# ACP / 2

Michael Glienecke, PhD

# Welcome again

- Architectural styles
- Communication protocols / patterns / methods
- A bit about containerization
- Introduction to microservices
- JSON
- REST in more detail
- REST techniques

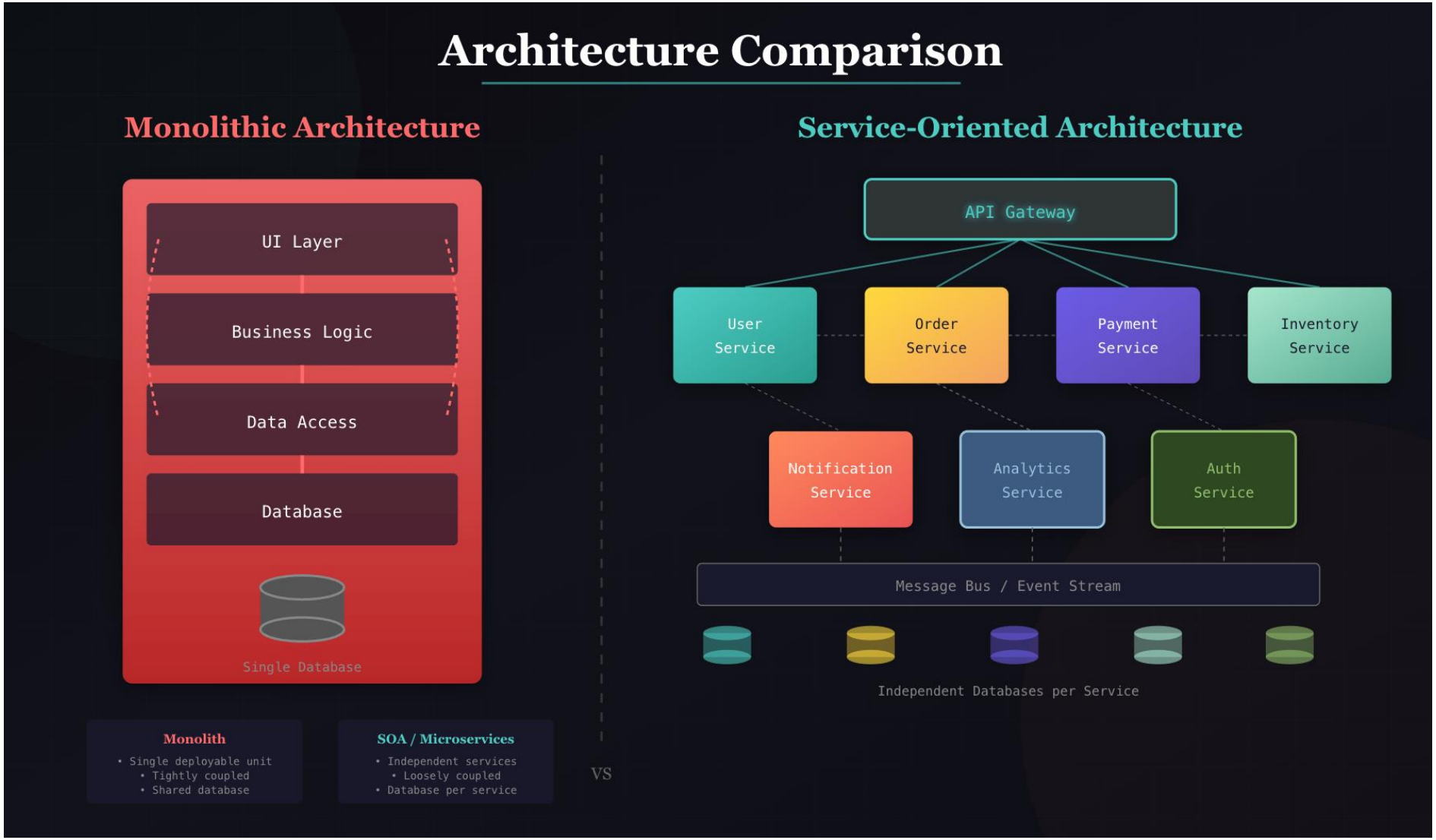
# Architectural styles (not all…)

- Monolith
  - Single process
  - Modular
  - Distributed
- Service oriented architectures (SOA)
  - Services collaborate to achieve some capabilities
  - Often shared databases
  - Larger logical blocks ("services")
  - Scaling often only on the service-level
    - not the service as such on several nodes / services often cannot be spread out

