Electronics and
Computer Science

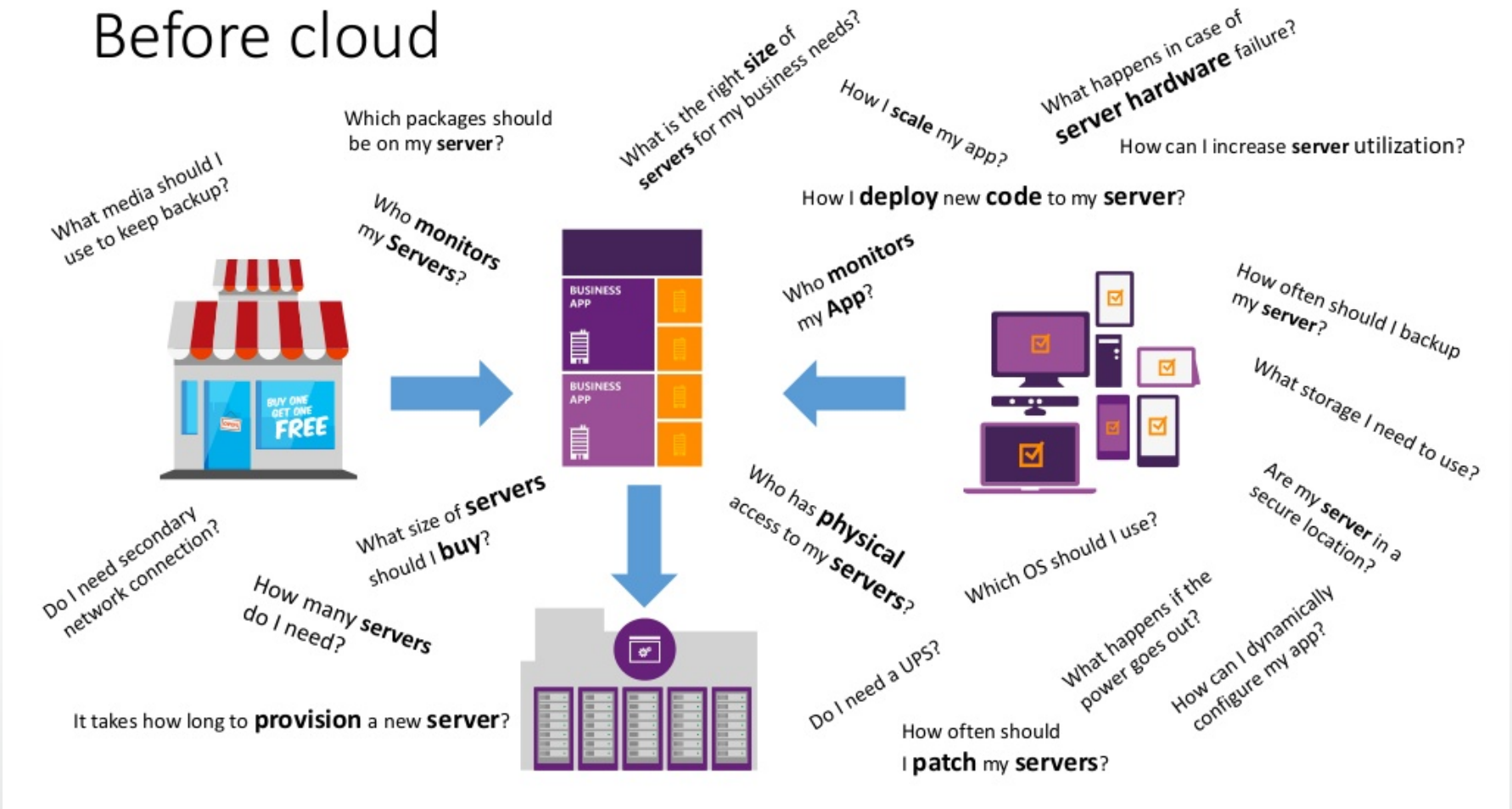
UNIVERSITY OF
Southampton

Cloud Computing
Serverless Architectures

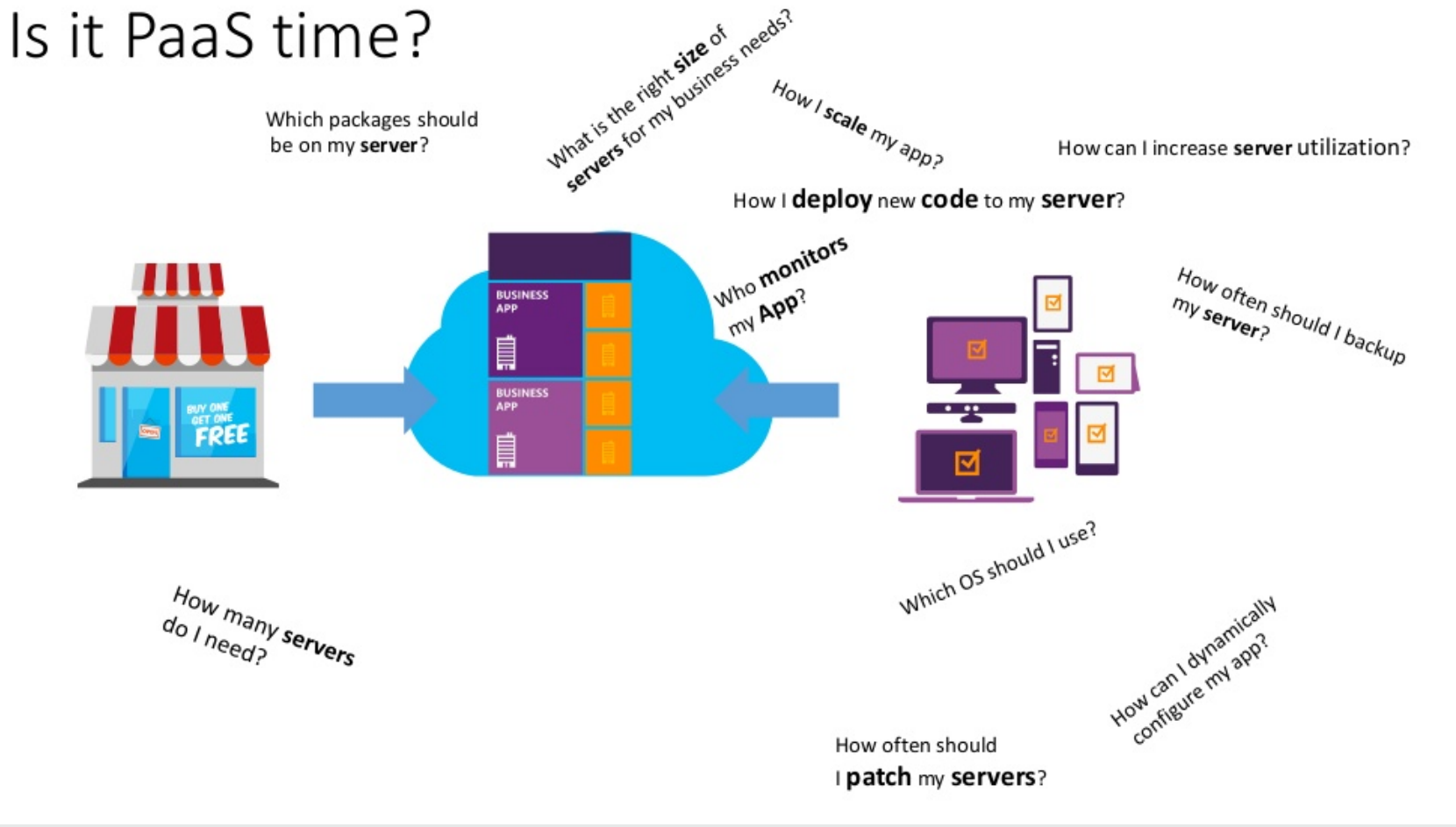
COMP3207/COMP6244

Dr A. Rezazadeh
October 19

What is Serverless?



What is Serverless?



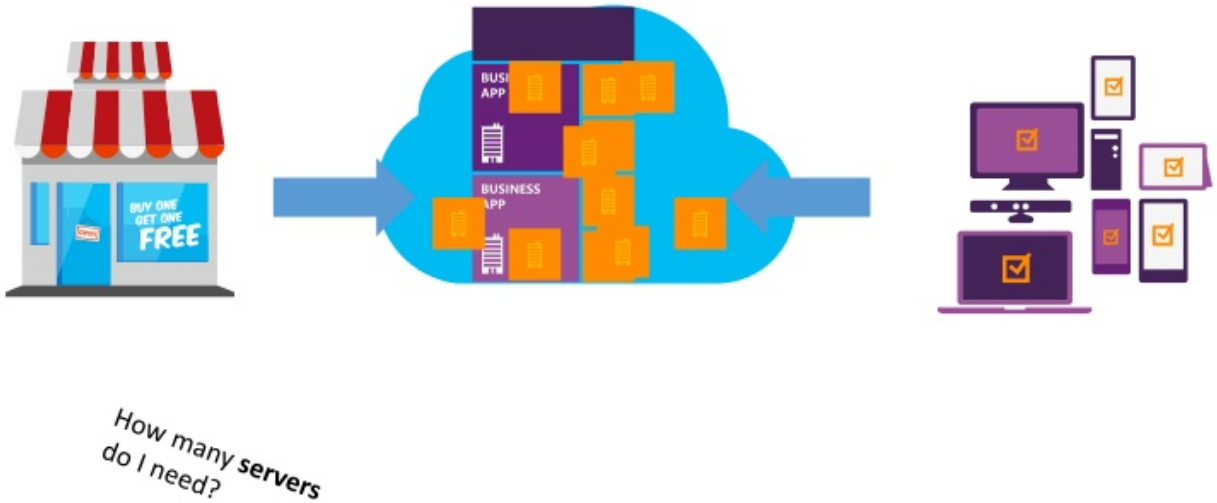
What is Serverless?

Serverless . . .

What is the right **size** of **servers** for my business needs?

How I **scale** my app?

How can I increase **server** utilization?



How many **servers** do I need?

What is Serverless?

Abstraction of servers

Event-driven/ instant scale

Sub-second billing

What is Serverless?

A **cloud-native** platform

for

short-running, **stateless** computation

and

event-driven applications

which

scales up and **down instantly** and **automatically**

and

charges for actual usage at a **millisecond granularity**

What Dose Serverless Mean?

