horizontal line

Versa Manual Deployment

**24th Sep 2019**

|  |  |  |
| --- | --- | --- |
| **Version #** | **Date** | **By** |
| 1.0 | 25/07/2018 | Prabhakaran Palanisamy |
| 2.0 | 10/2/2018 | PP |
| 3.0 | 10/30/2018 | Sathish Kumar |
| 4.0 | 02/21/2019 | Sathish Kumar |
| 4.0 | 05/09/2019 | Sathish Kumar |
| 5.0 | 25/09/2019 | Praveen Srinivasan |

# Table of Contents

[**Table of Contents**](#_7noxn4wpv78a) **2**

[**Prepare node**](#_11gesve87xp5) **8**

[Download Package](#_ib0foeq76g3c) 8

[QCow2 Image Contains](#_e5cxa067357l) 8

[Initial VM Setup](#_8t7fwurp8rl4) 8

[Change Hostname](#_gntspjscmtzy) 8

[Rsyslog Setup](#_xesir4fur18o) 9

[**Prepare Docker Images**](#_q7hesvnxgxc5) **9**

[Setup Nexus Authorization](#_qj8ozz5iyv9q) 9

[**Rsync Setup (Only for Shop - Multi Node)**](#_r14aju39d83s) **10**

[Dependencies](#_ibuisa5i6nbi) 10

[Increase Inotify Watch limit](#_awt4xb6rfxr) 10

[Create SSH key](#_2s8eyo1) 10

[Share SSH Key](#_17dp8vu) 11

[Append Authorized Key](#_jj5jdw8y6ptj) 11

[Verify SSH connection (make an entry in know host)](#_3rdcrjn) 11

[Rsync Shell Script](#_vqct8if2trz) 12

[For Media Folder](#_vopsmrcc1cd5) 12

[For Reseller Folder (Only before shop 1.3R6)](#_2ggi9tx33h6c) 13

[For Static Folder (Only before shop 1.3R6)](#_1ksv4uv) 13

[For WSDL Folder (Only before shop 1.3R6)](#_ianewu3sb7xk) 14

[Give Execute Permission (from Shop 1.3R6)](#_c9d9spegee8l) 14

[Give Execute Permission (before Shop 1.3R6)](#_xdpzoxw418x) 15

[Create Supervisord Config (from Shop 1.3R6)](#_pb8i5i6gkzsf) 15

[Create Supervisord Config (before Shop 1.3R6)](#_6n1meaq0csd0) 15

[Restart Supervisor](#_i63l5jfs3ahz) 16

[**Initial Media and Reseller Files Setup (Only for Shop)**](#_tlyx1ywey10x) **18**

[**Single Node Deployment**](#_g9ka0h7fzy55) **19**

[Init Swarm](#_clsnknknjbv2) 20

[Create Network](#_pex79lxwg4zo) 20

[Setup Gluu](#_k5gqexg5oivc) 20

[Deploy Consul, Registrator Services](#_rmw4n87wefif) 20

[Configure Gluu](#_mw7hja3iqk72) 20

[Deploy Ldap Service](#_ml2vgnm0ukn8) 21

[Deploy Nginx, Oxauth, Oxtrust, Oxpassport, Oxshibboleth Services](#_p3i7va55rh6e) 21

[Run LDAP Backup Script (Strictly run backup script only once)](#_6zhv15l14pao) 22

[Deploy Single Node Stack](#_6udocvkdcpvs) 22

[Copy MySQL Dump to Docker Mount Volume](#_jgk6mysr35mi) 22

[Modify Configs](#_ir4ihlemomdl) 22

[Modify Device Facing API Config](#_eektjejoulna) 23

[Configure logstash](#_gd7soi7a7lwb) 25

[Modify Stack File](#_r0g9rohveu32) 27

[Deploy Infra Stack (Only Saas)](#_ak8kucte5p02) 27

[Deploy Infra Stack (Saas + Shop)](#_p02oo661dtj7) 28

[Configure MongoDB](#_kvzqy14c557f) 28

[Configure Auth](#_ix2fpxh8zs2o) 28

[Copy Mongodb Dump to Mongo](#_tmyo0tzh6j4u) 28

[Initial Mongodb Dump](#_71gvnvhp1fnx) 29

[Configure Kafka](#_2fva0dofd5x9) 29

[Create Topics](#_e0ez1i6yglye) 29

[Setup MySQL (Only for Saas + Shop)](#_191akwm72l7o) 30

[Configure Setup.sh](#_if0m8qnawf10) 30

[Copy MySQL Setup to MySQL Galera Container](#_wbjh5sc8yjo3) 30

[Initial MySQL Dump](#_ocwhq08sqiui) 31

[Deploy Web (Only Saas)](#_lkn3imr54v4s) 31

[Deploy Web (Saas + Shop)](#_x1byo9ciejd4) 31

[**Multi-node Swarm Deployment**](#_ysmiuwepsorl) **32**

[Pre-requisites](#_7i0z8ot5iyvz) 33

[Create Cluster](#_wgi523xknin7) 33

[Init swarm (In manager0)](#_i7qm2xhvza17) 33

[Get Manager Join Token](#_htcr5uklxr6) 33

[Add manager1 and manager2 to the Swarm Cluster](#_bupd8lisj1m5) 33

[List the Attached Swarm Nodes](#_czgx4825wxfv) 33

[Create Node Label (Deploy Role)](#_mgqv588lxxw) 35

[Create Node Label (Zone)](#_v6oqm6wy4x1y) 35

[Create Network (Saas Only)](#_ydyqfw54ziof) 35

[Create Network (Saas+Shop)](#_9dnnbakd18hz) 35

[Copy MySQL Dump to Docker Mount Volume](#_ltmyxgqn671l) 35

[Configure logstash](#_q9pf0v2i0xkq) 36

[Deploy Infra Stack (Only Saas)](#_yj9xp1t4x0q8) 38

[Deploy Infra Stack (Saas + Shop)](#_ma3hpx83eth3) 38

[Setup Gluu](#_fc4d6erxjta6) 38

[Setup CSync](#_c4892ayeu6du) 38

[Deploy](#_jq8nuus9waky) 39

[Deploy Consul Services](#_5n9zu47n9qhb) 39

[Configure Gluu](#_ftwfqfd9zmxi) 39

[Deploy Ldap Manager Service](#_y8j07pmw7yrb) 40

[Deploy worker-1 (Replication of LDAP manager)](#_ung6gd3hn7ji) 41

[Deploy worker-2 (Replication of LDAP manager)](#_ycwzeqimmm0k) 41

[Deploy Registrator](#_j8qmidrolvvy) 41

[Deploy Nginx (Saas)](#_h6hr25u2ib11) 41

[Deploy Nginx (Saas+Shop)](#_cxub2a48mv89) 41

[Deploy Nginx, Oxauth, Oxtrust, Oxpassport, Oxshibboleth Services](#_75tp7z6lx72p) 42

[Run LDAP Backup Script (Strictly run backup script only once)](#_ox239fbochrw) 42

[Deploy and Configure MongoDB Replica Set (No Sharding)](#_sswra3n3tyb5) 42

[We are going to build our replica set](#_2bkm07km3tzl) 42

[Configure Auth](#_kdn7qk8l9b2o) 43

[Copy Mongodb Dump to Mongo Replica Set](#_t1j9je7fphp6) 43

[Initial Mongodb Dump](#_ed85b3zfhm32) 43

[Deploy and Configure Kafka](#_cscu167eplj) 44

[Modify Stack](#_l04zuug5wevt) 44

[Create Topics](#_8y30lr5rioav) 44

[Setup MySQL (Only for Saas + Shop)](#_aztlw7fhf6jq) 45

[Modify Stack](#_wvh1gyk3sipd) 45

[Configure Setup.sh](#_iclmpjgxlolz) 45

[Copy MySQL Setup to MySQL Galera Container](#_83ts8u3hsc9c) 45

[Initial MySQL Dump](#_wk8lel870n2) 46

[Deploy Application Stacks](#_rmvu0a38v3bd) 46

[Modify Configs](#_y6644g7buokm) 46

[Modify Stack File](#_fd37abr72hrf) 49

[Deploy (Only Saas)](#_slzps7wmolz6) 49

[Deploy (Saas + Shop)](#_4tuxo43lajm2) 49

[**HAProxy Setup (External Endpoint LB)**](#_fznnivxb9r7c) **49**

[Install new Haproxy server (VM)](#_et65ds1j6ni7) 49

[Prepare Certificate](#_944ogi32u8nz) 49

[Copy the certificate to external endpoint server](#_xxk8hp92kcg9) 50

[Rename and Copy certificate to haproxy](#_6zie4t84jzrr) 50

[Configure Haproxy](#_bzxls4xudimy) 50

[Restart Haproxy](#_xr2lopq2q5j) 52

[**Troubleshooting**](#_fp3ng9118zik) **52**

[Change Configuration and Redeploy](#_f5knochi0gwf) 52

[Remove unwanted docker data](#_1c2ppg6w83mo) 53

[Remove or Re-deploy Stack](#_t6itcwdcu9j) 53

[Rollback Docker Service](#_i4n3rqme4yud) 54

[Kafka: Produce and Consume](#_dr1bk9hyzyb1) 54

[Test Saas API -> Device facing api -> VD](#_7eaoovh5iudy) 55

[Essential Docker Commands](#_r7nnl7rdff77) 55

[To Remove MySQL Permission](#_4kg1xl6shmbz) 56

[**Appendix - A**](#_phddm8tjv5sc) **57**

[Node/Host Spec (Minimum)](#_s1ma7ynp2bhx) 57

[VM/Physical Host:](#_laapi3cvpcu6) 57

[HAProxy VM/Physical Host](#_tfqrvloizbck) 57

[Ports to open over versa-control interface](#_5snnlkpiloiz) 57

[Other Ports to open](#_brjpindqd6dz) 57

[Saas Dashboard/React ENV](#_mayzv7ok0gbu) 58

[Node Socket ENV](#_y3an0y3kfknl) 59

[Stream Processor ENV](#_h4xky6cldw1h) 60

[Saas API ENV](#_afr5s61valxp) 62

[Init Mongo Dump](#_8edhs1we50xb) 71

[**Appendix B: Docker Stack File**](#_m9s4yd2l0dza) **71**

[Version](#_p8n0sj5lnw0b) 71

[Networks](#_bc1gmi4gul91) 72

[Creating the Overlay Network during stack deployment,](#_fo0j1sf12s1g) 72

[Driver: Overlay](#_yzlkza102mir) 72

[Referencing External](#_25agnroatbmi) 73

[Adding Service to Overlay network](#_kl2o90v58ijz) 73

[Volumes](#_a40wa12t0blx) 73

[Attaching/Binding Volumes to Service](#_m0hci0liwqpp) 74

[Secrets](#_8j9w07ku33k5) 75

[Attaching Secrets to a Service](#_x18exxhg2oxk) 75

[Configs](#_n2g3svjrrn3i) 76

[Attaching Config to a Service](#_bigmm02kwrfa) 76

[Image](#_szqhr11eren7) 77

[Examples](#_1tci87um1hjd) 77

[Ports](#_2wujy9b3xd5w) 77

[Logging](#_3on5sw5e5bcb) 78

[Command](#_nk9fghqf42v) 79

[HAProxy](#_wja0o6pmul4x) 80

[MySQL](#_kzgfk6995k43) 80

[MongoDB](#_wc4sr3w87up) 80

[Data (mongodb\_cluster.yml)](#_h61atd70j81w) 81

[Config (mongodb\_cluster.yml)](#_529bolfb9mf6) 81

[Mongos Router (mongodb\_cluster.yml)](#_bls7eho7nx63) 81

[Replica Set Mongo (mongodb\_replicas.yml)](#_oxwm5mkrydwj) 82

[Deploy](#_goii8zjzrjjj) 82

[Global Mode](#_gwh76oqijj0s) 82

[Replicated Mode](#_30farwjipqb) 83

[Mode](#_7z13o7ky5mgj) 83

[Replicas](#_cc3zzkapbq5o) 83

[Placement: Constraints](#_sndpfukwhep4) 84

[Environment Variables](#_pldz7veycw6g) 85

[Zookeeper](#_k23haluy0bk2) 85

[Kafka Broker](#_alr1nz4nfkh0) 85

[KAFKA\_BROKER\_ID](#_se42iql5xm6a) 86

[KAFKA\_AUTO\_CREATE\_TOPICS\_ENABLE](#_qjxsr68xyx5k) 86

[KAFKA\_ADVERTISED\_LISTENERS](#_msye23k5h19r) 86

[KAFKA\_LISTENERS](#_woqetzwt8yrg) 86

[KAFKA\_INTER\_BROKER\_LISTENER\_NAME](#_q0x8zmf6tpr7) 87

[KAFKA\_ZOOKEEPER\_CONNECT](#_xp10wilb15vx) 87

[KAFKA\_ZOOKEEPER\_CONNECTION\_TIMEOUT\_MS](#_n4wwphnbxplo) 87

[KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP](#_35u4zott7yhw) 87

[KAFKA\_ADVERTISED\_PORT](#_kb0eodu18g2) 87

[KAFKA\_PORT](#_3vtscbx4k192) 87

[KAFKA\_JMX\_OPTS](#_vuic42bv7ew5) 88

[Stream Processor](#_eugs93ghorri) 88

[Values](#_f1uhb23imbmd) 88

[MySQL](#_yvg9scfuy94q) 88

[Galera First Primary Service](#_9mkw6rkm2hdq) 88

[Galera Other Services](#_j2q95z5w4wz9) 88

[MYSQL\_DATABASE](#_vok8cokt228x) 88

[MYSQL\_ROOT\_PASSWORD](#_c3p78gqzzrj) 88

[MYSQL\_USER and MYSQL\_PASSWORD](#_okc836of3sbs) 89

[MYSQL\_INIT\_CLUSTER\_ADDRESS](#_z1g7d65mo8s2) 89

[**Appendix C: Frequently Asked Questions**](#_amykp47ea9gf) **90**

[Backup and Restore MongoDB (Logical)](#_m8jbhq8gubn0) 92

[Backup and Restore MongoDB (Physical)](#_7e0lzkn14308) 94

[Backup and Restore Kafka-Zookeeper](#_3gxvt7e) 95

[Restore Backup](#_1w363f7) 96

[Re-create Kafka Topics](#_4g2tm30) 96

[**Appendix D: Case Study**](#_fukxktmo46og) **97**

[Fault Tolerance: 3 Node Cluster - 1 Node Failure](#_sartexo9ezs0) 97

[Gluu Cluster](#_bbc4urqe226g) 97

[Saas+Shop+Gluu Cluster](#_dlfxqr5u37pl) 97

[Fault Tolerance: 3 Node Cluster - 2 Node Failure](#_v40zn6g1it7s) 97

[Gluu Cluster](#_pwl7zegvfgh2) 97

[Saas+Shop+Gluu Cluster](#_q7cin7idohq) 98

[**Appendix E: DNS and Client Affinity**](#_8ht1vwd3xaqn) **98**

# Prepare node

Do this for the node(s) used for your swarm cluster (single or multiple node). See [node spec](#_s1ma7ynp2bhx)

## Download Package

The package can be downloaded from the box, the link would be shared via Email.

|  |
| --- |
| **$** wget https://<download\_link>/versa-deployment-stacks.tar.gz **$** mkdir -p ~/versa-deployment-stack **$** tar -xvf versa-deployment-stacks.tar.gz -C ~/versa-deployment-stack |

## QCow2 Image Contains

1. Deployment stacks and configs
2. Docker images (Latest)
3. Rsyslog config
4. Logrotate
5. Auto Script
6. Init.sh

## Initial VM Setup

|  |
| --- |
| **$ cd ~/ $ chmod +x init.sh**  **$ sed -i 's/\r//' init.sh**  # Installs the prerequisites for a Ubuntu 16.04 LTS Node  **$** sudo ./init.sh  # Move the deployment stacks to user “versa” home directory  $ mv ~/versa-deployment-stack/\* /home/versa/ |

## Change Hostname

To change the hostname run the below command in the respective node (Single or Multi-node)

|  |
| --- |
| **$** sudo hostname manager<0 or 1 or 2>  # Add/update hostname  **$** sudo nano /etc/hostname  # Add/update hostname to point 127.0.0.1 **$** sudo nano /etc/hosts |

## Rsyslog Setup

Do it in all respective nodes (for both Single node and Multi-node)

|  |
| --- |
| **$** cd ~/versa-deployment-stack **$** chmod +x setup-log.sh && sed -i 's/\r//' setup-log.sh  # This will copy versa.conf file to rsyslog,  # Restarts Rsyslog,  # And copy logrotate config “versalog”. **$** sudo ./setup-log.sh |

Check if the ports 514/udp, 515/udp, 614/udp are open (for rsyslogd) to receive logs,

|  |
| --- |
| **$** sudo netstat -alnp | grep 514  **$** sudo netstat -alnp | grep 515  **$** sudo netstat -alnp | grep 614 |

# 

# Prepare Docker Images

## Setup Nexus Authorization

*Note: Will reference this as Build Server*

Create or modify file /etc/docker/daemon.json

|  |
| --- |
| {  "insecure-registries": [  "saasrepo.versa-networks.com:8443"  ],  "disable-legacy-registry": true  } |

Restart Docker

|  |
| --- |
| **$** sudo service docker restart |

Login to nexus repository, so docker image pull and push will be seamless,

|  |
| --- |
| **$** sudo docker login -u versa -pVer9a2hj saasrepo.versa-networks.com:8443 |

Check the setup

|  |
| --- |
| **$** sudo docker pull saasrepo.versa-networks.com:8443/versa\_haproxy:latest |

# Rsync Setup (Only for Shop - Multi Node)

This setup is not required for Saas Deployment and for Single node Saas+Shop deployment

Do it in all respective nodes (Multi-node Saas+Shop Deployment).

