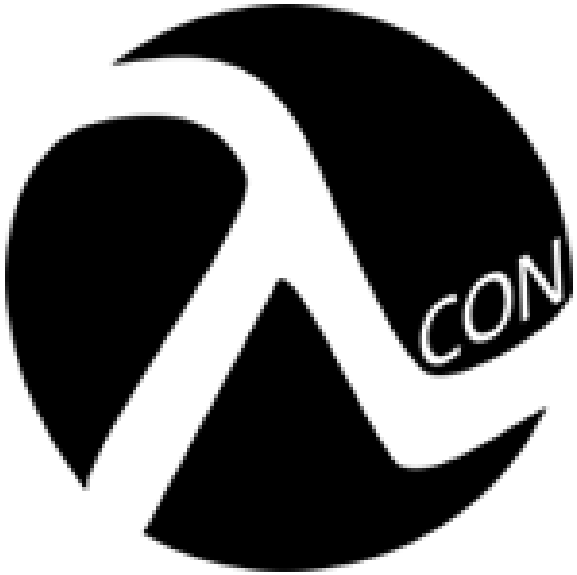
STREAMS ON TOP OF SCALA

BY @WAXZCE – QUENTIN ADAM

LAMBDACON 2015 BOLOGNA



WHO AM I ?

Quentin ADAM from the Clever Cloud

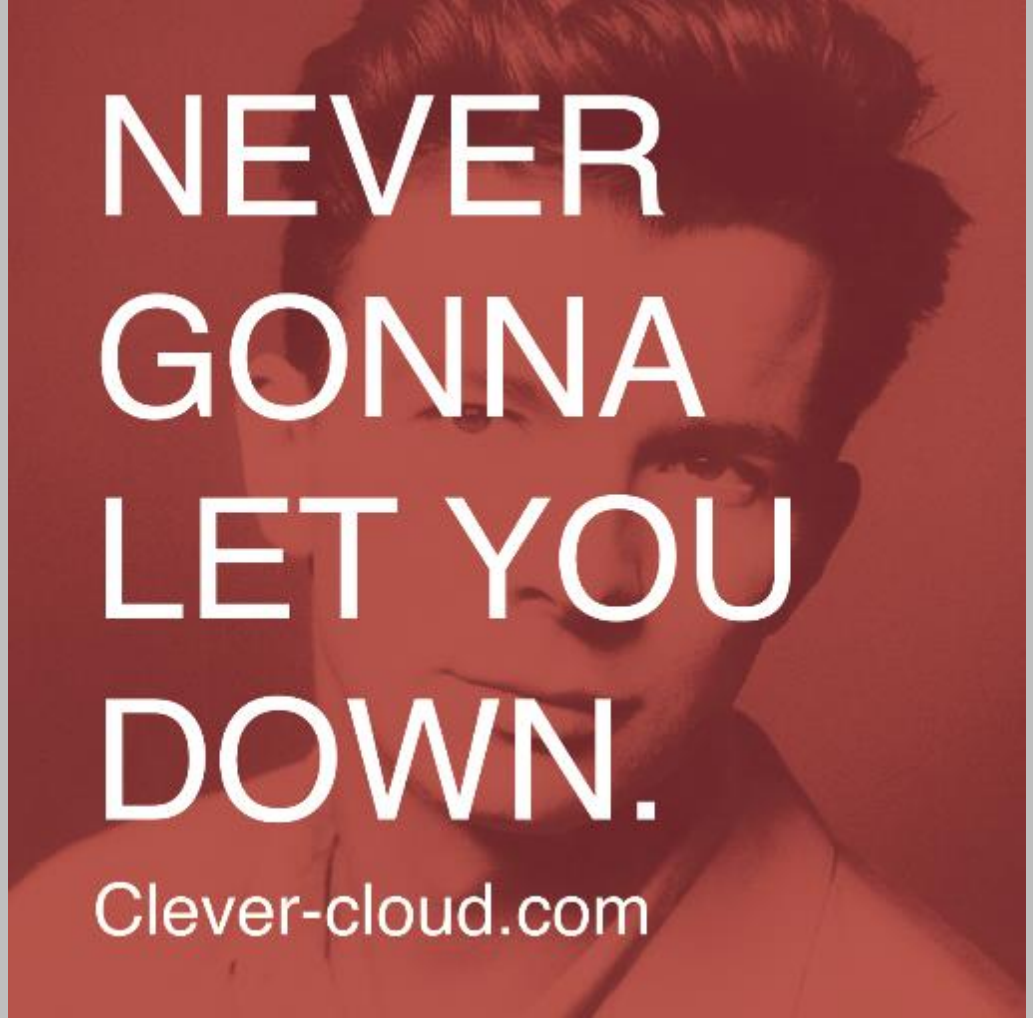
@waxzce on twitter – github- soundcloud – instagram ….

MY DAY TO DAY WORK : CLEVER CLOUD, MAKE YOUR APP RUN ALL THE TIME

clever cloud

KEEP YOUR APPS ONLINE. MADE WITH NODE.JS, SCALA, JAVA, RUBY, PHP, PYTHON, GO OR DOCKER...

And learn a lot of things about your code, apps, and good/bad design…



AND LEARN A LOT OF THINGS ABOUT YOUR CODE, APPS, AND GOOD/BAD DESIGN…

WHY ARE STREAMS SO TRENDY THESE DAYS?

2 MAIN REASONS

DATA IS BECOMING BIGGER AND BIGGER

WE NEED TO ACT WHILE DATA TRANSFERRING IS OCCURRING

&

WE ARE NOW USING LOTS OF PERMANENT CONNECTIONS

REALITY FOR MANY DEVELOPERS

WEBSOCKETS

HTTP/2

EXAMPLE

WHAT IS INSIDE AN HTTP REQUEST ?

