VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

MINISTERSTVO VNITRA ČESKÉ REPUBLIKY

Příjemci podpory:

Vysoké učení technické v Brně
Fakulta informačních technologií

Poskytovatel:

Ministerstvo vnitra ČR

***Integrovaná platforma pro zpracování digitálních dat z bezpečnostních incidentů* (TARZAN)**

Identifikační kód VI20172020062

Název předkládaného výsledku: tarzan-docker-infrastructure

Typ výsledku dle UV č. 837/2017	Evidenční číslo (příjemce)	Rok vzniku
R software		**2018**
ISBN-ISSN	**Webový odkaz na výsledek**	**Kde a kdy publikováno**
	https://www.fit.vut.cz/research/product/586/	

Anotace k výsledku:

Sada Docker obrazů s jednotlivými technologiemi platformy TARZAN (jako je Hadoop, Spark, či Cassandra) a nástroj pro integraci aplikací/komponent platformy běžících nad těmito technologiemi. Nástroj poskytuje prostředky pro snadnou konfiguraci a integraci technologií, jejich nasazení v rámci výpočetního clusteru a škálovatelnost dle aktuálních požadavku a volných prostředků.

Řešitelský tým: *Petr Matoušek (manažer a hlavní řešitel), Marek Rychlý (realizační tým)*

tarzan-docker-infrastructure

Docker Infrastructure for TARZAN Platform

Technical Documentation

RNDr. Marek Rychlý, Ph.D.

Tarzan Project Documentation
Brno Univeristy of Technology, Faculty of Information Technology

Docker Infrastructure for TARZAN Platform
Technical Documentation

Marek Rychlý

Brno University of Technology
rychly@fit.vutbr.cz

Abstract. A set of Docker images providing individual TARZAN platform technologies (such as Hadoop, Spark, and Cassandra) and an integration tool for applications/components utilising these technologies. The tool provides means of easy configuration and integration of the technologies, deployment within a computing cluster, and scalability according to current requirements and available resources. This document outlines technical aspects of the system.

1 Utilized Technologies

1.1 Distributed Computing

- Apache Spark – batch and stream processing
- Apache Hadoop – MapReduce batch processing, a platform base

1.2 Data Storage

- Apache Cassandra – a NoSQL database
- Apache HDFS – a file-system

1.3 Communication

- Apache Kafka – a message queue broker

1.4 API / Development Tools

- Apache Livy – REST API for Apache Spark
- Apache Zeppelin – Web UI for various distributed computation/data processing interpreters and data visualisation
- Halyard SDK and WebApps – horizontally scalable triple store with support for named graphs
- Plaso – tools for automatic creation of timelines to support digital forensic investigators/analysts
- Timeline Analyzer – a framework for efficient analysis of social network profiles and other related data

2 Acknowledgements

This work was supported by the Ministry of the Interior of the Czech Republic as a part of the project Integrated platform for analysis of digital data from security incidents VI20172020062.

tarzan-docker-infrastructure

Docker Infrastructure for TARZAN Platform

Installation Guide

RNDr. Marek Rychlý, Ph.D.

Tarzan Project Documentation
Brno Univeristy of Technology, Faculty of Information Technology

Docker Infrastructure for TARZAN Platform
Installation Guide

Marek Rychlý

Brno University of Technology
rychly@fit.vutbr.cz

Abstract. A set of Docker images providing individual TARZAN platform technologies (such as Hadoop, Spark, and Cassandra) and an integration tool for applications/components utilising these technologies. The tool provides means of easy configuration and integration of the technologies, deployment within a computing cluster, and scalability according to current requirements and available resources. This document provides an installation guide to the system.

1 Requirements

For the installation, both hardware and software requirements must be considered.

1.1 Hardware

- single-node or multi-node cluster
- homogeneous or heterogeneous hardware (nodes with high strength in CPUs, memory, or stotage)
- single (linux) or multiple (linux/windows/mac) operating systems

1.2 Software

- installed Docker
- installed docker-compose

2 Setup

To setup the system before the installation, the following actions should be performed:

- to build required components
- to check docker-compose.yml for service volumes that utilise built artefacts of the components (these artefacts need to be distributed to particular nodes running the corresponding service containers)

3 Deployment

The system supports several types of deployments as described below.

3.1 Single Node – Zeppelin Standalone

Just run:

```
./start-services-standalone.sh
```

Then, access Zeppelin WebUI on localhost on http://localhost:8080/.