## Dependencies

1. openssh-server
2. rsync
3. Inotify-tools
4. Supervisor

## Increase Inotify Watch limit

We need to Increase inotify watch limit from 8192 to 32768 in all nodes. Enter the following commands in all nodes.

|  |
| --- |
| **$** echo fs.inotify.max\_user\_watches=32768 | sudo tee -a /etc/sysctl.conf $ sudo sysctl -p |

## Create SSH key

Create the ssh key for root user (in Manager0)

|  |
| --- |
| $ ssh-keygen -f rsync Generating public/private rsa key pair. Enter file in which to save the key (/home/server1/.ssh/rsync):  Created directory '/home/server1/.ssh'. Enter passphrase (empty for no passphrase):  Enter same passphrase again:  Your identification has been saved in /home/server1/.ssh/rsync. Your public key has been saved in /home/server1/.ssh/rsync.pub. The key fingerprint is: SHA256:WkR22BCeP8qs2cFcuo990Bc/Bw6xK6rJIYT8peCTxL4 server1@server1 The key's randomart image is: +---[RSA 2048]----+ | ==. | | +.o. . | | + o | | o . . . o o | | \* . . S +. + + | | + = o B +o.o o.o| | = + o O. o . o| | o o \*.= . | | E \*.+.o. | +----[SHA256]-----+ |

## Share SSH Key

Scp or copy rsync file to other nodes (Manager1 and Manager2) in swarm cluster in path **/root/.ssh/rsync**.

Scp or copy rsync.pub file to other nodes (Manager1 and Manager2) in swarm cluster in path **/root/.ssh/rsync.pub**.

## Append Authorized Key

Append content of **rsync.pub** file into **authorized\_keys** (/root/.ssh/authorized\_keys), of all nodes in the swarm cluster.

|  |
| --- |
| **$** sudo su **$** cat /root/.ssh/rsync.pub > /root/.ssh/authorized\_keys |

This should be done in all Swarm Nodes (**Including manager0**). Now all nodes are using the same public key.

## Verify SSH connection (make an entry in know host)

Check whether ssh connection is working from manager0 to manager1 and manager2 servers.

|  |
| --- |
| $ ssh -i ~/.ssh/rsync root@<Manager1 IP> Are you sure you want to **continue** connecting (yes/no)? yes $ ssh -i ~/.ssh/rsync root@<Manager2 IP> Are you sure your want to **continue** connecting (yes/no)? yes |

Similarly check from other servers too

From manager1 Server to manager0 and manager2

|  |
| --- |
| $ ssh -i ~/.ssh/rsync root@<Manager0 IP> Are you sure you want to **continue** connecting (yes/no)? yes $ ssh -i ~/.ssh/rsync root@<Manager3 IP> Are you sure you want to **continue** connecting (yes/no)? yes |

From manager1 Server to manager0 and manager1

|  |
| --- |
| $ssh -i ~/.ssh/rsync root@<Manager0 IP> Are you sure you want to **continue** connecting (yes/no)? yes $ssh -i ~/.ssh/rsync root@<Manager1 IP> Are you sure you want to **continue** connecting (yes/no)? yes |

## Rsync Shell Script

Repeat the below steps in all nodes.

Sync table,

|  |  |
| --- | --- |
| **Nodes (Media, Resellers, Static)** | **Sync To** |
| Manager0 | Manager1 and Manager2 |
| Manager1 | Manager0 and Manager2 |
| Manager2 | Manager0 and Manager1 |

The example given below, to create a shell script is Sync from ***Manager0 to Manager1 and Manager2***. Likewise create shell script in ***Manager1 to sync with Manager0 and Manager2***, in ***Manager2 to Manager0 and Manager1***, change the shell script accordingly with the corresponding Manager IP.

### For Media Folder

Create the file media\_rsync.sh for manager0 in /opt/versa folder.

|  |
| --- |
| **$** sudo mkdir /opt/versa  **$** sudo nano /opt/versa/media\_rsync.sh |

|  |
| --- |
| *#!/bin/bash* **while** true; **do** **if** [ -d $1 ]; **then**  inotifywait -r -e modify,attrib,close\_write,move,create,delete /var/lib/docker/volumes/magento-media/  rsync -avz -e "ssh -i /root/.ssh/rsync -o StrictHostKeyChecking=no" /var/lib/docker/volumes/magento-media/ root@**<Manager1\_IP>**:/var/lib/docker/volumes/magento-media/ --delete  && rsync -avz -e "ssh -i /root/.ssh/rsync -o StrictHostKeyChecking=no"  /var/lib/docker/volumes/magento-media/ root@**<Manager2\_IP>**:/var/lib/docker/volumes/magento-media/ --delete fi done |

### 

### For Reseller Folder (*Only before shop 1.3R6*)

Create the file reseller\_rsync.sh for Manager0 server in /opt/versa folder

|  |
| --- |
| **$** sudo nano /opt/versa/reseller\_rsync.sh |

|  |
| --- |
| *#!/bin/bash*  while true; do if [ -d $1 ]; then  inotifywait -r -e modify,attrib,close\_write,move,create,delete /var/lib/docker/volumes/magento-reseller/  rsync -avz -e "ssh -i /root/.ssh/rsync -o StrictHostKeyChecking=no" /var/lib/docker/volumes/magento-reseller/ root@**<Manager1\_IP>**:/var/lib/docker/volumes/magento-reseller/ --delete  && rsync -avz -e "ssh -i /root/.ssh/rsync -o StrictHostKeyChecking=no" /var/lib/docker/volumes/magento-reseller/ root@**<Manager2\_IP>**:/var/lib/docker/volumes/magento-reseller/ --delete fi done |

### For Static Folder (*Only before shop 1.3R6*)

Create the file static\_cache\_rsync.sh for Manager0 server in /opt/versa folder

|  |
| --- |
| **$** sudo nano /opt/versa/static\_cache\_rsync.sh |

|  |
| --- |
| *#!/bin/bash*  **while** true; **do** **if** [ -d $1 ]; **then**  inotifywait -r -e modify,attrib,close\_write,move,create,delete /var/lib/docker/volumes/magento-static/  rsync -avz -e "ssh -i /root/.ssh/rsync -o StrictHostKeyChecking=no" /var/lib/docker/volumes/magento-static/ root@**<Manager1\_IP>**:/var/lib/docker/volumes/magento-static/ --delete  && rsync -avz -e "ssh -i /root/.ssh/rsync -o StrictHostKeyChecking=no" /var/lib/docker/volumes/magento-static/ root@**<Manager2\_IP>**:/var/lib/docker/volumes/magento-static/ --delete fi done |

### 

### For WSDL Folder (*Only before shop 1.3R6*)

Create the file wsdl\_rsync.sh for Manager0 server in /opt/versa folder

|  |
| --- |
| **$** sudo nano /opt/versa/wsdl\_rsync.sh |

|  |
| --- |
| *#!/bin/bash*  while true; do if [ -d $1 ]; then  inotifywait -r -e modify,attrib,close\_write,move,create,delete /var/lib/docker/volumes/magento-wsdl/  rsync -avz -e "ssh -i /root/.ssh/rsync -o StrictHostKeyChecking=no" /var/lib/docker/volumes/magento-wsdl/ root@<MANAGER1>:/var/lib/docker/volumes/magento-wsdl/ --delete  && rsync -avz -e "ssh -i /root/.ssh/rsync -o StrictHostKeyChecking=no" /var/lib/docker/volumes/magento-wsdl/ root@<MANAGER2>:/var/lib/docker/volumes/magento-wsdl/ --delete fi done |

### Give Execute Permission (*from Shop 1.3R6*)

Give executable permission for all files

|  |
| --- |
| **$** cd /opt/versa  **$** sudo chmod +x media\_rsync.**sh** |

### 

### Give Execute Permission (*before Shop 1.3R6*)

Give executable permission for all files

|  |
| --- |
| **$** cd /opt/versa  **$** sudo chmod +x media\_rsync.**sh** reseller\_rsync.**sh** static\_cache\_rsync.**sh** wsdl\_rsync**.sh** |

### 

### Create Supervisord Config (*from Shop 1.3R6*)

Create a file rsync.conf in /etc/supervisor/conf.d path and paste the below code, in all three nodes

|  |
| --- |
| $ sudo nano /etc/supervisor/conf.d/rsync.conf |

### 

|  |
| --- |
| [program:rsync-media] command=./opt/versa/media\_rsync.sh /var/**lib**/**docker**/**volumes**/**magento**-**media** autostart=true autorestart=true startretries=3 stderr\_logfile=/var/log/rsync\_shop.err.log stdout\_logfile=/var/log/rsync\_shop.**out**.log user=root |

### 

### Create Supervisord Config (*before Shop 1.3R6*)

Create a file rsync.conf in /etc/supervisor/conf.d path and paste the below code, in all three nodes

|  |
| --- |
| $ sudo nano /etc/supervisor/conf.d/rsync.conf |

### 

|  |
| --- |
| [program:rsync-media] command=./opt/versa/media\_rsync.sh /var/**lib**/**docker**/**volumes**/**magento**-**media** autostart=true autorestart=true startretries=3 stderr\_logfile=/var/log/rsync\_shop.err.log stdout\_logfile=/var/log/rsync\_shop.**out**.log user=root  [program:rsync-reseller] command=./opt/versa/reseller\_rsync.sh /var/**lib**/**docker**/**volumes**/**magento**-**reseller** autostart=true autorestart=true startretries=3 stderr\_logfile=/var/log/rsync\_shop.err.log stdout\_logfile=/var/log/rsync\_shop.**out**.log user=root  [program:rsync-static] command=./opt/versa/static\_cache\_rsync.sh /var/**lib**/**docker**/**volumes**/**magento**-**static** autostart=true autorestart=true startretries=3 stderr\_logfile=/var/log/rsync\_shop.err.log stdout\_logfile=/var/log/rsync\_shop.**out**.log user=root  [program:rsync-wsdl] command=./opt/versa/wsdl\_rsync.sh /var/**lib**/**docker**/**volumes**/**magento**-**wsdl** autostart=true autorestart=true startretries=3 stderr\_logfile=/var/log/rsync\_shop.err.log stdout\_logfile=/var/log/rsync\_shop.**out**.log user=root |

### Restart Supervisor

|  |
| --- |
| **$** sudo service supervisor restart |

Verify the shell script are running using the following command

|  |
| --- |
| **$** ps -ef | grep bash |

Output be like,

**from Shop 1.3R6**

|  |
| --- |
| root 32695 32691 0 Apr03 ? 00:00:00 /bin/bash ./opt/versa/shop/magento\_media\_rsync.sh /var/lib/docker/volumes/magento-media |

**before Shop 1.3R6**

|  |
| --- |
| root 32694 32691 0 Apr03 ? 00:00:00 /bin/bash ./opt/versa/shop/magento\_static\_cache\_rsync.sh /var/lib/docker/volumes/magento-static root 32695 32691 0 Apr03 ? 00:00:00 /bin/bash ./opt/versa/shop/magento\_media\_rsync.sh /var/lib/docker/volumes/magento-media root 32696 32691 0 Apr03 ? 00:00:00 /bin/bash ./opt/versa/shop/magento\_reseller\_rsync.sh /var/lib/docker/volumes/magento-reseller |

# 

# Initial Media and Reseller Files Setup (Only for Shop)

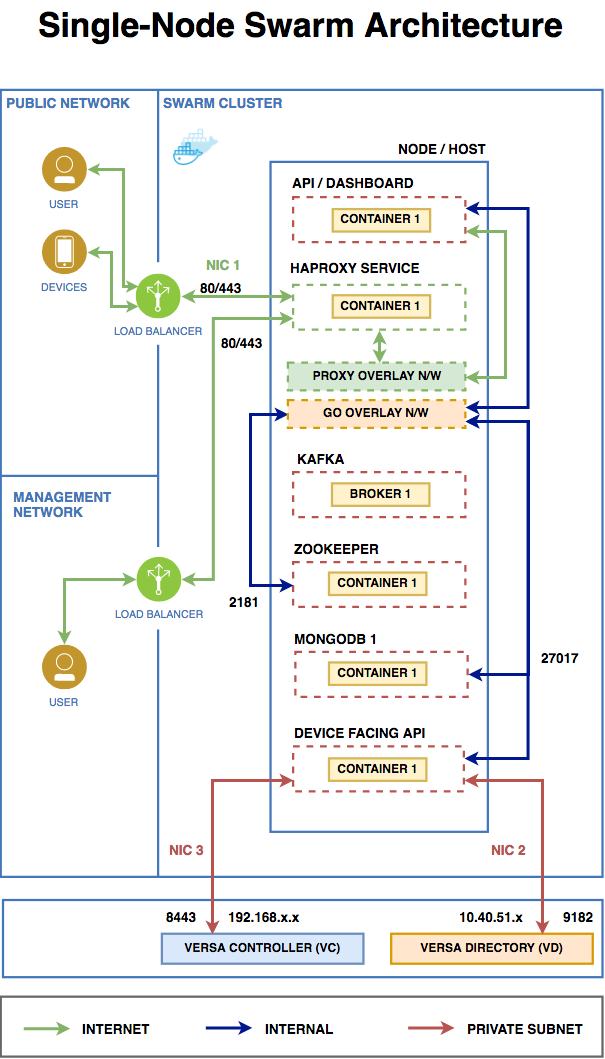
Do this step in manager0 node alone (**both single and multi node**),

For multi-node, as we have enabled Rsync between nodes, the media and reseller files would be copied automatically to other nodes.

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/init\_shop **$** chmod +x setup.sh && sed -i 's/\r//' setup.sh  # This extract and copy media, reseller files to volume **$** sudo ./setup.sh |

# 

# Single Node Deployment



## 

### Init Swarm

|  |
| --- |
| **$** sudo docker swarm init --advertise-addr host\_ip\_address |

### Create Network

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/saas\_shop/  **$** chmod +x create\_network.sh && sed -i 's/\r//' create\_network.sh  **$** sudo ./create\_network.sh |

## Setup Gluu

### Deploy Consul, Registrator Services

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/gluu  **$** sudo docker stack **deploy** -c init.yml gluu --with-registry-auth |

### Configure Gluu

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/gluu  **$** chmod +x config.sh && sed -i 's/\r//' config.sh **$** sudo ./config.sh  [I] Prepare cluster-wide configuration  [W] Unable to find configuration in Consul  [I] Creating new configuration, please input the following parameters  Enter Domain: 192.168.X.XX  Enter Country Code: IN  Enter State: TN  Enter City: Chennai  Enter Email: sathishkumar@contus.in  Enter Organization: Versa  Enter Admin/LDAP Password: versa123  **[I] The Endpoint must be without http:// or https:// and without trailing slashes (/) or path (/dashboard or /shop)**  Enter Saas Portal Endpoint [example.com]: versa.dev.contus.us  Enter Android Domain [com.versa.android]: com.versa.android  Enter IOS Domain [com.versa.saas]: com.versa.saas  Is Shop Included [Y/n]: Y  Continue with the above settings? [Y/n]:  [I] Generating configuration for the first time; this may take a moment |
|  |

If you see this after executing ./config.sh, do the following steps given below, else skip it.

|  |
| --- |
| Gluu already configured: https://No Cluster Host |

Then remove the consul service and data and [Redeploy Consul Service](#_rmw4n87wefif) and configure again, (Run the commands 2 times with the time interval of 1 minute)

|  |
| --- |
| # Remove all services and data  **$** sudo docker stack rm gluu **$** sudo rm -rf /opt/config-init/\* && sudo rm -rf /opt/consul/\* && sudo rm -rf /opt/opendj/\* && sudo rm -rf /opt/shared-shibboleth-idp/\* && sudo mkdir -p /opt/config-init/db /opt/consul /opt/opendj/config /opt/opendj/db /opt/opendj/ldif /opt/opendj/logs /opt/opendj/flag /opt/shared-shibboleth-idp |

### Deploy Ldap Service

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/gluu  **$** sudo docker stack **deploy** -c ldap.yml gluu --with-registry-auth |

Before deploying other services that need to run after LDAP is ready, we need to check if existing LDAP service has been fully ready: (Ldap service takes 5-10 minutes to get configured and indexed, wait till you get 'The Directory Server has started successfully')

|  |
| --- |
| **$** sudo tail -fn0 /var/log/gluu\_ldap.log | grep 'The Directory Server has started successfully' |

### Deploy Nginx, Oxauth, Oxtrust, Oxpassport, Oxshibboleth Services

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/gluu  **$** sudo docker stack **deploy** -c web.yml gluu --with-registry-auth |

Wait until you get 'Server:main: Started'

|  |
| --- |
| **$** sudo tail -fn0 /var/log/gluu\_oxtrust.log | grep 'Server:main: Started' |

Note: oxauth must be up before oxtrust (waits for 300 seconds). You can force update oxtrust if oxtrust fails after waiting for 5 minutes

|  |
| --- |
| **$** sudo docker service update gluu\_oxtrust --force |

### Run LDAP Backup Script (Strictly run backup script only once)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/gluu  **$** chmod +x backup-update.sh && sed -i 's/\r//' backup-update.sh  **$** sudo ./backup-update.sh |

