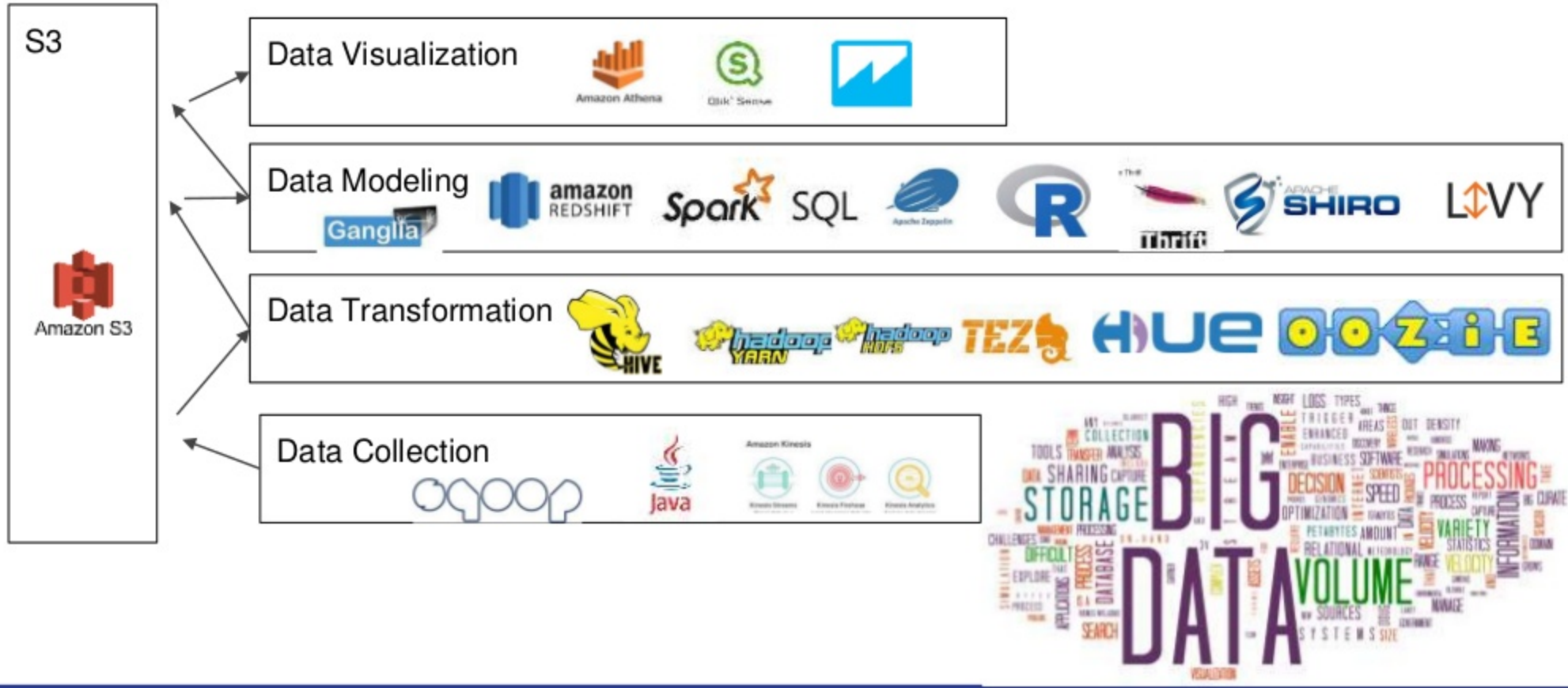

AWS Big Data Demystified #3
Spark SQL,Zeppelin,Livy,SparkR,Ganglia
And more… [Rstudio , Thrift, shiro] @ EMR
Omid Vahdaty, Big Data Ninja

TODAY'S BIG DATA APPLICATION STACK PaaS and DC...

Big Data Generic Architecture | Summary



Before we start… Bonus features

Q&A from last sessions

1. **Redshift VS EMR** https://amazon-aws-big-data-demystified.ninja/2018/06/03/when-should-we-emr-and-when-to-use-redshift/
2. **EMR cost reduction** https://amazon-aws-big-data-demystified.ninja/2018/06/09/massive-cost-reduction-on-aws-emr/
3. **Athena Cost reduction [and TPCH demo]** https://amazon-aws-big-data-demystified.ninja/2018/06/03/cost-reduction-on-athena/

Agenda for today...

- Zeppelin + SparkSQL [tpch demo]
- Thrift + SparkSQL
- Performance @ spark SQL
- [Zeppelin, Shiro, Livy] @emr
- R + spark R + Livy
- Ganglia + DEMO

