**Active Directory integrated with Jenkins**

* The Jenkins automation server is widely considered the de-facto standard in open source continuous integration tools
* It offers a wealth of integration possibilities for user authentication and authorization, such as Unix **user database, OpenID, Github authentication and SAML.**
* However, enterprises most commonly use **Active Directory Domain Services (AD DS**) by Microsoft as their preferred **users’ database**.

Jenkins offers an **easy integration with AD** by using the **LDAP plugin.**

Jenkins switches to using the **AD database** for **usernames, passwords and group memberships,** but still remains **dependent on its own user database for localized information**, such as email addresses, the last granted authorities to a user and the user’s API token.

* When an **employee leaves a company**, that status **must immediately be reflected in the AD database** to prevent unauthorized use of Jenkins.
* Some organizations **disable an account** rather than delete it when a **user leaves the organization**.
  + The account and data are **kept for a period of time in disabled state,** according to the data retention **policy defined by the organization**, and then **deleted from the AD database**.
  + **Disabling AD accounts** may also take place during a **long vacation or as a pre-layoff measure**.
* **User** that has been **disabled from AD** will still be **able to access the Jenkins master** with their original permissions using their **API token** without alerting the **Jenkins admin.**
* Employee had a chance to write down their **API token,** they could continue using the token for as long as they wish.

**Installing the needed Plugins:**

The first thing we need to do is install the plugins that will be needed to complete this configuration.

The two plugins we are going to use are:

* **The Active Directory Plugin**
* **The Role-based Authorization Strategy Plugin**

