Building Microservices with gRPC and NATS

Shiju Varghese

Consulting Solutions Architect

saltmarch MEDIA

GREAT INDIAN DEVELOPER SUMMIT

April 28, 2017

About Me

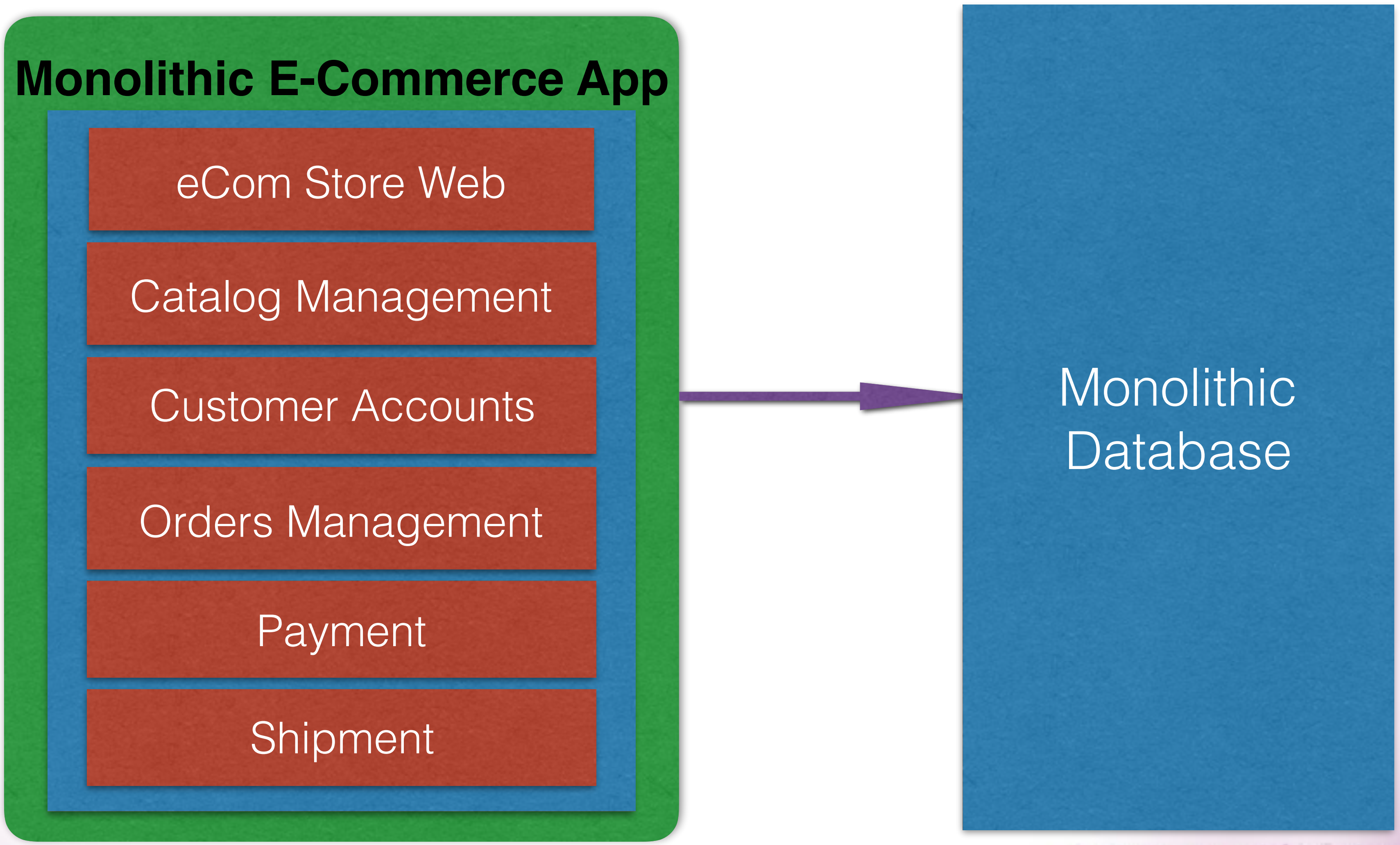
- Consulting Solutions Architect and Trainer
- Focused on Golang, Microservices and Cloud-Native distributed systems architectures
- Published Author: "Web Development with Go" and "Go Recipes"
- Honoured with Microsoft MVP award seven times
- Blog: https://medium.com/@shijuvar

