Build microservices applications with a serverless architecture

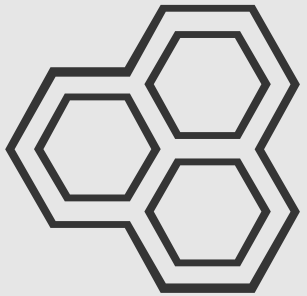
Lorenzo Barbieri
Cloud Solutions Architect
lorenzo.Barbieri@microsoft.com
linkedin.com/in/geniodelmale

CommunityDays.it

Microservices + Serverless

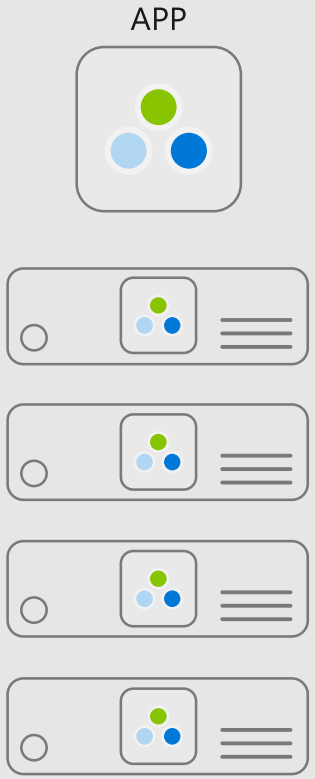
Microservices

Architectural approach for cloud native apps



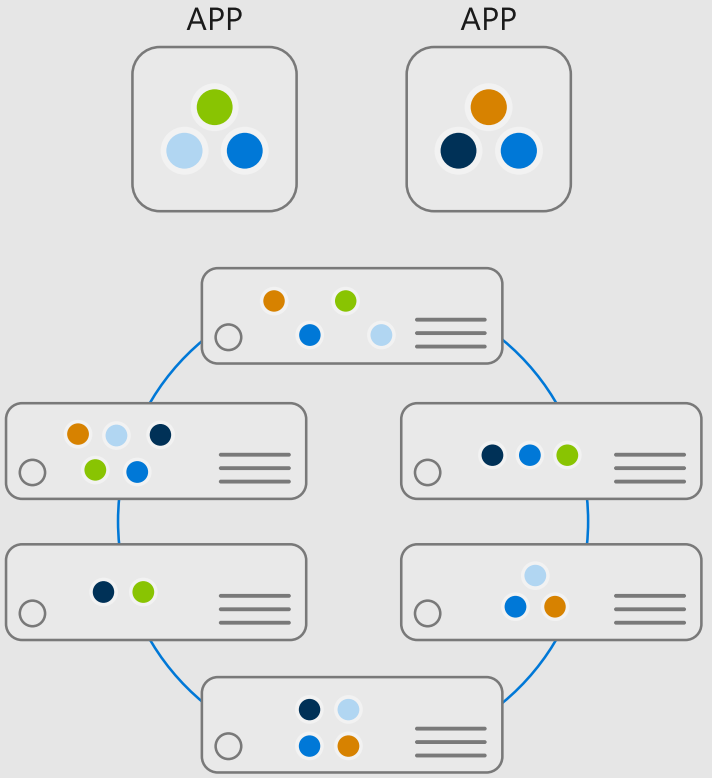
Monolithic

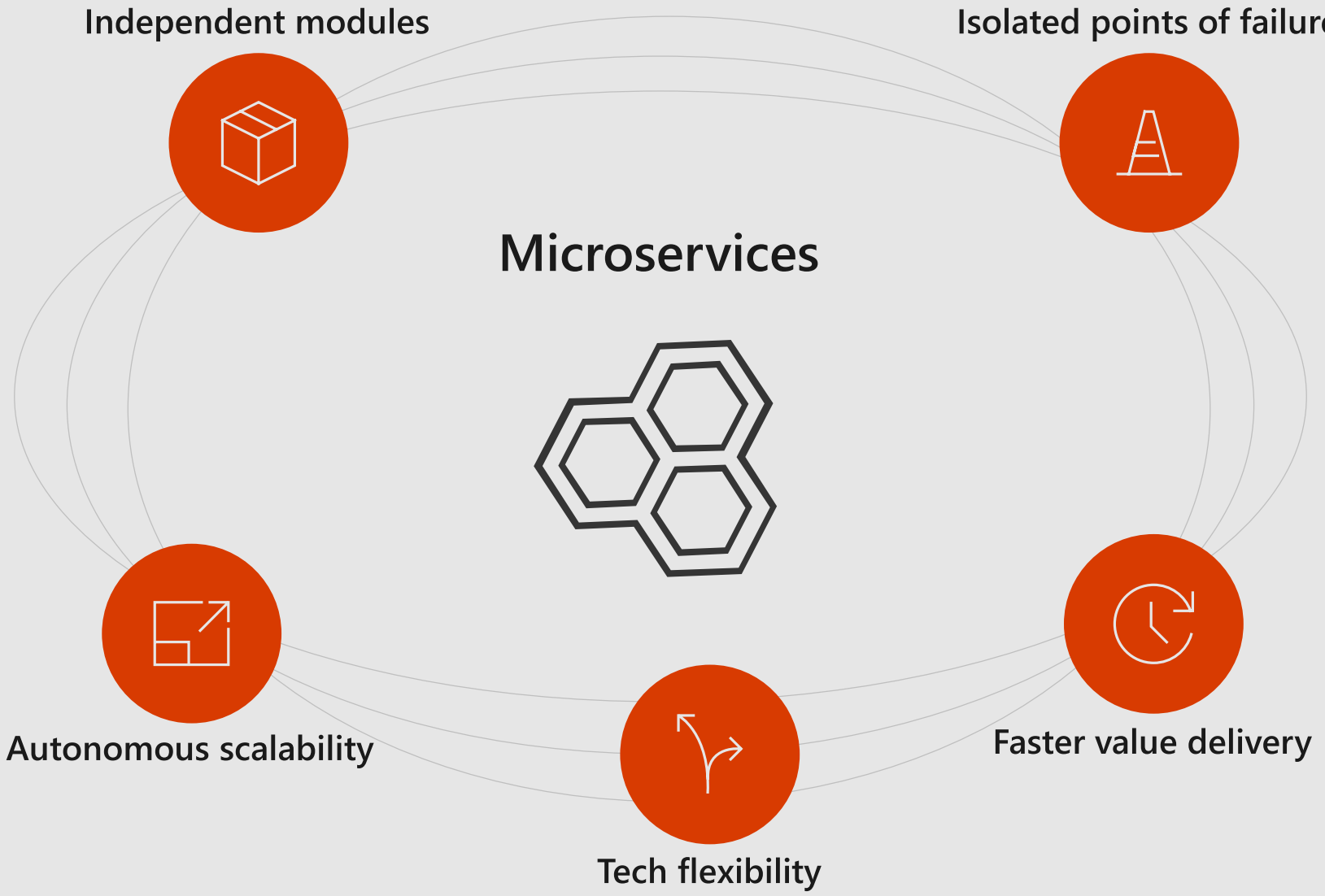
Large, all-inclusive app



Microservices

Small, independent services




Independent modules
Isolated points of failure
Microservices
Autonomous scalability
Faster value delivery
Tech flexibility

Azure Serverless benefits

Focus

Auto-scale based on workload
No infrastructure management
No wasted resources

Flexibility

Hosting options
Multiple languages
Development environments

Efficiency

Shorter time to market
Event-driven programming model
End-to-end dev experience

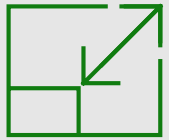
Why is it a great fit?

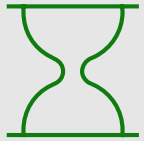
Problems of traditional microservices

1. Scaling of compute resources
2. Operations dependency
3. Pay per hosting nodes
4. Services discovery and managing services integration

Serverless solution

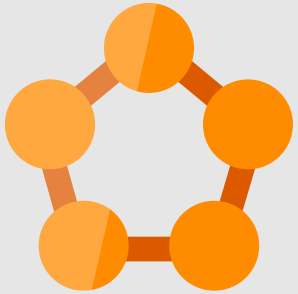
- Automatic scaling based on workload
- No infrastructure management
- Pay per execution
- Event-based programming model (triggers + bindings)



