Bruno Bossola

MSNOS

a spontaneus network operating system for microservice architectures

bbossola@gmail.com

bruno.bossola@workshare.com

Hey mate, who are you?

- Developer since 1988
- XP Coach 2000+
- Co-founder and coordinator of JUG Torino
- Java Champion since 2005
- VP of Engineering @Workshare.com

Agenda

- Why microservices?
- What are microservices?
- What are the issues?
- MSNOS to the rescue!
- demo, demo, demo!
-

msnos.io is open source! (MIT license)

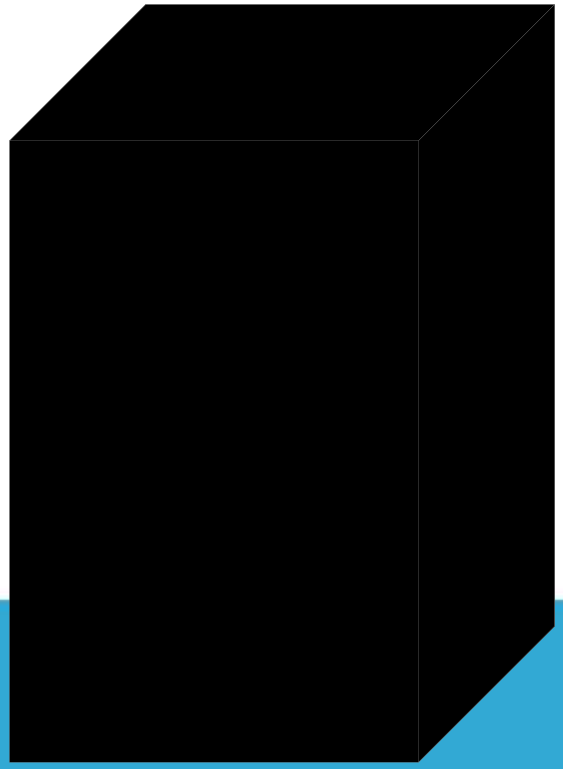
Agenda

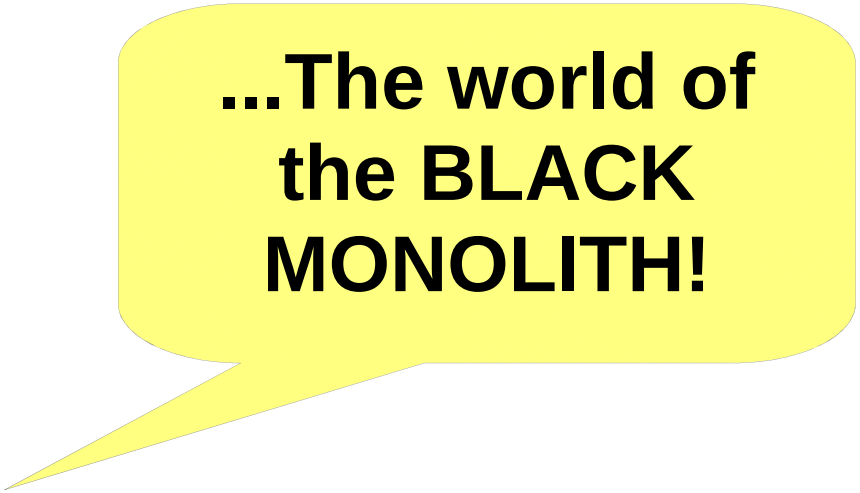
- **Why microservices?**
- What are microservices?
- What are the issues?
- MSNOS to the rescue!
- demo, demo, demo!
-

msnos.io is open source! (MIT license)

Why microservices?

- if you are asking this question you are probably living in this world:





Why avoid the monolith?

- the monolith usually provides HTML pages
  - these days we want APIs and single page apps
- the monolith scares developers!
  - the code base is massive
- the monolith is scary to deploy
  - everytime you have to deploy everything!
  - it's a stop-the-world moment

What's the issue here?