Agenda

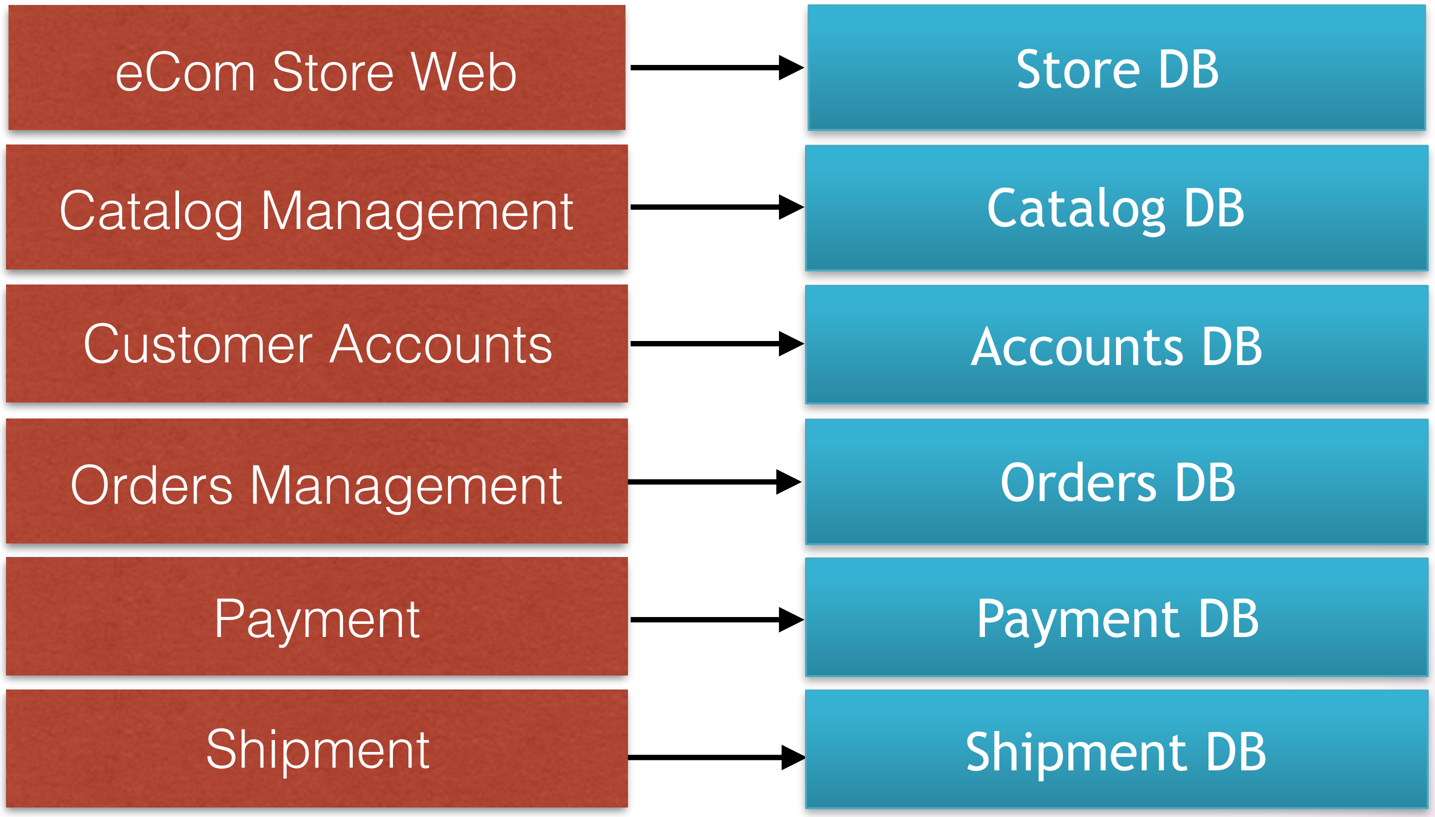
- Inter-Process communications in Microservices architecture
- Building high performance APIs with gRPC and Protocol Buffers
- Building Microservices with event-driven architectures using NATS

MICROSERVICES ARCHITECTURE

Monolithic Architecture



Moving to Microservices



Microservices

- Software broken up into functional components
- Componentization via Services in which each service is packaged as one unit of execution
- Independent, autonomous process with no dependency on other Microservices
- Autonomous services around Bounded Context
- Decentralization of data management
- Independently replaceable and upgradeable

Challenges

- A business transaction may span into multiple services
- Decentralization of data management
- Communications between Microservices without having any performance bottleneck

Inter-Process Communications between Microservices

- Communications over high performance APIs
- Event-Driven architecture using messaging systems

Design Considerations for Building APIs

- Scaling APIs into millions (even billions) of APIs calls
- Wire format; Serialisation and deserialisation of messages
- Building streaming APIs
- RESTful Vs RPC?
- Text encoding Vs binary encoding?

