Cloud Computing

Summer Term 2017

{Anton.Gulenko | Marcel.Wallschlaeger}@tu-berlin.de

Project Assignment No. 3

Due: 16.7.2017 23:59

This assignment complements your knowledge of working with IaaS platforms with container-based virtualization. Containers are often used inside of virtual machines to exploit the benefits of both virtualization solutions.

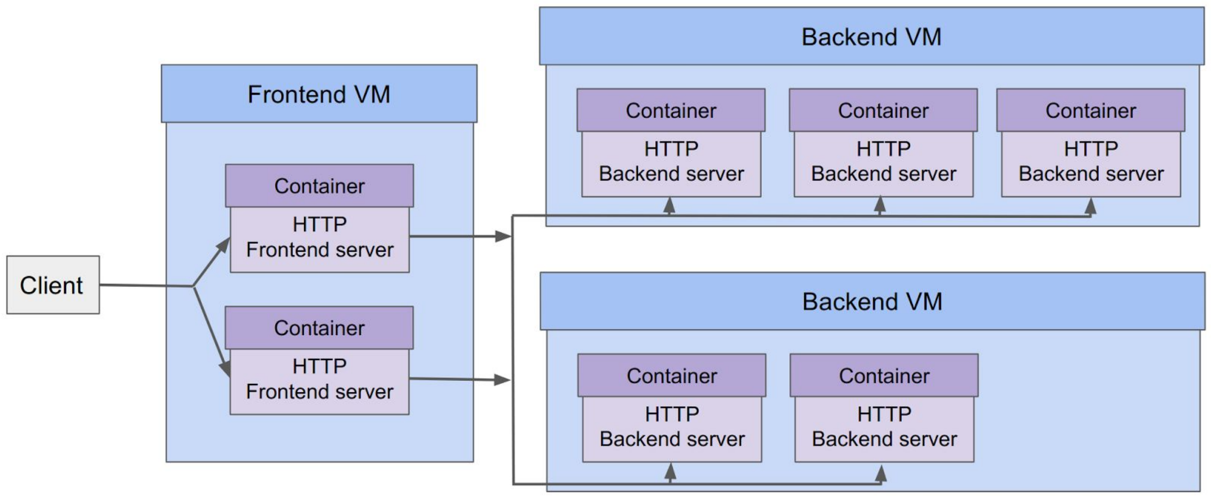
1. Preparation

- Download and unpack the file **assignment3-resources.zip** from ISIS
- Your main task is to complete the missing parts of some files in the archive
- Do not add or remove any files, only modify some of the existing files (see below)
- Parts of files that you have to modify or complete are marked with **[[TODO]]**, so you can search for that string to see if you completed all missing parts
- Your submission will be a zip archive named **cc17-assignment3-groupXX.zip** (replace XX with your group number, e.g. 05 or 14)
- Your submission must contain exactly the same directory structure and file names as the original archive, but some of the files must be modified by you
- Your submissions will be partially graded and tested by automated scripts, so file names etc. are important
- You do not have to keep any command line listings as in the previous assignments

Contents of assignment3-resources.zip

```
├──── Backend
│    ├──── docker-compose.yml  ← Must be completed
│    ├──── Dockerfile                  ← Must be completed
│    ├──── fix-nginx-conf.sh
│    └──── nginx.conf
├──── Frontend
│    ├──── docker-compose.yml  ← Must be completed
│    ├──── Dockerfile                  ← Must be completed
│    ├──── fix-nginx-conf.sh
│    └──── nginx.conf
├──── Scripts
│    ├──── openstack-cleanup.sh
│    └──── test-deployment.py
├──── create-stack.sh           ← Must be completed
├──── init-swarm.sh                    ← Must be completed
├──── server-landscape.yaml
└──── server.yaml
```

Target deployment
In this assignment you will use the virtual infrastructure created by the server-landscape.yaml Heat template to host services running in Docker containers managed by Docker Swarm. The target deployment is shown in the following diagram. The frontend VM will host a two-fold replicated frontend HTTP service, while the two backend VMs will host a five-fold replicated backend HTTP service. The frontend service will be exposed to clients and load balanced by a Docker networking feature. The frontend services will forward incoming requests to one of the backend VMs using a hash-based load balancing strategy. Inside the backend VMs, Docker networking will again load balance the incoming requests between the difference backend service instances.
The HTTP services themselves are simple nginx services configured to attach predefined HTTP headers to incoming requests. There is no real HTML content served. You can read the nginx.conf files in the provided resources to see what the nginx servers will be configured to do.



