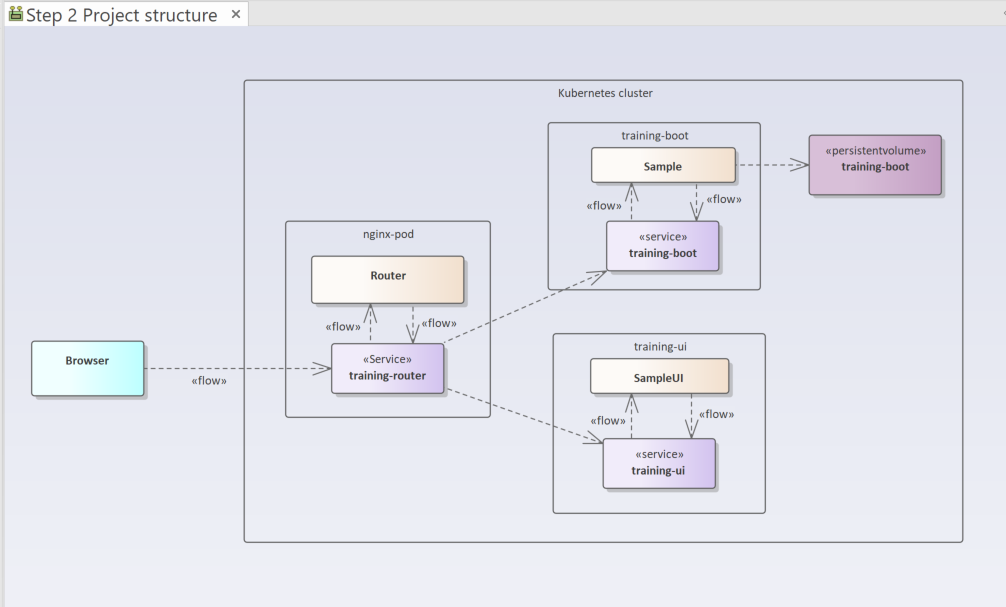
# Step 2 – Minimal Kubernetes

## Composition diagram

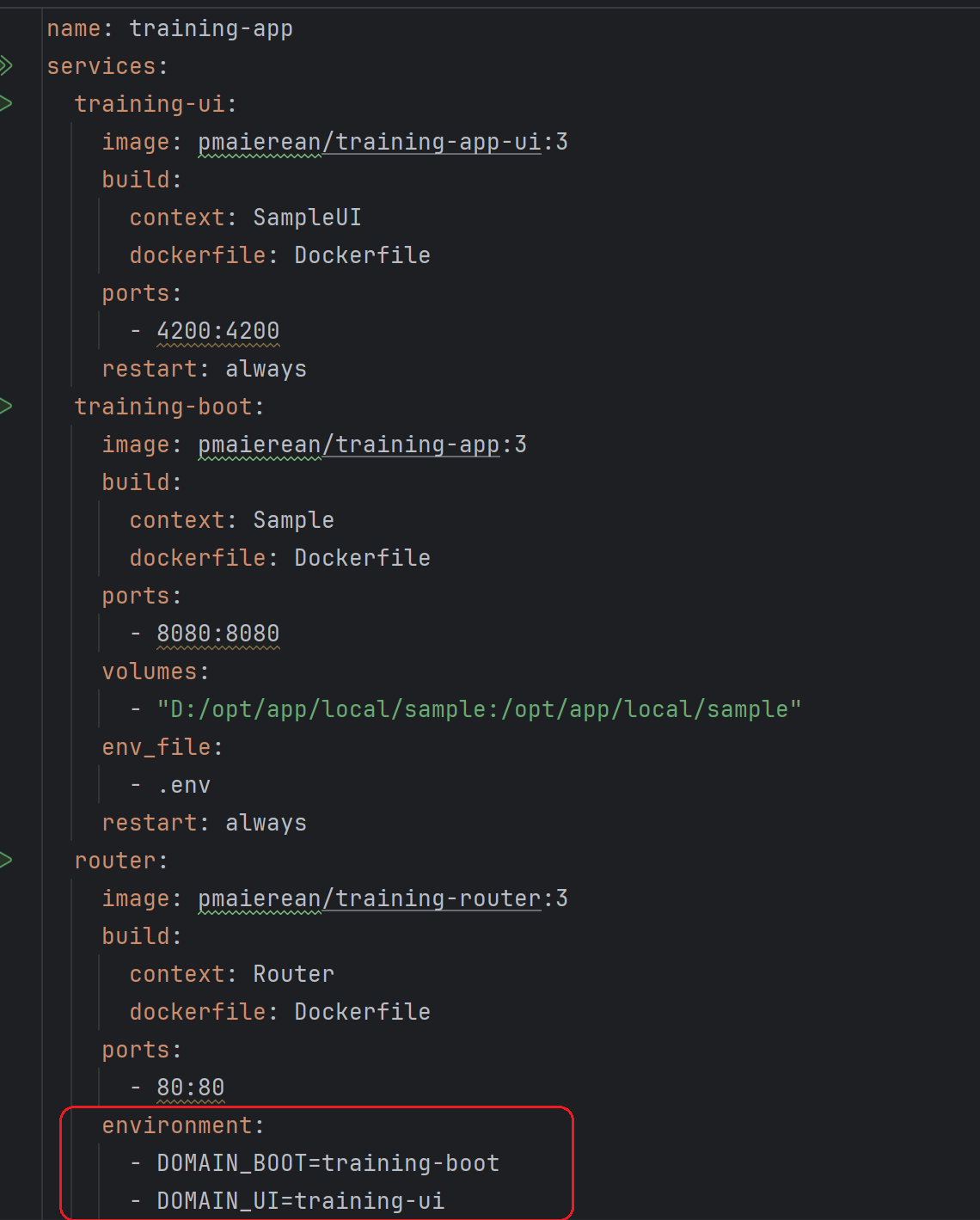
The goal of this step is to have a functioning [helm](https://helm.sh/) release that can be used to install the application in a Kubernetes cluster



