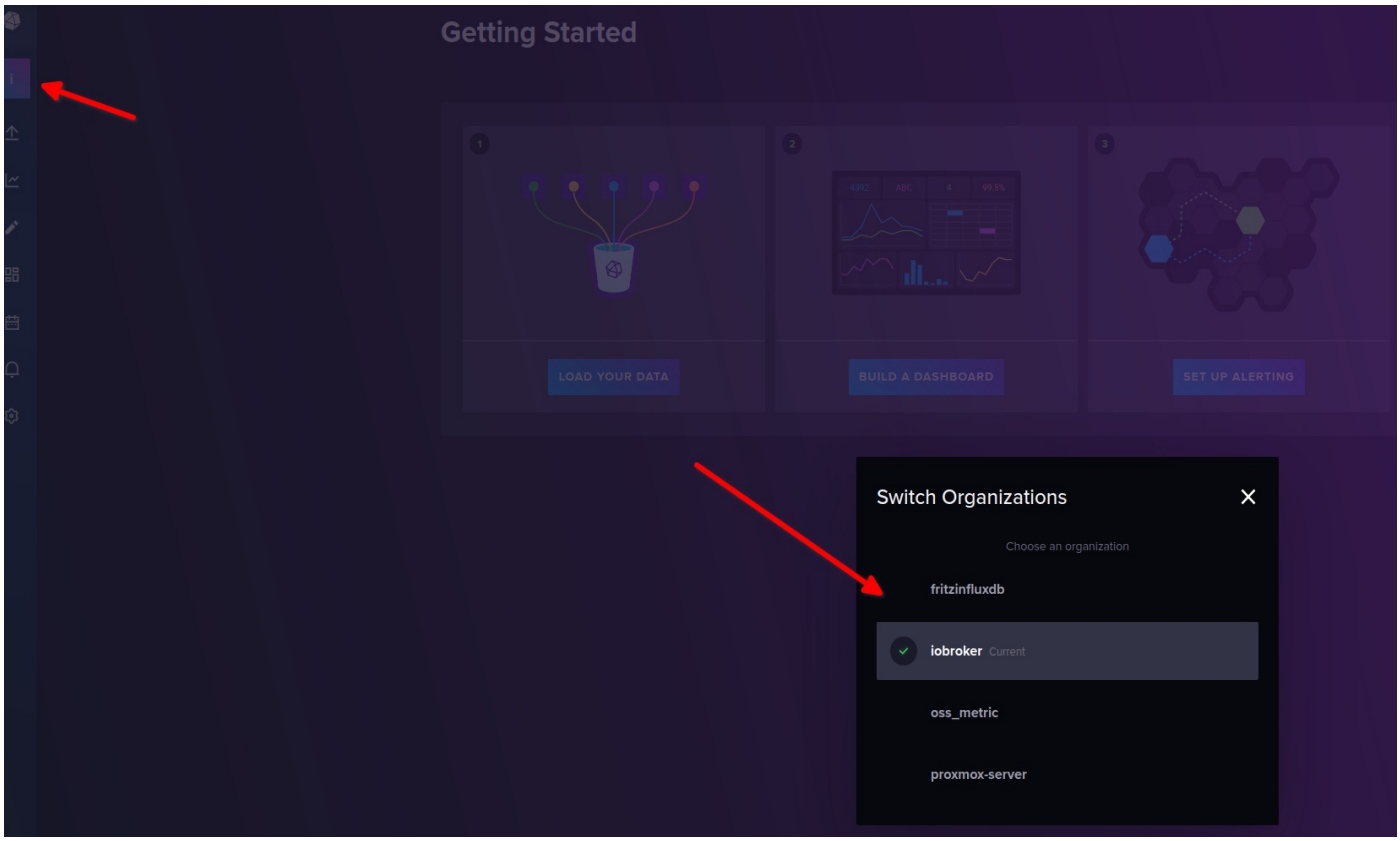
influxDB Metriken

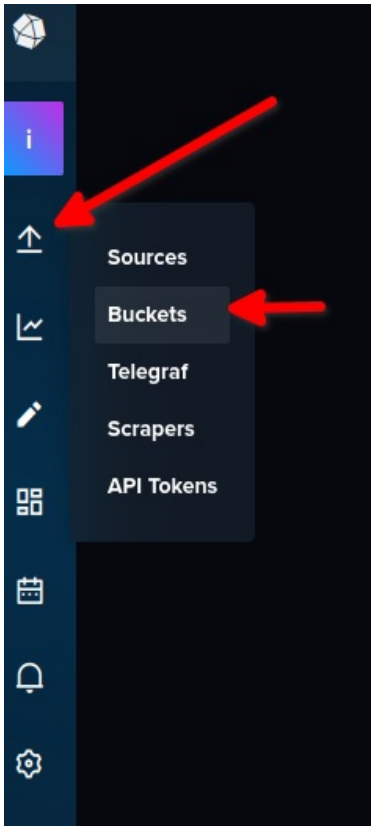
dieser Weg ist für alle, die schon eine fertig eingerichtete `influxDB` Installation haben und nicht von vorn beginnen möchten.