But nothing is a silver bullet…

Service Fabric
Designed for distributed apps



Azure Kubernetes Service
Fully managed orchestration

Azure Functions
Event-driven scenarios

COMPUTE
ENDPOINTS MANAGEMENT

API Management
API gateway and management

Our focus for the session

Service Fabric

Designed for distributed apps

Azure Kubernetes Service

Fully managed orchestration

Azure Functions

Event-driven scenarios

COMPUTE

ENDPOINTS MANAGEMENT

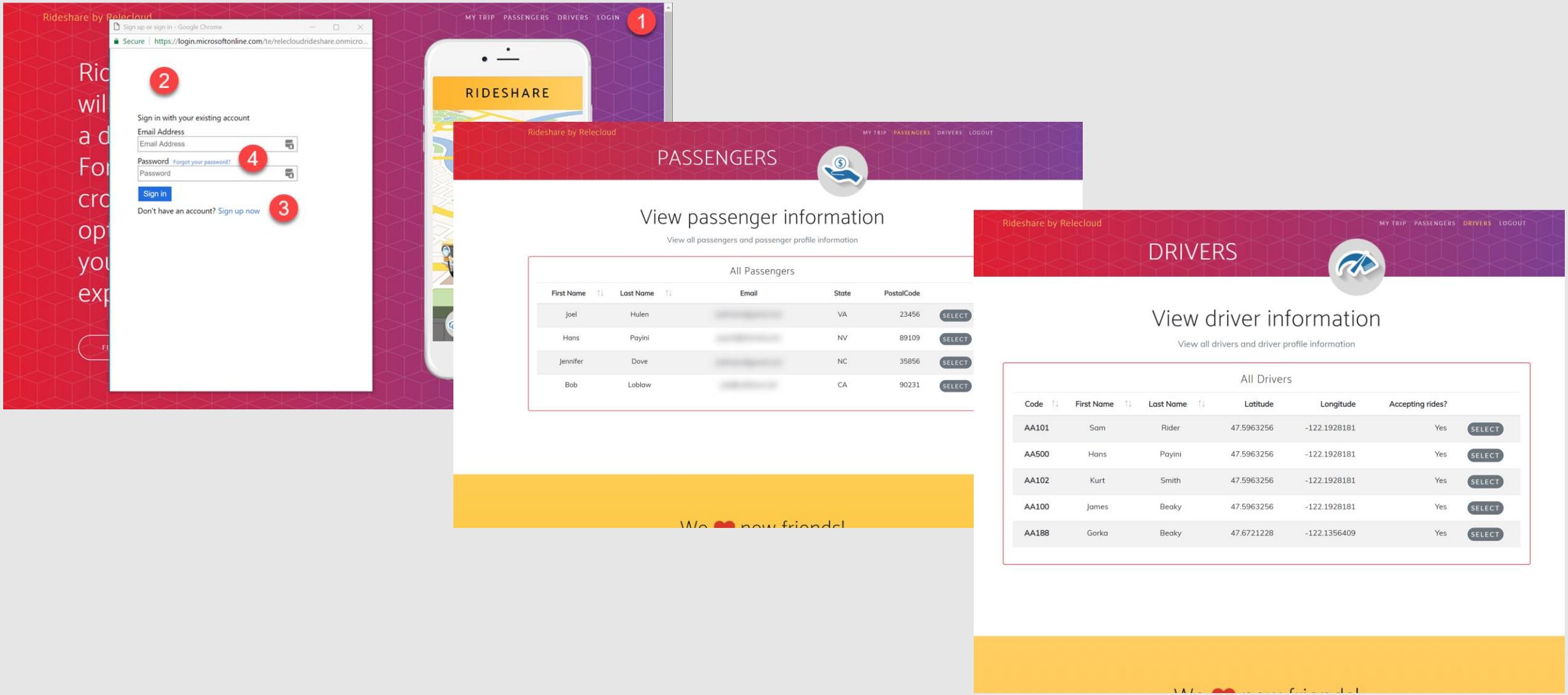
API Management

API gateway and management

Macro architecture

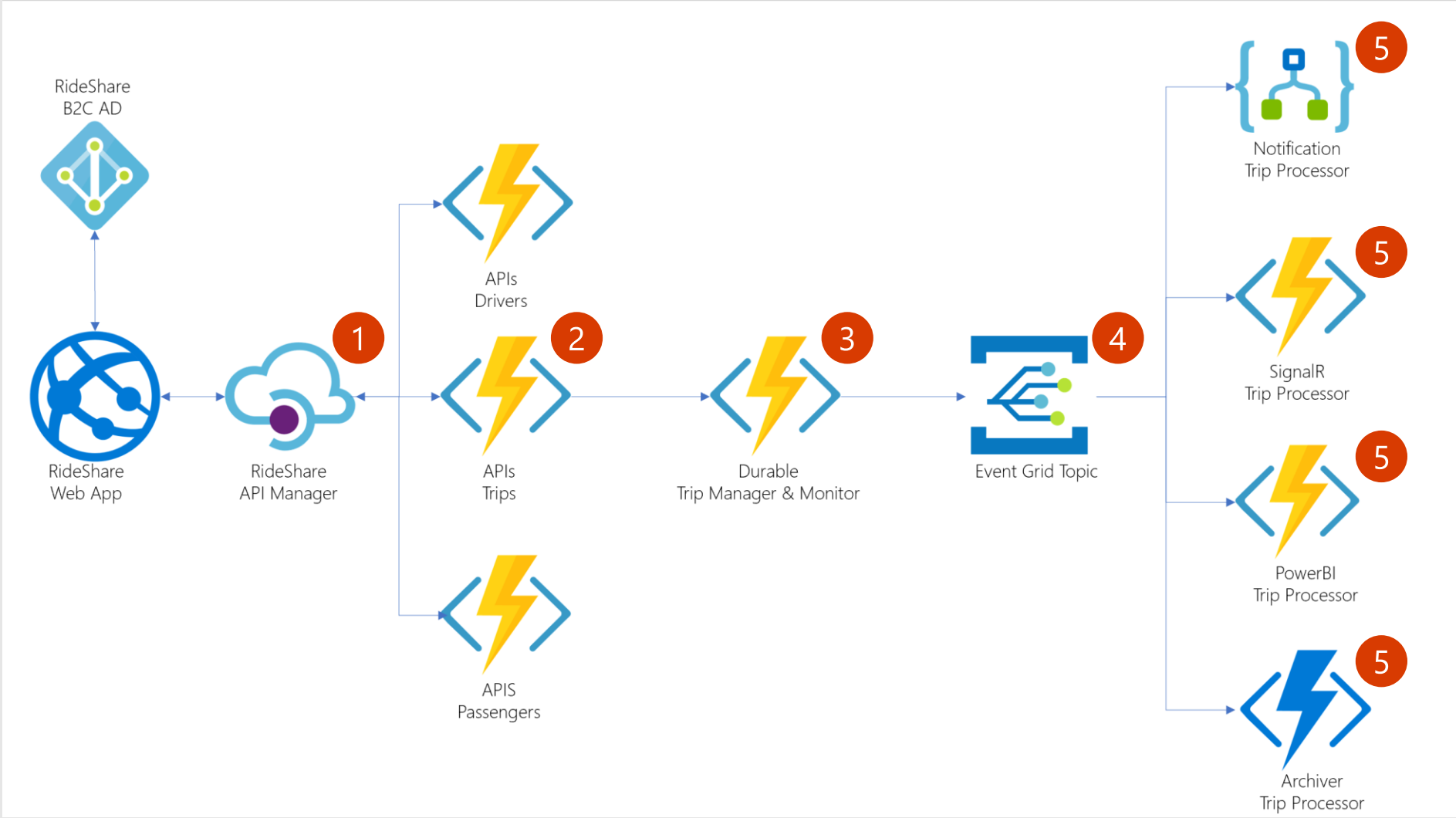
Our reference solution

Relecloud Rideshare



Our reference architecture

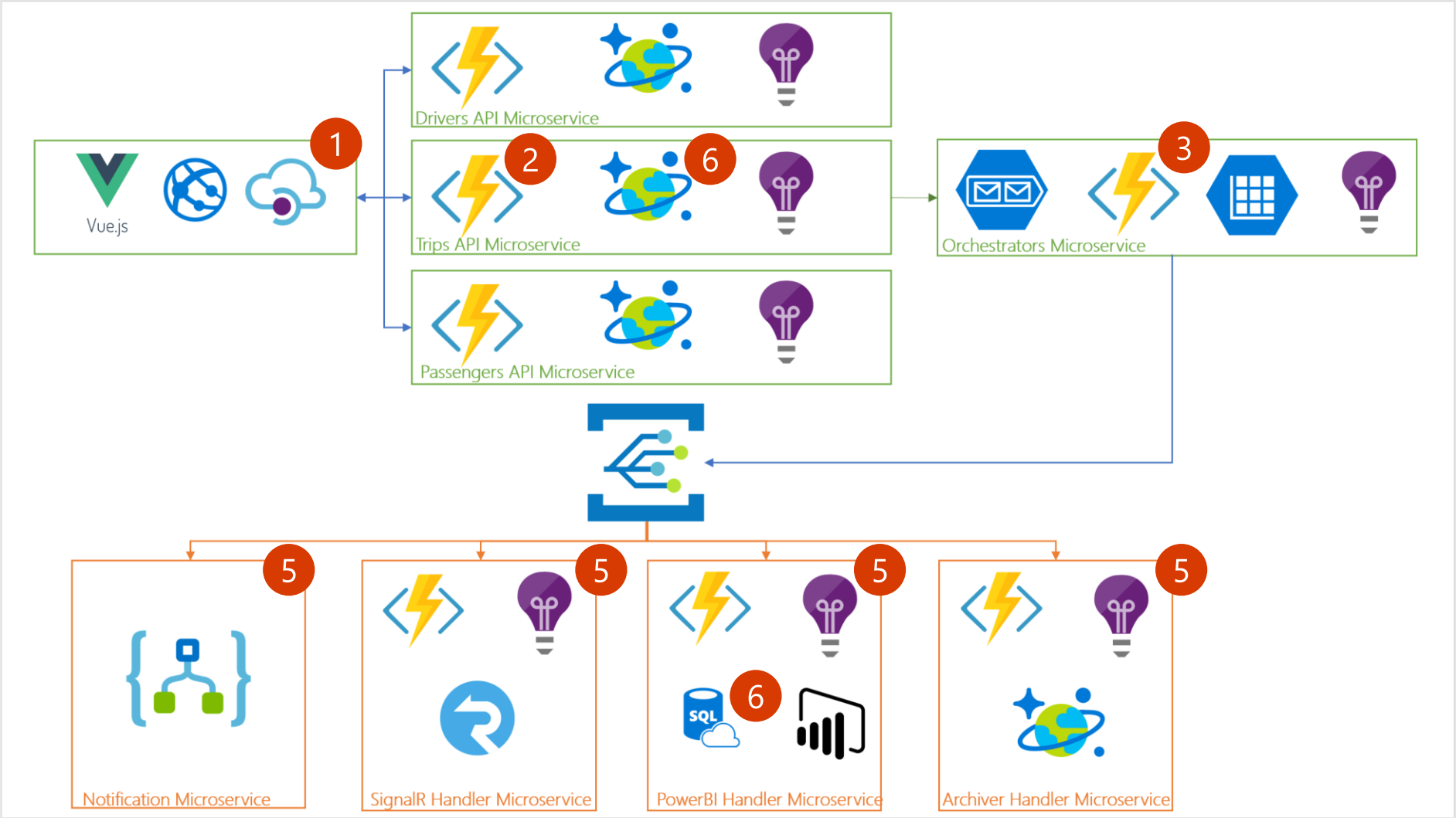
Relecloud Rideshare



1. Gateway
2. Entry point/s
3. Workflow orchestrator
4. Async queue
5. Backend services

Our reference architecture

Relecloud Rideshare



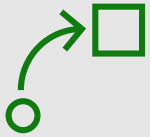
1. Gateway
2. Entry point/s
3. Workflow orchestrator
4. Async queue
5. Backend services
6. Data stores

Building the services

Microservices should be…

Small enough to take care of just one task

Smart endpoints and dumb pipes



Autonomously scalable

Independently deployed

FaaS is at the center of serverless

Functions-as-a-Service programming model use functions to achieve true serverless compute

Single responsibility

