horizontal line

Verte Manual Deployment

**31th Oct 2019**

|  |  |  |
| --- | --- | --- |
| **Version #** | **Date** | **By** |
| 1.0 | 31/10/2019 | Parthipan J |

# Table of Contents

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

[**Setup**](#_32fxcz5xc4sq) **4**

[Initial Requirement](#_42jkxnfpetp1) 4

[Directory Structure](#_83tz6abtlyrx) 4

[Install Docker CE](#_kbaxnjlv9kd9) 4

[**Single Node Deployment**](#_2u2vyfprg948) **6**

[Init Swarm](#_t7pijuik0imd) 6

[Load Image](#_ir4ihlemomdl) 6

[Manual Image Build](#_3hym6sy4wcs0) 6

[Deploy Single Node Stack](#_ndrz1kwksr1j) 7

[Modify Configs](#_ww1xo06ieupk) 7

[Modify Queue Service Config](#_eektjejoulna) 7

[Modify Chat/Admin Service Config](#_t6r8j4wxabek) 8

[Modify Stack File](#_r0g9rohveu32) 9

[Deploy Init Stack (Only Cassandra + RabbitMQ)](#_ak8kucte5p02) 9

[Deploy Node Stack (Chat + Admin + Queue)](#_vjw9dbpifyhs) 9

[Remove unwanted docker data](#_1c2ppg6w83mo) 9

[Remove unwanted docker images](#_vcf37pyf6kql) 10

[Remove or Re-deploy Stack](#_t6itcwdcu9j) 10

[Rollback Docker Service](#_i4n3rqme4yud) 10

[Essential Docker Commands](#_lbvnucflz6z7) 11

[**Appendix - A**](#_phddm8tjv5sc) **12**

[Node/Host Spec (Minimum)](#_s1ma7ynp2bhx) 12

[VM/Physical Host:](#_laapi3cvpcu6) 12

[Ports to open](#_brjpindqd6dz) 12

[Queue Service ENV](#_mayzv7ok0gbu) 12

[**Appendix B: Docker Stack File**](#_m9s4yd2l0dza) **13**

[Version](#_p8n0sj5lnw0b) 13

[Networks](#_bc1gmi4gul91) 13

[Creating the Overlay Network during stack deployment,](#_fo0j1sf12s1g) 14

[Adding Service to Overlay network](#_kl2o90v58ijz) 14

[Volumes](#_a40wa12t0blx) 14

[Attaching/Binding Volumes to Service](#_m0hci0liwqpp) 15

[Configs](#_n2g3svjrrn3i) 15

[Attaching Config to a Service](#_bigmm02kwrfa) 16

[Image](#_szqhr11eren7) 16

[Examples](#_1tci87um1hjd) 16

[Ports](#_2wujy9b3xd5w) 17

[Logging](#_3on5sw5e5bcb) 17

[Command](#_nk9fghqf42v) 18

[Deploy](#_goii8zjzrjjj) 18

[Replicated Mode](#_gwjjoqs4bcii) 18

[Replicas](#_cc3zzkapbq5o) 19

[Environment Variables](#_pldz7veycw6g) 19

[RabbitMQ](#_k23haluy0bk2) 19

[Chat](#_alr1nz4nfkh0) 19

[NODE ENV](#_se42iql5xm6a) 20

[Admin](#_65shpa94pag) 20

[NODE ENV](#_r39ayybhcc4x) 20

[Backup and Restore Cassandra (Physical)](#_7e0lzkn14308) 20

[Backup and Restore RabbitMQ (Physical)](#_6uzi575y55ga) 21

# 

# Setup

## Initial Requirement

Create a user with the *username*: **verte** and *password*: **verte123**

If it is multi node, we need to create the same username and password for all the VM and need to follow the below steps in **all the nodes**.

### 

### Directory Structure

The directory structure should be the following:

Home Directory : Should be **/home/verte**

Deployment Stack : You can download the stacks from mail attachments or from drive link shared in the mail and place it in **docker\_stacks**.

