
ST2DCE - DevOps and Continuous Deployment (M2, M2-APP-LSI, M2- PRO - 2324S9) - Project

**EFREI 2023 - 2024
ZAKARI MADOUGOU A.**

General Guidelines

The main goal of this project is to build an application and deploy it on a Docker/Kubernetes infrastructure. It's not a question of development and the provided application only need to be build and deployed. You will need all the tools/environment used for practical exercises.

Organize yourself in a team of at least 4 members, every team must design a team leader who will:

- Send a zip file to **abdoul-aziz.zakari-madougou@intervenants.efrei.net** containing all the deliverables (Yaml files, Dockerfile, Jenkinsfile, screenshots, Command lines) along with a report describing the results that you obtained.
OR
- Push everything in a private git repository with a readme file that explains the structure/content and give read access to **abdoul-aziz.zakari-madougou@intervenants.efrei.net**

*For scoring purposes, in order to get full grade, you must successfully complete the bonus section of each exercise... and **+2 will be attributed for the overall presentation of the report and final deliverables in relation to the quality and clarity of the explanations.** The work must be delivered by the end of February.*

Part One – Build and Deploy an application using Docker / Kubernetes and Jenkins pipeline. (+8)

Considering the application written in Go language available at Git url: <https://github.com/efrei2023/ST2DCE-PRJ.git> provide a pipeline to build and deploy the application:

-
1. Draw up a diagram of your solution and describe the target architecture and tool chain you suggest to achieve full continuous deployment of the application. **(+2)**
 2. Customize the application so that the **/whoami** endpoint displays your team's name and and deploy it on local docker engine by using Jenkins. **(+2)**
 3. Update the pipeline to deploy the application on your Kubernetes (Minikube) cluster according to the following scenario: **(+3)**

Deploy in the first environment named **development**, for example Following a validated test (A possible test is to perform a curl on one of the application endpoints and check that there are no errors in the response), continue with deployment in an environment we'll call **production**.

*If you don't want to use docker hub or any other registry, use the **'minikube image load image:tag'** command to make it accessible directly in your minikube cluster and specify **'imagePullPolicy: Never'** in your deployment descriptor.*

4. **BONUS (+1):** Build the docker image using the buildpack utility and describe what you observe in comparison with the Dockerfile option.

*See how to install the buildpack utility:
<https://buildpacks.io/docs/tools/pack/>*

Part Two – Monitoring and Incident Management for containerized application **(+7)**

1. Install Prometheus stack and Grafana by using their Official Helm Chart **(+3)**
Grafana: <https://grafana.github.io/helm-charts>
Prometheus: <https://prometheus-community.github.io/helm-charts>

Access to Grafana Web UI and configure a data source with the deployed Prometheus service URL.

2. Configure Alert Manager component and setup Alerts **(+3)**:

Go to <https://samber.github.io/awesome-prometheus-alerts/rules.html#kubernetes> and choose one alert to implement and get it on Alert Manager.

For example: Pod has been in a non-running state for longer than 5 minutes.

3. Bonus **(+1)**: Configure another alert and send it by e-mail to **abdoul-aziz.zakari-madougou@intervenants.efrei.net**. You must have your team's name in the alert subject/message in order to be identified.

Part Three – Logs Management **(+3)**

1. Install the **grafana/loki-stack** chart from Grafana Official Helm Chart **(+1)**

Remember that you already have a Grafana installation in the previous section. As the helm chart installs Grafana again, you need to disable the option.

2. On the Grafana application, after configuring the Loki datasource, create a query that displays all log lines containing the word 'error' for your 'Production' namespace. **(+2)**