Functions are single-purposed, reusable pieces of code that process an input and return a result

Short lived

Functions don't stick around when finished executing, freeing up resources for further executions

Stateless

Functions don't hold any persistent state and don't rely on the state of any other processes

Event driven and scalable

Functions respond to predefined events, and are instantly replicated as many times as needed

Azure Functions = FaaS++

An event-based, serverless compute experience that accelerates app development

Integrated programming model

Use built-in triggers and bindings to define when a function is invoked and to what data it connects

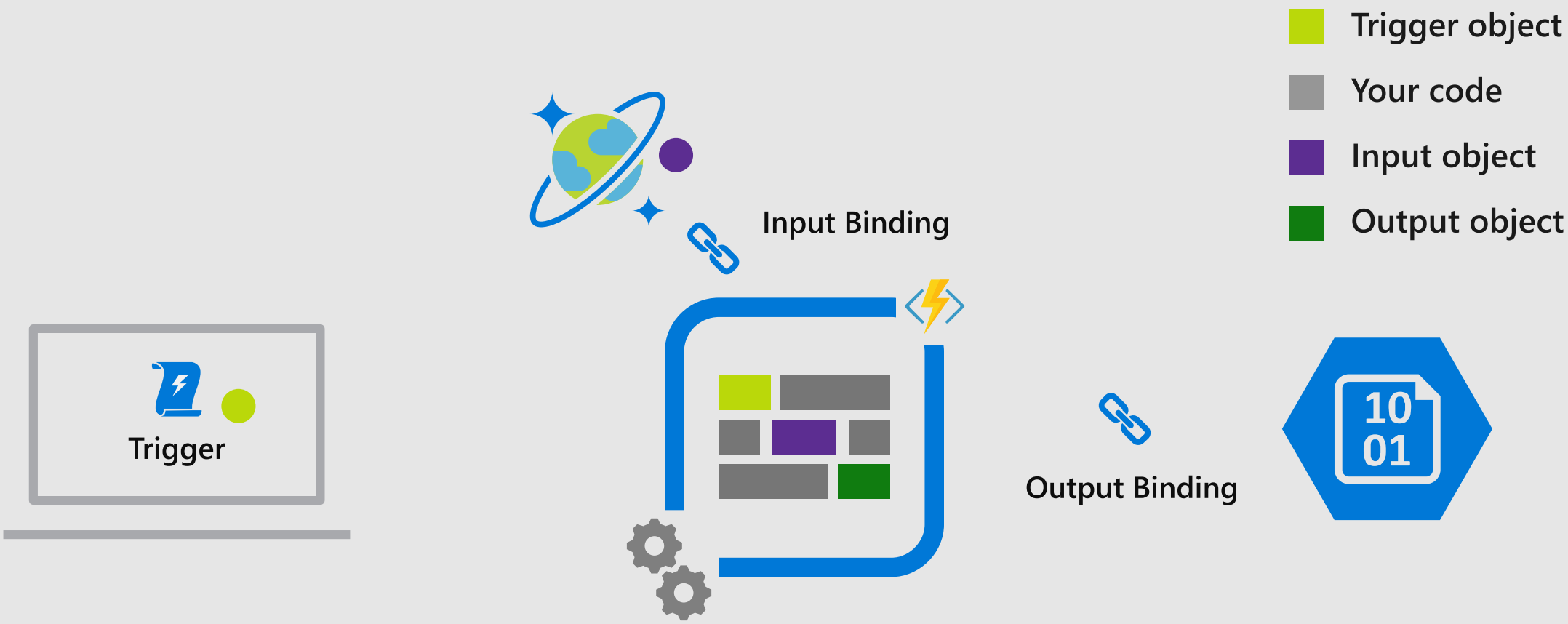
Enhanced development experience

Code, test and debug locally using your preferred editor or the easy-to-use web based interface including monitoring

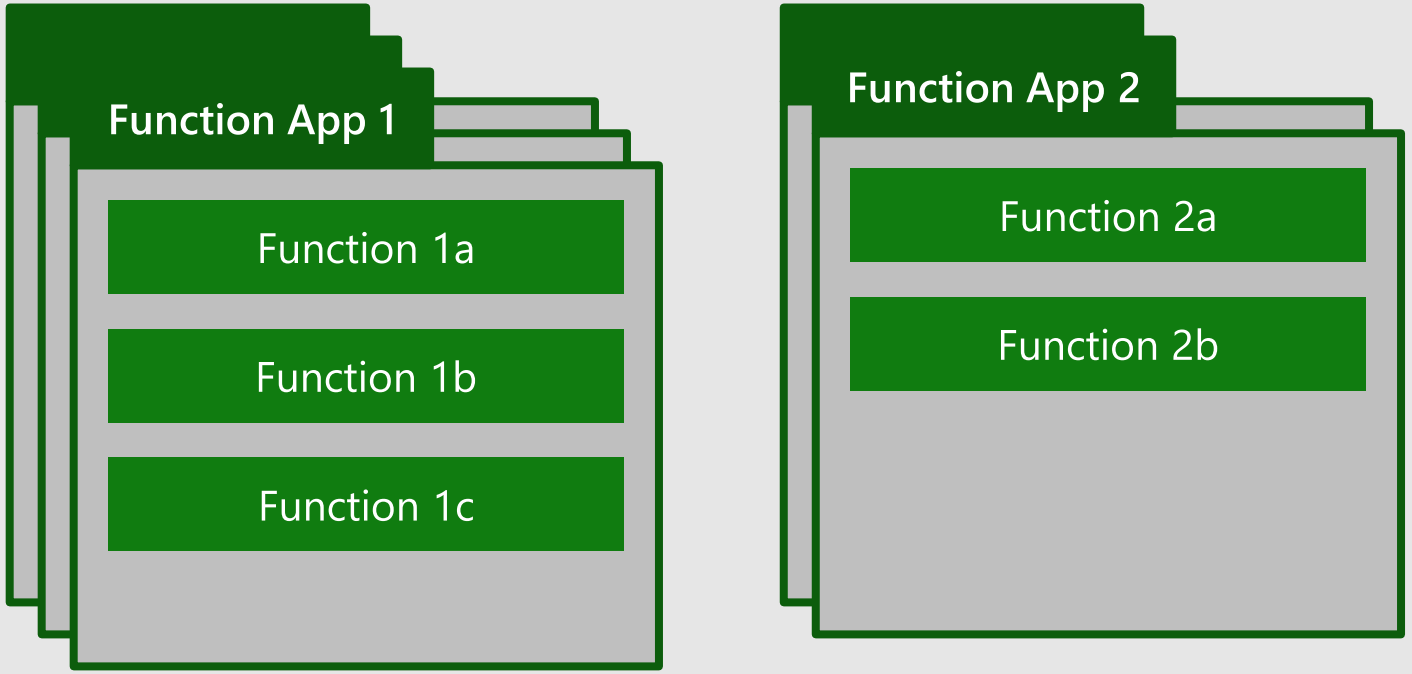
Hosting options flexibility

Choose the deployment model that better fits your business needs without compromising development experience

Productive programming model



Scaling happens at the Function App level



Usually, each Function App represents a different microservice

Each Function App would be an API with grouped endpoints (one function per endpoint)

Flexibility choosing the language for each microservice (one per Function App)

Language options

Generally available

Public preview

More on the way!

