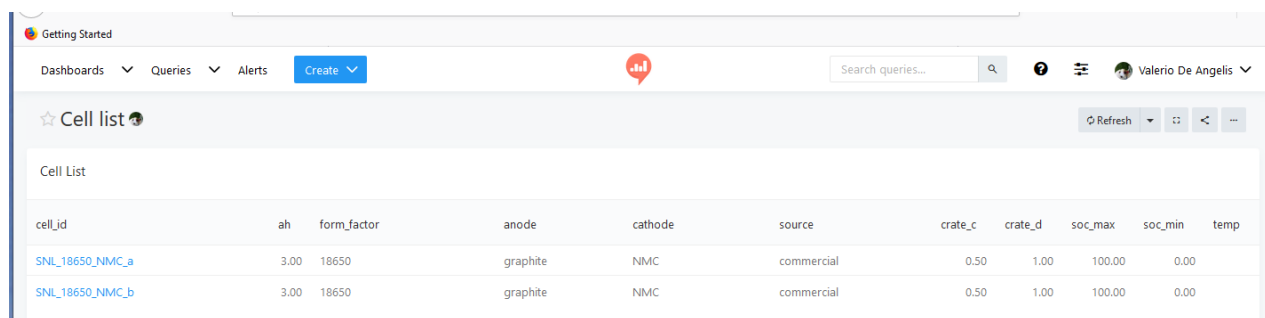


Battery Lifecycle (BLC) Framework

The Battery Lifecycle (BLC) Framework is an open-source platform that provides tools to visualize, analyze, and share battery data through the technology development cycle, including data from material characterization, cell testing, manufacturing, and field testing. The BLC framework provides users with a unified view of their data that they can use to optimize materials and cell configurations, validate cell performance under application conditions, and mitigate manufacturing variations and field failures. BLC has four components: data importers, one or more databases, a front-end for querying the data and creating visualizations, and an application programming interface to process the data. BLC supports multiple users with different access permissions. Instead of building the system from the ground up, we developed BLC around Redash, a robust open-source extract-transform-load engine. BLC has been deployed for two applications: (i) tracking the development of a single battery technology from the lab to a manufacturing line and systems installed in the field, and (ii) comparing studies of multiple cells of the same battery chemistry and configuration. The latter implementation is publicly available at www.BatteryArchive.org. This document summarizes the steps needed to install BLC and get started with an implementation similar to the Battery Archive public site. Details about the software architecture can be found De Angelis et. al. [1].

Installation guide for the BLC Framework

At the end of this installation guide, you should have two dashboards in an instance of BLC running on a Linux-based system. One dashboard includes a list of cells available in the database (Figure 1), and one dashboard shows graphs for the cells (Figure 2).



cell_id	ah	form_factor	anode	cathode	source	crate_c	crate_d	soc_max	soc_min	temp
SNL_18650_NMC_a	3.00	18650	graphite	NMC	commercial	0.50	1.00	100.00	0.00	
SNL_18650_NMC_b	3.00	18650	graphite	NMC	commercial	0.50	1.00	100.00	0.00	

Figure 1: Cell List dashboard that links to the cell data



Figure 2: Cell data dashboard

1. Open source files

The open source package is available on GitHub at <https://github.com/battery-lcf/battery-archive-sandbox>

It contains the following files:

- (1) Blc_installation_steps.pdf: this document
- (2) Setup.sh: prepares the environment
- (3) Docker-compose.yml: main installation file
- (4) Sub-folder data/snl
 - a. Cell_list.xlsx: contains cell and test metadata for 2 cells
 - b. Subfolder SNL_18650_NMC1.zip: contains test data for a cell
 - c. Subfolder SNL_18650_NMC2.zip: contains test data for a cell
- (5) Sub-folder datasources/postgres/cell_data
 - a. init.sql: database schema
- (6) Sub-folder scripts
 - a. data_import.py: data import from excel to the postgres database
 - b. battery-blc-library.yaml: configuration parameters
 - c. requirements.txt: libraries required by data_import.py
 - d. Projects folder:
 - i. Getting_started.ipynb: contains a simple Jupyter Notebook script to extract data from the database for further processing
 - e. Redash_PGSQL_queries: contains the .sql files for all the queries used in www.batteryarchive.org, Python scripts to import and export the queries, and a readme.txt file with instructions.

The installer will create and start docker containers with redash (the web interface) and a postgres database with a schema designed for battery data. It also contains a script to add data from Excel files generated from the Arbin Tester. The rest of the guide contains the steps to get the system up and running and to gain familiarity with it, so that you can import your data and create more plots.

2. Access a Linux machine

The BLC Framework runs on Linux. You can install the framework on any Linux machine with at least 4 GB of RAM. This guide discusses all the steps necessary to install the software on UBUNTU, but similar steps can be followed to install the software on other flavors of Linux. If you don't have access to a Linux machine, you can get a virtual machine on AWS, Google Cloud, or Digital Ocean. Setting up a virtual machine on Digital Ocean takes about ten minutes. The following section describes how to get started on Digital Ocean. If you already have a Linux machine, you can jump ahead to "Install required packages."

