Agile Automation Technology, LLC

Deploying a Self-Hosted PostgreSQL Timescale DB on Debian-Based OS (Ubuntu 22.04 LTS) as a Storage System for Ignition Tag Historian.

Rev.1.1 2024/02/12

Preface

The demand to write this document become a self-evident after several practical instances of deployed data base historians for Ignition based production systems.These systems required a large set of telemetry tags to be stored and managed over a vast time period. The first choice was the MS SQL DB engine, Express or Standard editions. While the Express edition imposes restrictions on CPU usage, db size and lack of built-in scheduler, the Standard edition requires paid license and maintenance plan to be developed and implemented. The alternative options were; the most popular MySql & PostgreSQL with Timescale extension. Both are open source and available for commercial use with free license. The PostgreSQL with Timescale option was chosen. The document provides a complete set of instructions and steps in order to deploy PostgreSQL DB with Timescale extension as a storage system for Ignition Tag Historian Module. The document does not provide instructions for installing the OS and Ignition software. It is expected the reader knows how to install the OS and Ignition software, and to configure communication driver with automation controller. It is also expected that the reader knows the foundation of networking technologies and protocols.

Introduction

Why PostgreSQL with Timescale? The primary objective to address the “Maintenance” issue. This issue is notorious with MS SQL DB. Log file for MS SQL DB (do not confuse with the data file) will swell over time, and may consume the entire disk space if maintenance plan is not deployed. The secondary objective to address performance degradation. While MS SQL DB engine offers an excellent performance in the beginning, later, as the amount of stored historical data grow, the performance issue begin to emerge. The issue becomes even more prominent if there no maintenance plan deployed. The “later” term in this case is vaguely defined as 1+ year worth of history for a 5000+ tags. How PostgreSQL with Timescale addressing the above issue? The “Maintenance” issue addressed by enabling 2 maintenance jobs; one for data retention and the other for data compression. Note, that these functions are only available for HyperTables, and this is why the Timescale extension for PostgreSQL is imperative. (Timescale extension enables the HyperTable functionality in PostgreSQL DB engine). The “Performance” issue addressed while using overall more efficient method of storing data (saves disk space) and utilization of BRIN(Block Range INdex). The benefit of it (BRIN) becomes prominent in data retrieval when the storage table grows in size (tens or hundreds of GB or TB).

Environment used for the case described depicted in (**Fig.1**). Both hosts are virtual machines.

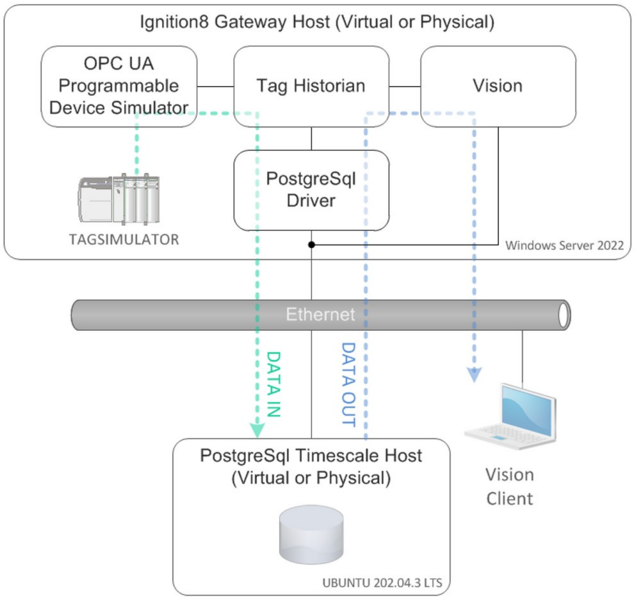

Ignition8 Gateway Host (Virtual or Physical)
OPC UA Programmable Device Simulator
Tag Historian
Vision
PostgreSql Driver
TAGSIMULATOR
Windows Server 2022
Ethernet
DATA IN
DATA OUT
PostgreSql Timescale Host (Virtual or Physical)
UBUNTU 202.04.3 LTS
Vision Client

Fig.1

Prerequisites

- Intermediate knowledge of Ignition 8 application.
- General knowledge of operating systems such as Microsoft Windows and UBUNTU.
- General knowledge of network technology.
- Basic-to-Intermediate knowledge of data base technology and SQL.

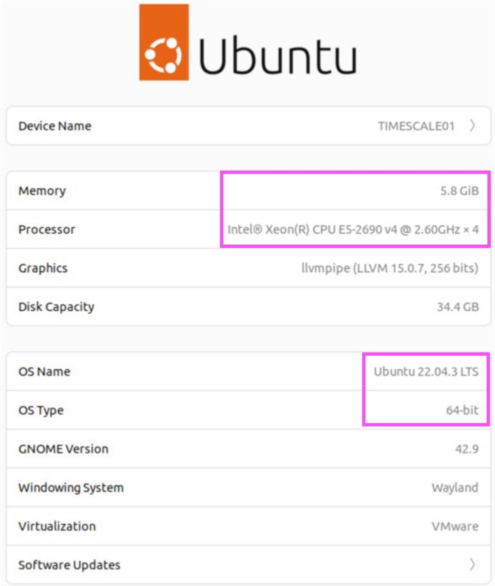
Credits

- This manual is partly based on https://docs.timescale.com/self-hosted/latest/install/installation-linux/ by the Timescale. Modifications and additional sections were introduced to address topics specific for integration with Ignition Tag Historian module.
- Virtual environment constructed with free version of ESXi 7.0U3g hypervisor provided by the VMWare.
- PostgreSQL 14.10 license free version provided by the PostgreSQL.
- PgAdmin4 version 8.2 Desktop free license provided by the pgAdmin Development Team.
- Ignition 8.1.37 LTS time limited license provided by the Inductive Automation, LLC.

Instructions (begins with UBUNTU OS host)

It is expected that 2 virtual or physical hosts already deployed, and Ignition software installed on one of them. The host with UBUNTU OS (scheduled for PostgreSQL installation) must have access to the Internet.

- Checking the UBUNTU OS version   Click the "Settings" > "About" (**Fig.2**).



Device Name	TIMESCALE01 〉
Memory	5.8 GiB
Processor	Intel® Xeon(R) CPU E5-2690 v4 @ 2.60GHz × 4
Graphics	llvmpipe (LLVM 15.0.7, 256 bits)
Disk Capacity	34.4 GB
OS Name	Ubuntu 22.04.3 LTS
OS Type	64-bit
GNOME Version	42.9
Windowing System	Wayland
Virtualization	VMware
Software Updates	〉

