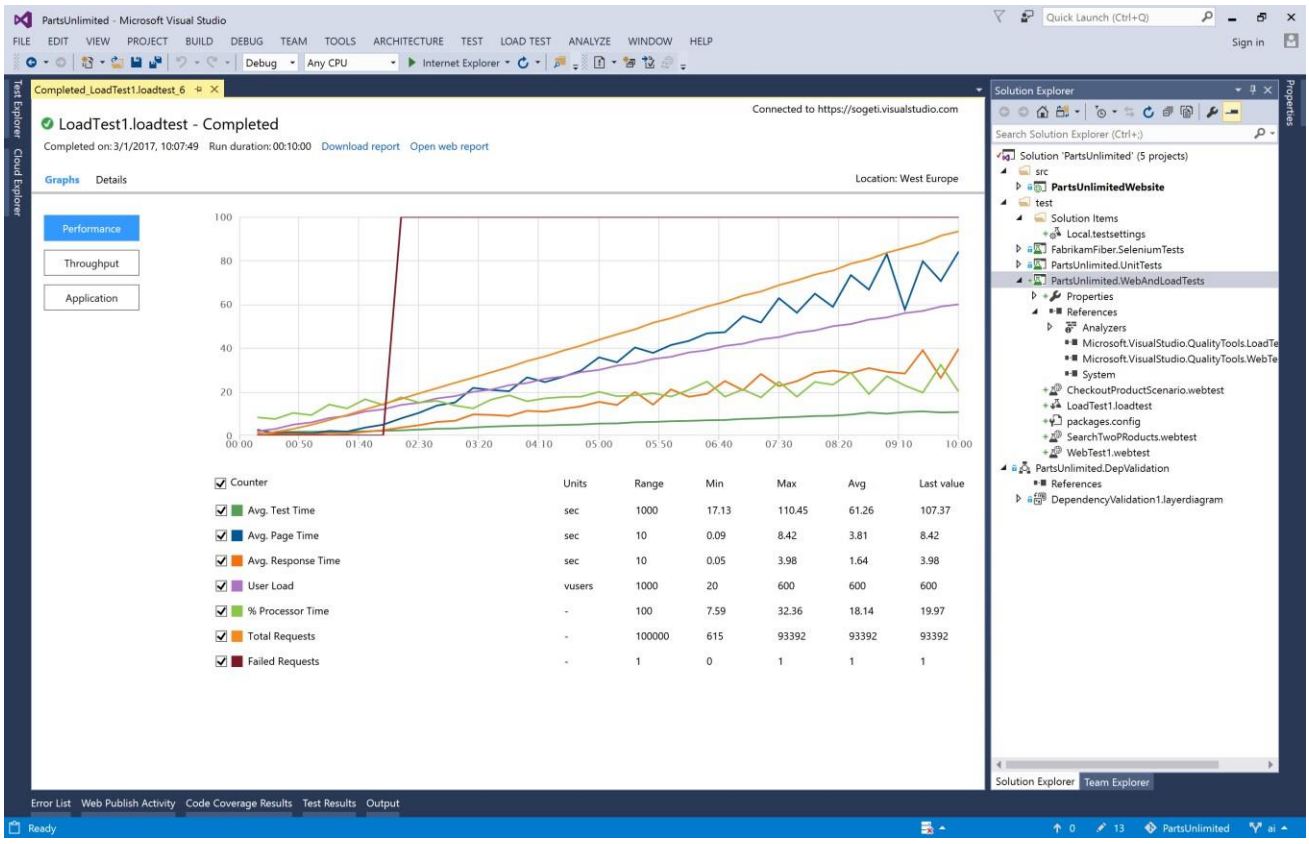


iii. Select the SearchProducts.webtest as the only test to execute.

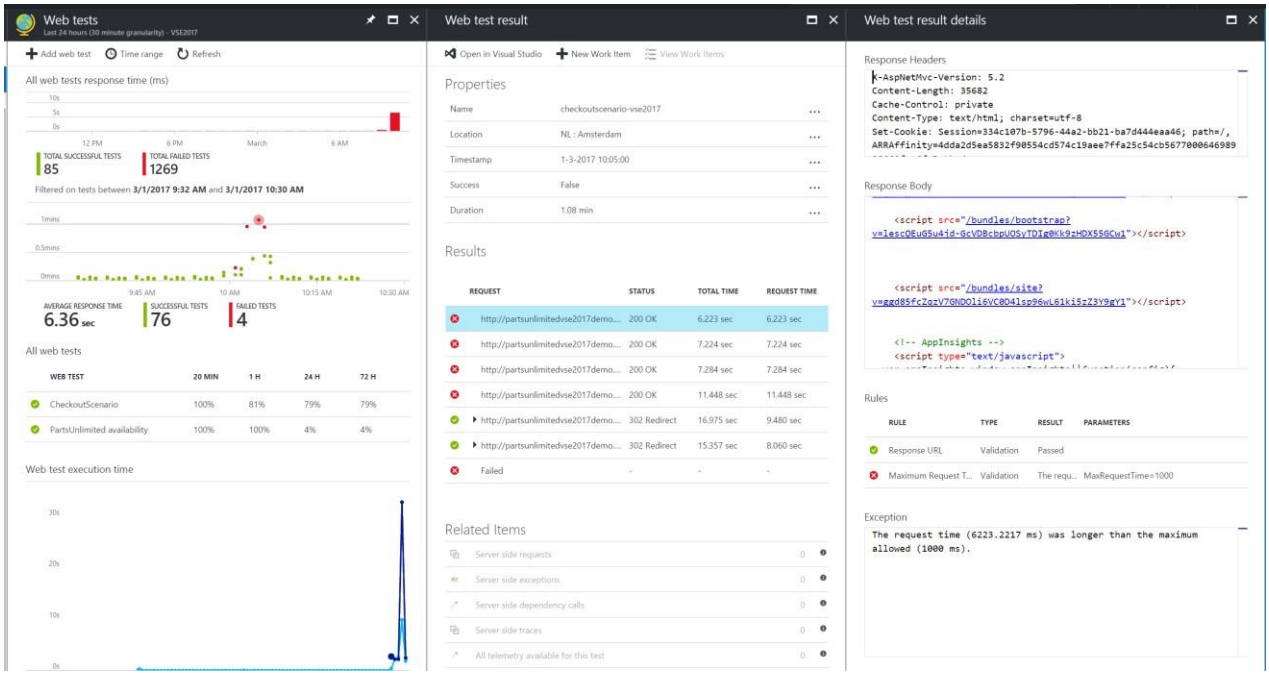


i) Run the load test.