Zeppelin + Spark SQL Demystified

AWS Big Data demystified

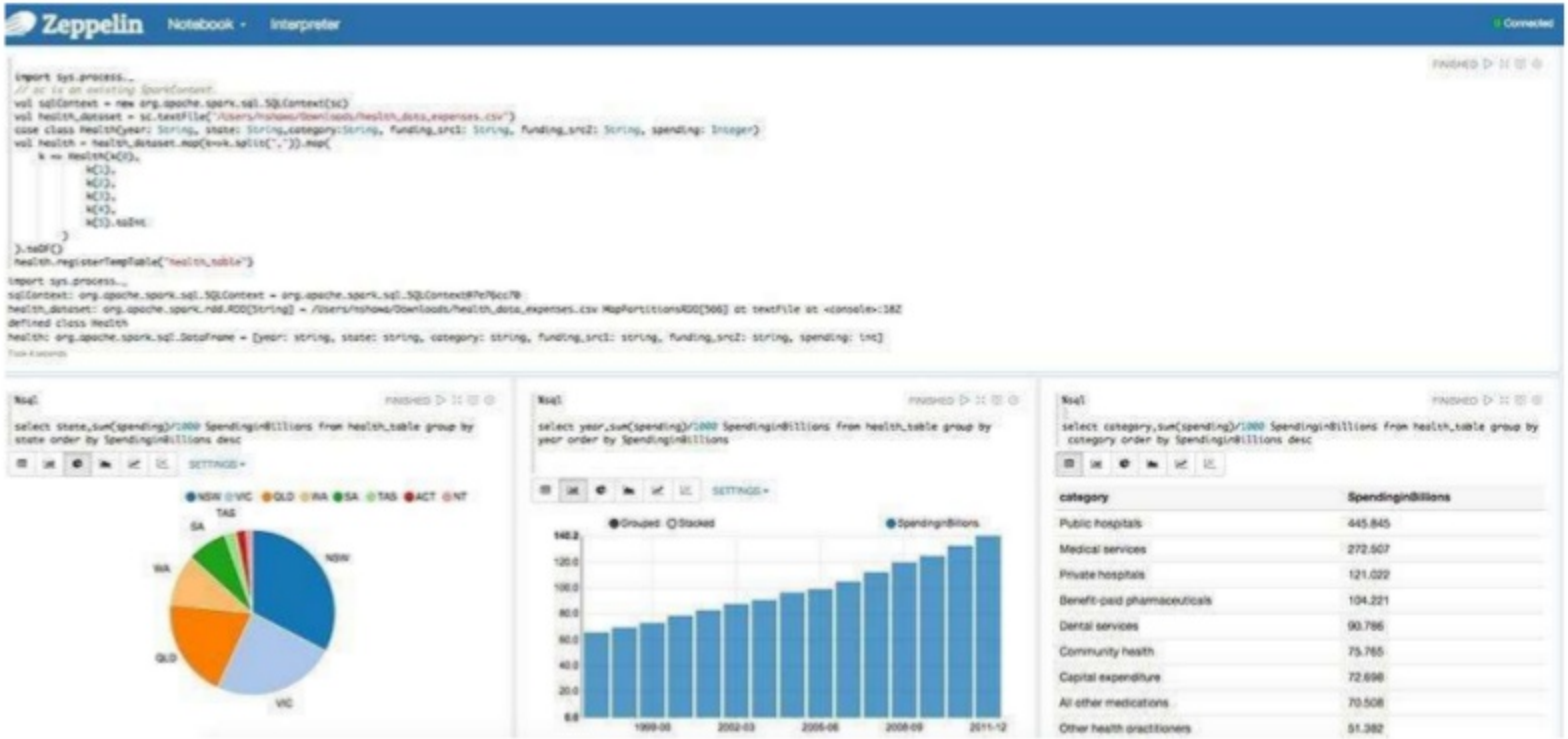
Omid Vahdaty, Big Data Ninja

Agenda

- Demo
  - Open cluster
  - Zeppelin TPCH demo
  - Interpreter setting
- What is Zeppelin?
- What is Spark SQL?
- Motivation?
- Features?
- Performance?
- Demo?

Zeppelin

A completely open web-based notebook that enables interactive data analytics. Apache **Zeppelin** is a new and incubating multi-purposed web-based notebook which brings data ingestion, data exploration, visualization, sharing and collaboration features

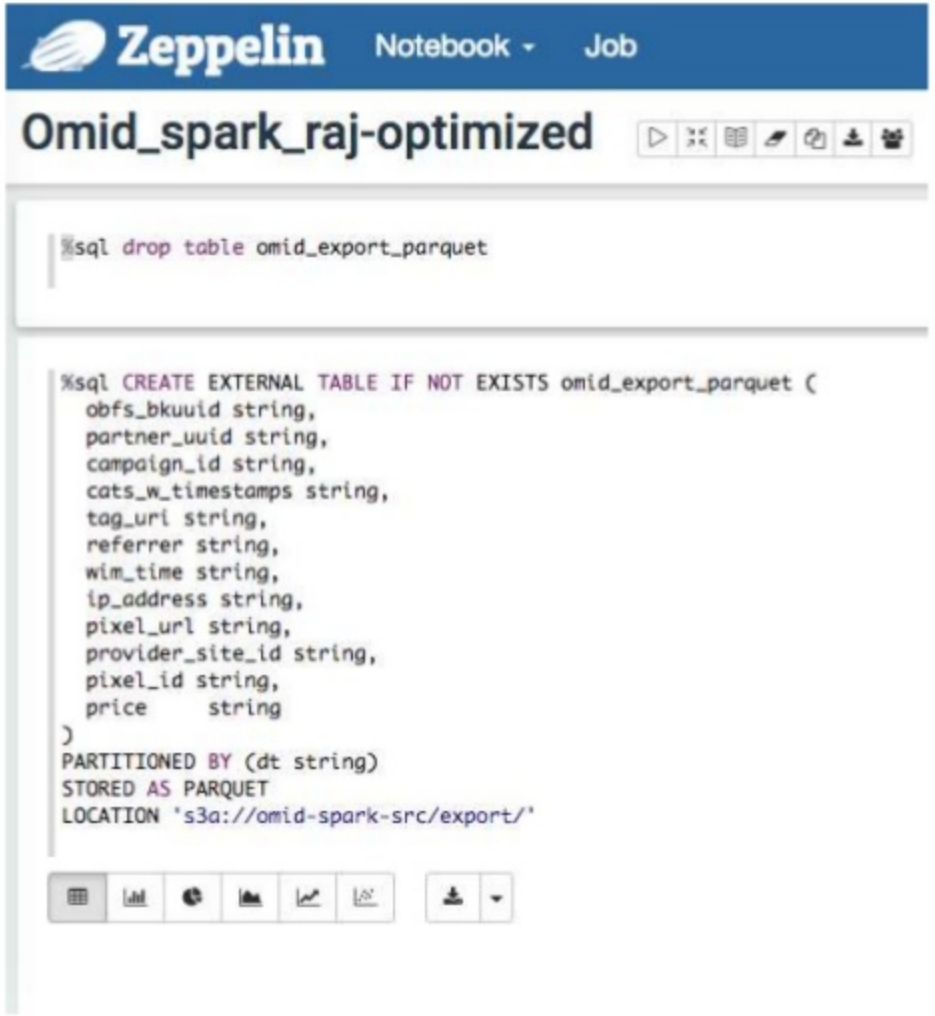


Zeppelin out of the box features

- Web Based GUI.
- Supported languages
  - SparkSQL
  - PySpark
  - Scala
  - SparkR
  - JDBC (Redshift,Athena, Presto,MySql ...)
  - Bash
- Visualization
- Users, Sharing and Collaboration
- Advanced Security features
- Built in AWS S3 support
- Orchestration

What is Spark SQL

- **Spark SQL** is a **Spark** module for structured data processing. Unlike the basic**Spark** RDD API, the interfaces provided by **Spark SQL** provide **Spark** with more information about the structure of both the data and the computation being performed. Internally, **Spark SQL** uses this extra information to perform extra optimizations.
- HiveSQL



Why Spark SQL?

- Simple
- Scalable
- Performance - faster than Hive
- External tables on S3
- Cost Reduction
- Decrease the GAP between Data Science and Data Engineering: HiveQL for ALL
- Get us one step closer to use sparkR / pyspark/ scala
- JDBC connection enabled via thrift server.
- Concurrency via Yarn Scheduler :)
- Join is runs better here than hive. [still not redshift]

Why Not SparkSql?

