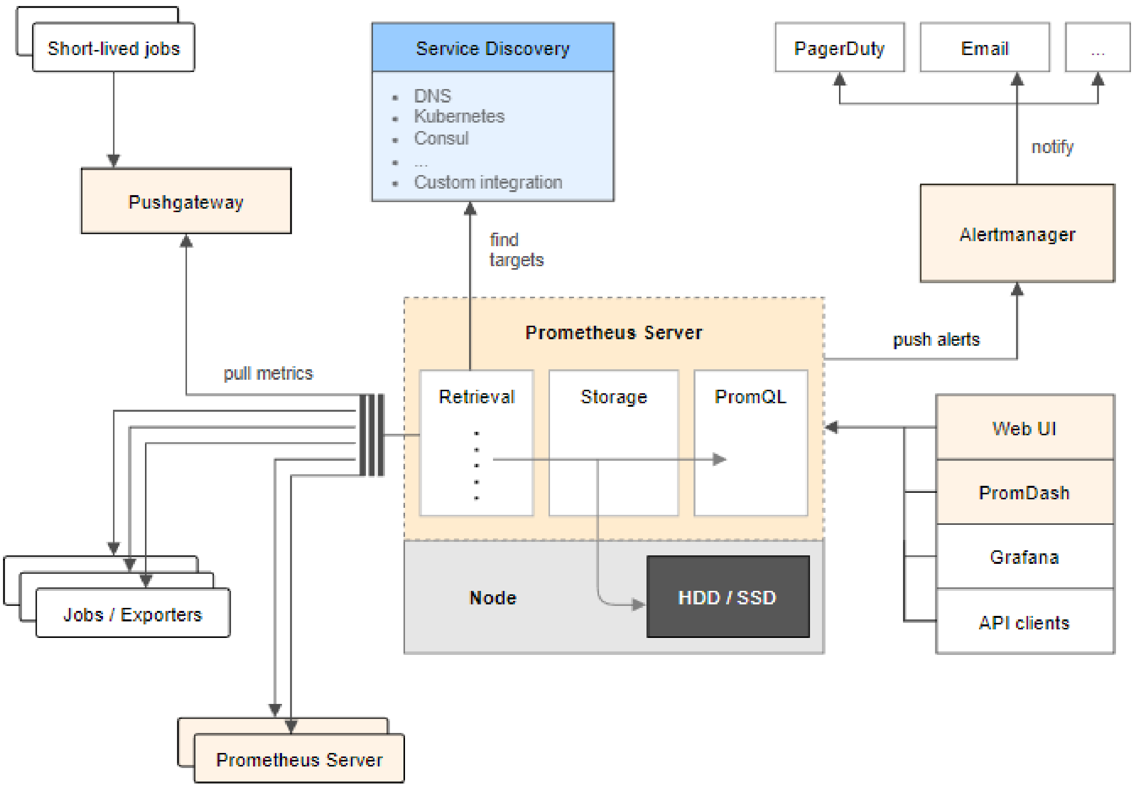
Stanislav Nikolov – Scalefocus

Monitoring Windows OS with Prometheus

Architecture overview



- **We will need to install Prometheus on a machine to store and analyze the metrics collected by the Windows Exporter.**
- **The Windows Exporter is a separate tool that runs on the Windows machine and collects system-level metrics, such as CPU usage and memory usage, which are exposed in a format that Prometheus can scrape**. Prometheus then stores these metrics and provides a user interface for visualizing and analyzing them.

Therefore, to monitor a Windows machine with Prometheus, you will need to:

1. Install Prometheus on a machine
2. Install the Windows Exporter on the Windows machine you want to monitor
3. Configure Prometheus to scrape metrics from the Windows Exporter running on the Windows machine

Firstly I will start with installing Prometheus with Docker:

- Create a new directory for the Prometheus configuration and data files.

- Create a file called **prometheus.yml** in the new directory, and add the following contents.
    - This configuration file specifies that Prometheus should scrape data from itself (running on **localhost:9090**) every 15 seconds.