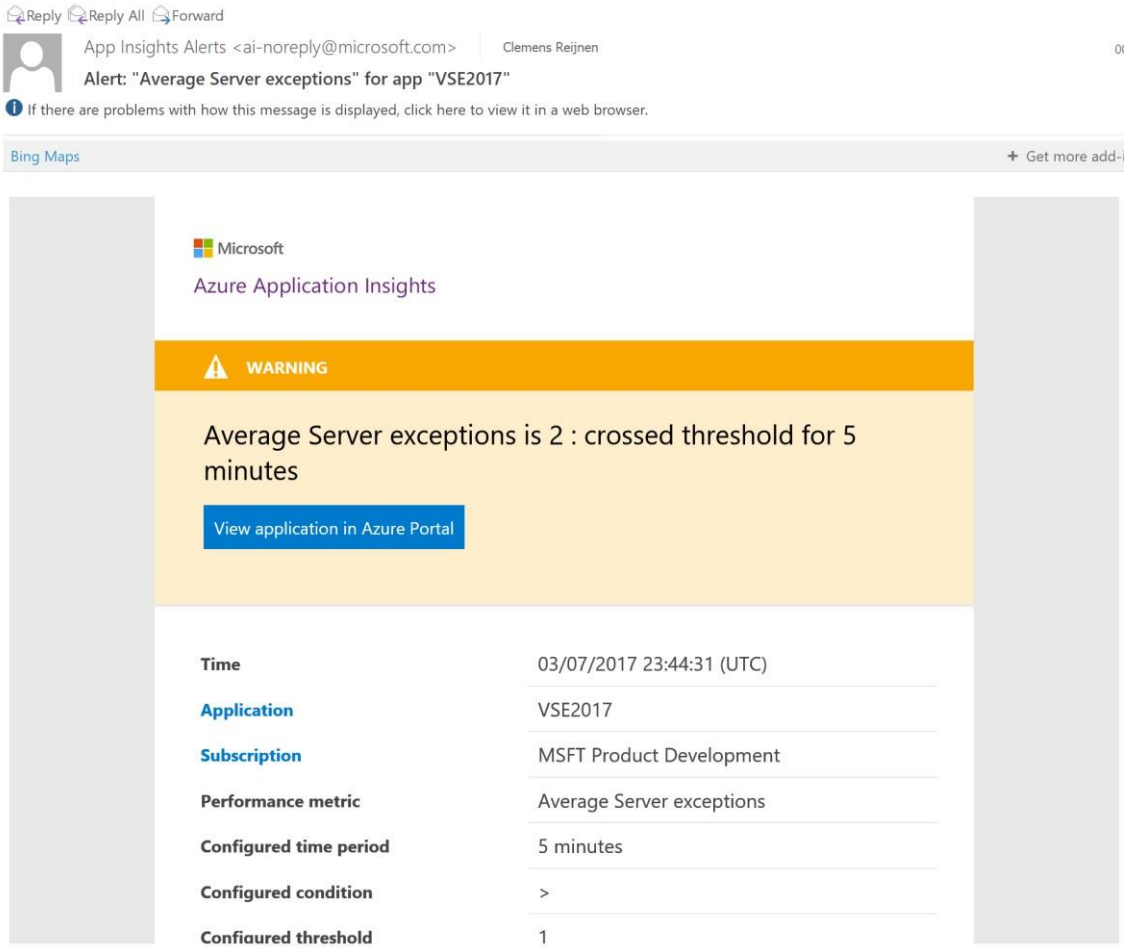
j) Investigate the result. Look at the graphs focus on the failing request and user load.



k) Investigate the Availability tests. Focus on the failing test during the load test. This indicates the checkout process in the US and EU are influenced by the search activity load.

