Ballerina

Cloud Native Application Development

July 2025

Hello!

Tharindu Weerasinghe

tharinduwe@wso2.com | Software Engineer **|** **@ballerinalang** **|** **WSO2**

Ravin Perera

ravin@wso2.com | Software Engineer **|** **@ballerinalang** **|** **WSO2**

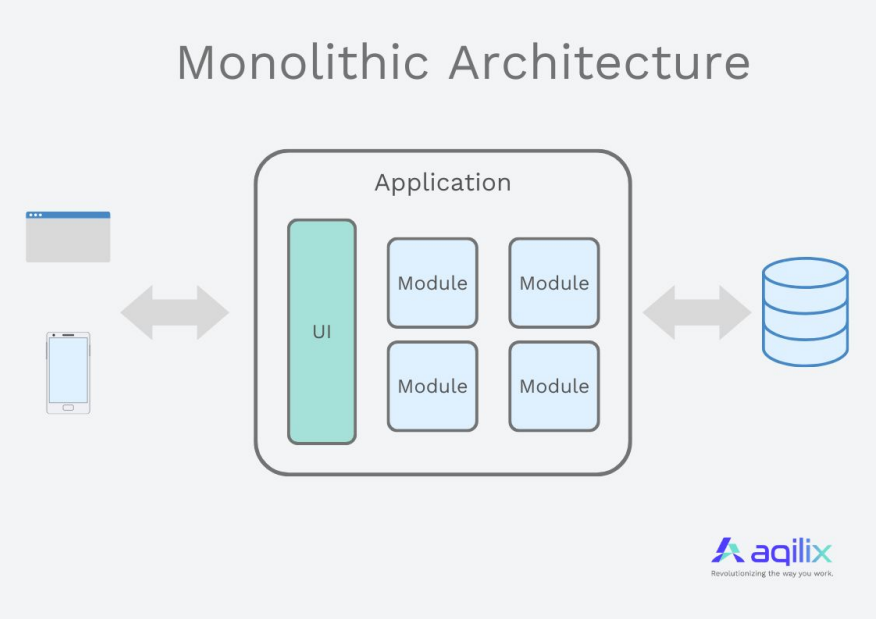
About this Session

Coming Up

- What is Cloud Native Software
- Fundamentals of API
- API driven Development
- Hands-On Session

What is Cloud Native Software

The Monolith: Challenges of Traditional Architecture



https://medium.com/@dollyaswin/modern-application-architectures-part-1-monolithic-architecture-90140987d0cc

Single, tightly coupled codebase.

Challenges:

- Changes in one module impact the entire system, leading to long release cycles.
- Requires scaling the entire application, even if only one component needs more resources.
- A failure in one part can bring down the entire system.
- Difficult to adopt new technologies without significant refactoring.

Architectural Style: Microservices

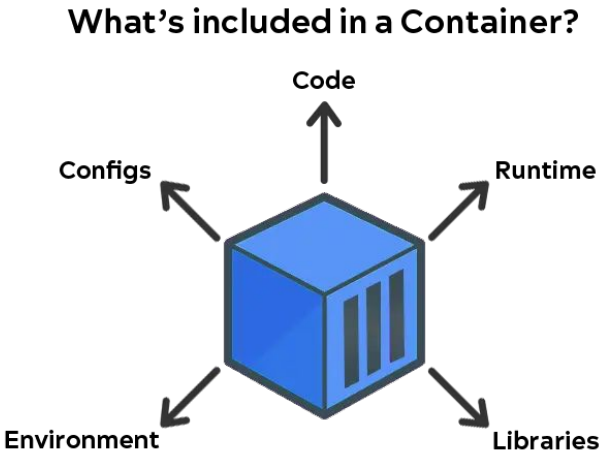
https://www.simform.com/blog/how-does-microservices-architecture-work/

An architectural approach where a large application is built as a suite of small, independent services.

Characteristics:

- Services can be deployed without affecting others.
- Services have minimal dependencies on each other.
- Each service focuses on a specific business function.
- Enables faster decision-making and development.

