Appsilon
DATA SCIENCE

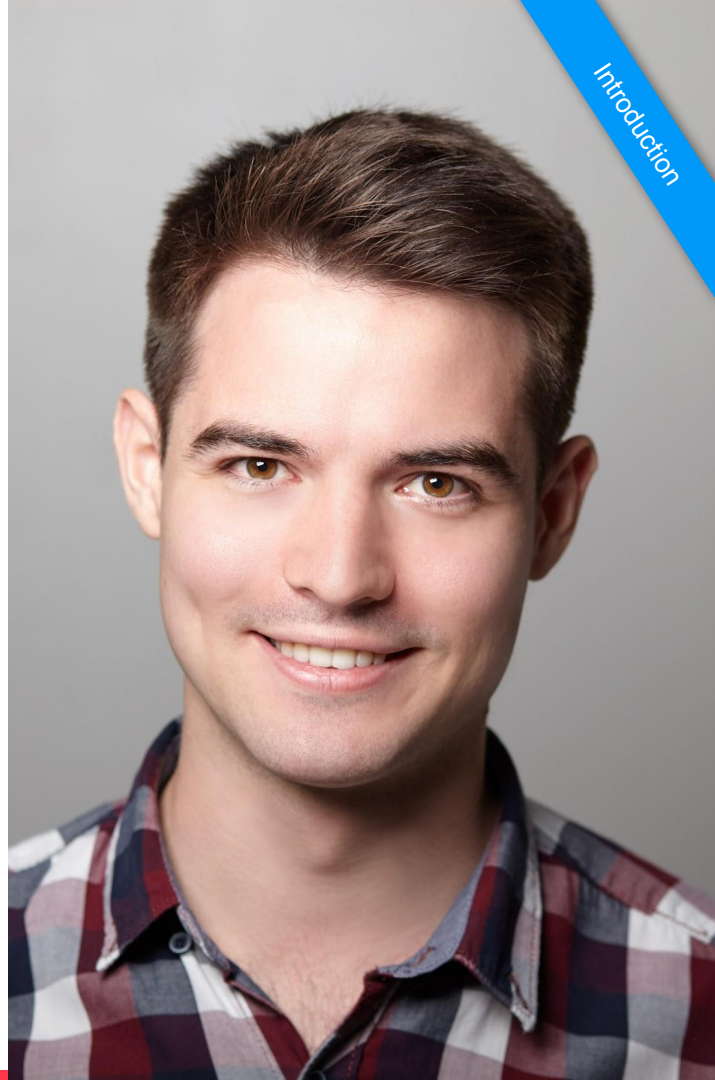
How to Scale a Shiny Dashboard

Damian Rodziewicz
damian@appsilon.com

28 July 2020 | RStudio Webinar

Appsilon
DATA SCIENCE

Hello



Damian Rodziewicz

VP of the Board & Co-Founder @ Appsilon

Passionate about Data Analysis and Programming

Previously worked at
Accenture, UBS, Microsoft, Domino Data Lab

Technology maniac

Loves psychology

Agenda

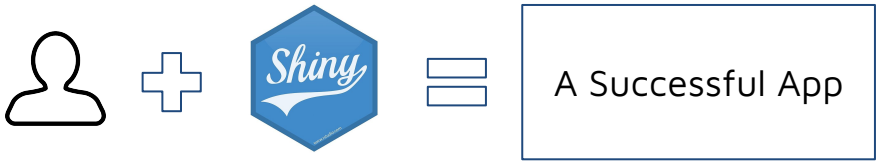
- Introduction
- Scaling
  - Leveraging Frontend
  - Extracting Computations
  - Architecture / Infrastructure

Introduction

You (or your Team)

