Azure Monitoring, Azure App Insights

Lecture 15

Deep Azure@McKesson

Zoran B. Djordjević

Why Monitoring

- Distributed applications and services running in the cloud are, by their nature, complex pieces of software that comprise many moving parts.
- In a production environment, it's important to be able to track the way in which users utilize your system, trace resource utilization, and generally monitor the health and performance of your system. You can use this information as a diagnostic aid to detect and correct issues, and also to help spot potential problems and prevent them from occurring.
- In Azure we could use Azure Monitoring Services to gain an insight into how well a system is functioning. Monitoring is a crucial part of maintaining quality-of-service targets.

Common scenarios for collecting monitoring data include:

- Ensuring that the system remains healthy.
- Tracking the availability of the system and its component elements.
- Maintaining performance to ensure that the throughput of the system does not degrade unexpectedly as the volume of work increases.
- Guaranteeing that the system meets any service-level agreements (SLAs) established with customers.
- Protecting the privacy and security of the system, users, and their data.
- Tracking the operations that are performed for auditing or regulatory purposes.
- Monitoring the day-to-day usage of the system and spotting trends that might lead to problems if they're not addressed.
- Tracking issues that occur, from initial report through to analysis of possible causes, rectification, consequent software updates, and deployment.
- Tracing operations and debugging software releases.

Monitoring on Azure

- Performance issues in the Cloud app can impact our business. Cloud application typically have multiple interconnected components and frequent releases, and degradations could happen in any component and at any time.
- When deploying a new app, your users usually discover issues that you didn't find in testing. You should know about these issues immediately, and have tools for diagnosing and fixing the problems.
- Furthermore, problems in your application can result from the underlying infrastructure on which those applications run, so having a holistic view of your application and infrastructure is key to monitoring your Azure environment.
- Microsoft Azure has a range of services for identifying and resolving such problems.
- Microsoft Azure utilizes standard system monitoring tools that are part of Windows or Linux operating systems, third party system and application monitoring tools and native Azure specific monitoring services.
- You could add your own or third party monitoring tools of your choice to your applications deployed to Azure to enhance your monitoring capabilities and could integrate those tools with Azure specific tools and services.
- Azure offers three main monitoring services: Azure Monitor, Azure Application Insights and Azure Log Analytics

Azure Monitor

Azure Monitor: https://azure.microsoft.com/en-us/services/monitor/

- Azure Monitor is an Azure service that operates as a consolidated pipeline for all monitoring data from Azure services.
- Azure Monitor gives us access to performance metrics and events that describe the operation of the Azure infrastructure and any Azure services you are using.
- Azure Monitor is a monitoring data pipeline for our Azure environment, and delivers monitoring data directly into Log Analytics as well as 3rd party tools where we could gain insight into content and behavior of data and combine monitoring data with other data from on premises or other cloud resources.

Application Insights

Application Insight https://azure.microsoft.com/en-us/services/application-insights/

- Application Insights is an Azure service that offers application performance monitoring and user analytics.
- Application Insights monitors the code we have written and applications we have deployed on Azure, on-premises, or other clouds.
- In order to use Application Insights we need to instrument our applications with the Application Insights SDK.
- Using instrumented App Insights Code we could get access to a range of data including response times of dependencies, exception traces, debugging snapshots, and execution profiles.
- Application Insights provides powerful tools for analyzing application telemetry while developing and operating our applicatio.
- Application Insights deeply integrates with Visual Studio to enable us to get right to the problematic lines of code so we could fix them.
- Application Insights also offers usage analytics which we could use to analyze customer usage of our applications. Such information is beneficial from both technical and marketing perspective.

Log Analytics

Log Analytics https://azure.microsoft.com/en-us/services/log-analytics/

- **Log Analytics** is an Azure service that ingests log and metric data from Azure services (via Azure Monitor), Azure VMs, and on-premises or other cloud infrastructure.
- Log Analytics offers flexible log search and out-of-the box analytics on top of this data.
- Log Analytics provides rich tools to analyze data across sources, allows complex queries across all logs, and can proactively alert on specified conditions.
- Using Log Analytics we could even collect custom data into its central repository so we could query and visualize the data.
- We could also take advantage of Log Analytic's built-in solutions to immediately gain insights into the security and functionality of our infrastructure.

- All Azure monitoring services are available in Azure Portal.
- We could also access monitoring functions for specific Azure resources by highlighting those resources and drilling down into their monitoring options

Basic Scenarios, Fix Errors under Development

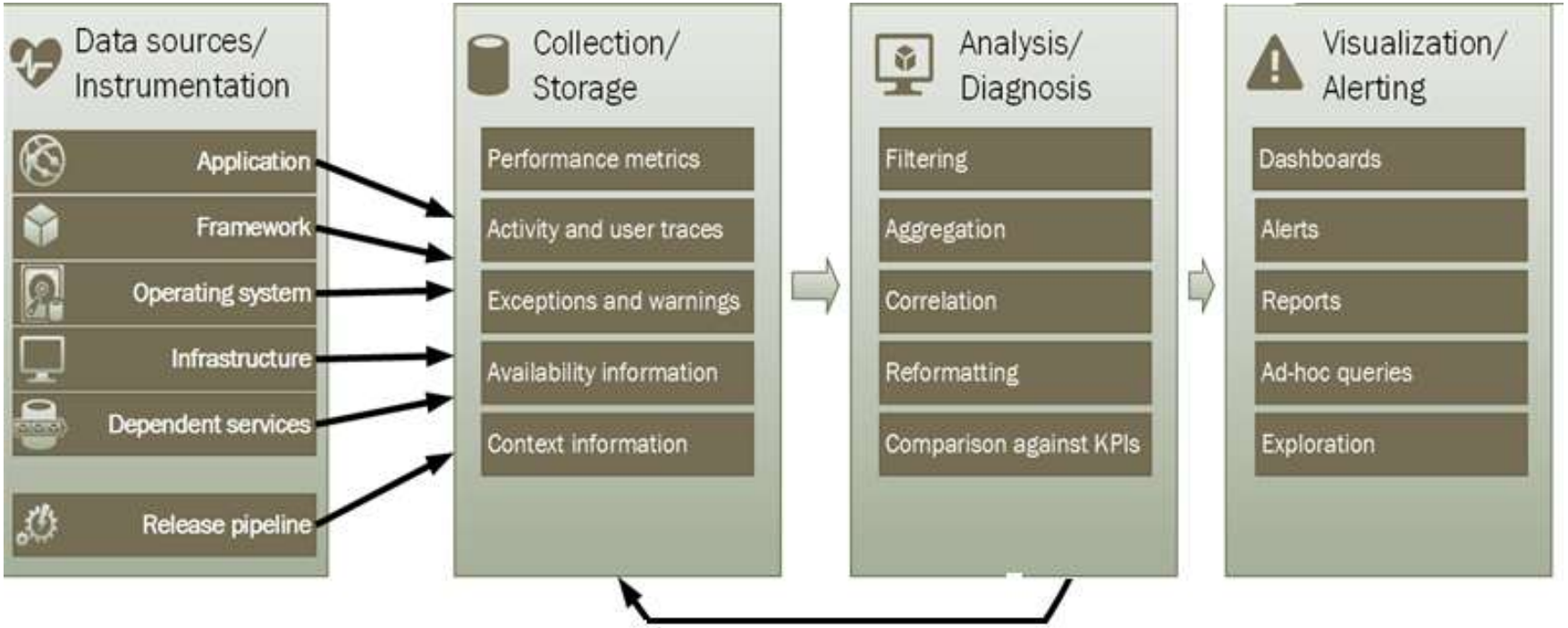
- **We could use Application Insights, Azure Monitor, and Visual Studio together**
- Azure provides the full power of the Visual Studio debugger in the cloud.
  1. We could configure Azure Monitor to send telemetry to Application Insights.
  2. We enable Visual Studio to include the Application Insights SDK in our application.
  3. Once in Application Insights, we could use the Application Map to discover visually which parts of your running application are healthy or not. For those parts that are not healthy, errors and exceptions are already available for exploration.
- We could use the various analytics in Application Insights to go deeper. If we are not sure about the error, we could use the Visual Studio debugger to trace into code and pin point a problem further.

Scenario, Azure .NET App Errors in Production

- Azure has recently introduced an Application Insights Snapshot Debugger which allows you analysis of run-time errors.
- When a certain error threshold occurs with production components, the system automatically captures telemetry in windows of time called “snapshots.“
- The amount captured is safe for a production cloud because it’s small enough not to affect performance but significant enough to allow tracing.
- The system could be configured to capture multiple snapshots.
- We could look at a point in time in the Azure portal or use Visual Studio for the full and detailed view.
- Inside Visual Studio, developers could walk through that snapshot as if they were debugging in real-time. Local variables, parameters, memory, and frames are all available.
- To use Snapshot data, developers must be granted access to this production data via an Azure Role-Based Access Control (RBAC) role: Application Insights Component Contributor

Monitoring Process as a Pipeline

- Monitoring a large-scale distributed system poses a significant challenge. There is likely to be a significant overlap in the monitoring and diagnostic data that's required for each scenario, although this data might need to be processed and presented in different ways.
- For these reasons, one should take a holistic view of monitoring and diagnostics. You can envisage the entire monitoring and diagnostics process as a pipeline that comprises the stages



Built-in roles for Azure role-based access control

- Azure Role-Based Access Control (RBAC) comes with the following built-in roles that can be assigned to users, groups, and services. You can't modify the definitions of built-in roles. However, you can create Custom roles in Azure RBAC to fit the specific needs of your organization.
- The following table provides brief descriptions of the built-in roles. Click the role name to see the detailed list of **actions** and **notactions** for the role. The **actions** property specifies the allowed actions on Azure resources. Action strings can use wildcard characters. The **notactions** property specifies the actions that are excluded from the allowed actions.+
- The action defines what type of operations you can perform on a given resource type. For example:
  - **Write** enables you to perform PUT, POST, PATCH, and DELETE operations.
  - **Read** enables you to perform GET operations.

Partial List of Roles

Role name	Description
API Management Service Contributor	Can manage API Management service and the APIs
API Management Service Operator Role	Can manage API Management service, but not the APIs themselves
API Management Service Reader Role	Read-only access to API Management service and APIs
Application Insights Component Contributor	Can manage Application Insights components
Automation Operator	Able to start, stop, suspend, and resume jobs
Backup Contributor	Can manage backup in Recovery Services vault
Backup Operator	Can manage backup except removing backup, in Recovery Services vault
Backup Reader	Can view all backup management services
Billing Reader	Can view all billing information
BizTalk Contributor	Can manage BizTalk services
ClearDB MySQL DB Contributor	Can manage ClearDB MySQL databases
Contributor	Can manage everything except access.
Data Factory Contributor	Can create and manage data factories, and child resources within them.
DevTest Labs User	Can view everything and connect, start, restart, and shutdown vi
Application Insights Component Contributor	Can manage Application Insights components
Automation Operator	Able to start, stop, suspend, and resume jobs
Backup Contributor	Can manage backup in Recovery Services vault
Backup Operator	Can manage backup except removing backup, in Recovery Services vault
Backup Reader	Can view all backup management services

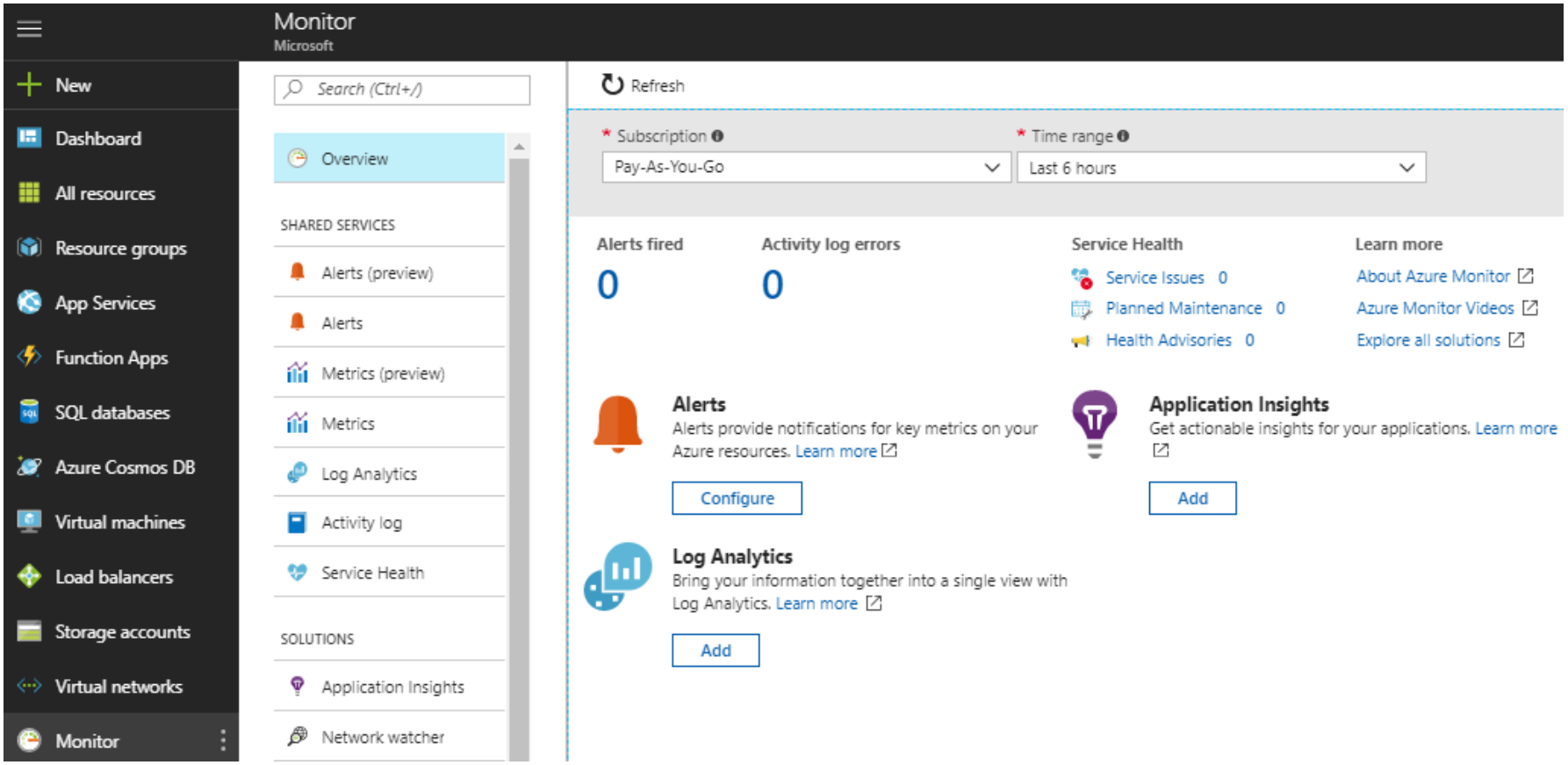
Full list could be found at:
https://docs.microsoft.com/en-us/azure/active-directory/role-based-access-built-in-roles

Azure Monitor

- Azure Monitor is an Azure service that provides a single source for monitoring Azure resources.
- With Azure Monitor, you can visualize, query, route, archive, and take action on the metrics and logs coming from resources in Azure.
- You can work with this data using:
  - Monitor Portal blade,
  - Monitor PowerShell Cmdlets,
  - Cross-Platform CLI, or
  - Azure Monitor REST APIs.
- We will walk through a few of the key components of Azure Monitor, using the portal for demonstration.

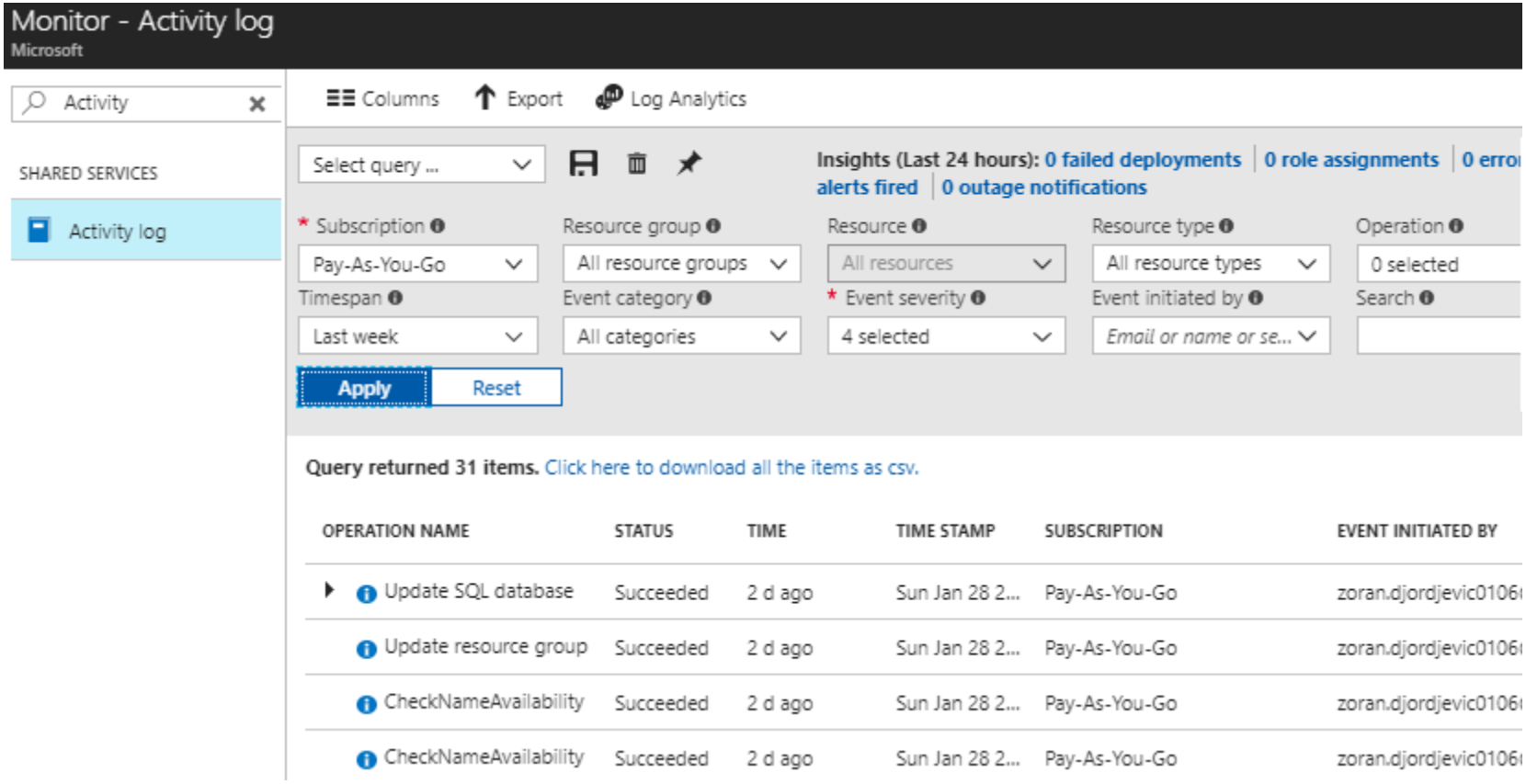
Setup Azure Monitor

- We will monitor the web App we build in one of our first class.
- In Azure Portal select +New, navigate to More services and then find Monitor, or if Monitor is already visible in the left navigation list, select and click the Monitor option.
- Monitor blade contains all your monitoring settings and data in one consolidated view.