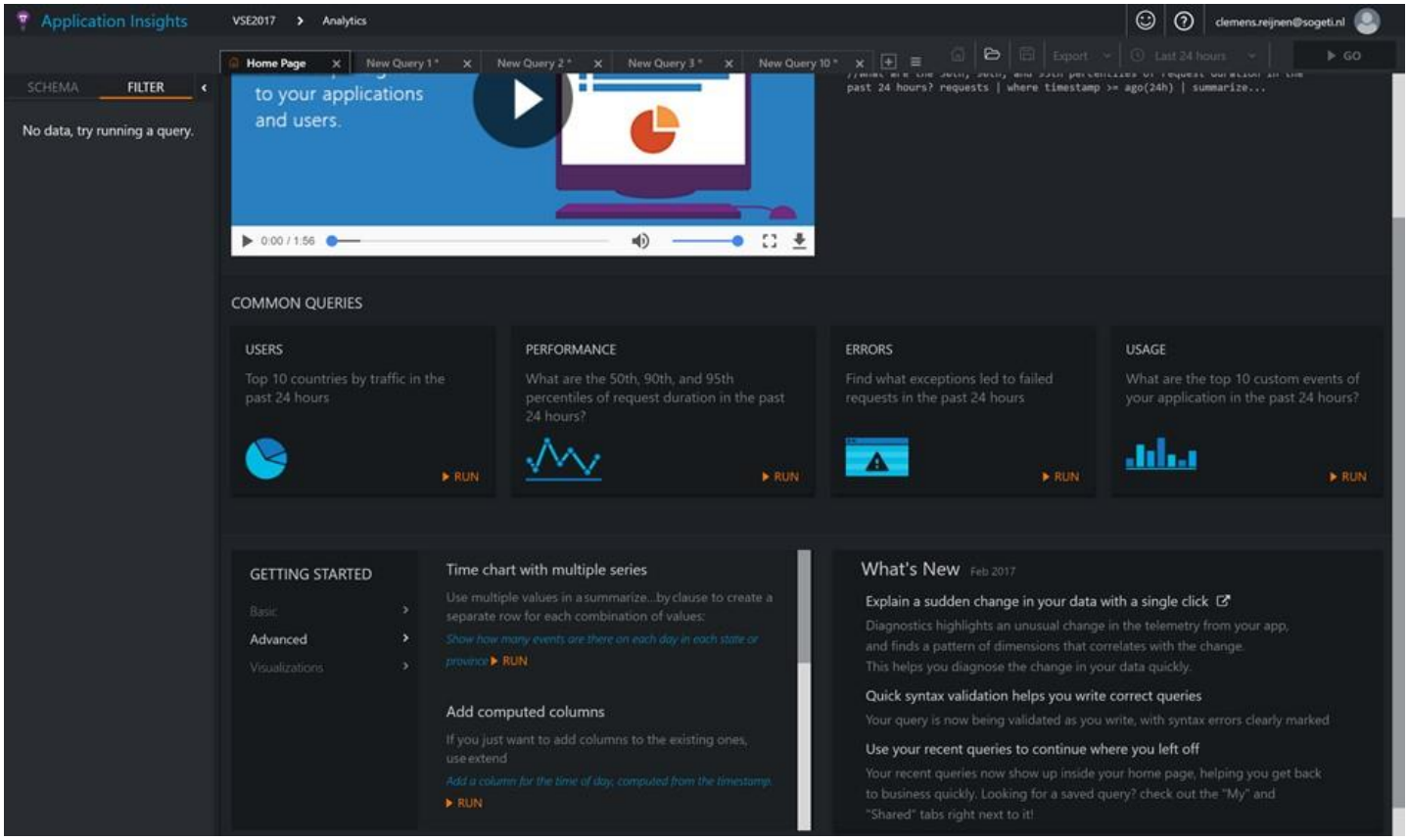


l) Investigate mailbox on send alerts from the WebTest.



Lab 8: Log Analytics with Application Insights

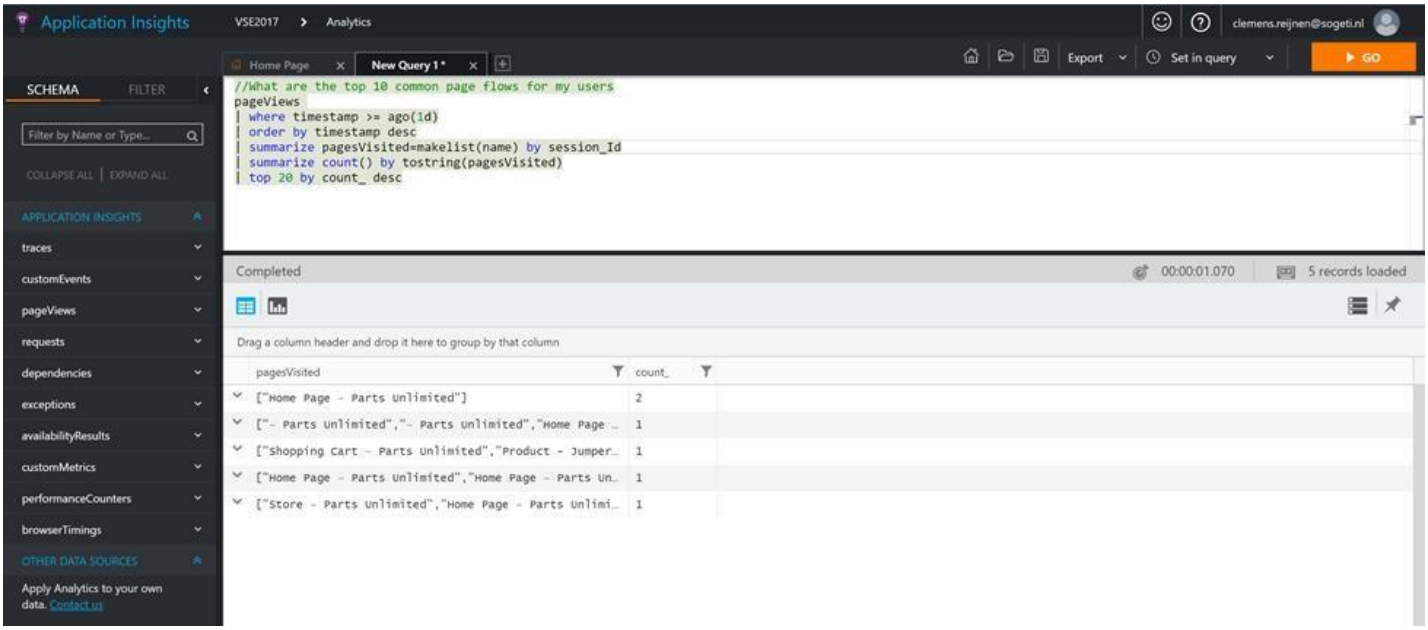
1. Open Azure portal.
2. Go to the created Application Insights resource.
3. Select ‘Analytics’ in the overview top-menu bar.
4. Investigate the query language capabilities.
5. Select ‘Getting Started’ → ‘Advanced’→ ‘Time chart with multiple series’.



