Gopher in an Event-driven playground

Tamimi Ahmad

Developer Advocate, CTO Office

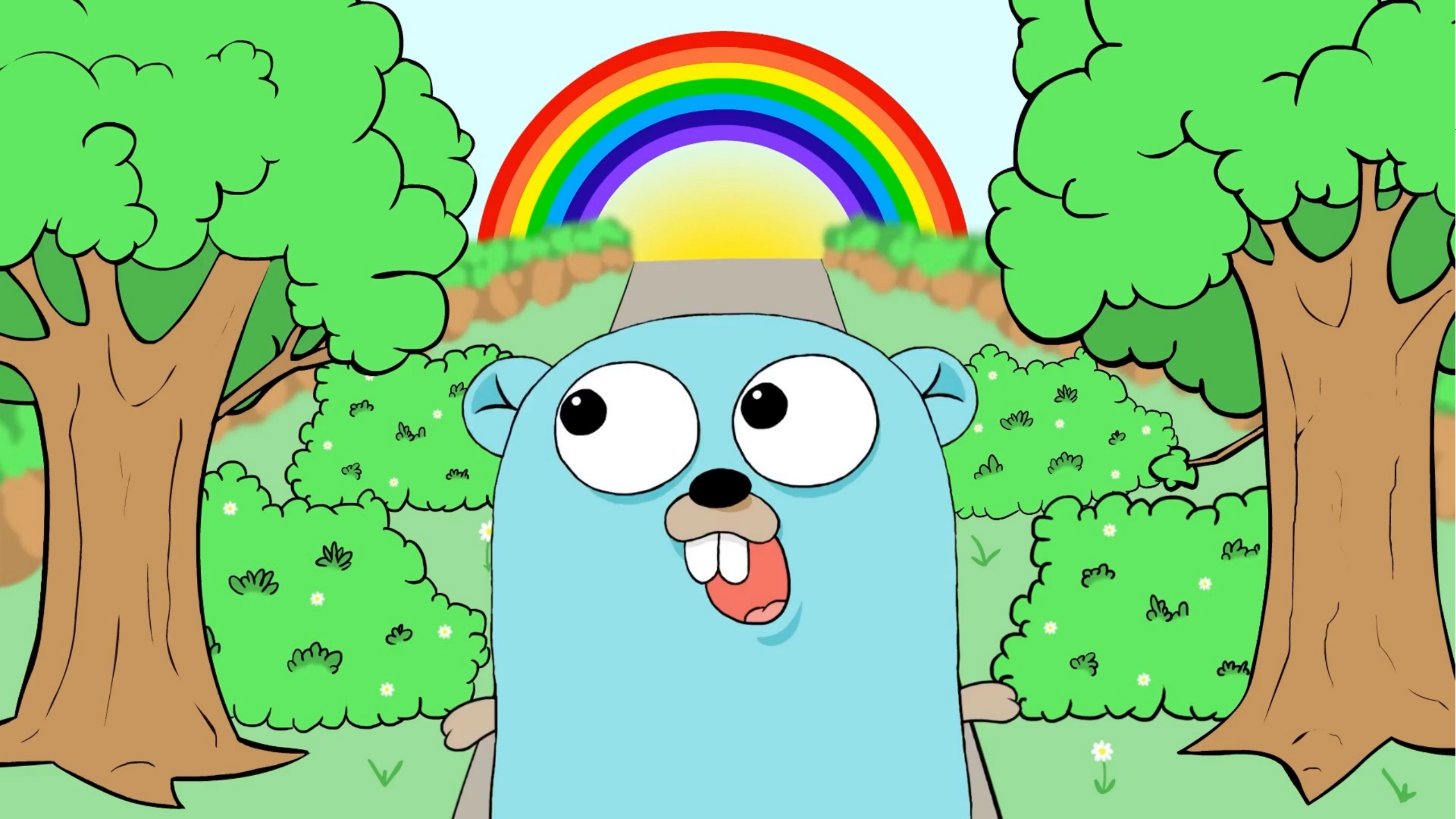
SOLACE