Activity Log

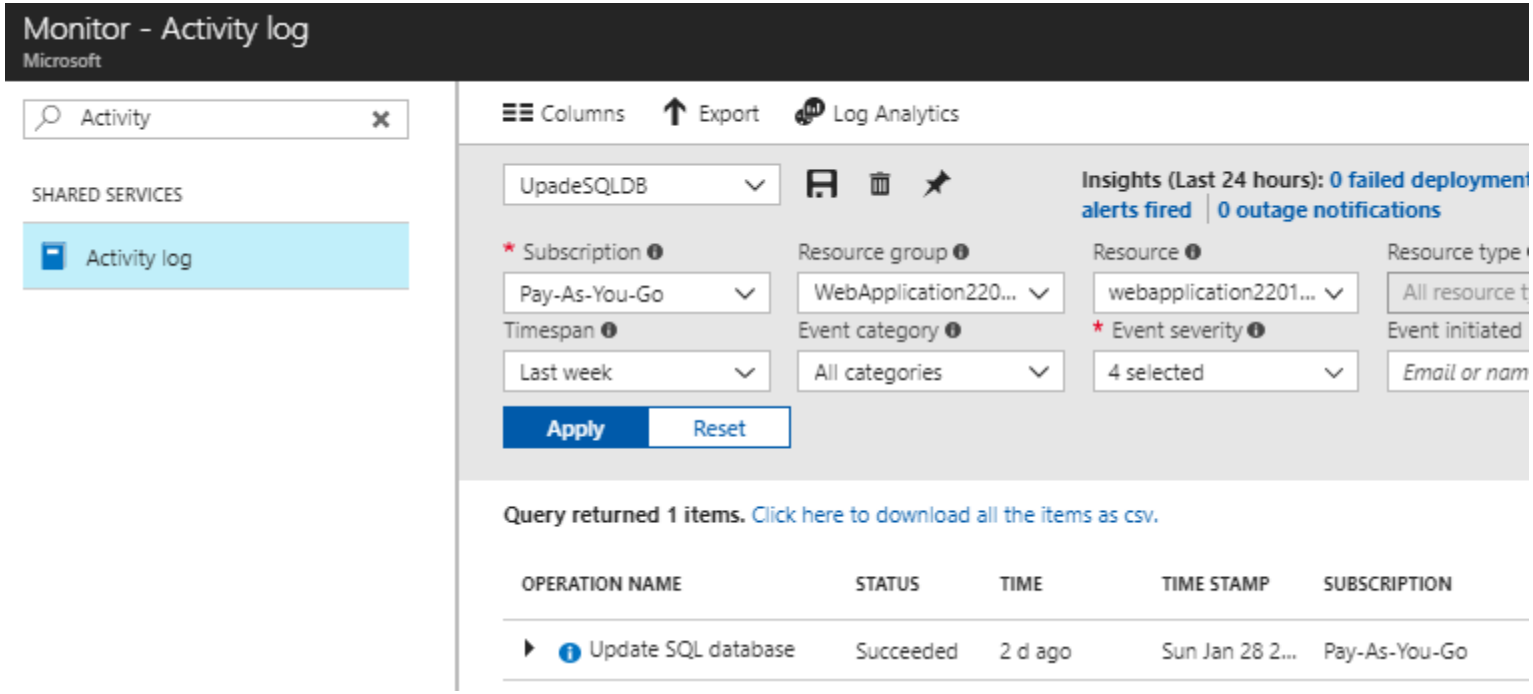
- Azure Monitor has three basic categories of monitoring data: The **activity log**, **metrics**, and **diagnostic logs.** Select Activity Log.



- The **activity log** describes all operations performed on resources in your subscription. You can see the 'what, who, and when' for any create, update, or delete operations on resources in your subscription.
- For example, the Activity Log tells you when a web app was stopped and who stopped it.
- Activity Log events are stored in the platform and available to query for 90 days.

Activity Log Queries

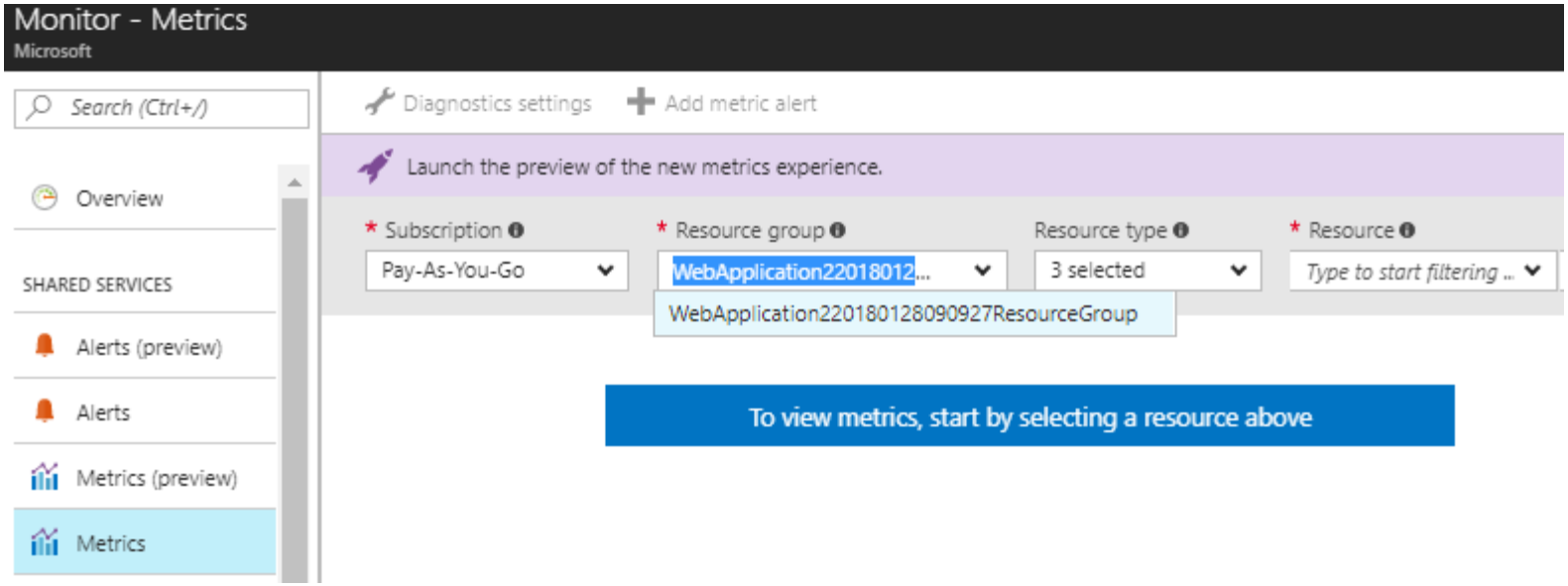
- We could create and save queries for common filters, then pin the most important queries to a portal dashboard so you'll always know if events that meet your criteria have occurred.
- We have selected the resource group supporting our app and DB resource.
- Then we saved the query as , say: UpdateSQLDB



- Please note that Monitoring system displays results with a long delay.

Metrics

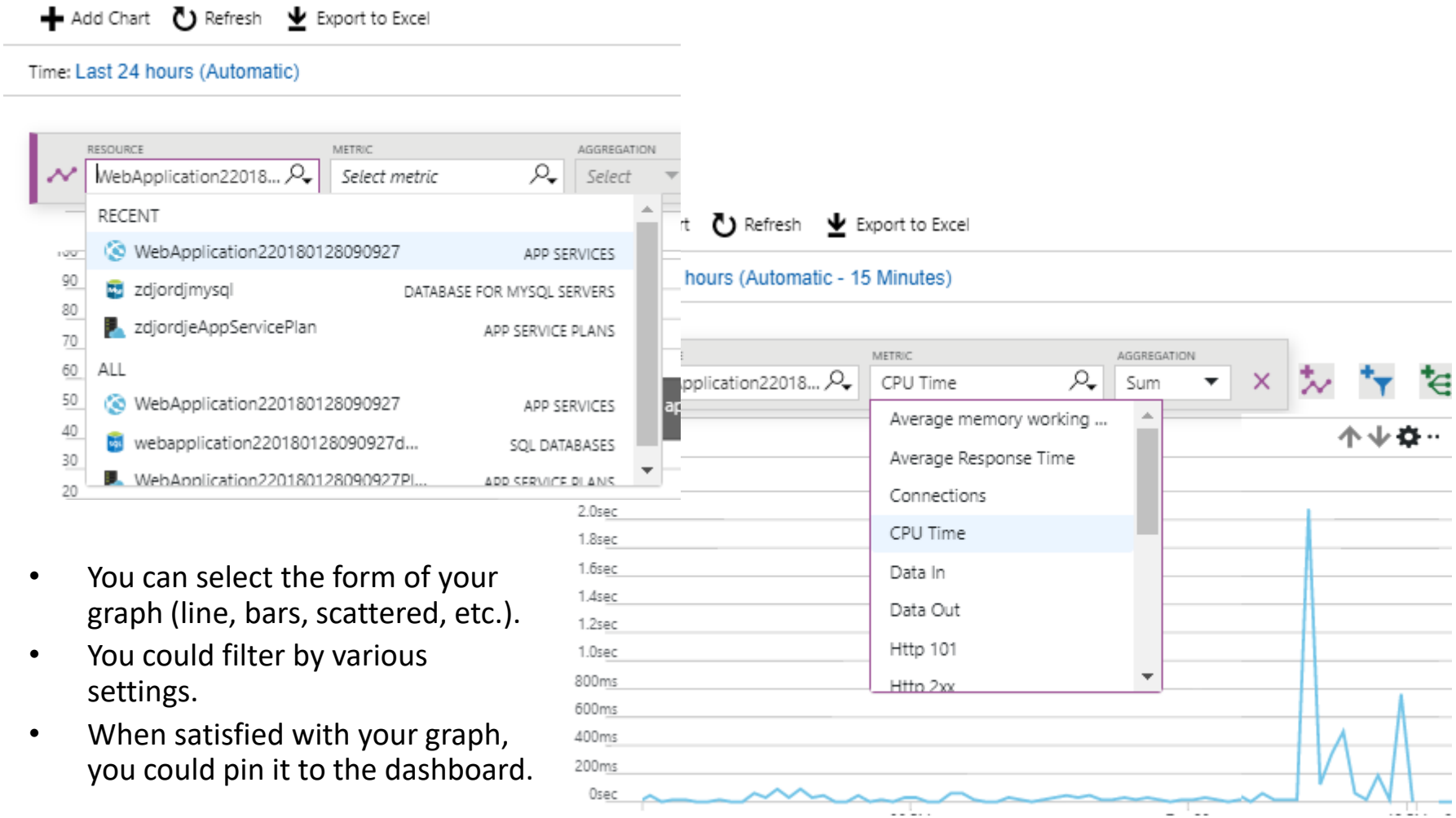
- While on the **Monitor** tile, click the **Metrics** (preview) section. You first need to select a resource by using the drop-down options at the top of the blade.



- All Azure resources emit **metrics**. Monitor view brings together all metrics in a single pane of glass so you can easily understand how your resources are performing.

Select Resource & Metrics

- Select Resource you want to monitor and then select the metrics you want displayed over selected period of time.



- You can select the form of your graph (line, bars, scattered, etc.).
- You could filter by various settings.
- When satisfied with your graph, you could pin it to the dashboard.

Metrics in Azure

- Azure Monitor enables you to consume telemetry to gain visibility into the performance and health of your workloads on Azure. The most important type of Azure telemetry data is the metrics (also called performance counters) emitted by most Azure resources.
- Metrics are a valuable source of telemetry and enable you to do the following tasks:
  - **Track the performance** of your resource (such as a VM, website, or logic app) by plotting its metrics on a portal chart and pinning that chart to a dashboard.
  - **Get notified of an issue** that impacts the performance of your resource when a metric crosses a certain threshold.
  - **Configure automated actions**, such as autoscaling a resource or firing a runbook when a metric crosses a certain threshold.
  - **Perform advanced analytics** or reporting on performance or usage trends of your resource.
  - **Archive** the performance or health history of your resource **for compliance or auditing** purposes.

Characteristics of Metrics

Metrics have the following characteristics:

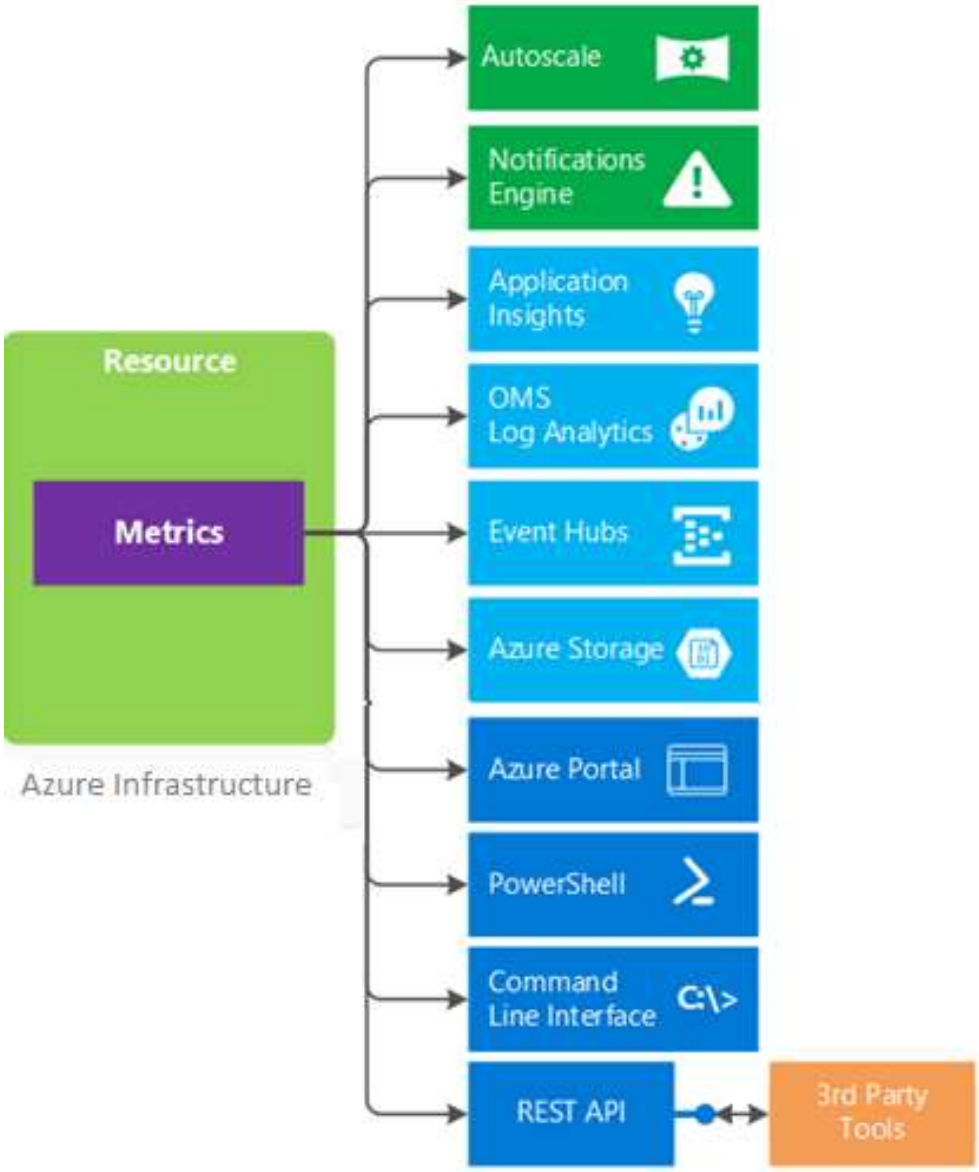
- All metrics have **one-minute frequency**. You receive a metric value every minute from your resource, giving you near real-time visibility into the state and health of your resource.
- Metrics are **available immediately**. You don't need to opt in or set up additional diagnostics.
- You can access **30 days of history** for each metric. You can quickly look at the recent and monthly trends in the performance or health of your resource.
- Some metrics can have name-value pair attributes called **dimensions**. These enable you to further segment and explore a metric in a more meaningful way.

Usage of Metrics

- We could configure a metric **alert rule to send a notification or takes automated action** when the metric crosses the threshold that you have set. Autoscale is a special automated action that enables scaling out of resources to meet incoming requests or loads on your website or computing resources. We can configure an Autoscale setting rule to scale in or out based on a metric crossing a threshold.
- **We could route** all metrics to Application Insights or Log Analytics (OMS) to enable instant analytics, search, and custom alerting on metrics data from your resources. You can also stream metrics to an Event Hub, enabling you to then route them to Azure Stream Analytics or to custom apps for near-real time analysis.
- **We could archive metrics to storage** for longer retention or use them for offline reporting. We could route Azure metrics to Azure Blob storage when we configure diagnostic settings for our resource.
- We could discover, access, and **view all metrics** via the Azure portal when we select a resource and plot the metrics on a chart.
- **We could consume** the metrics via the Azure Monitor REST APIs.
- **We could query** metrics by using the PowerShell cmdlets or the Cross-Platform REST API.

