José Silva Follow
Jun 10, 2020 · 5 min read ★ · Listen

Vue.js runtime environment variables

One build, one artifact, dynamic configurations, infinite deploys

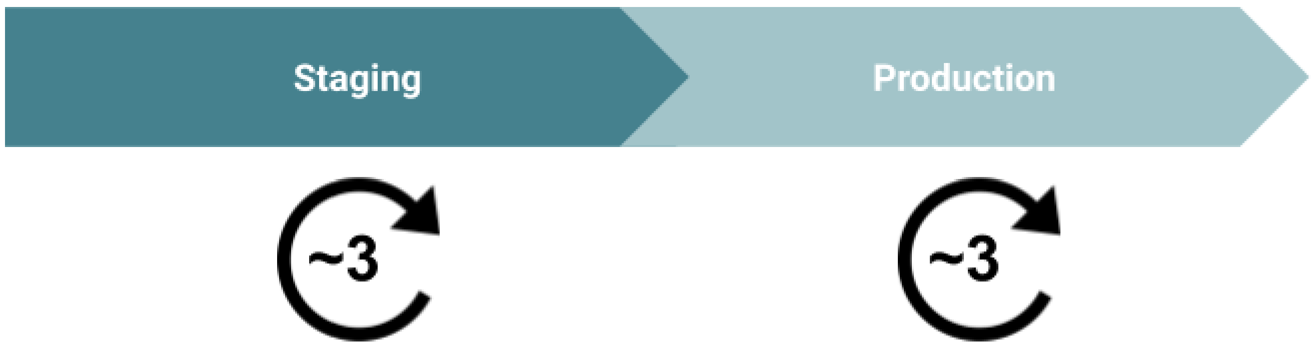
Photo by Aron Visuals on Unsplash

Frontend applications are a just bunch of static files tied with configurations provided

Let's imagine our application takes 3 minutes to build …

time to build the project

… and we have two environments.

