MICROSERVICES IN .NET CORE

13.10.2019

About me

- ASP.NET, C# Developer
- 6 years software engineer experience

Summary

Intro

- Understanding Microservices
- Monolith vs Microservices
- Microservices Features
- Advantages Of Microservices
- Microservices pitfalls
- Best practices

Summary

Application design

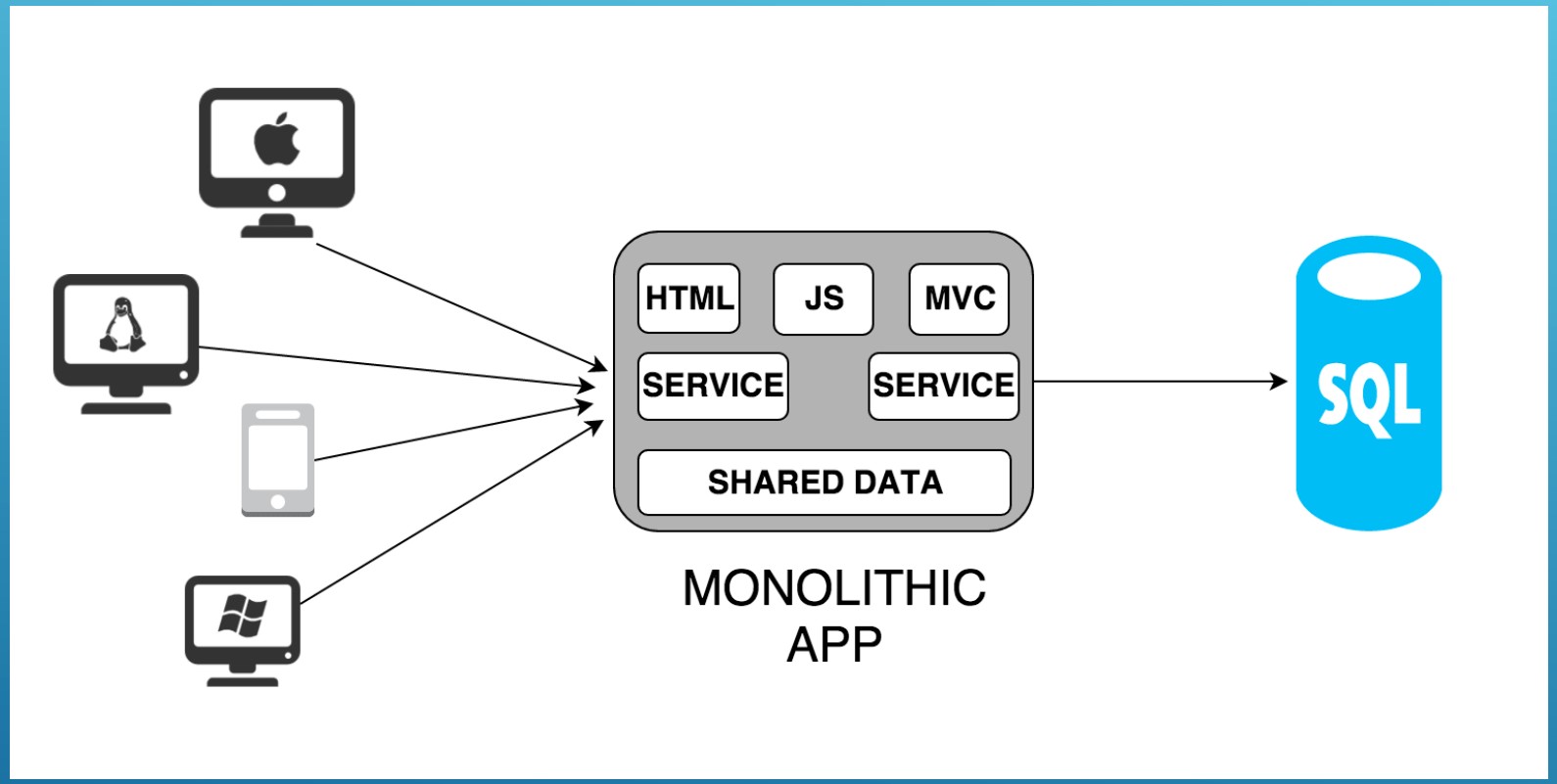
- Domain-driven design
- Design patterns
- Tools