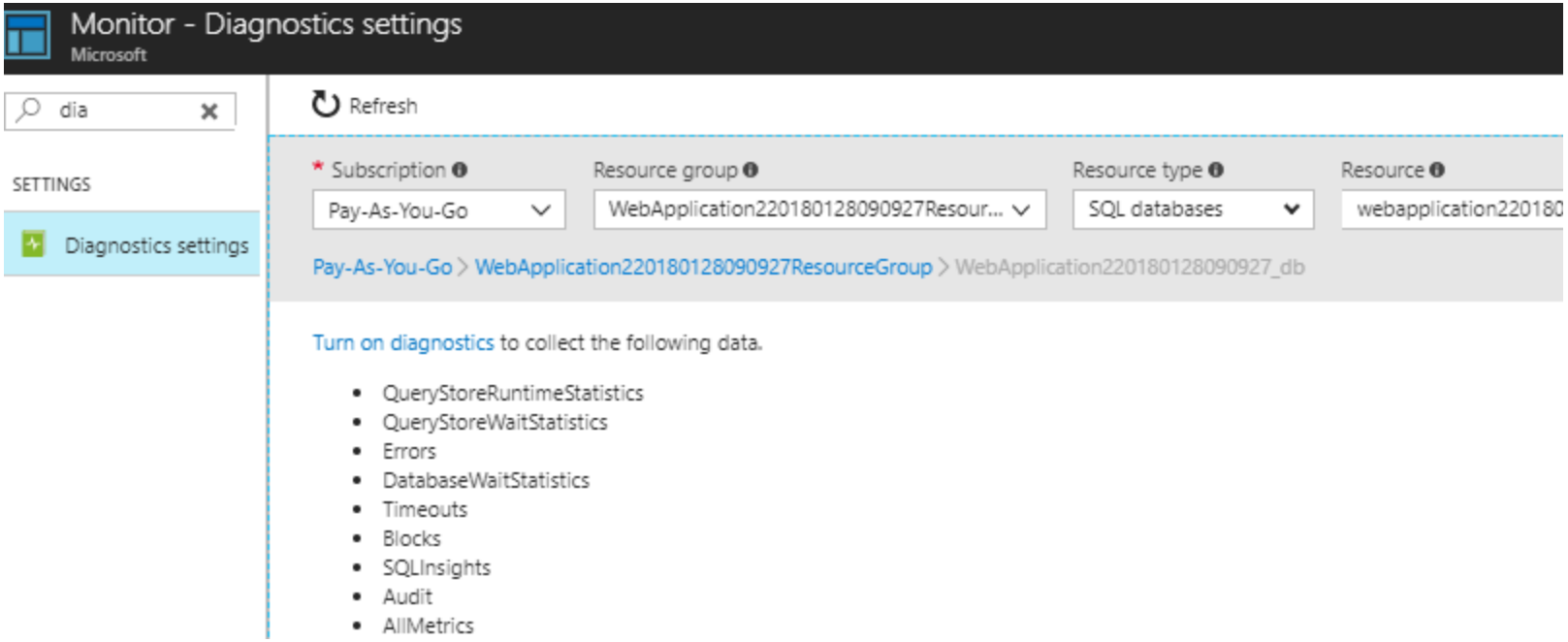
Azure Consumers of Metrics

- All resources and services generate metrics.
- Many services are equipped to consume metrics generated by other Azure services.



Diagnostic Settings

- Before you set Diagnostics Settings, you need a storage account. Create on if you do not have it, already.
- While on Monitor blade, in the search field, enter Diagnostics. Select Diagnostics Settings.
- Select a resource group, resource type and then select Turn on diagnostics



Send Metrics to Log Analyzer

- If you select to send collected metrics to the Log Analyzer, you have to set it up

Diagnostics settings

Save Discard Delete

* Name
diagnosticssettings

Archive to a storage account
Stream to an event hub
Send to Log Analytics

Log Analytics
Configure

LOG

QueryStoreRuntimeStatistics
QueryStoreWaitStatistics
Errors
DatabaseWaitStatistics
Timeouts
Blocks
SQLInsights
Audit

METRIC

AllMetrics

OMS Workspaces

Create New Workspace

None

OMS Workspace
Create new or link existing one created in OMS Por

Create New Link Existing

* OMS Workspace
zdjordjediagnostics

* Subscription
Pay-As-You-Go

* Resource group
Create new Use existing
WebApplication220180128090927Resourc

* Location
East US

* Pricing tier
Free

Pin to dashboard

OK

- After entering required fields, select OK below OMS Workspace.
- Once you are back to Diagnostic settings, Save icon will come in focus. Select it.

Log Analytics

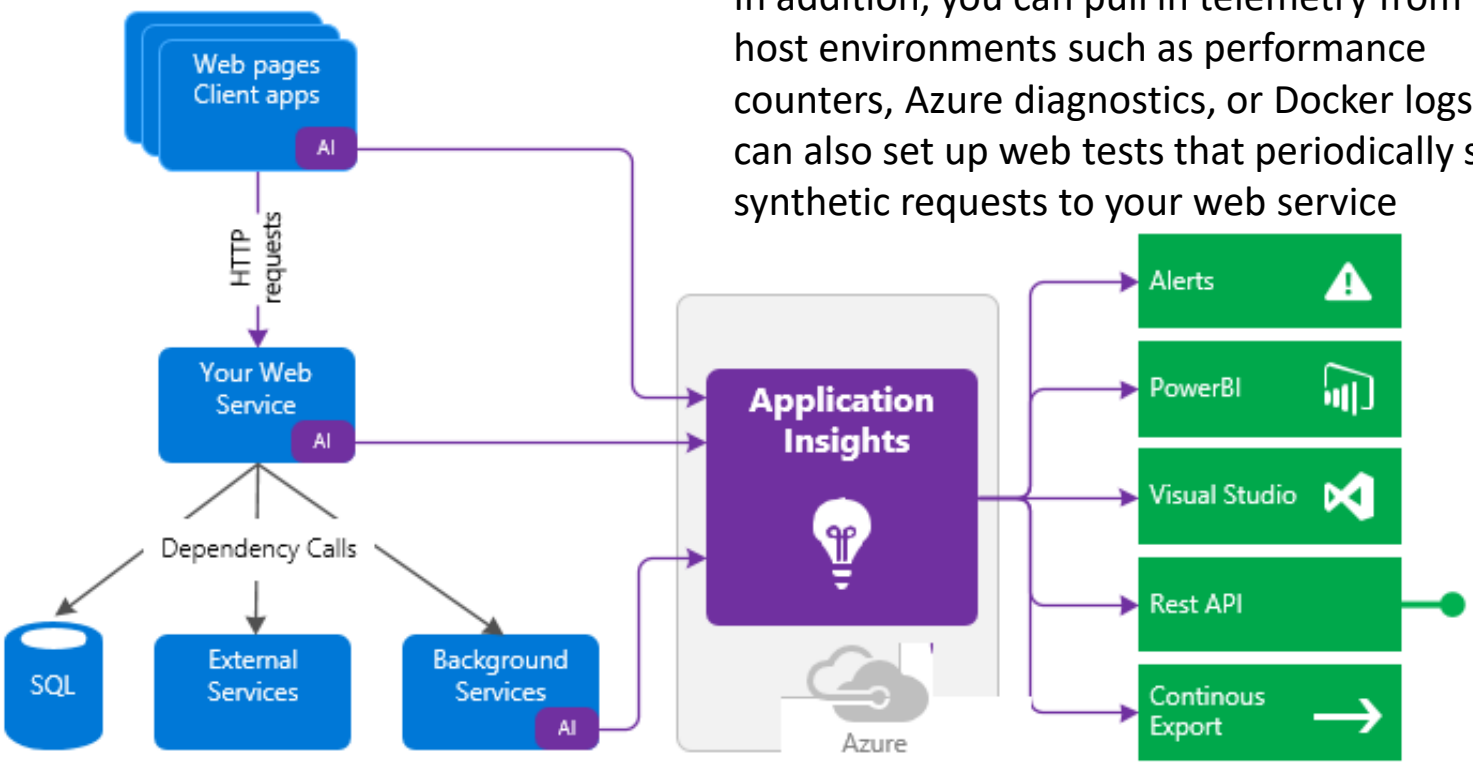
Application Insights

- Application Insights is an extensible Application Performance Management (APM) service for web developers on multiple platforms.
- Application Insights monitors your live web application. It will automatically detect performance anomalies.
- Application Insights includes powerful analytics tools to help you diagnose issues and to understand what users actually do with your app.
- Application Insights is designed to help you continuously improve performance and usability. It works for apps on a wide variety of platforms including .NET, Node.js and J2EE, hosted on-premises or in the cloud.
- Application Insights integrates with your DevOps process, and has connection points to a variety of development tools.
- Application Insights can monitor and analyze telemetry from mobile apps by integrating with Visual Studio App Center.

App Insights Mechanisms

- We install a small instrumentation package in our application, and set up an Application Insights resource in the Microsoft Azure portal.
- The instrumentation monitors our app and sends telemetry data to the portal. (The application can run anywhere - it doesn't have to be hosted in Azure.)
- You can instrument not only the web service application, but also any background components, and the JavaScript in the web pages themselves.
- The impact on your app's performance is very small.

- In addition, you can pull in telemetry from the host environments such as performance counters, Azure diagnostics, or Docker logs. You can also set up web tests that periodically send synthetic requests to your web service



What App Insights Monitors

Application Insights is aimed at the development team, to help you understand how your app is performing and how it's being used. It monitors:

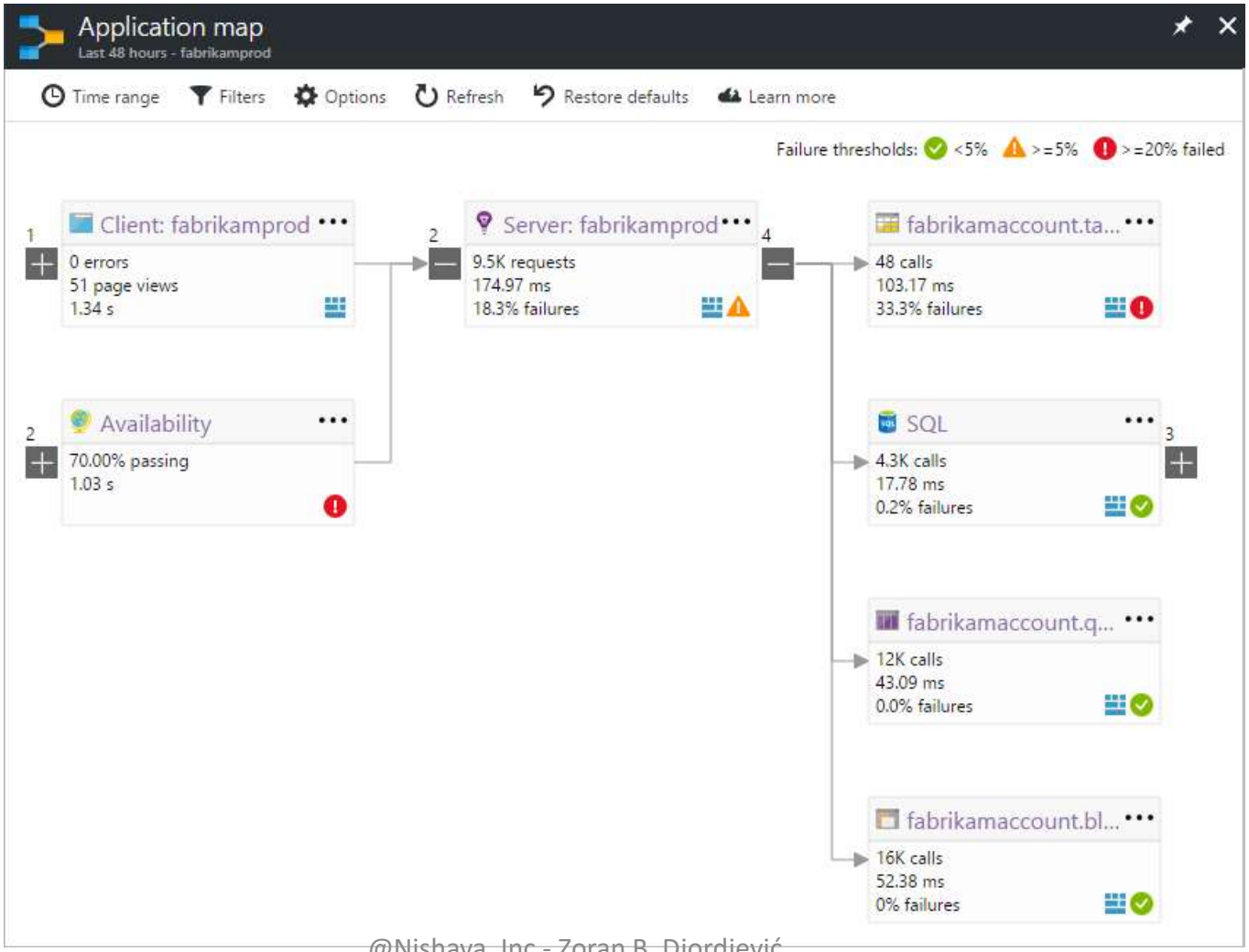
- **Request rates, response times, and failure rates** - Find out which pages are most popular, at what times of day, and where your users are. See which pages perform best. If your response times and failure rates go high when there are more requests, then perhaps you have a resourcing problem.
- **Dependency rates, response times, and failure rates** - Find out whether external services are slowing you down.
- **Exceptions** - Analyse the aggregated statistics, or pick specific instances and drill into the stack trace and related requests. Both server and browser exceptions are reported.
- **Page views and load performance** - reported by your users' browsers.
- **AJAX calls** from web pages - rates, response times, and failure rates.
- **User and session counts**.
- **Performance counters** from your Windows or Linux server machines, such as CPU, memory, and network usage.
- **Host diagnostics** from Docker or Azure.
- **Diagnostic trace logs** from your app - so that you can correlate trace events with requests.
- **Custom events and metrics** that you write yourself in the client or server code, to track business events such as items sold or games won.

Smart detection and manual alerts

- Automatic alerts adapt to your app's normal patterns of telemetry and trigger when there's something outside the usual pattern.
- You can also set alerts on particular levels of custom or standard metrics.
- Smart Detection automatically warns you of potential performance problems in your web application. It performs proactive analysis of the telemetry that your app sends to Application Insights. If there is a sudden rise in failure rates, or abnormal patterns in client or server performance, you get an alert. This feature needs no configuration. It operates if your application sends enough telemetry.
- You can access Smart Detection alerts both from the emails you receive, and from the Smart Detection blade.

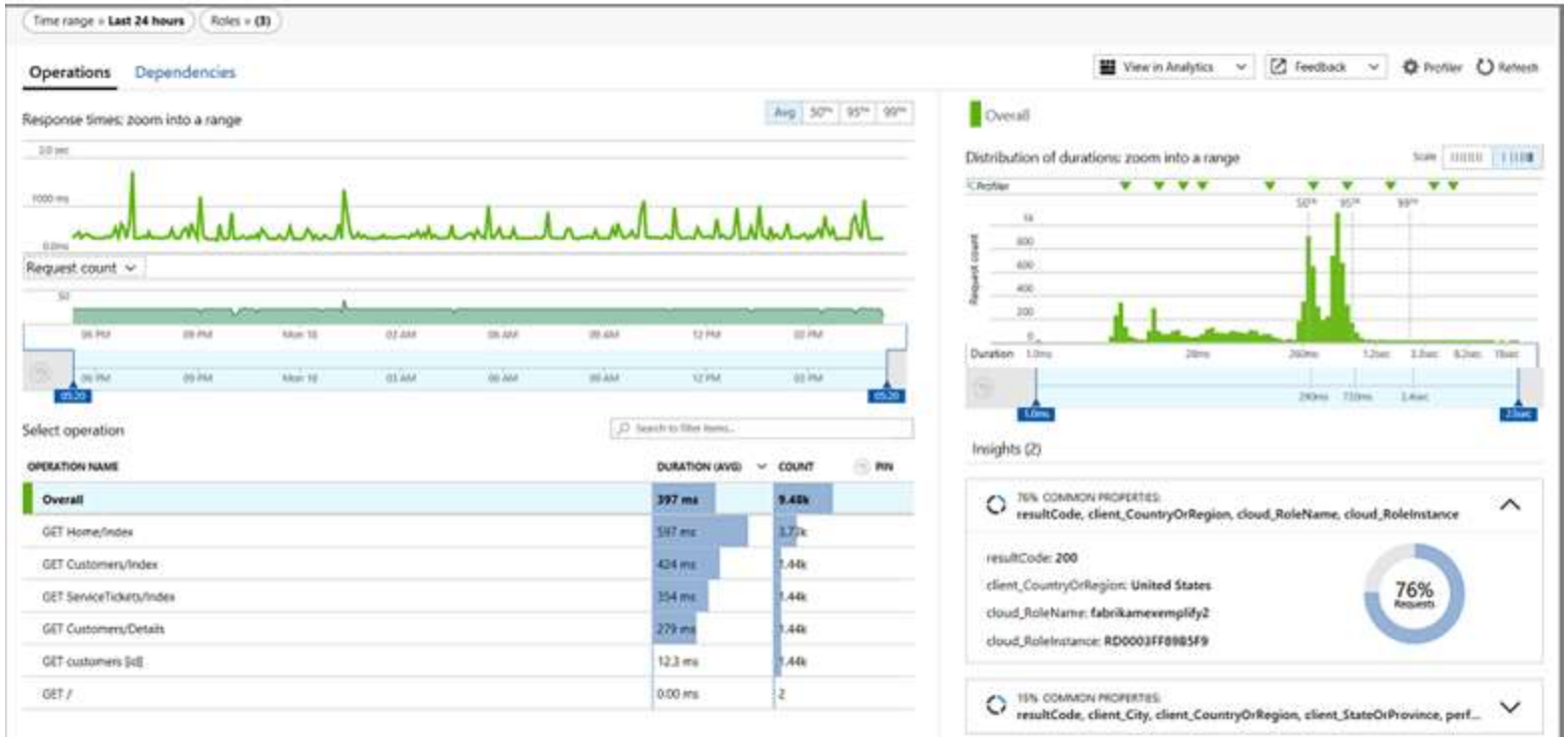
Application Map

- In Azure Application Insights, Application Map is a visual layout of the dependency relationships of your application components.
- Each component shows KPIs such as load, performance, failures, and alerts, to help you discover any component causing a performance issue or failure.