6. Open a new query screen.
7. Add the following query to get the top 10 common page flows for my users:

pageViews

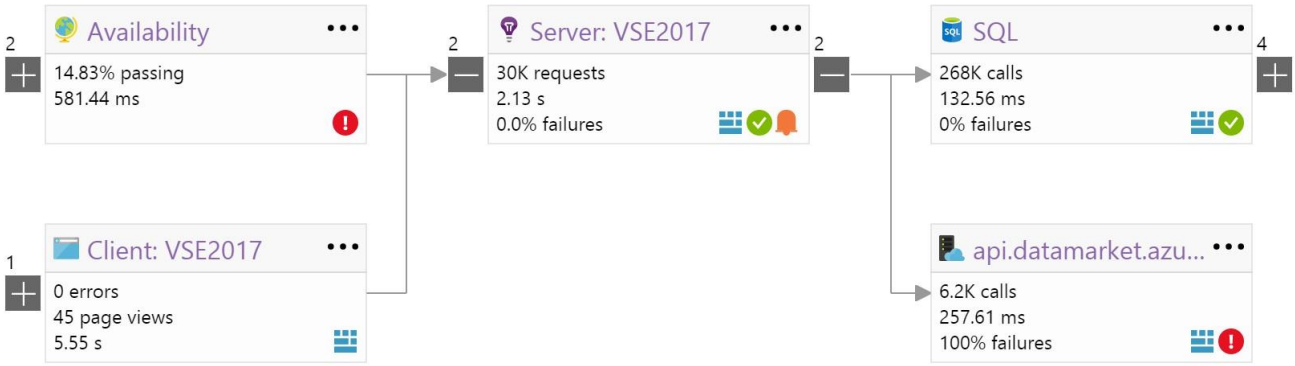
```
| WHERE TIMESTAMP >= AGO(1D)
| ORDER BY TIMESTAMP DESC
| SUMMARIZE PAGESVISITED=MAKELIST(NAME) BY SESSION_ID
| SUMMARIZE COUNT() BY TOSTRING(PAGESVISITED)
| TOP 20 BY COUNT_ DESC
```



Lab 9: Application Map

1. Go to the Azure Application Insights Overview Blade, click on the Application Map button to open the Application Map. Investigate the application Map. Dependencies are automatic capture, client/server side components, and SQL server components, along with status and basic performance metrics of each component.

2. Click on any red/orange warning symbols to open actual failure details to can click through and investigate.



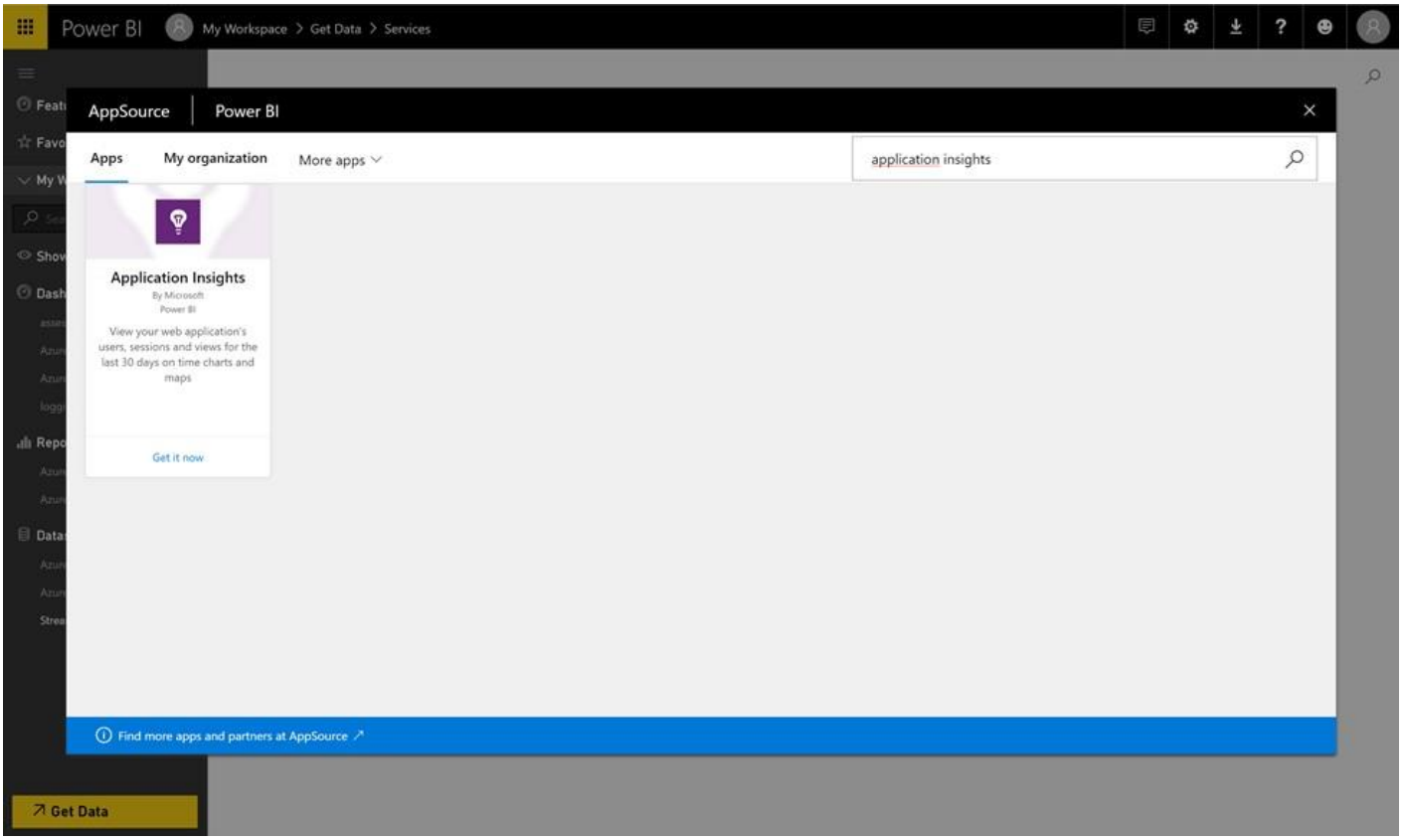
Scenario 4: Customize Reports and Add Custom Telemetry Data

The product owner and sales colleagues who are responsible for the PartsUnlimited website and sales want to access to the usages telemetry data so they can investigate the pages visited and make queries and reports themselves.

