Deploying a MERN application (with Docker, Atlas and Digital OCEAN)

- **Part 1: Dockerize**
  - Run baseline application
  - Dockerize api server
  - Dockerize react client
  - Set up docker compose
  - Find public mongo image
  - Enable hot reloading by mounting in src
- **Part 2: Productionize**
  - Break out seperate docker compose files
  - Move db to Mongo Atlas
  - Update Client Dockerfile to build production version
  - Use Caddy to serve front end files
  - Parameterize connection strings
  - Split local and production configurations
- **Part 3: Deployment**
  - Create Digital Ocean VM
  - Configure DNS
  - Configure network access in Atlas
  - Configure Caddy
  - Deploy

- Dockerfile for the backend side
(.dockerignore: node-modules)

- makefile
(.dockerignore: node-modules)

server > Makefile

```
build:
	docker build -t api-server .
```

Make build

server > Dockerfile > ...

```
FROM node:14-slim

WORKDIR /usr/src/app

COPY ./package.json ./
COPY ./yarn.lock ./

RUN yarn install

COPY . .

EXPOSE 5000

CMD [ "yarn", "start" ]
```

- Dockerfile for the frontend side
(.dockerignore: node-modules)

client > Makefile

```
build:
    docker build -t react-app .
```

Make build

Client image and Server image are build

client > Dockerfile > ...

```
FROM node:14-slim

WORKDIR /usr/src/app

COPY ./package.json ./
COPY ./yarn.lock ./

RUN yarn install

COPY . .

EXPOSE 3000

CMD [ "yarn", "start" ]
```

Dockercompose at the top level

docker-compose.yml > {} services

```yaml
version: "3"
services:
  react-app:
    image: react-app
    build: ./client/
    stdin_open: true
    ports:
      - "3000:3000"
    networks:
      - mern-app
  api-server:
    image: api-server
    build: ./server/
    ports:
      - "5000:5000"
    networks:
      - mern-app
    depends_on:
      - mongo
```

```yaml
  mongo:
    image: mongo:4.4-bionic
    ports:
      - "27017:27017"
    networks:
      - mern-app
    volumes:
      - mongo-data:/data/db
networks:
  mern-app:
    driver: bridge
volumes:
  mongo-data:
    driver: local
```

```
const connectionString = 'mongodb://mongo:27017/cinema'
```

Makefile

```makefile
run-dev:
	docker-compose up
```

Make run-dev

Dockercompose at the top level

docker-compose.yml > {} services

```yaml
version: "3"
services:
  react-app:
    image: react-app
    build: ./client/
    stdin_open: true
    ports:
      - "3000:3000"
    networks:
      - mern-app
  api-server:
    image: api-server
    build: ./server/
    ports:
      - "5000:5000"
    networks:
      - mern-app
    depends_on:
      - mongo
```

```yaml
  mongo:
    image: mongo:4.4-bionic
    ports:
      - "27017:27017"
    networks:
      - mern-app
    volumes:
      - mongo-data:/data/db
networks:
  mern-app:
    driver: bridge
volumes:
  mongo-data:
    driver: local
```

```
const connectionString = 'mongodb://mongo:27017/cinema'
```

Makefile

```makefile
run-dev:
	docker-compose up
```

Make run-dev

```yaml
volumes:
  - ./client/:/usr/src/app
  - /usr/src/app/node_modules
```

```yaml
volumes:
  - ./server/:/usr/src/app
  - /usr/src/app/node_modules
```