Verb

- The action

Resource

- The object of the action

Headers

- The context of the action

Body

- Optional
- The datas

IN MANY CASES THE REQUEST IS MANIPULATED ALL FROM MEMORY

EXAMPLE

UPLOADING A 20 GB FILE ON A POST REQUEST

AND STORING IT

IT'S A BAD IDEA TO PUT THE BODY PART IN MEMORY

CREATE A TEMP FILE TO STORE THE DATA

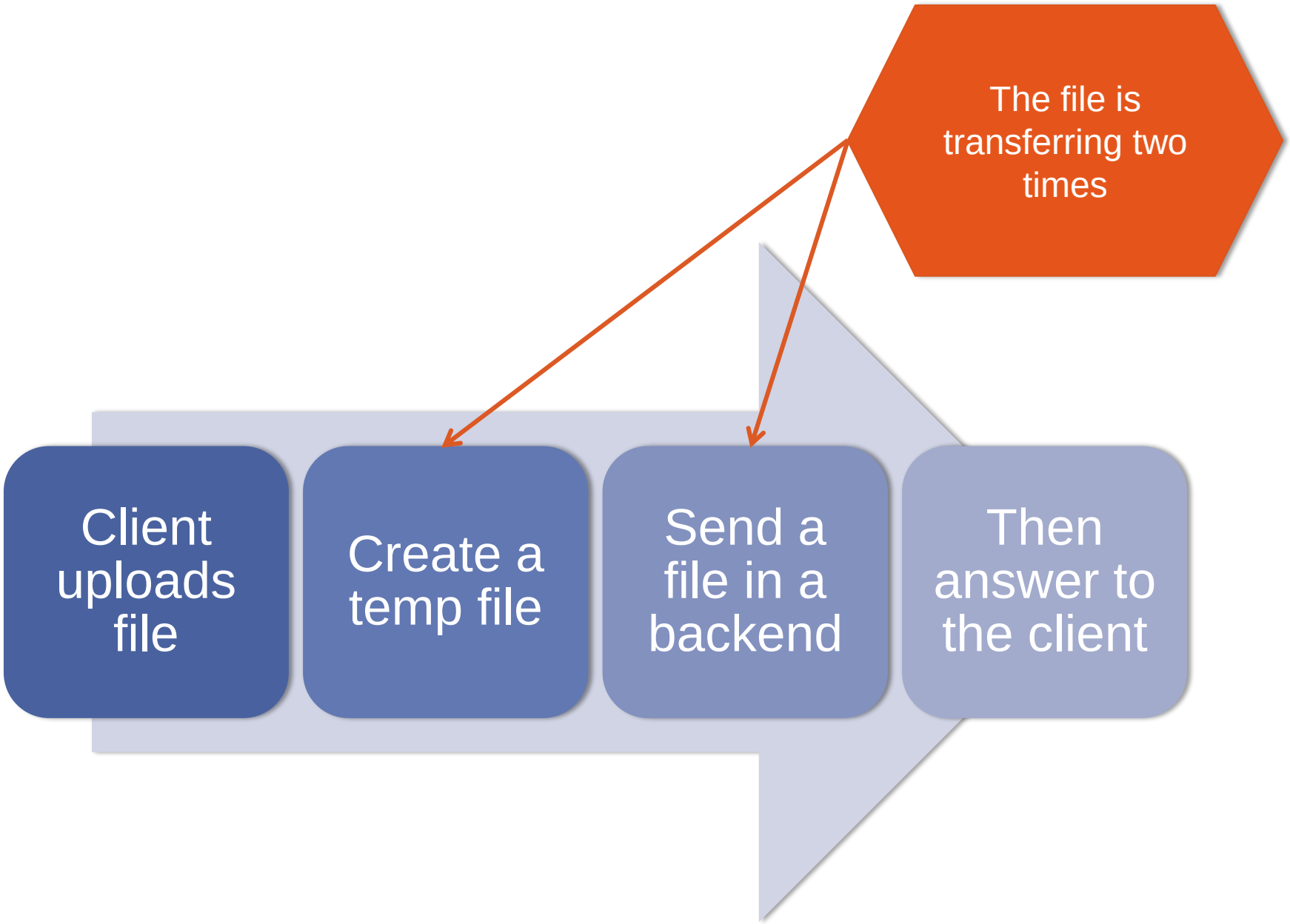
I HATE FILE SYSTEMS

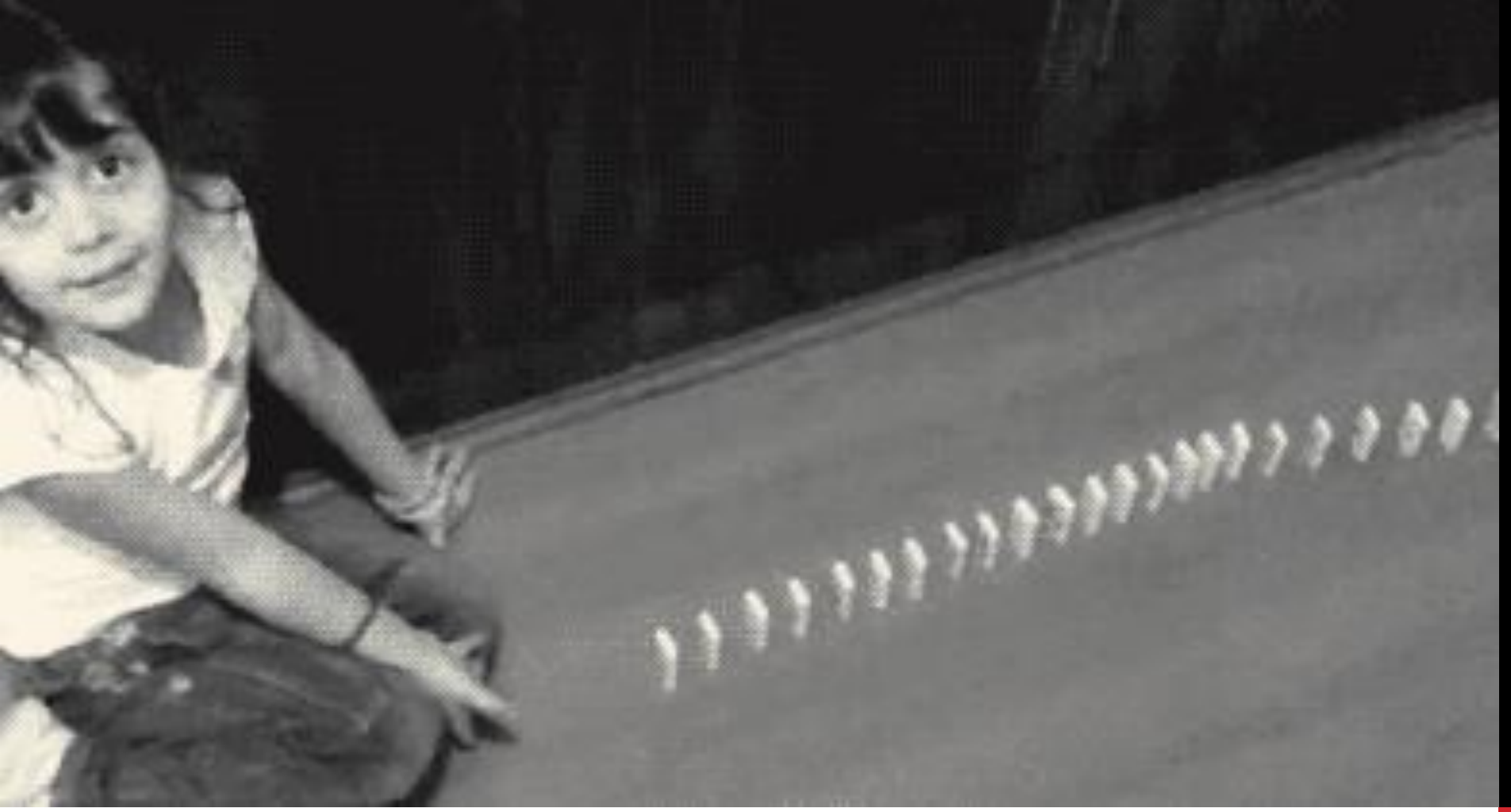
I HATE FILE SYSTEMS

DO NOT USE THE FILE SYSTEM AS A DATASTORE

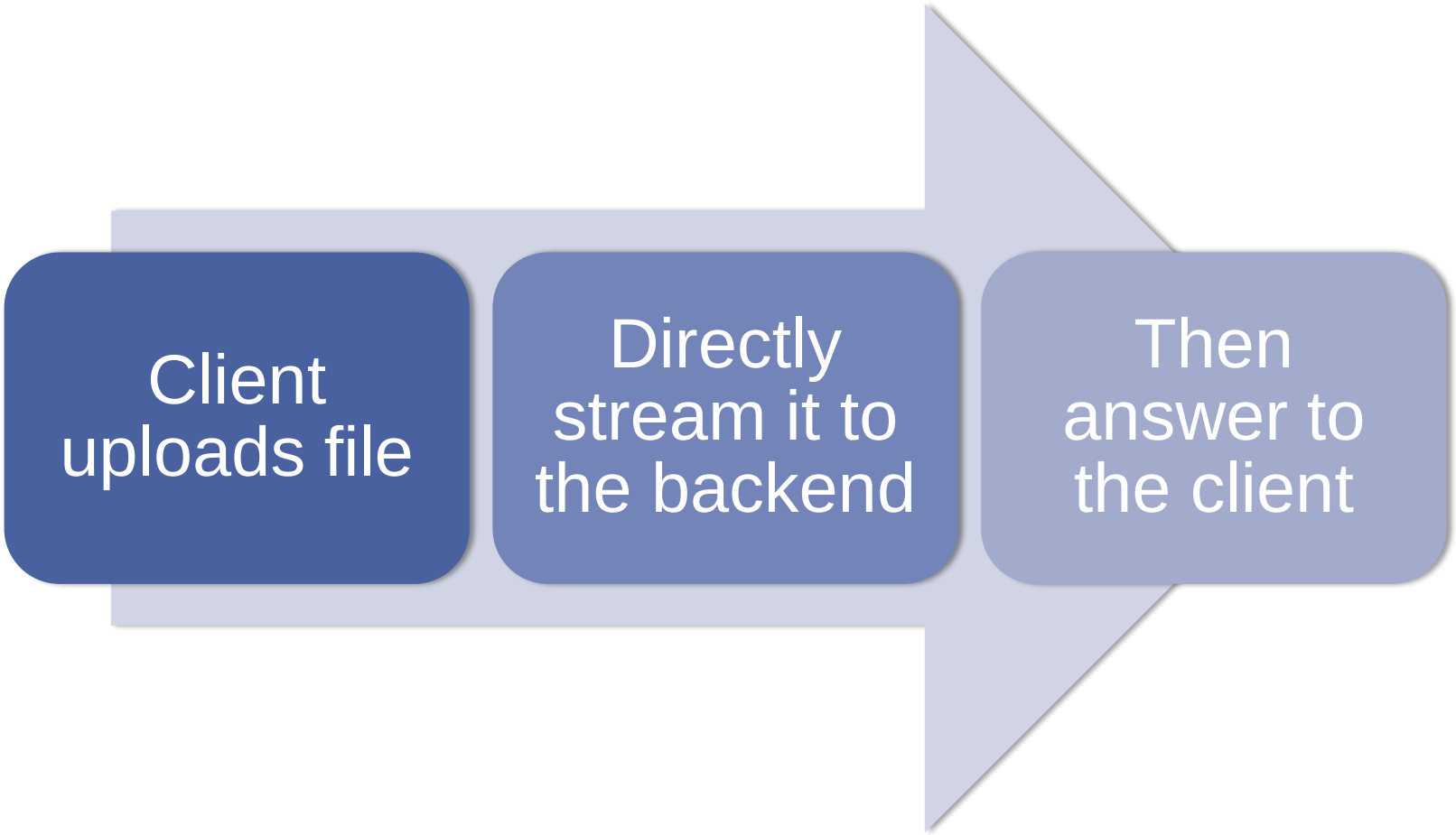
File systems are POSIX compliant

- POSIX is ACID
- POSIX is powerful but is a bottleneck
- File System is the nightmare of ops
- File System creates coupling (host provider/OS/language)
- SPOF-free multi tenant File System is a unicorn


The file is transferring two times
Client uploads file
Create a temp file
Send a file in a backend
Then answer to the client



NOT SO GOOD


Client uploads file
Directly stream it to the backend
Then answer to the client

LET'S ACT ON STREAMS!

CLASSIC JAVA STREAM MANAGEMENT

```
FileInputStream in = null;
FileOutputStream out = null;

try {
    in = new FileInputStream("xanadu.txt");
    out = new FileOutputStream("outagain.txt");
    int c;

    while ((c = in.read()) != -1) {
        out.write(c);
    }
} finally {
    if (in != null) {
        in.close();
    }
    if (out != null) {
        out.close();
    }
}
```

CLASSIC JAVA STREAM MANAGEMENT

```
FileInputStream in = null;
FileOutputStream out = null;

try {
    in = new FileInputStream("xanadu.txt");
    out = new FileOutputStream("outagain.txt");
    int c;

    while ((c = in.read()) != -1) {
        out.write(c);
    }
} finally {
    if (in != null) {
        in.close();
    }
    if (out != null) {
        out.close();
    }
}
```

- Buffers
  - Buffer management
  - Buffer exception
  - Buffer overflow

CLASSIC JAVA STREAM MANAGEMENT

- Low performances if not buffered
- Not modular
- Thread blocking
- Code is ugly
- No back pressure
- Error handling is bad
- i/o management and business code are mixed

