Microsoft

# AZ-305

# Designing Microsoft Azure Infrastructure Solutions

# AZ-305 Agenda

Module 01 Design a governance solution

Module 02 Design a compute solution

Module 03 Design a non-relational data storage solution

Module 04 Design a data storage solution for relational data

Module 05 Design a data integration solution

Module 06 Design an application architecture solution

Module 07 Design Authentication and Authorization Solutions

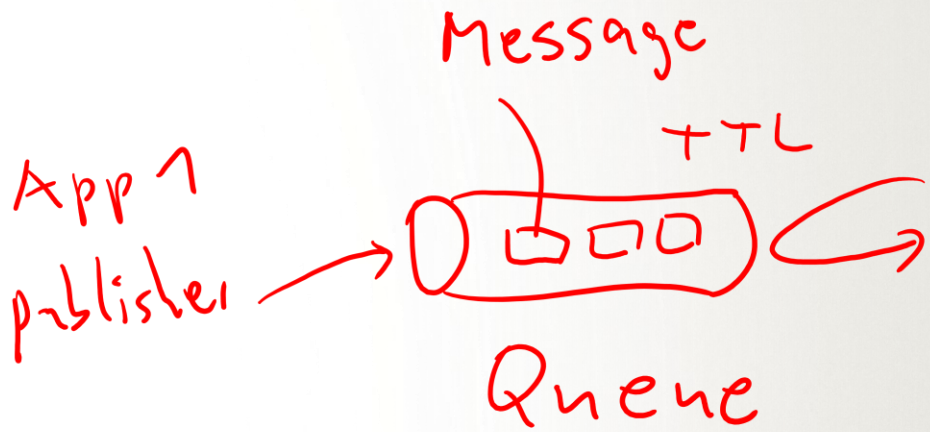
Module 08 Design a solution to log and monitor Azure resources

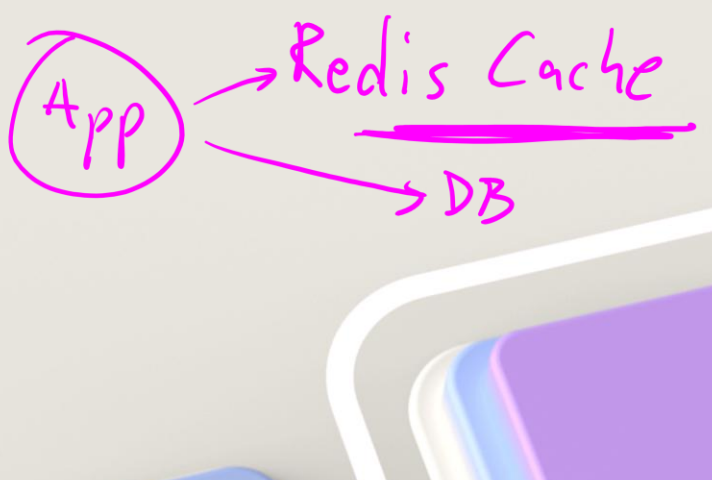
Module 09 Design a network infrastructure solution

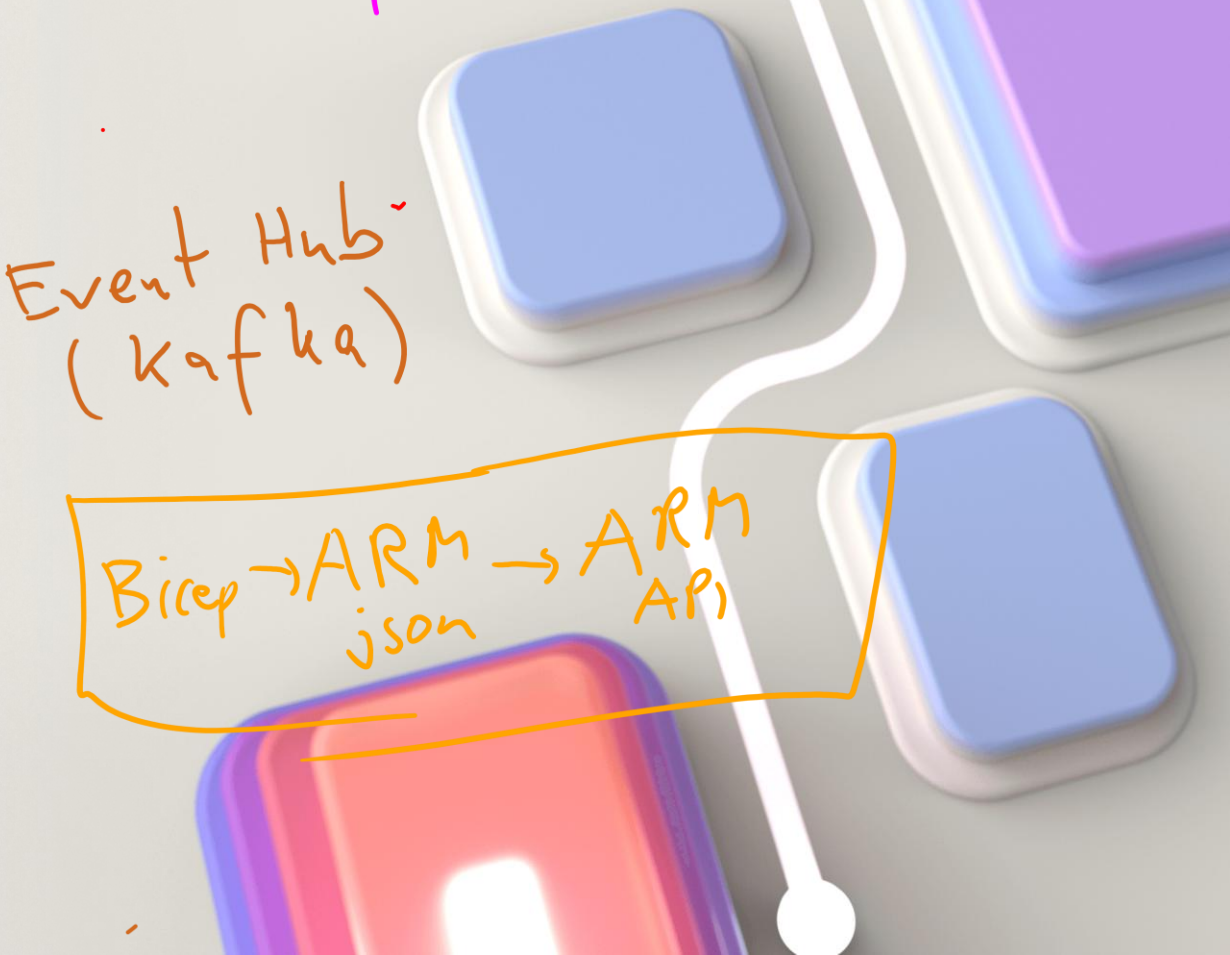
Module 10 Design a business continuity solution