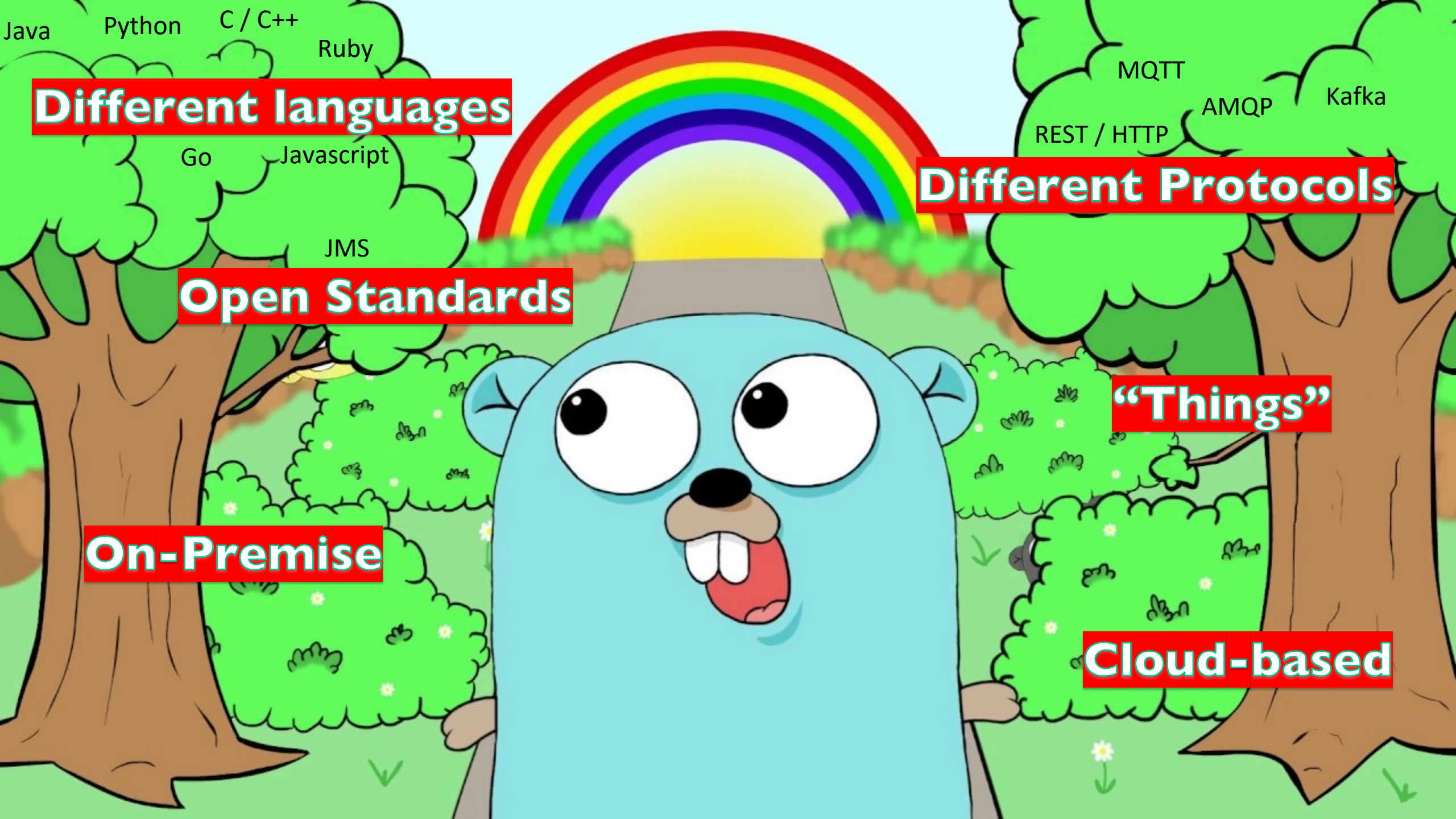
Conf42: Golang 2022

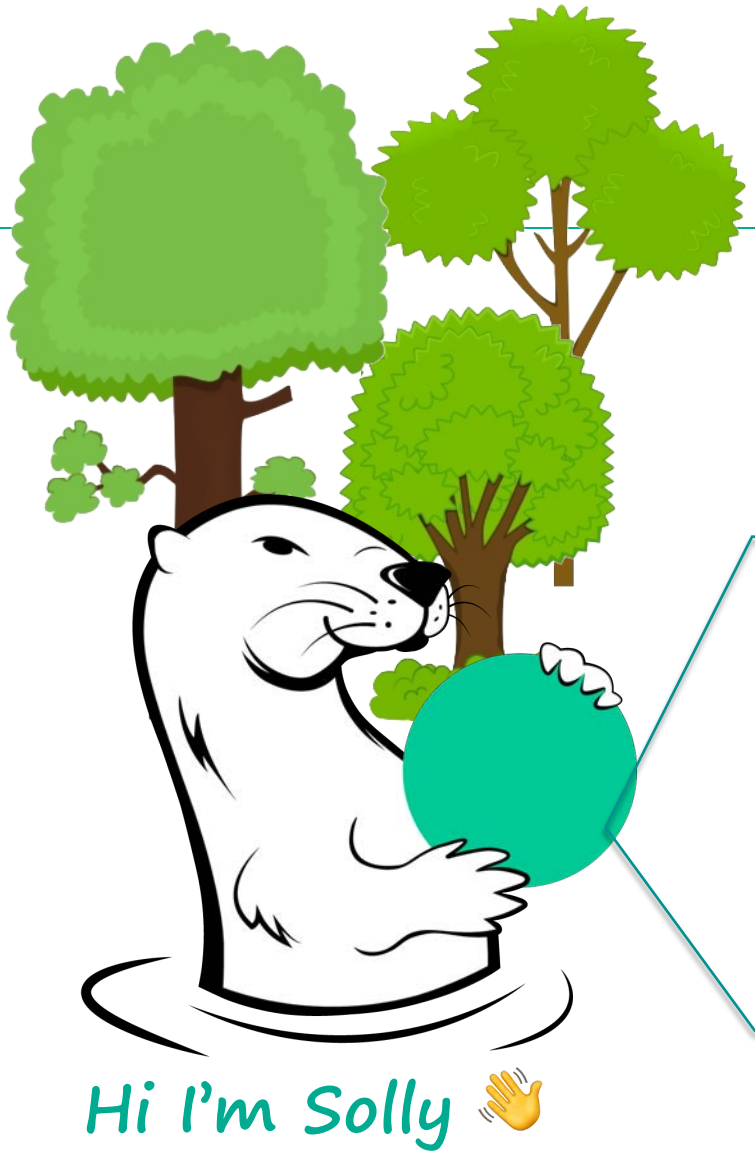
@SolaceDevs

@TweetTamimi

solace.




