Microservices

and RESTful APIs

Yegor Bugayenko

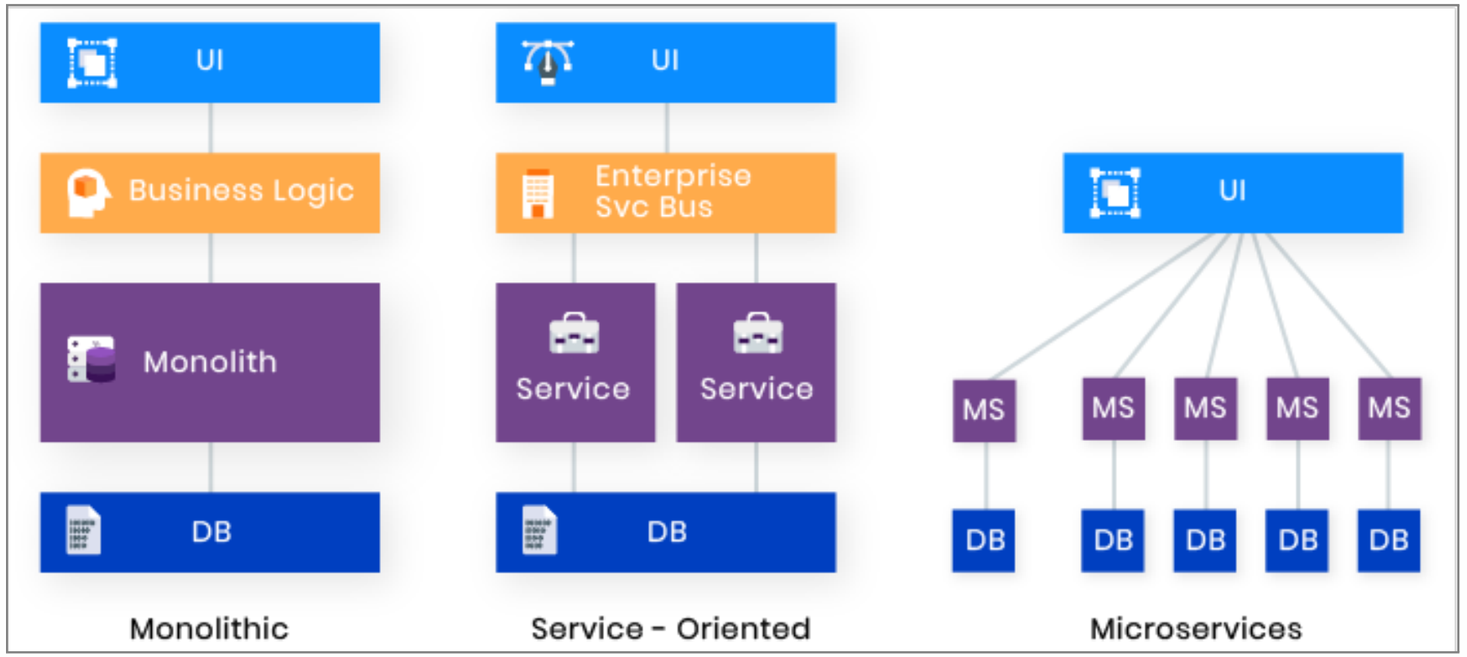
Lecture #11 out of 16
80 minutes

The slidedeck was presented by the author in this YouTube Video

- Service-Oriented Architecture (SOA)
- Microservices and RESTful API
- Protobuf and gRPC
- Internet of Things (IoT)
- Books, Venues, Call-to-Action

Chapter #1:

Service-Oriented Architecture (SOA)


UI
Business Logic
Monolith
DB
Monolithic
UI
Enterprise Svc Bus
Service
Service
DB
Service - Oriented
UI
MS
MS
MS
MS
MS
DB
DB
DB
DB
DB
Microservices

[RPC SOAP CORBA IDL EJB]

XML RPC

An example of a typical XML-RPC request would be:

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
        <value><i4>40</i4></value>
    </param>
  </params>
</methodCall>
```

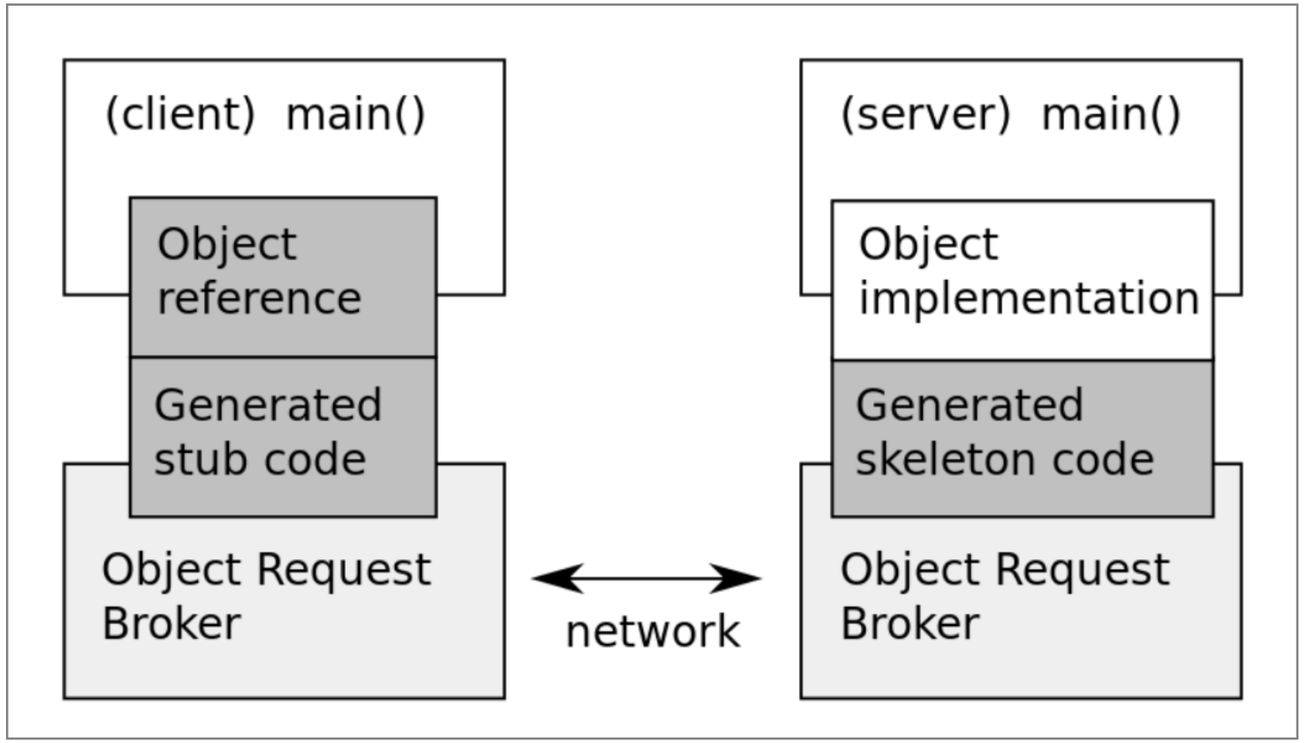
An example of a typical XML-RPC response would be:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>South Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

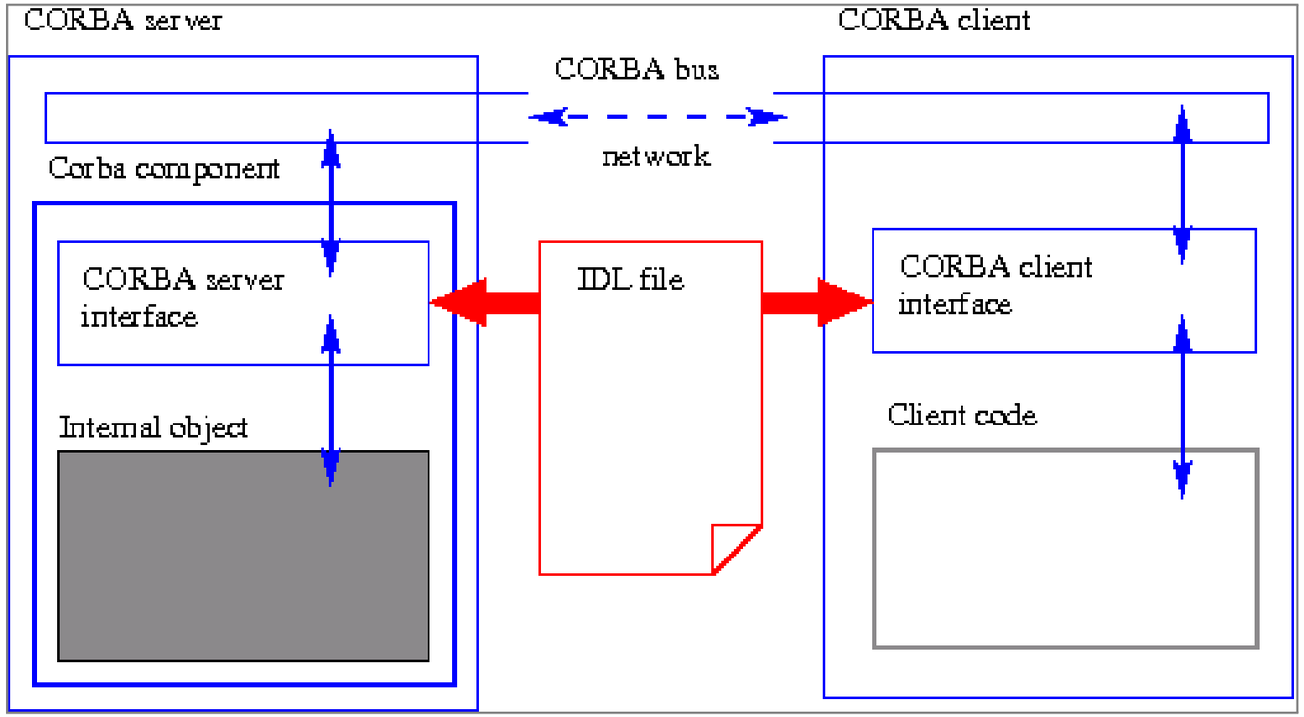
Simple Object Access Protocol (SOAP)

