It is possible to have 2 containers/services or more described in 1 .yaml file.

For example:

version: "3"

services:

ngnx: <-name of the container

image: nginx

ports:

- 280:80

alpine: <-name of the container

image: alpine

entrypoint: "ping ngnx" <-the entrypoint is that alpine will run command ‘ping’ for ngnx container

It is important that ngnx is the **name of the container** in the same file. One of the advantages of using a .yaml file is that when we create a few containers (services) in one file, the docker will establish the network connection between them by default, so we don't need to do it manually. That is allowing containers to interact between them. so they know each other and that's why we can provide an entrypoint in which the second container executes instruction that is concerning the first container.

Using this script the docker will start the containers in parallel and it might cause some issues. Sometimes we might want different behavior. It might be that we want the containers to run in a specific way, for example:first-ngnx, second-alpine, because of the logic of the program we don't need alpine to run before ngnx container is started, it doesn't have any sense for alpine to make a calls for ngnx if it doesn't even exist. In order to do it there is instruction: depends\_on: .

For example:

version: "3"

services:

ngnx:

image: nginx

ports:

- 280:80

alpine:

image: alpine

entrypoint: "ping ngnx"

**depends\_on:** <-using this field we can give the name/names of the container that we are

**-ngnx** depend on, it means that till that container will not be created (ngnx) and

run. The alpine container will wait and will start only after ngnx.

**To take away:**

1)-It is possible to run multiple containers in one ,yaml file

2)-There is a network connection between containers in one ,yaml file by default.

3)-Using depend\_on instruction we control the flow of creation of the containers.

There is a limitation in using depend\_on instruction. It controls only the **start** of the containers; it will not wait for a specific container to do some instructions mentioned in it. So it is simply a container starting order. One more time: it is just starting containers in a specific way, it is not about waiting until one container starts, finishes its instructions and then another one starts… What is happening inside of the container docker will not know, the only one it knows that it should start the containers in that way (described in depend\_on instruction)