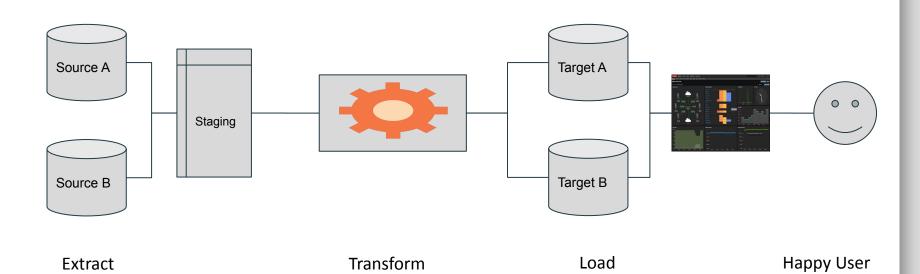
# Addressing Challenges of ETL Using Apache Spark

# About us

- We are a leading ride hailing service provider in the middle east touching millions of lives everyday.
- To improve customer satisfaction, we have enhanced data driven decision making.
- Our in-house ETL solution makes heavy use of Apache Spark to handle complex real world problems.
- I'm one of the lead contributors of our in-house solution to address the needs and demands of the ETL.

# ETL: From Exosphere

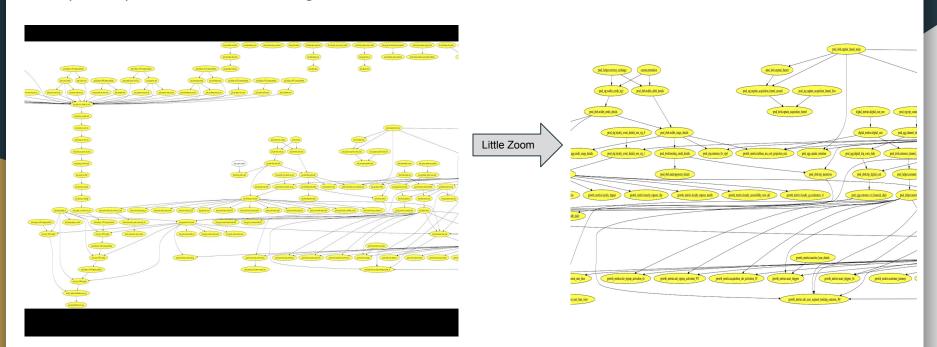


# ETL: In practice

- 1. Any process can fail
  - a. Trying to load into a table or read from a location which does not exist
  - b. Resources not enough for the process
  - c. Network issue or any unpredictable error
- 2. On failure a process can get manually triggered multiple times
- 3. In addition of running different process on different queues, different process can also be running over different clusters
- 4. Components can get added/removed/modified
  - a. New source/target added or existing removed
  - b. Sql file updated to support new KPI or column change
  - c. New mechanism of alerting/monitoring added
- 5. Not all sources are extracted at the same frequency, some involve batch processing while some need real time treatment
- 6. Security is needed at every stage
- 7. Maintaining data quality

# ETL: In practice

Complex Dependencies in Processing/Transformation



# Challenges of creating an ETL Solution

- 1. Supporting batch and streaming use cases
- 2. Early detection of failure or delay
- 3. Less manual intervention (Ideally zero)
- 4. Extensible in supporting new sources/targets/processing
- 5. Easy to be used by the end users
- 6. Quick in adding new dependency and handling complex dependencies without affecting others
- 7. Intelligent in resource consumption
- 8. Scale to support large data processing

# Possible Solution

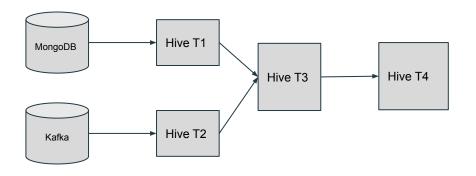
#### Using a scheduler like Airflow with SubDAGs

