Title: Passing Secrets to Docker

Post Body:

I have an artifact repository (set up on GitLab) for which downloading artifacts is protected by a secret token. When I try and build my project, which relies on downloading artifacts, in a Docker image it cannot download them because it does not have a settings file with the secret token to access the repository server.

My Setup

The server in my personal, local maven `settings.xml`

```
    <server>       <id>my-gitlab-maven-registry</id>       <configuration>         <httpHeaders>           <property>
```

The repository in my project's `pom.xml`

```
    <repository>       <snapshots>         <enabled>true</enabled>       </snapshots>       <id>my-gitlab-maven-registry</id>
```

My project's (simplified) `Dockerfile` (fails when it runs `mvn package`)

```
FROM maven:3.6.3-openjdk-15-slim AS build  COPY src /home/app/src COPY pom.xml /home/app  RUN mvn -f /home/app/pom.xml clean p
```

My project's (simplified) `docker-compose.yml`

```
version: "3" services:   my-app:     container_name: my-app     build: .     ports:       - 8080:8080
```

I know a possible solution is to copy my local settings (with an absolute path) into the Docker image, but considering I want this project to be used by multiple people on multiple computers, I want a way to generalize passing the settings or the secret token to the docker image; so I basically want anyone who has the secret token to be able to do some sort of simple configuration to be able to build the Docker image -- I want to avoid the simplest solution of actually uploading the secret token to version control.

My Question

So what are some possible ways I might accomplish a general solution to passing a user's maven settings or the secret token to a docker image?

My Thoughts on Solutions

One potential option would be to use an environment variable to hold the secret token, similar to what's used for GitLab CI:

```
<!-- This environment variable is used in GitLab CI, but I could use a different name for the variable. --> <value>${env.CI_JO
```

But then using an environment variable, I'm not really sure how to pass an environment variable to a service when running `docker-compose up` without actually hardcoding the environment variable within the `docker-compose.yml`.

Accepted Answer:

I would go with the env variable, as you already said

```
<value>${env.YOUR_TOKEN}</value>
```

In your Dockerfile

```
ARG YOUR_TOKEN RUN mvn package
```

Then in your docker-compose.yml

```
version: "3" services:   my-app:     container_name: my-app     build:       context: .       args:         YOUR_TOKEN: $YOUR_
```

Now this should work

```
export YOUR_TOKEN=abc123 docker-compose build
```

BUT You should not upload this build image, as it contains the token in its history.
If you have a multistage build and the maven step is your build step, which will be thrown away, it's ok.

A more elegant solution would be buildkit secrets. But I don't know if they already work with docker compose. See https://docs.docker.com/develop/develop-images/build_enhancements/ and How do you use Docker build secrets with Docker Compose?

If you need the variable at runtime (docker-compose up) and not buildtime (docker-compose build) you can do it like this:

```
version: "3" services:   my-app:     environment:       YOUR_TOKEN: $YOUR_TOKEN     ports:       - 8080:8080
```

Then

```
export YOUR_TOKEN=abc123 docker-compose up
```

Highest Rated Answer: None