Java
Python
C / C++
Ruby
Different languages
Go
Javascript
JMS
Open Standards
On-Premise
MQTT
AMQP
Kafka
REST / HTTP
Different Protocols
“Things”
Cloud-based



Hi Solly! How can I accomplish a language and protocol agnostic real-time communication with other microservices?

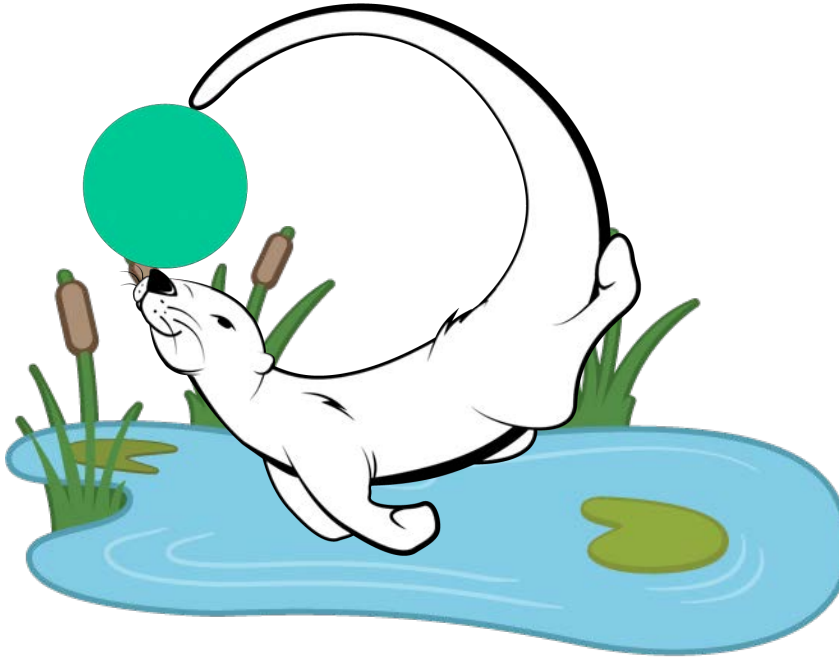
You can use an event-driven architecture approach!

But what is an event-driven architecture?

Let me tell you! Follow me

What is event-driven architecture?

Event Driven Architecture (EDA) involves **asynchronous communication** between applications via **publishing** and **subscribing** to events over an **event broker** using a **messaging protocol**



Interesting… tell me more!

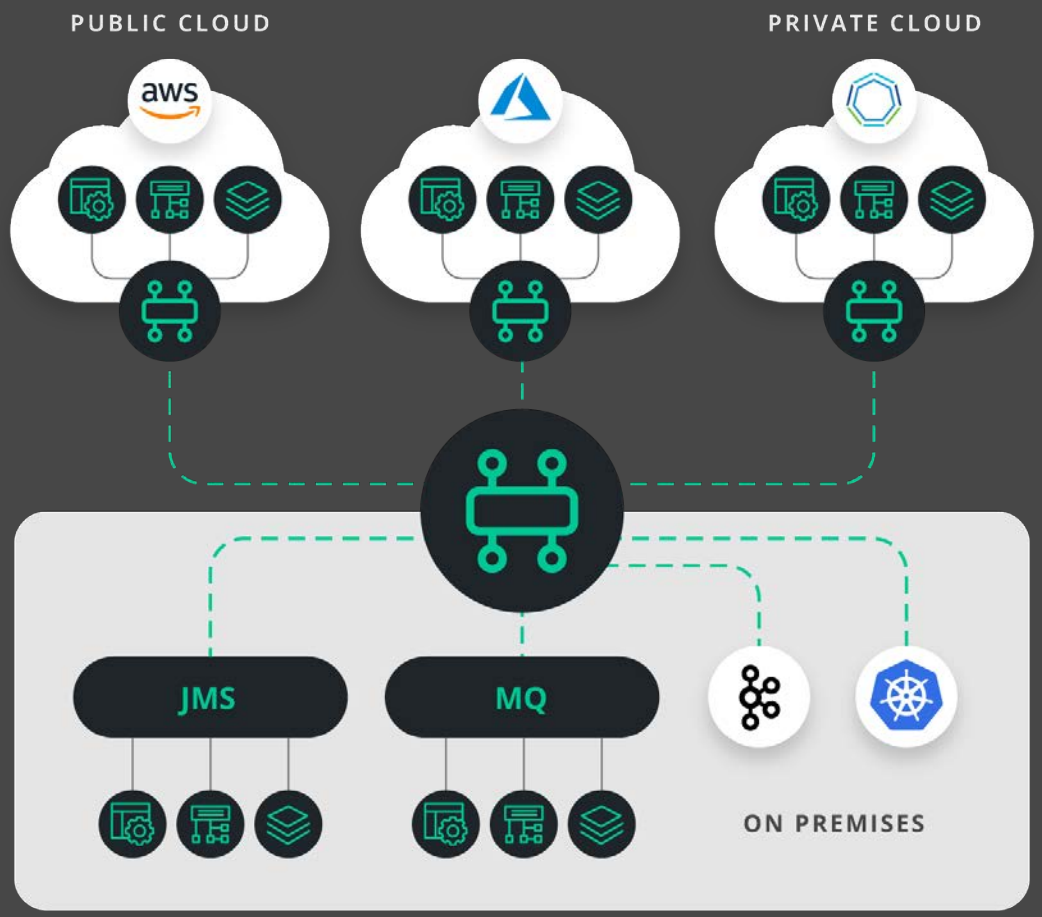
What is a messaging protocol?

