TURN IDEAS INTO REALITY

BUILDING STARTUPS ON SCALA

buildo

WHO ARE YOU?

buildo, a young italian company

we help **startups** everywhere
execute on their vision

OUR PILLARS

SUSTAINABILITY

FAIRNESS

QUALITY


LET’S TAKE A STEP BACK

A LONG TIME AGO
IN A GALAXY FAR FAR AWAY

- a wild **scary** large project appears!

FLASHBACK: NOT LONG BEFORE

- Intern at Google, MongoDB
- Freelancer
  - C++
  - Python (programming is fun again)
  - Java (university)
  - Haskell
  - Scala

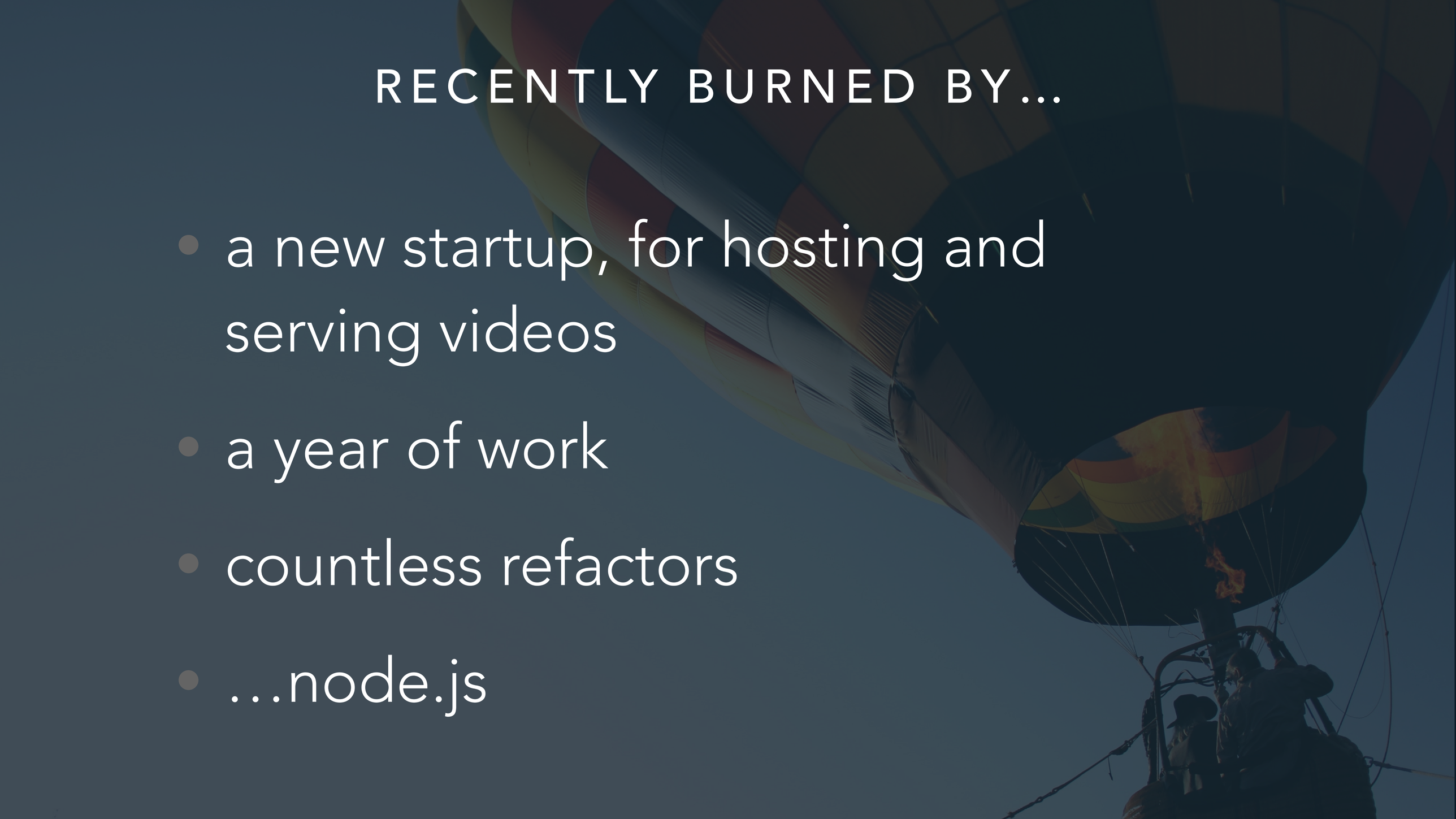
FLASHBACK: NOT LONG BEFORE

- Startupper and freelance dev
  - Java (university)
  - Objective-C <3
    - “blocks? That’s confusing…”
  - intrigued by Haskell
  - Scala (Martin’s online course)

THE REST OF THE (INITIAL) CREW

RECENTLY BURNED BY...

- a new startup, for hosting and serving videos
- a year of work
- countless refactors
- …node.js



BACK TO OUR SCARY PROJECT

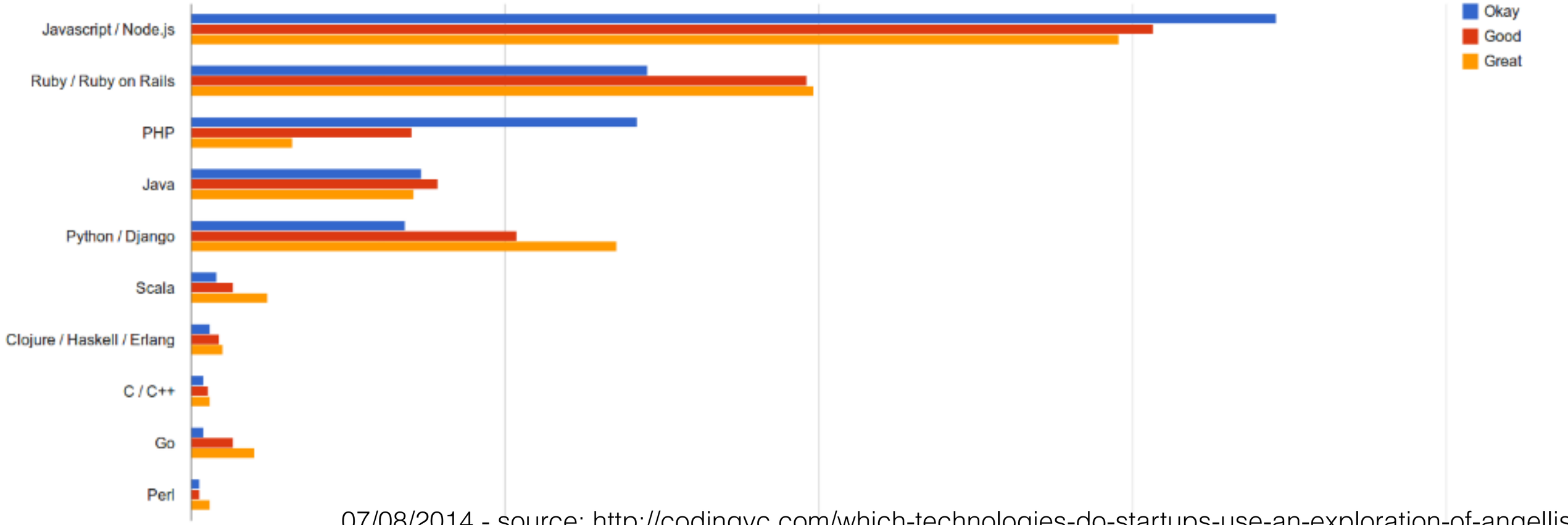
- we wanted
  - type safety
  - expressiveness
  - flexibility


FAST-FORWARD TO TODAY

BUILDO, TODAY

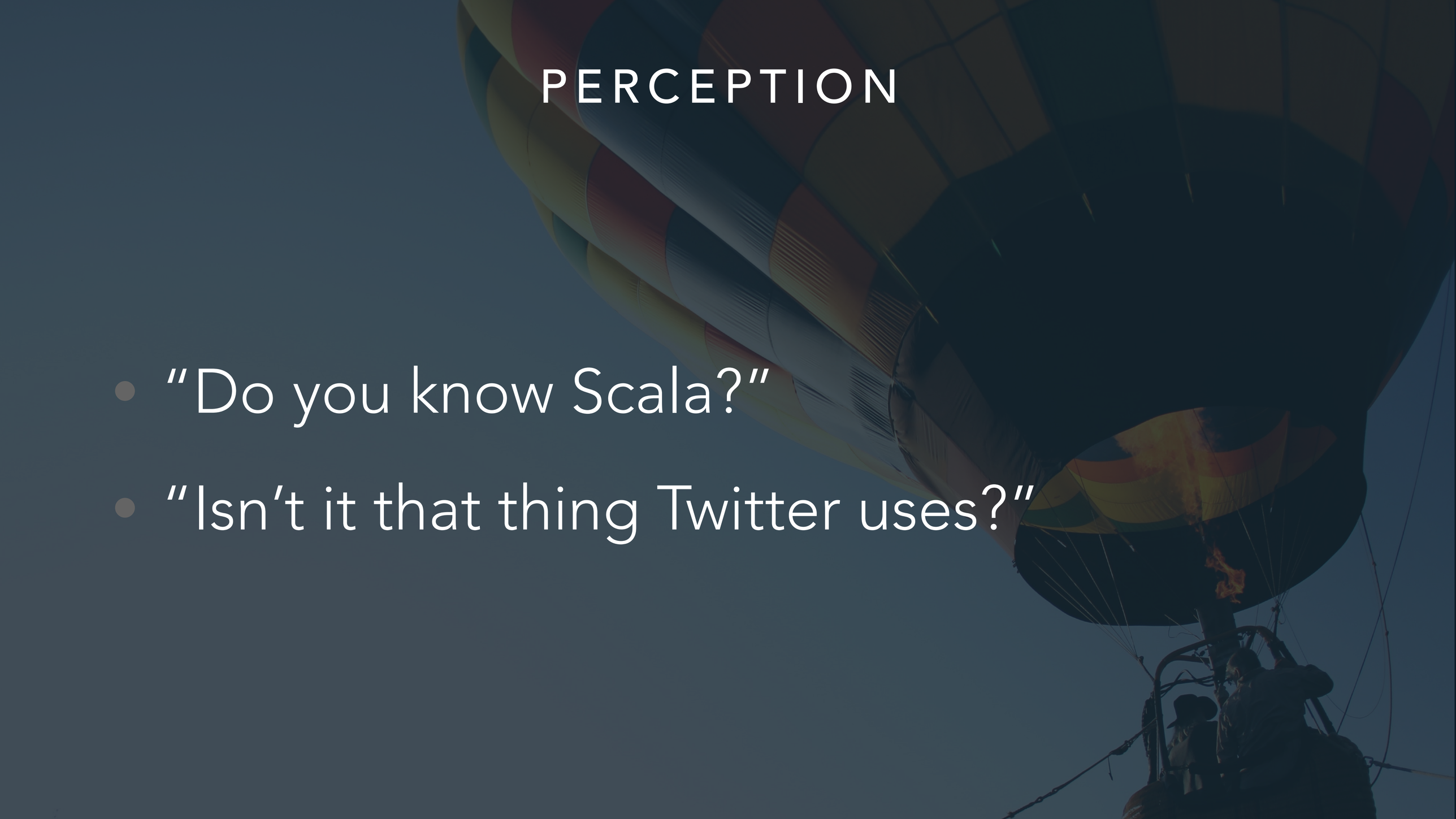
- startups, in scala?!
- everybody knows rails/node/php is the hipster startup language!
- waxed mustache for extra points

