**Agenda:** **Azure Monitoring and Log Analytics Workspace**

* Azure Monitor Overview
* Monitoring Metrics
* Monitoring Platform Logs
  + Resource Logs
  + Azure Activity Logs
* Managing Alerts
* Azure Log Analytic Workspace
* Collect data from an Azure VM
* Configure Azure SQL Database Auditing
* Azure Advisor
* Application Insight

### Azure Monitor Overview

* Monitoring is the act of **collecting and analyzing data** to determine the **performance**, **health**, and **availability** of your business application and the resources that it depends on.
* An effective monitoring helps you **increase** **your** **uptime** by **proactively** notifying you of critical issues so that you can resolve them before they become problems.
* **Azure Monitor** enables core monitoring for Azure services by allowing the collection of **metrics, activity logs, and diagnostic logs**. For example, the activity log tells you when new resources are created or modified.

### Key Capabilities of Azure Monitor

### 

### The following diagram gives a high-level view of Azure Monitor:

All data collected by Azure Monitor fits into one of two fundamental types:

### Metrics are numerical values that describe some aspect of a system at a particular point in time. They are lightweight and capable of supporting near real-time scenarios.

### Logs contain different kinds of data organized into records with different sets of properties for each type. Telemetry such as events and traces are stored as logs in addition to performance data so that it can all be combined for analysis.

### A screenshot of a computer Description automatically generated

### 

## **What Data does Azure Monitor collect?**

Azure Monitor can collect data from a **variety of sources**. You can think of monitoring data for your applications in tiers ranging from your application, any operating system and services it relies on, down to the platform itself.

Azure Monitor collects data from each of the following tiers:

* **Application monitoring data** (User Telemetry / Application Logs) - Data about the performance and functionality of the code you have written, regardless of its platform.
* **Guest OS monitoring data** (Linux Syslog and Windows Perf Counters) - Data about the operating system on which your application is running. This could be running in Azure, another cloud, or on-premises. Agents will be installed in the guest OS.
* **Azure resource monitoring data**: Data about the operation of an Azure resource including Metrics and Resource Logs.
* **Azure subscription monitoring data**: Data about the operation and management of an Azure subscription, as well as data about the health and operation of Azure itself.
* **Azure tenant monitoring data**: Data about the operation of tenant-level Azure services, such as Azure Active Directory.

### 

The following table lists the different ways that you can use metric and logs data in Azure Monitor.

|  |  |
| --- | --- |
| **Analyze** | **Use metrics explorer** to analyze collected metrics on a chart and compare metrics from different resources. |
| **Visualize** | Pin a chart from metrics explorer to an **Azure dashboard**. |
| **Alert** | Sends a **notification** or takes **automated action** when the metric value crosses a threshold. |
| **Automate** | To **increase or decrease** resources based on a metric value crossing a threshold. |
| **Export** | **Route Metrics to Logs** to analyze data in Azure Monitor Metrics together with data in Azure Monitor Logs and to store metric values for **longer than 93 days**. **Stream Metrics** to an Event Hub to route them to external systems. |
| **Retrieve** | Access metric values from a command line using **PowerShell cmdlets or CLI** Access metric values from custom application using **REST API**. |
| **Archive** | Archive the **performance or health history** of your resource for compliance, auditing, or offline reporting purposes. |

### Costs associated with monitoring

There **is no cost** for analyzing monitoring data that is collected by default. This includes the following:

* Collecting **platform metrics** and analyzing them with metrics explorer.
* Collecting **Activity log** and analyzing it in the Azure portal.
* Creating an **Activity log alert rule**.

There are no Azure Monitor costs for collecting and exporting logs and metrics, but there may be related costs associated with the destination:

* Costs associated with data ingestion and retention when collecting logs and metrics in **Log Analytics workspace**.
* Costs associated with data storage when collecting logs and metrics to an **Azure storage account**.
* Costs associated with event hub streaming when forwarding logs and metrics to **Azure Event Hubs**.

There may be Azure Monitor costs associated with the following:

* Accessing metrics through API.
* Running a log query.
* Creating a metric or log query alert rule.
* Sending a notification from any alert rule.

### <https://azure.microsoft.com/en-in/pricing/details/monitor/>

Collecting Logs and Metrics

Diagram

Description automatically generated

**Monitoring – Metrics**

* **Metrics** provide **performance statistics** for different resources. They are lightweight and capable of supporting **near real-time** scenarios.
* Metrics are **numerical values** that describe some aspect of a system at **a particular time**.
* Platform metrics are **collected automatically** into Azure Monitor Metrics with no configuration required. Create a **diagnostic setting** to send entries to Azure Monitor Logs or to forward them outside of Azure.
* Metrics are **collected at regular intervals** and are useful for **alerting**.
* You can **access metrics** from the Azure Portal, Monitor APIs (REST, and .Net) and analysis solutions such as the **Log Analytics** and **Event Hubs**.

Data collected by Azure Monitor Metrics is stored in a **time-series database** which is optimized for analyzing time-stamped data. Each set of metric values is a time series with the following properties:

* The time the value was collected
* The resource the value is associated with
* A namespace that acts like a category for the metric
* A metric name
* The value itself

Graphical user interface, text, application, email

Description automatically generated

Use **Metrics Explorer** to interactively analyze the data in your metric database and chart the values of multiple metrics over time. You can pin the charts to a dashboard to view them with other visualizations. You can also retrieve metrics by using the **Azure monitoring REST API**.

## **Retention of Metrics**

For most resources in Azure, metrics are stored for **93 days**. There are some exceptions:

**Guest OS metrics:** These are performance counters collected by agent and routed to an Azure storage account.

* **Classic guest OS metrics**: Retention for these metrics is 14 days.
* **Guest OS metrics sent to Azure Monitor Metrics**: Retention for these metrics is 93 days.
* **Guest OS metrics collected by Log Analytics agent**. Retention for these metrics is 31 days, and can be extended up to 2 years.

**Demo: Monitor Platform Logs**

**Option1:** Locate the resource 🡪 **Overview** 🡪 Note the graphs which are provided by Azure Monitor

**Option2:** Locate the resource 🡪 Monitoring Section 🡪 **Metrics** 🡪 Opens Metric Explorer 🡪 Click Add metric

**Option3: Using Monitor Service**

1. Azure Portal 🡪 All Services 🡪 **Monitor 🡪 Metrics**
2. Add metric 🡪 Resource = <Storage Account> . . .

**Monitoring Resource Logs**

* Azure resource logs are platform logs that provide **insight into operations** that were performed within an Azure resource (the data plane)
* Resource logs are **automatically generated** by Azure resources but **not collected** without a diagnostic setting. Create **a diagnostic setting** to send entries to Azure Monitor Logs or to forward them outside of Azure.
* **Example**: Getting a **secret from a Key Vault** or making a **request to a database** or **Read/Write/Delete operations of a Storage Account**.
* The content of resource logs varies by the Azure service and resource type.
* You must create a **diagnostic setting** for each Azure resource to send its resource logs to a **Log Analytics workspace** to use with Azure Monitor Logs, **Azure Event Hubs** to forward outside of Azure, or to **Azure Storage** for archiving.
  + Event Hub and Azure Storage must be in **same region** as the resource. Log Analytics Workspace can be **any region**.
  + Destination service can be in any Subscription.
  + For Azure Storage destination we can specify the **retention period** for each category of log (Retention can be between 1 to 365 days. 0 is infinite)

The tables used by resource logs depend on what type of collection the resource is using:

* **Azure diagnostics** - All data written is to the **AzureDiagnostics** table. It's in NoSQL Format
* **Resource-specific** - Data is written to individual table for each category of the resource.

Note: All Azure services will eventually migrate to the Resource-Specific mode.

Example: Storage Data is stored in these tables.

|  |  |
| --- | --- |
| **Table** | **Description** |
| StorageBlobLogs | Logs that describe activity in blob storage. |
| StorageFileLogs | Logs that describe activity in file shares. |
| StorageQueueLogs | Logs that describe activity in queues. |
| StorageTableLogs | Logs that describe activity in tables. |

**Azure Activity Logs (at subscription level)**

* Azure Activity Log is a subscription log that provides insight into subscription-level events that have occurred in Azure
* The Activity log is collected automatically with no configuration required and can be view in the Azure portal. Create a diagnostic setting to copy them to Azure Monitor Logs or to forward them outside of Azure.
* The Azure Activity Log is a **subscription log** that provides insight into **subscription-level events** that have occurred in Azure.
* For additional functionality, you should create a **diagnostic setting** to send the Activity log to Log Analytic workspace, to Azure Event Hubs to forward outside of Azure, or to Azure Storage for archiving.

