```
$docker rm rails
```

Step 8

Let's link our container to our existing PostgreSQL container. The rake command below is a one-time process that needs to be done in all Rails applications to initialize the database.

```
$docker run -i -t --name rails --link database:db training/notes rake db:crea
te db:migrate
```

```
==  CreateNotes: migrating ====================================================
==
-- create_table(:notes)
   -> 0.0062s
==  CreateNotes: migrated (0.0064s) ===========================================
==
```

The --link flag connects one container to another. We specify the name of the container to link to, database, and an alias for the link 'db'.

Step 9

We can remove the container.

```
docker rm rails
```

Step 10

The link provides a secure tunnel between containers. On our database container port 5432 has been exposed to the linked container. Docker will automatically set environment variables in our container, to indicate connection information. Let's see that information.

```
docker run --rm --link database:db training/notes env
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=1d5cae52b3f6
DB_PORT=tcp://172.17.0.33:5432
DB_PORT_5432_TCP=tcp://172.17.0.33:5432
DB_PORT_5432_TCP_ADDR=172.17.0.33
DB_PORT_5432_TCP_PORT=5432
DB_PORT_5432_TCP_PROTO=tcp
DB_NAME=/reverent_ptolemy/db
DB_ENV_PG_VERSION=9.3
HOME=/
```

Each variable is prefixed with the link alias: db. It includes connection information plus any environment variables set in the database container via ENV instructions. The --rm removes the container after it exits, as this is not a long running container.

Step 11