Structured way for applications to exchange payloads with other applications.

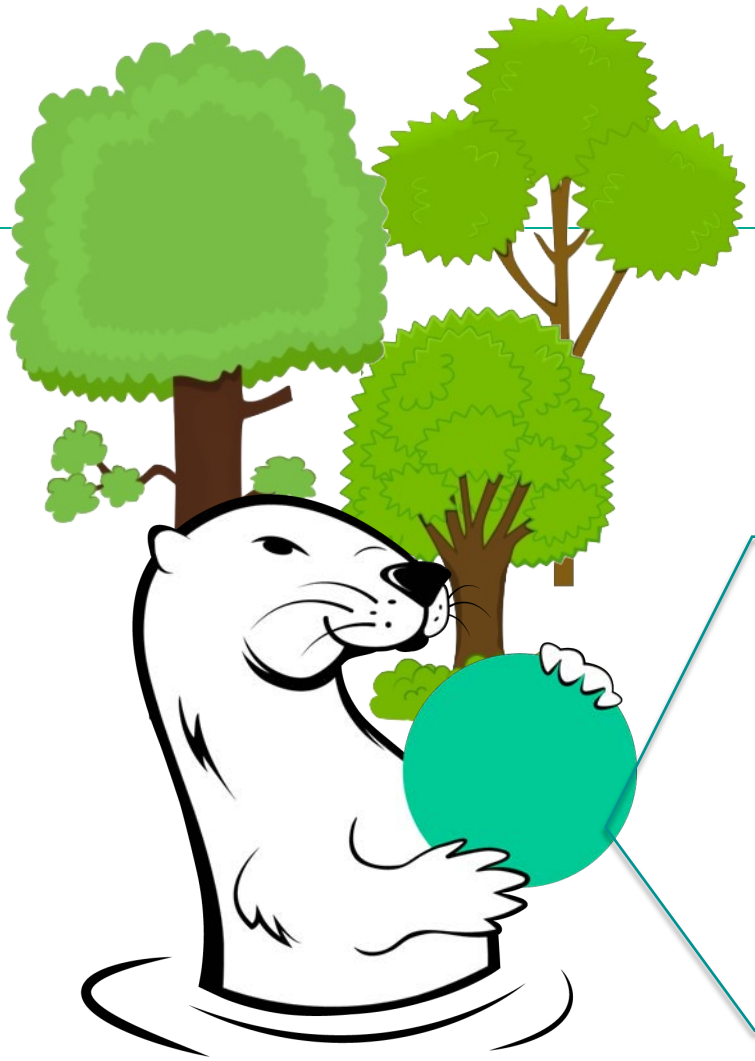
Protocols can also describe message processing, prioritization, routing, security, access control

Music to my ears!

Application Silos
Multi Protocols
Multi Languages
Different deployments



solace.



Can you give me a real world example?

Do you do online shopping?
Do you trade on the stock market?
Do you track airplanes in real-time?

Yes! Yes! Yes!

They are all examples of EDA!

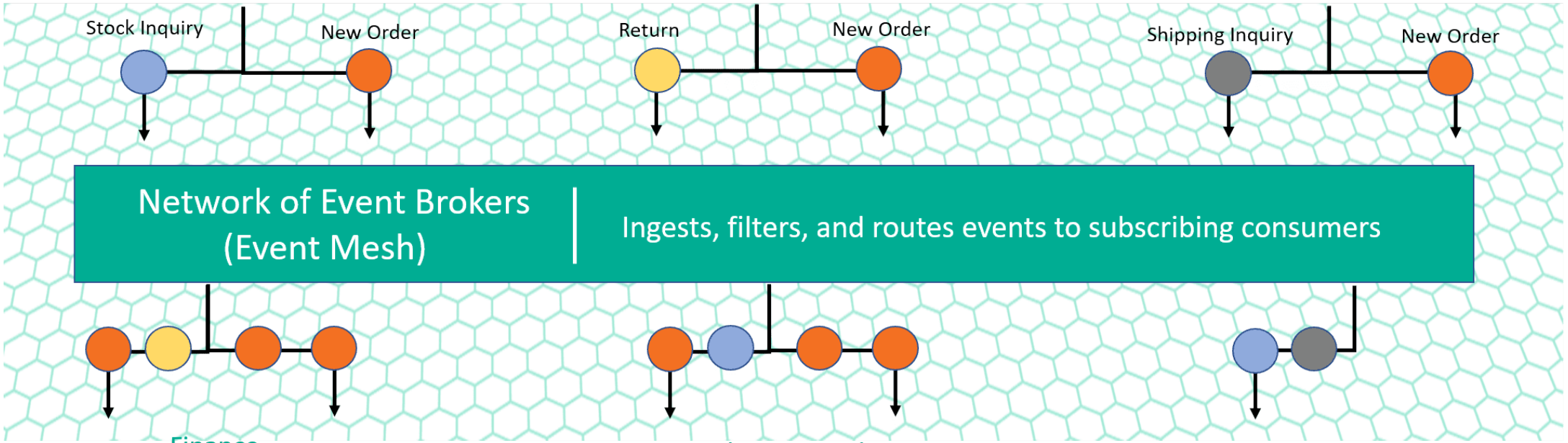
Event **producers** generate events through purchases, inquiries, and other actions