**Through activity logs, you can determine:**

* **What** operations were taken on the resources in your subscription.
* **Who** started the operation.
* **When** the operation occurred.
* The **status** of the operation.
* The **values** of other properties that might help you research the operation.

**Examples of Activity Log:**

1. Create or update Virtual Machine.
2. Change the Size of VM.
3. Start of the VM.
4. Create or Update SQL Database.
5. Change the configuration of SQL Database.

Note: For some events example configuration/size changes, you can view the **Change history**, which shows what changes happened during that event time.

## **Query the Activity Log Options:**

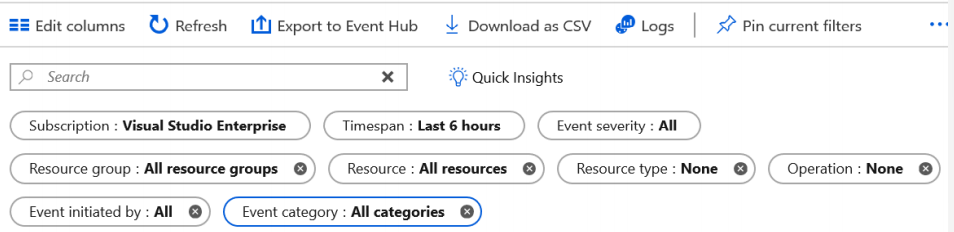
1. View the Activity Log for a particular resource from the **Activity Log** option in that resource's menu in **Azure Portal**.
2. View the **Activity Log** for all resources from the **Monitor** menu in the Azure portal.
3. You can also retrieve Activity Log records with **PowerShell, CLI, or REST API**

## **Activity Log retention**

Once created, Activity Log entries are not modified or deleted by the system. Also, you can't change them in the interface or programmatically. Activity Log events are stored for **90 days**. To store this data for longer periods, collect it in Azure Monitor or export it to **Storage Account, Log Analytics or Event Hubs**.

**Activity Log Filters:**

In the Azure portal, you can filter your Activity Log by these fields:



**Event Categories**

Graphical user interface, text, application, email

Description automatically generated

Configuring logs from the Activity Log blade allows you to collect the following specific log types:

* **Administrative**: Creating, updating, deleting, and action events. For example, creating a storage account or starting a virtual machine.
* **Security**: Security center alerts, such as suspicious double extension files executed.
* **Service Health**: Region-wide health events
* **Alert**: Alerts that you can define, for example, if you create an alert to monitor for high CPU usage.
* **Recommendation**: Azure Advisor recommendations,
* **Policy**: Events triggered by policies.
* **Autoscale**: VM Scale sets or Apps scaling out or in.
* **Resource** **Health**: Resource-specific health issues.

**About Alerts**

Alerts can be authorized in a consistent manner regardless of the monitoring service or signal type. All alerts fired and related details are available in single page.

Authoring an alert is a **three-step task** where the user first picks a target for the alert, followed by selecting the right signal and then specifying the logic to be applied on the signal as part of the alert rule.

**Managing Alerts**

You can alert on metrics and logs. These include but are not limited to:

* Metric values
* Log search queries
* Activity Log events
* Health of the underlying Azure platform
* Tests for web site availability

A picture containing text, sign, screenshot

Description automatically generated

### Action Groups

* Action groups enable you to configure a **list of actions** to take when the alert is triggered.
  + Automatic runbook
  + Azure Function
  + Email Azure Resource Manager role
  + Email/SMS/Push/Voice
  + Logic App
  + Webhook
  + ITSM - Connect Azure and a supported IT Service Management (ITSM) product/service. This requires an ITSM Connection.
* Action groups ensure that the same actions are taken each time an alert is triggered.
* There are several action types you can select when defining the group: Select **Email/**[**SMS**](https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-sms-alert-behavior)**/Push/Voice, Azure Function,** [**Logic App**](https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-action-groups-logic-app)**,** [**Webhook**](https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-activity-log-alerts-webhook)**,** [**IT Service Management**](https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-itsmc-overview)**, or Automation Runbook.**
* **Every alert can use one or more action groups.**
* New Action Group is required only if the set of Actions in a existing action group is different for a New Alert.
* You may configure up to 2,000 action groups in a subscription.

**Demonstration: Create Alert Rule**

Select Any Service 🡪 Monitoring Section 🡪 Alerts 🡪 **New alert rule**

OR

When you are monitoring metrics, you can choose the menu "**New alert rule**"

1. **Select Resource:** For example, Storage account.
2. **Add Condition**

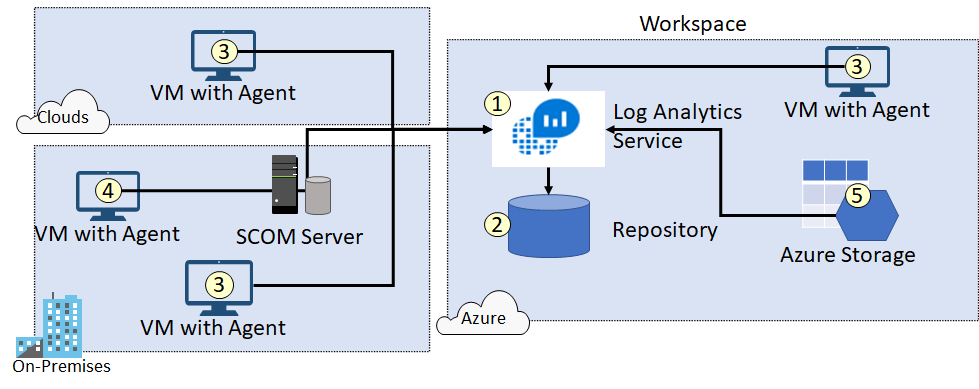
* Select signal. For example, **Used Capacity**.
* Configure signal logic. For example, **over a six-hour period whenever the Used Capacity is over 1000000 bytes**.
  + Operator = Greater Than
  + Aggregation Type = Average
  + Threshold Value = 1000000
  + Aggregation Granularity = 6 hours
  + Frequency of Evaluation = Every 5 minutes

1. **Define Action Group**. Create an action group to notify your team via **email** and **text** messages or automate actions using **webhooks** and **runbooks**.
2. **Define Alert Details:** Alert rule name, description.

**Azure Log Analytics Workspace**

* **Log Analytics** helps you collect, correlate, search, and act on log and performance data generated by **operating systems and applications**.
* Log Analytics gives you a single interface for consuming and correlating the data, covering both Linux and Windows Server.
* It gives you real-time operational insights using integrated search and custom dashboards to readily **analyze millions of records** across all your workloads and servers regardless of their physical location.

**Connected Sources: Connected sources are resources that generate data collected by Log Analytics**

****

Ensure you can locate each of the following.

* The Log Analytics service (1) collects data and stores it in the repository (2). The repository is hosted in Azure.
* Computer agents (3) generate data to the Log Analytics service. These agents can run on Windows or Linux computers, virtual or physical computers, on-premises or cloud computers, and Azure or other cloud providers.
* A System Center Operations Manager (SCOM) management group can be connected to Log Analytics. SCOM agents (4) communicate with management servers which forward events and performance data to Log Analytics.
* An Azure storage account (5) can also collect Azure Diagnostics data from virtual machine in Azure.

**Data Sources:** Data sources are the different kinds of data collected from each connected source.

1. Windows Event Logs
2. Windows Performance Counters
3. Linux Syslog (Performance Counters)
4. IIS Logs (Windows)
5. Custom Fields
6. Custom Logs

Each data source has additional configuration options. For example, the Windows Event Log can be configured to forward Error, Warning, or Informational messages.