Melting the app and the source code

Dockercompose-dev at the top level

Dockercompose-production at the top level

- Remove build/ volumes
- Remove mongo db
- http/https

docker-compose-production.yml > {} services

```yaml
version: "3"
services:
  react-app:
    image: react-app-production
    restart: unless-stopped
    stdin_open: true
    ports:
      - "80:80"
      - "443:443"
    networks:
      - mern-app
  api-server:
    image: api-server
    restart: unless-stopped
    ports:
      - "5000:5000"
    networks:
      - mern-app
networks:
  mern-app:
    driver: bridge
volumes:
  mongo-data:
    driver: local
```

client > Makefile

```makefile
build:
  docker build -t react-app .

build-production:
  docker build -t react-app-production -f Dockerfile.production .
```

client > Dockerfile.production > ...

```dockerfile
FROM node:14-slim

WORKDIR /usr/src/app

COPY ./package.json ./
COPY ./yarn.lock ./

RUN yarn install

COPY . .

RUN yarn build

### Copy into secondary Caddy stage
```

- server
  - config
    - dev.env
    - local.env
    - production.env

server > config > dev.env

```
MONGO_URI='mongodb://mongo:27017/cinema'
```

```
const connectionString = process.env.MONGO_URI
```

server > JS index.js > ...

```
const express = require('express')
var require: NodeRequire ('body-parser')
(id: string) => any
require('dotenv').config()
```

client > Makefile

```makefile
build-dev:
  docker build -t react-app .

build-local:
  docker build -t react-app-production -f Dockerfile.local .

build-production:
  docker build -t react-app-production -f Dockerfile.production .
```

build-dev

Makefile

```makefile
build-dev:
  cd client && $(MAKE) build-dev
  cd server && $(MAKE) build

run-dev:
  docker-compose up

###

build-local:
  cd client && $(MAKE) build-local
  cd server && $(MAKE) build

run-local:
  docker-compose up

###

build-production:
  cd client && $(MAKE) build-production
  cd server && $(MAKE) build

run-production:
  docker-compose up
```

client > Dockerfile.production > ...

```dockerfile
### first stage
FROM node:14-slim

WORKDIR /usr/src/app

COPY ./package.json ./
COPY ./yarn.lock ./

RUN yarn install

COPY . .

RUN yarn build

### second stage
FROM caddy:2.1.1-alpine

ARG CADDYFILE
COPY ${CADDYFILE} /etc/caddy/Caddyfile
```

client > Makefile

```makefile
build-dev:
  docker build -t react-app .

build-local:
  docker build \
    -t react-app-production:local \
    --build-arg CADDYFILE=Caddyfile.local \
    --build-arg BASE_URL=http://localhost:5000/api \
    -f Dockerfile.production .

build-production:
  docker build -t react-app-production -f Dockerfile.production .
```

client > Caddyfile.local

```caddyfile
http://localhost:80 {
  root * /srv
  route {
    reverse_proxy /api* api-server:5000
    try_files {path} {path}/ /index.html
    file_server
  }
}
```

client > src > api > index.js > baseURL

```index.js
import axios from 'axios'

const baseURL = process.env.REACT_APP_BASE_URL
```

client > Dockerfile.production > ...

```dockerfile
### first stage
FROM node:14-slim AS builder

WORKDIR /usr/src/app

COPY ./package.json ./
COPY ./yarn.lock ./

RUN yarn install

COPY . .

ARG BASE_URL
ENV REACT_APP_BASE_URL=${BASE_URL}

RUN yarn build
```

abc builder

The name of

Set the base
give this buil

```dockerfile
### second stage
FROM caddy:2.1.1-alpine

ARG CADDYFILE
COPY ${CADDYFILE} /etc/caddy/Caddyfile

COPY --from=builder /usr/src/app/build/ /srv

EXPOSE 80

EXPOSE 443
```