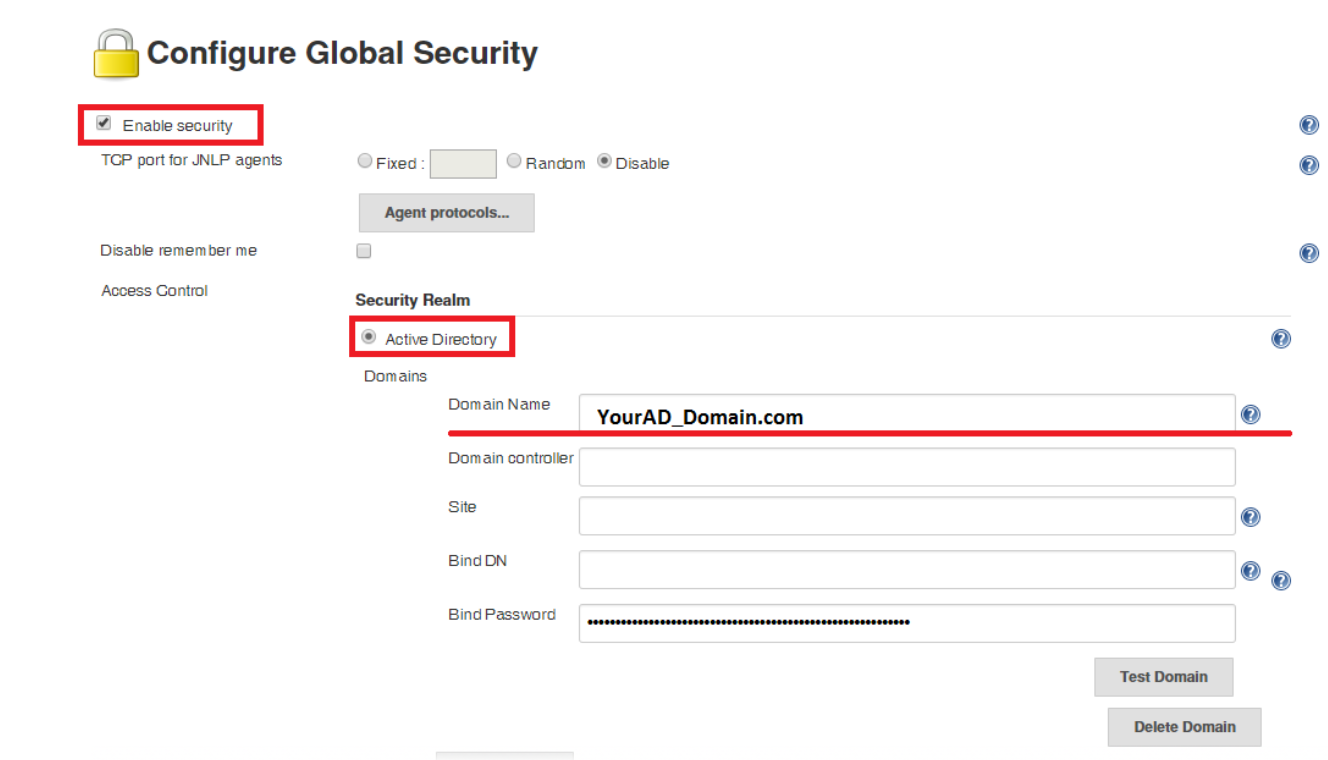
**Configure AD Integration:**

**Step1:**Jenkins is back up and running, we are going to navigate to the **Configure Global Security** page under **Manage Jenkins.**

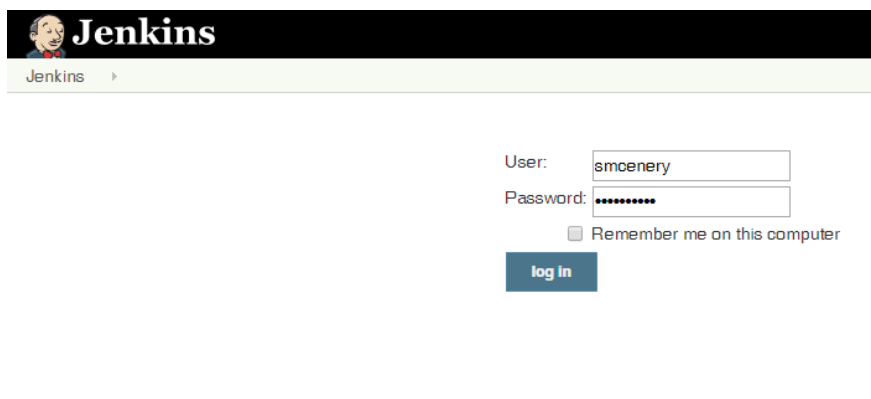
****

**Step2:**

* Ensure that the **Enable Security checkbox is ticked.**
* Under **Security Realm**, select the **Active Directory radio button**, then click the **Add Domain button** which appears to reveal the **configuration options pertinent** to AD.

****

* Once **saved the configuration, log out of Jenkins** to test it.



* If I click on my name, I noticed some additional details about the **AD account**, including all the **AD Groups** of my account.