**Demonstration – Log Analytics - Enable, Collect and View Data from Azure VM**

**Create a Log Analytics Workspace**

1. All Services 🡪 Log Analytics workspaces 🡪 +Add
2. Provide a name = *DefaultLAWorkspace* 🡪 . . . 🡪 OK

**Data Collection Rule**

1. **Search Data Collection Rule 🡪 + Create**
   1. **Resources 🡪 +VM**
   2. **Data Sources and LA Workspaces.**

Note: This will install AzureMonitorWindowsAgent as VM extension in all selected VMs

**~~Connect VM to Log Analytics~~**

1. **~~Under Log Analytic Workspace 🡪 Data Sources Section 🡪 Virtual Machine 🡪 Connect~~**

~~The agent (~~**~~VM Extension~~**~~:~~ **~~MicrosoftMonitoringAgent~~**~~) is automatically installed and configured for your Log Analytics workspace. This process takes a few minutes, during which time the~~ **~~Status~~** ~~is~~ **~~Connecting~~**~~.~~

~~After you install and connect the agent, the~~ **~~Log Analytics connection status~~** ~~will be updated with~~ **~~This workspace~~**~~.~~

~~Alternatively, agents can be installed via a PowerShell or Azure CLI script or as part of an ARM template when deploying VMs, which is ideal when you want to automate your deployments. The following is an example ARM template snippet we can use:~~

~~{~~

~~"type": "extensions",~~

~~"name": "~~**~~OMSExtension~~**~~",~~

~~"apiVersion": "[variables('apiVersion')]",~~

~~"location": "[resourceGroup().location]",~~

~~"dependsOn": [~~

~~"[concat('Microsoft.Compute/virtualMachines/', variables('vmName'))]"~~

~~],~~

~~"properties": {~~

~~"publisher": "Microsoft.EnterpriseCloud.Monitoring",~~

~~"type": "MicrosoftMonitoringAgent",~~

~~"typeHandlerVersion": "1.0",~~

~~"autoUpgradeMinorVersion": true,~~

~~"settings": {~~

~~"workspaceId": "xxxxxxxxx"~~

~~},~~

~~"protectedSettings": {~~

~~"workspaceKey": "xxxxxxxxxx"~~

~~}~~

~~}~~

~~}~~

~~Because the configuration can be written as JSON using an ARM template, you can also create an~~ **~~Azure Policy~~** ~~with a deployIfNotExists setting to automatically configure agents whenever a VM is created.~~

**~~To get workspaceId and workspaceKey: Log Analytics 🡪 Agent Management~~**

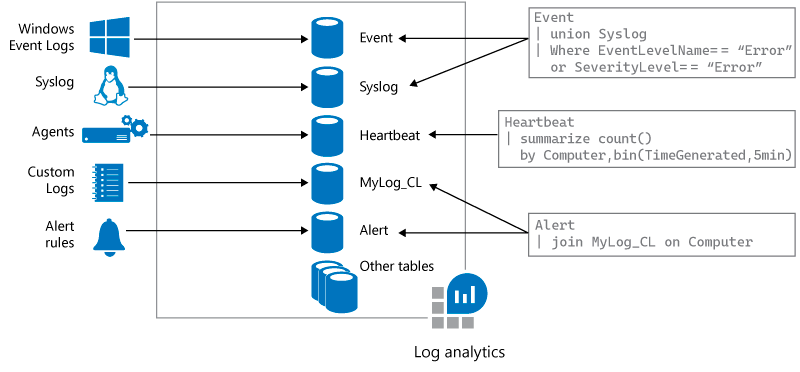
**To Ingest Data into Log Analytic Workspace:**

**Data Collection Rules with DataSources along with their destinations.**

[**https://learn.microsoft.com/en-us/azure/azure-monitor/vm/tutorial-monitor-vm-alert-recommended**](https://learn.microsoft.com/en-us/azure/azure-monitor/vm/tutorial-monitor-vm-alert-recommended) **(and related topics)**

**Query Language Syntax:**

Each data source and solution stores its data in dedicated tables in the Log Analytics workspace. Documentation for each data source and solution includes the name of the data type that it creates and a description of each of its properties.

****

**View data collected in Log Analytics**

Now that you have enabled data collection, lets run a simple log search example to see some data from the target VMs.

1. Select **Log Analytic Workspace** 🡪 **Logs**
2. Run Queries
   1. **Perf**
   2. **Event** | search "error" (This query searches the Event table for records that contain the term error in any property.)

Note: The pipe (|) character separates commands, so the output of the first command is the input of the next command.

**Add a filter to the query:**

1. There is an arrow to the left of each record. Click this arrow to open the details for a specific record.
2. Hover above a column name for the "+" and "-" icons to display. To add a filter that will return only records with the same value, click the "+" sign. Click "-" to exclude records with this value and then click **Run** to run the query again.

Event

| search "error"

| where EventLevel == 1

### Use the Time range control

To use the **Time range** control, select it in the top bar, and then select a value from the dropdown list

If the query explicitly sets a filter for **TimeGenerated**, the time picker control shows **Set in query**, and is disabled to prevent a conflict.

Event

| search "error"

| where EventLevel == 1

| where TimeGenerated > ago(3d)

## **View and modify charts**

* By default, results appear in a table. Select **Chart** above the table to see the results in a graphic view.
* The results appear in a stacked bar chart. Select other options like **Stacked Column** or **Pie** to show other views of the results.

**You can Save, load and export queries:**

* To load a saved query, select **Query explorer** at upper right.
* To export a query, select **Export** on the top bar, and then select **Export to CSV**

**Azure Storage Log Analytics queries in Azure Monitor**

1. To list all requests with anonymous access over the last three days.

StorageBlobLogs

| where TimeGenerated > ago(3d) and AuthenticationType == "Anonymous"

| project TimeGenerated, OperationName, AuthenticationType, Uri

1. To list the 10 most common errors over the last three days.

StorageBlobLogs

| where TimeGenerated > ago(3d) and StatusText !contains "Success"

| summarize count() by StatusText

| top 10 by count\_ desc

1. To list the top 10 operations that caused the most errors over the last three days.

StorageBlobLogs

| where TimeGenerated > ago(3d) and StatusText !contains "Success"

| summarize count() by OperationName

| top 10 by count\_ desc

**Single vs Multiple Log Analytics Workspace:**

A large multi-national organization may have multiple subscriptions either across regions or departments. We must then consider how systems are managed and monitored.

* For example, if there is a **single team** responsible for everything across the enterprise you would be better with a **single workspace**, with all resources, regardless of which subscription they are in, send the logs to that workspace. This can have **implications on network traffic** if those services are in different regions as you will incur **additional ingress and egress costs**.
* Alternatively, you would have **workspaces per region**, or **even per subscription**, this helps provide more **granular control**, however your data is now spread across multiple workspaces.
* **A hybrid approach could include sending some logs to a central workspace, and other logs to individual workspaces**. This design pattern is useful whereby different teams needs access to different logs and have different responsibilities. For example, a central workspace maybe used by a company-wide monitoring team, but individual service owners need visibility of application specific information and metrics, as we can see in the following diagram:

Graphical user interface, application

Description automatically generated

**Note: Log Analytics Workspace can be shared across multiple regions and multiple subscriptions.**

**Application Insight Integration of VM**

1. **~~Go to VM~~** ~~🡪 Insights 🡪~~ **~~Enable~~** ~~Button 🡪 Select the Log Analytic workspace.~~ **~~DependencyAgentWindows~~** ~~Extension will be installed in VM.~~
2. Go to VM 🡪 Insights 🡪 Azure Monitor 🡪 You'll see your VM with any other VMs in your subscription that are onboarded~~.~~

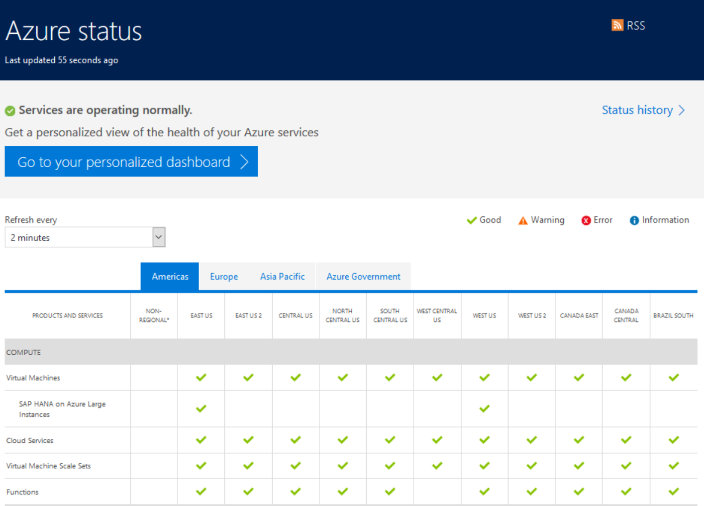
## **View data collected**

1. VM 🡪 Insights 🡪 **Performance** 🡪 This shows a select group of **performance counters** collected from the guest operating system of your VM.
2. VM 🡪 Insights 🡪 **Map** to open the maps feature which shows the processes running on the virtual machine and their dependencies.
   1. Select **Properties** to open the property pane if it isn't already open. Expand the processes for your virtual machine. Select one of the processes to view its details and to highlight its dependencies.
   2. Select your virtual machine again and then select **Log Events**. You see a list of tables that are stored in the Log Analytics workspace for the virtual machine. This list will be different depending whether you're using a Windows or Linux virtual machine. Click the **Event** table. This includes all events from the Windows event log. Log Analytics opens with a simple query to retrieve event log entries.

