Unified Data Applications with Shiny on Delta Lakes (**Shiny Lakehouse**)

Hossein Falaki

@mhfalaki

databricks

databricks

A unified data analytics platform for accelerating innovation across data engineering, data science, and analytics

- Global company with over 5,000 customers and 450+ partners
- Original creators of popular data and machine learning open source projects

Data Apps and Shiny

- Most enterprises (and vendors) focus on BI
  - Large addressable market
  - Easier to manage because of declarative SQL interface
- Data scientists (advanced users) prefer building dashboards using code
  - Data access is not restricted to SQL
  - Ability to use third-party packages
  - Non-trivial data manipulation
  - Ultimate control over UI layout
  - Easy transition from EDA to presentation
- **Shiny** is the most popular framework for data apps.

Data Apps and Big Data

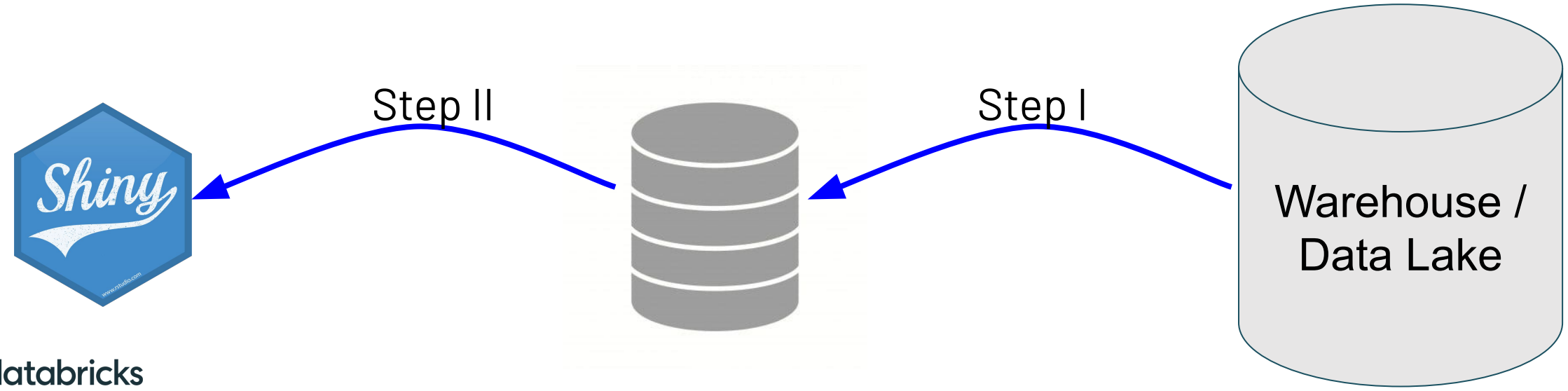
- Data grows fast (exponentially)
  - Very common quote: “our data grew beyond our assumptions”
- Data evolves
  - Data schema changes
  - New streams of data
  - New sources
  - New dimensions
  - New use cases
- Value/insight from data compounds as metadata grows
  - You want to join/merge additional datasets with your input

Two-stage architecture

Step I: Extract data from the Data Lake or Data Warehouse and stage it

- Filtering
- Aggregation (summarization)
- Sampling

Step II: Read staged data from disk or RDBMS into R (Shiny) and present



Pros and Cons

- Pros
  - Minimal change to existing Shiny application's data access patterns (Step I)
- Cons
  - Two data pipelines to maintain
  - Staged data gets stale
  - Data governance may have issues with the staging step

We have been working with many data teams to streamline and unify their data pipeline using the **lakehouse** pattern.