Introduction

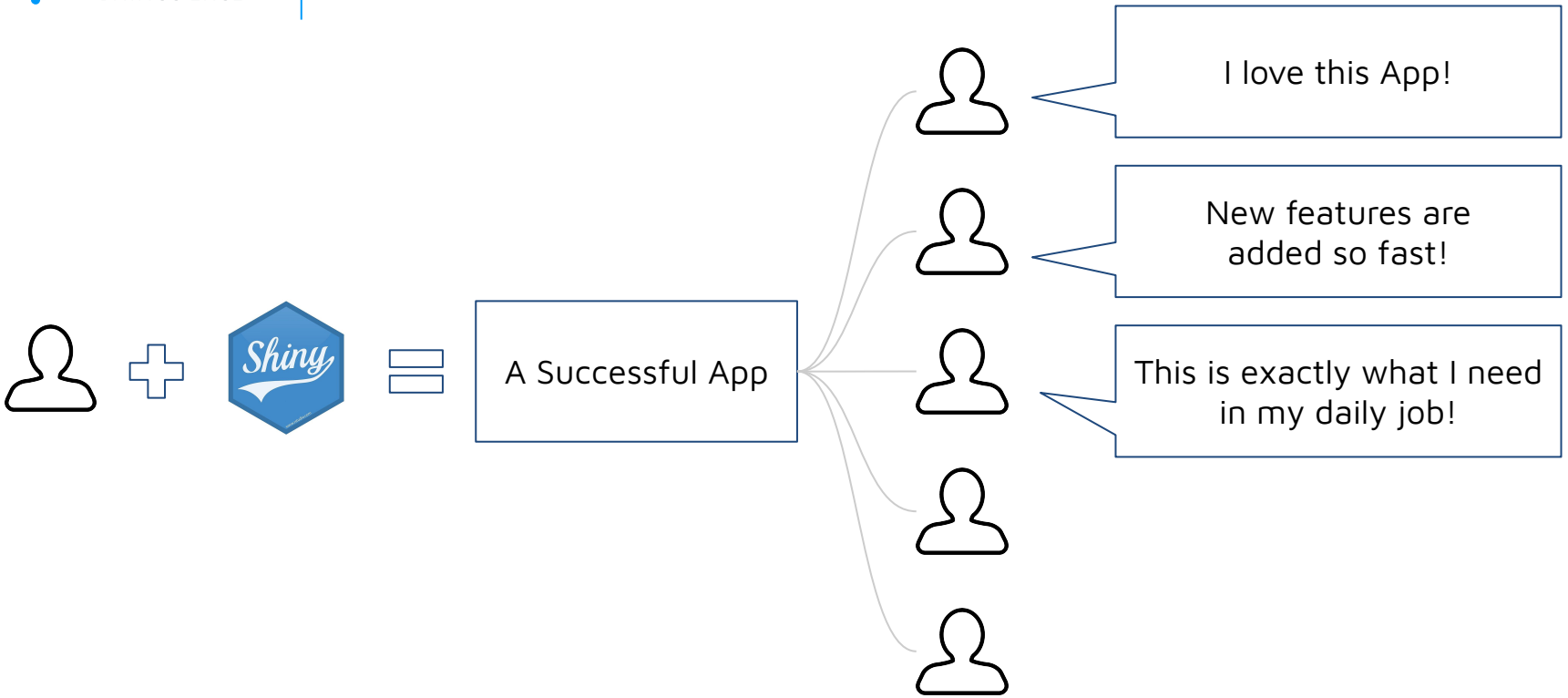
Introduction



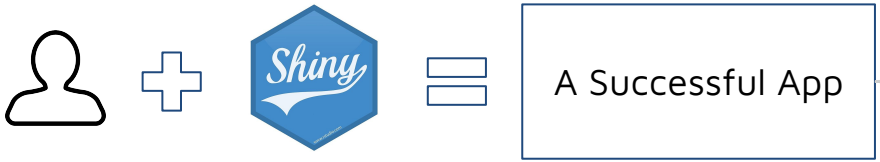
Introduction

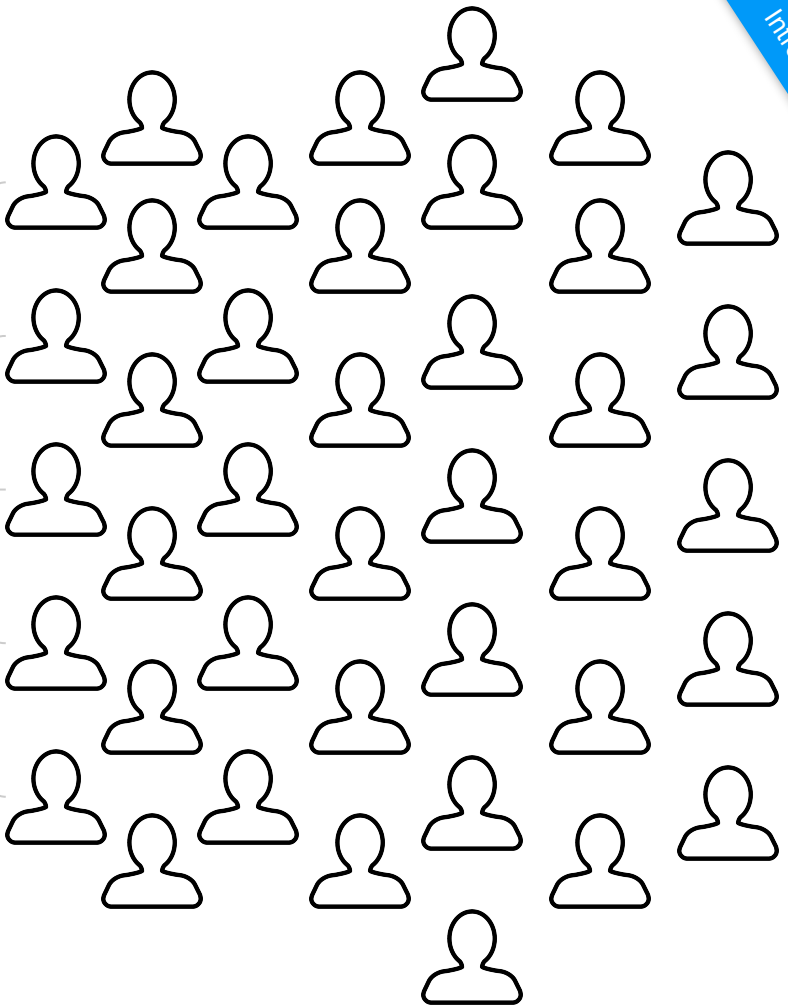
Introduction

Introduction



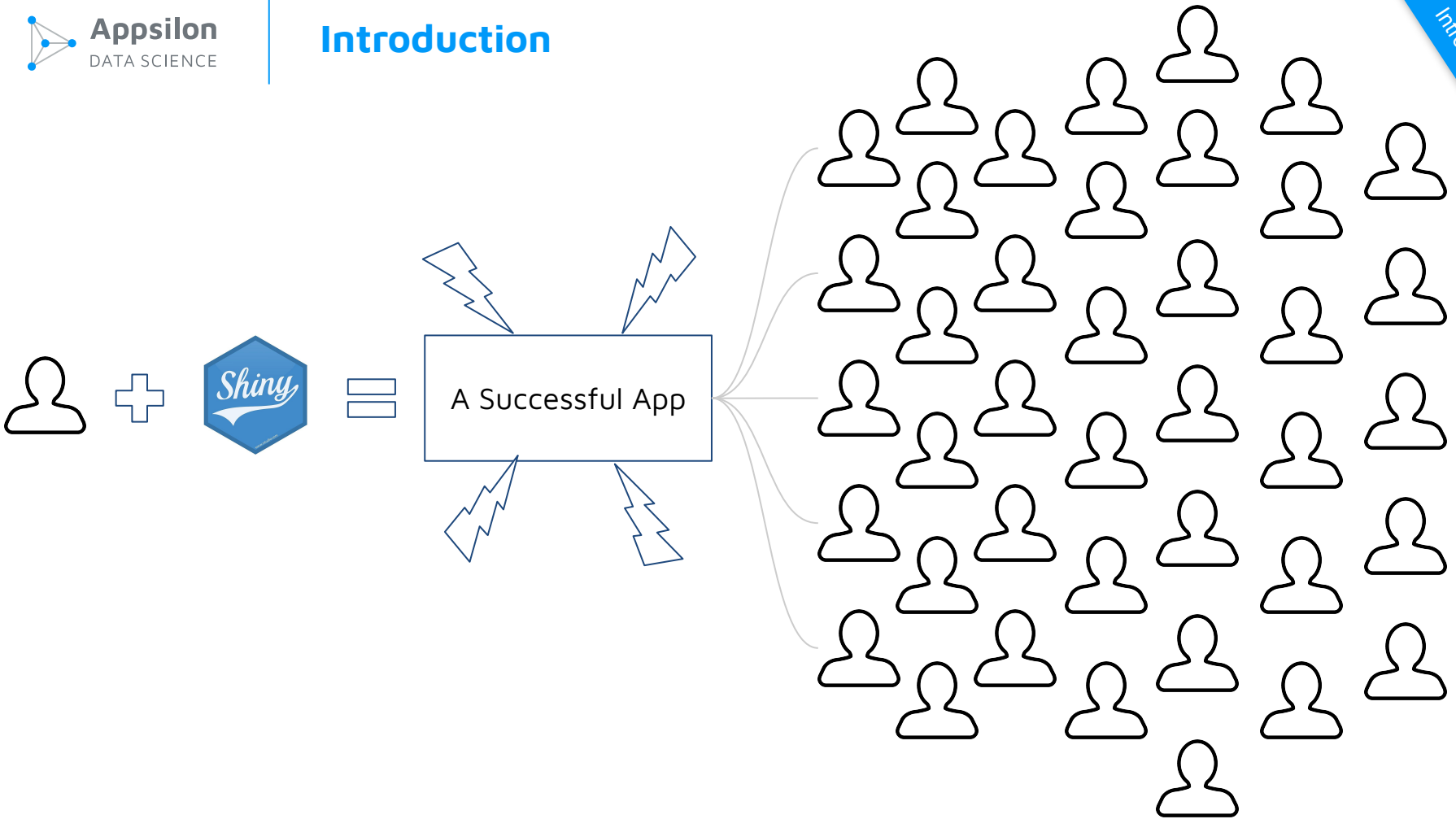
Introduction





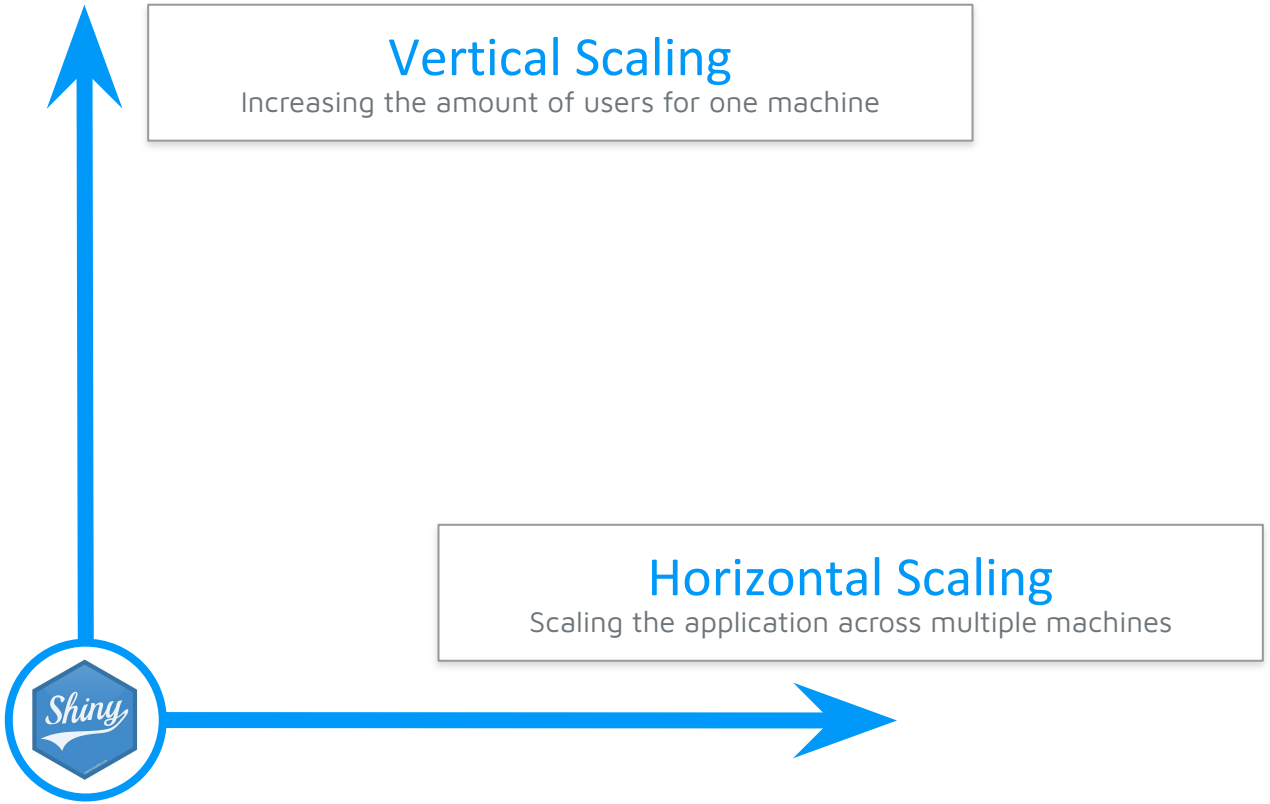
Appsilon
DATA SCIENCE

Introduction



Appsilon
DATA SCIENCE

Scalable Application



Scalable Application

Leverage Frontend

Use Javascript to handle fast user interactions that do not change data

Extract Computations

Handle resource intensive operations away from the application

Set Architecture

Prepare application to be used by many users

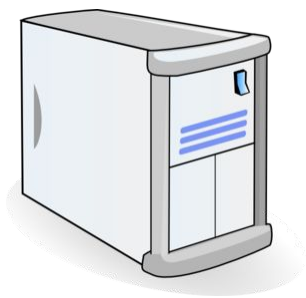
Make Shiny Layer Thin

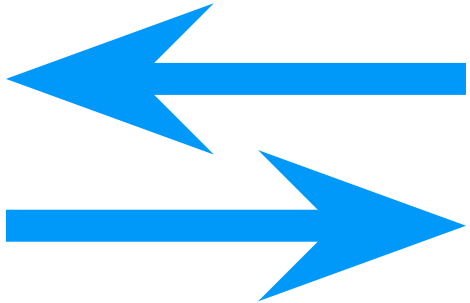
Shiny is a thin layer between the data and the interface

