Tutorial

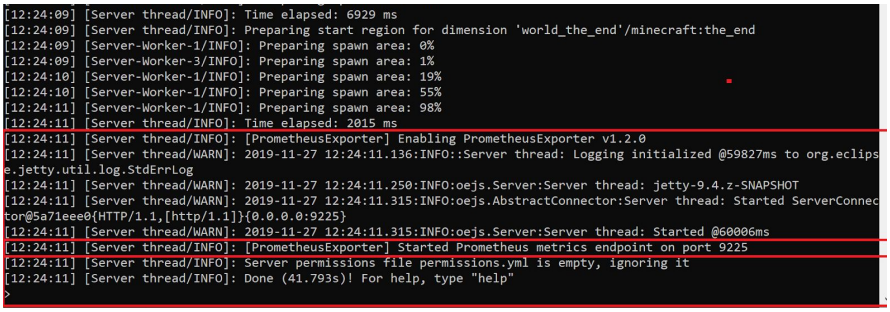
Application Monitoring with Prometheus and Grafana

In the tutorial below you will create a Minecraft server and connect some pre-made Grafana dashboards to monitor its status. You will then create a single Grafana panel. You will take two screenshots at specified locations and turn them in when you complete the tutorial.
After completing the tutorial you will complete the lab that follows.

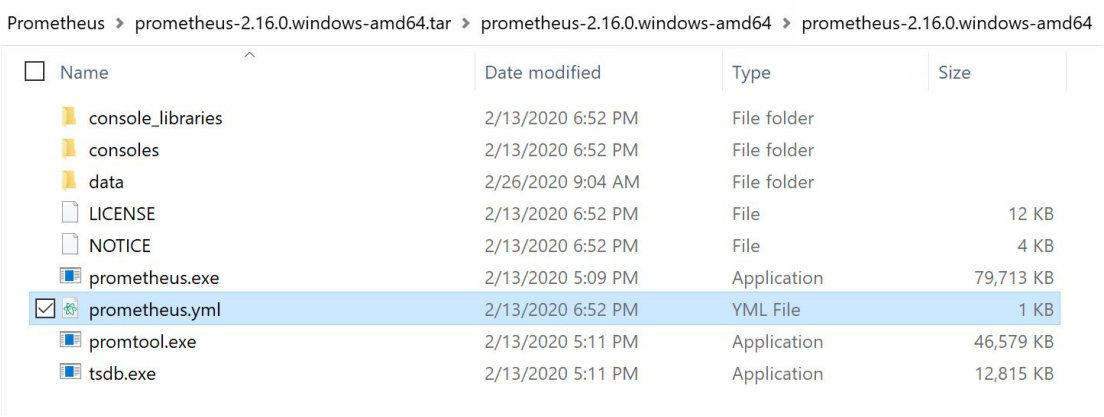
Connecting a Minecraft Server to Prometheus

In the Pre-class assignment, you installed an exporter onto the Minecraft server by placing the minecraft-prometheus-exporter-2.0.0.jar into the minecraft-server's plugins folder. This exporter takes the metrics Minecraft is already creating and formats it in a way that Prometheus can understand. The exporter then exposes the data to a predetermined port. For this exporter, the data is exposed on port 9225.

1. Make sure that you have set up your Minecraft server according to the pre-class assignment.
2. Open your Minecraft server folder
3. **Run the Minecraft server** using the given start script
4. Verify that the exporter was successfully installed.
    a. If it was installed successfully, your terminal should look like the screenshot below



5. **Navigate to the directory that contains the prometheus configuration file also known as the prometheus.yml file**
    a. **For Mac users:** Navigate to the directory where you placed the downloaded prometheus.yml during the pre-class assignment



6. **Open the prometheus.yml** file
    a. **For Mac users:** Open the prometheus.yml you **set to be the prometheus config.file in step 1.V** of the pre-class assignment

The Prometheus.yml file contains all the configuration for Prometheus. By configuring the Prometheus.yml file, you can tell Prometheus when, how, and where to scrape for data. We are going to configure it to scrape for data from our Minecraft server.

7. Scroll to the bottom of the file **to scrape_config:**
8. **Add a new job name, and target** for the Minecraft server
    a. Since the exporter is exposing the data on port 9225, we need to tell Prometheus to look for data on localhost:9225
    b. Copy and paste the scrape_config text below **over the old scrape config** in your prometheus.yml

