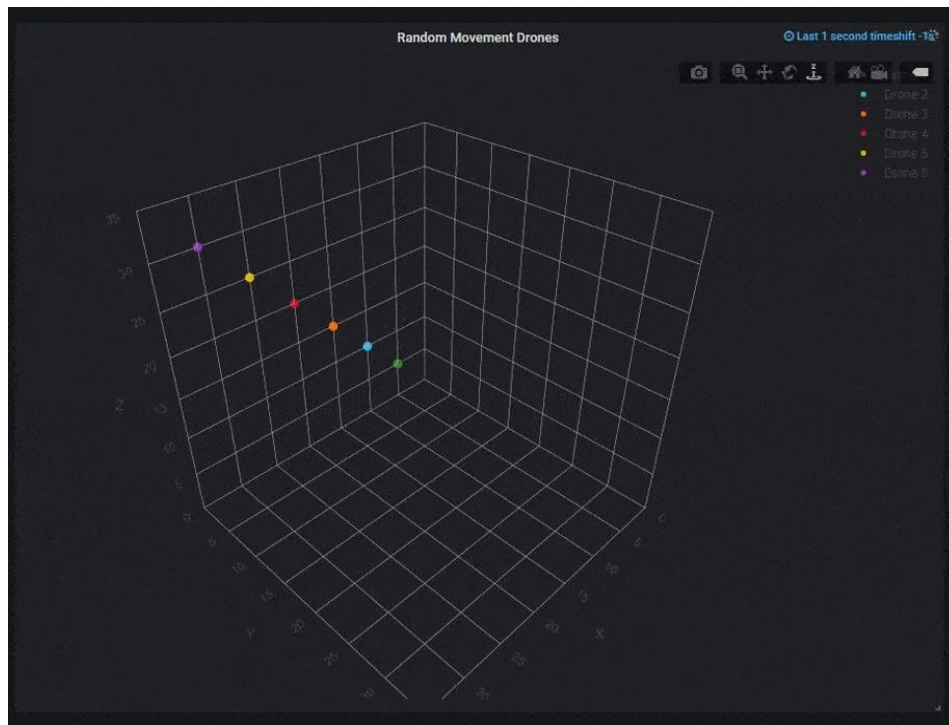


Configure Visualization Panel for Grafana/ Plotly 3D Scatterplot (Linux)

This panel will look something like this:



1. Enter Grafana (<http://localhost:3000>)
2. Install Plotly plugin

① Install the Panel

Use the grafana-cli tool to install Plotly from the commandline:

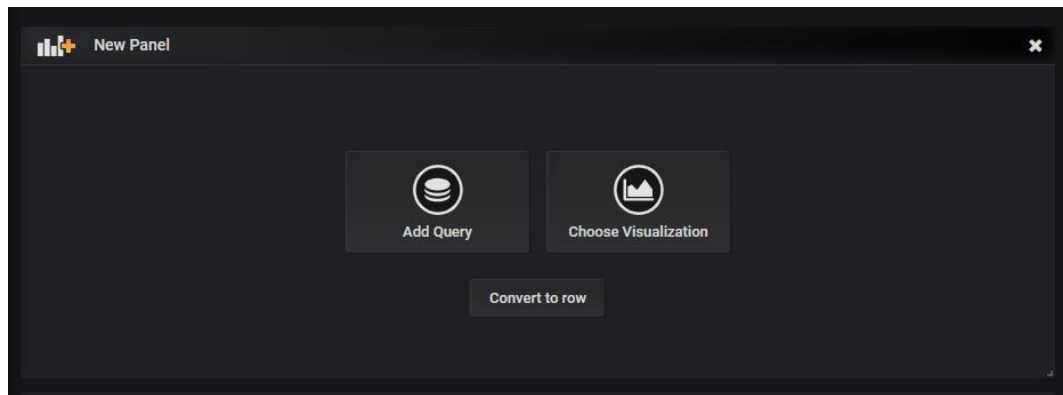
```
grafana-cli plugins install natel-plotly-panel
```

The plugin will be installed into your grafana plugins directory; the default is `/var/lib/grafana/plugins`. [More information on the cli tool.](#)

Note: Grafana 3.0 or greater is required to install and use plugins. [Download Grafana latest.](#)

Alternatively, you can manually [download the .zip file](#) and unpack it into your grafana plugins directory.

3. Go To Dashboard and Add Panel



- a. Click add Query or Choose Visualization and Follow the Remaining Steps

4. Configure Database Queries

- a. The Query Process starts with adding a data source in Grafana (assuming you haven't done this already)
- b. Set up the Data Source (InfluxDB)

A screenshot of the InfluxDB data source configuration page in Grafana. The page has a dark theme. At the top, there is a 'Name' field with the value 'Demo1' and a 'Default' toggle switch. Below this is the 'HTTP' section with a 'URL' field set to 'http://localhost:8086', an 'Access' dropdown set to 'Server (default)', and a 'Whitelisted Cookies' section with an 'Add Name' field and an 'Add' button. The 'Auth' section contains several toggle switches: 'Basic auth' (off), 'With Credentials' (off), 'TLS Client Auth' (off), 'With CA Cert' (off), 'Skip TLS Verify' (off), and 'Forward OAuth Identity' (off). The 'InfluxDB Details' section has a 'Database' field set to 'demo1', a 'User' field set to 'admin', a 'Password' field set to 'configured', and a 'reset' button. The 'HTTP Method' dropdown is set to 'GET'.