STARTUP TECHNOLOGIES



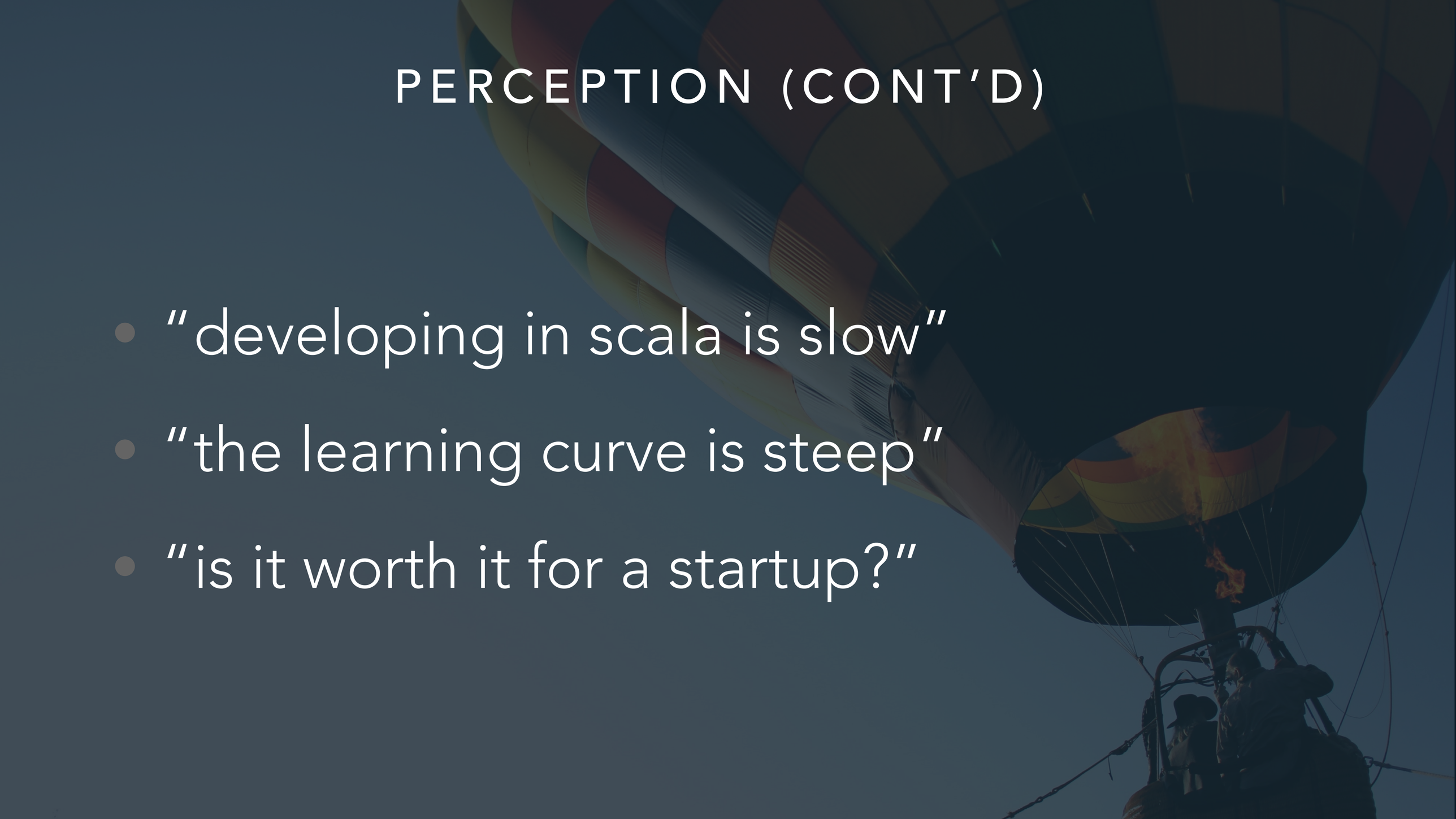
07/08/2014 - source: http://codingvc.com/which-technologies-do-startups-use-an-exploration-of-angellis

PERCEPTION



- “Do you know Scala?”
- “Isn’t it that thing Twitter uses?”

PERCEPTION (CONT'D)

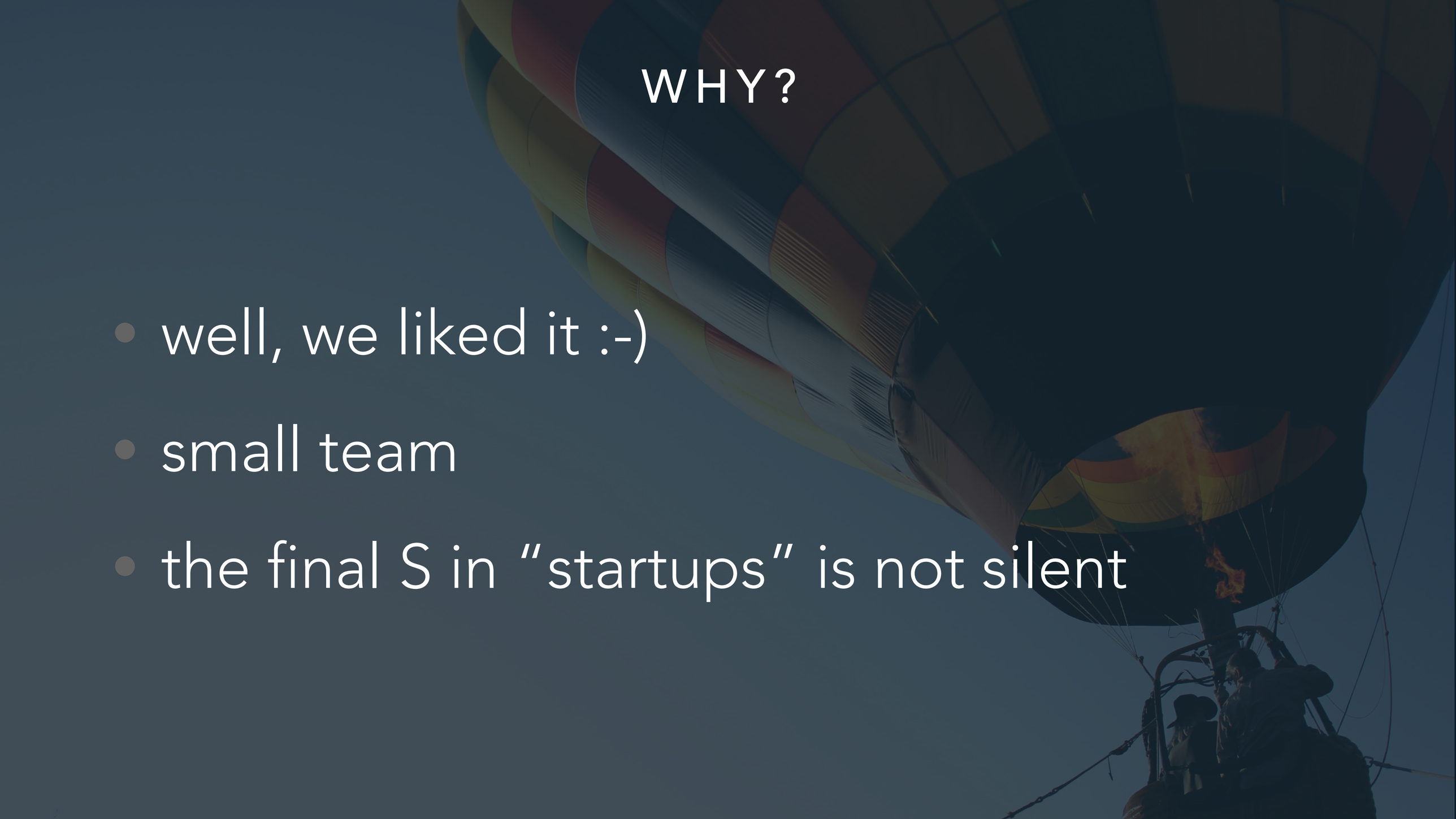


- "developing in scala is slow"
- "the learning curve is steep"
- "is it worth it for a startup?"