Why not REST

- Uses HTTP/1.x; Separate TCP Connection per request
- Text on the wire; Not performance efficient
- Harder API evolution
- Not Domain-Specific
- Not strongly-typed
- Lack of streaming capabilities

GRPC AND PROTOCOL BUFFERS

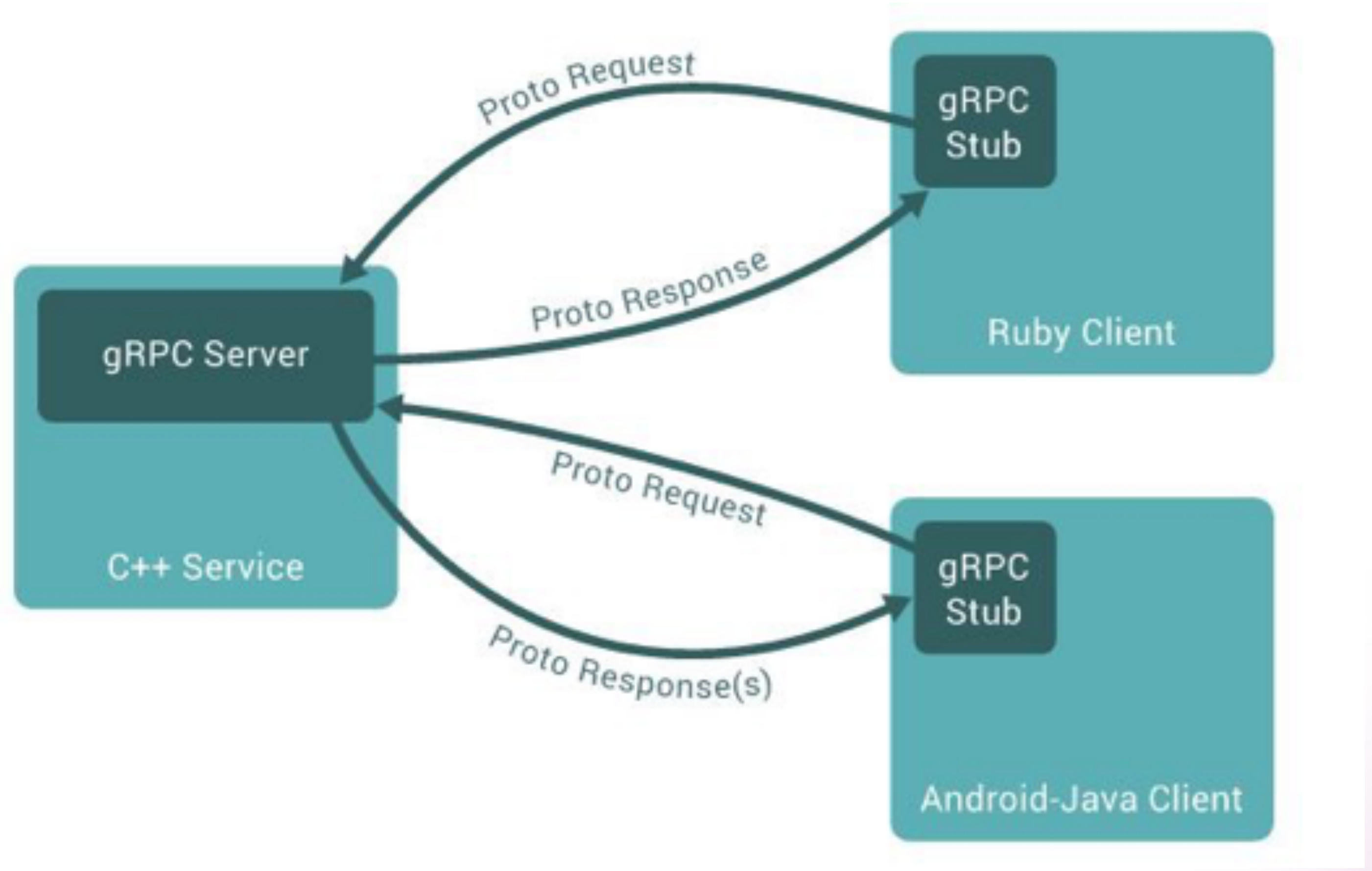
What is gRPC

- High performance, open-source universal RPC framework
- A Cloud Native Computing Foundation(CNCF) project
- Open source version of Google’s internal framework Stubby
- Uses Protocol Buffers as the IDL
- HTTP/2 for transport