I CAN’T SLEEP AT NIGHT

WE HAVE TO FIND A BETTER WAY

DATA STRUCTURES TO EXPRESS DATA STREAM MANAGEMENT

FIRST OPTION

PLAY-ITERATEES

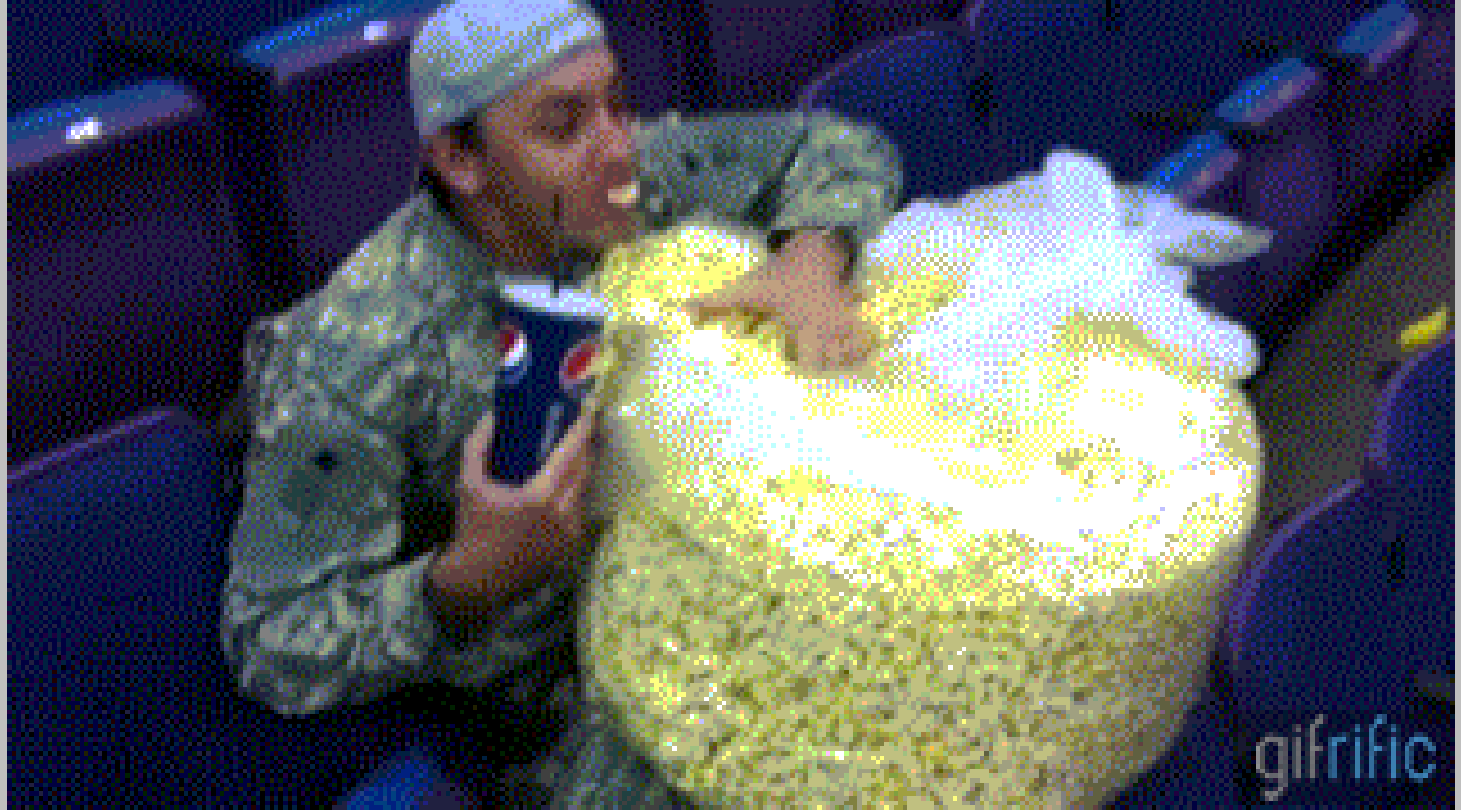
ITERATEE : HOW TO MANAGE A STREAM

Like a recipe

Consume the data

ENUMERATOR : DATA STREAM

Produce the data



ENUMERATEE

Set of tools to do cool things with Iteratee and Enumerator

SIMPLE EXAMPLE TO START

WHAT'S OUR GOAL

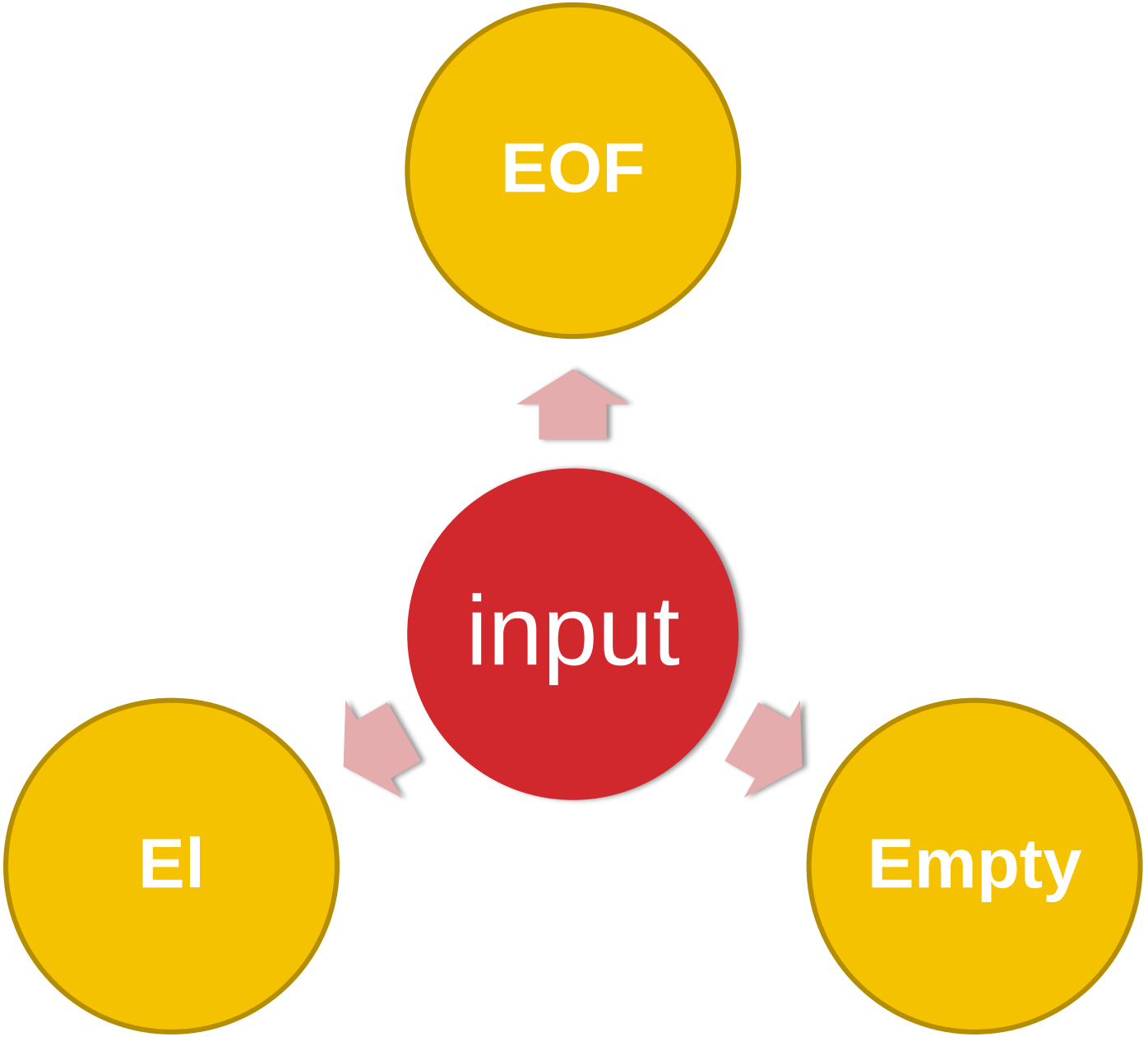
- **Get Reactive Manifesto words**
- **Stream it to an iteratee to group it by pair of word**
- **Print it**