Where the website files are hosted

client > Dockerfile > _

```
FROM node:14-slim

WORKDIR /usr/src/app

COPY ./package.json ./
COPY ./yarn.lock ./

RUN yarn install

COPY . .

EXPOSE 3000

ENV REACT_APP_BASE_URL=http://localhost:5000/api

CMD [ "yarn", "start" ]
```

client > Caddyfile.production

```
mern.mysuperawesomesite.com:443 {
  tls sid.palas@gmail.com
  root * /srv
  route {
    reverse_proxy /api* api-server:5000
    try_files {path} {path}/ /index.html
    file_server
  }
}
```

client > Makefile

```
build-dev:
  docker build -t react-app .

build-local:
  docker build \
    -t react-app-production:local \
    --build-arg CADDYFILE=Caddyfile.local \
    --build-arg BASE_URL=http://localhost:5000/api \
    -f Dockerfile.production .

build-production:
  docker build \
  -t react-app-production:production \
  --build-arg CADDYFILE=Caddyfile.production \
  --build-arg BASE_URL=https://mern.mysuperawesomesite.com/api \
  -f Dockerfile.production .
```

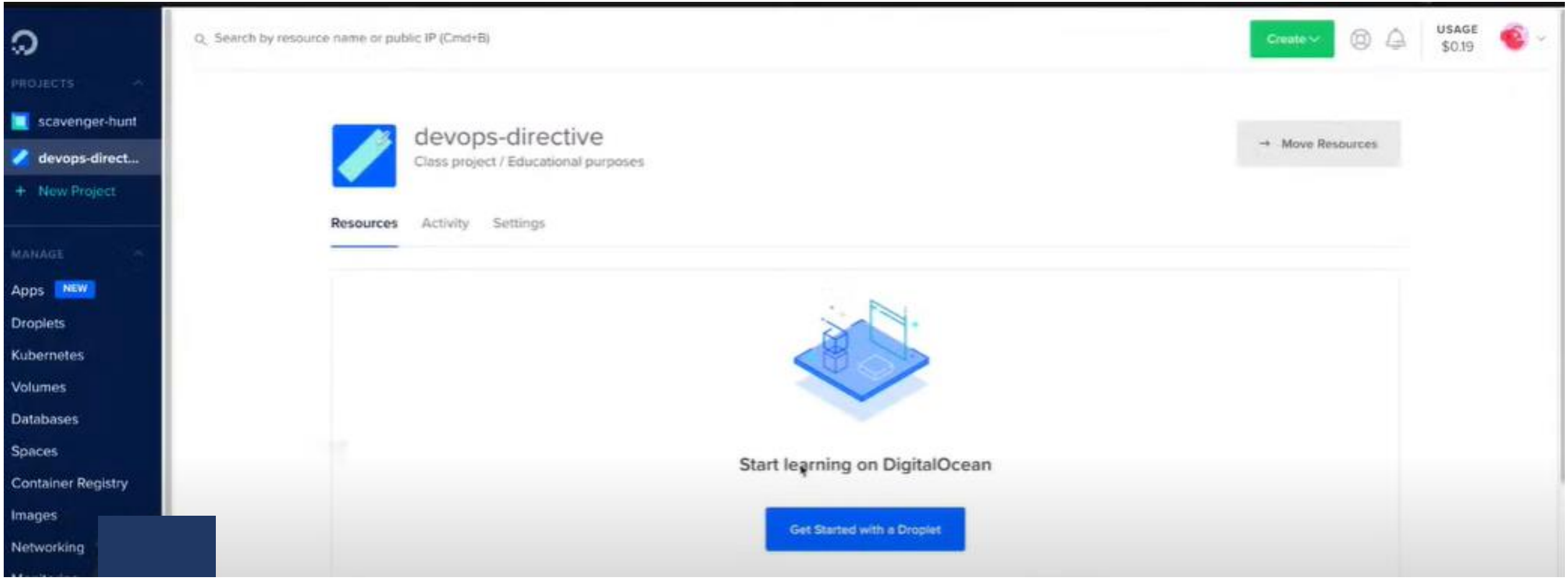
Makefile

```makefile
build-dev:
  cd client && $(MAKE) build-dev
  cd server && $(MAKE) build

run-dev:
  docker-compose -f docker-compose-dev.yml up

###

build-local:
  cd client && $(MAKE) build-local
  cd server && $(MAKE) build

run-local:
  ENV=local docker-compose -f docker-compose-production.yml up

###

build-production:
  cd client && $(MAKE) build-production
  cd server && $(MAKE) build

run-production:
  ENV=production docker-compose -f docker-compose-production.yml up
```