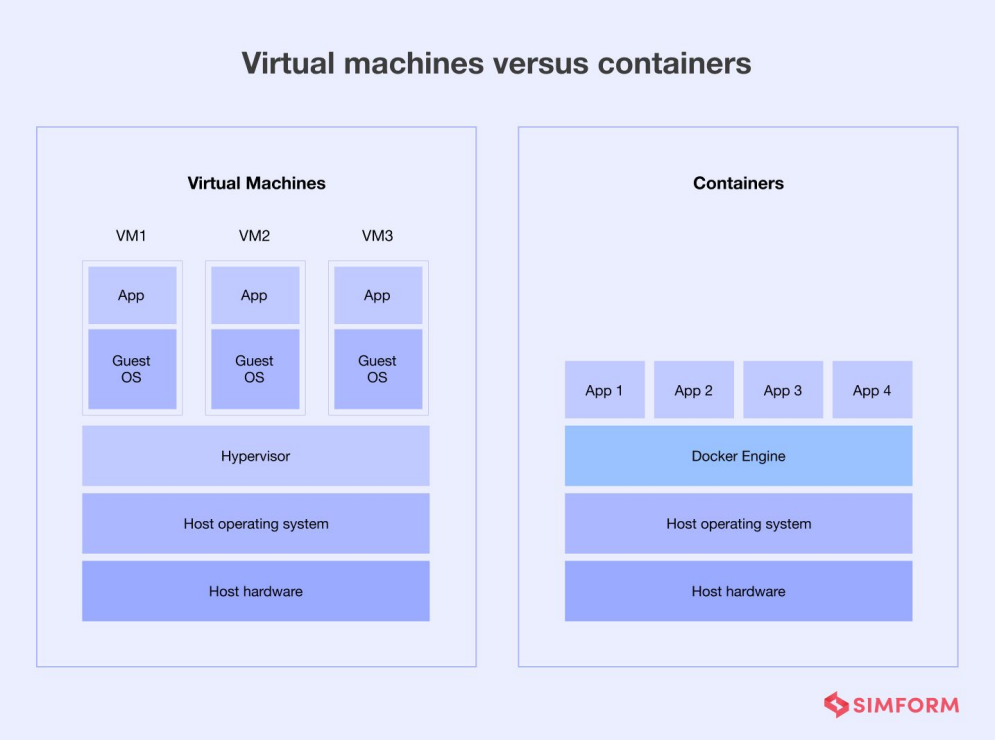
Infrastructure Foundation: Containers



https://blog.back4app.com/what-are-containers-in-cloud-computing/

- Package application code, runtime, system tools, libraries, and settings.
- Ensure consistent environments across development, testing, and production.
- Lightweight and isolated, sharing the host OS kernel.

Virtual Machines vs Containers



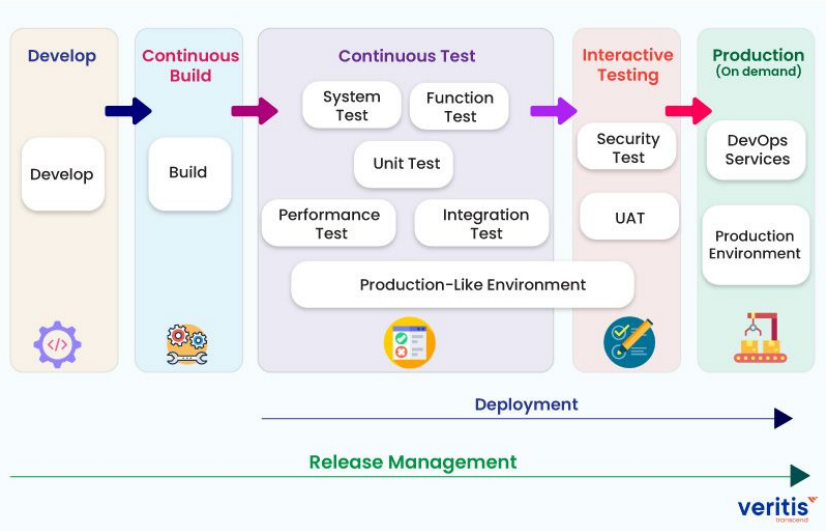
https://www.simform.com/blog/vms-vs-containers/

Infrastructure Foundation: Cloud Platforms

https://www.cloudjournee.com/blog/aws-kubernetes-resilient-infrastructure-guide/

- On-demand access to computing, storage, and services over the internet.
- Abstracts underlying physical infrastructure management.

Development Methodology: DevOps & CI/CD



https://www.veritis.com/blog/ci-cd-services-integrate-and-automate-devops/

Integrates software development and operations.

Key Practices:

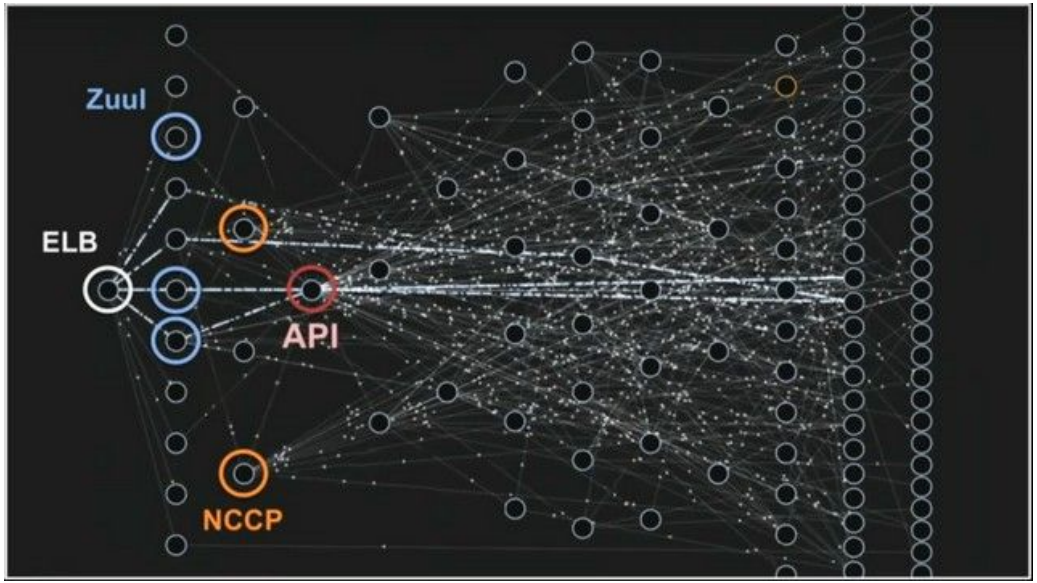
- Continuous Integration (CI): Automating code integration and testing.
- Continuous Delivery (CD): Automating releases to production.
- Infrastructure as Code (IaC): Managing infrastructure with configuration files.
- Monitoring & Feedback: Real-time visibility into application performance and health.

