



# FOSSASIA 2017

17-19<sup>th</sup> March 2017

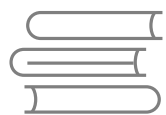
Singapore



# Improving fault detection and real time analytics with Telemetry

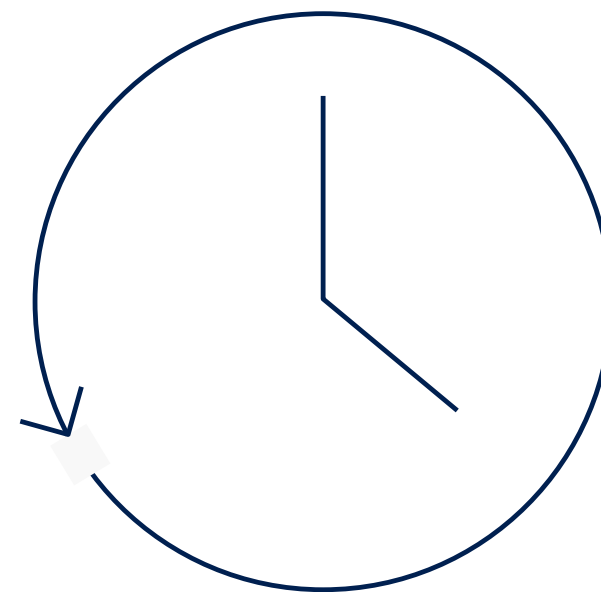
Sudheesh Singanamalla  
Research Fellow





1450

Printing Press



# Agenda of the talk

1. Overview
2. Current Issues with Debugging and Performance Monitoring
3. Common Mistakes while Engineering
4. Fixing mistakes with Telemetry's help
5. Importance of Telemetry
  - Developers
  - Product Owners
6. Capturing the correct telemetry
7. Common Questions
8. Conclusion

# Issues with Debugging & Perf. Monitoring

- Hard to find / diagnose issues.
- Long and iterative diagnostic process for errors and bugs.
- Little insight into customer usage pattern.
- Making data-driven decisions.
- Lack of context and insights to resolve incidents.
- Inability to quickly detect; diagnose and triage issues.

# Challenges with Modern Software Engineering

- Large code bases.

Many times we find ourselves working in unfamiliar code or utilizing large libraries.

- Agile development changed the way we work.

Moving away from the waterfall models we now deliver and ship software faster.

- Continuous delivery is becoming a norm.

Automation is becoming important and serves as a benchmark to send out good quality and tested code.

# Common Mistakes while Engineering

- Assumptions about how the users will use the system.
- Not adopting a test driven development method.
- Manual release management and lack of CI Systems.
- <X>ity. (Configurability, Security, Scalability, Extensibility ...)
- Not implementing analytics and tracking within the app.
- Not implementing monetization procedures within the app.
- Over engineering the solutions.

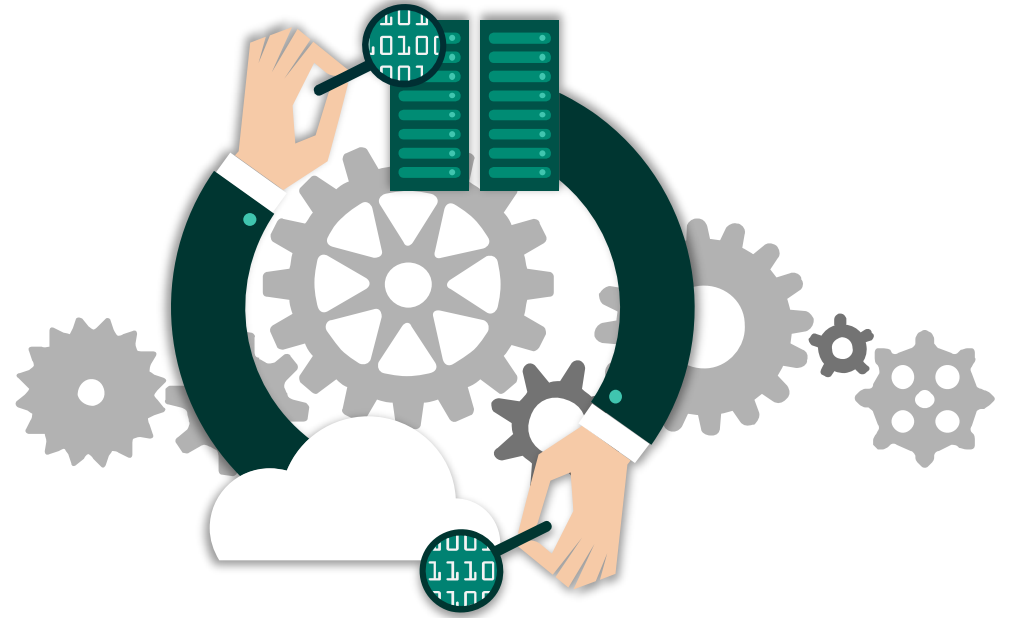


A classic case of  
over engineering.



