

Reactive Slick for Database Programming

Stefan Zeiger



Introduction

Slick 3.0 – *Reactive Slick*

- Completely new API for executing database actions
- Old API (*Invoker, Executor*) deprecated
 - Will be removed in 3.1
- Execution is asynchronous
(*Futures, Reactive Streams*)

Application Performance

- Keep the CPU busy



The Problem With Threads

- Context Switching is expensive
- Memory overhead per thread
- Lock contention when communicating between threads

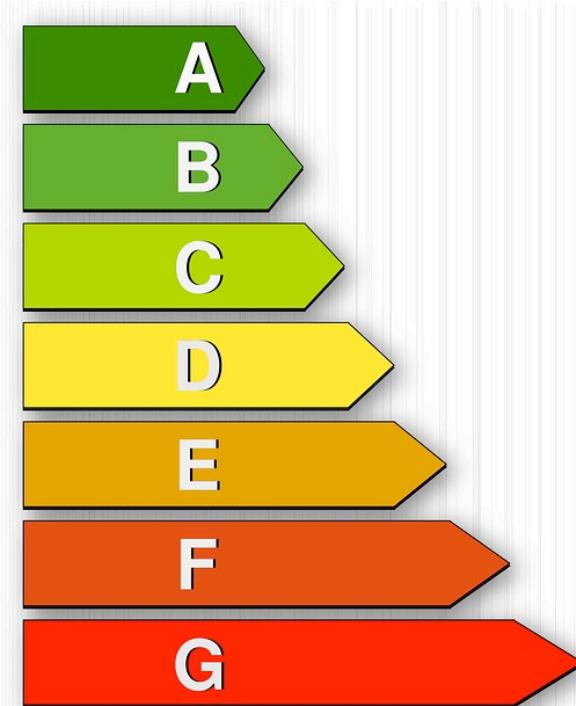
Does not scale!

Application Performance

- Keep the CPU busy



- Be efficient

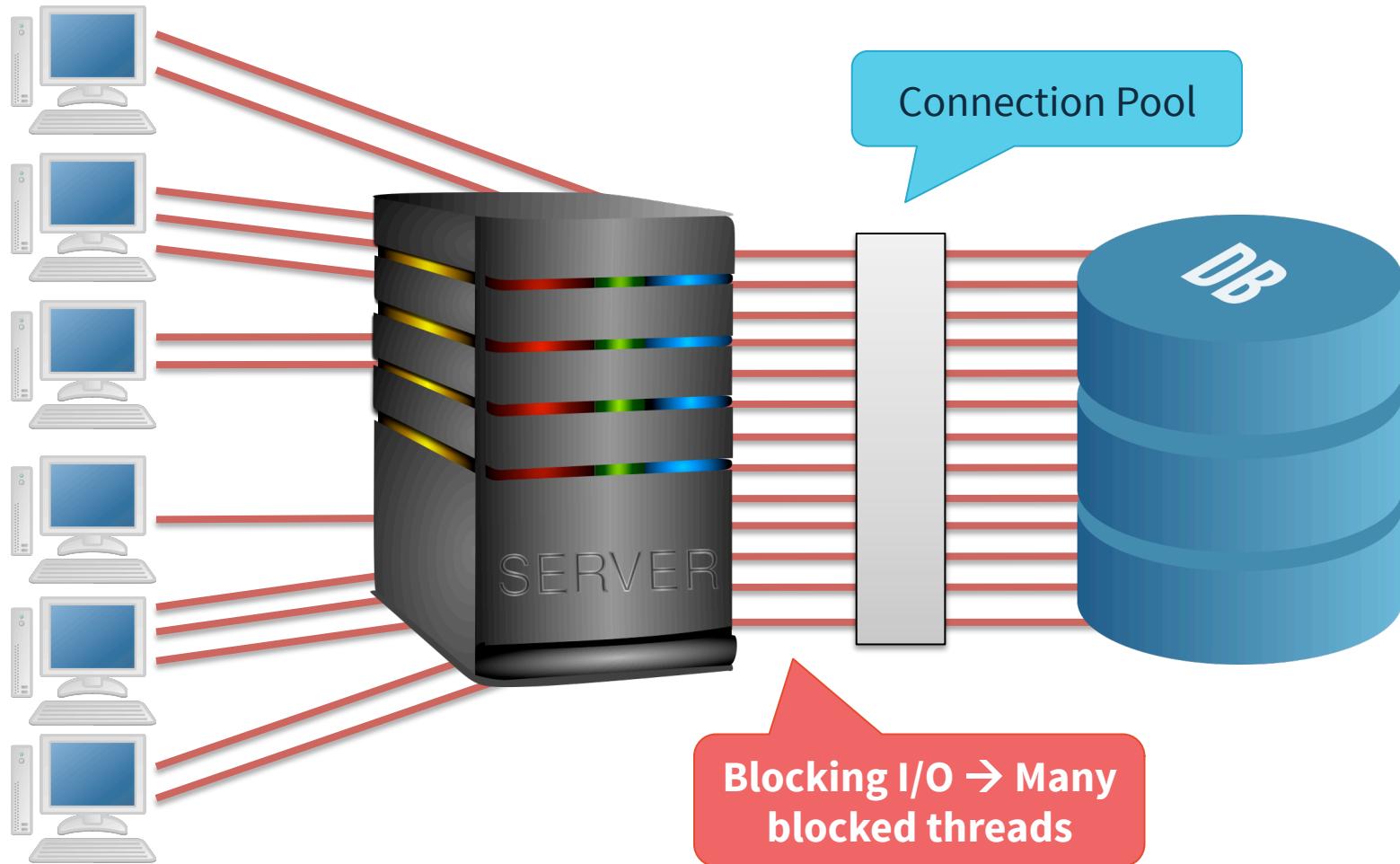


Blocking I/O

- JDBC is inherently blocking (and blocking ties up threads)
- How much of a problem is it really?

