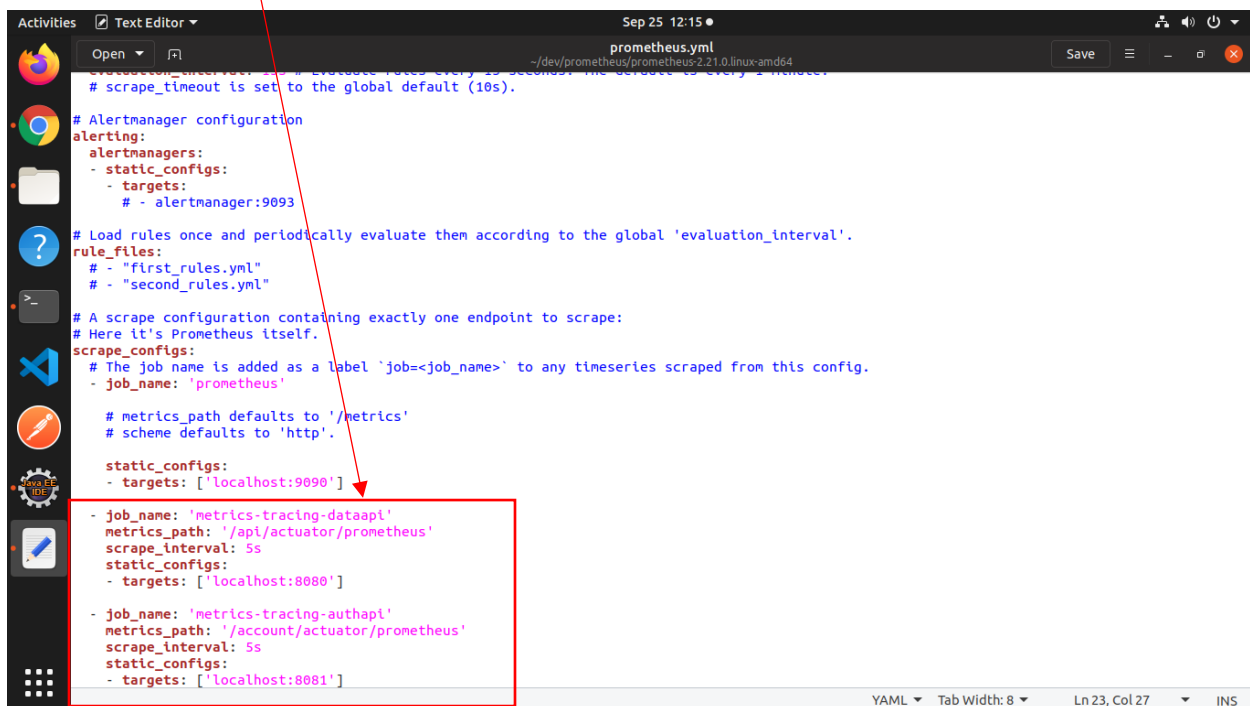


BAH MSD Project – Using Prometheus to Monitor Service metrics

1. Go to Project VM, Download the Prometheus distribution for Linux amd64.
<https://prometheus.io/download/>
2. Extract to home/osboxes/dev/prometheus directory (I had to create dev/prometheus directory)
3. cd to /home/osboxes/dev/prometheus/prometheus-2.21.0.linux-amd64
4. gedit prometheus.yml
5. Add job configuration as follows and saved it.



```
prometheus.yml
# 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_interval'.
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 this config.
  - job_name: 'prometheus'
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

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

  - job_name: 'metrics-tracing-dataapi'
    metrics_path: '/api/actuator/prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:8080']

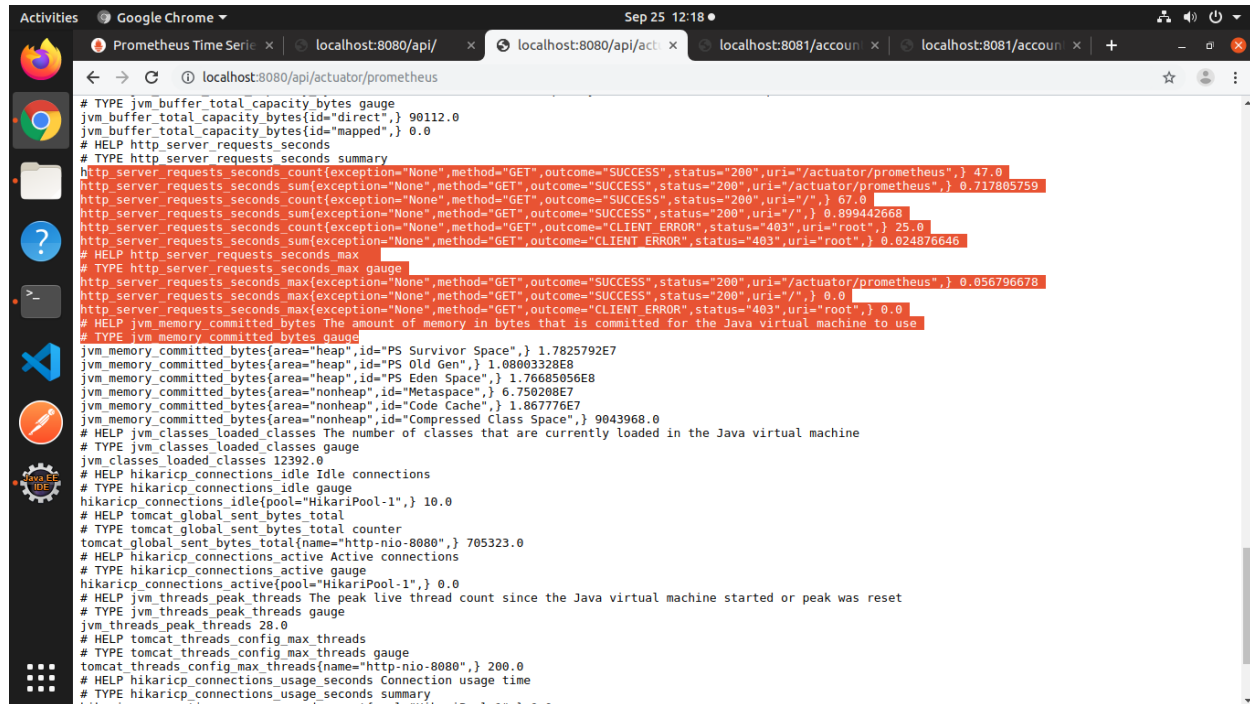
  - job_name: 'metrics-tracing-authapi'
    metrics_path: '/account/actuator/prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:8081']
```

6. start the prometheus.