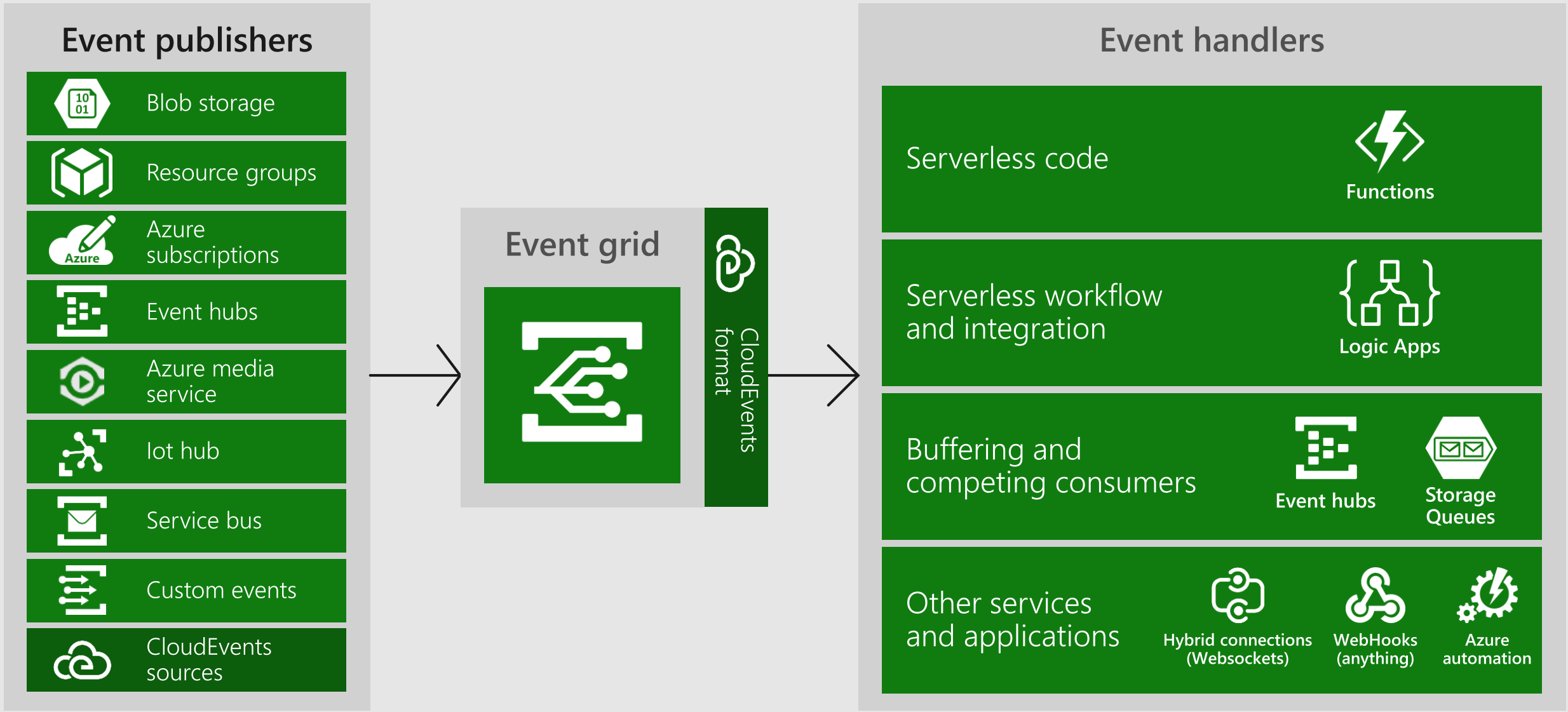
Services intercommunication

Azure Event Grid

Simplify app development with an event based pub-sub model

- Simple HTTP-based event delivery
- Build better, more reliable applications through reactive programming
- 1st party services or custom events

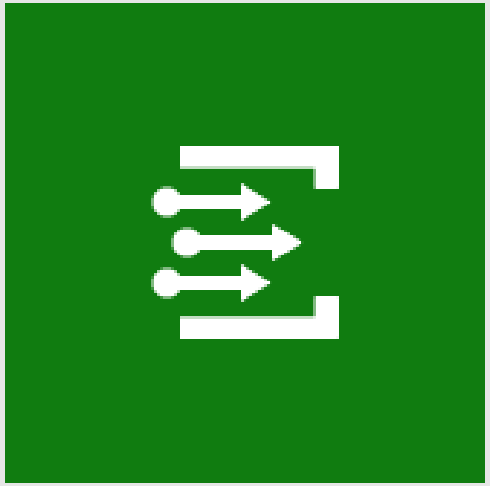
Eventing in the cloud



React to two kind of events

First-party events

Save coding time by subscribing to events triggered by other Azure services



Custom events

Create your own events using the data payload attribute to share the required information

Custom event schema

```
{
  "id": "f8d88d00-1a8b-4a1e-af46-ba29b19087ab",
  "subject": "BFYOC/stores/serverlessWorkshop/orders",
  "data": {
    "orderId": "4",
    "itemOrdered": "1",
    "email": "gorkma@microsoft.com"
  },
  "eventType": "BFYOC.IceCream.Order",
  "eventTime": "2018-09-24T02:35:54.387Z",
  "dataVersion": "2.0",
  "metadataVersion": "1",
  "topic": "/subscriptions/just-a-bunch-of-characters/resourceGroups/
       bfyocgorkma/providers/Microsoft.EventGrid/topics/BYOCgorkmaOrders"
}
```

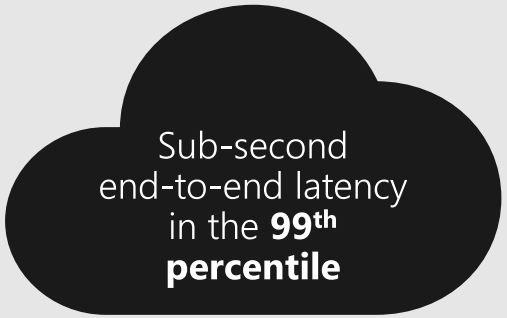
Publishing to a custom topic

```
let topicCreds = new msRestAzure.TopicCredentials(topicKey);
let egClient = new eventGrid(topicCreds);
let topicUrl = url.parse(topicEndPoint, true);
let topicHostName = topicUrl.host;
let currentDate = new Date();

let events = [
        {
                id: uuid(),
                subject: 'BFYOC/stores/serverlessWorkshop/orders’,
                dataVersion: '2.0’,
                eventType: 'BFYOC.IceCream.Order’,
                data: req.body,
                eventTime: currentDate
        }
];
egClient.publishEvents(topicHostName, events);
```

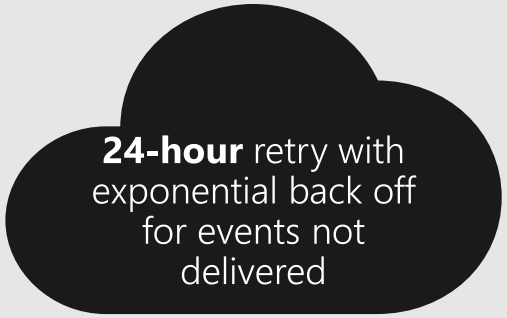
Event Grid guiding principles

- Always available
- Near real-time event delivery
- At least once delivery
- Dynamic scale
- Platform agnostic (WebHook)
- Language agnostic (HTTP protocol)
- Fan-out (multiple subscribers)



