100 XP

Introduction

3 minutes

Instrumenting, and monitoring, your apps helps you maximize their availability and performance.

After completing this module, you'll be able to:

- Explain how Azure Monitor operates as the center of monitoring in Azure.
- Describe how Application Insights works and how it collects events and metrics.
- Instrument an app for monitoring, perform availability tests, and use Application Map to help you monitor performance and troubleshoot issues.

Next unit: Explore Azure Monitor

Continue

How are we doing? ☆ ☆ ☆ ☆ ☆

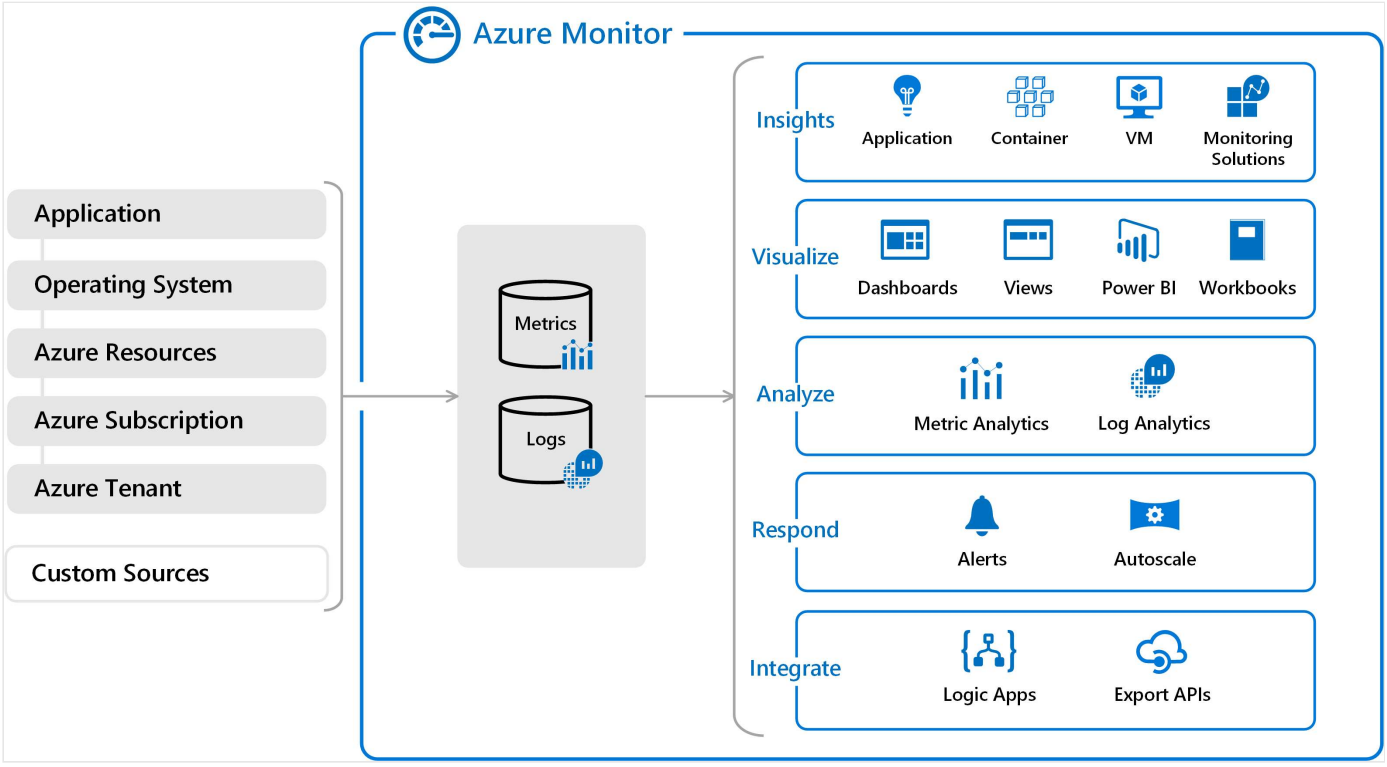
100 XP

Explore Azure Monitor

3 minutes

Azure Monitor delivers a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. This information helps you understand how your applications are performing and proactively identify issues affecting them and the resources they depend on.