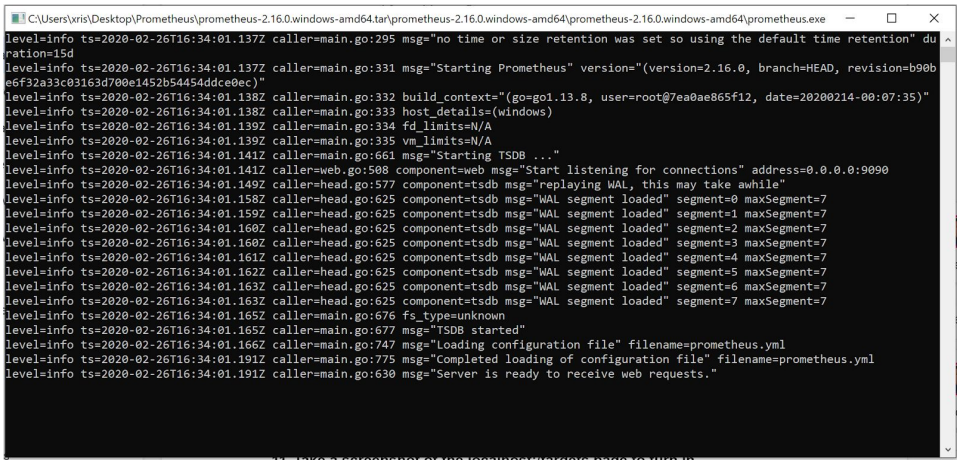
```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'
    static_configs:
    - targets: ['localhost:9090']

  - job_name: 'minecraft'
    static_configs:
    - targets: ['localhost:9225']
```

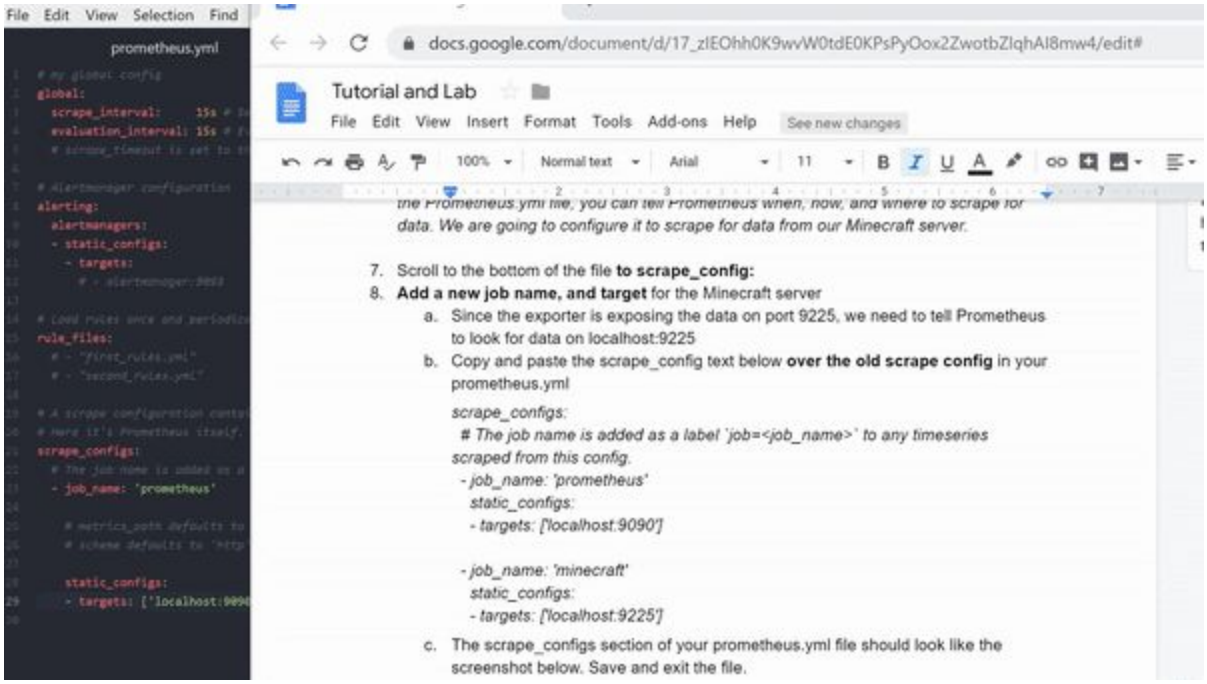
    c. The scrape_configs section of your prometheus.yml file should look like the screenshot below. **Save and exit the file**.

```
21 scrape_configs:
22   # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
23   - job_name: 'prometheus'
24     static_configs:
25     - targets: ['localhost:9090']
26
27   - job_name: 'minecraft'
28     static_configs:
29     - targets: ['localhost:9225']
30
```

