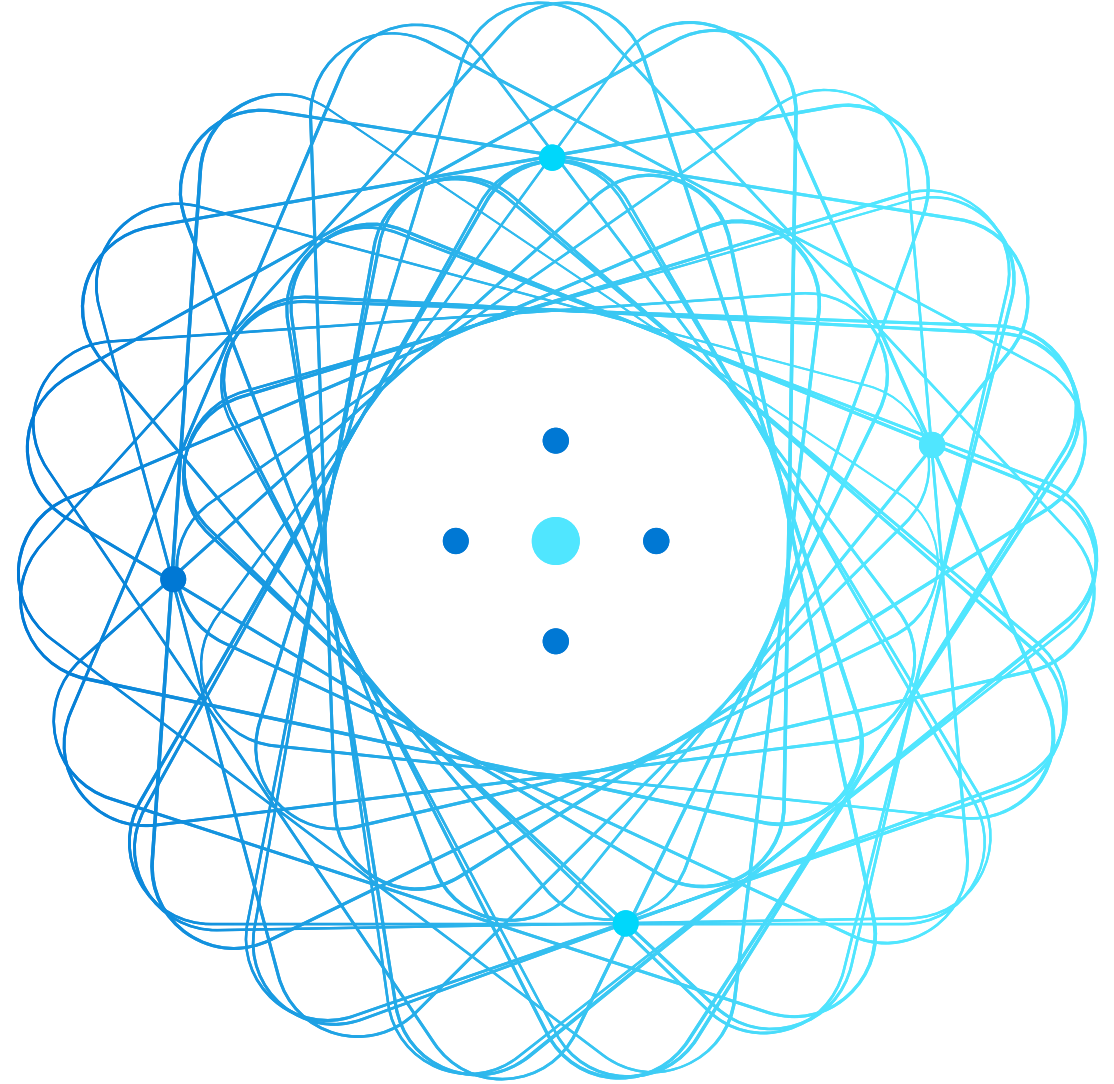**Microsoft Azure**

AZ-305

Designing Microsoft Azure Infrastructure Solutions

AZ-305 Agenda

Module 01 Design a governance solution

Module 02 Design a compute solution ← *VM App Service ACI AK*

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 ← *Event based Function LogicApp*

Module 07 Design Authentication and Authorization Solutions *Message*

Module 08 Design a solution to log and monitor Azure resources *Queue*
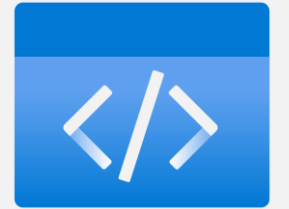
Module 09 Design a network infrastructure solution

Module 10 Design a business continuity solution

Module 11 Design a migration solution

Design an application architecture solution

Introduction

- Describe message and event scenarios
- Design a messaging solution *kafka*
- Design an event solution (Event Hub and Event Grid)
- Design an application optimization solution
- Design application lifecycle
- Case study
- Summary and resources

AZ-305: Design infrastructure solutions (25-30%)

Design an Application Architecture

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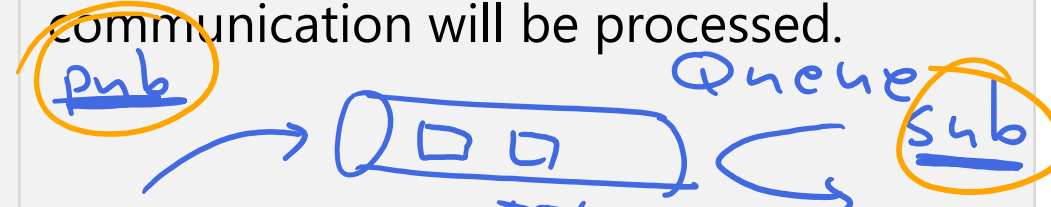- Recommend a solution for API integration

Describe message and event scenarios

Determine message and event scenarios

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

Cloud native ?

| Action	Description	When to use
Event	• Light weight • 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	• Contains raw data, produced by one component, that will be consumed by another component. • 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.

Massage

Pub *Queue* *Sub* *TTL* *Message*

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
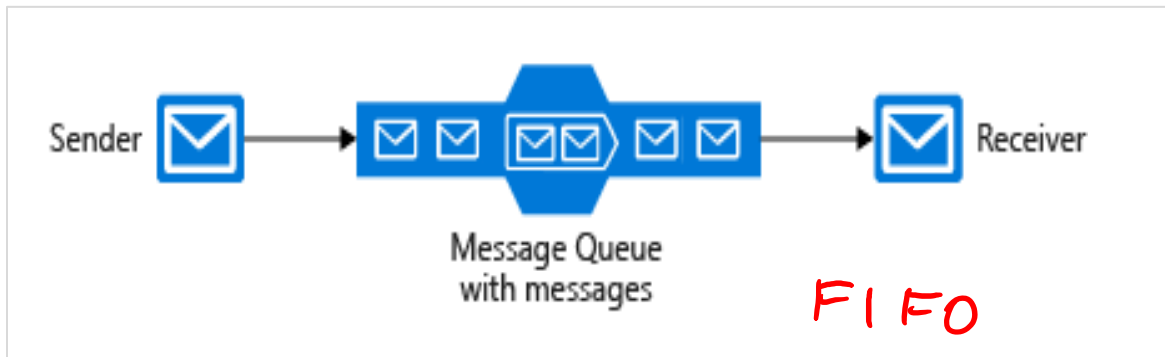

