MICROSERVICES

Little Services

Big Apps

OR

EVERYTHING I EVER NEEDED TO KNOW ABOUT MICROSERVICES I LEARNED FROM WCF

Who is this Doug guy, anyway?

douglampe.com

Why Microservices?

- Scale

Why Microservices?

- Scale
- Maintainability
- Testability
- Deployability
- Fault Isolation
- Technology Independence
- Cost
- Configuration Management

Chris Richardson: https://microservices.io/

Why NOT Microservices?

- Complexity
  - *How do they talk to each other?*
  - *How do I do integration testing?*
  - *How do I manage transactions across multiple services?*
  - *How do I deploy all of these services?*
- Cost
  - *Wait! Didn't you have "Cost" on the previous slide?*

Chris Richardson: https://microservices.io/

Microservices Concerns

- Service Registry/Discovery
- Communication
- Authentication/Authorization
- API Gateway
- Not Appearing on Their Own Slide:
  - *Data Storage/Synchronization*
  - *Caching*
  - *Tracing*
  - *Distributed Transactions*

Service Registry Providers

- Amazon Route 53 
- Consul
- Docker Swarm
- Doozer
- Etcd
- Eureka
- NSQ
- Service Fabric Naming Service 
- SmartStack
- Zookeeper
- Serf

API Protocols

- REST
- gRPC
- WebSockets
- SOAP

Authentication/Authorization

- Authentication
  - *OAuth2*
  - *OpenID Connect*
- Authorization
  - *Signed JWTs*
  - *Claims*

API Gateway Providers

- Amazon API Gateway 
- Azure Application Gateway 
- NGINX
- Traefik

WHAT DOES WCF HAVE TO DO WITH ANY OF THIS?

WCF Concepts

- Data Contract
- Service Contract
- Message Contract
- Binding
- Transport

WCF Concepts

- Data Contract
- Service Contract
- Message Contract
- Binding
- Transport

```
[ServiceContract(Namespace = "http://foo")]
public interface ICalculator
{
  [OperationContract]
  double Add(double n1, double n2);
  [OperationContract]
  double Subtract(double n1, double n2);
  [OperationContract]
  double Multiply(double n1, double n2);
  [OperationContract]
  double Divide(double n1, double n2);
}
```

WCF Concepts

- Data Contract
- Service Contract
- Message Contract
- Binding
- Transport

```
public class CalcParams
{
  double n1 { get; set; }
  double n2 { get; set; }
}
```

```
[ServiceContract(Namespace = "http://foo")]
public interface ICalculator
{
  [OperationContract]
  double Add(CalcParams params);
  [OperationContract]
  double Subtract(CalcParams params);
  [OperationContract]
  double Multiply(CalcParams params);
  [OperationContract]
  double Divide(CalcParams params);
}
```

