cloudera®

Ian Cook

`implyr`: a `dplyr` backend for Apache Impala

&

10 rules for creating a `dplyr` SQL backend

Apache Impala (incubating)

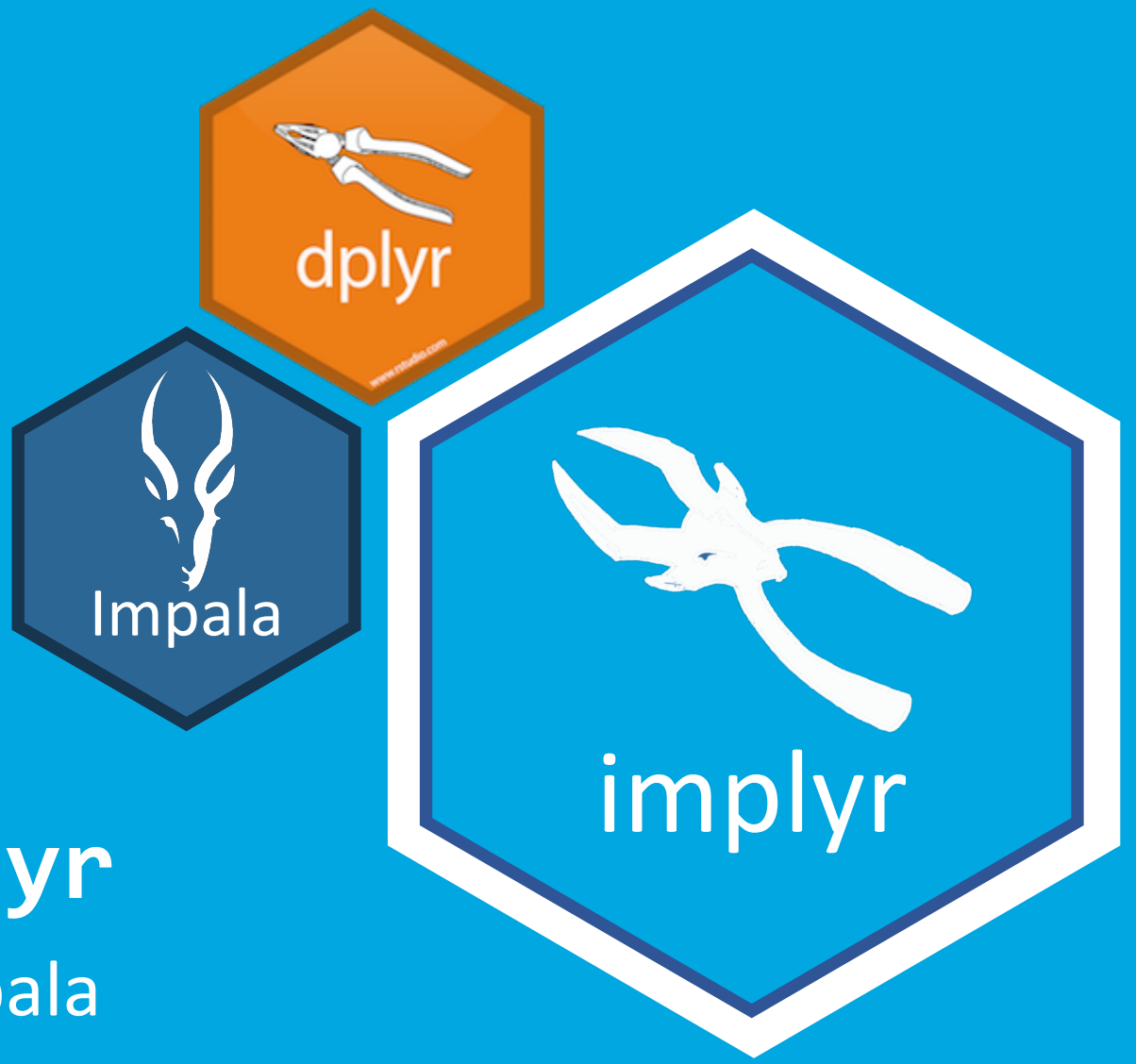
- massively parallel processing query engine
- enables low-latency SQL queries on data stored in
  - Hadoop Distributed File System (HDFS)
  - Apache HBase
  - Apache Kudu
  - Amazon Simple Storage Service (S3)

cloudera®

supporting R users

- RStudio partnership
  - `sparklyr` on the Cloudera platform
- Cloudera Data Science Workbench (CDSW)
- `implyr`