Profiler

- *Profiler feature of Application Insights is generally available for Azure App Service and is in preview for Azure compute resources.* Profiler allow us to find out how much time is spent in each method in our live web application.
- The Application Insights profiling tool shows detailed profiles of live requests that were served by your app, and highlights the *hot path* that uses the most time. The profiler currently works for ASP.NET and ASP.NET core web apps running on Azure App Service, in at least the **Basic** service tier.

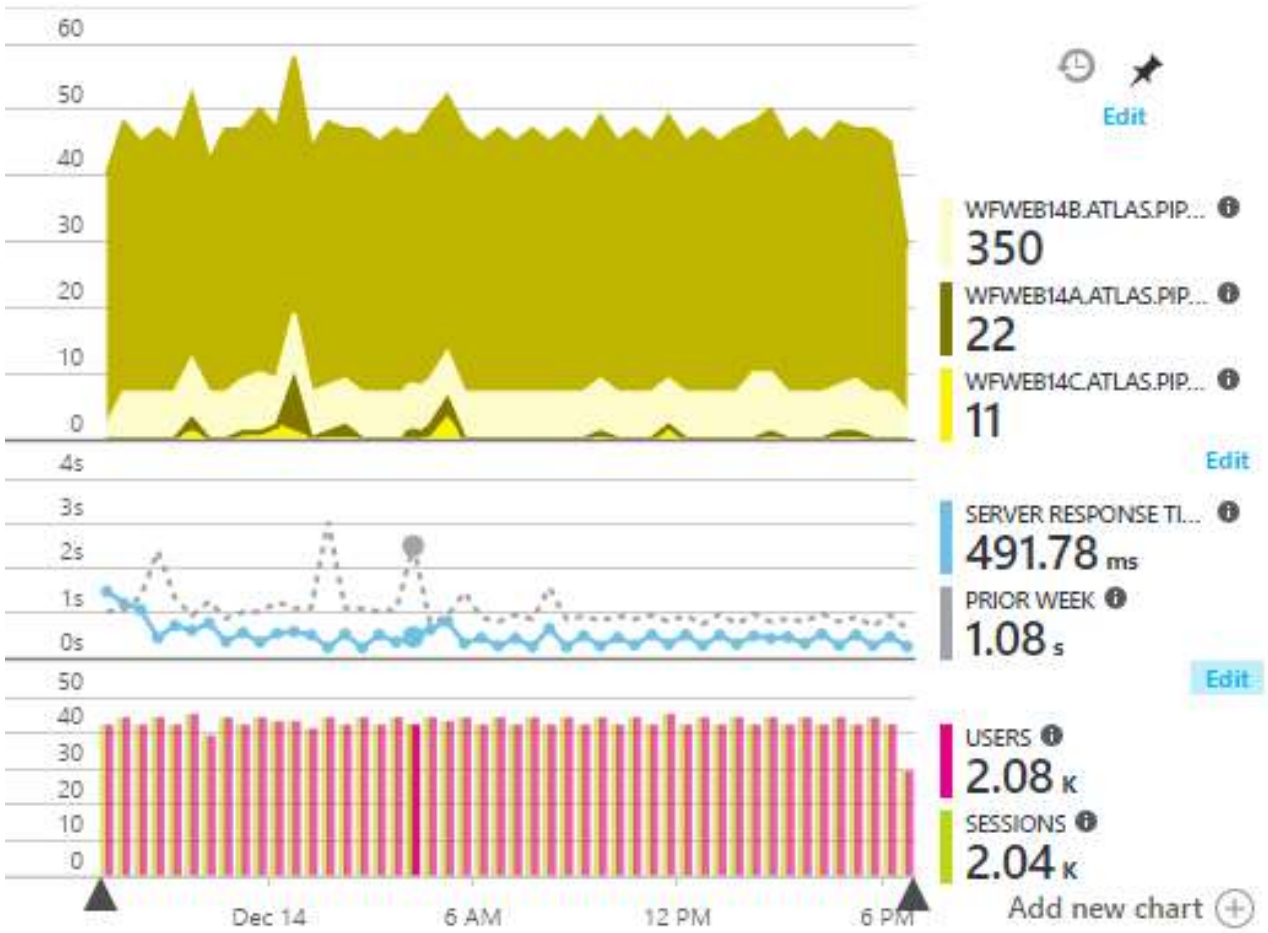


Usage analysis with Application Insights

- Usage Analysis tells you which features of your web or mobile app are most popular? Do your users achieve their goals with your app? Do they drop out at particular points, and do they return later?

Exploring Metrics in Application Insights

- Metrics in Application Insights are measured values and counts of events that are sent in telemetry from your application. Metrics and event counts are displayed in charts of aggregated values such as sums, averages, or counts.



Live Metrics Stream

- With Application insights you can impose Metrics generation with 1-sec latency
- With Live Metrics Stream, you can:
  - Validate a fix while it is released, by watching performance and failure counts.
  - Watch the effect of test loads, and diagnose issues live.
  - Focus on particular test sessions or filter out known issues, by selecting and filtering the metrics you want to watch.
  - Get exception traces as they happen.
  - Experiment with filters to find the most relevant KPIs.
  - Monitor any Windows performance counter live.
  - Easily identify a server that is having issues, and filter all the KPI/live feed to just that

Live Metric Streams

Other Features of Application Insights

Application Insights is a very rich framework and comes with several additional features:

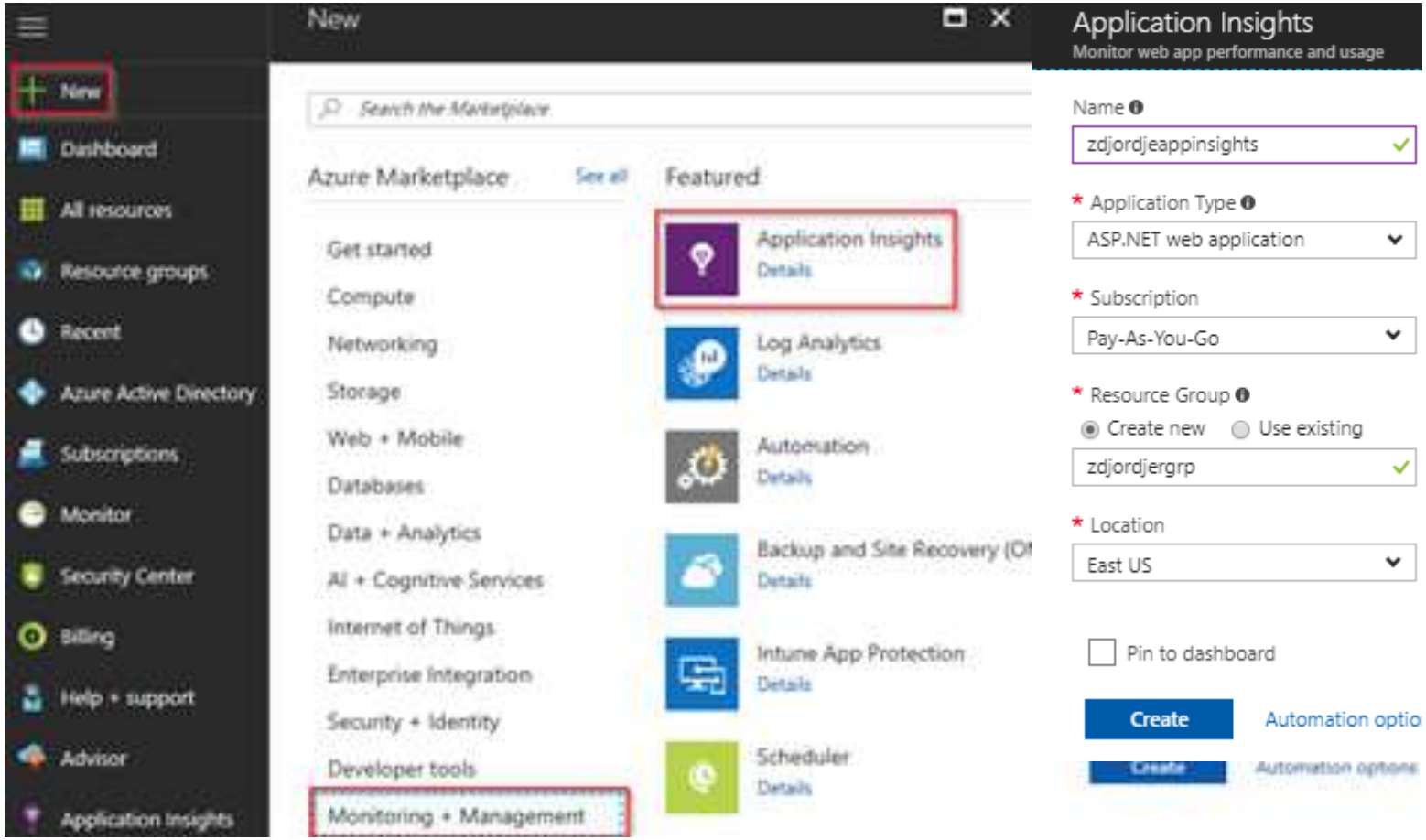
- **Diagnostic search for instance data**
  Search and filter events such as requests, exceptions, dependency calls, log traces, and page views.
- **Metrics Explorer for aggregated data**
  Explore, filter, and segment aggregated data such as rates of requests, failures, and exceptions; response times, page load times.
- **Dashboards**
  Mash up data from multiple resources and share with others. Great for multi-component applications, and for continuous display in the team room.
- **Analytics**
  Answer tough questions about your app's performance and usage by using this powerful query language.
- **Visual Studio and Eclipse Integration**
  See performance data in the code. Go to code from stack traces
- **Snapshot debugger**
  Debug snapshots sampled from live operations, with parameter values.
- **REST API**
  Write code to run queries over your metrics and raw data.
- **Continuous export**
  Bulk export of raw data to storage as soon as it arrives.

Prerequisites for running Application Insights

- To complete this quickstart:
- Install Visual Studio 2017 with the following workloads:
  - ASP.NET and web development
  - Azure development
- Install .NET Core 2.0 SDK
- You will need an Azure subscription and an existing .NET Core web application.
- If you don't have a ASP.NET Core web application, you can create one.

Enable Application Insights in Portal

- Application Insights can gather telemetry data from any internet-connected application, regardless of whether it's running on-premises or in the cloud. Use the following steps to start viewing this data.
- Select **New** > **Monitoring + Management** > **Application Insights**



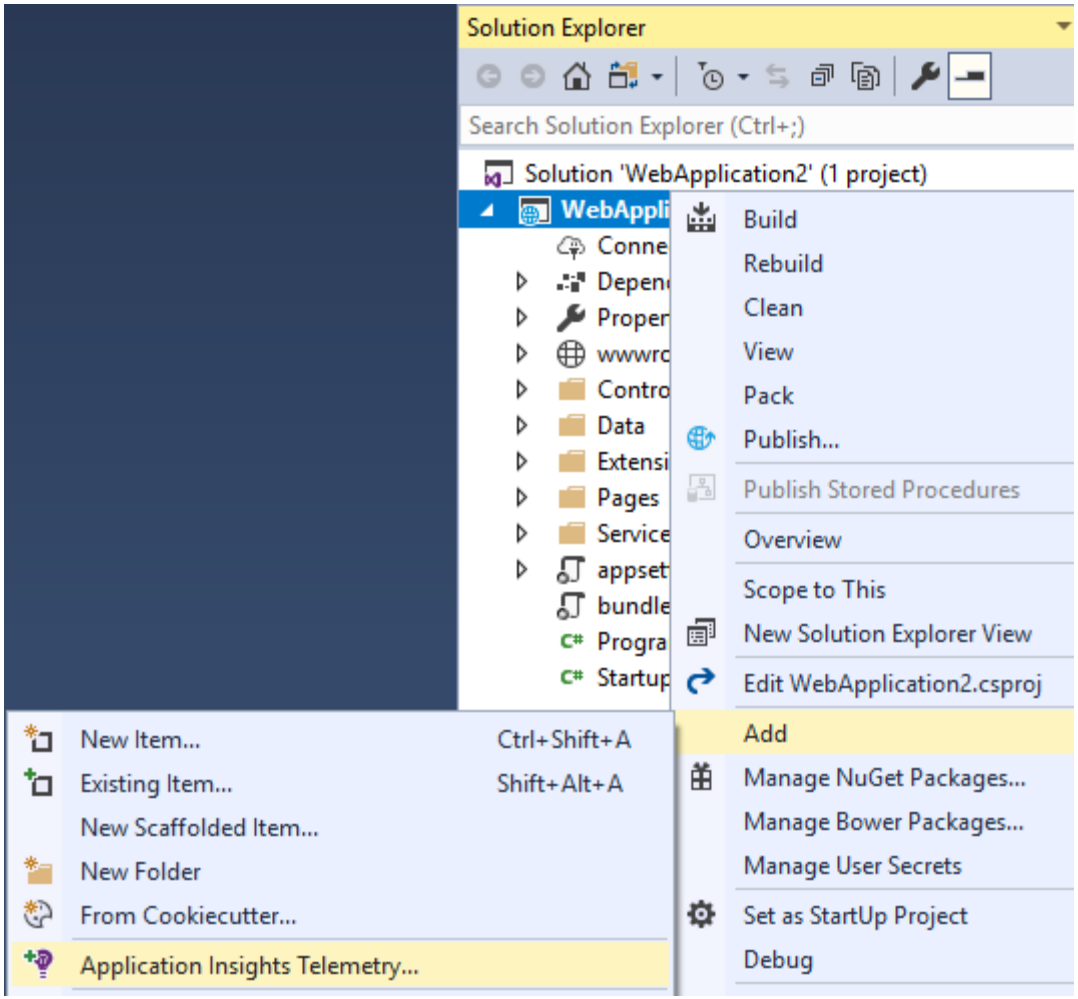
Configure App Insights in Portal

- In the box that appears enter the following values

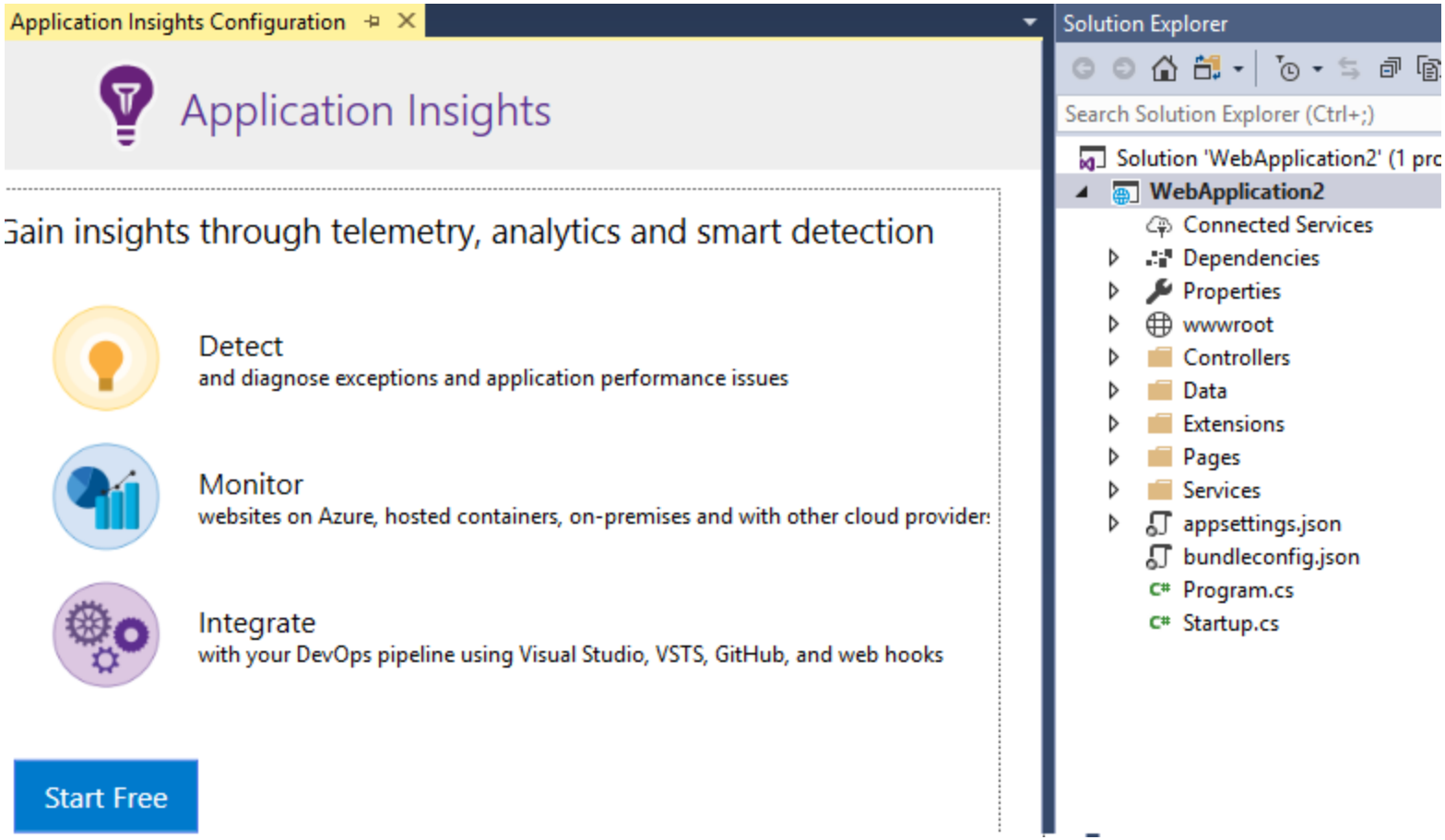
Settings	Value	Description
Name	Globally Unique Value	Name that identifies the app you are monitoring
Application Type	ASP.NET web application	Type of app you are monitoring
Resource Group	myResourceGroup	Name for the new resource group to host App Insights data
Location	East US	Choose a location near you, or near where your app is hosted

Configure App Insights SDK

- Open your ASP.NET Core Web App **project** in Visual Studio >
- Right-click on the AppName in the **Solution Explorer** > Select **Add** > **Application Insights Telemetry**.



