BUILDING MICROSERVICE REST APIS USING AZURE FUNCTIONS

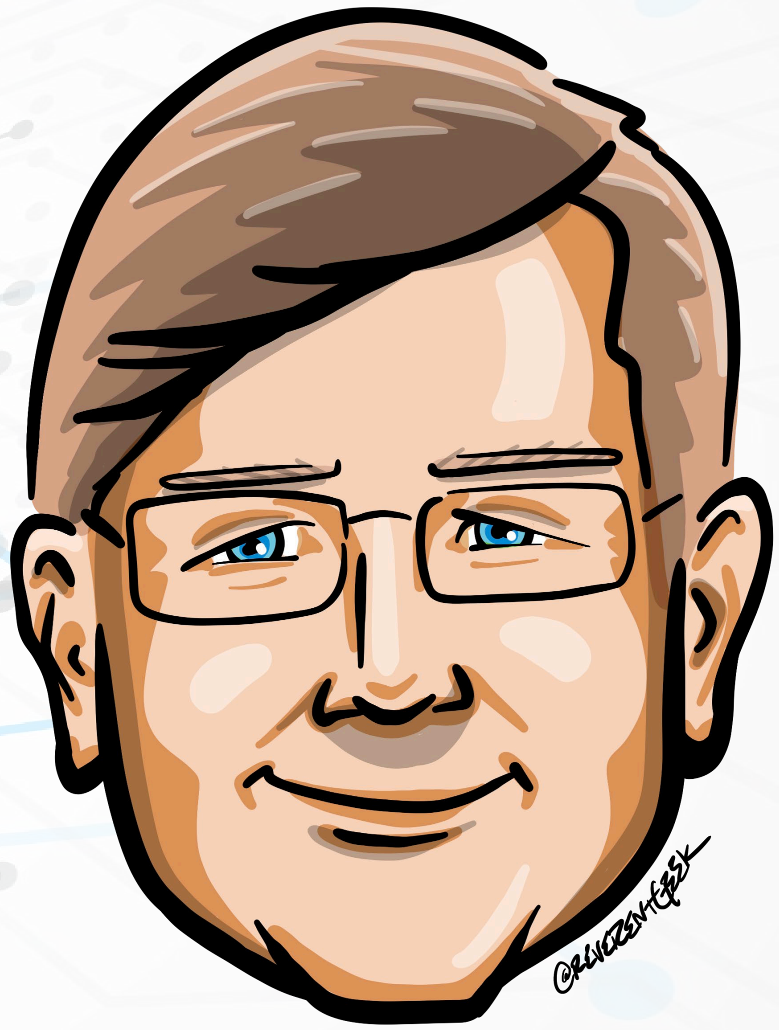
techbash

Who is Chad Green

Director of Architecture
Louisville, KY

glennis SOLUTIONS™

MVP Microsoft® Most Valuable Professional



Building Microservice REST APIs Using Azure Functions

TALELEARNCODE

CHADGREEN

Building Microservice REST APIs Using Azure Functions

