DOCKER-COMPOSE BY EXAMPLE

- Darren Wurf
- darren.wurf@gmail.com

DOCKER

- Docker is a useful abstraction for *processes*
  - Web application server (django/rails/tomcat/...)
  - Database
  - Reverse HTTP proxy (nginx/apache)
- ... each in their own Docker container.
- The process has all its code/binaries, libraries and configuration in a portable package.

DOCKER BENEFITS

Why dockerise your service?

- Remove platform dependencies
  - essential OS parts are packaged with your app)
- Simplify installation / running
  - `docker run <service>`
- Manage external dependencies
  - control over files/directories, network ports, databases and other dependent processes
- Make Dev like Prod
  - Develop locally with a full app stack, without complex install and management

INSTALLING DOCKER

- Linux: https://docs.docker.com/engine/installation/
- Windows/OS X: use docker toolbox

INSTALLING DOCKER

Or run these commands (Ubuntu):

```
sudo apt-get update
sudo apt-get -y install apt-transport-https
sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers
sudo sh -c "echo 'deb https://apt.dockerproject.org/repo u
sudo apt-get update
sudo apt-get -y install docker-engine
sudo apt-get -y install curl
sudo bash -c "curl -L https://github.com/docker/compose/re
sudo chmod u+x /usr/local/bin/docker-compose
sudo docker-compose -v
```

LIVE DEMOS!

LIVE DEMO: HELLO WORLD

Create + start

```
sudo docker run \
    --name hello \
    --rm \
    -p 80:5000 \
    training/webapp
```

See what's in the container here

LIVE DEMO: HELLO WORLD

DOCKER-COMPOSE

docker-compose.yml

```
hello:
    container_name: hello
    image: training/webapp
    restart: always
    ports:
        - 80:5000
```

Start/stop

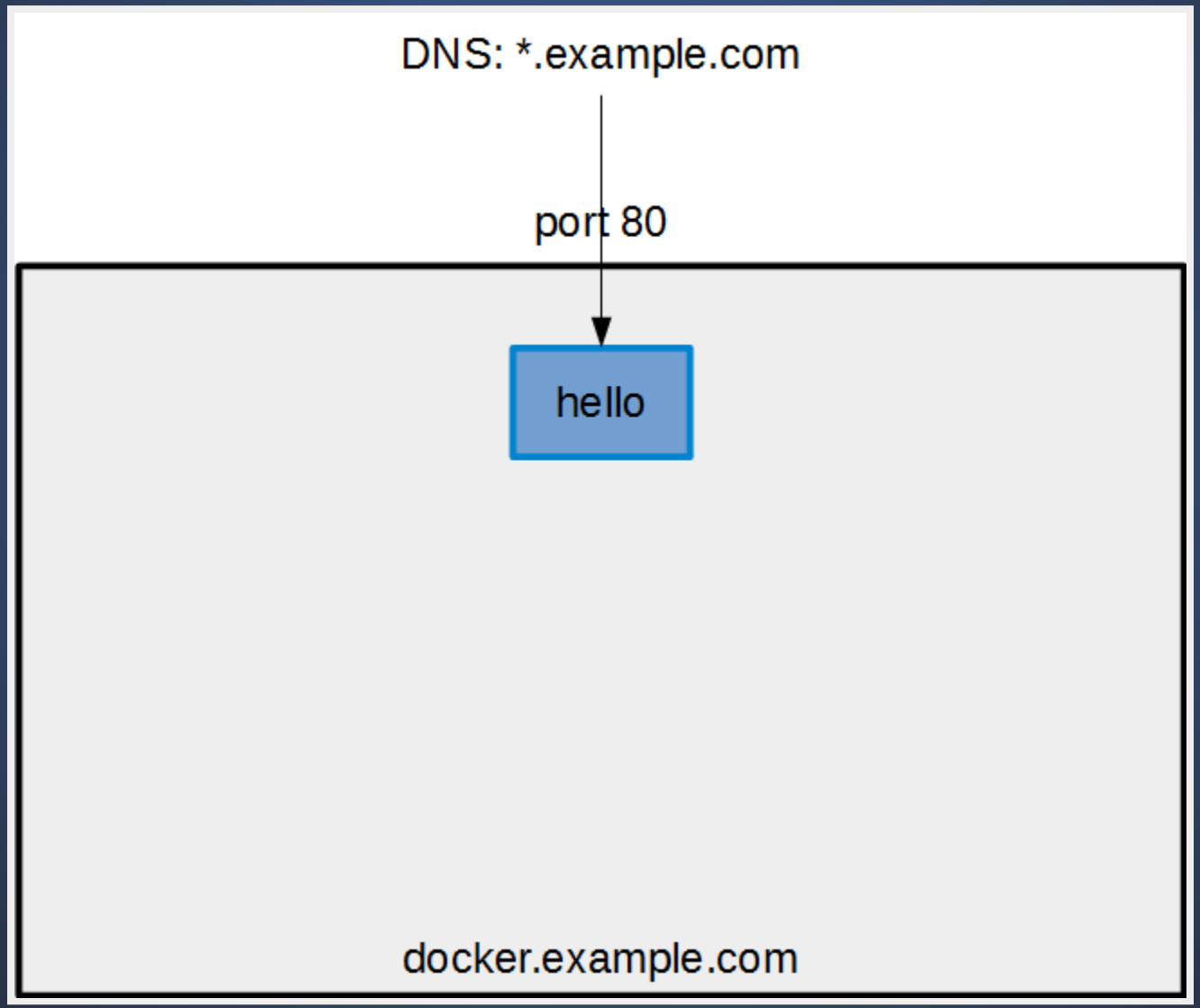
```
    sudo docker-compose up -d
    sudo docker-compose down
```

