ulm university universität
uulm

# Tools & Concepts for Cloud Deployments

Exercise 2: Monitoring and Vertical Scaling

Christopher B. Hauser

Institute of Information Resource Management, Ulm University

SummerSchool 2019, Curitiba

# Exercise 2: Monitoring and Vertical Scaling

## Overview

This exercise uses the Mediawiki deployment from exercise 1, and adds:

- Monitoring with InfluxData (CPU, Network, Apache Requests etc.)
- Benchmark Mediawiki application with artificial workload
- Vertical Scaling and its impact
- Understand how the Openstack CLI works

## Lessons

1. Monitoring with InfluxData
2. Vertical Scaling
3. OpenStack APIs and CLI

Exercise as PDF

## Exercise Solution

- Solution as Markdown
- Solution as PDF

# Lesson 1: Monitoring with InfluxData

## Research: InfluxData TICK stack

First we need a new Instance/VM, which will host the monitoring data and provide a web-based dashboard to access it. The monitoring solution we will use is the TICK stack of InfluxData. While Telegraf will run on the VM with Mediawiki installed, we need another VM for InfluxDB and Chronograf.

Become familiar with InfluxData: https://www.influxdata.com/products/open-source/

## Question: Parts and responsibilities of the TICK stack

Please describe the components of the TICK stack and their purpose. Which of these components are needed for a minimum TICK setup?

## Task: Install the monitoring data sink

We will now install the InfluxDB and Chronograf in a new VM, which will be the data sink for several Telegraf monitoring adaptors. This guide follows the official installation guide:

https://docs.influxdata.com/influxdb/v1.2/introduction/installation/

Create a new security group on OpenStack (go to “Network” and “Security Groups”):

- Name: monitoring
- Create, and go to “manage rules”:
  - Add Rules for TCP ingres ports 8888, 8086, 8088

Create a new instance on OpenStack:

- Name: monitoring
- Source: Create new Volume set to “No”, then select Ubuntu Server 16.04 (ubuntu-1604)
- Flavor: select “small”
- Select both the default and the monitoring security groups
- Validate that your ssh key is selected

Then launch the instance (attach a public floating ip as before if necessary). Log in via SSH to the new vm with username “ubuntu” and the public IP you got assigned. In the new vm, install InfluxDB and Chronograf:

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/lsb-release
echo ”deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable”
    | sudo tee /etc/apt/sources.list.d/influxdb.list
```

```
sudo apt-get update && sudo apt-get install influxdb
sudo systemctl start influxdb
wget https://dl.influxdata.com/chronograf/releases/chronograf_1.3.0_amd64.deb
sudo dpkg -i chronograf_1.3.0_amd64.deb
```

In your monitoring VM you should have two services influxdb and chronograf running now. Validate that both are running, using the `systemctl` command.

Access the chronograf dashboard via your webbrowser, pointing to your public IP of the monitoring vm and port 8888 (http://134.60.64.YZ:8888). Tell Chronograf where InfluxDB is located:

- Connection String: http://localhost:8086
- Name: InfluxDB
- Username and Password empty!
- Telegraf database: telegraf
- Check the "Make this the default source" box
- Click "add source" button.

You will then see a empty list of hosts. Let's install the Telegraf on the Mediawiki page, to have some data to display :-)

### Question: What ports are used by influxdb?

Now that you have InfluxDB and Chronograf running, what ports are used by InfluxDB? You can get this information from your monitoring VM or from the documentation.

Hint: In the monitoring VM run `sudo netstat -tulpen` to get all ports that are open. Find the InfluxDB application.

### Task: Add monitoring to mediawiki

Next, connect via SSH to the VM hosting your **mediawiki**. Let's install Telegraf, which will collect monitoring statistics and send it to the monitoring VM.

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/lsb-release
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable"
    | sudo tee /etc/apt/sources.list.d/influxdb.list
sudo apt-get update && sudo apt-get install telegraf
sudo service telegraf start
```

We need to configure telegraf to point to your monitoring vm. Open as root (with sudo) e.g. with vim the file /etc/telegraf/telegraf.conf. In the `[[outputs.influxdb]]` section change the urls field from `urls =`

`["http://localhost:8086"]` to point to the IP address of your monitoring vm.

