**Create ReToken MERN full-stack project**

<https://www.youtube.com/watch?v=7CqJlxBYj-M>

<https://www.youtube.com/watch?v=SQqSMDIzhaE>

Install Docker

1. Install Docker Desktop https://hub.docker.com/editions/community/docker-ce-desktop-windows
2. In VS Code, search ‘docker’ in extensions, install Docker extension (so that docker can be run in npm terminal)

Create react application (for frontend)

1. In retoken-app folder, create client folder, cd to client folder
2. Create a react app
3. $ npx create-react-app retoken-app

Create & run a backend node project

1. In retoken-app folder, create server folder
2. Cd server
3. $ npm init (entry point = server.js)
4. Install all other packages needed for backend (express, nodemon, mongoose, dotenv, cors)
5. $ npm i express nodemon mongoose dotenv cors body-parser
6. Create server.js (app.listen = start the server)
7. $ nodemon server (Start the server)
8. Or update script in package.json, add - "start": "nodemon server.js", then server can be started with $ npm start

Connect to Mongo DB

1. Create account in Mongodb [www.mongodb.com/cloud/atlas](http://www.mongodb.com/cloud/atlas)
2. Create an organization (REToken) and a cluster (REToken)
3. Setup security
   1. add IP address (need 0.0.0.0/0 to connect from anywhere)
   2. create db user (retokenadmin/FinalProject999)
4. in Server.js, add mongo connection - const mongoose = require('mongoose')
5. Create .env file in server folder, copy connection string from Mongo Atlas for ATLAS\_URI variable, substitute with the user password
6. $ npm start (Start the server and connect to Mongo DB)

Create database schema (models & routes) – api access to db

1. Create models folder in server folder
2. Create property.model.js (currently only 1 doc (table), we will add more models when we need to store other information (e.g., userprofile)
3. Create routes folder in server folder
4. Create properties.js
5. Add router file into server.js

Test Mongo WRITE with server running:

1. Install extension REST Client in VS Code
2. Create testPost.http
3. Add POST command in testPost.http
4. Make sure server is running, click send request link at the top of testPost.http
5. Create testGet.http to test GET

Create docker images

1. Create Dockerfile in server folder with instruction to create the server container
2. Create Dockerfile in client folder with instruction to create the client container
3. Create docker-compose.yml to execute and create 3 images
   1. Server image from Dockerfile in server folder
   2. Mongo image pull standard mongo image from docker hub
   3. Client image from Dockerfile in client folder

Create Docker images

1. Create a network
   1. docker network create app-network
   2. docker network ls
2. create Mongo DB image and run on that network (in server folder) – only if you want to create a locally hosted Mongodb:
   1. docker volume create mongo-data
   2. docker volume create mongo-config
   3. docker run --name db -p 27017:27017 -v mongo-data:/data/db -v mongo-config:/data/configdb --network app-network -d mongo
   4. docker network inspect app-network – to check if 1 container is attached to network
   5. Can ctrl-c
   6. Docker ps – show container running
3. create server image and run on that network (in server folder):
   1. build - docker build -t server .
   2. run - docker run --name server -p 5000:5000 --network app-network -d server:latest
   3. docker logs server
   4. Npm start to run server, should connect to mongodb
4. create client image and run on that network (in client folder):
   1. build - docker build -t client .
   2. run - docker run --name client -p 3000:3000 --network app-network -d client:latest
   3. docker logs server
   4. Npm start to run server, should connect to mongodb
5. Use docker-compose to run (3) & (4)
   1. Create docker-compose.yml file
   2. Run docker-compose up
   3. Or to rebuild & run image at detach mode - docker-compose up -d --force-recreate
6. To go inside a container – docker exec -it containername bash

Publish Docker images to Docker hub

1. Create repo retoken-client and retaken-server in docker hub
2. In VS Code run docker login to login to docker hub
3. Tag local images to docker hub repos
   1. docker tag client:latest shpang/retoken-client:latest
   2. docker tag server:latest shpang/retoken-server:latest
4. Push image to docker hub
   1. docker push shpang/retoken-client:latest
   2. docker push shpang/retoken-server:latest

Create Git repo and connect to github

1. Configure a git account
   1. git config --global user.email "sohhar@gmail.com”
   2. git config --global user.name "Soh Har Pang"
2. Go to project directory (retoken)
3. Create a git repository git init (a .git directory is created)
4. Create a .gitignore file to not include folders and files (e.g. node\_modules, build etc)
5. Add all files to git repo (git add -A)
6. To see what files are added in stage to be committed (git diff --name-only --cached)
7. Commit repo
   1. Run git commit
   2. Update COMMIT\_EDITMSG file, close it to commit
8. Go to github
9. Log in to your account
10. Click the new repository button in the top-right. You’ll have an option there to initialize the repository with a README file, but I don’t.
11. Click the “Create repository” button.
12. Publish to github - Run
    1. git remote add origin <https://github.com/sohhar/retoken-app-git>
    2. git branch -M master
    3. git push -u origin master