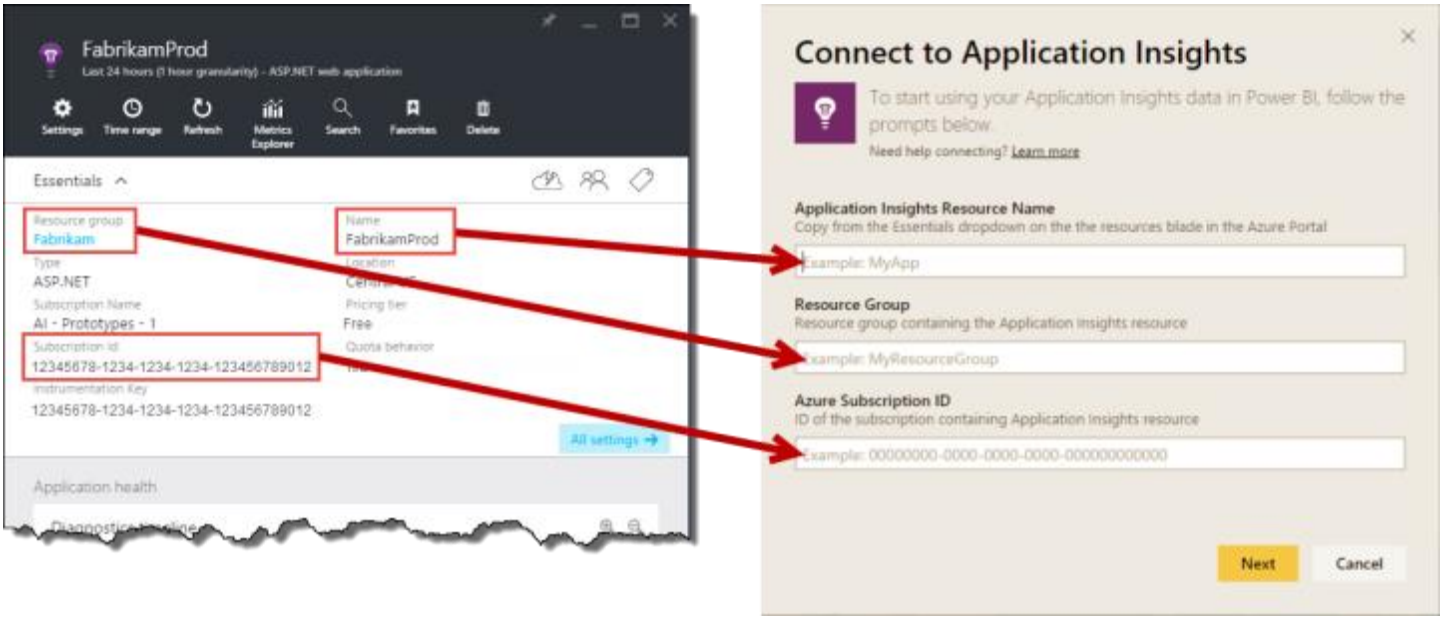
As a PartsUnlimited team member, enable Application Insights usages data to be available in PowerBI.

Lab 10: Telemetry Data in Power BI

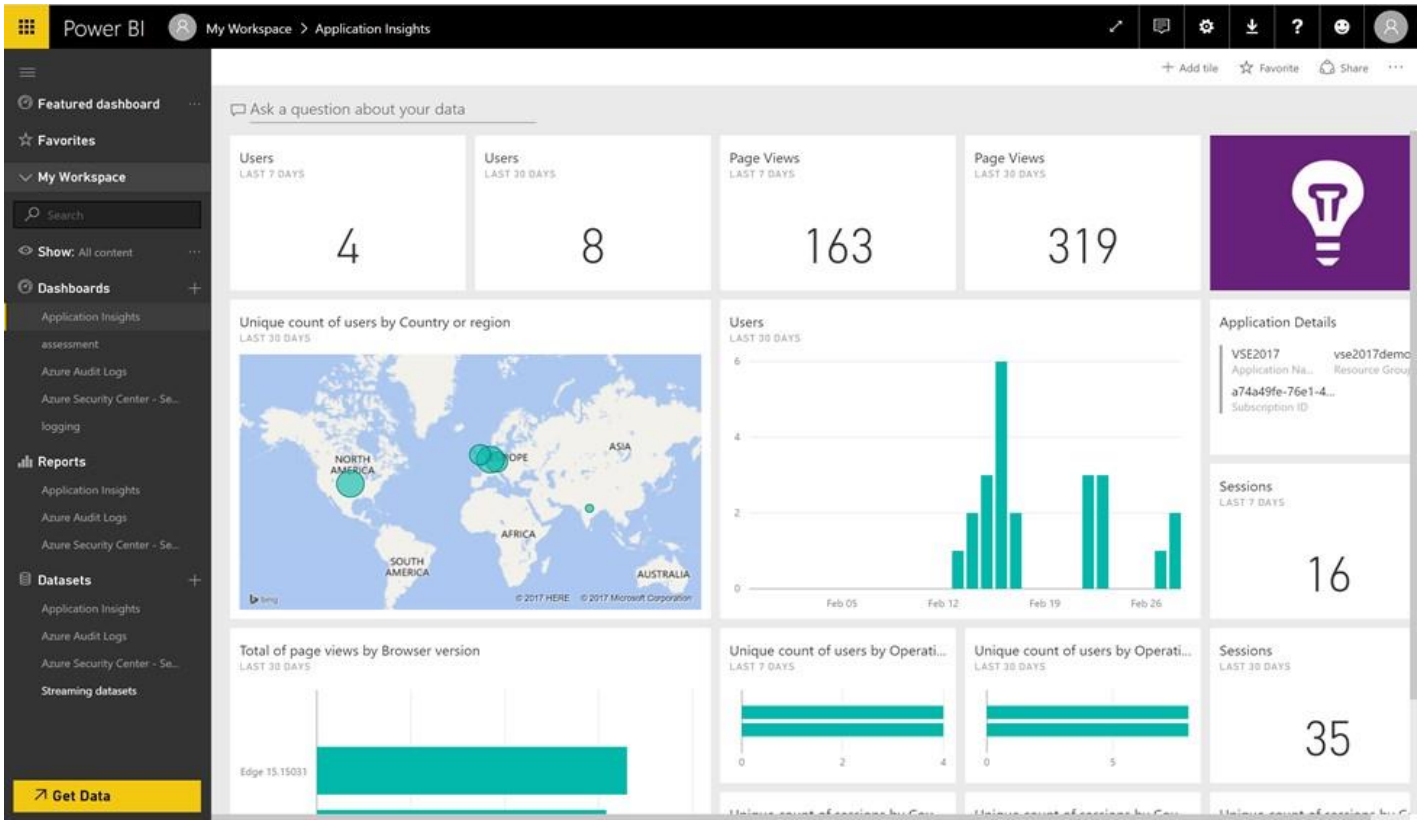
1. Sign up for PowerBI or sign in to an existing PowerBI instance. Follow the steps on: https://powerbi.microsoft.com/en-us/documentation/powerbi-service-self-service-signupfor-power-bi/ 2. Add PowerBI.



