STANDARD OPERATIONAL PROCEDURE

SUPPORT

Version : 1.1

Author : PT EQUNIX BUSINESS SOLUTIONS

No. : 20223776/SOP/IX/EQUNIX

Date : 21 June 2019

Document Modification

Version	Date	Comments / Changes	Name
1.0	21-06-2019	Initial Version	Iyan Iskandar
1.1	02-05-2023	Update content	Iyan Iskandar

Table of Contents

1. 11DB/Postgres Implementation

1.1. Preparation

1. Server should qualified with suggestion specification
2. Please find below table suggestion specifications

Hardware	Suggestion
CPU	Minimal CPU Clock is 2.5 GHz
Memory	Scala 1:8 with number of CPU cores
Storage	1. Transaction Log should be used fastest storage such as NVMe 2. Data Storage can be use SSD or SATA 3. Backup Storage can be use local HDD for archive WAL and for snapshot can be use any Storage which minimal 100MB/s

3. Volume mount point should follow Equnix standard, below are the table detail:

Purpose	Mount point	Storage Type	File System	Notes
Transaction Log	/equnix/wal	NVMe	EXT2	
Data Directory	/equnix/data	SSD or HDD	EXT4	
Archive WAL	/equnix/backup/archivewal	HDD	EXT4	
Snapshot	/equnix/backup/snapshot	HDD	EXT4	
Tablespace	/equnix/tblspc	NVME or SSD	EXT4	*optional
Indexspace	/equnix/index	NVME or SSD	EXT4	*optional

4. Write cache should be disabled
5. Transaction log partitions can use EXT3 or later but the journaling feature should be disabled.
6. User pgsql should be created and will be used as Database Owner

1.2. Binary Installation

1. 11DB/Postgres can be obtained from Equnix Repository, please coordinate with superordinate.
2. 11DB/Postgres are distributed in a protected compression such as zip with password.
3. 11DB/Postgres can be distributed via USB, SFTP, or MAIL
4. Binary should be located on /equnix/apps/11db
5. Data directory is located on /equnix/data and Transaction Log are symlinked from /equnix/data/pg_wal to /equnix/wal, below is the command:

```
/equnix/apps/11db/bin/initdb -D /equnix/data -X /equnix/wal
```

6. Start scripts are located on /etc/init.d/11db.sh
7. File configurations are located on /equnix/data/conf.d which are grouped by related parameters. Below is table detail:

File name	Parameter	Value	Notes
01_eqnx_connections.conf	max_connections	600	Memory related
	listen_addresses	*	HA pair only
	port	5758	
	password_encryption	scram-sha-256	
02_eqnx_resources.conf	shared_buffers	25GB	Memory related
	temp_buffers	32MB	Memory related
	work_mem	32MB	Memory related
	maintenance_work_mem	128MB	
	effective_cache_size	90GB	Memory related
	full_page_writes	off	
	synchronous_commit	off	

	bgwriter_delay	300	
	wal_writer_delay	300	
	commit_delay	300	
	min_wal_size	80MB	
	max_wal_size	16GB	
	shared_preload_libraries	'pg_stat_statements'	
	pg_stat_statements.max	10000	
	pg_stat_statements.track	all	
	track_activity_query_size	2048	
03_eqnx_replications.conf	wal_level	replica	
	max_wal_senders	20	
	wal_keep_segments	2048	
	hot_standby	on	
	wal_sender_timeout	60min	
	checkpoint_completion_target	0.9	
	max_replication_slots	10	
	max_parallel_workers	16	
	primary_conninfo	'user=equreps host=10.1.86.191 port=5758'	only activated for replica
	recovery_target_timeline	latest	
	promote_trigger_file	'/equnix/data/activate_standby'	
04_eqnx_logs.conf	log_destination	stderr	
	logging_collector	on	
	log_directory	'log'	
	log_filename	'11DB-%Y-%m-%d_%H%M%S.log	
	log_min_duration_statement	5000	