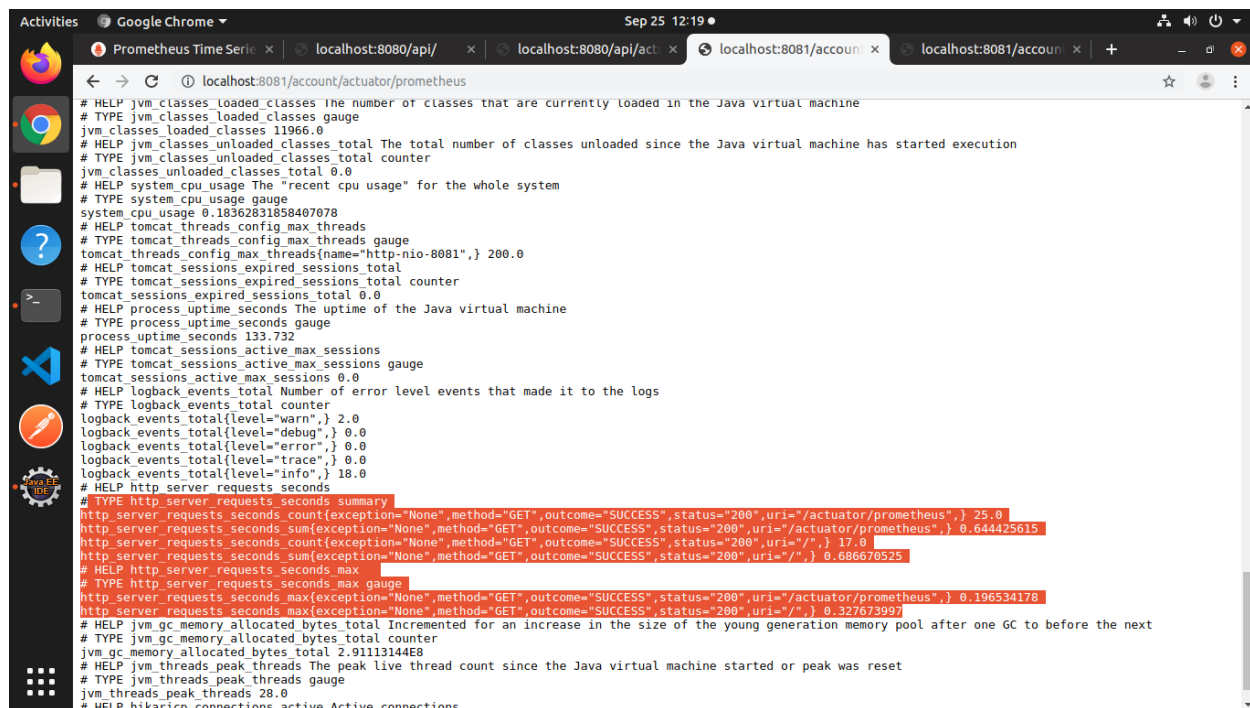
osboxes@osboxes:~/dev/prometheus/prometheus-2.21.0.linux-amd64\$ **./prometheus**

7. Launch the data service - a standalone instance listening at localhost:8080
8. Launch the auth service - a standalone instance listening at localhost:8081
9. Test the service end points using client or postman
10. Run a few page load reload requests