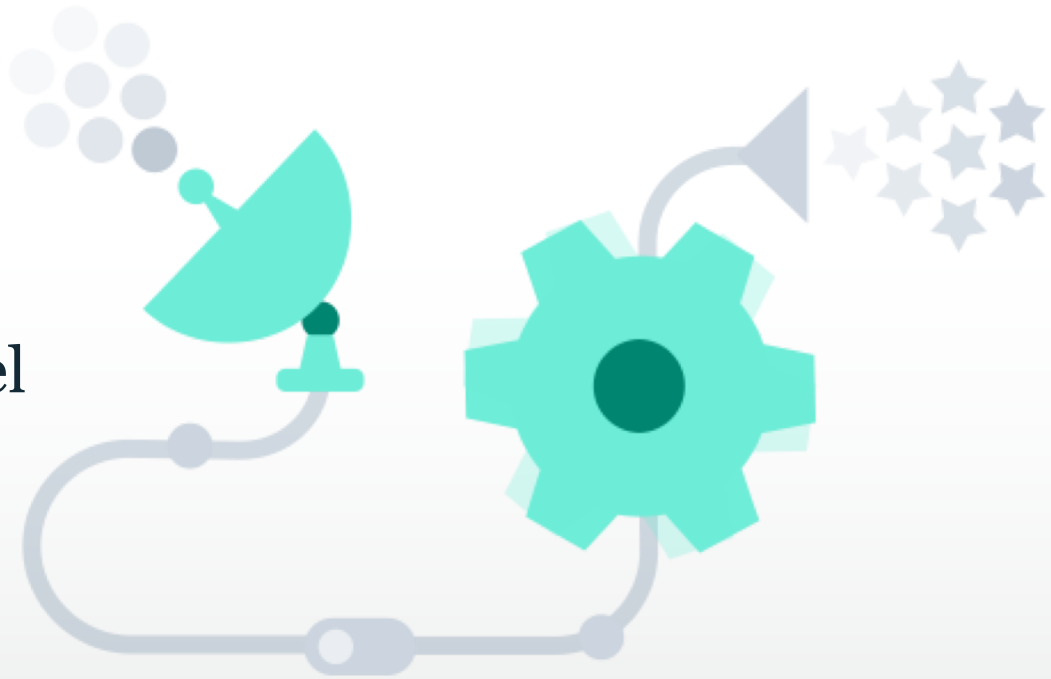
- Server-less means no servers?
- Or worry-less about servers?
  - Runs code **only** on-demand on a per-request basis
  - Serverless deployment & operations model

No servers

Just code

What triggers code execution?

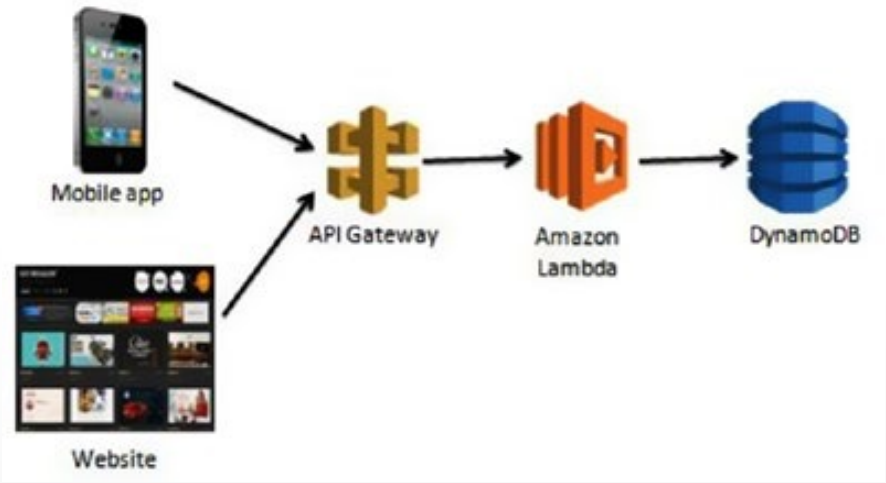
Runs code **in response** to events



Event-programming model

Event-driven Architecture

- *A request is made to the API Gateway, which then triggers the Lambda function for a response.*



- The functions are event consumers because they are expected to come alive when an event occurs and are responsible for processing it.

Examples of Events

- Examples of events that trigger serverless functions include these:
  - API requests
  - Object puts and retrievals in object storage
  - Changes to database items
  - Scheduled events
  - Voice commands (for example, Amazon Alexa)
  - Bots (such as AWS Lex and Azure LUIS, both natural-language–processing engines)

Why is Serverless attractive?

- Making app development dramatically **faster**, **cheaper**, **easier**
- Drives infrastructure cost savings

	On-prem	VMs	Containers	Serverless
Time to provision	Weeks-months	Minutes	Seconds-Minutes	Milliseconds
Utilization	Low	High	Higher	Highest
Charging granularity	CapEx	Hours	Minutes	Blocks of milliseconds

Serverless Service Models

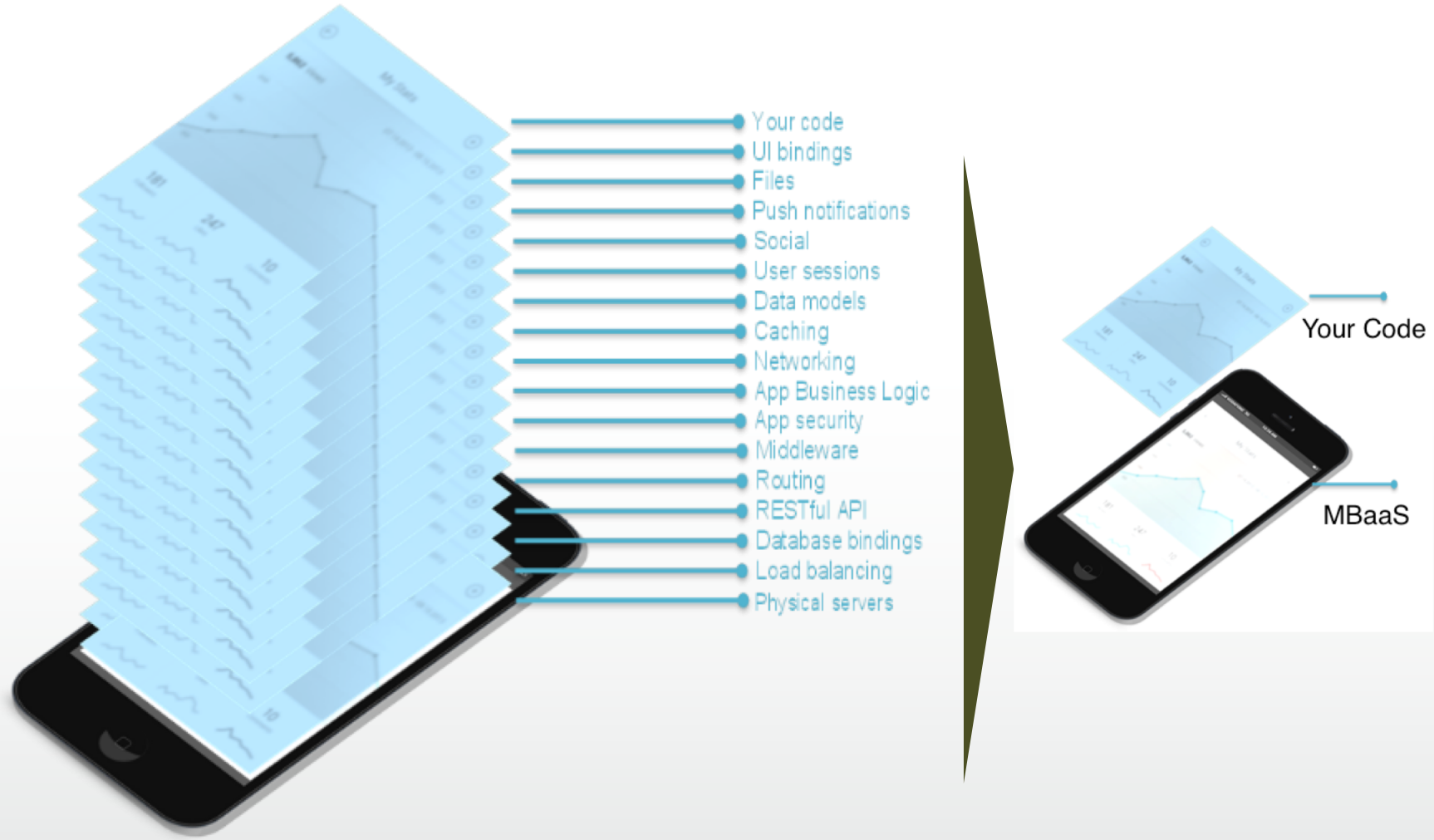
- **Serverless** architectures refer to applications that significantly depend on
  - third-party services, known as **Backend as a Service** (**BaaS**) or
  - on custom code that's run in **ephemeral** containers, known as **Function as a Service** (FaaS)

- By using these ideas, and by moving much behavior to the front end, such architectures remove the need for the traditional **'always on'** server system sitting behind an application.

Backend as a Service/MBaaS

- BaaS provide **web** and **mobile app** developers with a way to **connect** their applications to **backend cloud storage and processing**
  - Providing **common features** such as **user management**, **push notifications**, **social networking integration**.
  - Each of these services has its own **API** that can be individually **incorporated** into an app.
  - Providing a consistent way to manage **backend data** means that developers potentially saving both **time** and **money**.

MBaaS – Allows you to Focus on the Frontend!



MBaaS Advantages for Developer

- **Efficiency gains**: reducing the development cost, development time and maintenance cost
- **Faster Time To Market:** reduce obstacles from idea to production and operations overhead.
- **Optimized for Mobile and Tablets:** optimization of data and network for mobile apps, and lower fragmentation problems across multiple platforms and devices.
- **Secure & Scalable:** bundled infrastructure that deals with scalability, security, performance and other operational headaches.
- **Handle App Growth & Maintenance:** brings common and essential 3rd party API resources into a single stack, preventing developers from having to go gather them separately.

Some Examples of MBaas

- Firebase (Google)
- Parse
- Heroku
- PythonAnywhere
- Rackspace Cloud
- BaasBox (Open Source)
- Usergrid (Open Source)

Firebase

- Documentation:

  https://firebase.google.com/docs/?authuser=0

- Capabilities:
  - User authorization
  - Database storage
  - Storage for larger files
  - Cloud messaging
  - Push notifications
  - Analytics
  - Hosting of web content
  - …

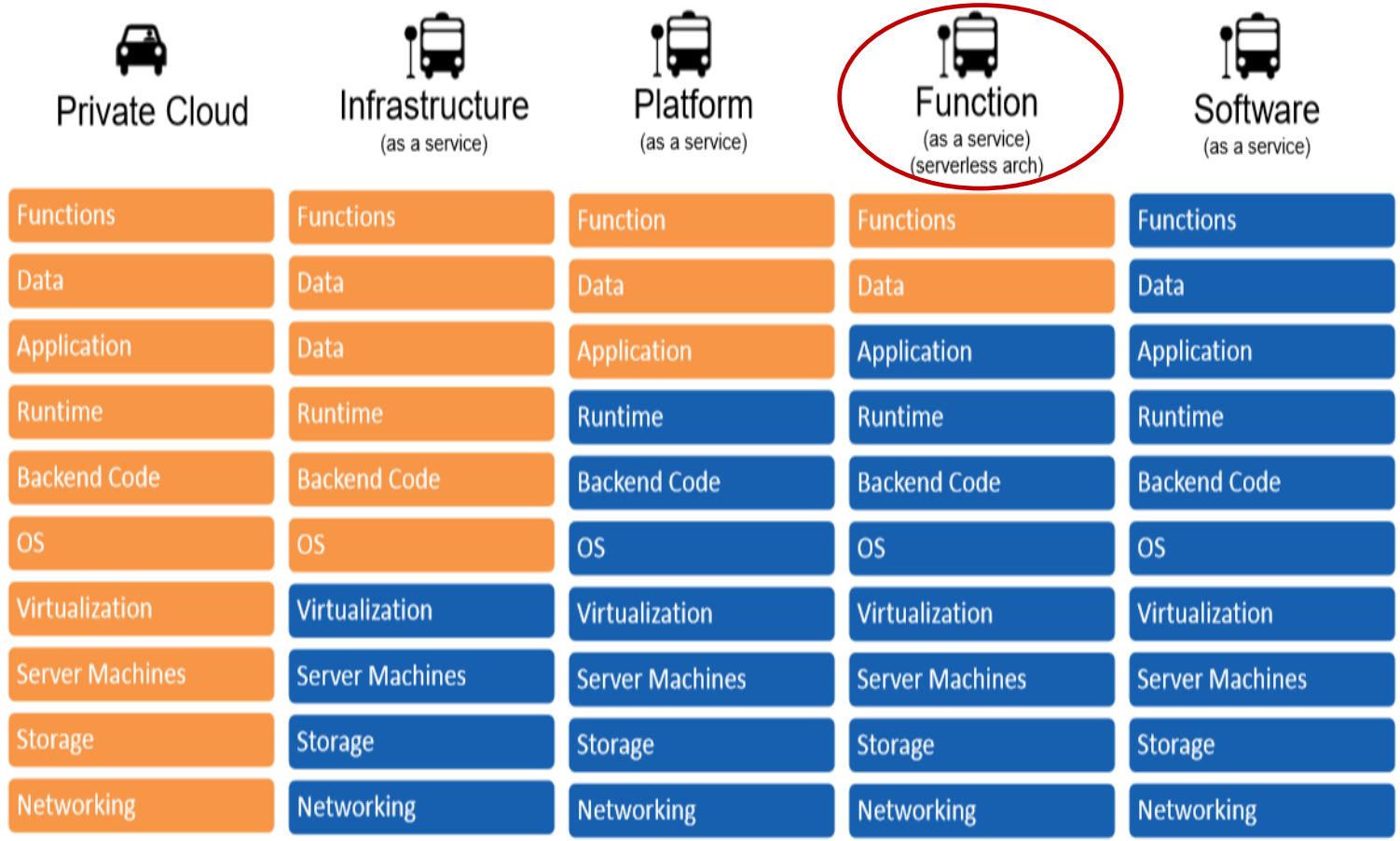
Functions as a Service (FaaS)