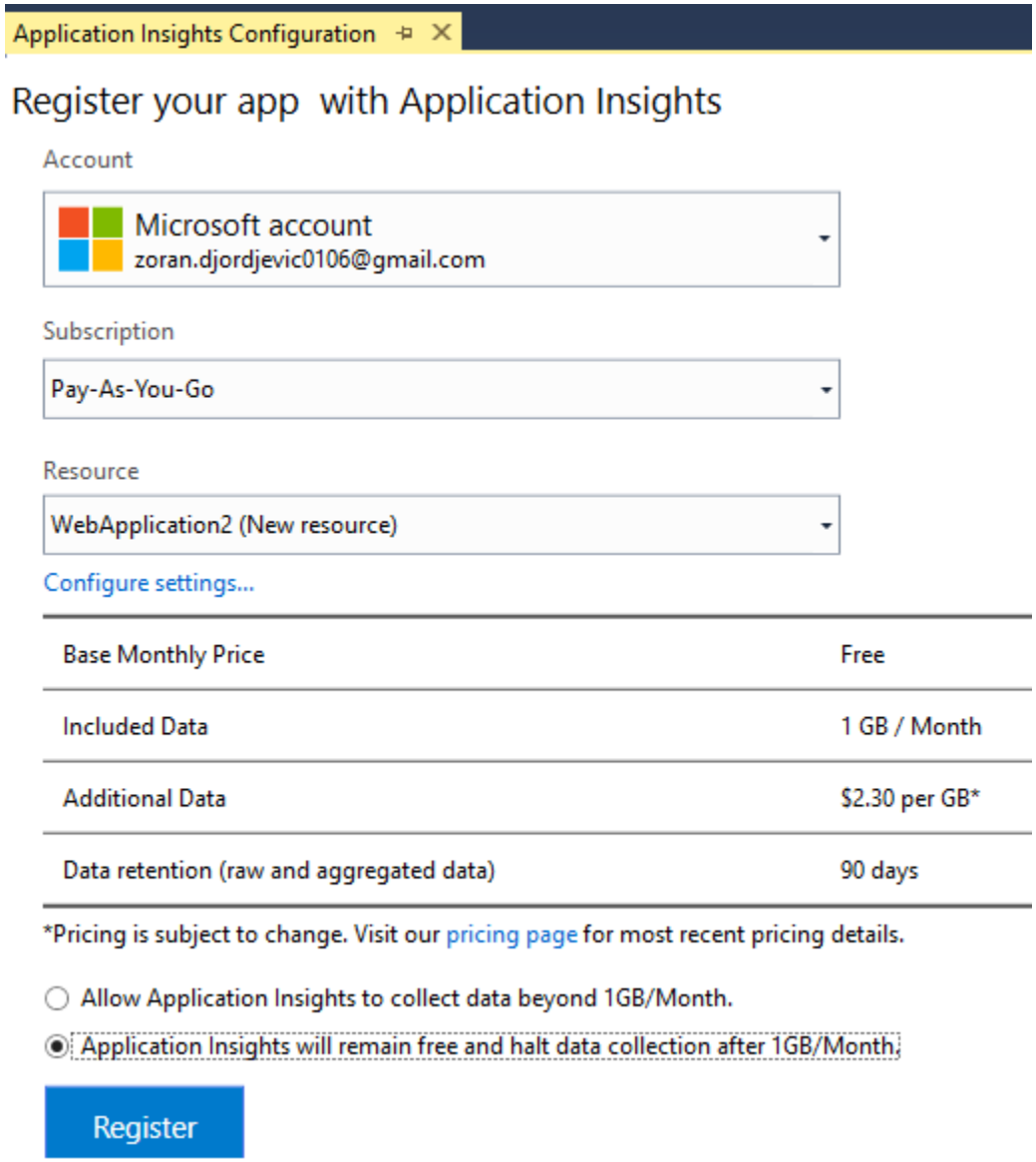
Start Free


Application Insights Configuration
Application Insights
Gain insights through telemetry, analytics and smart detection
Detect
and diagnose exceptions and application performance issues
Monitor
websites on Azure, hosted containers, on-premises and with other cloud provider
Integrate
with your DevOps pipeline using Visual Studio, VSTS, GitHub, and web hooks
Start Free
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WebApplication2' (1 pro
WebApplication2
Connected Services
Dependencies
Properties
wwwroot
Controllers
Data
Extensions
Pages
Services
appsettings.json
bundleconfig.json
Program.cs
Startup.cs

Register your App with App Insights

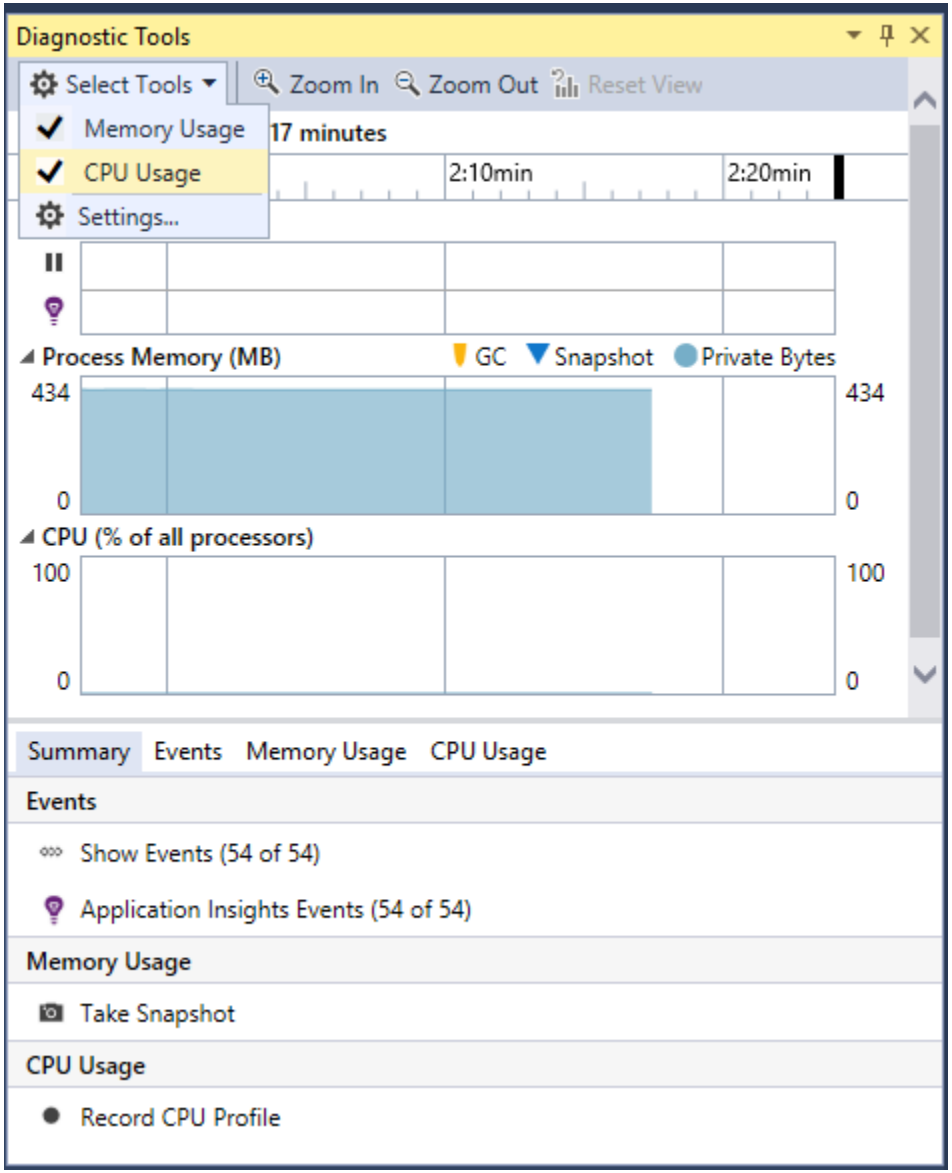
- Select the **Existing resource** you created in the Azure portal
- Perhaps change the bottom selection to prevent very large data being collected.
- Select Register.

- To start collecting data,
- Select **Debug** > **Start without Debugging** (Ctrl+F5) to Launch your app
- It takes 3-5 minutes before data begins appearing in the portal.
- If this app is a low-traffic test app, keep in mind that most metrics are only captured when there are active requests or operations.



Diagnostic Tools

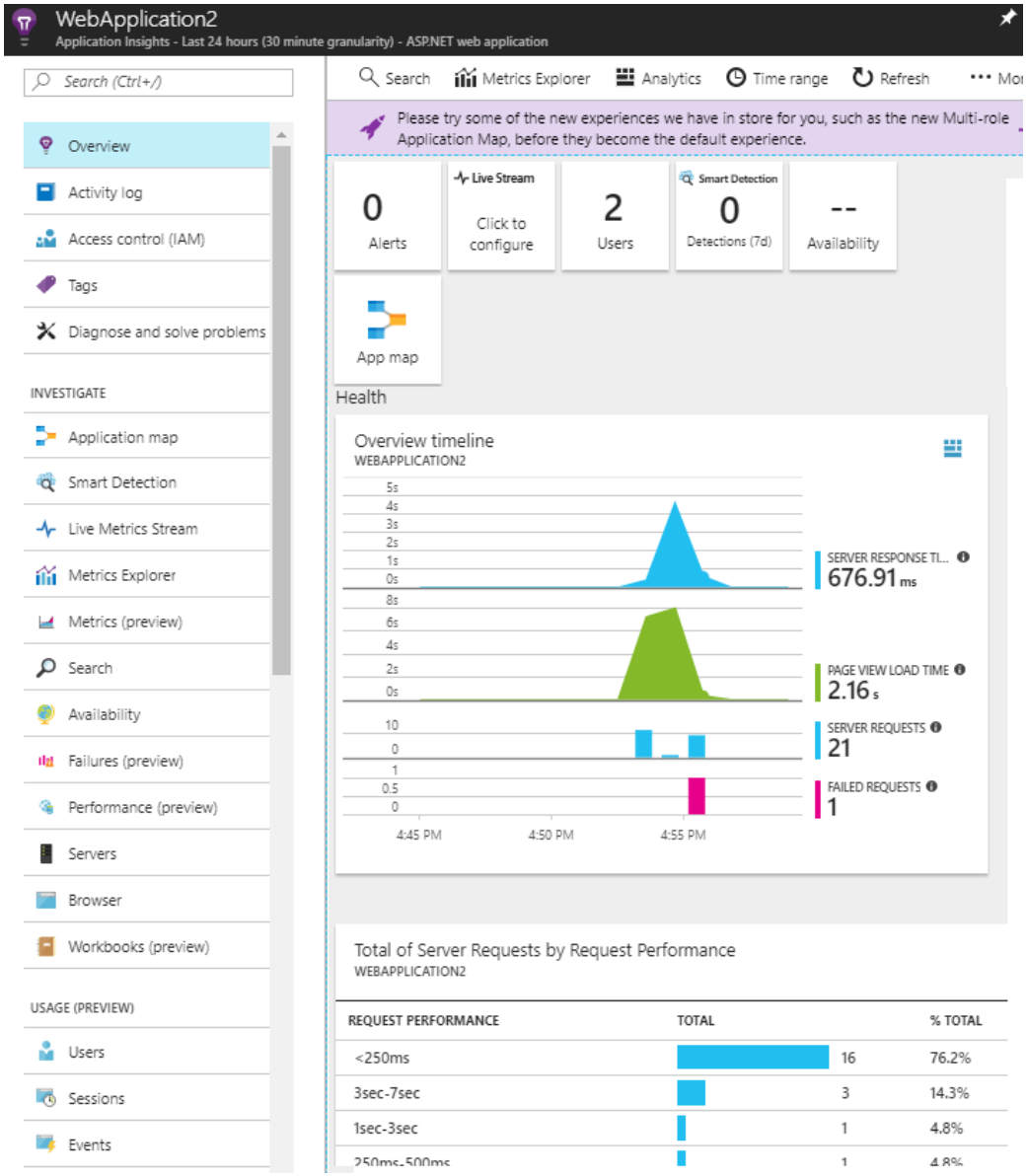
- You have to Publish your app to Azure, again.
- Once you start your application, Diagnostic Tools window will appear.



Start Monitoring in Azure Portal

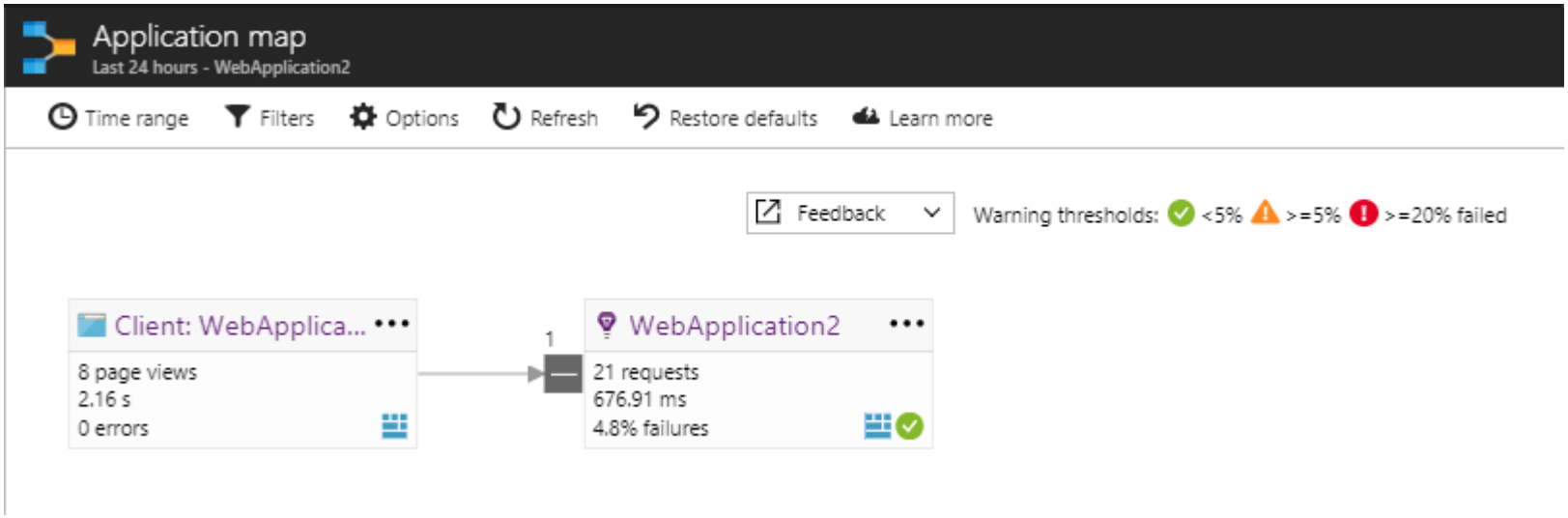
- You can now reopen the Application Insights **Overview** page in the Azure portal by selecting **Project** > **App lication Insights** > **Open Application Insights Portal**, to view details about your currently running application.

- Select App map for a visual layout of the dependency relationships between your application components. Each component shows KPIs such as load, performance, failures, and alerts.



App Map

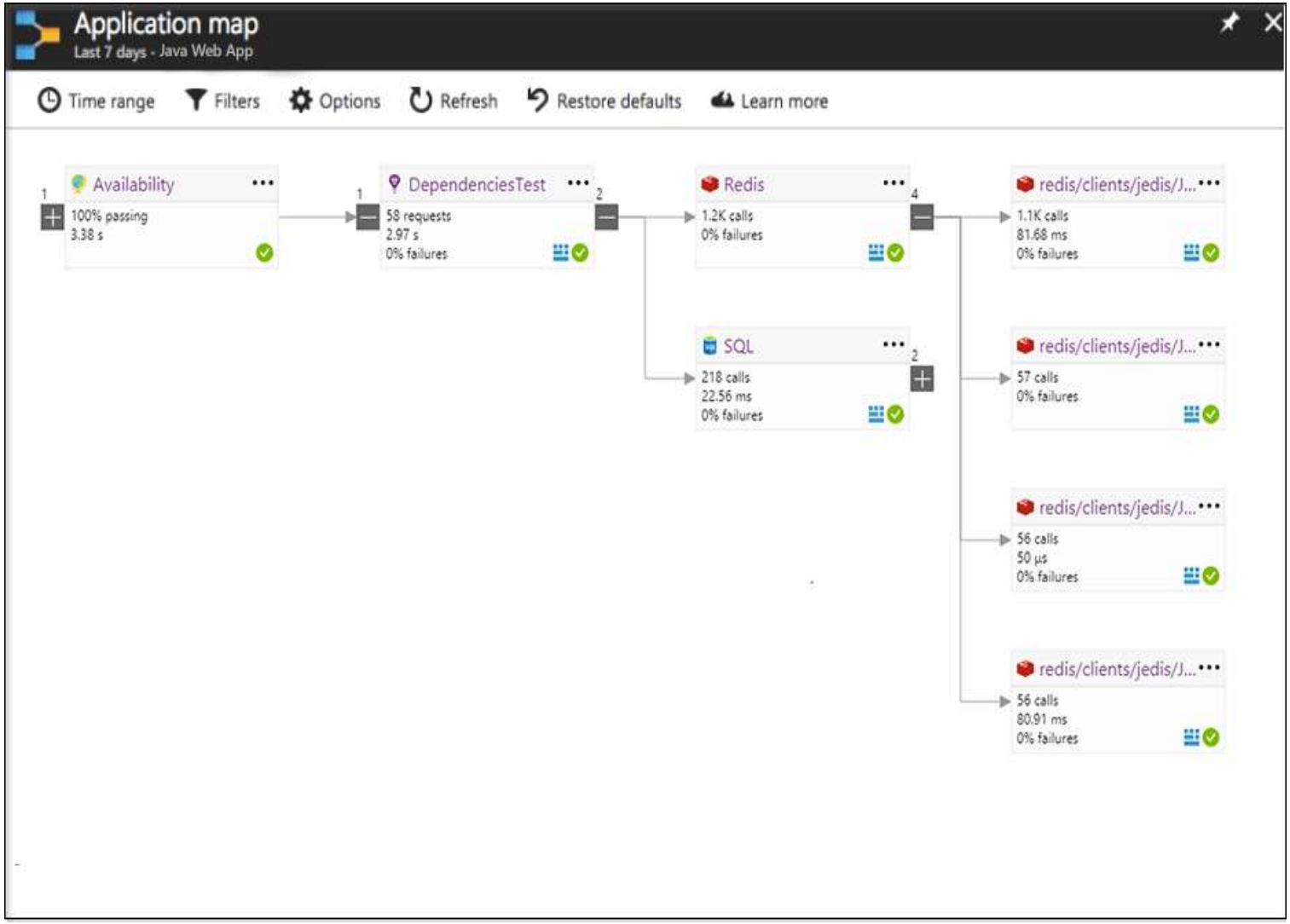
- Our application is rather simple. Yours will not be:



- Click on the App Analytics icon . This opens Application Insights Analytics, which provides a rich query language for analyzing all data collected by Application Insights.
- In this case, a query is generated for you that renders the request count as a chart.

