cQube – Technical Specifications Document

Jan 2021

Current Version 1.9

Document released by:
Sreenivas Nimmagadda
(System Architect)

Document reviewed by:
Sateesh Pullela
(Technical head)

Document acceptance by:
Arvind Gopalakrishnan
(Delivery head)

Contents

1. Background and Context

EkStep and Tibil Solutions have embarked on a project named as 'cQube' to create an analytics and decision making tool for the education system. This product can be used for monitoring the education system in a state and on a broader scale for monitoring the schools across various levels of administration.

1.1 Use of this document

This document will cover the technical points of following areas:

1. Data emission process.
2. Role of Java and Python programming.
3. Nifi Processor information - Processor Groups, Data validations, Query configurations, Output file formats.
4. Prometheus and Grafana monitoring tool connectivity.
5. Database and Data modelling information.
6. Node connectivity with S3 bucket.
7. Angular JS, Chart JS and Leaflet roles in the visualization.
8. Keycloak integration with authentication process.
9. Logs.

1.2 State of this document

This document is an evolving document and is under change control. To request a change to the technical document please contact the system architect or the project manager.

1.3 Acronyms

The following is a list of acronyms which will be used throughout this document:

Table – 1: Acronyms

Acronym	Description
S3	Simple Storage Service
NIFI	Apache NIFI
PK	Primary Key
FK	Foreign key
RDAC	Restricted Database Access Control

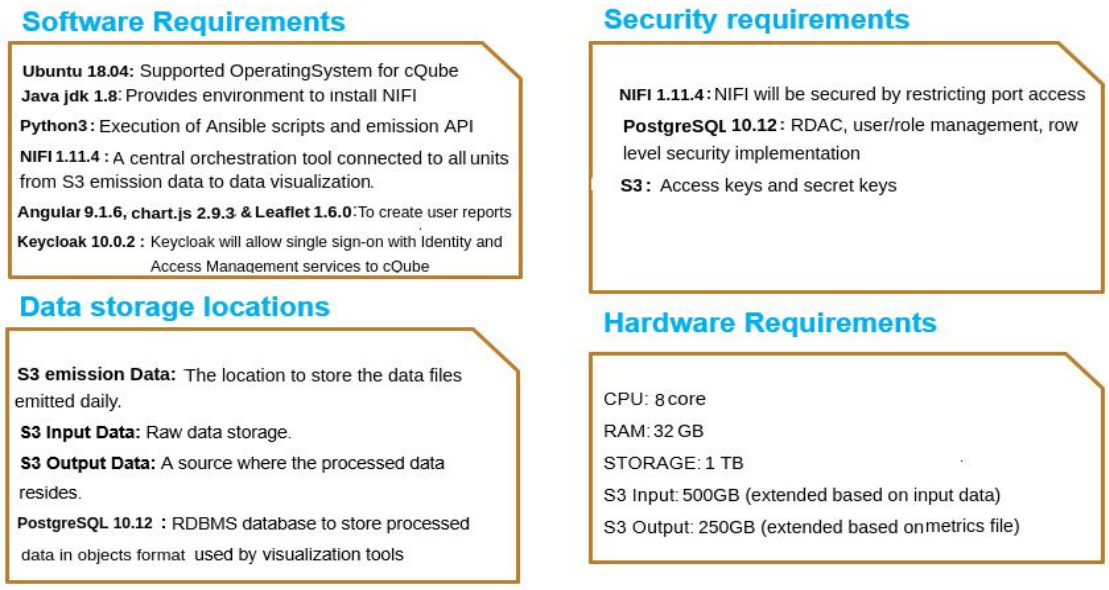
2. cQube Product setup

This section describes the prerequisites to install and configure the cQube product setup and the cQube product setup process. This section also describes the cQube network setup and the setup process

2.1 cQube product prerequisites

The cQube product installation process has a few system prerequisites that need to be followed. The system software, hardware, and security requirements have been derived and have to be adhered to before installation. The figure below gives an overview of the software requirements, security requirements, hardware requirements and the data storage locations:

Figure – 1: System hardware, software requirements, security requirements and data storage



Mentioned below are the prerequisites for the installation of the cQube product:

- The cQube product has to be installed in the AWS environment.
- Firstly an instance with Ubuntu 18.04 OS has to be created.
- All the hardware requirements mentioned in Section 4.1.4 have to be adhered to.
- Before starting installation, the network set-up has to be completed as mentioned in Section 5 of this document.

The below table describes the Infrastructure details to install the cQube

Activity	**Infrastructure used for cQube installation**	**Required Skills**
1-2 Weeks	2 Days	
Filling of the requirements in the required format and obtaining relevant approvals for procurement of the Infrastructure and securing funding for monthly/yearly spend.	AWS account 1. cQube Server (CPU - 8 Core RAM - 32 GB Storage - 500 GB) 2. S3 Buckets (S3 emission 100GB S3 input - 750 GB S3 Output - 250 GB)	AWS Basic Operation Skills - EC2 - S3 - Load Balancer - IAM - VPC - Route53 - Certificate Manager (SSL)
	OpenVPN Access Server (EC2 instance CPU - 1 Core RAM - 2 GB, Storage - 10 GB)	VPN admin Operations
	NGINX Reverse proxy(EC2 instance CPU - 2 Core RAM - 4 GB Storage - 30 GB)	NGINX configuration skills
	NAT gateway	AWS NAT gateway
	Ubuntu 18.04	Will be taken care while creating EC2 instance
	Java jdk 1.8	Will be installed through One-Step cQube Installation
	Python 3	Will be installed through One-Step cQube Installation
	NIFI 1.11.4	Will be installed through One-Step cQube Installation
	Angular 9.1.6, Chart.JS 2.9.3, Leaflet 1.6.0	Will be installed through One-Step cQube Installation
	PostgreSQL 10.12	Will be installed through One-Step cQube Installation