	log_line_prefix	'|%ml%rl%al%dl%ul%pl%el'	
	log_statement	'none'	
	log_connections	off	
	log_disconnections	off	
05_eqnx_vacuums.conf	autovacuum = on		
	log_autovacuum_min_duration	2000	
	autovacuum_max_workers	8	
	autovacuum_vacuum_threshold	500	
	autovacuum_analyze_threshold	500	
06_eqnx_archives.conf	archive_mode	on	
	archive_command	'test ! -f /equnix/backup/archivewal/%f && cp %p /equnix/backup/archivewal/%f	

8. x

1.3 Hardening

System Stability Insurance

Best practices for maintaining system stability are as follows:

1. Utilizing **pgAgent**, administrators should create a job to kill any idle connection for more than desired time frame. This action is intended to restore back resources for other important usage.
2. Enabling **statement_timeout** parameter to eliminate any hanging or resource hungry processes.
3. Limiting connection per database basis, based on hardware resources.
4. When necessary, enable database pool between application and database. This database pool will prevent overloaded resources from requesting applications, when the application is not an open-close connection type.

Security breach mitigation

Best practices for security breach mitigation are as follows:

1. Strengthen Role Based Access Control implementation.

   PostgreSQL has a built-in RBAC system for managing access not only to the database system. Best practices for RBAC are as follows:

   1. A database USER (ROLE with CONNECT) should be assigned to each type of application that is logging in to the database. Privilege separation by login role must be applied.
   2. Database objects (database, schemas, tables, etc.) should be owned by a ROLE that does not commonly modify the data in those tables. The application and ad-hoc users that are writing INSERT/UPDATE/DELETE/SELECT statements should not own them. This prevents accidental changes to the data model and structures.
   3. Access roles should be created based on application access patterns to provide access to individual database objects. Assigning access directly to logon roles should be avoided.
   4. Assign logon roles to the access roles in order to provide them access to a set or group of database objects.
   5. Avoid storing credentials on applications.

2. Enabling 2FA using HBA and Credentials.

   The PostgreSQL Host Based Access control system (pg_hba) provides a method to perform pre-authentication reject/accept of requests based on a set of firewall-like rules. This allows for protection of the database server above and beyond RBAC and any authentication methods. Best practices for HBA are as follows:

   1. Remove all “trust” lines from pg_hba.conf
   2. Add per-host lines for each application host in use, use per-subnet rules when possible. 3. Allow only necessary users to have network access from necessary network segments.
   3. Allow users and network segments to connect only to databases that they need access to. Avoid using “all” on the host field.

4. Use "md5" as an authentication method.
5. Add a default "all all 0.0.0.0/0 REJECT" rule to the bottom of the file.

3. Always use password authentication.

   The default method for authenticating to a PostgreSQL database is the internal password authentication system. This system stores only hashes of the role password in the database and validates that the hash provided by the inbound authentication request matches the hash as stored locally. Best practices for Password Authentication are as follows:

   1. Using *passwordcheck* module, administrators/superuser should enforce a minimum number of characters to be used for the password.
   2. Never use UNENCRYPTED options when creating ROLE, so PostgreSQL will always store encrypted passwords on the storage.
   3. When creating a ROLE, provide a timestamp using the VALID UNTIL option for password validity age.
   4. When a ROLE is not used anymore, DROP the object so there is no unauthorized access. In case the ROLE still wants to be used in the future, superuser/administrator should ALTER the ROLE with NOLOGIN option.
   5. By utilizing **pgAgent** features, administrator should create a job to periodically check whether a ROLE should be forced to change the password, since it exceeds the validity age. This job also should ALTER corresponding ROLE with NOLOGIN option, so setting pg_hba.conf to "trust" will not affect the validity checking.