Building Microservice REST APIs Using Azure Functions

Serverless

Microservice REST

APIs Serverless

Microservice REST

Azure Functions

APIs Serverless

Microservice

REST

APIs

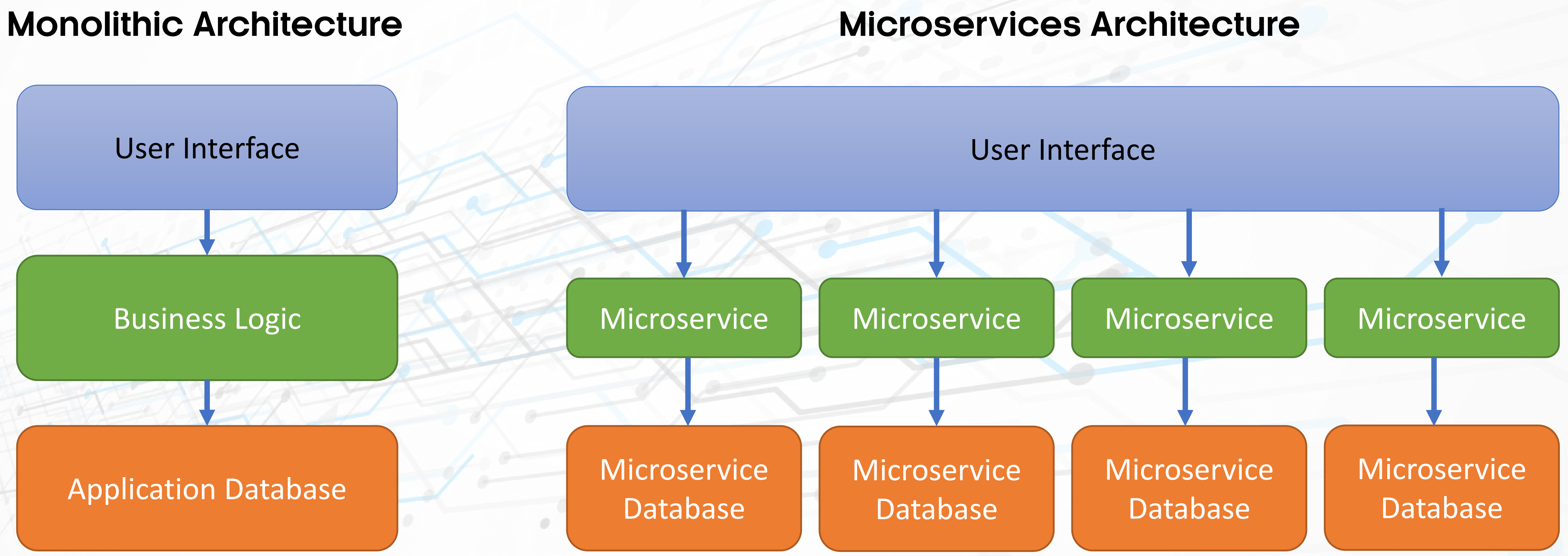
Serverless

Azure Functions

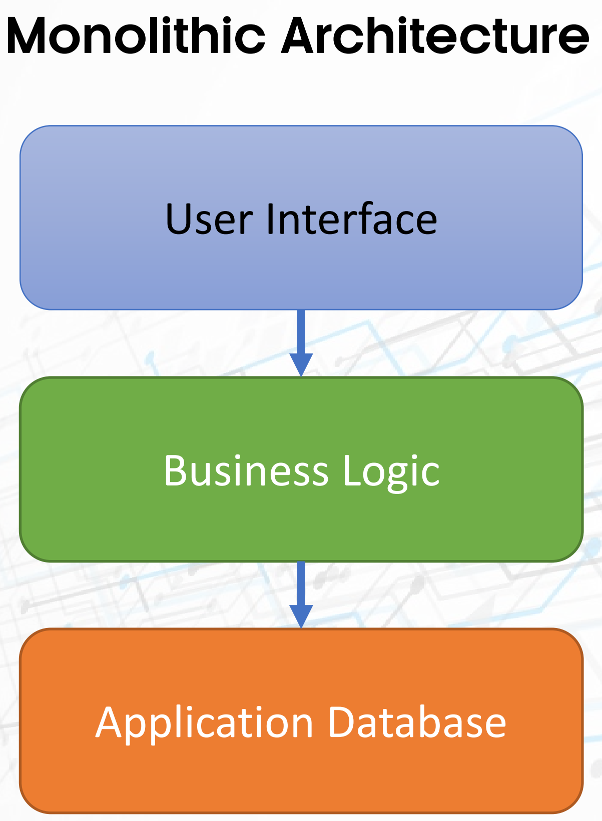
What are microservices?

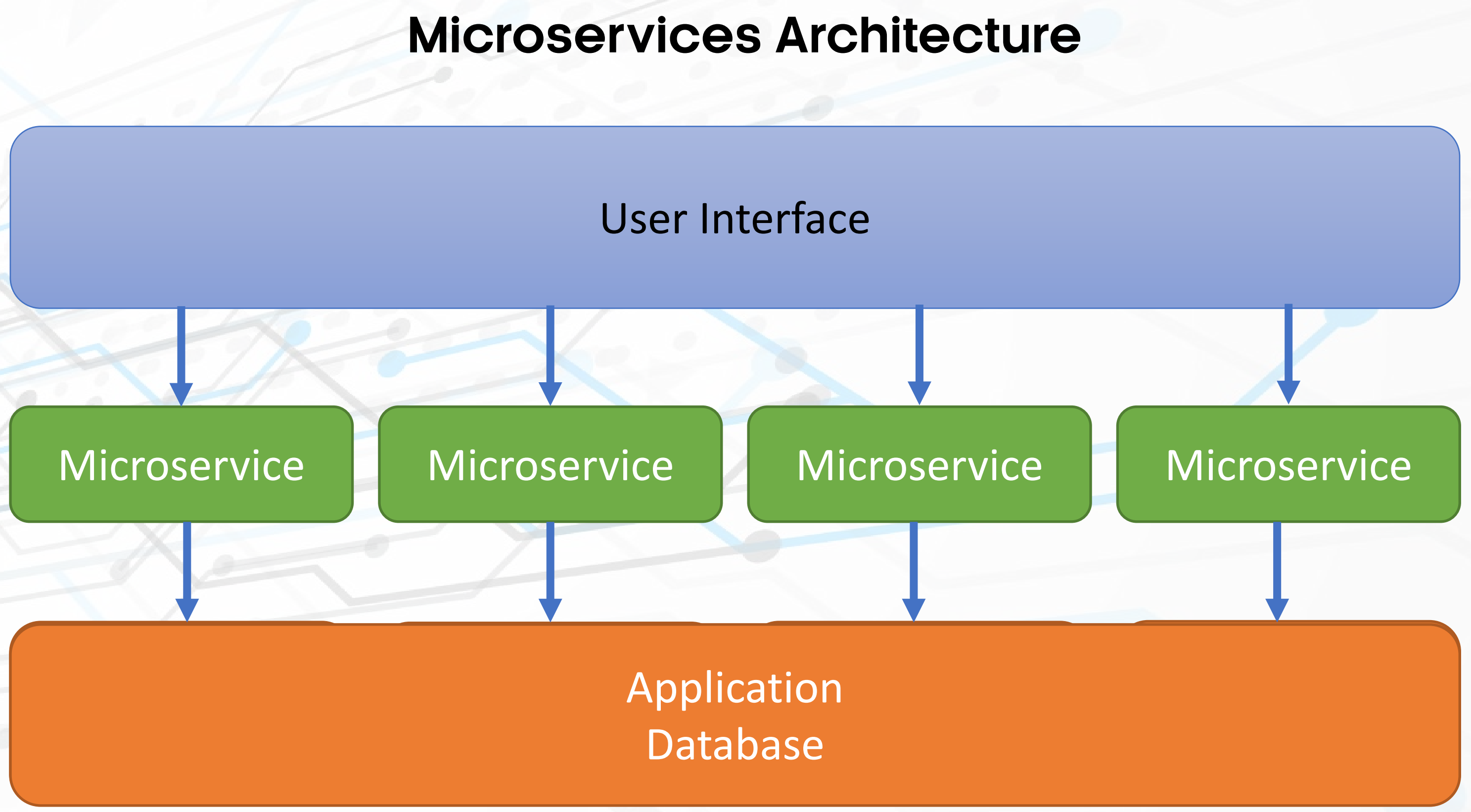
Building Microservice REST APIs Using Azure Functions