- Buggy
- Not as fast as scala
- Not code <----> SQL
- Known issues:
  - Performance over S3 → room for improvement
  - Insert Overwrite → overwriting all the partitions
  - Chunk size control → bug?
  - Dynamic partitions… non trivial
  - Beeline client/server version mismatch (CLI)

Why SparkSql + Zeppelin

- Sexy **Look and Feel** of any SQL web client
- **Backup** your SQL easily automatically via S3
- **Share** your work
- **Orchestration** & **Scheduler** for your nightly job
- Combine system **CLI commands + sql + visualization.**
- Advanced **Security** features.
- Combine all the DB's you need in **one place** including data transfer.
- Get **one step closer to spark and scala**
- **Visualize** your data easily.

Performance of Spark SQL+zeppelin

- EMR is already **pre-configured** in terms of spark configurations:
  - **spark.executor.instances (--num-executors)**
  - **spark.executor.cores (--executor-cores)**
  - **spark.executor.memory (--executor-memory)**
- X10 faster than hive in select aggregations
- X5 faster than hive when working on top of S3
- Performance Penalty is greatest on
  - Insert overwrite
  - Write to s3

Spark SQL | JDBC | Thrift | SSL How to...

- https://amazon-aws-big-data-demystified.ninja/2018/06/07/how-to-connect-via-jdbc-to-spark-sql-emr-on-aws/
- https://amazon-aws-big-data-demystified.ninja/2018/06/07/securing-spark-jdbc-thrift-connection-ssl-aws-emr/
- Excellent article to improve performance on thrift and spark sql (join included):
- https://community.microstrategy.com/s/article/Best-Practices-for-Spark-Thrift-Server-on-YARN

Performance Testing -- data transformation

- Observations
  - Penalty on s3 write
  - No Penalty on S3 read even if uncompressed
  - Compression is not always good...

Read/Write from aws s3	Hive	Spark SQL
Aggregation query	10 min	1 min
Text Gzip → Parquet	10 min	~2 min
Text Gzip → Parquet gzip	10 min	~18 min
parquet → Parquet-gzip		~2 min
Parquet-gzip → Parquet-gzip		~2 min

How to tune performance on Spark

- Good read: https://amazon-aws-big-data-demystified.ninja/2018/06/07/spark-performance-tuning/
- Another good read about dynamic resource allocation:
- https://amazon-aws-big-data-demystified.ninja/2018/06/07/here-is-example-to-demonstrate-how-to-work-with-maximizeresourceallocation-and-spark-dynamicallocation/
- Read about how to tune via configuration, and test carefully - **may have unexpected impact.**

DO NOT use CTAS

- Do not use create as select,
  - As the default behaviour is to write to **local HDFS**
  - use create (**with parquet, and compression)** and then insert separately

Future work

- Spark SQL with Tachyon (cache layer)
- https://www.oreilly.com/ideas/accelerating-big-data-analytics-workloads-with-tachyon
- https://amazon-aws-big-data-demystified.ninja/2018/06/07/aws-s3-caching-while-working-with-hive-spark-sql-and-external-table-llap/

Take away message to performance challenges

- Using Chunk size has minimal impact on performance. But helps on parallelism.
- Use compression
  - [in the create table definition]
  - Choose compression algorithm carefully
- Using S3DistCP - is
  - Slower than direct write to s3 with compression.
  - Makes you want to kill yourself when you work with dynamic partitions.
- Bottom line performance takeaways
  - Check compress when transforming Gzip text file → parquet
  - Use Auto scaling + Spot instances
  - R instances are good our use case as you lose about 50% of machine RAM on overhead.
  - Read about how to tune via configuration, and test carefully - may have unexpected impact.