Fig.2

- Installing the PostgreSQL application in Linux Terminal

```
sysadmin@TIMESCALE01:~$ sudo apt install gnupg postgresql-common apt-transport-https lsb-release wget
[sudo] password for sysadmin: ********
Reading package lists... Done
Building dependency tree... Done
Reading state information...
...
After this operation, 1,608 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.11 [1,510 B]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libcommon-sense-perl amd64 3.75-2build1 [21.1 kB]
...
Removing obsolete dictionary files:
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql.service → /lib/systemd/system/postgresql.service.
Processing triggers for man-db (2.10.2-1) ...
sysadmin@TIMESCALE01:~$ sudo /usr/share/postgresql-common/pgdg/apt.postgresql.org.sh
This script will enable the PostgreSQL APT repository on apt.postgresql.org on
your system. The distribution codename used will be jammy-pgdg.

Press Enter to continue, or Ctrl-C to abort.
```

```
Writing /etc/apt/sources.list.d/pgdg.list ...
sysadmin@TIMESCALE01:~$ sudo echo "deb https://packagecloud.io/timescale/timescaledb/ubuntu/ $(lsb_release -c -s)
main" | sudo tee /etc/apt/sources.list.d/timescaledb.list
deb https://packagecloud.io/timescale/timescaledb/ubuntu/ jammy main
sysadmin@TIMESCALE01:~$ wget --quiet -O - https://packagecloud.io/timescale/timescaledb/gpgkey | sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/timescaledb.gpg
sysadmin@TIMESCALE01:~$ sudo apt update
...
sysadmin@TIMESCALE01:~$ sudo apt install timescaledb-2-postgresql-14
eading package lists... Done
Building dependency tree... Done
Reading state information... Done
...
Processing triggers for libc-bin (2.35-0ubuntu3.6) ...
Processing triggers for man-db (2.10.2-1) ...
sysadmin@TIMESCALE01:~$ sudo timescaledb-tune
Using postgresql.conf at this path:
/etc/postgresql/14/main/postgresql.conf
Is this correct? [(y)es/(n)o]: y
Writing backup to:
/tmp/timescaledb_tune.backup202402012318

shared_preload_libraries needs to be updated
Current:
#shared_preload_libraries = ''
Recommended:
shared_preload_libraries = 'timescaledb'
Is this okay? [(y)es/(n)o]: y
success: shared_preload_libraries will be updated
Tune memory/parallelism/WAL and other settings? [(y)es/(n)o]: y
Recommendations based on 5.78 GB of available memory and 4 CPUs for PostgreSQL 14

Memory settings recommendations
Current:
shared_buffers = 128MB
#effective_cache_size = 4GB
#maintenance_work_mem = 64MB
#work_mem = 4MB
Recommended:
shared_buffers = 1479MB
effective_cache_size = 4437MB
maintenance_work_mem = 757333kB
work_mem = 5048kB
Is this okay? [(y)es/(s)kip/(q)uit]: y
success: memory settings will be updated

Parallelism settings recommendations
Current:
missing: timescaledb.max_background_workers
#max_worker_processes = 8
#max_parallel_workers_per_gather = 2
#max_parallel_workers = 8
Recommended:
timescaledb.max_background_workers = 16
max_worker_processes = 23
max_parallel_workers_per_gather = 2
max_parallel_workers = 4
Is this okay? [(y)es/(s)kip/(q)uit]: y
success: parallelism settings will be updated

WAL settings recommendations
Current:
#wal_buffers = -1
min_wal_size = 80MB
Recommended:
wal_buffers = 16MB
min_wal_size = 512MB
Is this okay? [(y)es/(s)kip/(q)uit]: y
success: WAL settings will be updated

Background writer settings recommendations
Current:
Recommended:
Is this okay? [(y)es/(s)kip/(q)uit]: y
success: background writer settings will be updated

Miscellaneous settings recommendations
Current:
#default_statistics_target = 100
#random_page_cost = 4.0
#checkpoint_completion_target = 0.9
max_connections = 100
#max_locks_per_transaction = 64
#autovacuum_max_workers = 3
#autovacuum_naptime = 1min
#effective_io_concurrency = 1
Recommended:
default_statistics_target = 100
random_page_cost = 1.1
checkpoint_completion_target = 0.9
```

```
max_connections = 75
max_locks_per_transaction = 64
autovacuum_max_workers = 10
autovacuum_naptime = 10
effective_io_concurrency = 256
Is this okay? [(y)es/(s)kip/(q)uit]: y
success: miscellaneous settings will be updated
Saving changes to: /etc/postgresql/14/main/postgresql.conf
```

- Installing psql client with the apt package manager

```
sysadmin@TIMESCALE01:~$ sudo apt-get update
...
sysadmin@TIMESCALE01:~$ sudo apt-get install postgresql-client
Reading package lists... Done
Building dependency tree... Done
...
After this operation, 9,083 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://apt.postgresql.org/pub/repos/apt jammy-pgdg/main amd64 postgresql-client-16 amd64 16.1-1.pgdg22.04+1 [1,889 kB]
Get:2 http://apt.postgresql.org/pub/repos/apt jammy-pgdg/main amd64 postgresql-client all 16+256.pgdg22.04+1 [68.9 kB]
Fetched 1,958 kB in 2s (1,296 kB/s)
...
Removing obsolete dictionary files:
sysadmin@TIMESCALE01:~$
```

- Setting up the TimescaleDB extension on UBUNTU system and creating new tsdb database for historian

```
sysadmin@TIMESCALE01:~$ sudo systemctl restart postgresql
sysadmin@TIMESCALE01:~$ systemctl enable postgresql.service
Synchronizing state of postgresql.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable postgresql
sysadmin@TIMESCALE01:~$ sudo -i -u postgres
postgres@TIMESCALE01:~$ psql
psql (16.1 (Ubuntu 16.1-1.pgdg22.04+1), server 14.10 (Ubuntu 14.10-1.pgdg22.04+1))
Type "help" for help.

postgres=# \password postgres
Enter new password for user "postgres": ********
Enter it again: ********
postgres=# \q
postgres@TIMESCALE01:~$ psql
psql (16.1 (Ubuntu 16.1-1.pgdg22.04+1), server 14.10 (Ubuntu 14.10-1.pgdg22.04+1))
Type "help" for help.

postgres=# create user ignition with password '12345';     ← use this password to setup Ignition Database connection
CREATE ROLE
postgres=# create database tsdb;          ← use this database name to setup Ignition Database connection
CREATE DATABASE
postgres=# grant all privileges on database tsdb to ignition;
GRANT
postgres=# \c tsdb
psql (16.1 (Ubuntu 16.1-1.pgdg22.04+1), server 14.10 (Ubuntu 14.10-1.pgdg22.04+1))
You are now connected to database "tsdb" as user "postgres".
tsdb=# CREATE EXTENSION IF NOT EXISTS timescaledb;
WARNING:
WELCOME TO
 _____ _                               _     ____________
|_   _(_)                             | |    |  _  \ ___ \
  | |  _ _ __ ___   ___  ___  ___ __ _| | ___| | | | |_/ /
  | | | |  _ ` _ \ / _ \/ __|/ __/ _` | |/ _ \ | | | ___ \
  | | | | | | | | |  __/\__ \ (_| (_| | |  __/ |/ /| |_/ /
  |_| |_|_| |_| |_|\___||___/\___\__,_|_|\___|___/ \____/
               Running version 2.13.1
For more information on TimescaleDB, please visit the following links:

 1. Getting started: https://docs.timescale.com/timescaledb/latest/getting-started
 2. API reference documentation: https://docs.timescale.com/api/latest

Note: TimescaleDB collects anonymous reports to better understand and assist our users.
For more information and how to disable, please see our docs https://docs.timescale.com/timescaledb/latest/how-to-guides/configuration/
telemetry.

CREATE EXTENSION
tsdb=# \dx
                                                List of installed extensions
    Name     | Version |   Schema   |                                      Description
-------------+---------+------------+-------------------------------------------------------------------------------------
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
 timescaledb | 2.13.1  | public     | Enables scalable inserts and complex queries for time-series data (Community Edition)
(2 rows)

tsdb=# \q
postgres@TIMESCALE01:~$ exit
logout
sysadmin@TIMESCALE01:~$
```

- ## Modify the PostgreSQL **postgresql.conf** configuration file

Open the PostgreSQL configuration file “**postgresql.conf**” using text editor. The file located in the “/etc/postgresql/<version nr>/main directory”. To open the file in Linux Terminal use the following command:

```
sysadmin@TIMESCALE01:~$ sudo nano /etc/postgresql/<version nr>/main/postgresql.conf
```

Find the line **#listen_addresses = 'localhost'** and uncomment it (remove the # character at the beginning of the line). Next, change the value to **listen_addresses = '*'**. This allows PostgreSQL to listen on all available IP addresses. Alternatively, a particular host’s IP address can be specified where the server will listen at.

```
# -----------------------------
# PostgreSQL configuration file
# -----------------------------
#
# This file consists of lines of the form:
#
#   name = value
#
# (The "=" is optional.)  Whitespace may be used.  Comments are introduced with
# "#" anywhere on a line.  The complete list of parameter names and allowed
# values can be found in the PostgreSQL documentation.
...
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -
#Comment this line:
#listen_addresses = 'localhost'                  # what IP address(es) to listen on;
#Add the following:
listen_addresses = '*'
                                                 # comma-separated list of addresses;
                                                 # defaults to 'localhost'; use '*' for all
                                                 # (change requires restart)
port = 5432                                      # (change requires restart)
max_connections = 75                             # (change requires restart)
#superuser_reserved_connections = 3              # (change requires restart)
unix_socket_directories = '/var/run/postgresql'  # comma-separated list of directories
```

Save the “**postgresql.conf**” file.

- ## Modify the PostgreSQL **pg_hba.conf** configuration file

Open the PostgreSQL configuration “**pg_hba.conf**” file using text editor. The file located in the “/etc/postgresql/<version nr>/main” directory. To open the file from the Linux Terminal use following command:

```
sysadmin@TIMESCALE01:~$ sudo nano /etc/postgresql/<version nr>/main/pg_hba.conf
```

Find the following section:

```
# IPv4 local connections:
host     all              all              127.0.0.1/32     md5
```

And modify it to:

```
# IPv4 local connections:
host     all              all              0.0.0.0/0        trust
```

```
# PostgreSQL Client Authentication Configuration File
# ===================================================
#
# Refer to the "Client Authentication" section in the PostgreSQL
# documentation for a complete description of this file.  A short
# synopsis follows.
#
# This file controls: which hosts are allowed to connect, how clients
# are authenticated, which PostgreSQL user names they can use, which
# databases they can access.  Records take one of these forms:
#
# local         DATABASE  USER  METHOD  [OPTIONS]
# host          DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
...
...
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
#Comment this line:
#host    all             all             127.0.0.1/32            scram-sha-256
#Add the following:
host    all             all              0.0.0.0/0              trust
...
```

Save the “**pg_hba.conf**” file.