Tasks

Message Queue
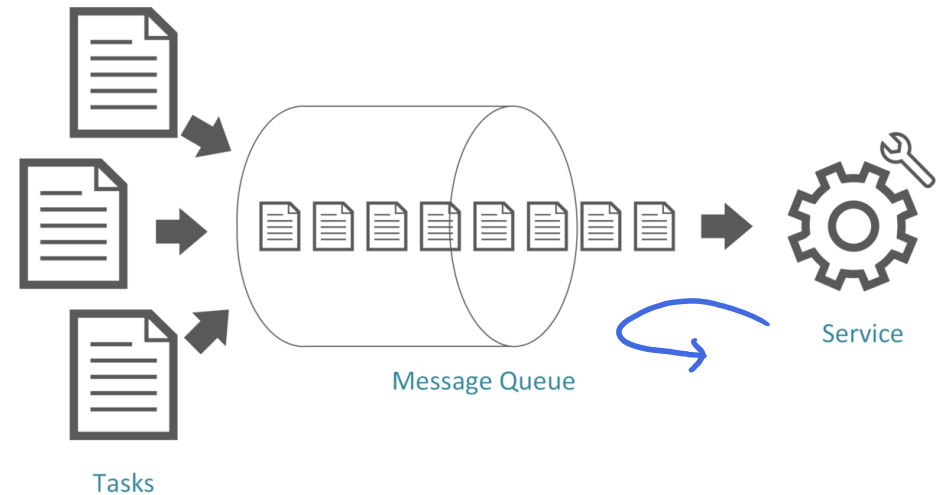
Service

- Create a backlog of work to process asynchronously

- Example: customer placing orders online added to the queue and processed

Design for Service Bus queues and topics

Service Bus decouples applications and services from each other.

Service bus queues	Service bus publish-subscribe topics

Sender → Message Queue with messages → Receiver

FIFO (handwritten)

Sender → Topic with three Subscriptions with messages → Receiver, Receiver, Receiver

- Built on top of a dedicated messaging infrastructure
- Holds messages until the target is ready to receive them – different from queues
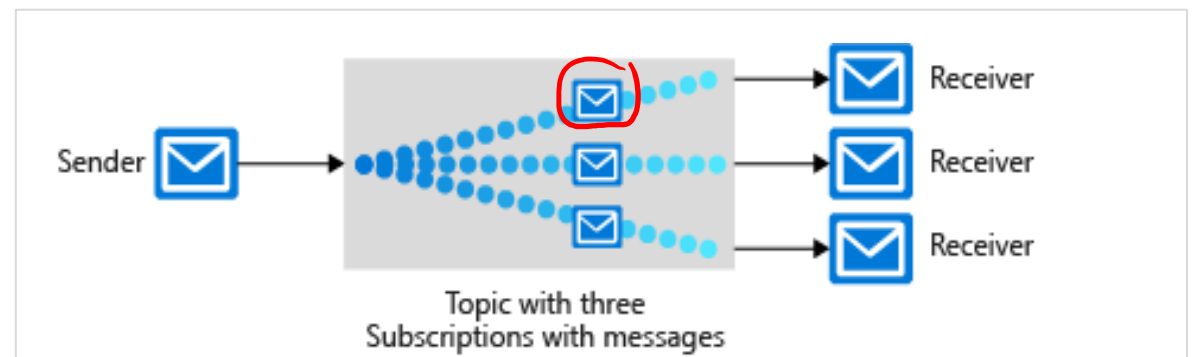
- Like bus queues but with multiple subscribers
- When a message is sent to a topic, multiple components can be triggered to perform a task

Compare messaging solutions

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

Event Subscription

Blobs

New
Delete

Events

Logic App

Trigger — if New

email

Connector
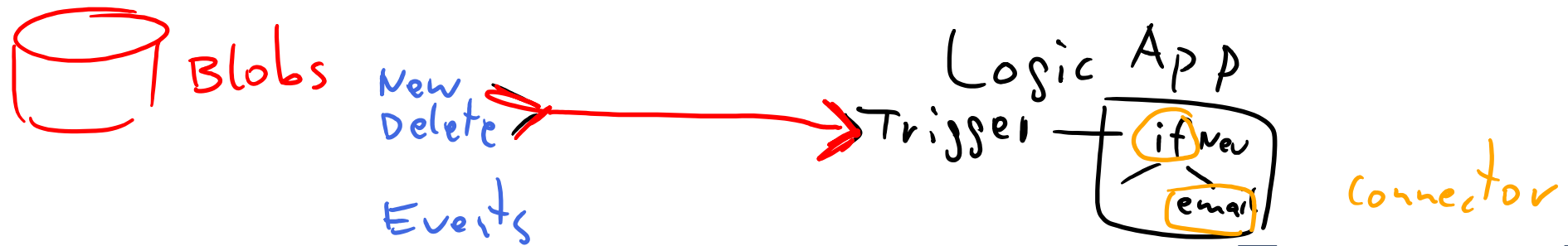
Design an event solution

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 IoT and real-time streaming data
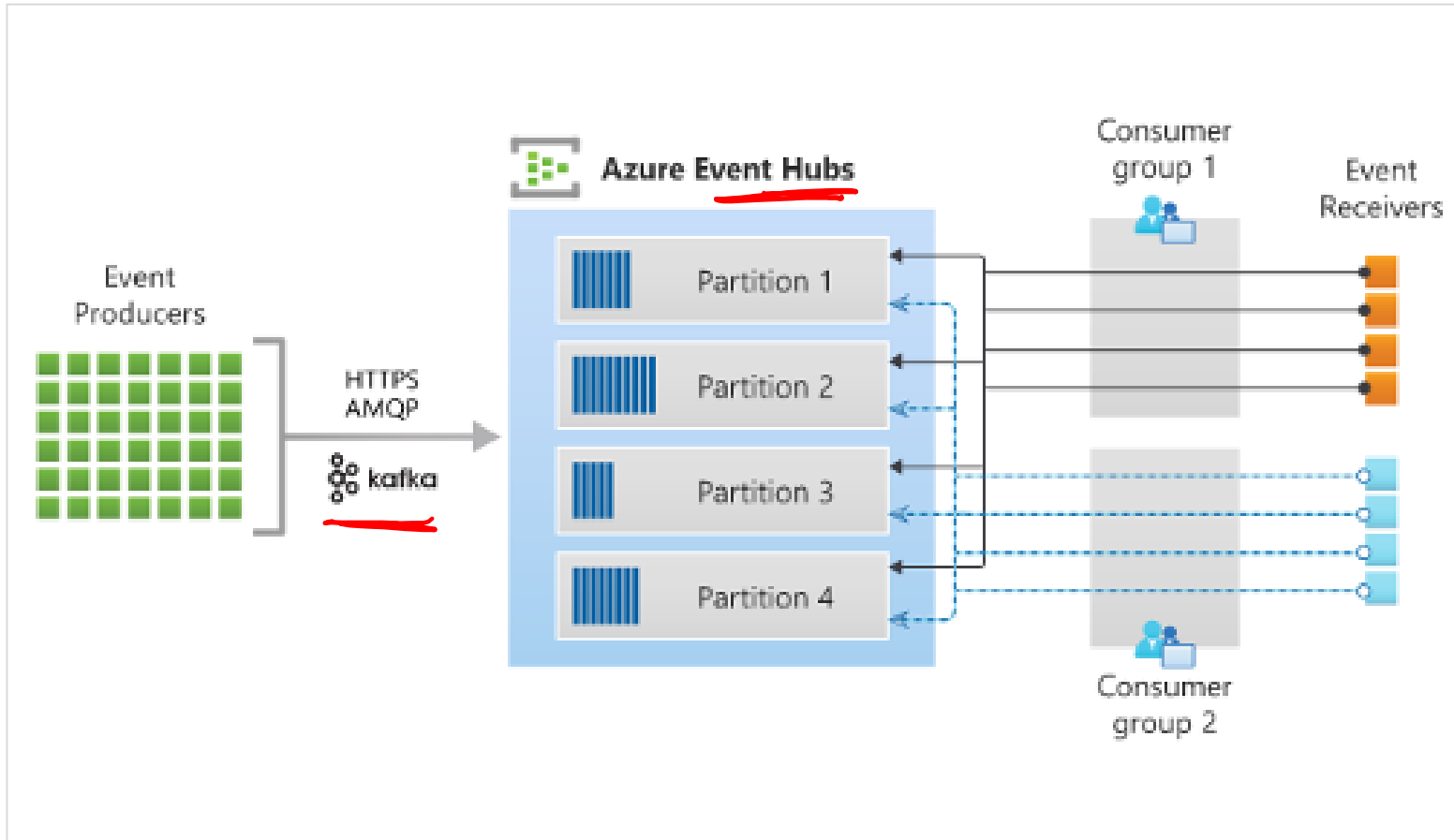
Design an event-driven solution

