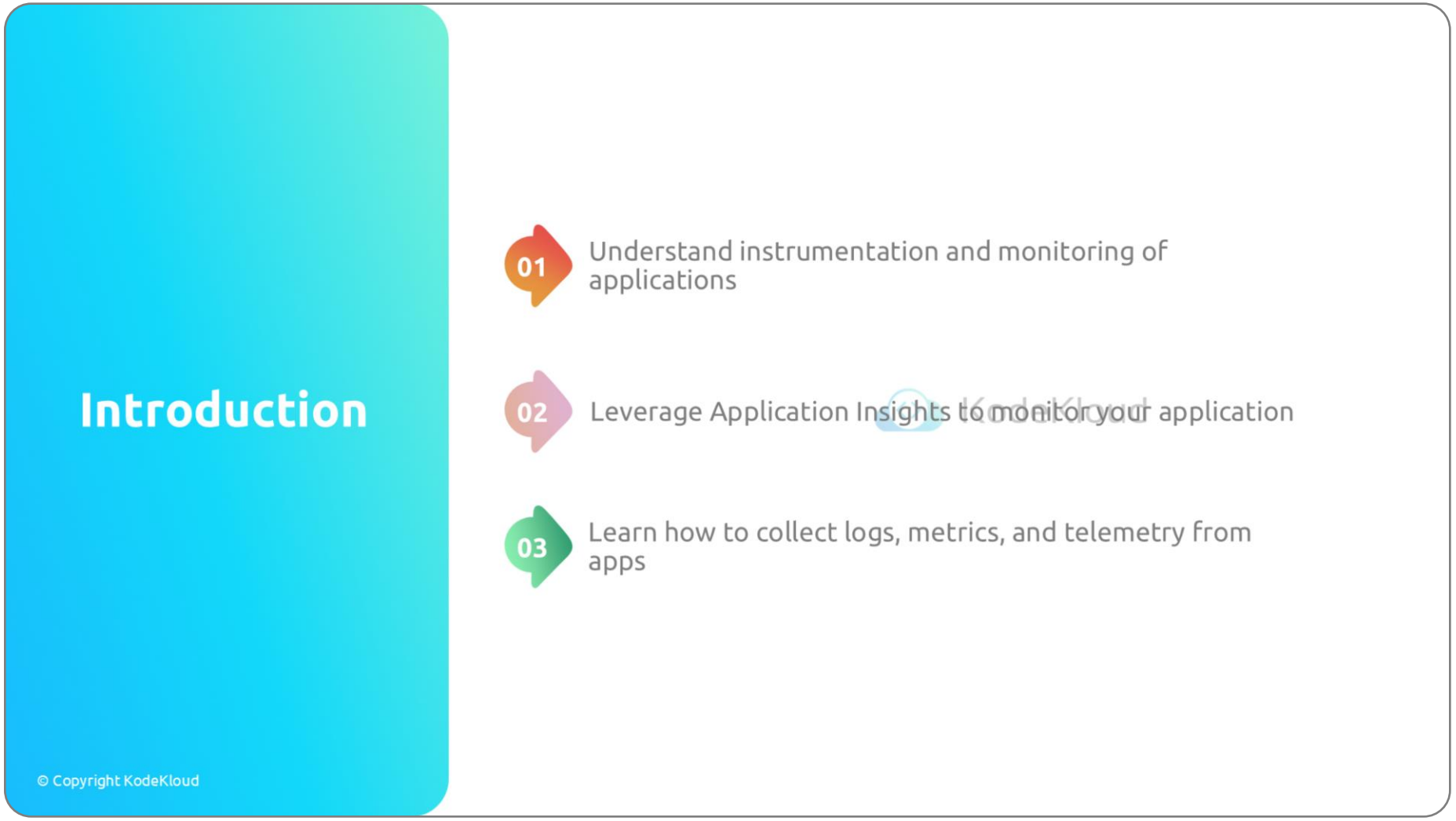
KodeKloud

Visit www.kodekloud.com to learn more.

Monitoring App Performance



Welcome to the section on Developing Message-Based Solutions in Azure. We'll start by laying down the foundation of message-based architecture in cloud applications and how Azure supports this pattern through its various services.

