System Administration & DevOps Code-Challenge

# Application

Write a simple containerized OCI-compatible *Ruby/Python/Java* webserver. The application should connect to a SQL database.

The database content can be anything you like and must additionally contain a timestamp column.

Depending on the requested endpoint:

* If it’s a GET on /greeting, it shall respond with a simple text with the total number of rows in the DB.
* Otherwise, if it’s a POST on /messages, it should add a new row, using some default values and shall return the content of the newly added row.
* The / route should temporarily redirect to the /greeting route.
* All operations should return the appropriate HTTP status codes.
* (Optional) If the requested data contains a date-time previous to the current date, the application should show all matching rows limited to 10. If the date-time is after the current date, it should return a response message explaining the particular situation.

# Server

* Create a *Vagrant* virtual machine running *Debian* or *Ubuntu* Linux vanilla image.
* The VM shall be auto-provisioned using Ansible and configured with the following:
* *Docker* Engine and *Docker Compose*.
* The application shall be executed by *Docker Compose*.
* The database can run inside the same VM or (optional) on a network-attached VM. *PostgreSQL* or *MySQL/MariaDB* database server shall be used.
* The application shall run on a two-replicas deployment and a load balancer or reverse proxy accessible from the VM host.
* An appropriately configured firewall.

The application building and deployment can be done using Ansible or directly using Vagrant configuration.

# Expected outcome

* The application source code.
* The Vagrantfile & Dockerfile.
* The Ansible playbook and roles.
* Idempotent Vagrant and Ansible.
* Completely reproducible setup and configuration.

You can host this using a cloud Git repository service.