4. Avoid SQL injection by enabling SQL/Protect.

   SQL/Protect is a module that allows a database administrator to protect Postgres databases from a variety of SQL injection attacks. SQL/Protect provides a DBA-managed layer of security in addition to normal database security policies by screening incoming queries for common SQL injection profiles. In addition, SQL/Protect can also be taught to accept learned 'friendly' queries and reject unfamiliar data request patterns.

5. Database is placed behind a firewall, not accessible from the public.

DMZ containing the database system must be secured with a strong perimeter to prevent access from the public. Enabling SSL if necessary. PostgreSQL has native support for using SSL connection to encrypt client/server communications for increased security. With SSL support compiled in, PostgreSQL will listen for both normal and SSL connections on the same TCP port, and will negotiate with any connecting client on whether to use SSL.

6. Password encryption should be use scram-sha-256
7. Host based authentication should be use scram-sha-256 for every connection
8. Local socket connection should be rejected
9. Only registered IP can be accessed the Database
10. Parameter work_mem and max_connections should be calculated with available memory
11. Database in DMZ location
12. Auto terminate non-application query which took longer than usual, for example 1 hours
13. Limit connection per database

2. 11DB/Postgres Replication

To configure Master to Standby replication follow below steps:

2.1. Master Server

1. Master server should be installed as well as SOP - 11DB/Postgres Implementation
2. The IP Standby server should be registered on pg_hba as a replication connection and has been applied by reloading the database.

```
host  replication   all       <IP Standby>/32        scram-sha-256
```

3. All parameters replication are configured on files 03_eqnx_replications.conf

2.2 Standby Server

1. Server specification is suggested to be similar with Master server especially for High Availability, such as CPU, Memory and Storage.
2. Create Standby instance by using 11db_basebackup

```
/equnix/apps/11db/bin/11db_basebackup -D /equnix/data --waldir=/equnix/wal
```

3. Ensure the parameter primary_conninfo is pointing to the Master with correct credentials
4. Copy start script from the Master server
5. Create standby.signal file on the data directory standby
6. Start standby instance

3. Multi-tiered Backup Implementation

1. Partition backup should be ready on the server Master (/equnix/backup/archivewal and /equnix/backup/snapshot)
2. Snapshot volume should have 3 time of data directory size and Archivewal volume should have 21 time WAL size per day
3. There are 2 scripts that will be configured such as equ_archive_clean.sh and equ_weekly_backup.sh
4. Locate the backup scripts on /equnix/scripts/backup/equ_weekly_backup,sh and /equnix/scripts/hosuekeep/equ_archive_clean.sh
5. The script equ_weekly_backup registered on the cron and should be run weekly at 00:00 every sunday, while the equ_archive_clean.sh should be run daily at 00:00.

4. High Availability Implementation

4.1. Preparation

1. Database already configured with the replication
2. The package pcsd, corosync, pacemaker should be installed on the both Nodes
3. HA scripts deployed on both Nodes and symlinked on /etc/init.d/
4. There should be 2 connections between Master and Standby and 1 connection should be used direct cable connection.
5. user pgsql should be registered for several command without password such as:

```
pgsql ALL = NOPASSWD:
/equnix/scripts/ha/equ_exec_failover.sh,
/usr/bin/docker,
/usr/bin/less,
/usr/sbin/ip,
/usr/bin/kill,
/usr/sbin/crm_mon,
/usr/sbin/pcs,
/usr/bin/netstat
```

6. Client side should prepared the points below:
    a. 3 IP Public should be prepared and and 2 IP Private
    b. UDP connections should be opened for port 5404 and 5405
    c. TCP connections should be opened for port 2224
7. SMTP on each server should be configured to ensure the High Availability alert can send the email.

