



cQube

cQube – Operational Details Document

June 2020

Version 1.1

Document released by:
Sreenivas Nimmagadda

Document reviewed by:
Radhika Prabhu

Document acceptance by:
Arvind Gopalakrishnan

Document Version

Project	Release Date	Release Version
cQube	15.07.2020	V 1.0
cQube	20.08.2020	V 1.1

Table of Contents

1. Purpose of this document	5
2. Background	5
2.1 Use of this document	5
2.2 State of this document	5
2.3 Acronyms	5
3. Prerequisites for cQube installation	6
4. cQube Requirements	8
4.1 cQube product prerequisites	8
4.1.1 Software Requirements	8
4.1.2 Data Storage Locations	9
4.1.3 Software security requirements	9
4.1.4 Hardware requirements	10
5. cQube network setup	11
5.2 Public Subnet:	12
5.3 AWS Load Balancer:	12
6. cQube - Product Installation	13
6.1 cQube Installation	14
6.2 cQube - Configurations	14
6.3 Keycloak Setup	18
Single-Sign On:	18
6.3.1 Keycloak Two-Factor Authentication Configuration	19
6.4 Workflow	22
7. Data emission process	22
7.1 The steps involved in the data Emission Process	23
7.1.1 Emission APIs	24
7.1.2 Emission order	24
7.2 Emission file naming conventions & structure	24
7.2.1 static Files Structure:	25
8. Database Structure	26
8.1 Types of database tables	26
9. cQube NIFI data file processing and data validations	27
9.1 S3 bucket partitioning	28
9.1.1 S3 emission bucket partitions	28
9.1.2 S3 input bucket partitions	28
9.1.3 S3 output bucket partitions	28

9.1.4 Data Validation after Ingestion	28
<u>10. Admin login process & Features</u>	<u>30</u>
10.1 Admin login process:	30
10.1.1 Admin - Features	31
<u>10.2 Report Viewer</u>	<u>34</u>
<u>11. cQube release upgradation process</u>	<u>35</u>
<u>11.1 Prerequisites</u>	<u>35</u>
<u>11.2 Steps for cQube release Upgradation</u>	<u>35</u>
<u>11.3 Limitations</u>	<u>36</u>
12. Reports	36
12.1 Static Reports	36
12.2 Dynamic Reports	37

1. Purpose of this document

The purpose of this document is to describe the operational details of the cQube product. The operational details include description of functionalities like the cQube product installation, admin functionalities and upgradation of the cQube product. Some of the technical details like the cQube network setup and the data emission process have been added to this document to understand the operations clearly.

2. Background

cQube is an analytics product 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 the state/ districts/ clusters/ blocks/ villages and schools.

2.1 Use of this document

This document can be used by anyone who works with the cQube product and whoever has the permission to view the documents of the completed project. This document helps the user understand the design and architecture, as well as the operational and technical details of the cQube.

2.2 State of this document

This document is in a final draft version and is under change control. To request a change to the technical document please contact the system architect or the project manager.

2.3 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

3. Prerequisites for cQube installation

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 the 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	
Filing 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
	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 1.10	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

The table below gives the estimated time for the cQube network setup and product installation

S No	Task Details	Estimated Time(Approx.)
1	To create the EC2 Instance with Ubuntu 18.04	5 mins
2	For overall network setup	30 mins - 40 mins
3	cQube Installation	30 mins
4	Data emission for Student attendance, CRC and Semester assessment data	1 min with AWS otherwise it depends on the internet speed
5	Data processing time for Student attendance, CRC and Semester assessment data	35 mins

For more details of the data processing time please refer section 6

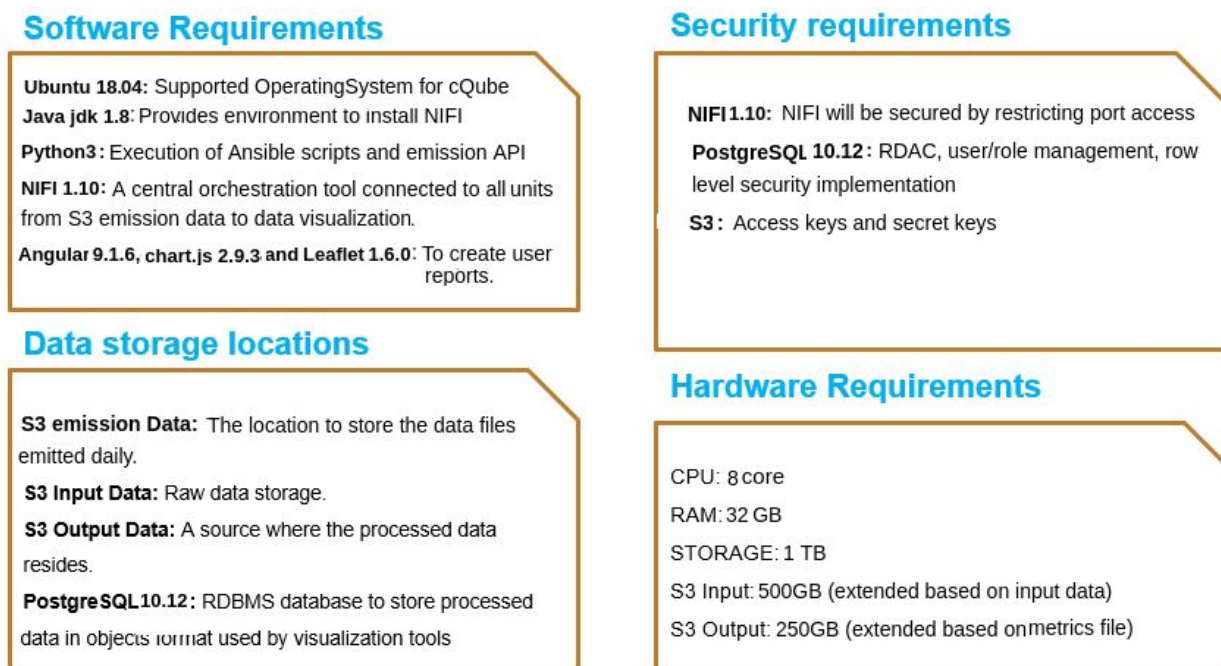
4. cQube Requirements

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

4.1 cQube product prerequisites

The cQube product installation process has a few system prerequisites that have 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



4.1.1 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 1.10: 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.

4.1.2 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 input 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.