2.1.1 Software Requirements

Ubuntu 18.04: This is the operating system that supports the cQube product

- Java JDK1.8: This provides an environment for NIFI installation
- Python3: Python plays a role in the execution of Ansible scripts and data emission API using a virtual environment.
- NIFI 1.11.4: A central orchestration tool which is connected to all the units and all the different sections where the data flows, starting from the S3 emission data location to the data visualization stage
- PostgreSQL 10.12: An RDBMS database to keep all the processed data in relational data format. The data stored here is used by NIFI to prepare the JSON format files which can be used in the visualization charts.
- Angular 9.1.6 + ChartJS 2.9.3 + Leaflet 1.6.0: Angular, ChartJS and Leaflet are used to create dashboards/ user reports. The data stored in the S3 output bucket can directly be used to create the reports.

2.1.2 Security requirements

- cQube product security will be provided by implementing the private subnet with AWS load balancer as described in the Section 2 of this document.
- All the ports will be accessed by Nginx server only, So those ports will not be accessed directly from the internet.
- S3 buckets will be secured by the AWS default security.
- PostgreSQL will be secured by the following ways
  (The database security will be implemented in the future versions of cQube)
    - **RDAC:** Postgres provides mechanisms to allow users to limit the access to their data that is provided to other users. Database super-users (i.e., users who have `pg_user.usesuper` set) silently bypass all of the access controls described below with two exceptions: manual system catalog updates are not permitted if the user does not have `pg_user.usecatupd` set, and destruction of system catalogs (or modification of their schemas) is never allowed.

- **User and Role Management:** the user roles will be granted with one or more cQube database will have three different types or roles:
    1. role role (identified by prefix r_)
    2. group role (identified by prefix g_)
    3. user role (generally personal or application names)
- **Row Level Security:** In addition to the SQL-standard privilege system available through GRANT, tables can have row security policies that restrict, on a per-user basis, which rows can be returned by normal queries or inserted, updated, or deleted by data modification commands. This feature is also known as Row-Level Security. When row security is enabled on a table all normal access to the table for selecting rows or modifying rows must be allowed by a row security policy.

2.1.3 Data Storage Locations

- S3 emission data Location: The data emitted from the state education system has to be stored until the NIFI reads the data. S3 emission storage buckets are the storage locations where the emitted data files are stored.
- S3 Input Data: This is a location where the raw data resides for all future references.
- PostgreSQL: All the transformed and aggregated data will be stored in PostgreSQL tables.
- S3 Output Data: A location where the processed data resides in JSON format.

2.1.4 Hardware Requirements

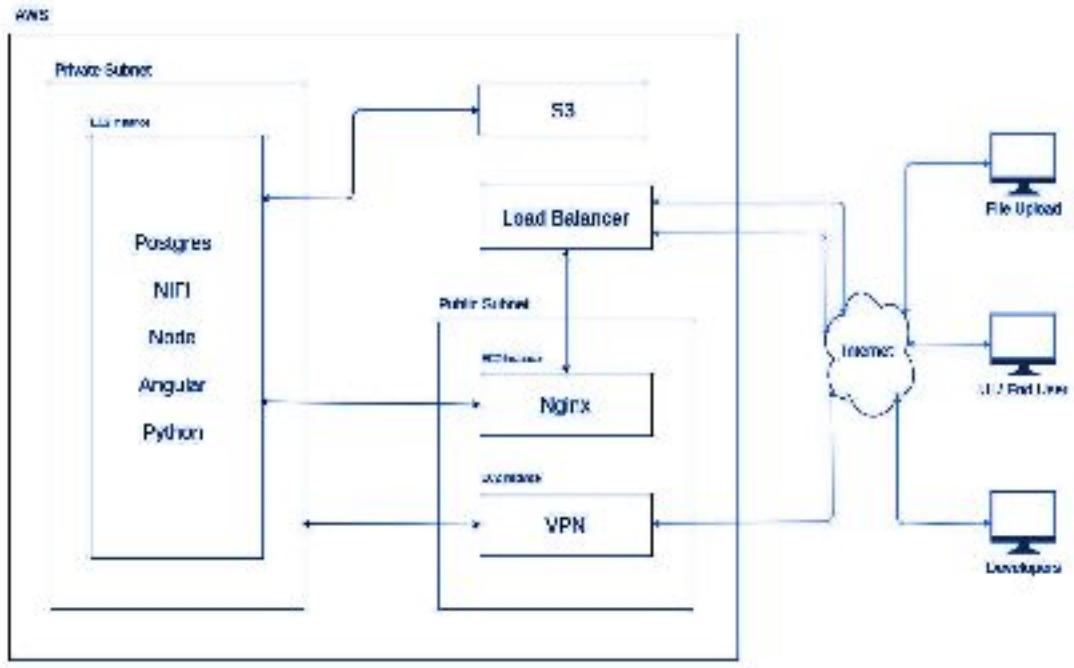
Listed below are the minimum hardware specifications/ requirements in order to install the cQube product

- 8 core CPU
- 32 GB RAM
- 500 GB - 1 TB harddisk
- Three S3 Buckets will be used - for cQube data emission, cQube data input and cQube data output

2.2 cQube network setup

The cQube network setup consists of the AWS, which encompasses the private subnet section which contain the EC2 instances Postgres, NIFI, node, angular and python, the public subnet section which comprises of the Nginx and VPN, S3 and the Load balancer. The cQube network setup process is described in the block diagram below:

Figure – 2: cQube network setup diagram



2.2.1 End User

- When a user accesses the cQube application through the browser, the request hits the load balancer of the AWS.
- Load balancer will forward the request to Nginx proxy server.
- Nginx will forward the request to angular application using private IP
- Angular sends the request to NodeJS server
- NodeJS will get the data from S3 bucket and respond to Angular
- Angular will process the data and display the results.