Scalable Application

server.R

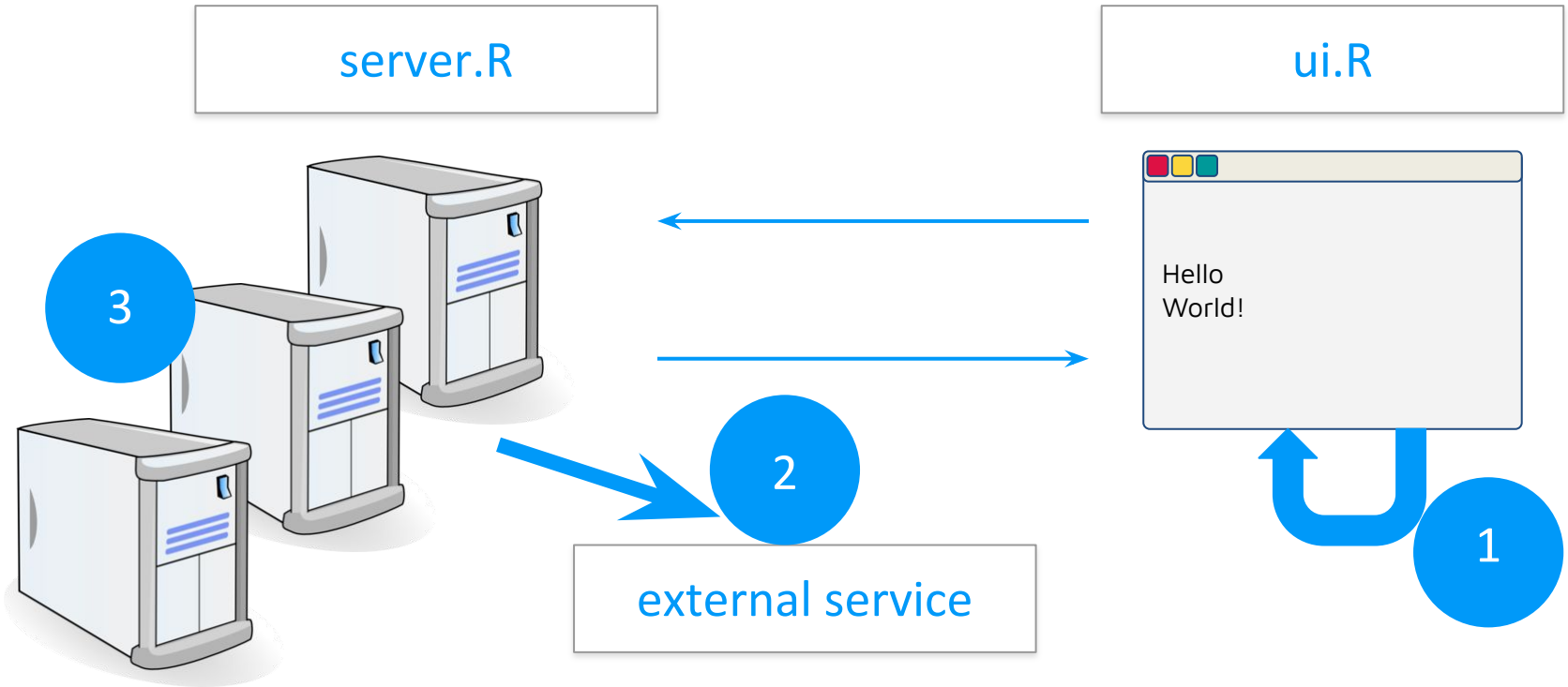
ui.R





Appsilon
DATA SCIENCE

Scalable Application



Appsilon
DATA SCIENCE

Leverage Frontend

Appsilon
DATA SCIENCE

Using the power of the browser

Leverage frontend

1. Render inputs in *ui.R* and only update them in *server.R*

```
ui <- fluidPage(
  uiOutput("render_input"),
  actionButton("click_button", label = "")
)

server <- function(input, output, session) {

  output$render_input <- renderUI({
    numericInput(
      "render_input",
      label = "I will be updated using reactivity",
      value = if(is.null(input$click_button)) 0 else
input$click_button,
      min = 0,
      max = 10
    )
  })
}
```

```
ui <- fluidPage(
numericInput(
    "update_input",
    label = "I will be updated using updateInput",
    value = 0,
    min = 0,
    max = 10
  ),
  actionButton("click_button", label = "")
)

server <- function(input, output, session) {

  observeEvent(input$click_button, {
    updateNumericInput(
      session,
      "update_input",
      value = input$click_button
    )
  })
}
```

Using the power of the browser

1. Render inputs in *ui.R* and only update them in *server.R*
2. Run inline JavaScript code with {shinyjs} package

```
ui <- fluidPage(
  actionButton("click_button", label = "")
)

server <- function(input, output, session) {

  observeEvent(input$click_button, {
    shinyjs::runjs("$('#js_update > i').toggleClass('fa-arrow-up');")
  })
}
```

Appsilon
DATA SCIENCE

Using the power of the browser

Leverage frontend

1. Render inputs in *ui.R* and only update them in *server.R*
2. Run inline JavaScript code with {shinyjs} package
3. Set all actions in JavaScript without *server.R* part

```
ui <- fluidPage(
  actionButton(
    "click_button",
    label = "I will update icons!",
    onclick = "$('#js_update > i').toggleClass('fa-arrow-up');"
  )
)
```

Appsilon
DATA SCIENCE

Extract Computations: Remote API

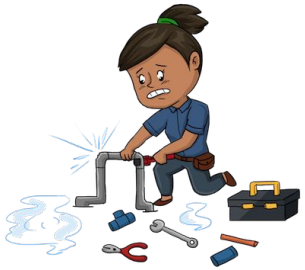
Appsilon
DATA SCIENCE

Plumber

Extract computations

Appsilon
DATA SCIENCE

Loading Data Into Shiny



Extract computations

LOAD ONLY WHAT IS NEEDED

The entire dataset is rarely needed in the application. Usually the first user action within the app is to filter/select a subset of data. First, select – then load.

USE EFFICIENT DATA LIBRARIES

Manipulate data with package like {fst}, {data.table}, {arrow}. These packages will often improve app performance.

BUILD REST API

Wrap data extraction logic into a simple API with {plumber} by adding special comments.

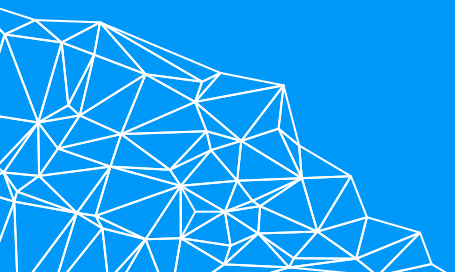
DEPLOY EASILY

Use RStudio Connect or Docker to host your API.

RStudio Connect

Appsilon
DATA SCIENCE

Extract Computations: Using a Database



Appsilon
DATA SCIENCE

Using a database - why?

