Microservices Elements



Antonio Goncalves
JAVA CHAMPION

@agoncal www.antoniogoncalves.org



Previous Module



Defining microservices

Bounded context

Software development lifecycle

Team more agile

Faster to develop

Time to market



Overview



Microservice terminology

Designing

Data store

Remote communication

Monitoring



Building a Monolith



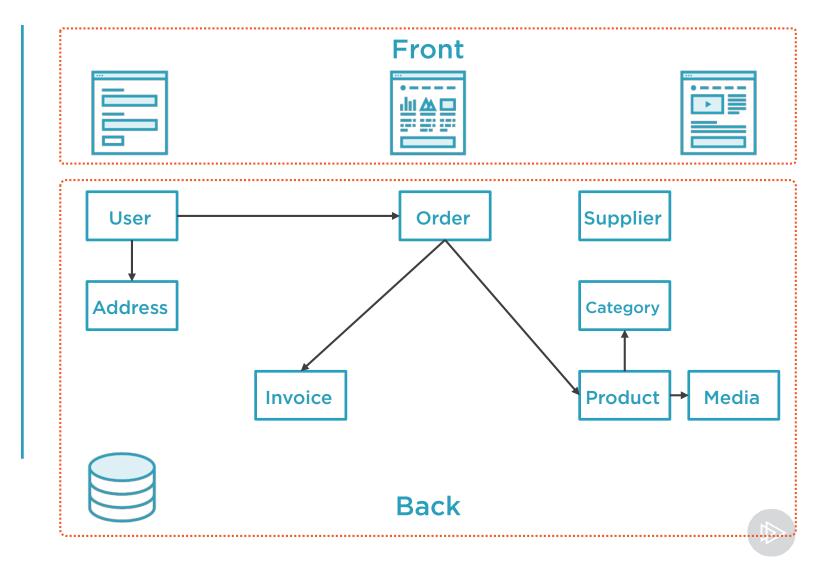
Designing a Monolith

eCommerce webapp

Model

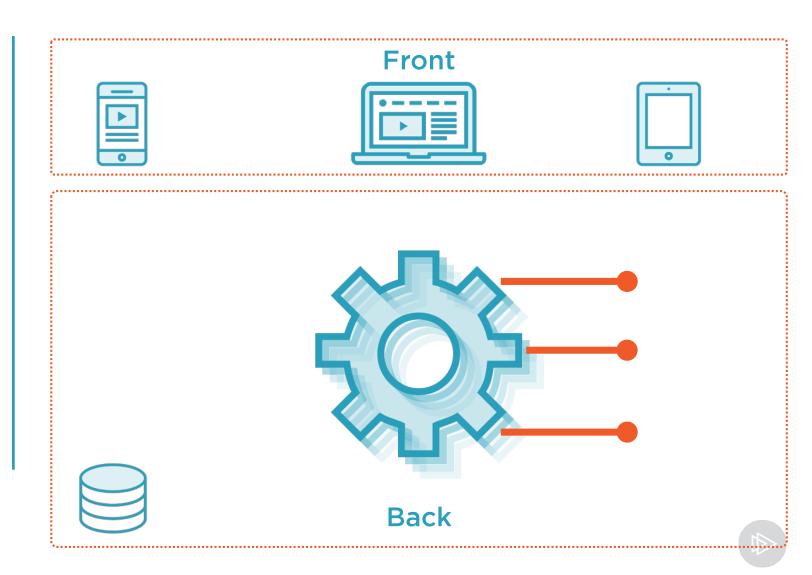
Single database

User interface



Deploying a Monolith

Single monolith
Single database
User interface
Expose APIs
Multiple instances



Monolith

Pros

Simple to develop
Simple to build
Simple to test
Simple to deploy
Simple to scale

Cons

New team members productivity **Growing teams** Code harder to understand No emerging technologies Scale for bad reasons Overloaded container **Huge database**



Building Microservices



Domain Driven Design

Domain

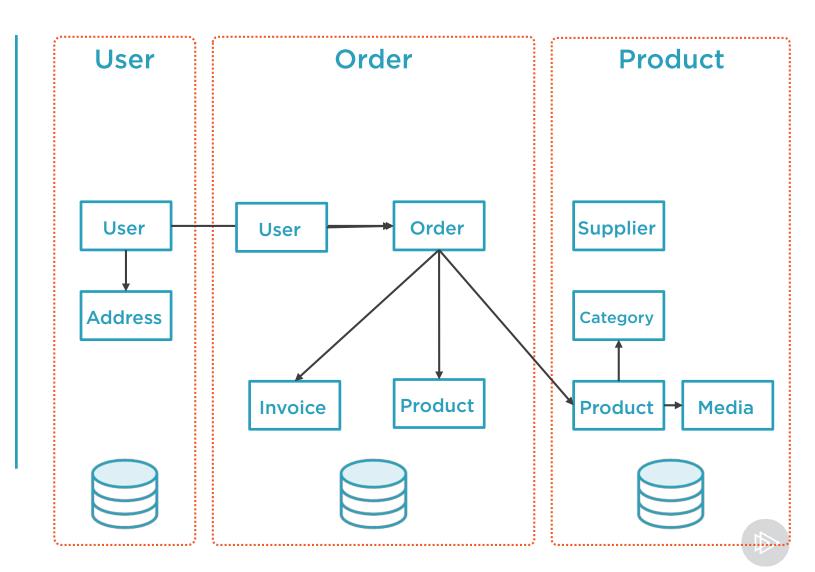
Subdomains

User

Order

Product

Dependencies

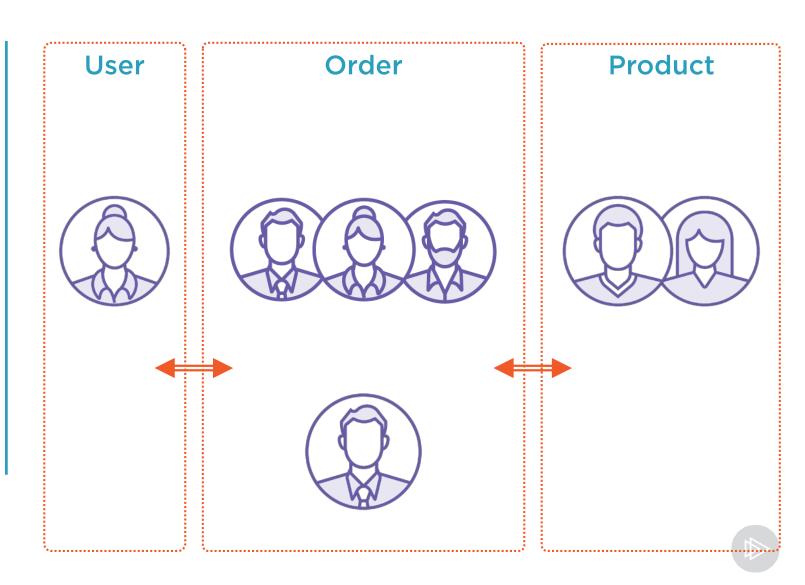


Organization



Teams

Team per subdomain **Right-sized teams** Independent Responsible **Agile and Devops** Communication Management



Codebase

Code

Documentation

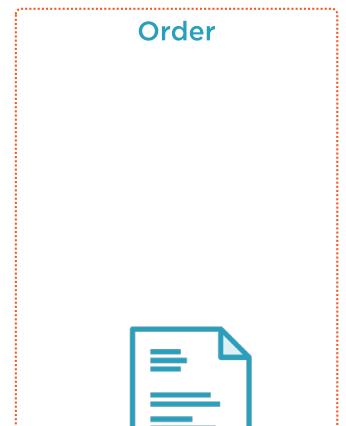
Repository

Git

Subversion

Version







Product

Data Store



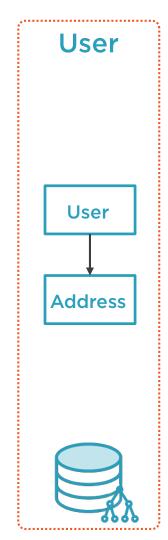
Data Store

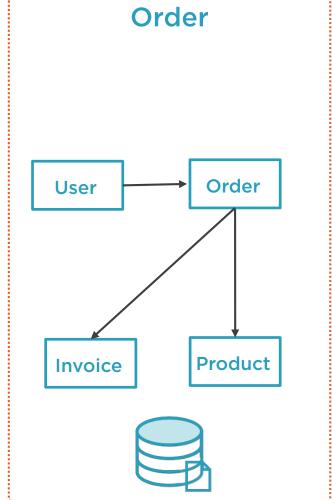
Independent

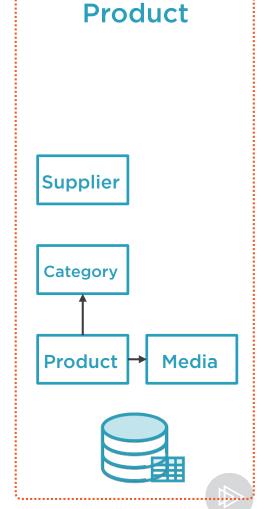
Different requirements

Relational

NoSQL

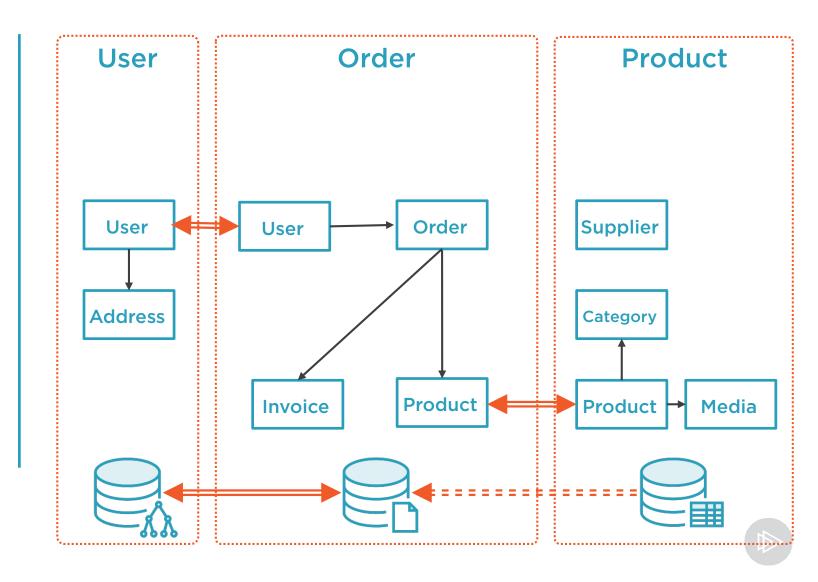






Data Synchronization

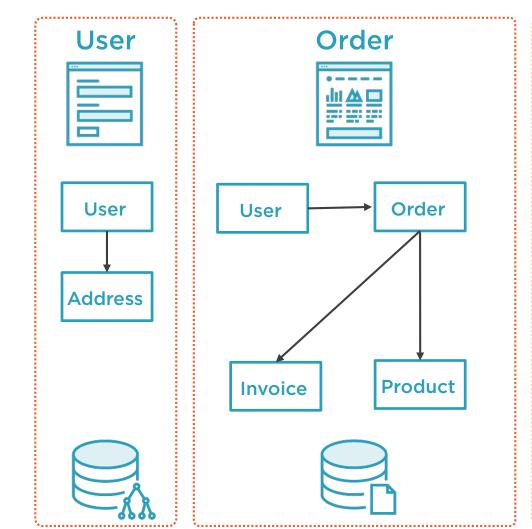
No distributed transaction Immediately consistent **Eventual consistency** Capture data change **Event sourcing** Akka, Kafka, Rabbit MQ Debezium

