Adapting Jenkins container

Dockerfile

```
FROM jenkins/jenkins
USER root
RUN apt-get -y update && \
apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common && \
curl -fsSL [https://download.docker.com/linux/ubuntu/gpg](https://download.docker.com/linux/ubuntu/gpg) |
apt-key add - && \
add-apt-repository \
“deb [arch=amd64] [https://download.docker.com/linux/$(.](https://download.docker.com/linux/$(.) /etc/os-
release; echo “$ID”) \
$(lsb_release -cs) \
stable” && \
apt-get update && \
apt-get -y install docker-ce docker-ce-cli containerd.io
RUN curl -L “[https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname]
(https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname) -s)-$(uname -m)” -o
/usr/local/bin/docker-compose && \
chmod +x /usr/local/bin/docker-compose && \
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
USER jenkins
```

docker-compose.yml

```
version: '3.1'
services:
jenkins:
build:
context: ./
restart: unless-stopped
volumes:
- ${HOST_DOCKER}:/var/run/docker.sock
- ${HOST_JENKINS_DATA}:/var/jenkins_home
ports:
- "${HOST_WWW}:8080"
- "${HOST_OTHER}:50000"
```

.env

```
HOST_WWW=8180
HOST_OTHER=8181
HOST_DOCKER=/var/run/docker.sock
HOST_JENKINS_DATA=/srv/www/jenkins
```

Solving docker.sock permission denied

There is docker user’s group with required permissions, so we add jenkins user to that group and restart docker:

```
usermod -aG docker jenkins
sudo service docker restart
```

This time everything works on the host, but we still get the same error when we run `docker command` on the container. Why?

Generally, container and host are two separate environments. All users created in both are also separate. If we want to have the same users, we need to configure them correctly. We already have jenkins user on host and container, but they are different users. A user is identified by uid, not username. So we need to make sure that both users have the same uid.
First, we need to verify uid on host:

> id -u jenkins
> 1004

As you see, we received uid — 1004. We’ll change our Dockerfile to reconfigure our jenkins user on the container. We do it by adding two lines in Dockerfile. First at the top of the file (after base image declaration) to define accepted arguments for build:

ARG HOST_UID=1004

Second, before we switch to jenkins user, to change user uid:

```
RUN usermod -u $HOST_UID jenkins
```

Next, we change our build definition in docker-compose.yml:

```
services:
jenkins:
build:
context: ./
args:
HOST_UID: ${HOST_UID}
```

In the end, we add a new line in .env:

```
HOST_UID=1004
```

Now our uid is correct, but we still have a problem — why? Both host and container see our user as the same (they are still different users, but we cheated the user verification). Permissions are set for a user group, not a specific user. What more jenkins user in the container is not part of that group. We talk about the docker user's group. That group in host and containers have the same name but different group id (the same problem as with user). First, we need to know gid for docker group on host:

> ```
> getent group | grep docker
> docker:x:999:jenkins
> ```

The third part of the data contains gid — 999 in our case. Next, we adjust our Dockerfile, to make sure that the docker group has the same id on host and container and that jenkins user is in that group. First, we add a new argument to build:

```
ARG HOST_GID=999
```

Next, we change the group and modify user, before we switch to jenkins user:

```
RUN groupmod -g $HOST_GID docker
RUN usermod -aG docker jenkins
```

Again we change build definition in docker-compose.yml:

```
services:
jenkins:
build:
context: ./
args:
HOST_UID: ${HOST_UID}
HOST_GID: ${HOST_GID}
```

We also add correct variable in .env file:

```
HOST_GID=999
```

Now we build a docker image again and run the container .

If you can't run a container with jenkins due permissions for Jenkins files, you need to adjust the files' ownership on the host. As you remembered we messed a little bit with users ids and that may be an issue here. In our example, we need to run on host:

```
sudo chgrp -R /srv/www/jenkins jenkins
sudo chown -R /srv/www/jenkins jenkins
```

That would be all. With this configuration, we can run Jenkins in docker and use docker for builds.