4.1.3 Software 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.

4.1.4 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

(Currently, the S3 buckets will be created automatically during installation. This will be revised in future releases and the AWS Admin will have to create the S3 buckets before starting the installation)

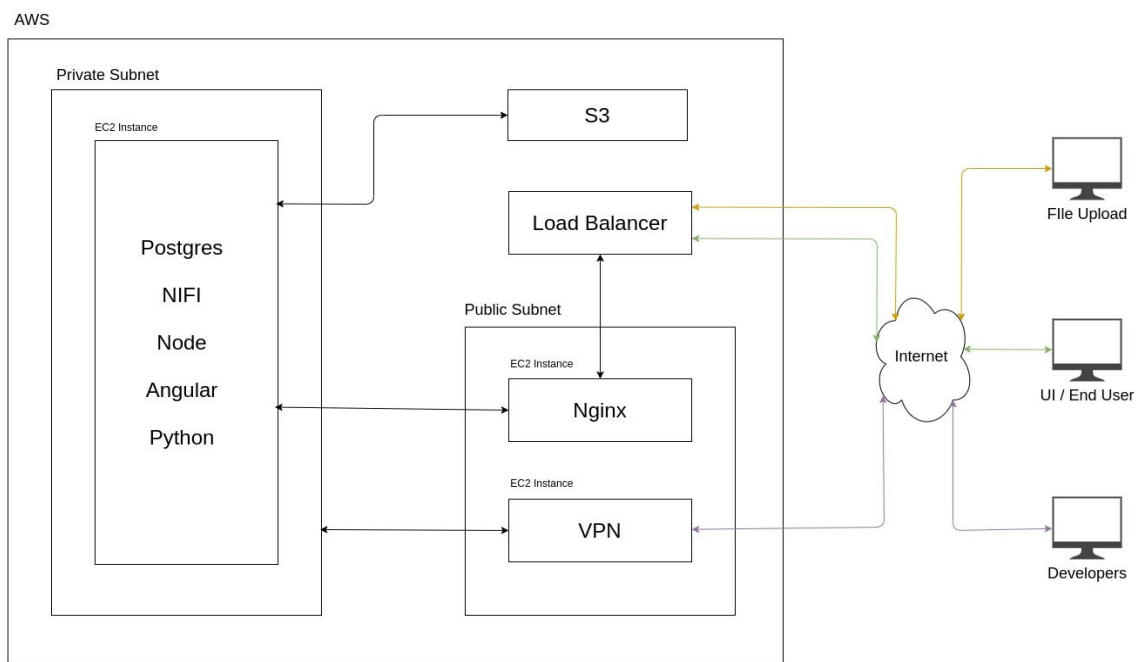
5. cQube network setup

The following steps define how the cQube setup and workflow is completed in AWS. cQube mainly consists of the areas mentioned below:

1. Private Subnet
2. Public Subnet
3. AWS Load Balancer

The cQube network setup process is described in the block diagram below:

Figure – 3: cQube network setup diagram



- The Yellow arrows in the network diagram indicates the file upload users' connectivity through the load balancer.
- The Green arrow in the network diagram indicates the end users connectivity through the load balancer.
- The purple colour arrow in the network diagram indicates the developers connectivity through the VPN.

5.1 Private Subnet:

Private subnet is used to secure the cQube server from public access. The instance will not be having the public ip. An EC2 instance will be created in a private subnet and all cQube components will be installed in this.

The steps involved to create EC2 instance in private subnet

- Create a virtual private cloud(VPC) in AWS
- Create a subnet in the created VPC with no Routing Table attached to Internet gateway
- Create an EC2 instance to install all the cQube software components.

5.2 Public Subnet:

Public subnet will contain two EC2 instances, one is for OpenVPN and another is for Nginx, which will act as a reverse proxy. It is used to provide connectivity with the private subnet.

The steps involved to create the public subnet

- Create a subnet in that same VPC where the private subnet was created with a Routing Table attached to the Internet Gateway.
- Create the first EC2 instance with OpenVPN AWS AMI and configure it to connect with the private subnet.
- Create the second EC2 instance with Ubuntu 18.04 AWS AMI and install Nginx to connect with the cQube server which is in the private subnet.

5.3 AWS Load Balancer:

AWS load balancer is used in cQube to avoid the security risks and also to control the traffic among the servers to improve uptime and easily scale the cQube by adding or removing servers, with minimal disruption to cQube traffic flows. cQube is using the Classic Load Balancer

The steps involved to create the load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/v2/home>
2. On the navigation bar, choose a Region for your load balancer. Be sure to select the same Region that you selected for your EC2 instances.
3. On the navigation pane, under LOAD BALANCING, choose Load Balancers.

4. Choose Create Load Balancer.
5. For Classic Load Balancer, choose Create.

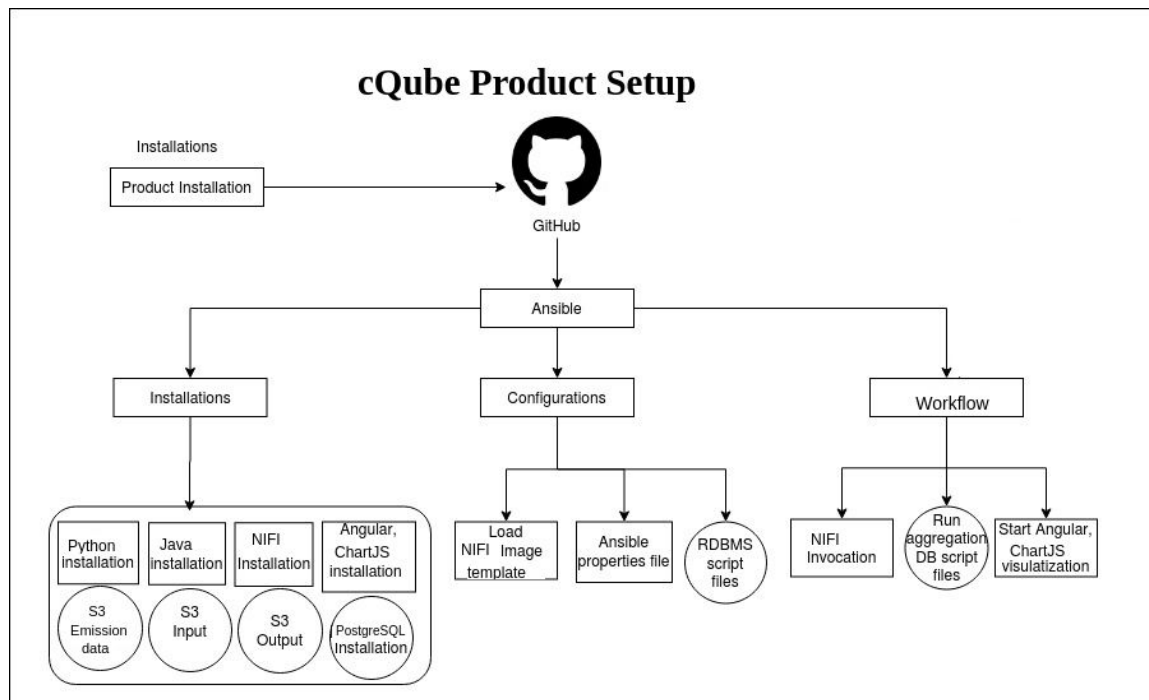
The steps involved to configure the load balancer with EC2 instance

- On the Configure Health Check page, leave Ping Protocol set to HTTP and Ping Port set to 80.
- For Ping Path, replace the default value with a single forward slash ("/"). This tells Elastic Load Balancing to send health check queries to the default home page for your web server, such as index.html.
- On the Add EC2 Instances page, select the Nginx instance to register with the load balancer.

6. cQube - Product Installation

The diagram below describes the cQube product installation process, the product configuration process and an overview of all the different software that have been used in the cQube tech stack.