* We have configured **Jenkins to Authenticate users** against an existing **AD Domain** and confirmed that users are now able to login to this instance of Jenkins using their **AD Domains.**

**Configuring Role Based Security:**

* Configured Jenkins to use AD Integration
* we can now take that **configuration a step** further **and manage our users’ permissions**, **access levels**, and rights by **leveraging the AD\Windows Groups.**

**Create Roles:**

* Prior to configuring **Role Based Security**
* Determine **what roles &what kinds of permissions**, would want those **roles to have.**

**Example:**

**Create 3 roles**: Admin, Developer, QA.

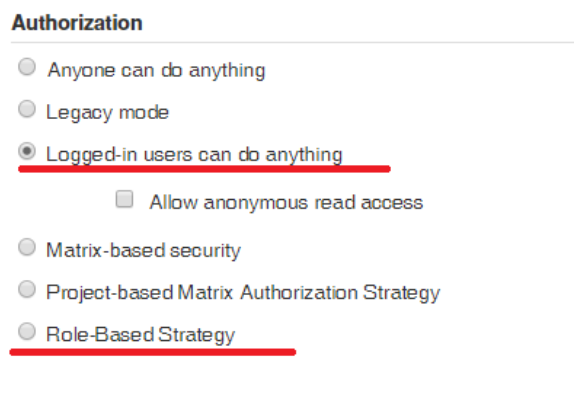
* **Admins** should be **able to do anything.**
* **Developers** should be able to **run jobs**, but **not edit any configurations.**
* **QA** should be able to **view the build history of pertinent jobs**.

