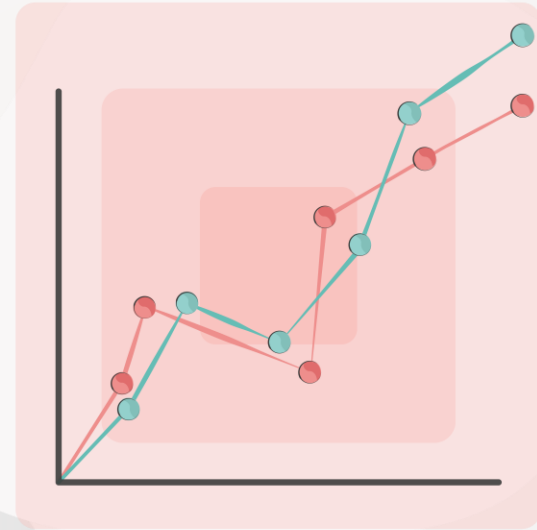




# Level 5 Data Engineer Module 5 Topic 7

## Putting it all together

Welcome to today's  
webinar.



# This week

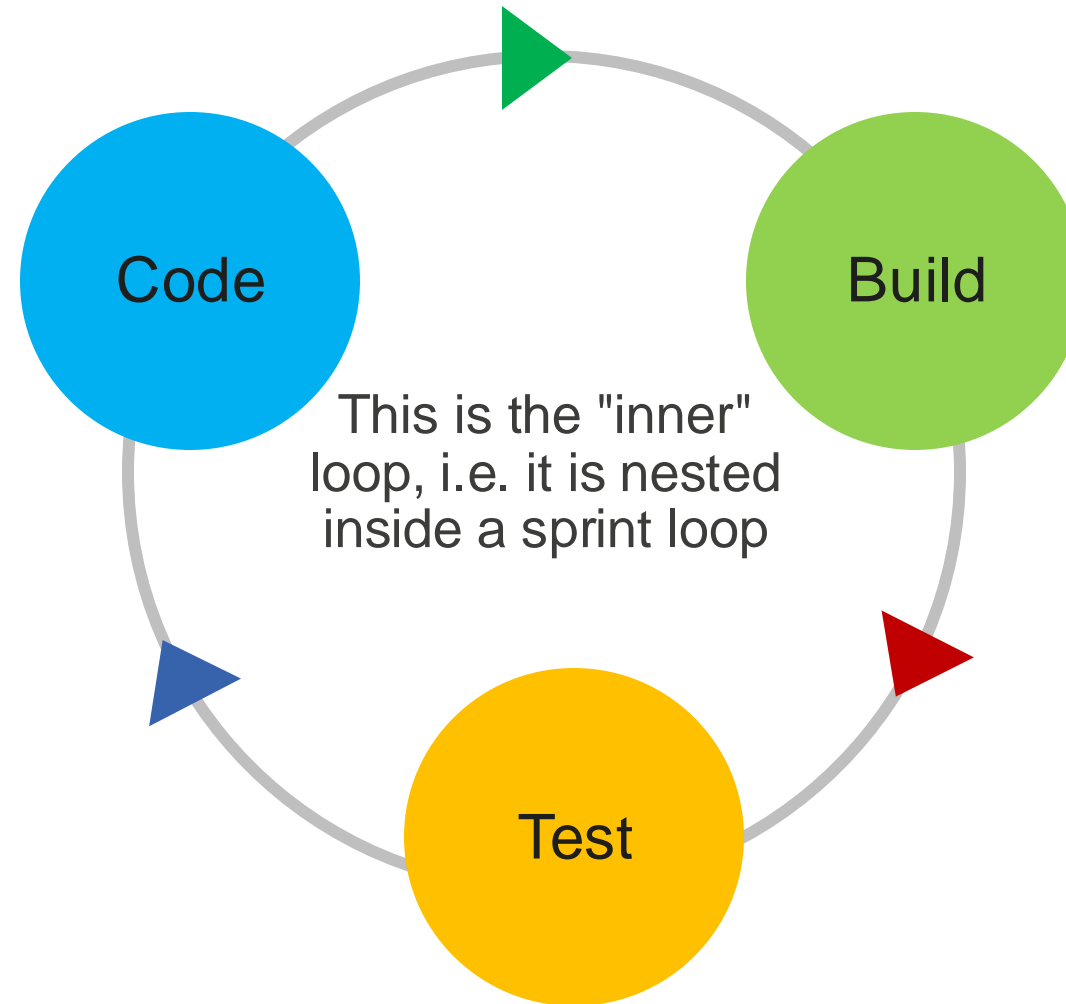
These are the learning outcomes for this week:

- ☐ Design cloud solutions based on user and business requirements.
- ☐ Evaluate use cases on whether they are suitable for cloud-based approaches.
- ☐ Monitor setup and running costs of cloud-native data products.
- ☐ Implement net-zero practices with right-sizing and auto-scaling policies.