- ## Allow tcp port 5432 through the firewall

To enable tcp traffic on port 5432 through the firewall, from the Linux Terminal use the following command:

```
sysadmin@TIMESCALE01:~$ sudo ufw allow 5432/tcp
Rules updated
Rules updated (v6)
```

- Restart PostgreSQL service

Run the following command from the Linux Terminal to restart PostgreSQL:

```
sysadmin@TIMESCALE01:~$ sudo service postgresql restart
```

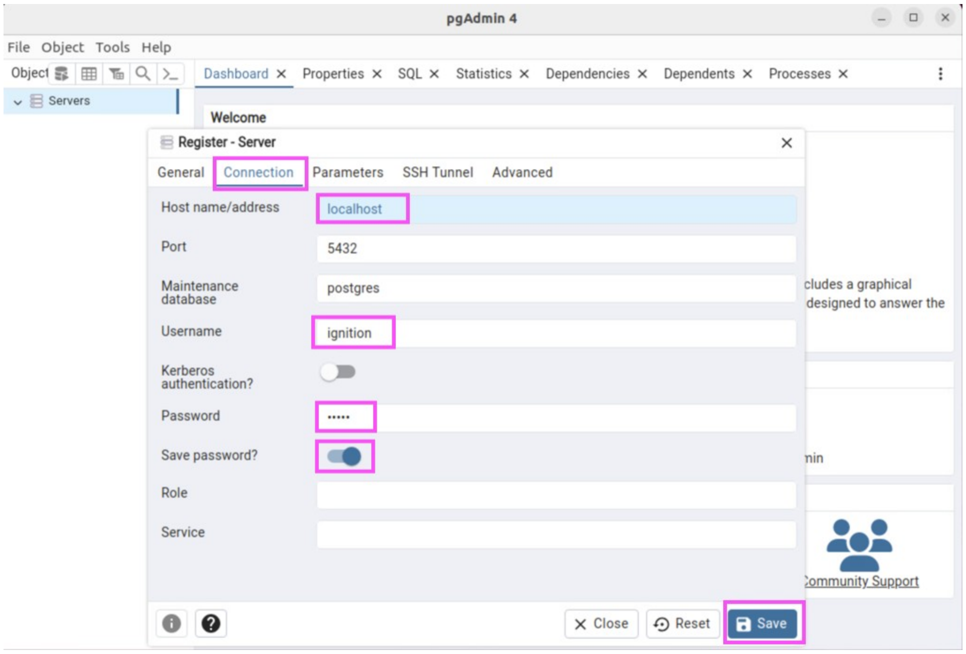
- Installing Pgadmin4 utility

```
sysadmin@TIMESCALE01:~$ sudo snap install curl
curl 8.1.2 from Wouter van Bommel (woutervb) installed
sysadmin@TIMESCALE01:~$ curl -fsS https://www.pgadmin.org/static/packages_pgadmin_org.pub | sudo gpg --dearmor -o 
/usr/share/keyrings/packages-pgadmin-org.gpg
sysadmin@TIMESCALE01:~$ sudo sh -c 'echo "deb [signed-by=/usr/share/keyrings/packages-pgadmin-org.gpg] 
https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/$(lsb_release -cs) pgadmin4 main" > 
/etc/apt/sources.list.d/pgadmin4.list && apt update'
Hit:1 http://apt.postgresql.org/pub/repos/apt jammy-pgdg InRelease
...
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
N: Skipping acquire of configured file 'main/binary-i386/Packages' as repository 'http://apt.postgresql.org/pub/repos/apt jammy-pgdg 
InRelease' doesn't support architecture 'i386'
sysadmin@TIMESCALE01:~$ sudo apt install pgadmin4 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
...
Processing triggers for ufw (0.36.1-4ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
sysadmin@TIMESCALE01:~$ cd /usr/pgadmin4/bin
sysadmin@TIMESCALE01:/usr/pgadmin4/bin$ sudo apt install pgadmin4
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
pgadmin4 is already the newest version (8.2).
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2 libllvm13
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
sysadmin@TIMESCALE01:/usr/pgadmin4/bin$ cd \
sysadmin@TIMESCALE01:~$ exit
```

- Run pgAdmin4 GUI for the first time

On UBUNTU deck click the “Show Application” icon. Locate and click the “pgAdmin4” icon .

In pgAdmin4 GUI right-click the “Servers” root and click the Register>Server item. Enter the “Name” then click the “Connection” tab and enter connection parameters. Click the “Save” button (**Fig.3**).



Host name/address: localhost
Port: 5432
Maintenance database: postgres
Username: ignition
Kerberos authentication? ○
Password: •••••
Save password? ●
Role:
Service:

Fig.3

In “Object Explorer” panel expand the tree to reach the “tsdb” node. Click the “tsdb” node (**Fig.4**).

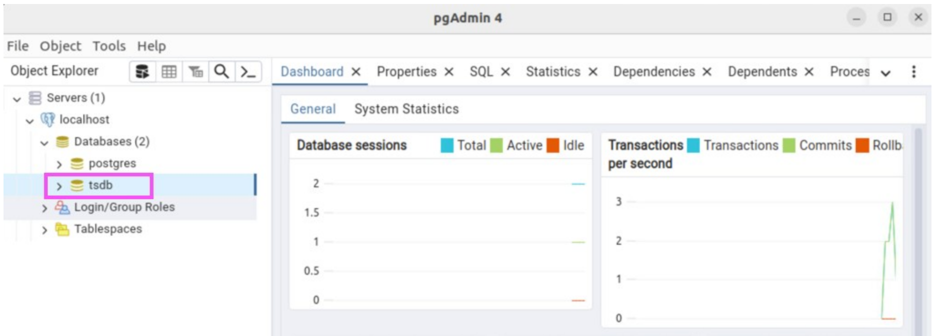


Fig.4

Important! At this point PostgreSQL and pgAdmin4 applications are installed and ready to accommodate Storage System infrastructure for Ignition Tag Historian.

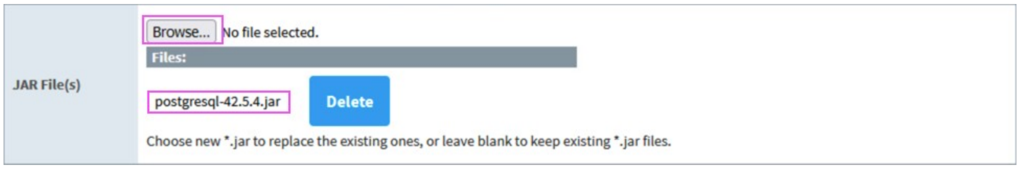
On Ignition 8 Gateway Host:

- ## Configuring the PostgreSQL driver on Ignition Gateway for the first time

Download (latest) JDBC postgresql driver from https://jdbc.postgresql.org/ to Ignition gateway host. Copy driver (for instance: postgresql-42.x.y.jar) to the “./Downloads” folder and update driver file in:
Ignition Gateway > Config > DATABASES > Drivers > PostgreSQL > Edit

Note! By default Ignition is using PostgreSQL driver version 9.X that does NOT support Hyper-table features.

Click the “Choose File” button, select driver file from the “./Downloads” folder and click the “Save Changes” button (**Fig.5**).



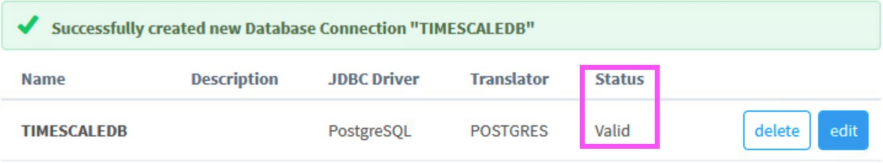
JAR File(s)	Browse... No file selected. Files: postgresql-42.5.4.jar Delete Choose new *.jar to replace the existing ones, or leave blank to keep existing *.jar files.

