

BAB V

IMPLEMENTASI

5.1 Implementasi

Pada tahap Implementasi sitem akan menjelaskan tentang implementasi pembuatan arsitektur dan juga implementasi pembuatan dashboard monitoring Hadoop dengan Grafana yang sudah dianalisis dan dirancang pada tahap sebelumnya.

5.1.1 Implementasi pembuatan Arsitektur

Pada bagian implementasi pembuatan arsitektur ini akan menjelaskan tentang apa saja arsitektur yang akan dibuat untuk menunjang kebutuhan dashboard monitoring. Pembuatan arsitektur meliputi pembuatan Container, penentuan target untuk mengirimkan data metrik, serta pembuatan script yang berugas sebagai agent monitoring untuk mendorong (*push*) data metrik agar bisa terbaca oleh Prometheus, dan juga Grafana

5.1.1.1 Membuat Docker Container

Hal yang pertama dilakukan dalam pembuatan arsitektur adalah menginstalasi aplikasi apa saja yang akan digunakan seperti Hadoop, Prometheus, dan juga Grafana. Namun pada penelitian kali ini instalasi Hadoop dilakukan pada satu komputer yang sama dengan menggunakan Docker Container.

```
services:
  datanode1:
    image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
    container_name: datanode1
    restart: always
    volumes:
      - hadoop_datanode1:/hadoop/dfs/data
```

environment:

SERVICE_PRECONDITION: "namenode:9870"

env_file:

- ./hadoop.env

datanode2:

image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

container_name: datanode2

restart: always

volumes:

- hadoop_datanode2:/hadoop/dfs/data

environment:

SERVICE_PRECONDITION: "namenode:9870"

env_file:

- ./hadoop.env

datanode3:

image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

container_name: datanode3

restart: always

volumes:

- hadoop_datanode3:/hadoop/dfs/data

environment:

SERVICE_PRECONDITION: "namenode:9870"

env_file:

- ./hadoop.env

volumes:

hadoop_datanode1:

hadoop_datanode2:

hadoop_datanode3:

setelah menjalankan perintah Docker Container akan menghasilkan satu Namenode dan juga tiga Datanode dengan spesifikasi yang sama. Adapun yang bertindak sebagai Namenode adalah localhost dan ketiga Datanode bisa dilihat dengan mengakses localhost:9870.

Selain membuat docker container untuk menjalankan Hadoop, perlu dibuat juga beberapa container untuk menjalankan Prometheus, Prometheus gateway, dan juga Grafana. Pembuatan container ini membuat *software* yang digunakan dapat diakses pada browser.

```
• sudo docker run -dit --name=hadoop -p 9001:9000 --name myhadoop  
hadoop/Hadoop
```

```
• sudo docker run -dit -p 9093:9090 --name myprometheus  
prom/Prometheus
```

```
• sudo docker run -dit -p 9092:9091 --name mypushgateway  
prom/pushgateway
```

```
• sudo docker run -dit --name=grafana -p 3001:3000 --name mygrafana  
grafana/Grafana
```

setelah membuat docker container untuk menjalankan Prometheus, Prometheus pushgateway, dan juga Grafana akan menjalankan *software* pada browser dengan mengakses localhost:9090 untuk membuka Prometheus, localhost:9091 untuk membuka Prometheus Pushgateway, dan juga localhost:3000 untuk membuka Grafana.

5.1.1.2 konfigurasi target

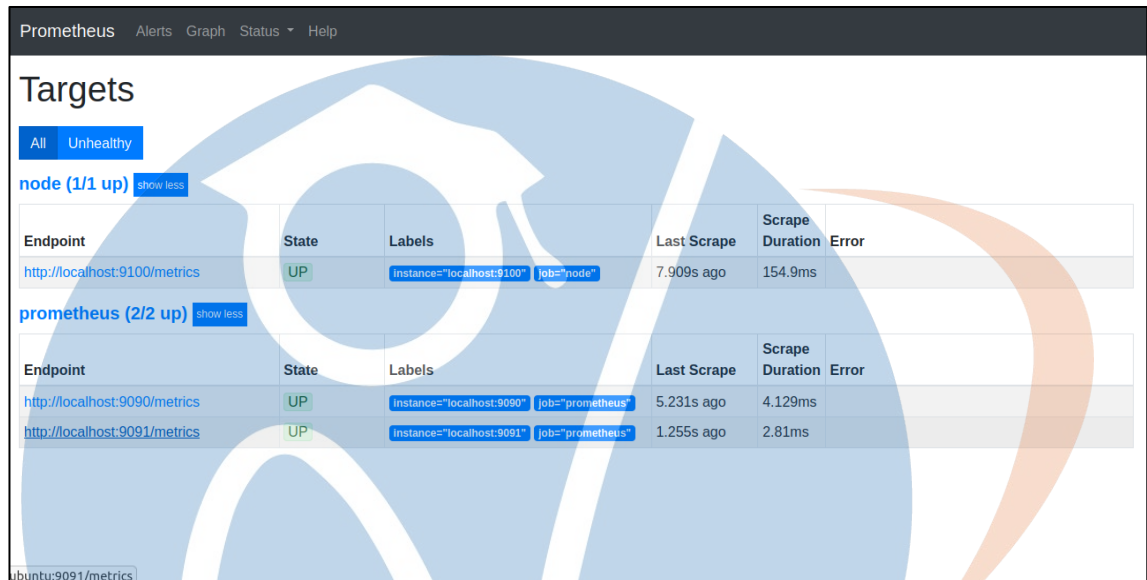
tahap selanjutnya setelah membuat docker container yang akan digunakan yaitu mengkonfigurasi atau mengatur target pada Prometheus. Pengaturan target ini dilakukan dengan menambahkan target yang akan diterima pada file Prometheus.yml.

scrape_configs:

- job_name: "prometheus"

static_configs:

- targets: ["localhost:9090","localhost:9091"]



Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
node (1/1 up) show less					
http://localhost:9100/metrics	UP	instance="localhost:9100" job="node"	7.909s ago	154.9ms	
prometheus (2/2 up) show less					
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	5.231s ago	4.129ms	
http://localhost:9091/metrics	UP	instance="localhost:9091" job="prometheus"	1.255s ago	2.81ms	

Gambar 5. 1 Target Prometheus

setelah mengatur target pada localhost:9091 prometheus sudah bisa menerima metrik-metrik yang dikirimkan oleh Prometheus pushgateway dan ditampilkan dalam bentuk *graph* dan *console*. Namun untuk mendapatkan metrik-metrik yang dimiliki oleh Hadoop perlu dibuat sebuah agent monitoring(*web scraping*).

5.1.1.3 Membuat program Python untuk Web Scraping

Pada tahap ini akan berfokus pada pembuatan program web scraping dengan menggunakan python *script*. *Web scraping* sendiri yaitu proses pengumpulan data web terstruktur yang dilakukan secara otomatis dengan menggunakan aplikasi atau kode pemrograman untuk mendorong (*push*) data metrik yang dimiliki oleh Hadoop melalui Prometheus pushgateway agar bisa diterima oleh Prometheus. Berikut python *script* yang digunakan sebagai agent monitoring *web scraping*:

1. namenode

```
# NOTE live namenode
cmd = "hdfs getconf -namenodes"
out = os.popen(cmd)
namen = out.readlines()
print("live namenode : ",len(namen),'\n')
lnamen = 'live_namen_1{unit="node"} %s\n' %(len(namen))
server_host = 'pushgateway'
pshgw_url = "http://localhost:9091/metrics/job/hadoop/instance/%s"%server_host
pshgw = requests.Session()
pshgw.post(pshgw_url,data=lnamen)Implementasi pembuatan dashboard
```

Python *script* diatas akan mengirimkan metrik Live Namenode dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu live_namen_1.

2. Live Datanode

```
cmd = "hdfs dfsadmin -report"
out = os.popen(cmd)
data = out.readlines()
# NOTE live datanode
mylist = data
r = re.compile("Live datanodes.*")
newlist = list(filter(r.match, mylist))
index = data.index(newlist[0])
data1 = data[index]
liven = ".join([n for n in data1 if n.isdigit()])
print("Live Datanode : ",liven,'\n')
data_live = 'data_live_2{unit="node"} %s\n' %(liven)
server_host = 'pushgateway'
pshgw_url = "http://localhost:9091/metrics/job/hadoop/instance/%s"%server_host
pshgw = requests.Session()
pshgw.post(pshgw_url,data=data_live)
```

Python *script* diatas akan mengirimkan metrik Live Datanode dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu data_live_2.

3. Configure Capacity Total

```
# NOTE Configured capacity total
r = re.compile("Configured Capacity.*")
mylist = data
newlist = list(filter(r.match, mylist))
index = data.index(newlist[0])
g = Gauge('config_cc_nm_1', 'kapasitas yang diset', ['config_cap_2'],
registry=registry)
for x in range(len(newlist)):
    index = data.index(newlist[x])
    if x == 0:
        data1 = data[index]
        spl = data1.split()
        ipn = ".join([n for n in spl[2] if n.isdigit()])
        g.labels(config_cap_2=x).set(ipn)
        print('configured capacity namenode :',ipn,'\n')
    x += 1
```

Python *script* diatas akan mengirimkan metrik Configured Capacity Total dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu config_cc_nm_1.

4. Configure Capacity Per Node

```
# NOTE Configured capacity
r = re.compile("Configured Capacity.*")
mylist = data
newlist = list(filter(r.match, mylist))
index = data.index(newlist[0])
g = Gauge('config_cc_4', 'kapasitas yang diset', ['config_cap_1'], registry=registry)
```

```

for x in range(len(newlist)):
    index = data.index(newlist[x])
    if x != 0:
        data1 = data[index]
        spl = data1.split()
        ipn = ".join([n for n in spl[2] if n.isdigit()])
        g.labels(config_cap_1=x).set(ipn)
        print('configured capacity :',ipn,"\n")
        x += 1

```

Python *script* diatas akan mengirimkan metrik Configured Capacity Per node dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu config_cc_4.

5. Present Capacity

```

# NOTE present capacity
mylist = data
r = re.compile("Present Capacity.*")
newlist = list(filter(r.match, mylist)) # Read Note
index = data.index(newlist[0])
data1=data[index]
pc=".join([n for n in data1 if n.isdigit()])
pc = round(int(pc)/10737418240000,2)
present = 'pre_capacity_2{unit="ms"} %s\n' %(pc)
server_host = 'pushgateway'
pshgw_url = "http://localhost:9091/metrics/job/hadoop/instance/%s"%server_host
pshgw = requests.Session()
pshgw.post(pshgw_url,data=present)

```

Python *script* diatas akan mengirimkan metrik Present Capacity dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu pre_capacity_2.