3. Authenticate PowerBI to access Application Insights.



4. Investigate the created dashboard.



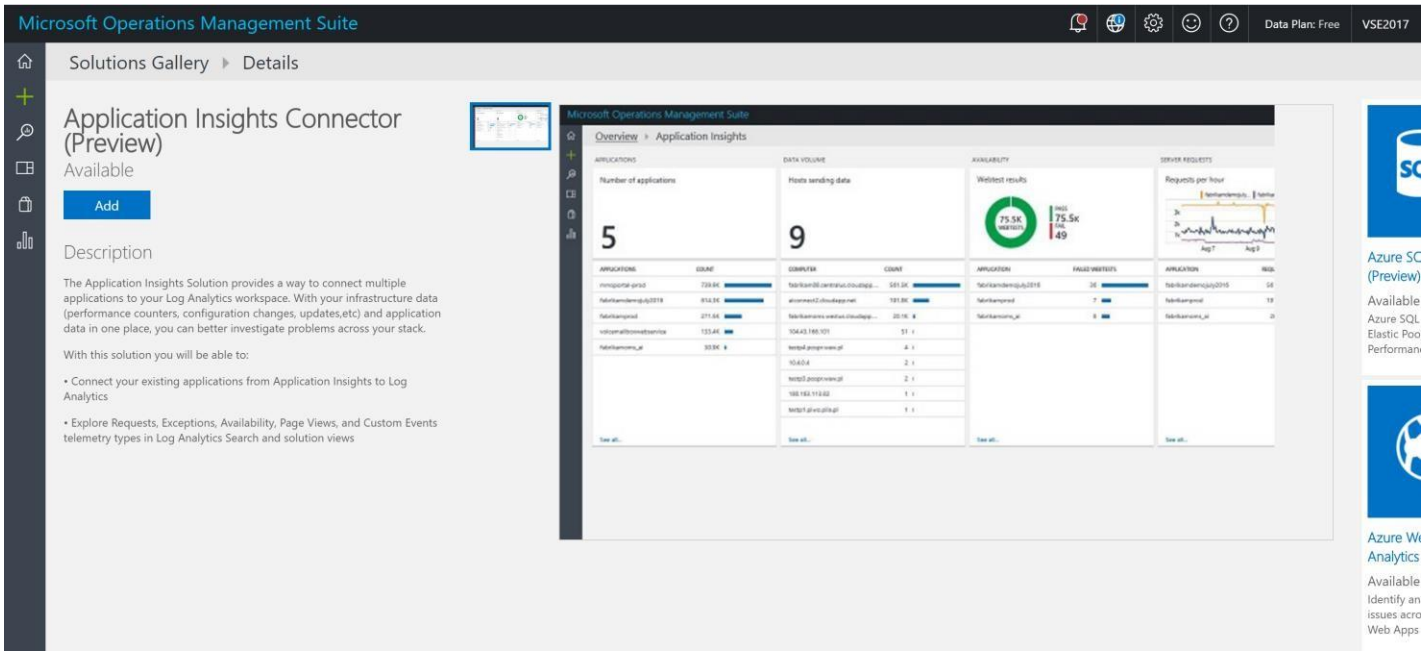
Scenario 5: OMS

Now that the sales and product owners of the PartsUnlimited website have their own view and query capabilities in their own tool, the DevOps team also wants to have their own dashboard with insights data, which is customizable for their own needs and within their own tool.

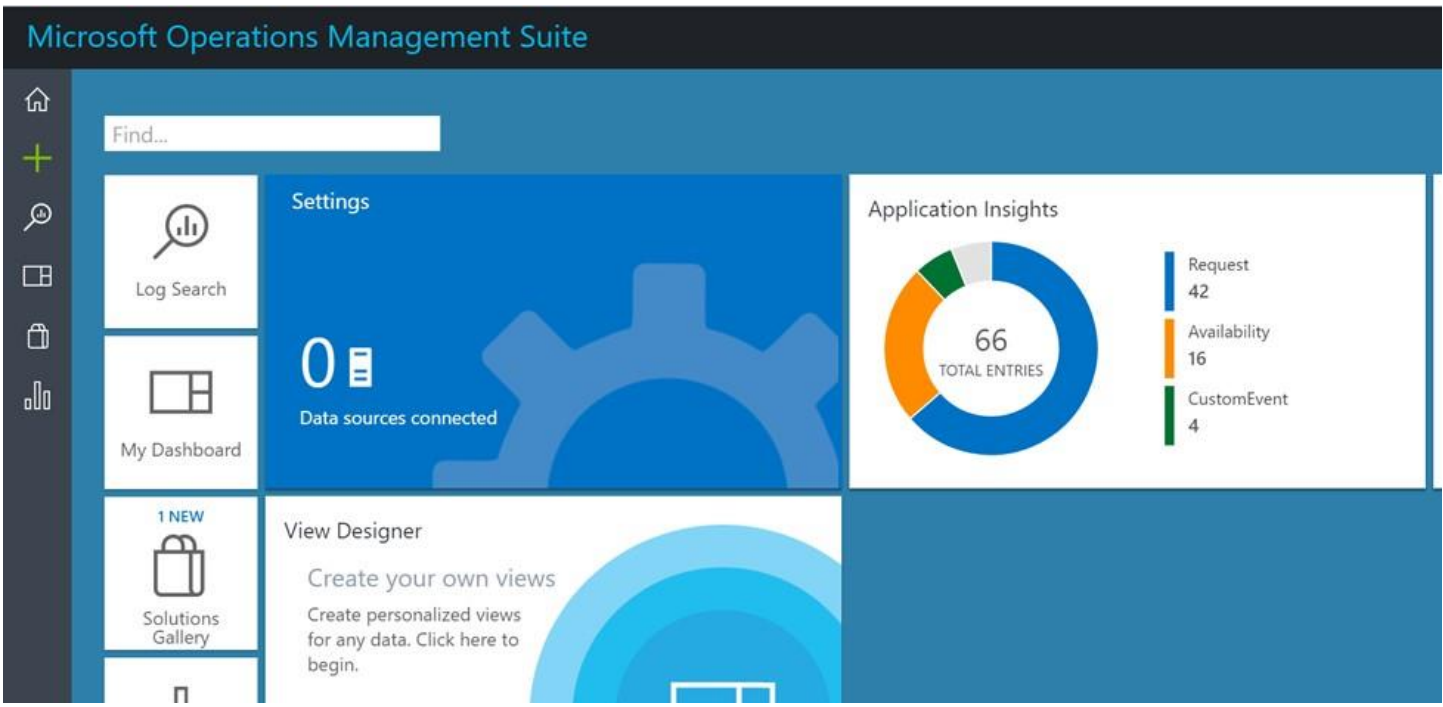
As a DevOps team member, configure Operations Management Suite to visualize Application Insights data from the PartsUnlimited website.