Azure Event Grid is a routing service connecting data sources with event handlers.
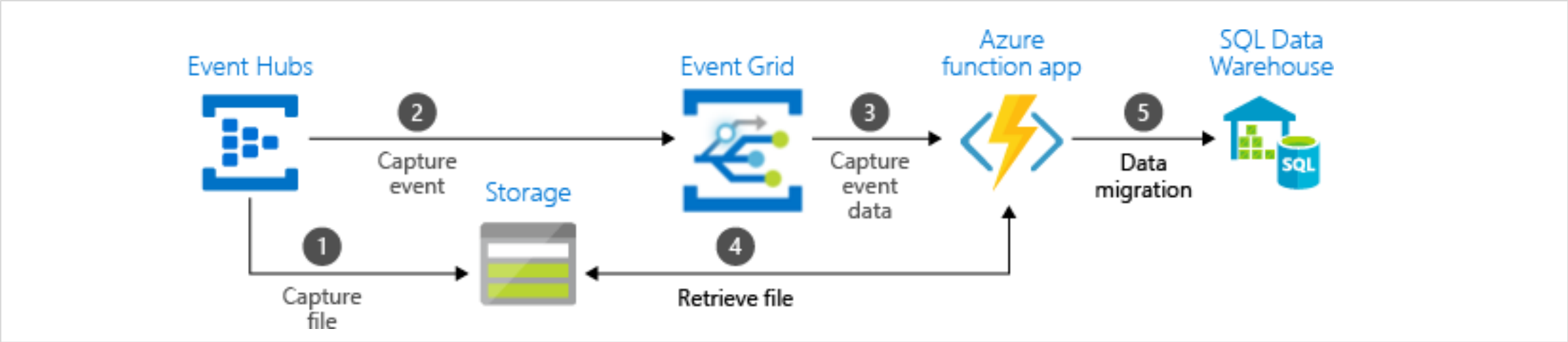
[handwritten: Event Subscription]

- 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

[handwritten: ITTT]
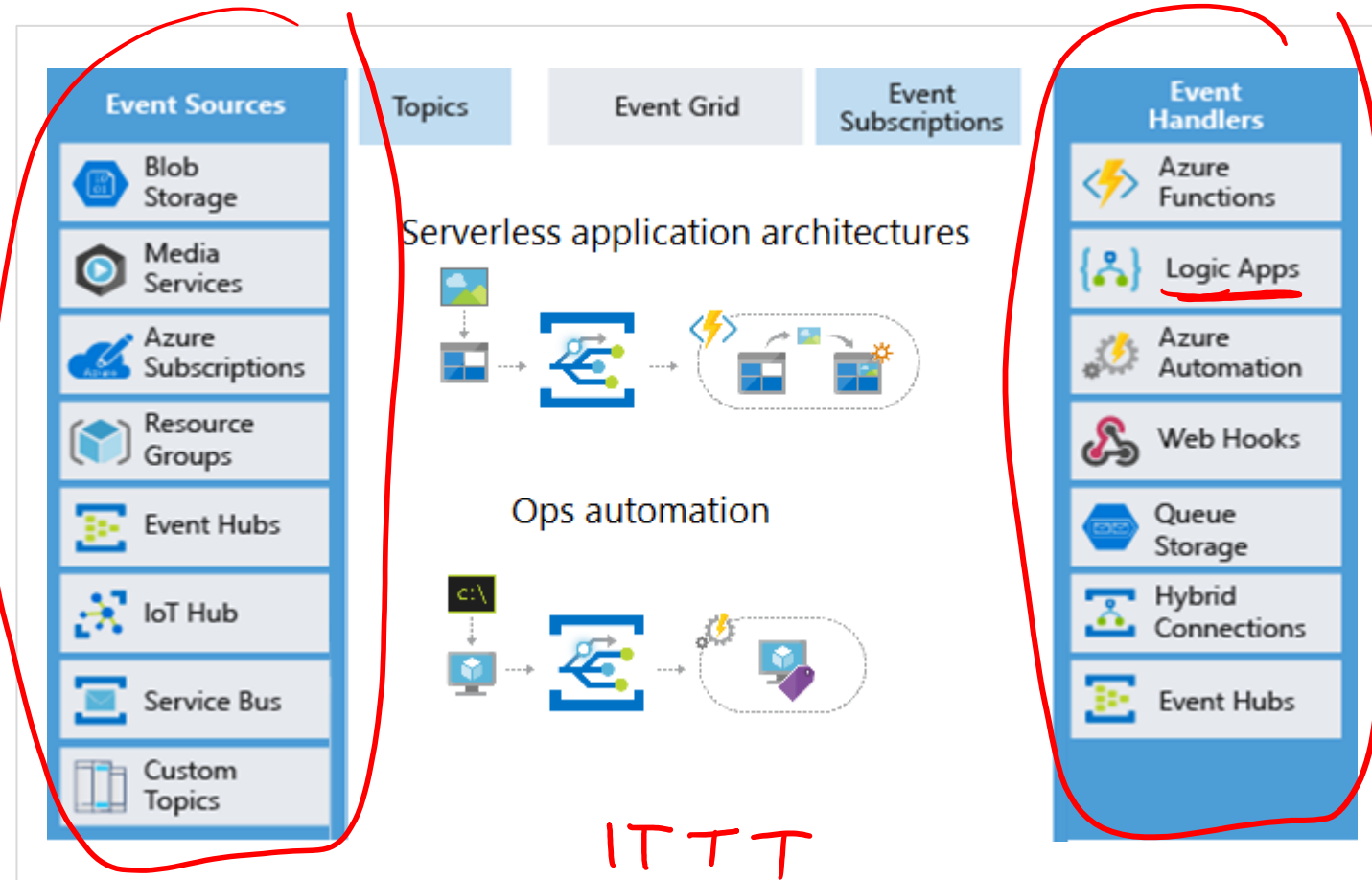
Comparison of message and event solutions
Consider combining several solutions

