**Application Insights – Spring Boot Java App**

This article walks you through creating a Spring Boot application using Spring Initializr, that uses Azure Application Insights Spring Boot Starter for end-to-end monitoring of Java applications on cloud.

Application Insights is an extensible Application Performance Management (APM) service for web developers on multiple platforms. Use it to monitor your live web application. It will automatically detect performance anomalies. It includes powerful analytics tools to help you diagnose issues and to understand what users actually do with your app. It's designed to help you continuously improve performance and usability. It works for apps on a wide variety of platforms including .NET, Node.js and Java EE, hosted on-premises, hybrid, or any public cloud. It integrates with your DevOps process, and has connection points to a variety of development tools.

**Pre-requirement**

* Visual Studio Code (<https://code.visualstudio.com/download>) or any other IDE
* Java SDK
* Apache Maven (<http://maven.apache.org/download.cgi>)

**Step 1:** Create Sample project using any IDE or follow below steps to create custom application using **Spring Initializer**

<https://start.spring.io/>

Project: **Maven Project**

Language: **Java**

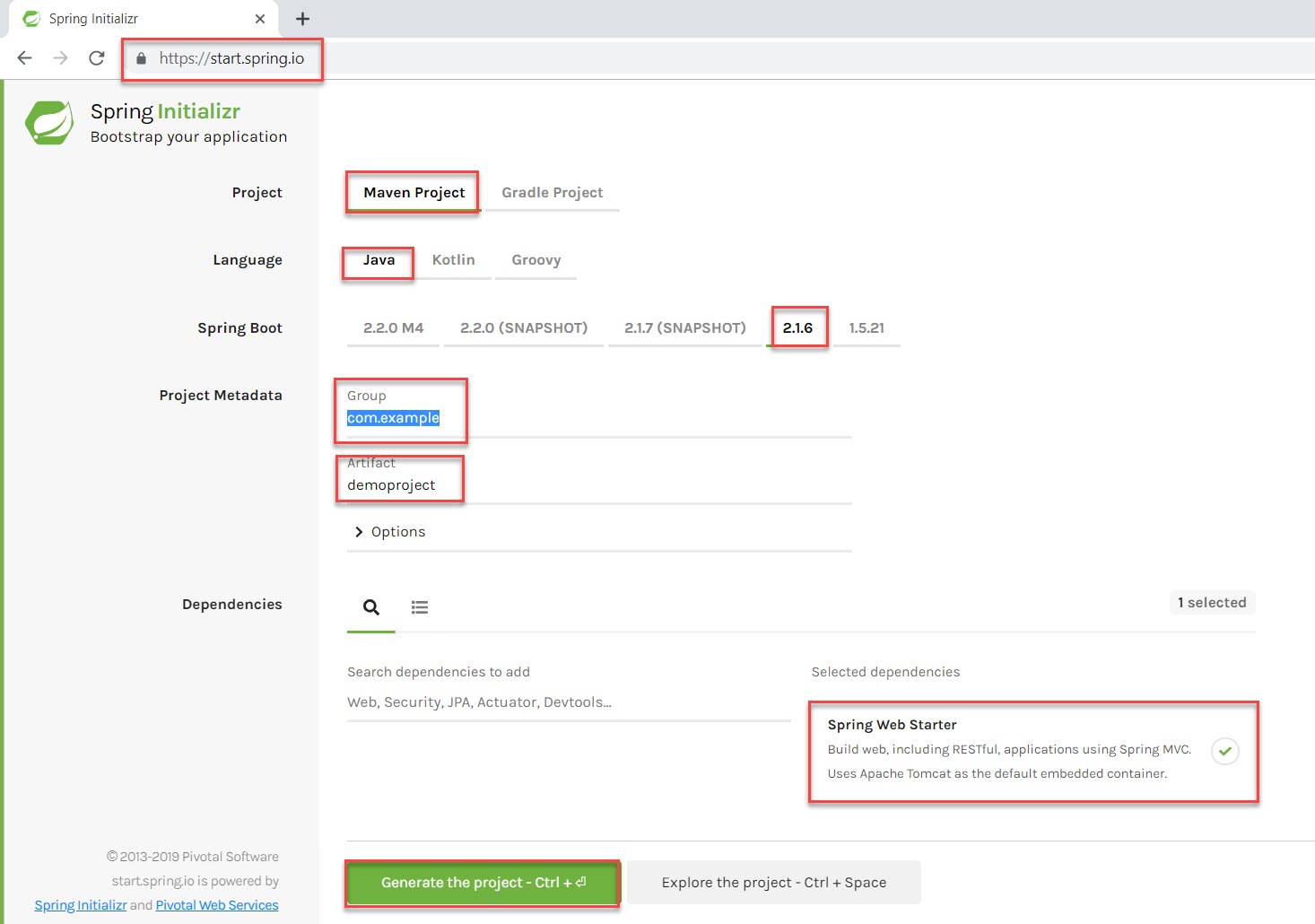
Spring Boot: **2.1.6** or any stable version