- Start a new Prometheus container using the official Prometheus Docker image:

    -
      ```
      docker run -d --name prometheus -p 9090:9090 -v C:\Users\stani\Desktop\prometheus-data:/prometheus-data prom/prometheus --config.file=/prometheus-data/prometheus.yml.
      ```

    - This command starts a new Prometheus container in detached mode (**-d**) with the name **prometheus**, maps port 9090 on the host to port 9090 in the container (**-p 9090:9090**), mounts the **prometheus-data** directory on the host to **/prometheus-data** in the container (**-v** C:\Users\stani\Desktop\prometheus-data:/prometheus-data), and specifies the path to the Prometheus configuration file using the **--config.file** option.

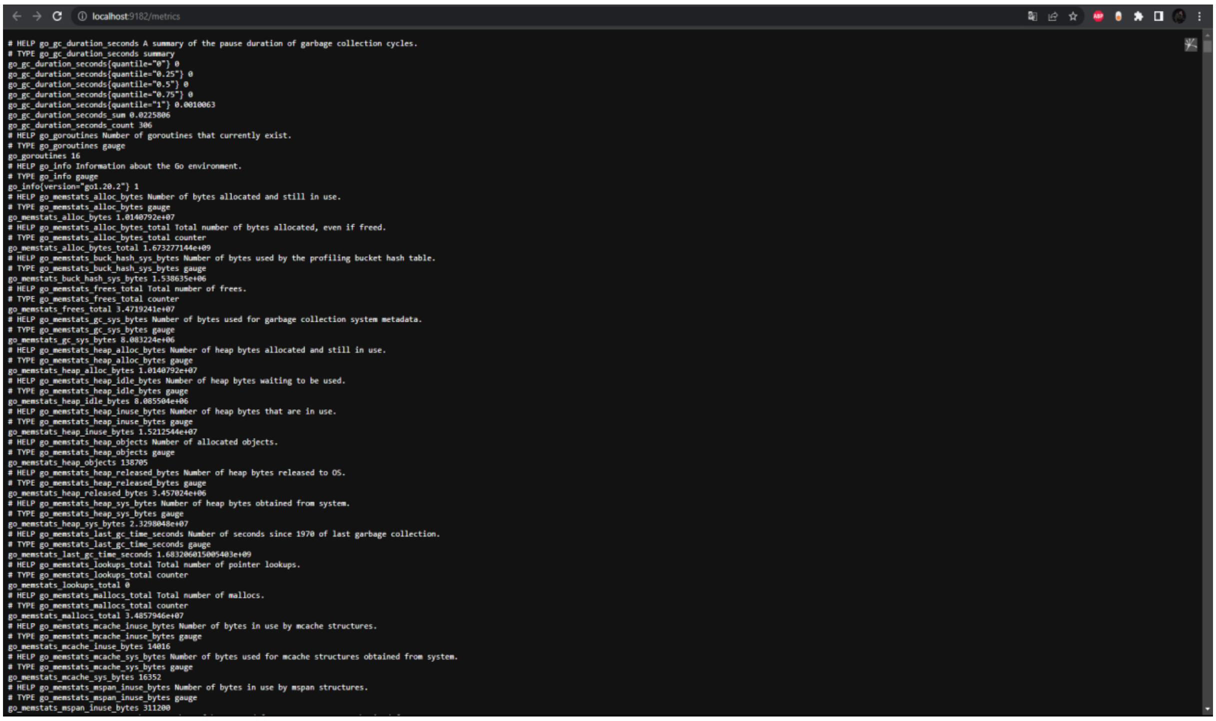
    - We can access the Prometheus web interface by opening a web browser and navigating to http://localhost:9090. From here, you can view metrics, set up alerting rules, and perform other monitoring tasks.