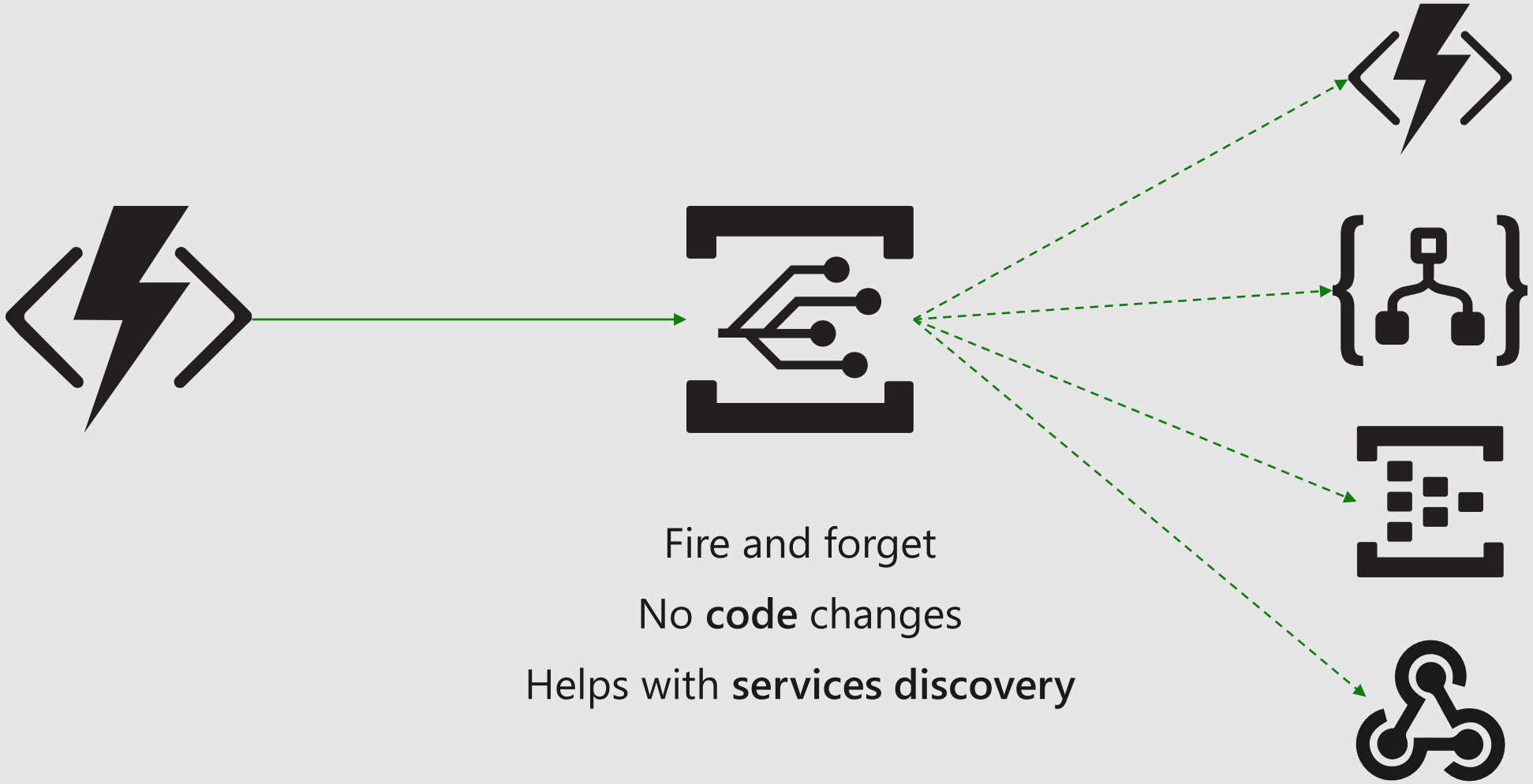
Near real-time

Massive scale-out



High reliability

Event driven communication = decoupled services



Fire and forget

No **code** changes

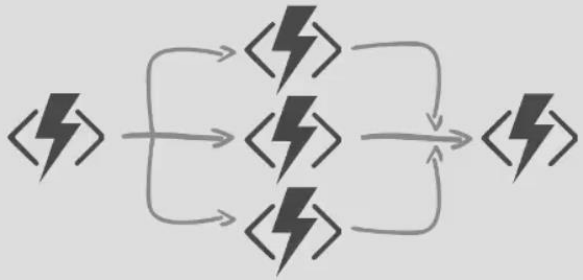
Helps with **services discovery**

Stateless microservices 

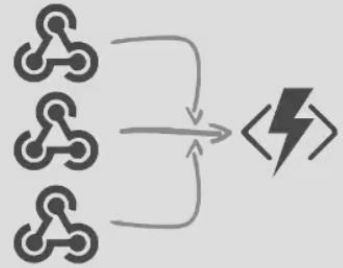
What about stateful operations?

Not everything fits on a few-seconds execution

Manageable sequencing
+ error handling/compensation

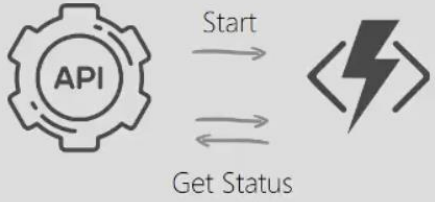


Fanning-out and fanning in



External events correlation

Flexible automated long-running
process monitoring



Http-based
Async long-running APIs

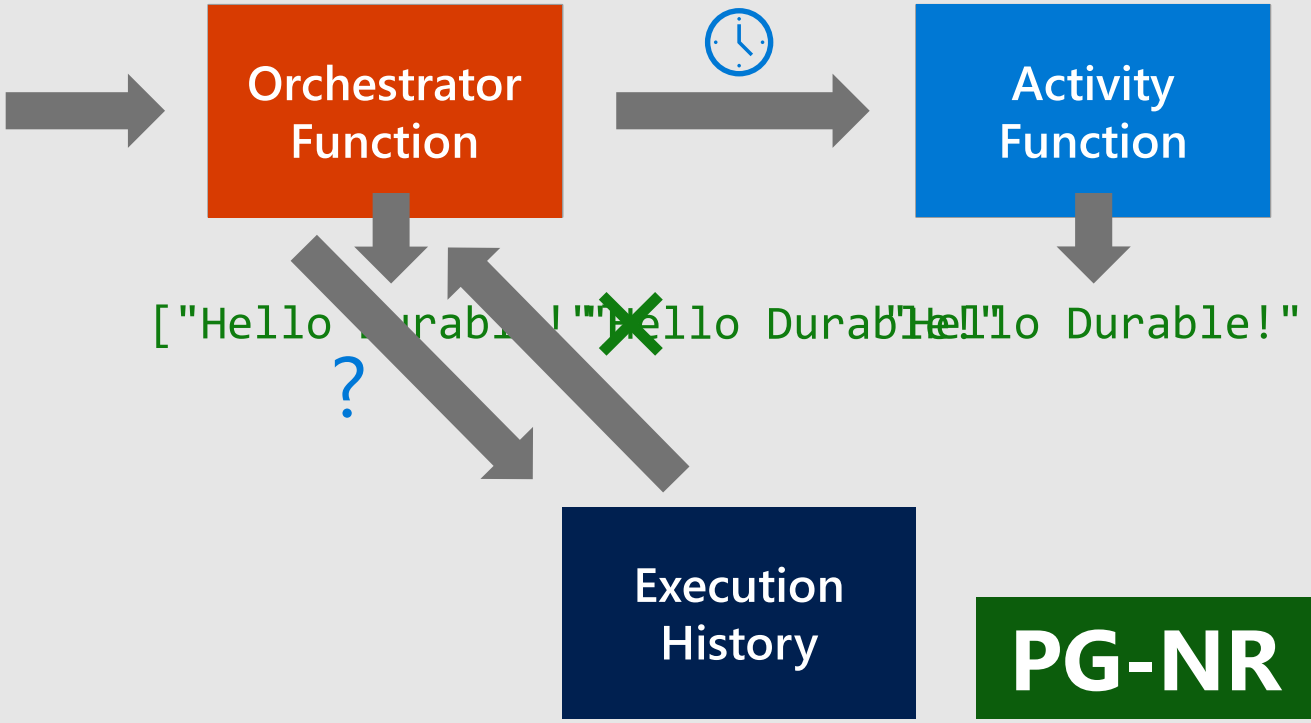
Human interaction

Orchestration with durable functions

```
var outputs = new List<string>();

outputs.Add(await context.CallActivityAsync<string>("SayHello", "Durable!"));

return outputs;
```



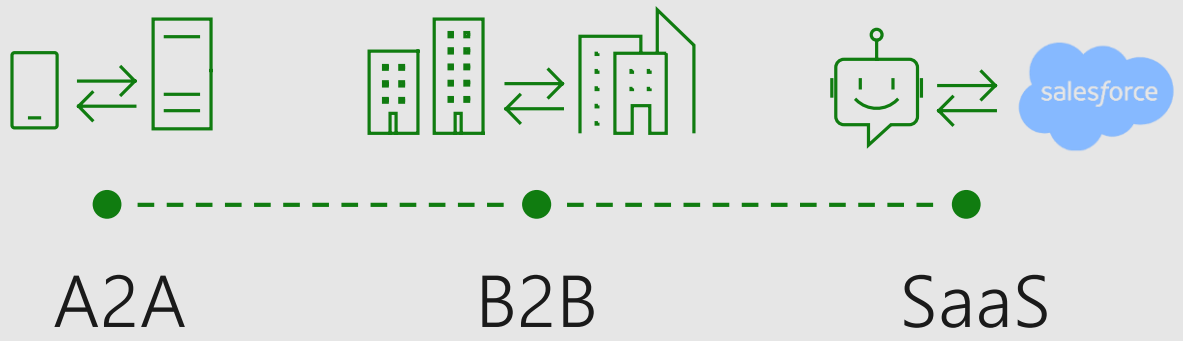
History Table

Orchestrator Started
Execution Started
Task Scheduled, SayHello, "Durable!"
Orchestrator Completed
Task Completed, "Hello Durable!"
Orchestrator Started
Execution Completed, ["Hello Durable!"]
Orchestrator Completed

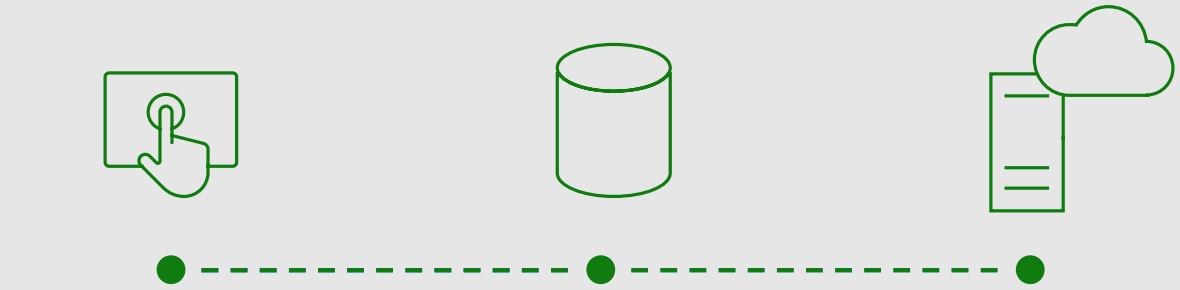
PG-NR FOR EDUCATION PURPOSES ONLY, NEVER DO A "HELLO WORLD" WITH DURABLE FUNCTIONS