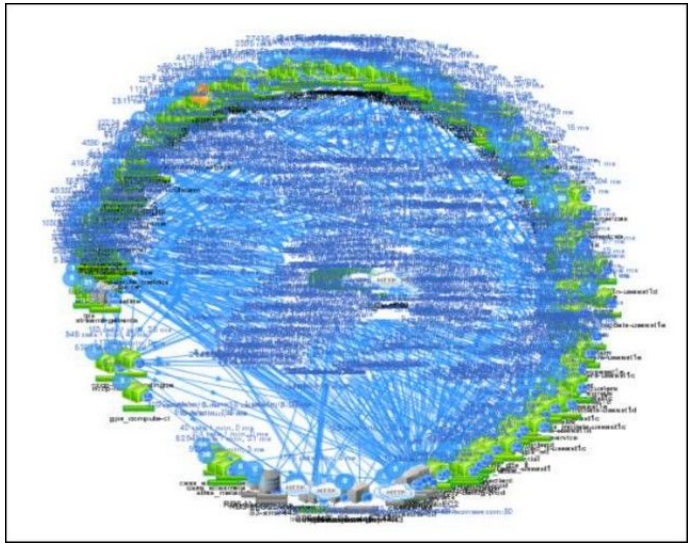
Evolution of Software Development

	Development Process	Application Architecture	Deployment & Packaging	Application Infrastructure
~ 1980 ~ 1990	Waterfall	Monolithic	Physical Server	Datacenter
~ 2000	Agile	N-Tie	Virtual Servers	Hosted
~ 2010	DevOps	Microservices	Containers	Cloud

https://www.oracle.com/cloud/cloud-native/what-is-cloud-native/

Service communication



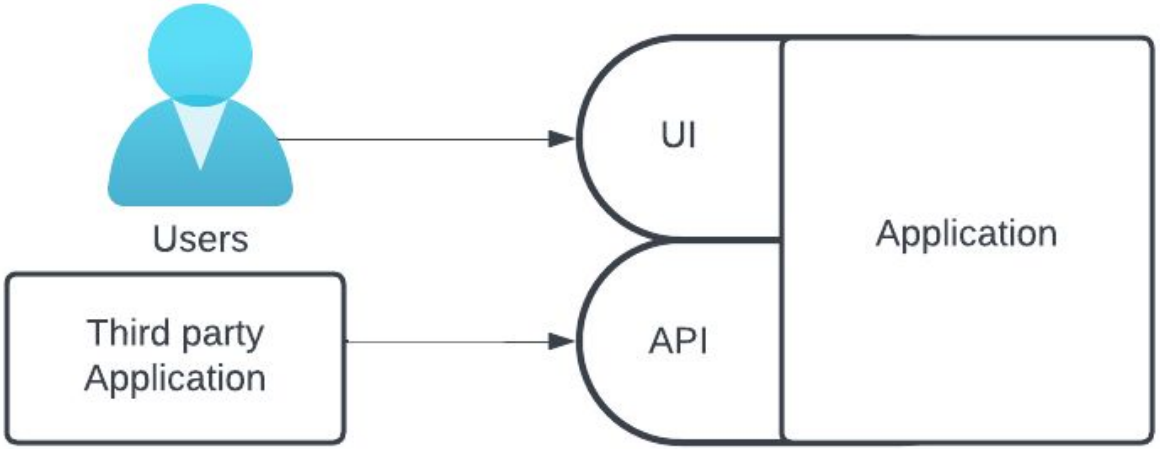


https://www.zdnet.com/article/to-be-a-microservice-how-smaller-parts-of-bigger-applications-could-remake-it/

Fundamentals of API

What is an API?

- **A**pplication **P**rogramming **I**nterface
- An interface for two software components to talk to each other.

