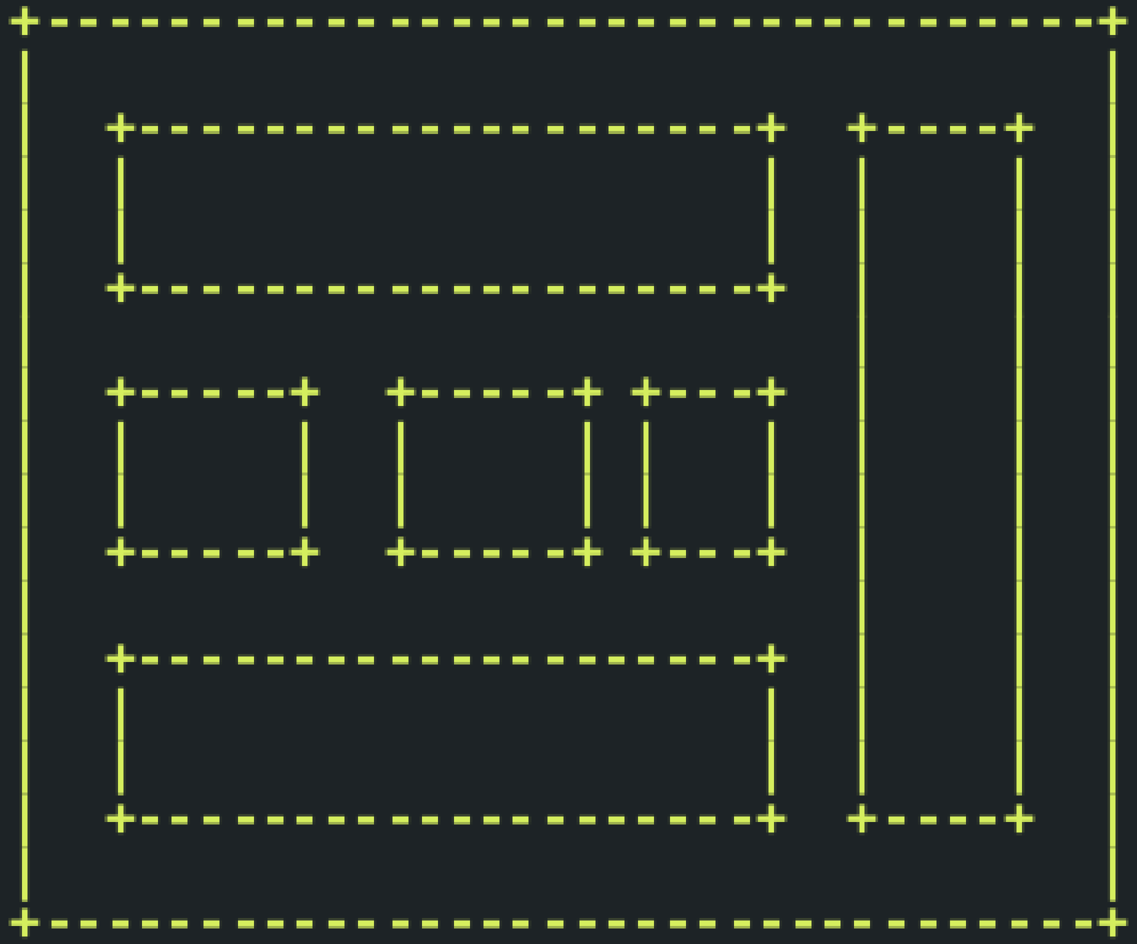
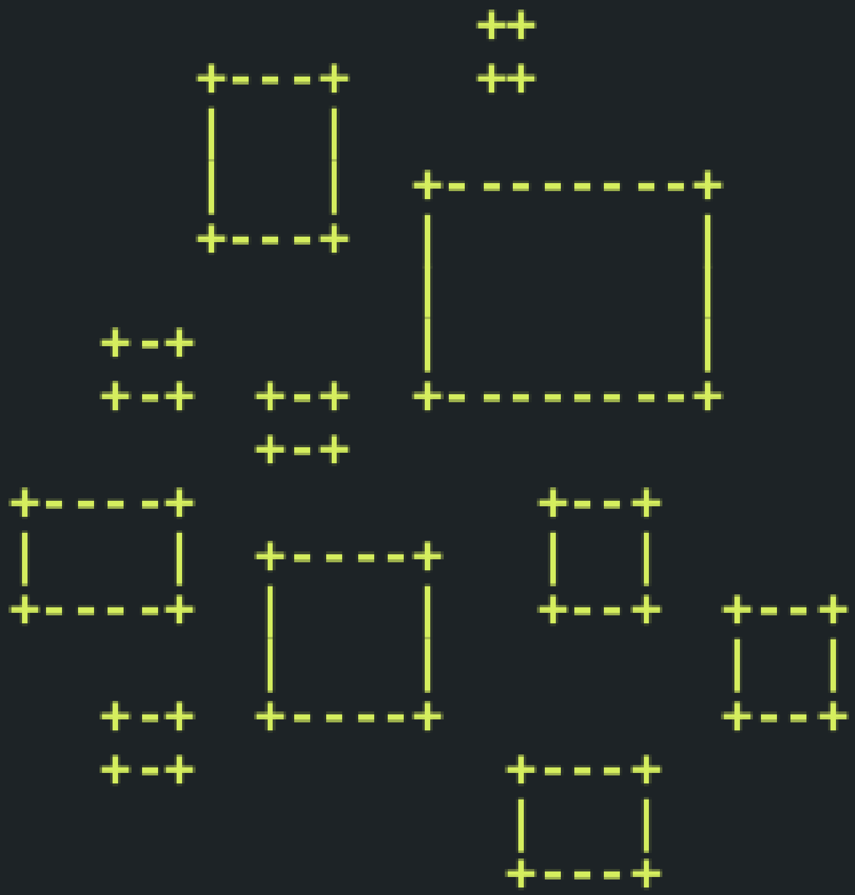
MICROSERVICES,
MICROSERVICES EVERYWHERE!
imgflip.com

