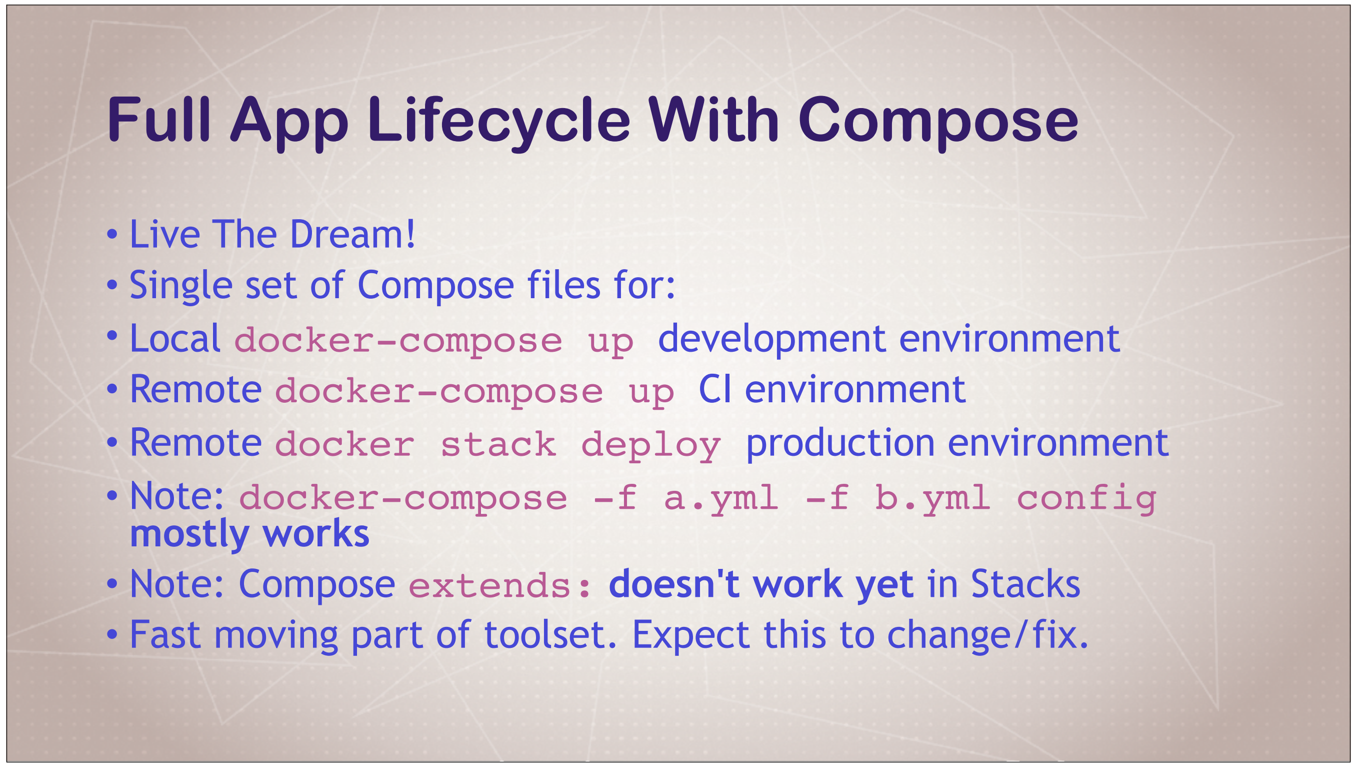
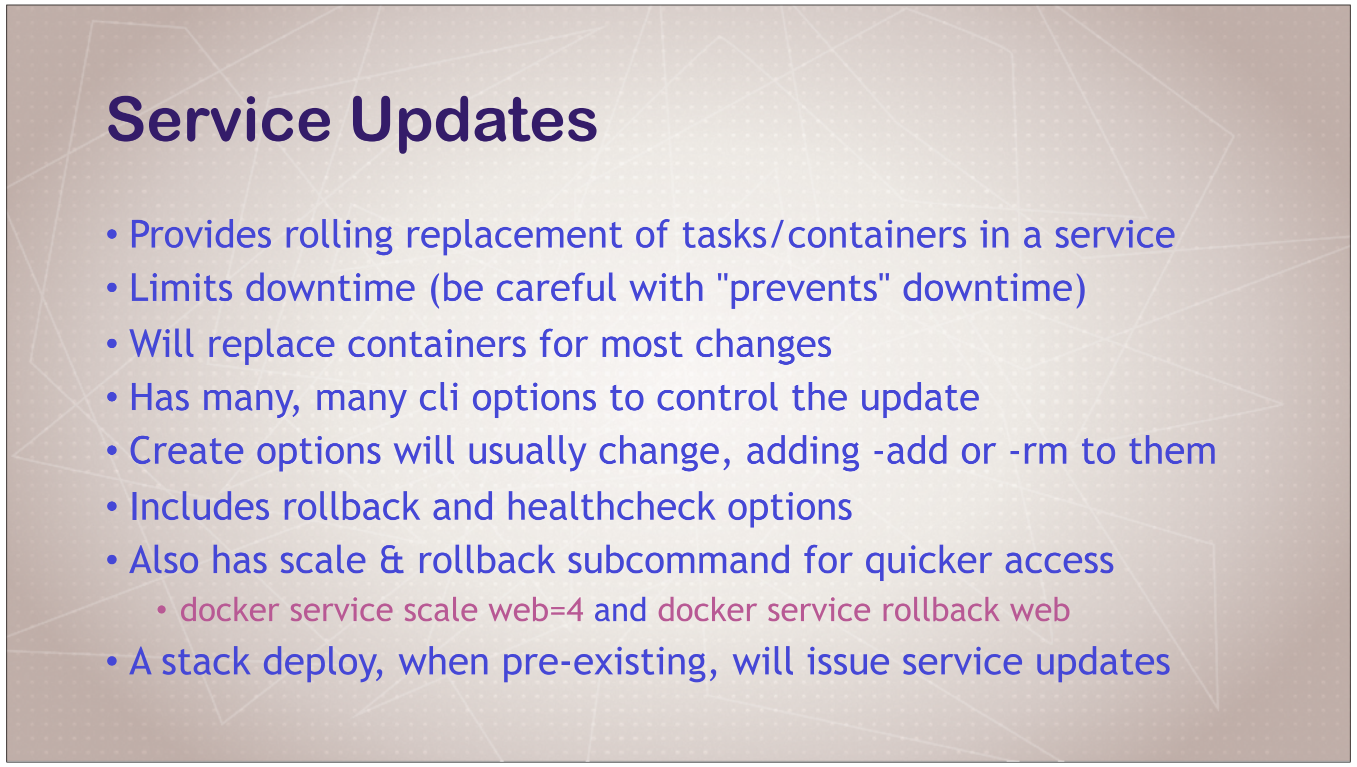