11. Open browser windows and check data api and auth api metrics

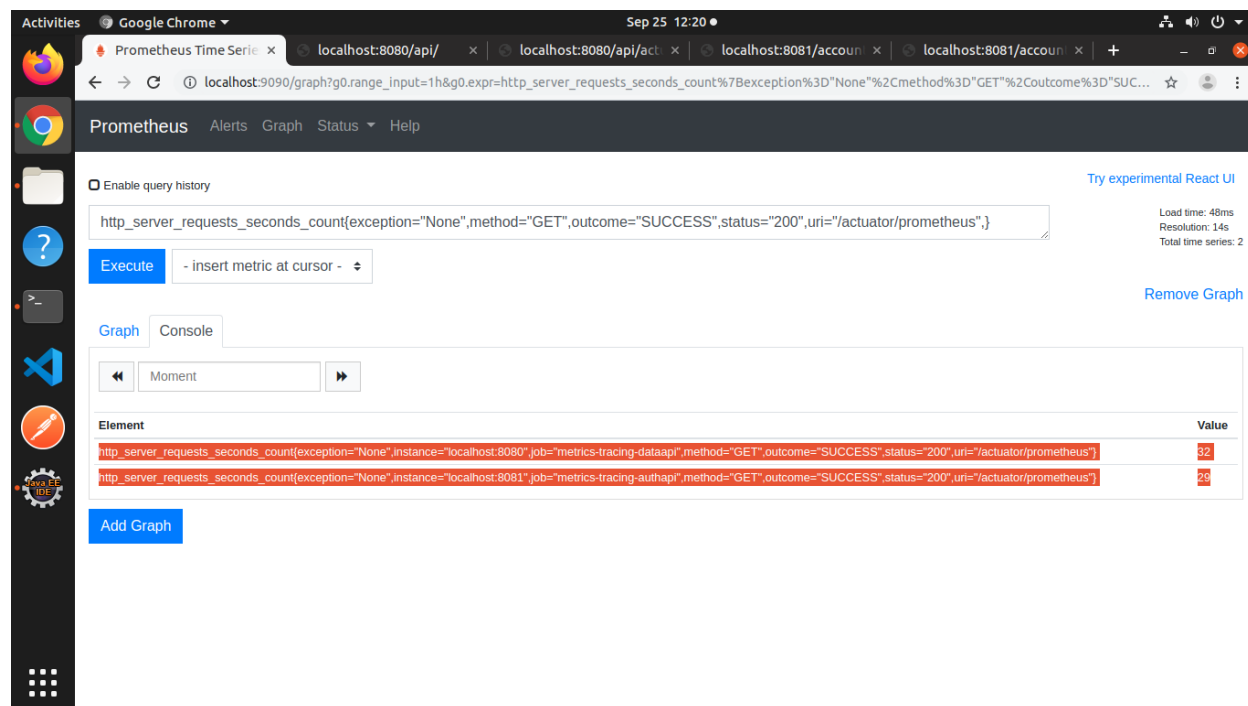


```
# TYPE jvm_buffer_total_capacity_bytes gauge
jvm_buffer_total_capacity_bytes{id="direct",} 90112.0
jvm_buffer_total_capacity_bytes{id="mapped",} 0.0
# HELP http_server_requests_seconds summary
# TYPE http_server_requests_seconds gauge
http_server_requests_seconds_count(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",) 47.0
http_server_requests_seconds_sum(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",) 0.717885759
http_server_requests_seconds_count(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/",) 67.0
http_server_requests_seconds_sum(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/",) 0.899442668
http_server_requests_seconds_count(exception="None",method="GET",outcome="CLIENT_ERROR",status="403",uri="/root",) 25.0
http_server_requests_seconds_sum(exception="None",method="GET",outcome="CLIENT_ERROR",status="403",uri="/root",) 0.624876646
# HELP http_server_requests_seconds_max gauge
http_server_requests_seconds_max(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",) 0.056796678
http_server_requests_seconds_max(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/",) 0.0
http_server_requests_seconds_max(exception="None",method="GET",outcome="CLIENT_ERROR",status="403",uri="/root",) 0.0
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for the Java virtual machine to use
# TYPE jvm_memory_committed_bytes gauge
jvm_memory_committed_bytes{area="heap",id="PS Survivor Space",} 1.7825792E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen",} 1.08003328E8
jvm_memory_committed_bytes{area="heap",id="PS Eden Space",} 1.76685056E8
jvm_memory_committed_bytes{area="nonheap",id="Metaspace",} 6.750208E7
jvm_memory_committed_bytes{area="nonheap",id="Code Cache",} 1.867776E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space",} 9043968.0
# HELP jvm_classes_loaded_classes The number of classes that are currently loaded in the Java virtual machine
# TYPE jvm_classes_loaded_classes gauge
jvm_classes_loaded_classes 12392.0
# HELP hikaricp_connections_idle Idle connections
# TYPE hikaricp_connections_idle gauge
hikaricp_connections_idle(pool="HikariPool-1",) 10.0
# HELP tomcat_global_sent_bytes_total
# TYPE tomcat_global_sent_bytes_total counter
tomcat_global_sent_bytes_total(name="http-nio-8080",) 705323.0
# HELP hikaricp_connections_active Active connections
# TYPE hikaricp_connections_active gauge
hikaricp_connections_active(pool="HikariPool-1",) 0.0
# HELP jvm_threads_peak_threads The peak live thread count since the Java virtual machine started or peak was reset
# TYPE jvm_threads_peak_threads gauge
jvm_threads_peak_threads 28.0
# HELP tomcat_threads_config_max_threads
# TYPE tomcat_threads_config_max_threads gauge
tomcat_threads_config_max_threads{name="http-nio-8080",} 200.0