Extract computations

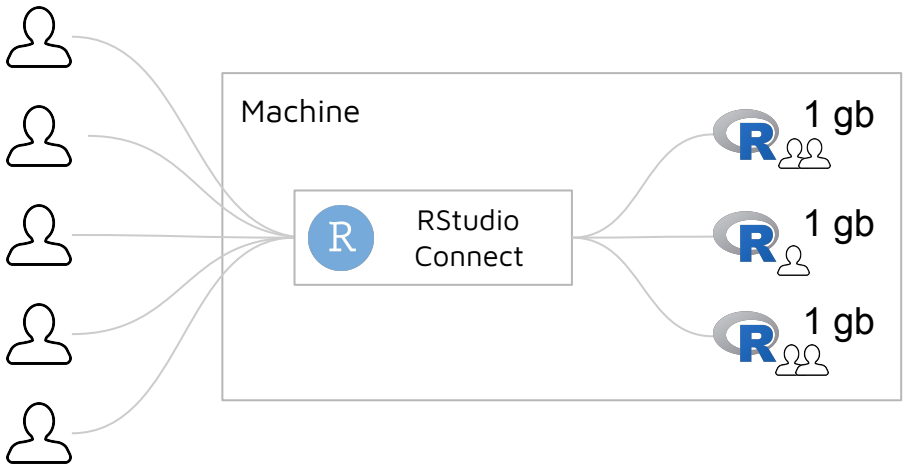
Pseudocode:

```
ui <- fluidPage(...)

data <- readRDS("./1gb_file.rds")

server <- function(input, output, session) {
  output$search_result <- … data %>% filter(value > input$query_value))
}

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

Appsilon
DATA SCIENCE

Using a database - why?