- the monolith code bloats your IDE
- it locks you into one language/platform
- adding a new API requires a congress
- on deploy the whole database may have to be migrated for a very small change
- running the tests can take hours
- it's boring to develop on that s**t

Agenda

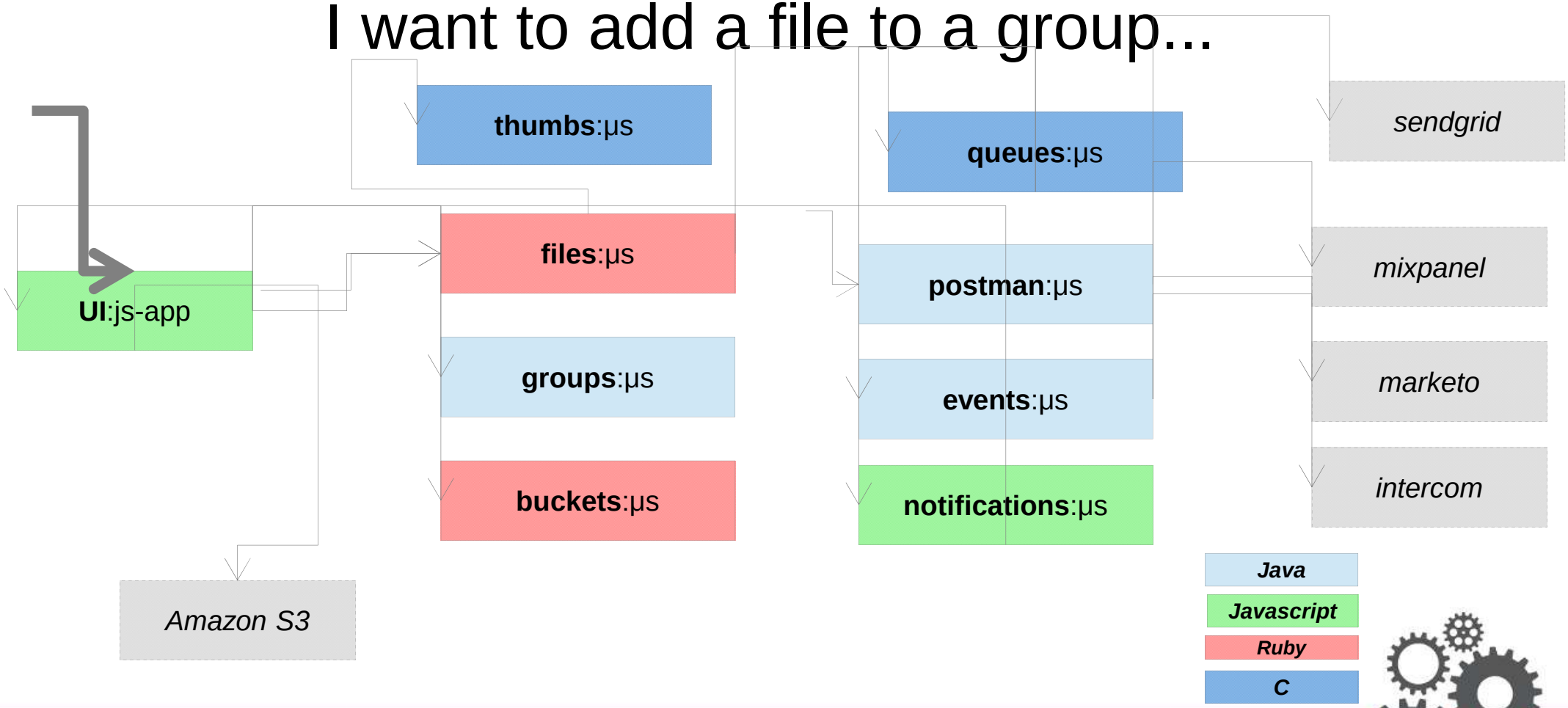
- Why microservices?
- **What are microservices?**
- What are the issues?
- MSNOS to the rescue!
- demo, demo, demo!
-

msnos.io is open source! (MIT license)