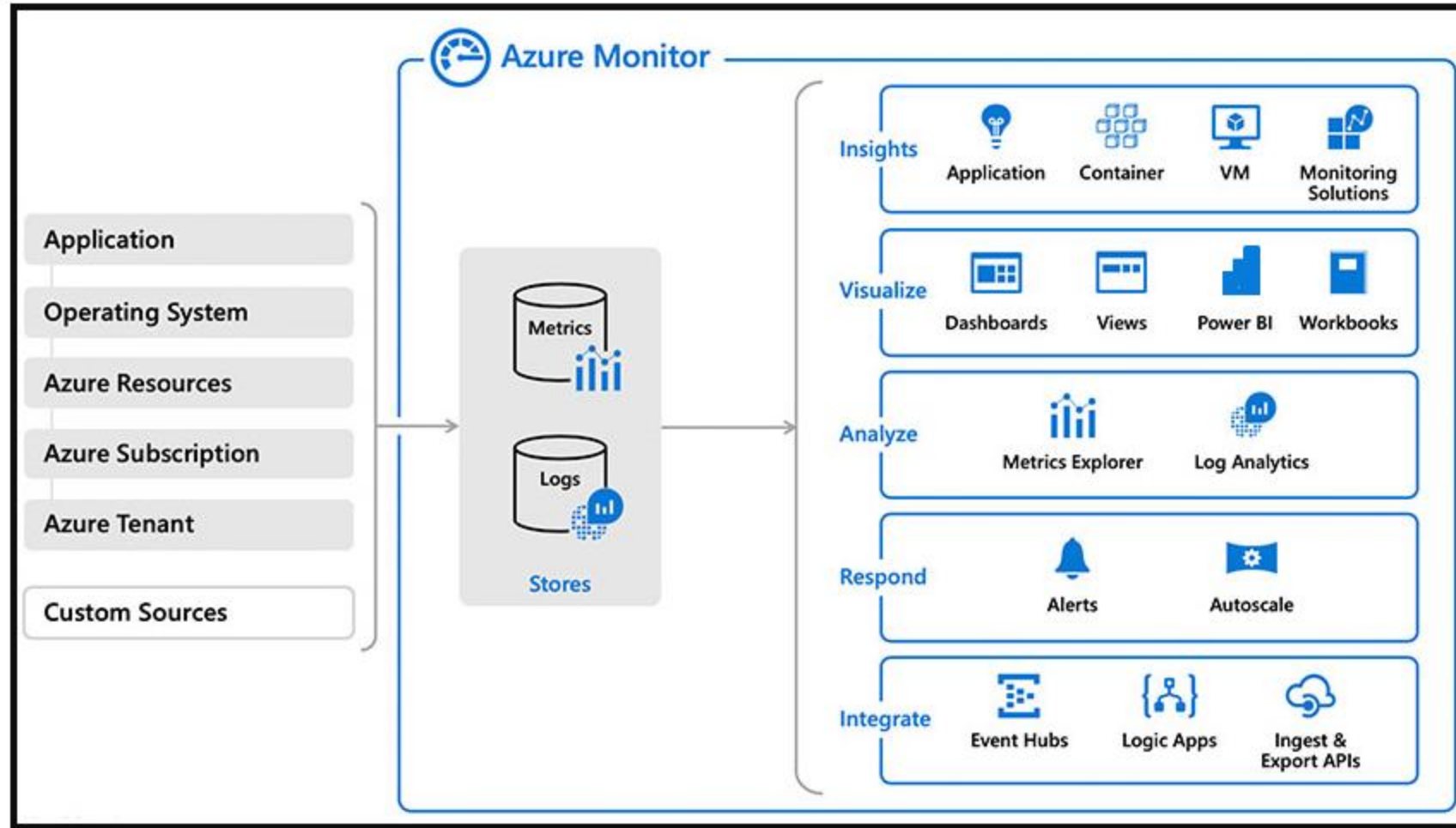
You have a hackathon to help you reinforce them!

And we will talk you through a couple of slides to help with your "hacking".