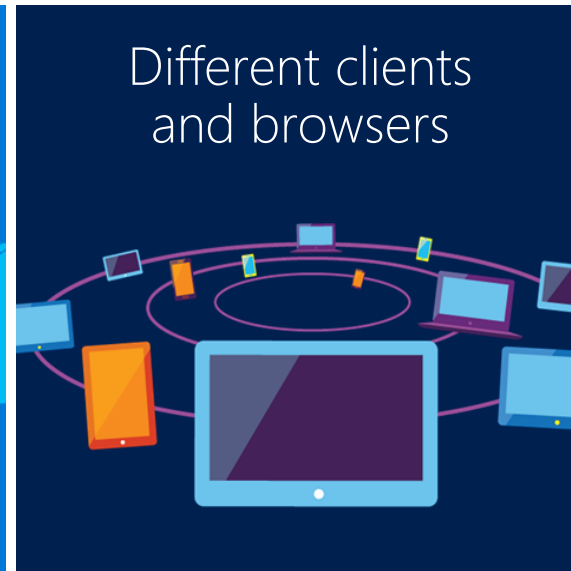
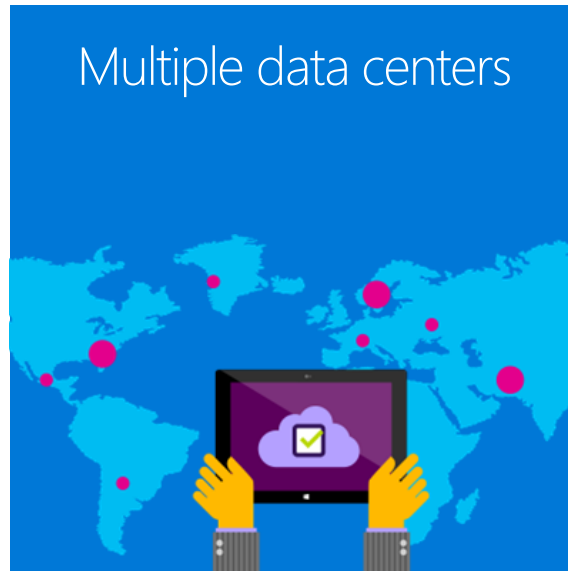
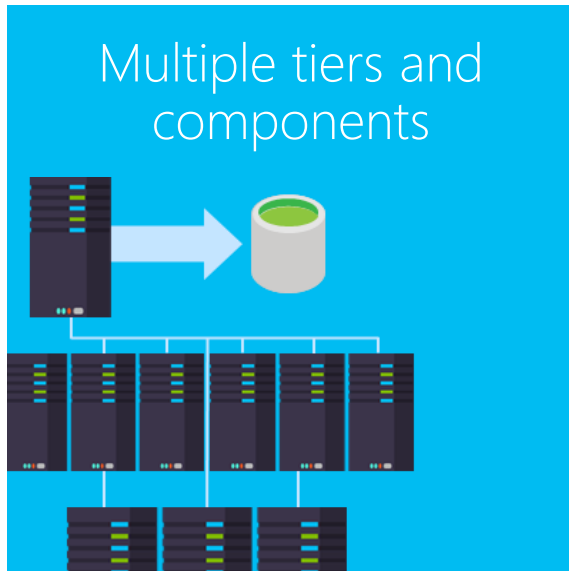
# What is Telemetry ?

Detect, Triage, Diagnose  
& Operationalize



# Why should I use Telemetry ?

- Diagnosing across the entire application stack across various data centers, components and usage pattern across the browsers is chaotic.



# Why should I use Telemetry ?

- Simplifies the entire diagnostic process used currently in software engineering and reduces time to fix a problem.

Customer complains about app issue

What is the overall error trend ?

Is the error unusual or recurring ?

When does the error occur and how frequent is it ?

Which set of users are having the worst experience ?

Understand user behavior and how the application was used

Did the users action cause the error ?

Is the page load time high ? Is the API response time high ?

Are there any other errors in the infrastructure ?

