Easy access to data with Presto

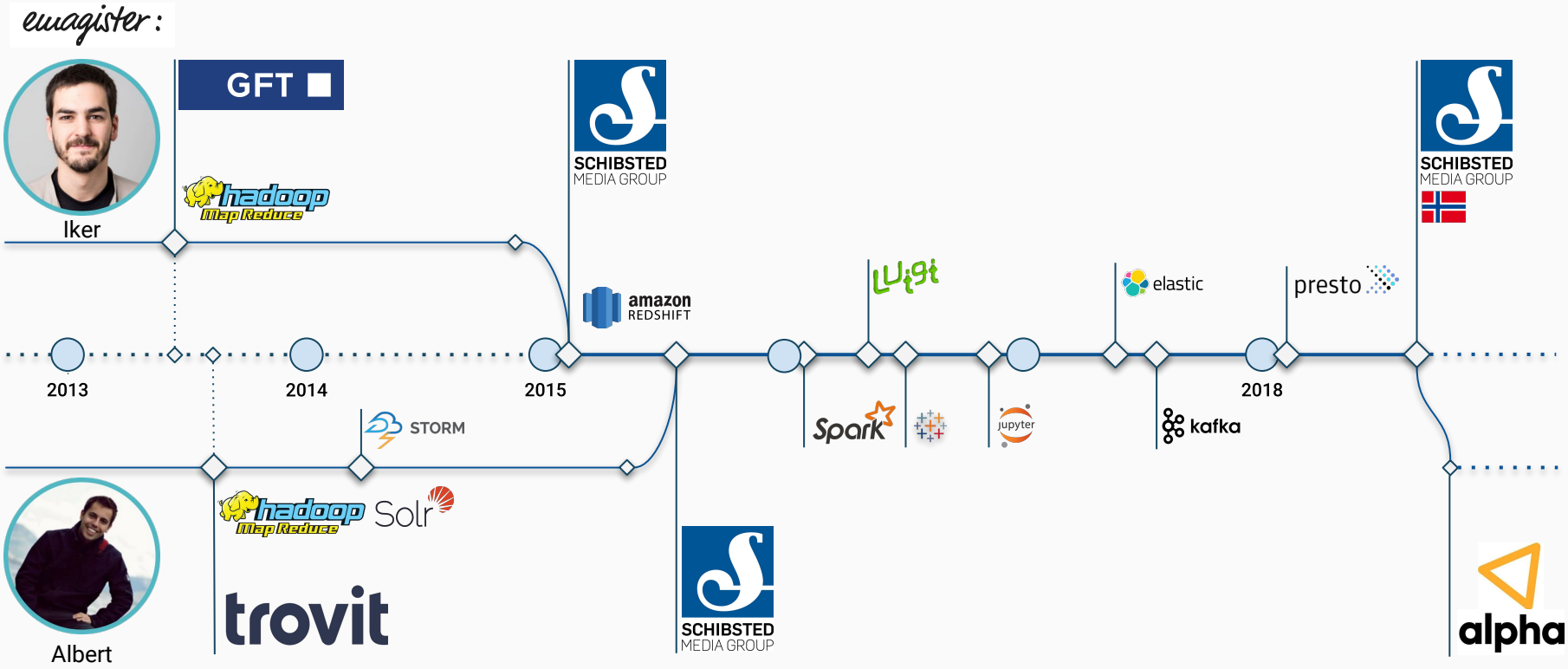
Iker Martinez de Apellaniz (@mitxino77)
Albert Franzi (@FranziCros)

DataEngConf 2018 - Schibsted Media Group

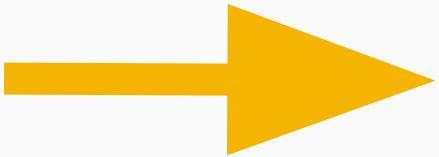
Slides available in:

<< bitly link here >>

About us


emagister:
GFT
Iker
hadoop
Map Reduce
SCHIBSTED
MEDIA GROUP
amazon
REDSHIFT
Luigi
elastic
presto
SCHIBSTED
MEDIA GROUP
2013
2014
2015
2018
STORM
Spark
jupyter
kafka
hadoop
Map Reduce
Solr
trovit
Albert
SCHIBSTED
MEDIA GROUP
alpha