***Note: The downloaded directory should be renamed “docker\_stacks”***

Docker Images: You can download the images from mail attachments or from drive link shared in the mail and place it inside **/home/verte/docker\_images** directory.

|  |
| --- |
| **/home/verte /home/verte/docker\_stacks /home/verte/docker\_images** |

## Install Docker CE

Follow the below ***steps*** to install Docker CE in Ubuntu or follow the steps in this [link](https://docs.docker.com/install/linux/docker-ce/ubuntu/).

|  |
| --- |
| **# Update** sudo apt-**get** update  **# Install packages to allow apt to use a repository over HTTPS** sudo apt-**get** install \  apt-transport-https \  ca-certificates \  curl \  gnupg-agent \  Software-properties-common -y  **# Add Dockerâ€TMs official GPG key**  curl -fsSL https:*//download.docker.com/linux/ubuntu/gpg | sudo apt-key add -*  **# Add fingerprint** sudo apt-key fingerprint 0EBFCD88  **# Set up the stable repository** sudo add-apt-repository \  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \  $(lsb\_release -cs) \  stable"  **# Update again** sudo apt-**get** update  **# Install latest docker** sudo apt-**get** install docker-ce docker-ce-cli containerd.io |

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

# Single Node Deployment

### Init Swarm

|  |
| --- |
| **$** sudo docker swarm init |

### Load Image

If the images are shared via mail or drive link, follow the below steps to load the image to docker.

|  |
| --- |
| **$** cd /home/verte/docker\_images  **#** Load the image to docker **$** docker image load < vertechat.tar  **$** docker image load < verteadmin.tar **$** docker image load < vertequeue.tar  **#** List the docker images **$** docker images |

### Manual Image Build

This section is only required if the image is not provided or any configuration changes needed in the **chat** or **admin** application.

Follow the below steps to build an image with the latest code.

**Chat**

|  |
| --- |
| **$** cd /home/verte  **#** Clone and checkout to latest code. Ignore **if** it is already cloned **$** git clone https://<<USERNAME>>@bitbucket.org/Apptha/svep4103-verte-shlomimoyal-node-chatapi.git <<CHAT\_DIRECTORY>> **$** cd <<CHAT\_DIRECTORY>> **$** git checkout <<LATEST\_RELEASE\_BRANCH>>  **#** Change the config based on environment (Check Modify Chat/Admin Service Config section) **$** sudo docker build --tag vertechat:<<UNIQUE\_ORDERED\_NUMBER or TIMESTAMP>> |

**Admin**

|  |
| --- |
| **$** cd /home/verte  **#** Clone and checkout to latest code. Ignore **if** it is already cloned **$** git clone https://<<USERNAME>>@bitbucket.org/Apptha/svep4103-verte-shlomimoyal-node-webadminapi.git <<ADMIN\_DIRECTORY>> **$** cd <<ADMIN\_DIRECTORY>> **$** git checkout <<LATEST\_RELEASE\_BRANCH>>  **#** Change the config based on environment (Check Modify Chat/Admin Service Config section) **$** sudo docker build --tag verteadmin:<<UNIQUE\_ORDERED\_NUMBER or TIMESTAMP>> |

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

## Deploy Single Node Stack

### Modify Configs

1. All application configs are found inside **configs** (/home/verte/docker\_stacks/configs)folder.
2. Modify the env **file** to match the environment.
3. Replace <APP\_ENDPOINT> with your Host endpoint URL
   1. In case of single node system please use the host IP.

### Modify Queue Service Config

1. Update RabbitMQ URI, Username, and Password
2. Update Queue names for the Android and IOS
3. Update Android and IOS Push notification keys and file path

|  |
| --- |
| RABBITMQ\_AMQP\_URI=amqp://<APP\_ENDPOINT> RABBITMQ\_USERNAME=verte RABBITMQ\_PASSWORD=verte@123 RABBITMQ\_ANDROID\_QUEUES=android,android\_message,android\_call,android\_channel RABBITMQ\_IOS\_QUEUES=ios,ios\_message,ios\_call,ios\_channel FCM\_SERVER\_KEY=AAAAuSWi5o4:APA91bEluvp152KZAwDmmLWDfuKDD4inBJLsIRfdsIl52YVcbp1rhmFo\_4jmVAMLR26SZCJSGEf8X9Mtb4kDgzeJXl3wi33q3LWL719h794-io33heAy-U81ZF4fDuCqkpRAM-sUdnTV APN\_PFX=/usr/app/MirrorFly\_Voip\_Certificates.p12 APN\_PASSPHRASE=123456 |

### 

### Modify Chat/Admin Service Config

For the chat and admin service configs are build with the code. Below configs are available inside the build image

1. MySQL and Cassandra DB connection (Host, Port, DB Name, Username and Password)
2. Email properties
3. API Explorer configuration

So the below properties need to be updated in code and build again. And make sure to update the latest image in the stack.

