[Docker Networking Disabled: WARNING: IPv4 forwarding is disabled. Networking will not work](https://stackoverflow.com/questions/41453263/docker-networking-disabled-warning-ipv4-forwarding-is-disabled-networking-wil)

Containers in a host "suddenly" loses connection to outside-world containers. However, some hosts were refreshed and suddenly we had the following situation:

1. The host can communicate with other hosts.
2. Containers running in the host cannot communicate with other hosts.

I added the following to **/etc/sysctl.conf**:

net.ipv4.ip\_forward=1

I then restarted the network service and validated the setting:

[root@pprdespap322 deploy]# systemctl restart network

[root@pprdespap322 deploy]# sysctl net.ipv4.ip\_forward

net.ipv4.ip\_forward = 1

# [Why am I getting “Cannot connect to the Docker daemon” when the daemon is running?](https://unix.stackexchange.com/questions/252684/why-am-i-getting-cannot-connect-to-the-docker-daemon-when-the-daemon-is-runnin)

No matter if you're on Arch or another distribution like Fedora or Ubuntu, Docker uses a socket file to communicate. When you run docker commands, it uses this socket to talk to the Docker daemon. Of course, the daemon must be running (and it's often disabled by default), but if your user can't access the socket, it wouldn't be able to communicate with the daemon either.

As mentioned above, the daemon may be disabled by default. If you want to use Docker, the daemon must be running.

Enable it (so it will be started on boot):

# systemctl enable docker

Start it now (or reboot):

# systemctl start docker

Now, by default (if the docker group is missing), the Docker socket is owned by root:

# ls -la /var/run/docker.sock

srw-rw---- 1 root root 0 Apr 28 17:22 /var/run/docker.sock

This is why a regular user is not able to talk to the docker daemon. A regular user does not have sufficient permissions to access the socket. It's not able to reach the daemon, so it assumes it's not running and shows this error: Cannot connect to the Docker daemon. Is the docker daemon running on this host?

This is why many people simply start all Docker commands as root, using sudo. But as described in the other answer, Docker has its own mechanism for that, so using sudo is not necessary.

Ideally, a group called docker is created when installing Docker. However, if that group does not exist when the daemon is started, the socket file is owned by root.

In some cases, that group used to have a different name, like [dockerroot on Fedora](https://bugzilla.redhat.com/show_bug.cgi?id=1195525). Check grep docker /etc/group to see if there is such a group on your system. If you're already using that group (your user is in it), you would need to configure Docker to use it:

In /etc/sysconfig/docker, add -G dockerroot (note: it's a workaround, not the best solution):

OPTIONS='--selinux-enabled -G dockerroot'

After restarting the daemon, your user will be able to access the socket:

# systemctl restart docker

# ls -la /var/run/docker.sock

srw-rw---- 1 root dockerroot 0 Apr 28 17:32 /var/run/docker.sock

Otherwise, the official way would be to use the group called docker. If it exists, Docker will automatically use it, i.e., set the socket's group to that group. If it doesn't exist, all you need to do is create it and restart the daemon:

# groupadd docker

# systemctl restart docker

The socket file will be owned by that group:

# srw-rw---- 1 root docker 0 Apr 28 17:42 /var/run/docker.sock

Your user must be in the docker group to be able to access the socket:

# usermod -aG docker (user)

You may have to log out and log back in again (or su - (user)), run id to see if you're in the group.

You can then use Docker without sudo/root:

$ docker version --format '{{.Server.Version}}'

1.9.1