Event producers are decoupled from consumers and events are broadcast to subscribers

Mobile Application

Point-of-Sale System

eCommerce Website



Finance

Warehouse Database

Customer Service

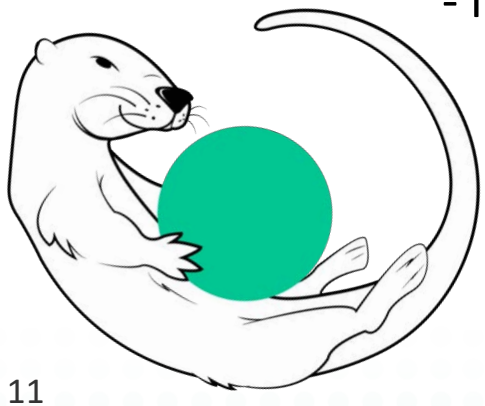
Event **consumers** subscribe to relevant events that affect their systems/actions

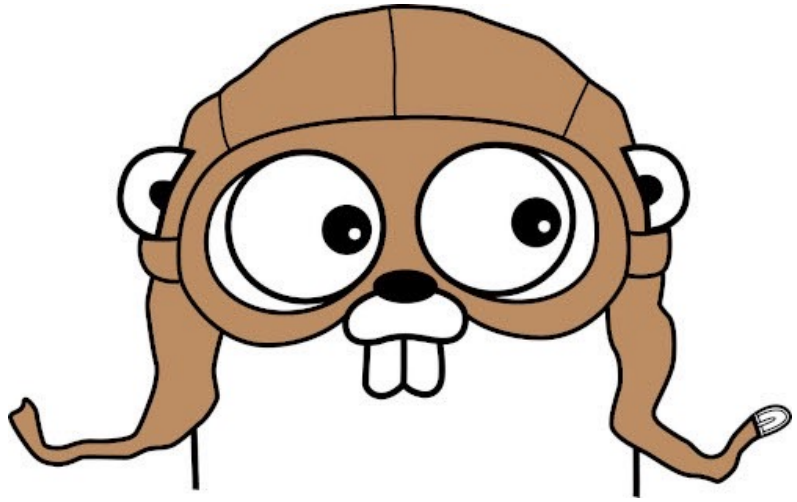
Events trigger real-time responses and can generate subsequent events

MQTT Protocol

MQTT is a simple **lightweight** messaging protocol based on the **publish-subscribe model**. It is an **OASIS standard message** protocol for IoT devices

- https://mqtt.org/





What is MQTT?

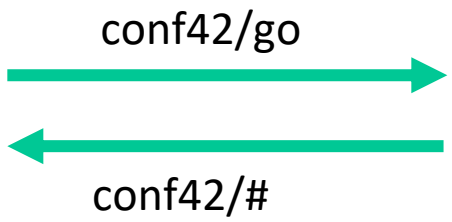
solace

Hands-on demo

Using Paho MQTT Library

- go mod init conf42Demo
- go get github.com/eclipse/paho.mqtt.golang

- Connect to MQTT broker
- Subscribe to a topic
- Publish messages
- Message Handler for received messages



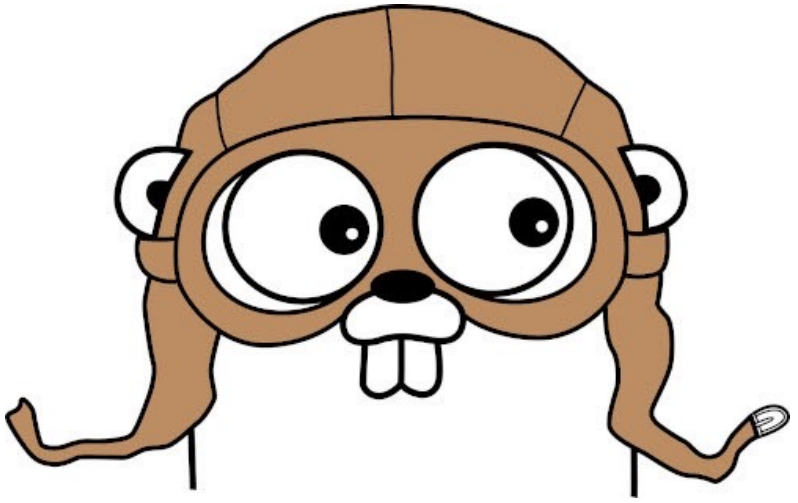
solace

Great! Now I have an idea on how to use Go with messaging protocols!

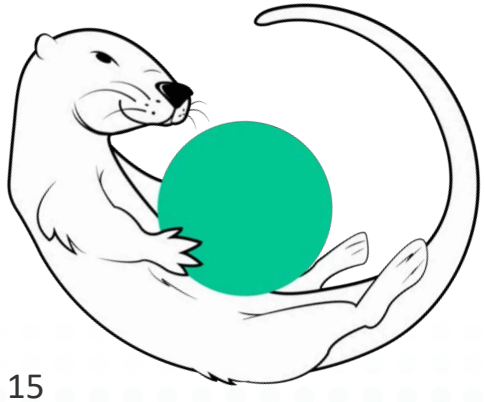
I’m still not sure how I can achieve a protocol and language agnostic approach?