3.2 Multiple Nodes – YARN, HDFS, Spark, Zeppelin

Run the platform services (see the title) including Treafik reverse-proxy server and load-balancer:

```
./start-services.sh
```

Then, you can access the following services:

- Zeppelin WebUI on https://localhost:8443/zeppelin/
- HDFS NameNode on https://localhost:8443/hdfs/
- YARN Resource Manager on https://localhost:8443/yarn/
- Spark History Server on https://localhost:8443/spark/
- Traefik Monitoring on https://localhost:8443/traefik/

3.3 Docker Swarm

The platform services can run in a Docker Swarm, see comments in docker-compose.yml file in the root directory of the project.

4 Management

If needed, start also Portainer Docker management to control the platform services:

```
./start-services-portainer.sh
```

To access the service Portainer Management, go to https://localhost:8443/portainer/.

5 Acknowledgements

This work was supported by the Ministry of the Interior of the Czech Republic as a part of the project Integrated platform for analysis of digital data from security incidents VI20172020062.

tarzan-docker-infrastructure

Docker Infrastructure for TARZAN Platform

User Guide

RNDr. Marek Rychlý, Ph.D.

Tarzan Project Documentation
Brno Univeristy of Technology, Faculty of Information Technology

Docker Infrastructure for TARZAN Platform
User Guide

Marek Rychlý

Brno University of Technology
rychly@fit.vutbr.cz

Abstract. A set of Docker images providing individual TARZAN platform technologies (such as Hadoop, Spark, and Cassandra) and an integration tool for applications/components utilising these technologies. The tool provides means of easy configuration and integration of the technologies, deployment within a computing cluster, and scalability according to current requirements and available resources. This document provides a user guide to the system.

1 Usage

The services can be utilised by platform applications/components, e.g., by

- Network Traces Analysis Using Apache Spark on https://github.com/nesfit/Tarzan – network traces analysis using Apache Spark distributed system (for a technical report, see [1]).
- PySpark Plaso on https://github.com/nesfit/pyspark-plaso – distributed extraction of timestamps from various files using extractors adapted from the Plaso engine to Apache Spark

2 Example: PCAP Analysis in Zeppelin Notebook (Network Traces Analysis Using Apache Spark)

This example demonstrates the Network Traces Analysis Using Apache Spark by using the first platform application/component mentioned above.

The example is based on the original sample from the GitHub repository in https://github.com/nesfit/Tarzan/blob/master/Java/zeppelin-note.

Go to the Zeppelin WebUI and create a new Spark note in the Notebook. Then, create and run the following paragraphs:

```
%spark
import org.ndx.tshark.scala.TShark
val packets = TShark.getPackets(sc,
    "hdfs://namenode/data/http_m2.json")
val smtpPackets = TShark.getPackets(sc,
    "hdfs://namenode/data/smtp.json")
```

```
TShark.registerHttpHostnames("httpHostnames",
    packets, spark)
TShark.registerKeywords("keywords", packets,
    List("sme.sk", "site.the.cz"), spark, sc)
TShark.registerDnsData("dnsData", packets, spark)
TShark.registerFlowStatistics("flowStatistics",
    packets, spark)
TShark.registerFlowStatistics("smtpFlowStatistics",
    smtpPackets, spark)
```

Get URLs from HTTP headers.

```
%sql
select url, count(*) from httpHostnames group by url
```

Get keywords.

```
%sql
select keyword, count from keywords
```

Get DNS query record types.

```
%sql
select recordType, count(*) from dnsData where
    recordType != "" group by recordType
```

Get a timeline of flows (flows/time).

```
%sql
select from_unixtime(unix_timestamp(first,
    'yyyy-MM-dd HH:mm:ss.SSS'), 'yyyy-MM-dd HH:mm'),
    count(*)
from flowStatistics group by first order by first
```

Get a timeline of packets (packets/time).

```
%sql
select from_unixtime(unix_timestamp(first,
    'yyyy-MM-dd HH:mm:ss.SSS'), 'yyyy-MM-dd HH:mm'),
    sum(packets)
from flowStatistics group by first order by first
```

Get a timeline of data (bytes/time).

```
%sql
select from_unixtime(unix_timestamp(first,
    'yyyy-MM-dd HH:mm:ss.SSS'), 'yyyy-MM-dd HH:mm'),
    sum(bytes)
from flowStatistics group by first order by first
```