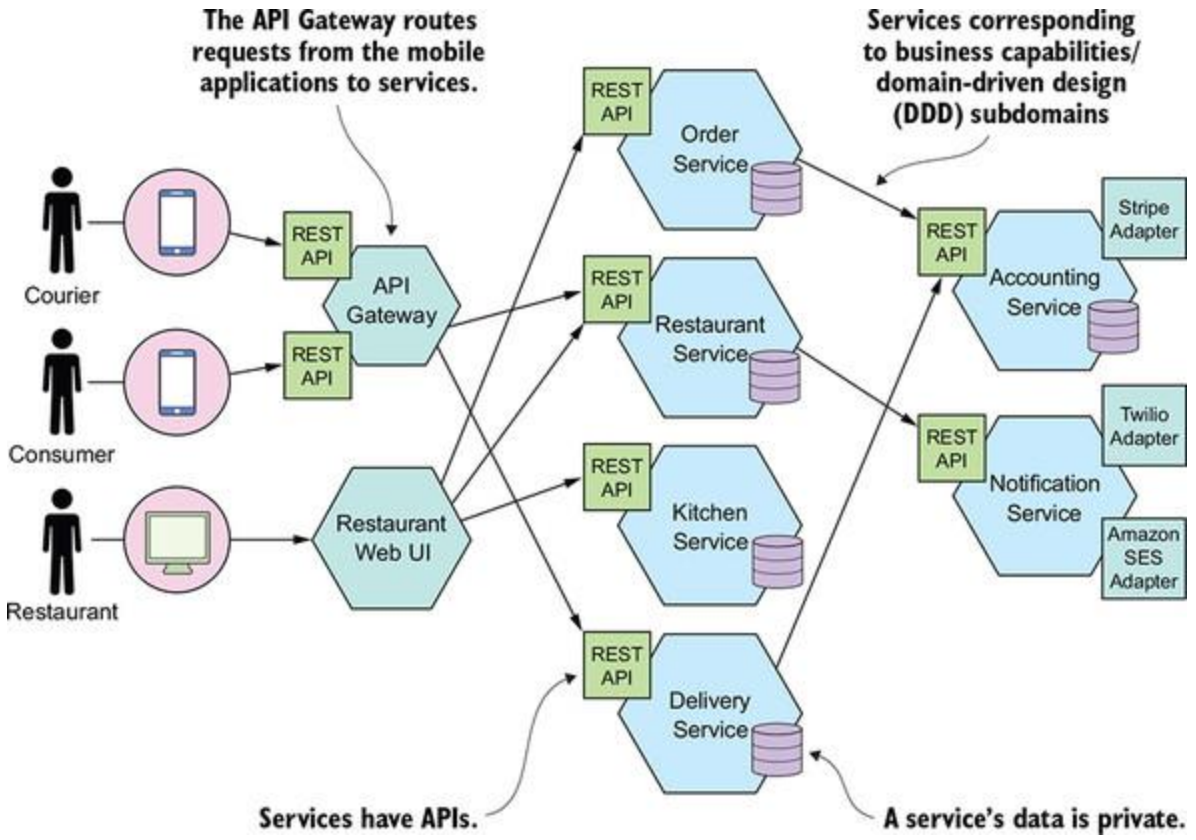
# Architectural styles /2

- Microservices
  - Simple
  - Meshes
- Flow / Event-driven

# Monolith vs Service oriented

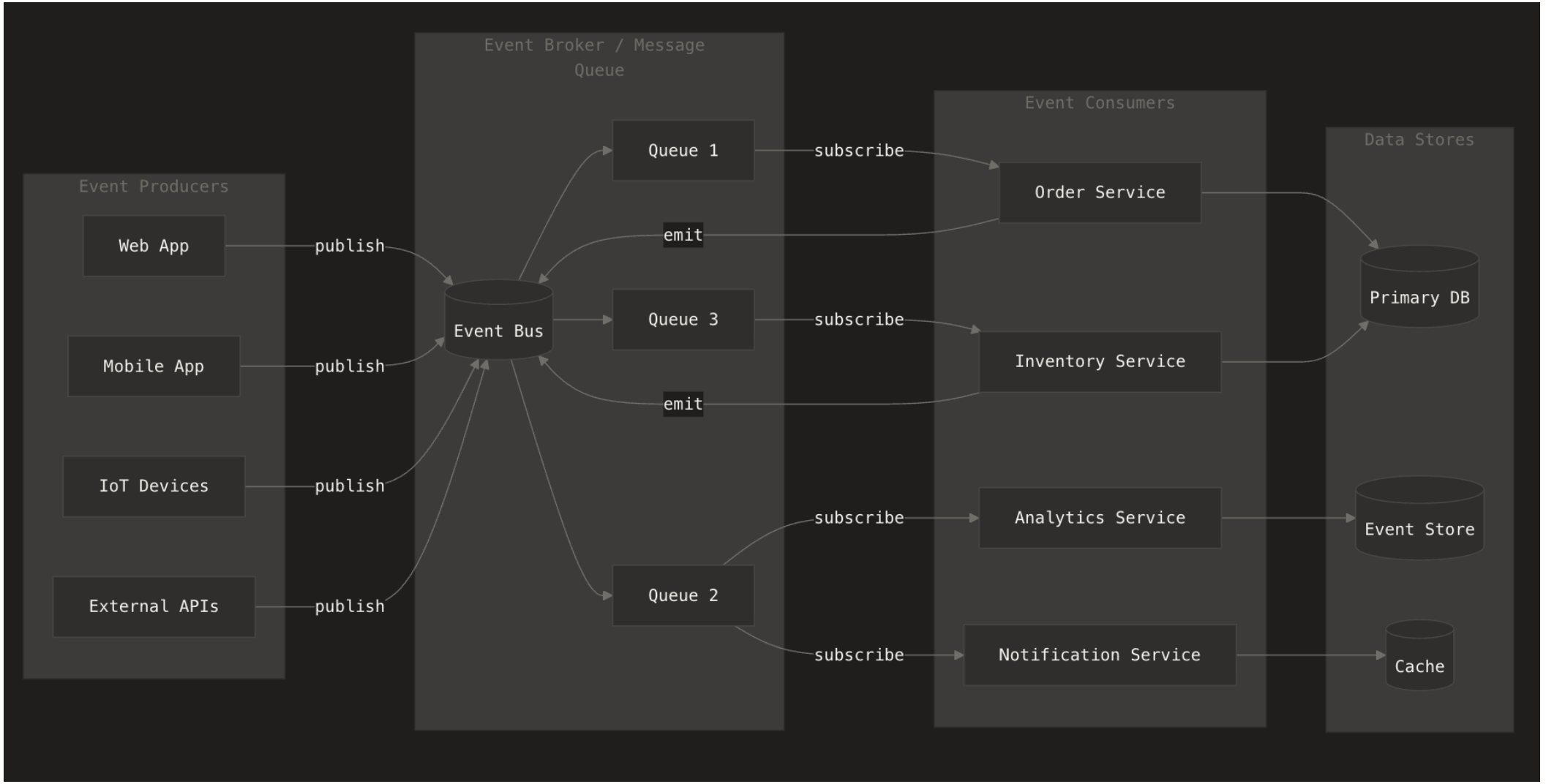


