Microsoft

Microsoft Cloud Workshop

Serverless architecture

Nick Ward
http://linkedin.com/in/nickward13
nickward@microsoft.com

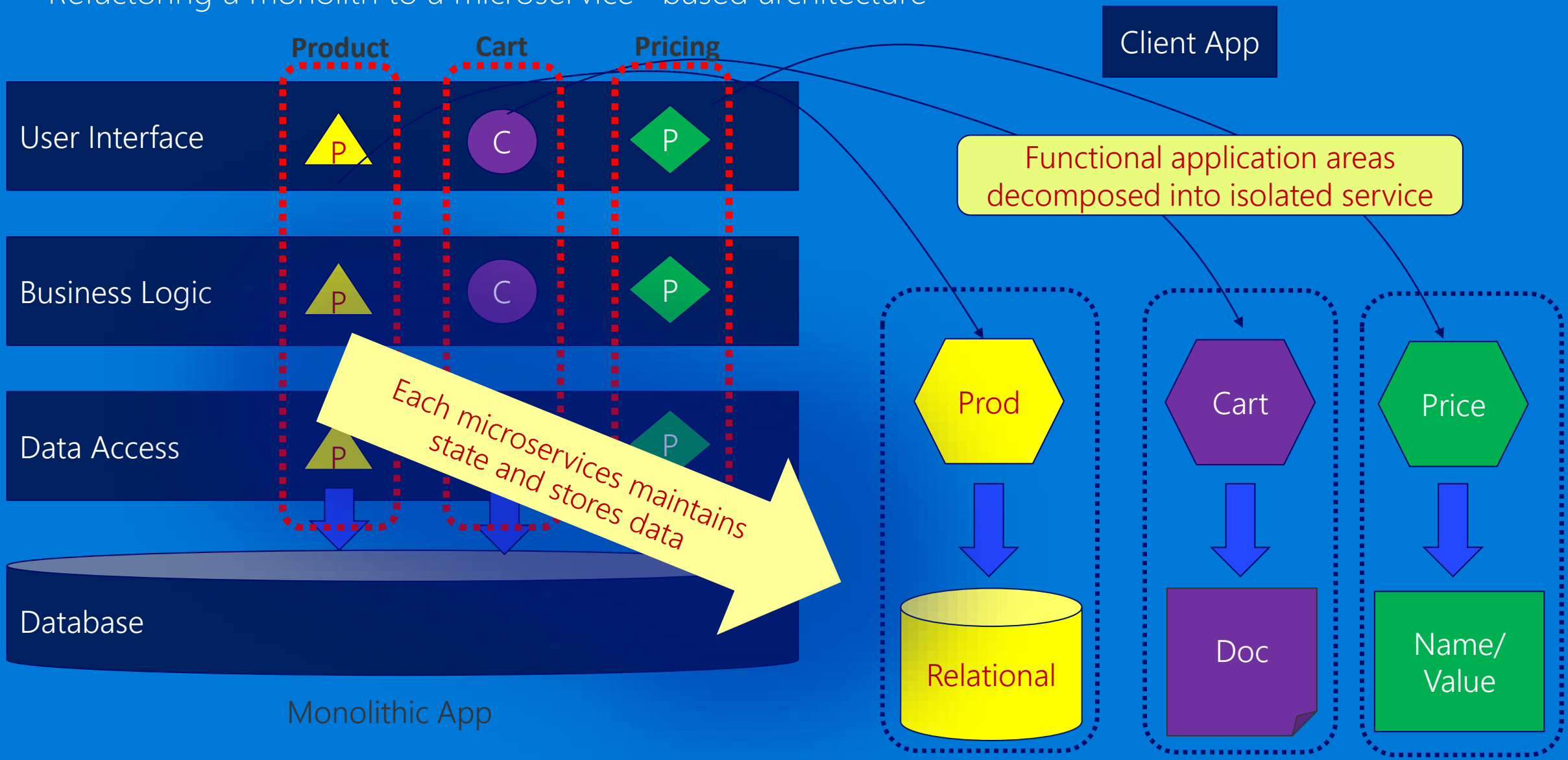
What are Microservices?

What are Microservices?