Connection Pools

Web Application Architecture: Connections



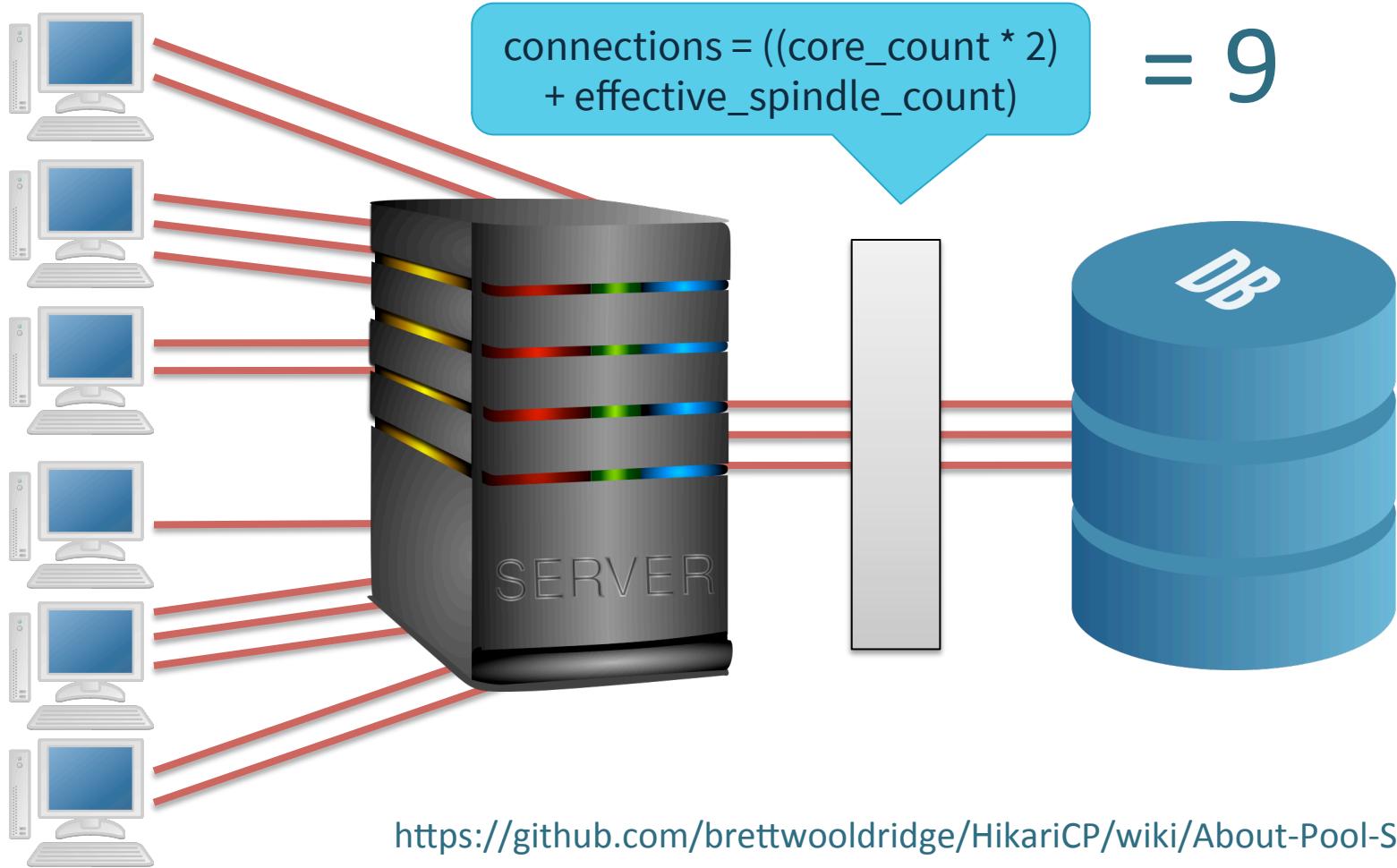
Quiz: Connection Pool Size

- Database server: Latest i7-based Xeon, 4 cores (8 with HyperThreading)
- 2 enterprise-grade 15000 RPM SAS drivers in RAID-1 configuration
- Beefy app server
- 10.000 concurrent connections from clients

What is a good connection pool size?

