Title: Passing Environment Variables With Docker to Spring Boot Application Not Working

Post Body:

I'm trying to pass environment variables to a Spring Boot application but it doesn't seem to be working.

Docker run command:

```
docker run my-image -e TEST_VAR='testing'
```

Spring Boot main():

```
@SpringBootApplication public class MyApplication {      public static void main(String[] args) {         ApplicationContext c
```

Output:

```
TEST_VAR: null
```

How can I successfully pass environmental variables with Docker?

Accepted Answer:

[`docker run`] does mention

> Additionally, the operator can set any environment variable in the container by using one or more -e flags, even overriding those mentioned above, or already defined by the developer with a Dockerfile ENV.
>
> If the operator names an environment variable without specifying a value, then the current value of the named variable is propagated into the container's environment:

But the example is:

```
docker run -e "deep=purple" -e today --rm alpine env
```

So check your quotes, and parameters order (as BMitch observes, any parameter done *after* the image name would be passed as `CMD` to your image ENTRYPOINT command, which is not what you want):

```
docker run -e "TEST_VAR=testing" my-image
```

Highest Rated Answer:

The order of options is important on the docker command line. There are flags you can pass before the run command, flags you can pass to the run command, and args that get passed to the image as your command to run. In your example:

```
docker run my-image -e TEST_VAR='testing'
```

The `-e TEST_VAR='testing'` gets passed on as the new value of `CMD` for the container to run (or argument to your entrypoint).

By reordering your command, you will tell `run` to pass the environment variable to the container as desired:

```
docker run -e TEST_VAR='testing' my-image
```