Ubuntu 16.04.x LTS and Ubuntu 18.04.x LTS Install Guide

all

Sirenia

September 20, 2019

Contents

1 Content

This document provides a Ubuntu 16.04.x LTS and 18.04.x LTS install guide. The guide can be followed for Ubuntu installation or serve as a starting point for installing on other Linux OS.
You should read the Deployment documentation beforehand, in order to understand the components and their roles.

2 Login to server

```
ssh user@<server>
sudo su
#password
cat /etc/issue
#Ubuntu 16.04.x LTS \n \l
# or
#Ubuntu 18.04.x LTS \n \l
```

3 Ensure access to repositories

If target machine has no internet, you could use a HTTP proxy. Otherwise skip this point. If your host is a mac: install squidman http://squidman.net/squidman/

```
#Open ssh tunnel from local host to enable HTTP proxy connections
ssh -R 8080:localhost:8080 root@<ip address of target machine>
#On the target machine
export http_proxy=http://localhost:8080
export https_proxy=http://localhost:8080
# with visudo add the text:
visudo
Defaults env_keep = "http_proxy https_proxy ftp_proxy"
```

4 Install Docker

On the target machine

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
| sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] \
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
apt-cache policy docker-ce
sudo apt-get install -y docker-ce
sudo systemctl status docker
```

If target machine has no internet add http(s) proxy to docker

```
nano /etc/default/docker
```

```
# Add these lines  #(maybe not needed?)
export http_proxy="http://localhost:8080"
export https_proxy="http://localhost:8080"
#Create a systemd drop-in directory for the docker service:
sudo mkdir -p /etc/systemd/system/docker.service.d
nano /etc/systemd/system/docker.service.d/http-proxy.conf
#Add these lines
[Service]
Environment="HTTP_PROXY=http://localhost:8080/"
#Flush changes:
sudo systemctl daemon-reload
#Restart Docker:
sudo systemctl restart docker
#Verify that the configuration has been loaded:
systemctl show --property=Environment docker
#Environment=HTTP_PROXY=http://localhost:8080/
```

5 Install Docker Compose

On the target machine

```
sudo curl -L https://github.com/docker/compose/releases/\
download/1.18.0/docker-compose-‘uname -s‘-‘uname -m‘ -o\
/usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
#docker-compose version 1.18.0, build 8dd22a9
```

6 Pull software

On the target machine pull some Sirenia software

```
mkdir /root/deploy
cd /root/deploy
```

Create a docker-compose file for your specific setup.

```
nano docker-compose.yml
```

You could take a base in this example. You must change at least kwanza version, cuesta version and <FQDN> of your server. You MUST use all small letters in the fqdn. eg. some.sirenia.io

```
version: '3'

services:
  kwanza:
    image: registry.gitlab.com/sirenia/dist/kwanza:v2.7.1
    restart: unless-stopped
    environment:
      KWANZA_DATABASE: pg://postgres:postgres@postgres/kwanza
      KWANZA_CERT_SUBJECTS: "<FQDN>"
      KWANZA_CERT: "/cert/cert.pem"
      KWANZA_KEY: "/cert/key.pem"
      KWANZA_SALT: kwanzified
      KWANZA_AUTH: jwt
    ports:
      - "8000:8000"    # HTTP(S)
      - "8001:8001"    # TCP (gRPC)
      - "6060:6060"    # Profiling
    volumes:
      - "/usr/local/etc/sirenia/cert:/cert"
      - "/usr/local/etc/sirenia/kwanza/conf:/etc/sirenia/kwanza"
    depends_on:
      - postgres

  cuesta:
    image: registry.gitlab.com/sirenia/dist/cuesta:v1.8.1
    restart: unless-stopped
    environment:
      CUESTA_CERT: "/cert/cert.pem"
      CUESTA_KEY: "/cert/key.pem"
      KWANZA_URL: "https://<FQDN>:8000/v1"
      KWANZA_STREAMURL: "wss://<FQDN>:8000/v1/stream"
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - "/usr/local/etc/sirenia/cert:/cert"
    depends_on:
      - kwanza

  postgres:
```

```
    image: postgres:10
    restart: always
    ports:
      - "5444:5432"
    environment:
      PGDATA: "/data"
    volumes:
      - "/root/postgresdata:/data"
```

Now pull some software from the repository and try to start the combined setup.

