Microsoft Azure Application Monitoring and Diagnostics Workshop*PLUS*

Student Lab Manual

Version 1.4

17 December 2019

Conditions and Terms of Use

Microsoft Confidential - For Internal Use Only

Contents

Lab 1: Application Insights for Web Apps

Introduction

Application Insights (AI) is an extensible analytics service that helps you understand the performance and usage of your live application. It is designed for developers to help you continuously improve the performance and usability of your app.

It works with both web and stand-alone apps on various platforms: .NET or J2EE, hosted on-premises or in the cloud, device apps on Windows, iOS, Android, Macintosh OS X, and other platforms. AI currently supports iOS, Android, and Windows apps, J2EE and ASP.NET web applications, Windows Communication Foundation (WCF) services. Web Apps can run on Microsoft Azure or your own on-premises servers. The AI JavaScript software development kit (SDK) can run on any web page. Application Insights works by adding an SDK into your app, which sends telemetry to the Azure Portal. There are different SDKs for the many combinations of platforms, languages, and IDEs that are supported.

This lab introduces you to AI with some simple instrumentation and show you how you can get to the information from the Azure Portal.

Objectives

After completing this lab, you will be able to:

- Set up Application Insights for a Web and API app
- Monitor your user and application performance

Prerequisites

- A Microsoft Azure subscription
- The Awesome Web App deployed on Azure Web App (completed in a previous lab)

Estimated Time to Complete This Lab

60 minutes

Scenario

Your Web App has been released and has many daily users. Now you need a way to gather telemetry data to prioritize feature development and ensure your team has a user centric approach. Furthermore, you also need to have insight when something goes wrong to quickly detect and troubleshoot issues. To achieve that goal, you will need KPI and reliable data.

Exercise 1: Creating and Exploring Application Insights

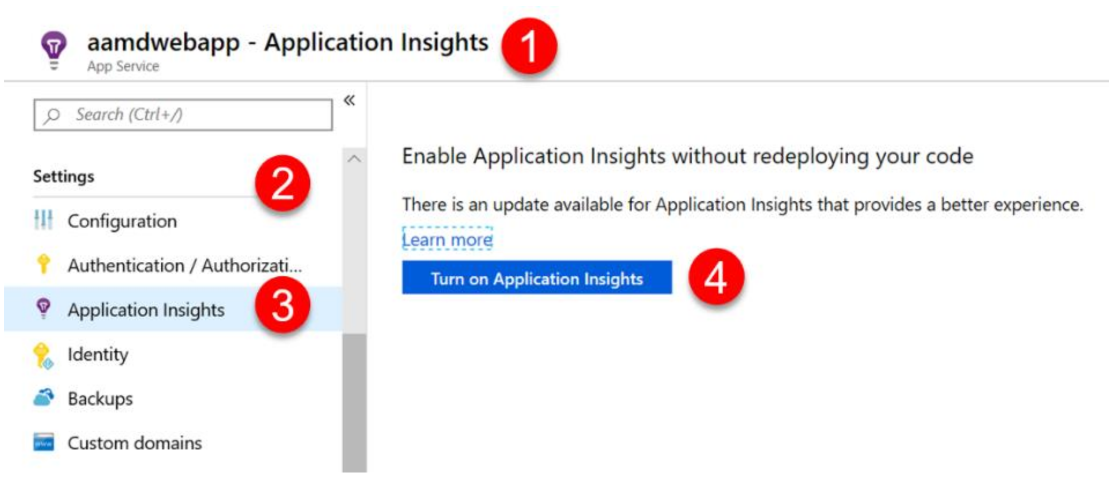
Objectives

After completing this exercise, you will be able to:

- Create an Application Insights resource

Task 1: Creating an Application Insights resource

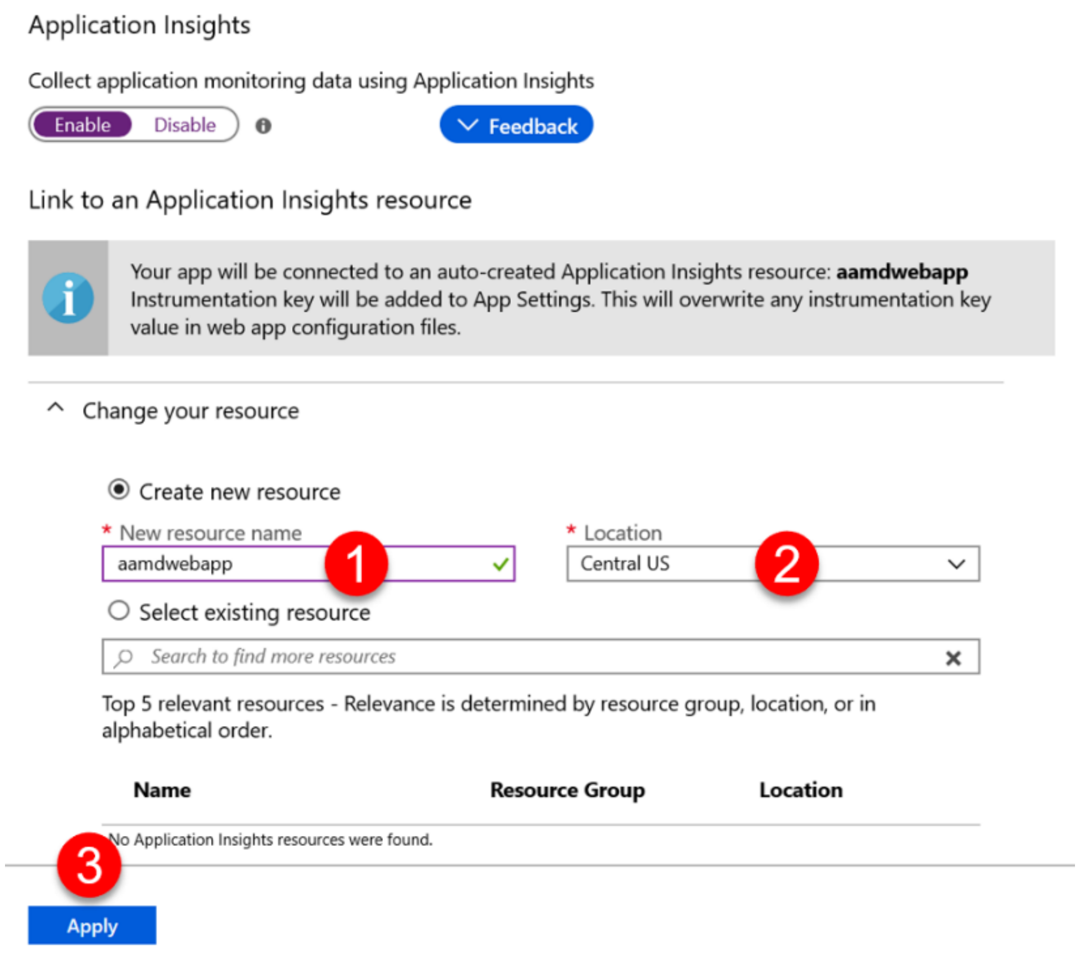
1. Log on to the Azure Portal
2. Navigate to the resource group **AAMDWEBAPP** created earlier
3. Select the **Web App** that is hosting **Awesome Web App** created earlier
4. Select **Application Insights** under **Settings** and click **Turn on Application Insights**



Enable Application Insights without redeploying your code

There is an update available for Application Insights that provides a better experience.

5. Create a new AI resource as shown leaving other values in their default setting and click **Apply**


Application Insights

Collect application monitoring data using Application Insights

Link to an Application Insights resource

Your app will be connected to an auto-created Application Insights resource: **aamdwebapp** Instrumentation key will be added to App Settings. This will overwrite any instrumentation key value in web app configuration files.