DNS TRICKS

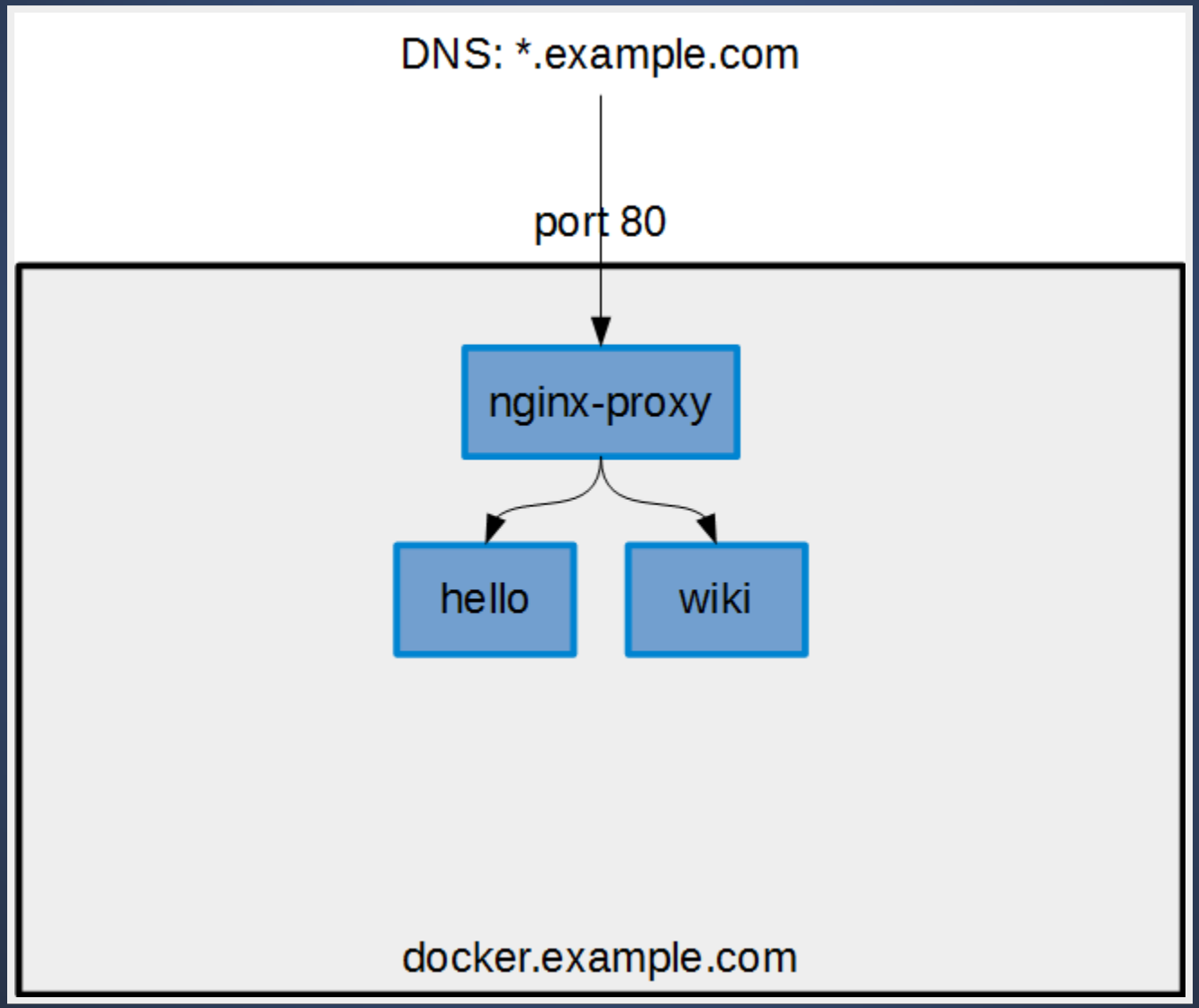
Quick and easy public docker containers

- Get a cheap domain (~$3/yr)
- Use Wildcard DNS to forward all requests to a single IP
- Use container `jwilder/nginx-proxy` for automatic HTTP routing

SINGLE-CONTAINER SETUP



MULTIPLE CONTAINER SETUP



LIVE DEMO: MULTIPLE CONTAINERS

proxy/docker-compose.yml

```
proxy:
    container_name: proxy
    image: jwilder/nginx-proxy
    restart: always
    volumes:
        - /var/run/docker.sock:/tmp/docker.sock:ro