Intro

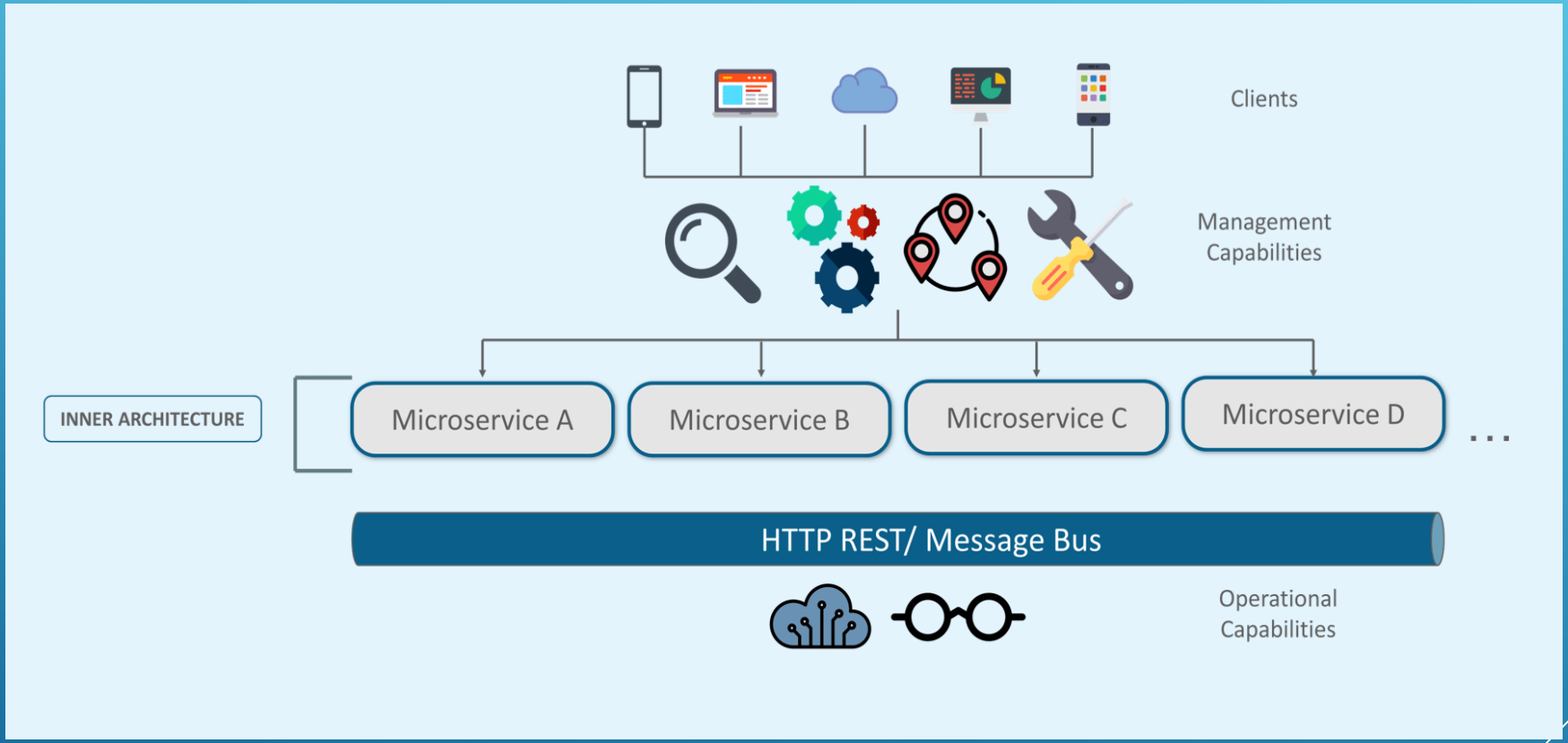
Understanding Microservices

- An approach to application development.
- A suite of modular components or services.
- Each microservice is responsible for a single feature.
- Communicate over well-defined APIs.
- Running in its own process.
- Built around business capabilities.

