PS AIS – DEMO APP

Installation Instructions

Contents

1 Installation 2

2 Directory Structure 2

2.1 Code 2

2.1.1 converter 2

2.1.2 es-dumper 2

2.1.3 parser 2

2.1.4 price-predictor 3

2.1.5 sentiment-analyzer 3

2.1.6 twitter-scraper 4

2.1.7 yahoo-finance-scraper 4

2.2 Deployment 4

2.2.1 node-scraper 4

2.2.2 node1 4

2.2.3 provisioning 5

2.3 Documentation 5

3 General Deployment 5

3.1 Get the Files 5

3.1.1 Install Subversion 5

3.1.2 Checkout the Project Repository 6

3.2 Set up your Virtual Machine 6

3.2.1 Install Vagrant & Virtual Box 6

3.2.2 Run the Virtual Machine 6

3.2.3 Root Access 7

3.3 Use your Services 7

3.3.1 Available Services 7

3.3.2 Start Services 8

3.4 Backups 9

3.4.1 Backup Elasticsearch data 9

3.4.2 Restore Elasticsearch data 9

4 Configuration & Expansion 9

4.1 Further Nodes 9

4.1.1 Ressources Configuration 10

4.1.2 Services Configuration 10

4.2 Deployment with Spark Standalone Cluster 11

4.2.1 Spark Standalone Cluster 11

4.2.2 Getting started with Spark Standalone Cluster 11

# Installation

Our goal was to create an easy installation process so that only a few steps are necessary to finish the installation. But first, let’s have a brief look to our project structure.

# Directory Structure

## Code

This subdirectory contains all source code that is necessary for our project. These directories are split into the certain logical parts sentiment analysis, price predictor, Twitter scraper, Yahoo Finance scraper and other tools to fetch data out of Elasticsearch. The following chapter shows a short overview of each subdirectory.

### ****converter****

├── converter

│   ├── PSAIS-HHN-Trainingsdata

│   │   ├── Readme.txt

│   │   ├── converter.py

│   │   └── results

│   └── Sentiment140-Trainingsdata

│   ├── Readme.txt

│   ├── converter.py

│   └── results

The converter directory contains files, which are used to convert the hand classified trainings data as well as trainings data classified by Sentiment140 into our specific trainings data format. E.g. each positive classified tweet is represented in one line within the file positive.txt.

### ****es-dumper****

├── es-dumper

│   ├── config.example.yaml

│   ├── es\_to\_csv.py

│   └── requirements.txt

The Elasticsearch dumper contains files that fetches data out of Elasticsearch and stores them into a CSV file. This is quite useful if you want to validate scraped tweets in other tools like KNIME.

### parser

├── parser

│   ├── requirements.txt

│   └── tweets\_csv\_parser.py

The tweet parser contains files that stream data out of twitter into a CSV file. With this file we fetched tweets and stored them into certain CSV files. These tweets were classified manually and are used to train the Naïve Bayes classifier.

### price-predictor

├── price-predictor

│   ├── config.example.yaml

│   ├── custom\_math.py

│   ├── price\_predictor.py

│   └── requirements.txt

The price predictor directory contains files to create a prediction of the stock market price. These predictions are stored in Elasticsearch and can be displayed in Grafana or Kibana.

### sentiment-analyzer

├── sentiment-analyzer

│   ├── aggregator.py

│   ├── classifiers

│   │   ├── \_\_init\_\_.py

│   │   ├── naive\_bayes

│   │   │   ├── \_\_init\_\_.py

│   │   │   ├── corpus

│   │   │   │   ├── negative.txt

│   │   │   │   ├── neutral.txt

│   │   │   │   └── positive.txt

│   │   │   ├── naive\_bayes.py

│   │   │   └── stop\_words.txt

│   │   └── polarity

│   │   ├── \_\_init\_\_.py

│   │   └── polarity.py

│   ├── config.example.yaml

│   ├── config.yaml

│   ├── es\_scripts

│   │   ├── delete\_field.groovy

│   │   └── polarity\_to\_class.groovy

│   ├── es\_snippets

│   │   ├── delete\_field.sh

│   │   └── test\_aggregation.sh

│   ├── requirements.txt

│   ├── sentiment\_analyzer.py

│   ├── sentiment\_analyzer\_spark.py

│   ├── sentiment\_check.py

│   └── spam\_filter.txt

The sentiment analyzer contains two different classifiers, the polarity classifier implemented with the python text mining library TextBlob and a Naïve Bayes classifier implemented with the python natural language processing library NLTK. The Naïve Bayes subdirectory contains files that are required to train and use the classifier like training datasets or stop words.