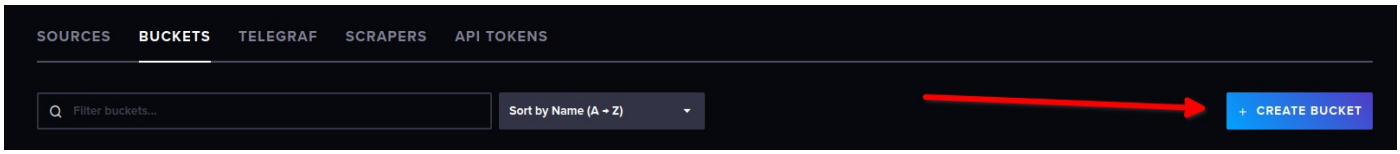
Organisation auswählen

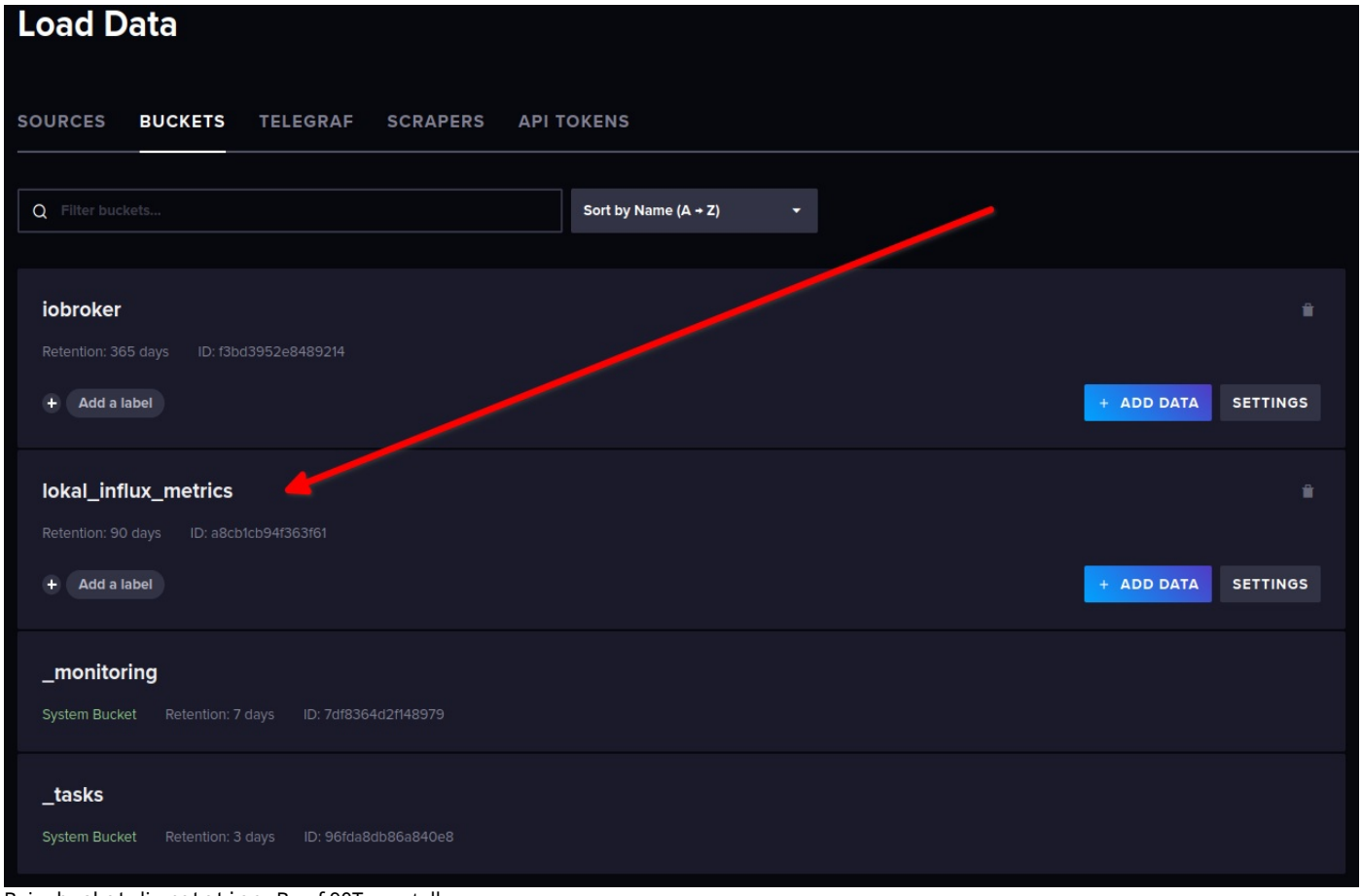


Oft gibt es nur eine, nicht wie bei mir viele, da ich mit unterschiedlichen Dingen Spiele