- An approach to application development in which a large solution is built as a suite of modular services
- Each service supports a specific business goal and uses a simple well-defined interface to communicate
- Each service is isolated, has its own dependencies, domain and logic and evolves independently
- Embracing cross-platform, each can be written in a different programming language, use different data storage technologies and scale independently

Microservice Architecture

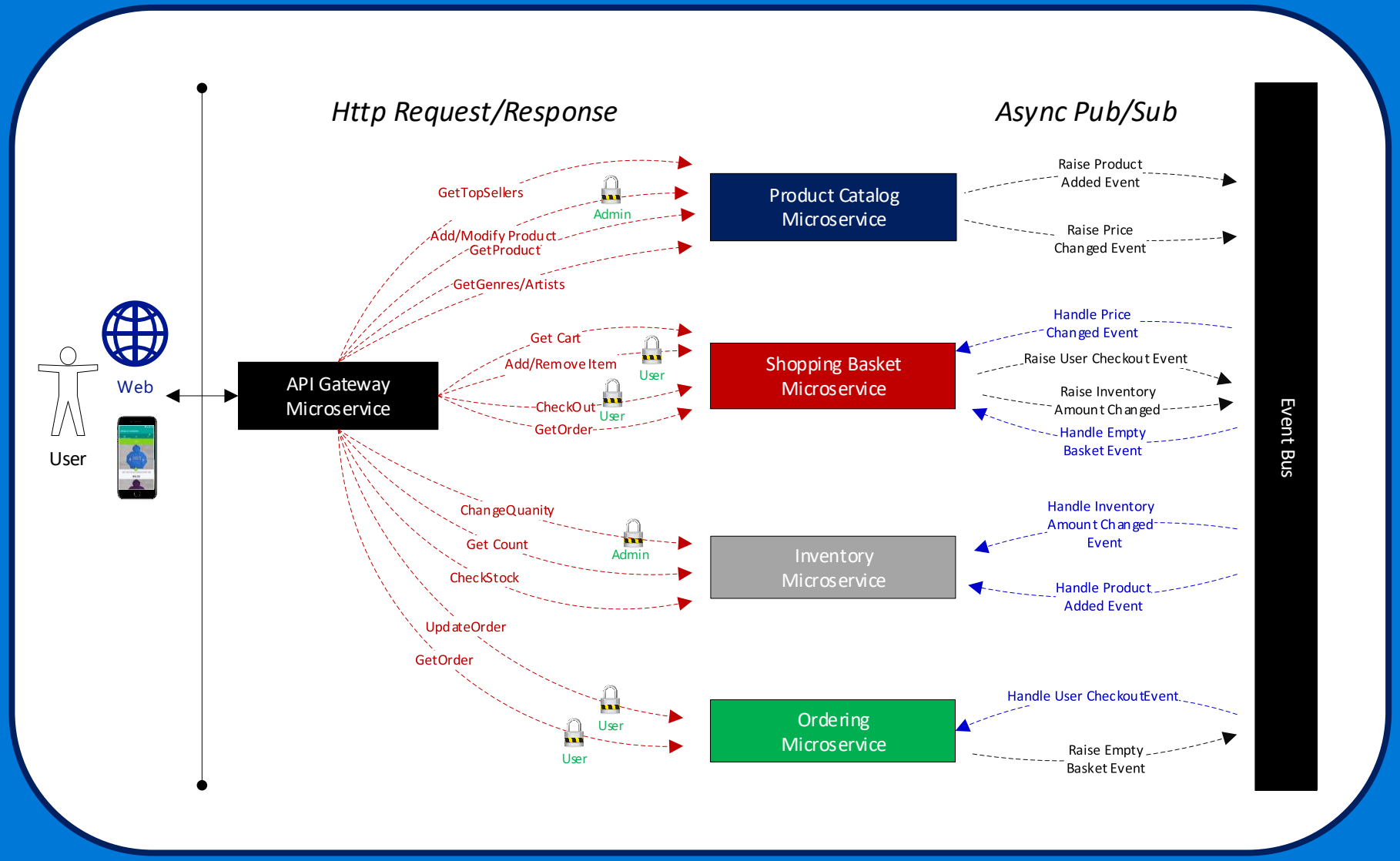
- Refactoring a monolith to a microservice –based architecture



Microservices != Containers

- Microservice: Isolated application with a single purpose
- Container: Encapsulate discrete components of app logic and the corresponding runtime
- Containers are frequently used to host microservices
  - Provide fine-grained execution environment
  - High degree of isolation
  - Fast initialization and execution