Our Journey



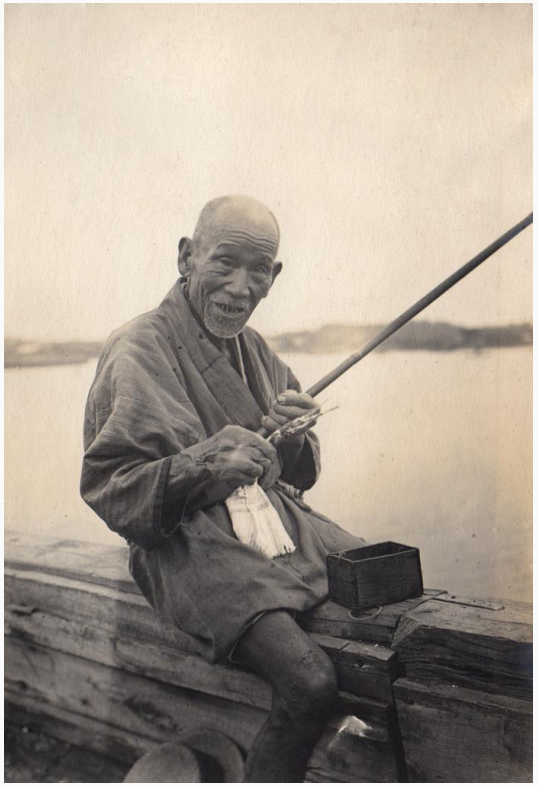
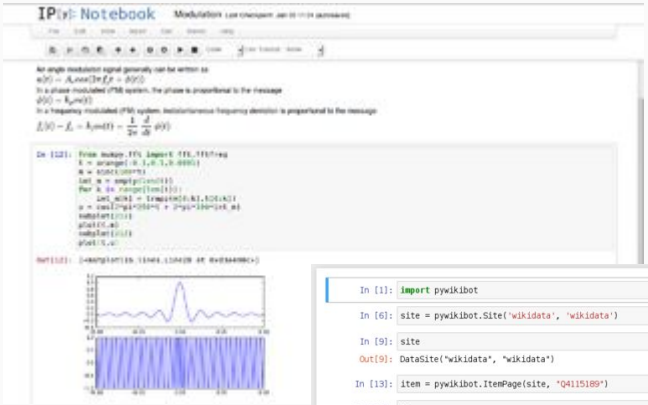


Tableau-aaS

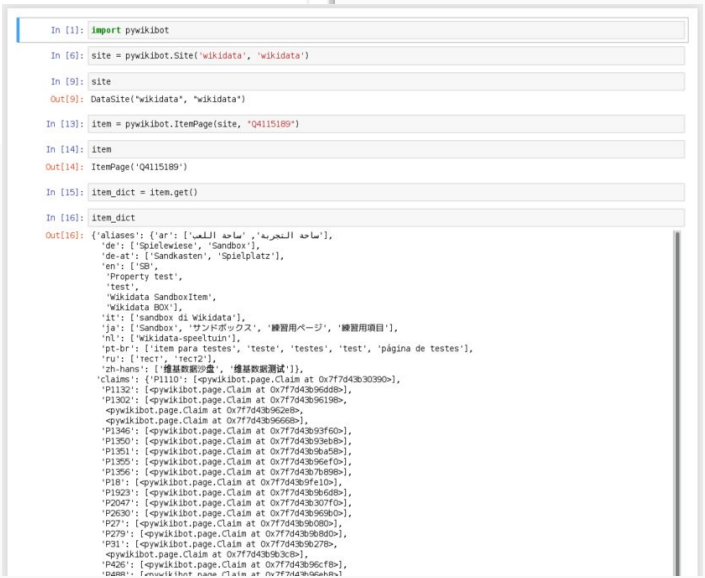

Traffic Info
Conversion
Pipeline
User Traffic
By Region
User Traffic Breakdown
By Month
Total Visitors
Unique Visitors
Boston, MA
3,441
User Region Traffic
By Channel
Direct
Indirect
Overall Returning vs Unique Visitors
Over Last Year
30%
Unique Visitors
70%
Returning Visitors
90%
Conversion
Mobile vs Desktop Usage
Over Last Year
9%
Desktop
91%
Mobile
90%
iOS
Browser Breakdown
By Device & Software
Mobile: Chrome 37%
Mobile: Safari 30%
Desktop: Chrome 12%
Desktop: Firefox 8%
Desktop: Opera 7%
Miscellaneous 4%
Desktop: Safari 1.5%
Desktop: IE 0.5%
Total Visitors vs Unique
Distribution of Unique Visitors vs Total by Month
Browser Type
All Browsers
Total Visitors
Unique Visitors
Apr '17
May '17
Jun '17
Jul '17
Aug '17
Sept '17
Oct '17