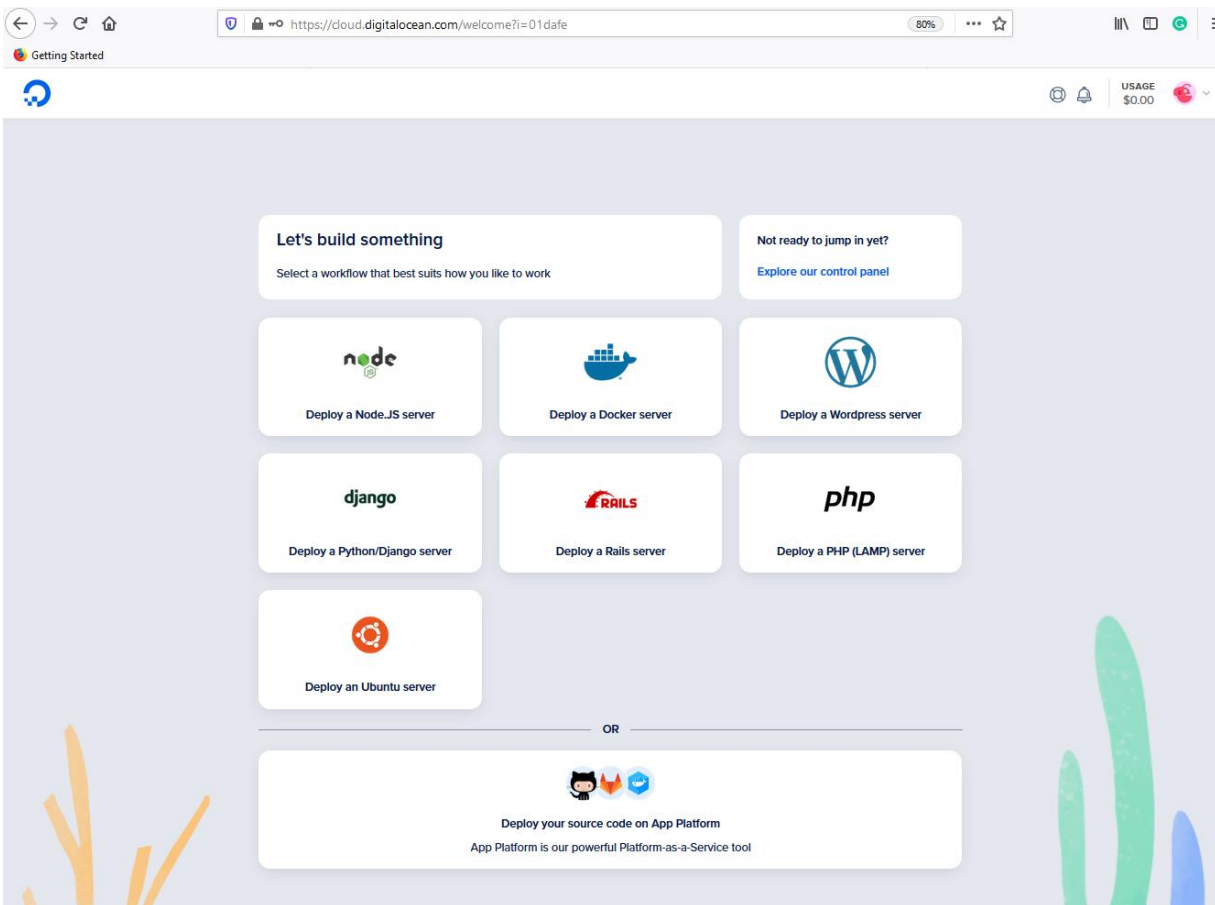
<https://www.digitalocean.com/>

Sign up for a new account

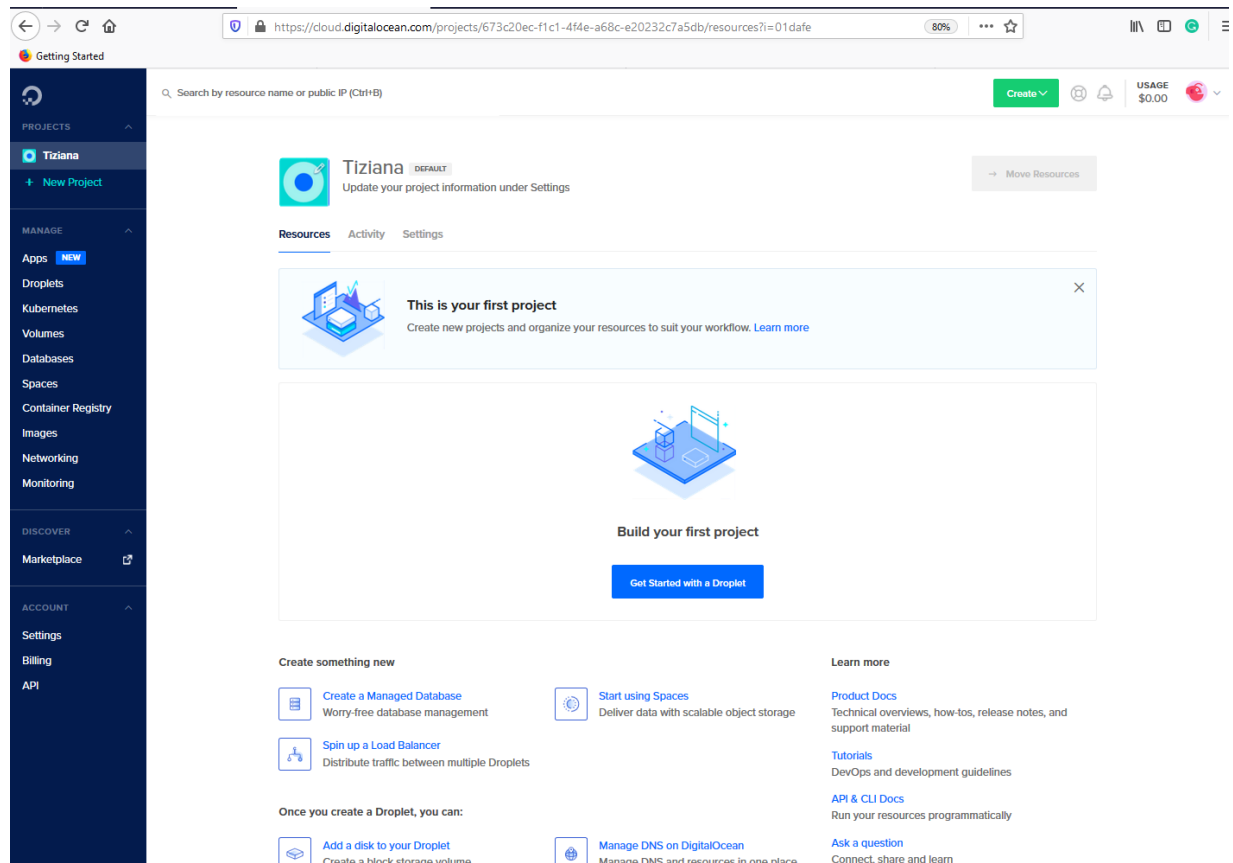
Add a credit card

Click on *Let's make something*

You will see a screen like:



Click on Explore our control panel. You will see a screen like:








Click on Get Started with a Droplet. You will see a page that looks like:

Create Droplets

Choose an image ?

[Distributions](#) [Container distributions](#) [Marketplace](#) [Custom images](#)

 Ubuntu 20.04 (LTS) x64 ▼	 FreeBSD Select version ▼	 Fedora Select version ▼	 Debian Select version ▼	 CentOS Select version ▼
---	---	--	--	--

Choose a plan

[Help me choose](#) ?

SHARED CPU	DEDICATED CPU			
Basic	General Purpose	CPU-Optimized	Memory-Optimized	Storage-Optimized NEW

Basic virtual machines with a mix of memory and compute resources. Best for small projects that can handle variable levels of CPU performance, like blogs, web apps and dev/test environments.

CPU options: ☐ Regular Intel with SSD ☒ Premium Intel with NVMe SSD NEW ☐ Premium AMD with NVMe SSD NEW

\$6/mo \$0.009/hour 1 GB / 1 Intel CPU 25 GB NVMe SSDs 1000 GB transfer	\$12/mo \$0.018/hour 2 GB / 1 Intel CPU 50 GB NVMe SSDs 2 TB transfer	\$18/mo \$0.027/hour 2 GB / 2 Intel CPUs 60 GB NVMe SSDs 3 TB transfer	\$24/mo \$0.036/hour 4 GB / 2 Intel CPUs 80 GB NVMe SSDs 4 TB transfer	\$48/mo \$0.071/hour 8 GB / 4 Intel CPUs 160 GB NVMe SSDs 5 TB transfer	\$96/mo \$0.143/hour 16 GB / 8 Intel CPUs 320 GB NVMe SSDs 6 TB transfer
--	--	---	---	--	---

i Our Basic Droplet plans, formerly called Standard Droplet plans, range from 1 GB of RAM to 16 GB of RAM. [General Purpose Droplets](#) have more overall resources and are best for production environment, and [Memory-Optimized Droplets](#) have more RAM and disk options for RAM intensive applications.

Each Droplet plan includes free outbound data transfer which is shared between all Droplets each billing cycle. Inbound bandwidth to Droplets is always free. [Learn more](#) or [try our price calculator](#). ?

Select the \$24/m choice to make sure that you have enough RAM (4GB) to run the web interface and the data importer. Keep scrolling. The bottom of the page will look like:

Authentication ?



SSH keys

A more secure authentication method



Password

Create a root password to access Droplet (less secure)

Create root password *

Type your password



You will not be sent an email containing the Droplet's details or password. Please store your password securely.

Finalize and create

How many Droplets?

Deploy multiple Droplets with the same [configuration](#).

Choose a hostname

Give your Droplets an identifying name you will remember them by. Your Droplet name can only contain alphanumeric characters, dashes, and periods.

— 1 Droplet +

ubuntu-s-2vcpu-4gb-intel-nyc3-01

Add tags

Use tags to organize and relate resources. Tags may contain letters, numbers, colons, dashes, and underscores.

Type tags here

Select Project

Assign Droplets to a project

This project has been selected
as you only have one project



Tiziana



Add backups



Enable backups

RECOMMENDED

A system-level backup is taken once a week, and each backup is retained for 4 weeks.