cloudera®



`implyr` = Impala + `dplyr`

- SQL backend to `dplyr` for Impala

cloudera®

File Edit View Navigate Run

Project Terminal access Clear Interrupt Stop Sessions

implyr.R

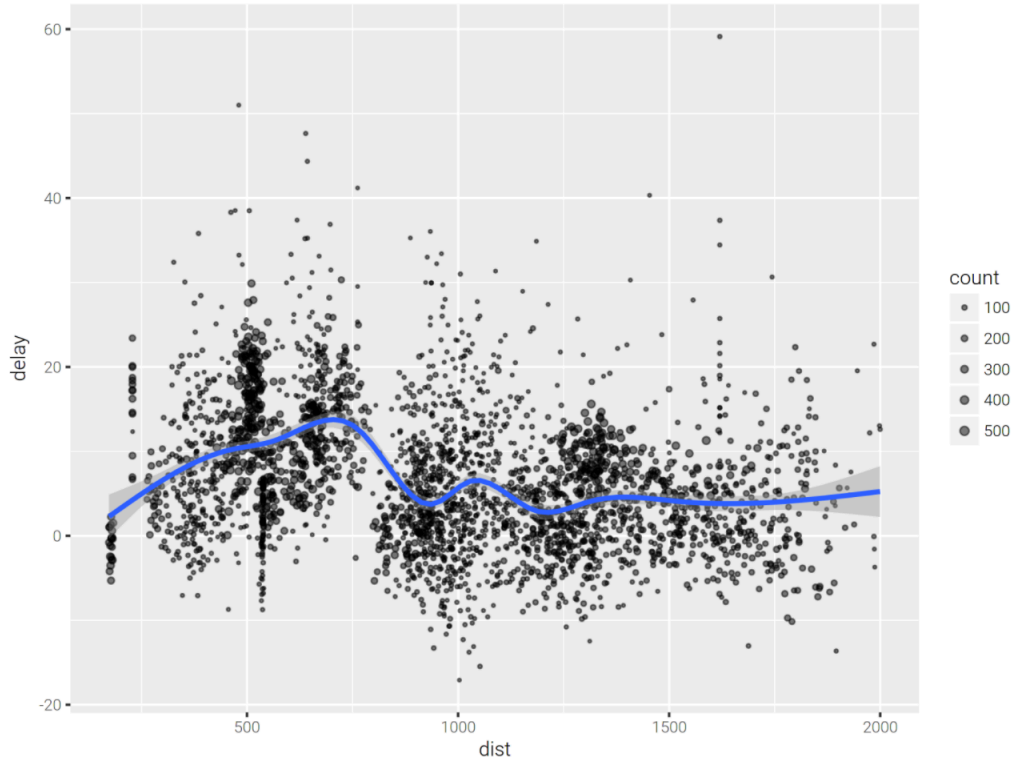
implyr
.odbc.ini
.odbcinst.ini
implyr.R
R

```
1  library(implyr)
2  library(dplyr)
3  library(ggplot2)
4
5  impala <- src_impala(
6    odbc::odbc(),
7    driver = "Impala ODBC Driver",
8    host = "10.0.0.126",
9    port = 21050
10 )
11
12 flights_tbl <- tbl(impala, "flights")
13
14 delay_tbl <- flights_tbl %>%
15   select(
16     tailnum,
17     distance,
18     arr_delay) %>%
19   group_by(tailnum) %>%
20   summarise(
21     n = n(),
22     dist = mean(distance),
23     delay = mean(arr_delay)) %>%
24   mutate(count = as.integer(n)) %>%
25   filter(
26     count > 20L,
27     dist < 2000L,
28     !is.na(delay)) %>%
29   arrange(
30     delay,
31     dist,
32     count)
33
34 delay <- delay_tbl %>% collect()
35
36 ggplot(delay, aes(dist, delay)) +
37   geom_point(
38     aes(size = count), alpha = 1/2) +
39   geom_smooth() +
40   scale_size_area(max_size = 2)
41
42 dbDisconnect(impala)
```

Line 40, Column 32 ★ 42 Lines R Spaces 2

```
    delay,
    dist,
    count)
> delay <- delay_tbl %>% collect()
> ggplot(delay, aes(dist, delay)) +
  geom_point(
    aes(size = count), alpha = 1/2) +
  geom_smooth() +
  scale_size_area(max_size = 2)
```

```
`geom_smooth()` using method = 'gam'
```



>

cloudera®

File Edit View Navigate Run

Project >_ Terminal access Clear Interrupt Stop Sessions

implyr.R

implyr
.odbc.ini
.odbcinst.ini
implyr.R
R

```
1  library(implyr)
2  library(dplyr)
3  library(ggplot2)
4
5  impala <- src_impala(
6    odbc::odbc(),
7    driver = "Impala ODBC Driver",
8    host = "10.0.0.126",
9    port = 21050
10 )
11
12 flights_tbl <- tbl(impala, "flights")
13
14 delay_tbl <- flights_tbl %>%
15   select(
16     tailnum,
17     distance,
18     arr_delay) %>%
19   group_by(tailnum) %>%
20   summarise(
21     n = n(),
22     dist = mean(distance),
23     delay = mean(arr_delay)) %>%
24   mutate(count = as.integer(n)) %>%
25   filter(
26     count > 20L,
27     dist < 2000L,
28     !is.na(delay)) %>%
29   arrange(
30     delay,
31     dist,
32     count)
33
34 delay <- delay_tbl %>% collect()
35
36 ggplot(delay, aes(dist, delay)) +
37   geom_point(
38     aes(size = count), alpha = 1/2) +
39   geom_smooth() +
40   scale_size_area(max_size = 2)
41
42 dbDisconnect(impala)
```

