



Microservices with Spring Cloud

Kasim Sert
2016



Agenda

- The Definitions
- Monolith vs Micro-Service
- Microservices Features
- Why HTTP,REST,JSON ?
- Case Study
- Demos
- Conclusion

THE LIFE OF A SOFTWARE
ENGINEER.

CLEAN SLATE. SOLID
FOUNDATIONS. THIS TIME
I WILL BUILD THINGS THE
RIGHT WAY.



MUCH LATER...

OH MY. I'VE
DONE IT AGAIN,
HAVEN'T I?





What are Microservices ?

- Decomposition of single system into a suite of small services each running as independent processes and intercommunicating via well known protocols



What are Microservices ?

- Decomposition of single system into a suite of small services each running as independent processes and intercommunicating via well known protocols
- Developing a single application as a suite of small services each running its own process and communicating with lightweight mechanisms often an HTTP resource API, Martin Fowler



What are Microservices ?

- Decomposition of single system into a suite of small services each running as independent processes and intercommunicating via well known protocols
- Developing a single application as a suite of small services each running its own process and communicating with lightweight mechanisms often an HTTP resource API, Martin Fowler
- **Fine-Grained SOA, Adrian Cockcroft-Netflix**



Microservices Features

- Composing a single application using small services
 - **rather than a single monolithic application**
- each running as independent processes
 - **not just modules in a single executable**
- intercommunicating via open protocols
 - **HTTP/Rest, or Messaging**
- Separately written deployed scaled and maintained
 - **potentially in different languages**
- Services are independently replaceable and upgradable



Microservices Are Not

- SOA
 - SOA is about integrating enterprise applications, Microservices are about decomposing single applications
- A Silver Bullet
 - Microservices approach involves drawbacks and risks
- New!



Microservices Are Trending

- Twitter moved from Ruby/Rails monolith to Microservices
- Facebook moved from PHP monolith to Microservices
- Netflix moved from Java monolith to Microservices



Benefits of a Microservice Approach

- Extreme Scalability
- Organisational Ownership
- Cheaper to Start
- Ideal for the Cloud
- Grasp latest technologies
- Promotes Agile Practices
- Promotes DevOps
- “It’s what the cool people do!”



Benefits of a Monolith

- Common code infrastructure
- More density/efficient allocation of hardware
- Less inter-process communication required, less network hops
- Easy to scale(up to some level with load balancer)
- Easy to test as a single unit(up to some limit)



Monolith vs Microservices

		<u>Monolithic</u>	<u>Microservices</u>
Dev	Codebase	Singular, unified, codependent	Smaller, independently evolving
	Technology Stack	Homogeneous	Best of breed
	Edit-Compile-Test	Minutes to hours	Seconds
	Dependencies	Compile-time; version lockstep	Network APIs; library independence
	Change	Slow, painful, risky	Isolated; consumer-driven contracts
	Rewrite	Major planning	One service at a time
Prod	Test & Release	Months, all in one	Focus on changed services
	Deploy & Start	Minutes to hours	Seconds or less
	Scaling	All or nothing	Hotspots
	Operations	Static, straightforward	Dynamic, complex automation



HTTP

- Available verbs GET, POST, PUT, DELETE (and more)
- Mechanisms for
 - caching and cache control
 - content negotiation
 - session management
 - user agent and server identification
- Status codes in response (200, 404, etc) for information, success, redirection, client error, server error
- Rich standardised interface for interacting over the net




JSON

- Minimal and popular data representation format
- Schemaless in principle, but can be validated if needed

```
▼ {  
  "description": "Spring Cloud Eureka Discovery Client",  
  "status": "UP",  
  ▶ "discoveryComposite": { ... }, // 4 items  
  ▶ "diskSpace": { ... }, // 4 items  
  ▼ "db": {  
    "status": "UP",  
    "database": "HSQL Database Engine",  
    "hello": 1  
  },  
  ▼ "refreshScope": {  
    "status": "UP"  
  },  
  ▼ "hystrix": {  
    "status": "UP"  
  }  
}
```