Extract computations

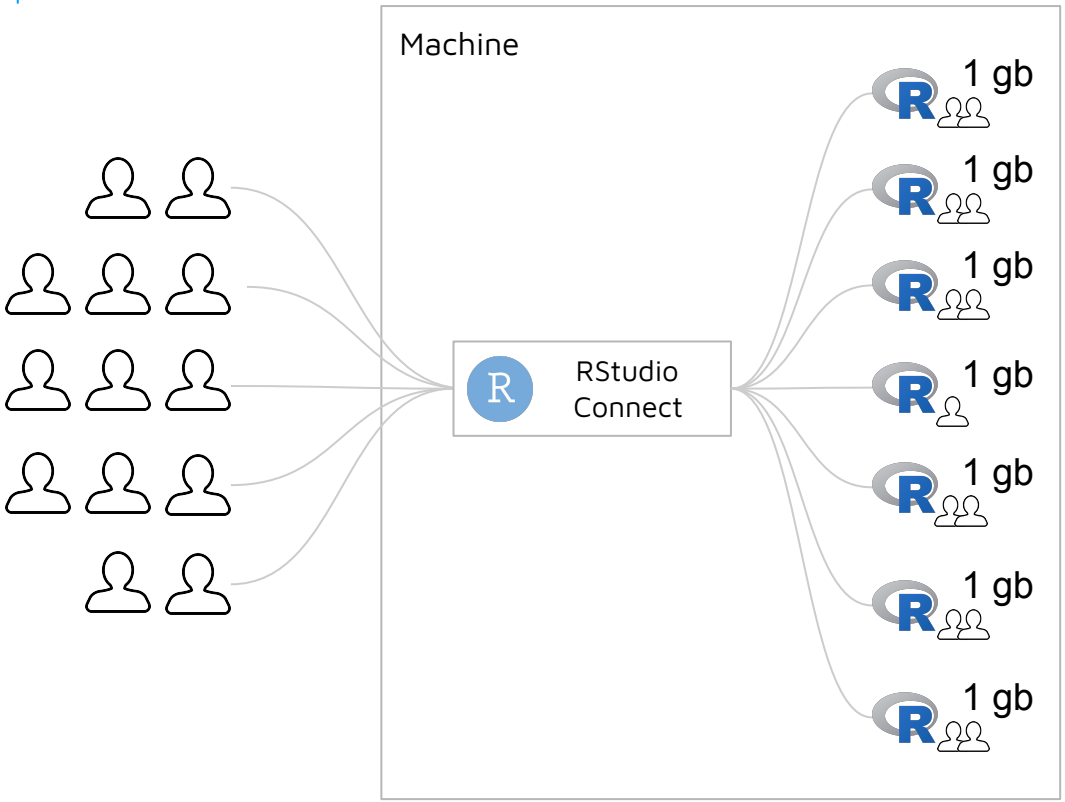


5 users

3 gb

Appsilon
DATA SCIENCE

Using a database - why?



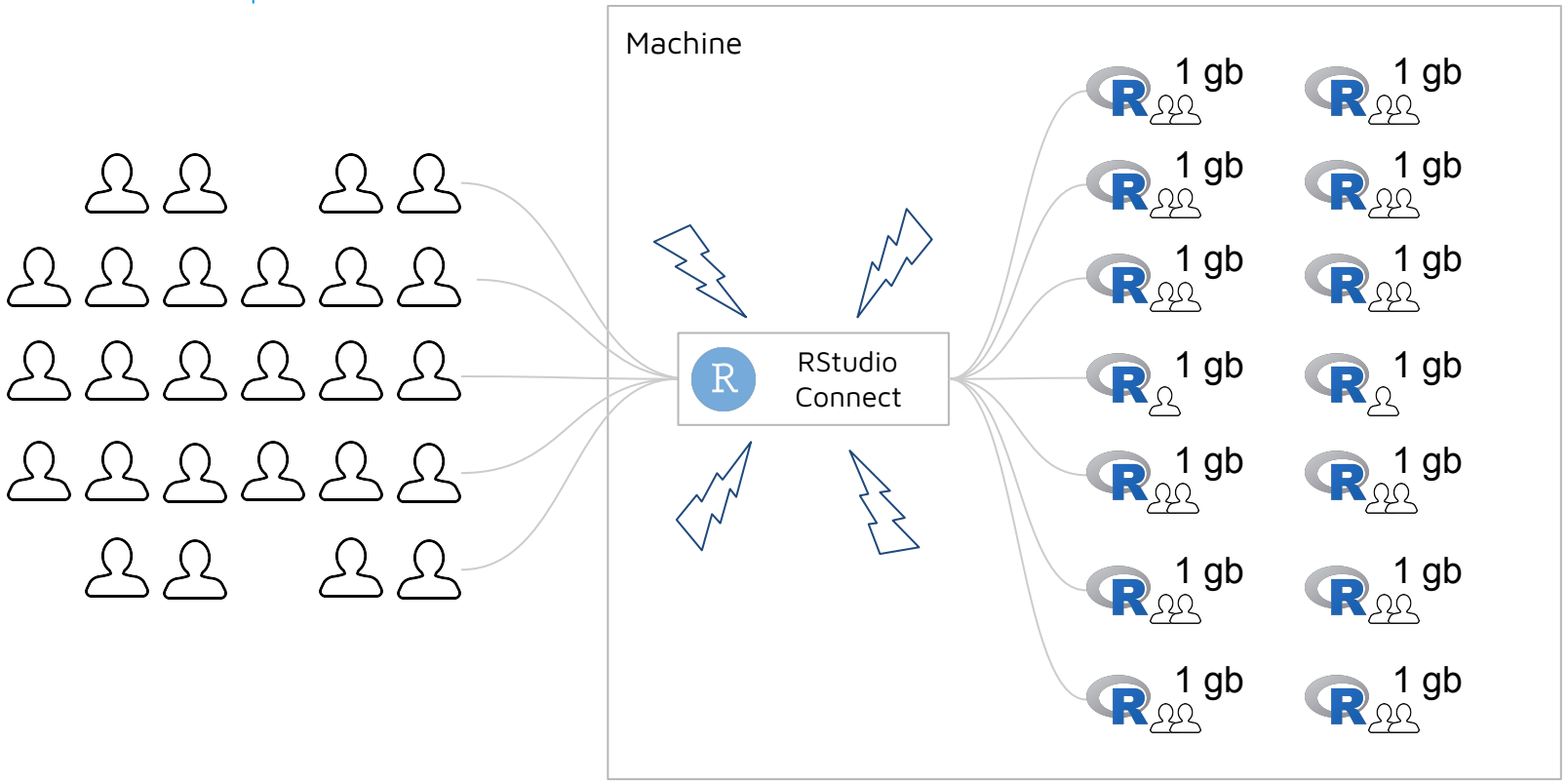
13 users

7 gb

Appsilon
DATA SCIENCE

Using a database - why?

Extract computations



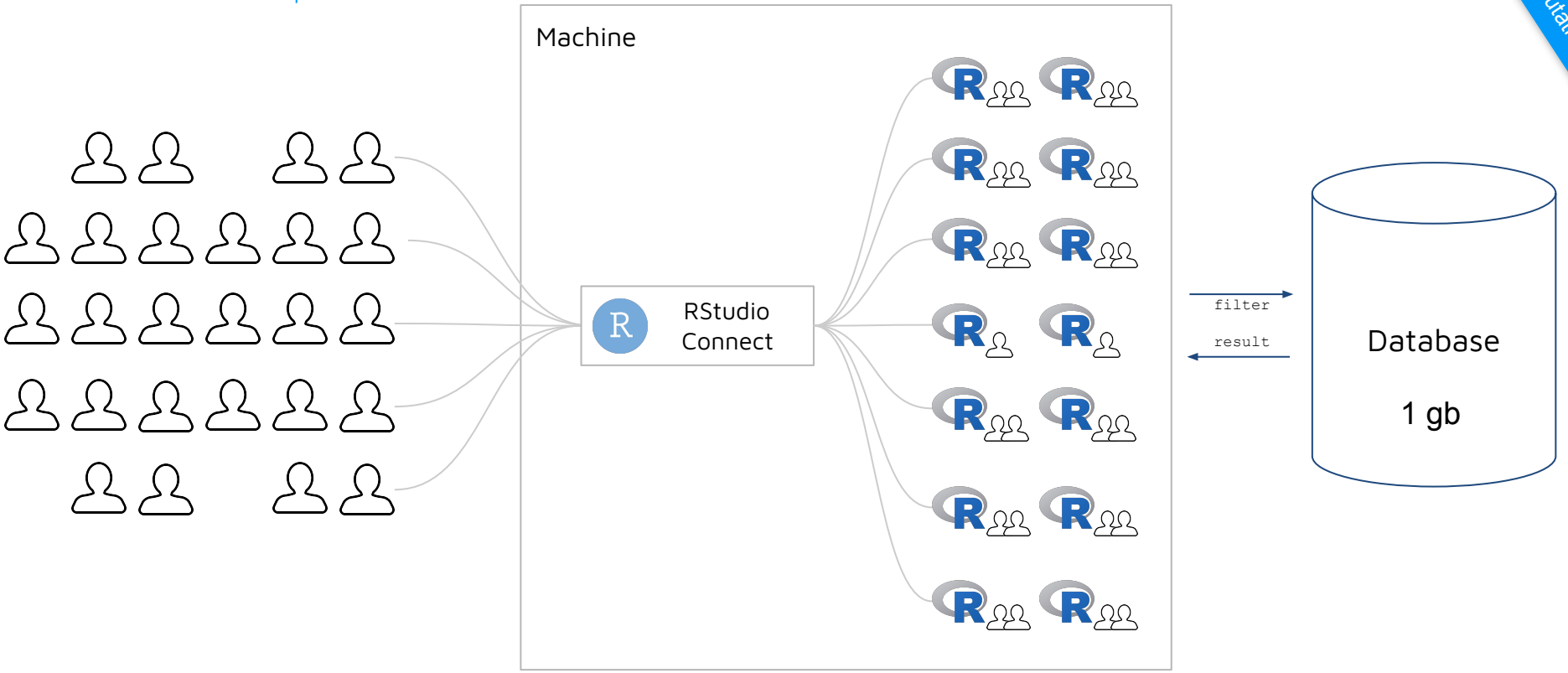
26 users

14 gb

Appsilon
DATA SCIENCE

Using a database - the result

