Jumpstart into Virtual actors with Orleans

Authors

Alexey Shcherbak

Cloud Architect

DRAWBOARD

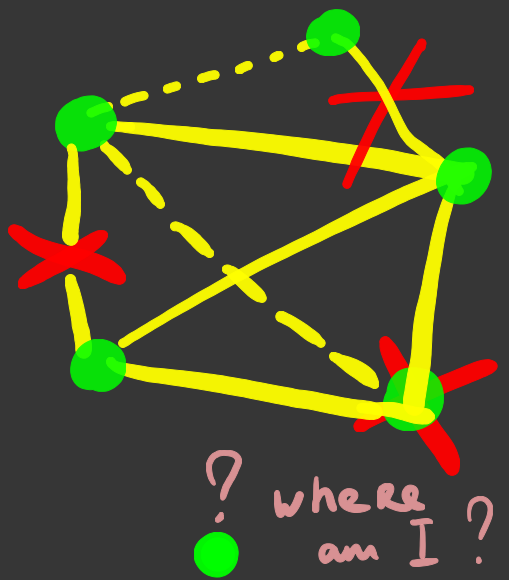
Reuben Bond

Microsoft MVP,

CEO & Engineer

Typical day in distributed system:

- Scaling is hard
- Concurrency is very hard ✓
- Failure is ~~coming~~ inevitable



