Simplify Project Deployment with Docker & Github Actions

Requirements for local computer:
- Node.js (see: SiteBuilderKit.pdf page 43)
- Github Desktop (see: SiteBuilderKit.pdf page 50)
- Docker Desktop

Docker Desktop can be installed from: https://www.docker.com/products/docker-desktop/

Cloud hosting used: DigitalOcean.

Create a Droplet

See video: 01-new-droplet.mp4

Now we have:

Public IP: 46.101.247.241 (private IP: 10.114.0.4)
User: root
Pwd: <root-password>

Note: IP here is just an example. Please replace with your droplet IP.

Connect to your server and add a new user (named: hobnob)

See video: 02-connect-adduser.mp4

```
ssh root@46.101.247.241
```

```
sudo apt-get update
```

```
adduser hobnob
```
(specify a password when prompted)

```
usermod -aG sudo hobnob
```

```
exit
```

Now we have:

A new user: hobnob
Pwd: <hobnob-password>

Add SSH key authentication to connect to your server

Check if SSH key exists:
```
ls -l ~/.ssh/id_*.pub
```

If it doesn't exists, create:
```
ssh-keygen -t rsa -b 4096 -C youremail@example.com
```

Copy to your server (for the root user):
```
ssh-copy-id root@46.101.247.241
```

Copy to your server (for the user, hobnob)
```
ssh-copy-id hobnob@46.101.247.241
```

On Windows, use:

> Copy to your server (for the root user):
> ```
> type C:\Users\<USER>\.ssh\id_rsa.pub | ssh root@46.101.247.241 "cat >> .ssh/authorized_keys"
> ```
>
> Copy to your server (for the user, hobnob)
> ```
> type C:\Users\<USER>\.ssh\id_rsa.pub | ssh hobnob@46.101.247.241 "cat >> .ssh/authorized_keys"
> ```

Test connecting:

```
ssh hobnob@46.101.247.241
```

This time you will be logged in immediately without being prompted to enter password.

Install Firewall & Docker

See video: 03-firewall-docker.mp4

Install Firewall:

```
sudo apt install ufw
```

```
sudo ufw status
```

```
sudo ufw app list
```

Allow SSH:

```
sudo ufw allow OpenSSH
```

```
sudo ufw enable
```

```
sudo ufw status
```

Install Docker:

```
sudo apt update
```

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

```
apt-cache policy docker-ce
```

```
sudo apt install docker-ce

sudo systemctl status docker

sudo usermod -aG docker ${USER}

su - ${USER}

mkdir -p ~/.docker/cli-plugins/

curl -SL https://github.com/docker/compose/releases/download/v2.2.3/docker-compose-linux-x86_64 -o
~/.docker/cli-plugins/docker-compose

chmod +x ~/.docker/cli-plugins/docker-compose

sudo chown $USER /var/run/docker.sock

docker compose version
```

Setup MongoDB

See video: 04-mongodb.mp4

```
docker pull mongo

mkdir -p ~/mongodata

docker run -it -v ~/mongodata:/data/db -p 27017:27017 --name mongodb -d mongo
```

Connect to the container using the interactive terminal:

```
docker exec -it mongodb bash
```

Then type:

```
mongosh
```

```
use admin
```

```
show dbs
```

Create a database user, for example:

User: dbadmin
Pwd: a378ag4y7m4

```
db.createUser(
{
    user: "dbadmin",
    pwd: "a378ag4y7m4",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase" ]
}
)
```

Type exit to leave the MongoDB shell:

```
exit
```

Then exit once again to leave the Interactive shell:

```
exit
```

```
docker stop mongodb
```

```
docker rm mongodb
```

```
docker run -d -it -p 27017:27017 \
--restart unless-stopped \
--log-driver json-file \
--log-opt max-size=10m \
--log-opt max-file=5 \
-e MONGO_INITDB_ROOT_USERNAME=dbadmin \
-e MONGO_INITDB_ROOT_PASSWORD=a378ag4y7m4 \
-v ~/mongodata:/data/db \
--name mongodb \
mongo
```

Connect to the container using the interactive terminal:

```
docker exec -it mongodb mongosh -u dbadmin -p a378ag4y7m4
```

Check:

```
use admin
```

```
db.getUsers()
```

Type exit to leave the MongoDB shell:

```
exit
```

Then exit once again to leave the Interactive shell:

```
exit
```

Then exit from your server:

```
exit
```

Install the mywebapp project (on your local computer)

Unzip **mywebapp.zip**.