\$4.80/mo (per Droplet)

20% of the Droplet price

Create Droplet

Create a root password that meets the requirements and click Create Droplet (the Create Droplet button turns from gray to green when you have entered an acceptable root password). In the next page, you will see the address of the Droplet you created. Note the IP address. You will need it to access your new Linux machine and later to access the BLC Framework.



Battery Info

DEFAULT

Update your project information under Settings

→ Move Resources

Resources

Activity

Settings

DROPLETS (1)



-s-1vcpu-2gb-nyc3-01

104.236.80.52



Go to a command window (CMD in Windows or Terminal on a MAC) and type:

```
ssh root@your IP
```

Follow the steps to login. You will see a screen like:

```
root@ubuntu: ~
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\vdeange>ssh root@104.236.80.52
root@104.236.80.52's password:
Permission denied, please try again.
root@104.236.80.52's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-140-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon Apr 19 05:21:52 UTC 2021

System load:  0.02               Users logged in:      1
Usage of /:   8.9% of 48.29GB    IP address for eth0:  104.236.80.52
Memory usage: 90%              IP address for eth1:  10.108.0.2
Swap usage:   0%                IP address for docker0: 172.17.0.1
Processes:   141                IP address for br-18c730c062d7: 172.18.0.1

31 packages can be updated.
13 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Apr 19 05:06:46 2021 from 98.185.251.32
root@ubuntu:~#
```

Congratulations! You have created your first Linux virtual machine and can now install the BLC Framework.

3. Install required packages

Install docker-compose, docker, unzip, pwgen, and pip3. If you are using UBUNTU, you can use apt-install to set up the machine:

```
sudo apt update
sudo apt install docker-compose
sudo apt install docker.io
sudo systemctl enable --now docker
sudo apt install pwgen
sudo apt install python3-pip
```

4. Install redash, Postgres and the database schema

At the end of this step you should have a working version of BLC accessible from a browser at the internet BLC web address of your Linux instance.

Download the package from Github by executing the command:

```
git clone https://github.com/battery-lcf/battery-archive-sandbox.git
```

After the code is downloaded, you can start the installation process by executing each of the commands below:

```
cd battery-archive-sandbox
sudo ./setup.sh
more env
```

The command `more env` will display the content of the env file. Note the value of the variables `POSTGRES_PASSWORD` and `PGADMIN_DEFAULT_PASSWORD`, you will need it later when you will connect the web interface to the database (`POSTGRES_PASSWORD`) and when you access the database administrator interface (`PGADMIN_DEFAULT_PASSWORD`).

5. Installation of the python data_import.py and import of the first dataset

Follow the steps below to set up and run the Python import script. Run each of the commands:

```
cd scripts
```

```
pip3 install -r requirements.txt
```

```
python3 data_import.py -m add -p ../data/snl/ &
```

The importer reads the list of files to be imported from a file named cell_list.xlsx located in the data/snl folder. The file has a well-defined format as shown in the figure below.

	A	B	C	D	E	F	G	H	I	J	K	L
1	file_id	cell_id	cathode	anode	temperature	soc_max	soc_min	source	crate_c	crate_d	ah	form_factor
2	SNL_18650_G5_NMC1	SNL_18650_NMC_a	NMC	graphite	35	100	0	commercial	0.5	1	3	18650
3	SNL_18650_G5_NMC2	SNL_18650_NMC_b	NMC	graphite	35	100	0	commercial	0.5	1	3	18650
4												
5												
6												

File_id corresponds to the name of a zip file (file_id.zip) that contains test data files from the cells to be imported. Cell_id is the identifier that the cell will be given in the database. The other quantities are used to characterize the cell (cathode, anode, source, ah, form_factor) and the test (temperature, soc_max, soc_min). The data from the Excel file are stored in two tables in Postgres (cell_metadata and test_metadata). When the importer runs, it opens the cell_list.xlsx file, reads the file_id's and unzips the data into a folder with the same name (file_id). It then scans the file_id folder for excel files and imports the data from the files in the Postgres database. The version of the importer we are releasing is designed to import data from Excel files generated by Arbin testers. The importer looks for worksheets named Channel and for the columns CycleIndex, DateTime, Test_time(s) Voltage(V), and Current(A). The cell_list.xlsx file included with the software contains entries for two NMC cells from Preger et. al. [2].

```
df_time_series['cycle_index_file'] = df_time_series_file['Cycle_Index']
df_time_series['test_time'] = df_time_series_file['Test_Time(s)']
df_time_series['i'] = df_time_series_file['Current(A)']
df_time_series['v'] = df_time_series_file['Voltage(V)']
df_time_series['date_time'] = df_time_series_file['Date_Time']
df_time_series['filename'] = filename
```

The importer runs in the background and imports data from the 2 NMC cells. The importer generates a log (blc-python.log) file with import progress. Analyze the blc-python.log file for import results. The script may run for about 10 minutes on a Linux machine with 4 GB of RAM. To check progress, type *tail blc-python.log*. When the import is completed, the log file will have a line at the end "Done adding files."

At the end of the step, the data in data/snl should have been imported into the SQL database. The data is imported in two tables (cycle_data and timeseries_data). The database schema is discussed at the end of the document. To verify that the data was imported run the export command:

```
python3 data_import.py -m export -p ../data/snl/
```

The command should only take a few minutes to run and will generate four files in ../data/snl

SNL_18650_NMC_a_cycle_data.csv

SNL_18650_NMC_a_timeseries_data.csv

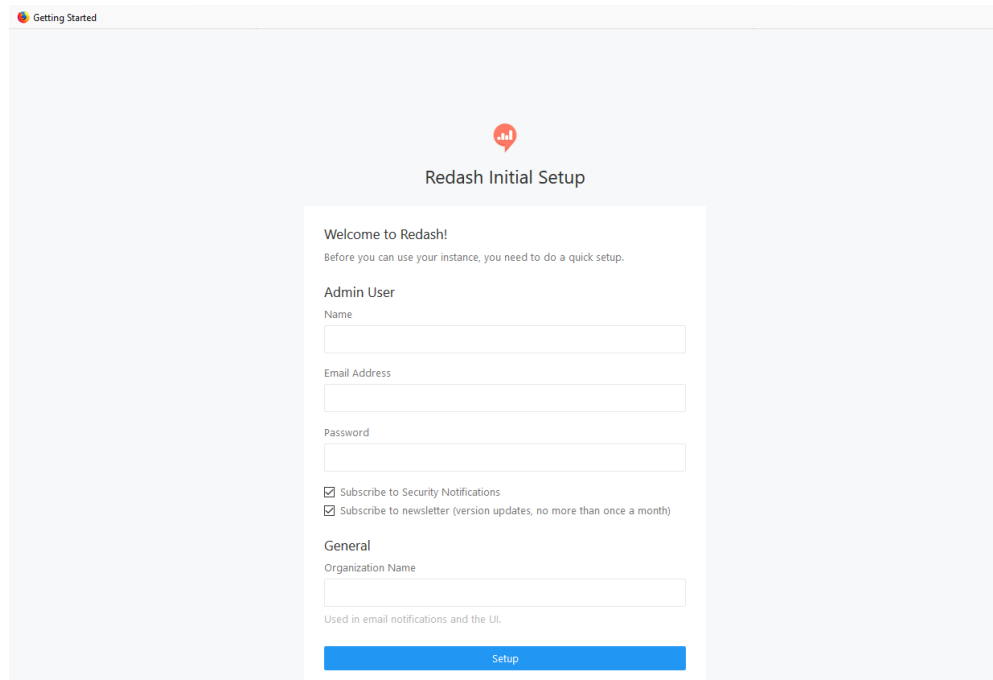
SNL_18650_NMC_b_cycle_data.csv

SNL_18650_NMC_b_timeseries_data.csv

At this point, the database is set up with sample imported data. You are now ready to set up the redash front end.