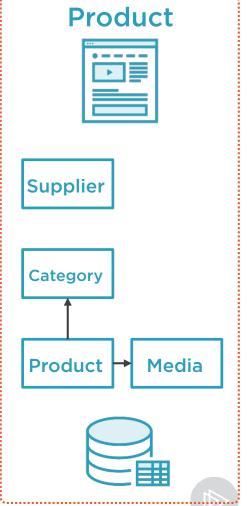




Independent teams

Own set of components

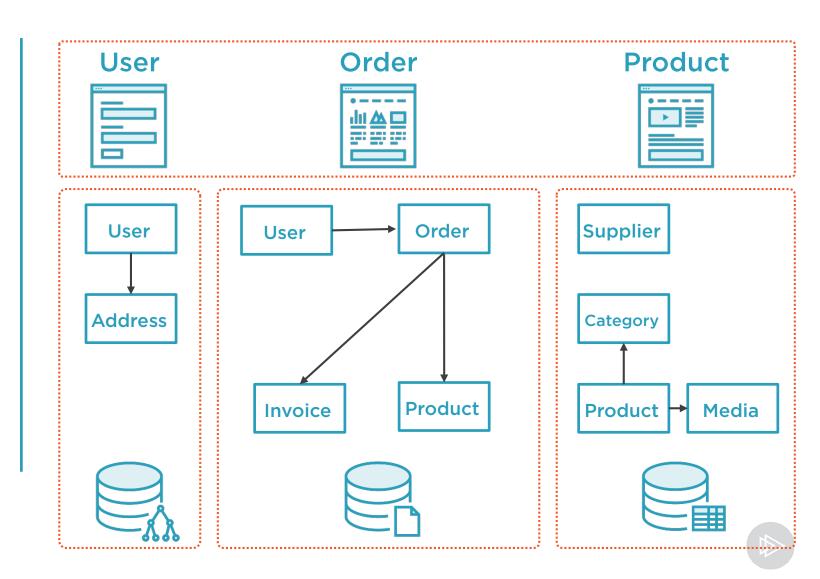




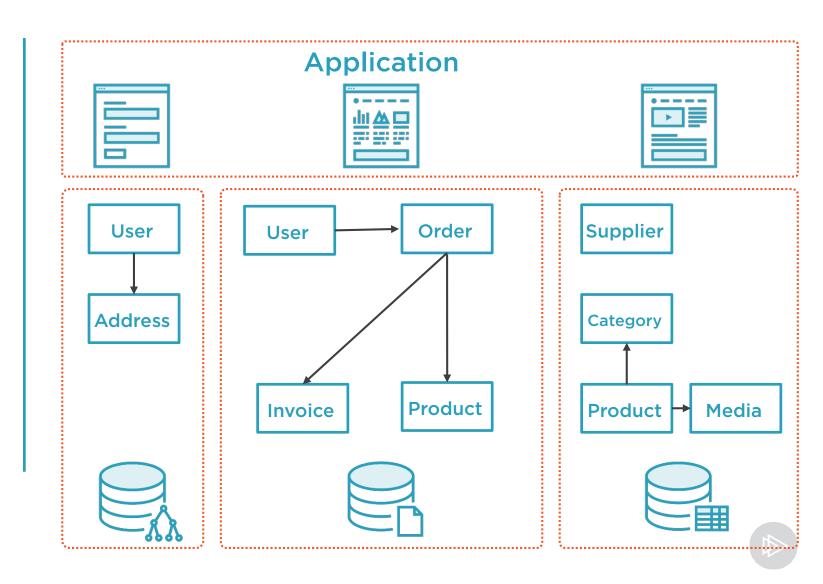
Independent teams

Own set of components

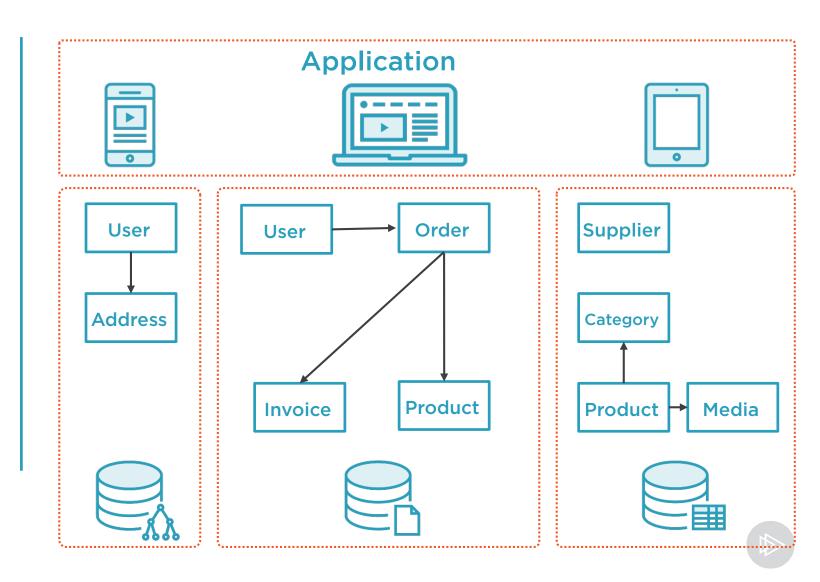
Unique UI



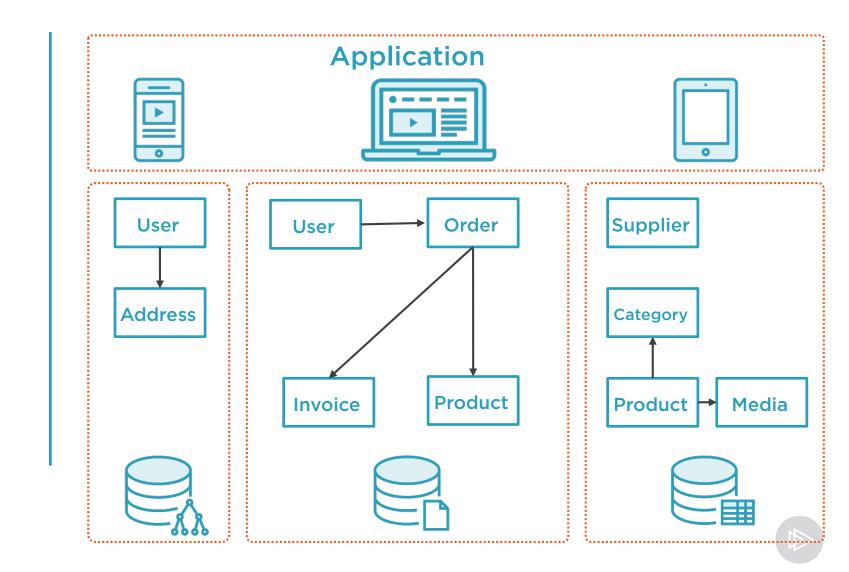
Independent teams Own set of components **Unique UI** Single application **UI** composition Server side Client side

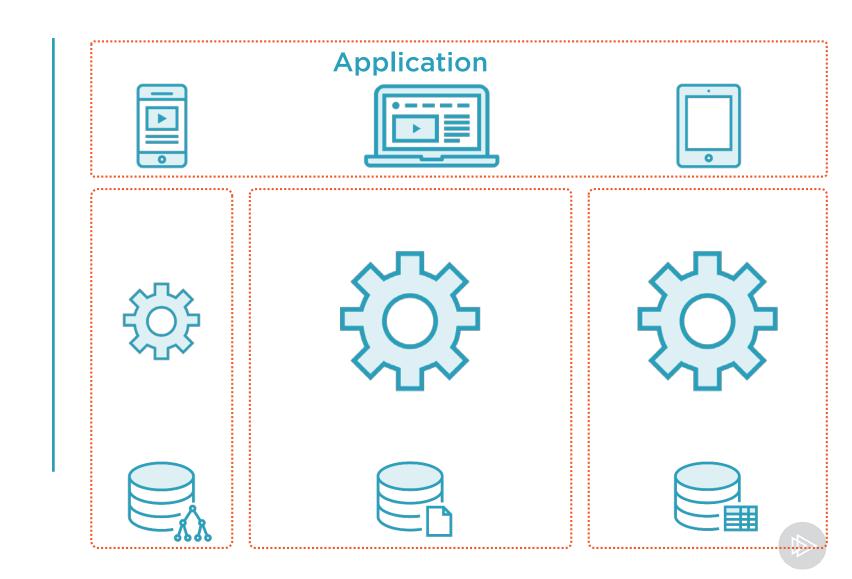


Independent teams Own set of components **Unique UI** Single application **UI** composition Server side Client side



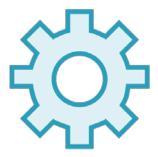














Remote Procedure Invocation

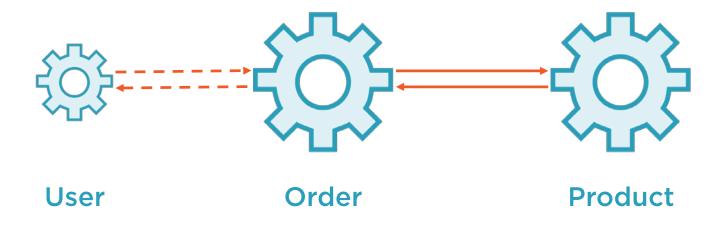
RPC

Request/reply

Synchronous

Asynchronous

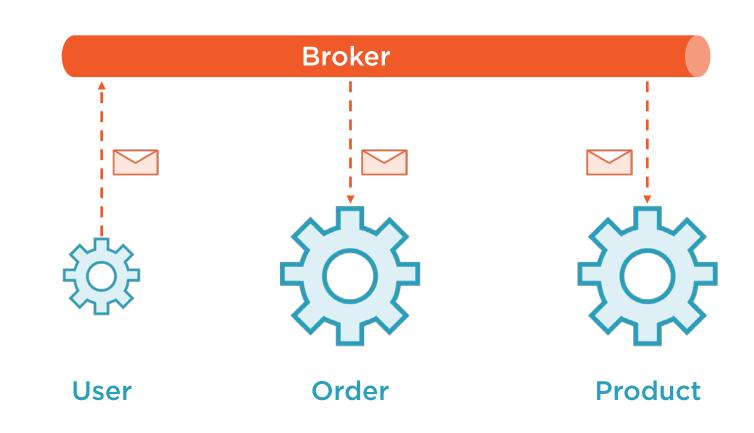
REST, SOAP, gRPC





Messaging

Message or event
Broker or channel
Publish
Subscribe
Loosely couple
Kafka, RabbitMQ