Full App Lifecycle With Compose

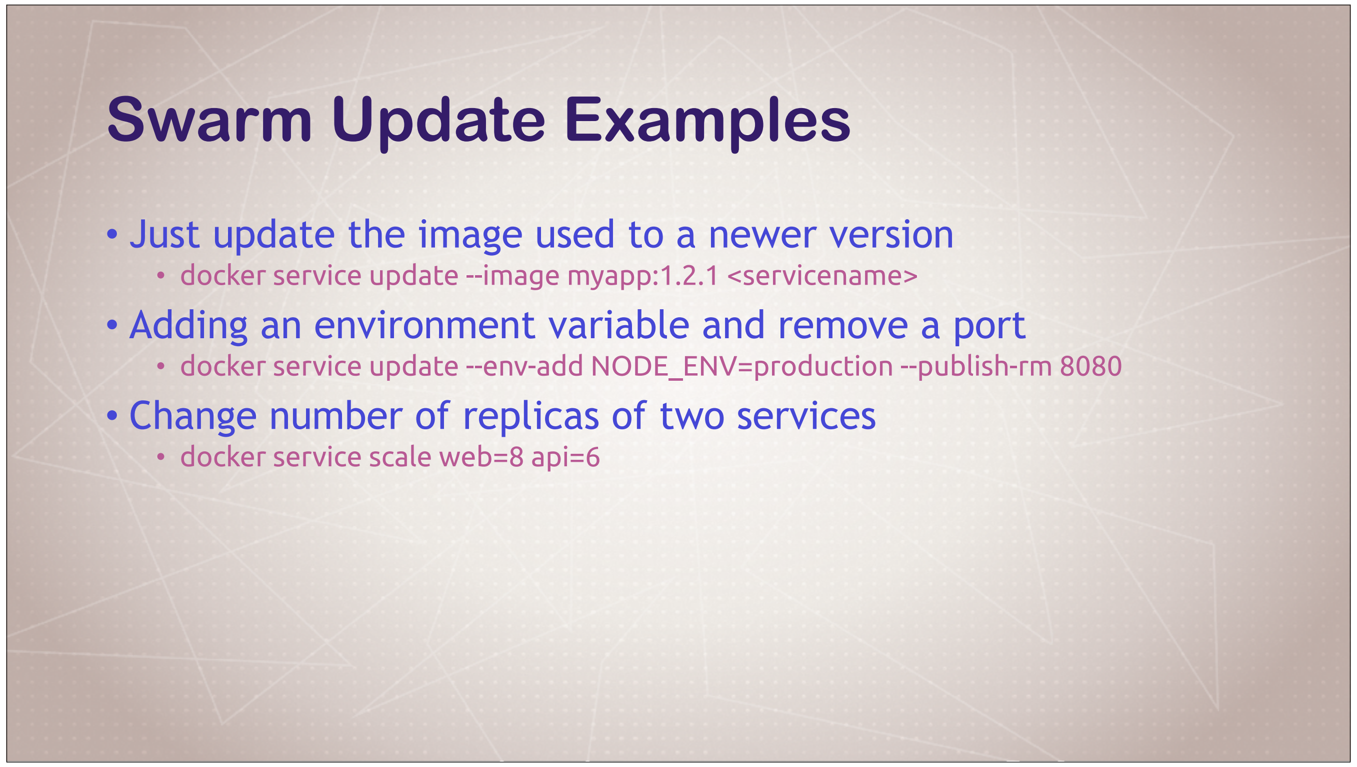
- Live The Dream!
- Single set of Compose files for:
- Local `docker-compose up` development environment
- Remote `docker-compose up` CI environment
- Remote `docker stack deploy` production environment
- Note: `docker-compose -f a.yml -f b.yml config` **mostly works**
- Note: Compose `extends:` **doesn't work yet** in Stacks
- Fast moving part of toolset. Expect this to change/fix.