# Point to point - mesh

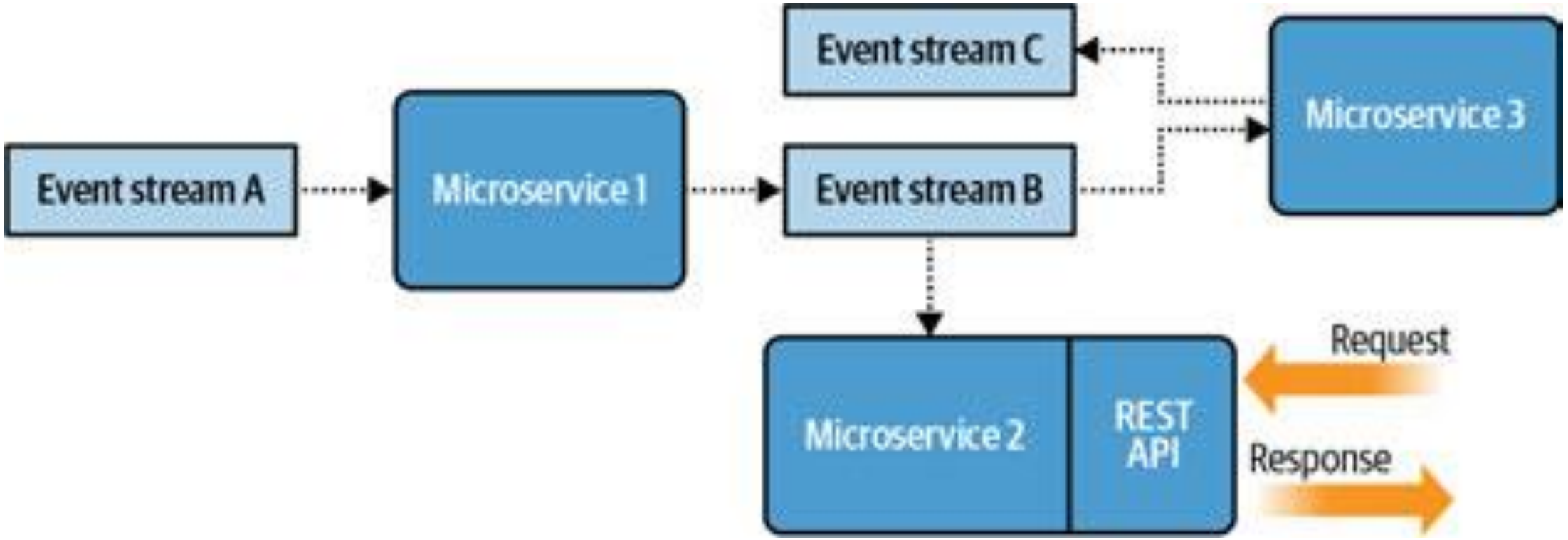



https://livebook.manning.com/book/microservices-patterns/chapter-1/106

# Flow Architecture



# Event based architectures /2



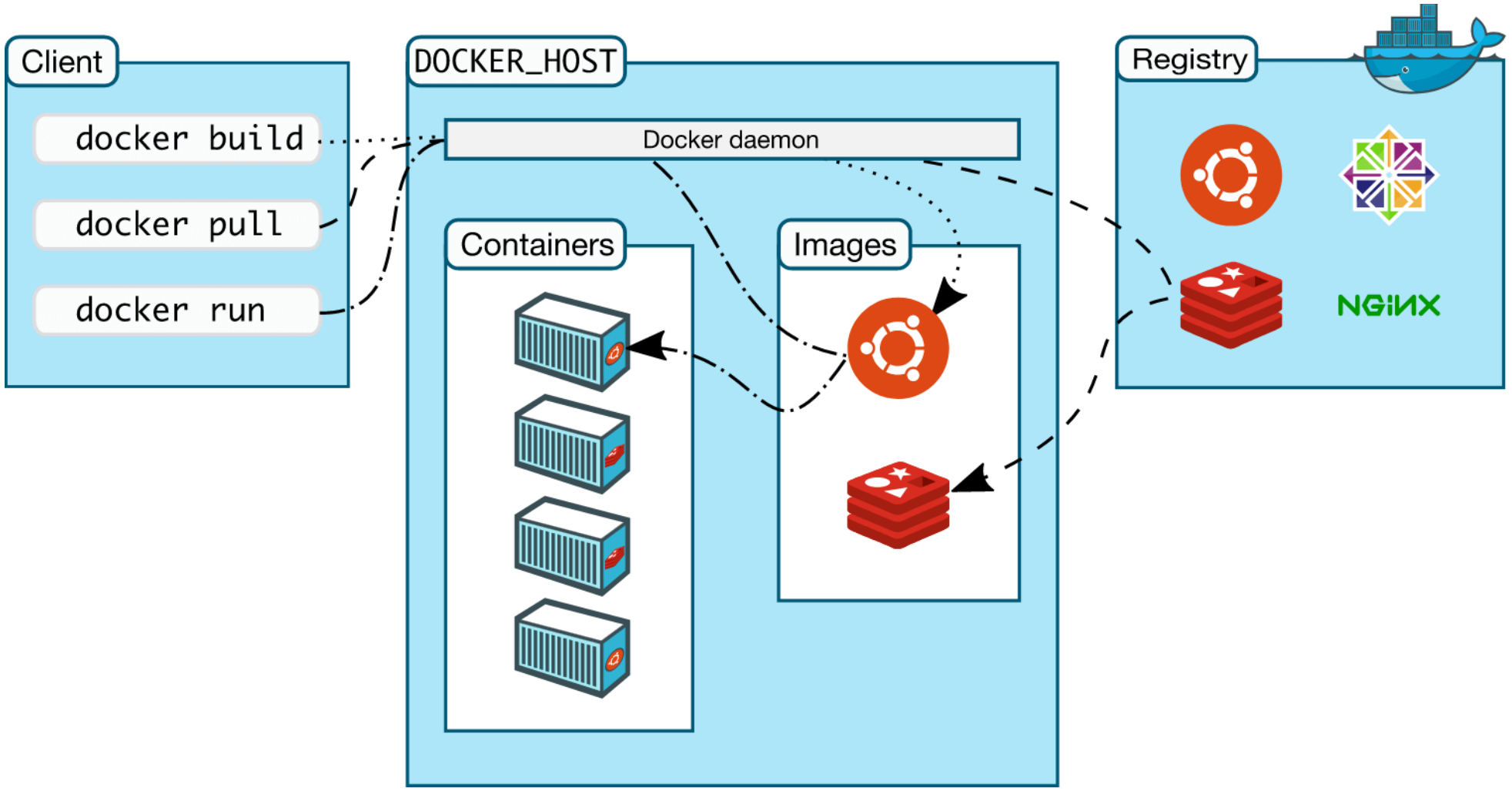