**Reinstall Steps on Any Error**

|  |
| --- |
| # Remove old files  **$** cd /home/versa/deployment\_stack/stacks/single\_node/gluu  **$** rm -rf tmp/\* volumes/\*  # Remove all services and data  **$** sudo docker stack rm gluu **$** sudo rm -rf /opt/config-init/\* && sudo rm -rf /opt/consul/\* && sudo rm -rf /opt/opendj/\* && sudo rm -rf /opt/shared-shibboleth-idp/\* && sudo mkdir -p /opt/config-init/db /opt/consul /opt/opendj/config /opt/opendj/db /opt/opendj/ldif /opt/opendj/logs /opt/opendj/flag /opt/shared-shibboleth-idp  # Then reinstall single node |

## Deploy Single Node Stack

### Copy MySQL Dump to Docker Mount Volume

|  |
| --- |
| **$** cd /home/versa/deployment\_stack  **$** sudo tar -C /opt/versa/mysql/ -xvf init\_mysql/versa\_shop.tar.gz |

### Modify Configs

1. All application configs are found inside **configs** (/home/versa/deployment\_stack/stacks/single\_node/configs)folder.
2. Modify the all **.env files** ( saas-config.json) to match the environment.
3. Replace <APP\_ENDPOINT> with your Haproxy/DNS endpoint URL
   1. In case of single node system please use the host IP.
   2. In case of multi node system, use the Global HAProxy/DNS.
4. Replace <OAUTH\_ENDPOINT> with your Oauth endpoint URL
   1. Note: It should be the IP of your GLUU IP, if it’s our own GLUU. In case of external gluu/oauth server please use the external OAUTH server IP
5. Replace <ANALYTICS\_HOST> and <ANALYTICS\_PORT> with appropriate value in api.env

### 

### 

### Modify Device Facing API Config

1. Update VD Username, Password, Director IP
2. Update Analytics and Speed test IP
3. Update Primary and Secondary Controller information

|  |
| --- |
| CONTROLLER\_NETWORk\_NAME\_WAN1 : Internet CONTROLLER\_NETWORk\_NAME\_WAN2 : MPLS tenant\_org\_staging : Provider tenant\_org\_staging\_id : 1 provider\_org\_staging : Provider provider\_org\_staging\_id : 1 staging\_director : False Network :  WAN : [ vni-0/0, vni-0/1, vni-0/100 ]  LAN : [ vni-0/2, vni-0/3, vni-0/4, vni-0/5 ]  WIFI : [ vni-0/200, vni-0/201, vni-0/202, vni-0/203 ] Security : [ "access-policy-update", "antivirus", "ips", "urls", "enterpriseSecurity", "url-categories", "url-filtering", "url-reputations", "ip-categories", "ip-filtering", "ip-reputations" ] TrafficStreeing : True PortForwarding : True QoS : True Deploy : True Publish : True WAN\_Network\_Update : True FlowVD : True FlowAppliance: False GetDeployedDevice : True GetConfiguredService: True UpdateWorkflow : False UpdateConfigurationTemplate : False SpeedTestOnConfig : True Control\_Interface : False Service\_Template : False Cgnat : True TunnelInterface : False Guest : True zone\_update : False IP\_Reservation : True FILES:  templatedirpath : templates/ FLOW:  VD : True  Appliance : False AUTHENTICATION:  vdusername : <vd\_username>  vdpassword : <vd\_password>  vdurltimeout : 120  directorIP: <director\_ip>  apiport : 9182  applianceusername : admin  appliancepassword : versa123  tokenFlag : True primaryController:  name : <controller\_1\_name>  siteid : <controller\_1\_site\_id>  ipaddress : <controller\_1\_ip> secondaryController:  name : <controller\_2\_name>  siteid : <controller\_2\_site\_id>  ipaddress : <controller\_2\_ip> ProviderDetails :  providerorganisationid : <default\_provider\_site\_id>  providerorganisationname : <default\_provider\_name>  dnsdomainname : google.com  \_firstOct : 10 VnfManager :  ipaddress : 172.30.11.0/24 analysticsdetails:  port : <analytics\_port>  ipaddress : <analytics\_ip>  analysticsCluster : SAAS speedTest:  serverIP : <speedtest\_ip> debug : True haMode : False SingleOrg : True urlProfileName : CaptivePortal urlReputationAlertName : CaptivePortalURLRepAlert urlReputationBlockName : CaptivePortalURLRepBlock urlReputationAlertCaptiveName : CaptivePortalAlertMsg urlReputationBlockCaptiveName : CustomCaptivePortalBlockMsg urlCategoryAlertName : CaptivePortalAlert urlCategoryBlockName : CaptivePortalBlock NumberOfLans : 4 NumberOfWifi : 1 accessPolicName : SECURITY\_POLICY dialoadBalance : true lteloadBalance : false MAX\_WAN\_RTI : 12 WANNETWORKNAME : [WAN1, WAN2] LTENETWORKNAME : [ LTE ] LANNETWORKNAME : [ LAN1, LAN2, LAN3, LAN4, LAN5, LAN6, LAN7, LAN8] WLANNETWORKNAME : [ WLAN1, WLAN2, WLAN3, WLAN4, WLAN5, WLAN6, WLAN7, WLAN8] |

### Configure logstash

The AMQP configuration available in path: /home/versa/deployment\_stack/stacks/single\_node/configs/logstash/amqp.conf. **Modify Rabbitmq Config**.

|  |
| --- |
| *# This input block will listen on port 10514 for logs to come in. # host should be an IP on the Logstash server. # codec => "plain" indicates that we expect the lines we're receiving to be in JSON format # type => "rabbitmq" is an optional identifier to help identify messaging streams in the pipeline.  input {  rabbitmq {  host => "<Host\_IP>"  queue => "lci-queue"  durable =>* ***false*** *key => "#"  exchange => "LCI-Exchange"  threads => 2  prefetch\_count => 50  port => 5672  user => "lci"  password => "amqp"  type => "rabbitmq"  codec => "plain"  metadata\_enabled =>* ***true*** *} }  filter {* ***if*** *[type] == "rabbitmq" {  mutate {  add\_field => { "host" => "AMQP-POC"}  }  } }  # This output block will send all events of type "rsyslog" to Elasticsearch at the configured # host and port into daily indices of the pattern, "rsyslog-YYYY.MM.DD"  # Single Node - <KAFKA\_BROKER\_IP:PORT> kafka:9092 # Multi Node - <KAFKA\_BROKER\_IP:PORT> broker1:9092,broker2:9092,broker3:9092  output { # Use file output for Debugging  file {  path => "/tmp/versa-amqp.log"  codec => "line"  }  kafka {  topic\_id => "versa-amqp"  bootstrap\_servers => "kafka:9092"  compression\_type => "snappy"  } }* |

The VAN/Analytics configuration available in path: /home/versa/deployment\_stack/stacks/single\_node/configs/logstash/van.conf. Modify if required.

|  |
| --- |
| *# This input block will listen on port 10514 for logs to come in.* *# host should be an IP on the Logstash server.* *# codec => "json" indicates that we expect the lines we're receiving to be in JSON format* *# type => "versa-analytics" is an optional identifier to help identify messaging streams in the pipeline.*  input {  tcp {  port => 10519  codec => "json"  type => "versa-analytics"  } }  filter {  } *# This output block will send all events of type "rsyslog" to Elasticsearch at the configured* *# host and port into daily indices of the pattern, "rsyslog-YYYY.MM.DD"*  *# Single Node - <KAFKA\_BROKER\_IP:PORT> kafka:9092* *# Multi Node - <KAFKA\_BROKER\_IP:PORT> broker1:9092,broker2:9092,broker3:9092*  output {  **if** [type] == "versa-analytics" {  kafka {  bootstrap\_servers => "kafka:9092"  topic\_id => "versa-analytics"  } *# Use it only for Debugging*  file {  path => "/tmp/versa-analytics.log"  codec => "line"  }  } } |

### Modify Stack File

1. Change the docker image reference to be deployed
2. Change the replicas (if required)

### 

### Deploy Infra Stack (Only Saas)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/saas/  **$** sudo docker stack deploy --compose-file infra.yml versa --with-registry-auth |

### Deploy Infra Stack (Saas + Shop)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/saas\_shop/  **$** sudo docker stack deploy --compose-file infra.yml versa --with-registry-auth |

**List the deployed services**

|  |
| --- |
| **$** sudo docker service ls |

## Configure MongoDB

### Configure Auth

Reference: <https://docs.mongodb.com/manual/tutorial/enable-authentication/>

|  |
| --- |
| **#** List the running containers with Container ID **$** sudo docker ps  **$** sudo docker exec -it $(sudo docker ps --filter name=mongo --format '{{.Names}}') bash  root@f4ca97d48c3c:/# mongo  **mongo>** use admin;  **mongo>** db.createUser({user:"admin", pwd:"versa@87", roles:[{role:"root", db:"admin"}]})  **mongo>** db.auth("admin", "versa@87")  **mongo>** use saas;  **mongo>** db.createUser({user:"versa", pwd:"versa123", roles:[{role:"readWrite", db:"saas"}]})  **mongo>** exit  root@f4ca97d48c3c:/# exit |

### Copy Mongodb Dump to Mongo

|  |
| --- |
| **$** cd /home/versa/deployment\_stack  **$** sudo docker cp init\_mongo $(sudo docker ps --filter name=mongodb --format '{{.Names}}'):. |

Note: After ***Container\_ID\_Mongo1*** there is a colon and the respective directory where the mongodump is available. In this case we have specified dot (.) to mention current directory.

### Initial Mongodb Dump

You could see a folder **init\_mongo**,which you have checked out from repo

|  |
| --- |
| **$** sudo docker exec -it $(sudo docker ps --filter name=mongodb --format '{{.Names}}') bash  **$** mongorestore --db saas --username versa --password versa123 --authenticationDatabase saas **init\_mongo**/  root@f4ca97d48c3c:/# mongo  **mongo>** use saas;  **mongo>** db.auth("versa", "versa123")  **mongo>** show collections;  **mongo>** exit  root@f4ca97d48c3c:/# exit |

## Configure Kafka

### Create Topics

|  |
| --- |
| **$** sudo docker exec -it $(sudo docker ps --filter name=kafka --format '{{.Names}}') bash  **$** cd /opt/kafka  **$** bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 -partitions 1 --topic versa-analytics  **$** bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 -partitions 1 --topic versa-internal  **$** bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 -partitions 1 --topic versa-monitoring  **$** bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 -partitions 1 --topic versa-amqp  **$** bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 -partitions 1 --topic versa-device-backend-update  **$** bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 -partitions 1 --topic versa-update-event  **# Only for Shop CE**  **$** bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 -partitions 1 --topic versashop  # Verify - Kafka Topic List  **$** bin/kafka-topics.sh --list --zookeeper zookeeper:2181  root@f4ca97d48c3c:/# exit |

## Setup MySQL (Only for Saas + Shop)

### Configure Setup.sh

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/init\_mysql **$** nano setup.sh |

1. Replace <APP\_ENDPOINT> with your Haproxy/DNS endpoint URL
   1. In case of single node system please use the host IP.
   2. In case of multi node system, use the Global HAProxy/DNS.
2. Replace <OAUTH\_ENDPOINT> with your Oauth endpoint URL
   1. Note: It should be the IP of your GLUU IP, if it’s our own GLUU. In case of external gluu/oauth server please use the external OAUTH server IP
3. Modify other values if required.

### Copy MySQL Setup to MySQL Galera Container

|  |
| --- |
| **$** cd /home/versa/deployment\_stack  **$** sudo docker cp init\_mysql/setup.sh $(sudo docker ps --filter name=mysql --format '{{.Names}}'):. |

Note: After ***Container\_ID\_Versa\_Mysql*** there is a colon and the respective directory where the mysql dump and setup.sh is available. In this case we have specified dot (.) to mention current directory.

### Initial MySQL Dump

You could see a folder **init\_mysql**,which you have checked out from repo

|  |
| --- |
| **$** sudo docker exec -it $(sudo docker ps --filter name=mysql --format '{{.Names}}') bash  **$** chmod +x setup.sh && sed -i 's/\r//' setup.sh  **$** ./setup.sh |
|  |

### Deploy Web (Only Saas)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/saas/  **$** sudo docker stack deploy --compose-file web.yml versa --with-registry-auth |

### Deploy Web (Saas + Shop)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/single\_node/saas\_shop/  **$** sudo docker stack deploy --compose-file web.yml versa --with-registry-auth |

# 

# Multi-node Swarm Deployment

## 

## 

## Pre-requisites

1. Have at-least 3 Manager nodes
2. Hostname should be manager0, manager1, manager2

## Create Cluster

### Init swarm (In manager0)

|  |
| --- |
| **$** sudo docker swarm init --advertise-addr <Manager0\_IP> --listen-addr <Manager0\_IP> --data-path-addr <Manager0\_IP> |

### Get Manager Join Token

|  |
| --- |
| **$** sudo docker swarm join-token manager  To add a manager to this swarm, run the following command:  docker swarm join --token SWMTKN-1-4oqrobnqbxzvtwzv3ya4i5mpfxw77c48mzuq4w7zbyiw7x289d0ep-6mdggb9j6enm7yc5hk84ad xxx.xxx.xx.xx:2377 |
|  |

### 

### Add manager1 and manager2 to the Swarm Cluster

Run the command (Manager Join Token) in the nodes manager1 and manager2

|  |
| --- |
| **$** sudo docker swarm join --token SWMTKN-1-4oqrobnqbxzvtwzv3ya4i5mpfxw77c48mzuq4w7zbyiw7x289d0ep-6mdggb9j6enm7yc5hk84ad xxx.xxx.xx.xx:2377 |

### List the Attached Swarm Nodes

|  |
| --- |
| **$** sudo docker node ls  ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS pm5j1qvz1hiqd \* manager0 Ready Active Leader djceegsp7cngqd manager1 Ready Active Reachable lkh52nuh0w0vrr manager2 Ready Active Reachable |

### 

### Create Node Label (Deploy Role)

|  |
| --- |
| $ sudo docker node **update** --label-add deploy.role=manager0 manager0 $ sudo docker node **update** --label-add deploy.role=manager1 manager1 $ sudo docker node **update** --label-add deploy.role=manager2 manager2 |

### Create Node Label (Zone)

|  |
| --- |
| $ sudo docker node **update** --label-add zone=server1 manager0 $ sudo docker node **update** --label-add zone=server2 manager1 $ sudo docker node **update** --label-add zone=server3 manager2 |

### 

### Create Network (Saas Only)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node/saas/  **$** chmod +x create\_network.sh && sed -i -e 's/\r$//' create\_network.sh && sudo ./create\_network.sh |

### 

### Create Network (Saas+Shop)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node/saas\_shop/  **$** chmod +x create\_network.sh && sed -i -e 's/\r$//' create\_network.sh && sudo ./create\_network.sh |

### Copy MySQL Dump to Docker Mount Volume

Important: Do this step in all 3 Nodes (all nodes in the cluster where you have mysql containers)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack  **$** sudo tar -C /opt/versa/mysql/ -xvf init\_mysql/versa\_shop.tar.gz |

### 

### Configure logstash

The AMQP configuration available in path: /home/versa/deployment\_stack/stacks/multi\_node/configs/logstash/amqp.conf. Modify if required.

|  |
| --- |
| *# This input block will listen on port 10514 for logs to come in. # host should be an IP on the Logstash server. # codec => "plain" indicates that we expect the lines we're receiving to be in JSON format # type => "rabbitmq" is an optional identifier to help identify messaging streams in the pipeline.  input {  rabbitmq {  host => "<Host\_IP>"  queue => "lci-queue"  durable =>* ***false*** *key => "#"  exchange => "LCI-Exchange"  threads => 2  prefetch\_count => 50  port => 5672  user => "lci"  password => "amqp"  type => "rabbitmq"  codec => "plain"  metadata\_enabled =>* ***true*** *} }  filter {* ***if*** *[type] == "rabbitmq" {  mutate {  add\_field => { "host" => "AMQP-POC"}  }  } }  # This output block will send all events of type "rsyslog" to Elasticsearch at the configured # host and port into daily indices of the pattern, "rsyslog-YYYY.MM.DD"  # Single Node - <KAFKA\_BROKER\_IP:PORT> kafka:9092 # Multi Node - <KAFKA\_BROKER\_IP:PORT> broker1:9092,broker2:9092,broker3:9092  output { # Use file output for Debugging  file {  path => "/tmp/versa-amqp.log"  codec => "line"  }  kafka {  topic\_id => "versa-amqp"  bootstrap\_servers => "broker1:9092,broker2:9092,broker3:9092"  compression\_type => "snappy"  } }* |

The VAN/Analytics configuration available in path: /home/versa/deployment\_stack/stacks/multi\_node/configs/logstash/van.conf. Modify if required.

|  |
| --- |
| *# This input block will listen on port 10514 for logs to come in.* *# host should be an IP on the Logstash server.* *# codec => "json" indicates that we expect the lines we're receiving to be in JSON format* *# type => "versa-analytics" is an optional identifier to help identify messaging streams in the pipeline.*  input {  tcp {  port => 10514  codec => "json"  type => "versa-analytics"  } }  filter {  }  *# This output block will send all events of type "rsyslog" to Elasticsearch at the configured* *# host and port into daily indices of the pattern, "rsyslog-YYYY.MM.DD"*  *# Single Node - <KAFKA\_BROKER\_IP:PORT> kafka:9092* *# Multi Node - <KAFKA\_BROKER\_IP:PORT> broker1:9092,broker2:9092,broker3:9092*  output {  **if** [type] == "versa-analytics" {  kafka {  bootstrap\_servers => "broker1:9092,broker2:9092,broker3:9092"  topic\_id => "versa-analytics"  } *# Use it only for Debugging*  file {  path => "/tmp/versa-analytics.log"  codec => "line"  }  } } |

### Deploy Infra Stack (Only Saas)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node/saas/  **$** sudo docker stack deploy --compose-file infra.yml versa --with-registry-auth |

### Deploy Infra Stack (Saas + Shop)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node/saas\_shop/  **$** sudo docker stack deploy --compose-file infra.yml versa --with-registry-auth |

**List the deployed services**

|  |
| --- |
| **$** sudo docker service ls |

## Setup Gluu

### Setup CSync

oxTrust and oxShibboleth rely on a mounted volume to share oxShibboleth configuration files. Given there are 3 nodes that need to share the same copy of oxShibboleth files, csync2 is used.