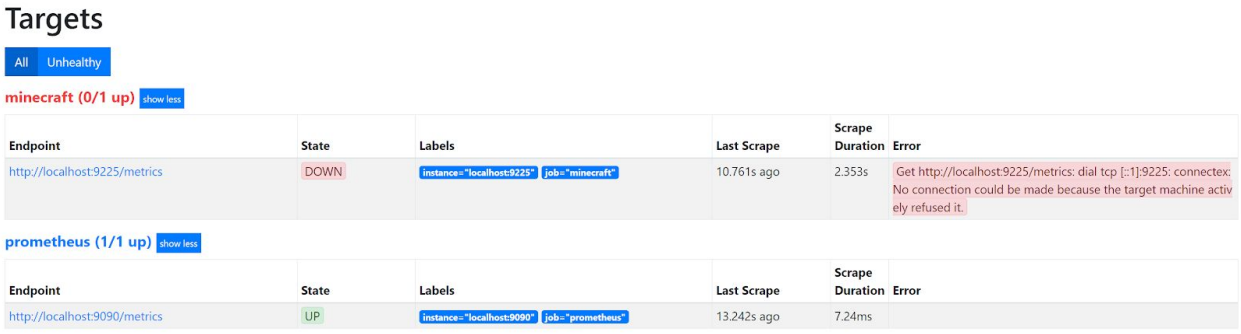
9. **If Prometheus was already running, stop it.**
    a. You can do this by exiting out of the prometheus terminal that is shown below



10. **Run Prometheus**
    a. use the following command to run prometheus
        i. Mac: **prometheus --config.file=<directory for your config file>/prometheus.yml**
        ii. Windows: Run the prometheus.exe file



11. Verify that the Minecraft server data **will be** collected by navigating to: http://localhost:9090/targets. It should look like the screenshot below.

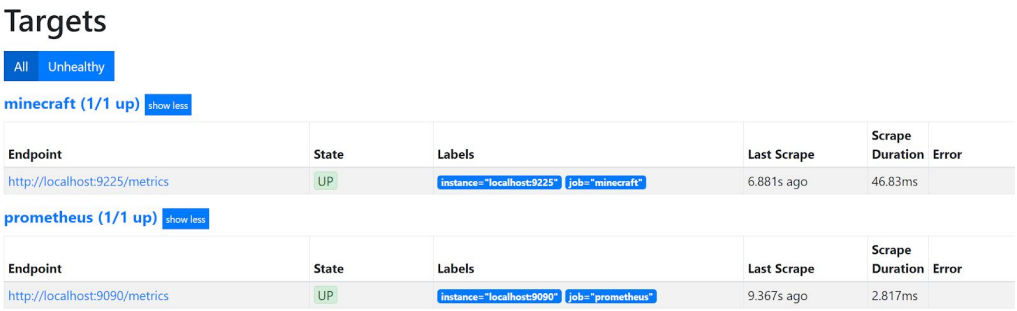


Connecting a Minecraft Server to Grafana

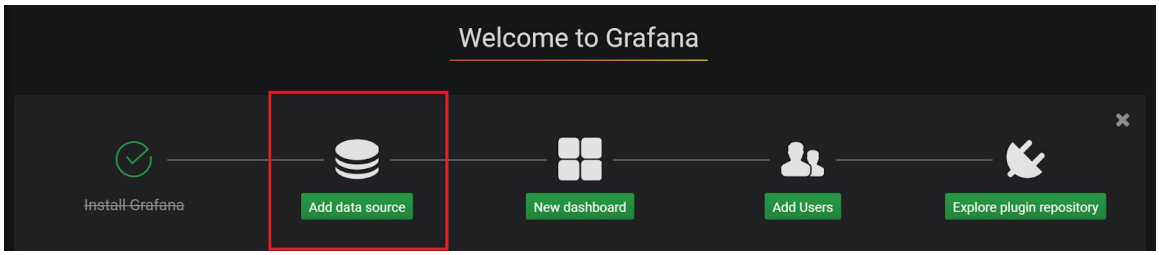
In order for Grafana to use data from Prometheus, Grafana needs to know where the Prometheus server is running. We will now create a data source in Grafana and connect it to our Prometheus server.

1. **Make sure Grafana is running**
    a. **Mac:** brew services start grafana

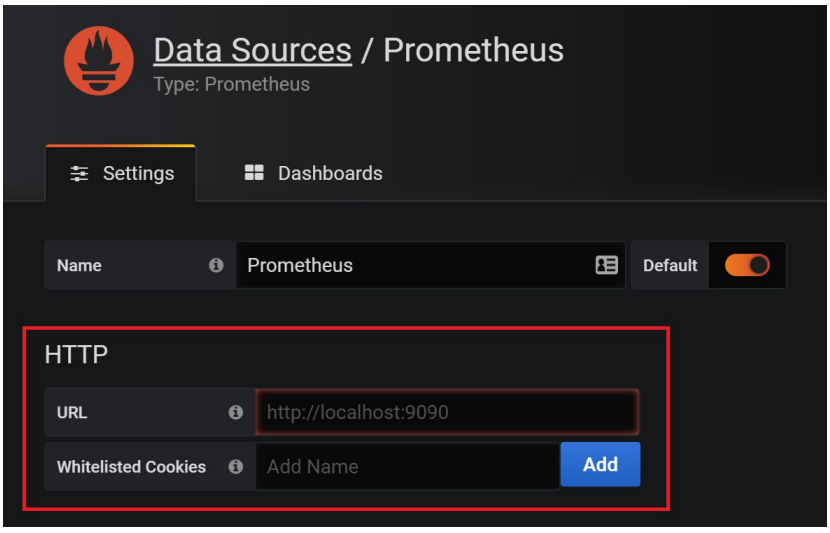
   b. **Windows:** execute the grafana-server.exe located in the grafana directory under the bin folder
2. **Make sure your minecraft server is running.**
   a. **Navigate to your minecraft server folder**
   b. Start the minecraft server by **executing the start.bat, start.command, or start.sh file** that you downloaded in the pre-class assignment
3. Verify that prometheus is scraping data from minecraft by navigating to http://localhost:9090/targets. Your screen should look like the screenshot below