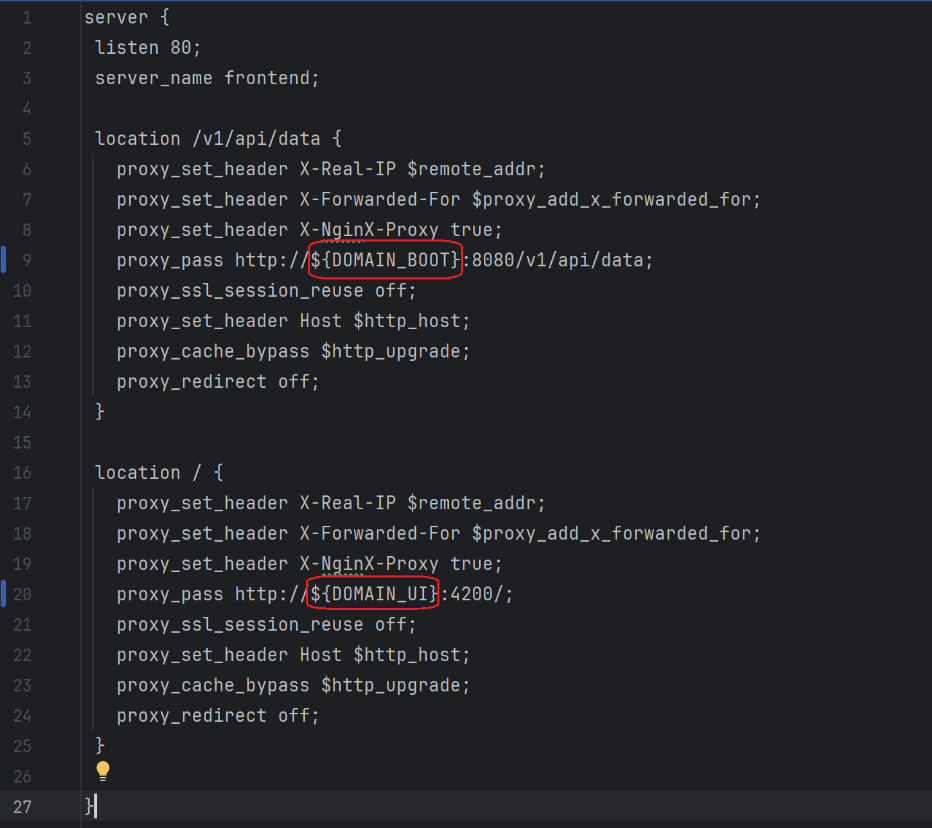
## Preliminary step

A preliminary step is to pass the domain names of the training boot and ui application in environment variables to nginx to allow for specific configurations of those links in Kubernetes.