- Pros
  - i. Handles dependencies with ease
  - ii. Supports task retries to reduce failures due to temporary issues
  - iii. With Subdags, logically same entity can be grouped together
- Cons
  - i. Not intuitive for running streaming jobs continuously
  - ii. Need to build application for each
    - 1. Source
    - 2. Target
    - 3. Processing logic
  - iii. High code redundancy when dealing with event sourcing, monitoring, alerting and any other feature common to all types of applications
  - iv. Need different service to keep track of the freshness of loaded data (ie. how recent was it successfully loaded)

# Shakti: Our In-House ETL PowerBooster

- Shakti means power
- Shakti powers our ETLs, handling complex dependencies of different types of user tasks at ease and at scale
- Spark-Scala based ETL boiler plate to:
  - Support variety of user tasks (Eg: Hive Task, MongoDB Task, Redshift Task etc.)
  - Provide optimisations across all the tasks and dependencies
  - Leverage the power of Spark and added customizations just by passing:
    - a json file containing parameters
    - optionally a configuration file (containing spark configurations)
    - a sql file(if it is a spark sql application)
  - Support various stages of ETL Pipeline: Ingestion till Presentation

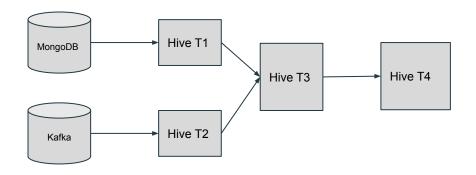
# Shakti: How it works



#### What information is needed:

- The location and name of MongoDB table/collection
- 2. The location and name of Kafka topic
- 3. The names of the hive tables T1 and T2
- 4. The query to load T3 from T1 and T2
- 5. The query to load T4 from T3
- 6. The dependency that: T3 depends on T1 and T2. Also, T4 depends on T3

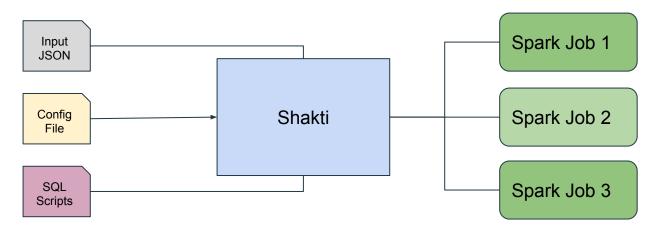
# Shakti: How it works



#### How to create this pipeline with Shakti:

- 1. Give the details in the JSON.
- 2. Some tasks are related ingestion, while some are transformations. Shakti takes care of all
- 3. Some tasks are batch and some are streaming, Shakti runs them all
- 4. The dependencies are optimised at run time
- 5. For hive tables, the partitions are swapped (since we use S3)
- 6. Fresh tasks do not execute
- 7. Parallelize the tasks, if they can be
- 8. No duplicate tasks run together
- 9. Dependencies can be across multiple JSONs
- 10. Blacklisted tasks and its children are skipped

# Shakti: How it works



Spark jobs are launched in parallel decided by user-defined config and following the dependencies in the JSON.

# Shakti: Tasks

- Tasks are extensible.
- Just extend the Executable trait and start using SparkSession inside.
- Override **exec()** method to provide what task should do
- Some of the existing tasks are:
  - Hive Task
  - MongoDb Task
  - Parquet Writer Task
  - Redshift Task
  - Compaction Task
  - Merge Task
  - External User Task

# Shakti: How JSON looks like

```
"task_id": "current_task",
                                                                    If that is not the case, current task will:
                                         For the
                                                                           Either, wait on the execution of
                                         current task to run,
                                                                           parent task (if the parent task is
                                         parent task must
                                                                           defined in the same ison)
                                         have succeeded in
                                                                           Or, time out (if the parent task is
"dependenciesWithMeta": {
                                         the last 1 hour
                                                                           defined in a different ison)
       "parent task": {
              "freshnessCriteriaInHours": 1}},
                                      If the current task has succeeded in the last 1 hour,
"sql": "location of sql file",
                                      it will be considered fresh and will not execute in the
"sla": 1800,
                                      current run
"task_type": "hive",
"post_sql": "sql for Data Quality",
"freshnessCriteria": 1,
```