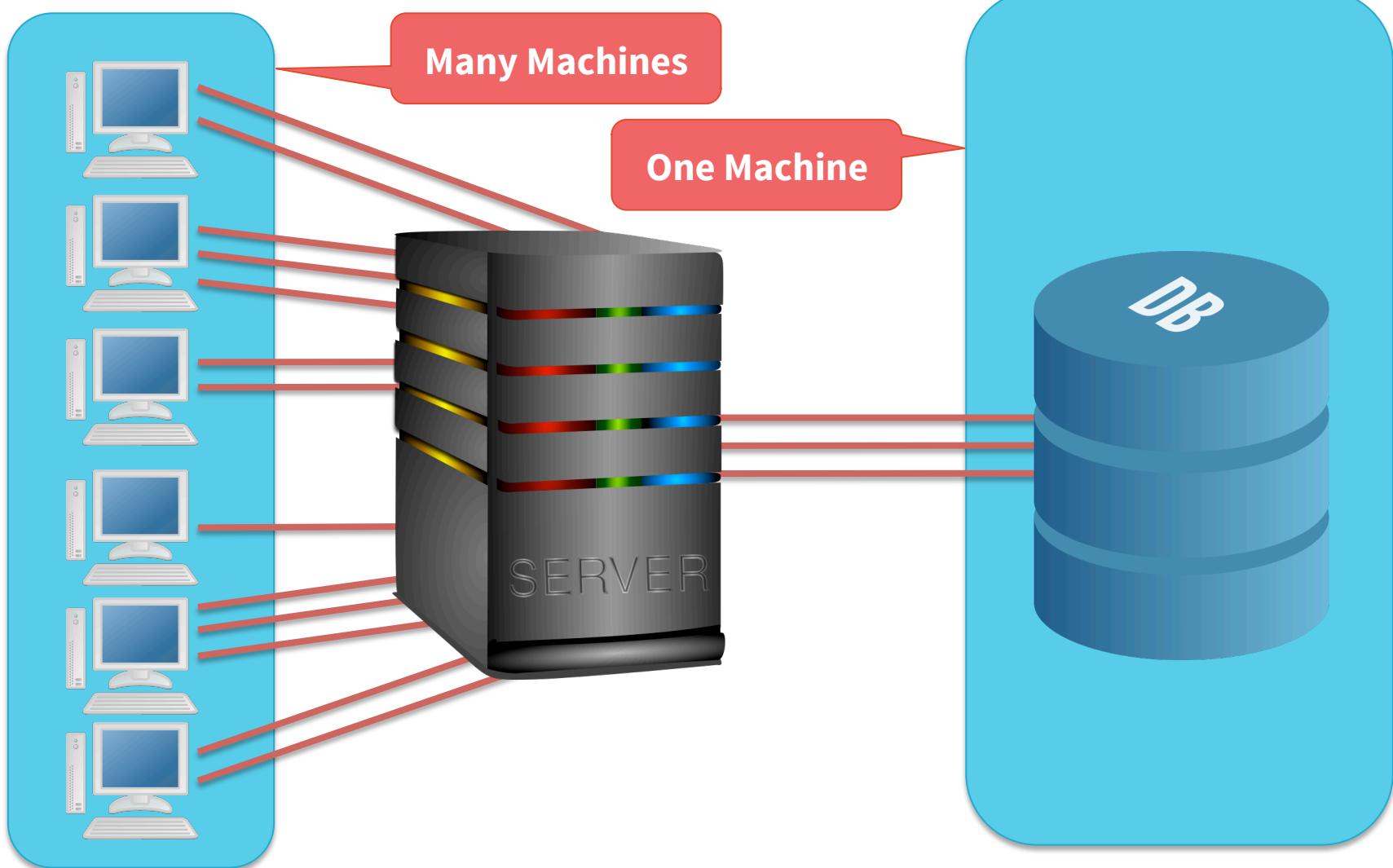
- 10
- 100
- 1.000
- 10.000

Web Application Architecture: Connections



Threading Models

Blocking Web Server Doesn't Scale – But DB Can



The Traditional Model (e.g. JEE)

- Fully synchronous
- One thread per web request
- **Contention for Connections** (`getConnection` blocks)
- Database back-pressure creates more blocked threads
- Problem: Doesn't scale

Asynchronous Web App: Naive Approach

- Blocking database calls in `Future(blocking(...))`
 - **Contention for Connections**
(but may be limited by the `ExecutionContext`)
 - A saturated thread pool blocks *all* I/O
-
- Problem: Scalability depends on correct configuration of `ExecutionContext` and connection pool
 - Back-pressure on one kind of I/O stops other kinds from working

Asynchronous Web App: *Play-Slick Plugin*

- Special *ExecutionContext* per database
 - Thread pool size limited by connection pool size
- **Contention for Threads**

Remaining Problems

- No clean separation of I/O and CPU-intensive work:

```
table1.insert(table2.filter(...))(session)
```

- Streaming with back-pressure handling either blocks or has a lot of overhead (everything done through Future)
- Resource management is hard to get right with asynchronous code:

```
db.withSession { session => Future(...) }
```



Because of explicit mutable state

Pure Functional I/O

THIS OBJECT IS
JUST A MONOID IN
THE CATEGORY OF
ENDOFUNCTORS

What is a Monad?

*In functional programming, a monad is **a structure that represents computations defined as sequences of steps**: a type with a monad structure defines what it means to chain operations, or nest functions of that type together. This allows the programmer to build pipelines that process data in steps, in which each action is decorated with additional processing rules provided by the monad. As such, monads have been described as "**programmable semicolons**"*

(Wikipedia)

The State Monad

```
val st = for {
    i <- State.get[Int]
    _ <- State.set(i + 3)
    j <- State.get
    _ <- State.set(j - 2)
    k <- State.get
} yield k
```

```
def st = {
    i = get[Int] ;
        set(i + 3) ;
    j = get ;
        set(j - 2) ;
    k = get ;
    return k
}
```

State.run(41, st) → 42

The *State* Monad



The *State* Monad



The State Monad

```
trait State[S, R] extends (S => (S, R))

object State {
  def apply[S, R](v: R): State[S, R] = new State[S, R] {
    def apply(s: S) = (s, v)
  }

  def get[S]: State[S, S] = new State[S, S] {
    def apply(s: S) = (s, s)
  }

  def set[S](v: S): State[S, Unit] = new State[S, Unit] {
    def apply(s: S) = (v, ())
  }

  def run[S, R](s: S, st: State[S, R]): R = st(s)._2
}
```

The *State* Monad

```
trait State[S, R] extends (S => (S, R)) { self =>

  def flatMap[R2](f: R => State[S, R2]): State[S, R2] =
    new State[S, R2] {
      def apply(s: S) = {
        val (s2, r) = self.apply(s)
        f(r)(s2)
      }
    }

  def map[R2](f: R => R2): State[S, R2] =
    flatMap[R2](r => State(f(r)))
}
```

The *IO* Monad

```
val io = for {
  i <- IO.get
  _ <- IO.set(i + 3)
  j <- IO.get
  _ <- IO.set(j - 2)
  k <- IO.get
} yield k
```

`new DB(41).run(io) → 42`

```
class DB(var i: Int) {
  def run[R](io: IO[R]): R = io(this)
}
```

The *IO* Monad

```
trait IO[R] extends (DB => R)

object IO {
    ...

    def set(v: Int): IO[Unit] = new IO[Unit] {
        def apply(db: DB) = db.i = v
    }
}
```

The *IO* Monad

```
trait IO[R] extends (DB => R) { self =>

  def flatMap[R2](f: R => IO[R2]): IO[R2] =
    new IO[R2] {
      def apply(db: DB) = f(self.apply(db))(db)
    }

  def map[R2](f: R => R2): IO[R2] =
    flatMap[R2](r => IO(f(r)))
}

}
```

Hiding The Mutable State

```
trait IO[R] extends (DB => R)
```

Hiding The Mutable State

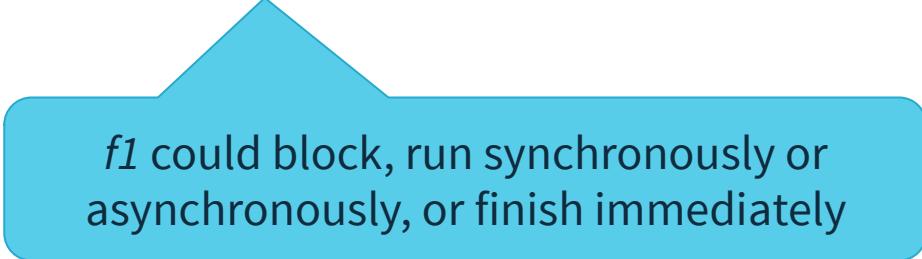
```
trait IO[R] {  
    def flatMap[R2](f: R => IO[R2]): IO[R2] =  
        new FlatMapIO[R2](f)  
}  
  
class FlatMapIO[R, R2](f: R => IO[R2]) extends IO[R2]  
  
class DB(var i: Int) {  
    def run[R](io: IO[R]): R = io match {  
        case FlatMapIO(f) => ...  
        case ...  
    }  
}
```