2.2.2 Emission User

- These users call the Rest API through python client-side script to upload the files
- After user authentication, Rest API (Written in python) provides the S3 one-time URL
- Client-side python upload the files using the S3 one-time URL

2.2.3 Developers

- Normally developers can deploy the changes through the Jenkins CI/CD pipeline.
- Developers use VPN to connect to the cQube production application if there are any direct changes, like technical changes, configuration or customizations are required.

2.3 cQube - for Installation

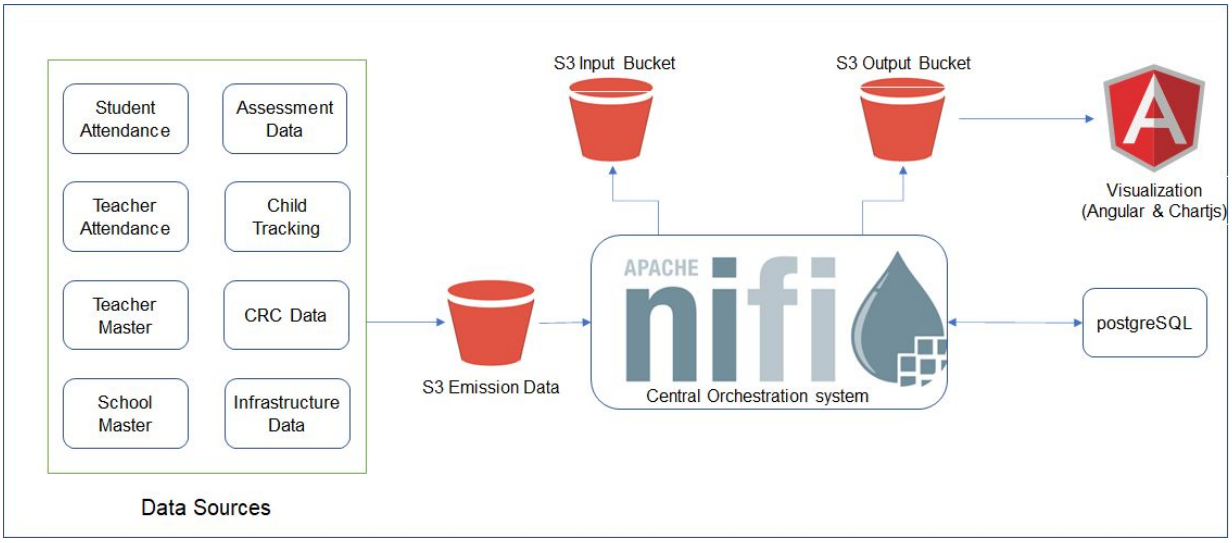
The cQube product can be installed as a one-step installation process. The one-step process installs the complete cQube stack, which includes Ansible automation scripts to install Java, Python, NIFI, Angular, ChartJS, Leaflet, S3 emission data, S3 input bucket, S3 output bucket and PostgreSQL installations.

Steps for Installation:

- As a first step, clone the files from GitHub using the following command:
  $ git clone https://github.com/project-sunbird/cQube.git.
- This downloads the cQube installation files. The cQube GitHub has all the installation files required for installing cQube.
- The README.md file has all the instructions that have to be followed for the cQube product installation.
- The install.sh script installs the complete cQube stack.
- The Install.sh file calls the Ansible playbooks in the background which will complete the cQube installation setup.
- The complete installation process takes approximately 30 minutes.
- Once the installation is complete, the message “cQube successfully installed” is displayed.