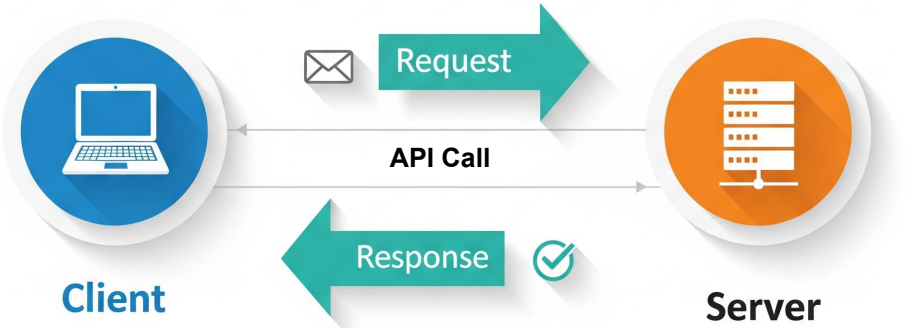


What is an API?

How does an API work?

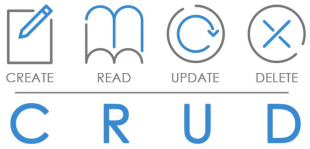
- **Client and Server**

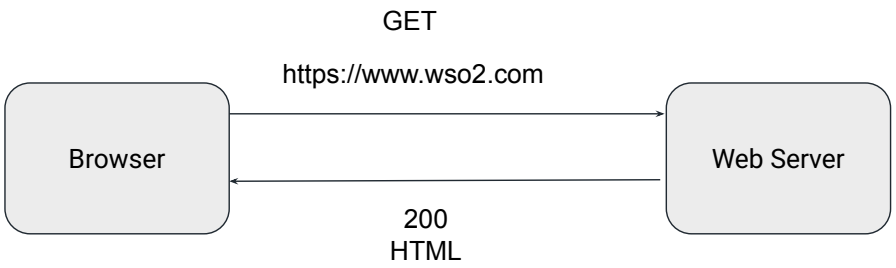
  Client: The application or user making the request.
  Server: The system/service that provides the data or functionality.



How does an API work?

- **Requests and Responses**
  - Common request methods: GET, POST, PUT, DELETE
  - Response: Includes the requested data or confirmation that an action was performed.



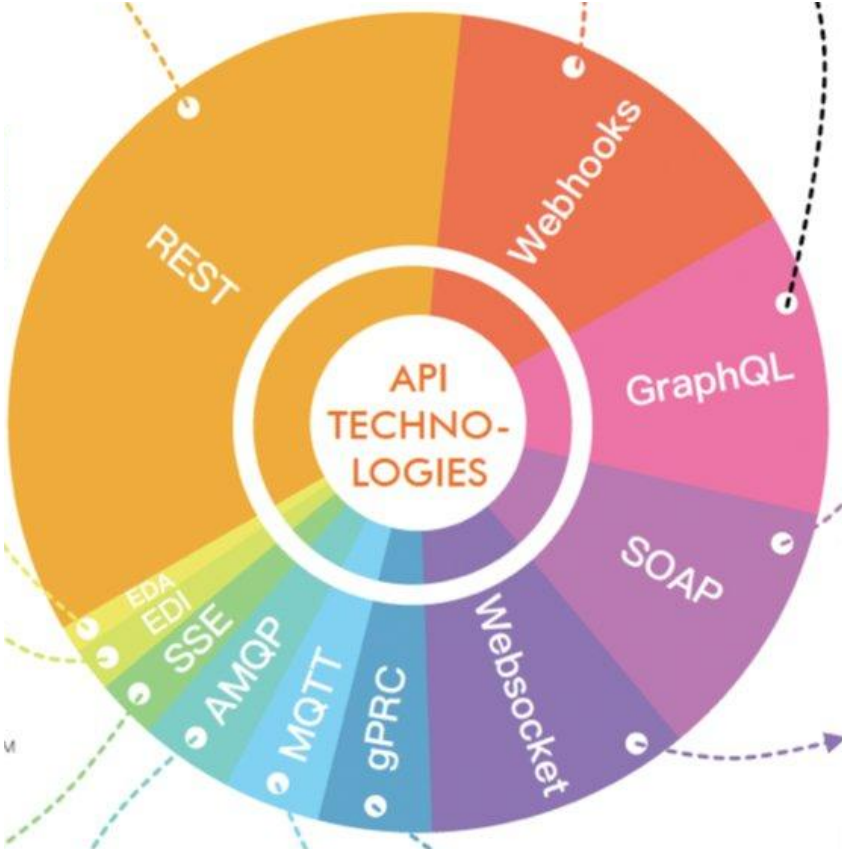


How does an API work?