After installing Prometheus we will need the Windows Exporter:

- The URL, that is used is the same, specified in the exercise: https://github.com/prometheus-community/windows_exporter/releases/tag/v0.22.0
- Downloading the **windows_exporter-*-amd64.msi.**
- After that we should install it:
    - Navigating to the directory Downloads/ in the terminal, after that executing:

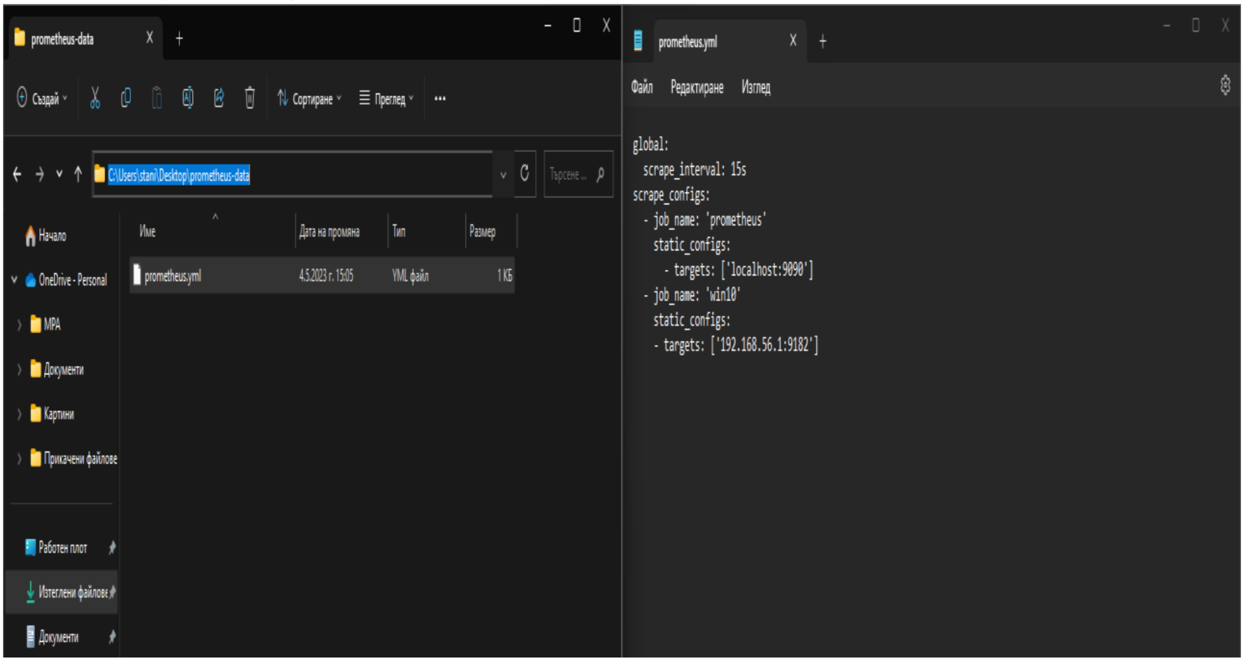
      ```
      msiexec /i .\windows_exporter-0.16.0-amd64.msi
      ```

- To verify whether Windows Exporter is working, open a web browser and visit http://localhost:9182/metrics. If you see the following output, then Windows Exporter is working.



```
# 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"} 0.0010063
go_gc_duration_seconds_sum 0.0225806
go_gc_duration_seconds_count 306
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 16
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.20.2"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.0140792e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.673277144e+09
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.538635e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 3.4719241e+07
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 8.083224e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 1.0140792e+07
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 8.085504e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 1.5212544e+07
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 138705
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 3.457024e+06
# HELP go_memstats_heap_sys_bytes Number of heap bytes obtained from system.
# TYPE go_memstats_heap_sys_bytes gauge
go_memstats_heap_sys_bytes 2.3298048e+07
# HELP go_memstats_last_gc_time_seconds Number of seconds since 1970 of last garbage collection.
# TYPE go_memstats_last_gc_time_seconds gauge
go_memstats_last_gc_time_seconds 1.683206015005403e+09
# HELP go_memstats_lookups_total Total number of pointer lookups.
# TYPE go_memstats_lookups_total counter
go_memstats_lookups_total 0
# HELP go_memstats_mallocs_total Total number of mallocs.
# TYPE go_memstats_mallocs_total counter
go_memstats_mallocs_total 3.4857946e+07
# HELP go_memstats_mcache_inuse_bytes Number of bytes in use by mcache structures.
# TYPE go_memstats_mcache_inuse_bytes gauge
go_memstats_mcache_inuse_bytes 14016
# HELP go_memstats_mcache_sys_bytes Number of bytes used for mcache structures obtained from system.
# TYPE go_memstats_mcache_sys_bytes gauge
go_memstats_mcache_sys_bytes 16352
# HELP go_memstats_mspan_inuse_bytes Number of bytes in use by mspan structures.
# TYPE go_memstats_mspan_inuse_bytes gauge
go_memstats_mspan_inuse_bytes 311200
```