That’s when you use abstraction frameworks or an advanced message broker!

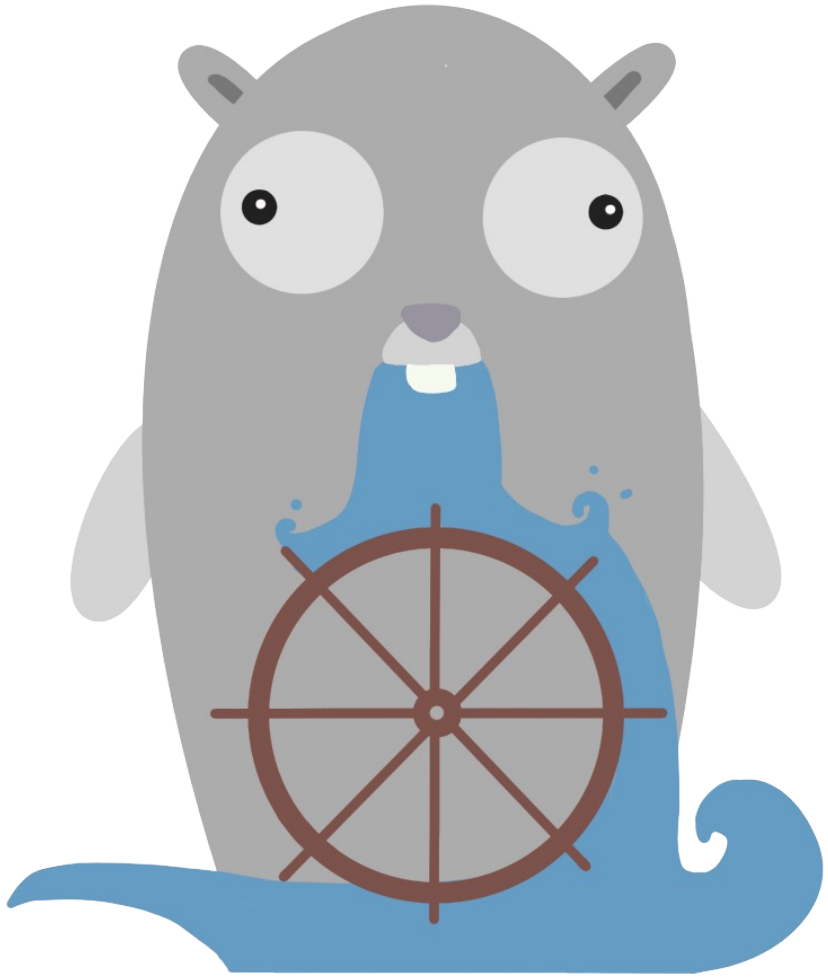
Go libraries that abstracts the underlying implementation of the event-driven technology via APIs. Let's look at an example!



What is an EDA framework?



Watermill.io



Golang Channel
A Pub/Sub implemented on Golang goroutines and channels

Kafka
A distributed streaming platform from Apache

HTTP
Call and listen to webhooks asynchronously

Google Cloud Pub/Sub
The fully-managed real-time messaging service from Google

NATS Streaming
A simple, secure and high performance open source messaging system

SQL
SQL-based Pub/Sub

RabbitMQ (AMQP)
The most widely deployed open source message broker

io.Writer/io.Reader
Pub/Sub implemented as Go stdlib's most loved interfaces

Bolt Pub/Sub
A pure Go key/value store

Firestore Pub/Sub
A scalable document database from Google

Two interfaces…

```
type Publisher interface {
    Publish(topic string, messages ...*Message) error
    Close() error
}

type Subscriber interface {
    Subscribe(ctx context.Context, topic string) (<-chan *Message, error)
    Close() error
}
```

At its core: listens for incoming messages and react to them

```go
err := publisher.Publish("conf42/go", msg)
if err != nil {
    panic(err)
}
```