EMR Zeppelin & Livy & shiro

AWS Big Data demystified

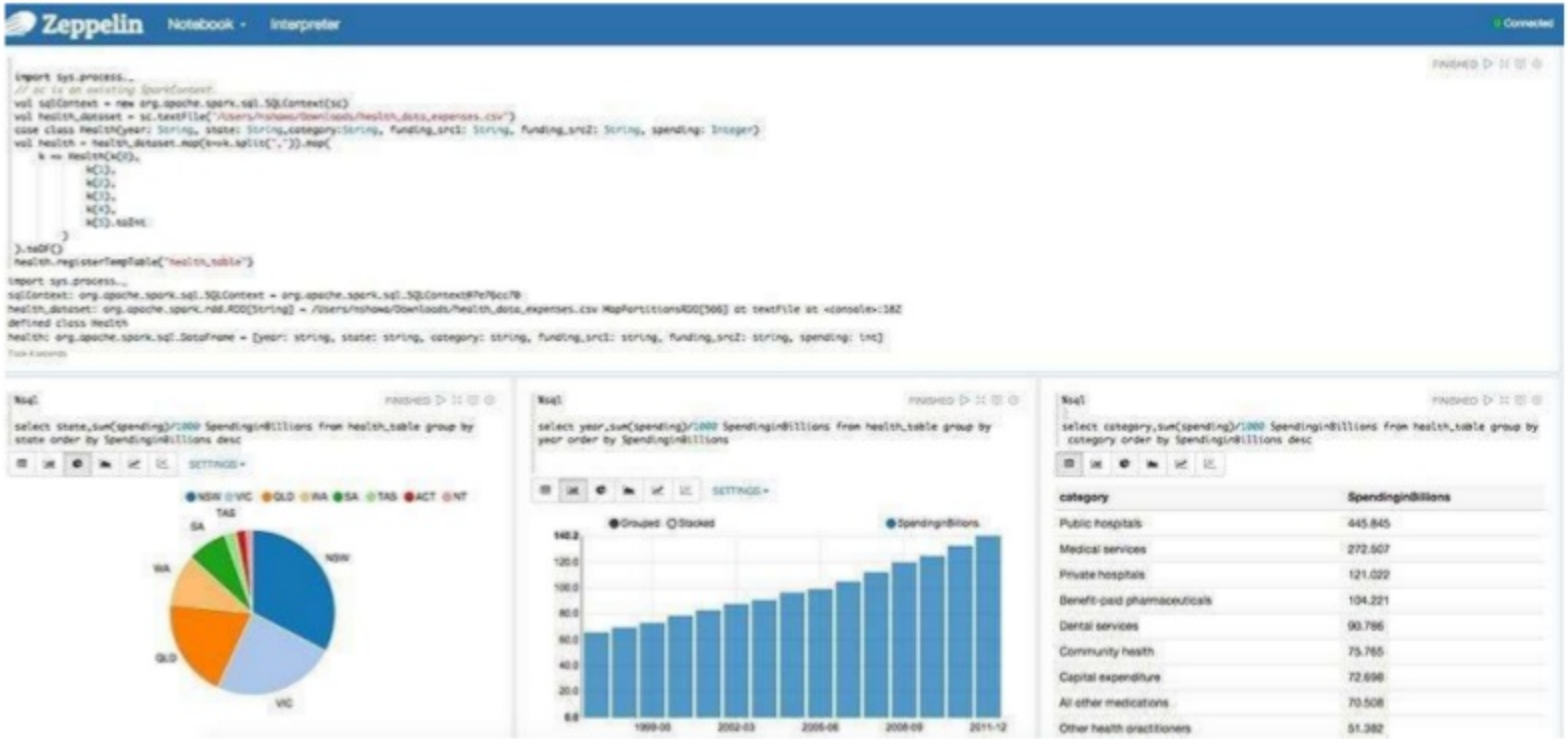
Omid Vahdaty, Big Data Ninja

Agenda

- What is Zeppelin?
- Motivation?
- Features?
- Performance?
- Demo?

Zeppelin

A completely open web-based notebook that enables interactive data analytics. Apache **Zeppelin** is a new and incubating multi-purposed web-based notebook which brings data ingestion, data exploration, visualization, sharing and collaboration features



Why Zeppelin?

- Sexy Look and Feel of any SQL web client
- Backup your SQL easily automatically via S3
- Share and collaborate your notebooks
- Orchestration & Scheduler for your nightly job
- Combine system commands + sql + scala spark visualization.
- Advanced Security features
- Combine all the DB’s you need in one place including data transfer.
- Get one step closer to pyspark and scala and sparkR
- Visualize your data easily.

EMR Zeppelin Interpreter

- The concept of Zeppelin interpreter allows any language/data-processing-backend to be plugged into Zeppelin. Currently, Zeppelin supports many interpreters such as Scala (with Apache Spark), Python (with Apache Spark), Spark SQL, JDBC, Markdown, Shell and so on.
- SparkContext, SQL context , Zeppelin cotext Z SparkContext, SQLContext and ZeppelinContext are automatically created and exposed as variable names sc, sqlContext and z, respectively, in Scala, Python and R environments. Staring from 0.6.1 SparkSession is available as variable spark when you are using Spark 2.x.
- https://zeppelin.apache.org/docs/latest/manual/interpreters.html

Zeppelin Binding modes

1. In ***Scoped* mode,** Zeppelin still runs **single** interpreter **JVM** process but **multiple *Interpreter*** *Group* serve each Note
2. In ***Shared* mode**, **single JVM** process and **single *Interpreter*** *Group* serves all Notes.
3. ***Isolated* mode** runs **separate interpreter** process **for each Note**. So, each Note have absolutely isolated session.

Binding modes

spark %spark, %sql, %dep, %pyspark, %r

Option

The interpreter will be instantiated Globally in shared process.

- Globally
- Per Note
- Per User

☐ Connect to existing process

☐ Set permission

Binding modes

spark %spark, %sql, %dep, %pyspark, %r

Option

The interpreter will be instantiated Per Note in scoped process.

- scoped per note
- isolated per note

Connect to existing process

Set permission

Binding modes - share mode

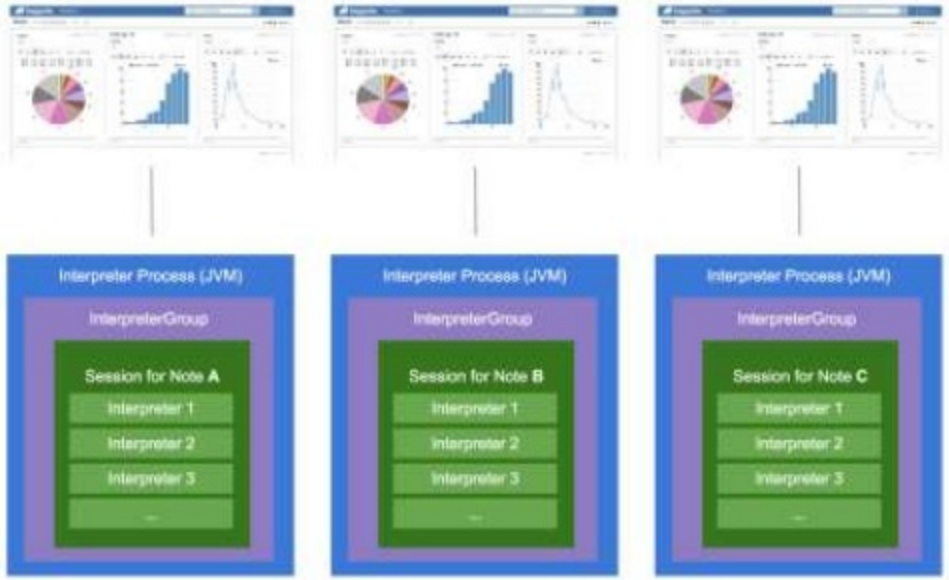
In **Shared** mode, single JVM process and a **single session serves all notes.** As a result, `note A` can access variables (e.g python, scala, ..) directly created from other notes..

Binding modes - scoped mode

In **Scoped** mode, Zeppelin still runs a **single interpreter JVM** process but, in the case of per note scope, **each note runs in its own dedicated session**. (Note it is still possible to share objects between these notes via ResourcePool)

Binding modes - Isolated mode

Isolated mode runs a separate interpreter process for each note in the case of **per note** scope. So, each note has an absolutely isolated session. (But it is still possible to share objects via ResourcePool)



When to use each binding mode?