Fig.5

Verify the driver file copied to the “C:\Program Files\Inductive Automation\Ignition\user-lib\jdbc” folder.
Once driver is updated, configure a new database connection to Postgres tsdb DB using previously defined credentials.
Ignition Gateway > Config > DATABASES > Connections > Create new Database Connection > PostgreSQL > Next

```
        Name            = TIMESCALEDB
        JDBC Driver     = PostgreSQL's
        Connect URL     = jdbc:postgresql://<ip_address>:5432/tsdb
        Username        = ignition
        Password        = 12345  (See: the “Setting up the TimescaleDB extension on UBUNTU system” section)
        The rest of the settings are left as default.
```

Click the “Create New Database Connection” button. Verify created connection status is valid (**Fig.6**)



✓ Successfully created new Database Connection "TIMESCALEDB"

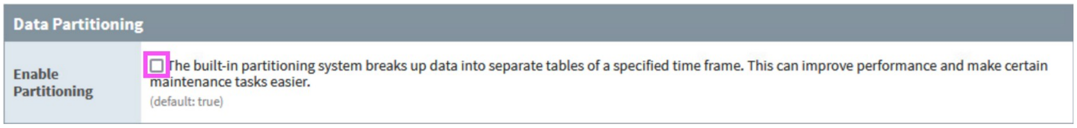
Name	Description	JDBC Driver	Translator	Status	
TIMESCALEDB		PostgreSQL	POSTGRES	Valid	delete edit

Fig.6

- ## Configure tags history provider

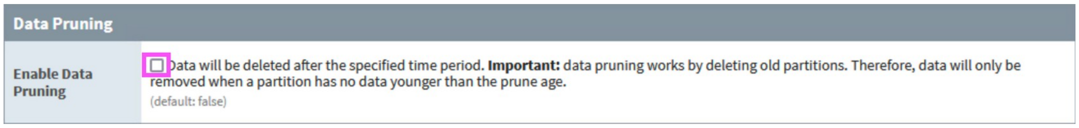
Ignition Gateway > Config > TAGS > History > TIMESCALEDB > edit

Important! The “Data Partitioning” & the “Data Pruning” settings must be disabled! (**Fig.7**, **Fig.8**). Both theses functions are to be carried out by the Timescale Hyper-Table extension features. Click the “Save Changes” button.



Data Partitioning	
Enable Partitioning	☐ The built-in partitioning system breaks up data into separate tables of a specified time frame. This can improve performance and make certain maintenance tasks easier. (default: true)

Fig.7



Data Pruning

Enable Data Pruning	☐ Data will be deleted after the specified time period. **Important:** data pruning works by deleting old partitions. Therefore, data will only be removed when a partition has no data younger than the prune age. (default: false)

Fig.8

- Create Tags using simulation program

Ignition Gateway > Config > Opcua > Devices > Create new Device > Programmable Device Simulator > Next

```
Name = TAGSIMULATOR
Enabled = True
Repeat Program = True
Base Rate (ms) = 1000     - period
```

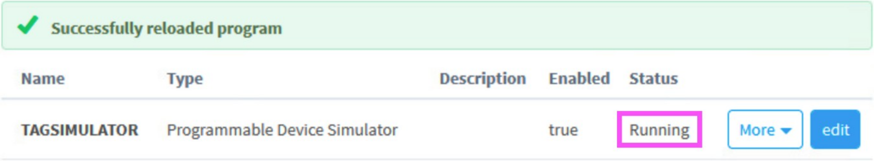
Click the “Create New Device” button.

Create an empty text file with the name “tags_sim_program.csv” and save it to the “./Download” folder.
Populate the file with the following data below:

```
Time Interval, Browse Path, Value Source, Data Type
"0","Ramp/Ramp10","ramp(0.0, 100.0, 10, true)","Float"
"0","Ramp/Ramp20","ramp(0.0, 100.0, 20, true)","Float"
"0","Ramp/Ramp30","ramp(0.0, 100.0, 30, true)","Float"
"0","Ramp/Ramp40","ramp(0.0, 100.0, 40, true)","Float"
"0","Ramp/Ramp50","ramp(0.0, 100.0, 50, true)","Float"
"0","Ramp/Ramp60","ramp(0.0, 100.0, 60, true)","Float"
"0","Sine/Sine10","sine(0.0, 100.0, 10, true)","Float"
"0","Sine/Sine20","sine(0.0, 100.0, 20, true)","Float"
"0","Sine/Sine30","sine(0.0, 100.0, 30, true)","Float"
"0","Sine/Sine40","sine(0.0, 100.0, 40, true)","Float"
"0","Sine/Sine50","sine(0.0, 100.0, 50, true)","Float"
"0","Sine/Sine60","sine(0.0, 100.0, 60, true)","Float"
```

In Ignition Gateway > Config > Opcua > Devices > TAGSIMULATOR click the “More” then “edit program”

Set Load Program = Load from CSV, click the “Browse…” button.
Select the “tags_sim_program.csv” from “./Download” folder and click the “Open”.
Click the “Load Simulator Program”.
Verify the list of simulated tags and parameters displayed.
Click the “Save Program” button. And verify the “TAGSIMULATOR” has status “Running” (**Fig.9**).



✓ Successfully reloaded program

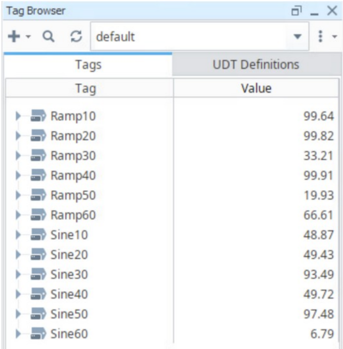
Name	Type	Description	Enabled	Status		
TAGSIMULATOR	Programmable Device Simulator		true	Running	More ▾	edit

Fig.9

- Create new Ignition Project

Open Ignition designer application and create a new “TIMESACLE” project.

Add device tags to the project: In “Tag Browser” panel click the “+” drop-down list, select the “Browse Devices…” . Expand the “Ignition OPC UA Server” to reach the “Ramp” folder. With the “Shift” key select all tags in Ramp folder & add them with the “→” key to default provider window. Perform the same action for all tags in “Sine” folder.
Click the “OK” button. The “Tag Browser” panel must look like this (**Fig.10**):



Tag	Value
Ramp10	99.64
Ramp20	99.82
Ramp30	33.21
Ramp40	99.91
Ramp50	19.93
Ramp60	66.61
Sine10	48.87
Sine20	49.43
Sine30	93.49
Sine40	49.72
Sine50	97.48
Sine60	6.79

Fig.10

- ### Create new Tag Group for Timescale

Create “Timescale” tag group. In Tag Browser” panel click the ⋮ ▾ button, then click the “Edit Group Tags”.

```
Add tag group.   Name = Timescale
                 Mode = Direct
                 Rate(ms) = 1000
                 Data Mode = Subscribed
                 ...
                 Min Time Between Samples = 1 sec
                 Max Time Between Samples = 1 min
```

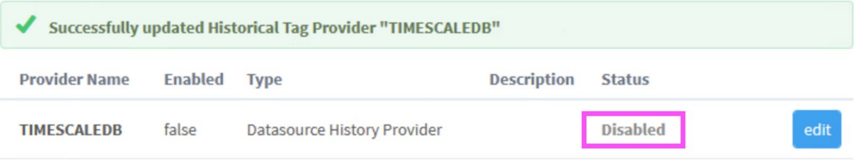
Click the “OK” to save the group.

- ### Initializing Storage System table infrastructure for Ignition Tag Historian

In Tag Browser” panel double-click the “**Sine10**” tag. In “Tag Editor” pop-up window select the “History” property from the “Categories” list. Change the following settings as shown:

```
                 History Enabled = true
                 Storage Provider = TIMESACLEDB
                 Deadband Mode = Percent
                 Historical Deadband = 0.01
                 Sample Mode = Tag Group
                 ...
                 Historical Tag Group = Timescale
```

VERY IMPORTANT! Click the “Apply” button, and within 2-5 sec. set the History Enabled for the “Ramp10” tag back to false. Click the “Apply” button again. At this point there must be absolutely no tags with enabled history provider pointing to the “TIMESCALEDB”.
Disable the history provider “TIMESCALEDB” in Ignition Gateway > Config > Tags > History >TIMESCALEDB (**Fig.11**) to reassure that.