Bucket erstellen

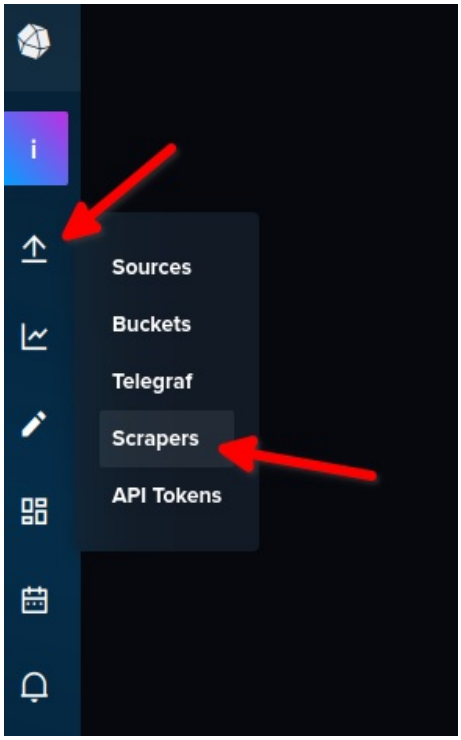


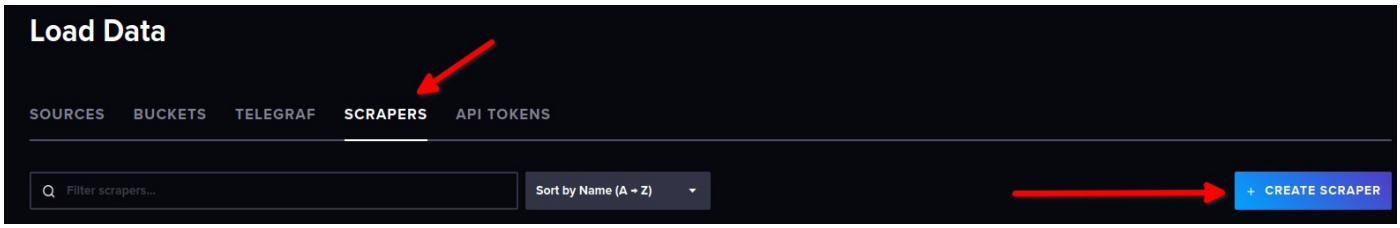


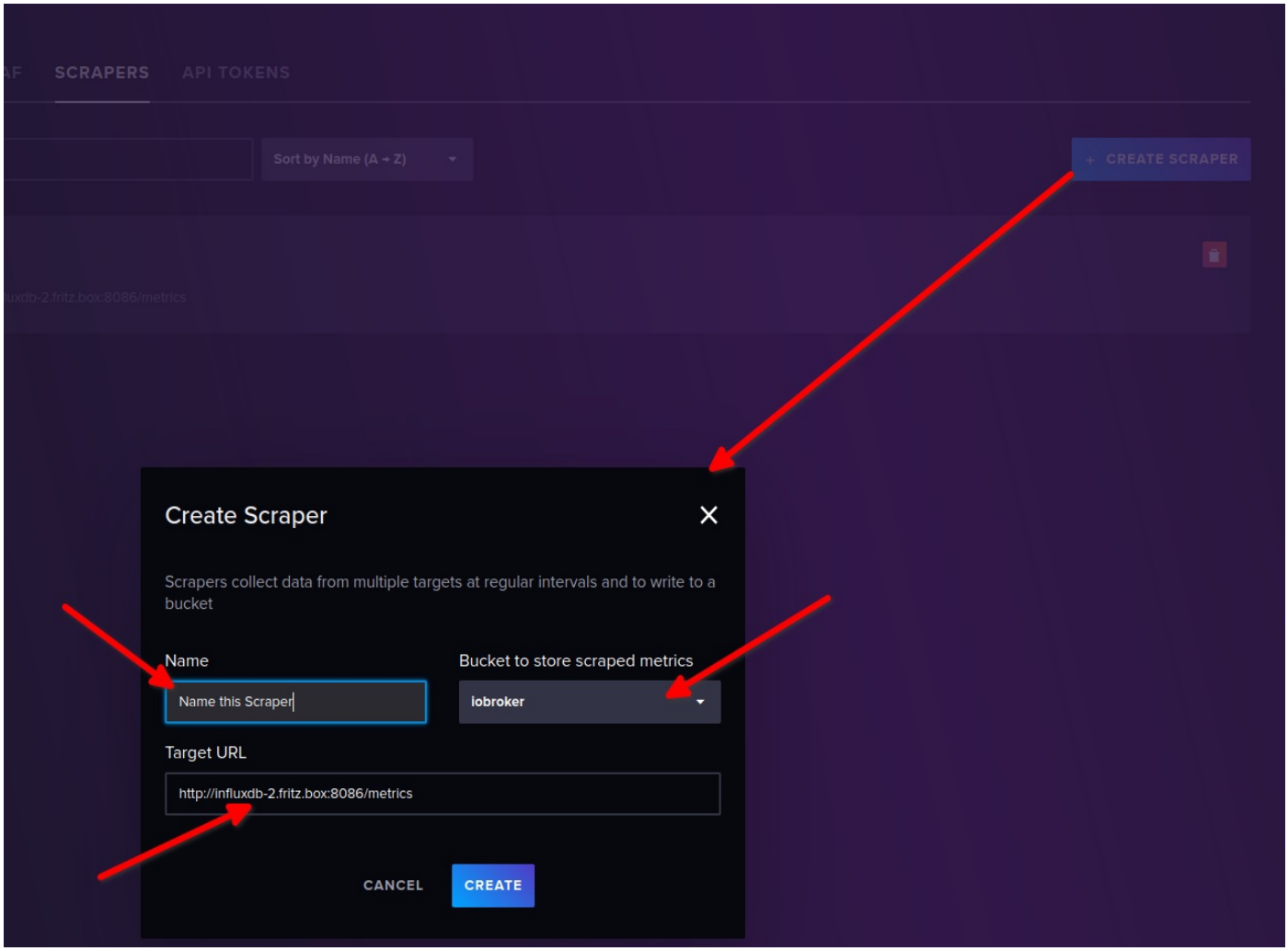


Beim `bucket` die `retetion` zB auf 90Tage stellen

Scraper erstellen

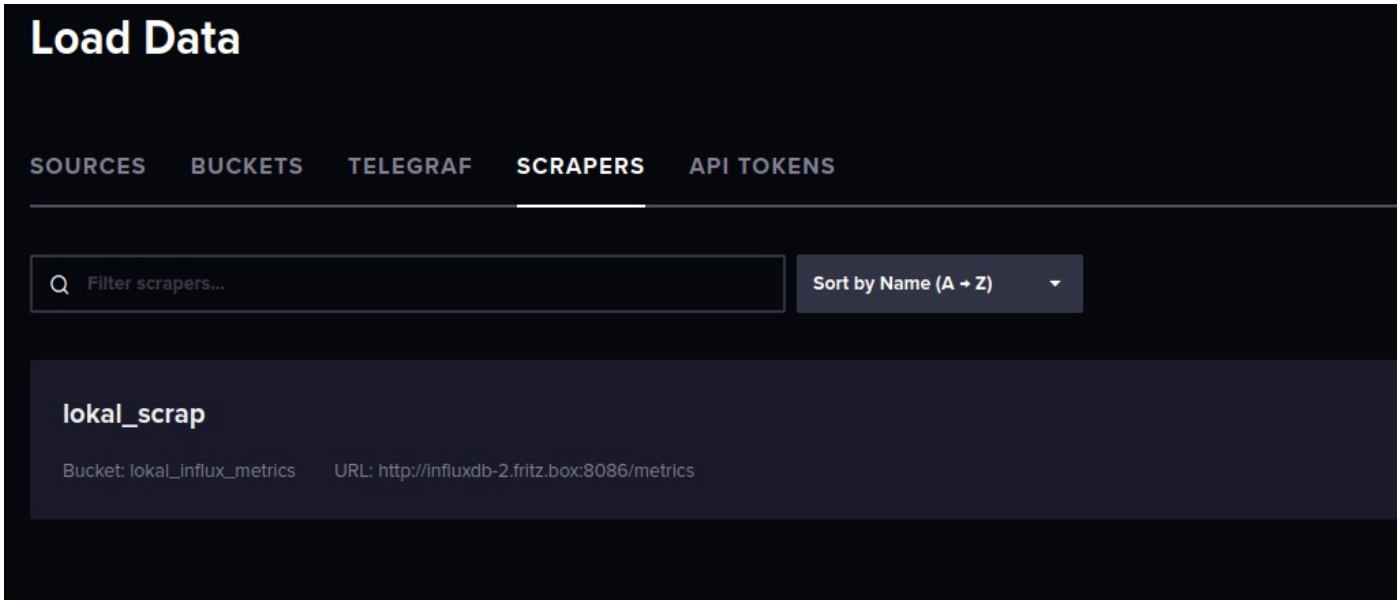

Sources
Buckets
Telegraf
Scrapers
API Tokens


Load Data
SOURCES
BUCKETS
TELEGRAF
SCRAPERS
API TOKENS
Filter scrapers...
Sort by Name (A → Z)
+ CREATE SCRAPER

