Centos 7 and 8 Install Guide

October 25, 2021

Contents

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.

Login to server

```
ssh user@<server>
sudo su
#password
cat /etc/centos-release
#CentOS Linux release 7 eller 8
```

Get the Essentials

```
sudo yum -y install epel-release
sudo yum install -y htop
sudo yum install -y nano
sudo yum install -y wget
```

```
sudo wget https://github.com/bcicen/ctop/releases/download/v0.7.3/ctop-0.7.3-linux-amd64 -O /usr/local/bin/ctop
sudo chmod +x /usr/local/bin/ctop
sudo yum install -y postgresql
```

Remove non-essentials

```
systemctl stop rpcbind.service
systemctl disable rpcbind.service
systemctl stop rpcbind.socket
systemctl disable rpcbind.socket
```

For Centos 8, remove firewalld and install iptables

```
sudo systemctl stop firewalld
sudo systemctl disable firewalld
sudo systemctl mask firewalld
sudo yum install -y iptables-services
sudo systemctl start iptables
sudo systemctl start ip6tables
sudo systemctl enable iptables
sudo systemctl enable ip6tables
```

Install Docker

On the target machine

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
wget https://download.docker.com/linux/centos/7/x86_64/edge/Packages/containerd.io-1.2.6-3.3.el7.x86_64.rpm
yum install -y containerd.io-1.2.6-3.3.el7.x86_64.rpm
sudo yum install -y docker-ce docker-ce-cli containerd.io
sudo systemctl start docker
sudo docker run hello-world
sudo systemctl enable docker
sudo systemctl status docker
ctrl-c to stop
```

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

Install Docker Compose

On the target machine

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
echo TMPDIR="/opt/compose-tmp" >> /etc/environment
mkdir -p /opt/compose-tmp
echo "export PATH=/usr/local/bin:$PATH" >> /root/.bashrc
source /root/.bashrc
docker-compose --version
#docker-compose version 1.27.4, build 40524192
```

Allow inter-docker communication

```
sysctl net.bridge.bridge-nf-call-iptables=0
sysctl net.bridge.bridge-nf-call-arptables=0
sysctl net.bridge.bridge-nf-call-ip6tables=0
echo 'net.bridge.bridge-nf-call-iptables=0' >> /etc/sysctl.conf
echo 'net.bridge.bridge-nf-call-arptables=0' >> /etc/sysctl.conf
echo 'net.bridge.bridge-nf-call-ip6tables=0' >> /etc/sysctl.conf
```

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 `${HOSTNAME}` of your server. You MUST use all small letters in the fqdn. eg. some.sirenia.io

```
version: '3'

networks:
  default:
    ipam:
      driver: default
      config:
        - subnet: "172.27.0.0/24"

services:
  kwanza:
    image: registry.sirenia.io/kwanza:v2.16.2
    restart: unless-stopped
    environment:
      KWANZA_DATABASE: pg://postgres:postgres@postgres/kwanza
      KWANZA_MINTLSVERSION: 1.2
      KWANZA_CIPHERSUITES: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
         TLS_RSA_WITH_AES_128_GCM_SHA256
         TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
         TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256"
      KWANZA_PREFERSERVERCIPHERSUITES: "True"
      KWANZA_STRICTTRANSPORTSECURITY: "True"
      KWANZA_CERT_SUBJECTS: "${HOSTNAME}"
      KWANZA_CERT_DURATION: 87600h
      KWANZA_CERT: "/cert/cert.pem"
      KWANZA_KEY: "/cert/key.pem"
      KWANZA_SALT: kwanzified
      KWANZA_AUTH: jwt
      KWANZA_MAXSTREAMSPERSUBSCRIBER: 102400
      KWANZA_MAXAUTHTHROTTLEDKEYS: -1
      KWANZA_MAXTHROTTLEDKEYS: -1
    ports:
      - "8000:8000"    # HTTP(S)
      - "8001:8001"    # TCP (gRPC)
      - "127.0.0.1:6060:6060"    # Profiling to host-only
      - "127.0.0.1:8080:8080"    # Expvar to host-only
    volumes:
      - "/usr/local/etc/sirenia/cert:/cert"
      - "/usr/local/etc/sirenia/kwanza/conf:/etc/sirenia/kwanza"
    depends_on:
      - postgres