Service Updates

- Provides rolling replacement of tasks/containers in a service
- Limits downtime (be careful with "prevents" downtime)
- Will replace containers for most changes
- Has many, many cli options to control the update
- Create options will usually change, adding -add or -rm to them
- Includes rollback and healthcheck options
- Also has scale & rollback subcommand for quicker access
  - docker service scale web=4 and docker service rollback web
- A stack deploy, when pre-existing, will issue service updates

Currently 77 options for update command



Swarm Update Examples

- Just update the image used to a newer version
  - docker service update --image myapp:1.2.1 <servicename>
- Adding an environment variable and remove a port
  - docker service update --env-add NODE_ENV=production --publish-rm 8080
- Change number of replicas of two services
  - docker service scale web=8 api=6

```
docker service create -p 8088:80 --name web nginx:1.13.7

docker service scale web=5

docker service update --image nginx:1.13.6 web

docker service update --publish-rm 8088 --publish-add 9090:80 web

docker service update --force web
```

Swarm Updates in Stack Files

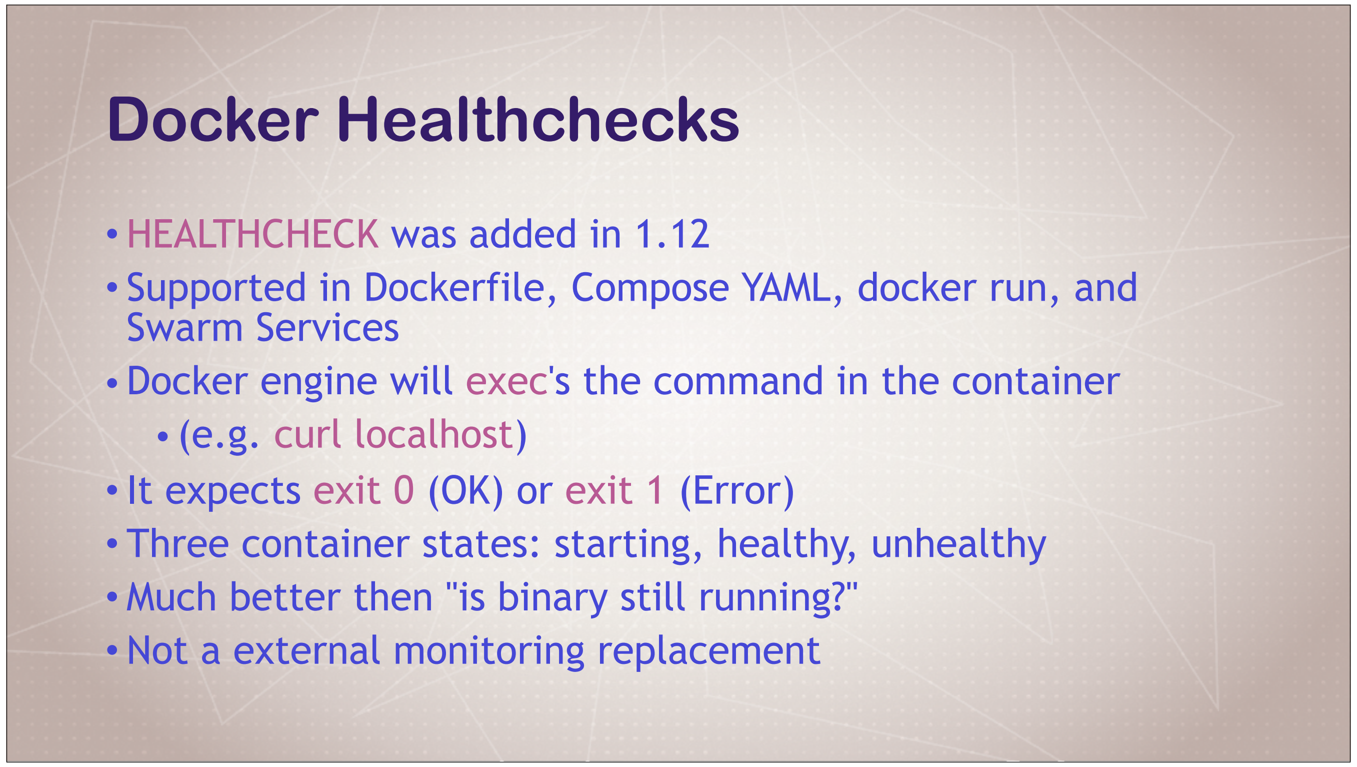
Same command. Just edit the YAML file, then

```
docker stack deploy -c file.yml <stackname>
```