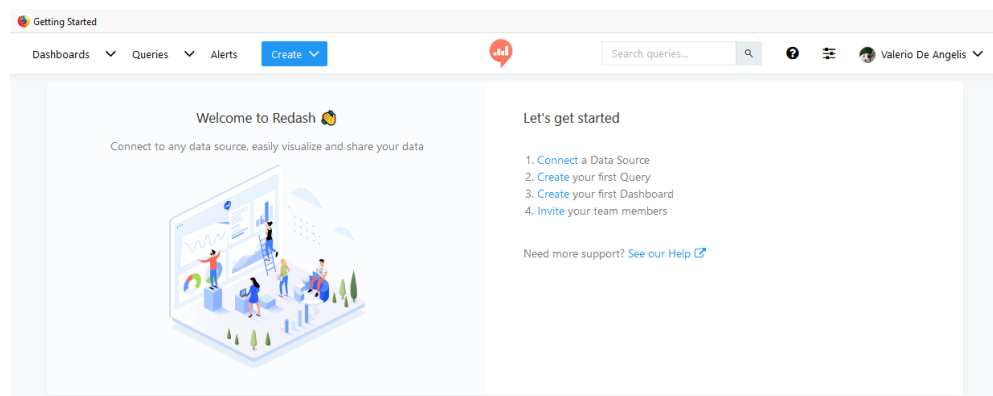
6. Set up Redash

Go with a web browser to your BLC IP address. This is the IP of your Linux machine. You should see a page like:



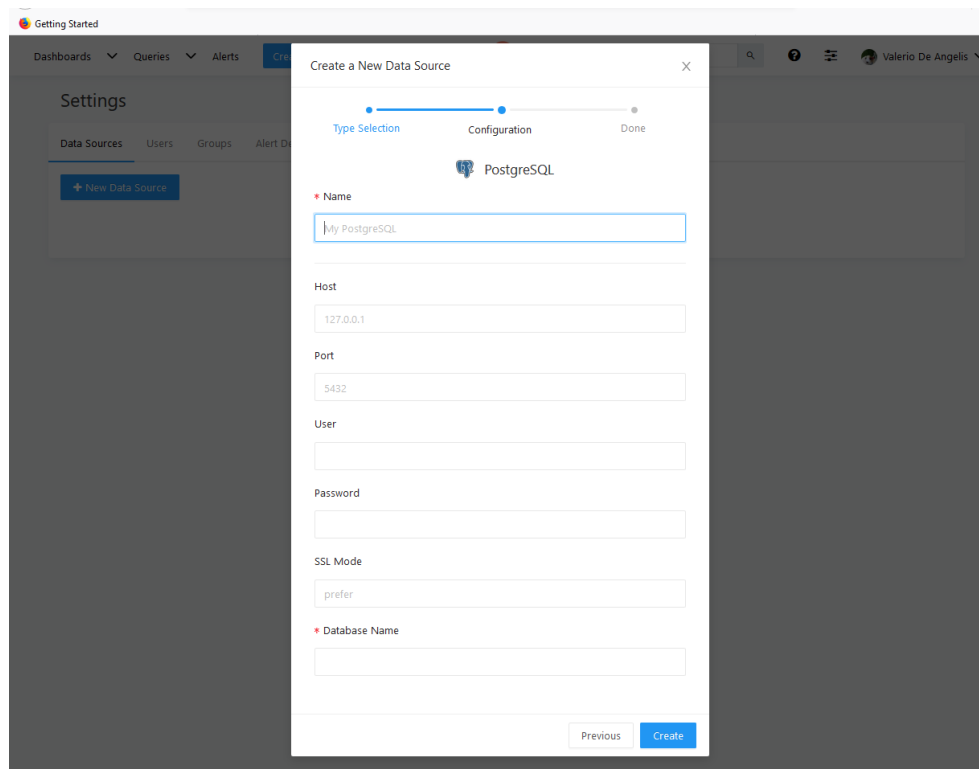
The screenshot shows the 'Redash Initial Setup' page. At the top, there's a 'Getting Started' header with a red dot icon. Below it, the Redash logo is centered. The main heading is 'Redash Initial Setup'. A white box contains the following content: 'Welcome to Redash!' followed by 'Before you can use your instance, you need to do a quick setup.' Under the 'Admin User' section, there are three input fields for 'Name', 'Email Address', and 'Password'. Below these are two checked checkboxes: 'Subscribe to Security Notifications' and 'Subscribe to newsletter (version updates, no more than once a month)'. The 'General' section has an 'Organization Name' input field with a note 'Used in email notifications and the UI.' at the bottom. A blue 'Setup' button is at the very bottom of the form.

Create an admin user: enter a name, email, passwords, and organization name. Click set up. You will be directed to a page that looks like:



The screenshot shows the Redash main dashboard after setup. The top navigation bar includes 'Dashboards', 'Queries', 'Alerts', and a 'Create' button. A search bar for queries is on the right, along with a user profile for 'Valerio De Angelis'. The main content area has a 'Welcome to Redash' message with an illustration of people working on a dashboard. To the right, a 'Let's get started' section lists four steps: '1. Connect a Data Source', '2. Create your first Query', '3. Create your first Dashboard', and '4. Invite your team members'. A link to 'See our Help' is at the bottom of this list.

Click on Connect a Data Source and select Postgres from the list of data connections. You will see a screen like:



On this page, you will connect to the database that contains the sample data you imported in the previous section. As Name enter local. As Host enter the IP address of the BLC web app. As Port enter 5432. The User is postgres, the Database Name is postgres. The password is the value of variable POSTGRES_PASSWORD in the env file in the battery-archive-sandbox folder. To see the content of the file, type more env in the battery-archive-sandbox folder as discussed in Section 4. Click on Create and then click on Test connection. You now have a working connection to the database and can create queries and dashboards to visualize the data, as described in the following sections.

7. Build queries:

The table and graphs in Figures 1 and 2 are generated by four queries to access the data from the postgres database that we connected to the web interface. Go to Create > Query using the ribbon at the top of the page.

Query 1. Cell list:

Copy the query text below to generate the table of cell and test metadata that will be used in Figure 1. Name the query Cell List (to rename a query click on New Query), then Save and Publish it. Finally, Execute the script. This should produce a small table with all of the cell metadata.

```
SELECT cell_metadata.cell_id,
       ah,
       form_factor,
       anode,
       cathode,
       SOURCE,
       crate_c,
       crate_d,
       soc_max,
       soc_min,
       TEMP
FROM cell_metadata
INNER JOIN test_metadata ON cell_metadata.cell_id = test_metadata.cell_id
ORDER BY cell_id
```

Dashboards

Queries

Alerts

Create

Search queries...