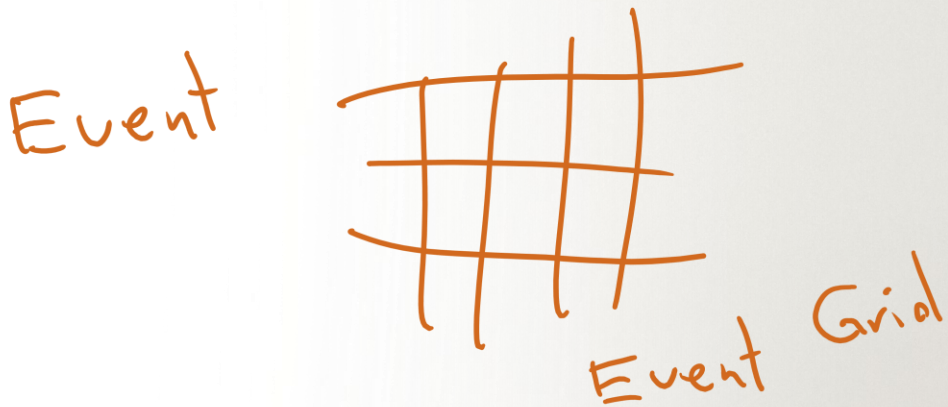
Module 11 Design a migration solution

# Design an application architecture solution



# Learning Objectives

- Describe message and event scenarios
- Design a messaging solution
- Design an event solution (Event Hub and Event Grid)
- Design an application optimization solution
- Design application lifecycle
- Case study
- Learning recap

AZ-305: Design infrastructure solutions (30-35%)

Design an Application Architecture

- Recommend a messaging architecture
- Recommend an event-driven architecture
- Recommend a solution for API integration
- Recommend a caching solution for applications
- Recommend an application configuration management solution
- Recommend an automated deployment solution for applications

# Describe message and event scenarios

# Determine message and event scenarios

## Does the sending component expect the communication to be processed in a specific way?

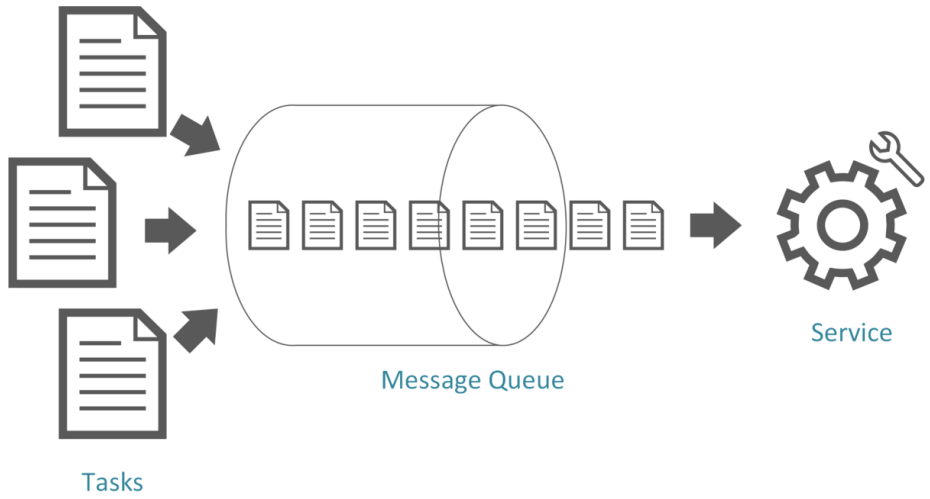
| Action | Description | When to use |
| --- | --- | --- |
| Event | • Light weight<br>• Includes a publisher and a subscriber | Used for broadcasts and are often ephemeral. Ephemeral means the communication might not be handled by any receiver if none is currently subscribing. |
| Message<br>ServiceBus | • Contains raw data, produced by one component, that will be consumed by another component.<br>• Contains the data itself, not just a reference to that data. ? | Used where the distributed application requires a guarantee that the communication will be processed. |

# Design a messaging solution

# Design for Azure Queue storage

## Azure Storage Queue is a service for storing large number of messages.

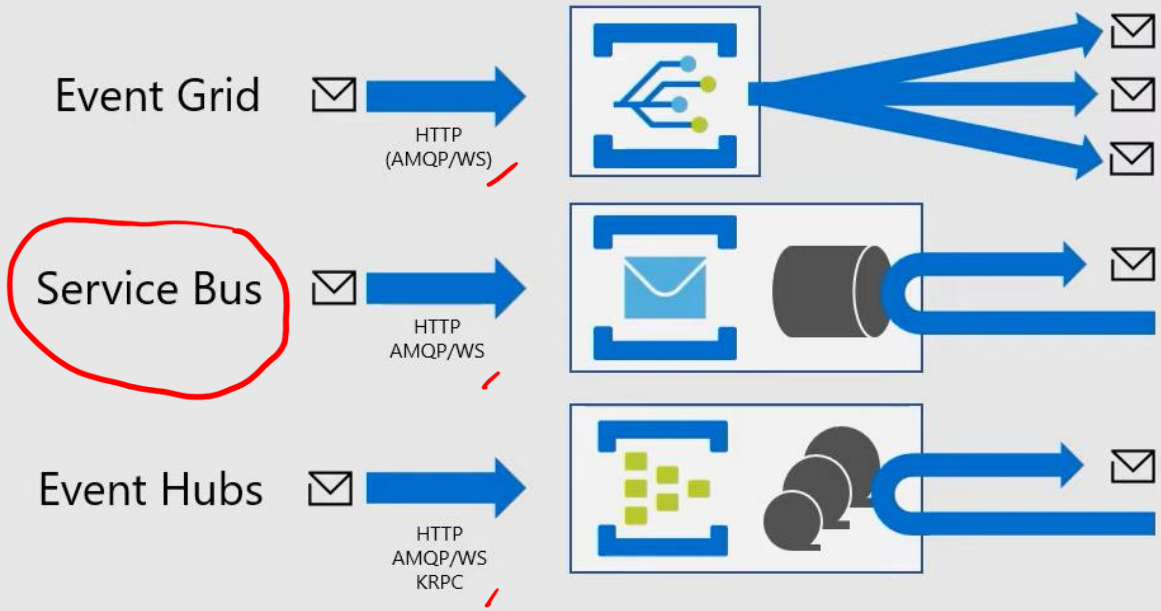
- Accessed with authenticated calls using HTTP or HTTPS
- Messages can be up to 64 KB in size
- May contain millions of messages, up to the total capacity limit of a storage account




- Create a backlog of work to process asynchronously
- Example: customer placing orders online added to the queue and processed

# Azure Eventing and Messaging Core Services

Message Oriented Services




Push-style distribution of discrete events to serverless workloads

Pull-style, queue-based transfer of jobs and control via message queues and topics

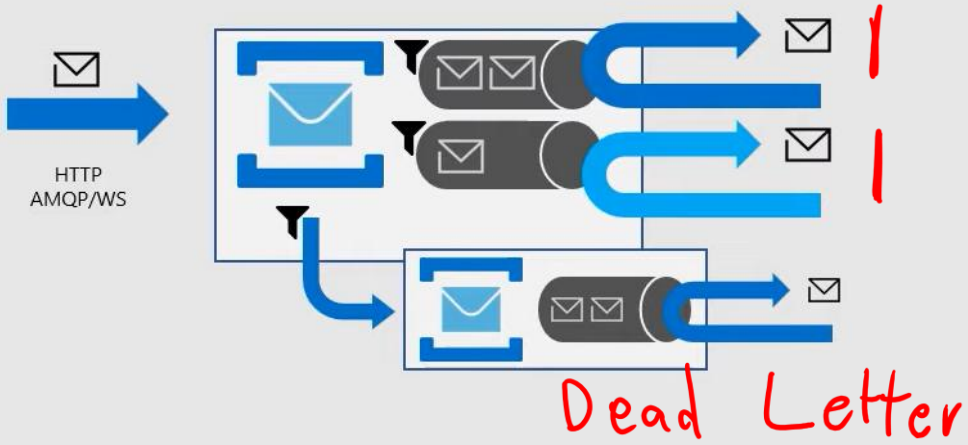
Partitioned, high-volume, tape-drive-style sequential recording and unlimited, pull-style re-reads of event streams.

