LIGHTBEND ACTIVATOR / **AKKA HTTP MICROSERVICE EXAMPLE**

Akka HTTP microservice example

Iterators (https://iterato.rs) Source (https://github.com/theiterators/akka-http-microservice#master) April 22, 2016 akka-http (/activator/templates#filter:akka-http) akka (/activator/templates#filter:akka) reactive-platform (/activator/templates#filter:reactive-platform) reactive (/activator/templates#filter:reactive) streams (/activator/templates#filter:streams) scala (/activator/templates#filter:scala) starter (/activator/templates#filter:starter) microservice (/activator/templates#filter:microservice)

Simple (micro)service which demonstrates how to accomplish tasks typical for REST service using Akka HTTP. Project includes: starting standalone HTTP server, handling simple file-based configuration, logging, routing, deconstructing requests, unmarshalling JSON entities to Scala's case classes, marshaling Scala's case classes to JSON responses, error handling, issuing requests to external services, testing with mocking of external services.

How to get "Akka HTTP microservice example" on your computer

There are several ways to get this template.

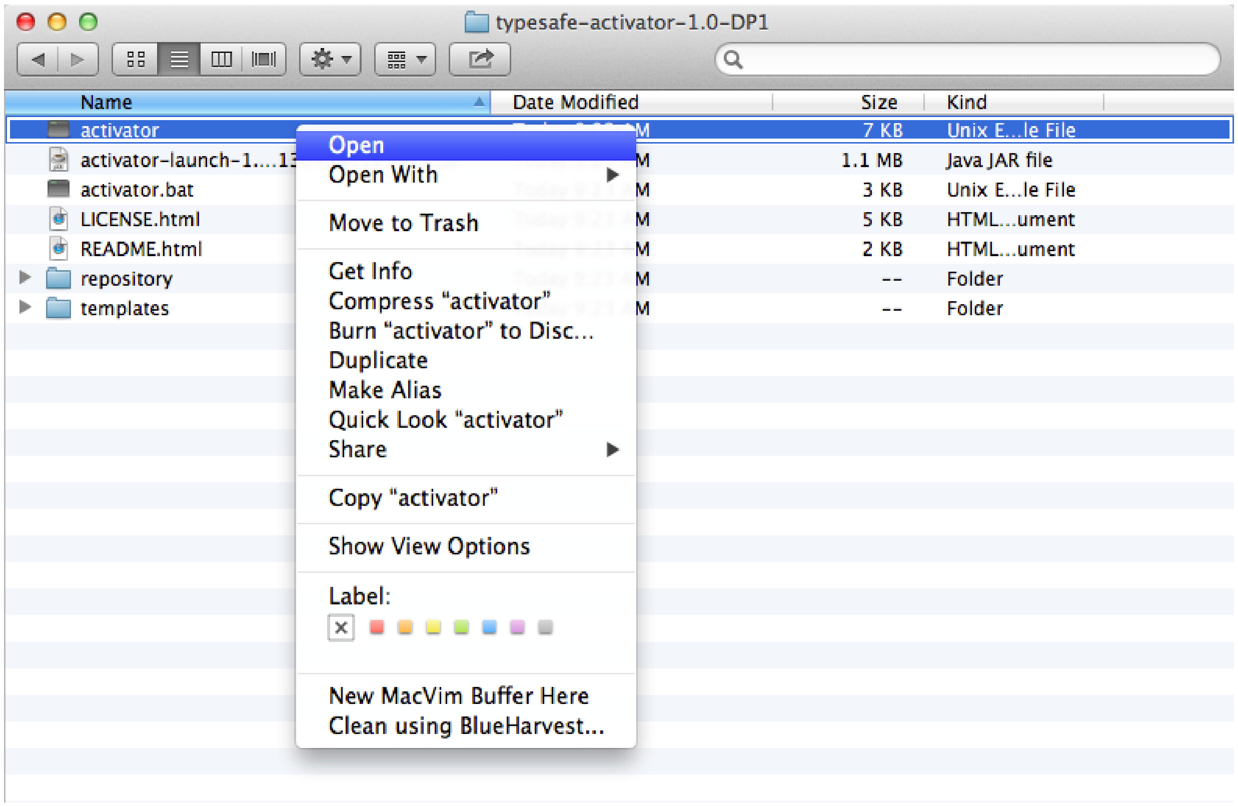
Option 1: Choose `akka-http-microservice` in the Lightbend Activator UI.