What are microservices?



What are microservices?





Advantages of Microservices

Quick

Advantages of Microservices

Quick

Quick

Advantages of Microservices

Quick

Resilience

Advantages of Microservices

Quick

Resilience

Advantages of Microservices

Quick

Resilience

Scalability

Advantages of Microservices

Quick

Resilience

Scalability

Advantages of Microservices

Quick

Resilience

Scalability

Maintainability

Advantages of Microservices

Quick

Resilience

Scalability

Flexibility

Advantages of Microservices

Quick

Resilience

Scalability

Maintaina

Flexibility

Advantages of Microservices

Quick

Resilience

Scalability

Maintaina

Flexibility

Advantages of Microservices

Quick

Resilience

Scalability

Maintainability

Flexibility

REST

Building Microservice REST APIs Using Azure Functions

Introduction to REST

REST

REpresentational State Transfer

Introduction to REST

REST

sentational **S**tate **T**ransfer

Introduction to REST

REST

REpresentational **S**tate **T**ransfer

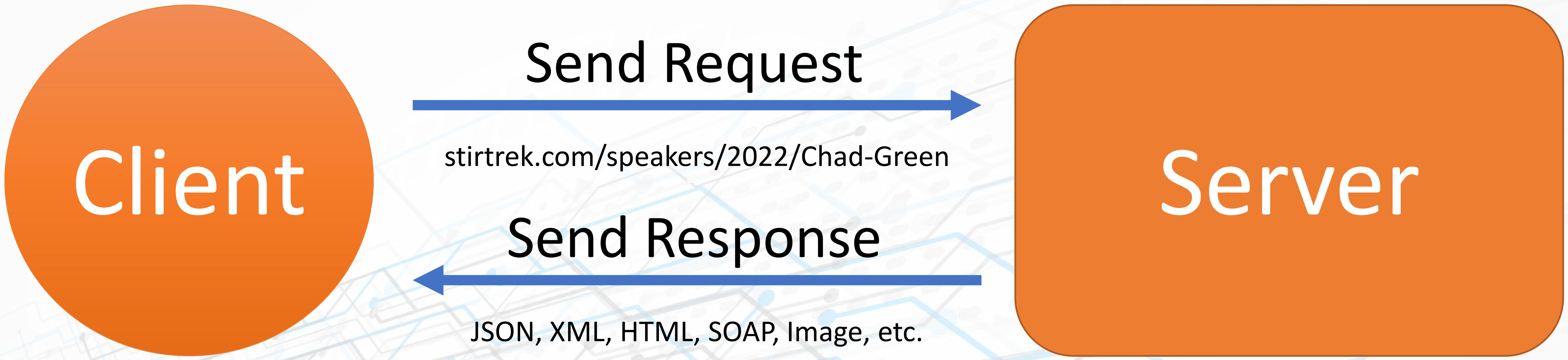
Introduction to REST

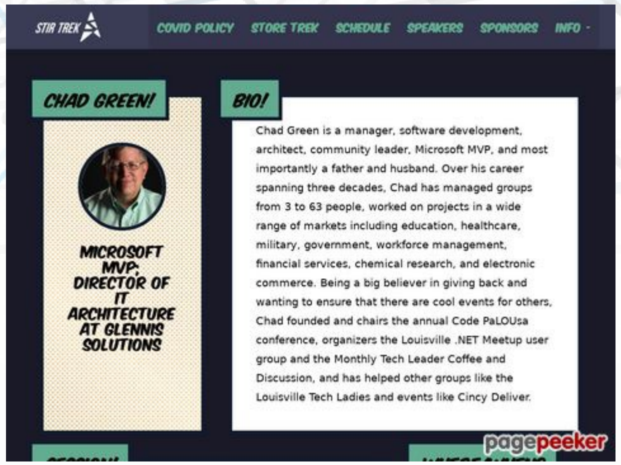
REST

Server

REpresentational **S**tate **T**ransfer

Introduction to REST


Client
Send Request
stirtrek.com/speakers/2022/Chad-Green
Send Response
JSON, XML, HTML, SOAP, Image, etc.
Server



CHAD GREEN!

MICROSOFT MVP; DIRECTOR OF IT ARCHITECTURE AT GLENNIS SOLUTIONS

BIO!

