How To Pass Environment Info During Docker Builds

A detailed guide to Docker's ENV and ARG with examples

Bhargav Bachina Follow

Sep 16, 2019 · 6 min read

Photo by mostafa meraji on Unsplash

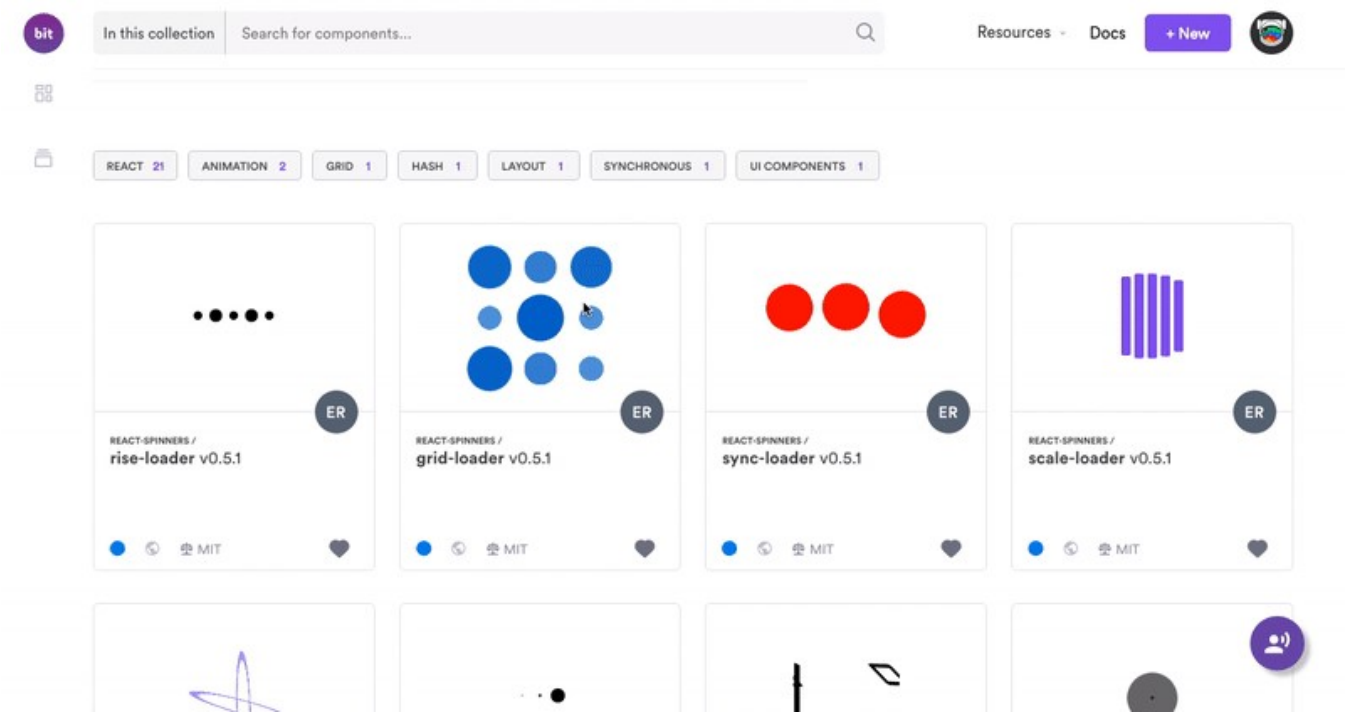
Sometimes we have to pass environment information to the running container or during the image build. We use both ENV and ARG instructions to achieve that. Let's

create one simple nodejs project with express as an example.

- *Example Project*
- *ARG Instruction*
- *ENV Instruction*
- *Difference Between ENV and ARG*
- *Pass Info with ENV and ARG*
- *Summary*
- *Conclusion*

Tip: Optimize teamwork by using the right tools for code-sharing

Use **Bit** to share, install and collaborate on individual JS modules and UI components. Stop wasting time configuring packages, managing multiple repositories or maintaining cumbersome monorepos.



Components with Bit: Easily share across projects as a team

Example Project

This is a simple node js express server serving at the port `3080` and we have a Dockerfile which builds the image.

```
//clone the project
git clone https://github.com/bbachi/docker-environment-read.git

//install and run the project
npm install
npm start
```

We can build the docker image with this command `docker build -t nodejs-sever .`

Once built we can see the images with this command `docker images`

```
Bhargavs-MacBook-Pro:docker-environment-read bhargavbachina$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nodejs-server       latest              752810026a0f        3 days ago          911MB
node                latest              fa0699832a2a        6 days ago          908MB
Bhargavs-MacBook-Pro:docker-environment-read bhargavbachina$
```

docker images

Let's create a container and run it with this command `docker run --name node-server -p 3080:3080 nodejs-server`

```
Bhargavs-MacBook-Pro:docker-environment-read bhargavbachina$ docker run --name node-server -p 3080:3080 nodejs-server
Server listening on the port::3080
```

ARG Instruction

ARG instruction defines a variable that can be passed at build time. Once it is defined in the Dockerfile you can pass with this flag `--build-arg` while building the image. We can have multiple ARG instruction in the Dockerfile. ARG is the only instruction that can precede the FROM instruction in the Dockerfile.

