TECHNISCHE UNIVERSITÄT WIEN

Data-intensive Computing

2.0 VU / 3.0 ECTS, 2024S

Introduction to Cloud + Serverless + Exercise 3

Shashi Ilager
shashikant.ilager@tuwien.ac.at

TU Wien
Institute of Information Systems Engineering
Data Science Research Unit

Favoritenstraße 9-11/194-1,
1040 Vienna, Austria

Trends in Distributed Systems

Distributed systems has been undergoing a period of significant change and this can be traced back to a number of influential trends:

- The emergence of **pervasive networking** technology
  - **Pervasive** means "diffused throughout every part of."

- The emergence of **ubiquitous computing** coupled with the desire to support user mobility in distributed systems;
  - **Ubiquitous** means everywhere

- The increasing demand for **multimedia services**.

- The view of distributed systems as **utility**.

Distributed Computing as Utility

Distributed resources can be viewed as a commodity or utility similar to *water* or *electricity*.

Resources are provided by appropriate service suppliers and **rented** rather than **owned** by an end user.

The term **cloud computing** is used to capture this vision.

Cloud Computing Definitions

A report from the ***University of California Berkeley [1]*** defines cloud as the "data center hardware and software that provide services"

summarized the key characteristics of cloud computing as:

> "(1) the illusion of infinite computing resources;
>
> (2) the elimination of an up-front commitment by cloud users; and
>
> (3) the ability to pay for use ... as needed ..."

The National Institute of Standards and Technology (NIST):

"Cloud computing is a model for enabling **ubiquitous**, **convenient**, **on-demand** network access to a **shared pool** of configurable **computing resources** (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, and R. Katz, Above the Clouds: A Berkeley View of Cloud Computing, UC Berkeley Reliable Adaptive Distributed Systems Laboratory White Paper, 2009.

Cloud Computing

An IT paradigm that enables access to **shared pools** of configurable system resources in form of **services** that can be rapidly provisioned with minimal management effort, often over the **Internet**.

Why Clouds?

Conventional Computing

Every 18 months?

- Buy & Own
  - Hardware, System Software, Applications
  - Often to meet peak needs.
- Install, Configure, Test, Verify, Evaluate, Manage
- …
- Finally, use it!
- $$$$....$(VERY expensive!)

Cloud Computing

- Subscribe
- Use

- $ - pay for what you use, based on Quality of Service (QoS).

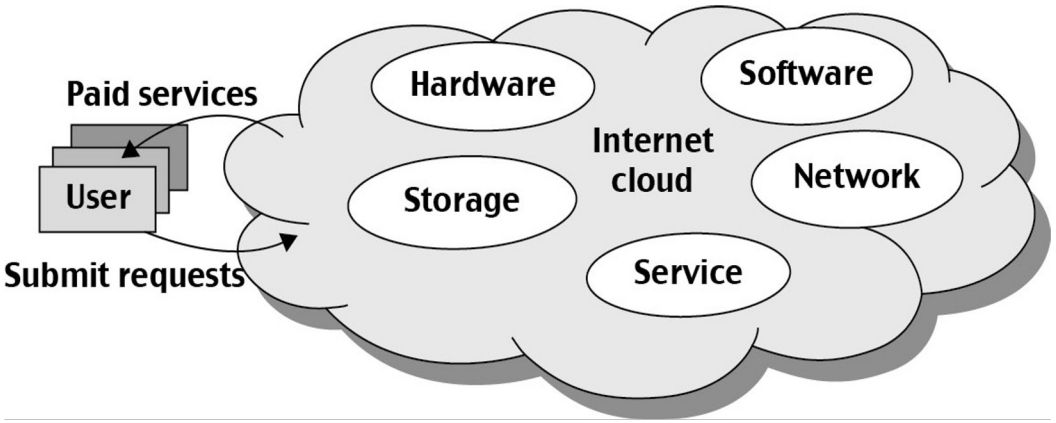
(Courtesy of Rajkumar Buyya, 2012)

Basic Concept

Cloud is the “invisible” backend to many applications

- e.g., Dropbox, Linkedlin, Github, Office 365, Amazon, Office365, Slack,…

The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure.

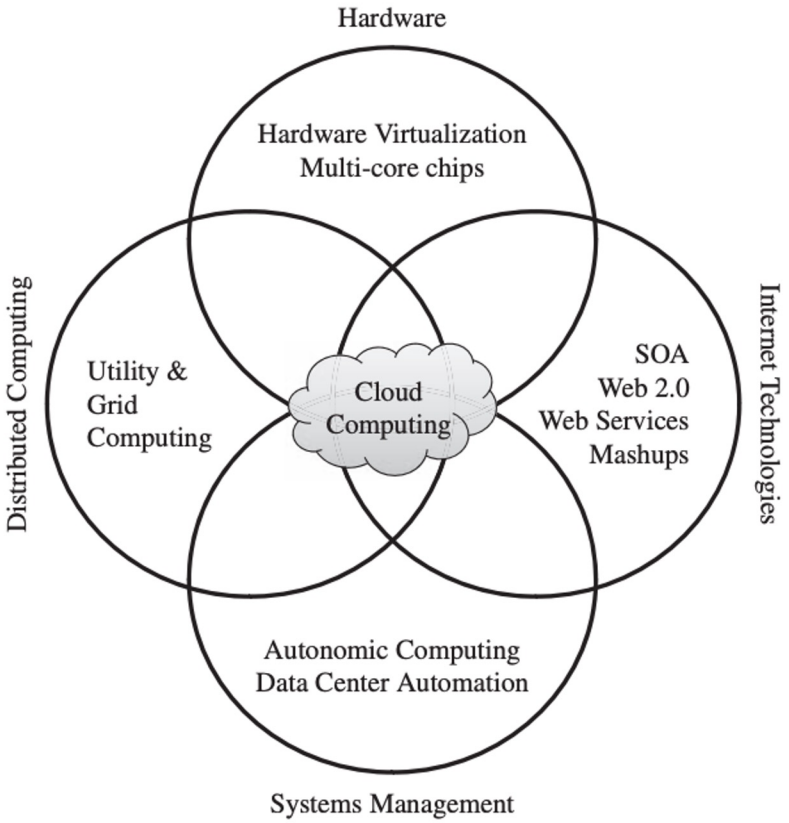


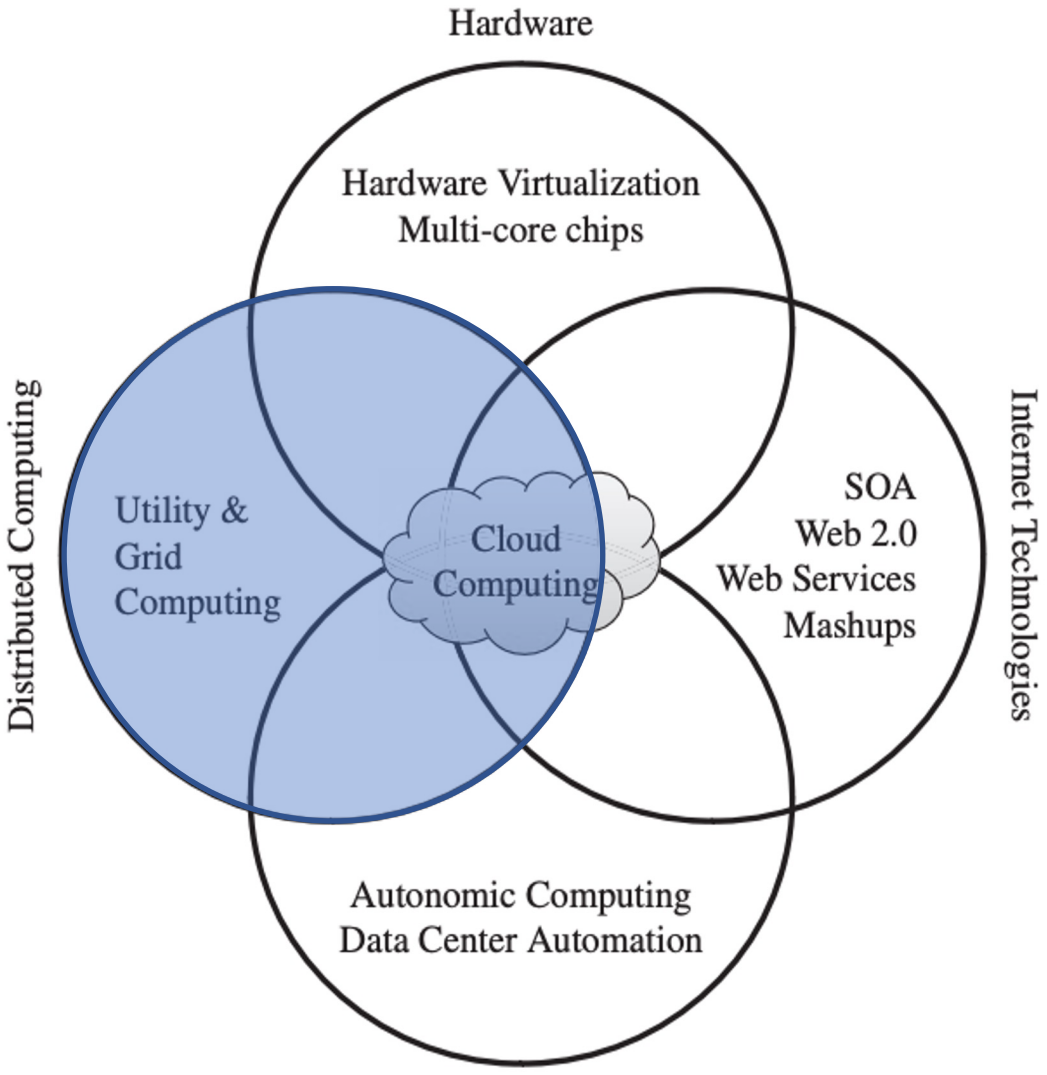
The long-held dream of computing as a utility