|  |
| --- |
| db: {  port: <<MYSQL\_PORT>>,  Host: <<MYSQL\_HOST>>,  password: <<MYSQL\_PASSWORD>>,  username: <<MYSQL\_USERNAME>>,  database: <<MYSQL\_DATABASE>>  },  cassandra: {  host: <<CASSANDRA\_DATABASE>>,  port: <<CASSANDRA\_DATABASE>>,  keyspace: <<CASSANDRA\_DATABASE>>,  password: <<CASSANDRA\_DATABASE>>,  name: <<CASSANDRA\_DATABASE>>,  user: <<CASSANDRA\_DATABASE>>,  connectTimeout: 0,  readTimeout: 30000,  },  emailDataSource: {  name: "emailDataSource",  connector: "mail",  transports: [  {  type: "SMTP",  host: "smtp.gmail.com",  secure: true,  port: 465,  auth: {  user: <<MAIL\_USERNAME>>,  pass: <<MAIL\_PASSWORD>>  }  }  ]  } |

### Modify Stack File

1. Change the docker image reference to be deployed. Update the <<IMAGE\_NAME>> in stack file with the shared image name or your manual image name.
2. Change the replicas (if required)

### 

### Deploy Init Stack (Only Cassandra + RabbitMQ)

|  |
| --- |
| **$** cd /home/verte/docker\_stacks/ **$** sudo docker stack deploy --compose-file init.yml verte |

Warning: As the service is depend on the docker host, the host needs to be created first. If a service is trying to deploy before the host creation, deployment will fail. So if this happens we need to run the deploy command again.

### 

### Deploy Node Stack (Chat + Admin + Queue)

**Warning:** Make sure the Cassandra and RabbitMQ service is up and running before deploying the node stack.

|  |
| --- |
| **$** cd /home/verte/docker\_stacks/stacks/ **$** sudo docker stack deploy --compose-file node.yml verte |

**List the deployed services**

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

### 

## 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 init/node |

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

|  |
| --- |
| *# Filter the node volumes* $ sudo docker **volume** ls | grep "verte"  *# Remove volumes* $ sudo docker **volume** rm verte |

Redeploy Stack

|  |
| --- |
| *# Redeploy stack* $ sudo docker stack deploy -c init.yml/**node**.**yml** verte |

## Rollback Docker Service

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

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

## 

## 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 "init/node"  *# Filter the list of container* $ docker ps | grep "init/node"  *# 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 |

# 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 |

### Ports to open

|  |  |  |
| --- | --- | --- |
| **Port** | **Protocol** | **Destination** |
| 3000 | tcp | Admin Service |
| 3001 | tcp | Chat Service |
| 5672 | tcp | RabbitMQ Client |
| 15672 | tcp | RabbitMQ Management |
| 9042 | tcp | Cassandra |

## 

## Queue Service ENV

|  |  |  |
| --- | --- | --- |
| **ENV Variable** | **Default Value** | **Description** |
| RABBITMQ\_AMQP\_URI | amqp://<<HOST\_IP>> | Hosted server base url for internal url configuration |
| RABBITMQ\_USERNAME | verte | RabbitMQ access user name |
| RABBITMQ\_PASSWORD | verte@123 | RabbitMQ access password |
| RABBITMQ\_ANDROID\_QUEUES | android,android\_message,android\_call,android\_channel | Android related queues |
| RABBITMQ\_IOS\_QUEUES | ios,ios\_message,ios\_call,ios\_channel | IOS related queues |
| FCM\_SERVER\_KEY | AAAAuSWi5o4:APA91bEluvp152KZAwDmmLWDfuKDD4inBJLsIRfdsIl52YVcbp1rhmFo\_4jmVAMLR26SZCJSGEf8X9Mtb4kDgzeJXl3wi33q3LWL719h794-io33heAy-U81ZF4fDuCqkpRAM-sUdnTV | FCM server key for push notification |
| APN\_PFX | <<Base\_Path>>/<<certificate\_file\_name>> | IOS push notification configuration file |
| APN\_PASSPHRASE | 123456 | IOS push notification pass |

# 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 with 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: "3.7"** |

## Networks

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

### Creating the Overlay Network during stack deployment,

|  |
| --- |
| networks:  host: |

### Adding Service to Overlay network

|  |
| --- |
| **services:  admin:  networks:  - host** |

## 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:  cassandra:  rabbitmq:  storage: |

### Attaching/Binding Volumes to Service

|  |
| --- |
| **services:  cassandra:  Image: <<IMAGE>>  networks:  - host  volumes:  - type: volume  source: cassandra  target: /var/lib/cassandra  - "./init/cassandra-init.sh:/cassandra-init.sh"  - "./config/cassandra.yaml:/etc/cassandra/cassandra.yaml"** |

* 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*.