✓ Successfully updated Historical Tag Provider "TIMESCALEDB"

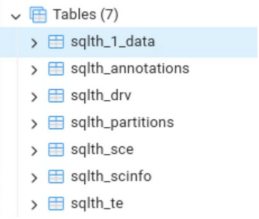
Provider Name	Enabled	Type	Description	Status	
TIMESCALEDB	false	Datasource History Provider		Disabled	edit

Fig.11

On UBUNTU OS PostgreSQL Host:

- ### Switch back to the PostgreSQL Timescale Historian DB Host

Open the “pgAdmin4” GUI, in “Object Explorer” navigate to:
Servers > Localhost > Databases > tsdb > Schemas > public > Tables
There the Storage System for Ignition Tag Historian has crated a set of tables (**Fig.12**)

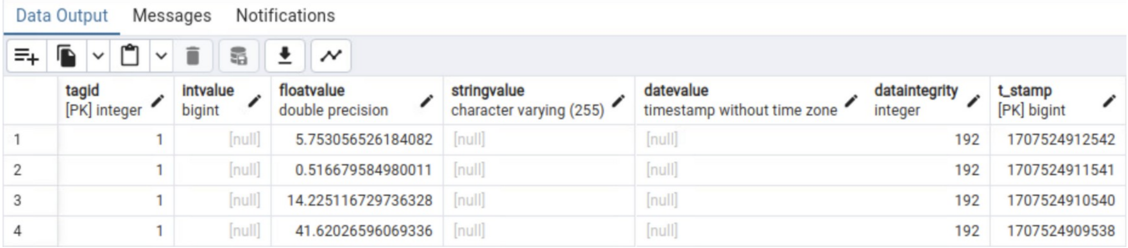


Tables (7)
- sqlth_1_data
- sqlth_annotations
- sqlth_drv
- sqlth_partitions
- sqlth_sce
- sqlth_scinfo
- sqlth_te

Fig.12

On first time historization is enabled for a tag, Ignition creates a table to store historical data with the name “sqlth_1_data”. Subsequent setup from scratch on the same Ignition gateway will create another table named “sqlth_2_data”, Each subsequent setup will increment the table index by 1. Table index value is managed in Ignition gateway. If the name format other that “sqlth_X_data” is observed, it indicates an error in historical provider configuration. Typically the result of the “Data Partitioning” option left enabled.

Right-click the “sqlth_1_data” table name and click the “Query Tool” item. In “Query” window execute the following query: select * from public.sqlth_1_data; Data Output must show several recorded samples of “**Sine10**” tag (**Fig.13**)



	tagid [PK] integer	intvalue bigint	floatvalue double precision	stringvalue character varying (255)	datevalue timestamp without time zone	dataintegrity integer	t_stamp [PK] bigint
1	1	[null]	5.753056526184082	[null]	[null]	192	1707524912542
2	1	[null]	0.516679584980011	[null]	[null]	192	1707524911541
3	1	[null]	14.225116729736328	[null]	[null]	192	1707524910540
4	1	[null]	41.62026596069336	[null]	[null]	192	1707524909538

Fig.13

- Upgrading the “sqlth_1_data” table to a Hyper-Table

```
Open Linux Terminal and execute the following commands:
sysadmin@TIMESCALE01:~$ sudo -i -u postgres
postgres@TIMESCALE01:~$ psql
psql (16.1 (Ubuntu 16.1-1.pgdg22.04+1), server 14.10 (Ubuntu 14.10-1.pgdg22.04+1))
Type "help" for help.

postgres=# \c tsdb
psql (16.1 (Ubuntu 16.1-1.pgdg22.04+1), server 14.10 (Ubuntu 14.10-1.pgdg22.04+1))
You are now connected to database "tsdb" as user "postgres".
tsdb=# select * from sqlth_1_data;
 tagid | intvalue |     floatvalue     | stringvalue | datevalue | dataintegrity |    t_stamp
-------+----------+--------------------+-------------+-----------+---------------+---------------
     1 |          |  5.753056526184082 |             |           |           192 | 1707524912542
     1 |          |  0.516679584980011 |             |           |           192 | 1707524911541
     1 |          | 14.225116729736328 |             |           |           192 | 1707524910540
     1 |          |  41.62026596069336 |             |           |           192 | 1707524909538
(4 rows)

tsdb=# SELECT * FROM create_hypertable('sqlth_1_data','t_stamp', if_not_exists => True, chunk_time_interval => 
86400000, migrate_data => True);
WARNING:  column type "character varying" used for "stringvalue" does not follow best practices
HINT:  Use datatype TEXT instead.
WARNING:  column type "timestamp without time zone" used for "datevalue" does not follow best practices
HINT:  Use datatype TIMESTAMPTZ instead.
NOTICE:  migrating data to chunks
DETAIL:  Migration might take a while depending on the amount of data.
 hypertable_id | schema_name |  table_name  | created
---------------+-------------+--------------+---------
             1 | public      | sqlth_1_data | t
(1 row)
Note! Chunks time interval is set to 86400000 ms OR 24 hrs.

tsdb=# ALTER TABLE sqlth_1_data SET (timescaledb.compress, timescaledb.compress_orderby = 't_stamp DESC', 
timescaledb.compress_segmentby = 'tagid');
ALTER TABLE

tsdb=# CREATE OR REPLACE FUNCTION unix_now() returns BIGINT LANGUAGE SQL STABLE as $$ SELECT (extract(epoch from 
now())*1000)::bigint $$;
CREATE FUNCTION

tsdb=# SELECT set_integer_now_func('sqlth_1_data', 'unix_now');
 set_integer_now_func
----------------------

(1 row)
*********************************************************************************************************************************************
*         set_integer_now_func()
*
*                   This function is only relevant for hypertables with integer (as opposed to TIMESTAMP/TIMESTAMPTZ/DATE)time values.
*                   For such hypertables, it sets a function that returns the now() value (current time) in the units of the time column.
*                   This is necessary for running some policies on integer-based tables. In particular, many policies only apply to chunks
*                   of a certain age and a function that returns the current time is necessary to determine the age of a chunk.
*********************************************************************************************************************************************

tsdb=# SELECT add_compression_policy('sqlth_1_data', BIGINT '2678400000');
 add_compression_policy
------------------------
                   1000      - Note! Returns JOB ID added to the
(1 row)
*********************************************************************************************************************************************
*         add_compression_policy()
*
*                   Allows to set a policy by which the system compresses a chunk automatically in the background after it reaches a given
*                   age. Compression policies can only be created on hypertables or continuous aggregates that already have compression
*                   enabled. To set timescaledb.compress and other configuration parameters for hypertables, use the ALTER TABLE command.
*                   To enable compression on continuous aggregates, use the ALTER MATERIALIZED VIEW command. To view the policies that you
*                   set or the policies that already exist, see informational views. Important! For hypertables with integer-based
*                   timestamps: the time interval should be an integer type (this requires the set_integer_now_func() to be set).
*********************************************************************************************************************************************
The above example of add_compression_policy command compresses a chunk automatically in the background after it reaches age of 2678400000
ms or 31 day. (Typical for Ignition Historian with 1 year data retention policy).

tsdb=# SELECT add_retention_policy('sqlth_1_data', BIGINT '31622400000');
 add_retention_policy
----------------------
                 1001
(1 row)
*********************************************************************************************************************************************
*         add_retention_policy()
*
*                   Create a policy to drop chunks older than a given interval of a particular hypertable or continuous aggregate on a
*                   schedule in the background. For more information, see the drop_chunks section. This implements a data retention policy
*                   and removes data on a schedule. Only one retention policy may exist per hypertable.
*********************************************************************************************************************************************
The above example of add_retention_policy() command  (Typical for Ignition Historian with 1 year (366 days) data retention policy.
         Set the “fixed_schedule” flag to TRUE and set specific “next_start” schedules for both jobs.
         Note! In this case the “retention” job 1001 will execute every day at 20:10 (UTC -6 hrs), and
         the “compression” job 1000 will follow 5 min later at 20:15 (UTC -6 hrs).

tsdb=# SELECT alter_job(1000, fixed_schedule => true, initial_start => '2024-02-08 20:15:00.000000-06', next_start => 
'2024-02-09 20:15:00.000000-06');


tsdb=# SELECT alter_job(1001, fixed_schedule => true, initial_start => '2024-02-08 20:10:00.000000-06', next_start => 
'2024-02-09 20:10:00.000000-06');

*********************************************************************************************************************************************
*         fixed_schedule
*
*                   Set to FALSE if you want the next start of a job to be determined as its last finish time plus the schedule interval.
*                   Set to TRUE if you want the next start of a job to begin schedule_interval after the last start.
*********************************************************************************************************************************************
```

- Checking hypertable maintenance jobs are configured correctly

```
tsdb=# SELECT job_id, application_name, schedule_interval, retry_period, scheduled, fixed_schedule, config, 
next_start, hypertable_name FROM timescaledb_information.jobs where hypertable_name ='sqlth_1_data';
 job_id |     proc_name      | schedule_interval | scheduled | fixed_schedule |       next_start       | hypertable_name
--------+--------------------+-------------------+-----------+----------------+------------------------+-----------------
   1000 | policy_compression | 1 day             | t         | t              | 2024-02-09 20:15:00-06 | sqlth_1_data
   1001 | policy_retention   | 1 day             | t         | t              | 2024-02-09 20:10:00-06 | sqlth_1_data
(2 rows)
```