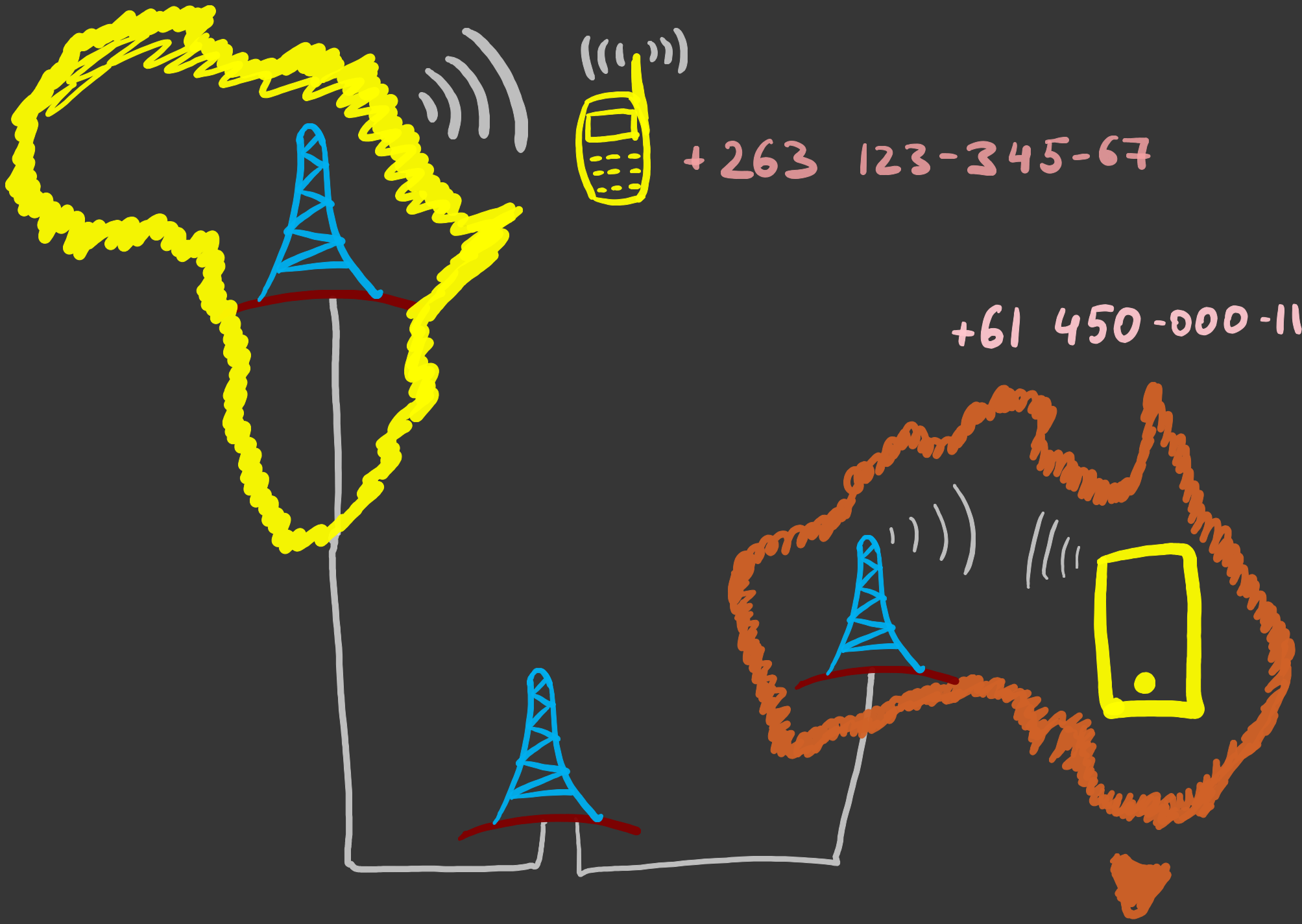
Orleans to the rescue

- Non-blocking performance
- Location transparency
- Single-threaded grain execution
- Battle-tested:
  - Halo 4 & Skype
  - Extreme Computing Group
  - MS Research

Conceptual model, p.1