    ports:
        - 80:80
```

LIVE DEMO: MULTIPLE CONTAINERS

hello/docker-compose.yml

```
hello:
    container_name: hello
    image: training/webapp
    restart: always
    environment:
        - VIRTUAL_HOST=hello.example.com
        - VIRTUAL_PORT=5000
```

wiki/docker-compose.yml

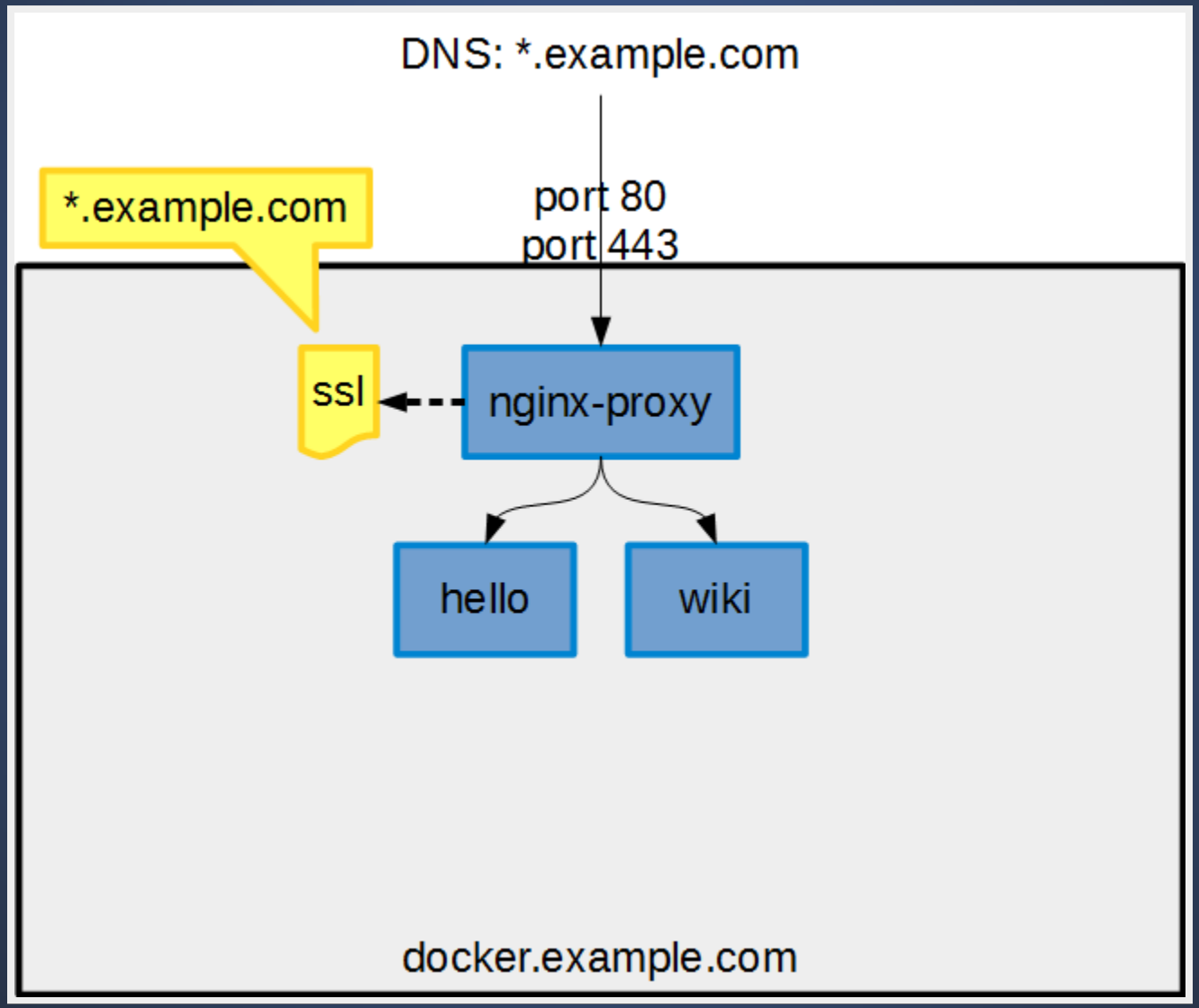
```
wiki:
    container_name: wiki
    image: mazzolino/tiddlywiki
    restart: always
    environment:
        - VIRTUAL_HOST=wiki.example.com
        - VIRTUAL_PORT=8080
```