- Verifying hypertable maintenance jobs execution statistics after the first scheduled run (can be done later)

```
tsdb=# SELECT job_id, last_successful_finish, last_run_status, job_status, next_start, total_runs, total_failures
FROM timescaledb_information.job_stats WHERE hypertable_name ='sqlth_1_data';
 job_id |     last_successful_finish     | last_run_status | job_status |       next_start       | total_runs | total_failures
--------+-------------------------------+-----------------+------------+------------------------+------------+----------------
   1000 | 2024-02-09 20:15:00.023025-06 | Success         | Scheduled  | 2024-02-10 20:15:00-06 |          1 |              0
   1001 | 2024-02-09 20:10:00.017209-06 | Success         | Scheduled  | 2024-02-10 20:10:00-06 |          1 |              0
(2 rows)
```

- Create BRIN (Block Range Index) for the “sqlth_1_data” hypertable

```
tsdb=# CREATE INDEX sqlth_1_data_tagid_t_stamp_brinx ON sqlth_1_data USING BRIN (tagid, t_stamp) with 
(pages_per_range=64, autosummarize = on);
CREATE INDEX
tsdb=# SELECT indexname, indexdef FROM pg_indexes WHERE tablename ='sqlth_1_data';
            indexname             |                                                   indexdef
----------------------------------+------------------------------------------------------------------------------------------------------------
 sqlth_1_data_pkey                | CREATE UNIQUE INDEX sqlth_1_data_pkey ON public.sqlth_1_data USING btree (tagid, t_stamp)
 sqlth_1_datat_stampndx           | CREATE INDEX sqlth_1_datat_stampndx ON public.sqlth_1_data USING btree (t_stamp)
 sqlth_1_data_tagid_t_stamp_brinx | CREATE INDEX sqlth_1_data_tagid_t_stamp_brinx ON public.sqlth_1_data USING brin (tagid, t_stamp) WITH
                                  | (pages_per_range='64')
(3 rows)
Note! Indexes  “sqlth_1_data_pkey” & “sqlth_1_datat_stampndx” are created automatically on Storage System table infrastructure for 
Ignition Tag Historian initialization.
```

- Optionally, - disable the “sqlth_1_datat_stampndx” index

```
tsdb=# UPDATE pg_index SET indisvalid = false WHERE indexrelid = 'sqlth_1_datat_stampndx'::regclass;
UPDATE 1
```

- Reindex the “sqlth_1_table” hypertable

```
tsdb=# REINDEX (VERBOSE) TABLE public.sqlth_1_data;
INFO:  index "1_1_sqlth_1_data_pkey" was reindexed
DETAIL:  CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s
INFO:  index "_hyper_1_1_chunk_sqlth_1_datat_stampndx" was reindexed
DETAIL:  CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s
INFO:  index "_hyper_1_1_chunk_sqlth_1_data_tagid_t_stamp_brinx" was reindexed
DETAIL:  CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s
...
...
INFO:  index "3_3_sqlth_1_data_pkey" was reindexed
DETAIL:  CPU: user: 0.07 s, system: 0.03 s, elapsed: 0.11 s
INFO:  index "_hyper_1_3_chunk_sqlth_1_datat_stampndx" was reindexed
DETAIL:  CPU: user: 0.04 s, system: 0.01 s, elapsed: 0.06 s
INFO:  index "_hyper_1_3_chunk_sqlth_1_data_tagid_t_stamp_brinx" was reindexed
DETAIL:  CPU: user: 0.04 s, system: 0.00 s, elapsed: 0.04 s
REINDEX
```

- Optional (requires some historical data accumulated) example to demonstrate BRIN index in action

```
tsdb=#  EXPLAIN (ANALYZE) SELECT * FROM public.sqlth_1_data WHERE tagid =1 and t_stamp between 1707524933560 and
1708525044573;
                                                                            QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on _hyper_1_3_chunk  (cost=9.84..1896.22 rows=25903 width=556) (actual time=0.053..22.420 rows=25653 loops=1)
   Recheck Cond: ((tagid = 1) AND (t_stamp >= '1707524933560'::bigint) AND (t_stamp <= '1708525044573'::bigint))
   Rows Removed by Index Recheck: 150743
   Heap Blocks: lossy=1298
   ->  Bitmap Index Scan on _hyper_1_3_chunk_sqlth_1_data_tagid_t_stamp_brinx  (cost=0.00..3.36 rows=33622 width=0) (actual
time=0.040..0.040 rows=12980 loops=1)
         Index Cond: ((tagid = 1) AND (t_stamp >= '1707524933560'::bigint) AND (t_stamp <= '1708525044573'::bigint))
 Planning Time: 0.557 ms
 Execution Time: 23.208 ms
(8 rows)
```

- Enable autovacuum with custom parameters for the “sqlth_1_data” hypertable end exit psql and terminal

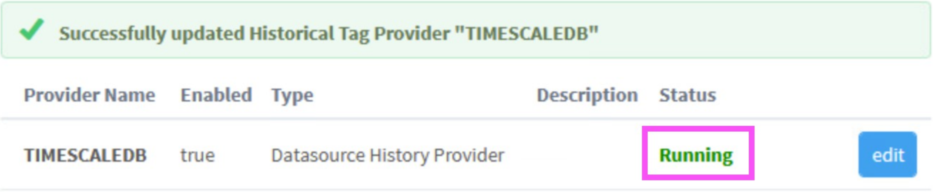
```
tsdb=# ALTER TABLE sqlth_1_data SET (autovacuum_enabled = on, autovacuum_analyze_scale_factor = 0.01,
autovacuum_vacuum_scale_factor = 0.05, autovacuum_analyze_threshold = 500, autovacuum_vacuum_threshold = 1000);
ALTER TABLE

tsdb=# \q
postgres@TIMESCALE01:~$ exit
sysadmin@TIMESCALE01:~$ exit
```

On Ignition 8 Gateway Host:

- Re-enable the “TIMESCALE” historical provider

Re-enable the history provider “TIMESCALEDB” in Ignition Gateway > Config > Tags > History >TIMESCALEDB (**Fig.14**).



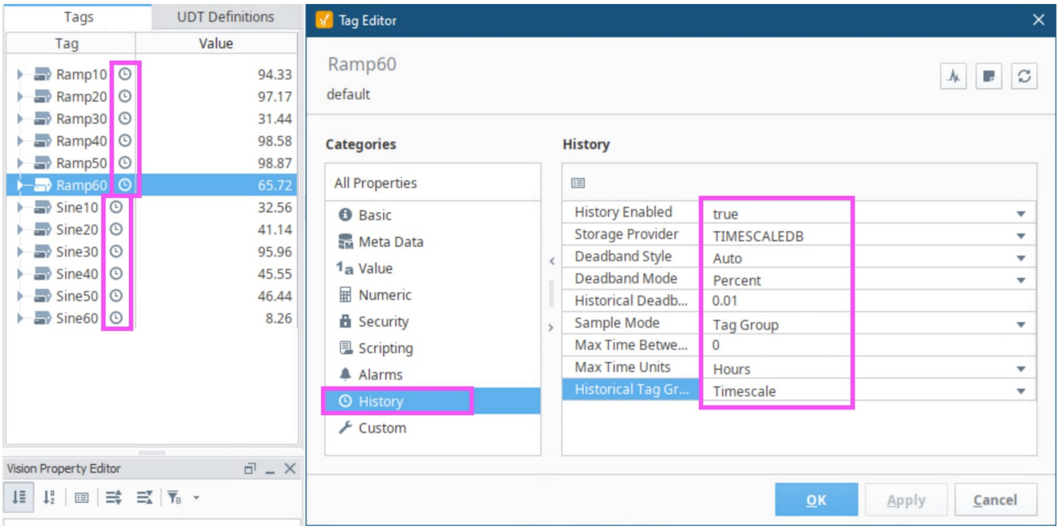
✓ Successfully updated Historical Tag Provider "TIMESCALEDB"

Provider Name	Enabled	Type	Description	Status	
TIMESCALEDB	true	Datasource History Provider		Running	edit

Fig.14

- On Ignition8 host host open the “TIMESACLE” project and finish history setup for simulated all tags.

Enable history for all remaining tags in the “TIMESCALE” project using Storage Provider = TIMESCALEDB and Historical Tag Group = Timescale (**Fig.15**).



