Azure Serverless & Microservices Briefing

Edinburgh, April 26th 2018

David.Gristwood@microsoft.com
Ross.P.Smith@microsoft.com
Ben.Coleman@microsoft.com

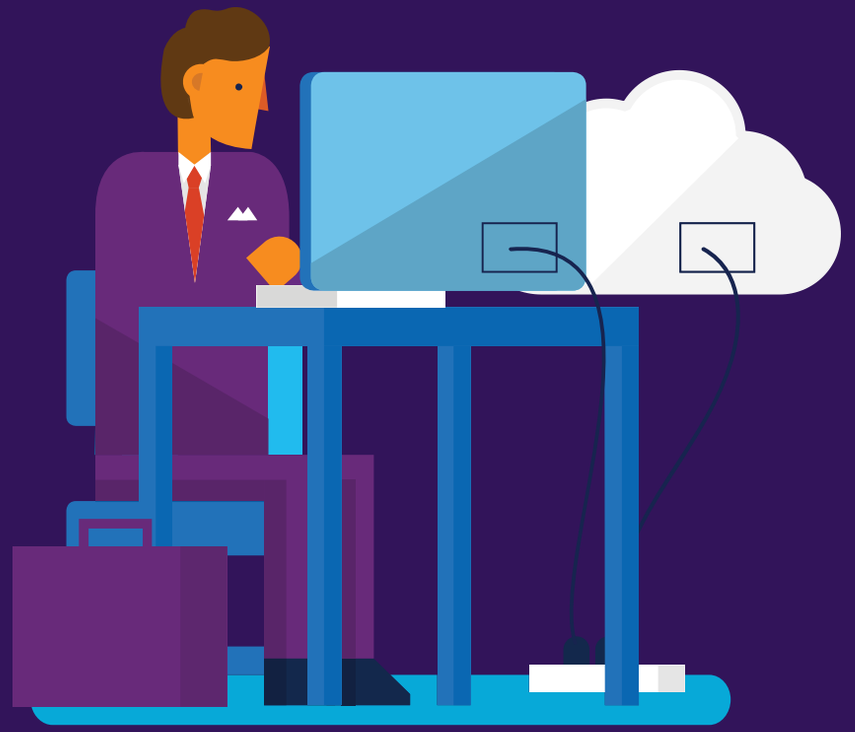
Adam.Jackson@microsoft.com

Wifi Code:
MSFTGUEST
msevent42sb

Microsoft

Slides: aka.ms/azureevent

#Azure
#BuildWithAzure



vipazure@microsoft.com

Agenda

- Microservices overview
- Smilr overview
- Service Fabric
- Orleans
- Kubernetes
- Serverless and Azure Functions
- Q&A and wrap-up

Coffee breaks mid morning and mid afternoon

Lunch around 12:30

Finish about 4:00

30 results returned in date order

Practice

Application Innovation ×
Business Apps ×
Cloud Application Development ×
Cloud Infrastructure & Management ×
Data Platform & Analytics ×
Internet of Things ×
Modern Workplace ×
MPN ×

Product

AI ×
Azure ✓
Citrix ×
Cloud ✓
Data & AI ×
Dynamics 365 ×
Enterprise Mobility + Security ×
GDPR ×
IoT ×
Microsoft 365 ×
Office 365 ×
PowerBI ×
SAP ×
Security ×
Windows 10 ×

Audience

Education ×
Licensing ×
Sales ×
Technical ✓

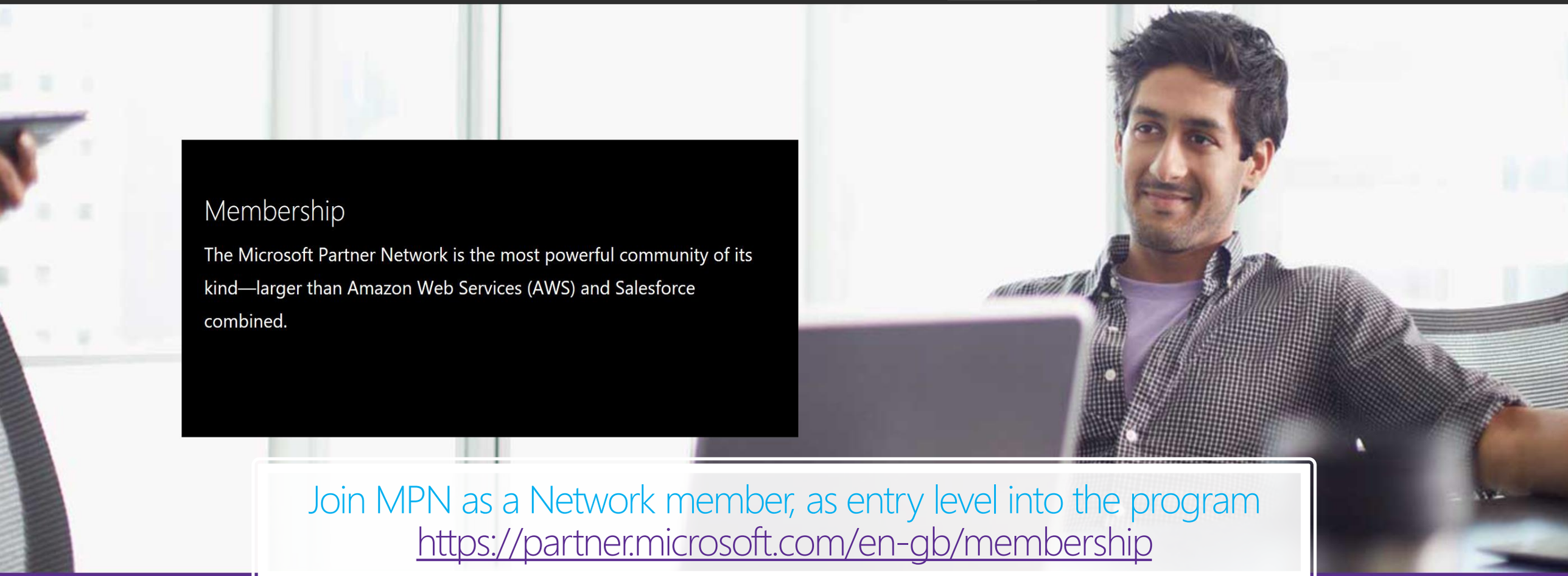
Level

Get Started ×
Grow ×
Optimise ×

Sort

Recent

http://aka.ms/upskill



Membership

The Microsoft Partner Network is the most powerful community of its kind—larger than Amazon Web Services (AWS) and Salesforce combined.

