Reference Guide: Sherlock Innovation Accelerator Platform – Data Science
At a high level, the data platform framework has four broad stages: ingest, store, process and analyze and visualize.

![Data Platform Framework Diagram]

**STAGE 1 & 2: Ingest & Store**

The PI could use something like WinSCP to upload files ([https://winscp.net/eng/docs/guide_amazon_s3](https://winscp.net/eng/docs/guide_amazon_s3)) or use the AWS CLI to create multiple upload streams. ([https://netdevops.me/2018/uploading-multiple-files-to-aws-s3-in-parallel](https://netdevops.me/2018/uploading-multiple-files-to-aws-s3-in-parallel)). Of course the AWS python SDK can be used as well ([https://aws.amazon.com/sdk-for-python/](https://aws.amazon.com/sdk-for-python/)).

An S3 bucket will be created to facilitate data ingest. Each Big Data project will be provided its own bucket. The PI will be provided an AWS Access Key ID and Secret Key to upload files. The PI can use any method to upload Files to the S3 bucket.

The S3 URL and AWS Access Key ID and Secret Key will be provided to the PI when the Project is initiated. Once the files have been uploaded, the S3 bucket will be closed to external internet access and only the Project hosts will be allowed access internal. The AWS Access Key ID and Secret Key will also be disabled.

Since S3 is priced based on use, you will only pay for what you use.

---

1 **AWS S3 Configuration and Security**

1. S3 bucket created solely for PI project
2. Access will [require SSL/TSL to access](https://www.hackingteam.io/ssl-tls-encryption/) (encryption in transit)
3. Access to S3 bucket [restricted to Host IP address via S3 bucket Policy](https://aws.amazon.com/bucket-policy/)
4. **Access restricted to AWS IAM Account** created to access this S3 bucket for the purpose of the PI’s project
5. S3 Bucket will use [AWS S3-SSE KMS](https://aws.amazon.com/kms/) for data encryption at rest (AES-256). An Project specific KMS key will be created and used specifically for the PI’s project and data encryption.
   a. S3 policy set to [require all uploads to use S3-SSE](https://aws.amazon.com/s3-policy/)

**AWS Security Monitoring of S3**

1. Once S3 policies are set, all changes to [the S3 bucket policies will be monitored using CloudTrail](https://aws.amazon.com/cloudtrail/)
2. Security will configure CloudWatch alarms and filter on the logs created by CloudTrail.
3. Security will setup SNS notification topic (with them as subscribers) for the CloudWatch alarms
4. [AWS Macie](https://aws.amazon.com/macie/) is an option for additional security but would be very expensive
SAMPLE COST OF S3 STORAGE FOR 15 TB PER MONTH.

If S3 storage is for less than month, the cost will be calculated by GB/Hr at $0.023 per GB.

STAGE 3: Analyze and Process

STAGE 3A: Access to secure environment
Once the data is uploaded to S3, the data can be processed and analyzed in a secure environment. A user would access their data environment by first logging on to a Windows RDS Jumpbox using 2FA (DUO) via a Remote Desktop Gateway (sherlock-rdp.sdsc.edu). This Jumpbox is specific to their project and access to it limited by AWS Security Groups and access controls using Active Directory. This Windows Jumpbox is needed because direct SSH access to the environment either from the internet or from VPN is not allowed for security reasons. Using a Windows RDP session allows access while adding this layer of security.

This Jumpbox will by default have an IAM Policy applied to it to allow anyone logged into the RDS host to have access to the files on the S3 bucket. The access to S3 from the RDS host would be through an S3 VPC endpoint. This endpoint allows internal access to S3 while blocking all internet access to the S3 Bucket.

Below is a diagram of a Default Baseline installation to access the environment and the S3 Files.
STAGE 3B: Processing and Analysis Options
Users are more open to change to newer or better technologies if the switching cost is minimal. The intent of this type of project is to make available the programs they are used to using in their current environment (i.e. Python-based or R) and offering them in our secure environment as host or cluster based processing solution.

