

Liquefy the cloud



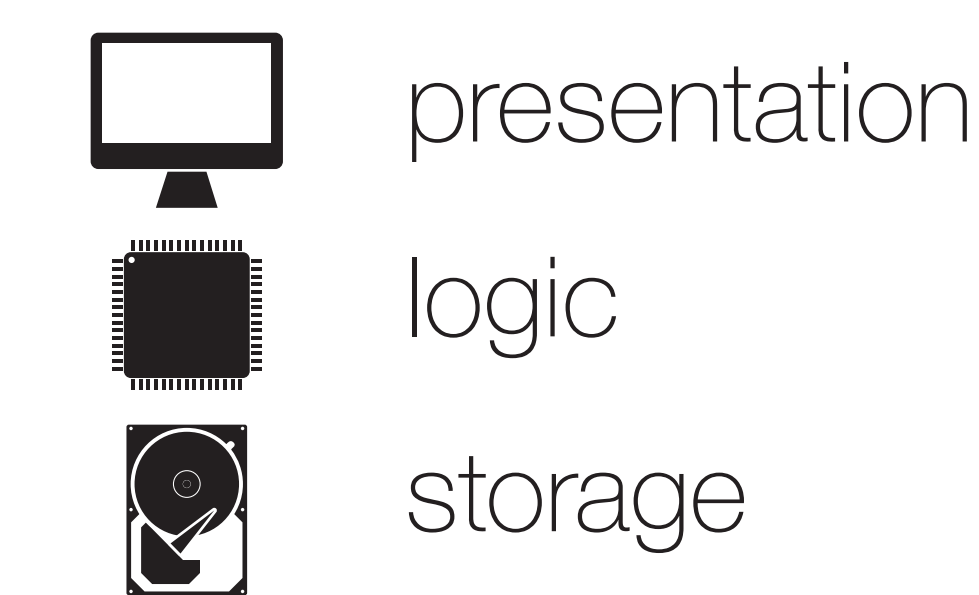
Etienne Brodu, Stéphane Frenot, Frédéric Oblé, Fabien Cellier

How to abstract web services' usage variation from developpement ?

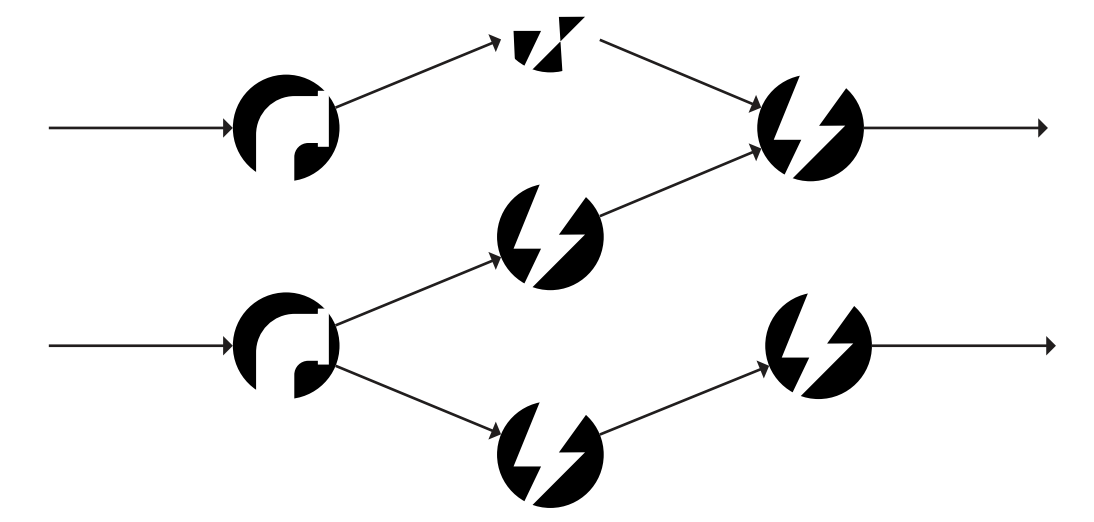
A popular web service might grow from thousands to millions of users in a matters of days.

