Make Container more flexible:

Approach 1: use nix-shell and don't build container at runtime.

Start:
1. Config.nix file specifies environment
2. Container from minimal Docker image is running and config.nix file is available in this container

First time or after config change

- nix-shell downloads ① and builds the dependencies given in the config.nix.

Rem.: download from binary cache and store in shared store

Consecutive repl re-starts

- Assume all dependencies are already available in shared store, which is mounted into each container

nix-shell: Evaluates config.nix

nix-shell: ② sets environment variables to the correct derivation path in the store.

↓

End: Environment where all dependencies are available is ready.
→ run scripts, start shell

Performance bottlenecks

① Downloading packages can take a long time
Building of the dependencies is fast
Solution: Seeded store which has commonly used packages pre-downloaded

② Evaluating config.nix & exporting environment variables can induce a small delay.
Solution: Persist environment variables between restarts of nix-shell and invalidate cache if config.nix changes.

Approach 2: Build new container at runtime every time the config changes.

Start: Config file specifies environment and some container is running.

Nix docker builder

- Assume that one builder container is running.

1 Run builder with config.nix expression.

↓

2 Builder builds the environment and creates an image using nix.

↓

Load this image into docker and start container.

↓

End: Environment ready

Performance evaluation

1 One builder is a bottleneck.

Solution: Enable simultaneous builds by running multiple builders. Each nix builder uses persistent Nix-store shared among the builders.

2 Avoid unnecessary builds by only building new image if config.nix changes.

Nixery:

- Pull ad-hoc image based on config from Nixery container registry.

↓

Nixery builds and serves image

↓

End: Environment is ready

Performance evaluation

- If image is not available locally it takes some time for Nixery to build the image and then pull it to local disk
- Disk usage can grow quite large due to the large number of layers created and the different order in which packages are specified.

Open questions:

- Inclusion and caching of libraries that are used with a programming language
- Flexible configuration of package versions which are not in nixpkgs repository