SCALING SCALA DOWN

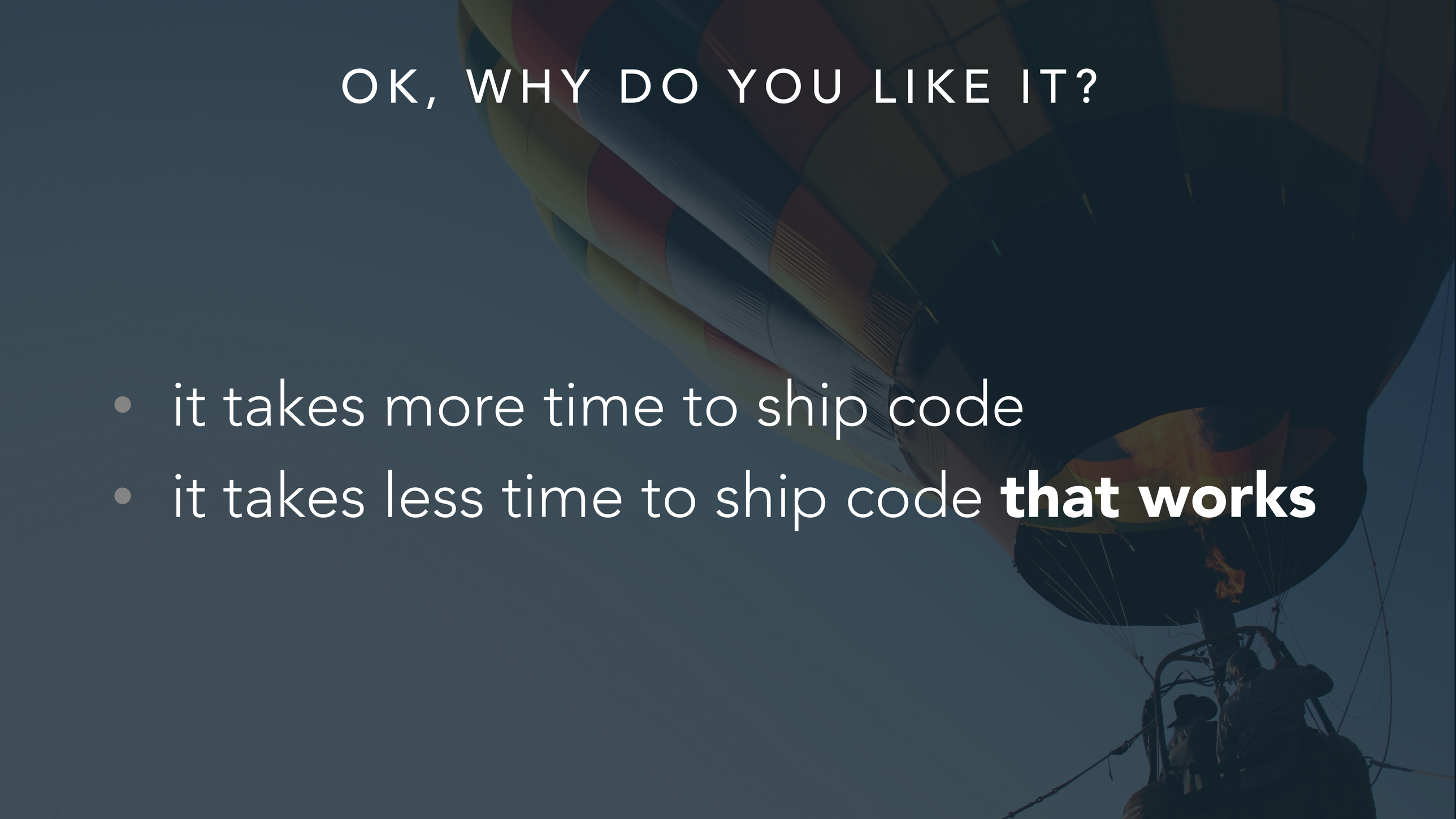
- we started with a large project
- and applied what we learned to startups

WHY?



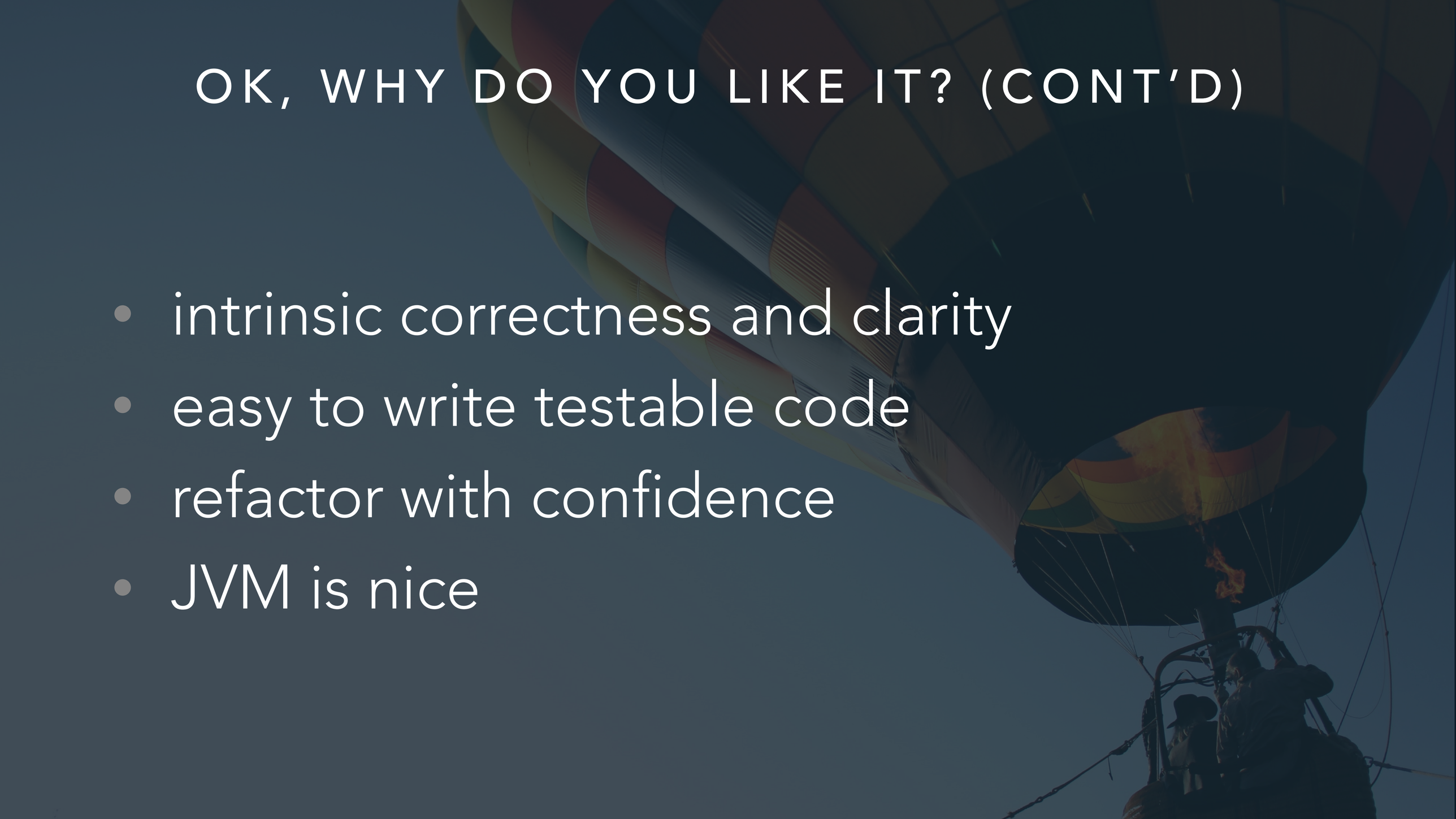
- well, we liked it :-)
- small team
- the final S in “startups” is not silent

OK, WHY DO YOU LIKE IT?



- it takes more time to ship code
- it takes less time to ship code **that works**

OK, WHY DO YOU LIKE IT? (CONT'D)



- intrinsic correctness and clarity
- easy to write testable code
- refactor with confidence
- JVM is nice