Already have Lightbend Activator (get it here (/activator/download))? Launch the UI then search for **`akka-http-microservice`** in the list of templates.

Option 2: Download the `akka-http-microservice` project as a zip archive

If you haven't installed Activator, you can get the code by downloading the template bundle for **`akka-http-microservice`**.

- Download the Template Bundle for "Akka HTTP microservice example" (/activator/template/bundle/akka-http-microservice)
- Extract the downloaded zip file to your system
- The bundle includes a small bootstrap script that can start Activator. To start Lightbend Activator's UI:
  In Finder, navigate to the directory that the template was extracted to, right-click on the file named "activator", then select "Open", and if prompted with a warning, click to continue:



Or from a command line:

```
typesafe:~/akka-http-microservice$ ./activator ui
```

This will start Lightbend Activator and open this template in your browser.

Option 3: Create a `akka-http-microservice` project from the command line

If you have Lightbend Activator, use its command line mode to create a new project from this template. Type **`activator new PROJECTNAME akka-http-microservice`** on the command line.

Option 4: View the template source

The creator of this template maintains it at https://github.com/theiterators/akka-http-microservice#master (https://github.com/theiterators/akka-http-microservice#master).

Option 5: Preview the tutorial below

We've included the text of this template's tutorial below, but it may work better if you view it inside Activator on your computer. Activator tutorials are often designed to be interactive.

Preview the tutorial

Akka HTTP microservice

This template lets you learn about:

- starting standalone HTTP server,
- handling simple, file-based configuration,
- logging,
- routing,
- deconstructing requests,
- unmarshalling JSON entities to Scala's case classes,
- marshaling Scala's case classes to JSON responses,

- error handling,
- issuing requests to external services,
- testing with mocking of external services.

It focuses on the HTTP part of the microservices and doesn't talk about database connection handling, etc.

Check out the code and don't forget to comment or ask questions on Github (https://github.com/theiterators/akka-http-microservice) and Twitter (https://twitter.com/luksow).

On next pages you will find a brief tutorial about how the service works. You can:

- learn Activator what a microservice is,
- check Activator what our microservice does,
- or you can go Activator straight to the code.

What is a microservice?

Microservice is a tiny standalone program that can be used as a component of bigger distributed system. Microservices:

- are short and concise,
- process only one bounded domain.

In order to be readable and rewritable, code in microservices is usually very short and brief. It's usually responsible for processing only one type of data (in this project it is an IP location data). They rarely use high level of abstraction over databases, networking, and other components. It all makes them easier to understand and easier to reuse in multiple projects.

Next: Activator What does the example microservice do?

Geolocation of IP addresses

Our example microservice has two main features. It should:

- locate an IP address,
- and compute distances between locations of two IP addresses.

It should do all that by exposing two HTTP JSON endpoints:

- `GET /ip/X.X.X.X` — which returns given IP's geolocation data,
- `POST /ip` — which returns distance between two IPs geolocations given JSON request `{"ip1": "X.X.X.X", "ip2": "Y.Y.Y.Y"}`.

Next: Let's see how to Activator run it!

Running the template

Check Activator Run to see if the microservice has successfully compiled and is already running.

You can check out where are Google DNS servers by opening `http://localhost:9000/ip/8.8.8.8` (http://localhost:9000/ip/8.8.8.8). As you can see in the URL, the browser will send GET request to the first endpoint.

You can also check our endpoints using `curl` command line tool:

```
$ curl http://localhost:9000/ip/8.8.8.8
```

and

```
$ curl -X POST -H 'Content-Type: application/json' http://localhost:9000/ip -d '{"ip1": "8.8.8.8", "ip2":
"8.8.4.4"}'
```

for the second endpoint.