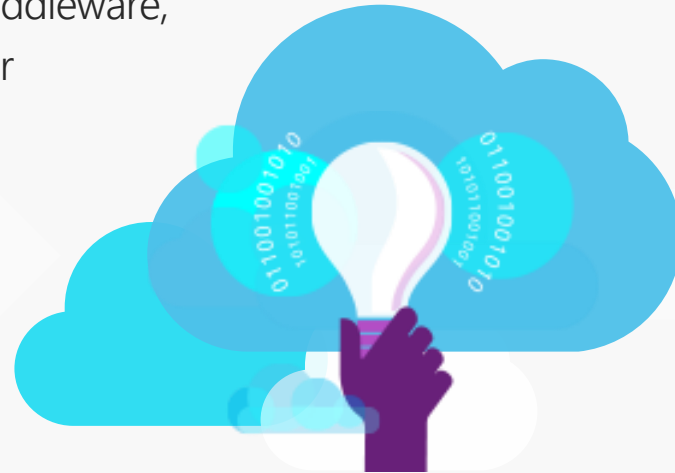
# Where should I use Telemetry ?

- Telemetry can be obtained and should be ideally obtained from all parts of your application.
  - Mobile Apps
  - Desktop Apps
  - Web Apps
  - Servers & Services
    - Infrastructure level telemetry like CPU usage, Crashes, Exception etc.,
    - Application level telemetry.

# How to Capture Telemetry ?

1

Telemetry is collected at each tier: server backend, middleware, web service & browser

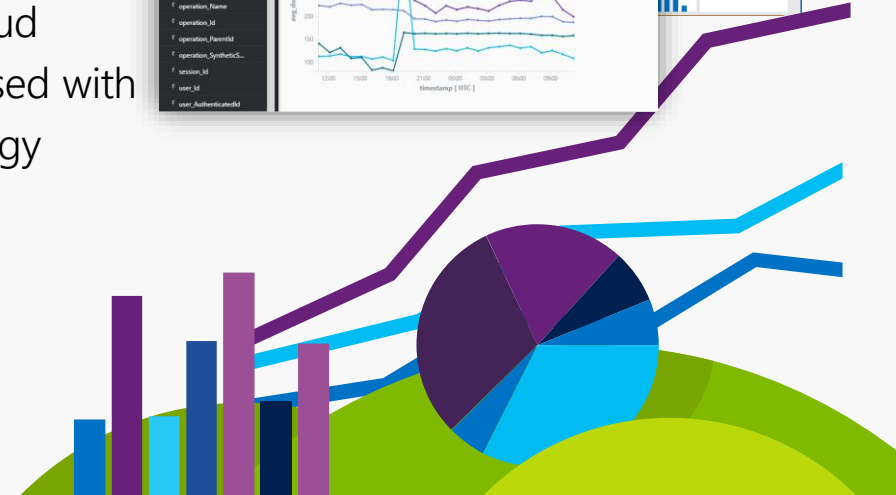
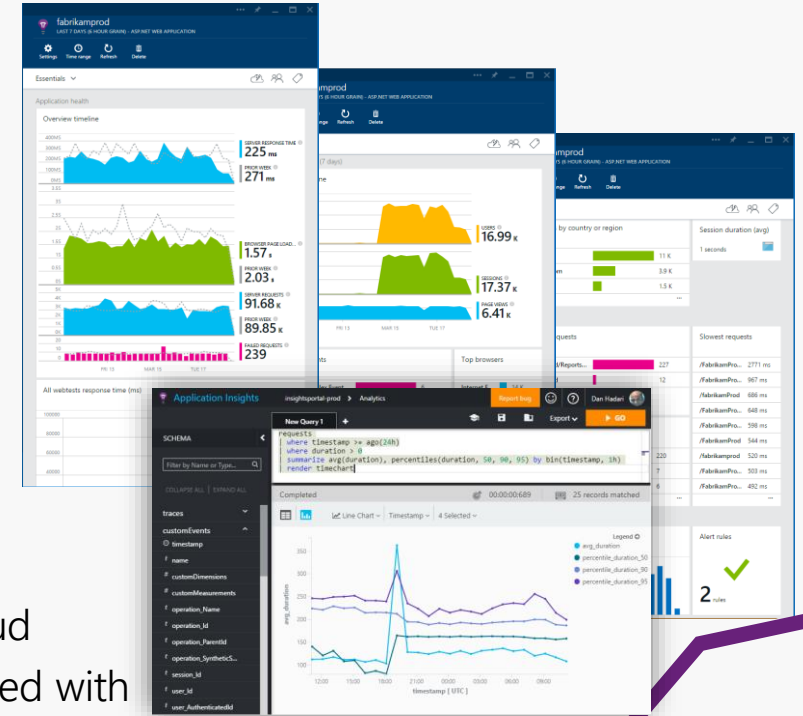


2

Telemetry arrives in the cloud where it is stored & processed with Machine Learning technology

3

Detect & Diagnose problems in Azure Portal; Ask ad-hoc queries in Analytics; Integrate, Extend & Customize



# Importance of Telemetry for Developers

Know how the app is performing inside out.