Change your resource

[✓ Create new resource]

* New resource name: aamdwebapp

* Location: Central US

Select existing resource

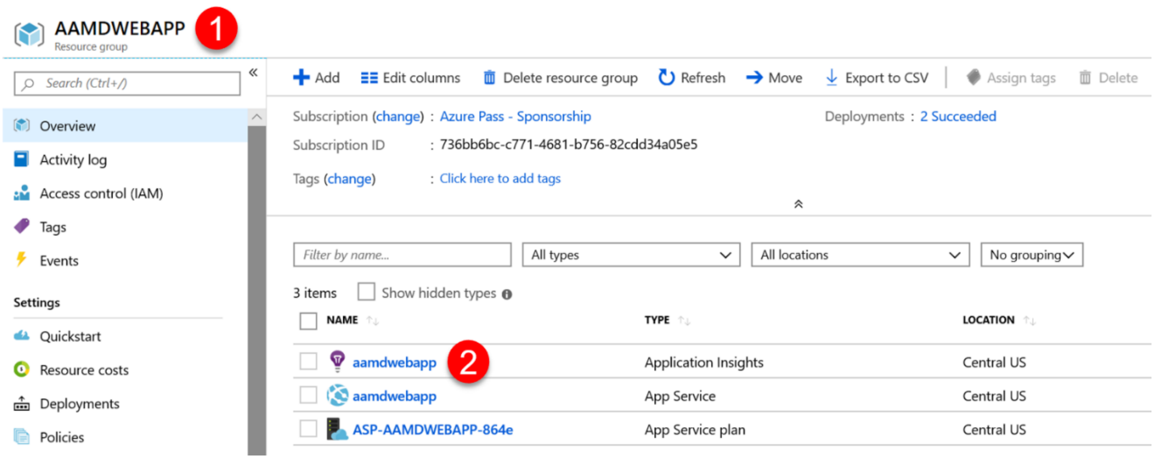
Search to find more resources

Top 5 relevant resources - Relevance is determined by resource group, location, or in alphabetical order.

Name	Resource Group	Location
No Application Insights resources were found.		

Task 2: Configuring Application Insights for Server Data

1. In the Azure Portal, open the **Application Insights** resource associated with your **Web App**. This can be reached from the **AAMDWEBAPP** resource group.



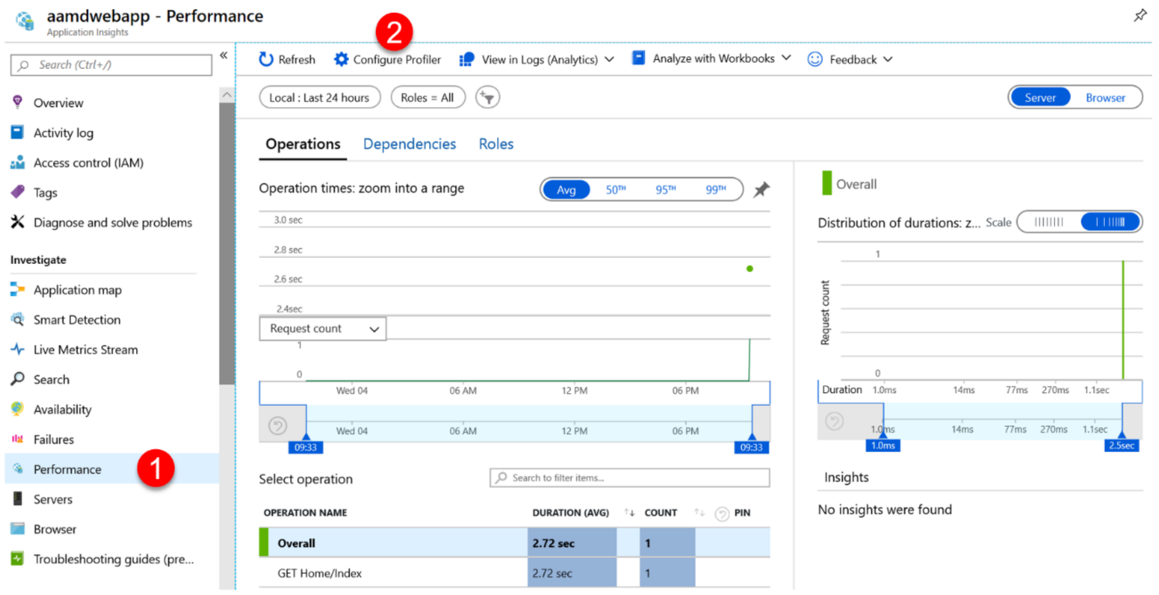
Subscription (change) : Azure Pass - Sponsorship
Deployments : 2 Succeeded
Subscription ID : 736bb6bc-c771-4681-b756-82cdd34a05e5
Tags (change) : Click here to add tags

3 items

NAME	TYPE	LOCATION
aamdwebapp	Application Insights	Central US
aamdwebapp	App Service	Central US
ASP-AAMDWEBAPP-864e	App Service plan	Central US

2. Click **Performance** under the **Investigate** section

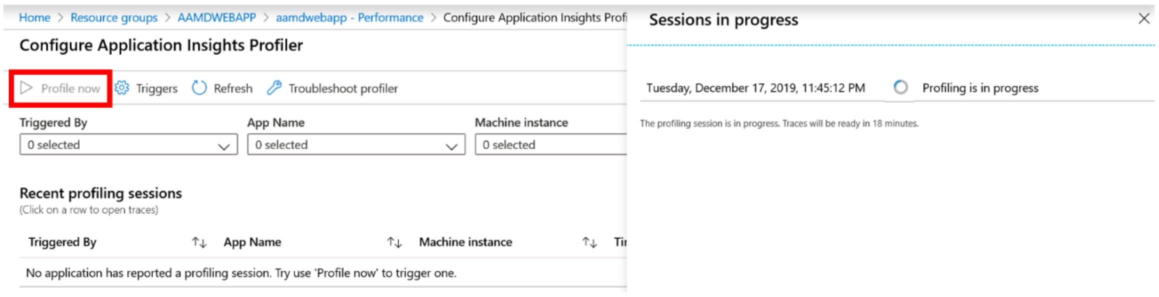
3. Click on **Configure Profiler**



OPERATION NAME	DURATION (AVG)	COUNT	PIN
Overall	2.72 sec	1	
GET Home/Index	2.72 sec	1	

No insights were found

4. Click **Profile now**


Home > Resource groups > AAMDWEBAPP > aamdwebapp - Performance > Configure Application Insights Profi

Configure Application Insights Profiler

Triggered By: 0 selected
App Name: 0 selected
Machine instance: 0 selected

Recent profiling sessions
(Click on a row to open traces)

Triggered By	App Name	Machine instance	Ti
No application has reported a profiling session. Try use 'Profile now' to trigger one.			

Sessions in progress

Tuesday, December 17, 2019, 11:45:12 PM — Profiling is in progress

The profiling session is in progress. Traces will be ready in 18 minutes.

