Prometheus e Grafana

Laboratorio di Reti

Filippo Poltronieri
filippo.poltronieri@unife.it
and credits to:
Luca Pasquali Evangelisti

1. Prometheus
    1.1. Introduzione
    1.2. Monitoraggio
    1.3. Metriche
    1.4. Funzionamento
    1.5. Architettura
    1.6. Exporters
2. Installazione Prometheus e Node-Exporter
3. Prometheus Web UI
4. Grafana
    4.1. Installazione
    4.2. Configurazione
5. Esempio Prometheus + Grafana
    5.1. CPU Usage
    5.2. Memory Usage
    5.3. Disk Usage
    5.4. Network Traffic

Obiettivo Laboratorio

Questo laboratorio ha lo scopo di realizzare un sistema di monitoraggio delle metriche base principali di una VM e una successiva realizzazione di Dashboard utili alla presentazione dei dati raccolti.

Gli strumenti utilizzati saranno i seguenti:

- Prometheus - strumento per la raccolta dei dati

- Grafana - strumento per la realizzazione di dashboards

Introduzione a Prometheus

Cos'è Prometheus?

Prometheus è un sistema di monitoraggio che viene utilizzato in scenari dinamici in cui è presente un elevato numero di elementi da controllare

Perchè viene utilizzato?

Le applicazioni moderne sono composte da centinaia di processi e servizi interconnessi che molto spesso si trovano su macchine differenti.

Prometheus permette il monitoraggio automatico e costante dei vari processi e macchine, in modo che al verificarsi di un problema, venga generato e inviato un allarme, facilitandone la sua individuazione e correzione.

Monitoraggio

- Gli elementi che si vogliono monitorare vengono chiamati ***target***. Alcuni esempi di *target*:
  - Linux/Windows Servers
  - Apache Server
  - Singole applicazioni
  - Database
- Ogni *target* possiede diverse caratteristiche che possono essere monitorate, queste prendono il nome di ***metriche***. Alcuni esempi di *metriche:*
  - Target: Servers
    - Stato della CPU
    - Utilizzo del disco
    - Utilizzo di RAM
  - Target: Applicazione
    - Numero di richieste
    - Durata richieste
    - Numero di eccezioni

Metriche

- **Metrica:** misura quantitativa di un determinato attributo
- **Tipi di metriche:**
  - **Counter** → quante volte è avvenuto un evento
  - **Gauge** → qual'è il valore attuale di una determinata metrica
  - **Histogram** → mostra la dimensione e la durata di una metrica
- Le metriche vengono salvate in un formato testuale *human-readable*. In cima ad ogni valore sono esposti due attributi utili alla descrizione della metrica esposta:
  - **HELP:** breve descrizione della metrica esposta
  - **TYPE:** indica il tipo di metrica rappresentata

```
# HELP node_cpu_guest_seconds_total Seconds the CPUs spent in guests (VMs) for each mode.
# TYPE node_cpu_guest_seconds_total counter
node_cpu_guest_seconds_total{cpu="0",mode="nice"} 0
node_cpu_guest_seconds_total{cpu="0",mode="user"} 0
# HELP node_cpu_seconds_total Seconds the CPUs spent in each mode.
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="0",mode="idle"} 129.77
node_cpu_seconds_total{cpu="0",mode="iowait"} 1.49
node_cpu_seconds_total{cpu="0",mode="irq"} 0
node_cpu_seconds_total{cpu="0",mode="nice"} 0.95
node_cpu_seconds_total{cpu="0",mode="softirq"} 0.67
node_cpu_seconds_total{cpu="0",mode="steal"} 0
node_cpu_seconds_total{cpu="0",mode="system"} 8.34
node_cpu_seconds_total{cpu="0",mode="user"} 6.34
# HELP node_disk_ata_write_cache ATA disk has a write cache.
# TYPE node_disk_ata_write_cache gauge
node_disk_ata_write_cache{device="sda"} 1
```

Esempio metriche

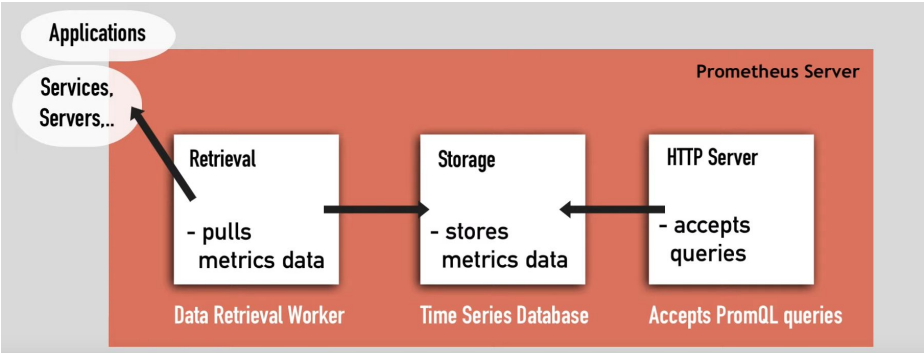
Funzionamento

- Monitoraggio eseguito secondo un modello **Pull**, allo scadere dell' intervallo specificato, Prometheus interroga i vari *targets* **richiedendo le *metriche* interessate**
- Recupero metriche:
  - **Prometheus Server:** entità che si occupa di eseguire il ***pull*** (la richiesta) delle metriche
  - Il *target* è l'elemento interrogato da Prometheus Server e fornisce le metriche richieste
- Per fornire le metriche in modo adatto, i *targets* devono o supportare nativamente le richieste Prometheus oppure devono aver installato un componente chiamato ***exporter***

Architettura Prometheus Server

Prometheus Server è l'elemento che si occupa del monitoraggio e quindi della raccolta delle *metriche*. Questo è composto da 3 parti:

1. **Time Series Database:** archivia i dati relativi alle ***metriche***.
2. **Recupero dei dati:** raccoglie le *metriche* dai *target* specificati
3. **HTTP Server:** Server che espone un'interfaccia HTTP del database contenente le metriche raccolte. E' possibile *interrogare* l'HTTP Server con strumenti terzi per ottenere i dati salvati nel database. Linguaggio di interrogazione: ***promql***



Exporters

- Un *exporter* è uno script che:
  - Recupera un insieme di statistiche da un sistema che non supporta Prometheus
  - Trasforma le statistiche recuperate in *metriche* compatibili con Prometheus
  - Realizza un web-server che espone le metriche recuperabili utilizzando specifici **URL** (endpoints)
- Gli ***endpoints*** nei quali vengono inserite le varie metriche sono specifici ***URL*** i quali se interrogati da richieste Prometheus forniscono i dati “contenuti”
- **URL** nella forma: *<target-address>/<metrics_name>*
- Esistono diversi tipi di *exporters* (più di 160), ognuno dei quali riguarda un insieme di statistiche differenti
- Come primo esempio, vedremo **Node Exporter**, il quale permette di estrapolare metriche a livello di Server e Sistema Operativo

Esempio Laboratorio

Di seguito, verrà mostrato un esempio di monitoraggio di 3 macchine virtuali. *Si può replicare l'esempio utilizzando anche una sola macchina o macchine fisiche differenti.*

VM Prometheus_Server: VM che **ospiterà Prometheus** e avrà lo scopo di eseguire il monitoraggio delle metriche delle due seguenti macchine virtuali

1. **VM Node_1:** questa VM rappresenta uno dei due nodi da monitorare. Per essere monitorata, questa macchina deve esporre le metriche relative all'utilizzo del suo hardware, per questo motivo verrà installato **Node-Exporter**

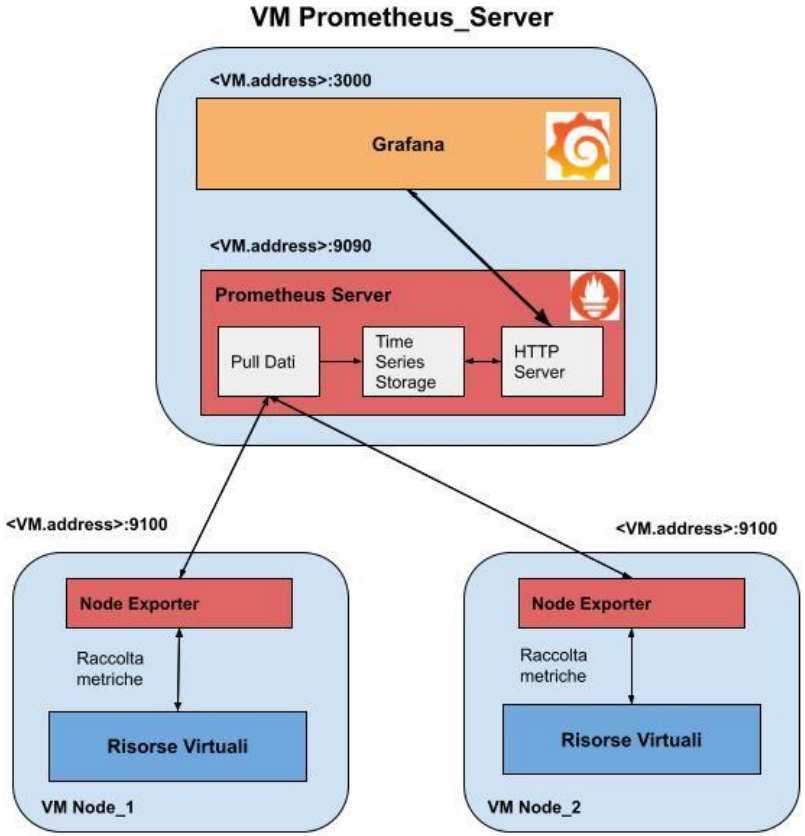
2. **VM Node_2:** anche questa VM dovrà essere monitorata da **VM Prometheus_Server**, per questo motivo anche su questa macchina deve essere installato **Node-Exporter**