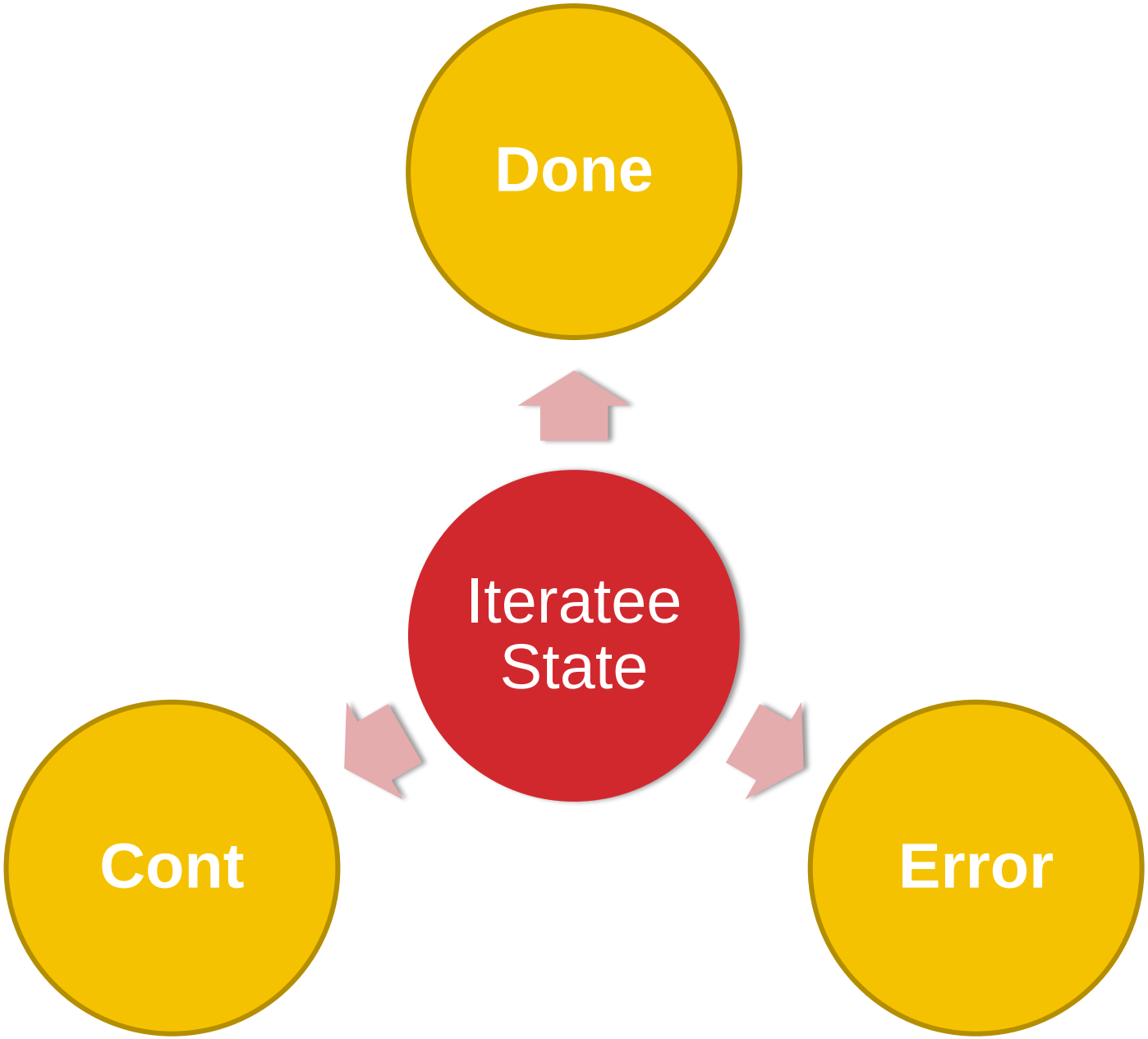
AN ENUMERATOR

```
val enum1:Enumerator[String] = Enumerator.repeat( {
  Random.shuffle(reactivemanifestowords).head
})
```

AN ITERATEE

```
val wordsAgregation: Iteratee[String, String] = {
  def step(previous:Option[String])(i: Input[String]): Iteratee[String, String] = i match {
    case Input.EOF => Done(previous.getOrElse(""),Input.EOF)

    case Input.Empty => Cont[String, String](i => step(previous)(i))

    case Input.El(e) =>{
      previous.fold(
          Cont[String, String](i => step(Some(e))(i))
      )(
          p =>
              Done(p + " " + e, Input.Empty)
      )

    }

  }
  (Cont[String, String](i => step(None)(i)))
}
```


EOF
input
El
Empty

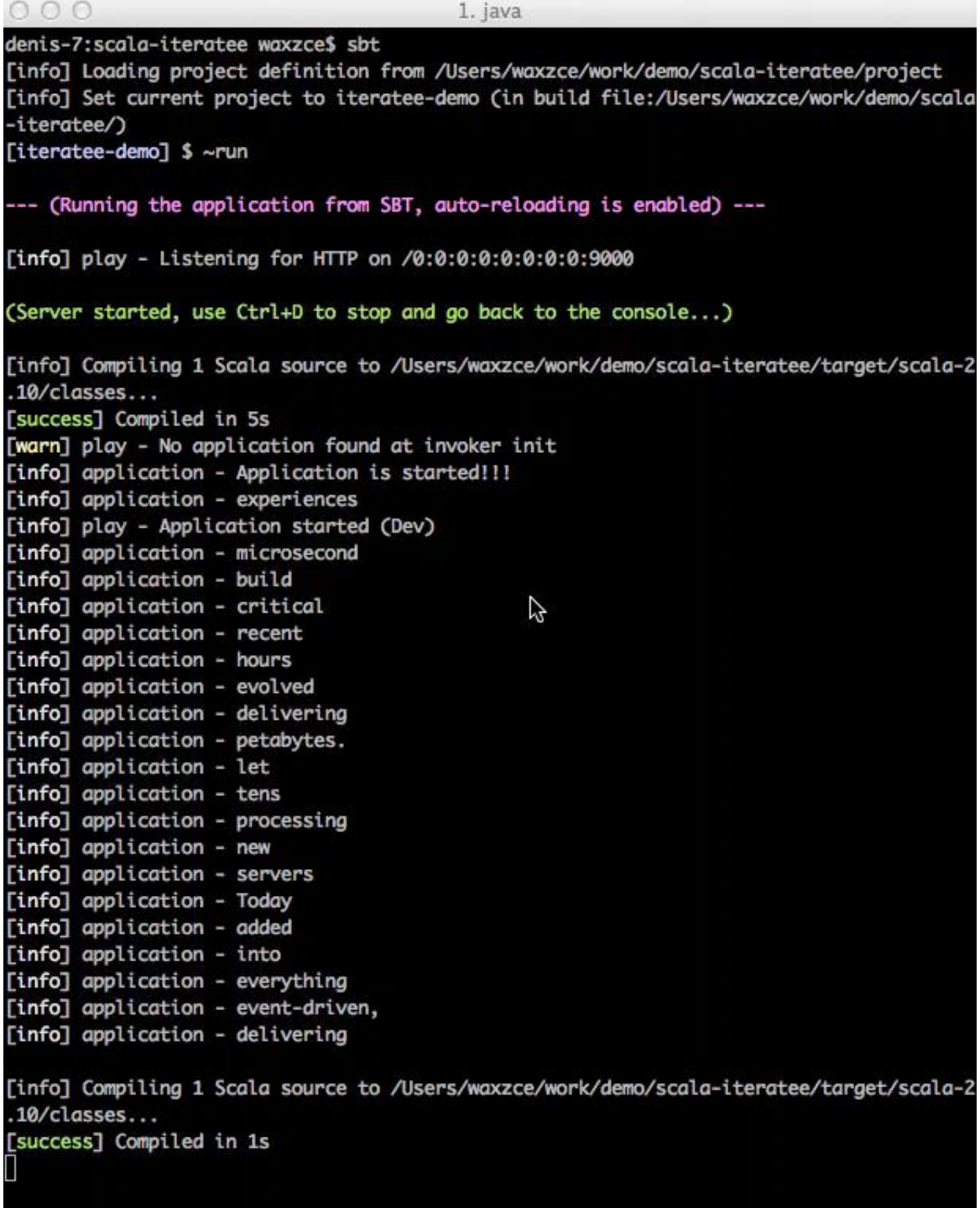

Done
Iteratee
State
Cont
Error

AN ITERATEE

```
val wordsAgregation: Iteratee[String, String] = {
  def step(previous:Option[String])(i: Input[String]): Iteratee[String, String] = i match {
    case Input.EOF => Done(previous.getOrElse(""),Input.EOF)

    case Input.Empty => Cont[String, String](i => step(previous)(i))

    case Input.El(e) =>{
      previous.fold(
          Cont[String, String](i => step(Some(e))(i))
      )(
          p =>
              Done(p + " " + e, Input.Empty)
      )

    }

  }
  (Cont[String, String](i => step(None)(i)))
}
```

