Cloud Computing

Winter Term 2020/2021

Practical Assignment No. 2

Due: 06.12.2020 11:55 pm

The primary goal of this assignment is to gain insight into a cloud computing platform from a provider perspective. Building upon gained experience from practical assignment no. 1, you will set up you own OpenStack cloud computing platform, configure the network, perform functionality checks, execute the benchmarks from the previous assignment and compare the results.

Prerequisites

This assignment consists of several steps. Work with checkpoints. If you reach a working status, wrap everything up to a script and verify that the outcome is reproducible. Move on to the next step after that. It will help you to not get lost around all required tools and systems.
Being stuck at some point, a good strategy is to clean everything, run the deployment until a know and working checkpoint and start again from there.
It is also important to shut down your VMs when you do not need them. Checkpoints will help to get to you previous working state.

Linux OS

Linux is required. If you use, Mac OS or Windows privately, work in a virtual machine (either locally, such as VirtualBox, or use one of the Cloud platforms). This task was tested with Ubuntu and Debian. Therefore, we recommend using one of those.

Google Cloud

Refresh your knowledge about the google cloud platform from practical assignment no. 1

1. Virtual Machines

Write a well-commented Shell script that prepares a virtual environment on the gc platform. It will be used to deploy OpenStack.

- You can first manually experiment with the `gcloud` commands and make notes, before finally combining the commands into a script.
- Each command in the scripts should be commented, make sure the scripts are clean and not cluttered with unnecessary text.
- After writing the script, test it: delete all running disks, images, VMs, networks and firewall rules, run the script, and check that the result is satisfactory.

- Do **not** include <u>any private information</u> such as your access key or secret key in your submission (if necessary, replace them with dummy strings).
- All requirements listed below should be covered by at least one command.
- The key from practical assignment no. 1 can be used. We therefore skip the key creation here.

Requirements:

1. Create two additional VPC networks "cc-network1" and "cc-network2" with subnet-mode "custom".
2. For each created network, create a respective subnet "cc-subnet1" and "cc-subnet2" and assign different IP ranges to them. Furthermore, "cc-subnet1" needs a secondary range (check the --secondary-range parameter).
3. Read up about nested virtualization on the gc platform: https://cloud.google.com/compute/docs/instances/enable-nested-virtualization-vm-instances?hl=en
4. Create a disk based on an "Ubuntu Server 18.04" image in your default zone and set the size to at least 100GB.
5. Use the disk to create a custom image and include the required license for nested virtualization.
6. Start 3 VMs that allow nested virtualization:
   - Name them "controller", "compute1", "compute2" (Hint: Start with 1 VM and verify everything below before extending it to 3).
   - The image should be the previously created custom image that supports nested virtualization.
   - Set the tag "cc" for each VM.
   - Choose machine type "n2-standard-2"
   - The VMs must have 2 NICs (check the --network-interface parameter). The first NIC should be connected to cc-subnet1, the second NIC should be connected to cc-subnet2, the alias IPs from the secondary range should be assigned to first NIC of the **controller** VM (only the controller VM!).
7. Create a firewall rule that allows all tcp, icmp and udp traffic for the IP ranges of cc-subnet1 and cc-subnet2. Restrict it to VMs that have the "cc"-tag.
8. You can read up on all required ports of OpenStack: https://docs.openstack.org/install-guide/firewalls-default-ports.html
   You must open them to be accessible from any external IP address. For simplicity, you can create a firewall rule, that allows all tcp and icmp traffic from external IPs for cc-network1 (Caution: This is OK for this assignment, in production you need to be careful opening up your environment to the internet.). Restrict it to VMs that have the "cc"-tag.

Verify:

- These verifications must not be part of the submission but helps to verify that the environment is ready for OpenStack deployment.
- Check nested virtualization support. Each VM should have VMX enabled. Execute: `grep -cw vmx /proc/cpuinfo`
  The response indicates the number of CPU cores that are available for nested virtualization. A non-zero return value means that nested virtualization is available. If you get 0, you need to revisit step 3.

- Each instance should have 2 NICs and 2 internal IP addresses. Check via gc dashboard, gc command line or run `ifconfig` or `ip link show` within the VMs.
- ssh to all VMs must work:
  `ssh -i /path/to/id_rsa <user>@<public_ip>`
  If it is not working, check you firewall rules (especially step 8).
- Connect via ssh to the VMs and try to ping both internal IP addresses of the other 2 VMs. If it does not work, check firewall rules (step 7).
- Connect via ssh to instances and test tcp traffic to internal IP addresses of the other 2 instances: `nc -z -v <internal ip adress> 22`
  It must be successful. If it does not work, check firewall rules (step 7).