Join MPN as a Network member, as entry level into the program
https://partner.microsoft.com/en-gb/membership

You want to grow your business. We know how.

Partnering with Microsoft pays off.

What is

ocp

Adam Jackson

Partner Dev Manager / ocp

Microsoft

- *Partner Development Manager & Technical Evangelist*
- *Hacks & Technical Events*
- *1:1 Engagement & Support Services*
- *Commercial Engagement & Connections*
- *Sell with us!*

@adamj89

Adam Jackson on LinkedIn

adam.jackson@microsoft.com

Microsoft

Lets talk about Microservices …

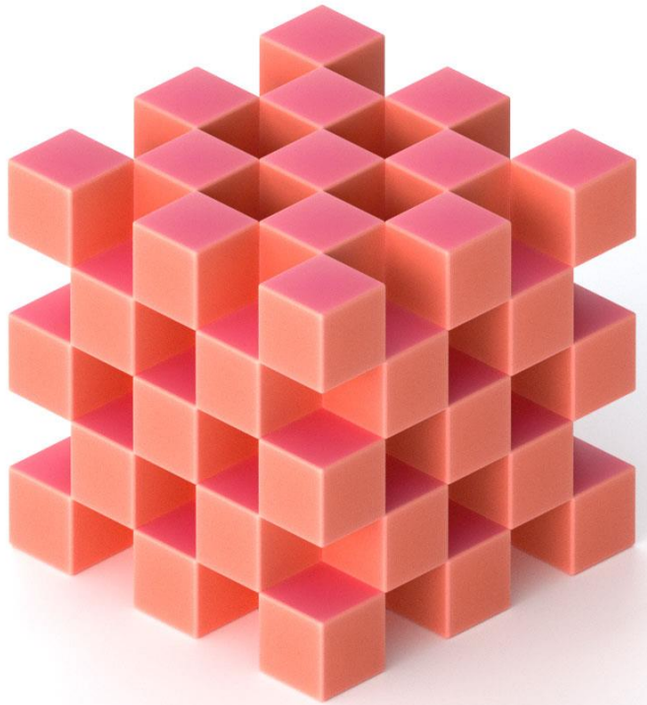
Today's Microservices demo is brought to you by the letter … S

aka.ms/smilr

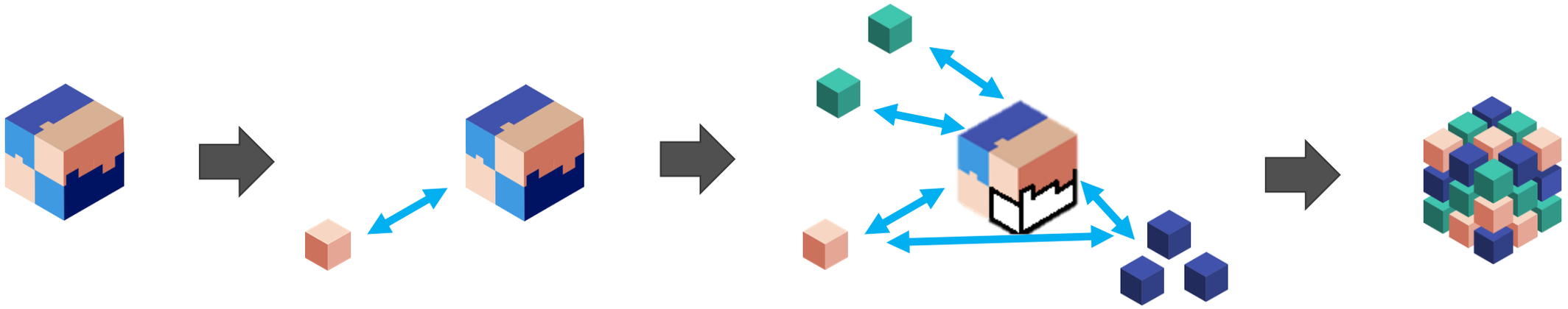
MICROSERVICES IS AN ARCHITECTURAL DESIGN PATTERN

Loosely coupled collection of **small**, **autonomous** services.

Each service is **self-contained** and should implement a **single** business capability.



Evolution to Microservices



Monolith

Client/Server

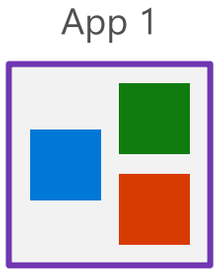
3-tier

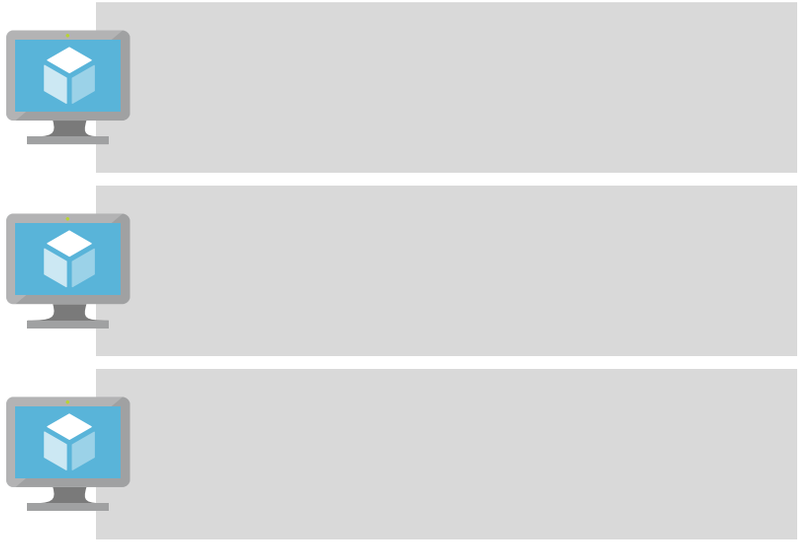
Microservices

Architecture and Deployment

Traditional Application

- Has its functionality within a few processes that are componentized with layers and libraries.
- Scales by deploying the whole app on multiple servers or VMs