Line 35, Column 1 ★ 42 Lines R Spaces 2

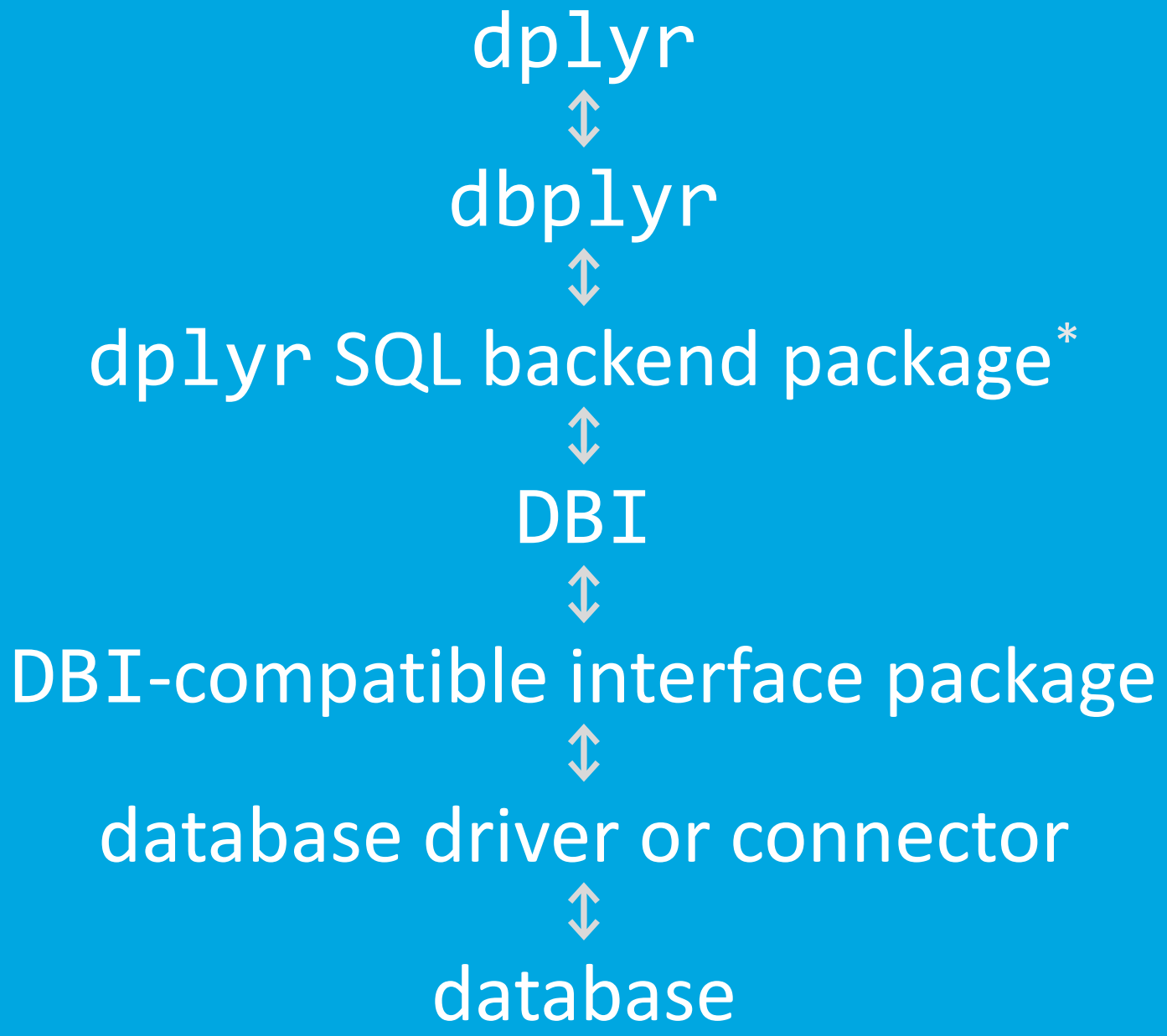
```
> delay_tbl <- flights_tbl %>%
    select(
      tailnum,
      distance,
      arr_delay) %>%
    group_by(tailnum) %>%
    summarise(
      n = n(),
      dist = mean(distance),
      delay = mean(arr_delay)) %>%
    mutate(count = as.integer(n)) %>%
    filter(
      count > 20L,
      dist < 2000L,
      !is.na(delay)) %>%
    arrange(
      delay,
      dist,
      count)
```

```
> delay_tbl %>% show_query()

<SQL>
SELECT *
FROM (SELECT `tailnum`, `n`, `dist`, `delay`, cast(`n` as int) AS `count`
FROM (SELECT `tailnum`, count(*) AS `n`, AVG(`distance`) AS `dist`, AVG(`arr_d
elay`) AS `delay`
FROM (SELECT `tailnum` AS `tailnum`, `distance` AS `distance`, `arr_delay` AS
 `arr_delay`
FROM `flights`) `zuskkmdcay`
GROUP BY `tailnum`) `endheabuya`) `ixhyhbdmpv`
WHERE ((`count` > 20) AND (`dist` < 2000) AND (NOT(((`delay`) IS NULL))))
ORDER BY `delay`, `dist`, `count`
```

