Title: How to access the Host's machine's localhost 127.0.0.1 from docker container

Post Body:

I hosted Git daemon on local host i.e. `'/usr/bin/git daemon --listen=127.0.0.1 --base-path=/opt'` as a `systemd` service and I am trying to access it from docker container. I didn't mentioned the port because I don't want to expose the port to outside network.

Dockerfile:

```
RUN git clone git://127.0.0.1/repo/ repo_dir
```

But its not working, its looks like inside container its trying to connect localhost of container.

So How to connect connect localhost of Host machine from Docker container?

Accepted Answer:

> its not working, its looks like inside container its trying to connect localhost of container.

Yes, that is the all idea behind the isolation provided by container, even `docker build` (which builds one container per Dockerfile line, and commits it in an intermediate image).

As commented by the OP dhairya, and mentioned in the documentation I referred in my comments: 'Docker container networking' and `docker build`, you can set to **host** the networking mode for the `RUN` instructions during build: then `localhost` would refer to the host localhost.

```
docker build --network='host'
```

This is since API 1.25+ only, in docker v.1.13.0-rc5 (January 2017)

> `POST /build` accepts `networkmode` parameter to specify network used during build.

But if you don't need Git in your actual built image (for its main process to run), it would be easier to

- clone the repo locally (directly on the host) *before* the docker build.
  You need to clone it where your Dockerfile is, as it must be relative to the source directory that is being built (the context of the build).

  use a `COPY` directive in the Dockerfile, to copy the host folder representing the checked out Git repo.
  Note: add `.git/`: to your `.dockerignore` (file in the same folder as your Dockerfile), in order to *not* copy the `repo_dir/.git` folder of that cloned repo, if you don't have git in your target image.

  ```
  COPY repo_dir .
  ```

(Here '`.`' represent the current `WORKDIR` within the image being built)

Highest Rated Answer: None