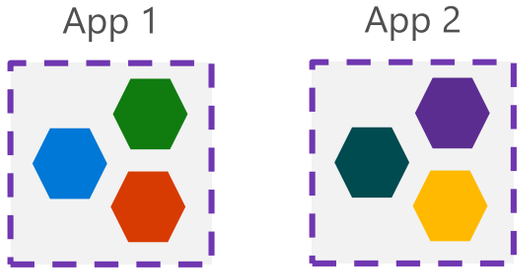


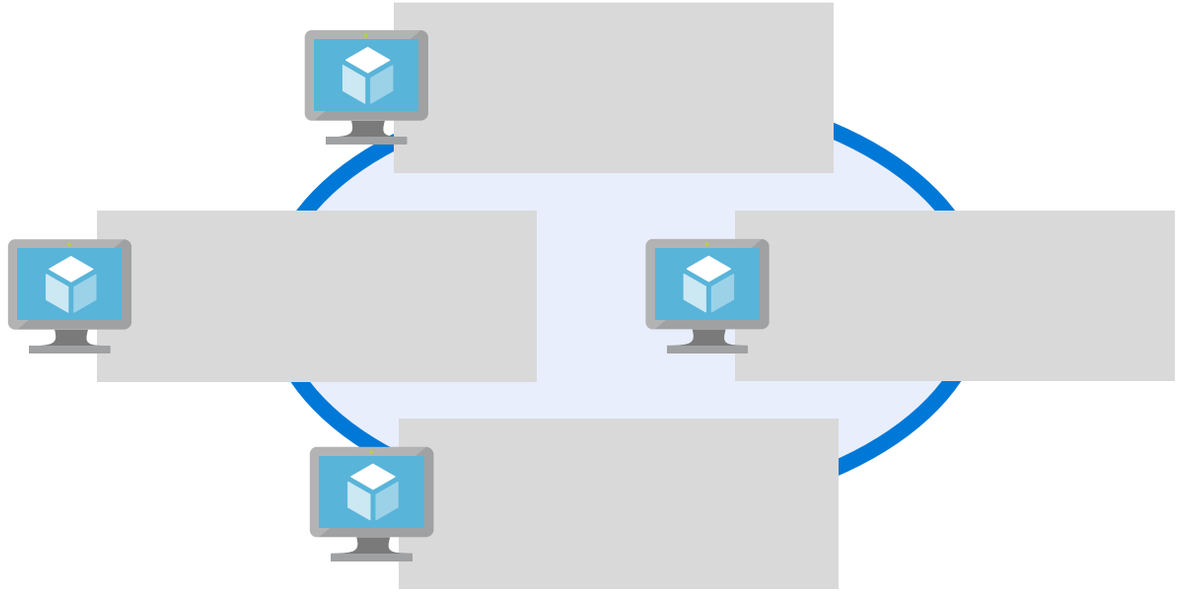


- **Course grained scaling**
- **Deploy entire app stack each time**
- **Difficult resource optimization**

Microservices Application

- Application functionality segregated into separate smaller services.
- Scaled by deploying services independently with multiple instances across VM clusters

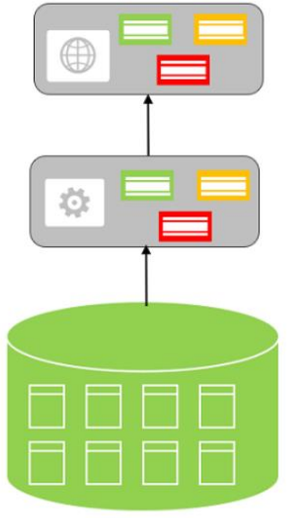




- **Fine grained scaling**
- **Deploy individual services as needed**

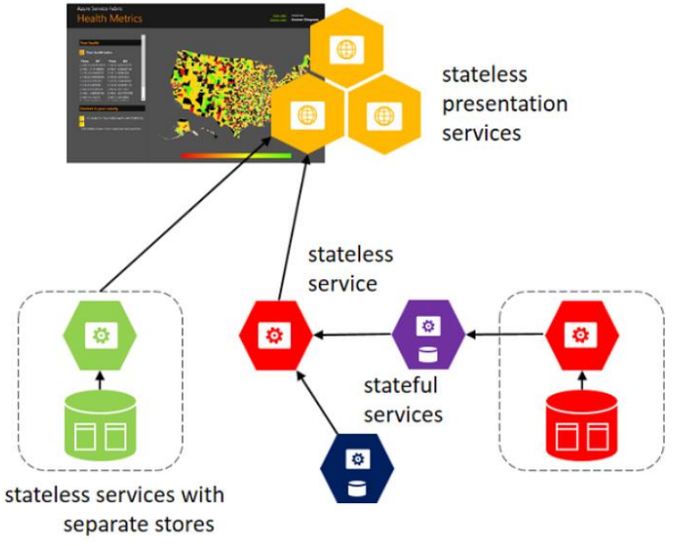
Microservices Challenges

State in Monolithic approach



The monolithic approach has a single database and tiers of specific technologies.