Delle metriche raccolte dalle due VM Node, andremo a visualizzare le principali:

- Utilizzo della CPU
- Memory Usage (RAM)
- Disk Usage (disco fisso)
- Traffico di rete in entrata e uscita

Il tool che utilizzeremo per visualizzare le metriche raccolte sarà **Grafana**, il quale verrà installato sulla VM che ospita **Prometheus** (VM Prometheus_Server)

Esempio Laboratorio



Installazione Prometheus

Di seguito viene mostrato come installare Prometheus e Node-Exporter. Se seguiamo l'esempio nella slide precedente, bisogna tenere in considerazione che solo **UNA** macchina utilizza Prometheus mentre le **altre due** macchine utilizzano solo Node-Exporter

Prometheus sarà accessibile sulla porta ***:9090***

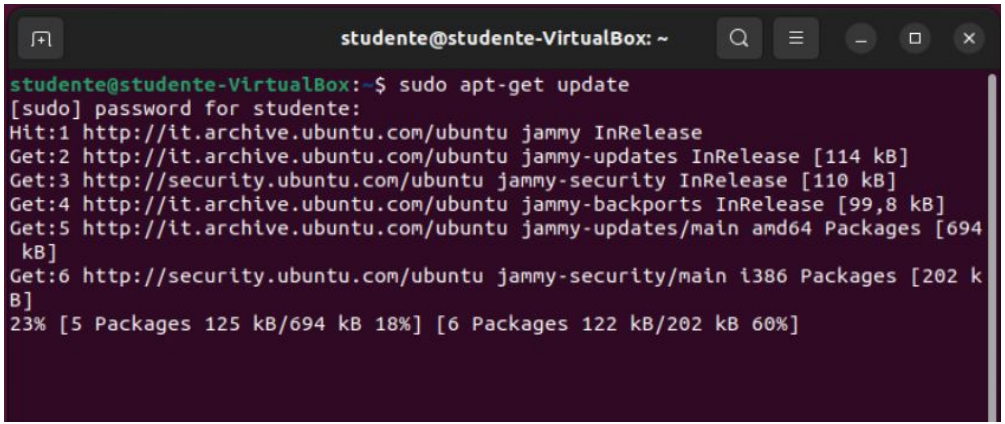
Node-Exporter sarà accessibile sulla porta ***:9100***

Installazione: aggiornamento dipendenze

- Per prima cosa è necessario aggiornare il sistema di dipendenze delle librerie di Linux

```
$ sudo apt-get update
```

- Attendere che l'aggiornamento termini



Installazione

- Una volta terminato è possibile utilizzare il seguente comando per effettuare l'installazione all-in-one

```
$ sudo apt -y install prometheus prometheus-node-exporter
```

Questa comprende

- Creazione dell'utente Prometheus e relativo User Group
- Creazione dell'utente Node-Exporter e relativo User Group
- Download ed estrazione dei file
- Creazione di tutte le cartelle necessarie

Per altre opzioni di installazione, si consulti https://prometheus.io/docs/prometheus/latest/installation/

Installazione - File di configurazione *prometheus.yml*

Una volta completata è possibile visualizzare e modificare il file configurazione di prometheus alla seguente path:

`/etc/prometheus/prometheus.yml`

Prima di far partire il Server Prometheus e il Node-Exporter, è bene guardare a fondo il file di configurazione di prometheus: *prometheus.yml*

In questo file sono presenti le regole che definiscono il monitoraggio, come quali metriche Prometheus deve raccogliere.

Installazione - File di configurazione *prometheus.yml*

- Queste regole serviranno a indicare a Prometheus:
  - Da quali macchine dovrà recuperare le metriche (ogni Host da monitorare verrà chiamato Job)
  - Le regole da utilizzare per attivare gli allarmi
  - Gli intervalli temporali con i quali:
    - richiedere le metriche ai targets definiti
    - valutare le regole definite

NB

- Il file *prometheus.yml* si trova nella cartella /etc/prometheus → */etc/prometheus/prometheus.yml*
- Il file di configurazione **deve essere modificato solo** nella VM che si utilizzerà per monitorare le altre (VM Prometheus_Server)

Installazione - File di configurazione *prometheus.yml*

In questo caso, suddividiamo in 3 parti il file *prometheus.yml:*

1. **Configurazioni globali**, definizione di due intervalli:
   a. *scrape_interval:* reperimento metriche
   b. *evaluation_interval:* valutazione delle regole

2. Regole per la generazione degli allarmi, vengono indicati in quali file le regole sono scritte

3. In questa sezione vengono definiti i ***jobs*** (identificati da **jobs_name**), all'interno dei quali vengono definiti:

   a. *scrape_interval:* per il job interessato, va a sostituire quello globale
   b. gli indirizzi delle macchine da monitorare (***static_configs/targets***)

file prometheus.yml

Prometheus - Configurazione esempio

Una volta eseguita l'installazione in tutte le VM, bisogna identificare la macchina che svolgerà il compito di *monitoring* (VM Prometheus_Server) e le VM che svolgeranno da nodi (VM Node).

Nella "VM Prometheus_Server" occorrerà modificare il file *prometheus.yml*.
Siccome anche in questa VM è stato installato Node-Exporter, nel file troveremo un ***job_name*** (identificabile dal target ***localhost:9100***) che indicherà che anche la macchina stessa deve essere monitorata. Per questo esempio non ci interessa, quindi questo Job può essere commentato (#)

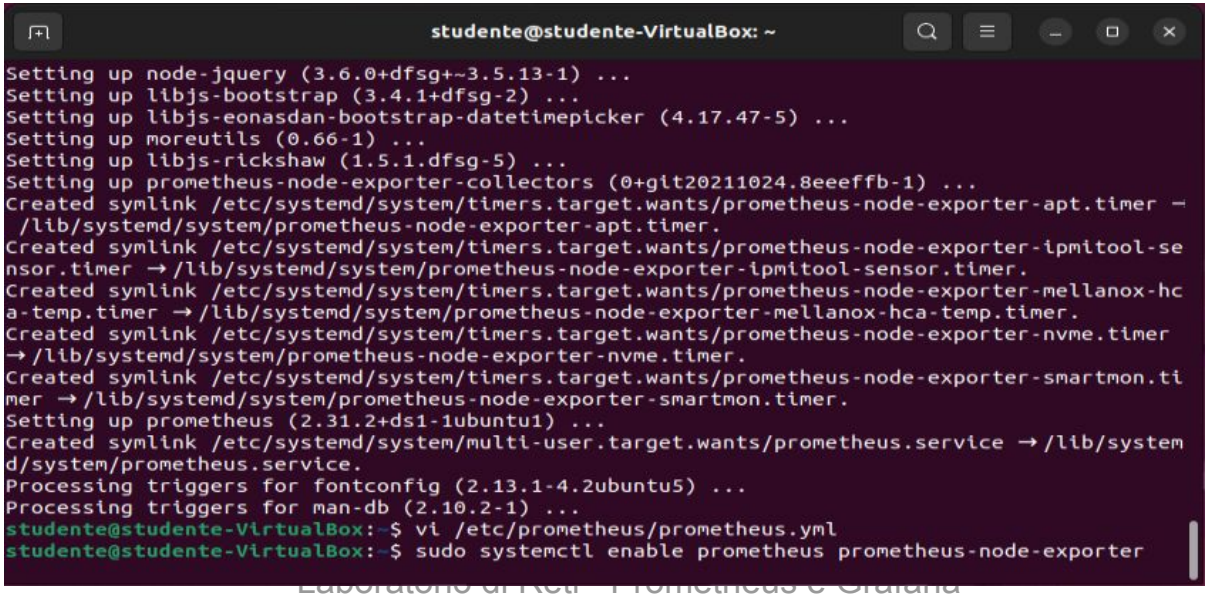
All'interno di *scrape_configs* invece devono essere inserite le VM da monitorare. Quindi dovremo inserire due ***job_name*** (Nodo_1 e Nodo_2) i quali avranno come ***targets***: ['<VM.Address>:9100']. (vedere dal file come devono essere scritte queste cose) e come *scrape_interval* un intervallo da 10s

(per vedere gli indirizzi dei singoli nodi, andare sul terminale dei nodi e digitare *$ ifconfig*)

Installazione - abilitazione dei servizi

Infine per far partire il processo e per eseguirlo sempre durante lo startup del sistema operativo è necessario usare il seguente comando. Una volta eseguito **l'installazione sarà completata**

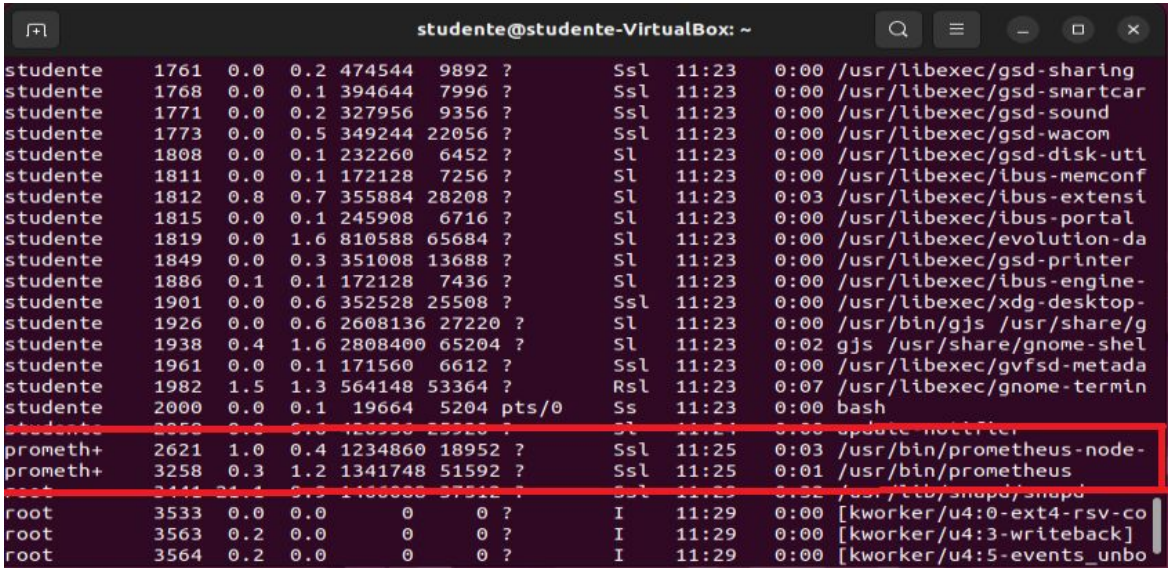
```
$ sudo systemctl enable prometheus prometheus-node-exporter
```