- **Isolated** means **high utilization of resources but less availability to share options to share objects**
- In **Scoped mode**, each note has its own Scala REPL. So variable defined in a note can not be read or overridden in another note. However, a single SparkContext still serves all the sessions. And all the jobs are submitted to this SparkContext and the fair scheduler schedules the jobs. **This could be useful when user does not want to share Scala session, but want to keep single Spark application and leverage its fair scheduler.**
- In **Shared mode,** a SparkContext and a Scala REPL is being shared among all interpreters in the group. **So every note will be sharing single SparkContext**

Import/Export Notebooks

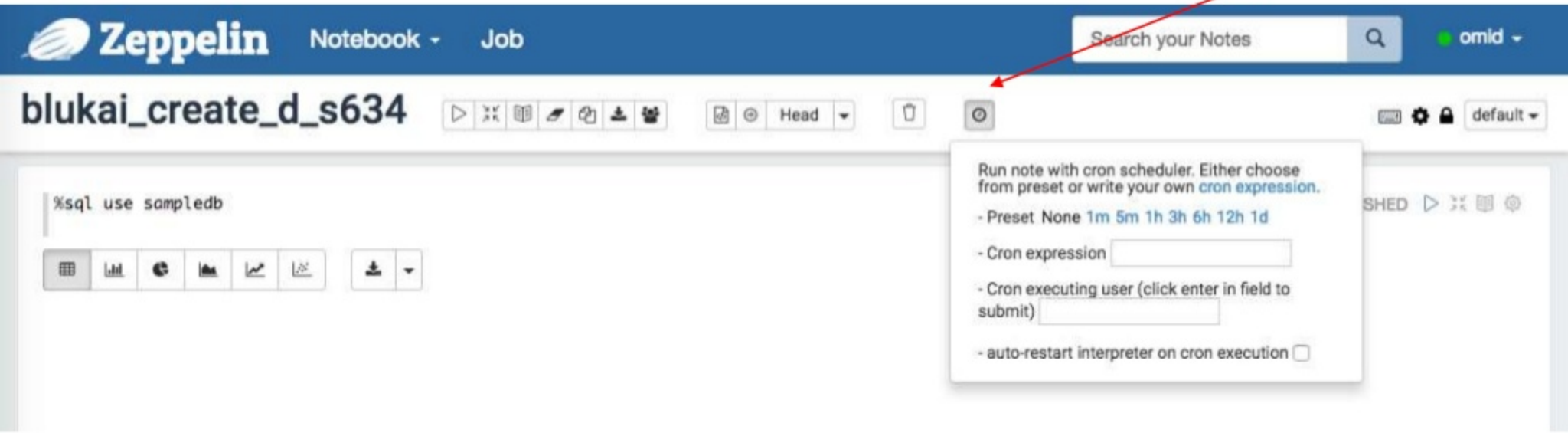
- U can import /export notebooks into from Url, local disk or Zeppelin Storage: S3 and GIT
- Zeppelin storage s3 notes.
  - Need to import from local disk the first time
  - U can use roles to provide access to S3 instead of access key / secret key
  - Each notebook is saved on s3 in a specific path (see docs)
  - Can't open directly from S3- bug?
  - Yes, you can use encryption of S3…
  - https://aws.amazon.com/blogs/big-data/running-an-external-zeppelin-instance-using-s3-backed-notebooks-with-spark-on-amazon-emr/

Advanced Security

- Shiro
  - LDAP
  - User management
  - Notebook sharing to group of users
  - Kerberos
  - Permissions for notes
  - Resources explainations: https://amazon-aws-big-data-demystified.ninja/2018/06/07/emr-zeppelin-secuirty/
- Zeppelin HTTPS / SSL
  - SSH tunnel
  - Add HTTPS to the gui , step by step and some resources: https://amazon-aws-big-data-demystified.ninja/2018/06/07/emr-zeppelin-secuirty/

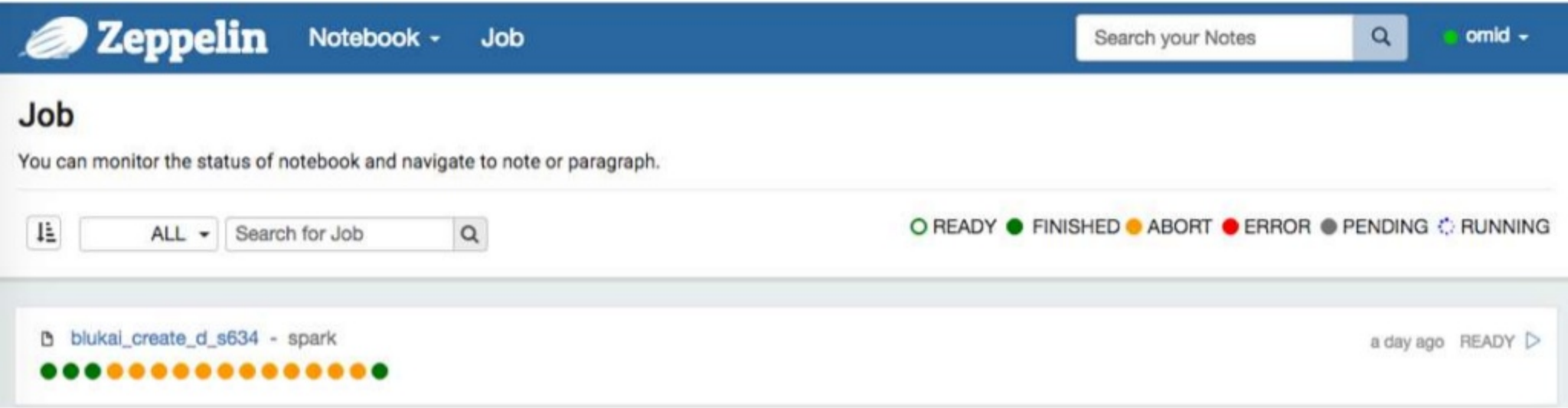
Orchestration & Scheduling

You can go to any Zeppelin notebook and click on clock icon to setup scheduling using **CRON**. You can use this link to generate the CRON expression for the time interested - http://www.cronmaker.com/.



Orchestration & Scheduling

You can ran any job if our have permission and see their status



bootstrapping Zeppelin in an EMR STEP

https://amazon-aws-big-data-demystified.ninja/2018/06/08/bootstrapping-zeppelin-emr/

Zeppelin - enable API HTTPS

https://amazon-aws-big-data-demystified.ninja/2018/06/08/accessing-emr-zeppelin-api-via-ssl-https/