```xml
<?xml version='1.0' Encoding='UTF-8' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
                env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2007-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
                env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
   <n:name>Fred Bloggs</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2007-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2007-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference></p:seatPreference>
   </p:return>
  </p:itinerary>
 </env:Body>
</env:Envelope>
```

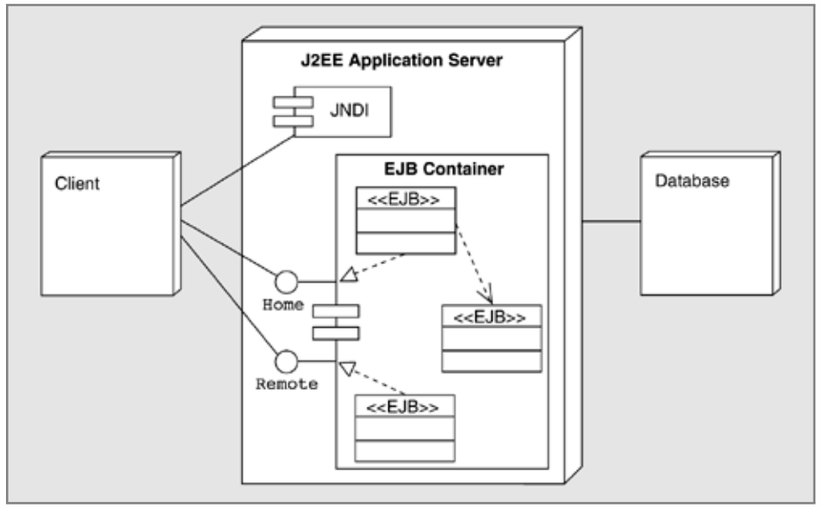