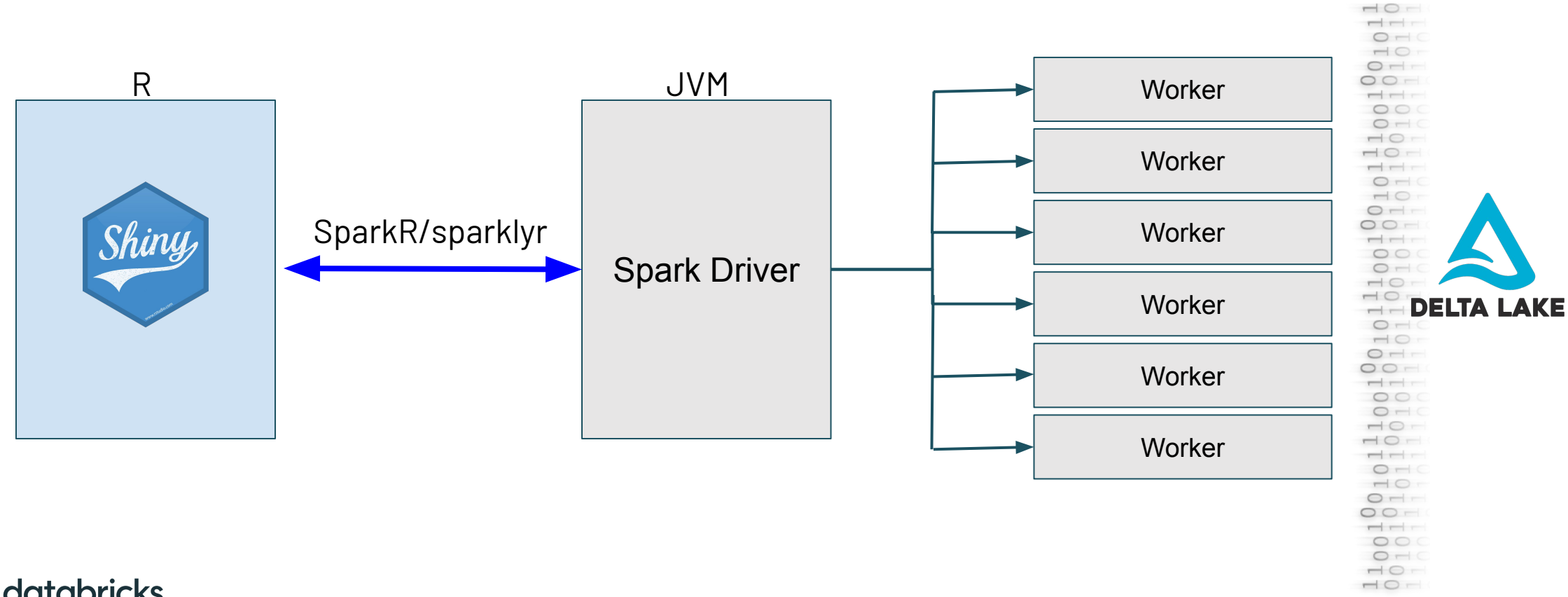
What is the Lakehouse

Data management features similar to *data warehouses*, directly on the low cost storage used for *data lakes*.

- ACID Transactional support
- Schema enforcement and governance
- Decoupled storage from compute
- Open storage formats
- Diverse data type support: e.g., images, audio, semi-structured data
- Enabling end-to-end streaming
- Support for diverse workloads

The Unified Architecture: Shiny Lakehouse

With Apache Spark and Delta you can directly operate on the Data Lake



SparkR/sparklyr

R packages to program Apache Spark.

- Provide R API front-end to (wrappers for) Apache Spark
- Expose Spark DataFrames (a.k.a. tables)
- Convenient interoperability between R data.frames and Spark DF

Robust distributed processing, vast data sources, in-memory computing, ...

+

Dynamic environment, interactivity, large package ecosystem, visualization, ...