Restart telegraf with `sudo systemctl restart telegraf` and check the Chronograf dashboard. You should see monitoring data from host "main-server" now. You will see the basic system metrics of this vm. Let's add monitoring data from apache and mariadb next. Re-open the file /etc/telegraf/telegraf.conf and uncomment the following lines:

```
...
[[inputs.apache]]
   urls = ["http://localhost/server-status?auto"]
...
[[inputs.mysql]]
   servers = ["root:PASSWORD_FOR_DATABASE@tcp(127.0.0.1:3306)/"]
...
```

Now restart again telegraf, to apply your changes to the configuration. After some seconds, check the chronograf dashboard again. You have apache2 and mysql monitoring data now as well.

# Lesson 2: Vertical Scaling

## Task: Stressing the Wiki

Now that we have a monitoring and our mediawiki running, it is time to put some load on mediawiki.

1. Run a stressing application from your computer (client) that queries the wiki. For this purpose you can use the apache benchmarking tool. After installing it (https://httpd.apache.org/docs/2.4/programs/ab.html), run two separate load tests; one without and one with concurrency. WARNING: before running the test, make sure you are pointing it to YOUR wiki installation.
    1. `ab -n 5000 http://PUBLIC-IP/wiki/index.php/Main_Page`
    2. `ab -c 10 -n 5000 http://PUBLIC-IP/wiki/index.php/Main_Page`
2. Explain what consequences the permanent querying has for your application. In particular, identify the (e.g. CPU, memory and network) load caused for your server via the Chronograf dashboard.
3. How many request/per second can your server handle in both test cases (output of ab)? How many requests on the apache and the database are shown in the Chronograf dashboard?
4. What do you think is the bottleneck of the current setup? The network? The apache webserver? The database? Use the Chronograf dashboard to find possible bottlenecks.

## Task: Vertical Scaling

Try to scale your application vertically, by adding more virtual hardware resources.

Before we create a new instance with more CPU and Memory, we should back up our work, by creating a new snapshot, e.g. named "mediawiki-snapshot-2" of your main_server. Remove the first snapshot from exercise 1, since it has no monitoring installed.

To vertically scale your mediawiki setup, you can either use the "Resize Instance" option and change the flavour of the existing instance. Or you can recreate the instance from the snapshot. To resize, select the instance's menu, select "Resize Instance" and select the **new flavor medium**. Please note, that you have to confirm the resize action twice.

To recreate (e.g. if resize does not work), after the snapshot finished, delete your main_server instance with your mediawiki application, and launch a new instance. In the **source tab**, select in "Select Boot Source" the option "Instance Snapshot", and then pick **your snapshot**. Make sure to select "No" in "Create New Volume". Select **flavor medium**. If you need to attach a floating IP, Reassign the previously used one.

Next, validate that mediawiki and the monitoring data are still working properly.

Please note: the host attached to your public IP has now changed. When connecting via ssh, the host key verification will fail. You first have to remove the cached host key, which identified your old instance. On Linux, this can be done by editing the file `~/.ssh/known_hosts` or using `ssh-keygen -R YOUR_PUBLIC_IP`

## Question: Vertical Scaling

Repeat the steps for stressing the mediawiki. Compare the results of both benchmarks (without concurrency, with concurrency).

What differences can you see / not see? Can you explain them?

## Lesson 3: OpenStack APIs and CLI

OpenStack is an API driven software system: all interactions (e.g. working with the OpenStack Dashboard) issue requests to REST interfaces. Available REST API endpoints are listed in the OpenStack Dashboard (Project > Compute > API Access).

The API driven approach allows to use other clients like third party tools or the official OpenStack command line client.

### Task: Install and use OpenStack CLI

To explore the features provided by the OpenStackClient, install it on your machine:

https://docs.openstack.org/python-openstackclient/pike/#getting-started

Next, download the "OpenStack RC File v3" from the API Access site on the OpenStack Dashboard. Inside a terminal, you need to `source` this file, in order to load your credentials to the current terminal.

### Task: Explore the OpenStack CLI

After a successful installation, make sure you sourced the credentials into your current terminal.

Then explore the OpenStack CLI, e.g. `openstack server list`.

Can you manage to create the instances from exercise 1 and 2 via the OpenStack CLI only?