Choosing the Appropriate Queue Mechanism:

We will Begin by understanding the different queue mechanisms available in Azure. We'll compare Azure Queue Storage and Azure Service Bus, discussing scenarios where one might be more suitable than the other. Remember, the choice hinges on factors like the need for ordering, duplicate detection, transaction support, and the size and

rate of the messages.

Messaging Entities in Azure Service Bus:

We'll then get into the nitty-gritty of Azure Service Bus, focusing on its core messaging entities: Queues, Topics, and Subscriptions. Each plays a key role in ensuring messages flow smoothly between different parts of your system.

Using .NET with Service Bus Queues:

For those of you developing with .NET, we'll explore how to send and receive messages using Service Bus Queues. We will cover hands-on examples and best practices for integrating Service Bus with your .NET applications.

Key Components of Azure Queue Storage:

The we will provide an overview of Azure Queue Storage's key components, emphasizing how it differs from Service Bus and when it's the best fit for your workloads. We'll talk about queues, messages, and visibility timeouts.

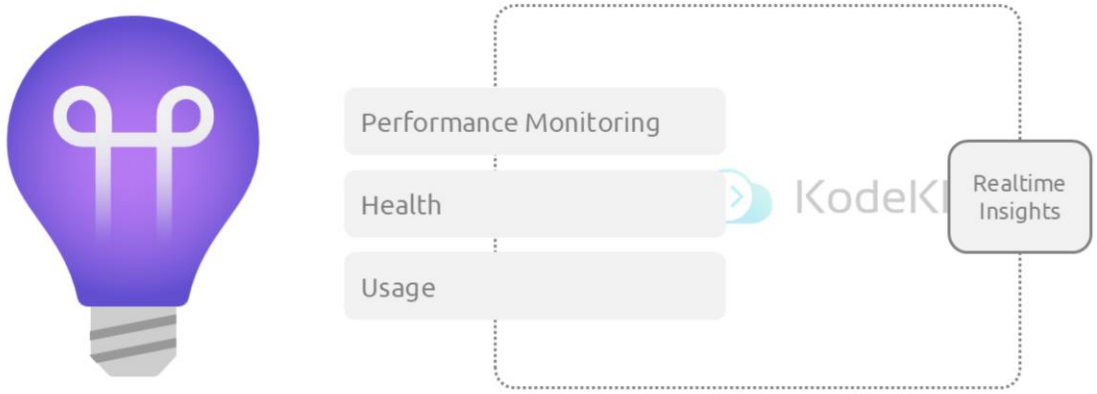
Managing Azure Queue Storage with .NET:

Finally, we'll wrap up this introduction by looking at how to create and manage queues in Azure Queue Storage using .NET. This will include practical demonstrations on pushing and pulling messages to ensure you have the skills to implement a robust message-based solution.

By the end of this module, you'll be confident in choosing and working with the right Azure messaging service to meet your application's needs.