- The long-held dream of delivering computing as a utility has been realized with the advent of **cloud computing**

- Over the years, **several technologies** have matured and significantly contributed to make cloud computing viable

    - Utility & Grid Computing
    - Webservices
    - Virtualization
    - Autonomic Computing



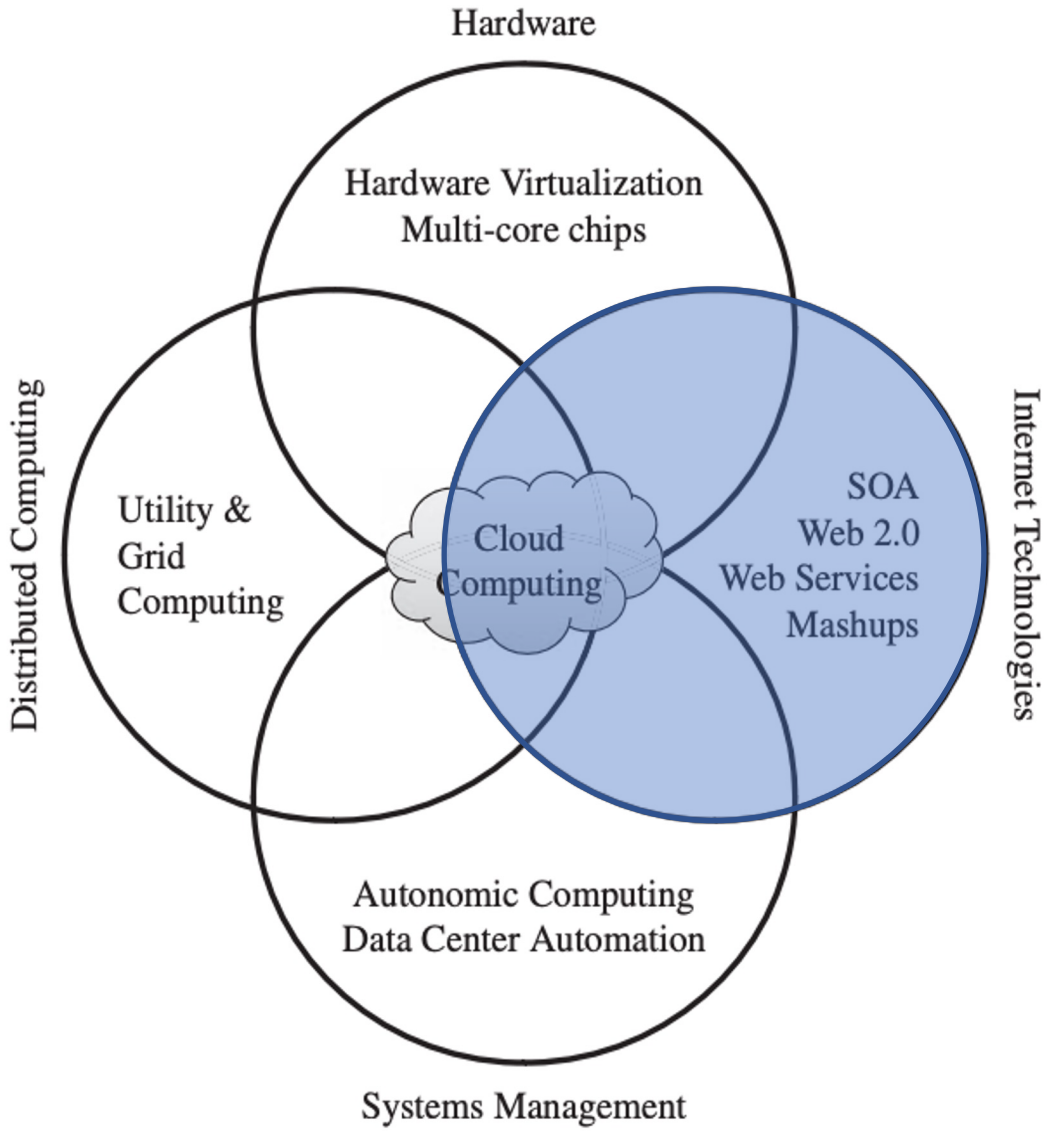

Hardware
Hardware Virtualization
Multi-core chips
Distributed Computing
Utility &
Grid
Computing
Cloud
Computing
SOA
Web 2.0
Web Services
Mashups
Internet Technologies
Autonomic Computing
Data Center Automation
Systems Management

Alessandro Volta in Paris in 1801 inside French National Institute shows the battery while in the presence of Napoleon I

Fresco by N. Cianfanelli (1841)

(Zoological Section "La Specula" of National History Museum of Florence University)


What ?!?!
This is a mad man…
….and in the future, I imagine a Worldwide Power (Electrical) Grid …...


Hardware
Hardware Virtualization
Multi-core chips
Distributed Computing
Utility &
Grid
Computing
Cloud
Computing
SOA
Web 2.0
Web Services
Mashups
Internet Technologies
Autonomic Computing
Data Center Automation
Systems Management