The following diagram gives a high-level view of Azure Monitor. At the center of the diagram are the data stores for metrics and logs, which are the two fundamental types of data used by Azure Monitor. On the left are the sources of monitoring data that populate these data stores. On the right are the different functions that Azure Monitor performs with this collected data. This includes such actions as analysis, alerting, and streaming to external systems.



What data does Azure Monitor collect?

Azure Monitor can collect data from a variety of sources. This ranges 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**: Data about the performance and functionality of the code you have written, regardless of its platform.
- **Guest OS monitoring data**: Data about the operating system on which your application is running. This could be running in Azure, another cloud, or on-premises.
- **Azure resource monitoring data**: Data about the operation of an Azure resource. For a complete list of the resources that have metrics or logs, visit What can you monitor with Azure Monitor?
- **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.

Monitoring data platform

All data collected by Azure Monitor fits into one of two fundamental types, **metrics** and **logs**. 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.

For many Azure resources, you'll see metric data collected by Azure Monitor right in their Overview page in the Azure portal. Log data collected by Azure Monitor can be analyzed with queries to quickly retrieve, consolidate, and analyze collected data. You can create and test queries using Log Analytics in the Azure portal.

Insights and curated visualizations

Monitoring data is only useful if it can increase your visibility into the operation of your computing environment. Some Azure resource providers have a "curated visualization" which gives you a customized monitoring experience for that particular service or set of services. They generally require minimal configuration. Larger scalable curated visualizations are known at "insights" and marked with that name in the documentation and Azure portal. Some examples are:

- **Application Insights:** Application Insights monitors the availability, performance, and usage of your web applications whether they're hosted in the cloud or on-premises. It leverages the powerful data analysis platform in Azure Monitor to provide you with deep insights into your application's operations. It enables you to diagnose errors without waiting for a user to report them.
- **Container Insights:** Container Insights monitors the performance of container workloads that are deployed to managed Kubernetes clusters hosted on Azure Kubernetes Service (AKS) and Azure Container Instances. It gives you performance visibility by collecting metrics from controllers, nodes, and containers that are available in Kubernetes through the Metrics API. Container logs are also collected.
- **VM Insights:** VM Insights monitors your Azure virtual machines (VM) at scale. It analyzes the performance and health of your Windows and Linux VMs and identifies their different processes and interconnected dependencies on external processes.

Next unit: Explore Application Insights

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Explore Application Insights

