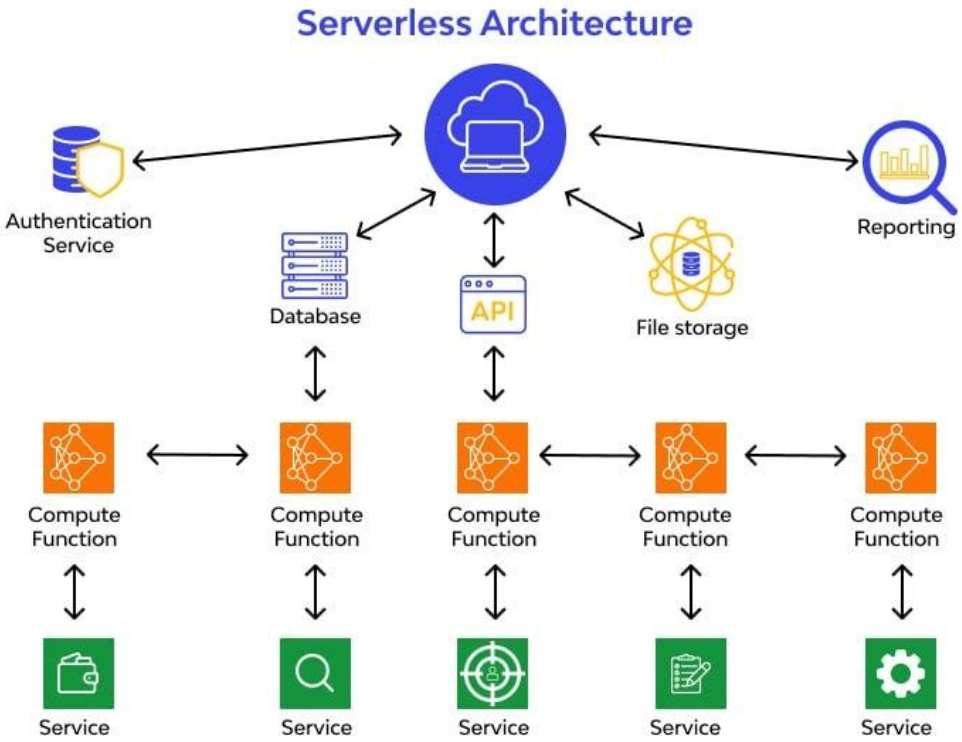
# COMP2511

Tutorial 10

# Serverless Architecture

- Importantly, ‘serverless’ **doesn’t** mean no servers.
- A serverless architecture is a method of building and running applications and services where infrastructure is handled by an external provider.
  - Ultimately there’s still a server, but the important thing is that **you** aren’t managing the servers and infrastructure yourself; it’s managed by an external cloud provider.
  - Apps are typically built as a collection of *stateless* functions (no shared memory or state), which are executed in response to events (e.g. a HTTP request, or file upload). This is the idea of **Functions as a Service (FaaS)**.
  - Common serverless platforms include **AWS Lambda** (Amazon), **Azure Functions** (Microsoft) and **Cloud Run functions** (Google).

# Serverless Architecture



# Serverless Architecture

- Relation to key architectural characteristics:
  - **Cost**: Serverless architectures operate on a ‘pay as you go’ model. Cost-effective if you aren’t expecting a high volume of requests, but not if it should be up 24/7.
  - **Scalability**: Cloud providers adjust resources based on real-time demand, so scalability is inherently built-in to the system.
  - **Deployability**: Since there is less operational overhead, you’re able to focus on implementing the core functionalities of the system, leading to a faster **time-to-market**.
  - **Performance**: Some latency may be introduced by *cold starts*: if the application isn’t used frequently, there’s a ‘boot-up’ time whenever you need to make a new container.
  - **Consistency**: If you’re dealing with a stateless application, you will receive the same output every time you give the same input. This is important if a specific operation fails; you can simply just retry it without any drawbacks.

# Serverless Architecture

- In which situation(s) could a serverless architecture be suitable?
  - **E-Commerce:** Serverless functions automatically scale to handle a large number of concurrent requests. For example, each time a user places an order or views a product, a function can be triggered to update inventory or process payments. During high traffic periods, this elasticity ensures smooth performance without provisioning infrastructure ahead of time, and you only pay for actual usage.
- In which situation(s) would you avoid using a serverless architecture?
  - If you have specific constraints on stuff like the hardware or operating system you're using. Serverless architectures don't really provide any flexibility here.
  - If your application needs to be up 24/7, costs can rack up very quickly.
  - If your application needs to manage state, it needs to be managed externally through something like DynamoDB - database calls introduce latency, and would not be suitable for low-latency applications (e.g. trading).