4.2. Configuration

1. Both IP Private should be registered as a Hostname for each server on both Nodes. , for example:

```
192.168.99.41 EQNX_NODE1
192.168.99.42 EQNX_NODE2
```

2. Start pcsd service

```
$ systemctl start pcsd
```

3. Authenticate Pacemaker Nodes using hacluster user, which has been set with specified password

```
$ pcs cluster auth EQNX_NODE1 EQNX_NODE2
```

versi 0.10

```
$ pcs host auth EQNX_NODE1 EQNX_NODE2
```

4. Create Pacemaker Cluster by specifying Cluster Name and Member Nodes

```
$ pcs cluster setup --name EQNX_GROUP EQNX_NODE1 EQNX_NODE2
```

versi 0.10

```
$ pcs cluster setup EQNX_GROUP EQNX_NODE1 EQNX_NODE2
```

5. Start the cluster

```
$ pcs cluster start
```

6. Set token timeout and update the config (sudo vim /etc/corosync/corosync.conf)

```
totem {
    ......
    ......
    ......
    ......
    token: 5000     <---- change totem timeout to 5s.
}
```

7. Synchronize the cluster config

```
$ pcs cluster sync
```

8. Reload the cluster

```
$ pcs cluster reload corosync
```

9. Disable stonith on Pacemaker

```
$ pcs property set stonith-enabled=false
```

10. Set quorum policy to ignore on Pacemaker

```
$ pcs property set no-quorum-policy=ignore
```

11. Register resources to be executed by Pacemaker.

```
$ pcs resource create equ_forced_failover lsb:equ_forced_failover.sh --group EQNX_GROUP
$ pcs resource create public_vip ocf:heartbeat:IPaddr2 ip=172.18.3.153 --group EQNX_GROUP
$ pcs resource create equ_db_check lsb:equ_db_check.sh op monitor interval=5s --group
EQNX_GROUP
$ pcs resource create pingpub ocf:pacemaker:ping host_list=[PUBLIC NETWORK GATEWAY]
```

```
attempts=3 dampen=5s multiplier=1000 op monitor interval="1s" --clone
v10.0
$ pcs resource create pingpub ocf:pacemaker:ping host_list=[PUBLIC NETWORK GATEWAY]
attempts=3 dampen=5s multiplier=1000 op monitor interval="1s" clone
$ pcs constraint location public_vip rule score=-INFINITY pingd lt 1 or not_defined pingd
```

12. Set resource stickiness to prevent auto failback.

```
$ pcs resource defaults resource-stickiness=100
```

v10.0

```
$ pcs resource defaults update resource-stickiness=100
```

13. Check High Availability Status

```
$ sudo pcs status
```

4.3. Script Configuration

To implement Equnix HA configuration, there are some IP configuration need to be adjusted in given scripts, based on the deployed environment:

1. /equnix/scripts/ha/equ_forced_failover.sh
   a. On Node 1:

```
PUBLIC_PEER=<Public IP of Node 1> (e.g. 10.201.19.222)
PRIVATE_PEER=<Private IP of Node 1> (e.g 192.168.99.42)
PUBLIC_VIP=<Public Virtual IP/Network Segment> (e.g. 10.201.17.73/21)
```

   b. On Node 2:

```
PUBLIC_PEER=<Public IP of Node 2> (e.g. 10.201.19.221)
PRIVATE_PEER=<Private IP of Node 2> (e.g 192.168.99.41)
PUBLIC_VIP=<Public Virtual IP/Network Segment> (e.g. 10.201.17.73/21)
```

2. /equnix/scripts/ha/equ_db_check.sh
   a. On Node 1:

```
EQNX_NODE=<Private IP of Node 2> (e.g. 192.168.99.42)
REPLICA_NODE=<Private IP of Node 3 (Replica Node)> (e.g. 192.168.99.43)
REPLICA_HOSTN=<Hostname of Node 3 (Replica Node)> (e.g. node2.domain.id)
PGPORT=<PostgreSQL Listening Port on Node 1> (e.g. 5432)
```

b. On Node 2:

```
EQNX_NODE=<Private IP of Node 1> (e.g. 192.168.99.41)
REPLICA_NODE=<Private IP of Node 3 (Replica Node)> (e.g. 192.168.99.43)
REPLICA_HOSTN=<Hostname of Node 3 (Replica Node)> (e.g. node1.domain.id)
PGPORT=<PostgreSQL Listening Port on Node 2> (e.g. 5432)
```