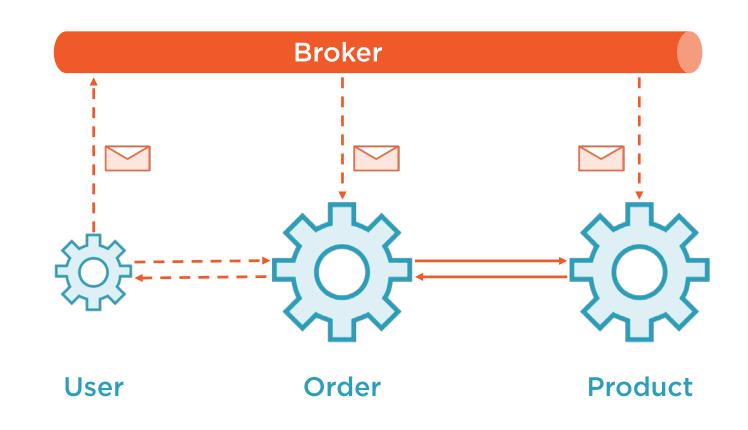


Protocol Format Exchange

Text

XML, JSON, YAML Human readable Easy implement

> Binary gRPC More compact





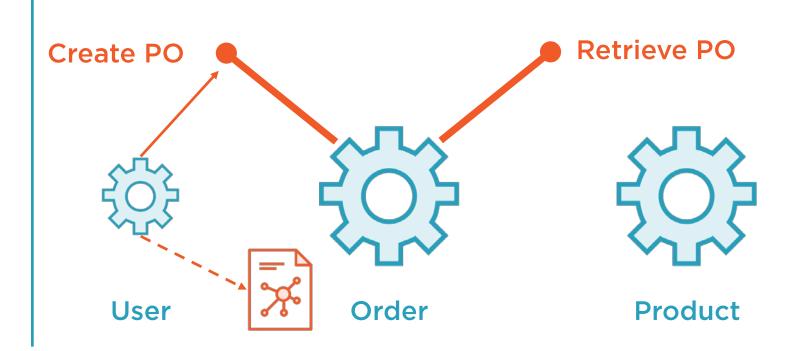
APIs and Contracts

Application program interface

Contract

SOAP, REST, gRPC

WSDL, Swagger, IDL





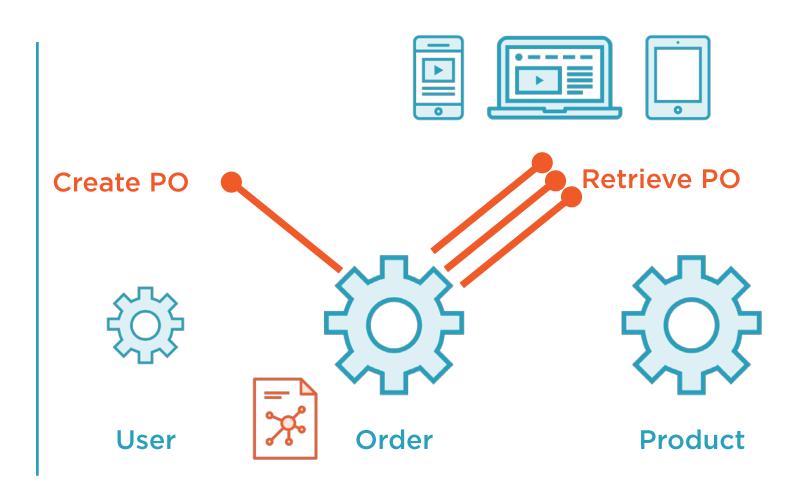
APIs and Contracts per Device

Different devices

Different needs

Different APIs

Different contracts





Distributed Services



Service Registry

Locations change

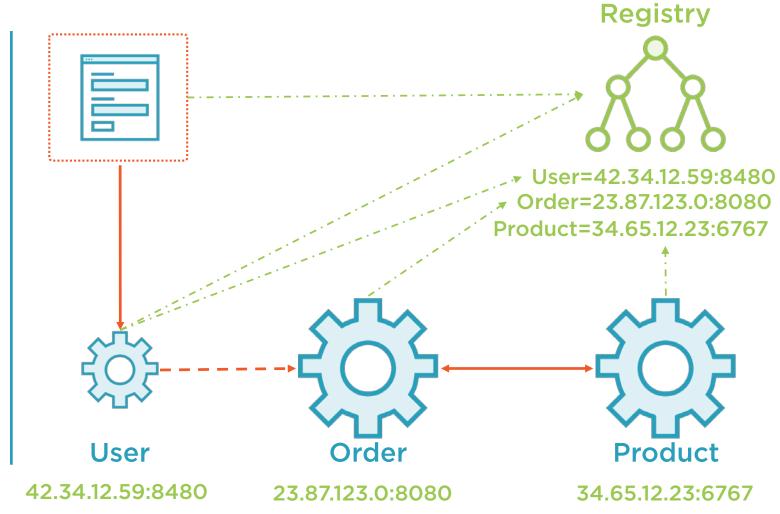
Phone book

Self registration

Discovery

Invocation

Eureka, Zookeeper, Consul





Cross-origin Resource Sharing

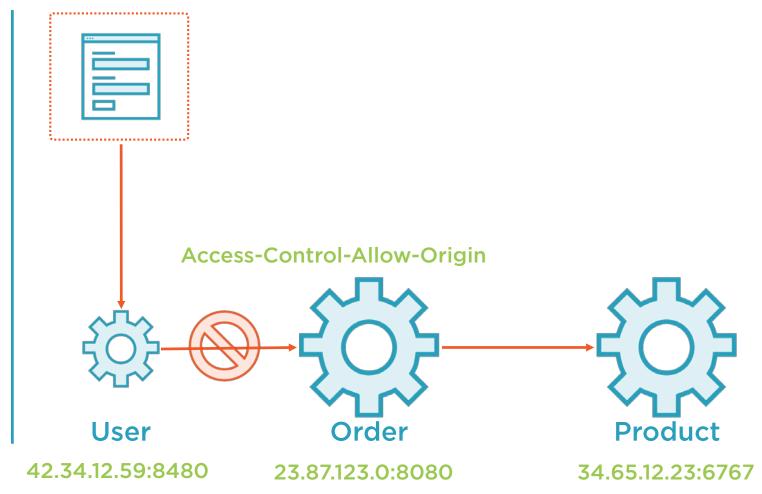
CORS

Same-origin policy

Same protocol, server, port

Restrict cross-origin

HTTP headers

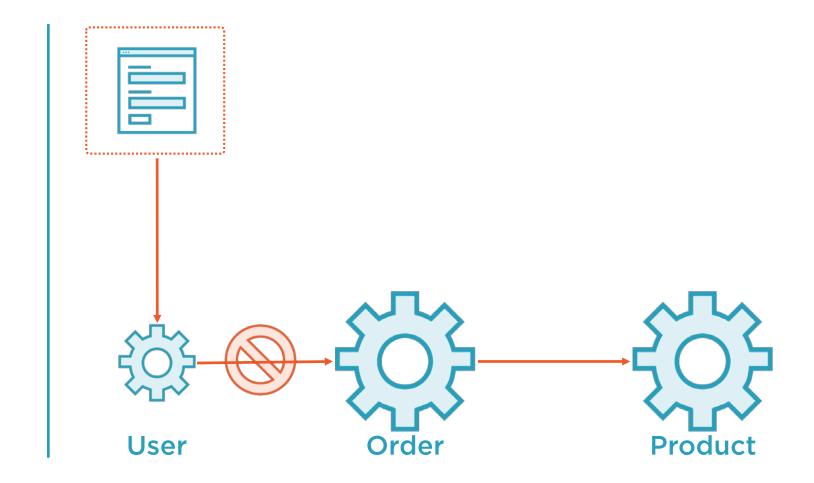




Circuit Breaker

Services available

Network failure
Heavy load
Domino effect





Circuit Breaker

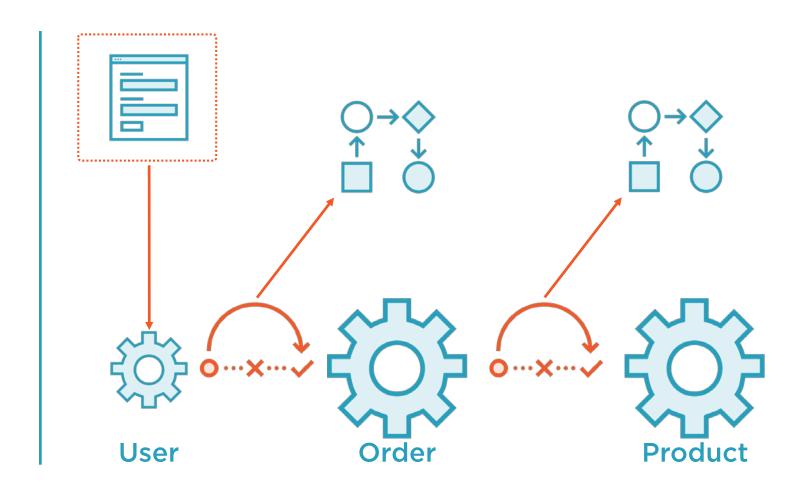
Services available

Network failure
Heavy load
Domino effect

Invoke via proxy

Deviate calls Reintroduce traffic