- After installing the Windows Exporter and Prometheus, we should should be able to add Windows Exporter to Prometheus.
  - We need our IPv4 Address: Mine is: 192.168.56.1
- Now we need to open the Prometheus configuration file and add the job, also specify the target IP address of our Windows computer:

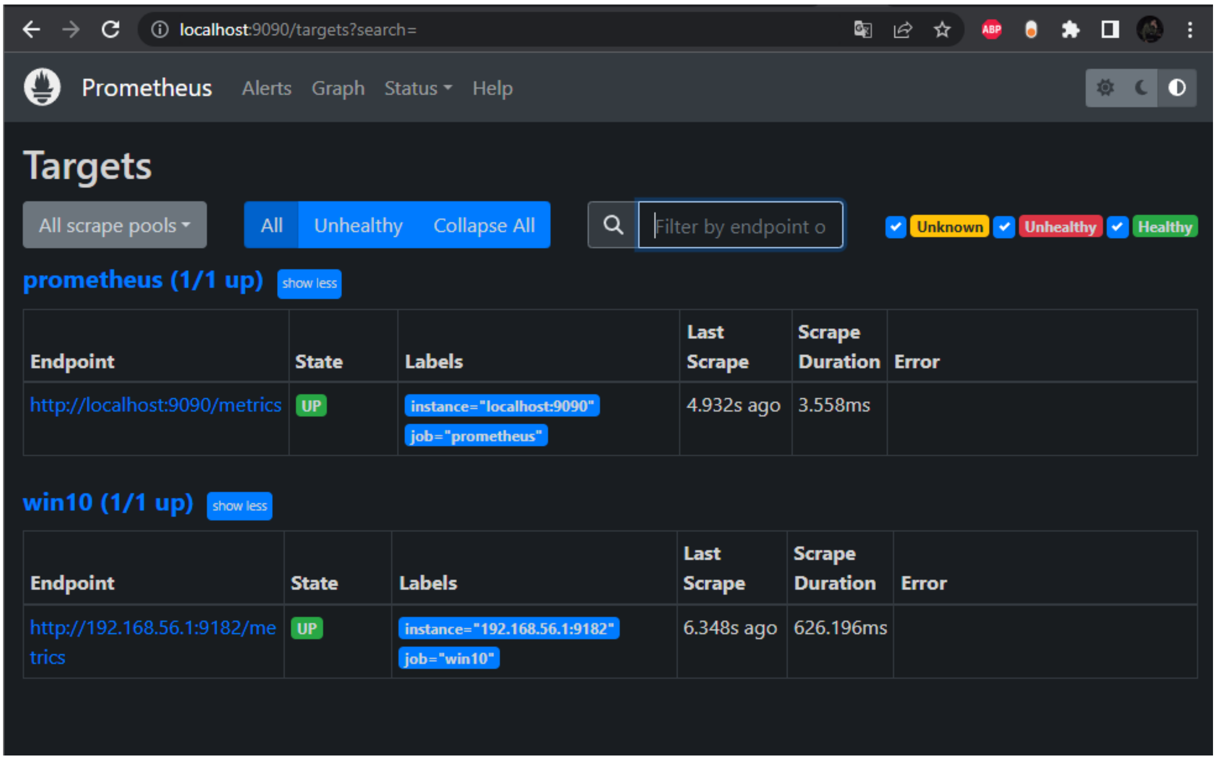


C:\Users\stani\Desktop\prometheus-data

Име	Дата на промяна	Тип	Размер
prometheus.yml	4.5.2023 г. 15:05	YML файл	1 КБ

```
global:
  scrape_interval: 15s
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'win10'
    static_configs:
    - targets: ['192.168.56.1:9182']
```

