Setting up simple project

https://github.com/shaojieyew/docker-react

Project: React
Github → Travis → AWS beanstalk

Dockerfile

```
// copy over the package to perform installation and then copy over the other context. This is to make sure npm install does not run everytime context is changed, as it is after the npm install line
FROM node:alpine as builder
WORKDIR '/app'
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx
EXPOSE 80
COPY --from=builder /app/build /usr/share/nginx/html
```

Dockerfile.dev

```
FROM node:alpine
WORKDIR '/app'
COPY package.json .
RUN npm install

COPY . .
CMD ["npm", "run", "start"]
```

.travis.yml

```
sudo: required
services:
  - docker

before_install:
  - docker build -t stephengrider/docker-react -f Dockerfile.dev .

script:
  - docker run stephengrider/docker-react npm run test -- --coverage

deploy:
  provider: elasticbeanstalk
  region: "us-west-2"
```

```
app: "docker"
env: "Docker-env"
bucket_name: "elasticbeanstalk-us-west-2-306476627547"
bucket_path: "docker"
on:
  branch: master
access_key_id: $AWS_ACCESS_KEY
secret_access_key:
  secure: "$AWS_SECRET_KEY"
```

Setting up Multiple container locally

https://github.com/shaojieyew/multi-docker

Client

Dockerfile

```
FROM node:alpine as builder
WORKDIR '/app'
COPY ./package.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx
EXPOSE 3000
COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/build /usr/share/nginx/html
```

Dockerfile.dev

```
FROM node:alpine
WORKDIR '/app'
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "start"]
```

Nginx

Nginx is used for routing

Dockerfile

```
FROM nginx
COPY ./default.conf /etc/nginx/conf.d/default.conf
```

Dockerfile.dev

```
FROM nginx
COPY ./default.conf /etc/nginx/conf.d/default.conf
```

Default.conf: setting for nignx

```
upstream client {
  server client:3000;
}
upstream api {
  server api:5000;
}
server {
  listen 80;
  location / {
    proxy_pass http://client;
  }
  location /sockjs-node {
    proxy_pass http://client;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
  }
  location /api {
    rewrite /api/(.*) /$1 break;
    proxy_pass http://api;
  }
}
```

Server

Dockerfile

```
FROM node:alpine
WORKDIR "/app"
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "start"]
```

Dockerfile.dev

```
FROM node:alpine
WORKDIR "/app"
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "dev"]
```

Worker

Dockerfile

```
FROM node:alpine
WORKDIR "/app"
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "start"]
```

Dockerfile.dev

```
FROM node:alpine
WORKDIR "/app"
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "dev"]
```

Travis

.travis.yml

```
sudo: required
services:
  - docker

before_install:
  - docker build -t stephengrider/react-test -f ./client/Dockerfile.dev ./client

script:
  - docker run stephengrider/react-test npm test -- --coverage

after_success:
  - docker build -t stephengrider/multi-client ./client
  - docker build -t stephengrider/multi-nginx ./nginx
  - docker build -t stephengrider/multi-server ./server
  - docker build -t stephengrider/multi-worker ./worker
  # Log in to the docker CLI
  - echo "$DOCKER_PASSWORD" | docker login -u "$DOCKER_ID" --password-stdin
  # Take those images and push them to docker hub
  - docker push stephengrider/multi-client
  - docker push stephengrider/multi-nginx
  - docker push stephengrider/multi-server
  - docker push stephengrider/multi-worker

deploy:
```

```
  provider: elasticbeanstalk
  region: us-west-1
  app: multi-docker
  env: MultiDocker-env
  bucket_name: elasticbeanstalk-us-west-1-306476627547
  bucket_path: docker-multi
  on:
    branch: master
  access_key_id: $AWS_ACCESS_KEY
  secret_access_key:
    secure: $AWS_SECRET_KEY
```

docker-compose.yml

```
version: '3'
services:
  postgres:
    image: 'postgres:latest'
  redis:
    image: 'redis:latest'
  nginx:
    restart: always
    build:
      dockerfile: Dockerfile.dev
      context: ./nginx
    ports:
      - '3050:80'
  api:
    build:
      dockerfile: Dockerfile.dev
      context: ./server
    volumes:
      - /app/node_modules
      - ./server:/app
    environment:
      - REDIS_HOST=redis
      - REDIS_PORT=6379
      - PGUSER=postgres
      - PGHOST=postgres
      - PGDATABASE=postgres
      - PGPASSWORD=postgres_password
      - PGPORT=5432
  client:
    build:
      dockerfile: Dockerfile.dev
```

```
    context: ./client
  volumes:
    - /app/node_modules
    - ./client:/app
worker:
  environment:
    - REDIS_HOST=redis
    - REDIS_PORT=6379
  build:
    dockerfile: Dockerfile.dev
    context: ./worker
  volumes:
    - /app/node_modules
    - ./worker:/app
```