See video: 05-unzip-project-packages.mp4

Install:

See video: 06-mywebapp-install.mp4

Go to the mywebapp folder:
`cd mywebapp`

`npm install`

Open file **.env.development**

See video: See video: 07-mywebapp-run.mp4

Update the variables:

```
MAIN_HOST=localhost:3000

NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET=INp3IvdIlaMcoGAgFGoA66DdCclzzSqnXJZkgz8PSzx

WEB_ASSETS_PATH=~/
WEB_ASSETS_URL=http://localhost:8081

MONGODB_USERNAME=dbadmin
MONGODB_PASSWORD=a378ag4y7m4
MONGODB_HOSTNAME=46.101.247.241
MONGODB_PORT=27017
MONGODB_DATABASE=nextsite
```

Note: as seen we have a NEXTAUTH_SECRET variable that will be used for the app.
You can generate a secret value using: https://generate-secret.now.sh/32

Try run the project:
`npm run dev`

Test:

See video: 08-mywebapp-test.mp4

Open:

http://localhost:3000

Press CTRL-C to stop.

Build docker image:
(make sure you have Docker Desktop installed and run on your local computer)

See video: 09-mywebapp-docker.mp4

```
docker build . --file Dockerfile -t mywebapp
```

Run:

```
docker run -v ~/uploads:/app/uploads/  --env MAIN_HOST=localhost:3000 --env NEXTAUTH_URL=http://localhost:3000 --env NEXTAUTH_SECRET=INp3IvdIlaMcoGAgFGoA66DdCclzzSqnXJZkgz8PSzx --env MONGODB_USERNAME=dbadmin --env MONGODB_PASSWORD=a378ag4y7m4 --env MONGODB_HOSTNAME=46.101.247.241 --env MONGODB_PORT=27017 --env MONGODB_DATABASE=nextsite  --env WEB_ASSETS_PATH=/app --env WEB_ASSETS_URL=http://localhost:8081 --restart unless-stopped --name mywebapp -p 3000:3000 mywebapp
```

Check again:
http://localhost:3000

Press CTRL-C to stop.

Install the mywebassets project (on your local computer)

Unzip **mywebassets.zip**.

Install:

See video: 10-mywebassets-install.mp4

Go to the mywebassets folder:

```
cd mywebassets
```

```
npm install
```

```
docker build . --file Dockerfile -t mywebassets
```

```
docker run -p 8081:8081 -v ~/uploads:/app/uploads/ mywebassets
```

Test:
http://localhost:3000
http://localhost:8081 (empty)

Publish to Github

See video: 11-github-push.mp4

See video: 12-github-push.mp4

Open Github Desktop and publish the projects to your Github:

- mywebapp
- mywebassets

Create SSH key on your server & Configure Github

See video: 13-github-ssh.mp4

Connect to your server:
```
ssh hobnob@46.101.247.241
```

```
ssh-keygen -t rsa -b 4096 -C youremail@example.com
```

Show key:
```
cat .ssh/id_rsa.pub
ssh-rsa AAA.....................@example.com
```

Copy the key shown.

Create file:
```
vi .ssh/authorized_keys
```
Paste the key here, save and exit.

```
chmod 700 .ssh/authorized_keys
```

Go to:
https://github.com/settings/keys

Add the key:
  Name: CICD
  Key: <paste key>

Add Github secrets for the mywebapp project:

See video: 14-github-secrets.mp4

```
cat ~/.ssh/id_rsa
```

Copy the text (SSH secret) shown.