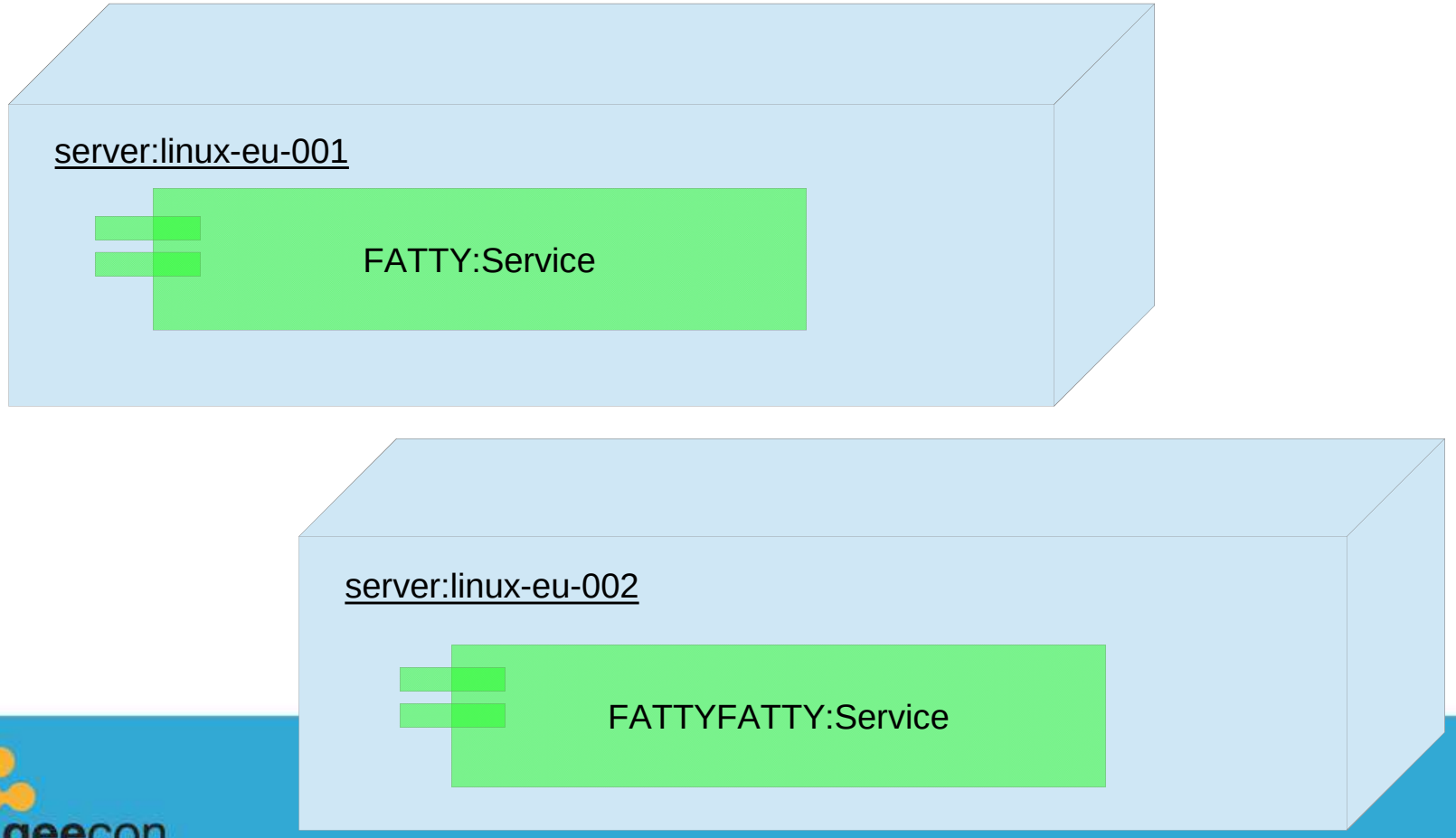
What are microservices?

- independent processes communicating with each other using language-agnostic APIs
- small, highly decoupled and focus on doing a small task
- can be written in any programming language
  - we do use Ruby, C#. Java and Javascript
- micro means small, very!
  - we run 20 of them on one single machine