Jupyter-aaS



jupyter

Jupyter as a Service

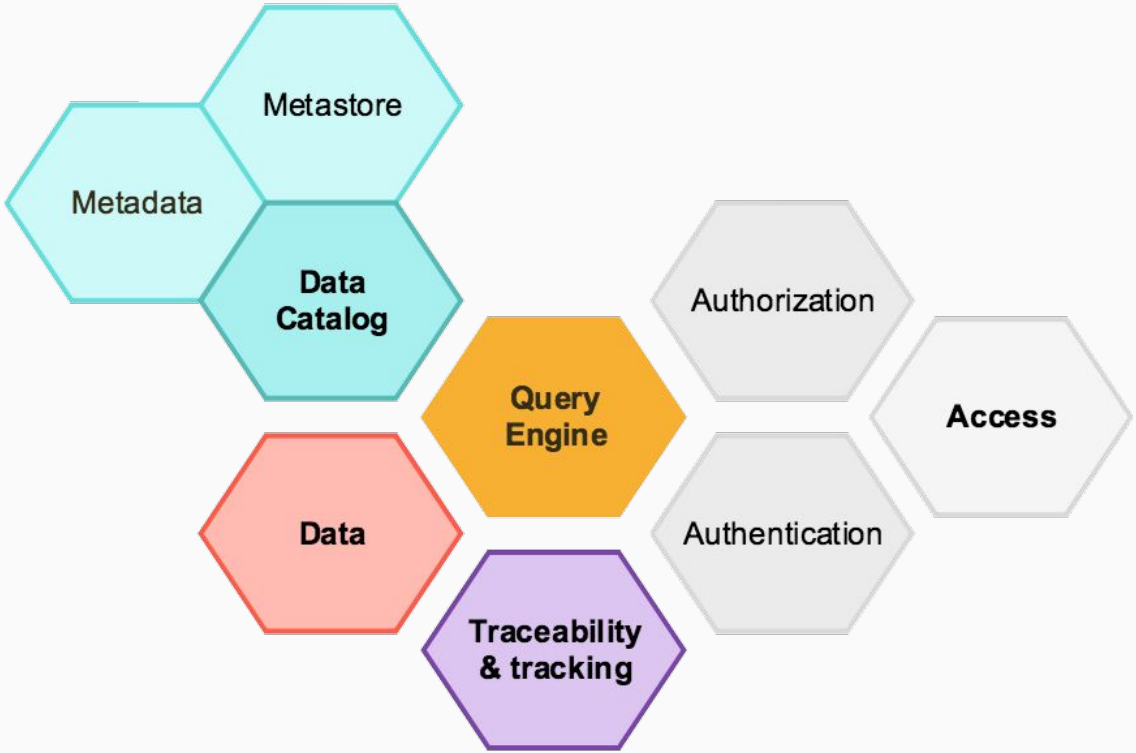


```
In [1]: import pywikibot

In [6]: site = pywikibot.Site('wikidata', 'wikidata')

In [9]: site
Out[9]: DataSite("wikidata", "wikidata")

In [13]: item = pywikibot.ItemPage(site, "Q4115189")

In [14]: item
Out[14]: ItemPage('Q4115189')

In [15]: item_dict = item.get()

In [16]: item_dict
Out[16]: {'aliases': {'ar': ['ساحة اللعب', 'ساحة التجربة'],
                      'de': ['Spielewiese', 'Sandbox'],
                      'de-at': ['Sandkasten', 'Spielplatz'],
                      'en': ['SB',
                       'Property test',
                       'test',
                       'Wikidata SandboxItem',
                       'Wikidata BOX'],
                      'it': ['sandbox di Wikidata'],
                      'ja': ['Sandbox', 'サンドボックス', '練習用ページ', '練習用項目'],
                      'nl': ['Wikidata-speeltuin'],
                      'pt-br': ['item para testes', 'teste', 'testes', 'test', 'página de testes'],
                      'ru': ['тест', 'тест2'],
                      'zh-hans': ['维基数据沙盒', '维基数据测试']},
           'claims': {'P111O': [<pywikibot.page.Claim at 0x7f7d43b30390>],
                      'P1132': [<pywikibot.page.Claim at 0x7f7d43b96dd8>],
                      'P1302': [<pywikibot.page.Claim at 0x7f7d43b96198>,
                       <pywikibot.page.Claim at 0x7f7d43b962e8>,
                       <pywikibot.page.Claim at 0x7f7d43b96668>],
                      'P1346': [<pywikibot.page.Claim at 0x7f7d43b93f60>],
                      'P1350': [<pywikibot.page.Claim at 0x7f7d43b93eb8>],
                      'P1351': [<pywikibot.page.Claim at 0x7f7d43b9ba58>],
                      'P1355': [<pywikibot.page.Claim at 0x7f7d43b96ef0>],
                      'P1356': [<pywikibot.page.Claim at 0x7f7d43b7b898>],
                      'P18': [<pywikibot.page.Claim at 0x7f7d43b9fe10>],
                      'P1923': [<pywikibot.page.Claim at 0x7f7d43b9b6d8>],
                      'P2047': [<pywikibot.page.Claim at 0x7f7d43b307f0>],
                      'P2630': [<pywikibot.page.Claim at 0x7f7d43b969b0>],
                      'P27': [<pywikibot.page.Claim at 0x7f7d43b9b080>],
                      'P279': [<pywikibot.page.Claim at 0x7f7d43b9b8d0>],
                      'P31': [<pywikibot.page.Claim at 0x7f7d43b9b278>,
                       <pywikibot.page.Claim at 0x7f7d43b9b3c8>],
                      'P426': [<pywikibot.page.Claim at 0x7f7d43b96cf8>],
                      'P488': [<pywikibot.page.Claim at 0x7f7d43b96eb8>],
```

