Title: Understanding ports and links in docker compose

Post Body:

From my understanding of docker compose / fig, creating a link between two services/images is one main reason if you do not want to exposes ports to others.

like here **db** does not expose any ports and is only linked:

```
web:   build: .   links:    - db   ports:    - '8000:8000'    db:   image: postgres
```

Does **web** thinks **db** runs on its localhost? Would i connect from a script/program in **web** to localhost:5432 (standard port from postgresql) to get a database connection?

And if this is correct, how can you change port 5432 to 6432, without exposing? would i just run postgresql on a different port?

Update:

useful links after some input:

http://docs.docker.com/userguide/dockerlinks/

https://docs.docker.com/compose/yml/#links

Accepted Answer:

web thinks **db** runs on the host pointed to by the env variable DOCKER_DB or something like that. Your services should point to that variable (host), not localhost.

The db container exposes ports (via EXPOSE) to its linked containers, again in variables. You can run the db on whatever port you want, as long as it's EXPOSEd.

Highest Rated Answer:

`docker-compose` / `fig` is mainly a workhorse for starting/managing multiple images at once which are somehow dependent on one another.

To fully understand the **links** between containers you should know that there is a host entry created, usually inside `/etc/hosts` file, which maps that container hostname within docker managed network to a specific ip address. So if you want to access *postgres* db you have to point to the `db` hostname instead of `localhost`.

Regarding exposed ports, most of the images already have some ports exposed, so it might be the case that you just use an image and that port is exposed, you can always override it to something else.

Update in Docker Compose 1.6.0+

> By default Compose sets up a single network for your app. Each container for a service joins the default network and is both reachable by other containers on that network, and discoverable by them at a hostname identical to the container name. source

You can define links between containers only when you want to define an alias to a container, like so:

```
version: '2' services:   web:     build: .     links:       - 'db:database'   db:     image: postgres
```