Micro Service is an **architectural concept** that aims to decouple a solution by decomposing functionality into discrete services

with communication over lightweight mechanisms, often an HTTP API


Monolythic / layered


Microservice

My personal definition

My personal definition

A small problem domain Bounded Context might be the thing

My personal definition

A small problem domain Bounded Context might be the thing

Built and deployed by itself standalone and isolated

My personal definition

A small problem domain Bounded Context might be the thing

Built and deployed by itself standalone and isolated

Runs in its own process

My personal definition

A small problem domain Bounded Context might be the thing

Built and deployed by itself standalone and isolated

Runs in its own process

Integrates via well-known interfaces

While HTTP isn’t always the best answer, it’s a damn fine first guess

My personal definition

A small problem domain Bounded Context might be the thing

Built and deployed by itself standalone and isolated

Runs in its own process

Integrates via well-known interfaces

While HTTP isn't always the best answer, it's a damn fine first guess

Owns its own data ultimate goal

Some problems with monolythic architecture

Some problems with monolythic architecture

Even when layered, hidden coupling

Some problems with monolythic architecture

Even when layered, hidden coupling

Single runtime, allows in memory calls

Some problems with monolythic architecture

Even when layered, hidden coupling

Single runtime, allows in memory calls

FUD: if it works don’t fix it don’t touch it

Some problems with monolythic architecture

Even when layered, hidden coupling
Single runtime, allows in memory calls
FUD: if it works don’t fix it don’t touch it
Good diagrams not always make it to good code

> “Problems of developing software derive from essential complexity and its nonlinear increases with size; leading to difficulty of communication among team members, cost overruns, schedule delays.
>
> The Mythical Man-Month
>
> — Fred Brooks

