AERO-WEBB
Technical Direction Docs

How to configure cluster monitoring and testing

How to configure cluster monitoring and testing

Monitoring

Azure cluster VM: IP address and name

Name	Public IP	Private IP
Load Balancer	52.143.138.14	10.2.0.5
App1	52.143.142.99	10.2.0.4
App2	52.143.186.146	10.2.0.9
BDD	52.143.185.43	10.2.0.10
Redis	4.233.81.16	10.2.0.6

Prerequisites

Make sure that your public SSH key is recognized by the different VMs. If it is not the case, generate an SSH key with: `ssh-keygen -t ed25519 -C "your_email@example.com"`, get its value from the id_ed25519.pub file located in the .ssh directory created with the previous command. Once you have your public SSH key, ask someone from DevOps team to add your key to the accepted one all VMs.

Prometheus

First you need to connect to LoadBalancer VM from the cluster environment with the command:

```
ssh azureuser@52.143.138.14
```

Then execute the following commands to install and configure prometheus:

Creation of a dedicated prometheus user

```
sudo useradd prometheus --shell /bin/bash
```

Creation of dedicated directories

```
sudo mkdir /opt/prometheus
sudo mkdir /opt/prometheus/data
```

Download prometheus

```
sudo wget -P /opt/prometheus
https://github.com/prometheus/prometheus/releases/download/v2.51.1/prometheus-2.51.1.linux-
amd64.tar.gz
```

Unzip the file in dedicated directory

```
sudo tar xvzf /opt/prometheus/prometheus-2.51.1.linux-amd64.tar.gz -C /opt/prometheus
```

Clean zip file

```
sudo rm /opt/prometheus/prometheus-2.51.1.linux-amd64.tar.gz
```

Rename unzipped directory

```
sudo mv /opt/prometheus/prometheus-2.51.1.linux-amd64 /opt/prometheus/prometheus
```

Change owner of prometheus directory

```
sudo chown prometheus:prometheus -R /opt/prometheus
```

Rename unzipped directory

```
sudo mv /opt/prometheus/prometheus-2.51.1.linux-amd64 /opt/prometheus/prometheus
```

Create and fill prometheus configuration file

```
sudo nano /opt/prometheus/prometheus/prometheus.yml
```

```
# Global conf
global:
  # Global scrape interval (default 1 min)
  scrape_interval: 15s
  # Global rule evaluation interval (default 1 min)
  evaluation_interval: 15s
  # Global timeout (default 10s)
  scrape_timeout: 10s

# Alerting
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Rules
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# Data scraper configuration (data retrievers)
scrape_configs:
  # Prometheus
  # "job_name" is added to each data which is retrieved by this configuration
  - job_name: "Prometheus - Local"
    static_configs:
      # IP target
      - targets: ["localhost:9090"]

    # Username and password needed to access Prometheus
    basic_auth:
      username: identifiant
      password: mdp
```

Start retrieving data

```
/opt/prometheus/prometheus/prometheus --config.file=/opt/prometheus/prometheus/prometheus.yml --
web.config.file=web.yml
```

Creation of prometheus service file

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

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
Restart=always
RestartSec=1
ExecStart=/opt/prometheus/prometheus/prometheus \
    --config.file=/opt/prometheus/prometheus/prometheus.yml \
    --web.config.file=/opt/prometheus/prometheus/web.yml \
    --web.listen-address=:9090 \
    --storage.tsdb.path=/opt/prometheus/data \
    --storage.tsdb.retention.time=30d


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

Configuration and launch of prometheus service daemon

Reload the daemon

```
sudo systemctl daemon-reload
```

Enable prometheus service to start when the VM starts

```
sudo systemctl enable prometheus.service
```

Start prometheus service

```
sudo systemctl start prometheus
```

Check if prometheus service works

```
sudo systemctl status prometheus
```

Postgresql database monitoring

First you need to connect to LoadBalancer VM from the cluster environment with the command:

```
ssh azureuser@52.143.138.14
```