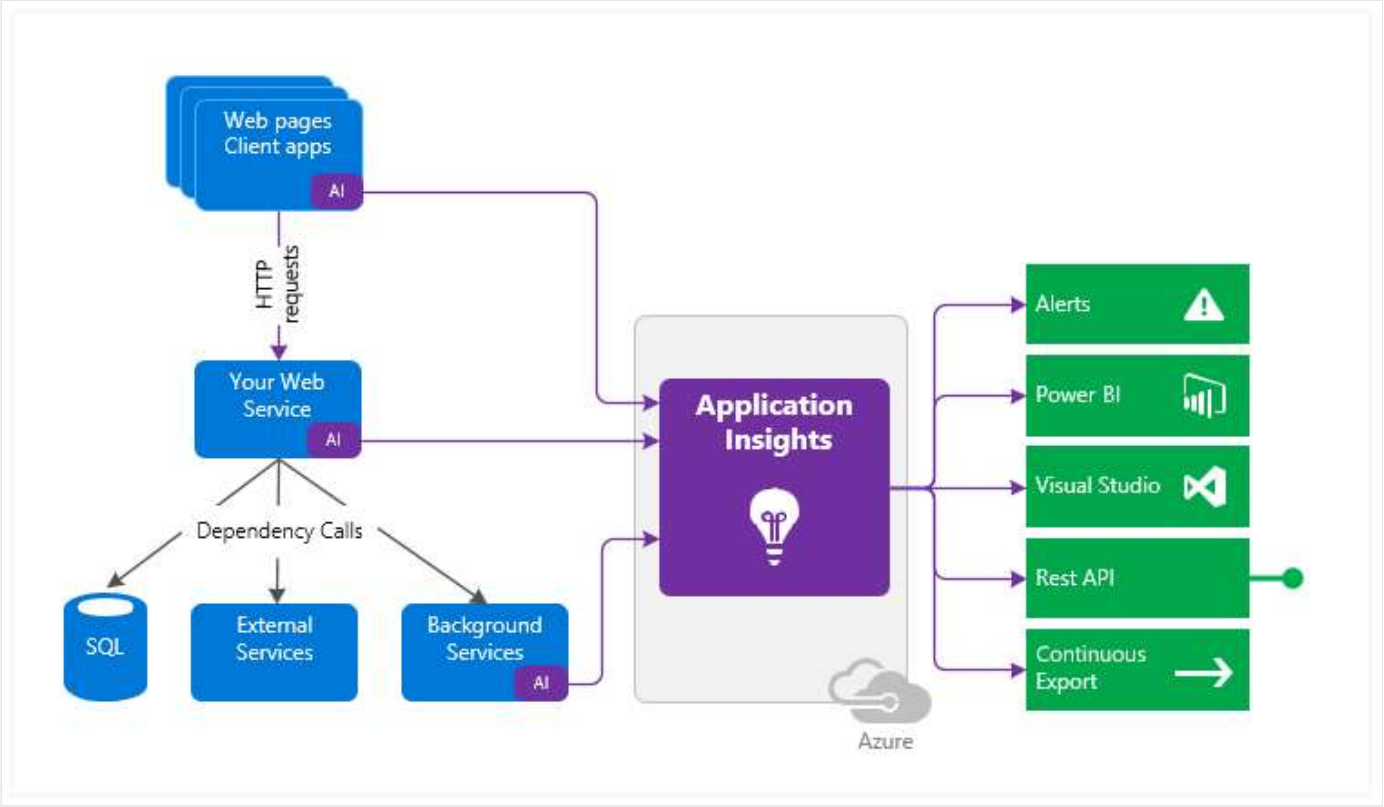
3 minutes

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.

How Application Insights works

You install a small instrumentation package (SDK) in your application or enable Application Insights using the Application Insights Agent when supported. The instrumentation monitors your app and directs the telemetry data to an Azure Application Insights Resource using a unique GUID that we refer to as an Instrumentation Key.

You can instrument not only the web service application, but also any background components, and the JavaScript in the web pages themselves. The application and its components can run anywhere - it doesn't have to be hosted in Azure.



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 into Azure Monitor. In the Azure portal, you can apply powerful analytic and search tools to the raw data. The impact on your app's performance is small. Tracking calls are non-blocking, and are batched and sent in a separate thread.

What Application 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 -** 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** 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.

Use Application Insights

Application Insights is one of the many services hosted within Microsoft Azure, and telemetry is sent there for analysis and presentation. It is free to sign up, and if you choose the basic pricing plan of Application Insights, there's no charge until your application has grown to have substantial usage.

There are several ways to get started monitoring and analyzing app performance:

- **At run time:** instrument your web app on the server. Ideal for applications already deployed. Avoids any update to the code.
- **At development time:** add Application Insights to your code. Allows you to customize telemetry collection and send additional telemetry.
- **Instrument your web pages** for page view, AJAX, and other client-side telemetry.
- **Analyze mobile app usage** by integrating with Visual Studio App Center.
- **Availability tests** - ping your website regularly from our servers.

Next unit: Discover log-based metrics

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

Discover log-based metrics

3 minutes

Application Insights log-based metrics let you analyze the health of your monitored apps, create powerful dashboards, and configure alerts. There are two kinds of metrics:

- **Log-based metrics** behind the scene are translated into Kusto queries from stored events.
- **Standard metrics** are stored as pre-aggregated time series.