# Service Bus Architectural Patterns

Queues


- Assignment of work with load-aware balancing
- Load-leveling for "spiky" workload traffic shapes
- Transactional, once-and-only-once processing
- Multiplex handling of in-order message sequences
- Deduplication, deferral, and "poison" handling

Topics




- All of the above, plus:
- Copies to 100s of concurrent subscribers
- Filter rules and message markup
- Message routing

**Service Bus is a "swiss army knife" for messaging-driven workloads.**

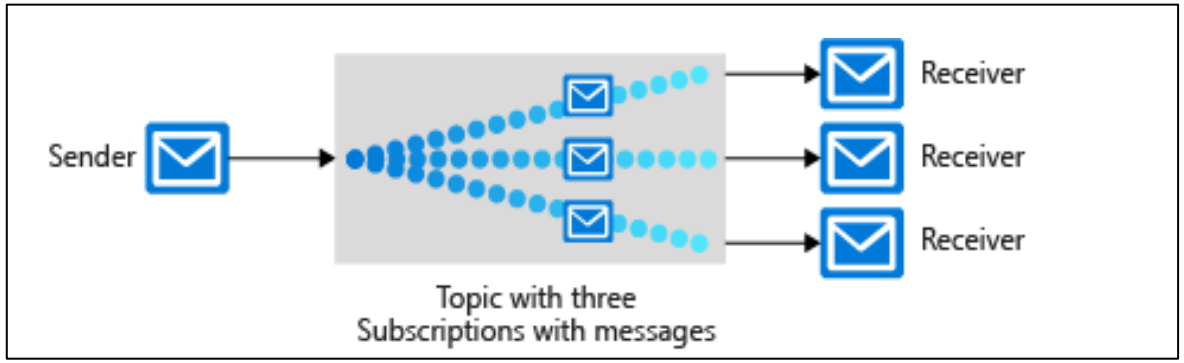
# Design for Service Bus queues and topics

## Service Bus decouples applications and services from each other.

### Service bus queues


- Built on top of a dedicated messaging infrastructure
- Holds messages until the target is ready to receive them – different from queues

### Service bus publish-subscribe topics




- Like bus queues but with multiple subscribers
- When a message is sent to a topic, multiple components can be triggered to perform a task

# Compare messaging solutions

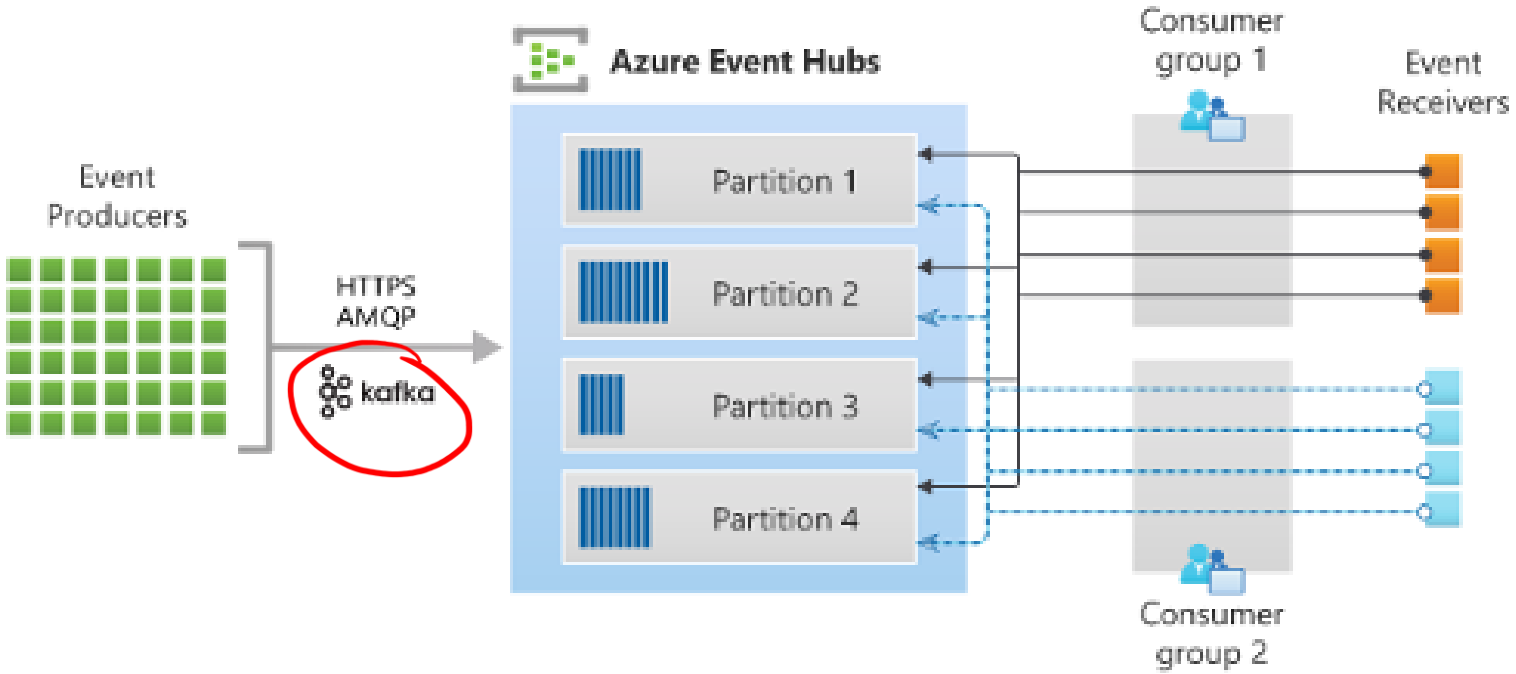
| Solution | Usage cases | SLA |
| --- | --- | --- |
| Queue storage | • A simple queue to organize messages.<br>• Queue to exceed 80 GB in size.<br>• To track progress for processing a message inside of the queue. | Based on storage tier |
| Service bus queues | • A first-in-first-out guarantee.<br>• At-Least-Once message processing (PeekLock receive mode)<br>• At-Most-Once message processing (ReceiveAndDelete receive mode)<br>• Can group operations into transactions<br>• Receive messages without polling the queue.<br>• Publish and consume batches of messages. | 99.9% |
| Service bus topics | • Multiple receivers to handle each message.<br>• Multiple destinations for a single message. | 99.9% |