## 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:  queue-**en**v:  **file**: ./config/.env |

* ***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:  queue:  image: <<Image>>  networks:  - host  configs:  - source: queue-env  target: /usr/app/.env** |

* ***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: <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:  chat:  image: <<Image\_Name>>** |

### 

### Examples

|  |
| --- |
| image: rabbitmq image: chat |

## 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: rabbitmq  networks:  - host  ports:  - target: 5672  published: 5672  protocol: tcp  mode: host  - target: 15672  published: 15672  protocol: tcp  mode: host** |

## Logging

Logging configuration for the service.

|  |
| --- |
| logging:  driver: json-**file**  options:  max-size: "10m"  max-**file**: "10" |

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:  cassandra:  image: <<Image>>  logging:  driver: "json-file"  options:  max-size: "10m"  max-file: "10"** |

For Logging Options check here: <https://docs.docker.com/config/containers/logging/json-file/>

## Command

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

|  |
| --- |
| **services:  cassandra:  Image: <<Image>>  command: "sh /cassandra-init.sh"** |

## 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/)

### 

### Replicated Mode

|  |
| --- |
| services:  cassandra:  image: cassandra:3.11.4  deploy:  replicas: 1  restart\_policy:  condition: on-failure  delay: 5s  max\_attempts: 3  window: 120s |

### 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).

## Environment Variables

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

### RabbitMQ

Details: <https://hub.docker.com/_/rabbitmq>

|  |
| --- |
| **services:  rabbitmq:  image: <<Image>>  environment:  RABBITMQ\_DEFAULT\_USER: verte  RABBITMQ\_DEFAULT\_PASS: verte@123** |

### Chat

|  |
| --- |
| **chat:  image:<<Image>>  environment:  NODE\_ENV: prod** |

#### NODE ENV

The NODE\_ENV environment variable is to point the chat service to the particular config and datasource file inside chat application.

**Note:** If the NODE\_ENV value is prod, then server/config.prod.js and server/datasources.prod.js files must exist inside the application.

### Admin

|  |
| --- |
| **admin:  image:<<Image>>  environment:  NODE\_ENV: prod** |

#### NODE ENV

The NODE\_ENV environment variable is to point the admin service to the particular config and datasource file inside admin application.

**Note:** If the NODE\_ENV value is prod, then server/config.prod.js and server/datasources.prod.js files must exist inside the application.

## Backup and Restore Cassandra (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 **cassandra** before copying the files. Otherwise, you will copy the files in an invalid state.

Here in our architecture we have 1 cassandra db replication docker services. Each service will be attached with a docker volume verte\_cassandra, located in accordingly in each node respectively.

We should take a backup of these 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/verte\_cassandra/\_data. How to find this path/mount point ?

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

Take a backup of this data,

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

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

**Restore backup**

|  |
| --- |
| *# Remove init completely* $ docker stack rm init  *# Remove volumes data in each node, verte\_cassandra* $ rm -rf /var/**lib**/**docker**/**volumes**/**verte\_cassandra**/**\_data**/\*  *# Copy the backup data* $ cp verte\_cassandra\_<Date>.tar.gz /var/**lib**/**docker**/**volumes**/**verte\_cassandra**/**\_data**/   $ gunzip verte\_cassandra\_<Date>.tar.gz $ tar -zxvf verte\_cassandra\_<Date>.tar  *# Check data is available in \_data folder* $ ls -l /var/**lib**/**docker**/**volumes**/**verte\_cassandra**/**\_data**/  *# Now start or deploy init again* $ cd /home/verte/docker\_stacks/stacks $ sudo docker stack deploy --compose-file init.yml verte |

## Backup and Restore RabbitMQ (Physical)

Like Cassandra, you must stop all writes to the **RabbitMQ** before copying the files. Otherwise, you will copy the files in an invalid state.

Here in our architecture we have 1 rabbitmq replication docker services. Each service will be attached with a docker volume verte\_rabbitmq, located in accordingly in each node respectively.

We should take a backup of these 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/verte\_rabbitmq/\_data. How to find this path/mount point ?

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

Take a backup of this data,

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

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

**Restore backup**

|  |
| --- |
| *# Remove init completely* $ docker stack rm init  *# Remove volumes data in each node, verte\_rabbitmq* $ rm -rf /var/**lib**/**docker**/**volumes**/**verte\_rabbitmq**/**\_data**/\*  *# Copy the backup data* $ cp verte\_rabbitmq\_<Date>.tar.gz /var/**lib**/**docker**/**volumes**/**verte\_rabbitmq**/**\_data**/   $ gunzip verte\_rabbitmq\_<Date>.tar.gz $ tar -zxvf verte\_rabbitmq\_<Date>.tar  *# Check data is available in \_data folder* $ ls -l /var/**lib**/**docker**/**volumes**/**verte\_rabbitmq**/**\_data**/  *# Now start or deploy init again* $ cd /home/verte/docker\_stacks/stacks $ sudo docker stack deploy --compose-file init.yml verte |