TAKEAWAY

Separate things that change with a

different pace

TAKEAWAY

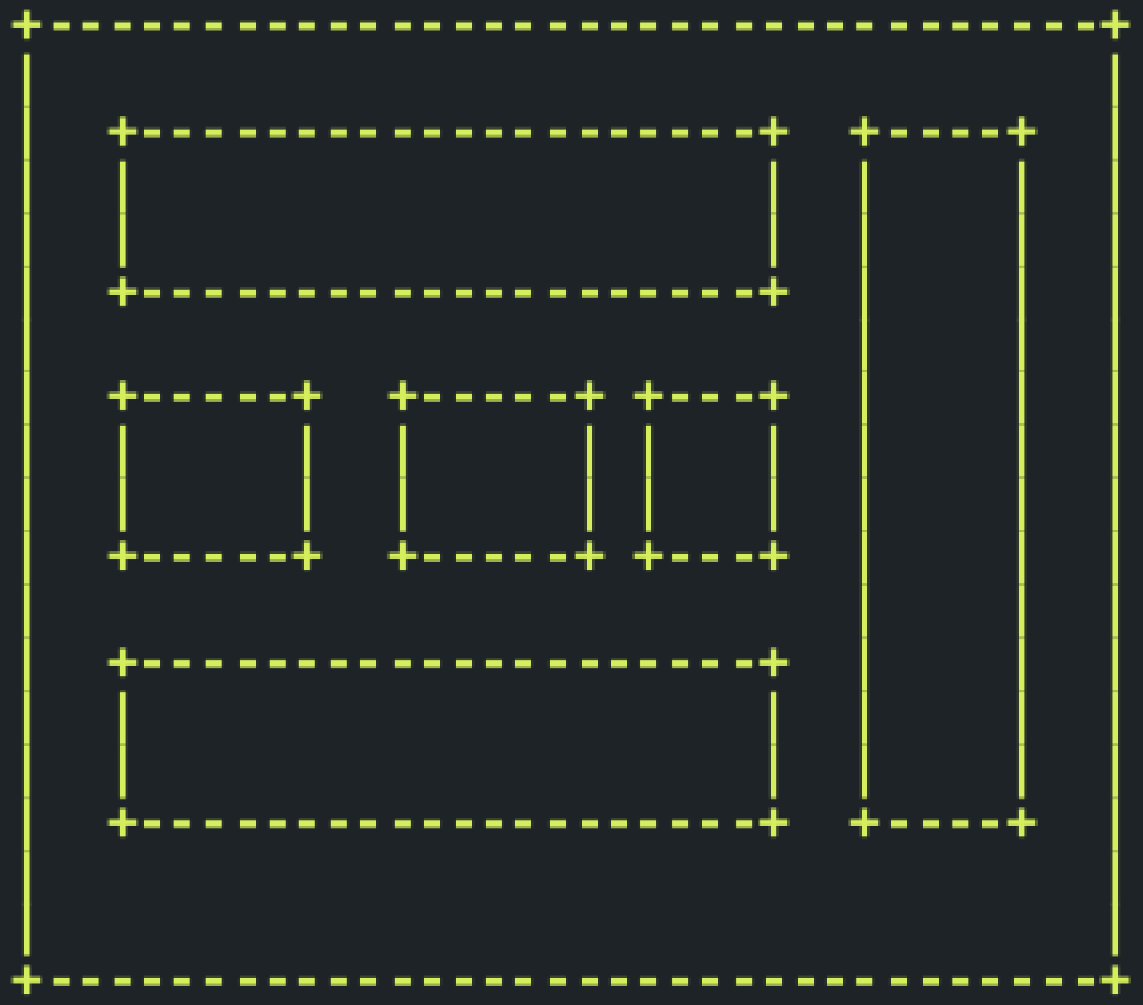
If you get the mindset

everything is a service

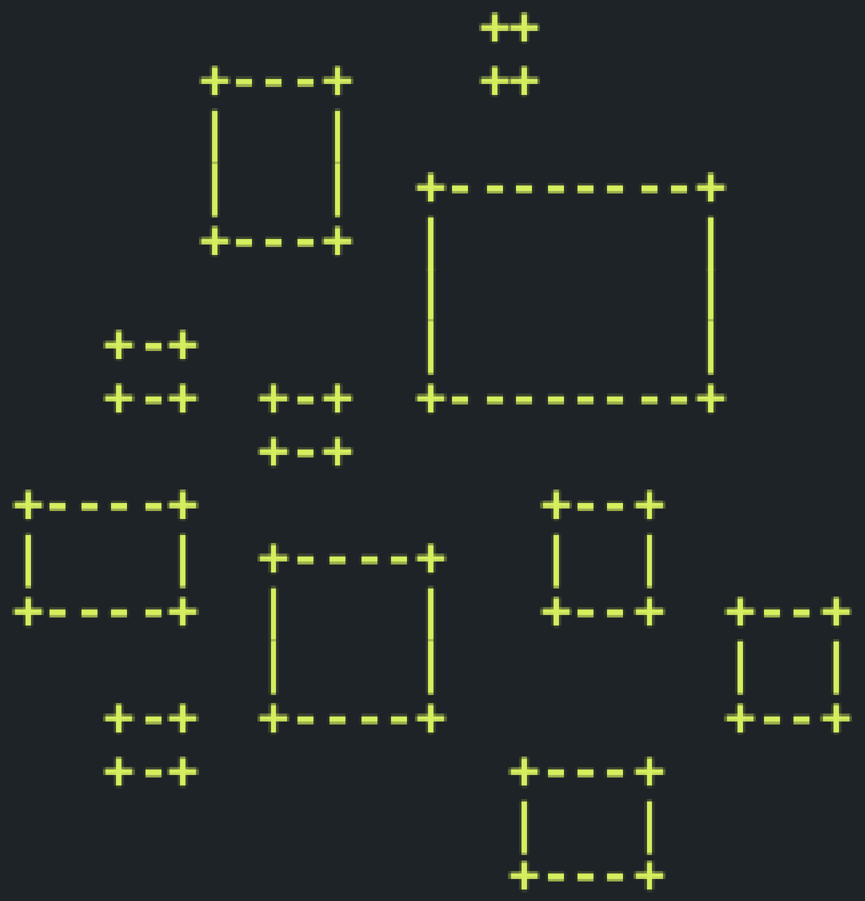