Get a timeline of HTTP traffic (packets/time).

```
%sql
select from_unixtime(unix_timestamp(first,
    'yyyy-MM-dd_HH:mm:ss.SSS'), 'yyyy-MM-dd_HH:mm') as
    time, sum(packets)
from flowStatistics where service == "80" group by
    time order by time
```

Get a timeline of HTTPS traffic (packets/time).

```
%sql
select from_unixtime(unix_timestamp(first,
    'yyyy-MM-dd_HH:mm:ss.SSS'), 'yyyy-MM-dd_HH:mm') as
    time, sum(packets)
from flowStatistics where service == "443" group by
    time order by time
```

Get a size of HTTP and HTTPS traffic.

```
%sql
select service, sum(packets) from flowStatistics
    where service == "80" or service == "443" group by
    service
```

Get a size of LAN and WAN traffic.

```
%sql
select lanWan, count(*) from flowStatistics where
    lanWan == "lan" or lanWan == "wan" group by lanWan
```

Get domains from DNS requests.

```
%sql
select domain, count(*) from dnsData where domain !=
    "" and isResponse == false group by domain limit 10
```

Get the email traffic structure.

```
%sql
select email, sum(bytes) from smtpFlowStatistics
    where email != "" group by email
```

Get web-servers producing the most traffic.

```
%sql
select srcAddr, sum(bytes) from flowStatistics where
    srcPort == "80" or srcPort == "443" group by
    srcAddr limit 10
```

Get end-points receiving the most traffic.

```
%sql
select dstAddr, sum(bytes) from flowStatistics group
    by dstAddr limit 10
```

Get SMTP/SMTPS clients producing the most traffic.

```
%sql
select srcAddr, sum(bytes) from smtpFlowStatistics
where (email == "smtp" or email == "smtps") and
    direction == "up" group by srcAddr limit 10
```

Get SMTP/SMTPS servers producing the most traffic.

```
%sql
select srcAddr, sum(bytes) from smtpFlowStatistics
where (email == "smtp" or email == "smtps") and
    direction == "down" group by srcAddr limit 10
```

3 Example: Time-line Analysis of Events Extracted from Files in a File-system Dump (PySpark Plaso)

The TARZAN Docker Infrastructure is utilized in PySpark Plaso to deploy system components.

For sample deployment, see docker-compose.yml files in the PySpark Plaso project on https://github.com/nesfit/pyspark-plaso/tree/master/deployment/docker-compose.

The infrastructure components are also utilized in the (alternative) Kubernetes deployment on https://github.com/nesfit/pyspark-plaso/tree/master/deployment/kubernetes.

4 Acknowledgements

This work was supported by the Ministry of the Interior of the Czech Republic as a part of the project Integrated platform for analysis of digital data from security incidents VI20172020062.

References

1. Béder, M.: *Network Traces Analysis Using Apache Spark* . Master's thesis, Brno University of Technology, Faculty of Information Technology, 2018.
   URL https://www.fit.vut.cz/study/thesis/20651/

tarzan-docker-infrastructure

Docker Infrastructure for TARZAN Platform

Reference Documentation

RNDr. Marek Rychlý, Ph.D.

Tarzan Project Documentation
Brno Univeristy of Technology, Faculty of Information Technology

Docker Infrastructure for TARZAN Platform
Reference Documentation

Marek Rychlý

Brno University of Technology
rychly@fit.vutbr.cz

Abstract. A set of Docker images providing individual TARZAN platform technologies (such as Hadoop, Spark, and Cassandra) and an integration tool for applications/components utilising these technologies. The tool provides means of easy configuration and integration of the technologies, deployment within a computing cluster, and scalability according to current requirements and available resources. This document introduces a reference documentation of the system.

1 Infrastructure Definition Files

The infrastructured is defined in docker-compose.yml files on https://github.com/nesfit/tarzan-docker-infrastructure:

- docker-compose.yml – a multiple-node deployment with YARN scheduler, HDFS storage, Spark engine, and Zeppelin WebUI
- docker-compose-standalone.yml – a single-node deployment of a standalone Zeppelin WebUI with integrated Spark engine
- docker-compose-traefik.yml – a reverse proxy server for the multiple-node deployment to publish all the services in the one end-point
- docker-compose-portainer.yml – the Portainer Docker management tool to control the platform services above

For details of those deployments, see the corresponding docker-compose.yml files above.

2 Acknowledgements

This work was supported by the Ministry of the Interior of the Czech Republic as a part of the project Integrated platform for analysis of digital data from security incidents VI20172020062.