SSL TRICKS

Quick and easy *secure* public docker containers

- Use Wildcard DNS to forward all requests to a single IP
- Get a wildcard SSL certificate (~$55 https://ssl2buy.com)
- Use container `jwilder/nginx-proxy` for automatic **HTTPS** routing

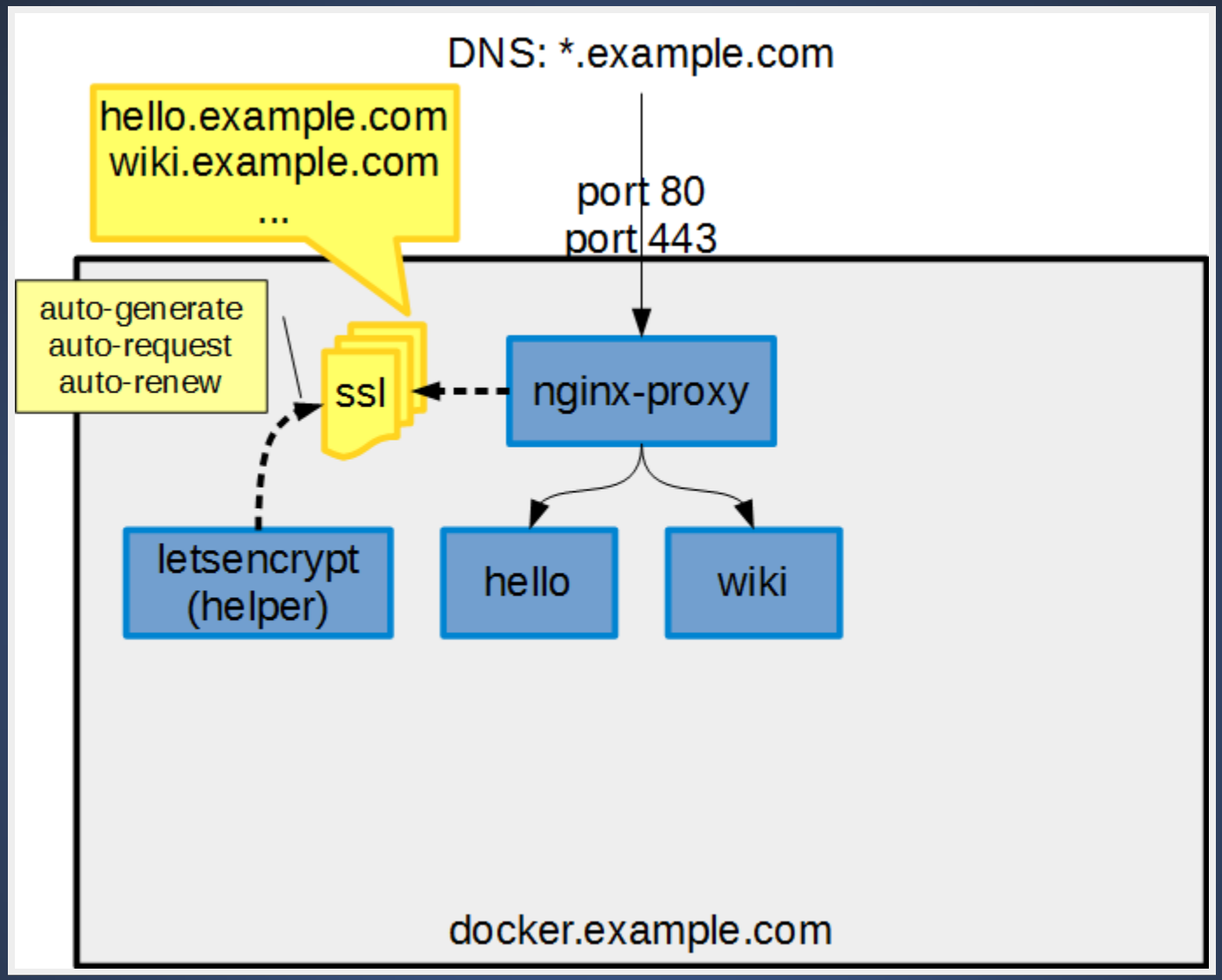
MULTIPLE CONTAINERS WITH SSL



LET'S ENCRYPT

- Free automatic domain validated SSL certs for everyone!
- But what about our proxy? :(
- Already solved: `jrcs/letsencrypt-nginx-proxy-companion`
- Beware! Rate limit is 5 certs per domain per week (including subdomains)

MULTIPLE CONTAINERS WITH LET'S ENCRYPT SSL


DNS: *.example.com
hello.example.com
wiki.example.com
...
port 80
port 443
auto-generate
auto-request
auto-renew
ssl
nginx-proxy
letsencrypt
(helper)
hello
wiki
docker.example.com

LIVE DEMO: MULTIPLE CONTAINERS + LET'S ENCRYPT SSL

proxy/docker-compose.yml

```
proxy:
    container_name: proxy
    image: jwilder/nginx-proxy
    restart: always
    volumes:
        - /var/run/docker.sock:/tmp/docker.sock:ro
        - /srv/docker/letsencrypt:/etc/nginx/certs:ro
        - /etc/nginx/vhost.d
        - /usr/share/nginx/html
    ports:
        - 80:80
        - 443:443
letsencrypt:
    container_name: letsencrypt
    image: jrcs/letsencrypt-nginx-proxy-companion
    restart: always
```

```
    volumes:
        - /var/run/docker.sock:/var/run/docker.sock:ro
        - /srv/docker/letsencrypt:/etc/nginx/certs:rw
    volumes_from:
        - proxy
```