just not always very micro

Get more practical?!?!?!

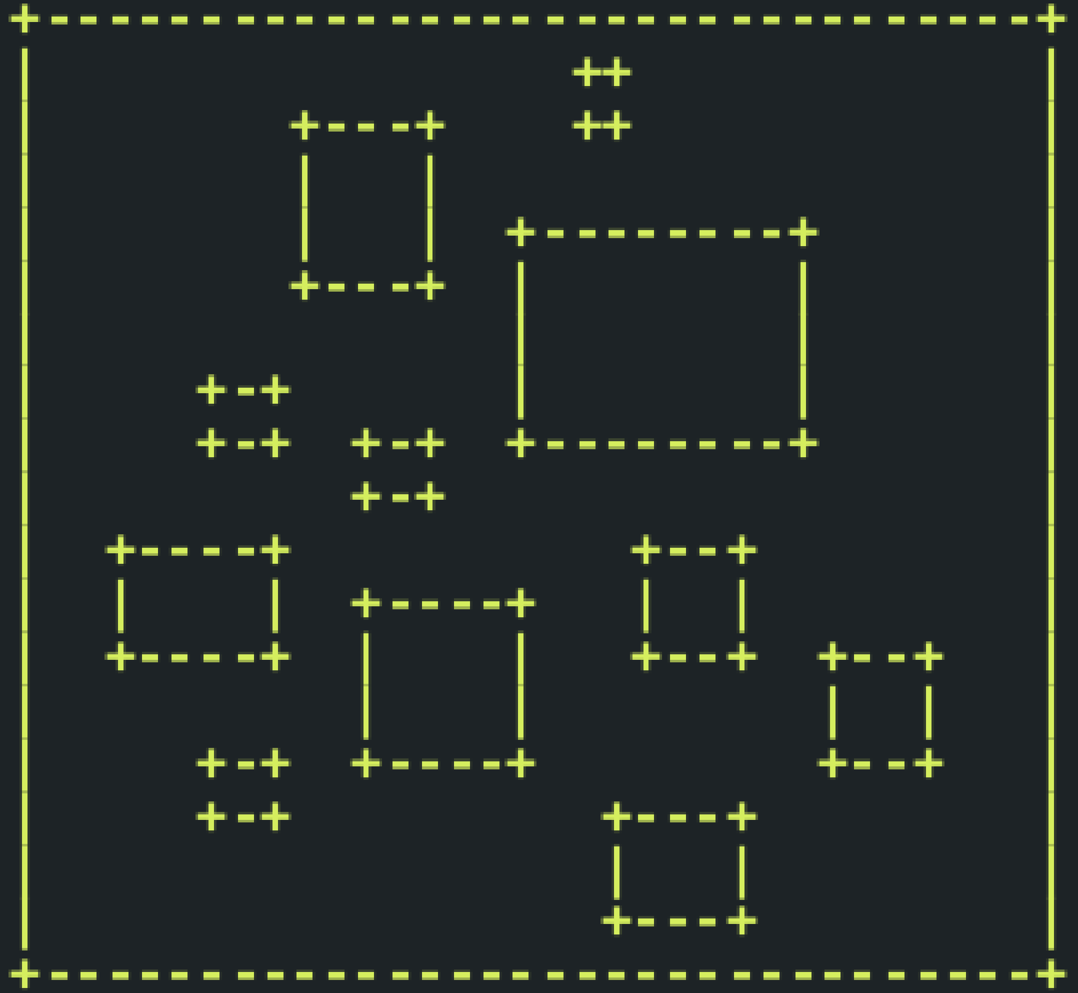
How to package?
How to deploy?
How to scale?
How to orchestrate?
How to monitor?
How to discover?