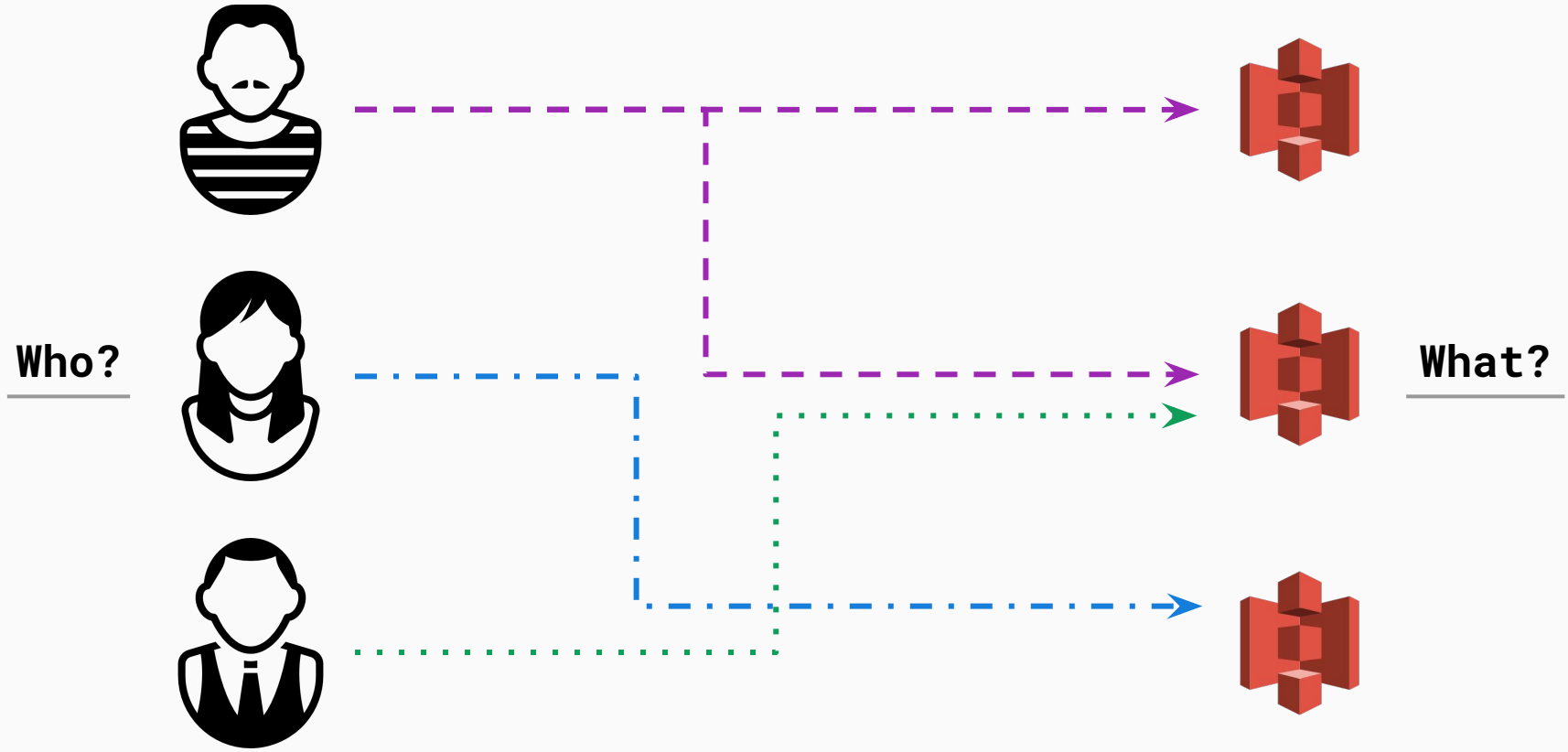
```
SELECT the_data_I_need
FROM   the_table_Im_looking_for
WHERE  my_analysis = 'Makes Sense';
```


S3
Query Engine

Providing Presto as a Service



Auth* (authentication & authorization)



Auth* (authentication & authorization)

PasswordAuthenticator

AWS Keys

IAM
Authenticator


```
{
 "User": {
 "UserName": "albert.franzi",
 "PasswordLastUsed": "2018-11-24T09:15:20Z",
 "CreateDate": "2015-10-13T06:32:28Z",
 "UserId": "XXXXXXXXXXXXXXX",
 "Path": "/",
 "Arn": "arn:aws:iam::0000000000:user/albert.franzi"
 }
}
```