3. Software Architecture

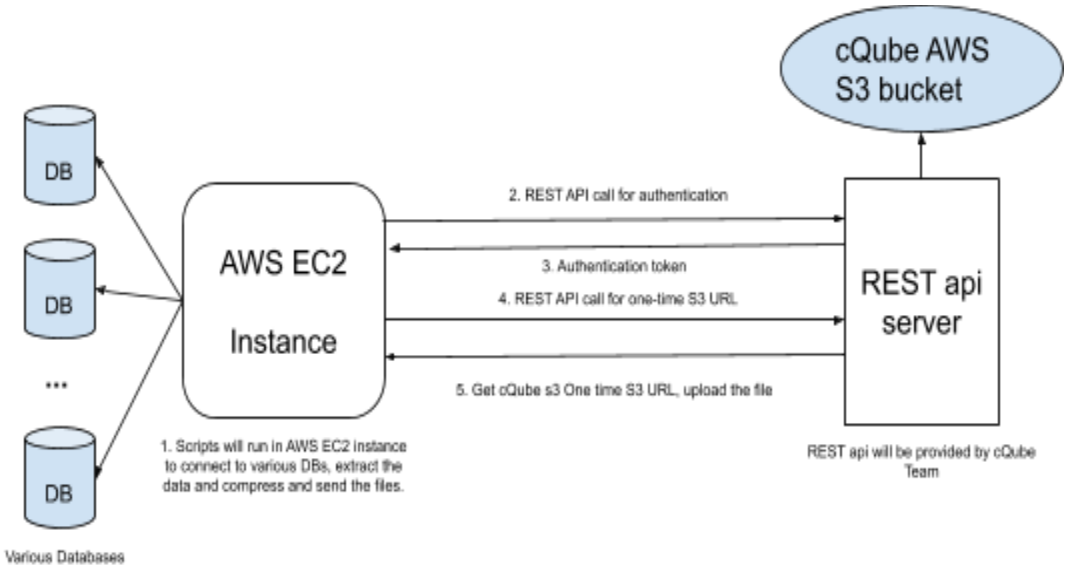
Figure – 3: cQube software architecture



3.1 Data emission process

cQube provides an API for the data ingestion. Authenticated API call provides onetime S3 URL and emits the data into the S3 emission bucket.

Figure – 5: Data emission process



The steps involved in the data Emission Process

- The data emitters will work from the data source location.
- Data emission will be performed periodically as per the specified time interval from different data sources with the help of an automated extraction process.
- Once the emission process extracts the data fields which are used by cQube, it is converted into csv formatted, pipe delimited files.
- The CSV data files will then be placed into the state data center by the automated process.
- Emission automated processes will invoke cQube data-ingestion APIs to emit the data.
- The API will have different end points as mentioned below:
    - Data emission users will request the cQube admin for the emission API token.
    - Data emission users incorporate the API token into the emission process code.
    - The Emission process makes an API call to generate AWS S3, one time presigned URL.
    - API calls to emit the data files using the https protocol into cQube.
    - API takes the data file as a parameter.
    - The API sends an acknowledgement on successful emission.

3.1.2 Certificate based authentication for emission & download API

The certificate based authentication is an additional security layer to authenticate the client based on the client certificate & client key along with the user credentials. Without the certificate & key user would not be able to emit the files to cQube or download the data from cQube. Below are the steps to add the certificate based authentication.

- Create a new subdomain for api and assign that api domain name to nginx server.
- Create a server config in nginx on nginx server.
- Using letsencrypt (or anything is fine), create an ssl certificate for that api domain and configure the ssl certificates to api domain.
- Create a self-signed client certificate and configure it.
- Copy the client certificate and key to the client machine (where emission happens).
- Restart the nginx on nginx server.
- Unassign the api domain name to nginx server.
- Comment out or delete the api selection in main nginx server configuration.

- Create a new load balancer with TCP listener.
- Assign api domain name to load balancer's A record.
- Add the load balancer's sg to nginx sg on port 443.

- The example spec for the emission API will be as like below

```
headers = {'Authorization': 'Bearer access_token', 'Content-Type': 'application/json'}

GET https://cqube.tibilprojects.com/data/list_s3_buckets
Body: {"input": "cqube-qa10-input", "output": "cqube-qa10-output", "emission":
"cqube-qa10-emission"}

POST https://cqube.tibilprojects.com/data/list_s3_files
Body: { "bucket": "cqube-qa10-emission"}

POST https://cqube.tibilprojects.com/data/download_uri
Body:
{"filename":"school_master/2020/2020-06/2020-06-04_school_master/04-06-2020_13:12:59.574_5e1
60862-c5b3-4121-9a96-ecefa34fc264_school_mst.zip","bucket":"cqube-gj-input"}
```

3.2 NIFI Processor Groups

- NIFI will have processor groups to combine similar NIFI processors, i.e the processes that are related to the same functionality into batches.
- Separate processor groups enable only the required data source transformation and this supports further scaling.
- The processor groups are loosely coupled, which enables them to make specific changes required to any particular processor group without affecting all the other processor groups.