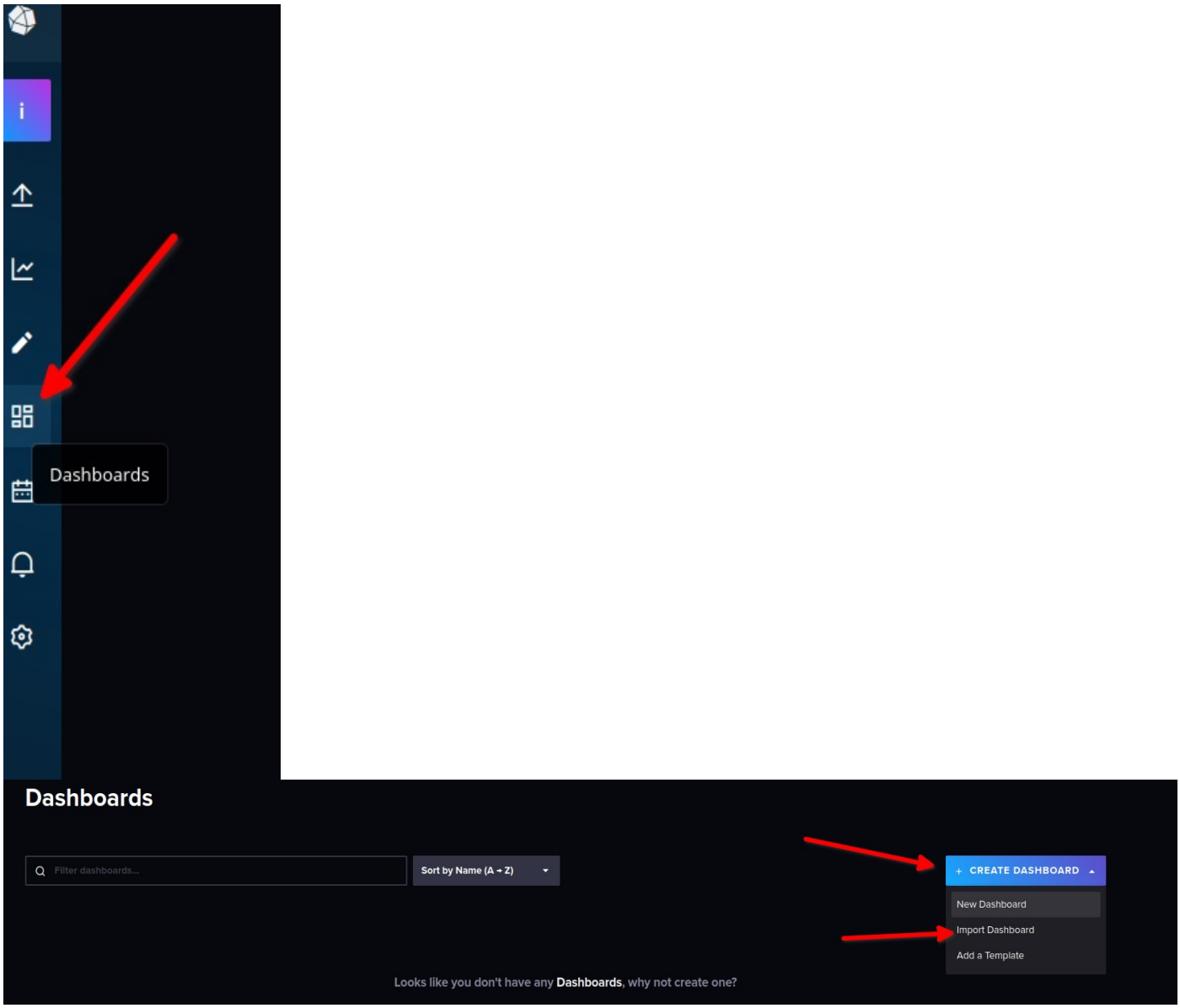


* Name = Name des Scraper * Bucket = das gerade zuvor angelegte `bucket` auswählen * Target URL = über Hostname, IP oder localhost und den passenden Port eintragen <- bei Docker hat es bei mir nur über `localhost` funktioniert

Das sollte dann unter Scrapers auftauchen



Dashboard anlegen



Ich habe hier mal ein funktionierendes Dashboard als `*.json` an gehangen, `` [{ "apiVersion":"influxdata.com/v2alpha1", "kind":"Dashboard", "metadata":{ "name":"practical-gagarin-877001" }, "spec":{ "charts":[{ "height":2, "kind":"Markdown", "name":"Name this Cell", "note":"#### This Dashboard gives you an overview of some of the metrics that are available from the Local Metrics endpoint located at/metrics`. Check out our documentation page for configuring Scrapers if you don't see any data below.“, "staticLegend":{

```
      },
      "width":12
   },
   {
      "colors":[
         {
            "id":"base",
            "name":"laser",
            "type":"text",
            "hex":"#00C9FF"
         }
      ],
      "decimalPlaces":2,
      "height":2,
      "kind":"Single_Stat",
      "name":"Uptime",
      "note":"This shows the amount of time your current InfluxDB 2 instance has been running, in hours. Keep it up
      "queries":[
         {
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{

      },
      "suffix":" hrs",
      "width":3,
      "yPos":2
   },
   {
      "colors":[
         {
            "id":"base",
            "name":"laser",
            "type":"text",
            "hex":"#00C9FF"
         }
      ],
      "decimalPlaces":2,
      "height":1,
      "kind":"Single_Stat",
      "name":"",
      "note":"An Organization is a workspace where you and your team can organize your data, Dashboards, Tasks, and
      "queries":[
         {
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{

      },
      "suffix":" Orgs",
      "width":3,
      "yPos":4
   },
   {
      "colors":[
         {
            "id":"base",
            "name":"laser",
            "type":"text",
            "hex":"#00C9FF"
         }
      ],
      "decimalPlaces":2,
      "height":1,
      "kind":"Single_Stat",
      "name":"",
      "note":"InfluxDB 2 can create and store your Telegraf agent configs. Telegraf is the world's best data collec
      "queries":[
         {
```

```
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{

      },
      "suffix":" Telegrafs",
      "width":3,
      "yPos":5
   },
   {
      "axes":[
         {
            "base":"10",
            "name":"x",
            "scale":"linear"
         },
         {
            "base":"10",
            "name":"y",
            "scale":"linear"
         }
      ],
      "geom":"line",
      "height":4,
      "kind":"Xy",
      "name":"Local Object Store IO",
      "queries":[
         {
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{

      },
      "width":12,
      "yPos":6
   },
   {
      "axes":[
         {
            "base":"10",
            "name":"x",
            "scale":"linear"
         },
         {
            "base":"10",
            "name":"y",
            "scale":"linear"
         }
      ],
      "geom":"line",
      "height":4,
      "kind":"Xy",
      "name":"Query Requests",
      "queries":[
         {
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{

      },
      "width":12,
      "yPos":10
   },
   {
      "axes":[
         {
            "base":"10",
            "name":"x",
```