Asynchronous Programming

The Future Monad

- You already use monadic style for asynchronous programming in Scala
- Futures abstract over blocking:

f1.flatMap { _ => f2 }



f1 could block, run synchronously or asynchronously, or finish immediately

- But Futures are not sequential
 - Only their results are used sequentially

Asynchronous Database I/O

```
trait DatabaseDef {  
  
  def run[R](a: DBIOAction[R, NoStream, Nothing])  
    : Future[R]  
}
```

- Lift code into DBIO for sequential execution in a database session
- Run DBIO to obtain a Future for further asynchronous composition

DBIO Combinators

- ```
val a1 = for {
 _ <- (xs.schema ++ ys.schema).create
 _ <- xs ++= Seq((1, "a"), (2, "b"))
 _ <- ys ++= Seq((3, "b"), (4, "d"), (5, "d"))
} yield ()
```
- ```
val a2 =
  (xs.schema ++ ys.schema).create >>
  (xs ++= Seq((1, "a"), (2, "b")))) >>
  (ys ++= Seq((3, "b"), (4, "d"), (5, "d"))))
```


- ```
val a3 = DBIO.seq(
 (xs.schema ++ ys.schema).create,
 xs ++= Seq((1, "a"), (2, "b")),
 ys ++= Seq((3, "b"), (4, "d"), (5, "d")))
)
```

# Execution Contexts

```
trait DBIO[+R] { // Simplified

 def flatMap[R2](f: R => DBIO[R2])
 (implicit executor: ExecutionContext)
 : DBIO[R2] =
 FlatMapAction[R2, R](this, f, executor)

 def andThen[R2](a: DBIO[R2])
 : DBIO[R2] =
 AndThenAction[R2](this, a)

}
```

Fuse synchronous DBIO actions

# Streaming Results

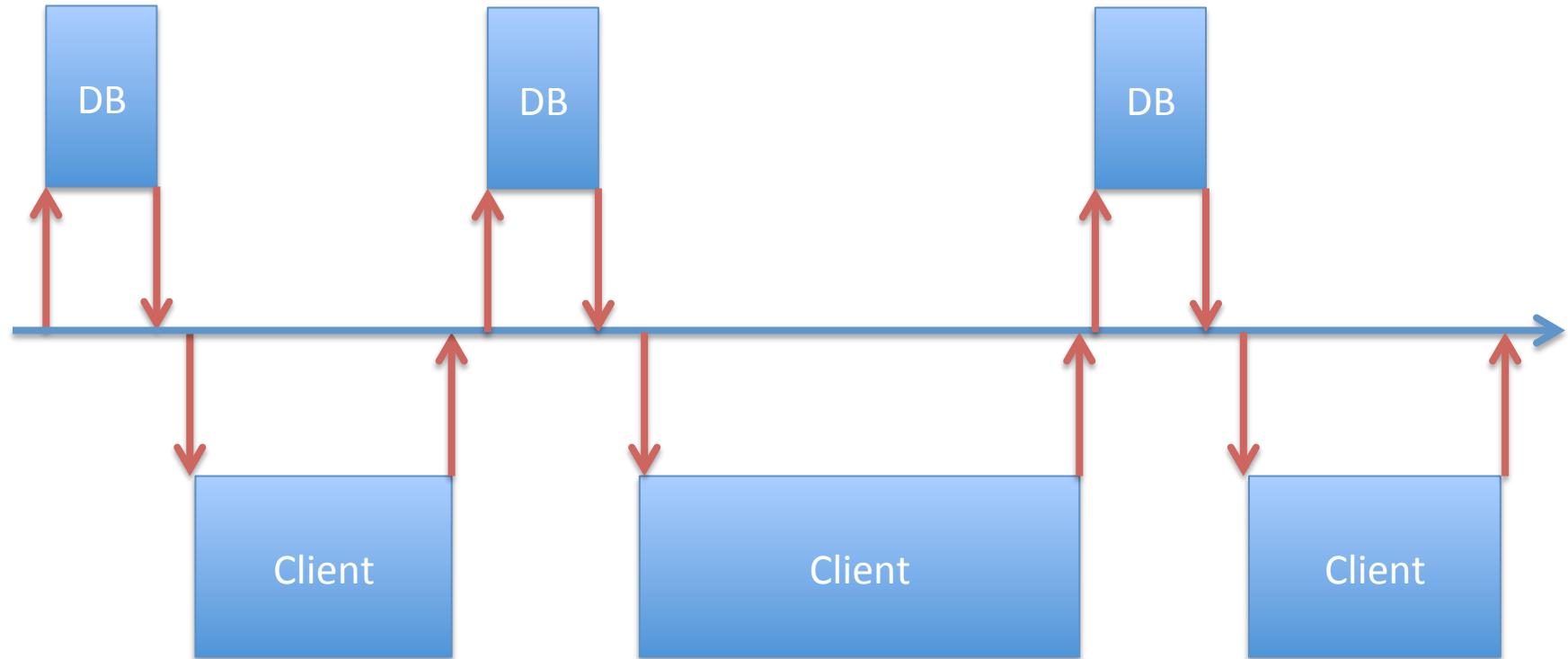
# Streaming Queries

- `val q = orders.filter(_.shipped).map(_.orderID)`
- `val a = q.result`
- `val f: Future[Seq[Int]] = db.run(a)`
- `db.stream(a).foreach(println)`