**Step3:**

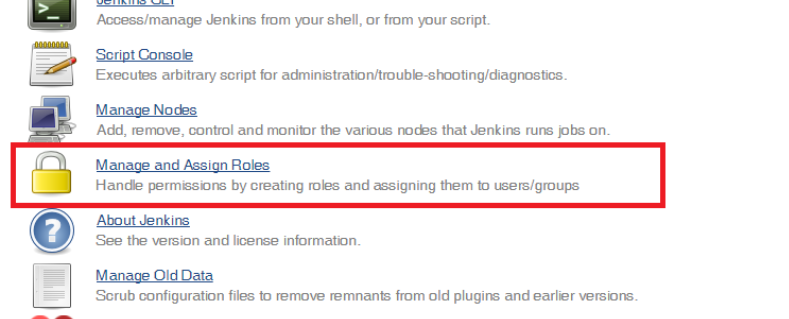
* Now we navigate to Manage and Assign roles
* Creating roles like Admin, Developer, QA
* Based on roles need to give permissions to each account

**Configure Role Based Security:**

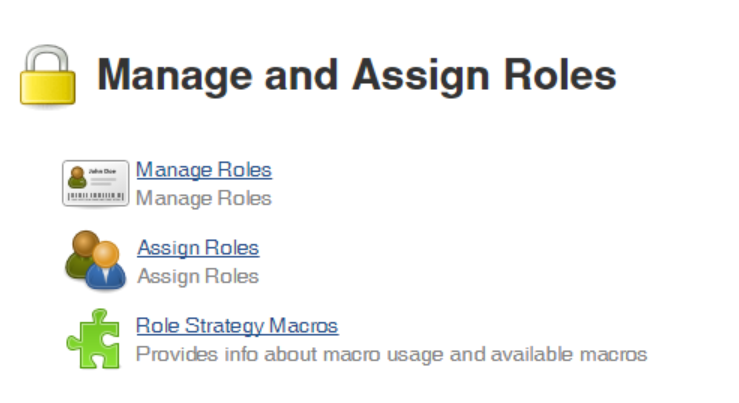
We configured the **AD Integration,** you will find a section titled **Authorization** and select the “**Role-Based Security option**”.



Once you select this radio button and click Save, go back to the **Manage Jenkins screen** and you will see a **new option** available in the **list of options.**

