
Ubuntu 16.04.x and 18.04.x LTS Install Guide

October 25, 2021



Contents

Login to server	2
Ensure access to repositories	2
Get the Essentials	3
Install Docker	3
Install Docker Compose	4
Pull software	4
Configure Kwanza	6
Add a certificate	7
Test	7
Sirenia Analytics	7
Configure Elastic Search	11
Configure Nginx Proxy	13
Test	13
Restart Server	14

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

```

1 ssh user@<server>
2 sudo su
3 #password
4 cat /etc/issue
5 #Ubuntu 16.04.x LTS \n \l
6 # or
7 #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/>

```

1 #Open ssh tunnel from local host to enable HTTP proxy connections
2 ssh -R 8080:localhost:8080 root@<ip address of target machine>

```

```
3 #On the target machine
4 export http_proxy=http://localhost:8080
5 export https_proxy=http://localhost:8080
6 # with visudo add the text:
7 visudo
8 Defaults env_keep = "http_proxy https_proxy ftp_proxy"
```

Get the Essentials

```
1 sudo apt install -y htop
2 sudo apt install -y nano
3 sudo apt install -y wget
4 sudo wget https://github.com/bcicen/ctop/releases/download/v0.7.3/ctop
   -0.7.3-linux-amd64 -O /usr/local/bin/ctop
5 sudo chmod +x /usr/local/bin/ctop
6 sudo apt install -y postgresql
```

Install Docker

On the target machine

```
1 curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
2 | sudo apt-key add -
3 sudo add-apt-repository "deb [arch=amd64] \
4 https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
5 sudo apt-get update
6 apt-cache policy docker-ce
7 sudo apt-get install -y docker-ce
8 sudo systemctl start docker
9 sudo docker run hello-world
10 sudo systemctl enable docker
11 sudo systemctl status docker
```

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

```
1 nano /etc/default/docker
2 # Add these lines #(maybe not needed?)
3 export http_proxy="http://localhost:8080"
4 export https_proxy="http://localhost:8080"
5 #Create a systemd drop-in directory for the docker service:
6 sudo mkdir -p /etc/systemd/system/docker.service.d
```

```
7 nano /etc/systemd/system/docker.service.d/http-proxy.conf
8 #Add these lines
9 [Service]
10 Environment="HTTP_PROXY=http://localhost:8080/"
11 #Flush changes:
12 sudo systemctl daemon-reload
13 #Restart Docker:
14 sudo systemctl restart docker
15 #Verify that the configuration has been loaded:
16 systemctl show --property=Environment docker
17 #Environment=HTTP_PROXY=http://localhost:8080/
```

Install Docker Compose

On the target machine

```
1 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
2 sudo chmod +x /usr/local/bin/docker-compose
3 docker-compose --version
4 #docker-compose version 1.27.4, build 40524192
```

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 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

```
1 version: '3'
2
3 networks:
4   default:
```

```
5     ipam:
6         driver: default
7         config:
8             - subnet: "172.27.0.0/24"
9
10    services:
11    kwanza:
12        image: registry.sirenia.io/kwanza:v2.16.2
13        restart: unless-stopped
14        environment:
15            KWANZA_DATABASE: pg://postgres:postgres@postgres/kwanza
16            KWANZA_MINTLSVERSION: 1.2
17            KWANZA_CIPHERSUITES: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
18                TLS_RSA_WITH_AES_128_GCM_SHA256
19                TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
20                TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256"
21            KWANZA_PREFERSERVERCIPHERSUITES: "True"
22            KWANZA_STRICTTRANSPORTSECURITY: "True"
23            KWANZA_CERT_SUBJECTS: "${HOSTNAME}"
24            KWANZA_CERT_DURATION: 87600h
25            KWANZA_CERT: "/cert/cert.pem"
26            KWANZA_KEY: "/cert/key.pem"
27            KWANZA_SALT: kwanzified
28            KWANZA_AUTH: jwt
29            KWANZA_MAXSTREAMSPERSUBSCRIBER: 102400
30            KWANZA_MAXAUTHTHROTTLEDKEYS: -1
31            KWANZA_MAXTHROTTLEDKEYS: -1
32    ports:
33        - "8000:8000"      # HTTP(S)
34        - "8001:8001"      # TCP (gRPC)
35        - "127.0.0.1:6060:6060"  # Profiling to host-only
36        - "127.0.0.1:8080:8080"  # Expvar to host-only
37    volumes:
38        - "/usr/local/etc/sirenia/cert:/cert"
39        - "/usr/local/etc/sirenia/kwanza/conf:/etc/sirenia/kwanza"
40    depends_on:
41        - postgres
42
43    cuesta:
44        image: registry.sirenia.io/cuesta:v1.14.17
45        restart: unless-stopped
46        environment:
47            CUESTA_CERT: "/cert/cert.pem"
```

```
45     CUESTA_KEY: "/cert/key.pem"
46     KWANZA_URL: "https://${HOSTNAME}:8000/v1"
47     KWANZA_STREAMURL: "wss://${HOSTNAME}:8000/v1/stream"
48     ports:
49       - "80:80"
50       - "443:443"
51     volumes:
52       - "/usr/local/etc/sirenia/cert:/cert"
53     depends_on:
54       - kwanza
55
56     postgres:
57       image: postgres:10
58       restart: always
59       ports:
60       - "127.0.0.1:5432:5432"
61     environment:
62       PGDATA: "/data"
63       POSTGRES_PASSWORD: "postgres"
64     volumes:
65       - "/root/postgresdata:/data"
```

Configure Kwanza

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

paste this