# Design an event solution

# Design an Event Hub messaging solution

## Azure Event Hubs is a fully managed, real time data ingestion service

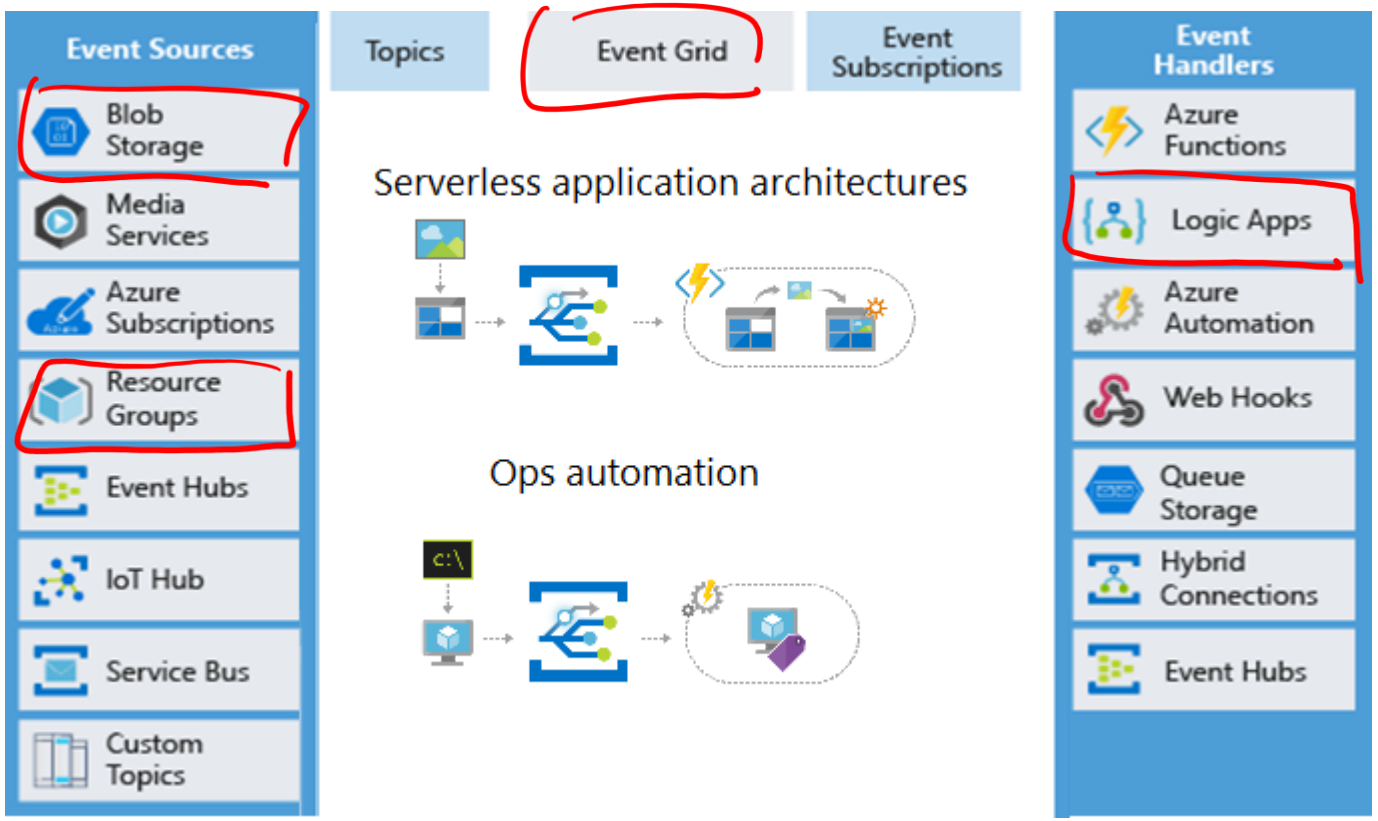
- Orders events by when they are received - by time offsets
- Uses a pull model allowing multiple reads from consumers
- Scaling is controlled by how many throughput units or processing units you purchase
- Receiving real-time streaming data



# Design an event-driven solution

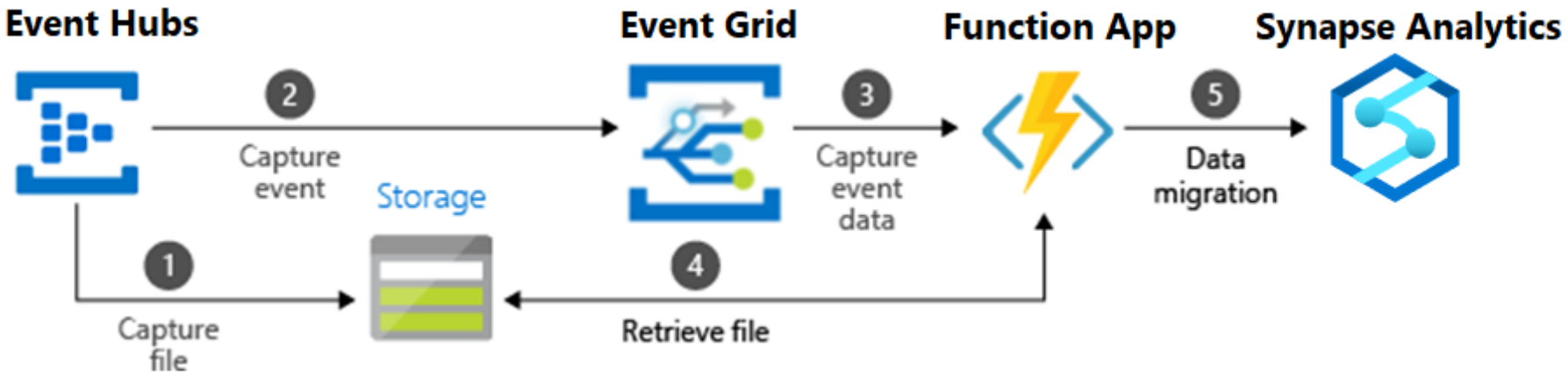
## Azure Event Grid is a routing service connecting data sources with event handlers.

- Events sources include Azure resources or custom topics (you create)
- Event handlers react to an event
- Useful for serverless applications and operations automation
- Uses a pay-per-operation or pay-per-use pricing models



# Comparison of message and event solutions

## Consider combining several solutions




| Service | Purpose | Type | When to use |
|---|---|---|---|
| Event Grid | Reactive programming | Event distribution (discrete) | React to status changes |
| Event Hubs | Big data pipeline | Event streaming (series) | Telemetry and distributed data streaming |
| Service Bus | High-value enterprise messaging | Message | Order processing and financial transactions |
| Storage Queues | Large-volume messaging queues | Message | Cost-effective, simple messaging mechanism |