>

cloudera®



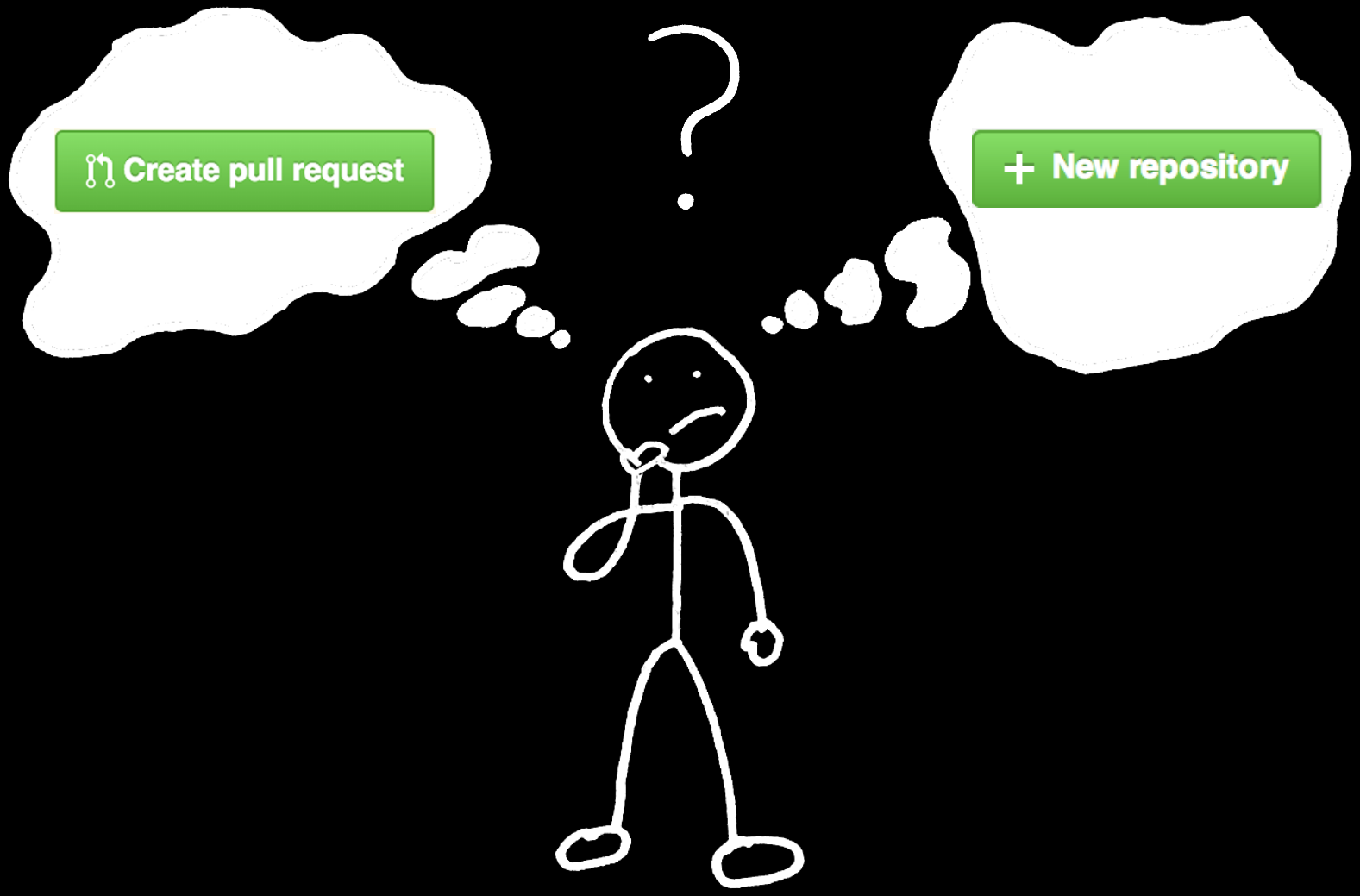
* optional

cloudera®

10 rules for creating a `dplyr` `SQL` backend

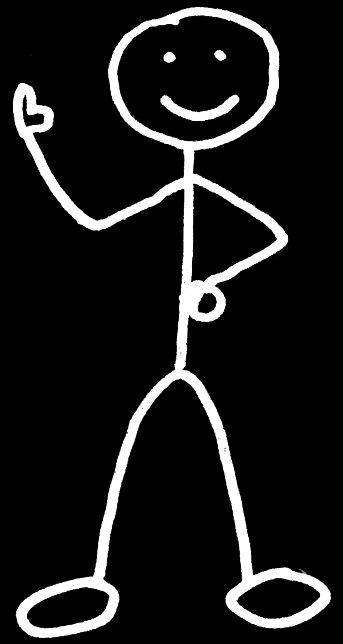
cloudera®