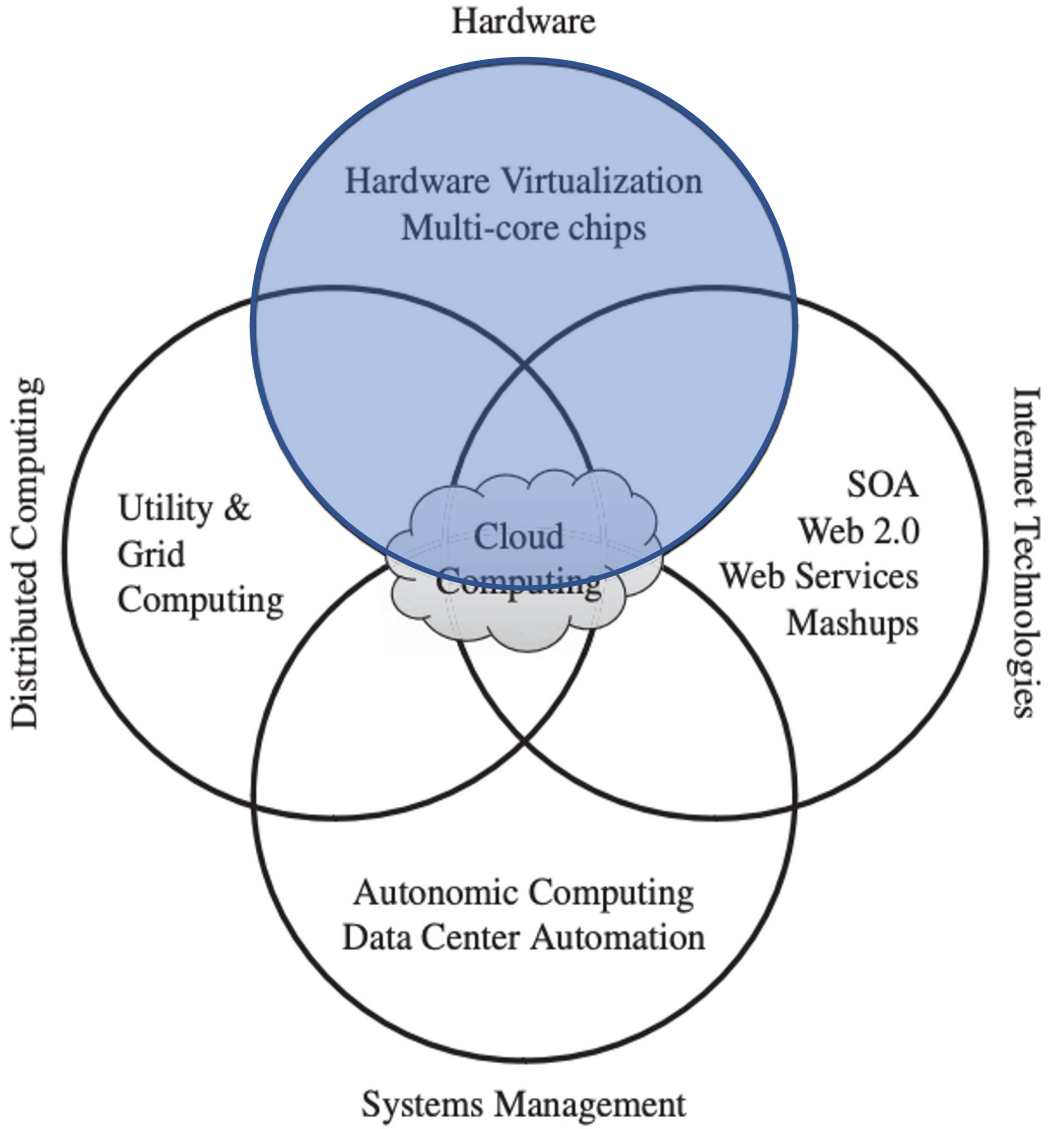
Web Services

A **Web service** is a software system designed to support interoperable machine-to-machine interaction over a network using HTTP and technologies such as XML, SOAP, WSDL, and UDDI.

Web services are increasingly important in distributed systems.

- They support **interoperability** across the global Internet, including the key area of business to-business integration and also the emergent ‘mashup’ culture enabling third-party developers to creative innovative software on top of the existing service base.
- A **mashup** describes a Web application that combines multiple services into a single application.

Many service providers, such as Amazon, Facebook, and Google, make their service APIs publicly accessible using standard protocols such as SOAP and REST.

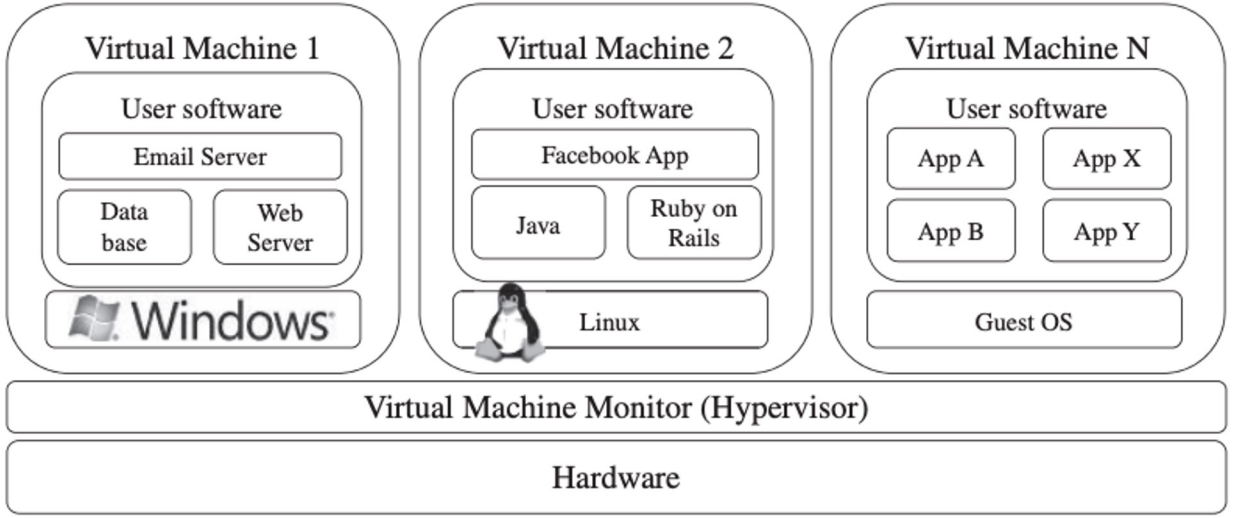

Hardware
Hardware Virtualization
Multi-core chips
Distributed Computing
Utility &
Grid
Computing
Cloud
Computing
SOA
Web 2.0
Web Services
Mashups
Internet Technologies
Autonomic Computing
Data Center Automation
Systems Management

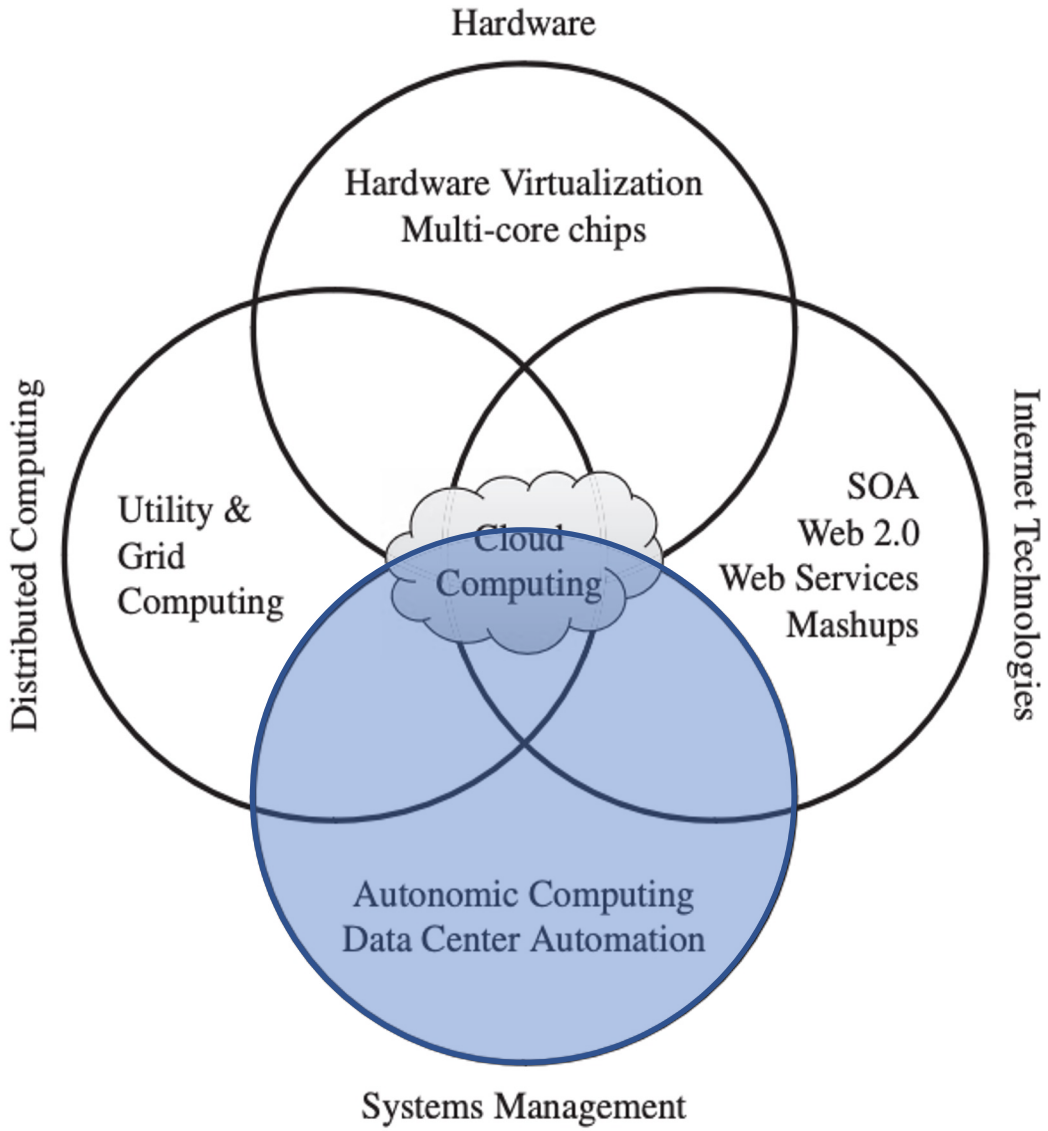
Hardware Virtualization

Virtualization hides the physical characteristics of a computing platform from the users, presenting instead an abstract computing platform

The software that controlled virtualization is called **hypervisor** or **virtual machine monitor,** e.g., Xen, KVM, VMWare ESXi.