If you don't have curl installed you can install it from the source (http://curl.haxx.se/docs/install.html), using your OS package manager or you can use Postman REST Client in your browser.

Next: Let's see Activator how our responses look like.

The Geolocation IP responses

Responses should look like that:

```
{
  "city": "Mountain View",
  "query": "8.8.8.8",
  "country": "United States",
  "lon": -122.0881,
  "lat": 37.3845
}
```

for the first endpoint and

```
{
  "distance": 4347.6243474947,
  "ip1Info": {
    "city": "Mountain View",
    "query": "8.8.8.8",
    "country": "United States",
    "lon": -122.0881,
    "lat": 37.3845
  },
  "ip2Info": {
    "city": "Norwell",
    "query": "93.184.216.34",
    "country": "United States",
    "lon": -70.8228,
    "lat": 42.1508
  }
}
```

for the second endpoint. Notice that they are pretty-printed. In the Activator Run tab you can see the request/response logs the app generates.

Next: Now as we know what our microservice does Activator let's open up the code.

Code overview

There are four significant parts of the code. These are:

- build.sbt and plugins.sbt — the build scripts,
- application.conf — the seed configuration for our microservice,
- AkkaHttpMicroservice.scala — our main Scala file.
- ServiceSpec.scala — an example `akka-http` server test.

The build scripts let SBT download all the dependencies for our project (including `akka-http`). They are described inside Activator build scripts part of the tutorial.

Configuration for our microservice is described in the Activator configuration part of the tutorial.

The code implementing our microservice's logic is described in the Activator "microservice's code" section.

Build scripts

build.sbt and plugins.sbt hold the configuration for our build procedure.

build.sbt

`build.sbt` provides our project with typical meta-data like project names and versions, declares Scala compiler flags and lists the dependencies.

- `akka-actor` is the corner stone of Actor system that `akka-http` and `akka-stream` are based on.
- `akka-stream-experimental` is the library implementing Reactive Streams using Akka actors — a framework for building reactive applications.
- `akka-http-core-experimental` is core library for creating reactive HTTP streams.
- `akka-http-experiental` is a library for processing HTTP requests using akka-streams and actors.
- `akka-http-spray-json-experimental` is a library for marshalling Scala data types into JSON.
- `akka-http-testkit-experimental` is a library that helps testing `akka-http` routing and responses.
- `org.scalatest` is a standard Scala testing library.

Notice that some of the dependencies are still in `experimental` phase — this means that their API can be subject to major changes.

plugins.sbt

There are three plugins used in our project. These are:

- `sbt-revolver` which is helpful for development. It recompiles and runs our microservice every time the code in files changes. Notice that it is initialized inside `build.sbt`.
- `sbt-assembly` is a great library that lets us deploy our microservice as a single .jar file.
- `sbt-native-packager` is needed by Heroku to stage the app.

Next: As we know what are the dependencies of our project, let's see Activator what is the minimal configuration needed for the project.

Configuration

The seed configuration for our microservice is available in the application.conf. It consists of three things:

- `akka` — Akka configuration,
- `http` — HTTP server configuration,
- `services` — external endpoints configuration.

The Akka part of the configuration will let us see more log messages on the console when developing the microservice.

HTTP interface needs to be given a interface that it will run on and port that will listen for new HTTP requests.

Our microservice uses external service `https://freegeoip.net` to find where the IP we're trying to find is.

When deploying microservice as a `.jar` file, one can overwrite the configuration values when running the jar.

```
java -jar microservice.jar -Dservices.freeGeoIpPort=8080
```

Using configuration management system is also recommended as amount of variables rises quickly. It is hard to maintain configuration files across more complex microservice architecture.

Next: Let's see how is our configuration used in Activator the code.