Once you are connected do the following commands:

Creation of a dedicated postgres user

```
sudo useradd postgres --shell /bin/bash
```

Creation of a dedicated directory

```
sudo mkdir /opt/postgres_exporter
```

Download postgres_exporter

```
sudo wget -P /opt/postgres_exporter https://github.com/prometheus-
community/postgres_exporter/releases/download/v0.15.0/postgres_exporter-0.15.0.linux-amd64.tar.gz
```

Unzip the file in dedicated directory

```
sudo tar xvzf /opt/postgres_exporter/postgres_exporter-0.15.0.linux-amd64.tar.gz -C
/opt/postgres_exporter
```

Clean zip file

```
sudo rm /opt/postgres_exporter/postgres_exporter-0.15.0.linux-amd64.tar.gz
```

Rename unzipped directory

```
sudo mv /opt/postgres_exporter/postgres_exporter-0.15.0.linux-amd64
/opt/postgres_exporter/postgres_exporter
```

Change owner of directory

```
sudo chown postgres:postgres -R /opt/postgres_exporter
```

Creation of environment variable file

```
sudo nano /opt/postgres_exporter/postgres_exporter_52.143.185.43_5432.env
```

```
# Database to monitor
DATA_SOURCE_NAME="postgresql://postgres:123erty!@10.2.0.10:5432/aerowebb?sslmode=disable"
```

Creation of postgres_exporter service file

```
sudo nano /etc/systemd/system/postgres_exporter_52.143.185.43_5432.service
```

```
[Unit]
Description=Prometheus exporter for Postgresql
Wants=network-online.target
After=network-online.target

[Service]
User=postgres
Group=postgres
Type=simple
Restart=always
RestartSec=1

WorkingDirectory=/opt/postgres_exporter
EnvironmentFile=/opt/postgres_exporter/postgres_exporter_52.143.185.43_5432.env
ExecStart=/opt/postgres_exporter/postgres_exporter/postgres_exporter \
    --web.listen-address=:9187 \
    --collector.database_wraparound \
    --collector.locks \
    --collector.long_running_transactions \
    --collector.postmaster \
    --collector.stat_activity_autovacuum \
    --collector.stat_bgwriter \
    --collector.stat_database \
    --collector.stat_statements \
    --collector.stat_user_tables \
    --collector.statio_user_indexes \
    --collector.statio_user_tables \
    --collector.wal


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

Configuration and launch of postgres_exporter service daemon

Reload the daemon

```
sudo systemctl daemon-reload
```

Enable postgres_exporter service to start when VM starts

```
sudo systemctl enable postgres_exporter_52.143.185.43_5432.service
```

Start postgres_exporter service

```
sudo systemctl start postgres_exporter_52.143.185.43_5432.service
```

Check if postgres_exporter service works

```
sudo systemctl status postgres_exporter_52.143.185.43_5432.service
```

Link postgres_exporter with prometheus

```
sudo nano /opt/prometheus/prometheus/prometheus.yml
```

Add at the end of prometheus.yml file

```
# PostgreSQL Monitoring with postgres_exporter
 - job_name: "PostgreSQL Exporter"
   scrape_interval: 10s
   scrape_timeout: 10s
   static_configs:
     # Target
     - targets: ["127.0.0.1:9187"]
       labels:
         # "instance" is added to each data retrieved and takes the value of <targets> parameter
         "instance": "Cluster database"
```

JVM monitoring

First you need to connect to App1 or App2 VM which has Tomcat installed in the cluster environment with the command:

```
ssh azureuser@52.143.142.99
```

OR

```
ssh azureuser@52.143.186.146
```

Info

You will need to do the jmx_exporter part for the two App VMs in order to monitor the two tomcat.

Creation of a dedicated jmx_exporter user

```
sudo useradd jmx_exporter --shell /bin/bash
```

Creation of a dedicated directory

```
sudo mkdir /opt/jmx_exporter
```

Download jmx_exporter

```
sudo wget -P /opt/jmx_exporter
https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/1.0.1/jmx_prometheus_javaa
1.0.1.jar
```

Creation of a config.yml file