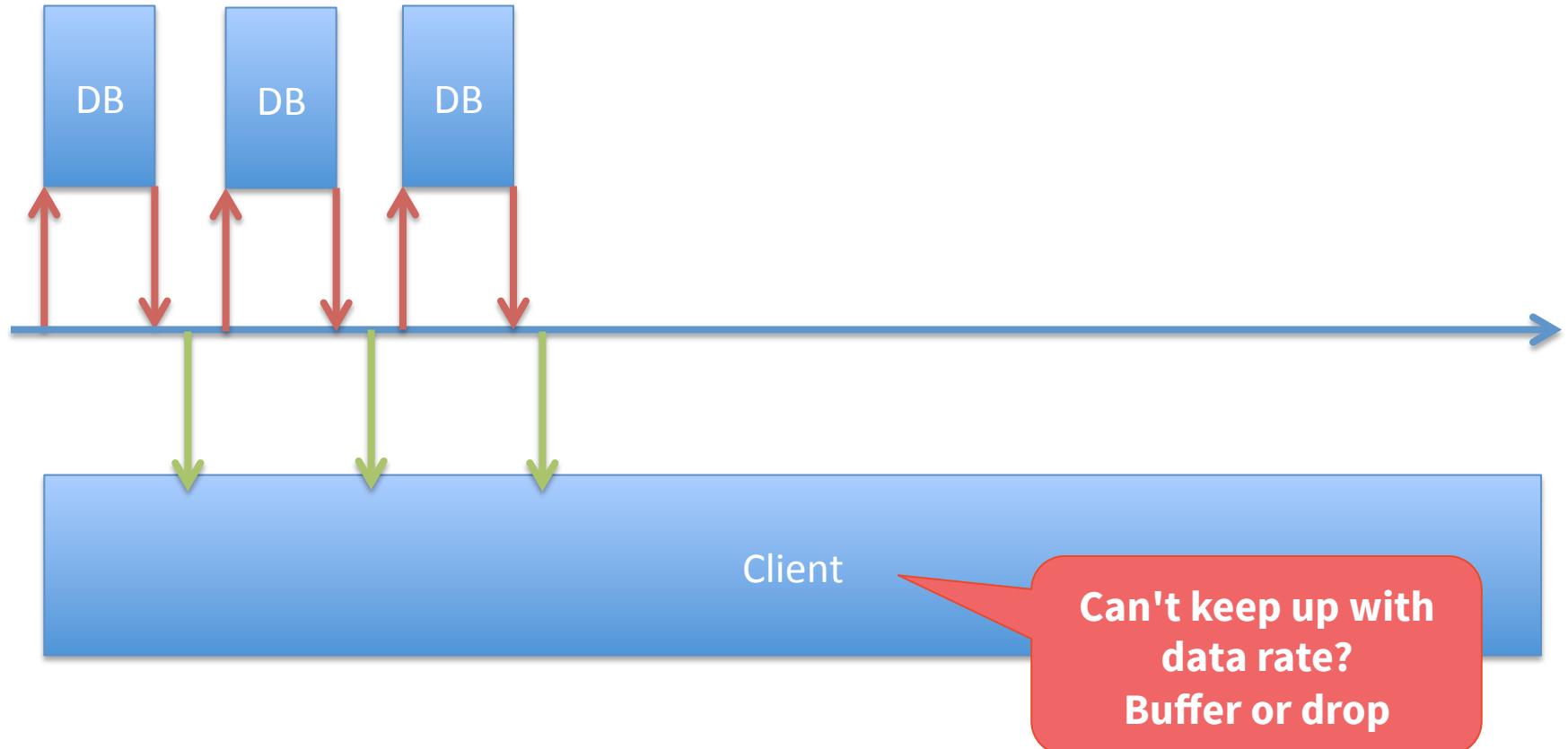
# Reactive Streams

- Reactive Streams API: <http://www.reactive-streams.org/>
- Slick implements Publisher for database results
- Use *Akka Streams* for transformations
- *Play 2.4* will support Reactive Streams
- Asynchronous streaming with back-pressure Handling

