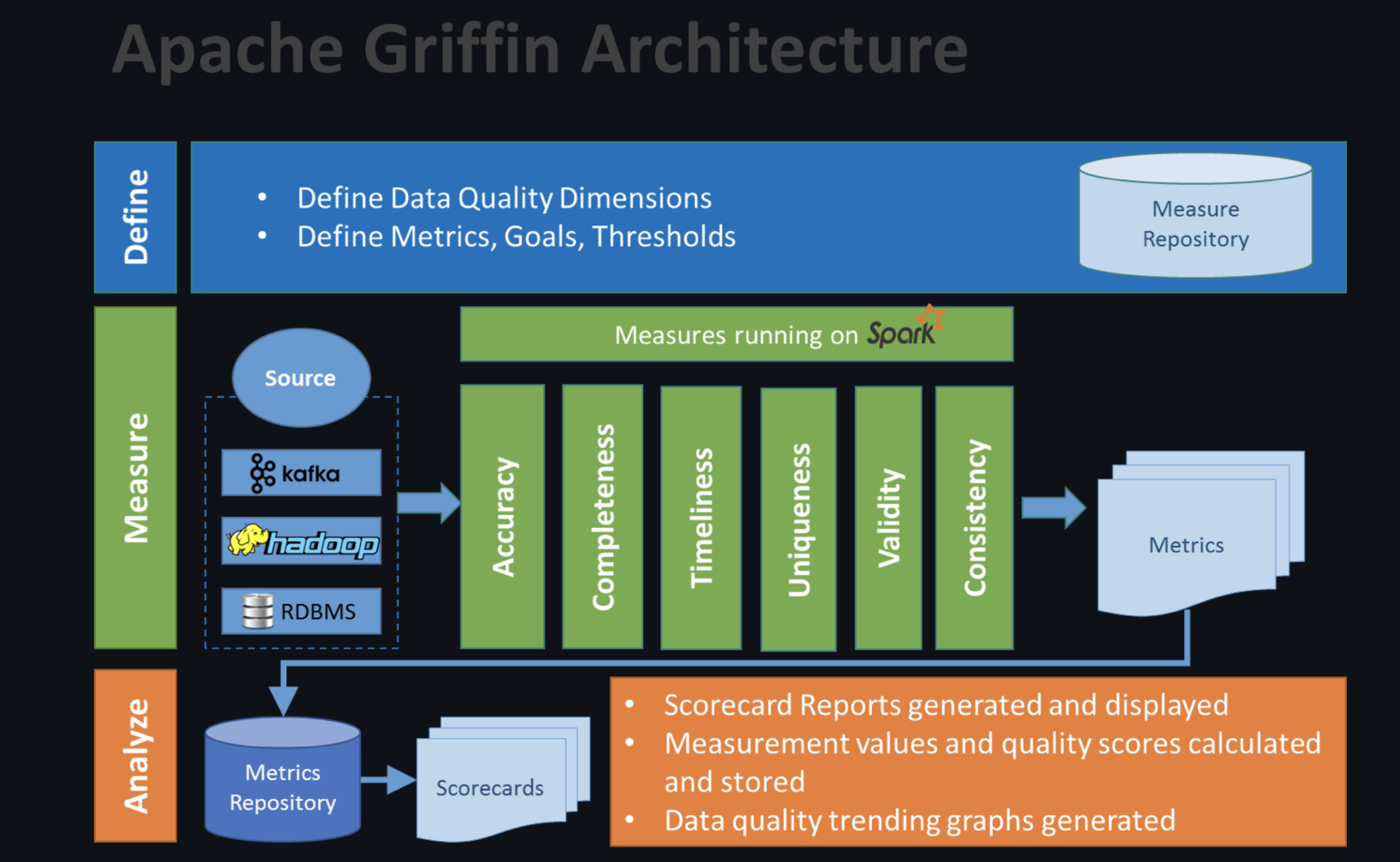
**Apache Griffin Runbook: Automatic Data Validation Tool integrated with Amazon EMR, Glue Catalog, Hive, Spark, Athena, MWAA(Airflow).**