docker-compose-production.yml > {} services > {} react-app

```yaml
version: "3"
services:
  react-app:
    image: react-app-production:${ENV}
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    networks:
      - mern-app
  api-server:
    image: api-server
    restart: unless-stopped
    env_file: ./server/config/${ENV}.env
    ports:
      - "5000:5000"
    networks:
      - mern-app
networks:
  mern-app:
    driver: bridge
volumes:
  mongo-data:
    driver: local
```

localhost:80


Search by resource name or public IP (Cmd+B)
Create
USAGE
$0.19
PROJECTS
scavenger-hunt
devops-direct...
New Project
MANAGE
Apps
NEW
Droplets
Kubernetes
Volumes
Databases
Spaces
Container Registry
Images
Networking
devops-directive
Class project / Educational purposes
Move Resources
Resources
Activity
Settings
Start learning on DigitalOcean
Get Started with a Droplet

Create Droplets

Choose an image

Distributions | Container distributions | Marketplace | Custom images

Ubuntu	FreeBSD	Fedora	Debian	CentOS
20.04 (LTS) x64	Select version	Select version	Select version	Select version

Choose a plan

Help me choose

SHARED CPU	DEDICATED CPU			
Basic	General Purpose	CPU-Optimized	Memory-Optimized	Storage-Optimized NEW

Create Droplets

Choose an image

Distributions | Container distributions | Marketplace | Custom images

Search keyword

See all Marketplace Apps →

Recommended for you

WordPress 5.5.1 on Ubuntu 20.04 Details	OpenVPN Access Server 2.8.5 o... Details	Docker 19.03.12 on Ubuntu 20.04 Details
LAMP on Ubuntu 20.04 Details	Jitsi Server 2.1-273 on Ubuntu 18.04 Details	Plesk 18.0 on Ubuntu 18.04 Details

Choose a plan

Help me choose

SHARED CPU	DEDICATED CPU			
Basic	General Purpose	CPU-Optimized	Memory-Optimized	Storage-Optimized NEW

Basic virtual machines with a mix of memory and compute resources. Best for small projects that can handle variable levels of CPU performance, like blogs, web apps and dev/test environments.

$5/mo	$10/mo	$15/mo	$20/mo	$40/mo	$80/mo
$0.007/hour	$0.015/hour	$0.022/hour	$0.030/hour	$0.060/hour	$0.119/hour
1 GB / 1 CPU	2 GB / 1 CPU	2 GB / 2 CPUs	4 GB / 2 CPUs	8 GB / 4 CPUs	16 GB / 8 CPUs
25 GB SSD Disk	50 GB SSD Disk	60 GB SSD Disk	80 GB SSD Disk	160 GB SSD Disk	320 GB SSD Disk
1000 GB transfer	2 TB transfer	3 TB transfer	4 TB transfer	5 TB transfer	6 TB transfer

Select additional options

- [ ] IPv6
- [ ] User data
- [ ] Monitoring

Authentication

- (•) **SSH keys** — A more secure authentication method
- ( ) **Password** — Create a root password to access Droplet (less secure)

Choose your SSH keys

- [x] sid macbook pro

New SSH Key

Search by resource name or public IP (Cmd+B)

Create

USAGE
$0.19

docker-ubuntu-s-2vcpu-2gb-nyc3-01 was added to devops-directive.

devops-directive

Class project / Educational purposes

Resources Activity Settings

DROPLETS (1)

docker-ubuntu-s-2vcpu-2gb-nyc3-01

Create something new

Create a Managed Database
Worry-free database management

Start using Spaces
Deliver data with scalable object storage

Learn more

Product Docs
Technical overviews, how-tos, release notes, and support material

Search by resource name or public IP (Cmd+B)

Create

USAGE
$0.19

Networking

Domains Floating IPs Load Balancers VPC Firewalls PTR records

Create Firewall

Name	Droplets	Rules	Created	
http-traffic	0	6	15 days ago	More

http-traffic
6 Rules / 0 Droplets

Rules Droplets Destroy

Firewall rules control what inbound and outbound traffic is allowed to enter or leave a Droplet.