- Bi-Directional streaming
- RPC is efficient, domain-specific and strongly-typed
- Works across languages and platforms

Protocol Buffers

- Google's language-neutral, platform-neutral, extensible mechanism for serialising structured data
- IDL - Describe once and generate interfaces for multiple languages
- Structure of the Request and Response
- Binary format for network transmission
- Supports multiple languages

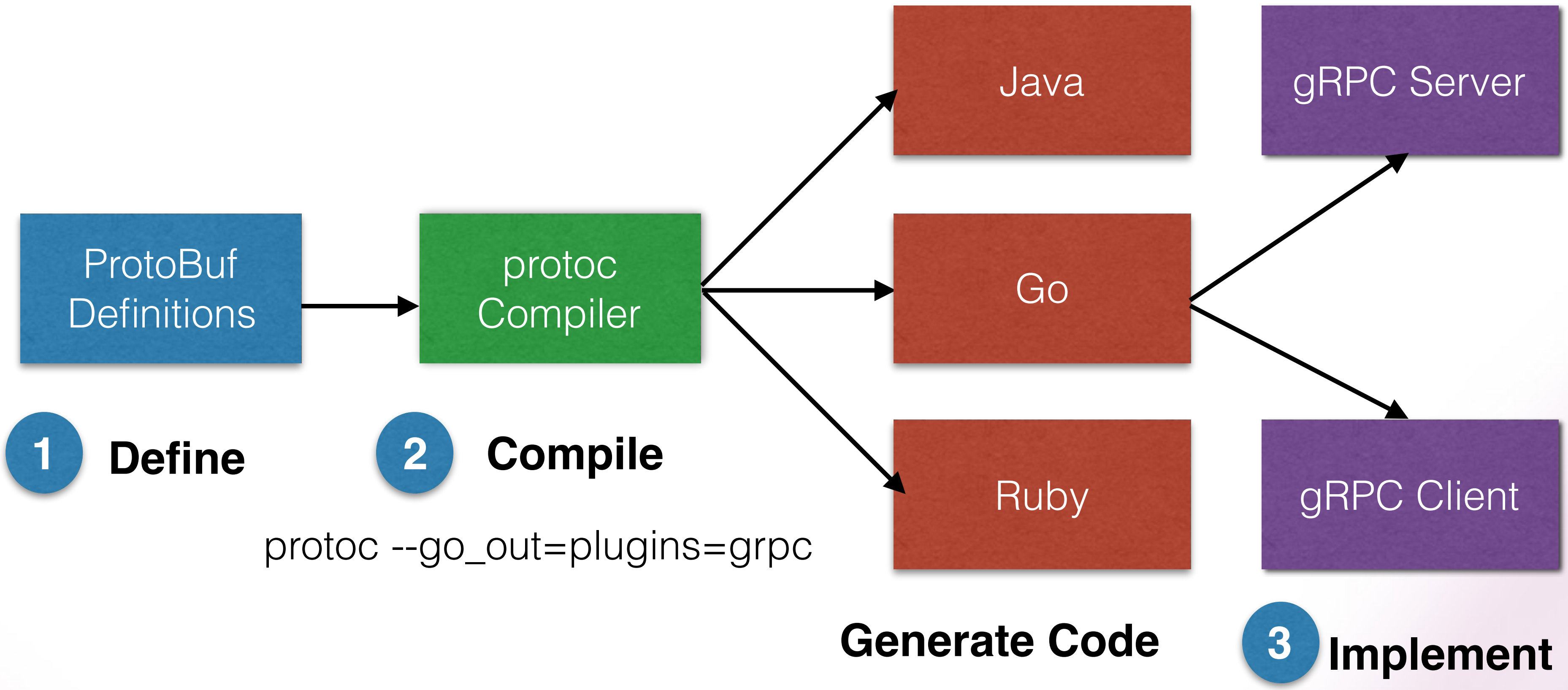
Communication between gRPC Server and Client app