Common Object Request Broker Architecture (CORBA)



Interface Description Language (IDL)



Enterprise Java Beans (EJB)



Java Naming and Directory Interface (JNDI) is an API that provides naming and directory functionality to applications written using the Java.

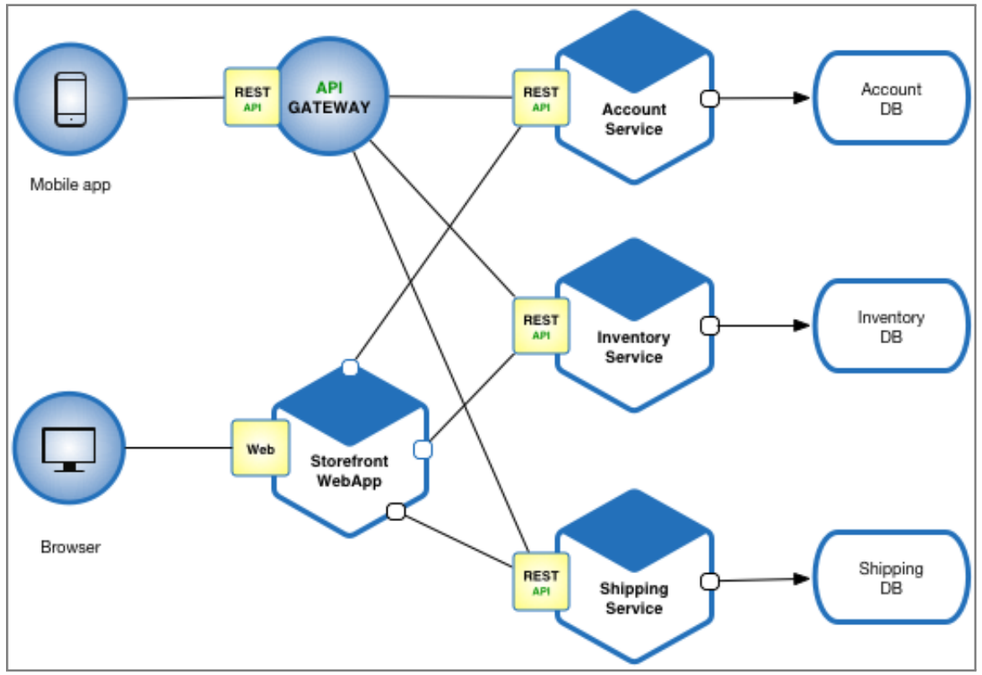
Remote Method Invocation (RMI) is a Java API that performs remote method invocation, the object-oriented equivalent of remote procedure calls, with support for direct transfer

of serialized Java classes and distributed garbage-collection.

Chapter #2:

Microservices and RESTful API

Microservices



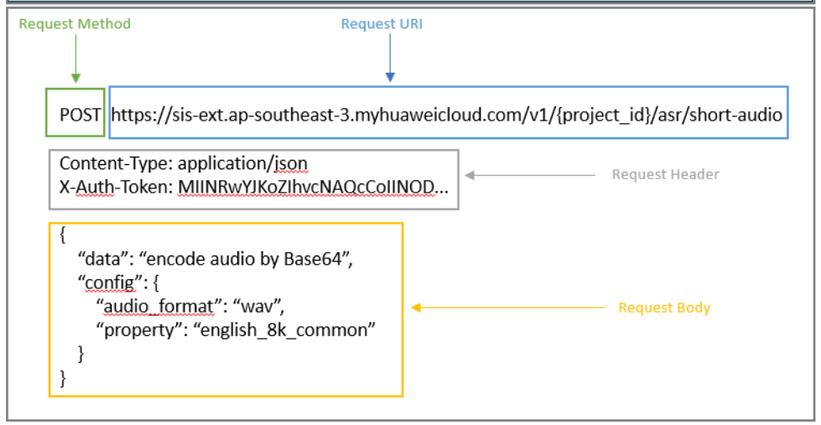
