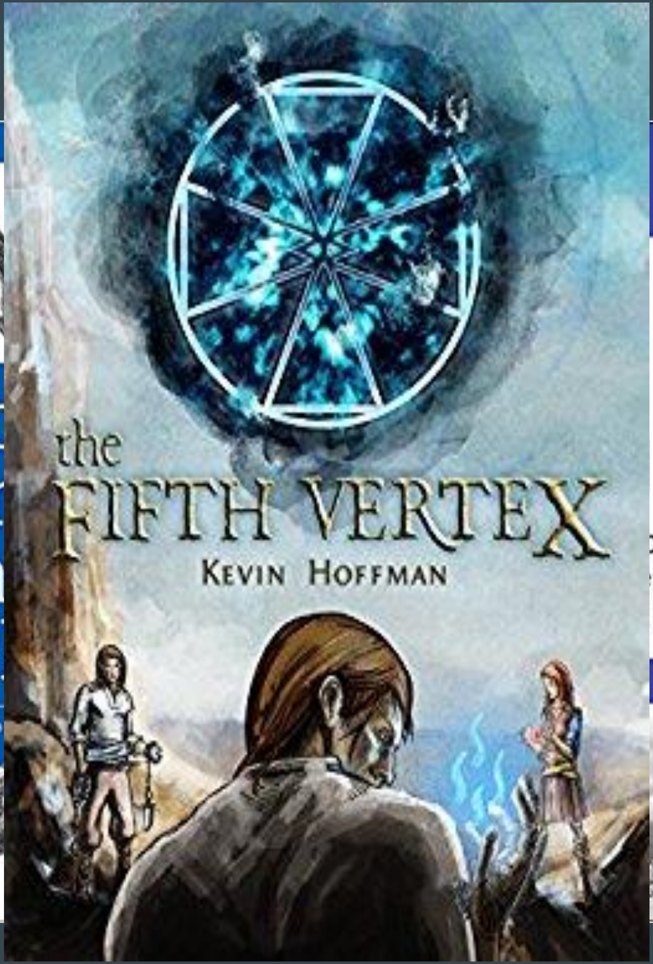
Reactive Microservices with .NET Core

Kevin Hoffman
Lead Engineer, Capital One
@KevinHoffman autodidaddict

About Me

O'REILLY
Buil
Micr
with
ASP.
DEVELOP, TEST, AND
IN THE CLOUD


the
FIFTH VERTEX
KEVIN HOFFMAN

SIGILORD CHRONICLES BOOK 2
the
BLOOD SIGIL
KEVIN HOFFMAN

Agenda

- What is a microservice?
- Distributed Transactions
- Complex Data Flow Modeling w/Microservices
- 100% Fully Buzzword Compliant Demo
  - Not a “hello world” sample
- Lessons Learned So Far
- Q&A

Is this a Microservice?

