Environment configuration

The purpose of this document is to provide a step by step on how to set up and configure the environment of the Digital Twin Monitoring Application on the mac operating system. We will go over each tool used and how to set it up. It is highly recommended that the tools are installed in the below order.

This configuration tutorial assumes that the user has docker desktop installed.

# Prometheus node exporter

The Prometheus Node Exporter has the role of collecting data about the computers operation and then sending it to Prometheus. It needs to be installed and configured on each computer that needs to be monitored.

Below are the steps need it to install node exporter on a mac.

1. Go to <https://github.com/prometheus/node_exporter/releases> and download the newest release of the darwin-amd64 version then extract it.
2. Go into the extracted folder using a terminal and then move the node explorer executable to /usr/local/bin/ using the following command:

sudo mv node\_exporter /usr/local/bin/

1. Node explorer can now be started. But it won’t automatically start when the computer starts. To do that it is necessary to make a **launchd** configuration.
2. Open a terminal type the following command twice:

cd ..

1. Create a file in Library/LaunchDaemons/ called node\_exporter.plist using the following command:

nano Library/LaunchDaemons/node\_exporter.plist

1. Open the newly created file with a text editor and paste the following:

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">  
<plist version="1.0">  
<dict>  
 <key>KeepAlive</key>  
 <true/>  
 <key>Label</key>  
 <string>node\_exporter</string>  
 <key>ProgramArguments</key>  
 <array>  
 <string>/usr/local/bin/node\_exporter</string>  
 </array>  
 <key>RunAtLoad</key>  
 <true/>  
</dict>  
</plist>

1. Run the following command to start node exporter:

sudo launchctl load /Library/LaunchDaemons/node\_exporter.plist

1. If an error due to permissions on file is shown, run the following commands to provision permissions on the file.

chmod 600 /Library/LaunchDaemons/node\_exporter.plist  
  
sudo chown root /Library/LaunchDaemons/node\_exporter.plist  
  
sudo chgrp wheel /Library/LaunchDaemons/node\_exporter.plist

1. After grating the permission run the command again to start node exporter.
2. Check if its functioning by going to <http://localhost:9100/metrics>

# Apache spark

Apache spark is one of the applications we must monitor. How to set up the application properly will be covered in the configuration document made by Razvan Budurovici. This will only detail additional steps that are needed to be taken after the installation to ensure data can be collected from Apache Spark.

## Configuration for Loki

1. When creating Apache Spark use the following command to mount a container to Apache Spark

docker run -d --name spark-master --network spark-network --mount type=volume,source=lokipromtail\_shared-logs,target=/opt/bitnami/spark/logs -h spark-master -p 8080:8080 -p 7077:7077 bitnami/spark:latest /opt/bitnami/spark/bin/spark-class org.apache.spark.deploy.master.Master

1. If the container for Apache Spark is already started delete it and remake it using the previous command or mount the container In a different way.
2. Create a file called log4j2.propreties. Using a text editor (such as nano). Add the following to the file:

rootLogger.level = info

rootLogger.appenderRefs = console, file

rootLogger.appenderRef.console.ref = ConsoleAppender

rootLogger.appenderRef.file.ref = RollingFile

appender.console.type = Console

appender.console.name = ConsoleAppender

appender.console.layout.type = PatternLayout

appender.console.layout.pattern = %-5p %c:%L - %m%n

appender.RollingFile.type = RollingFile

appender.RollingFile.name = RollingFile

appender.RollingFile.fileName = /opt/bitnami/spark/logs/spark.log

appender.RollingFile.filePattern = /opt/bitnami/spark/logs/spark-%d{yyyy-MM-dd-HH}.log

appender.RollingFile.layout.type = PatternLayout

appender.RollingFile.layout.pattern = %-5p %c:%L - %m%n

appender.RollingFile.policies.type = Policies

appender.RollingFile.policies.size.type = SizeBasedTriggeringPolicy

appender.RollingFile.policies.size.size = 10MB

appender.RollingFile.strategy.type = DefaultRolloverStrategy

appender.RollingFile.strategy.max = 10

1. Go into a terminal and go into the folder with the log4j2.propreties file.
2. Use the following command to copy the file to Apache spark:

docker cp log4j2.properties spark-master:/opt/bitnami/spark/conf/

1. Restart Apache Spark to update the changes.

## Configuration for Prometheus

1. Create a file called metrics.propreties. Using a text editor (such as nano). Add the following to the file:

\*.sink.prometheusServlet.class=org.apache.spark.metrics.sink.PrometheusServlet

\*.sink.prometheusServlet.path=/metrics

1. Go into a terminal and go into the folder with the metrics.propreties file.
2. Use the following command to copy the file to Apache spark:

docker cp metrics.propreties spark-master:/opt/bitnami/spark/conf/

1. Restart Apache Spark to update the changes.

# Prometheus

Prometheus role is to store metrics data from multiple sources so it can be further used for different sources.