Apache Livy

rest api to manage spark jobs

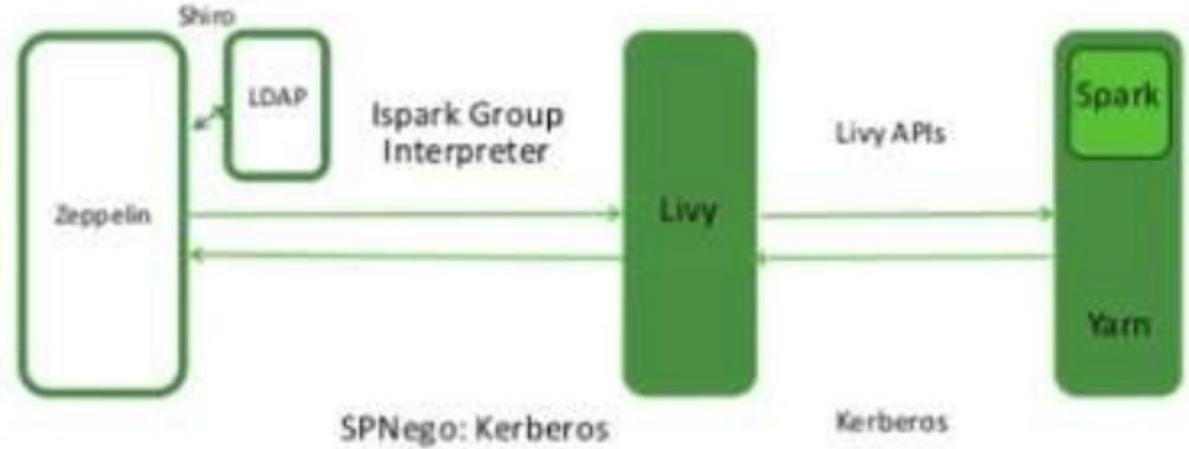
- Interactive Scala, Python and R shells
- **Batch submissions** in Scala, Java, Python
- **Multi users** can share the same zeppelin server (impersonation support)
- Can be used for **submitting jobs from anywhere with REST**
- Does not require any code change to your programs

Livy + Zeppelin use case

Multi tenant users/jobs:

- Sharing of Spark context across multiple Zeppelin instances.
- When the Zeppelin server runs with **authentication enabled, the Livy interpreter propagates user identity** to the Spark job so that the job runs as the originating user. This is especially useful when multiple users are expected to connect to the same set of data repositories within an enterprise.

Livy + Zeppelin Architecture



Rstudio & remote SparkR cluster

Omid Vahdaty, Big Data Ninja

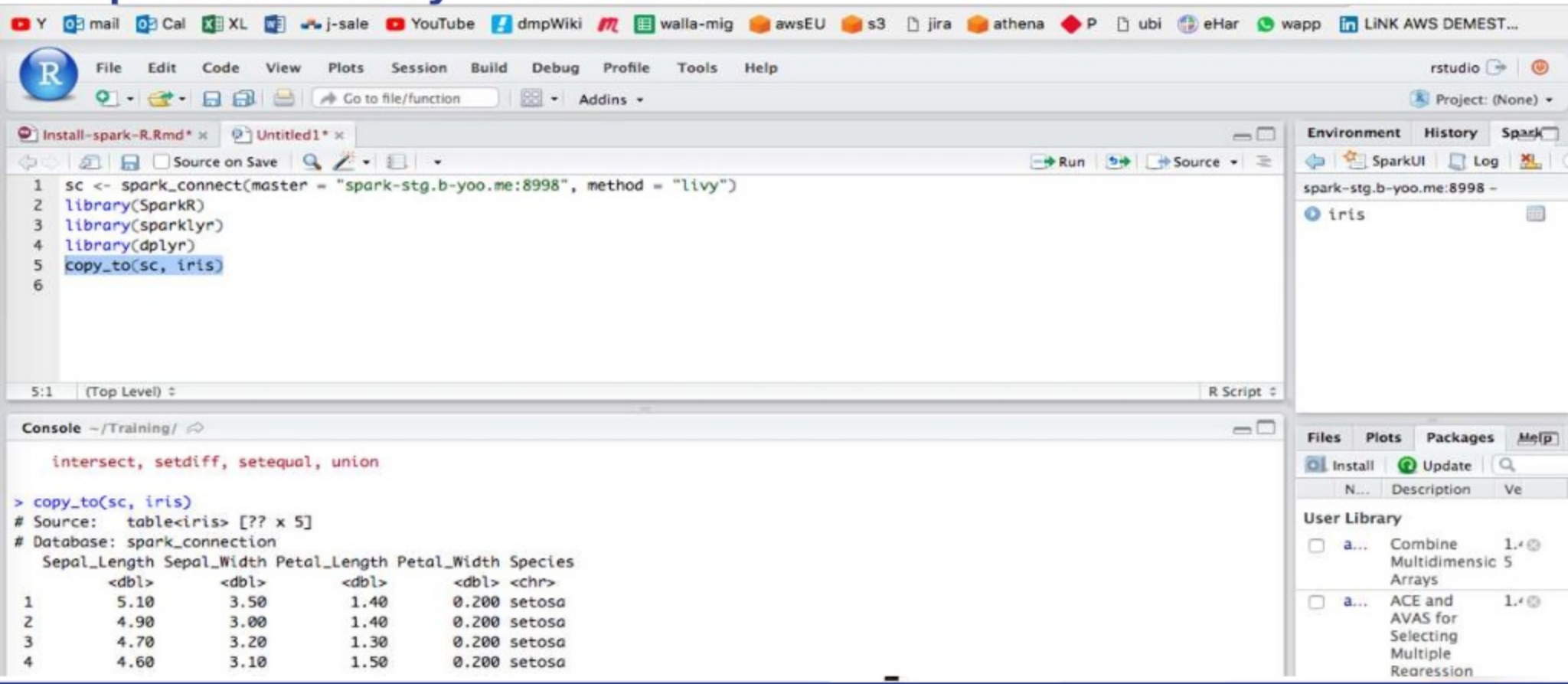
Spark R+ Livy + R studio

You needs to follow the below steps to properly install RStudio server, SparkR, sparklyr, and finally connecting to a spark session within a **remote** EMR cluster:

Spark R+ Livy + R studio steps by step

https://amazon-aws-big-data-demystified.ninja/2018/06/08/working-with-r-studio-and-a-remote-spark-cluster-spark-r/

Spark R+ Livy + R studio



Spark R+ Livy + R studio

LIVY Sessions

Interactive Sessions

Show 10 entries Search:

Session Id	Application Id	Owner	Proxy User	Session Kind	State	Logs
0				spark	idle	session
1				spark	idle	session