Task 3: Configuring Application Insights for Browser Data

1. Copy the following code snippet. This script enables the collection of client-side telemetry data. See this reference for more information

```
<!--
To collect user behavior analytics about your application,
insert the following script into each page you want to track.
Place this code immediately before the closing </head> tag,
and before any other scripts. Your first data will appear
automatically in just a few seconds.
-->
<script type="text/javascript">
  var appInsights=window.appInsights||function(config){
    function i(config){t[config]=function(){var
i=arguments;t.queue.push(function(){t[config].apply(t,i)})}}var
t={config:config},u=document,e=window,o="script",s="AuthenticatedUserContext",h
="start",c="stop",l="Track",a=l+"Event",v=l+"Page",y=u.createElement(o),r,f;y.s
rc=config.url||"https://az416426.vo.msecnd.net/scripts/a/ai.0.js";u.getElements
ByTagName(o)[0].parentNode.appendChild(y);try{t.cookie=u.cookie}catch(p){}for(t
.queue=[],t.version="1.0",r=["Event","Exception","Metric","PageView","Trace","D
ependency"];r.length;)i("track"+r.pop());return
i("set"+s),i("clear"+s),i(h+a),i(c+a),i(h+v),i(c+v),i("flush"),config.disableEx
ceptionTracking||(r="onerror",i("_"+r),f=e[r],e[r]=function(config,i,u,e,o){var
s=f&&f(config,i,u,e,o);return s!==!0&&t["_"+r](config,i,u,e,o),s}),t
    }({
        instrumentationKey:"<insert instrumentation key>"
    });

    window.appInsights=appInsights;
    appInsights.trackPageView();
</script>
```

2. Navigate to the **Web App** resource within the Azure Portal

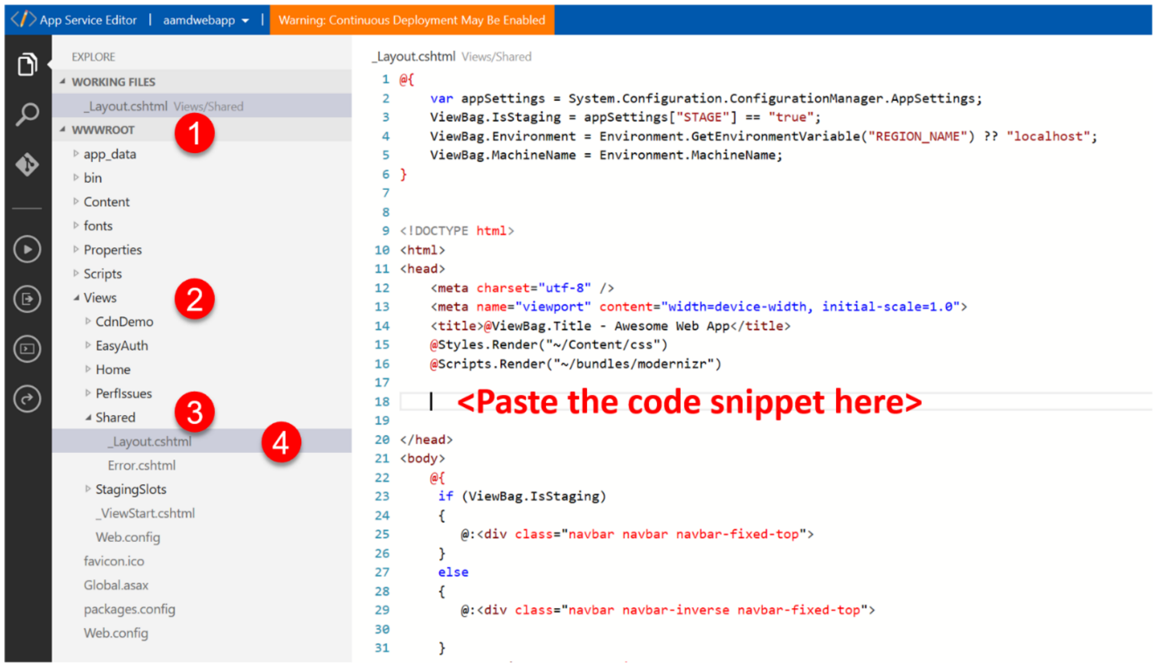
3. Select **App Service Editor (Preview)** under the **Development Tools** section

4. Click **Go**

5. This will open **App Service Editor** in a separate tab

6. In the **Explore** section, navigate to **WWWROOT → Views → Shared → _Layout.cshtml**

7. Paste the code snippet copied from step 1 before the **</head>** tag.



```
@{
    var appSettings = System.Configuration.ConfigurationManager.AppSettings;
    ViewBag.IsStaging = appSettings["STAGE"] == "true";
    ViewBag.Environment = Environment.GetEnvironmentVariable("REGION_NAME") ?? "localhost";
    ViewBag.MachineName = Environment.MachineName;
}


<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Awesome Web App</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

    |  <Paste the code snippet here>

</head>
<body>
    @{
     if (ViewBag.IsStaging)
     {
        @:<div class="navbar navbar navbar-fixed-top">
     }
     else
     {
        @:<div class="navbar navbar-inverse navbar-fixed-top">

     }
```

8. Copy the **Intrumentation Key** from the Application Insights **Overview** blade. Replace the string **<insert instrumentation key>** with the Instrumentation Key

9. Your changes are saved automatically

Task 4: Generating Load to review the data

1. Create a folder named **Scripts** under **C:** drive (C:\Scripts)

2. Copy **tinyget.exe** from current lab folder to **C:\Scripts\** folder.

3. Copy below content and save it in a file named **GenerateRequests.ps1**

   ***Note**: You can also use the file provided with the lab.*

```
# Script to send multiple requests to the cloud service deployed on Azure
# When executed, it will send multiple request to the server simulating load.
# It will prompt for Cloud Service Name (e.g. yourapp.azurewebsites.net - note
#this is without http://)
# It will prompt for no. of threads to use (for e.g 10, 20, 50)
# It will prompt for no. of times to loop (for e.g. 100, 200, 500)
# It will start TineyGet instances and send multiple requests to different pages
#of the application

function Get-ScriptDirectory
{
$Invocation = (Get-Variable MyInvocation -Scope 1).Value
Split-Path $Invocation.MyCommand.Path
}

$ScriptDirectory = Split-Path -Path $MyInvocation.MyCommand.Definition -Parent
$program = Join-Path $ScriptDirectory “tinyget.exe"
$argServerURL = "-server:" + (Read-Host 'Please input the Web App URL')
$argThreads = "-threads:" + (Read-Host 'Please input no. of threads to use (for
e.g 10, 20, 50)')
$argLoop = "-loop:" + (Read-Host 'Please input no. of times to loop (for e.g.
100, 200, 500)')

$uriDefault = '/Home/Index'
$uriMemory = '/PerfIssues/MemLeak'
$uriHighCPU = '/PerfIssues/SpinCpuBtnHandler'
$uriException = '/ProduceError/500'
$uriLongRunning='/PerfIssues/SleepBtnHandler'
$uriDiagnosticLogging = '/Home/ProduceLogs'

$argPageURL = "-uri:$uriDiagnosticLogging"
$arguments = "$argServerURL $argPageURL  $argThreads $argLoop"
Write-Host "Started Requests for" $uriDiagnosticLogging
Start-Process $program -ArgumentList $arguments -WindowStyle Hidden

$argPageURL = "-uri:$uriDefault"
$arguments = "$argServerURL $argPageURL  $argThreads $argLoop"
Write-Host "Started Requests for" $uriDefault $arguments
Start-Process $program -ArgumentList $arguments -WindowStyle Hidden

$argPageURL = "-uri:$uriException"
$arguments = "$argServerURL $argPageURL  $argThreads $argLoop"
Write-Host "Started Requests for" $uriException $arguments
Start-Process $program -ArgumentList $arguments -WindowStyle Hidden
```

```
$argPageURL = "-uri:$uriLongRunning"
$arguments = "$argServerURL $argPageURL  $argThreads $argLoop"
Write-Host "Started Requests for" $uriLongRunning $arguments
Start-Process $program -ArgumentList $arguments -WindowStyle Hidden

$argPageURL = "-uri:$uriHighCPU"
$arguments = "$argServerURL $argPageURL  $argThreads $argLoop"
Write-Host "Started Requests for" $uriHighCPU $arguments
Start-Process $program -ArgumentList $arguments -WindowStyle Hidden
```