```

```
  cuesta:
    image: registry.sirenia.io/cuesta:v1.14.17
    restart: unless-stopped
    environment:
      CUESTA_CERT: "/cert/cert.pem"
      CUESTA_KEY: "/cert/key.pem"
      KWANZA_URL: "https://${HOSTNAME}:8000/v1"
      KWANZA_STREAMURL: "wss://${HOSTNAME}: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:
      - "127.0.0.1:5432:5432"
    environment:
      PGDATA: "/data"
      POSTGRES_PASSWORD: "postgres"
    volumes:
      - "/root/postgresdata:/data"
```

Configure Kwanza

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

paste this

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

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

```
cd /root/deploy
docker login registry.sirenia.io
#dist-<username> / <password>
# ... Login Succeeded
docker-compose up
<ctrl-c> (stop again)
```

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.

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://<FQDN>/`
- `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
```

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).

```
mkdir /root/deploy-elk
cd /root/deploy-elk
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'

networks:
  default:
    ipam:
      driver: default
      config:
        - subnet: "172.28.0.0/24"

services:

  nginx-proxy:
    container_name: nginx-proxy
    image: jwilder/nginx-proxy
    ports:
      - "81:443"
    restart: always
    #environment:
    volumes:
      - "/var/run/docker.sock:/tmp/docker.sock:ro"
      - "./nginx-proxy/htpasswd:/etc/nginx/htpasswd"
      - "/usr/local/etc/sirenia/cert:/etc/nginx/certs"

  aripuana-stats:
    image: registry.sirenia.io/aripuana:v1.5.1
    restart: unless-stopped
    environment:
      ARIPUANA_MINTLSVERSION: 1.2
      ARIPUANA_CIPHERSUITES: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
          TLS_RSA_WITH_AES_128_GCM_SHA256
          TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
          TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256"
      ARIPUANA_PREFERSERVERCIPHERSUITES: "True"
      ARIPUANA_STRICTTRANSPORTSECURITY: "True"
      ARIPUANA_CERT_SUBJECTS: "${HOSTNAME}"
```

```
      ARIPUANA_CERT_DURATION: 87600h
      ARIPUANA_CERT: "/cert/cert.pem"
      ARIPUANA_KEY: "/cert/key.pem"
      ARIPUANA_SALT: "fishy"
      ARIPUANA_WRITERS: 1
      ARIPUANA_PORT: 8083
      ARIPUANA_LOGNAME: "stats.manatee"
      ARIPUANA_OUTPUTDIR: "/data"
    ports:
      - "8082:8082"
      - "8083:8083"
    volumes:
      - "/usr/local/etc/sirenia/cert:/cert"
      - "./aripuana/data:/data"

  aripuana-logs:
    image: registry.sirenia.io/aripuana:v1.5.1
    restart: unless-stopped
    environment:
      ARIPUANA_MINTLSVERSION: 1.2
      ARIPUANA_CIPHERSUITES: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
          TLS_RSA_WITH_AES_128_GCM_SHA256
          TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
          TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256"
      ARIPUANA_PREFERSERVERCIPHERSUITES: "True"
      ARIPUANA_STRICTTRANSPORTSECURITY: "True"
      ARIPUANA_CERT_SUBJECTS: "${HOSTNAME}"
      ARIPUANA_CERT_DURATION: 87600h
      ARIPUANA_CERT: "/cert/cert.pem"
      ARIPUANA_KEY: "/cert/key.pem"
      ARIPUANA_SALT: "fishy"
      ARIPUANA_WRITERS: 1
      ARIPUANA_PORT: 8085
      ARIPUANA_LOGNAME: "all.manatee"
      ARIPUANA_OUTPUTDIR: "/data"
    ports:
      - "8084:8084"
      - "8085:8085"
    volumes:
      - "/usr/local/etc/sirenia/cert:/cert"
      - "./aripuana/data:/data"