- Serverless can also mean applications where **some amount of server-side logic** is still written by the application developer but
  - unlike traditional architectures this is run in stateless compute containers that are **event-triggered**, **ephemeral** (may only last for one invocation),
  - They are fully managed by the cloud service provider.
- One way to think of this is **Functions as a Service** (FaaS).

Serverless Computing (FaaS)

- Function as a Service (Azure Functions)
- Platform to develop, run, and manage application
- Without the complexity of building and maintaining the infrastructure
- Similar Technologies
  - AWS Lambda
  - Google Cloud Functions
  - Open Whisk (IBM)

IaaS – PaaS, **FaaS**, SaaS



Awesome Vizualisation picked from : Ref : http://www.slideshare.net/manuel_silveyra/austin-cf-meetup-20150224/3

PS: We expect Container as a Service term in 2017-18 too, there is a separate section on it later

What is Serverless good for?

Serverless is **good** for

short-running
stateless
event-driven

- Microservices
- Mobile Backends
- Bots, ML Inferencing
- IoT
- Modest Stream Processing
- Service integration

Serverless is **not good** for

long-running
stateful
number crunching

- Databases
- Deep Learning Training
- Heavy-Duty Stream Analytics
- Numerical Simulation
- Video Streaming

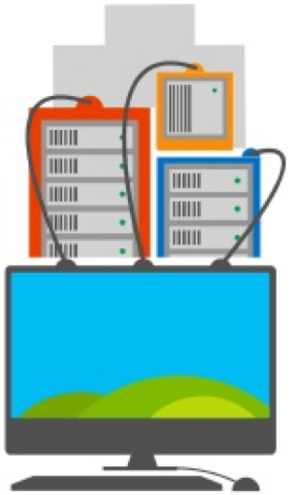
Characteristics of Serverless/FaaS

- Focus on business logic
- Event driven
- Short lived
- Stateless
- Auto scaling and auto deployment
- Billing per execution
- Prototypes become production code really quickly

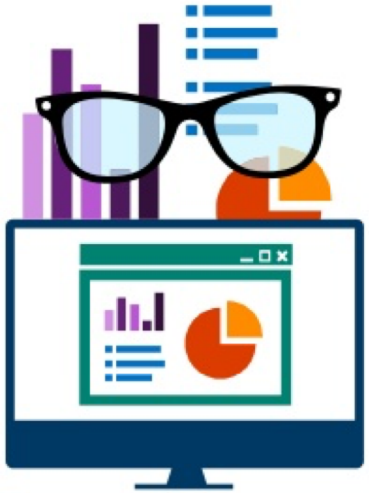
Challenges of Serverless/FaaS

- New architectural style
- Management of large populations of functions
- Vendor lock-in
- Execution duration limit
- Start up latency
- Network latency among functions

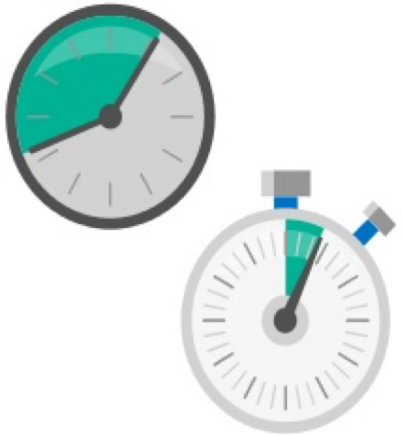
Benefits of Serverless



Reduced DevOps



Focus on Business Logic



Reduced Time To Market

Serverless Apps Lifecycle

Advantages of Serverless

- Rapid development and deployment
  - The developer is responsible only for the application itself, no need for time to be spent on server setup.
- Ease of use
  - The triggers that are necessary to execute your function are managed by the provider. Testing, logging, and versioning are all also managed for you.
- Lower cost
- Enhanced scalability
- No maintenance of infrastructure

Limitations of Serverless Computing

- You want control of your infrastructure.
- You're designing for a long-running server application.
- You want to avoid vendor lock-in.
- You are worried about the effect of "cold start."
- You want to implement a shared infrastructure.
- There are a limited number of out-of-the-box tools to test and deploy locally.

Serverless Function

We have gone from **monoliths** to **microservices** to **functions**

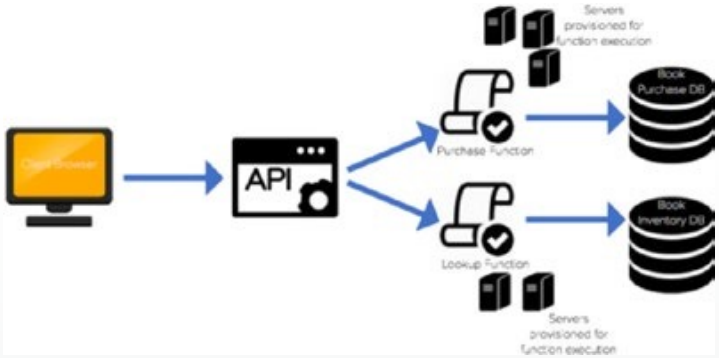
- **Microservices**
  - Smaller-grained services
  - Specified Functions
  - Defined Capabilities

- **Principles of FaaS:**
  - Complete abstraction of servers away from the developer
  - Billing based on consumption and executions, not server instance sizes
  - Services that are event-driven and instantaneously scalable

Architecture

- *Traditional architecture in which the server is provisioned and managed by the developer*

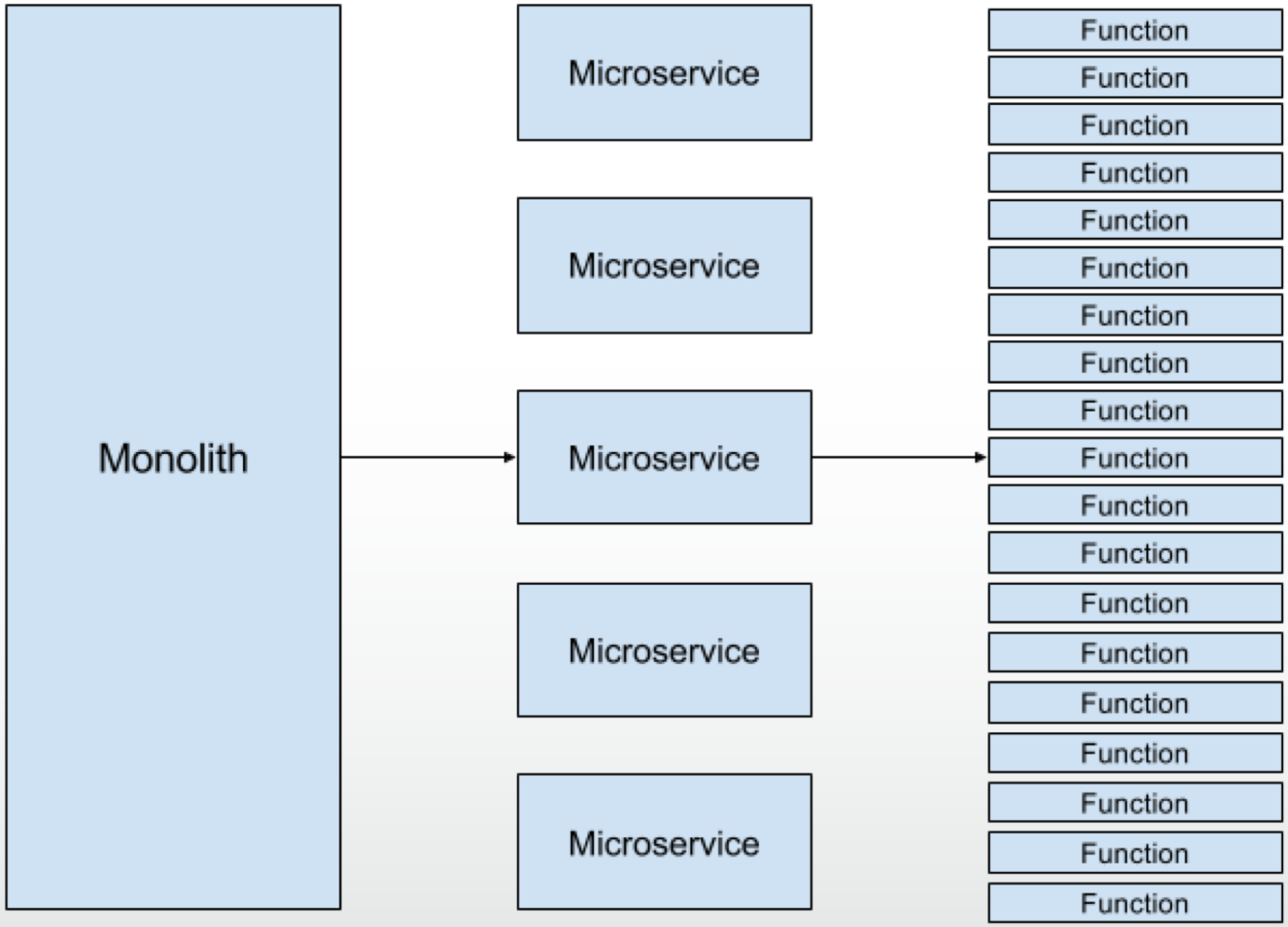
- *Serverless architecture where servers are spun up and down based on demand*



- *The implication on the* Development
  - Monolithic application vs microservices approach
  - Functions should be lightweight, scalable, and should serve a single purpose.