Figure – 2: Flow diagram for the cQube product setup



6.1 cQube Installation

The cQube product can be installed with 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 approx 30 minutes.

6.2 cQube - Configurations

Configurations are done as a part of the installation process by the ansible scripts. All the properties of the sensible information like URLs and passwords are saved in the ansible properties file which is encrypted by default. Configuration is an automated process, no manual interactions are required in this stage.

cQube configurations through Ansible code:

- Nifi template would be uploaded & instantiated with the variables & parameters.
- All the Nifi controllers will be enabled and all the Nifi processors will be started.
- Postgres static, transaction, aggregation tables will be created in the newly created database.

Database configuration stage implementation:

cQube is having the flexibility to support multiple state data with minimal changes in UI code for the fluctuated data sources like Infrastructure, Infrastructure score weights, Semester assessment subjects.

Below Steps will be performed to implement the database configuration stage:

- After cQube installation is completed , cQube needs to be configured depending on the emission data fields and configuration file .
- During configuration users can activate/deactivate the data fields, but not the key columns (ex : student_id,school_id cannot be deactivated). Fields like subject names, infrastructure attributes for particular states can be activated/deactivated.
- Once the configuration file is emitted configuration process will be invoked
- In configuration stage fields will be activated/deactivated for that state, based on requirement
- Based on the configuration, the changes are made to handle state specific data fields
- After successful validations, data tables are created with only active and required data fields The active data fields are processed and metrics are generated
- The metrics generated will be stored in JSON files, which is used for visualization

For the database configuration stage implementation example:

- Based on the status, the active fields will be selected from the configuration table to create a transaction & aggregation table.
- The transaction and aggregation tables are created based on the above configuration.
- Based on the data fields selected in the configuration phase the database queries will be created for the active data fields.
- Metrics will be calculated only for the active data fields in the transaction table.
- The generated metrics will be stored in aggregate tables.
- The JSON specifications will be generated based on the active data fields in nifi.

Infrastructure configuration table

id	Infrastructure_name	infrastructure_category	score	Status
infrastructure_1	toilet	toilet	20	true
infrastructure_2	handwash	water	10	true
infrastructure_3	solar_panel	Energy resource	0	false
infrastructure_4	electricity	Energy resource	10	true
infrastructure_5	playground	play ground	20	true

infrastructure_6	digital_board	study	0	false
infrastructure_7	book_bank	study	0	false
infrastructure_8	drinking_water	water	20	true
infrastructure_9	hand_pumps	water	10	true
infrastructure_10	library	study	10	true

Infrastructure Transaction table

column	data type	constraints
school_id	bigint	primary key not null
last_inspection_id	bigint	
toilet	smallint	
handwash	smallint	
electricity	smallint	
playground	smallint	
drinking_water	smallint	
hand_pumps	smallint	
library	smallint	

Infrastructure Aggregation table

column	data type	constraints
school_id	bigint	primary key not null
last_inspection_id	bigint	

toilet_score	smallint	
handwash_score	smallint	
electricity_score	smallint	
playground_score	smallint	
drinking_water_score	smallint	
hand_pumps_score	smallint	
library_score	smallint	

- Calculation of infrastructure score is done based on the infrastructure availability of that school.
- If a school has infrastructure available then it will be considered for the calculation of score.
- For example if a school has drinking water, handwash, electricity, toilet, playground, hand pumps, library then the score would be calculated as below

$$20*1 + 10*1 + 10*1 + 20*1 + 20*1 + 10*1 + 10*1 = 100$$

- The total infrastructure score would be 100, if the school does not have any infrastructure available, then for that particular infrastructure it would be awarded with 0.
- If the school does not have playground, handpump, library, the score would be calculated as

$$20*1 + 10*1 + 10*1 + 20*0 + 20*1 + 10*0 + 10*0 = 60$$

- The total infrastructure score would be calculated to 60.
- If we need to calculate the score at different levels such as district, block, cluster, the school infrastructure count would be added for all the blocks, clusters, districts.
- To get the infrastructure score at district, block, cluster add all the infrastructure of all the schools available.

Weight * (infrastructure of all the school/ Total number of schools)

- For example if there are two schools in cluster the infrastructure score would be calculated as

$$20*(1+1)/2 + 10*(1+0)/2 + 10*(1+1)/2 + 20*(0+0)/2 + 20*(1+1)/2 + 10*(0+0)/2 + 10*(0+0)/2 = 55$$

- Infrastructure score for the cluster would be 55.

The Metrics will be stored in the JSON files for infrastructure visualization. An example JSON document for the district infrastructure is given below:

```
{
  "district": {
    "id": 2401,
    "value": "Kachchh",
    "block": {
      "value": "Abdasa",
      "id": 240107
    },
    "infra_score": {
      "value": "79.00",
      "average": {
        "value": "170",
        "percent": "79.81"
      },
      "total_schools": {
        "value": 213,
        "total_schools_data_received": {
          "value": "213",
          "handwash": {
            "value": "213",
            "percent": "100.00"
          },
          "solar_panel": {
            "value": "203",
            "percent": "95.31"
          },
          "library": {
            "value": "210",
            "percent": "98.59"
          },
          "drinking_water": {
            "value": "13",
            "percent": "6.10"
          },
          "tap_water": {
            "value": "13",
            "percent": "6.10"
          },
          "hand_pumps": {
            "value": "158",
            "percent": "74.18"
          },
          "playground": {
            "value": "159",
            "percent": "74.65"
          },
          "news_paper": {
            "value": "213",
            "percent": "100.00"
          },
          "digital_board": {
            "value": "3",
            "percent": "1.41"
          },
          "electricity": {
            "value": "207",
            "percent": "97.18"
          },
          "toilet": {
            "value": "213",
            "percent": "100.00"
          },
          "boys_toilet": {
            "value": "213",
            "percent": "100.00"
          },
          "girls_toilet": {
            "value": "213",
            "percent": "100.00"
          }
        }
      }
    }
  }
}
```

Steps to update infrastructure weights:

1. Download the existing infrastructure weights from api endpoint /infra_score
2. Modify the infrastructure scores and call the api endpoint /infra_weights

Ex: code [Snippet](#) to download the file & update the score.

6.3 Keycloak Setup

Keycloak is an open source Identity and Access Management solution aimed at modern applications and services. It makes it easy to secure applications and services with little to no code.

Single-Sign On:

Users authenticate with Keycloak rather than individual applications. This means that your applications don't have to deal with login forms, authenticating users, and storing users. Once logged-in to Keycloak, users don't have to login again to access a different application.

This also applied to logout. Keycloak provides single-sign out, which means users only have to logout once to be logged-out of all applications that use Keycloak.

Steps followed for keycloak implementation:

1. Install keycloak & configure postgres database as datasource.
2. Booting the Server
3. Create Realm
4. Create client and configure client
5. Create users
6. Create roles
7. User mapping to role
8. Integrate cqube dashboard, admin portal, flask api code with Keycloak server.

Installing Distribution Files:

- Download the Keycloak, unzip the keycloak-10.0.2.[zip|tar.gz] file.
- Download the postgres SQL jar file and configure the datasource and driver details in the standalone.sh file.
- boot the Keycloak server, go to the bin directory of the server distribution and run the standalone.sh.

