

1. Install Prometheus Server

- Configuration basic authentication username/password
- Screenshot of login prompt while trying to access prometheus
- Screenshot of prometheus dashboard

Prometheus is installed in our system (Server1 with ip address 192.168.1.64) by downloading a tar.gz file and extracting and installing which used following commands:

wget

<https://github.com/prometheus/prometheus/releases/download/v2.28.0/prometheus-2.28.0.linux-amd64.tar.gz>

tar xvzf prometheus-2.28.0.linux-amd64.tar.gz

```
bj@vm2:~/prometheus$ tar xvzf prometheus-2.28.0.linux-amd64.tar.gz
prometheus-2.28.0.linux-amd64/
```

sudo mv -v prometheus-2.28.0.linux-amd64 /opt/prometheus

sudo chown -Rfv root:root /opt/prometheus

sudo chmod -Rfv 0755 /opt/prometheus

A prometheus user is created:

```
bj@vm2:~$ sudo useradd --system --no-create-home --shell /usr/sbin/nologin prometheus
bj@vm2:~$ ls
```

Then configuration for prometheus is edited in prometheus.yml file using this command:

sudo nano /opt/prometheus/prometheus.yml

And inside the yml file, mostly default values may work, the server target of prometheus must be correctly specified in scrape_configs as shown in following figure:

```
bj@vm2: ~/prometheus x root@vm2: /etc/nginx x bj@batman: /etc/nginx x
GNU nano 4.8 /opt/prometheus/prometheus.yml Modified
# my global config
global:
  scrape_interval: 15s
  # Set the scrape interval to every 15 seconds. Default is every 1 minute.

  evaluation_interval: 15s
  # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).
  # Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
      - targets:
          # - alertmanager:9093
# Load rules once and periodically evaluate them according to the global 'evaluation_in>
rule_files:
  # - "first_rules.yml" # - "second_rules.yml"
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label 'job=<job_name>' to any timeseries scraped from th>
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

  static_configs:
    - targets: ['localhost:9090']
```

To store prometheus metrics, a directory named data is created:

Mkdir -v /opt/prometheus/data

Then a systemd service file is created for prometheus:

sudo nano /etc/systemd/system/prometheus.service

```
GNU nano 4.8 /etc/systemd/system/prometheus.service
[Unit]
Description=Monitoring system and time series database

[Service]
Restart=always
User=prometheus
ExecStart=/opt/prometheus/prometheus --config.file=/opt/prometheus/prometheus.yml --sto>
ExecReload=/bin/kill -HUP $MAINPID
TimeoutStopSec=20s
SendSIGKILL=no
LimitNOFILE=8192

[Install]
WantedBy=multi-user.target
```

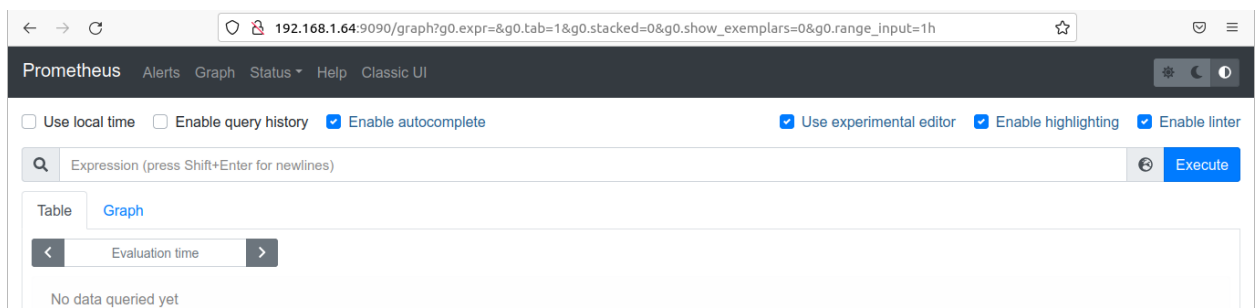
sudo systemctl daemon-reload

sudo systemctl enable prometheus && sudo systemctl start prometheus

sudo systemctl status prometheus

```
bj@vm2:~/prometheus$ sudo systemctl status prometheus.service
● prometheus.service - Monitoring system and time series database
   Loaded: loaded (/etc/systemd/system/prometheus.service; disabled; vendor preset: enabled)
   Active: active (running) since Fri 2021-12-03 00:26:39 +0545; 4s ago
     Main PID: 5939 (prometheus)
        Tasks: 5 (limit: 2711)
      Memory: 17.1M
      CGroup: /system.slice/prometheus.service
              └─5939 /opt/prometheus/prometheus --config.file=/opt/prometheus/prometheus
```

Now when can access the dashboard using socket address: 192.168.1.64:9090



For authentication, we used a nginx server, running at port 80, which acts as a reverse proxy server for the prometheus server and we set up the authentication in the nginx server. For that, htpasswd file is generated which is used to authenticate the nginx server.