STAGE 3B.1: Processing and Analysis using Host-Based Solution
If user would like a host based solution, we can offer a large host (i.e. r524x-large) on demand at Spot instance prices. The architecture would look like this:

Large hosts are needed to run data analytics for a project. However these hosts are very expensive and would be cost prohibitive to leave running while a PI sets up his software on the host. So we will setup an inexpensive EC2 instance (t3.medium) for the PI to install and configure his software packages. This host is temporary and once the PI has completed installation, we will create a snapshot of this EC2 and create a custom AMI (which includes the software). This custom AMI will be used to create spot instances of an r5.24xlarge. We will discuss this piece further below.

---

3 To access S3 data from within R a user would need to use the aws.s3 client package (https://CRAN.R-project.org/package=aws.s3). A python user can access S3 by using the AWS Python SDK (https://aws.amazon.com/sdk-for-python/).
This temporary host will have access to any GIT repos the PI needs through our Web proxy. This web proxy only allows whitelisted sites so the PI will need to provide the GIT URLs for whitelisting. We can also whitelist any python repos or other sites that are requested.

The storage and compute cost for this is about $67.00 a month. This can be prorated if EC2 and EBS is not needed for a month. There will be an EBS cost for the storage of the AMI template at $0.10 a GB.

At this stage the Jumpbox (<PROJ>-MGMT-RDS), and the S3 bucket have been created. The temporary EC2 (t3.medium) has been turned into a Custom AMI and shut down. We are now ready to launch the large compute host. This EC2 is expensive and the intent is to only have this host running during a job and then have it shutdown. The cost of an r5.24xlarge is as follows.

<table>
<thead>
<tr>
<th>EC2 Type</th>
<th>Cost</th>
<th>Specs</th>
</tr>
</thead>
<tbody>
<tr>
<td>r5.24xlarge</td>
<td>$ 1.6999</td>
<td>$ 6.0480, 768.0 GB, 96 vCPUs, EBS only, 25 Gigabit</td>
</tr>
</tbody>
</table>

It would be cheaper to run these processes using a Spot Instance than an On-demand instance. If you were to run the Host for ~300 hours during a month the following would be the cost of Spot vs On-Demand:

<table>
<thead>
<tr>
<th>Hours Provisioned</th>
<th>Instance Type</th>
<th>Pricing</th>
<th>Monthly Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>300</strong></td>
<td>r5.24xlarge</td>
<td>Spot</td>
<td><strong>$509.97</strong></td>
</tr>
<tr>
<td><strong>300</strong></td>
<td>r5.24xlarge</td>
<td>On-Demand</td>
<td><strong>$1,814.40</strong></td>
</tr>
</tbody>
</table>
In order to take advantage of the Spot Instance option, we will need to create an instance (and bid for the Spot instance⁴) each time a process needs to be run. And the instance will need to be deleted upon completion (~12-36 hours later when the run completes).

PI will be able to create an EC2 by executing a batch script on the Jumpbox (<PROJ>-MGMT-RDS), which will execute a AWS Lambda function that runs a Cloudformation configured to launch an r5 instance, using the AMI we previously created, creates a temp EBS storage volume and attaches the temp EBS storage to the r5 Host (<PROJ>-APP-01).

If the PI would like to run CLI, the PI can SSH into the newly created EC2 instance (<PROJ>-APP-01) from (<PROJ>-MGMT-RDS) using the PI’s AD account and run processes. If the PI would like to run Jupyter or Zeppelin notebooks or R studio, the PI can access these on the R5 host from a web browser on <PROJ>-MGMT-RDS.

When the processing is done, the PI will run the shutdown batch script (from <PROJ>-MGMT-RDS) which launches another Lambda function that deletes the Cloudformation stack. When this stack is deleted, the EC2 instance and the temp EBS volume will be deleted.

**STAGE 3B.2: Processing using Cluster-Based Solution**

**IPython Cluster**

While the above option allows a PI to continue to access individual files in a way they might have been accustomed, it suffers from the limitations of vertical scalability. A host can only get so large. If a PI would like to continue to expand resources while still using their code base, we can offer Jupyter clusters on demand.