# Shakti: How application conf looks like

```
execution {
  number-of-threads = 2 ; Controls the number of spark jobs to run in parallel
  slack-notification-endpoint = "https://hooks.slack.com/services/TXXXXX/XXXXX/XXXXXX"
task {
  kafka = {
   brokers = "kafka101:9093,kafka102:9093,kafka201:9093,kafka202:9093,kafka301:9093,kafka302:9093"
    producer = { ... } ; Configs to apply on kafka producer
spark {
  application.name = XXXX
  config { ...; configs to be contained by the SparkContext
    spark.dynamicAllocation.initialExecutors = 4 : One example
```

# Shakti: How to run

#!/usr/bin/env bash

--executor-memory ... \
--executor-cores 4 \

--conf ... \
--files ... \

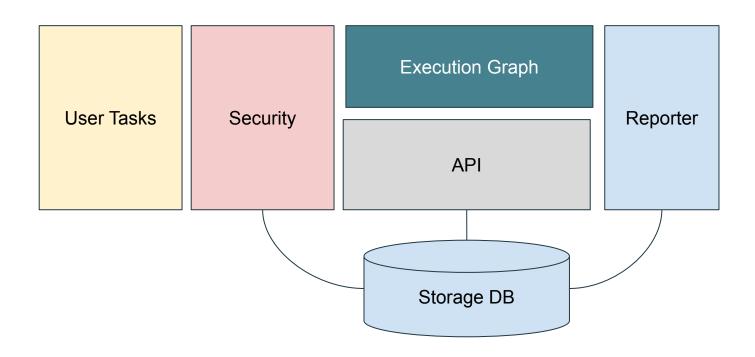
application\_config="path\_to\_application\_config"
tasks\_json="path\_to\_tasks\_json"
specific\_tasks="specific\_tasks\_to\_run"

spark-submit \
--packages ... \
--class shakti.spark\_sql\_etl.Main \
--jars ... \
--driver-memory ... \
--num-executors ... \

--driver-java-options="-Dconfig.file=\${application\_config}" \

shakti-assembly\*.jar \${tasks json} \${specific tasks}

# Shakti: Components



# Shakti: Why choose Spark

- Fast and performant in-memory processing
- Support to cache frequent data
- Powerful and elegant data unification, composing streaming, sql and RDD operations together in the same application
- Support in multiple languages, we use Scala
- Rapid development, fits we

# Shakti: Web API

```
\leftarrow \  \  \, \rightarrow \  \  \, \textbf{C} \quad \, \textbf{\textcircled{S}} \quad \text{shakti-backend.careem-engineering.com:} \\ 8899/\text{shakti/speculation/v1/status/blacklisted}
```

cli\_test\_1
prod\_agg.customer\_financial\_stats\_daily

# Shakti: Operations Alerting



#### slacktee APP 6:49 AM

-e --Airflow ETL Run--

Task Info: dag\_id=[DWH\_Production\_ETL\_Pipeline], task\_id=
[dwh\_external\_user\_task\_etl\_cluster\_1], state=[success]

-e --Airflow ETL Run--

Task Info: dag\_id=[DWH\_Production\_ETL\_Pipeline], task\_id=
[dwh\_fact\_tables\_1\_trip\_etl\_cluster\_4], state=[success]