# The inner loop



# Azure Log Analytics example

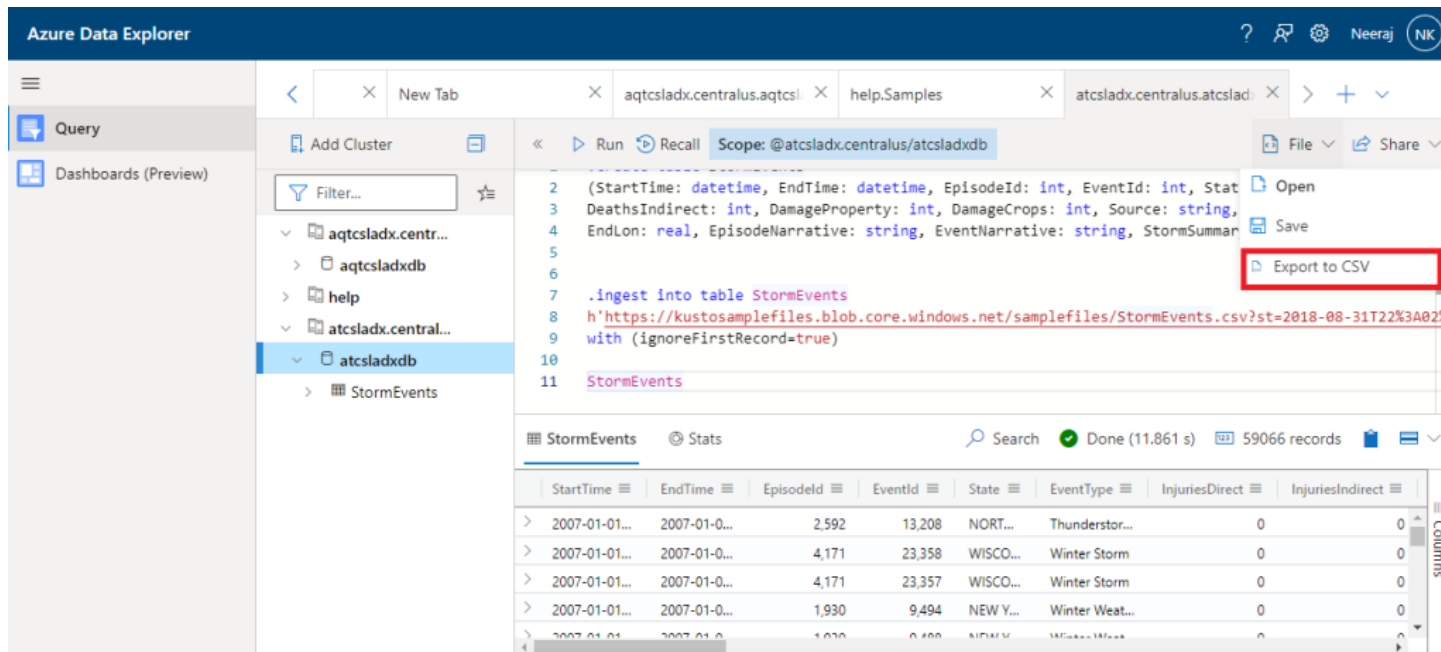


# Kusto Query Language (KQL)

KQL is the language used to query Log Analytics. You may want to explore it:

<https://learn.microsoft.com/en-us/kusto/query/>

1. Supports data exploration, queries and control commands
2. Supported within multiple Azure services



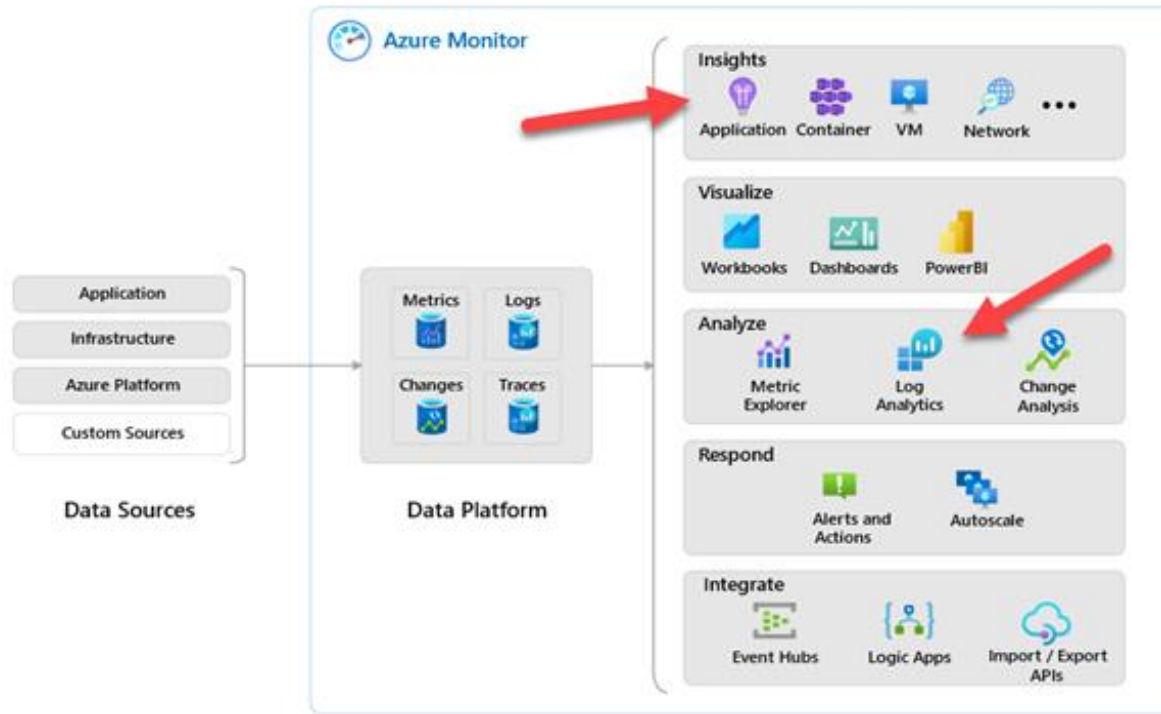
The screenshot displays the Azure Data Explorer web interface. On the left, a sidebar shows a tree view of clusters and databases, with 'atcsladxdb' and its table 'StormEvents' selected. The main panel is divided into a query editor and a results pane. The query editor contains a KQL query that ingests data from a CSV file into a table named 'StormEvents'. The results pane shows a table with columns: StartTime, EndTime, EpisodeId, EventId, State, EventType, InjuriesDirect, and InjuriesIndirect. The table contains several rows of data, including storm events from 2007. A red box highlights the 'Export to CSV' button in the top right corner of the query editor.

```
2 (StartTime: datetime, EndTime: datetime, EpisodeId: int, EventId: int, State: string,
3 DeathsIndirect: int, DamageProperty: int, DamageCrops: int, Source: string,
4 EndLon: real, EpisodeNarrative: string, EventNarrative: string, StormSummary: string)
5
6
7 .ingest into table StormEvents
8 h'https://kustosamplefiles.blob.core.windows.net/samplefiles/StormEvents.csv?st=2018-08-31T22%3A02%3A00Z'
9 with (ignoreFirstRecord=true)
10
11 StormEvents
```