- Total page load time
  - Client side processing time
  - Send request time
  - Receiving response time
- Dependency durations
  - SQL/NoSQL Databases
  - External APIs

# Importance of Telemetry for Developers

Know when and why there is an issue



- Failed Requests
- Exceptions on client side
- Exceptions on server side
- Failed external dependencies
- Abnormal rise in errors and faults
- Statistical and time based analysis
  - Affected 50 users
  - 75% had same exception
  - 89% failed INSERT DB queries



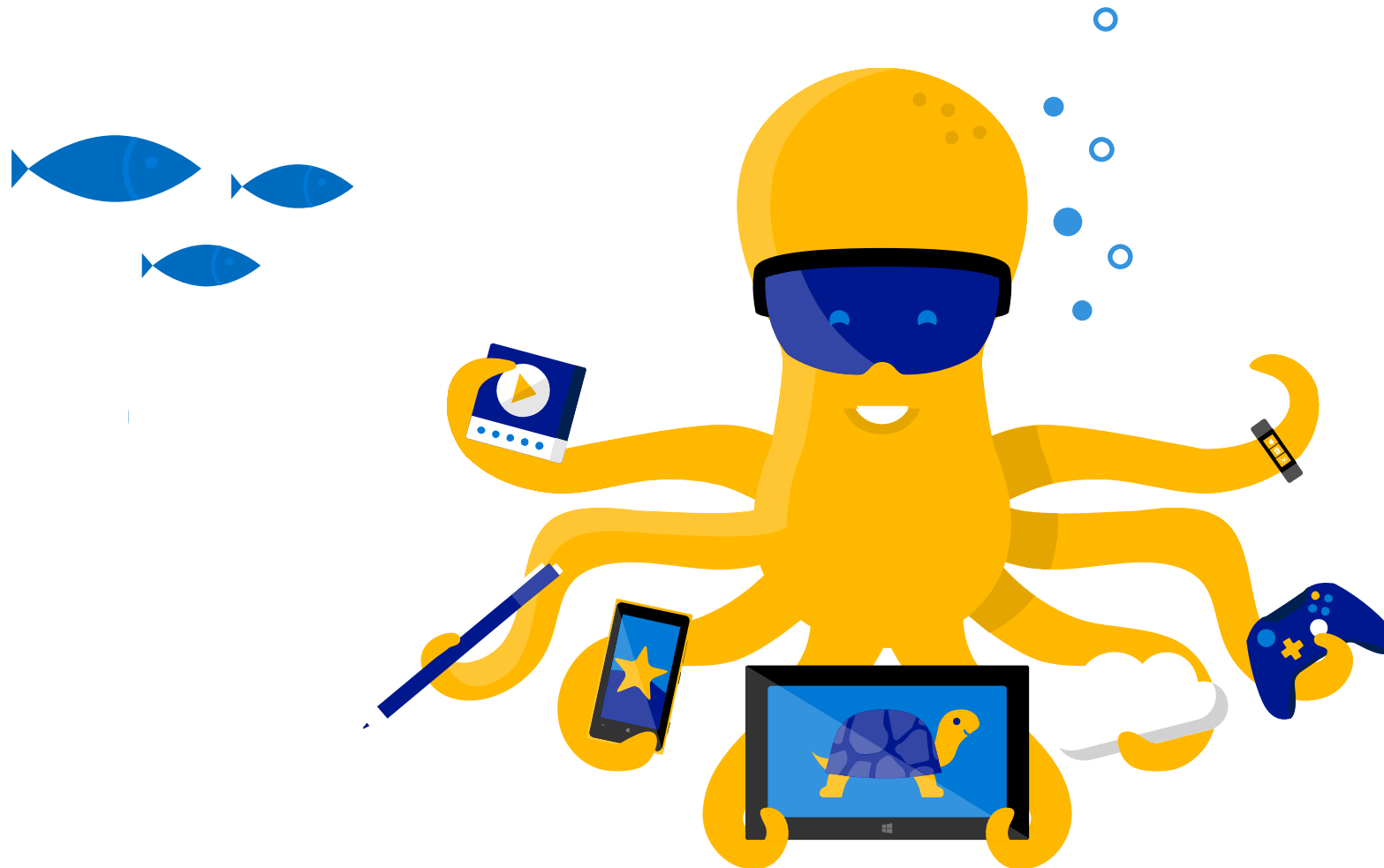
# Importance of Telemetry for Project Managers

Know which features deserve more attention

- Track features with Page Views and Custom Logging.
  - How many users actually use feature X ?
  - What type of user uses feature Y most ?
- “Flight” multiple implementations against each other
  - Release the same feature in different implementations or designs and compare usage adoption
  - Understand pain points from customers before they can complain about them



# Importance of Telemetry for Startups



Every crash or downtime could cost the startup potential users who would never visit it again.