Exploring Application Insights

Exploring Application Insights

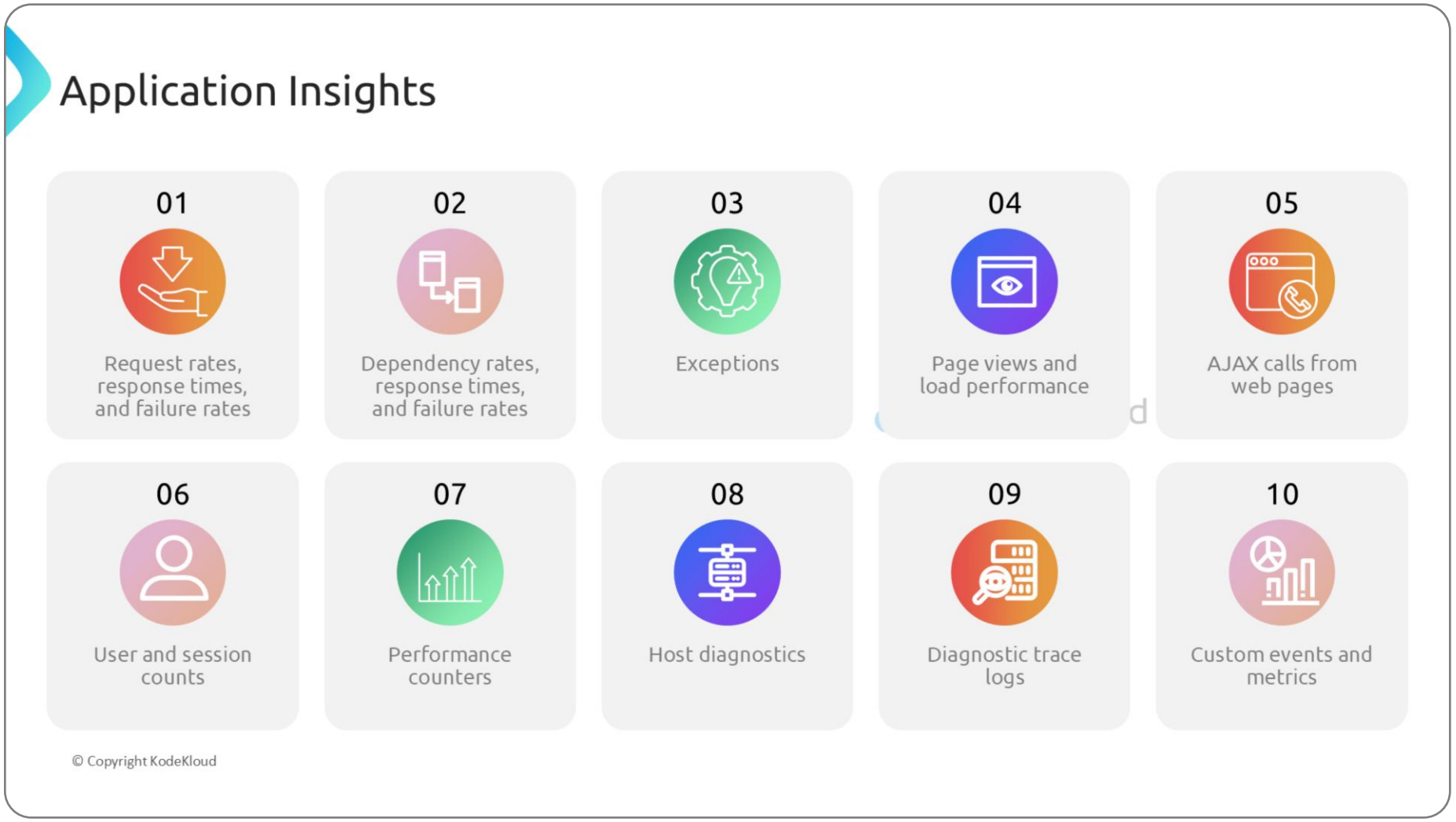


Exploring application insights In this module, we are going to explore the key features and capabilities of Azure Application Insights, a powerful tool for monitoring the performance, health, and usage of your applications As we go through these features, you'll see how application insights provide realtime insight into how your application is performing and where improvements can be made

Application Insights – Features

	Description
Live Metrics	Observe activity from your deployed application in real time with no effect on the host environment
Availability	Also known as “Synthetic Transaction Monitoring,” these probe your applications' external endpoint(s) to test the overall availability and responsiveness over time
GitHub/DevOps Integration	Create GitHub or Azure DevOps work items in the context of Application Insights data
Usage	Understand which features are popular with users and how users interact and use your application
Smart Detection	Automatic failure and anomaly detection through proactive telemetry analysis
Application Map	A high-level top-down view of the application architecture and at-a-glance visual references to component health and responsiveness
Distributed Tracing	Search and visualize an end-to-end flow of an execution or transaction

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.