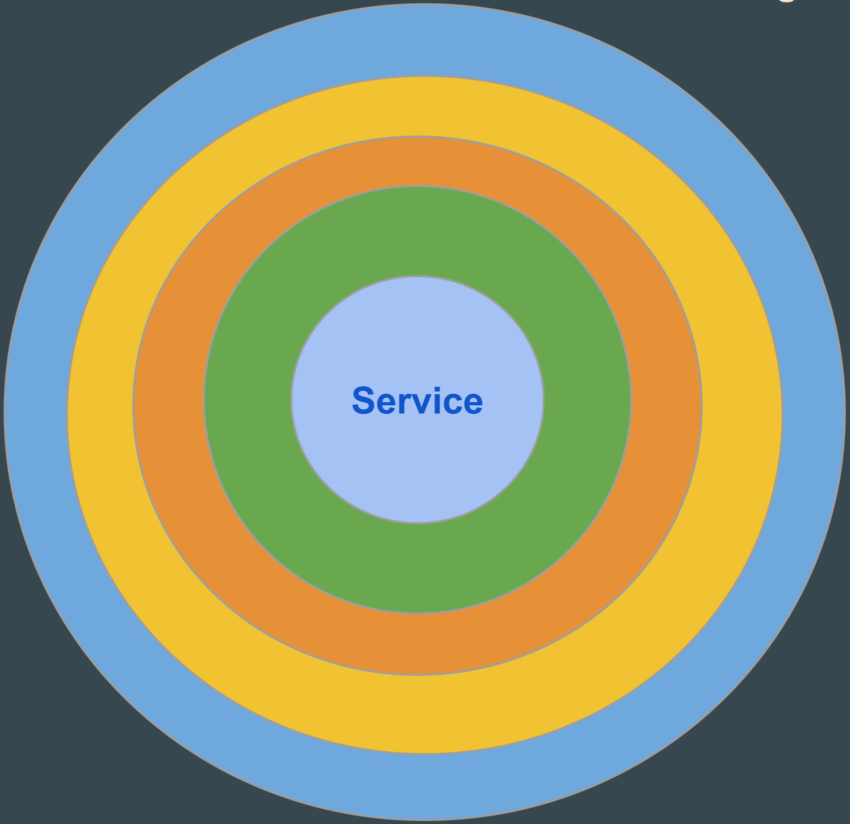
```
[Route("api/[contr      )]
public class Valu      oller : Controller
{
    // GET api    ues
    [HttpGet]
    public IE    rable<str       ()
    {
        retur    ew string[] { "          "value2" };
    }

    // PUT, PO    DELETE, etc
    ...
}
```

This is a protocol handler … a facade.

This is not your service.

Some microservices are RESTful …
Not all RESTful services are micro

The Microservices Onion



Storage

Metrics / Monitoring

Protocol

Discovery

Elastic Scaling

Tracing

0 Downtime Deploy

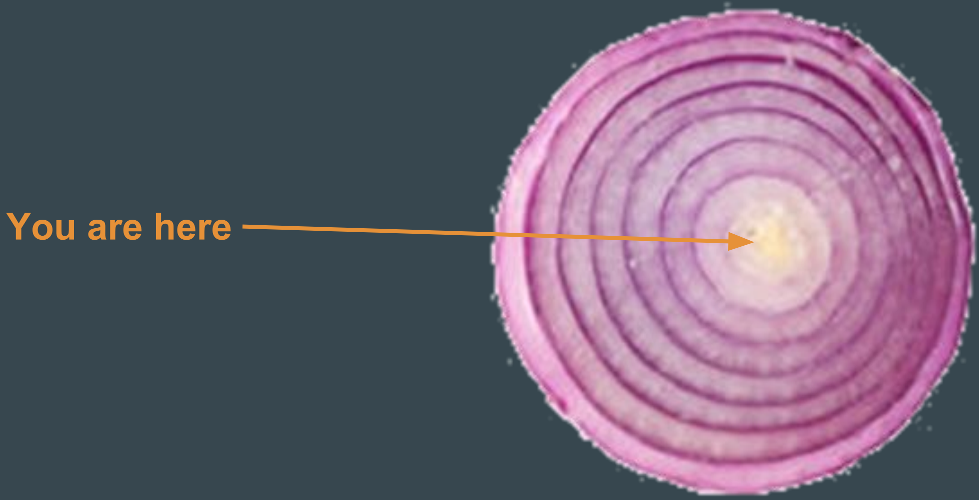
Security

Logging

Networking

What is a Microservice

A microservice is a discrete unit of functionality that adheres to the Single Responsibility Principle, asks nothing* of its host, and can be deployed without impacting other services in the ecosystem.



But...I must worry about the onion...right?