Process to push github with changes

* git add -A . filename (new file to be added, if any)
* git commit -m “your comment on changes”
* ggpush

To pull from github into your local repository –

<https://code.visualstudio.com/docs/editor/github>

https://kbroman.org/github\_tutorial/pages/fork.html

**Other GIT Commands**

List files that are tracking in git

git ls-tree -r master --name-only

# Remove the files from the index (not the actual files in the working copy)

$ git rm -r --cached

# Add these removals to the Staging Area...

$ git add

# ...and commit them!

$ git commit -m "Clean up ignored files"

**How can I push my local GIT repositories to my GitHub organization?**

*### Step 1. Make sure you have a local copy of all "old repo"*

*### branches and tags.*

*# Fetch all of the remote branches and tags:*

**git fetch origin**

*# View all "old repo" local and remote branches:*

**git branch -a**

*# If some of the remotes/ branches doesn't have a local copy,*

*# checkout to create a local copy of the missing ones*:

**git checkout -b <branch> origin/<branch>**

*# Now we have to have all remote branches locally.*

*### Step 2. Add a "new repo" as a new remote origin:*

**git remote add new-origin**[**git@github.com**](mailto:///git%40github.com)**:user/repo.git**

*### Step 3. Push all local branches and tags to a "new repo".*

*# Push all local branches (note we're pushing to new-origin):*

**git push --all new-origin**

*# Push all tags:*

**git push --tags new-origin**

*### Step 4. Remove "old repo" origin and its dependencies.*

*# View existing remotes (you'll see 2 remotes for both fetch and push)*

**git remote -v**

*# Remove "old repo" remote:*

**git remote rm origin**

*# Rename "new repo" remote into just 'origin':*

**git remote rename new-origin origin**

**Deploy to Heroku**

<https://www.youtube.com/watch?v=4axmcEZTE7M>

<https://www.youtube.com/watch?v=Z9SJTEC0wEs>

https://www.youtube.com/watch?v=71wSzpLyW9k

<https://www.youtube.com/watch?v=2AIL1c-cJM0&list=PLhuTX2Fe98b6JI6mm7litU7pS23tvh1XK&index=24>

<https://medium.com/@basics.aki/deploy-a-mern-stack-application-to-heroku-b9a98b569469>

<https://yflooi.medium.com/deploy-a-react-app-to-heroku-with-docker-ca4d9bd811c6>

https://developer.okta.com/blog/2020/06/24/heroku-docker-react

1. Create account in Heroku
2. Create pipeline (retoken), create app (retoken-app) in pipeline
3. Set environment variables (e.g., PORT, MONGO connection), go to retaken-app setting -> Config vars, add

PORT = 500

ATLAS\_URI = mongodb+srv://dbuser:dbuser@cluster0.zthpw.mongodb.net/retokendb?retryWrites=true&w=majority

NODE\_ENV = production

1. in Heroku app->Deploy, enable autodeploy, so every push is deployed automatically
2. Install Heroku in VS Code

npm i -g heroku

1. install Heroku CLI extension in VS Code
2. in VS Code Terminal, set the stack of our app to container so that Heroku knows that we are deploying a container –

heroku stack:set container -app retoken-app

1. add and existing Heroku app in git remote repo

heroku git:remote -a retoken-app

or create a new Heroku app

Heroku create retokenapp

**Docker deploy by Container Registry & Runtime**

1. create a Dockefile at project root to install & deploy both server and client

docker build -t retoken-app .

docker tag retoken-app registry.heroku.com/retoken-app/web

docker push registry.heroku.com/retoken-app/web:latest

heroku container:release web --remote heroku

**Build using Dockerfile & Heroku.yml and deploy with git**

1. In retoken-app project folder, create heroku.yml and use Dockerfile in root

git add -a .

git commit -m ‘commit for heroku deployment’

git push origin/heroku master

heroku login

heroku container:login

heroku stack:set container -a <herokuAppName>

git push heroku master

heroku ps:scale web=1

heroku open

git add -A .

git commit -m ‘message’

git push origin master

docker build -t retokenapp .

docker run --name retokenapp -p 5000:5000 --network app-network -d retokenapp:latest

docker tag retokenapp:latest shpang/retokenapp:latest

docker push shpang/retokenapp:latest

docker tag retokenapp:latest registry.heroku.com/retokenapp/web:latest

docker push registry.heroku.com/retokenapp/web:latest

heroku container:release web --remote heroku

heroku open

**Smart Contract**

npm I -g truffle

**other dependencies:**

"@openzeppelin/contracts-upgradeable": "^3.4.1",

"@truffle/contract": "^4.3.8",

"@truffle/hdwallet-provider": "^1.2.3"

**Compile**

* Update truffle-config.js file

Truffle compile

Truffle migrate

Configure remix

Share the dir where node\_modules is going to be

remixd -s c:/bccode/retoken-app/client

$ remixd -s c:/bccode/retoken-app

use injected web3, use network address found in JSON