Microservice's code

All of the code is held in AkkaHttpMicroservice.scala. We can distinguish 6 parts of the code. These are:

- the imports,
- type declarations and business domain,
- protocols,
- networking logic,
- routes,
- main App declaration.

The names, order and configuration are not standardized, but the list above will make it easier for us to reason about this code.

We won't get into much details about imports. The only thing worth remembering is that there are many `implicit values` imported and one should be cautious when removing the imports, as many of them can be marked as unused by one's IDE.

This section of tutorial explains:

- Activator How to use Scala types in HTTP microservice?
- Activator How to do external HTTP requests?
- Activator How to declare HTTP routes?
- Activator What our tests do?

Scala types and protocols

To see the usage of Scala types and protocols inside our microservice open up the AkkaHttpMicroservice.scala. We have two type of types there:

- `IpPairSummaryRequest` — a case class that models our JSON HTTP request's body.
- `IpInfo` and `IpInfoSummary` — case classes used as an intermediate form of data that can be converted to response JSON.

Modeling requests

`akka-http` can unmarshall any JSON request into a type. This way we can validate incoming requests and pass only the ones that are well formed and complete. The easiest way to model requests is to create algebraic data types and instrument them with validations in a constructor (typical methods include using Scala's Predef library with its `require` method). Example:

```
  case class IpPairSummaryRequest(...) {
    ...
    require(ip1..split('.').map(_.toInt).map({s => s >= 0 && s <= 255}).fold(true)(_ && _), "wrong IP
address")
    ...
}
```

Forming JSON response

One of the great features of `akka-http` is response marshalling. The responses will be implicitly converted into JSON whether they are `Option[T]`, `Future[T]`, etc. Proper errors and response codes will also be generated.

Using this feature requires:

- `import akka.http.scaladsl.marshallers.sprayjson.SprayJsonSupport._`,
- declaring implicit JSON converters (here it's done inside `Protocols` trait).

Next: Activator Making external HTTP requests.

Making external HTTP request

Handling communication with external HTTP services is done inside `Service` trait.

Making a HTTP request

Making a proper HTTP request using `akka-http` leverages the Reactive Streams approach. It requires:

- defining an external service HTTP connection flow,
- defining a proper HTTP request,
- defining this request as a source,
- connecting the request source through external service HTTP connection flow with so called `Sink`.

In order for the flow to run, we also need `FlowMaterializer` and `ExecutionContext`. After the request is done, we get standard `HttpResponse` that we need to handle.

Handling the response

Handling `HttpResponse` consists of:

- checking if request was successful,
- unmarshalling HTTP Entity into a case class.

The unmarshalling uses the protocol implicit values defined Activator earlier. Unmarshalling works using `Futures[T]` so we can always handle any errors and exceptions raised by our validation logic.

Next: Activator Declaring routes and responding to HTTP requests.

Routing and running server

Routing directives can be found in the `Service` trait.

`akka-http` provides lots of useful routing directives. One can use multiple directives by nesting them inside one another. The request will go deeper down the nested structure if only it complies with each of the directive's requirements. Some directives filter the requests while others help deconstructing it. If the request passes all directives, the final `complete(...) {...}` block gets evaluated as a `HttpResponse`.

Routing & filtering directives

Directives responsible for routing are:

- `pathPrefix("ip")` — filters the request by it's relative URI beginning,
- `path("ip"/"my")` — filters the request by it's part of the URI relative to the hostname or `pathPrefix` directive in which it is nested,
- `get` — filters GET requests,
- `post` — filters POST requests,
- and many more. (http://doc.akka.io/api/akka-stream-and-http-experimental/current/#akka.http.scaladsl.server.Directives)

Deconstructing request

Directives that let us deconstruct the request:

- `entity(as[IpPairSummaryRequest])` — unmarshalls HTTP entity into an object; useful for handling JSON requests,
- `formFields('field1, 'field2)` — extracts form fields form POST request,
- `headerValueByName("X-Auth-Token")` — extracts a header value by its name,
- `path("member" / Segment / "books")` — the `Segment` part of the directive lets us extract a string from the URI,
- and many more. (http://doc.akka.io/api/akka-stream-and-http-experimental/current/#akka.http.scaladsl.server.Directives)

Directives can provide us with some values we can use later to prepare a response:

```
headerValueByName("X-Requester-Name") { requesterName =>
  Ok("Hi " + requesterName)
}
```

There are other directives like `logRequestResult` that don't change the flow of the request. We can also create our own directives whenever needed.