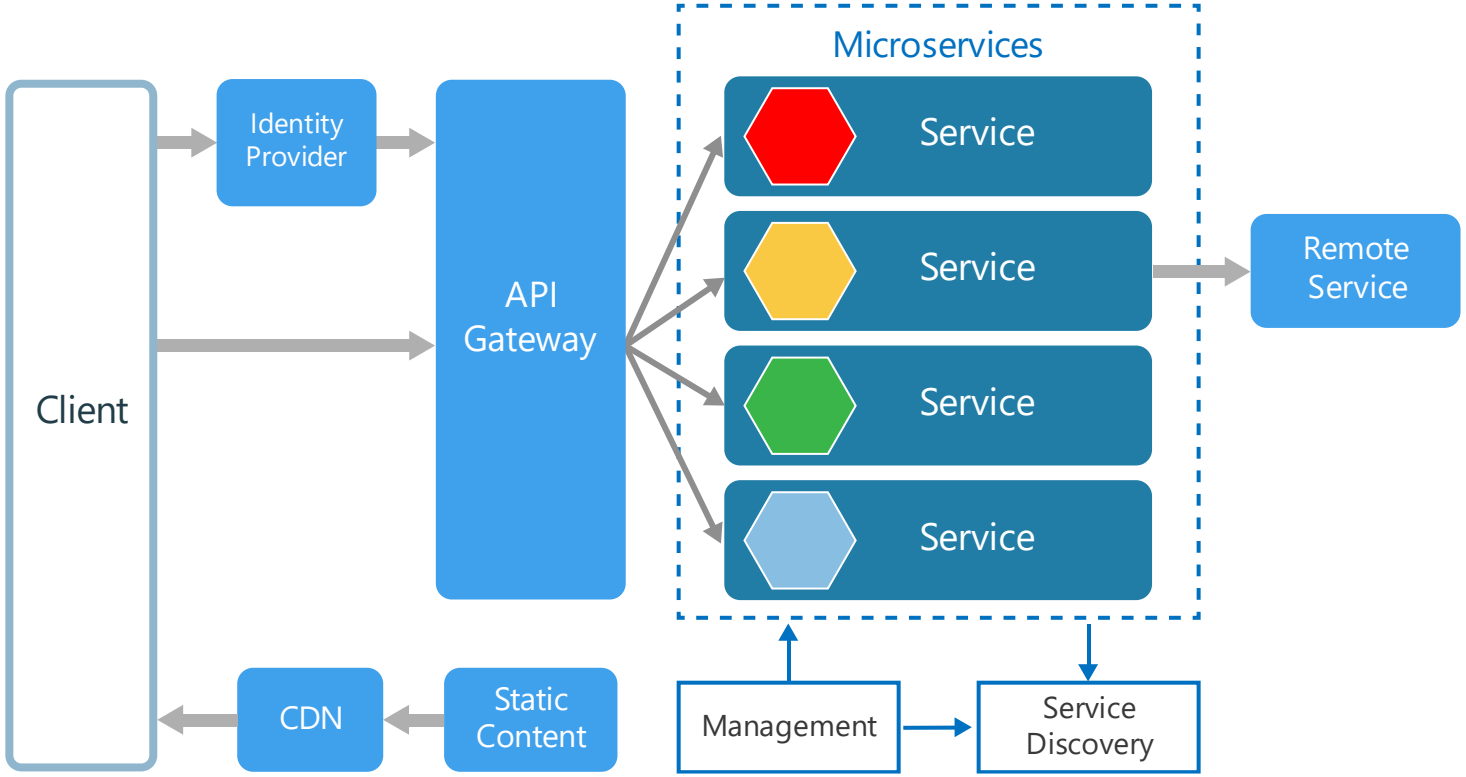
Monolith vs Microservices



Monolith vs Microservices



Monolith vs Microservices



Microservices Features

Microservices Features

- **Decoupling** – Services within a system are largely decoupled. So the application as a whole can be easily built, altered, and scaled
- **Componentization** – Microservices are treated as independent components that can be easily replaced and upgraded

Microservices Features

- **Business Capabilities** – Microservices are very simple and focus on a single capability
- **Autonomy** – Developers and teams can work independently of each other, thus increasing speed

Microservices Features

- **Continuous Delivery** – Allows frequent releases of software, through systematic automation of software creation, testing, and approval
- **Responsibility** – Microservices do not focus on applications as projects. Instead, they treat applications as products for which they are responsible

Microservices Features

- **Decentralized Governance** – The focus is on using the right tool for the right job. That means there is no standardized pattern or any technology pattern. Developers have the freedom to choose the best useful tools to solve their problems
- **Agility** – Microservices support agile development. Any new feature can be quickly developed and discarded again