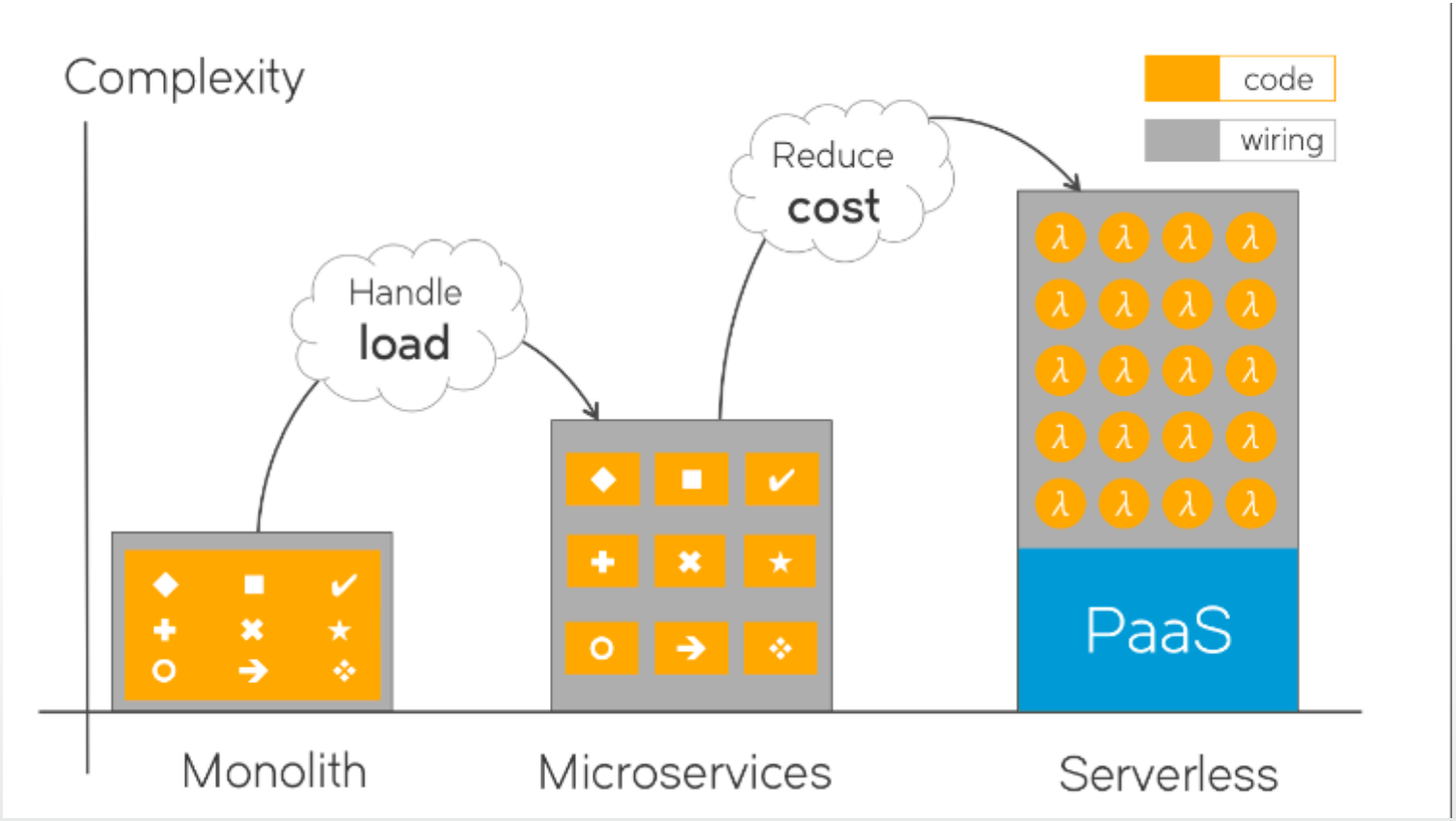
Serverless Function

From monoliths to microservices to functions

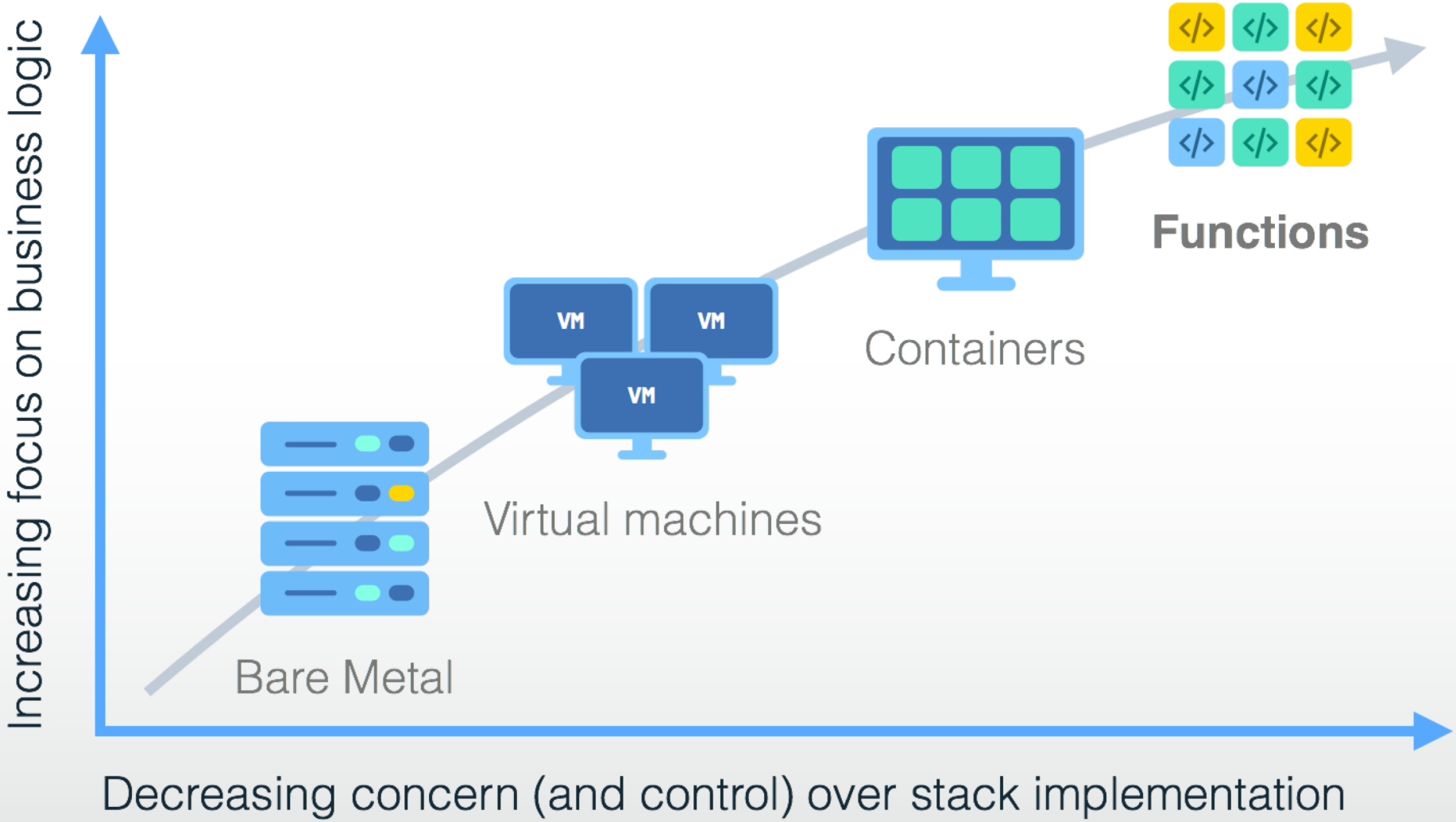


Cost vs Complexity (managed by the provider)

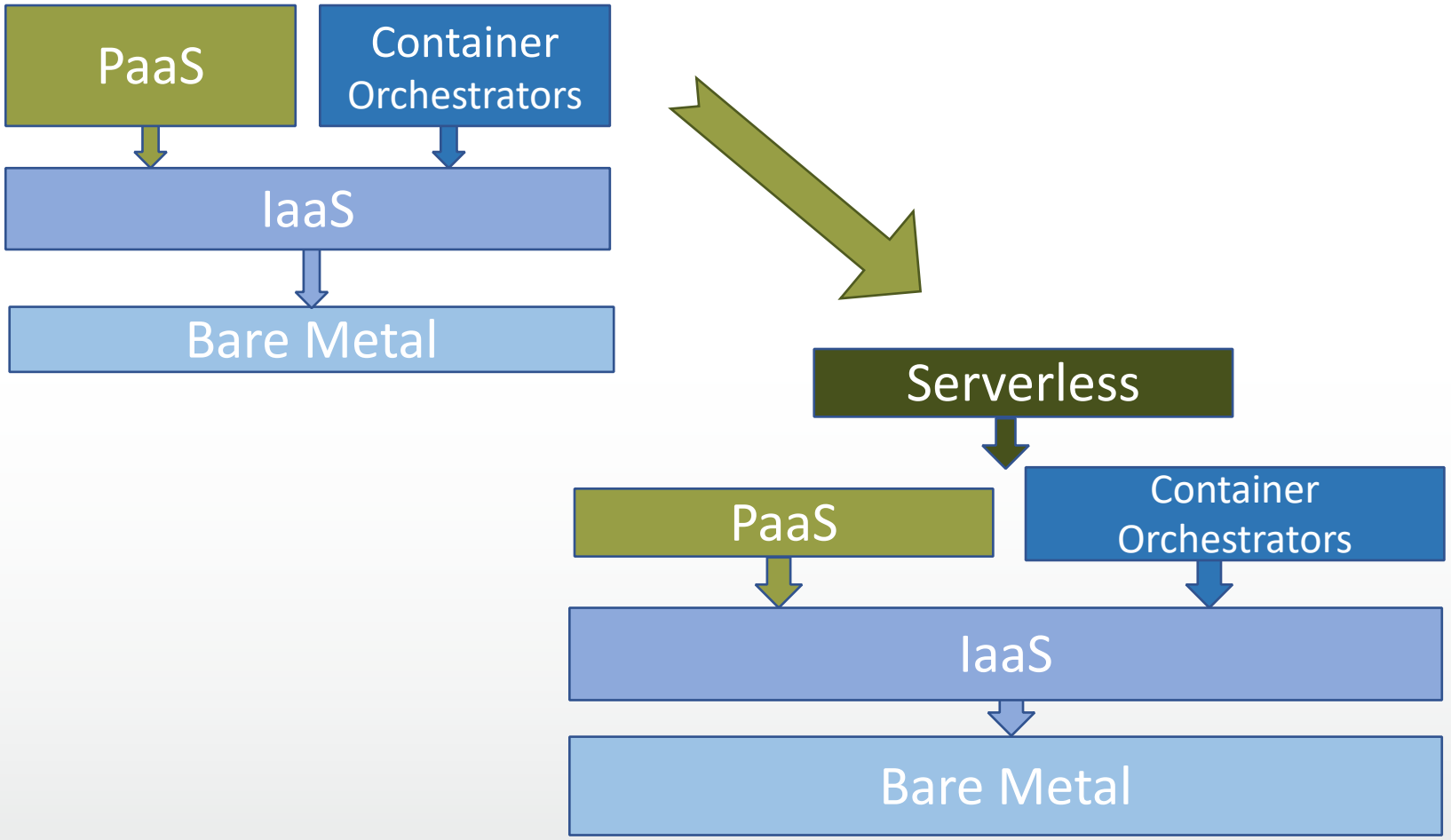
Cost of operation, usage and scale is reduced and Architectural Complexity is Increased with Serverless