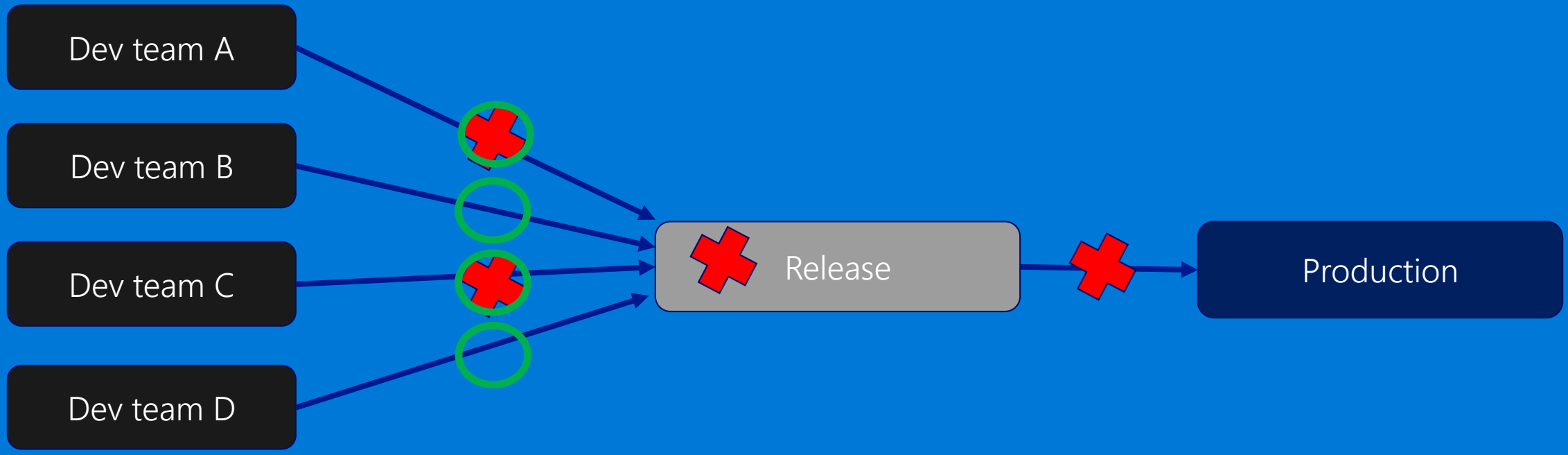
Microservices Architecture - Detail View



Microservices:
Benefits and Challenges

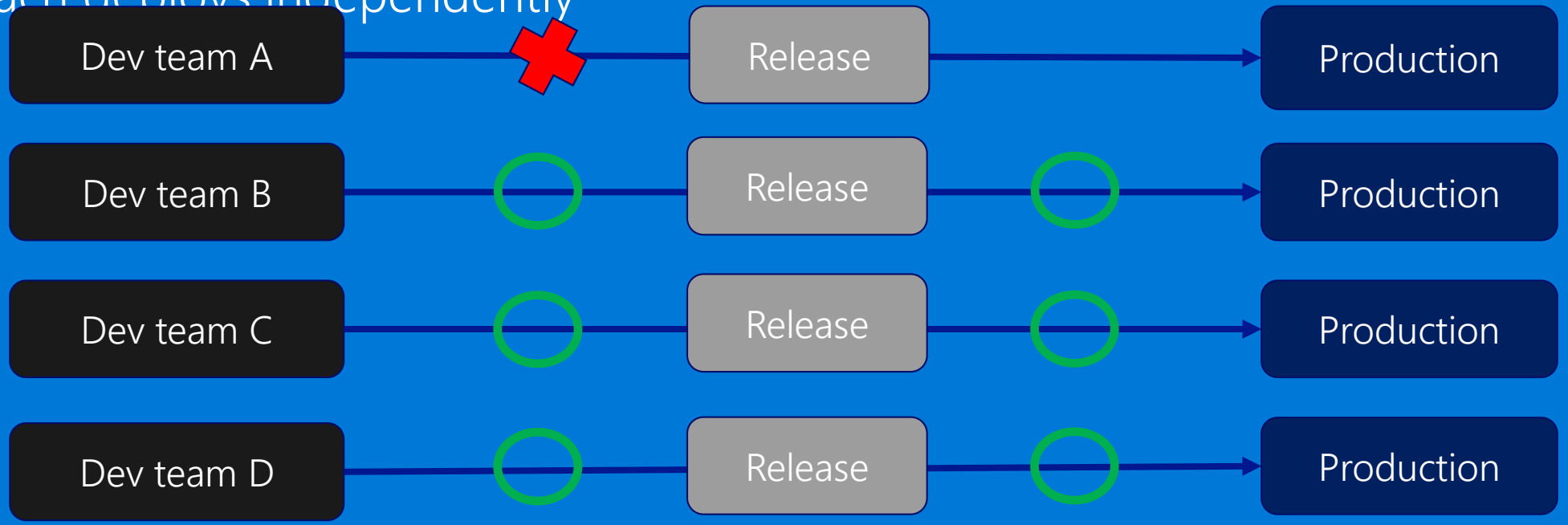
How Monoliths Destroy Agility

- Single codebase with single release pipeline
  - All teams share same dependencies – tightly-coupled
  - All teams release in the same cadence
  - A defect in a dependency can block multiple teams and the release itself