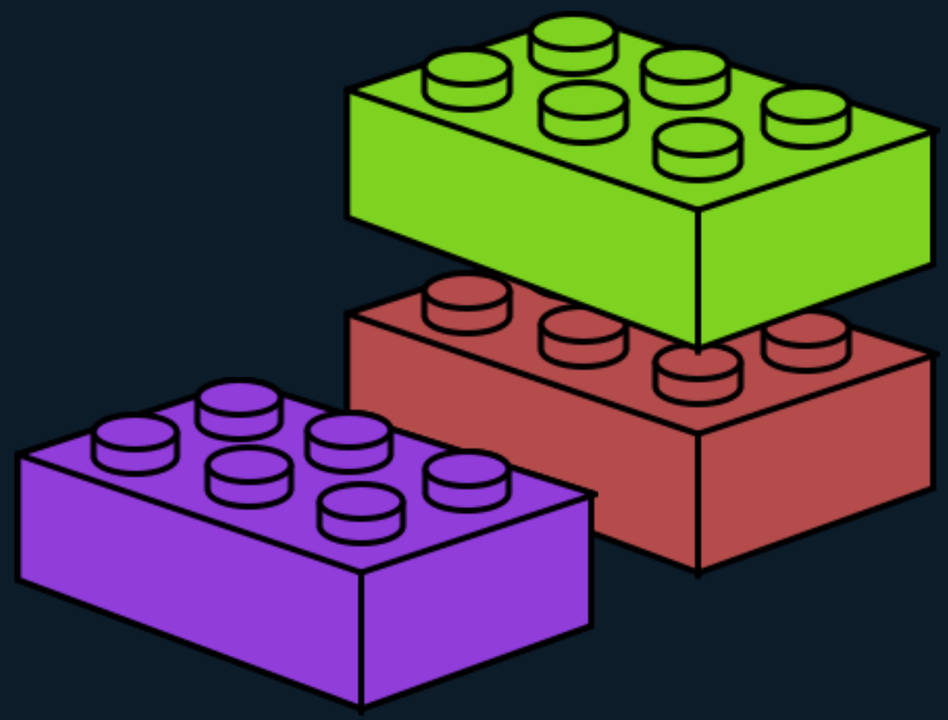

behind the scenes

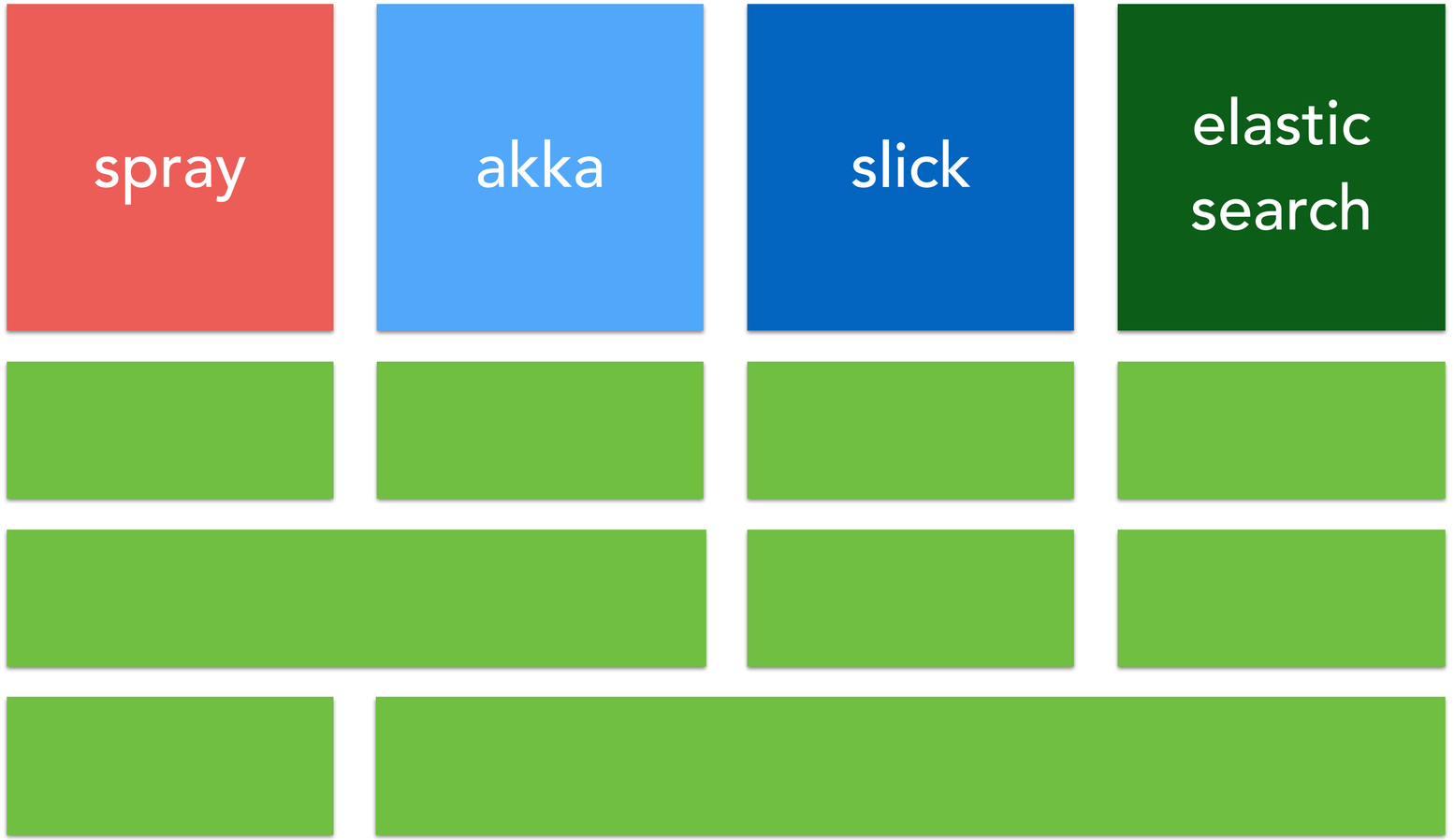
HOW DO WE DO THIS?

1. dealing with a diverse and evolving set of requirements
2. akka
3. controllers
4. open problems