Lab 11: Setup and OMS with Application Insights

1. Setup and OMS Workspace.
    a) Navigate to http://microsoft.com/oms and click the Try for free button. Sign in with your Microsoft account such as Outlook.com, or with an Organizational account provided by your company or educational institution to use with Office 365 or other Microsoft services.
    b) Provide a unique Workspace Name. A workspace is a logical container where your management data is stored. It provides you a way to partition data between different teams in your organization, as the data is exclusive to its workspace. Specify an email address and the region where you would like to have your data reside.
    c) Next, you can create a new Azure subscription or link to an existing Azure subscription.
2. Add the Application Insights Collector to the OMS workspace.



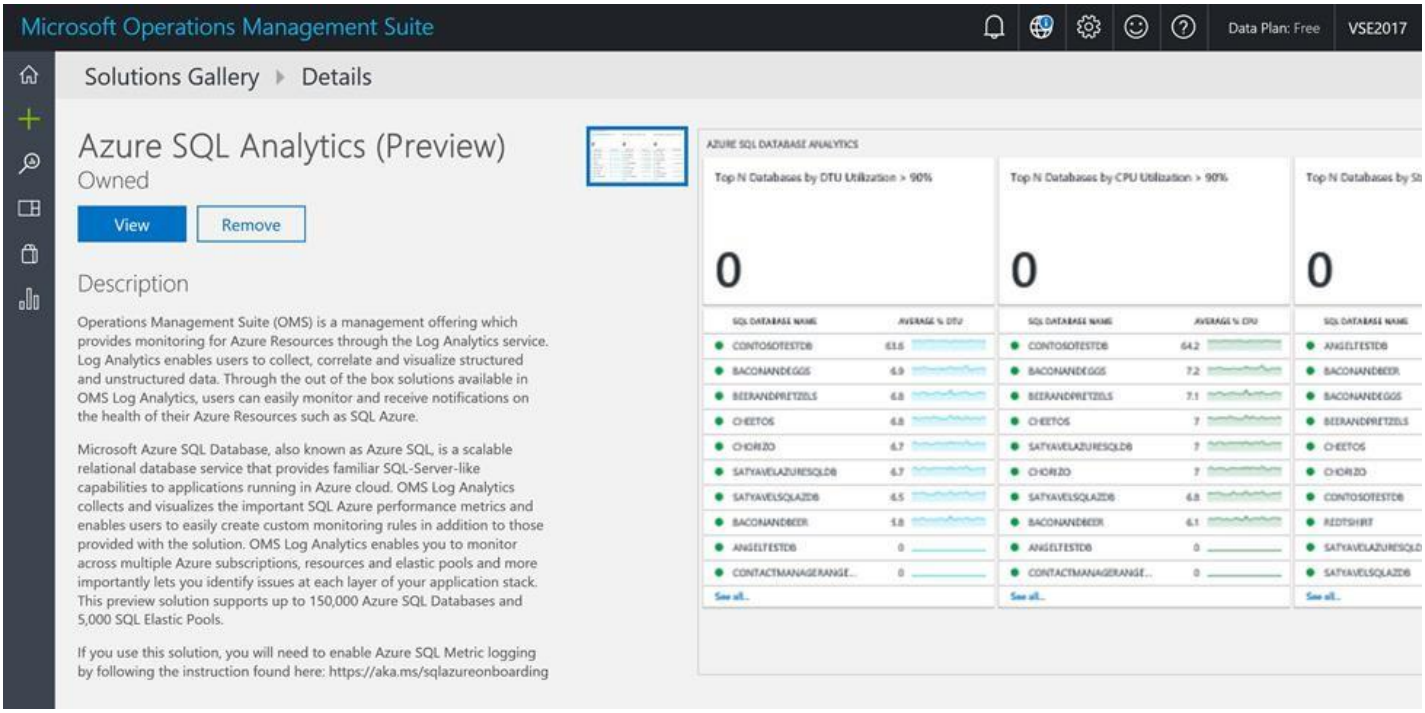
3. Configure the Application Insights Connector by linking the PartsUnlimitedWebsite Azure Resource.