AN ITERATEE TO MANAGE THE STREAM

```
val myIteratee =  Enumeratee.grouped[String](wordsAgregation)  &>> logstream
```


1. java
denis-7:scala-iteratee waxzce$ sbt
[info] Loading project definition from /Users/waxzce/work/demo/scala-iteratee/project
[info] Set current project to iteratee-demo (in build file:/Users/waxzce/work/demo/scala-iteratee/)
[iteratee-demo] $ ~run
--- (Running the application from SBT, auto-reloading is enabled) ---
[info] play - Listening for HTTP on /0:0:0:0:0:0:0:0:9000
(Server started, use Ctrl+D to stop and go back to the console...)
[info] Compiling 1 Scala source to /Users/waxzce/work/demo/scala-iteratee/target/scala-2.10/classes...
[success] Compiled in 5s
[warn] play - No application found at invoker init
[info] application - Application is started!!!
[info] application - experiences
[info] play - Application started (Dev)
[info] application - microsecond
[info] application - build
[info] application - critical
[info] application - recent
[info] application - hours
[info] application - evolved
[info] application - delivering
[info] application - petabytes.
[info] application - let
[info] application - tens
[info] application - processing
[info] application - new
[info] application - servers
[info] application - Today
[info] application - added
[info] application - into
[info] application - everything
[info] application - event-driven,
[info] application - delivering
[info] Compiling 1 Scala source to /Users/waxzce/work/demo/scala-iteratee/target/scala-2.10/classes...
[success] Compiled in 1s

AN ITERATEE

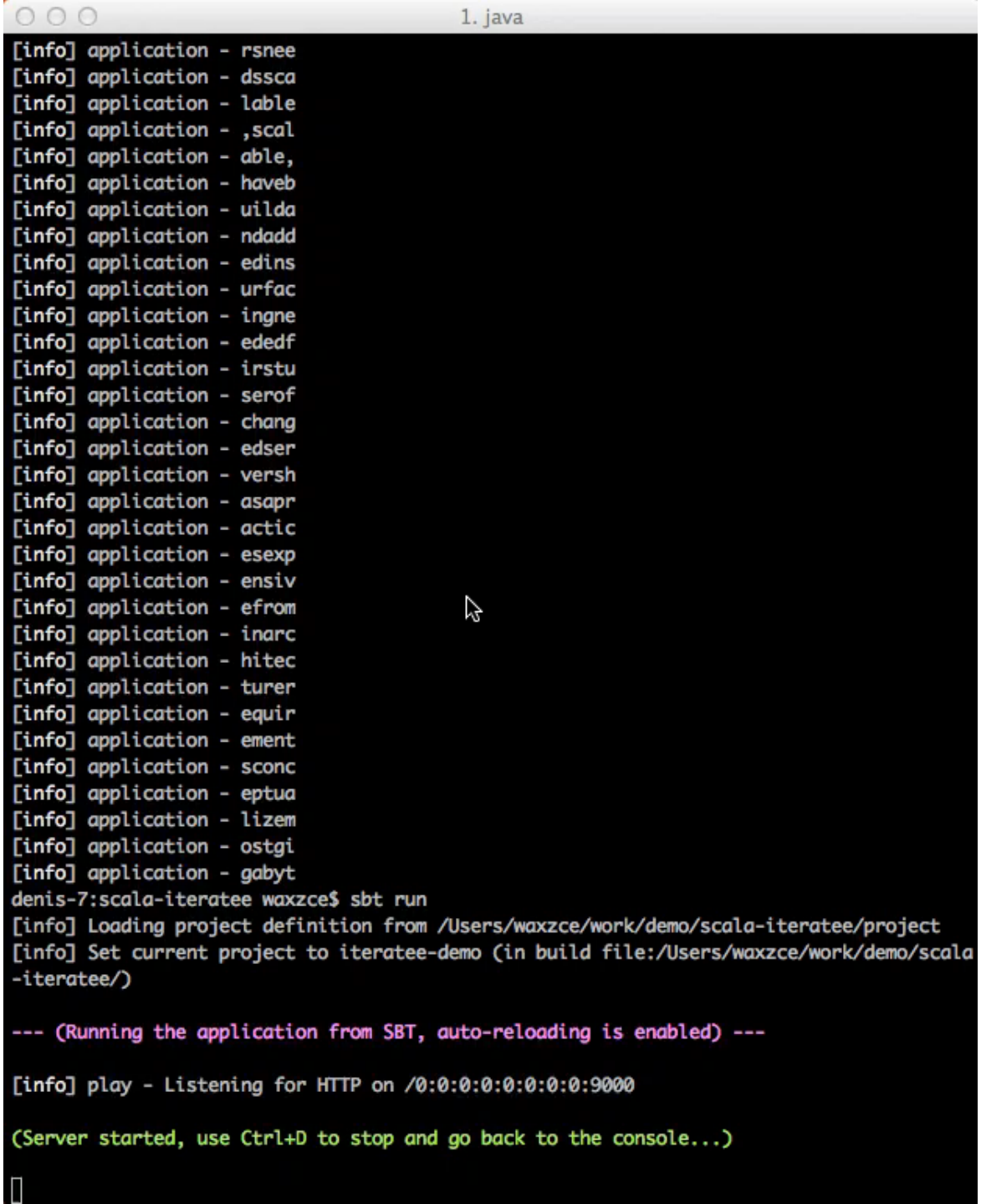
```
val wordLength = 5

def qualifyWord: Iteratee[String, String] = {
  def step(c_buff: String)(i: Input[String]): Iteratee[String, String] = i match {
    case Input.EOF => Done(c_buff, Input.EOF)
    case Input.Empty => Cont[String, String](i => step(c_buff)(i))
    case Input.El(e) =>
      val n_buff = c_buff ++ e
      n_buff.length match {
        case x if x > wordLength => {
          val (done, next) = n_buff.splitAt(wordLength)
          Done(done, Input.El(next))
        }
        case _ => Cont[String, String](i => step(n_buff)(i))
      }
  }
  (Cont[String, String](i => step("")(i)))
}
```