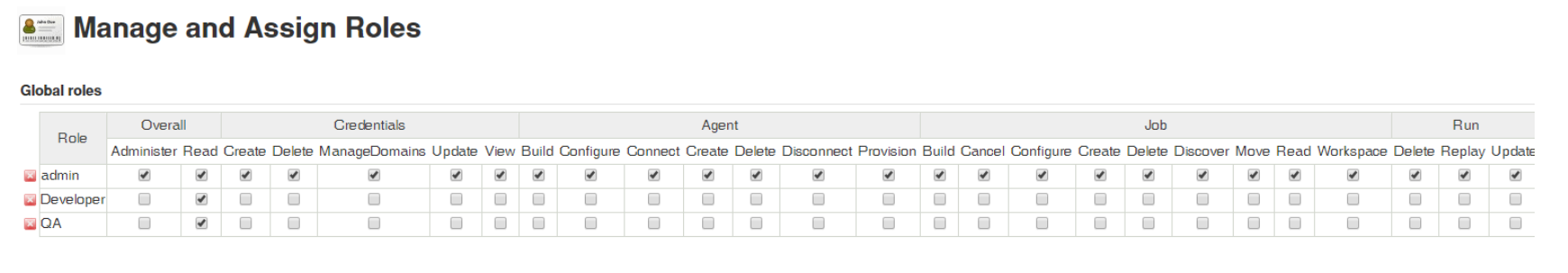


**Next Navigate to Manage & Assign Roles:**

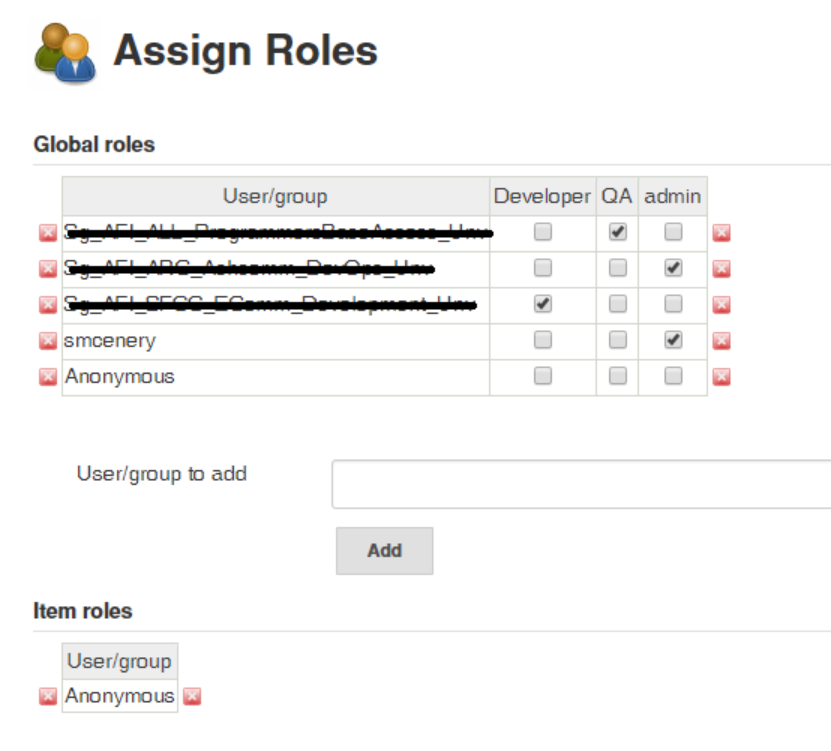
****

Create **our 3 roles as Global Roles** and ensure they all have the **Overall:Read** permission.

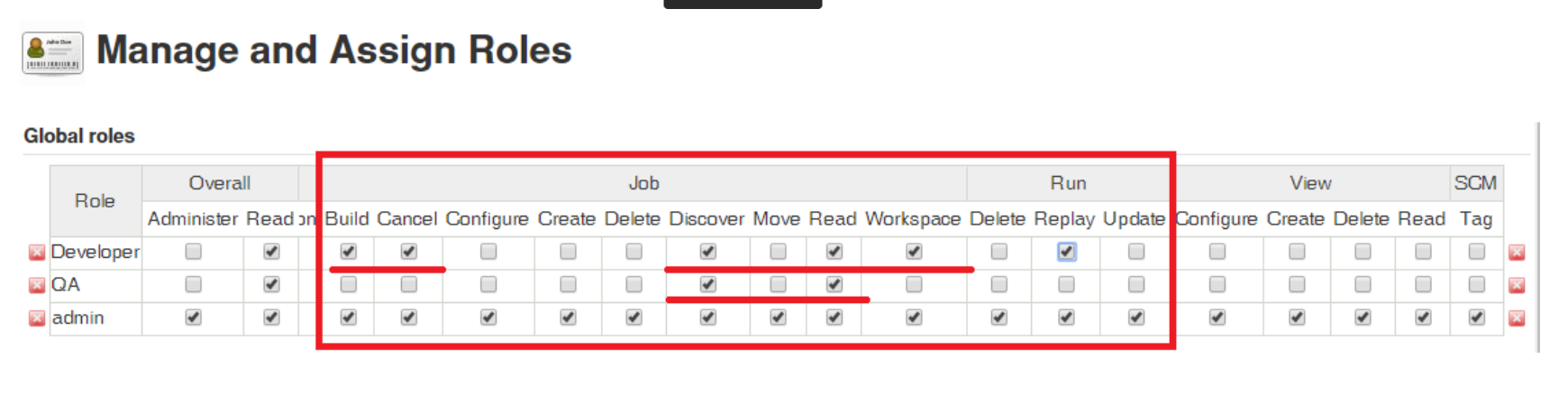
The Admin role will exist by default and will have all permissions by default as well.