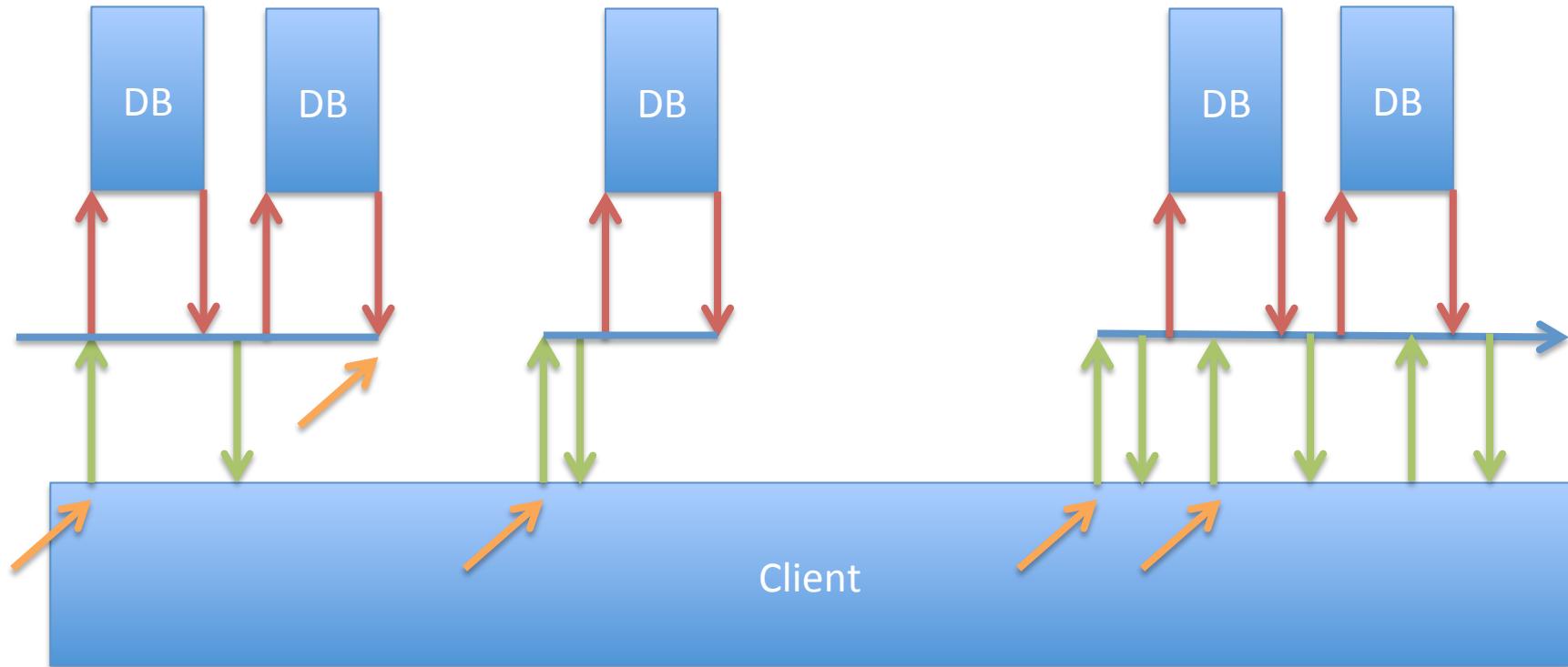
# Synchronous (Blocking) Back-Pressure



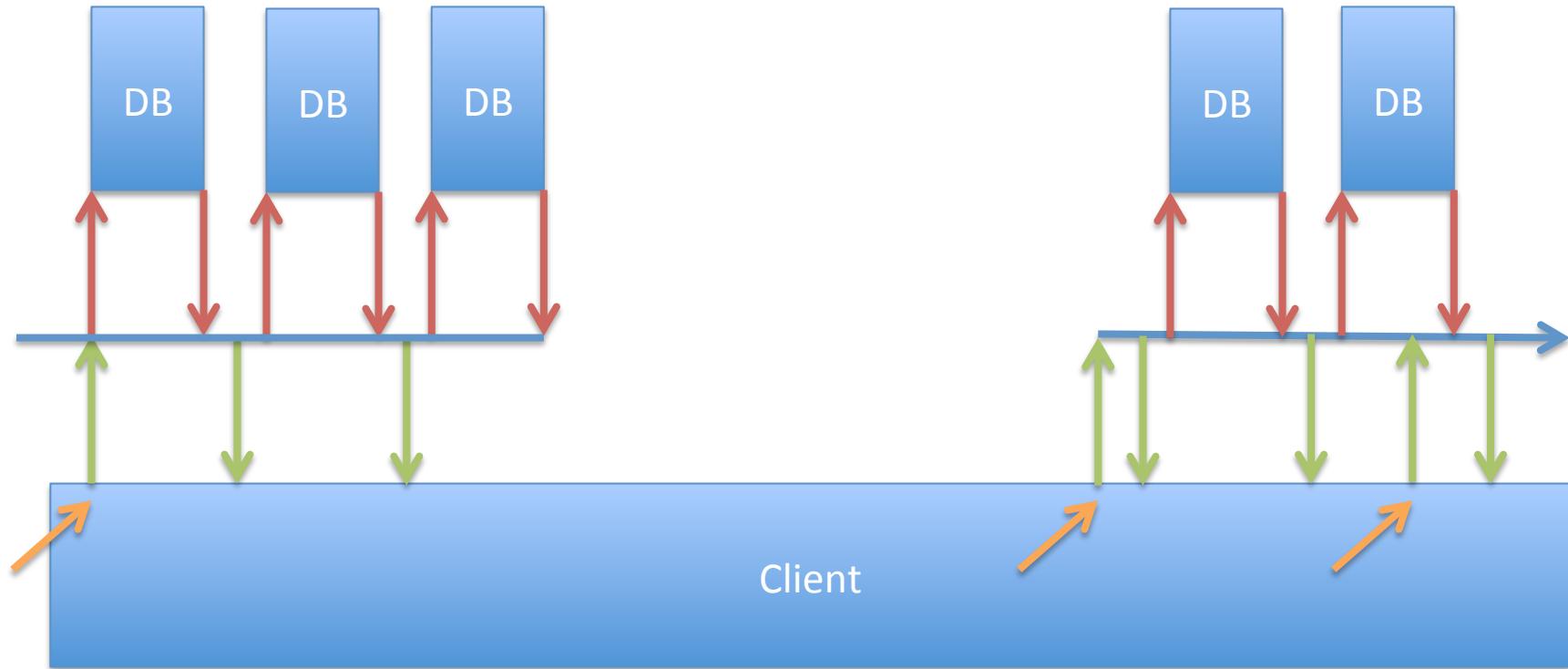
# Asynchronous Client: Naive Approach



# Asynchronous Client: Request 1



# Asynchronous Client: Request 2



# Asynchronous Database I/O

```
trait DatabaseDef {

 def run[R](a: DBIOAction[R, NoStream, Nothing])
 : Future[R]

 def stream[T](a: DBIOAction[_, Streaming[T], Nothing])
 : DatabasePublisher[T]
}
```

- Every *Streaming* action can be used as *NoStream*
- Collection-valued database results are *Streaming*
- The action runs when a *Subscriber* is attached

# Try it Yourself

# Hello Slick (Slick 3.0)

The screenshot shows the Typesafe Activator interface. The top navigation bar has tabs for 'Code' (selected), 'src/main/scala/HelloSlick.scala', 'Tutorial' (selected), and 'Intro to Slick'. On the left, there's a 'Browse Code' sidebar with project files like 'project', 'src', 'target', 'tutorial', 'activator.properties', 'build.sbt', 'LICENSE', 'README.md', and 'RUNNING\_PID'. The main area displays the Scala code for 'HelloSlick.scala':

```
1 import scala.concurrent.{Future, Await}
2 import scala.concurrent.ExecutionContext.Implicits.global
3 import scala.concurrent.duration.Duration
4 import slick.backend.DatabasePublisher
5 import slick.driver.H2Driver.api._

6
7 // The main application
8 object HelloSlick extends App {
9 val db = Database.forConfig("h2mem1")
10 try {
11
12 // The query interface for the Suppliers table
13 val suppliers: TableQuery[Suppliers] = TableQuery[Suppliers]
14
15 // the query interface for the Coffees table
16 val coffees: TableQuery[Coffees] = TableQuery[Coffees]
17
18 val setupAction: DBIO[Unit] = DBIO.seq(
19 // Create the schema by combining the DDLs for the Suppliers and Coffees
20 // tables using the query interfaces
21 (suppliers.schema ++ coffees.schema).create,
22
23 // Insert some suppliers
24 suppliers += (101, "Acme, Inc.", "99 Market Street", "Groundsville", "CA",
25 suppliers += (49, "Superior Coffee", "1 Party Place", "Mendocino", "CA",
26 suppliers += (150, "The High Ground", "100 Coffee Lane", "Meadows", "CA",
27)
28 }
```

To the right of the code editor, the 'Tutorial' section titled 'Intro to Slick' contains the following text:

Slick is a Functional Relational Mapping (FRM) library for Scala where you work with relational data in a type-safe and functional way. Here is an example:

```
coffees.filter(_.price < 10.0).map(_.name)
```

This will produce a query equivalent to the following SQL:

```
select COF_NAME from COFFEES where PRICE < 10.0
```

Developers benefit from the type-safety and compositability of FRM as well as being able to reuse the typical Scala collection APIs like `filter`, `map`, `groupBy`, etc. This template will get you started learning Slick using a working application. Continue the tutorial to learn about how to run the application, run the tests, and explore the basics of Slick.

- Typesafe Activator: <https://typesafe.com/get-started>

# Slick 3.0

- DBIO Action API
  - Improved Configuration via *Typesafe Config*
  - Nested Options and Properly Typed Outer Joins
  - Type-Checked Plain SQL Queries
- 
- ~~RC2 Available Now!~~
  - RC1 Available Now!