1. DEALING WITH A DIVERSE SET OF REQUIREMENTS

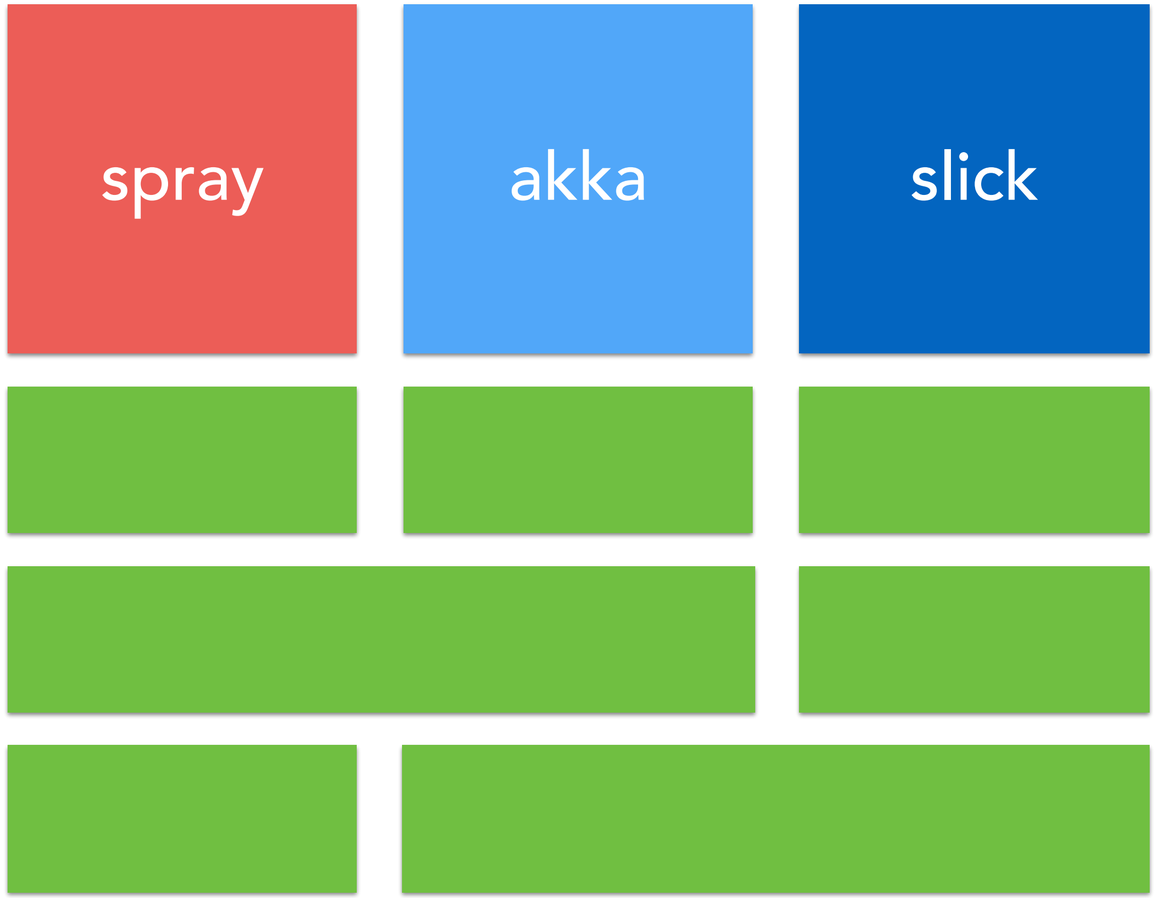
MIX-MATCH INFRASTRUCTURE



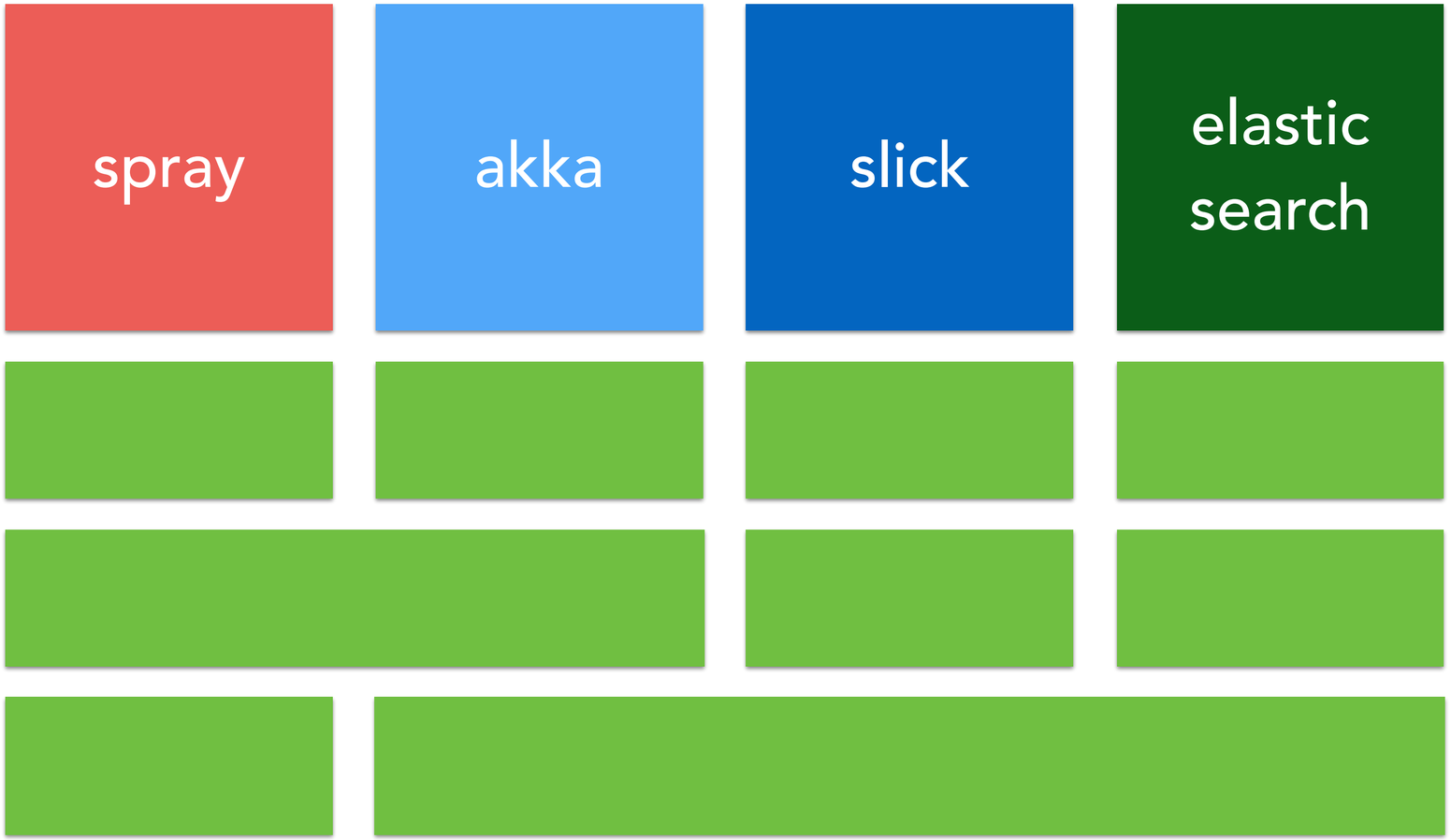

spray
akka
slick
elastic
search

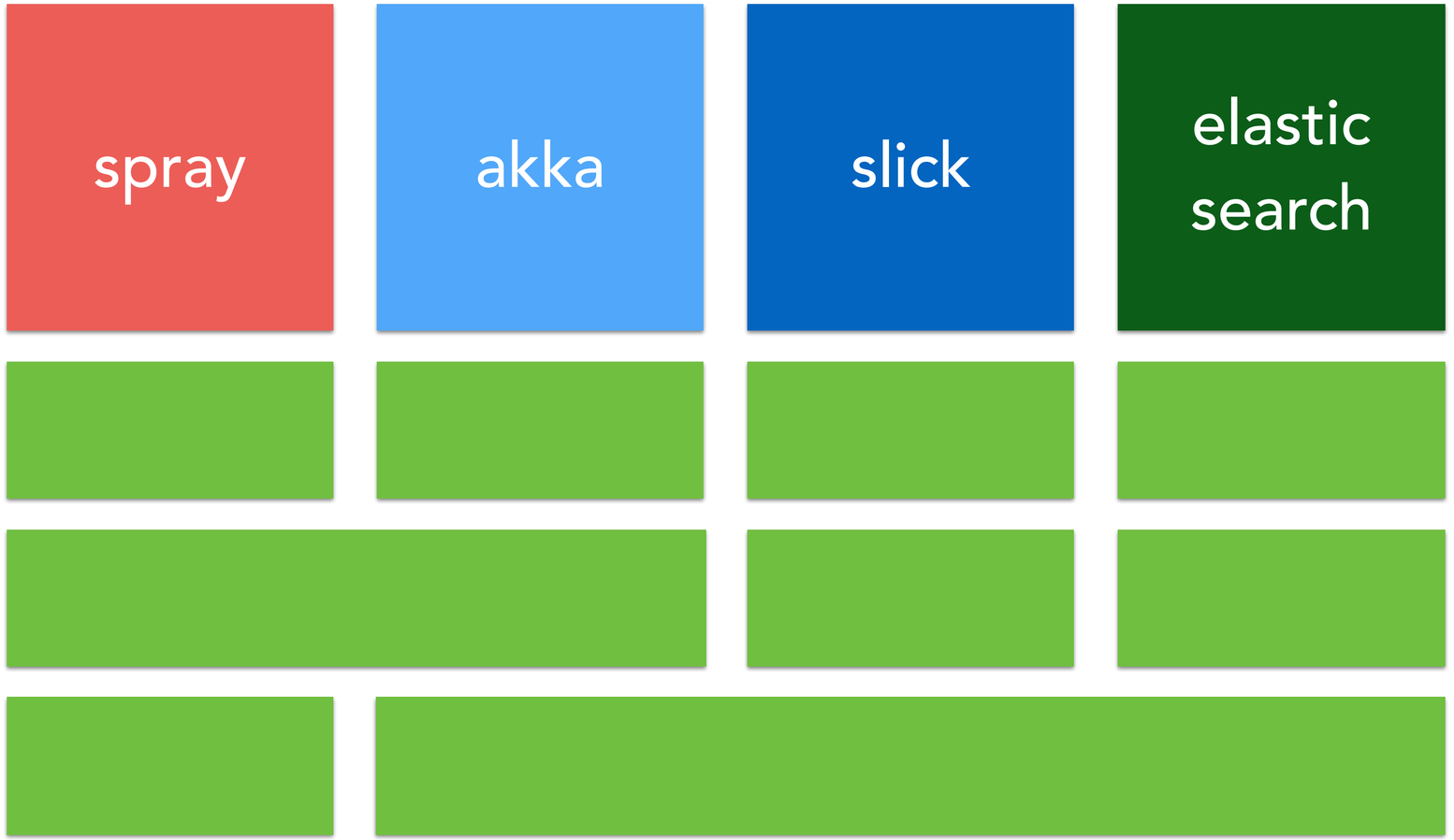
early stage start-up,
simple requirements