6. Dfs Used Total

```
# NOTE DFS Used total
r = re.compile("DFS Used:.*")
mylist = data
newlist = list(filter(r.match, mylist))
index = data.index(newlist[0])
g = Gauge('dfs_usednm_1', 'used dfs nm', ['set_dfsum'], registry=registry)
for x in range(len(newlist)):
    index = data.index(newlist[x])
    data1 = data[index]
    spl = data1.split()
    if x == 0:
        ipn = ".join([n for n in spl[2] if n.isdigit()])
        g.labels(set_dfsum=x).set(ipn)
        print('dfs used namenodes :',ipn,"\n")
    x += 1
```

Python *script* diatas akan mengirimkan metrik DFS Used Total dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu `dfs_usednm_1`.

7. Dfs Used Per Node

```
# NOTE DFS Used
r = re.compile("DFS Used:.*")
mylist = data
newlist = list(filter(r.match, mylist))
index = data.index(newlist[0])
g = Gauge('dfs_used_3', 'used dfs', ['set_dfsum'], registry=registry)
for x in range(len(newlist)):
    index = data.index(newlist[x])
    data1 = data[index]
    spl = data1.split()
    if x != 0:
```



```

ipn = ".join([n for n in spl[2] if n.isdigit()])
g.labels(set_dfsu=x).set(ipn)
print('dfs used nodes :',ipn,"\n")
x += 1

```

Python *script* diatas akan mengirimkan metrik DFS Used Per node dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu dfs_used_3.

8. Dfs Remains Total

```

# NOTE Dfs remaining namenode
r = re.compile("DFS Remaining:.*")
mylist = data
newlist = list(filter(r.match, mylist))
index = data.index(newlist[0])
data1 = data[index]
spl = data1.split()
g = Gauge('dfs_remains_nmnode_1', 'DFS Remaining namenodes', ['set_nmdfsrn'],
registry=registry)
for x in range(len(newlist)):
    index = data.index(newlist[x])
    data1 = data[index]
    spl = data1.split()
    if x == 0:
        drn=spl[2]
        g.labels(set_nmdfsrn=x).set(drn)
        print('dfs remaining namenodes :',drn,"\n")
    x += 1

```

Python *script* diatas akan mengirimkan metrik DFS Remaining Total dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu dfs_remains_nmnode_1

9. Dfs Remains Per Node

```
# NOTE DFS Remaining
r = re.compile("DFS Remaining.*")
mylist = data
newlist = list(filter(r.match, mylist))
index = data.index(newlist[0])
data1 = data[index]
spl = data1.split()
g = Gauge('dfs_remains_node_2', 'DFS Remaining nodes', ['set_dfsm'],
registry=registry)
for x in range(len(newlist)):
    index = data.index(newlist[x])
    data1 = data[index]
    spl = data1.split()
    if x != 0:
        drn=spl[2]
        g.labels(set_dfsm=x).set(drn)
        print('dfs remaining nodes :',drn,"\n")
    x += 1
```

Python *script* diatas akan mengirimkan metrik DFS Remains per node dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu dfs_remains_node_2.

10. Non Dfs Used

```
# NOTE NON DFS USED
r = re.compile("Non DFS Used.*")
mylist = data
newlist = list(filter(r.match, mylist))
index = data.index(newlist[0])
g = Gauge('non_dfs_used_1', 'non used dfs', ['set_nondfsu'], registry=registry)
for x in range(len(newlist)):
    index = data.index(newlist[x])
```

```
data1 = data[index]
spl = data1.split()
spl = spl[3]
g.labels(set_nondfsu=x).set(spl)
print('non dfs used nodes :',spl,"\n")
x += 1
```

Python *script* diatas akan mengirimkan metrik Non DFS Used dari Hadoop kepada Prometheus dengan menggunakan Prometheus pushgateway. Nama metrik yang didorong(*push*) yaitu `non_dfs_used_1`.

5.1.1.4 Tampilan Prometheus

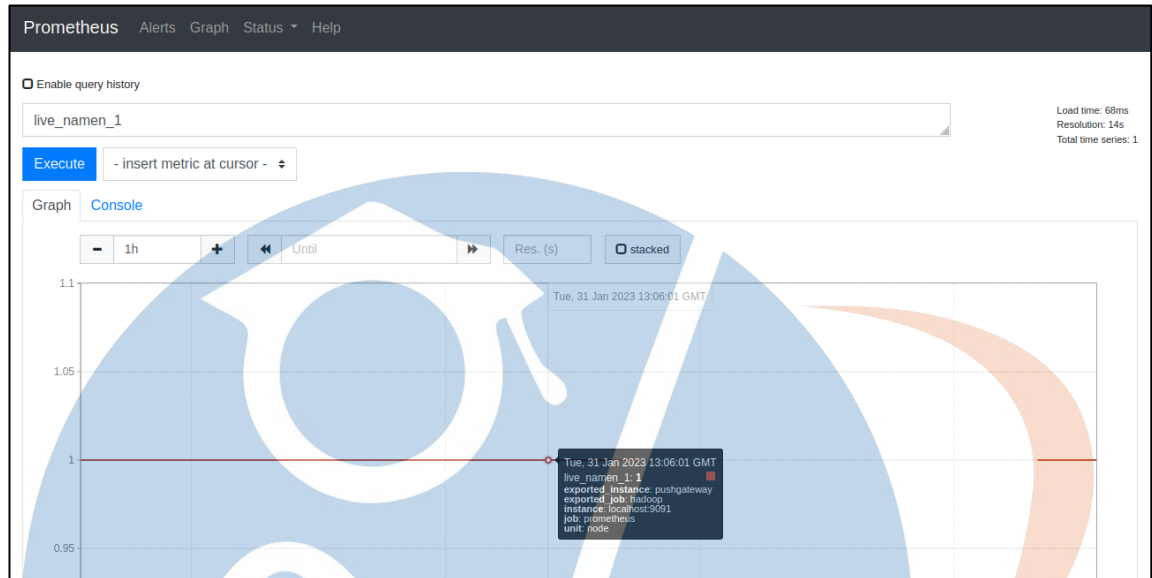
Pada tahap ini akan menjelaskan tentang data metrik Hadoop apa saja yang akan ditampilkan pada Prometheus setelah melakukan *push* pada Prometheus Gateway. Data metrik yang akan ditampilkan dalam bentuk *graph* dan juga *console*. Untuk data metrik Hadoop yang sudah di *push* dan akan ditampilkan pada Grafana diantaranya sebagai berikut:

1. Live Namenode

Matrik yang akan ditampilkan pada Prometheus mengikuti dengan metrik yang telah di *push* pada Prometheus Gateway. Untuk mendapatkan data metrik Live Namenode masukan nama metrik yang sebelumnya sudah dibuat

STT - NF

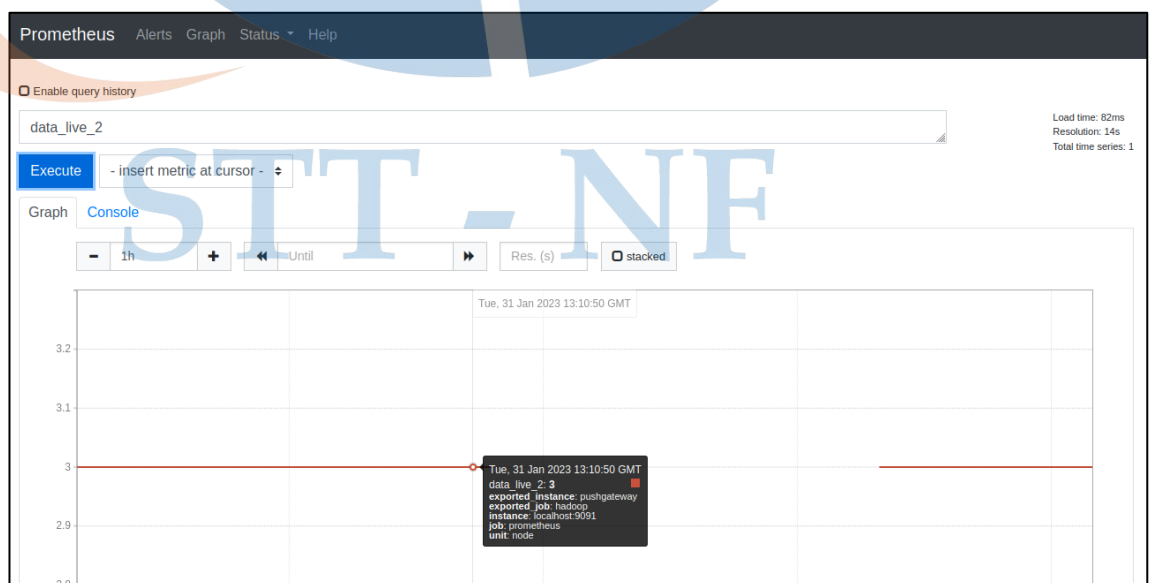
yaitu `live_namen_1`. Data yang ditampilkan adalah jumlah Namenode yang hidup atau aktif pada komputer.



Gambar 5. 2 Prometheus Namenode

2. Live Datanode

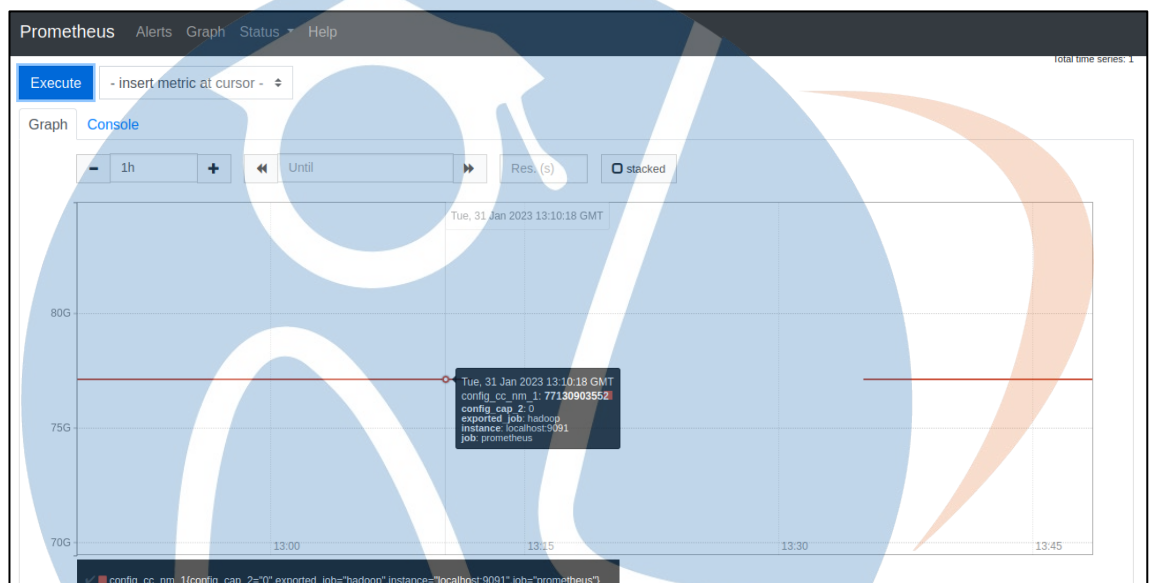
Selanjutnya untuk mendapatkan data metrik Live Datanode masukan nama metrik yang sebelumnya sudah dibuat yaitu `data_live_2`. Setelah memasukan metrik tersebut Prometheus akan menampilkan data metrik jumlah Datanode yang hidup.