3. /equnix/scripts/ha/equ_activate_standby.sh

```
EQNX_NODE=<Private IP of Node 1> (e.g. 192.168.99.41)
PROMOTEFILE=<Location of promote file> (e.g. /equnix/data/promote_standby)
```

4. /equnix/scripts/backup/equ_dbresync.sh

a. On Node 1:

```
FLOATINGIP=<Private Virtual IP> (e.g. 192.168.99.44)
PGFOLLOWHOST=<Private IP of Node 2> (e.g. 192.168.99.42)
PGFOLLOWPORT=<PostgreSQL Listening Port on Node 2> (e.g. 5432)
```

b. On Node 2:

```
FLOATINGIP=<Private Virtual IP> (e.g. 192.168.99.44)
PGFOLLOWHOST=<Private IP of Node 1> (e.g. 192.168.99.41)
PGFOLLOWPORT=<PostgreSQL Listening Port on Node 1> (e.g. 5432)
```

5. Monitoring Implementation

5.1. Components

Dashboard Monitoring builded from 3 component:

1. 11db_monserv, as the Dashboard Service which will show the data in visualization.
2. 11db_mondb, as the Database which will store the data from the agent which will be used by monserv.
3. 11db_agent, as Collector data from each server and send it into mondb

5.2. Preparation

1. Infrastructure team should provide the server/VM to implement the Monitoring, with minimal specification: (a) CPU: 2 Core (4HT); (b) Memory: 8GB; (c). Storage: 100GB
2. Open port 8086 between Database servers and Monitoring server and port 5759 from local access.

5.3. Configuration

1. Put the binary of 11db_monserv and 11db_mondb on the Monitoring Server, meanwhile 11db_agent is installed on the Database Server. All binary files are placed on directory /equnix/apps/monitoring.
2. Run the 11db_mondb service with the following command:
   /equnix/apps/monitoring/11db_mondb/11db_mondb.sh
3. Login to influxdb:

```
/equnix/apps/monitoring/11db_mondb/usr/bin/influx
```

4. Create a user for 11db_agent:

```
CREATE USER agent_user WITH PASSWORD 'p@55w0Rd' WITH ALL PRIVILEGES;
```

5. Create a database for 11db_agent:

```
CREATE DATABASE mondb;
```

6. Create a retention policy for 11db_agent:

Create Retention Policy:

```
CREATE RETENTION POLICY "[retention_name]" ON "[database_name]" DURATION 52w
REPLICATION 1 Default;
```

Show result of Retention Policy::

```
SHOW RETENTION POLICIES ON "[database_name]";
```