innovation,
evolved requirements

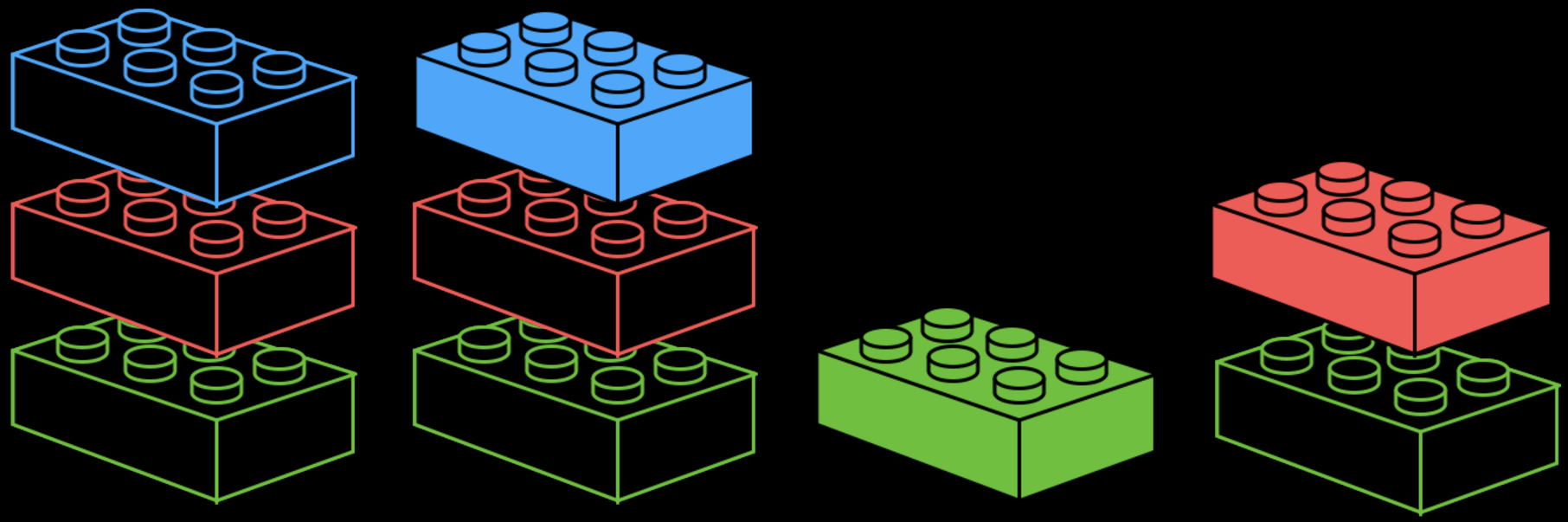


complex system

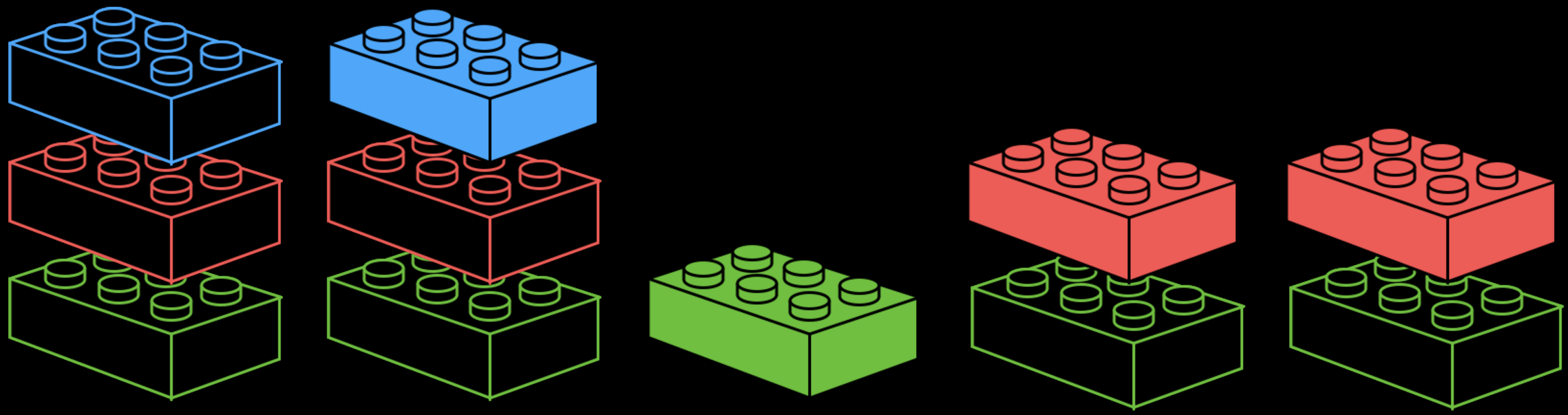



spray
akka
slick
elastic
search

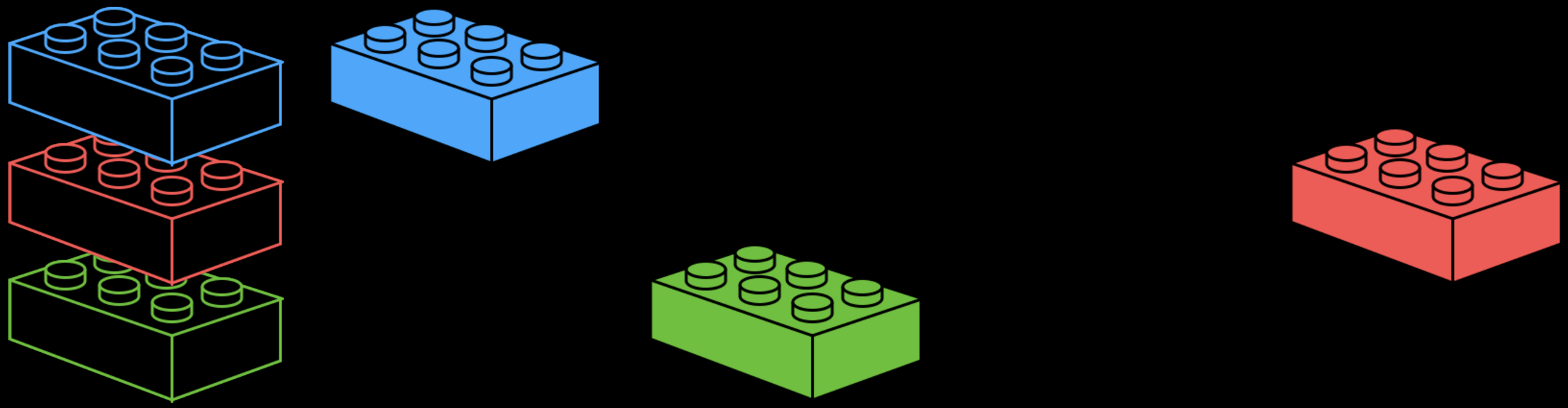
CAKE PATTERN



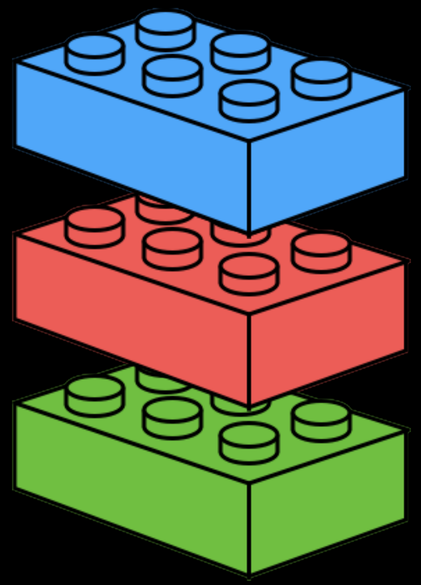
CAKE PATTERN



CAKE PATTERN



CAKE PATTERN



cake pattern

```
trait UserService extends DataModule {
  trait UserServiceDef {
    def login(username: String, password: String): Option[Token]
  }

  def userService: UserServiceDef
}
```

cake pattern

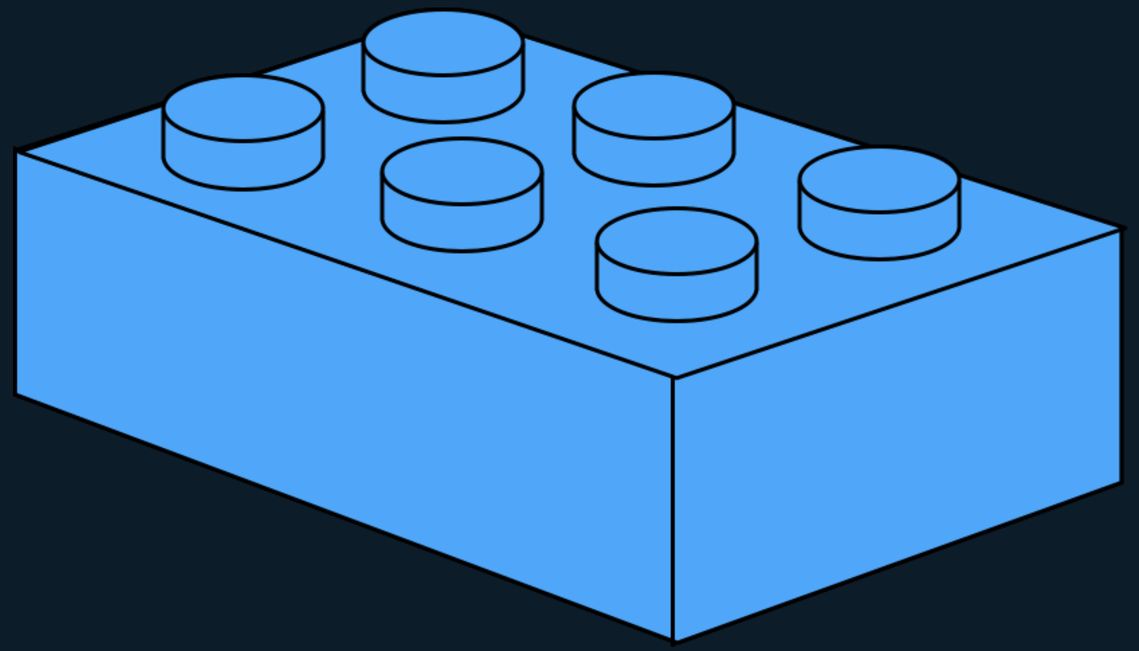
```
trait DataModule {
  type Token

  trait DataModuleDef {
    def getUserByName(username: String): Option[User]
    def storeToken(user: User, token: Token): Boolean
  }

  def dataModule: DataModuleDef
}

trait UserService extends DataModule {
  trait UserServiceDef {
    def login(username: String, password: String): Option[Token]
  }

  def userService: UserServiceDef
}
```

```
trait DataModule {
  type Token

  trait DataModuleDef {
    def getUserByName(username: String): Option[User]
    def storeToken(user: User, token: Token): Boolean
  }

  def dataModule: DataModuleDef
}

trait MemoryDataModule { self: DataModule =>
  type Token = String

  object dataModule extends DataModuleDef {
    val users = new scala.collection.mutable.Map[String, User]
    val tokens = new scala.collection.mutable.Map[User, Token]
    def getUserByName(username: String): Option[User] =
      users.get(username)
    def storeToken(user: String, token: Token): Boolean =
      tokens += user -> token
  }
}
```

```
trait MemoryDataModule { self: DataModule => … }

trait AppUserService { self: UserService => … }

object app extends
    MemoryDataModule
  with AppUserService

println(app.login("utaal", "secretpassword"))
```