Go to:
https://github.com/<your-github-username>/mywebapp/settings/secrets/actions

Add secrets:

```
DEPLOY_HOST: 46.101.247.241
DEPLOY_KEY: SSH secret (from cat ~/.ssh/id_rsa above)
DEPLOY_PORT: 22
DEPLOY_USER: hobnob
USERNAME: <your-github-username>
MAIN_HOST: sitebuilderkit.com
NEXTAUTH_URL: http://sitebuilderkit.com
NEXTAUTH_SECRET: INp3IvdIlaMcoGAgFGoA66DdCclzzSqnXJZkgz8PSzx
MONGODB_HOSTNAME: mongodb (use container name, not IP)
MONGODB_PORT: 27017
MONGODB_USERNAME: dbadmin
MONGODB_PASSWORD: a378ag4y7m4
MONGODB_DATABASE: nextsite
WEB_ASSETS_PATH: /app
WEB_ASSETS_URL: http://sitebuilderkit.com
```

Note: please change <your-github-username> with your github username.

Go to Actions tab and deploy.

Add Github secrets for the mywebassets project:

See video: 15-github-secrets.mp4

Go to:
https://github.com/<your-github-username>/mywebassets/settings/secrets/actions

Add secrets:

```
DEPLOY_HOST: 46.101.247.241
DEPLOY_KEY: paste SSH secret (from: cat ~/.ssh/id_rsa above)
DEPLOY_PORT: 22
DEPLOY_USER: hobnob
USERNAME: <your-github-username>
```

Go to Actions tab and deploy.

Check Docker & Add Write Permission

See video: 16-check-docker.mp4

```
docker ps
```

Note:
To run manually on the server:

```
docker run -v ~/uploads:/app/uploads/ --link mongodb:mongodb --env MAIN_HOST=sitebuilderkit.com --
env NEXTAUTH_URL=http://sitebuilderkit.com --env
NEXTAUTH_SECRET=INp3IvdIlaMcoGAgFGoA66DdCclzzSqnXJZkgz8PSzx --env MONGODB_USERNAME=dbadmin --env
MONGODB_PASSWORD=a378ag4y7m4 --env MONGODB_HOSTNAME=mongodb --env MONGODB_PORT=27017 --env
MONGODB_DATABASE=nextsite  --env WEB_ASSETS_PATH=/app --env WEB_ASSETS_URL=http://sitebuilderkit.com
--restart unless-stopped --name mywebapp -p 3000:3000 ghcr.io/<your-github-username>/mywebapp:main
```

Add write permission for file upload:

```
sudo chmod 0777 ~/uploads
```

Test

Before testing, we make a fix on video 14-github-secrets.mp4 when adding MONGODB_HOSTNAME.
MONGODB_HOSTNAME: <server-ip>
Should be:
MONGODB_HOSTNAME: mongodb (Docker container name)
You don't need to perform the fix if you follow the steps in this document.

See video: 17-fix-secret-test.mp4

To test, open:
http://46.101.247.241/api/test

You should see:
"MongoDB Connection Successful".

Install NGINX

See video: 18-install-nginx.mp4

```
sudo apt install nginx
```

```
sudo ufw app list
```

```
sudo ufw allow 'Nginx Full'
```

```
sudo vi /etc/nginx/sites-available/sitebuilderkit.com
```

Add:

```
server {
    listen 80;
```

```
    listen [::]:80;

    server_name sitebuilderkit.com www.sitebuilderkit.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
    }
    location /uploads/ {
        proxy_pass http://localhost:8081;
        proxy_set_header Host $host;
    }
}
```

```
sudo ln -s /etc/nginx/sites-available/sitebuilderkit.com /etc/nginx/sites-enabled/
```

```
sudo vi /etc/nginx/nginx.conf
```

Enable:

```
http {
    . . .
    server_names_hash_bucket_size 64;
    . . .
}
```

Check:

```
sudo nginx -t
```

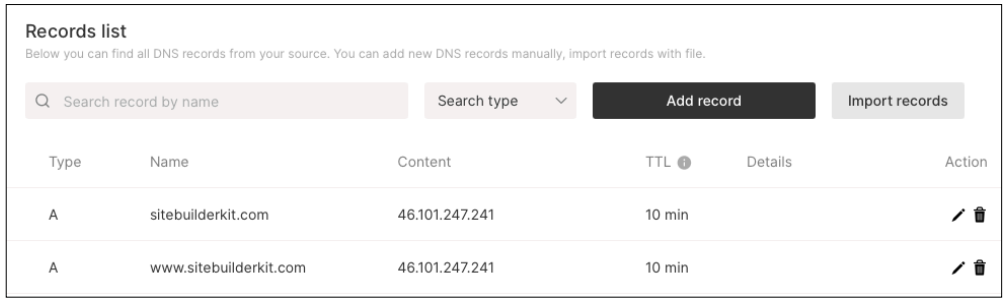
If there is no error, restart nginx:

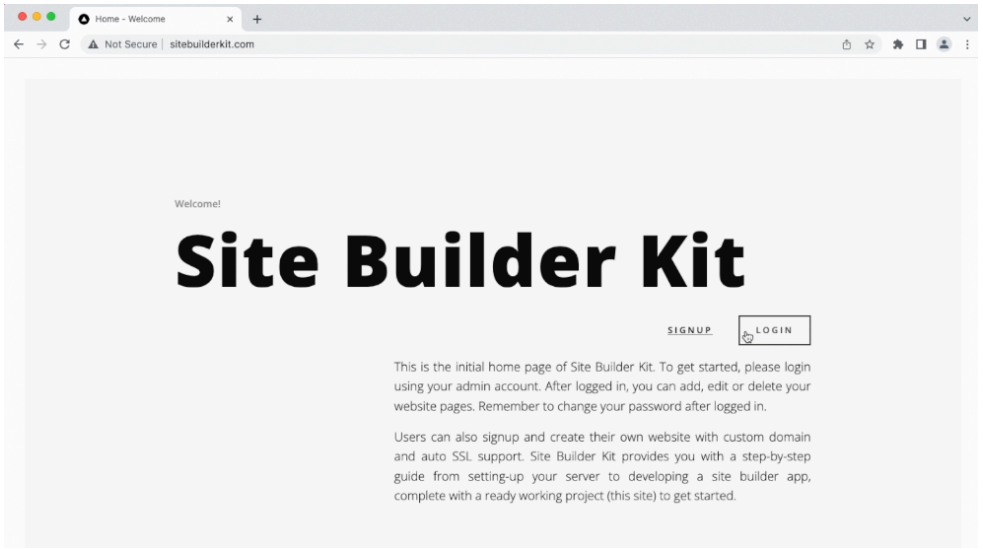
```
sudo systemctl restart nginx
```

Test:
http://sitebuilderkit.com/

See video: 19-test.mp4

Please make sure you have added an A record that points to your server IP (46.101.247.241) on your DNS service.





Install OpenResty

See video: 20-install-openresty.mp4

```
sudo systemctl disable nginx

sudo systemctl stop nginx

sudo apt-get -y install wget gnupg ca-certificates apt-utils curl

sudo wget -O - https://openresty.org/package/pubkey.gpg | sudo apt-key add -

echo "deb http://openresty.org/package/ubuntu $(lsb_release -sc) main" \    | sudo tee /etc/apt/
sources.list.d/openresty.list

sudo apt-get update

sudo apt-get -y install openresty

sudo apt-get -y install luarocks

sudo luarocks install lua-resty-auto-ssl
```

Note: if error: **sh: 1: make: not found** persists,
then, run:

```
sudo apt-get install build-essential
```

And re-run:

```
sudo luarocks install lua-resty-auto-ssl

sudo mkdir /var/log/openresty

sudo mkdir /etc/resty-auto-ssl

sudo chown www-data:www-data /etc/resty-auto-ssl

sudo openssl req -new -newkey rsa:2048 -days 3650 -nodes -x509 -subj '/CN=sni-support-required-for-valid-ssl' -keyout /etc/ssl/resty-auto-ssl-fallback.key -out /etc/ssl/resty-auto-ssl-fallback.crt

cd /etc/openresty

sudo rm nginx.conf

sudo vi nginx.conf
```