Types of RPC Methods

- Simple RPC
- Server-side streaming RPC
- Client-side streaming RPC
- Bi-directional streaming RPC

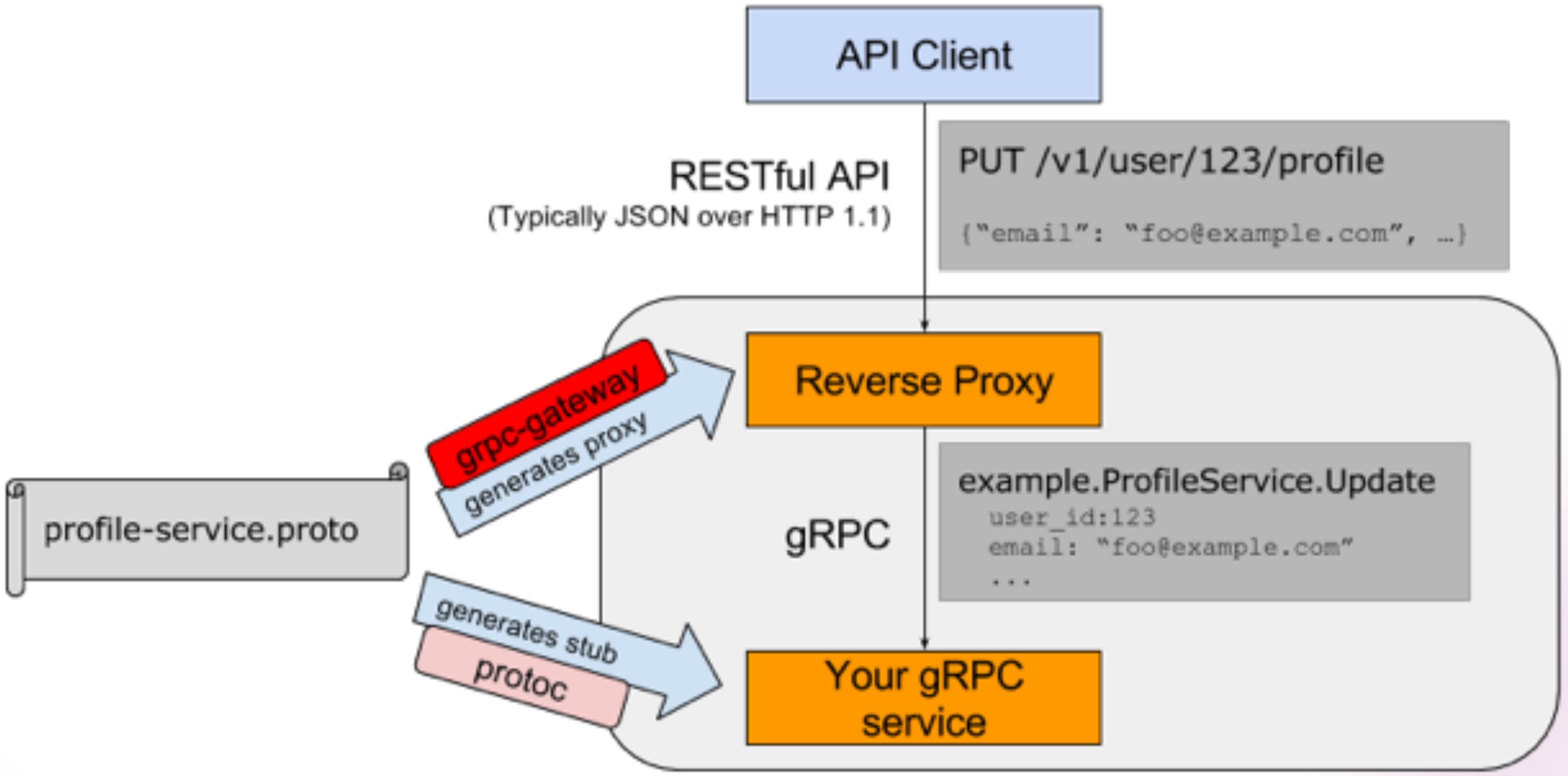
gRPC Workflow



```
protoc --go_out=plugins=grpc
```

DEMO

grpc-gateway - gRPC to JSON Proxy Generator



PUT /v1/user/123/profile

```
{"email": "foo@example.com", …}
```

example.ProfileService.Update

```
  user_id:123
  email: "foo@example.com"
  ...
```

BUILDING MICROSERVICES WITH EVENT-DRIVEN ARCHITECTURE USING NATS

Inter-Process Communication Using an Event-Driven Architecture

Event-Sourcing

Aggregate ID	Aggregate Type	Event ID	Event Type	Event Data
301	Order	1001	OrderCreated	...
301	Order	1002	OrderApproved	...
301	Order	1003	OrderShipped	...
301	Order	1004	OrderDelivered	...