6.3.1 Keycloak Two-Factor Authentication Configuration

Keycloak will allow single sign-on with Identity and Access Management services to cQube. Keycloak will be configured to cQube at the time of installation in the execution process of install.sh command.

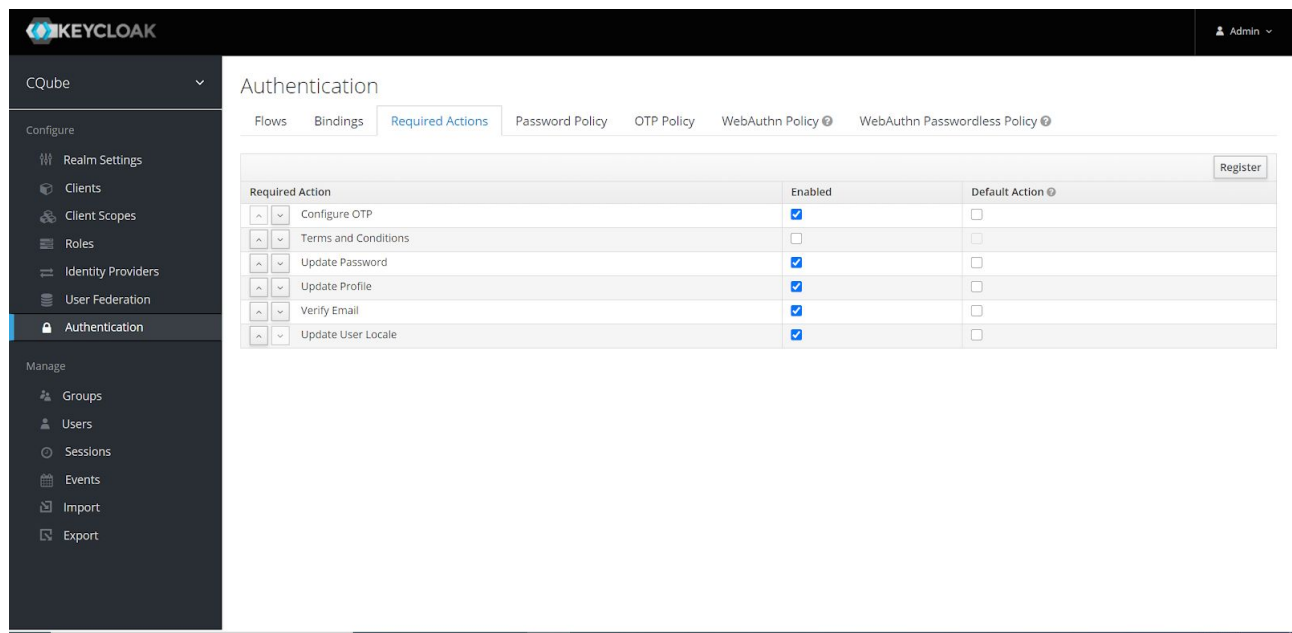
The manual configuration for the Keycloak is the two-factor authentication. The two-factor authentication will be provided as below.

Keycloak authenticates users using:

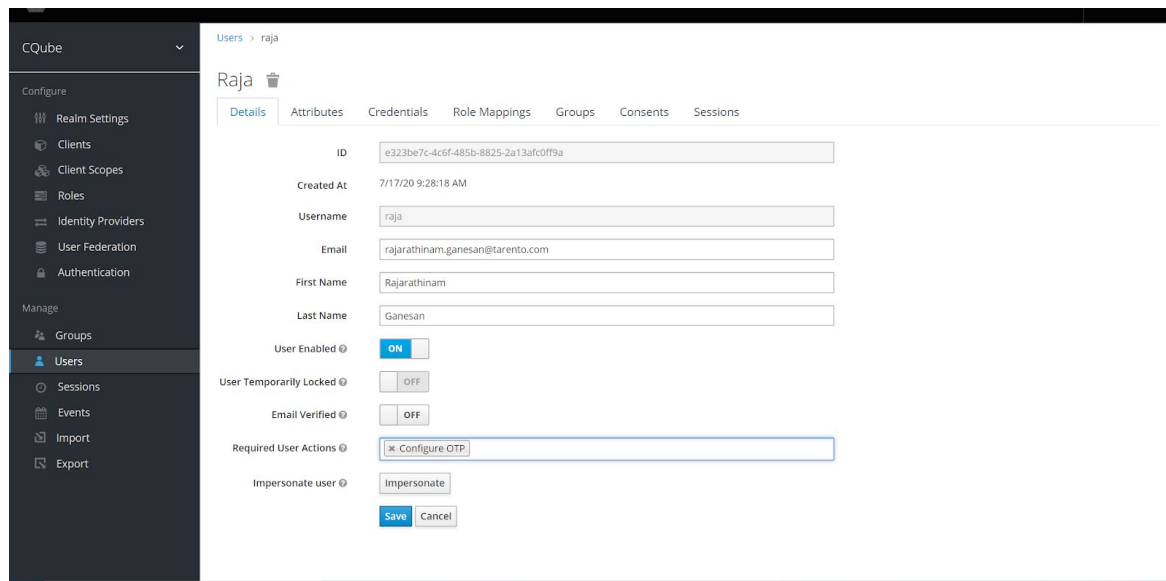
- Password
- One-time password (OTP)

To enforce new users to configure OTP, Open Keycloak admin page, open Authentication, go to the Required Actions tab. Click on the Default Action in the Configure OTP row.