https://eu01.alma.exlibrisgroup.com/leganto/public/44UOE_INST/citation/45644351890002466?auth=SAML

# Containerization - how

- OCI based
  - https://opencontainers.org/about/overview/
  - https://opencontainers.org/community/overview/

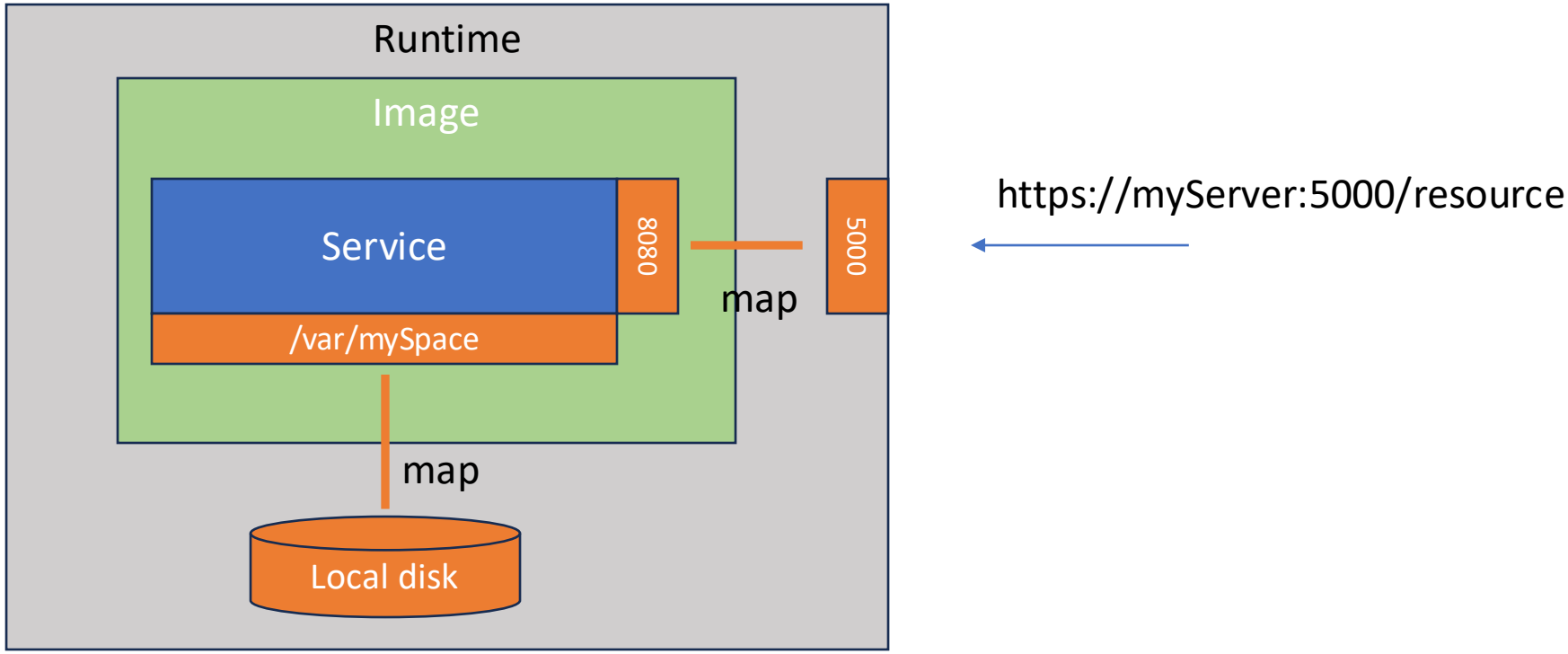
- Images form the runnable applications
- Are loaded on demand
- Executed when needed
- All networking is mapped
- File system access is mapped to volumes