# Design an IoT Hub solution

PaaS

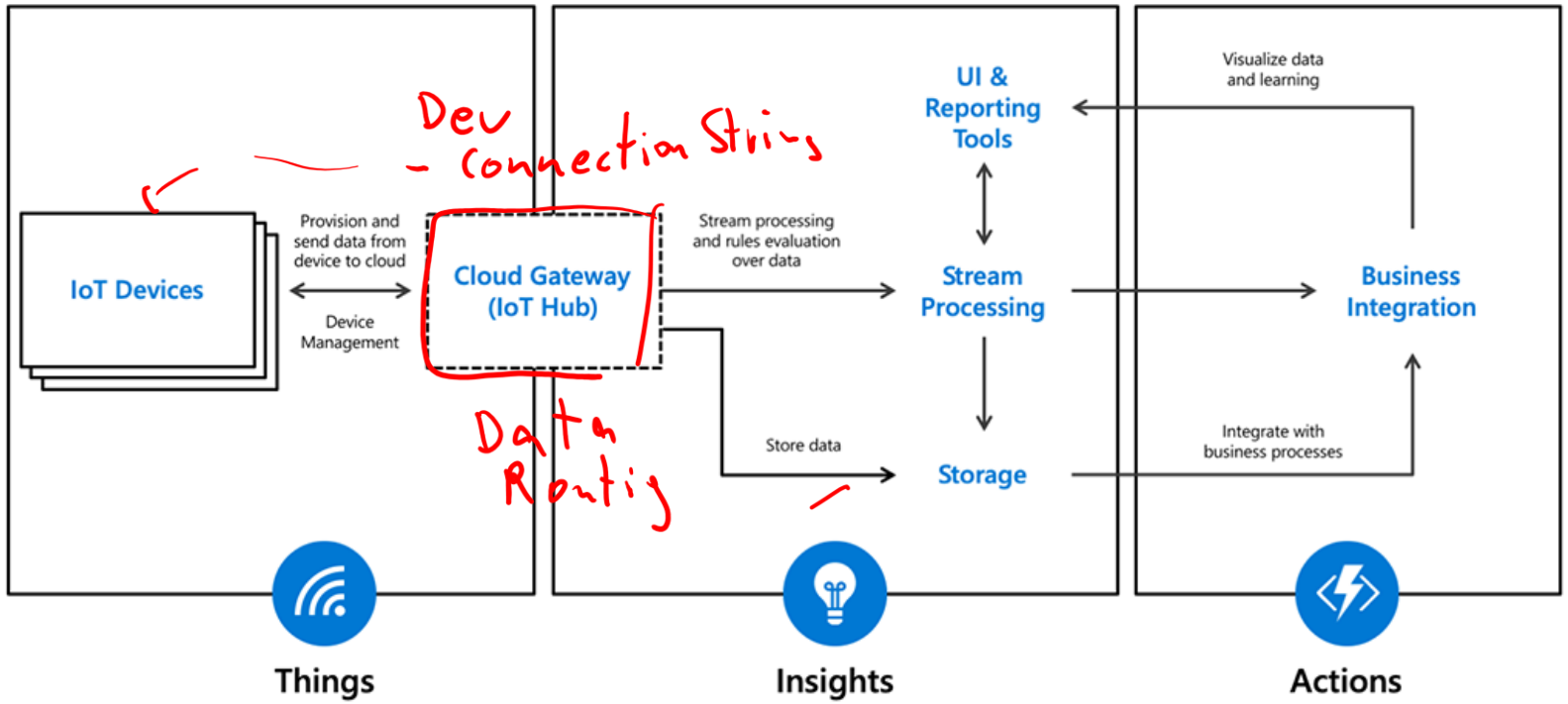
## Central message hub for IoT applications and its attached devices.

### When to use IoT Hub?

- Application complexity
- Data throughput
- Securing solution end to end allowing for per-device authentication
- Bi-directional communication