Installazione

E’ possibile verificare se l’installazione usando il comando per listare tutti processi degli utenti

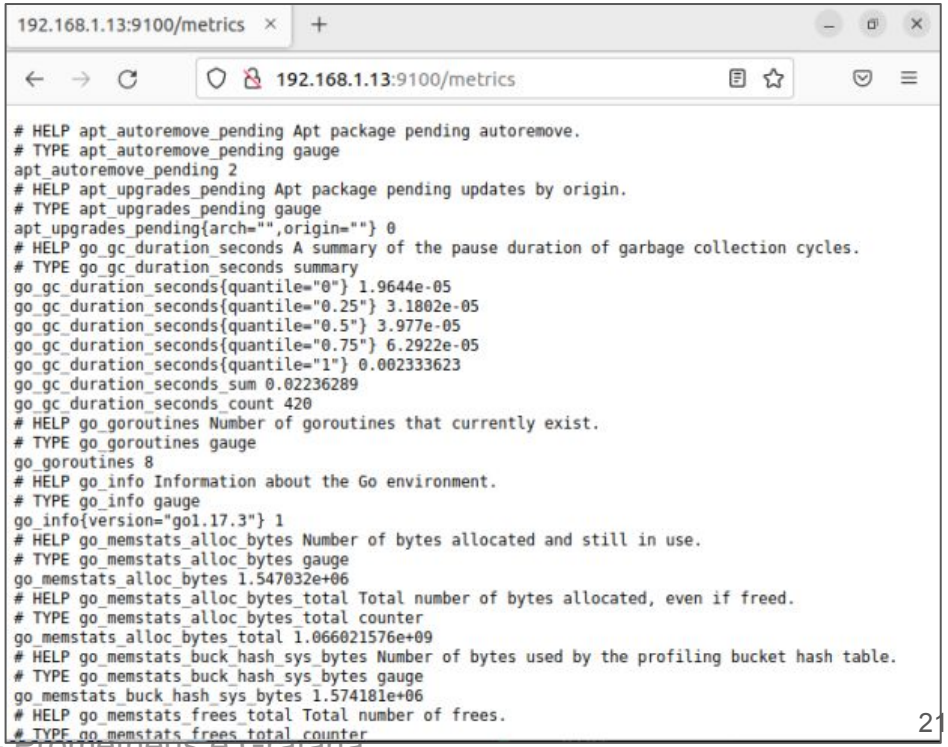
$ ps aux | grep prometheus



Node Exporter - Metriche

E’ possibile vedere le metriche esposte dalle VM Nodo, andando sul Browser e inserendo:

<VM.Node.Address>:9100/metrics



Prometheus - Web UI

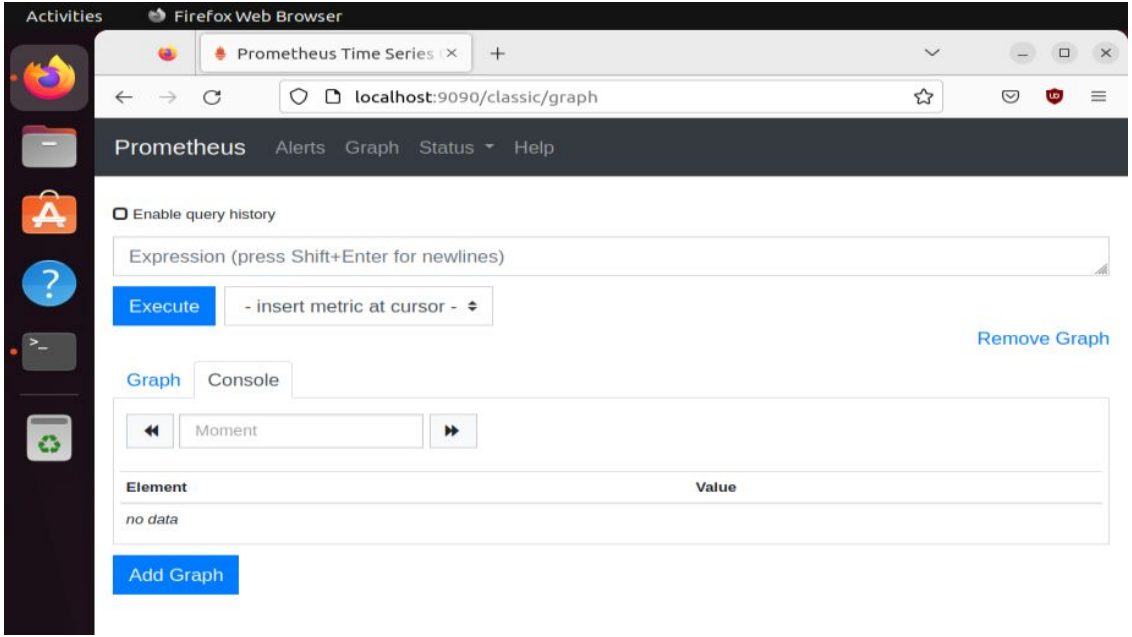
Una volta installato Prometheus, è possibile accedere alla Web User Interface (Web UI) inserendo su barra di ricerca del Browser:

<VM.Indirizzo>:9090

Nel caso volessimo accedere alla Web UI dal Browser della VM sul quale è installato Prometheus, possiamo inserire anche:

localhost:9090

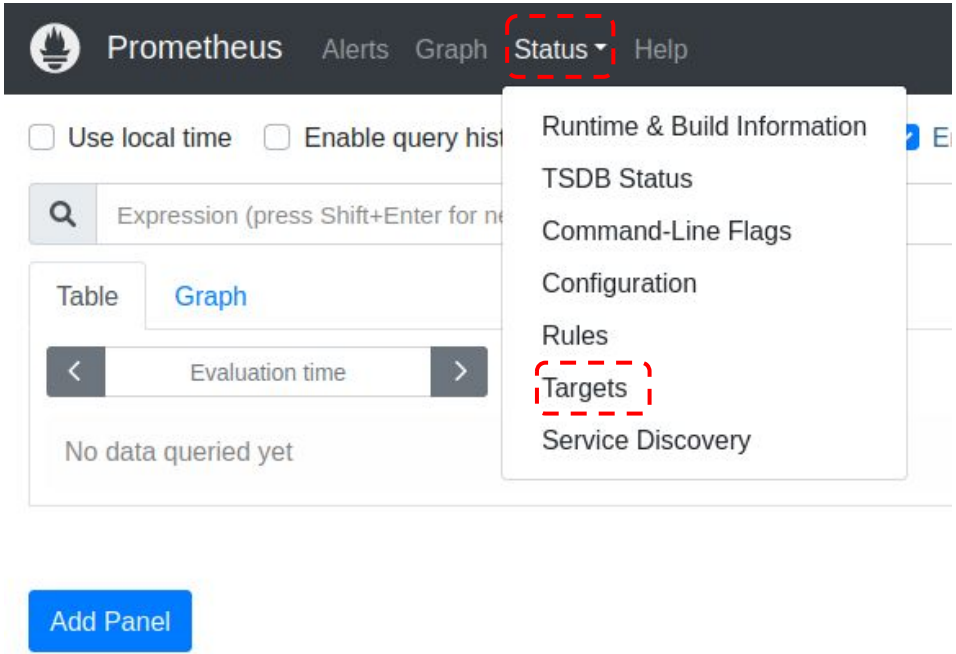
come in figura



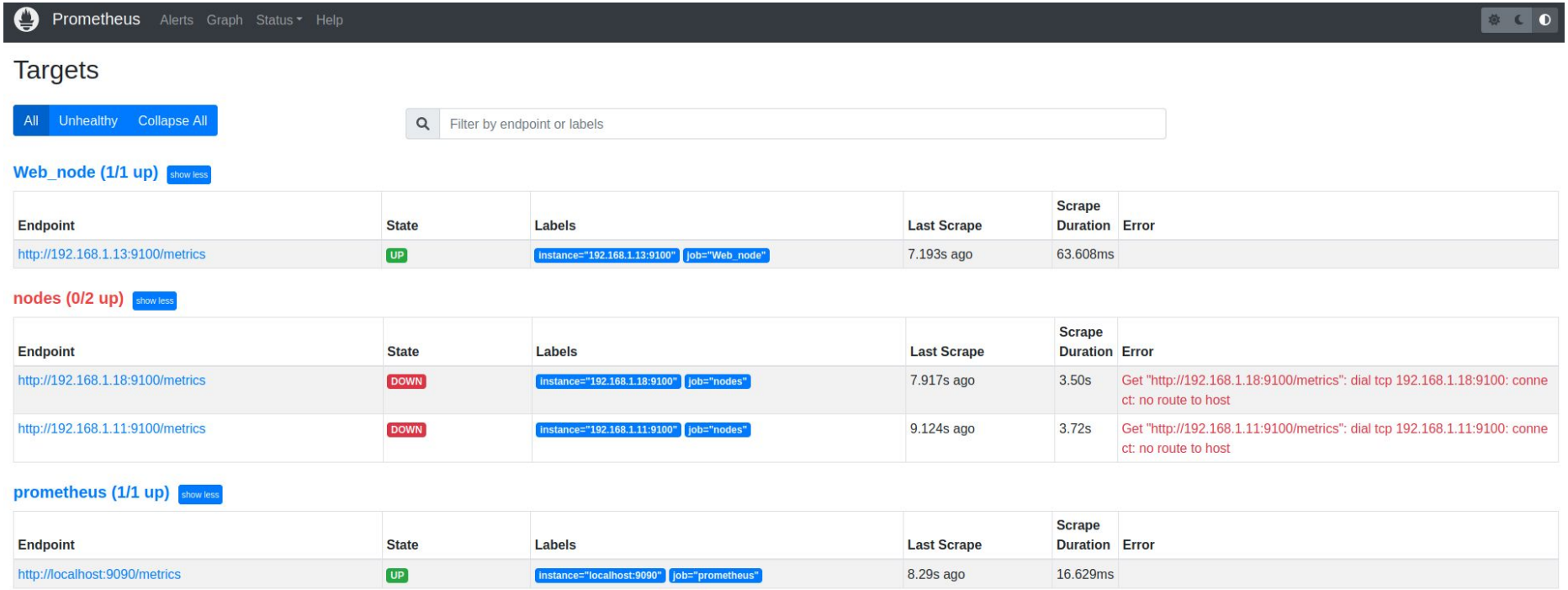
Prometheus Web UI - Nodi Monitorati

Una volta entrati nella Web UI è possibile vedere i nodi a cui Prometheus è collegato cliccando:

Status -> Targets



Prometheus Web UI - Nodi Monitorati



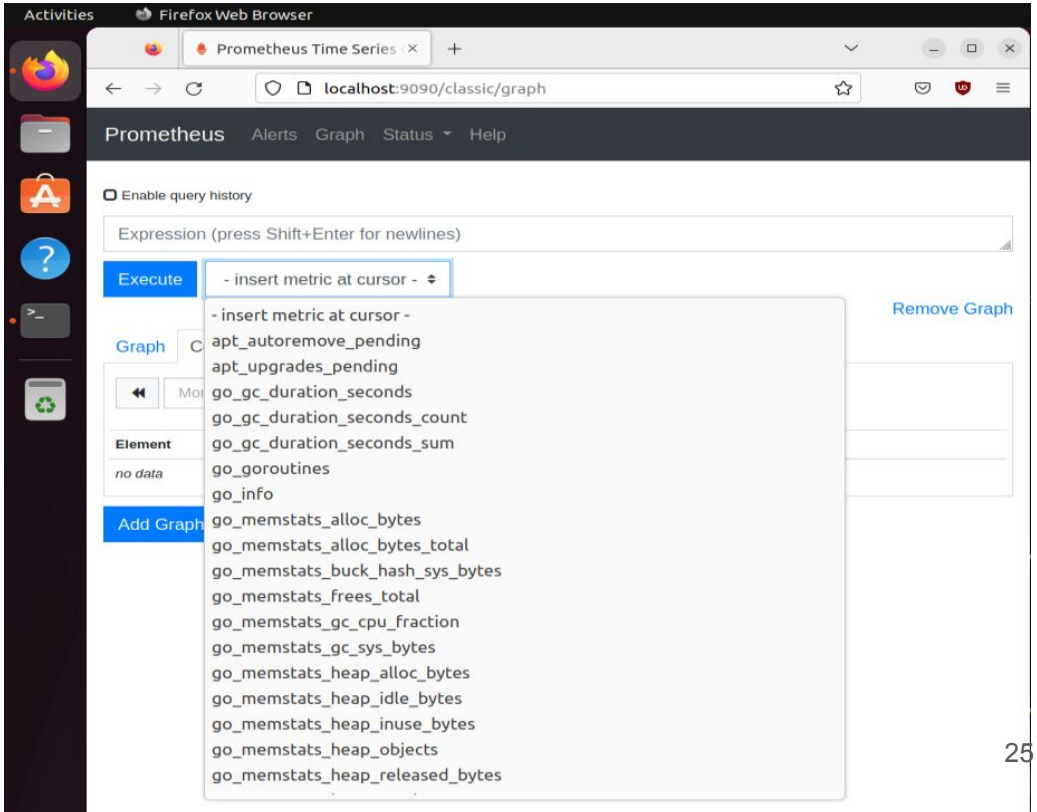
Prometheus Web UI - Query

Nella pagina principale è possibile interrogare il database di Prometheus tramite l'utilizzo di espressioni.

Le espressioni permettono il recupero delle metriche mostrandole all'utente sotto forma di grafici.

In questo modo viene permesso il monitoraggio di gruppi di metriche

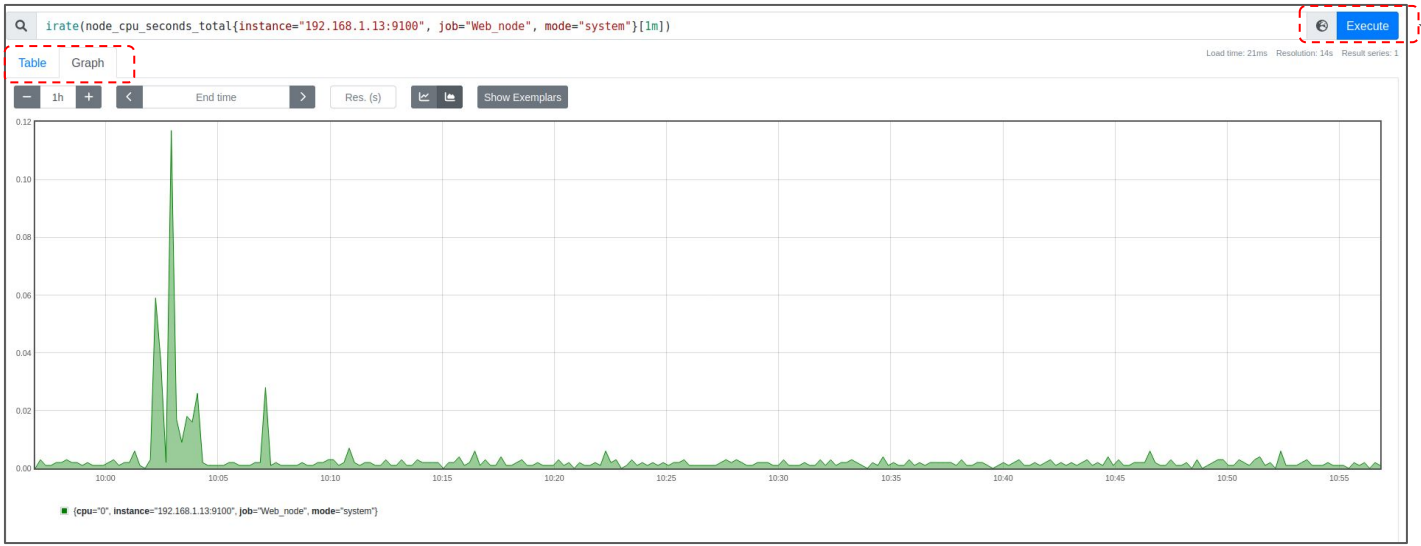
E' possibile scrivere manualmente le espressioni inserendole nel relativo riquadro, altrimenti è possibile sceglierle dal menù a tendina



Prometheus Web UI - Graph

Una volta inserita l'espressione, tramite il pulsante *Execute* è possibile inviare la richiesta al database.
I risultati possono essere visti come

- Grafico
- Tabella



Cliccando sul tondo si apre la finestrella con tutte le espressioni possibili

Grafana

I punti forti di Prometheus riguardano la **raccolta delle metriche e la realizzazione di allarmi**.

Sebbene Prometheus offra anche una Web Interface dalla quale è possibile vedere non solo i targets monitorati ma anche dei grafici relativi alle metriche raccolte, questi ultimi non è possibile gestirli adeguatamente.

Per questo motivo, Prometheus viene spesso utilizzato in combinazione a **Grafana**, tool di visualizzazione dei dati che permette di interrogare il database Prometheus, reperire le metriche interessate e mostrarle su dashboard perfettamente configurabili.

Installazione Grafana

L'installazione di Grafana verrà eseguita sulla macchina che ospita Prometheus.

Per la sua installazione è necessario seguire i seguenti passaggi (Ubuntu/Debian):

1. **Scaricare e aggiungere il package libfontconfig1**

   ```
   $ sudo apt-get install -y adduser libfontconfig1 musl
   ```

2. **Scaricare il package di Grafana**

   ```
   $ wget https://dl.grafana.com/enterprise/release/grafana-enterprise_10.4.0_amd64.deb
   ```

3. **Installare il package scaricato**

   ```
   $ sudo dpkg -i grafana-enterprise_10.4.0_amd64.deb
   ```

Per ulteriori Info, vedere il sito ufficiale di Grafana e selezionare la versione 9.2.6 enterprise:
https://grafana.com/grafana/download/9.2.6

Grafana - Configurazione

Una volta scaricato, è necessario eseguire la configurazione di Grafana Server

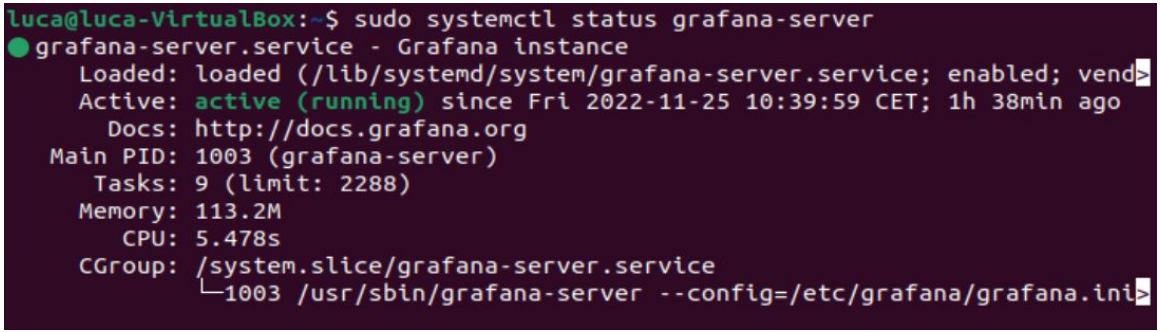
Per prima cosa è necessario abilitare il server da terminale con i seguenti comandi:

- *$ sudo systemctl start grafana-server*
- *$ sudo systemctl enable grafana-server*

Per verificare se il server di Grafana è partito correttamente, inserire il seguente comando:

- *$ sudo systemctl status grafana-server*

Se partito correttamente, dovrebbe comparire a terminale:



Grafana - Configurazione

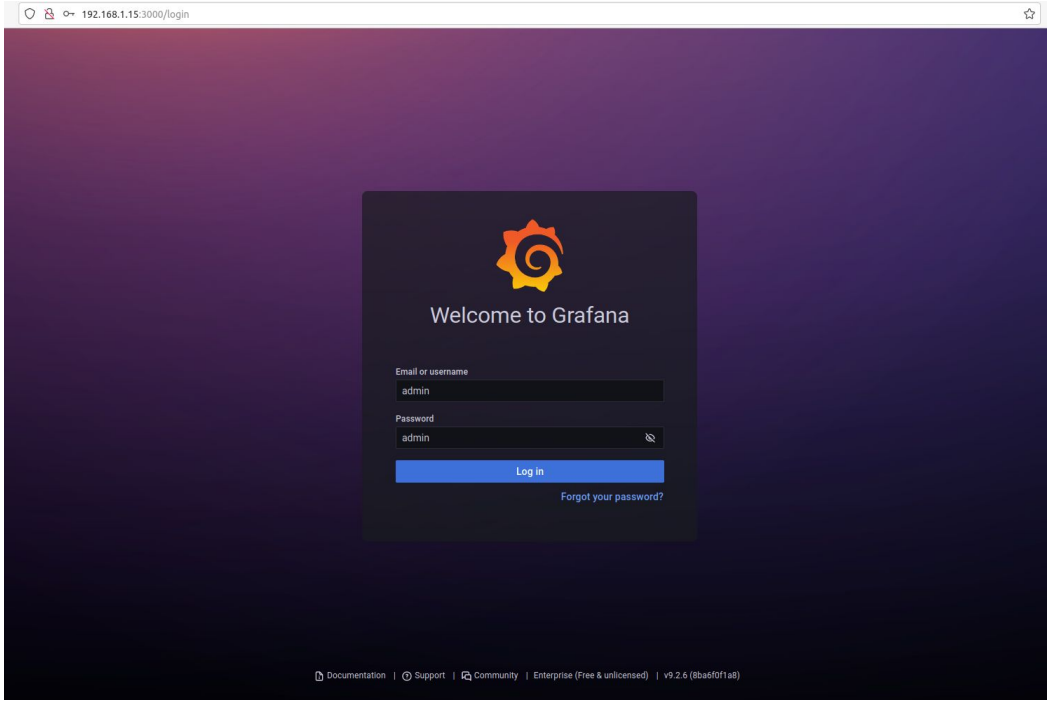
Una volta configurato e fatto partire il server, è possibile accedere all'applicazione web di Grafana inserendo sul Browser:

<VM.Address>:3000

Una volta inserito, compare la pagina di login, la quale richiede l'inserimento delle seguenti credenziali:

- **username**: admin
- **pwd**: admin

Una volta inserite si entrerà nella homepage di Grafana

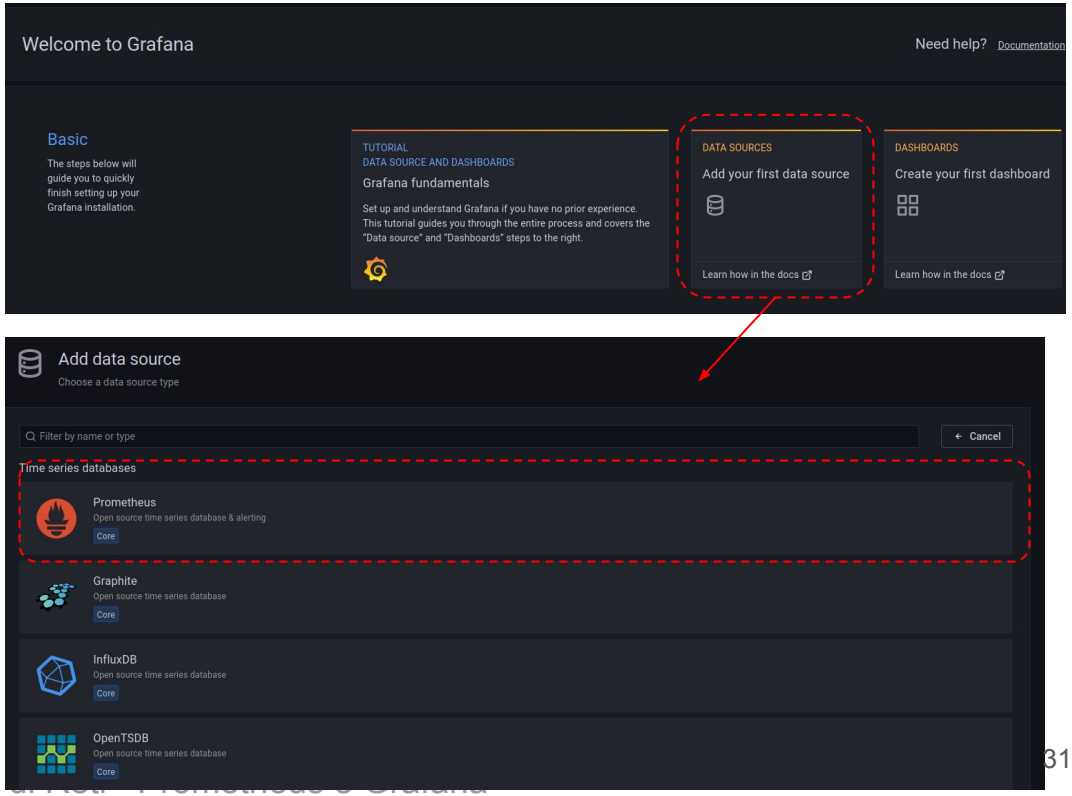


Grafana - Add Data Source

Come prima cosa è necessario indicare a Grafana qual è la sorgente dei dati che vogliamo visualizzare.

In questo laboratorio, la fonte dei dati è Prometheus, quindi:

1. Cliccare su ***Data Sources***
2. Selezionare tra le varie opzioni ***Prometheus***



Configurazione Data Source

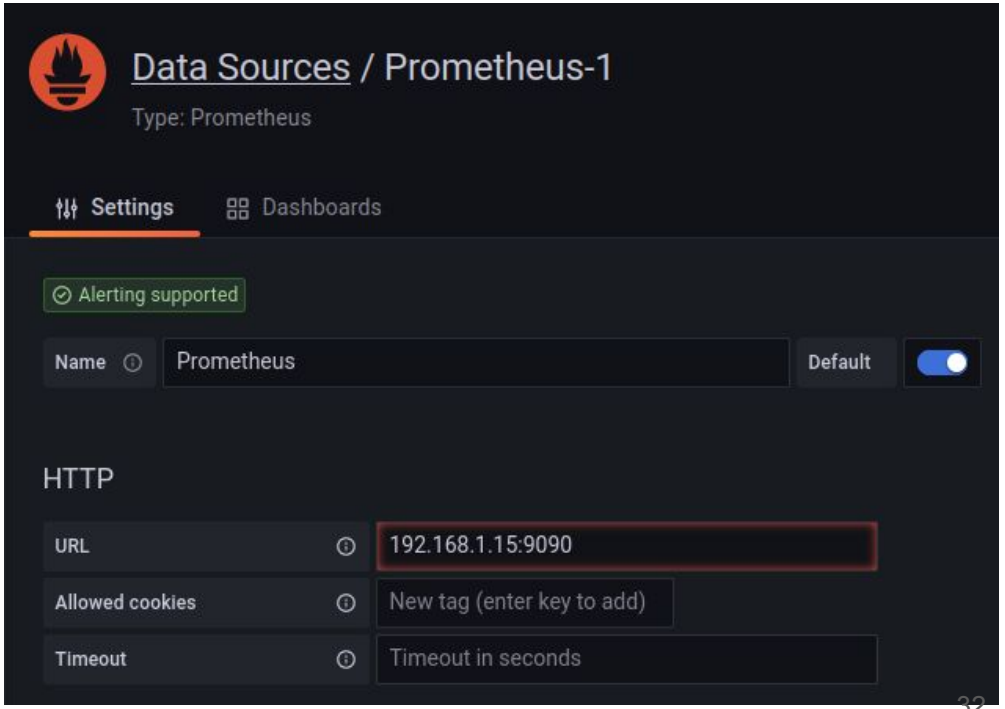
Scelta la fonte dei dati, è necessario configurarla.