Workflows orchestration

Solutions aren’t isolated



Integration challenges



Interfaces and APIs

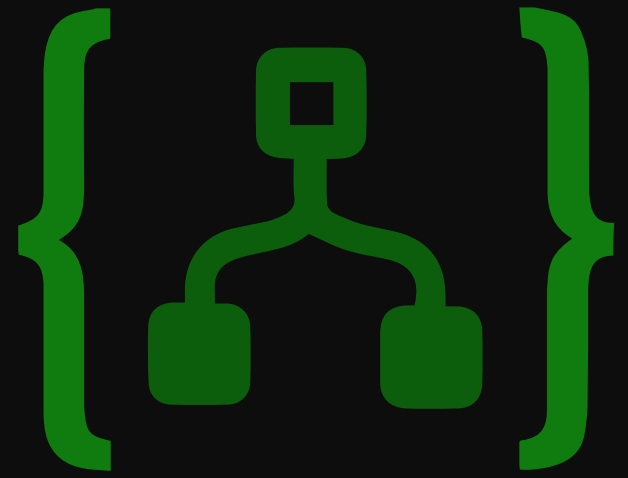
Data sources and formats

On premises or cloud

Azure Logic Apps

Easily automate workflows and orchestrate business processes

- Out-of-the-box connectors reduce integration challenges
- Connect and integrate data from cloud and on-premises systems
- B2B and enterprise messaging in the cloud
- Web-based graphic workflow designer



Out-of-box connectors

SAAS

- AppFigures
- Azure DevOps
- Azure Machine Learning
- Azure Service Bus
- Azure Storage Blob
- BaseCamp
- Bing Search
- Blogger
- Box
- Common Data Model
- Dropbox
- Dynamic AX Online
- Dynamics CRM Online
- Facebook
- GitHub
- Google Calendar
- Google Drive
- Google Sheets
- Google Tasks
- Insightly
- Instagram
- MailChimp
- Mandrill
- Microsoft Project Online
- Microsoft Translator
- Office 365
- Office 365 Users
- OneDrive
- OneDrive for Business
- Outlook.com
- PagerDuty
- Pinterest
- Project Online
- Salesforce
- SendGrid
- SharePoint Online
- Slack
- SmartSheet
- SparkPost
- SQL Database
- Todoist
- Trello
- Twilio
- Twitter
- Wunderlist
- Yammer
- YouTube

Protocols

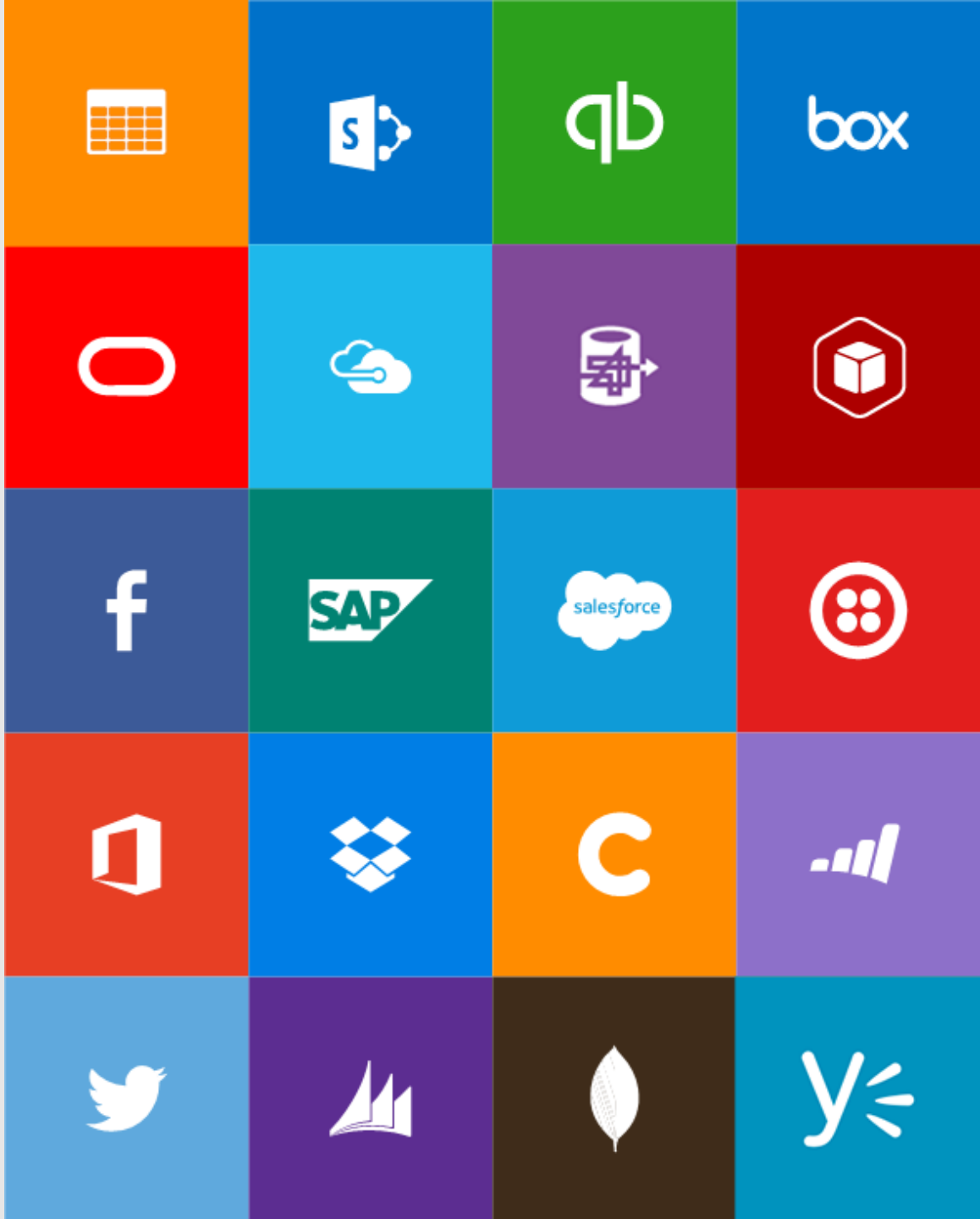
- HTTP, HTTPS
- HTTP Webhook
- FTP, SFTP
- SMTP
- RSS
- Delay
- Workflow

Enterprise messaging

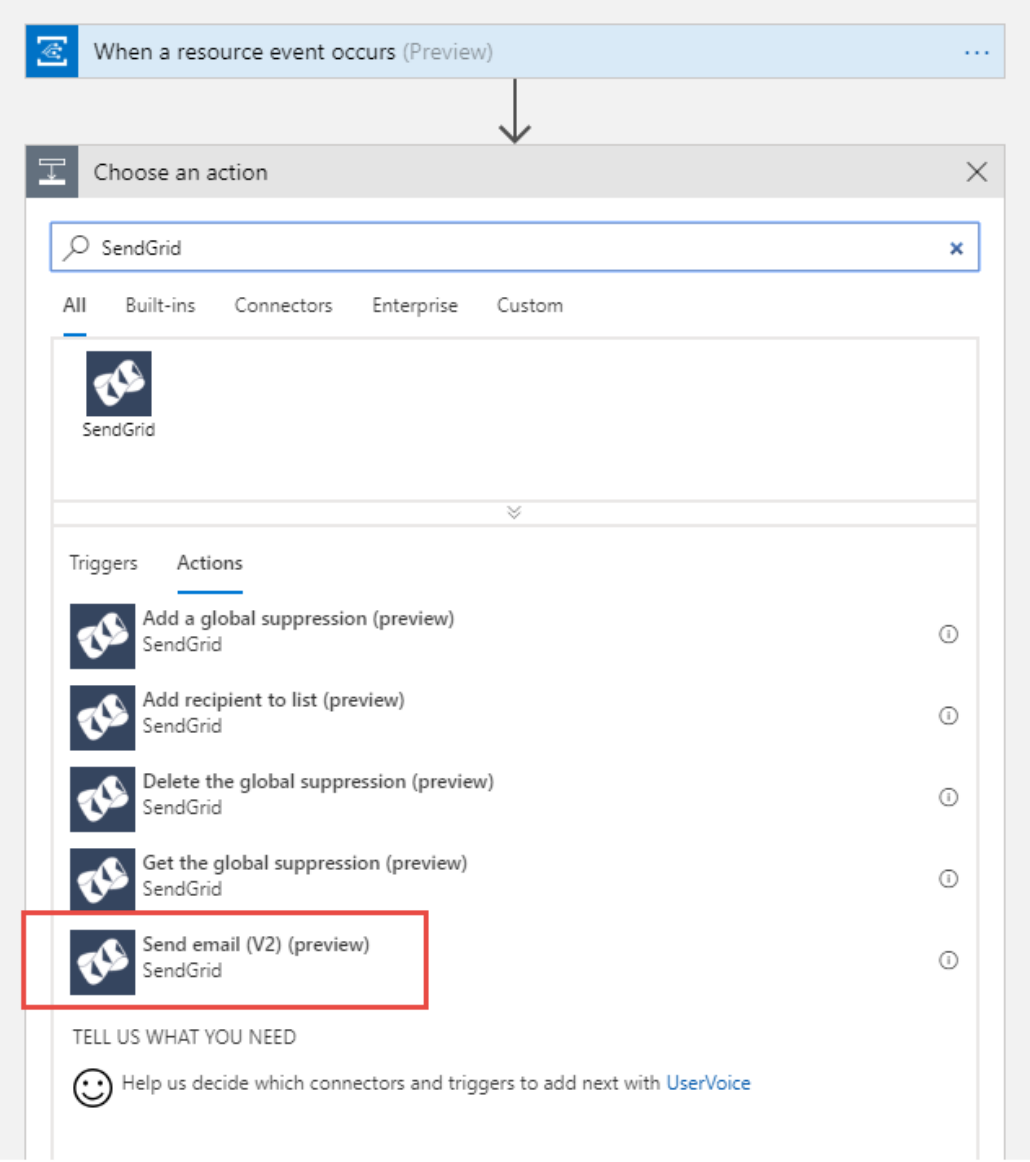
- Validate
- Transform (+Mapper)
- Convert (XML-FF)
- X12
- AS2
- EDIFACT

Hybrid

- DB2
- SQL Server
- Informix
- SharePoint Server
- BizTalk Server
- WebSphere MQ



Serverless workflows with Logic Apps



When to use what?

