Corsound

Corsound Interview Practical Test

DevOps and Integration Engineer

January 2024

This test goal is to provide a comprehensive assessment of the candidate's practical skills in Docker and Kubernetes, covering key areas like containerization, orchestration, scaling, and troubleshooting.

Corsound

Part 1: Docker

1. **Containerize an Application**:
Create simple FAST API app that reads some data from DB and caches it in Redis
The app has to have tests.
Build CI/CD pipeline for production and dev env.

Possible tools and technologies to use:

- Python and Flask (or Node.js)
- PostgreSQL
- Redis
- Bitbucket or Github pipelines
- Unittest framework
- Docker
- You can use any cloud provider.

- Create a Dockerfile to containerize the application.
- Bonus: Optimize the Dockerfile for size and build speed.

2. Compose Multiple Services:
- I app above use multiple services (e.g., a web app, a database, and a caching service).
- Write a docker-compose.yml file to run these services together.

3. Debugging a Docker Issue:

Below is an example of a Dockerfile meant to containerize a simple Python Flask application. However, there's a subtle mistake in it. See if you can identify and fix the issue:

Corsound

Python
```dockerfile
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

This Dockerfile is intended to create a Docker image for a Python Flask application. The application code (including app.py and requirements.txt) is assumed to be in the same directory as the Dockerfile. The subtle mistake in this Dockerfile might affect the application's functionality. Once you've spotted the mistake, consider how it could be corrected.

Corsound

Part 2: Kubernetes

1. Deploying to Kubernetes:
- Take web application and its Docker image from the previous part.
- Create Kubernetes manifests (Deployment, Service, etc.) to deploy this application in a Kubernetes cluster on GCP or Azure.

2. Configuring Persistent Storage:
- Modify the Kubernetes manifest to include persistent storage for a database service.

3. Scaling and Load Balancing:
- Configure horizontal pod autoscaling based on CPU/Memory usage.
- Set up load balancing and ingress controllers.

4. Troubleshooting a Kubernetes Cluster:
- Describe a common issues in a Kubernetes environment
- Explain how you would diagnose and resolve the issue.

Bonus tasks (Optional)

- CI/CD Pipeline Integration: Integrate the application deployment with a simple CI/CD pipeline using tools like Jenkins, GitLab CI, or GitHub Actions.
- Monitoring and Logging: Set up basic monitoring and logging for the Kubernetes cluster.