Advantages Of Microservices

- **Independent Development** – All microservices can be easily developed based on their individual functionality
- **Independent Deployment** – Based on their services, they can be individually deployed in any application

Advantages Of Microservices

- **Fault Isolation** – Even if one service of the application does not work, the system still continues to function
- **Mixed Technology Stack** – Different languages and technologies can be used to build different services of the same application

Advantages Of Microservices

- **Granular Scaling** – Individual components can scale as per need, there is no need to scale all components together

Microservices pitfalls

- **Distribution** - Distributed systems are harder to program, since remote calls are slow and are always at risk of failure.
- **Eventual Consistency** - Maintaining strong consistency is extremely difficult for a distributed system, which means everyone has to manage eventual consistency.

Microservices pitfalls

- **Operational Complexity** - You need a mature operations team to manage lots of services, which are being redeployed regularly.

Best practices

- https://microservice.guide/
- https://morioh.com/p/ff384fcdad34

Application design

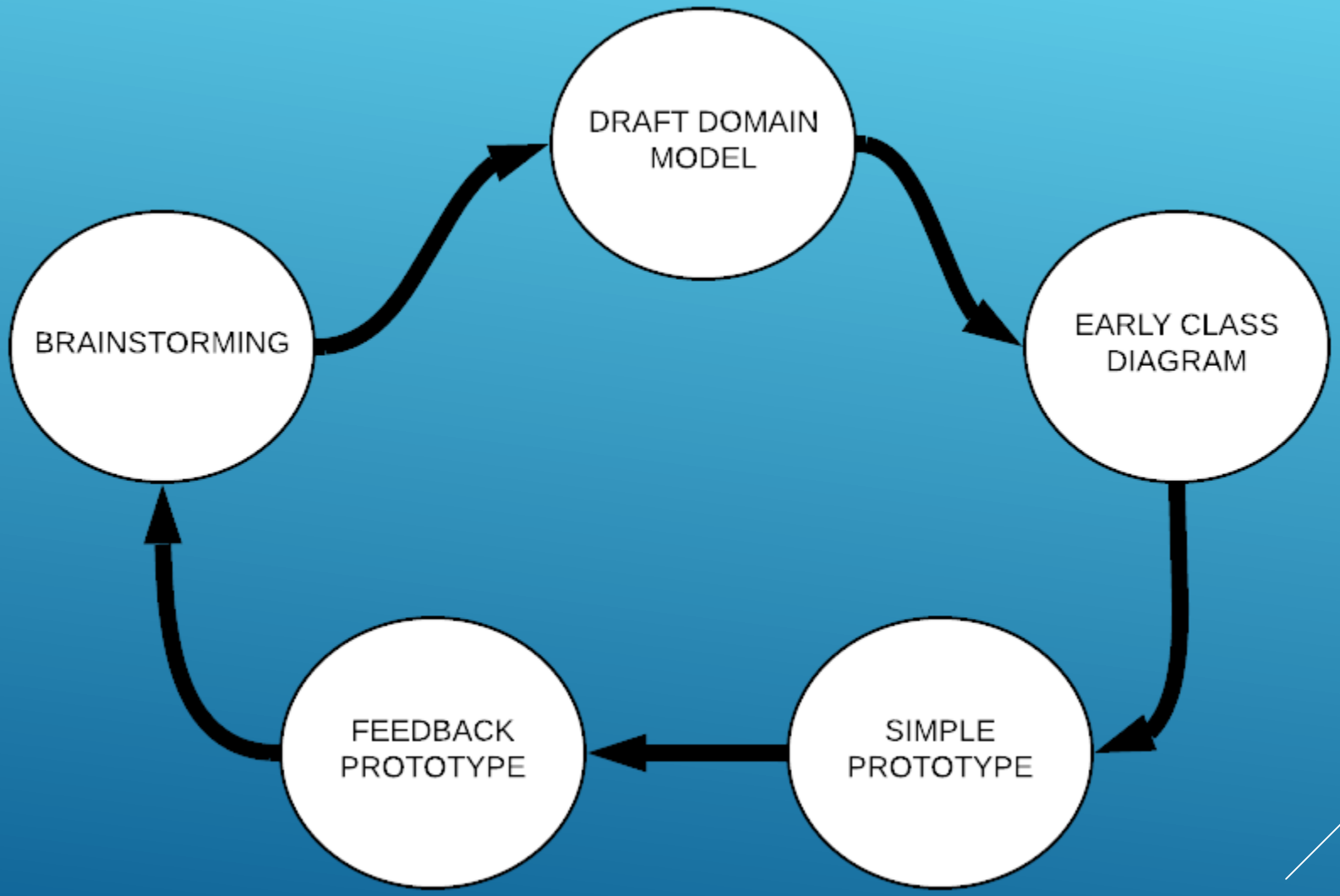
Domain-driven design

- **Context** - This is the setting in which a word or statement appears that determines its meaning.
- **Domain** - This is a sphere of knowledge (ontology), influence, or activity. The subject area to which the user applies a program is the domain of the software.

Domain-driven design

- **Model** - This is a system of abstractions that describes selected aspects of a domain and can be used to solve problems related to that domain.
- **Ubiquitous language** - This is a language structured around the domain model and used by all team members to connect all the activities of the team with the software.

Steps to follow



Single knowledge block

- Domain model
- The ubiquitous language
- Code

Ubiquitous language

- Class names and their functions related to the domain
- Terms to discuss the domain rules included on the model
- Names of analysis and design patterns applied to the domain model

How DDD can help MS?