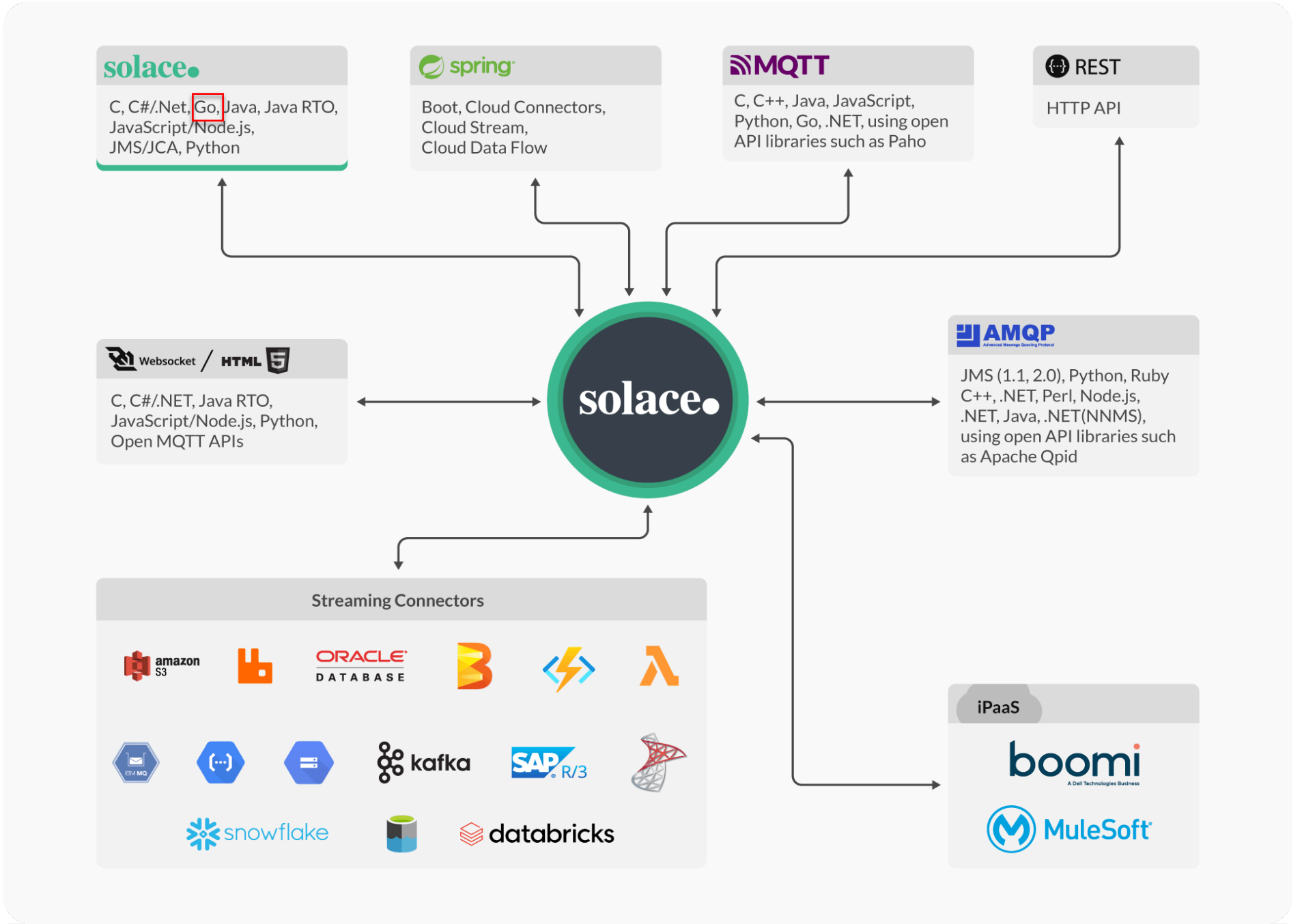
```go
messages, err := subscriber.Subscribe(ctx, conf42/go)
if err != nil {
    panic(err)
}

for msg := range messages {
    fmt.Printf("received message: %s, payload: %s\n", msg.UUID, string(msg.Payload))
    msg.Ack()
}
```

Abstracts the implementation of the underlying PubSub messaging API

```
// ...
   go process(messages)

    publisher, err := amqp.NewPublisher(amqpConfig, watermill.NewStdLogger(false, false))
    if err != nil {
        panic(err)
    }

    publishMessages(publisher)
}

func publishMessages(publisher message.Publisher) {
    for {
        msg := message.NewMessage(watermill.NewUUID(), []byte("Hello, world!"))

        if err := publisher.Publish(conf42/go, msg); err != nil {
            panic(err)
        }

        time.Sleep(time.Second)
// ...
```

Tho how can I achieve a multiprotocol and multi language EDA microservices?

Use an advanced message broker that supports multiple languages and protocols!