#### RIP APP 6:19 PM

ETL failed notification

#### 10.0.63.78

Identifier: user360.booking\_aggregation\_to\_csv

Message: java.lang.RuntimeException: Caught Hive MetaException attempting to get

partition metadata by filter

StartTime: 2020-01-14T14:18:38.475Z

reportingTime: 2020-01-14T14:19:07.503Z @bdp\_oncall



#### RIP APP 10:49 AM

ETL Execution Skipped

#### 10.0.63.248

Identifier: EtlSkipped#prod\_agg.customer\_financial\_stats\_daily

Message: Task prod\_agg.customer\_financial\_stats\_daily has been skipped as chosen by

the owner

StartTime: 2020-01-15 06:49:20.9

reportingTime: 2020-01-15 06:49:20.9 @bdp\_oncall

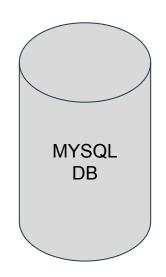
Owner: NA

#### Priority

High

# Shakti: Operations State Management

etl_meta_entries			
etl_id	int(11)		
task_id	varchar(512)		
etl_type	varchar(64)		
start_time	timestamp		
end_time	timestamp		
status	varchar(64)		

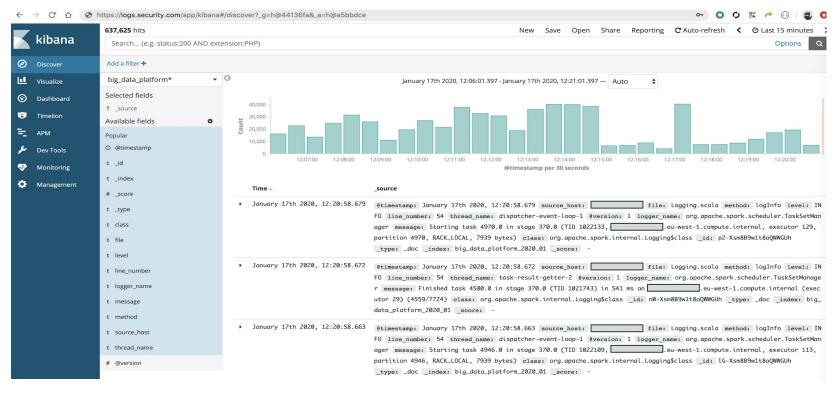


etl_speculation				
etl_id	int(11)			
task_id	varchar(512)			
requested_by	varchar(64)			
status	varchar(20)			
creation_date	timestamp			
expiration_date	timestamp			

# Shakti: Operations Event Sourcing

```
{
  "clusterName": "ip-10-0-62-120",
  "appName": "nrt_15.json",
  "executionId": "c90e4706-15eb-46d0-8ac1-84051a282b4b",
  "taskId": "nrt_avail_hr",
  "status": "succeeded",
  "message": "task succeeded",
  "ts": 1578563293559
}
```

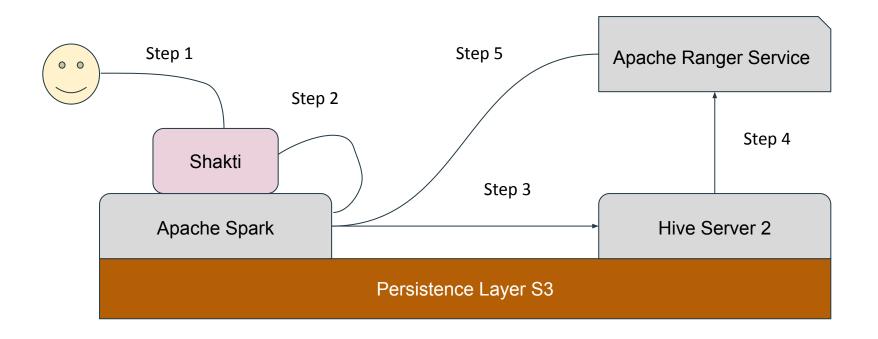
# Shakti: Operations Central Logging



# Shakti: Operations Monitoring Resource Utilisation

- Use NewRelic to monitor the resources requested by the application and the resources actually used by it.
- Graphana can also be used along with a time series DB like prometheus/graphite.
- Make use of a JMX exporter jar
  - -javaagent:/home/hadoop/newrelic/newrelic.jar