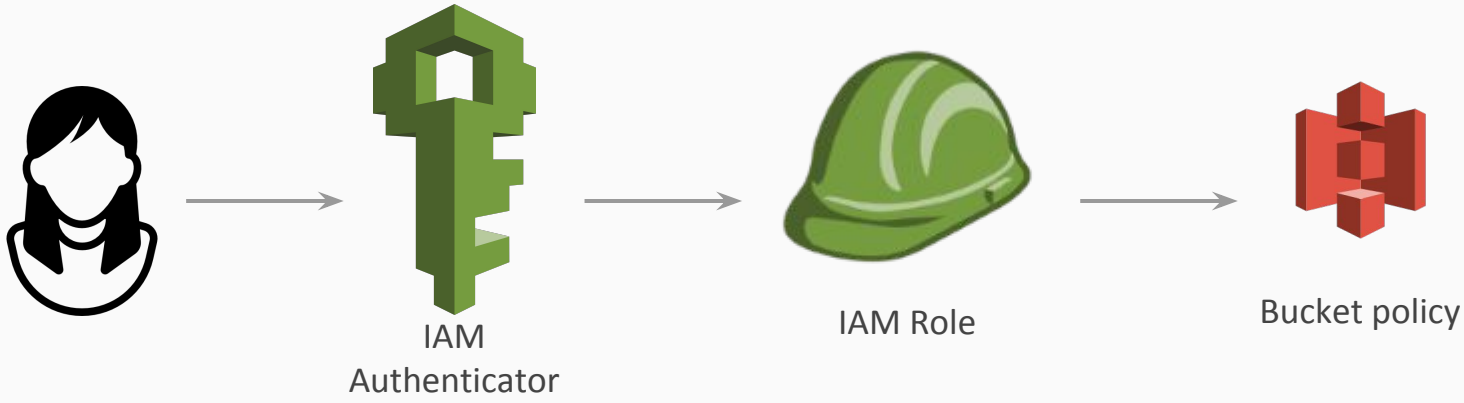
Auth* (authentication & authorization)

```
val authenticationCache: LoadingCache[Credentials, Principal] = CacheBuilder
  .newBuilder()
  .expireAfterWrite(cacheTtl.toMillis, MILLISECONDS)
  .build(new CacheLoader[Credentials, Principal] {
    override def load(key: Credentials): Principal = authenticate(key)
  })
```

Ref: *github.com/google/guava/wiki/CachesExplained*

Avoid multiple IAM calls by implementing a cache.

Auth* (authentication & authorization)



Traceability

IamAuthenticator

EventListener (Logs): Report to Datadog all created and completed queries.

SystemAccessControl: Report to Datadog all user interactions with Presto.

Traceability - SPI plugins < SystemAccessControl >

```
import com.facebook.presto.spi.security.AccessDeniedException._
import com.facebook.presto.spi.security.{Identity, SystemAccessControl}
import com.facebook.presto.spi.CatalogSchemaTableName

trait TableAccessControl extends SystemAccessControl {
  self: Logging with DatadogReporter =>

  override def checkCanCreateTable(identity: Identity, table: CatalogSchemaTableName): Unit = {
    reportTableEvent("can_create_table", identity, table)
  }

  override def checkCanDropTable(identity: Identity, table: CatalogSchemaTableName): Unit = {
    reportTableEvent("can_drop_table", identity, table)
  }

  override def checkCanRenameTable(identity: Identity, table: CatalogSchemaTableName,
      newTable: CatalogSchemaTableName): Unit = {
    reportTableEvent("can_rename_table", identity, table, allowed = false)
    denyRenameTable(table.toString, newTable.toString)
  }

  ...
}
```

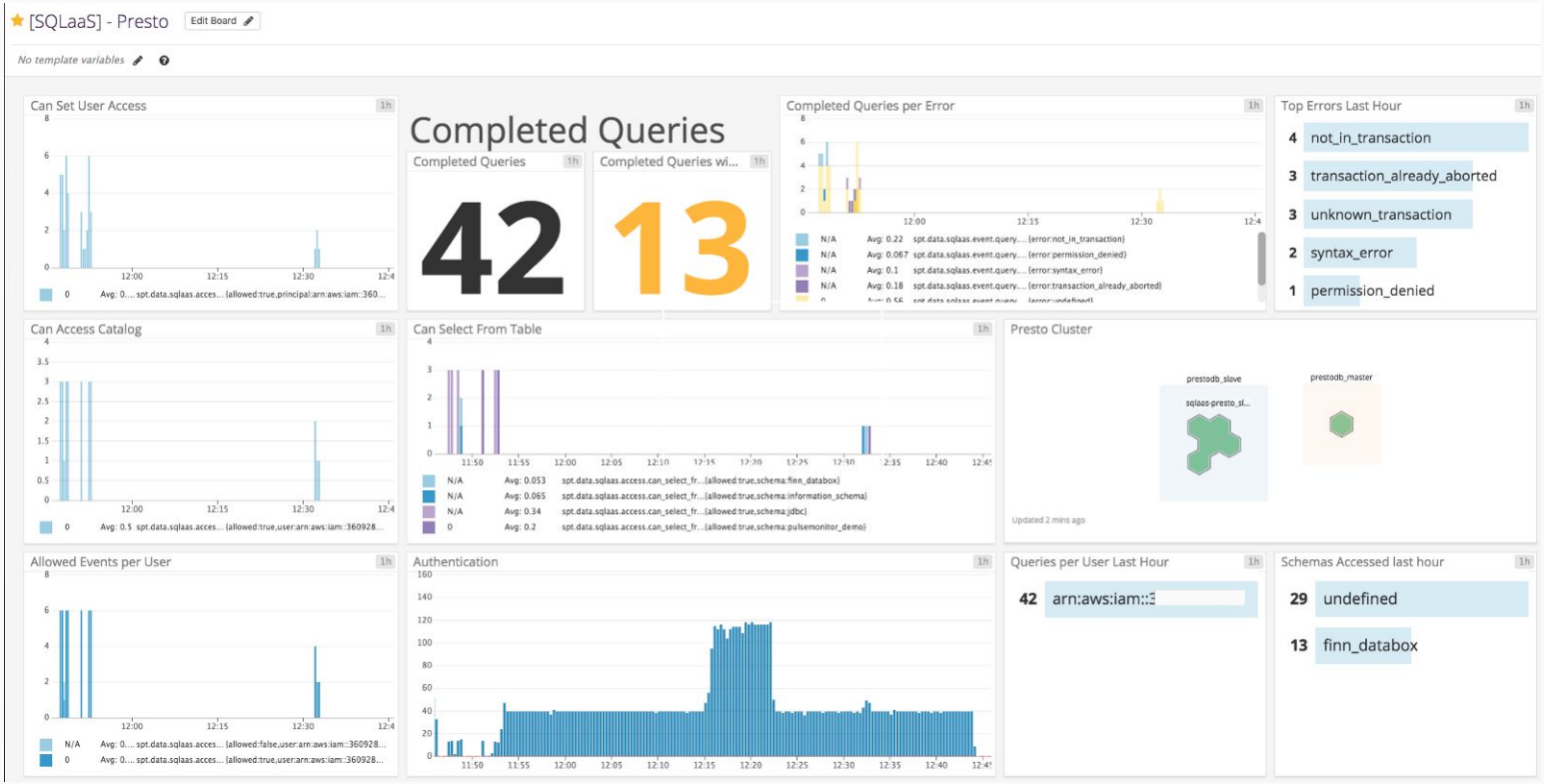
Traceability - SPI plugins < SystemAccessControl >

```
class SaasSystemAccessControl(val statsd: DatadogMetricsClient) extends SystemAccessControl
  with UserAccessControl
  with CatalogAccessControl
  with SchemaAccessControl
  with TableAccessControl
  with ViewAccessControl
  with DatadogReporter
  with Logging
```