**Azure Service Health**

Service Health provides you with a customizable dashboard which tracks the health of your Azure services in the regions where you use them.

**Azure status provides a global view of the health of Azure services and regions**

****

**Service health events:**

1. **Service issues** - Problems in the Azure services that affect you right now.
2. **Planned maintenance** - Upcoming maintenance that can affect the availability of your services in the future.
3. **Health advisories** - Changes in Azure services that require your attention. Examples include deprecation of Azure features or upgrade requirements (e.g upgrade to a supported PHP framework).
4. **Security advisories** - Security related notifications that may affect the availability of your Azure services

**Service Health allows you to:**

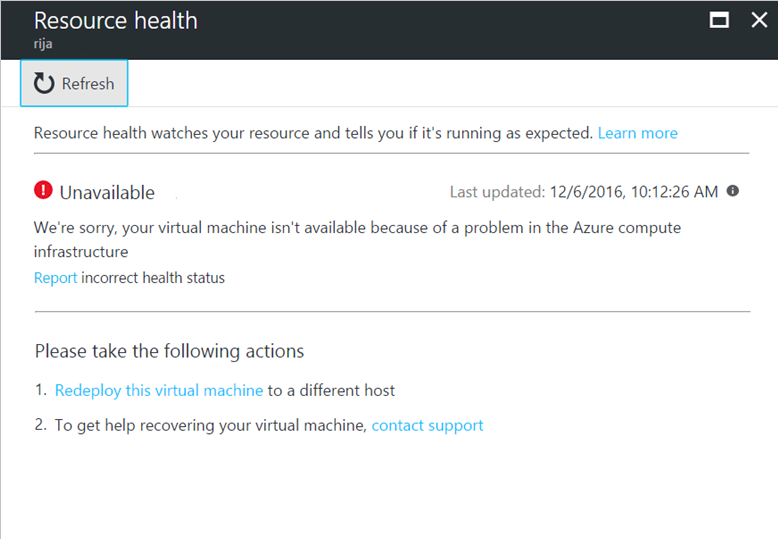
* See current issues which impact your services
* Get links and downloadable explanations
* Get support from Microsoft
* Pin a personalized health map to your dashboard
* Configure service health alerts

**Resource Health Overview**

Azure Resource Health helps you diagnose and get support for service problems that affect your Azure resources. It reports on the current and past health of your resources.

Resource definition and health assessment

* **Health status:** The health of a resource is displayed as one of the following statuses.
  + Available
  + Unavailable (includes platform events and non-platform events)
  + Unknown
  + Degraded
* **Reporting an incorrect status**: If you think that the current health status is incorrect, you can tell us by selecting Report incorrect health status
* **History information**: You can access up to 30 days of history in the Health history section of Resource Health.



**Configure Azure SQL Database Auditing**

Azure SQL Database Auditing tracks database events and writes them to an audit log in your Azure Storage account.

Auditing can help you maintain regulatory compliance, understand database activity, and gain insight into discrepancies and anomalies that could indicate business concerns or suspected security violations.

**SQL Database Auditing allows you to:**

* **Retain** an audit trail of selected events. You can define categories of database actions to be audited.
* **Report** on database activity. You can use preconfigured reports and a dashboard to get started quickly with activity and event reporting.
* **Analyze** reports. You can find suspicious events, unusual activity, and trends.

1. **Create a Storage Account**
2. Navigate to **DemoDb (database**) 🡪 Security blade 🡪 **Auditing**
3. **Clear Inherit settings from server** check box and apply the following settings:
4. Auditing: **ON**
5. Auditing log destination: Check **Storage and Log Analytics**
6. **Save**

**Note: If Blob Auditing is enabled on the server, it will always apply to the database, regardless of the database settings.**

**To view audit logs**

1. Perform CRUD Operations
2. Select Database 🡪 **Auditing 🡪 View Audit logs.**
3. Then, you have two ways to view the logs:
   1. Clicking on **Log Analytics**
   2. Clicking **View dashboard** at the top of the **Audit records**

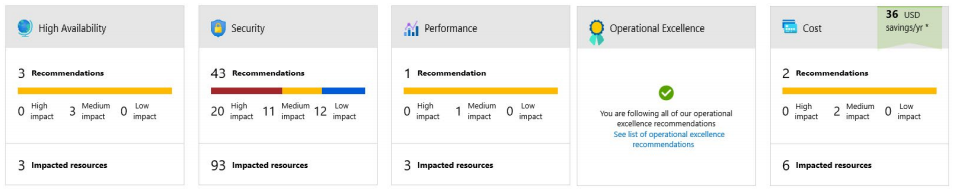
**More:** <https://docs.microsoft.com/en-us/azure/azure-sql/database/auditing-overview>

|  |
| --- |
| [Log Analytics Querying Demonstration](https://portal.loganalytics.io/demo) page with Dummy Data:  <https://portal.loganalytics.io/demo#/discover/query/main>  This page provides a live demonstration workspace where you can run and test queries. |

**Azure Advisor**

Advisor is a personalized cloud consultant that helps you follow best practices to optimize your Azure deployments. It analyzes your resource configuration and usage telemetry and then recommends solutions that can help you improve the **cost effectiveness, performance, high availability, and security** of your Azure resources.

The Advisor cost recommendations page helps you optimize and reduce your overall Azure spend by identifying idle and underutilized resources.



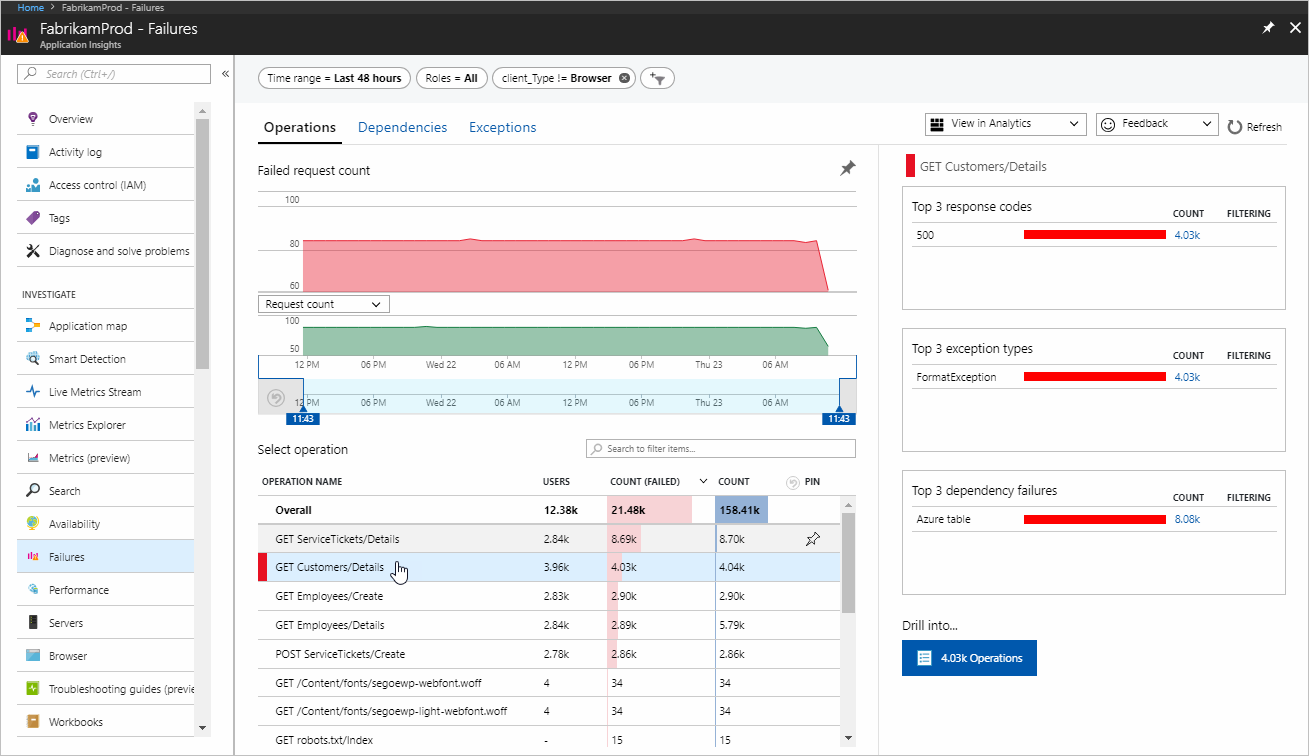
✔️ Advisor provides recommendations for virtual machines, availability sets, application gateways, App Services, SQL servers, and Redis Cache.

