Title: How to build Docker Images with Dockerfile behind HTTP_PROXY by Jenkins?

Post Body:

Building Docker images works in a desktop without a problem. Installing Node.js NPM dependencies work as usual. However, when using a continuous integration server such as Jenkins that is hosted behind a corporate proxy, the build Docker Images fail.

### Node.js NPM Dependencies

While building Node.js pacakges, the command **npm install** fails when it cannot connect to GIT while cloning GIT dependencies.

```
e1ce5e8407d1: Already exists Status: Image is up to date for node:0.10.33  ---> e1ce5e8407d1 Step 1 : RUN mkdir -p /usr/src/ap
```

### Java Maven, Ruby, Go Docker Images with Dependencies

The same occurs when building Java, Ruby or Go containers, where dependencies are located in repository servers across your corporate Proxy server.

Knowing that you can configure Docker with HTTP_PROXY environment variable, how to properly configure Docker to properly build images in CI environments?

Accepted Answer:

Note: Docker 1.9 *might* help solve this:

- 'Issue 14634': **Builder - Build-time argument passing** (e.g., `HTTP_PROXY`)
- 'PR 15182': **Support for passing build-time variables in build context**

Usage (proposed):

```
docker build --build-arg http_proxy=http://my.proxy.url  --build-arg foo=bar <<MARK FROM busybox RUN <command that need http_p
```

Highest Rated Answer:

Docker has multiple ways to set proxies that take effect at different times.

---

If your `docker build` has to **retrieve a base image through a proxy**, you'll want to specify `build-args`:

```
docker build --build-arg HTTP_PROXY=$http_proxy \ --build-arg HTTPS_PROXY=$http_proxy --build-arg NO_PROXY="$no_proxy" \ --bui
```

where `$http_proxy` and `$no_proxy` were set in my bashrc. I used both `HTTP_PROXY` and `http_proxy` because different utilities will check different variables (`curl` checks both, `wget` only checks the lowercase ones, etc).

---

If your `docker build` has a **`RUN curl/wget/etc` command that has to go through the proxy**, you'll need to specify an environment variable inside your docker image:

```
ENV https_proxy=http://proxy-us02.company.com:8080 ENV http_proxy=http://proxy-us02.company.com:8080 ENV HTTP_PROXY=http://pro
```

If you don't want this environment variable inside your image at runtime, you can remove all these at the end:

```
RUN unset http_proxy https_proxy no_proxy HTTP_PROXY HTTPS_PROXY NO_PROXY
```