****

**3 separate Windows Groups** that accurately represent the **different roles** I want people to play within Jenkins.

****

I have configured security in the following manner;

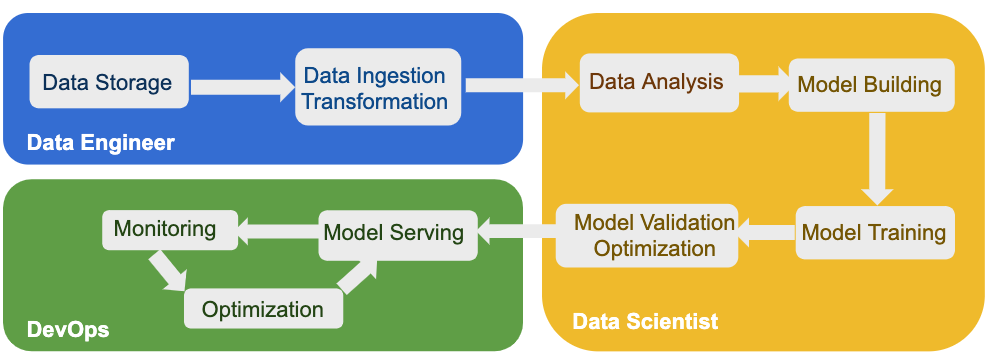
****

* This will allow users in the “Developer” role to Find and View jobs, Build and Cancel jobs & view the contents of the Workspace.
* Developers can also Re-run a job if they see the need. They will not be able to modify the configuration of any Jobs or Jenkins settings.
* Users in the QA Group are able to Find jobs and view the console output from previous builds.

**Open DataHub(ODH):**

**Open DataHub(ODH):** [**https://opendatahub.io/docs/getting-started/legacy/basic-tutorial.html**](https://opendatahub.io/docs/getting-started/legacy/basic-tutorial.html)

* The **Operational Data Hub (ODH**) is a **“newish**” concept—an enterprise architecture pattern— that is a categorization of an **architecture that a customer builds to support multiple lines of business**.
* A **Data hub** is a **centralized service that connects all of your IT systems**, whether they be Web applications, IoT devices, SaaS solutions, or core business platforms, such as CRM or ERP.
* A **Data hub** **manages the connections** to each of the systems and orchestrates the data flow amongst them.
* A **Data hub** is a logical architecture which enables data sharing by connecting producers of data (**applications, processes, and teams**) with consumers of data (**other applications, process, and teams**).
* **Endpoints interact with the data hub**, Provisioning data into it or receiving data from it, and the hub provides a point of mediation and governance and **visibility to how data is flowing across the enterprise.**
* **Data Hubs** enable **efficiency, scale, and agility.**
* **Data hubs also** simplify the data governance requirements as the **data is persisted** at a **central location.**
* **Open Data Hub(ODH):** Currently provides services on OpenShift for AI data services such as data storage and ingestion/transformation.
* For data storage and availability, ODH provides [Ceph](https://ceph.com/), with multi-protocol support including block, file and S3 object API support, both for persistent storage within the containers and as a scalable object storage data lake that AI applications can store and access data from Rook operator can be used to easily deploy and integrate Ceph into the OpenShift and ODH ecosystem.