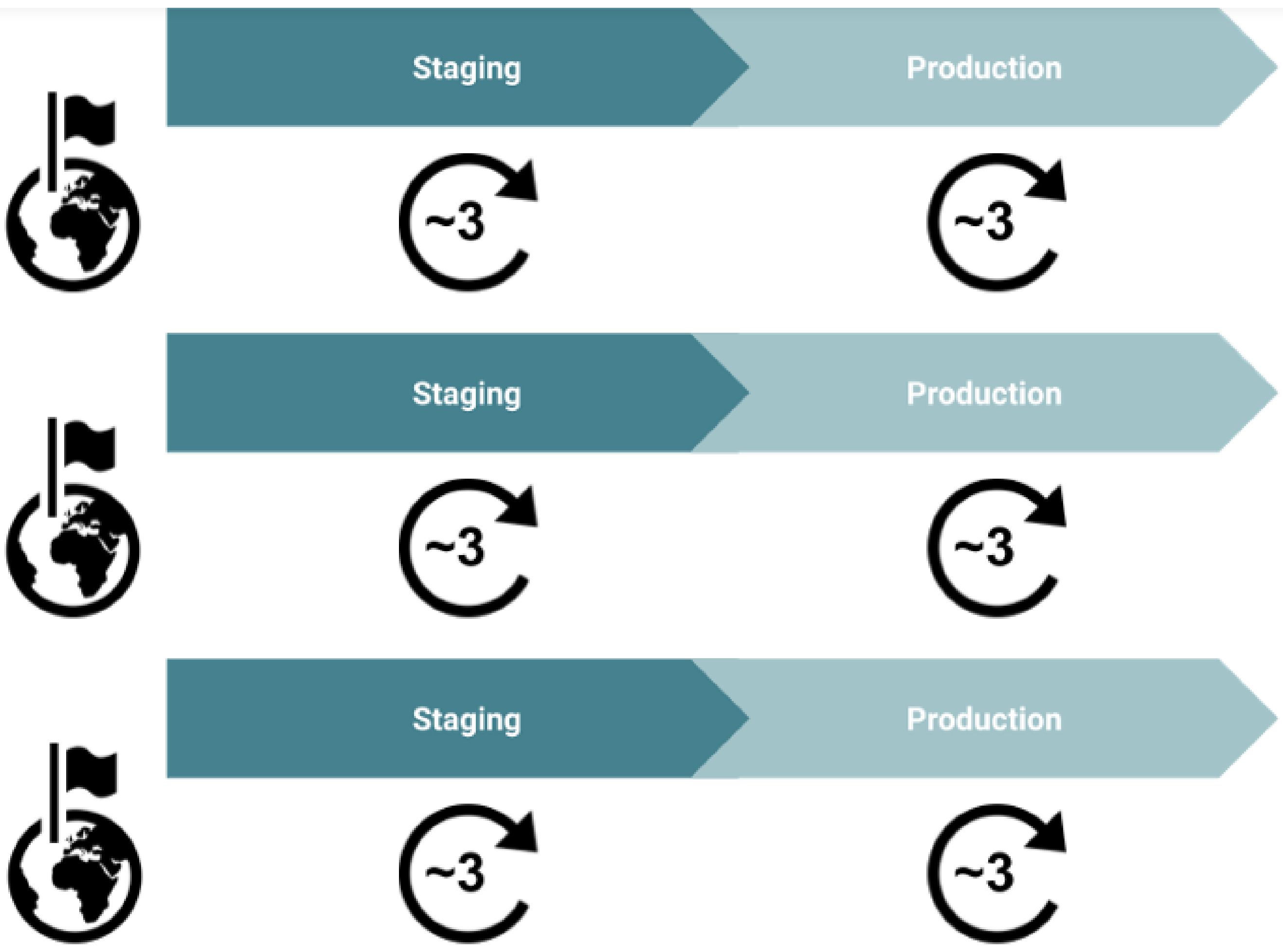


staging and production environments

Each environment has its own configurations which means that the project has to be built for each. Considering that we don't have to fix anything, we waste around 6 minutes just building the application before going live.

time to build the project in two environments

Now, let's assume that the business is going pretty well and our company decided to



staging and production environments by domain

We end up with different countries, different environments, different configurations, six builds, six artifacts, and 18 minutes of building time.

time to build the project for three countries, each with two environments

This solution is not scalable. We want to deliver fast, but every time we open a new country, we add at least more 6 minutes of building time. Once we are operating in 10 countries, we spend around **60 minutes** just building the application.

We must be able to create an artifact that can be served to every environment and every country independently of the configurations. It means that our goal is to have only 3 minutes of building time.

Implementation

1. Start by creating a new project using Vue CLI. You can use the default preset for this experiment.

2. Edit the file `public/index.html` to add a placeholder that will be replaced by the dynamic configurations.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script>
      // CONFIGURATIONS_PLACEHOLDER
    </script>
  ...
```

3. Create a file named `entrypoint.sh` in the root of the project.

```
#!/bin/sh

JSON_STRING='window.configs = { \
  "VUE_APP_VARIABLE_1":"'"${VUE_APP_VARIABLE_1}"'", \
  "VUE_APP_VARIABLE_2":"'"${VUE_APP_VARIABLE_2}"'" \
}'

sed -i "s@// CONFIGURATIONS_PLACEHOLDER@${JSON_STRING}@"
```

And make it executable:

```
chmod +x entrypoint.sh
```

Its function is to replace the placeholder in the `index.html` by the configurations, injecting them in the browser window.

4. Create a file named `src/utils/env.js` with the following utility function:

```
export default function getEnv(name) {
  return window?.configs?.[name] || process.env[name]
}
```

Wich allows us to easily get the value of the configuration. If it exists in `window.configs` (used in remote environments like staging or production) it will have priority over the `process.env` (used for development).

5. Replace the content of the `App.vue` file with the following:

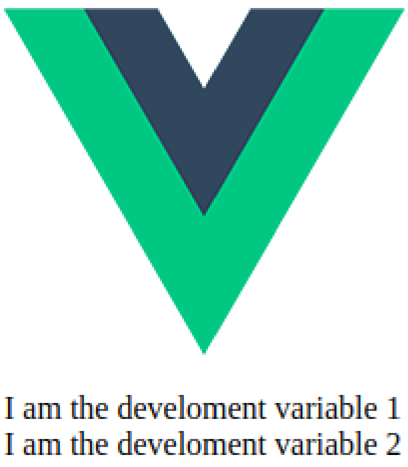
```
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <div>{{ variable1 }}</div>
    <div>{{ variable2 }}</div>
  </div>
</template>

<script>
import getEnv from '@/utils/env'

export default {
  name: 'App',
  data() {
    return {
      variable1: getEnv('VUE_APP_VARIABLE_1'),
      variable2: getEnv('VUE_APP_VARIABLE_2')
    }
```

Here, we import the nearly created utility and print our environment variables.

At this point, if we create the `.env.local` file, in the root of the project, with the values for the printed variables…

```
VUE_APP_VARIABLE_1='I am the develoment variable 1'
VUE_APP_VARIABLE_2='I am the develoment variable 2'
```

… and run the development server (yarn serve or npm run serve) we should see those values printed in the application (http://localhost:8080).



```
I am the develoment variable 1
I am the develoment variable 2
```

We are not using the `entrypint.sh` file yet because it is not useful to develop locally. Let's see where it shines ⭐.

6. Create two more files to simulate the staging and production variables.

`.env.staging.local`

```
VUE_APP_VARIABLE_1=I am the staging variable 1
VUE_APP_VARIABLE_2=I am the staging variable 2
```

`.env.production.local`

7. Create a file named `Dockerfile` in the root of the project with the following content:

```
# build stage
FROM node:lts-alpine as build-stage
WORKDIR /app
COPY package*.json ./
RUN yarn install
COPY . .
RUN yarn build

# production stage
FROM nginx:stable-alpine as production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html
COPY entrypoint.sh /usr/share/nginx/
ENTRYPOINT ["/usr/share/nginx/entrypoint.sh"]
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

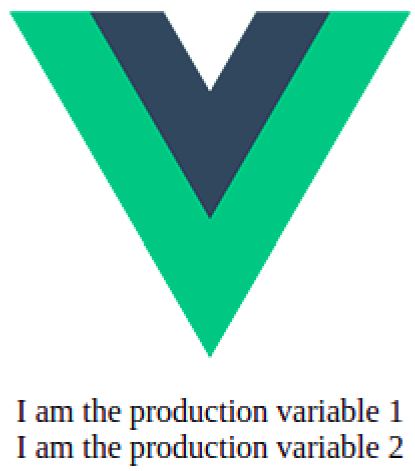
Now let's build the Docker image of our Vue.js app:

```
docker build -t vuejs-runtime-environment-variables .
```

Finally, let's run our Vue.js app in a Docker container (http://localhost:8080):

```
docker run -it -p 8080:80 --env-file=.env.staging.local --rm vuejs-
runtime-environment-variables
```

```
docker run -it -p 8080:80 --env-file=.env.production.local --rm
vuejs-runtime-environment-variables
```


I am the production variable 1
I am the production variable 2

You can also specify the value of the variable directly on the command, which will produce the same output.

```
docker run -it -p 8080:80 -e VUE_APP_VARIABLE_1='I am the production
variable 1' -e VUE_APP_VARIABLE_2='I am the production variable 2' -
-rm vuejs-runtime-environment-variables
```

Note

Time 🕛 is precious and I believe everyone enjoys seeing their work delivered as fast as possible 🚀. This kind of solution allows us to deliver faster and consequently makes us feel better.

I used Vue.js and Docker to achieve the intended goal but it can be done using a completely different set of tools.

Source code available in Github.