### Capabilities over Event Hub:

- Per-device identity
- File upload from devices
- Device provisioning service

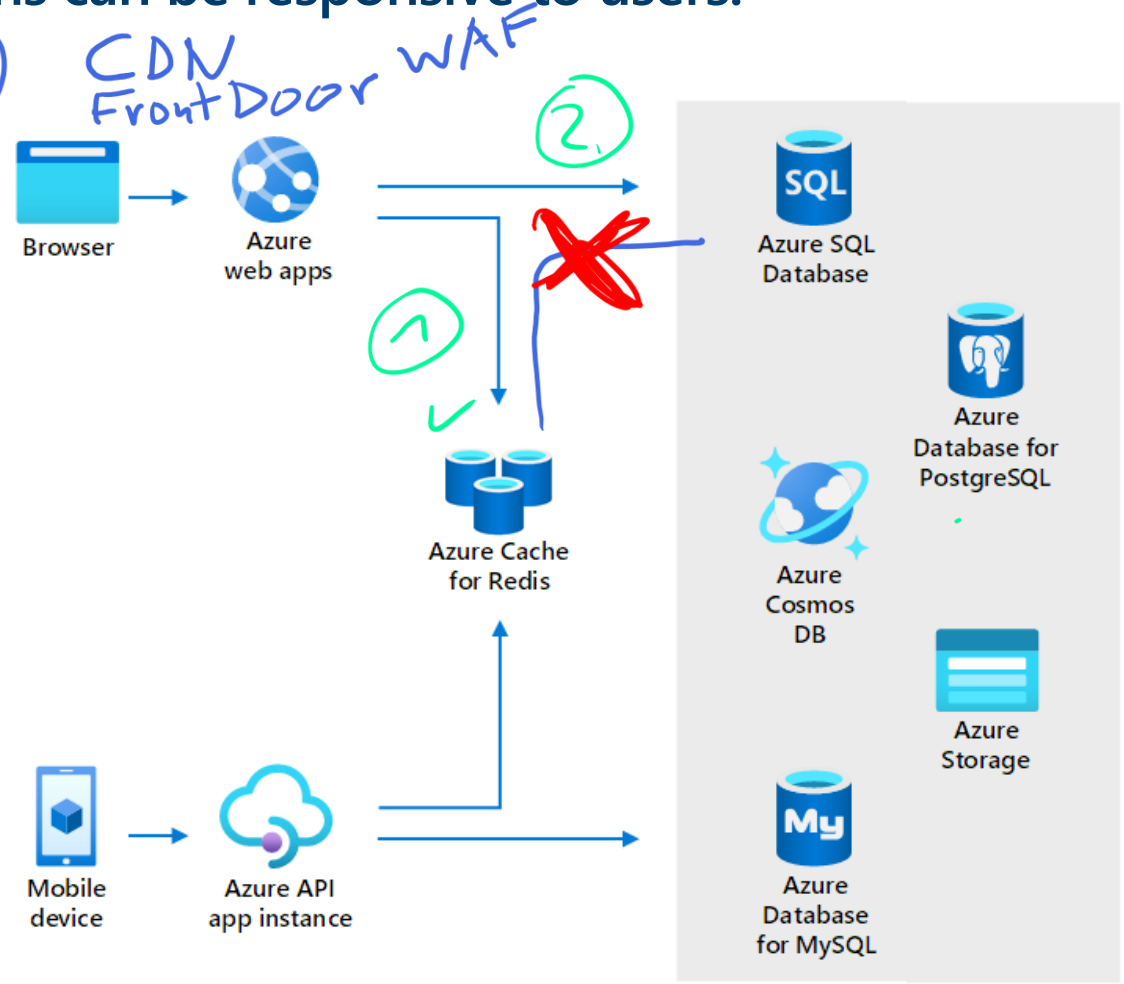


# Design an application optimization solution

# When to use Azure Cache for Redis

## Store frequently accessed data so that applications can be responsive to users.

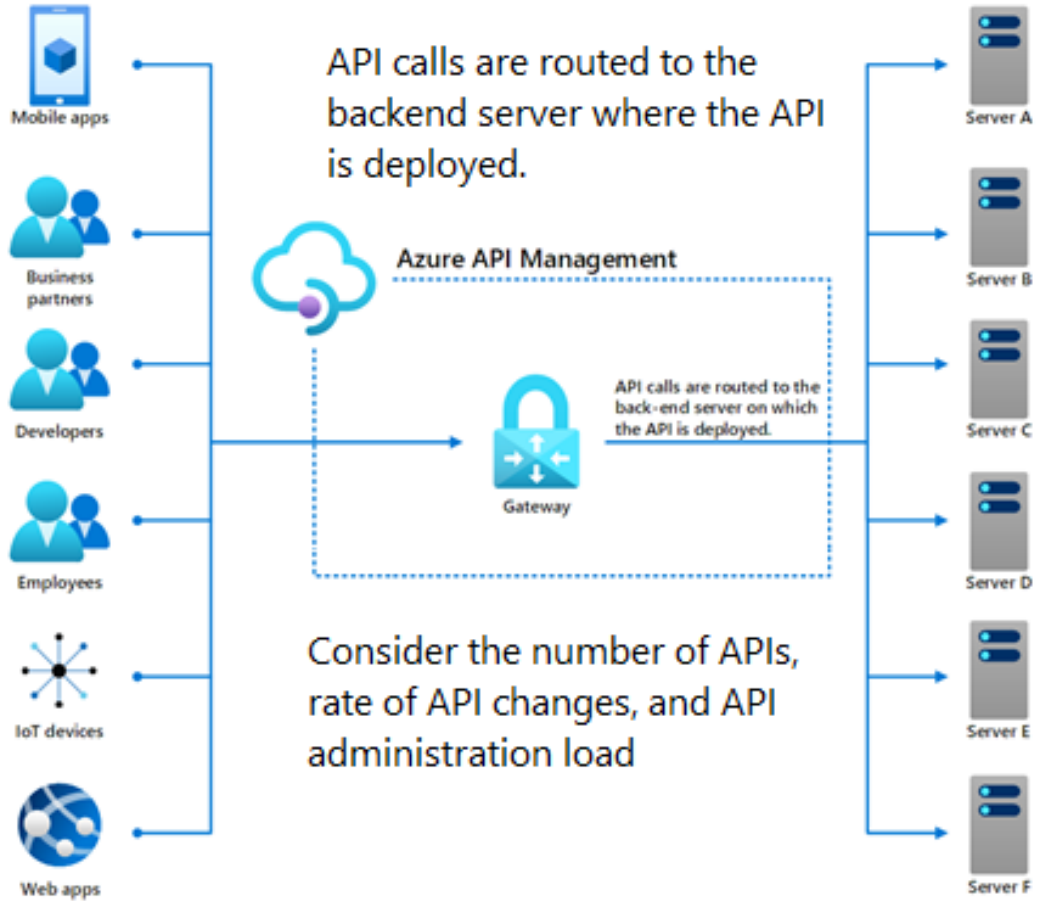
- Key scenarios - data cache, content cache, session store, job and message queuing, and distributed transactions
- Fully managed solution
- High availability - responds automatically to both anticipated and unanticipated changes in demand
- Same performance and scaling benefits throughout the world – network isolation, data encryption in transit



# Design an Azure API management solution

## Publish, secure, maintain, and analyze all your company's APIs.

- Bring multiple APIs under a single administrative umbrella – centralized management
- Manage permissions and access
- Ensure compliance across API
- Standardize API specs
- Protect the APIs from malicious usage



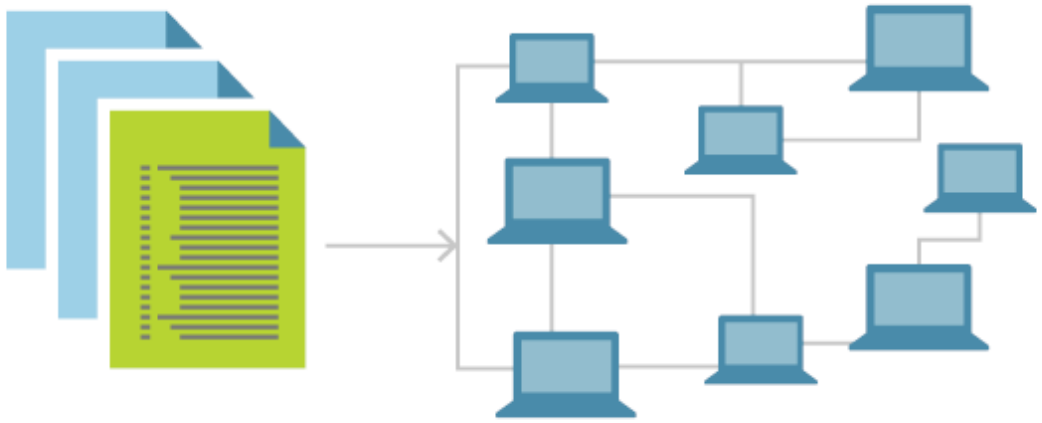
# Design an application lifecycle

# What is Infrastructure as Code?

Azure Bicep → ARM Template json

**Infrastructure as Code (IaC) is the process of automating your infrastructure provisioning.**

- The IaC model generates the same environment every time it is applied
- Solves the problem of environmental drift
- Enables teams to test applications in production-like environments early
- Where possible, uses declarative definition files