4. Start the **PowerShell** console and navigate to **C:\ Scripts**

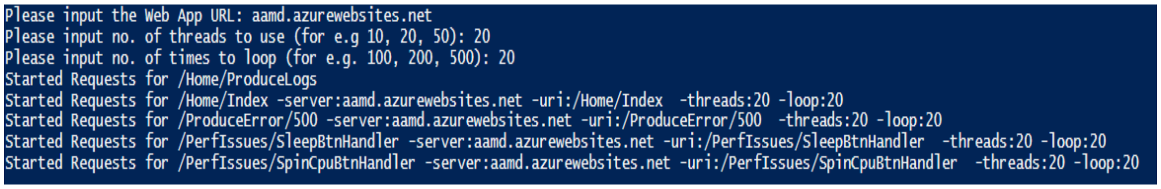
5. Execute the following **PowerShell** command:

   Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
   (Note: This command relaxes the security on running PowerShell scripts that are not digitally signed)

6. Type **Y** and press **Enter** to proceed to change the execution policy

7. Execute **.\GenerateRequests.ps1** and enter the parameters **(Note: The Web App URL should be without "http://")**

8. This will initiate 100s of request to our web app

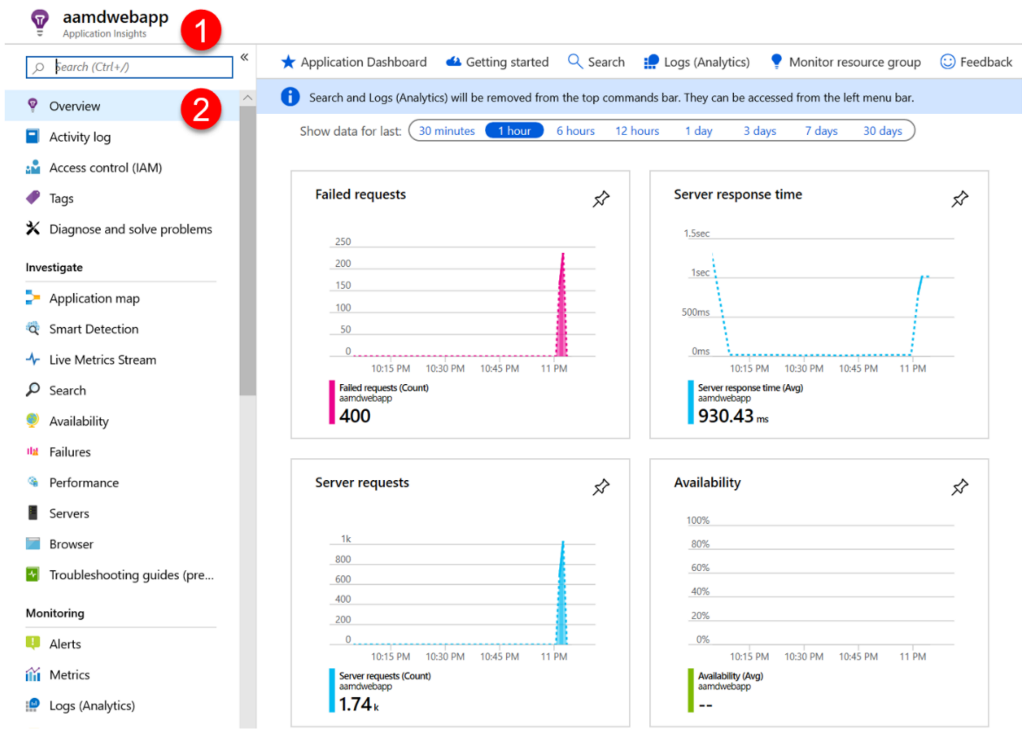


```
Please input the Web App URL: aamd.azurewebsites.net
Please input no. of threads to use (for e.g 10, 20, 50): 20
Please input no. of times to loop (for e.g. 100, 200, 500): 20
Started Requests for /Home/ProduceLogs
Started Requests for /Home/Index -server:aamd.azurewebsites.net -uri:/Home/Index  -threads:20 -loop:20
Started Requests for /ProduceError/500 -server:aamd.azurewebsites.net -uri:/ProduceError/500  -threads:20 -loop:20
Started Requests for /PerfIssues/SleepBtnHandler -server:aamd.azurewebsites.net -uri:/PerfIssues/SleepBtnHandler  -threads:20 -loop:20
Started Requests for /PerfIssues/SpinCpuBtnHandler -server:aamd.azurewebsites.net -uri:/PerfIssues/SpinCpuBtnHandler  -threads:20 -loop:20
```

9. This can be repeated as many times as needed during the exercise to observe the data points in **Application Insights**

Task 5: Exploring Application Insights

5. In the Azure Portal, navigate to the **Application Insights** resource associated with your **Web App**
6. In the **Overview** blade, you can see the overall health and request summary



Search and Logs (Analytics) will be removed from the top commands bar. They can be accessed from the left menu bar.