Select cQube instead of Master

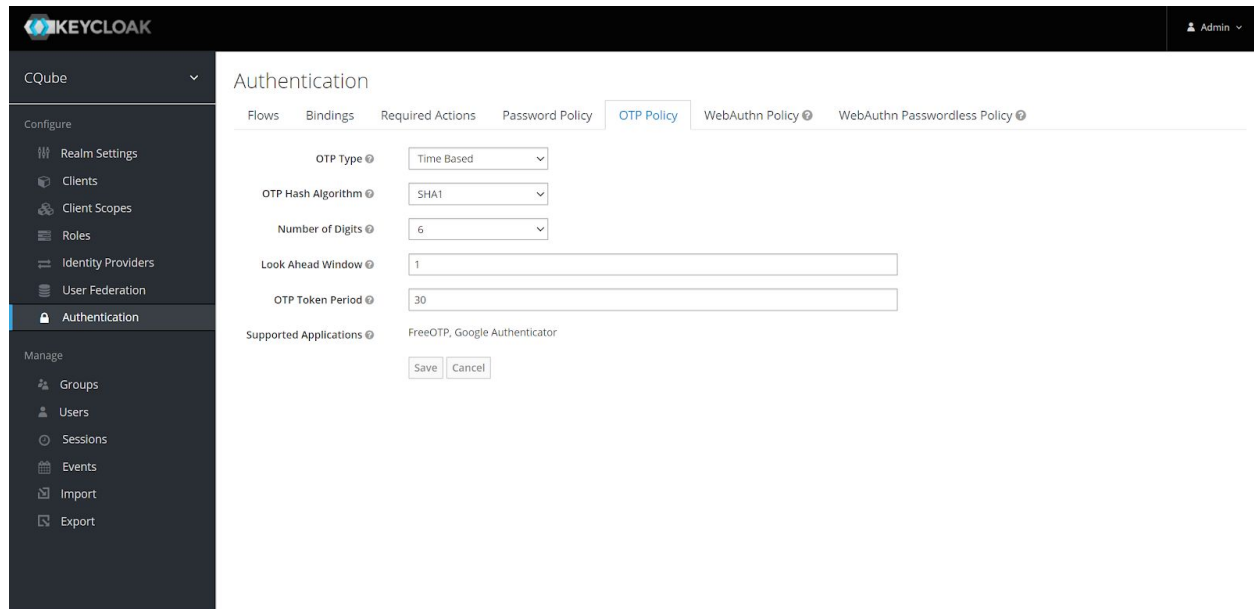


To Enforce an existing user to configure OTP, Open Keycloak admin page, open Users, select a user, go to the Details tab. In the Required User Actions list select Configure OTP.



For OTP Policy configuration, Open Keycloak admin page, open Authentication, go to the OTP Policy tab.

Select cQube instead of Master

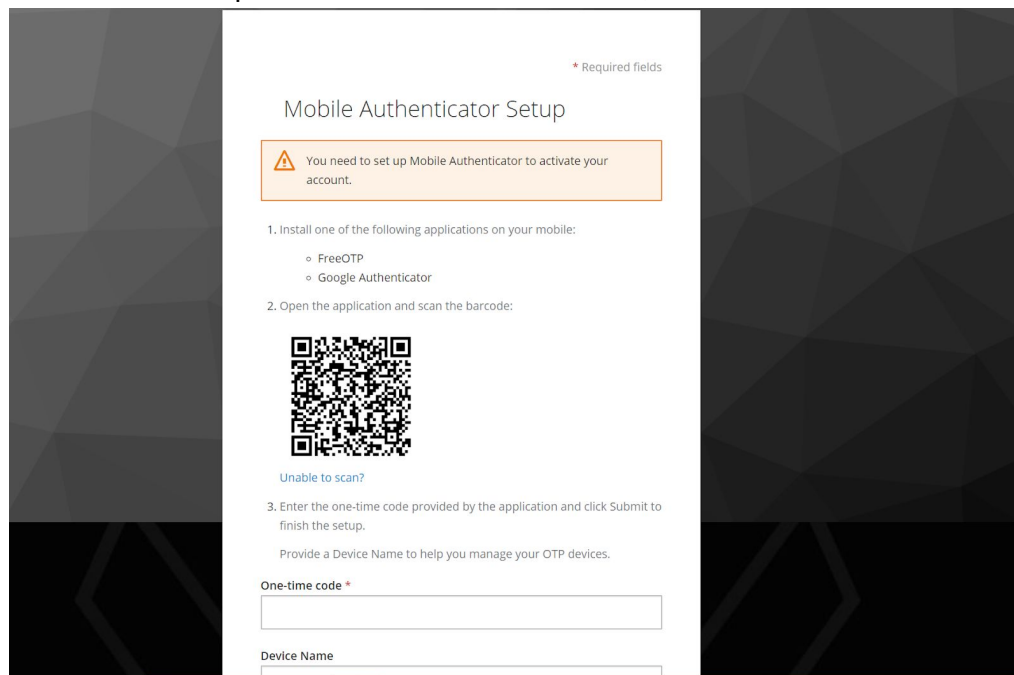


The screenshot shows the Keycloak Admin Console interface. The left sidebar has a 'CQube' dropdown menu. The main content area is titled 'Authentication' and has several tabs: 'Flows', 'Bindings', 'Required Actions', 'Password Policy', 'OTP Policy' (which is selected), 'WebAuthn Policy', and 'WebAuthn Passwordless Policy'. The 'OTP Policy' tab contains the following configuration options:

- OTP Type: Time Based (dropdown)
- OTP Hash Algorithm: SHA1 (dropdown)
- Number of Digits: 6 (dropdown)
- Look Ahead Window: 1 (text input)
- OTP Token Period: 30 (text input)
- Supported Applications: FreeOTP, Google Authenticator (text input)

At the bottom of the configuration area are 'Save' and 'Cancel' buttons.

For one-time setup of user's mobile authenticator,



The screenshot shows the 'Mobile Authenticator Setup' page. At the top right, there is a note: '* Required fields'. The main heading is 'Mobile Authenticator Setup'. Below the heading is a warning box with a triangle icon and the text: 'You need to set up Mobile Authenticator to activate your account.' Below the warning box are two numbered steps:

1. Install one of the following applications on your mobile:
 - FreeOTP
 - Google Authenticator
2. Open the application and scan the barcode:

Below step 2 is a QR code. Under the QR code is a link: 'Unable to scan?'. Below step 3 is the text: '3. Enter the one-time code provided by the application and click Submit to finish the setup.' Below this text is a prompt: 'Provide a Device Name to help you manage your OTP devices.' At the bottom are two input fields: 'One-time code *' and 'Device Name'.

Next steps are,

- Download Google Authenticator app into your mobile.
- Scan the QR code to make the integration of cQube with Google authenticator app.

- Provide the OTP to login.

6.4 Workflow

cQube workflow will be done in two parts

1. Data emission
2. Data processing

Data emission: Data from the state education system will be emitted into the S3 emission data folder using the cQube data emission API
emission API is described in the section 4

Data processing:

NIFI is the central orchestration system which handles the complete workflow process and generates JSON files. These JSON files are used for creation/ generation of the visualization reports. Data from the state education system will be emitted into the S3 emission data folder using the cQube data emission API.

The data processing is described in the section 4 of the technical document which can be accessed from below link

<https://docs.google.com/document/d/15oakVPuguAwc4oakeJPrLAiE4ipW4F8KFer85w2jjZ8/edit?usp=sharing>

More detailed information is available in section 7 of this document.

