Integrated Public Insight and Data Platform

Data Collection:

There are several wa1ys to collect citizen feedback depending on the context and the purpose of gathering feedback. Here are some effective methods:

1) **Surveys and Questionnaires:** Surveys can be conducted online, over the phone, or in person to collect feedback from citizens. You can design and distribute surveys through various channels such as websites, social media, email, or even physical mail.

2) **Public Meetings and Town Halls:** Organizing public meetings or town halls provides an opportunity for citizens to share their feedback in person. These events can be held at local community centers, government buildings, or online platforms to allow for broader participation.

3) **Online Platforms and Websites:** Creating dedicated online platforms or websites where citizens can provide feedback is an effective way to engage a large number of people. These platforms can include forms, discussion forums, or comment sections to gather feedback on specific topics.

4) **Social Media:** Utilizing social media platforms such as Twitter, Facebook, or Instagram allows you to reach a wide audience and collect feedback in real-time. Create polls, ask questions, or encourage citizens to share their opinions through comments or direct messages.

5) **Focus Groups:** Conducting focus groups involves gathering a small group of citizens to have a structured discussion on specific topics or policies. This method allows for in-depth exploration of opinions and experiences.

6) **Mobile Applications:** Developing mobile apps dedicated to collecting citizen feedback can provide a convenient and accessible way for people to share their opinions. These apps can include features like surveys, feedback forms, and notification systems for updates or new initiatives.

7) **Feedback Boxes and Suggestion Boxes:** Placing physical or digital feedback boxes in public spaces, government offices, or community centers can encourage citizens to provide anonymous feedback or suggestions.

8) **Citizen Hotlines:** Establishing dedicated phone lines where citizens can call and provide their feedback or voice their concerns is another way to gather citizen input.

9) **Email and Postal Mail:** Providing email addresses or postal mail addresses for citizens to send their feedback is a traditional yet effective method. Ensure clear instructions on how to provide feedback and make it easily accessible.