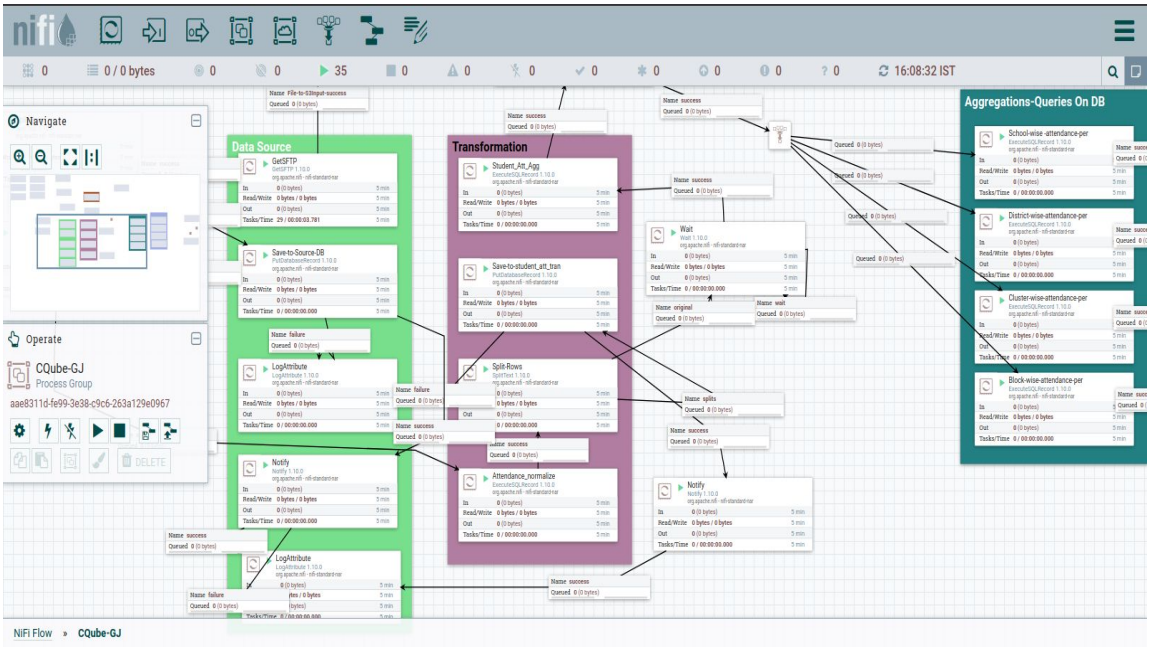
3.2.1 NIFI Data Process

- NIFI acts as the central orchestration system and all the aggregations are performed in the NIFI using PostgreSQL databases.
- NIFI fetches the aggregated data from the PostgreSQL database and sends them to the S3 output bucket.

3.2.2 NIFI data pipeline

cQube uses Apache NIFI to automate the data flow

Figure – 6: NIFI flow screenshot



- Apache NIFI Data Source Processor fetches the data from S3 emission data bucket.
- Stores raw data into S3 Input bucket.
- NIFI processes perform the data validations.
- Transformation Processor: Data is transformed using the queries in PostgreSQL and the transformed data is sent to the PostgreSQL.
- Aggregation Processors: Data aggregations are performed on the transformed data.
- NIFI stores the aggregated data and static data into S3 output buckets in JSON format.

3.2.3 NIFI Data Validations

- Data validations will take place at the following different levels:
    - The emitted data undergoes their first set of data validations before being copied into the S3 Input bucket
    - Second set of data validation takes place after the files are copied into the S3 Input bucket and before the Nifi process starts processing the data for cQube report creation.

NIFI fetches the data from the S3 emission data bucket and sends it to the S3 input bucket, after performing the following validations on the data:

- **Emitted data file size check:** NIFI gets the file size of the emitted data from the Manifest file and performs a check to see if the emitted file size matches with the original file size.
  NIFI does not allow the file into the S3 Input bucket if the file size does not match. A notification is sent through an email to the data emitter to check the file size and re-emit the data file.
- **Record count at the emitted file check:** NIFI gets the count of the number of records in the emitted data from the Manifest file. Nifi checks if the emitted file records count matches with the original file records count.
  If the file records do not match, the NIFI will not process the file into the S3 Input bucket. A notification email is sent to the data emitter to re-emit the data file.

NIFI fetches the data from the S3 emission data bucket and sends it to the S3 input bucket, after performing the following validations:

- **Column level validations**: Column datatype mismatch, Number of columns, Data exceeding the column size.
- **Improper Data handling**: Missing/ null data values for mandatory fields, Empty data files, Special characters, Blank lines in data files.
- **Duplicate records validation:** NIFI validates the duplicate records by grouping the same kind of records together.

For the record which is having duplicate values for all fields (mirror image record) NIFI will consider the first record and the rest of the records will be eliminated.

For the rest of the duplicate records where the records are having the same ID (student ID/ assessment ID/ infra ID/ CRC visit ID) and different values will not be inserted into the database tables as ID is the primary key.

For the Duplicate records with different lat long details, NIFI eliminates the records which have the same id and different lat long details and the records which have different ids and the same lat long details.

For semester report, The records which are having the same values for fields Student ID, School ID, semester, studying class and different values for the subjects then NIFI will eliminate those records.

- **Overlapping data validation**: Overlapping data validation takes place based on the data source.

  The NIFI process for student attendance reports will check the last updated day's record from the transactional table and will process the records from the day after the last updated date. The records from all of the previous days will not be considered for NIFI processing.

  For the other data sources, duplicate records where the records are having the same ID (student ID/ assessment ID/ infra ID/ CRC visit ID) and different values will not be inserted into the database tables as ID is the primary key.

- **Other data issues**: Data handling in cases like job failures, missing data for certain days and late receipt of the data (receiving data after a few days), updating the wrong data, upon request (when issue identified at the report)

3.3 Prometheus and Grafana monitoring tool connectivity

Prometheus:

- Configure a job to scrape for the server's RAM, CPU and Storage status on port 9100
- Configure a job to scrape for the Nifi's process and Memory status on port 9092
- Configure a job to scrape prometheus itself health check on port 9090

Node exporter:

- Configure node_exported to send the metrics about the node (cQube server)

Grafana:

- Configure prometheus as new Datasource
- Import the dashboard given in cQube git repository located on cQube/ development/ grafana/ cQube_Monitoring_Dashboard.json

3.4 cQube Data Model

This section describes the student attendance, student assessment, infrastructure data and the teacher attendance data structure.

Tables classification

All the entities are classified into four types

- Static tables
- Metadata tables
- Hierarchical tables
- Dynamic tables
- Aggregated tables

Static tables: The tables which contain the static information of the entities like geo master, school master, student master, subject master

Metadata tables: The tables which are used to store the processing information of the emission process like incoming files and process status. To store the information of the days when the student attendance data is processed.

Hierarchical tables: These tables contain the rarely updated values or dimensions that change rarely such as student class, academic year.