State in Microservices approach

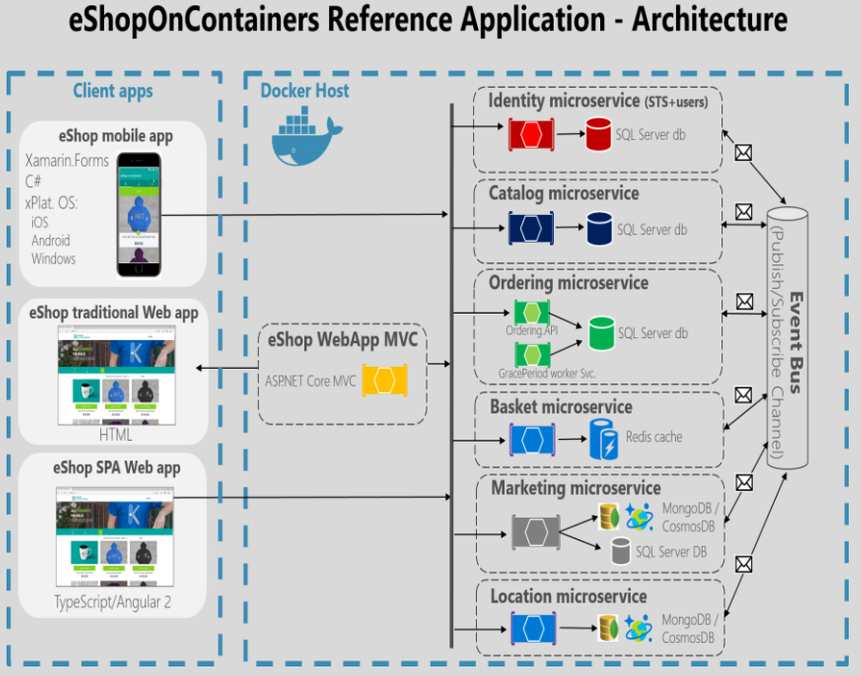


The microservices approach has a graph of interconnected microservices where state is typically scoped to the microservice and various technologies

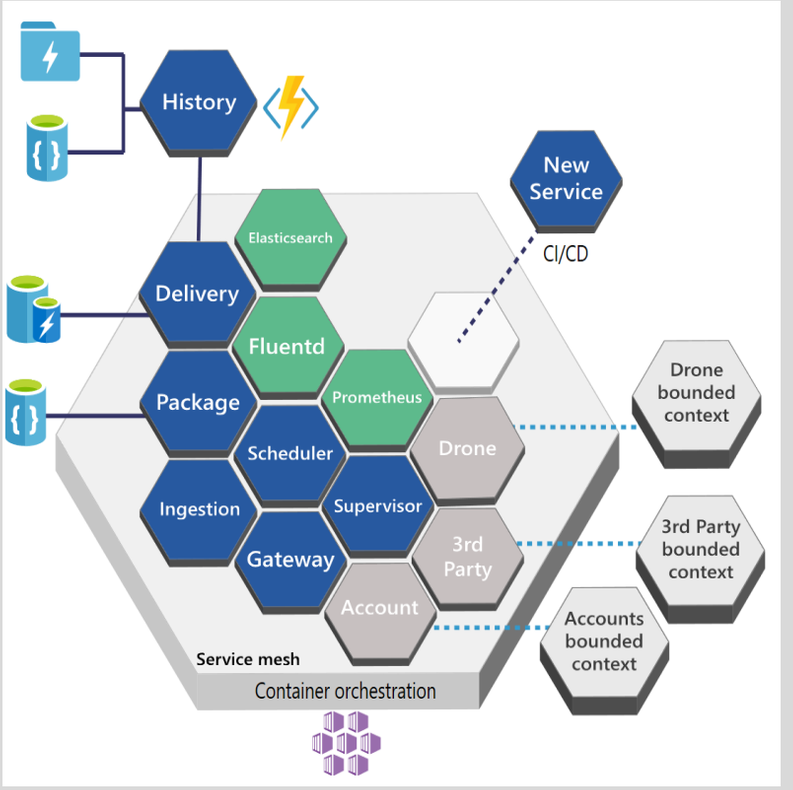
- #1: How to define the boundaries of each microservice
  - "user" could be in CRM, a customer, logged on account, etc
- #2: How to create queries that retrieve data from several microservices
  - API Gateway, CQRS with query/reads tables, big data repository
- #3: How to achieve consistency across multiple microservices
  - CAP theorem
- #4: How to design communication across microservice boundaries
  - Blocking, chaining, coupling, etc