Lecture Cleanup

- Remove the service we created

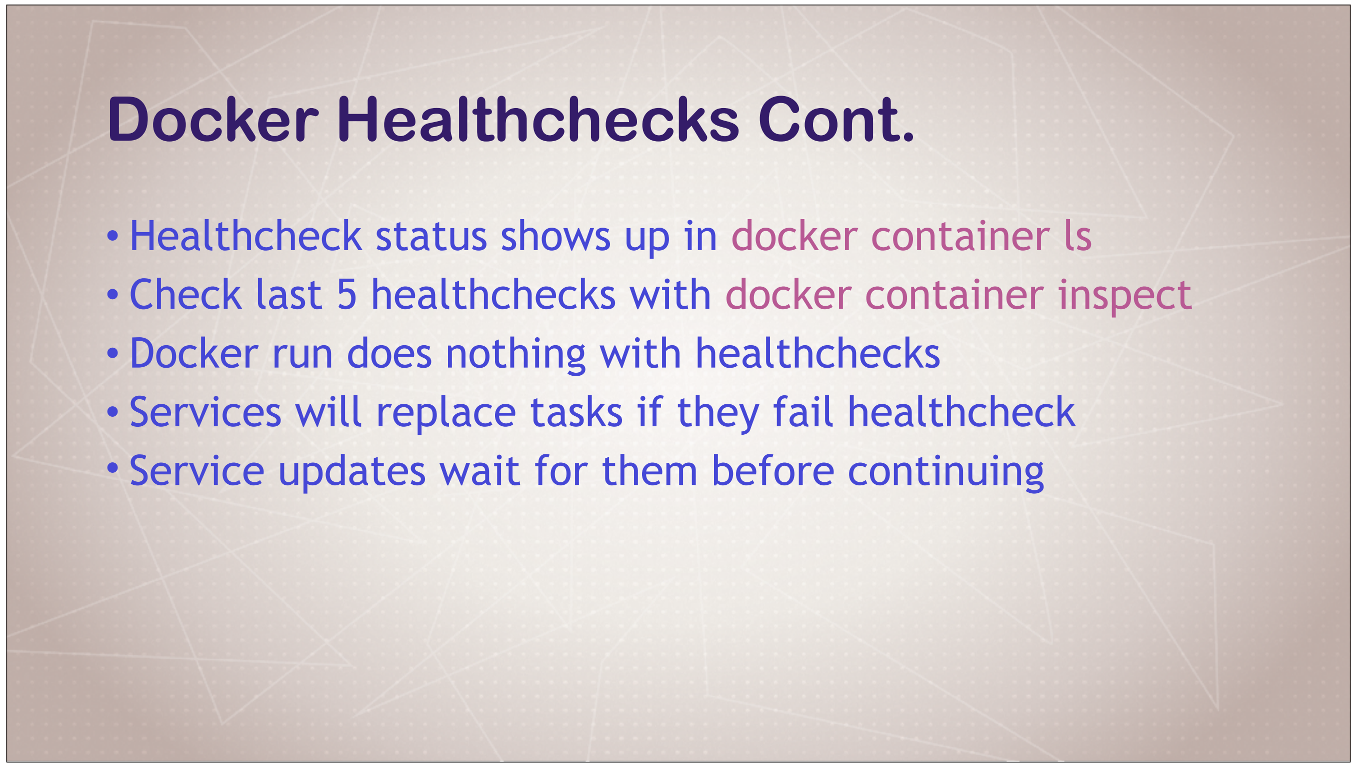
```
> docker service rm web
```



Docker Healthchecks

- HEALTHCHECK was added in 1.12
- Supported in Dockerfile, Compose YAML, docker run, and Swarm Services
- Docker engine will exec's the command in the container
  - (e.g. curl localhost)
- It expects exit 0 (OK) or exit 1 (Error)
- Three container states: starting, healthy, unhealthy
- Much better then "is binary still running?"
- Not a external monitoring replacement

RESOURCES
https://docs.docker.com/engine/reference/builder/#healthcheck
sample healthchecks https://github.com/docker-library/healthcheck



Docker Healthchecks Cont.

- Healthcheck status shows up in docker container ls
- Check last 5 healthchecks with docker container inspect
- Docker run does nothing with healthchecks