1. determine whether it is necessary



cloudera®

2. persist

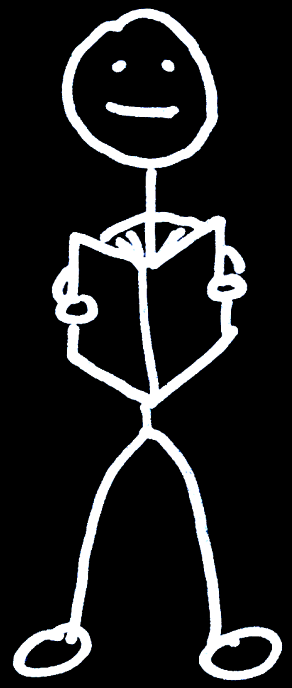


cloudera®

3. know the database inside and out

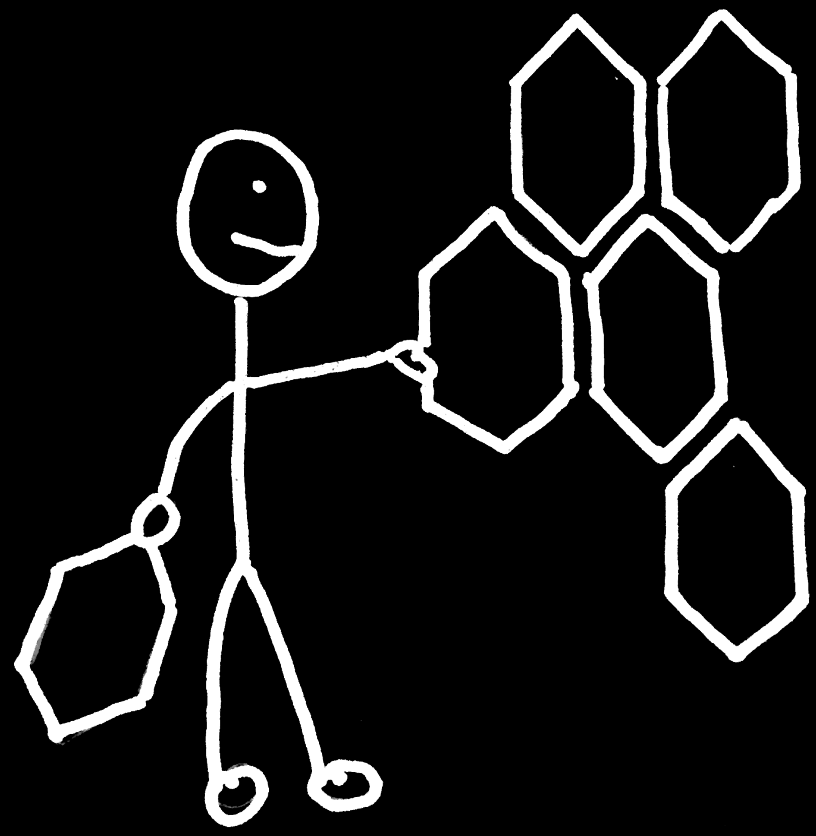