```
docker login registry.gitlab.com
#dist-<username> / <password>
# ... Login Succeeded
docker-compose up
<ctrl-c> (stop again)
```

7 Add a certificate

Kwanza will generate self-signed cert at startup. Alternatively copy valid cert for prod here `/usr/local/etc/sirenia/cert` It must be a valid x.509 certificate with a full trust chain to a CA in PEM format.

8 Configure Kwanza

```
cd /usr/local/etc/sirenia/kwanza/conf
nano .kwanza.yml
```

paste this

```
users:
  admin:
     d224cfd0914713837084 24f3e494f8029b456b0e559fe82ee9adb5b66a7f1e55
  john:
     d224cfd0914713837084 24f3e494f8029b456b0e559fe82ee9adb5b66a7f1e55
  jonathan:
     d224cfd0914713837084 24f3e494f8029b456b0e559fe82ee9adb5b66a7f1e55
```

9 Test

Ok, we are ready to test the complete setup

```
cd /root/deploy/
docker-compose stop
docker-compose up
```

Look for errors etc in the logs. Login to Cuesta

- https://localhost/
- user:john pass:1234

If no errors show up, we are ready to go. Start the setup as background processes.

```
docker-compose stop
docker-compose up -d
```

10 Sirenia Analytics

If you have acquired a license to the Data Driven Operational Intelligence solution Sirenia Analytics, follow the instalation guide here. You can deploy this on the same server as Cuesta and Kwanza (assuming it is sized coorectly), or on is's own. If you install on a new server, you must first install docker and docker-compose as explained above.

Create a docker-compose file for your specific setup (or add to existing).

```
nano docker-compose.yml
```

You could take a base in this example. You must change at least versions and <FQDN> of your server.

```
version: '2'
services:

  nginx-proxy:
    container_name: nginx-proxy
    image: jwilder/nginx-proxy
    ports:
      - "80:80"
    restart: always
    volumes:
      - "/var/run/docker.sock:/tmp/docker.sock:ro"
```

```
12        - "./nginx-proxy/htpasswd:/etc/nginx/htpasswd"
13
14    fluentd:
15      container_name: fluentd
16      image: registry.gitlab.com/sirenia/dist/analytics/sirenia-fluentd
            :1.0.0
17      restart: always
18      volumes:
19          - "./fluentd/etc/:/fluentd/etc/"
20          - "./fluentd/data/:/fluentd/log/"
21      ports:
22         - "8080:8080/udp"
23         - "8081:8081/udp"
24         - "8082:8082/udp"
25         - "8090:8090/tcp"
26
27    elk6:
28      container_name: elk6
29      environment:
30         ES_JAVA_OPTS: "-Xmx3024m -Xms3024m"
31         EL_JAVA_OPTS: "-Xmx256m -Xms256m"
32         VENDOR: Sirenia
33         ELASTICSEARCH_START: 1
34         LOGSTASH_START: 1
35         KIBANA_START: 1
36         VIRTUAL_HOST: my.hosts.fqdn # will be fwd by nginx proxy
37         VIRTUAL_PORT: 5601 # will be fwd by nginx proxy
38
39      image: registry.gitlab.com/sirenia/dist/analytics/sirenia-elk
            -6:6.0.1
40      restart: always
41      volumes:
42          - "./elk6/conf.d/:/etc/logstash/conf.d/"
43          - "./fluentd/data/:/etc/logstash/indata/"
44          - "./elk6/elk-data:/var/lib/elasticsearch/" #OBS: Required
               chown 991:991 elk6/elk-data/
45      expose:
46         - "5601"
```

Pull the software and initialize folder structure.