StartTime	EndTime	EpisodeId	EventId	State	EventType	InjuriesDirect	InjuriesIndirect
2007-01-01...	2007-01-0...	2,592	13,208	NORT...	Thunderstor...	0	0
2007-01-01...	2007-01-0...	4,171	23,358	WISCO...	Winter Storm	0	0
2007-01-01...	2007-01-0...	4,171	23,357	WISCO...	Winter Storm	0	0
2007-01-01...	2007-01-0...	1,930	9,494	NEW Y...	Winter West...	0	0

# Azure Monitor

What used to be known as Application Insights and Log Analytics independent offerings - are now a part of Azure Monitor



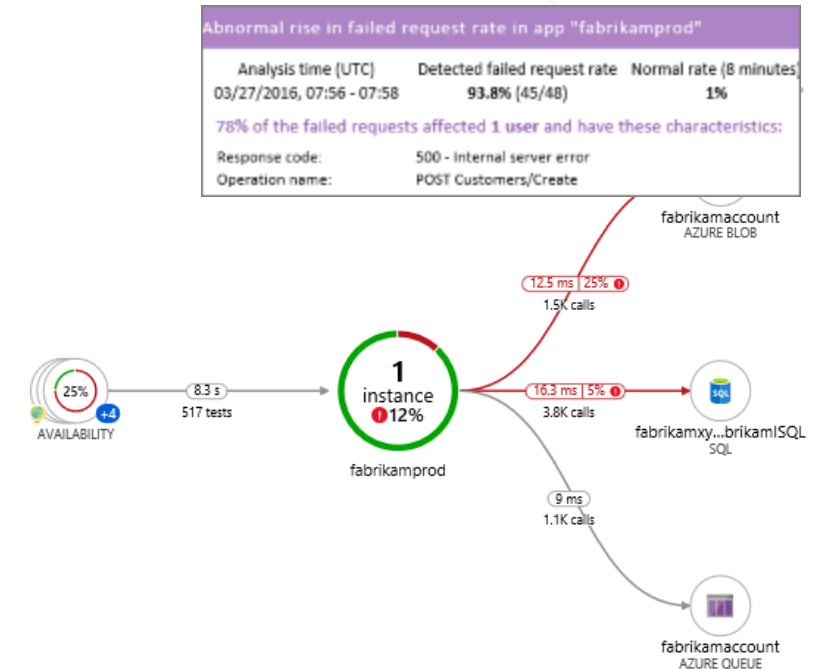
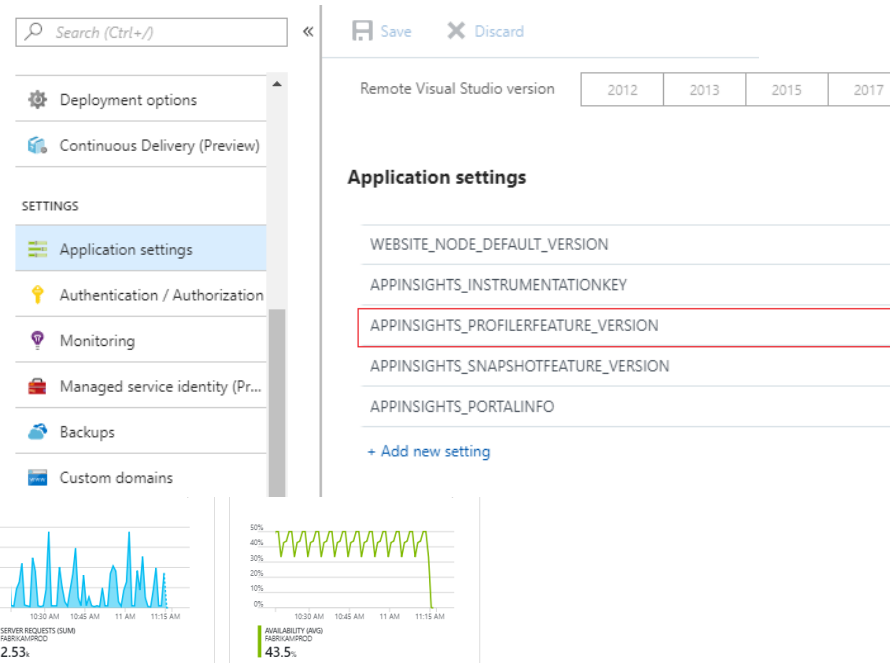
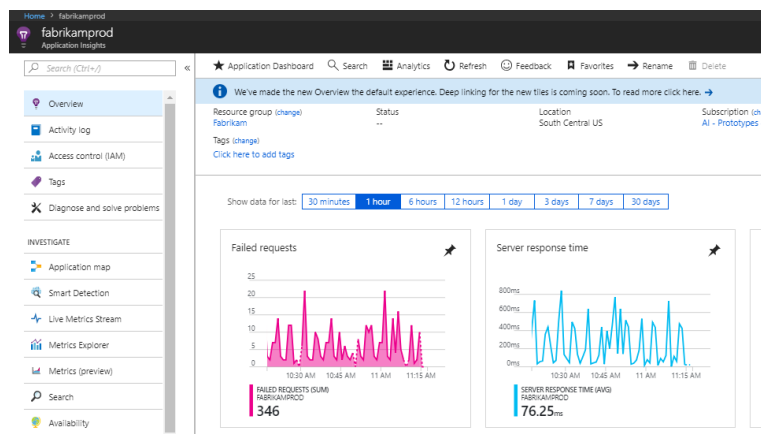
# Monitoring application performance