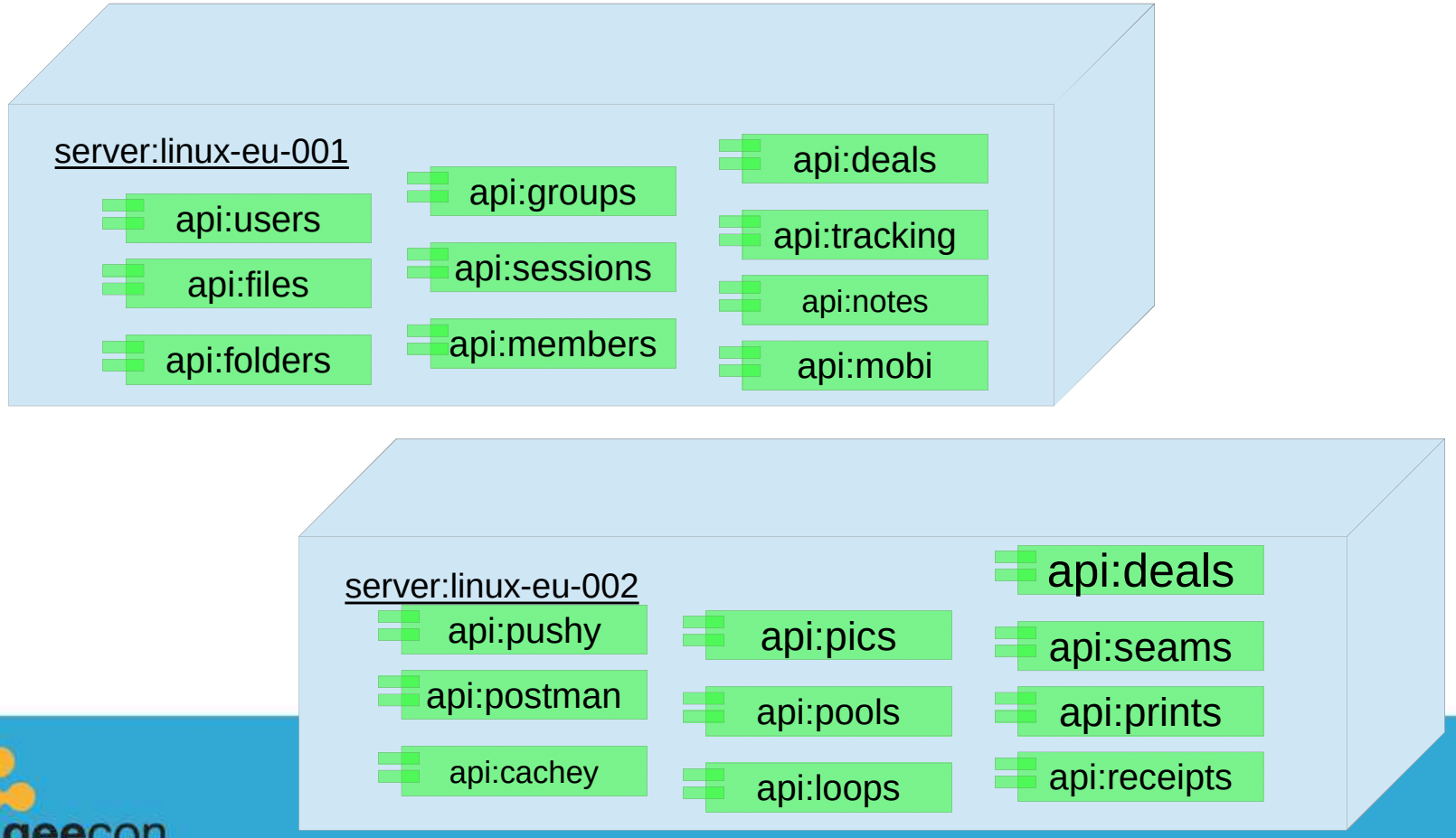
I want to add a file to a group...