- After that we can navigate to localhost:9090/targets:



Targets

prometheus (1/1 up)

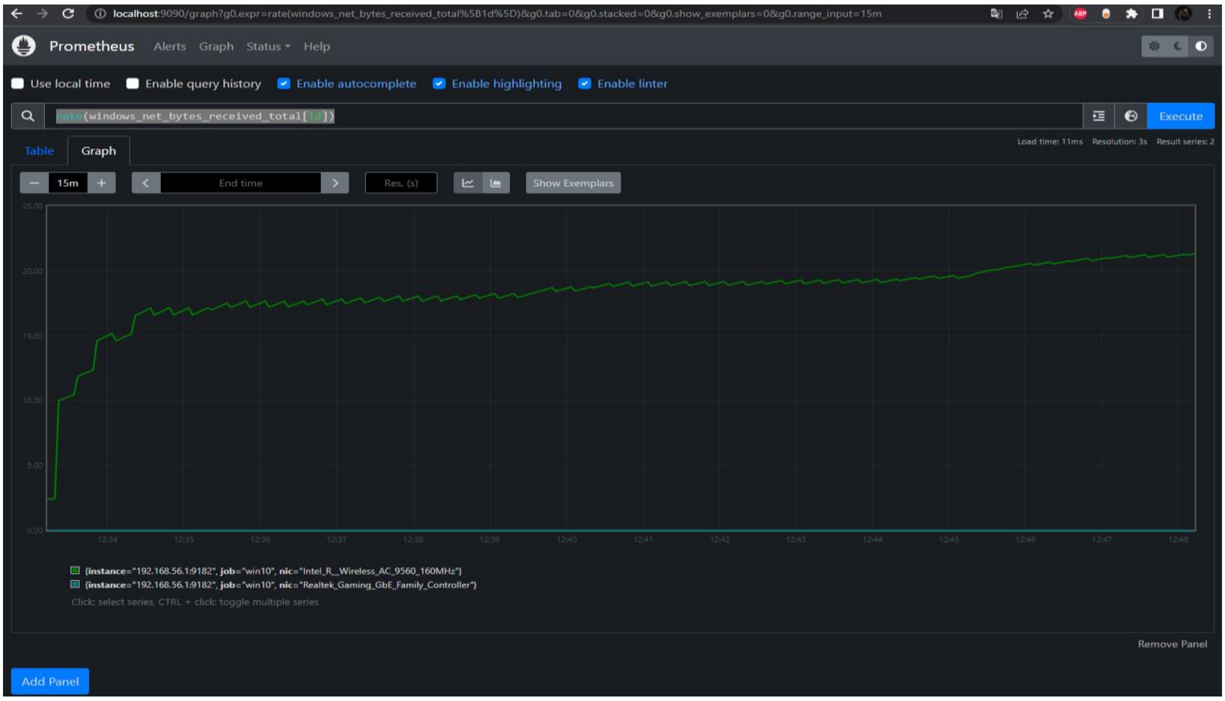
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	4.932s ago	3.558ms	

win10 (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.56.1:9182/metrics	UP	instance="192.168.56.1:9182" job="win10"	6.348s ago	626.196ms	

- In order to monitor Windows with Prometheus we can visit localhost:9090/graph

- Also monitor the download speed of your Windows computer, run the expression **rate(windows_net_bytes_received_total[1m])**.
- You should see a graph of the download speed of your Windows computer, as shown in the screenshot below.



□ Use local time □ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

```
rate(windows_net_bytes_received_total[1d])
```

Load time: 11ms Resolution: 3s Result series: 2

🟩 {instance="192.168.56.1:9182", job="win10", nic="Intel_R__Wireless_AC_9560_160MHz"}
🟩 {instance="192.168.56.1:9182", job="win10", nic="Realtek_Gaming_GbE_Family_Controller"}
Click: select series, CTRL + click: toggle multiple series