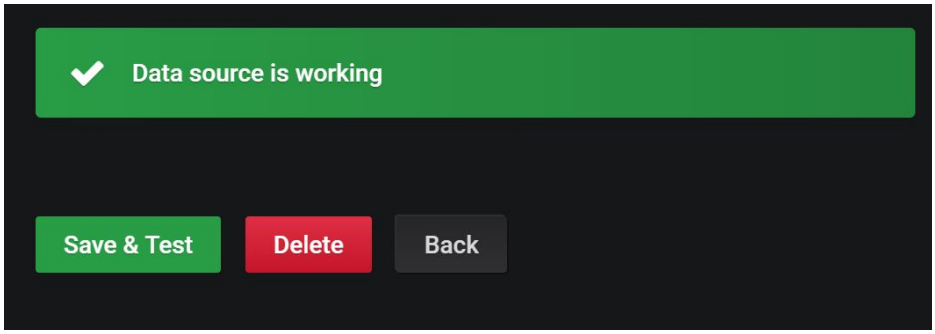
4. **Open up Grafana** at http://localhost:3000 *(this is the default port number)*
5. **Click -> Add a data source**



6. Select the Prometheus time series database
7. **Enter in your Prometheus URL**. The default URL is : http://localhost:9090
   a. This is **NOT** the URL where you are exposing your Minecraft data.



8. Save and Test
    a. If you see what is shown in the screenshot below, you are good to continue.

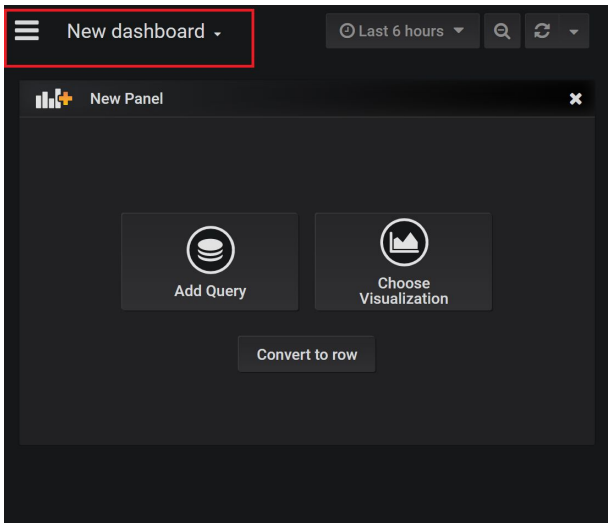


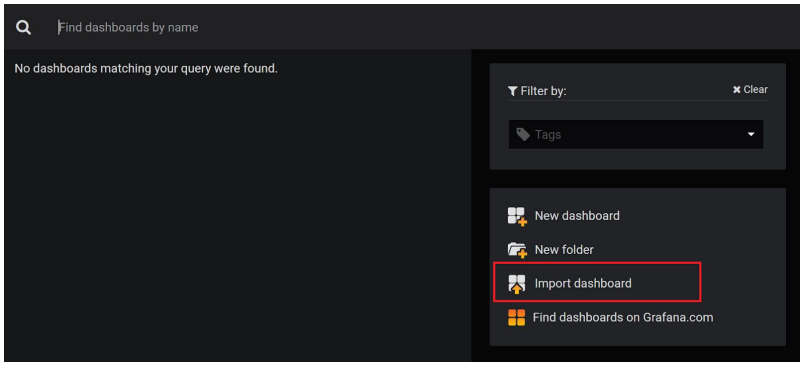
Import a New Dashboard for Minecraft

A Grafana dashboard is how Grafana displays the data. We are going to import a pre-made dashboard.