Reference Architectures and Guides

aka.ms/MicroservicesArchitecture



docs.microsoft.com/azure/architecture/microservices

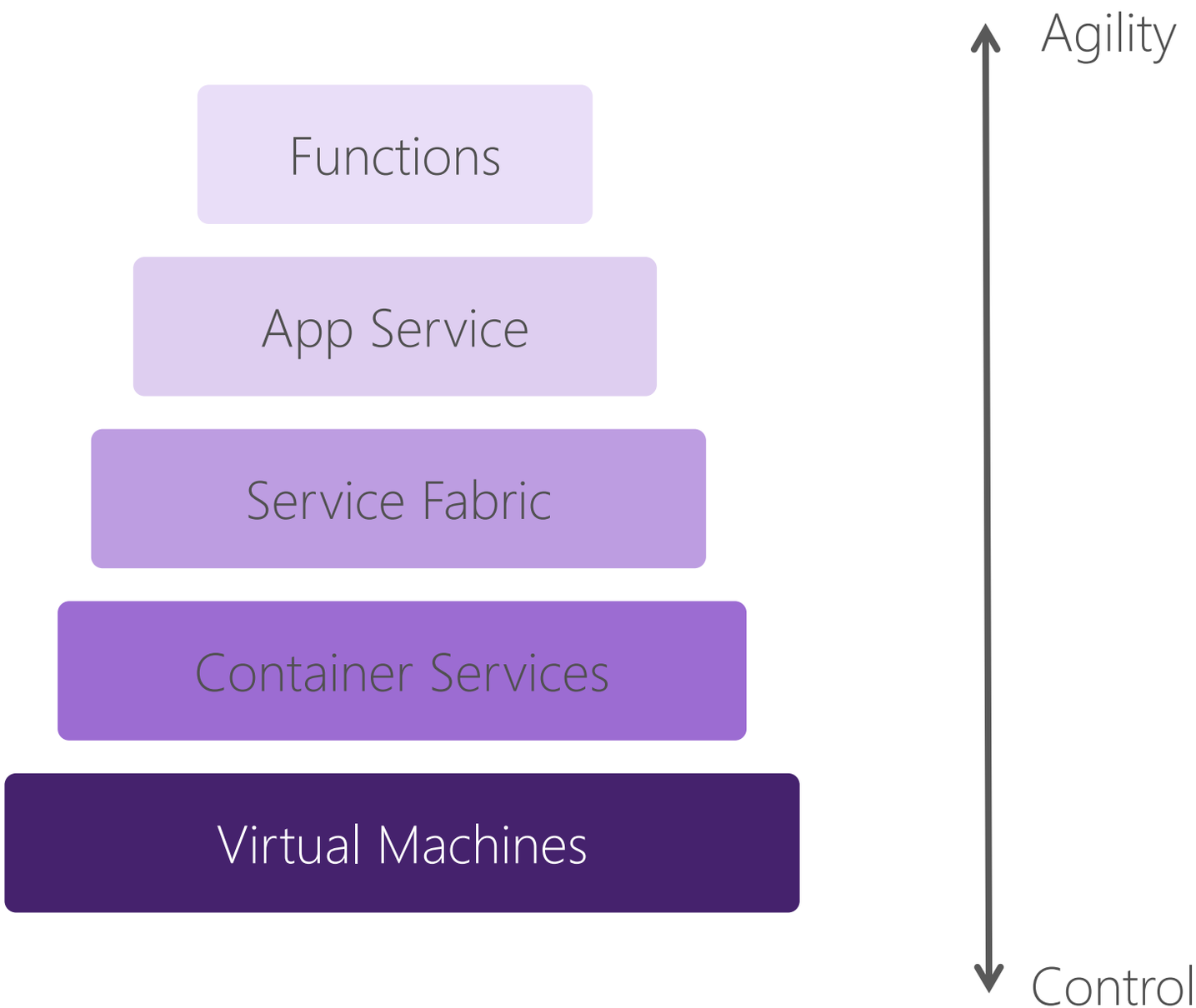


aka.ms/microservicesebook

Free eBook

Microservices and Azure

Code


Functions
App Service
Service Fabric
Container Services
Virtual Machines
Agility
Control

Today's Focus – where does your code run?

- *Containers - Ross*
- *Platform as a Service / App Services – Ben*
- *Service Fabric – Ross & David*
- *Kubernetes – Ben*
- *Azure Functions – David & Ben*

We will explore the trade-offs for each choice

Data

"human data"

Transactional integrity, operational information, etc.

"machine data"

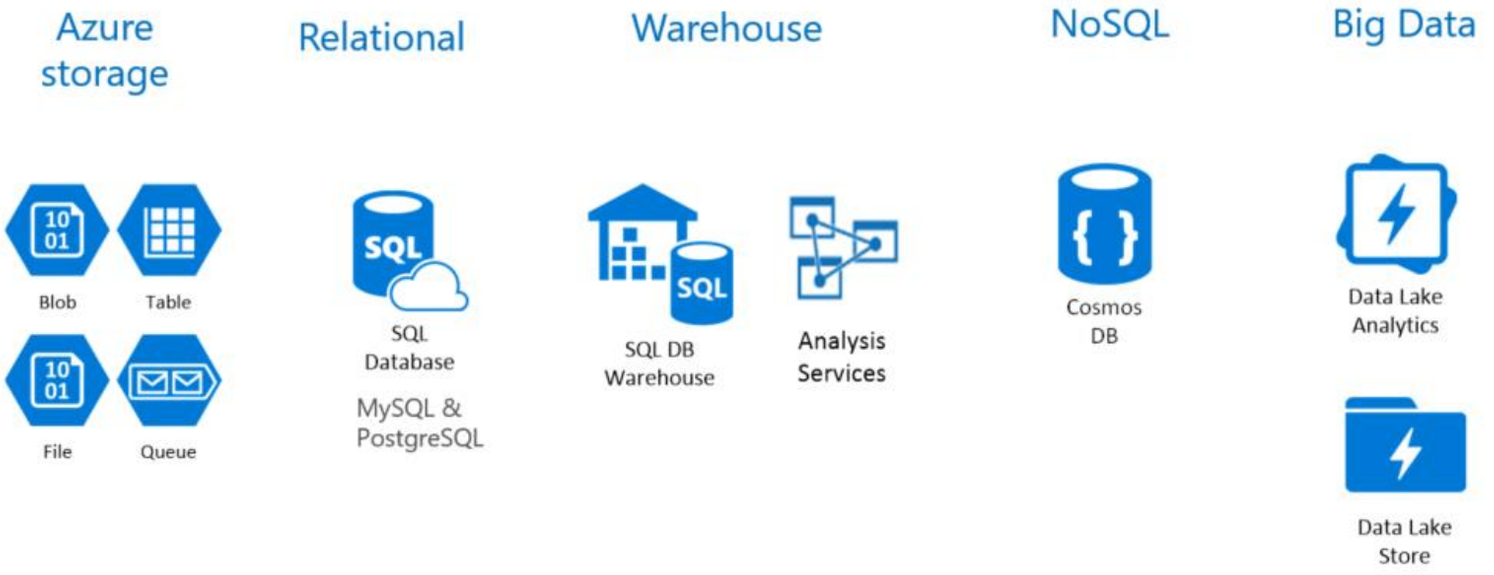
Independent, telemetry, insights, etc.