Service	Purpose	Type	When to use
Event Grid	Reactive programming	Event distribution (discrete)	React to status changes *Monitoring*
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 Queue
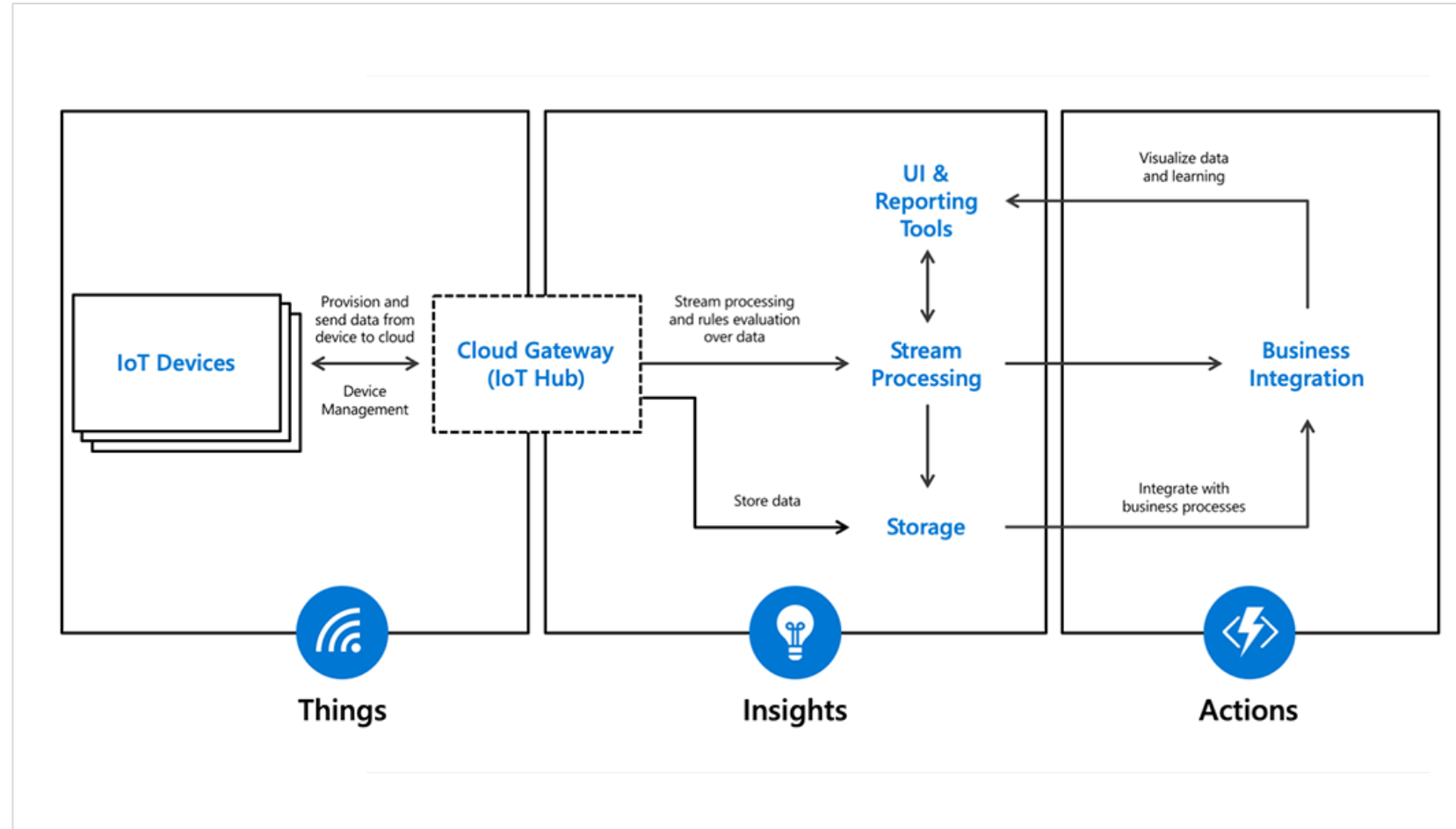
Design an IoT Hub solution