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

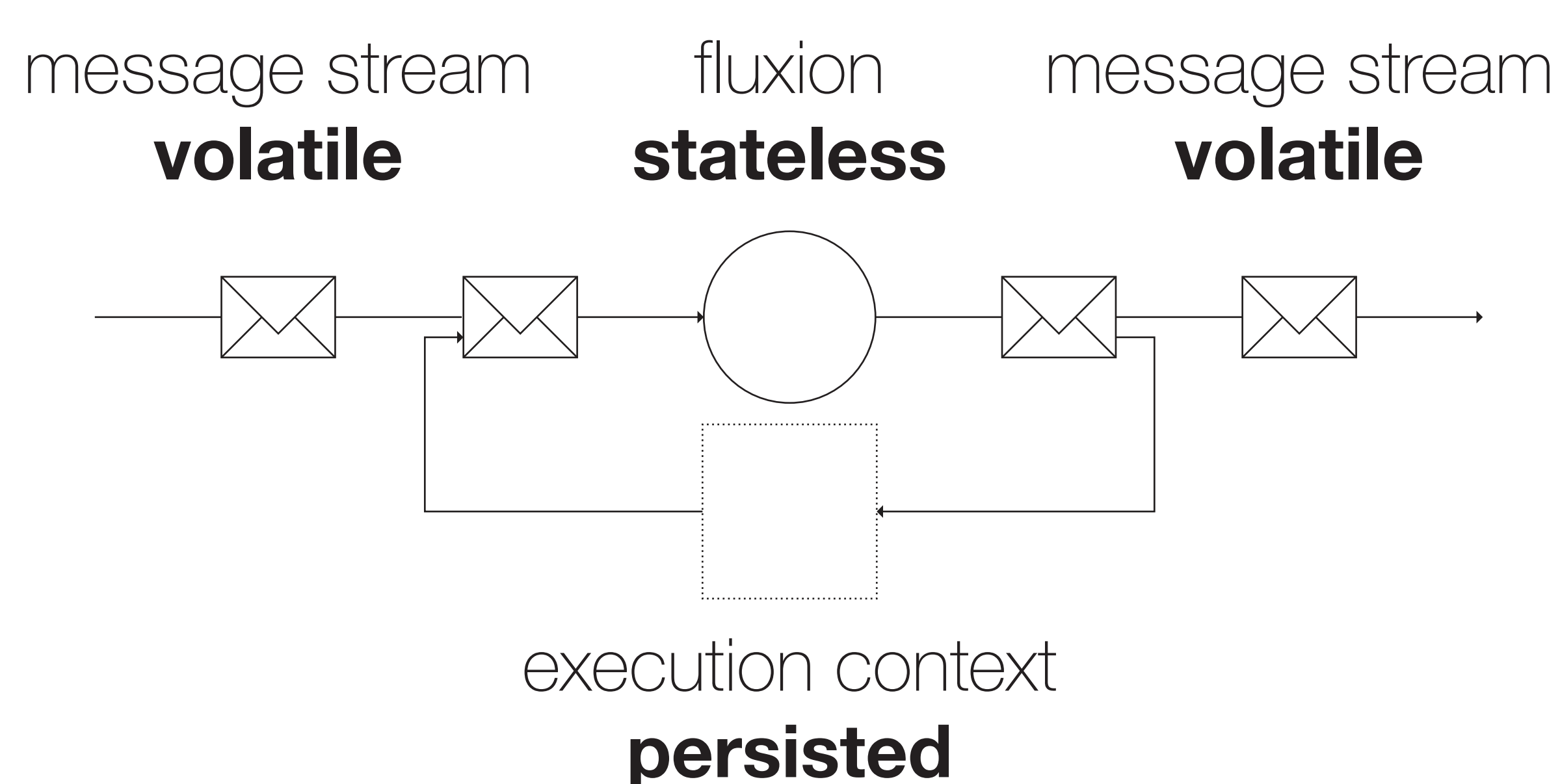
- contains the logic,
- listen for messages,
- modify and send messages to other fluxions.