### twitter-scraper

├── twitter-scraper

│   ├── config.example.yaml

│   ├── config.yaml

│   ├── requirements.txt

│   └── twitter\_scraper.py

The Twitter scraper contains files which were used to stream tweets with the Twitter Stream API in Elasticsearch.

### yahoo-finance-scraper

└── yahoo-finance-scraper

├── config.example.yaml

├── config.yaml

├── finance\_scraper.py

└── requirements.txt

The Yahoo! Finance scraper contains files which were used to get the data out of the stock exchange market into Elasticsearch.

## Deployment

### node-scraper

├── node-scraper

│   └── config

│   ├── sentiment-analyzer.yaml

│   ├── twitter-scraper.yaml

│   ├── vagrant.yaml

│   └── yahoo-finance-scraper.yaml

### node1

├── node1

│   ├── Vagrantfile

│   └── config

│   └── vagrant.yaml

### provisioning

└── provisioning

├── grafana.db

├── private\_key

├── private\_key.ppk

├── provisioner.sh

└── upstart

├── grafana.conf

├── kibana.conf

├── sentiment-analyzer.conf

├── twitter-scraper.conf

└── yahoo-finance-scraper.conf

## Documentation

In this subdirectory you can find this installation instruction.

# General Deployment

This chapter provides you all information needed for running the demo app. All instructions and commands should be performed or executed while you're on the server you want to use for the demo app.

If there are commands, that can be executed from any pc in the same network (or by establishing a VPN-connection), e.g. through an SSH-session, it will be mentioned in the affected chapter.

## Get the Files

Subversion (SVN) is a software versioning and revision control system that is required for our project. We use it to maintain current and historical versions of files such as source code, deployment and documentation.

### Install Subversion

If Subversion isn’t already installed on your local system or you’re running an obsolete version of SVN you should follow the steps on the official installation guide.

<https://subversion.apache.org/packages.html>

Done? Perfect! Let’s try if the installation was successful by typing this into a terminal.

$ svn --version

### Checkout the Project Repository

Now, navigate to a directory in your terminal where you want to store the repository (e.g. /etc/repositories/ in Linux) and type this into your terminal.

$ svn checkout https://redmine.hs-heilbronn.de/subversion/win-psais-2016ss-demo

Log in with your university credentials and that’s it. Subversion now creates a copy of the repository on your local system.

## Set up your Virtual Machine

Vagrant is a command line utility to create, manage and configure virtual environments. We use Vagrant to deploy our software on virtual machines. You can read the full documentation at <https://www.vagrantup.com/docs/>.

### Install Vagrant & Virtual Box

If the latest version of Vagrant isn't already installed, download and install or update the proper Vagrant package for your operating system by following the instructions on the website:

<https://www.vagrantup.com/downloads.html>

Also make sure, that you have VirtualBox installed, as it's a standard backend provider for Vagrant. It can be downloaded from the following website:

<https://www.virtualbox.org/wiki/Downloads>.

### Run the Virtual Machine

Before running the Virtual Machine, please set up your configuration as described in 4.1.1 Ressources Configuration.

Navigate in your terminal to path/to/repository/deployment/node1 and type this into your terminal.

$ vagrant up --provider=virtualbox

Vagrant now starts your virtual machine with all specifications that are made in the Vagrantfile and config folder (see 4.1.1 Ressources Configuration). This will install MapR and all required programs within your virtual machine, so it may take some time. If the process is finished, the virtual machine should be up and running. You can check the status of all set up virtual machines by typing this into your terminal.

$ vagrant global-status

You should see any virtual machine, that was set up with Vagrant and that has not been 'destroyed' yet.

### Root Access

To establish an SSH session and start services with root privileges (which is highly recommended) you need to connect with the required private key. This key is available in the file path/to/repository/deployment/provisioning/private\_key

to get root access on Unix-based systems e.g. Linux or OSX. For Windows use the file

path/to/repository/deployment/provisioning/private\_key.ppk

Type the following into your terminal to connect with a running(!) Virtual Machine through SSH (accessible by any pc that is connected to the same network, e.g. via VPN).

$ ssh -i path/to/repository/deployment/provisioning/private\_key vagrant@[IP address of the virtual machine]

Default SSH user and password is both set to “*vagrant*”.

## Use your Services

To start your services, we will need another virtual machine. The folder is already in your local repository. First, create a shortcut (as described in 4.1 Further Nodes) to the Vagrantfile of node1 into the directory path/to/repository/deployment/node-scraper and specify your services and configuration (as described in 4.1.1 Ressources Configuration and 4.1.2 Services Configuration).