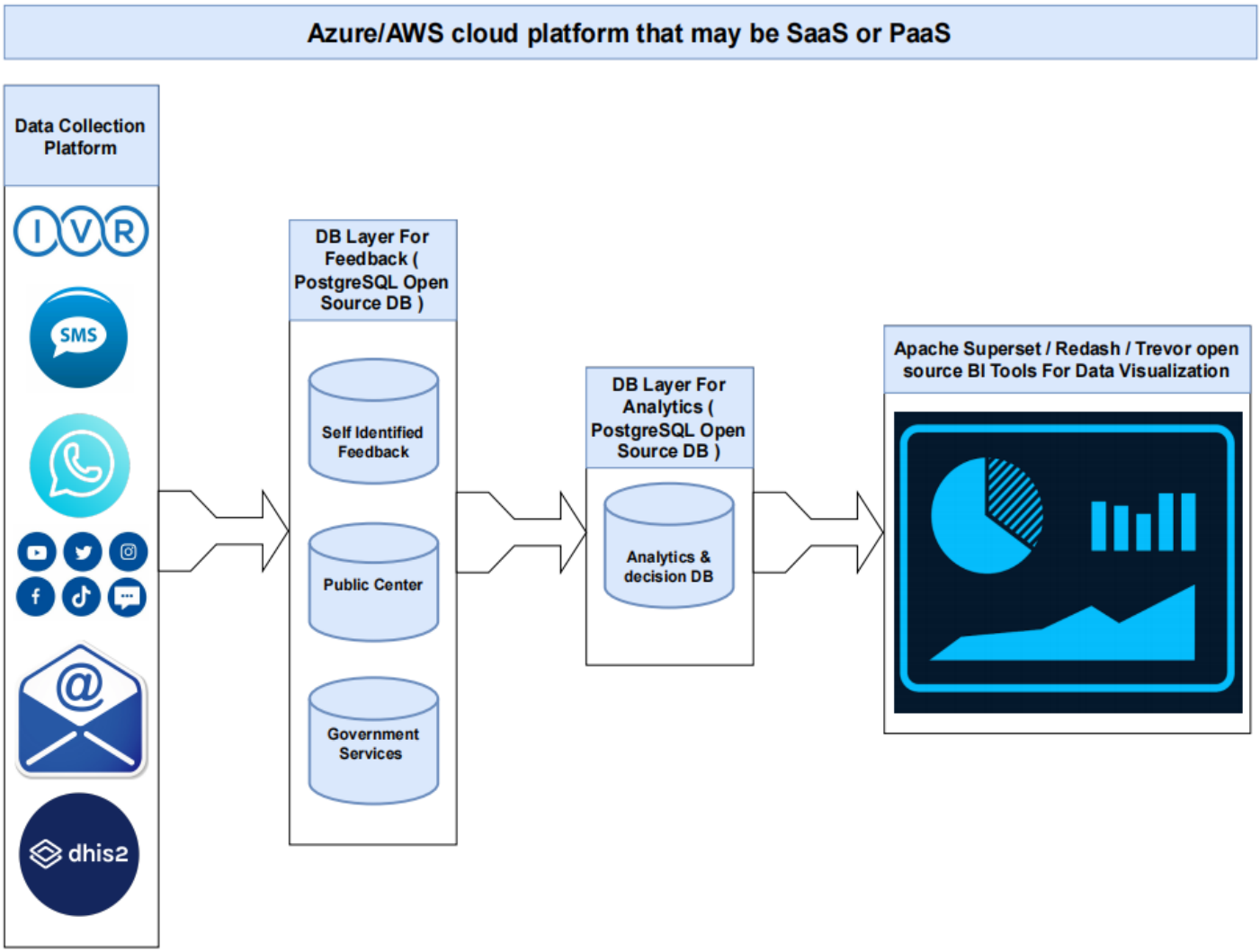

Figure: Combined diagram of Source Data Collection, Data Storage, Processing, and Visualization

Proposed Technology: We will collect data from different sources like Interactive Voice Response System, Short Message System, Social Media Platform, E-mail Feedback, Different type of Mobile Applications (DHIS) where users are freely able to express their opinions.

So, we will collection these type of data through API Integration as well as data scraping from various media. To maintain all the above process, what open-source tools, we are supposing to use that are enlisted below:

1) **For data collection:** Most of the data will be generated from Citizen Feedback. But, if we need more data then we will collect these types of data from social media scraping, web scraping, e-mail feedback, etc. This will be done by **Python because Python is an Open-Source Scripting Language**.

   Python can easily process billion of records with high throughput ration. If need any data cleaning issue is raised then using python panda's library, we can do that run time. So, for data scraping, processing and cleaning purpose, there is no comparable tools with Python.

2) **Data Storage:** To store the collected data from version source, we will use PostgreSQL because it's open source. PostgreSQL can store **32 TB** data easily and its query processing engine is like Oracle. **So, as an open source, PostgreSQL is the best solution for Database**.

   Also, we can use **PostgreSQL Database as Analytics Database**.

3) **Data Visualization:** There are many open sources data visualization tools like Power BI, Tableau, Quick View. But these are paid tools. So, for open source, we can use **Apache Superset, Redash or Trevor**.

   The main advantage of Apache Superset is that it can handle near real time data visualization. And Redash or Trevor are also good for API integration and large volume data management.

So, from the above discussion, we are supposing to use, all are free and open-source tools for every steps. To maintain all resource, we can consider **Azure Cloud Platform or Amazon Web Service**.

Cost analysis:

Azure: If we consider a Virtual Machine of **64GB RAM** and **1TB SSD** then cost may be **$320/month** (US East Region). It may vary region to region.

AWS: If we consider a Virtual Machine of **64GB RAM** and **1TB SSD** then cost may be **$350/month** (US East Region). It may vary region to region.

Open-Source Technology Sync up into Cloud Platform

A) To configure an Azure virtual machine (VM) for a PostgreSQL database

1) **Provision an Azure virtual machine:** Sign in to the Azure portal and create a new virtual machine. Select the desired VM size, operating system (e.g., Linux or Windows), and other configuration options. Ensure that the VM has sufficient resources to run PostgreSQL.

2) **Connect to the VM:** Once the VM is provisioned, establish a remote connection to the VM using an SSH client (for Linux) or Remote Desktop (for Windows). This will allow you to interact with the VM's operating system.

3) **Update the system:** It's recommended to update the system packages before proceeding with any software installation. Run the relevant package manager commands to update the OS and install any necessary updates.

4) **Install PostgreSQL:** Use the package manager of your operating system to install PostgreSQL. For Linux distributions like Ubuntu or CentOS, you can use apt or yum respectively. On Windows, you can download the PostgreSQL installer from the official website and run it.

5) **Configure PostgreSQL:** Once PostgreSQL is installed, you need to configure it. The configuration files are typically located in the /etc/postgresql directory on Linux or the installation directory on Windows.