Htpasswd -c .htpasswd prometheus

```
Processing triggers for libc-bin (2.31-0ubuntu0.21.10) ...
root@vm2:/etc/nginx# htpasswd -c .htpasswd prometheus
New password:
Re-type new password:
Adding password for user prometheus
root@vm2:/etc/nginx#
```

The server configuration for nginx is as:

Nano /etc/nginx/sites-available/nginx.conf

For reverse proxy and authentication, this content was saved in nginx.conf file:

```

server_name 192.168.1.64;

location / {

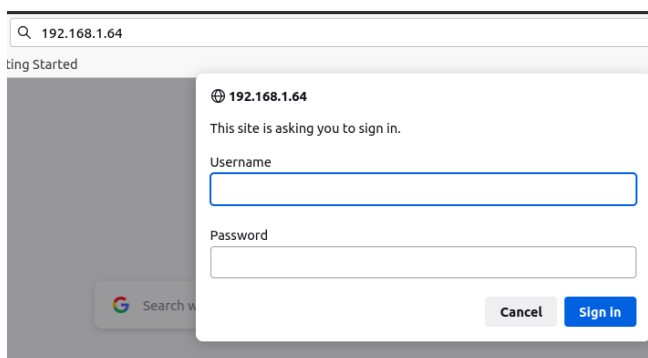
    proxy_pass http://192.168.1.64:9090;
    proxy_set_header Connection keep-alive;
    auth_basic "prometheus";
    auth_basic_user_file "../.htpasswd";

    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404
    try_files $uri $uri/ =404;
}

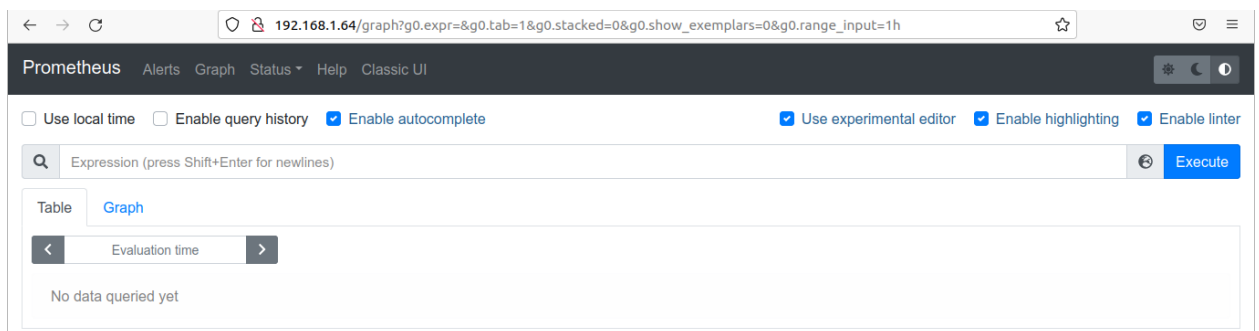
```

Systemctl restart nginx

Now when we access the server ip: 192.168.1.64:80, it prompts for authentication.



And only after providing correct credentials provided before, we can access the prometheus server.



2. Install node exporter on another machine than the server

- Add that machine target to server configuration
- Share screenshot from status->targets to show the available nodes
- Share configuration of node exporter & prometheus server

We installed Node Exporter on another server with ip 192.168.1.67 which is my host machine in this case.

Node exporter is installed by downloading a tar file and extracting it in our location.

wget

https://github.com/prometheus/node_exporter/releases/download/v1.1.2/node_exporter-1.1.2.linux-amd64.tar.gz

```
bj@batman:~$ cd prometheus/  
bj@batman:~/prometheus$ wget https://github.com/prometheus/node_exporter/releases/download/v1.1.2/node_exporter-1.1.2.linux-amd64.tar.gz  
--2021-12-03 00:51:42-- https://github.com/prometheus/node_exporter/releases/download/v1.1.2/node_exporter-1.1.2.linux-amd64.tar.gz  
Resolving github.com (github.com)... 20.205.243.166
```

tar xzf node_exporter-1.1.2.linux-amd64.tar.gz

sudo mv -v node_exporter-1.1.2.linux-amd64/node_exporter /usr/local/bin/

sudo chown root:root /usr/local/bin/node_exporter

```
bj@batman:~/prometheus$ tar xzf node_exporter-1.1.2.linux-amd64.tar.gz  
bj@batman:~/prometheus$ ls  
node_exporter-1.1.2.linux-amd64  node_exporter-1.1.2.linux-amd64.tar.gz  
bj@batman:~/prometheus$ sudo mv -v node_exporter-1.1.2.linux-amd64/node_exporter /usr/local/bin/  
copied 'node_exporter-1.1.2.linux-amd64/node_exporter' -> '/usr/local/bin/node_exporter'  
removed 'node_exporter-1.1.2.linux-amd64/node_exporter'  
bj@batman:~/prometheus$ sudo chown root:root /usr/local/bin/node_exporter
```

Now a service file is created for node-exporter.

