Ali Sial DE3 Final

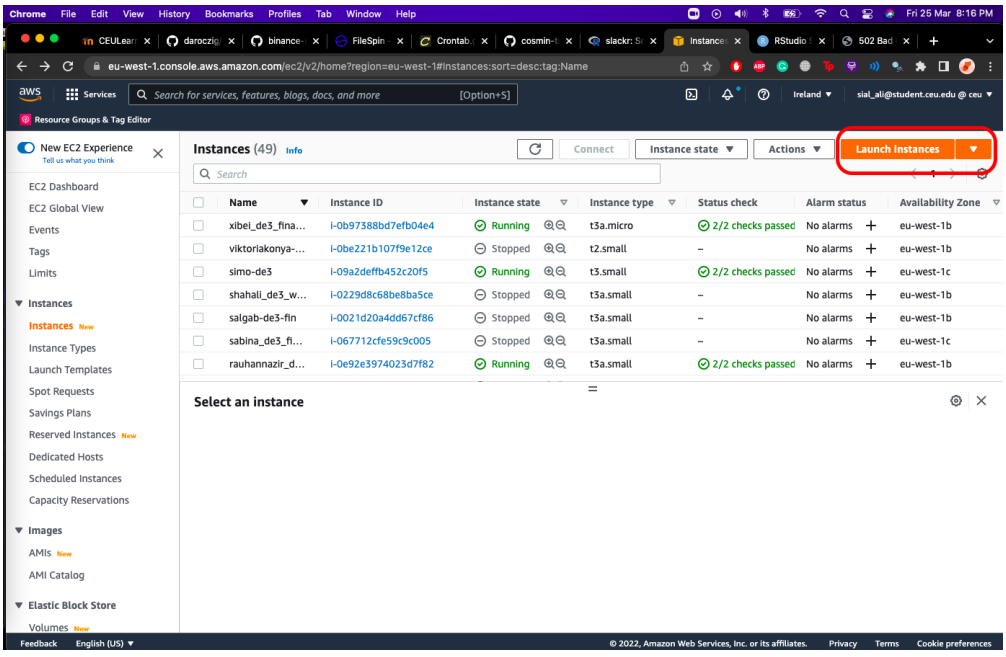
- Name: Ali Hasnain Khan Sial
- Student ID: 2101874
- Instance ID: i-0b10a7d685944b522

Introduction

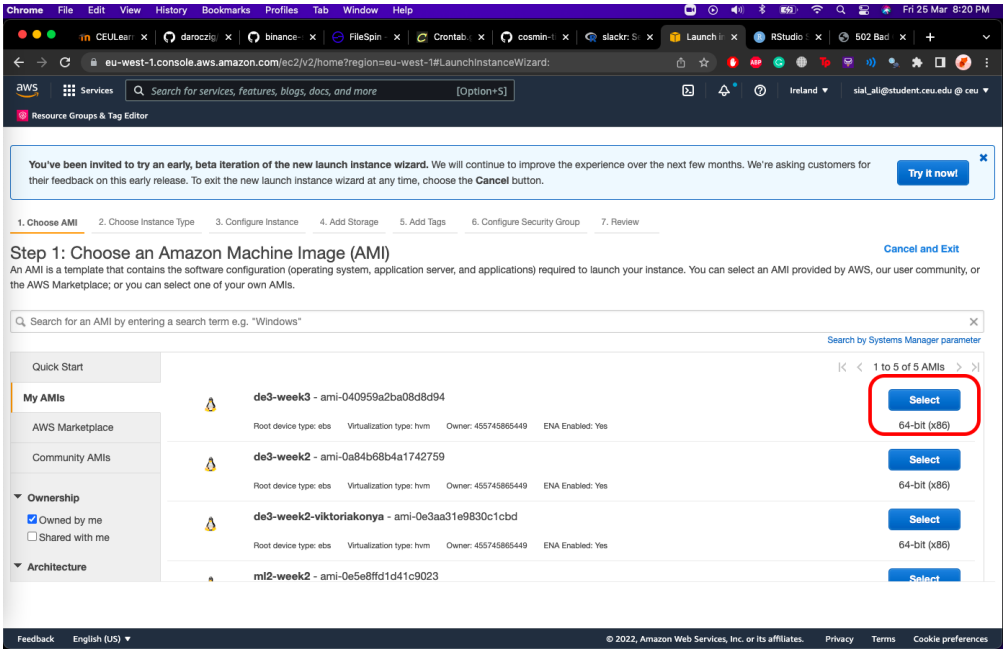
The aim of this project is Create a stream processing application using the `AWR.Kinesis` R package's daemon and Redis. For this purpose the following steps are taken.

I started by setting up the Amazon web services based on the given configurations, I opened EC2. Further, I selected the Launch Instances and selected *My AIMs* from the left corner. After this, I selected de3-week3 AMI, as it this instance is already configured. Then I choose an Instance Type *t3a.small*. Below are the screenshots of the steps I followed.

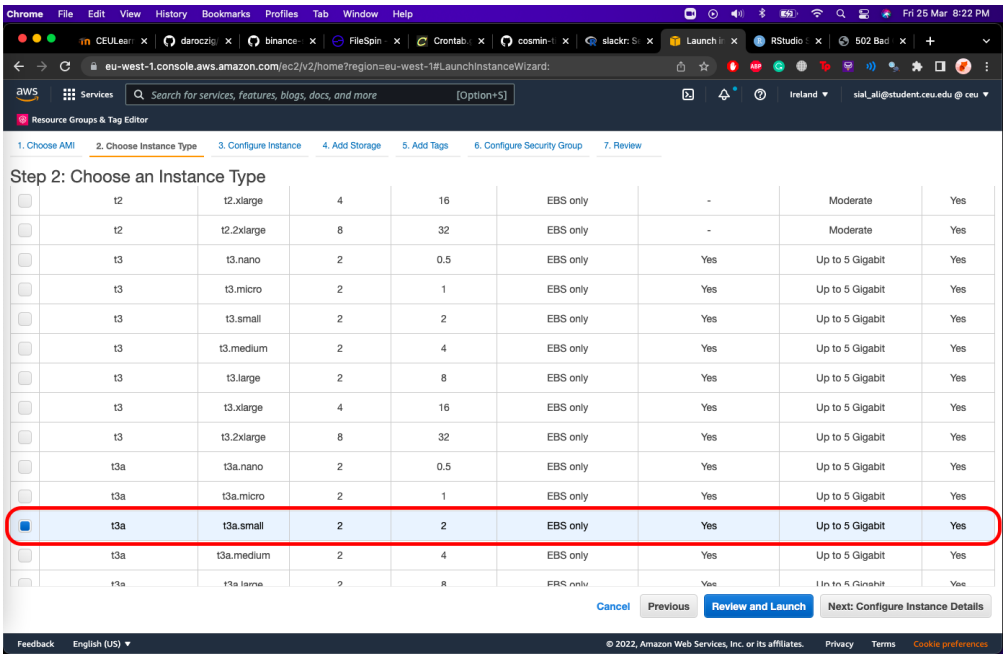
1. Launching Instance



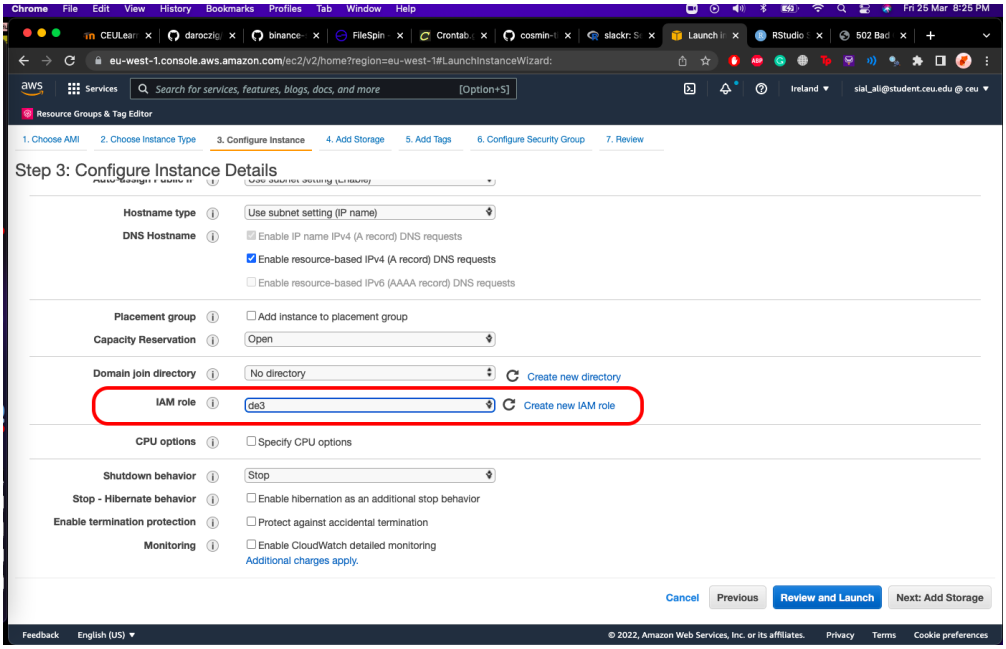
2. Selecting AMI



3. Selecting Machine Size

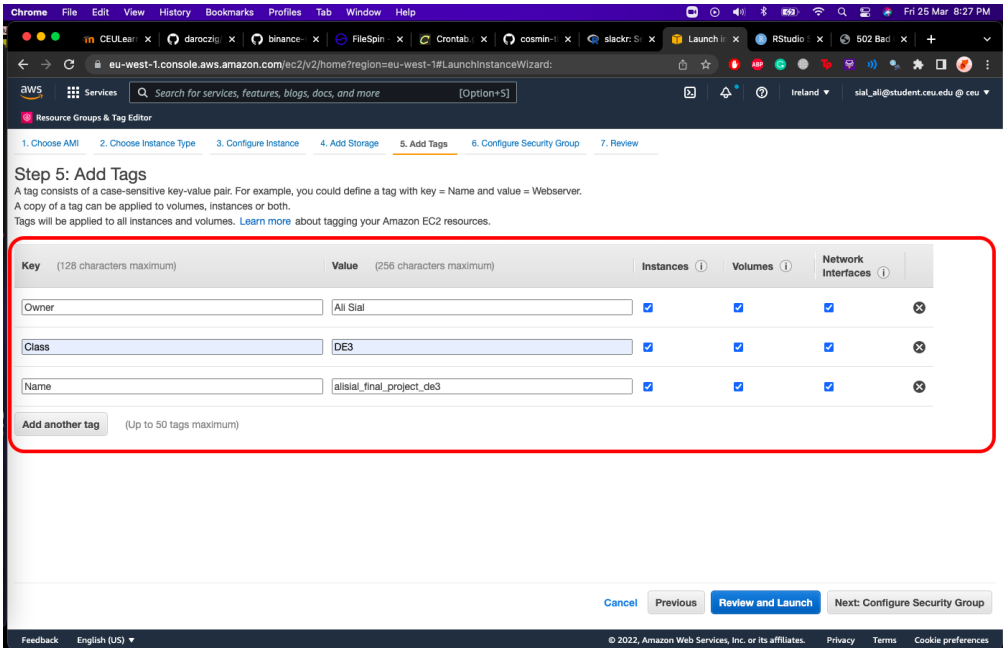


4. Selecting IAM Role



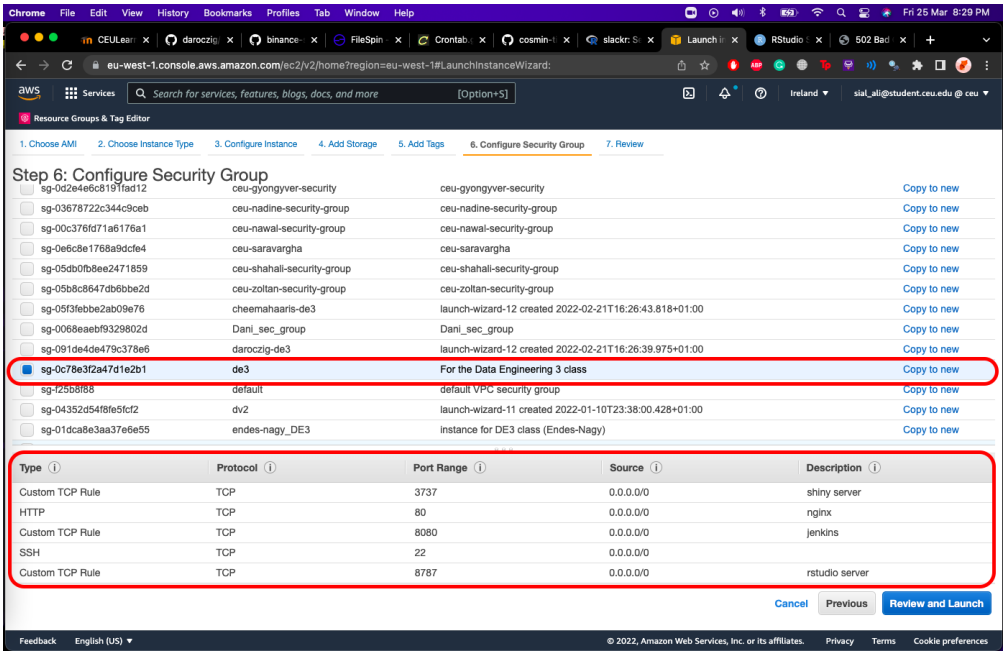
5. Adding Tags

- Owner: Ali Sial
- Class: DE3i
- Name: alisial_final_project_de3

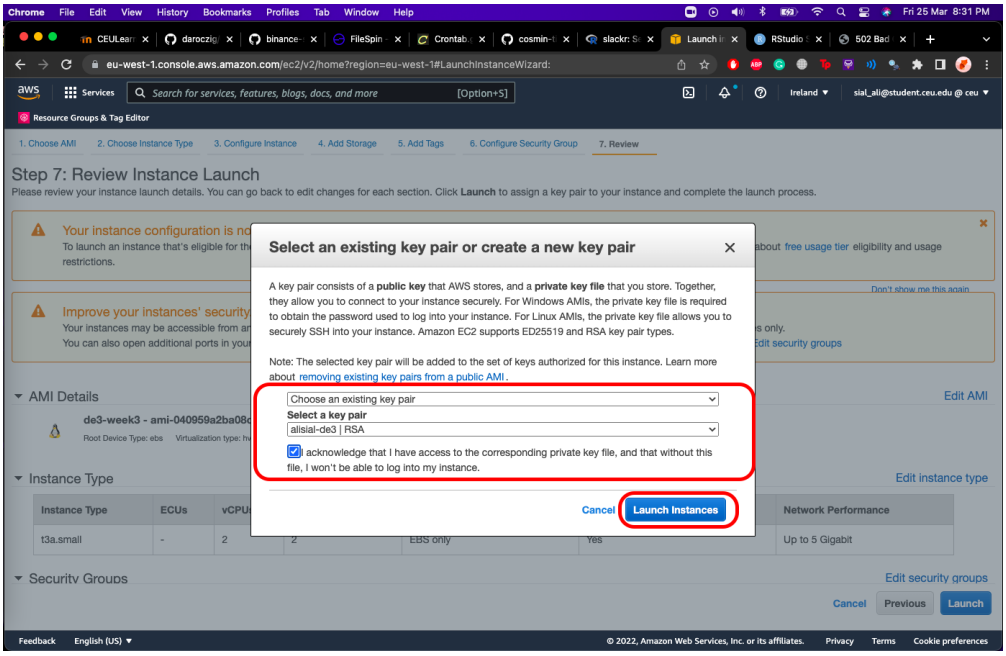


6. Selecting Security Group

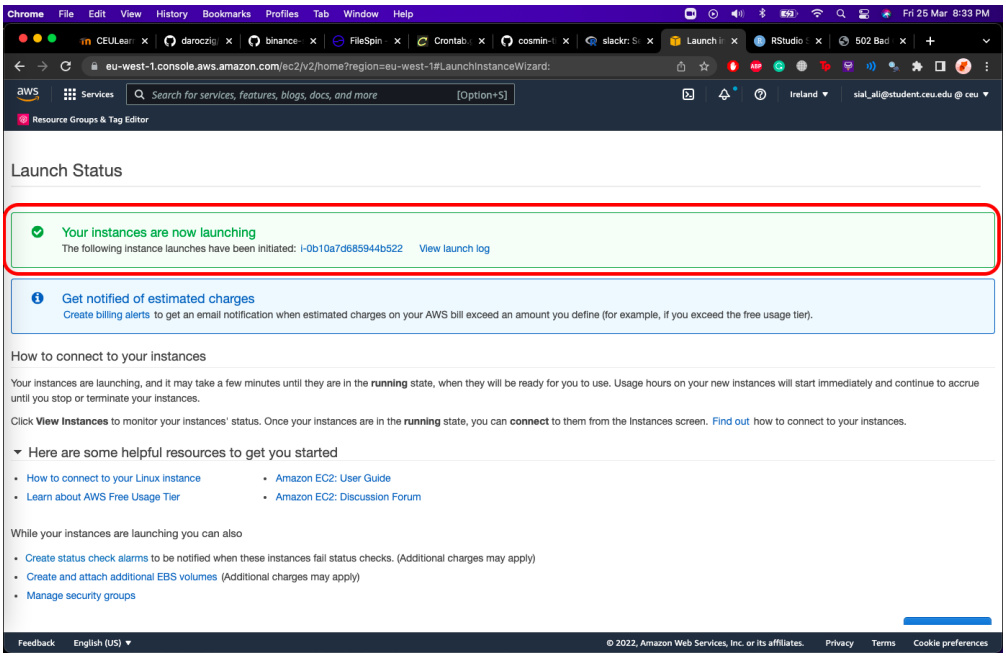
An important step is to select the security group. Thus, in the edit security group, I selected the existing security group, and I selected the de3 security group.



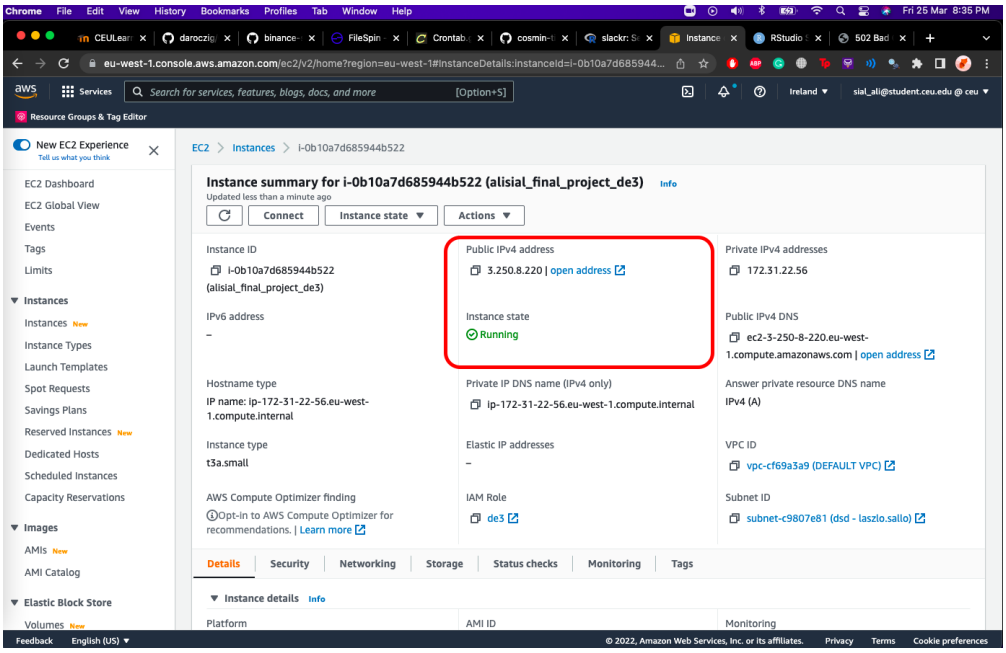
7. Selecting Key Pairs



8. Instance launched



9. Instance Running

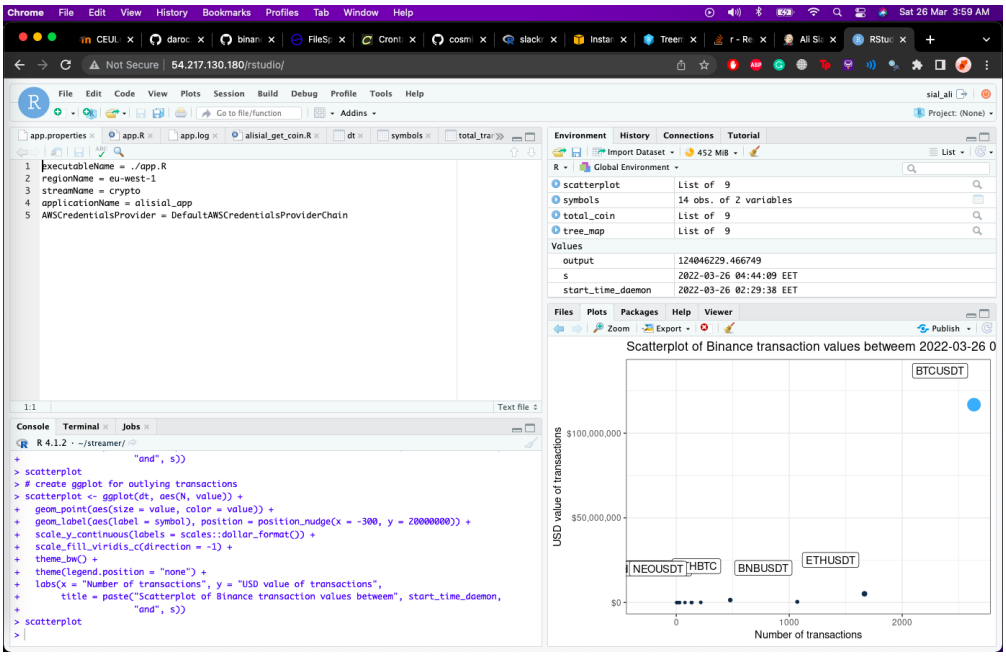


Stream Processor Daemon

First I created a new folder for the Kinesis consumer files: **streamer**. Then I created an app.properties text file in the streamer folder. The codes are as following:

```
executableName = ./app.R
regionName = eu-west-1
streamName = crypto
applicationName = alisial_app
AWSCredentialsProvider = DefaultAWSCredentialsProviderChain
```

Screenshot

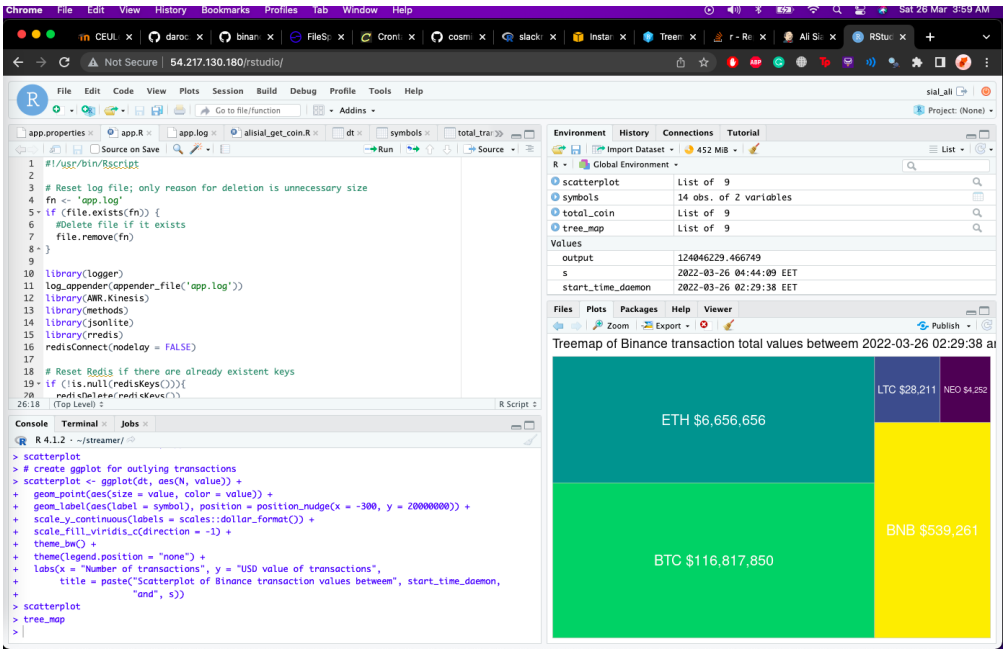


Then I created app.R file using the following codes:

```
#!/usr/bin/Rscript

# Reset log file; only reason for deletion is unnecessary size
fn <- 'app.log'
if (file.exists(fn)) {
  #Delete file if it exists
  file.remove(fn)
}

library(logger)
log_appender(appender_file('app.log'))
library(AWR.Kinesis)
library(methods)
library(jsonlite)
library(rredis)
redisConnect(nodelay = FALSE)

# Reset Redis if there are already existent keys
if (!is.null(redisKeys())){
  redisDelete(redisKeys())
}
```

```
# save the start time into Redis
redisSet("time_of_start", .POSIXct(Sys.time(), "EET"))

kinesis_consumer(

  initialize = function() {
    log_info('Hello, connected to Redis')
  },

  processRecords = function(records) {
    log_info(paste('Received', nrow(records), 'records from Kinesis'))
    for (record in records$data) {
      symbol <- fromJSON(record)$s
      log_info(paste('Found 1 transaction on', symbol))
      redisIncr(paste('alisial_symbol', symbol, sep = ':'))
    }
  },

  updater = list(
    list(1/6, function() {
      log_info('Checking overall counters')
      symbols <- redisMGet(redisKeys('alisial_symbol:*'))
      log_info(paste(sum(as.numeric(symbols)), 'records processed so far'))
    })),

  shutdown = function()
    log_info('Bye'),

  checkpointing = 1,
  logfile = 'app.log')
```

screenshot



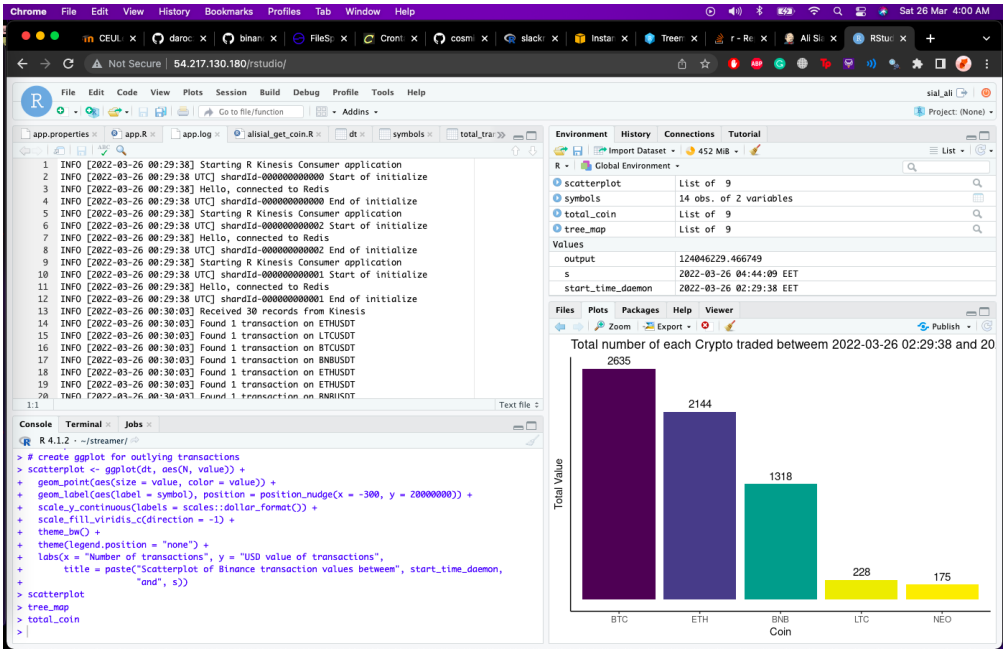
To convert the above R script into an executable I used the following code in the *Terminal*

```
cd streamer
chmod +x app.R
```

I ran the app using the Terminal using the below code

```
/usr/bin/java -cp /usr/local/lib/R/site-library/AWR/java/*:/usr/local/lib/R/site-library/AWR.Kinesis/ja
    com.amazonaws.services.kinesis.multilang.MultiLangDaemon \
    ./app.properties
```

As a result a new file called *app.log* is created in the streamer folder which contains the data.

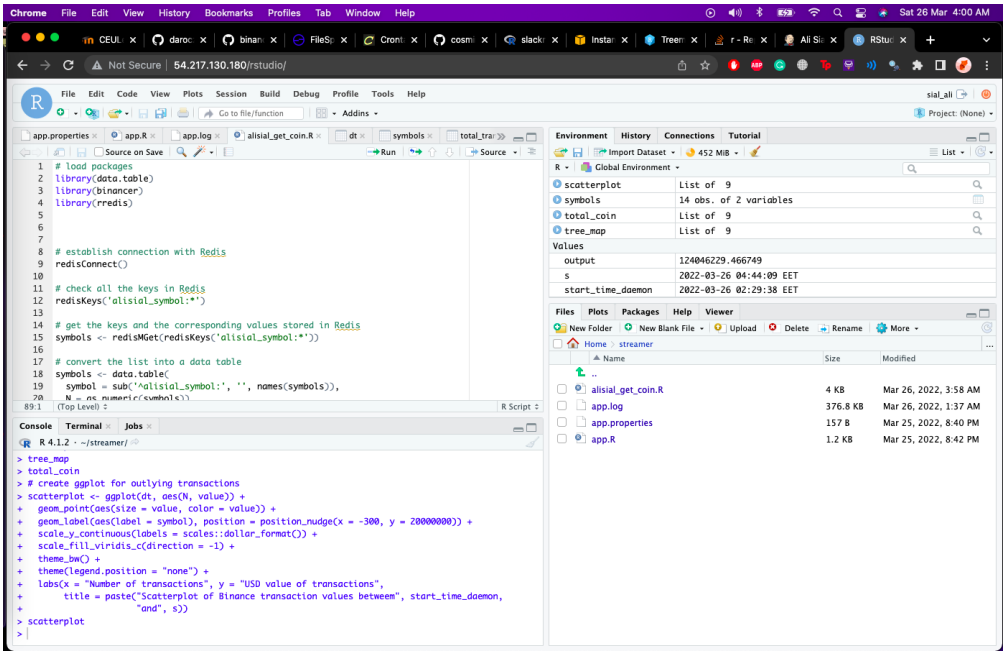


Furthermore, using the following codes, first I ran the streaming app to process the data from Binance transactions and updated the values in Redis. Then I sent the visualization to the `#bots-final-project` slack channel.

```
# load packages
library(data.table)
library(binancer)
library(rredis)



# establish connection with Redis
redisConnect()

# check all the keys in Redis
redisKeys('alisial_symbol:*')

# get the keys and the corresponding values stored in Redis
```

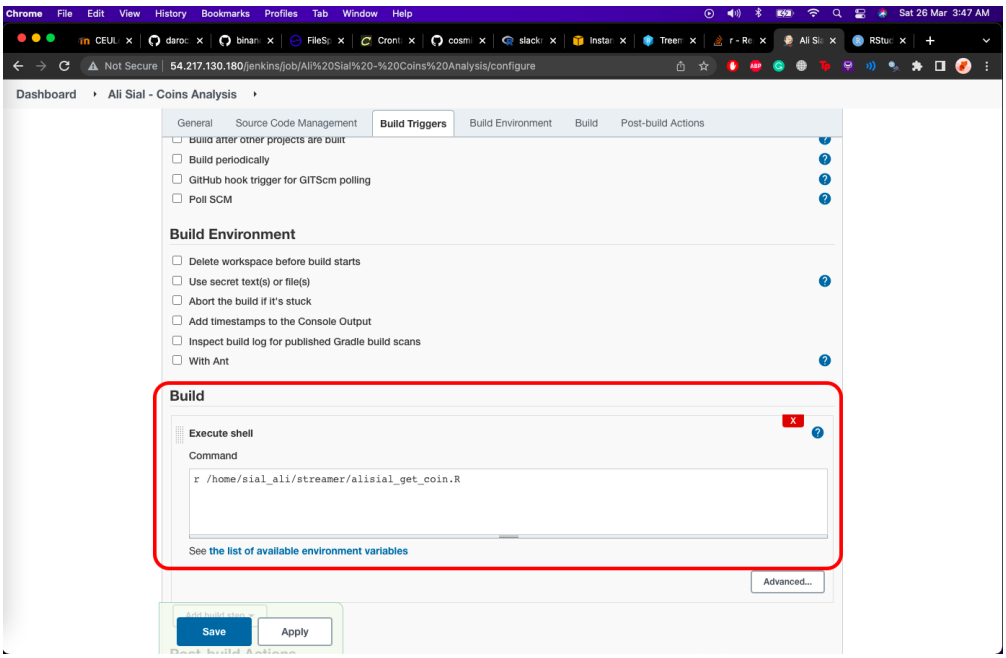
```
symbols <- redisMGet(redisKeys('alisial_symbol:*'))

# convert the list into a data table
symbols <- data.table(
  symbol = sub('^alisial_symbol:', '', names(symbols)),
  N = as.numeric(symbols))
symbols

# extract the 'from' currency
symbols[, from := substr(symbol, 1, 3)]

# group by from and sum the quantities
symbols[, .(quantity = sum(N)), by = from]

# get the real-time prices in USD
prices <- binance_coins_prices()

# merge the two tables
dt <- merge(symbols, prices, by.x = 'from', by.y = 'symbol', all.x = TRUE, all.y = FALSE)

# calculate value in USD
dt[, value := as.numeric(N) * usd]

# calculate overall USD value of transactions
output <- dt[, sum(value)]

# calculate overall USD value of transactions by coin
dt[, sum(value), by = from]

# save the time in Eastern European Time
s <- Sys.time()
s <- .POSIXct(s, "EET")

# Print the message
print(paste0('The overall value of Binance transactions at ', s,
             ' is: ',scales::dollar(output)))

# get time of stream start from Redis
start_time_daemon <- redisGet("time_of_start")

library(botor)
botor(region = 'eu-west-1')
## better way to get the Slack token
token <- ssm_get_parameter('slack')

library(slackr)
slackr_setup(username = 'Ali Sial', token = token, icon_emoji = ':exploding_head:')

# Start off by sending an informative slack message
slackr_msg(text = paste0('The overall value of Binance transactions between ',
                         start_time_daemon, ' EET',' and ', s, ' EET',
                         ' is: ',scales::dollar(output)),
           channel = '#bots-final-project')
```

```
library(ggplot2)

#  Number of each coin traded
total_coin <-
  dt[,sum(N),by = from] %>%
  ggplot(aes(x = reorder(from, -V1), y = V1)) +
  geom_col(aes(fill = V1)) +
  scale_fill_viridis_c(direction = -1) +
  labs(x = "Coin", y = "Total Value", title = paste( "Total number of each Crypto traded betweem", star
                                                      "and", s)) +
  geom_text(aes(label = round(V1)), size = 4, hjust = 0.5, vjust = -0.5, position = "stack") +
  theme_classic() +
  theme(legend.position="none",axis.text.y=element_blank(),axis.ticks.y=element_blank())

slackr_setup(username = 'Ali Sial', token = token, icon_emoji = ':exploding_head:')
ggslackr(plot = total_coin, channels = '#bots-final-project', width = 12, height = 8)


# create ggplot for outlying transactions
scatterplot <- ggplot(dt, aes(N, value)) +
  geom_point(aes(size = value, color = value)) +
  geom_label(aes(label = symbol), position = position_nudge(x = -300, y = 20000000)) +
  scale_y_continuous(labels = scales::dollar_format()) +
  scale_fill_viridis_c(direction = -1) +
  theme_bw() +
  theme(legend.position = "none") +
  labs(x = "Number of transactions", y = "USD value of transactions",
       title = paste("Scatterplot of Binance transaction values betweem", start_time_daemon,
                     "and", s))

# send to slack
ggslackr(plot = scatterplot, channels = '#bots-final-project', width = 10, height = 8)
slackr_setup(username = 'Ali Sial', token = token, icon_emoji = ':exploding_head:')

library(scales)
library(treemapify)

# Total transaction value per coin
tree_map <- dt[,.(sum(value),sum(N)),by = from] %>%
  ggplot(aes(area = V2, fill = from, lable = paste(from,V1, sep = "\n"))) +
  geom_treemap() +
  scale_fill_viridis_d(direction = -1) +
  geom_treemap_text(aes(label = paste(from, dollar(round(V1),sep = "\n"))),
                    colour = "white",
                    place = "centre",
                    size = 15) +
  theme(legend.position="none") +
  labs(title = paste("Treemap of Binance transaction total values betweem", start_time_daemon,
                     "and", s))

slackr_setup(username = 'Ali Sial', token = token, icon_emoji = ':exploding_head:')
ggslackr(plot = tree_map, channels = '#bots-final-project', width = 12, height = 8)
```

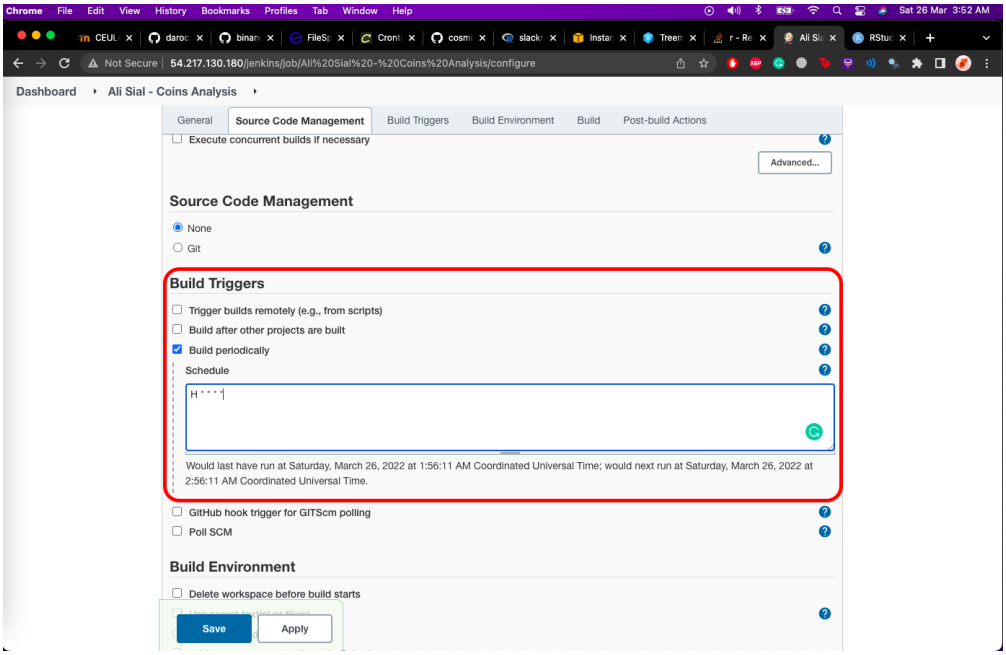


Next I created a Jenkins job that reads from Redis, and printed the overall value (in USD) of the transactions based on the coin prices reported by the Binance API and send the result to Slack. The project name in the Jenkins is *Ali Sial - Coin Analysis*. I made the following configurations.