Tag	Value
Ramp10	94.33
Ramp20	97.17
Ramp30	31.44
Ramp40	98.58
Ramp50	98.87
Ramp60	65.72
Sine10	32.56
Sine20	41.14
Sine30	95.96
Sine40	45.55
Sine50	46.44
Sine60	8.26

Ramp60
default

History

History Enabled	true
Storage Provider	TIMESCALEDB
Deadband Style	Auto
Deadband Mode	Percent
Historical Deadb...	0.01
Sample Mode	Tag Group
Max Time Betwe...	0
Max Time Units	Hours
Historical Tag Gr...	Timescale

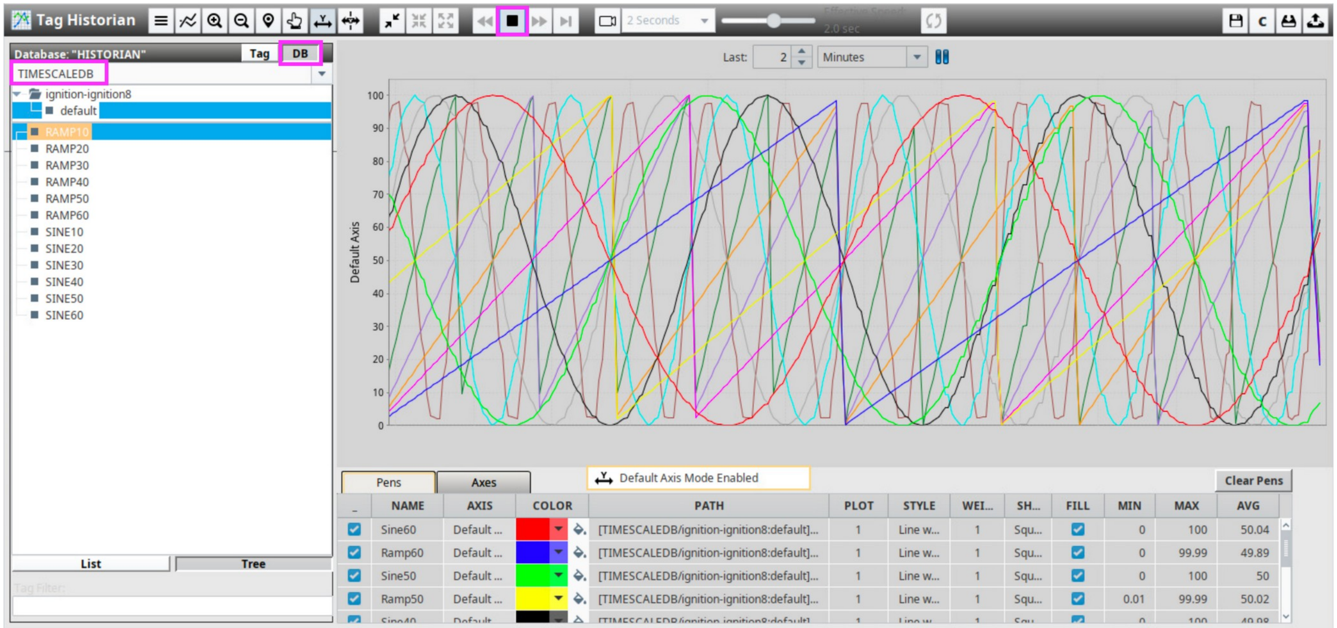
Fig.15

- Create a vision startup page with historical data trends view

Open Ignition designer application and create a startup page with Easy Chart component or similar to allow historical data trending of all 12 tags. In provided example the more versatile trending solution based on Ad Hoc Trends by Matthew Raybourn was used. It can be downloaded from the Ignition Exchange site: https://inductiveautomation.com/exchange/46/overview

- Open the TIMESCALE project in Ignition Vision Client Launcher

Example of Ad Hoc Trends component (near real-time) trending TIMESCALE DB historical data for all 12 simulated tags at the same time (**Fig.16**).



_	NAME	AXIS	COLOR	PATH	PLOT	STYLE	WEI...	SH...	FILL	MIN	MAX	AVG
☑	Sine60	Default ...	🟥	[TIMESCALEDB/ignition-ignition8:default]...	1	Line w...	1	Squ...	☑	0	100	50.04
☑	Ramp60	Default ...	🟦	[TIMESCALEDB/ignition-ignition8:default]...	1	Line w...	1	Squ...	☑	0	99.99	49.89
☑	Sine50	Default ...	🟩	[TIMESCALEDB/ignition-ignition8:default]...	1	Line w...	1	Squ...	☑	0	100	50
☑	Ramp50	Default ...	🟨	[TIMESCALEDB/ignition-ignition8:default]...	1	Line w...	1	Squ...	☑	0.01	99.99	50.02
☑	Sine40	Default	⬛	[TIMESCALEDB/ignition-ignition8:default]	1	Line w...	1	Squ...	☑	0	100	49.98

Fig.16

Enjoy!

APPENDIX A Useful queries for tags sample data, formats, and historical tables infrastructure

Time stamps conversion examples to bigint from dt and from bigint to dt:

```
tsdb=# SELECT now() AS Now_dt, CAST (extract(epoch from now()) *1000 AS BIGINT) AS Now_bigint
, to_timestamp(MAX(t_stamp/1000)) AS MAX_dt, MAX(t_stamp) AS MAX_bigint FROM public.sqlth_1_data;
           now_dt            |  now_bigint   |         max_dt          |  max_bigint
-----------------------------+---------------+-------------------------+---------------
 2024-02-12 17:27:42.7403-06 | 1707780462740 | 2024-02-12 17:27:40-06 | 1707780460382
(1 row)
```

Extract tag’s historical data samples with tagname and t_stamp in dt format:

```
tsdb=# SELECT T1.tagid, T2.tagpath, T2.datatype, T1.floatvalue, T1.dataintegrity, T1.t_stamp,
to_timestamp(T1.t_stamp/1000) t_stamp_dt
from public.sqlth_1_data T1, public.sqlth_te T2 where T1.tagid = T2.id AND T2.tagpath ='sine60'
ORDER BY T1.t_stamp DESC FETCH FIRST 10 ROWS ONLY;
 tagid | tagpath | datatype |     floatvalue     | dataintegrity |    t_stamp    |       t_stamp_dt
-------+---------+----------+--------------------+---------------+---------------+------------------------
     6 | sine60  |        1 | 32.468379974365234 |           192 | 1707780633578 | 2024-02-12 17:30:33-06
     6 | sine60  |        1 | 27.660436630249023 |           192 | 1707780632577 | 2024-02-12 17:30:32-06
     6 | sine60  |        1 | 23.102643966674805 |           192 | 1707780631576 | 2024-02-12 17:30:31-06
     6 | sine60  |        1 | 18.840131759643555 |           192 | 1707780630574 | 2024-02-12 17:30:30-06
     6 | sine60  |        1 | 14.919697761535645 |           192 | 1707780629573 | 2024-02-12 17:30:29-06
     6 | sine60  |        1 | 11.384378433227539 |           192 | 1707780628572 | 2024-02-12 17:30:28-06
     6 | sine60  |        1 |  8.272985458374023 |           192 | 1707780627572 | 2024-02-12 17:30:27-06
     6 | sine60  |        1 | 5.6196770668029785 |           192 | 1707780626571 | 2024-02-12 17:30:26-06
     6 | sine60  |        1 | 3.4516685009002686 |           192 | 1707780625570 | 2024-02-12 17:30:25-06
     6 | sine60  |        1 | 1.7970842123031616 |           192 | 1707780624569 | 2024-02-12 17:30:24-06
(10 rows)
```

Tags history statistical (test for Ignition Programmable Device Simulator):

```
tsdb=# select distinct t1.tagid, t2.tagpath, AVG(t1.floatvalue), PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY
t1.floatvalue) as median ,MIN(t1.floatvalue), MAX(t1.floatvalue), count(*) FROM public.sqlth_1_data t1,
public.sqlth_te t2 where t1.tagid = t2.id AND t1.dataintegrity =192 group by t1.tagid, t2.tagpath order by t1.tagid;
 tagid | tagpath |        avg         |       median       |          min          |        max        | count
-------+---------+--------------------+--------------------+-----------------------+-------------------+-------
     1 | sine10  |  49.99550888854456 |                 50 |                     0 |               100 | 98046
     2 | sine20  |  49.99418173383984 | 49.952877044677734 |                     0 |               100 | 97742
     3 | sine30  | 50.002804217046375 |  50.22514533996582 |                     0 |               100 | 97166
     4 | sine40  | 50.011398382560586 |     50.46337890625 |                     0 |               100 | 95935
     5 | sine50  |  50.00126923737531 |  49.27745819091797 | 3.947841946683184e-07 |               100 | 95053
     6 | sine60  |  50.00151066182858 |  50.13351821899414 |                     0 |               100 | 94072
     7 | ramp10  |  52.32011318169635 |               56.5 |                     0 | 99.98999786376953 | 37461
     8 | ramp20  |  51.91464345282335 |  55.27250099182129 |                     0 | 99.99500274658203 | 17670
     9 | ramp30  |  52.45803161829689 |  57.74333190917969 |                     0 | 99.99666595458984 | 12147
    10 | ramp40  |  51.09869536130902 |   54.3387508392334 |                     0 | 99.99749755859375 |  9378
    11 | ramp50  |  51.26109527409996 |  53.57699966430664 |                     0 | 99.99800109863281 |  7620
    12 | ramp60  |  50.43747796853882 | 49.310001373291016 |                     0 | 99.99666595458984 |  6391
(12 rows)
```