**AI workflow** is initiated by **Data Engineers** that acquire the **data from different sources** and perform the required transformations.

**Data Engineers** are also responsible to **store and provide access to the transformed data to Data Scientist** or Data Analysts that work on the second phase in the AI workflow

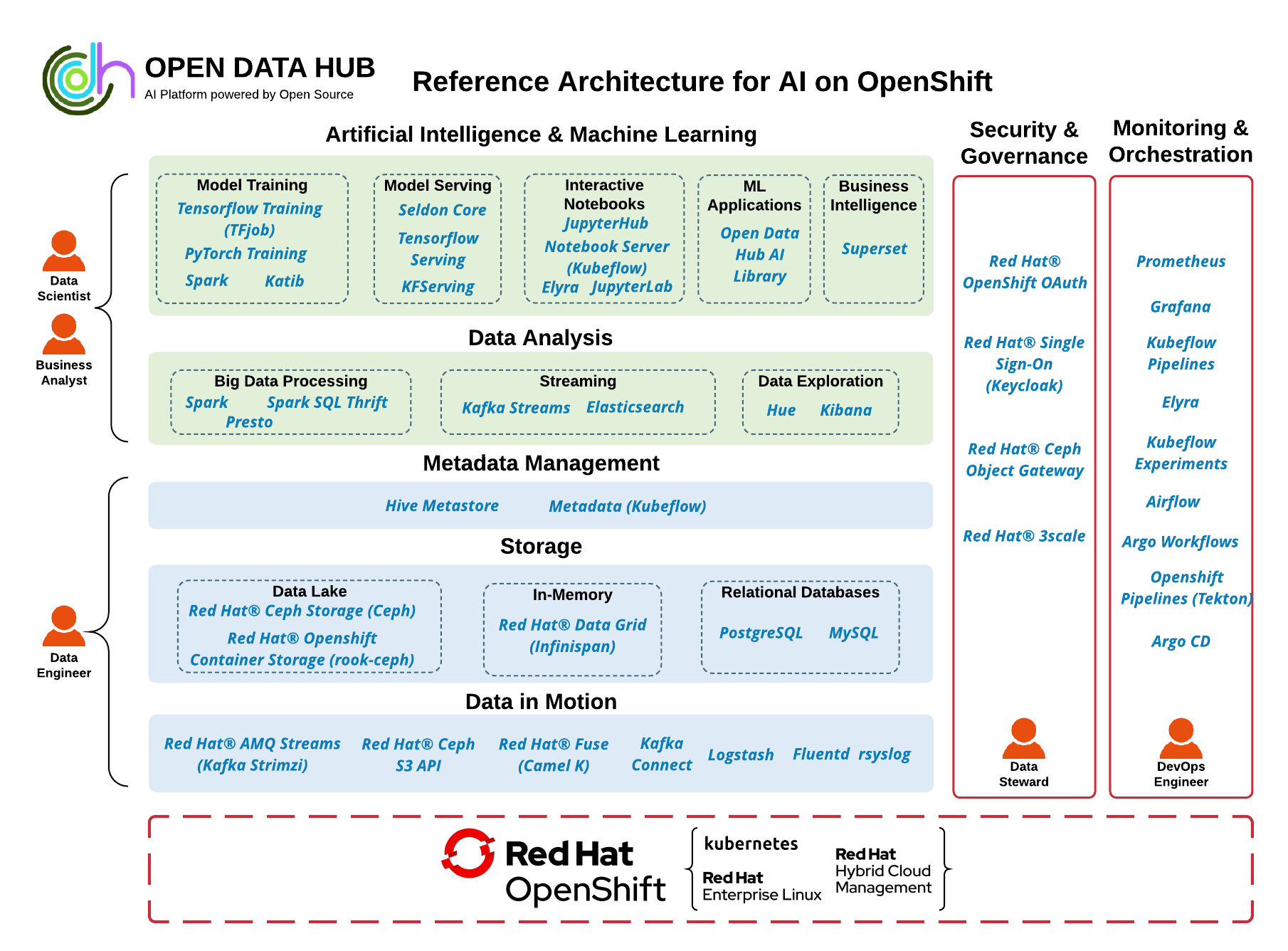
**Pre-requisites:**

* OpenShift 4.5
* Spark
* JupyterHub
* S3 Object Storage

**JupyterHub** and **Spark** are installed by default with Open Data Hub.

**Object Storage:** Access data on an Object Store (such as Ceph or AWS S3) using the S3 API.

**Spark + Object Storage:** Same data file from Spark so you can analyze data.



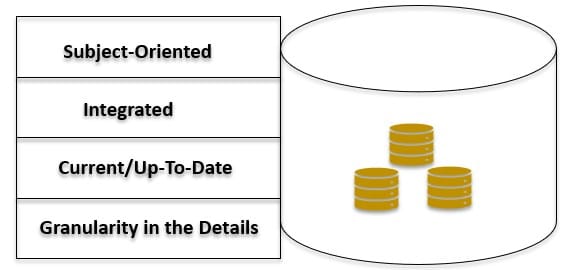
**Characteristics of Operational Data Store Systems:**

Here are the attributes of Operational Data Stores (ODS):

* ODS systems are highly available and fault tolerant.
* They occupy less space due to the compression of data and operations.
* ODS systems host configurable, easily accessible, and fast real-time comprehensive data.
* ODS systems are connected to one or more data sources.
* They do not host large amounts of historical data, and thus cannot handle huge data transactions.
* An ODS system makes the creation of back-ups and recovery processes effortlessly since the size of the data is small.

**Use Cases of Operational Data Store Systems**

* The main purpose of an ODS is to integrate data from diverse source systems into a single entity, through technologies such as Extract, Transform and Load (ETL), Data Federation, or Data Virtualization.



In the creation of an Operational Data Store, multiple data sources can be integrated. However, each data source system must have the following principles to qualify:

**Subject-Oriented:** The design of the Operational Data Store should be built based on the business’s functional requirements, especially in regard to a specific area under discussion.

**Integrated:** All the data from diverse sources must undergo the ETL process, which involves cleaning junk for redundancy, data transformation into a single format, and loading of the dataset into the ODS as indicated by the business policies for control and regularity of data.

**Up to Date:** The ODS data should be current and thus updated to host all recent functions of the application connected to the Data Warehouse, as well as to depict the data’s existing status from each Data Warehouse linked source.

**Detailed:** As the rules are implemented, it is crucial to maintain the business’s comprehensive detailing level for the proper execution of respective functions, which are mainly supporting the operational business requirements or functions.