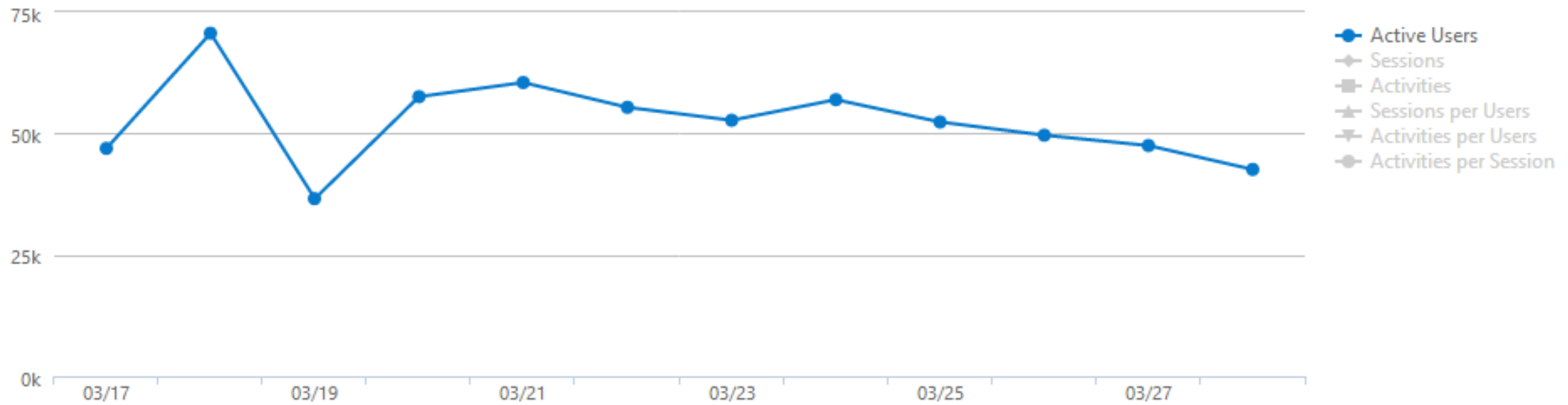
Data driven approach to marketing and understanding the users.

Delivering features and testing them out faster.