## Objectives:

- Deploy Azure App Service web apps
- Generate and monitor Azure web app application traffic
- Investigate Azure web app performance
- Track Azure web app usage
- Create Azure web app alerts

# Why monitor apps?

1. Monitor
2. Detect, diagnose
3. Build, measure, learn



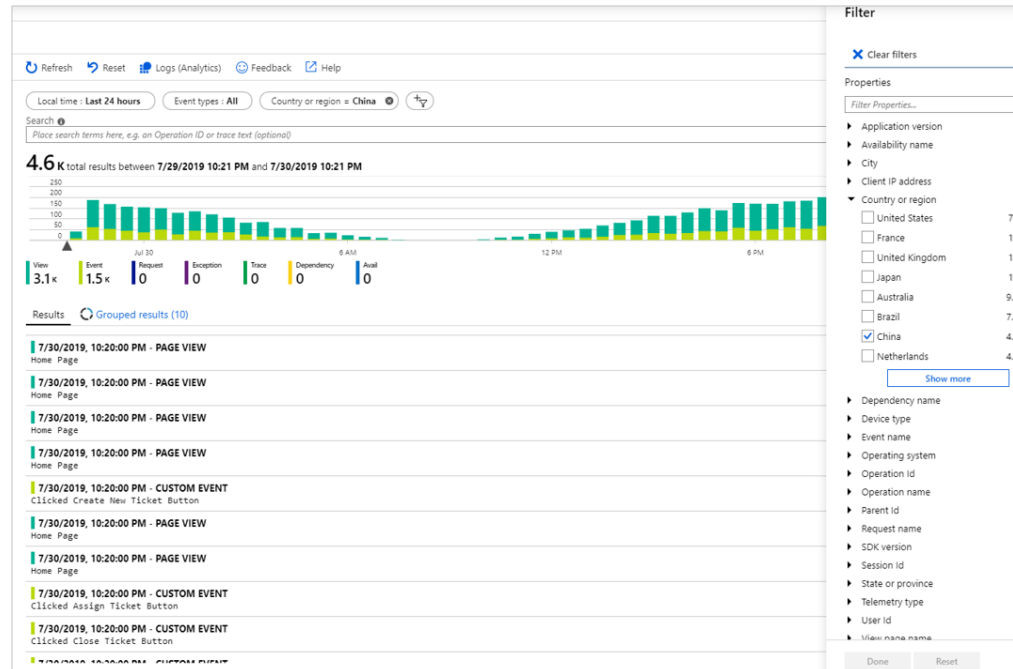
Abnormal rise in failed request rate in app "fabrikamprod"		
Analysis time (UTC)	Detected failed request rate	Normal rate (8 minutes)
03/27/2016, 07:56 - 07:58	93.8% (45/48)	1%
78% of the failed requests affected 1 user and have these characteristics:		
Response code:	500 - Internal server error	
Operation name:	POST Customers/Create	





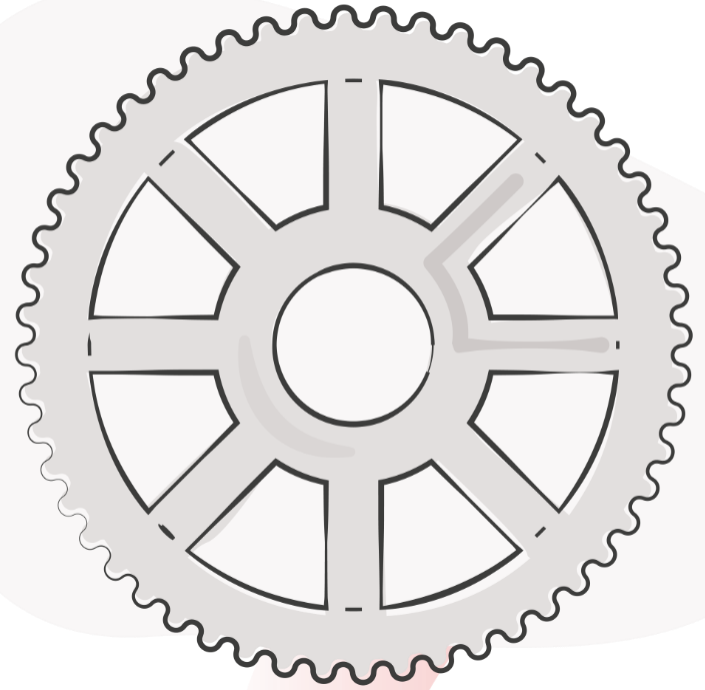
# App Center Diagnostics

1. App Center Diagnostics is a cloud service that helps developers monitor the health of an application, delivering the data needed to understand what happens when an app fails during testing or in the wild
2. In the App Center Diagnostics UI, you can attach, view and download one binary and one test attachment to your crash reports
3. Track events leading up to a crash to capture useful information about the state of your app