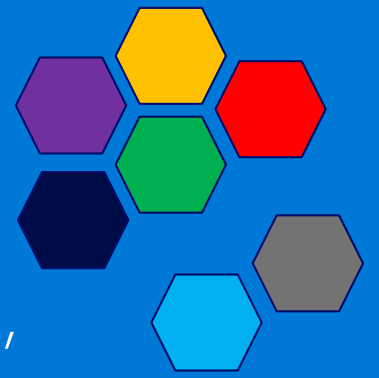


Agility Benefits of Microservices

- Each team owns it own service and deploys separately
  - Services are isolated and *do not directly share dependencies*
  - Each has its own release cadence
  - Each deploys independently



Microservices - Benefits



- Encapsulates business functionality into small targeted services, organized around business capabilities
- Services deploy frequently and evolve independently
- Services scale Independently
- Enables technology diversity
  - Mix multiple programming platforms and data-storage technologies – best tool for the job
  - Rewriting/modernizing a single service is feasible
  - "Future-proofs" application investment against obsolete technology stacks
- Failure in one service less likely to cause system-wide failure

Microservices - Challenges

- Architectural and operational complexity increase
- Remote calls escalate network I/O, congestion and latency
- Distributed services expose points of failure, reducing reliability
- Decentralized data require eventual consistency delays
- Integration and versioning concerns become critical
- Service discovery and routing concerns must be handled
- Testing – Stubs/mocks become key
- Orchestration, management and monitoring are mandatory

Microservice Candidates

- Strategic enterprise systems too complex to manage as a monolith
- Systems that that will change frequently
- Systems developed by large teams with frequent churning
- Systems with components that must scale independently
- Systems constructed with different technology stacks

Microservice Considerations

- Defining service boundaries
- Implementing a Gateway
- Implementing inter-service communication
- Data integrity (across stores)
- Resiliency (partial failure)
- Hosting

