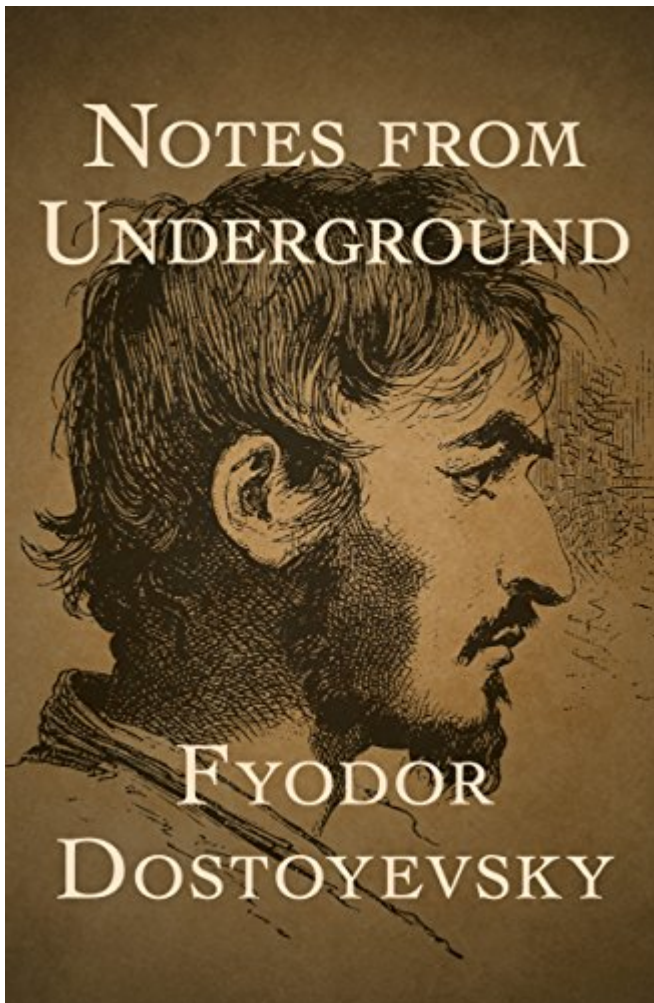


Notes from Underground



I. Introduction

IT is bizarre not to consider moving your websites to [Docker](#) environment in 2022. While reducing hardware and software cost is one gain, another gain is [scalability](#); the pain is to learn a new set of CLI Commands and write [Dockerfile](#), [docker-compose.yml](#) and *optionally* [Makefile](#). It's always easy to get started but difficult to become an expert. The only way is step by step learning and applying it in projects, there is no quick way to it.

II. Architecture

When one starts to create a website, it is not uncommon to confront with multiple environments, ie: development, staging and production... [Rule of three](#) states that two instances of similar code do not require refactoring, but when similar code is used three times, it should be extracted into a new procedure.

Using multiple [docker-compose.yml](#) files enables you to customize a application for different environments or different workflows^[2]:

By default, Compose reads two files, a [docker-compose.yml](#) and an optional [docker-compose.override.yml](#) file. By convention, the [docker-compose.yml](#) contains your base configuration.

The override file, as its name implies, can contain configuration overrides for existing services or entirely new services.

If a service is defined in both files, Compose merges the configurations using the rules described in Adding and overriding configuration.

To use multiple override files, or an override file with a different name, you can use the `-f` option to specify the list of files. Compose merges files in the order they're specified on the command line. See the docker-compose command reference for more information about using `-f`.

docker-compose.yml

```
version: "3"
services:
  nginx:
    image: nginx:stable-alpine
    volumes:
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
      - ./ssl:/etc/nginx/ssl
    depends_on:
      - node-app

  node-app:
    image: albert0i/node-app:1.1
    environment:
      - PORT=3000
    depends_on:
      - mongo
      - redis

  mongo:
    image: mongo:4.4
    environment:
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=root
    volumes:
      - ./data/db:/data/db

  redis:
    image: redis:6.2.6
```

docker-compose.dev.yml

```
version: "3"
services:
  nginx:
    ports:
      - "3000:80"
      - "3443:443"
```

```
node-app:
  build:
    context: .
    args:
      NODE_ENV: development
  volumes:
    - ./:/app
    - /app/node_modules
  environment:
    - NODE_ENV=development
    - MONGO_USER=root
    - MONGO_PASSWORD=root
    - SESSION_SECRET=secret
  command: npm run dev
```