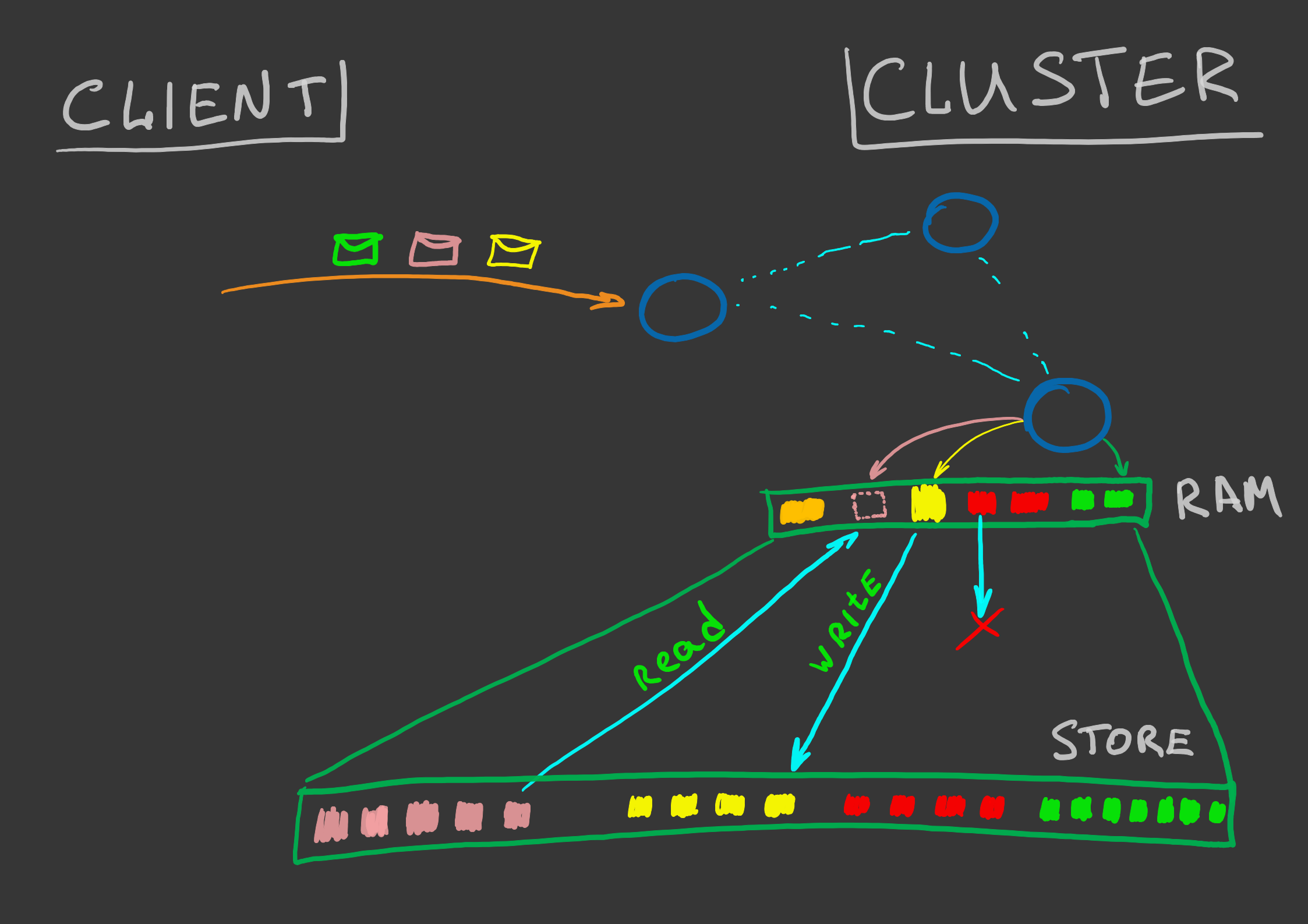
- Server (in cluster) aka silo
  - Auto-discovery
  - Peer-to-peer monitoring
- Virtual actor aka grain
  - Unit of state + behaviour
  - Grain full address: Type + ID ← LOCATION TRANSPARENT