Run init-node-setup.sh on manager0

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node/gluu **$** chmod +x init-**node-setup**.sh **$** sudo ./init-**node-setup**.sh |

Move created crt and key to other nodes

|  |
| --- |
| $ cd /home/versa/deployment\_stack/stacks/multi\_node/gluu  $ sudo scp -r volumes/\* versa@<Manager1\_IP>:/home/versa/deployment\_stack/stacks/multi\_node/gluu/volumes  $ sudo scp -r volumes/\* versa@<Manager2\_IP>:/home/versa/deployment\_stack/stacks/multi\_node/gluu/volumes |

Run other-node-setup.sh on manager1 and manager2

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node/gluu **$** chmod +x other-**node-setup**.sh **$** sudo ./other-**node-setup**.sh |

## 

### Deploy

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node/gluu |

#### Deploy Consul Services

|  |
| --- |
| **$** cd ~/deployment\_stack/stacks/multi\_node/gluu/  **$** sudo docker stack deploy -c consul.yml gluu --with-registry-auth |
|  |

#### Configure Gluu

|  |
| --- |
| **$** chmod +x config.sh && sed -i -e 's/\r$//' config.sh **$** sudo ./config.sh  [I] Prepare cluster-wide configuration  [W] Unable to find configuration in Consul  [I] Creating new configuration, please input the following parameters  Enter Domain: 192.168.x.x  Enter Country Code: IN  Enter State: TN  Enter City: Chennai  Enter Email: sathishkumar@contus.in  Enter Organization: Versa  Enter Admin/LDAP Password: versa123  [I] The Endpoint must be without http:// or https:// and without trailing slashes (/) or path (/dashboard or /shop)  a [example.com]: versa.dev.contus.us  Enter Android Domain [com.versa.android]: com.versa.android  Enter IOS Domain [com.versa.saas]: com.versa.saas  Is shop included [Y/n]: Y  Continue with the above settings? [Y/n]  [I] Generating configuration for the first time; this may take a moment |

If you see this after executing ./config.sh, do the following steps given below, else skip it.

|  |
| --- |
| Gluu already configured: https://No Cluster Host |

Then remove the consul service, data (in all 3 nodes) and [Redeploy Consul Service](#_jq8nuus9waky) and configure again,

|  |
| --- |
| # Remove all services and data  **$** sudo docker stack rm gluu **$** sudo rm -rf /opt/config-init/\* && sudo rm -rf /opt/consul/\* && sudo rm -rf /opt/opendj/\* && sudo rm -rf /opt/shared-shibboleth-idp/\* && sudo mkdir -p /opt/config-init/db /opt/consul /opt/opendj/config /opt/opendj/db /opt/opendj/ldif /opt/opendj/logs /opt/opendj/flag /opt/shared-shibboleth-idp |

#### Deploy Ldap Manager Service

|  |
| --- |
| **$** sudo docker stack deploy -c ldap-manager.yml gluu --with-registry-auth |

Before deploying other services that need to run after LDAP is ready, we need to check if existing LDAP service has been fully ready: (Ldap service takes 5-10 minutes to get configured and indexed)

|  |
| --- |
| **$** sudo tail -fn0 /var/log/gluu\_ldap.log | grep 'The Directory Server has started successfully' |

#### Deploy worker-1 (Replication of LDAP manager)

|  |
| --- |
| **$** sudo docker stack deploy -c ldap-worker-1.yml gluu --with-registry-auth |

We need to check if existing LDAP service has been fully ready: (Ldap service takes 5-10 minutes to get configured, indexed and replicated). - Check this on manager1

|  |
| --- |
| **$** sudo tail -fn0 /var/log/gluu\_ldap.log | grep 'The Directory Server has started successfully' |

#### Deploy worker-2 (Replication of LDAP manager)

|  |
| --- |
| **$** sudo docker stack deploy -c ldap-worker-2.yml gluu --with-registry-auth |

We need to check if existing LDAP service has been fully ready: (Ldap service takes 5-10 minutes to get configured, indexed and replicated). - Check this on manager2

|  |
| --- |
| **$** sudo tail -fn0 /var/log/gluu\_ldap.log | grep 'The Directory Server has started successfully' |

#### Deploy Registrator

|  |
| --- |
| **$** sudo docker stack deploy -c registrator.yml gluu --with-registry-auth |

#### Deploy Nginx (Saas)

Nginx is deployed here because oxauth, oxtrust requires LDAP Cluster proxy endpoint which is served by Nginx.

**Note: This Nginx service will be updated automatically while deploying Saas application stack (web.yml)**

|  |
| --- |
| **$** sudo docker stack deploy -c nginx\_saas.yml versa --with-registry-auth |

#### 

#### Deploy Nginx (Saas+Shop)

|  |
| --- |
| **$** sudo docker stack deploy -c nginx\_saas\_shop.yml versa --with-registry-auth |

#### Deploy Nginx, Oxauth, Oxtrust, Oxpassport, Oxshibboleth Services

|  |
| --- |
| # $DOMAIN is the domain value that's entered when running `./config.sh`  #For gluu india stack **$** DOMAIN=titan.versa-networks.com sudo docker stack deploy -c web.yml gluuindia --with-registry-auth  #For gluu stack  DOMAIN=titan.versa-networks.com sudo docker stack deploy -c web.yml gluu --with-registry-auth |

Wait until you get 'Server:main: Started'

|  |
| --- |
| **$** sudo tail -fn0 /var/log/gluu\_oxtrust.log | grep 'Server:main: Started' |

Note: oxauth must be up before oxtrust (waits for 300 seconds). You can force update oxtrust if oxtrust fails after waiting for 5 minutes

|  |
| --- |
| **$** sudo docker service update gluu\_oxtrust --force |

#### Run LDAP Backup Script (Strictly run backup script only once)

|  |
| --- |
| **$** chmod +x backup-update.sh  **$** sed -i -e 's/\r$//' backup-update.sh **$** sudo ./backup-update.sh |
|  |

## Deploy and Configure MongoDB Replica Set (No Sharding)

Reference: <https://docs.mongodb.com/manual/tutorial/deploy-replica-set/>

### We are going to build our replica set

|  |
| --- |
| **$** sudo docker exec -it $(sudo docker ps --filter name=mongo --format '{{.Names}}') bash -c "echo 'rs.initiate({\_id : \"datars\", members: [{ \_id : 0, host : \"mongodb1\", \"priority\" : 1 },{ \_id : 1, host : \"mongodb2\", \"priority\" : 1 },{ \_id : 2, host : \"mongodb3\", \"priority\" : 1 }] }, { \_id : 3, host : \"mongodbarb\", \"arbiterOnly\" : true })' | mongo" |

### Configure Auth

Reference: <https://docs.mongodb.com/manual/tutorial/enable-authentication/>

|  |
| --- |
| **$** sudo docker exec -it $(sudo docker ps --filter name=mongo --format '{{.Names}}') bash  root@f4ca97d48c3c:/# mongo  **mongo>** use admin;  **mongo>** db.createUser({user:"admin", pwd:"versa@87", roles:[{role:"root", db:"admin"}]})  **mongo>** db.auth("admin", "versa@87")  **mongo>** use saas;  **mongo>** db.createUser({user:"versa", pwd:"versa123", roles:[{role:"readWrite", db:"saas"}]})  **mongo>** exit  root@f4ca97d48c3c:/# exit |

### Copy Mongodb Dump to Mongo Replica Set

|  |
| --- |
| **$** cd /home/versa/deployment\_stack  **$** sudo docker cp init\_mongo $(sudo docker ps --filter name=mongo --format '{{.Names}}'):. |

### Initial Mongodb Dump

You could see a folder **init\_mongo**,which you have checked out from repo

|  |
| --- |
| **$** sudo docker exec -it $(sudo docker ps --filter name=mongo --format '{{.Names}}') bash  **$** mongorestore --db saas --username versa --password versa123 --authenticationDatabase saas **init\_mongo**/  root@f4ca97d48c3c:/# mongo  **mongo>** use saas;  **mongo>** db.auth("versa", "versa123")  **mongo>** show collections;  **mongo>** exit  root@f4ca97d48c3c:/# exit |

## Deploy and Configure Kafka

### Modify Stack

Change the node.labels.deploy.role as manager0, manager1, manager2 for broker1, broker2 and broker3 respectively in kafka\_zookeeper.yml. Configure all the 3 Brokers accordingly.

### Create Topics

|  |
| --- |
| **$** sudo docker exec -it $(sudo docker ps --filter name=versa\_broker1 --format '{{.Names}}') sh  **$** cd /opt/kafka  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 4 -partitions 8 --topic versa-analytics  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-internal  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-monitoring  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-amqp  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-device-backend-update  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-update-event  **# Only for Shop CE**  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versashop  # Verify - Kafka Topic List  **$** bin/kafka-topics.sh --list --zookeeper zookeeper1:2181  # Exit after verifying kafka topics |
| **$** exit |

## Setup MySQL (Only for Saas + Shop)

MySQL setup is mandatory if you choose to deploy both Saas + Shop.

### Modify Stack

1. Change the node.labels.deploy.role as manager0, manager1, manager2 for galera1, galera2 and galera3 respectively in mysql\_cluster.yml. Configure all the 3 Galera accordingly.
2. Modify root user password (Environment variable: MYSQL\_ROOT\_PASSWORD), if required.

### Configure Setup.sh

|  |
| --- |
| **f$** cd /home/versa/deployment\_stack/init\_mysql  **$** chmod +x setup.sh **$** nano setup.sh |

1. Replace <APP\_ENDPOINT> with your Haproxy/DNS endpoint URL
   1. In case of single node system please use the host IP.
   2. In case of multi node system, use the Global HAProxy/DNS.
2. Replace <OAUTH\_ENDPOINT> with your Oauth endpoint URL
   1. Note: It should be the IP of your GLUU IP, if it’s our own GLUU. In case of external gluu/oauth server please use the external OAUTH server IP
3. Modify other values if required.

### Copy MySQL Setup to MySQL Galera Container

|  |
| --- |
| **$** cd /home/versa/deployment\_stack  **$** sudo docker cp init\_mysql/setup.sh $(sudo docker ps --filter name=mysql --format '{{.Names}}'):. |

Note: After ***Container\_ID\_Mysql1*** there is a **colon** and the respective directory where the mysql dump and setup.sh is available. In this case we have specified **dot** (.) to mention current directory.

### Initial MySQL Dump

You could see a folder **init\_mysql**,which you have checked out from repo

|  |
| --- |
| **$** sudo docker exec -it $(sudo docker ps --filter name=mysql --format '{{.Names}}') bash  **$** chmod +x setup.sh  **$** sed -i 's/\r//' setup.sh  **$** ./setup.sh |

## Deploy Application Stacks

### Modify Configs

1. All application configs are found inside **configs** (/home/versa/deployment\_stack/stacks/multi\_node/configs) folder.
2. Modify the all **.env files** (saas-config.json) to match the environment.
3. Replace <APP\_ENDPOINT> with your external Haproxy/DNS endpoint URL
   1. In case of single node system please use the host IP.
   2. In case of multi node system, use the Global HAProxy/DNS.
4. Replace <OAUTH\_ENDPOINT> with your Oauth endpoint URL
   1. Note: It should be the IP of your GLUU IP, if it’s our own GLUU. In case of external gluu/oauth server please use the external OAUTH server IP

**Modifications in Config/ENV**

1. Replace <APP\_ENDPOINT> with your external Haproxy/DNS endpoint URL
2. Replace <OAUTH\_ENDPOINT> with your Oauth endpoint URL
3. Replace <ANALYTICS\_HOST> and <ANALYTICS\_PORT> with appropriate value in api.env

**Modification in Device Facing API Config**

1. Update VD Username, Password, Director IP
2. Update Analytics and Speed test IP
3. Update Primary and Secondary Controller information

|  |
| --- |
| CONTROLLER\_NETWORk\_NAME\_WAN1 : Internet CONTROLLER\_NETWORk\_NAME\_WAN2 : MPLS tenant\_org\_staging : Provider tenant\_org\_staging\_id : 1 provider\_org\_staging : Provider provider\_org\_staging\_id : 1 staging\_director : False Network :  WAN : [ vni-0/0, vni-0/1, vni-0/100 ]  LAN : [ vni-0/2, vni-0/3, vni-0/4, vni-0/5 ]  WIFI : [ vni-0/200, vni-0/201, vni-0/202, vni-0/203 ] Security : [ "access-policy-update", "antivirus", "ips", "urls", "enterpriseSecurity", "url-categories", "url-filtering", "url-reputations", "ip-categories", "ip-filtering", "ip-reputations" ] TrafficStreeing : True PortForwarding : True QoS : True Deploy : True Publish : True WAN\_Network\_Update : True FlowVD : True FlowAppliance: False GetDeployedDevice : True GetConfiguredService: True UpdateWorkflow : False UpdateConfigurationTemplate : False SpeedTestOnConfig : True Control\_Interface : False Service\_Template : False Cgnat : True TunnelInterface : False Guest : True zone\_update : False IP\_Reservation : True FILES:  templatedirpath : templates/ FLOW:  VD : True  Appliance : False AUTHENTICATION:  vdusername : <vd\_username>  vdpassword : <vd\_password>  vdurltimeout : 120  directorIP: <director\_ip>  apiport : 9182  applianceusername : admin  appliancepassword : versa123  tokenFlag : True primaryController:  name : <controller\_1\_name>  siteid : <controller\_1\_site\_id>  ipaddress : <controller\_1\_ip> secondaryController:  name : <controller\_2\_name>  siteid : <controller\_2\_site\_id>  ipaddress : <controller\_2\_ip> ProviderDetails :  providerorganisationid : <default\_provider\_site\_id>  providerorganisationname : <default\_provider\_name>  dnsdomainname : google.com  \_firstOct : 10 VnfManager :  ipaddress : 172.30.11.0/24 analysticsdetails:  port : <analytics\_port>  ipaddress : <analytics\_ip>  analysticsCluster : SAAS speedTest:  serverIP : <speedtest\_ip> debug : True haMode : False SingleOrg : True urlProfileName : CaptivePortal urlReputationAlertName : CaptivePortalURLRepAlert urlReputationBlockName : CaptivePortalURLRepBlock urlReputationAlertCaptiveName : CaptivePortalAlertMsg urlReputationBlockCaptiveName : CustomCaptivePortalBlockMsg urlCategoryAlertName : CaptivePortalAlert urlCategoryBlockName : CaptivePortalBlock NumberOfLans : 4 NumberOfWifi : 1 accessPolicName : SECURITY\_POLICY dialoadBalance : true lteloadBalance : false MAX\_WAN\_RTI : 12 WANNETWORKNAME : [WAN1, WAN2] LTENETWORKNAME : [ LTE ] LANNETWORKNAME : [ LAN1, LAN2, LAN3, LAN4, LAN5, LAN6, LAN7, LAN8] WLANNETWORKNAME : [ WLAN1, WLAN2, WLAN3, WLAN4, WLAN5, WLAN6, WLAN7, WLAN8] |

### Modify Stack File

1. Change the docker image reference to be deployed ($ docker images)
2. Change the replicas (if required)

### Deploy (Only Saas)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node/saas/  **$** sudo docker stack deploy --compose-file web.yml versa --with-registry-auth |

### 

### Deploy (Saas + Shop)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node/saas\_shop/  **$** sudo docker stack deploy --compose-file web.yml versa --with-registry-auth |

**List the deployed services**

|  |
| --- |
| **$** sudo docker service ls |

# HAProxy Setup (External Endpoint LB)

To Setup HAProxy

## Install new Haproxy server (VM)

Requirements,

1. Ubuntu 16.04 LTS
2. HAproxy 1.6+
3. Liblua5.3-0\_5.3.3-1
4. Have the certificate (.pem) ready **/etc/haproxy/certs/example.com.pem**.