# docker




https://www.cherryservers.com/blog/a-complete-overview-of-docker-architecture

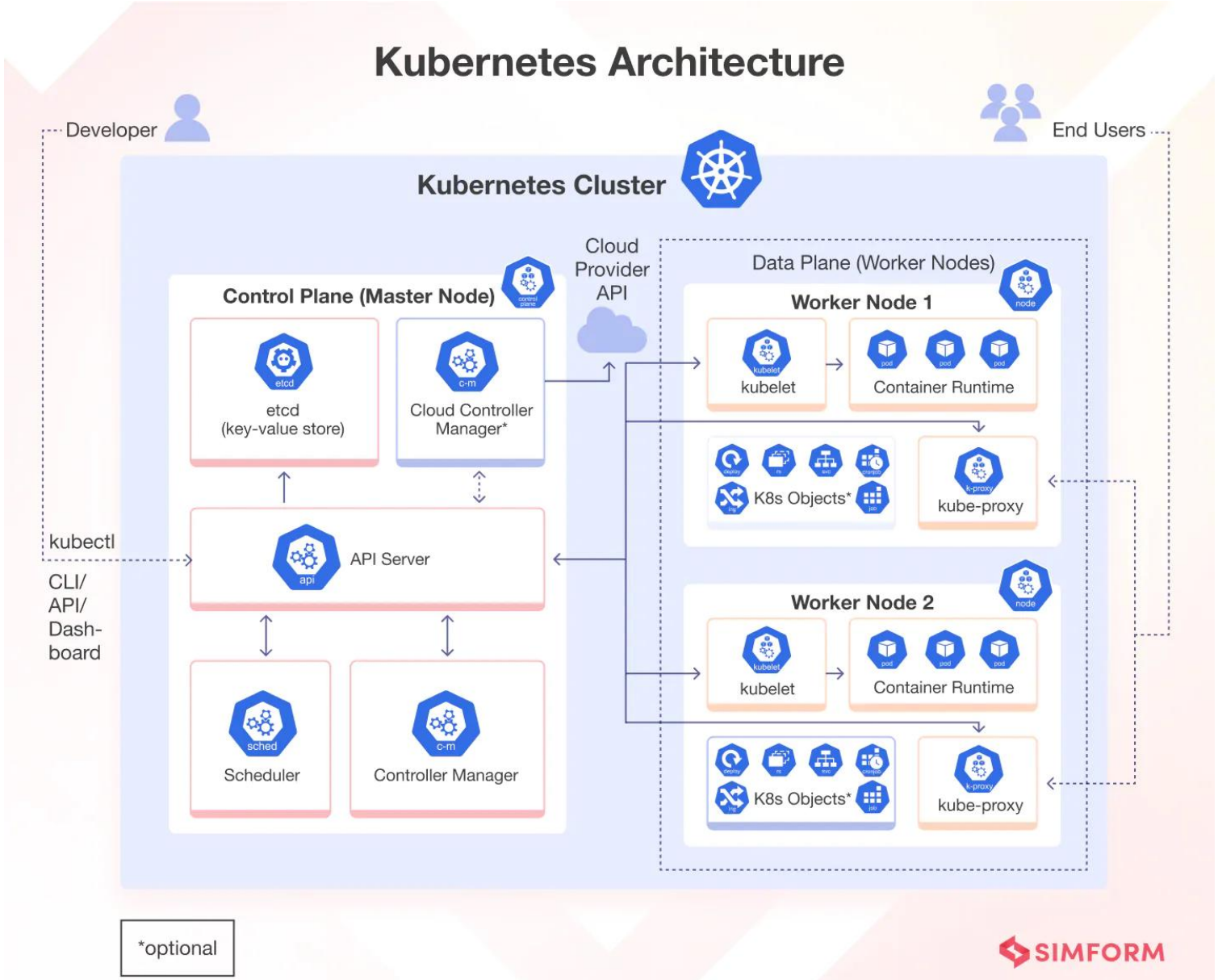
# A microservice in docker



# Running a microservice in docker

- One service / image
- Sidecar as an option

# Kubernetes




https://www.simform.com/blog/kubernetes-architecture/

# Communication protocols

- Request - Response
  - **gRPC**
  - http native
  - SOAP
  - **REST**
  - **GraphQL**
- Messaging
  - **Kafka**
  - MQTT
  - Pulsar
  - AMQP (RabbitMQ, …)