In the previous docker build, we have used `node:slim` as a base image. Let's pass the tag as a build argument. Here is the Dockerfile which uses ARG instruction for node version or tag.

```
# base image
ARG TAG=slim
FROM node:$TAG

# setting the work direcotry
WORKDIR /usr/src/app

# copy package.json
COPY ./package.json .

# install dependencies
RUN npm install

# COPY index.js
COPY ./index.js .

EXPOSE 3080

RUN node -v

CMD ["node","index.js"]
```

Dockerfile hosted with ❤ by **GitHub** **view raw**

Dockerfile With args

Here is the build command with the `--build-arg` flag.

```
// with slim tag
docker build -t nodejs-server -f Dockerfile.arg --build-arg TAG=slim
.
```

```
// with latest tag

docker build -t nodejs-server -f Dockerfile.arg --build-arg
TAG=latest .
```

If I run `docker images` command after the above two, I will have two images: one built with slim tag and another built with latest tag.

```
Bhargavs-MacBook-Pro:docker-environment-read bhargavbachina$ docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
nodejs-server-latest    latest              f8dc69a61e66        13 seconds ago      911MB
nodejs-server           latest              3ac71654dba6        4 hours ago         155MB
node                    slim                4828e4f6acb0        6 days ago          152MB
node                    latest              fa0699832a2a        6 days ago          908MB
```

docker images

An ARG declared before the FROM instruction can't be used after the FROM. If you want to use it you need to declare it without the value as below.

```
ARG VERSION=latest
FROM busybox:$VERSION
ARG VERSION
RUN echo $VERSION > image_version
```

The scope of this ARG instruction comes into picture only after it is declared in the Dockerfile. it goes out of scope once the build is complete. So the scope of the ARG instruction is limited to build stage.

If you want to pass multiple build arguments with `docker build` command you have to pass each argument with separate `– build-arg`.

```
docker build -t <image-name>:<tag> --build-arg <key1>=<value1>
--build-arg <key2>=<value2> .
```

ENV Instruction

ENV instruction sets the environment variable and this sets the environment for the subsequent build instructions. It takes two forms: one with a single variable `ENV <key> <value>` and another with multiple variables `ENV <key> =<avlue> <key> = <value>`. Here is an example

```
ENV PORT=3000
```

Let's have an example where we set the port as an environment variable. Here is the Docker file. We are passing port as an environment value.

```
# base image
FROM node:slim

# setting the work direcotry
WORKDIR /usr/src/app

ENV PORT=3000

# copy package.json
COPY ./package.json .

# install dependencies
RUN npm install

# COPY index.js
COPY ./index.js .

EXPOSE 3080

RUN node -v

CMD ["node","index.js"]
```

Dockerfile hosted with ❤ by **GitHub** **view raw**

Dockerfile.env

We have to read that in the nodejs in this way `process.env.PORT`. The following is the index.js file.

```
const express = require('express');
const app = express(),
      port = process.env.PORT || 3080;

app.get('/', (req,res) => {
    res.send('App Works !!!');
});

app.get('/time', (req,res) => {
    res.send(new Date());
});

app.listen(port, () => {
    console.log(`Server listening on the port::${port}`);
});
```

index.js hosted with ❤ by GitHub view raw

index.js file with reading environment variables

Let's build the image, run the container and test it.

```
// build the image
docker build -t nodejs-server-env -f Dockerfile.env .

// run the container
docker run -d --name node-server-env -p 3000:3000 nodejs-server-env
```

We can now access the app at the port **3000**.

the app is running on port 3000

You can pass the environment variable with the run command as well. Let's use the following command.

```
// run command with env
docker run -d --name node-server-env1 --env PORT=3070 -p 3070:3070
nodejs-server-env
```

We can now access the app at the port **3070**.

the app is running on port 3070

Difference Between ENV and ARG

It is evident from the above that ARG instruction can only be accessible during build time and not available when we are running the container. It can be passed with the build command with the flag `--build-arg`.

ENV instruction is available during build time and also when running the container with the flag `--env`. But, we can't pass ENV instruction while building the image.

What if we want to pass PORT during the build time instead and use that as an environment variable?

Pass Info with ENV and ARG

Let's use both ENV and ARG together to pass environment info during the build time and change that while running the container as well.

Here is the Dockerfile using both ARG and ENV together to take the PORT as the build argument.

```
# base image
ARG TAG=slim
FROM node:$TAG

ARG PORT_ARG=3080

# setting the work direcotry
WORKDIR /usr/src/app

ENV PORT=$PORT_ARG

# copy package.json
COPY ./package.json .

# install dependencies
RUN npm install

# COPY index.js
COPY ./index.js .

EXPOSE 3080

RUN node -v

CMD ["node","index.js"]
```

Dockerfile hosted with ❤ by **GitHub** **view raw**

Dockerfile

```
// docker build
docker build -t nodejs-server-both1 --build-arg PORT_ARG=3090 -f
Dockerfile.both .

//run the conatiner
docker run -d --name node-server-env -p 3000:3000 nodejs-server-env
```

Here is the app running on the port **3090**.

The app running on the port 3090

In this way, we can pass as many arguments as possible to pass as environment variables while building the image.

Summary

- ARG instruction defines a variable that can be passed at build time and we can pass inline with `--build-arg`
- ENV instruction sets the environment variable and this sets the environment for the subsequent build instructions.
- The limitation of ENV instruction is that we can't pass inline while building the image.
- The limitation of ARG instruction is that it can't persist after building the image.
- We need to use both ENV and ARG to pass the environment info inline while building the image.

Conclusion

Almost all the applications need their environment information to read from somewhere dynamically. We can pass environment info to the docker images using both ARG and ENV instructions as above. We can pass as many env variables as possible in this way.

Learn More

Let Everyone In Your Company Share Your Reusable Components

Share your existing technology in a visual way to help R&D, Product, Marketing and everyone else build together.

blog.bitsrc.io

Meet Bit's Component Cloud

Share atomic code in the cloud to build modular software without limits.

blog.bitsrc.io

How to Share React UI Components between Projects and Apps

A simple guide to help you organize, share and sync React components between your team's apps.

blog.bitsrc.io

Get an email whenever Bhargav Bachina publishes.

Subscribe

Docker Software Development Web Development Programming Software Engineering