### 

### Prepare Certificate

Copy gluu\_https.crt, gluu\_https.key from nginx container and create gluu\_https.pem file (**From manager0**),

|  |
| --- |
| **$** sudo docker cp $(sudo docker ps --filter name=gluu\_nginx --format '{{.ID}}'):/etc/certs/gluu\_https.crt . && \ sudo docker cp $(sudo docker ps --filter name=gluu\_nginx --format '{{.ID}}'):/etc/certs/gluu\_https.key . && \ cat gluu\_https.crt gluu\_https.key > gluu\_https.pem |

### Copy the certificate to external endpoint server

Copy gluu\_https.pem to your Oauth endpoint server (Haproxy or Nginx) from manager0,

|  |
| --- |
| $ scp gluu\_https.pem versa@<External\_EndPoint\_IP>:~/ |

### Rename and Copy certificate to haproxy

|  |
| --- |
| $ sudo cp ~/gluu\_https.pem /etc/haproxy/certs/example.com.pem |

## Configure Haproxy

Path: /etc/haproxy/haproxy.cfg

Public IP: 205.255.x.x

|  |
| --- |
| **global**  tune.ssl.default-dh-param 2048  ssl-**default**-bind-ciphers TLS13-AES-256-GCM-SHA384:TLS13-AES-128-GCM-SHA256:TLS13-CHACHA20-POLY1305-SHA256:EECDH+AESGCM:EECDH+CHACHA20  ssl-**default**-bind-options no-sslv3 no-tlsv10 no-tlsv11  defaults  mode http  timeout connect 5000ms  timeout client 100000ms  timeout server 100000ms  frontend https-**in**  bind <APP\_ENDPOINT>:443 ssl crt /etc/haproxy/certs/example.com.pem  default\_backend hasserver  **option** forwardfor  reqadd X-Forwarded-Proto:\ https  reqadd X-Forwarded-Port:\ 443  backend hasserver  balance source  hash-type consistent  server s01 <MANAGER0\_IP>:443 check ssl verify none  server s02 <MANAGER1\_IP>:443 check ssl verify none  server s03 <MANAGER2\_IP>:443 check ssl verify none |

## 

## Restart Haproxy

|  |
| --- |
| **$** sudo service haproxy restart |

# 

# Troubleshooting

## Change Configuration and Redeploy

Update logstash.yml, Update the highlighted content from logstash-env to logstash-env-1 (logstash-env-X, X is variable)

|  |
| --- |
| **$** cd /home/versa/deployment\_stack/stacks/multi\_node  **$** nano configs/logstash.conf  *# After config changes, update stack file to deploy with new config* **$** nano infra.yml  *# Update logstash.yml, Update the highlighted content from* ***logstash-env*** *to* ***logstash-env-1 (logstash-env-X, X is variable)***  version: "3.5"  networks:  go:  external: true  configs:  logstash-env:  file: logstash.conf  services:  logstash:  image: nexus-versa.contus.us:8083/logstash\_kafka:6.3.1  ports:  - target: 10514  published: 10514  protocol: udp  - target: 10515  published: 10515  protocol: udp  networks:  - go  configs:  - source: logstash-env  target: /usr/share/logstash/pipeline/logstash.conf  logging:  driver: syslog  options:  syslog-address: "udp://127.0.0.1:515"  tag: "logstash/{{.ImageName}}/{{.ID}}"  syslog-facility: local2  syslog-format: rfc5424micro  deploy:  mode: replicated  replicas: 1 |

## 

## Remove unwanted docker data

Remove all unused containers, networks, images (both dangling and unreferenced), and volumes.

Warning: Do this only if all service and container are running, else you may lose important data.

Reference: <https://docs.docker.com/engine/reference/commandline/system_prune/>

|  |
| --- |
| **$** sudo docker system prune -a --volumes |

**Remove unwanted docker images**

The docker image prune command allows you to clean up unused images. By default, docker image prune only cleans up dangling images.

Warning: Do this only if all service and container are running, else you may lose important data.

Reference: <https://docs.docker.com/config/pruning/#prune-images>

|  |
| --- |
| **$** sudo docker image prune -a |

## Remove or Re-deploy Stack

|  |
| --- |
| *# List stacks available* **$** sudo docker stack ls *# Remove the deploy stack. Removes services, containers and Networks deployed* **$** sudo docker stack rm mongodb |

Remove all volumes related to mongodb manually, (do this in all nodes in swarm cluster)

|  |
| --- |
| *# Filter the mongodb volumes* $ sudo docker **volume** ls | grep "mongodb"  *# Remove volumes* $ sudo docker **volume** rm mongodb\_cfg1 $ sudo docker **volume** rm mongodb\_data1 |

Redeploy Stack

|  |
| --- |
| **#** Redeploy stack **$** sudo docker stack deploy -c mongodb\_cluster.yml mongodb |

## Rollback Docker Service

Reference: <https://docs.docker.com/engine/reference/commandline/service_rollback/>

|  |
| --- |
| **#** Check the previous setting of the service **$** sudo docker service inspect versa\_api **$** sudo docker service rollback versa\_api |

## Kafka: Produce and Consume

|  |
| --- |
| **#** List the running containers with Container ID **$** docker ps  **$** docker exec -it <Container\_ID\_Kafka> bash  **$** cd /opt/kafka  **# Single Node #** Produce in SSH terminal 1 **$** bin/kafka-console-producer.sh --broker-list <host\_ip\_address>:9094 --sync --topic versa-analytics  **#** Consume in SSH terminal 2 **$** bin/kafka-console-consumer.sh --bootstrap-server <host\_ip\_address>:9094 --topic versa-analytics --from-beginning  **# Multi Node #** Produce in SSH terminal 1 **$** bin/kafka-console-producer.sh --broker-list <host\_ip\_address\_1>:9094,<host\_ip\_address\_2>:9094,<host\_ip\_address\_3>:9094 --sync --topic versa-analytics  **#** Consume in SSH terminal 2 **$** bin/kafka-console-consumer.sh --bootstrap-server <host\_ip\_address\_1>:9094,<host\_ip\_address\_2>:9094,<host\_ip\_address\_3>:9094 --topic versa-analytics --from-beginning |

## 

## Test Saas API -> Device facing api -> VD

|  |
| --- |
| **#** List the running containers with Container ID **$** docker ps  **$** docker exec -it <Container\_ID\_Versa\_API> bash  **$** curl -X GET http://devicefacingapi:80/api/v1.0/orgs/org/ |

## Essential Docker Commands

|  |
| --- |
| **#** Docker swarm and other information, can be checked all the swarm nodes **$** docker info  **#** List the stacks, with stack name and ID, deployed in the swarm **$** docker stack ls  **#** List the services of specific stack **$** docker stack services <Stack\_Name>  **#** List the services with Service name and ID **$** docker service ls  **#** List the tasks of particular service  **$** docker service ps <Service\_Name>  **#** Filter the list of service **$** docker service ls | grep "mongo"  **#** Filter the list of container **$** docker ps | grep "kafka"  **#** Login the container (Use bash or sh) **$** docker exec -it <Container\_ID> bash  **#** List the nodes in the swarm **$** docker node ls  **#** List the config **$** docker config ls  **#** List the volume **$** docker volume ls |

## To Remove MySQL Permission

|  |
| --- |
| # List the running containers with Container ID **$** sudo docker ps --filter "name=mysql\_galera" --format "table {{.ID}}\t{{.Names}}"  **$** sudo docker exec -it <Container\_ID\_Mysql\_Galera1> bash  **$** mysql -u root -p # Prompt for root password **$** password:  # For Example **mysql>** DROP USER 'haproxy'@'10.0.%'; |

# Appendix - A

## Node/Host Spec (Minimum)

### VM/Physical Host:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CPU Unit** | **RAM** | **Disk Space** | **Processor Type** | **Preferred OS** |
| 8 | 24GB | 250GB | 64 Bit | Ubuntu 16.04 LTS |

### HAProxy VM/Physical Host

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CPU Unit** | **RAM** | **Disk Space** | **Processor Type** | **Preferred OS** |
| 2 | 16GB | 100GB | 64 Bit | Ubuntu 16.04 LTS |

### Ports to open over versa-control interface

|  |  |  |
| --- | --- | --- |
| **Port** | **Protocol** | **For** |
| 2377 | tcp | Swarm management traffic |
| 7946 | tcp + udp | Between swarm nodes |
| 4789 | udp | Overlay network traffic |
|  | ip protocol 50 (ESP) | Overlay network with encryption |

### Other Ports to open

|  |  |  |  |
| --- | --- | --- | --- |
| **Port** | **Protocol** | **Destination** | **Source** |
| 80 | tcp | Saas Master | Global Haproxy |
| 9182/9183 | tcp | Director | Saas Master |
| 5672 | tcp | AMQP Server | Saas Master |
| 10512 | tcp | Saas Master | Analytics Node |

## 

## Saas Dashboard/React ENV

|  |  |  |
| --- | --- | --- |
| **ENV Variable** | **Default Value** | **Description** |
| REACT\_APP\_SERVER\_URL | <https://saas.versa-networks.com> | Hosted server base url for internal url configuration |
| REACT\_APP\_SOCKET\_PATH | /nodesocket | Node socket server backend url |
| REACT\_APP\_SASS\_API | <https://saas.versa-networks.com/api/v1/organizations> | Backend API URL |
| REACT\_APP\_GOOGLE\_API\_KEY | AIzaSyApqNdcSz6tetfWE3LPtdJKPr6iT1Y5QFU | Google API Key for google maps |
| REACT\_APP\_EXPIRY\_DAYS\_DIFF\_COUNT | 90 | Device Expire limit to check in how many days the device is going to expire |
| REACT\_APP\_IDLE\_TIME | 900000 | React APP User IDLE time logout limit in seconds |
| REACT\_APP\_HOMEPAGE | /dashboard/ | Since we have dashboard in base url, for routing app will not consider the dashboard in the url since it takes only domain name so to set dashboard for routing page this is used |
| REACT\_APP\_ENVIRONMENT | dev | To verify which environment the app is deployed this variable is used to show the logos based on customer. |
| REACT\_APP\_ENVIRONMENT\_TITLE | Versa Customer Portal | To change the title of the application based on the different environment and customer |
| REACT\_APP\_OAUTH\_EXPIRY\_CHECK | 60 | Oauth Login Silent Refresh time before it expires |
| REACT\_APP\_SUPPORT\_URL | <https://v7groups.freshdesk.com> | Freshdesk support page url link |
| REACT\_APP\_LOGOUT\_TIME | 5000 | This field is to show the login again button in the logout page.  The login again button will be showed after the mentioned time.  This is field is not needed if the logout time is 5000. If we want to extend the logout time we have to add this field with the specific time in milliseconds. |

# 

## Node Socket ENV

|  |  |  |
| --- | --- | --- |
| **ENV Variable** | **Default Value** | **Description** |
| KAFKA\_HOST | <IP>:9094  Or  <IP1>:9094,<IP2>:9094,<IP3>:9094 | Define the kafka host , which will be used during message produced and consumed in the topic.  For single node the value will be broker:9094. Comma separated for multiple brokers (Multi-node HA). broker1:9094,broker2:9094,broker3:9094 |
| KAFKA\_TOPIC | versa-internal | Holds filtered Alarm logs in nodesocket server,which will be used during message produced and consumed in the topic. |
| SOCKET\_SERVER\_PORT | Default value is 80 | Defines the default server port for node socket server. |
| KAFKA\_SAAS\_SERVER | Default: https://saas.versa-networks.comapi/v1/organizations | Defines the api endpoint to verify the token when websocket connection is made. |

## Stream Processor ENV

|  |  |  |
| --- | --- | --- |
| **ENV Variable** | **Default Value** | **Description** |
| KAFKA\_HOST | <IP>:9094  Or  <IP1>:9094,<IP2>:9094,<IP3>:9094 | Define the kafka host , which will be used during message produced and consumed in the topic.  For single node the value will be broker:9094. Comma separated for multiple brokers (Multi-node HA). broker1:9094,broker2:9094,broker3:9094 |
| KAFKA\_MONITERING\_TOPIC | versa-monitoring | Holds filtered Alarm logs in monitoring instance,which will be used during message produced and consumed in the topic. |
| KAFKA\_VERSA\_INTERNAL\_TOPIC | versa-internal | Holds filtered alarm logs in node socket server,which will be used during message produced and consumed in the topic. |
| KAFKA\_ANALYTICS\_TOPIC | versa-analytics | Holds Alarm logs from reston vpn server via rsyslogs, which will be used during message produced and consumed in the topic. |
| KAFKA\_AMQP\_TOPIC | versa-amqp | Holds AMQP logs from reston vpn server via rsyslogs, which will be used during message produced and consumed in the topic. |
| KAFKA\_UPDATE\_TOPIC | versa-update-event | Holds filtered AMQP logs in reverse instance,which will be used during message produced and consumed in the topic. |
| MONGODB\_PORT |  | Defines mongodb port for the environment used while connecting to database. |
| MONGODB\_HOST |  | Defines mongodb host for the environment used while connecting to database. |
| MONGODB\_DATABASE |  | Defines the database name has to be connected. |
| MONGODB\_USER\_NAME |  | Defines the authenticate username for the database. |
| MONGODB\_PASSWORD |  | Defines the authenticate username for the database. |
| LOG\_FILE\_PATH | /var/log/debug.log | Defines log file path. |
| LOG\_FOLDER\_PATH | /var/log | Defines logs folder path. |
| LOG\_FILE\_SIZE\_MAX\_BYTE | 5242880 | Defines log file Maximum size in bytes. |
| LOG\_FILE\_BACKUP\_COUNT | 7 | Defines Maximum log file backup count. |
| LOG\_LEVEL | INFO,DEBUG,WARN,ERROR | Defines the levels of logs to be logged. |
| LOG\_TYPE | STREAM | Defines the log type. Accepted Values are “FILE, STREAM”. If Log type is “FILE”, then all logs are saved in files. If Log type is “STREAM”, then all log are pushed to stream/syslog. |
| KAFKA\_PARTITION\_KEY | Enterprise | Defines the partition key in the key value pair of the logs messages. |
| SECONDS\_THRESHOLD | 2 | Defines the seconds threshold to compare log messages timestamp to trigger reverse api. |
| APP\_DEBUG | 1 | Defines the seconds threshold to compare log messages timestamp to trigger reverse api. |
| REVERSE\_API\_ENDPOINT |  | Api endpoint to trigger reverse api |
| AUTH\_ENDPOINT |  | Api endpoint to trigger reverse api |
| AUTH\_USERNAME |  | Authenticate username for oauth server based on access token. |
| AUTH\_PASSWORD |  | Authenticate password for oauth server based on access token. |

