Liquefy the cloud

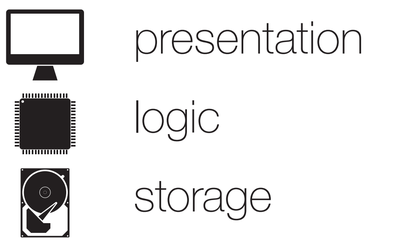
Etienne Brodu, Stéphane Frenot, Frédéric Oblé, Fabien Cellier

How to abstract web services' usage variation from developpement ?

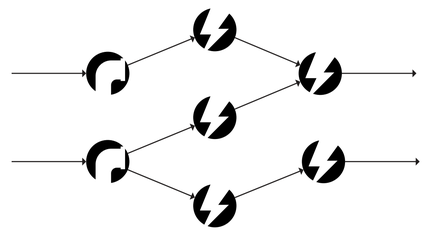
A popular web service might grow from thousands to millions of users in a matters of days.

To react to such variation of usage, they have to be scalable.

The classical approaches - the **three tiers** architecture, frameworks like **storm** or langages like **erlang** - allow developer to split web services into well defined parts in order to be scalable.



three tiers architecture



Storm

Instead we want to **automatically** split a web service into **stateless parts**, and making them communicate by **volatile data streams**, thus making the web service scalable.

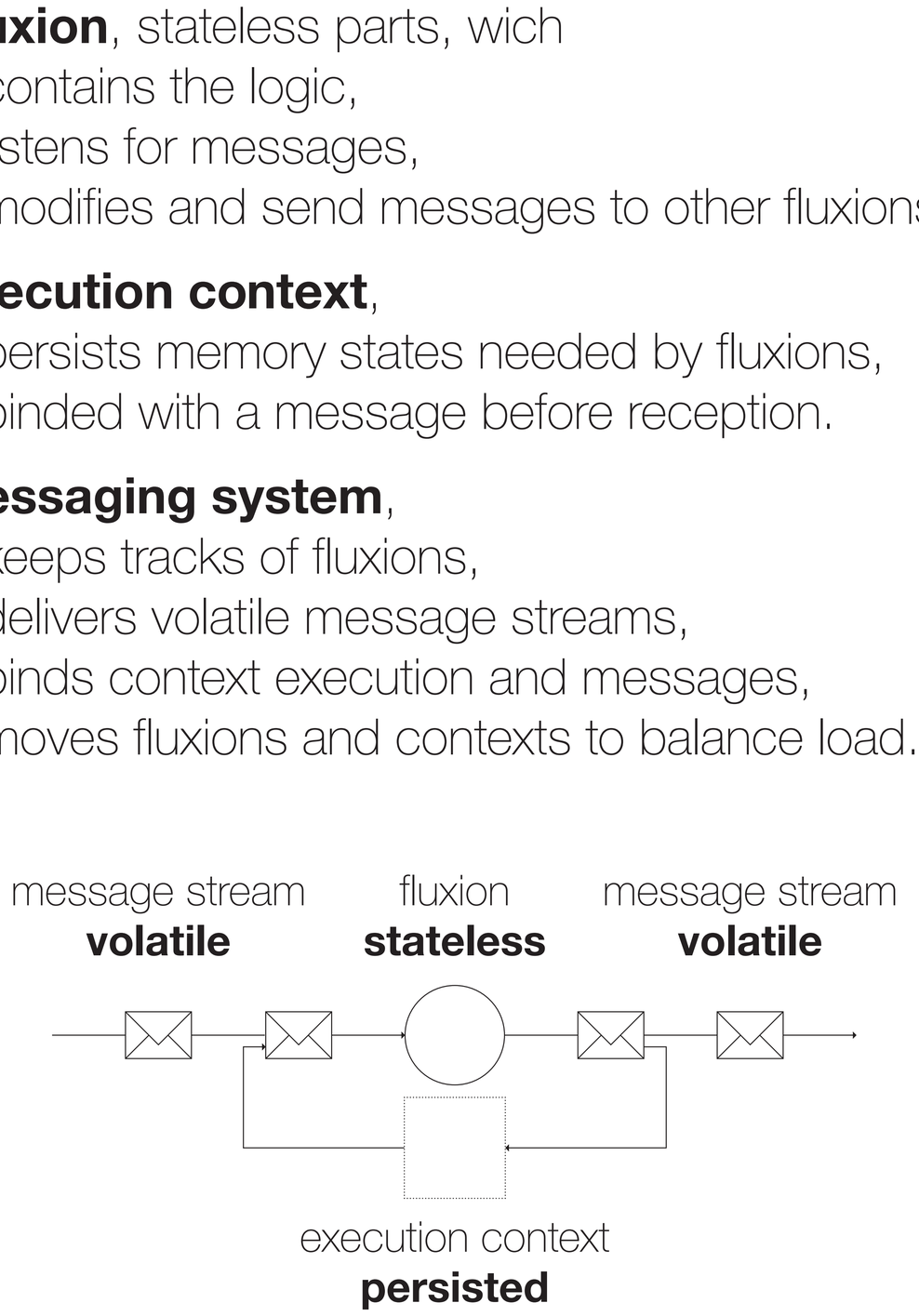
We want to abstract the scalability from developement constrains.