docker-compose.prod.yml

```
version: "3"
services:
  nginx:
    ports:
      - "80:80"
      - "443:443"

  node-app:

    deploy:
      replicas: 2
      restart_policy:
        condition: any
      update_config:
        parallelism: 1
        delay: 15s

    build:
      context: .
      args:
        NODE_ENV: production

    environment:
      - NODE_ENV=production
      - MONGO_USER=${MONGO_USER}
      - MONGO_PASSWORD=${MONGO_PASSWORD}
      - SESSION_SECRET=${SESSION_SECRET}
    command: npm run start

  mongo:
    environment:
      - MONGO_INITDB_ROOT_USERNAME=${MONGO_INITDB_ROOT_USERNAME}
      - MONGO_INITDB_ROOT_PASSWORD=${MONGO_INITDB_ROOT_PASSWORD}
    deploy:
```

```

    replicas: 1
    placement:
      constraints: [node.role == manager]

visualizer:
  image: dockersamples/visualizer
  volumes:
    - "/var/run/docker.sock:/var/run/docker.sock"
  ports:
    - "8080:8080"
  deploy:
    replicas: 1
    placement:
      constraints: [node.role == manager]

```

Dockerfile

```

FROM node:16.8.0-alpine
WORKDIR /app
COPY package.json .
#RUN npm install
ARG NODE_ENV
RUN if [ "$NODE_ENV" = "development" ]; \
    then npm install; \
    else npm install --only=production; \
    fi
COPY . .
ENV PORT 3000
EXPOSE $PORT
CMD ["npm", "run", "dev"]

```

.env

```

#
# Attention: No space before the equal sign! Otherwise, will show:
# "docker: poorly formatted environment: variable 'XXX ' contains
# whitespaces."
#
PORT=3000

```

```

# Development
docker-compose -f docker-compose.yml -f docker-compose.dev.yml [options]
[COMMAND] [ARGS...]

# Production
docker-compose -f docker-compose.yml -f docker-compose.dev.yml [options]
[COMMAND] [ARGS...]

```

III. Development environment

```
# Create and start containers
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d

# Create and start containers.
# Build images before starting containers
# Recreate anonymous volumes instead of retrieving data from the previous
containers.
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d --
build -V

# Scale SERVICE to NUM instances. Overrides the "scale" setting in the
Compose file if present.
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d --
scale node-app=2

# Validate and view the Compose file
docker-compose -f docker-compose.yml -f docker-compose.dev.yml config

# Start service "node-app". Do not start linked services.
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d --no-
deps node-app

# Start service "mongo"
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d mongo

# View output from containers
docker-compose logs node-app

# Execute a command in a running container
docker-compose exec node-app printenv

# Execute a command in a running container
docker-compose exec mongo mongo -u root -p root
...
    db.books.insert({"name": "Harry Potter"});
...

# Stop and remove containers, networks, images, and volumes
# Remove named volumes declared in the "volumes" section of the Compose
file and
# anonymous volumes attached to containers.
docker-compose -f docker-compose.yml -f docker-compose.dev.yml down -v
```

IV. Production environment (Standalone mode)

```
# Create and start containers
docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d

# Create and start containers.
# Build images before starting containers
docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d --
build

# Create and start containers.
# Build images before starting containers
# Start service "node-app". Do not start linked services.
docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d --
build --no-deps node-app

# Create and start containers
# Show more output
docker-compose --verbose -f docker-compose.yml -f docker-compose.prod.yml
up -d

# Create and start containers
# Show more output. Set log level (DEBUG, INFO, WARNING, ERROR, CRITICAL)
docker-compose --verbose --log-level=DEBUG -f docker-compose.yml -f docker-
compose.prod.yml up -d

# Stop and remove containers, networks, images, and volumes
docker-compose -f docker-compose.yml -f docker-compose.prod.yml down -v
```

V. Production environment (Swarm mode)

Links online

[http](#)

[https](#)

[visualizer](#)

Node

```
cd myapp

# List nodes in the swarm
docker node ls
```

Stack

```
# Deploy a new stack or update an existing stack
docker stack deploy -c docker-compose.yml -c docker-compose.prod.yml myapp
```

```
# List stacks
docker stack ls

# List the tasks in the stack
docker stack ps myapp

# List the services in the stack
docker stack services myapp

# Remove one or more stacks
docker stack rm myapp
```

