CLOUD INFRASTRUCTURE - LAB 11: DOCKER SWARM

In this two-week lab you gonna learn how to use Docker Swarm and how it can help you to scale your Docker setup to a whole other level. Your report of this lab will be rated. Work in teams of 2.

PREPARATION

It is required for this lab that you are able to run up to 6 Docker instances at once - each with about 1GB RAM. If your notebook doesn't have at least 8GB RAM, please consider running this lab on a 2.103 HSR workstation (16GB RAM) or on a cloud provider of your choice (Google Cloud, Amazon, Azure). The most clouds offer some free credit to get started:

- Azure: 200$ free credit: https://azure.microsoft.com/en-us/free/
- Google Cloud: 300$ free credit: https://cloud.google.com/free/

Important: Check the VM limit of your cloud provider. E.g. DigitalOcean does not allow you to spin up more than 5 VMs (or "Droplets" how they call it) by default.

To simplify the process we recommend using docker-machine to set up the Docker Swarm nodes. Docker Machine is a tool to provision and manage hosts with Docker Engine on them. Typically, you install Docker Machine on your local system to manage remote hosts. Check https://docs.docker.com/machine/ or https://github.com/docker/docker.github.io/blob/master/machine/AVAILABLE_DRIVER_PLUGINS.md to see the supported providers (VirtualBox, VMware Workstation, Parallels, Azure, DigitalOcean, etc.). You should have already docker-machine installed on your notebook back from the Cloud Infrastructure lab 5.

TASKS / REQUIREMENTS

SUB-TASK 1

Set up a redundant Docker Swarm environment:

- The cluster must have 6 nodes (3 masters and 3 workers).
- Document the required steps and describe each with a few sentences (What does this command do? Whats the result from it? Why is it needed?)

SUB-TASK 2

In the CloudInf lab 5 you dockerized a three-tier application. Read the full Sub-Task instructions before you start to document the requirements or change anything in the docker-compose.yml file. The goal of this sub-task is to extent and enhance your existing docker-compose.yml file of this application to be runnable on Docker Swarm. Use https://docs.docker.com/compose/compose-file/ to see the documentation of Docker Swarm deployment related Dockerfile statements.

It must address the following minimal requirements:

- There should be 3 instances (replicas) of the web and app tier running (each).
- A single db instance is sufficient for the moment. We don't want to set up a whole db cluster.
- If any replica crashes, a new one should be started automatically.
- The three tier application must be deployed as a replicated service and non of the replicas should run on a master node (masters are also workers by default).
- Define a solid and working service update configuration for all services. Describe why you have chosen each value and how the service should be updated in case of a new release.
- Because we only run a single db instance, we would like to monitor its health via a separate healthcheck. Implement one and explain it.
- Security should be considered. Make sure the web and app tier service replicas are running with a service user and not with root. Access a running replica of a service and use "`ps aux`" so see under which user the processes are running inside the contianer. Deliver screenshots in your report in order to prove that your images are properly secured.

Think of additional requirements for this application. To what else should be taken care of for this specific 3 tier application when its deployed on a Docker Swarm cluster?

Make sure your Docker service is deployed and running on the cluster - try to access the application via http://<docker-master-or-worker-ip>:8080. Add the service inspection JSON output of all your services to the documentation appendix (command: "`docker service inspect <your-service-name-here> --pretty`").

Hint: Docker images can't be build on the Docker Swarm cluster itself, any "build" statement inside your docker-compose.yml file will be ignored during the deployment of the service stack. This results in the need to build these images on your notebook and push them to a registry. You can simply use Docker Hub for this task or deploy and use a local registry. See https://docs.docker.com/registry/deploying/ for instructions on how to set up a local registry.

Important: Make sure the local registry is reachable from all Docker Swarm nodes in order to be able to deploy service replicas on all nodes! If you see "0/3" replicas are active then there's a good chance that some worker nodes are not able to pull the image defined by the service. However, wait a few seconds up to 3 minutes after the service start before you try to trouble shoot this issue. The worker nodes still needs some time to initionally pull the Docker images from the registry.

SUB-TASK 3

Once your three tier application is up and running, you should continue to analyse how Docker Swarm works by answering the following questions. Provide screenshots and console output in the documentation.

Application failure:

1. Open another shell and make sure the Docker CLI is connected to a master by using the docker-machine environment variable export feature (command: "`eval $(docker-machine env master2)`"). Test the connection to the Swarm using the "`docker node ls`" command on your notebook/system.
2. Locate the worker node which currently hosts the postgres replica. SSH to this worker node and kill the postgres container (command: "`docker kill <container-id>`").
3. On the newly opened shell, check how long it takes Docker Swarm to recreate the postgres replica by running "`docker service ls`" or "`docker service ps <postgres-service-name-or-id>`". How many seconds passed to restore the postgres service?

Leader/master failure:

1. Check to which master node your shell is currently connected to by using the command "`env | grep DOCKER_HOST`".
2. Stop the other two master VMs (command: "`docker-machine stop <masterX-vm>`").
3. Are you still able to view the services using the Docker CLI?
4. Try to scale your web service using "`docker service scale <service-name>=4`". Does Docker Swarm still accept this change? If so or if not, why?
5. Is your service still accessible via HTTP?
6. Restart the previously halted Docker Swarm masters (command: "`docker-machine start <masterX-vm>`") to get a clean base for the next task.

SUB-TASK 4

It would be helpful to provide the development team a self-service portal which allows them to deploy and manage their Docker services on the Swarm cluster by them self. Because some of them have never used a CLI, the self-serice portal needs to have a web UI. Install such a solution and briefly document the required steps. Provide a short overview of your solution and explain why you have chosen it.

Recently during the testing phase you realized that some services have crashed in the past and nobody noticed anything. Because of this, it is now up to you to find an accuarate Docker supporting monitoring solution which gives a nice overview of the Docker Swarm health and its services. Set up a microservice monitoring solution and briefly describe how its done, how the solution works and why you have chosen this product.

Note: These two requirements above may be also fulfilled with a single solution. This doesn't matter and its up to you, if you would like to deploy two single solutions or a solution which handles both.

SUB-TASK 5

Lately some IT team leaders decided to also host some public available services on the same Docker Swarm environment. Because you are a Docker Swarm expert, they asked you what the key points are to make such a change possible.

Important: This sub-task does not need to be implemented on your lab setup because of environment limitations. You only have to deliver a briefly write-up and a map, which shows how the requirements can be fulfilled.

The minimal requirements are:

- Some services like the web tier of your three-tier application need to be available from the internet. Research and find the best Docker Swarm networking setup for this purpose.
- The external reachable services need to be hardened and separated at network level. Describe the required steps and how you would protect company internal services from beeing accessed from the internet.
- Services need to be high available.
- Your web service currently only runs with HTTP. To publish the service on the internet, it is required to publish it via HTTPS. Its not possible to change the web service application itself to HTTPS because of further internal dependencies.
- The public services need to be reachable via URL.

Keep in mind that there are multiple possible solutions. The map should show how the public services are reachable from the WWW and also include networks and IPs. Do not limit yourself to the minimal requirements provided above, also think of other requirements which should be taken care of.

DELIVERY OF YOUR REPORT

You will deliver a structured report which will answer the questions from above and explain your deployment strategy. Its also required to deliver your Docker Compose file. Date of delivery at latest is **19.12.2017, 10:00**, as .zip file to abgabe@ins.hsr.ch.

All used external resources must be marked as such. If you used external resources, make sure to add a bibliography to your document. Copied work will be marked with 1.