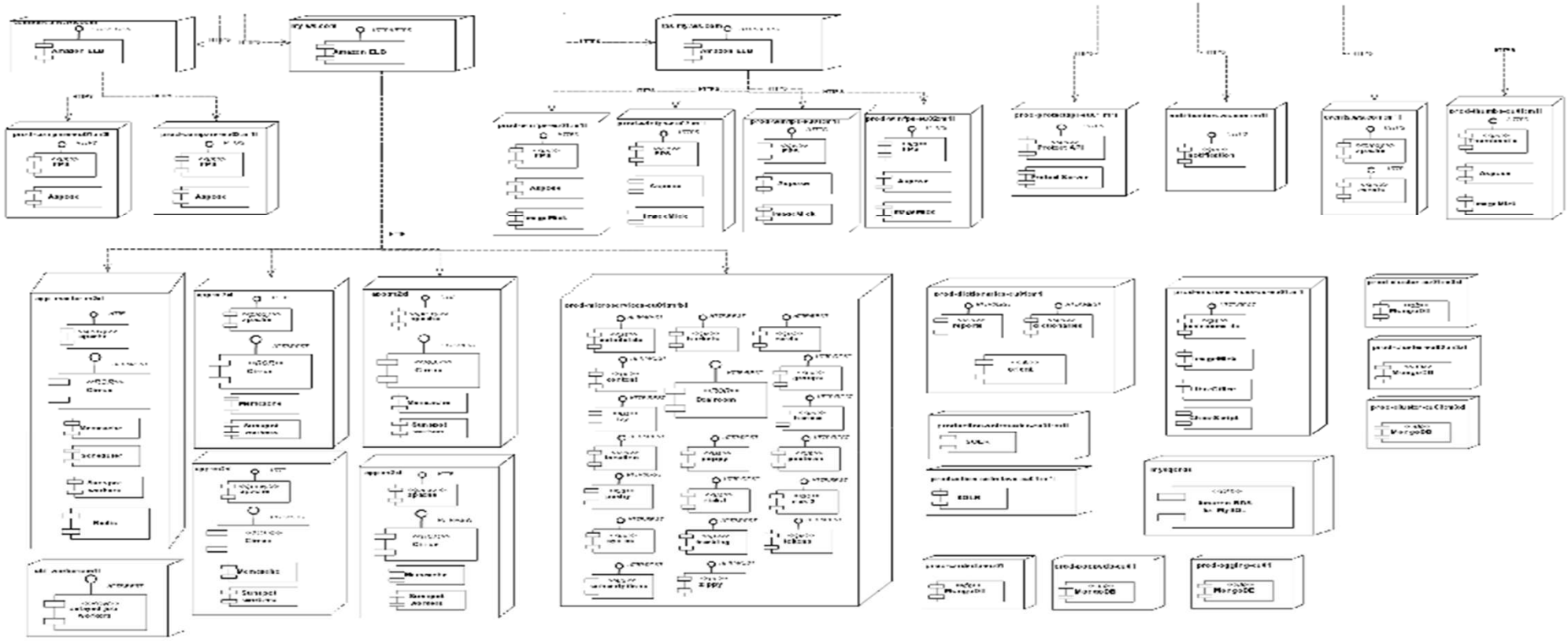
Microservices??



Microservices!



Microservices!



Many advantages!

- highly cohesive, loosely coupled
- clean api contracts and SOC
- very easy to write and replace
- self-contained, very easy to deploy
- scaling can be very focused
- small releases, less risky to deploy
- separate databases for separate migrations

Demo!

- a simple location service that translates IP in geo-location
- a more complex service that serves files and folders

What is the context?

- you use a microservice architecture
- microservice are distributed across the world
- each microservice provide different APIs
- several instances of the same microservices are available
- they need to talk to each other to fullfill a task

Cons!

- Lots of them!
- Every microservice is doing one thing, so a lot of collaboration is required to perform a task
- Scaling requires considerable effort
- Frequent duplicated configuration
- Proxying for external apps is not easy
- management overhead (more moving parts)

Few issues ahead!

- Any insight?

Agenda

- Why microservices?
- What are microservices?
- **What are the issues?**
- MSNOS to the rescue!
- demo, demo, demo!
-

msnos.io is open source! (MIT license)

Problem #1: how do I find an API?

- **I need to use an API exposed by another microservice: how do I find the endpoint?**
  - I am the *Session Tracking microservice* and I need to translate an IP to a geo location
  - I heard that a *Location microservice* does this
  - How do I know which endpoint I need to call?
    - where is it?
    - what's the nearest?
    - what's the less loaded?

Issue #2: how do I expose my APIs?

- **Hey, I am online and I want to publish my APIs to everybody: how can I do that?**
  - I am a new service, I am a new instance of an existing service
  - I am ready to receive calls!!
  - How do the other microservices get to know about me? :(

Issue #3: how do I report issues?

- **OMG I am not able to connect to my database anymore!!!**
  - I need to alert the other microservices around that I cannot accept calls at the moment!
  - If they knew this, they could call another instance of me that actually works...
  - How do the other microservices get to know about me being faulty?

Issue #4: how do I call from clients?

- **I am a single page app / mobile app and I want to use some backend APIs**
  - How can I make sure the APIs I want to use are proxied and available on the web?
  - How can I get load the best possible service instance to use?
  - How can I avoid connecting to a “bad” API and be sure that I will always use a working endpoint?

In an ideal world...