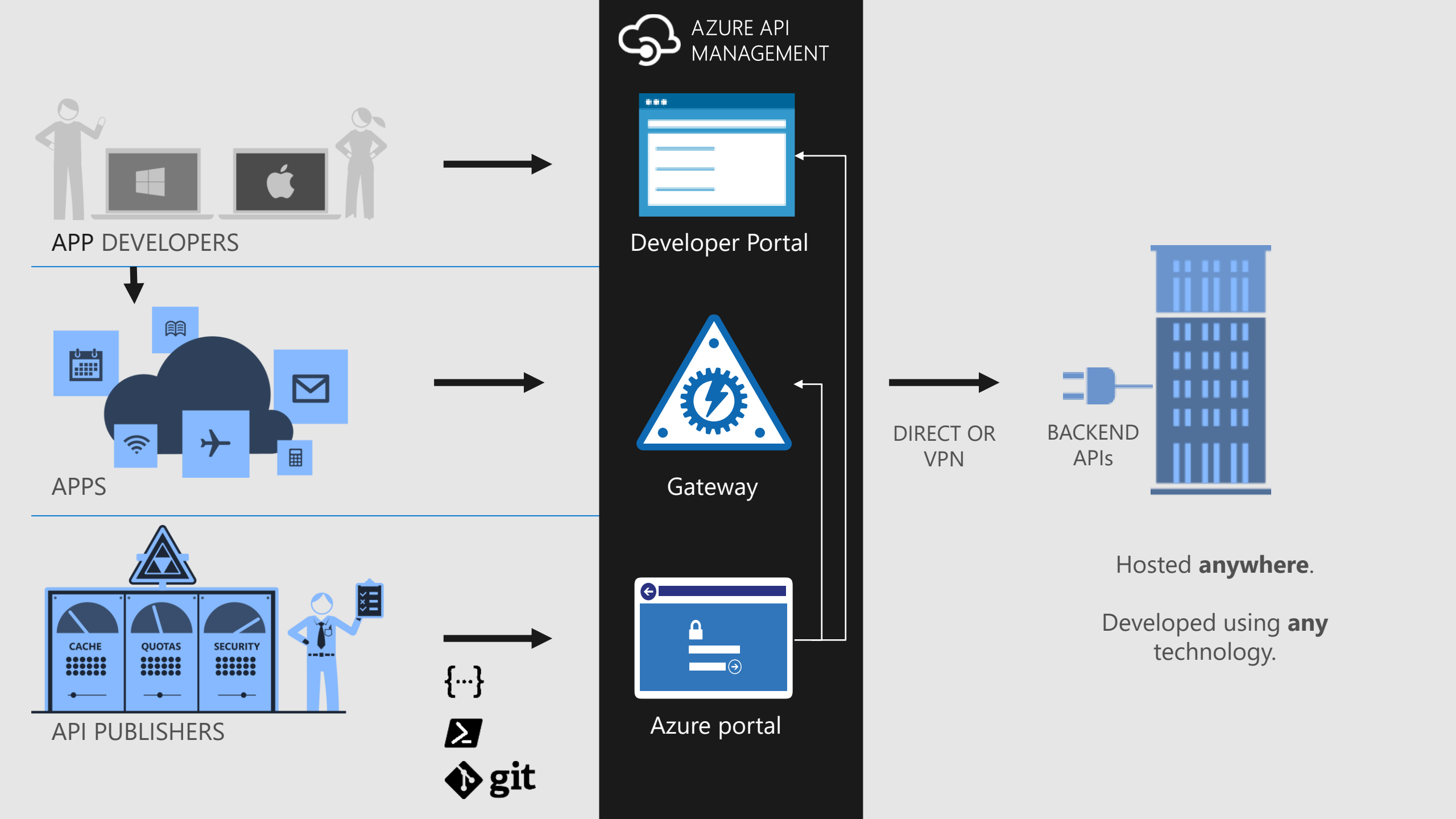
	Durable functions	Logic Apps
Development	Code-first (imperative)	Designer-first (declarative)
Connectivity	About a dozen built-in binding types, write code for custom bindings	Large collection of connectors, Enterprise Integration Pack for B2B scenarios, build custom connectors
Actions	Each activity is an Azure function; write code for activity functions	Large collection of ready-made actions
Monitoring	Azure Application Insights	Azure portal, Azure Monitor, Log Analytics
Management	REST API, Visual Studio	Azure portal, REST API, PowerShell, Visual Studio
Execution context	Can run locally or in the cloud	Runs only in the cloud

