DevOps and Security Lab 2 - Infrastructure as Code with clouds (alternate lab flavor)

> individual assignment

The lab goal is to understand and learn the core tools and techniques that are used in day by day DevOps routine.

Task 1 - IaC Theory

Briefly answer for the following questions what is and for what:

- **ansible** directory:
  - `ansible.cfg`
  - `inventory` folder
  - `roles` folder
    - `tasks`
    - `defaults/group_vars`
    - `handlers`
    - `templates`
    - `vars`
  - `playbooks` folder
  - `meta` folder
- **terraform** folder:
  - `main.tf`
  - `variables.tf`
  - `outputs.tf`

Task 2 - Prepare your application

Find and choose (for sure it's better if you take your own developed project) a simple application. For example, it could be a web server with the static HTML page, time zones server or a currency calculator. Use whatever programming language that you want (python, golang, C#, java...). Include the link to VCS (link to the remote repository) where your application is stored.

Bonus: prepare a microservices instead of standalone application, e.g. full stack web application with web server, database...

Task 3 - Dockerize your application

1. Build Docker image for your application (make *Dockerfile*).

   Try to look at the best *Dockerfile* practices and try to follow them.

Bonus: use docker-compose in the case of microservices.

Task 4 - Deliver your application using Software Configuration Management

1. Get your personal cloud account. Free tiers for a AWS, GCP and some other popular cloud providers doesn't worked in Russia more. If you already have a such account, it may work and be enough for this lab. If not, try other cloud providers with a free subscription, i.g. Yandex.Cloud. **If you will not be able to work with cloud, you have to proceed within the local deployment for the whole Task 4.** Thus, think about a local virtual machine for the further tasks. Include a small explanation into the report why you were forced to work locally.
2. Use Terraform to deploy your required cloud instance. Please notice that to run `terraform init` command you have to use VPN. Look for the best Terraform practices and try to follow them. If for a some reason you will not able to use any cloud service, prepare a local VM using Vargant tool. Include the explanation into the report about the inability to work with cloud or if it was impossible for you to setup a VPN.
3. Choose Software Configuration Management (SCM) tool. Ansible is the **industry standard**. Of course, we have other SCM solutions such as SaltStack, Puppet, Chef. You can try them but remember that it is probably more difficult to work with these tools and you are responsible for your choice.
4. Using SCM tool, write a playbook tasks to deliver and run your application to cloud instance/local VM. *Try to separate your configuration files into **inventory/roles/playbooks** files. In real practice, we almost newer use poor playbooks where everything all in one.*
5. Provide the link to git repository with all your labs configurations.

   Also try to use the best practices for you SCM tool, e.g. Ansible.

Bonus: use Ansible Molecula and Ansible Lint to test your application before to deliver it to cloud.

As the final result after applying all configurations, you should be able to show the working execution of your running app on the cloud instance (local VM) according to this app destiny. For example, if you've prepared a web server, then you have to be able to open http://yourdomain.com in your browser and see the web site page.

Task 5 - Play with SCM cases

1. Complete at least one "advanced" case using Ansible or other SCM tool. Actually it's coincides with the task 3 from the default lab flavor:

- NTP
- Apache Web Server
- Kubernetes helm chart deployment
- Nginx & PHP-FPM & MariaDB & WordPress installation and configuration
- development environments organization (ssh keys management...)
- certificates creation (let's encrypt)
- Golang application with microservices
- Resize volume partitions on the target hosts
- Make a network configuration between several virtual hosts (vlans, IPs, routing, firewall ...)
- ...

Requirements:

- never forget to follow to the best practices
- use *templates/jinja2*

- separate your configuration files into **inventory/roles/playbooks** files
- do not expose sensitive date and secrets in plain text!
- play with *handlers* and *meta* (optional)
- try to avoid a raw shell command in Ansible tasks as much as you can
- follow to **Idempotence** principle!
- play with Ansible variables priorities (precedence) (not mandantory)
- do not hard code variables values in `tasks`
- **DO NOT COPY/PAST EXISTING ANSIBLE PLAYBOOKS!** If you use any code as a base and example, just mention it and list what you've changed and how you understand it

Bonus: use Ansible Molecula and Ansible Lint to test your role before to run the corresponding playbook.

`If you use an other SCM tool, replace the task goals according to your tool specifications.`

2. Provide the link to git repository with all your labs configurations.

Appendix 1 - Some Docker best practices

- Do not use `:latest`
- Exclude with .dockerignore
- Minimize the number of layers
- Build the smallest image possible
- Follow tag versioning of your images
- Don't be a root user

Appendix 2 - Some Terraform best practices

- Make understandable naming
- Don’t commit the .tfstate file
- Back up state files
- Manipulate state only through commands
- Use variables
- Use validate and plan commands frequently
- Abstract code to maximize reuse
- Use modules where it is necessary