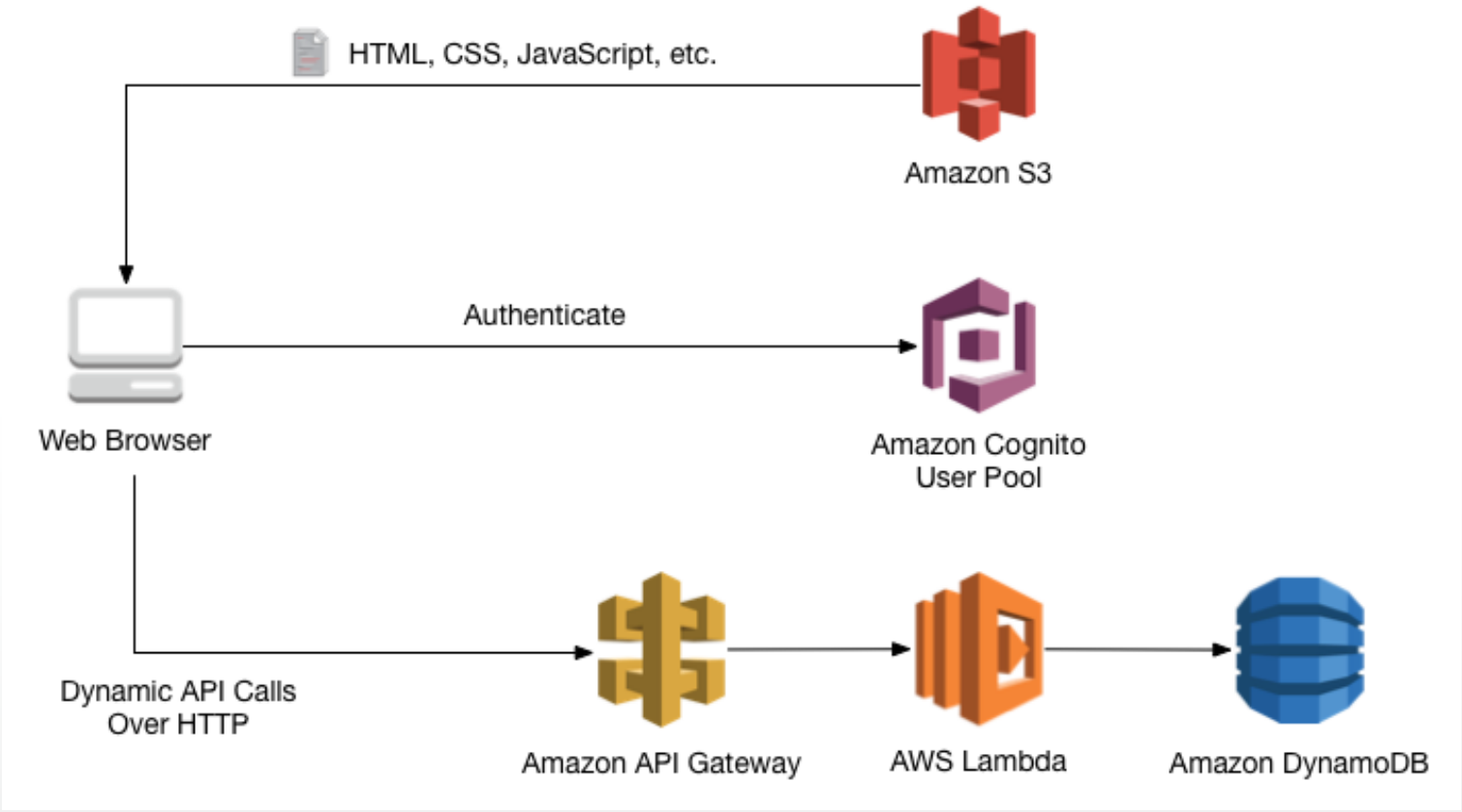
Serverless at a Glance



Evolution Of Serverless

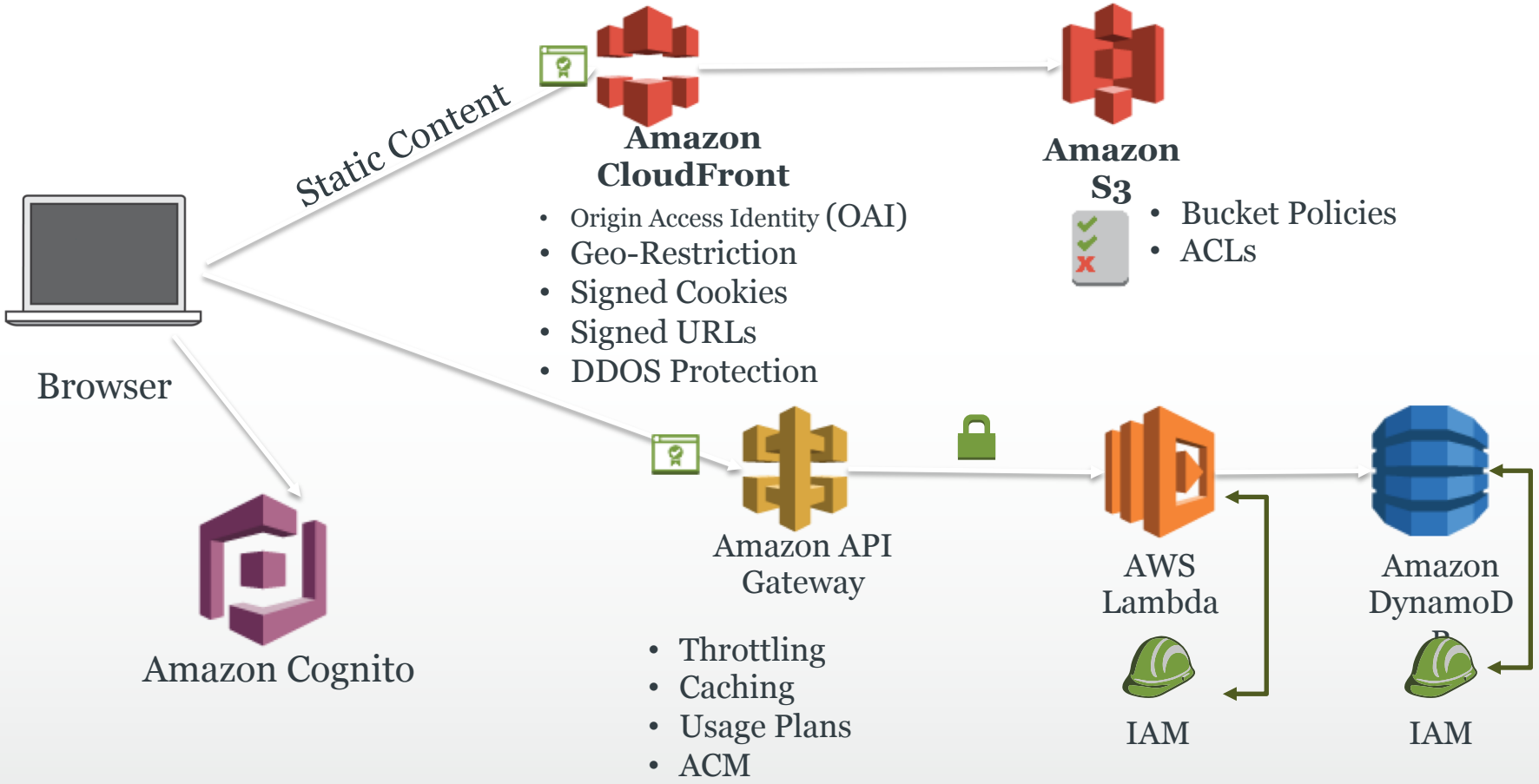


Serverless Web Application AWS

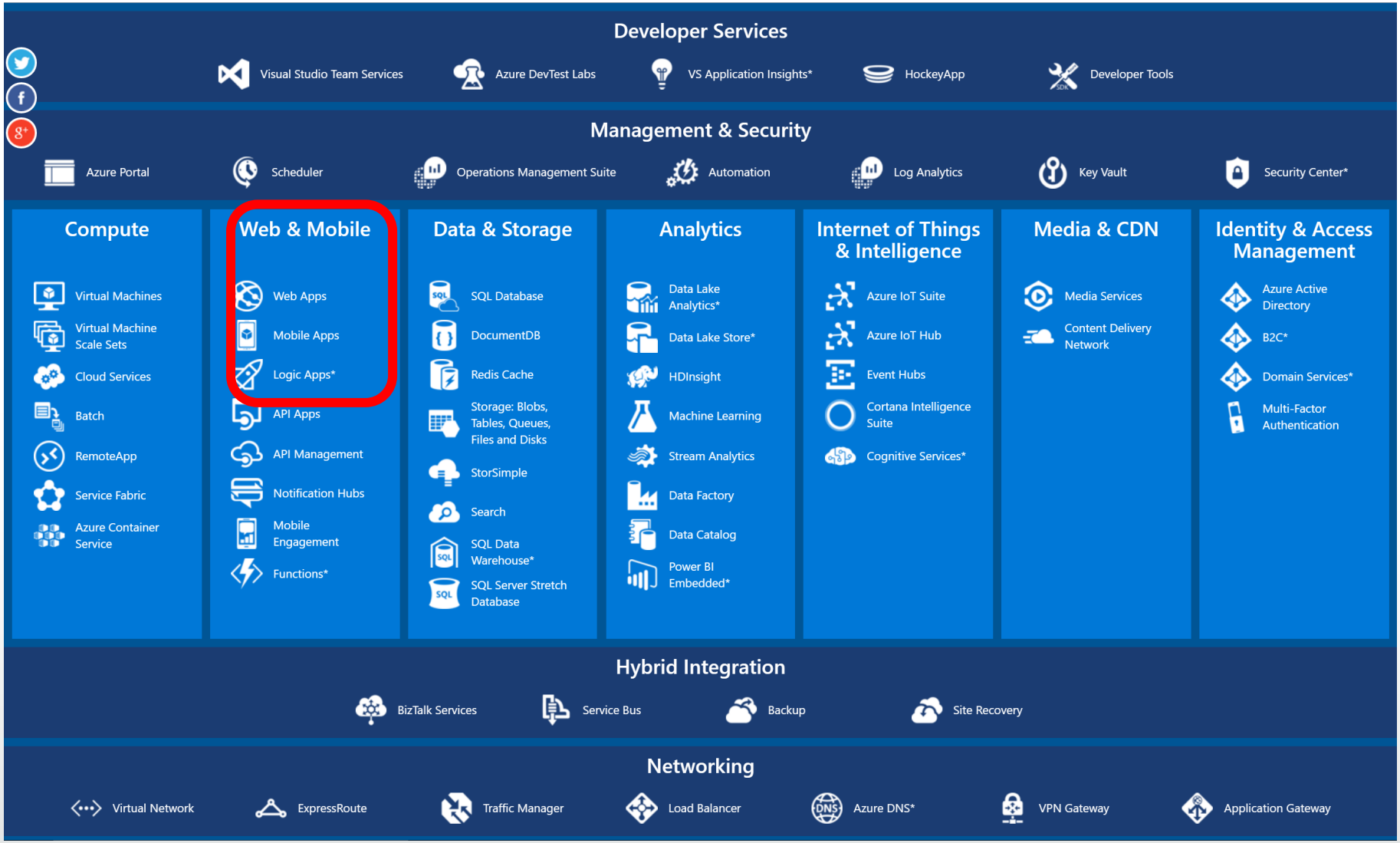


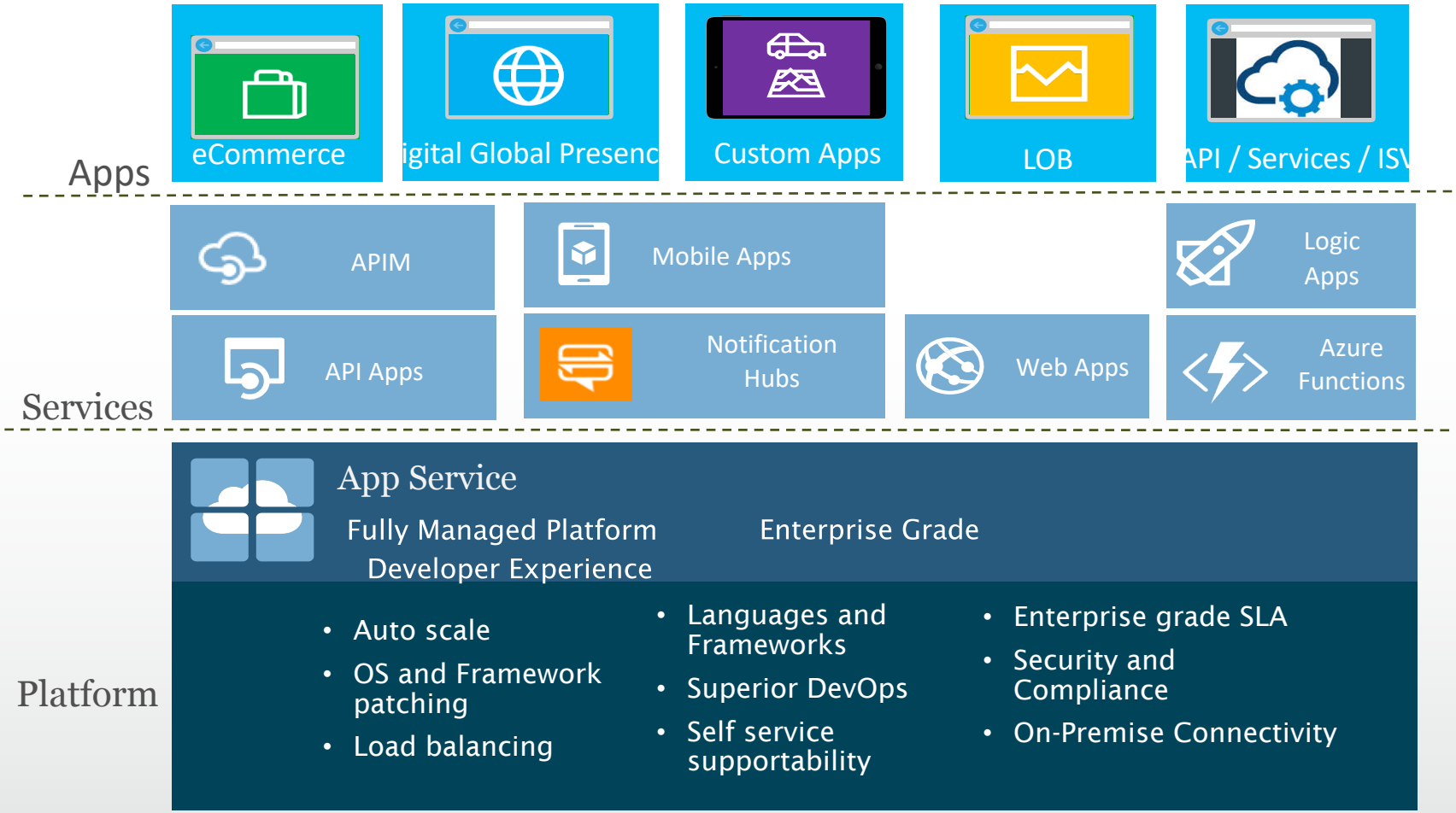
https://github.com/aws-samples/aws-serverless-workshops/tree/master/WebApplication

Serverless Web Application Architecture - AWS

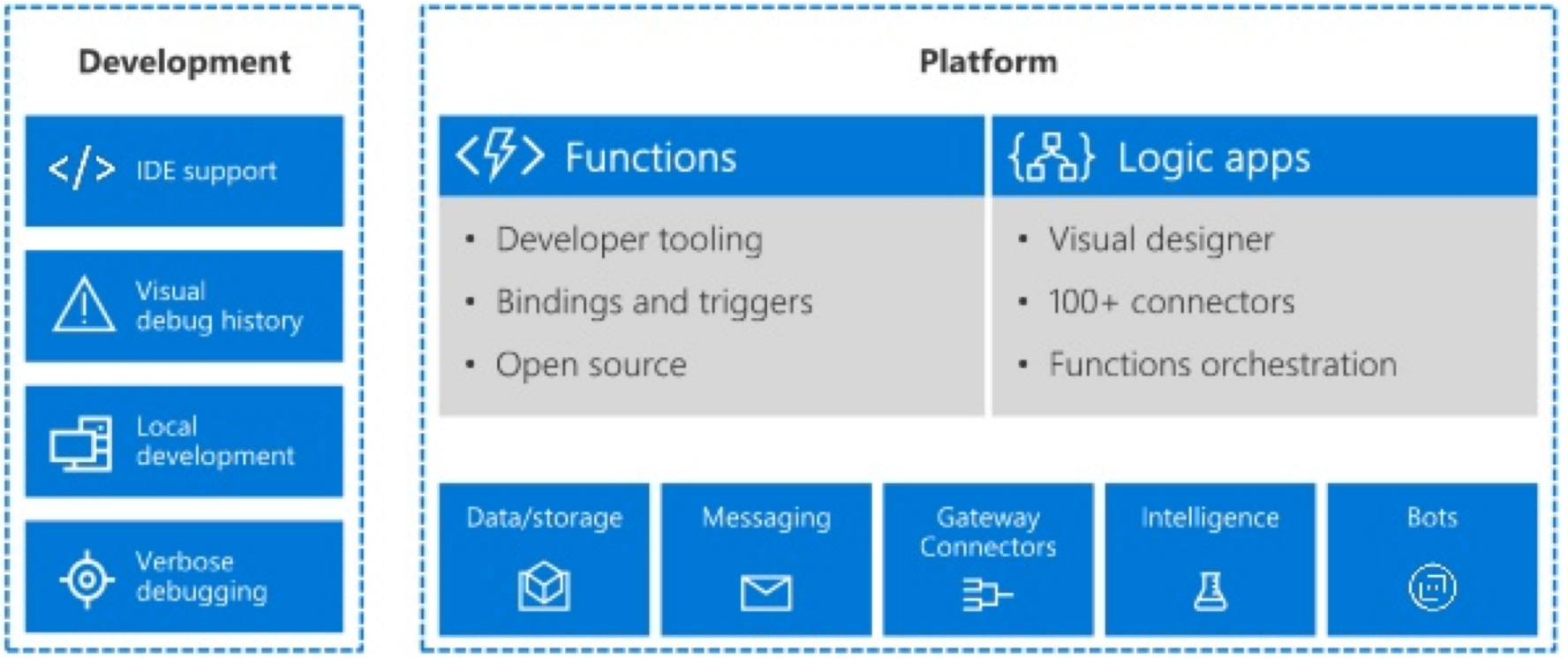


Azure Platform



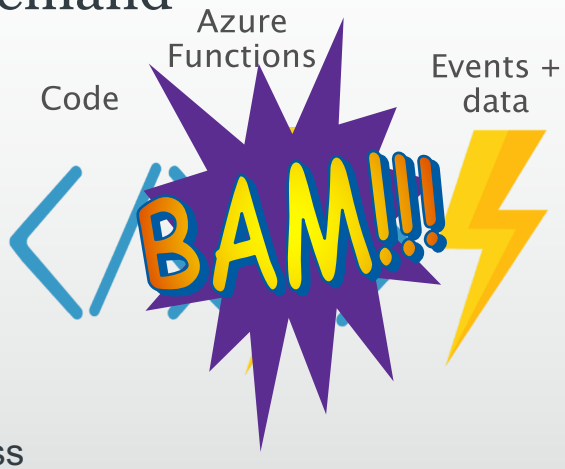

Apps
eCommerce
igital Global Presenc
Custom Apps
LOB
API / Services / ISV
Services
APIM
Mobile Apps
Logic Apps
API Apps
Notification Hubs
Web Apps
Azure Functions
Platform
App Service