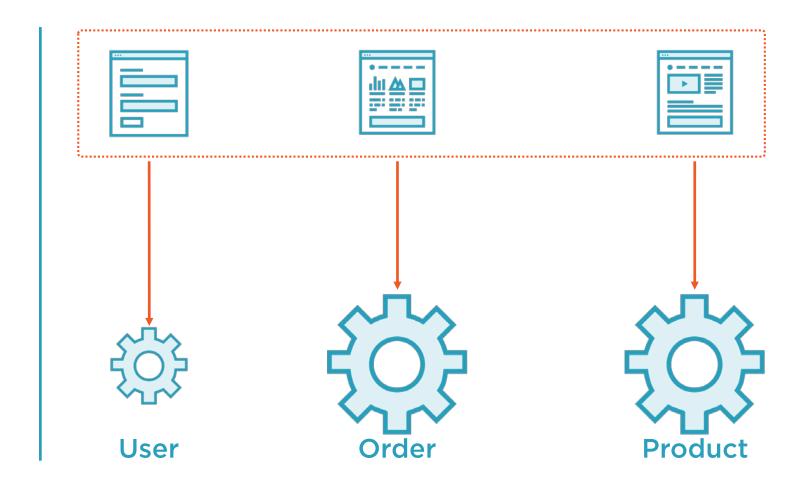
Hystrix, JRugged





Gateway

Access individual services





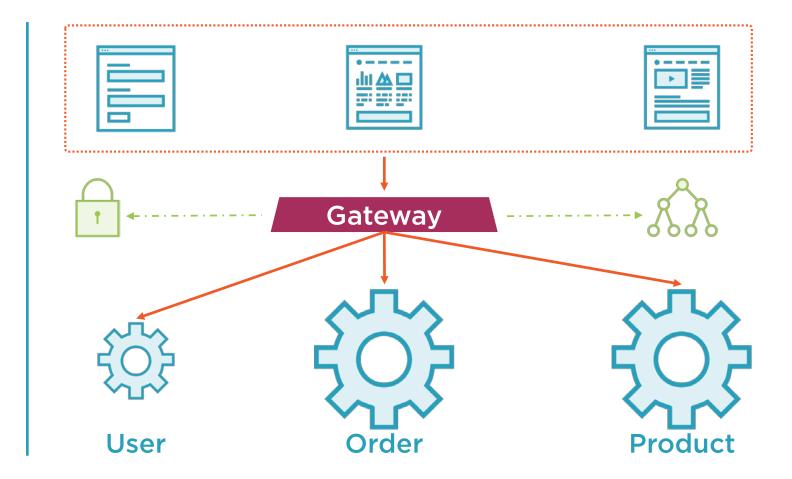
Gateway

Access individual services

Single entry point

Unified interface

Cross-cutting concerns





Gateway

Access individual services

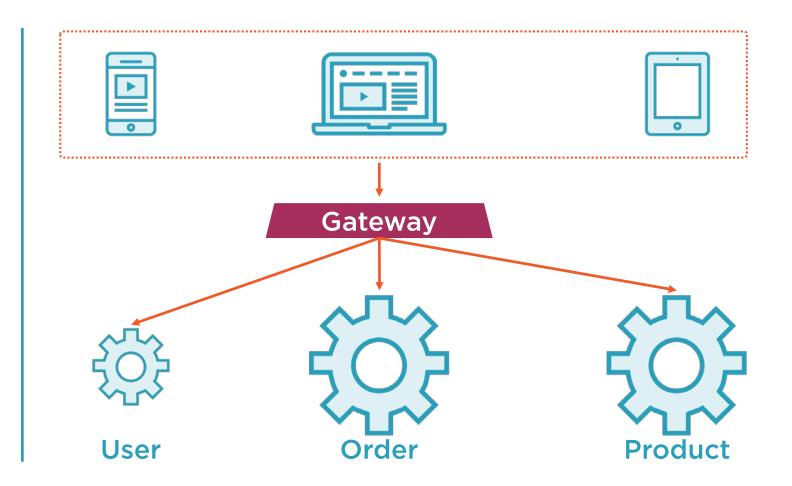
Single entry point

Unified interface

Cross-cutting concerns

API translation

Zuul, Netty, Finagle





Security



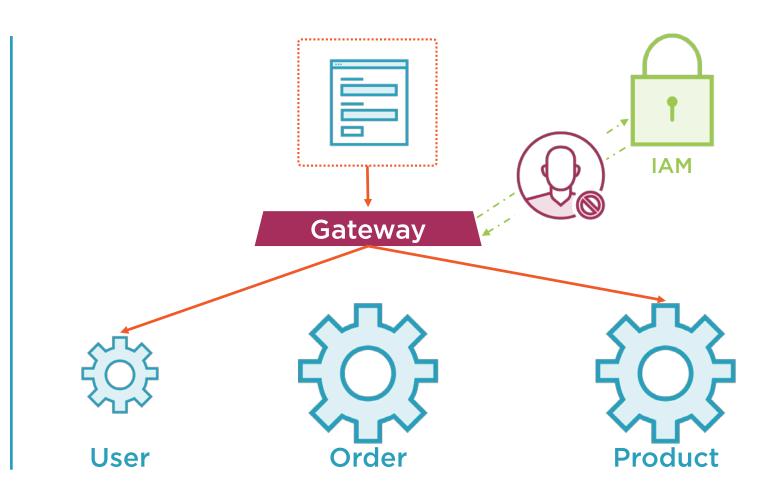
Authorization and Authentication

Identity and Access Management

Single Sign-on

Kerberos, OpenID, OAuth 2.0, SAML

Okta, Keycloak, Shiro





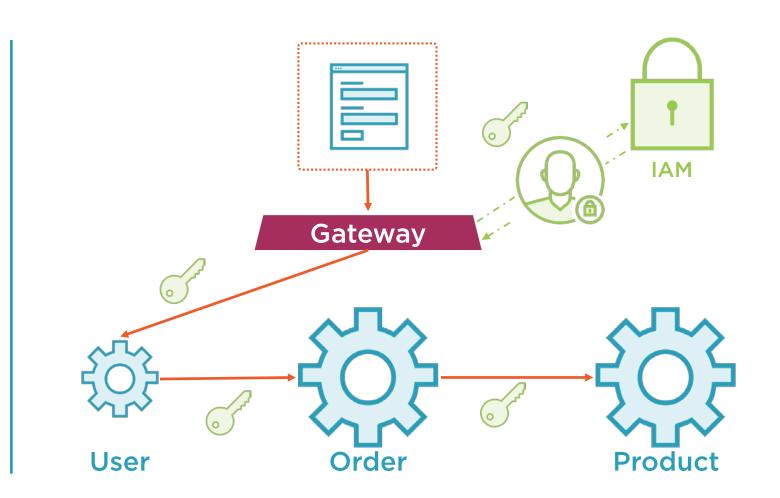
Access Token

Stores information about user

Exchanged between services

JSON Web Token

Cookie





Scalability and Availability



Scalability

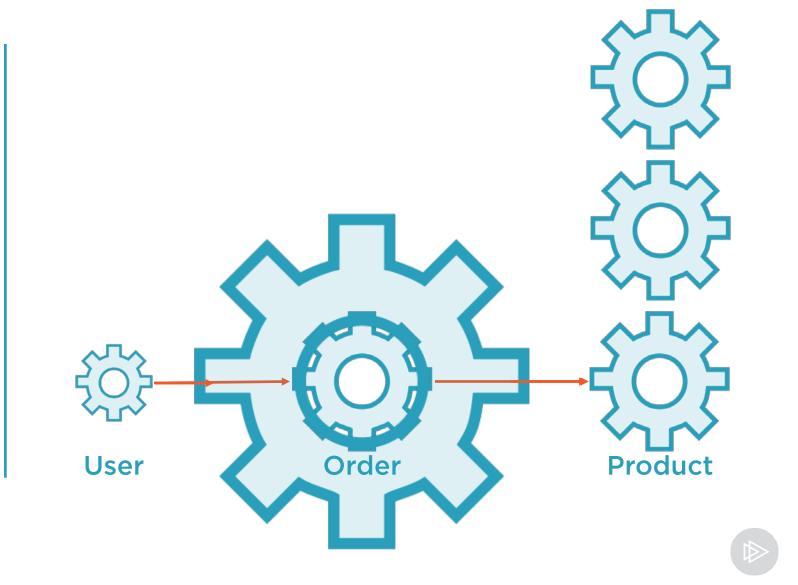
Vertical

More CPU and RAM

Horizontal

More machines
Service replication
Clustering

Scale up and down



Client Load Balancing

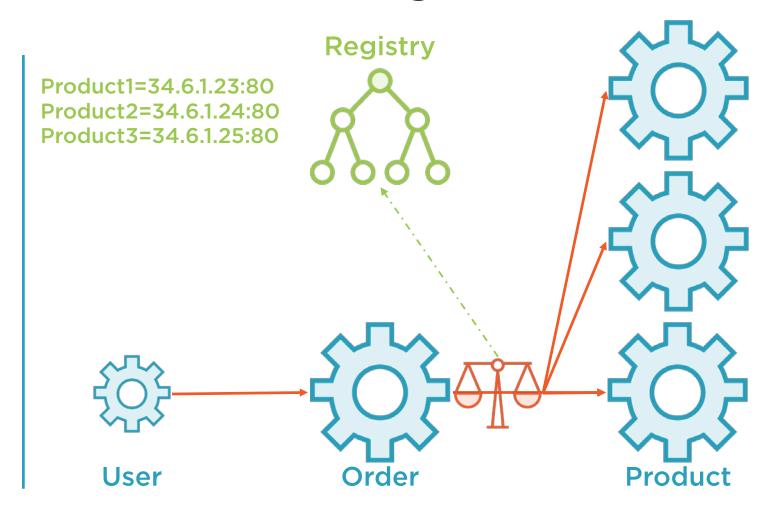
Several instances

Which instance to choose

Registry

Round-robin, weight, capacity