Chad Green is a manager, software development, architect, community leader, Microsoft MVP, and most importantly a father and husband. Over his career spanning three decades, Chad has managed groups from 3 to 63 people, worked on projects in a wide range of markets including education, healthcare, military, government, workforce management, financial services, chemical research, and electronic commerce. Being a big believer in giving back and wanting to ensure that there are cool events for others, Chad founded and chairs the annual Code PaLOUsa conference, organizers the Louisville .NET Meetup user group and the Monthly Tech Leader Coffee and Discussion, and has helped other groups like the Louisville Tech Ladies and events like Cincy Deliver.

REST Architectural Constraints

Client-Server Architecture

REST Architectural Constraints

Client-Server Architecture

Statelessness

REST Architectural Constraints

Client-Server Architecture

Statelessness

Cacheability

REST Architectural Constraints

Client-Server Architecture

Statelessness

Cacheability

Layered System

REST Architectural Constraints

- Client-Server Architecture
- Statelessness
- Cacheability
- Layered System
- Code on Demand (optional)

REST Architectural Constraints

- Client-Server Architecture
- Statelessness
- Cacheability
- Layered System
- Code on Demand (optional)
- Uniform Interface

REST Verbs

POST

Create

```
POST https://exampleapi.com/employees
```

	HTTP Status Code	Description
Success	201	Created
Failure	400	Bad Request
	409	Conflict

REST Verbs

POST
Create

GET
Read

```
GET https://exampleapi.com/employees/{id}
```

	HTTP Status Code	Description
Success	200	OK
Failure	400	Bad Request
	404	Failure

REST Verbs

POST	GET	PUT
Create	Read	Update

```
PUT https://exampleapi.com/employees/{id}
```

	HTTP Status Code	Description
Success	204	No Content
	201	Created
Failure	400	Bad Request
	404	Failure

REST Verbs

POST	GET	PUT	PATCH
Create	Read	Update	~Update

```
PATCH https://exampleapi.com/employees/{id}
```

	HTTP Status Code	Description
Success	204	No Content
Failure	400	Bad Request
	404	Failure

REST Verbs

POST	GET	PUT	PATCH	DELETE
Create	Read	Update	~Update	Delete

```
DELETE https://exampleapi.com/employees/{id}
```

	HTTP Status Code	Description
Success	200	OK
Failure	400	Bad Request
	404	Failure

What is Serverless?

Building Microservice REST APIs using Azure Functions

The evolution of application platforms

On-Premises

What media should I use to keep **backups?**

What **size** of **servers** should I **buy**?

How do I **deploy** new **code** to my **servers**?

What happens in case of server hardware **failure?**

How can I increase **server** utilization?

Who **monitors** my **apps**?

How often should I **backup** my **server?**

How can I **scale** my app?

Which packages should be on my **server**?

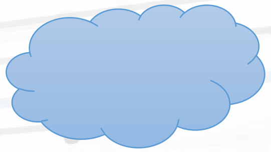
Are my **servers** in a **secure** location?

Do I need a secondary **Network connection?**

How do I keep the **operating system** up to date?

Who has **physical** access my **servers?**

Who **monitors** my **servers**?



What **storage** do I need to use?

How **many** servers do I need?

What happens in case of server hardware **failure?**

Do I need a **UPS**?

How can I dynamically configure my app?

It takes how long to **provision** a new **server?**

How often should I **patch** my **severs?**

The evolution of application platforms

IaaS

What media should I use to keep **backups?**

What **size** of **servers** should I **buy**?

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

How can I increase **server** utilization?

How do I **deploy** new **code** to my **servers?**

How **many** servers do I need?

How can I **scale** my application?

Who **monitors** my **apps**?

What happens in case of server hardware **failure?**

How can I increase **server** utilization?

How often should I **backup** my **server?**

Are my **servers** in a **secure** location?

How can I **scale** my app?

Which packages should be on my **server**?

Do I need a secondary **Network connection?**

How often should I **patch** my **servers**?

Who has **physical** access my **servers?**

How often should I **backup** my **server**?

How **many** servers do I need?

Which **packages** should be on my **server**?

Who **monitors** my **servers**?

How do I keep the **operating system** up to date?

How do I **deploy** new **code** to my **server**?

How do I keep the **operating system** up to date?

What **storage** do I need to use?

Who **monitors** my application?

What happens in case of server hardware **failure?**

How can I dynamically configure my app?

Do I need a **UPS**?

It takes how long to **provision** a new **server?**

How often should I **patch** my **severs?**

The evolution of application platforms

PaaS

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

How can I increase **server** utilization?

How **many** servers do I need?

How can I **scale** my application?

How often should I **patch** my **servers**?

How often should I **backup** my **server**?

Which **packages** should be on my **server**?

How do I **deploy** new **code** to my **server**?

How do I keep the **operating system** up to date?

Who **monitors** my application?