# Use Cases and Examples

# Tracking the launch of an app

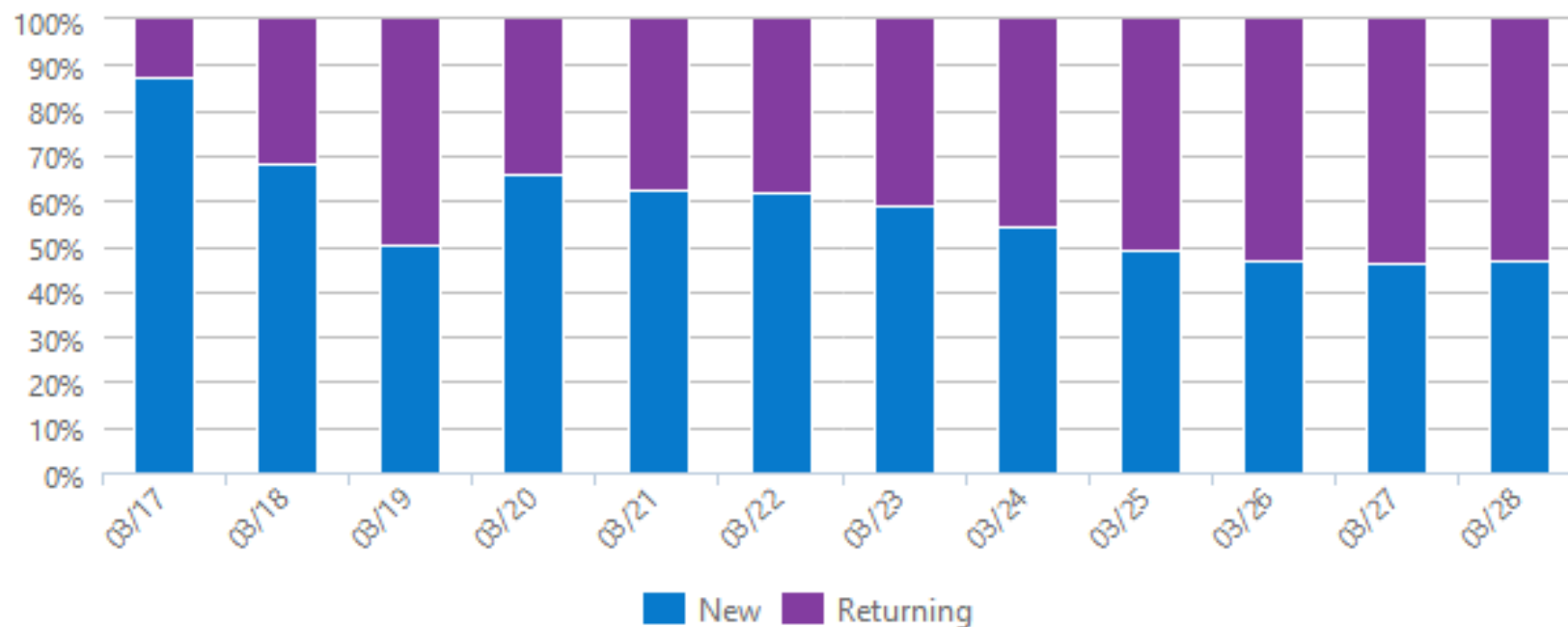
Trend of active users by day



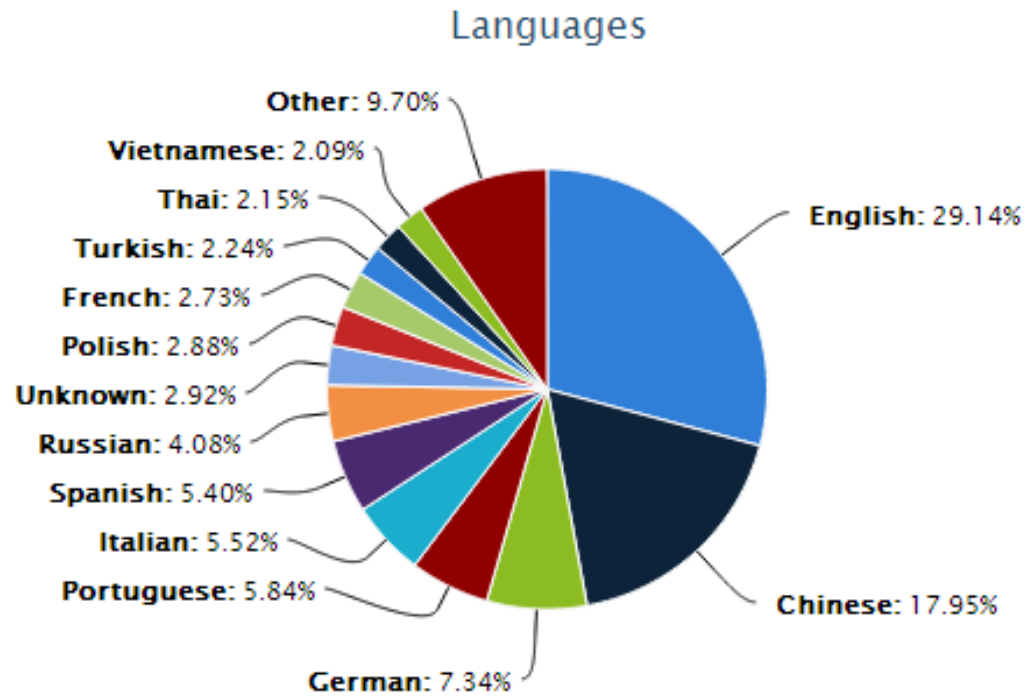
# Yes! Returning Users

## New vs. Returning

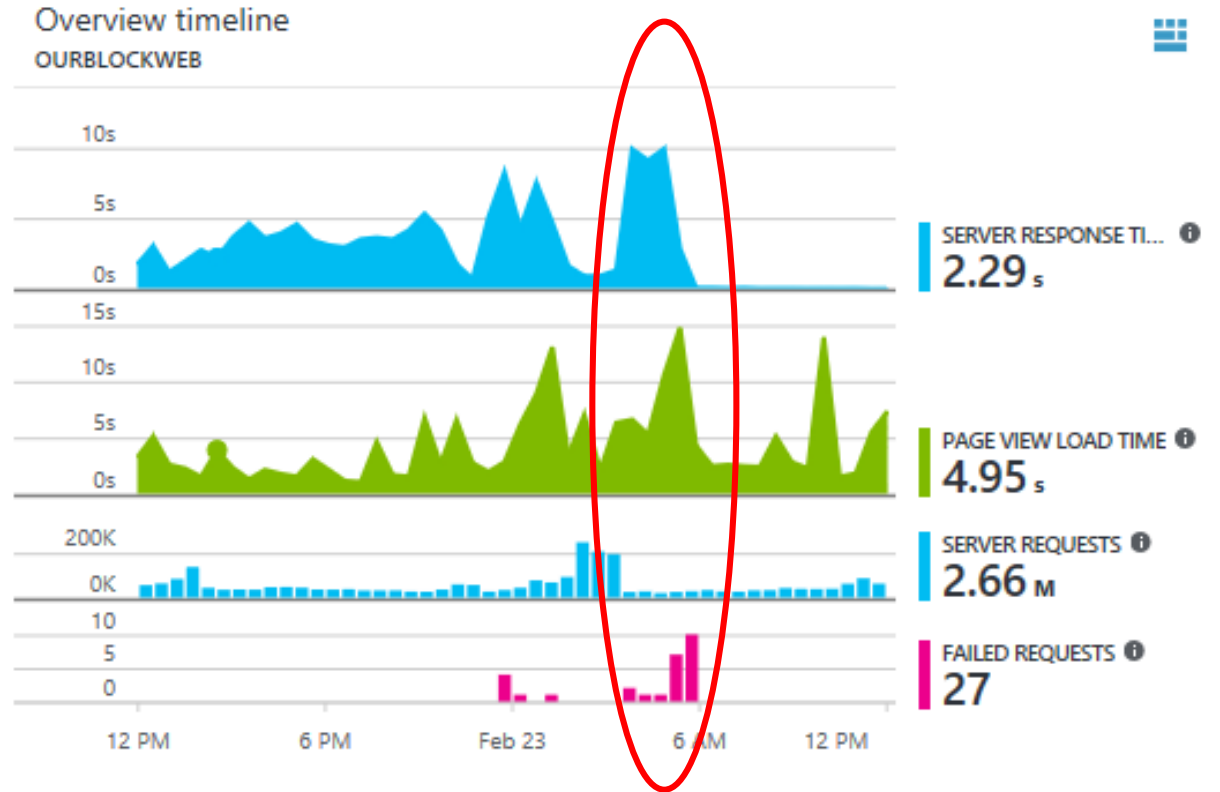
What percentage of sessions are from new users?