**Overview:**

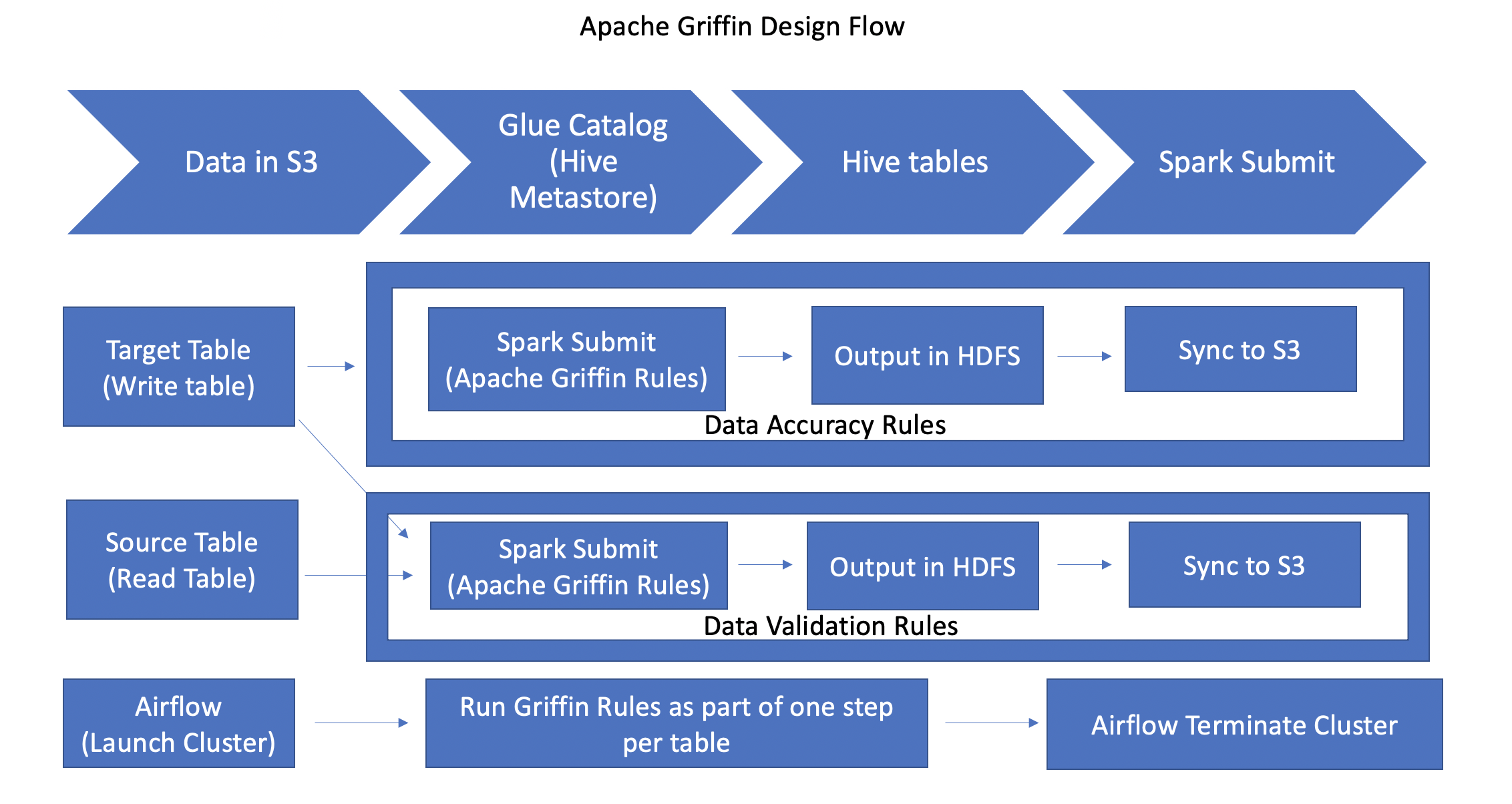
Apache Griffin is a Data Quality Service Platform (DQSP) built on top of Apache Hadoop and Apache Spark. It provides a comprehensive framework that processes different tasks like defining data quality model, executing data quality measurement, automating data profiling and validation, as well as an unified data quality visualization across multiple data systems. It aims to address challenges from data quality domain in big data applications.

When people dances with big data (Hadoop or other streaming systems), they will hit a big challenge, measurement of data quality. Different teams have built customized tools to detect and analyze data quality issues within their own domains. However, it's extremely possible to take a platform approach as commonly occurring patterns, which is what we think. As such, we are building a platform to provide shared Infrastructure and generic features to solve common pain points of data quality. This would help us to build trusted data assets.

**Architecture Diagram:**

****

**Customized Architecture as per GoDaddy Requirement:**



**Environment Preparation**

1. You need to prepare the environment for Apache Griffin measure module, including the following software:
2. Launch EMR cluster with Below Services.

* Spark
* Hive
* Glue Catalog/RDS Metastore.
* Use Glue Catalog for Hive/Spark.
* Optional Services:
  + - Maven
    - Athena
    - Airflow (MWAA)
    - QuickSight

1. Build Apache Griffin Measure Module

* Download Apache Griffin source package [**here**](https://downloads.apache.org/griffin/0.6.0/)**.**
* Unzip the source package.
* unzip griffin-0.6.0-source-release.zip
* cd griffin-0.6.0-source-release
* Build Apache Griffin jars.
  + mvn clean install

1. Move the built apache griffin measure jar to your work path.

* mv measure/target/measure-0.6.0.jar <work path>/griffin-measure.jar

1. For Simplification, I have built the jar with all the changes needed. I have put that griffin-measure.jar on GIT.
   1. **You can download the jar from below link and you can skip steps 3 and 4. If you don’t want to take latest version and work on stable version which is working fine.**