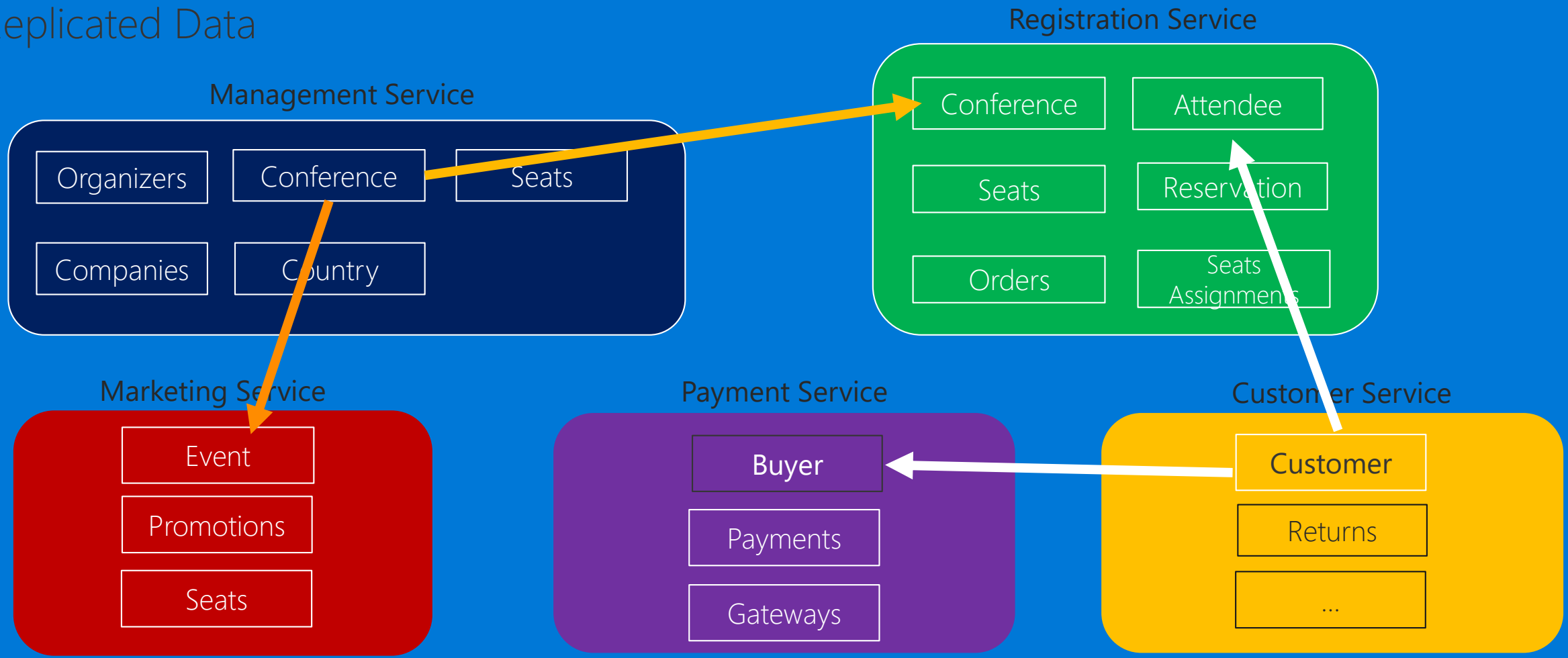
Microservices: Decomposing and Modeling

Identifying Services

- First, you must identify service boundaries
- A common strategy is to decompose by business capability
- Each distinct capability becomes a *Bounded Context*
  - DDD pattern for dividing up large domain models
- Each bounded context is autonomous and has its own…
  - Domain model
  - Data Store
  - Ubiquitous language
  - Invariants, rules, code

Bounded Context Example

- Conference Management System
- Functional Boundaries
- Replicated Data



Azure Services for Microservices, Serverless and Containers

Xamarin

Web Apps

PowerApps

API Management

AKS

Service Fabric

App Service

Functions

Logic Apps

IaaS

Low Level PaaS

High Level PaaS

Containers

Serverless

Microservices

+

Containers

+

Serverless

+

APIs

=

Faster Innovation

Abstract and learning objectives

Abstract

Setup and configure a serverless architecture, breaking down the solution to smaller components that are individually scalable, and allowing the customer to only pay for what they use.

Learning objectives

- Independently scale and break down business logic into discrete components
- Use computer vision algorithms within Azure
- Create a Logic App to act as a workflow
- Monitor the serverless topology, observing how well the solution scales when under load
- Implement a Continuous Deployment DevOps process in the serverless architecture

Step 1: Review the customer case study

Outcome

Analyze your customer needs

Timeframe

15 minutes

Customer situation

Litware, Inc. manages a number of toll booths, taking photos of vehicles and billing the drivers.

License plate detection is currently a manual process.

They are trying to cope with being overloaded due to faster-than-expected growth.

Customer situation

Litware is confident that their billing system can handle the load.

They are concerned about how rapidly they can automate the license plate processing while ensuring the new automated solution can scale to meet demand, especially during spikes in traffic.

Customer situation

Would like to use machine learning service to perform license plate recognition task

Needs to store images in cloud storage and data in database for export and manual verification

Wants an automated workflow to export license plate data and send conditional alerts

Customer situation

Our directors want to see where we can take the notion of a serverless architecture, and see if there truly are long-term performance and cost benefits.

With the unexpected windfall of the toll booths contract, they want to make sure we have a tested strategy we can fall back on in the future…