Project Metadata: **com.example**

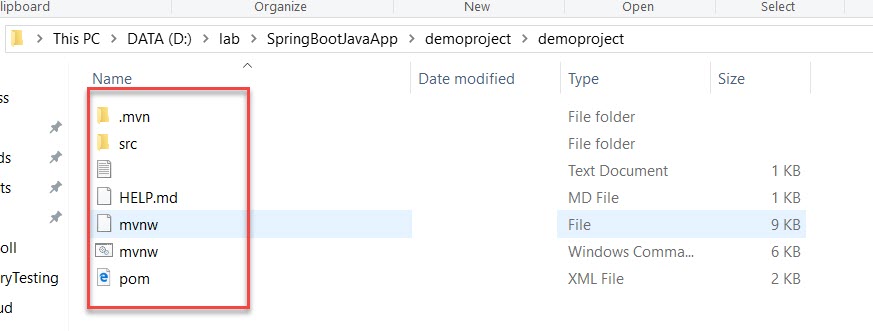
Artifact: **demoproject**

Dependencies: search for **Web** and Select **Spring Web Starter**

Click on **Generate the Project** button

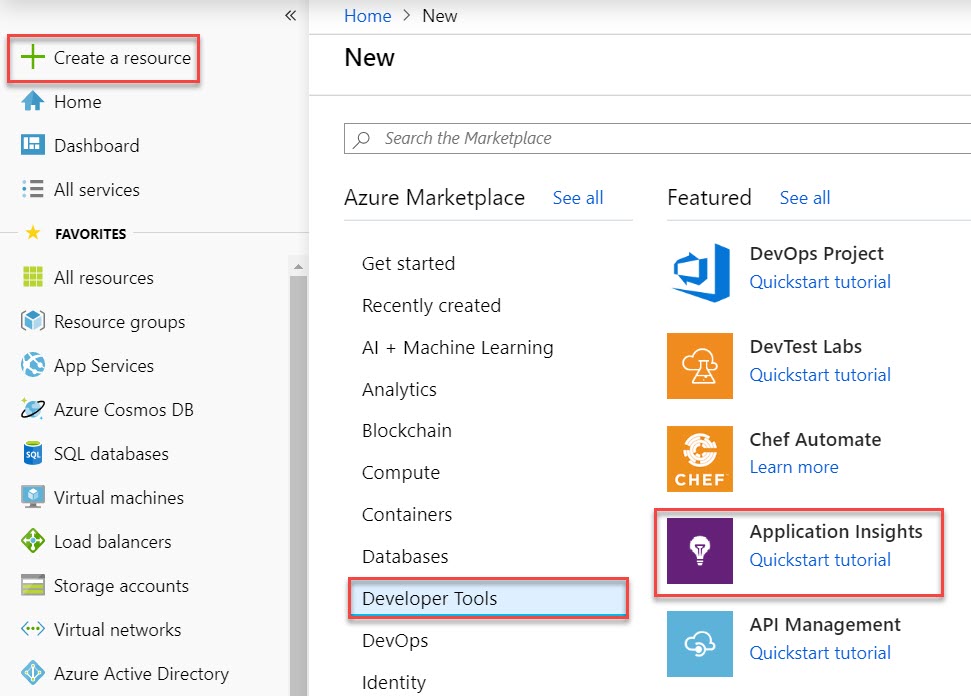


**Step 2:** Extract Sample Project folder.



**Step 3:** Create Application Insights

**+ Create a resource -> Developer Tools -> Application Insights**



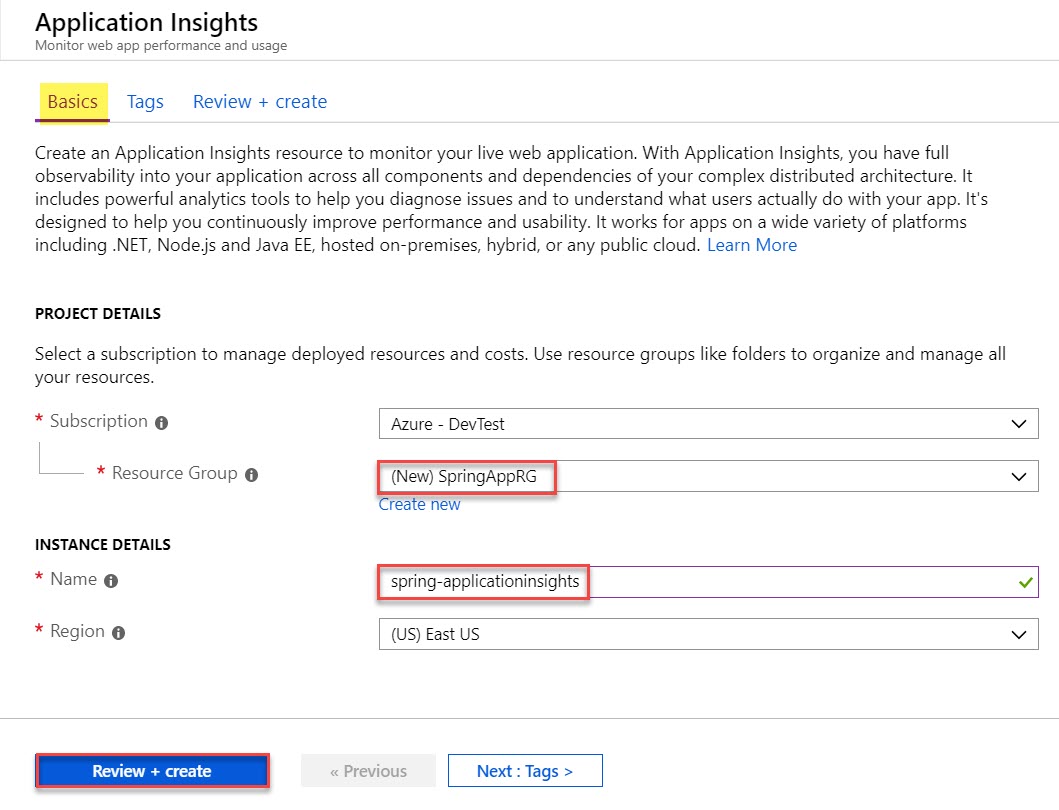
**Step 4:** Create Application Insights

Subscription: **Choose Subscription**

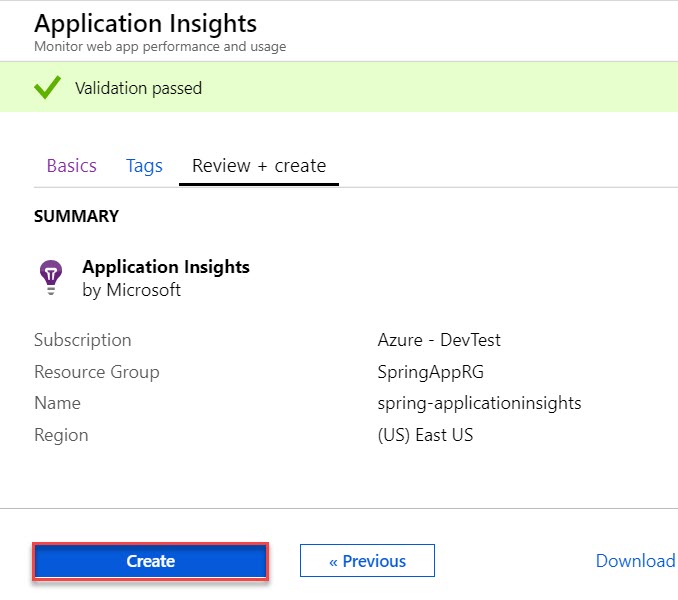
Resource Group: **Create New Resource Group** Ex. **SpringAppRG**