- i. Name it whatever you want, that's for your organization
- ii. Url is the port at which your datasource is, localhost:8086 is Influxdb's default
- iii. Choose the database that you are using
- iv. Put in your UserName and Password
- v. Test and Save

c. Writing the Query

The screenshot shows the InfluxDB query builder interface. It features a dark theme with a grid of buttons for constructing a query. The interface includes a dropdown menu at the top left labeled 'A'. The main area contains several rows of buttons for different query clauses: 'FROM', 'SELECT', 'GROUP BY', 'FORMAT AS', and 'ALIAS BY'. Each row has a button for the clause followed by input fields for values and a '+' button to add more items. The 'FROM' row has 'default' and 'm1' in the input fields. The 'SELECT' row has 'field (x)', 'last ()', and 'field (y)', 'last ()' in the input fields. The 'GROUP BY' row has 'time (\$_interval)' and 'fill (null)' in the input fields. The 'FORMAT AS' row has a dropdown menu set to 'Time series'. The 'ALIAS BY' row has a text input field with the placeholder 'Naming pattern'.

FROM	default	m1	WHERE	+	
SELECT	field (x)	last ()	+		
	field (y)	last ()	+		
	field (z)	last ()	+		
GROUP BY	time (\$_interval)	fill (null)	+		
FORMAT AS	Time series ▼				
ALIAS BY	Naming pattern				

- i. Pick the datasource that you just created
- ii. Above is a template for a query for a single point/device
- iii. x,y,z here are integer values and are used for plotting the point

Query InfluxDB-1

▼ A

FROM	default	m1	WHERE	+	
SELECT	field (x)	last ()	+		
	field (y)	last ()	+		
	field (z)	last ()	+		
GROUP BY	time (\$__interval)	fill (null)	+		
FORMAT AS	Time series ▼				
ALIAS BY	Naming pattern				

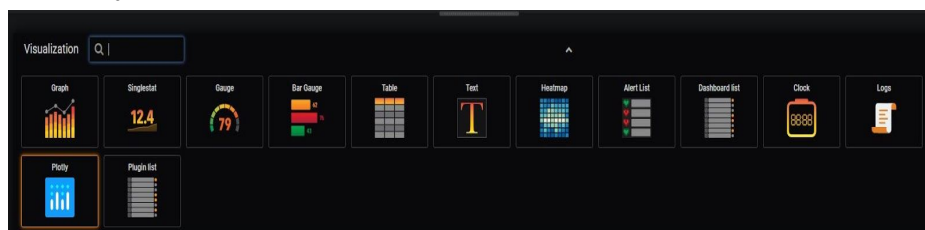
▶ B `SELECT last("x"), last("y"), last("z") FROM "m2" WHERE $timeFilter GROUP BY time($__interval) fill(null)`
 ▶ C `SELECT last("x"), last("y"), last("z") FROM "m3" WHERE $timeFilter GROUP BY time($__interval) fill(null)`
 ▶ D `SELECT last("x"), last("y"), last("z") FROM "m4" WHERE $timeFilter GROUP BY time($__interval) fill(null)`
 ▶ E `SELECT last("x"), last("y"), last("z") FROM "m5" WHERE $timeFilter GROUP BY time($__interval) fill(null)`
 ▶ F `SELECT last("x"), last("y"), last("z") FROM "m6" WHERE $timeFilter GROUP BY time($__interval) fill(null)`

Min time interval ⓘ 0 Relative time 1s Time shift 1s Hide time info ☐

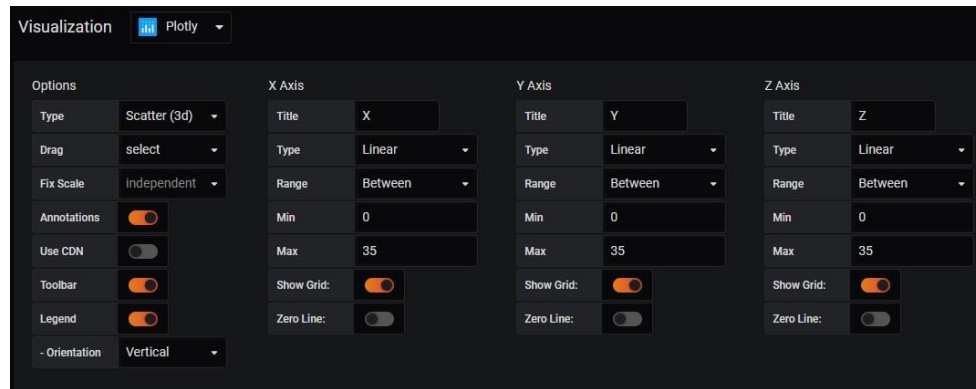
- iv. For this example, there are 6 devices each with its own query on a different measurement
- v. The six measurements are m1-m6 and you can duplicate the first query and just change the measurement name to be more efficient