Cells List

cell_data

Search schema...

cell_metadata

cycle_data

test_metadata

timeseries_data

```

1 SELECT cell_metadata.cell_id,
2       ah,
3       form_factor,
4       anode,
5       cathode,
6       SOURCE,
7       crate_c,
8       crate_d,
9       soc_max,
10      soc_min,
11      TEMP
12 FROM cell_metadata
13 INNER JOIN test_metadata ON cell_metadata.cell_id = test_metadata.cell_id
14 ORDER BY cell_id
15

```

Save

Execute

Table

+ New Visualization

cell_id	ah	form_factor	anode	cathode	source	crate_c	crate_d	sc
SNL_18650_NMC_a	3.00	18650	graphite	NMC	commercial	0.50	1.00	
SNL_18650_NMC_b	3.00	18650	graphite	NMC	commercial	0.50	1.00	

2 rows

0 seconds runtime

Updated 2 minutes ago

Query 2. Cell DD:

Create a new query. The query text below generates a list of cells that will be used to populate the dropdown in Figure 2. Name the query Cell DD, then Save and Publish. Finally, Execute the script. This should produce a short table of Cell IDs.

```

SELECT cell_id
FROM cell_metadata
ORDER BY cell id

```

Dashboards ▾ Queries ▾ Alerts Create ▾ Search queries... ? ⌵ ⌵

☆ Cell DD V2 Unpublished 🚀 Publish 📊 Show Data Only ⋮

cell_data ▾
 ↻

cell_metadata
cycle_data
test_metadata
timeseries_data

```

1 SELECT cell_id
2 FROM cell_metadata
3 ORDER BY cell_id
4

```

{{ }} 📊 ⚡

Save ▶ Execute

Table + New Visualization

cell_id
SNL_18650_NMC_a
SNL_18650_NMC_b

Query 3. Cycle statistics

Create a new query. This query text below generates the data needed to populate the graphs at the top of Figure 2 with the cell statistics for the cells selected. Name the query Cycle statistics and Save.

```





SELECT KEY || ': ' || r.cell_id AS series,
       r.cycle_index,
       r.test_time,
       value
FROM
  (SELECT cell_id,
         trunc(cycle_index,0) AS cycle_index,
         test_time,
         json_build_object('e_d', round(e_d,3), 'ah_d', round(ah_d,3)) AS line
   FROM cycle_data
   WHERE cell_id IN ({{cell_id}})) AS r
JOIN LATERAL json_each_text(r.line) ON (KEY ~ '[e,ah]_[d]')
WHERE cast(value AS numeric)!=0
GROUP BY r.cell_id,
         r.cycle_index,
         r.test_time,
         json_each_text.key,
         json_each_text.value
ORDER BY r.cell_id,
         r.cycle_index,
         KEY

```


After you click Save, a box appears in the query editor to enter the value of the parameter cell_id. To complete the setup of the dropdown.

- (1) Click on the setting wheel next to cell_id
- (2) Select “Query Based Dropdown List” as Type
- (3) Select Cell DD as query
- (4) Check the box to Allow multiple values
- (5) Select Single Quotation Marks under quotation

Select (with a check mark) the 2 cells in the dropdown and click on Apply Changes.

Dashboards ▾ Queries ▾ Alerts **Create ▾**  Search queries...    ▾




☆ Cycle statistics V2 Unpublished Publish Show Data Only ...





cell_data ▾
 Search schema... 

- cell_metadata
- cycle_data
- test_metadata
- timeseries_data

```





1 SELECT KEY || ':' || r.cell_id AS series,
2   r.cycle_index,
3   r.test_time,
4   value
5 FROM
6   (SELECT cell_id,
7     trunc(cycle_index,0) AS cycle_index,
8     test_time,
9     json_build_object('e_d', round(e_d,3), 'ah_d', round(ah_d,3)) AS line
10  FROM cycle_data
11  WHERE cell_id IN ({{cell_id}}) AS r
12  JOIN LATERAL json_each_text(r.line) ON (KEY ~ '[e,ah]_d')
13  WHERE cast(value AS numeric)!=0
14  GROUP BY r.cell_id,
15   r.cycle_index,
16   r.test_time,
  
```

({{}})    Save Execute


cell_id 
 SNL_18650_... X SNL_18650_... X 
☒ SNL_18650_NMC_a 
☐ SNL_18650_NMC_b 

* does not exist LINE 11: WHERE cell_id IN (SNL_18650_NMC_a)) AS r ^

Execute the query to see the results. This should produce a table with capacity and energy statistics for each cycle. Then, Publish the query.

Dashboards ▾ Queries ▾ Alerts **Create ▾**  Search queries...    ▾




☆ Cycle statistics V2 Unpublished Publish Show Data Only ...

cell_data ▾
 Search schema... 

- cell_metadata
- cycle_data
- test_metadata
- timeseries_data

```

1 SELECT KEY || ':' || r.cell_id AS series,
2   r.cycle_index,
3   r.test_time,
4   value
5 FROM
6   (SELECT cell_id,
7     trunc(cycle_index,0) AS cycle_index,
8     test_time,
9     json_build_object('e_d', round(e_d,3), 'ah_d', round(ah_d,3)) AS line
10  FROM cycle_data
11  WHERE cell_id IN ({{cell_id}}) AS r
12  JOIN LATERAL json_each_text(r.line) ON (KEY ~ '[e,ah]_d')
13  WHERE cast(value AS numeric)!=0
14  GROUP BY r.cell_id,
15   r.cycle_index,
16   r.test_time,
  
```

({{}})    Save Execute


cell_id 
 SNL_18650_... X SNL_18650_... X ▾

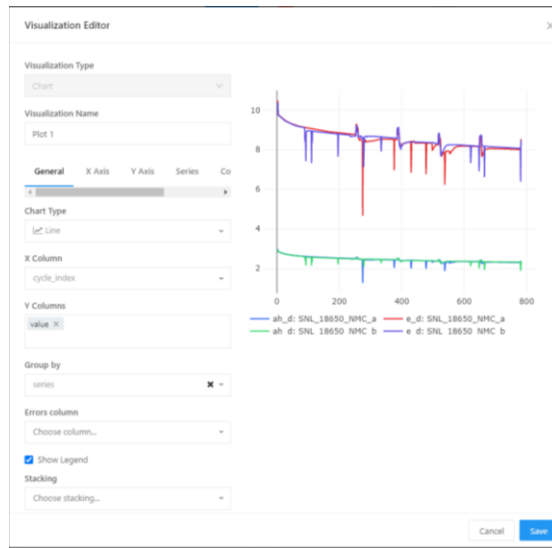
Table + New Visualization

series	cycle_index	test_time	value
ah_d: SNL_18650_NMC_a	1.00	15,998.06	2.964
e_d: SNL_18650_NMC_a	1.00	15,998.06	10.485
ah_d: SNL_18650_NMC_a	2.00	32,178.93	2.954
e_d: SNL_18650_NMC_a	2.00	32,178.93	10.466

Add description

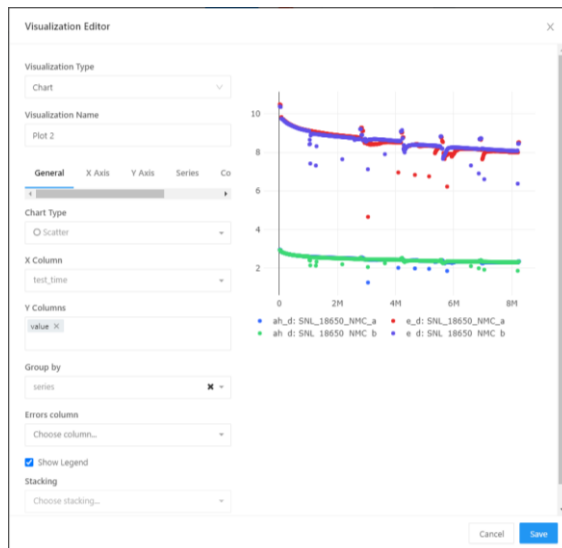
Add Plot 1: The plot will show charge and discharge energy (e_c, e_d) and capacity (ah_c, ah_d) as a function of cycle number.

- (1) Click on New Visualization and name it
- (2) Select Line as Chart Type
- (3) Select cycle_index as X
- (4) Select value as Y
- (5) Select series as Group by
- (6) Save the visualization



Add Plot 2: The plot will show charge and discharge energy (e_c , e_d) and capacity (ah_c , ah_d) as a function of test time.

- (1) Click on New Visualization and name it
- (2) Select Scatter as Chart Type
- (3) Select test_time as X
- (4) Select value as Y
- (5) Select series as Group by
- (6) Save the visualization.