Name: **spring-applicationinsights**

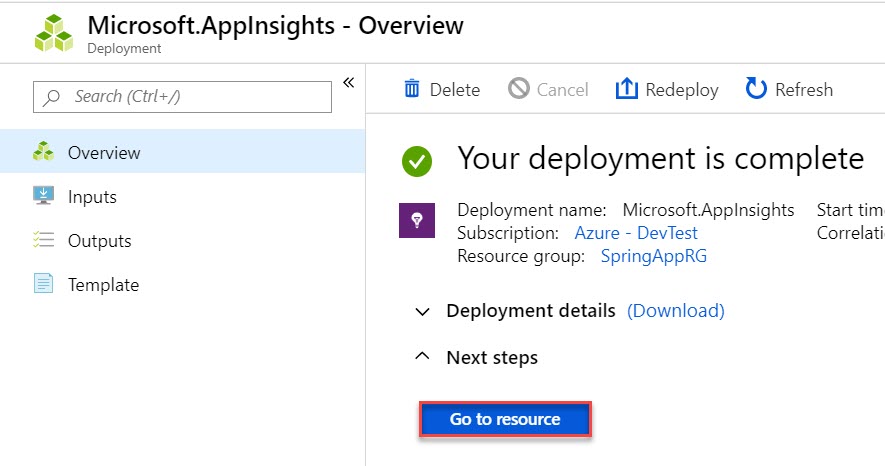
Click on **Review + Create** button



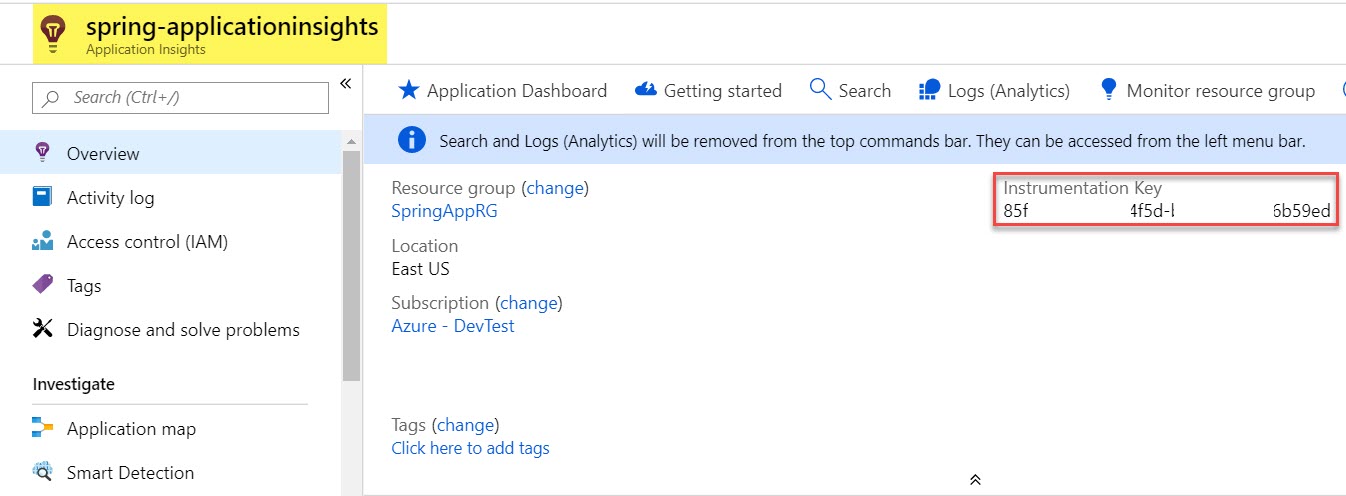
**Step 5:** Click on **Create** button.