1. java

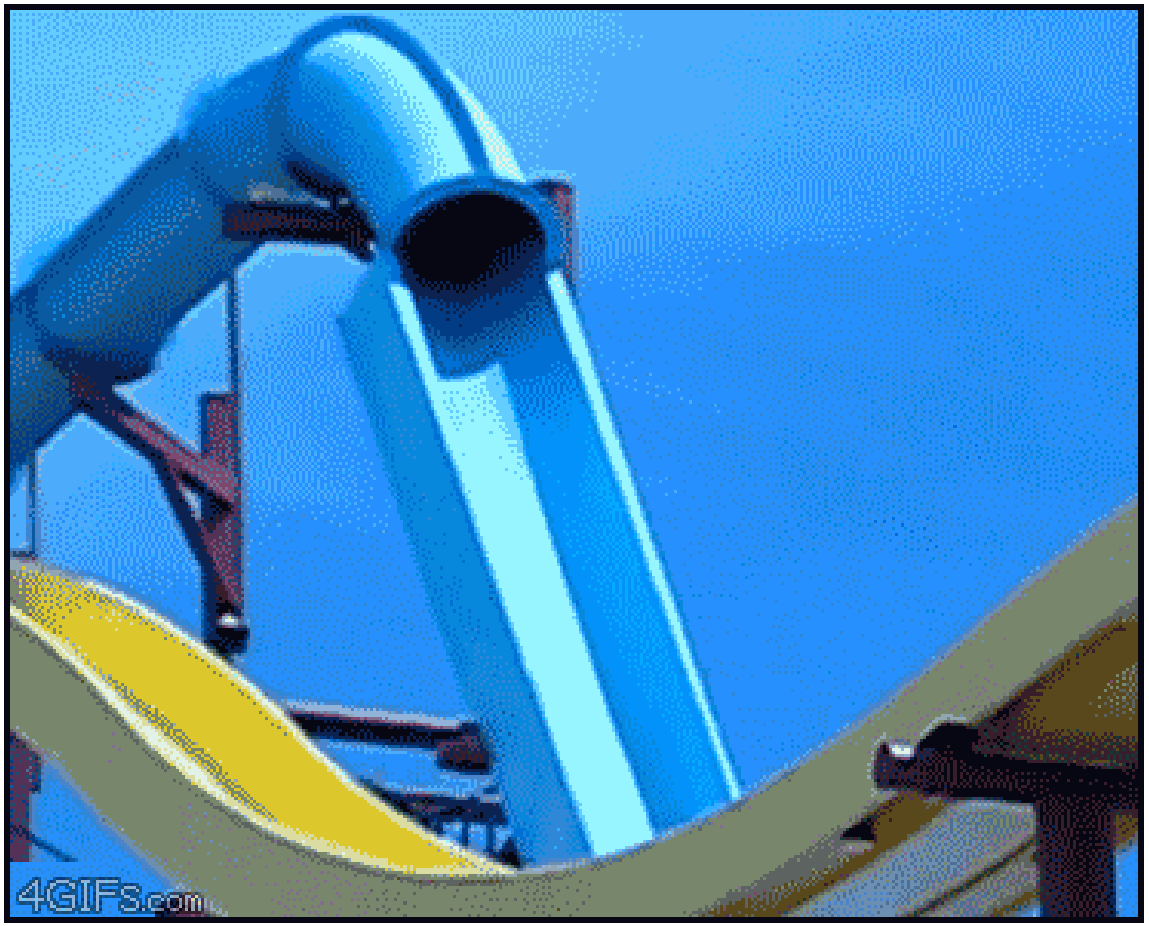
```
[info] application - servers, The
[info] application - let practices
[info] application - that complex,
[info] application - characteristics needs
[info] application - demand for
[info] application - has 100%
[info] application - and Finance
[info] application - added Google
[info] application - response Finance
[info] application - and architectures.
[info] application - of feel,
[info] application - running Google
[info] application - deployed deployed
[info] application - surfacing into
[info] application - application industries.
[info] application - to today's
[info] application - New gigabytes
[info] application - architecture and
[info] application - to with
[info] application - large backed
[info] application - years expanding
[info] application - real-time the
[info] application - reactive. are
[info] application - now have
[info] application - scalable, technologies.
[info] application - are Manifesto
[info] application - of applications
[info] application - in call
[info] application - companies to
[info] application - multi-threading. and
[info] application - application reactive.
[info] application - clusters build
denis-7:scala-iteratee waxzce$ sbt run
[info] Loading project definition from /Users/waxzce/work/demo/scala-iteratee/project
[info] Set current project to iteratee-demo (in build file:/Users/waxzce/work/demo/scala
-iteratee/)

--- (Running the application from SBT, auto-reloading is enabled) ---

[info] play - Listening for HTTP on /0:0:0:0:0:0:0:0:9000

(Server started, use Ctrl+D to stop and go back to the console...)
```

ITERATEE COMPOSITION

```
Enumeratee.grouped[String](qualifyWord) ><> Enumeratee.grouped[String](wordsAgregation) &>> logstream
```


1. java
[info] application - rsnee
[info] application - dssca
[info] application - lable
[info] application - ,scal
[info] application - able,
[info] application - haveb
[info] application - uilda
[info] application - ndadd
[info] application - edins
[info] application - urfac
[info] application - ingne
[info] application - ededf
[info] application - irstu
[info] application - serof
[info] application - chang
[info] application - edser
[info] application - versh
[info] application - asapr
[info] application - actic
[info] application - esexp
[info] application - ensiv
[info] application - efrom
[info] application - inarc
[info] application - hitec
[info] application - turer
[info] application - equir
[info] application - ement
[info] application - sconc
[info] application - eptua
[info] application - lizem
[info] application - ostgi
[info] application - gabyt
denis-7:scala-iteratee waxzce$ sbt run
[info] Loading project definition from /Users/waxzce/work/demo/scala-iteratee/project
[info] Set current project to iteratee-demo (in build file:/Users/waxzce/work/demo/scala
-iteratee/)
--- (Running the application from SBT, auto-reloading is enabled) ---
[info] play - Listening for HTTP on /0:0:0:0:0:0:0:0:9000
(Server started, use Ctrl+D to stop and go back to the console...)

PURELY FUNCTIONAL



TYPE SAFE

COMPOSITION

BUT ITERATEES ARE GOING TO DIE

WHAT'S TRENDY?

SCALAZ STREAMS

WHAT'S IN THE BOX ?

BASED ON SCALAZ BASICS

PROCESS IS KEY

SIMPLY USE THE COLLECTION API

DEMO

BINDINGS WITH

- **http with (netty or http4s)**
- **scodec**
- **…**

BUT WE NEED TO TALK WITH OTHERS…

LIKE OUR FRIENDS ON TOP OF JVM

AND THEY HAVE LESS EXPRESSIVE TYPE SYSTEM

+ POLITICS

ENTER

REACTIVE STREAMS

CURRENT 1.0-M5 (MILESTONE) (M5 OUT YESTEDAY)