LIVE DEMO: MULTIPLE CONTAINERS

hello/docker-compose.yml

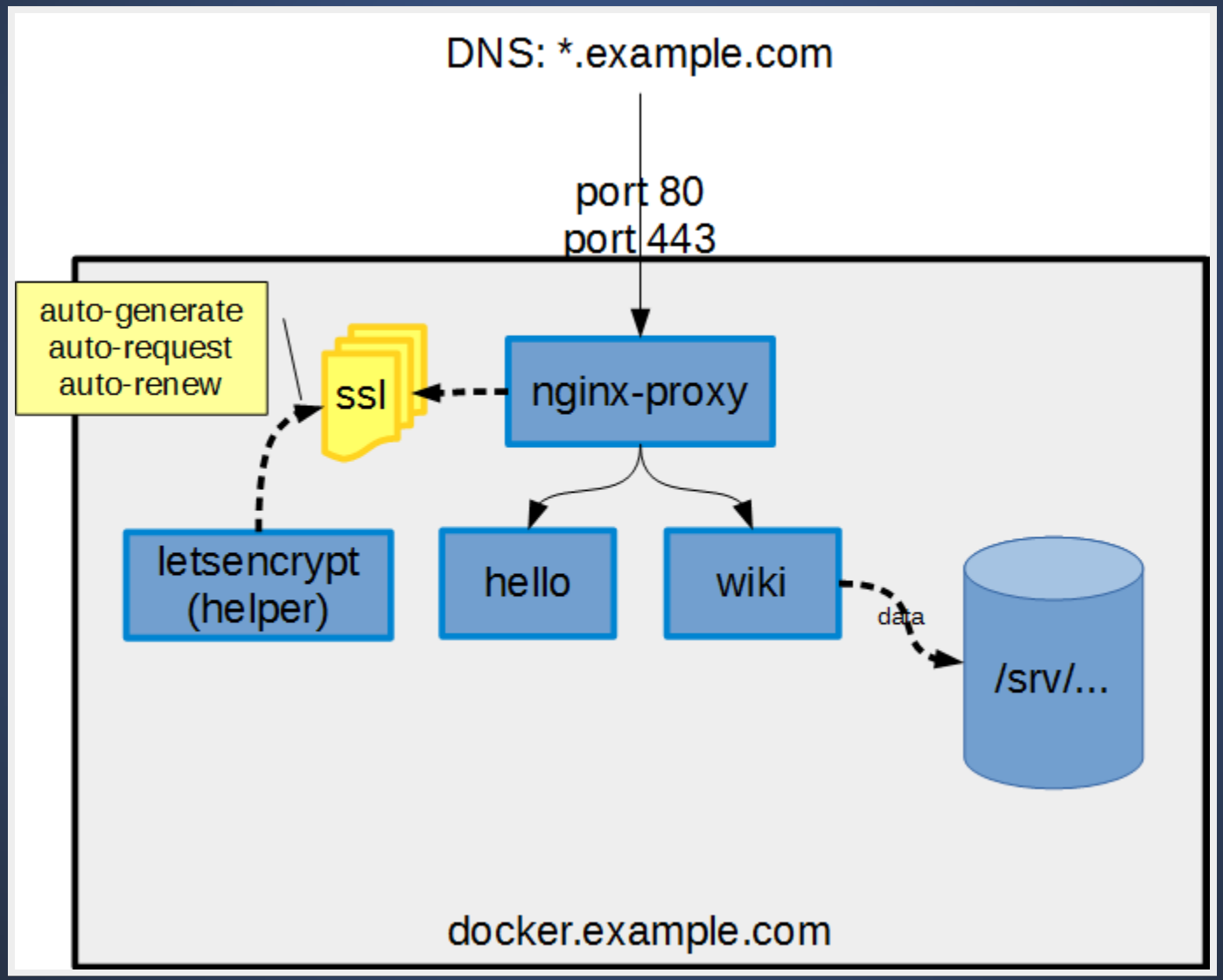
```
hello:
    container_name: hello
    image: training/webapp
    restart: always
    environment:
        - VIRTUAL_HOST=hello.example.com
        - VIRTUAL_PORT=5000
        - LETSENCRYPT_HOST=hello.example.com
        - LETSENCRYPT_EMAIL=hello@example.com
```

wiki/docker-compose.yml

```
wiki:
    container_name: wiki
    image: mazzolino/tiddlywiki
    restart: always
    environment:
        - VIRTUAL_HOST=wiki.example.com
        - VIRTUAL_PORT=8080
```

```
      - LETSENCRYPT_HOST=wiki.example.com
      - LETSENCRYPT_EMAIL=wiki@example.com
```