REST

- Uniform Interface By;
 - Use of known HTTP verbs for manipulating resources
 - Resource manipulation through representations which separated from internal representations
 - Hypermedia as the engine of application state (HATEOAS): Response contains all allowed operations and the resource identifiers needed to trigger them



```

{
  "number" : 12345,
  "balance" : -20.00,
  "currency" : "EUR",
  "links" : [ {
    "rel" : "self",
    "href" : "https://bank.com/account/12345"
  }, {
    "rel" : "deposit",
    "href" : "https://bank.com/account/12345/deposit"
  } ]
}

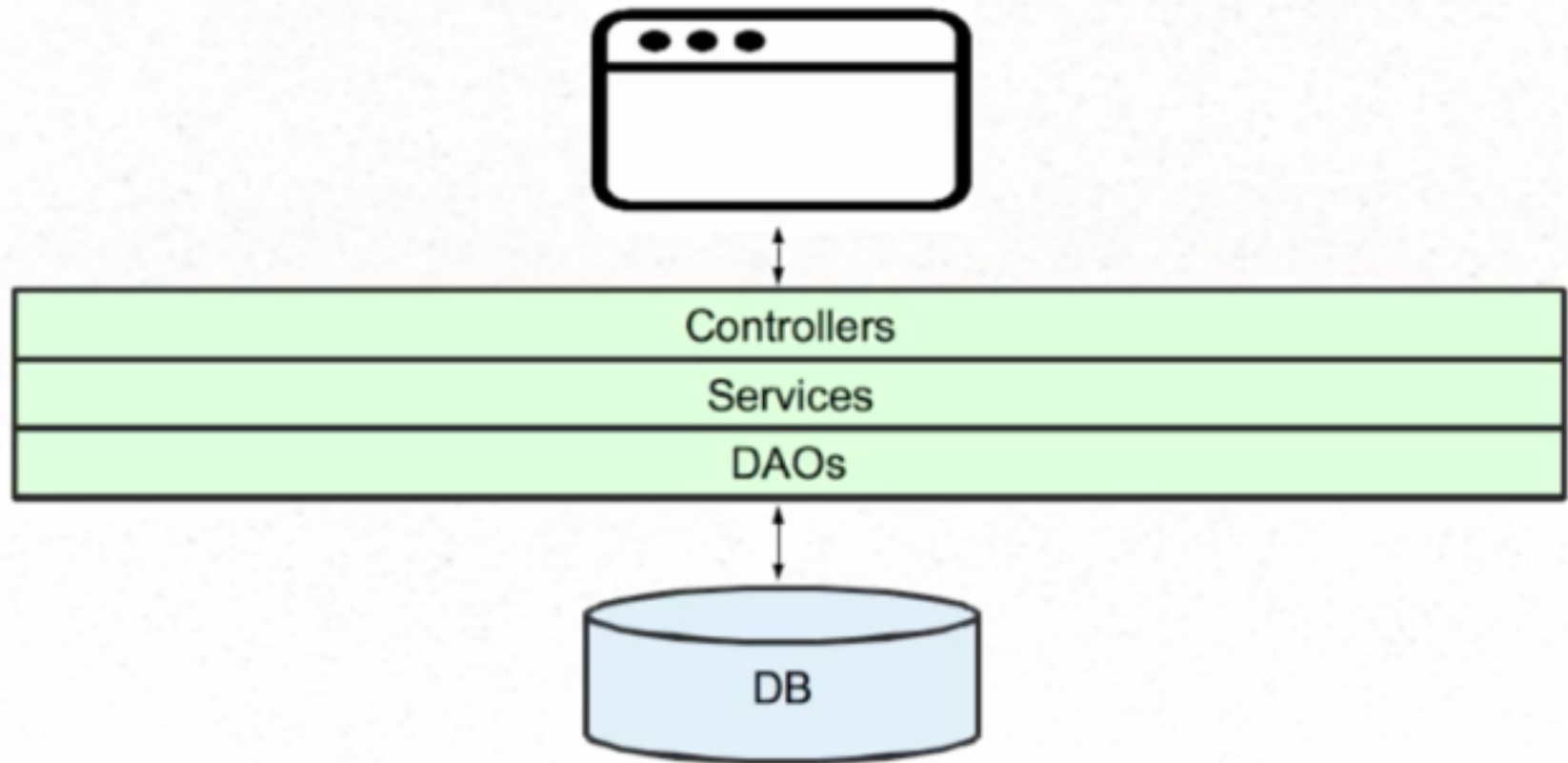
```