cloudera®

4. know `dbplyr` inside and out



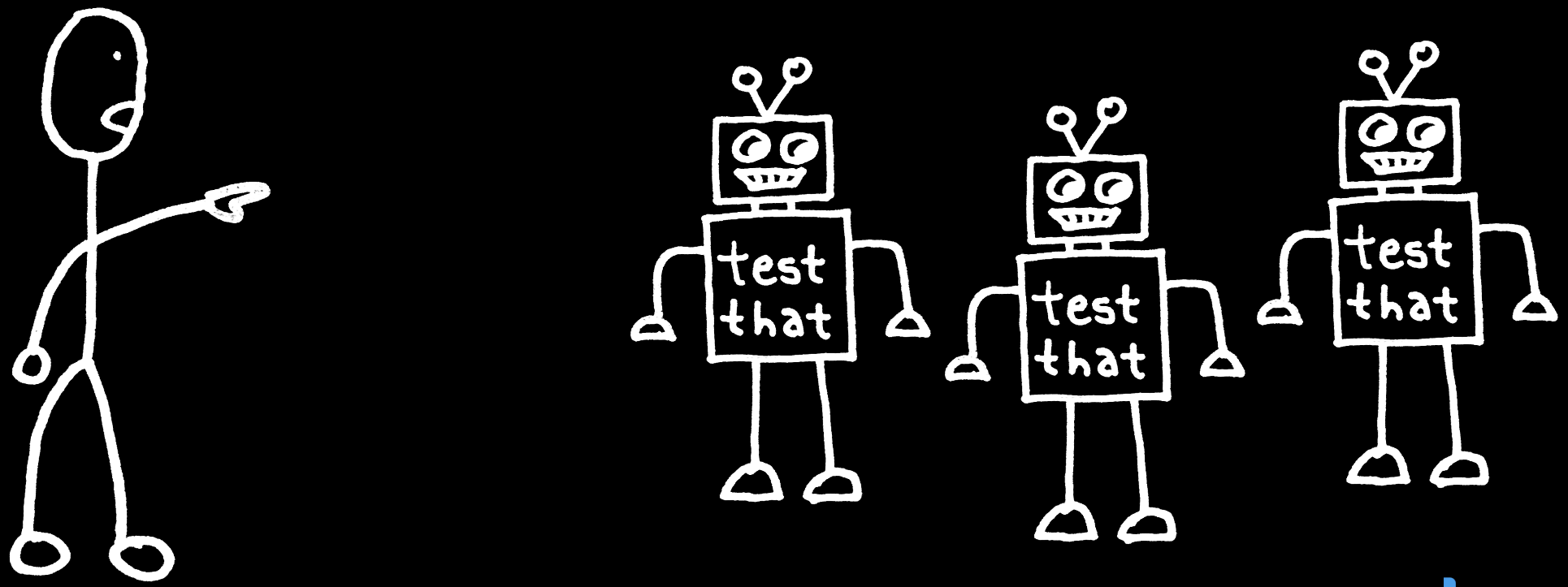
cloudera®

5. stay idiomatic



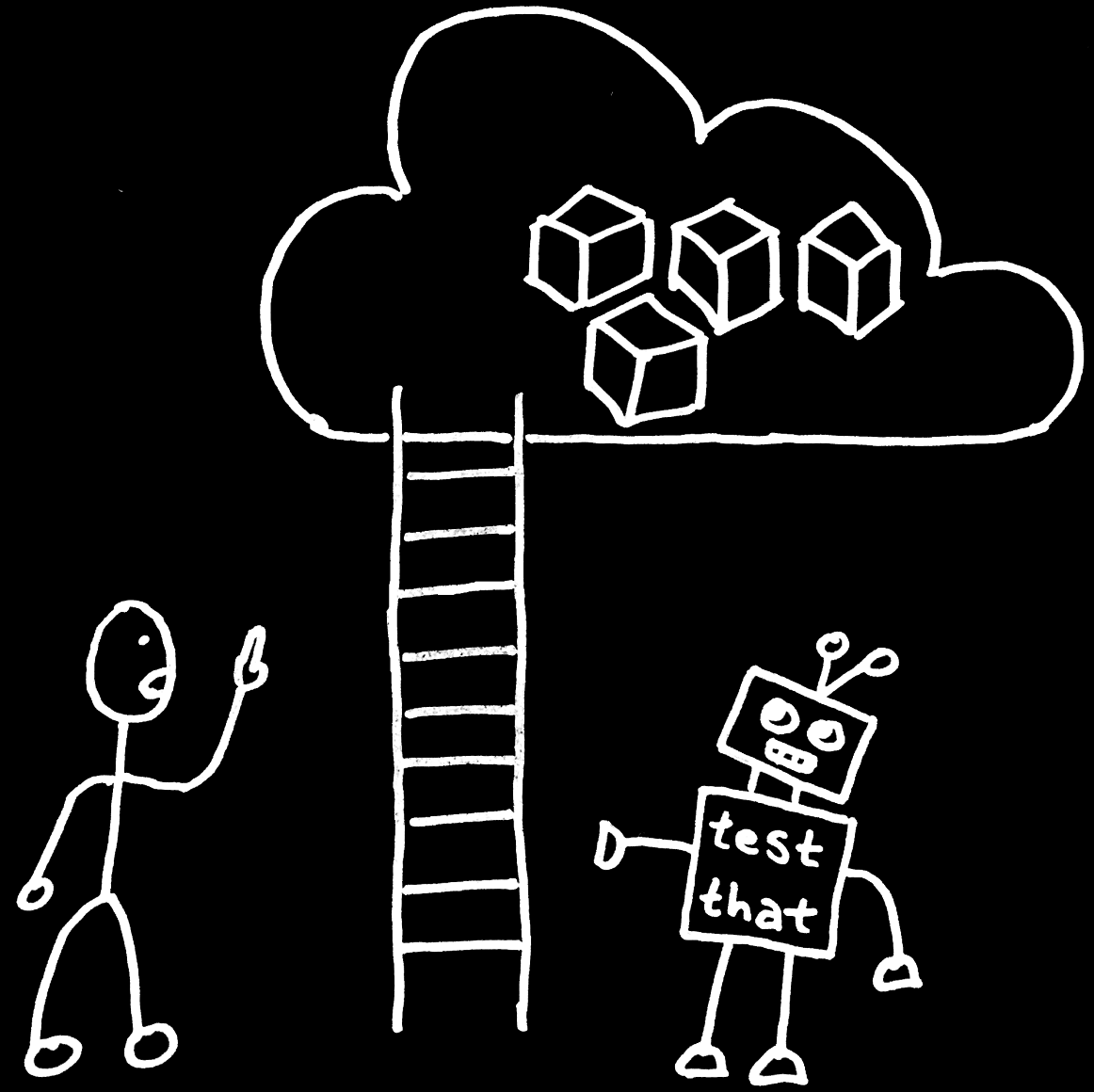
cloudera®

6. create automated tests