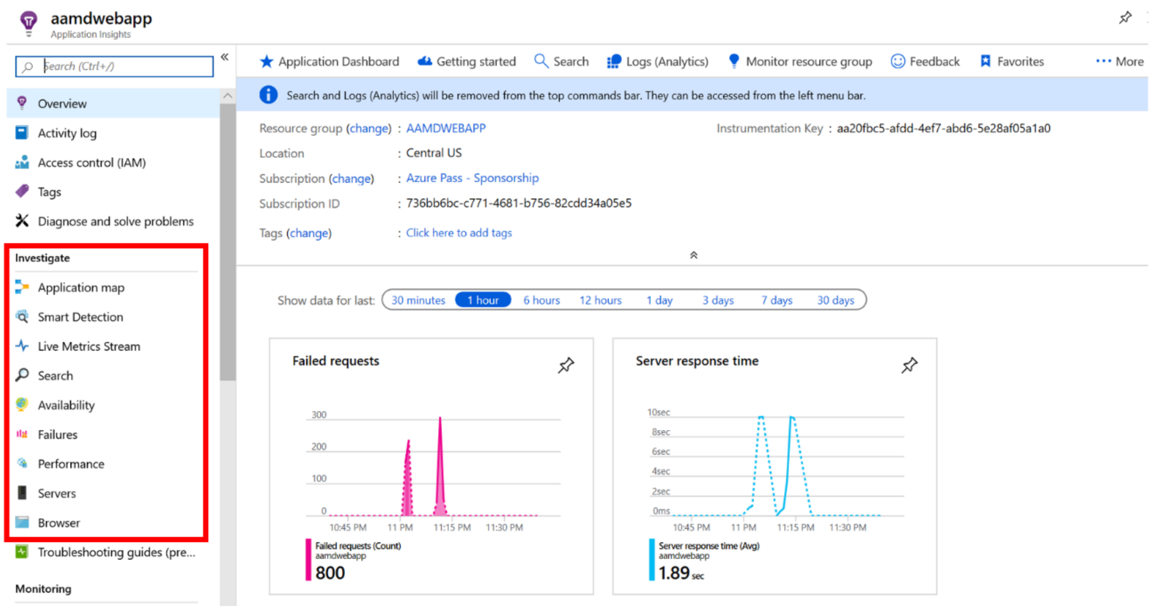
Failed requests (Count) aamdwebapp 400

Server response time (Avg) aamdwebapp 930.43 ms

Server requests (Count) aamdwebapp 1.74 k

Availability (Avg) aamdwebapp --

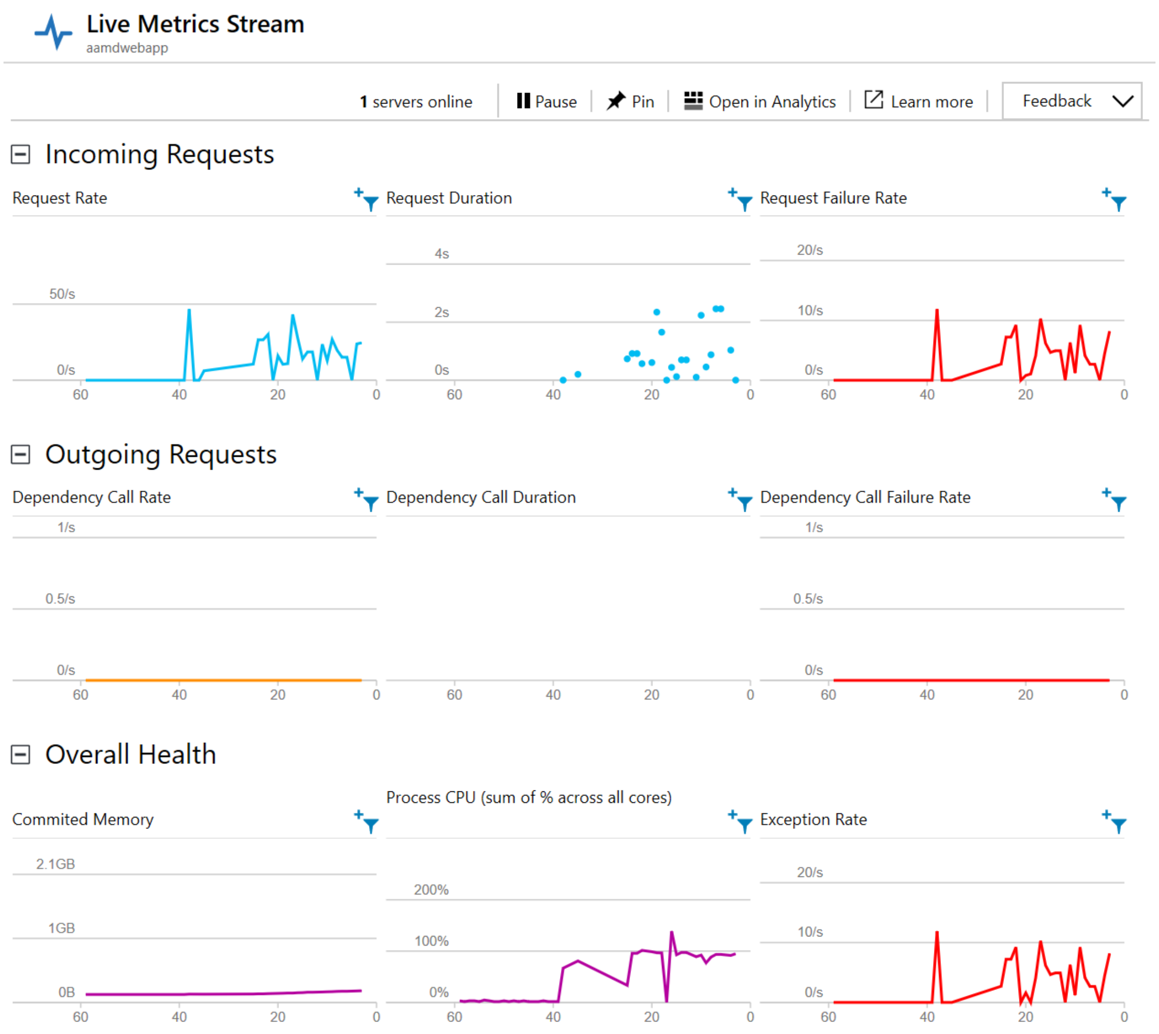
Over the next steps, you will see features useful for investigating issues and usage



Search and Logs (Analytics) will be removed from the top commands bar. They can be accessed from the left menu bar.

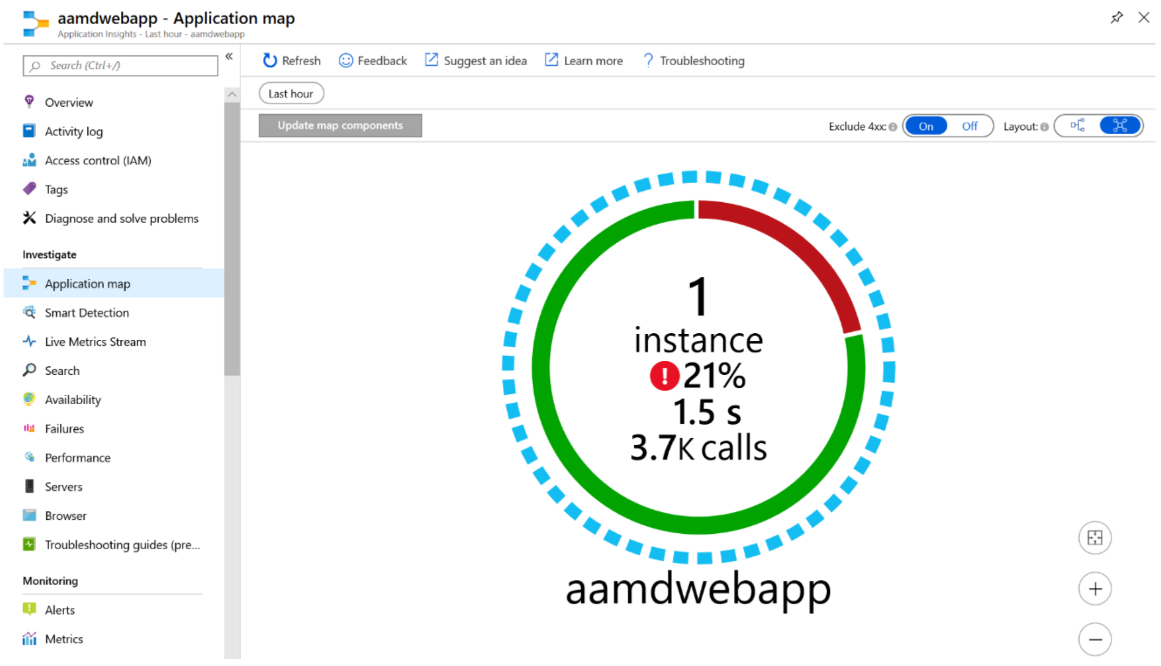
Resource group (change)	: AAMDWEBAPP
Location	: Central US
Subscription (change)	: Azure Pass - Sponsorship
Subscription ID	: 736bb6bc-c771-4681-b756-82cdd34a05e5
Tags (change)	: Click here to add tags
Instrumentation Key	: aa20fbc5-afdd-4ef7-abd6-5e28af05a1a0

7. Click on **Live Metrics Stream**. If **GenerateRequests.ps1** is still executing, you will see movement in the telemetry