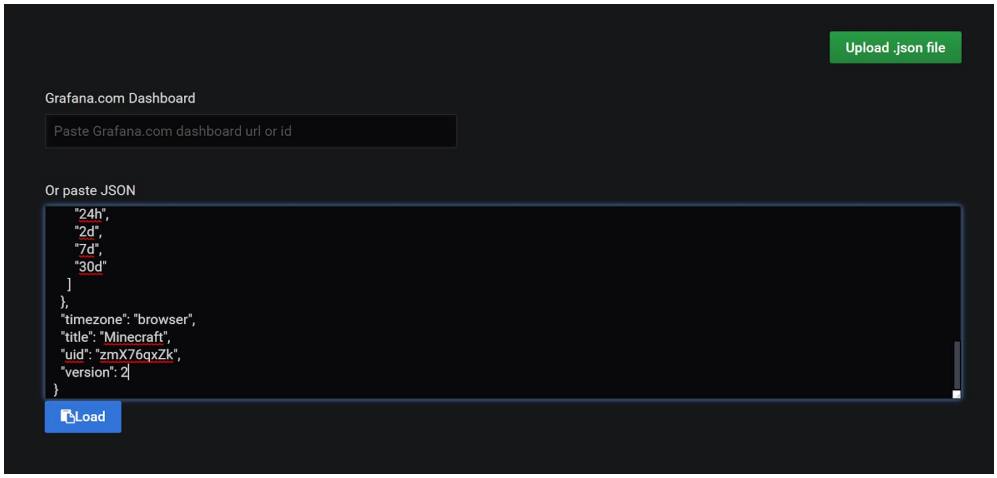
1. **Create a New dashboard** by clicking “New dashboard”

2. **Open the dashboard menu** by clicking the top left menu button and **select Import Dashboard**





3. **Copy and Paste in the JSON text** found in this file: https://drive.google.com/file/d/1MlHd9b5RCwo7kIdPT1SCEAXDbDiYWycE/view?usp=sharing



4. Click -> Load
5. Click -> Import
6. Your dashboard should now look like the screenshot below:

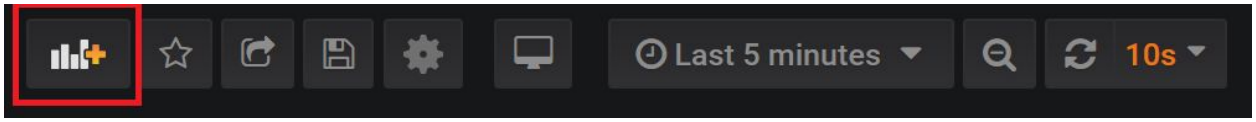
7. This Grafana dashboard is now showing data that is being generated by your Minecraft server. Not a lot is happening on this dashboard. That is because not a lot is changing or happening on your Minecraft server.
    a. *(optional)* If you have a Minecraft account, you can log in to your localhost server and interact with the server to get more interesting data
8. Compare the Grafana displays to the data being scraped from the Minecraft server at: http://localhost:9225/metrics

Create a New Entities Dashboard Panel for Minecraft

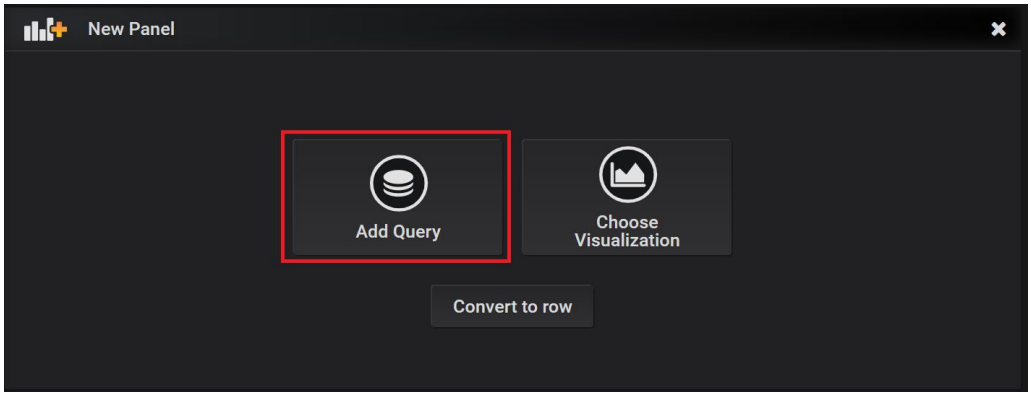
You will now create your own Grafana dashboard. This dashboard will display the total number of Minecraft entities on your Minecraft server, and the total number of living Minecraft entities on

your server. The data is already being collected by Prometheus and can be seen at http://localhost:9225/metrics as mc_entities_total and mc_living_entities_total.

