Ubuntu 16.04.x and 18.04.x LTS Install Guide

March 20, 2025

Contents

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.

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
```

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"
```

Get the Essentials

```
sudo apt install -y htop
sudo apt install -y nano
sudo apt 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 apt install -y postgresql
```

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 start docker
sudo docker run hello-world
sudo systemctl enable docker
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/
```

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
docker-compose --version
#docker-compose version 1.27.4, build 40524192
```

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:
    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: 8087
      ARIPUANA_LOGNAME: "perf.manatee"
      ARIPUANA_OUTPUTDIR: "/data"
    ports:
      - "8086:8086"
      - "8087:8087"
    volumes:
      - "/usr/local/etc/sirenia/cert:/cert"
      - "./aripuana/data:/data"

  elk6:
    container_name: elk6
    environment:
       ES_JAVA_OPTS: "-Xmx1500m -Xms1500m"
       EL_JAVA_OPTS: "-Xmx256m -Xms256m"
       VENDOR: Sirenia
       ELASTICSEARCH_START: 1
       LOGSTASH_START: 1
       KIBANA_START: 1
       VIRTUAL_HOST: "${HOSTNAME}" # will be fwd by nginx proxy
       VIRTUAL_PORT: 5601 # will be fwd by nginx proxy
       CERT_NAME: linked_for_nginx
    image: registry.sirenia.io/sirenia-elk-7:7.2.0.1
    restart: always
    volumes:
        - "./elk6/conf.d/:/etc/logstash/conf.d/"
        - "./aripuana/data:/etc/logstash/indata/"
        - "./elk6/elk-data:/var/lib/elasticsearch/" #OBS: Required
           chown 991:991 elk6/elk-data/
    expose:
       - "5601"

  #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

```
1 input {
2   file {
3     #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
      index => "all-manatee-1"
    }
  }
  if [type] == "perf-manatee" {
    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

```
1  cd ../../nginx-proxy/htpasswd/
2  apt install -y apache2-utils
3  htpasswd -nb user password >> <FQDN>
```

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

- 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

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

Ensure that the containers are running as expected

```
1  docker-compose ps
```

Should produce output showing five containers running un Up state.

```
1       Name                     Command               State
                                   Ports
2  ------------------------------------------------------------------------------------------
```

```
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
```