Gambar 5. 3 Prometheus Datanode

3. Configure Capacity Total

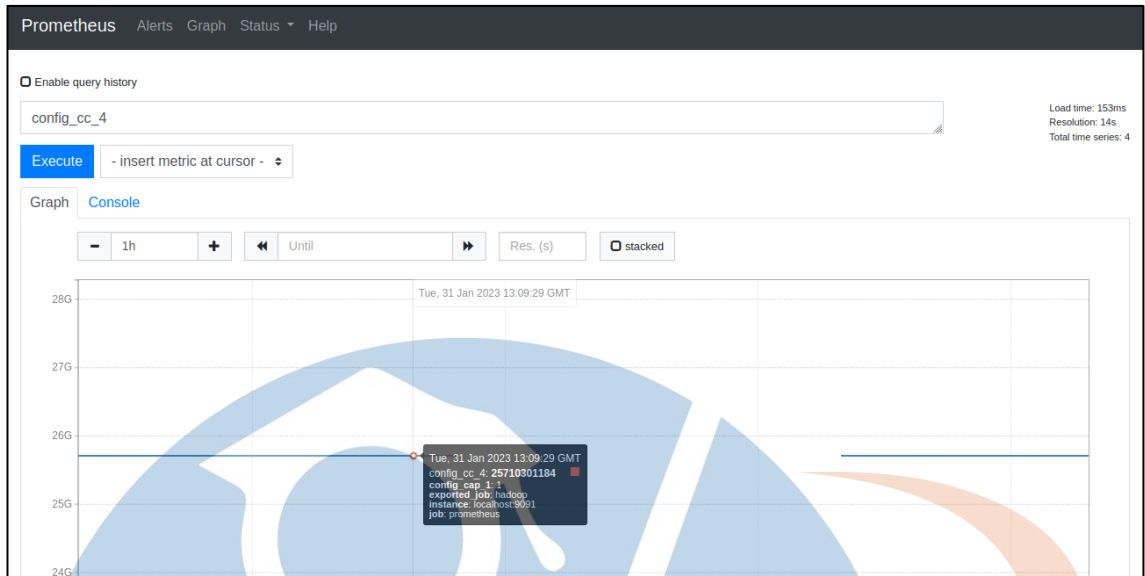
Selanjutnya untuk mendapatkan data metrik Configure Capacity Total masukan nama metrik yang sebelumnya sudah dibuat yaitu `config_cc_nm_1`. Setelah memasukan metrik tersebut Prometheus akan menampilkan data metrik jumlah total configure capacity yang di atur.



Gambar 5. 4 Prometheus Configure Capacity Total

4. Configure Capacity Per Node

Selanjutnya untuk mendapatkan data metrik Configure Capacity per node masukan nama metrik yang sebelumnya sudah dibuat yaitu `config_cc_4`. Setelah memasukan metrik tersebut Prometheus akan menampilkan data metrik jumlah configure capacity yang di atur pada masing-masing node.



Gambar 5. 5 Prometheus Configure Capacity Per Node

5. Present Capacity

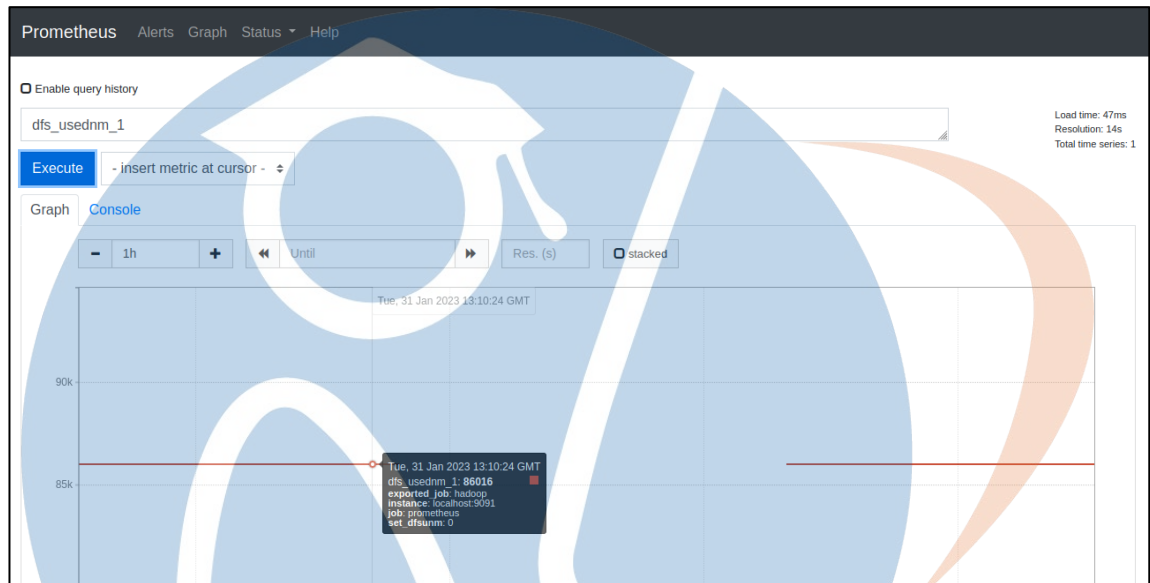
Selanjutnya untuk mendapatkan data metrik Present Capacity masukan nama metrik yang sebelumnya sudah dibuat yaitu pre_capacity_2. Setelah memasukan metrik tersebut Prometheus akan menampilkan data metrik jumlah total present capacity saat ini.



Gambar 5. 6 Prometheus Present Capacity

6. Dfs Used Total

Selanjutnya untuk mendapatkan data metrik DFS Used Total masukan nama metrik yang sebelumnya sudah dibuat yaitu `dfs_usednm_1`. Setelah memasukan metrik tersebut Prometheus akan menampilkan data metrik jumlah total DFS Used yang saat ini digunakan.

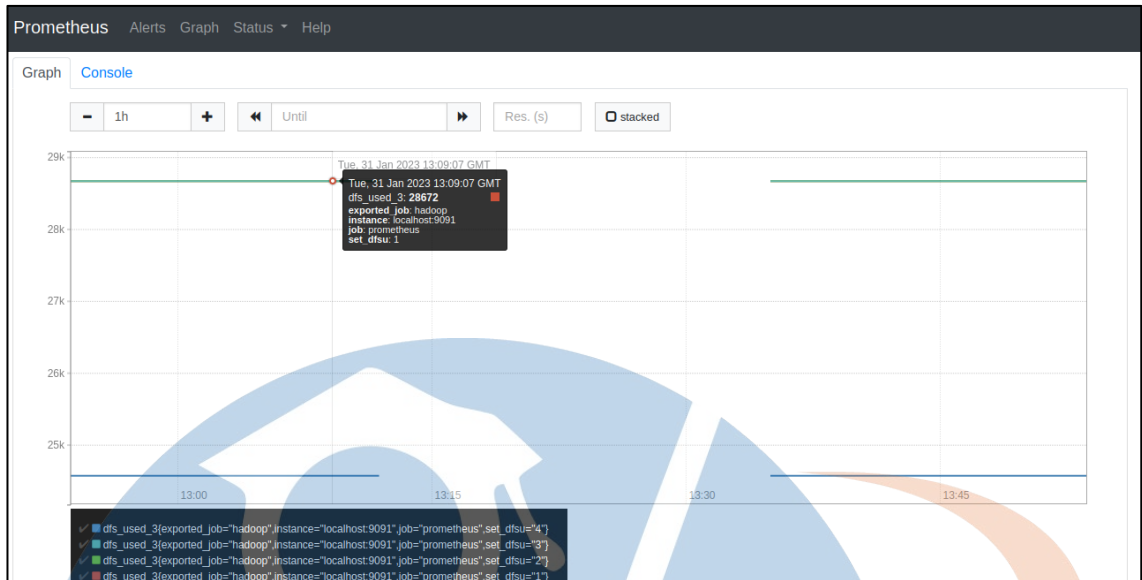


Gambar 5. 7 Prometheus DFS Used Total

7. Dfs Used Per Node

Selanjutnya untuk mendapatkan data metrik DFS Used per Node masukan nama metrik yang sebelumnya sudah dibuat yaitu `dfs_used_3`. Setelah memasukan metrik tersebut Prometheus akan menampilkan data metrik jumlah dfs used dari masing-masing node.