“Microservices are a modern interpretation of service-oriented architectures used to build distributed software systems." — Wikipedia

Stateless vs. Stateful Architecture

“A stateless process or application can be understood in isolation. There is no stored knowledge of or reference to past transactions. Each transaction is made as if from scratch for the first time.” — RedHat

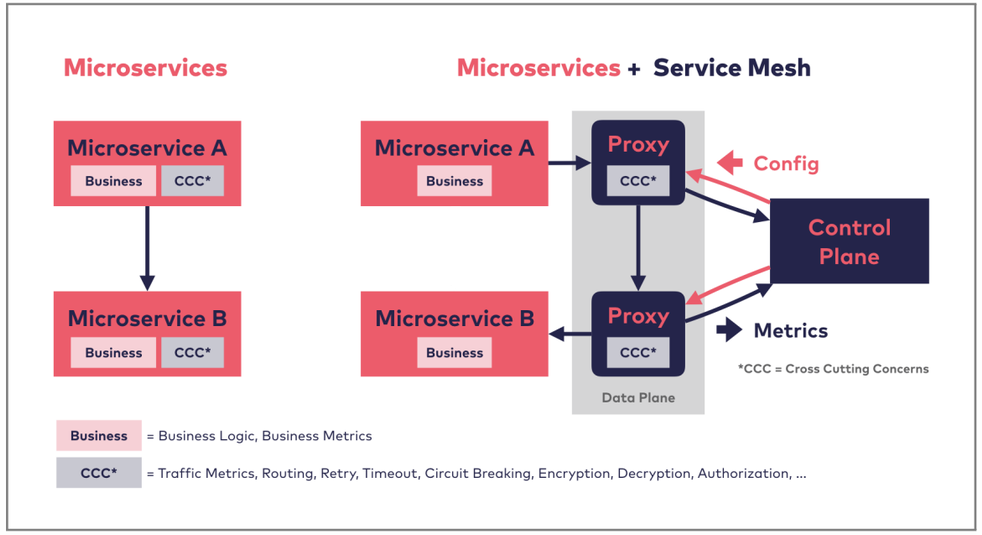
Representational State Transfer (REST)

Task	Method	Path
Create a new customer	POST	/customers
Delete an existing customer	DELETE	/customers/{id}
Get a specific customer	GET	/customers/{id}