sudo nano /etc/systemd/system/node-exporter.service

And this content is saved in the service file:

```
GNU nano 4.8 /etc/systemd/system/node-exporter.service
[Unit]
Description=Prometheus exporter for machine metrics
Wants=network-online.target
After=network-online.target

[Service]
Restart=always
User=root
ExecStart=/usr/local/bin/node_exporter --collector.textfile.directory=/prometheus/metrics
ExecReload=/bin/kill -HUP $MAINPID
TimeoutStopSec=20s
SendSIGKILL=no

[Install]
WantedBy=multi-user.target
```

Daemon is reloaded and node-exporter is started:

sudo systemctl daemon-reload

sudo systemctl enable node-exporter.service && sudo systemctl start node-exporter.service

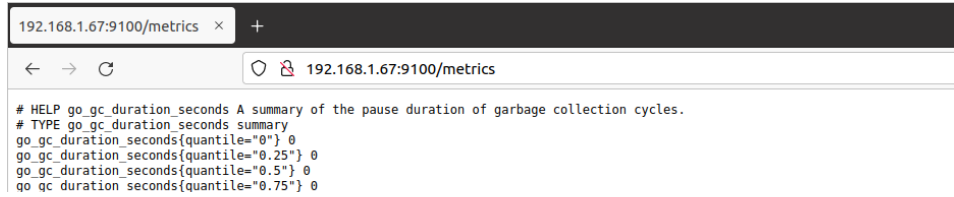
```
bj@batman:~/prometheus$ sudo nano /etc/systemd/system/node-exporter.service
bj@batman:~/prometheus$ sudo systemctl daemon-reload
bj@batman:~/prometheus$ sudo systemctl start node-exporter
bj@batman:~/prometheus$ sudo systemctl enable node-exporter
Created symlink /etc/systemd/system/multi-user.target.wants/node-exporter.service → /etc/systemd/system/node-exporter.service.
bj@batman:~/prometheus$ sudo systemctl status node-exporter
```

sudo systemctl status node-exporter.service

```
bj@batman:~/prometheus$ sudo systemctl status node-exporter.service
● node-exporter.service - Prometheus exporter for machine metrics
   Loaded: loaded (/etc/systemd/system/node-exporter.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2021-12-03 01:10:14 +0545; 40s ago
     Main PID: 47249 (node_exporter)
        Tasks: 6 (limit: 9110)
       Memory: 2.7M
      CGroup: /system.slice/node-exporter.service
              └─47249 /usr/local/bin/node_exporter --collector.textfile.directory=/prometheus/metrics
```

Now with our ip, 192.168.1.67 we can see metrics in our browser using url:

192.168.1.67:9100/metrics

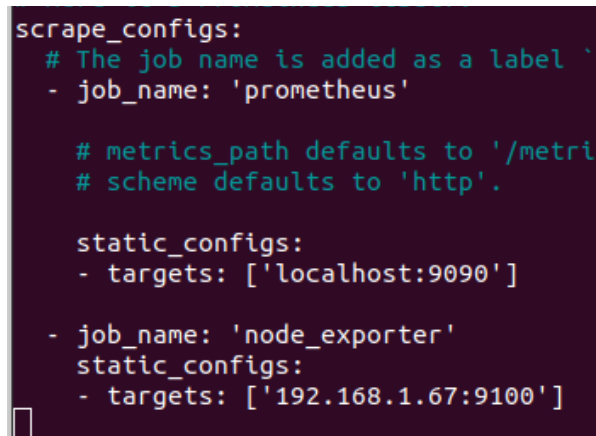


```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
```

Now, node exporter was added to prometheus in prometheus.yml file:

sudo nano /opt/prometheus/prometheus.yml

And inside this file, this part was appended in scrape_configs section:



```
scrape_configs:
  # The job name is added as a label `
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'node_exporter'
    static_configs:
      - targets: ['192.168.1.67:9100']
```

And prometheus was restarted:

sudo systemctl restart prometheus

Now in our browser, accessing the Prometheus server, we can see the targets with some details as follows.

| Prometheus Alerts Graph Status ▾ Help Classic UI | | | | | |
|--|-------|---|-------------|-----------------|---|
| Targets | | | | | |
| All Unhealthy Collapse All | | | | | |
| node_exporter (0/1 up) show less | | | | | |
| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
| http://192.168.1.67:9100/metrics | DOWN | instance="192.168.1.67:9100" job="node_exporter" | 22.317s ago | 10.0s | Get "http://192.168.1.67:9100/metrics": context deadline exceeded |
| prometheus (1/1 up) show less | | | | | |
| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
| http://localhost:9090/metrics | UP | instance="localhost:9090" job="prometheus" | 3.914s ago | 4.021ms | |

Since we are in different servers, we have to allow traffic from the first server in second server with following command:

Sudo ufw allow from 192.168.1.64 to 192.168.1.67 port 9100

Sudo ufw reload

```
bj@batman:~/prometheus$ sudo ufw allow from 192.168.1.64 to 192.168.1.67 port 9100
Rule added
bj@batman:~/prometheus$ sudo ufw status
```

Now all of the targets are up. We can manually see graphs of metrics also.

| Prometheus Alerts Graph Status ▾ Help Classic UI | | | | | |
|--|-------|---|-------------|-----------------|-------|
| Targets | | | | | |
| All Unhealthy Collapse All | | | | | |
| node_exporter (1/1 up) show less | | | | | |
| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
| http://192.168.1.67:9100/metrics | UP | instance="192.168.1.67:9100" job="node_exporter" | 12.161s ago | 74.325ms | |
| prometheus (1/1 up) show less | | | | | |
| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
| http://localhost:9090/metrics | UP | instance="localhost:9090" job="prometheus" | 8.762s ago | 26.406ms | |

3. Install grafana server on same server as prometheus

- Add prometheus data source to grafana, should be connected through basic auth
- Screenshot of working data source config
- Import & apply dashboard for node_exporter
- Screenshot of dashboard of nodes with live metrics shown.

First, I ssh into the server containing prometheus with ssh command as:

```
bj@batman:~/Downloads/LeapFrog Internship Documents$ ssh bj@192.168.1.64
bj@192.168.1.64's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-41-generic x86_64)
```

And installed the required packages by adding gpg key and updating packages and then finally installing grafana with the following commands:

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a
/etc/apt/sources.list.d/grafana.list
sudo apt-get update
sudo apt-get install grafana
```

```
root@vm2:/home/bj/prometheus# sudo apt-get install grafana
Reading package lists... Done
Building dependency tree
```

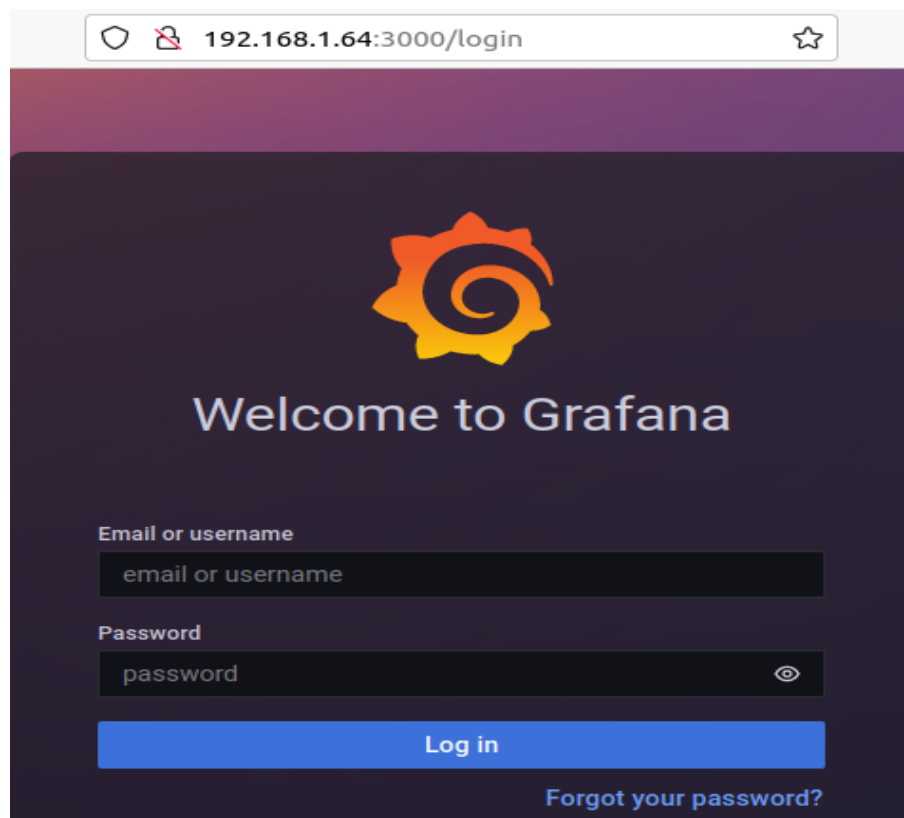
To start and enable grafana, we used following commands:

```
sudo systemctl start grafana-server
sudo systemctl enable grafana-server
sudo systemctl status grafana-server
```

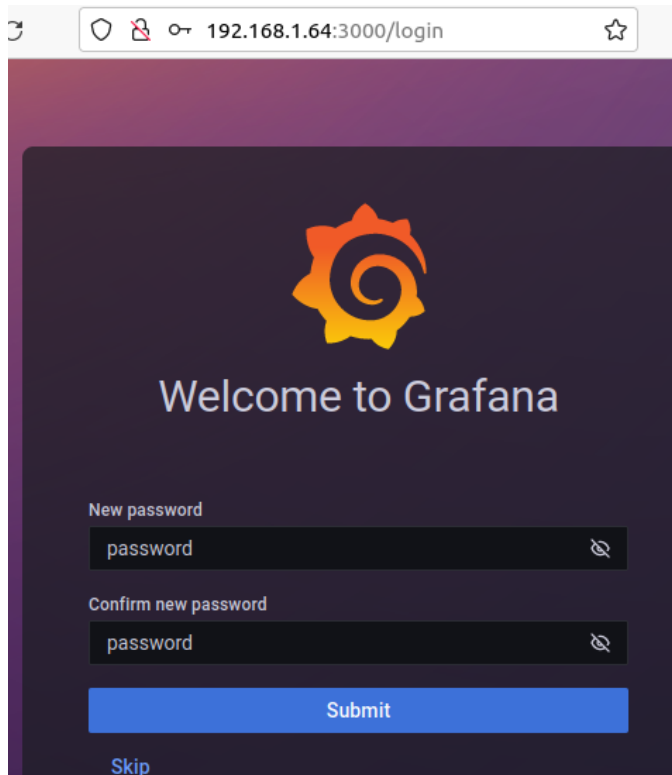
```
root@vm2:/home/bj/prometheus# sudo systemctl start grafana-server
root@vm2:/home/bj/prometheus# sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
   Loaded: loaded (/lib/systemd/system/grafana-server.service; disabled; vendor preset: enabled)
   Active: active (running) since Fri 2021-12-03 23:08:00 +0545; 8s ago
     Docs: http://docs.grafana.org
   Main PID: 3800 (grafana-server)
    Tasks: 9 (limit: 2711)
   Memory: 29.9M
   CGroup: /system.slice/grafana-server.service
           └─3800 /usr/sbin/grafana-server --config=/etc/grafana/grafana
```

Now that the grafana server is up and running, we can visit the dashboard in browser:

With server address- 192.168.1.64:3000

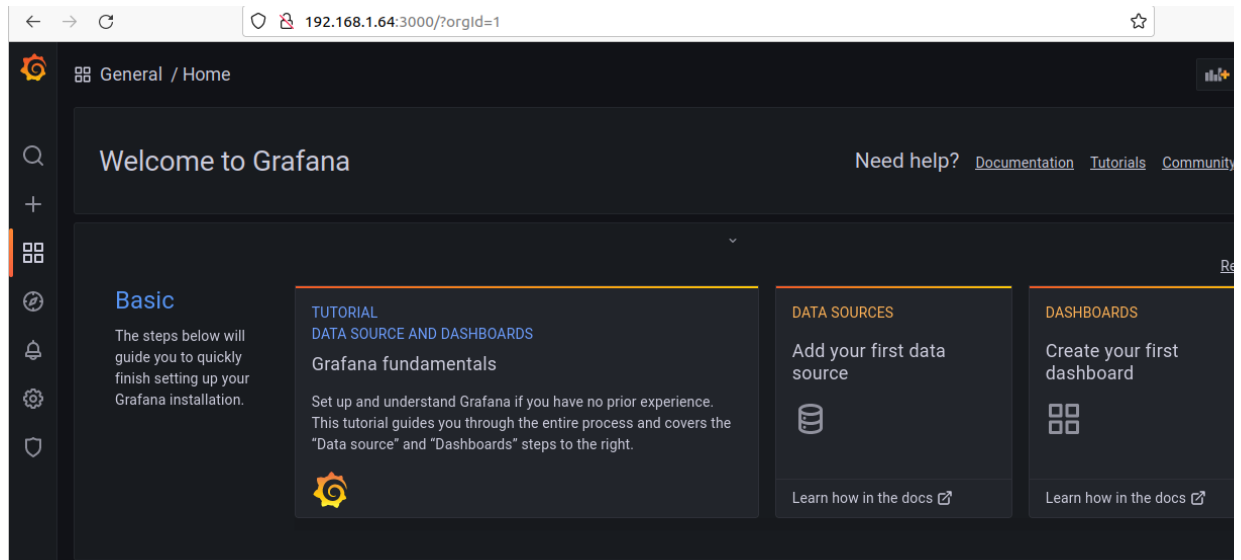


By logging in with default username '*admin*' and password '*admin*' we can change the default password for the first time:



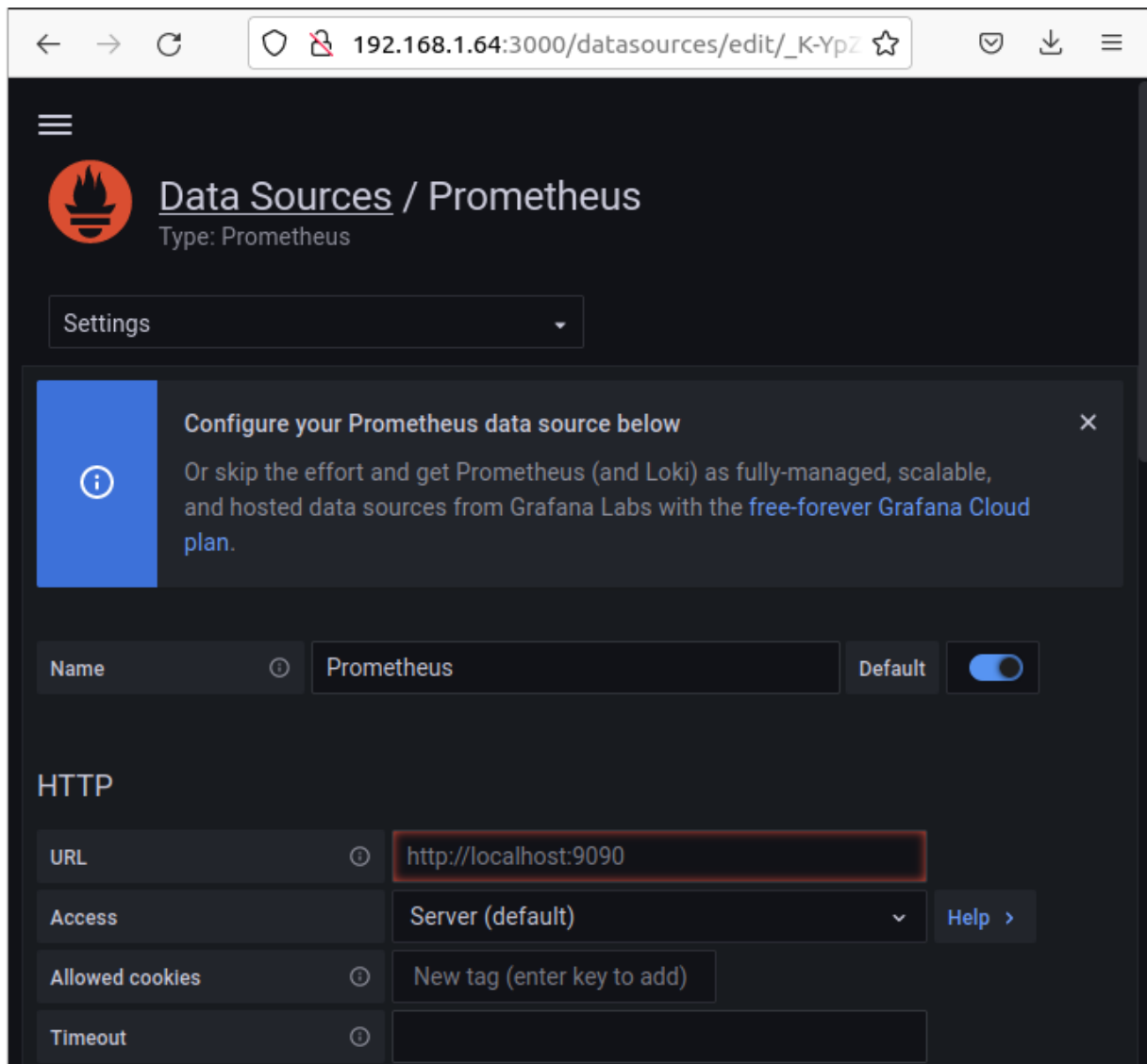
A screenshot of the Grafana login page. The browser address bar shows '192.168.1.64:3000/login'. The page features the Grafana logo (an orange and yellow gear with a spiral) and the text 'Welcome to Grafana'. Below this, there are two password input fields. The first is labeled 'New