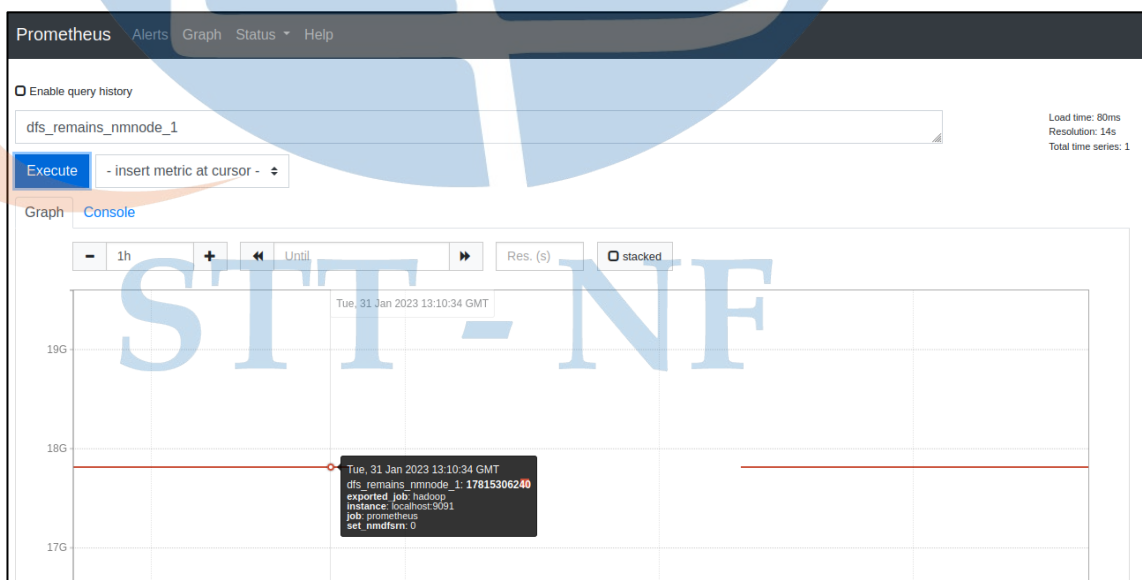
STT - NF



Gambar 5. 8 Prometheus DFS Used per Node

8. Dfs Remains Total

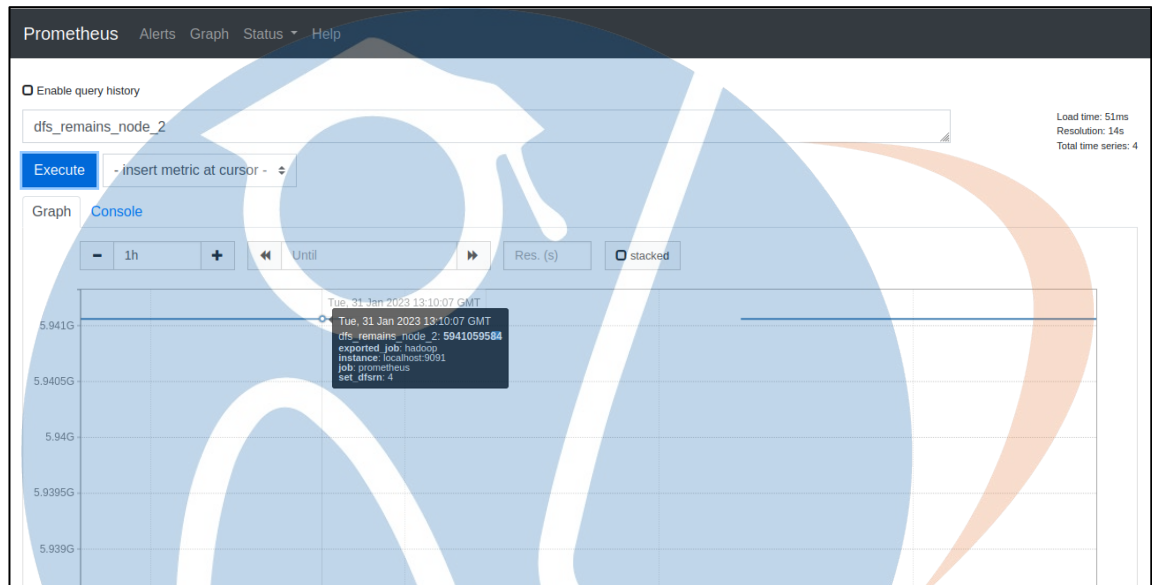
Selanjutnya untuk mendapatkan data metrik DFS Remains Total masukan nama metrik yang sebelumnya sudah dibuat yaitu `dfs_remains_nmnode_1`. Setelah memasukan metrik tersebut Prometheus akan menampilkan data metrik jumlah dfs remains.



Gambar 5. 9 Prometheus DFS Remains Total

9. Dfs Remains Per Node

Selanjutnya untuk mendapatkan data metrik DFS Remains Per Node masukan nama metrik yang sebelumnya sudah dibuat yaitu `dfs_remains_node_2`. Setelah memasukan metrik tersebut Prometheus akan menampilkan data metrik jumlah DFS Remains dari masing-masing Node.

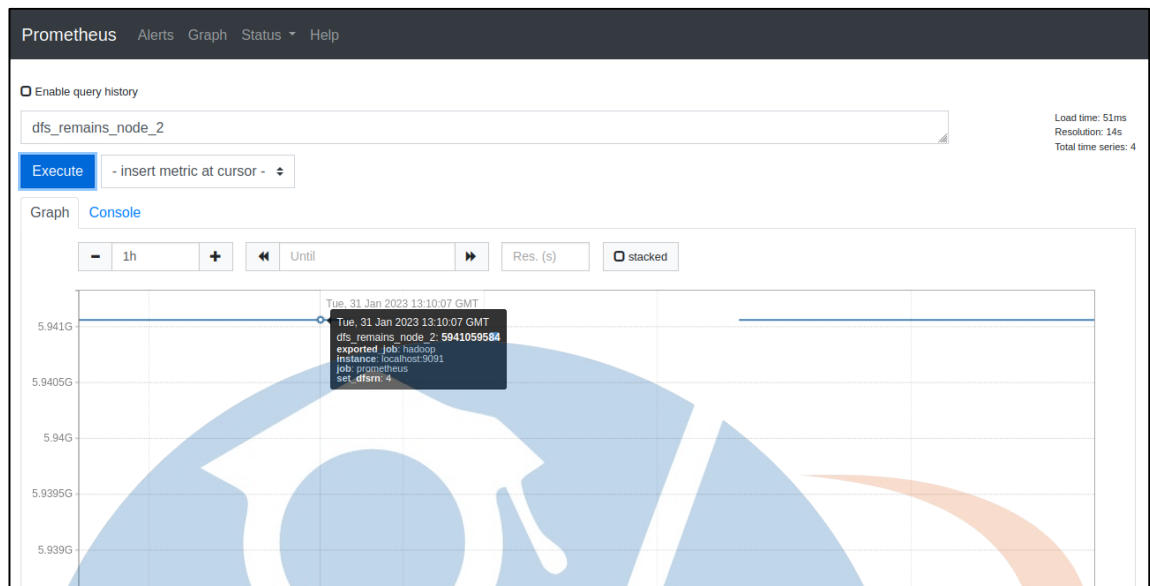


Gambar 5. 10 Prometheus DFS Remains Per Node

10. Non Dfs Used

Selanjutnya untuk mendapatkan data metrik Non DFS Used masukan nama metrik yang sebelumnya sudah dibuat yaitu `non_dfs_used_1`. Setelah memasukan metrik tersebut Prometheus akan menampilkan data metrik jumlah non DFS Used yang digunakan.

STT - NF



Gambar 5. 11 Prometheus Non DFS Used

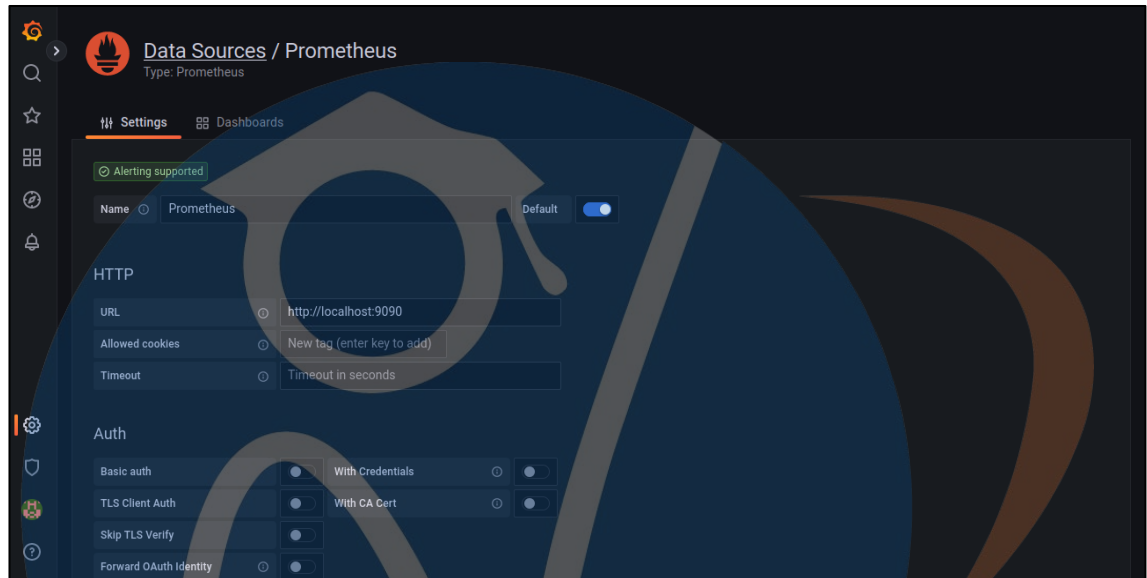
Setelah dapat menampilkan seluruh data metrik pada Hadoop menandakan bahwa Hadoop dan Prometheus sudah terkoneksi dengan baik. Data yang ditampilkan pada Prometheus merupakan data metrik dalam bentuk time series.

5.1.2 Implementasi pembuatan dashboard

Pada tahap ini akan menjelaskan bagaimana implementasi pembuatan dashboard monitoring Hadoop menggunakan Grafana. Data metrik yang akan ditampilkan pada Grafana adalah data yang bersumber dari Prometheus. Berbeda dengan Prometheus yang hanya bisa menampilkan data dalam bentuk *time series*, Grafana dapat menampilkan data dalam bentuk table. Sehingga memiliki fleksibilitas yang lebih memadai.

Hal pertama yang harus dilakukan adalah mengatur sumber data (*Data Source*) pada Grafana kepada Prometheus agar metrik-metrik Hadoop yang dimiliki oleh

Prometheus bisa ditampilkan pada Grafana. Dengan Grafana juga dapat merubah data yang bentuknya Time series menjadi berbagai macam *chart*.