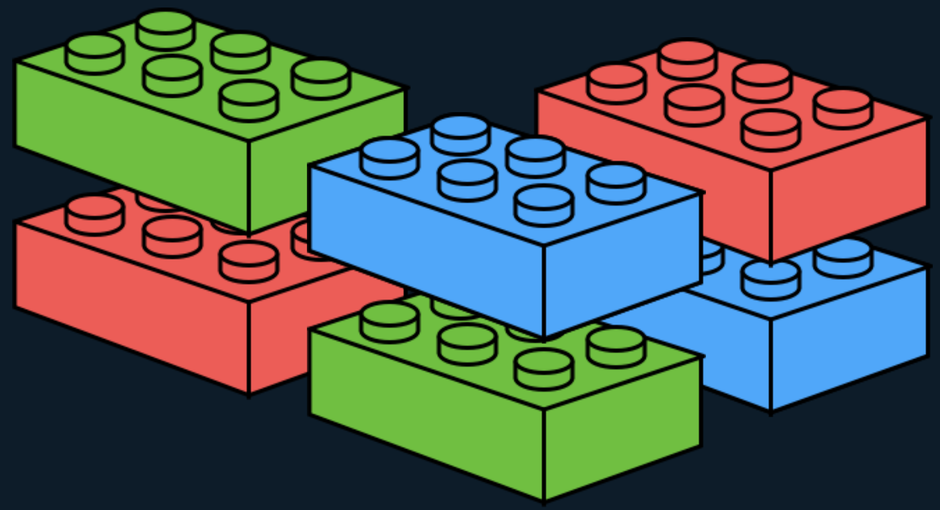

router
controller
mysql
mongodb

FRAMEWORKS



NO, THANKS

github.com/buildo/ingredients



typesafe router

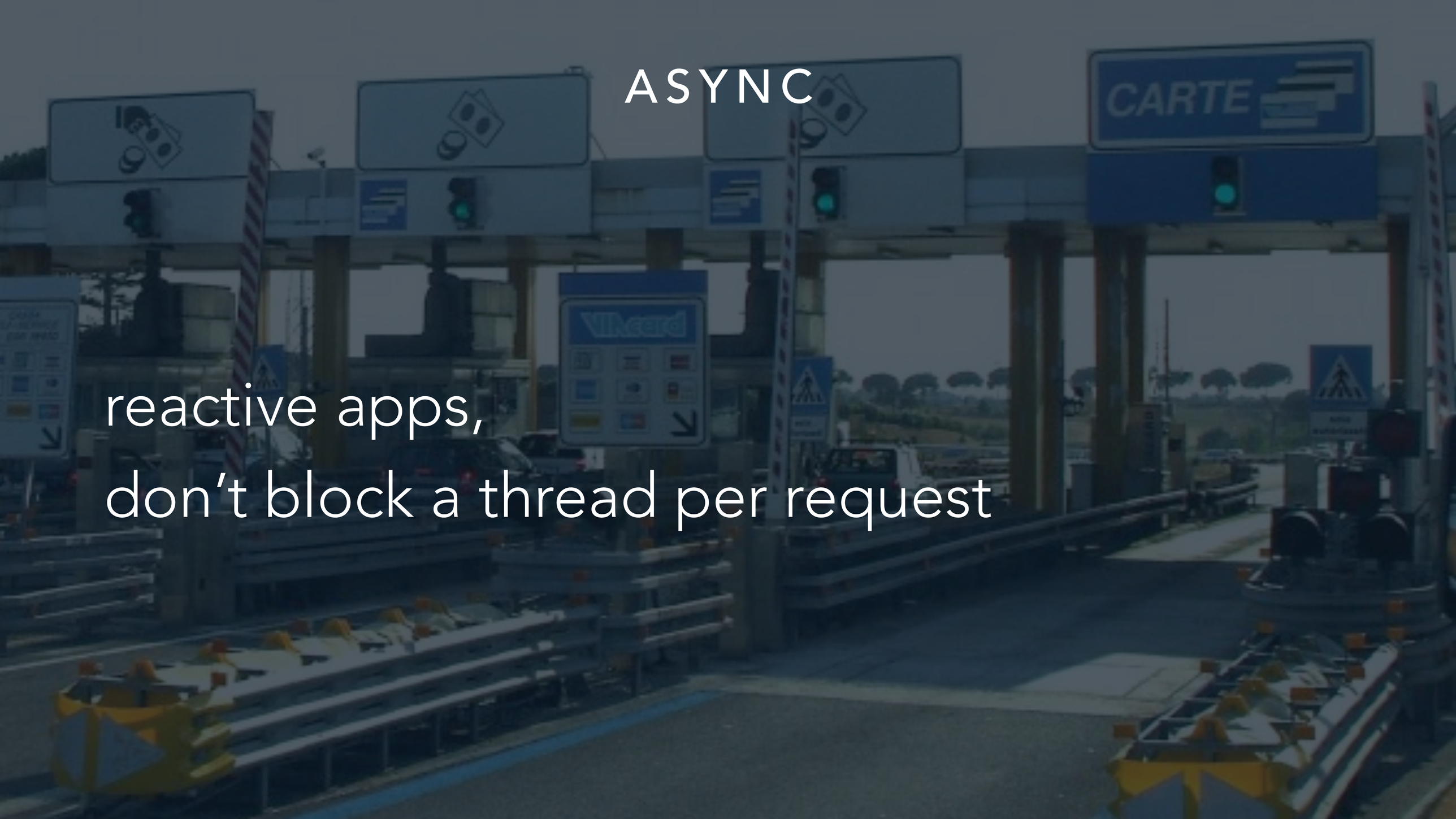
```
// POST /consultation/15/confirm
(post & path(“consultations” / IntNumber / “confirm”)) { consultationId =>
  complete(confirmConsultation(consultationId))
} ~
```

1. DEALING WITH A DIVERSE SET OF REQUIREMENTS

- cake pattern
- tests
- type safety

2. AKKA

ASYNC



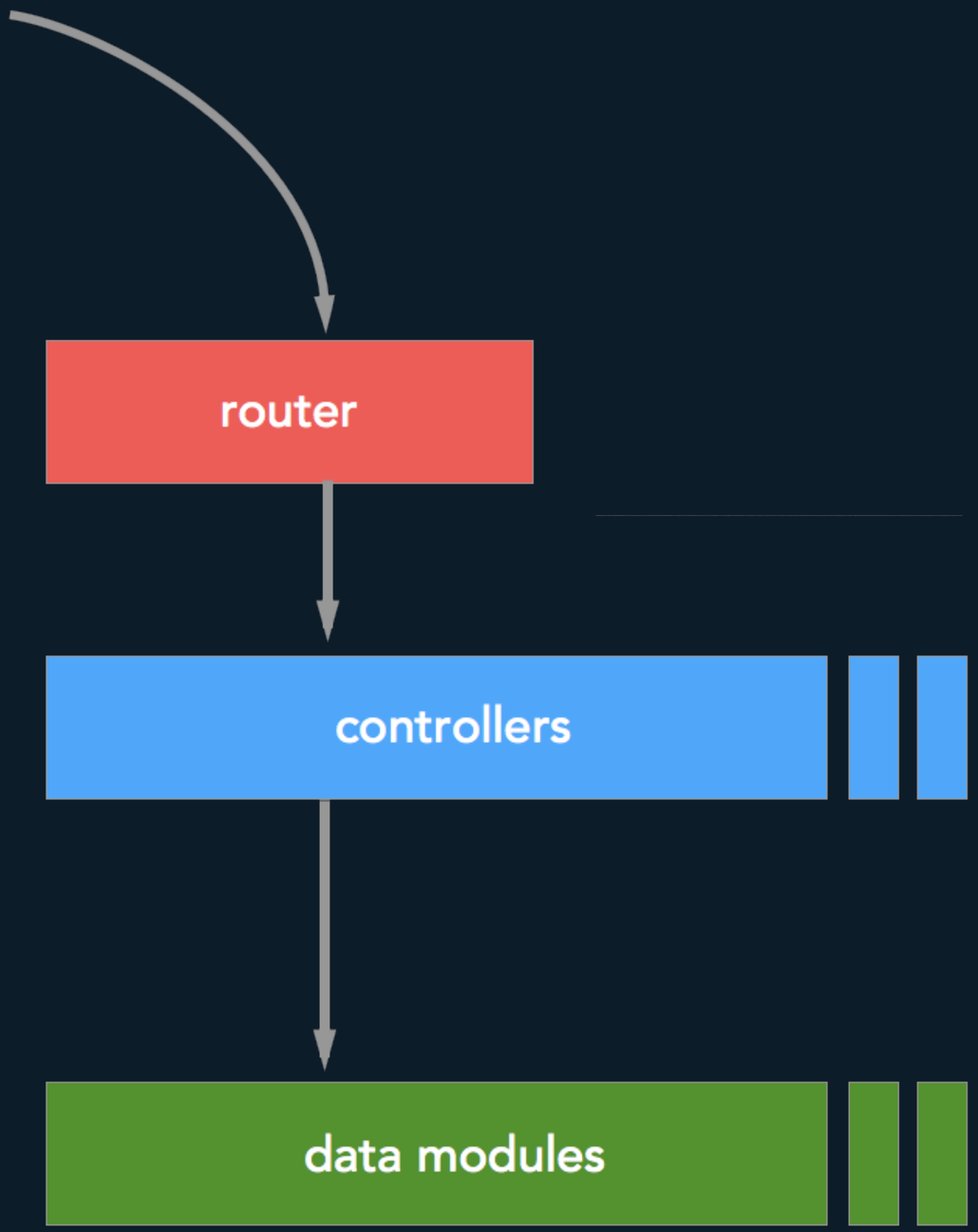
reactive apps,
don’t block a thread per request