Event Table

NATS

- Open source, lightweight, high-performance cloud native messaging system
- Highly performant, extremely light-weight; Capable of sending 11-12 million messages per second
- Publish-Subscribe messaging system
- Available in two interoperable modules:
  - NATS Server
  - NATS Streaming Server (with persistent messaging)

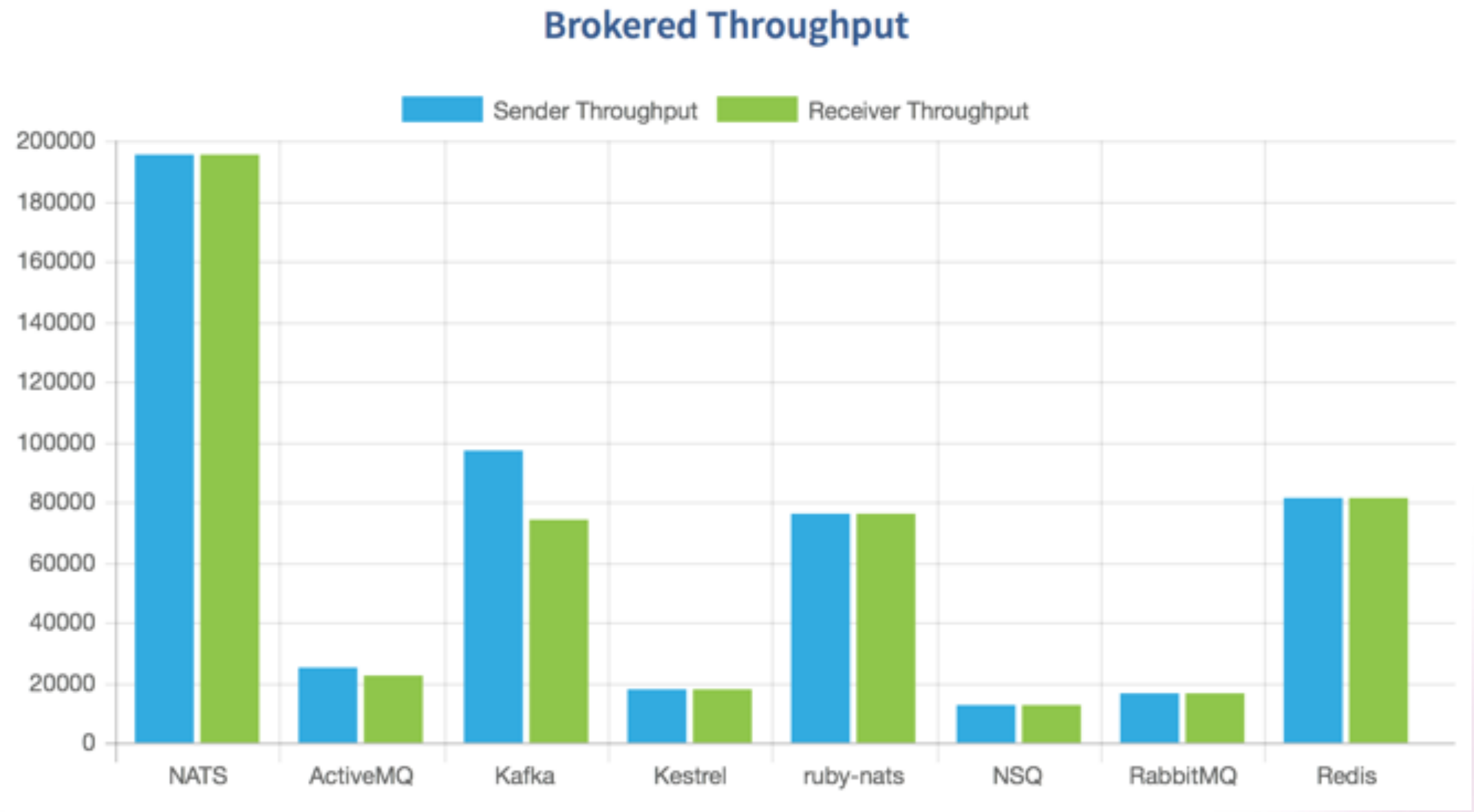


Chart source: bravenewgeek.com/dissecting-message-queues

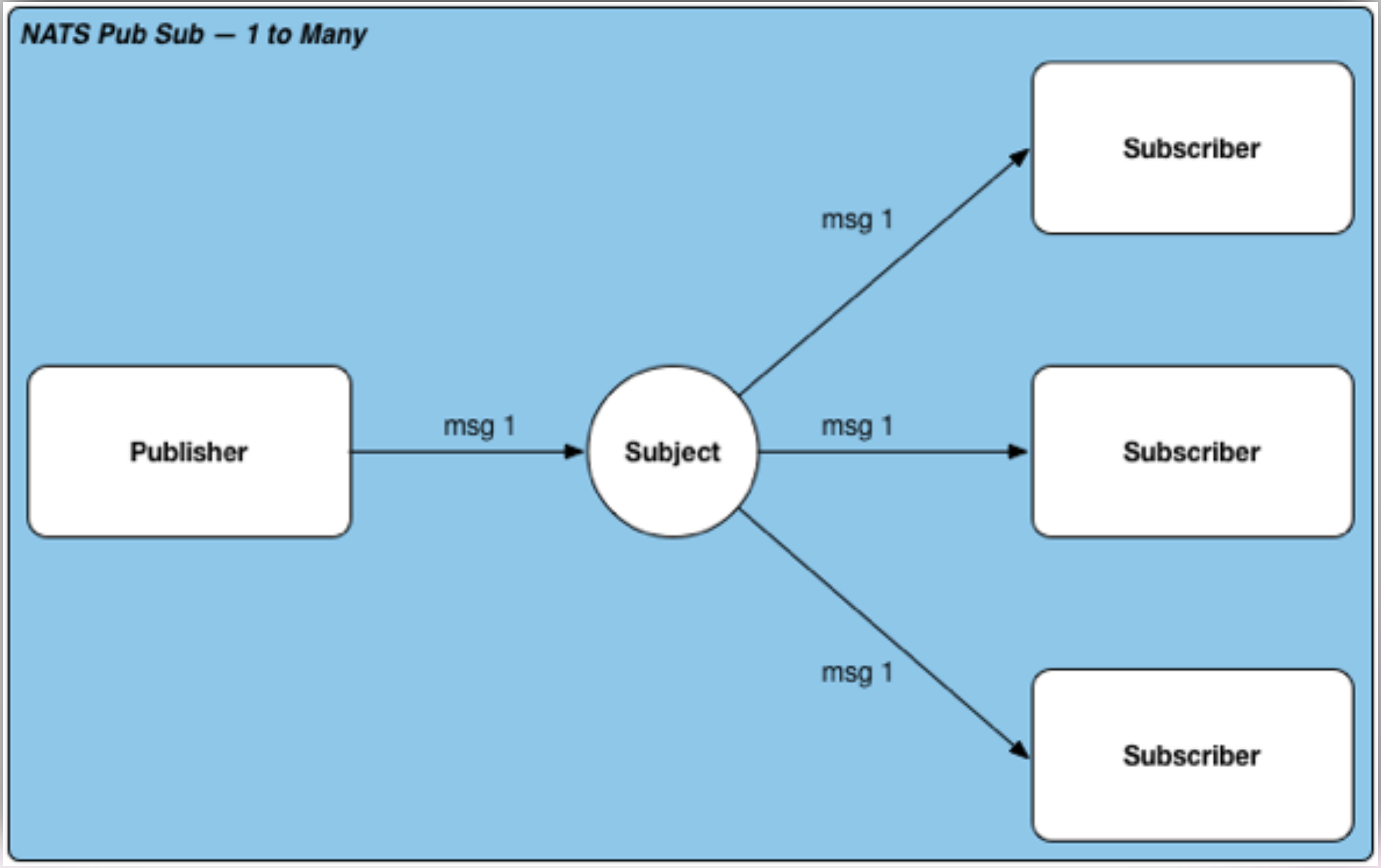
Messaging Patterns

- Publish-Subscribe
- Queueing
- Request-Replay