Search for customers	GET	/customers
Update an existing customer	PUT	/customers/{id}



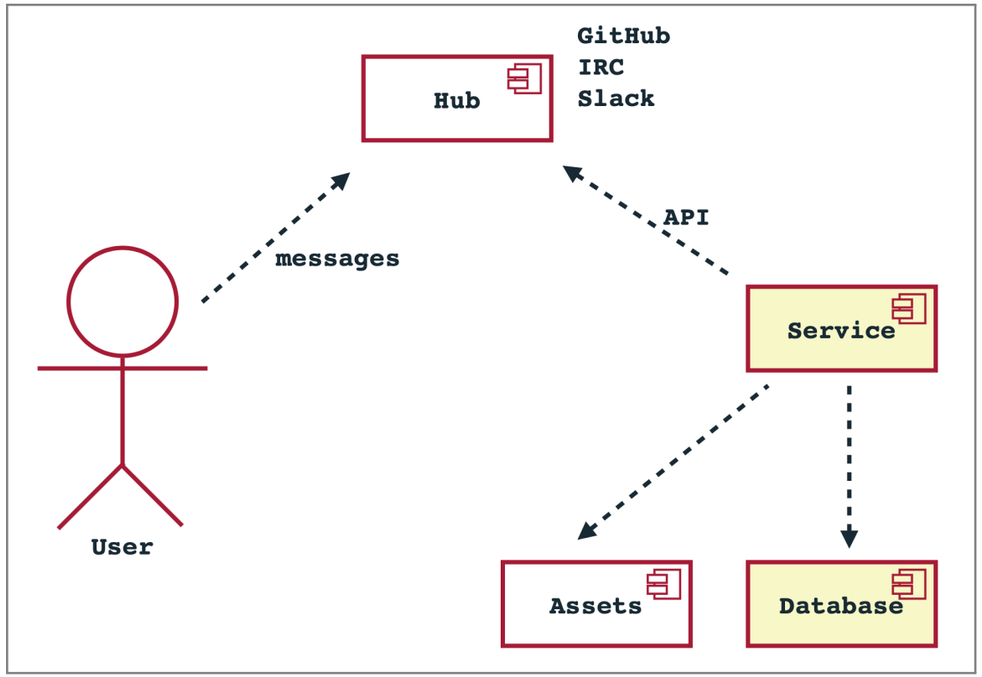
Hypermedia As The Engine Of Application State (HATEOAS)

Service Mesh



"A service mesh is a dedicated infrastructure layer for facilitating service-to-service communications between services or microservices, using a proxy" — Wikipedia

Chatbots

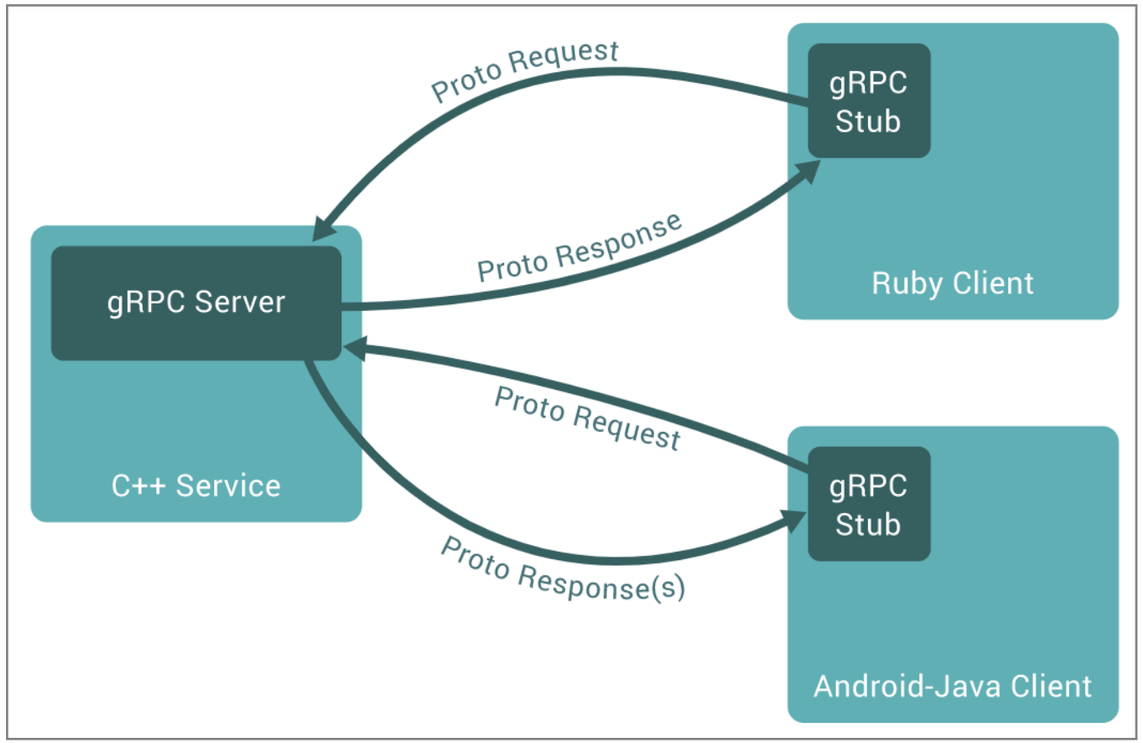

GitHub
IRC
Slack
Hub
API
messages
Service
User
Assets
Database

Chapter #3:

Protobuf and gRPC

[gRPC Protobuf]

gRPC



[gRPC Protobuf]

Protobuf

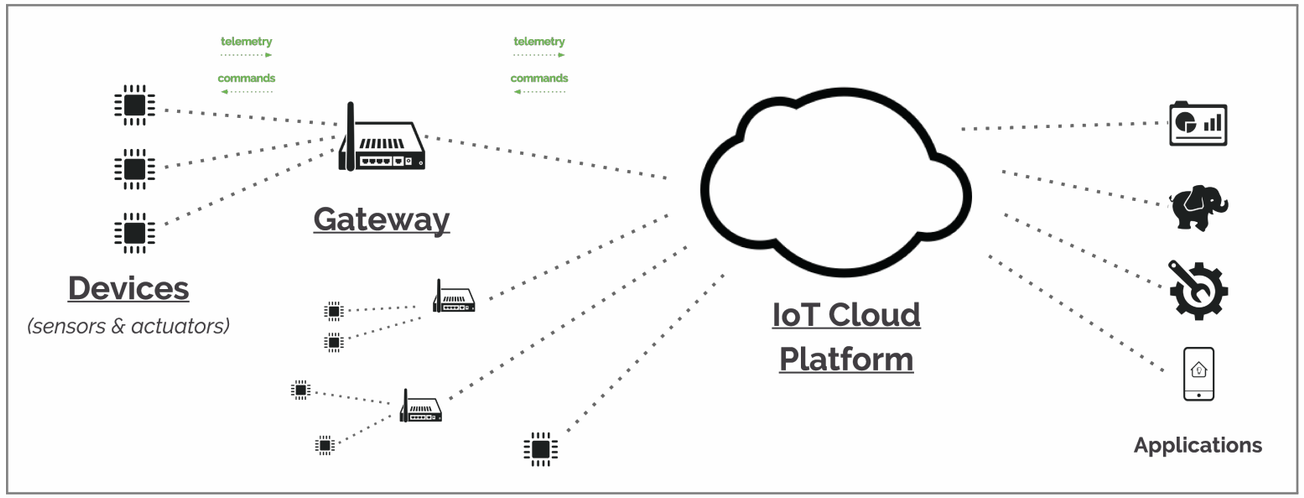
```
syntax = "proto2";