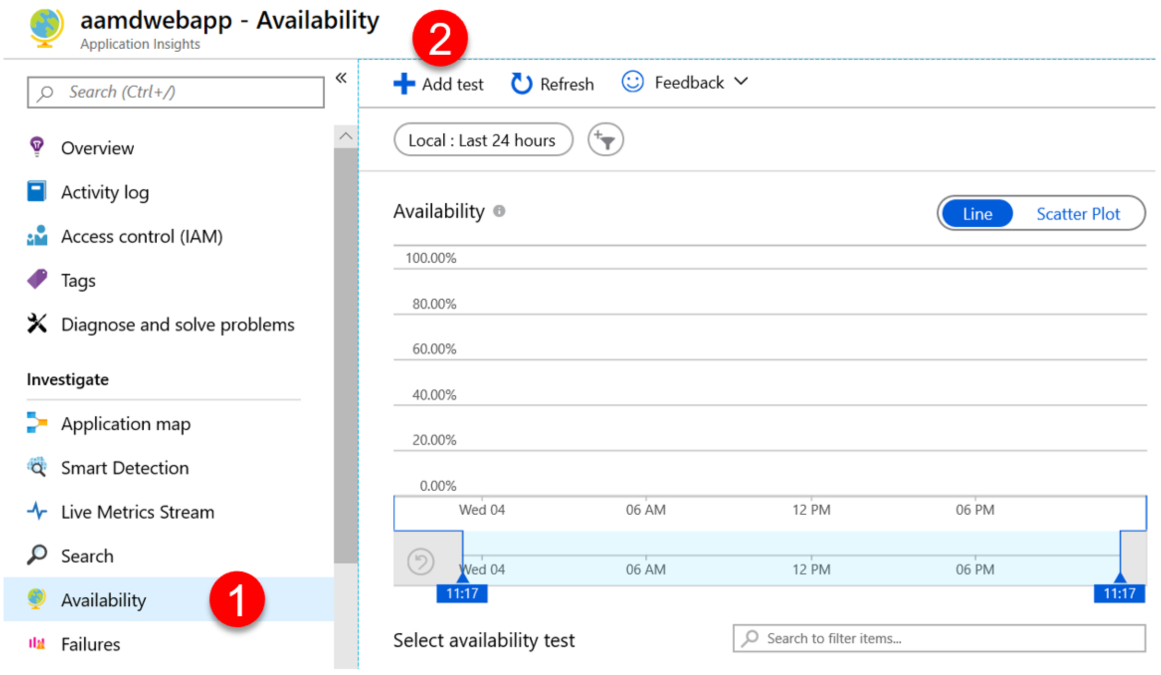
8. Explore individual metrics, filter, etc.

9. Click on **Application map** to look at the application components. Explore the features

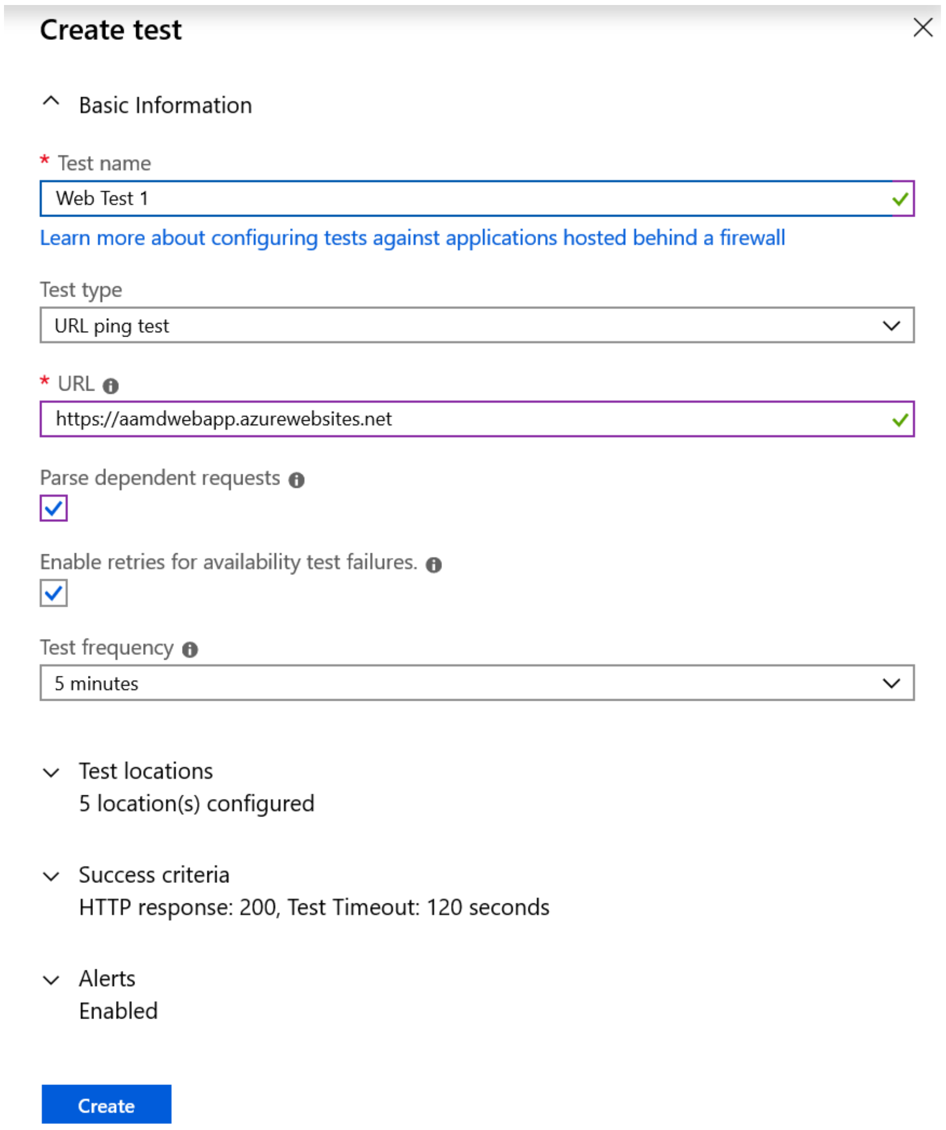


10. Click on **Availability**. Here you can configure availability tests for your application from different geographical locations around the world

11. Click **Add test** to configure tests as per your requirement



Here's an example of a test:



Create test

Basic Information

* Test name

Web Test 1

Learn more about configuring tests against applications hosted behind a firewall

Test type

URL ping test

* URL

https://aamdwebapp.azurewebsites.net

Parse dependent requests

☑

Enable retries for availability test failures.

☑

Test frequency

5 minutes

Test locations

5 location(s) configured

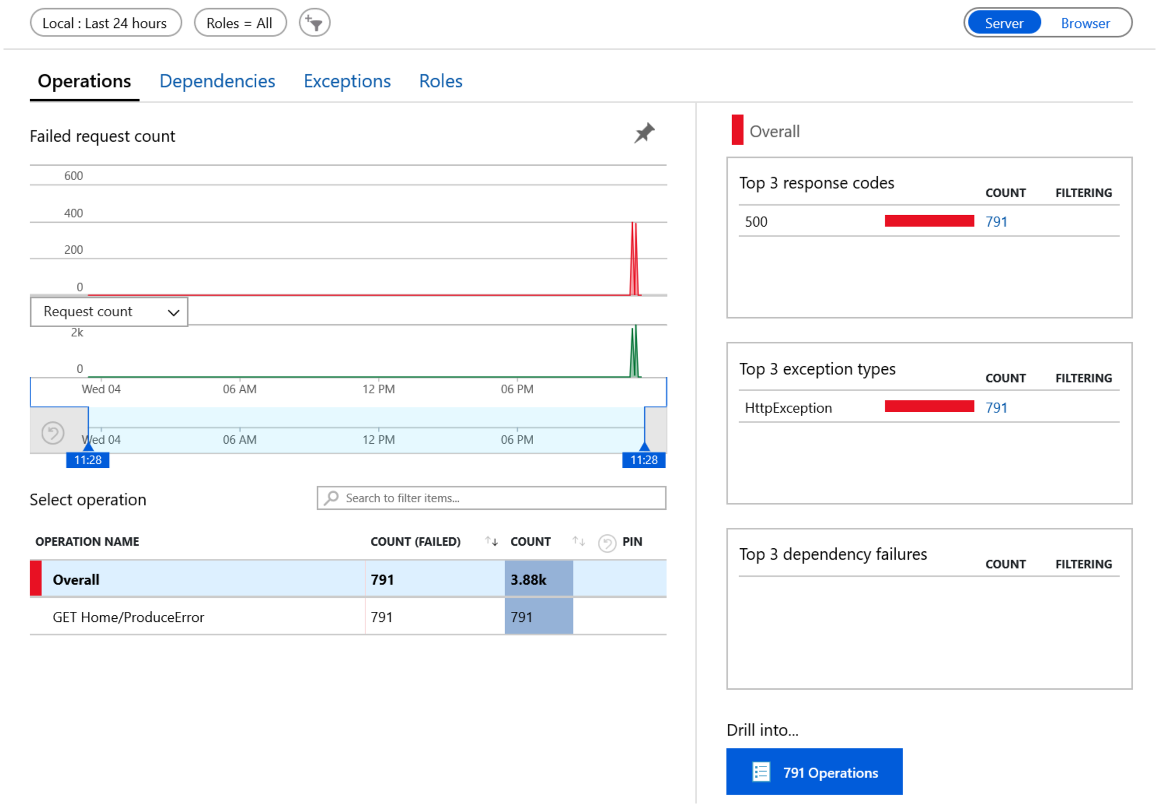
Success criteria

HTTP response: 200, Test Timeout: 120 seconds

Alerts

Enabled

12. Click **Failures.** Here you can see all the requests that have failed. You can find more information about failed requests by clicking on them