In questo esempio basta una configurazione base, inserendo solamente:

1. Il nome da associare alla sorgente
2. L'indirizzo di tale fonte. In questo caso il server Prometheus che funge da fonte dei dati da visualizzare è:

   192.168.1.15:9090

Una volta inseriti, in fondo alla pagina cliccare '*Save & test*'



Grafana Dashboards

Una volta eseguite queste configurazioni è possibile realizzare le dashboards desiderate

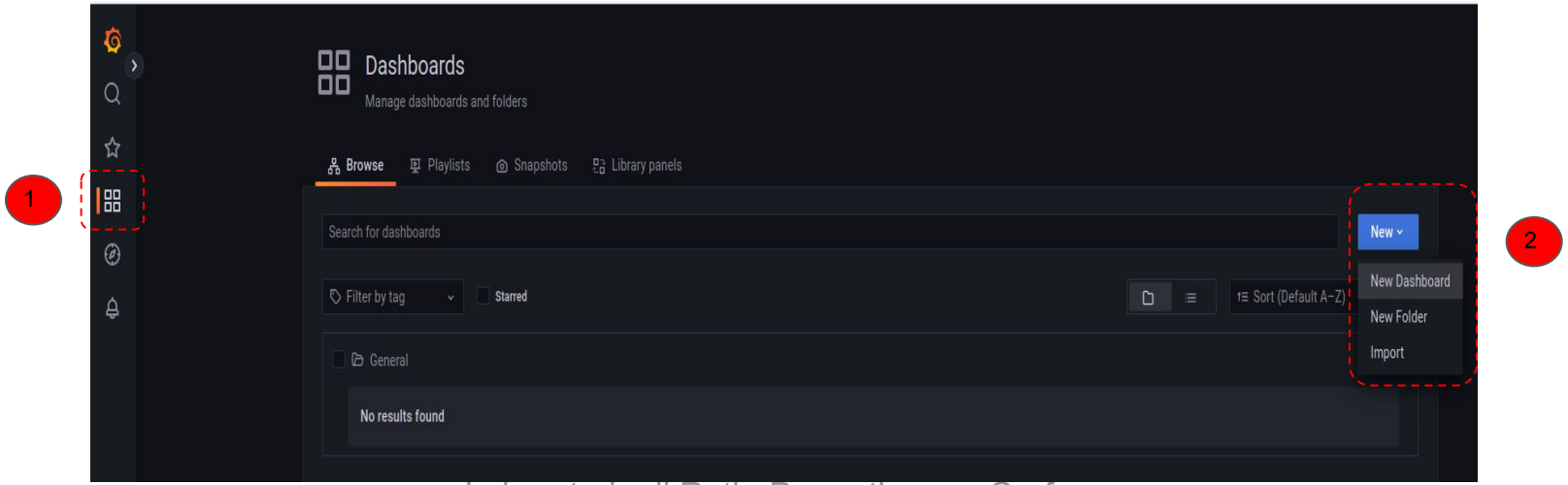
In questo laboratorio andremo a monitorare le quattro metriche principali per una VM:

1. CPU Usage
2. Memory Usage
3. Disk Usage
4. Network Traffic

Grafana - Dashboards

Per creare una nuova Dashboard, seguire i riquadri evidenziati in figura.

1. Permette di entrare nel riquadro della gestione delle Dashboard
2. Permette di creare o una nuova Dashboard, una nuova cartella per contenere più Dashboards oppure importare una cartella o una singola dashboard

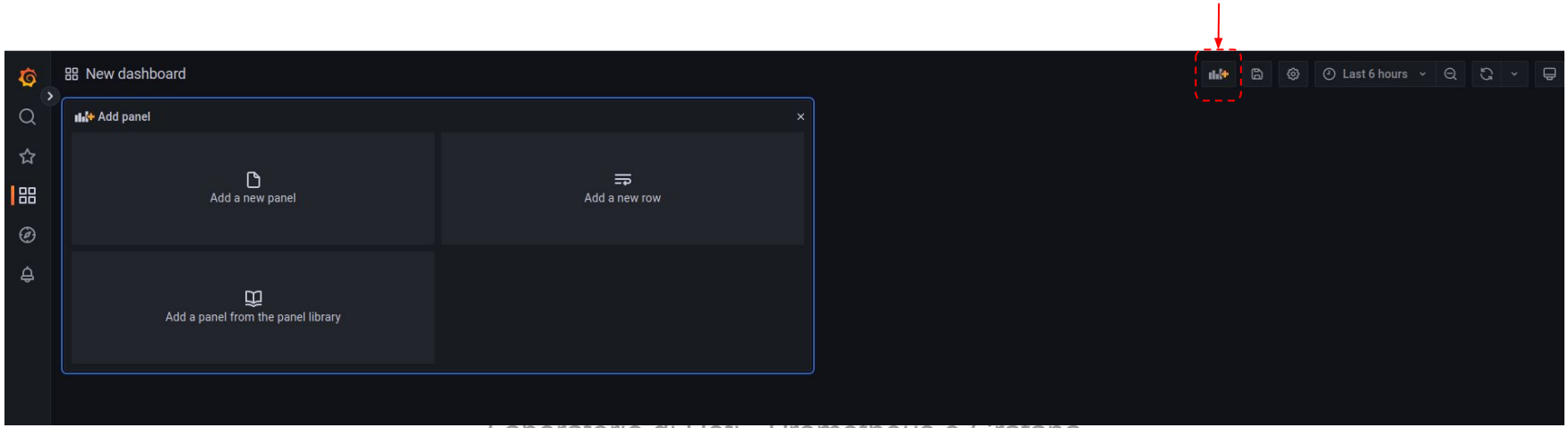


Grafana - Dashboards

Una volta cliccato su *'New Dashboard'* si entrerà nella schermata raffigurata nell'immagine sotto.

Selezionare nel riquadro mostrato '*Add a new panel'.* In questo modo creeremo il primo grafico della nostra Dashboard.

Da notare il pulsante evidenziato in alto a dx che ci permetterà successivamente di creare nuovi *panels* per realizzare altri grafici



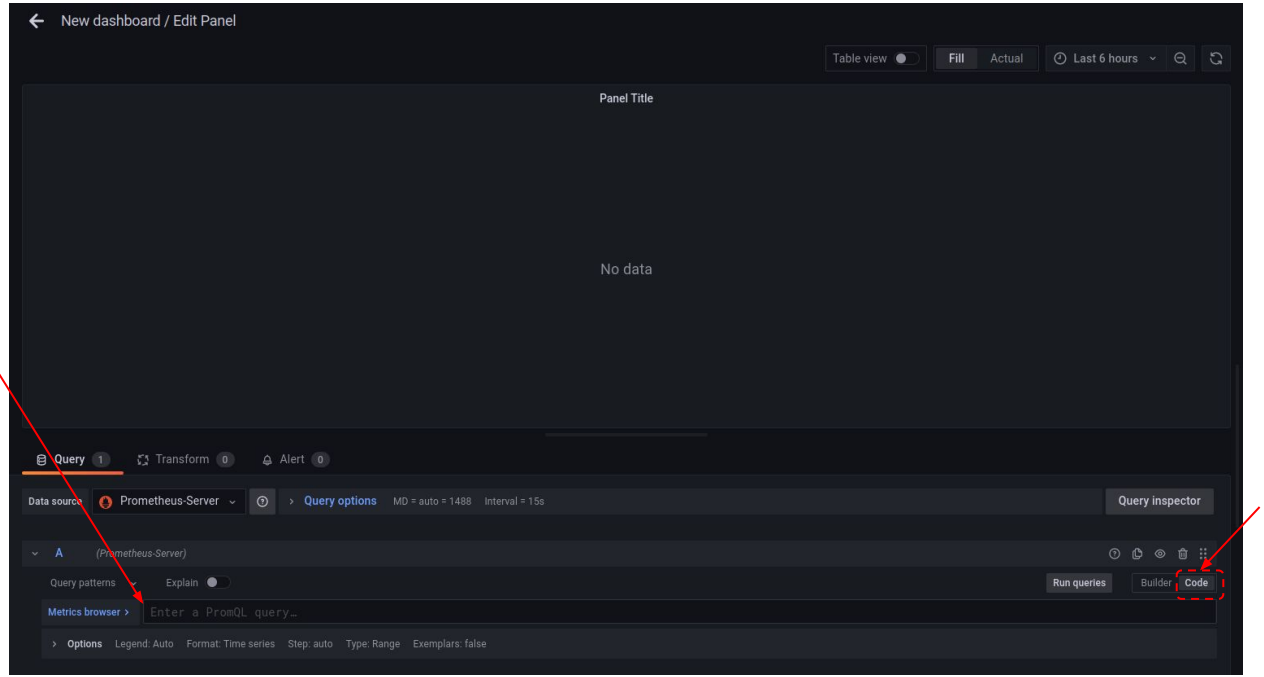
Grafana - Dashboards

Una volta cliccato su *“Add a new panel”* verrà mostrata la seguente schermata. Questa può essere suddivisa in due parti:

1. **Riquadro inferiore:** Utilizzato per inserire le espressioni necessarie per recuperare i dati presenti nella sorgente (in questo caso Prometheus)
2. **Riquadro superiore**: grafico nel quale vengono presentati i dati recuperati

NB Per inserire le espressioni manualmente è necessario selezionare il pulsante evidenziato ‘*code’*