**Step 6:** When Deployed, click on **Go to resource** button.

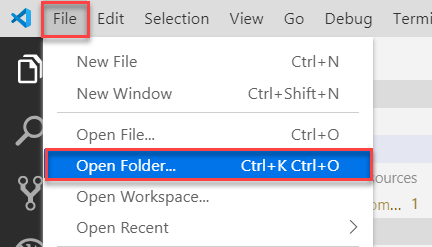


**Step 7:** Copy **Instrumentation Key** of Application Insights for further steps



**Step 8:** Start Visual Studio Code  
**File -> Open Folder….**

Browse **DemoProject** folder



**Step 9:** Open **POM.xml** file and add **Microsoft Azure Dependency**

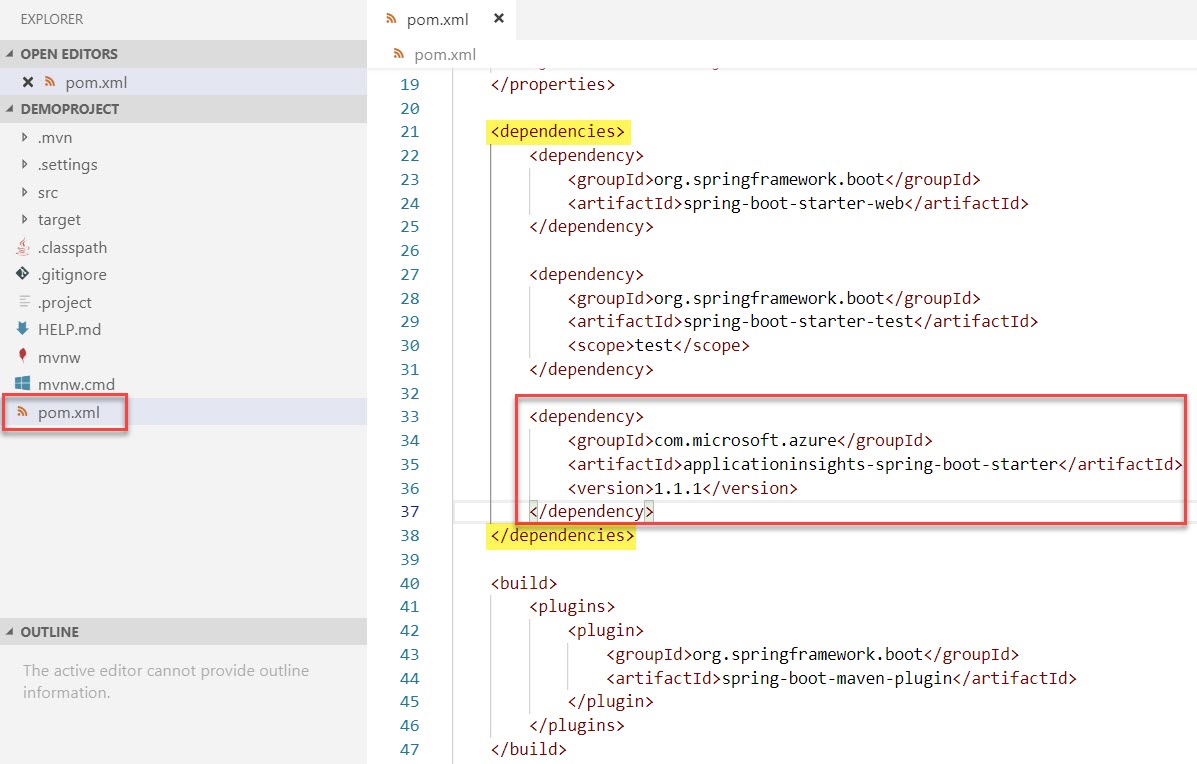
<dependency>

            <groupId>com.microsoft.azure</groupId>

            <artifactId>applicationinsights-spring-boot-starter</artifactId>

            <version>1.1.1</version>

        </dependency>



**Step 10:** Open **application.properties** file

**src -> main -> resources ->** open **application.properties** file

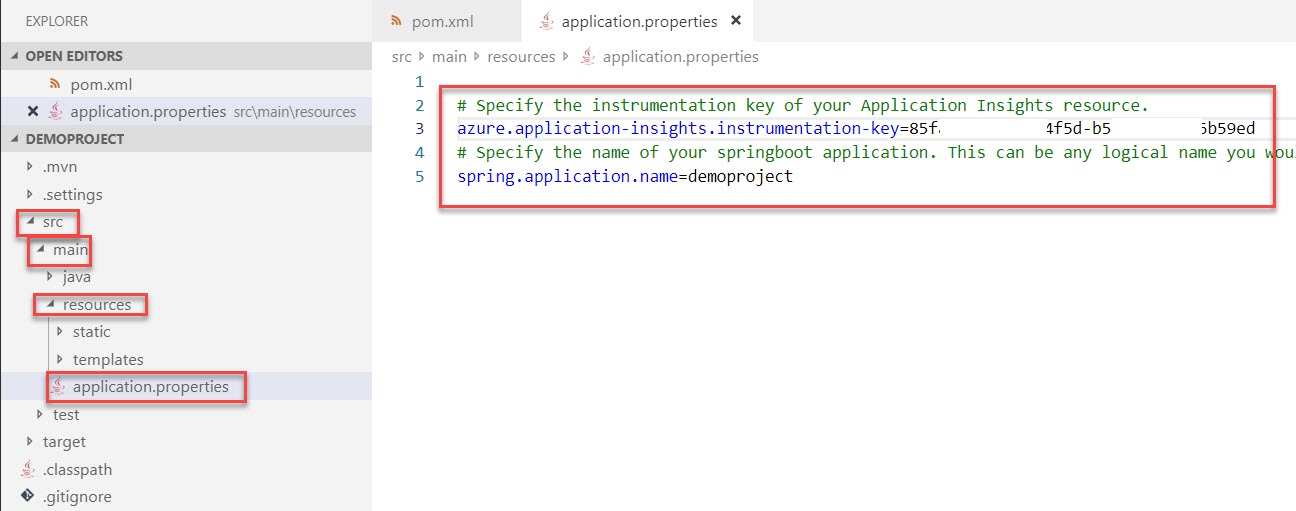
Add below code. Update Application Insight Instrumentation Key and application name

# Specify the instrumentation key of your Application Insights resource.

azure.application-insights.instrumentation-key=85jd69c-0626-44sd-b58b-192ks6b59ed