6) **postgresql.conf:** This file contains various settings for PostgreSQL. You may need to adjust parameters like listen_addresses to specify the IP addresses on which PostgreSQL will listen for connections, max_connections to control the maximum number of concurrent connections, and shared_buffers to allocate memory for caching.

7) **pg_hba.conf:** This file manages client authentication. Configure the allowed IP addresses or ranges, authentication methods (such as passwords or certificates), and access privileges for different users and databases.

8) **Firewall settings:** Ensure that the VM's firewall allows incoming connections to the PostgreSQL port (default is 5432) if you want to access the database from outside the VM. You can configure the firewall rules using the operating system's built-in firewall or any network security group associated with the VM in Azure.

9) **Start and test PostgreSQL:** Once the configuration is complete, start the PostgreSQL service. On Linux, you can use the systemctl command (e.g., sudo systemctl start postgresql) to start the service. On Windows, you can start the service from the Services management console.

Verify that PostgreSQL is running correctly by connecting to it from the VM itself or using a remote client. You can use tools like psql (command-line interface) or graphical clients such as pgAdmin to interact with the database.

Remember to follow security best practices, such as using strong passwords, regularly applying updates, and implementing appropriate network security measures to protect your PostgreSQL database on the Azure VM.

B) Ensure PostgreSQL DB Security:

1) **Restrict network access:** Utilize Azure Network Security Groups (NSGs) to limit inbound and outbound traffic to the VM. Configure NSGs to allow only necessary ports (e.g., PostgreSQL default port 5432) and IP addresses.

2) **Virtual Network (VNet) service endpoints:** Leverage VNet service endpoints to secure access to the PostgreSQL server within the virtual network. This allows direct access to the server without exposing it to the public internet.

3) **Private Link:** Consider using Azure Private Link to securely access the PostgreSQL server over a private network connection. This eliminates exposure to the public internet and improves security.

4) **Encryption in transit:** Enable SSL/TLS encryption to secure client-server communication. Configure PostgreSQL to accept only encrypted connections, and use valid SSL/TLS certificates for encryption.

C) Apache Superset Install and Maintenance in Azure VM

Apache Superset has several dependencies that need to be installed. These typically include Python, pip (Python package manager), and other libraries. Install these dependencies using the appropriate package manager or by downloading them from their official websites.

1) **Create a virtual environment (optional):** It is recommended to set up a virtual environment to isolate the Apache Superset installation from other Python packages on your system. You can create a virtual environment using tools like venv or virtualenv.

2) **Install Apache Superset:** Use pip to install Apache Superset and its dependencies. Run the command pip install apache-superset to install the latest version of Superset. If you want to install a specific version, you can specify it in the command (e.g., pip install apache-superset==1.0.0).

3) **Set up the database:** Apache Superset requires a database to store its metadata and configurations. You can choose a database like PostgreSQL, MySQL, or SQLite. Install and configure the chosen database on the VM, and create a new empty database specifically for Superset.

4) **Initialize the Superset database:** Run the command superset db upgrade to initialize the Superset database schema and tables. This will create the necessary tables for Superset to store its metadata.

5) **Create an admin user:** Run the command superset fab create-admin to create an initial admin user for Superset. Provide the required information, such as username, email, and password.

6) **Initialize Superset assets:** Run the command superset load_examples to load example datasets and dashboards into Superset. This step is optional but can help you explore the capabilities of Superset.

7) **Start Apache Superset:** Run the command superset run -p 8080 --with-threads --reload --debugger to start the Superset server. This will start the Superset web application on port 8080 of the VM.

8) **Access Apache Superset:** Open a web browser and enter the IP address or domain name of your Azure VM, followed by the port number (e.g., http://<vm-ip>:8080). You should see the Apache Superset login page. Use the admin credentials created in step 5 to log in.