OPERATION NAME	COUNT (FAILED)	COUNT	PIN
Overall	791	3.88k	
GET Home/ProduceError	791	791	

Top 3 response codes

	COUNT	FILTERING
500	791	

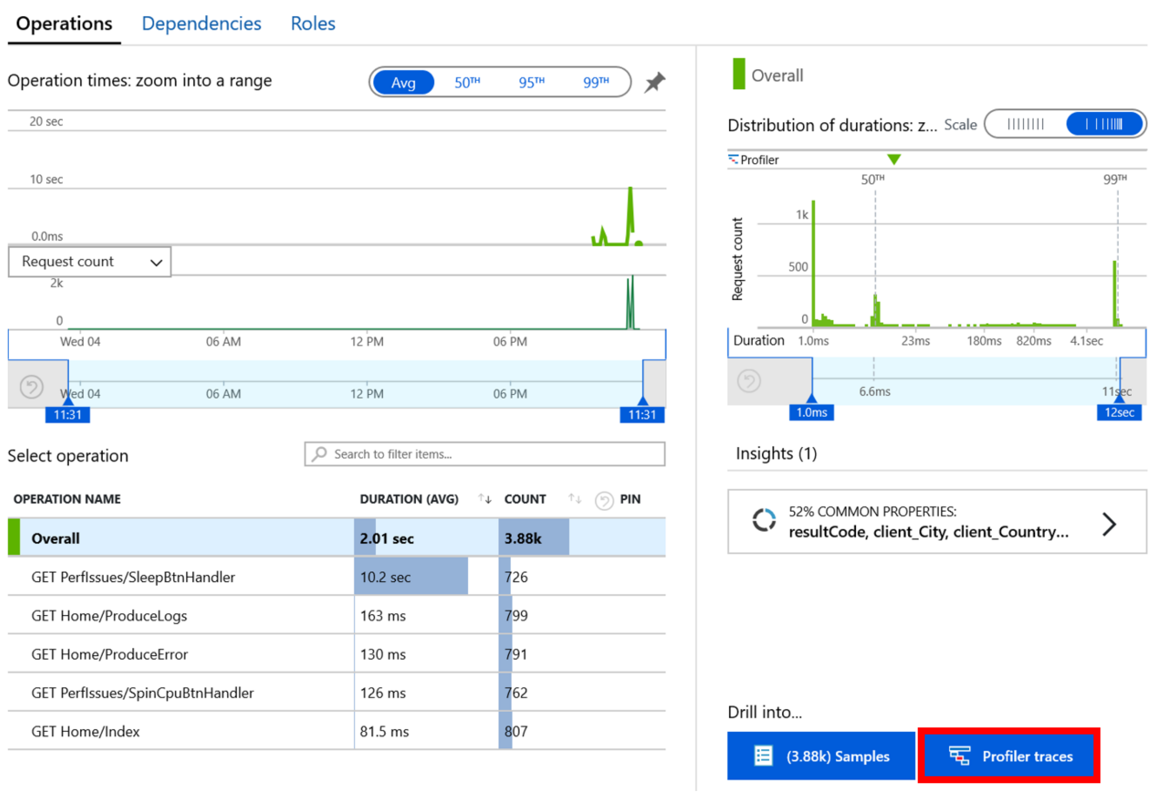
Top 3 exception types

	COUNT	FILTERING
HttpException	791	

Top 3 dependency failures

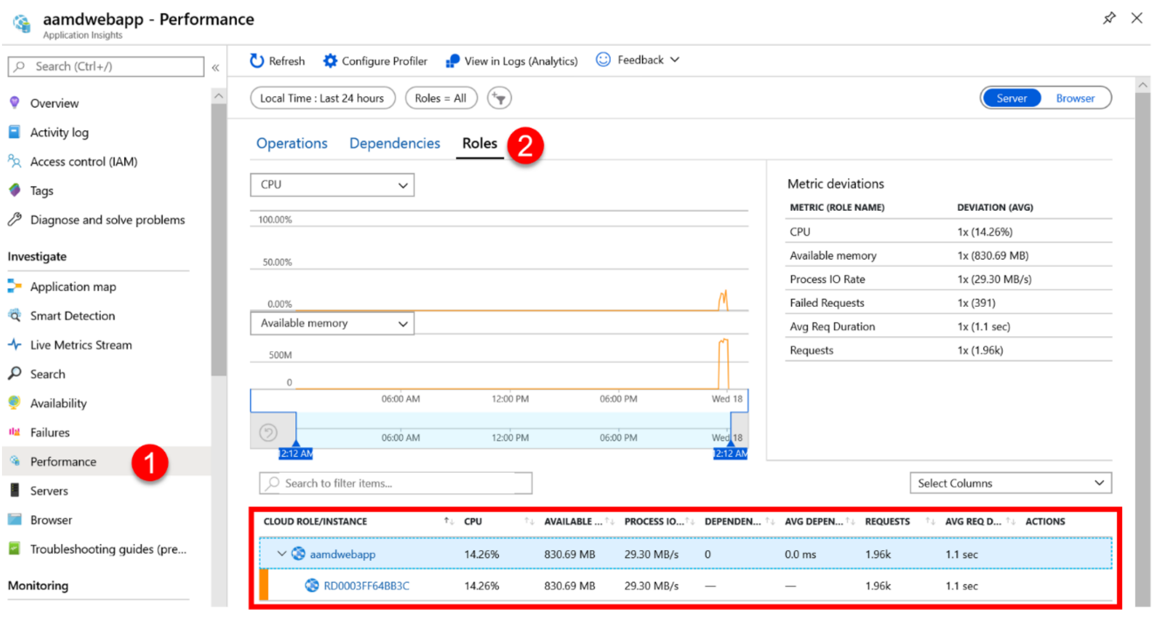
	COUNT	FILTERING

13. Click on **Performance**. Here you will obtain insights about the application's performance
14. Click on **Profiler traces** to obtain more detail about requests and their execution



OPERATION NAME	DURATION (AVG)	COUNT	PIN
Overall	2.01 sec	3.88k	
GET PerfIssues/SleepBtnHandler	10.2 sec	726	
GET Home/ProduceLogs	163 ms	799	
GET Home/ProduceError	130 ms	791	
GET PerfIssues/SpinCpuBtnHandler	126 ms	762	
GET Home/Index	81.5 ms	807	