# Specify the name of your springboot application. This can be any logical name you would like to give to your app.

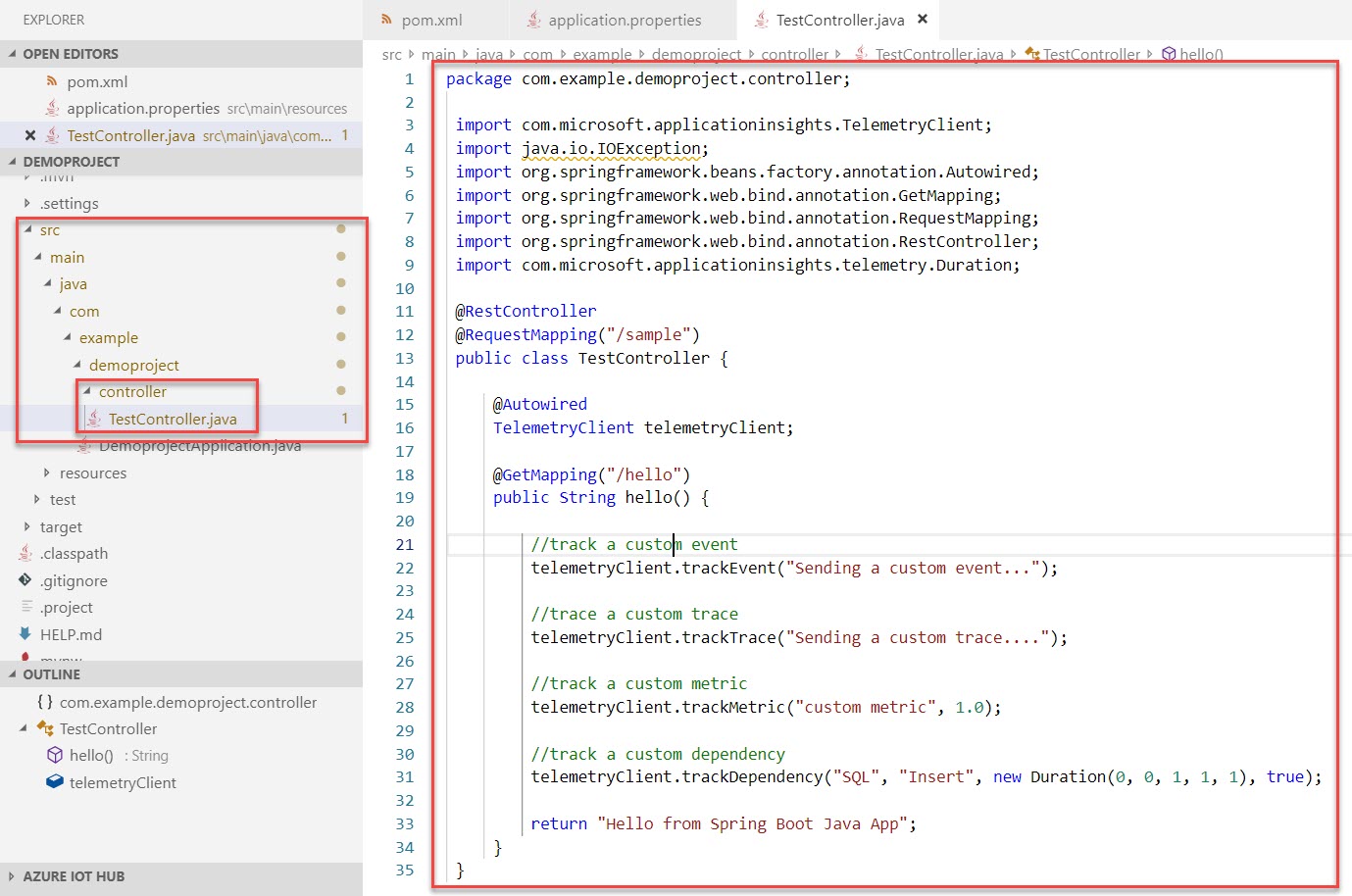
spring.application.name=demoproject



**Step 11:** **Create Folder and File**.

**Folder: src -> main -> java -> com -> example -> demoproject -> Create controller folder**

Create **TestController.java** file into newly created folder



Add below code:

package com.example.demoproject.controller;

import com.microsoft.applicationinsights.TelemetryClient;

import java.io.IOException;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.microsoft.applicationinsights.telemetry.Duration;

@RestController

@RequestMapping("/sample")

public class TestController {

@Autowired

TelemetryClient telemetryClient;

@GetMapping("/hello")

public String hello() {

//track a custom event

telemetryClient.trackEvent("Sending a custom event...");

//trace a custom trace

telemetryClient.trackTrace("Sending a custom trace....");

//track a custom metric

telemetryClient.trackMetric("custom metric", 1.0);

//track a custom dependency

telemetryClient.trackDependency("SQL", "Insert", new Duration(0, 0, 1, 1, 1), true);

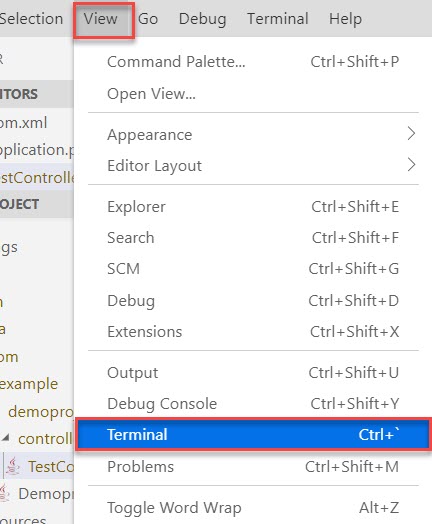
return "Hello from Spring Boot Java App";

}

}

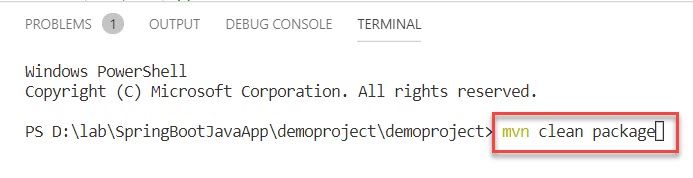
**Step 12:** Now **Build and Run application**

**View** menu -> **Terminal** option



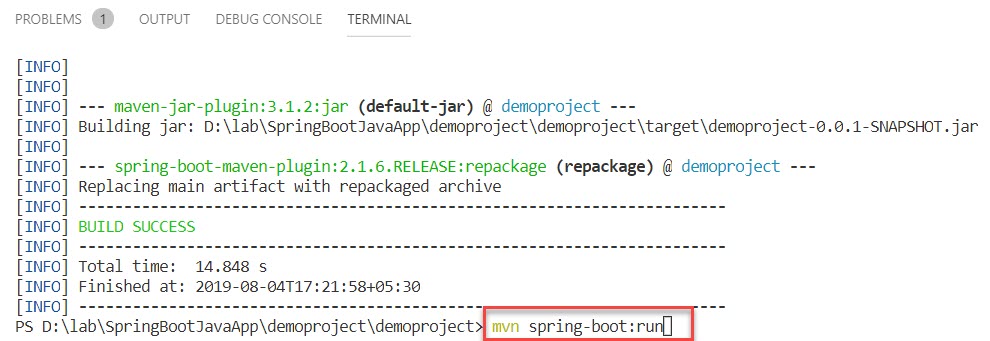
**Step 13:** Run below command to compile your code and package it.