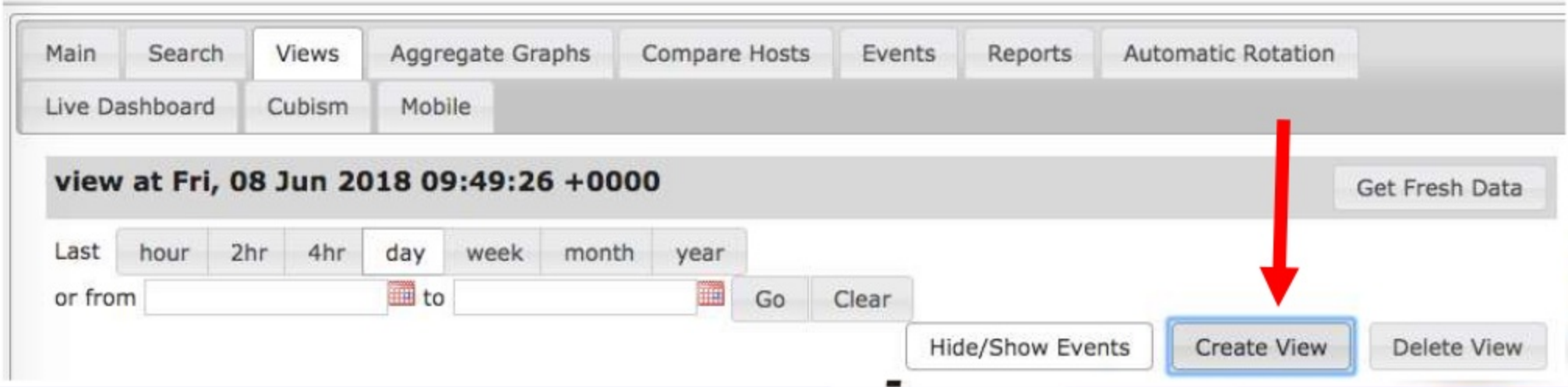
Showing 1 to 2 of 2 entries Previous 1 Next