Outputs:

- Script **spinup-gcp-for-openstack.sh**
    - Containing all commands for preparing and starting your VMs, including comments explaining what the commands do.

2. OpenStack Setup

We will use Ansible and kola-ansible to deploy OpenStack on the 3 previously created virtual machines. The steps below are mostly from the official kolla-ansible guide (https://docs.openstack.org/kolla-ansible/latest/user/quickstart.html) but include some additional comments to catch frequent pitfalls.
The output of each relevant step is piped into an output log file via `tee`. You will need to submit these output files.

General requirements:

1. You must have a non-root user with sudo rights on your local machine (Or on the machine that you use to run these scripts from, e.g. another VM. Hereinafter, we will use the term "local machine" although it might not be your real local machine). It will make your life easier if a password is not requested when running commands with sudo.
2. Check the kolla-ansible quick start guide and install required kolla-ansible dependencies for your local machine.
3. Create and activate a python3 virtual environment on your local machine: https://docs.python.org/3/tutorial/venv.html
4. Use pip to install ansible and kolla-ansible.

Prepare kolla-ansible:

1. Download the provided kolla-ansible archive from ISIS and unpack into you working directory.
2. Open the files "multinode" and "globals.yml". Replace the placeholders "[******<>******]" with respective values.
   Example:
   ```
   kolla_internal_vip_address: "10.34.122.100"
   ansible_user=alex
   ```

Install OpenStack:
We prepared a script `deploy-openstack.sh`. However, it is recommended to execute the commands manually one after another and verify that they are executed without errors. Otherwise debugging will be very hard.

1. `cd` into your working directory.
2. Run `ansible -m ping all -i ./multinode` to verify that ansible is setup correctly and the VMs are reachable. In case of failure:
   - Check correct installation of ansible
   - Revisit **Prepare kolla-ansible** and check if the placeholders were replaced correctly.
   - Check again if you can connect via ssh to the gc VMs.
3. Run the commands from `deploy-openstack.sh`.
   - If the kola-ansible "prechecks" command fails for localhost (especially the OS version check), it is OK and you can continue.
   - In some rare cases the kola-ansible "deploy" command fails due to runtime conditions (especially when pulling docker images). You can rerun all ansible and kolla-ansible scripts. Verify that the occurred error is reproducible before searching for a solution.
   - All steps on the VMs must pass. If not, try to resolve problems. Googling the error usually helps.
   - `tee` removes colours from the ansible output. For initial testing and possible debugging, you can remove the piping into `tee`. Add it again when generating logs for final submission.
   - Do not be afraid to tear everything down and redeploy it. Remember to work with checkpoints.
4. Check the public IP address of the controller VM via your browser. It should display the OpenStack login landing page. If not, check your firewall rules.

Congratulations, you are now a cloud provider and are hosting your own cloud computing platform.

Outputs:

- Files **pre-bootstrap.log**, **bootstrap-servers.log**, **prechecks.log** and **deploy.log**

3. Configure OpenStack

Execute the steps and verify that everything is working. Write a well-commented Shell script **prepare-openstack.sh** for steps 7, 8, 9 and 10. Add at least one comment for each of the steps.

1. Activate the previous python3 virtual environment on your local machine and install the openstack CLI client: https://docs.openstack.org/newton/user-guide/common/cli-install-openstack-command-line-clients.html
2. Get the keystone_admin_password value from passwords.yml to login via the OpenStack dashboard (user is "admin").
3. Download the admin-openrc.sh file (upper right drop-down menu) and place it into you working directory.