## 

## Saas API ENV

|  |  |  |
| --- | --- | --- |
| **ENV Variable** | **Default Value** | **Description** |
| APP\_ENVIRONMENT | local | Define current app environment, By default value will be local |
| APP\_DEBUG | 0 | Define application is run with debug or not, By default the value will be 0 |
| APP\_API\_DOCUMENT | 0 | Define application should load the api document in root URI or not , By default the value will be 0 |
| KAFKA\_HOST | <IP>:9094  Or  <IP1>:9094,<IP2>:9094,<IP3>:9094 | Define the kafka host, which will be used during message produced.  For single node the value will be broker:9094. Comma separated for multiple brokers (Multi-node HA). broker1:9094,broker2:9094,broker3:9094 |
| KAFKA\_TOPIC | versa-internal | Define the kafka topic, the message from application going to produced, By default the value will be versa |
| ALLOWED\_HOSTS | \* | Define A list of strings representing the host/domain names that this Django site can serve, By default the value will be \* |
| LANGUAGE\_CODE | en-us | Define the language of the application, By default the value will be en-us |
| TIME\_ZONE | UTC | Define the timezone of the application, By default the value will be UTC |
| USE\_I18N | 1 | A boolean that specifies whether Django’s translation system should be enabled. This provides an easy way to turn it off, for performance. If this is set to False, Django will make some optimizations so as not to load the translation machinery. By default the value will be 1 |
| USE\_L10N | 1 | A boolean that specifies if localized formatting of data will be enabled by default or not. If this is set to True, e.g. Django will display numbers and dates using the format of the current locale. By default the value will be 1 |
| USE\_TZ | 1 | A boolean that specifies if datetimes will be timezone-aware by default or not. If this is set to True, Django will use timezone-aware datetimes internally. Otherwise, Django will use naive datetimes in local time. By default the value will be 1 |
| MONGODB\_PORT | 27017 | Set the MongoDB port which used during the connection to database from application, By default the value will be 27017 |
| MONGODB\_HOST | localhost | Set the MongoDB host which used during the connection to database from application, By default the value will be localhost. Comma separated for multiple mongo routers mongos1,mongos2,mongos3 |
| MONGODB\_DATABASE | versa | Set the MongoDB database application going to use, By default the value will be versa |
| MONGODB\_USER\_NAME |  | Set the MongoDB database authentication credentials user name, By default the value will be None |
| MONGODB\_PASSWORD |  | Set the MongoDB database authentication credentials password, By default the value will be None |
| TEST\_MONGODB\_HOST | localhost | Set the MongoDB host which used during the connection to database from application while running the test, By default the value will be localhost |
| TEST\_MONGODB\_PORT | 27017 | Set the MongoDB port which used during the connection to database from application while running the test, By default the value will be 27017 |
| TEST\_MONGODB\_DATABASE | versa\_test | Set the MongoDB database application going to use while running the test, By default the value will be versa\_test |
| TEST\_MONGODB\_USER\_NAME |  | Set the MongoDB database authentication credentials user name while running the test, By default the value will be None |
| TEST\_MONGODB\_PASSWORD |  | Set the MongoDB database authentication credentials password while running the test, By default the value will be None |
| CORS\_ORIGIN\_WHITELIST |  | A list of origin hostnames that are authorized to make cross-site HTTP requests. The value 'null' can also appear in this list, and will match the Origin: null header that is used in "privacy-sensitive contexts", such as when the client is running from a file:// domain. Defaults to []. |
| CORS\_ALLOW\_HEADERS |  | The list of non-standard HTTP headers that can be used when making the actual request. Defaults to:  CORS\_ALLOW\_HEADERS = (  'accept',  'accept-encoding',  'authorization',  'content-type',  'dnt',  'origin',  'user-agent',  'x-csrftoken',  'x-requested-with',  )  The default can be imported as corsheaders.defaults.default\_headers so you can extend it with your custom headers. This allows you to keep up to date with any future changes |
| DEFAULT\_GUEST\_WIFI\_NETWORK\_NAME | Guest | Define the guest WIFI network name or SSID, By default the value will be Guest |
| DEFAULT\_EMPLOYEE\_WIFI\_NETWORK\_NAME | Employee | Define the Employee WIFI network name or SSID, By default the value will be Employee |
| DEFAULT\_CORPORATE\_WIFI\_NETWORK\_NAME | Corporate | - Define the corporate WIFI network name or SSID, By default the value will be Corporate |
| DEFAULT\_NETWORK\_IP | None | Define the network IP used while generating DHCP pool for various LAN and WIFI interfaces, By default the value will be None |
| DEFAULT\_NAME\_SERVER\_PRIMARY | None | Define the default primary Name server used in the application, By default the value will be None |
| DEFAULT\_NAME\_SERVER\_SECONDARY | None | Define the default secondary Name server used in the application, By default the value will be None |
| DEFAULT\_INSTALL\_METHOD | gztp | Define the default installation or activation method for a device, valid value is gztp, By default the value will be gztp |
| ENTERPRISE\_DEVICE\_LIMIT | 890 | Define the maximum number of device allowed per organization, values should  Be a integer, By default the value will be 890, You need to make sure configuration is sync with Versa director configuration |
| DEVICE\_DEPLOY\_CALL\_TYPE | org | Define the device deploy call type based on the configuration call device deploy on various api actions, values should any of org/product/device, By default the value will be org |
| SITE\_DEVICE\_LIMIT | 2 | Define the maximum number of device allowed per site, values should  Be a integer, By default the value will be 2 |
| VD\_HOST | “” | Define the VD api endpoint which included the host and URI, By default the value will be Empty string |
| VD\_SOFTWARE\_VERSION | 16.1R2 | Define the VD api software version used, By default the value will be 16.1R2 |
| VD\_USER\_NAME | “” | Define the VD api basic authorization user name, By default the value will be Empty string |
| VD\_PASSWORD | “” | Define the VD api basic authorization password, By default the value will be Empty string |
| VD\_ENABLED | 0 | Define the VD api call is allowed or not, By default the value will be 0 |
| IP\_ALLOCATION\_START | 2 | Define the DHCP pool start IP address , valid value should be integer , the value make sure the ip allocation to the connected device will be alway equal or greater than defined here, By default the value will be 2 |
| TOTAL\_IP\_ALLOCATIONS | 1022 | Define the total ip address allocated per DHCP pool generated, valid value should be integer , the value will be used generate the end IP address of the network, By default the value will be 1022 |
| ORGANIZATION\_IDS\_RANGE | 6-512 | Define the Organization id start and end range, valid value should be a string and in format of start-end. So whenever a new organization is created application make sure the generated value will be greater or equal than start and lesser or equal than end, By default the value will be 6-512. You need to make sure configuration is sync with Versa director configuration |
| DEVICE\_UIDS\_RANGE | 101-16000 | Define the Device UID id start and end range, valid value should be a string and in format of start-end. So whenever a new device is created application make sure the generated value will be greater or equal than start and lesser or equal than end, By default the value will be 101-16000. You need to make sure configuration is sync with Versa director configuration |
| DEFAULT\_PROVIDER\_ORG | Provider | Define the default organization provider, this changes based on the Versa director configuration.  By default the value will be Provider |
| SMTP\_USER\_EMAIL |  | Username for simple mail transfer protocol to access the SMTP to send a email to users |
| SMTP\_USER\_PASSWORD |  | Password for simple mail transfer protocol to access the SMTP to send a email to users, This password is generated from google account. |
| FRESHDESK\_API\_KEY |  | API key for freshdesk account to access a freshdesk to create, update a ticket |
| FRESHDESK\_DOMAIN - |  | Domain name for freshdesk account |
| FRESHDESK\_PASSWORD |  | Password for freshdesk to access a account |
| FRESHDESK\_URL |  | Configure a freshdesk base url |
| FRESHDESK\_SECRET\_KEY |  | The shared secret acquired from freshdesk |
| FRESHDESK\_CLIENT\_ID |  | GLUU server client id |
| FRESHDESK\_CLIENT\_SECRET |  | GLUU server client secret key |
| FRESHDESK\_GRANT\_TYPE |  | Type of permission grant seeking for the GLUU server |
|  |  |  |
| FCM\_API\_KEY |  | API key for firebase account which is used to send a push notification to user mobile. |
| LOG\_FILE\_PATH | /var/log/debug.log | Defines log file path. Default value is “/var/log/debug.log” |
| LOG\_FOLDER\_PATH | /var/log | Defines logs folder path. Default value “/var/log” |
| LOG\_FILE\_SIZE\_MAX\_BYTE | 5242880 | Defines log file Maximum size in bytes. Default value is 5242880 |
| LOG\_FILE\_BACKUP\_COUNT | 7 | Defines Maximum log file backup count. Default value is 7 |
| LOG\_LEVEL | INFO,DEBUG,WARN,ERROR | Defines the levels of logs to be logged. Default value is “INFO,DEBUG,WARN,ERROR” |
| LOG\_TYPE | STREAM | Defines the log type. Accepted Values are “FILE” or “STREAM”. If Log type is “FILE”, then all logs are saved in files. If Log type is “STREAM”, then all log are pushed to stream/syslog. Default Value is “STREAM” |
| GOOGLE\_GEOCODE\_API\_KEY |  | AIzaSyApqNdcSz6tetfWE3LPtdJKPr6iT1Y5QFU |
| GOOGLE\_GEOCODE\_API |  | <https://maps.googleapis.com/maps/api/geocode/json>?address={0}&key={1} |
| ENABLE\_GLUU | 0 | This flag is used to validate a JWT token by using JOSE package and also flag is not enable than its validate by using django |
| JWKS\_URL |  | To validate a JWT token based on gluu configuration, This URL returns a JWT configuration. |
| OAUTH\_REDIRECT\_URI |  |  |
| OAUTH\_USERINFO\_ENDPOINT |  | This URL return a user information from oauth server based on access token |
| OAUTH\_USERS\_ENDPOINT |  | SCIM endpoint is used to access a oauth server to create, update, get and delete a user |
| OAUTH\_TOKEN\_ENDPOINT |  | To obtain an Access Token, an ID Token, and optionally a Refresh Token, the RP (Client) sends a Token Request to the Token Endpoint to obtain a Token Response. Token Endpoint requires Client Authentication methods to authenticate clients to the authorization server |
| OAUTH\_AUTHORIZATION\_ENDPOINT\_URI |  | The Authorization Endpoint performs Authentication of the end-user. This is done by sending the User Agent to the Authorization Server's Authorization Endpoint for Authentication and Authorization, using request parameters defined by OAuth 2.0 and additional parameters and parameter values defined by OpenID Connect. |
| OAUTH\_REGISTRATION\_ENDPOINT\_URI |  | This URL return a user register form which is given by oauth server to directly create a user |
| OAUTH\_LOGOUT\_SESSION\_ENDPOINT\_URI |  | This url is used to logout the user which is configured from oauth server |
| USER\_REGISTRATION\_ENDPOINT\_URI |  | This user register endpoint is used to register the user to client and oauth server and also this url is used to show the user register form in to mobile web page |
| CUSTOMER\_SUPPORT\_CHAT\_URI |  | This is base Url for customer support chat |
| CUSTOMER\_SUPPORT\_EXIT\_REDIRECT\_URI |  | This endpoint is used to exit a customer support chat |
| OAUTH\_BASE\_URL |  | configured oauth base url |
| SCIM\_CLIENT\_ID |  | SCIM API client id which is generated from oauth server is used to access a oauth users |
| SCIM\_CLIENT\_SECRET |  | SCIM API client id which is generated from oauth server is used to access a oauth users |
| IOS\_REDIRECT\_URI |  | Endpoint is used to redirect to ios app after user register from user registration page |
| ANDROID\_REDIRECT\_URI |  | Endpoint is used to redirect to android app after user register from user registration page |
| VERSA\_OPERATOR\_ORGANIZATION |  | This is used to create a operator while the user registration process (company name is versa) |
| VERIZON\_OPERATOR\_ORGANIZATION |  | This is used to create a operator while the user registration process (company name is verizon) |
| IMPORT\_AMAZON\_ORDER |  | Define the ECOMM api endpoint which included the host and URI, used to synchronize the amazon order with the application, By default the value will be Empty string |
| ANALYTICS\_HOST |  | HQ server endpoint to retrieve analytics data |
| ANALYTICS\_USERNAME |  | Username to access HQ server endpoints |
| ANALYTICS\_PASSWORD |  | Password to access HQ server endpoints |
| MOCK\_DATA\_URL |  | Mock data for user details(company, role, etc) in local API |
| LIVE\_MOCK\_DATA\_URL |  | User data coming from verizon based on user accountnumber and userid |
| MOCK\_DATA\_USER\_ID |  | User id for LIVE\_MOCK\_DATA\_URL |
| DEFAULT\_USER\_ATTRIBUTES |  | For userinfo mapping from auth server to database |
| MOCK\_API\_USERNAME |  | Username for LIVE\_MOCK\_DATA\_URL |
| MOCK\_API\_PASSWORD |  | Password for LIVE\_MOCK\_DATA\_URL |
| MOCK\_USER\_ROLE |  | Set operator as a default user role  For **siteminder** |
| MOCK\_USER\_COMPANY |  | Set test1 as a default user company  For **siteminder** |
| JWKS\_KEYS |  | This json payload contain jwks keys which is generated manually from the siteminder CERTIFICATE. This jwks key will serve from API JWKS\_URL for **Siteminder**. |
| TOKEN\_ISSUER |  | Set current oauth issuer  For **siteminder** |

## Init Mongo Dump

**application** - Holds the master for application and same used for validate the selected applications

**category** - Holds the master for reputations and same used for validate the selected categories

**ips** - Holds the master for ips and same used for validate the selected ips

**reputations** - Holds the master for reputations and same used for validate the selected ips

**roles** - Holds the role based api access information and same used for check api access for user

# Appendix B: Docker Stack File

## Version

Version is to indicate the docker *compose file format* version. It is required to specify because, the references used in the stack file should be compatible to this version.

When you execute deploy stack file command, docker validates the stack file before deploying referencing to the version we have specified.

|  |  |
| --- | --- |
| **Compose file format** | **Docker Engine release** |
| 3.7 | 18.06.0+ |
| 3.6 | 18.02.0+ |
| 3.5 | 17.12.0+ |
| 3.4 | 17.09.0+ |
| 3.3 | 17.06.0+ |
| 3.2 | 17.04.0+ |
| 3.1 | 1.13.1+ |
| 3 | 1.13.0+ |

|  |
| --- |
| **version: "3.5"** |

## Networks

Reference: <https://docs.docker.com/compose/compose-file/#network-configuration-reference>

### Creating the Overlay Network during stack deployment,

|  |
| --- |
| networks:  go:  name: go  driver: overlay  driver\_opts:  encrypted: "true"  proxy:  name: proxy  driver: overlay  driver\_opts:  encrypted: "true" |

#### **Driver: Overlay**

The default driver depends on how the Docker Engine you’re using is configured, but in most instances it is bridge on a single host and overlay on a Swarm. The Docker Engine returns an error if the driver is not available.

**Driver\_opts**

Reference: <https://docs.docker.com/compose/compose-file/#driver_opts>

**Encrypted: True**

Reference: <https://docs.docker.com/network/overlay/#encrypt-traffic-on-an-overlay-network>

### Referencing External

Referencing the external [Overlay networks created](#_v6oqm6wy4x1y) in before deployment.

If *external* set to *true*, specifies that this network has been created outside of Compose. *Docker stack deploy -c* does not attempt to create it, and raises an error if it doesn’t exist.

*external* cannot be used in conjunction with other network configuration keys (*driver, driver\_opts, ipam, internal*).

|  |
| --- |
| networks:  go:  external: true  proxy:  external: true |

### Adding Service to Overlay network

|  |
| --- |
| services:  haproxy:  **networks:  - proxy** |

## Volumes

Reference:

1. <https://docs.docker.com/compose/compose-file/#volumes>
2. <https://docs.docker.com/storage/volumes/>

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While [bind mounts](https://docs.docker.com/storage/bind-mounts/) are dependent on the directory structure of the host machine, volumes are completely managed by Docker. Volumes have several advantages over bind mounts:

* Volumes are easier to back up or migrate than bind mounts.
* You can manage volumes using Docker CLI commands or the Docker API.
* Volumes work on both Linux and Windows containers.
* Volumes can be more safely shared among multiple containers.
* Volume drivers let you store volumes on remote hosts or cloud providers, to encrypt the contents of volumes, or to add other functionality.
* New volumes can have their content pre-populated by a container.

|  |
| --- |
| volumes:  api-logs:  name: api-logs  zdata: {}  zlog: {}  data: {}  config: {} |

### Attaching/Binding Volumes to Service

|  |
| --- |
| services:  api:  image: versa\_api:0.7  networks:  - proxy  - go  **volumes:  - type: volume  source: api-logs  target: /var/www/html/logs** |

* The ***type*** of the mount, which can be [*bind*](https://docs.docker.com/storage/bind-mounts/)*, volume*. This topic discusses volumes, so the type is always ***volume***.
* The ***source*** of the mount. For named volumes, this is the name of the volume. For anonymous volumes, this field is omitted. May be specified as *source* or *src*.
* The ***destination*** takes as its value the path where the file or directory is mounted in the container. May be specified as *destination*, *dst*, or *target*.

## Secrets

Reference:

1. <https://docs.docker.com/engine/swarm/secrets/>
2. <https://docs.docker.com/compose/compose-file/#secrets>

Grant access to secrets on a per-service basis using the per-service secrets configuration

|  |
| --- |
| secrets:  kafka-keystore:  file: ./configs/kafka/server.keystore.jks  kafka-truststore:  file: ./configs/kafka/server.truststore.jks |

### Attaching Secrets to a Service

After attaching, the secret files can be accessed inside the container in the target path specified.

|  |
| --- |
| services:  broker1:  image: versa\_kafka-docker:1  networks:  - go  **secrets:  - source: kafka-keystore  target: /etc/ssl/server.keystore.jks  - source: kafka-truststore  target: /etc/ssl/server.truststore.jks** |

* ***source***: The name of the secret as it exists in Docker.
* ***target***: The name of the file to be mounted in /run/secrets/ in the service’s task containers. Defaults to source if not specified.
* ***uid*** and ***gid***: The numeric UID or GID that owns the file within /run/secrets/ in the service’s task containers. Both default to 0 if not specified.
* ***mode***: The permissions for the file to be mounted in /run/secrets/ in the service’s task containers, in octal notation. Example: 0644, 0755, 400

## Configs

Reference:

1. <https://docs.docker.com/engine/swarm/configs/>
2. <https://docs.docker.com/compose/compose-file/#configs-configuration-reference>

The top-level configs declaration defines or references [configs](https://docs.docker.com/engine/swarm/configs/) that can be granted to the services in this stack. The source of the config is either file or external.

|  |
| --- |
| configs:  haproxy-env:  file: ./configs/haproxy\_saas.cfg  api-env:  file: ./configs/api.env  dashboard-env:  file: ./configs/dashboard.env  nodesocket-env:  file: ./configs/nodesocket.env  streamprocessor-env:  file: ./configs/streamprocessor.env  device-facing-api-env:  file: ./configs/device\_facing\_api.yml |

* ***file***: The config is created with the contents of the file at the specified path.
* ***external***: If set to true, specifies that this config has already been created. Docker does not attempt to create it, and if it does not exist, a config not found error occurs.
* ***name***: The name of the config object in Docker. This field can be used to reference configs that contain special characters. The name is used as is and will **not** be scoped with the stack name. Introduced in version 3.5 file format.

### Attaching Config to a Service

|  |
| --- |
| services:  haproxy:  image: versa\_haproxy  networks:  - proxy  **configs:  - source: haproxy-env  target: /usr/local/etc/haproxy/haproxy.cfg** |

* ***source***: The name of the config as it exists in Docker.
* ***target***: The name of the file to be mounted in /usr/local/etc/haproxy/ in the service’s task containers. Defaults to source if not specified.
* ***uid*** and ***gid***: The numeric UID or GID that owns the file within /usr/local/etc/haproxy/ in the service’s task containers. Both default to 0 if not specified.
* ***mode***: The permissions for the file to be mounted in /usr/local/etc/haproxy/ in the service’s task containers, in octal notation. Example: 0644, 0755, 400

## Image

Reference:

1. <https://docs.docker.com/compose/compose-file/#image>

Specify the image to start the container from. Can either be a repository/tag or a partial image ID.

|  |
| --- |
| services:  haproxy:  **image: versa\_haproxy** |

### 

### Examples

|  |
| --- |
| image: redis image: ubuntu:14.04 image: tutum/influxdb image: example-registry.com:4000/postgresql image: a4bc65fd |

## Ports

Reference:

1. <https://docs.docker.com/compose/compose-file/#ports>

Expose ports. Either specify both ports (HOST:CONTAINER), or just the container port (an ephemeral host port is chosen).