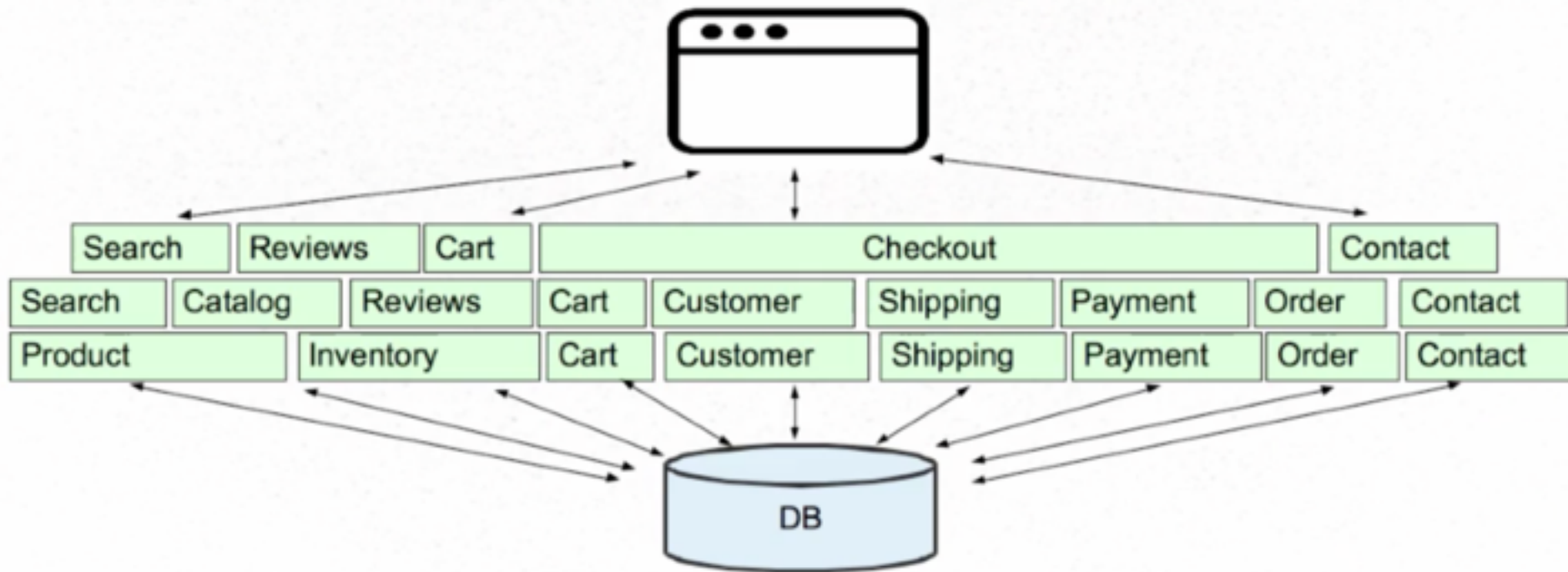
Case Study

- Consider Monolithic Online Shopping Application
 - Web/Mobile Interfaces
 - Functions for:
 - Searching for products
 - Product catalog
 - Inventory management
 - Shopping cart
 - Checkout
 - Order Fulfilment
 - How would this look with microservices?

