
PROLOGUE
Design and Develop a
Serverless Event-Driven
Microservice-Based Solution

Introduction to Serverless

Beyond Traditional Infrastructure

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?**

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

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

How can I increase **server** utilization?

How **many** servers do I need?

How can I **scale** my application?

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

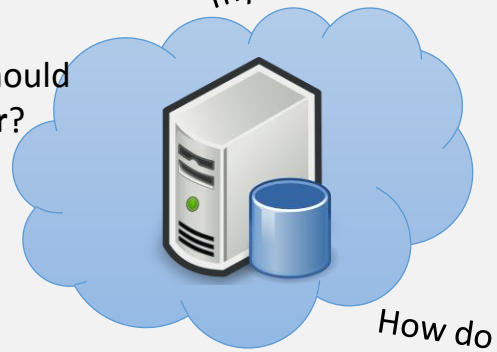
What happens in case of server hardware **failure?**

How can I increase **server** utilization?

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

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

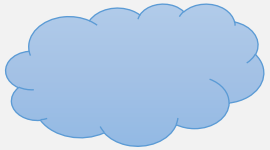
How often should I **backup** my **server**?

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

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

How **many** servers do I need?

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?

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

Who **monitors** my application?

How can I dynamically configure my app?

Do I need a **UPS**?

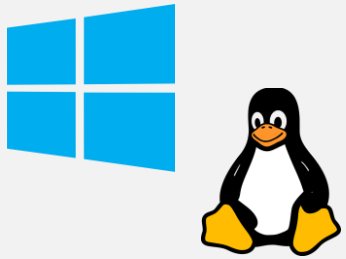
It takes how long to **provision** a new **server?**

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

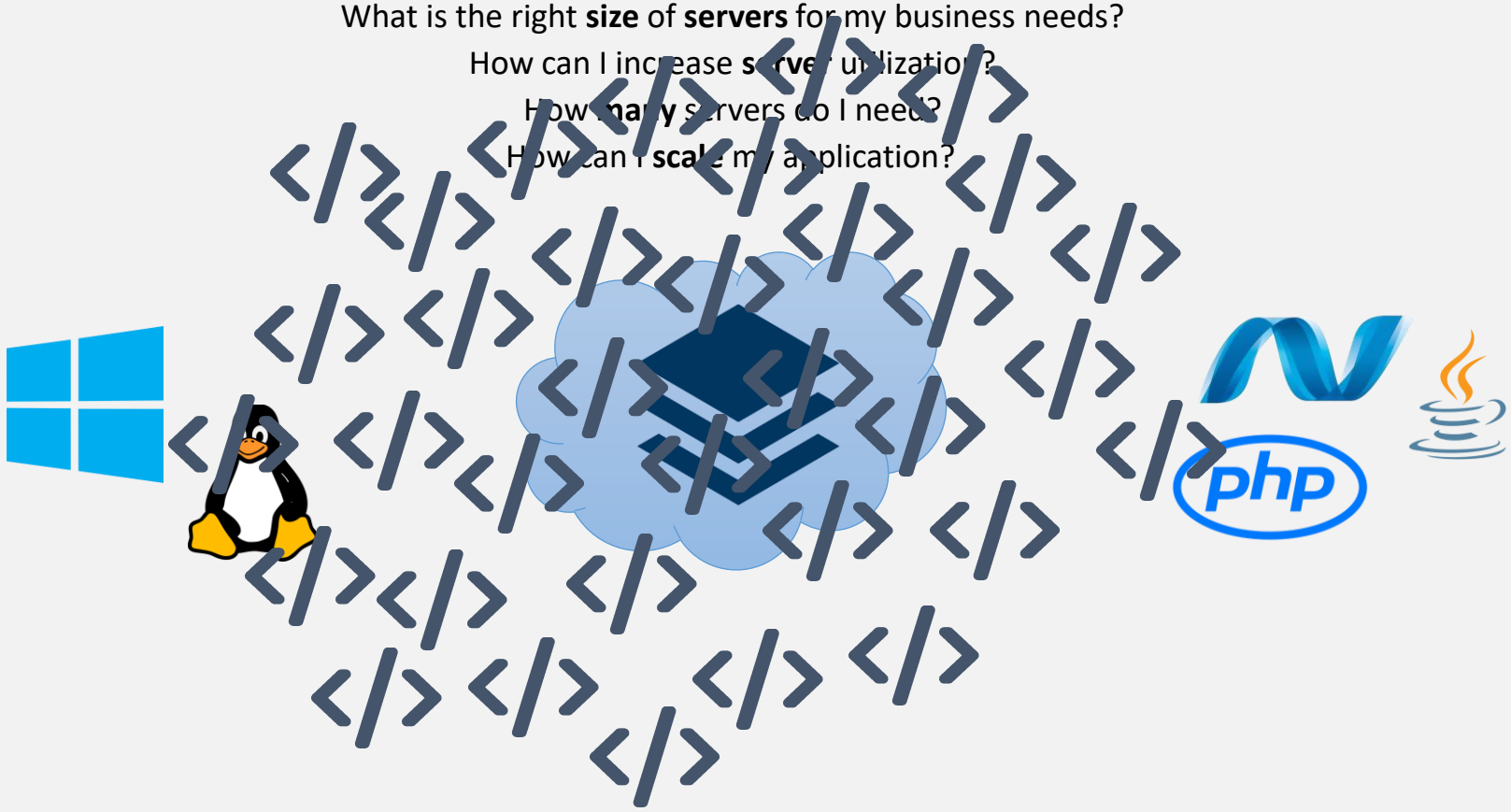
Serverless

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?