Extract computations



26 users

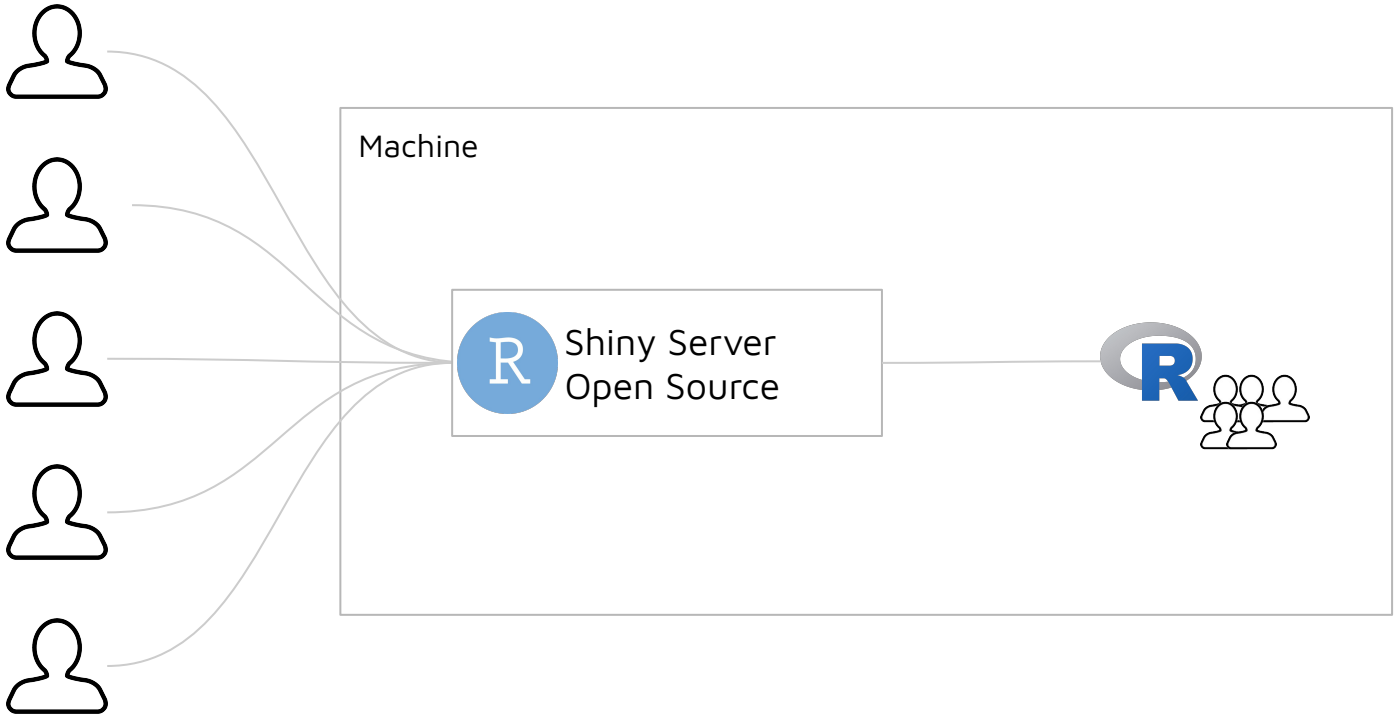
Appsilon
DATA SCIENCE

Set Architecture

Appsilon
DATA SCIENCE

Shiny Server Open Source

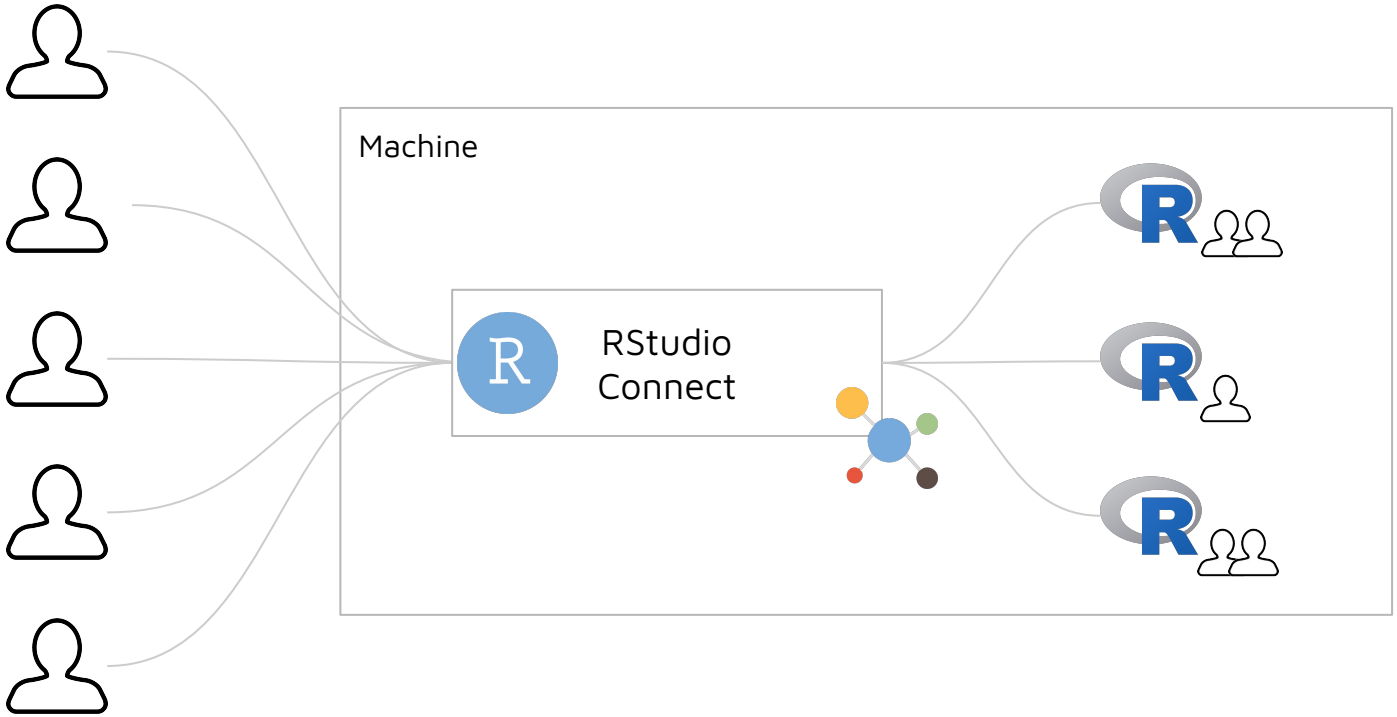
Set architecture



Appsilon
DATA SCIENCE

RStudio Connect

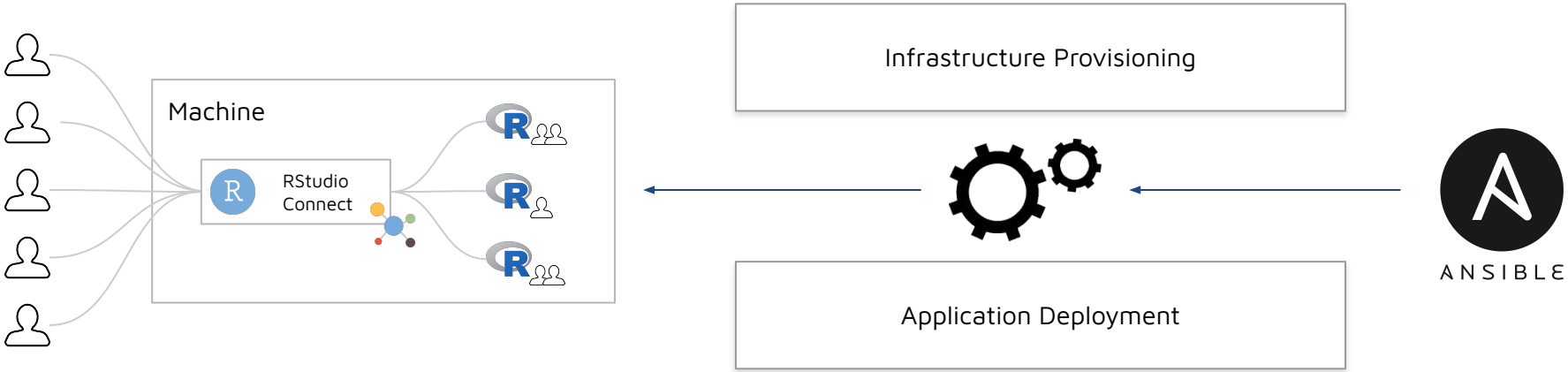
Set architecture



Appsilon
DATA SCIENCE