Polyglot Persistence

Different databases are designed to solve different problems. Using a single database engine for all of the requirements usually leads to non-optimal solutions

e.g.:

- *User session*
- *Catalogue data*
- *Product search*
- *Shopping cart*
- *Orders database*
- *Analytics*
- *Reporting, ….*



Events and Messages

Messages and Events

Messages

- Typically carry information needed for a step in a defined workflow
- May express inherent monetary value or commands to performs actions
  - Consider Azure Service Bus or Azure Queues

Events

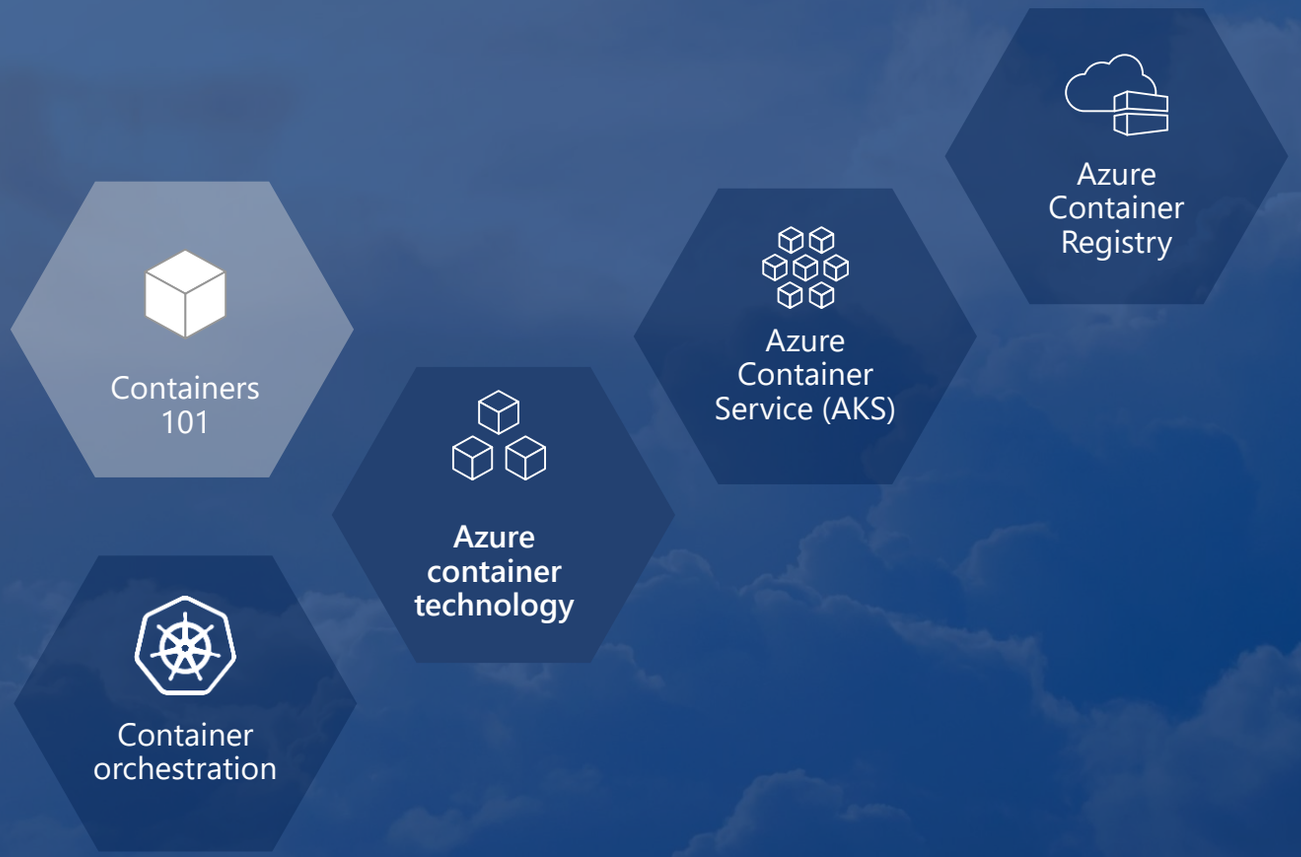
- Don't generally convey publisher intent, other than to inform

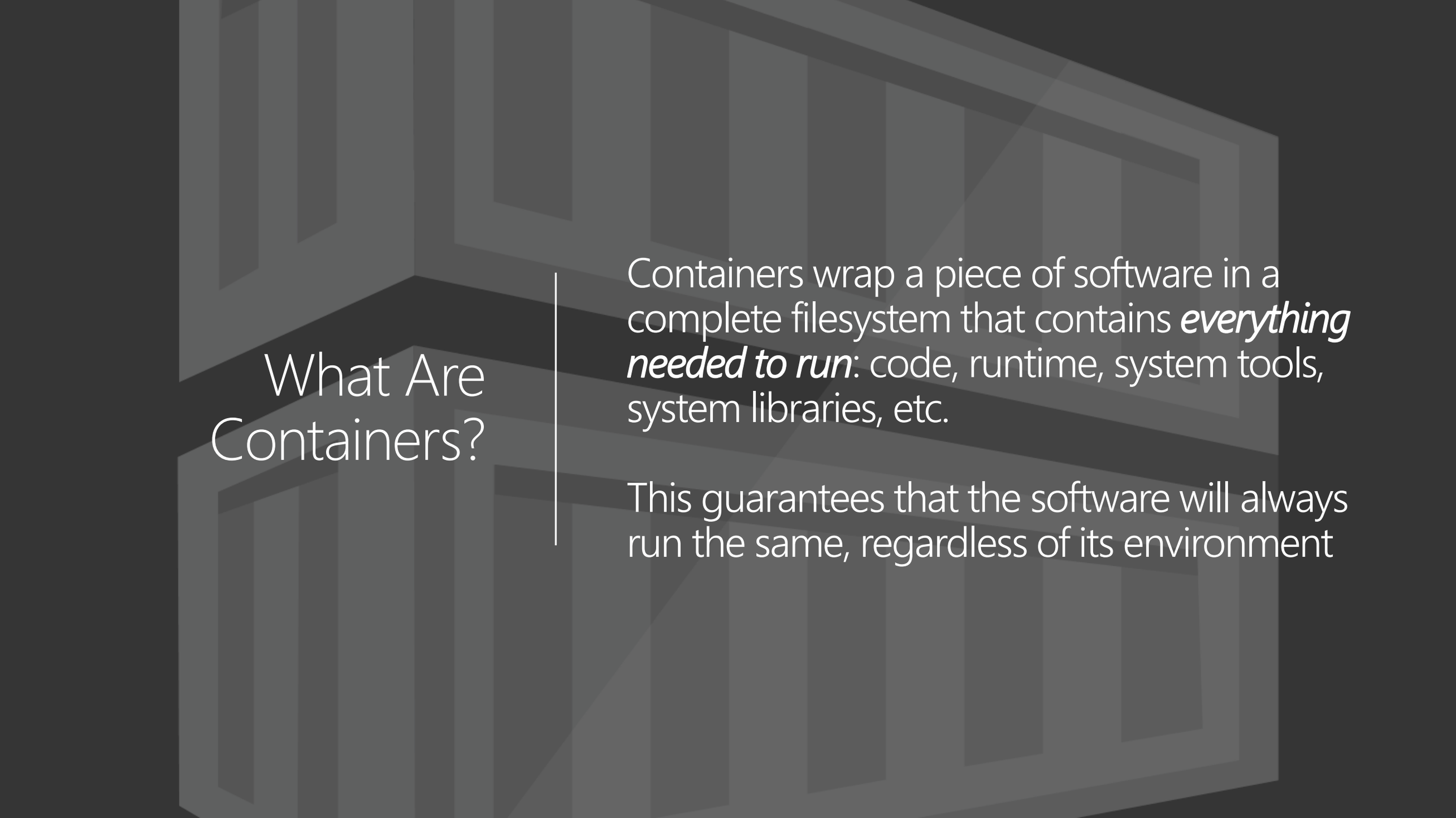
1. "Business logic activity" carried out by publishing application
   - Something has happened in system X that may be of interest elsewhere
   - Consider Azure Event Grid or Logic Apps
2. Informational data points from continuously published stream: IoT, etc
   - Logic often related to changes in pattern (such as sensor temperature rising) rather than individual data points
   - "Complex Event Processing" model
   - Consider Azure Event Hubs / IoT Hubs