Exposing the endpoints

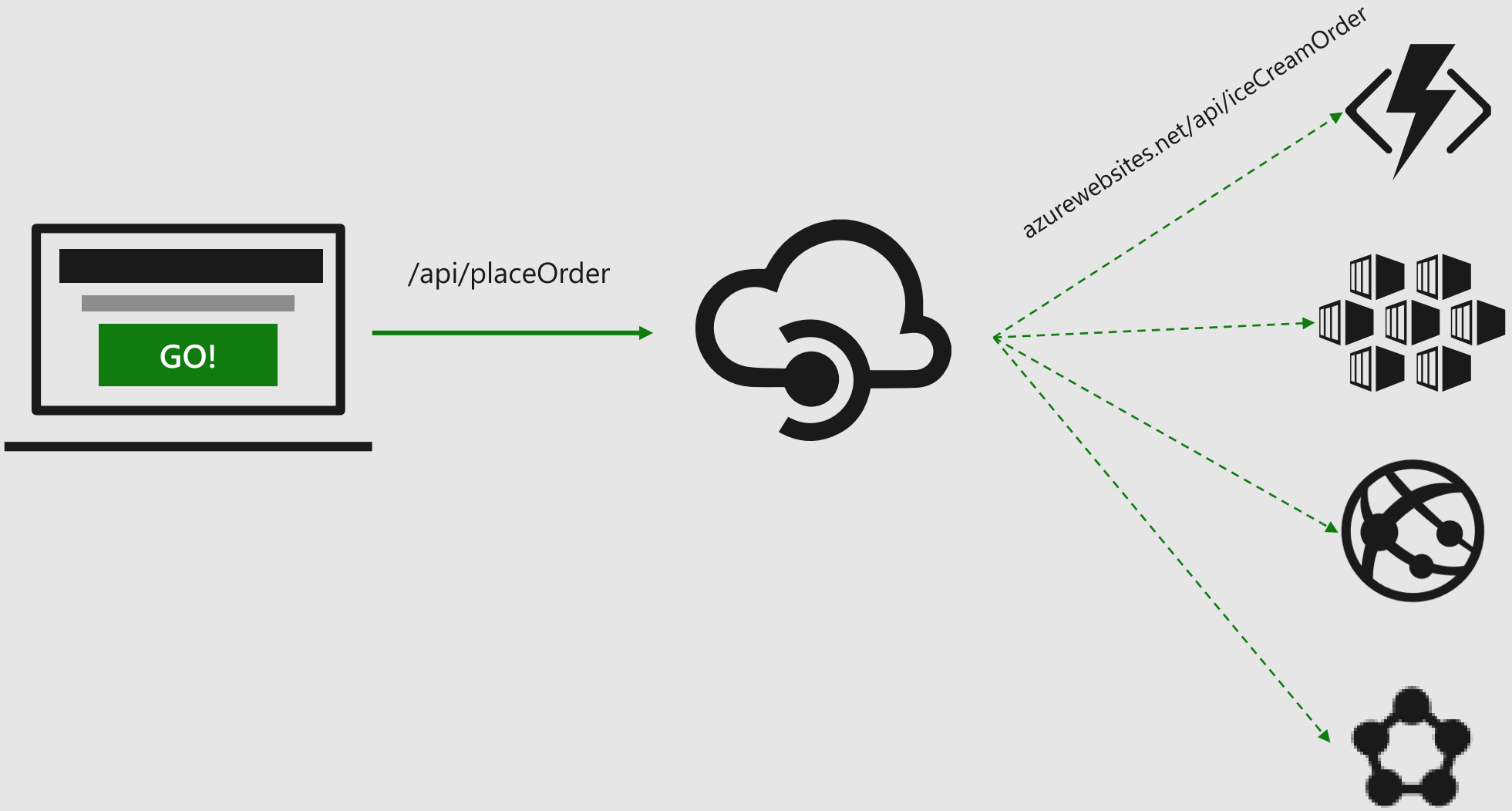
Azure API Management

Publish APIs safely and connect to backend systems hosted anywhere

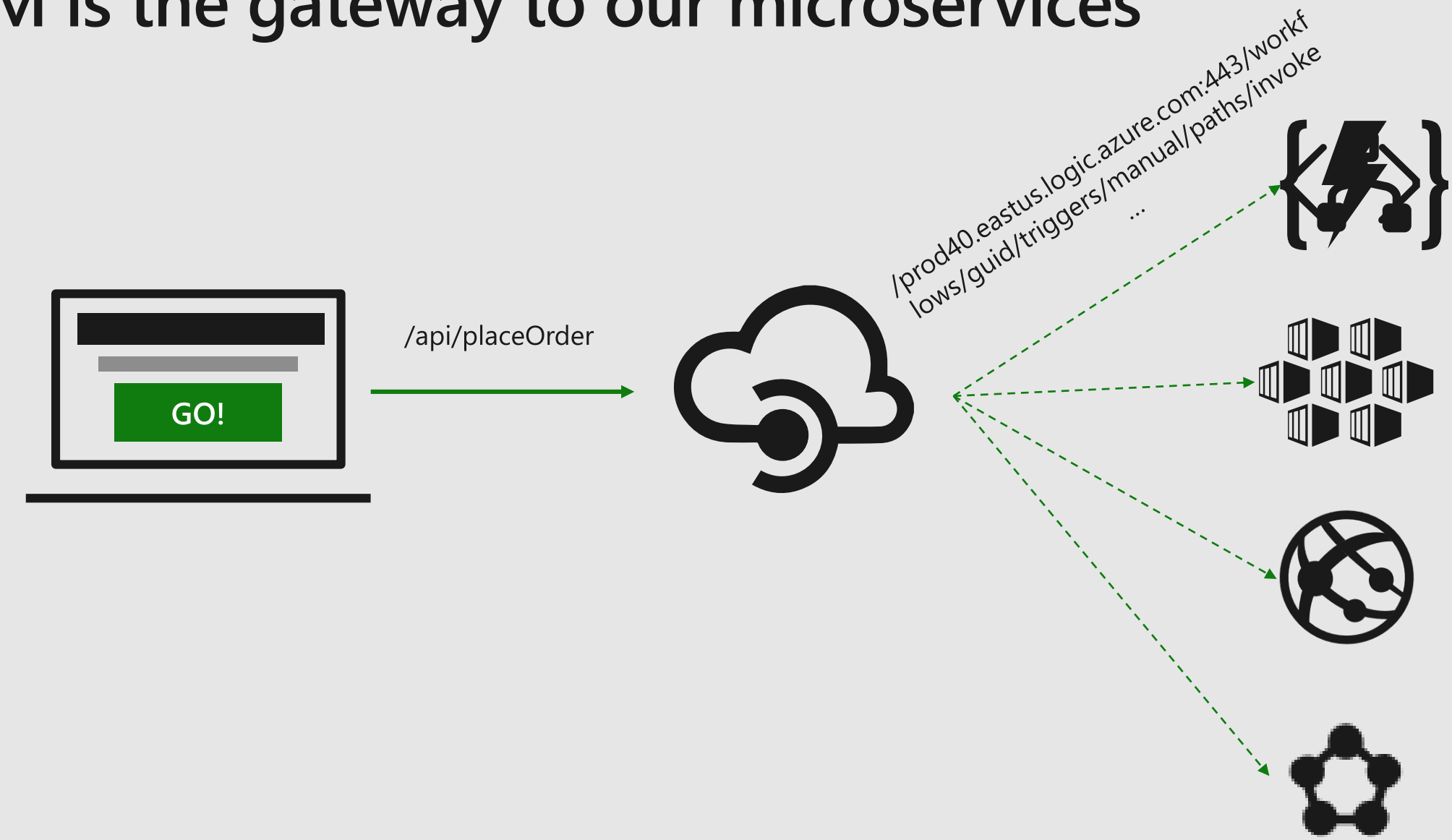
- Centralize all operations over your APIs
- Redirect to your endpoints hosted on multiple services
- Manage security layers, versioning, testing, mocking…
- Monitor your APIs and gain insights


AZURE API MANAGEMENT
APP DEVELOPERS
Developer Portal
APPS
Gateway
DIRECT OR VPN
BACKEND APIs
CACHE
QUOTAS
SECURITY
API PUBLISHERS
Azure portal
git
Hosted anywhere.
Developed using any technology.

APIM is the gateway to our microservices



APIM is the gateway to our microservices



[NEW] API Management tier

“Consumption”

API management layer for event-driven architectures

- Instant provisioning
- Automated scaling—out and back to zero
- Built-in high availability
- Per action pricing

IN PREVIEW

And we can go on and on...

Additional considerations

Monitoring and correlation

Unit and integration tests

DevOps and deployment pipelines

Coding/time savings vs what needs to be done manually

Wrapping up...

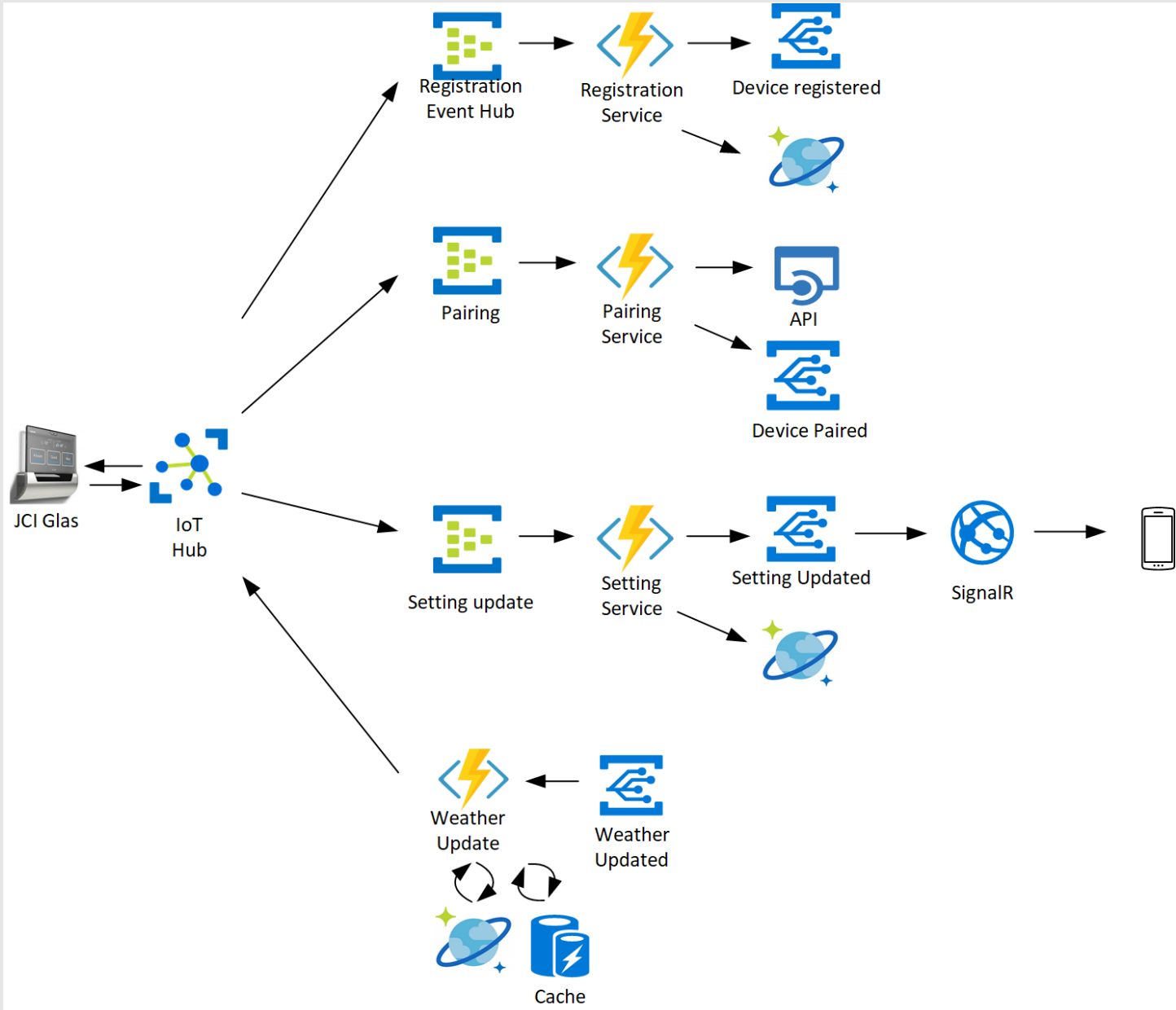
Not just theory

GLAS® Smart Thermostat by Johnson Controls

Microservices architecture based on Functions and managed services

Find out more at Microsoft Ignite 2018 session: **BRK2202**

Serverless real use cases and best practices



Resources

1. Relecloud RideShare—https://aka.ms/serverless-sample-relecloud
2. Content Reactor—http://aka.ms/serverless-microservices-sample
3. Microservices in Azure—http://aka.ms/azure-microservices
4. Azure Serverless platform—http://aka.ms/serverless-azure
5. Create Serverless applications: learning path
   —https://docs.microsoft.com/en-us/learn/paths/create-serverless-applications/index

Grazie!

- Il materiale sarà online nei prossimi giorni su http://www.communitydays.it

Lorenzo Barbieri

Cloud Solutions Architect

lorenzo.Barbieri@microsoft.com

linkedin.com/in/geniodelmale

CommunityDays.it