Inbound Rules

Set the Firewall rules for incoming traffic. Only the specified ports will accept inbound connections. All other traffic will be blocked.

Type	Protocol	Port Range	Sources	
SSH	TCP	22	All IPv4 All IPv6	More
HTTP	TCP	80	All IPv4 All IPv6	More
HTTPS	TCP	443	All IPv4 All IPv6	More

New rule

http-traffic
6 Rules / 0 Droplets

Rules Droplets Destroy

Add Droplet

docker-ubuntu-s-2vcpu-2gb-nyc3-01

Search for a Droplet or a tag

Add Droplet

Choose Droplets

Your Firewall isn't applied to any Droplet. Select to which ones it should apply or use tags to select groups of Droplets.

Search by resource name or public IP (Cmd+B)

Create

USAGE
$0.19

PROJECTS

scavenger-hunt

devops-directi...

+ New Project

MANAGE

Apps NEW

Droplets

Kubernetes

Volumes

Databases

Spaces

Container Registry

docker-ubuntu-s-2vcpu-2gb-nyc3-01

in devops-directive / 2 GB Memory / 60 GB Disk / NYC3 - Docker 19.03.12 on Ubuntu 20.04

ON

ipv4: 104.131.22.28 ipv6: Enable now Private IP: 10.108.0.2 Floating IP: Enable now Metrics agent: ••• Console:

Graphs
Access
Power
Volumes
Resize
Networking
Backups
Snapshots
Kernel
History

NEW! Upgrade your Droplet for additional metrics and alerting.

Learn How to Update

docker-ubuntu-s-2vcpu-2gb-nyc3-01

in devops-directive / 2 GB Memory / 60 GB Disk / NYC3 - Docker 19.03.12 on Ubuntu 20.04

ON

ipv4: 104.131.22.28 Copy ipv6: Enable now Private IP: 10.108.0.2 Floating IP: Enable now Metrics agent: ••• Console:

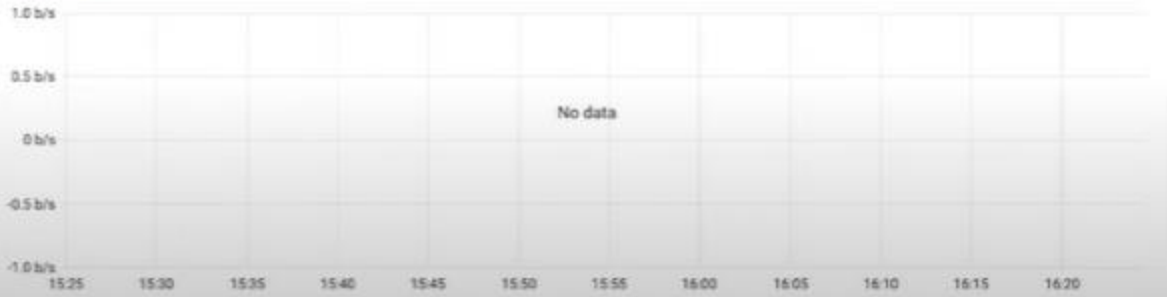
Graphs
Access
Power
Volumes
Resize
Networking
Backups
Snapshots
Kernel
History
Destroy
Tags
Recovery

NEW! Upgrade your Droplet for additional metrics and alerting.