7. Run the 11db_monserv service:

   /equnix/apps/monitoring/11db_monserv/11db_monserv.sh

8. Configure the 11db_agent on each Database server, which following configuration:

```
[agent]
  hostname = "equnode1" #nama host
[[outputs.influxdb]] #setting influx
  database = "monitoring"                 #database on 11db_mondb
  urls = [ "http://192.168.8.163:8086" ]  #ip of 11d_mondb
  username = "agent_user"                 #username on 11db_mondb
  password = "p@55w0Rd"                   #password on 11db_mondb

[[inputs.postgresql]]
  address = "host=127.0.0.1 port=5758 user=equreps database=internal password=equnix"

#sesuaikan dengan ip, port, user, database serta password dari postgresql
#setting untuk master side

[[inputs.postgresql]]
   address = "host=127.0.0.1 port=5758 dbname=internal password=equnix user=equreps"
databases = ["internal"]

[[inputs.postgresql_extensible]]
  address = "host=127.0.0.1  port=5758 dbname=internal password=equnix user=equreps"
databases = ["internal"]

[[inputs.postgresql_extensible.query]]
sqlquery="select count(1) as amount, state from pg_stat_activity where usename != 'equreps'
group by state" version=901 withdbname=false tagvalue="state" measurement="pg_connections"

#setting untuk Replika
[[inputs.postgresql_extensible]] address = "host=127.0.0.1 port=5758 dbname=internal
password=equnix user=equreps" databases = ["internal"]
```

```
[[inputs.postgresql_extensible.query]] sqlquery="select case when pg_is_in_recovery() is
true then 1 else 0 end as is_in_recovery;" version=901 withdbname=false tagvalue=""
measurement="pg_is_in_recovery"

[[inputs.postgresql_extensible.query]] sqlquery="SELECT CASE WHEN
pg_last_xlog_receive_location() = pg_last_xlog_replay_location() THEN 0 ELSE EXTRACT (EPOCH
FROM now() - pg_last_xact_replay_timestamp()) END AS log_delay;" version=901
withdbname=false tagvalue="" measurement="pg_log_delay"
```

9. Binary dari 11db_agent dipasang di server yang akan dimonitoring, untuk binary tersebut dipasang pada /equnix/apps/monitoring/;
   The binary of 11db_agent is installed on the server that is going to be monitored. The binary is installed on /equnix/apps/monitoring/;
10. Sesuaikan variable untuk 11db_agent.conf pada /equnix/apps/monitoring/11db_agent/etc/11db_agent/11db_agent.conf
    Adjust the variable for 11db_agent.conf on /equnix/apps/monitoring/11db_agent/etc/11db_agent/11db_agent.conf

```
[agent]
 hostname = "equnode1" #nama host

[[outputs.influxdb]] #setting influx
 database = "monitoring" #database 11db_mondb
 urls = [ "http://192.168.8.163:8086" ] #ip 11d_mondb
 username = "equmon" #username 11db_mondb
 password = "equnix" #password 11db_mondb

[[inputs.postgresql]]
 address = "host=127.0.0.1 port=5758 user=equreps database=internal password=equnix"

#sesuaikan dengan ip, port, user, database serta password dari postgresql
#setting untuk master side
```

```
[[inputs.postgresql]]
  address = "host=127.0.0.1 port=5758 dbname=internal password=equnix user=equreps" databases = ["internal"]

[[inputs.postgresql_extensible]]
  address = "host=127.0.0.1  port=5758 dbname=internal password=equnix user=equreps" databases = ["internal"]

[[inputs.postgresql_extensible.query]]
sqlquery="select count(1) as amount, state from pg_stat_activity where usename != 'equreps' group by state" version=901 withdbname=false tagvalue="state" measurement="pg_connections"

#setting untuk Replika
[[inputs.postgresql_extensible]] address = "host=127.0.0.1 port=5758 dbname=internal password=equnix user=equreps" databases = ["internal"]

[[inputs.postgresql_extensible.query]] sqlquery="select case when pg_is_in_recovery() is true then 1 else 0 end as is_in_recovery;" version=901 withdbname=false tagvalue="" measurement="pg_is_in_recovery"

[[inputs.postgresql_extensible.query]] sqlquery="SELECT CASE WHEN pg_last_xlog_receive_location() = pg_last_xlog_replay_location() THEN 0 ELSE EXTRACT (EPOCH FROM now() - pg_last_xact_replay_timestamp()) END AS log_delay;" version=901 withdbname=false tagvalue="" measurement="pg_log_delay"
```

11. masukan ip server untuk monitoring pada postgresql pg_hba.conf di semua server yang akan di monitoring baik master maupun standby;

    Insert IP Server for monitoring in pg_hba.conf on every server that is going to be monitored, be it the master as well as the standby;