Multi-core chips, *paravirtualization*, *hardware-assisted virtualization*, and *live migration* of VMs—has contributed to an increasing adoption of virtualization on server systems.




Hardware
Hardware Virtualization
Multi-core chips
Distributed Computing
Utility &
Grid
Computing
Cloud
Computing
SOA
Web 2.0
Web Services
Mashups
Internet Technologies
Autonomic Computing
Data Center Automation
Systems Management

Autonomic Computing

- The increasing complexity of computing systems
  - motivated research on **autonomic computing**.
- Autonomic computing refers to the **self-managing** characteristics of distributed computing resources, adapting to unpredictable changes while hiding intrinsic complexity to operators and users.
- IBM's Autonomic Computing Initiative has contributed to define the four properties of autonomic systems:
  - self-configuration
  - self-optimization
  - self-healing
  - self-protection.
- IBM has also suggested a reference model for autonomic control loops of autonomic managers, called MAPE-K (Monitor Analyse Plan Execute—Knowledge)

Cloud Deployment Models

Private

- Organization has its own private cloud, might be on-premise or off-premise

Public

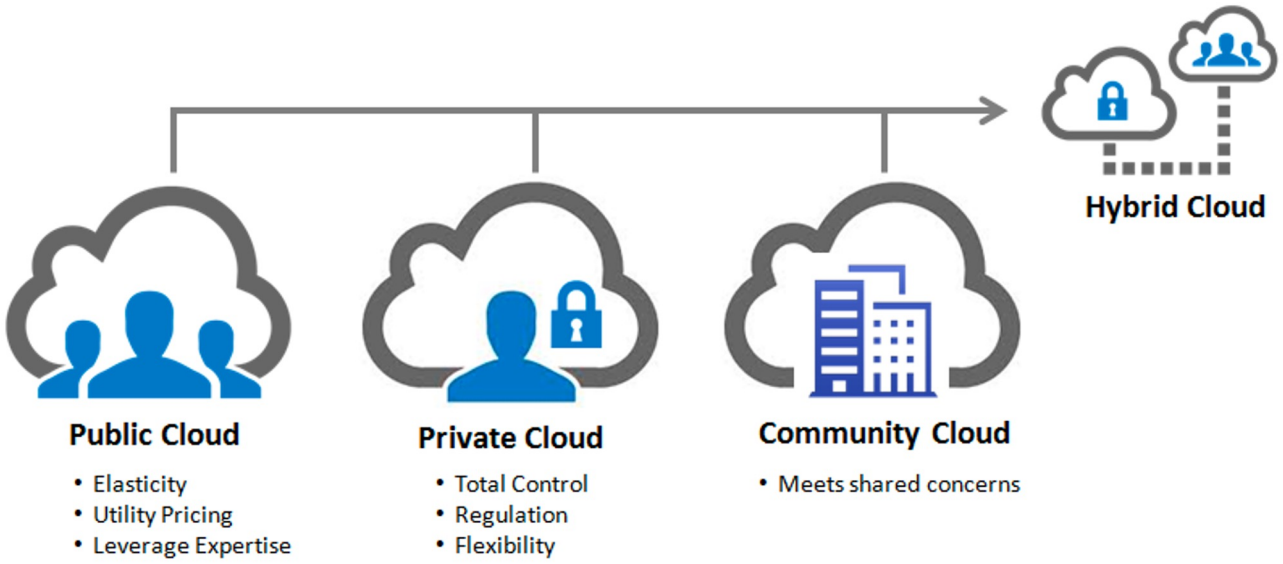
- Organizations shares cloud with general public.

Community

- Shared by several entities that have common purpose.

Hybrid

- Any combination of two or more private, community or public clouds.



Cloud Service Models

Software as a Service (SaaS)

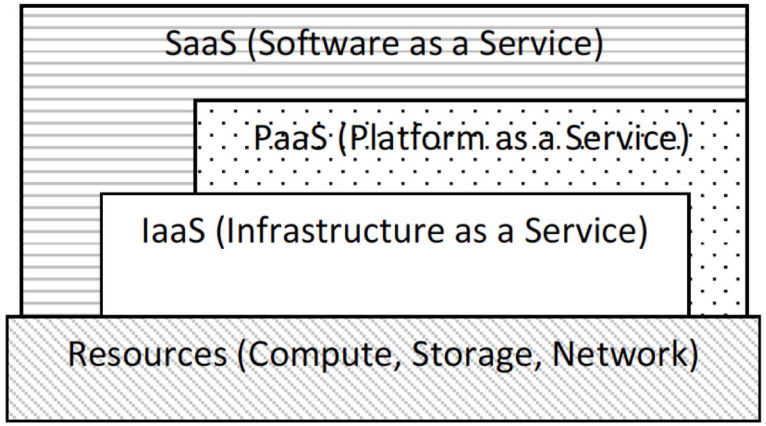
- provides applications and software to the customer in utility-based model which is accessible from a thin client interface such as a Web browser/ Mobile apps
  - Salesforce.com

Platform as a Service (PaaS)

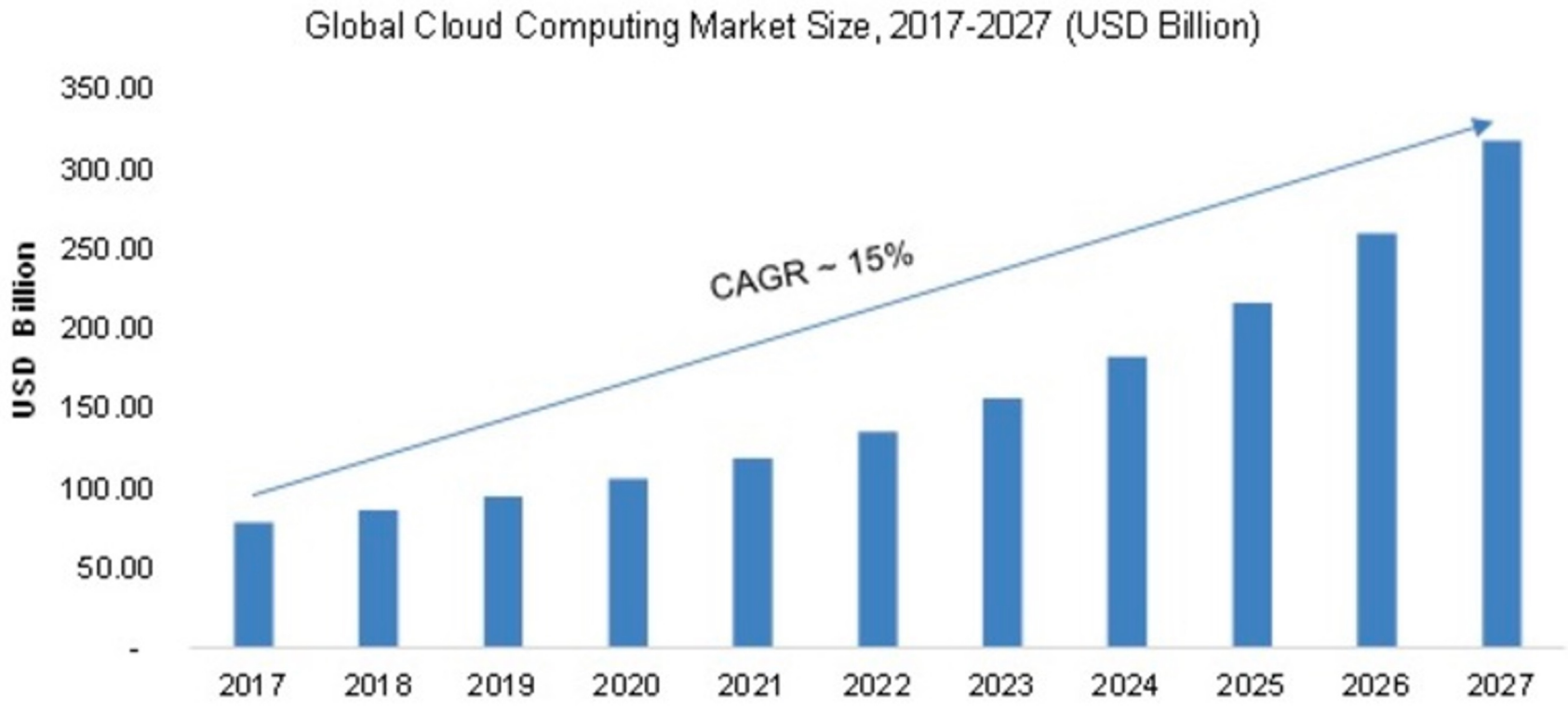
- provides programming languages and tools to deploy application onto the cloud infrastructure
  - Google App Engine

