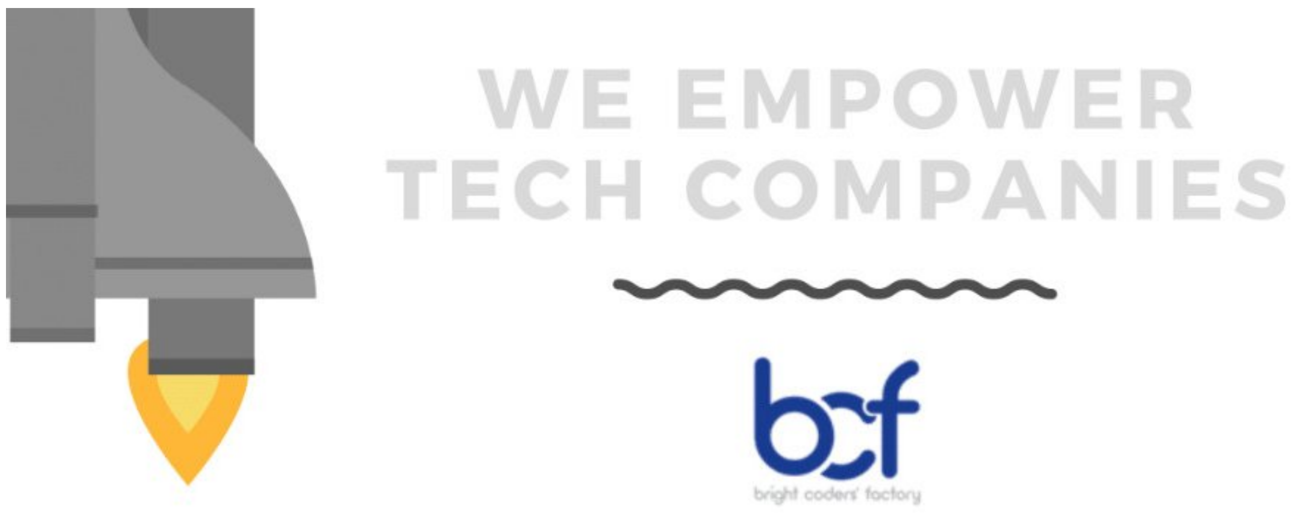

Introduction to C++ DevOps project

"Done is better than perfect."

Sheryl Sandberg

Week 1 - Building C++ Application

Tasks

1. Create a simple C++ application which will make use of `libyaml-cpp` library. Application should allow users to provide a name and to answer few questions. There should be an option to write all answers to a `*.yml` file.

EXAMPLE YAML OUTPUT:

```
name: Jan Kowalski
    family_members: 4
    income: 5000
name: Ania Nowak
    family_members: 2
    income: 2700
```

1. Prepare a `Makefile` which will build your application. Optionally you can generate a `Makefile` using `CMake` .

2. Create a *Docker* image based on *Ubuntu 18.04 LTS* which will contain all packages needed to build your application (without actual source code).

3. Create a container from the image created in step 3. Mount a directory with application source code to this container, build it and execute via interactive bash session.

Acceptance criteria

- C++ application source code with `Makefile` (or `CMakeLists.txt`) and `Dockerfile` published via any repository platform (Github, GitLab, Bitbucket, ...).
- Application source code is written according to C++ best practices.
- Included `README.md` file describing how to:
    - create Docker image,
    - start the container,
    - build the application in the container.

Week 2 - CI/CD Build System

Tasks

1. Write a `Vagrantfile` which will create a *Ubuntu 18.04 LTS* VM on *Virtualbox*.
2. Prepare *Ansible* playbook which will:
    - install *Docker*,
    - create *Docker Registry*,
    - start *Drone.io* instance and integrate it with your application repository,
    - start *Drone.io Docker Runner*.
3. Execute *Ansible* playbook on *Vagrant* VM and verify that you are able to login using credentials from repository platform of your choice and that you can activate your repository via *Drone.io* interface. Use free `ngrok` tool to make your machine reachable for incoming webhooks.
4. Import your *Docker* image into the registry.
5. Add *Drone.io* pipeline configuration to your application repository.
6. Push a commit to your repository and trigger a build on your CI/CD server.

Acceptance criteria

- `Vagrantfile` and *Ansible* playbook with all resource files published via any repository platform (Github, GitLab, Bitbucket, ...).
- *Ansible* playbook is written according to best practices described in the official documentation.

- *Ansible* playbook creates all services listed in task 2.
- With the help of included `README.md` file the reviewer should be able to:
    - recreate the CI/CD environment locally,
    - integrate CI/CD server with his fork of C++ application repository,
    - import / recreate *Docker* image used to build C++ application,
    - trigger the build via update on his fork of C++ application.