In questo modo potremo inserire le query (o espressioni) manualmente



Grafana - Monitoring CPU Usage

- Il primo grafico che vogliamo visualizzare riguarda **il tempo di utilizzo della CPU da parte della VM**
- Per fare questo dobbiamo scrivere un'espressione (o query) che permetta a Grafana di ottenere le relative metriche dal database di Prometheus al quale è collegato (fonte dei dati, configurazione vista precedentemente)
- Un'espressione che ci permette di recuperare le metriche interessate è la seguente:

file *prometheus.yml* su VM Prometheus_Server

```
# A scrape configuration containing exactly one endpoint t
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to a
  - job_name: "prometheus"

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

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

  - job_name: "nodes"
    scrape_interval: 5s
    static_configs:
      - targets: ['10.58.6.232:9100', '192.168.1.11:9100']
```

node_cpu_seconds_total{job="nodes", instance="10.58.6.232:9100", mode!="idle"}

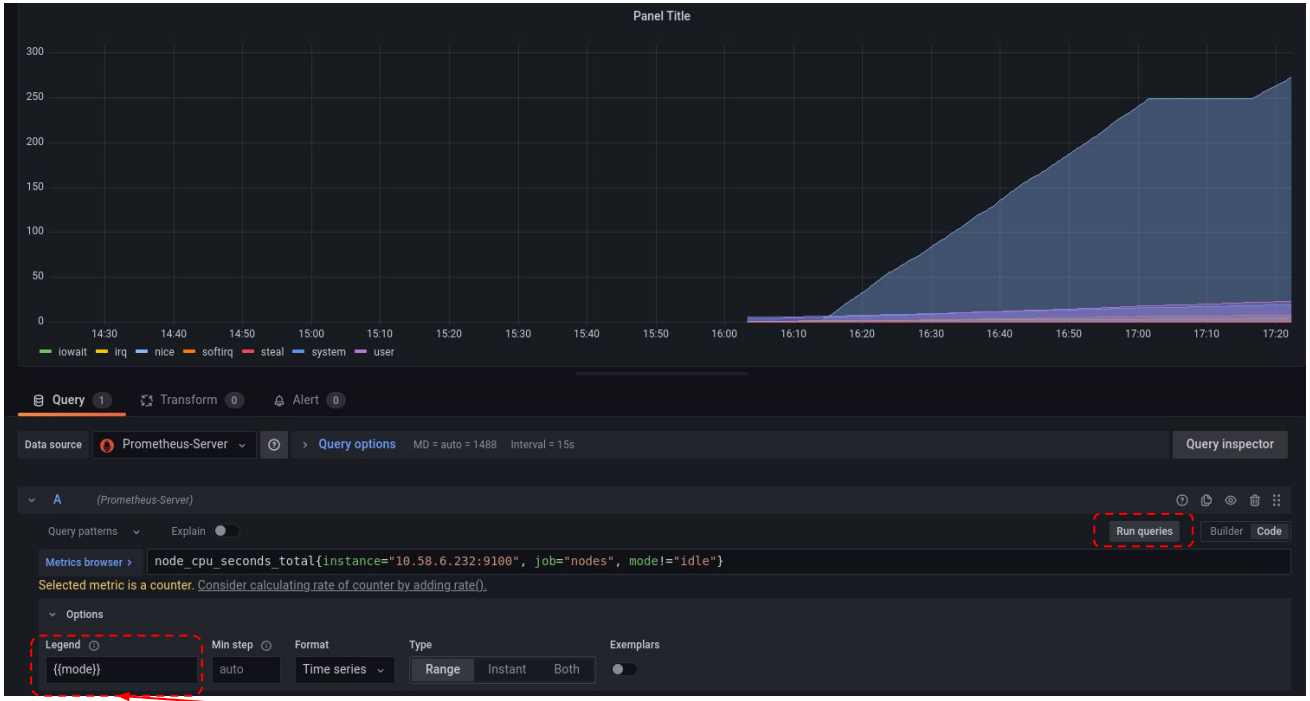
Per ogni espressione che andremo a scrivere, è necessario specificare il nodo target (*instance)* e il Job di cui fa parte, dati consultabili sul file *prometheus.yml.* L'argomento: *mode!="idle"* ci permette di specificare che si vuole ignorare le metriche relative a quando la CPU è in stato di Idle (stato in cui la CPU non viene utilizzata)

Grafana - Monitoring CPU Usage

Una volta inserita l'espressione, per eseguirla, cliccare su '*Run queries*'.

Da notare che sono presenti diverse linee, ognuna delle quali mostra una particolare modalità di utilizzo della CPU. Ad esempio:

- **System**: mostra il tempo della CPU impiegato per eseguire operazioni nel Kernel
- **iowait**: tempo della CPU utilizzato per eseguire operazioni di Input/Output



ci permette di modificare la legenda, mostrando come nome un attributo desiderato

Grafana - Monitoring CPU Usage

Come si può vedere, il grafico ha un andamento crescente.
Questo è dovuto dal fatto che le metriche ottenute sono di tipo *count*, quindi ogni qualvolta vengono ottenute, ne viene eseguita una somma.

Il grafico così ottenuto ci porta poca informazione, quello che vorremmo vedere è l'andamento dell'utilizzo della CPU, per fare ciò dobbiamo modificare l'espressione utilizzata.

Ciò che andremo a inserire nell'espressione è la funzione *irate()*. Tale funzione permette di calcolare il *rateo* dei dati ottenuti in un certo intervallo di tempo, in modo da mostrarne l'andamento.

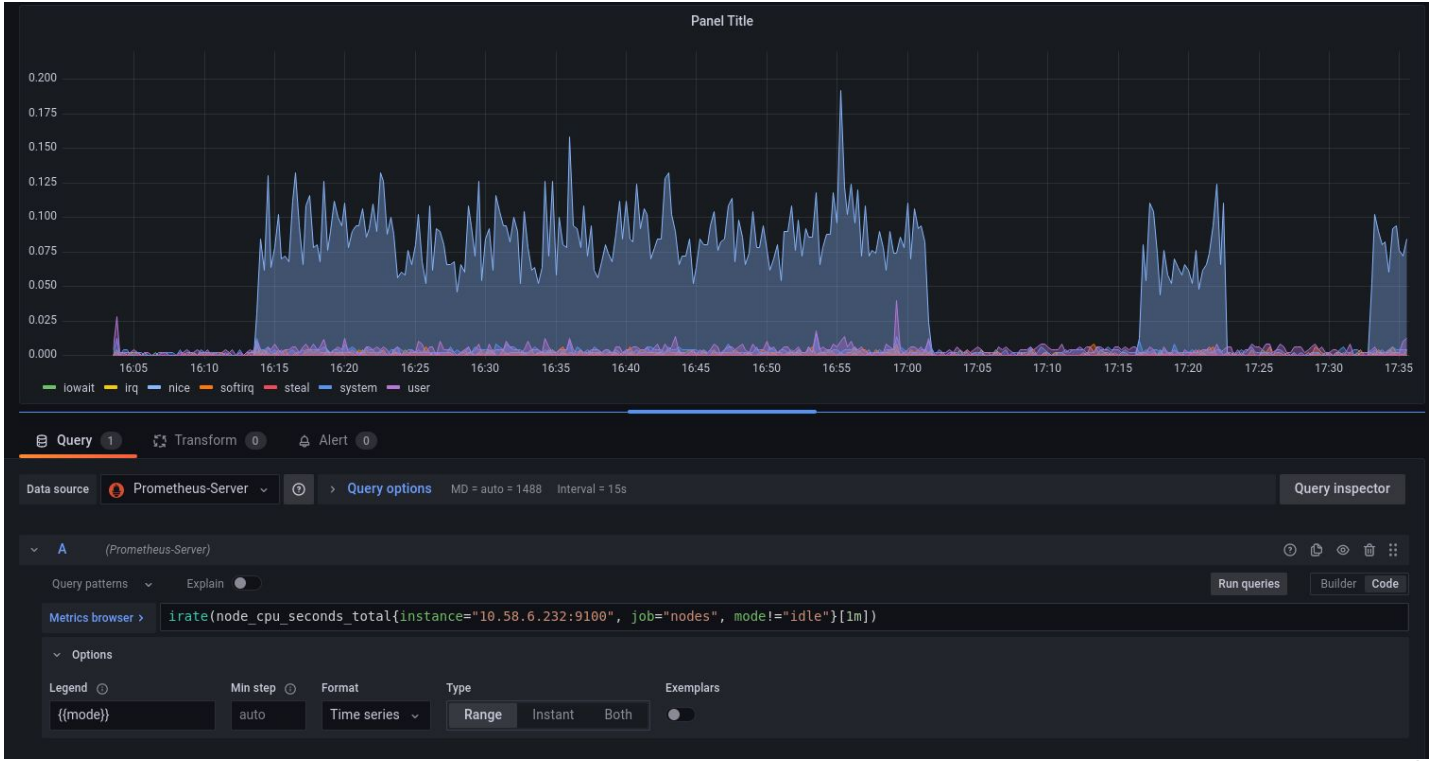
L'espressione diventa come segue:

intervallo di tempo specificato

irate(node_cpu_seconds_total{job="", instance="localhost", mode!="idle"}[1m])

Grafana - Monitoring CPU Usage

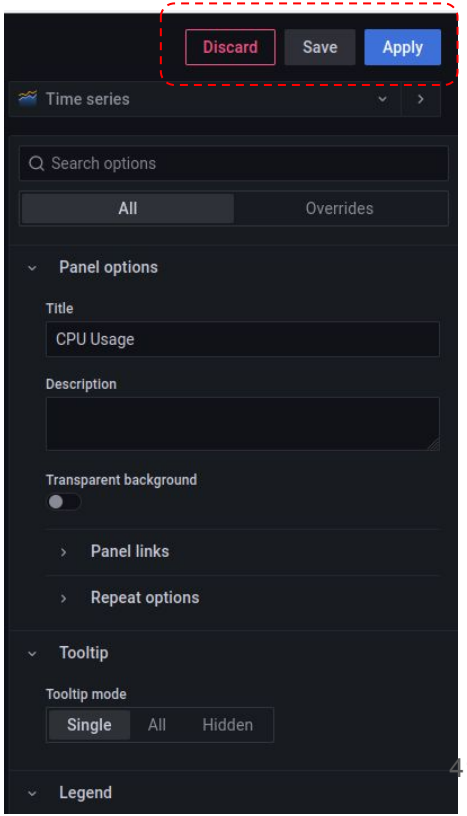
Applicando *irate()* in un intervallo specificato, possiamo vedere l'andamento dell'utilizzo della CPU nelle sue varie modalità



Grafana - Salvare Panel

Per salvare il grafico appena realizzato, è possibile cliccare ‘*Save*’ o ‘*Apply*’ nel riquadro di dx.

Così facendo è possibile rinominare il grafico appena creato e inserirlo in una cartella a propria scelta



Grafana - Memory Usage

Per creare un nuovo grafico:

- Cliccare in alto a dx '*Add panel*'
- Nel riquadro creato '*Add a new panel*'

Grafana - Memory Usage

In questo nuovo grafico, vogliamo monitorare l'utilizzo della memoria della VM target.