```
sudo nano /opt/jmx_exporter
```

```
lowercaseOutputLabelNames: true
lowercaseOutputName: true
whitelistObjectNames: ["java.lang:type=OperatingSystem"]
blacklistObjectNames: []
rules:
  - pattern: 'java.lang<type=OperatingSystem><>
(committed_virtual_memory|free_physical_memory|free_swap_space|total_physical_memory|total_swap_spac
    name: os_$1_bytes
    type: GAUGE
    attrNameSnakeCase: true
  - pattern: 'java.lang<type=OperatingSystem><>((?!process_cpu_time)\w+):'
    name: os_$1
    type: GAUGE
    attrNameSnakeCase: true
```

Change owner of directory

```
sudo chown jmx_exporter:jmx_exporter -R /opt/jmx_exporter
```

Configuration and launch of jmx_exporter

Edit setenv.sh file in tomcat

```
sudo nano /opt/tomcat/bin/setenv.sh
```

Start jmx_exporter as javaagent when starting tomcat

```
# Adds Glowroot support
export CATALINA_OPTS=$CATALINA_OPTS" ""-javaagent:/opt/jmx_exporter/jmx_prometheus_javaagent-
1.0.1.jar=12345:/opt/jmx_exporter/config.yaml"
```

For the following part, you need to connect to LoadBalancer VM from the cluster environment with the command:

```
ssh azureuser@52.143.138.14
```

Link jmx_exporter with prometheus

```
sudo nano /opt/prometheus/prometheus/prometheus.yml
```

Add at the end of prometheus.yml file

```
  # App1 JVM monitoring
  - job_name: "Java-app1"
    static_configs:
      - targets: ["10.2.0.4:12345"]
        labels:
          # "instance" is added to each data retrieved and takes the value of <targets> parameter
          "instance": "App1 JVM"
  # App2 JVM monitoring
  - job_name: "Java-app2"
    static_configs:
      - targets: ["10.2.0.9:12345"]
        labels:
          # "instance" is added to each data retrieved and takes the value of <targets> parameter
          "instance": "App2 JVM"
```

Redis monitoring

Follow this documentation https://ruan.dev/blog/2020/05/05/how-to-setup-a-redis-exporter-for-prometheus

Grafana

First you need to connect to LoadBalancer VM from the cluster environment with the command:

```
ssh azureuser@52.143.138.14
```

Once you are connected do the following commands:

Creation of a dedicated grafana user

```
sudo useradd grafana --shell /bin/bash
```

Creation of a dedicated directory

```
sudo mkdir /opt/grafana
```

Download postgres_exporter

```
sudo wget -P /opt/grafana https://dl.grafana.com/enterprise/release/grafana-enterprise-
11.0.0.linux-amd64.tar.gz
```

Unzip the file in dedicated directory

```
sudo tar xvzf /opt/grafana/grafana-enterprise-11.0.0.linux-amd64.tar.gz -C /opt/grafana
```

Clean zip file

```
sudo rm /opt/grafana/grafana-enterprise-11.0.0.linux-amd64.tar.gz
```

Rename unzipped directory

```
sudo mv /opt/grafana/grafana-v11.0.0 /opt/grafana/grafana
```

Change owner of directory

```
sudo chown grafana:grafana -R /opt/grafana
```

Start Grafana software

```
grafana server
```

Creation of grafana service file

```
sudo nano /etc/systemd/system/grafana.service
```

```
[Unit]
Description=Grafana
Wants=network-online.target
After=network-online.target

[Service]
User=grafana
Group=grafana
Type=simple
Restart=always
RestartSec=1
WorkingDirectory=/opt/grafana/grafana
ExecStart=/opt/grafana/grafana/bin/grafana server


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

Configuration and launch of postgres_exporter service daemon

Reload the daemon

```
sudo systemctl daemon-reload
```

Enable postgres_exporter service to start when the VM starts

```
sudo systemctl enable grafana.service
```

Start postgres_exporter service

```
sudo systemctl start grafana.service
```

Check if postgres_exporter service works

```
sudo systemctl status grafana.service
```

Access Grafana

Go to http://52.143.138.14:3000/dashboards to access Grafana

Login/password: admin/admin

Testing

Installation and configuration of Jmeter

To test Aero-Webb in a cluster configuration we can use Aero-Webb frontend available at http://52.143.138.14/login, or we can use Jmeter to stress test the application.

