Final Project

Description

The final project is a practical exam that will test your ability to deploy a Django application in a Kubernetes cluster, configure CI/CD pipelines, set up monitoring tools, and configure Nginx with TLS. You need to set up the infrastructure for the Django application, write the necessary Kubernetes configuration files, create a CI/CD pipeline in GitLab, set up monitoring tools, and configure Nginx with TLS. You need to submit the zip file and the YouTube video link to the Moodle platform.

About the Thumbnails (Django) application

The Thumbnails application is a simple Django application that provides a single functional endpoint `/thumbnails/{max_height}x{max_width}/{url}/` that takes max sizes and a URL, and returns a thumbnail of the given URL with the specified dimensions. The application also provides 4 endpoints:

- `/healthcheck/` - returns the health status of the application
- `/api/v1/metrics/` - returns the metrics of the application
- `/api/v1/swagger/` - returns the Swagger documentation of the application
- `/admin/` - Django admin panel

You do not need to modify the Django application or even learn Django to complete the project. The application is containerized and ready to be deployed in a Kubernetes cluster. Please look at the `Dockerfile`, `docker-compose.yml`, `Makefile`, and `README.md` files in the Django application root directory for more information. These are the only files you need to know about the project as a DevOps engineer.

The architecture of the Thumbnails application is as follows:

- Django application - a web application that provides the functionality described above
- PostgreSQL database - a database to store the data
- Redis broker - a message broker for Celery
- Celery worker - a worker to process asynchronous tasks

Note, the static and media files are served by the Django application itself. You do not need to set up a separate server to serve these files. See the LOCAL_SERVE_STATIC and LOCAL_SERVE_MEDIA environment variables in the `docker-compose.yml` file.

Pay attention to the environment variables used. You will need to set these environment variables in the Kubernetes configuration files.

Please, also look at the response of the `/api/v1/metrics/` endpoint to get all the application metrics. You will find some metrics with the

prefix `resize_image_` that you need to monitor and configure alerts for them. The metrics with the prefix `resize_image_` are set by the `/thumbnails/{max_height}x{max_width}/{url}/` endpoint. It generates the following metrics:

- `resize_image_request_count_*` - the number of requests.
- `resize_image_process_count_*` - the number of images processed. If the image is already processed, it will not be processed again.
- `resize_image_process_time_*` - the time is taken to process the given images as a summary.

Part 1: Infrastructure setup

You need to set up the infrastructure. You need to deploy the Thumbnails application in a Kubernetes cluster and launch other services in any way you like. You need to set up the following services:

- Minikube - a single-node Kubernetes cluster to deploy the Thumbnails application
- Gitlab Repository - a Gitlab repository to store the Thumbnails application code
- Docker Hub - a Docker Hub registry to store the Docker image of the Thumbnails application
- SonarQube - a SonarQube server to scan the Thumbnails application for security vulnerabilities
- Prometheus - a Prometheus server to collect the metrics of the Thumbnails application
- Grafana - a Grafana server to visualize the metrics of the Thumbnails application
- AlertManager - an AlertManager server to send alerts based on the metrics of the Thumbnails application
- Telegram bot and channel - a Telegram bot and channel to receive alerts from AlertManager
- Nginx with TLS - an Nginx server as a reverse proxy to forward the requests to the Thumbnails application running in Minikube and the monitoring tools running in the host machine

You need to set up the infrastructure in the following way:

- Set up Minikube in your local machine **manually**
- Set up Gitlab Repository **manually**
- Set up Gitlab Runner **manually**
- Set up Docker **manually**
- Set up Docker Hub **manually**
- Set up Telegram bot and channel **manually**

- Set up SonarQube using **Ansible**
- Set up Grafana, Prometheus, and AlertManager using **Ansible**
- Set up Nginx with TLS **Ansible**

For Ansible setups, you need to create the necessary Ansible playbooks to set up the services. You need to create a directory called `ansible` in the root of the Thumbnails application directory and put all the Ansible playbooks in this directory.

Part 2: Kubernetes configuration files

You must write the necessary Kubernetes configuration files to deploy the Thumbnails application in Minikube. The Thumbnails application should be deployed in Minikube. You must create the appropriate Kubernetes resources to deploy the Thumbnails application, PostgreSQL database, Redis broker, and Celery worker. You need to expose the Thumbnails application to the outside world using a NodePort service. You must also set up the necessary environment variables in the Kubernetes configuration files. Do not forget to configure the persistent volumes for the PostgreSQL database and Redis broker (you can read about the persistent volumes in the Kubernetes documentation).

Create a directory called `k8s` in the root of the Thumbnails application directory and put all the Kubernetes configuration files in this directory.

Part 3: CI/CD pipeline

You need to set up a CI/CD pipeline for the Thumbnails application in GitLab. You need to create the following CI/CD pipeline:

- Build job - this job should build the Docker image of the Thumbnails application and push it to the Docker Hub registry.
- Check the code formatting job - this job should check the code formatting of the Thumbnails application. See the `Makefile` in the Thumbnails application to understand how to check the code formatting.
- Test the code job - this job should test the Thumbnails application. See the `Makefile` in the Thumbnails application to understand how to test the Thumbnails application.
- Scan the code job - this job should scan the Thumbnails application for security vulnerabilities. See the `Makefile` in the Thumbnails application to understand how to scan the Thumbnails application.
- Sonar scan job - this job should scan the Thumbnails application with SonarQube. You need to set up SonarQube either in Minikube or in the host machine. You need to configure the SonarQube server in the job to scan the Thumbnails application.
- Deploy the application to Minikube job - this job should deploy the Thumbnails application in Minikube. You need to use the Kubernetes

configuration files you created in Part 1 to deploy the Thumbnails application in Minikube. You have to make it a manual job with confirmation to deploy the application to Minikube. This is a best practice to avoid accidental deployments in the real-world scenario.