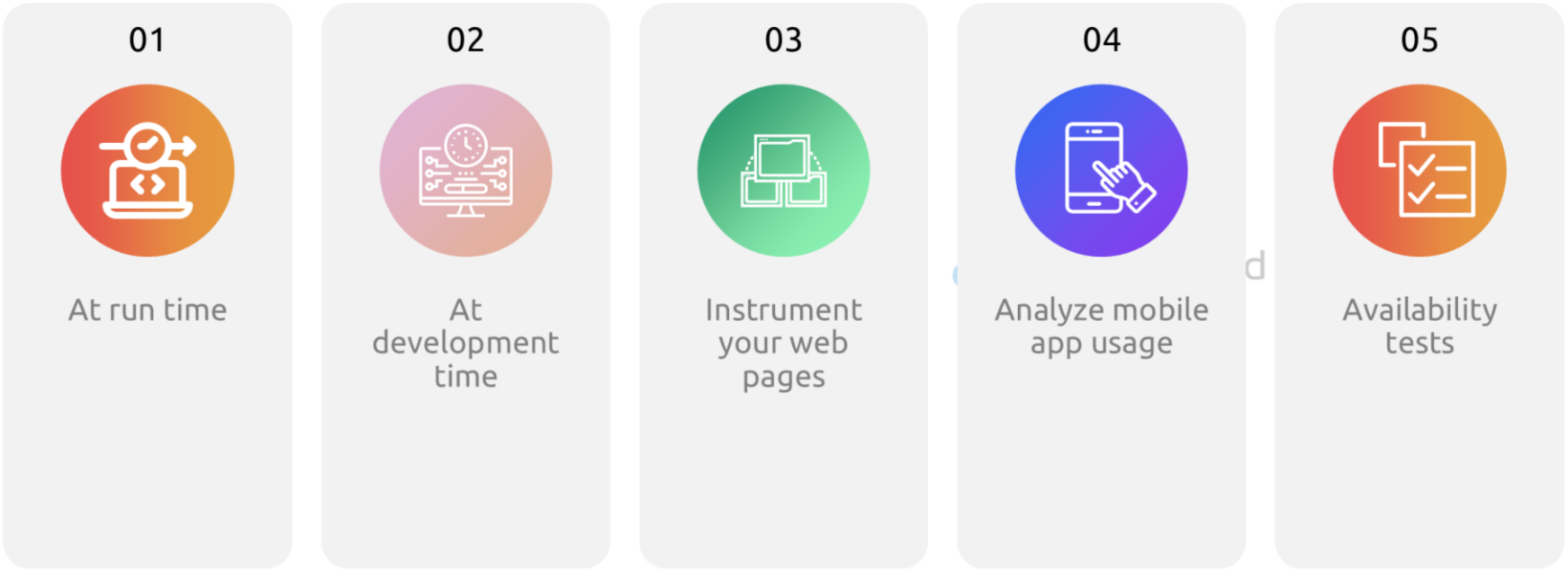


Application Insights monitors:

- Request rates, response times, and failure rates
- Dependency rates, response times, and failure rates
- Exceptions
- Page views and load performance
- AJAX calls from web pages

- User and session counts.
- Performance counters
- Host diagnostics
- Diagnostic trace logs
- Custom events and metrics

Application Insights

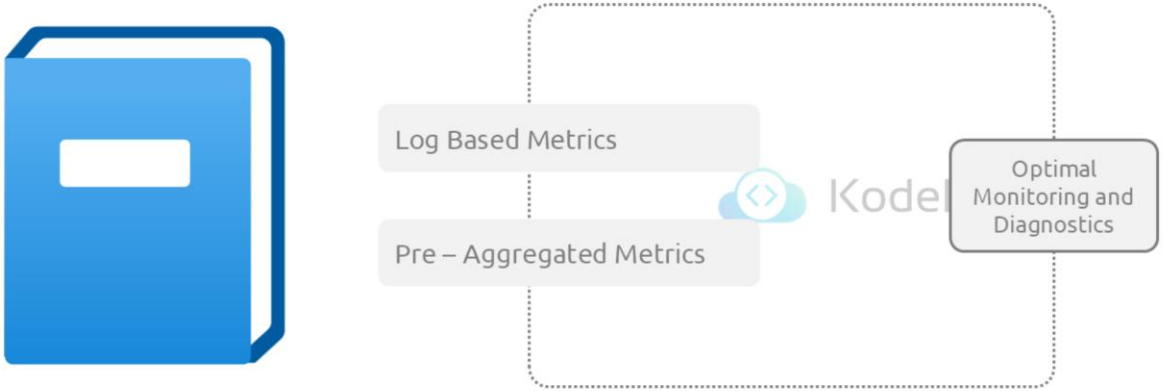


Several ways to get started monitoring and analyzing performance:

- At run time
- At development time
- Instrument your web pages
- Analyze mobile app usage
- Availability tests

Collecting Application Logs

Collecting Application Logs



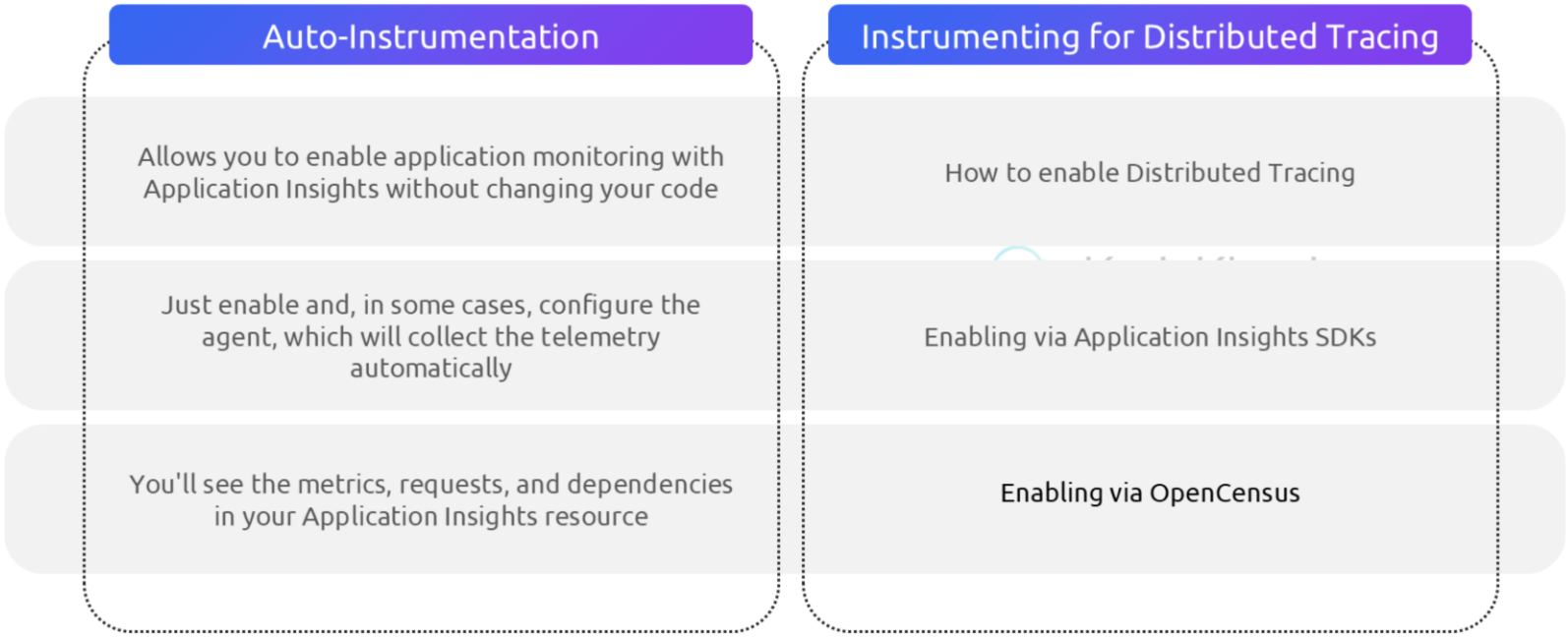
Collecting application logs In this slide, we will explore two fundamental metrics types within application insights logbased metrics and pre aggregated metrics, both serve different use cases and others essential to understand when to use each one for optimal monitoring and diagnostics in your applications

Log-Based Metrics Options

Log-Based Metrics	Pre-Aggregated Metrics
The Application Insights backend stores all collected events as logs.	These are stored as pre-aggregated time series, and only with key dimensions.
The Application Insights blades in the Azure portal act as an analytical and diagnostic tool for visualizing event-based data from logs.	The newer SDKs (Application Insights 2.7 SDK or later for .NET) pre-aggregate metrics during collection.
Using logs to retain a complete set of events can bring great analytical and diagnostic value.	For SDKs that don't implement pre-aggregation, the Application Insights backend still populates the new metrics by aggregating the events received by the Application Insights event collection endpoint.
Collecting a complete set of events may be impractical (or even impossible) for applications that generate a large volume of telemetry.	