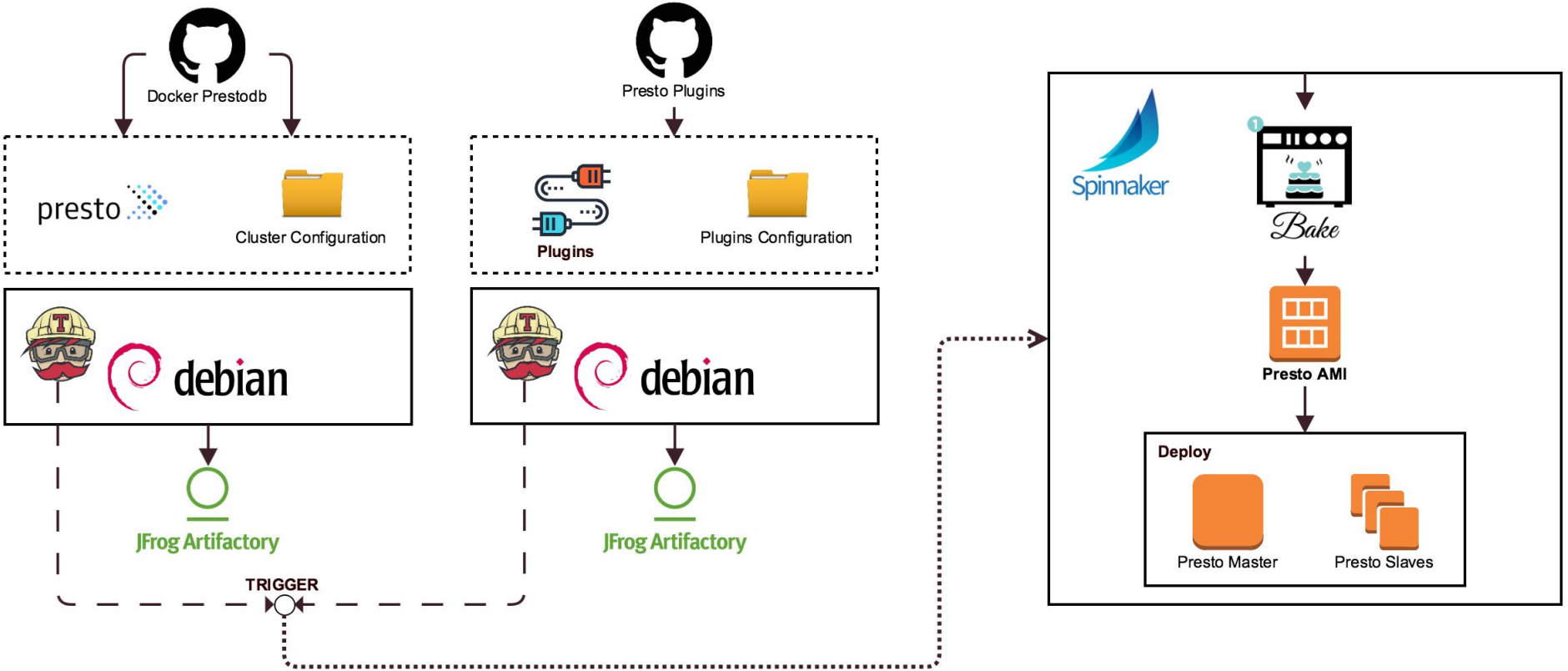
Traceability - SPI plugins < Event Listener >

```
import com.facebook.presto.spi.eventlistener._

class SaasEventListener(statsd: DatadogMetricsClient) extends EventListener with Logging {

  override def queryCreated(queryCreatedEvent: QueryCreatedEvent): Unit = {
    val context = queryCreatedEvent.getContext

    val tags = Map(
      User -> context.getUser,
      PrincipalName -> context.getPrincipal.orElse(Undefined),
      Source -> context.getSource.orElse(Undefined),
      Catalog -> context.getCatalog.orElse(Undefined),
      Schema -> context.getSchema.orElse(Undefined),
      Resource -> context.getResourceGroupName.orElse(Undefined)
    )

    statsd.increment("query.created", tags)
  }
  ...
}
```