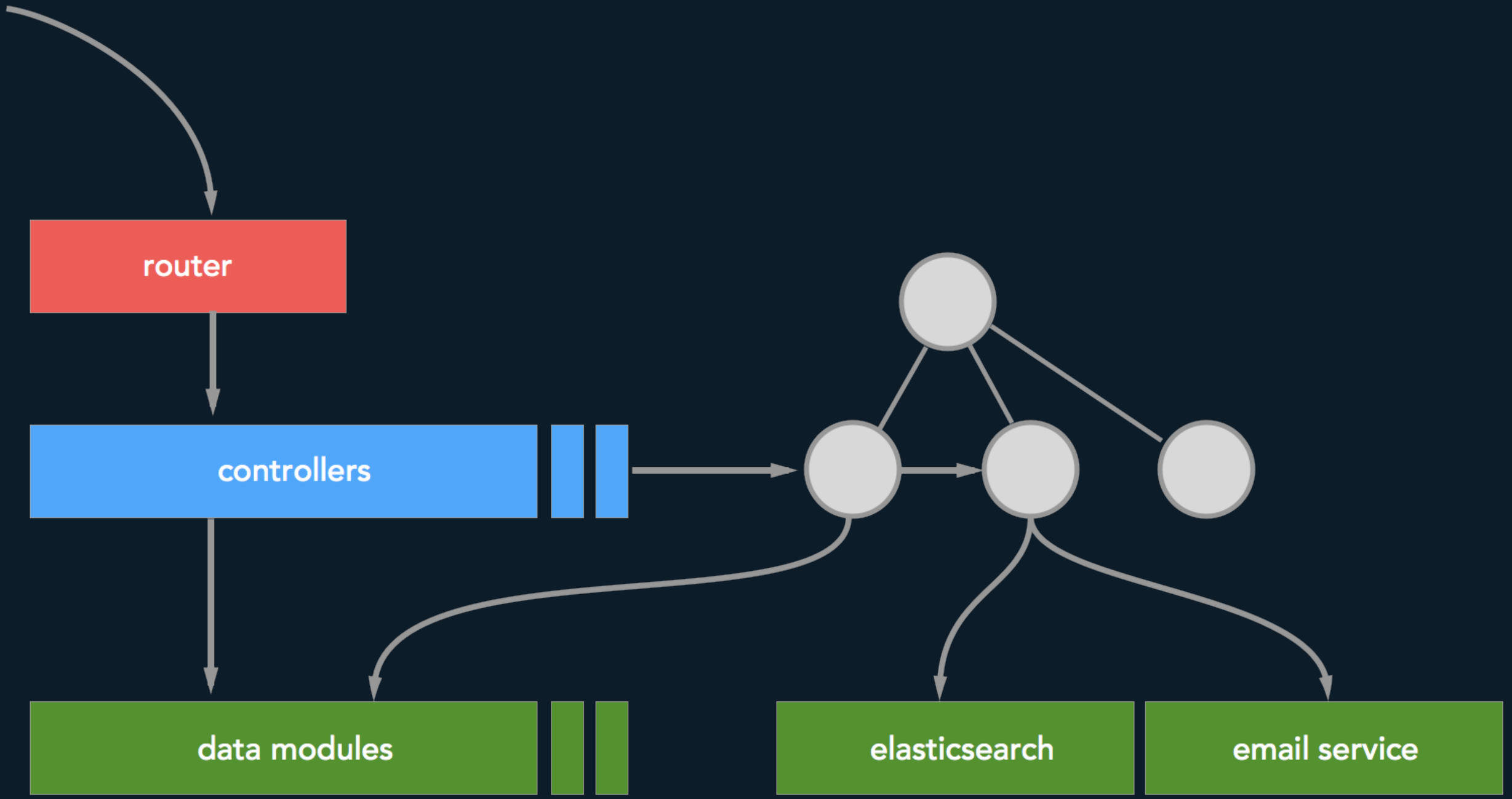
AKKA

akka tell “!” is awkward with request-response

akka ask “?” gets you Futures that are not bound to an Actor

just use Futures


router
controllers
data modules


router
controllers
data modules
elasticsearch
email service

2. AKKA

- use akka for dispatching requests (in spray)
- Futures for business logic
- Akka for async tasks (notifications, elasticsearch updates)

3. CONTROLLERS

typesafe router

```
trait ConsultationControllerDef {
  def confirmConsultation(consultationId: Int):
    Future[ControllerResponse[Unit]]
}

// POST /consultation/15/confirm
(post & path(“consultations” / IntNumber / “confirm”)) { consultationId =>
  complete(
    (confirmConsultation(consultationId))
      : Future[ControllerResponse[Unit]]
  )
} ~
```

response semantics

```
sealed trait ControllerResponse[+T]

object ControllerResponse {
  sealed trait ControllerError

  case object NotFound extends ControllerError
  case class InvalidOperation(desc: String) extends ControllerError
  case class Forbidden(desc: String) extends ControllerError
  … … …

  case class Ok[T](value: T) extends ControllerResponse[T]
  case class UserError(err: ControllerError) extends ControllerResponse[Nothing]
}
```

controller workflow

```
def confirmConsultation(consultationId: Int, user: User):
  Future[ControllerResponse[Unit]] =
```

```
checkConsultationData.updateStatus(
  consultationId,
  ConsultationStatus.Confirmed) map (
    throwOnUnmatched {
      case DataActionResult.Ok(_) => OkEmpty
    })
```

controller workflow

```
def confirmConsultation(consultationId: Int, user: User):
  Future[ControllerResponse[Unit]] =

  checkRole(user, UserRole.Patient) {
    checkConsultationExists(consultationId) { c =>
      checkConsultationOwner(c, user) {
        checkConsultationPending(c) {
          checkConsultationData.updateStatus(
            consultationId,
            ConsultationStatus.Confirmed) map (
              throwOnUnmatched {
                case DataActionResult.Ok(_) => OkEmpty
              })
        }
      }
    }
  }
```

CALLBACK HELL

monadic controllers

```
type CtrlFlow[A] = \/[CtrlError, A]

sealed trait CtrlError

object CtrlError {
  case class InvalidOperation(desc: String) extends CtrlError
  case class Forbidden(desc: String) extends CtrlError
  case object NotFound extends CtrlError
  … … …
}
```

monadic controllers

```
type FutureCtrlFlow[B] = EitherT[Future, CtrlError, B]
```

```
def confirmConsultation(consultationId: Int, user: User): FutureCtrlFlow[Unit] =
  for {
    _ <- eitherT(checkRole(user, UserRole.Patient).point[Future])
    c <- checkConsultationExists(consultationId)
    _ <- eitherT(checkConsultationOwner(c, user).point[Future])
    _ <- eitherT(checkConsultationPending(c).point[Future])
    res <- eitherT(checkConsultationData.updateStatus(consultationId,
             ConsultationStatus.Confirmed).map(_.point[CtrlFlow]))
  } yield {
    throwIfNotOk(res)
    Ok(())
  }
```

Li Haoyi
@li_haoyi

Programming #Scala isn't about being clever, it's setting things up so you can bang mindlessly against the computer and still make progress.

RETWEETS 17 FAVORITES 22

6:42 AM - 3 May 2015

3. COMPLEX CONTROLLER LOGIC

- monad!
- non-trivial to design, straightforward to use

4. OPEN PROBLEMS

MODULARITY

- modularity comes with a cost
- boilerplate to translate between internal and external representations
- we can’t trace a request across different layers

PEOPLE

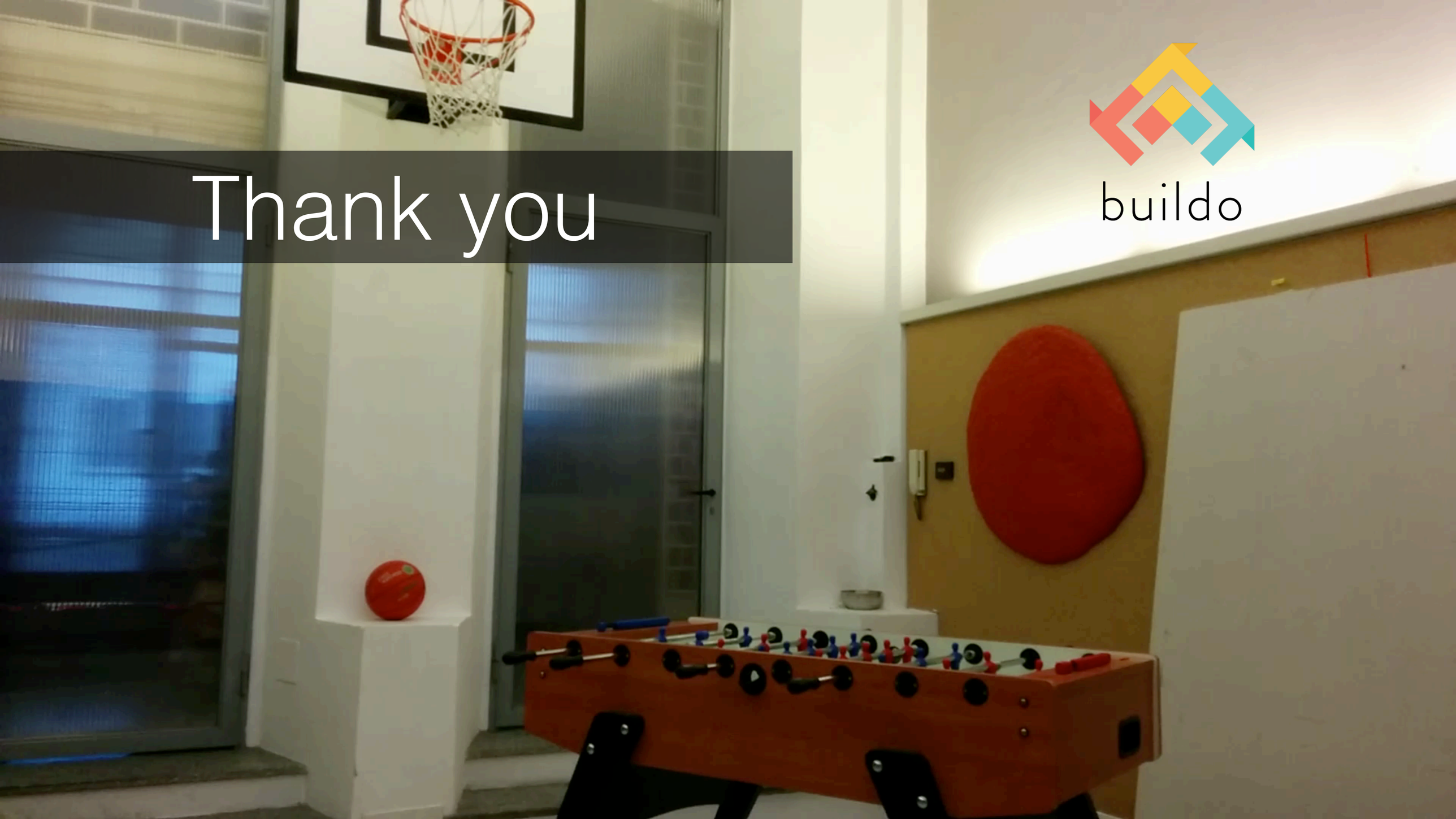
- you can’t throw developers at this
- you need good programmers
- scala is (still) a good hiring signal

Good programmers are up to 30 times better than mediocre programmers, according to "individual differences" research.

Given that their pay is never commensurate, they are the biggest bargains in the software field.

Robert L. Glass

@milanoscala


Thank you
buildo