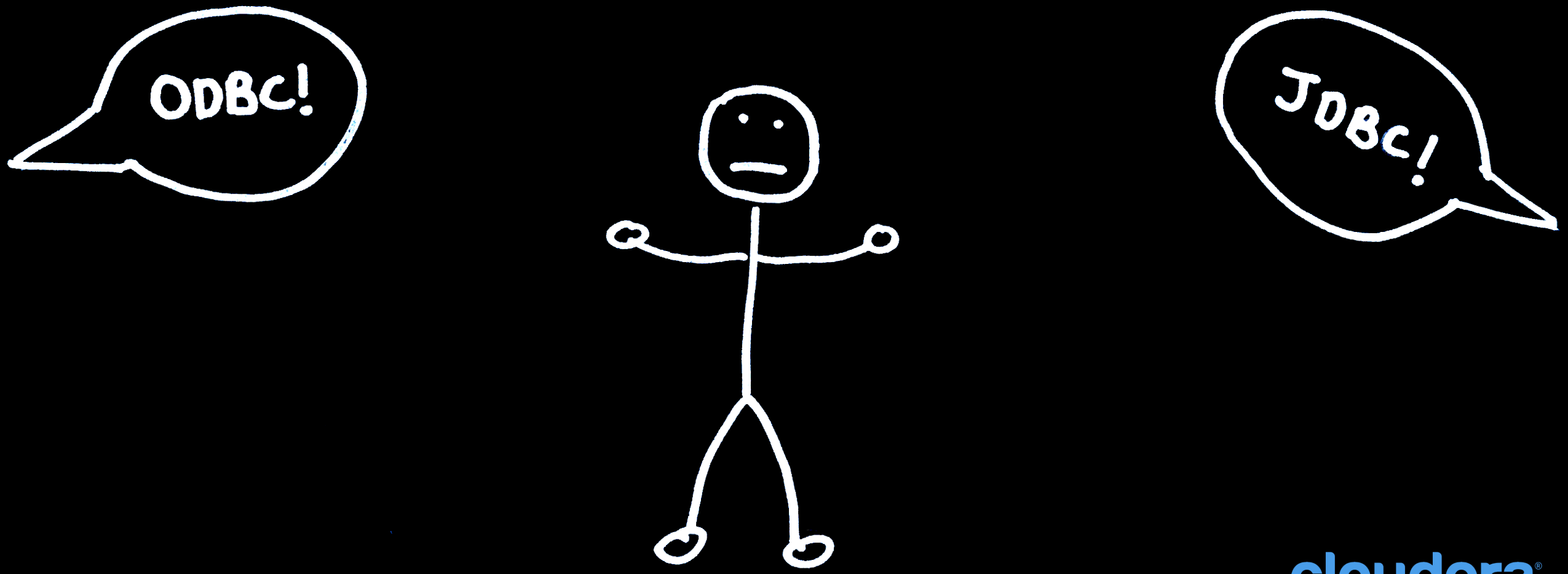
cloudera®

7. run a dedicated database instance



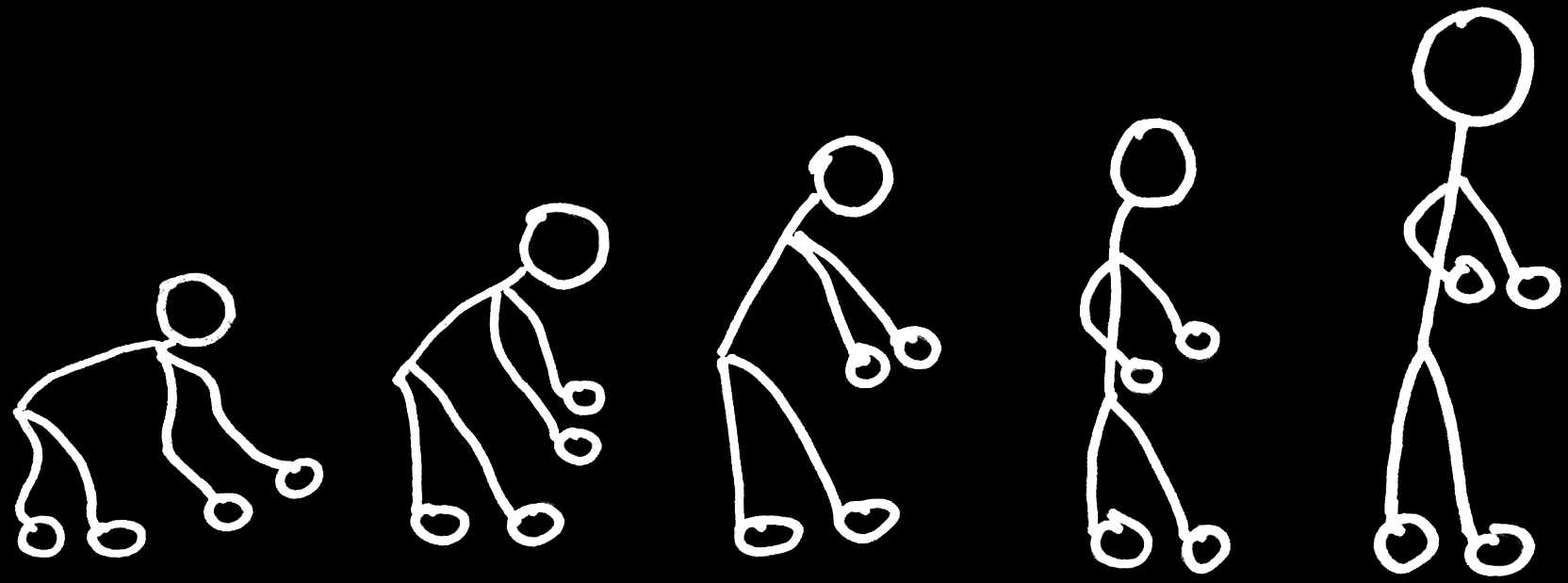
cloudera®

8. allow different ways of connecting