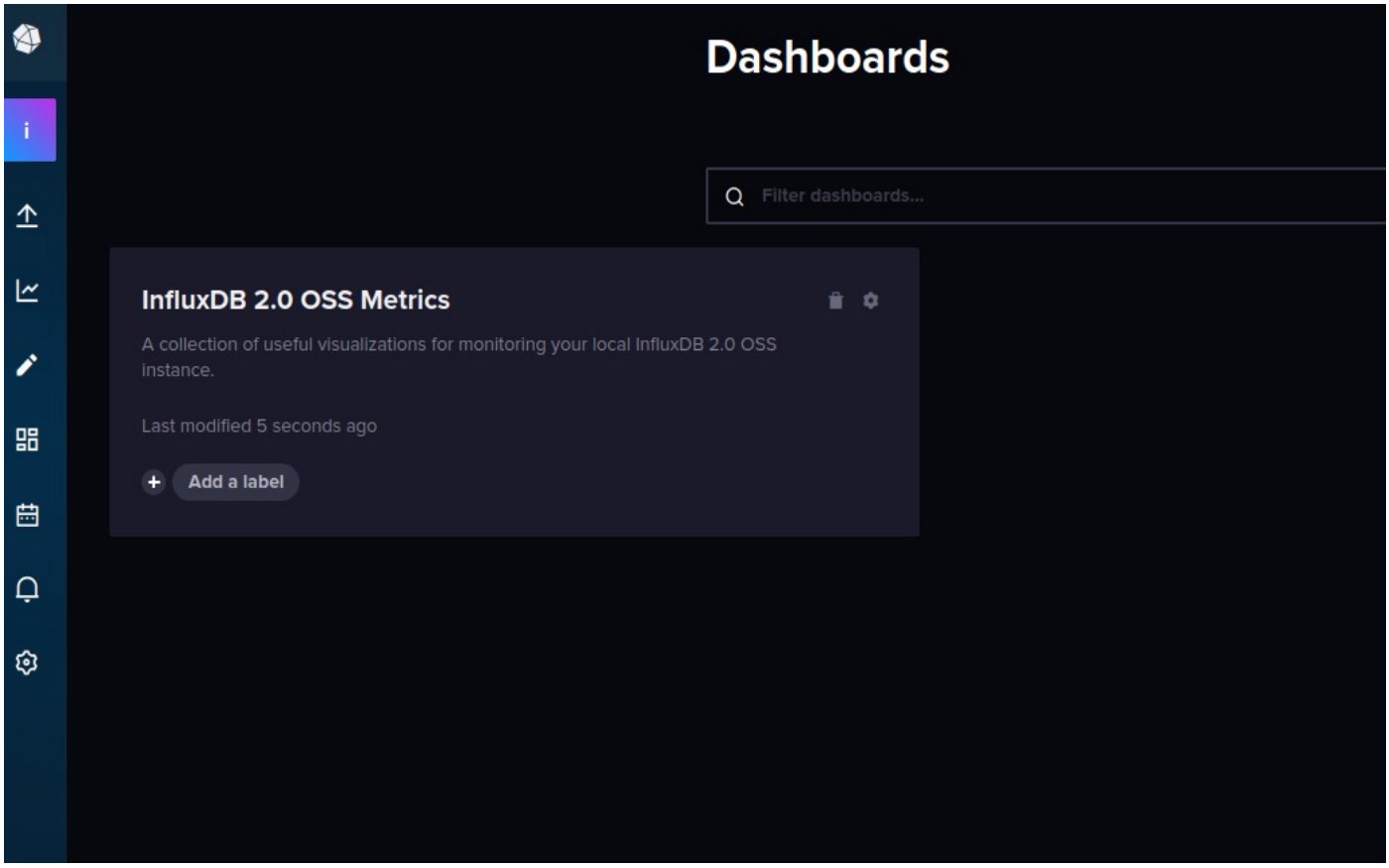
```
            "scale":"linear"
         },
         {
            "base":"10",
            "name":"y",
            "scale":"linear"
         }
      ],
      "geom":"line",
      "height":3,
      "kind":"Xy",
      "name":"Memory Allocations (Bytes)",
      "queries":[
         {
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{

      },
      "width":4,
      "yPos":14
   },
   {
      "colors":[
         {
            "id":"base",
            "name":"laser",
            "type":"text",
            "hex":"#00C9FF"
         }
      ],
      "fieldOptions":[
         {
            "visible":true
         },
         {
            "displayName":"result",
            "fieldName":"result",
            "visible":true
         },
         {
            "displayName":"table",
            "fieldName":"table",
            "visible":true
         },
         {
            "displayName":"Architecture",
            "fieldName":"Architecture",
            "visible":true
         },
         {
            "displayName":"Build Date",
            "fieldName":"Build Date",
            "visible":true
         },
         {
            "displayName":"Github Commit",
            "fieldName":"Github Commit",
            "visible":true
         },
         {
            "displayName":"CPUs",
            "fieldName":"CPUs",
            "visible":true
         },
         {
            "displayName":"OS",
            "fieldName":"OS",
            "visible":true
         },
```

```
      {
         "displayName":"Version",
         "fieldName":"Version",
         "visible":true
      }
   ],
   "height":2,
   "kind":"Table",
   "name":"Instance Info",
   "note":"This cell gives you information about your running instance of InfluxDB 2, but you probably already k
   "queries":[
      {
         "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
      }
   ],
   "staticLegend":{

   },
   "tableOptions":{
      "verticalTimeAxis":true
   },
   "timeFormat":"YYYY-MM-DD HH:mm:ss",
   "width":9,
   "xPos":3,
   "yPos":2
},
{
   "colors":[
      {
         "id":"base",
         "name":"laser",
         "type":"text",
         "hex":"#00C9FF"
      }
   ],
   "decimalPlaces":2,
   "height":1,
   "kind":"Single_Stat",
   "name":"",
   "note":"This lets you know how many users have access to your InfluxDB 2 instance. You can add new users from
   "queries":[
      {
         "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
      }
   ],
   "staticLegend":{

   },
   "suffix":" Users",
   "width":3,
   "xPos":3,
   "yPos":4
},
{
   "colors":[
      {
         "id":"base",
         "name":"laser",
         "type":"text",
         "hex":"#00C9FF"
      }
   ],
   "decimalPlaces":2,
   "height":1,
   "kind":"Single_Stat",
   "name":"",
   "note":"Dashboards are a great way to group together and view data in InfluxDB 2. You can create new ones fro
   "queries":[
      {
         "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
      }
```