Query 4. Voltage and Current:

Create a new query. The query text below generates the voltage and current plots displayed at the bottom of Figure 2.

```

SELECT KEY || ': ' || r.cell_id AS series_1,
KEY || ' ' || cycle_index || ': ' || r.cell_id AS series_2,
      r.cycle_index,
      r.test_time,
      r.cycle_time,
      value
FROM
  (SELECT timeseries_data.cell_id,
        cycle_index,
        test_time,
        cycle_time,
        json_build_object('V', v, 'C', i) AS line
   FROM timeseries_data TABLESAMPLE BERNOULLI ({{sample}})
   WHERE cell_id IN ({{cell_id}})
        AND cycle_index IN ({{cycles}})) AS r
JOIN LATERAL json_each_text(r.line) ON (KEY ~ '[V,C]')
ORDER BY r.cell_id,
      r.test_time,
      r.cycle_time,
      KEY

```

This query takes 3 parameters. For the cell_id parameter, repeat the steps used to set up the cycle statistics query. All Cell IDs in the dropdown menu should be selected (with a check mark). For the sample parameter enter 50 and for the cycles parameter enter numbers separated by commas, e.g., 1,10,20. Execute the query to see the results. This should produce a table with voltage curves for each cycle. Save and Publish the query.

The screenshot shows the 'Voltage and Current' query editor. The SQL query is pasted into the editor. The parameters are set as follows:

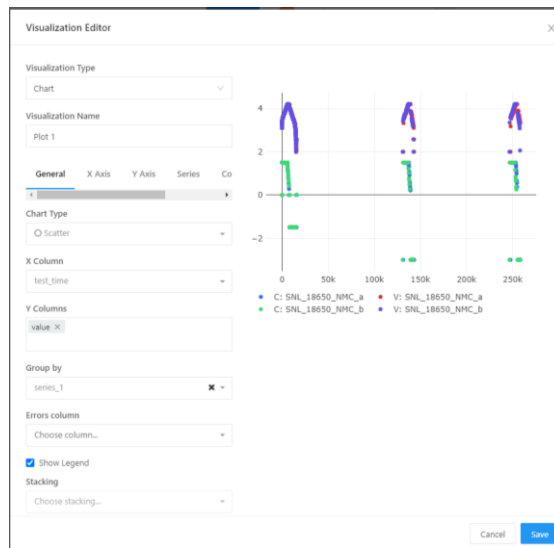
- sample: 50
- cell_id: SNL_18650_NMC_a
- cycles: 1,10,20

The 'Table' visualization is selected, and the resulting data is shown in the table below:

series_1	series_2	cycle_index	test_time	cycle_time	value
C: SNL_18650_NMC_a	C 1: SNL_18650_NMC_a	1.00	40.01	30.01	0.0

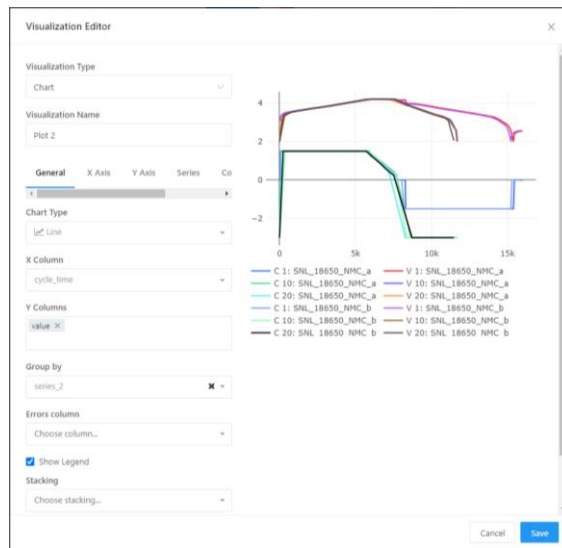
Add Plot 1: The plot will show voltage and current (v, I) as a function of test time for the cycles selected.

- (1) Click on New Visualization
- (2) Select Scatter as Chart Type
- (3) Select test_time as X
- (4) Select value as Y
- (5) Select series_1 as Group by
- (6) Save the visualization



Add Plot 2 The plot will show voltage and current (v, I) as a function of cycle time for the cycles selected (the curves overlap)

- (1) Click on New Visualization
- (2) Select Line as Chart Type
- (3) Select cycle_time as X
- (4) Select value as Y
- (5) Select series_2 as Group by
- (6) Save the visualization

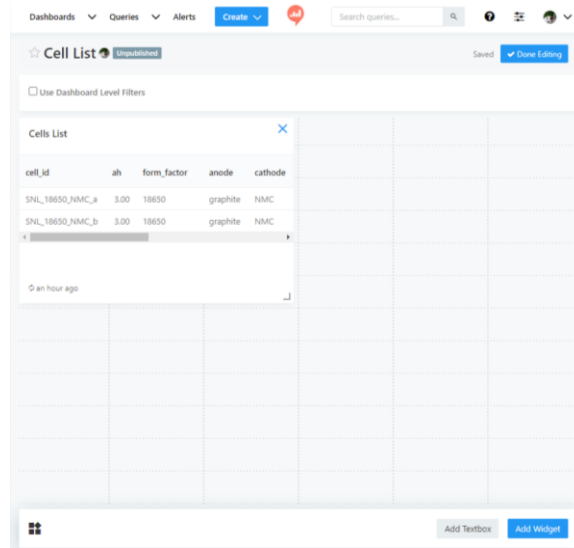


Dashboards

Dashboards can be used to display the results of multiple queries on one page.

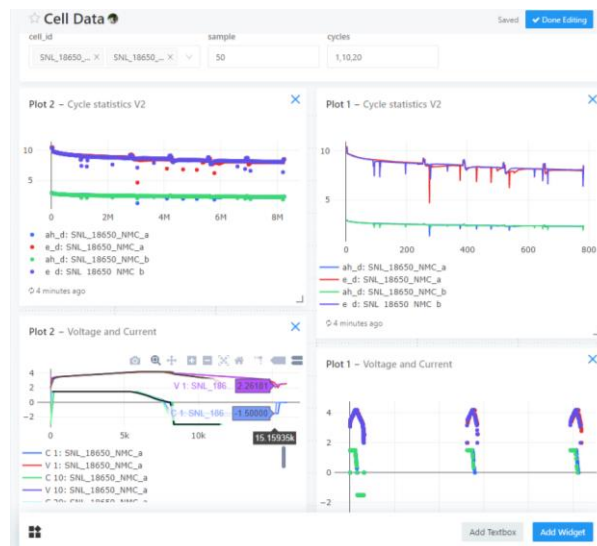
Cell list dashboard:

- 1) Create a dashboard named Cell List
- 2) Select Add Widget
- 3) Select Cell List
- 4) Stretch the table to the desired length and width
- 5) Click on Done Editing
- 6) Publish the dashboard



Cell data dashboard:

- 1) Create a dashboard named Cell Data
- 2) Select Add Widgets
- 3) Add the 2 plots under Cycle Statistics. Each plot is added as a new widget. Make sure that all cells are selected in the cell_id dropdown menu at the top of the page.
- 4) Add the 2 plots under Voltage and Current.
- 5) Stretch each of the plots to the desired length and width.
- 6) Click on Done Editing
- 7) Publish the dashboard



Link the two dashboards

This will allow you to click on a Cell ID in the Cell list dashboard and go directly to all of the plots for it in the Cell data dashboard.