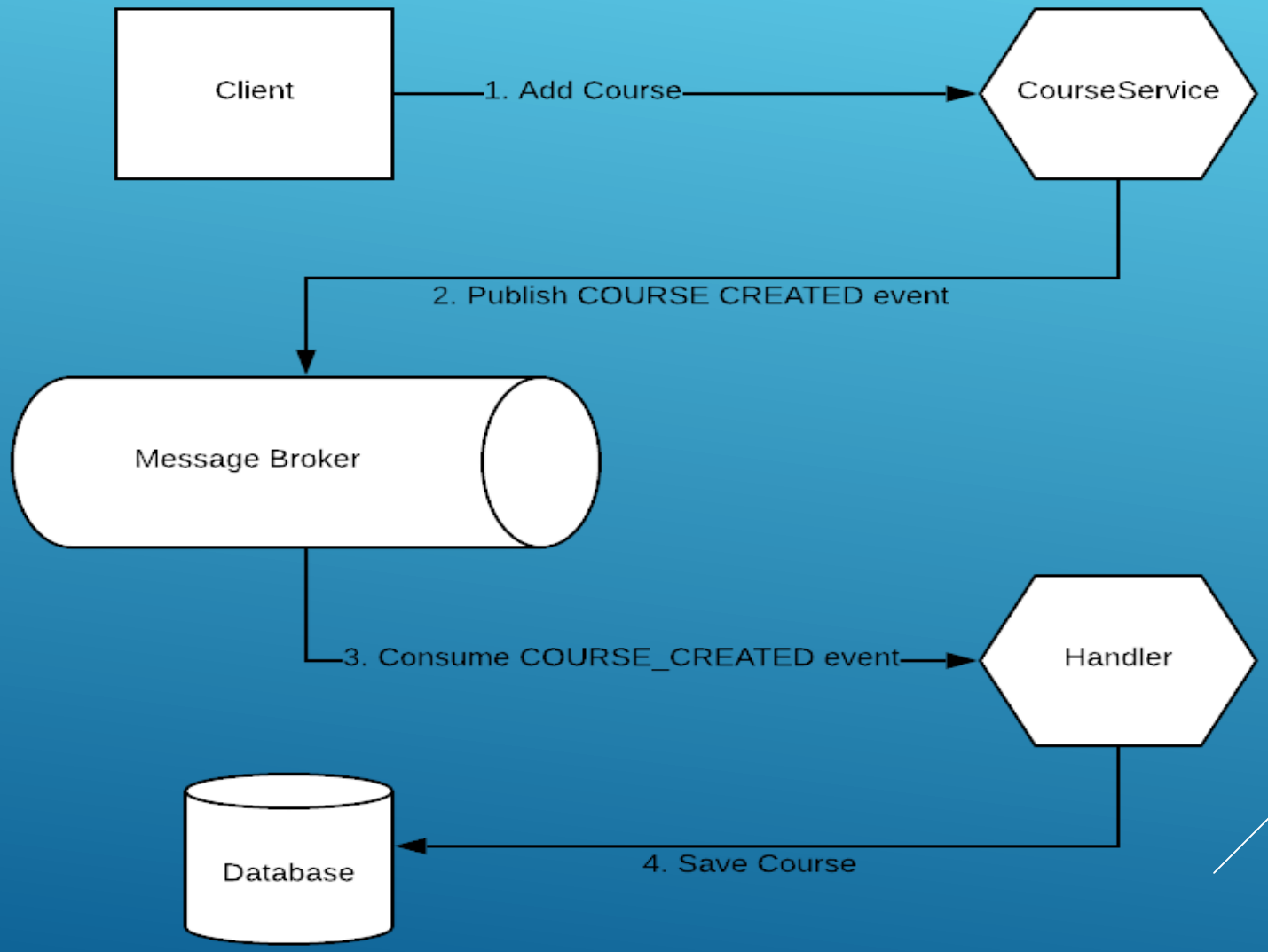
- Scope
- Size

DDD in practice

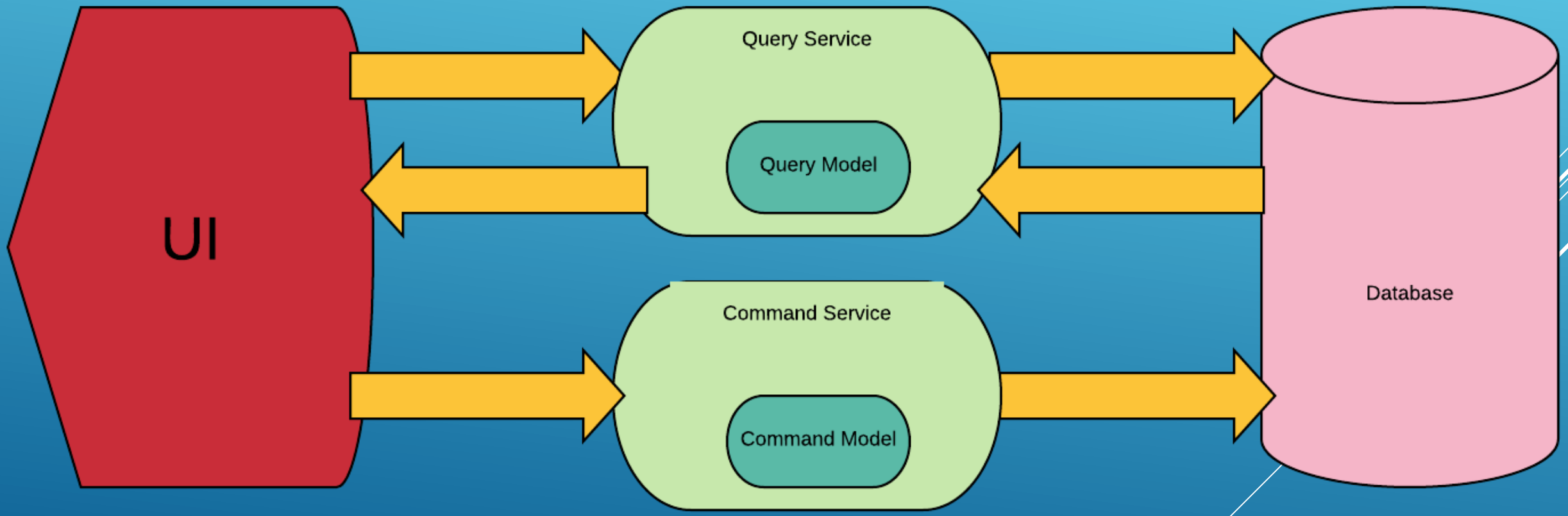
Design patterns

- Event Sourcing
- CQRS

Event Sourcing



CQRS


UI
Query Service
Query Model
Command Service
Command Model
Database

Tools

- RabbitMQ
- Postman

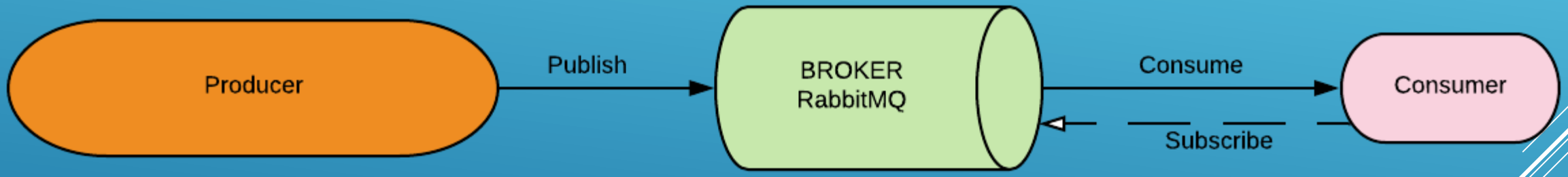
RabbitMQ

- RabbitMQ is a message broker
- **Producer** - Application that sends the messages.
- **Consumer** - Application that receives the messages.
- **Queue** - Buffer that stores messages.