7. Data emission process

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

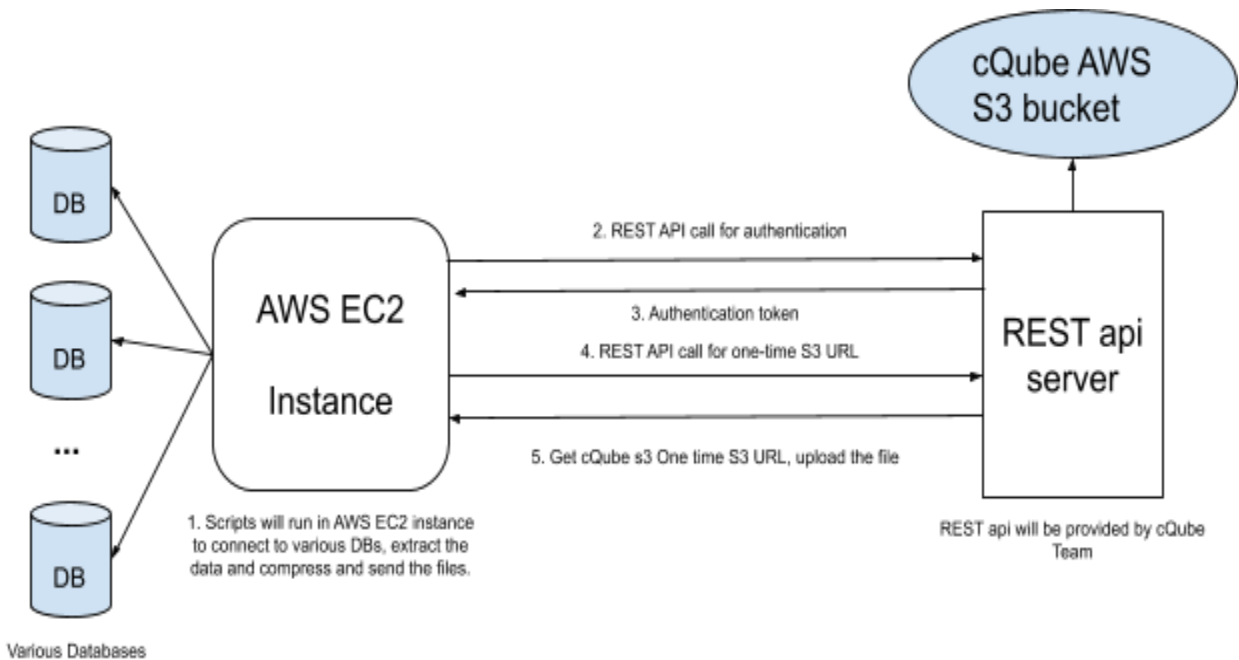


Figure – 4: Data emission process

7.1 The steps involved in the data Emission Process

- The data emission users will work from the data source location.
- Data emission users develop the automation extraction process which is outside cQube to extract the data from the sources.
- Data emission will be performed periodically as per the specified time interval from different data sources with the help of the above automated extraction process.
- Once the automation extraction process extracts the data fields which are required by cQube(Mentioned in section 4.2), it is converted into csv pipe delimited formatted files.
- The CSV data files will then be placed into the state data center by the automation extraction process.
- Emission process will be invoked by the automation extraction process once the extraction is completed.
- The emission process will be authenticated, An emission API will be invoked by authenticating the API token.
- The emission process makes an API call to generate AWS S3, one time presigned URL.
- API calls with the data file as a parameter to emit the data files using the https protocol into cQube.

- The emission details will be captured by the log files and will be available to the admin for review.

7.1.1 Emission APIs

- For configuring the emission job, fill the details in config.py like emission URL, password, file path which is available in the configuration file at below location
[cQube/development/python/config.py](#)
- After configuration to emit the files into the cQube system, invoke the
[cQube/development/python/client/client.py](#) program.

7.1.2 Emission order

- Static files which are defined in section 7.2.1 in this document need to be emitted first.
- For the CRC report, the inspection_master needs to be emitted prior to user_location_master to visualize the CRC reports.
- Below is the list of tables used in various reports.

Data Source Name	Used in report
district_master	SAR,CRC,SEM
block_master	SAR,CRC,SEM
cluster_master	SAR,CRC,SEM
school_master	SAR,CRC,SEM
inspection_master	CRC
user_location_master	CRC
student_attendance	SAR
infrastructure_master	INFRA
semester	SEM
diksha	DIKSHA

7.2 Emission file naming conventions & structure

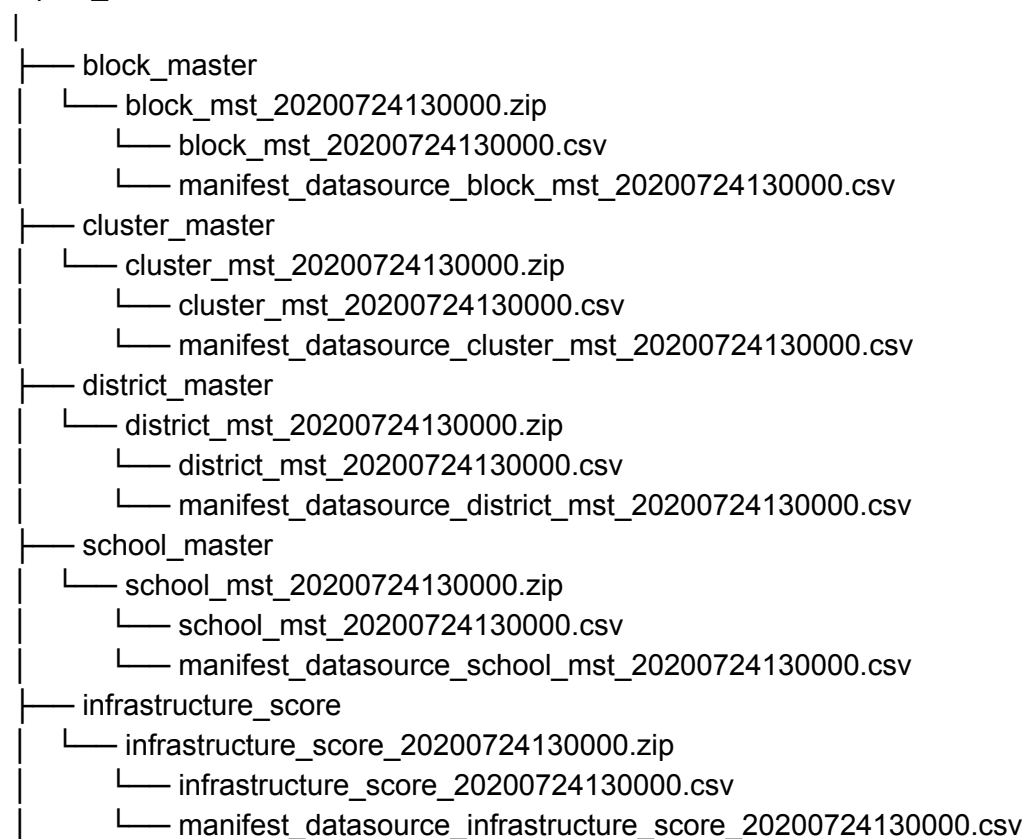
- The data files can be sent individually or all the files can be sent at a time, based on the requirement.