- Discovery - **Steeltoe** + Eureka, Consul, DNS
- Logging and Monitoring
  - Splunk, Sumologic, Grafana, App dynamics, Prometheus, Dynatrace, ad nauseum
- Security - OAuth, OIDC (DIY middleware, third party middleware, Azure)
- Protocol and Transport - HTTP, gRPC, Protobufs, JSON
- Configuration - Spring Config Server (via Steeltoe), etcd, env vars, ...
- 0 Downtime Deploy - Kubernetes, Cloud Foundry, GKE, etc
  - Containerize your app
- Tracing - **Steeltoe** + Zipkin, OpenTracing, Jaeger, ...
- Storage - Cloud platform, S3, ...
- Elastic Scaling - Kubernetes, EKS, Cloud Foundry, GKE, etc
  - Containerize your app
- Messaging - Rabbit, SQS, Azure, Kafka, PubNub, ...

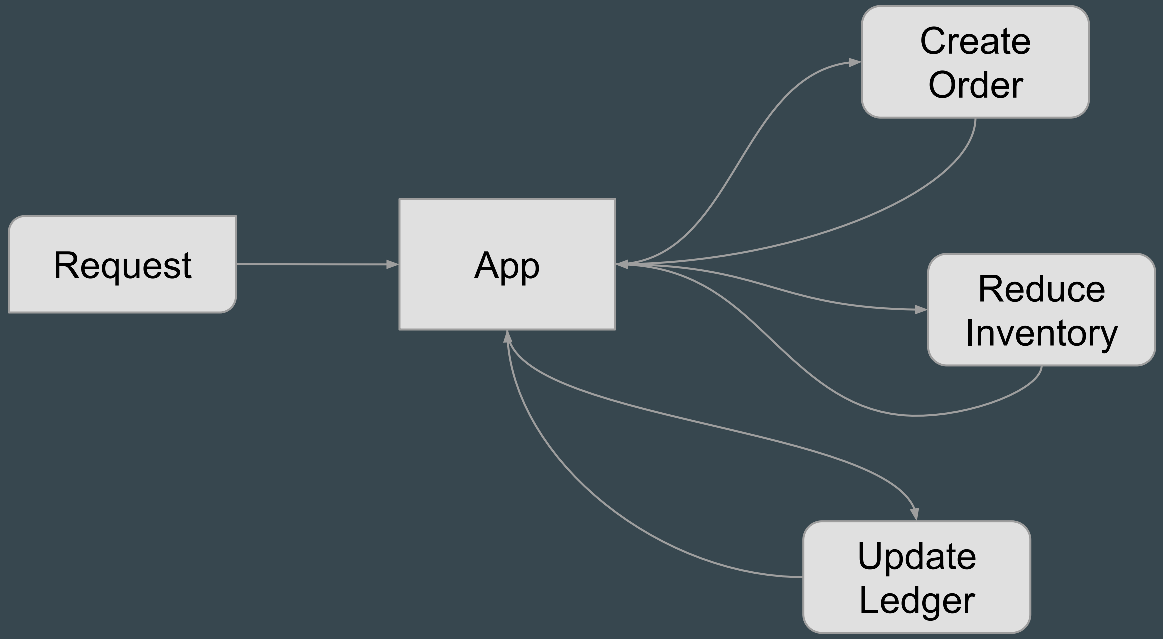
Embrace the onion... trust the onion...

- Focus on your service, trust the onion layers
- Let the experts do what they do best
- Use professional-grade wheels, don't re-invent your own
  - *Unless your core business makes money from the sale of wheels*
- Cloud platforms are plentiful, available, and *mature*.
  - Rely on them wherever you can
- **If your service is buggy or broken, none of these onion layers can save you.**

