

*System Admin*

*Training Assignments*

|  |  |
| --- | --- |
| **Program Code** |  |
| **Issue/Revision** | **x/y** |
| **Effective date** | **04/Aug /2023** |

**Assignment: Docker Compose**

**Part 1: Pre requirement**

* 01 Server with Docker, Docker compose. This server will be named “Docker”. This is the server to deploy Granfa and Prometheus. This server should be in Public subnet.
* 01 Server with Node-exporter. This server will be named “Node01”. This is the server to be monitored. This server can be in Public subnet or Private subnet.
* Make sure that both servers can connect each other. Although, in the AWS lab environment, there is table route allow Public subnet and Private subnet can connect, please check the connection before move to next step.
* Make sure the security groups was allow port 9090 for prometheus and 3000 for granfana. Please check the open port before move to the next step.
* Make sure all firewall on both servers are turned off or allow nessarry ports. (9100, 3000, 9090)
* Download Granfa dashboard : <https://github.com/rfmoz/grafana-dashboards>

**Part 2: Setup on “Docker” server.**

1. Login to “Docker” server
2. Create new folder, named “Monitoring” (Folder name is depended on you)
3. Go to folder “Monitoring”
4. Create new file “docker-compose.yml”, input below content to your

*version: '3.8'*

*services:*

*# Services name prometheus – This will create container name*

*prometheus:*

*image: prom/prometheus:latest*

*container\_name: prometheus*

*ports:*

*- '9090:9090'*

*networks:*

*- monitoring*

*volumes:*

*- ./prometheus.yml:/etc/prometheus/prometheus.yml:ro*

*command:*

*- '--config.file=/etc/prometheus/prometheus.yml'*

*- '--storage.tsdb.path=/prometheus'*

*- '--web.console.libraries=/etc/prometheus/console\_libraries'*

*- '--web.console.templates=/etc/prometheus/consoles'*

*- '--web.enable-lifecycle'*

*# Services name grafana – This will create container name*

*grafana:*

*image: grafana/grafana*

*container\_name: grafana*

*ports:*

*- '3000:3000'*

*networks:*

*- monitoring*

*environment:*

*- GF\_SECURITY\_ADMIN\_PASSWORD=admin #grafana default password*

*depends\_on:*

*- prometheus*

*networks:*

*monitoring:*

*driver: bridge*

1. Create new file “prometheus.yml”. This file must be in the same folder with “docker-compose.yml” file that was create above. Input below content to your “prometheus.yml” file:

*global:*

*scrape\_interval: 15s*

*scrape\_configs:*

*- job\_name: 'node'*

*static\_configs:*

*- targets: ['your "Node" server IP:9100']*

*#Example*

*# - job\_name: 'node'*

*# static\_configs:*

*# - targets: ['10.0.0.80:9100']*

1. Run command *docker-compose up -d* to start build your container.
2. After containers were created, check the containers status if they were runing or not.

**Part 3: Setup on “Node” server**

1. Download node-exporter to your “Node” server with command wget

*wget* [*https://github*](https://github)*.com/prometheus/node\_exporter/releases/download/v1.7.0/node\_exporter-1.7.0.linux-amd64.tar.gz*

1. Extract file node “*node\_exporter-1.7.0.linux-amd64.tar.gz*”

tar *node\_exporter-1.7.0.linux-amd64.tar.gz*

1. Run file “node\_exporter\*” in the folder *node\_exporter-1.7.0.linux-amd64*

**Part 4: Grafana and Prometheus setting.**

1. Access to your Prometheus by <http://your-docker-server-ip:9090>  
   Status > Targets > Make sure your Endpoint state = UP (Endpoint is your “Node” server).
2. Access to your Grafana by <http://your-docker-server-ip:3000>
3. Login with admin/admin then change your password.
4. Open menu > Connections > Add new connection (Or Search for “Connection”).
5. In “Add new connection”, Search for “Prometheus” > “Add new data source”.
6. Name the data source as you want. In “Prometheus server URL”, input the prometheus link: <http://your-prometheus-container-ip:9090>. Then click “Save & Test”.
7. Open menu > Dashboards > New > Import (Or Search for “Dashboards”). Import Dashboard that you just download (“node-exporter-full.json”) > Select data source “Prometheus” that you just create above. > Import.