- The manifest file is mandatory for each of the dataset, it has to be named as 'manifest_datasource_' followed by the name of the dataset.
- All the files should be in a CSV format with PIPE("|") separated.
- The list of columns which have to be emitted for a data source are provided in the [file](#).
- The header should be in the same order as described in the [file](#), for that respective data set.
- The header for the manifest file is provided in the [document](#).
- Data emission overview of data flow can be viewed [here](#)

Naming conventions of the files and the file structure should be followed as mentioned below:

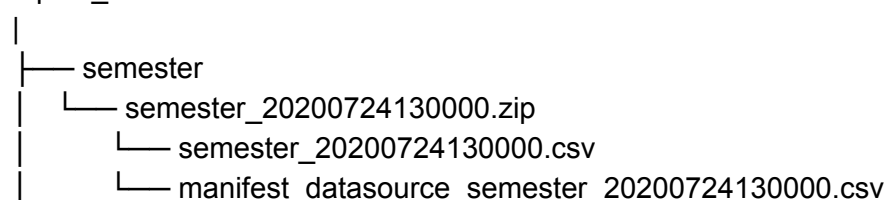
7.2.1 static Files Structure:

cqube_emission



7.2.2 Transactional Files Structure:

cqube_emission



```

├── student_attendance
│   ├── student_attendance_20200724130000.zip
│   │   ├── student_attendance_20200724130000.csv
│   │   └── manifest_datasource_student_attendance_20200724130000.csv
│   └── user_location_master
│       ├── user_location_master_20200724130000.zip
│       │   ├── user_location_master_20200724130000.csv
│       │   └── manifest_datasource_user_location_master_20200724130000.csv
│       └── inspection_master
│           ├── inspection_master_20200724130000.zip
│           │   ├── inspection_master_20200724130000.csv
│           │   └── manifest_datasource_inspection_master_20200724130000.csv
│           └── infra_trans
│               ├── infra_trans_20200724130000.zip
│               │   ├── infra_trans_20200724130000.csv
│               │   └── manifest_datasource_infra_trans_20200724130000.csv

```

8. Database Structure

This section describes the database structure, the types of database tables and the naming conventions

8.1 Types of database tables

The data tables are divided into static tables and transactional tables. All the data tables can be divided under these two table types as shown below:

Static tables:

- Block_master
- Cluster_master
- District_master
- School_master
- Infrastructure_score

Transactional tables:

- Student_attendance
- User_location_master
- Inspection_master
- Infrastructure_master

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/1s1TJU5c_Iz5zIQvpRWEb4jWDVMgkI8/view?usp=sharing

9. cQube NIFI data file processing and data validations

This section describes the cQube data file processing calculations and data validations. The files should be loaded into S3 emission bucket as per the emission process schedule timings.

The file processing will be controlled by NIFI and the process will be run as batch mode within the scheduled timing. The NIFI process will be run by the scheduler at 10PM every day as default. The administrator can change the time manually from the NIFI configurations.

All the NIFI process details and the processing time will be captured into the NIFI log files which will be available for the Admin to review.

Environment: EC2 instance

Hardware used: 4 core 16 GB Ram

Nifi memory: 8 GB

cQube performance summary					
Stage	Type	Zipped File name	Zip file Size	Records	Time taken
Files processed and time taken from API to S3 Output bucket	Static/Master files	school_master	3.2 kb	54728	
		cluster_master	32 kb	3247	
		block_master	1.3 kb	254	
		district_master	1.3 kb	38	
	CRC files	inspection_master	22.4 mb	497075	5 minutes
		user_location_master	57.8 mb	3054698	
	Semester files	semester	128.8 mb	3746754	10 minutes
	Student attendance files(one month)	student_attendance	250.4 mb	10473331	20 minutes

Overall time taken					~35 minutes
--------------------	--	--	--	--	-------------

9.1 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.

9.1.1 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.

9.1.2 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`

9.1.3 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.

9.1.4 Data Validation after Ingestion

File size mismatch: 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 check for the emitted file: 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, 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.

Validation for duplicate records: Record which have duplicate values for all fields (mirror image record) then NIFI will consider the first record and the rest of the records will be eliminated.

Records with duplicate ID: 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.

Example: For the Duplicate records with different lat long details, NIFI eliminates the records which have the same id and different lat long details or different names or different values.

Records with same values: 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)

All the validation results will be captured into the NIFI log files as per the table below,

SI No	Validation type	Error message
-------	-----------------	---------------

1	File size mismatch	"ERROR: file size mismatch for <filename>"
2	Record count check for the emitted file	"ERROR: record count mismatch for <filename>"
3	Validation for duplicate records	"ERROR: duplicate key value violates the data_source" along with record
4	Records with duplicate ID	"ERROR: Detail: Key (school_id)=(<id>) already exists"
5	Records with same values	"ERROR: duplicate key value violates data_source" along with record
6	Overlapping data validation	"ERROR: duplicate key value violates data_source" along with record

Other errors will be displayed with appropriate "ERROR: <error message>".

Admin dashboard will contain the details of the results of the validations as

1. The last files uploaded with the file name and timestamp
2. Total number of records in the file
3. Number of successfully uploaded records
4. Number of duplicate records
5. Number of invalid records.

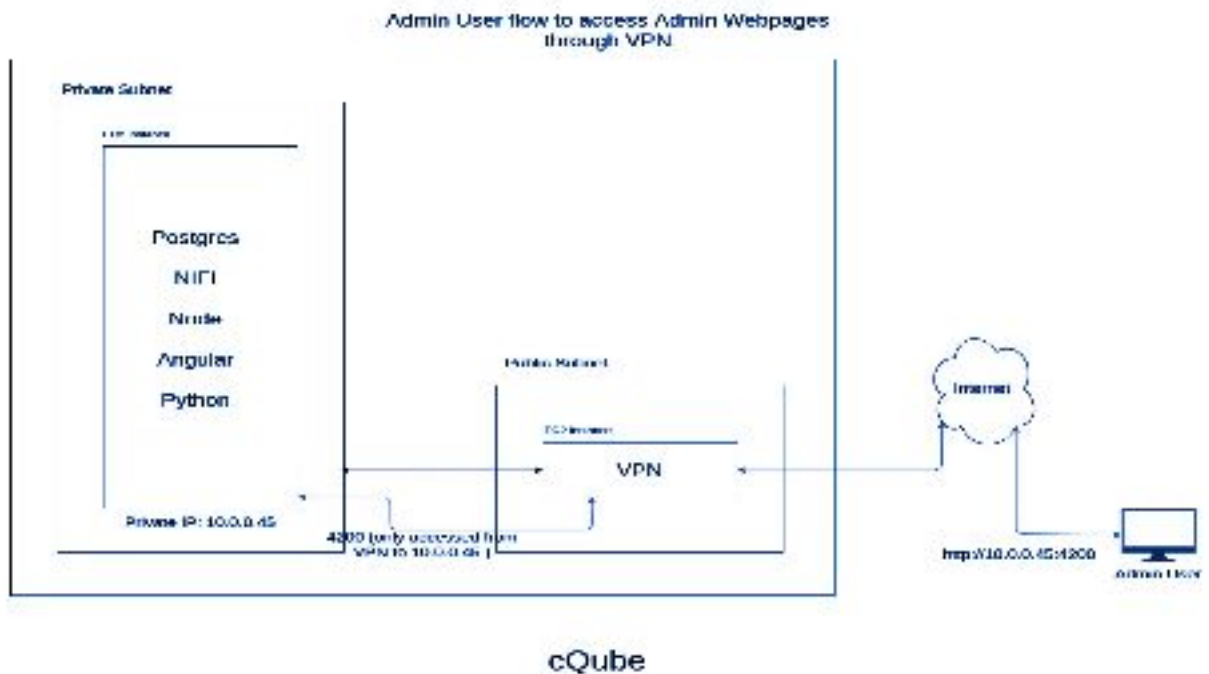
10. Admin login process & Features

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.

10.1 Admin login process:

Figure - 5: Admin user flow through VPN



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 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.