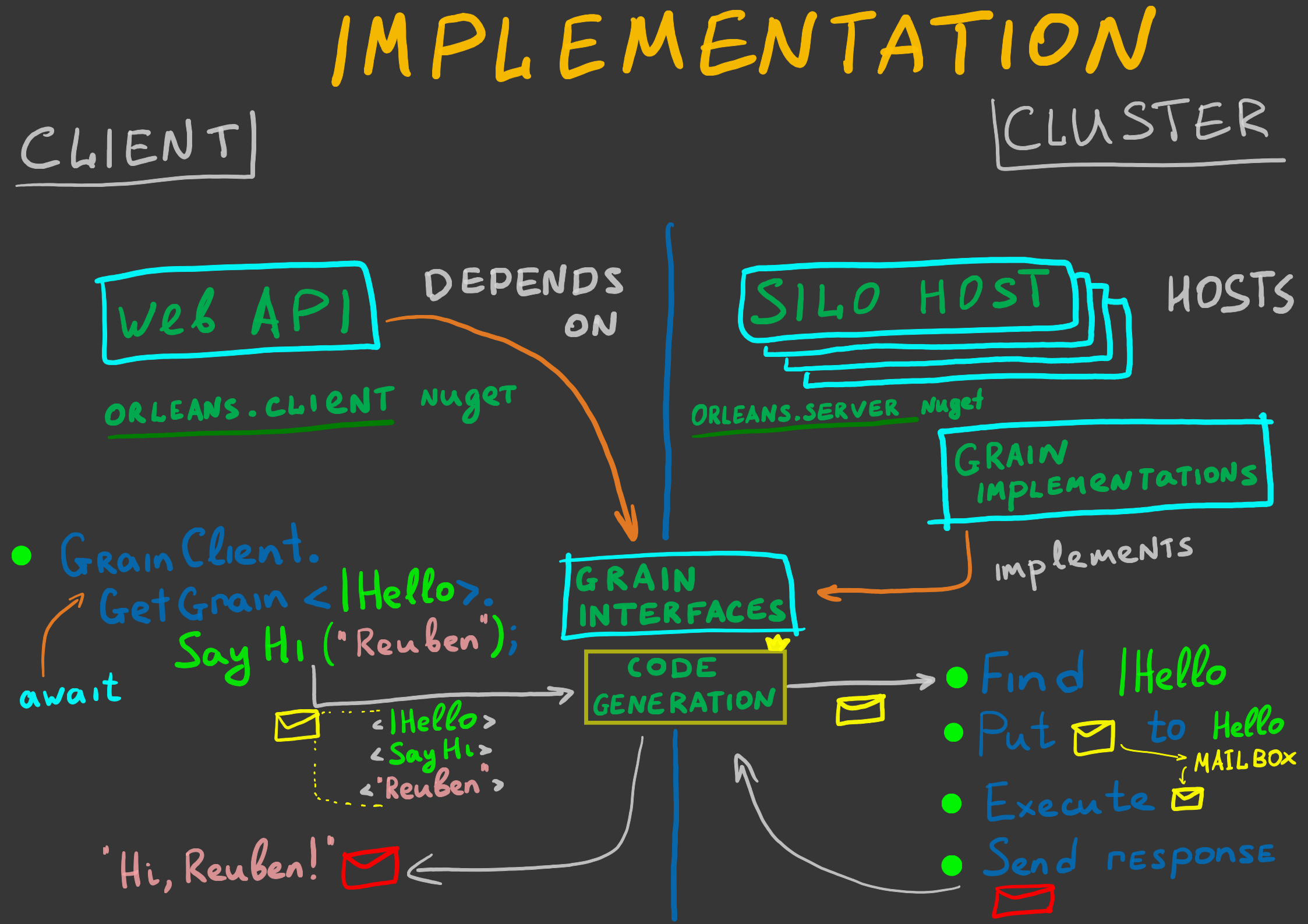
FARM GLOSSARY

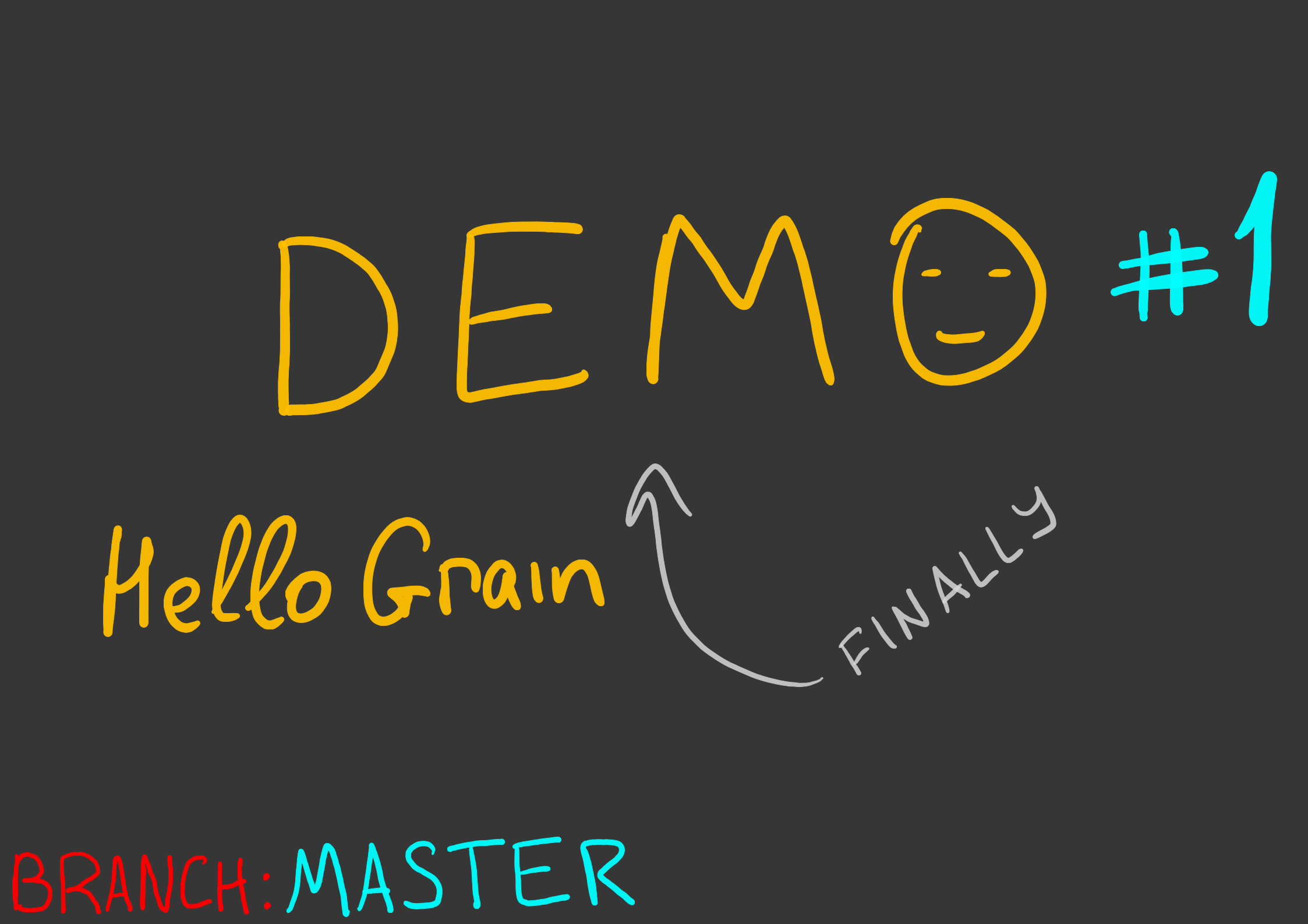

+263 123-345-67
+61 450-000-111

Conceptual model, p. 2

- Grain has plain C# interface
  - Simple programming model
  - Async/await all the things
- Runtime is doing all plumbing
  - Grain lifecycle management
  - DI, Routing, Messaging


CLIENT
CLUSTER
RAM
Read
WRITE
STORE


IMPLEMENTATION
CLIENT
CLUSTER
Web API
DEPENDS ON
SILO HOST
HOSTS
ORLEANS.CLIENT Nuget
ORLEANS.SERVER Nuget
GRAIN IMPLEMENTATIONS
implements
GRAIN INTERFACES
CODE GENERATION
GrainClient.
GetGrain<IHello>.
SayHi("Reuben");
await
<IHello>
<SayHi>
<"Reuben">
"Hi, Reuben!"
Find IHello
Put to Hello
MAILBOX
Execute
Send response


DEMO #1
Hello Grain
FINALLY
BRANCH: MASTER

STORAGE

- Simple declarative persistence
  - State object to store
  - Plug-in Provider you need || WRITE OWN ONE
  - ReadState, WriteState, ClearState