2. Prepare Docker Images

- Get familiar with Docker
- Read the server-landscape.yaml Heat template and complete the **create-stack.sh** script by adding an openstack command that instantiates the template with all necessary parmeters. Use the **--wait** parameter to make the CLI wait until the stack is fully created.
- Use the create-stack.sh script to instantiate the Heat template. All 3 VMs will run an up-to-date version of Docker.
- Copy the folders **Frontend** and **Backend** to one of the VMs and log into that VM.
- Develop two **Dockerfiles**, one for the frontend container and one for the backend container, with the following requirements:
    - Both Dockerfiles must be based on the official nginx container image
    - They must expose port 80
    - The file **nginx.conf** must be available inside the container as **/etc/nginx/nginx.conf**
    - The file **fix-nginx-conf.sh** must be available in the container in an arbitrary directory
        - Note: the frontend and backend containers have different versions of nginx.conf and fix-nginx-conf.sh
    - Define a volume named **/hypervisor_etc**

    - When starting up, the containers must first execute the **fix-nginx-conf.sh** script and then run **nginx** without parameters.
        - Note: the **fix-nginx-conf.sh** of the frontend container needs a parameter, which will be available as an environment variable called **CC_BACKEND_SERVERS**
- Test your Dockerfiles by building them locally and running them. Make sure to map the port 80 of each container to a port of the VM, otherwise you will not be able to reach the nginx service inside the container. For the frontend server, define the environment variable **CC_BACKEND_SERVERS**. Read the two **nginx.conf** files, reconstruct what he frontend and backend servers are doing, and figure out how you can test your services using the **curl** command. This is not part of your submission, but it will help you debug your containers.
- Upload your working containers into a publicly available Docker registry. You have two choices here:
    - Either register on Dockerhub and upload your images publicly
    - Or use the TU Gitlab, create a public repository and use the Gitlab Docker Registry integration
- Do not delete your uploaded container images, as they are part of your submission and you will need them later.
- Make sure to put the **Dockerfiles** into your submission archive.

3. Prepare the Docker swarm

Complete the script **init-swarm.sh**. It has multiple places marked **[[TODO]]** indicating missing parts. Read the comments in the script and the rest of the code to understand what the missing parts should be doing.
Hints:

- Line 12: use the command **jq -r @tsv** to convert a JSON array to a list of plain values.
- Line 15: for now, this will copy the incomplete docker-compose.yml files, but they will become useful later. You can also skip this part for now, but you will have to complete it later.
- Lines 49, 53: These lines are commented out for now, you will complete them later.

Execute the completed **init-swarm.sh** script. It should prepare your VMs for proper Docker swarm operation.

4. Prepare the Docker stacks

- Get familiar with Docker Compose files (Docker Compose Rerence)
- Complete the two **docker-compose.yml** files in the Backend and Frontend directories, see below for the requirements. You can develop the files on the frontend VM and copy them into your submission archive afterwards. You can use the **docker-compose** command to test the files (or alternatively the **docker stack** command family).

Requirements:

- Both docker-compose.yml files:
    - Mount the **/etc** directory of the VM into the container as **/hypervisor_etc**. This matches the volume that you defined in the Dockerfiles. Note: This would usually not be done, as it exposes a lot of

information about the Docker host to the container. In this case, it is done to make your setup well testable.

- Frontend/docker-compose.yml:
  - Map the Container port 80 to port 80 of the VM
  - As image, use the frontend Docker image that you uploaded into a public Docker registry earlier.
  - The service must be replicated two-fold and the containers must automatically restart upon failure.
  - Add a placement constraint that ensures that both replicas are launched on the frontend VM. Use the hostname of the frontend VM (which is [STACK-NAME]-frontend) as constraint filter.
  - Add an environment variable **CC_BACKEND_SERVERS** and set it to the value **${CC_BACKEND_SERVERS}**. This might seem redundant, but it declares that the named variable will be forwarded into the created container instances.
- Backend/docker-compose.yml:
  - Map the Container port 80 to port 8000 of the VM
  - As image, use the backend Docker image that you uploaded into a public Docker registry earlier.
  - The service must be replicated five-fold and the containers must automatically restart upon failure.
  - Add a placement constraint that ensures that the five replicas are launched only on the backend VMs. Use the hostname of the frontend VM (which is [STACK-NAME]-frontend) as negative constraint filter.

5. Launch the Docker stacks

- In the **init-swarm.sh** script, comment in line 49 and 53 (the line numbers might have changed after you completed the script earlier) and complete the two lines. They should launch Docker stacks based on the two docker-compose.yml files copied by the script above.
- Re-execute the init-swarm.sh script. It might print warnings or errors because the Docker swarm is already initialized, but it will also spin up the two new stacks.
- Test your deployment with the **Scripts/test-deployment.py** script. Launch it and pass it the floating IP of your frontend VM as parmeter. Your grading will depend on whether your deployment passes the tests of this script. After submitting your solution, let the deployment run and do not delete it.

6. Submission Deliverables

Your submission on ISIS must be a single file following the following criteria:

- Submission has **.zip** file format, the same directory and file structure as the original archive, and is named **cc17-assignment3-groupXX.zip** *(3 points)*
- The **create-stack.sh** script is completed (3 points)
- The two Dockerfiles are completed (5 points each)
- The **init-swarm.sh** script is completed (5 points)
- The two **docker-compose.yml** files are completed (5 points each)
- The Docker images used in the docker-compose.yml files must be available and can be executed (3 points each)
- The entire setup is running in the OpenStack cloud and passes the test of the test-deployment.py script (10 points)

Total points: 46