- Services will replace tasks if they fail healthcheck
- Service updates wait for them before continuing

Healthcheck Docker Run Example

```
docker run \
  --health-cmd="curl -f localhost:9200/_cluster/health || false" \
  --health-interval=5s \
  --health-retries=3 \
  --health-timeout=2s \
  --health-start-period=15s \
  elasticsearch:2
```

Healthcheck Dockerfile Examples

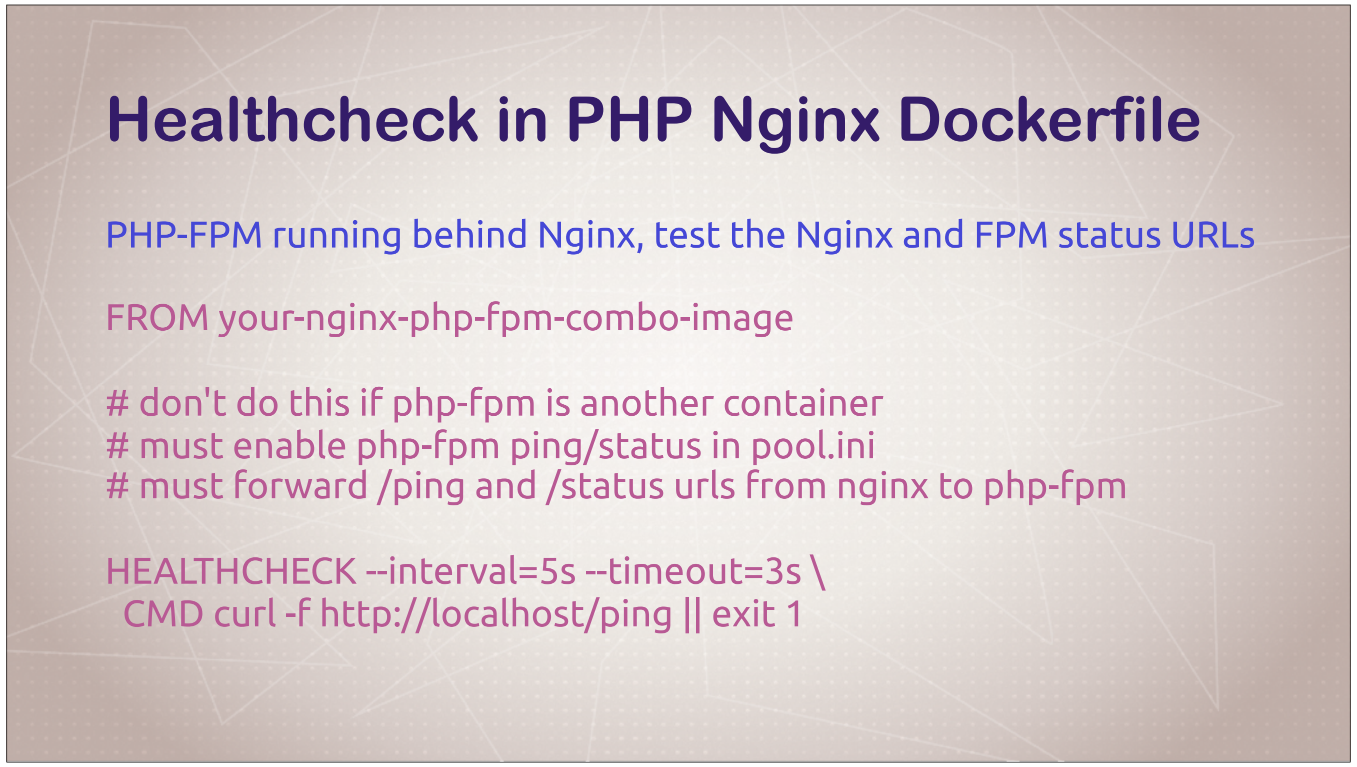
- Options for healthcheck command
  - --interval=DURATION (default: 30s)
  - --timeout=DURATION (default: 30s)
  - --start-period=DURATION (default: 0s) (17.09+)
  - --retries=N (default: 3)
- Basic command using default options
  - HEALTHCHECK curl -f http://localhost/ || false
- Custom options with the command
  - HEALTHCHECK --timeout=2s --interval=3s --retries=3 \
  - CMD curl -f http://localhost/ || exit 1