## Shakti: User Authentication and Authorization



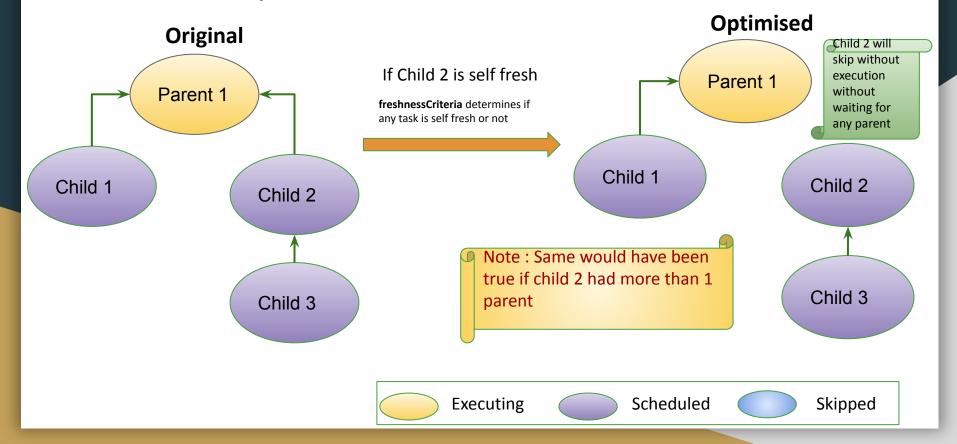
### Shakti: User Authentication and Authorization

- 1. User submits the query.
- 2. User query gets intercepted based by spark extension manager, this allow us to add custom logic to dispatch SQL query operation to hive using HiveAuthImpl.
- 3. HiveAuthorizer is invoked on the proposed LogicalPlan of SparkSQL transformation.
- 4. HiveAuthorizer then invokes RangerHiveAuthorizerFactory, this is an entry point of Out of the box Ranger support for Hive.
- 5. Authorization is performed and based upon the grant access user operation either succeeds or access-denied error is thrown with detailed message.

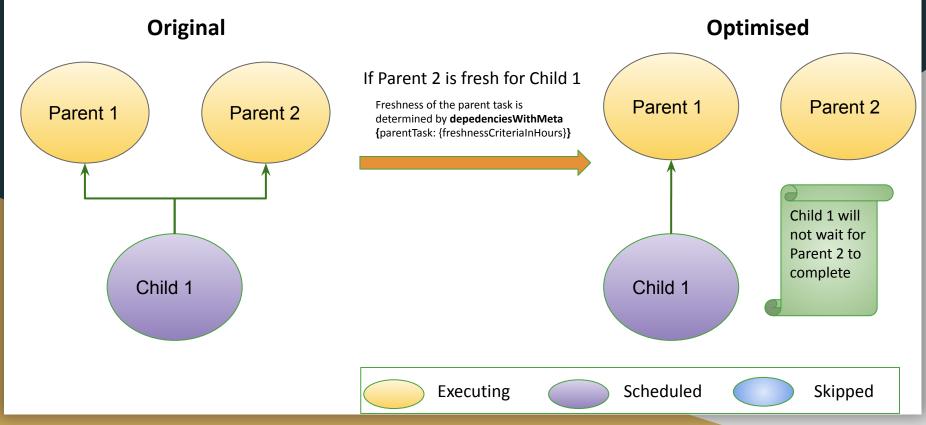
# Shakti: Optimisations

- Use of dynamic allocation of executors
  - o spark.dynamicAllocation.enabled = true
  - spark.shuffle.service.enabled = true
- Tuning parameters based on the requirements of the spark job
  - --driver-memory
  - --num-executors
  - --executor-memory
  - --executor-cores
- Making use of adaptive execution
  - We are currently using Spark 2.4.0
- Using different version of fileoutputcommitter to write to S3
  - o mapreduce.fileoutputcommitter.algorithm.version = 2
- Creating a swap directory in S3 for writing current data
  - Point the hive table location or the partition location to the swap location and prune it later
- Considering data freshness
- Restructure DAG