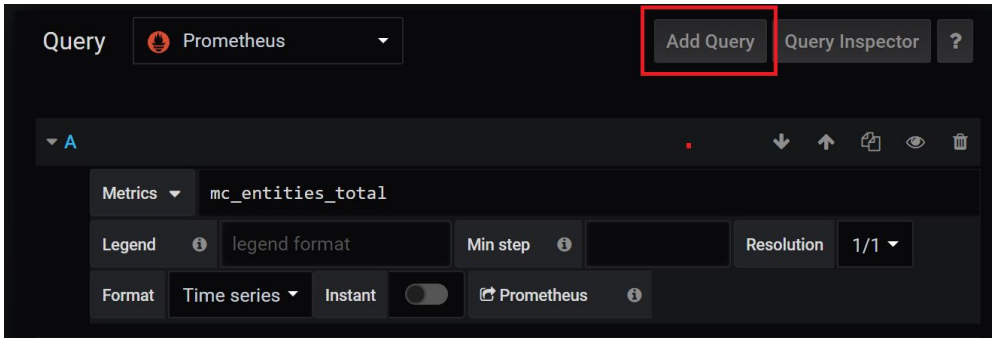
1. Click on the **Create New Panel** Button



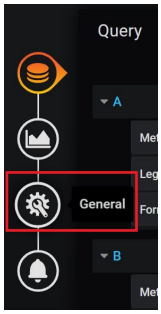
2. Click **Add Query**



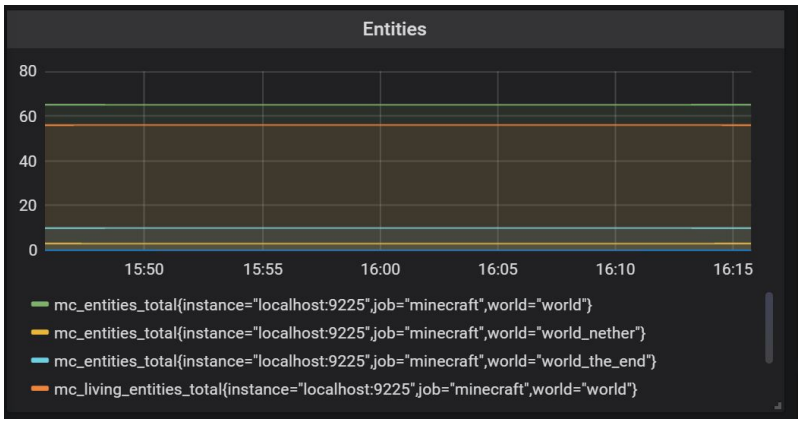
3. **Change Query** from default **to the name you gave your Prometheus Data Source, the default name is : Prometheus**
4. Select the metrics field and fill in mc_ to see all the Minecraft metrics available
5. Select the mc_entities_total
6. We are now going to add another query to display on this panel.
7. **Click on the Add Query** button found in the panel menu, as shown in the screenshot below.



8. In the second Query field that was created , **type in mc_living_entities_total**
9. **Click on the General tab** on the left and **change the title of the panel** from “Panel Title” to “Entities”



10. Click on your browser's back button to see the dashboard reflecting the changes you have made.
11. If everything is being imported correctly, the dashboard should look like the screenshot below

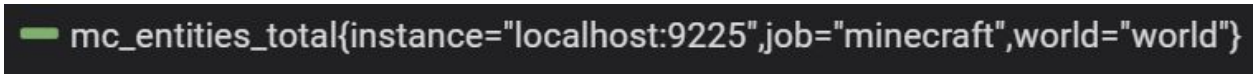


*This dashboard is displaying more than just two sets of data. This is because mc_living_total has several subsections. By querying for mc_living_total, you are asking for **all** of the mc_living_total data available.*

Clean up the Panel Key

At this point you have a dashboard that displays the entities data, but the panel is not very clear. We are going to edit it to make the graph easier to read.

1. **Select the Entities panel** you just created
2. **Click Edit**
3. Each metric has identifiers as seen in the screenshot below:



   a. For this metric there are three
      i. **Instance:** identifies which IP address this metric is from
      ii. **Job :** identifies which Prometheus job this metric is from
      iii. **World :** identifies which Minecraft world this metric is from
4. We can use these identifiers in the dashboard's legend to make reading the data easier to read. We are going to use the world identifier.

5. Add **{{world}}** into the **legends** field in both the mc_living_entitied and mc_entities_total query.
    a. This will remove all the extra information in the dashboards legend. Showing only the world that the information is coming from.
6. Now we need to differentiate the living entities from the total entities
7. Replace *{{world}}* with *{{world}}_total* in the mc_entities_total query's legend field.
8. Replace *{{world}}* with *{{world}}*_living in the mc_living_entities_total query's legend field .
9. Back out and save the dashboard
10. **Take a screenshot of your whole dashboard to turn in**

When Finished

After you have completed this tutorial:

1. Stop your Minecraft server, you will no longer need it
    a. You can do this by typing the following command into your minecraft terminal : *stop*
    b. Keep track of the dashboard screenshot to be turned in later
2. **Complete the Lab found below**

Lab

Application Monitoring with Prometheus and Grafana

Introduction

While there are many exporters that automatically collect data from certain programs, exporters will not always collect the data you want or be available at all. **Instrumenting is how you can manually expose data** you want to Prometheus.

The program that you will instrument today is a very simple server. It runs a binary search tree that randomly adds and deletes a value every second. Even though you probably will not encounter a server like this one in the workforce, it will provide a simplified environment to practice exposing data to Prometheus.