WCF Concepts

- Data Contract
- Service Contract
- Message Contract
- Binding
- Transport

Microservice Concepts

- Data Contract
- Service Contract
- Message Contract
- Binding
- Transport

WCF Concepts

- Data Contract
- Service Contract
- Message Contract
- Binding
- Transport

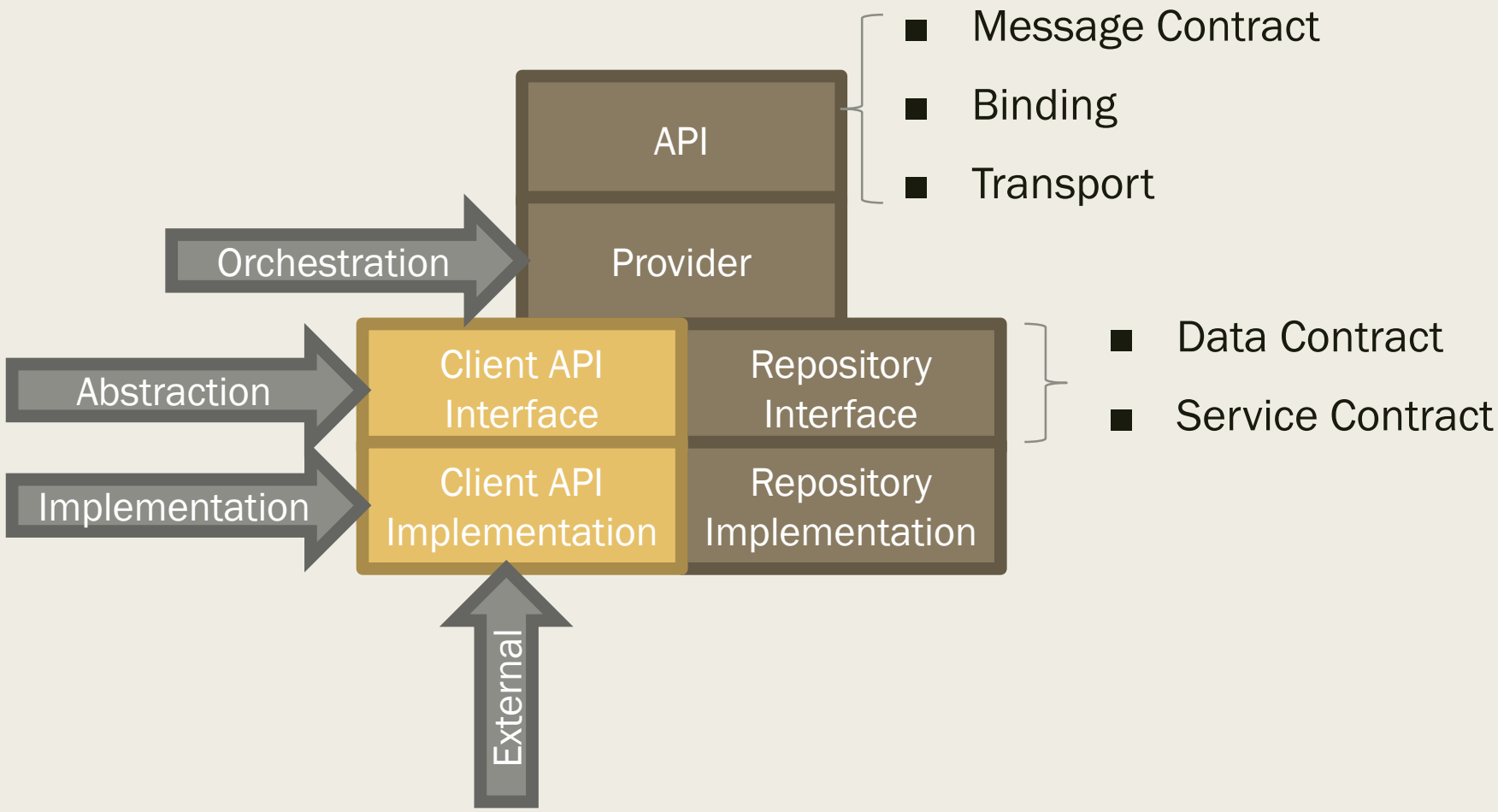
Microservice Concepts

- Data Contract
- Service Contract
- Message Contract
- Binding
- Transport

Domain

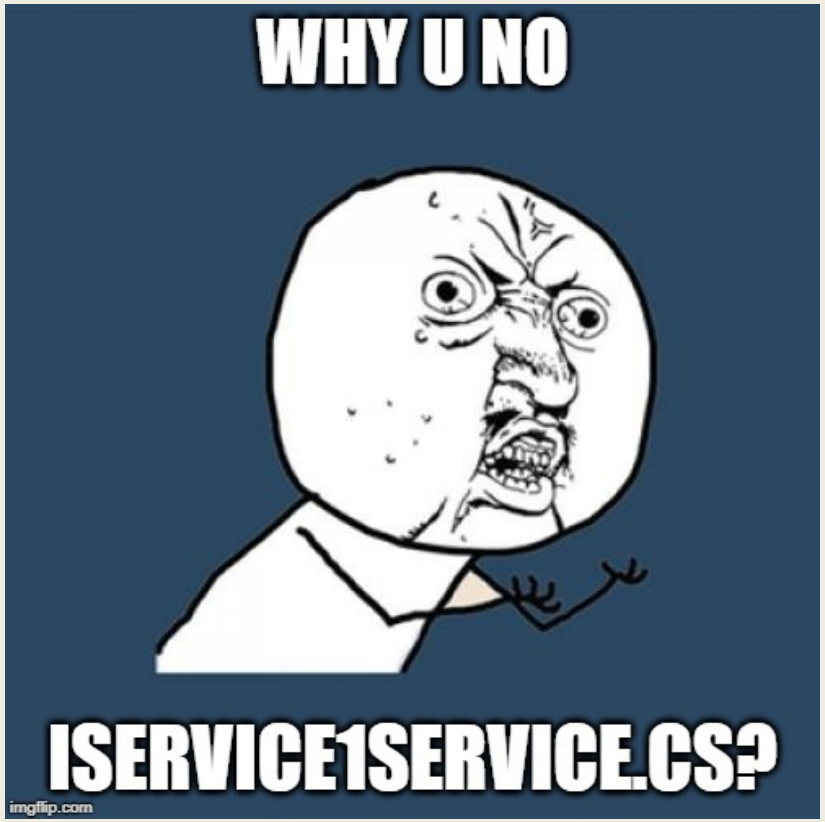
API

Putting It Together



Sample Solution Structure

- Service1.Core
  - *IService1Client.cs*
  - *IService1Repository.cs*
  - *Service1Model.cs*
  - *Service1Provider.cs*
  - *\References*
    - Service2.Core
- Service1.Rest
  - *Service1Rest.cs*
  - *Service1RestClient.cs*
- Service1.Grpc
  - *Service1Grpc.cs*
  - *Service1GrpcClient.cs*
- Service1.Sql
  - *Service1Sql.cs*
- Service1.Redis
  - *Service1Redis.cs*
- Service1.App
  - *Program.cs*
  - *\References*
    - Service1.Sql
    - Service1.Grpc
    - Service2.Rest



QUESTIONS

(ME FIRST)

Questions

- How hard is it to build a monolithic application out of the sample solution?
- How hard is it to convert a monolithic application to a microservices application?

QUESTIONS?

(Your Turn)