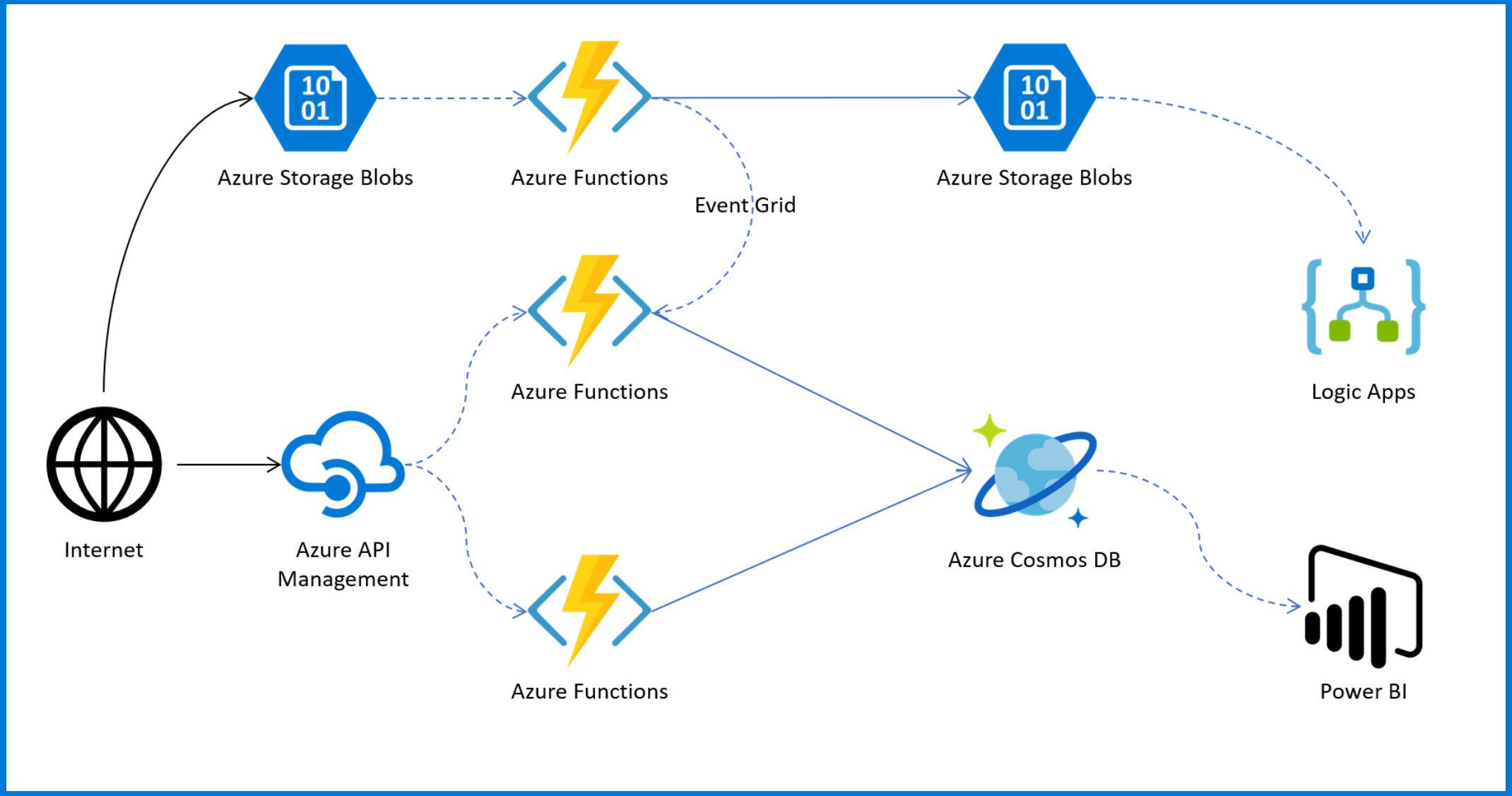
Customer needs

- Automate manual process using serverless
- Use ready-made machine learning service
- Manually enter license plate numbers for images that could not be processed
- Scalable solution that can handle unexpected demand
- Automated workflow that exports data
- Options to locally develop and automate deployment pipeline
- Centralized monitoring dashboard with real-time and historical viewing options

Customer objections

- How can the serverless components talk to each other?
- Will the dynamic scalability of the serverless components end up costing us a lot of money?
- How do we combat against erroneous image processing?

Common scenarios



Step 2: Design the solution

http://aka.ms/serverless-workshop

Outcome

Design a solution and prepare to present the solution to the target customer audience in a 15-minute chalk-talk format.

Timeframe

60 minutes

Business needs (10 minutes)	• Respond to questions outlined in your guide and list the answers on a flipchart.
Design (35 minutes)	• Design a solution for as many of the stated requirements as time allows. Show the solution on a flipchart.
Prepare (15 minutes)	• Identify any customer needs that are not addressed with the proposed solution. • Identify the benefits of your solution. • Determine how you will respond to the customer's objections. • Prepare for a 15-minute presentation to the customer.

Step 3: Present the solution

Outcome

Present a solution to the target customer in a 15-minute chalk-talk format.

Timeframe

30 minutes (15 minutes for each team to present and receive feedback)

Directions

- Pair with another table
- One table is the Microsoft team, and the other table is the customer
- The Microsoft team presents their proposed solution to the customer
- The customer asks one of the objections from the list of objections in the case study
- The Microsoft team responds to the objection
- The customer team gives feedback to the Microsoft team

Wrap-up

Outcome

- Identify the preferred solution for the case study
- Identify solutions designed by other teams