Below are the steps need it to install node exporter on a mac:

1. Create a folder for storing docker compose files this will be used to store the files well be using to run Prometheus and other aps in docker.
2. Create a folder in the original storage folder called Prometheus or another descriptive name.
3. Create a file in the Prometheus folder create a file called docker-compose.yml . The easiest way to do this is by opening a terminal going to the Prometheus folder and then running the following command:

nano docker-compose.yml

1. After running the command paste the following into nano to write into the file:

services:

prometheus:

image: prom/prometheus

container\_name: prometheus

command:

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

- '--storage.tsdb.retention.time=4w'

ports:

- 9090:9090

restart: unless-stopped

volumes:

- ./prometheus-config.yml:/etc/prometheus/prometheus-config.yml

- prom\_data:/prometheus

volumes:

prom\_data:

1. file by using control O and then exit with control X.
2. Create a file called prometheus-config.yml in the Prometheus folder. This can be done with the command:

nano prometheus-config.yml

1. After running the command paste the following into nano to write into the file:

global:

scrape\_interval: 15s

scrape\_timeout: 10s

evaluation\_interval: 15s

alerting:

alertmanagers:

- static\_configs:

- targets: []

scheme: http

timeout: 10s

api\_version: v2

scrape\_configs:

- job\_name: prometheus

honor\_timestamps: true

scrape\_interval: 15s

scrape\_timeout: 10s

metrics\_path: /metrics

scheme: http

static\_configs:

- targets:

- localhost:9090

- job\_name: hardwareMonitoring

static\_configs:

- targets: ['host.docker.internal:9100']

- targets: ['145.93.88.194:9100']

- job\_name: spark

metrics\_path: /metrics

static\_configs:

- targets: ['host.docker.internal:8080']

1. Save the changes to the file by using control O and then exit with control X.
2. Use the terminal to go into the file with the folder with the docker-compose.yml file and run the flowing command to start Prometheus:

docker compose up -d

1. To verify that Prometheus is running go to <http://localhost:9090/> there should be an Ui allowing you to run queries and check the connection with hardware monitoring.

# Loki+Promtail

Promtail is used to collect logs from Apache spark and send them to Loki while Loki is used to store logs and send them to Grafana.

1. Create a folder for storing docker compose files this will be used to store the files well be using to run Prometheus and other aps in docker.
2. Create a folder in the original storage folder called Loki+Promtail or another descriptive name.
3. Create a file in the Prometheus folder create a file called docker-compose.yml . The easiest way to do this is by opening a terminal going to the Prometheus folder and then running the following command:

nano docker-compose.yml

1. After running the command paste the following into nano to write into the file:

services:

loki:

image: grafana/loki

container\_name: loki

volumes:

- ./loki-config.yaml:/mnt/config/loki-config.yaml

- loki\_data:/loki/data

user: "root"

ports:

- "3100:3100"

command:

- '--config.file=/mnt/config/loki-config.yaml'

promtail:

image: grafana/promtail

container\_name: promtail

volumes:

- ./promtail-config.yaml:/mnt/config/promtail-config.yaml

- lokipromtail\_shared-logs:/var/log

depends\_on:

- loki

command: -config.file=/mnt/config/promtail-config.yaml

volumes:

lokipromtail\_shared-logs:

external: true

loki\_data:

1. Save the changes to the file by using control O and then exit with control X.
2. Create a file called promtail-config.yaml in the Loki+Promtail folder. This can be done with the command:

nano promtail-config.yaml

1. After running the command paste the following into nano to write into the file:

server:

http\_listen\_port: 9080

grpc\_listen\_port: 0

positions:

filename: /tmp/positions.yaml

clients:

- url: http://host.docker.internal:3100/loki/api/v1/push

scrape\_configs:

- job\_name: system

static\_configs:

- targets:

- localhost

labels:

job: varlogs

\_\_path\_\_: /var/log/\*log

1. Save the changes to the file by using control O and then exit with control X.
2. Create a file called loki-config.yaml in the Loki+Promtail folder. This can be done with the command:

nano loki-config.yaml

1. After running the command paste the following into nano to write into the file:

auth\_enabled: false

server:

http\_listen\_port: 3100

grpc\_listen\_port: 9096

common:

instance\_addr: 127.0.0.1

path\_prefix: /tmp/loki

replication\_factor: 1

ring:

kvstore:

store: inmemory

query\_range:

results\_cache:

cache:

embedded\_cache:

enabled: true

max\_size\_mb: 100

schema\_config:

configs:

- from: 2020-10-24

store: tsdb

object\_store: filesystem

schema: v13

index:

prefix: index\_

period: 24h

compactor:

working\_directory: data/retention

compaction\_interval: 10m

retention\_enabled: true

retention\_delete\_delay: 2h

retention\_delete\_worker\_count: 150

delete\_request\_store: filesystem

storage\_config:

tsdb\_shipper:

active\_index\_directory: data/index

cache\_location: data/index\_cache

limits\_config:

retention\_period: 744h

ruler:

alertmanager\_url: http://localhost:9093

analytics:

reporting\_enabled: false

1. Save the changes to the file by using control O and then exit with control X.
2. Use the terminal to go into the file with the folder with the docker-compose.yml file and run the flowing command to start Prometheus:

docker compose up -d

1. To verify that Loki is running go to <http://localhost:3100/>ready there should be a website with a message.

# Grafana

Grafana is used to properly display our gathered monitoring data through a web interface.

Below are the steps required to start Grafana on a mac:

1. In the Docker folder created when installing Prometheus create a folder called Grafana (this is to make organizing easier).
2. In the Grafana folder create a file called docker-compose.yml . The easiest way to do this is by opening a terminal going to the Prometheus folder and then running the following command:

nano docker-compose.yml

1. After running the command paste the following into the nano terminal:

version: "3.8"

services:

grafana:

image: grafana/grafana

container\_name: grafana

restart: unless-stopped

ports:

- '3000:3000'

volumes:

- grafana-storage:/var/lib/grafana

volumes:

grafana-storage: {}

1. Save the changes to the file using control X.
2. In the Grafana folder run the following command to start Grafana:

docker compose up -d

1. To verify that Grafana go to <http://localhost:3000/>. Input the username admin and password admin to access Grafana.
2. To connect Prometheus to Grafana go to data sources > Add new data source select Prometheus and then in connection write:

http://host.docker.internal:9090

1. Save and test the connection.
2. To connect Loki to Grafana go to data sources > Add new data source select Loki and then in connection write:

http://host.docker.internal:3100

1. Save and test the connection.