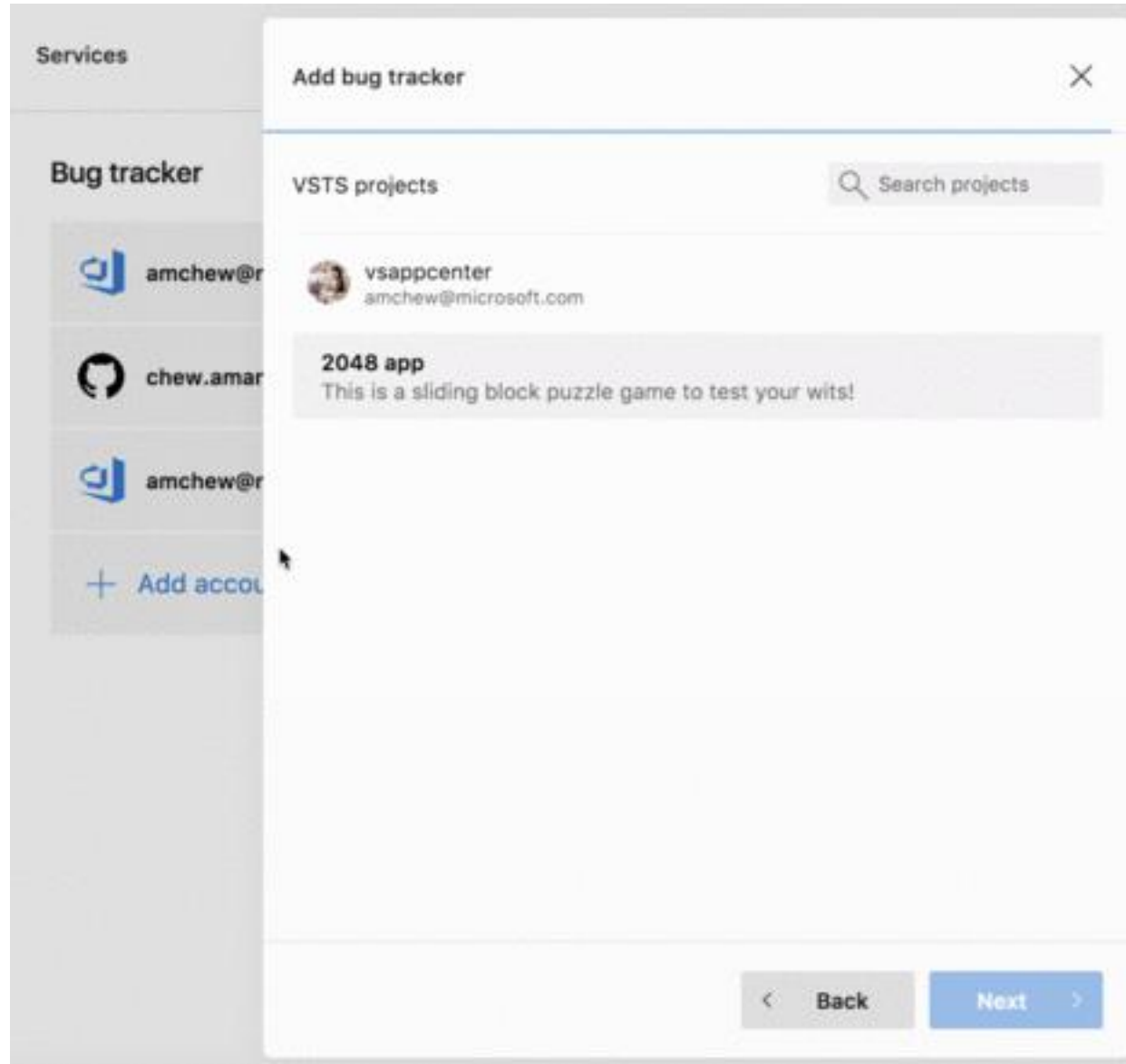


# Configure alerts

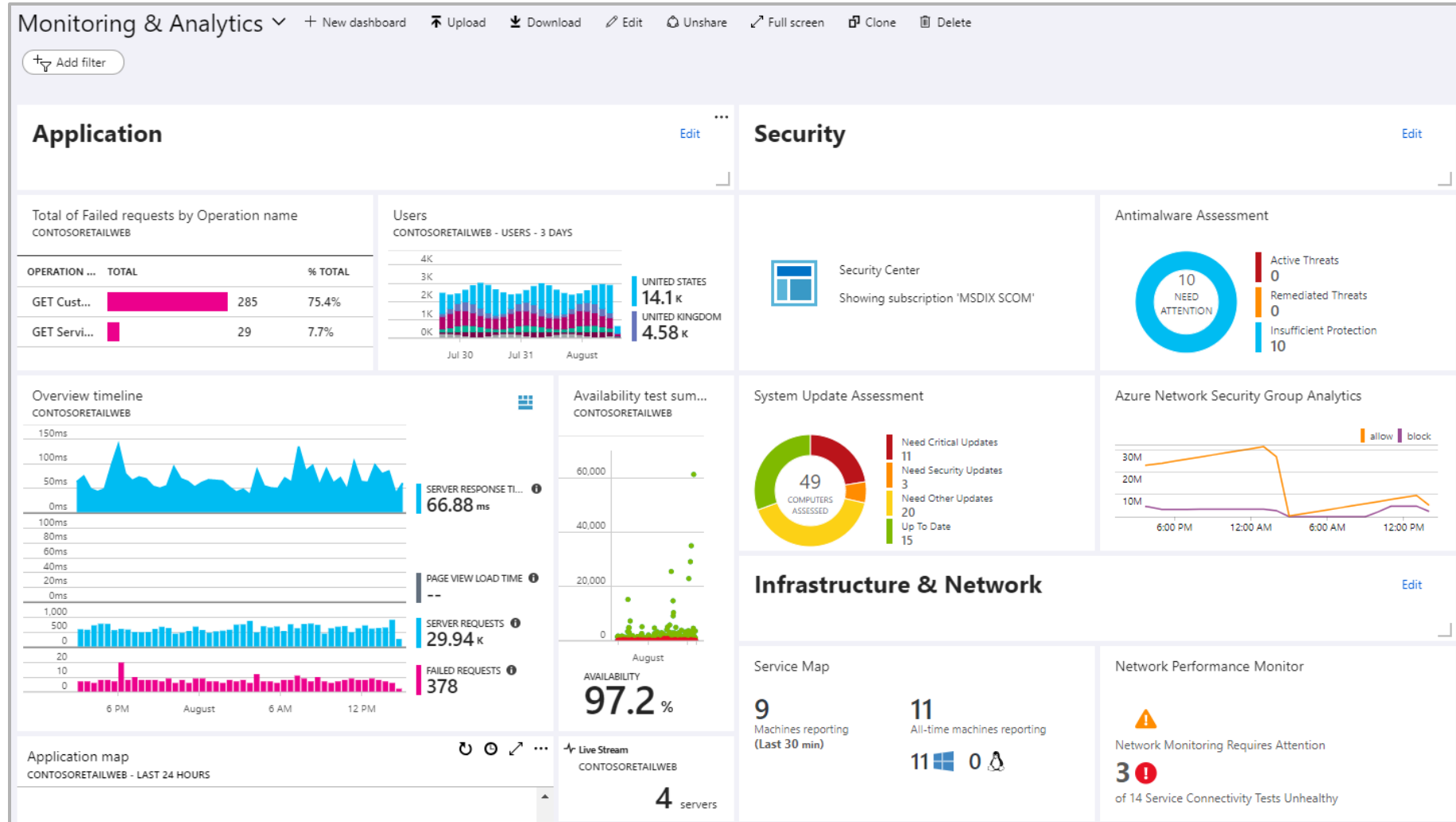
1. Log into App Center and select your app
2. In the left menu, navigate to **Settings**
3. Click on **Email Notifications**
4. Select when a new crash group is created



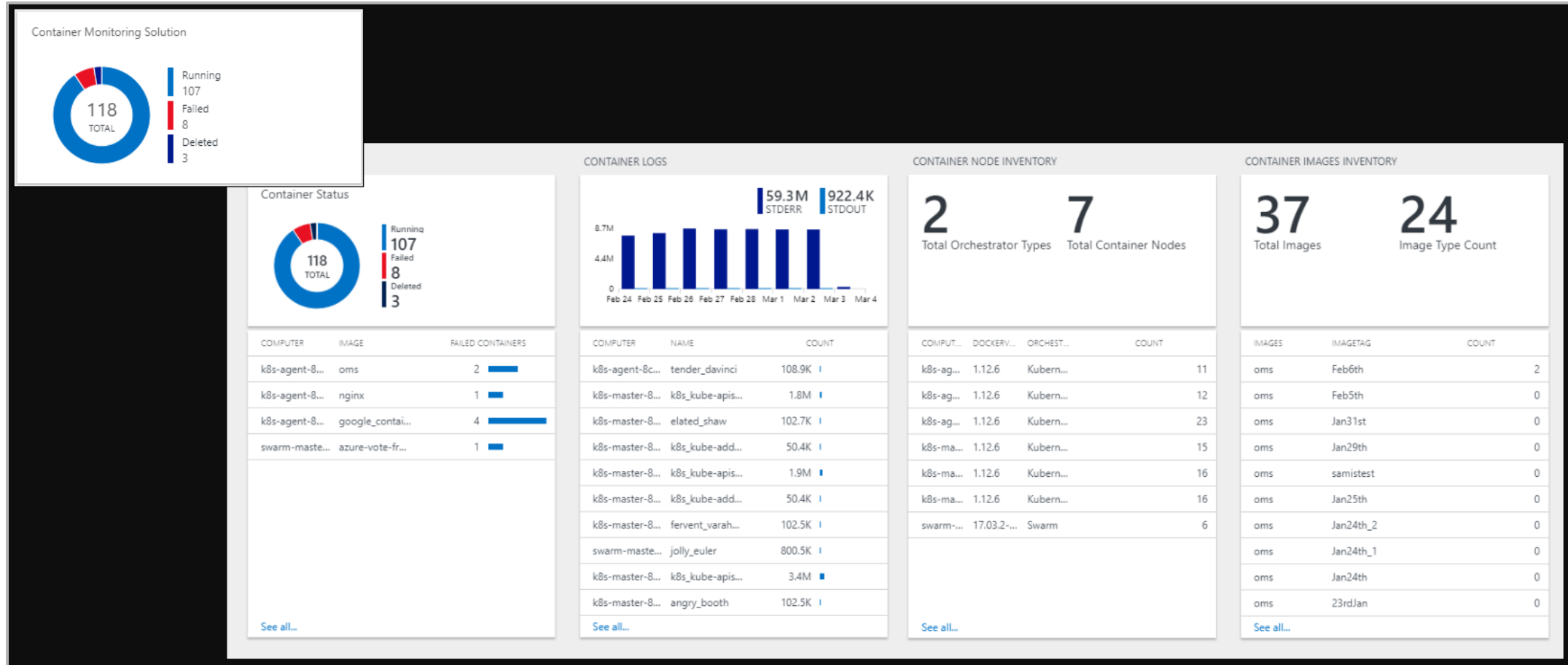
# Create a bug tracker



# Azure Dashboards



# View Designer in Azure Monitor



# Azure Monitor workbooks

## Analysis of Page Views

Page views correspond to user activity in your app. Understanding how your users interact with your pages will give you good insights into what is working in your app and what aspects need improvements.