package tutorial;

message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    optional string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phones = 4;
}

message AddressBook {
  repeated Person people = 1;
}
```

```
// name
inline bool has_name() const;
inline void clear_name();
inline const ::std::string& name() const;
inline void set_name(const ::std::string& value);
inline void set_name(const char* value);
inline ::std::string* mutable_name();

// id
inline bool has_id() const;
inline void clear_id();
inline int32_t id() const;
inline void set_id(int32_t value);

// email
inline bool has_email() const;
inline void clear_email();
inline const ::std::string& email() const;
inline void set_email(const ::std::string& value);
inline void set_email(const char* value);
inline ::std::string* mutable_email();
```

[gRPC Protobuf]

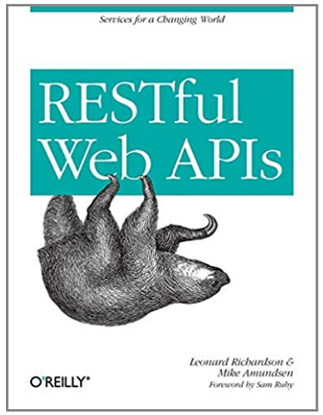
Chapter #4:

Internet of Things (IoT)

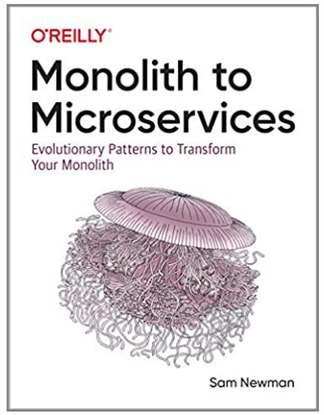

telemetry
commands
telemetry
commands
Gateway
Devices
(sensors & actuators)
IoT Cloud
Platform
Applications

Chapter #5:

Books, Venues, Call-to-Action



"RESTful Web APIs: Services for a Changing World" by Leonard Richardson et al.



"Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith" by Sam Newman

Where to go:

IEEE International Conference on Software Architecture (ICSA)

Call to Action:

Design your own RESTful API and publish it at rapidapi.com or similar place, where APIs are “published.”

Still unresolved issues:

- How to validate an API?
- How to generate an API from object model?
- How to test API automatically?
- How to spot integration mistakes between APIs?

Bibliography