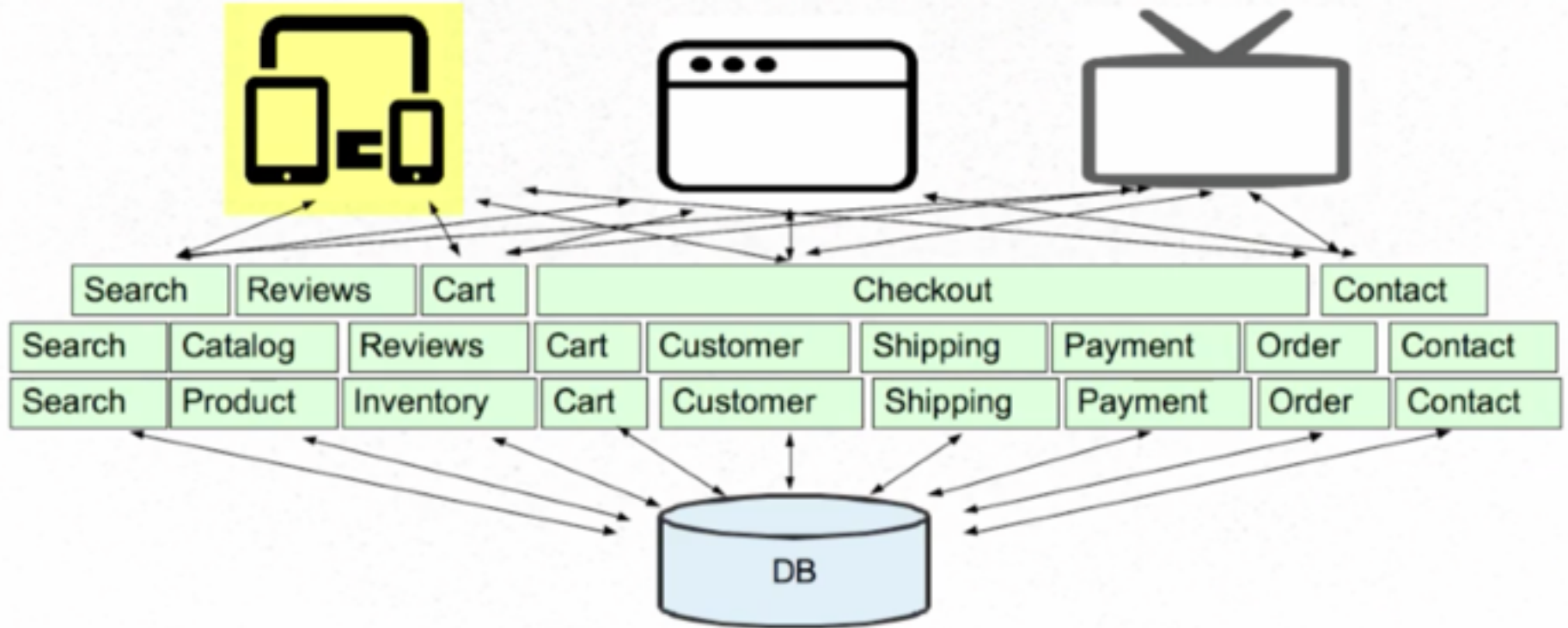
Basic Diagram



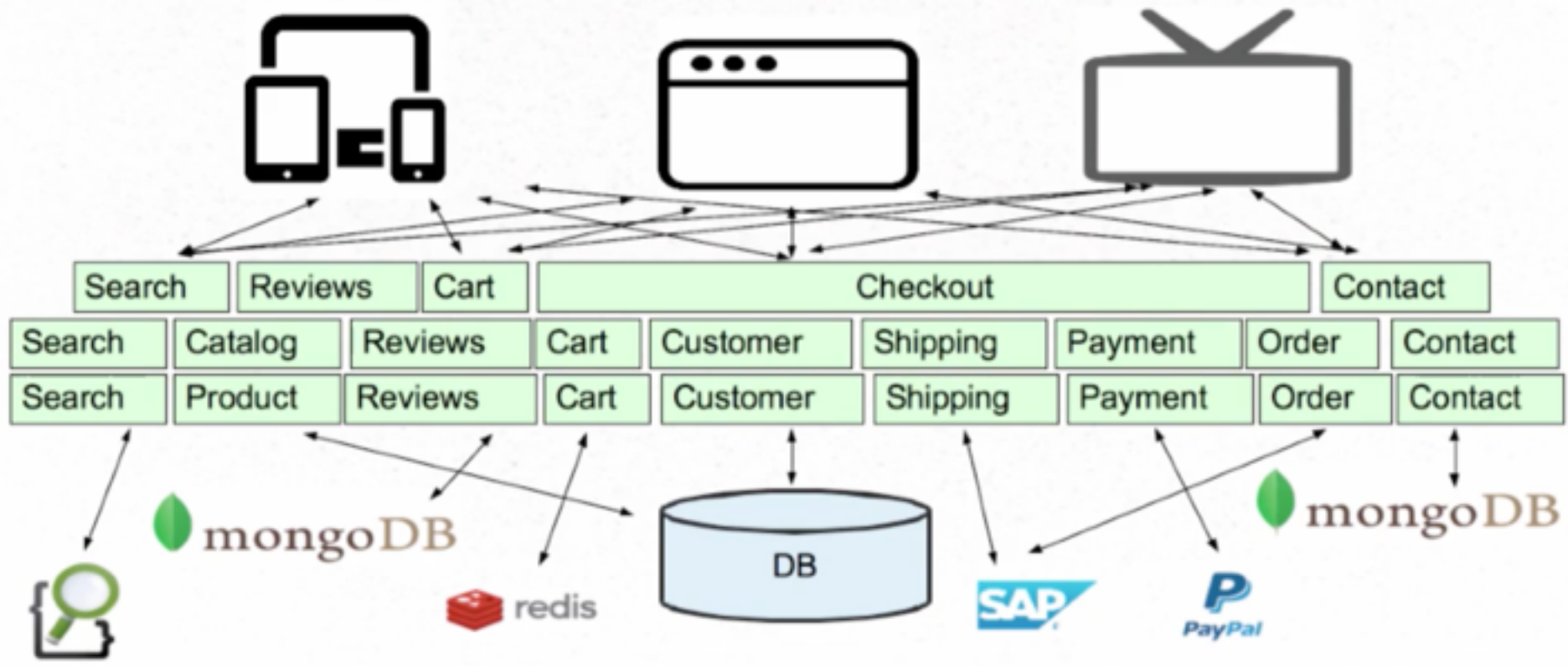
Understanding Monolith Application