Fully Managed Platform
Developer Experience
Enterprise Grade
Auto scale
OS and Framework patching
Load balancing
Languages and Frameworks
Superior DevOps
Self service supportability
Enterprise grade SLA
Security and Compliance
On-Premise Connectivity

Serverless Application Components

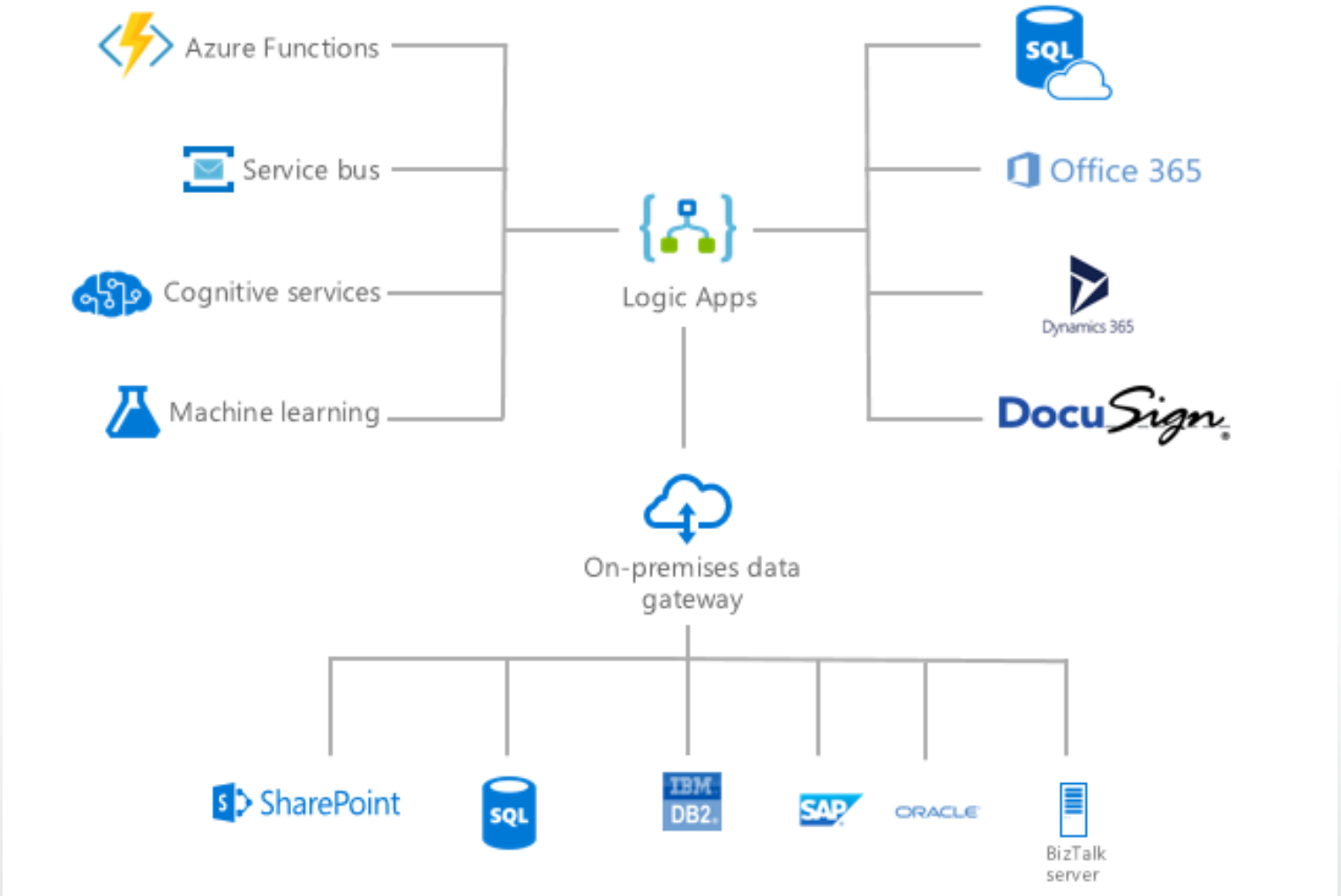


Azure Functions

- Process events with Serverless code.
  - Make composing Cloud Apps insanely easy
  - Develop Functions in C#, Node.js, Python, PHP, Batch and more
  - Easily schedule event-driven tasks across services
  - Expose Functions as HTTP API endpoints
  - Scale Functions based on customer demand
  - Easily integrate with Logic Apps
-