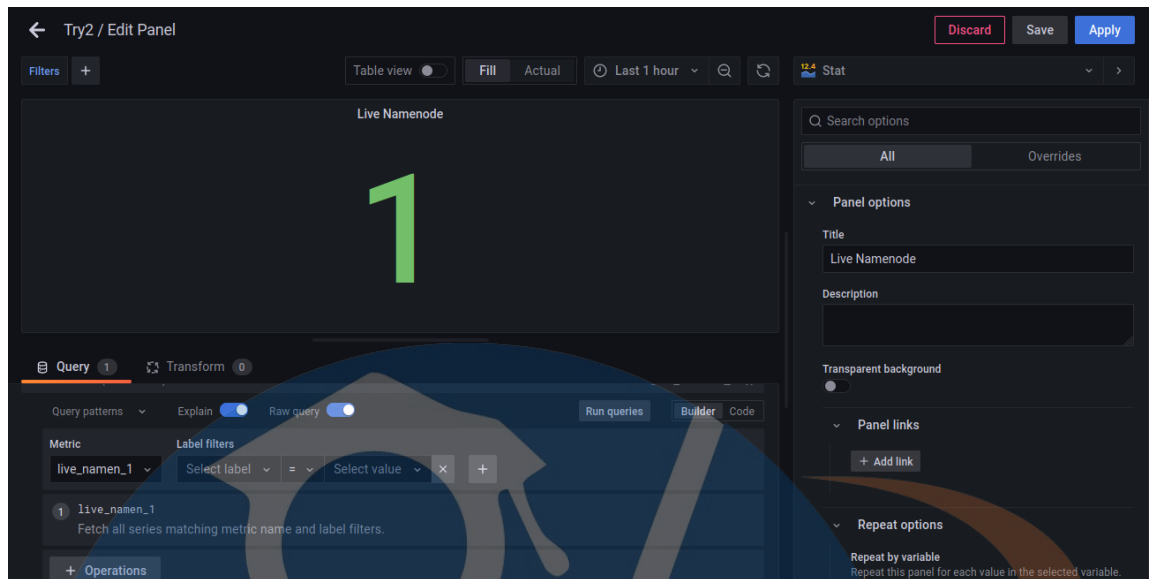


Gambar 5. 12 Grafana Data Source

Setelah mengatur sumber data (*Data Source*) hal yang perlu dilakukan adalah menampilkan metrik-metrik Hadoop dengan tampilan yang menyesuaikan dengan kebutuhan dan lebih mudah untuk dipahami. Berikut adalah metrik-metrik Hadoop yang akan ditampilkan pada Grafana:

1. Live Namenode

Pertama penulis akan menampilkan data metrik live Namenode dengan memasukan metrik `live_namen_1` sesuai dengan yang ada pada Prometheus. Data metrik akan disuguhkan dalam bentuk Statistik, agar dapat terlihat jelas namenode yang sedang aktif.



Gambar 5. 13 Grafana Live Namenode

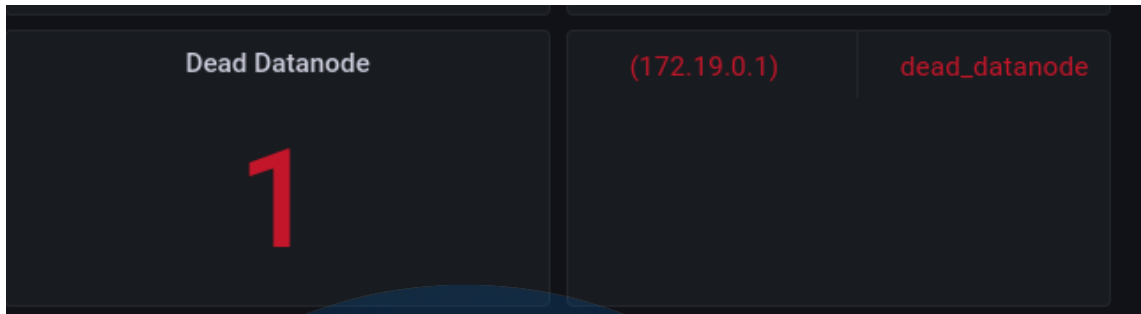
2. Live Datanode

Kedua penulis akan menampilkan data metrik Live Datanode dengan memasukan metrik data_live_2. Data metrik akan disuguhkan dalam bentuk *table*, agar dapat terlihat dengan jelas berapa jumlah datanode yang sedang aktif.

Live Datanode	IP	Status
	(172.19.0.2)	live_datanode
	(172.19.0.6)	live_datanode
	(172.19.0.7)	live_datanode

Gambar 5. 14 Grafana Live Datanode

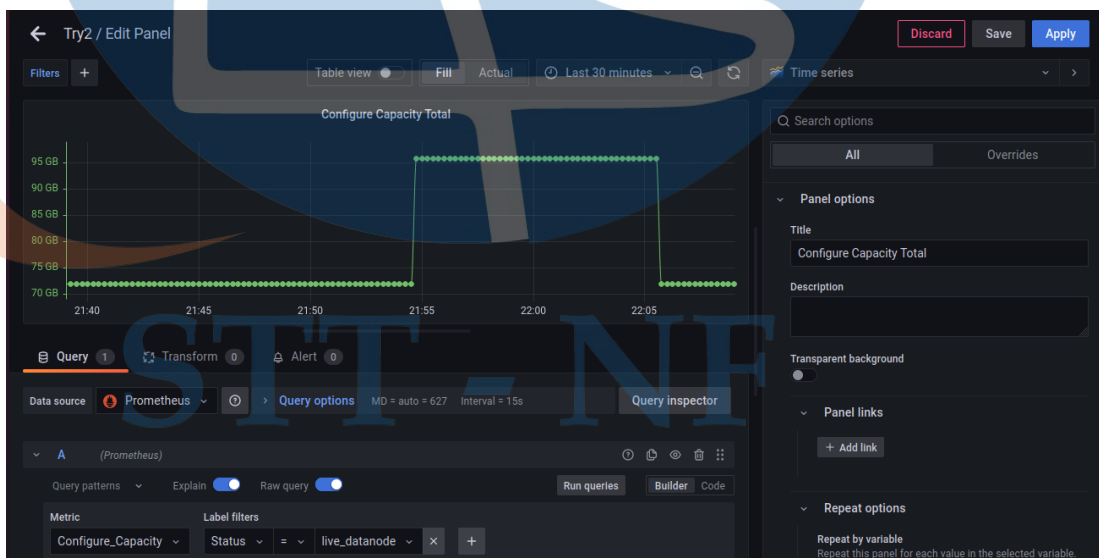
Namun bukan hanya datanode yang aktif saja yang akan ditampilkan pada panel ini tetapi juga akan menampilkan jumlah datanode yang tidak aktif. Dengan menggunakan *table* ini dapat dengan mudah melakukan perbandingan antara banyaknya datanode yang aktif dan juga banyaknya data node yang tidak aktif dengan menampilkan nama dan jumlahnya.



Gambar 5. 15 Grafana Live & Dead Datanode

3. Configure Capacity Total

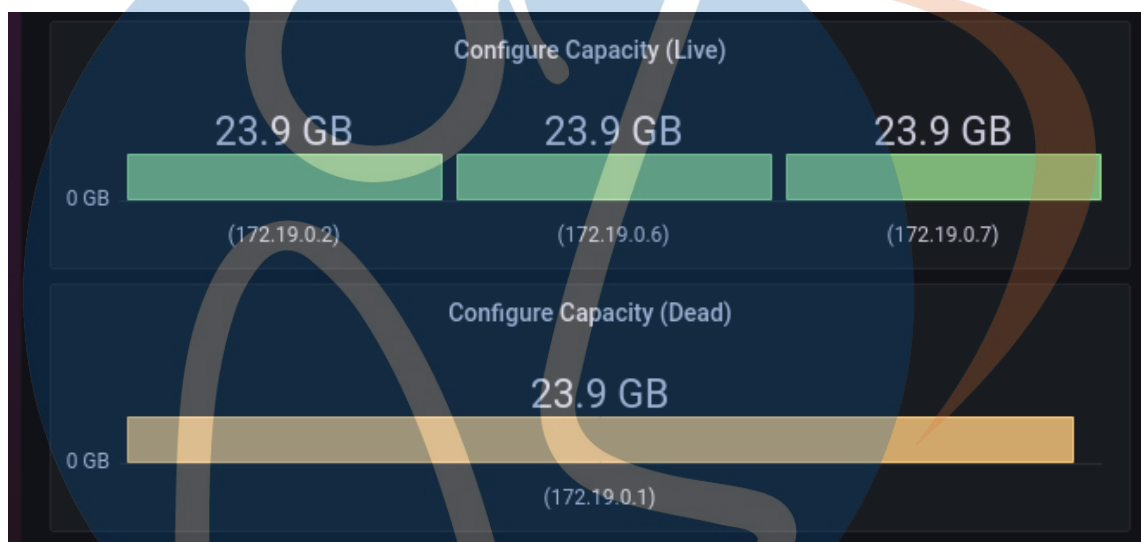
Ketiga penulis akan menampilkan data metrik Configure Capacity total atau merupakan hasil dari penjumlahan configure capacity yang di atur pada masing-masing node yang aktif dengan memasukan metrik `config_cc_nm_1`, metrik yang diberikan oleh Prometheus masih dalam ukuran KB sehingga penulis mengkonversinya menjadi ukuran GB agar lebih mudah untuk dilihat. Data metrik ini akan disuguhkan menggunakan *time series* agar dapat memonitor dalam jangka waktu berkala.



Gambar 5. 16 Grafana Configure Capacity Total

4. Configure Capacity Per Node

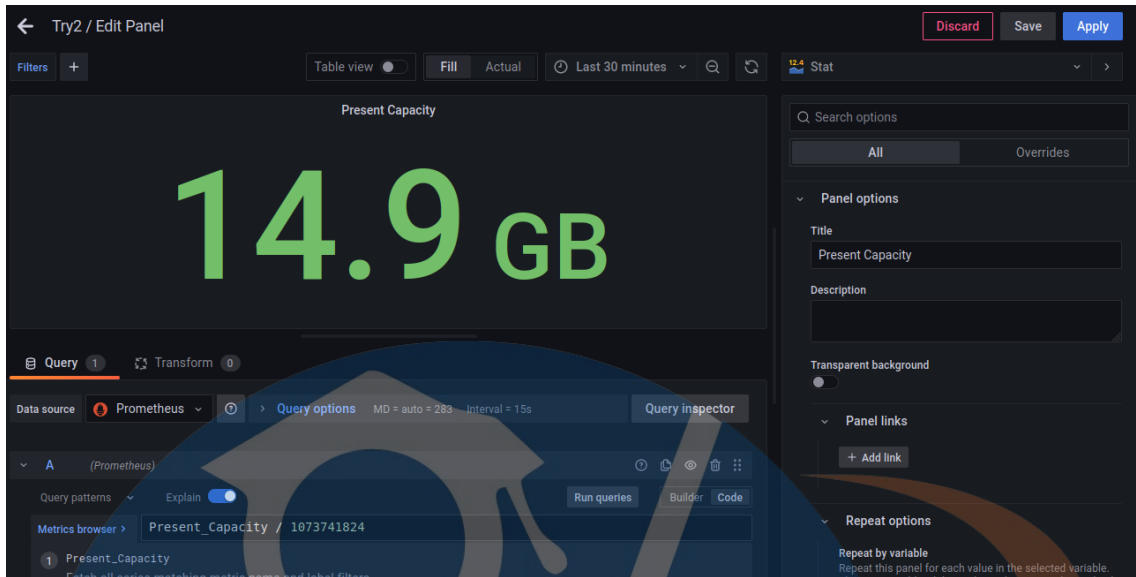
Keempat penulis akan menampilkan data metrik Configure Capacity Per node atau ukuran kapasitas yang diatur pada masing-masing node dengan memasukan metrik `config_cc_4` pada Grafana, metrik yang diberikan Prometheus ini masih dalam satuan KB sehingga penulis mengkonversinya dalam bentuk GB sehingga lebih mudah untuk dilihat. Data metrik ini akan ditampilkan dengan menggunakan *Bar Chart*, agar mudah membandingkan ukuran penyimpanan node mana yang lebih besar ataupun lebih kecil.