  aripuana-perf:
```

```
73      image: registry.sirenia.io/aripuana:v1.5.1
74      restart: unless-stopped
75      environment:
76        ARIPUANA_MINTLSVERSION: 1.2
77        ARIPUANA_CIPHERSUITES: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
              TLS_RSA_WITH_AES_128_GCM_SHA256
              TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
              TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256"
78        ARIPUANA_PREFERSERVERCIPHERSUITES: "True"
79        ARIPUANA_STRICTTRANSPORTSECURITY: "True"
80        ARIPUANA_CERT_SUBJECTS: "${HOSTNAME}"
81        ARIPUANA_CERT_DURATION: 87600h
82        ARIPUANA_CERT: "/cert/cert.pem"
83        ARIPUANA_KEY: "/cert/key.pem"
84        ARIPUANA_SALT: "fishy"
85        ARIPUANA_WRITERS: 1
86        ARIPUANA_PORT: 8087
87        ARIPUANA_LOGNAME: "perf.manatee"
88        ARIPUANA_OUTPUTDIR: "/data"
89      ports:
90        - "8086:8086"
91        - "8087:8087"
92      volumes:
93        - "/usr/local/etc/sirenia/cert:/cert"
94        - "./aripuana/data:/data"
95
96    elk6:
97      container_name: elk6
98      environment:
99         ES_JAVA_OPTS: "-Xmx1500m -Xms1500m"
100        EL_JAVA_OPTS: "-Xmx256m -Xms256m"
101        VENDOR: Sirenia
102        ELASTICSEARCH_START: 1
103        LOGSTASH_START: 1
104        KIBANA_START: 1
105        VIRTUAL_HOST: "${HOSTNAME}" # will be fwd by nginx proxy
106        VIRTUAL_PORT: 5601 # will be fwd by nginx proxy
107        CERT_NAME: linked_for_nginx
108     image: registry.sirenia.io/sirenia-elk-7:7.2.0.1
109     restart: always
110     volumes:
111         - "./elk6/conf.d/:/etc/logstash/conf.d/"
112         - "./aripuana/data:/etc/logstash/indata/"
```

```
113         - "./elk6/elk-data:/var/lib/elasticsearch/" #OBS: Required
                chown 991:991 elk6/elk-data/
114     expose:
115        - "5601"
116
117   #elk6-readonly:
118   #  container_name: elk6-readonly
119   #  environment:
120   #     VENDOR: Sirenia
121   #     KIBANA_START: 1
122   #     VIRTUAL_HOST: "ro-${HOSTNAME}" # will be fwd by nginx proxy
123   #     VIRTUAL_PORT: 5601 # will be fwd by nginx proxy
124   #     CERT_NAME: linked_for_nginx
125   #  image: registry.gitlab.com/sirenia/dist/analytics/sirenia-elk-7-
          readonly:7.2.0.6
126   #  restart: always
```

Make sym-links for cert for proxy use

```
1 cd /usr/local/etc/sirenia/cert
2 ln -s key.pem linked_for_nginx.key
3 ln -s cert.pem linked_for_nginx.crt
```

Pull the software and initialize folder structure.

```
1 cd /root/deploy-elk
2 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
```

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

```
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
  }
  file {
    #perf for perf only
    type => "perf-manatee"
    path => "/etc/logstash/indata/perf.manatee*.log"
    #start_position => "beginning"
    start_position => "end"
    codec => json
  }
}
filter {
  #NOOP
}
output {
  if [type] == "all-manatee" {
    elasticsearch {
      hosts => ["localhost"]
      manage_template => false
      index => "all-manatee-1"
    }
  }
  if [type] == "bi-manatee" {
    elasticsearch {
      hosts => ["localhost"]
      manage_template => false
```

```
42        index => "all-manatee-1"
43      }
44    }
45    if [type] == "perf-manatee" {
46      elasticsearch {
47        hosts => ["localhost"]
48        manage_template => false
49        index => "all-manatee-perf-1"
50      }
51    }
52  }
```

Configure Nginx Proxy

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

```
cd ../../nginx-proxy/htpasswd/
yum install -y httpd-tools
htpasswd -nb user password >> <FQDN>
```

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

- http://<FQDN>:81/
- 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 five containers running un Up state.

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

aripuana-logs    aripuana run                      Up
   0.0.0.0:8084->8084/tcp, 0.0.0.0:8085->8085/tcp
aripuana-perf    aripuana run                      Up
   0.0.0.0:8086->8086/tcp, 0.0.0.0:8087->8087/tcp
aripuana-stats   aripuana run                      Up
   0.0.0.0:8082->8082/tcp, 0.0.0.0:8083->8083/tcp
elk6             /usr/local/bin/start.sh           Up      5044/tcp,
   5601/tcp, 9200/tcp, 9300/tcp
nginx-proxy      /app/docker-entrypoint.sh  ...    Up
   0.0.0.0:81->443/tcp, 80/tcp
```

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
```