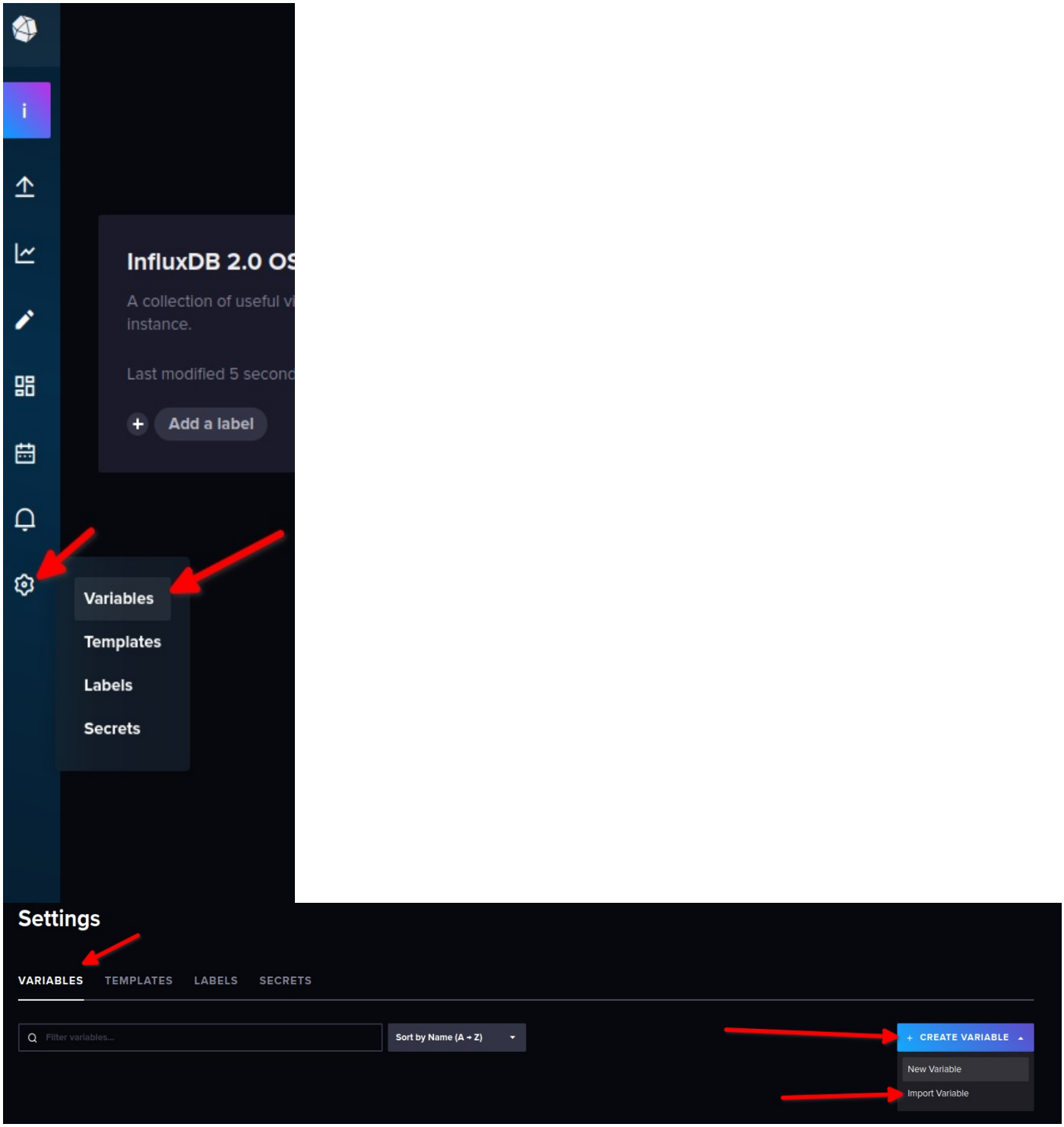
```
      ],
      "staticLegend":{

      },
      "suffix":" Dashboards",
      "width":3,
      "xPos":3,
      "yPos":5
   },
   {
      "axes":[
         {
            "base":"10",
            "name":"x",
            "scale":"linear"
         },
         {
            "base":"10",
            "name":"y",
            "scale":"linear",
            "suffix":"%"
         }
      ],
      "geom":"line",
      "height":3,
      "kind":"Xy",
      "name":"Memory Usage (%)",
      "queries":[
         {
            "query":"bytes_used = from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop
         }
      ],
      "staticLegend":{

      },
      "width":4,
      "xPos":4,
      "yPos":14
   },
   {
      "colors":[
         {
            "id":"base",
            "name":"laser",
            "type":"text",
            "hex":"#00C9FF"
         }
      ],
      "decimalPlaces":2,
      "height":1,
      "kind":"Single_Stat",
      "name":"",
      "note":"A Bucket is where you store your time series data and each one has a set retention policy. You create
      "queries":[
         {
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{

      },
      "suffix":" Buckets",
      "width":3,
      "xPos":6,
      "yPos":4
   },
   {
      "colors":[
         {
            "id":"base",
            "name":"laser",
```

```
            "type":"text",
            "hex":"#00C9FF"
         }
      ],
      "decimalPlaces":2,
      "height":1,
      "kind":"Single_Stat",
      "name":"",
      "note":"InfluxDB 2 can natively scrape data from Prometheus endpoints, including its own metrics. For more in
      "queries":[
         {
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{

      },
      "suffix":" Scrapers",
      "width":3,
      "xPos":6,
      "yPos":5
   },
   {
      "axes":[
         {
            "base":"10",
            "name":"x",
            "scale":"linear"
         },
         {
            "base":"10",
            "name":"y",
            "scale":"linear"
         }
      ],
      "geom":"line",
      "height":3,
      "kind":"Xy",
      "name":"Memory Allocs & Frees (Bytes)",
      "queries":[
         {
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{

      },
      "width":4,
      "xPos":8,
      "yPos":14
   },
   {
      "colors":[
         {
            "id":"base",
            "name":"laser",
            "type":"text",
            "hex":"#00C9FF"
         }
      ],
      "decimalPlaces":2,
      "height":1,
      "kind":"Single_Stat",
      "name":"",
      "note":"A Token allows you to access your instance from an external client such as a command line or a client
      "queries":[
         {
            "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
         }
      ],
      "staticLegend":{
```

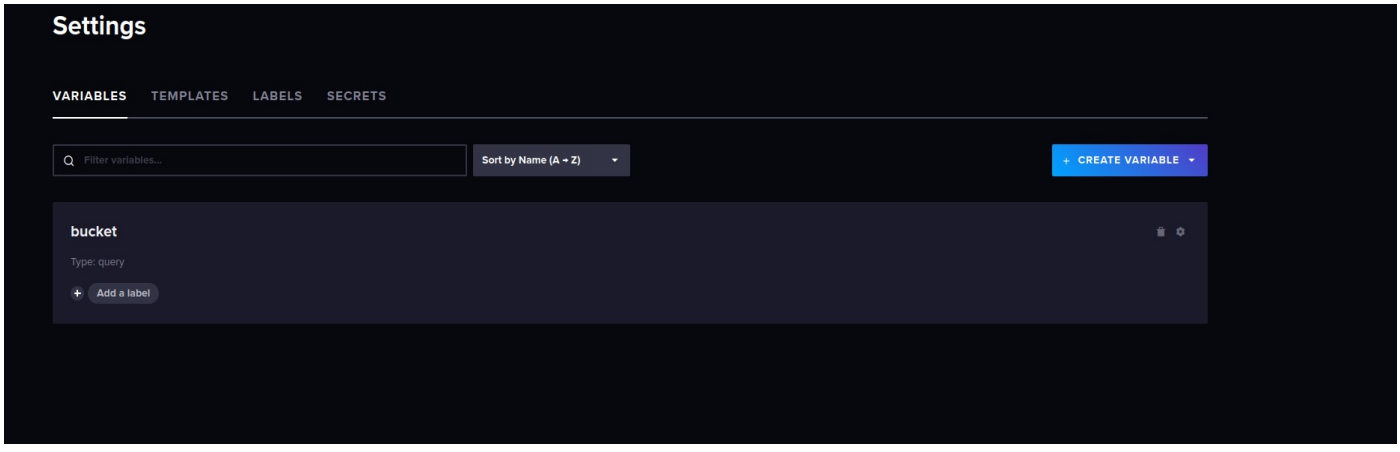
```
            },
            "suffix":" Tokens",
            "width":3,
            "xPos":9,
            "yPos":4
         },
         {
            "colors":[
               {
                  "id":"base",
                  "name":"laser",
                  "type":"text",
                  "hex":"#00C9FF"
               }
            ],
            "decimalPlaces":2,
            "height":1,
            "kind":"Single_Stat",
            "name":"",
            "note":"Tasks allow you to automate Flux queries for things like data rollups and enrichment. You can create
            "queries":[
               {
                  "query":"from(bucket: v.bucket)\n  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)\n  |> filte
               }
            ],
            "staticLegend":{

            },
            "suffix":" Tasks",
            "width":3,
            "xPos":9,
            "yPos":5
         }
      ],
      "description":"A collection of useful visualizations for monitoring your local InfluxDB 2.0 OSS instance.",
      "name":"InfluxDB 2.0 OSS Metrics"
   }


}]```
```