Gambar 5. 17 Grafana Configure Capacity per Node

5. Present Capacity

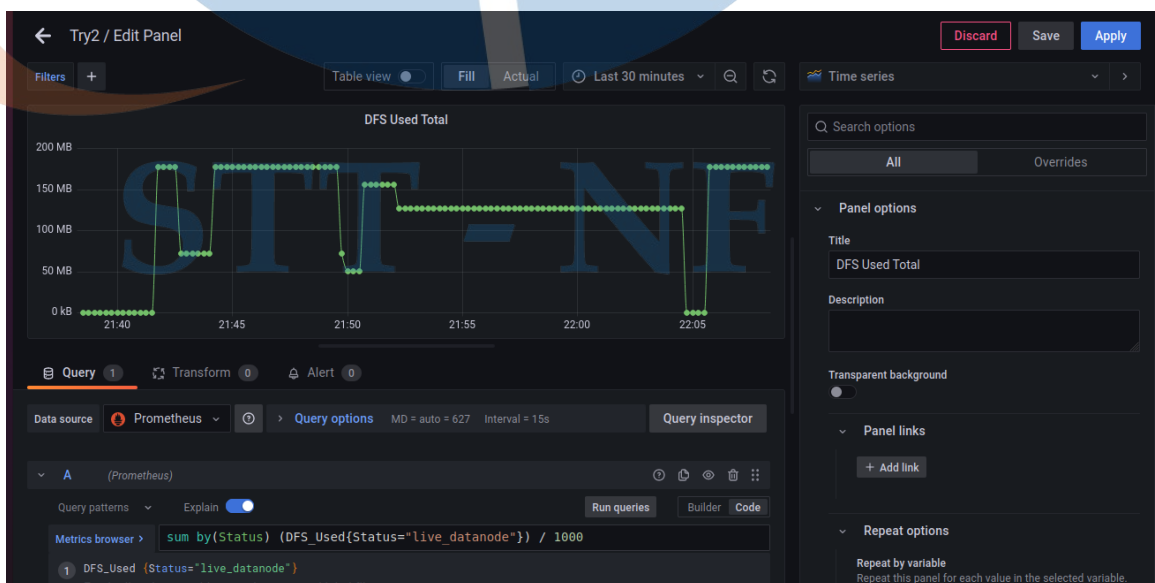
Kelima penulis akan menampilkan Present capacity atau penyimpanan terkini yang terdapat pada disk dengan memasukan data metrik `pre_capacity_2`. Data metrik ini akan ditampilkan dalam bentuk *statistik*, agar dapat mengukur besaran data.



Gambar 5. 18 Grafana Present Capacity

6. Dfs Used Total

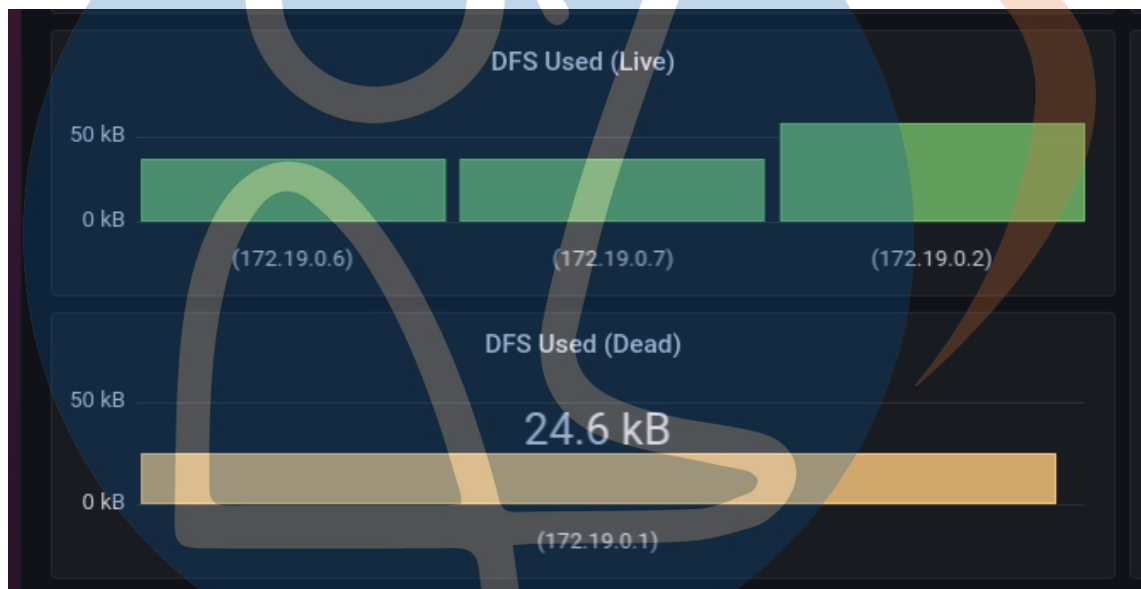
Keenam penulis akan menampilkan data metrik DFS Used Total atau merupakan hasil dari penjumlahan DFS Used Total yang di atur pada masing-masing node yang aktif dengan memasukan metrik `dfs_usednm_1`, metrik yang diberikan Prometheus ini masih dalam satuan KB sehingga penulis mengkonversinya dalam bentuk GB sehingga lebih mudah untuk dilihat. Data metrik ini akan disuguhkan menggunakan *time series* agar dapat memonitor dalam jangka waktu berkala.



Gambar 5. 19 Grafana DFS Used Total

7. Dfs Used Per Node

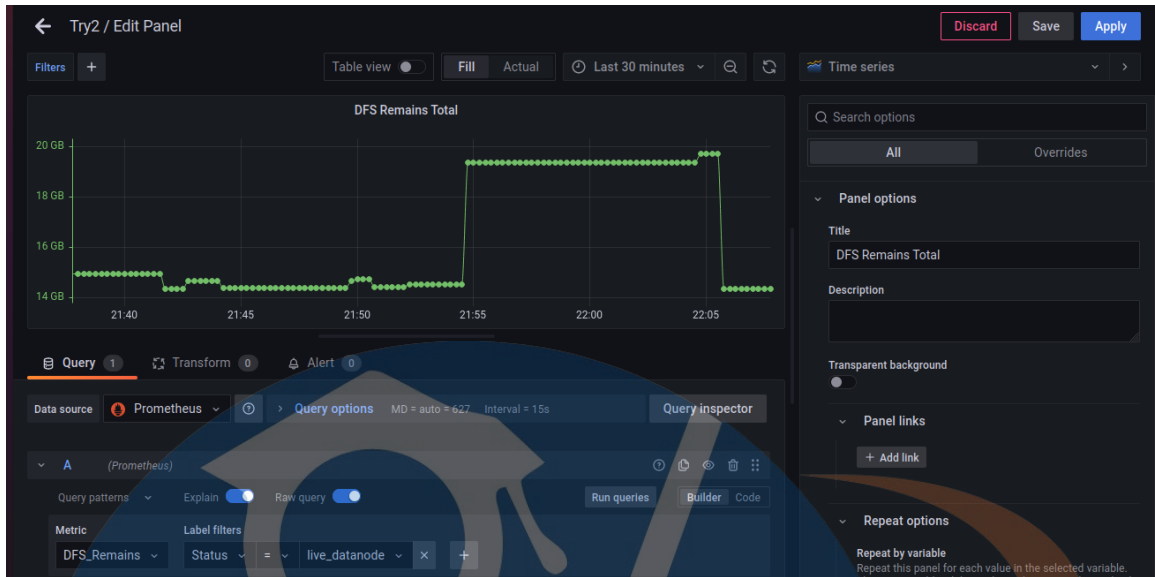
Ketujuh penulis akan menampilkan data metrik DFS Used Per node atau ukuran penggunaan DFS yang diatur pada masing-masing node dengan memasukan metrik `dfs_used_3` pada Grafana, metrik yang diberikan Prometheus ini masih dalam satuan KB sehingga penulis mengkonversinya dalam bentuk GB agar lebih mudah untuk dilihat. Data metrik ini akan ditampilkan dengan menggunakan *Bar Chart*, agar mudah membandingkan ukuran penyimpanan DFS mana yang lebih besar ataupun lebih kecil.