**Azure Application Insight**

* Application Insights, a feature of Azure Monitor, is an extensible **Application Performance Management (APM) service** for developers and DevOps professionals. Use it to monitor your **live applications**.
* It will automatically detect **performance anomalies**, and 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, Java, and Python hosted on-premises, hybrid, or any public cloud.
* It can monitor and analyze telemetry from mobile apps by integrating with Visual Studio App Center.

**What does Application Insights monitor?**

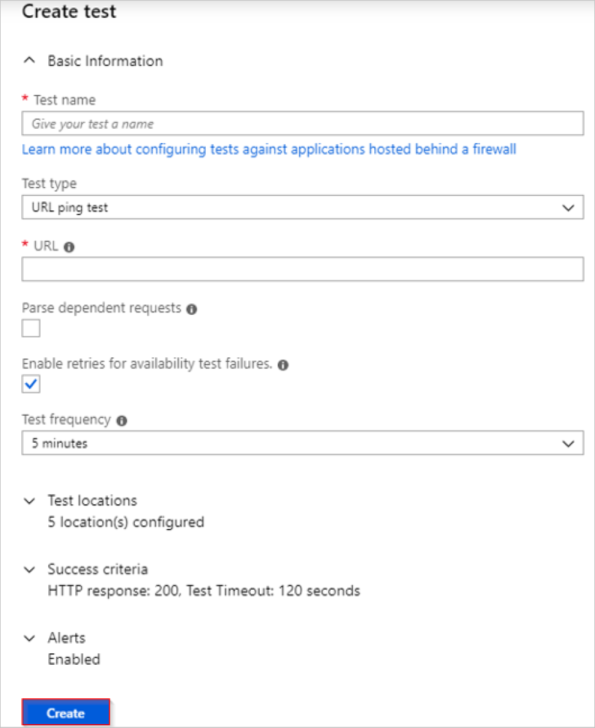
* **Request rates, response times, and failure rates** - Find out which pages are most popular, at what times of day, and where your users are. See which pages perform best. If your response times and failure rates go high when there are more requests, then perhaps you have a resourcing problem.
* **Dependency rates, response times, and failure rates** - Find out whether external services are slowing you down.
* **Exceptions** - Analyze 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** ingested 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.



**URL ping test:**

It uses more advanced HTTP request functionality to validate whether an endpoint is responding. It also measures the performance associated with that response, and adds the ability to set custom success criteria coupled with more advanced features like parsing dependent requests, and allowing for retries.

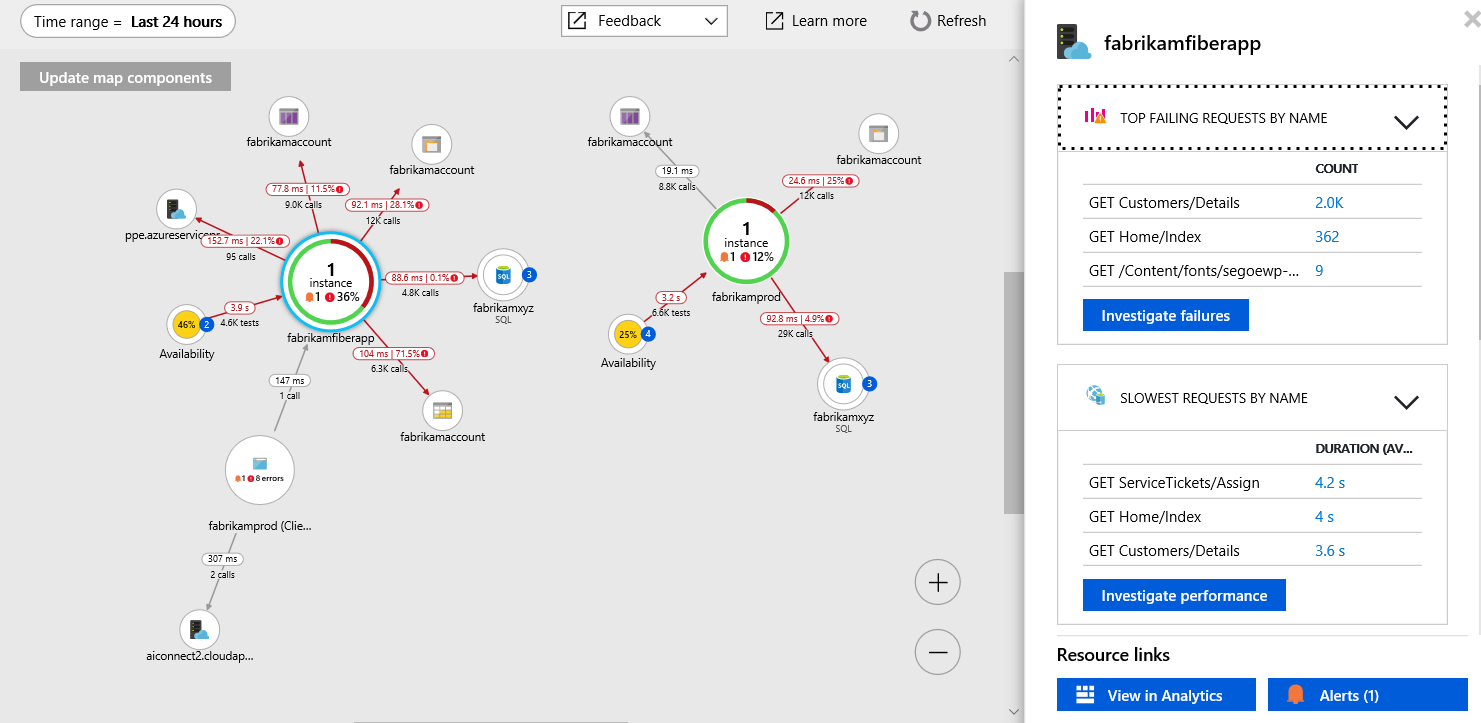
**AppInsight 🡪 Availability**

****

**Application Maps:**

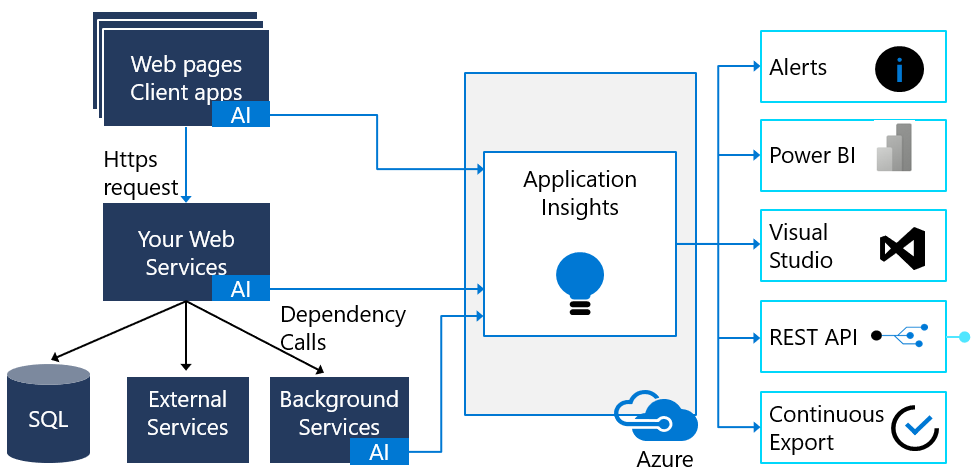
* The SDK also analyzes your application for a typology.
* You can analyze this typology by reviewing the Application map view

Application Map helps you **spot performance bottlenecks** or **failure hotspots** across all components of your distributed application. Each node on the map represents an application component or its dependencies; and has health KPI and alerts status. You can select through from any component to more detailed diagnostics, such as Application Insights events. If your app uses Azure services, you can also select through to Azure diagnostics, such as SQL Database Advisor recommendations.