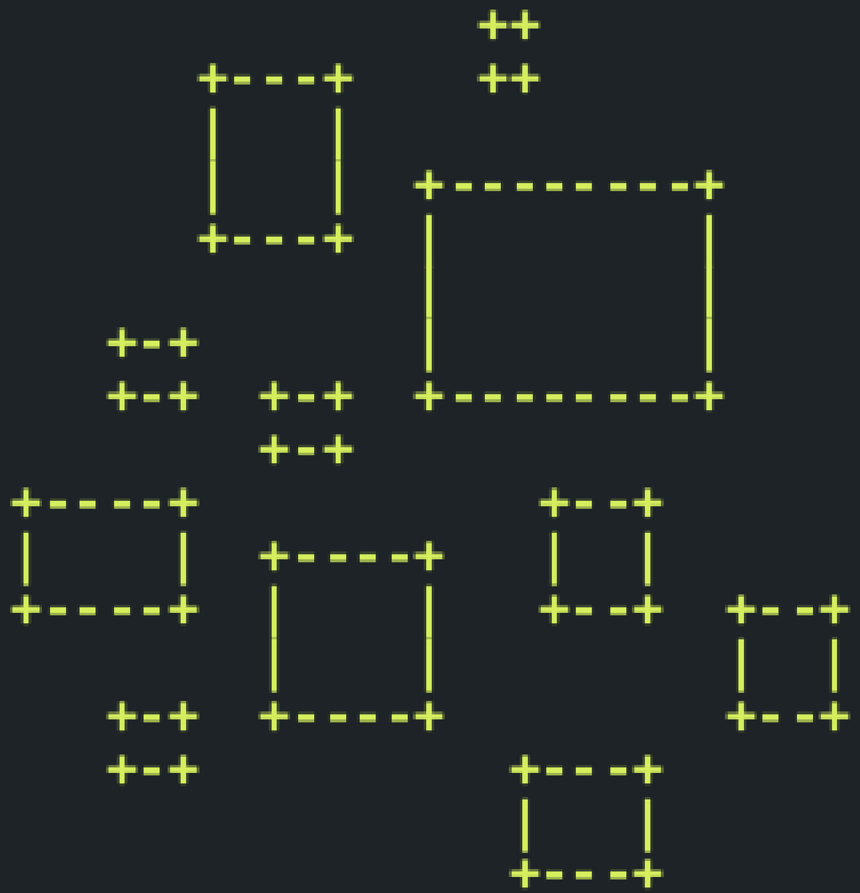
Monolythic / layered



Microservice



Something in between
(components ?)



Microservices

What if?

What if?

each services runs its own process? JVM

What if?

each services runs its own process? JVM

deployed independently if needed

What if?

each services runs its own process? JVM

deployed independently if needed

talks with other through HTTP? or some message bus

What if?

each services runs its own process? JVM

deployed independently if needed

talks with other through HTTP? or some message bus

covers small domain problem IAM, results, content

What if I use Vert.X?

What if I use Vert.X?

build services as modules

What if I use Vert.X?

build services as modules

deploy independently or together

What if I use Vert.X?

build services as modules

deploy independently or together

communicate over event bus calls RPC style

What if I use Vert.X?

build services as modules

deploy independently or together

communicate over event bus calls RPC style

decouple frontend (JS) and backend HTTP calls

Challenges?

APIs (external, web, external)
Versioning generic services
Consumer driven contracts
Scaling individual services (behind event bus)