PERSISTENT STORAGE

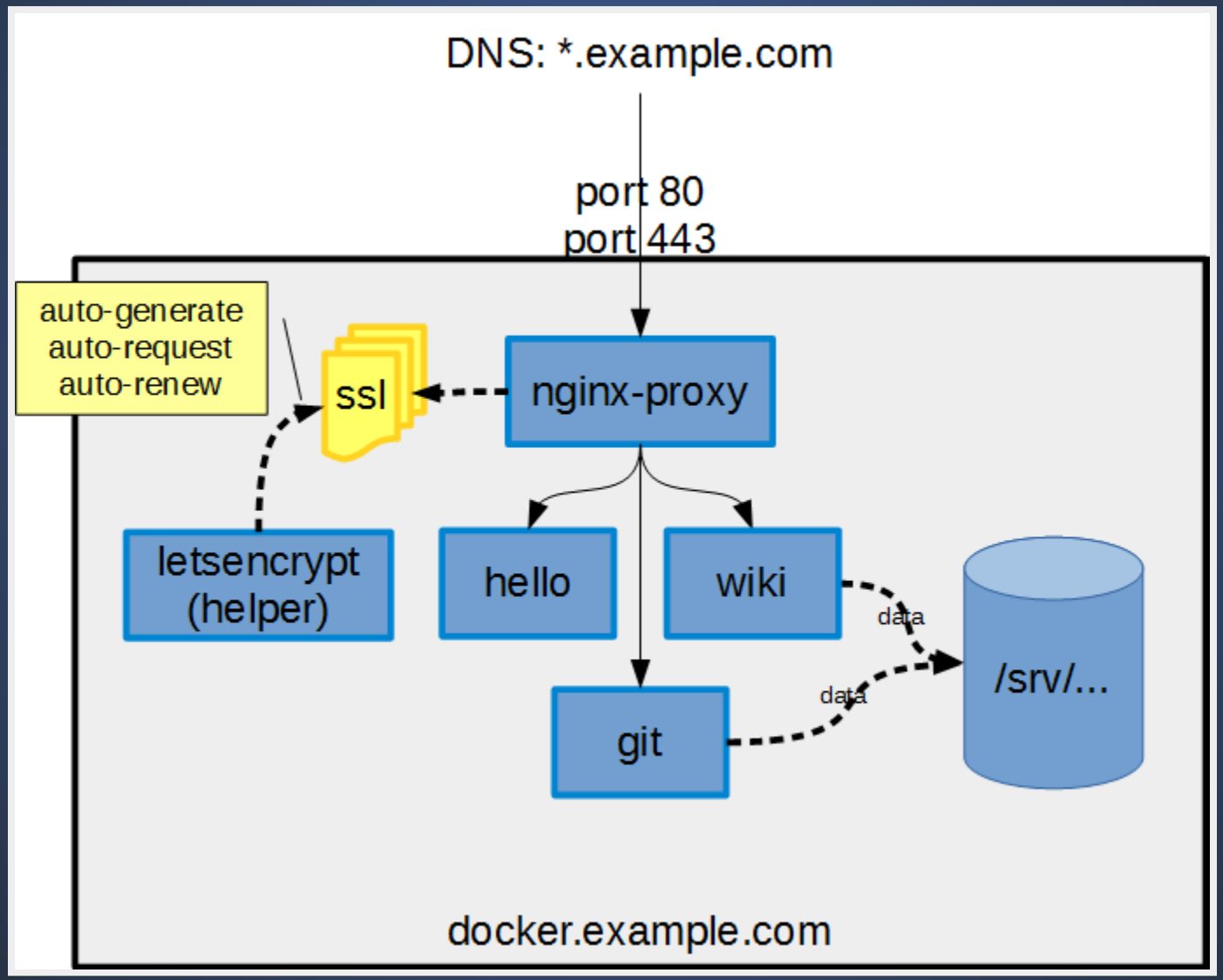


LIVE DEMO: PERSISTENT STORAGE

wiki/docker-compose.yml

```
wiki:
    container_name: wiki
    image: mazzolino/tiddlywiki
    restart: always
    volumes:
        - /srv/docker/wiki:/var/lib/tiddlywiki
    environment:
        - VIRTUAL_HOST=wiki.example.com
        - VIRTUAL_PORT=8080
        - LETSENCRYPT_HOST=wiki.example.com
        - LETSENCRYPT_EMAIL=wiki@example.com
```