****

**How does Application Insight work?**

* You install a small **instrumentation package** in your application, and set up an Application Insights resource in the Microsoft Azure portal. The instrumentation monitors your 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 webpages themselves.
* 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.
* All these telemetry streams are integrated in the Azure portal, where you can apply powerful analytic and search tools to the raw data.



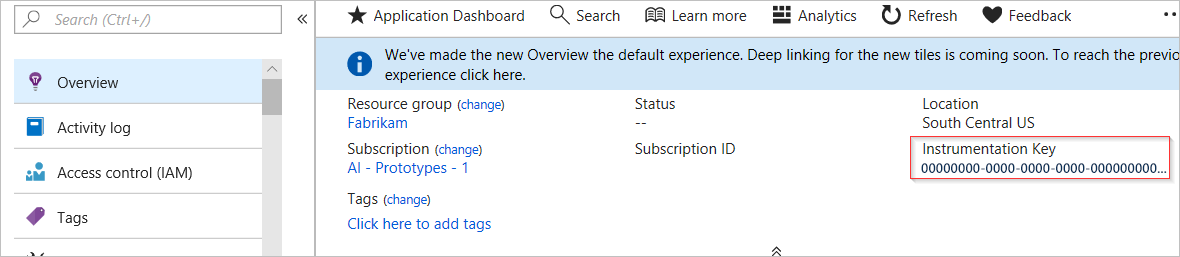
**Retention of Data in Azure Application Insight:**

The retention period for data in Azure Application Insights varies depending on the pricing tier of the Application Insights instance. Here are the retention periods for each pricing tier:

* **Free and Basic**: The retention period for Application Insights data is 7 days by default. You can increase the retention period to 90 days by configuring a Continuous Export feature to export the data to an Azure storage account or by upgrading to the Standard tier.
* **Standard**: The retention period for Application Insights data is 90 days by default. You can increase the retention period up to 730 days (2 years) by configuring a Continuous Export feature to export the data to an Azure storage account.

**To integrate Application Insights with your applications:**

1. Set up Application Insights resource
2. Install an instrumentation package in your application.



**There are two ways to configure your app to send data to Application Insights:**

* **Runtime instrumentation:** Runtime instrumentation captures telemetry without requiring you to change the web app’s source code. You can quickly enable this turnkey solution from the Azure portal when you first create your web app or anytime afterwards. Use this method when you want to set up Application Insights without involving developers or when code management policies prevent you from changing the app’s source code. Note that some advanced data displays aren’t available when you use only runtime instrumentation.
* **Build-time instrumentation.** With this method, developers add a server-side SDK to the web app’s code. For example, in an ASP.NET Core app, a developer could reference a **NuGet package** to access the SDK. When you instrument your app with the Application Insights SDK, you can enable full functionality and the richest set of visualizations in Application Insights. This type of instrumentation also enables you to add custom events and telemetry to your code to monitor unusual or unique behavior.

**Right Click on Project** 🡪 **Application Insight Telemetry**

**To Enable Server Side telemetry in the app:**

1. Add the NuGet Package:
   1. **Microsoft.ApplicationInsights.AspNetCore**
2. ConfigureSevice:

**services.AddApplicationInsightsTelemetry();**

1. appsettings.json

"**ApplicationInsights": { "InstrumentationKey": "putinstrumentationkeyhere" }**

Note: Set the browser window for the app side-by-side with the portal showing the Live Metrics Stream. Notice the incoming requests on the **Live Metrics Stream** as you navigate around the web app.

There are several parameters that you can set, although in most cases, you shouldn't need to. For example, you can disable or limit the number of AJAX calls reported per page view (to reduce traffic). Or you can set debug mode to have telemetry move rapidly through the pipeline without being batched.

To set these parameters, add them after the **instrumentationKey** as properties of the same JSON object.

**// Send telemetry immediately without batching.**

**// Remember to remove this when no longer required, as it can affect browser performance.**

enableDebug: boolean,

**// Don't log browser exceptions.**

disableExceptionTracking: boolean,

**// Don't log ajax calls.**

disableAjaxTracking: boolean,

**// Limit number of Ajax calls logged, to reduce traffic.**

maxAjaxCallsPerView: 10, // default is 500

**// Time page load up to execution of first trackPageView().**

overridePageViewDuration: boolean,

**// Set dynamically for an authenticated user.**

accountId: string,

**Adding JavaScript SDK**

* To automatically inject the JavaScript SDK and necessary configuration into pages served by your web app, add a new application setting named **APPINSIGHTS\_JAVASCRIPT\_ENABLED** and set the value to **true**.

OR

* add the below script to every page just before </head> (Ideally do in Master Layout page)

<script type="text/javascript">

var appInsights=window.appInsights||function(a){

function b(a){c[a]=function(){var b=arguments;c.queue.push(function(){c[a].apply(c,b)})}}var c={config:a},d=document,e=window;setTimeout(function(){var b=d.createElement("script");b.src=a.url||"https://az416426.vo.msecnd.net/scripts/a/ai.0.js",d.getElementsByTagName("script")[0].parentNode.appendChild(b)});try{c.cookie=d.cookie}catch(a){}c.queue=[];for(var f=["Event","Exception","Metric","PageView","Trace","Dependency"];f.length;)b("track"+f.pop());if(b("setAuthenticatedUserContext"),b("clearAuthenticatedUserContext"),b("startTrackEvent"),b("stopTrackEvent"),b("startTrackPage"),b("stopTrackPage"),b("flush"),!a.disableExceptionTracking){f="onerror",b("\_"+f);var g=e[f];e[f]=function(a,b,d,e,h){var i=g&&g(a,b,d,e,h);return!0!==i&&c["\_"+f](a,b,d,e,h),i}}return c

}({ instrumentationKey:"<your instrumentation key>" });

window.appInsights=appInsights,appInsights.queue&&0===appInsights.queue.length&&appInsights.trackPageView();

</script>

OR

**\_ViewImports.cshtml, add injection**

@inject Microsoft.ApplicationInsights.AspNetCore.JavaScriptSnippet JavaScriptSnippet