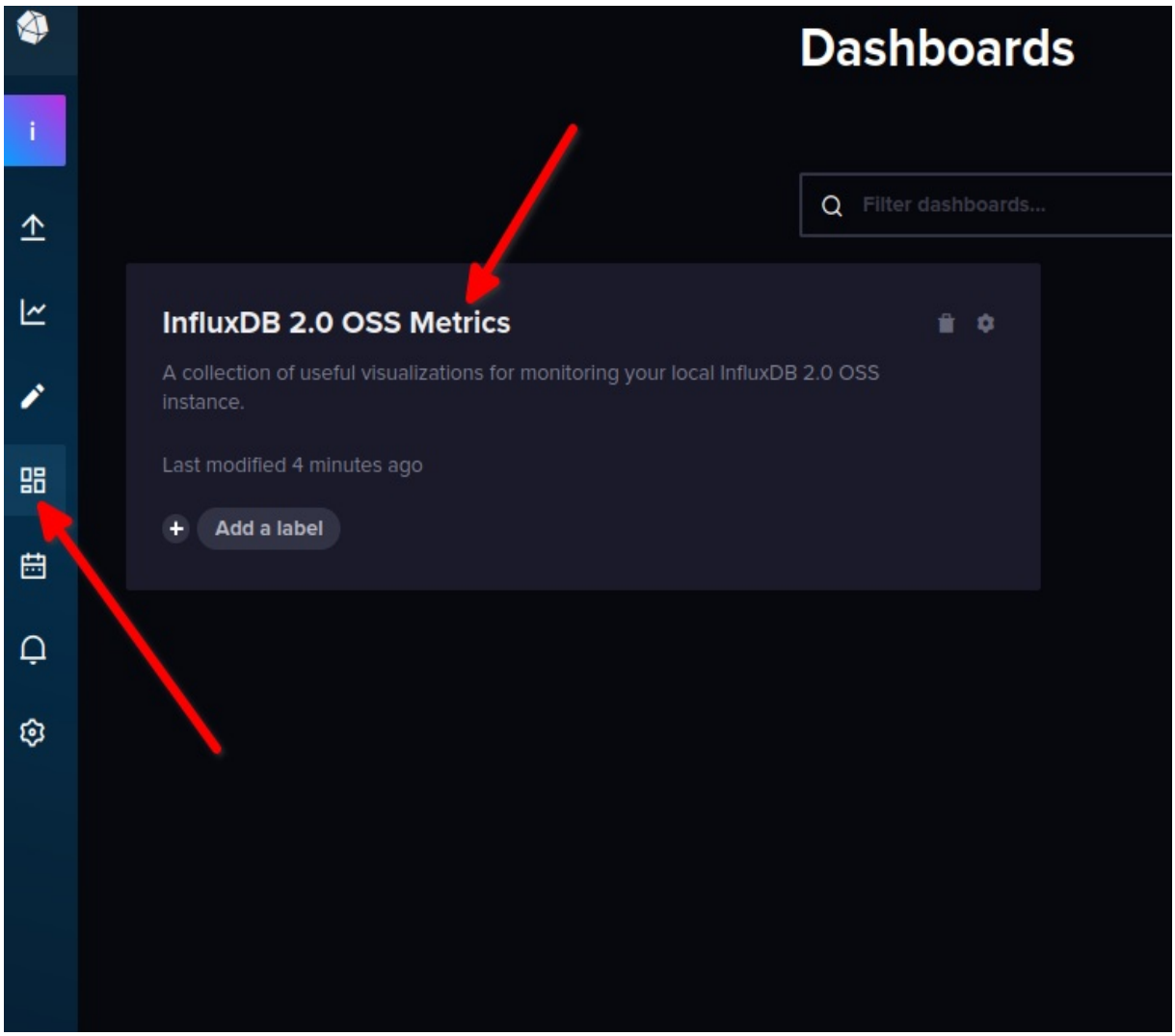
Variable erstellen

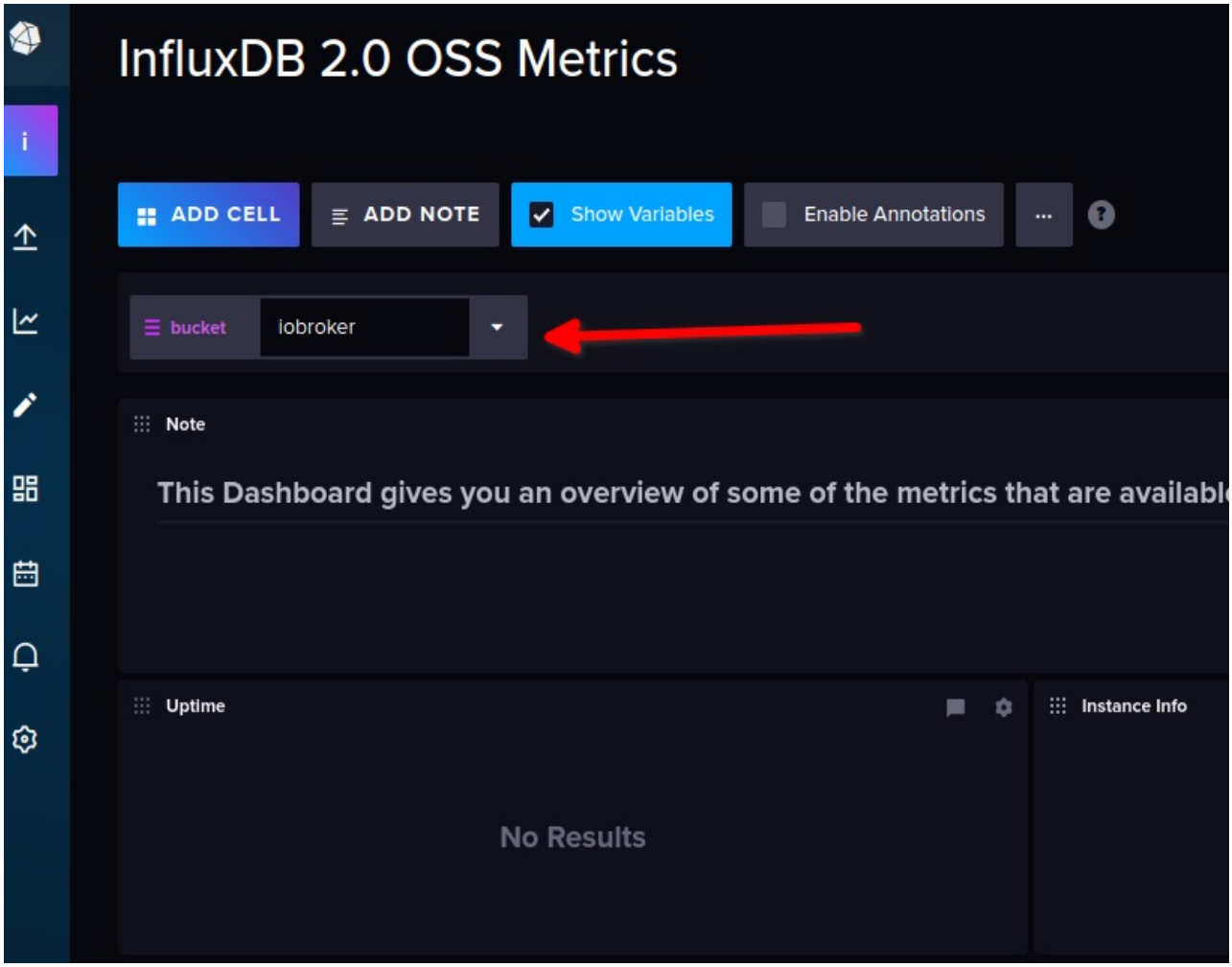


Ach hier habe ich mal die fertige `*.json` an gehangen, `[{ "apiVersion":"influxdata.com/v2alpha1", "kind":"Variable", "metadata":{ "name":"sad-tesla-477001" }, "spec":{ "language":"flux", "name":"bucket", "query":"buckets()\n |> filter(fn: (r) => r.name !~ /^_/)\n |> rename(columns: {name: \"_value\"})\n |> keep(columns: [\"_value\"])", "type":"query" } }]`



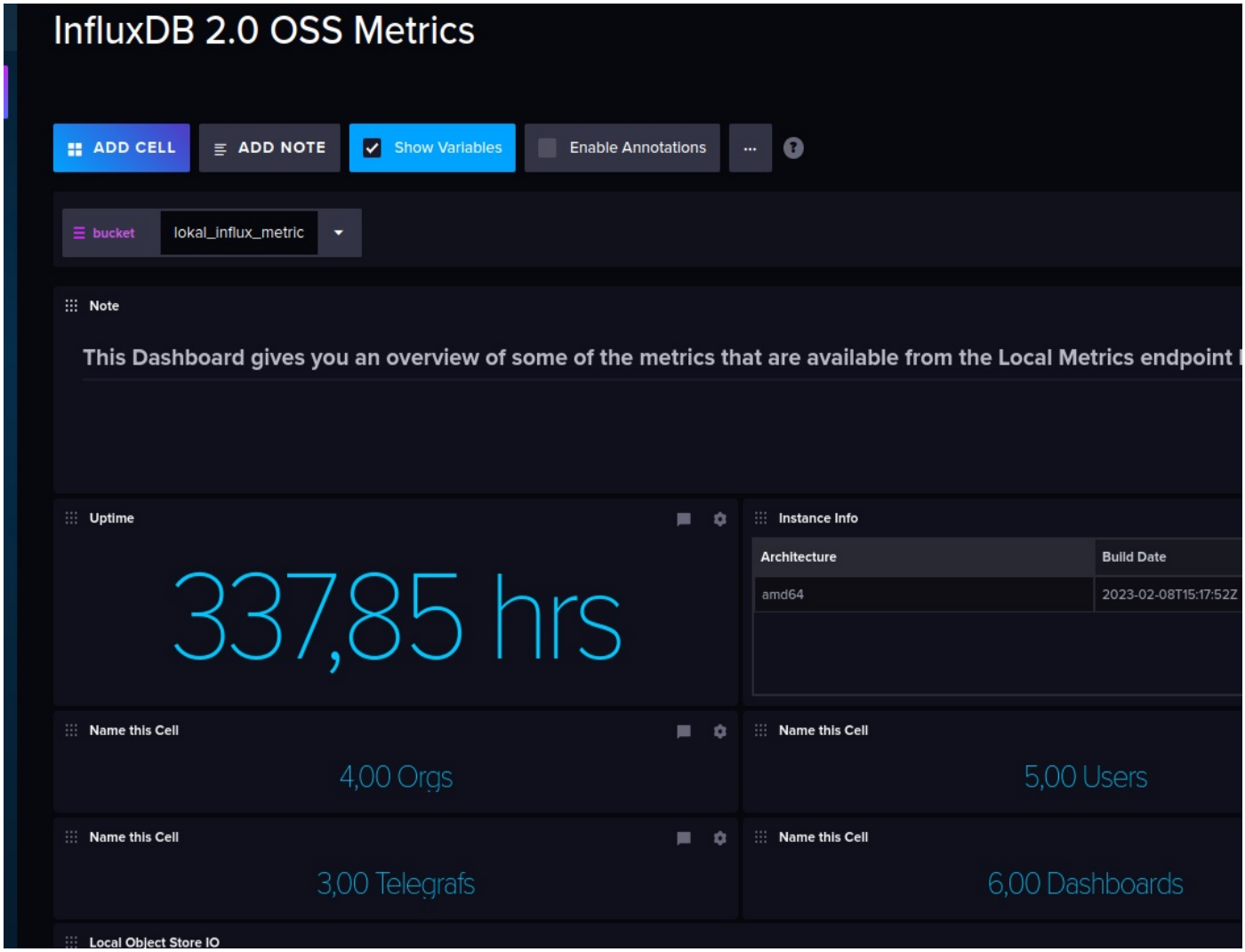
Das Dashboard aufrufen





Unter Bucket das aus wählen welches wir oben erstellt haben

Ergebnis


InfluxDB 2.0 OSS Metrics
ADD CELL
ADD NOTE
Show Variables
Enable Annotations
bucket
lokal_influx_metric
Note
This Dashboard gives you an overview of some of the metrics that are available from the Local Metrics endpoint
Uptime
337,85 hrs
Instance Info
Architecture
Build Date
amd64
2023-02-08T15:17:52Z
Name this Cell
4,00 Orgs
Name this Cell
5,00 Users
Name this Cell
3,00 Telegrafs
Name this Cell
6,00 Dashboards
Local Object Store IO