# HELP hikaricp_connections_usage_seconds Connection usage time
# TYPE hikaricp_connections_usage_seconds summary
hikaricp_connections_usage_seconds_count(pool="HikariPool-1",) 0.0
```



```
# HELP jvm_classes_loaded_classes The number of classes that are currently loaded in the Java virtual machine
# TYPE jvm_classes_loaded_classes gauge
jvm_classes_loaded_classes 11966.0
# HELP jvm_classes_unloaded_classes_total The total number of classes unloaded since the Java virtual machine has started execution
# TYPE jvm_classes_unloaded_classes_total counter
jvm_classes_unloaded_classes_total 0.0
# HELP system_cpu_usage The "recent cpu usage" for the whole system
# TYPE system_cpu_usage gauge
system_cpu_usage 0.18362831858407078
# HELP tomcat_threads_config_max_threads
# TYPE tomcat_threads_config_max_threads gauge
tomcat_threads_config_max_threads{name="http-nio-8081",} 200.0
# HELP tomcat_sessions_expired_sessions_total
# TYPE tomcat_sessions_expired_sessions_total counter
tomcat_sessions_expired_sessions_total 0.0
# HELP process_uptime_seconds The uptime of the Java virtual machine
# TYPE process_uptime_seconds gauge
process_uptime_seconds 133.732
# HELP tomcat_sessions_active_max_sessions
# TYPE tomcat_sessions_active_max_sessions gauge
tomcat_sessions_active_max_sessions 0.0
# HELP logback_events_total Number of error level events that made it to the logs
# TYPE logback_events_total counter
logback_events_total{level="warn",} 2.0
logback_events_total{level="debug",} 0.0
logback_events_total{level="error",} 0.0
logback_events_total{level="trace",} 0.0
logback_events_total{level="info",} 18.0
# HELP http_server_requests_seconds summary
# TYPE http_server_requests_seconds gauge
http_server_requests_seconds_count(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",) 25.0
http_server_requests_seconds_sum(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",) 0.644425615
http_server_requests_seconds_count(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/",) 37.0
http_server_requests_seconds_sum(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/",) 0.686670525
# HELP http_server_requests_seconds_max gauge
http_server_requests_seconds_max(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",) 0.196534178
http_server_requests_seconds_max(exception="None",method="GET",outcome="SUCCESS",status="200",uri="/",) 0.327673997
# HELP jvm_gc_memory_allocated_bytes_total Incremented for an increase in the size of the young generation memory pool after one GC to before the next
# TYPE jvm_gc_memory_allocated_bytes_total counter
jvm_gc_memory_allocated_bytes_total 2.91113144E8
# HELP jvm_threads_peak_threads The peak live thread count since the Java virtual machine started or peak was reset
# TYPE jvm_threads_peak_threads gauge
jvm_threads_peak_threads 28.0
# HELP hikaricp_connections_active Active connections
```