Components of Messaging Architecture

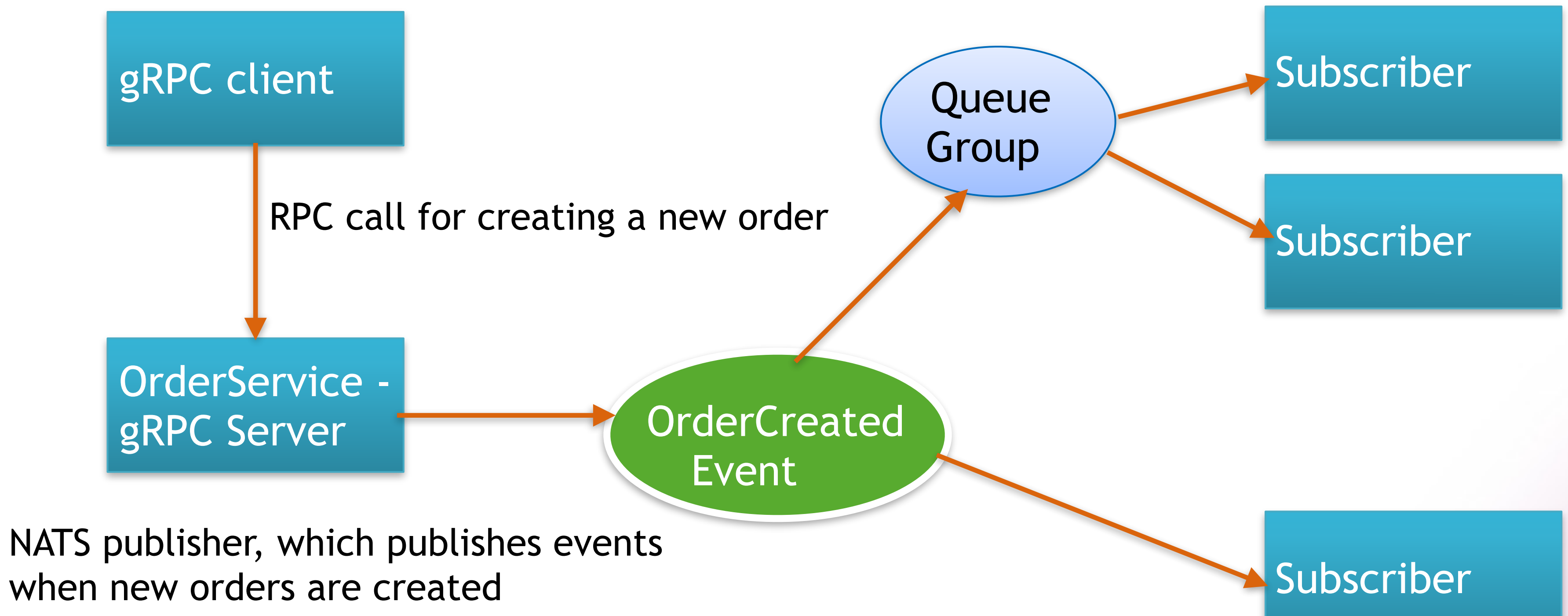
- **Message**: Messages are the unit of data exchange. A payload, which is used for exchanging the data between applications.
- **Subject**: Subject specifies the destination of messages.
- **Producer**: Producers send messages to the NATS server.
- **Consumer**: Consumers receive messages from the NATS server.
- **Messaging Server**: NATS Server distributes the messages from producers to consumers.

Publish-Subscribe



DEMO

Demo - Workflow


gRPC client
RPC call for creating a new order
OrderService - gRPC Server
OrderCreated Event
Queue Group
Subscriber
Subscriber
Subscriber
NATS publisher, which publishes events when new orders are created

Example Source: https://github.com/shijuvar/gokit/tree/master/examples/grpc-nats

Thank you

saltmarch

GREAT INDIAN DEVELOPER SUMMIT

MODS

MOBILE & DISRUPTIVE TECHNOLOGY SUMMIT

October 5-6, 2017
Indian Institute of Science, Bangalore
www.modsummit.com

Register early and get the best discounts

GREAT INDIAN DEVELOPER SUMMIT™ 2018

April 23-28, 2018
Indian Institute of Science, Bangalore
www.developersummit.com