Port mapping is incompatible with *network\_mode: host*

The long form syntax allows the configuration of additional fields that can’t be expressed in the short form.

* ***target***: the port inside the container
* ***published***: the publicly exposed port
* ***protocol***: the port protocol (tcp or udp)
* ***mode***: *host* for publishing a host port on each node, or *ingress* for a swarm mode port to be load balanced.

|  |
| --- |
| services:  haproxy:  image: versa\_haproxy  networks:  - proxy  **ports:  - target: 80  published: 80  protocol: tcp  mode: host** |

## Logging

Logging configuration for the service.

|  |
| --- |
| logging:  driver: syslog  options:  syslog-address: "tcp://192.168.0.42:123" |

The driver name specifies a logging driver for the service’s containers, as with the --log-driver option for docker run ([documented here](https://docs.docker.com/engine/admin/logging/overview/)).

|  |
| --- |
| services:  haproxy:  image: versa\_haproxy  **logging:  driver: syslog  options:  syslog-address: "udp://127.0.0.1:514"  tag: "haproxy/{{.ImageName}}/{{.ID}}"  syslog-facility: local3  syslog-format: rfc5424micro** |

For Logging Options check here: <https://docs.docker.com/config/containers/logging/syslog/#options>

|  |  |  |
| --- | --- | --- |
| **Option** | **Description** | **Example value** |
| syslog-address | The address of an external syslog server. The URI specifier may be [tcp | udp|tcp+tls]://host:port, unix://path, or unixgram://path. If the transport is tcp, udp, or tcp+tls, the default port is 514. | --log-opt syslog-address=tcp+tls://192.168.1.3:514, --log-opt syslog-address=unix:///tmp/syslog.sock |
| syslog-facility | The syslog facility to use. Can be the number or name for any valid syslog facility. See the [syslog documentation](https://tools.ietf.org/html/rfc5424#section-6.2.1). | --log-opt syslog-facility=daemon |
| tag | A string that is appended to the APP-NAME in the syslog message. By default, Docker uses the first 12 characters of the container ID to tag log messages. Refer to the [log tag option documentation](https://docs.docker.com/config/containers/logging/log_tags/) for customizing the log tag format. | --log-opt tag=mailer |
| syslog-format | The syslog message format to use. If not specified the local UNIX syslog format is used, without a specified hostname. Specify rfc3164 for the RFC-3164 compatible format, rfc5424 for RFC-5424 compatible format, or **rfc5424micro** for RFC-5424 compatible format with microsecond timestamp resolution. | --log-opt syslog-format=rfc5424micro |

## Command

Override the default command mentioned in the [dockerfile](https://docs.docker.com/engine/reference/builder/#cmd) while building the base image.

|  |
| --- |
| services:  haproxy:  image: versa\_haproxy  **command: /bin/sh -c "/sbin/syslogd -O /proc/1/fd/1 && haproxy -f /usr/local/etc/haproxy/haproxy.cfg"** |

The command can also be a list, in a manner similar to [dockerfile](https://docs.docker.com/engine/reference/builder/#cmd):

|  |
| --- |
| command: ["haproxy", "-f", "/usr/local/etc/haproxy/haproxy.cfg"] |

### HAProxy

|  |
| --- |
| command: /bin/sh -c "/sbin/syslogd -O /proc/1/fd/1 && haproxy -f /usr/local/etc/haproxy/haproxy.cfg" |

*/sbin/syslogd -O /proc/1/fd/1* - This is to enable syslog inside haproxy container, to log the client IP in the log.

*haproxy -f /usr/local/etc/haproxy/haproxy.cfg* - Start the haproxy with the configuration.

### MySQL

Reference: <http://galeracluster.com/documentation-webpages/mysqlwsrepoptions.html>

Galera First Service

|  |
| --- |
| command: --wsrep-**cluster**-address=gcomm:// --wsrep-node-address=galera1 |

Galera Other Services

|  |
| --- |
| command: --wsrep-**cluster**-address=gcomm://galera1 --wsrep-node-address=galera2 |

**--wsrep-cluster-address**

<http://galeracluster.com/documentation-webpages/mysqlwsrepoptions.html#wsrep-cluster-address>

**--wsrep-node-address**

<http://galeracluster.com/documentation-webpages/mysqlwsrepoptions.html#wsrep-node-address>

### MongoDB

#### 

#### **Data (mongodb\_cluster.yml)**

|  |
| --- |
| command: mongod --keyFile /data/key --shardsvr --replSet datars --port 27017 |

*--keyfile /data/key* - <https://docs.mongodb.com/manual/tutorial/enforce-keyfile-access-control-in-existing-replica-set/#enforce-keyfile-access-control-on-existing-replica-set>

*--shardsvr* - Use this only for sharding mongodb cluster and not for mongodb replica set

*--replSet datars* - Only for replica set <https://docs.mongodb.com/manual/tutorial/deploy-replica-set/#procedure>

*--port 27017* - Exposes a port.

#### **Config (mongodb\_cluster.yml)**

|  |
| --- |
| command: mongod --keyFile /data/key --configsvr --replSet cfgrs --port 27017 |

*--keyfile /data/key* - <https://docs.mongodb.com/manual/tutorial/enforce-keyfile-access-control-in-existing-replica-set/#enforce-keyfile-access-control-on-existing-replica-set>

*--configsvr* - Use this only for sharding mongodb cluster and not for mongodb replica set. <https://docs.mongodb.com/manual/tutorial/deploy-shard-cluster/#create-the-config-server-replica-set>

*--replSet cfgrs* - Only for replica set <https://docs.mongodb.com/manual/tutorial/deploy-replica-set/#procedure>

*--port 27017* - Exposes a port.

#### 

#### **Mongos Router (mongodb\_cluster.yml)**

|  |
| --- |
| command: mongos --keyFile /data/key --configdb cfgrs/cfg1:27017,cfg2:27017,cfg3:27017 --bind\_ip 0.0.0.0 |

*--keyfile /data/key* - <https://docs.mongodb.com/manual/tutorial/enforce-keyfile-access-control-in-existing-replica-set/#enforce-keyfile-access-control-on-existing-replica-set>

--configdb - List out the config services host:port separated by comma. <https://docs.mongodb.com/manual/tutorial/deploy-shard-cluster/#connect-a-mongos-to-the-sharded-cluster>

*--bind-ip 0.0.0.0* - The IP addresses and/or full Unix domain socket paths on which [mongos](https://docs.mongodb.com/manual/reference/program/mongos/#bin.mongos) and [mongod](https://docs.mongodb.com/manual/reference/program/mongod/#bin.mongod) should listen for client connections.

#### **Replica Set Mongo (mongodb\_replicas.yml)**

|  |
| --- |
| command: mongod --keyFile /data/key --replSet datars --bind\_ip 0.0**.0.0** |

*--keyfile /data/key* - <https://docs.mongodb.com/manual/tutorial/enforce-keyfile-access-control-in-existing-replica-set/#enforce-keyfile-access-control-on-existing-replica-set>

*--replSet datars* - Only for replica set <https://docs.mongodb.com/manual/tutorial/deploy-replica-set/#procedure>

*--bind-ip 0.0.0.0* - The IP addresses and/or full Unix domain socket paths on which [mongos](https://docs.mongodb.com/manual/reference/program/mongos/#bin.mongos) and [mongod](https://docs.mongodb.com/manual/reference/program/mongod/#bin.mongod) should listen for client connections.

## Deploy

Reference: <https://docs.docker.com/compose/compose-file/#deploy>

Specify configuration related to the deployment and running of services. This only takes effect when deploying to a [swarm](https://docs.docker.com/engine/swarm/) with [docker stack deploy](https://docs.docker.com/engine/reference/commandline/stack_deploy/)

### Global Mode

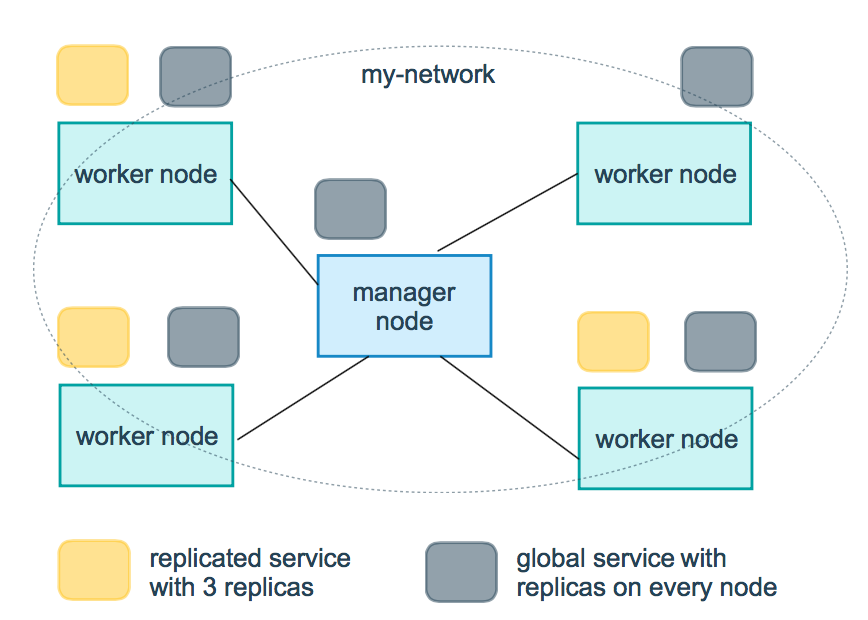
|  |
| --- |
| services:  haproxy:  image: versa\_haproxy  networks:  - proxy  ports:  - target: 80  published: 80  protocol: tcp  mode: host  configs:  - source: haproxy-env  target: /usr/local/etc/haproxy/haproxy.cfg  logging:  driver: syslog  options:  syslog-address: "udp://127.0.0.1:514"  tag: "haproxy/{{.ImageName}}/{{.ID}}"  syslog-facility: local3  syslog-format: rfc5424micro  command: /bin/sh -c "/sbin/syslogd -O /proc/1/fd/1 && haproxy -f /usr/local/etc/haproxy/haproxy.cfg"  **deploy:  mode: global** |

### Replicated Mode

|  |
| --- |
| services:  zookeeper1:  image: versa\_zookeeper  **deploy:  mode: replicated  replicas: 1  placement:  constraints:  - node.labels.deploy.role == manager0** |

#### **Mode**

Reference: <https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/#replicated-and-global-services>



### Replicas

If the service is *replicated* (which is the default), specify the number of containers that should be running at any given time (When the stack is deployed).

#### **Placement: Constraints**

Reference: <https://docs.docker.com/engine/reference/commandline/service_create/#specify-service-constraints---constraint>

You can limit the set of nodes where a task can be scheduled by defining constraint expressions. Multiple constraints find nodes that satisfy every expression (AND match).

We use node.labels for defining the constraints. Swarm administrators add node.labels for operational purposes by using the [*docker node update*](https://docs.docker.com/engine/reference/commandline/node_update/) command.

Adding node.labels to the node,

|  |
| --- |
| $ sudo docker node **update** --label-add **deploy.role=manager0** manager0 $ sudo docker node **update** --label-add deploy.role=manager1 manager1 $ sudo docker node **update** --label-add deploy.role=manager2 manager2 |

Specify the node.label in the placement constraints,

|  |
| --- |
| services:  zookeeper1:  image: versa\_zookeeper  deploy:  mode: replicated  replicas: 1  placement: constraints:  - node.labels.**deploy.role == manager0** |

## Environment Variables

Specify Environment values in the stack file, so that can be used as an Environment variables inside the container.

### Zookeeper

Details: <https://docs.docker.com/samples/library/zookeeper/#replicated-mode>

|  |
| --- |
| services:  zookeeper1:  image: versa\_zookeeper  **environment:  ZOO\_MY\_ID: 1  ZOO\_SERVERS: server.1=0.0.0.0:2888:3888 server.2=zookeeper2:2888:3888 server.3=zookeeper3:2888:3888** |

### Kafka Broker

Details:

1. <https://github.com/wurstmeister/kafka-docker>
2. <https://kafka.apache.org/documentation/#configuration>

|  |
| --- |
| broker1:  image: versa\_kafka-docker:1  **environment:  KAFKA\_BROKER\_ID: 1  KAFKA\_AUTO\_CREATE\_TOPICS\_ENABLE: "false"  KAFKA\_ADVERTISED\_LISTENERS: INSIDE://:9092,OUTSIDE://:9094  KAFKA\_LISTENERS: INSIDE://:9092,OUTSIDE://:9094  KAFKA\_INTER\_BROKER\_LISTENER\_NAME: INSIDE  KAFKA\_ZOOKEEPER\_CONNECT: zookeeper1:2181,zookeeper2:2181,zookeeper3:2181  KAFKA\_ZOOKEEPER\_CONNECTION\_TIMEOUT\_MS: 16000  KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT  KAFKA\_ADVERTISED\_PROTOCOL\_NAME: OUTSIDE  KAFKA\_ADVERTISED\_PORT: 9094  KAFKA\_PROTOCOL\_NAME: INSIDE  KAFKA\_PORT: 9092  KAFKA\_JMX\_OPTS: "-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=true -Dcom.sun.management.jmxremote.ssl=true -Dcom.sun.management.jmxremote.rmi.port=31099 -Dcom.sun.management.jmxremote.password.file=/etc/ssl/jmxremote.password -Dcom.sun.management.jmxremote.access.file=/etc/ssl/jmxremote.access -Djavax.net.ssl.keyStore=/etc/ssl/keystore.jks -Djavax.net.ssl.keyStorePassword=kafkA34T -Djavax.net.ssl.trustStore=/etc/ssl/truststore.jks -Djavax.net.ssl.trustStorePassword=kafkA34T"**  volumes:  - /var/run/docker.sock:/var/run/docker.sock  deploy:  mode: replicated  replicas: 1  placement:  constraints:  - node.labels.deploy.role == manager0 |

#### **KAFKA\_BROKER\_ID**

<https://github.com/wurstmeister/kafka-docker#broker-ids>

The broker id for this server. If unset, a unique broker id will be generated.To avoid conflicts between zookeeper generated broker id's and user configured broker id's, generated broker ids start from reserved.broker.max.id + 1.

#### **KAFKA\_AUTO\_CREATE\_TOPICS\_ENABLE**

Enable auto creation of topic on the server. False will disable the auto creation of topic.

#### **KAFKA\_ADVERTISED\_LISTENERS**

Listeners to publish to ZooKeeper for clients to use, if different than the `listeners` config property. In IaaS environments, this may need to be different from the interface to which the broker binds. If this is not set, the value for `listeners` will be used. Unlike `listeners` it is not valid to advertise the 0.0.0.0 meta-address.

#### **KAFKA\_LISTENERS**

Listener List - Comma-separated list of URIs we will listen on and the listener names. If the listener name is not a security protocol, listener.security.protocol.map must also be set. Specify hostname as 0.0.0.0 to bind to all interfaces. Leave hostname empty to bind to default interface. Examples of legal listener lists: PLAINTEXT://myhost:9092,SSL://:9091 CLIENT://0.0.0.0:9092,REPLICATION://localhost:9093

#### **KAFKA\_INTER\_BROKER\_LISTENER\_NAME**

Name of listener used for communication between brokers. If this is unset, the listener name is defined by security.inter.broker.protocol. It is an error to set this and security.inter.broker.protocol properties at the same time.

#### **KAFKA\_ZOOKEEPER\_CONNECT**

Zookeeper host string.

Specifies the ZooKeeper connection string in the form hostname:port where host and port are the host and port of a ZooKeeper server. To allow connecting through other ZooKeeper nodes when that ZooKeeper machine is down you can also specify multiple hosts in the form hostname1:port1,hostname2:port2,hostname3:port3.

#### **KAFKA\_ZOOKEEPER\_CONNECTION\_TIMEOUT\_MS**

The max time that the client waits to establish a connection to zookeeper. If not set, the value in zookeeper.session.timeout.ms is used

#### **KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP**

Map between listener names and security protocols. This must be defined for the same security protocol to be usable in more than one port or IP. For example, internal and external traffic can be separated even if SSL is required for both. Concretely, the user could define listeners with names INTERNAL and EXTERNAL and this property as: `INTERNAL:SSL,EXTERNAL:SSL`.

As shown, key and value are separated by a colon and map entries are separated by commas. Each listener name should only appear once in the map. Different security (SSL and SASL) settings can be configured for each listener by adding a normalised prefix (the listener name is lowercased) to the config name. For example, to set a different keystore for the INTERNAL listener, a config with name `listener.name.internal.ssl.keystore.location` would be set. If the config for the listener name is not set, the config will fallback to the generic config (i.e. `ssl.keystore.location`).

#### **KAFKA\_ADVERTISED\_PORT**

OPTIONAL: only used when `advertised.listeners` or `listeners` are not set. Use `advertised.listeners` instead. The port to publish to ZooKeeper for clients to use. In IaaS environments, this may need to be different from the port to which the broker binds. If this is not set, it will publish the same port that the broker binds to.

#### **KAFKA\_PORT**

OPTIONAL: only used when `listeners` is not set. Use `listeners` instead. the port to listen and accept connections on

#### **KAFKA\_JMX\_OPTS**

For monitoring purposes you may wish to configure JMX. Additional to the standard JMX parameters, problems could arise from the underlying RMI protocol used to connect

* java.rmi.server.hostname - interface to bind listening port
* com.sun.management.jmxremote.rmi.port - The port to service RMI requests
* Check the [JMX SSL Options](https://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html#gdeum) details.