password' and the second is labeled 'Confirm new password'. Both fields contain the text 'password'. To the right of each field is a small icon to toggle password visibility. Below the input fields is a blue 'Submit' button. At the bottom left, there is a blue link labeled 'Skip'.

And after setting new password, we can access the dashboard:



A screenshot of the Grafana dashboard home page. The browser address bar shows '192.168.1.64:3000/?orgId=1'. The page has a dark theme. At the top, there is a navigation bar with the Grafana logo, a search icon, and a 'General / Home' breadcrumb. Below the navigation bar, there is a 'Welcome to Grafana' message and a 'Need help?' link with sub-links for 'Documentation', 'Tutorials', and 'Community'. The main content area is divided into three columns. The first column is titled 'Basic' and contains a 'TUTORIAL' section titled 'DATA SOURCE AND DASHBOARDS' with a sub-section 'Grafana fundamentals'. The second column is titled 'DATA SOURCES' and contains a section 'Add your first data source'. The third column is titled 'DASHBOARDS' and contains a section 'Create your first dashboard'. Each of these three sections has a 'Learn how in the docs' link with an external link icon.

Now Prometheus is to be added as data sources from settings>DataSources:

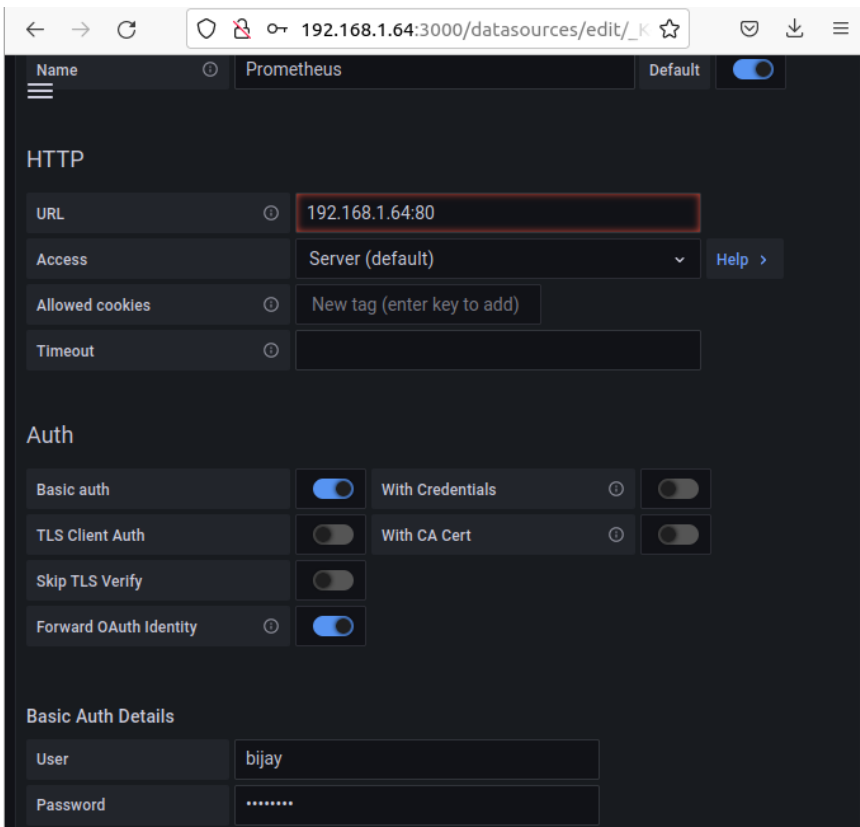