azure.microsoft.com/blog/events-data-points-and-messages-choosing-the-right-azure-messaging-service-for-your-data

Container Basics

Azure container technology



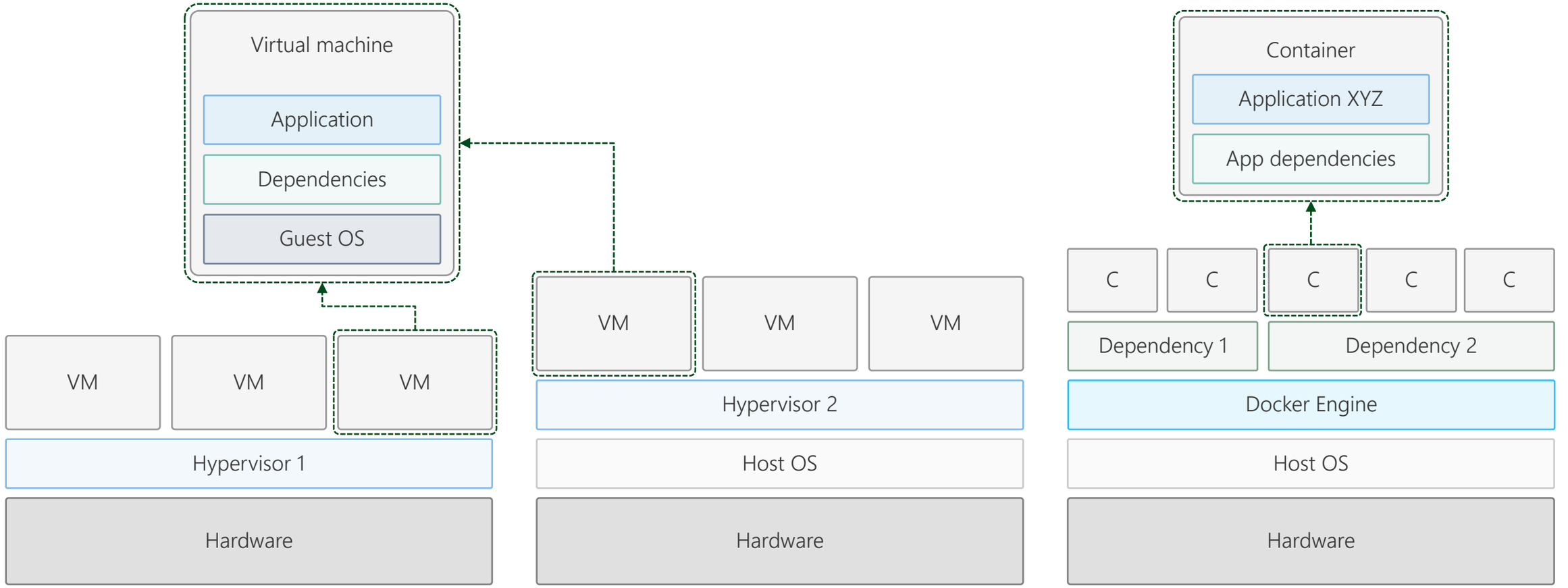


What Are Containers?

Containers wrap a piece of software in a complete filesystem that contains ***everything needed to run***: code, runtime, system tools, system libraries, etc.

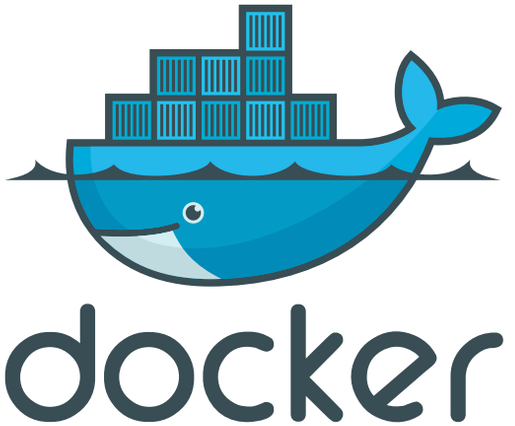
This guarantees that the software will always run the same, regardless of its environment