RabbitMQ

- **Message** - Information that is sent from the producer to a consumer through RabbitMQ.
- **Connection** - A connection is a TCP connection between your application and the RabbitMQ broker.
- **Channel** - A channel is a virtual connection inside a connection. When you are publishing or consuming messages from a queue - it's all done over a channel.

RabbitMQ


Producer
Publish
BROKER
RabbitMQ
Consume
Consumer
Subscribe

API Testing Tools

- **Postman** is a powerful tool used to test web services. It was developed for sending HTTP requests in a simple and quick way.

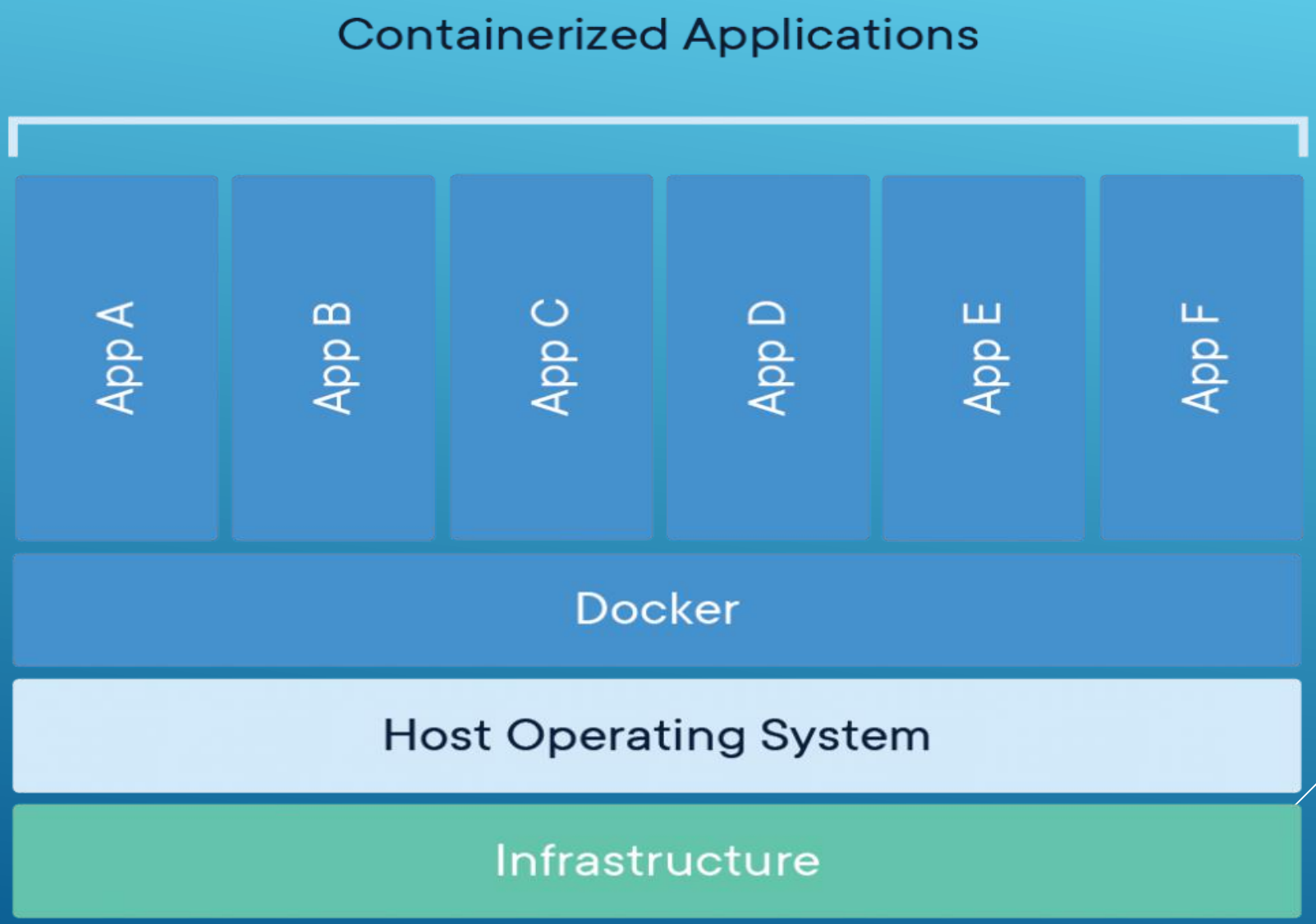
API Testing Tools

- **Curl** is a command-line tool used to deliver requests via HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, LDAP, DAP, DICT, TELNET, FILE, IMAP, POP3, SMTP and RTSP protocols.