Paste:

```
user www-data;
events {
    worker_connections 1024;
}
http {
    lua_shared_dict auto_ssl 1m;
    lua_shared_dict auto_ssl_settings 64k;
    resolver 8.8.8.8 ipv6=off;

    init_by_lua_block {
       auto_ssl = (require "resty.auto-ssl").new()
       auto_ssl:set("allow_domain", function(domain)
           return true
       end)
```

```
        auto_ssl:set("ca", "https://acme-staging-v02.api.letsencrypt.org/directory")
        auto_ssl:init()
    }

    init_worker_by_lua_block {
        auto_ssl:init_worker()
    }

    server {
        listen 443 ssl;
        ssl_certificate_by_lua_block {
            auto_ssl:ssl_certificate()
        }
        ssl_certificate /etc/ssl/resty-auto-ssl-fallback.crt;
        ssl_certificate_key /etc/ssl/resty-auto-ssl-fallback.key;
    }
    server {
        listen 80;
        location /.well-known/acme-challenge/ {
            content_by_lua_block {
                auto_ssl:challenge_server()
            }
        }
    }
    server {
        listen 127.0.0.1:8999;
        client_body_buffer_size 128k;
        client_max_body_size 128k;
        location / {
            content_by_lua_block {
                auto_ssl:hook_server()
            }
        }
    }
}
```

```
sudo systemctl restart openresty
```

```
sudo systemctl status openresty
```

Test:

See video: 21-test.mp4

OpenResty welcome page opens, but the secure icon is not shown.

To fix:

See video: 22-continue.mp4

1. Update config (remove staging):

```
sudo vi /etc/openresty/nginx.conf
```

```
    #init_by_lua_block {
    #    auto_ssl = (require "resty.auto-ssl").new()
    #    auto_ssl:set("allow_domain", function(domain)
    #        return true
    #    end)
    #    auto_ssl:set("ca", "https://acme-staging-v02.api.letsencrypt.org/directory")
    #    auto_ssl:init()
    #}
    init_by_lua_block {
        auto_ssl = (require "resty.auto-ssl").new()
        auto_ssl:set("allow_domain", function(domain)
          return true
        end)
        auto_ssl:init()
    }
```

2. Remove & recreate folder

```
sudo rm -r /etc/resty-auto-ssl
```

```
sudo mkdir /etc/resty-auto-ssl
```

```
sudo chown www-data:www-data /etc/resty-auto-ssl
```

3. Restart

```
sudo systemctl restart openresty
```

4. Open: https://sitebuilderkit.com

See video: 23-https.mp4

Configure to open our app:

See video: 24-config.mp4

```
sudo vi /etc/openresty/nginx.conf
```

Update with the following:

```
user www-data;
events {
    worker_connections 1024;
}
http {
    lua_shared_dict auto_ssl 1m;
    lua_shared_dict auto_ssl_settings 64k;
    resolver 8.8.8.8 ipv6=off;

    #init_by_lua_block {
    #    auto_ssl = (require "resty.auto-ssl").new()
    #    auto_ssl:set("allow_domain", function(domain)
    #        return true
    #    end)
    #    auto_ssl:set("ca", "https://acme-staging-v02.api.letsencrypt.org/directory")
    #    auto_ssl:init()
    #}
    init_by_lua_block {
        auto_ssl = (require "resty.auto-ssl").new()
        auto_ssl:set("allow_domain", function(domain)
          return true
        end)
        auto_ssl:init()
    }
    init_worker_by_lua_block {
        auto_ssl:init_worker()
    }

    server {
        listen 443 ssl;
        ssl_certificate_by_lua_block {
```

```
            auto_ssl:ssl_certificate()
        }
        ssl_certificate /etc/ssl/resty-auto-ssl-fallback.crt;
        ssl_certificate_key /etc/ssl/resty-auto-ssl-fallback.key;

        location / {
            proxy_pass http://localhost:3000;
            proxy_set_header Host $host;
        }
        location /uploads/ {
            proxy_pass http://localhost:8081;
            proxy_set_header Host $host;
        }
    }
    server {
        listen 80;
        location /.well-known/acme-challenge/ {
            content_by_lua_block {
                auto_ssl:challenge_server()
            }
        }

        location / {
            return 301 https://$host$request_uri;
        }
    }
    server {
        listen 127.0.0.1:8999;
        client_body_buffer_size 128k;
        client_max_body_size 128k;
        location / {
            content_by_lua_block {
                auto_ssl:hook_server()
            }
        }
    }
}
```

```
sudo systemctl restart openresty
```

Test:

Open: https://sitebuilderkit.com

See video: 25-test.mp4

See video: 26-login.mp4

On the first start, an admin user account will be automatically created. You can login using:

Email: you@example.com
Password: demo

You can change your email and password after logging in.