**\_Layout.cshtml:**

<script>

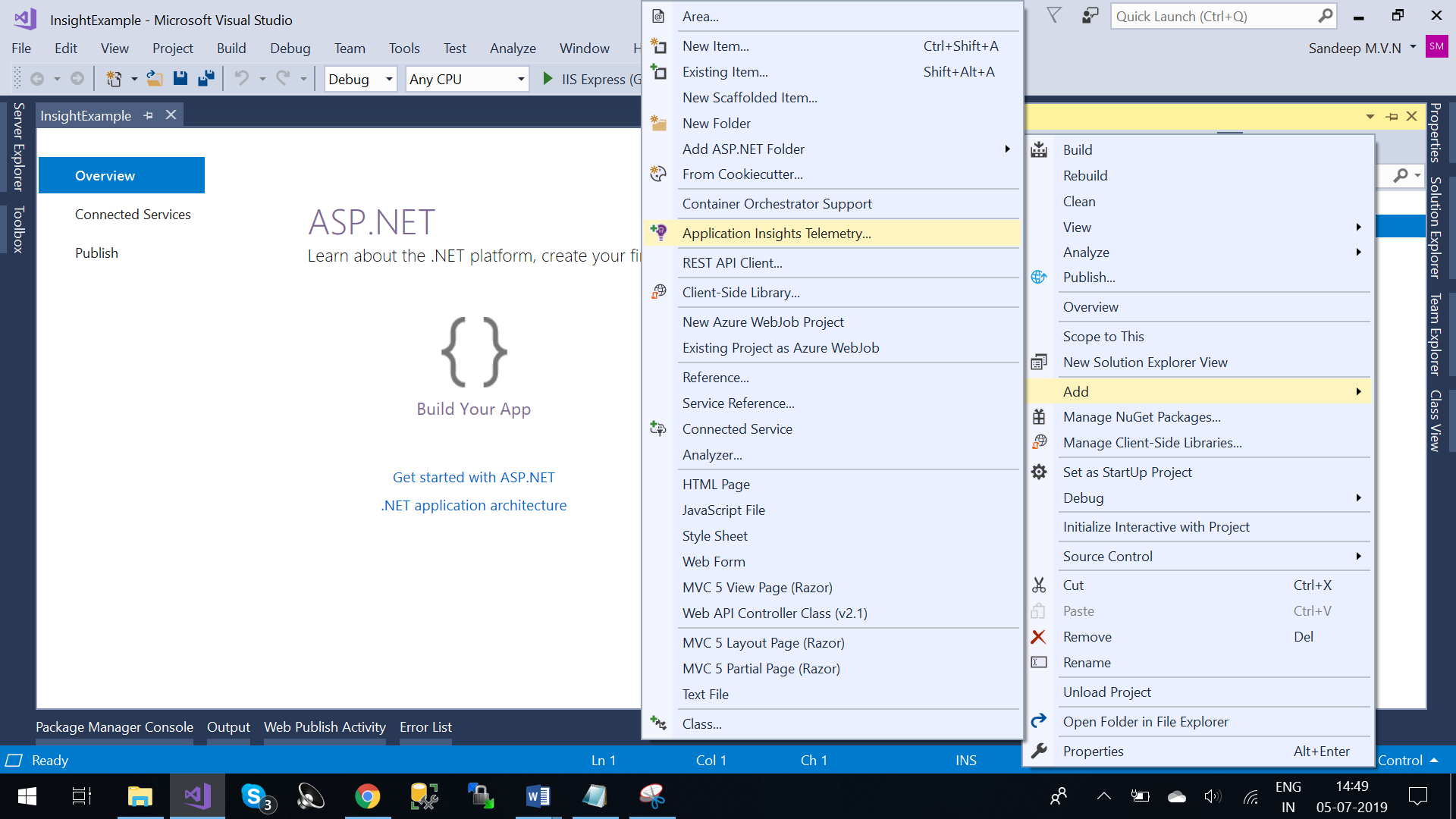
@Html.Raw(JavaScriptSnippet.ScriptBody)

</script>

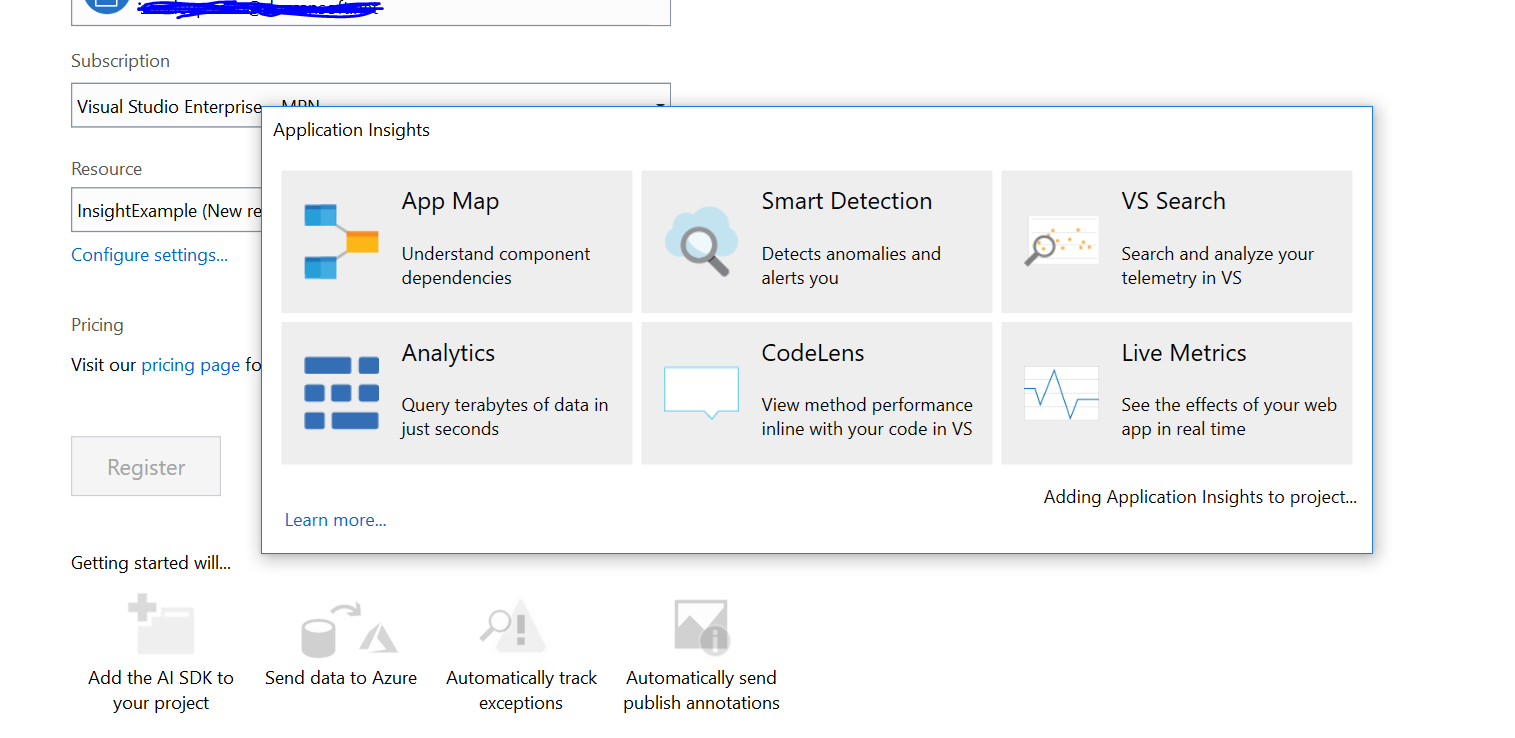
Integration with Visual Studio:

**Configuring Application Insights with an ASP.Net Applications**

1. Create a Sample ASP.Net Application
2. Right Click on Project🡪Click on Add🡪Select Application Insight Telemetry



1. Click on “Get Started”
2. Register the app with Application Insights
3. Provide your Microsoft account, subscription and Resource🡪Click on Register
4. Get Started will
   1. Add the AI SDK to Project
   2. Send data to azure
   3. Automatically track the exception
   4. Automatically send publish annotations
5. Application Insights includes



* 1. **App Map**: Application Map helps you spot performance bottlenecks or failure hotspots across all components of your distributed application
  2. **Smart Detection**: Smart Detection automatically warns you of potential performance problems and failure anomalies 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.
  3. **VS Search**: Search and Analyze your telemetry in Visual Studio
  4. **Analytics**
  5. **CodeLens:** View method performance inline with your code in visual studio
  6. **Live Metrics.**

1. After Adding the SDK to project, Let’s check what are the changes are made by application insights: -
   1. Packages: -
      1. Microsoft.AI.Agent.Intercept
      2. Microsoft.AI.DependecyCollector
      3. Microsoft.AI.PerfCounterCollector
      4. Microsoft.AI.ServerTelemetryChannel
      5. Microsoft.AI.Web
      6. Microsoft.AI.WindowsServer
   2. FilterConfig.cs