```
1  docker-compose up
```

Wait for download of software and start-up of all dockers. Is expected til give errors, as the setup have not been configured yet.

```
ctrl-c to stop
```

11 Configure Elastic Search

To configure Elastic do the following

```
chown 991:991 elk6/elk-data/
echo "vm.max_map_count=262144" >> /etc/sysctl.conf
sysctl -w vm.max_map_count=262144
cd elk6/conf.d
nano logstash-in-out.conf
```

Add this to the file

```
input {
  file {
    #All for debug
    type => "all-manatee"
    path => "/etc/logstash/indata/all.manatee*.log"
    #start_position => "beginning"
    start_position => "end"
    codec => json
  }
  file {
    #Stats for BI only
    type => "bi-manatee"
    path => "/etc/logstash/indata/stats.manatee*.log"
    #start_position => "beginning"
    start_position => "end"
    codec => json
  }
}
filter {
  #NOOP
}
output {
  if [type] == "all-manatee" {
    elasticsearch {
```

```
25        hosts => ["localhost"]
26        manage_template => false
27        index => "all-manatee"
28      }
29    }
30    if [type] == "bi-manatee" {
31      elasticsearch {
32        hosts => ["localhost"]
33        manage_template => false
34        index => "all-manatee-bi"
35      }
36    }
37  }
```

12 Configure Fluentd

To configure Fluentd do the following

```
1  cd ../../fluentd/etc/
2  nano fluent.conf
```

Add this to the file

```
1  #UDP input
2  <source>
3    @type udp
4    #8081 is stats (info-log)
5    tag manatee.8081 # required
6    format none
7    port 8081 # optional. 5160 by default
8    bind 0.0.0.0 # optional. 0.0.0.0 by default
9    message_length_limit 1MB
10 </source>
11
12 <source>
13   @type udp
14   #8082 is everything (debug-log)
15   tag manatee.8082 # required
16   format none
17   port 8082 # optional. 5160 by default
18   bind 0.0.0.0 # optional. 0.0.0.0 by default
```

```
  message_length_limit 1MB
</source>

#Filters. Everything to stdout
<filter **>
  @type stdout
</filter>

#Output
<match manatee.8081>
  @type file
  format single_value
  path          /fluentd/log/stats.manatee
  buffer_type memory
  flush_interval 0s
  append       true
</match>

<match manatee.8082>
  @type file
  format single_value
  path          /fluentd/log/all.manatee
  buffer_type memory
  flush_interval 0s
  append       true
</match>
```

13 Configure Nginx Proxy

To configure teh Nginx Proxy do the following. Change user and password according to your desired setup

```
cd ../../nginx-proxy/htpasswd/
apt install apache2-utils -y
htpasswd -nb user password >> my.hosts.fqdn
```

14 Test

Ok, we are ready to test the complete DDOI setup. Start all dockers

```
cd ../../
docker-compose up
```

Look for errors etc in the logs. Login to Sirenia Analytics

- https://my.hosts.fqdn/
- user:user pass:password

If no errors show up, we are ready to go. Start the setup as background processes. ctrl-c to stop

```
docker-compose up -d
```

Ensure that the containers are running as expected

```
docker-compose ps
```

Should produce output showing three containers running un Up state.

```
    Name                   Command               State
                                                                  Ports
------------------------------------------------------------------------------------------

elk6          /usr/local/bin/start.sh          Up      5044/tcp, 5601/
    tcp, 9200/tcp, 9300/tcp
fluentd       /bin/entrypoint.sh /bin/sh ...   Up      24224/tcp, 5140/
    tcp, 0.0.0.0:8080->8080/udp, 0.0.0.0:8081->8081/udp,
    0.0.0.0:8082->8082/udp, 0.0.0.0:8090->8090/tcp
nginx-proxy   /app/docker-entrypoint.sh  ...   Up      0.0.0.0:80->80/
    tcp
```

15 Restart Server

You should always finish an install procedure with a complete servere restart, to test that all services starts after a complete host restart

```
reboot -n
```