- new microservices APIs are automatically seen and used by any other microservice
- unhealthy microservices do not receive calls
- the load is distributed trough advanced strategies (load, location, availability...)
- there are no single point of failures
- add/remove of microservices is transparent
- “public” APIs are magically proxied to the web

Agenda

- Why microservices?
- What are microservices?
- What are the issues?
- **MSNOS to the rescue!**
- demo, demo, demo!
-

msnos.io is open source! (MIT license)

MSNOS to the rescue!

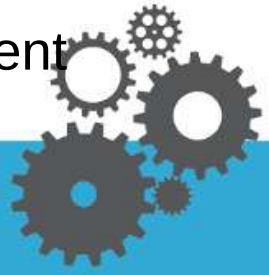
- A network operating system for architectures based on microservices
  - self discovery
  - communication
  - load balancing
  - routing
  - web proxying
  - *...and more :)*

What is NOT

- A replacement for a deployment mechanism
  - still your job to deploy and run a microservice
  - docker does a good job I heard :)
- A replacement for any configuration
  - still your job to configure enough to start
  - it provides a shared key-value store tough :)
- An infinite scaling solution
  - works well with a couple of hundreds
  - we are not in the thousands (yet)

Why do we need this? [1/2]

- µservices should automatically discover themselves
  - any service can find any api with zero effort
- µservices should be able to scale on demand
  - work should be handled in a balanced manner
  - you should be able to spinoff a new instance so that it can transparently join the cloud and start working
- µservices should work across network boundaries
- the cloud of services should be deployable across two or more different physical clouds



Why do we need this? [2/2]

- µservices api should be exposed automagically to the external world
- µservices public APIs should be proxied transparently
- you should not need a VM for

How do I use it? [1/2]

- msnos is a native library you link into your app

```
cloud = new Microcloud(params.uuid());

service = new Microservice(params.name());
service.join(cloud);

 RestApi[] { apis = new RestApi[] {
    new RestApi("/hello", URI_GREET, port),
    new RestApi("/wassup", URI_WASSUP, port),
    new RestApi("/checkme", URI_HEALTH, port).asHealthCheck(),
    new RestApi("/msnos", URI_MSNOS, port, Type.MSNOS_HTTP),
 };
service.publish(apis);
```

Create a cloud, a service, let the service join the cloud

Declare and publish your service APIs

What do I get?

- Out of the box fast HTTP reverse proxy
  - fast, non blocking, based on Netty
  - exposes your public APIs to the web
- Out of the box WWW gateway
  - helps discovery across network boundaries
- A simple sms-style messaging system
- Free live visualization tool for monitoring and basic management of your cloud
  - available now at msnos.io

Demo!

Future

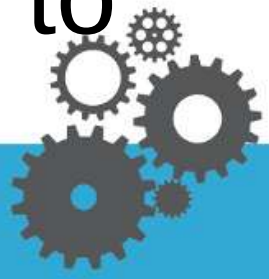
- Release 1.0 (1m)
  - core, proxy, www gateway
- Native Ruby support (2w)
- Support for Azure (2w)
- Completion of distributed configuration (1m)
- Native .NET support (1m)
- STUN protocol support - RFC 5389 (tbc)

Q&A

- Msnos code:
  - https://github.com/workshare/ms-nos
  - msnos.io
- Myself:
  - https://about.me/bbossola / @bbossola
- Company
  - https://www.workshare.com

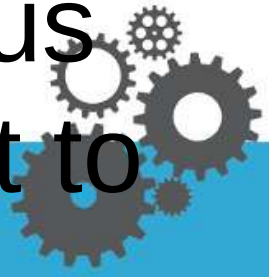
Anticipated Q&A

- **Q.** Why did you not release a final version yet?
- **A.** Because, even if we are using it in production, we still consider it not final and we prefer to wait in order to release top-quality code



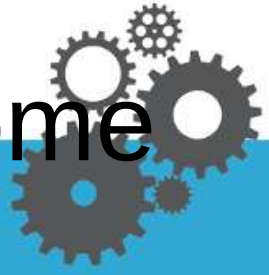
Anticipated Q&A

- **Q.** Why don't you simply use a dynamic DNS to manage discovery??
- **A.** One of my favourites :)
  - a DNS entry does not provide us enough granularity, as we want to talk about API endpoints, not