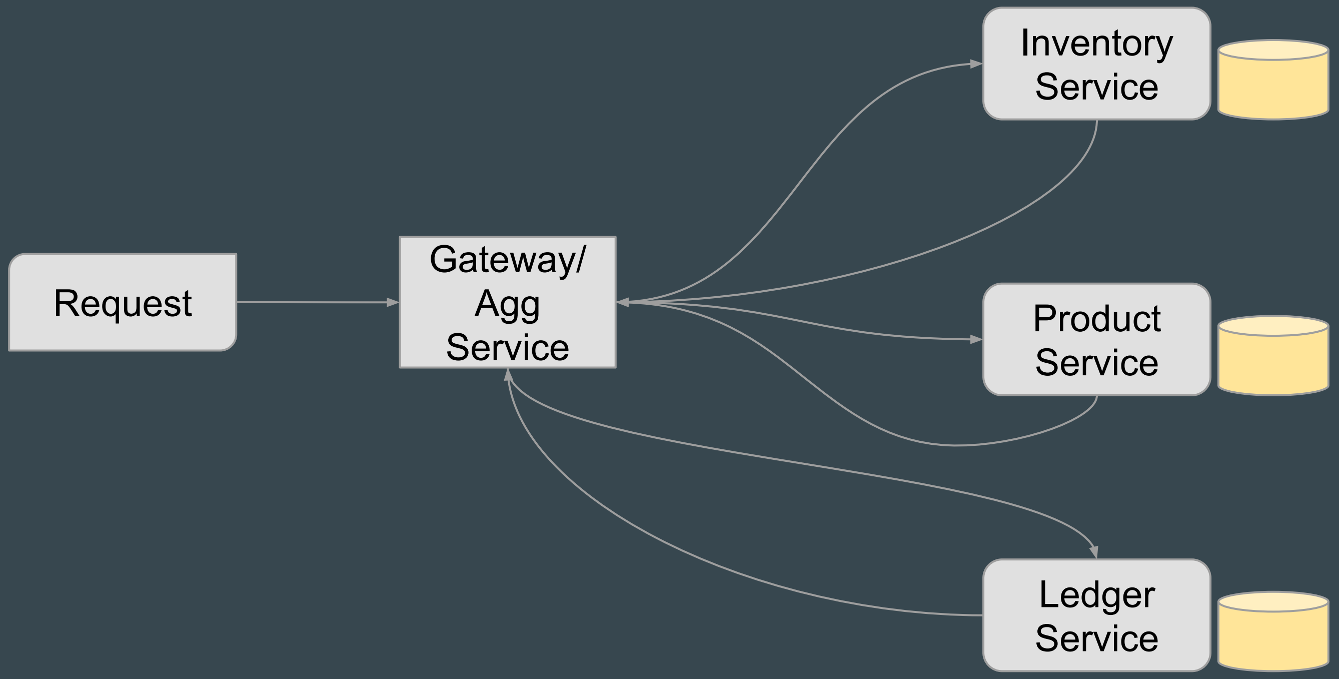
Too much worrying about outer layers is a smell

Sidecar... onions ... Side-onions!