mvn clean package



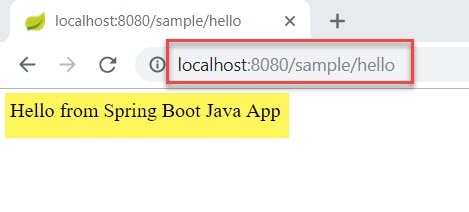
**Step 14:** Now run an application

mvn spring-boot:run

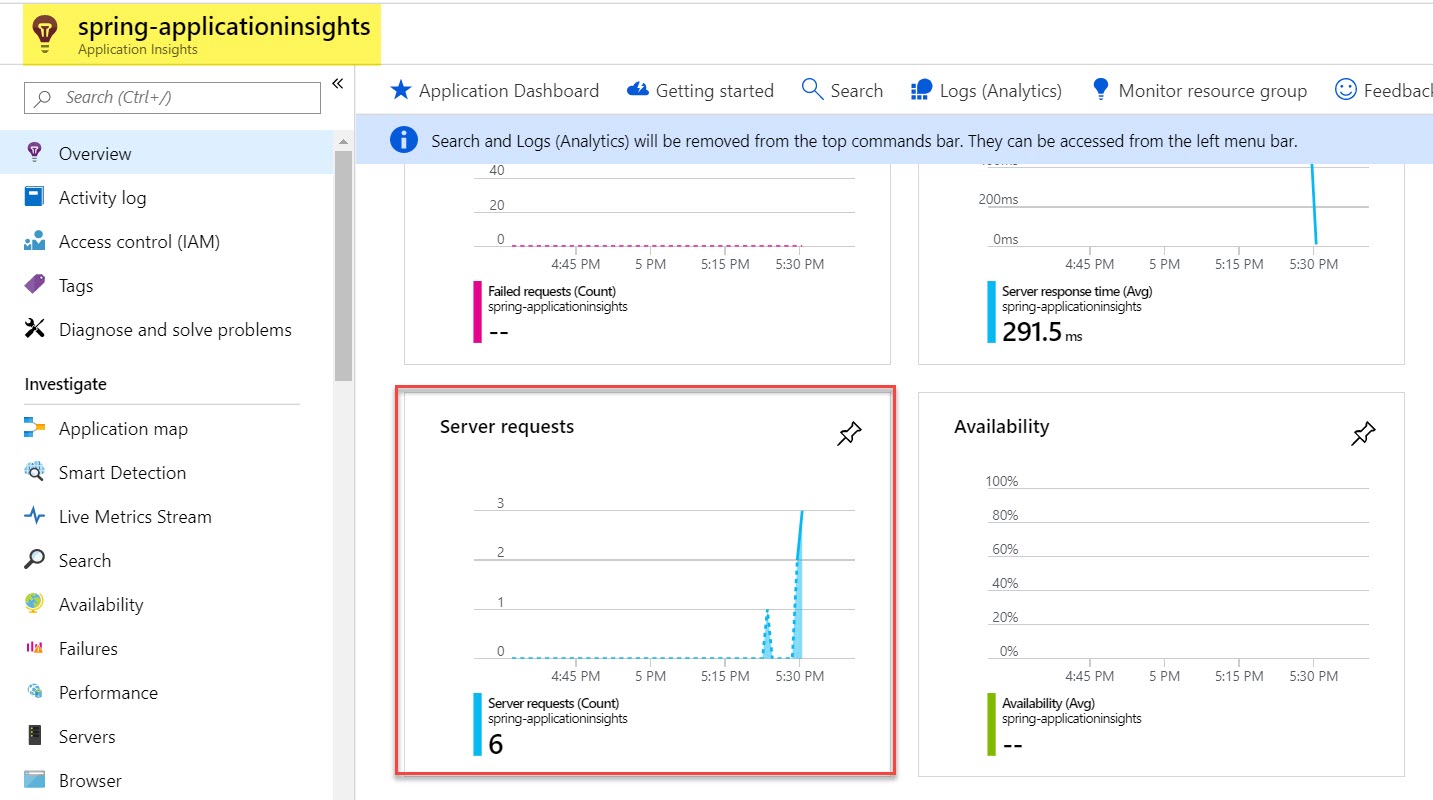


**Step 15:** Wait for few seconds and Run in Browser

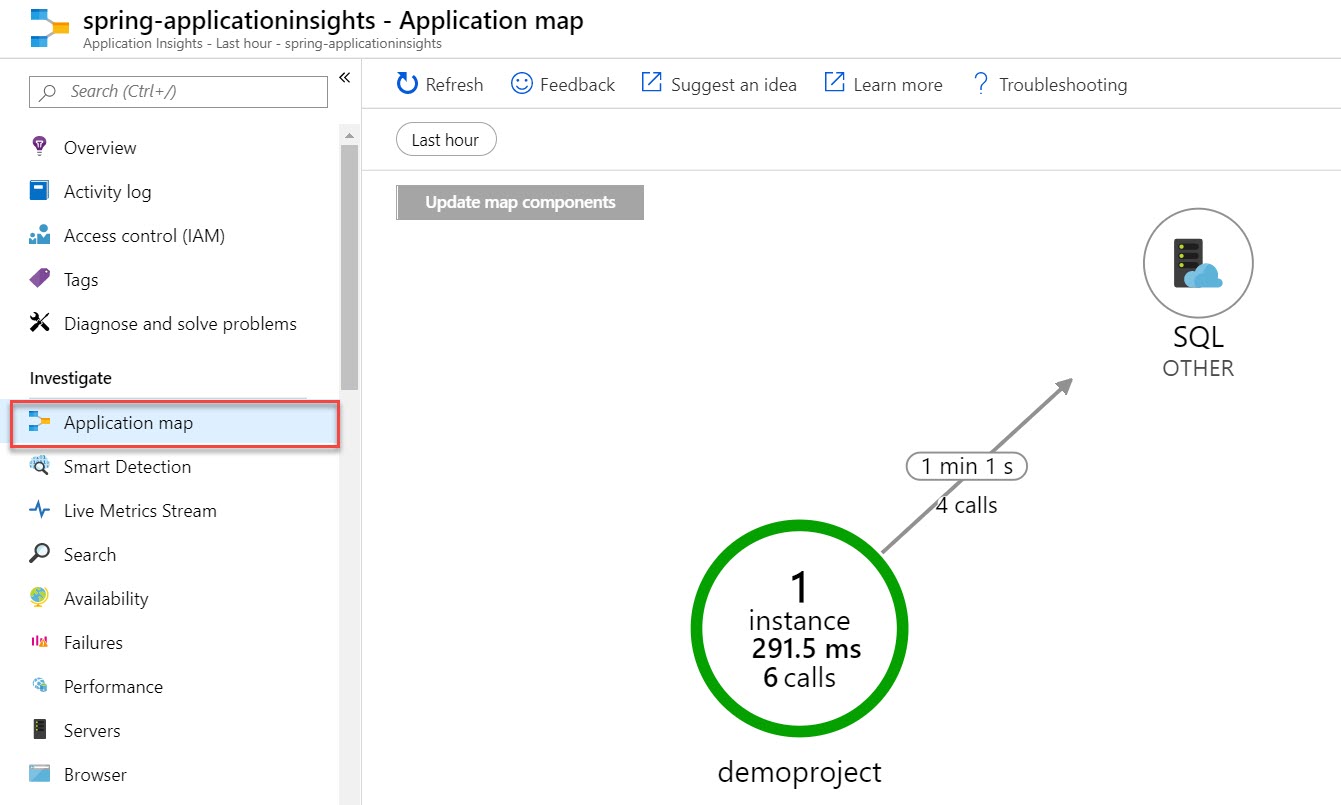
<http://localhost:8080/sample/hello>



**Step 16:** Navigate to **Azure Portal** and verify Application Insights Ex. Server requests.



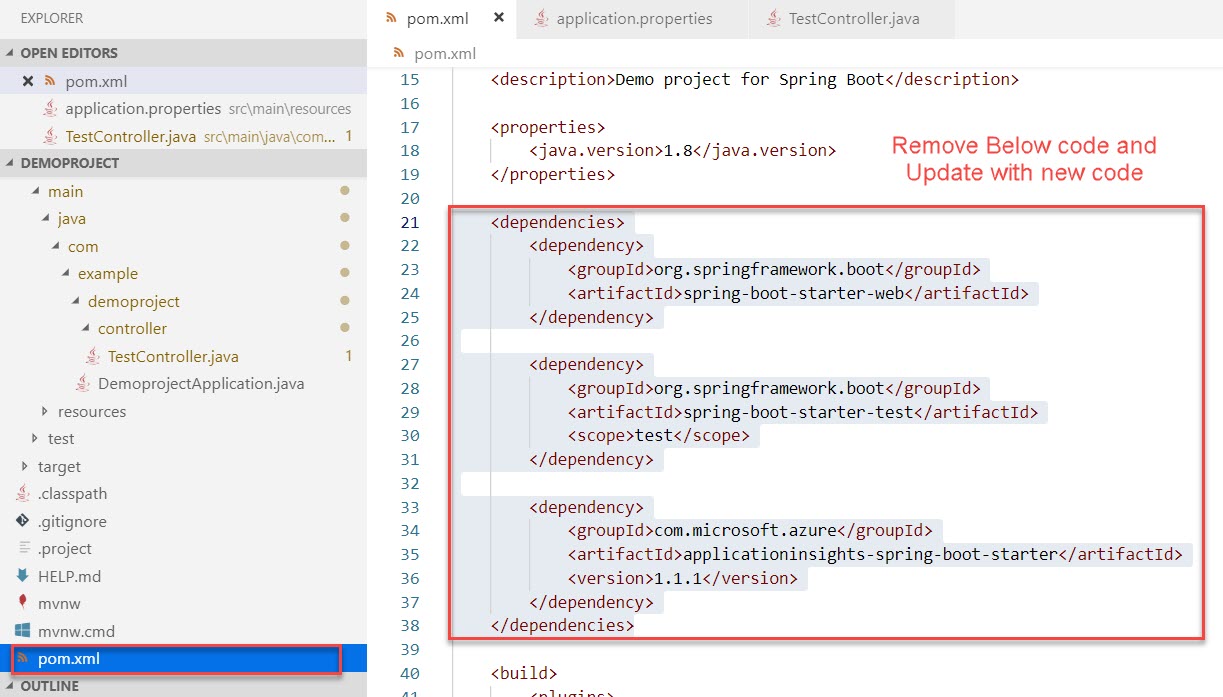
**Step 17:** You can click on **Application Map** tile to view high level components and their interaction with each other. This is a recommended place to get a high-level overview of entire application. Each Spring Boot Microservice is recognized by the spring application name.



**Step 18:** Log4j is a fast, flexible and reliable logging framework (APIs) written in Java.

Open **POM.xml** file and **replace complete dependency section**.

Add few more dependency such as springboot, Log4j, etc.



<dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