# User demographics and Failure detection.



Sessions: **741,170** Activities: **10,282,592**



# So How Do I use telemetry in my application ?



<https://github.com/Microsoft/ApplicationInsights-Home>



# Configuring for Server side applications

## Python

```
from applicationinsights import TelemetryClient
tc = TelemetryClient('<YOUR INSTRUMENTATION KEY>')
tc.track_event('WinGame', { 'Game': 'GameName', 'Difficulty': 'Hard' }, { 'GameScore': 20, 'Opponents': 1 })
tc.flush()
tc.track_metric('GameScore', 20, { 'Game': 'GameName', 'Difficulty': 'Hard' })
tc.flush()
```

## PHP

```
$tc = new ApplicationInsightsTelemetry_Client();
$tc->getContext()->setInstrumentationKey('YOUR INSTRUMENTATION KEY');
$tc->trackEvent('WinGame', ['Game' => 'GameName', 'Difficulty' => 'Hard'], ['GameScore'=> 20, 'Opponents' => 1]);
$tc->flush();
$tc->trackMetric('GameScore', 20, ['Game' => 'GameName', 'Difficulty' => 'Hard']);
$tc->flush();
```

And many other languages.



# Configuring for Client side applications

## What is Application Insights?



Application Insights is an extensible analytics service that monitors the clients, server and dependencies of your live web application. Detect, triage and diagnose performance issues and failures in the clients, server or dependencies of your application. Write your own events, metrics and traces for even more detailed usage analysis and diagnostic power. Application Insights is designed for developers, to help you continuously improve the availability, performance and usability of your app.

## What can Application Insights do?



### MONITOR AND DIAGNOSE SERVER SIDE APPLICATION

Detect server side performance issues and failures. Diagnose with correlated exceptions, dependency calls and your application traces.



### MONITOR AND DIAGNOSE CLIENT SIDE APPLICATION

Add our JavaScript snippet to analyze usage patterns and to detect and diagnose client side performance issues and failures.



### ENRICH TELEMETRY WITH CUSTOM METRICS AND EVENTS

How many red apples were sold today? How many users clicked the "Try-Now" button? How good is the new feature?



## Client side telemetry

Detect and diagnose performance issues and failures in web pages. Understand how your application is being used.


- See correlated client side, server side and custom telemetry, in the context of a user session, all in one place.
- Set up alerts on the client side metrics collected by default or custom metrics reported using the JavaScript SDK.
- Slice and dice client side metrics alongside server and custom telemetry to trace the causes of performance issues and failures.

### Learn more

[Application insights client side monitoring](#)

[Privacy statement](#)