- Optional, quick way to get started
- Handy for unit-testing

EXISTING PROVIDERS

- In-Memory ← Unit tests BFF 4ever
- Azure Table || Blob
- SQL Storage
- Document DB
- Redis
- Raven DB
- <Something DB> written by you!

...

Observers

- Simple Pub/Sub model
- Server → Client messaging
- Works with almost any C# Interface
- May be deprecated

Don't worry, Virtual Streams have you covered

DEMO 2

STORAGE && OBSERVERS

BRANCH: Chat Demo


QUESTIONS

Jumpstart into Virtual actors with Orleans

Part I

QUICK RECAP

- Silo and Grains ← Animal Farm
- Distributed, scalable, non-blocking, location transparent, ... more buzz words ?
- Storage Providers
- Client-side Observers

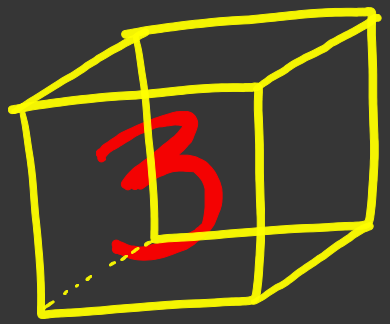
TIMERS & REMINDERS

- TIMER = .NET Timer + 1-threaded
  - Does NOT survive grain deactivation
  - Typical usage: polling, cache updates
  - Frequent, but unimportant