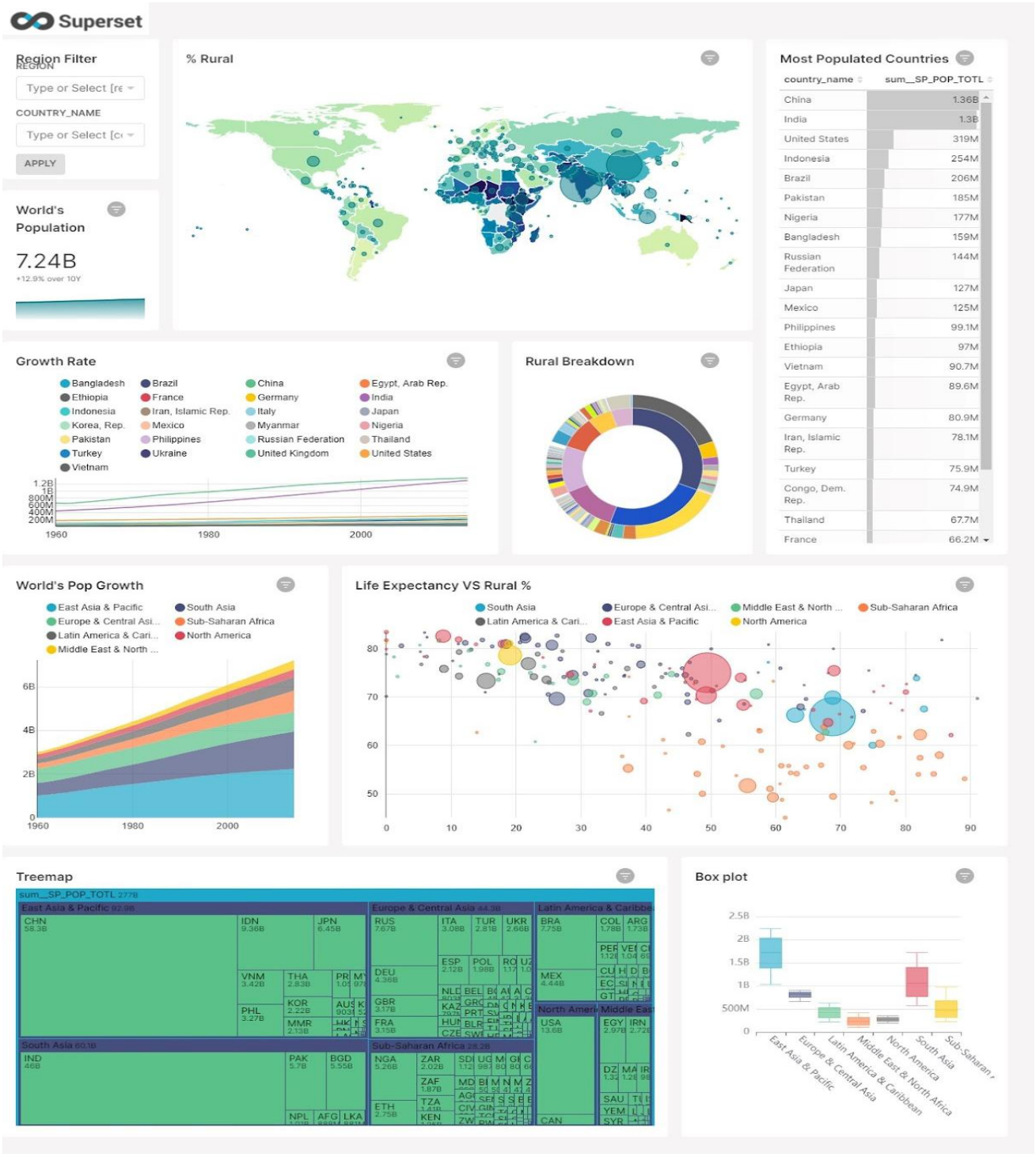

Figure: Example of Apache Superset Data Visualization

D) Create a New Data Source in Redash

1) Log into Redash, click on your profile and click "Data Sources"
2) Click the " New Data Source" button
3) On the configuration tab, set the following properties:
    - ✓ **Name**: Name the data source (e.g. PostgreSQL (CData Connect))
    - ✓ **Host**: The full URL to your CData Connect instance (e.g. https://myinstance.cdatacloud.net)
    - ✓ **Port**: The port of the CData Connect PostgreSQL endpoint (e.g. 3306)
    - ✓ **User**: A CData Connect user
    - ✓ **Password**: The password for the above user
    - ✓ **Database name**: The name of the virtual database for PostgreSQL (e.g. PostgreSQL1)
    - ✓ Click the checkbox to Use SSQL
4) Click Create
5) Click the "Test Connection" button to ensure you have configured the connection properly

With the new Data Source created, we are ready to visualize our PostgreSQL data.

E) Create a PostgreSQL Data Visualization into Redash

1. Click Create -> New Query
2. Select the newly created Data Source (you can explore the data structure in the New Query wizard)
3. Write a SQL statement to retrieve the data, for example:
4. Click the "Execute" button to load PostgreSQL data into Redash via CData Connect
5. Use the Visualization Editor to create and analyze graphs from PostgreSQL data
6. You can schedule the query to refresh and update the visualization periodically

Figure: Example of Redash Data Visualization

The End