Execution context,

- persist memory states needed by fluxions,
- binded with a message before reception.

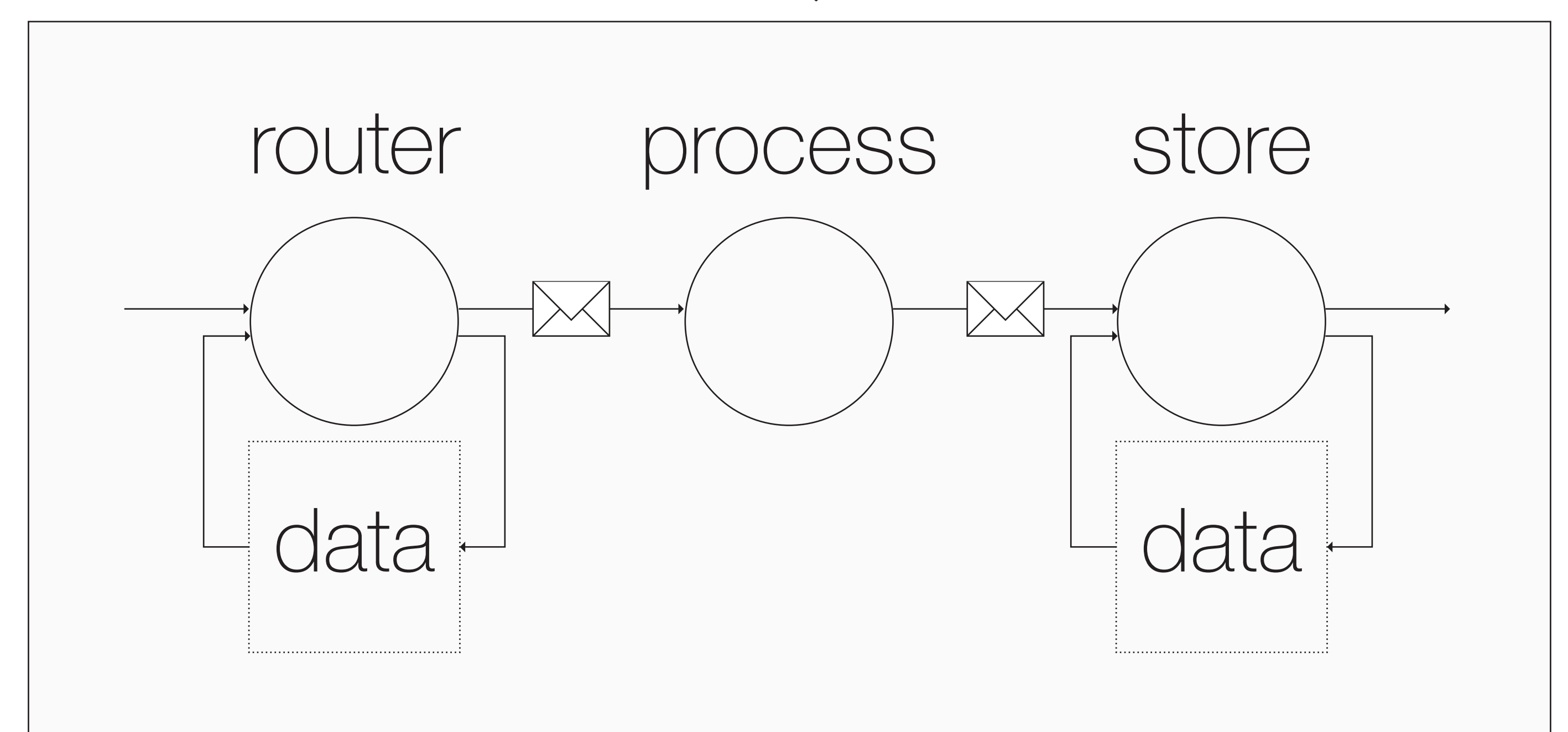
Messaging system,

- keep tracks of fluxions,
- deliver volatile message streams,
- bind context execution and messages,
- move fluxions and contexts to balance load.



node[®] express[™]

```
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 constraints on developpement.