Learn How to Update

Last 1 hour

Bandwidth



CPU Usage

Sid's Org - 2020-09-11 | Access Manager | Support | Billing | All Clusters | Sid

Project 0 | Atlas | Realm | Charts

DATA STORAGE
- Clusters
- Triggers
- Data Lake

SECURITY
- Database Access
- Network Access
- Advanced

SID'S ORG - 2020-09-11 > PROJECT 0

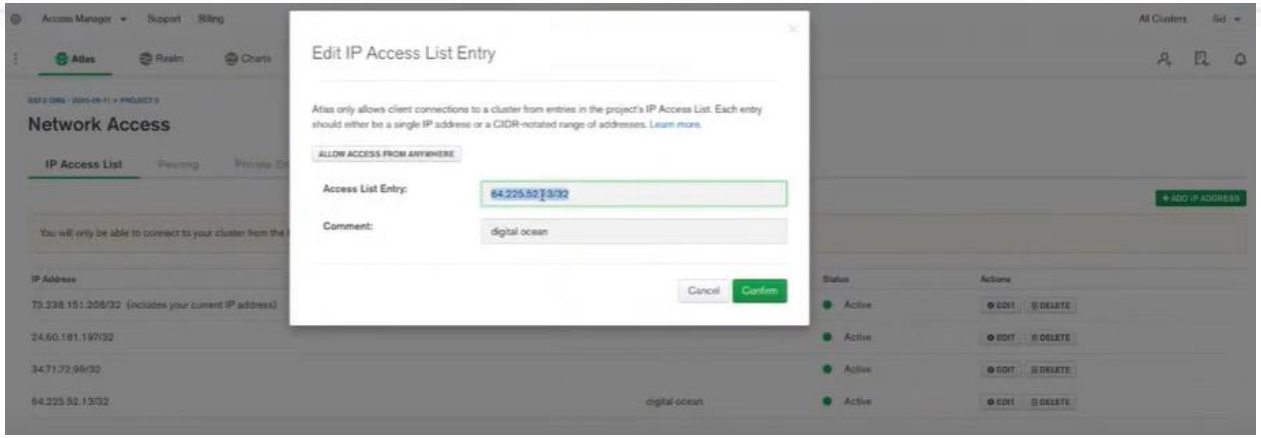
Network Access

IP Access List | Peering | Private Endpoint

+ ADD IP ADDRESS

You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment	Status	Actions
73.238.151.208/32 (includes your current IP address)	Norton	Active	EDIT DELETE
24.60.181.197/32		Active	EDIT DELETE
34.71.72.99/32		Active	EDIT DELETE
64.225.52.13/32	digital ocean	Active	EDIT DELETE



Makefile

```
7
8   ###
9
10  build-local:
11    cd client && $(MAKE) build-local
12    cd server && $(MAKE) build
13
14  run-local:
15    ENV=local docker-compose -f docker-compose-production.yml up
16
17  ###
18
19  build-production:
20    cd client && $(MAKE) build-production
21    cd server && $(MAKE) build
22
23  run-production:
24    ENV=production docker-compose -f docker-compose-production.yml up
25
26  SSH_STRING:=root@104.131.22.28
27
28  ssh:
29    ssh $(SSH_STRING)
```