[slick.typesafe.com](http://slick.typesafe.com)



@StefanZeiger

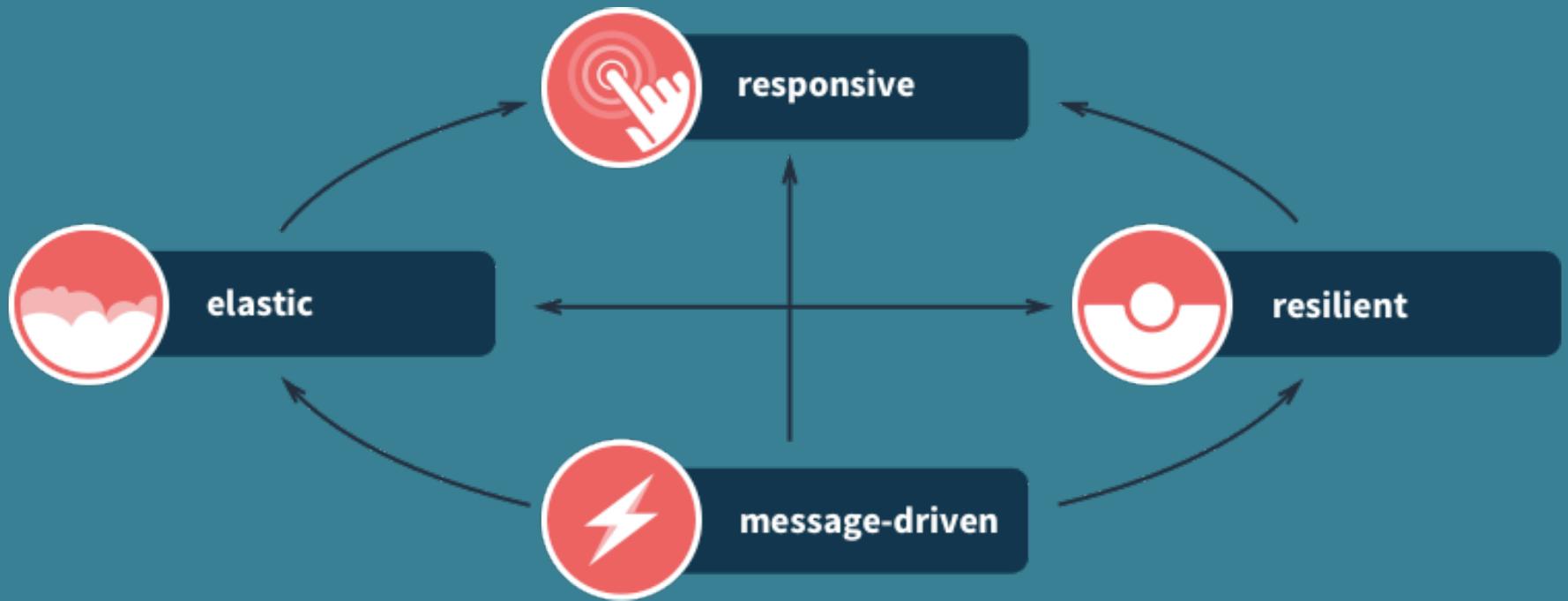


© Typesafe 2015 – All Rights Reserved

Clean title slides.

Break titles evenly when the  
title wraps onto two lines.

# Reactive Traits



*“A quote is highlighted in a Typesafe Red box for maximum impact and effect,  
like titles try to force the quote to word wrap evenly across lines.  
Space around the quote is a good thing”*

Name, Title, Company

# Title, Bullets & Quote

- Bullet Section Heading
  - Bullet one
  - Bullet two
  - Bullet three

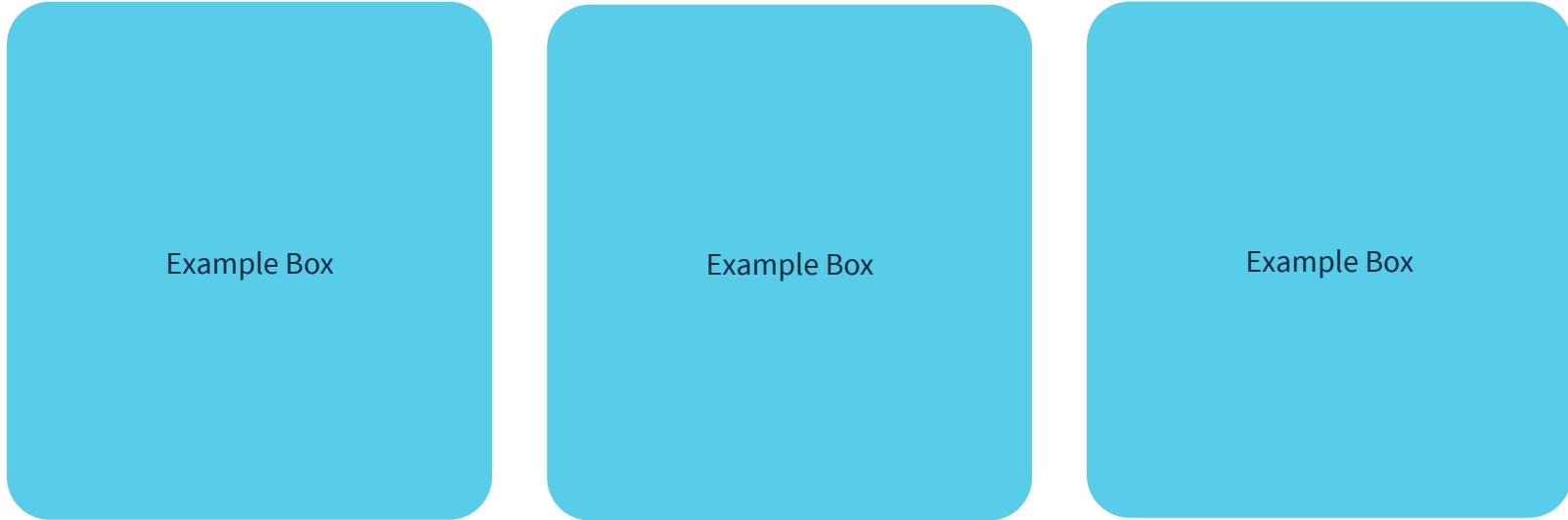
*“A quote is highlighted in a Typesafe Red box for maximum impact and effect.*