Building a response

If we use Activator JSON marshalling, it is very easy to build a JSON response. All we need to do is to return marshallable type in `complete` directive (ex. `String`, `Future[T]`, `Option[T]`, `StatusCode`, etc.). Most of the HTTP status codes are already implemented in `akka-http`. Some of them are:

- `Ok` — 200 response
- `NotFound` — 404 response which is automatically generated when `None` is returned.
- `Unauthorized` — 401 response
- `Bad Request` — 400 response

Next: Activator Testing `akka-http`.

Tests

Check out simple tests that we prepared and don't forget to Activator run them on your computer!

The interesting parts of the tests are:

- the syntax of route checking,
- the way external requests are mocked.

Next: Activator Tutorial summary.

Summary

And that's it! We hope you enjoyed this tutorial and learned how to write a small microservice that uses `akka-http`, responds to `GET` and `POST` requests with JSON and connects with external services through HTTP endpoint.

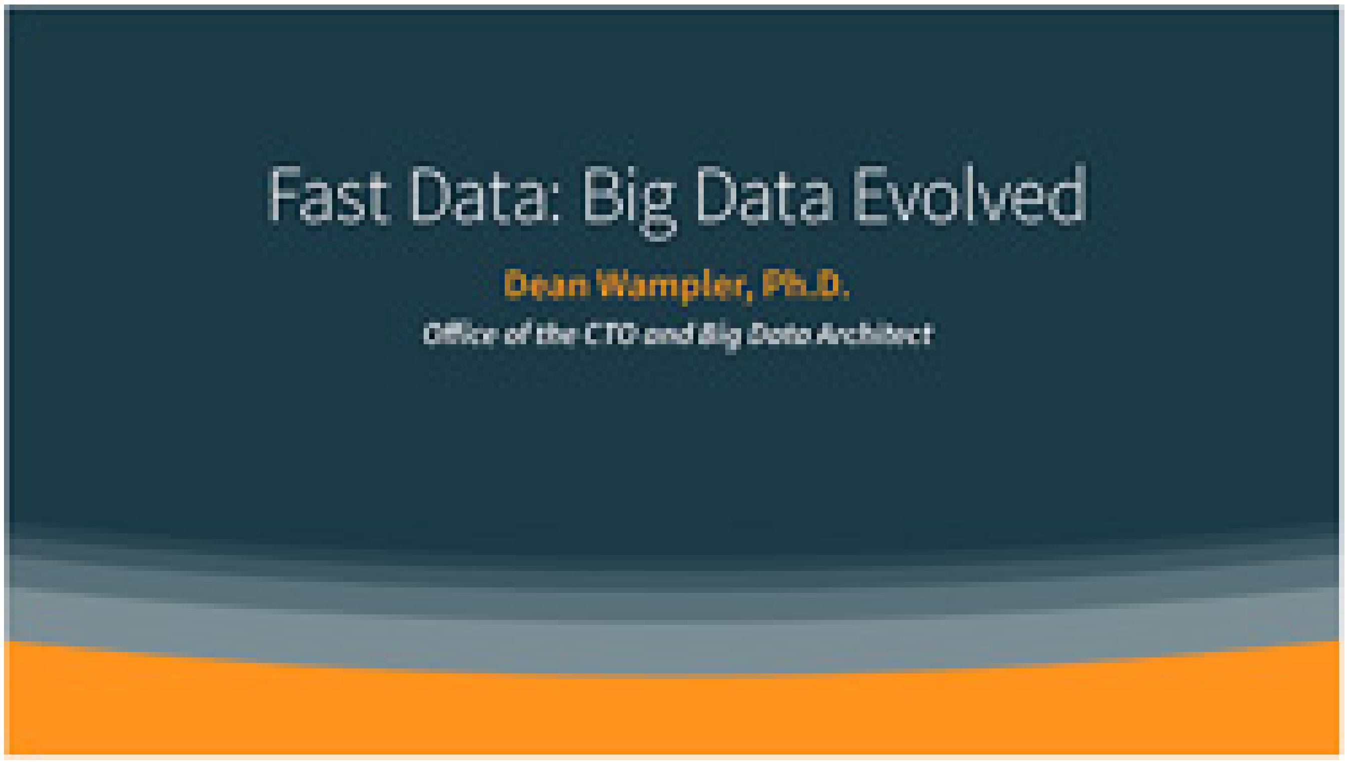
Be sure to ping us on Github (https://github.com/theiterators/akka-http-microservice) or Twitter (https://twitter.com/luksow) if you liked it or if you have any questions.