Ribbon, Meraki





Availability

Be operational

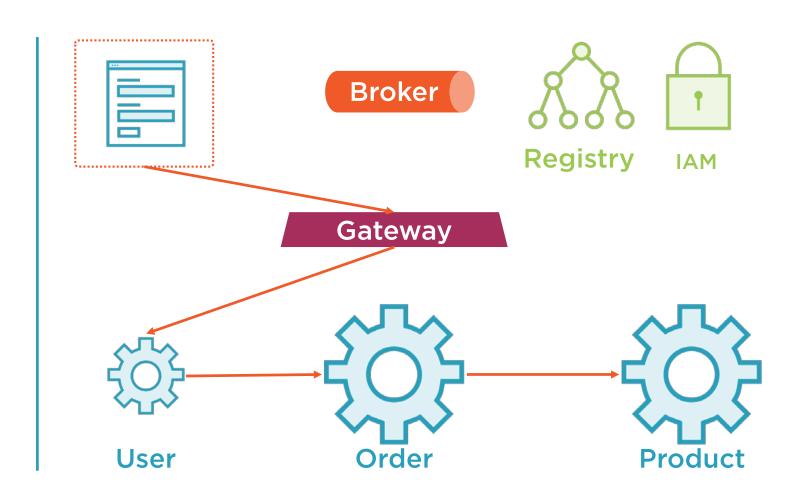
Single Point of Failure

Gateway

Broker

Registry

IAM





Availability

Be operational

Single Point of Failure

Gateway

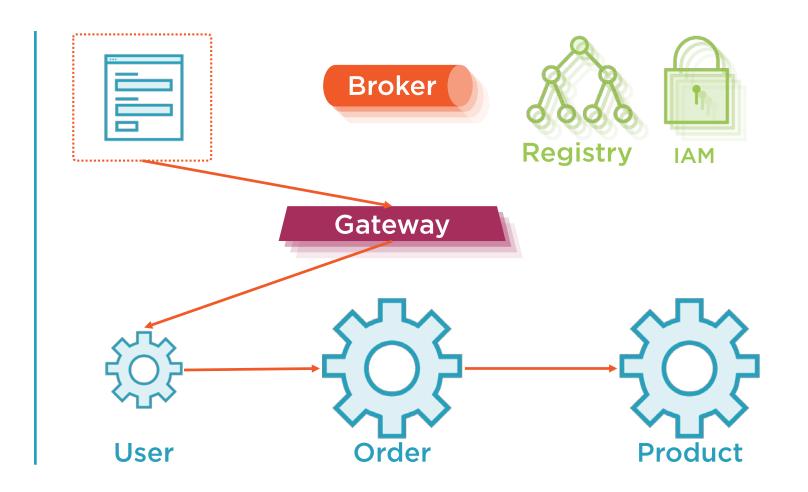
Broker

Registry

IAM

Multiply instances

Sync





Monitoring



Monitoring and Dashboard

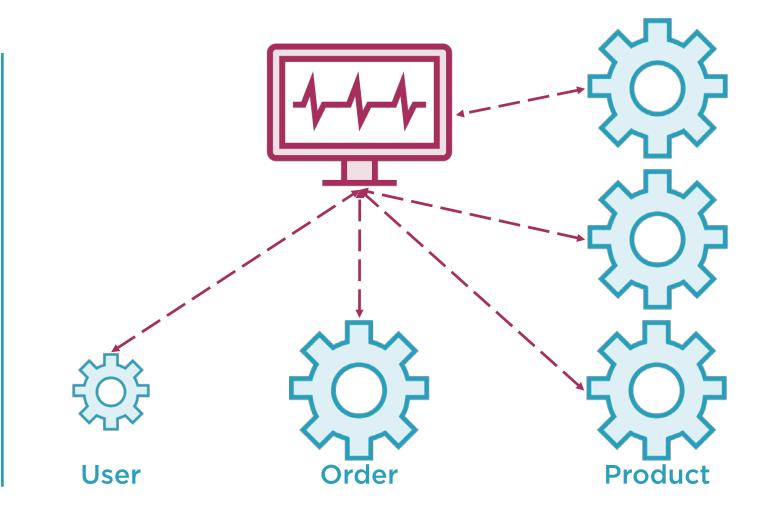
Many moving parts

Many machines

Centralized

Visual

Kibana, Grafana, Splunk





Health Check

Service running

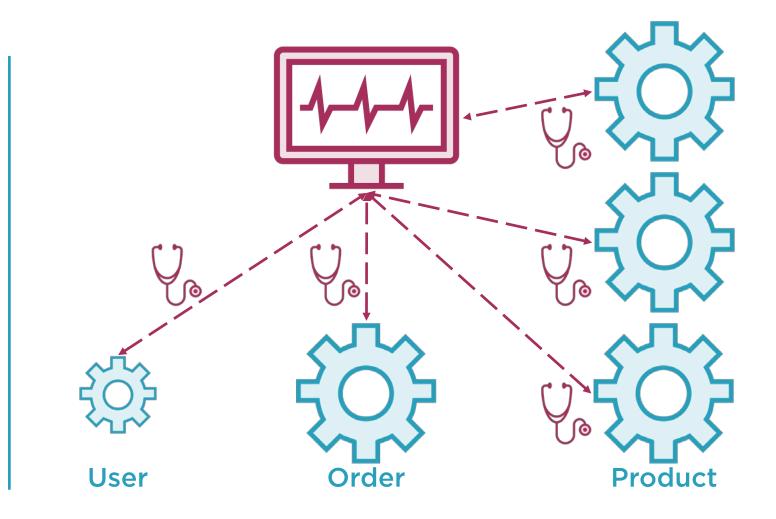
Incapable handling requests

Health check API

Database status
Host status

...

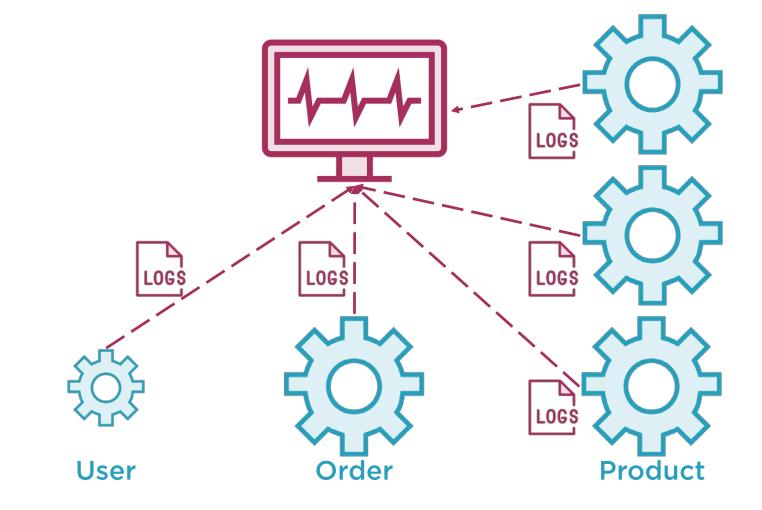
Heart bits





Log Aggregation

Understand behavior
Write logs
Read each log file
Aggregate logs
LogStash, Splunk,
PaperTrail





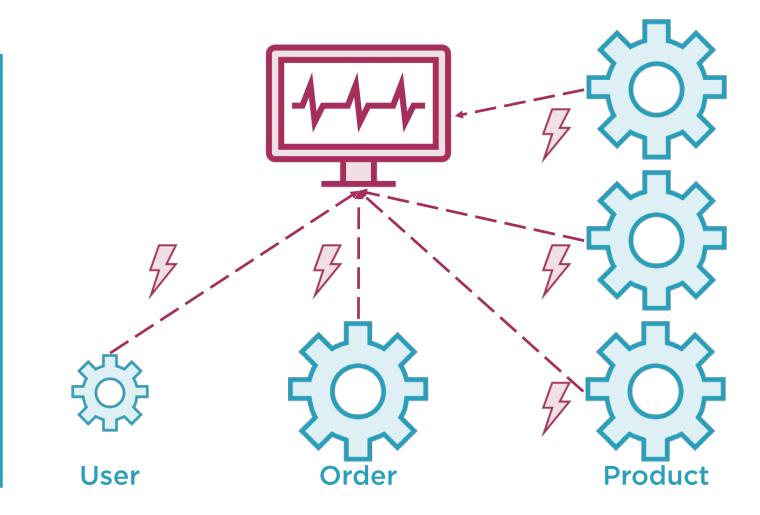
Exception Tracking

Errors

Throw an exception

Record exceptions

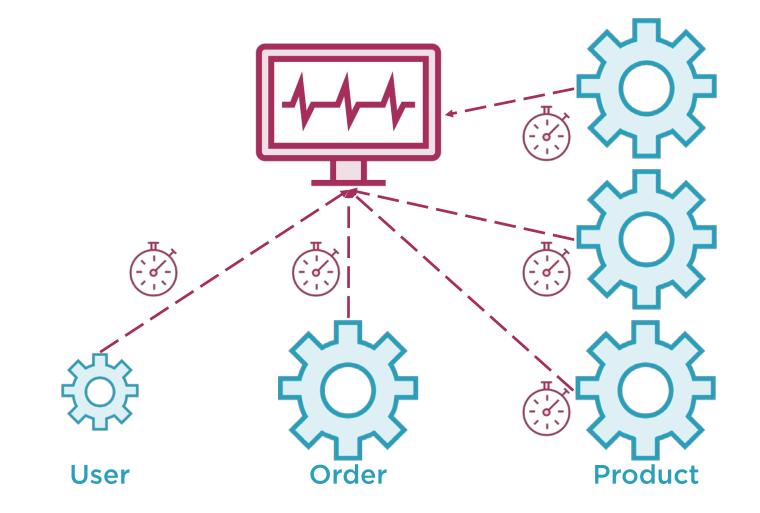
Investigated and resolved





Metrics

System slowing down
Performance issues
Gather statistics
Aggregate metrics
DropWizard,
Actuator, Prometheus