Infrastructure as a Service (IaaS)

- provides capabilities for the customers to provision computational resources such as processing, storage, network, and other fundamental computing resources Virtual Machines(VMs)/Containers
  - Example: Amazon EC2/S3, Azure



Cloud Computing Market Research Report - Global Forecast to 2027



https://www.marketresearchfuture.com/reports/cloud-computing-market-1013

Amazon Web Services (1)

Compute
EC2
Lightsail
ECR
ECS
EKS
Lambda
Batch
Elastic Beanstalk
Serverless Application Repository

Storage
S3
EFS
FSx
S3 Glacier
Storage Gateway
AWS Backup

Database
RDS
DynamoDB
ElastiCache
Neptune
Amazon Redshift
Amazon QLDB
Amazon DocumentDB

Robotics
AWS RoboMaker

Blockchain
Amazon Managed Blockchain

Satellite
Ground Station

Management & Governance
AWS Organizations
CloudWatch
AWS Auto Scaling
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Managed Services
Control Tower
AWS License Manager
AWS Well-Architected Tool
Personal Health Dashboard
AWS Chatbot

Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
QuickSight
Data Pipeline
AWS Glue
AWS Lake Formation
MSK

Security, Identity, & Compliance
IAM
Resource Access Manager
Cognito
Secrets Manager
GuardDuty
Inspector
Amazon Macie
AWS Single Sign-On
Certificate Manager
Key Management Service
CloudHSM
Directory Service
WAF & Shield
Artifact
Security Hub

Business Applications
Alexa for Business
Amazon Chime
WorkMail

End User Computing
WorkSpaces
AppStream 2.0
WorkDocs
WorkLink

Internet Of Things
IoT Core
Amazon FreeRTOS
IoT 1-Click
IoT Analytics
IoT Device Defender
IoT Device Management
IoT Events
IoT Greengrass
IoT SiteWise
IoT Things Graph

Game Development
Amazon GameLift

Amazon Web Services (2)

Migration & Transfer
AWS Migration Hub
Application Discovery Service
Database Migration Service
Server Migration Service
AWS Transfer for SFTP
Snowball
DataSync

Networking & Content Delivery
VPC
CloudFront
Route 53
API Gateway
Direct Connect
AWS App Mesh
AWS Cloud Map
Global Accelerator

Developer Tools
CodeStar
CodeCommit
CodeBuild
CodeDeploy
CodePipeline
Cloud9
X-Ray

Media Services
Elastic Transcoder
Kinesis Video Streams
MediaConnect
MediaConvert
MediaLive
MediaPackage
MediaStore
MediaTailor
Elemental Appliances & Software

Machine Learning
Amazon SageMaker
Amazon Comprehend
AWS DeepLens
Amazon Lex
Machine Learning
Amazon Polly
Rekognition
Amazon Transcribe
Amazon Translate
Amazon Personalize
Amazon Forecast
Amazon Textract
AWS DeepRacer

Mobile
AWS Amplify
Mobile Hub
AWS AppSync
Device Farm

AR & VR
Amazon Sumerian

Application Integration
Step Functions
Amazon EventBridge
Amazon MQ
Simple Notification Service
Simple Queue Service
SWF

AWS Cost Management
AWS Cost Explorer
AWS Budgets
AWS Marketplace Subscriptions

Customer Engagement
Amazon Connect
Pinpoint
Simple Email Service

Summary

- Cloud computing is all about delivering computing services over the Internet.
- Technologies significantly contributed to make cloud computing viable: Grid computing, web services, virtualization, autonomic computing
- Cloud Deployment Models: private, community, public, hybrid
- Cloud Service Models: IaaS, PaaS, SaaS
- Cloud and distributed computing skills are at the top of the most demanded job skills right now.

TECHNISCHE UNIVERSITÄT WIEN

Serverless and FaaS

Shashi Ilager
shashikant.ilager@tuwien.ac.at

TU Wien
Institute of Information Systems Engineering
Data Science Research Unit

Favoritenstraße 9-11/194-1,
1040 Vienna, Austria

What is FaaS - Function as a Service?

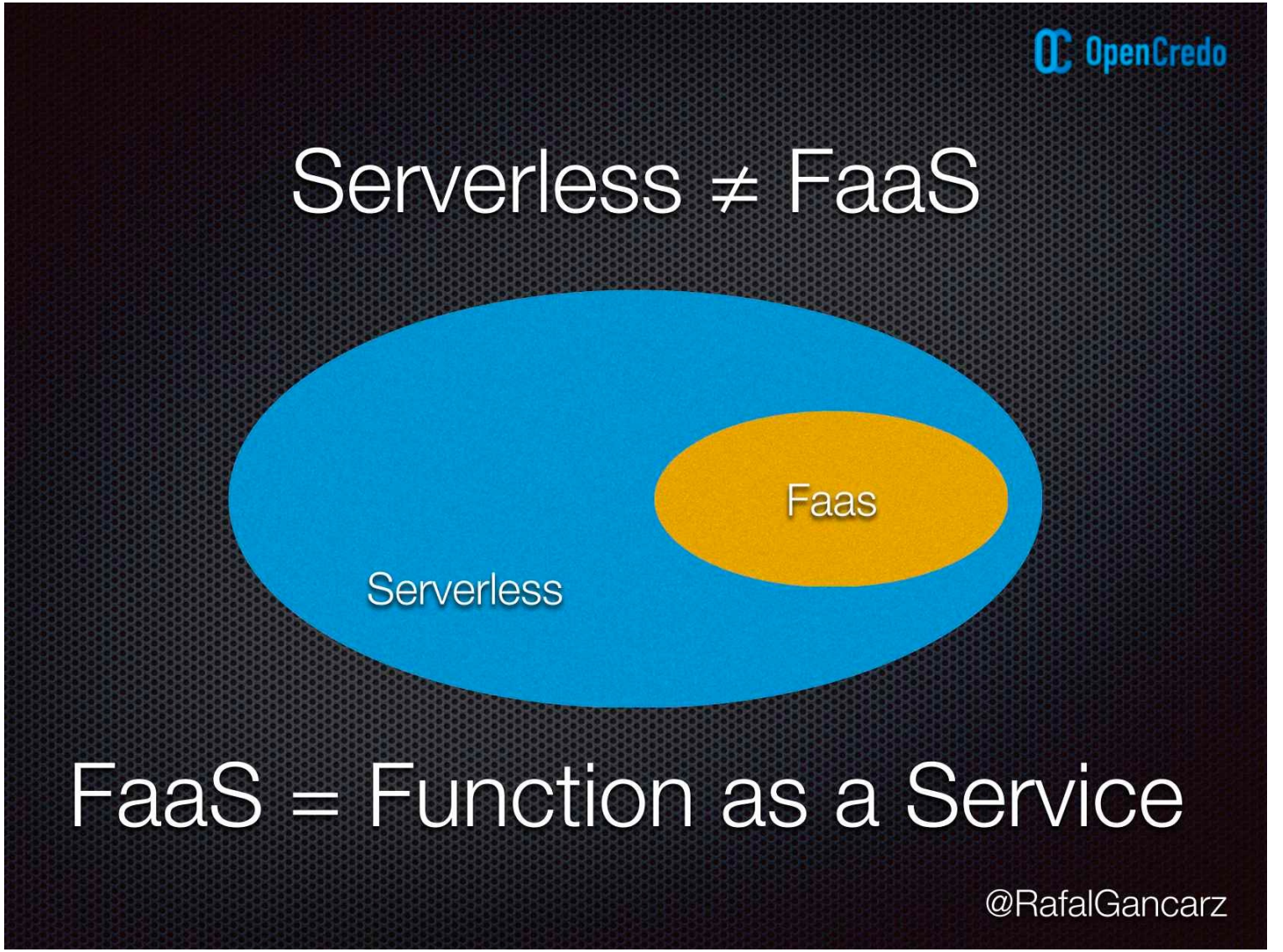
“Focus on your **application**, not the **infrastructure**”

Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as **events** or **HTTP requests**.