Dockerrun.aws.json

Use by aws elastic beanstalk for multiple container

```
{
  "AWSEBDockerrunVersion": 2,
  "containerDefinitions": [
    {
      "name": "client",
      "image": "stephengrider/multi-client",
      "hostname": "client",
      "essential": false,
      "memory": 128
    },
    {
      "name": "server",
      "image": "stephengrider/multi-server",
      "hostname": "api",
      "essential": false,
      "memory": 128
    },
    {
      "name": "worker",
      "image": "stephengrider/multi-worker",
      "hostname": "worker",
      "essential": false,
      "memory": 128
    },
    {
      "name": "nginx",
      "image": "stephengrider/multi-nginx",
      "hostname": "nginx",
```

```
      "essential": true,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80
        }
      ],
      "links": ["client", "server"],
      "memory": 128
    }
  ]
}
```

Kubernetes with multiple containers pod

https://github.com/shaojieyew/multi-k8s

client-cluster-ip-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: client-cluster-ip-service
spec:
  type: ClusterIP
  selector:
    component: web
  ports:
    - port: 3000
      targetPort: 3000
```

client-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: client-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      component: web
  template:
    metadata:
      labels:
```

```
      component: web
  spec:
    containers:
      - name: client
        image: stephengrider/multi-client
        ports:
          - containerPort: 3000
```

worker-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: worker-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      component: worker
  template:
    metadata:
      labels:
        component: worker
    spec:
      containers:
        - name: worker
          image: stephengrider/multi-worker
          env:
            - name: REDIS_HOST
              value: redis-cluster-ip-service
            - name: REDIS_PORT
              value: '6379'
```

Server-cluster-ip-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: server-cluster-ip-service
spec:
  type: ClusterIP
  selector:
    component: server
  ports:
    - port: 5000
      targetPort: 5000
```

Server-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: server-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      component: server
  template:
    metadata:
      labels:
        component: server
    spec:
      containers:
        - name: server
          image: stephengrider/multi-server
          ports:
            - containerPort: 5000
          env:
            - name: REDIS_HOST
              value: redis-cluster-ip-service
            - name: REDIS_PORT
              value: '6379'
            - name: PGUSER
              value: postgres
            - name: PGHOST
              value: postgres-cluster-ip-service
            - name: PGPORT
              value: '5432'
            - name: PGDATABASE
              value: postgres
            - name: PGPASSWORD
              valueFrom:
                secretKeyRef:
                  name: pgpassword
                  key: MYPASSWORD
```

```
// create keyvalue secret
Kubectl create secret generic pgpassword --from-literal MYPASSWORD=mypgpassword123
```

Postgres-cluster-ip-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: postgres-cluster-ip-service
```

```
spec:
  type: ClusterIP
  selector:
    component: postgres
  ports:
    - port: 5432
      targetPort: 5432
```

Postgres-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      component: postgres
  template:
    metadata:
      labels:
        component: postgres
    spec:
      volumes:
        - name: postgres-storage
          persistentVolumeClaim:
            claimName: database-persistent-volume-claim
      containers:
        - name: postgres
          image: postgres
          ports:
            - containerPort: 5432
          volumeMounts:
            - name: postgres-storage
              mountPath: /var/lib/postgresql/data
              subPath: postgres
          env:
            - name: PGPASSWORD
              valueFrom:
                secretKeyRef:
                  name: pgpassword
                  key: PGPASSWORD
```

Database-persistent-volume-claim.yaml

```
apiVersion: v1
```

```
kind: PersistentVolumeClaim
metadata:
  name: database-persistent-volume-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

Redis-cluster-ip-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: redis-cluster-ip-service
spec:
  type: ClusterIP
  selector:
    component: redis
  ports:
    - port: 6379
      targetPort: 6379
```

Redis-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      component: redis
  template:
    metadata:
      labels:
        component: redis
    spec:
      containers:
        - name: redis
          image: redis
          ports:
            - containerPort: 6379
```