```
1 users:
2   john:
3     d224cfd091471383708424f3e494f8029b456b0e559fe82ee9adb5b66a7f1e55
4   martin:
5     d224cfd091471383708424f3e494f8029b456b0e559fe82ee9adb5b66a7f1e55
6   jonathan:
7     d224cfd091471383708424f3e494f8029b456b0e559fe82ee9adb5b66a7f1e55
```

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

```
1 cd /root/deploy
2 docker login registry.sirenia.io
```

```
3 #dist-<username> / <password>
4 # ... Login Succeeded
5 docker-compose up
6 <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

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

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

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
3 networks:
4   default:
5     ipam:
6       driver: default
7       config:
8         - subnet: "172.28.0.0/24"
9
10  services:
11
12    nginx-proxy:
13      container_name: nginx-proxy
14      image: jwilder/nginx-proxy
15      ports:
16        - "81:443"
17      restart: always
18      #environment:
19      volumes:
20        - "/var/run/docker.sock:/tmp/docker.sock:ro"
21        - "./nginx-proxy/htpasswd:/etc/nginx/htpasswd"
22        - "/usr/local/etc/sirenia/cert:/etc/nginx/certs"
23
24    aripuana-stats:
25      image: registry.sirenia.io/aripuana:v1.5.1
26      restart: unless-stopped
27      environment:
28        ARIPUANA_MINTLSVERSION: 1.2
29        ARIPUANA_CIPHERSUITES: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
30          TLS_RSA_WITH_AES_128_GCM_SHA256
31          TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
32          TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256"
33        ARIPUANA_PREFERSERVERCIPHERSUITES: "True"
34        ARIPUANA_STRICTTRANSPORTSECURITY: "True"
35        ARIPUANA_CERT_SUBJECTS: "${HOSTNAME}"
36        ARIPUANA_CERT_DURATION: 87600h
37        ARIPUANA_CERT: "/cert/cert.pem"
38        ARIPUANA_KEY: "/cert/key.pem"
39        ARIPUANA_SALT: "fishy"
40        ARIPUANA_WRITERS: 1
41        ARIPUANA_PORT: 8083
```



```
39     ARIPUANA_LOGNAME: "stats.manatee"
40     ARIPUANA_OUTPUTDIR: "/data"
41     ports:
42     - "8082:8082"
43     - "8083:8083"
44     volumes:
45     - "/usr/local/etc/sirenia/cert:/cert"
46     - "./aripuana/data:/data"
47
48     aripuana-logs:
49     image: registry.sirenia.io/aripuana:v1.5.1
50     restart: unless-stopped
51     environment:
52     ARIPUANA_MINTLSVERSION: 1.2
53     ARIPUANA_CIPHERSUITES: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
54     TLS_RSA_WITH_AES_128_GCM_SHA256
55     TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
56     TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256"
57     ARIPUANA_PREFERSERVERCIPHERSUITES: "True"
58     ARIPUANA_STRICTTRANSPORTSECURITY: "True"
59     ARIPUANA_CERT_SUBJECTS: "${HOSTNAME}"
60     ARIPUANA_CERT_DURATION: 87600h
61     ARIPUANA_CERT: "/cert/cert.pem"
62     ARIPUANA_KEY: "/cert/key.pem"
63     ARIPUANA_SALT: "fishy"
64     ARIPUANA_WRITERS: 1
65     ARIPUANA_PORT: 8085
66     ARIPUANA_LOGNAME: "all.manatee"
67     ARIPUANA_OUTPUTDIR: "/data"
68     ports:
69     - "8084:8084"
70     - "8085:8085"
71     volumes:
72     - "/usr/local/etc/sirenia/cert:/cert"
73     - "./aripuana/data:/data"
74
75     aripuana-perf:
76     image: registry.sirenia.io/aripuana:v1.5.1
77     restart: unless-stopped
78     environment:
79     ARIPUANA_MINTLSVERSION: 1.2
80     ARIPUANA_CIPHERSUITES: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
81     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-
126 #       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
```

```
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 file {
19     #perf for perf only
20     type => "perf-manatee"
21     path => "/etc/logstash/indata/perf.manatee*.log"
22     #start_position => "beginning"
23     start_position => "end"
24     codec => json
25 }
26 }
27 filter {
28     #NOOP
29 }
30 output {
31     if [type] == "all-manatee" {
32         elasticsearch {
33             hosts => ["localhost"]
34             manage_template => false
35             index => "all-manatee-1"
36         }
37     }
38     if [type] == "bi-manatee" {
39         elasticsearch {
40             hosts => ["localhost"]
41             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

```
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
2  -----                                -
```

```
3 aripuana-logs    aripuana run                Up
   0.0.0.0:8084->8084/tcp, 0.0.0.0:8085->8085/tcp
4 aripuana-perf    aripuana run                Up
   0.0.0.0:8086->8086/tcp, 0.0.0.0:8087->8087/tcp
5 aripuana-stats   aripuana run                Up
   0.0.0.0:8082->8082/tcp, 0.0.0.0:8083->8083/tcp
6 elk6             /usr/local/bin/start.sh     Up      5044/tcp,
   5601/tcp, 9200/tcp, 9300/tcp
7 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 server restart, to test that all services starts after a complete host restart

```
1 reboot -n
```