# Revision

- What is a possible design pattern you could use in each of the following scenarios?
  - Sorting students in different ways, such as in order of their last name, or in order of their zID etc.
    - **Strategy**: strategies for each of the different ways to compare between students
  - Listing the contents of a file system.
    - **Composite**: file systems follow a tree-like structure.
  - Updating a UI component when the state of a program changes.
    - **Observer**: a change happens in response to a specific event.
  - Creating a cross-platform application that needs to generate different types of UI components (buttons, menus, dialogs) that match the look and feel of each operating system (Windows, macOS, Linux).
    - **Abstract Factory**: we have different products (buttons, menus, dialogs) separated into different categories (operating systems), each with different functionalities.

# Revision

- Does the presence of a code smell **guarantee** that there is a design issue?
  - No. Code smells are only *indications* of particular design issues.
  - Switch statements can be considered a code smell.They *could* be indicative of a violation of the open-closed principle if an alternative could be used in its place (e.g. the Strategy pattern), but using a switch statement to deal with different strings obtained from user input isn't really a design issue; it's just a logical approach to dealing with different cases.
- Should you always aim to apply a Design Pattern in your code?
  - No. Always think about whether a specific pattern is actually applicable for the problem and doesn't overcomplicate it.

# Revision

- What is the primary benefit of serverless computing for developers?
  - a) Complete control over the operating system
  - b) No need to write functions
  - c) No infrastructure management required
  - d) Permanent server allocation

# Revision

- What is the primary benefit of serverless computing for developers?
  - a) Complete control over the operating system
  - b) No need to write functions
  - **c) No infrastructure management required**
  - d) Permanent server allocation

# Revision

- In a serverless system, when does a function typically execute?
  - a) At scheduled time intervals only
  - b) Continuously e.g. a background process
  - c) Only when the server is restarted
  - d) In response to specific events e.g. the user pressing a button or file uploads

# Revision

- In a serverless system, when does a function typically execute?
  - a) At scheduled time intervals only
  - b) Continuously e.g. a background process
  - c) Only when the server is restarted
  - d) **In response to specific events e.g. the user pressing a button or file uploads**

# Revision

- You’re building a simple online ordering system for a small cafe because they would like an electronic way of managing their orders. There are only a handful of features (creating an order, marking an order as complete and deleting an order) as the owner would like to test the system out before committing to it. What architecture is most suitable?
  - a) Microservices
  - b) Event-Driven
  - c) Modular Monolith
  - d) Layered Architecture

# Revision

- You’re building a simple online ordering system for a small cafe because they would like an electronic way of managing their orders. There are only a handful of features (creating an order, marking an order as complete and deleting an order) as the owner would like to test the system out before committing to it. What architecture is most suitable?
  - a) Microservices
  - b) Event-Driven
  - c) Modular Monolith
  - **d) Layered Architecture**

# Revision

- An education platform needs clear boundaries between features like course content, assessment, and user management. Teams are aligned to business domains. What is the best fit?
  - a) Modular Monolith
  - b) Serverless
  - c) Layered Architecture
  - d) Microservices

# Revision

- An education platform needs clear boundaries between features like course content, assessment, and user management. Teams are aligned to business domains. What is the best fit?
  - **a) Modular Monolith**
  - b) Serverless
  - c) Layered Architecture
  - d) Microservices

# Revision

- An IoT device continuously sends JSON health data to the cloud. The system must process this data and generate alerts immediately. What is the most efficient choice?
  - a) Microservices
  - b) Event-Driven
  - c) Layered
  - d) Monolithic

# Revision

- An IoT device continuously sends JSON health data to the cloud. The system must process this data and generate alerts immediately. What is the most efficient choice?
  - a) Microservices
  - **b) Event-Driven** (using serverless to *implement* this would be appropriate).
  - c) Layered
  - d) Monolithic

# Revision

- Your organisation wants to scale user management, billing, and content recommendation independently for a streaming platform like Netflix. What architecture is most suitable?
  - a) Microservices
  - b) Event driven
  - c) Layered Architecture
  - d) Modular Monolith

# Revision

- Your organisation wants to scale user management, billing, and content recommendation independently for a streaming platform like Netflix. What architecture is most suitable?
  - **a) Microservices**
  - b) Event driven
  - c) Layered Architecture
  - d) Modular Monolith