Logic Apps Connect Everything

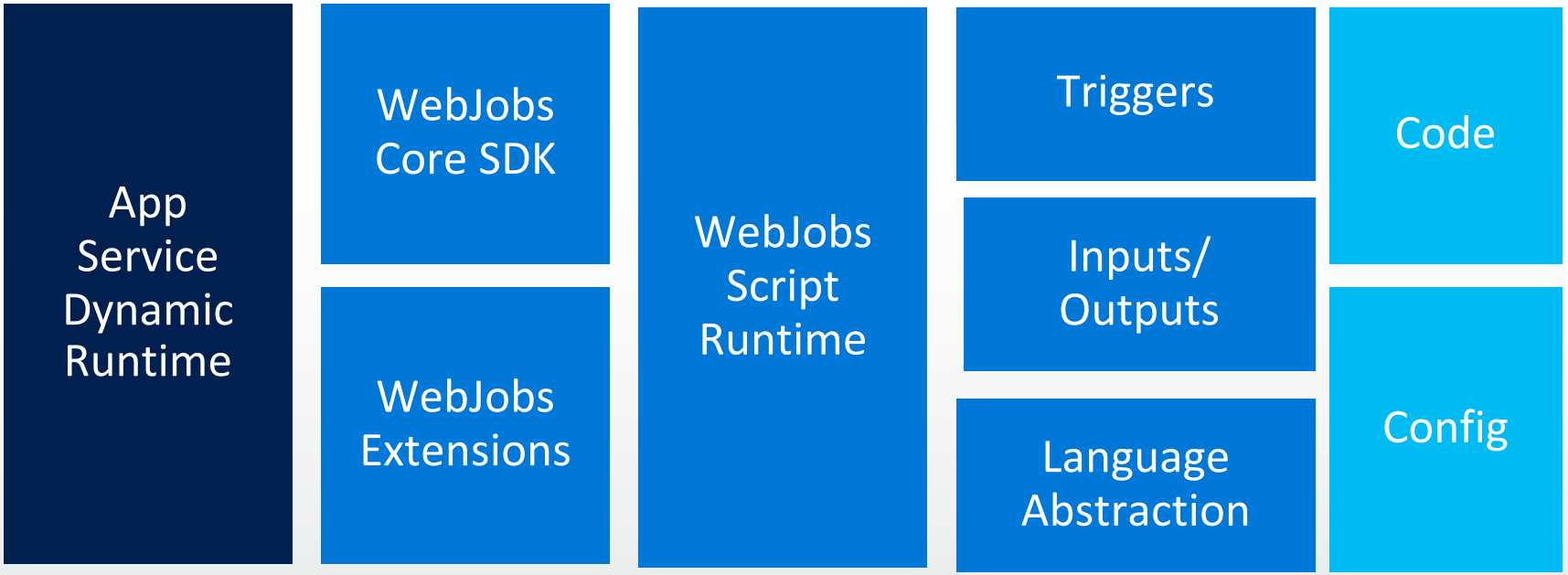


Azure Serverless Apps Design

- Distributed Architecture
  - Design stateless and ASync solutions to enable scaling.
  - Connect with other Azure Services via triggers and bindings.
  - Use Logic Apps to orchestrate workflows
  - Use managed connectors to abstract calls to cloud and on-premises services.
- Cloud DevOps
  - Design for automation. Use ARM templates.
  - Design DevOps for the cloud
  - Monitor the running apps with App Insights and tune for best experience.

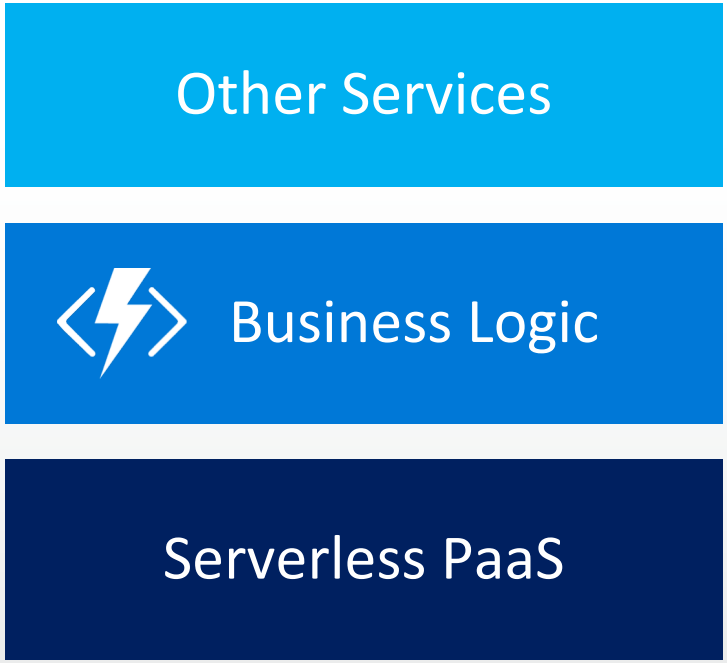
Azure Functions architecture

- Azure Functions is built around the WebJobs SDK runtime. The WebJobs SDK makes it easy to react to events and work with data in a consistent abstracted fashion.



Dual Abstraction

- Serverless compute abstracts away the compute
- Bindings abstract away the services you interact with



Triggers and Bindings

Triggers and Bindings

Type	Service	Trigger	Input	Output
Schedule	Azure Functions	✓		
HTTP (REST or webhook)	Azure Functions	✓		✓*
Blob Storage	Azure Storage	✓	✓	✓
Events	Azure Event Hubs	✓		✓
Queues	Azure Storage	✓		✓
Queues and topics	Azure Service Bus	✓		✓
Tables	Azure Storage		✓	✓
Tables	Azure Mobile Apps		✓	✓
No-SQL DB	Azure DocumentDB		✓	✓
Push Notifications	Azure Notification Hubs			✓
Twilio SMS Text	Twilio			✓

Platform and scaling

App Service offers **dedicated** and **dynamic** tiers.

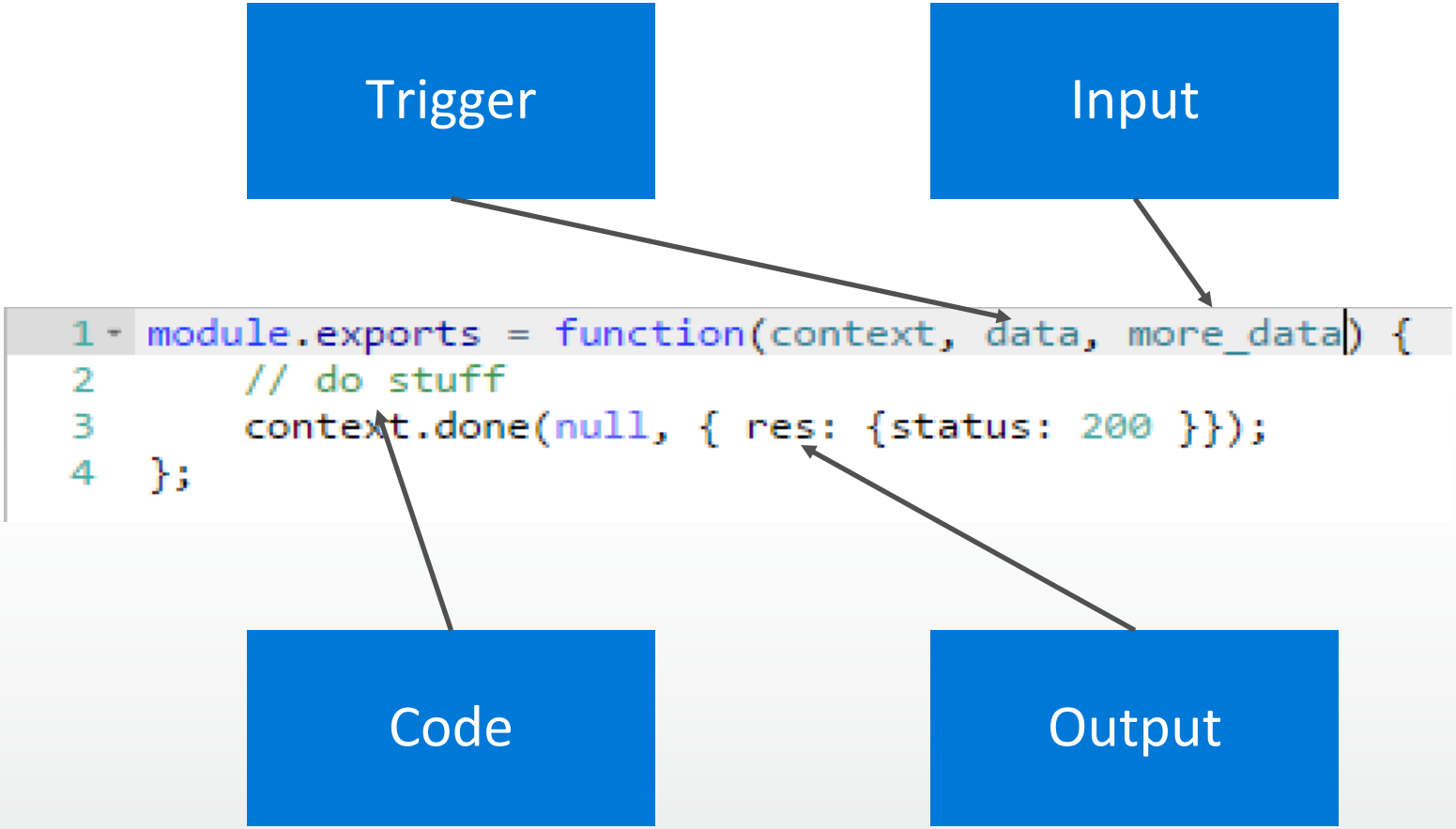
- Dedicated is the existing App Service plan tiers
  - Basic, Standard, Premium
  - Pay based on # of reserved VMs
  - You're responsible for scale
- Dynamic
  - Pay on number of executions
  - Platform responsible for scale