Auditing

Behavior of users

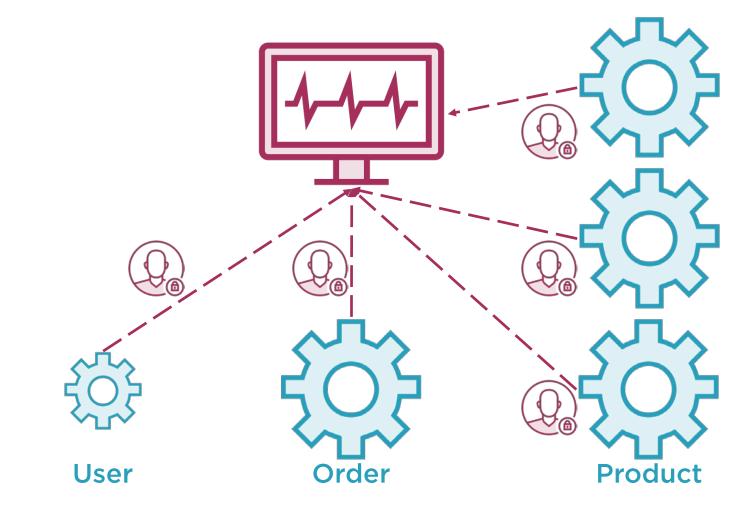
Login

Logout

Visited pages

Browsed products

Record user activity





Rate Limiting

Third-party access

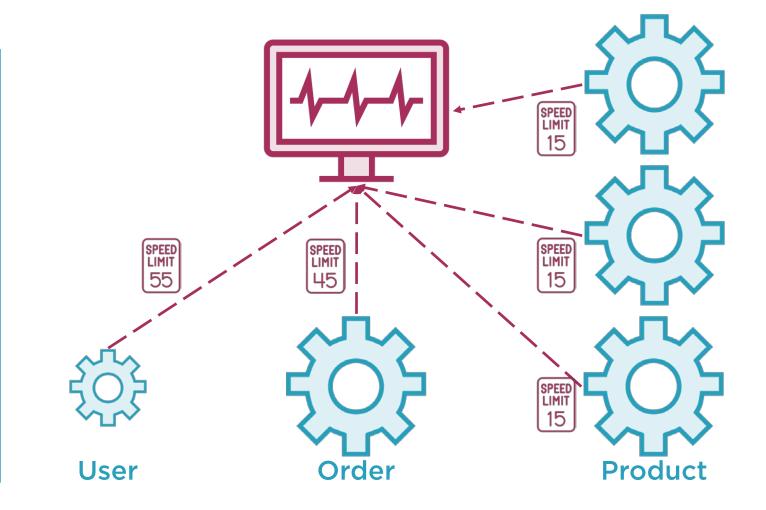
Control API usage

Defend DoS attacks

Limit traffic

In a period of time

Monetize our APIs





Alerting

Tons of information

How to be proactive

Fix error when occurs

Configure threshold

Trigger alerts

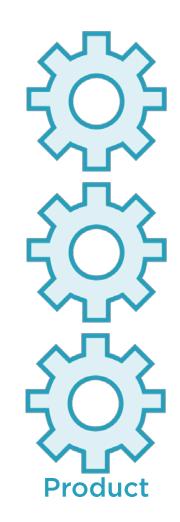






User







Distributed Tracing

Requests span services

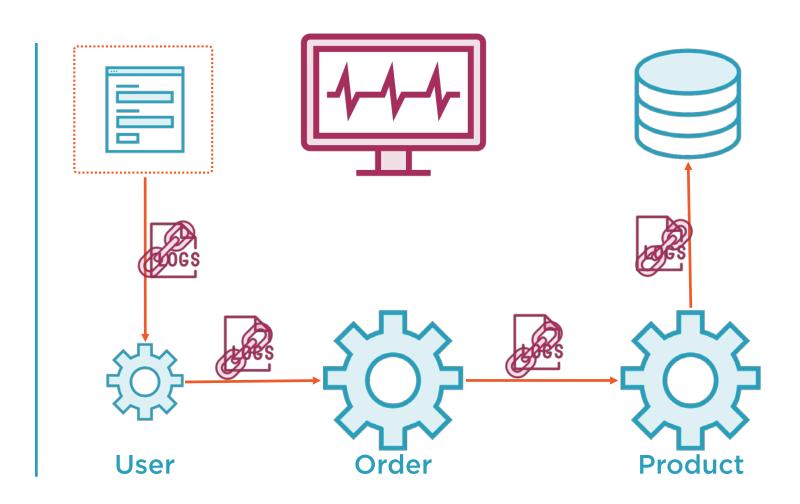
Logs

Trace entire request

Correlation id

Chain of calls

Dapper, HTrace, Zipkin





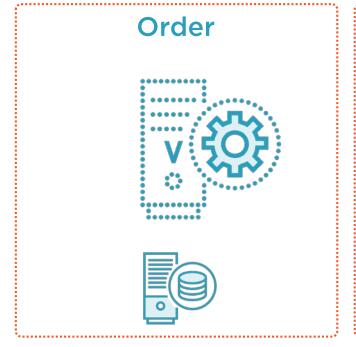
Deployment



Host

Physical server
Virtual server
On-premise
In the cloud









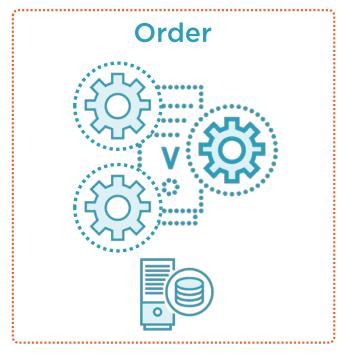
Multiple Services per Host

Microservice per host

Several services per host

Several services per virtual host





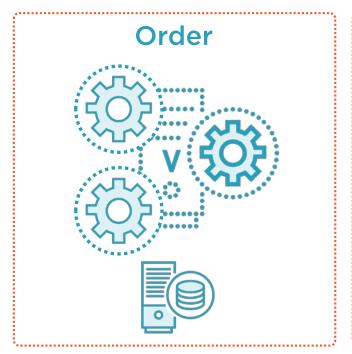




Containers

Packaging microservice With dependencies