Virtualization versus **containerization**



Docker and Containers

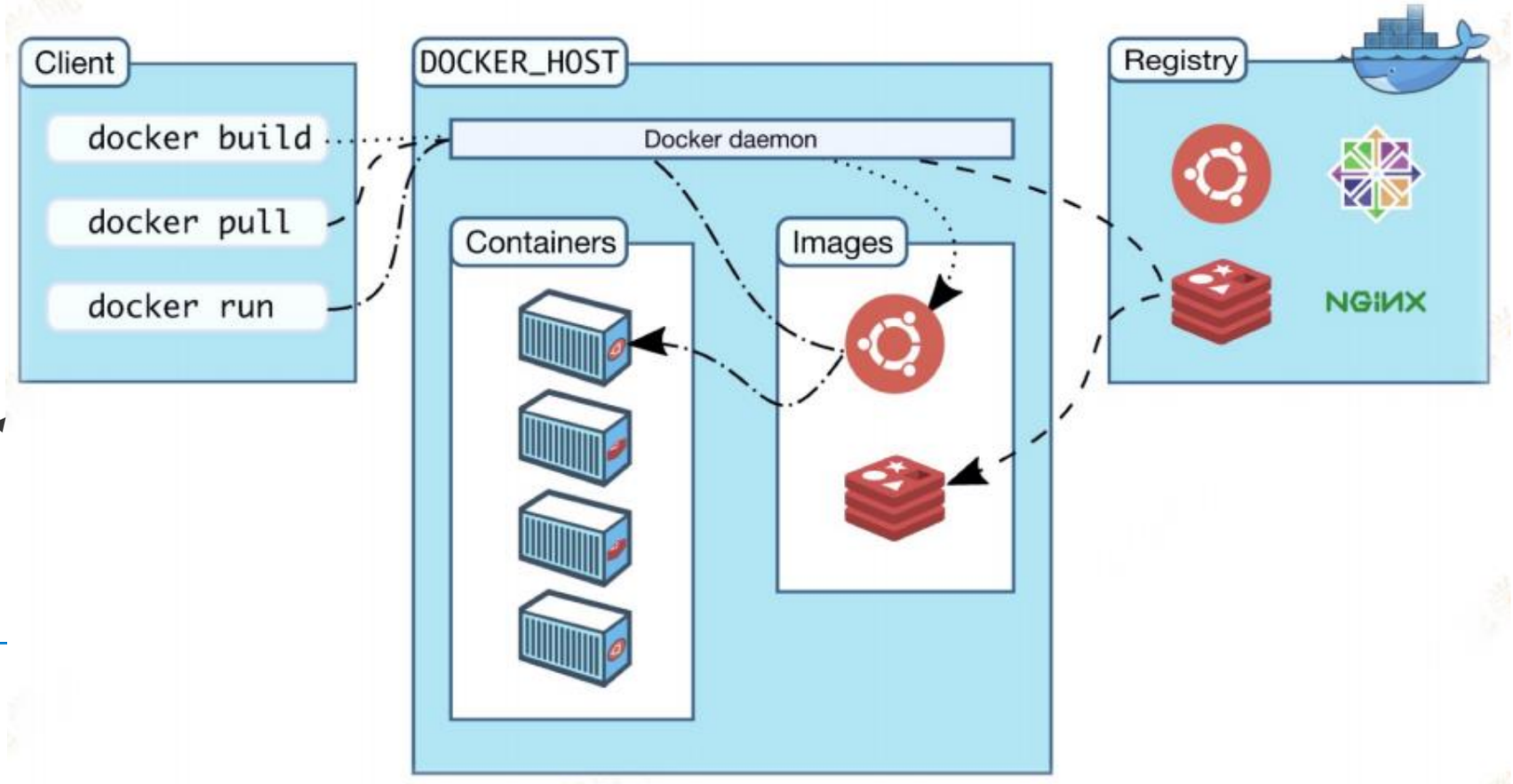
- Containers have been around for **many years** (80s)
  - Developed on top of the Linux Kernel (cgroups)
- **Docker Inc.** did not invent them
  - They created open source software to build and manage containers
- Docker made containers **easy**
- Docker is the de facto standard **container format** and set of **tools**
  - Docker CLI, Docker Engine, Docker Swarm, Docker Compose, Docker Machine etc. (more from Ben later…)



Running Docker

```
> docker image build --tag dockeronwindows/ch02-powershell-env .

Sending build context to Docker daemon 3.584kB
Step 1/3 : FROM microsoft/nanoserver
 ---> d9bccb9d4cac
Step 2/3 : COPY scripts/print-env-details.ps1 c:\\print-env.ps1
 ---> a44026142eaa
Removing intermediate container 9901221bbf99
Step 3/3 : CMD powershell.exe c:\print-env.ps1
 ---> Running in 56af93a47ab1
 ---> 253feb55a9c0
Removing intermediate container 56af93a47ab1
Successfully built 253feb55a9c0
Successfully tagged dockeronwindows/ch02-powershell-env:latest
```




```dockerfile
# Restore, build & publish in temp build image
FROM microsoft/aspnetcore-build:2.0 AS build
WORKDIR /src
COPY . .
WORKDIR /src/API
RUN dotnet restore
RUN dotnet publish -c Release -o /app

 # ---------------------------------------------

# Copy published binaries to final image
FROM microsoft/aspnetcore:2.0
WORKDIR /app
COPY --from=build /app .
ENTRYPOINT ["dotnet", "API.dll"]
```

What about Serverless?

Serverless Computing

Serverless computing is an event-driven application design and deployment paradigm in which computing resources are provided as scalable cloud services.

In a serverless computing deployment, the cloud customer only pays for service usage; there is never any cost associated with idle time.

Source - http://searchitoperations.techtarget.com/definition/serverless-computing

Why build Serverless applications on Azure?

Benefit from a fully managed service

Spare your teams the burden of managing servers. By utilising fully managed services, you can focus on your business logic and avoid administrative tasks. With serverless architecture, you simply deploy your code and it runs with high availability.

Scale flexibly

Serverless compute scales from nothing to handle tens of thousands of concurrent functions almost instantly (within seconds), to match any workload, and without requiring scale configuration – it reacts to events and triggers in near-real time.

Only pay for resources you use

With serverless architecture, you only pay for the time your code is running. Serverless computing is event-driven, and resources are allocated as soon as they're triggered by an event. You're only charged for the time and resources it takes to execute your code – through sub-second billing.

Azure Serverless Resources:

Functions, Storage, Cosmos DB, Event Grid, Service Bus, Logic Apps, Stream Analytics, Event Hub, Bot Service, Cognitive Services, …