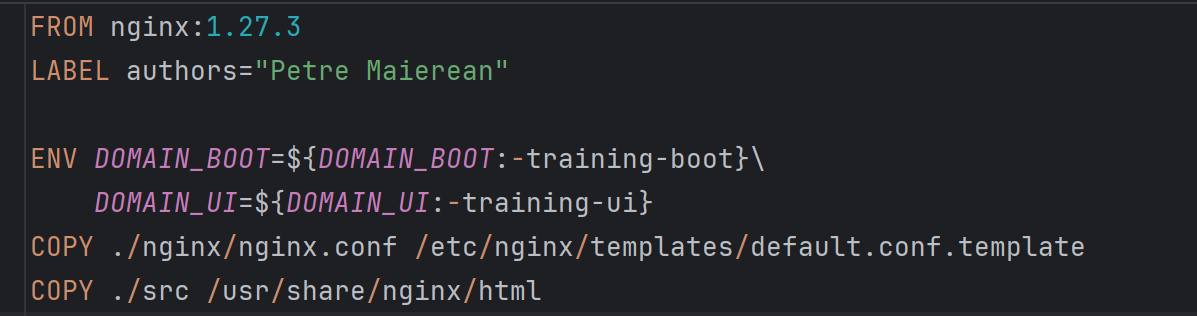
In docker-compose.yaml add:



Modify the configuration file for nginx to use environment variables rather than hardcoded domain names.



Modify Dockerfile for nginx to use environment variables in the configuration template.

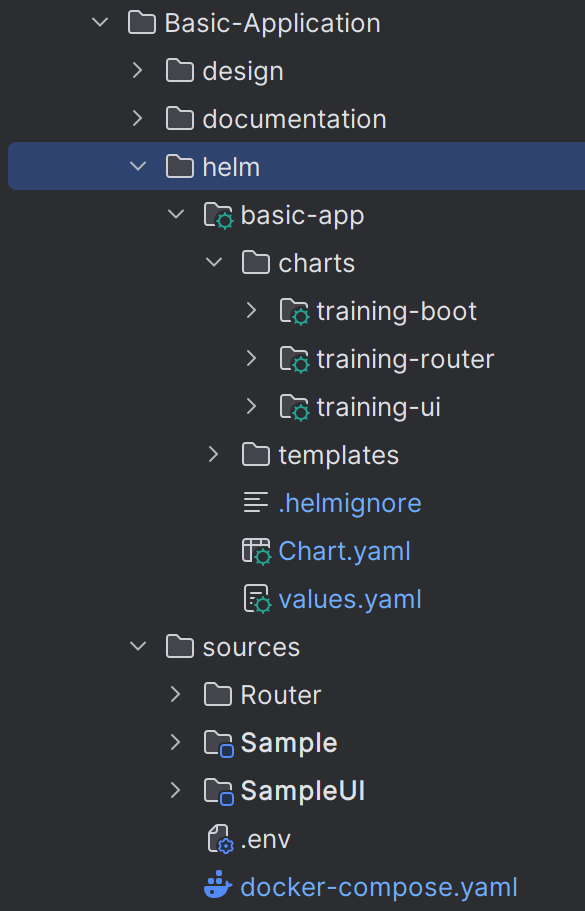


Note that placing the nginx.conf into the templates folder under the default.conf.template name instructs the application to compile that file and to move it into the expected location (‘/etc/nginx/conf.d/default.conf’) upon starting up.

Modify the angular.json to allow for a specific configuration to be used in the containerized setting.

## Using helm CLI to generate charts for those components

The [create](https://helm.sh/docs/helm/helm_create/) command can be used to generate Charts files. The structure of the helm project:



Note the importance of passing proper values for the two environment variables that the pod containing nginx uses to connect to the other two containers. Those variables must include the names of services associated with the training app and training-ui. Otherwise, the router container will fail.