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

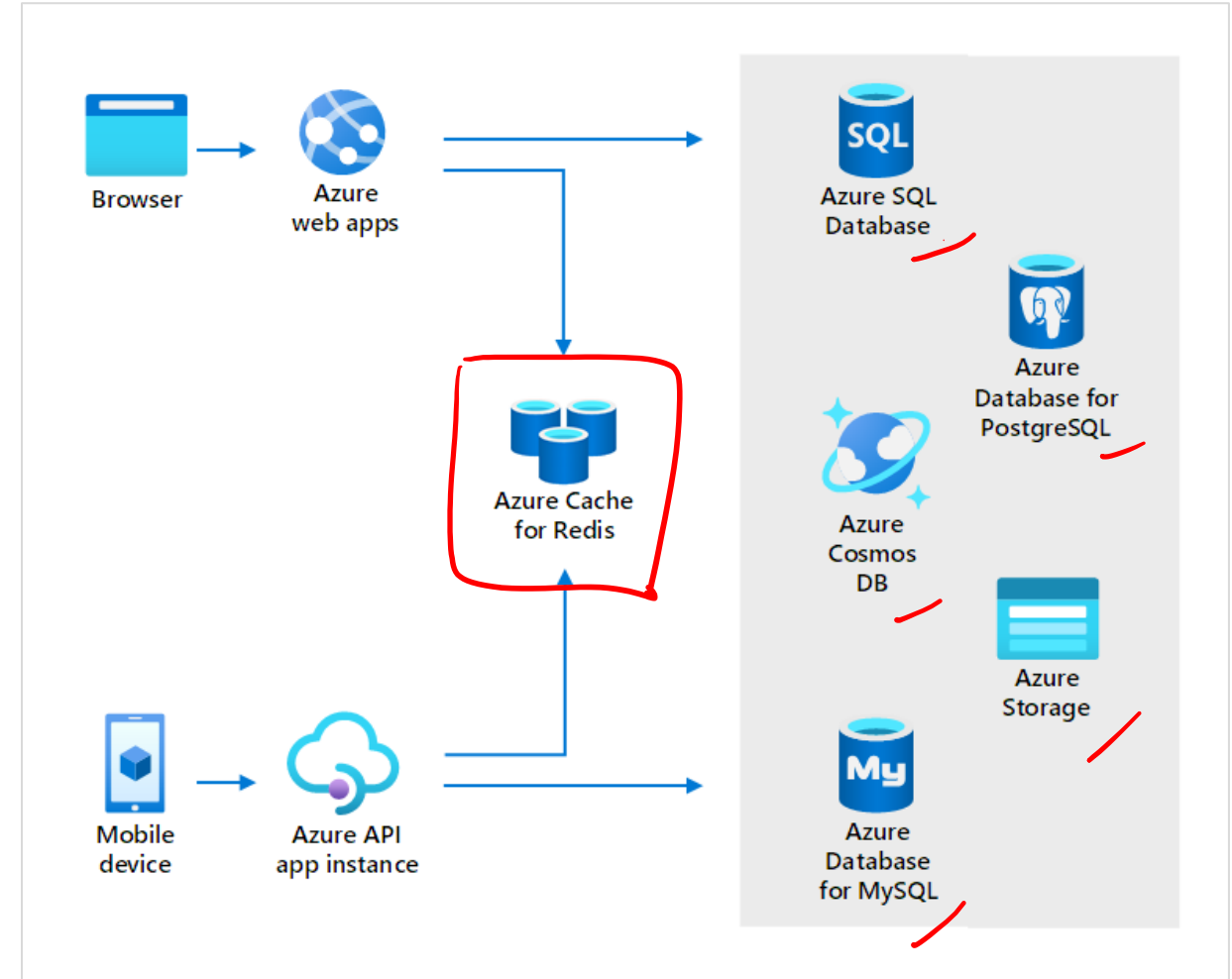- File upload from devices

- 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
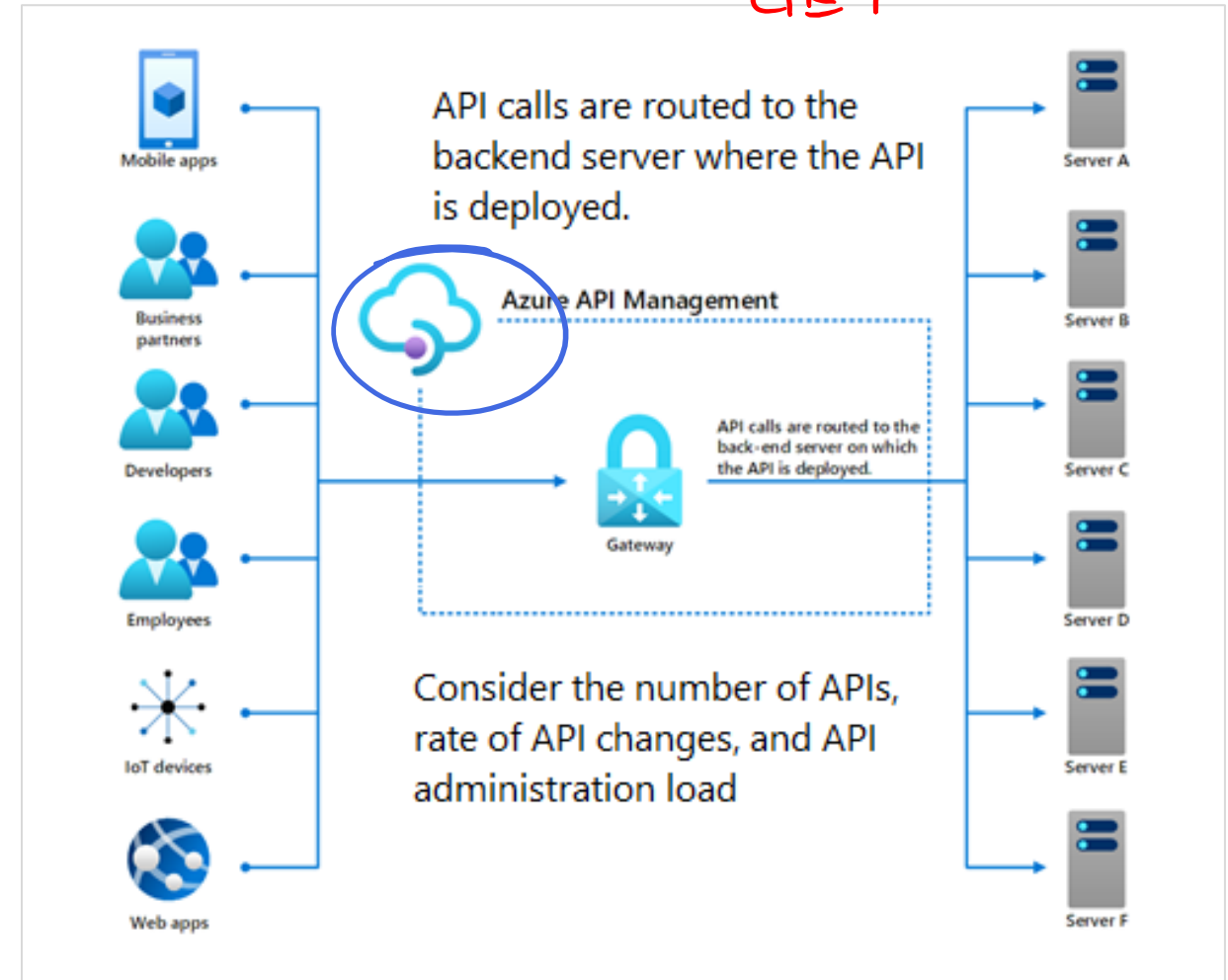
Design an Azure API management solution

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

Mobile apps

Business partners

Developers

Employees

IoT devices

Web apps

API calls are routed to the backend server where the API is deployed.

Azure API Management

API calls are routed to the back-end server on which the API is deployed.

Gateway

Consider the number of APIs, rate of API changes, and API administration load

Server A

Server B

Server C

Server D

Server E

Server F

Design an application lifecycle

What is Infrastructure as Code?

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

(handwritten annotations)