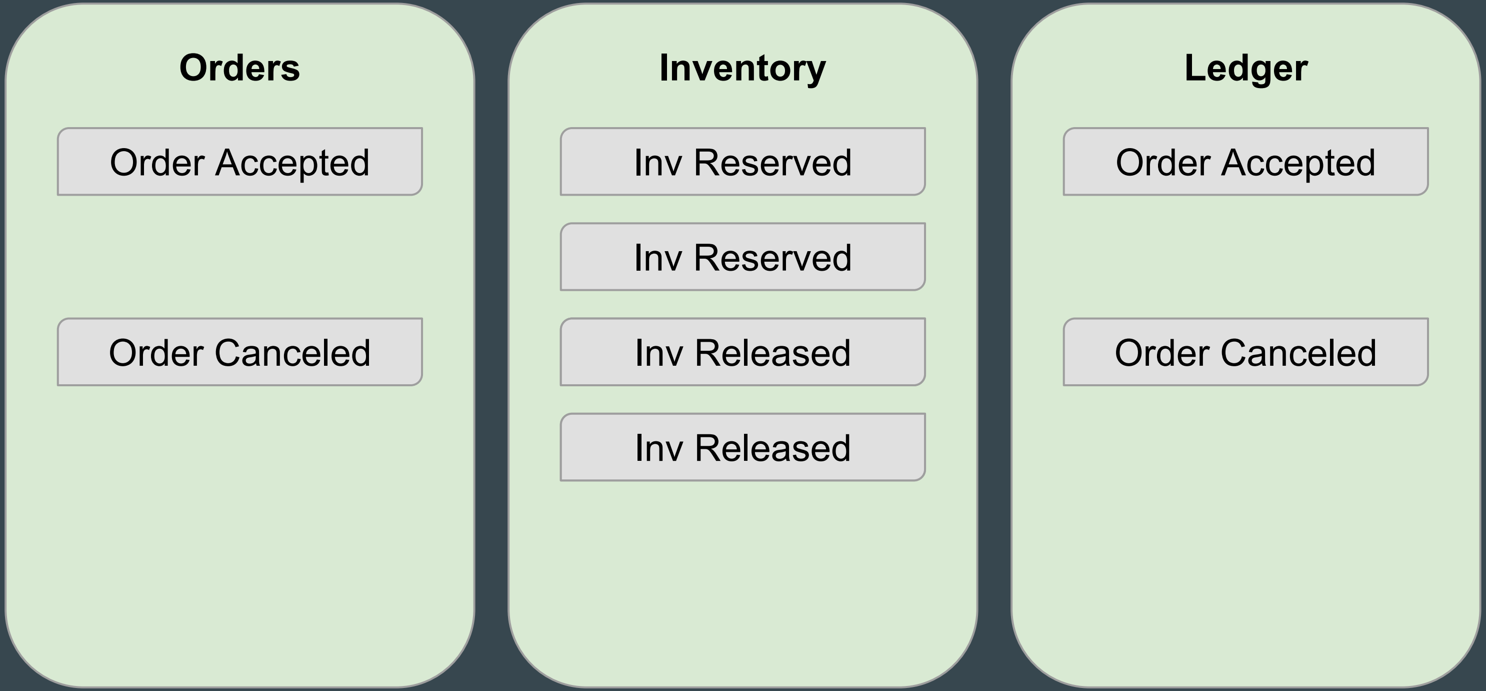
Classic Distributed Transaction



Distributed Failure via Microservices



Immutable Events and Shared-Nothing Activity Modeling



Life Beyond Distributed Transactions - http://queue.acm.org/detail.cfm?id=3025012, *Pat Helland*

Facts vs State

FACT
Order Created @ 2:12PM by “bob”
Inventory Reserved SKU 12345
Inventory Reserved SKU 12345
Order 12 Canceled @ 9:21PM by “bob”
Inventory Released SKU 12345