The evolution of application platforms

Serverless

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

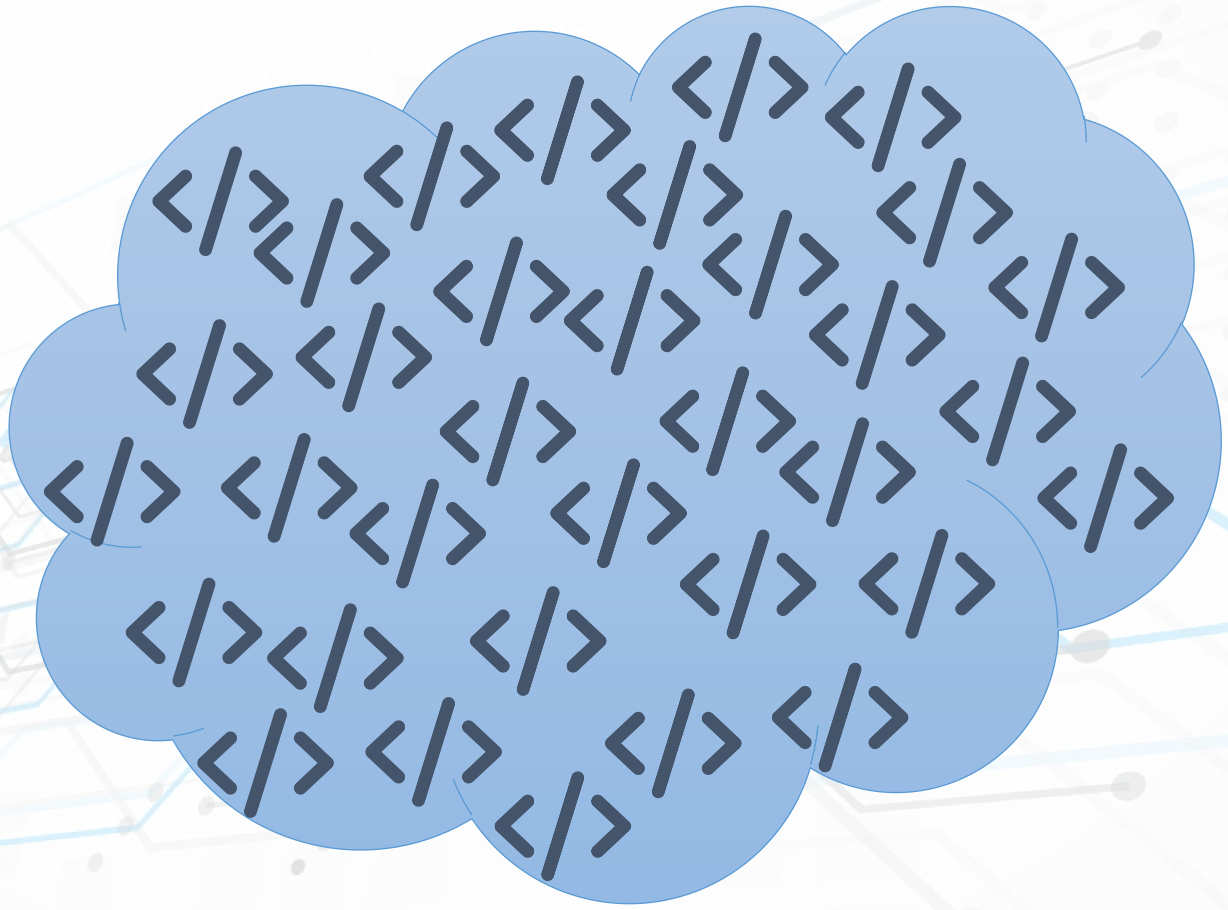
How can I increase **server** utilization?

How **many** servers do I need?

How can I **scale** my application?

The evolution of application platforms

Serverless



Not there isn't servers

Just, you can think about the servers less

~~Server Configuration~~

~~Server Scaling~~

Serverless Benefits

No Server Management

Serverless Benefits

No Server Management

Serverless Benefits

No Server Management

Simplified Scalability

Serverless Benefits

No Server Management

Simplified Scalability

Serverless Benefits

No Server Management

Simplified Scalability

Lower Costs

Serverless Benefits

No Server Management

Simplified Scalability

Lower Costs

Serverless Benefits

No Server Management

Simplified Scalability

Lower Costs

Quicker Turnaround

Serverless Benefits

No Server Management

Simplified Scalability

Lower Costs

Quicker Turnaround

Serverless Benefits

No Server Management

Simplified Scalability

Lower Costs

Quick TUrnaround

Simplified Code

Serverless Benefits

No Server Management

Simplified Scalability

Lower Costs

Quick Turnaround

Simplified Code

Serverless Benefits

No Server Management

Simplified Scalability

Lower Costs

Quicker Turnaround

Simplified Code

Disadvantages of Serverless

Testing Challenges

Disadvantages of Serverless

Testing Challenges

Disadvantages of Serverless

Testing Challenges

Security Concerns

Disadvantages of Severless

Testing Challenges

Security Concerns

Disadvantages of Serverless

Testing Challenges

Security Concerns

Short –Running Processes

Disadvantages of Serverless

No Server Management

Simplified Scalability

Short-Running Processes

Disadvantages of Serverless

- Testing Challenges
- Security Concerns
- Short-Running Processes
- Cold Starts

Disadvantages of Serverless

Testing Challenges

Security Concerns

Short-Running Processes

Cold Starts

Disadvantages of Serverless

Testing Challenges

Security Concerns

Short-Running Processes

Cold St

Vendor Lock-In

Disadvantages of Serverless

Testing Challenges

Security Concerns

Short-Running Processes

Cold St

Vendor Lock-In

Disadvantages of Serverless

- Testing Challenges
- Security Concerns
- Short-Running Processes
- Cold Starts
- Vendor Lock-In