# Shakti: Optimisations



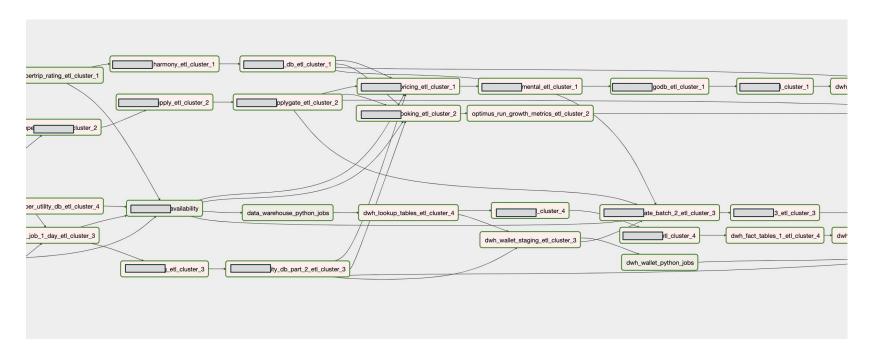
# Shakti: Optimisations



# Shakti: IAC on Cloud

A Disea edul formation aau fair to EDRT  Sigulation  Foreign Concerning interesting  subtreat, United Sigulation  Foreign Sigu	stack_id			
Table 1. Life to the control branches.  Individual 1. See that the section of the	oldon_lo	uud-formation stack ID for EMR.		m4.large
Suboric Claid  Figure 1. Suboric Claid Subor	key_name	BigMama		machine type for core instance.
Lear Lot		Key to ssh cluster instances.	Laste Continues a Long	
Beach    Configuration   Confi	subnet_id		task_instance_type	m4.large
original food organish floods or ground in charactering products on the product or ground in ground is empty, the plug needs clone the repository, before it can display the contents. This may take some time if you there a slow contention or repository is single.  The bases to base from the your time a slow contention or repository is single.  Stocked name of the machine where all AC eleaded configurations are stored.  The products of the machine where a many or time and the content of the machine where a many or time and the second configuration. The machine is producted under ore-demand crimina.  The products of the machine is the time of the machine where ranger is running or the machine where ranger is running or the machine where solving the policies and the machine where solving the policies of the machine where solving the machine where solving the machine where solving the machine where solving the policies of the machine where solving the policies of the machine where solving the machine where solving the policies of the machine where solving the policies	Branch	origin/f#f0824		machine type for task instance.
service_role emr-role		origin/ut-onboarding origin/#10091	core_instance_count	
The second published by the se				Number of instances in core instance group.
Solucial valence all ACP related configurations are stored.  Solucial valence all ACP related configurations are stored.  Propio  Entre Manager describer to external liver-metal-store.  Propio  Finance (Propios Sor cluster relatences. If this is set to zero [0] then relatence will be launched under on-demand criteria.  Propiosal valence (Park an agent and the machine where ranger is running  service_note  Propiosal valence (Park an agent and the machine where ranger is running  service_note  Propiosal valence (Park an agent and the machine where ranger is running  service_note  Propiosal valence (Park an agent and the machine where ranger is running  service_note  Propiosal valence (Park an agent and the machine where ranger is running  ranger_service  ranger_service  ranger_service  ranger_service  ranger_service  ranger_service  ranger frost  ranger_service  ranger_service  ranger frost  ranger_service  ranger_		The branch to build from	task_instance_count	2
erv prod prod prod prod prod prod prod prod	comiguration_bucket_mame			Number of instances in task instance group.
Prod Prod Prod Prod Prod Prod Prod Prod	metastore_secret_name		region	eu-west-1
pol_price  inv  constraints and cell by a price sport duster instance. If this is set to zero [0] then instance will be launched under on-demand criteria.  Bid prices for cluster instances. If this is set to zero [0] then instance will be launched under on-demand criteria.  Bid prices for cluster instances. If this is set to zero [0] then instance will be launched under on-demand criteria.  Bole for the machine service.  Role for the machine launched in cluster.  Spone separated additional security group lds  master_instance_type machine type for master instance.  machine type for		Secret Manager identifier for external hive-meta-store.		
spot_price	env	prod		AWS region