Since *standard metrics* are pre-aggregated during collection, they have better performance at query time. This makes them a better choice for dashboarding and in real-time alerting. The *log-based metrics* have more dimensions, which makes them the superior option for data analysis and ad-hoc diagnostics. Use the namespace selector to switch between log-based and standard metrics in metrics explorer.

Log-based metrics

Developers can use the SDK to either emit these events manually (by writing code that explicitly invokes the SDK) or they can rely on the automatic collection of events from auto-instrumentation. In either case, the Application Insights backend stores all collected events as logs, and the Application Insights blades in the Azure portal act as an analytical and diagnostic tool for visualizing event-based data from logs.

Using logs to retain a complete set of events can bring great analytical and diagnostic value. For example, you can get an exact count of requests to a particular URL with the number of distinct users who made these calls. Or you can get detailed diagnostic traces, including exceptions and dependency calls for any user session. Having this type of information can significantly improve visibility into the application health and usage, allowing to cut down the time necessary to diagnose issues with an app.

At the same time, collecting a complete set of events may be impractical (or even impossible) for applications that generate a large volume of telemetry. For situations when the volume of events is too high, Application Insights implements several telemetry volume reduction techniques, such as sampling and filtering that reduce the number of collected and stored events. Unfortunately,

lowering the number of stored events also lowers the accuracy of the metrics that, behind the scenes, must perform query-time aggregations of the events stored in logs.

Pre-aggregated metrics

The pre-aggregated metrics are not stored as individual events with lots of properties. Instead, they are stored as pre-aggregated time series, and only with key dimensions. This makes the new metrics superior at query time: retrieving data happens much faster and requires less compute power. This consequently enables new scenarios such as near real-time alerting on dimensions of metrics, more responsive dashboards, and more.

> **ⓘ Important**
>
> Both, log-based and pre-aggregated metrics coexist in Application Insights. To differentiate the two, in the Application Insights UX the pre-aggregated metrics are now called "Standard metrics (preview)", while the traditional metrics from the events were renamed to "Log-based metrics".

The newer SDKs (Application Insights 2.7 SDK or later for .NET) pre-aggregate metrics during collection. This applies to standard metrics sent by default so the accuracy isn't affected by sampling or filtering. It also applies to custom metrics sent using GetMetric resulting in less data ingestion and lower cost.

For the SDKs that don't implement pre-aggregation (that is, older versions of Application Insights SDKs or for browser instrumentation) the Application Insights backend still populates the new metrics by aggregating the events received by the Application Insights event collection endpoint. This means that while you don't benefit from the reduced volume of data transmitted over the wire, you can still use the pre-aggregated metrics and experience better performance and support of the near real-time dimensional alerting with SDKs that don't pre-aggregate metrics during collection.

It is worth mentioning that the collection endpoint pre-aggregates events before ingestion sampling, which means that ingestion sampling will never impact the accuracy of pre-aggregated metrics, regardless of the SDK version you use with your application.

Next unit: Instrument an app for monitoring

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Instrument an app for monitoring

3 minutes

Earlier in this module we highlighted that an app can be monitored in Application Insights with, or without, adding code. In this unit we'll cover that in more detail.

Auto-instrumentation

Auto-instrumentation allows you to enable application monitoring with Application Insights without changing your code.

Application Insights is integrated with various resource providers and works on different environments. In essence, all you have to do is enable and - in some cases - configure the agent, which will collect the telemetry automatically. In no time, you'll see the metrics, requests, and dependencies in your Application Insights resource, which will allow you to spot the source of potential problems before they occur, and analyze the root cause with end-to-end transaction view.

The list of services that are supported by auto-instrumentation changes rapidly, visit this page for a list of what is currently supported.

Instrumenting for distributed tracing

Distributed tracing is the equivalent of call stacks for modern cloud and microservices architectures, with the addition of a simplistic performance profiler thrown in. In Azure Monitor, we provide two experiences for consuming distributed trace data. The first is our transaction diagnostics view, which is like a call stack with a time dimension added in. The transaction diagnostics view provides visibility into one single transaction/request, and is helpful for finding the root cause of reliability issues and performance bottlenecks on a per request basis.