Healthcheck in Nginx Dockerfile

Static website running in Nginx, just test default URL

```
FROM nginx:1.13

HEALTHCHECK --interval=30s --timeout=3s \
  CMD curl -f http://localhost/ || exit 1
```



Healthcheck in PHP Nginx Dockerfile

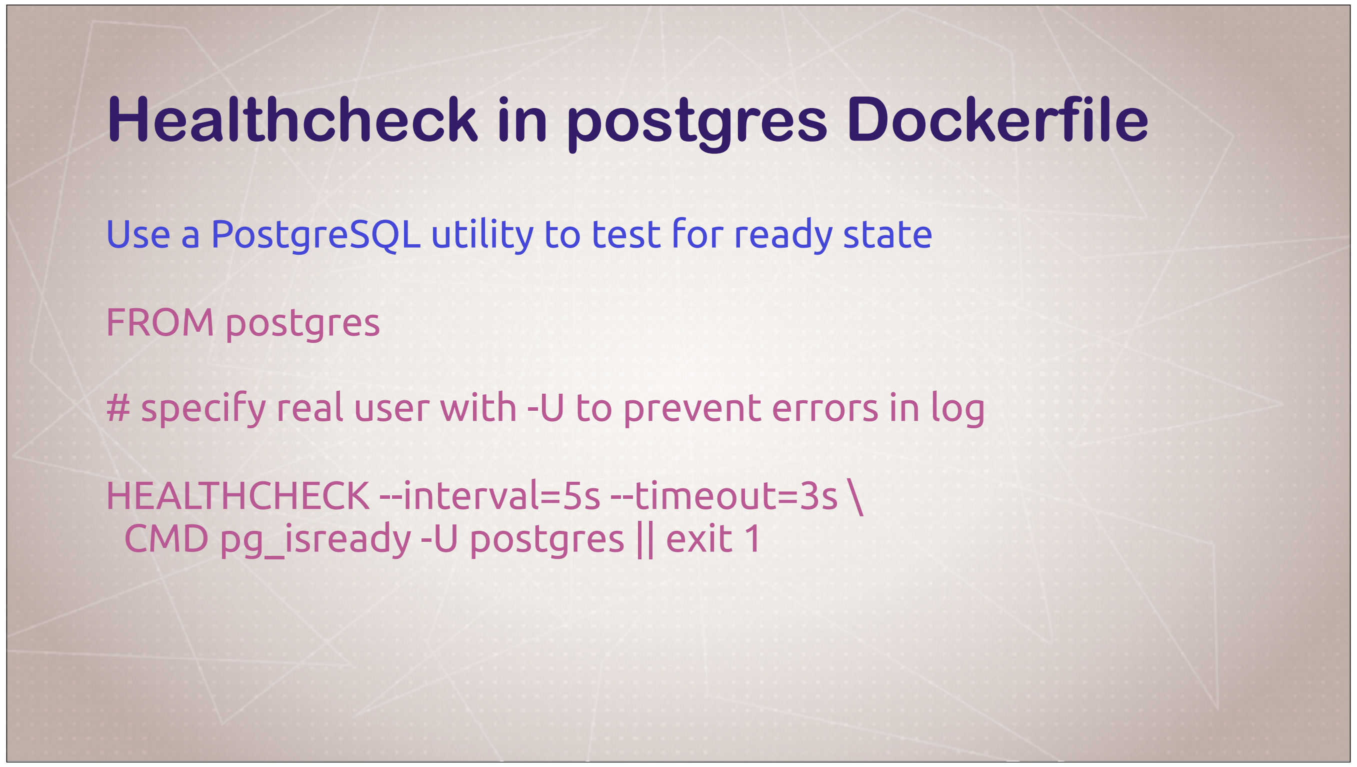
PHP-FPM running behind Nginx, test the Nginx and FPM status URLs

```
FROM your-nginx-php-fpm-combo-image

# don't do this if php-fpm is another container
# must enable php-fpm ping/status in pool.ini
# must forward /ping and /status urls from nginx to php-fpm

HEALTHCHECK --interval=5s --timeout=3s \
  CMD curl -f http://localhost/ping || exit 1
```