Anticipated Q&A

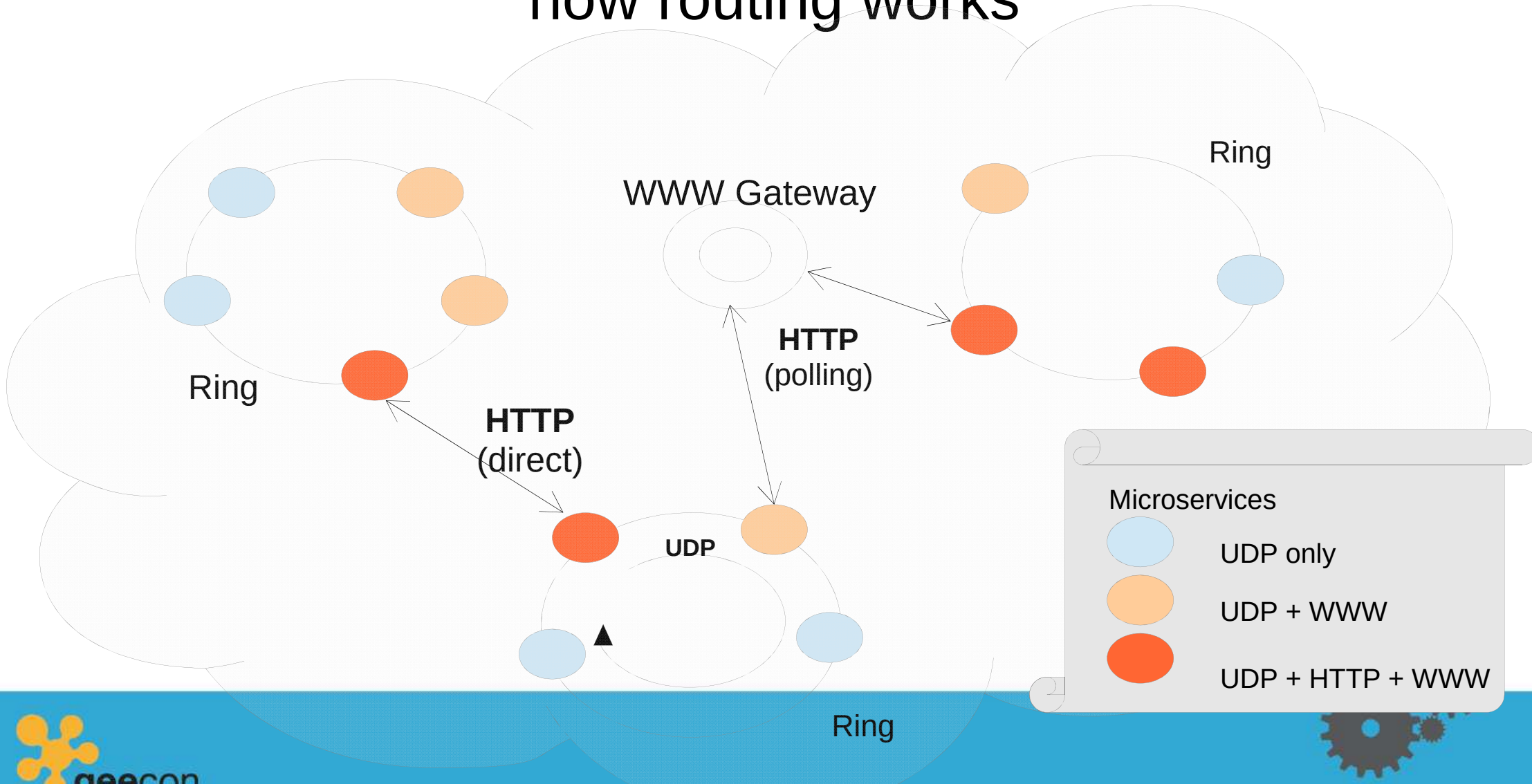
- **Q.** Why don't you use ${other-framework}?
- **A.** Well, there are different answer for each of them, and such frameworks are becoming more popular by the minute! Anyway, some common issues we found so far:



Tech-stuff: features

- full messaging platform
  - broadcast of messages
  - point-to-point messages
  - reliable messages
- self discovery of other services
- publish and locate an API in the cloud
- debug in the cloud
- automatic reverse proxying of public APIs
- sharing configuration within the cloud *(wip)*
- events publish/subscribe *(future)*

how routing works



how proxy works