Start the Virtual Machine with Vagrant (as described in 3.2.2 Run & access the Virtual Machine). The services are now installed on your virtual machine.

### Available Services

The configuration of the available services is described in detail in 4.1.2 Services Configuration.

#### twitter-scraper

From now, all relevant tweets will be stored in your Elasticsearch database (index 'twitter'). Change the keyword by replacing the line below “track” in the file "twitter-scraper.yaml" or add another one by placing it in a new line below the other keyword, to scrape for a specific company e.g. –google

#### yahoo-finance-scraper

From now, all relevant stock market data will be stored in your Elasticsearch database (index 'stock\_data'). You can change the stocks by replacing the line below “ticker\_symbols” in the file “yahoo-finance-scraper.yaml" or add another stock by placing it in a new line below the other keyword, e.g.

- GOOG

#### sentiment-analyzer

The Analyzer creates a sentiment analysis based on the classifier configured in the file “sentiment-analyzer.yaml”. There are two classifier available, TextBlob a simple polarity classifier and a Naive Bayes classifier. If you start the sentiment analyzer service, the results are stored in your Elasticsearch database.

#### kibana

Kibana displays the results from Elasticsearch. Kibana is able to show certain tweets with its analyzed and classified values in a very simple way. Kibana is one out of two ways to display our results.

Kibana is much better than Grafana in showing detailed data of each tweet and its analyzed values.

A brief overview of Kibana can be found here:

Kibana: <https://www.elastic.co/de/products/kibana>

Kibana will run by default on port 5601 http://<Server IP>:5601

#### grafana

The second way to display our results is Grafana. Grafana show the result with a higher aggregation than Kibana and is able to display more than one y-axis. This is necessary to display price prediction, sentiment analysis and the stock data.

If you need a view of a higher aggregation, Grafana shows analyzed, scraped and predicted data much better than Kibana.

A brief overview of Grafana can be found here:

Grafana: <http://grafana.org/features/>

Grafana will by default run on port 3000 http://<Server IP>:3000

To login to Grafana use the default user name and password which is both“*admin*”.

### Start Services

Before you start a service, make sure you've read 4.1.2 Services Configuration and followed the instructions for configuring.

First, get access to your virtual machine by establishing an SSH session (see chapter 3.2.2 Run & access the Virtual Machine) using the private key (see chapter 3.2.3 Root Access).

Once the SSH session is established, you’re able to start the services by typing this into your terminal.

$ service [service-name] start

You can check if the service is running by typing the following into your terminal (while in SSH session).

$ service [service-name] status

That’s it! Now you’re able to start each service.

## Backups

### Backup Elasticsearch data

Use this comand to backup the data in Elasticsearch. You need to run it on the physical server (not in the VM). Make sure Elasticsearch is stopped.

rsync -ahv --delete-after --progress \  
 -e "ssh -i /Volumes/LaCie/PSAIS/PSAIS\_20161/win-psais-2016ss-demo/deployment/provisioning/private\_key" \  
 root@141.7.63.160:/var/lib/elasticsearch \  
 /Volumes/LaCie/PSAIS/PSAIS\_20161/win-psais-2016ss-demo/backup

### Restore Elasticsearch data

Run this command on the physical server. Make sure Elasticsearch is stopped.

rsync -ahv --delete-after --progress \  
 -e "ssh -i /Volumes/LaCie/PSAIS/PSAIS\_20161/win-psais-2016ss-demo/deployment/provisioning/private\_key" \  
 /Volumes/LaCie/PSAIS/PSAIS\_20161/win-psais-2016ss-demo/backup/elasticsearch/ \  
 root@141.7.63.160:/var/lib/elasticsearch

Then, run this in the VM:

chown -R elasticsearch:elasticsearch /var/lib/elasticsearch

# Configuration & Expansion

It is necessary to change some specifications in the configuration files. This chapter provides all information needed for configuring your virtual machines or expand your app by using further nodes.

## Further Nodes

You can set up more VMs parallel by creating a new folder, e.g. path/to/repository/deployment/node2 and a shortcut to *"Vagrantfile"* which is in the folder path/to/repository/deployment/node1. On Linux, type the following into your terminal while you're in the deployment folder:

$ ln -s /node1/Vagrantfile /node2/Vagrantfile

On MAC OSX simply press "⌘ + L" while marking *"Vagrantfile"* in Finder and copy the shortcut into the new folder, make sure the name of the shortcut is *"Vagrantfile"*.

Additionally, copy the folder path/to/repository/deployment/node1/config

into the folder of your new node, e.g. path/to/repository/deployment/node2.