Another App Map

- On another application the App Map might look like the following:



App Insights Query Window

```
// Request count
let startTime = todatetime("2017-08-16T16:59:06.029Z");
let endTime = todatetime("2017-08-17T16:59:06.029Z");
let grain = 30m;
requests
| where timestamp > startTime and timestamp <= endTime
| where client_Type=="PC"
| summarize requestCount=sum(itemCount) by bin(timestamp, grain)
| render timechart
```

Completed .NET Core Dev 00:00:03.574 48 records

TABLE CHART Line Timestamp RequestCount Sum

For Smart Diagnostics click on a highlighted data variation

Page View Load Time

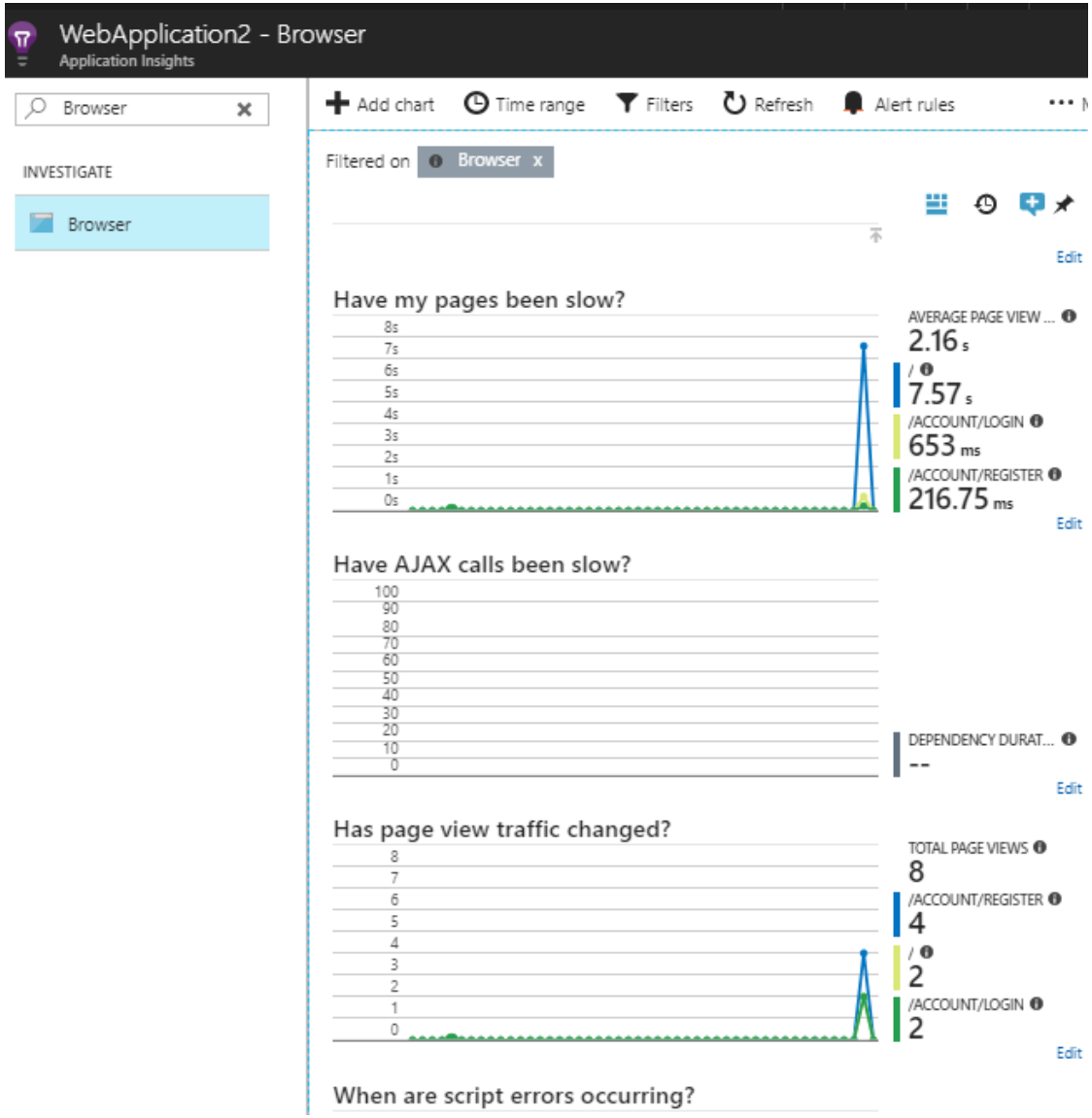
- To enable the **Page View Load Time** chart to populate with **client-side telemetry** data, add this script to each page that you want to track:

```
<!--
To collect end-user usage 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.src=c
onfig.url||"https://az416426.vo.msecnd.net/scripts/a/ai.0.js";u.getElementsByTagN
ame(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","Dependency"
];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.disableExce
ptionTracking||(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>
:
```

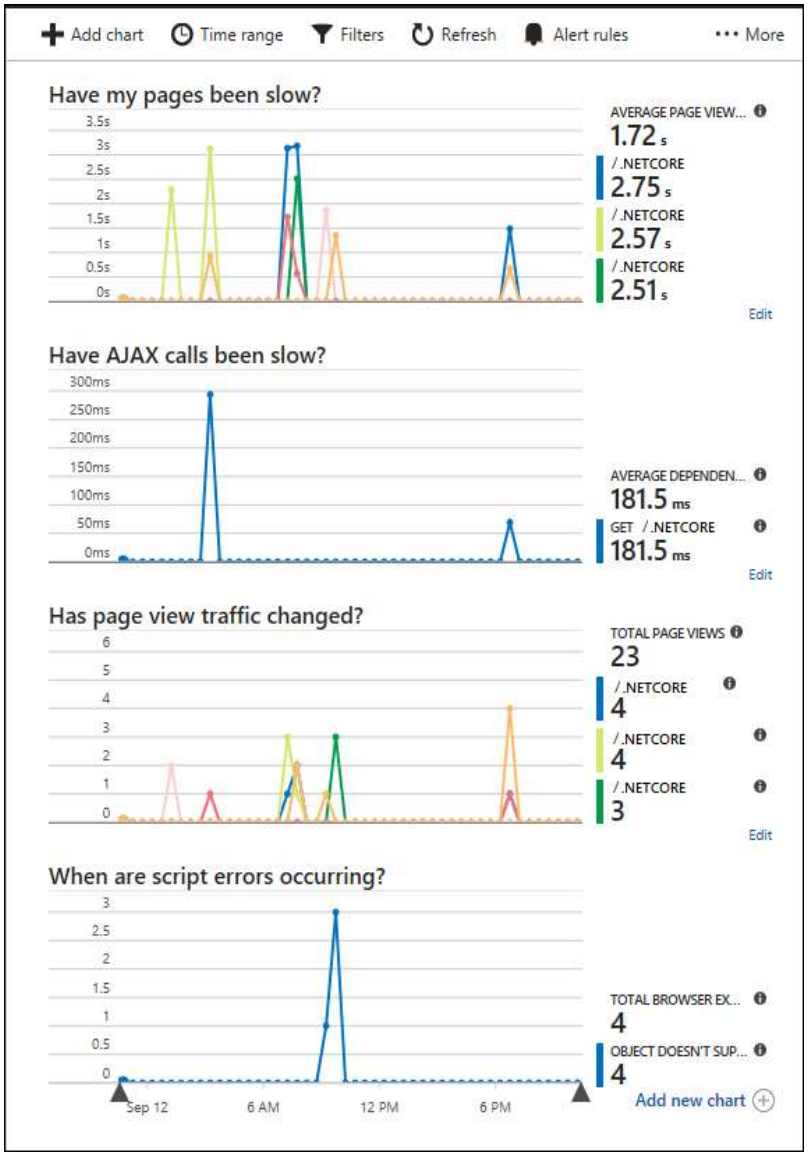
Page Response time

- In the search field enter Browser.
- Resulting view will show to you page response times and few other important HTML page related statistics



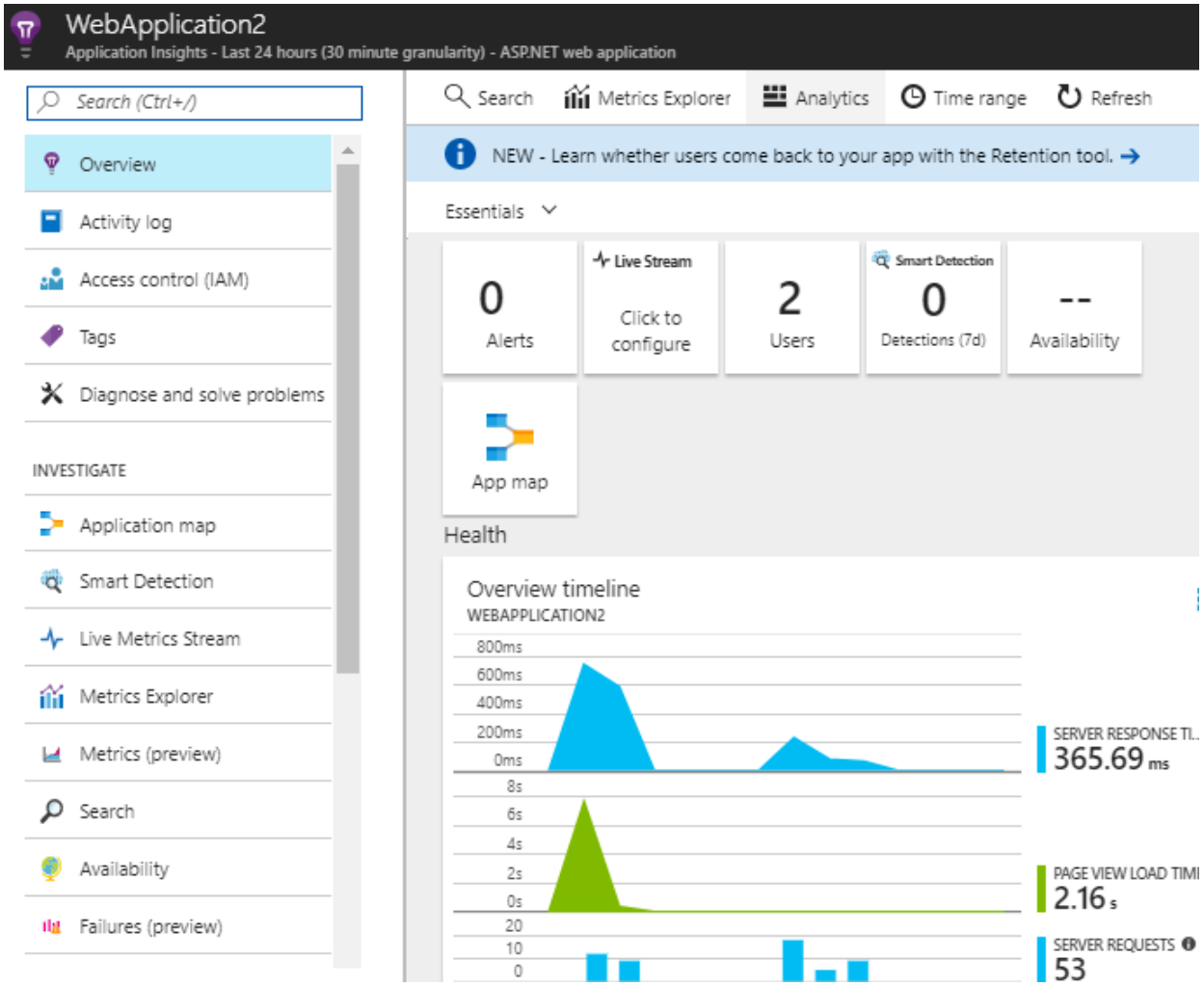
Page Response Time

- On another application this might look like

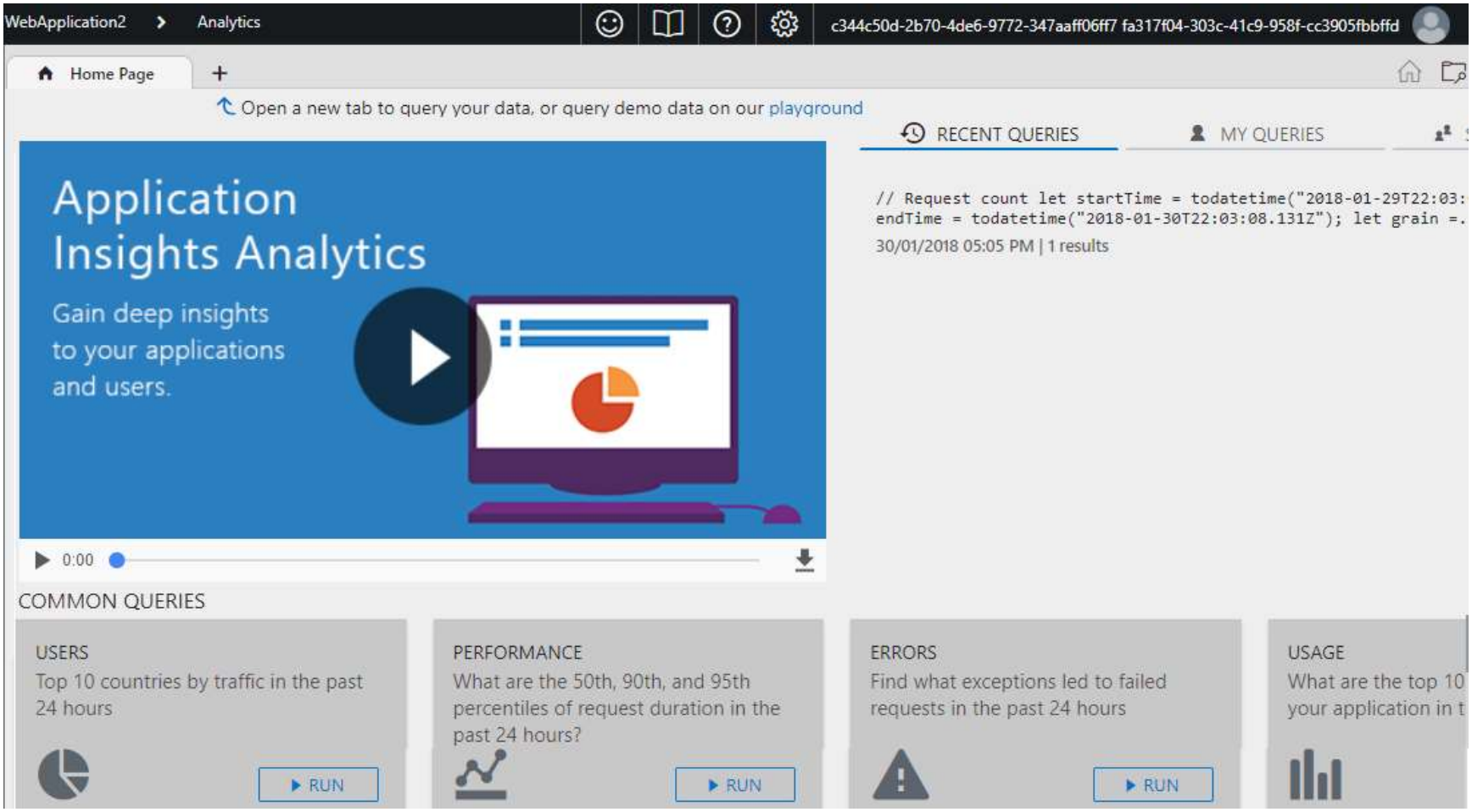


Analytics in Application Insight

- In the Search field of the Application Insights pane, enter Overview and then select Analytics on the bar atop of charts



App Insights Analytics

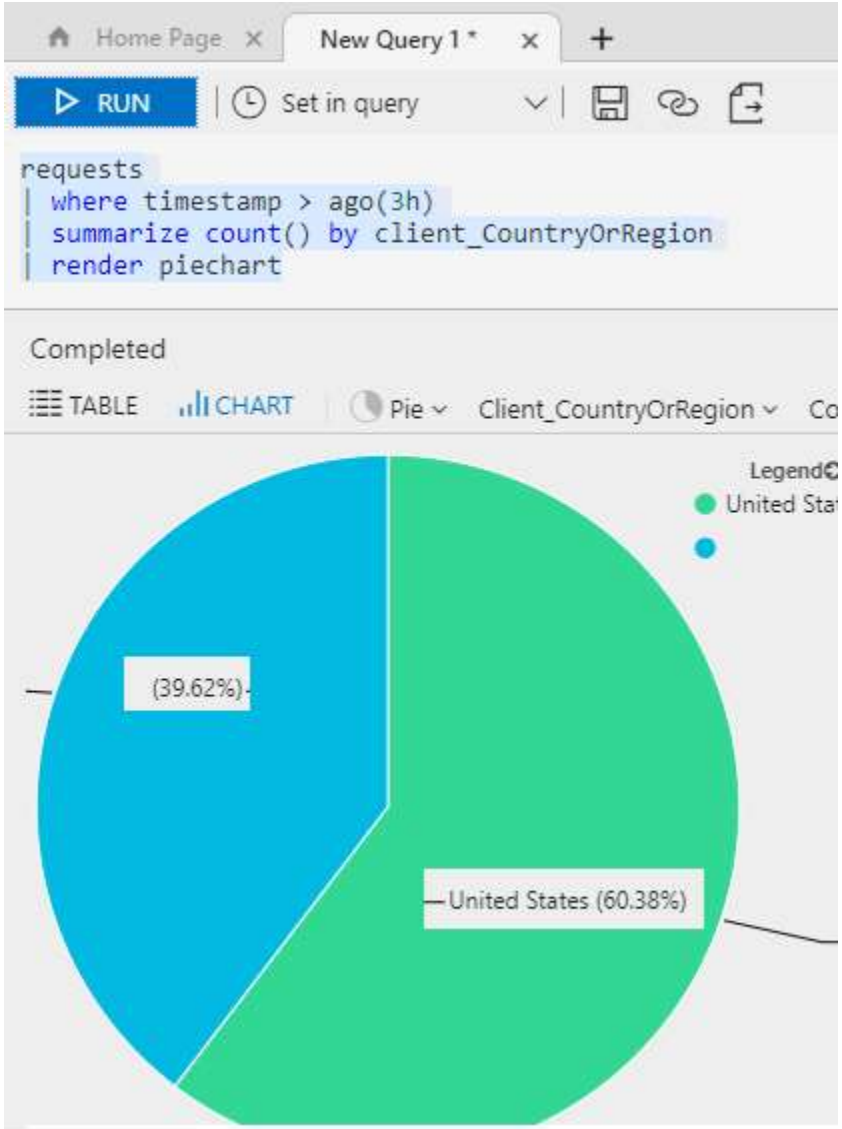


Select recent queries and hit it twice.