Traceability - Datadog Dashboards



CI / CD


Docker Prestodb
presto
Cluster Configuration
Presto Plugins
Plugins
Plugins Configuration
debian
debian
JFrog Artifactory
JFrog Artifactory
TRIGGER
Spinnaker
Bake
Presto AMI
Deploy
Presto Master
Presto Slaves

CI / CD - Env Properties

```
# etc.template/config.properties.template.https
{%- if HTTPS_ENABLED is defined and HTTPS_ENABLED %}
http-server.https.enabled=true
http-server.https.port={{HTTPS_PORT}}
http-server.https.keystore.path=/opt/keystore/keystore.jks
http-server.https.keystore.key=presto
{% endif %}

{%- if AUTHENTICATION_TYPE is defined %}
http-server.authentication.type={{AUTHENTICATION_TYPE}}
{% endif %}
```

Env variables defined in the Spinnaker Pipeline

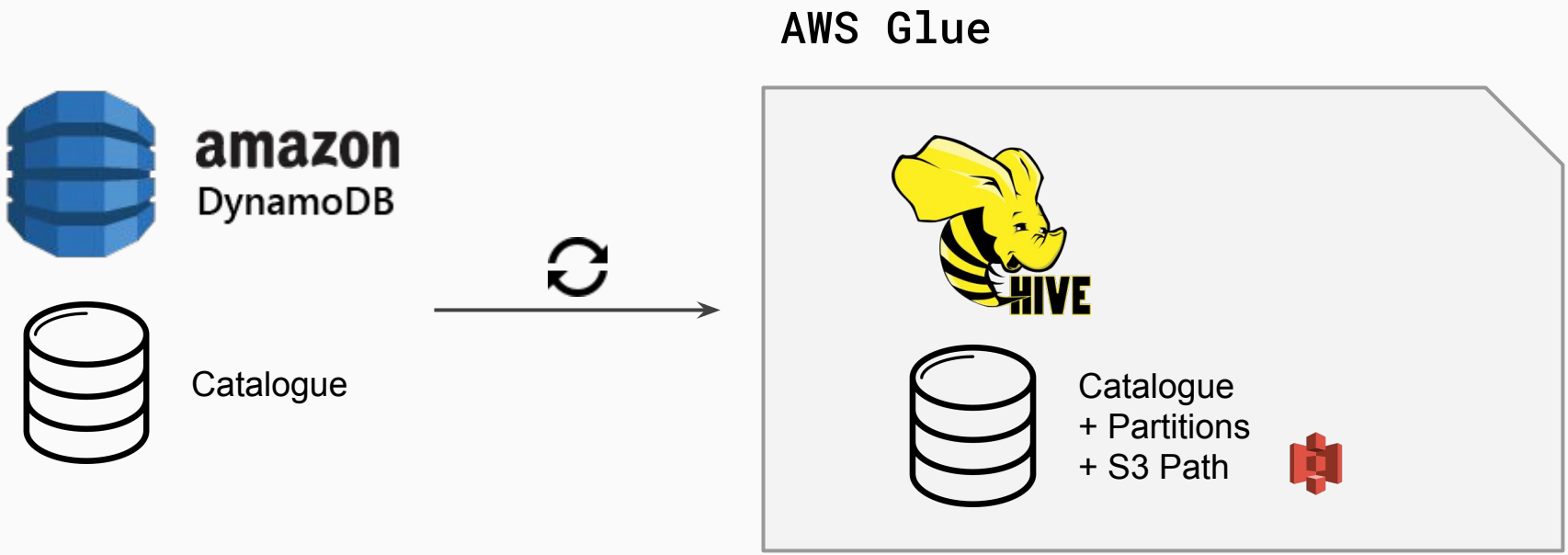
```
# etc.template/config.properties.template
presto.version={{PRESTO_VERSION}}.schibsted
{% include "config.properties.template.coordinator" %}
{% include "config.properties.template.http" %}
{% include "config.properties.template.https" ignore missing %}
...
```

Metastore - Aws Glue

```
# etc/catalog/hive.properties
hive.metastore=glue
hive.metastore.glue.region=eu-west-1
hive.metastore.glue.assume-role=arn:aws:iam::0000000000:role/presto
```

