Title: Use shared database docker container in microservice architecture

Post Body:

From the count of questions tagged with `docker` i assume StackOverflow is the right place to ask (instead of e.g. DevOps), if not, please point me to the right place or move this question accordingly.

My scenario is the following:

- multiple applications consisting of frontend (web GUI) and backend (REST services) are being developed following SOA/microservice approaches, each application has its own git repository
- some applications require a shared additional resource like frontend needs a HTTP server and multiple backend applications need a database server (with persistent storage)
- focus is primarily on offline mobile development (on the road) so a quick setup of required services/applications should be possible and the amount of resource overhead should be minimal. But of course the whole thing will be deployed/published at some point so i dont want to obstruct that if both can be managed
- development is done on windows and linux host machines
- access to all services from host machine is required for development purposes

What i am trying to achieve is to have a `docker-compose.yaml` file in the application repositories which i invoke via `docker-compose up` which would then start all required containers if not running already, e.g. the database container is started when i invoke `docker-compose up` in a backend application repository.

My approach was to have a new git repository which defines all shared docker images/containers, with its own `docker-compose.yaml` where all devs would have to run `docker-compose build` whenever something changed (might be automated with a git commit hook in the future). The central `docker-compose.yaml` looks like this

```
version: '3' services:   postgres:     build: ./images/postgres     image: MY-postgres     container_name: MY-postgres-server
```

The `Dockerfile` describing how each image is built is in its own subfolder and i think not relevant for the question, basically the default images for alpine + apache/postgres.

So the problem: how would a `docker-compose.yaml` in the application git repository look like that references the services/containers defined by the above central `docker-compose.yaml`.

Now since this is no new problem scenario, i did some research and honestly the variety of approaches and proposed solutions was confusing, for once the various versions and compatibilities, features that were deprecated, etc.

- We want one single database instance for now for performance reasons and simplicity (reddit) or is this the problem because it is truly considered an anti-pattern (via this answer). Each application would be using its own database within the container, so no sync required on application level.
- I am reading about volumes or data only containers to solve this problem, yet i cant understand how to implement
- Some (Single Host scenario) suggest links (with `depends_on`) while i think this concept has been superseeded by networks but is it still applying? There seemed to be an `extends` option as well
- `docker-compose` has an option `--no-deps` which is described as `Don't start linked services.`. If i omit it, i would assume it does what i need, but here i think then problem is the difference in meaning of image/container/service
- Can a combination of multiple compose files solve this problem? This would add a hard requirement on project paths though
- If i cant start the containers from my application directory, id like to at least link to them, is `external_links` the right approach?
- There are some feature requests (feature: including external docker-compose.yml, allow sharing containers across services) so maybe its just not possible currently with docker means? Then how to solve it with third-party like dcao include (which doesnt support `version 3`)?

Wow, that escalated quickly. But i wanted to show the research i have done since i just cant believe that its currently not possible.

Accepted Answer: None

Highest Rated Answer: None