Azure Monitor also offers an application map view which aggregates many transactions to show a topological view of how the systems interact, and what the average performance and error rates are.

How to enable distributed tracing

Enabling distributed tracing across the services in an application is as simple as adding the proper SDK or library to each service, based on the language the service was implemented in.

Enabling via Application Insights SDKs

The Application Insights SDKs for .NET, .NET Core, Java, Node.js, and JavaScript all support distributed tracing natively.

With the proper Application Insights SDK installed and configured, tracing information is automatically collected for popular frameworks, libraries, and technologies by SDK dependency auto-collectors. The full list of supported technologies is available in the Dependency auto-collection documentation.

Additionally, any technology can be tracked manually with a call to `TrackDependency` on the `TelemetryClient`.

Enable via OpenCensus

In addition to the Application Insights SDKs, Application Insights also supports distributed tracing through OpenCensus. OpenCensus is an open source, vendor-agnostic, single distribution of libraries to provide metrics collection and distributed tracing for services. It also enables the open source community to enable distributed tracing with popular technologies like Redis, Memcached, or MongoDB.

Next unit: Select an availability test

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Select an availability test

3 minutes

After you've deployed your web app or website, you can set up recurring tests to monitor availability and responsiveness. Application Insights sends web requests to your application at regular intervals from points around the world. It can alert you if your application isn't responding or responds too slowly.

You can set up availability tests for any HTTP or HTTPS endpoint that's accessible from the public internet. You don't have to make any changes to the website you're testing. In fact, it doesn't even have to be a site that you own. You can test the availability of a REST API that your service depends on.

You can create up to 100 availability tests per Application Insights resource, and there are three types of availability tests:

- URL ping test (classic): You can create this simple test through the portal to validate whether an endpoint is responding and measure performance associated with that response. You can also set custom success criteria coupled with more advanced features, like parsing dependent requests and allowing for retries.
- Standard test (Preview): This single request test is similar to the URL ping test. It includes SSL certificate validity, proactive lifetime check, HTTP request verb (for example `GET`, `HEAD`, or `POST`), custom headers, and custom data associated with your HTTP request.
- Custom TrackAvailability test: If you decide to create a custom application to run availability tests, you can use the TrackAvailability() method to send the results to Application Insights.

> **Note**
>
> **Multi-step test** is a fourth type of availability test, however that is only available through Visual Studio 2019. **Custom TrackAvailability test** is the long term supported solution for multi request or authentication test scenarios.

> **Important**

The **URL ping test** relies on the DNS infrastructure of the public internet to resolve the domain names of the tested endpoints. If you're using private DNS, you must ensure that the public domain name servers can resolve every domain name of your test. When that's not possible, you can use custom **TrackAvailability tests** instead.

Visit the troubleshooting article for guidance on diagnosing availability issues.

Next unit: Troubleshoot app performance by using Application Map

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Troubleshoot app performance by using Application Map

3 minutes

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 click through from any component to more detailed diagnostics, such as Application Insights events. If your app uses Azure services, you can also click through to Azure diagnostics, such as SQL Database Advisor recommendations.

Components are independently deployable parts of your distributed/microservices application. Developers and operations teams have code-level visibility or access to telemetry generated by these application components.