- **Endpoints :** URLs where the client can make requests
- **Data Format :** Data is exchanged mostly in JSON or XML
- **Authentication**: Client specific ‘API Key’
- **Rate Limiting**: to avoid overloading the server
- **Error Handling :** Common codes like 4xx and 5xx are used

200 404 500 400 201 403 503 429

API Protocols



Ballerina
Swan Lake

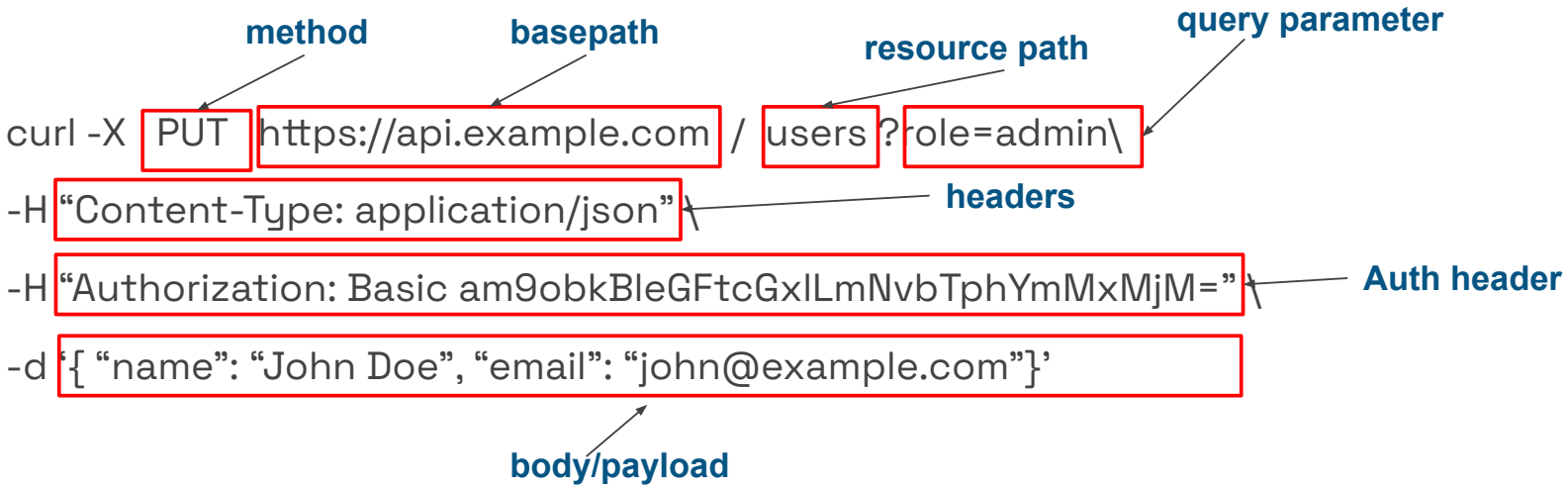
REST (**RE**presentational **S**tate **T**ransfer)

- Most widely used architectural style leverages HTTP protocol
- Uses the concept of resources
- Resources can be accessed via verbs and resource paths.

  Eg: GET /api/users

- Each resource has a standard format to represent data; server sends - client understands

Example REST API Call



GraphQL

- Relatively new protocol developed by Facebook
- Fast adaptation from the major companies
- Query language for APIs
- Data is structured as a hierarchical structure
- Has a single endpoint
- Clients can request exactly what they want, server responds with exactly what was requested

REST vs GraphQL

REST:

GET https://api.example.com/users/123

Response:

```
{
  "id": "123",
  "name": "Bob Smith",
  "email": "bob@gmail.com",
  "age": 30,
  "isActive": true,
  "profilePicture": "https://example.com/profile.jpg",
  "address": {
    "street": "1600 Pennsylvania Ave.",
    "city": "Washington, DC",
    "zip": "20500"
  }
}
```

GraphQL:

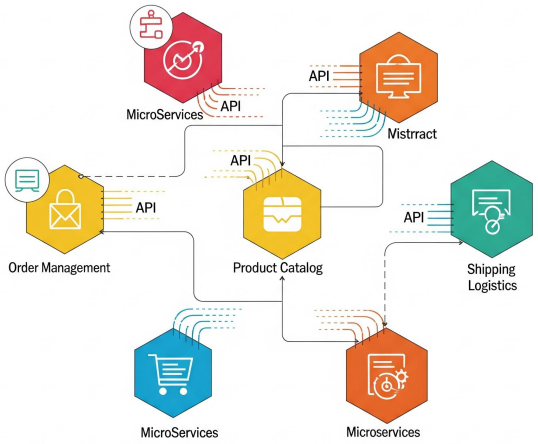
```
query {
 user(id: "123") {
  name
  profilePicture
 }
}
```

Response:

```
{
 "data": {
  "user": {
   "name": "Bob Smith",
   "profilePicture": "https://example.com/profile.jpg"
  }
 }
}
```