Analytics Queries

- A typical query starts with a table name followed by a series of *operators* separated by |. For example, let's find out how many requests our app received from different countries, during the last 3 hours.
- We start with the table name *requests* and add piped elements as needed. First we define a time filter to review only records from the last 3 hours. We then count the number of records per country (that data is found in the column *client_CountryOrRegion*). Finally, we render the results in a pie chart.

```
requests
| where timestamp > ago(3h)
| summarize count() by client_CountryOrRegion
| render piechart
```



Query Language

- The query language has many attractive features:
  - Filter your raw app telemetry by any fields, including your custom properties and metrics.
  - Join multiple tables – correlate requests with page views, dependency calls, exceptions and log traces.
  - Powerful statistical aggregations.
  - Immediate and powerful visualizations.
  - REST API that you can use to run queries programmatically, for example from PowerShell.

Another simple query

- Let us execute query that reads:

```
requests
| search "azurewebsites"
```

- This query searches the *requests* table for records that contain the term "azurewebsites".
- Queries can start with either a table name or a *search* command. The above example starts with the table name *requests*, which defines the scope of the query. The pipe (|) character separates commands, so the output of the first one in the input of the following command. You can add as many commands as required Another way to write that query would be.
- In this example, *search* is scoped to the *requests* table, and in it search for all records that contain "azurewebsites".
- When running a query, pay attention to the following:
- Line breaks - a single break makes your query clearer. Multiple line-breaks split it into separate queries.
- Cursor - be sure to put the cursor inside or at the end of the query before executing it.
- Time range - A time range of "last 24 hours" is set by default. To use a different range, use the time-picker (located next to the *Go* button) or add an explicit time range filter to your query.
- Understand the schema
- The schema is a collection of tables, grouped visually under a logical category. In the screenshot below we see the title "Application Insights" which covers different tables, all related to the same product. For example, *traces* and *customEvents* are names of tables:

Query Results

- It appears that you can see all calls to our Web site.

RUN | Last 24 hours

```
requests
| search "azurewebsites"
```

Completed. Showing results from the last 24 hours. WebApplication2 00:00:00.625

TABLE CHART Columns Display time (UTC+

Drag a column header and drop it here to group by that column

$table	timestamp [UTC]	id	name	url	success
requests	2018-01-30T22:24:05.572	|de52559b-46d.	GET /Index	http://webapplication220180128090927.azurewebsites.n…	True
requests	2018-01-30T22:24:07.038	|de52559c-46d.	GET /css/site.min.css	http://webapplication220180128090927.azurewebsites.n…	True
requests	2018-01-30T22:24:07.119	|de52559d-46d.	GET /images/banner1.svg	http://webapplication220180128090927.azurewebsites.n…	True
requests	2018-01-30T22:24:07.140	|de52559e-46d.	GET /js/site.min.js	http://webapplication220180128090927.azurewebsites.n…	True
requests	2018-01-30T22:24:07.142	|de52559f-46d.	GET /images/banner3.svg	http://webapplication220180128090927.azurewebsites.n…	True
requests	2018-01-30T22:24:07.151	|de5255a0-46d.	GET /images/banner2.svg	http://webapplication220180128090927.azurewebsites.n…	True
requests	2018-01-30T22:24:07.154	|de5255a1-46d.	GET /images/banner4.svg	http://webapplication220180128090927.azurewebsites.n…	True
requests	2018-01-30T22:24:07.240	|de5255a2-46d.	GET /favicon.ico	http://webapplication220180128090927.azurewebsites.n…	True
requests	2018-01-30T22:24:12.347	|de5255a3-46d.	GET /Account/Login	http://webapplication220180128090927.azurewebsites.n…	True

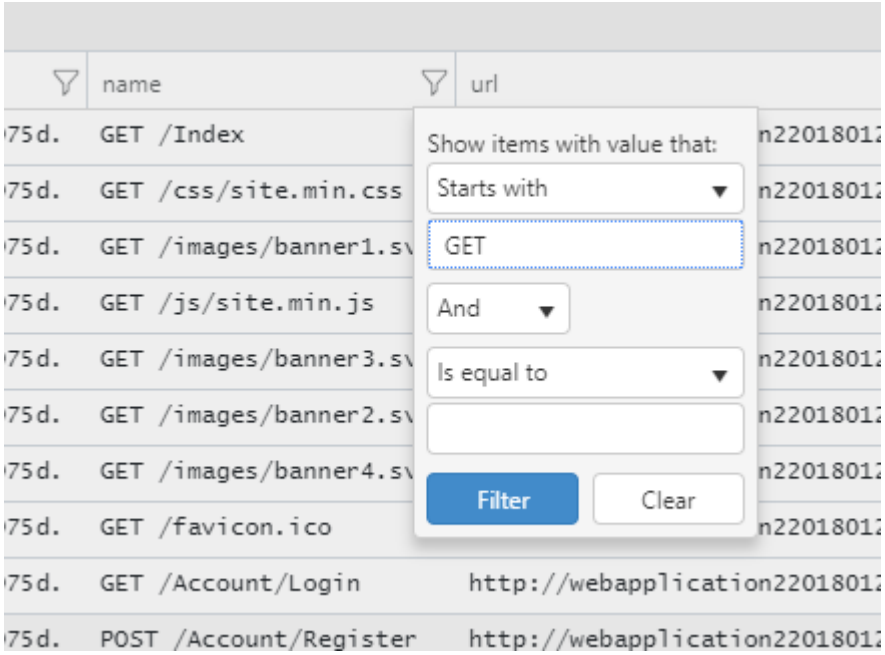
Result Tables

- In each table, data is organized in columns of different types, as indicated by the icons. For example, the *requests*table (expanded in the screenshot) contains the column *timestamp* which is a date-time, *URL* as a text column, and *duration* as a number.

- How many results did you get?
- The Analytics portal automatically scopes results by:
  - Time range - by default, queries are limited to the last 24 hours.
  - Number of results - results are limited to maximum of 10,000 records.
- This query is very general, and it returns too many results to make sense of. We can filter the results either through the table elements, or by explicitly adding a filter to the query.
- Note that while filtering results through the table elements applies to the existing result set, adding a filter to the query itself will return a new, filtered, result set and could therefore produce more accurate results.

Filter Results

- Let's focus on requests that execute GET. When reviewing the results, we can identify the relevant information is in the *name* column.
- Click the *Filter* icon next to the column title, and in the pop-up window select values that *Starts with* the text GET.
- Inspection of the result will show that we retained only queries with GET at the start of the name.



Examine Each Record

- Select the down arrow next to request field in a specific record.
- You will see the full content of that record

Drag a column header and drop it here to group by that column

$table	timestamp [UTC]	id	name	url
requests	2018-01-30T22:24:05.572	|de52559b-4d8051d5b78b975d.	GET /Index	http://webapplication2201
requests	2018-01-30T22:24:07.038	|de52559c-4d8051d5b78b975d.	GET /css/site.min.css	http://webapplication2201

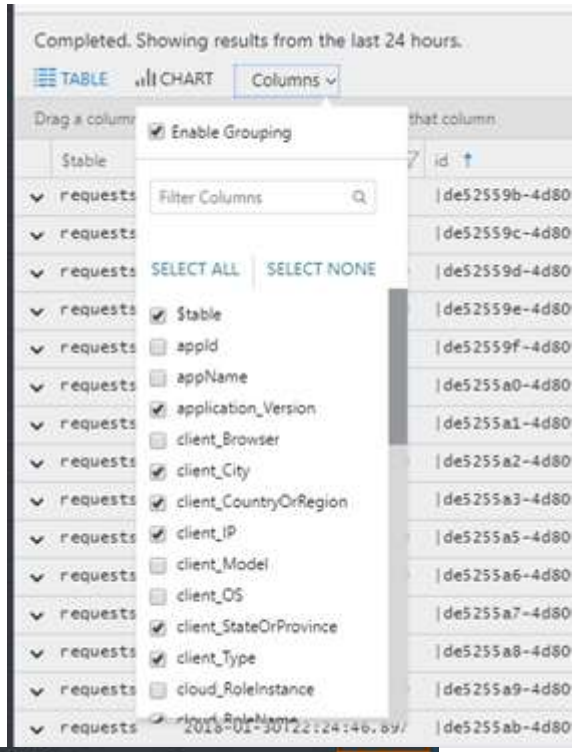
$table	requests
timestamp [UTC]	2018-01-30T22:24:07.038Z
id	|de52559c-4d8051d5b78b975d.
name	GET /css/site.min.css
url	http://webapplication220180128090927.azurewebsites.net/css/site.min.css?v=kHvJwvVAK
success	True
resultCode	200
duration	28.4037
performanceBucket	<250ms
customDimensions	{"AspNetCoreEnvironment":"Production","httpMethod":"GET"}
operation_Name	GET /css/site.min.css
operation_Id	de52559c-4d8051d5b78b975d
operation_ParentId	de52559c-4d8051d5b78b975d

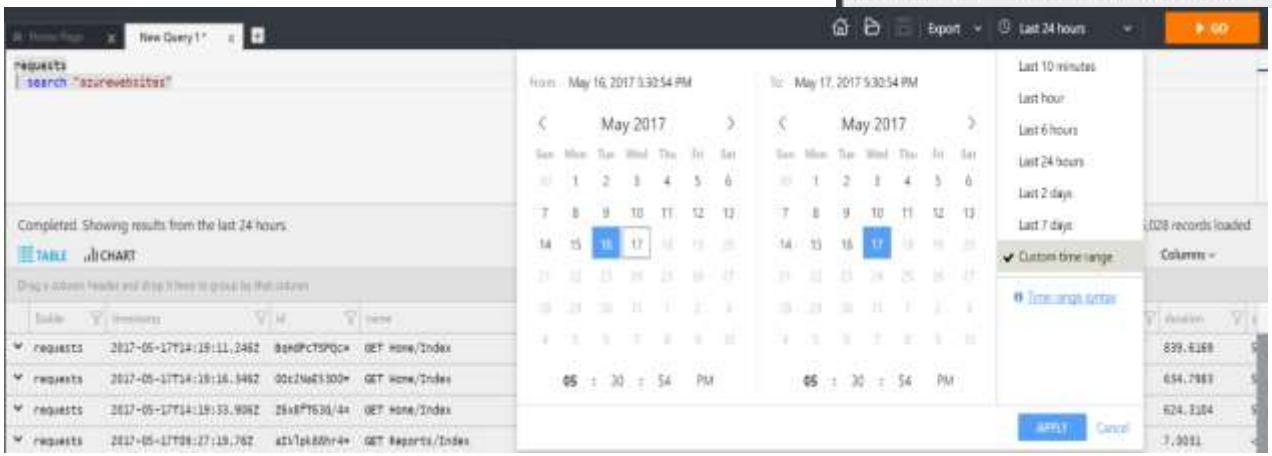
Sort and group results

- Now we have narrowed down the results to include only failed GET requests, from the last 24 hours. However, the results are not sorted in any way. To sort the results by a specific column - such as *timestamp* - click the column title. One click sorts in ascending order, two - descending.
- One of the common ways to organize results is by groups.
- To group results by a specific column, simply drag the column header above the other columns. To create subgroups - drag other columns the upper bar as well.

Select Columns to Show, Select Time Range

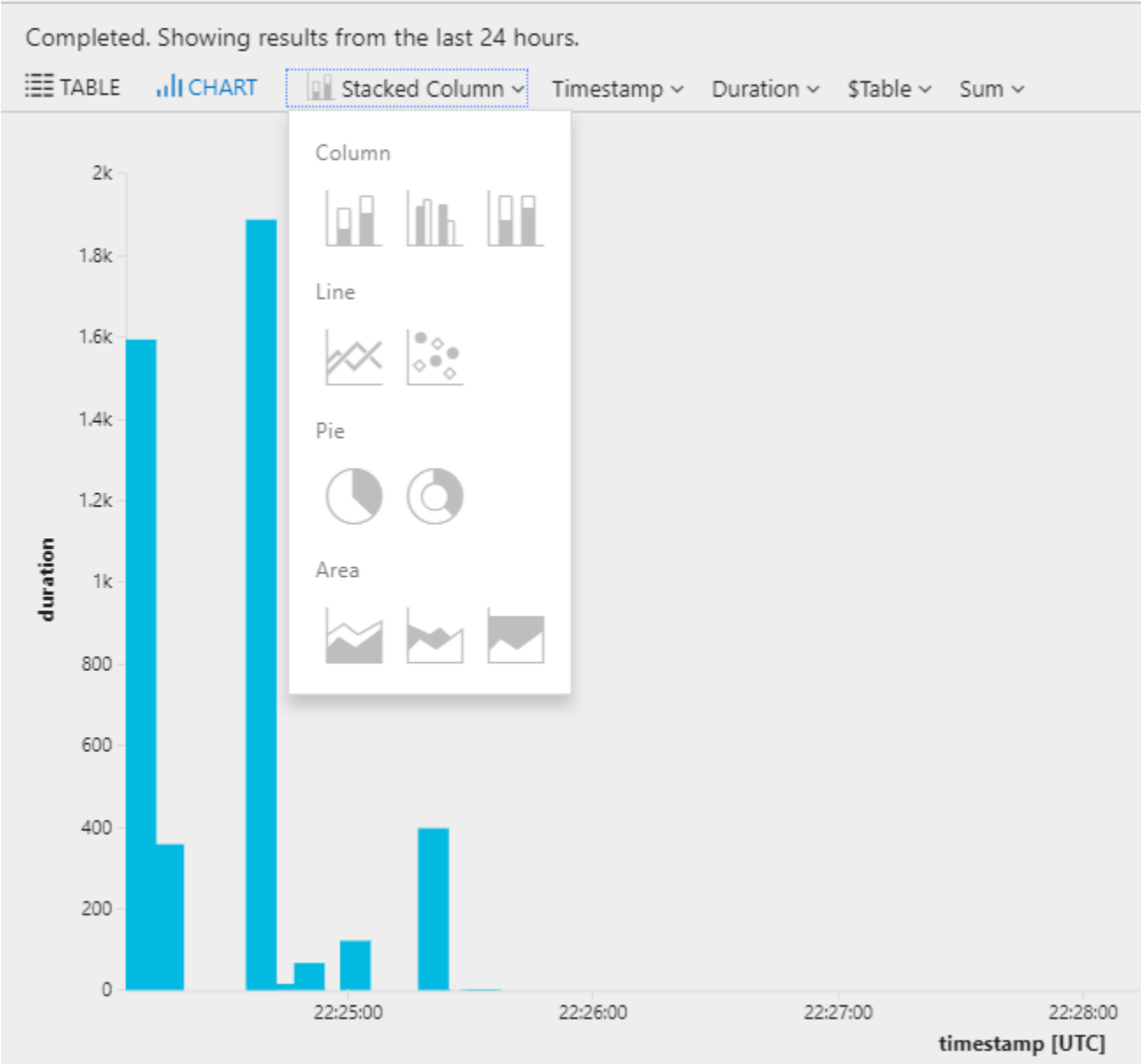
- The results table often includes a lot of columns. You might find that some of the returned columns are not displayed by default, or you may want to remove some the columns that are displayed.
- To select the columns to show, click the *Columns* button.
- By default, the Analytics portal applies the "last 24 hours" time range. To use a different range, simply select another value through the time picker, and click "Go".





Presenting Results as Charts

- Query results can be presented in different ways, depending on your preferences.