Azure Functions

Building Microservice REST APIs Using Azure Functions

Features of Azure Functions

Choice of Language

C# JS Java python™

Features of Azure Functions

Choice of Language

Pay-Per-Use Pricing

Consumption Plan

Features of Azure Functions

Choice of Language

Pay-Per-Use Pricing

Consumption Plan

Premium Plan

Features of Azure Functions

Choice of Language

Pay-Per-Use Pricing

Consumption Plan

Premium Plan

Azure App Service Plan

Features of Azure Functions

Choice of Language

Pay-Per-Use Pricing

Bring Your Own Dependencies

Features of Azure Functions

- Choice of Language
- Pay-Per-Use Pricing
- Bring Your Own Dependencies
- Integrated Security

Features of Azure Functions

Choice of Language

Pay-Per-Use Pricing

Bring Your Own Dependencies

Integrated Security

Simplified Integration

Features of Azure Functions

- Choice of Language
- Pay-Per-Use Pricing
- Bring Your Own Dependencies
- Integrated Security
- Simplified Integration
- **Flexible Development**

Features of Azure Functions

Choice of Language

Pay-Per-Use Pricing

Bring Your Own Dependencies

Integrated Security

Simplified Integration

Flexible Development

Open Source

https://github.com/Azure/Azure-Functions

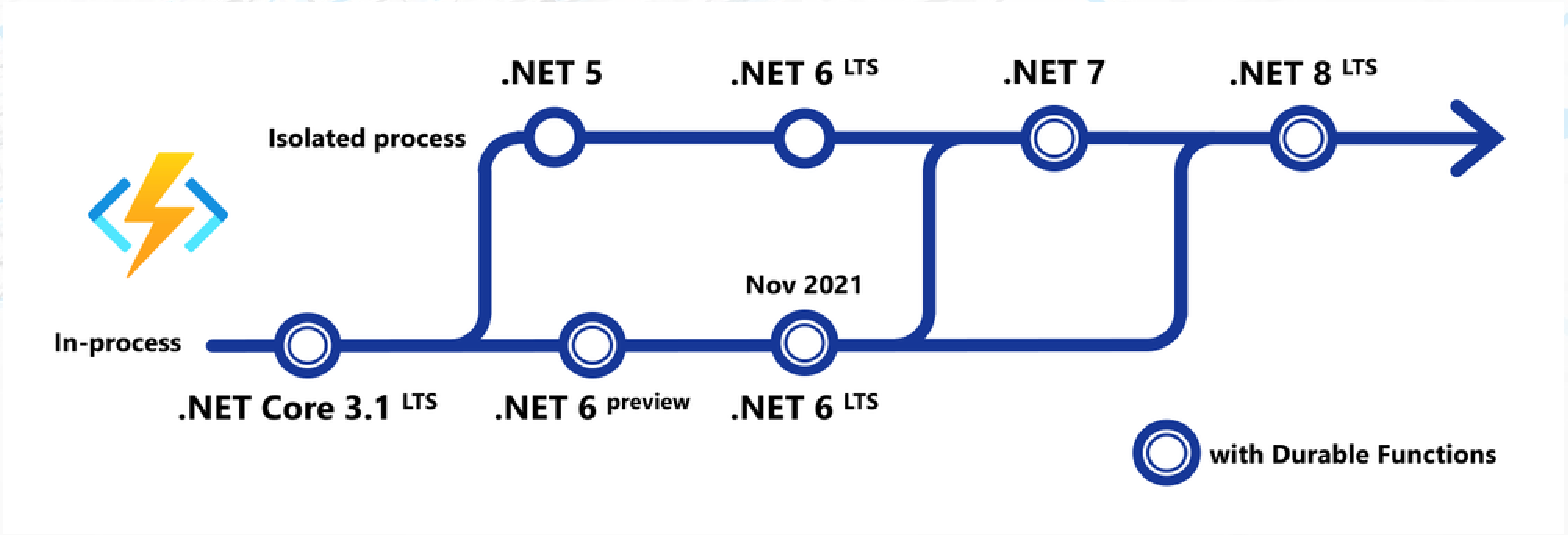
Triggers and Bindings

- Blob Storage
- Cosmos DB
- Event Grid
- Event Hubs
- External Table
- HTTP
- Microsoft Graph Excel tables
- Microsoft Graph OneDrive Files
- Microsoft Graph Outlook email
- Microsoft Graph Events
- Microsoft Graph Auth tokens
- Mobile Apps
- Notification Hubs
- Queue Storage
- SendGrid
- Sever Bus
- Table Storage
- Timer
- Twilio

