Candidate is provided 48hours time to come back with a working prototype.

Provide a word document containing

- Link to Git repository with Dockerfile.
- Link to CI Server and Application.
- Screenshots of Docker images, running containers, program outputs, CI/CD executions, highlighted features.
- Provide some description for each screenshot attached. The screenshots should be logically placed in the word document to provide a clear flow.
- Provide executed image/container commands (in plain text) so they may be replicated while evaluation.

TEST1

There are many benefits to using multi-stage build techniques when creating Dockerfiles. One such benefit is mitigating security risks, this is accomplished because the attack surface size of the image can be greatly reduced when the image no longer contains unnecessary files, packages, or binaries. It can also enhance caching on layers in previous build steps that no longer need to be clustered in a single RUN statement for optimal layering because the image is discarded and only those artifacts necessary are kept.

Multistage reference: https://docs.docker.com/develop/develop-images/multistage-build/

Create a docker image for an environment with the required tools for building and testing a java/react application. Use multi stage docker and optimise caching and structure in your final solution.

Optimise each stage in multistage dockerfile :

a. Load a base linux/unix alpine image

b. install git and build tools (node/npm/yarn/gradle or similar) as per application build requirement

c. Install and run a Jenkins/Bamboo instance in Docker. The image is freely available on Docker hub.

d. Install the unit testing tools like junit/jest or similar.

For screenshots:

- Show the multiple docker images with their optimised sizes.
- Provide link to the optimised dockerfile in Git.
- Show all the running containers and their functionality.
- Include images of running Jenkins/Bamboo with unit test tools.

TEST2

The following test will require you to do the following:

- Create a simple application which has a single "/healthcheck" endpoint.
- Containerise your application as a single deployable artifact, encapsulating all dependencies.
- Create a CI pipeline for your application

The application can be written in any programming language. We'd recommend using one the following: NodeJS or GoLang.

Please indicate your preferred programming language.

The application should be a simple, small, operable web-style API or service provider. It should implement the following:

- An endpoint which returns basic information about your application in JSON format which is generated; The following is expected:
  - Applications Version.
  - Description. ("static variable")
  - Last Commit SHA.

API Example Response

```
"myapplication": [
{
"version": "1.0",
"description" : "pre-interview technical test",
"lastcommitsha": "xxyyzz123456"
}
]
```

The application should have a CI pipeline that is executed when new code is commit and pushed, this pipeline should be comprehensive and cover aspects such as quality, and security; Travis or similar, for example.

Other things to consider as additions:

- Create tests or a test suite; the type of testing is up to you.
- Describe or demonstrate any risks associated with your application/deployment.
- Write a clear and understandable README which explains your application and its deployment steps.

The application code should be within a Github or Gitlab repository where we can review your source code and any configuration required for your project to execute. Please make sure the repository is public so it's viewable.

For screenshots:

- Include Git repository link
- Show successful CI build and deployment
- Show Containerised App.
- Run of Test suite and API response