New Types of Client



New Types of Persistence/ Services



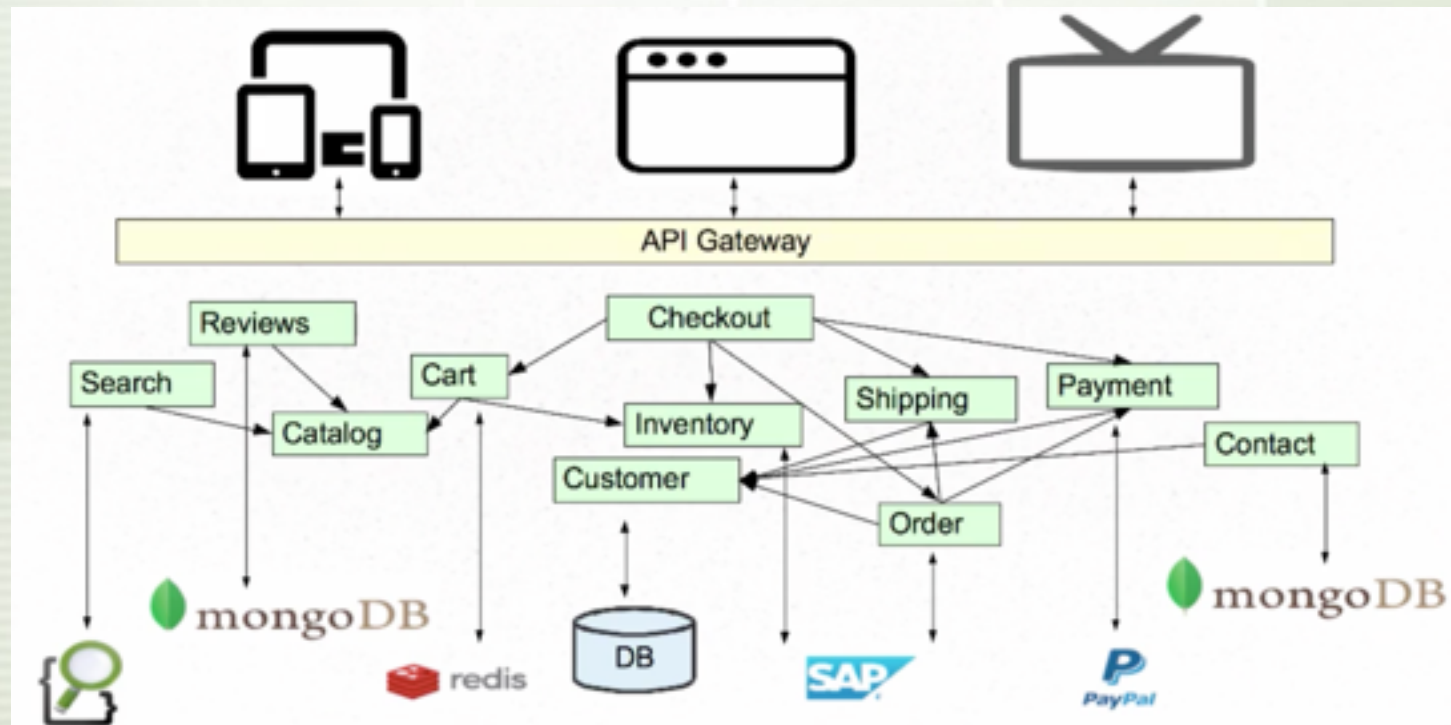
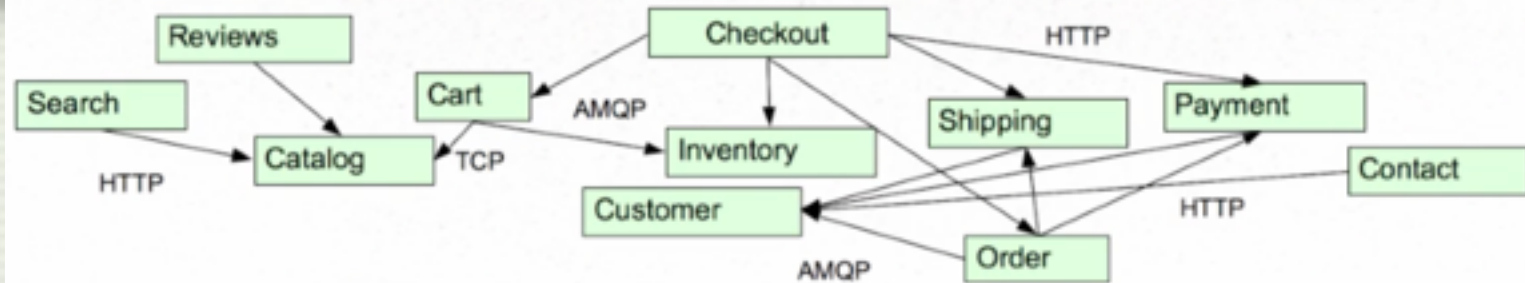