Fully managed compute Provisioning, patching, scaling, monitoring, and logging are provided out-of-the-box

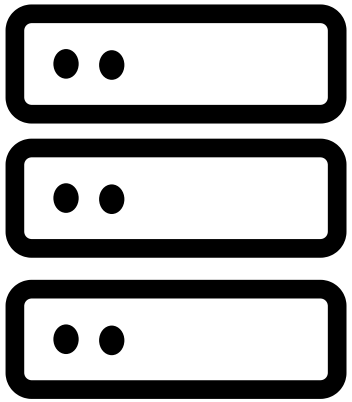
Deploy your code Just package and upload the code

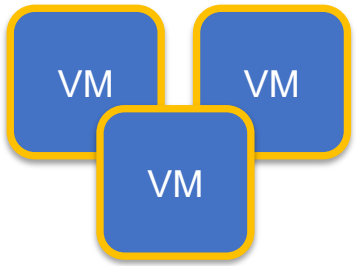
Pay for actual usage Getting charged only upon code execution

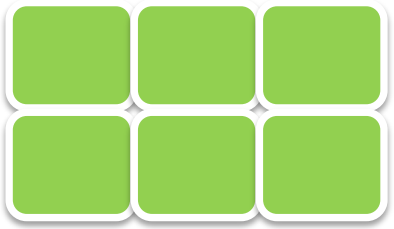

OpenCredo
Serverless ≠ FaaS
Faas
Serverless
FaaS = Function as a Service
@RafalGancarz

Evolution of Serverless

Increasing focus on business logic



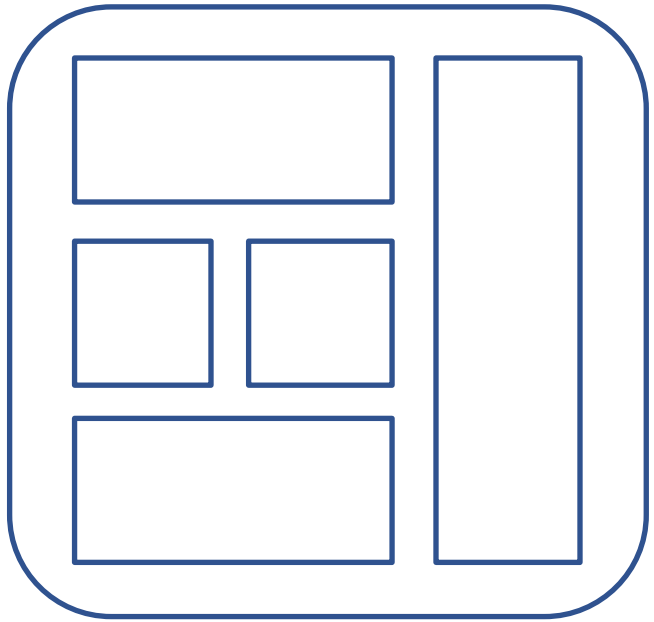




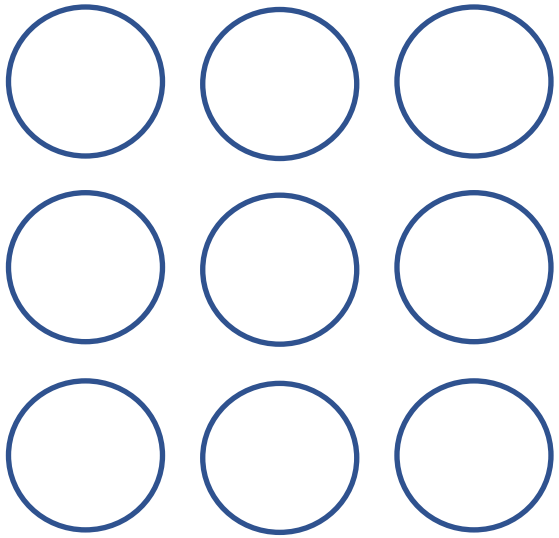
Bare Metal Servers **Virtual Machines** **Containers** **Functions**

Decreasing concern and control over stack implementation

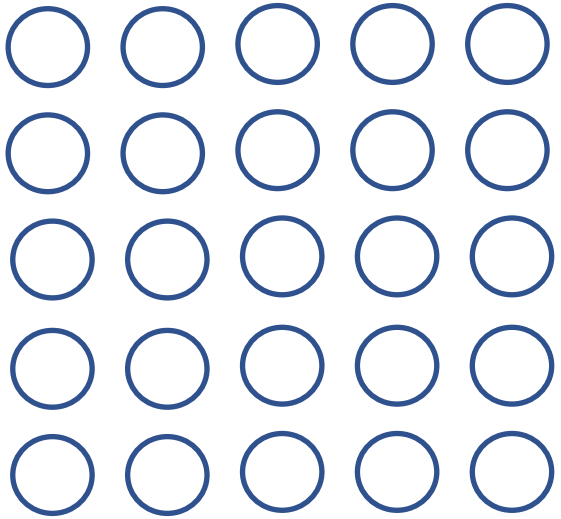
Architectures



Monolith

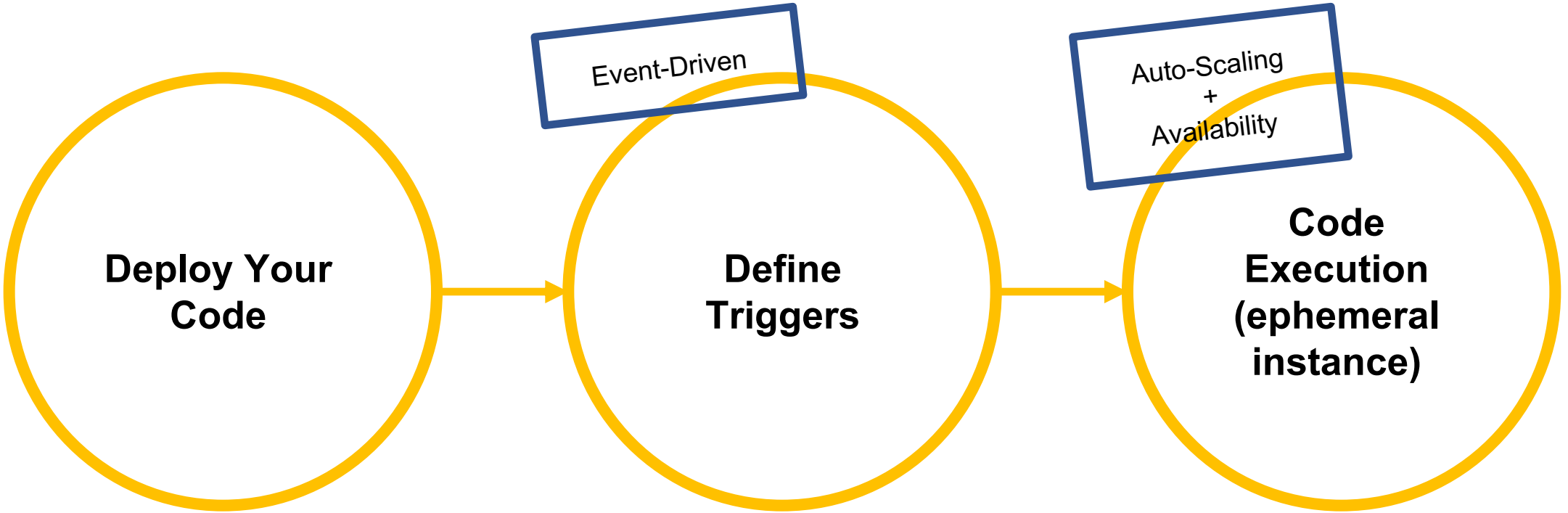


Microservices



FaaS

How it works


Event-Driven
Auto-Scaling
+
Availability
Deploy Your Code
Define Triggers
Code Execution (ephemeral instance)

A function written in JavaScript.

```
function main(params, context) { return
{payload: 'Hello, ' + params.name + ' from '
        + params.place}; }
```

Example of Events

- HTTP requests
- Storage (e.g. file upload)
- DB (e.g. row insert)
- Events stream
- Queue/Messaging
- Scheduled job
- Monitoring
- Infrastructure events
- Code build events

Many more

Limitations

- Stateless: Instances are **ephemeral** (memory, storage) but not necessarily disposable
- No **sticky sessions**
- Provider **limits**: Execution Time (5mins), Memory (1.5GB), Disk (500MB), Package (zip): 50MB*
- Latency: **Cold start** on first or infrequent calls, inconsistent latency
- **Vendor lock-in**: The real lock-in is the integrated services, A trade-off: control vs. convenience
- Costs

	AWS Lambda	EC2
Duration	Per 100ms	Per 1 hour
Cost of Ownership	Low	High
Additional Costs	Requests	Redundancy, Storage
Cost effective	For low and changing load	For high utilization

* Limits change across provider and time

Use Cases

Multimedia processing: The implementation of functions that execute a transformational process in response to a file upload

Database changes or change data capture: auditing or ensuring changes meet quality standards

IoT sensor input messages: The ability to respond to messages and scale in response