The Lab

Create a new Maven IntelliJ project. Then download the source files found at the link below and place the .java files in the src folder of your IntelliJ project.

Source Files:
https://drive.google.com/file/d/1CiKD3vPLbD-PKH4XS3M1EufPXwH7tOkp/view?usp=sharing

Main.java

- The main method:
  - Runs the algorithm in a background thread
  - Creates an HTPPServer which is a Prometheus exporter
  - Runs the server on a port
- The Main.java has a single counter metric defined as an example to follow.
- Make sure to read comments in the Main.java file to understand how the different functions work.

Dependencies

- **Add the dependencies and maven compiler plugin** that are found in the link below to your pom file: https://docs.google.com/document/d/1UsHdoW6hyX2yJexgFwRV4Z5g4p9n44L4kSXseFQRCDE/edit?usp=sharing
  - The maven compiler plugin sets the java language level to 8
  - The dependencies allow you to expose data in a format that Prometheus can read

BST, BSTNode, and FailedRemoveException

- These are the classes of the Binary Search Tree that the main method will be using.
- **DO NOT MODIFY**
- You do not need to touch these files to instrument your code.

Configure Prometheus:

1. **Stop Prometheus**
2. **Update your prometheus.yml** to make Prometheus scrapes data from your server .
   a. Copy and paste the scrape_config text below **over the old scrape config** in your prometheus.yml

```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'
    static_configs:
    - targets: ['localhost:9090']

  - job_name: 'minecraft'
    static_configs:
    - targets: ['localhost:9225']

  - job_name: 'javaSever'
    static_configs:
    - targets: ['localhost:8080']
```

3. **Start Prometheus** again
   a. use the following command to run prometheus

i. Mac: **prometheus --config.file=<directory for your config file>/prometheus.yml**
ii. Windows: Run the prometheus.exe file

Requirements

You are to instrument the given server to display the data given below. Then take a screenshot of the Grafana dashboard displaying the information scraped from the running server.

1. **Instrument the Main.java file**
   Instrumenting Java code is done by creating a metric object and then using this metric object to increment, decrement, keep track of a timer, or keep track of anything else the metric is capable of doing. By creating and registering the metric object, the data it collects becomes viewable by the Prometheus Server when your server is running.

   - The specific metrics and java code you will need to instrument the server is described in the Prometheus Java Guide in the link below: https://docs.google.com/document/d/1lTysUo1-1vW8Epqyx6LKSEl9vd9Ns4dSd5dC_guTdUs/edit?usp=sharing

   - The example metric is there to show you how a counter is initialized and how it is used to keep track of the number of times the server runs through its thread. It can be ignored or deleted.

   The data you will need to collect for this Lab is shown below:

   1. Count the number of failed removes **(use Counter Metric)**
      - Increment a counter when the server fails to remove a number for any reason

   2. Count the number of failed adds **(use Counter metric)**
      - Increment a counter when the server fails to add a number for any reason

   3. Keep track of the number of nodes currently on the Server **(use Gauge metric)**
      - Describes the number of BST nodes currently on the server

   4. Measure the amount of time it takes for the BST add method to run **(use Summary Metric)**
      - You will have to use a timer to measure how long it takes for the add method to run.

   To verify your instrumentation, navigate to http://localhost:8080/metrics

- This page will display the raw data being collected by Prometheus from port 8080.
- **Note:** Data will only be collected while your server is running

2. **Display the data you gather in your Main.java file on a Grafana Dashboard**
   Once the data is being collected, you will create your own Grafana Dashboard with panels to display the data you collect.

   Create a new Grafana Dashboard
   1. On the left menu of Grafana, **Click the plus symbol,** then **Click "Dashboard"** from the menu displayed by the Plus Symbol
      a. This will create a new empty Dashboard. We will be adding 4 panels to this dashboard.

   Create Four Grafana Panels that displays the following information about the server:

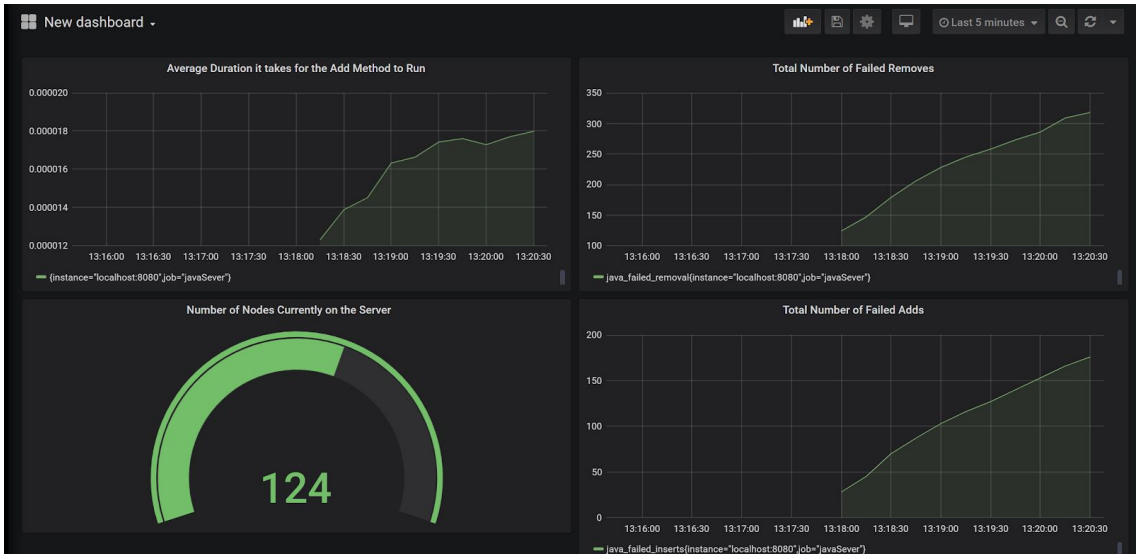
   1. The total number of failed removes **(use Counter Metric)**

   2. The total number of failed adds **(use Counter metric)**

   3. The number of nodes currently on the Server **(use Gauge metric)**
      - Use a Gauge Grafana visual to display this data
        - Follow the picture guide at the bottom of this document if you don't know how to use a Gauge Visual

   4. The average duration of the BST add method**(use Summary Metric)**
      - In your Main.java instrumentation, you kept track of the time it took to iterate through the add function. Using prometheus query expressions, we can use the data collected to get the average duration.
      - **<u>The section below explains how to use a Prometheus query expression to calculate the values for this panel</u>**

   At the end your Grafana Panels should look similar to the screenshot below



To expose the data gathered by your metrics to Prometheus, **run the main method** found in the Main.java file. Your server will then expose data gathered by the metric objects you created. Prometheus will then be able to collect this data from localhost:8080.

3. **Take a screenshot of the Grafana Dashboard you created**

How to get Average Rate from a Prometheus Query

1. To display the Average duration of the add function you will use the Prometheus Query Function: Rate.

To calculate the average request duration during the last 5 minutes from a histogram or summary called `http_request_duration_seconds`, use the following expression:

```
  rate(http_request_duration_seconds_sum[5m])
/
  rate(http_request_duration_seconds_count[5m])
```

2. A summary metric has a few different parts that can be pulled out.
    a. Adding "_sum" to the end of your summary metric query gets the sum of the observations
    b. Adding "_count" to the end of your summary metric query gets the number of observations
3. In Grafana, this is done by placing the metric within the rate function in the Query field of the dashboard.

What to turn in

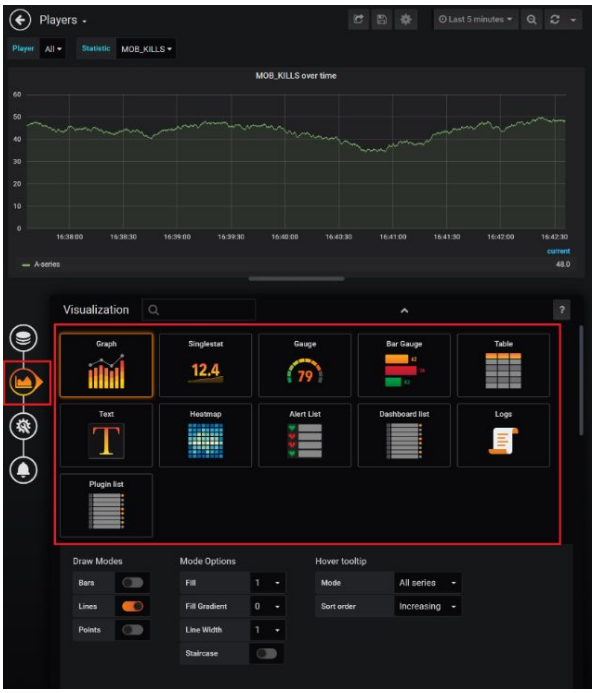
To receive credit for the lab, turn in:

- **Your modified Main.java**
- **1 screenshot of your Minecraft Grafana Dashboard**
- **1 screenshot of your working Java server Grafana Dashboard**

Hints

- Make sure your Prometheus.yml is configured to target your Intellj server and collect data from it
- Use http://localhost:9090/targets to make sure that your server's data is being collected
- Use http://localhost:8080/metrics to see what your defined metrics look like
- **If you change or add a Metric in your java program, make sure to update the DataSource on Grafana to get updated metrics.**
    - You do this by going to Configure -> Data Source -> Your_Data_Source -> Save & Test. This is shown in screenshots below
- If Grafana doesn't seem to be displaying your data, refresh the data by making Grafana display the last 5 minutes of data.

How to Change the Visualization of A Panel (in Dashboard Edit menu)



How to Change Grafana Time Range