Stock Market time series analysis using OpenStack, Gnocchi and Grafana

Cloud Computing final project - Prof. Carlo Vallati

Leonardo Turchetti
Lorenzo Tonelli
Ludovica Cocchella
Rambod Rahmani

Msc. in Artificial Intelligence and Data Engineering

May 23, 2021

Contents

1 Introduction

The project focused on deploying Gnocchi - an open-source time series database - and Grafana - an open source analytics and monitoring solution - on the preexisting OpenStack installation. After deployment, some preliminary tests were carried out in order to make sure everything was running smoothly. Finally, Gnocchi metrics were created and populated using Stock Market data. Grafana was then used to visualize the data and extract useful statistics.

In what follows, the deployment procedures and the ad-hoc required configurations are detailed. Just for reference, the preexisting OpenStack installation consists of:

```
172.16.3.218    Juju Controller
172.16.3.238    Compute Node 0/OpenStack Controller
172.16.3.177    Compute Node 1
172.16.3.174    Compute Node 2
172.16.3.227    Compute Node 3
```

The entire codebase is available at https://github.com/lorytony/CloudComputing.

2 Gnocchi Deployment

The Gnocchi database was deployed[1] to compute node 0 in a container using Juju

```
$ juju deploy --to lxd:0 cs:gnocchi
$ juju deploy --to lxd:0 cs:memcached
$ juju add-relation gnocchi mysql
$ juju add-relation gnocchi memcached
$ juju add-relation gnocchi keystone
$ juju add-relation gnocchi ceph-mon
```

[1] https://jaas.ai/gnocchi/37

Right after deployment, some preliminary tests - creating a metric, pushing and reading measurements - were performed using Gnocchi REST API to check if the deployment was successful.

3 Grafana Deployment

Grafana was deployed[2] to compute node 0 using Juju

```
$ juju deploy cs:grafana
```

The choice was made not to deploy Grafana inside a container in order to be able to easily access its web interface and avoid further configurations related to port forwarding.

3.1 Gnocchi Datasource

The Gnocchi datasource was installed via grafana.net[3]:

```
$ grafana-cli plugins install gnocchixyz-gnocchi-datasource
$ systemctl restart grafana-server
```

4 Fetching Stock Market Data

In order to fetch stock market data a Python3 script was written using the Yahoo! Finance[4] API:

Listing 1: fetch_stock_prices.py

```
#!/usr/bin/env python

##
# Retrieves last minute prices for the stock tickers defined in
# tickers[]. The retrieved prices are pushed to the corresponding
# stock metric in Gnocchi DB.
##
import json
import datetime
import yfinance as yf
import urllib.request
import dateutil.parser

__author__ = "Leonardo Turchetti, Lorenzo Tonelli, Ludovica ..."
__copyright__ = "Copyright (C) 2021 Leonardo Turchetti, Lorenzo ..."
__license__ = "GPLv3"

# keystone token
keystone_token = "gAAAAABgo3ljkfH41aeXCgwj1Ois7rLwtji13hI3MbAXpJ..."

# stock tickers to be retrieved
tickers = ["ENEL.MI", "ISP.MI", "STLA", "ENI.MI", "RACE.MI", ...];
tickers_metrics = ["faafee03-ec51-4929-b530-0452eef75464", ...]
```

[2] https://jaas.ai/grafana/40
[3] https://grafana.com/grafana/plugins/gnocchixyz-gnocchi-datasource
[4] https://finance.yahoo.com/

```
24
25 while True:
26   for i in range(len(tickers)):
27   # retrieve last minute price
28   msft = yf.Ticker(tickers[i])
29   print("Processing: " + msft.info['shortName'])
30   pricesDF = msft.history(period="1d", interval="1m")
31
32   # yfinance api timeouts might result in empty dataframes
33   if not pricesDF.empty:
34     # extract new highest price and timestamp
35     priceDateString = str(pricesDF.index[-1])
36     priceValue = pricesDF['High'][-1]
37     priceDate = dateutil.parser.isoparse(priceDateString)
38     print(priceDate.strftime("%Y-%m-%dT%H:%M:%S") + " - " + ...)
39
40     # push data to gnocchi
41     conditionsSetURL = "http://252.3.238.176:8041/v1/metric/" + ...
42     newConditions = [{"timestamp": priceDate.strftime( ...
43     params = json.dumps(newConditions).encode('utf8')
44     req = urllib.request.Request(conditionsSetURL, data=params, ...
45     response = urllib.request.urlopen(req)
46     print(response.read().decode('utf8'))
```

4.1 Docker Container

5 Grafana Dashboard

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam aliquam justo eget nunc condimentum, eget iaculis justo venenatis. Aenean id tellus at velit vestibulum tempor nec eu mi. Maecenas lobortis eu mi quis condimentum. Maecenas mi ipsum, semper at felis in, consequat lobortis lorem. Ut imperdiet, ante mattis interdum tristique, orci leo feugiat lorem, facilisis volutpat diam tortor ac quam. Integer faucibus, odio sed consequat sagittis, nunc mi aliquet turpis, ut laoreet velit dolor non nisi. Vivamus vitae libero a est feugiat iaculis ac id mauris.

5.1 Stock Market data Statistics

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam aliquam justo eget nunc condimentum, eget iaculis justo venenatis. Aenean id tellus at velit vestibulum tempor nec eu mi. Maecenas lobortis eu mi quis condimentum. Maecenas mi ipsum, semper at felis in, consequat lobortis lorem. Ut imperdiet, ante mattis interdum tristique, orci leo feugiat lorem, facilisis volutpat diam tortor ac quam. Integer faucibus, odio sed consequat sagittis, nunc mi aliquet turpis, ut laoreet velit dolor non nisi. Vivamus vitae libero a est feugiat iaculis ac id mauris.