
Centos 7 Install Guide

all

Sirenia

September 20, 2019



Contents

1	Content	2
2	Login to server	2
3	Install Docker	3
4	Install Docker Compose	3
5	Pull software	3
6	Add a certificate	5
7	Configure Kwanza	5
8	Test	6
9	Sirenia Analytics	6
10	Configure Elastic Search	8
11	Configure Fluentd	9
12	Configure Nginx Proxy	10
13	Test	11
14	Restart Server	12

1 Content

This document provides a Centos 7 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

```
1 ssh user@<server>
2 sudo su
3 #password
4 uname -r
5 #3.10.0-957.27.2.el7.x86_64
```

3 Install Docker

On the target machine

```
1 sudo yum install -y yum-utils device-mapper-persistent-data lvm2
2 sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
3 sudo yum install -y docker-ce docker-ce-cli containerd.io
4 sudo systemctl start docker
5 sudo docker run hello-world
6 sudo systemctl status docker
```

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

4 Install Docker Compose

On the target machine

```
1 sudo curl -L "https://github.com/docker/compose/releases/download/1.23.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
2 sudo chmod +x /usr/local/bin/docker-compose
3 docker-compose --version
4 #docker-compose version 1.23.1, build b02f1306
```

5 Pull software

On the target machine pull some Sirenia software

```
1 mkdir /root/deploy
2 cd /root/deploy
```

Create a docker-compose file for your specific setup.

```
1 yum install -y nano
2 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

```
1 version: '3'
2
3 services:
4   kwanza:
5     image: registry.gitlab.com/sirenia/dist/kwanza:v2.7.1
6     restart: unless-stopped
7     environment:
8       KWANZA_DATABASE: pg://postgres:postgres@postgres/kwanza
9       KWANZA_CERT_SUBJECTS: "<FQDN>"
10      KWANZA_CERT: "/cert/cert.pem"
11      KWANZA_KEY: "/cert/key.pem"
12      KWANZA_SALT: kwanzified
13      KWANZA_AUTH: jwt
14     ports:
15       - "8000:8000" # HTTP(S)
16       - "8001:8001" # TCP (gRPC)
17       - "6060:6060" # Profiling
18     volumes:
19       - "/usr/local/etc/sirenia/cert:/cert"
20       - "/usr/local/etc/sirenia/kwanza/conf:/etc/sirenia/kwanza"
21     depends_on:
22       - postgres
23
24   cuesta:
25     image: registry.gitlab.com/sirenia/dist/cuesta:v1.8.1
26     restart: unless-stopped
27     environment:
28       CUESTA_CERT: "/cert/cert.pem"
29       CUESTA_KEY: "/cert/key.pem"
30       KWANZA_URL: "https://<FQDN>:8000/v1"
31       KWANZA_STREAMURL: "wss://<FQDN>:8000/v1/stream"
32     ports:
33       - "80:80"
34       - "443:443"
35     volumes:
```

```
36     - "/usr/local/etc/sirenia/cert:/cert"
37     depends_on:
38     - kwanza
39
40     postgres:
41     image: postgres:10
42     restart: always
43     ports:
44     - "5444:5432"
45     environment:
46     PGDATA: "/data"
47     volumes:
48     - "/root/postgresdata:/data"
```

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

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

6 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.

7 Configure Kwanza

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

paste this

```
1 users:
2   admin:
3     d224cfd091471383708424f3e494f8029b456b0e559fe82ee9adb5b66a7f1e55
```

```
3 john:
    d224cfd091471383708424f3e494f8029b456b0e559fe82ee9adb5b66a7f1e55
4 jonathan:
    d224cfd091471383708424f3e494f8029b456b0e559fe82ee9adb5b66a7f1e55
```

8 Test

Ok, we are ready to test the complete setup

```
1 cd /root/deploy/
2 docker-compose stop
3 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.

```
1 docker-compose stop
2 docker-compose up -d
```

9 Sirenia Analytics

If you have acquired a license to the Data Driven Operational Intelligence solution Sirenia Analytics, follow the installation guide here. You can deploy this on the same server as Cuesta and Kwanza (assuming it is sized correctly), or on its 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).

```
1 mkdir /root/deploy-elk
2 cd /root/deploy-elk
3 nano docker-compose.yml
```

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

```
1 version: '2'
2 services:
```

```
3
4  nginx-proxy:
5     container_name: nginx-proxy
6     image: jwilder/nginx-proxy
7     ports:
8         - "80:80"
9     restart: always
10    volumes:
11        - "/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
17            :1.0.0
18        restart: always
19        volumes:
20            - "./fluentd/etc:/fluentd/etc/"
21            - "./fluentd/data:/fluentd/log/"
22        ports:
23            - "8080:8080/udp"
24            - "8081:8081/udp"
25            - "8082:8082/udp"
26            - "8090:8090/tcp"
27
28    elk6:
29        container_name: elk6
30        environment:
31            ES_JAVA_OPTS: "-Xmx3024m -Xms3024m"
32            EL_JAVA_OPTS: "-Xmx256m -Xms256m"
33            VENDOR: Sirenia
34            ELASTICSEARCH_START: 1
35            LOGSTASH_START: 1
36            KIBANA_START: 1
37            VIRTUAL_HOST: my.hosts.fqdn # will be fwd by nginx proxy
38            VIRTUAL_PORT: 5601 # will be fwd by nginx proxy
39
40        image: registry.gitlab.com/sirenia/dist/analytics/sirenia-elk
41            -6:6.0.1
42        restart: always
43        volumes:
44            - "./elk6/conf.d:/etc/logstash/conf.d/"
45            - "./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.

```
1 ctrl-c to stop
```

10 Configure Elastic Search

To configure Elastic do the following

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

Add this to the file

```
1 input {
2   file {
3     #All for debug
4     type => "all-manatee"
5     path => "/etc/logstash/indata/all.manatee*.log"
6     #start_position => "beginning"
7     start_position => "end"
8     codec => json
9   }
10  file {
11    #Stats for BI only
12    type => "bi-manatee"
13    path => "/etc/logstash/indata/stats.manatee*.log"
14    #start_position => "beginning"
15    start_position => "end"
```



```
16     codec => json
17   }
18 }
19 filter {
20   #NOOP
21 }
22 output {
23   if [type] == "all-manatee" {
24     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 }
```

11 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
19   message_length_limit 1MB
20 </source>
21
22 #Filters. Everything to stdout
23 <filter **>
24   @type stdout
25 </filter>
26
27 #Output
28 <match manatee.8081>
29   @type file
30   format single_value
31   path          /fluentd/log/stats.manatee
32   buffer_type memory
33   flush_interval 0s
34   append        true
35 </match>
36
37 <match manatee.8082>
38   @type file
39   format single_value
40   path          /fluentd/log/all.manatee
41   buffer_type memory
42   flush_interval 0s
43   append        true
44 </match>
```

12 Configure Nginx Proxy

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

```
1 cd ../../nginx-proxy/htpasswd/
2 yum install -y httpd-tools
3 htpasswd -nb user password >> my.hosts.fqdn
```

13 Test

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

```
1 cd ../../
2 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

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

Ensure that the containers are running as expected

```
1 docker-compose ps
```

Should produce output showing three containers running un Up state.

1	Name	Command	State	Ports
2	-----			
3	elk6	/usr/local/bin/start.sh	Up	5044/tcp, 5601/tcp, 9200/tcp, 9300/tcp
4	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
5	nginx-proxy	/app/docker-entrypoint.sh ...	Up	0.0.0.0:80->80/tcp

14 Restart Server

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

```
1 reboot -n
```