Dynamic tables: These tables contain the daily or frequently changing values such as the student attendance, Teacher attendance, CRC data, Semester assessment.. These tables are updated more frequently.

Aggregated tables: These tables will hold the school-wise aggregated values which will be ready to be converted into JSON files. Multiple aggregation tables will be created based on the visualization report requirements.

Initialization stage for Infrastructure dataset: After the configuration stage is completed, the infrastructure master tables are initialized with the infrastructure columns of that state, based on the columns present in the infrastructure transaction table columns. Once initialization is completed the infrastructure weights can be configured through the emission API.

Click on the link below for the data dictionary:

https://docs.google.com/spreadsheets/d/1OM5jClFb3shyk0KKqXk0jtiThcwHntUXdzTyam_lwDo/edit?usp=sharing

Click on the link below for the latest ERD:

https://drive.google.com/file/d/1s1TJUs5c_IZ5zlQvpRWEnb4jWDVMgkI8/view?usp=sharing

3.5 Node connectivity with S3 bucket

1. Connectivity of aws s3 to nodejs is being done by an package called "aws-sdk"

2. Stored all the aws access related keys in the .env file

3. Maintained the config file to store all the s3 connect details (Ex: accessKeyId, secretAccessKey, bucketName)

4. Common function to read the files from s3 by passing the filename as a parameter to the s3 file reading code

Ex: s3Key parameter variable -> stores the filename of the every api request and pass to the function

```
const readS3File = (s3Key) => {
    return new Promise((resolve, reject) => {
        try {
            const_data['getParams']['Key'] = s3Key;
            const_data['s3'].getObject(const_data['getParams'], function (err, data) {
                if (err) {
                    logger.error(err);
                    reject({ errMsg: "Something went wrong" });
                } else if (!data) {
                    logger.error("No data found in s3 file");
                    reject({ errMsg: "No such data found" });
                } else {
                    var jsonData = JSON.parse(data.Body.toString());
                    resolve(jsonData)
                }
```

```
            });
        } catch (e) {
            reject(e)
        }
    })
}
```

4. Once the response received from the file, the required logics are done on the data set received from the s3 file

5. Logging the application information and error using the library called winston

6. All the data sent back as a json format to the Angular through the different apis.

3.6 AngularJS, chart js and leaflet roles in the visualization

1. Angular - is used as web application of the project where all types of visualization reside on.

2. All the headers and token for backend api are passed using HTTP Interceptors

3. The API calling functions is written in the different service files according to the reports.

4. Integrated the keycloak for authentication with the application

5. All the environment variables (backend api url, keycloak url, keycloak clientId, realm name) are stored respectively in environment.prod.ts file

6. Leaflet - The open source library has been used to show the visualization of maps related reports.

7. Chartjs - Has been used to show the visualization of Line charts, bar charts, scatter plots for the reports developed

8. Some of the reports are showcased using the bootstrap datatables in a tabular format.

3.7 Logs

Following list log files are linked in <base_dir>/cqube/logs directory

Nifi logs:

- <base_dir>/cqube/nifi/nifi/logs/nifi-app.log as nifi-app.log

- <base_dir>cqube/nifi/nifi/logs/nifi-bootstrap.log as nifi-bootstrap.log

Postgres logs:

- /var/log/postgresql/postgresql-10-main.log as postgresql-10-main.log

Emission app logs:

- <base_dir>/cqube/emission_app/python/access.log as emission_app-access.log

- <base_dir>/cqube/emission_app/python/error.log as emission_app-error.log

UI & Admin UI logs:

- /home/<system_user_name>/.pm2/logs/client-side-error.log as client_side-error.log

- /home/<system_user_name>/.pm2/logs/client-side-out.log as client_side-out.log

- /home/<system_user_name>/.pm2/logs/server-side-error.log as **server_side-error.log**

- /home/<system_user_name>/.pm2/logs/server-side-out.log as server_side-out.log

- /home/<system_user_name>/.pm2/logs/admin-client-side-error.log as admin_client_side-error.log

- /home/<system_user_name>/.pm2/logs/admin-client-side-out.log as admin_client_side-out.log

- /home/<system_user_name>/.pm2/logs/admin-server-side-error.log as admin_server_side-error.log

- /home/<system_user_name>/.pm2/logs/admin-server-side-out.log as admin_server_side-out.log

System logs:

- /var/log/syslog as syslog

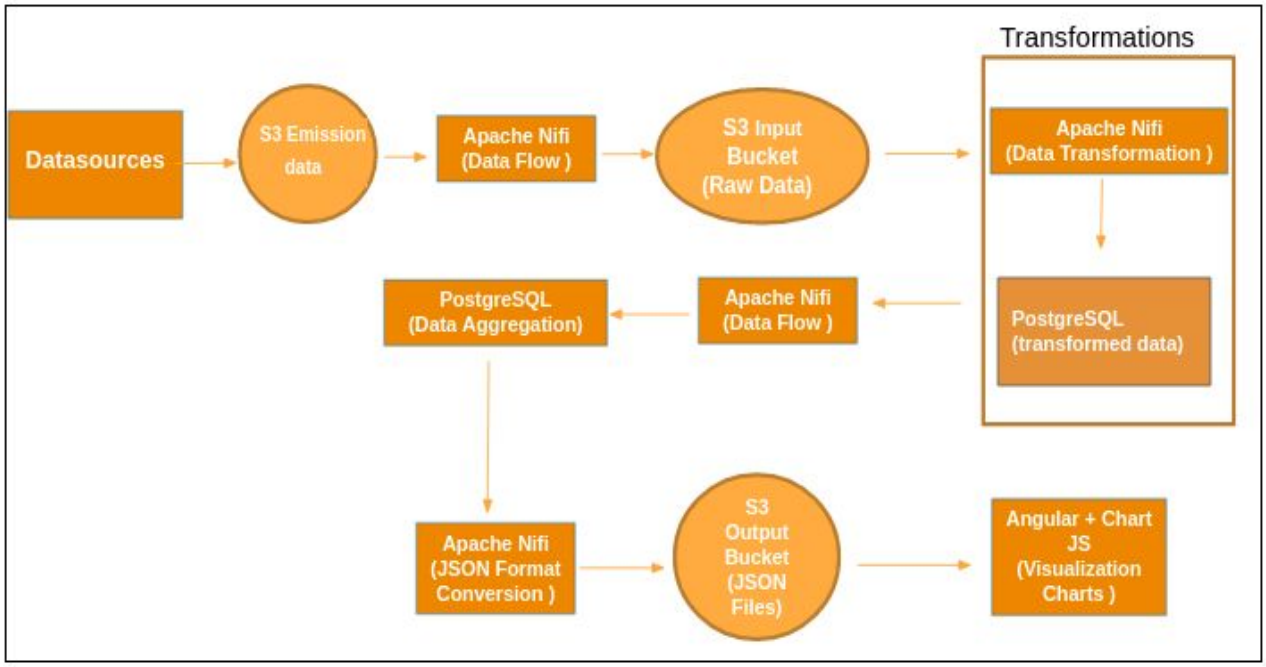
Keycloak logs:

- <base_dir>/cqube/keycloak/standalone/log/server.log as server.log

- <base_dir>/cqube/keycloak/standalone/log/audit.log as audit.log

4. cQube data flow

Figure – 4: cQube data process flow



Description of the data flow:

Table – 2: cQube data flow tables

S3 emission data	Is the location where the emitted data resides. This folder is created in AWS S3 Raw data files are emitted to S3 emission data buckets periodically.
S3 emission data - NIFI	NIFI collects the raw data files from the S3 emission data location and after that removes them from the S3 emission data folders.
NIFI	NIFI sends the raw data to the S3 (input) bucket. NIFI does the validation of data fields and values. NIFI inserts the transformed data to PostgreSQL. NIFI specifies the scripts to PostgreSQL to perform the necessary aggregations for visualization reports. NIFI picks the aggregated data from PostgreSQL and places them into the S3 bucket (output) to create JSON files which can be used for the visualization reports
PostgreSQL	PostgreSQL base tables are populated by NIFI PostgreSQL transformed tables are populated by NIFI PostgreSQL aggregate tables are populated by NIFI
S3 Output Bucket	S3 Output bucket contains the files in JSON format Angular reads the data from S3 output bucket by using Node API
Angular, ChartJS/ Leaflet	Angular fetches the data using Node JS from S3 to create reports using ChartJS/ Leaflet

5. Security Implementations

- **EC2**: A pair of public and private keys are generated, and the public key is stored in the EC2 server. The client with the private key gets authenticated with the server during login only if the keys match.
- **S3 emission data bucket**: cQube provides a data-ingestion API to emit the data. The API will have different secured endpoints like:
    1. User authentication
    2. API call to emit the data files using the https protocol into cQube.
    3. API takes the data file as a parameter.
    4. The API will provide acknowledgement on successful emission.
- **S3 Input & Output buckets**: A pair of access keys and secret keys are generated to secure the S3 location.
- **NIFI**: Only authorized users can gain access to the NIFI Dashboard. The confidential keys such as username and password will be encrypted.

- **PostgreSQL**: PostgreSQL will be secured as follows:
    1. User Access Control (RDAC)
    2. Server configuration
    3. User and role management
    4. Logging
    5. PostgreSQL audit extension (pgAudit)
    6. Security patches
- **Angular**: Role based authentication will be provided to prevent access to unauthorized users. A reverse proxy server has been used that usually stays behind the firewall of a private network. Reverse proxies are also used as a means of caching common content and compressing inbound and outbound data, resulting in a faster and smoother flow of traffic between the clients and servers.

- Role based authentication: Will be provided to admin users, and based on the user roles, relevant access will be provided to users at district levels, block levels, cluster levels and school levels.
- Normal/ regular users can access public reports without any authentication.

6 S3 bucket partitioning

S3 buckets will contain partitions for the data files to store. The partitions are created at the S3 input bucket & the S3 Output bucket.

6.1 S3 emission bucket partitions

- All the files in the S3 emission bucket will be in the CSV format.
- S3 emission bucket follows the folder hierarchy based on the data sources.
  S3 -> Bucket name -> Data source -> emitted zip files with timestamp
- The emitted zip file contains the CSV data files with timestamp and a manifest file with a timestamp.
- The folders and the files will be removed from the S3 emission bucket once NIFI copies the data.
- Unprocessed data files will remain in the S3 emission bucket for one week and then they will be deleted automatically at the end of the week.