AKKA STREAMS

BASIC EXAMPLE

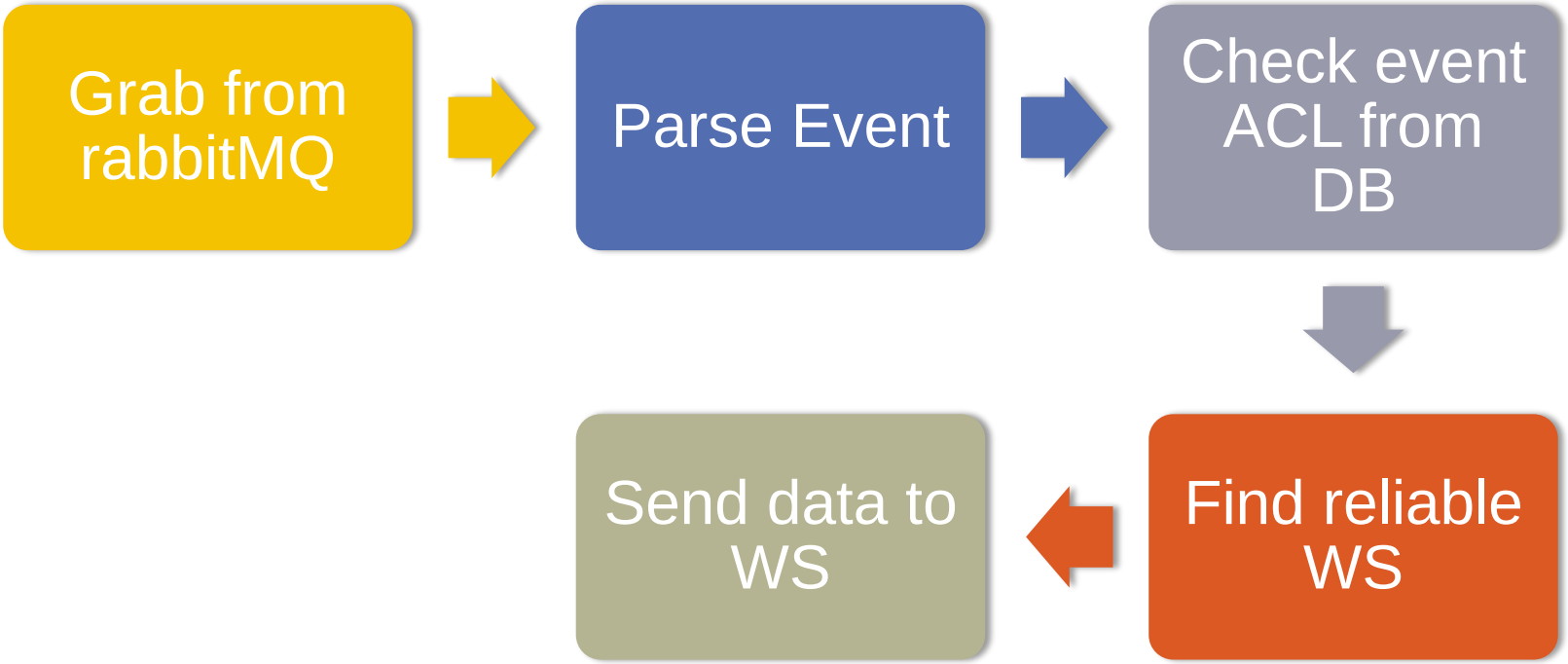
```
23 trait FlowFactory {
24
25
26
27   def print[A](implicit ex: ExecutionContext): Flow[A, A, _] = {
28     Flow[A]
29       .map(e => {
30         Logger.info(e.toString())
31         e
32       })
33   }
34
35
36    def toUpperCase(implicit ex: ExecutionContext): Flow[String, String, _] = {
37     Flow[String]
38       .map(e => {
39        e.toUpperCase()
40       })
41   }
42
43
44 }
```

BASIC EXAMPLE

```
def start = {
  implicit val actorSystem = ActorSystem("akka-stream-1")

  import actorSystem.dispatcher

  implicit val materializer = ActorFlowMaterializer()

  import akka.stream.scaladsl.FlowGraph.Implicits._

  Logger.info("Exchanges, queues and bindings declared successfully.")

  val f = scala.io.Source.fromFile(new File("/Users/waxzce/work/testperso/scalazstreamsexample/testdata/asimov.html")).getLines()

  val in = Source(iterable = f.toStream)

  val out = Sink.ignore()

  val g = FlowGraph.closed() { implicit b =>

    in ~> toUpperCase ~> print[String] ~> out

  }

  g.run()

}
```

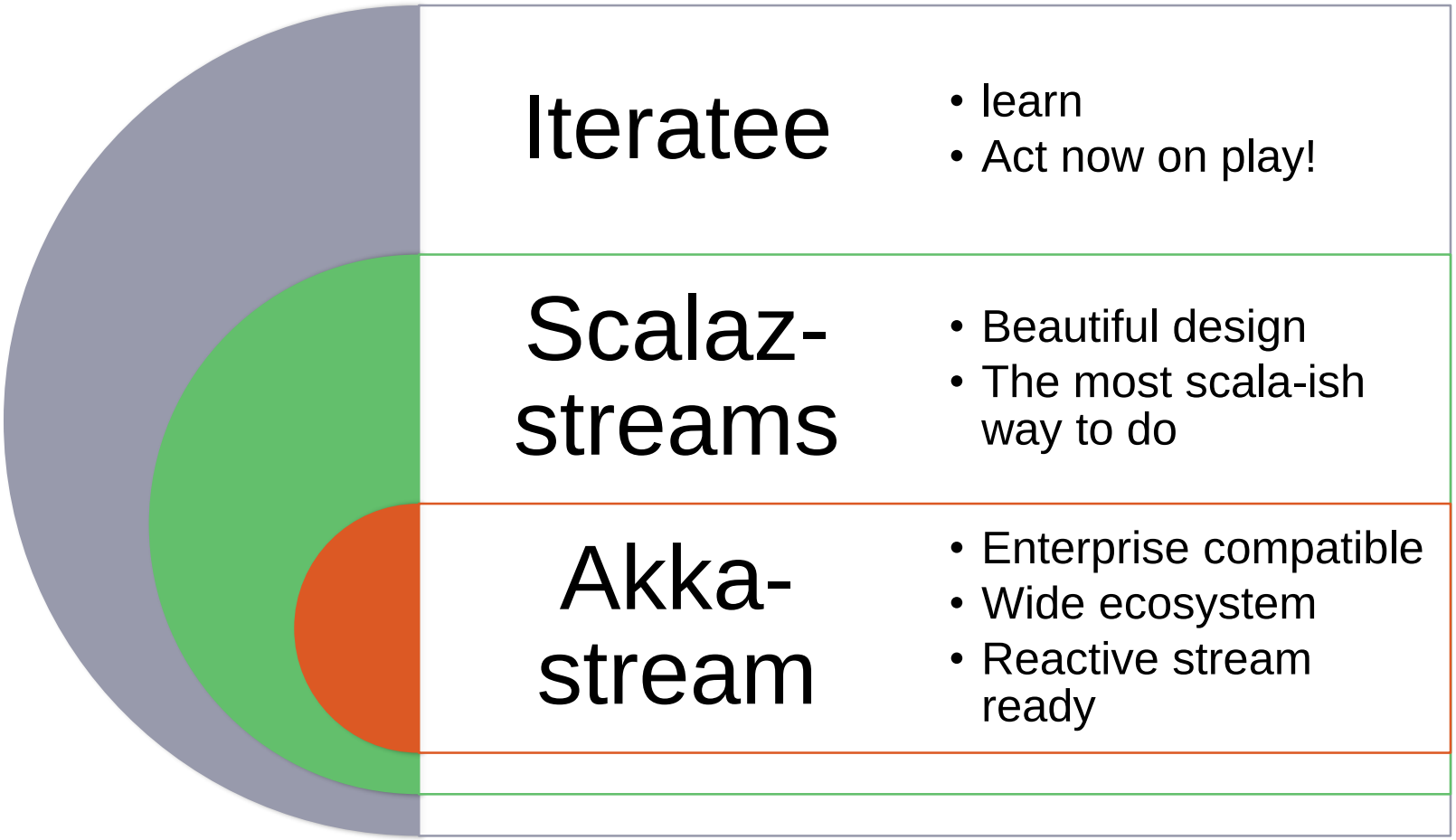
DEMO AGAIN

MORE POSSIBILITIES



ARE YOU CONVINCE TO USE STREAM ?

GO NOW


Iteratee
• learn
• Act now on play!
Scalaz-streams
• Beautiful design
• The most scala-ish way to do
Akka-stream
• Enterprise compatible
• Wide ecosystem
• Reactive stream ready

DATA STREAMING PROGRAMMING IS NOW TRENDING

THX TO @CLEMENTD
CLEMENT DELAFARGUE

THX FOR LISTENING & QUESTIONS TIME

I'm @waxzce on twitter

I'm the CEO of

clever cloud

A PaaS provider, give it a try ;-)

Coupon for Clever Cloud trial :

lambdacon