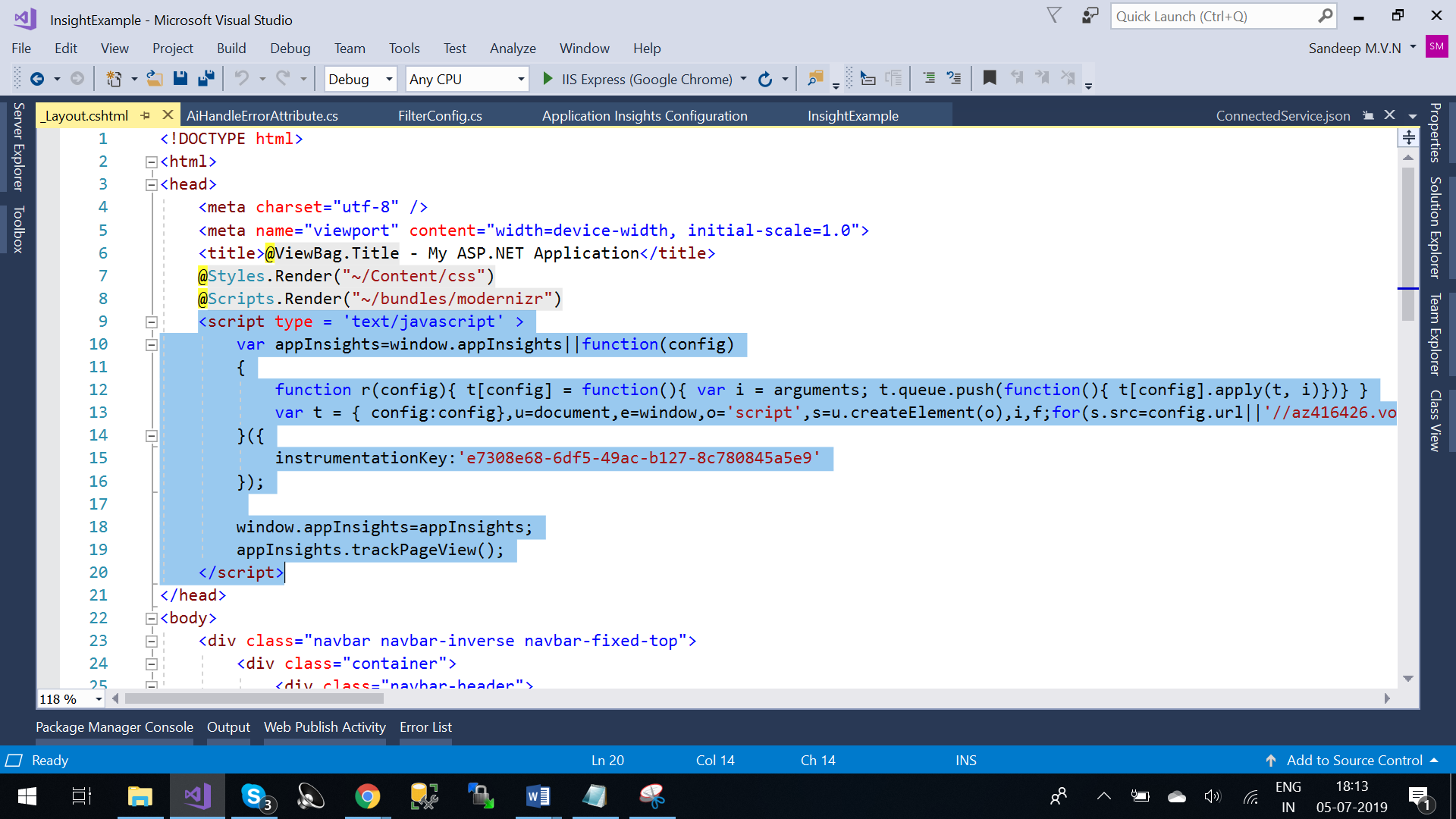
|  |
| --- |
| Filter.Add(new HandleErrorAttribute());  to  filters.Add(new ErrorHandler.AiHandleErrorAttribute()); |

* 1. Added ConnectedService.json

Graphical user interface, text, application, email

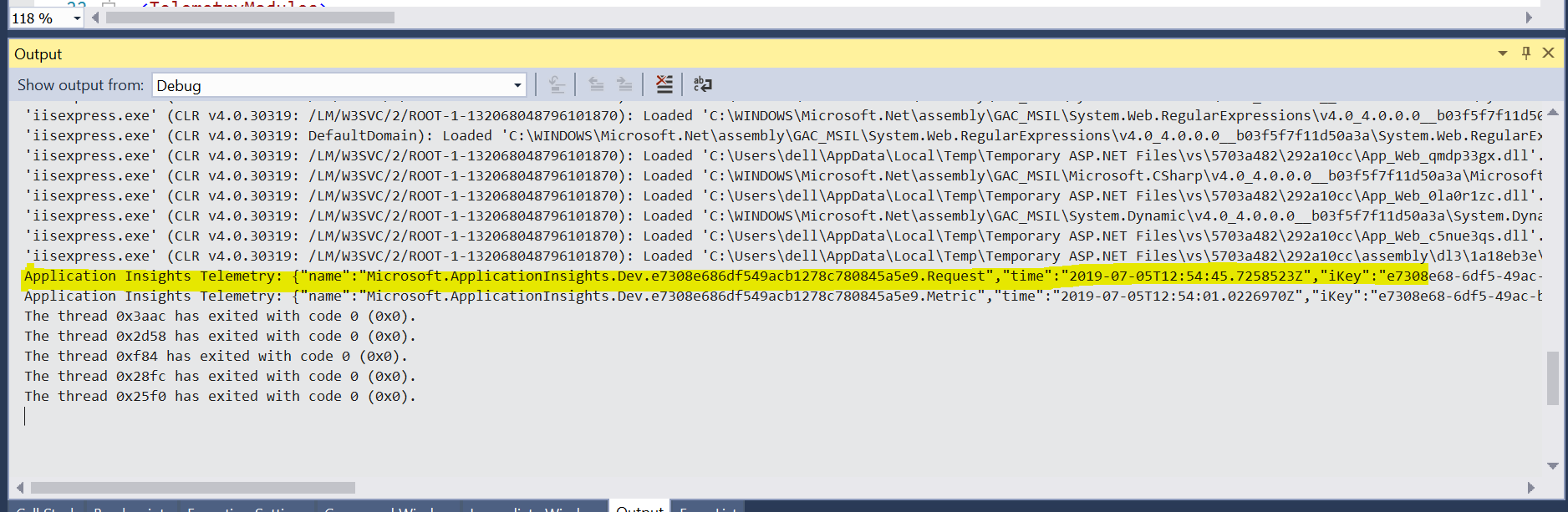
Description automatically generated

* 1. Added AiHandleErrorAttribute class for reporting the exception
  2. Layout.cshtml



* 1. ApplicationInsights.config:- Each Telemetry Module collects a specific type of data and uses the core API to send the data. The modules are installed by different NuGet packages, which also add the required lines to the .config file.
  2. When you set up Application Insights monitoring for your web app, you create an Application Insights resource in Microsoft Azure. You open this resource in the Azure portal in order to see and analyze the telemetry collected from your app. The resource is identified by an instrumentation key (which is going add the key in layout.cshtml and configuration file)

1. Once you have configured your Application Insights in project🡪Now run the project
2. We can view telemetry request in output window



Application Insights Telemetry: {"name":"Microsoft.ApplicationInsights.Dev.e7308e686df549acb1278c780845a5e9.Request","**time**":"2019-07-05T12:54:45.7258523Z","**iKey**":"e7308e68-6df5-49ac-b127-8c780845a5e9","tags":{"ai.internal.sdkVersion":"web:2.4.1-1362","**ai.internal.nodeName**":"DESKTOP-UOENECT","**ai.session.id**":"fq7P+","ai.operation.id":"6FsAfbJR7W0=","**ai.location.ip**":"::1","**ai.cloud.roleInstance**":"DESKTOP-UOENECT","**ai.user.id**":"W88ot","**ai.operation.name**":"GET Home/Index"},"**data**":{"baseType":"RequestData","baseData":{"ver":2,"id":"|6FsAfbJR7W0=.a8e57a89\_","**name**":"GET Home/Index","duration":"00:00:02.0677589","success":true,"**responseCode**":"200","url":"http://localhost:50822/","**properties**":{"DeveloperMode":"true**","\_MS.ProcessedByMetricExtractors**":"(Name:'Requests', Ver:'1.0')"}}}}