10.1.1 Admin - Features

1. **Activate, Deactivate and Create users:**

- Admin can create new cQube users from the user interface page.
- Admin can search for the user

- Admin is able to activate and deactivate the users by updating the start date / end date in the calendar field from the user interface page.

2. **S3 buckets** - Admin can perform view files and download actions in the S3 buckets by the following steps:

- By using the user interface admin is able to view the raw data files from the S3 input bucket, Metrics files from the S3 output buckets, Transformed and Aggregated data files from the S3 output bucket files.
- Admin can able to download the metrics files in the JSON format, transformed and Aggregated data files in CSV format from the S3 output buckets
- Apart from the user interface Admin users can call the cQube API to download the metrics files in JSON format.
- They can use the same download API to download the transactional data files and aggregated data files in CSV format from the S3 output bucket.
- The admin users use the API to download the raw files from the S3 input bucket in the zip format also.

Admin can enter the AWS console and perform the files cleanup, files Download activities

- Raw data will be downloaded in CSV format and the metrics data will be downloaded in JSON format.

3. **Database Controls** - Admin can perform the database actions mentioned below:

- Delete data
- Truncate tables
- Partitions handling operations
- SQL dump
- [Restoring the Dump](#)
- Data Archive

The steps mentioned below describe how the admin user can perform the above mentioned operations:

- After logging in, the admin will be provided with a user interface page to perform these operations.
- Admin can select the desired operation from the options provided.
- Admin has to select the database schema and choose the tables to perform the operations.

- Admin should get the options to kill the active database sessions when required.
 - All operations can be executed as mentioned below,
 - Admin will select the table which he wants to perform operations like Archiving data, Downloading the data and Dumping the data.
 - Admin will click on the required action button
 - An API will be executed which is called a shell script to execute the desired functionality.
4. **Application Controls:** Admin can perform the operations mentioned below using the buttons provided:
- Emission API start, stop and restart
 - AngularJS & NodeJS start, stop and restart
 - PostgreSQL start, stop and restart
 - Nifi start, stop and restart
5. **View Logs:**
- All the log files will be consolidated into a common folder using soft links.
 - New log files will be created every day and the old logs are retained for 7 days. At any point of time, there are 7 log files in the folder, old files are deleted when new files are created.
 - Admin will have an option to view the last 500 lines on the browser with the help of the user interface provided. There will be a download option provided to download the entire log file.
 - Log files will be downloaded in .log format.
 - The list of log files provided to the admin are as follows:
 - System logs
 - NIFI logs
 - Data emission process logs
 - Database logs
 - User Interface logs (Angular and Node logs)
 - S3 logs can be accessed using AWS cloud watch by having the AWS IAM (Identity and Access Management) user permissions.

Grafana Dashboard Connectivity:

Admin is having an user interface screen "Monitoring Details" screen in the admin dashboard, which connects to Grafana. And also Admin can use the Prometheus APIs to integrate with Grafana for

viewing the cQube application health. The following details are displayed in the grafana dashboard:

- RAM usage
- CPU Usage
- Data storage disk usage
- Nifi heap memory usage
- Data source file metrics
- JVM usage

10.2 Report Viewer

- They are defined in a hierarchical manner,
For example: If the users belong to the State administrative office, they can view the insight statistics of the districts that belong only to that state.
If the users belong to the district administrative office, they can view the statistics of the blocks which belong to the district.

The hierarchy is as shown below:

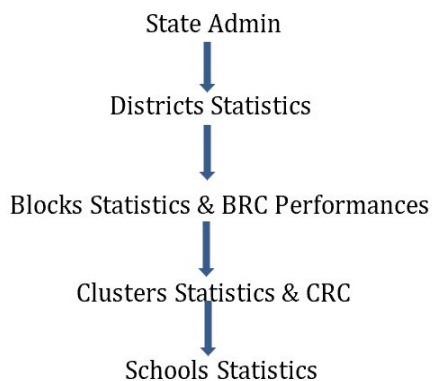


Figure – 6: Report viewer hierarchy

- They can view static dashboards according to their access scope. The selection option dropdown or drilldown options will be provided according to the access scope.
- They will not be able to access any other dashboards which are outside their scope.
- They can have insights into the performance and identify the low and high performers based on the benchmark in the dashboard.

- They can communicate with the users at lower hierarchical levels and provide them with the actionable items or exceptional performance via email (functionality provided in the dashboard)
 - They can alert the low performing district/ block/ cluster/ school based on their performance
 - They can notify the average performing district/ block/ cluster/ school based on their designation, to improve the performance
 - They can recognize the high performing district/ block/ cluster/ school based on their designation, to maintain the performance

11. cQube release upgradation process

11.1 Prerequisites

- A new release branch will be created in Github.
- The latest code of cQube should be committed to the new branch of Github repository
- The existing data structure should be placed as SQLscripts in the new branch of Github repository.
- The latest data structure for cQube should be placed as SQLscripts in the new branch of Github repository.

11.2 Steps for cQube release Upgradation

- Each release will have to execute an upgrade.sh file.
- upgrade.sh file will call the scripts of ansible.
- Ansible scripts will perform the updates as mentioned below:
 - An ansible script will run to take data backup from existing data.
 - An ansible script will run the NIFI toolkit. By executing the NIFI toolkit, the NIFI will update the latest flow version to the release environment.
 - An ansible script will be run to update the AngularJS code by getting the latest code from Github
 - An ansible scripts will be execute the latest SQL scripts by comparing the below database structural changes
 - SQL scripts will check each table from the SQL backup file with the Database schema

- If the table is already available, All the table columns will be compared.
- If there are no changes in the existing table & table fields, then the table will be skipped and no changes will be made on the existing table.
- If there are any changes in the existing table fields, the Ansible scripts run the alter query to perform the changes on the table
 - If the table does not exist, Then Ansible scripts perform the create the new table using create table query.
- S3 buckets data will not be touched if there are no changes in the existing tables. All the metrics and the calculated data will remain the same.

11.3 Limitations

- Metrics will be regenerated if there are any changes in the existing table fields such as adding/deleting the new columns to the existing tables.
- Metrics will be regenerated if there are any structural changes at the data source.
- The output files should be regenerated with the recalculated metrics by having the modified fields information.

12. Reports

cQube generates two types of reports

1. Static Reports
2. Dynamic Reports

12.1 Static Reports

- Static reports will be generated using Node JS, Angular JS, and Chart JS.
- Node JS provides the connectivity to S3 output bucket and fetch the aggregated data files
- Static reports generated using the metrics which are available in the S3 output buckets.
- Admin users and Report viewer users can view these reports to understand the metrics.
- Report Creator users create the customized new metrics or reports using the existing report Analytics.
- Static Reports will have the drill down functionality from state level to school level.

- Static reports will have the data download capability.
- Static report access will be provided as per the user levels (State, District, Block, Cluster or School level)

12.2 Dynamic Reports

- Dynamic reports will be generated using Tableau or any other related tools.
- Dynamic reports can use VPN to access the database tables.
- Ad-hoc users can create these reports by using the real time metrics.
- Ad-hoc users will have the access to the required tables only.