This report will help

- Usage
- Time spent on p
- Time to first int
- Exit rates

If your telemetry do

Pages: Home Page, D

OptimizeCalculations

### Usage

This section helps you understand how

Page N...	Unique Users	As % of
Overall	8813	
Home P...	8811	
Create	8792	
Details	8780	

- The As % of App Users/Sessions/V

### Time Spent on Page

This report helps you understand the time spent by customers in your pages. Longer time spent pages usually indicates good engagement and is generally the desired behavior.

IgnoreDurationsOver: 3600s

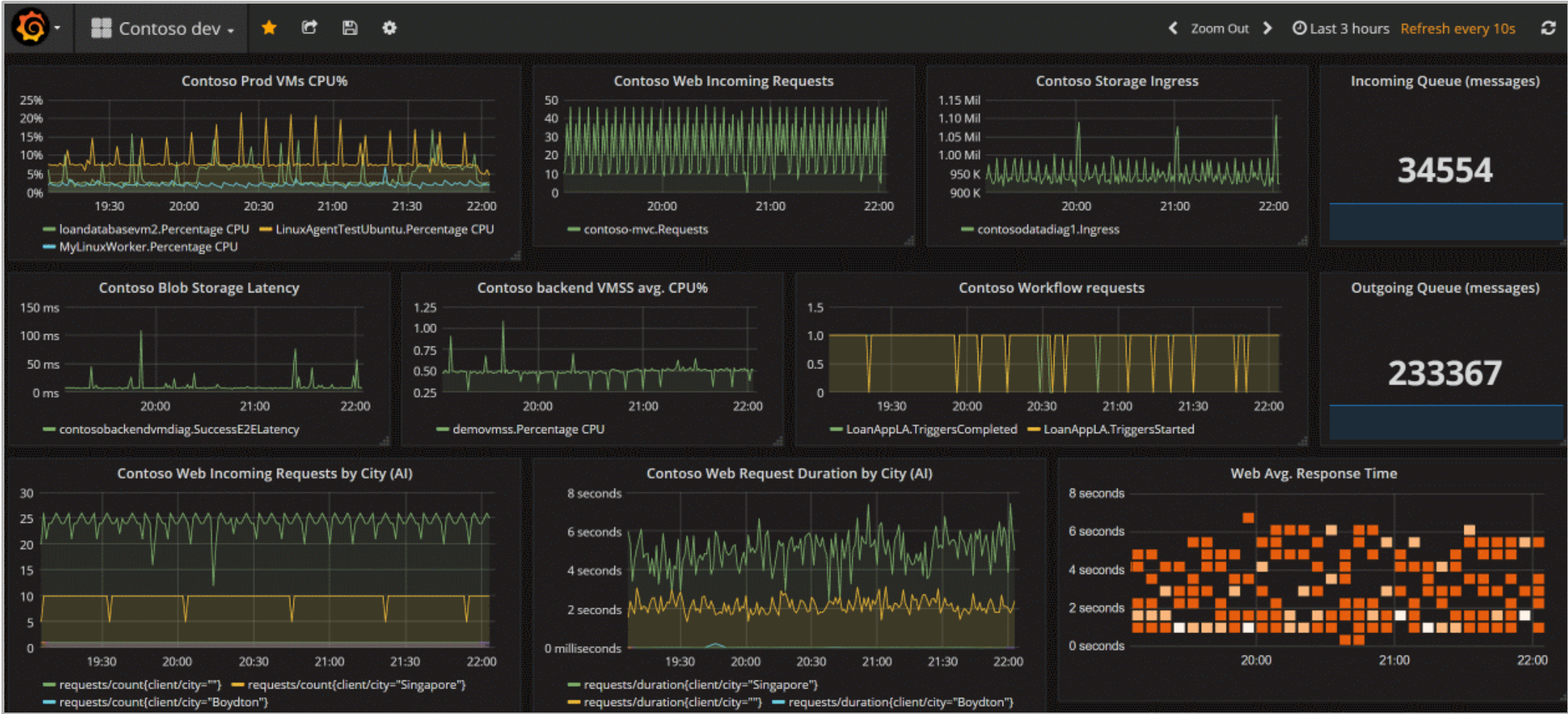
Page Name	Sampled Page ...	Median (second...	75th Percentile (seconds)	90th Percentile ...	Mean (seconds)
Overall	107150	149.7	375	750.5	294.3
Create	29522	174.4	540	896.5	348.1
Details	33086	169.2	480	900	351.5
Home Page	44542	105.6	300	500	216.2

- The calculations may use sampling based on the `optimizeCalculationsFor` parameter.
- Time Spent on Page does not consider exit pages (last page of the session) in this calculations. The `Sampled Page Views` column may be fewer than the sampling count of 100000 because of this.





# Grafana



# Build your own custom application

## Advantages:

- Complete flexibility
- Combine metrics and log data

## Disadvantages:

- Significant engineering required





**Thank you**

**Do you have any questions,  
comments, or feedback?**