**Notable operational databases include:**

| **Database platform** | **Database model** | [**SQL**](https://en.wikipedia.org/wiki/SQL)**Support** | [**NoSQL**](https://en.wikipedia.org/wiki/NoSQL)**Support** | **Managed objects** | **ACID-transactions** |
| --- | --- | --- | --- | --- | --- |
| [Aerospike](https://en.wikipedia.org/wiki/Aerospike_(database)) | Key–Value Store | No | **Yes** | key-value pairs | None |
| [Altibase](https://en.wikipedia.org/wiki/Altibase) | Relational database | **Yes** | No | tabular data | Real-time ACID transactions |
| [Apache Cassandra](https://en.wikipedia.org/wiki/Apache_Cassandra) | Key-value store | No | **Yes** | key-value pairs | None |
| [Cloudant](https://en.wikipedia.org/wiki/Cloudant) | Document-Oriented Database | No | **Yes** | JSON | None |
| [Clusterpoint](https://en.wikipedia.org/wiki/Clusterpoint) | Document-Oriented Database | **Yes** (essential SQL) | **Yes** | XML, JSON, text data | Distributed ACID-transactions |
| [Clustrix](https://en.wikipedia.org/wiki/Clustrix) | Relational Database | **Yes** (newSQL) | No | tabular data | ACID-transactions |
| [Couchbase](https://en.wikipedia.org/wiki/Couchbase) | Document-Oriented Database | **Yes** (N1QL) | **Yes** | JSON, key-value pairs | Distributed Multi-Document ACID transactions |
| [CouchDB](https://en.wikipedia.org/wiki/CouchDB) | Document-Oriented Database | No | **Yes** | JSON | None |
| [Db2](https://en.wikipedia.org/wiki/IBM_DB2) | Relational Database, Data Warehouse | **Yes** | **Yes** | JSON, key-value pairs, tabular data, XML | ACID-transactions |
| [EnterpriseDB](https://en.wikipedia.org/wiki/EnterpriseDB) | Relational Database | **Yes** | No | tabular data | ACID-transactions |
| [FoundationDB](https://en.wikipedia.org/wiki/FoundationDB) | Key-value store | **Yes** | No | key-value pairs | ACID-transactions |
| [Ingres](https://en.wikipedia.org/wiki/Ingres_(database)) | Relational Database | **Yes** | No | tabular data | ACID-transactions |
| [MarkLogic](https://en.wikipedia.org/wiki/MarkLogic) | Document-Oriented Database | Yes | **Yes** (XQuery and Javascript) | XML and JSON | ACID-transactions |
| [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) | Relational Database | **Yes** | No | tabular data | ACID-transactions |
| [MongoDB](https://en.wikipedia.org/wiki/MongoDB) | Document-Oriented Database | No | **Yes** | BSON | None |
| [NuoDB](https://en.wikipedia.org/wiki/NuoDB) | Relational Database | **Yes** (newSQL) | No | tabular data | ACID-compliant |
| [Oracle](https://en.wikipedia.org/wiki/Oracle_Database) | Relational Database | **Yes** | Yes | multi-model | ACID-transactions |
| [OrientDB](https://en.wikipedia.org/wiki/OrientDB) | Document-oriented Database | **Yes** | Yes | key-value pairs | ACID-transactions[[4]](https://en.wikipedia.org/wiki/Operational_database#cite_note-4) |
| [Riak](https://en.wikipedia.org/wiki/Riak) | Key-value store | No | **Yes** | key-value pairs | None |
| [SAP HANA](https://en.wikipedia.org/wiki/SAP_HANA) | Relational Database | **Yes** | No | tabular data | ACID-transactions |
| [VoltDB](https://en.wikipedia.org/wiki/VoltDB) | Relational Database | **Yes** (newSQL) | No | tabular data | ACID-transactions |

**Integration Bitbucket with Jenkins:**

* Click on Manage Jenkins go to Manage Plugins **search** Available plugin Bitbucket Branch Source Plugin**.**
* Create job select Bitbucket team and write pipeline.

Bit Bucket 🡪 Settings 🡪 App Passwords 🡪 Create app password

Jenkins 🡪 Create BitBucket Team 🡪 Owner: Team ID…….; Scan Credentials: …; 🡪 apply & Save

**FMDB (Flying Meat Database):**

<https://www.objc.io/issues/4-core-data/SQLite-instead-of-core-data/>

http://www.theappguruz.com/blog/sqlite-database-in-ios#:~:text=Step%201%20What%20is%20FMDB,.com%2Fccgus%2Ffmdb.

* FMDatabase - represents a single SQLite database
* FMResultSet - the query results from the database
* FMDatabaseQueue - used to make queries in threads
* It is used as a full featured **wrapper for SQLite database**.
* Is an open source **Objective-C wrapper for SQLite.**
* It is used to **create a database, creating table, inserting records in table, fetching records, or deleting records** from table.