Monolith Challenges

- Language/Framework Lock
- Digestion
 - Single developer can not digest
 - Single team can not manage a large application(amazon's 2 pizza rule)
- Deployment as single unit
 - Can not independently deploy single change (risky)
 - Ready changes should be wait(next deploy)



Case Study with Microservices





Microservices Advantages

- Encapsulate Business capabilities
- Not dependent on technology stack
- Easy to digest each service
- Easy to test, deploy manage, version and scale each service.
- Changes can be deployed as soon as ready



Microservices Disadvantages

- Complexity moved to operation layer
 - fallacies of distributed computing
- Services will be unavailable
 - Design for failure
 - Much more need for monitoring(Healthcheck?)
- Remote Calls more expensive than in-process calls
- Problem of Transactions (Eventual consistency over ACID)
- Features span multiple services (Hard to Integration Test)



Fallacies of Distributed Computing

- Network is reliable
- Latency is zero
- Bandwidth is infinite
- The Network is secure
- Topology does not change
- There is one administrator
- Transport cost is zero
- The network is homogenous



Spring Cloud Summary

- Spring cloud(Using Netflix OSS internally)
 - Apply common patterns needed in distributed/cloud applications
 - Distributed/Versioned/Centralised Config. Management
 - Service Registration/Discovery
 - Load Balancing
 - Circuit Brakers
 - Monitoring
 - Based on Spring Boot



Spring Cloud Summary

- Spring Boot
Stand-alone, production-grade Spring-based applications
- Spring Data REST / Spring HATEOAS
Spring-based applications following HATEOAS principles
- Spring Cloud Config
Centralised external configuration management, backed by Git
- Netflix Eureka
REST-based service discovery and registration for failover and load-balancing
- Netflix Ribbon
IPC library with built-in client-side software load-balancers
- Netflix Hystrix
Latency and fault tolerance for distributed system
- Netflix Hystrix Dashboard
Web-based UI for monitoring Hystrix



Netflix

- Reinvented itself in 2007
- Moved from DVD mailing to video-on-demand
 - Once USPS biggest customer
 - Now biggest source of Internet Traffic
- Became pioneer in Cloud computing
 - since all applications run on AWS
- Open sources its cloud projects with name Netflix Open Source Software