4. Click on the Application Insights widget to see the charts and metrics right within OMS.

Lab 12: Add SQL Data to the OMS Workspace

1. Open the OMS Workspace created in Lab 11.
2. Add the Azure SQL Analytics solution to the workspace.



3. Configure OMS to monitor the PartsUnlimited SQL Server (See also steps in this blog).
    a) Install the latest set of AzureRM modules on your workstation: PS C:\> install-module -

Name AzureRM -Force

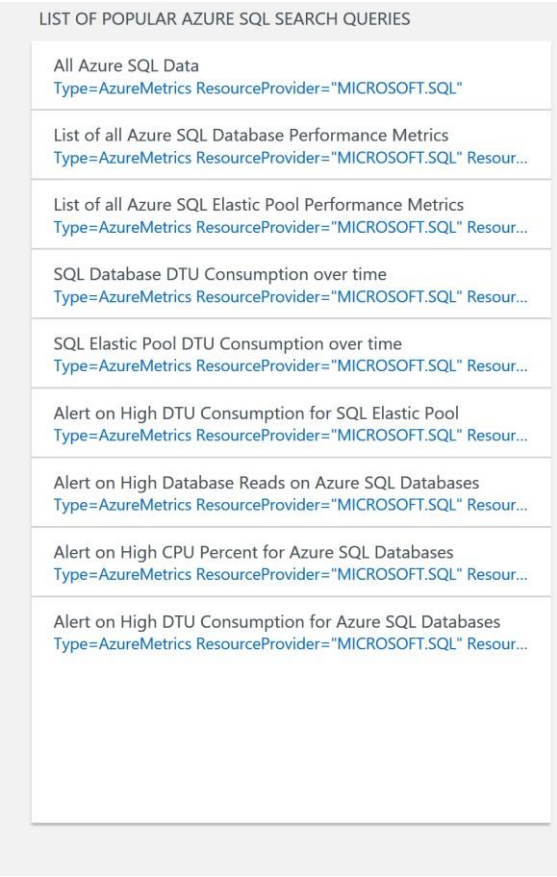
b) Open PowerShell in Administrator mode: Run as Administrator.

c) Save the Enable-AzureRMDiagnostics.ps1 script file locally, run the following command, and provide a path to store the script:
PS C:\> save-script -Name Enable-AzureRMDiagnostics -Path "C:\users\<username>\desktop\temp"

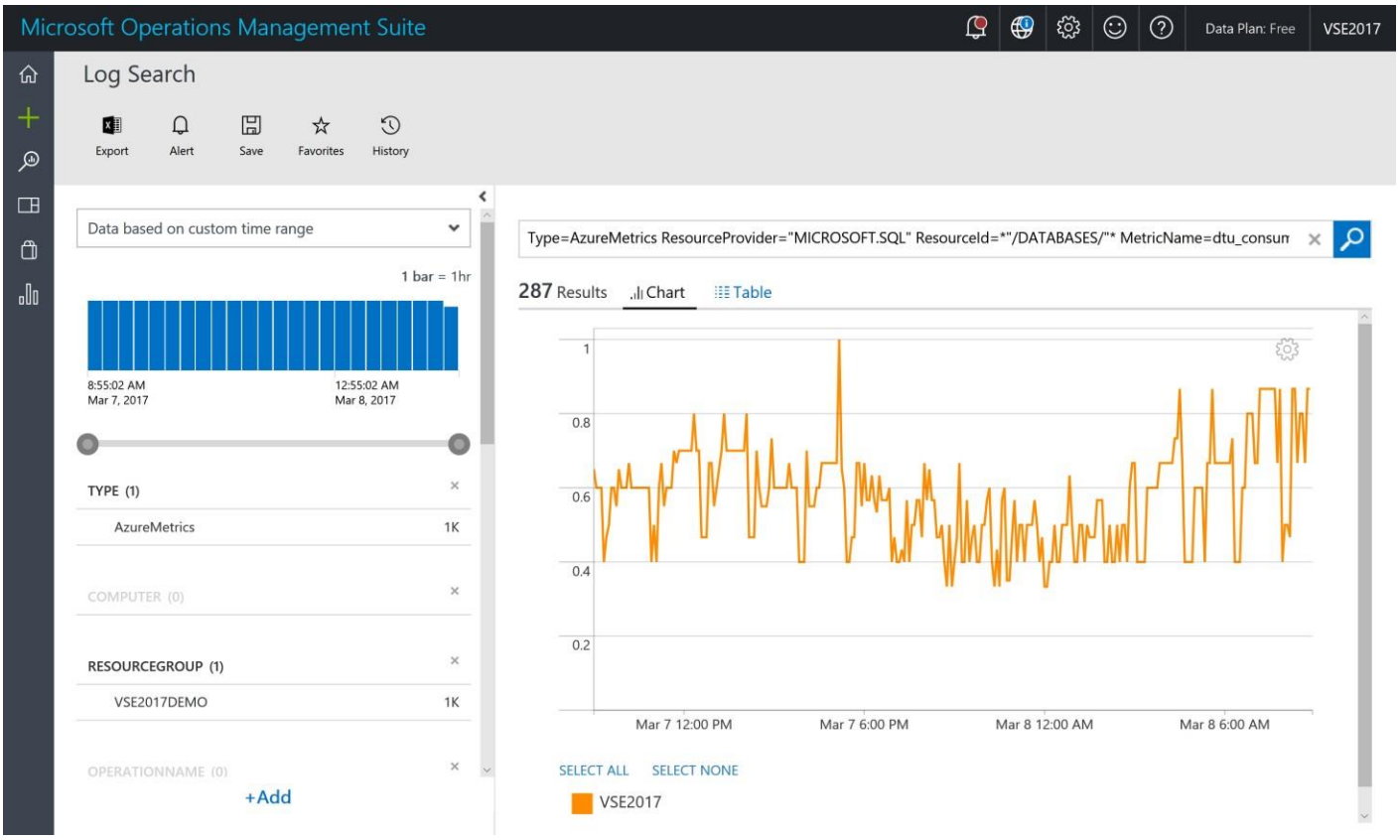
d) Go to the folder where you saved the script, and execute Enable-AzureRMDiagnostics.ps1. PS C:\users\<username>\Desktop\temp> .\Enable-AzureRMDiagnostics.ps1
Note: Narrow down the monitoring to PartsUnlimited with the resource group switch in the command -ResourceGroup " PartsUnlimited-RG"

4. Configure alerting on high DTU consumption.

a) Select, in the list of popular SQL queries, “Alert on High DTU Consumption for SQL Databases”



b) Select the Alert option to configure the alert.



c) Set the required fields.