Spot_price 0.1  Big prices for cluster instances. If this is set to zero [0] then instance will be launched under on-demand criteria.  Bed prices for cluster instances. If this is set to zero [0] then instance will be launched under on-demand criteria.  Bole for function EMR as a service.  Pole for the machine launched in cluster.  Role for the machine launched in cluster.  Role for the machine launched in cluster.  Role for the machine launched in cluster.  Solir_host  Solir_host  Malarge  machine type for master instance.  machine type for the machine where ranger is running.  name of tanger service having the machine where solicies.  machine type for the machine type fo		env	ranger host	
service_role  mr-role Role for launched EMR as a service. Role for launched in cluster.  Role for the machine launched in cluster.  Role for the machine launched in cluster.  Role for the machine launched in cluster.  Solr_host  master_instance_type machine spee for master iestance.  machine spee for master iestance is the machine where solr is running  machine spee for master iestance.  machine spee for master iestance.  machine spee for master iestance.  machine spee for master iestance is the machine where solr is running  machine spee for master iestance.  machine spee for master iestance.  machine spee for master iestance is the machine where solr is running  machine spee for machine spee for master iestance.  machine spee for machine such as a special control of the machine where solr is running  machine spee for machine spee	spot_price	0.1	] Tangor_noor	
Role for faunched EMR as a service.  Role for the machine launched in cluster.  Role for the machine launched in cluster.  Role for the machine launched in cluster.  Solir_host  Solir_host  Malarge  machine lype for master instance. lype machine lype for master instance.  machine lype for master		Bid prices for cluster instances. If this is set to zero [0] then instance will be launched under on-demand criteria.		hostname of the machine where ranger is running
Role for funched EMR as a service.  Role for the machine Issunched in cluster.  additional_sgs  additional_sgs  master_instance_type machine type for master instance.  machine type for master	service_role	emr-role	ranger service	
Role for the machine launched in cluster.  additional_sgs additional_sgs Space separated additional security group ids master_instance_type machine type for master instance.  machine type for		Role for launched EMR as a service,	rangor_convice	
additional_sgs  Space separated additional security group ids  master_instance_type machine type for master instance.  emr_release_tag  emr_s2.2.0  EMB version.	job_flow_role			name of ranger service having the policies
Space separated additional security group ids  master_instance_type machine type for master instance.  m4.large machine type for master instance.  m4.large  machine type for master instance.  m4.large  m6.P5.22.0  EMB version		Role for the machine launched in cluster.	- T. Y A	
master_instance_type	additional_sgs		solr_host	
master_instance_type m4.large machine type for master instance.  machine type for master instance.  m4.large m6.large emr_felease_tag emr_fele		Space separated additional security group ids		
core_instance_type m4.large = m4.large = EMR_version	master_instance_type	m4.large		nostname of the machine where soir is running
FMR varion		machine type for master instance.	emr_release_tag	emr-5.22.0
machine type for core instance.		m4.large		EMD version
		machine type for core instance.		EMIN VEISION

## Shakti: With Airflow



# Shakti: Usage Report

- >150 big jobs running every hour powering all of organization's data team.
- Daily hands-on support to all verticals, including Warehouse, Data Science and BI teams.
- 2TB of data processed per hour.
- Near Real time Data lake in production with critical use cases.
- > \$250K cost reduction of the platform cost.
- Massive work on tech compliance.
- Users **running ~20K queries** per day through presentation layer.

# Shakti: Road Ahead

- Query cost computation: With feedback, decide at run time the resources to be allocated
- Maintaining current hot data in HDFS and background sync to S3
- Analyzing DeltaLake and EMRFS
- Web-based self service data platform, for enhanced UX
- Open source the project