Diploma task

Create a simple Web-application (see the description in the “Application” section below), CI/CD infrastructure and pipeline for it.

Hi Vladislav,

Sending you the final task. You can find the common part attached. Your variant:

Variant 9. For a marketing campaign collect information about Star Wars universe using open API https://swapi.dev/. Store the following data in the local database: character: Name, Gender, homeworld, and other if it is necessary; starship: Name, model, manufacturer and other if it is necessary. Create a report about characters who are using ships with the most (top 10) weight of lading.

Let me remind you that it has status of a Diploma task which demonstrate success of your finishing the course. It is highly recommended to perform it during the education (you can find recommended modules in the common part). If you defend your diploma before finishing the modules, it will allow you to be hired during the education.

In case of any blockers with performing the task you can contact your Resource Manager or the authors of the task: Aleksandr Tolkachev Aleksandr\_Tolkachev@epam.com; Sergei Smozhevskii Sergei\_Smozhevskii@epam.com; Dmitrii Overchenko Dmitrii\_Overchenko@epam.com

# Acceptance Criteria and presentation

A short presentation (.ppt or other) which contains description of the solution should be prepared and sent to the commission before a demo session.

The working application with the pipeline is to be demonstrated live on a “protection of the diploma” session for experts with comments and explanation of the details of the implementation, reasons of choosing tools and technologies.

Detailed requirements/criteria:

|  |  |  |
| --- | --- | --- |
| Criteria | Reqiurements | Related Module |
| SCM | Application sources should be placed in Git repository. Branching strategy should be explained. | Git |
| Tests\* | CI pipeline may contain unit tests, smoke tests, linter check. | CI/CD |
| Quality gate | CI/CD pipeline should use some quality/vulnerability control tool like a Sonar or Anchore. | CI/CD |
| IaC | CI/CI and runtime infrastructure should be described as a code using Terraform, CloudFormation, or any similar tool. On the demonstration deployment procedure should be shown. | Cloud, Terraform, Ansible |
| Orchestration | All non cloud-native tools should be spinned up inside a K8S/OpenShift cluster inside a cloud. Application runtime environments should be inside the cluster too. | Kubernetes |
| Logging | Infrastructure should have centralized log collection/display system. Logs of the application components and infra components should be collected. | Monitoring and Logging |
| Monitoring | Infrastructure should have centralized metric collection/display system. Metrics of the application components and infra components should be collected. | Monitoring and Logging |
| Runtime/Deployment | Runtime infrastructure should have production and non production environments. Deploy/release strategy should be explained. | CI/CD |
| Scalability/redundancy | Scalability should be provided and demonstrated | Kubernetes |
| Cloud and Cost efficiency\*\* | Cloud resources and services must be used for the task. Report about the Cloud resource usage and the cost must be provided in the presentation. It should be efficient (minimal) – in accordance to the solving tasks. You can choose any cloud provider taking into account possible extra costs for the resources. | Cloud |

*\* Nice to have – optional*

*\*\* Be careful with the Cloud resource usage and check the costs for not to exceed limits! Switch off your machines when you are not using them!*

# Application

Develop a simple (lightweight) 3-tire application (front-end, back-end, database).

Back-end (collects data) must:

1. Retrieve a portion of data from API (see in your Variant) and store it in a database

2. Update data on demand

3. Update DB schema if needed on app’s update

Front-end (outputs data) must:

1. Display any portion of the data stored in the DB

2. Provide a method to trigger data update process

Database:

1. Choose Database type and data scheme in a suitable manner.

2. Data must be stored in a persistent way

3. It’s better to use cloud native DB solutions like an RDS/AzureSQL/CloudSQL.

**You’ll get your Variant of the application individually.**