SELF-HOSTED GITHUB CLONE



LIVE DEMO: GITBUCKET

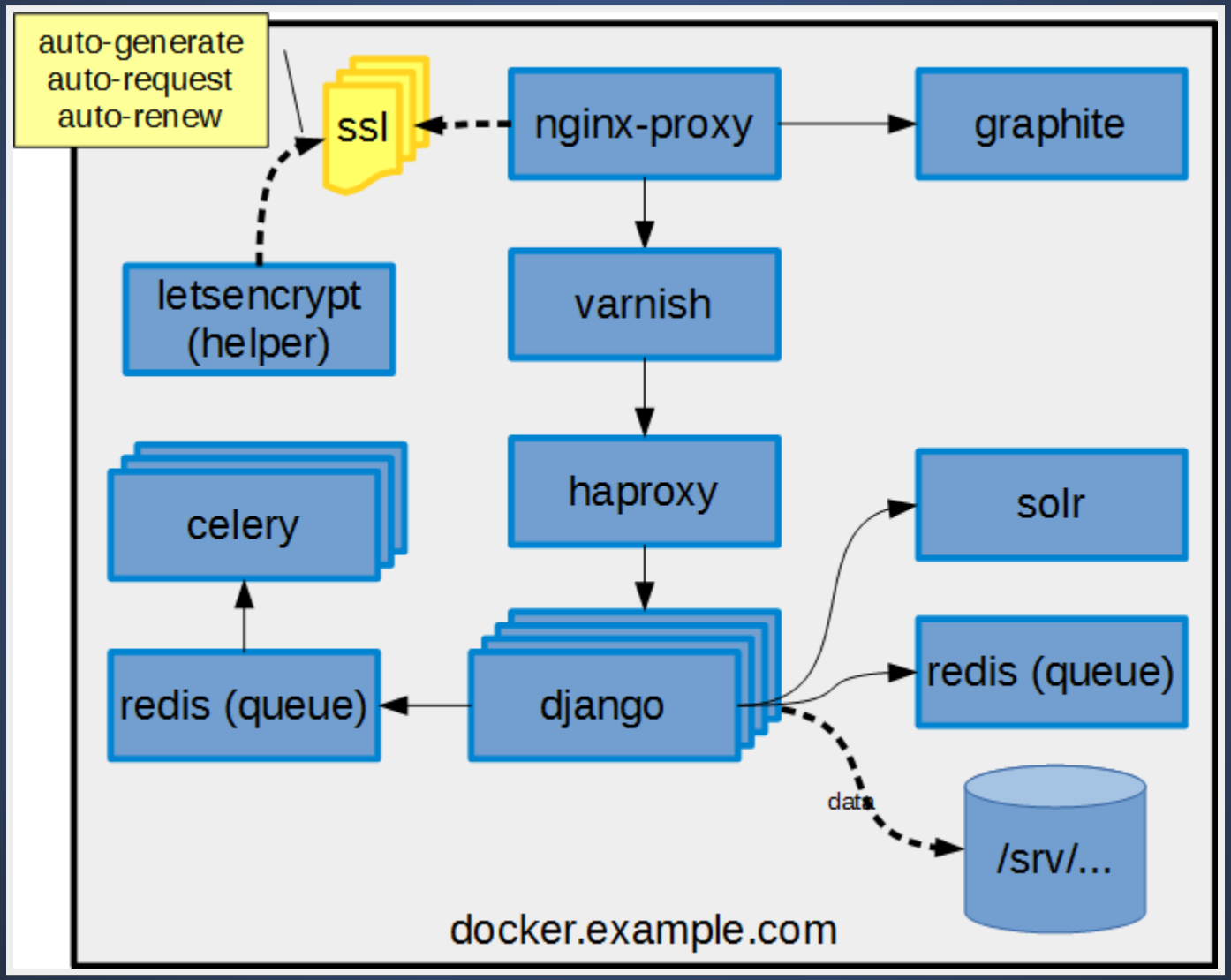
git/docker-compose.yml

```
git:
    container_name: git
    image: f99aq8ove/gitbucket
    restart: always
    volumes:
        - /srv/docker/git:/gitbucket
    ports:
        - 29418:29418
    environment:
        - VIRTUAL_HOST=git.example.com
        - VIRTUAL_PORT=8080
        - LETSENCRYPT_HOST=git.example.com
        - LETSENCRYPT_EMAIL=git@example.com
```

WEB STACK

- How do we harness docker to write a modern scalable web app?
- Some great presentations on web stacks:
  - A New Default Web Stack - Simon Willison (EventBrite)
  - The inner guts of Bitbucket - Erik van Zijst (Atlassian)

A DEFAULT WEB STACK ON DOCKER?



LIVE DEMO: FULL WEB STACK

https://github.com/dwurf/default-web-stack/blob/master/docker-compose.yml

BACKUP CONSIDERATIONS

- Use Volumes to control the location of data
  - Bind mount volumes are the easiest to work with
  - Find your data by inspecting `Dockerfiles`
- Place all data in one location
  - Standard location? I use `/srv/docker/<container>`
  - Use standard backup tools
  - Practice restores!

END