TIMERS & REMINDERS

- Reminder = Scheduled INVOCATION
  - Stateful = survives restarts
  - Invoked by Runtime
  - Typical usage: delayed (up to 49 days) method calls

DEMO

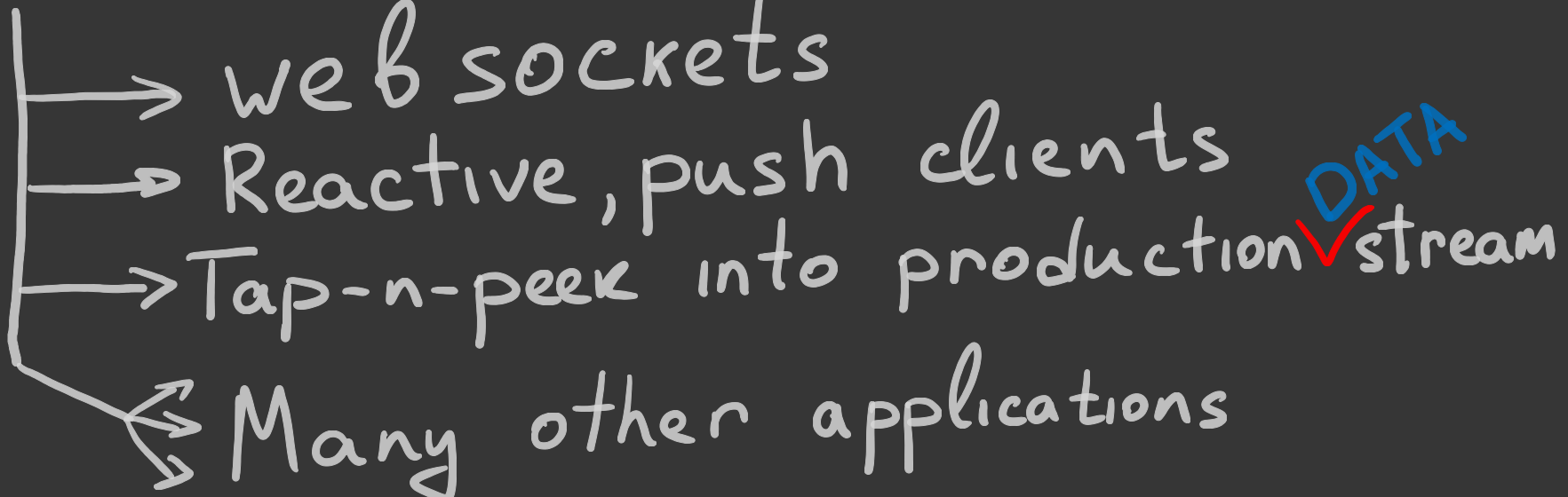


TIMERS & REMINDERS

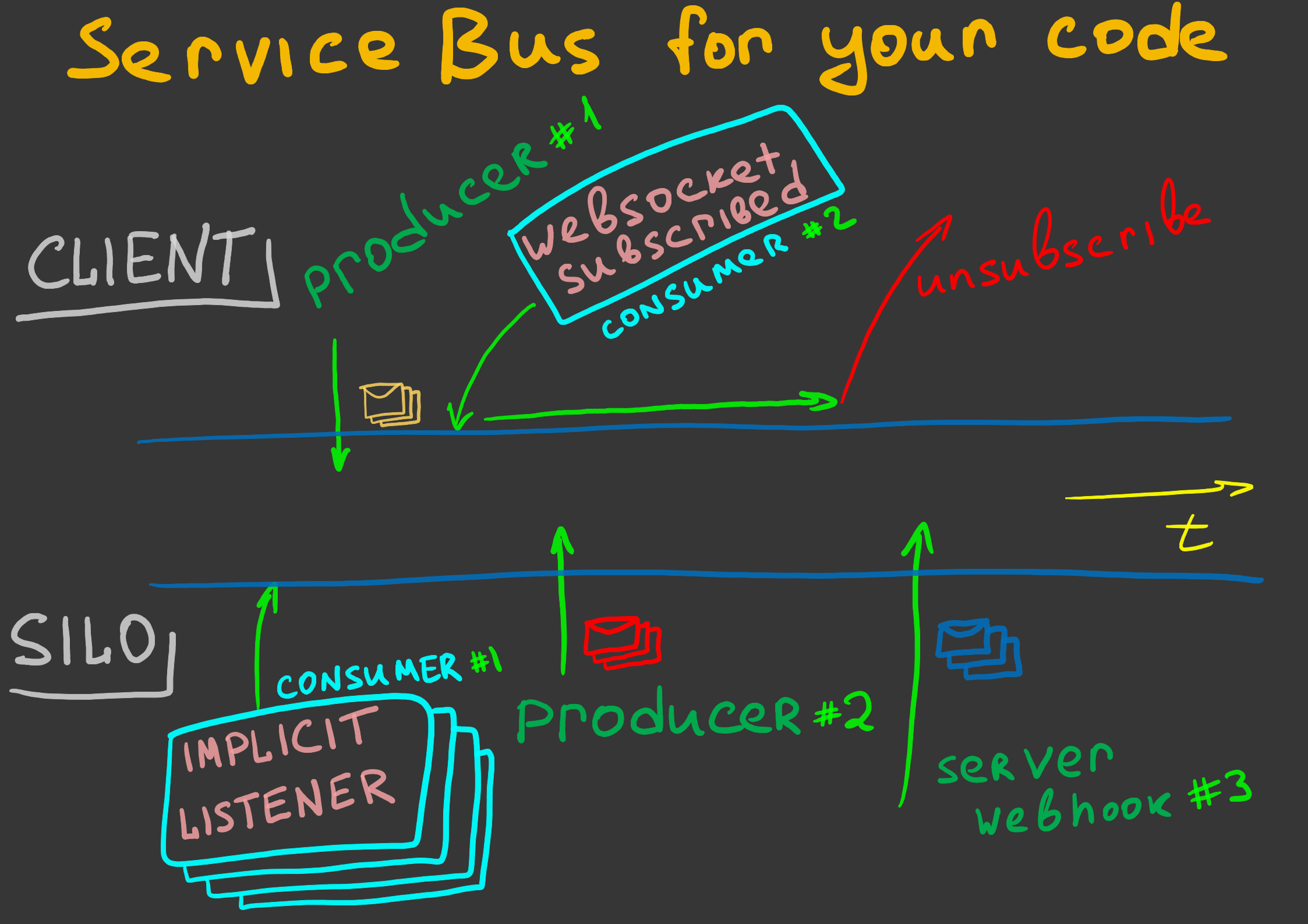
BRANCH: EggTimerDemo

VIRTUAL STREAMS

- Decouple producers from consumers
- Implicit and explicit subscription
- Client-side subscribers



Service Bus for your code



DEMO 2^2

VIRTUAL STREAMS

BRANCH: Chat Stream Demo

PATTERNS

- Pulling Client
- Fan-In && Fan-Out
- Re-streamer
- More at GitHub/OrleansContrib

WHERE TO GO FROM HERE

- Choose your hosting flavour
  - Console / Windows service
  - Azure Cloud Services
  - Service Fabric
  - Future: AWS, Docker

WHERE TO GO FROM HERE

- Dependency Injection
- Pluggable Telemetry providers
- .NET Core „COMING SOON"
- Your awesome #pullreq

JOIN US

https://gitter.im/dotnet/orleans <- Come and say "Hi" :)

https://github.com/dotnet/orleans <- Send PRs here

https://github.com/OrleansContrib <- Or here ;)

Every help counts

THANK YOU
for being here

FIND US

- @CENTUR
- github.com/Centur
- @REUBENBOND
- github.