13. Copy the metric name and verify on Prometheus UI



The screenshot shows the Prometheus web interface in a Google Chrome browser. The address bar displays the URL `localhost:9090/graph?g0.range_input=1h&g0.expr=http_server_requests_seconds_count%7Bexception%3D%22None%22method%3D%22GET%22outcome%3D%22SUCCESS%22status%3D%22200%22uri%3D%22%2F%2Flocalhost%2Fapi%2Fprometheus%2F%22%7D`. The interface includes a top navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below this, there's a section for 'Enable query history' with a text input field containing the query `http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",}`. An 'Execute' button is next to the input field. To the right of the input field, performance metrics are shown: 'Load time: 48ms', 'Resolution: 14s', and 'Total time series: 2'. Below the input field, there's a 'Graph' tab and a 'Console' tab. The 'Graph' tab is active, showing a table of results. The table has two columns: 'Element' and 'Value'. The first row shows the metric name and its value '32'. The second row shows the same metric name and its value '20'. An 'Add Graph' button is located at the bottom left of the table.

Enable query history

`http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",}`

Execute - insert metric at cursor -

Try experimental React UI

Load time: 48ms
Resolution: 14s
Total time series: 2

Remove Graph

Graph Console

◀ Moment ▶

Element	Value
<code>http_server_requests_seconds_count{exception="None",instance="localhost:8080",job="metrics-tracing-dataapi",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",}</code>	32
<code>http_server_requests_seconds_count{exception="None",instance="localhost:8081",job="metrics-tracing-authapi",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",}</code>	20

Add Graph