Access big data directly from a Shiny app

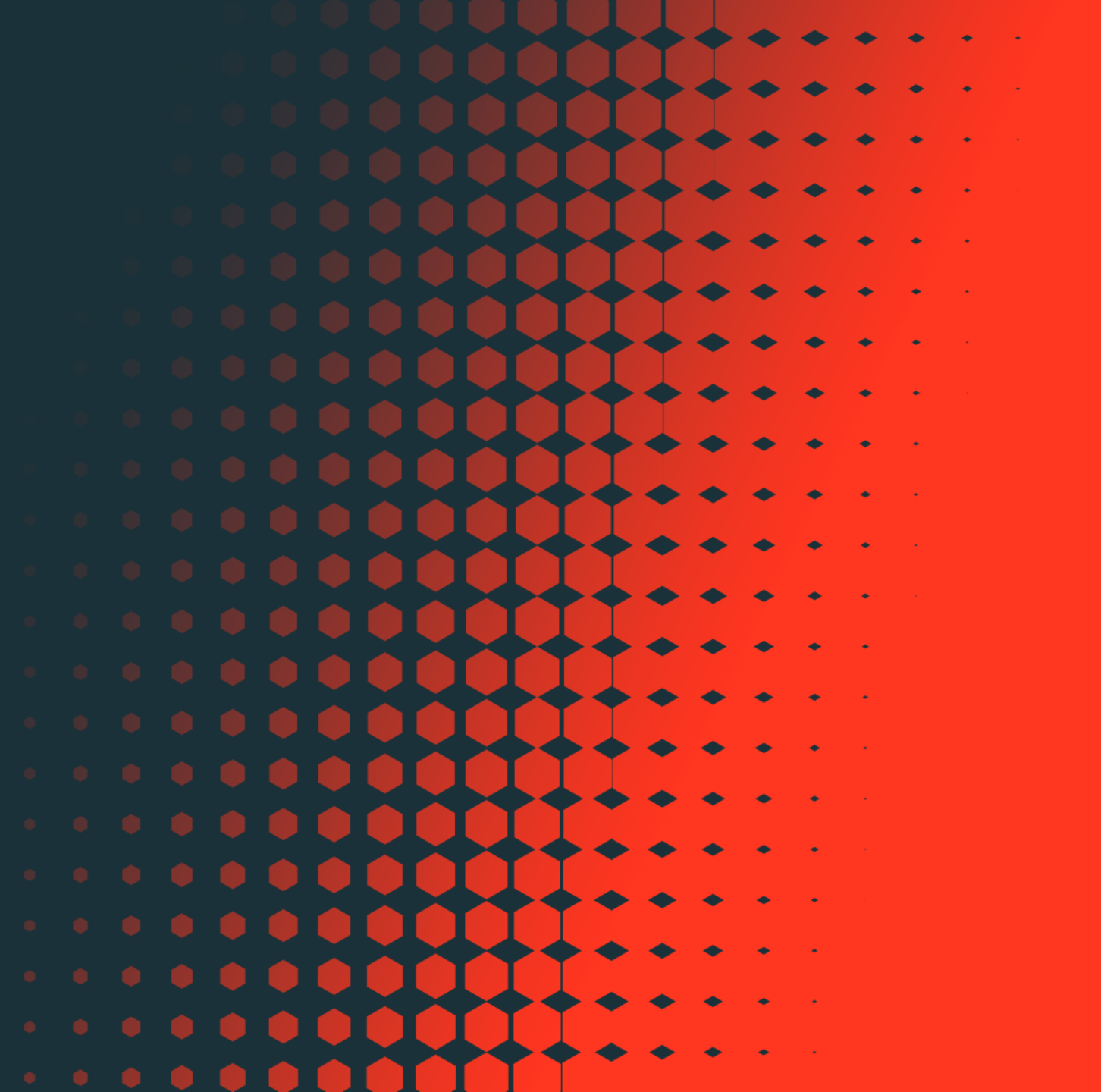
```
library('shiny')
library(SparkR)
sparkR.session()

uiFunc <- basicPage( ... )

serverFunc <- function(input, output) {
  df <- read.df(source = "delta", path = input$path)
  output$records <- nrow(df)
}

shinyApp(ui = uiFunc, server = serverFunc)
```