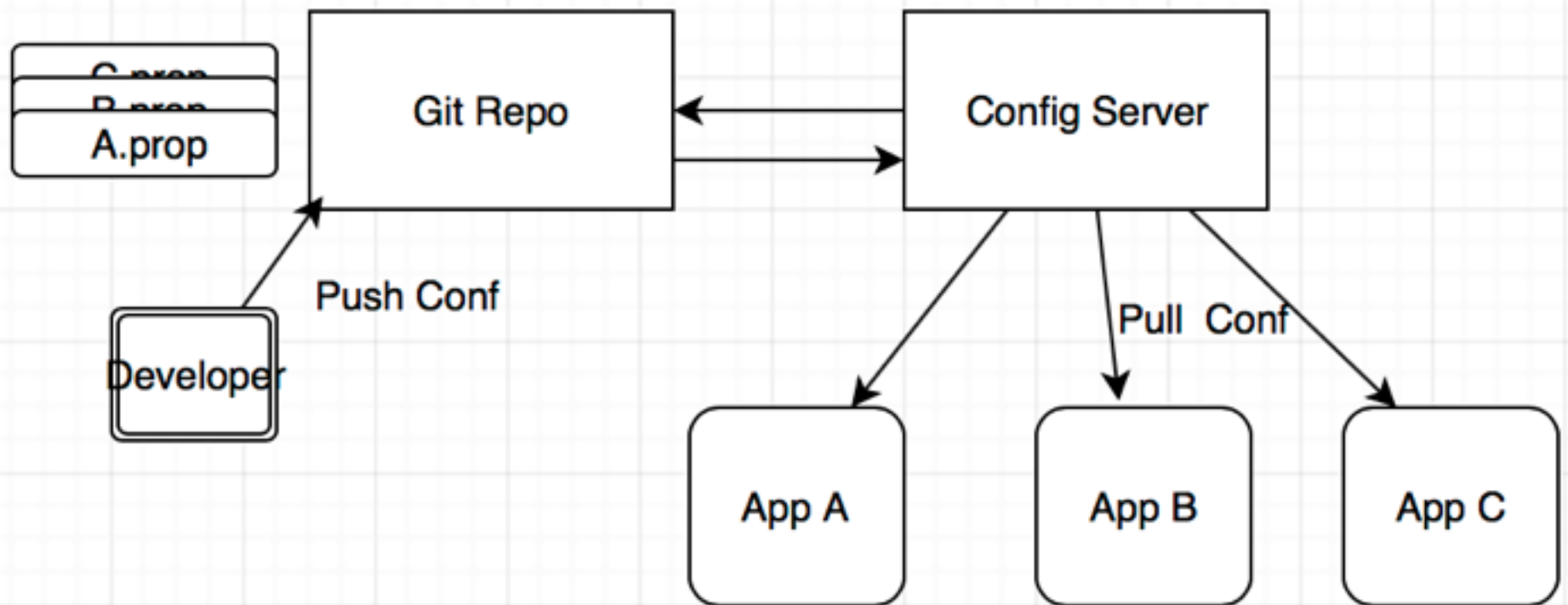


Conclusion

- Monoliths always bad?
- How micro is micro?
 - Small enough for one developer to digest
 - Predictable and easy to experiment
 - If necessary can be rewritten in different technology stack in two weeks