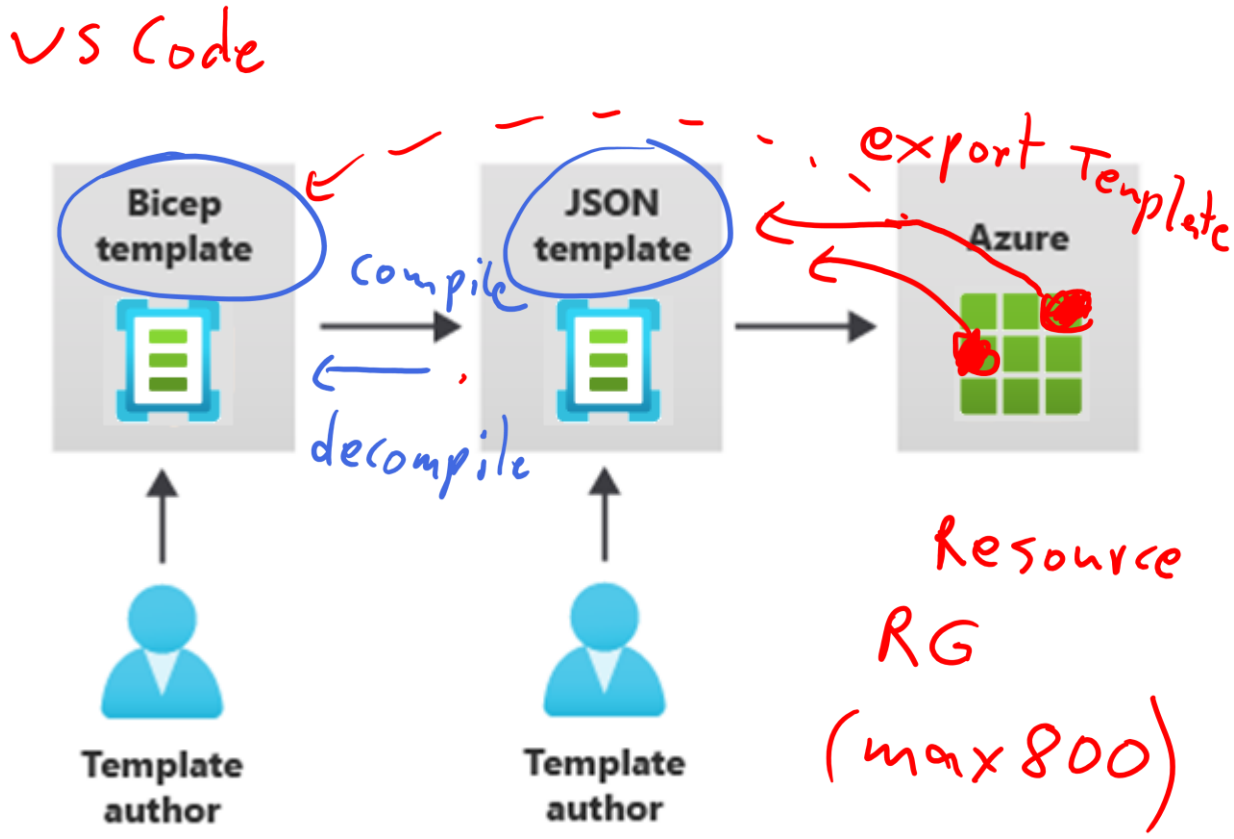
Multi Terraform

AWS Cloud Formation

# Provision resources with Infrastructure as Code

## Azure supports IaC with Azure Resource Manager and third-party platforms.

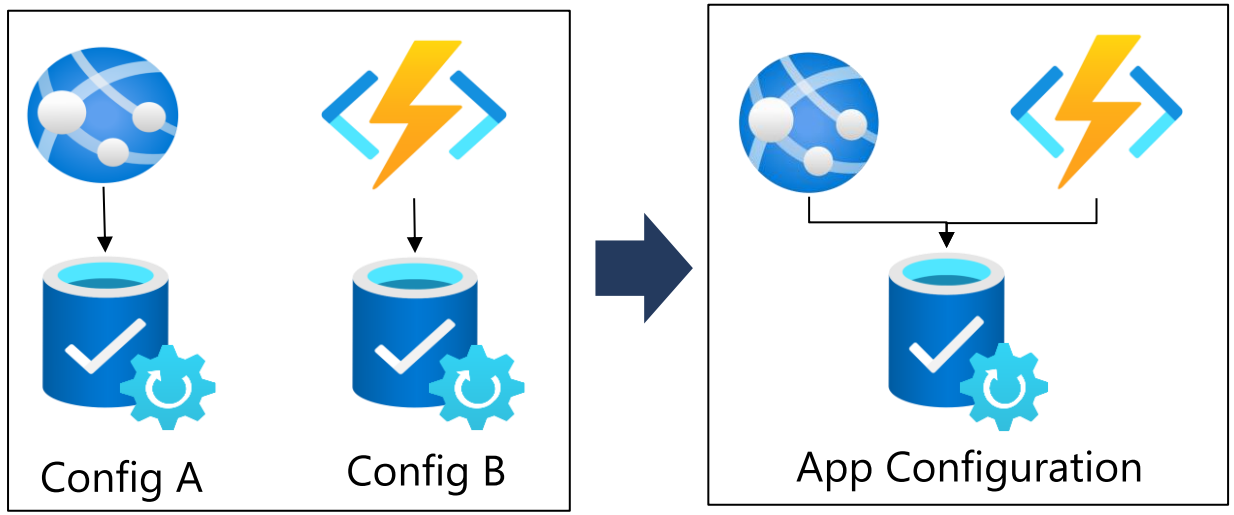
- Azure Resource Manager templates – Bicep, JSON
- Azure Automation
- Azure DevOps services
- GitHub actions
- Terraform
- Jenkins



# Design an Azure App Configuration solution

## Azure App Configuration centrally manages application settings and feature flags.

- Flexible key representations and mappings
- Point-in-time replay of settings
- Dedicated UI for feature flag management
- Comparison of two sets of configurations on custom-defined dimensions
- Enhanced security through Azure-managed identities and encryption



AZ-305 Tag 3

# Case study and review

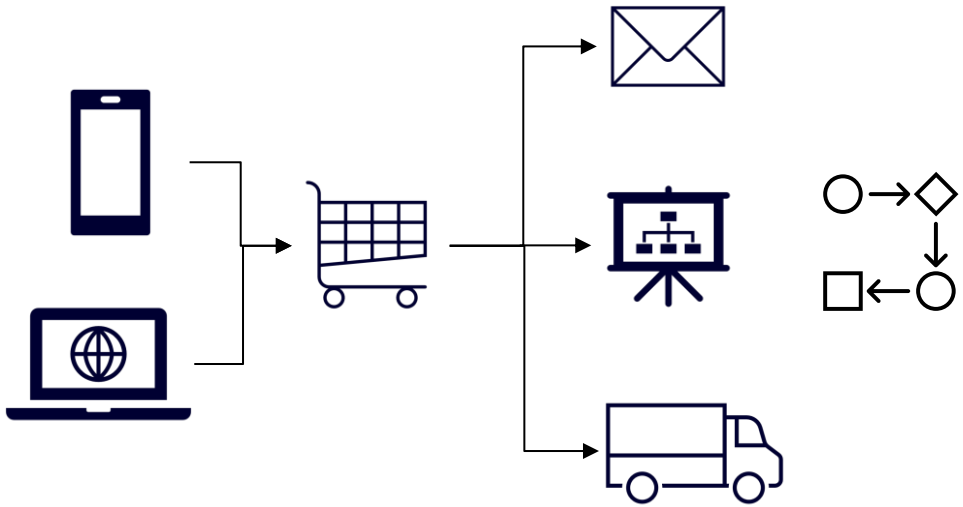
Guten Morgen!