6.2 S3 input bucket partitions

- All the files in the S3 input bucket will be in the CSV format.
- S3 input bucket follows a hierarchical partitioning based on the Data source, Year, Month, date and timestamp
  S3 -> Bucket name -> Data source -> Year -> Year - Month -> date_Source name

  Example for the S3 input bucket:

  S3/cqube-gj-input/student_attendance/2020/2020-05/2020-05-29_student_attendance

6.3 S3 output bucket partitions

- All the files in the S3 Output bucket will be in the JSON format.
- S3 output bucket follows the hierarchical partitioning based on the data source, Year, Month, date and timestamp, similar to the partitioning that the S3 input bucket follows.
- Metadata files will have information of the latest updated output files which helps cQube to consider the latest output file during the visualization stage.

7. cQube users - Technical activities

The cQube product can be used by a variety of users and each user will have different functions that he/she can perform and different user privileges The cQUbe users can be divided into the following categories:

1. Admin
2. Ad-hoc Analyst

The various roles and the functions that each role can perform have been described in the table below.

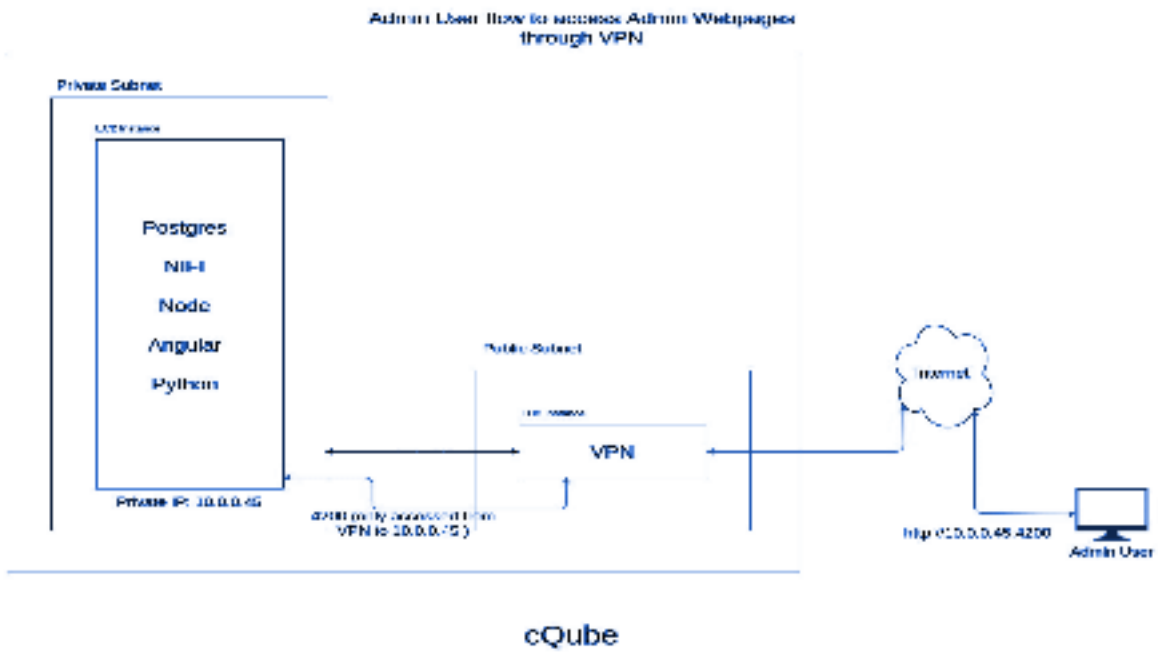
7.1 Admin

cQube has two different interfaces, an interface for the cQube administrator and another interface/ dashboard for the cQube reports:

- Administrator pages: Administrator activities will run separately in the VPN and admin must login through the 2 factor authentication process to perform the admin tasks.Change password functionality will be included in this login.
- cQube dashboard for report: This is a normal login which will have the cQube insights of the metrics.

7.1.1. Admin login process:

Figure - 7: Admin user flow through VPN



cQube

Admin must follow the 2 factor authentication process to perform the admin tasks. The Admin user should connect to the VPN to avail the 2 factor authentication by following the steps mentioned below:

- Admin has to request the devops team for OpenVPN access details.
- Admin has to download the google authenticator app to his phone and register the app with the QR code showing in the OPENVPN Access server page.
- By providing the credentials & Google authenticator code to download the user-locked profile(client.ovpn) file.
- Admin has to install the client openvpn-connect application
- Admin has to create the OpenVPN profile from the client.ovpn file
- Admin has to validate the user authentication and Google authenticator code to login to the cQube VPN.
- With the successful authentication admin can access the cQube admin features by opening the local ip in the browser.

7.2 Ad-hoc analyst

- They are the dynamic report creators of cQube.
- The ad hoc analyst can make use of third-party visualization tools like Metabase, Tableau or any other visualization tool to create dynamic dashboards and develop dynamic reports by directly accessing the database.
- Ad-hoc users can download the JSON files from the S3 output bucket through the API.
- The example spec for the download API will be as like below

```
headers = {'Authorization': 'Bearer access_token', 'Content-Type': 'application/json'}

GET https://cqube.tibilprojects.com/data/list_s3_buckets
{"input": "cqube-qa10-input", "output": "cqube-qa10-output", "emission": "cqube-qa10-emission"}

POST https://cqube.tibilprojects.com/data/list_s3_files
Body: { "bucket": "cqube-qa10-emission"}

POST https://cqube.tibilprojects.com/data/download_uri
Body:
{"filename":"school_master/2020/2020-06/2020-06-04_school_master/04-06-2020_13:12:59.574_5e160862-c5b3-4121-9a96-ecefa34fc264_school_mst.zip","bucket":"cqube-gj-input"}
```