0.1 DevOps

Setup description

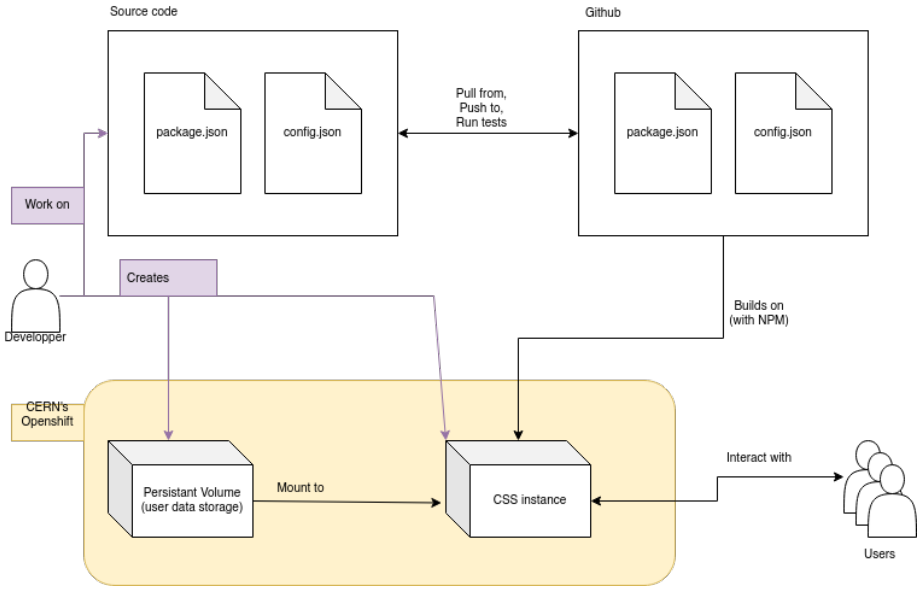

Figure 1: DevOps architecture diagram

As see in the figure 1, the global architecture for the development operations consists of 3 main entities:

- the source code, where the developer(s) can edit CERN's CSS recipe
- the remote repository, where the source code is published and reviewed, edited by other coders
- the server in itself, hosted inside CERN's Openshift infrastructure, is built of two virtual machines (VM):
  - The App VM: serving the CSS instance, this VM leverage OpenShift CI/CD to automatically build our CSS recipe from a git repository. If the repository contains a `package.json` file, OpenShift is smart enough to understand that it has to build an NPM package. Therefore, it will first try to build the package with the default command `npm build`. Once the package is built, it will try to run the package with the default command `npm run`. The former can be edited through the global variable `NPM_RUN`, which we can set when starting the VM build. In our case, we do need to modify use `NPM_RUN` to set the server correctly:
    * we set up the running port 8080 to match OpenShift configuration:
    * we set up the path where CSS should store users' data which is the path where we mount the persistent volume for user data (

see next section)
    * we set up the basename. For Cross-Origin Resource Sharing reasons and to display links with the server's full URL correctly, CSS needs to know on what is the base name of our server URL (i.e. it's domain name)
- Persistent Volume (PV): this VM acts as a simple file system that can be mounted to other VMs; it is where our CSS recipe stores users' data. Such a VM should be built only once, as rebuilding it would erase users data. A good improvement would be to apply backup regularly to prevent any data loss . Since the goal of this thesis is mainly experimental, it has not been judged necessary to deploy any backup process.

The choice of separating data and application has a few benefices. The main one is that it allows the creation and experimentation of different recipes hosted on different VM. Then, the persistent volume can be attached to a new application from the Openshift interface. It allows to test the application in the hosted environment (instead of locally) before publishing it to end-users. Secondly, we were not able to create a script to update a recipe within an App VM; whereas rebuilding the whole application can be done with a few lines of bash script . In addition, it permits to roll back to a previous build from the Openshift platform.

Bash script to deploy a CSS recipe to openshift from a git repository

```
# Fill the following variables

APP_NAME=""
PROJECT_NAME=""
GIT_REPO=""
# path where the PV is mounted on openshift
DATA_STORAGE_PATH=""
# required but doesnt need to be meaningfull
PROJECT_DESC="$PROJECT_NAME"
# default openshift URL
BASE_NAME="https://${APP_NAME}-${PROJECT_NAME}.app.cern.ch"

echo "remember to run sshuttle and login with oc"

# comment/uncomment the desired options:
# Create an app in a new project...
oc new-project "$PROJECT_NAME" \
  --description "$PROJECT_DESC"

# ... or form an existing one
# oc project $PROJECT_NAME
```

```
oc new-app "$GIT_REPO" \
  --name "$APP_NAME"

oc create route edge \
  --service=$APP_NAME \
  --insecure-policy='Redirect' \
  --port=8080

oc annotate route $APP_NAME \
  --overwrite haproxy.router.openshift.io/ip_whitelist=''

oc start-build $APP_NAME \
  --env=NPM_RUN="start -- -p 8080 -b $BASE_NAME \
    -f $DATA_STORAGE_PATH"
```