LS lab 3 - Infrastructure as Code (IaC)

> New team of two or work individually except Task 5

The goal is 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:

- **git** repository:
  - `/.git`
  - `/.github`
  - `.gitignore`
  - `.gitmodules`
- **ansible** directory:
  - `ansible.cfg`
  - `inventory` folder
  - `roles` folder
    - `tasks`
    - `defaults`
    - `files`
    - `handlers`
    - `templates`
  - `playbooks` folder
- **terraform** folder:
  - `main.tf`
  - `variables.tf`
  - `outputs.tf`

Task 2 - Choose your application/microservices

Base level (it means that this task will be evaluated very meticulously): find (much better to write) a simple application with REST interface. For example, it could be a web server that returns "Hello to SNE family! Time is: < current time in Innopolis >". Use whatever programming language that you want (python, golang, C#, java...).

Semi-Bonus: prepare 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).

   Look for the best Docker practices, try to follow them and put into report.

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

Task 4 - Deliver your app using Software Configuration Management

1. Get your personal cloud account. To avoid some payment troubles, take a look on free tier that e.g. AWS or GCP offers for you (this account type should be fit for a lab).
2. Use Terraform to deploy your required cloud instance.

   Look for the best Terraform practices, try to follow them and put into report.

   Bonus: use Packer to deploy your cloud image/docker container.
3. Choose SCM tool. Suggestions:
   - Ansible (**default choice**)
   - SaltStack
   - Puppet
   - Chef
   - ...
4. Using SCM tool, write a playbook tasks to deliver your application to cloud instance. *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.*

   Also try to use the best practices for you SCM tool and put them into report.

   Use *Molecula and Ansible Lint* to test your application before to deliver to cloud (or their equivalents).

Task 5 - Teamwork with the version control system (optional)

You can do this task if you have a different projects with your cooperation team and you want to be more familiar with git.

In production, before deploying the version of the service/application, we receive a review from colleagues.

1. Create and log in to your personal version control system account on the `git` engine:

- github (**default choice**)
- gitlab
- bitbucket
- ...

2. Create a repository with your application/microservices and all required scripts/configs.
3. Synchronize your local and remote repository.
4. Create a separate branch for development, as well as protect your master branch in the repository from direct commits to it.
5. Create a Pull Request from your developer branch to the master branch.
6. Your colleague should get explanation about your work and conduct a review of your PR.
7. Receive an approvement, merge your PR and synchronize the local and remote repositories.

Bonus: implement steps 2 and 4 via Terraform.

As the final result after applying all configurations files, you should be able to open https://yourdomain.com in your browser and see the message "Hello to SNE family! Time is: < current time in Innopolis >". Or show other final result of your running app on cloud instance according to its purpose.

Task 6 - Play with SCM cases

The goal is to easily, reliably and quickly maintain different kinds of systems at once. Prepare two different guest systems e.g. an Ubuntu Server and Fedora. Then play with some Software Configuration Management (SCM) to automate system preparations on those. You might create and provide your own ideas or follow to some suggestions:

- NTP
- default shell
- Apache Web Server
- Docker
- Nginx & PHP-FPM & MariaDB & WordPress
- Ansible Vault (definitely good choice)
- IPtables (close all machine ports exclude ssh, http, https)
- development environments organization (ssh keys management...)
- ...

Never forget to try to use the best practices. Complete as many tasks as possible (at least two cases for a team of two people, one case for an individual work).

Some best practices