# Case Study – Application architecture

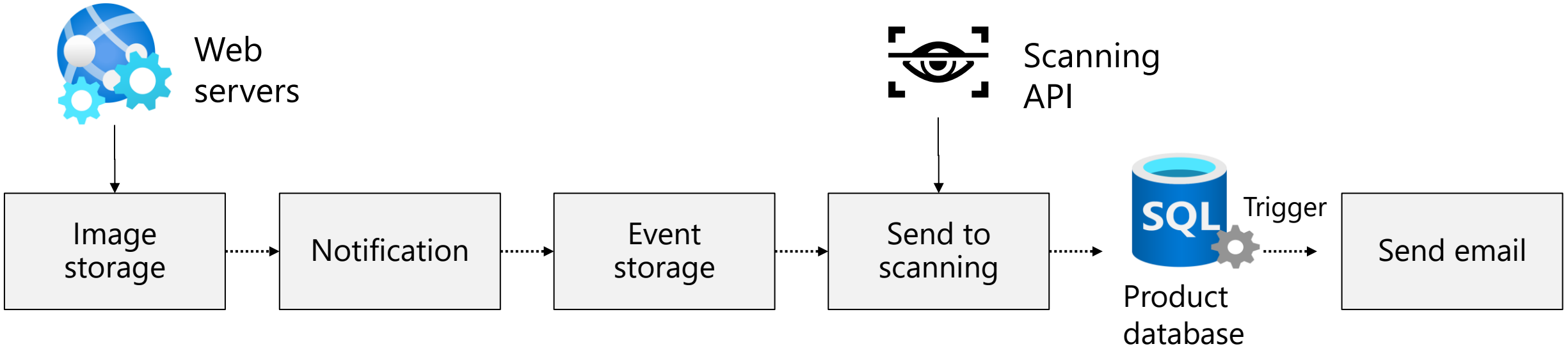
Tailwind

## A new product catalog design

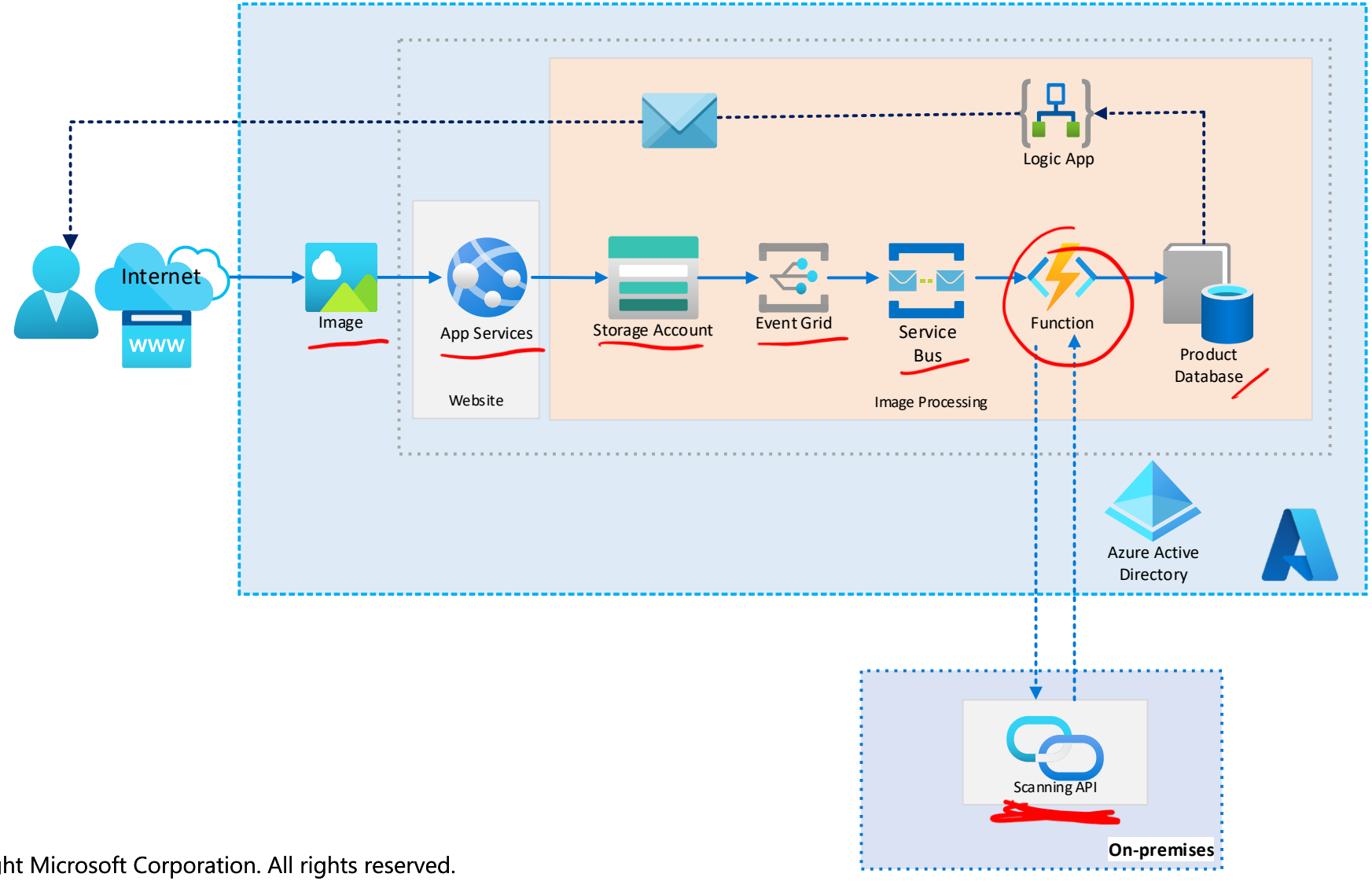
- New product catalog, ordering process, and shopping cart
- Services will rely on a combination of relational and non-relational data
- It is critical that the service hosting the application supports rapid autoscaling and high availability



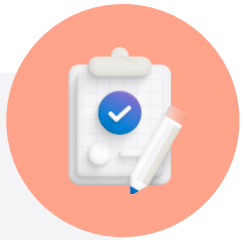
# Instructor case study discussion




# Instructor Solution Diagram



# Learning recap – application architecture solutions

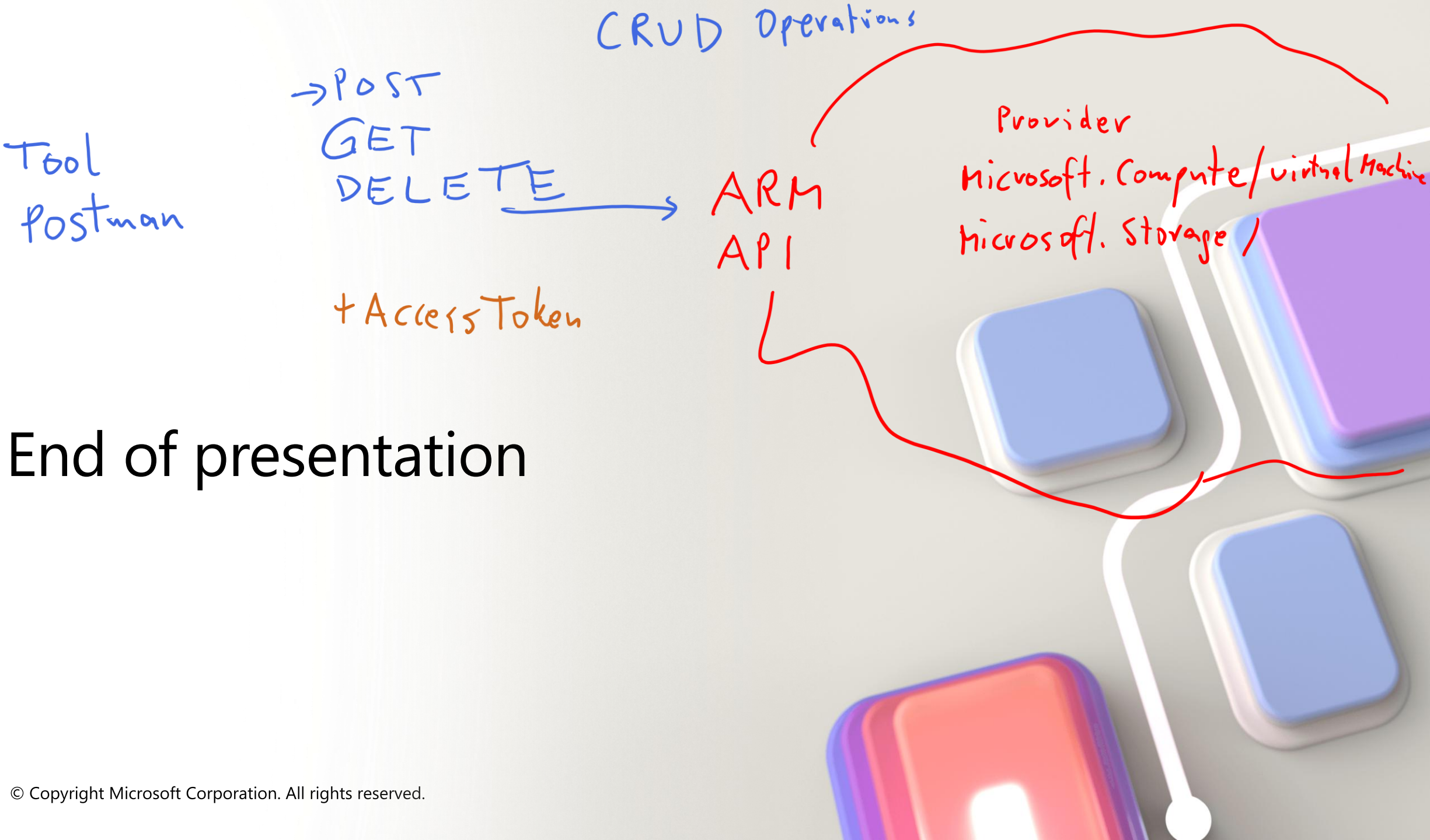


Check your knowledge questions and review

- Choose a messaging model in Azure to loosely connect your services
- Introduction to Event Hubs
- Deploy Azure infrastructure by using JSON ARM templates
- Introduction to infrastructure as code using Bicep
- Message queues and stream processing

Optional hands-on exercises:

- Implement a Service Bus topic and queue




# End of presentation