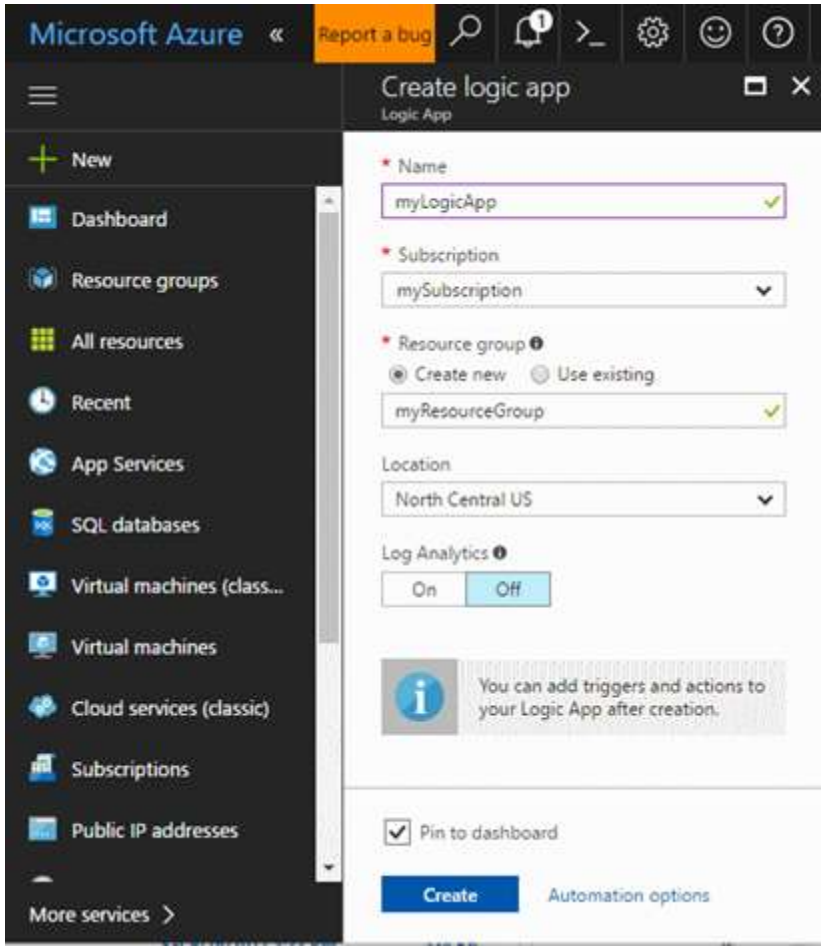


Send Notification on a Metric Value

- Azure Monitor makes metrics available for many Azure resources. These metrics convey the performance and health of those resources. In many cases metric values can point to something being wrong with a resource.
- You can create metric alerts to monitor for abnormal behavior and be notified if it occurs.

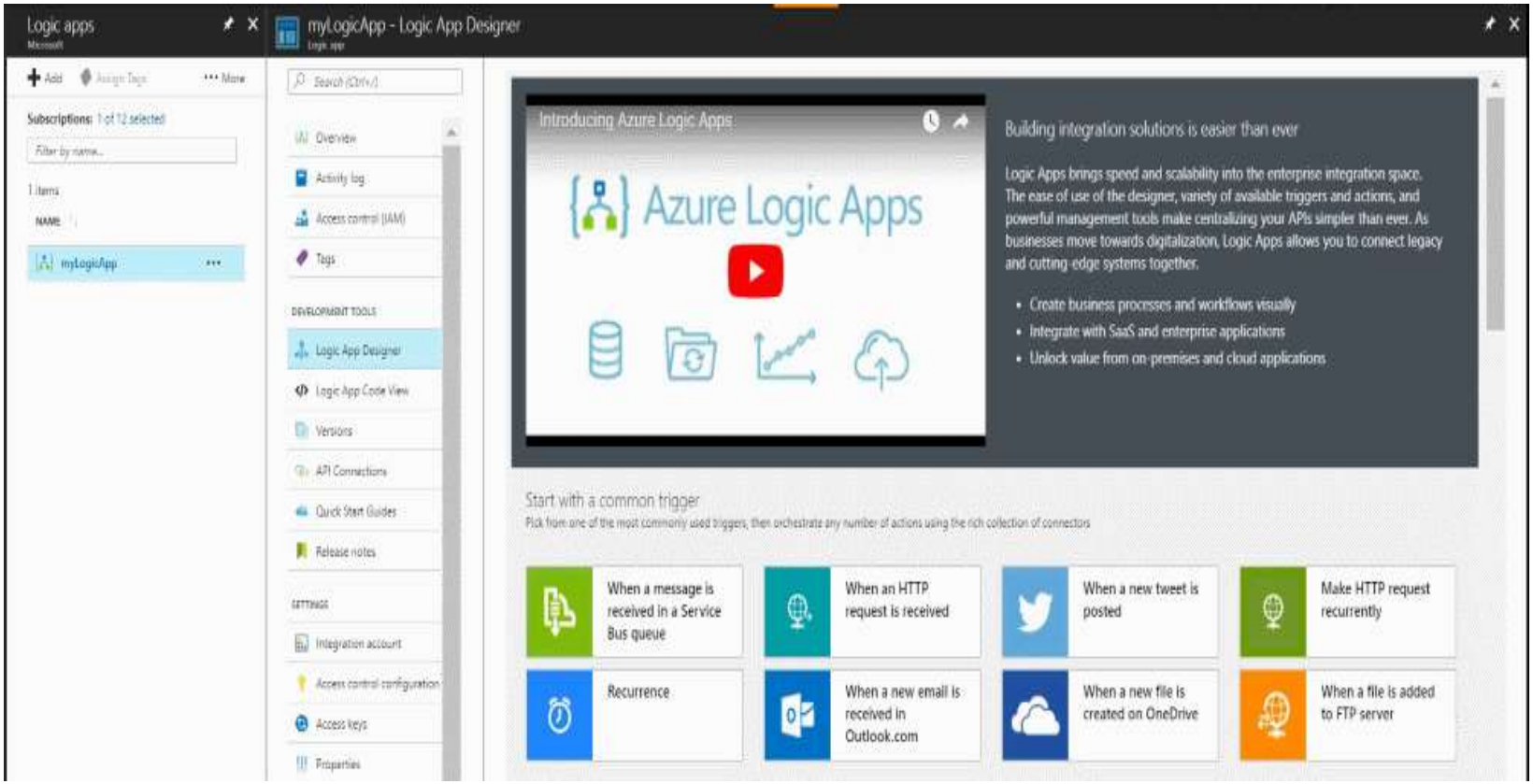
Create a Logic App

- Click the **New** button found on the upper left-hand corner of the Azure portal.
- Search for and select **Logic App**. Create a new resource group named **myResourceGroup** Use the default location. Click the **Create** button.
- Enter the logic app information and check the **Pin to Dashboard** option. When complete, click **Create**.
- The logic app should be pinned to your dashboard. Navigate to the logic app by clicking on it.



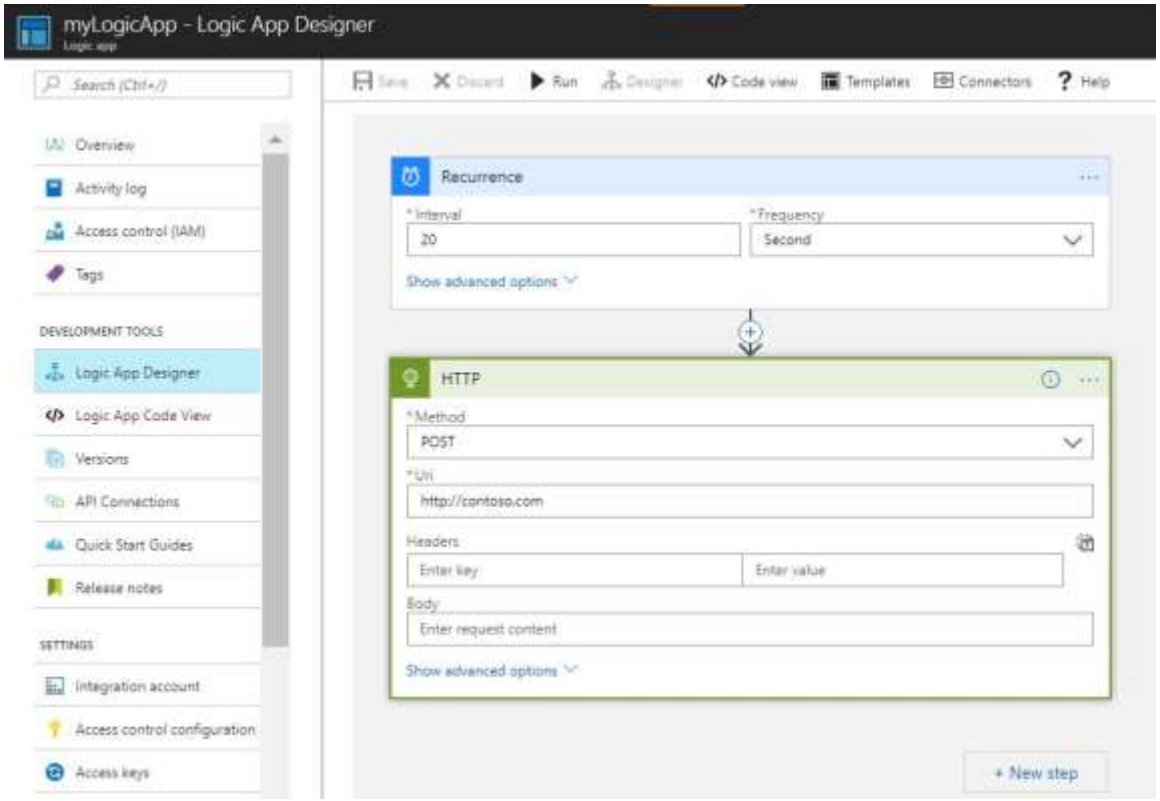
Logic App Designer

- In the Logic App panel, select the **Logic App Designer**



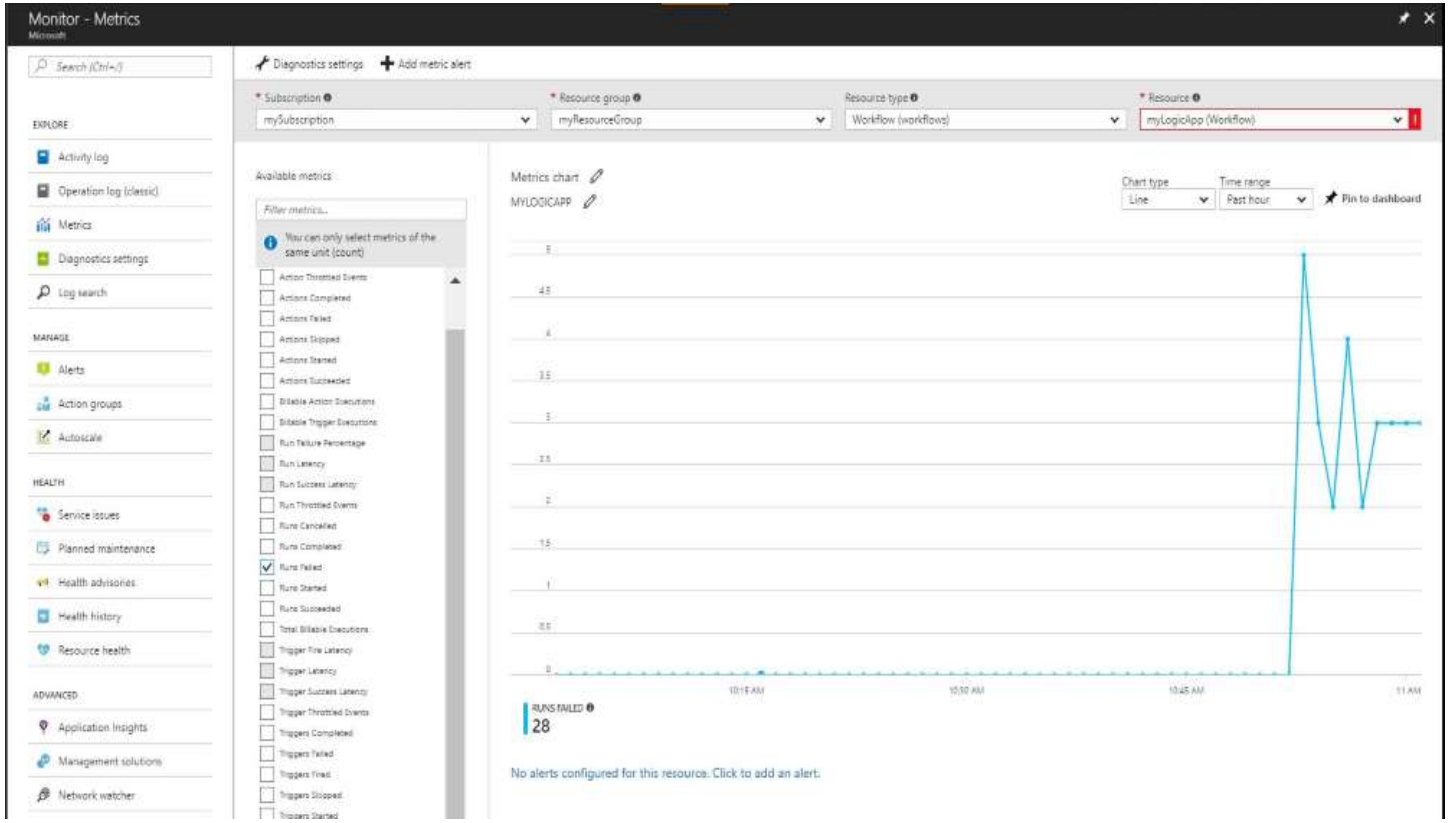
Logic App Setup

- Set up you values as seen in the following diagram.
- In the designer, select the **Recurrence** trigger.
- Set an interval of 20 and a frequency of second to ensure your logic app is triggered every 20 seconds.
- Click the **New Step** button, and select **Add an action**.
- Choose the **HTTP** option, and select **HTTP-HTTP**.
- Set the **Method** as POST and the **Uri** to a web address of your choice.
- Click **Save**.



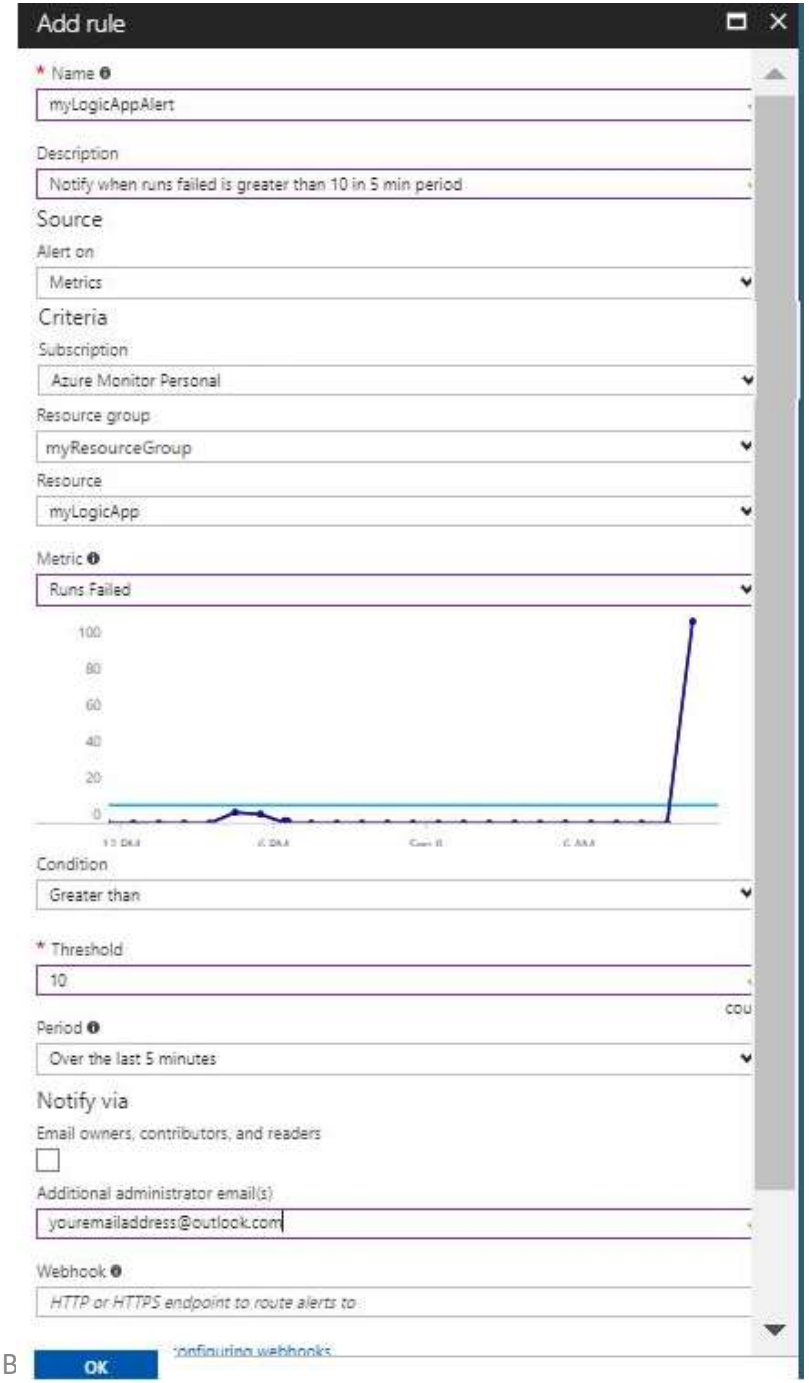
View Metrics for the Logic App

- Click the **Monitor** option in the left-hand navigation pane.
- Select the **Metrics** tab, fill in the **Subscription**, **Resource Group**, **Resource Type** and **Resource**information for your logic app.
- From the list of metrics, choose **Runs Started**.
- Modify the **Time range** of the chart to display data for the past hour.
- You should see a chart with the total number of runs your app has started over the past hour.



Metric alert

- In the top right portion of the metrics panel click the **Add metric alert** button.
- Name your metric alert 'myLogicAppAlert', and provide a brief description for the alert.
- Set the **Condition** for the metric alert as 'Greater than', set the **Threshold** as '10', and set the **Period**as 'Over the last 5 minutes'.
- Finally, under **Additional administrator email(s)** enter your email address. This alert ensures that you receive an email in the event your logic app has more than 10 failed runs within a period of 5 minutes.



Receive Metric Alert

- Within a few moments, you should receive an email from 'Microsoft Azure Alerts' to inform you the alert has been 'activated'.
- Navigate back to your logic app and modify the recurrence trigger to an interval of 1 and frequency of hour.
- Within a few minutes, you should receive an email from 'Microsoft Azure Alerts' informing you the alert has been 'resolved'.