* GoDaddy Git Link 🡪 [Click me](https://github.secureserver.net/GDLakeDataProcessors/util/blob/dev-private/Griffin_Artifacts/griffin-measure.jar)

1. Define data quality measure
   1. Apache Griffin env configuration
      1. Create environment config file: **env.json**
         1. Check Sample file on GIT: [**env.json**](https://github.secureserver.net/GDLakeDataProcessors/util/blob/dev-private/Griffin_Artifacts/env.json) **🡪** This file contains information about where you want to store results on HDFS. You can configure results here.
   2. Define griffin data quality
      1. Create DQ config file: **dq.json**
         1. Check Sample file on GIT: [**dq.json**](https://github.secureserver.net/GDLakeDataProcessors/util/blob/dev-private/Griffin_Artifacts/dq.json) 🡪 This file contains information about which tables to compare, which columns to check, business functionality etc.
2. Measure data quality
   1. Submit the measure job to Spark, with config file paths as parameters.
      1. **spark-submit --class org.apache.griffin.measure.Application --master yarn --deploy-mode client --queue default --driver-memory 1g --executor-memory 1g --num-executors 2 <path>/griffin-measure.jar <path>/env.json <path>/dq.json**
3. Report data quality metrics
   1. Then you can get the calculation log in console, after the job finishes, you can get the result metrics printed. The metrics will also be saved in hdfs:
      1. hdfs:///griffin/persist/<job name>/<timestamp>/\_METRICS.
4. Refine Data Quality report
   1. Depends on your business, you might need to refine your data quality measure further till you are satisfied.
5. Once your job completes, you can run command to sync output from HDFS to S3 to analyze further and build tables/dashboards on top of it for business users.
   1. aws s3 sync <<hdfs path>> <<s3 path>>
6. After we store output in S3 for all tables, we can build Athena table on top of it. This table will automatically get refreshed everyday and we can analyze it further if needed.
   1. Check Sample Athena table DDL on GIT: [Athena table](https://github.secureserver.net/GDLakeDataProcessors/util/blob/dev-private/Griffin_Artifacts/Griffin_Athena_Tables.sh) 🡪 This table contains all the data like source count, target count, mismatch records etc.
   2. As above table is raw table created on top of S3, We can create customized view on top of it.
      1. Check Sample Athena View DDL on GIT: [Athena View](https://github.secureserver.net/GDLakeDataProcessors/util/blob/dev-private/Griffin_Artifacts/Griffin_Athena_Tables.sh) 🡪 This view contains all the data like source count, target count, mismatch records, row count difference etc.
7. If we are going to use Persistent EMR Cluster for this validation, we are done after step 11. If we are going to use transient EMR cluster for validation. Please follow below steps as well.
8. Create Folder on S3, put bootstrap script which will copy all files like griffin jar, script to run spark job, dq.json file, env file on EMR once cluster launches.
   1. Check sample bootstrap file here on GIT: [Bootstrap file](https://github.secureserver.net/GDLakeDataProcessors/util/blob/dev-private/Griffin_Artifacts/bootstrap_griffin.sh) 🡪 This will copy all necessary files from S3 to EMR after we launch the cluster.
9. Create Airflow Dag which will launch EMR cluster, run .sh file contains spark job, Terminate the cluster.
   1. Check sample Airflow Dag file here on GIT: [Airflow Dag](https://github.secureserver.net/GDLakeDataProcessors/util/blob/dev-private/Griffin_Artifacts/griffin_dag_dp_stage_ref_fraud_order.py) 🡪 This Dag contains all steps like EMR launch cluster, .sh file to run as a step on EMR, Terminate the cluster.

**Below is the running code in prod:**

* + - 1. Bootstrap File 🡪 Git Path 🡪 <https://github.secureserver.net/GDLakeDataProcessors/util/blob/master/ref_fraud_order/bootstrap_griffin.sh>
      2. Airflow Dag 🡪 Git Path 🡪 <https://github.secureserver.net/GDLakeDataProcessors/uds_dag/blob/master/dm_reference/dp_stage_ref_fraud_order.py>
      3. Validation Script 🡪 Git Path 🡪 <https://github.secureserver.net/GDLakeDataProcessors/uds-griffin-validation/tree/master/ref_fraud_order>

We have completed all the steps here to run Validation end to end. For any issues or concerns you can get in touch with me on **(**[**dmahajan@godaddy.com/mahdipal@amazon.com**](mailto:dmahajan@godaddy.com/mahdipal@amazon.com)**).**

**Sample Outputs from Prod:**

1. I have captured how we can visualize the data in tables. Please check confluence page for tables etc.: [Validation Queries](https://confluence.godaddy.com/display/AWSHADOOP/Griffin_Validation_Results).