View Data of Historian DB Infrastructure Timescale Tables and Re-Initialize
To Re-Initialize execute each commented the “delete” command individually:
Note! Ignition historical provider that uses this database must be disabled.

```
-- Delete Tags Historical Records
select count(*) from public.sqlth_1_data;
--delete from public.sqlth_1_data;
--
-- Active/Retired Ignition Tags Historian Configuration
select id, tagpath, scid, datatype, querymode, TO_TIMESTAMP(created/1000), TO_TIMESTAMP(retired/1000) from
public.sqlth_te;
--delete from public.sqlth_te;
--
--Ignition Tags & their Historian Group(s)
select * from public.sqlth_scinfo;
--delete from public.sqlth_scinfo;
--
--Historian driver(s) for Participated Hosts(name) & Tag Provider(s)
select * from public.sqlth_drv;
--delete from public.sqlth_drv;
--
--Historian driver(s) timeframe & rate
select scid, TO_TIMESTAMP(start_time/1000), TO_TIMESTAMP(end_time/1000), rate from public.sqlth_sce;
--delete from public.sqlth_sce;
--
```

```
-- Historian data partitions(tables) & drivers references
select pname, drvid, TO_TIMESTAMP(start_time/1000), TO_TIMESTAMP(end_time/1000), blocksize, flags from 
public.sqlth_partitions;
--delete from public.sqlth_partitions;
--
-- annotations
select * from public.sqlth_annotations;
select * from public.saved_graphs;
select * from public.chart_annotations;
--delete from public.sqlth_annotations;
--delete from public.saved_graphs;
--delete from public.chart_annotations;
```

In case if chart_annotations and/or saved_graphs table(s) did not get created (this behaviour was observed):

```
-- DROP SEQUENCE IF EXISTS public.chart_annotations_seq;
CREATE SEQUENCE IF NOT EXISTS public.chart_annotations_seq
    INCREMENT 1
    START 1
    MINVALUE 1
    MAXVALUE 9223372036854775807
    CACHE 1;

ALTER SEQUENCE public.chart_annotations_seq
    OWNER TO ignition;

-- DROP TABLE IF EXISTS public.chart_annotations;
CREATE TABLE IF NOT EXISTS public.chart_annotations
(
    id integer NOT NULL DEFAULT nextval('chart_annotations_seq'::regclass),
    penname character varying(45) COLLATE pg_catalog."default" NOT NULL,
    xvalue timestamp(0) without time zone NOT NULL,
    yvalue double precision NOT NULL,
    note text COLLATE pg_catalog."default" NOT NULL,
    enteredby character varying(45) COLLATE pg_catalog."default" NOT NULL,
    lastupdated timestamp(0) without time zone NOT NULL,
    CONSTRAINT chart_annotations_pkey PRIMARY KEY (id)
) TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.chart_annotations
    OWNER to ignition;

-- DROP INDEX IF EXISTS public.idx_tstamp;
CREATE INDEX IF NOT EXISTS idx_tstamp
    ON public.chart_annotations USING btree
    (xvalue ASC NULLS LAST) TABLESPACE pg_default;
--*************************************************
-- DROP SEQUENCE IF EXISTS public.saved_graphs_seq;
CREATE SEQUENCE IF NOT EXISTS public.saved_graphs_seq
    INCREMENT 1
    START 1
    MINVALUE 1
    MAXVALUE 9223372036854775807
    CACHE 1;

ALTER SEQUENCE public.saved_graphs_seq
    OWNER TO ignition;

-- DROP TABLE IF EXISTS public.saved_graphs;
CREATE TABLE IF NOT EXISTS public.saved_graphs
(
    id integer NOT NULL DEFAULT nextval('saved_graphs_seq'::regclass),
    title character varying(45) COLLATE pg_catalog."default" NOT NULL,
    tagpens text COLLATE pg_catalog."default" NOT NULL,
    axes text COLLATE pg_catalog."default" NOT NULL,
    chartmode integer,
    startdate timestamp(0) without time zone,
    enddate timestamp(0) without time zone,
    rangestartdate timestamp(0) without time zone,
    rangeenddate timestamp(0) without time zone,
    unit integer,
    unitcount integer,
    username character varying(45) COLLATE pg_catalog."default",
    lastmodified timestamp(0) without time zone NOT NULL,
    CONSTRAINT saved_graphs_pkey PRIMARY KEY (id)
) TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.saved_graphs
    OWNER to ignition;
```

APPENDIX B Useful queries for Postgresql Timescale features and maintenance

View Autovacuum activities:

```
tsdb=# SELECT relname, last_vacuum, last_autovacuum FROM pg_stat_user_tables
WHERE last_autovacuum is NOT NULL ORDER BY last_autovacuum DESC;
     relname      | last_vacuum |        last_autovacuum
------------------+-------------+-------------------------------
 sqlth_sce        |             | 2024-02-12 17:45:17.738351-06
 _hyper_1_5_chunk |             | 2024-02-12 16:47:24.420229-06
 _hyper_1_4_chunk |             | 2024-02-11 01:09:40.624457-06
 _hyper_1_3_chunk |             | 2024-02-10 15:38:59.95357-06
 bgw_job_stat     |             | 2024-02-09 20:07:52.350248-06
 _hyper_1_2_chunk |             | 2024-02-09 17:43:39.675193-06
(6 rows)
```

View specific database size:

```
tsdb=# SELECT pg_size_pretty(pg_database_size('tsdb'));
 pg_size_pretty
----------------
 73 MB
(1 row)
```

Get the disk space used by an index on a hypertable, including the disk space needed to provide the index on all chunks. The size is reported in bytes:

```
tsdb=# SELECT hypertable_index_size('sqlth_1_data_tagid_t_stamp_brinx');
 hypertable_index_size
-----------------------
                147456
(1 row)
```

Show hypertable chunks:

```
tsdb=# SELECT show_chunks('sqlth_1_data');
              show_chunks
----------------------------------------
 _timescaledb_internal._hyper_1_1_chunk
 _timescaledb_internal._hyper_1_2_chunk
 _timescaledb_internal._hyper_1_3_chunk
 _timescaledb_internal._hyper_1_4_chunk
 _timescaledb_internal._hyper_1_5_chunk
(5 rows)
```

Maintenance jobs setup and modifications

```
-- ==== Maintenance JOBs =====
-- Show all jobs for timescaledb:
SELECT * FROM timescaledb_information.jobs;
--
-- ==== HYPERTABLE RETENTION POLICY ====
-- to remove JOB for retention policy on a given 'sqlth_X_data' hypertable
SELECT remove_retention_policy('sqlth_1_data');
--
-- to add retention policy to 'sqlth_X_data' hypertable to drop chunks older than:
--  30 days (2592000000  ms), OR
-- 366 days (31622400000 ms)
SELECT add_retention_policy('sqlth_1_data', BIGINT '2592000000');
--
-- ==== HYPERTABLE COMPRESSION POLICY ======
-- to remove JOB for compression policy on a given 'sqlth_X_data' hypertable
SELECT remove_compression_policy('sqlth_1_data');
--
-- to add compression policy to 'sqlth_X_dat' hypertable for chunks age more than:
--  7 days (604800000  ms), OR
-- 30 days (2592000000 ms), OR
-- 31 days (2678400000 ms)
SELECT add_compression_policy('sqlth_1_data', BIGINT '604800000');
```

APPENDIX C References

Ignition Tag Historian Module:
https://inductiveautomation.com/resources/article/scada-historian

BRIN Indexes by PostgreSQL:
https://www.postgresql.org/docs/current/brin-intro.html

BRIN Index for PostgreSQL by PERCONA:
https://www.percona.com/blog/brin-index-for-postgresql-dont-forget-the-benefits/

Optimizing Postgres's Autovacuum for High-Churn Tables by Adam Hendel:
https://tembo.io/blog/optimizing-postgres-auto-vacuum

Enabling and Disabling Autovacuum in Postgresql by Hans-Jürgen Schönig:
https://www.cybertec-postgresql.com/en/enabling-and-disabling-autovacuum-in-postgresql/

Postgres Indexing: When Does BRIN Win? By Paul Ramsey:
https://www.crunchydata.com/blog/postgres-indexing-when-does-brin-win

Create an BRIN Index at a Fraction of the Normal Size by Digoal Zhou:
https://www.alibabacloud.com/blog/create-an-brin-index-at-a-fraction-of-the-normal-size_595138

Install self-hosted TimescaleDB on Linux:
https://docs.timescale.com/self-hosted/latest/install/installation-linux/

{EOF}