Stream processing at scale: processing data within a potentially infinite stream of messages

Chat bots: scaling automatically for peak demands

Batch jobs scheduled tasks: Jobs that require intense parallel computation, IO or network access

HTTP REST APIs and web apps: traditional request and response workloads

Mobile back ends: ability to build on the REST API backend workload above the BaaS APIs

Business logic: The orchestration of microservice workloads that execute a series of steps

Continuous integration pipeline: The ability to remove the need for pre-provisioned hosts

Where is serverless used?	What do they use serverless computing for?
Aegex	Xamarin application that customers can use to monitor real-time sensor data from IoT devices.
Abilisense	Manages an IoT messaging platform for people with hearing difficulties. They estimated they could handle all the monthly load for less than $15 a month.
A Cloud Guru	Uses functions to perform protected actions such as payment processing and triggering group emails. In 2017 they had around 200K users and estimated $0.14 to deliver video course to a user.
Coca-Cola	Serverless Framework is a core component of the Coca-Cola Company's initiative to reduce IT operational costs and deploy services faster. One particular use case is the use of serverless in their vending machine and loyalty program, which managed to have 65% cost savings at 30 million hits per month.
Expedia	Expedia did "over 2.3 billion Lambda calls per month" back in December 2016. That number jumped 4.5 times year-over-year in 2017 (to 6.2 billion requests) and continues to rise in 2018. Example applications include integration of events for their CI/CD platforms, infrastructure governance and autoscaling.
Glucon	Serverless mobile backend to reduce client app code size and avoid disruptions.
Heavywater Inc	Runs Website and training courses using serverless (majority of cost per user is not serverless but storage of video). Serverless reduced their costs by 70%.

FaaS on Public Cloud and FaaS on Premises

AWS Lambda

Google Functions

Azure Functions

Alternatively you can deploy your own FaaS frameworks:

OpenWhisk

OpenFaaS

OracleFN

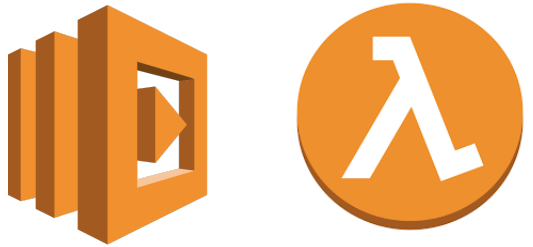
Kubeless

Fission

Iron Functions

Nuclio

Case Study: AWS Lambda



AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you.

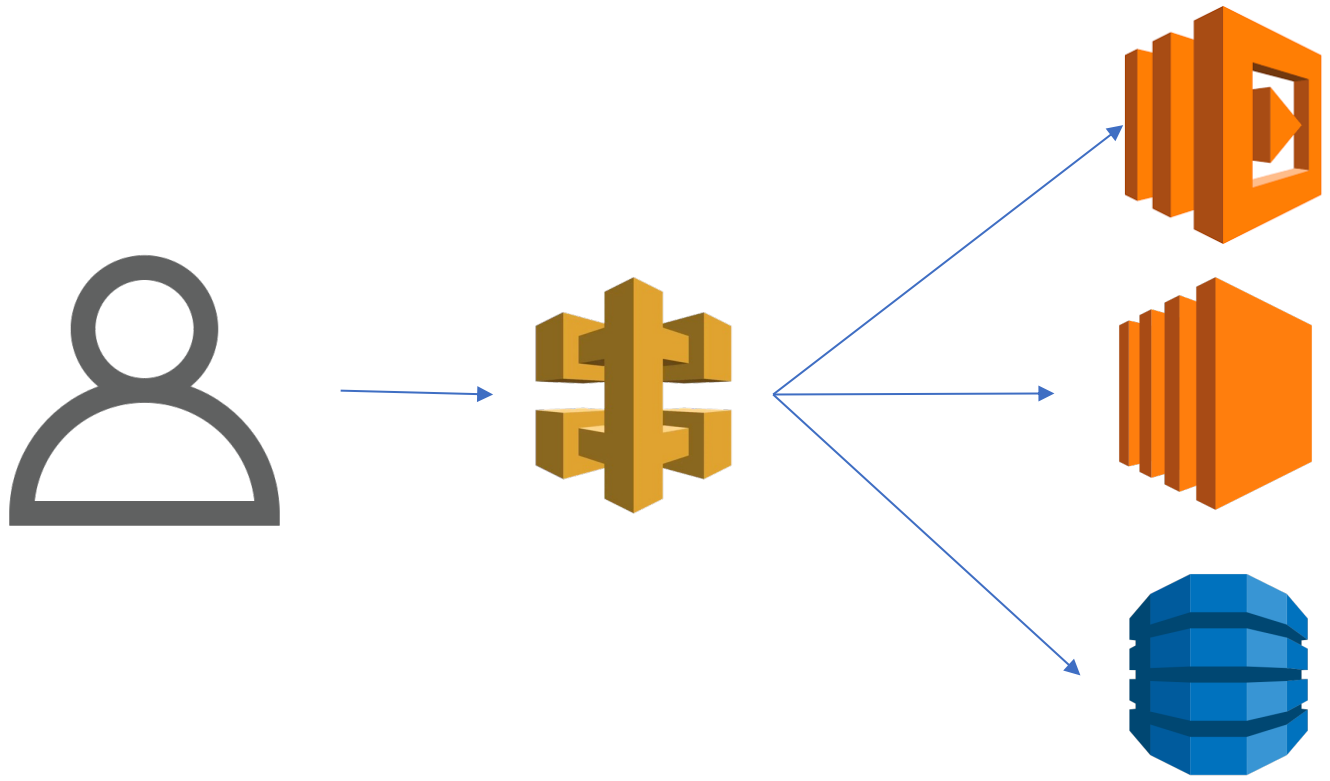
- run code without provisioning or managing servers.
- pay only for the compute time you consume.
- zero administration. Upload your code and Lambda takes care of everything required to run and scale your code with high availability.
- No worries about operating systems, patching, scaling, availability, etc.

How can you use Lambda?

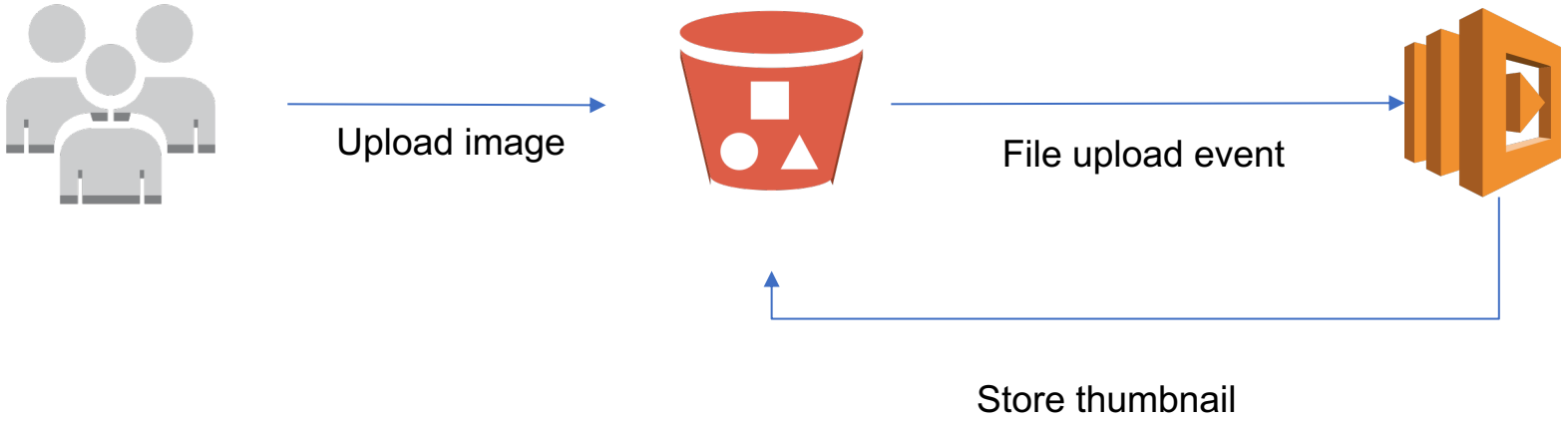
- As an event-driven compute service
  - In response to events
    - Changes in data of an Amazon S3 Bucket
    - Changes Amazon DynamoDB table
- As a compute service
  - in response to HTTP requests
    - Amazon API Gateway
    - AWS SDK