az group deployment -- file
main.bicep

Terraform
HCL
ARM
AWS
vmware
plugin

ARM API

json

2013 ARM Templates
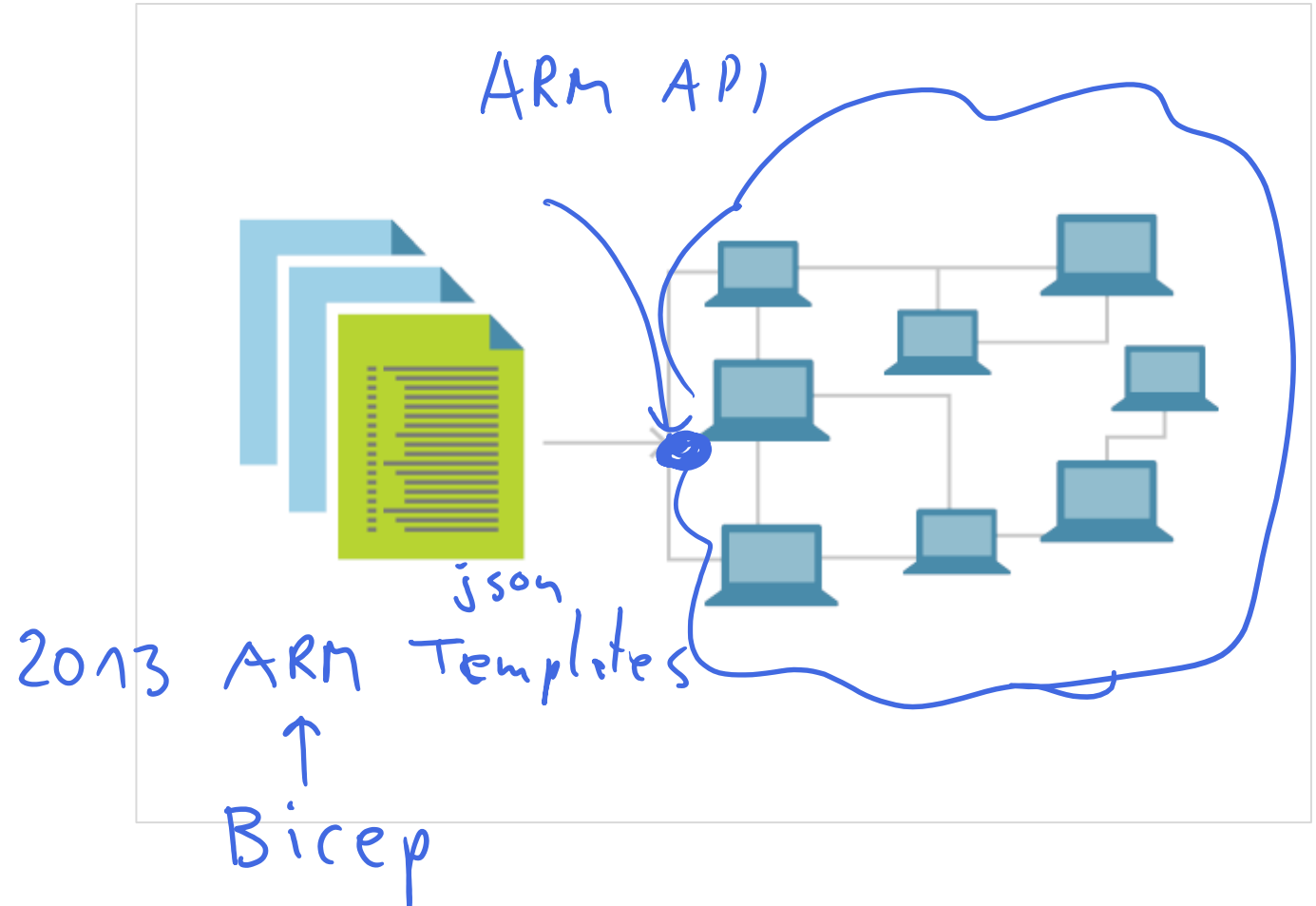
Bicep

Provision resources with Infrastructure as Code

* a C

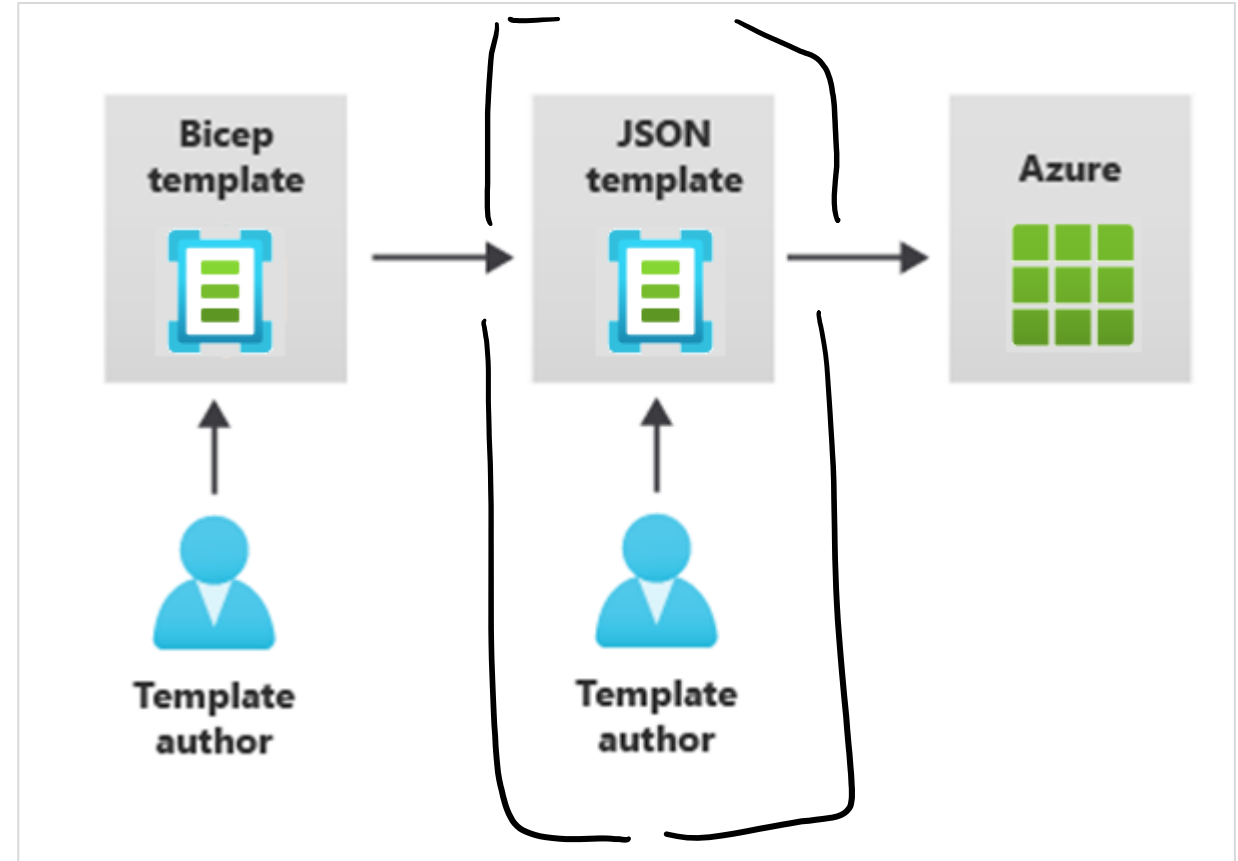## Azure supports IaC with Azure Resource Manager and third-party platforms.