Ballerina
Swan Lake

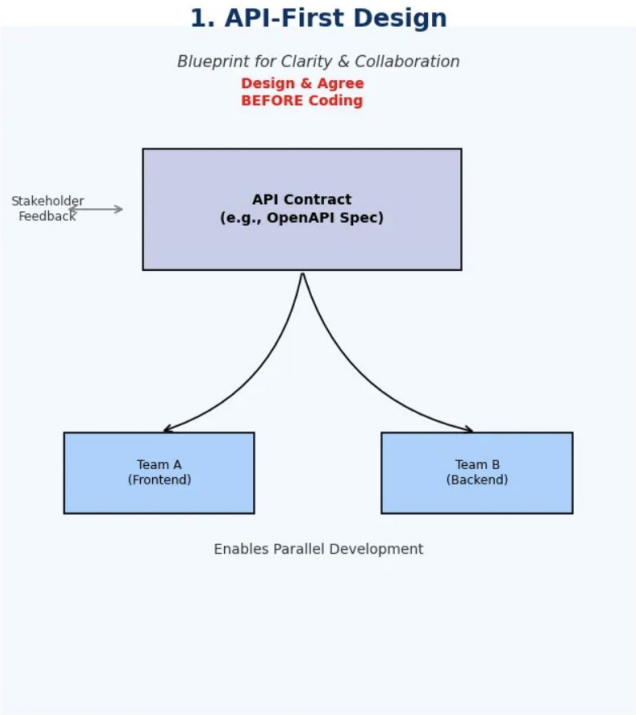
APIs and Microservices



- APIs enable communication between microservices
- Microservices use APIs to expose functionalities
- APIs define how services interact with each other
- They ensure loose coupling of services

API-Driven Development

API-Driven Development



https://machine-learning-made-simple.medium.com/api-driven-development-the-necessary-prerequisite-to-mcps-a100cc203c52

An approach where Application Programming Interfaces (APIs) are treated as first-class products.

They are designed and defined before or concurrently with the implementation of the services themselves.

Why it's crucial:

- In a microservice architecture , APIs are the primary means of communication.
- It shifts focus from internal implementation details to external contracts and interfaces, ensuring interoperability.

Core Principles of API-Driven Design

Contract-First Design:

- Define the API (endpoints, data formats, security) using formal specs (e.g., OpenAPI) before coding.
- Enables clear team alignment, prevents integration issues.

Consumer-Centric Design:

- Design APIs based on the needs of their consumers (e.g., frontend, partners).
- Involve consumers early for feedback to ensure usability.

Documentation & Discoverability:

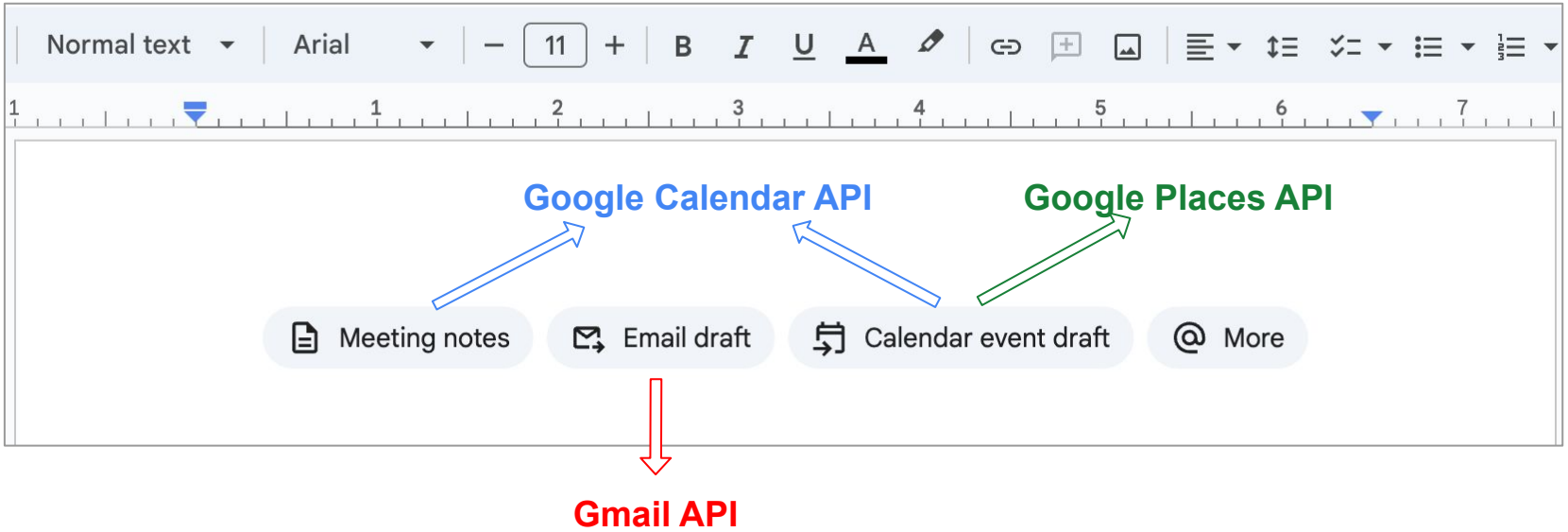
- APIs should be well-documented and easy to find.
- Good documentation lowers the integration barrier and can be auto-generated from specs.