- 1) Go to the Cell List query
- 2) Click on Edit Source
- 3) Click on Edit Visualization at the bottom of the page

- 4) Under the cell_id menu, for the Display As dropdown menu select Link
- 5) Enter `/dashboard/cell-data?p_cell_id=%5B"%5D` in the URL Template box. Be sure to remove any text that is in the box by default.
- 6) Unselect Open in new tab

The screenshot shows the 'Visualization Editor' window. On the left, the 'Columns' tab is active, showing a list of columns: 'cell_id' and 'ah'. Both are checked. Below this, there are settings for 'Display as' (set to 'Link' for cell_id and 'Number' for ah), 'URL template' (set to 'p_cell_id=%5B"%5D'), and 'Open in new tab' (unchecked). On the right, a preview of the table is shown with columns: cell_id, ah, form_factor, anode, cathode, and source. The data rows are: SNL_18650_NMC_a, 3.00, 18650, graphite, NMC, comme; and SNL_18650_NMC_b, 3.00, 18650, graphite, NMC, comme.

cell_id	ah	form_factor	anode	cathode	source
SNL_18650_NMC_a	3.00	18650	graphite	NMC	comme
SNL_18650_NMC_b	3.00	18650	graphite	NMC	comme

You should now have 2 populated dashboards and be ready to explore.

8. Next steps

You now have a web-based repository of battery data connected to a Postgres database (cell_data) and the basic blocks to import your data from Arbin or other testers into the database.

The database schema is very simple. It contains four tables: cell_metadata, test_metadata, cycle_data, and timeseries_data. The columns in the tables are typical of battery time series and statistics.

cell_metadata	test_metadata
cell_id	v_min
anode	cell_id
cathode	soc_max
source	soc_min
form_factor	v_max
ah	crate_c
cycle_data	temp
v_max	crate_d
cell_id	timeseries_data
v_d_mean	cycle_time
v_min	temp_2
v_c_mean	v
i_max	ah_c
ah_c	test_time
ah_eff	cell_id
ah_d	date_time
test_time	e_d
e_c	e_c
e_d	temp_1
e_eff	ah_d
i_min	cycle_index
cycle_index	i

As a next step, you can try to import some of your own Arbin files. Make an Excel file like cell_list.xlsx, and add it to a sub-folder of the data folder (i.e., new_data). Then copy in new_data folder zip files with the data named as file_id.zip. Finally run the importer as

```
python3 data_import.py -m export -p ../data/new_data/&
```

Check the log using the command

```
tail blc-python.log
```

to check your progress. When the import is completed, the log file will have a line at the end “Done adding files.”

You can also try to build your own queries. The queries in this document can serve as a guide. If you have questions on how to get data in a format you need, please let us know and we can help with building the appropriate query. If you need help at any point, you can reach us at info@batteryarchive.org.

9. Start Jupyter Notebook

You can use Jupyter Notebook to analyze the data and compare them with models. If you already have developed Python scripts, they can be immediately used in Jupyter Notebook. Matlab scripts can also be converted. To install Jupyter Notebook on UBUNTU follow these steps:

```
pip3 install jupyter
cd projects
jupyter notebook --allow-root
```

IF you are running the framework on your local machine, then you can use the localhost token shown when you start Jupyter Notebook. If you are running Jupyter Notebook on a hosted server (like the one on Digital Ocean) from a CMD window execute:

```
ssh -L 8888:localhost:8888 your_server_username@your_server_ip
```

You can then access Jupyter Notebook from your computer using the same localhost address displayed when Jupyter Notebook starts.

```
root@ubuntu-s-2vcpu-4gb-intel-nyc3-01:~/battery-archive-sandbox/scripts/projects# jupyter notebook --allow-root
[I 17:24:12.156 NotebookApp] Serving notebooks from local directory: /root/battery-archive-sandbox/scripts/projects
[I 17:24:12.156 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 17:24:12.156 NotebookApp] http://localhost:8888/?token=482ccac4b8e673b8402a3555701fb9c4707b63c4b123c35d
[I 17:24:12.156 NotebookApp] or http://127.0.0.1:8888/?token=482ccac4b8e673b8402a3555701fb9c4707b63c4b123c35d
[I 17:24:12.156 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 17:24:12.160 NotebookApp] No web browser found: could not locate runnable browser.
[C 17:24:12.160 NotebookApp]

To access the notebook, open this file in a browser:
file:///root/.local/share/jupyter/runtime/nbserver-26937-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=482ccac4b8e673b8402a3555701fb9c4707b63c4b123c35d
or http://127.0.0.1:8888/?token=482ccac4b8e673b8402a3555701fb9c4707b63c4b123c35d
```

You can then open the `getting_started.ipynb` notebook that includes the basic commands to connect to the database and plot cycle data and time series data.

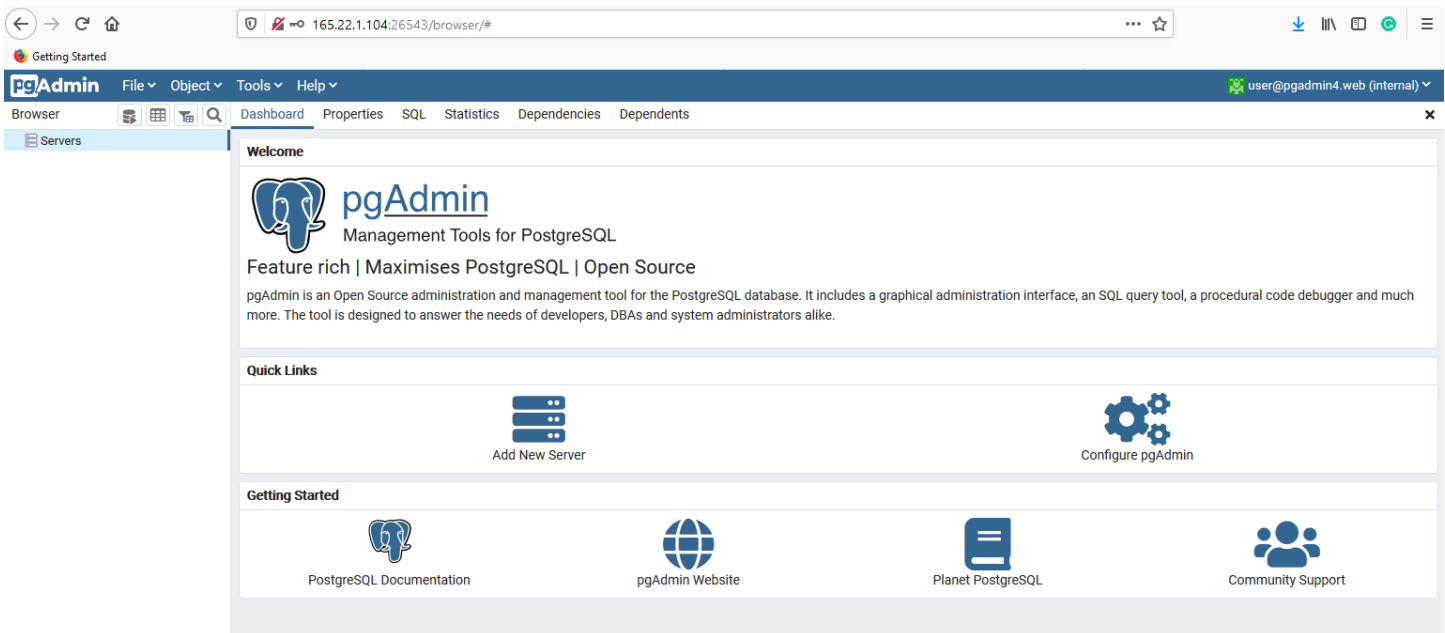
10. Accessing the Postgres Admin interface (pgAdmin4)

There is one last (more advanced) tool that you can use and comes pre-installed with the framework. You can use pgAdmin to connect to the database. On a web browser, go to:

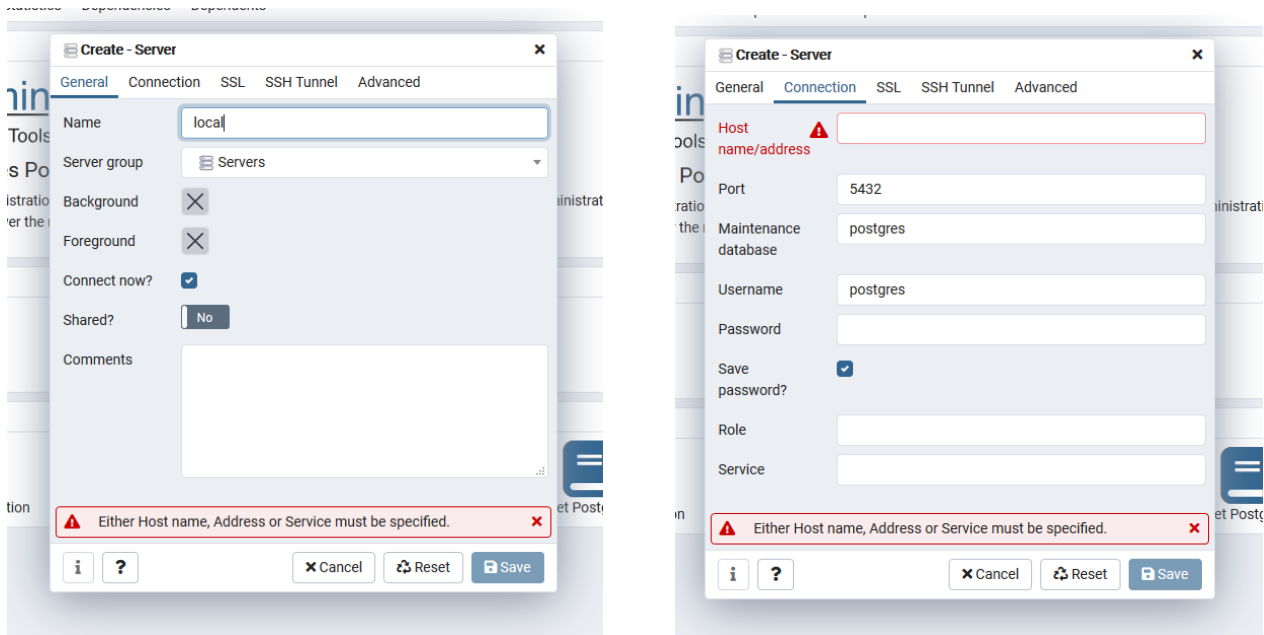
http://your_ip:26543

You can login using as username user@pgadmin4.web and as password the value of the variable `PGADMIN_DEFAULT_PASSWORD` in the env file.

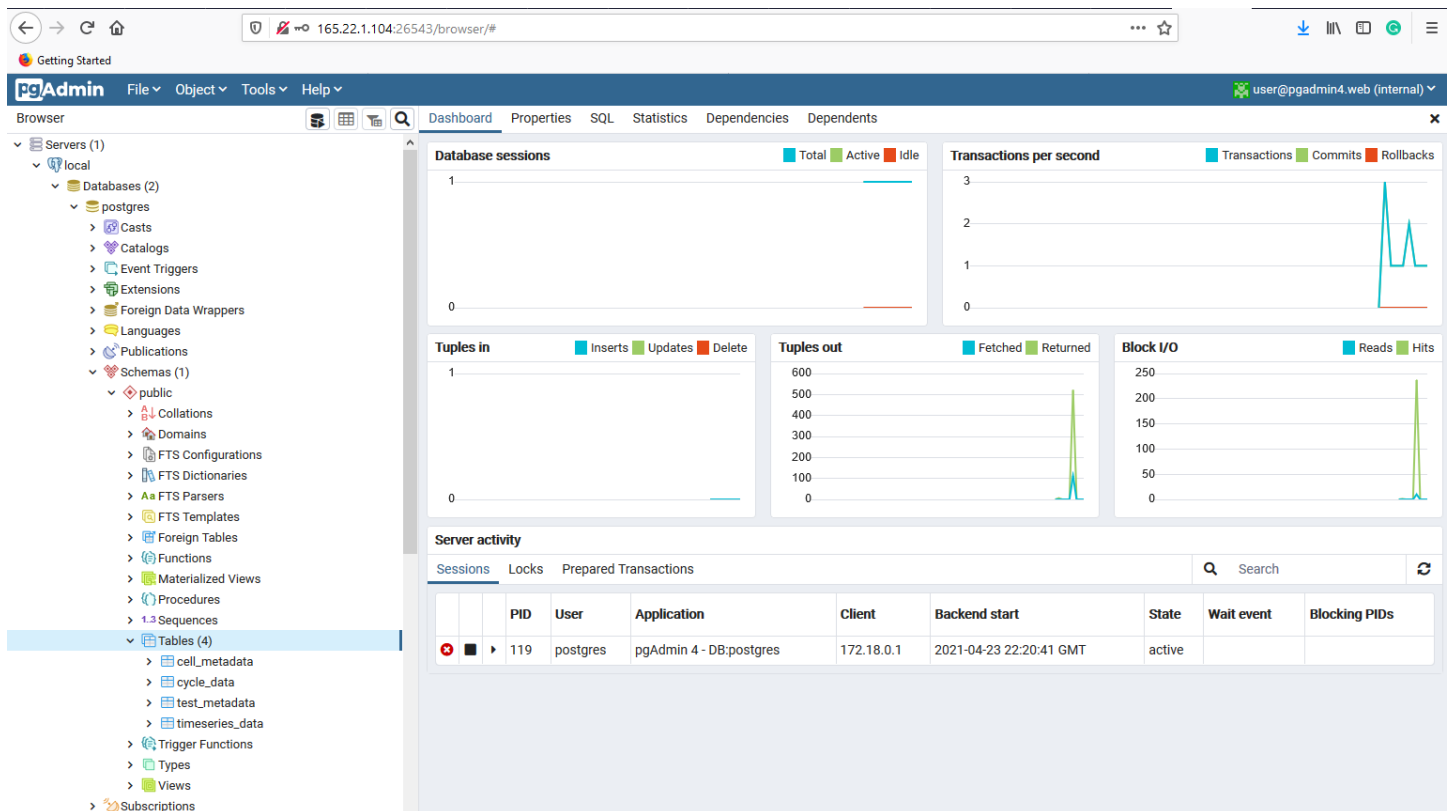
After you login, you can connect to the postgres database with the data you imported. Click on Add New Server.



In the windows that comes up enter local under Name, then click on the Connection tab



Under Host name/address enter the IP of your server. As username enter postgres. The password is the value of the variable POSTGRES_PASSWORD in the env file. Select Save Password. You should see a screen like the screen below and should be able to browse to the database and database tables.



You can use pgAdmin to customize the database schema for your specific applications. You can, for example, add a new column (like dv/dq) to the test and cycle data table, and write a script either in the data_import.py file or in Jupyter Notebook, to populate it with data. Even if you have never used SQL, you will find it intuitive and easy to learn. There are many SQL tutorials available on the web, but you can start from here <https://www.w3schools.com/sql/>

We hope that you found this guide useful. If you decide to give the software a try, please contact us at info@batteryarchive.org. We will be happy to help you install the system or customize it for your needs.

References

[1] De Angelis, V., Preger, Y., & Chalamala, B. R. (2021, March 13). Battery Lifecycle Framework: A Flexible Repository and Visualization Tool for Battery Data from Materials Development to Field Implementation.

<https://doi.org/10.1149/osf.io/h7c24>

[2] Preger, Y.; Barkholtz, H. M.; Fresquez, A.; Campbell, D. L.; Juba, B. W.; Romàn Kustas, J.; Ferreira, S. R.; and Chalamala, B. R. J. Electrochem. Soc. 2020, 167, 120532.

Notice

Copyright 2021 National Technology & Engineering Solutions of Sandia, LLC (NTESS). Under the terms of Contract DE-NA0003525 with NTESS, the U.S. Government retains certain rights in this software.