ulm university universität
uulm

Tools & Concepts for (Cloud) Deployments

Exercise 4: Automating Cloud Deployments

Christopher B. Hauser

Institute of Information Resource Management, Ulm University

2018-05-24

Exercise 4: Automating Cloud Deployments

Welcome to exercise 4. This time we will have the following lessons:

1. Concept of Cloud Computing
2. Terraform and cloud-init on OpenStack

Lessons learned

- Cloud computing supports scalable and elastic applications
- Terraform let you script your cloud infrastructure
- cloud-init allows virtual machine configuration and installation
- How to deploy instances with terraform and cloud-init in OpenStack

Lesson 1: Concept of Cloud Computing

Before we continue with the practical part of the exercise, we first need to understand the characteristics and the concepts of cloud computing.

Research: Essential Characteristics

According to the *NIST Defition of Cloud Computing* [1] there are several essential characteristics of cloud computing. Read the document, especially focus on the characteristics.

[1] http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Question: Essential Characteristics

- What are the essential characteristics according to the NIST Definition of Cloud Computing?
- What characteristics and features are needed in order to provide “rapid elasticity”?
- What is the difference between scalability and elasticity?

Research: Infrastructure and Application Deployment

The cloud stack we used so far looks like the following:

Cloud Stack	Example	Deployment Tool
Application Component	Mediawiki	cloud-init
Virtual Resource	Instance m1.small	Terraform
Cloud Platform	OpenStack	-

We started on OpenStack as a Cloud Platform, manually created virtual resources (virtual machines of flavor m1.small) and manually installed the mediawiki application inside the virtual machines.

We will now start automating the manual deployment, from bottom to top. The first tool we will use automates the creation of virtual resources on a cloud platform. Please make yourself familiar with the features presented on the product page of Terraform [1]. The second tool will then automate the deployment of mediawiki inside the virtual resources. Please make yourself familiar with the features of cloud-init [2].

[1] https://www.terraform.io/

[2] https://cloudinit.readthedocs.io/en/latest/

Question: Infrastructure and Application Deployment

- What are the three stages a Terraform script walks through?
- Which cloud platforms are supported by Terraform?
- How can cloud-init be used to deploy an application inside a virtual machine?

Lesson 2: Terraform and cloud-init on OpenStack

Clean up your OpenStack project. *Remove all the virtual machines, the private networking, the router, and the security groups.* We will now create the mediawiki from scratch - automated!

Task: Install Terraform on your workstation

Terraform runs on your workstation. It reads a terraform script from your file system, and then connects to the specified cloud platform to create the infrastructure.

Download [1] and install [2] the newest version (v0.11.7) of Terraform on your workstation. There might be a package for your linux distribution.

[1] https://www.terraform.io/downloads.html

[2] https://www.terraform.io/intro/getting-started/install.html

Task: Deploy Mediawiki with Terraform

Download and extract the `terraform.zip` file from Moodle to your workstation, where you installed terraform. The extracted folder will be named `working directory` in the following. It contains several terraform files and bash scripts, which are necessary to deploy mediawiki on OpenStack.

Before you can deploy the mediawiki with terraform, please change your OpenStack username and tenant. In the working directory, edit the file `provider.tf` and change `xyz12` to your user id. Look up the name of your ssh key pair in OpenStack (via Access & Security, Key Pairs), edit the file `instances.tf` and replace the existing name in the field `key_pair` with your ssh key name. If you are comfortable with writing your password in the password field, you can start right away. But there's a more secure way for your password (tested with linux bash):

- open a new terminal
- navigate to the working directory
- read in your password into a variable: `read -sr OS_PASSWORD_INPUT`
- export the variable `export OS_PASSWORD=$OS_PASSWORD_INPUT`

Within the same terminal, your password will now be known to applications you start. If you close the terminal, your password will not be stored.

Let's start the mediawiki deployment:

- use the terminal from before
- make sure you are in the working directory
- use `terraform init` to install additional terraform dependencies in your workspace
- use `terraform plan` to see the actions that will happen

- use `terraform apply` to run those actions

Relax. Check the OpenStack dashboard. You should see three virtual machines: loadbalancer, database, mediawiki-1. If you navigate your browser to http://floating_ip/wiki you should see a working mediawiki instance.

Task: Extend the Terraform deployment

As you can see, we have only one mediawiki vm, and no monitoring vm. Take the existing Terraform scripts as a starting point and extend. You can use the Terraform docs for OpenStack, if needed [1].

Extend the setup by:

1. Add two more mediawiki vms (3 in total)
2. Add a monitoring vm
3. Add Telegraf to all your vms

With your scaled out deployment you can easily run the stress benchmarks again, to see how the system can handle requests with 3, 4, or even 5 mediawiki vms.

[1] https://www.terraform.io/docs/providers/openstack/index.html

Questions: Terraform and cloud-init

- Where can you watch and validate the execution of a cloud-init script?
- How does Terraform help you with scaling elastically?
- Can you imagine how to automatically scale your setup when the load increases/decreases?