Containers

Packaging microservice

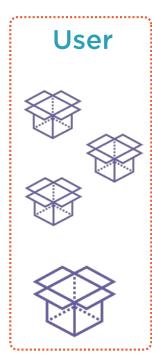
With dependencies

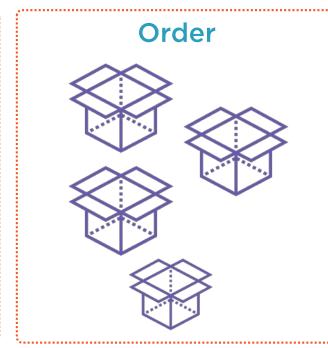
Container image

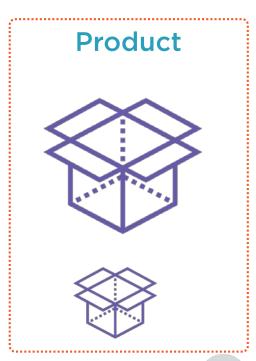
Easy to move from environment

Scale up and down

Docker, rkt









Orchestrator

Multiple containers

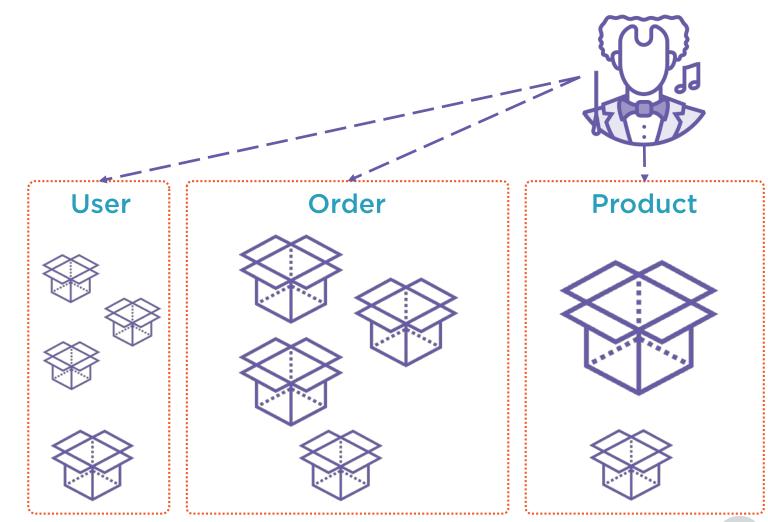
Multiple machines

Start at the right time

Failed containers

Kubernetes, Mesos,

Docker Swarm





Continuous Delivery

Automate deployment

Cost-effective Quick Reliable

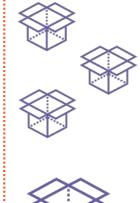
Build, test, deploy

Jenkins, Asgard, Aminator

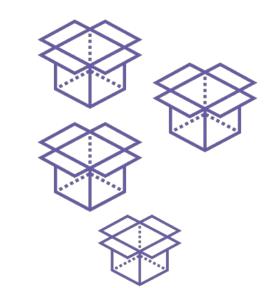




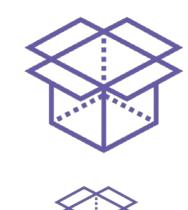
















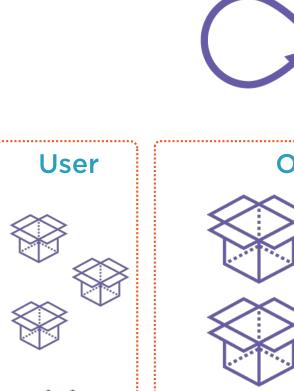
Environments

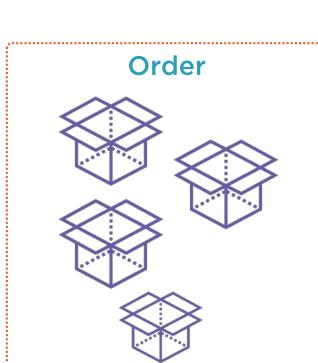
Production

Dev, test, QA, staging

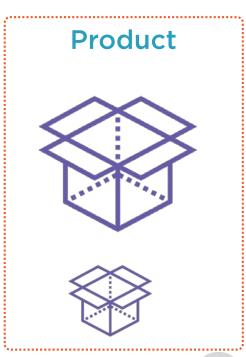
Integration

Versionning











External Configuration

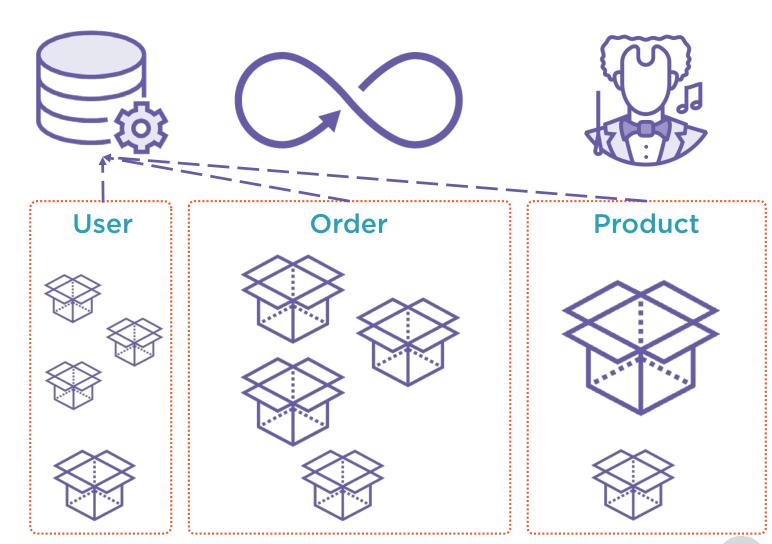
Different environments

Different configuration

Activate functionality