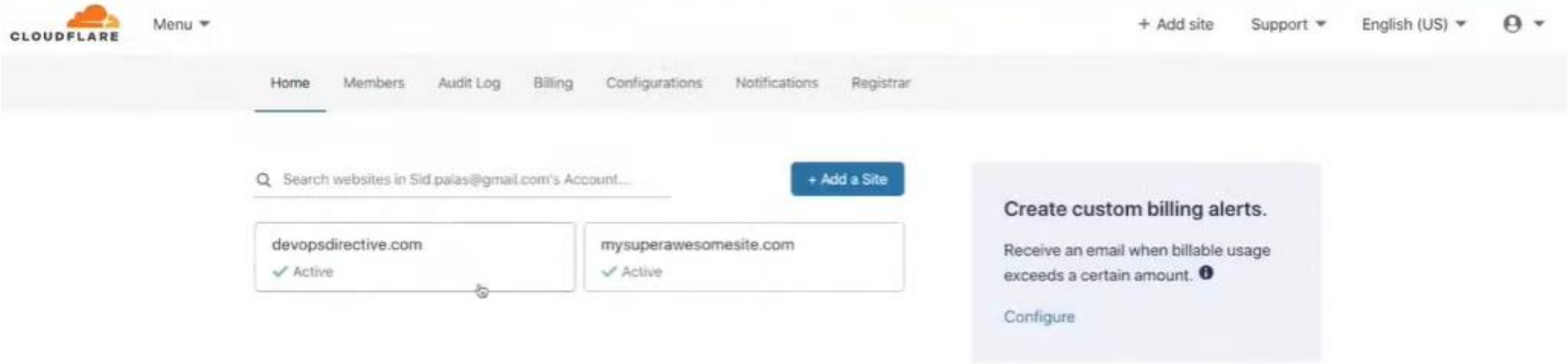
make ssh

Cloudflare est conçu pour sécuriser l'ensemble des équipements que vous connectez à Internet de manière privée, rapide et fiable.

•Sécurisez vos sites web, API et applications Internet.

•Protégez vos réseaux d'entreprises, vos employés et vos appareils.

•Rédigez et déployez du code qui s'exécute à la périphérie du réseau.



CLOUDFLARE mysuperawesomesite.com + Add site Support English (US)

Overview Analytics DNS SSL/TLS Firewall Access Speed Caching Workers Page Rules Network Traffic Stream Custom P... Apps Scrape S...

24 Hours 7 Days 30 Days

Unique Visitors

—

Total Requests

—

Percent Cached

Quick Actions

Purge Cache

DNS Settings

Under Attack Mode

Show visitors a JavaScript challenge when visiting your site

Off

Development Mode

Temporarily bypass our cache. See changes to your origin server in realtime.

Off

CLOUDFLARE mysuperawesomesite.com + Add site Support English (US)

✓ Add an MX record for your **root domain** so that mail can reach **@mysuperawesomesite.com** addresses.

DNS management for **mysuperawesomesite.com**

+ Add record | Search DNS Records | Advanced

Type	Name	Content	TTL	Proxy status	
CNAME	_domainconnect	connect.domains.google.com	Auto	Proxied	Edit
CNAME	mysuperawesomesite.com	c.storage.googleapis.com	Auto	Proxied	Edit

DNS management for **mysuperawesomesite.com**

+ Add record | Search DNS Records | Advanced

mern.mysuperawesomesite.com points to **104.131.22.28** and has its traffic proxied through Cloudflare.

Type	Name	IPv4 address	TTL	Proxy status
A	mern	104.131.22.28	Auto	Proxied

Cancel | Save

CLOUDFLARE | mysuperawesomesite.com | + Add site | Support | English (US)

Overview | Analytics | DNS | SSL/TLS | Firewall | Access | Speed | Caching | Workers | Page Rules | Network | Traffic | Stream | Custom P... | Apps | Scrape S...

Overview | Edge Certificates | Client Certificates | Origin Server | Custom Hostnames

We added a Recommender to SSL/TLS settings. Tell us what you think. ×

Your SSL/TLS encryption mode is Flexible

This setting was last changed a few seconds ago

- Off (not secure)
  No encryption applied
- **Flexible**
  Encrypts traffic between the browser and Cloudflare
- Full
  Encrypts end-to-end, using a self signed certificate on the server
- Full (strict)

```
copy-files:
  scp -r ./* $(SSH_STRING):/root/
```

make build-production

```
root@docker-ubuntu-s-2vcpu-2gb-nyc3-01:~# docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED
            SIZE
api-server              latest              bfd76ecc01de        12 secon
ds ago      193MB
react-app-production    production          017ae89db5dc        21 secon
ds ago      44.9MB
<none>                  <none>              3c8ed7fe9c6c        24 secon
```

make run-production