**Creating a Production-Grade Workflow**

* **Development Workflow:**

**Diagram

Description automatically generated**

* **Flow specifics:**

**Diagram

Description automatically generated**

* **Dockers Purpose:**

**A picture containing application

Description automatically generated**

* **Project Generation:**

**Table

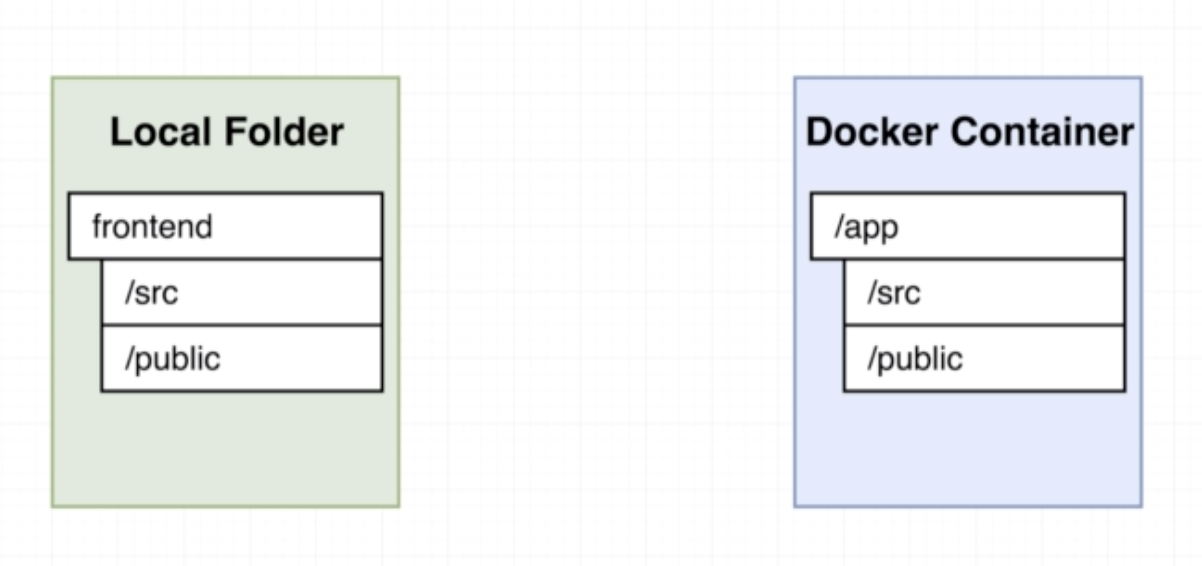
Description automatically generated**

* **Creating the Dev Dockerfile:**

Diagram

Description automatically generated

* **Docker Volumes:**

****

**Volumes:**

**Diagram

Description automatically generated**

Volumes are used to reference the file system which is external to the container to the running container.

This is similar to port mapping (port mapping maps the port of localhost to the port inside the container). In case of volumes we are referencing a file system.

Attaching volume using docke run command:

Diagram

Description automatically generated

To map a volume to the container, we need both the run time argument, if we provide just the second argument. It will end up in error.

docker run -it -p 3000:3000 -v $(pwd):/app c66d2c90683c

Reason for above error is, it doesn’t find the node\_modules folder.

docker run -it -p 3000:3000 -v /app/node\_modules -v $(pwd):/app image\_id

* **Running tests on container:**

Build the container using below command.

docker build -f Dockerfile.dev .

docker run image\_id npm run test

* **Live updating tests:**

Run the docker-compose up, get the id of the docker by running “docker ps”

Now use the exec command to run the test cases (live update)

docker exec -it container\_id npm run test

* **Docker compose for running tests:**

Create another service for running the test inside the docker-compose file as below. Now as it has a different mounting, it will run the test cases whenever there is a change.

Note: With this approach, we will not be able to get the console input control. Whatever you type inside the command, it will not accept anything.

* **Shortcomings on Testing:**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

We won’t get the input command even after using the attach, here is the problem behind it.

Docker container creates a different id for each process, in our case we are running npm run test.

But docker is not creating a single process for “npm run test”, it is creating a separate process for “npm” and another one for “run test”. So we were attaching it to the “npm” process not for the test process, because of this it wasn’t working.

Diagram

Description automatically generated

Diagram

Description automatically generated

Note: Whenever we use “docker attach” it will always attach to the primary process, not to the secondary process.

* **Need for Nginx:**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

Note: Whenever we run “npm run build” this will create the output for production.

For development server, it will have it won dev server which takes care of request handling, but for production, we will not have implicit “dev server”. For production we have to use an explicit server.

Diagram

Description automatically generated

* **Multi-Step Docker builds:**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

* **Implementing Multi-step builds:**