Eventual Consistency

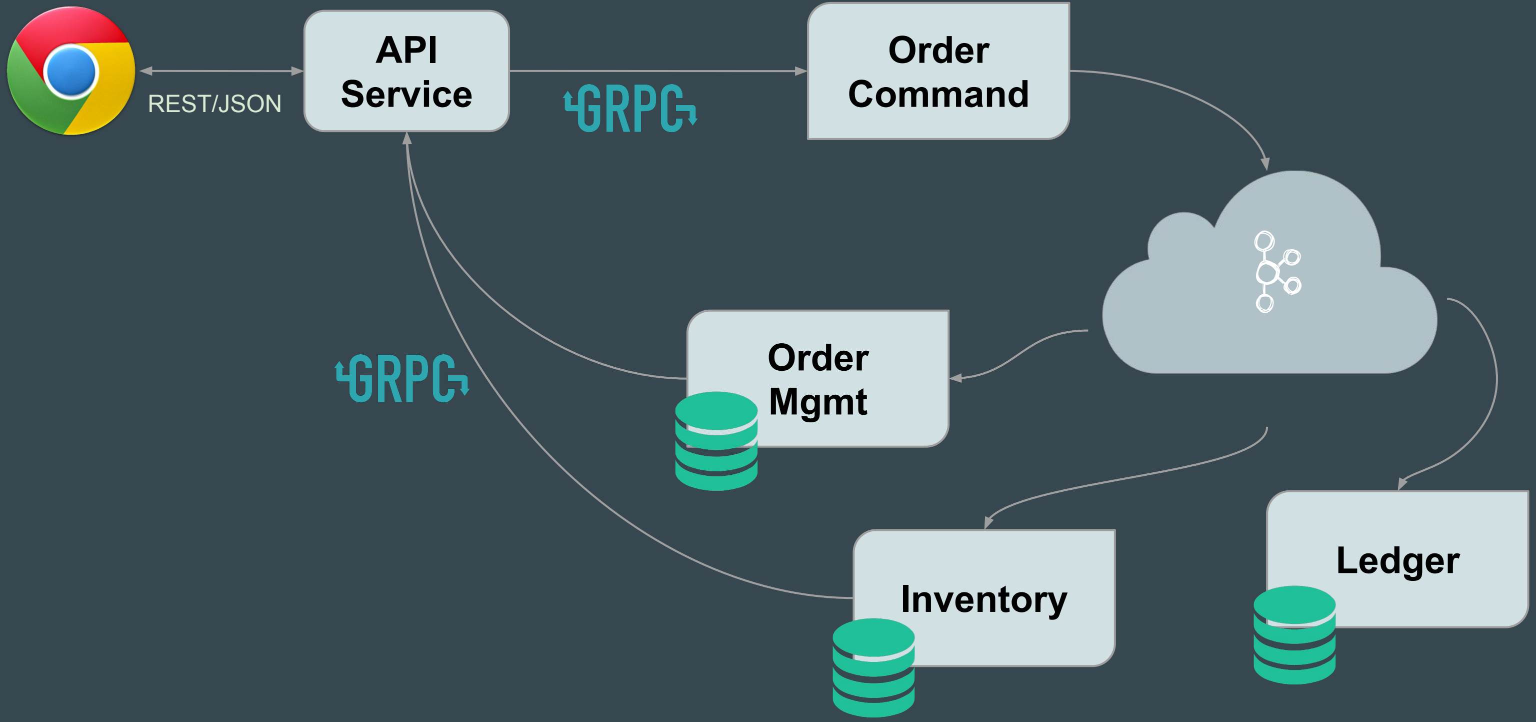
STATE
Order { ID = 12, Status = Open, … }
Item 12345 { WarehouseQty = 99 }
Item 12345 { WarehouseQty = 98 }
Order { ID = 12, Status = Canceled, … }
Item 12345 { WarehouseQty = 99 }

Being right 5 seconds from now is always better than being wrong right now.

Partial Foods - Sample Application

- Transactions modeled as immutable activities
- State exposed as eventually consistent, materialized views
- Designed for Scale, Throughput, Durability, Reliability
- More than just “Hello World”
- Online Store
  - Sells groceries
  - Orders fulfilled asynchronously
  - Inventory releases and holds
  - Order creation and cancellation
  - Durable message broker

“Partial Foods” - Microservices Architecture



Partial Foods

Sample App Demo
&
Code Walkthrough

- gRPC Services
- Pub/Sub Messaging
  - Kafka
- Entity Framework Core 2.0
  - Postgres
- Eventual Consistency
- Modeling **Activities** instead of Distributed Transactions
- Embrace Shared-Nothing

Recap / Lessons Learned

- Not all microservices are RESTful JSON services
    - Show **gRPC** some love
- Modeling entities and immutable, distributed activities can be a simple solution to a complex problem
    - Event Sourcing and CQRS are for more than just Netflix
    - Get it working first, materialize your views *later*.
- .NET Core 2.0 is *excellent*, and ready for production.
    - 2.0 tooling is superb, 1.x ... *not so much*
- Containerize your workloads
- Build cloud-first or cloud native
    - Kubernetes, Cloud Foundry, AWS, Azure, Google Cloud, etc.
    - 12/15 factors
- Microservices are an architectural pattern, ***NOT*** a framework/library
    - Nor are they a panacea
- Disappointed that Partial Foods doesn't have a proper logo

Q&A

- Twitter @KevinHoffman
- Always available to chat
- Partial Foods Code
  - http://github.com/microservices-aspnetcore/partialfood-*
  - Requires Postgres, Zookeeper & Kafka, .NET Core 2.0
- Resources
  - grpc.io
  - https://kafka.apache.org/quickstart