RESOURCES
https://github.com/BretFisher/php-docker-good-defaults



Healthcheck in postgres Dockerfile

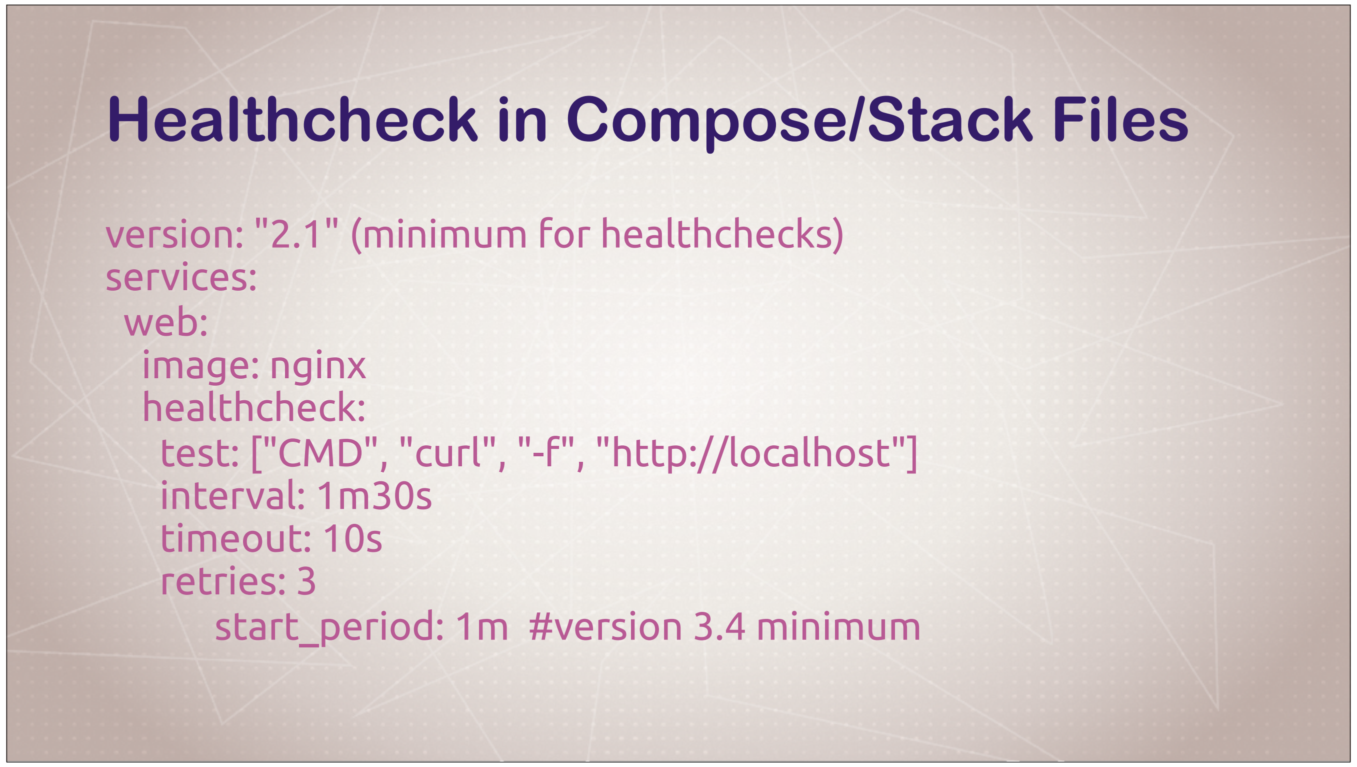
Use a PostgreSQL utility to test for ready state

```
FROM postgres

# specify real user with -U to prevent errors in log

HEALTHCHECK --interval=5s --timeout=3s \
  CMD pg_isready -U postgres || exit 1
```