- Azure Resource Manager templates – Bicep, JSON

- Azure Automation

- Azure DevOps services

- GitHub actions

- Terraform

- Jenkins

- Ansible
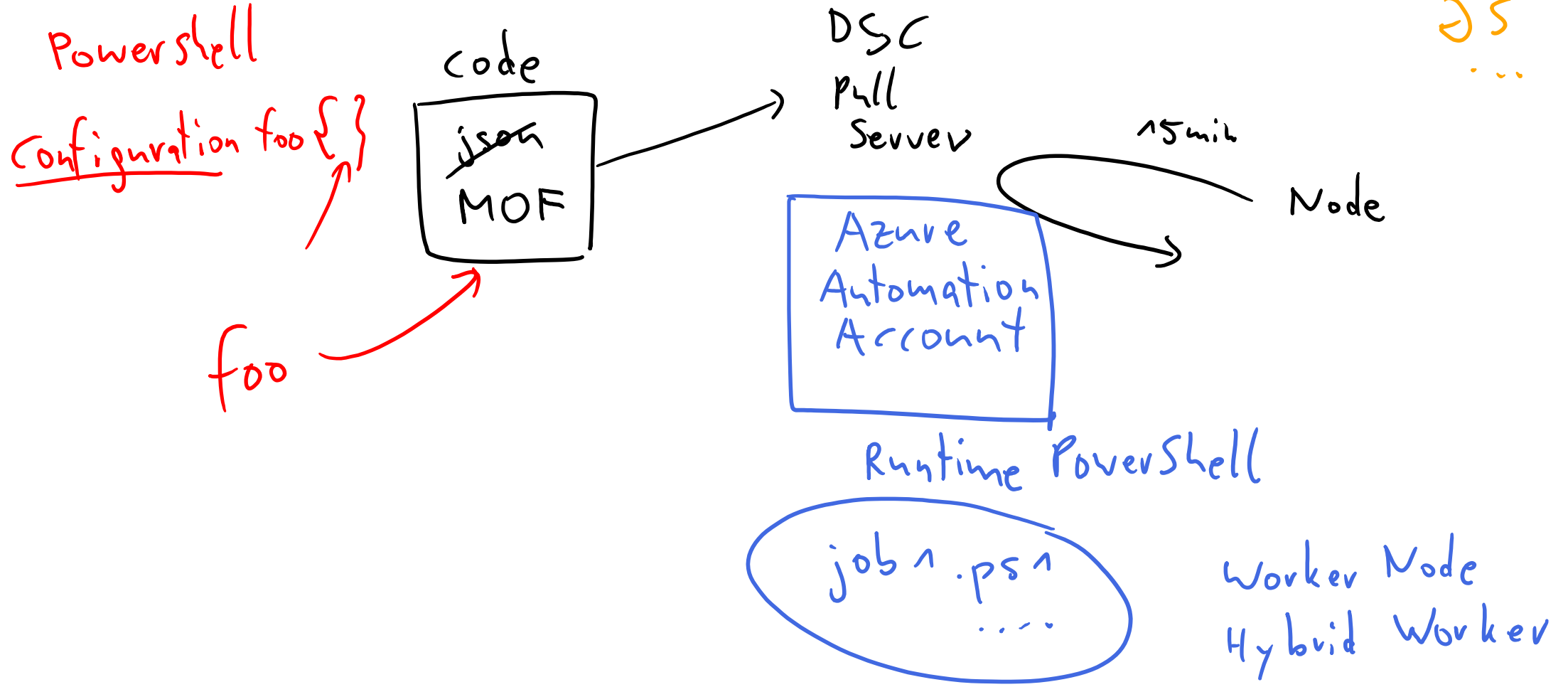
- Pulumi

- DSC

- Chef
- Puppet

Microstt Desired State Config (DSC)

Azure Functions
PowerShell
.Net
JS
...

PowerShell
Configuration foo{ }

foo

Code
~~json~~
MOF

DSC
Pull
Server

~15min

Node

Azure
Automation
Account

Runtime PowerShell

job1.ps1
.....

Worker Node
Hybrid Worker

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

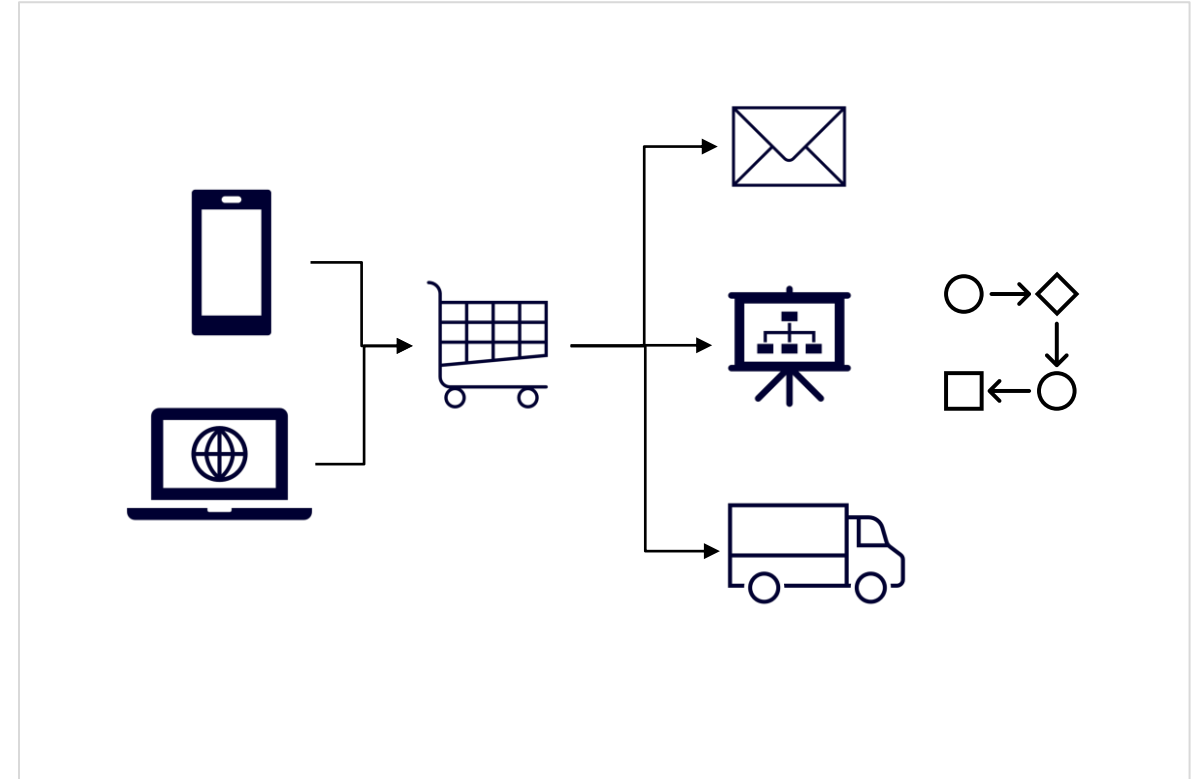- Enhanced security through Azure-managed identities and encryption

Config A   Config B   App Configuration
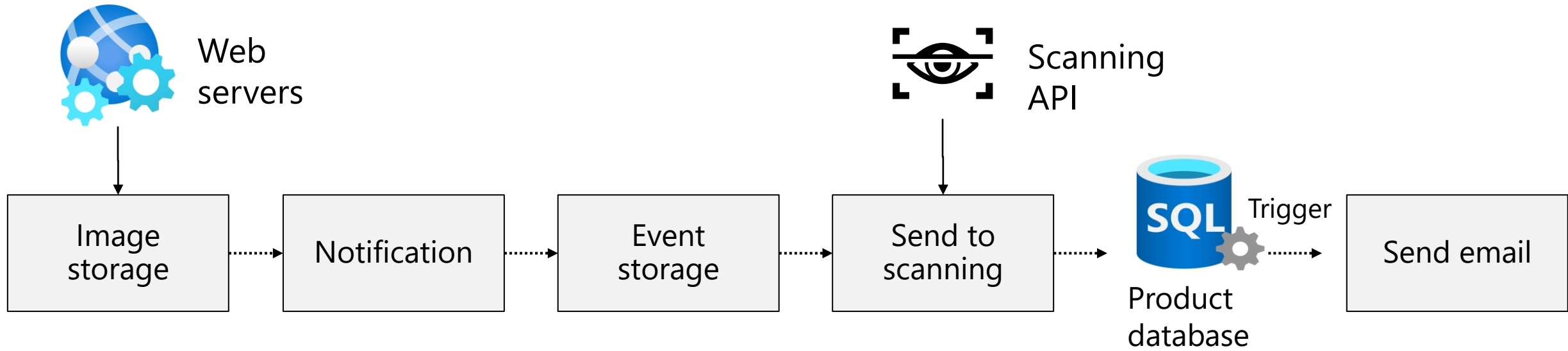
web.config
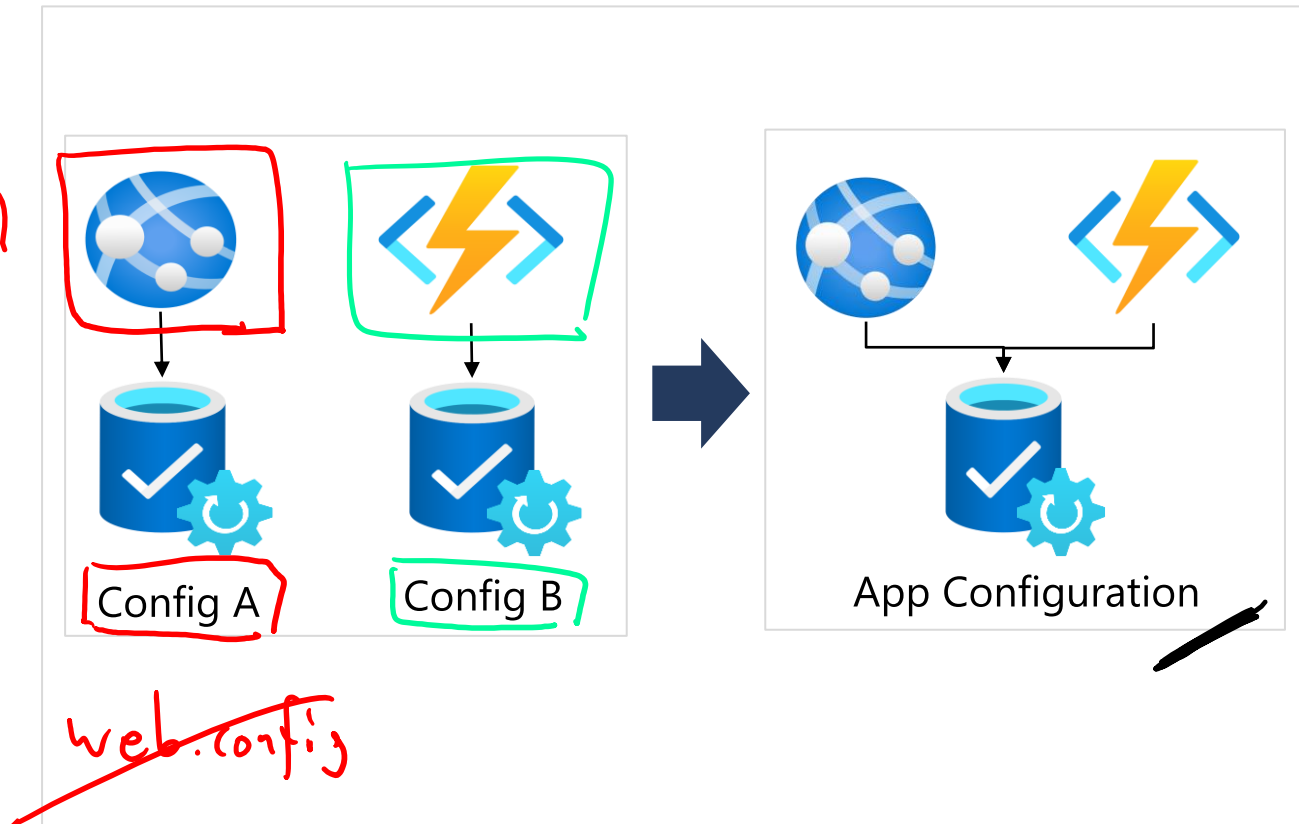
Review