C# Process Models

In-Process

Isolated Process



Code Walkthrough

Building Microservice REST APIs Using Azure Functions

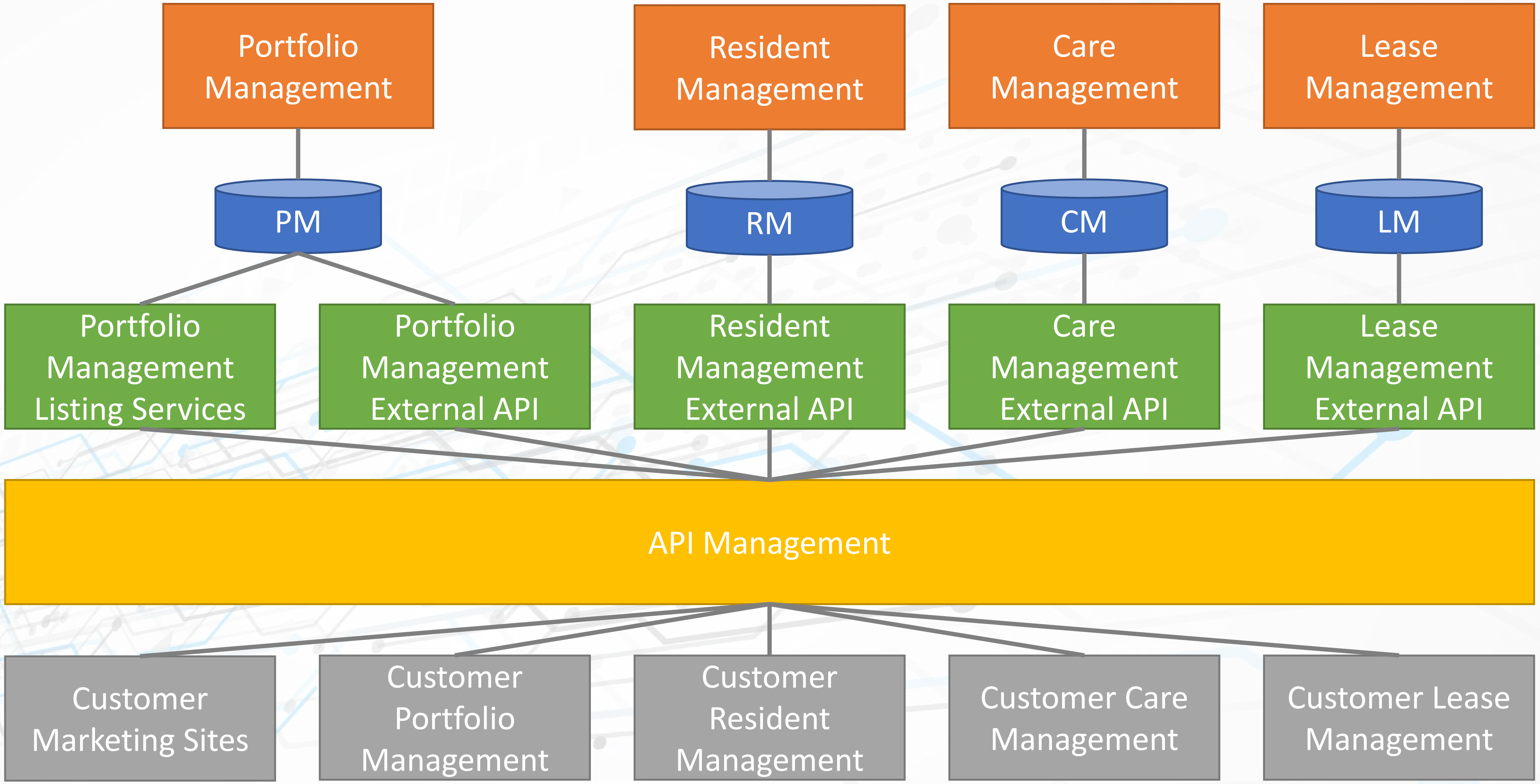
EST. 2022

SENIOR LIVING

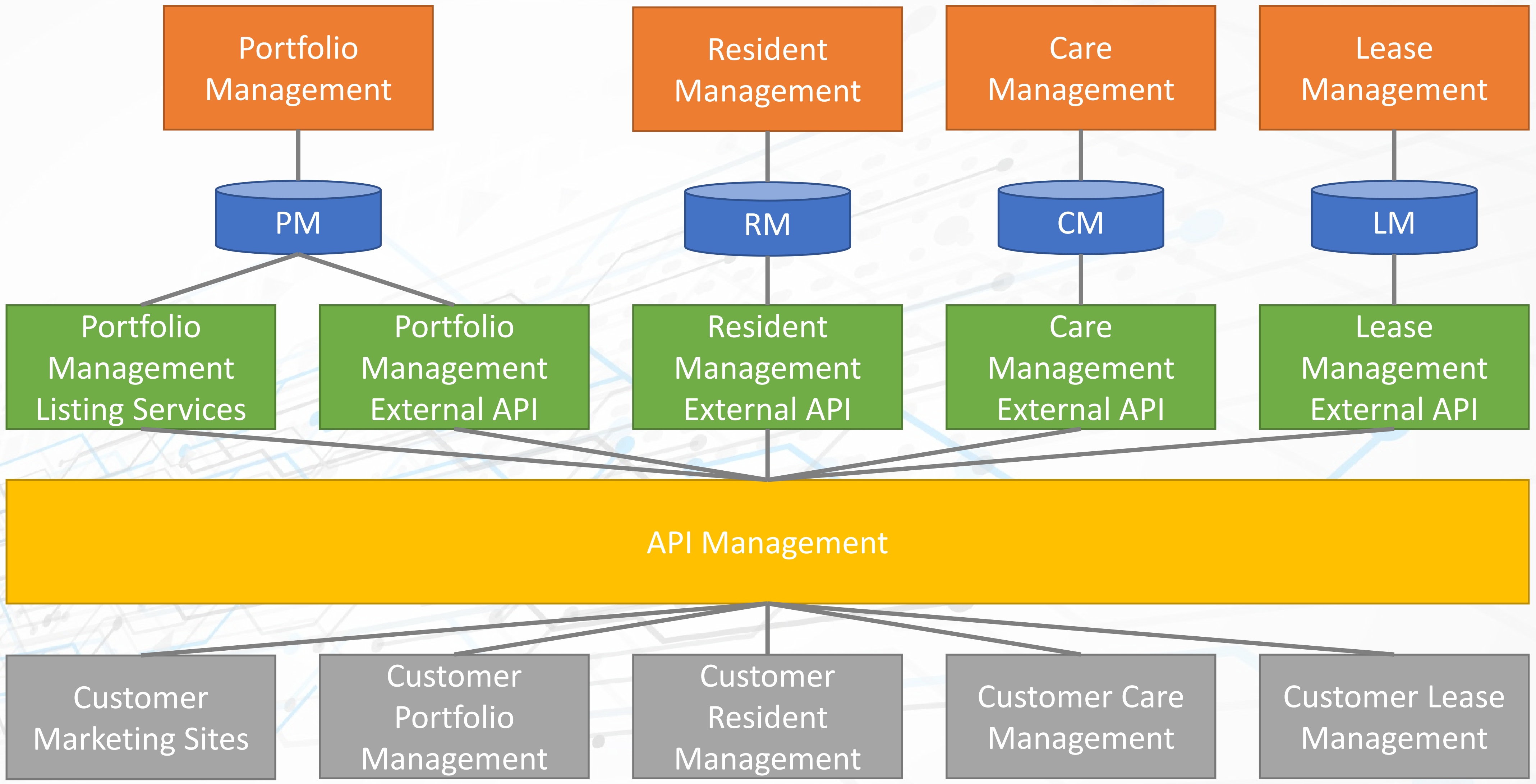
SYSTEMS

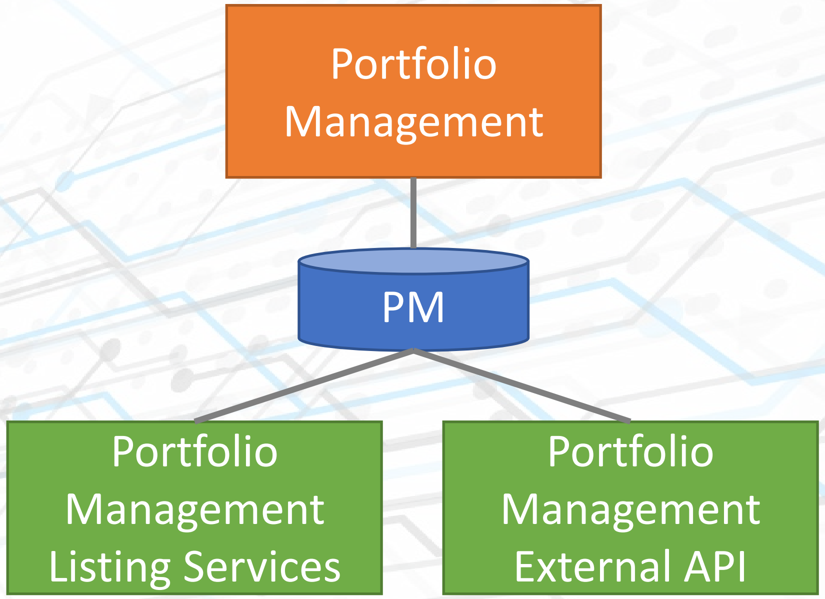
EST. 2022
SENIOR LIVING
SYSTEMS

EST. 2022
SENIOR LIVING
SYSTEMS



Portfolio Management
Resident Management
Care Management
Lease Management
PM
RM
CM
LM
Portfolio Management Listing Services
Portfolio Management External API
Resident Management External API
Care Management External API
Lease Management External API
API Management
Customer Marketing Sites
Customer Portfolio Management
Customer Resident Management
Customer Care Management
Customer Lease Management


Portfolio Management
Resident Management
Care Management
Lease Management
PM
RM
CM
LM
Portfolio Management Listing Services
Portfolio Management External API
Resident Management External API
Care Management External API
Lease Management External API
API Management
Customer Marketing Sites
Customer Portfolio Management
Customer Resident Management
Customer Care Management
Customer Lease Management


Portfolio Management
PM
Portfolio Management Listing Services
Portfolio Management External API

Serverless Microservice REST API Demo

CRUD Operations using HTTP Trigger

Serverless Microservice REST API Demo

Generating Open API Details for Endpoints

Serverless Microservice REST API Demo

Using the Azure SQL Bindings

Best Practices

Building Microservice REST APIs Using Azure Functions

Best Practices

Avoid long running functions

Best Practices

Avoid cross functional communication

Best Practices

Write functions to be stateless

Best Practices

Organize functions for performance and scaling

Best Practices

Share and manage connections

Best Practices

Avoid sharing storage accounts

Best Practices

Use async code but avoid blocking calls

Thank You

chadgreen@chadgreen.com

TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen