

Stream processing with R in AWS

AWR, AWR.KMS, AWR.Kinesis (R packages) used in ECS

Gergely Daroczi

@daroczig

Feb 25, 2018



About this class



Big Data Day LA 2017

@BigDataDayLA

Follow

v

Card.com's [@daroczig](#) talks to
[#BigDataDayLA](#) about Stream Processing
with R and Amazon Kinesis



2:53 PM - 9 Jul 2016

About this class



Arun Srinivasan

@arun_sriniv

Following



@daroczig wrapping up his awesome talk..



4:50 AM - 5 Jul 2017

About this class



Gergely Daróczi
@daroczig

Following

Generated a random maze in #Minecraft from #rstats and checking if @revodavid can find his way out :) @rOpenSci text by @kwbroman #runconf17





Jeff Leek
@jtleek



Question for the **#rstats** community:
do you think it is possible to create a
stand alone Shiny app that generates
\$50k+ year in revenue?

9:08 AM · 06 Dec 17



earino @ NIPS @earino · 1d

Replies to [@jtleek](#)

I think that's an odd question. The technology that is used to implement a product is not in the top N attributes that defines the revenue potential of that product.

1



7





earino @ NIPS @earino · 1d

By itself? Maybe, depends on the load. You can make 50k a year with 1 customer if you do the right thing. With a modern devopsed autoscale and load balancer infrastructure? Absolutely without a doubt.

1

1

2

✉



earino @ NIPS

@earino

Replying to [@earino](#) and [@jtleek](#)

I think the inimitable [@daroczig](#) could teach a master class on this I bet.

9:19 AM · 06 Dec 17



Gergely Daróczi
@daroczig

Following



Replying to @earino @jtleek

lol -- I'd be honored and love to do a class for fun on how to build such thing from scratch eg on AWS, but at the end of the day and all the fun, I think I'd just keep it pretty damn short: shinyproxy.io

9:34 AM - 6 Dec 2017

18 Likes



About me



Pázmány
Péter
Katolikus
Egyetem



BUDAPESTI
METROPOLITAN
FŐISKOLA



CEU CENTRAL
EUROPEAN
UNIVERSITY

3,633 contributions in the year before last

Lead R Developer



3,470 contributions in the last year

Director of Analytics



SYSTEM1



CARD.com's View of the World



Gergely Daróczi @daroczig · Apr 11

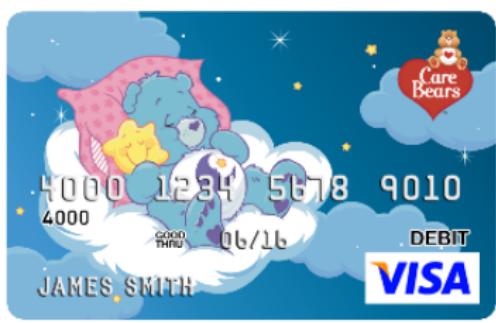
Just received my "I ❤️ R" prepaid debit card from @CARD. Will be fun to use this #rstats designed card at #user2015 :)

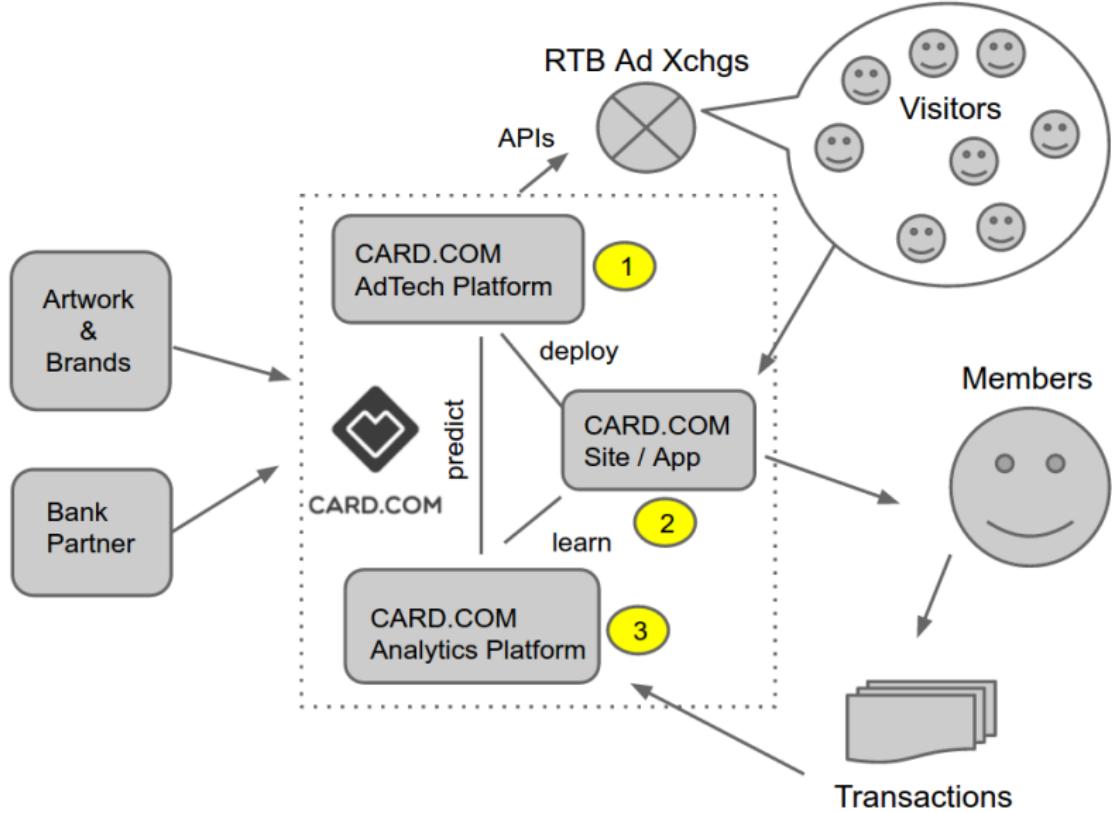


RETWEETS
10

FAVORITES
16



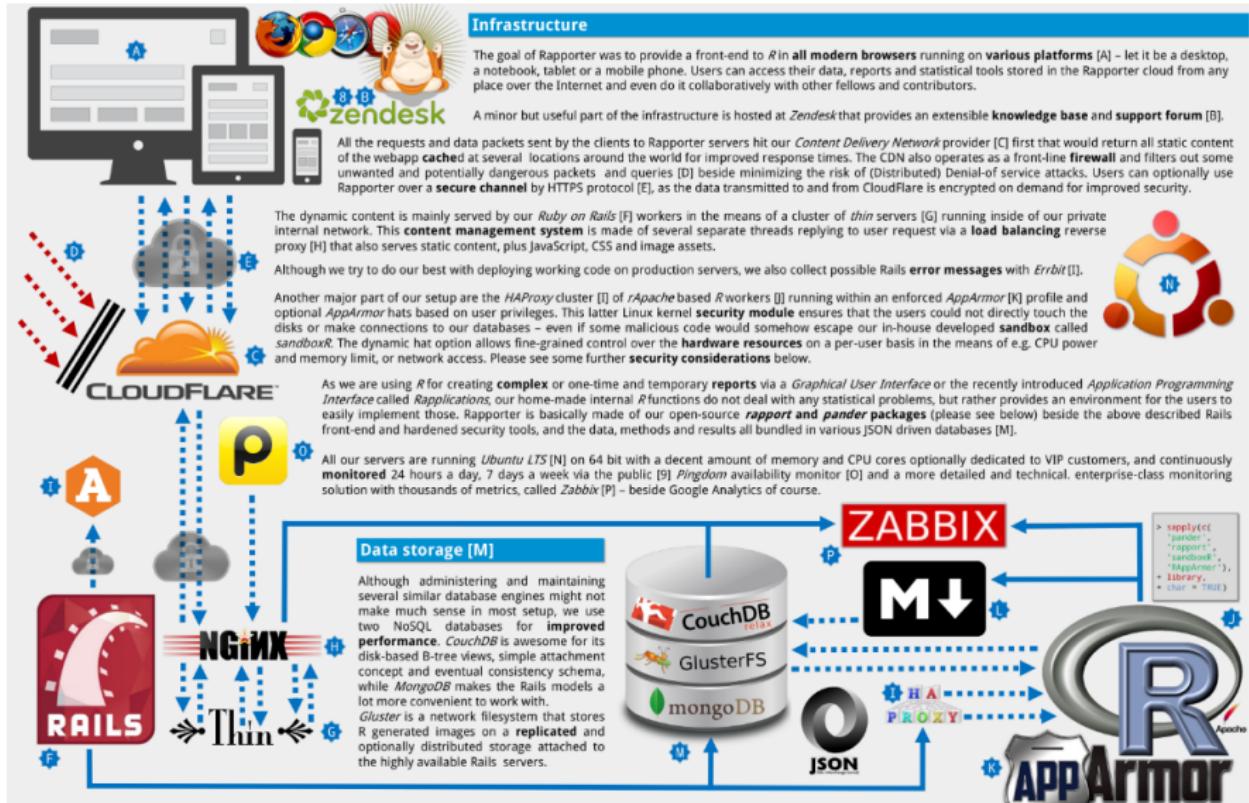




- card transaction processors
- card manufacturers
- CIP/KYC service providers
- online ad platforms
- remarketing networks
- licensing partners
- communication engines
- others



Why not Hadoop instead of MySQL?





Stream Processing ... Why R?

User Defined Java Class

Step name: ShortenDate

Classes and code fragments:

- Classes
- Code Snippets
- Input fields
- Getting fields...please wait
- Info fields
- Getting fields...please wait
- Output fields
- Getting fields...please wait

Class code

```
Processor
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;
import java.util.TimeZone;

private SimpleDateFormat df1 = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS");

private SimpleDateFormat df2 = new SimpleDateFormat("yyyy-MM-dd HH");

public boolean processRow(StepMetaInterface smi, StepDataInterface sdi) throws KettleException, ParseException
{
    Object[] r = getRow();
    if (r == null) {
        setOutputDone();
        return false;
    }

    if (first)
    {
        first = false;
    }

    // It is always safest to call createOutputRow() to ensure that your output row's Object[] is large
    // enough to handle any new fields you are creating in this step.
    r = createOutputRow(r, data.outputRowMeta.size());

    df2.setTimeZone(TimeZone.getTimeZone("America/Los_Angeles"));

    Line #: 0
}
```

Line #: 0

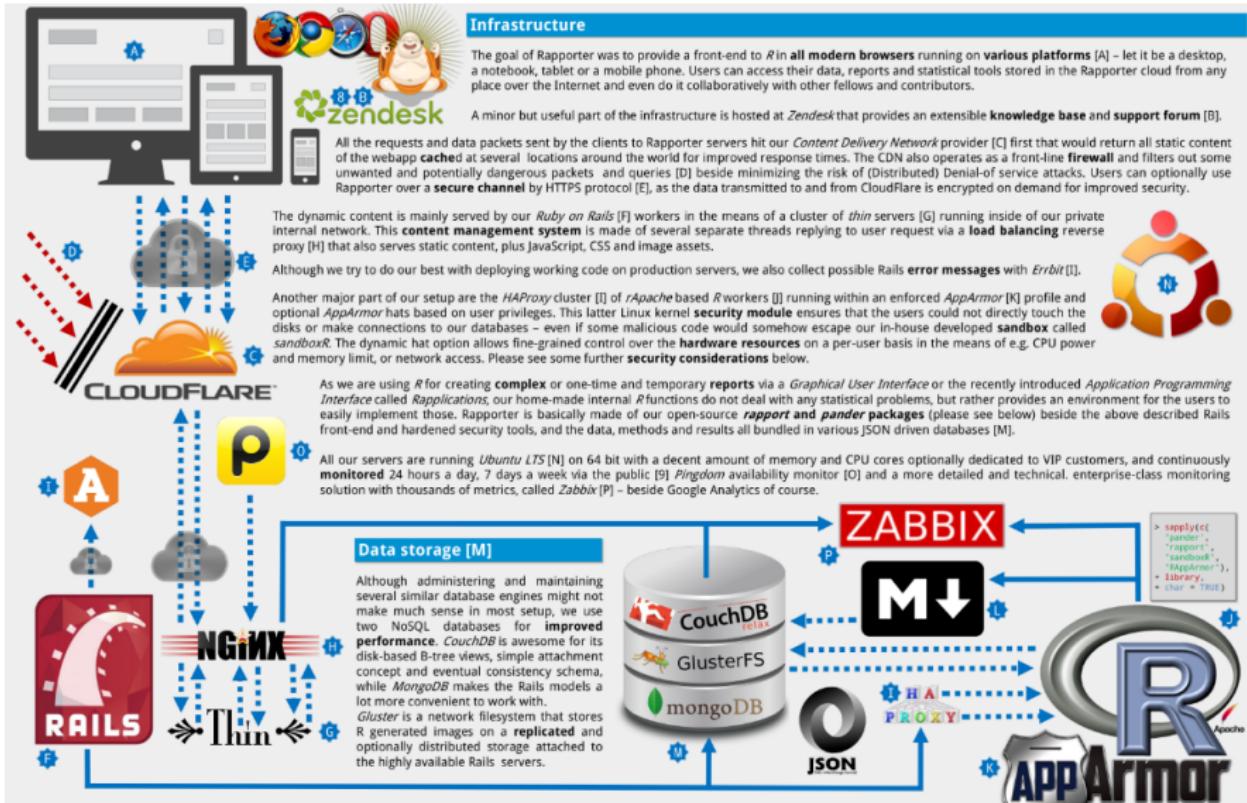
Fields Parameters Info steps Target steps

Fields Clear the result fields?

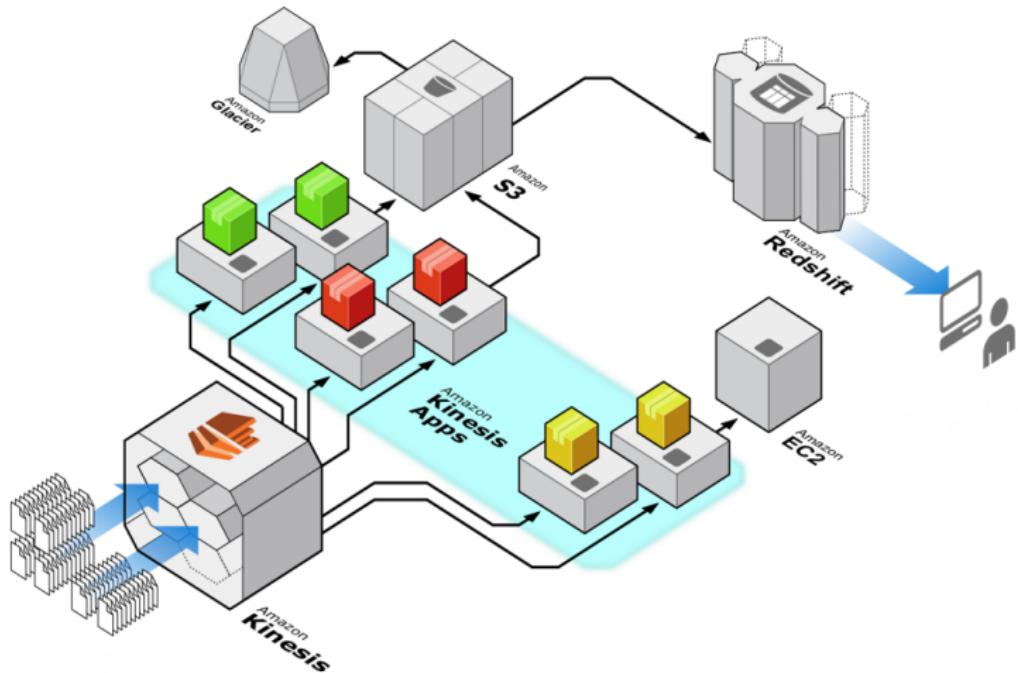
#	Fieldname	Type	Length	Precision
1	RPT_DATE_SHORT	String		

Help OK Cancel Test class

Stream Processing ... Why AWS?

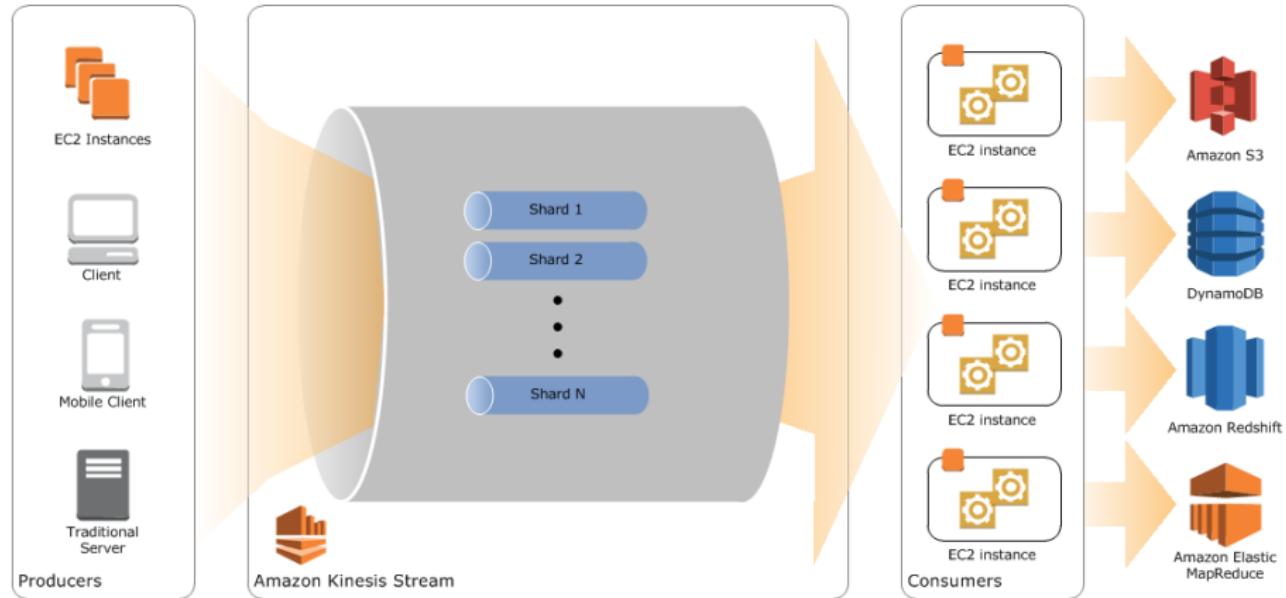


Intro to Amazon Kinesis



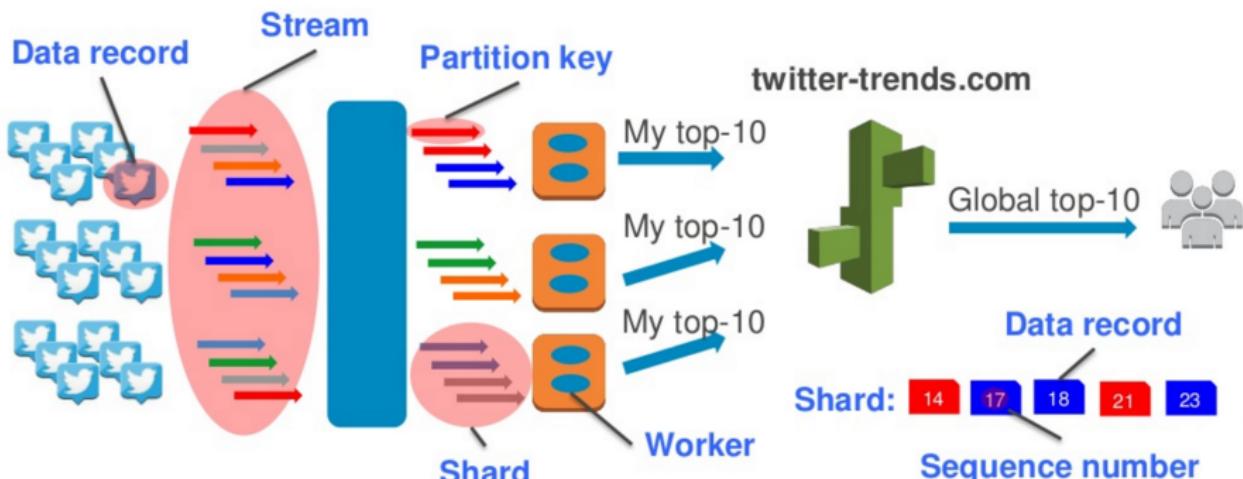
Source: Kinesis Product Details

Intro to Amazon Kinesis Streams



Source: Kinesis Developer Guide

Intro to Amazon Kinesis Shards



Source: AWS re:Invent 2013

A Very Deep Learning

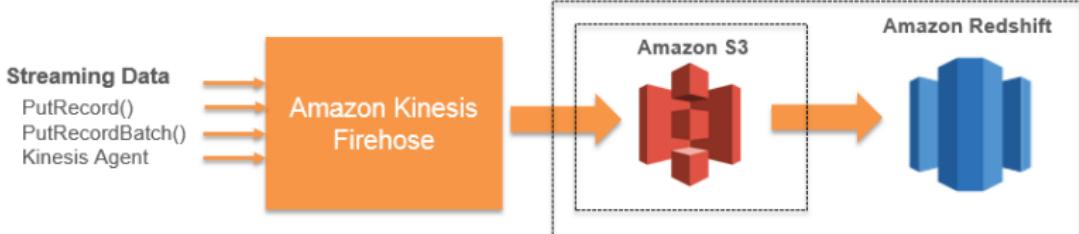


*Capture & submit
streaming data to Firehose*

*Firehose loads streaming data
continuously into S3 and Redshift*

Analyze streaming data using your favorite BI tools

A Very Deep Learning



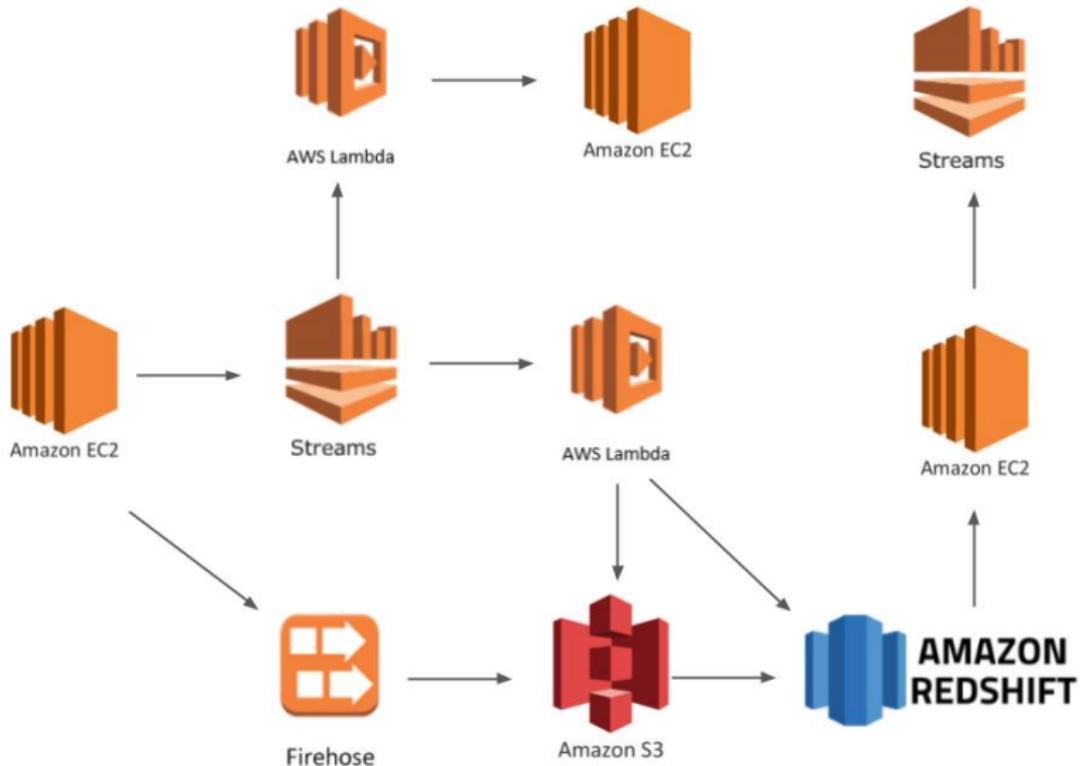
The S3 Object System

```
> x <- 3.14
> attr(x, 'class') <- 'standard'

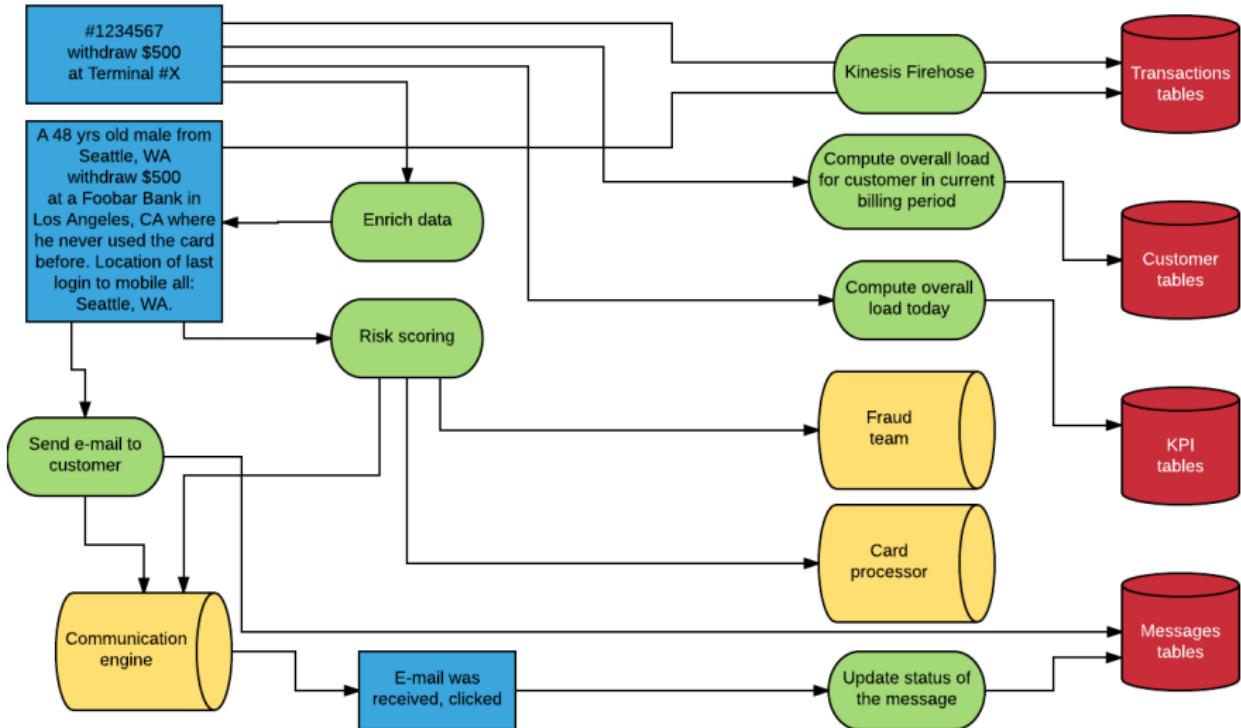
> print.standard <- function(x, ...) {
+     ## SLA
+     if (runif(1) * 100 > 99.9) {
+         Sys.sleep(20)
+     }
+     futile.logger::flog.info(x)
+ }
```

```
> while (TRUE) print(x)
INFO [2017-03-03 22:27:57] 3.14
INFO [2017-03-03 22:27:57] 3.14
INFO [2017-03-03 22:27:57] 3.14
INFO [2017-03-03 22:28:17] 3.14
INFO [2017-03-03 22:28:17] 3.14
```

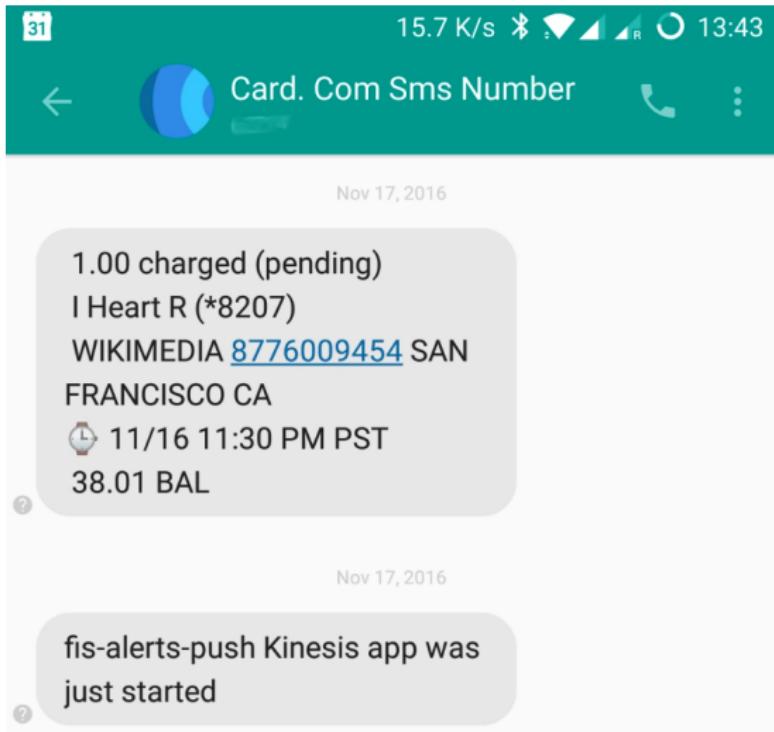
S4: Multiple Dispatch



Example use-case



Example use-case



Writing data to the stream:

- Amazon Kinesis Streams API, SDK
- Amazon Kinesis Producer Library (KPL) from Java
- flume-kinesis
- Amazon Kinesis Agent

Reading data from the stream:

- Amazon Kinesis Streams API, SDK
- Amazon Kinesis Client Library (KCL) from Java, Node.js, .NET, Python, Ruby

Managing streams:

- Amazon Kinesis Streams API (!)

Now We Need an R Client!

```
> library(rJava)
> .jinit(classpath = list.files('~/Projects/AWR/inst/java/', full.names = TRUE))

> kc <- .jnew('com.amazonaws.services.kinesis.AmazonKinesisClient')
> kc$setEndpoint('kinesis.us-west-2.amazonaws.com', 'kinesis', 'us-west-2')

> sir <- .jnew('com.amazonaws.services.kinesis.model.GetShardIteratorRequest')
> sir$setStreamName('test_kinesis')
> sir$setShardId(.jnew('java/lang/String', '0'))
> sir$setShardIteratorType('TRIM_HORIZON')
> iterator <- kc$getShardIterator(sir)$getShardIterator()

> grr <- .jnew('com.amazonaws.services.kinesis.model.GetRecordsRequest')
> grr$setShardIterator(iterator)
> kc$getRecords(grr)$getRecords()
[1] "Java-Object{[SequenceNumber: 495628941604494443321533463710843135723243616650,
ApproximateArrivalTimestamp: Tue Jun 14 09:40:19 CEST 2016,
Data: java.nio.HeapByteBuffer[pos=0 lim=6 cap=6],PartitionKey: 42]}"

> sapply(kc$getRecords(grr)$getRecords(),
+         function(x)
+             rawToChar(x$getData()$array()))
[1] "foobar"
```

Let's merge two shards:

```
> ms <- .jnew('com.amazonaws.services.kinesis.model.MergeShardsRequest')
> ms$setShardToMerge('shardId-000000000000')
> ms$setAdjacentShardToMerge('shardId-000000000001')
> ms$setStreamName('test_kinesis')
> kc$mergeShards(ms)
```

What do we have now?

```
> kc$describeStream(StreamName = 'test_kinesis')$getStreamDescription()$getShards()
[1] "Java-Object{[
{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey: 170
SequenceNumberRange: {
StartingSequenceNumber: 49562894160427143586954815717376297430913467927668719618,
EndingSequenceNumber: 49562894160438293959554081028945856364232263390243848194}},
{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey: 17014118346046923173
SequenceNumberRange: {
StartingSequenceNumber: 4956289416044944332153346340517833149186116289174700050,
EndingSequenceNumber: 49562894160460594704752611652087392082504911751749828626}},
{ShardId: shardId-000000000002,
ParentShardId: shardId-000000000000,
AdjacentParentShardId: shardId-000000000001,
HashKeyRange: {StartingHashKey: 0,EndingHashKey: 3402823669209384634633746074317682
SequenceNumberRange: {StartingSequenceNumber: 495620049914976730997049241472701952
Gergely Daroczi (@daroczig) Stream processing with R in AWS Data Infra in Prod @ CEU BA 22 / 70
```

- An *easy-to-use* programming model for processing data

```
java -cp amazon-kinesis-client-1.7.3.jar \
com.amazonaws.services.kinesis.multilang.MultiLangDaemon \
app.properties
```

- *Scalable* and *fault-tolerant* processing (checkpointing via DynamoDB)
- Logging and metrics in CloudWatch
- The **MultiLangDaemon** spawns processes written in any language, communication happens via JSON messages sent over stdin/stdout
- Only a few events/methods to care about in the consumer application:
 - ① initialize
 - ② processRecords
 - ③ checkpoint
 - ④ shutdown

① initialize:

- Perform initialization steps
- Write “status” message to indicate you are done
- Begin reading line from STDIN to receive next action

② processRecords:

- Perform processing tasks (you may write a checkpoint message at any time)
- Write “status” message to STDOUT to indicate you are done.
- Begin reading line from STDIN to receive next action

③ shutdown:

- Perform shutdown tasks (you may write a checkpoint message at any time)
- Write “status” message to STDOUT to indicate you are done.
- Begin reading line from STDIN to receive next action

④ checkpoint:

- Decide whether to checkpoint again based on whether there is an error or not.

Again . . . Why R?

User Defined Java Class

Step name: **Shorten Date**

Classes and code fragments:

- Clases
- Code Snippets
- Input fields
- Getting fields...please wait
- Info fields
- Getting fields...please wait
- Output fields
- Getting fields...please wait

Class code

```

Processor
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;
import java.util.TimeZone;

private SimpleDateFormat df1 = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS");

private SimpleDateFormat df2 = new SimpleDateFormat("yyyy-MM-dd HH");

public boolean processRow(StepMetaInterface smi, StepDataInterface sdi) throws KettleException, ParseException
{
    Object[] r = getRow();
    if (r == null) {
        setOutputDone();
        return false;
    }

    if (first)
    {
        first = false;
    }

    // It is always safest to call createOutputRow() to ensure that your output row's Object[] is large
    // enough to handle any new fields you are creating in this step.
    r = createOutputRow(r, data.outputRowMeta.size());

    df2.setTimeZone(TimeZone.getTimeZone("America/Los_Angeles"));
}
  
```

Line #: 0

Fields Parameters Info steps Target steps

#	Fieldname	Type	Length	Precision
1	RPT_DATE_SHORT	String		

Clear the result fields?

 Help OK Cancel Test class

R Script Interacting with KCL

```
#!/usr/bin/r -i

while (TRUE) {

    ## read and parse JSON messages
    line <- fromJSON(readLines(n = 1))

    ## nothing to do unless we receive records to process
    if (line$action == 'processRecords') {

        ## process each record
        lapply(line$records, function(r) {

            business_logic(fromJSON(rawToChar(base64_dec(r$data))))
            cat(toJSON(list(action = 'checkpoint', checkpoint = r$sequenceNumber)))

        })
    }

    ## return response in JSON
    cat(toJSON(list(action = 'status', responseFor = line$action)))
}

}
```

R Script Interacting with KCL

```
#!/usr/bin/r -i

while (TRUE) {

  ## read and parse JSON messages
  line <- fromJSON(readLines(n = 1))

  ## nothing to do unless we receive records to process
  if (line$action == 'processRecords') {

    ## process each record
    lapply(line$records, function(r) {

      business_logic(fromJSON(rawToChar(base64_dec(r$data))))
      cat(toJSON(list(action = 'checkpoint', checkpoint = r$sequenceNumber)))

    })
  }

  ## return response in JSON
  cat(toJSON(list(action = 'status', responseFor = line$action)))
}

}
```

```
> install.packages('AWR.Kinesis')
also installing the dependency 'AWR'

trying URL 'https://cloud.r-project.org/src/contrib/AWR_1.11.89.tar.gz'
Content type 'application/x-gzip' length 3125 bytes

trying URL 'https://cloud.r-project.org/src/contrib/AWR.Kinesis_1.7.3.tar.gz'
Content type 'application/x-gzip' length 3091459 bytes (2.9 MB)

* installing *source* package 'AWR' ...
** testing if installed package can be loaded
trying URL 'https://gitlab.com/cardcorp/AWR/repository/archive.zip?ref=1.11.89'
downloaded 58.9 MB
* DONE (AWR)

* installing *source* package 'AWR.Kinesis' ...
* DONE (AWR.Kinesis)
```

Business logic coded in R (demo_app.R):

```
library(AWR.Kinesis)
kinesis_consumer(processRecords = function(records) {
  flog.info(jsonlite::toJSON(records))
})
```

Business logic coded in R (demo_app.R):

```
library(AWR.Kinesis)
kinesis_consumer(processRecords = function(records) {
  flog.info(jsonlite:: toJSON(records))
})
```

Note

This is not something you should run in RStudio.

Business logic coded in R (demo_app.R):

```
library(AWR.Kinesis)
kinesis_consumer(processRecords = function(records) {
  flog.info(jsonlite::toJSON(records))
})
```

Config file for the MultiLangDaemon (demo_app.properties):

```
executableName = ./demo_app.R
streamName = demo_stream
applicationName = demo_app
```

Start the MultiLangDaemon:

```
/usr/bin/java -cp AWR/java/*:AWR.Kinesis/java/*:./ \
  com.amazonaws.services.kinesis.multilang.MultiLangDaemon \
  ./demo_app.properties
```

'Advanced' AWR.Kinesis features

```
library(futile.logger)
library(AWR.Kinesis)

kinesis_consumer(
  initialize      = function()
    flog.info('Hello'),
  processRecords = function(records)
    flog.info(paste('Received', nrow(records), 'records from Kinesis')),
  shutdown       = function()
    flog.info('Bye'),
  updater        = list(
    list(1, function()
      flog.info('Updating some data every minute')),
    list(1/60*10, function()
      flog.info(paste(
        'This is a high frequency updater call',
        'running every 10 seconds')))),
  checkpointing = 1,
  logfile = '/logs/logger.log')
```

Note

In theory you could, but this is not something you should run in RStudio.

- ① Create a Kinesis Stream
- ② Create an IAM user with DynamoDB and Kinesis permissions
- ③ Write data to the Stream
- ④ Run the MultiLangDaemon referencing the properties file

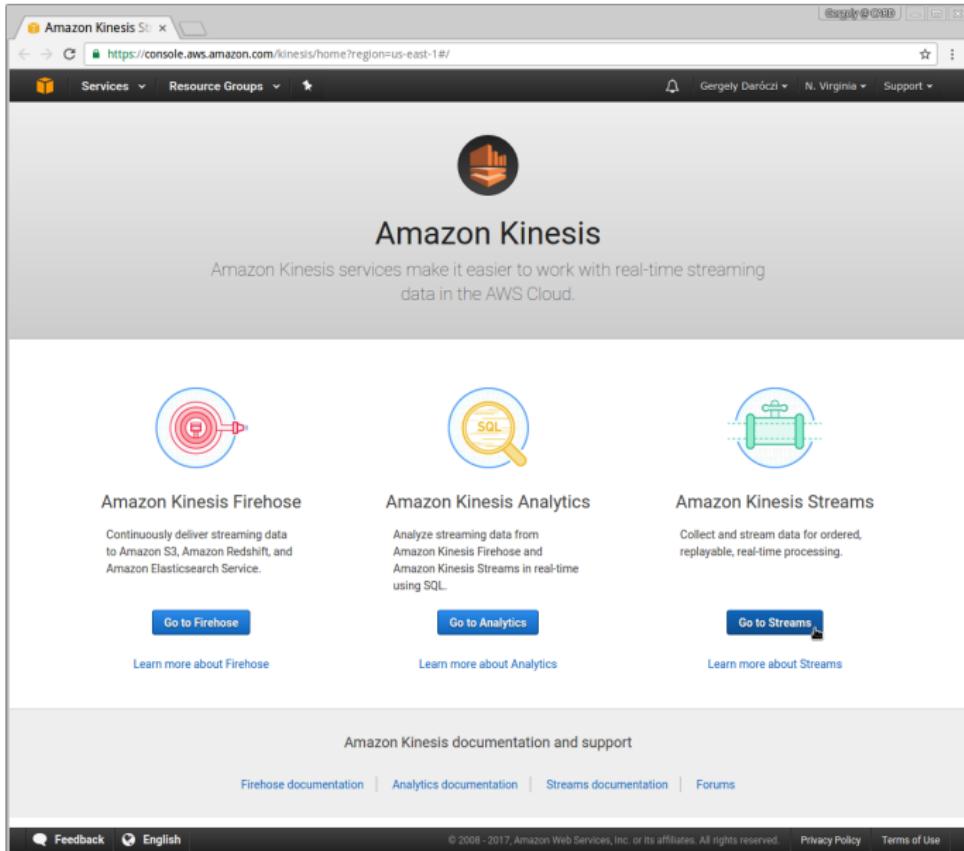
Note

In theory you could, but this is not something you should run in RStudio.

- ① Create a Kinesis Stream
- ② Create an IAM user with DynamoDB and Kinesis permissions
- ③ Write data to the Stream
- ④ Run the MultiLangDaemon referencing the properties file



Create a Kinesis Stream



The screenshot shows the Amazon Kinesis service page in the AWS Management Console. The URL in the browser is <https://console.aws.amazon.com/kinesis/home?region=us-east-1#/>. The page features a central logo with a bar chart icon and the text "Amazon Kinesis". Below it, a sub-headline reads: "Amazon Kinesis services make it easier to work with real-time streaming data in the AWS Cloud." Three main service icons are displayed: "Amazon Kinesis Firehose" (represented by a red fire hydrant icon), "Amazon Kinesis Analytics" (represented by a magnifying glass over a SQL icon), and "Amazon Kinesis Streams" (represented by a green valve icon). Each service has a brief description, a "Go to [Service]" button, and a "Learn more about [Service]" link. At the bottom, there's a footer with links to documentation and forums, and a navigation bar with "Feedback", "English", and other language options.

Amazon Kinesis

Amazon Kinesis services make it easier to work with real-time streaming data in the AWS Cloud.

 Amazon Kinesis Firehose

Continuously deliver streaming data to Amazon S3, Amazon Redshift, and Amazon Elasticsearch Service.

[Go to Firehose](#)

[Learn more about Firehose](#)

 Amazon Kinesis Analytics

Analyze streaming data from Amazon Kinesis Firehose and Amazon Kinesis Streams in real-time using SQL.

[Go to Analytics](#)

[Learn more about Analytics](#)

 Amazon Kinesis Streams

Collect and stream data for ordered, replayable, real-time processing.

[Go to Streams](#)

[Learn more about Streams](#)

Amazon Kinesis documentation and support

[Firehose documentation](#) | [Analytics documentation](#) | [Streams documentation](#) | [Forums](#)

Feedback English © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Create a Kinesis Stream

Amazon Kinesis Stream x https://console.aws.amazon.com/kinesis/home?region=us-east-1#streams/create

Services Resource Groups Gergely Darócz N. Virginia Support

Create stream

Stream name* test-AWR

Streams Firehose Analytics

Shards

A shard is a unit of throughput capacity. Each shard ingests up to 1MB/sec and 1000 records/sec, and emits up to 2MB/sec. To accommodate for higher or lower throughput, the number of shards can be modified after the stream is created using the API. [Learn more](#)

Producers → Stream → Consumers

Estimate the number of shards you'll need

Number of shards* 1 You can provision up to 48 more shards before hitting your account limit of 50. [Learn more or request a shard limit increase for this account](#)

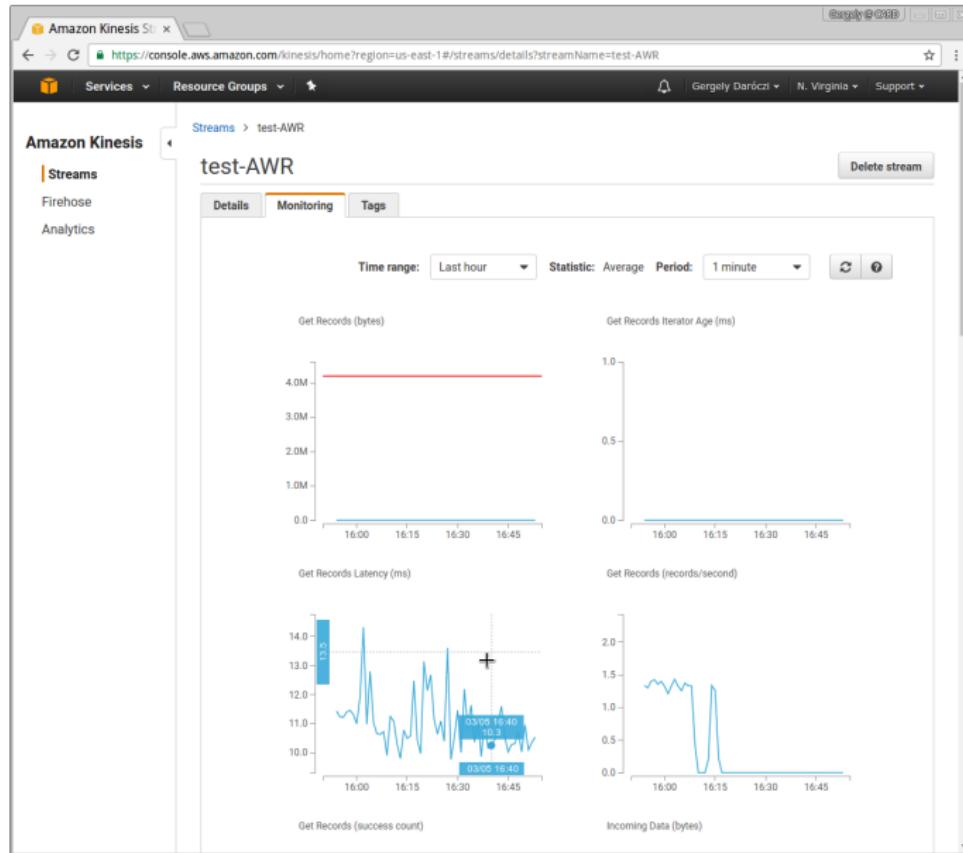
Total stream capacity Values are calculated based on the number of shards entered above.

Write 1 MB per second
1000 Records per second

Read 2 MB per second

* Required Cancel Create stream

Check the Kinesis Stream



Create an IAM user

The screenshot shows the AWS IAM Management console interface. The left sidebar navigation bar includes links for Dashboard, Groups, Users (which is selected), Roles, Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main content area displays the details for a user named "AWR". The user's ARN is listed as "arn:aws:iam::[REDACTED]user/AWR", with a path of "/". The creation time is noted as "2017-02-13 16:04 PST". Below these details, there are tabs for "Permissions", "Groups (0)", "Security credentials", and "Access Advisor". The "Permissions" tab is active, showing a list of attached policies: "AmazonEC2ContainerRegistryFullAccess - AWS Managed policy", "CloudWatchFullAccess - AWS Managed policy", "cloudwatch-putmetrics - Managed policy", "AmazonDynamoDBFullAccess - AWS Managed policy", and "AmazonKinesisFullAccess - AWS Managed policy". A button labeled "Add inline policy" is located at the bottom right of this section. The footer of the page includes links for Feedback, English, and other AWS services.

Write Data to the Stream from R

```
library(rJava)
.jcall("java/lang/System", "S", "setProperty", "aws.profile", "personal")

library(AWR.Kinesis)
library(jsonlite)
library(futile.logger)
library(nycflights13)
while (TRUE) {

    ## pick a ~car~flight
    flight <- flights[sample(1:nrow(flights), 1), ]

    ## prr <- .jnew('com.amazonaws.services.kinesis.model.PutRecordRequest')
    ## prr$setStreamName('test1')
    ## prr$setData(J('java.nio.ByteBuffer')$wrap(.jbyte(charToRaw(toJSON(car)))))
    ## prr$setPartitionKey(rownames(car))
    ## kc$putRecord(prr)

    res <- kinesis_put_record(stream = 'test-AWR', region = 'us-east-1',
                               data = toJSON(flight), partitionKey = flight$dest)
    flog.info(paste('Pushed a new flight to Kinesis:', res$sequenceNumber))

}
```

Write Data to the Stream from R



```
*Minibuf-1*
```

File Edit Options Buffers Tools Minibuf Help

```
> library(rJava)
> .jcall("java/lang/System", "", "setProperty", "aws.profile", "personal")
> library(AMR.Kinesis); library(jsonlite); library(futile.logger); library(ncflights13)
> while (TRUE) {
+   flight <- flights[sample(nrow(flights), 1), ]
+   res <- kinesis_put_record("kinesis-test", "us-east-1", data = toJSON(flight),
+                             partitionKey = flight$dest)
+   log.info(paste("Pushed a new flight to Kinesis: ", res$sequenceNumber))
+ }
```

INFO [2017-03-06 21:47:16] Pushed a new flight to Kinesis: 49571082550613725758991014186133813750336290428328869938

INFO [2017-03-06 21:47:17] Pushed a new flight to Kinesis: 495710825506137257589910141863804346175376484888406578

INFO [2017-03-06 21:47:17] Pushed a new flight to Kinesis: 49571082550613725758991014186610130526454391882580018

INFO [2017-03-06 21:47:18] Pushed a new flight to Kinesis: 49571082550613725758991014187394723380194348794986364978

INFO [2017-03-06 21:47:18] Pushed a new flight to Kinesis: 49571082550613725758991014187394723380194348794986364978

INFO [2017-03-06 21:47:19] Pushed a new flight to Kinesis: 49571082550613725758991014187636508544117274698647076914

INFO [2017-03-06 21:47:19] Pushed a new flight to Kinesis: 49571082550613725758991014187857741969106751837618307122

INFO [2017-03-06 21:47:20] Pushed a new flight to Kinesis: 495710825506137257589910141880753486166373835157784895538

INFO [2017-03-06 21:47:20] Pushed a new flight to Kinesis: 49571082550613725758991014188250642860481506388117291058

INFO [2017-03-06 21:47:21] Pushed a new flight to Kinesis: 495710825506137257589910141883057262005201203143980294194

INFO [2017-03-06 21:47:21] Pushed a new flight to Kinesis: 495710825506137257589910141883057262005201203143980294194

INFO [2017-03-06 21:47:21] Pushed a new flight to Kinesis: 495710825506137257589910141890158929427567624425777202

INFO [2017-03-06 21:47:22] Pushed a new flight to Kinesis: 49571082550613725758991014189266140548957795032309432370

INFO [2017-03-06 21:47:22] Pushed a new flight to Kinesis: 49571082550613725758991014189516388193618023271473610802

INFO [2017-03-06 21:47:23] Pushed a new flight to Kinesis: 49571082550613725758991014189794441132129388119094987454

INFO [2017-03-06 21:47:23] Pushed a new flight to Kinesis: 495710825506137257589910141900156745711886528066214962

INFO [2017-03-06 21:47:24] Pushed a new flight to Kinesis: 4957108255061372575899101419034268452841190521536882482

INFO [2017-03-06 21:47:24] Pushed a new flight to Kinesis: 49571082550613725758991014190521536882482

INFO [2017-03-06 21:47:25] Pushed a new flight to Kinesis: 49571082550613725758991014190772462120197623190151757874

INFO [2017-03-06 21:47:25] Pushed a new flight to Kinesis: 4957108255061372575899101419096589025135963858104746034

INFO [2017-03-06 21:47:25] Pushed a new flight to Kinesis: 495710825506137257589910141911170198714852754956514

INFO [2017-03-06 21:47:26] Pushed a new flight to Kinesis: 495710825506137257589910141913974767689388654219437602

INFO [2017-03-06 21:47:26] Pushed a new flight to Kinesis: 49571082550613725758991014191629590526304395412457390130

INFO [2017-03-06 21:47:27] Pushed a new flight to Kinesis: 4957108255061372575899101419191080131645089989922962

INFO [2017-03-06 21:47:27] Pushed a new flight to Kinesis: 495710825506137257589910141921796001200868951010

INFO [2017-03-06 21:47:28] Pushed a new flight to Kinesis: 49571082550613725758991014192275036789024283116106

INFO [2017-03-06 21:47:28] Pushed a new flight to Kinesis: 4957108255061372575899101419259552226517648419176101490

INFO [2017-03-06 21:47:28] Pushed a new flight to Kinesis: 49571082550613725758991014192772042525840220119993679922

INFO [2017-03-06 21:47:29] Pushed a new flight to Kinesis: 49571082550613725758991014192981169592633550967217848370

INFO [2017-03-06 21:47:29] Pushed a new flight to Kinesis: 4957108255061372575899101419320629795081871993713197106

INFO [2017-03-06 21:47:30] Pushed a new flight to Kinesis: 4957108255061372575899101419341396503605558280482136114

INFO [2017-03-06 21:47:30] Pushed a new flight to Kinesis: 4957108255061372575899101419360981101883315280678453662

INFO [2017-03-06 21:47:31] Pushed a new flight to Kinesis: 495710825506137257589910141936110883315280678453662

INFO [2017-03-06 21:47:31] Pushed a new flight to Kinesis: 4957108255061372575899101419404315388074810491232350546

INFO [2017-03-06 21:47:31] Pushed a new flight to Kinesis: 495710825506137257589910141942764666470351564544173106

INFO [2017-03-06 21:47:32] Pushed a new flight to Kinesis: 49571082550613725758991014194544310677395266696271364146

INFO [2017-03-06 21:47:32] Pushed a new flight to Kinesis: 49571082550613725758991014194752459183689829143208264618

INFO [2017-03-06 21:47:33] Pushed a new flight to Kinesis: 49571082550613725758991014194932532613600048313933874

INFO [2017-03-06 21:47:34] Pushed a new flight to Kinesis: 4957108255061372575899101419547810335957375170138714930

INFO [2017-03-06 21:47:34] Pushed a new flight to Kinesis: 495710825506137257589910141955067328701041496216259552208896020076

INFO [2017-03-06 21:47:35] Pushed a new flight to Kinesis: 4957108255061372575899101419640605434960179341477305394

INFO [2017-03-06 21:47:36] Pushed a new flight to Kinesis: 4957108255061372575899101419681104658917269667372351090

INFO [2017-03-06 21:47:36] Pushed a new flight to Kinesis: 4957108255061372575899101419681104658917269667372351090

INFO [2017-03-06 21:47:36] Pushed a new flight to Kinesis: 495710825506137257589910141970068925719502660002571602

INFO [2017-03-06 21:47:37] Pushed a new flight to Kinesis: 49571082550613725758991014197254722364971265580840517682

INFO [2017-03-06 21:47:37] Pushed a new flight to Kinesis: 49571082550613725758991014197395399607428572507101362

INFO [2017-03-06 21:47:38] Pushed a new flight to Kinesis: 495710825506137257589910141974044667741726187619

INFO [2017-03-06 21:47:38] Pushed a new flight to Kinesis: 4957108255061372575899101419814570694207247488757399602

INFO [2017-03-06 21:47:39] Pushed a new flight to Kinesis: 49571082550613725758991014198412873301620805363674644948

INFO [2017-03-06 21:47:40] Pushed a new flight to Kinesis: 49571082550613725758991014198640151354249630889931702322

INFO [2017-03-06 21:47:41] Pushed a new flight to Kinesis: 495710825506137257589910141993534175878226217172782898

INFO [2017-03-06 21:47:41] Pushed a new flight to Kinesis: 49571082550613725758991014199624197145939106862006322

130 U: R-kinesis-putrecord IESS [R]: run 109137 : 0 99%

This Kinesis app is being run

```
library(futile.logger)
library(AWR.Kinesis)

kinesis_consumer(
    initialize      = function()
        flog.info('Hello'),
    processRecords = function(records)
        flog.info(paste('Received', nrow(records), 'records from Kinesis')),
    shutdown       = function()
        flog.info('Bye'),
    updater        = list(
        list(1, function()
            flog.info('Updating some data every minute')),
        list(1/60*10, function()
            flog.info(paste(
                'This is a high frequency updater call',
                'running every 10 seconds')))),
    checkpointing = 1,
    logfile = '/logs/logger.log')
```

Running the MultiLangDaemon locally

```
Terminix: Default
1 / 1 + ⌂ ⌂

1: daroczig@gergely-CARD: ~/Projects/card-rocker/r-kinesis-example/files ~
~/Projects/card-rocker/r-kinesis-example/files | master ? export AWS_PROFILE=personal
~/Projects/card-rocker/r-kinesis-example/files | master ? /usr/bin/java -cp \
"/usr/local/lib/R/site-library/AWR/java":~/usr/local/lib/R/site-library/AWR.Kinesis/java/*:." \
com.amazonaws.services.kinesis.multilang.MultilangDaemon ./app.properties
Mar 05, 2017 5:32:33 PM com.amazonaws.services.kinesis.clientlibrary.config.KinesisClientLibConfigurator getConfiguration
INFO: Value of workerId is not provided in the properties. WorkerId is automatically assigned as:
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.clientlibrary.config.KinesisClientLibConfigurator withProperty
INFO: Successfully set property regionName with value us-east-1
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multilang.MultilangDaemonConfig buildExecutorService
INFO: Using a cached thread pool.
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multilang.MultilangDaemonConfig <init>
INFO: Running AWR-demo-app to process stream test-AWR with executable /app/app.R
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multilang.MultilangDaemonConfig prepare
INFO: Using workerId:
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multilang.MultilangDaemonConfig prepare
INFO: Using credentials with access key id:
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multilang.MultilangDaemonConfig prepare
INFO: MultiLangDaemon is adding the following fields to the User Agent: amazon-kinesis-client-library-java-1.7.3 amazon-kinesis-multi-lang-dae
mon/1.0.1 R /app/app.R
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.leases.impl.LeaseCoordinator <init>
INFO: With failover time 10000 ms and epsilon 25 ms, LeaseCoordinator will renew leases every 3308 ms, takeleases every 20050 ms, process maxi
mum of 2147483647 leases and steal 1 lease(s) at a time.
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker initialize
INFO: Initialization attempt 1
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker initialize
INFO: Initializing LeaseCoordinator
Mar 05, 2017 5:32:35 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker initialize
INFO: Syncing Kinesis shard info
Mar 05, 2017 5:32:36 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker initialize
INFO: Starting LeaseCoordinator
Mar 05, 2017 5:32:36 PM com.amazonaws.services.kinesis.leases.impl.LeaseTaker computeLeasesToTake
INFO: Worker [REDACTED] needed 2 leases but none were expired, so it will steal lease shardId-000000000002 from befs
4447-3adb-444f-8dc6-67504e5c86ef
Mar 05, 2017 5:32:36 PM com.amazonaws.services.kinesis.leases.impl.LeaseTaker computeLeasesToTake
INFO: Worker [REDACTED] saw 3 total leases, 0 available leases, 2 workers. Target is 2 leases, I have 0 leases, I wi
ll take 1 leases
Mar 05, 2017 5:32:36 PM com.amazonaws.services.kinesis.leases.impl.LeaseTaker takeLeases
INFO: Worker [REDACTED] successfully took 1 leases: shardId-000000000002
Mar 05, 2017 5:32:46 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker run
INFO: Initialization complete. Starting worker loop.
Mar 05, 2017 5:32:46 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker infoForce
INFO: Created new shardConsumer for : ShardInfo [shardId=shardId-000000000002, concurrencyToken=
shardIds=[shardId-000000000000], checkpoint={SequenceNumber: TRIM_HORIZON, SubsequenceNumber: 0}]
Mar 05, 2017 5:32:46 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.BlockOnParentShardTask call
INFO: No need to block on parents [shardId-000000000000] of shard shardId-000000000002
Mar 05, 2017 5:32:47 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisDataFetcher initialize
```

Running the MultiLangDaemon locally

Terminix: Default

```
[ec2-user@ip-10-10-1-11 logs]$ head -n 44 logger.log
INFO [2017-03-05 03:35:23] Starting R Kinesis Consumer application
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Start of initialize
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Hello
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 End of initialize
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:24 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:25 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:26 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:27 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:28 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:29 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:30 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:31 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:32 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:33 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:33 UTC] shardId-000000000000 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:35:34 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:35 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:36 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:37 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:38 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:39 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:40 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:41 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:42 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:43 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:43 UTC] shardId-000000000000 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:35:44 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:45 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:46 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:47 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:48 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:49 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:50 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:51 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:52 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:53 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:54 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:54 UTC] shardId-000000000000 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:35:55 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:56 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:57 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:58 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:59 UTC] shardId-000000000000 Received 3 records from Kinesis
[ec2-user@ip-10-10-1-11 logs]$
```

① Dockerize your Kinesis Consumer:

- Java
- R
- AWR, AWR.Kinesis packages
- app.R
- app.properties
- startup command

② Put it on Docker Hub

③ Run as a EC2 Container Service Task:

- Create an ECS cluster
- Create ECS Task Role
- Create a Task definition
- Run it (as a service)

Dockerize your Kinesis Consumer



CENTRAL
EUROPEAN
UNIVERSITY

The screenshot shows a GitHub repository page for `card-rocker/Dockerfile`. The URL is <https://github.com/cardcorp/card-rocker/blob/master/r-kinesis/Dockerfile>. The repository has 13 watchers, 6 stars, and 2 forks. The Dockerfile content is as follows:

```
FROM cardcorp/r-aws-java-pandoc:latest
MAINTAINER Gergely Daroczi <gergely.daroczi@card.com>

## Install extra AWR packages
RUN install2r --error \
  AWR.KMS \
  AWR.Kinesis \
  && rm -rf /tmp/downloaded_packages/ /tmp/*.rds

## Run MultiLangDaemon on /app
ENTRYPOINT ["/usr/bin/java", \
  "-cp", \
  "/usr/local/lib/R/site-library/AWR/java/*:/usr/local/lib/R/site-library/AWR.Kinesis/java/*", \
  "com.amazonaws.services.kinesis.multilang.MultiLangDaemon"]

## Override this if the consumer app is mounted elsewhere or the config file has a different name
CMD ["app/app.properties"]
```

Dockerize your Kinesis Consumer



CENTRAL
EUROPEAN
UNIVERSITY

The screenshot shows a GitHub repository page for `card-rocker/Dockerfile`. The URL is <https://github.com/cardcorp/card-rocker/blob/master/r-kinesis-example/Dockerfile>. The repository has 13 watchers, 6 stars, and 2 forks. The Dockerfile content is as follows:

```
FROM cardcorp/r-kinesis:latest
MAINTAINER Gergely Daroczi <gergely.daroczi@card.com>
## Add consumer
COPY files /app
```

Dockerize your Kinesis Consumer



CENTRAL
EUROPEAN
UNIVERSITY

The screenshot shows a GitHub repository page for 'cardcorp/card-rocker'. The repository has 13 forks and 6 stars. A commit by 'daroczig' titled 'run demo in east region + log in central folder' was made 22 hours ago. The file 'app.R' is an executable file with 24 lines (17 sloc) and 619 Bytes. The code in 'app.R' is as follows:

```
1 #!/usr/bin/r --vanilla
2 library(futile.logger)
3 library(AWS.Kinesis)
4
5 kinesis_consumer(
6
7     initialize      = function()
8         flog.info('Hello'),
9
10    processRecords = function(records)
11        flog.info(paste('Received', nrow(records), 'records from Kinesis')),
12
13    shutdown        = function()
14        flog.info('Bye'),
15
16    updater         = list(
17        list(1, function()
18            flog.info('Updating some data every minute')),
19        list(1/60*10, function()
20            flog.info('This is a high frequency updater call running every 10 seconds'))),
21
22    checkpointing = 1,
23    logfile       = '/logs/logger.log')
```

Dockerize your Kinesis Consumer

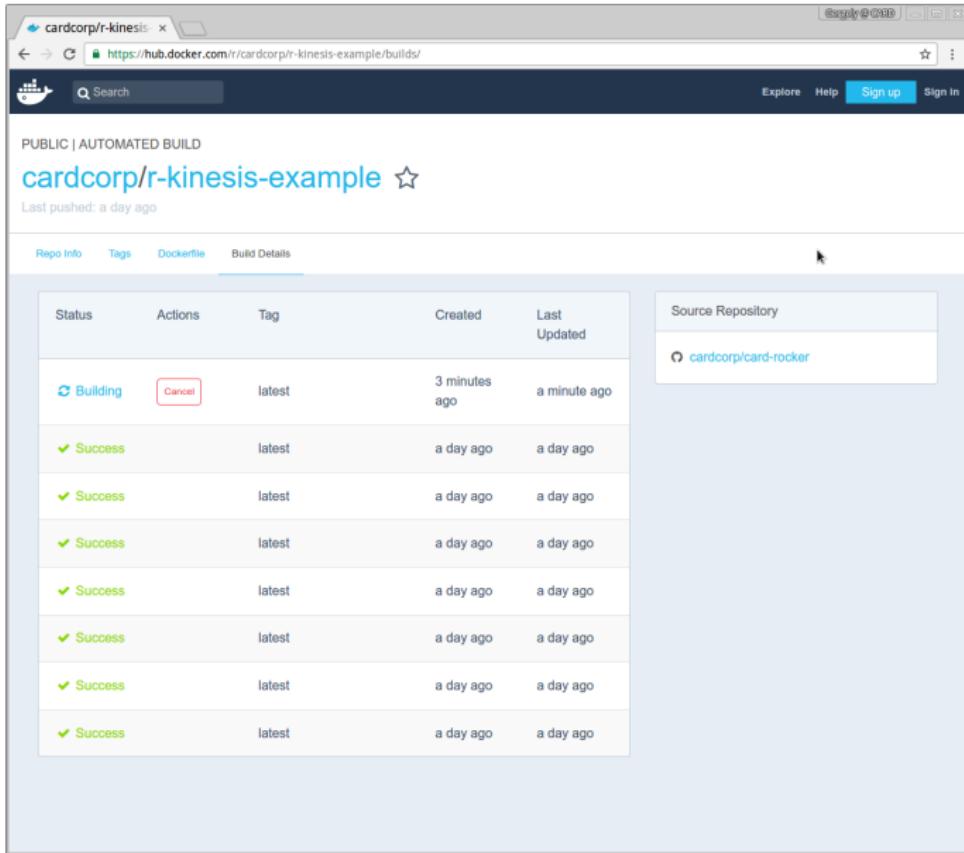


CENTRAL
EUROPEAN
UNIVERSITY

The screenshot shows a GitHub repository page for 'cardcorporation/card-rocker'. The specific file displayed is 'app.properties' located in the 'r-kinesis-example/files' directory. The code content is as follows:

```
1 executableName = /app/app.R
2 streamName = test-AWR
3 applicationName = AWR-demo-app
4 AWSCredentialsProvider = DefaultAWSCredentialsProviderChain
5 processingLanguage = R
6 regionName = us-east-1
```

Put it on Docker Hub



The screenshot shows the Docker Hub interface for the repository `cardcorp/r-kinesis-example`. The page title is "PUBLIC | AUTOMATED BUILD". Below it, the repository name is displayed with a star icon. A note says "Last pushed: a day ago". There are tabs for "Repo Info", "Tags", "Dockerfile", and "Build Details", with "Build Details" being the active tab.

The "Build Details" section contains a table with the following data:

Status	Actions	Tag	Created	Last Updated
 Building	<button>Cancel</button>	latest	3 minutes ago	a minute ago
 Success		latest	a day ago	a day ago
 Success		latest	a day ago	a day ago
 Success		latest	a day ago	a day ago
 Success		latest	a day ago	a day ago
 Success		latest	a day ago	a day ago
 Success		latest	a day ago	a day ago
 Success		latest	a day ago	a day ago

To the right of the table, there is a sidebar titled "Source Repository" containing the URL `https://hub.docker.com/r/cardcorp/card-rocker`.

Create an ECS cluster

The screenshot shows the 'Create Cluster' wizard in the AWS Management Console. The left sidebar shows 'Amazon ECS' and 'Clusters' selected. The main form is titled 'Create Cluster' and contains the following fields:

- Cluster name***: AWR-test
- Create an empty cluster
- EC2 Instance type***: t2.medium
- Number of instances***: 1
- EC2 Ami Id***: amzn-ami-2016.09.f-amazon-ecs-optimized [ami-b2df2ca4]
- EBS storage (GiB)***: 22
- Key pair**: (dropdown menu)

A note below the key pair field states: "You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the EC2 console."

Networking

Configure the VPC for your container instances to use. A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You can choose an existing VPC, or create a new one with this wizard.

- VPC**: Create a new vpc
- CIDR Block**: 10.0.0.0/16

Create ECS Task Role

IAM Management

https://console.aws.amazon.com/iam/home?region=us-east-1#roles

Services Resource Groups Gergely Darócz Gergely Darócz Global Support

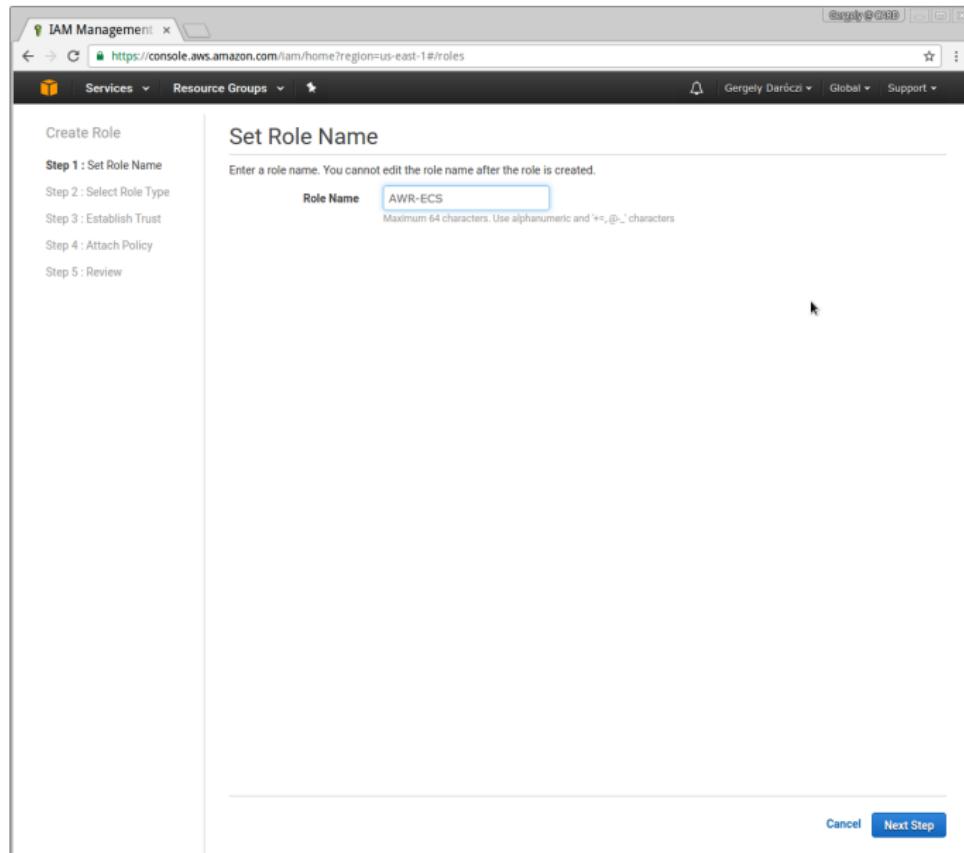
Create Role Step 1 : Set Role Name Step 2 : Select Role Type Step 3 : Establish Trust Step 4 : Attach Policy Step 5 : Review

Set Role Name

Enter a role name. You cannot edit the role name after the role is created.

Role Name AWR-ECS Maximum 64 characters. Use alphanumeric and '-' characters

Cancel Next Step



Create ECS Task Role

IAM Management Copy & CSEB

https://console.aws.amazon.com/iam/home?region=us-east-1#roles

Services Resource Groups

Create Role

Step 1 : Set Role Name

Step 2 : Select Role Type

Step 3 : Establish Trust

Step 4 : Attach Policy

Step 5 : Review

Select Role Type

AWS Service Roles (Role to allow Auto Scaling to access and update EC2 services)

- > **Amazon EC2 Role for EC2 Container Service** (Role to allow EC2 instances in an Amazon ECS cluster to access Amazon ECS.)
- > **Amazon EC2 Container Service Role** (Allows ECS to create and manage AWS resources on your behalf.)
- > **Amazon EC2 Container Service Task Role** (Allows ECS tasks to call AWS services on your behalf.)
- > **Amazon EC2 Spot Fleet Role** (Role to Allow EC2 Spot Fleet to request and terminate Spot Instances on your behalf.)
- > **Amazon Elastic MapReduce** (Role to allow EMR to access other AWS services such as EC2 on your behalf.)

Role for Cross-Account Access

Role for Identity Provider Access

The screenshot shows the 'Create Role' wizard in the AWS IAM Management console. The user is at Step 2: Select Role Type. The 'AWS Service Roles' section is selected, displaying a list of roles that grant access to specific AWS services. The 'Amazon EC2 Container Service Task Role' is highlighted with a blue border, indicating it is the chosen role type. Other options like 'Role for Cross-Account Access' and 'Role for Identity Provider Access' are also listed.

Create ECS Task Role

IAM Management > Services > Resource Groups

https://console.aws.amazon.com/iam/home?region=us-east-1#roles

Create Role

Step 1 : Set Role Name

Step 2 : Select Role Type

Step 3 : Establish Trust

Step 4 : Attach Policy

Step 5 : Review

Attach Policy

Select one or more policies to attach. Each role can have up to 10 policies attached.

Filter: Policy Type ▾ kinesis Showing 7 results

	Policy Name	Attached Entities	Creation Time	Edited Time
<input type="checkbox"/>	AmazonKinesisFullAccess	1	2015-02-06 10:40 PST	2015-02-06 10:40 PST
<input checked="" type="checkbox"/>	AmazonKinesisReadOnlyA...	1	2015-02-06 10:40 PST	2015-02-06 10:40 PST
<input type="checkbox"/>	AmazonKinesisAnalyticsF...	0	2016-09-21 12:01 PST	2016-09-21 12:01 PST
<input type="checkbox"/>	AmazonKinesisAnalyticsR...	0	2016-09-21 11:16 PST	2016-09-21 11:16 PST
<input type="checkbox"/>	AmazonKinesisFirehoseFu...	0	2015-10-07 11:45 PST	2015-10-07 11:45 PST
<input type="checkbox"/>	AmazonKinesisFirehoseR...	0	2015-10-07 11:43 PST	2015-10-07 11:43 PST
<input type="checkbox"/>	AWSLambdaKinesisExecu...	0	2015-04-09 08:14 PST	2015-04-09 08:14 PST

Cancel Previous Next Step

Create a Task definition

The screenshot shows the 'Create a Task Definition' page in the AWS CloudWatch Metrics console. The task definition name is set to 'AWR-logger'. The task role is 'AWR-ECS'. The network mode is 'Bridge'. There is one constraint defined: 'Add constraint'. Under 'Container Definitions', there is one container named 'logger' with the image 'awslogs/awslogs'. Under 'Volumes', there is one volume named 'logger' with the source path '/var/log'. The bottom of the page has a 'Configure via JSON' button.

Amazon EC2 Cont... https://console.aws.amazon.com/ecs/home?region=us-east-1#taskDefinitions/create Gergely Darócz Gergely Darócz N. Virginia Support

Services Resource Groups

Amazon ECS Clusters Task Definitions Repositories

Create a Task Definition

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. [Learn more](#)

Task Definition Name* AWR-logger

Task Role AWR-ECS

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon EC2 Container Service Task Role in the [IAM Console](#).

Network Mode Bridge

Constraint

Constraints allow you to filter the instances used for your placement strategies using built-in or custom attributes. The scheduler first filters the instances that match the constraints and then applies the placement strategy to place the task.

Type	Expression
Add constraint	

Container Definitions

Add container

Container Name	Image	Hard/Soft memory limits (MB)	Essential
logger	awslogs/awslogs		

No results

Volumes

Add volume

Name	Source Path
logger	/var/log

No results

Configure via JSON

Create a Task definition

The screenshot shows the AWS ECS Task Definitions creation interface. A modal window titled "Add volume" is open, prompting for a volume name ("logs") and source path ("/logs"). The main page includes sections for "Constraints", "Type" (with an "Add constraint" button), "Container Definitions" (empty), "Volumes" (empty), and a "Configure via JSON" button.

A task definition for your cluster

Add volume

Name* logs

Source path /logs

*Required

Cancel Add

Constraint

Constraints allow you to filter the instances used for your placement strategies using built-in or custom attributes. The scheduler first filters the instances that match the constraints and then applies the placement strategy to place the task.

Type

Add constraint

Container Definitions

Add container

Container Name	Image	Hard/Soft memory limits (MB)	Essential
No results			

Volumes

Name	Source Path
No results	

Add volume

Configure via JSON

Create a Task definition

The screenshot shows the 'Create a Task Definition' wizard in the AWS ECS console. The left sidebar lists 'Amazon ECS', 'Clusters', 'Task Definitions' (which is selected), and 'Repositories'. The main area is titled 'Add container' under 'Standard' type. It includes fields for 'Container name*' (set to 'logger'), 'Image*' (set to 'cardcorp/r-kinesis-example:latest'), and 'Memory Limits (MB)*' (set to '512'). A note explains that hard and soft limits are defined in task definitions. Below this is a 'Port mappings' section with 'Host port', 'Container port', and 'Protocol' (set to 'tcp') dropdowns, along with a 'Add port mapping' button. The 'Advanced container configuration' section contains an 'ENVIRONMENT' tab with 'CPU units' (empty) and 'Essential' checked. A tooltip for 'CPU units' states: 'The number of cpu units to reserve for the container. A container instance has 1.024 cpu units for every CPU core.' At the bottom are 'Required' and 'Cancel' buttons, and a large blue 'Add' button.

Create a Task definition

The screenshot shows the 'Create a Task Definition' wizard in the AWS ECS console. The left sidebar lists 'Amazon ECS', 'Clusters', 'Task Definitions' (which is selected), and 'Repositories'. The main area is titled 'Create a Task Definition' with the sub-instruction 'A task definition specifies the resources required for your containers to run'. The 'Add container' step is active.

STORAGE AND LOGGING

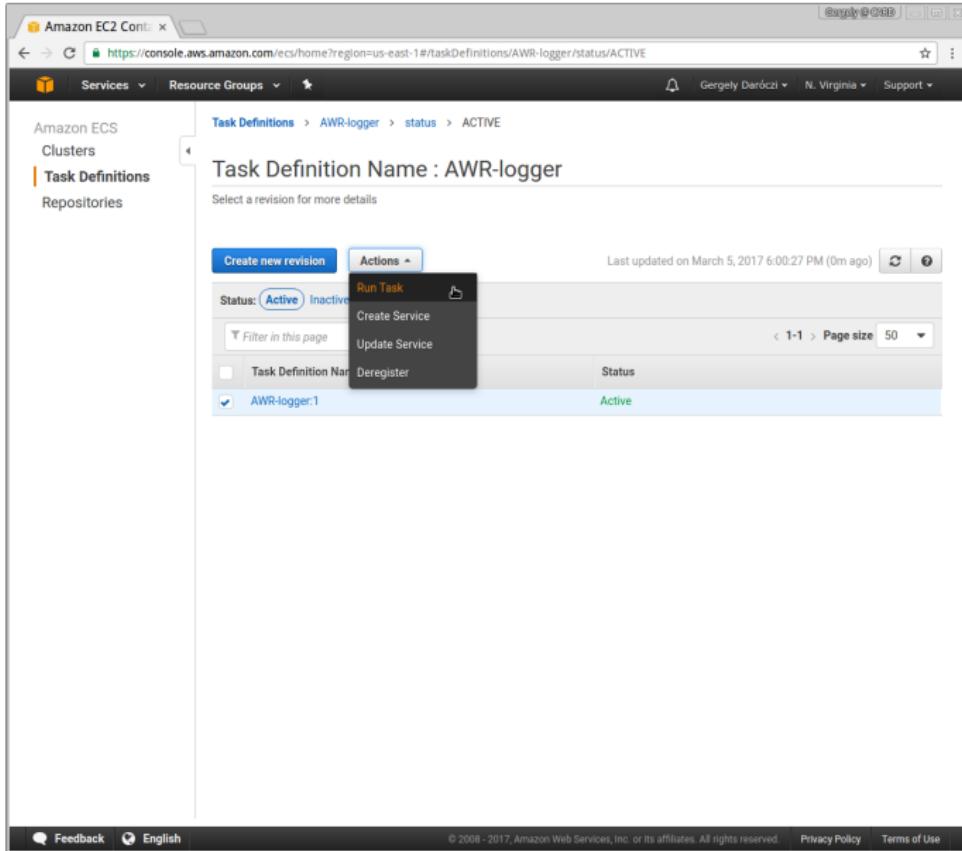
- Read only root file system**: An unchecked checkbox.
- Mount points**: A section for mapping volumes from the host to the container.
 - Type**: Set to 'Source volume'.
 - Container path**: Set to '/logs'.
 - Read only**: An unchecked checkbox.
- Add mount point**: A button to add more mount points.
- Volumes from**: A section for mapping volumes from another container.
 - Type**: Set to 'Source container'.
 - Read only**: An unchecked checkbox.
- Add volumes**: A button to add more volumes.
- Log configuration**: A section for log driver and options.
 - Log driver**: Set to '<none>'.
 - Log options**: A table with two rows:
 - Key**: 'Add key'
 - Value**: 'Add value'

SECURITY

* Required

Add

Run the ECS Task



The screenshot shows the AWS ECS Task Definitions console. The URL is <https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/AWR-logger/status/ACTIVE>. The left sidebar shows 'Task Definitions' selected. The main page displays the 'Task Definition Name : AWR-logger'. A dropdown menu under 'Actions' shows 'Run Task' as the selected option. The table below lists the task definition with one entry: 'AWR-logger:1' which is 'Active'. The status was last updated on March 5, 2017 at 6:00:27 PM.

Task Definition Name	Status
AWR-logger:1	Active

Run the ECS Task

The screenshot shows the AWS ECS console interface. The left sidebar has 'Amazon ECS' selected under 'Clusters'. The main area shows the 'Cluster : AWR' page. The status is 'ACTIVE'. It displays metrics: Registered container instances (1), Pending tasks count (0), and Running tasks count (1). Below this is a table of tasks.

Task	Task Definition	Group	Container Inst...	Last status	Desired status	Started By
0c9f224a-780b... 0c9f224a-780b...	AWR-logger:1	family:AWR-log...	27b493570e2...	RUNNING	RUNNING	

At the bottom, the URL is https://console.aws.amazon.com/ecs/home?region=..., and the footer includes links for Privacy Policy and Terms of Use.

Run the ECS Task

Terminix: Default

```

1: ec2-user@[REDACTED] ~
/ ssh -i ~/.ssh/[REDACTED] ec2-user@[REDACTED]
Last login: Mon Mar  6 02:05:29 2017 from [REDACTED]

[REDACTED]
|_ \_|_/_| Amazon ECS-Optimized Amazon Linux AMI 2016.09.f
 \_\_|_/_/ [REDACTED]

For documentation visit, http://aws.amazon.com/documentation/ecs
4 package(s) needed for security, out of 6 available
Run "sudo yum update" to apply all updates.
[ec2-user@[REDACTED] ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS          NAMES
AMES                cardcorp/r-kinesis-example:latest   "/usr/bin/java -cp /u"   22 hours ago      Up 22 hours         e
cs-AWR-logger-1-logger-92ca888bd5d681e59d01   amazon/amazon-ecs-agent:latest    "/agent"           23 hours ago      Up 23 hours         e
cs-agent
[ec2-user@[REDACTED] ~]$ pgrep app.R
29435
[ec2-user@[REDACTED] ~]$ head -n 10 /logs/logger.log
INFO [2017-03-05 03:35:23] Starting R Kinesis Consumer application
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Start of initialize
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Hello
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 End of initialize
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:24 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:25 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:26 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:27 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:28 UTC] shardId-000000000000 Received 2 records from Kinesis
[ec2-user@[REDACTED] ~]$
```

Scaling the Kinesis Consumer up

Terminix: Default

```

1: ec2-user@ip-172-31-10-15: ~ %
INFO [2017-03-05 03:43:01 UTC] shardId-000000000000 Received 1 records from Kinesis
INFO [2017-03-05 03:43:01 UTC] shardId-000000000000 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:43:09 UTC] shardId-000000000000 Shutting down
INFO [2017-03-05 03:43:09 UTC] shardId-000000000000 Bye
INFO [2017-03-05 03:43:15] Starting R Kinesis Consumer application
INFO [2017-03-05 03:43:15 UTC] shardId-000000000002 Start of initialize
INFO [2017-03-05 03:43:15 UTC] shardId-000000000002 Hello
INFO [2017-03-05 03:43:15 UTC] shardId-000000000002 End of initialize
INFO [2017-03-05 03:43:15] Starting R Kinesis Consumer application
INFO [2017-03-05 03:43:16 UTC] shardId-000000000001 Start of initialize
INFO [2017-03-05 03:43:16 UTC] shardId-000000000001 Hello
INFO [2017-03-05 03:43:16 UTC] shardId-000000000001 End of initialize
INFO [2017-03-05 03:44:32 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:32 UTC] shardId-000000000002 Updating some data every minute
INFO [2017-03-05 03:44:32 UTC] shardId-000000000002 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:44:33 UTC] shardId-000000000002 Received 3 records from Kinesis
INFO [2017-03-05 03:44:34 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:35 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:36 UTC] shardId-000000000001 Received 2 records from Kinesis
INFO [2017-03-05 03:44:36 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:36 UTC] shardId-000000000001 Updating some data every minute
INFO [2017-03-05 03:44:36 UTC] shardId-000000000001 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:44:37 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:38 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:39 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:39 UTC] shardId-000000000001 Received 2 records from Kinesis
INFO [2017-03-05 03:44:40 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:40 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:41 UTC] shardId-000000000001 Received 2 records from Kinesis
INFO [2017-03-05 03:44:42 UTC] shardId-000000000001 Received 2 records from Kinesis
INFO [2017-03-05 03:44:43 UTC] shardId-000000000002 Received 2 records from Kinesis
INFO [2017-03-05 03:44:43 UTC] shardId-000000000002 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:44:44 UTC] shardId-000000000002 Received 1 Records from Kinesis
INFO [2017-03-05 03:44:45 UTC] shardId-000000000002 Received 2 records from Kinesis
INFO [2017-03-05 03:44:45 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:46 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:46 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:47 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:47 UTC] shardId-000000000001 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:44:47 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:48 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:48 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:49 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:49 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:50 UTC] shardId-000000000001 Received 1 records from Kinesis
```

```

Nice example project, but ...

- I might want to avoid publishing my Consumer on Docker Hub
- I might want to avoid publishing my code on GitHub
- I might want to avoid committing credentials etc to the repo

Problems:

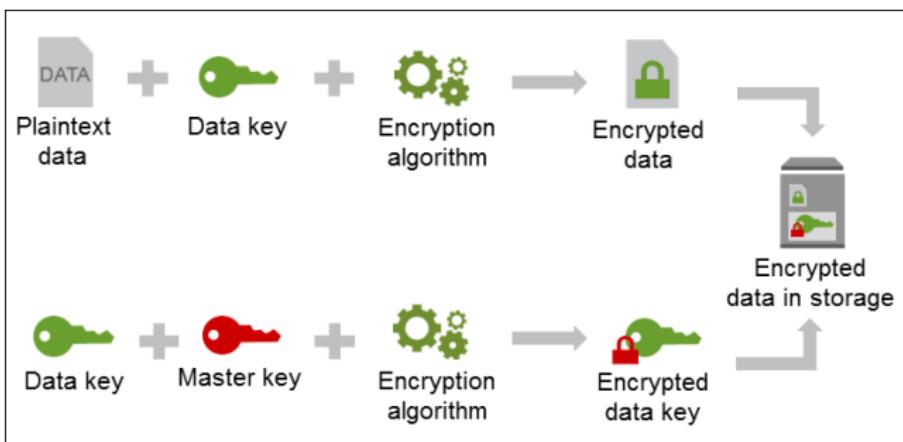
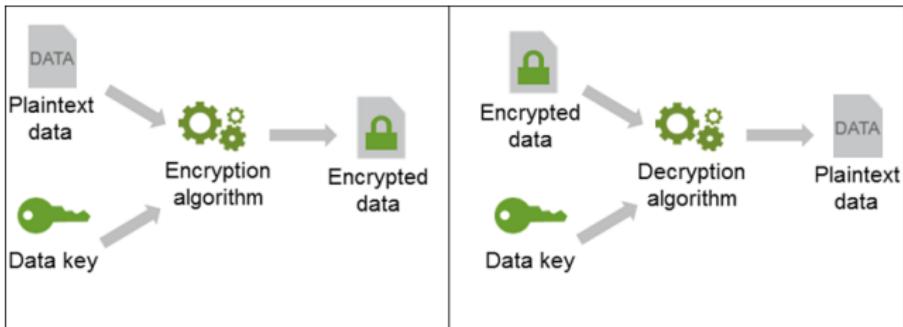
- How to store credentials in the Docker images?
- Where to store the Docker images?

Nice example project, but ...

- I might want to avoid publishing my Consumer on Docker Hub
- I might want to avoid publishing my code on GitHub
- I might want to avoid committing credentials etc to the repo

Problems:

- How to store credentials in the Docker images? **KMS**
- Where to store the Docker images? **ECR**



Source: AWS Encryption SDK

- encrypt up to 4 KB of arbitrary data:

```
> library(AWR.KMS)
> kms_encrypt('alias/mykey', 'foobar')
[1] "Base-64 encoded ciphertext"
```

- decrypt such Base-64 encoded ciphertext back to plaintext:

```
> kms_decrypt('Base-64 encoded ciphertext')
[1] "foobar"
```

- generate a data encryption key:

```
> kms_generate_data_key('alias/mykey')
$cipher
[1] "Base-64 encoded, encrypted data encryption key"
$key
[1] "alias/mykey"
$text
[1] 00 01 10 11 00 01 10 11 ...
```

# Encrypting Data Larger Than 4 KB?

```
let's say we want to encrypt the mtcars dataset stored in JSON
library(jsonlite)
data <- toJSON(mtcars)

generate a 256-bit data encryption key (that's supported by digest::AES)
library(AWR.KMS)
key <- kms_generate_data_key('alias/mykey', byte = 32L)

convert the JSON to raw so that we can use that with digest::AES
raw <- charToRaw(data)
the text length must be a multiple of 16 bytes
https://github.com/sdoyen/r_password_crypt/blob/master/crypt.R
raw <- c(raw, as.raw(rep(0, 16 - length(raw) %% 16)))

encrypt the raw object with the new key + digest::AES
the resulting text and the encrypted key can be stored on disk
library(digest)
aes <- AES(key$text)
base64_enc(aes$encrypt(raw))

decrypt the above returned ciphertext using the decrypted key
rawToChar(aes$decrypt(base64_dec(...), raw = TRUE))
```

# Example “Production” Consumer App

```
library(AWR.Kinesis); library(jsonlite); library(AWR.KMS); library(futile.logger); flog.threshold(DEBUG)

kinesis_consumer(
 initialize = function() {
 flog.info('Decrypting Redis hostname via KMS')
 host <- kms_decrypt('AQECAHiiz4GEPFQLL9AAON5TY/lDR5euQQScpXQU9iYTn+u... ')
 flog.info('Connecting to Redis')
 library(rredis); redisConnect(host = host)
 flog.info('Connected to Redis')
 },
 processRecords = function(records) {
 flog.info(paste('Received', nrow(records), 'records from Kinesis'))
 for (record in records$data) {
 flight <- fromJSON(record)$dest
 if (!is.null(flight)) {
 flog.debug(paste('Adding +1 to', flight))
 redisIncr(sprintf('flight:%s', flight))
 } else {
 flog.error('Flight destination not found')
 }
 }
 },
 updaters = list(
 list(1/6, function() {
 flog.info('Checking overall counters')
 flights <- redisKeys('flight:*')
 for (flight in flights) {
 flog.debug(paste('Found', redisGet(flight), sub('^flight:', '', flight)))
 }
 })),
 logfile = '/logs/redis.log')
```

Dockerfile:

```
FROM cardcorp/r-kinesis:latest
MAINTAINER Gergely Daroczi <gergely.daroczi@card.com>

Install R package to interact with Redis
RUN install2.r --error rredis && rm -rf /tmp/downloaded_packages/ /tmp/*.rds

Add consumer
COPY files /app
```

Build and push to ECR:

```
docker build -t cardcorp/r-kinesis-secret .
`aws ecr get-login --region us-east-1`
docker tag -f cardcorp/r-kinesis-secret:latest \
 ***.dkr.ecr.us-east-1.amazonaws.com/cardcorp/r-kinesis-secret:latest
docker push ***.dkr.ecr.us-east-1.amazonaws.com/cardcorp/r-kinesis-secret:latest
```

# Shiny Dashboard

```
library(treemap);library(highcharter);library(nycflights13)
library(rredis);redisConnect(host = '***', port = '***')

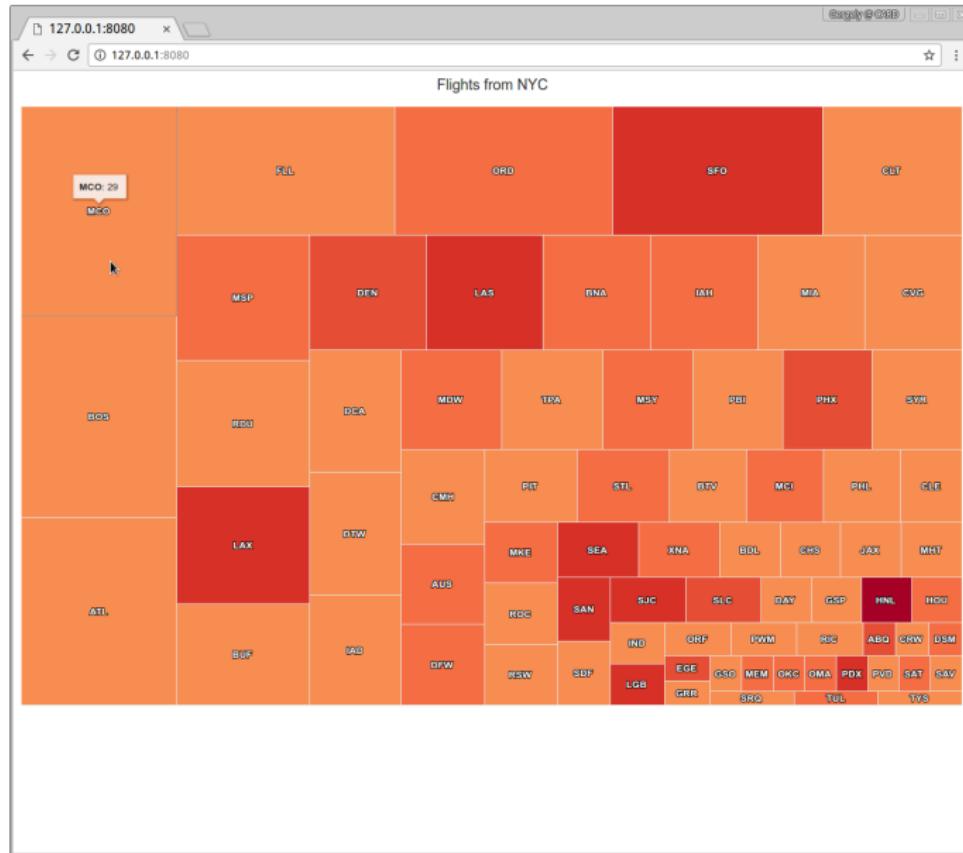
ui <- shinyUI(highchartOutput('treemap', height = '800px'))
server <- shinyServer(function(input, output, session) {

 destinations <- reactive({
 reactiveTimer(2000)()
 flights <- redisMGet(redisKeys('flight:*'))
 flights <- data.frame(faa = sub('^flight:', '', names(flights)),
 N = as.numeric(flights))
 merge(flights, airports, by = 'faa')
 })

 output$treemap <- renderHighchart({
 tm <- treemap(destinations(), index = c('faa'),
 vSize = 'N', vColor = 'tz',
 type = 'value', draw = FALSE)
 hc_title(hctreemap(tm, animation = FALSE), text = 'Flights from NYC')
 })
}

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

# Shiny Dashboard



- AWR repo:
  - 9.7 GB
  - 388 tags/versions
  - GitLab + CI + drat

```
install.packages('AWR', repos = 'https://cardcorp.gitlab.io/AWR')
```

- AWR repo:
  - 9.7 GB
  - 388 tags/versions
  - GitLab + CI + drat

```
install.packages('AWR', repos = 'https://cardcorp.gitlab.io/AWR')
```

- Submitted to CRAN on
  - 2016-12-05

- AWR repo:
  - 9.7 GB
  - 388 tags/versions
  - GitLab + CI + drat

```
install.packages('AWR', repos = 'https://cardcorp.gitlab.io/AWR')
```

- Submitted to CRAN on
  - 2016-12-05
  - 2017-01-09
  - 2017-01-10
  - 2017-01-11
  - 2017-01-11
  - 2017-01-13

- AWR repo:
  - 9.7 GB
  - 388 tags/versions
  - GitLab + CI + drat

```
install.packages('AWR', repos = 'https://cardcorp.gitlab.io/AWR')
```

- Submitted to CRAN on
  - 2016-12-05
  - 2017-01-09
  - 2017-01-10
  - 2017-01-11
  - 2017-01-11
  - 2017-01-13
- Release cycle: 2 minor, ~125 patch versions in the past 12 months
- CI



```
> library(rJava)
> kc <- .jnew('com.amazonaws.services.s3.AmazonS3Client')
> kc$getS3AccountOwner()$getDisplayName()
[1] "foobar"
```



Because "S"  
is so 1992.