Real-world Examples

Sharing a post on Facebook

Ballerina
Swan Lake

Smartchips on GoogleDocs



Hands-on

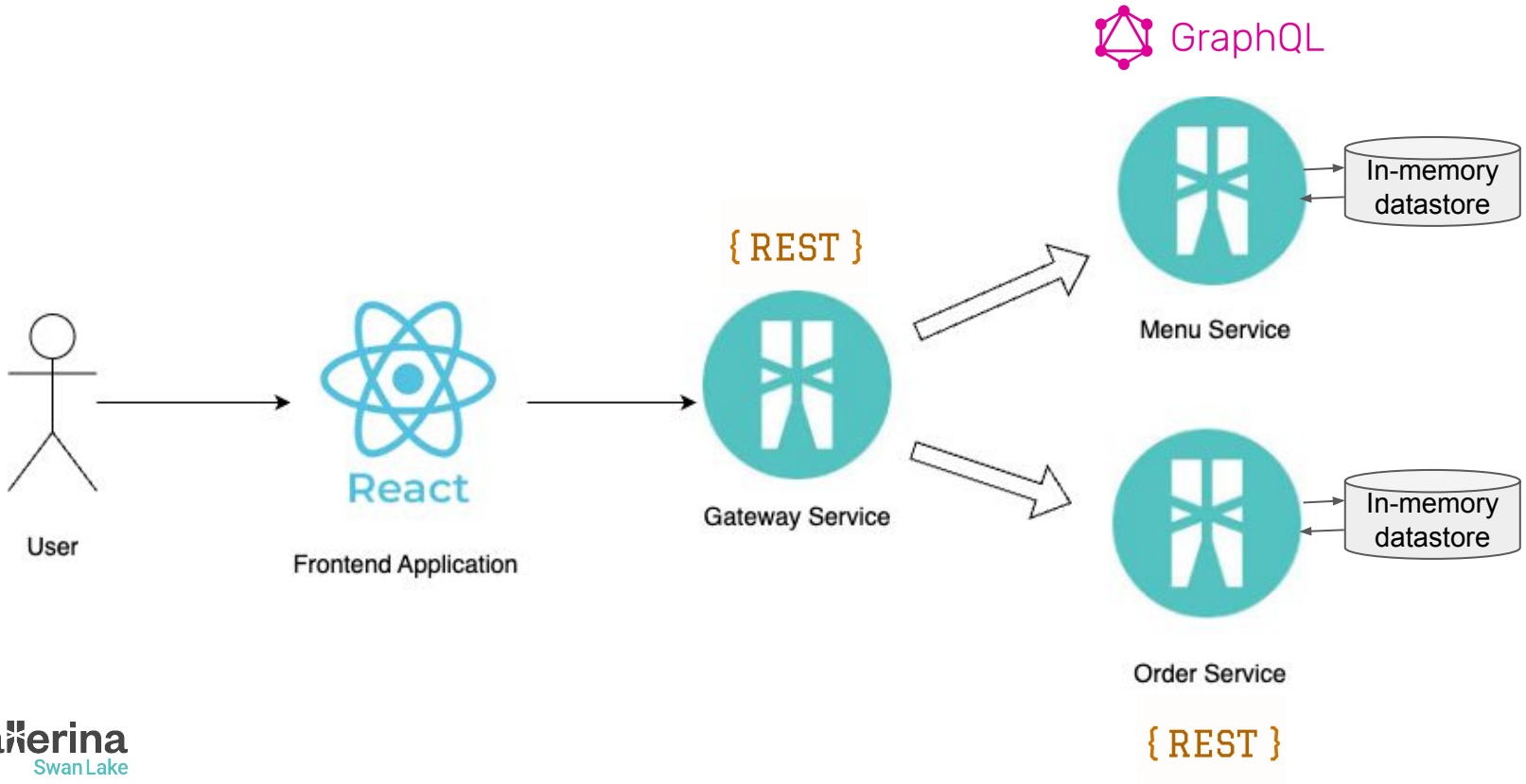
Prerequisites

- Ballerina
- VSCode
- VSCode extension for Ballerina

We will be building an order application…

- Based on Microservice Architecture
  - Menu Service 
  - Order Service 
- APIs to
  - Create new orders
  - View all orders
  - View details of a specific order
  - View all food items in the menu

Application Architecture



Ballerina Swan Lake

- Fully open-source programming language, powered by WSO2
- 6+ years of effort with 300+ contributors
- Cloud-native programming language optimized for integration
- Both textual syntax and graphical form
- Network Oriented Programming (DOP) paradigm