A PI would still use S3 as before but now we would offer a Jupyter Notebook Server as a Cluster Master accessible from Project RDS Jumpbox via a Web browser. This server will always be on. It is a smaller instance as it will not be doing the computation but storing the Jupyter Notebooks and running the cluster scheduling. Now rather than running a CloudFormation to spin up a single large node, a PI could run a CloudFormation to spin up a cluster of spot instances as Slave nodes to the Cluster-Master (running IPython Parallel). We would setup Chef to configure the slave-nodes. These hosts do not need to be R5 instances but can be smaller and cheaper and still achieve more than 48 cores for the cluster.⁵

---

⁴ We will bid for the Spot instance using the On-demand pricing as our max bid. This will guarantee when the re-bid for the instance occurs each hour, we will not lose our Spot instance if the bidding price was raised.

⁵ While this is an example of the setup in Google Cloud, we would be able to use CloudFormation with Chef to create this cluster in AWS ([How to setup an IPython Parallel cluster on Google Compute Engine](https://cloud.google.com/compute/docs/instance-groups/using-instance-groups-with-parallel-cluster)).
**IPython Cluster Alternatives**

We can also offer StarCluster as an alternative to IPython Parallel.

If a PI is open to using Hadoop we can automate an EMR installation using CloudFormation with PySpark and Zeppelin Notebook.

**STAGE 3B.3: Analysis using Cluster-Based Solution**

While the above cluster options provide processing of data as well as the potential of analysis, it does not offer an on-demand SQL option to use directly with Analysis and Visualization tools like Tableau, Cognos, QuickInsight or other SQL based applications. In this case, AWS Athena would be a good option.

AWS Athena is a service that allows a user to run SQL queries directly on data in S3. Athena uses AWS managed Hadoop Spark clusters with Pesto to run SQL queries on S3 data. Athena can be accessed using

---

6 See further: [http://star.mit.edu/cluster/](http://star.mit.edu/cluster/)


8 See further: [https://aws.amazon.com/quicksight/](https://aws.amazon.com/quicksight/)

9 [https://aws.amazon.com/emr/features/presto/](https://aws.amazon.com/emr/features/presto/)
CLI, AWS SDKs and database connectors (jdbc/odbc). No servers need be setup by the user and the user only pays for TB scanned.

**Athena Setup**

Some initial setup is required before a user can use Athena. Since Athena is running Hadoop Spark in the background, Hive External Table Schemas must be created to allow Spark to read any data on S3 that a user would like to query using Athena. Spark needs Hive External Table Schemas because it uses schema on read on the data stored in S3. The Data and the Schemas are separate.

So for example, you had the following files on in an S3 bucket named: “test-bucket”.

```bash
$ aws s3 ls s3://test-bucket/files/
2016-11-23 02:21:15 0
2016-11-23 00:21:21 69043364 file_a.txt.gz
2016-11-23 00:21:29 11243304 file_b.txt.gz
2016-11-23 00:21:32 10253602 file_c.txt.gz
2016-11-23 00:21:39 12256401 file_d.txt.gz
```

You could create a Hive external table on top of those files. You have to specify the structure of the files by giving columns names and types and the location and all files found in that location folder would then be part of that table.

```
CREATE EXTERNAL TABLE entries (name STRING, comment, count INT)
LOCATION 's3://test-bucket/files/';
```

Athena will store these Table Schemas in Glue’s Data Catalog. An AWS Account can have only one Data Catalog. This Data Catalog services all Athena usage for all potential projects. A Data Catalog can have many Databases. A Database is a logical collection of Tables. IAM Access Policies can be applied to the Data Catalog, Database and Tables but cannot be applied to rows or columns or partitions within a table. Each PI will be assigned a Database under which External Tables can be created. No other projects will have access to the Database metadata of another project.

For a user to run a query using Athena, a user must not only have permissions to access the Database/Table metadata but a user must have S3 permissions to the underlying data defined by location in the Table Metadata. A user must have permissions to both the Data Catalog -> Database -> Table and the S3 data to run a query.

Any new files added to the folder defined in an existing Hive External Table (as long as the file has the same schema) will be included in all future queries.

---

10 These DB connectors can be installed on the Jumpbox and service Tableau or Cognos or any other SQL based application.

11 Athena costs $5.00 per TB scanned. A 15TB query would cost $75.00. Therefore a PI would want to setup data in S3 in such a way that only pertinent data is scanned per query and all data files are compressed. See further: https://aws.amazon.com/athena/pricing/.