# General

- REST <> http
- REST is
  - Uniform
  - Stateless
  - Client-Server
  - Cacheable
- URL
  - Syntax: https://www.w3.org/Addressing/URL/url-spec.html#:~:text=URL%20syntax,in%20an%20a%20later%20section.
  - BNF: https://www.w3.org/Addressing/URL/5_BNF.html

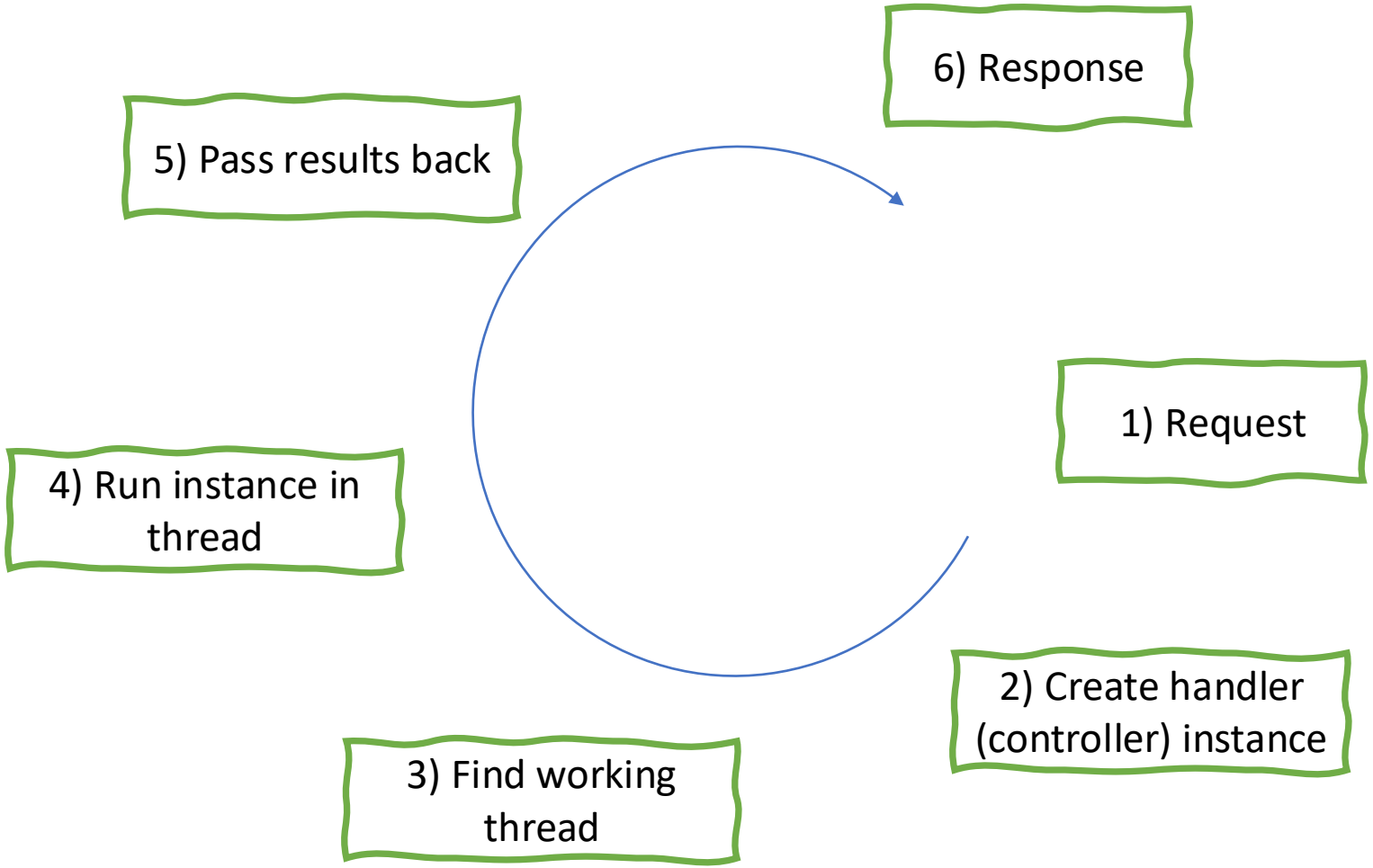
# JSON

- https://www.json.org/json-en.html
- Lightweight, human readable data-interchange format
- JSON vs XML: https://json.org/example.html
- Downsides:
  - No error handling
  - Less expressive and flexible than XML
  - No comments, namespaces, attributes (hard to add metadata)
  - No versioning
  - No security
  - No XSD
- But… Everybody knows it, everybody uses it…

