## Deploying Kaltura Clusters Using docker containers

Below are instructions for deploying Kaltura Clusters using docker we used centos 6(RPM) as OS but could be changed in the future.

### Before You Get Started Notes

* The docker principle states that containers are easy to build and that the image is already ready to deploy, in Kaltura we would like to achieve that but it is not possible as of now, since the Kaltura installation procedure requires shared nfs and connection between the containers, in the future we will try to minimize the procedure.
* All post-run scripts accept answers-file automatically as parameter, this is used for silent-automatic installs. If you want please edit the file prior to the installation and then install
* For Load Balancing we used Nginx with round robin
* The LB is expecting two front end machines with specific host names, but this could be edited to whatever names you want
* The machine host-names must the installation files host names, otherwise they will not work
* All answer files must match to each other (in the future they should be on a single docker volume)
* The Batch scripts simply follow the guide from github:

[Deploying Kaltura Clusters](https://github.com/kaltura/platform-install-packages/blob/Mercury-13.14.0/doc/rpm-cluster-deployment-instructions.md#backup-and-restore-practices)

Combined with the Kaltura docker installation instruction:

[Installing Kaltura Docker container](https://github.com/kaltura/platform-install-packages/blob/Mercury-13.14.0/doc/install-docker.md)

* Currently we use the docker link to connect the clusters , we are aware that this is legacy and we will move to docker network at a future time
* Docker documentation can be found in the docker website and in google but I am providing some important points:
* Docker images are base state of a ready to deploy machine,
  + Images can be created via docker build command
  + Images can be pooled from docker hub
* Containers are running states of images:
  + The creation of container from an image is done via the ‘docker run’ command
* The following flags will be used in the run command:
  + -h 🡪 the host name must be the same as in the answer file
  + -d 🡪 detach flag, if this flag is not shown the current session will be locked to the container meaning you will have to stay in the container forever
  + --privileged 🡪 gives additional privileges to the containers (allows writing to the NFS)
  + --name 🡪 the container name will use us to link containers
  + –link <container> links the container to another running container (via host file)
    - Note that link is detracted, and we will migrate to network in the future
* Docker exec is running a command on a running container you will see the following flag
  + -it 🡪 interactive mode allows you to see output of command
    - Example: running ‘docker exec -it mysql bash‘ will get you into the container bash

## Pre-installation requirements

Installed [Docker Engine](https://docs.docker.com/engine/installation) on a valid host(atomic-OS is recommended )

* <https://www.projectatomic.io/>
* If you would like to install widget on the atomic please use:
  + rpm-ostree install vim
* also, there are two version of docker on the atomic, we recommend using the latest, this can be achieved via:
  + <https://access.redhat.com/articles/2317361>

## Step-by-step Installation

### The NFS

Run command:

* docker run -d --storage-opt size=120G --net=bridge --privileged --name <container name> -h <host name> <image name:tag>
  + example: docker run -d --storage-opt size=120G --net=bridge --privileged --name kalt-nfs -h kalt-nfs roiebeck/nfs:latest

Notes:

* All containers must run with docker ‘--privileged’ flag in order to connect to NFS
* --storage-opt size=120G give the NFS container root partition 120 GB of space.

### MY-SQL DB Server

Run command:

* docker run –name <container name > -d -h <host name> <image name:tag>
  + Example: docker run --name mysql -d -h docker-mysql-host roiebeck/kaltura-mysql:version4

### Sphinx Indexing Server

Pre-requirement

* MySQL instance up and running

Run command:

* docker run -d --link <mysql container name> --name <container name>-h <sphinx-host--name> <image name: version>
  + Example: docker run -d --link mysql --name sphinx1 -h docker-sphinx1-host roiebeck/kaltura-sphinx:version1

then execute the following command:

* docker exec -it <name of sphinx machine> /root/install/sphinx\_settings.sh
  + Example: docker exec -it sphinx1 /root/install/sphinx\_settings.sh

### The first Front node

Pre-requirement

* MySQL container up and running
* Sphinx container up and running

Run command:

* docker run --name <name > -h <host name> -d --link <MySQL container name > --link <sphinx container name> --link <NFS container name> --privileged <Image name>
  + Example: docker run --name fefirstnode -h first-front-host -d --link mysql --link sphinx1 --link kaltura-nfs --privileged roiebeck/kaltura-fe:version3

Execute command:

* docker exec -it <container name> /root/install/fe\_settings.sh
  + example: docker exec -it fefirstnode /root/install/fe\_settings.sh
* execute command:

docker exec -it <container name> /root/install/fe\_db\_settings.sh

* + example: docker exec -it fefirstnode /root/install/fe\_db\_settings.sh

### Additional Front nodes

Note: you can install as many FE additional as you want, but you will need to configure them in the LB

Pre-requirement

* MySQL container up and running
* Sphinx container up and running

Run command:

* docker run --name <container name> -h <host name> -d --link <MySQL container name> --link sphinx container name> --link <NFS container name> --privileged <Image name>
  + docker run --name fesecondnode -h second-front-host -d --link mysql --link sphinx1 --link kaltura-nfs --privileged roiebeck/kaltura-fe-additional-nodes:version1

Execute command:

* docker exec -it <Container name> /root/install/fe\_settings.sh
  + example: docker exec -it fesecondnode /root/install/fe\_settings.sh

### Nginx LB Node

Note:

* if you add additional FE nodes you will need to edit the /etc/nginx/conf.d/default.conf file in the container and restart the nginx service

Link: [restart nginx service](http://blog.tobiasforkel.de/en/2016/08/18/reload-nginx-inside-docker-container/)

Run Command:

* docker run -d --link <FE 1 container Name>….--link <FE n container Name> --name=docker-lb -h docker-host -p 80:80 -p 443:443 -p 1935:1935 <image name>
  + example: docker run -d --link fefirstnode --link fesecondnode --name=docker-lb -h docker-host -p 80:80 -p 443:443 -p 1935:1935 roiebeck/load-balancer-nginx:version1

### Batch Node

Run Command:

* docker run --name <node name > -h <host name> -d --link <lb container name> --link <mysql name> --link <sphinx name> --link <nfs name> --privileged <image name>
  + example: docker run --name batchfirstnode -h first-batch-host -d --link docker-lb --link mysql --link sphinx1 --link kaltura-nfs --privileged roiebeck/kaltura-batch:version1

Execute Command:

* docker exec -it <batch container name> /root/install/batch\_settings.sh
  + example: docker exec -it batchfirstnode /root/install/batch\_settings.sh

### Nginx VOD Node

Run Command:

* docker run --name <container name > -h <host name> -p 88:88 -d --link <lb container name> --link <MySQL container name> --link <sphinx container name> --link <NFS container name> --privileged <Image Name>
  + example: docker run --name nginx\_vod -h docker-packager -p 88:88 -d --link docker-lb --link mysql --link sphinx1 --link kaltura-nfs --privileged roiebeck/kaltura-nginx-vod:version1

Execute Command:

* docker exec -it <container name> /root/install/nginx\_settings.sh
  + example: docker exec -it nginx\_vod /root/install/nginx\_settings.sh

### DWH Node

Run Command:

* docker run --name <container name > -h <host name -d --link <lb container name> --link <MySQL container name> --link <sphinx container name> --link <NFS container name> --privileged <Image Name>
  + docker run --name dwh -h dwh-host -d --link docker-lb --link mysql --link sphinx1 --link kaltura-nfs --privileged roiebeck/kaltura-dwh:version1

Execute Command:

* docker exec -it <container name> /root/install/dwh\_settings.sh
  + example: docker exec -it dwh /root/install/dwh\_settings.sh