The screenshot shows the Grafana web interface for configuring a Prometheus data source. The browser address bar shows the URL `192.168.1.64:3000/datasources/edit/_K-YpZ`. The page title is "Data Sources / Prometheus" with the subtitle "Type: Prometheus". A "Settings" dropdown menu is visible. A blue information box contains the text: "Configure your Prometheus data source below" and "Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafana Labs with the [free-forever Grafana Cloud plan](#)." Below this, the "Name" field is set to "Prometheus" and is marked as the "Default" source with a toggle switch. The "HTTP" section contains a table of configuration options:

| Field | Value |
|-----------------|------------------------------------|
| URL | <code>http://localhost:9090</code> |
| Access | Server (default) |
| Allowed cookies | New tag (enter key to add) |
| Timeout | |

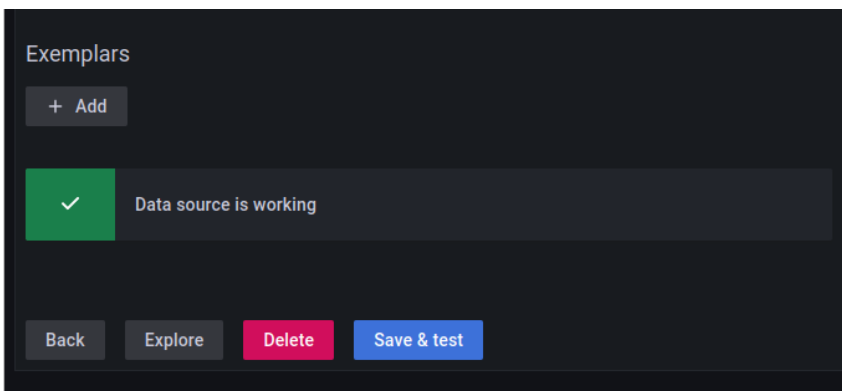
And the server address of prometheus is entered in URL field:

If needed, basic auth can be used for basic authentication to the server.



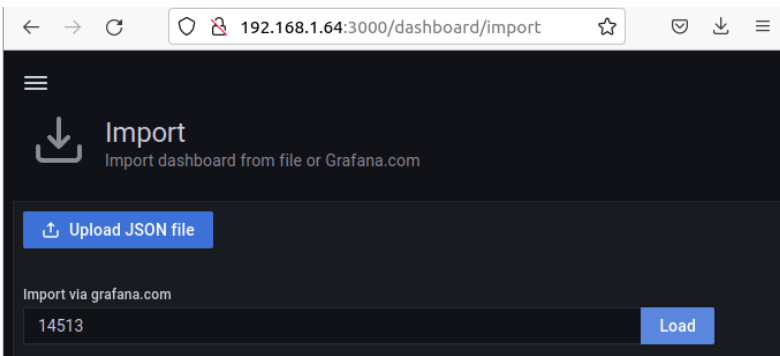
The screenshot shows the 'Prometheus' data source configuration page in Grafana. The browser address bar displays '192.168.1.64:3000/datasources/edit/_k'. The configuration is set to 'Default' and is enabled. Under the 