Create a `.gitlab-ci.yml` file in the root of the Thumbnails application repository and put the CI/CD pipeline in this file.

Part 4: Monitoring

You need to set up monitoring tools for the Thumbnails application. You need to set up Prometheus to collect the metrics of the Thumbnails application. You need to set up Grafana to visualize the metrics of the Thumbnails application. You need to set up AlertManager to send alerts based on the metrics of the Thumbnails application. You need to set up a Telegram bot and channel to receive alerts from AlertManager. You need to create alerts in Grafana for the given metrics.

You need to import the Grafana dashboard for the Django application by using the following link: https://grafana.com/grafana/dashboards/17658-django/

You need to create another Grafana dashboard for custom metrics with the prefix `resize_image_`. You need to create the following visualizations:

1. Visualization of # of requests.
2. Visualization of # of images processed.
3. Visualization of the average time for the last 1 minute to process the image.
4. Visualization of the ratio of the # of images processed to the # of requests.
5. Visualization of the intervals when the # of images processed is less than the # of requests by 50%.
6. Visualization of the intervals when the # of images processed is greater than the # of requests by 10%.
7. Visualization of the intervals when the average time to process the image is more than 5 seconds.
8. Visualization of the intervals when the average time to process the image is less than 1 second.

You need to create the following alerts in Grafana:

1. Alert when the # of images processed is greater than the # of requests by 10%.
2. Alert when the average time to process the image is more than 5 seconds.

You need to configure AlertManager to send alerts to the Telegram bot and channel.

Part 5: Nginx with TLS

You need to set up an Nginx server as a reverse proxy on the host machine to forward the requests to the Thumbnails application running in Minikube and the monitoring tools (Prometheus, Grafana, AlertManager) running in the host machine. You need to set up the following configurations:

- Nginx configuration to forward the requests to the Thumbnails application running in Minikube. The Thumbnails application should be accessible at `https://thumbnails.devops.final`.
- Nginx configuration to forward the requests to the monitoring tools (Prometheus, Grafana, AlertManager) running in the host machine. The monitoring tools should be accessible at `https://grafana.devops.final`, `https://prometheus.devops.final`, and `https://alertmanager.devops.final`. Do not forget to configure external URLs in Prometheus and AlertManager.
- Nginx configuration to forward the requests to the SonarQube server running in the host machine. The SonarQube server should be accessible at `https://sonarqube.devops.final`.

You need to set up the TLS certificates for each domain. You can use the self-signed certificates for the domains.

You need to create a directory called `nginx` in the root of the Thumbnails application directory and put all the Nginx configuration files and TLS certificates in this directory.

Submission

You need to submit the zip file and the YouTube video link to the Moodle platform.

Zip file

You need to submit all the files as a single zip file. The zip file should contain the following files:

- The Django application project directory
- The Ansible playbooks
- The Kubernetes configuration files
- The `.gitlab-ci.yml` file
- The Grafana dashboards exported as JSON files (you can export the dashboards from Grafana)
- The Configuration files for Prometheus, Grafana, and AlertManager
- The Nginx configuration files and TLS certificates

Video recording (optional)

You have to also provide a video recording of the deployment process with explanations. The video should be 10-15 minutes long. You can use any screen recording software to record the video.

The video should contain the following: - Demonstration of Ansible playbooks to set up SonarQube, Prometheus, Grafana, AlertManager, and Nginx. Note, you need to destroy the existing infrastructure and set up the infrastructure from scratch. - Demonstration of CI/CD pipeline in GitLab and the deployment process to Minikube. Note, that you need to destroy the existing deployment and deploy the Thumbnails application from scratch using CI/CD pipeline. During video recording, you need to show the deployment process. - Demonstration of the SonarQube scan job in the CI/CD pipeline. You may need to configure SonarQube after recreating the infrastructure. Please prepare the configuration steps in advance. - Demonstration of the monitoring tools (Prometheus, Grafana, AlertManager) and the alerts in Grafana. You may need to configure monitoring after recreating the infrastructure. Please prepare the configuration steps in advance. - Demonstration of the Telegram bot and channel to receive alerts from AlertManager and the alerts in the Telegram channel. - Demonstration of Nginx with TLS to forward the requests to the Thumbnails application running in Minikube, to the monitoring tools (Grafana, Prometheus, AlertManager), and SonarQube running in the host machine.

You need to upload the video to YouTube, add the necessary permissions to view the video and provide the link to the video in the submission.

If you do not provide the video recording, you need to demonstrate the deployment process during the final exam which is really risky. So, it is highly recommended to provide video recordings to avoid any issues during the final exam.

Grading criteria

The final project will be evaluated based on the following criteria:

- Infrastructure setup - 20%
- Kubernetes configuration files - 20%
- CI/CD pipeline - 20%
- Monitoring - 20%
- Nginx with TLS - 20%