API Gateway

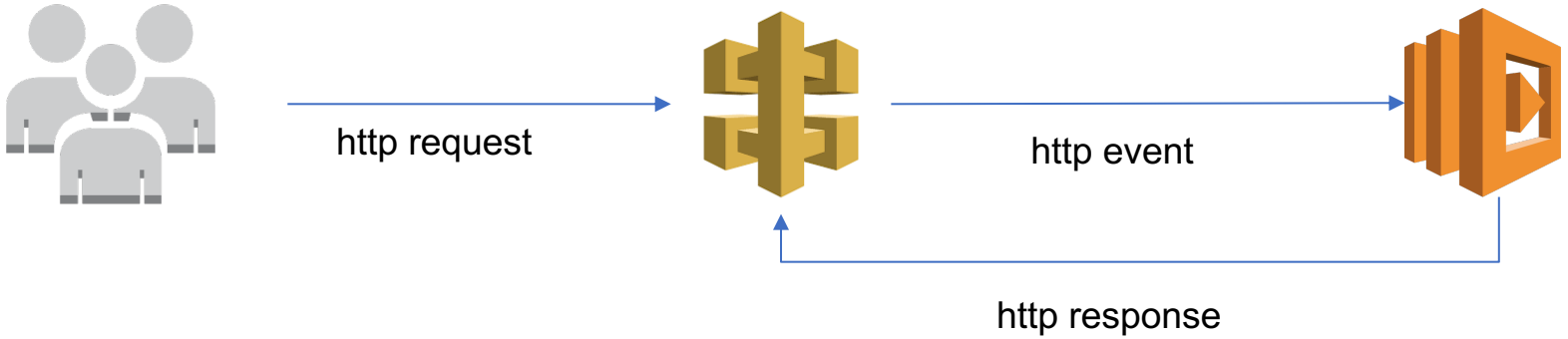
- Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure **APIs at any scale**.
- APIs act as the "**front door**" for applications to
- API Gateway creates RESTful APIs that:
  - Are HTTP-based.
  - Enable stateless client-server communication.
  - Implement standard HTTP methods such as GET, POST, PUT, PATCH, and DELETE.



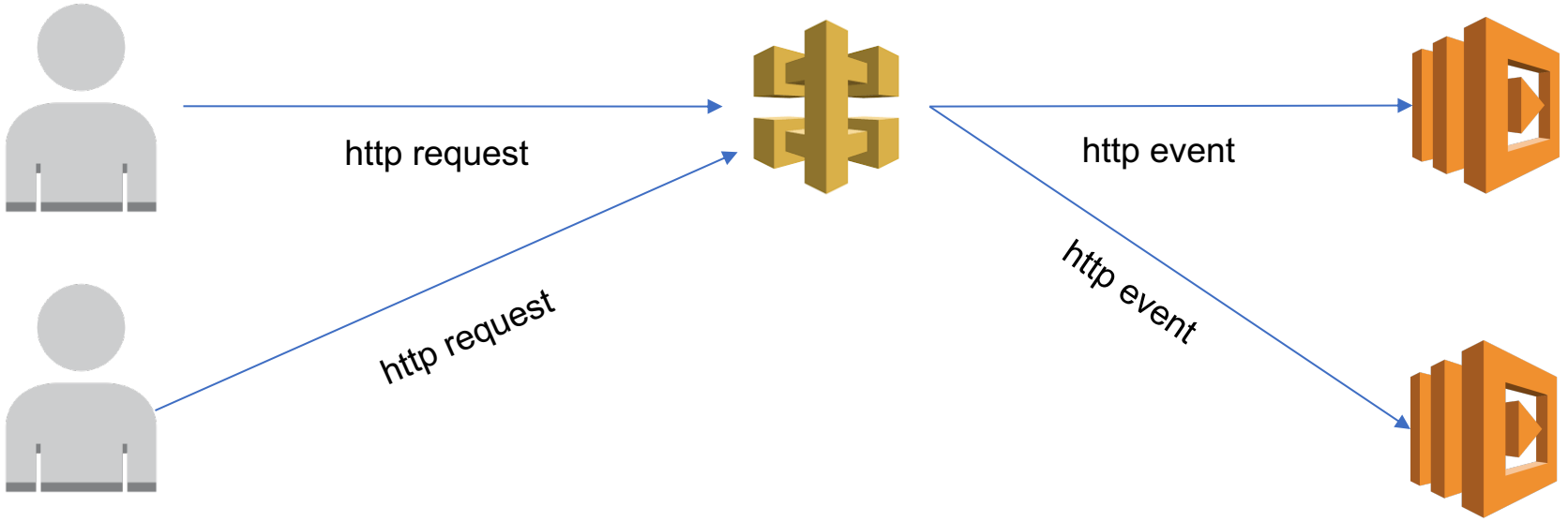
As an event-driven compute service



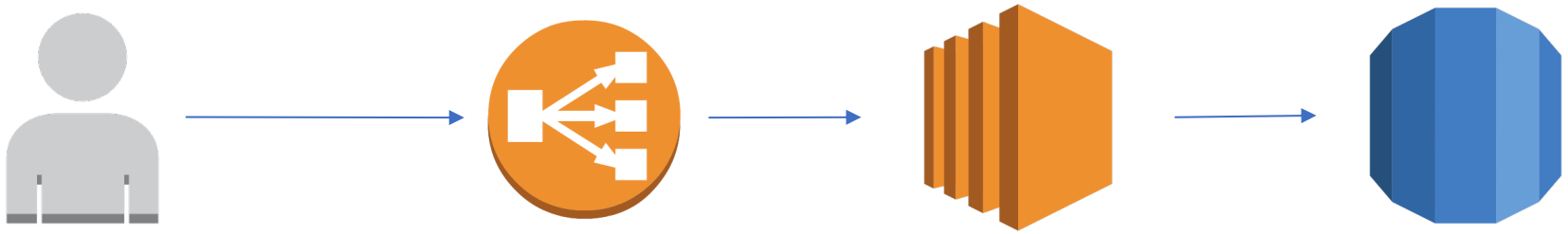
As a compute service in response to HTTP requests



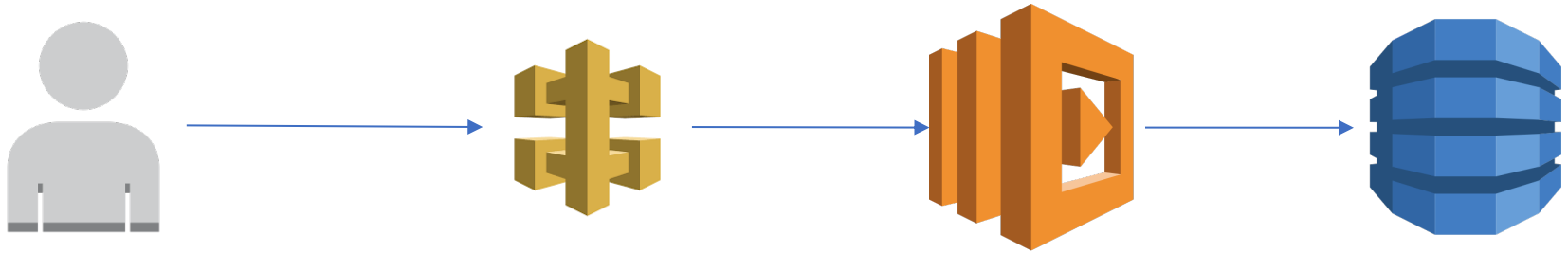
How lambda scales?



Traditional vs Serverless Architecture



Traditional



Serverless

Languages AWS lambda supports

- Node.js
- Java
- Python
- C#
- Go
- PowerShell

How is it priced?

- Number of requests
  - Per million requests payment
  - First 1,000,000 requests are free

- Duration
  - How long your function runs (100ms, 1mins, etc.)

DEMO

Your first Lambda function with S3 trigger.

References

- Benny Bauer, "**The Rise of Serverless -When to FaaS?**" https://www.slideshare.net/bennybauer1/serverless-when-to-faas"
- Christina Cardoza "**10 use cases for serverless**" https://www.itopstimes.com/cloud/10-use-cases-for-serverless/
- Paul Castro, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. 2019. "**The rise of serverless computing.**" Commun. ACM 62, 12 (November 2019), 44-54. https://doi.org/10.1145/3368454
- *Distributed Systems: Concepts and Design*, G. Coulouris, J. Dollimore, T. Kindberg, G. Blair, 5th ed. 2012 , ©Pearson Education Chapter 1, Characterization of Distributed Systems
- *Cloud Computing: Principles and Paradigms* ,Rajkumar Buyya, James Bromberg, Andrzej M. Goscinski
©John Wiley & Sons, 2010, Chapter 1 Introduction to Cloud Computing