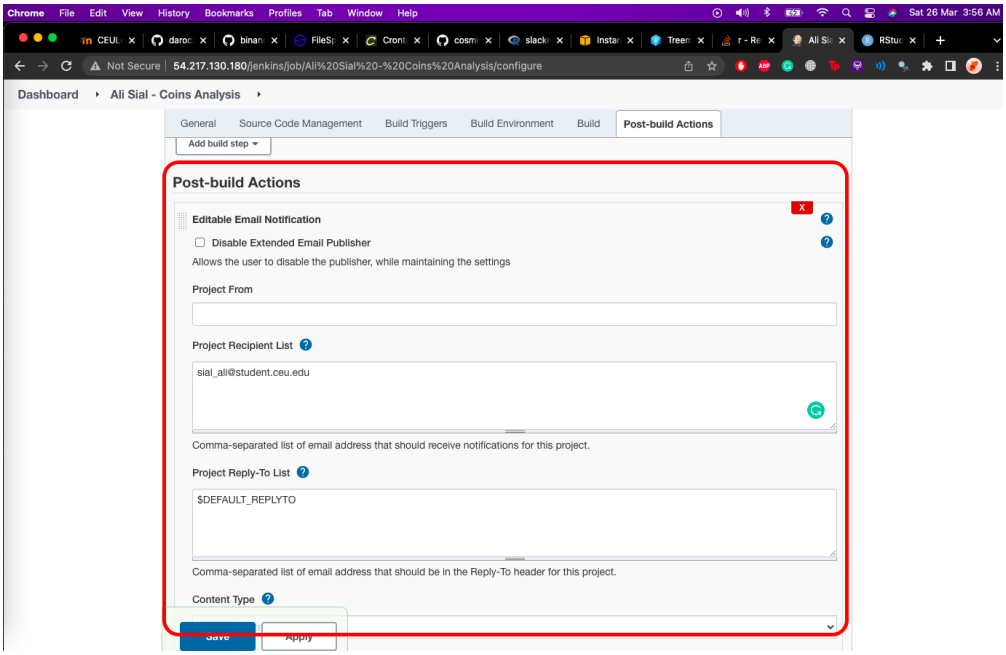
1. Executable Shell



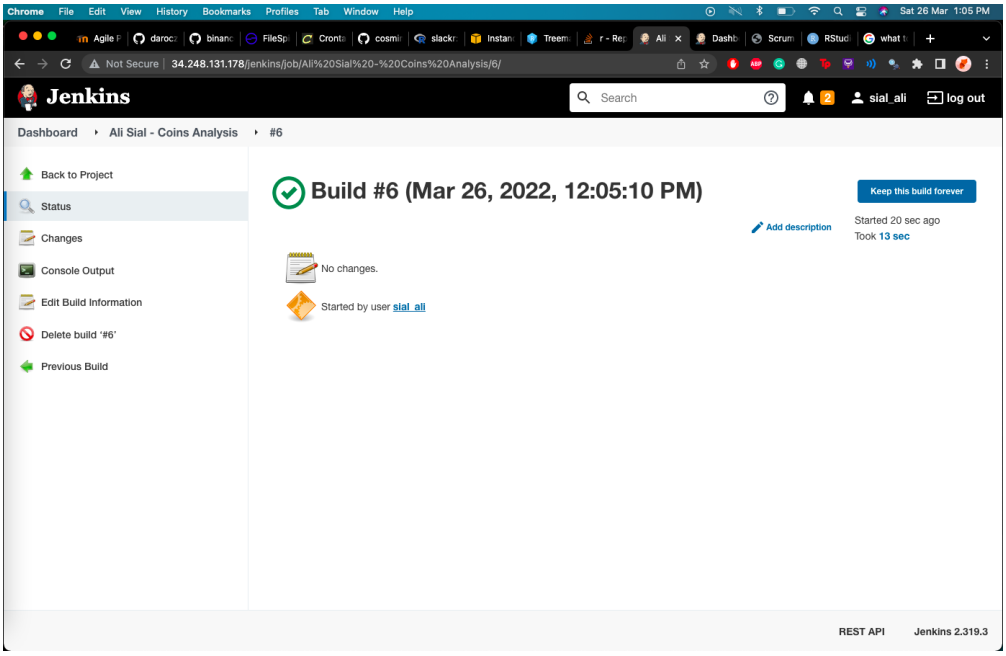
2. The report is scheduled hourly



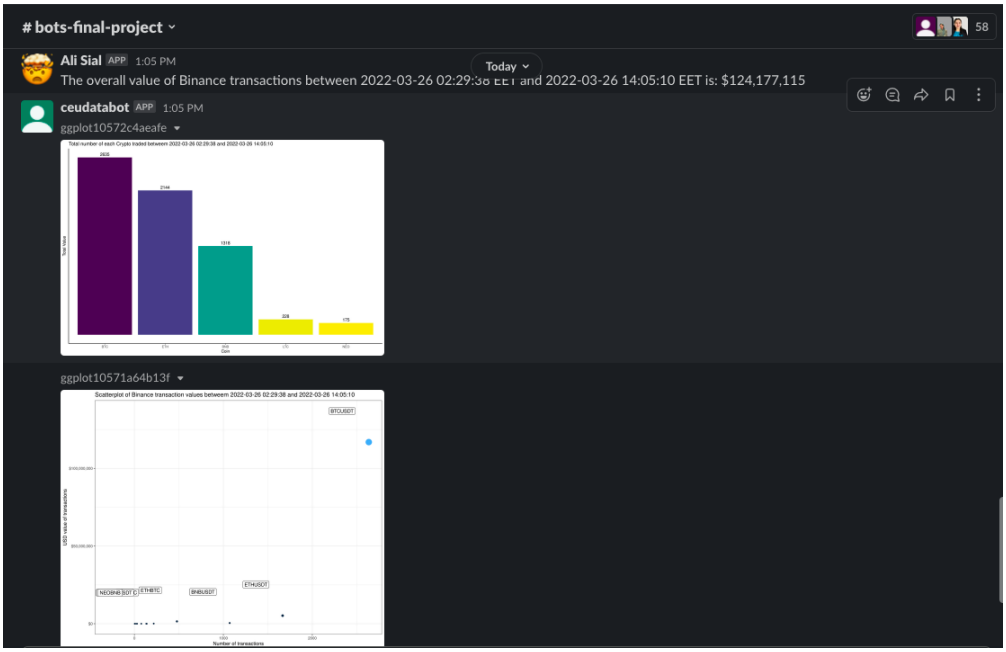
3. Editable Email Notification - In case of failure send email to the added project recipient.

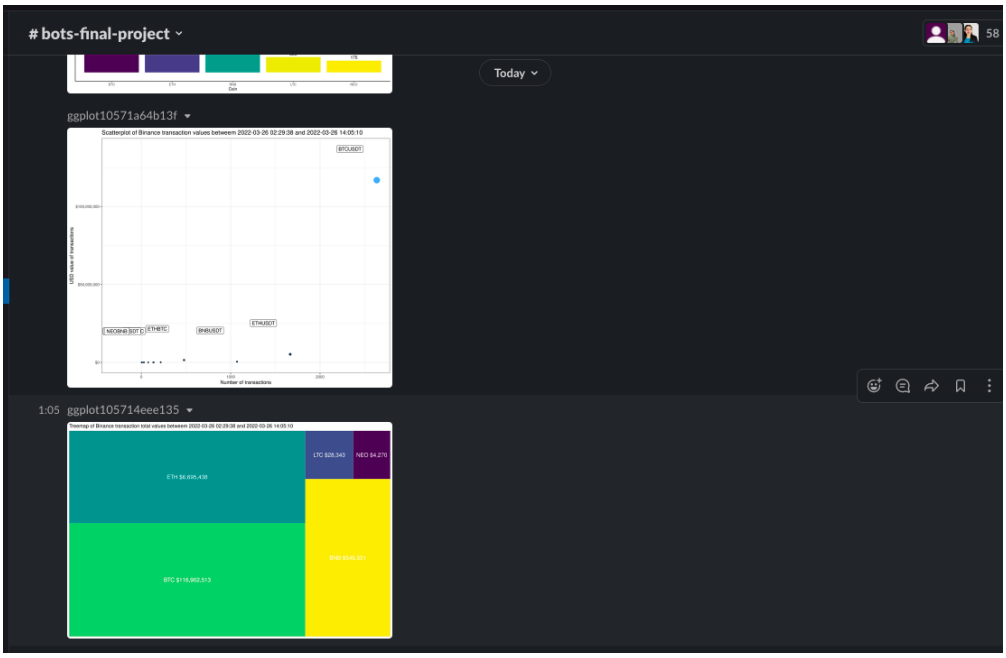


The Jenkins job has been successfully sent to the slack channel.



The result is send to the slack channel called #bots-final-project




bots-final-project
58
Today
ggplot10571a64b13f
Scatterplot of Binance transaction values between 2022-03-26 02:29:38 and 2022-03-26 14:05:10
BTCUSDT
ETHUSDT
BNBUSDT
ETHBTC
Number of transactions
1:05 ggplot105714eee135
Treemap of Binance transaction total values between 2022-03-26 02:29:38 and 2022-03-26 14:05:10
ETH $6,695,438
LTC $28,343
NEO $4,270
BTC $116,962,513