solace

Hands-on demo

Using Solace PubSub+ Messaing API for Go

Solace PubSub+ Messaging API for Go is available for download on April 1st

** Not an April Fools Joke!

Final Remarks

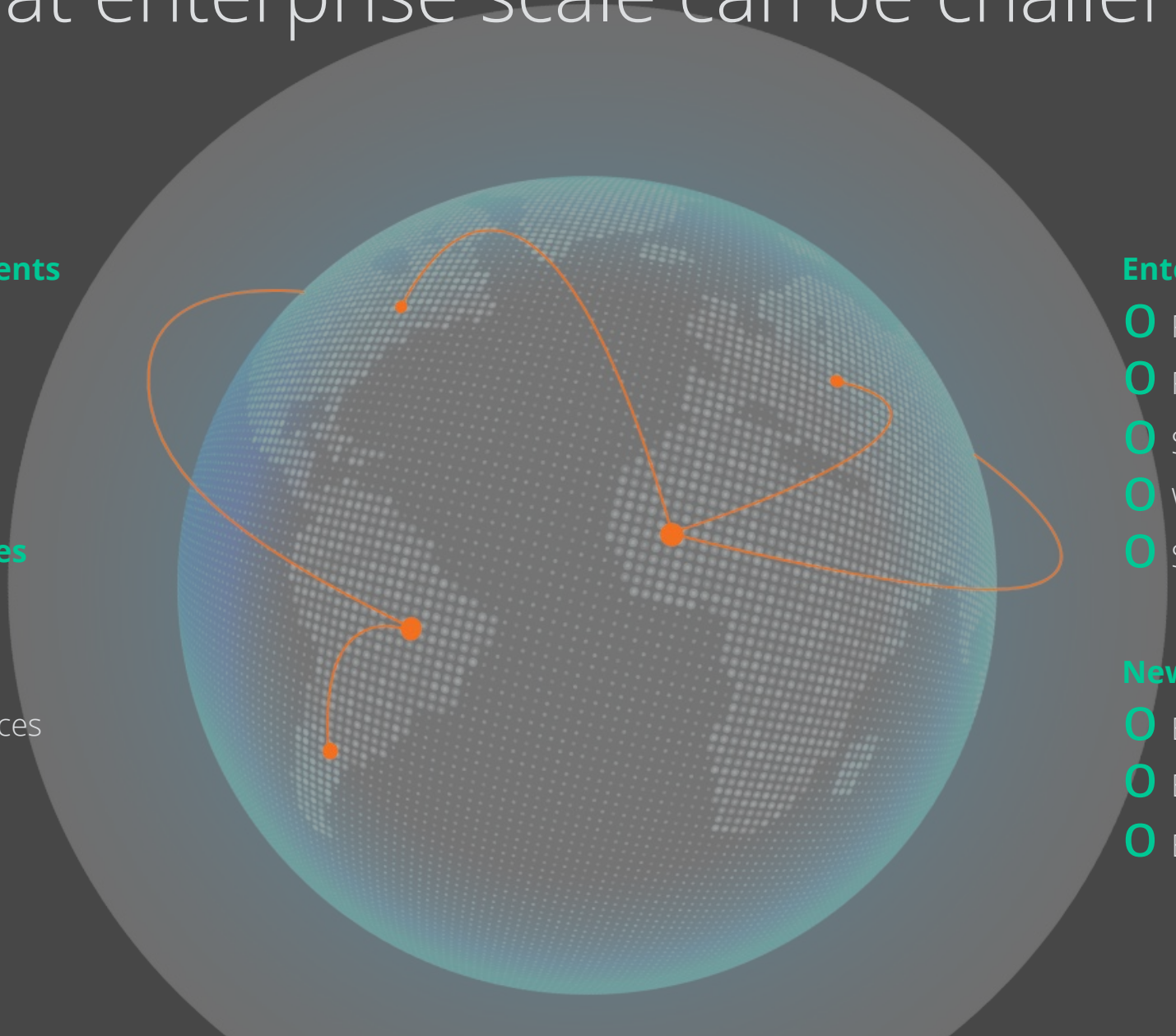
EDA at enterprise scale can be challenging

Complex Environments

- Hybrid cloud
- Multi-cloud
- Multi-geos

Diverse Technologies

- Microservices
- IoT
- Cloud native services
- iPaaS
- Legacy ESB, ERP



Enterprise Demands

- Performance
- Reliability
- Scalability
- WAN optimization
- Security & compliance

New Capabilities Required

- Event design
- Event discovery
- Event governance

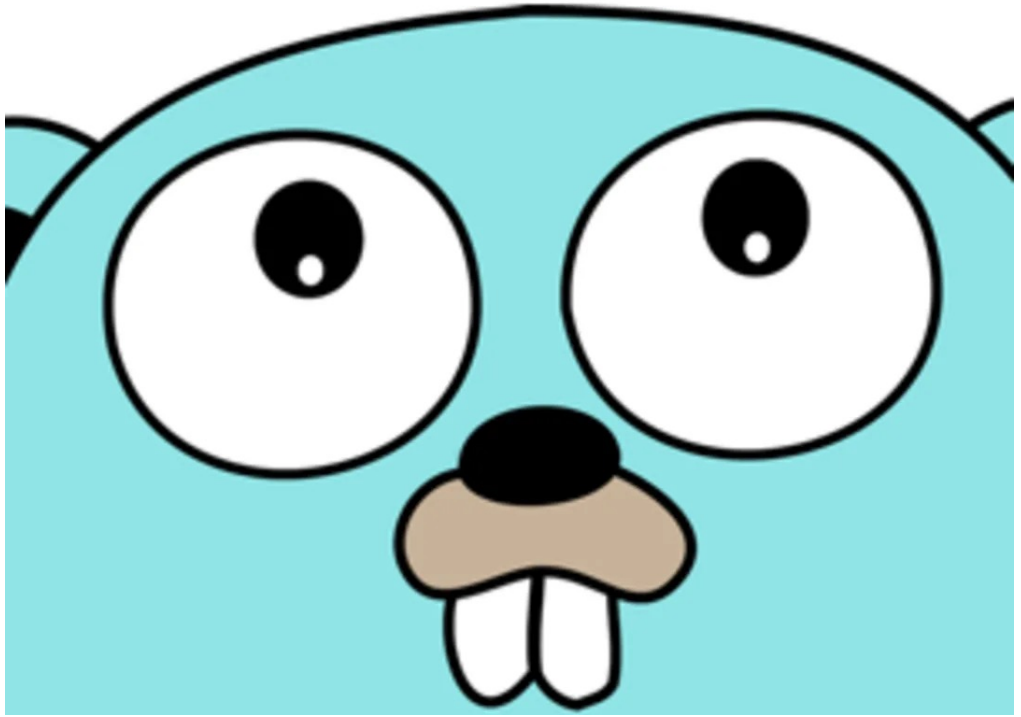
solace.

Live Coding Event
April 12. Mark your calandars!
Youtube – Linkedin

Solace.community
Easter egg hunt!

@SolaceDevs
@TweetTamimi

What next?!



https://edasummit.com/