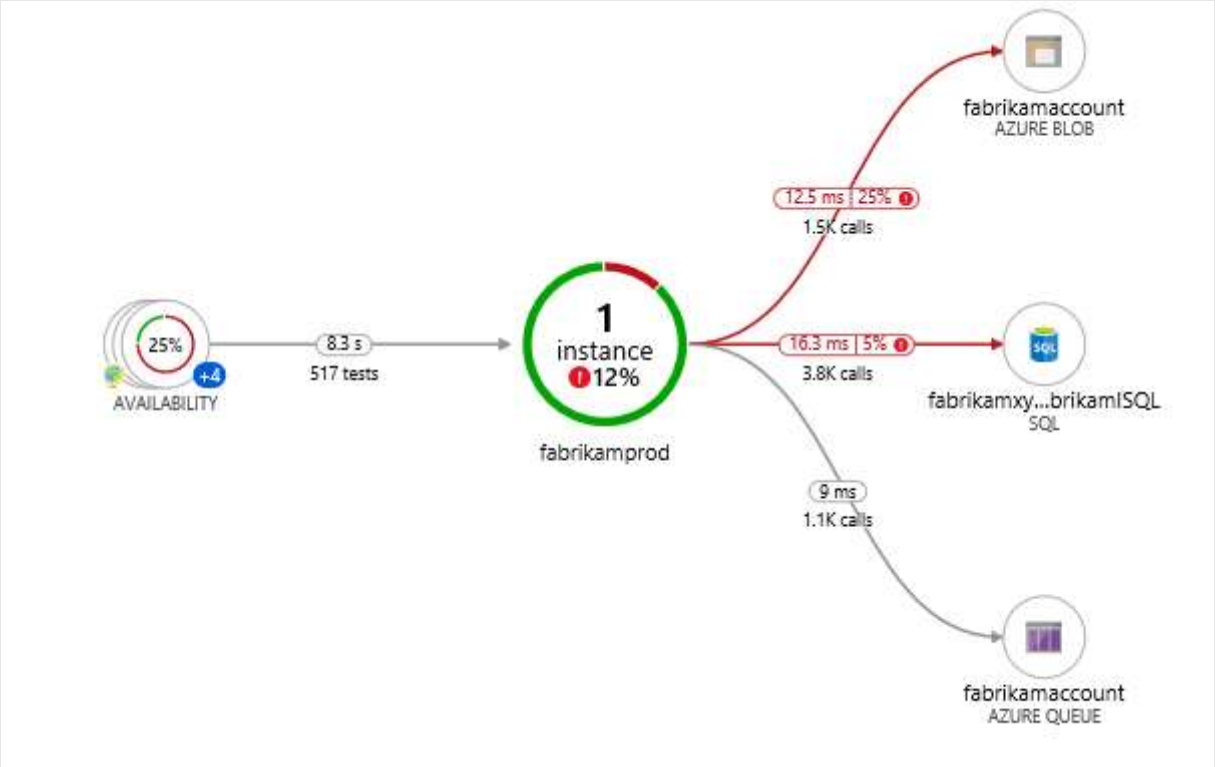
- Components are different from "observed" external dependencies such as SQL, Event Hubs, etc. which your team/organization may not have access to (code or telemetry).
- Components run on any number of server/role/container instances.
- Components can be separate Application Insights instrumentation keys (even if subscriptions are different) or different roles reporting to a single Application Insights instrumentation key. The preview map experience shows the components regardless of how they are set up.

You can see the full application topology across multiple levels of related application components. Components could be different Application Insights resources, or different roles in a single resource. The app map finds components by following HTTP dependency calls made between servers with the Application Insights SDK installed.