12 AWS Glue Limits can be found at: https://docs.aws.amazon.com/glue/latest/dg/troubleshooting-service-limits.html
Optimizing Athena Costs

Athena charges by the TB scanned. There are several ways to structure files in S3 reduce scans and increase query speeds.

Compress Files
Athena supports the following compression formats: SNAPPY – the default compression format for files in the Parquet data storage format; ZLIB – the default compression format for files in the ORC data storage format; LZO and GZIP. For data in CSV, TSV, and JSON, Athena determines the compression type from the file extension. If it is not present, the data is not decompressed. If your data is compressed, make sure the file name includes the compression extension, such as gz.\(^\text{13}\) If a PI were to get a 3:1 compression gain, that would reduce scans cost by two thirds.

Use Partitions in Hive Tables\(^\text{14}\)
Adding Partitions to Table Schemas allow an Athena query to only scan those files that pertain to the WHERE statement in the query. So for instance, if you have your table partitioned by day then you could query between two dates and only the data between those two dates would be scanned. This reduces the data that would have been scanned but never needed. Partitions work best if it reflects common range filtering (e.g. by locations, timestamp ranges). If a PI is going to use Athena on large data in an interactive way, a restructuring of random data into folders and creating Tables with partitions would lead to huge cost savings.

Use Column-based Storage Formats
Most queries do not require all columns. However if you are using a row-based storage format such as JSON or CSV, you will initially get all the columns of a row and then Athena would discard those columns not included in the SELECT statement. This increases the cost of Athena scans. If you compress your file and also convert it to a columnar format like Apache Parquet, achieving 3:1 compression, you would still end up with 1 TB of data on Amazon S3. But, in this case, because Parquet is columnar, Amazon Athena can read only the column that is relevant for the query being run. Because the query in question only references a single column, Athena reads only that column and can avoid reading two thirds of the file. Since Athena only reads one third of the file, it scans just 0.33TB of data from S3.\(^\text{15}\)

If a PI had a Hive External Table defined in his Database that was in CSV or JSON format and he wanted to create a new table from this data in the Parquet or ORC storage format, he could perform a CTAS (Create Table As Select). For example:

The following example creates a CTAS query that stores the results as a text file:

```sql
CREATE TABLE ctas_csv_unpartitioned
WITH (  
  format = 'TEXTFILE',
  external_location = 's3://my_athena_results/ctas_csv_unpartitioned/')
AS SELECT key1, name1, address1, comment1
```

\(^\text{13}\) [https://docs.aws.amazon.com/athena/latest/ug/compression-formats.html](https://docs.aws.amazon.com/athena/latest/ug/compression-formats.html)

\(^\text{14}\) See Appendix 1 for more information about partition in Hive External Tables

\(^\text{15}\) [https://aws.amazon.com/athena/pricing/](https://aws.amazon.com/athena/pricing/)
FROM table1;

In the following example, results are stored in Parquet, and the default results location is used:

```sql
CREATE TABLE ctes_parquet_unpartitioned
WITH (format = 'PARQUET')
AS SELECT key1, name1, comment1
FROM table1;
```

Athena uses a default results location. Previously, this is the same location for all Athena queries in a single AWS Account. It looks like with Athena Workgroups, we will be able to setup location and encryption settings per project.

**STAGE 4: Visualization Options**

With Athena, a PI has the ability to connect any Visualization tool to a JDBC/ODBC connection and operate on the data directly. This could be BI tools like Tableau or Cognos or AWS QuickSight or it could be running the data through a Machine Learning service such as AWS SageMaker. AWS SageMaker has direct access to Athena and Glue’s Data Catalog.

---

16 [Athena Workgroups](https://docs.aws.amazon.com/athena/latest/ug/workgroups-settings.html) was released in Feb ’19 and this will allow Sherlock to set a default results location per workgroup (in our case per Project) and control access controls of result sets. See further:

17 See: [https://aws.amazon.com/sagemaker/](https://aws.amazon.com/sagemaker/)