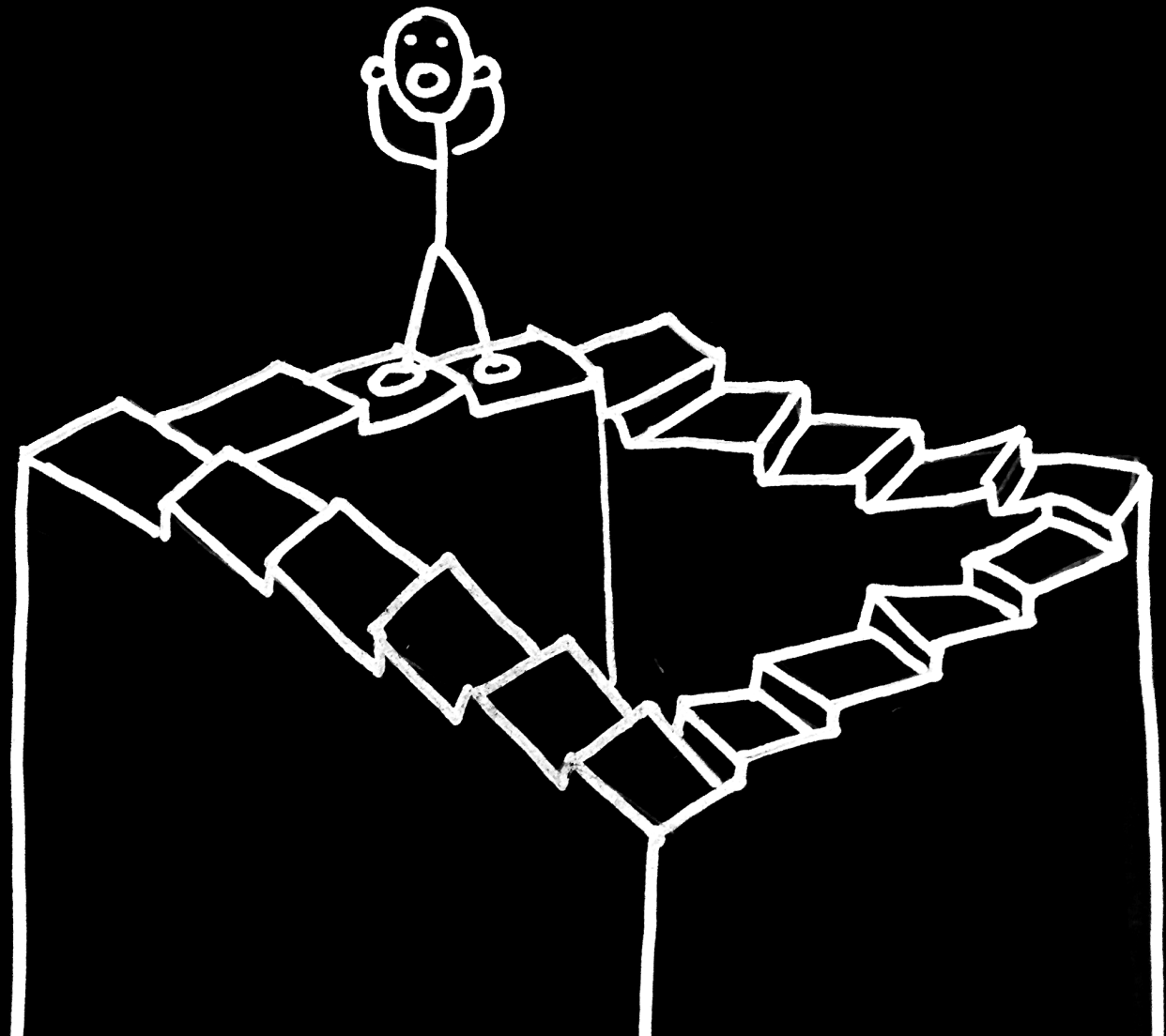
cloudera®

9. evolve with dplyr



cloudera®

10. evolve with the database



cloudera®

cloudera®

Thank you

Ian Cook
@ianmcook
ian@cloudera.com