### Ressources Configuration

In every node you want to set up, you have to change the file "vagrant.yaml" of the folder path/to/repository/deployment/node[number]/config

Open the file, replace at least the MAC address and SSH port that will be used by the virtual machine. You may also specify RAM (memory\_GB) in Gigabytes or the number of used CPUs (cpus). Save the file when you're finished.

### Services Configuration

By default all configuration files that are necessary for your services are in the folder

path/to/repository/deployment/node-scraper/config

There should be three files next to "vagrant.yaml": "twitter-scraper.yaml"; "yahoo-finance-scraper.yaml" and "sentiment-analyter.yaml".

If there are no or not all .yaml-files in the config-folder of node-scraper, you may need to copy them from path/to/repository/code. There are three folders containing a file named "config.example.yaml": twitter-scraper, yahoo-finance-scraper and sentiment-analyzer. Copy each file into the folder node-scraper/config and rename it as the folder it came from (e.g. "sentiment-analyzer.yaml").

If your Elasticsearch database isn't accessible by the default IP address/port (127.0.0.1:9200), change [database][host] of ALL(!) .yaml files in the config-folder into the desired 'IP:port'.

#### twitter-scraper

For scraping tweets it's necessary to configure your service. Open the file path/to/repository/deployment/node-scraper/config/twitter-scraper.yaml

and add (or change) your access secret, access token, consumer key and consumer secret as followed:

twitter:

access\_secret = ThisIsYourAccessSecret

access\_token = ThisIsYourAccessToken

consumer\_key = ThisIsYourConsumerKey

consumer\_secret = ThisIsYourConsumerSecret

If you want to change the keyword it looks for, you have to change the following.

scraper:

track:

- KeywordToTrack

#### yahoo-finance-scraper

You can change the company whose stock prices are saved by opening the file

path/to/repository/deployment/node-scraper/config/yahoo-finance-scraper.yaml

Change the ticker symbol (e.g. GOOG for Google) or add others by putting it in the following place:

scraper:

ticker\_symbols:

- TickerSymbolToTrack

- AnotherTickerSymbolToTrack

#### sentiment-analyzer

You can declare the used classifier for your sentiment analysis in the file

path/to/repository/deployment/node-scraper/config/sentiment-analyzer.yaml

Two classifiers are available: naive-bayes and textblob. Set the classifier in the line

classifier: DesiredClassifier

## Deployment with Spark Standalone Cluster

### Spark Standalone Cluster

Spark comes with a built-in cluster mode, which is used for the project. The following steps give a quick-start guide to start working with the Spark Standalone Cluster. Instructions based on <https://spark.apache.org/docs/1.6.1/>

In Table 1 the used configurations for the Spark Standalone Cluster are listed.

|  |  |
| --- | --- |
| **Spark Component** | **Node** |
| Spark Master | 141.7.63.156 (node1) |
| Spark Slaves | 141.7.63.156 (node1)  141.7.63.161 (node2) |

Table 1: Used configurations for Spark Standalone Cluster

### Getting started with Spark Standalone Cluster

1. Login via SSH (as described in 3.2.3 Root Access) to your Virtual Machine, on which you want to execute the Spark Master.
   1. Navigate to 'conf' folder of Spark:

$ cd /opt/spark/conf/

1. Check which slaves are “registered” to your Spark Standalone Cluster by typing the following command into your terminal.

$ cat slaves

If the ‘slaves’ file doesn’t exist, then you have to create it using the file ‘slaves.template’ and save it within the same folder as the file ‘slaves’

1. To add a slave, you have to put its IP-address to the configuration file ‘slaves’
   1. Use your favoured text-editor, e.g. nano to open the ‘slaves’ file
   2. Enter a line break
   3. Enter your slaves IP-address(es) (🡪 each in a new line)
   4. Save and exit
2. Execute to start/stop Spark Master:
   1. Navigate to your Spark folder

$ cd /opt/spark/

* 1. Execute to start Spark Master:

$ ./sbin/start-master.sh

* 1. Execute to stop Spark Master:

$ ./sbin/stop-master.sh

1. Login via SSH (as described in 3.2.3 Root Access) to your Virtual Machine/s, on which you want to execute a Spark Slave.
   1. Navigate to Spark folder:

$ cd /opt/spark/

* 1. Execute to start a Spark Slave:

$ ./sbin/start-slave.sh <spark-master-URL>

note: <master-spark-URL> (hint: inclusive port (default: 7077)) can be found using http://<IP-of-your-spark-master>>:8080

* 1. Repeat steps 2.a. and 2.b. for every Spark slave node

1. Check your Spark Standalone Cluster via the Web-UI:  
   http://<IP-of-your-spark-master>>:8080