// Use **Helm** to install ingress-nginx on kubernetes then after, ingress-service.yaml will be use as its configuration

// Need to **create a service account** for Helm to execute installation

> Kubectl create serviceaccount --namespace kube-system tiller
> Kubectl create clusterrolebinding tiller-cluster-rule --clusterrole-cluster-admin --serviceaccount-kube-system:tiller
> Helm init --service-account tiller --upgrade

Ingress-service.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-service
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /
    certmanager.k8s.io/cluster-issuer: 'letsencrypt-prod'    //← added for https
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
  tls:
    - hosts:
        - k8s-multi.com
        - www.k8s-multi.com
      secretName: k8s-multi-com
  rules:
    - host: k8s-multi.com
      http:
        paths:
          - path: /
            backend:
              serviceName: client-cluster-ip-service
              servicePort: 3000
          - path: /api/
            backend:
              serviceName: server-cluster-ip-service
              servicePort: 5000
    - host: www.k8s-multi.com
      http:
        paths:
          - path: /
            backend:
              serviceName: client-cluster-ip-service
              servicePort: 3000
          - path: /api/
            backend:
              serviceName: server-cluster-ip-service
              servicePort: 5000
```

Issuer.yaml //← added for https

```
apiVersion: certmanager.k8s.io/v1alpha1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: 'ste.grider@gmail.com'
    privateKeySecretRef:
      name: letsencrypt-prod
    http01: {}
```

Certificate.yaml //← added for https

```
apiVersion: certmanager.k8s.io/v1alpha1
kind: Certificate
metadata:
  name: k8s-multi-com-tls
spec:
  secretName: k8s-multi-com
  issuerRef:
    name: letsencrypt-prod
    kind: ClusterIssuer
  commonName: k8s-multi.com
  dnsNames:
    - k8s-multi.com
    - www.k8s-multi.com
  acme:
    config:
      - http01:
          ingressClass: nginx
        domains:
          - k8s-multi.com
          - www.k8s-multi.com
```

.travis.yml

```
sudo: required
services:
  - docker
env:
  global:
    - SHA=$(git rev-parse HEAD)
    - CLOUDSDK_CORE_DISABLE_PROMPTS=1
before_install:
  - openssl aes-256-cbc -K $encrypted_0c35eebf403c_key -iv $encrypted_0c35eebf403c_iv
-in service-account.json.enc -out service-account.json -d
//
```

1. Need to download service-account.json from google
2. Encypt the file using travis into service-account.json.enc

//

```
  - curl https://sdk.cloud.google.com | bash > /dev/null;
  - source $HOME/google-cloud-sdk/path.bash.inc
  - gcloud components update kubectl
  - gcloud auth activate-service-account --key-file service-account.json
  - gcloud config set project skilful-berm-214822
  - gcloud config set compute/zone us-central1-a
  - gcloud container clusters get-credentials multi-cluster
  - echo "$DOCKER_PASSWORD" | docker login -u "$DOCKER_USERNAME"
--password-stdin
  - docker build -t stephengrider/react-test -f ./client/Dockerfile.dev ./client

script:
  - docker run stephengrider/react-test npm test -- --coverage

deploy:
  provider: script
  script: bash ./deploy.sh
  on:
    branch: master
```

deploy.sh

```
docker build -t stephengrider/multi-client:latest -t stephengrider/multi-client:$SHA -f
./client/Dockerfile ./client
docker build -t stephengrider/multi-server:latest -t stephengrider/multi-server:$SHA -f
./server/Dockerfile ./server
docker build -t stephengrider/multi-worker:latest -t stephengrider/multi-worker:$SHA -f
./worker/Dockerfile ./worker

docker push stephengrider/multi-client:latest
docker push stephengrider/multi-server:latest
docker push stephengrider/multi-worker:latest

docker push stephengrider/multi-client:$SHA
docker push stephengrider/multi-server:$SHA
docker push stephengrider/multi-worker:$SHA

kubectl apply -f k8s
kubectl set image deployments/server-deployment server=stephengrider/multi-server:$SHA
kubectl set image deployments/client-deployment client=stephengrider/multi-client:$SHA
kubectl set image deployments/worker-deployment
worker=stephengrider/multi-worker:$SHA
```