Ballerina is a full platform

Azure SDK for Java AWS SDK for Java

Gradle

HIBERNATE

MVN REPOSITORY

Maven

Netty project

Java

spring

gson

Spring Boot

GraphQL Java

TestNG

JUnit 5

Ballerina
Swan Lake

https://x.com/sameerajayasoma/status/1813314829974950061

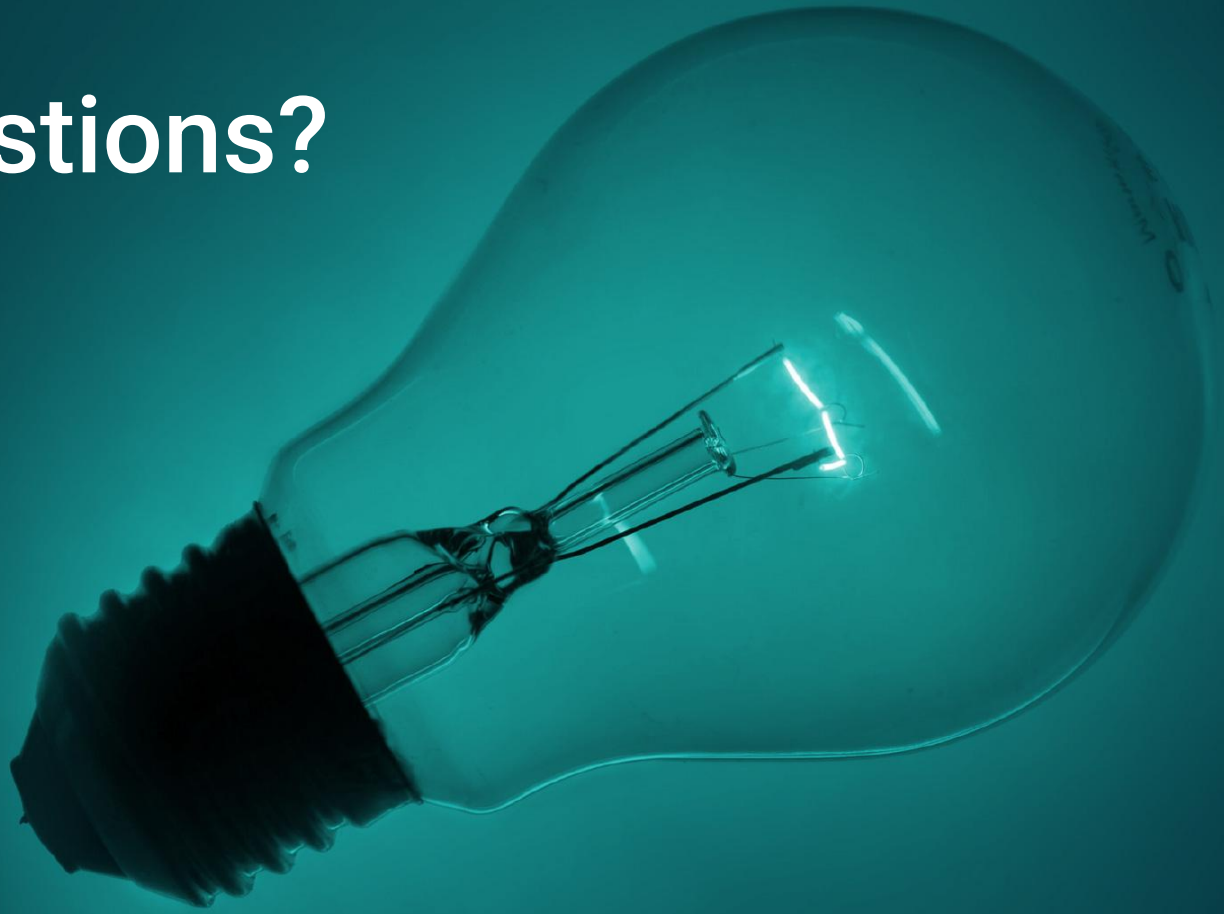
Ballerina is a full platform

- VSCode plugin
  - Source and graphical editing
  - Debugging
- Tools for working with various protocols (REST, GraphQL, gRPC)
- Generate API documentation & test framework
- Ballerina standard library and extended library
- Ballerina Central (https://central.ballerina.io/)
  - Module sharing platform

Let’s code!

Source code: https://github.com/tharindu-nw/hotel_order_service

Questions?



Ballerina community

- Ballerina is an open source project
  https://github.com/ballerina-platform/ballerina-lang/
- Seeking open source contributors
  - Contribute and get rewarded
  - Has good first issues for external contributors
- Ballerina student engagement program
  https://ballerina.io/community/student-program/

Find out more…

- Ballerina documentation
  - Ballerina use cases : Microservices
    - ballerina.io/usecases/microservices/
  - Ballerina by example
    - ballerina.io/learn/by-example/
  - API Documentation
    - https://central.ballerina.io/ballerina-library

- Join the Ballerina community

Discord
ballerinalang

COLLECTIVES™ on stackoverflow
WSO2 Collective

@ballerinalang

GitHub
ballerina-lang

Inter-university Ballerina Hackathon

🔗 Register Now:
https://innovatewithballerina.com/

Organized by the IEEE Student Branch of University of Moratuwa

Thank you!