EMR and Ganglia - Yarn Monitoring

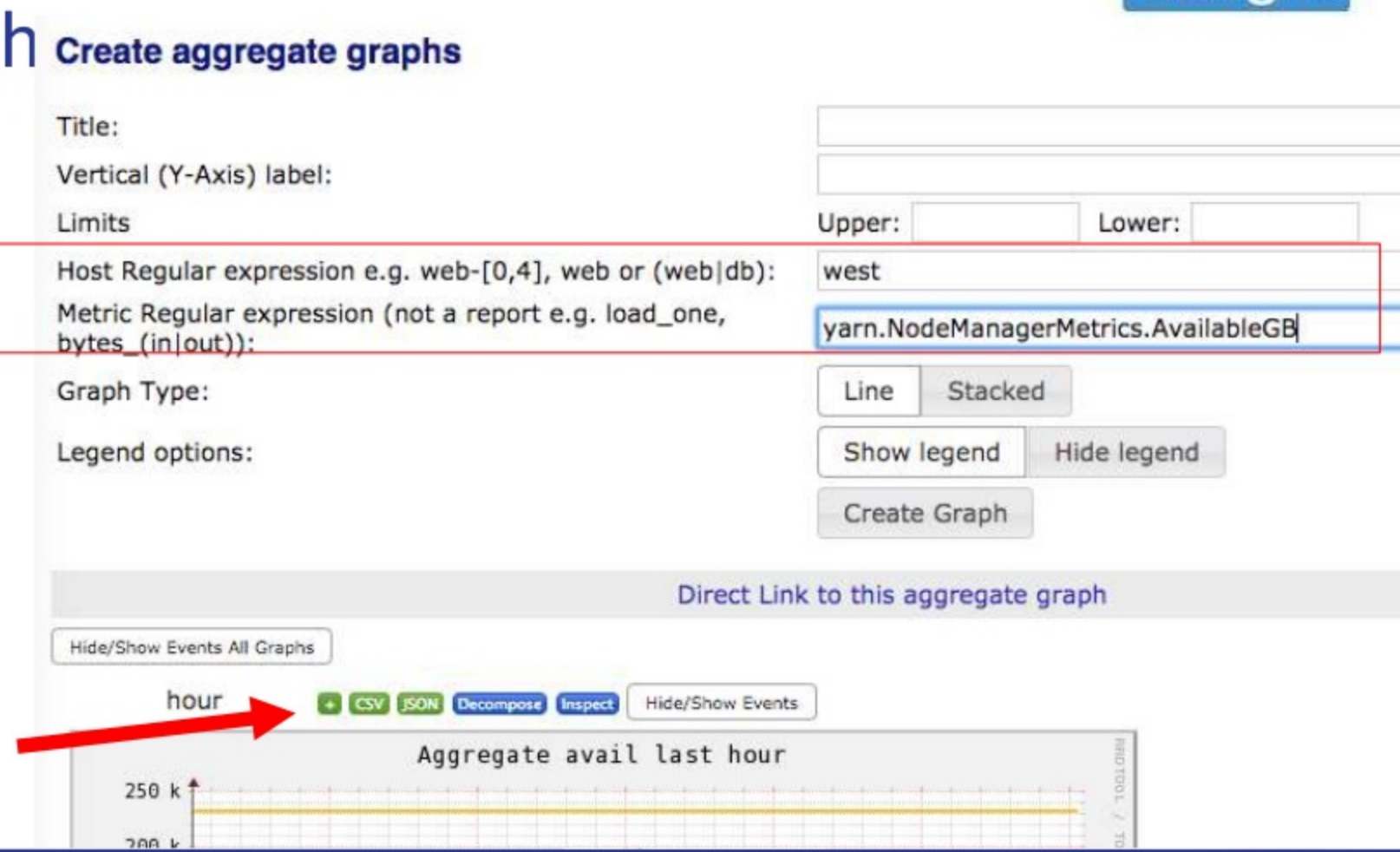
Omid Vahdaty, Big Data Ninja

Create View

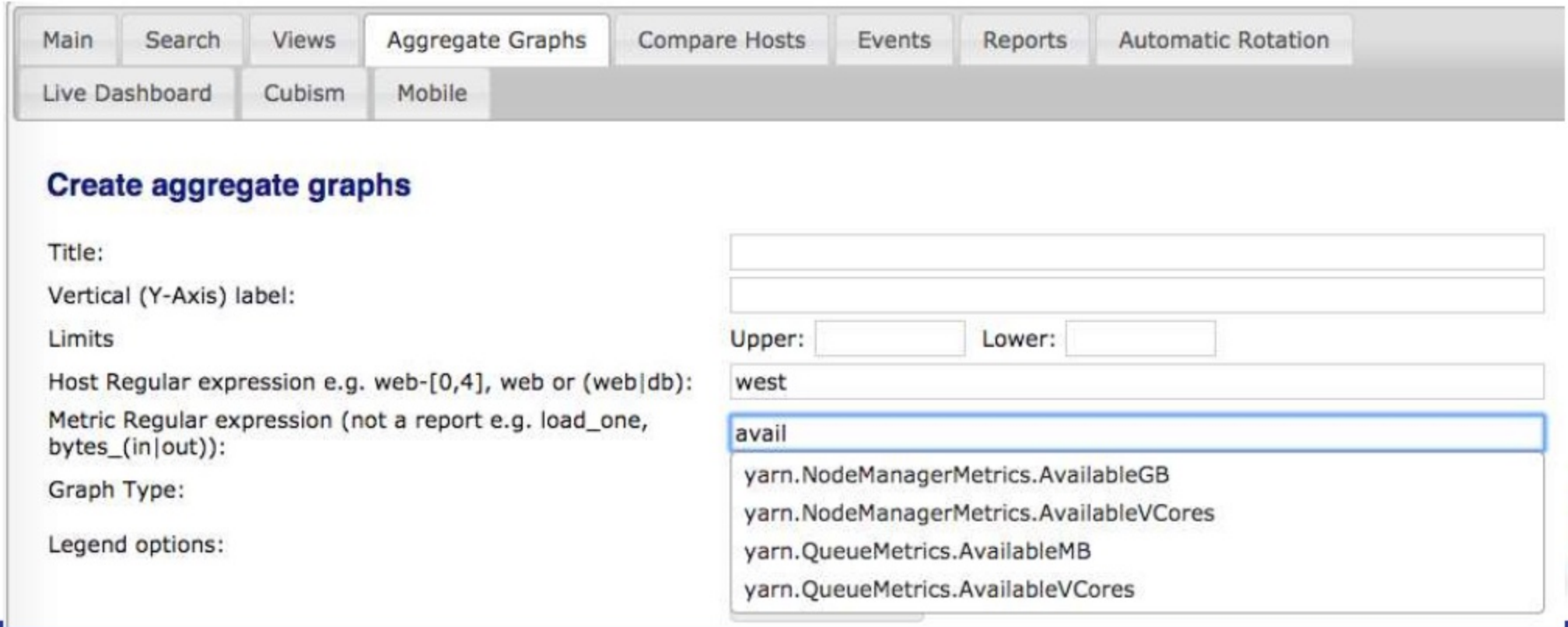
- View is like a dashboard
- Then add graphs via “Aggregate Graphs” tab



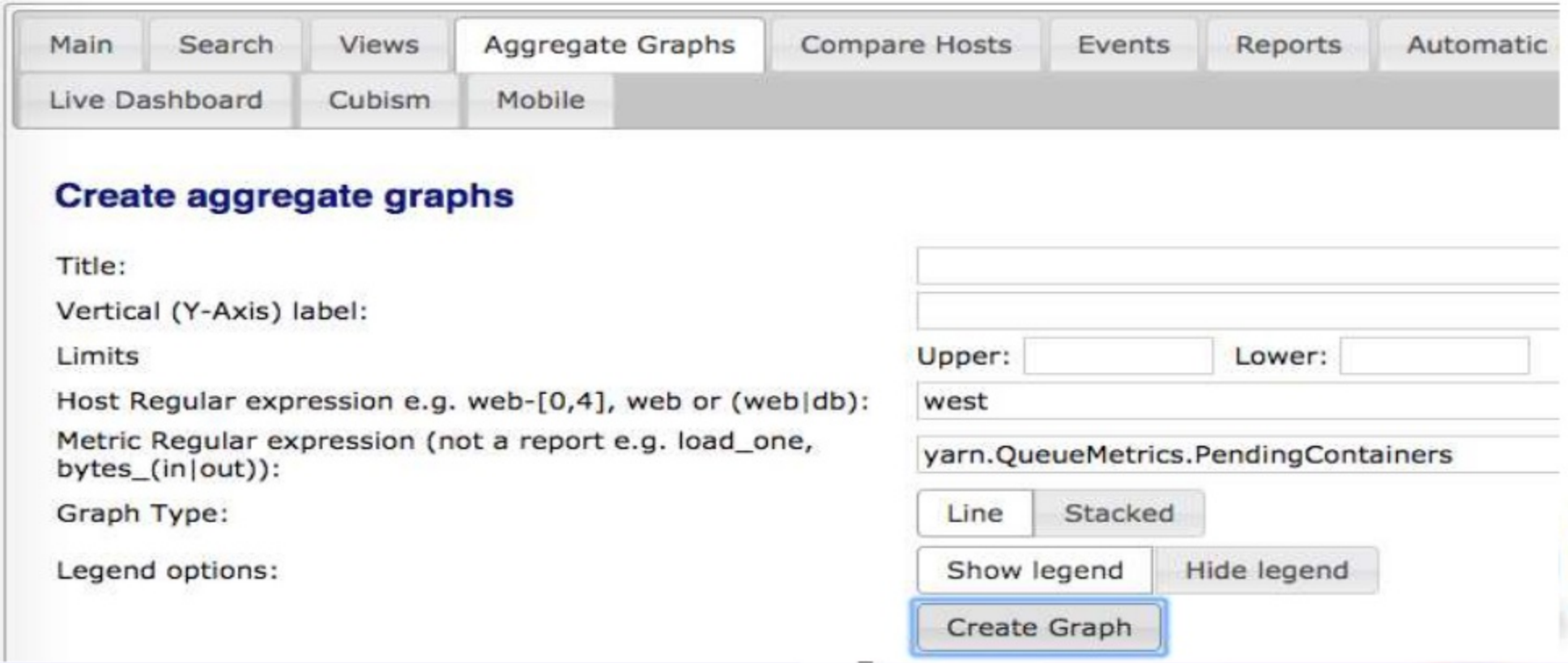
Add graph


Create aggregate graphs
Title:
Vertical (Y-Axis) label:
Limits
Upper:
Lower:
Host Regular expression e.g. web-[0,4], web or (web|db):
west
Metric Regular expression (not a report e.g. load_one, bytes_(in|out)):
yarn.NodeManagerMetrics.AvailableGB
Graph Type:
Line
Stacked
Legend options:
Show legend
Hide legend
Create Graph
Direct Link to this aggregate graph
Hide/Show Events All Graphs
hour
CSV
JSON
Decompose
Inspect
Hide/Show Events
Aggregate avail last hour
250 k
200 k

Aggregate Graphs - available



yarn.QueueMetrics.PendingContainers



Stay in touch...

- Omid Vahdaty
- +972-54-2384178
- https://amazon-aws-big-data-demystified.ninja/
- Join our meetup, FB group and youtube channel
  - https://www.meetup.com/AWS-Big-Data-Demystified/
  - https://www.facebook.com/groups/amazon.aws.big.data.demystified/
  - https://www.youtube.com/channel/UCzeGqhZIWU-hIDczWa8GtgQ?view_as=subscriber