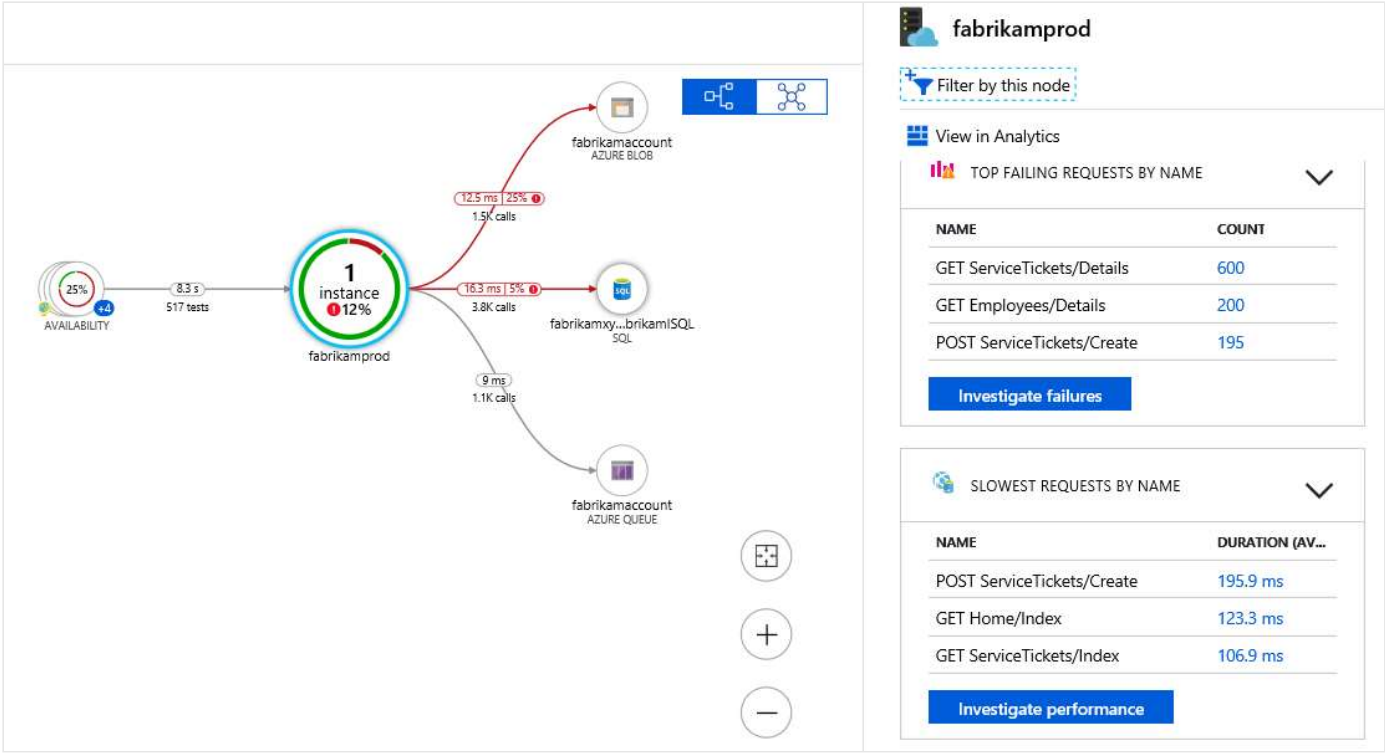
This experience starts with progressive discovery of the components. When you first load the application map, a set of queries is triggered to discover the components related to this component. A button at the top-left corner will update with the number of components in your application as they are discovered.

On clicking "Update map components", the map is refreshed with all components discovered until that point. Depending on the complexity of your application, this may take a minute to load.

If all of the components are roles within a single Application Insights resource, then this discovery step is not required. The initial load for such an application will have all its components.



One of the key objectives with this experience is to be able to visualize complex topologies with hundreds of components. Click on any component to see related insights and go to the performance and failure triage experience for that component.



Application Map uses the cloud role name property to identify the components on the map. You can manually set or override the cloud role name and change what gets displayed on the Application Map.

Next unit: Knowledge check

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Knowledge check

3 minutes

Check your knowledge

1. Which of the following availability tests is recommended for authentication tests?

- ○ URL ping
- ○ Standard
- ◉ Custom TrackAvailability

✔ **That's correct. Custom TrackAvailability test is the long term supported solution for multi request or authentication test scenarios.**

2. Which of the following metric collection types below provides near real-time querying and alerting on dimensions of metrics, and more responsive dashboards?

- ◉ Log-based

✘ **That's incorrect. Log-based metrics are aggregated at query time and require more processing to produce results.**

- ○ Pre-aggregated

✔ **That's correct. Pre-aggregated metrics are stored as a time series and only with key dimensions which enables near real-time alerting on dimensions of metrics, more responsive dashboards.**

- ○ Azure Service Bus

Next unit: Summary

Continue

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Summary

3 minutes

In this module, you learned how to:

- Explain how Azure Monitor operates as the center of monitoring in Azure.
- Describe how Application Insights works and how it collects events and metrics.
- Instrument an app for monitoring, perform availability tests, and use Application Map to help you monitor performance and troubleshoot issues.

Module complete:

Unlock achievement

How are we doing? ★ ★ ★ ★ ★ Great