RStudio Connect - provisioning

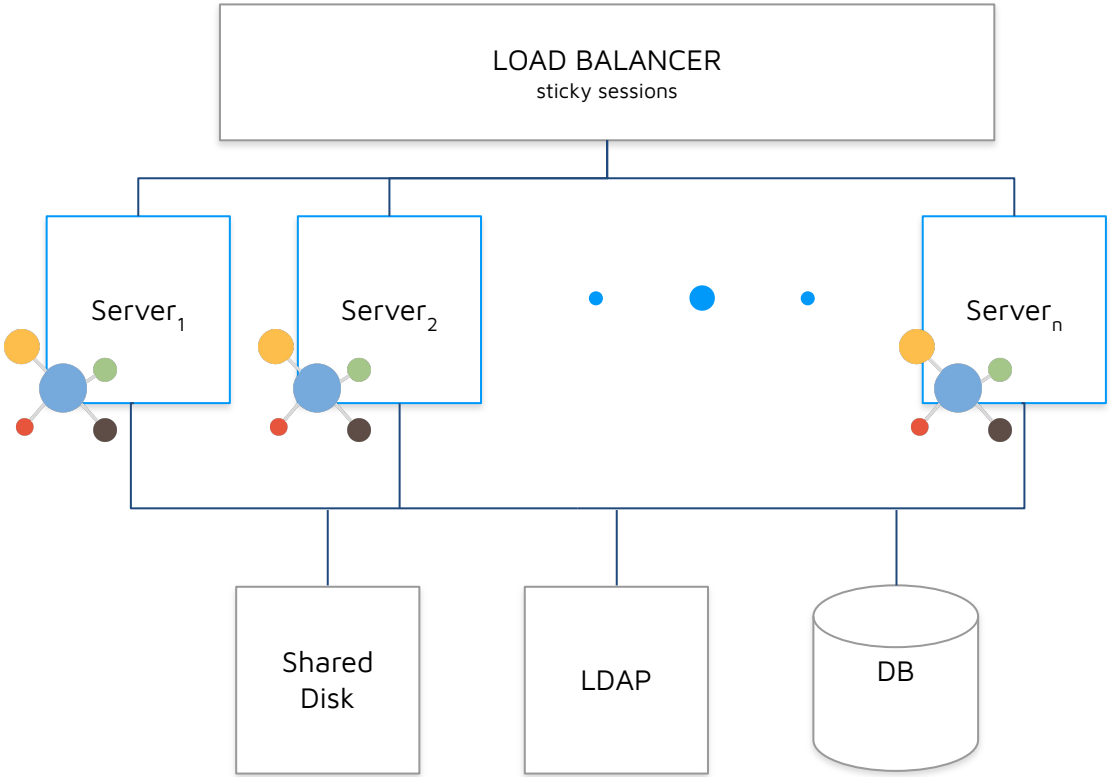
Set architecture



Appsilon
DATA SCIENCE

RStudio Connect - horizontal scaling

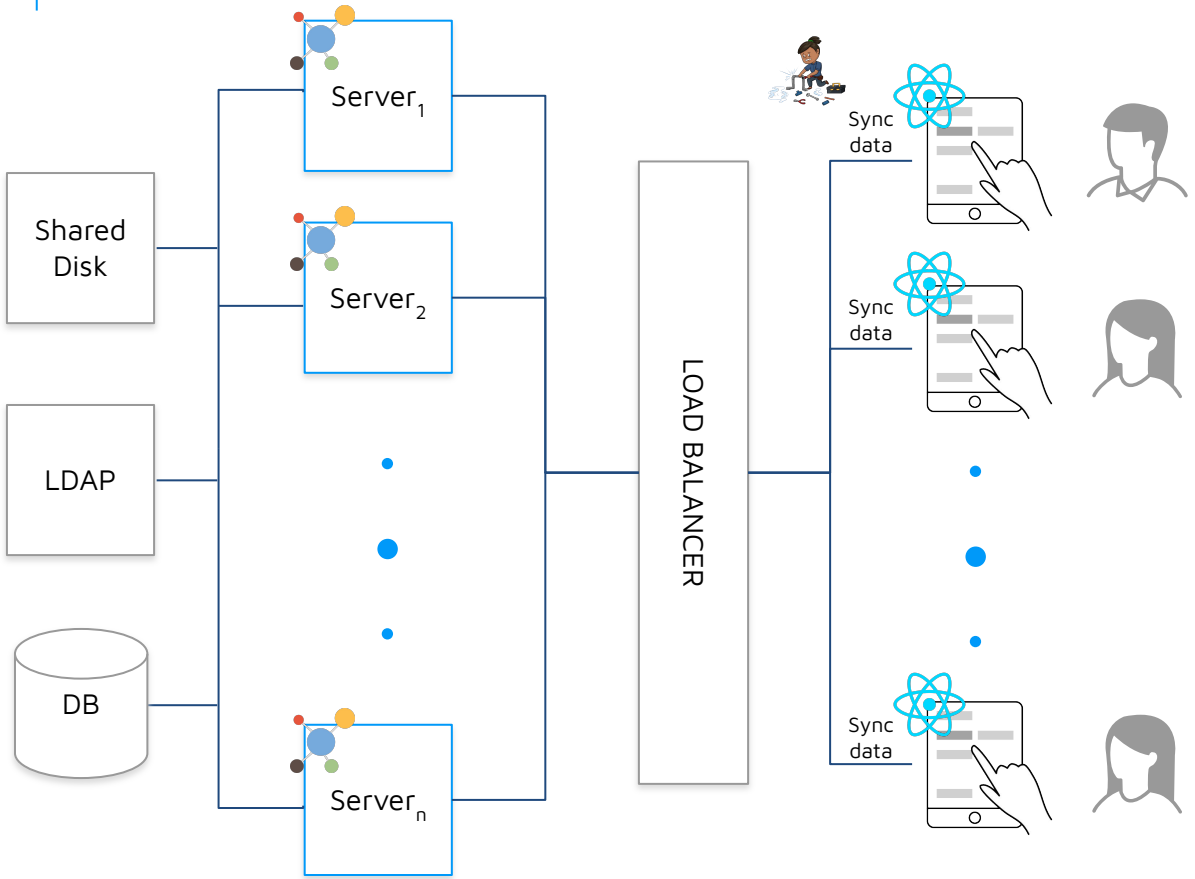
Set architecture



Appsilon
DATA SCIENCE

Architecture

Set architecture



Appsilon
DATA SCIENCE

Thank you

@D_Rodziewicz

damian@appsilon.com

appsilon.com