Title: How can I make docker-compose bind the containers only on defined network instead of 0.0.0.0?

Post Body:

In recent versions `docker-compose` automatically creates a new network for the services it creates. Basically, every `docker-compose` setup is getting its own IP range, so that in theory I could call my services on the network's IP address with the predefined ports. This is great when developing multiple projects at the same time, since there is then no need to change the ports in `docker-compose.yml` (i.e. I can run multiple `nginx` projects at the same time on port 8080 on different interfaces)

However, this does not work as intended: every exposed port is still exposed on 0.0.0.0 and thus there are port conflicts with multiple projects. It is possible to put the bind IP into `docker-compose.yml`, however this is a killer for portability -- not every developer on the team uses the same OS or works on the same projects, therefore it's not clear which IP to configure.

It's be great to define the IP to bind the containers to in terms of the network created for this particular project. `docker-compose` should both know which network it created as well as its IP, so this shouldn't be a problem, however I couldn't find an easy way to do it. Is there a way or is this something yet to be implemented?

EDIT: An example of a port conflict: imagine two projects, each with an application server running on port 8080 and a MySQL database running on port 3306, both respectively exposed as '8080:8080' and '3306:3306'. Running the first one with `docker-compose` creates a network called something like `app1_network` with an IP range of 172.18.0.0/16. Every exposed port is exposed on 0.0.0.0, i.e. on 127.0.0.1, on the WAN address, on the default bridge (172.17.0.0/16) and also on the 172.18.0.0/16. In this case I can reach my application server of all of 127.0.0.1:8080, 172.17.0.1:8080, 172.18.0.1:8080 and als on `$WAN_IP:8080`. If I start the second application now, it starts a second network `app2_network` 172.19.0.0/16, but still tries to bind every exposed port on all interfaces. Those ports are of course already taken (except for 172.19.0.1). If there had been a possibility to restrict each application to its network, application 1 would have available at 172.18.0.1:8080 and the second at 172.19.0.1:8080 and I wouldn't need to change port mappings to 8081 and 3307 respectively to run both applications at the same time.

Accepted Answer: None

Highest Rated Answer:

In your service configuration, in docker-compose.yml:

```
ports:  - "127.0.0.1:8001:8001"
```

Reference: https://github.com/compose-spec/compose-spec/blob/master/spec.md#ports