Gambar 5. 20 Grafana DFS Used Per node

8. Dfs Remains Total

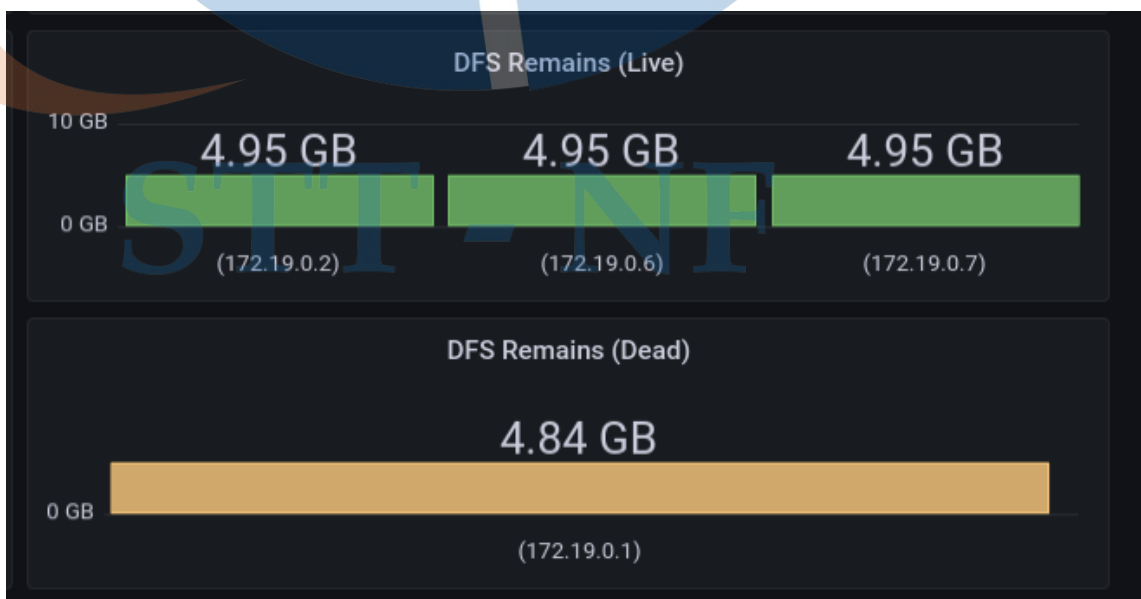
Kedelapan penulis akan menampilkan data metrik DFS Remains Total atau merupakan hasil dari penjumlahan DFS Remains yang di atur pada masing-masing node yang aktif dengan memasukan metrik `dfs_remain_nmnode_1`, metrik yang diberikan Prometheus ini masih dalam satuan KB sehingga penulis mengkonversinya dalam bentuk GB sehingga lebih mudah untuk dilihat. Data metrik ini akan disuguhkan menggunakan *time series* agar dapat memonitor dalam jangka waktu berkala



Gambar 5. 21 Grafana DFS Remains Total

9. Dfs Remains Per Node

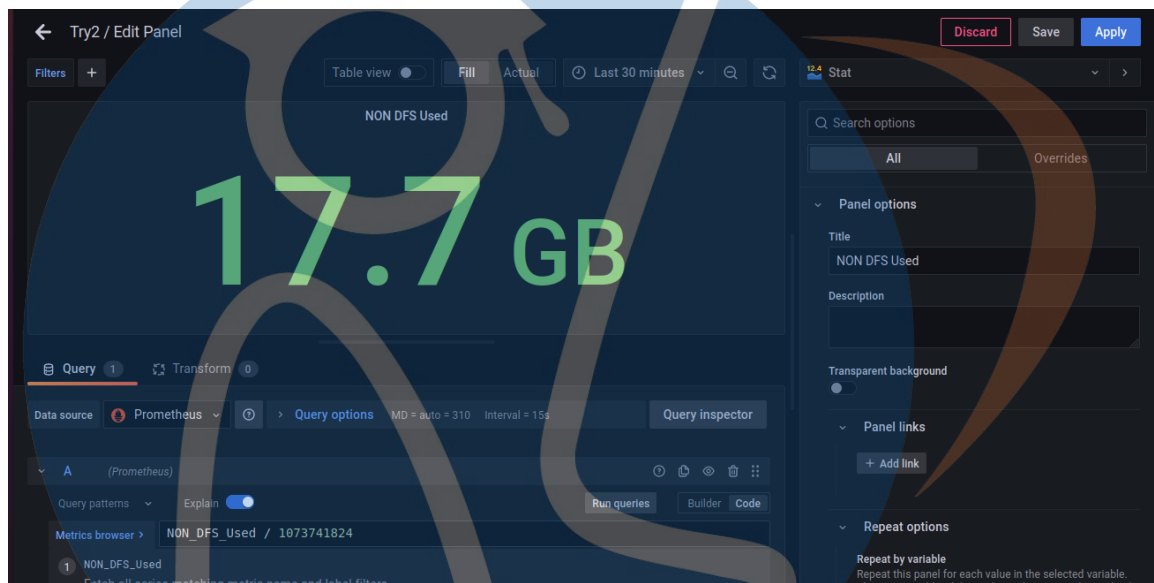
Kesembulan penulis akan menampilkan data metrik DFS Remains Per node atau ukuran penggunaan DFS yang digunakan pada masing-masing node dengan memasukan metrik `dfs_remains_node_2` pada Grafana, metrik yang diberikan Prometheus ini masih dalam satuan KB sehingga penulis mengkonversinya dalam bentuk GB agar lebih mudah untuk dilihat. Data metrik ini akan ditampilkan dengan menggunakan *Bar Chart*, agar mudah membandingkan ukuran penyimpanan DFS mana yang lebih besar ataupun lebih kecil.



Gambar 5. 22 Grafana DFS Remains per Node

10. Non Dfs Used

Kesepuluh penulis akan menampilkan data metrik NON DFS Used Per node atau ukuran penggunaan NON DFS yang diatur pada masing-masing node dengan memasukan metrik `non_dfs_used_1` pada Grafana, metrik yang diberikan Prometheus ini masih dalam satuan KB sehingga penulis mengkonversinya dalam bentuk GB agar lebih mudah untuk dilihat Data metrik ini akan ditampilkan dalam bentuk *statistik*, agar dapat mengukur besaran data.



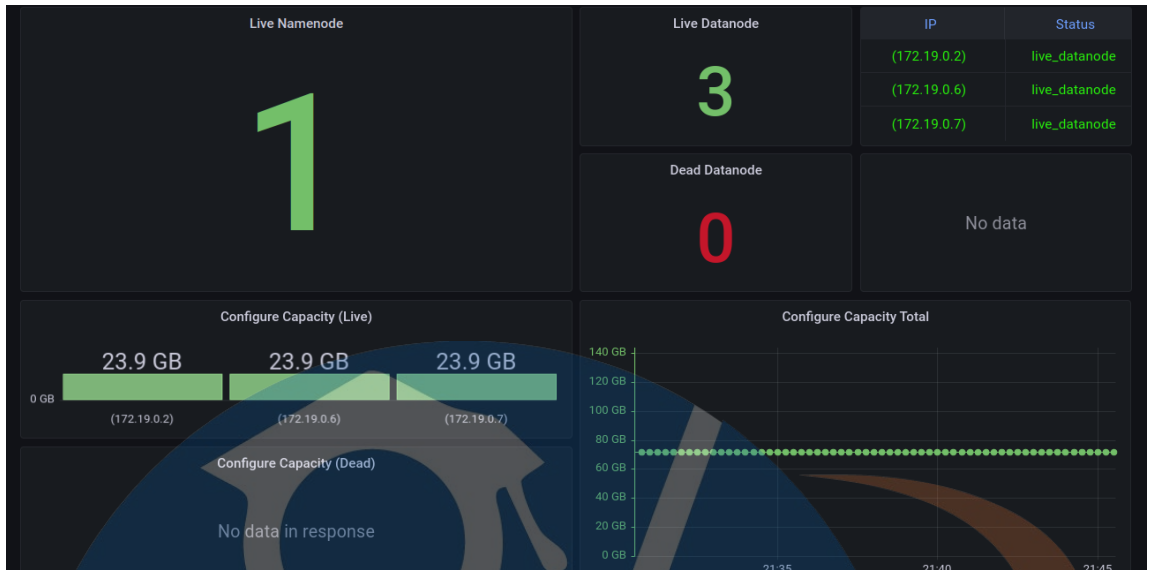
Gambar 5. 23 Grafana NON DFS Used

11. Tampilan keseluruhan

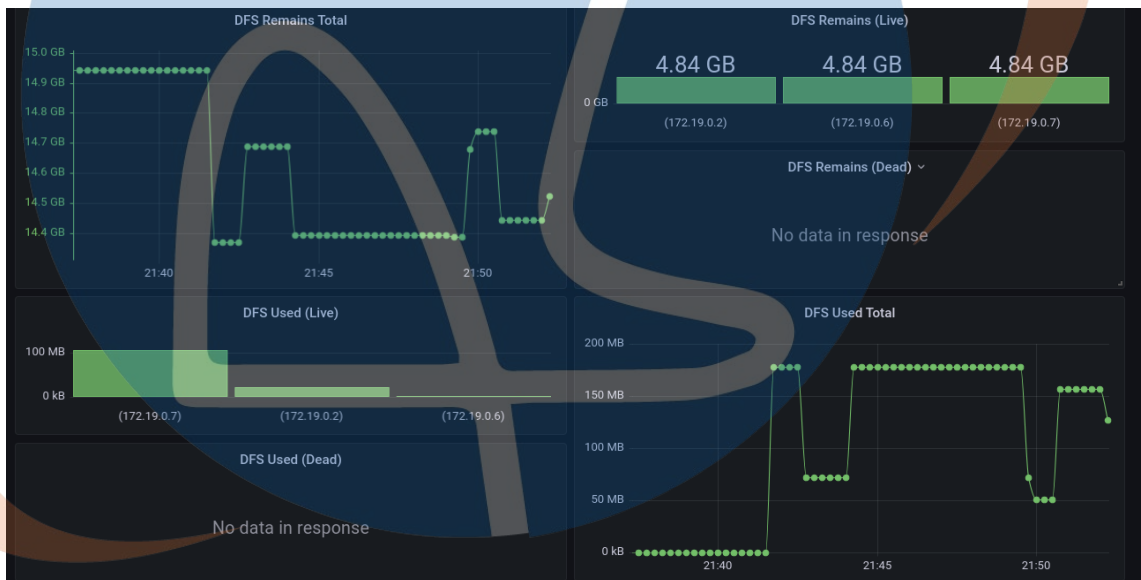
Secara keseluruhan tampilan dari metrik yang dikirimkan oleh Prometheus bisa ditampilkan dengan baik pada Grafana dengan berbagai macam *chart* yang mengikuti kebutuhan pengguna dan juga kondisi pada Hadoop.

a. Seluruh Datanode hidup

Pada tampilan ini terlihat keseluruhan datanode hidup, dan data yang ditampilkan adalah satu Namenode (localhost) dan Juga tiga datanode yang wakikan dengan Ip address.