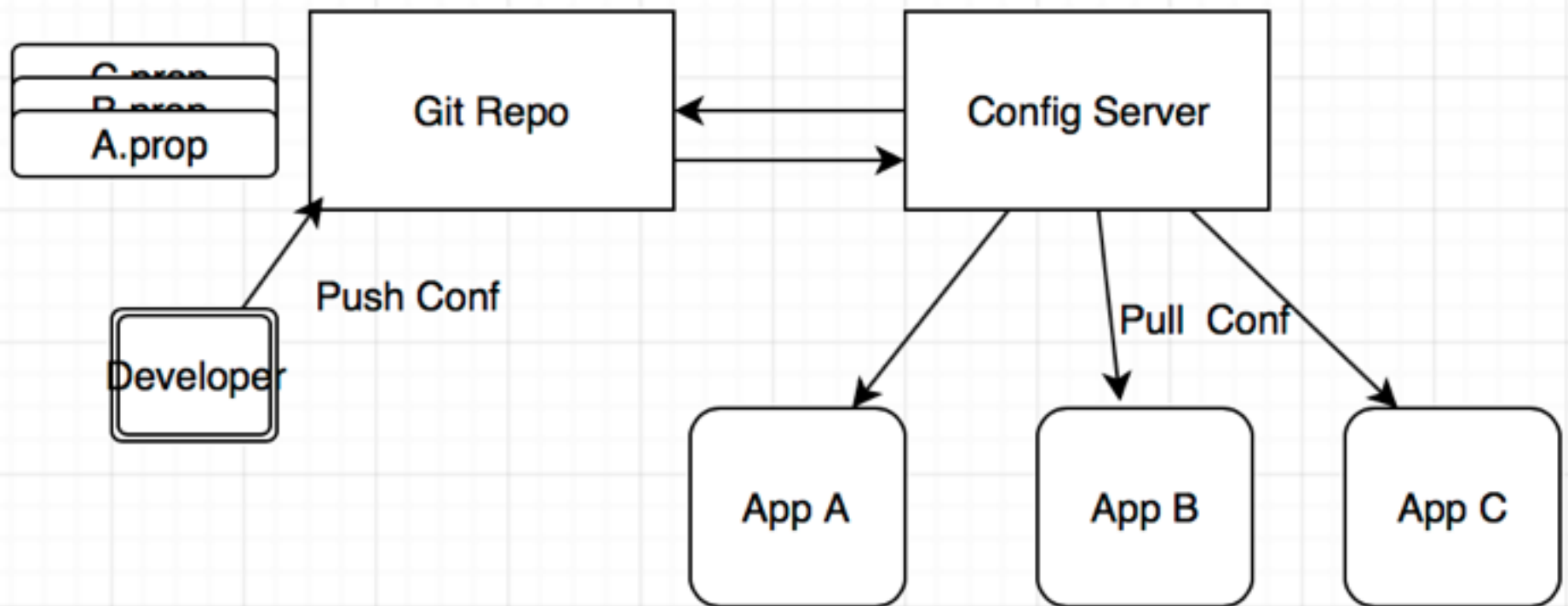


DEMO1-Spring Data REST

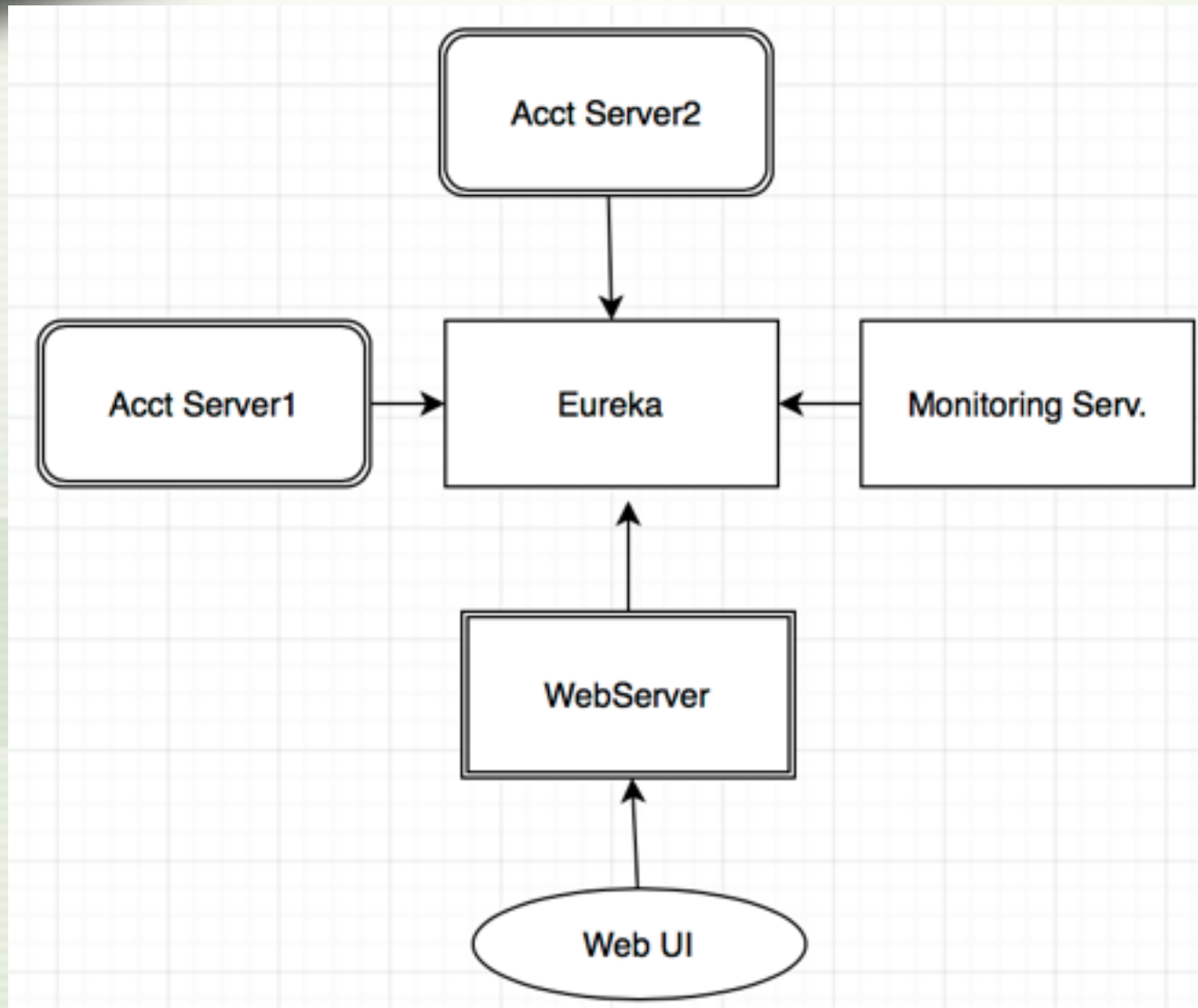




DEMO2-Spring Config Server



DEMO3-Spring Cloud





References

- <https://en.wikipedia.org/wiki/Microservices>
- <http://martinfowler.com/articles/microservices.html>
- <https://start.spring.io>
- <https://projects.spring.io/spring-cloud/>



Questions?