12. untuk menjalankan 11db_agent dapat menggunakan start script 11db_agent.sh pada cd .. adapun untuk start script dibuat/disesuaikan dengan path dari binary 11db_agent/usr/bin/11db_agent dan dengan config file pada 11db_agent/etc/11db_agent/11db_agent.conf sebagai berikut
    To start 11db_agent, the start script 11db_agent.sh on /equnix/apps/monitoring/11db_agent/11db_agent.sh can be run. The start script is created/adjusted to the path of the binary 11db_agent/usr/bin/11db_agent and also adjusted to the configfile on 11db_agent/etc/11db_agent/11db_agent.conf as follows:

    ```
    cd /equnix/apps/monitoring/11db_agent || exit 0;
    ./usr/bin/11db_agent --config /equnix/apps/monitoring/11db_agent/etc/11db_agent/11db_agent.conf
    2>> logs/11db-agent.log >> logs/11db-agent.log &
    ```

13. Untuk mengatur tampilan dashboard kita dapat langsung mengakses ip dari server monitoring dengan port 5759 melalui browser;
    To set up the display of the dashboard, we can directly access the IP from monitoring server with port 5759 via a browser;
14. Untuk login pertama username dan password dari dashboard adalah username : admin password : admin, dan akan diminta untuk memperbaharui password tersebut dengan password baru;
    For the first login, the username and password from the dashboard is username : admin password : admin. After logged in successfully, we will be asked to update the password with the new one;
15. Setelah masuk pada halaman monitoring server kita perlu menambah data source dari 11db_mondb/influx dan Database, untuk menambahkan data source dapat memilih menu configuration lalu pilih Data Source, lalu klik tombol Add data source;
    After being directed to the monitoring server page, we need to add the data source from 11db_mondb/influx and Database. After adding the data source, we can choose the menu configuration, select Data Source, and then click on the Add data source button;
16. Setelah mengklik tombol Add data source lalu pilih time series databases InfluxDB lalu masukan URL 11db_mondb serta sesuaikan untuk nama Database, User dan Password, jika sudah klik tombol Save & Test, jika koneksi berhasil maka akan ada notifikasi Data source is working;
    After clicking the Add data source button, then select time series databases InfluxDB. After that, insert the 11db_mondb URL and adjust the Database name, User, and Password. If everything is set, click on Save & Test button. If the connection is successful then there will be a notification that the Data source is working;
17. Sama seperti sebelumnya untuk data source dari Database dapat dilakukan dengan hal yang sama dengan cara klik Add data source lalu pilih SQL PostgreSQL lalu sesuaikan untuk Host, Database, User, Password serta

PostgreSQL Version, setelah selesai klik tombol Save & Test, jika koneksi berhasil maka akan muncul notifikasi Database Connection OK;

It goes the same for the Data Source from Database. We can add the data source by clicking on the Add data source button and then select SQL PostgreSQL. After that, make an adjustment for the Host, database, User, Password and PostgreSQL Version. After everything is set, click on Save & Test button. If the connection is successful, there will be notification Database Connection OK;

18. untuk membuat Dashboard kita dapat mengimpor data dashboard dengan format json pada menu Create lalu klik Import, setelah itu klik Import Upload Json File lalu sesuaikan Nama, Folder Serta UID jika diperlukan, setelah selesai lalu klik import, dan cek kembali datasource jika grafik data tidak muncul.

    To create a Dashboard, we can import dashboard data with JSON format on the Create menu and then click Import. After that, click Import Upload Json File and adjust the Name, Folder, and UID if needed. After everything is set, click Import and then check the datasource again if the graph is not showing.

5.3. Dashboard Setup

1. Load json on the dashboard
2. Change the datasource to influx
3.

6. Job and Alert Implementation

1. Alert are monitoring the Database and Server resource which run by schedule
2. Connection alert
3. Locking alert
4. Slow query alert
5. Archive alert
6. Replication alert
7. Archive fail alert

7. DRC Implementation

1. DRC server location should be far at least 75 KM from current data center
2. Replication wich used is asynchronous replication
3. Setup replication can follow the SOP 2 Replication