Dynamic tier pricing

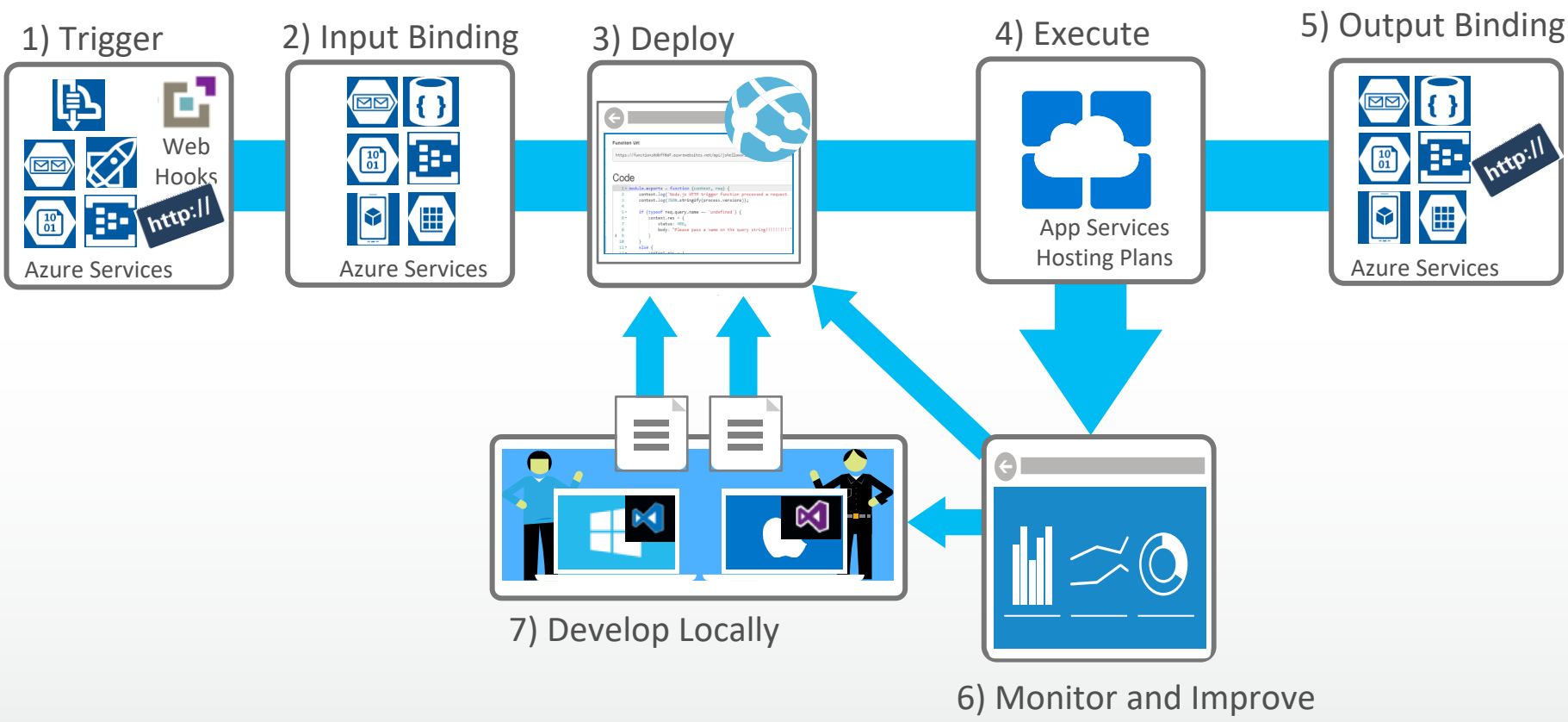
Pay per execution model - two meters, three units

- Number of executions
- Duration of execution × reserved memory

Functions programming concepts



Azure Function

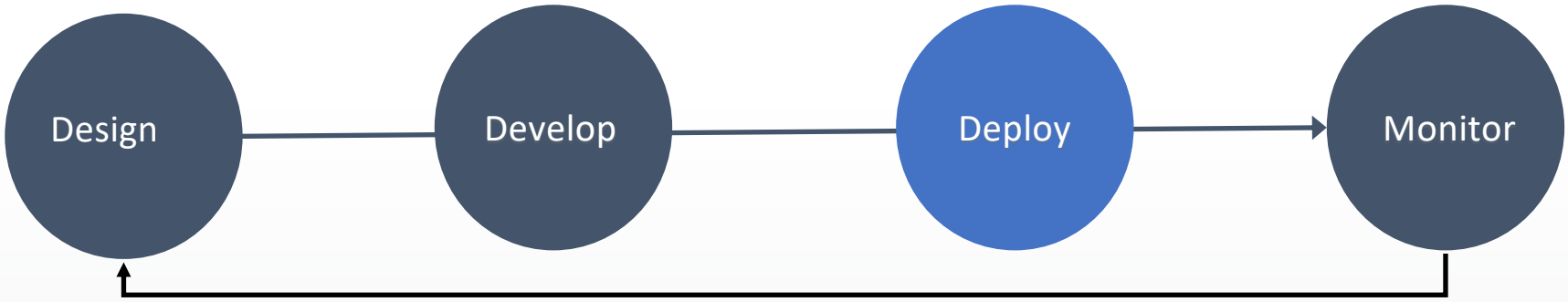


Develop

Local Development Tooling Options

- **Azure Functions Core Tools**
  - Provides the entire Functions runtime
  - Trigger off of Azure events and debug locally
- **JavaScript**
  - Use Visual Studio Code or any Node debugger
- **C#**
  - Use Visual Studio 2017 or 2019
  - Use class libraries with attributes in Visual Studio 2019

Deploy



Deployment Options

Resource deployment

- Azure Resource Manager (i.e. ARM)

Content deployment

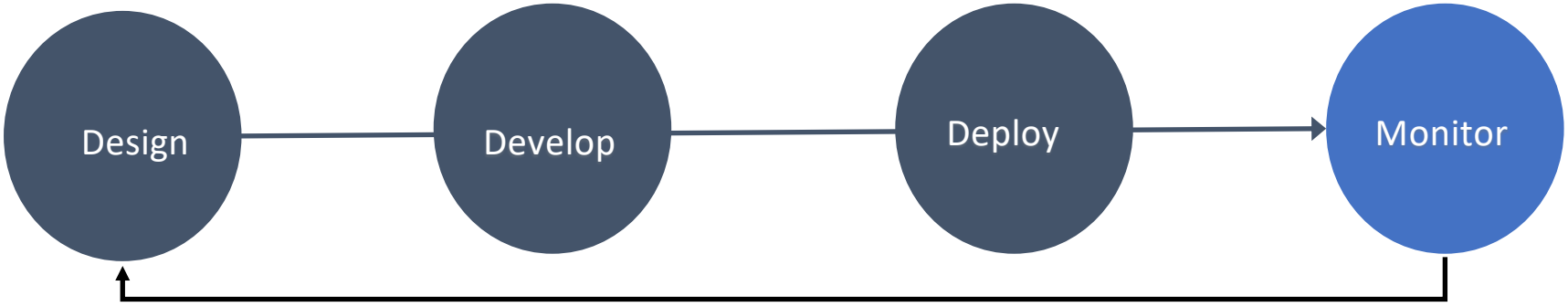
- Visual Studio
- Azure CLI (Logic App)
- Azure Functions Core Tools (Function App)
- CI/CD

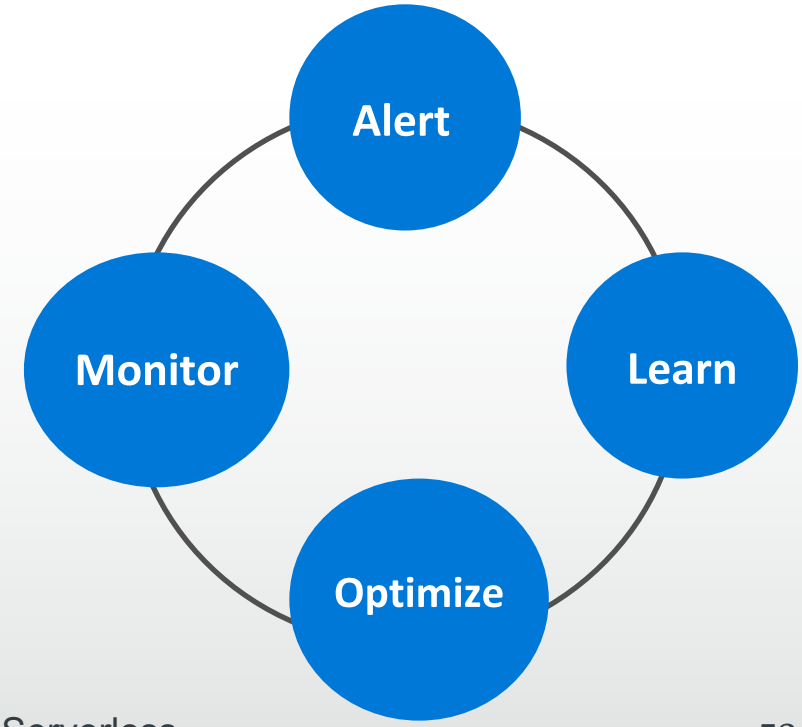
https://www.visualstudio.com/en-us/docs/build/get-started/aspnet-4-ci-cd-azure-automatic

Safe deployment practices

- Use Azure Functions deployment slots for environment separation and swap deployments

Monitor


Design
Develop
Deploy
Monitor


Alert
Monitor
Learn
Optimize

Summary

- Benefits of “serverless”
  - “Pinnacle of PaaS compute”
  - Not just hardware “servers”, but software servers are also **managed for you**
  - Focus on **business logic**, not solving technical problems not **core to business**
  - Lower effort to get started makes it easier to experiment (bots, etc.)

Signs that a serverless pattern might be useful for a given scenario

1. Stateless → Scale
2. Too complicated to deploy a traditional backend
3. Workload is sporadic (very low & high scale)
4. (Human) Operational costs need to stay low
5. Lots of different services involved

AWS vs Azure Function Price Comparison

	AWS Lambda	Azure Functions
Free Tier	1,000,000 free requests per month Up to 3.2 million seconds of compute time per month	1 million executions per month 400,000 GB-s per of Execution Time per month.
	$0.00001667 FOR EVERY GB-SECOND $0.20 PER 1M REQUESTS	Execution Time: $0.000016/GB-s Total Executions: $0.20 per million executions.

AWS and Azure NoSQL database price Comparison

	Amazon DynamoDB	Azure Cosmos DB
Free Tier	25 GB of Storage, 25 Units of Read Capacity and 25 Units of Write Capacity – enough to handle up to 200M requests per month	Free for 12 months.
	Provisioned Throughput (200M Writes) $0.47/month Provisioned Throughput (5.2M Read) $0.09/month	Reserved Request Units (RU)/second (400 RUs minimum) $0.008/hour => $5.86/month
Data Storage (per GB)	$0.25	$0.25

Resources:

- Thanks to the University's Office365 subscription, all students and staff are already pre-registered and we just have to sign on at the Azure portal at:

  https://portal.azure.com/#home

  – There's no credit card involved, and everything's free (unless they use more that a million "Function" calls per month).

- This page is the right tutorial for how to create your first Azure Function:

  – https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-azure-function

Resources:

- **Build Your First Serverless Web Application on Azure**
  - https://mikepfeiffer.io/blog/azure-serverless-101
- You can try Functions – https://functions.azure.com
  - There is a short video on the bottom of this page, you can watch
- As an extra one you can try App Service – https://tryappservice.azure.com

Extra Tool & Resources

Serverless Framework Deployment Tool

- Serverless Framework, not to be confused with serverless architecture, is an open source application framework based on Node.js that lets you easily build serverless architectures.
- This framework allows you to deploy auto-scaling, pay-per-execution, event-driven functions to AWS, Azure, Google Cloud, and IBM's OpenWhisk.
- It makes configuring and deploying your function to your given provider incredibly easy, which also contributes to a more rapid development time.

Using the Serverless Framework

- The benefits to using the Serverless Framework to deploy your work include:
  - Fast deployment: You can provision and deploy quickly using a few lines of code in the terminal.
  - Scalability: You can react to billions of events on Serverless Framework; and you can deploy other cloud services that might interact with your functions (this includes trigger events that are necessary to execute your function).
  - Simplicity: The easy-to-manage serverless architecture is contained within one ***yml*** file that the framework provides out of the box.
  - Collaboration: Code and projects can be managed across teams.

Serverless.com Resources

- **Azure Functions Provider Documentation**
  - https://serverless.com/framework/docs/providers/azure/
- **How to Create a REST API with Azure Functions and the Serverless Framework - Part 1**
  - https://serverless.com/blog/serverless-azure-functions-v1/
- **Azure Functions Support with Serverless v1.8**
  - https://serverless.com/blog/serverless-v1.8.0

Questions?