Resources for this Lecture has a github repo with samples from some official images



Healthcheck in Compose/Stack Files

```
version: "2.1" (minimum for healthchecks)
services:
 web:
  image: nginx
  healthcheck:
   test: ["CMD", "curl", "-f", "http://localhost"]
   interval: 1m30s
   timeout: 10s
   retries: 3
      start_period: 1m  #version 3.4 minimum
```

Start a postgres container
docker container run --name p1 -d postgres

Then start one with healthcheck
docker container run --name p2 -d --health-cmd="pg_isready -U postgres || exit 1" postgres

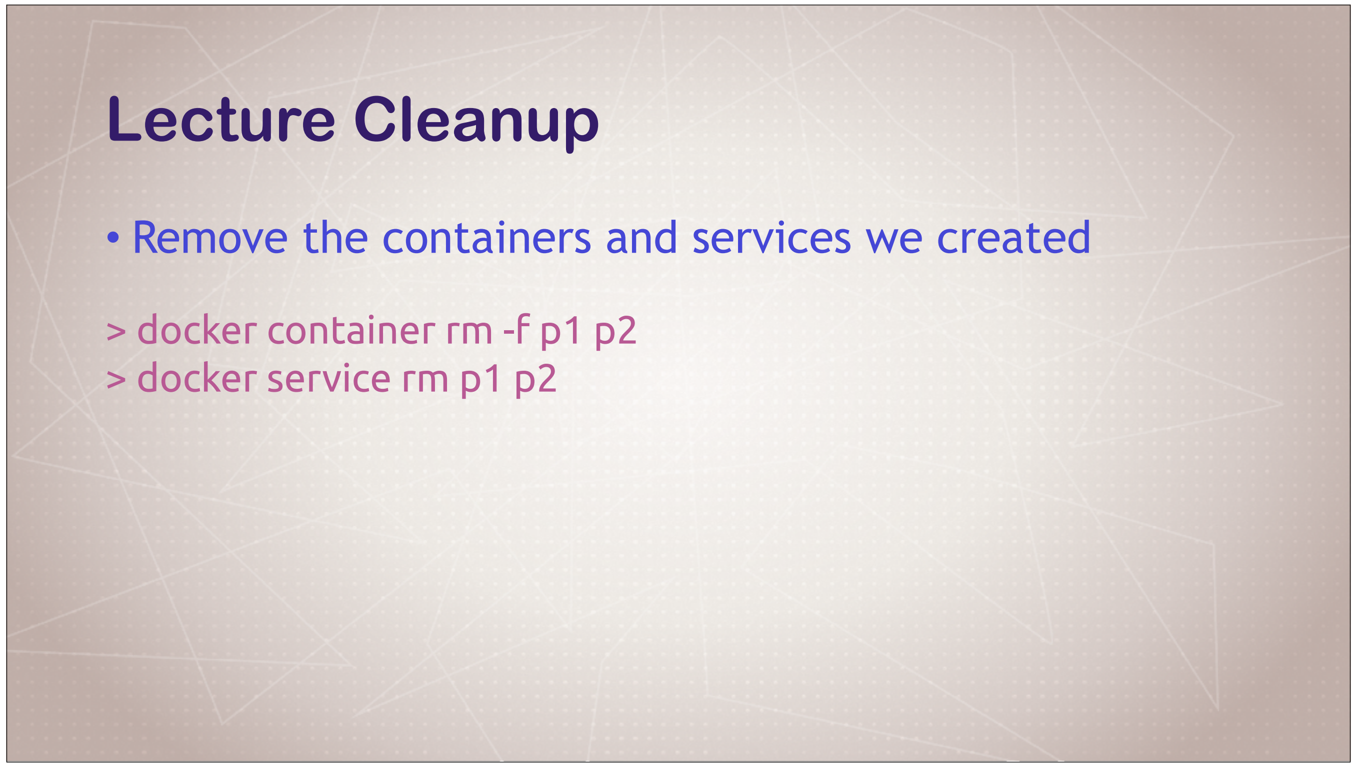
docker container ls

docker container inspect p2

docker service create --name p1 postgres

docker service create --name p2 --health-cmd="pg_isready -U postgres || exit 1" postgres

preparing
starting
running



Lecture Cleanup

- Remove the containers and services we created

```
> docker container rm -f p1 p2
> docker service rm p1 p2
```