To install Jmeter, you can follow this documentation: how to install jmeter.

Once you installed Jmeter on your Windows or WSL, download plugin manager jar at: https://jmeter-plugins.org/install/Install/ and put the jar in /opt/apache-jmeter/lib/ext.

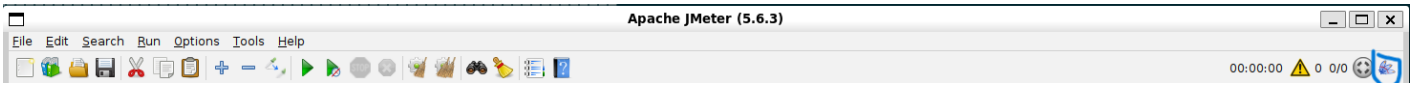
If you are on WSL,do the following command in /opt/apache-jmeter to clone the performance-testing repository:

```
cd /opt/apache-jmeter/

git clone git@innersource.soprasteria.com:2moro/technical-direction/performance-testing.git
```

If you are on Windows, clone the performance-testing repository where it will be accessible by Jmeter.

Once jmeter-plugin is installed, you can launch Jmeter in GUI mode with `sudo ./opt/apache-jmeter/latest/bin/jmeter` on WSL or click on the .exe file in Windows. Once opened, click on the button framed in blue on the image.



It will open a pop-up, click on available plugin and write Ultimate Thread Groups in the search bar. You will find Custom Thread Group, download it and restart Jmeter.

You can now open the optimal-cluster.jmx file to start your test.

> **Information**
>
> The path in TestDataSheet variable in PERF_RUN in the Test Plan needs to be changed to \path\to\performance-testing\on\Windows\performance-testing...

Configuration of Jmeter scripts

The optimal-cluster.jmx script is composed of six Ultimate Thread Groups that will launch Test Case simulating Aero-Webb user actions in different domains:

- Logistics
- Maintenance
- Fleet management
- Engineering data
- Human resources
- Flight operations

Each thread group has different parameters you can configure in PERF_RUN just under Test Case:

- ramp_up represents the time it will take to Jmeter to start all user actions. It is used in "Startup Time,sec" column. To illustrate: if you choose to simulate one hundred users for a duration of one thousand seconds, one user will start every ten seconds
- hold_load represents the time when all the threads/users will be online and doing the different test case. It is used in "Hold Load For, sec" column

- ramp_down represents the time it will take to Jmeter to stop all threads. Note that each thread will finish their treatment before stopping. It is used in "Shutdown Time" column.

There is one last parameter that you have to parameter on each Thread Group separately:

- "Start Thread count", it represents the number of user Jmeter will simulate during the test.

Start Threads Count	Initial Delay, sec	Startup Time, sec	Hold Load For, sec	Shutdown Time
10	0	${ramp_up}	${hold_load}	${ramp_down}

Launching the test

To make the test work, you need to be on SSG Staff connexion or VPN to have access to Load Balancer VM ip address. You also need to execute the following command where "My.LocalIp.Address" is the one in WSL or Windows which you can get with `ifconfig` and `ipconfig` respectively:

```
for each in $(seq 1 150); do ifconfig eth0:$each My.LocalIp.Address.$each; done
```

> **Warning**
>
> You need to do this command everytime you restart your Windows or WSL

Once you are done, update all the ..._spoofing.csv files with your IP address.

Results

In order to get the results, you can go into Aggregate report located after the Thread Groups in Jmeter and click on "Save Table Data" to save an Excel with the information.