Big Data Africa · 10 months ago
Hello, guys. we currently shifting to developing with this template. How do I handle url query parameters with it?
Reply · Share ›

Łukasz Sowa ➜ Big Data Africa · 10 months ago
Answered here: https://github.com/theiterator...
Reply · Share ›

WHITE PAPER

Fast Data: Big Data Evolved



Fast Data: Big Data Evolved

Dean Wampler, Ph.D.

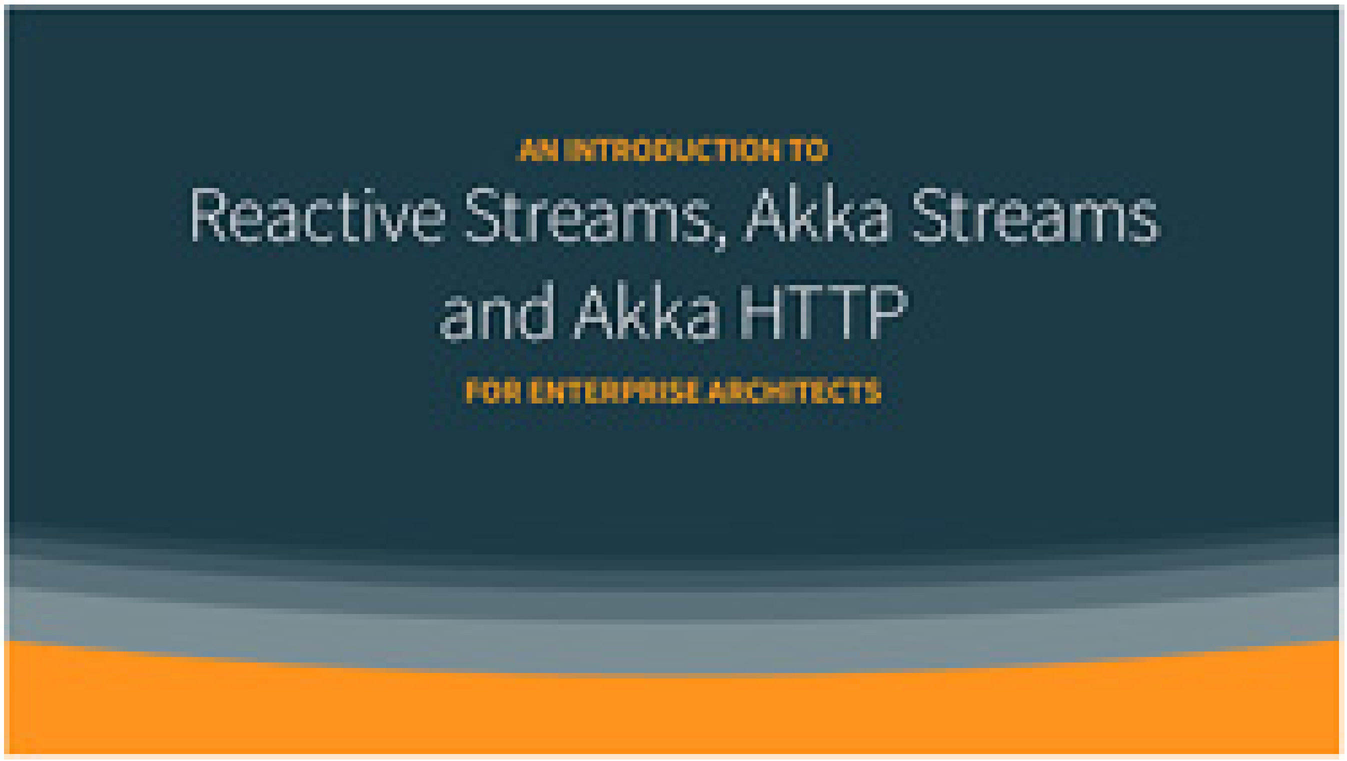
Office of the CTO and Big Data Architect