'HTTP' section, the 'URL' field is highlighted with a red border and contains '192.168.1.64:80'. The 'Access' dropdown is set to 'Server (default)'. The 'Allowed cookies' field contains 'New tag (enter key to add)'. The 'Timeout' field is empty. Under the 'Auth' section, 'Basic auth' is enabled, and 'With Credentials' is disabled. 'TLS Client Auth' is disabled, and 'With CA Cert' is disabled. 'Skip TLS Verify' is disabled. 'Forward OAuth Identity' is enabled. Under the 'Basic Auth Details' section, the 'User' field contains 'bijay' and the 'Password' field contains masked characters '*****'.

Then save and test returns the result 'Data source is working'

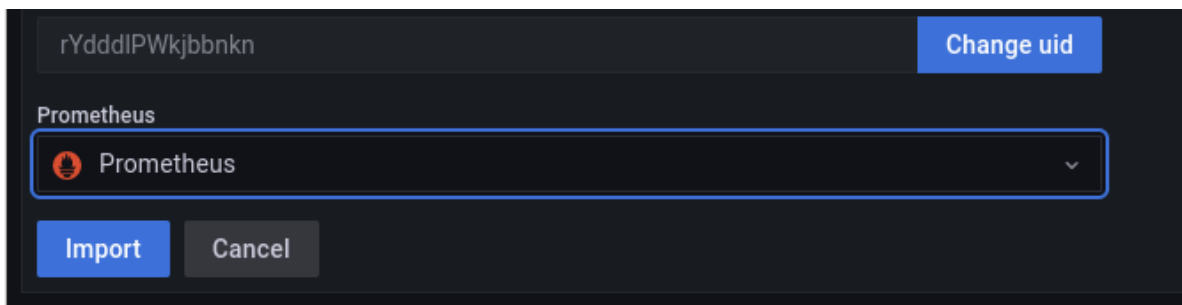


The screenshot shows the 'Exemplars' section in Grafana. At the top, there is a '+ Add' button. Below it, a green checkmark icon is displayed next to the text 'Data source is working'. At the bottom, there are four buttons: 'Back', 'Explore', 'Delete', and 'Save & test'.

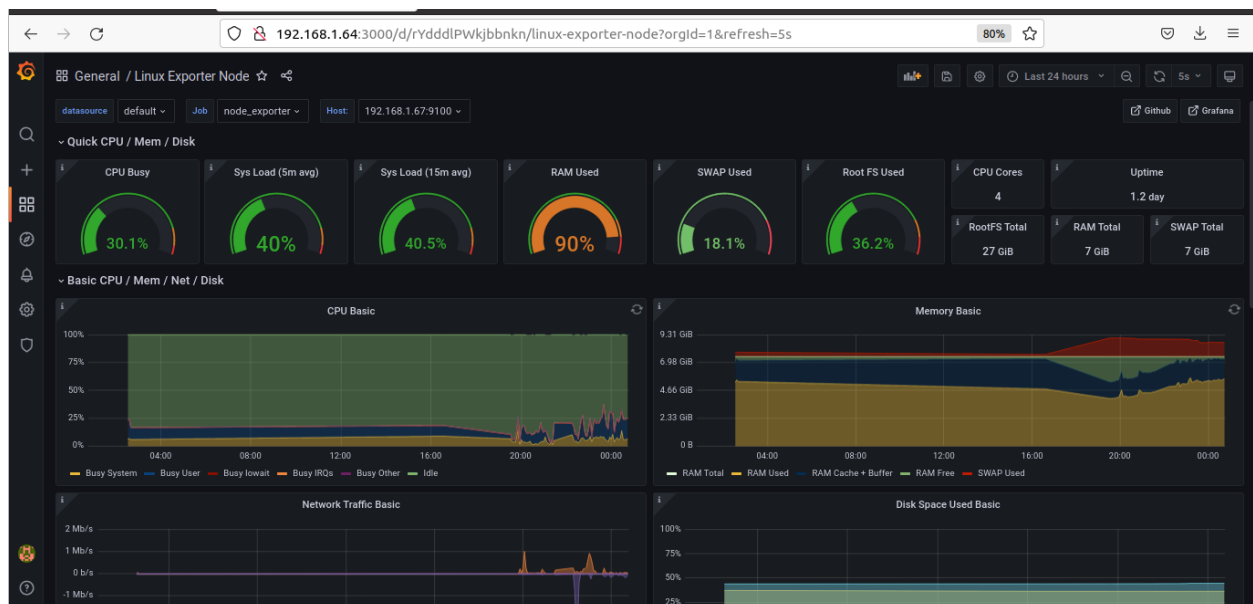
To build dashboard in grafana, we use 14513 to import grafana by selecting + button on left side of the dashboard and then import:



Prometheus is selected for datasource and click on import.



Now we can see live metrics which are refreshed every 5 seconds.



We can see other panels too which can be expanded just by clicking.