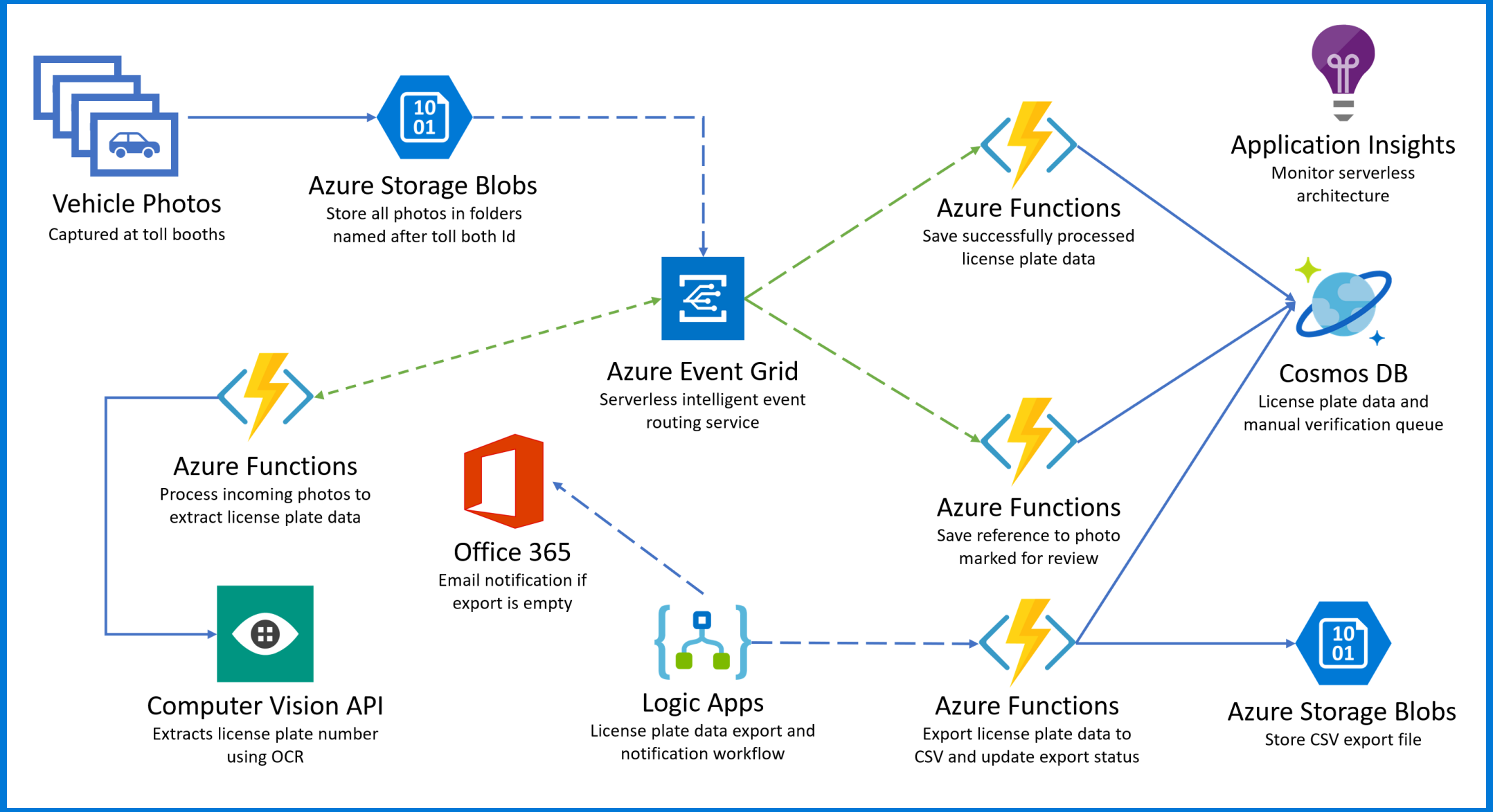
Timeframe

15 minutes

Preferred target audience

- Abby Burris, Chief Information Officer, Litware, Inc.
- The primary audience is business decision makers and technology decision makers.
- Usually, we talk to the Infrastructure Managers who report into the CIOs, or to application sponsors (like a VP LOB, CMO) or to those that represent the Business Unit IT or developers that report into application sponsors.

Preferred solution



Preferred solution

License plate processing serverless components

- Orchestrate event-driven activities with Event Grid
- Use Azure Functions for serverless compute with the Consumption plan
- Research and test downstream services to see if they can handle high demand – implement rate-limiting and resiliency strategy as needed
- Use Cosmos DB to store license plate data

Preferred solution

License plate OCR

- Use the Cognitive Services Computer Vision API and its built-in OCR capabilities
- The image processing function can make a REST call to send the photo and read the JSON data in return

Azure Functions
POSTs photo

Computer Vision API
Extracts text with OCR

JSON Response
Function reads JSON

Preferred solution

Data export workflow

Create a Logic App with the following:

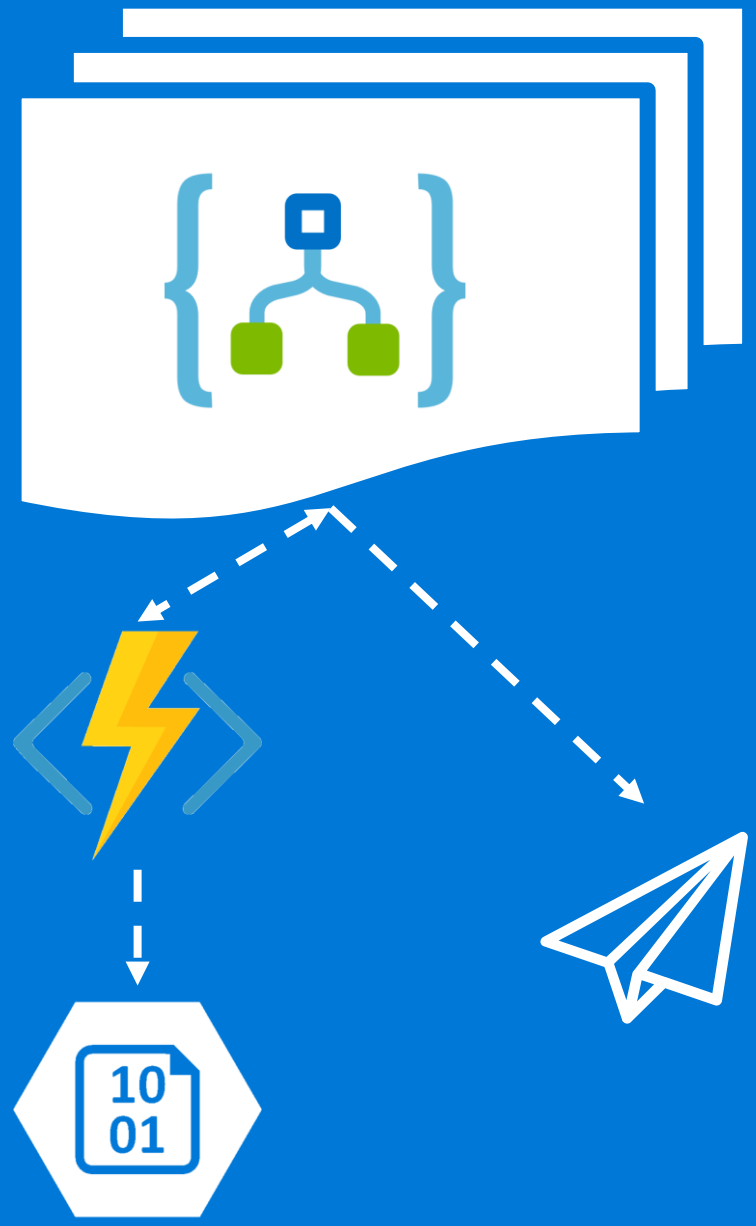
Recurrence trigger that executes every hour

then

Executes Azure function that exports CSV to storage

then

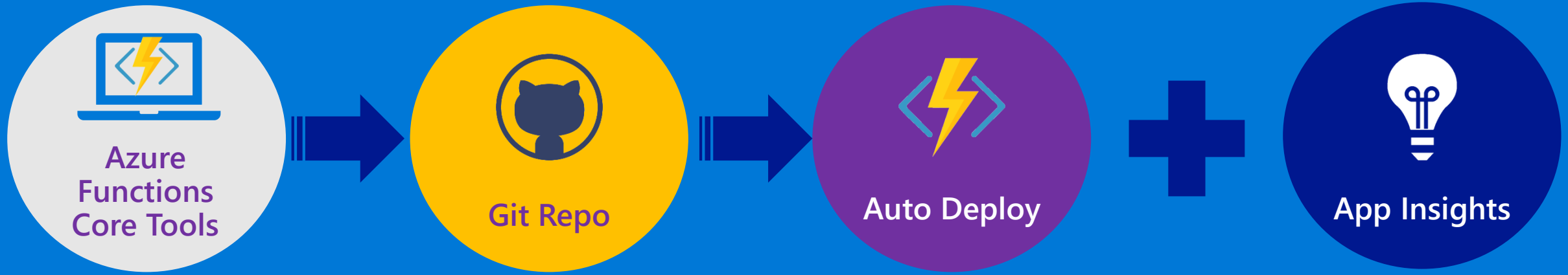
Uses conditional logic to evaluate response code from the function and send email using Office 365 Outlook connector



Preferred solution

Monitoring and DevOps

- Develop & debug Function Apps locally with Azure Functions Core Tools
- Automate function deployments through App Service continuous integration – use integrated source repo like GitHub, DropBox or VSTS
- Monitor all executing serverless components with App Insights, in real-time, and use it to configure alerts and view historical telemetry



Customer objections

- How can the serverless components talk to each other?
- Will the dynamic scalability of the serverless components end up costing us a lot of money?
- How do we combat against erroneous image processing?

Customer quote

"Thanks to Azure's serverless components, and the ease in which we can use them, we have been able to rapidly build a cost-effective and robust solution to replace our manual license plate recognition process. The first-class monitoring tools we can use with our new architecture has helped us confidently move forward and competently meet the high demands of our expanding customer base."

Abby Burris, CIO, Litware, Inc.

Microsoft