Demo



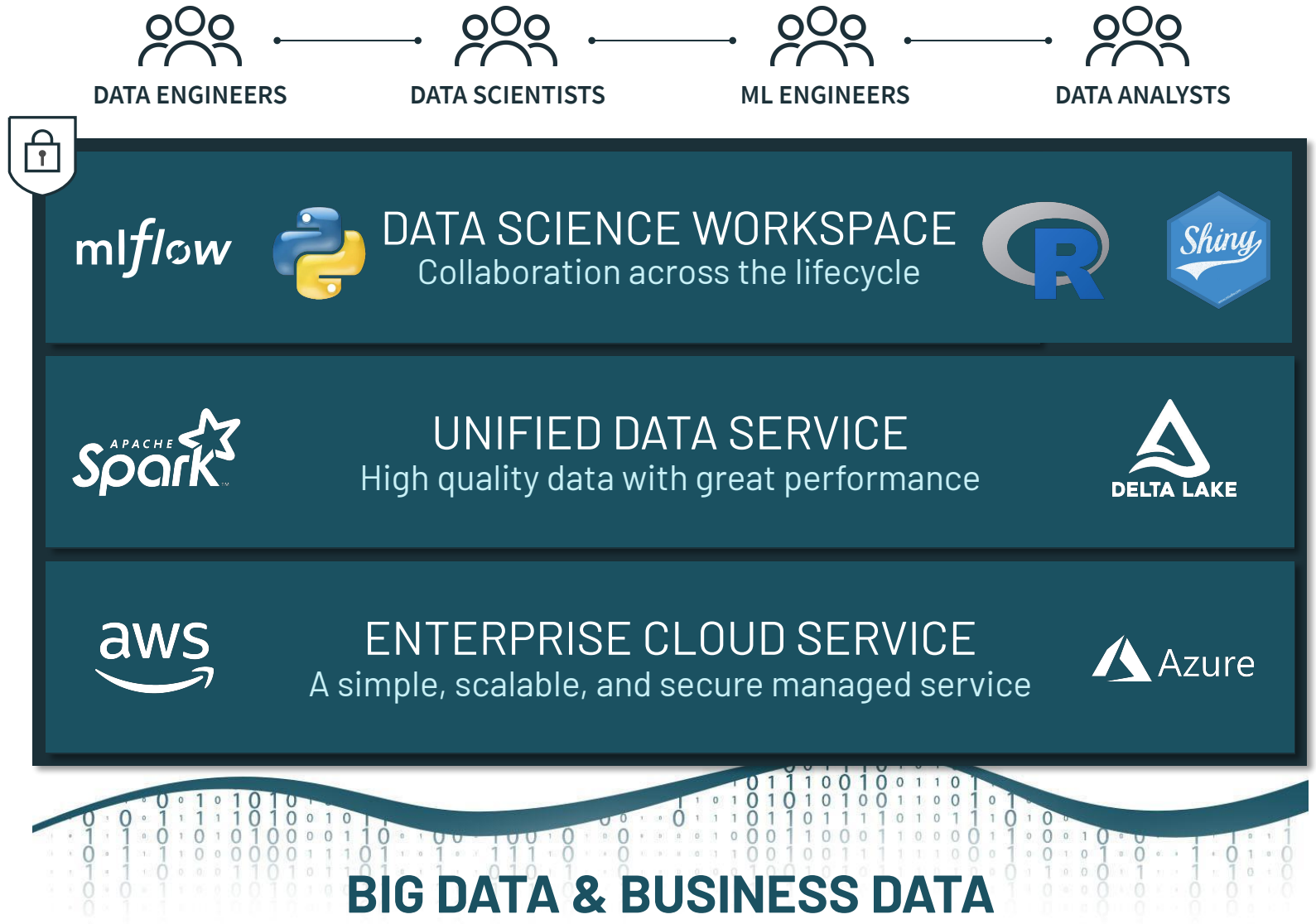
databricks

Databricks Unified Data Analytics Platform

Data science, ML, and analytics on one cloud platform

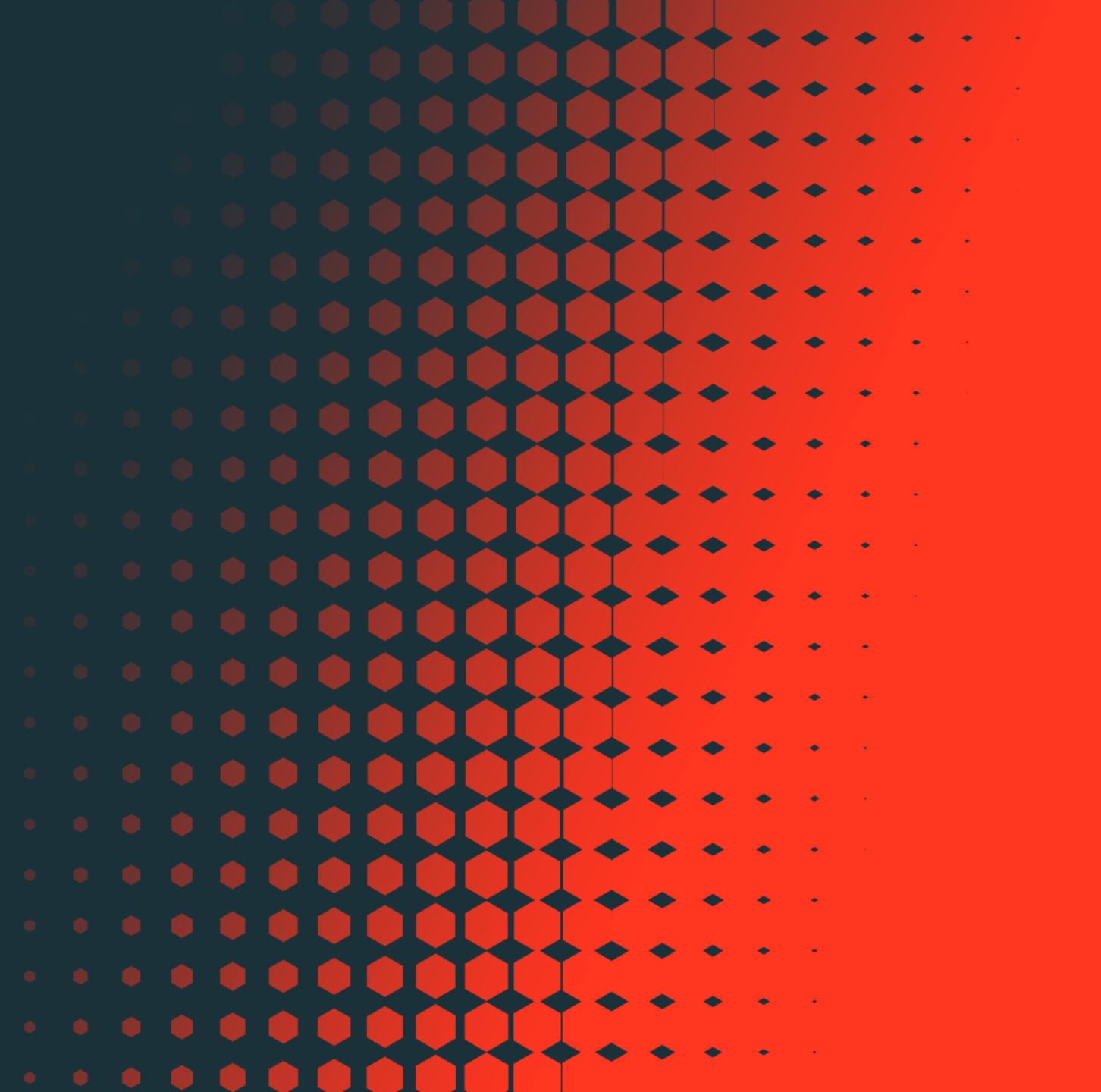
Access all business and big data in **open data lake**

Securely integrates with your **cloud ecosystem**



databricks

Q&A



databricks

For fun: big compute from a Shiny app

```
library('shiny')
library(SparkR)
sparkR.session()

monte.carlo <- function() { … }

ui <- basicPage( numericInput('num', label = 'Number of runs, value = 1))

server <- function(input, output) {
  res <- spark.lapply(1:input$num, monte.carlo)
  output$mean <- renderPrint({ mean(res) })
  output$sd <- renderPrint({ sd(res) })
}
shinyApp(ui, server)
```