            <exclusions>

                <exclusion>

                    <groupId>org.springframework.boot</groupId>

                    <artifactId>spring-boot-starter-logging</artifactId>

                </exclusion>

            </exclusions>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-log4j2</artifactId>

        </dependency>

        <dependency>

            <groupId>com.microsoft.azure</groupId>

            <artifactId>applicationinsights-spring-boot-starter</artifactId>

            <version>1.1.1</version>

        </dependency>

        <dependency>

            <groupId>com.microsoft.azure</groupId>

            <artifactId>applicationinsights-logging-log4j2</artifactId>

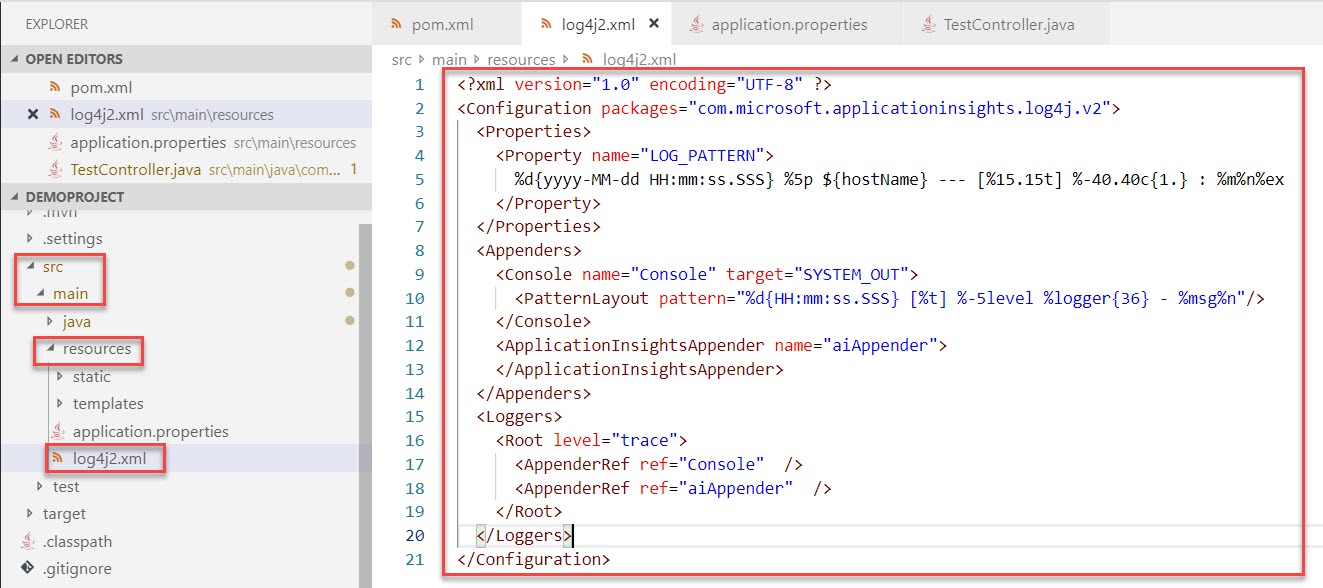
            <version>2.1.1</version>

        </dependency>

    </dependencies>

**Step 19:** Now add **log4j2.xml** file

**src -> main -> resources -> Create New file log4j2.xml**



Add below code:

<?xml version="1.0" encoding="UTF-8" ?>

<Configuration packages="com.microsoft.applicationinsights.log4j.v2">

<Properties>

<Property name="LOG\_PATTERN">

%d{yyyy-MM-dd HH:mm:ss.SSS} %5p ${hostName} --- [%15.15t] %-40.40c{1.} : %m%n%ex

</Property>

</Properties>

<Appenders>

<Console name="Console" target="SYSTEM\_OUT">

<PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>

</Console>

<ApplicationInsightsAppender name="aiAppender">

</ApplicationInsightsAppender>

</Appenders>

<Loggers>

<Root level="trace">

<AppenderRef ref="Console" />

<AppenderRef ref="aiAppender" />

</Root>

</Loggers>

</Configuration>

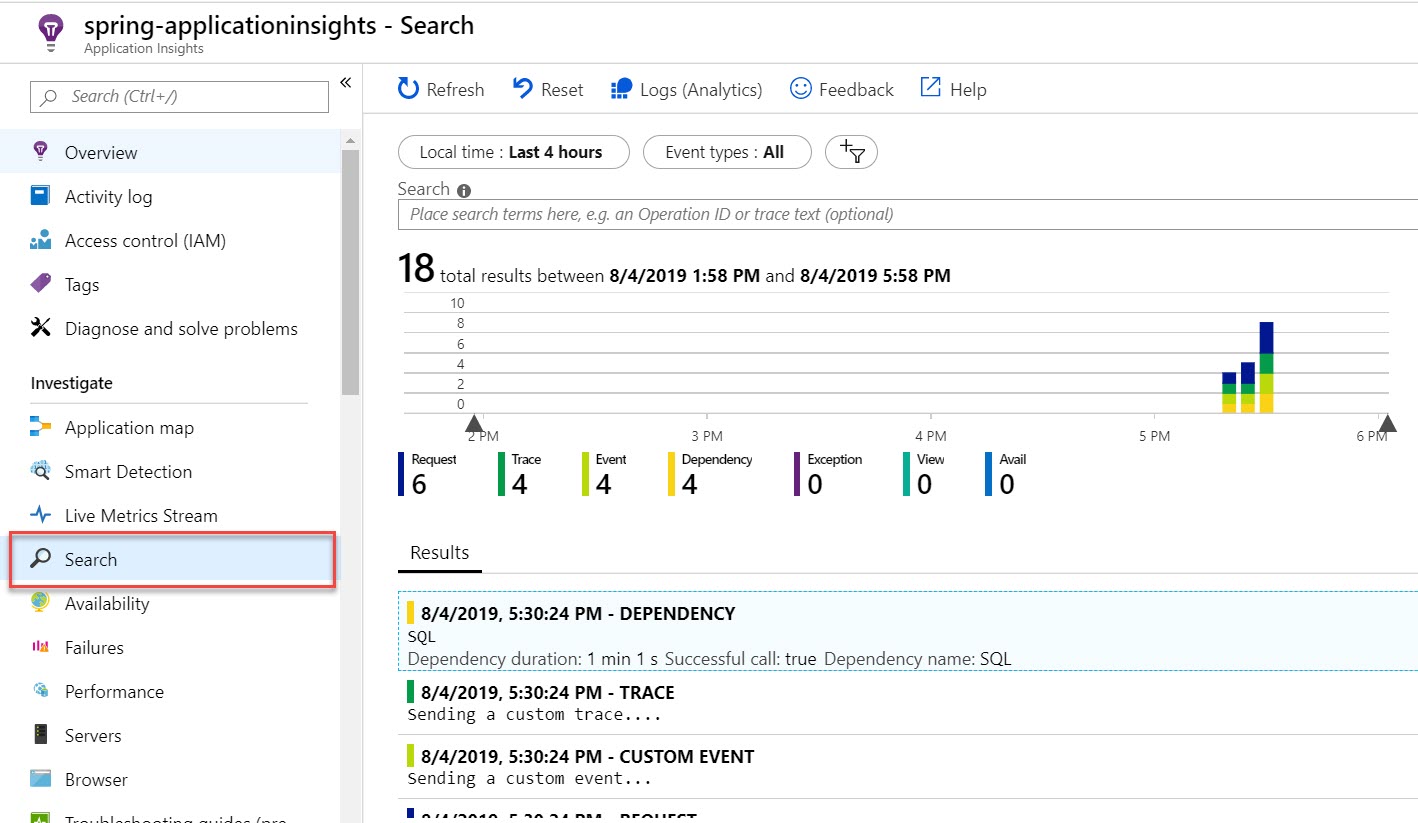
**Step 20:** Now Build and Run once again

mvn clean package

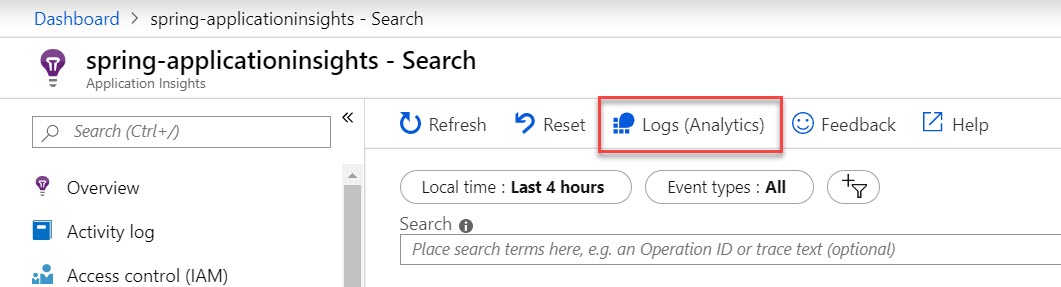
mvn spring-boot:run

Run <http://localhost:8080/sample/hello>

**Step 21:** Navigate to **Azure Portal** and select **Search** option



**Step 22:** Select **Logs (Analytics)** option



**Step 23:** Run default query or write new query to get the result