5. Configure Visualization

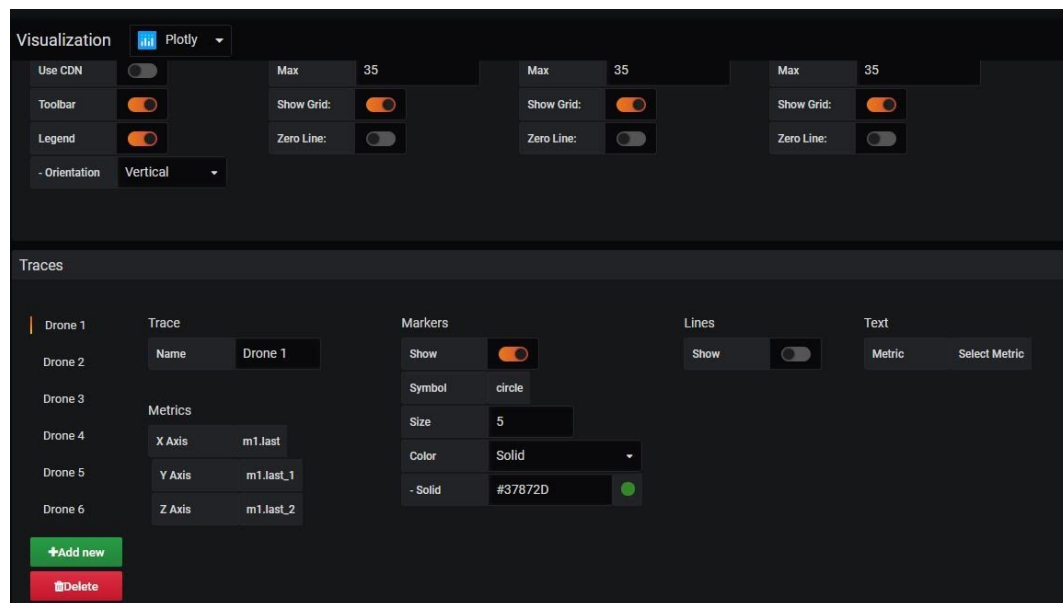
- a. Pick Plotly



- b. Configure the Plot

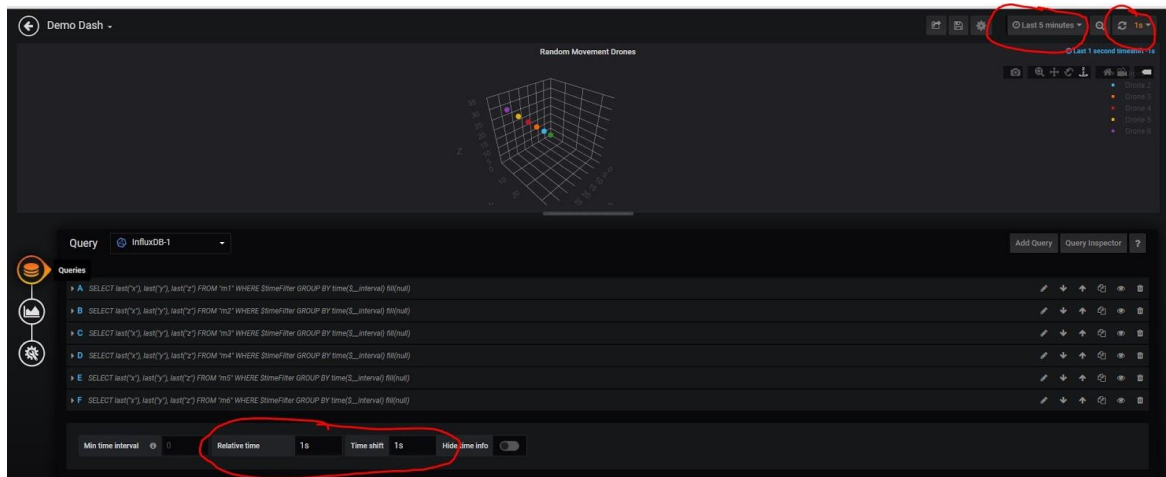


- i. Select Scatter(3D)
 - ii. You can set the range of the axes, in this example every one is set between 0 & 35
- c. Create the Traces



- i. Above is an example of a Trace for the plot
- ii. Name is irrelevant, except for you and the legend
- iii. x,y,z should match the values that you have for each measurement
- iv. For each other trace created, make sure that the metrics match the measurement that its representing
 1. i.e. Trace 2 should have m2 as the metric
 2. The metrics would be m2.last, etc

6. Set Refresh Time



- You can change the relative time of various panels to be different than the relative time of the whole dashboard
- For this script it is important that the relative time is 1s and the time shift is 1s for the display to work properly

7. Run Script