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".

Instrumenting an App for Monitoring



Distributed tracing is the equivalent of call stacks for modern cloud and microservices architectures, with the addition of a simplistic performance profiler thrown in.

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.

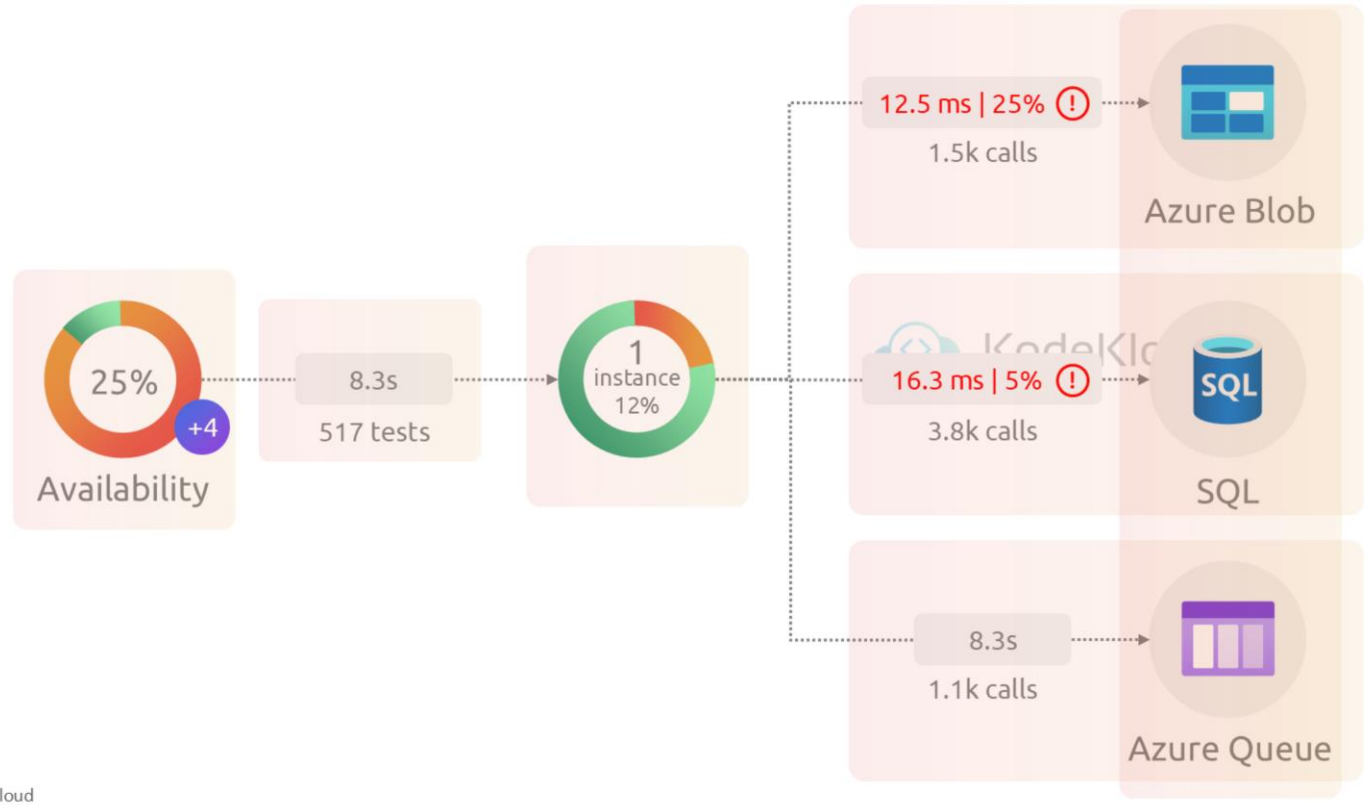
Additionally, any technology can be tracked manually with a call to TrackDependency on the TelemetryClient.

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.

Troubleshooting App Performance by Using Application Map

Troubleshooting App Performance by Using Application Map



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
Components are independently deployable parts of your distributed/microservices application

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.

KodeKloud

Visit www.kodekloud.com to learn more.