Our approach is composed of :

Fluxion, stateless parts, wich
+ contains the logic,
+ listens for messages,
+ modifies and send messages to other fluxions.

Execution context,
+ persists memory states needed by fluxions,
+ binded with a message before reception.

Messaging system,
+ keeps tracks of fluxions,
+ delivers volatile message streams,
+ binds context execution and messages,
+ moves fluxions and contexts to balance load.



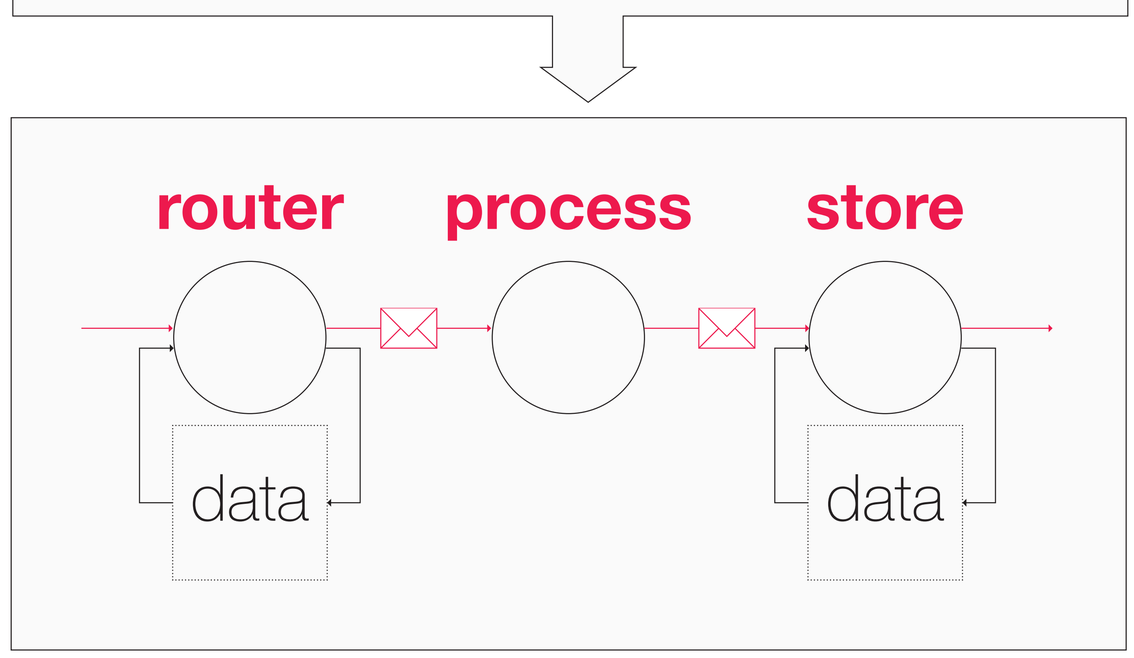
node.js express 3.0.0

```
function router(req, res) {
  // gather data
  next()
}

function process(req, res) {
  // process request
  next()
}

function store(req, res) {
  // store data
  next()
}

...
```



We aim to be able to handle more request than a monolithic approach, but without the constraints on developpement.