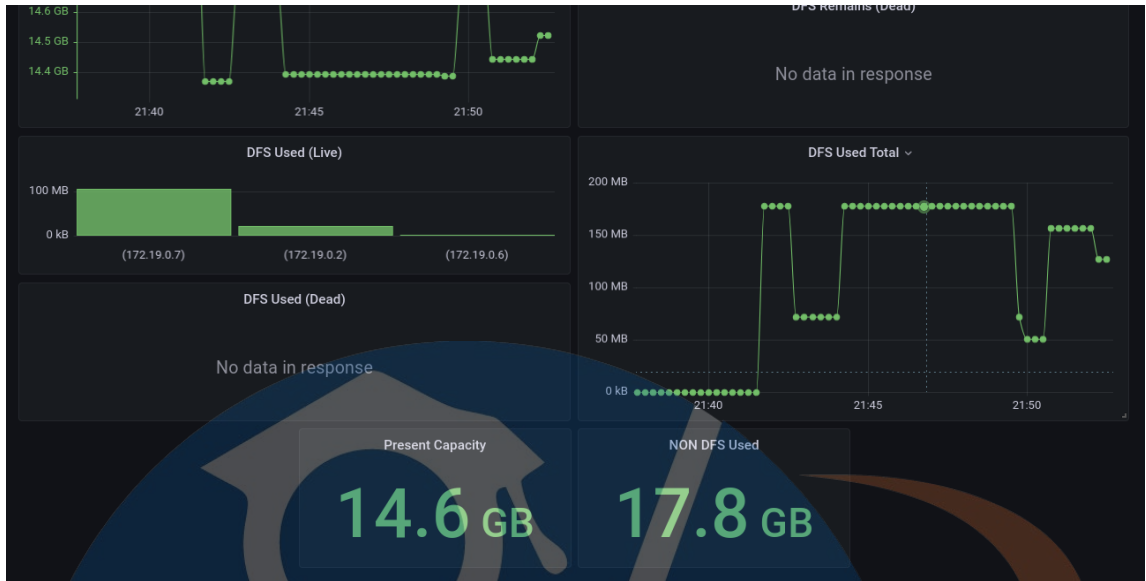


Gambar 5. 24 Grafana Live 1



Gambar 5. 25 Grafana Live 2

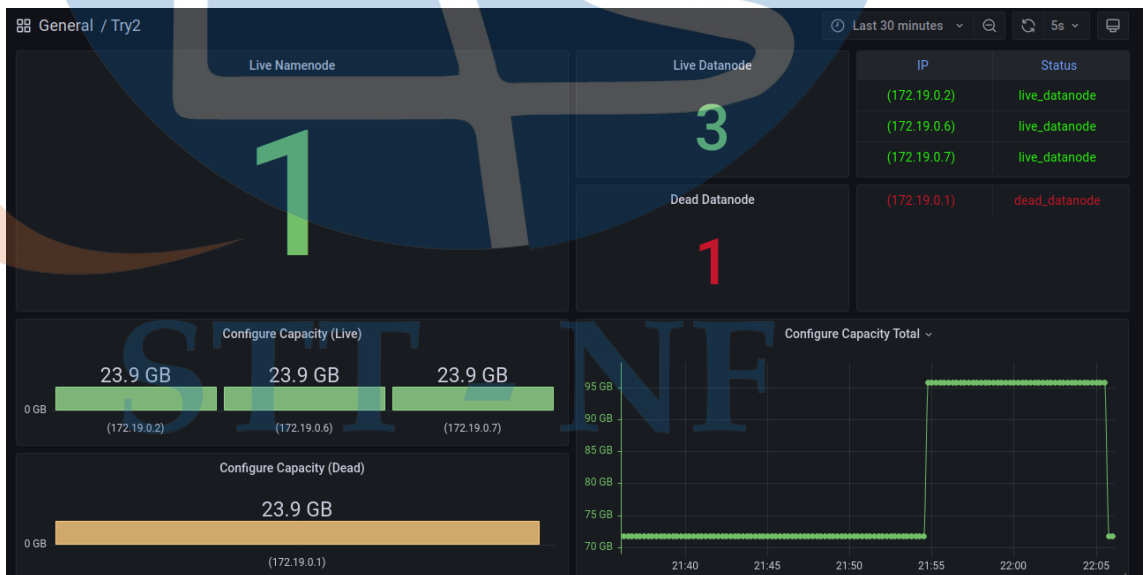
STT - NF



Gambar 5. 26 Grafana Live 3

b. Terdapat dead Datanode

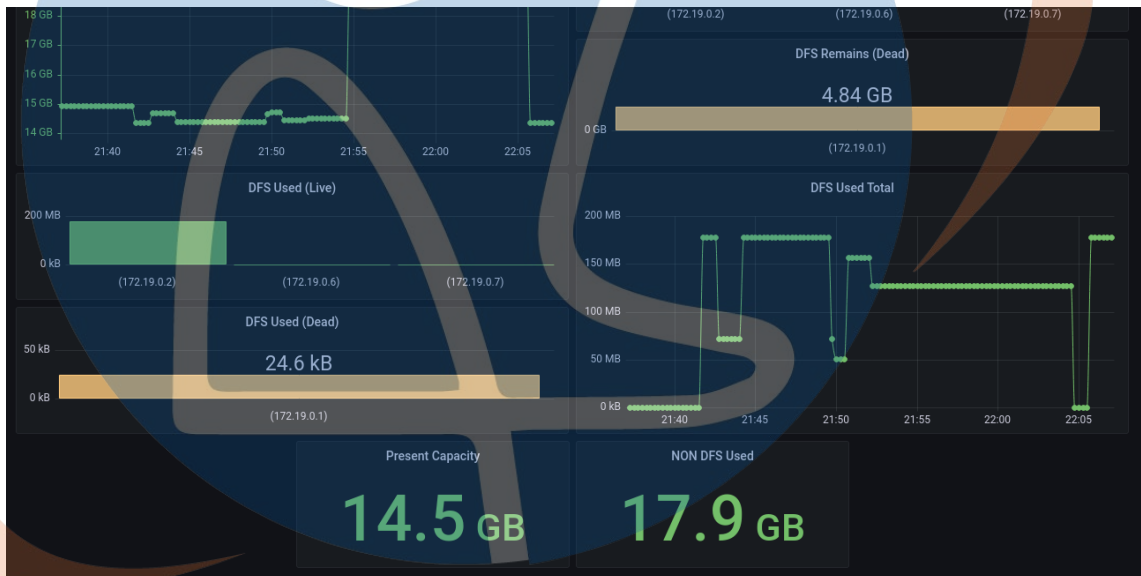
Pada tampilan ini terlihat terdapat datanode yang tidak aktif sehingga tampilan pada Grafana mengikuti dengan perkembangan data pada Hadoop, dan data yang ditampilkan adalah satu Namenode (localhost) dan Juga empat datanode yang wakilkkan dengan Ip address.



Gambar 5. 27 Grafana Live & Dead 1



Gambar 5. 28 Grafana Live & Dead 2



Gambar 5. 29 Grafana Live & Dead 3

5.2 Implementasi Pengujian

Pada tahap ini akan dilakukan implementasi pengujian terhadap perancangan dashboard monitoring Hadoop menggunakan Grafana. Pada pengujian ini akan dilakukan dua macam pengujian yaitu pengujian arsitektur dan pengujian dashboard monitoring yang telah dibuat sebelumnya.

5.2.1 Pengujian Arsitektur

Pada tahap ini akan menjelaskan implementasi pengujian arsitektur perancangan dashboard monitoring Hadoop menggunakan Grafana. Berikut tabel yang akan menjelaskan tentang implementasi pengujian arsitektur:

Table 5. 1 Implementasi pengujian arsitektur

No	Pengujian	Deskripsi pengujian	Hasil yang diharapkan	Hasil Uji
1	Membuat Datanode	Membuat tiga datanode aktif dengan menggunakan docker	Berhasil membuat tiga datanode aktif	Berhasil
2	Mengatur target pada Prometheus	Merubah target pada Prometheus agar dapat mengakses Prometheus pushgateway	Berhasil mengatur target Prometheus dengan pushgateway	Berhasil
3	Membuat koneksi	Membuat docker container Prometheus, Prometheus Pushgateway, dan grafana	Berhasil menjalankan Prometheus, Prometheus push gateway, dan Grafana	Berhasil
4	Membuat script	Membuat script python untuk menarik data Hadoop	Berhasil menarik data Hadoop	Berhasil
5	Mengumpulkan data	Mengumpulkan data Hadoop yang akan di gunakan	Berhasil mengumpulkan data Hadoop	Berhasil
6	Mengirim data ke Prometheus	Mengirim data Hadoop dengan	Berhasil mengirim data Hadoop	Berhasil

		Prometheus push Gateway		
7	Menampilkan data pada Prometheus	Menampilkan metrik Hadoop pada Prometheus	Berhasil menampilkan data Hadoop pada Grafana	Berhasil
8	Menghubungkan ke Grafana	Membuat sumber data Prometheus pada Grafana	Berhasil menghubungkan Prometheus dengan Grafana	Berhasil
9	Menerima data dari Prometheus	Menampilkan data yang terdapat pada Prometheus	Berhasil menampilkan data Prometheus	Berhasil
10	Menampilkan data	Menampilkan data metrik Hadoop pada Grafana	Berhasil membuat dashboard monitoring Hadoop dengan Grafana	Berhasil

5.2.2 Pengujian Monitoring

Pada tahap ini akan menjelaskan implementasi pengujian perancangan dashboard monitoring Hadoop menggunakan Grafana. Berikut tabel yang akan menjelaskan tentang implementasi pengujian monitoring:

Table 5. 2 Implementasi pengujian monitoring

NO	Pengujian	Deskripsi pengujian	Hasil yang diharapkan	Hasil uji
1	Live Namenode	Menampilkan metrik Live Namenode pada Grafana	Berhasil menampilkan Live Namenode	Berhasil
2	Live Datanode	Menampilkan metrik Live Datanode pada Grafana	Berhasil menampilkan Live Datanode Berhasil menampilkan Dead Datanode	Berhasil

3	Configure capacity Total	Menampilkan metrik Configure Capacity total pada Grafana	Berhasil menampilkan Configure Capacity Total	Berhasil
4	Configure capacity per Node	Menampilkan metrik Configure Capacity per Node pada Grafana	Berhasil menampilkan Configure Capacity per Node	Berhasil
5	Present Capacity	Menampilkan metrik Present Capacity pada Grafana	Berhasil menampilkan Present Capacity	Berhasil
6	DFS Used Total	Menampilkan Matrik DFS Used Total pada Grafana	Berhasil menampilkan DFS Used Total	Berhasil
7	DSF Used per Node	Menampilkan metrik DFS Used per Node pada Grafana	Berhasil menampilkan DFS Used per Node	Berhasil
8	DFS Remains Total	Menampilkan metrik DFS Remains Total pada Grafana	Berhasil menampilkan DFS Remains Total	Berhasil
9	DFS Remains per Node	Menampilkan Matrik DFS Remains per Node pada Grafana	Berhasil menampilkan DFS Remains per Node	Berhasil
10	Non DFS Used	Menampilkan metrik NON DFS Used pada Grafana	Berhasil menampilkan NON DFS Used	Berhasil