15. Go back to the **Performance** blade and click **Roles** to explore the metrics against each instance

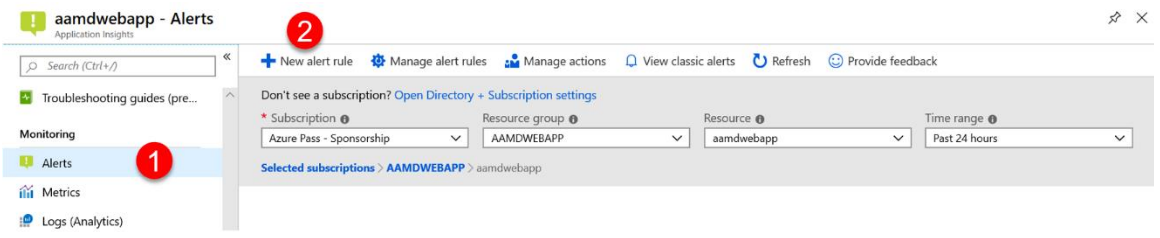


Metric deviations

METRIC (ROLE NAME)	DEVIATION (AVG)
CPU	1x (14.26%)
Available memory	1x (830.69 MB)
Process IO Rate	1x (29.30 MB/s)
Failed Requests	1x (391)
Avg Req Duration	1x (1.1 sec)
Requests	1x (1.96k)

CLOUD ROLE/INSTANCE	CPU	AVAILABLE ...	PROCESS IO...	DEPENDEN...	AVG DEPEN...	REQUESTS	AVG REQ D...	ACTIONS
aamdwebapp	14.26%	830.69 MB	29.30 MB/s	0	0.0 ms	1.96k	1.1 sec	
RD0003FF64BB3C	14.26%	830.69 MB	29.30 MB/s	—	—	1.96k	1.1 sec	

16. Under **Monitoring**, you can configure **Alerts**



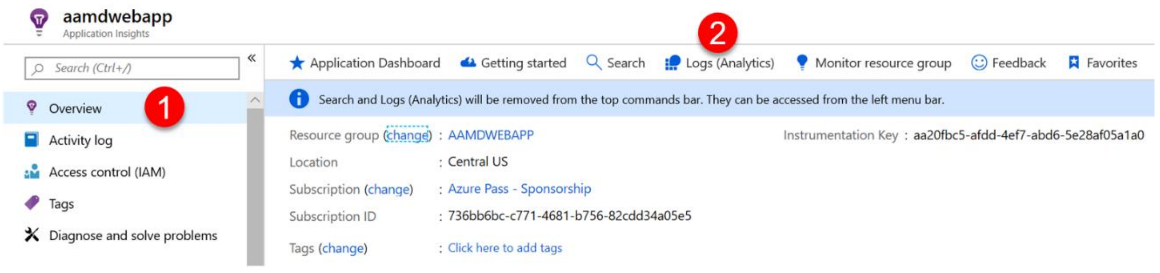
Don't see a subscription? Open Directory + Subscription settings

* Subscription: Azure Pass - Sponsorship
Resource group: AAMDWEBAPP
Resource: aamdwebapp
Time range: Past 24 hours

Selected subscriptions > AAMDWEBAPP > aamdwebapp

Task 6: Exploring Application Insights with Azure Logs

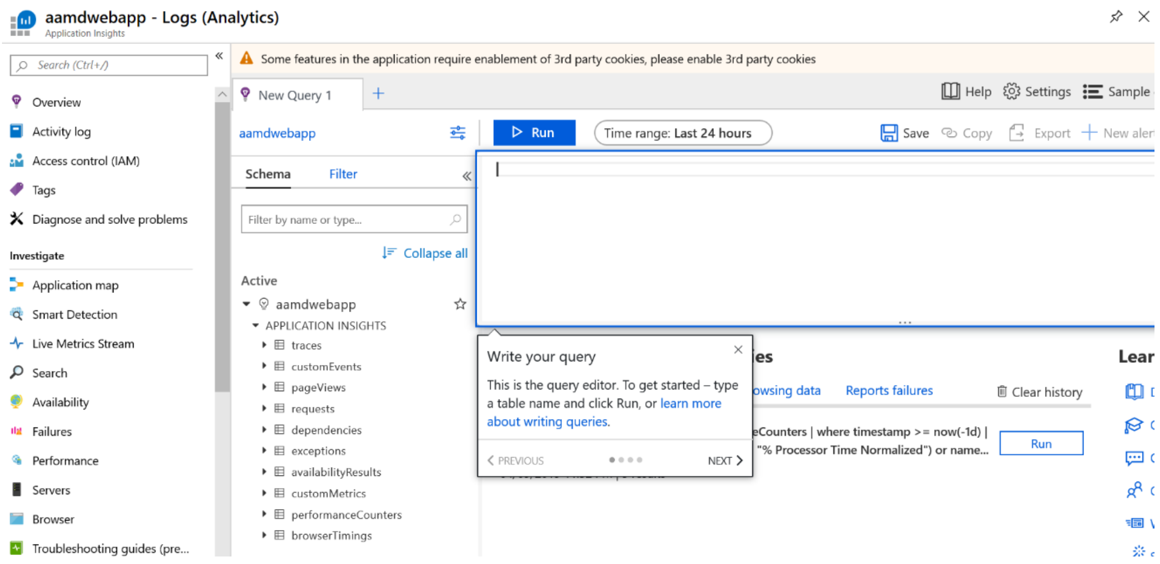
1. In the **Application Insights Overview** blade, click on **Logs (Analytics)**.



Search and Logs (Analytics) will be removed from the top commands bar. They can be accessed from the left menu bar.

Resource group (change) : AAMDWEBAPP
Location : Central US
Subscription (change) : Azure Pass - Sponsorship
Subscription ID : 736bb6bc-c771-4681-b756-82cdd34a05e5
Tags (change) : Click here to add tags
Instrumentation Key : aa20fbc5-afdd-4ef7-abd6-5e28af05a1a0

2. This will open **Azure Logs** (you may have to first click **Get Started**)



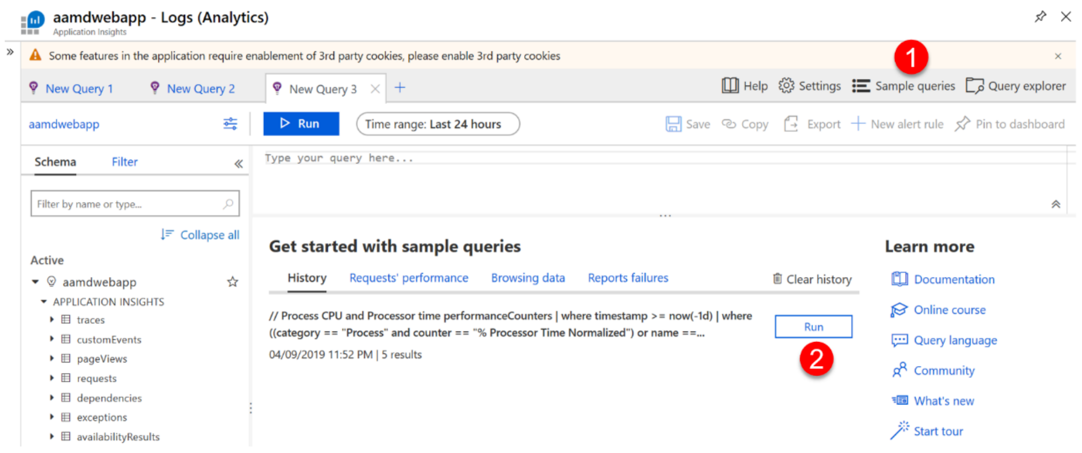
Some features in the application require enablement of 3rd party cookies, please enable 3rd party cookies

Write your query

This is the query editor. To get started – type a table name and click Run, or learn more about writing queries.

eCounters | where timestamp >= now(-1d) | "% Processor Time Normalized") or name...

3. Try queries from **Get started with sample queries**.



// Process CPU and Processor time performanceCounters | where timestamp >= now(-1d) | where ((category == "Process" and counter == "% Processor Time Normalized") or name ==...
04/09/2019 11:52 PM | 5 results

4. You can also write your own queries to generate insights as needed

5. Explore the features, write various queries and get yourself familiar with **Azure Logs**