- **SoapUI** is a free tool used to test SOAP and RESTful Web Services.
- **vs-rest-api** - A Visual Studio Code (VS Code) extension that provides a REST API to control your editor.

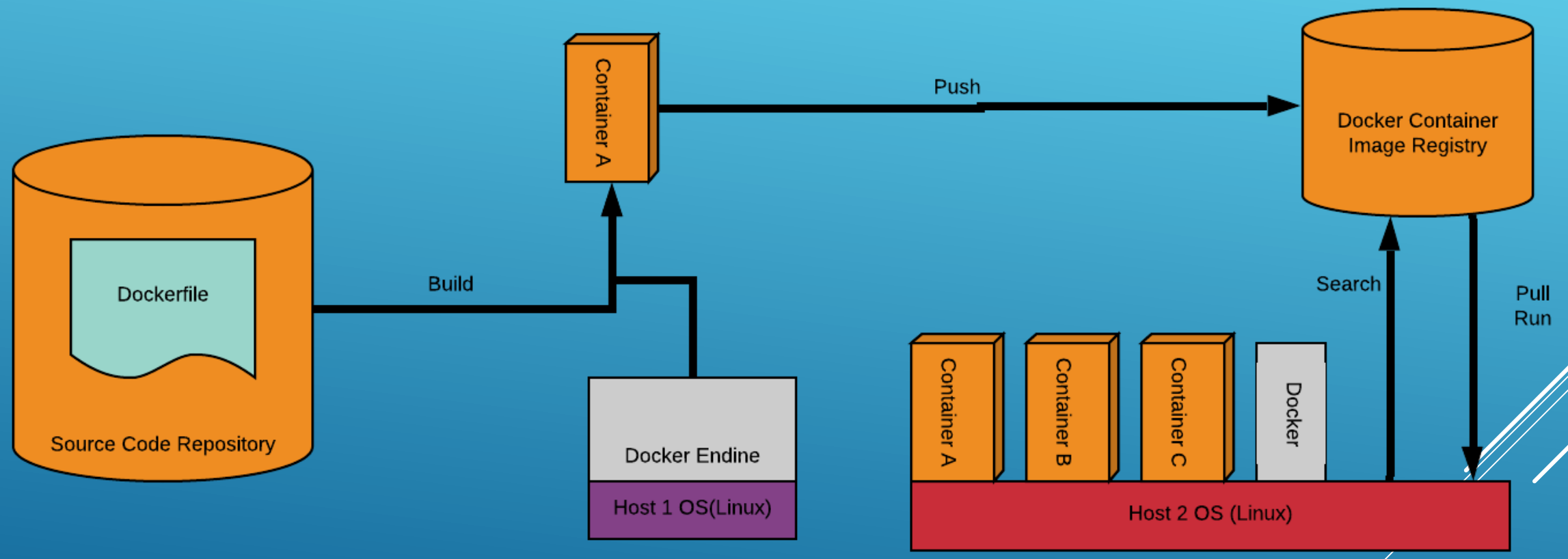
Docker

- The Docker project is a container platform, which lets you run your applications in isolated environments.

Docker


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure

Docker flow


Container A
Push
Docker Container
Image Registry
Dockerfile
Source Code Repository
Build
Docker Endine
Host 1 OS(Linux)
Search
Pull
Run
Container A
Container B
Container C
Docker
Host 2 OS (Linux)

http://bit.ly/peak-it-2019-feedback

FEEDBACK

Completați aici, în sală

Durează 2-3 minute

Feedback anonim - pentru formator si AgileHub