(https://info.lightbend.com/COLL-20XX-Fast-Data-Big-Data-Evolved-WP_LP.html?lst=WS&lsd=COLL-20XX-Fast-Data-Big-Data-Evolved-WP)

GET WHITE PAPER (HTTPS://INFO.LIGHTBEND.COM/COLL-20XX-FAST-DATA-BIG-DATA-EVOLVED-WP_LP.HTML?LST=WS&LSD=COLL-20XX-FAST-DATA-BIG-DATA-EVOLVED-WP)

WHITE PAPER

An Introduction to Reactive Streams, Akka Streams and Akka HTTP


AN INTRODUCTION TO
Reactive Streams, Akka Streams and Akka HTTP
FOR ENTERPRISE ARCHITECTS

(https://info.lightbend.com/COLL-20XX-Enterprise-Architect-Akka-Streaming-Guide_LP.html?lst=WS&lsd=COLL-20XX-Enterprise-Architect-Akka-Streaming-Guide)

GET WHITE PAPER (HTTPS://INFO.LIGHTBEND.COM/COLL-20XX-ENTERPRISE-ARCHITECT-AKKA-STREAMING-GUIDE_LP.HTML?LST=WS&LSD=COLL-20XX-ENTERPRISE-ARCHITECT-AKKA-STREAMING-GUIDE)

Lightbend Newsletter

Subscribe to our newsletter for regular updates from Lightbend.

SUBSCRIBE TO OUR NEWSLETTER (/NEWSLETTER)

Have a Question?

Contact us (/contact) to learn more about our training, support and subscription offerings. Are you a subscriber needing support (https://together.typesafe.com/account/support/login)?

CONTACT US (/CONTACT)

From the blog

How Reactive systems help PayPal's squbs scale to billions of transactions daily (/blog/how-reactive-systems-help-paypal-squbs-scale-to-billions-of-transactions-daily)

Recently, at Scala Days New York 2016, I had a chance to sit down with Akara Sucharitakul (https://github.com/akara), Principal Member of Technical Staff at PayPal, and the main force behind PayPal's open source squbs project (https://github.com/paypal/squbs). Akara had recently published an in-depth article (https://www.paypal-engineering.com/2016/05/11/squbs-a-new-reactive-way-for-paypal-to-build-applications/) about squbs, describing it as the Reactive way to for PayPal to build, deploy and manage crucial services that have to process billions of transactions per day on as little infrastructure as possible.

CONTINUE READING... (/BLOG/HOW-REACTIVE-SYSTEMS-HELP-PAYPAL-SQUBS-SCALE-TO-BILLIONS-OF-TRANSACTIONS-DAILY)

Call for respondents: Cloud, Containers, Microservices and the Influence of Development Teams on Architecture (/blog/survey-cloud-containers-microservices-and-the-influence-of-development-teams-on-architecture)

Got 10 minutes to share your experiences in an anonymous survey? Each completion goes towards a **donation to charity**!