*Like titles, try to force the quote to word wrap evenly across lines.*

*Space around the quote is a good thing”*

Name, Title, Company

# Use Fade on elements with simple backgrounds



Example Box

Example Box

Example Box

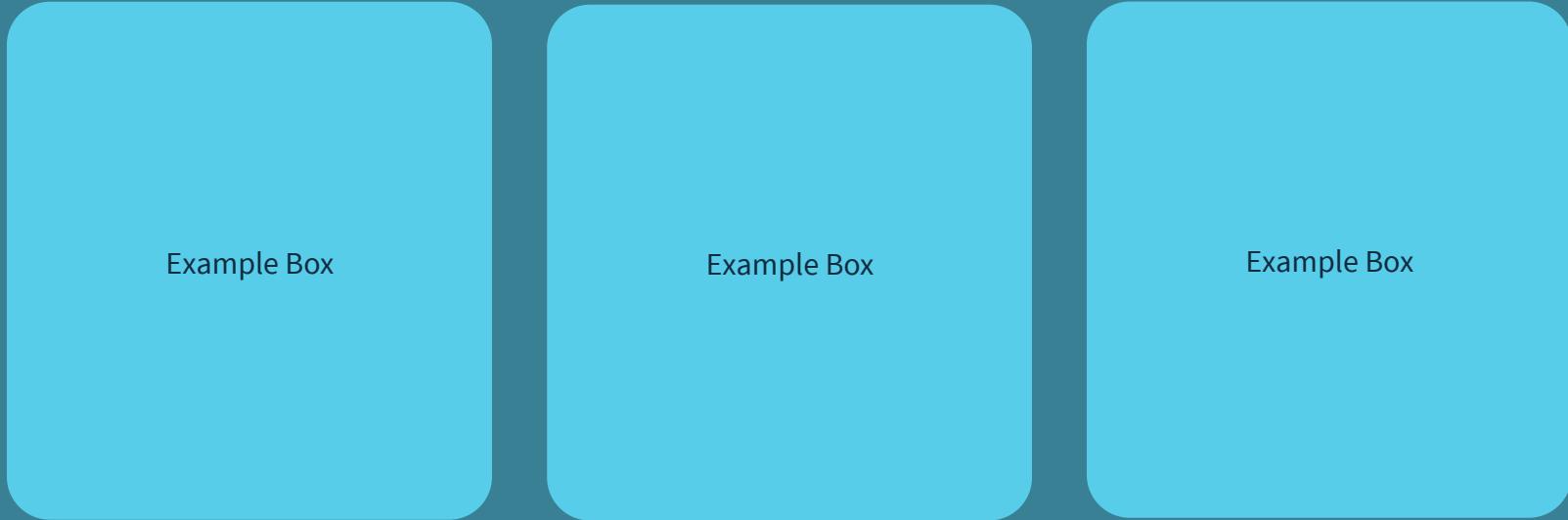
# Use Fade on elements with simple backgrounds

Example Box

Example Box

Example Box

# Use Fade on elements with simple backgrounds



Example Box

Example Box

Example Box

# Bullets build as groups – use ‘Fade’ effect

- JVM Based Developer Tools and Runtime
- Play Framework for Web Applications
  - Ideal for Responsive Web Apps
  - Rest based Services and Web Socket Apps
  - Supports Java and Scala
- Akka Runtime
  - Highly Scalable Runtime for Java and Scala Applications
  - Implementation of the Actor Model
- Scala Programming Language
  - Scalable and Performant
- Activator
  - Integrated Console for Application Profiling
  - Ensures Adopters are Successful from the Start

# Bullets build as groups – use ‘Fade’ effect

- JVM Based Developer Tools and Runtime
- Play Framework for Web Applications
  - Ideal for Responsive Web Apps
  - Rest based Services and Web Socket Apps
  - Supports Java and Scala
- Akka Runtime
  - Highly Scalable Runtime for Java and Scala Applications
  - Implementation of the Actor Model
- Scala Programming Language
  - Scalable and Performant
- Activator
  - Integrated Console for Application Profiling
  - Ensures Adopters are Successful from the Start

# All items build

- Developer and Production Support
  - Proactive tips and techniques
  - Older version maintenance
  - Security Vulnerability alerts
- Backstage Pass
  - Ask the Expert Webinars
  - Early access to online courses
  - Other customer only content
- Community Spotlight
  - Posting of job openings on community page
  - Projects highlighted on Typesafe content sites
  - Speaking opportunities at meet ups and conferences



*"A quote is highlighted in a Typesafe Red box for maximum impact and effect.*

*Like titles, try to force the quote to word wrap evenly across lines.*

*Space around the quote is a good thing"*

Name, Title, Company

# All items build

- Developer and Production Support
  - Proactive tips and techniques
  - Older version maintenance
  - Security Vulnerability alerts
- Backstage Pass
  - Ask the Expert Webinars
  - Early access to online courses
  - Other customer only content
- Community Spotlight
  - Posting of job openings on community page
  - Projects highlighted on Typesafe content sites
  - Speaking opportunities at meet ups and conferences



*"A quote is highlighted in a Typesafe Red box for maximum impact and effect.*

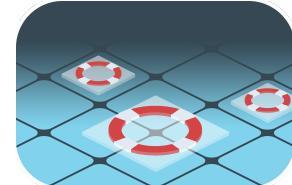
*Like titles, try to force the quote to word wrap evenly across lines.*

*Space around the quote is a good thing"*

Name, Title, Company

# Quote up – Bullets build

- Developer and Production Support
  - Proactive tips and techniques
  - Older version maintenance
  - Security Vulnerability alerts
- Backstage Pass
  - Ask the Expert Webinars
  - Early access to online courses
  - Other customer only content
- Community Spotlight
  - Posting of job openings on community page
  - Projects highlighted on Typesafe content sites
  - Speaking opportunities at meet ups and conferences



*"A quote is highlighted in a Typesafe Red box for maximum impact and effect.*

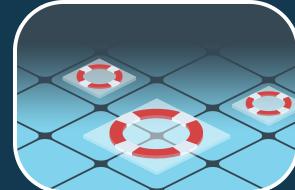
*Like titles, try to force the quote to word wrap evenly across lines.*

*Space around the quote is a good thing"*

Name, Title, Company

# Quote up – Bullets build

- Developer and Production Support
  - Proactive tips and techniques
  - Older version maintenance
  - Security Vulnerability alerts
- Backstage Pass
  - Ask the Expert Webinars
  - Early access to online courses
  - Other customer only content
- Community Spotlight
  - Posting of job openings on community page
  - Projects highlighted on Typesafe content sites
  - Speaking opportunities at meet ups and conferences



*"A quote is highlighted in a Typesafe Red box for maximum impact and effect.*

*Like titles, try to force the quote to word wrap evenly across lines.*

*Space around the quote is a good thing"*

Name, Title, Company