### Stream Processor

The environment variable STREAM\_PROCESSOR\_CATEGORY is used to categories the functionality of stream processor services.

#### **Values**

FILTERING, MONITORING or UPDATE\_EVENT

### MySQL

#### **Galera First Primary Service**

|  |
| --- |
| environment:  MYSQL\_DATABASE: shop  MYSQL\_ROOT\_PASSWORD: Ver@Ka98Ij  MYSQL\_USER: versa  MYSQL\_PASSWORD: versa123 |

#### **Galera Other Services**

|  |
| --- |
| environment:  MYSQL\_INIT\_CLUSTER\_ADDRESS: galera1  MYSQL\_ROOT\_PASSWORD: Ver@Ka98Ij |

#### **MYSQL\_DATABASE**

Application Database name which would be created automatically if not exist.

#### **MYSQL\_ROOT\_PASSWORD**

Mandatory. Mysql root user password, without this root user will not be created and mysql galera service will be down.

#### **MYSQL\_USER and MYSQL\_PASSWORD**

Mysql Username and password to access the application database from the application.

#### **MYSQL\_INIT\_CLUSTER\_ADDRESS**

We need to specify the galera node 1 or initial node of the cluster. As per the galera installation, one node should be initiate the cluster and other node should added one by one later.

Here this environment variable is used to check the if the initial galera node is up, and waits till its up and running. Once the initial node is up, the current node will get added to the cluster.

# 

# Appendix C: Frequently Asked Questions

**Q. What are the benefits of the cluster? How do we make sure the cluster is healthy? Any status calls? Connectivity checks?**

Some of the benefits of Cluster,

*Increased resource availability:* If one Server in a cluster fails, the other Servers in the cluster can pick up the workload. This prevents the loss of valuable time and information if a server fails.

*Increased performance:* Multiple machines provide greater processing power.

*Greater scalability:* As your user base grows and report complexity increases, your resources can grow.

We use swarm cluster in Saas Master (6 VM’s - 3 Manager and 3 Worker). This is the administration reference for docker swarm,

<https://docs.docker.com/engine/swarm/admin_guide/>

To make sure the cluster is healthy you should know about the fault-tolerance matrix of the swarm cluster,

You should maintain an odd number of managers in the swarm to support manager node failures. For example, in a swarm with 5 nodes, if you lose 3 nodes, you don’t have a quorum. Therefore you can’t add or remove nodes until you recover one of the unavailable manager nodes or recover the swarm with disaster recovery commands. See [Recover from disaster](https://docs.docker.com/engine/swarm/admin_guide/#recover-from-disaster).

|  |  |  |
| --- | --- | --- |
| **Swarm Manager Nodes** | **Majority** | **Fault Tolerance** |
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 2 | 1 |
| 4 | 3 | 1 |
| 5 | 3 | 2 |
| 6 | 4 | 2 |
| 7 | 4 | 3 |
| 8 | 5 | 3 |
| 9 | 5 | 4 |

Use this command (in any of the manager node) to see the list of nodes in the cluster its status, availability, manager status.

|  |
| --- |
| **$** sudo docker node ls ID HOSTNAME MEMBERSHIP STATUS AVAILABILITY MANAGER STATUS 1mhtdwhvsgr3c26xxbnzdc3yp node05 Accepted Ready Active 516pacagkqp2xc3fk9t1dhjor node02 Accepted Ready Active Reachable 9ifojw8of78kkusuc4a6c23fx \* node01 Accepted Ready Active Leader ax11wdpwrrb6db3mfjydscgk7 node04 Accepted Ready Active bb1nrq2cswhtbg4mrsqnlx1ck node03 Accepted Ready Active Reachable di9wxgz8dtuh9d2hn089ecqkf node06 Accepted Ready Active |

In 3 Manager cluster, if all 3 managers are up (worker’s manager status value is empty) then the cluster is healthy. If 2 managers are up and one is down, the cluster is still healthy (new leader would be elected) but can't withstand another failure. ([How to recover from losing quorum?](https://docs.docker.com/engine/swarm/admin_guide/#recover-from-losing-the-quorum))

**Using Haproxy to check node connectivity**

To enable the HAProxy stats page, add the following to the bottom of the file /etc/haproxy/haproxy.cfg (adding your own username and password to the final line):

|  |
| --- |
| listen stats *# Define a listen section called "stats"*  bind :9000 *# Listen on localhost:9000*  mode http  stats enable *# Enable stats page*  stats hide-version *# Hide HAProxy version*  stats realm Haproxy\ Statistics *# Title text for popup window*  stats uri /haproxy\_stats *# Stats URI*  stats auth Username:Password *# Authentication credentials* |

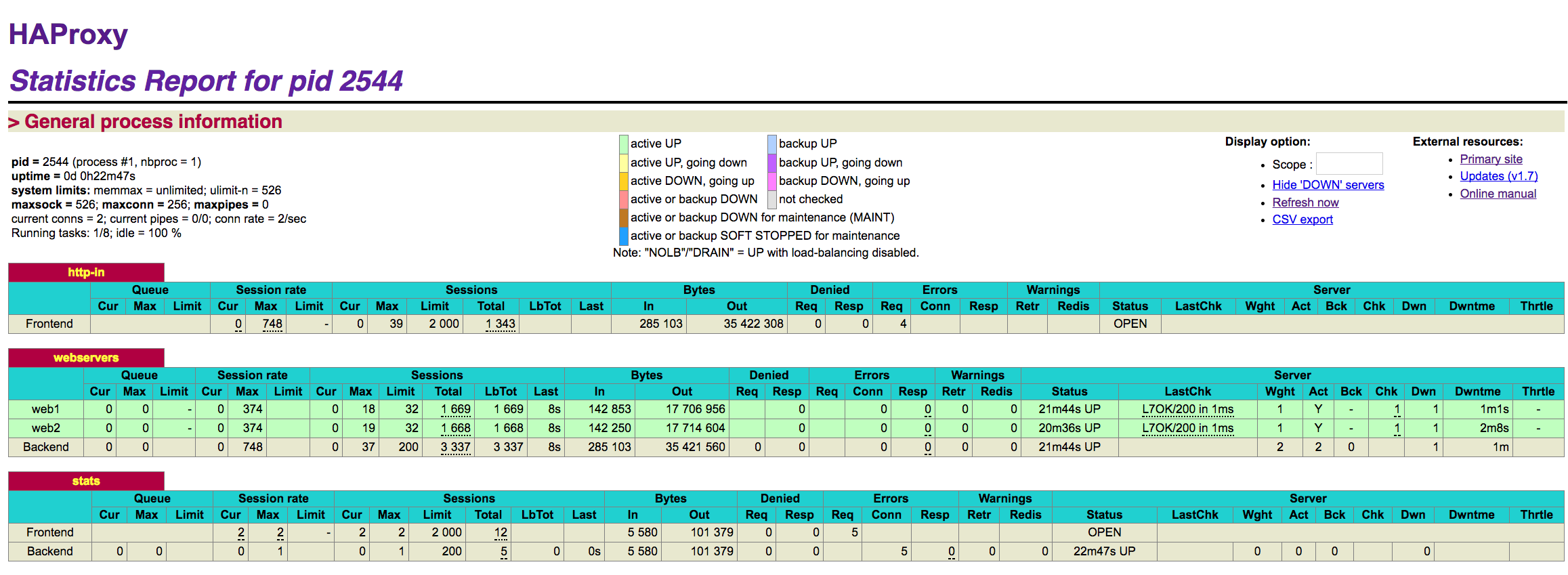
If you’re comfortable with session interruption, you can restart HAProxy with

|  |
| --- |
| **$** sudo service haproxy restart |

After restarting HAProxy with your modified configuration, you can access a stats page like the one below after authenticating via the,

|  |
| --- |
| URL: http://<YourHAProxyServer>:9000/haproxy\_stats |

You would see the stats page as given below,

****

For more information: <https://www.datadoghq.com/blog/how-to-collect-haproxy-metrics/>

## Backup and Restore MongoDB (Logical)

To backup mongodb we should perform mongodump,

|  |
| --- |
| **#** List the running containers with Container ID **$** sudo docker ps --filter "name=mongodb\_mongo" --format "table {{.ID}}\t{{.Names}}"  **$** docker exec -it <Container\_ID\_Mongodb\_Mongo> bash root@f4ca97d48c3c:/# mkdir -p /backup root@f4ca97d48c3c:/# mongodump --db versa\_saas --username versasaas --password Versa@19 --authenticationDatabase versa\_saas -o /backup/ root@f4ca97d48c3c:/# cd /backup root@f4ca97d48c3c:/# tar -zcvf versa\_saas\_<Date>.tar versa\_saas root@f4ca97d48c3c:/# gzip versa\_saas\_<Date>.tar  **#** Output file would be versa\_saas\_<Date>.tar.gz root@f4ca97d48c3c:/# exit |

**Copy the dump to your host**

|  |
| --- |
| $ mkdir -p /home/versa/backup\_mongo  $ cd /home/versa/backup\_mongo  $ sudo docker ps --filter "name=mongodb\_mongo" --format "table {{.ID}}\t{{.Names}}"  *# Copy the dump from container to current folder in host machine* $ docker cp <Container\_ID\_Mongodb\_Mongo>:/backup/versa\_saas\_<Date>.tar.gz . |

**Restore Mongodb Dump**

You could see a folder **backup\_mongo**,which you have checked out from repo

|  |
| --- |
| # List the running containers with Container ID  **$** sudo docker ps --filter "name=mongodb\_mongo" --format "table {{.ID}}\t{{.Names}}"  $ sudo docker cp /home/versa/backup\_mongo/versa\_saas\_<Date>.tar.gz <Container\_ID\_Mongo1>:/backup  **$** sudo docker exec -it <Container\_ID\_Mongo1> bash  $ cd /backup  $ gunzip versa\_saas\_<Date>.tar.gz  $ tar -zxvf versa\_saas\_<Date>.tar  # You can see a folder named versa\_saas  root@f4ca97d48c3c:/# mongorestore --db versa\_saas --username versa --password versa123 --authenticationDatabase versa\_saas **/backup/versa\_saas**  root@f4ca97d48c3c:/# mongo  **mongo>** use versa\_saas;  **mongo>** db.auth("versasaas", "Versa@19")  **mongo>** show collections;  **mongo>** exit  root@f4ca97d48c3c:/# exit |

# 

## 

## Backup and Restore MongoDB (Physical)

If your storage system does not support snapshots, you can copy the files directly using **cp, rsync**, or a similar tool. Since copying multiple files is not an atomic operation, you must stop all writes to the [mongod](https://docs.mongodb.com/manual/reference/program/mongod/#bin.mongod) before copying the files. Otherwise, you will copy the files in an invalid state.

Here in our architecture we have 6 mongodb replication docker services, which is located in each swarm node. Each service will be attached with a docker volume mongodb\_data1 - mongodb\_data6, located in accordingly in each node respectively.

We should take backup of this files inside this volume. Taking backup from one node is enough, even if we have 6 nodes. As we have replica set, once the data is restored, it gets replicated to other nodes.

The data files are located in this path, /var/lib/docker/volumes/mongodb\_data1/\_data. How to find this path/mount point ?

|  |
| --- |
| **$** sudo docker volume ls --filter "name=mongodb\_data" **$** docker volume inspect --format "{{ .Mountpoint }}" <Volume\_Name\_mongodb\_data1>  **#** Example: sudo docker volume inspect --format "{{ .Mountpoint }}" mongodb\_data1 **#** Result: /var/lib/docker/volumes/mongodb\_data1/\_data |

Take a backup of this data,

|  |
| --- |
| **$** cd /var/lib/docker/volumes/mongodb\_data1/ **$** tar -zcvf mongodb\_data\_<Date>.tar \_data/\*  **$** gzip mongodb\_data\_<Date>.tar # Resulting file would be mongodb\_data\_<Date>.tar.gz |

After taking backup you can push to backup server using scp.

**Restore backup**

|  |
| --- |
| # Remove mongodb completely **$** docker stack rm mongodb  # Remove volumes data in each node, mongodb\_data1 to mongodb\_data6 **$** rm -rf /var/lib/docker/volumes/mongodb\_data1/\_data/\*  # Copy the backup data **$** cp mongodb\_data\_<Date>.tar.gz /var/lib/docker/volumes/mongodb\_data1/\_data/  **$** gunzip mongodb\_data\_<Date>.tar.gz **$** tar -zxvf mongodb\_data\_<Date>.tar  # Check data is available in \_data folder **$** ls -l /var/lib/docker/volumes/mongodb\_data1/\_data/  # Now start or deploy mongodb again **$** cd /home/versa/deployment\_stack/stacks **$** sudo docker stack deploy --compose-file mongodb\_replicas.yml mongodb --with-registry-auth |

## Backup and Restore Kafka-Zookeeper

Here in our architecture we have 3 broker and 3 Zookeeper docker services, which is located in each swarm node. Each zookeeper service will be attached with 2 docker volume kafka\_zdata1 (in-memory database) and kafka\_zlog1 (Transaction logs). The volume will be available in 3 nodes.

We should take backup of this files inside these zookeeper volume. Taking backup from one node is enough, even if we have 6 nodes. As we have replicas of zookeeper, once the data is restored, it gets replicated to other nodes.

The data files are located in this path, /var/lib/docker/volumes/kafka\_zdata1/\_data. We can also take backup of volume kafka\_zlog (Transaction logs), repeat the steps given below. How to find this path/mount point ?

|  |
| --- |
| **$** sudo docker volume ls --filter "name=kafka\_zdata" **$** docker volume inspect --format "{{ .Mountpoint }}" <Volume\_Name\_kafka\_zdata>  **#** Example: sudo docker volume inspect --format "{{ .Mountpoint }}" kafka\_zdata1 **#** Result: /var/lib/docker/volumes/kafka\_zdata1/\_data |

Take a backup of this data,

|  |
| --- |
| **$** cd /var/lib/docker/volumes/kafka\_zdata1/ **$** tar -zcvf kafka\_zdata\_<Date>.tar \_data/\*  **$** gzip kafka\_zdata\_<Date>.tar # Resulting file would be mongodb\_data\_<Date>.tar.gz |

After taking backup you can push to backup server using scp.

### Restore Backup

|  |
| --- |
| # Remove Kafka completely **$** docker stack rm kafka  # Remove volumes data in each node, kafka\_zdata1 to kafka\_zdata3 **$** rm -rf /var/lib/docker/volumes/kafka\_zdata1/\_data/\*  # Copy the backup data **$** cp kafka\_zdata\_<Date>.tar.gz /var/lib/docker/volumes/kafka\_zdata1/\_data/  **$** gunzip kafka\_zdata\_<Date>.tar.gz **$** tar -zxvf kafka\_zdata\_<Date>.tar  # Check data is available in \_data folder **$** ls -l /var/lib/docker/volumes/kafka\_zdata1/\_data/  # Now start or deploy mongodb again **$** cd /home/versa/deployment\_stack/stacks **$** sudo docker stack deploy --compose-file mongodb\_replicas.yml mongodb --with-registry-auth |

## Re-create Kafka Topics

If we cannot restore zookeeper data, which contains the kafka topics, then we can redeploy and recreate topics,

|  |
| --- |
| # List the running containers with Container ID  **$** sudo docker ps --filter "name=kafka\_broker1" --format "table {{.ID}}\t{{.Names}}"  **$** sudo docker exec -it <Container\_ID\_Kafka\_Broker1> sh  **$** cd /opt/kafka  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 4 -partitions 8 --topic versa-analytics  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-internal  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-monitoring  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-amqp  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-device-backend-update  **$** bin/kafka-topics.sh --create --zookeeper zookeeper1:2181 --replication-factor 3 -partitions 6 --topic versa-update-event  # Verify - Kafka Topic List  **$** bin/kafka-topics.sh --list --zookeeper zookeeper1:2181 |

# Appendix D: Case Study

## Fault Tolerance: 3 Node Cluster - 1 Node Failure

### Gluu Cluster

In a 3 node cluster, 1 node failure does not affect the cluster, but had few seconds delay in website.

### Saas+Shop+Gluu Cluster

1 node failure does not affect the cluster

## Fault Tolerance: 3 Node Cluster - 2 Node Failure

### Gluu Cluster

1. In a 3 node cluster, 2 node failure is affecting the cluster.
2. Cluster is down
3. When the 2 nodes are up after failure, the cluster is automatically up. But sometimes not up automatically, need manual updating of services like oxauth, and oxtrust as ldap take time to load.
4. The cluster was completely down and once i need to completely recover consul and re-initiate it. Manual updated the services like oxauth, and oxtrust as ldap take time to load.

### Saas+Shop+Gluu Cluster

1. Cluster affected and down
2. When the 2 nodes are up after failure, the cluster is automatically up.
3. Galera: Was down only one replica is available as Non-primary (Readonly). Straight away the galera cluster was up when failed 2 nodes where up.

# Appendix E: DNS and Client Affinity

1. DNS is configured with Round-Robin of all Swarm Nodes.
2. Traffic is forwarded to port 443 of Nginx, exposed in all Swarm Nodes.
3. From Nginx to OxAuth, Oxtrust, OpenDJ, Shop, Shop Admin the traffic is forwarded with Client Affinity enabled, so next request from same client to these services will go to same container.
4. OxAuth, Oxtrust, OpenDJ, Shop, Shop Admin - These services will have multiple replica of containers. The [ip\_hash](http://nginx.org/en/docs/http/ngx_http_upstream_module.html#ip_hash) of nginx upstream setup is done here to maintain client affinity