Per fare questo dobbiamo considerare che la memoria viene utilizzata per i seguenti scopi:

- caching
- buffering

Quindi per scoprire l'effettivo utilizzo della memoria, al di fuori di questi due campi dovremo eseguire la seguente operazione di sottrazione:

Mem.Utilizzata= Mem.Totale - Mem.Libera - Mem.Cached - Mem.Buffer

Come risultato, otterremo l'effettivo utilizzo della memoria della VM

Grafana - Memory Usage

Di seguito vengono riportate le relative espressioni per ottenere le metriche viste precedentemente

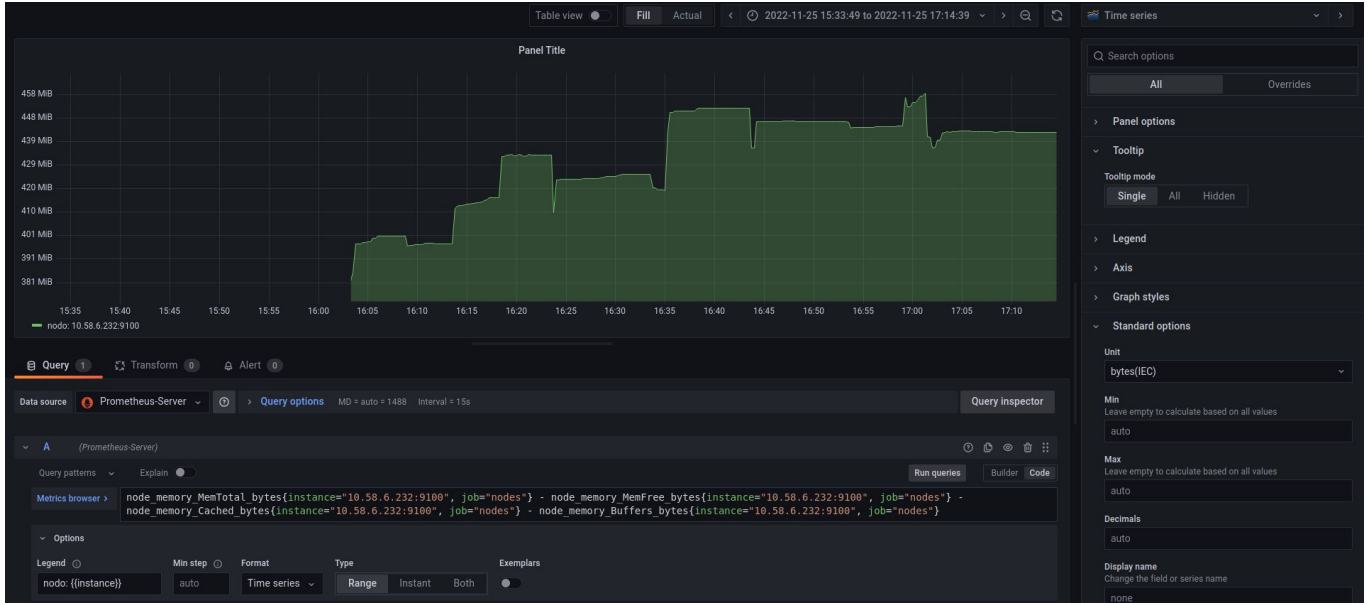
- **Memoria totale:** *node_memory_MemTotal_bytes{instance="...", job="..."}*
- **Memoria libera:** *node_memory_MemFree_bytes{instance="..", job=".."}*
- **Memoria Cache:** *node_memory_Cached_bytes{instance="..", job=".."}*
- **Memoria Buffer:** *node_memory_Buffers_bytes{instance="..", job=".."}*

Da notare che in questi comandi viene espressa la parola *bytes*. I valori presenti nelle metriche ricevute faranno riferimento all'unità di misura *bytes*

Grafana - Memory Usage

In questo caso le metriche ricevute sono di tipo *gauge*, il quale ci da il valore istantaneo permettendo quindi di realizzare un grafico che mostri direttamente l'andamento dell'utilizzo della memoria.

E' possibile stabilire anche l'unità di misura dell'asse Y dal riquadro di dx.



In questo caso è stata scelta l'unità di misura *bytes(IFC)* la quale esprime la memoria secondo la potenza di 2

Grafana - Disk Usage

In questo nuovo grafico, si vuole mostrare l'andamento dell'utilizzo del ***disco*** della VM monitorata.

Anche in questo caso le metriche reperite saranno di tipo *gauge* le quali permetteranno direttamente di mostrare l'andamento.

Per ottenere le metriche interessate dobbiamo riferirci al file system della VM considerata. E' possibile che i valori ottenuti siano "contaminati" da altri dispositivi di memoria che vengono considerati effettivamente come memoria di disco. Per fare ciò, le metriche relative a questi dispositivi non devono essere recuperate

Grafana - Disk Usage

Per ottenere le metriche relative all'utilizzo del disco, si può utilizzare la seguente espressione:

```
node_filesystem_size_bytes{ job="..", instance="..",
                            device !~"/dev/loop.*", device!~"tmpfs | nsfs",
                            device!="gvfsd-fuse"
                            }
```

dove:

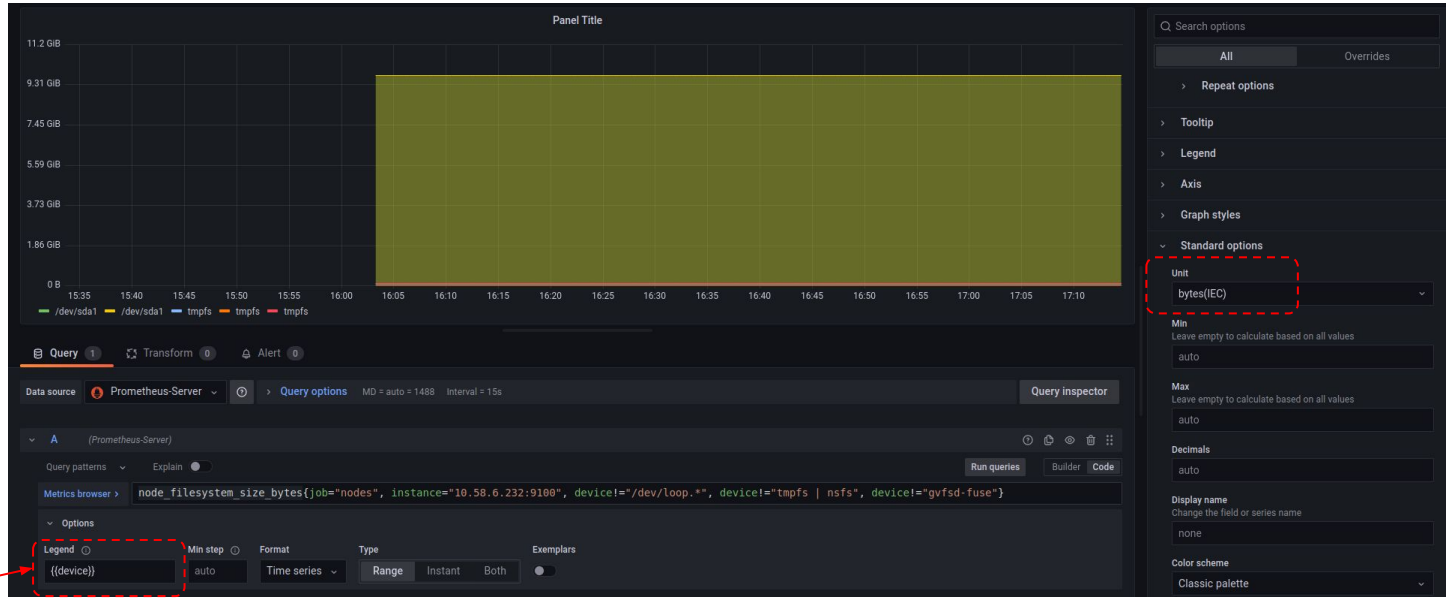
- ***device !~"/dev/loop.*"*** → indica di non considerare i loop device (dispositivi che vengono visti come memorie facenti parte di quella di massa)
- ***device!~"tmpfs | nsfs"*** → indica di non considerare file system temporanei (tmpfs) e network file system (nsfs)
- ***device!="gvfsd-fuse"*** → indica che non deve considerare file system virtuali associati ai singoli utenti (gvfsd)

Grafana - Disk Usage

Anche in questo caso si è scelta come unità di misura: *bytes(IEC)*

E' possibile che nella VM ci siano più unità Disco, per questo nel grafico risultano diverse linee

Come legenda si è scelto di mostrare solamente il nome del dispositivo di memoria, inserendo *{{device}}*



Grafana - Network Traffic

In quest'ultimo grafico vogliamo invece vedere il traffico che passa sulla macchina, in entrata ed in uscita.

Le espressioni che utilizzeremo sono le seguenti:

- **Traffico in entrata:** *irate(node_network_receive_bytes_total{job="..", instance=".."}[1m])*
- **Traffico in uscita:** *irate(node_network_transmit_bytes_total{job="..", instance".."}[1m])*

Come si può vedere, ritorna l'utilizzo della funzione *irate()* in quanto le metriche ricevute dall'espressione *node_network_receive_bytes_total* e *node_network_transmit_bytes_total* sono di tipo *count*.

Senza l'utilizzo della funzione *irate()* le metriche ricevute vengono sommate, non permettendo la visione dell'andamento del traffico.

Grafana - Network Traffic

Unità di misura selezionata: *kilobytes/sec* .

E’ possibile visualizzare i dati di due espressioni distinte sullo stesso grafico, selezionando il riquadro *“+ Query”* .

Per facilitare la comprensione del grafico, nella legenda viene mostrato il nome dell’interfaccia di rete monitorata

Grafana - Dashboard Complessiva