PR: *github.com/prestodb/presto/pull/10864*

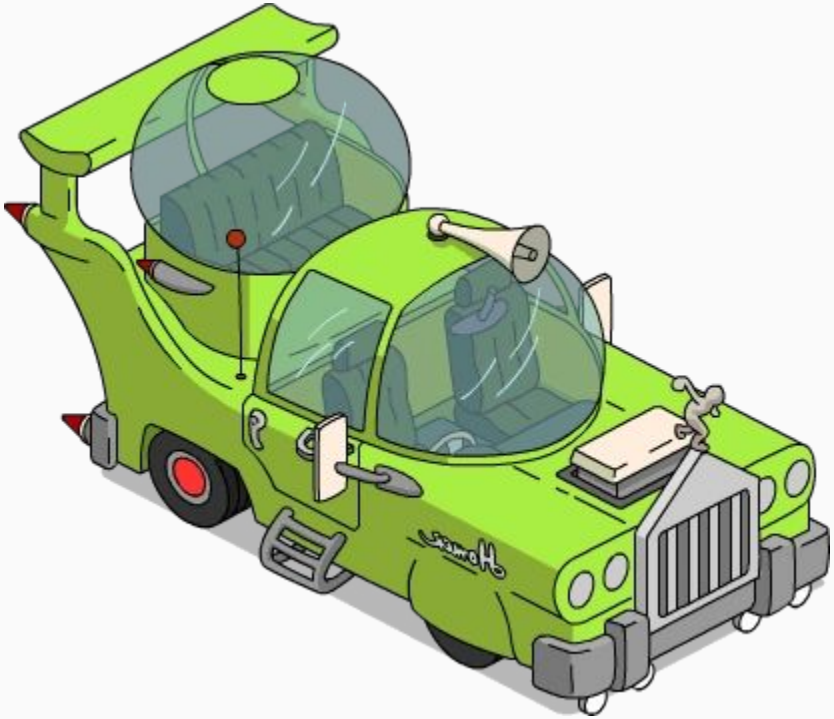
Metastore - Aws Glue - Catalogue Synchronizer



Metastore - Aws Glue - Catalogue Synchronizer

```
class Synchronizer(dynamoSynchronizer: DynamoSynchronizer, glueSynchronizer: GlueSynchronizer) {

  def execute(): Unit = {
    //- 1. Get Status from DynamoDB
    val metaCatalogue: MetaCatalogue = dynamoSynchronizer.getDynamoDbMetastore

    //- 2. Get Status from Glue (Schemas, Tables & Partitions)
    val glueCatalogue: GlueCatalogue = glueSynchronizer.getGlueMetastore(metaCatalogue.schemas.map(_.id))

    //- 3. Update Glue Metastore
    //- 3.1. Create Missing Schemas
    glueSynchronizer.createMissingSchemas(metaCatalogue, glueCatalogue)
    //- 3.2. Create Missing Tables
    glueSynchronizer.createMissingTables(metaCatalogue, glueCatalogue)
    //- 3.3. Update outdated Tables
    glueSynchronizer.syncExistingTables(metaCatalogue, glueCatalogue)
    //- 3.4. Update table partitions
    glueSynchronizer.syncTablePartitions(metaCatalogue)
  }
}
```

Data



Data

Data: Schema Hell

```
"object": {
  "id": 123,
  "location": "barcelona"
},
"sch:custom": "{ 'this': 'json'},
"@type": "event",
"ref-v2": "something"
```

```
"object": {
  "ID": "123",
},
"sch:custom": { "this": "json"},
"@type": "event"
```

```
"object": {
  "id": "123",
  "location": {"lat", "long"}
},
"sch:custom": { "this": "json"},
"@type": "event",
"ref-V2": "something"
```

```
"schema-hell": {
  "id": "123",
  "location": "barcelona"
},
```

Data: Hard Rules / Soft Rules

```
"event_id": "ABC-1234",
"provider": "sdrn:fotocasa",
"timestamp": "2018-09-14T01:02Z",
"type": "view",
"object": {
  "id": 123,
  "type": "classified_ad",
  "category": "loft"
},
"tracker": {
  "type": "JS",
  "version": "1.2.3.4-alpha",
},
"location": {
  "lat": 10.0,
  "long": 10.0
},
"ref_v2": "something"
```

Ad Views x Category

```
"provider": "sdrn:fotocasa",
"timestamp": "2018-09-14T01:02Z",
"type": "view",
"object": {
  "id": 123,
  "type": "classified_ad",
  "category": "loft"
}
```

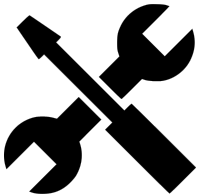
NumOff Events x version

```
"event_id": "ABC-1234",
"provider": "sdrn:fotocasa",
"timestamp": "2018-09-14T01:02Z",
"tracker": {
  "type": "JS",
  "version": "1.2.3.4-alpha"
}
```

Summary

SQL on S3 Data is Super Cool and gives value from minute 1

Configuring Presto to your needs is possible, if you have capacity



No magic tool will help you get gold of your crappy data

Links of Interest

Accessing S3 data through SQL with Presto

bit.ly/access-data-presto

BigData SQL Query Engine benchmark

bit.ly/query-engine-benchmark

Latest Presto News & Articles

starburstdata.com/newsletter