Case Study – Application architecture

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

Web servers → Image storage ⟶ Notification ⟶ Event storage ⟶ Send to scanning ← Scanning API ⟶ Product database (SQL) — Trigger → Send email

Event Grid   Function   Logic App   Blob   Automation   Service Bus   Queues

Instructor Solution Diagram

Internet

Image

App Services

Website

Storage Account

Event Grid

Service Bus

Function

Product Database

Image Processing

Logic App

Azure Active Directory

Scanning API

On-premises
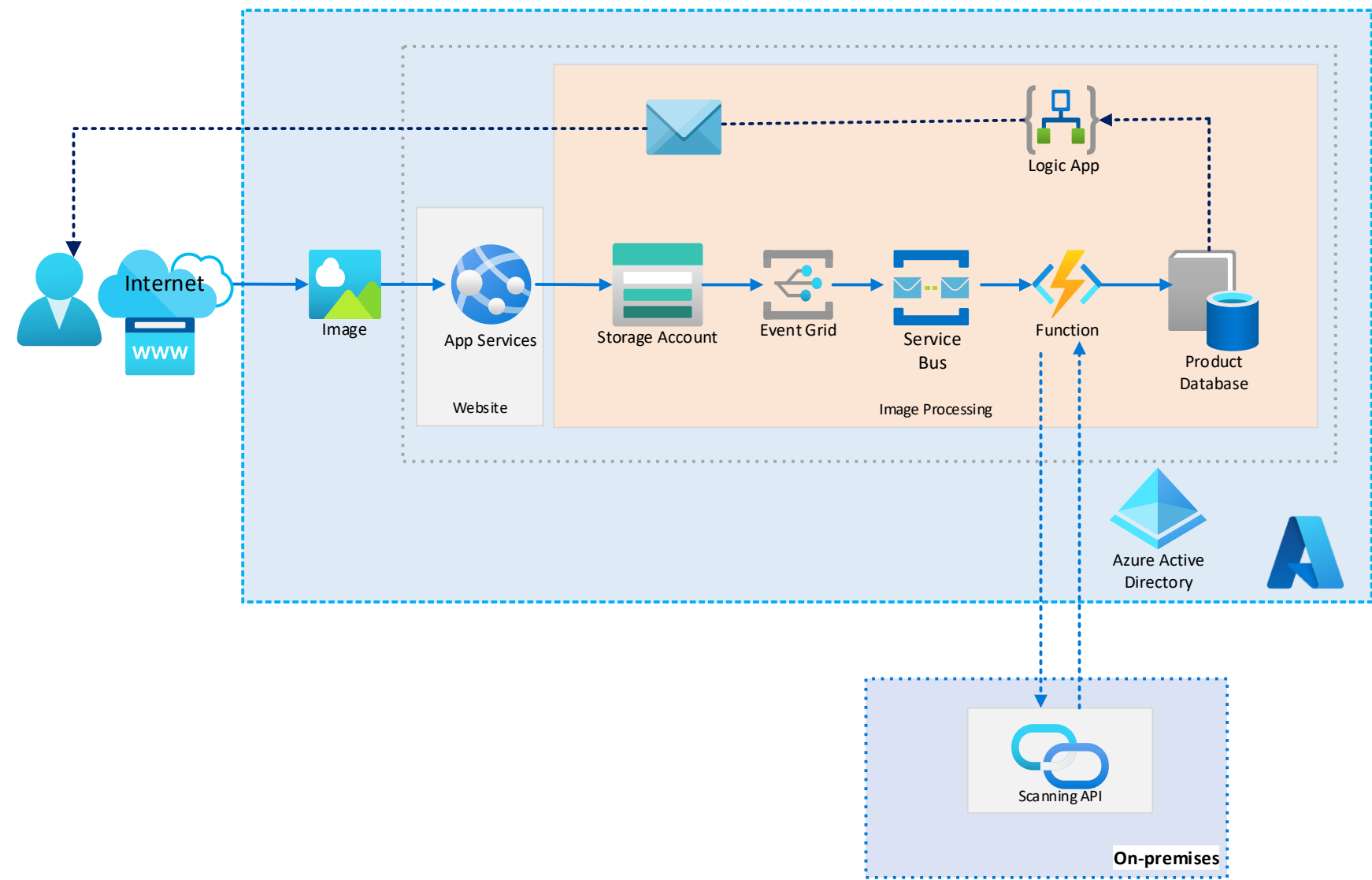
Summary and resources

Check your knowledge	Microsoft Learn Modules (docs.microsoft.com/Learn)

Choose a messaging model in Azure to loosely connect your services

Introduction to Azure API Management

Introduction to Event Hubs

Deploy Azure infrastructure by using JSON ARM templates

Introduction to infrastructure as code using Bicep

Message queues and stream processing

Introduction to Azure Cache for Redis

Optional hands-on exercise - Implement a Service Bus topic and queue - Learn | Microsoft Docs

End of presentation