# Introduction to microservices

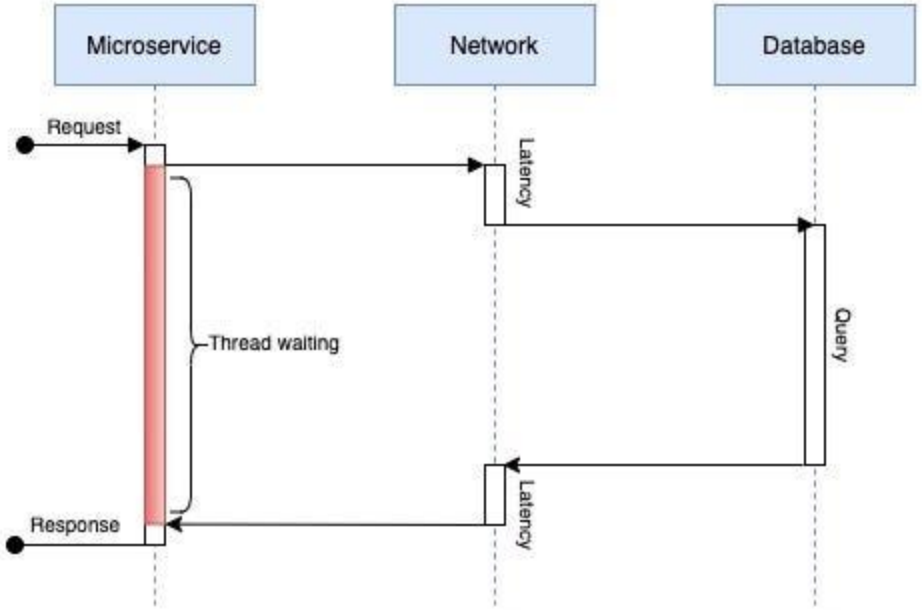
- https://microservices.io/
- Independent releasable / deployable services
- Should model a business domain - "DDD"
- All functionality and data is encapsulated
  - Each service has its own data / function
  - "Information hiding"
- All functionality and data is made available to others via networks (as API)
  - The word API is very confusing sometimes…
- Loose coupling / High Cohesion

# The structure of a microservice (Spring)



# Microservices structure

- Spring Boot uses Tomcat
  - Some limitations (https://oskar-uit-de-bos.medium.com/the-performance-challenge-in-java-microservices-e51cce3977e9)
- Blocking threads and consuming resources



# Example microservice in Java

- Main entry point: https://github.com/mglienecke/IlpTutorialRestService

# Tools to test microservices

- REST
  - Postman
  - curl
  - Swagger
  - IntelliJ (built in)
- gRPC
  - gRPCurl
  - Postman
- Kafka / RabbitMQ
  - Command line (shell) tools & UI

# Communication styles in general

- Synchronous
- Asynchronous

# Asynchronous

- One-to-one
  - Async request / response
  - One way notifications
- One-to-many
  - Publish / subscribe
  - Publish / async responses
- Issue Request - Response will arrive at some time
  - or never -> error handling is crucial
  - Sometimes "fire and forget" is intended (status updates) -> like UDP
  - Usually handled with promises (e.g. JavaScript and AJAX) in client-programming

# Asynchronous

- True async response handling is tricky
  - Where does the server send the response to?
    - Web-Callback
    - Polling (yes, an ugly word, but sometimes useful and cheap – like a Spin-Lock Semaphore)
  - Signal-R (Chat, Async notifications, mostly done using UDP)
- Often the client starts a "sync" background thread to wait for the answer to simulate async behaviour
- Messaging is async by design (Kafka, MQ, AQMP, …)
  - And reliable

# Synchronous

- Request -> Response with wait
  - Causes blocking of the requester until response arrives or timeout
- Can be achieved with "promises" in asynchronous manner as well
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
- Sometimes the easiest and most useful to do
  - e.g. logon -> you really want to wait until you get the response
- In general, avoid if you can
  - Load issues, scaling, resource blocking

# How does this map to the Azure Blob API?

- https://learn.microsoft.com/en-us/rest/api/storageservices/blob-service-rest-api
- Azure Blob is a collection of endpoints in REST
- There presumably are several "controllers" which instantiate handlers for requests
- These perform the actual request and return the results