What is Serverless?

Serverless computing is a cloud computing model where the cloud provider dynamically manages the allocation and provisioning of servers.

“What’s in a name?”

Not there isn’t servers

Just, you can think about the servers less

~~Server Configuration~~

~~Server Scaling~~

Types of Serverless Architecture

Function as a Service (FaaS)

Backend as a Service (BaaS)

Function-as-a-Service

Event-Driven

Function-as-a-Service

Event-Driven

Short-Lived

Function-as-a-Service

Event-Driven

Short-Lived

Automatic Scaling

Function-as-a-Service

Event-Driven

Short-Lived

Automatic Scaling

Pay-Per-Execution

Function-as-a-Service

Event-Driven

Short-Lived

Automatic Scaling

Pay-Per-Execution

Abstraction of Infrastructure

Benefits

Cost Efficiency

Benefits

Cost Efficiency

Auto-Scaling

Benefits

Cost Efficiency

Auto-Scaling

Reduced Operational Overhead

Benefits

- Cost Efficiency
- Auto-Scaling
- Reduced Operational Overhead
- **Faster Time-to-Market**

Benefits

Cost Efficiency

Auto-Scaling

Reduced Operational Overhead

Faster Time-to-Market

Trigger-Driven Architecture

Benefits

Cost Efficiency

Auto-Scaling

Reduced Operational Overhead

Faster Time-to-Market

Trigger-Driven Architecture

High Availability

Benefits

Cost Efficiency

Auto-Scaling

Reduced Operational Overhead

Faster Time-to-Market

Trigger-Driven Architecture

High Availability

Micro-Billing

Challenges

Loss of Control

Challenges

Loss of Control

Cold Starts

Challenges

Loss of Control

Cold Starts

Usage-Based Pricing

Challenges

Loss of Control

Cold Starts

Usage-Based Pricing

Provider Lock-In

Challenges

Loss of Control

Cold Starts

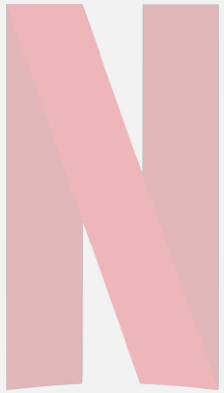
Usage-Based Pricing

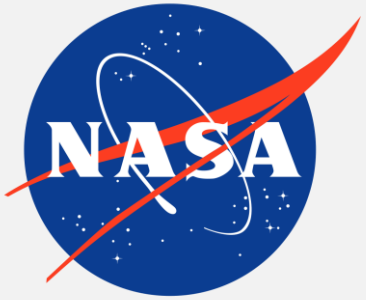
Provider Lock-In

Testing and Debugging

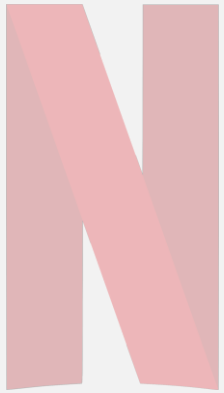
Implementation Examples

Implementation Examples

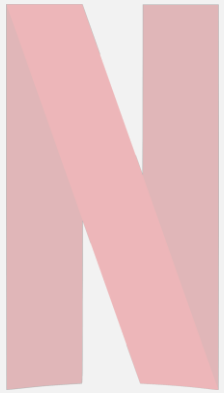




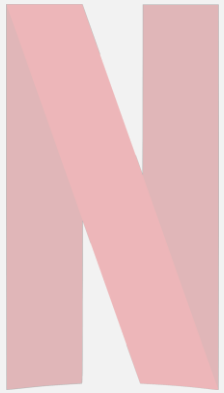
Implementation Examples



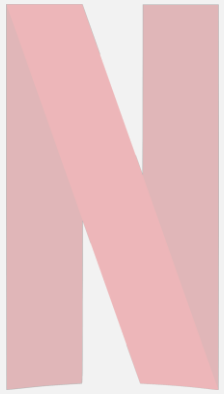
Implementation Examples



Implementation Examples



Implementation Examples



Conclusion

- Serverless Architecture offers numerous benefits like cost efficiency and easy scalability.
- However, it presents challenges like vendor lock-in and the cold start problem.
- Despite these challenges, serverless is becoming popular for many organizations due to its numerous advantages.

Overview of Azure Serverless

Harnessing the Power of Microservice Azure