4. Use `source admin-openrc.sh` to set environment variables that are used for authentication. (Hint: You can adjust the admin-openrc.sh file to prevent repeated password input).
5. Test if the openstack CLI is working: `openstack host list.` It must work. If not, check the firewall rules.
6. Run the scripts `import-images.sh` and `create-ext-network.sh`
   - Depending on the network connection of your local machine, the image upload might take a while.
   - To prevent additional steps, we will use Ubuntu16.04. Images of newer versions need adjustment when using them with nested virtualization.
7. Create a new security group "open-all" (security groups are used to define firewall rules in OpenStack).
8. Add rules to the "open-all" OpenStack security group. Allow all TCP, UDP and ICMP ingress and egress traffic (Since it is a firewall behind a firewall, we consider it to not be too much of a threat. However, keep in mind that you need to set such rules cautiously when working on production systems).
9. Create another key-pair and import it to OpenStack. Copy the private key to your gc controller VM (home or .ssh directory). Adjust its permissions to 400.
10. Start a VM instance on OpenStack. Make sure to use the ubuntu image, the medium flavor, admin-net, the default security group, and your imported public key. The VM must reach state "RUNNING". Check it via cli or Dashboard.
11. Assign a floating IP to the VM.
12. From you gc controller VM, try to ping the floating IP of your OpenStack VM. It is not working. Think about why and answer question 2.
13. Copy the `iptables-magic.sh` script to you gc controller VM. Run it with **sudo**. Look into it and answer question 3.
14. From you gc controller VM, try to ping the floating IP of your OpenStack VM. It should work.
15. From your gc controller VM, try to connect to the OpenStack VM via ssh. If you followed step 8, the key should be located on the gc controller VM. The username is "ubuntu".
16. From the OpenStack VM, try to ping an external IP address, e.g. 8.8.8.8. It should work.
17. From the OpenStack VM, execute
    ```
    wget 169.254.169.254/openstack/2018-08-27/network_data.json
    ```
    You will need to submit the **network_data.json** file.

Bonus (worth 6 Points):
The nested VMs have internet access and can be reached from within the gc controller VM.
Due to NAT of the floating IP range on the controller node, it is not possible to reach the VMs from outside, i.e. (usually) you will not be able to ping the OpenStack VMs from your local machine.
We have configured an alias IP range for "cc-network1" and assigned it to the controller VM.

Write a shell script **access-openstack-vms.sh** that makes OpenStack VMs reachable from the internet, i.e. outside of the gc controller and compute VMs.

Questions:

1. Give a short overall explanation in your own words of what you did in this assignment (max. 200 words). (5 Points)
2. After creating all gc VMs, deploying OpenStack and starting an OpenStack VM, how many virtual networks are involved to establish the connectivity? (1 Point)
3. Initially, the OpenStack VM was not reachable from the gc controller VM (step 11). Why? (2 Points)
4. Look into the iptables-magic.sh script. What is happening there? Describe every command with 1-2 sentences. (5 Points)

Outputs:

- One shell script **prepare-openstack.sh**
  - Containing all commands from step 7, 8, 9 and 10 with respective comments.
- The **metadata.txt** file.
- The **network_data.json** file from step 17
- Text with answers to the questions.
  - Paste the text into the submission text field on ISIS
  - Please use max. 200 words per question
- **Bonus**: Shell script **access-openstack-vms.sh** that configures the gc controller VM such that OpenStack VMs are accessible from the internet.

4. Execute Performance Benchmarks

Install sysbench (and optionally bc) inside of the OpenStack VM.
Use the benchmark script from Assignment sheet 1. Copy it to the OpenStack VM (first to the gc controller VM and after that to the OpenStack VM). Again, run the sysbench experiment for two days and create the result file and respective plots as in Assignment 1.

Hints:

- When executing long-running experiments in a public cloud, remember to keep an eye on your credits. You should have more than enough credit to run these benchmarks, but make sure to shut down your VMs when you don't actually use them.

Outputs:

- One CSV file **`results-openstack.csv`**
- 4 plots generated by `plot.py`:
  - **`[cpu|mem|diskRand|diskSeq]-plot.png`**
- Compare each plot to the respective plot generated in assignment no. 1. Discuss the differences.
  - Write 3-4 sentences for each plot.

- Paste the text into the submission text field on ISIS

Submission Deliverables

Submit your solution on the ISIS platform as individual files. Please submit ONLY the necessary files and use exactly the given file names!
Submit the text with answers to the questions directly in the submission text field.

Expected submission files:

- spinup-gcp-for-openstack.sh
    - **15 points**
- `pre-bootstrap.log`, `bootstrap-servers.log`, `prechecks.log` and `deploy.log`
    - **10 Points (1, 2, 2, 5)**
- `prepare-openstack.sh`
    - **7 points**
- `network_data.json`
    - **5 points**
- Answers to questions of task 4
    - **13 points**
- `results-openstack.csv` and `[cpu|mem|diskRand|diskSeq]-plot.png`
    - **5 points**
- Comparison of plots
    - **2 points per plot (8 points)**
- Bonus: `access-openstack-vms.sh`
    - **6 points**

Total points: 63 (+6)

Final Warning: Please make sure your VMs are shut down when you are not using them! Make good use of your credits :-)