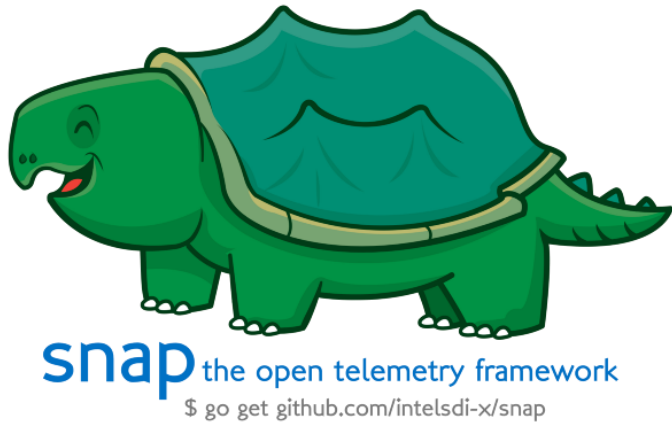
## Guidance

 Easy to get started. Simply paste the following into your master page

```
<!--
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.
      "https://az416426.vo.msecnd.net/scripts/a/ai.0.js";u.getElementsByTa
      ,i(c+a),i(h+v),i(c+v),i("flush"),config.disableExceptionTracking||(r
      ){{
        instrumentationKey:"0054a86f-8647-40e0-af97-ad3c314a75f5"
      }});

    window.appInsights=appInsights;
    appInsights.trackPageView();
  }
  </script>
```

# But Wait .... What are the alternatives ?



1. Completely open source.
2. Plugins to send data to AWS, Cassandra, RabbitMQ etc.,
3. Individual plugins required for each application like elasticsearch, SQL databases etc.,
4. Collected system level telemetry and not customizable.
5. Does not have a query language to filter and make sense of data.
6. Need to host it causing additional dev ops nightmares

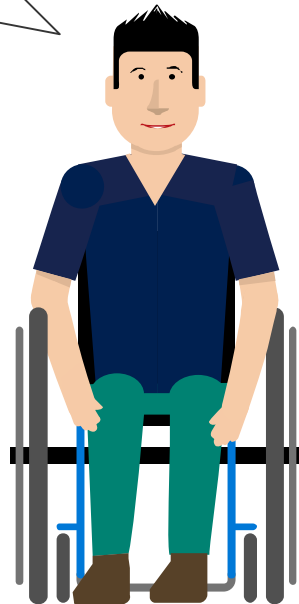
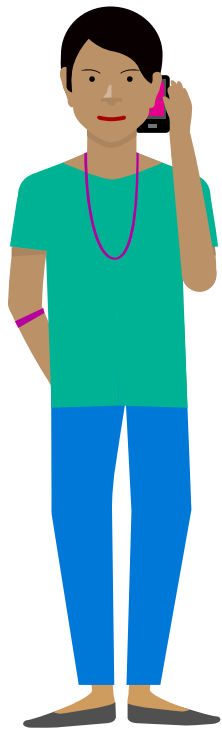


1. Not completely open source.
2. Rigorously tested and SaaS ensuring uptime.
3. Ability to fetch telemetry from any platform and services.
4. Query language for filtering on the data.
5. Easy exports to commonly used formats like CSV, JSON, XLSX and visualize with Power BI

# Common Questions

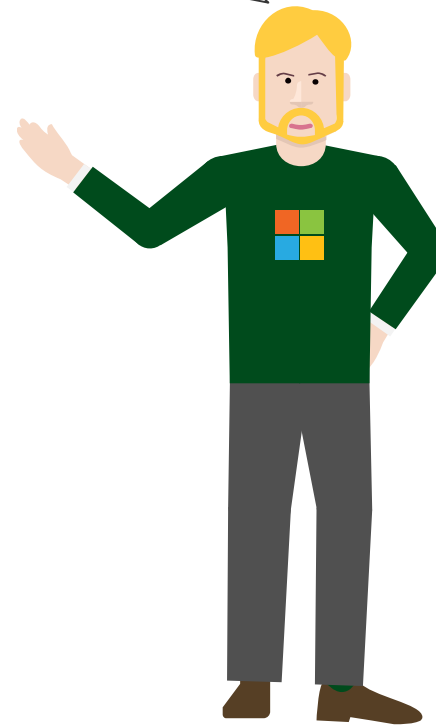
What about the system performance overhead due to telemetry ?

How long is my telemetry data available ?



How much does it cost ?

Will this slow down my webpages ?



# Impact of having a good Telemetry



Used by over 65% of fortune 500 companies.

- Ability to reliably query data and real time analytics of the sensor network
- Ability to move operations from on premise to cloud reliably for customers.

# Conclusion

- Always implement telemetry into your applications
- Use a telemetry platform to push all application telemetry into one single place.
- Make data driven decisions while shipping out features

# Open Source at Microsoft



Other tools like AirSIM, Azimuth, R, Maker.js, Naiad, Dryad

Microsoft Research: <https://github.com/MicrosoftResearch>

Microsoft: <https://github.com/Microsoft>

Azure: <https://github.com/Azure>

Office: <https://github.com/OfficeDev>

Sharepoint: <https://github.com/Sharepoint>

Microsoft Graph: <https://github.com/microsoftgraph>

Edge: <https://github.com/MicrosoftEdge>



# 1 Month Free Azure Passes

<https://aka.ms/cfdazurepass>