Externalize configuration

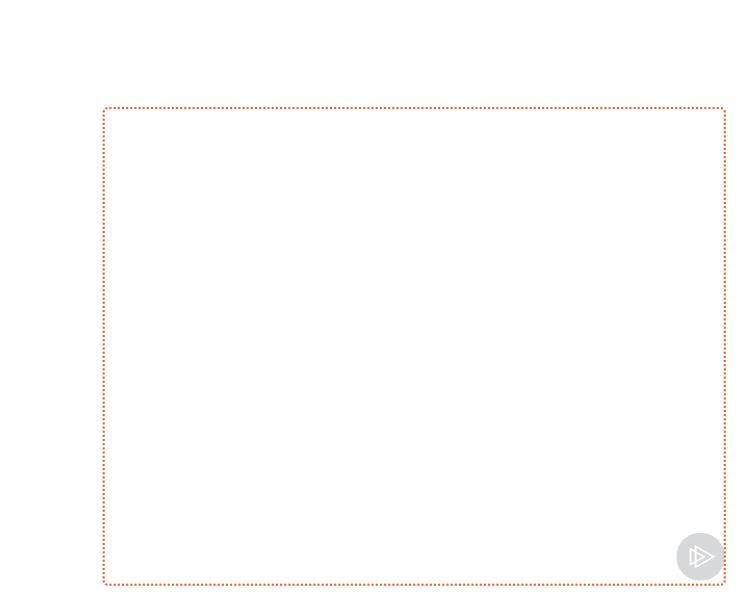
Archaius, Consul, Decider

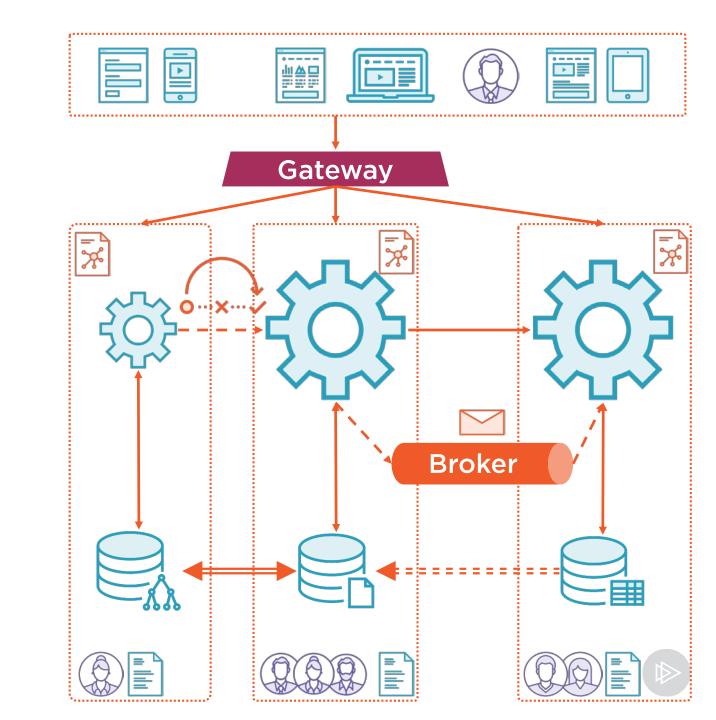


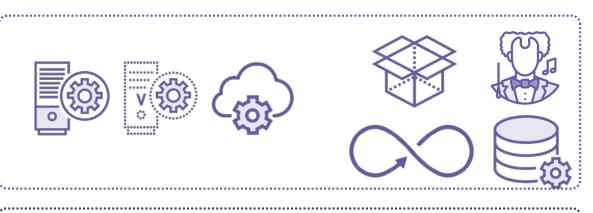


Revisiting the Microservice Elements















Firewall

Switch

Router













````



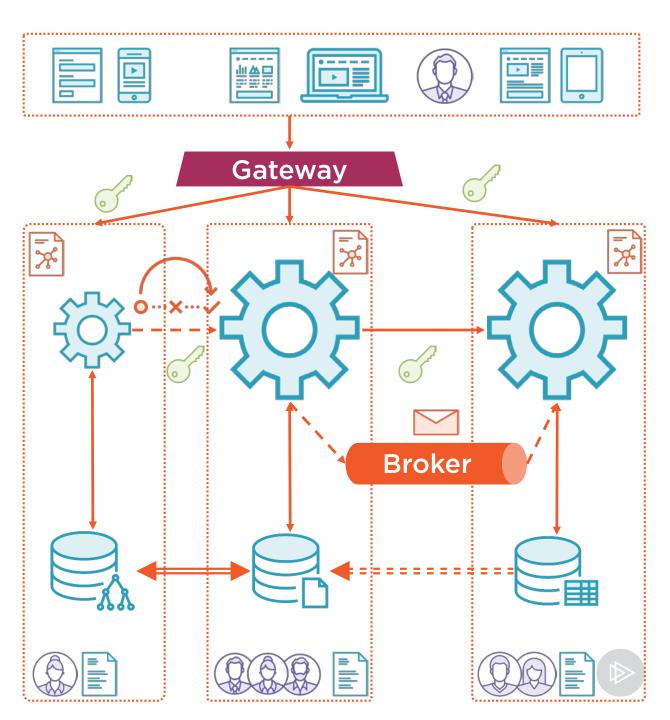


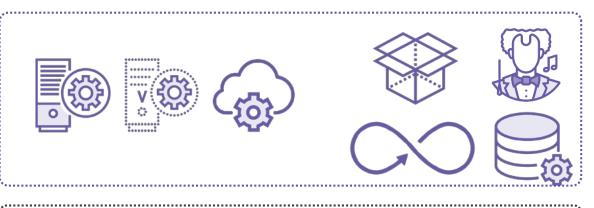


















**Firewall** 

Switch

Router

















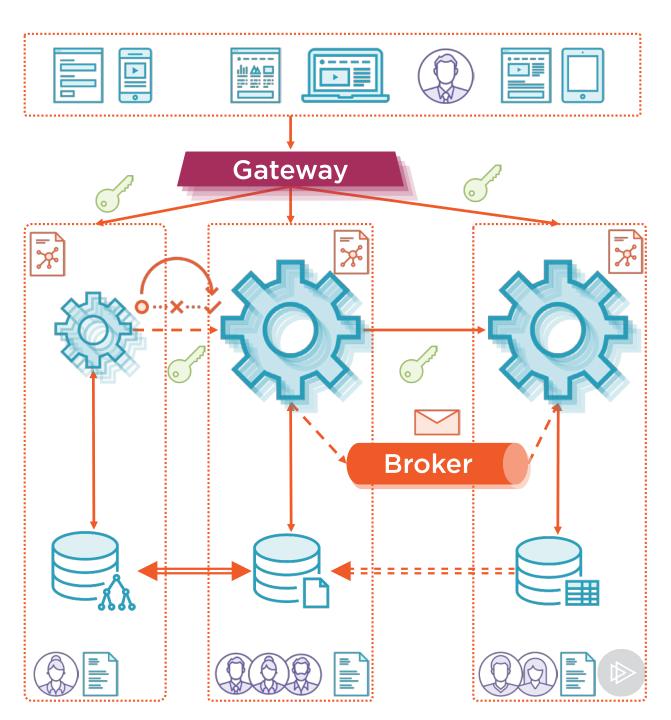


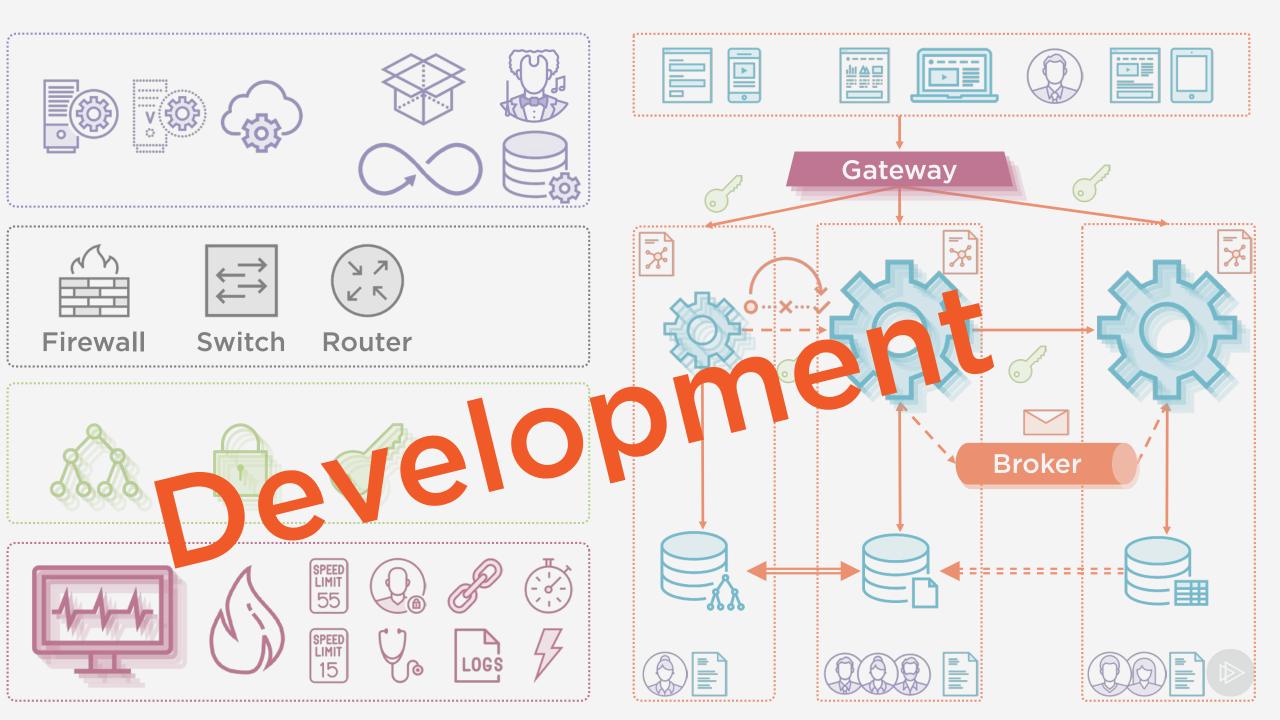


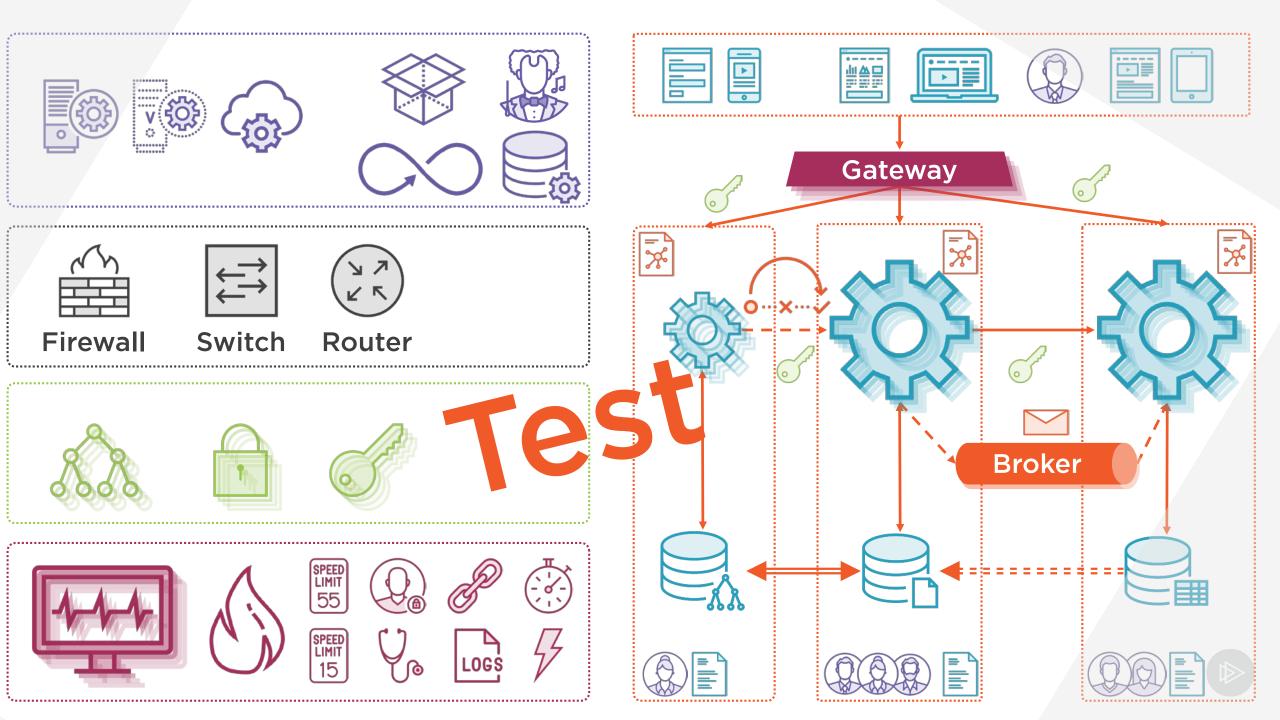


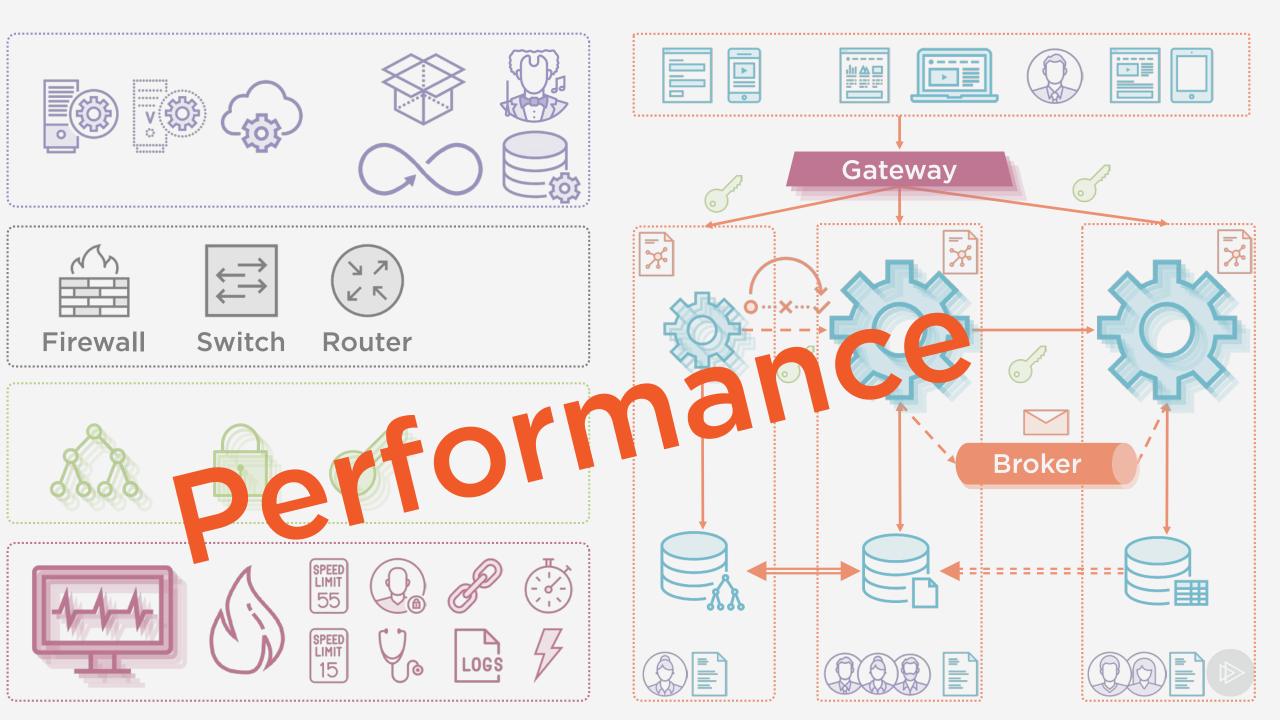


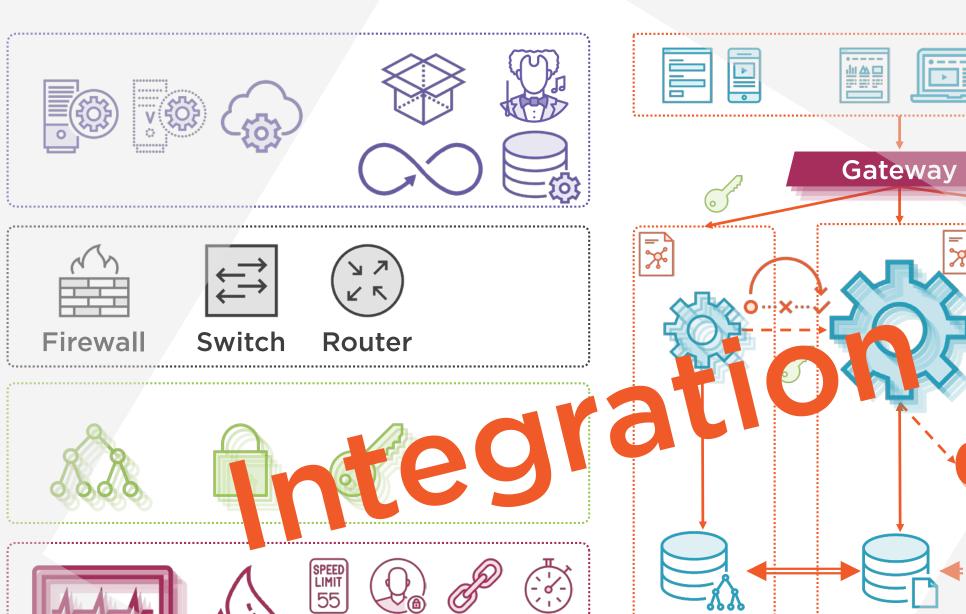






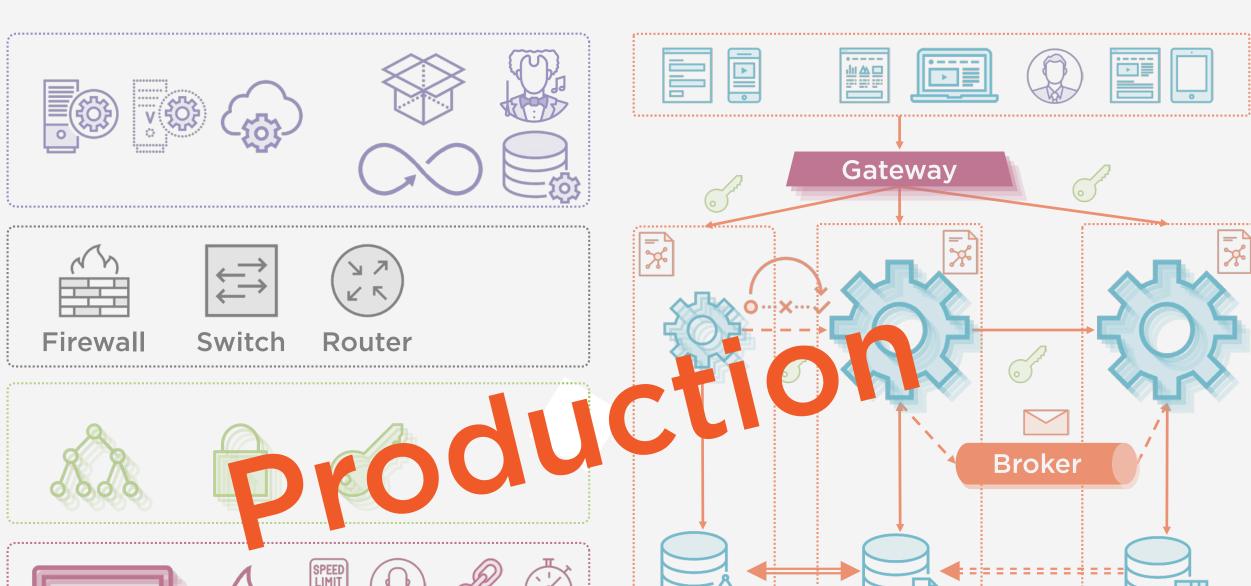


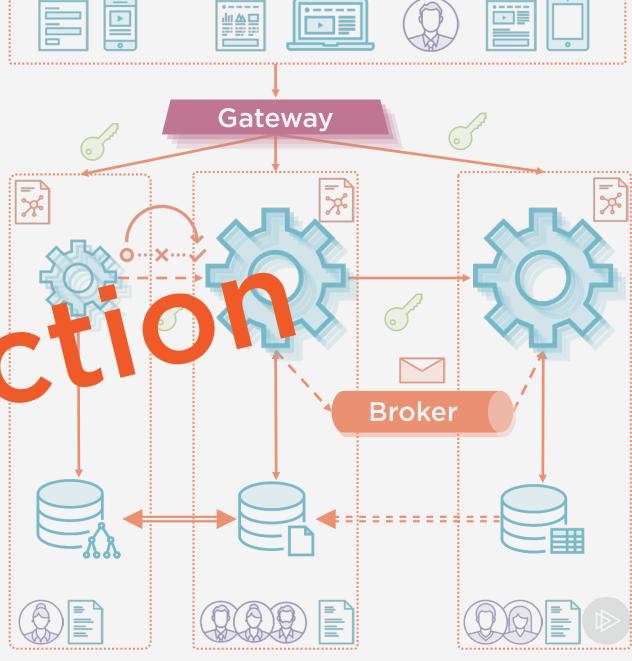




·

Broker





## Summary



Microservices terminology

Organize your team

Data storage

**User interface** 

**Distributed services** 

**Security** 

**Monitoring** 

**Deployment** 



#### Next Module



Do you need microservices?

**Challenges** 

Dos and don'ts

**Business** 

**Technical** 

**Deployment** 

