Final Project
Virtualization and Containers

Instructions

Virtualization

- Create a new repository which name applies the following convention: `gh-runner-devops-team-##`
- Write the code to create 2 VMs
- Each VMs must be provisioned and configured as a GitHub Runner at the global organization settings.
- The name of each VM must apply the following convention: `devops-runner-team-##`
- The VM 1 has to be provisioned with Docker.
- The VM 2 has to be provisioned with Podman.

Containers

Define and up the following stack by using Docker-Compose: (These containers are going to be provisioned in the VM 1)

1. Stack with Docker-Compose
    - Create a `docker-compose.yaml` file in the root of your project.
    - Define the following services:
        - **Sonarqube:** This service has to use an external database in Postgres (or MySQL).
        - **Jenkins:** Create and run a Jenkins service.
        - **Nexus:** Create and run a Nexus 3 OSS service.
        - **Database:** This service should be used by Sonarqube service (Postgres or MySQL).
        - **Portainer:** Create and run a Portainer service.
2. Demonstrate how much you learned about Docker and Docker Compose
    - Create and use networks and volumes among the services.
    - Define an order of starting with `depends_on`
    - All the services have to share a user-defined network bridge at least.
    - Version your `docker-compose file` in your repo.

Requirements

- Send an email to the trainer in which you have to present:
    - Link of source code repo (Public Repo).

- Define your docker-compose file in the way in which all the services, networks and volumes should be created and running with a simple `docker-compose up -d` command..
- Make sure that your docker-compose file works without bugs or issues.
- You'll have 5 services (total) in your compose file (Sonarqube, Database, Jenkins, Nexus, Portainer).
- Except the Database container, all the services must be accessible from outside the VMs

Live Demo Day

- **Friday, October 7th, 2022**

Considerations:

- Official documentation for Docker Compose: https://docs.docker.com/compose/compose-file/compose-file-v3/
- You're totally free to apply all the knowledge you want, such as volumes, networks, environment variables, ports, builds, depends_on, policies, tagging, publishing, etc..
- Apply Git best practices, such as:
    - **No commits with sensitive data.**
    - Official code in the main branch.
    - Create Pull Request to integrate your code changes.
    - Remove feature branches after approving Pull Requests.
    - Apply significant commits messages or apply conventions (https://www.conventionalcommits.org/en/v1.0.0/).