CONTINUE READING... (/BLOG/SURVEY-CLOUD-CONTAINERS-MICROSERVICES-AND-THE-INFLUENCE-OF-DEVELOPMENT-TEAMS-ON-ARCHITECTURE)

Newsroom

Lightbend To Modernize the Enterprise Application Stack for “Fast Data” Through Support for IBM Data Science Experience (/company/news/lightbend-to-modernize-the-enterprise-application-stack-for-fast-data-through-support-for-ibm-data-science-experience)

eWeek: Lightbend Helps Developers Build Fast Data, Microservices Apps (/company/news/lightbend-helps-developers-build-fast-data-microservices-apps)

Lightbend Closes $20 Million in Series C Funding Led by Intel Capital (/company/news/lightbend-closes-20-million-in-series-c-funding-led-by-intel-capital)

TechRepublic: How one e-commerce giant uses microservices and open source to scale like crazy (/company/news/techrepublic-how-one-e-commerce-giant-uses-microservices-and-open-source-to-scale-like-crazy)

First Impressions of Lagom and Initial Comparison to Spring Boot/Cloud (/company/news/first-impressions-of-lagom-and-initial-comparison-to-spring-cloud)

Lightbend Announces Acquisition of BoldRadius as Demand for Reactive Systems Grows (/company/news/lightbend-announces-acquisition-of-boldradius-as-demand-for-reactive-systems-grows)

PRODUCTS (/PRODUCTS/LIGHTBEND-REACTIVE-PLATFORM)

Lightbend Reactive Platform (/products/lightbend-reactive-platform)

- ConductR (/products/conductr)
- Monitoring (/products/monitoring)
- Lagom (/lagom)
- Apache Spark (/products/spark)

SERVICES (/SERVICES/EXPERT-SUPPORT)

Expert Support (/services/expert-support)
Training (/services/training)
Consulting (/services/consulting)

CUSTOMERS (/RESOURCES/CASE-STUDIES-AND-STORIES)

Case Studies & Stories (/resources/case-studies-and-stories)
Our Customers Are Hiring (/customers/our-customers-are-hiring)

COMPANY (/COMPANY)

About (/company)
Leadership (/company/leadership)
Careers (/company/careers)
News & Press Releases (/company/news)
Contact (/contact)
Open Source Position Statement (/open-source-position-statement)

BLOG (/BLOG)
Tech Blog (/blog)

RESOURCES (/RESOURCES)
Featured Content (/resources)
Activator Templates (/activator/templates)
E-Books (/resources/e-books)
Videos (/resources/videos)
Collateral (/resources/collateral)
Lightbend Podcast Channel (https://soundcloud.com/lightbend)
(http://feeds.soundcloud.com/users/soundcloud:users:100369928/sounds.rss)

PARTNERS (/PARTNERS/OVERVIEW)
Overview (/partners/overview)
Resellers (/partners/resellerpartners)
Technology (/partners/technology-partners)
System Integrators (/partners/systemintegrator-partners)
Global System Integrators (/partners/globalsystemintegrator-partners)
Training (/partners/training-partners)
Become a Partner (/partners/become-a-partner)

COMMUNITY (/COMMUNITY/PROJECTS)
Projects (/community/projects)
Activator Templates (/activator/templates)
Partners (/partners/overview)
Resources (/resources)
Webinars (/resources/videos#filter:webinar)
Events (/community)
Online Store (SWAG) (http://shop.typesafe.com)

ACTIVATOR (/ACTIVATOR/DOWNLOAD)
Download Activator (/activator/download)
Activator Templates (/activator/templates)

LIGHTBEND ACCOUNT (/ACCOUNT)
Login (/account/login)
Sign up (/account/register)

Follow

(http://www.facebook.com/typesafe)

(http://twitter.com/intent/follow?source=followbutton&variant=1.0&screen_name=lightbend)

(https://plus.google.com/+Typesafe/videos) (http://www.linkedin.com/company/typesafe)

(http://www.youtube.com/channel/UCmbAzITOKm4KAhV8FBECfMA?sub_confirmation=1)

(https://www.typesafe.com/blog/rss.xml)