Service

```
# List services
docker service ls

# List the tasks of one or more services
docker service ps myapp_node-app

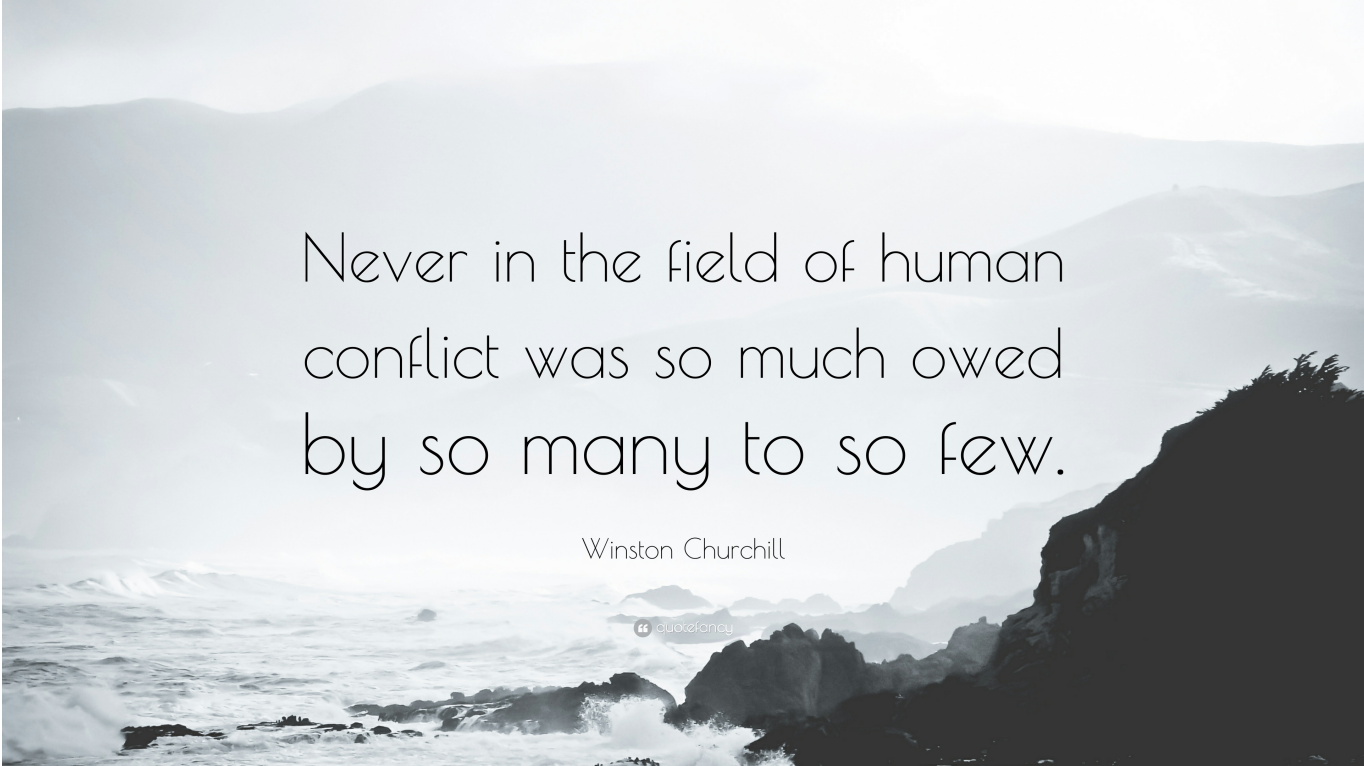
# Fetch the logs of a service or task
docker service logs myapp_node-app

# Scale one or multiple replicated services
docker service scale myapp_node-app=4
docker service scale myapp_nginx=2
```

```
albert0i@penguin: ~ x ubuntu@instance-20220502-1505: ~/ x albert0i@penguin: ~ x + _ □ x

ubuntu@instance-20220502-1505:~/app$ docker service ls
ID                NAME                MODE                REPLICAS            IMAGE                PORTS
ppml2nfhs350     myapp_mongo         replicated           1/1                 mongo:4.4
paoat4g9cqt2     myapp_nginx         replicated           0/1                 nginx:stable-alpine *:80->80/tcp
v98u8z3vvsjy     myapp_node-app      replicated           2/2                 albert0i/node-app:1.0
h6ujmntgk75      myapp_redis         replicated           0/1                 redis:6.2.6
y4cx4cbawh32     myapp_visualizer    replicated           1/1                 dockersamples/visualizer:latest *:8080->8080/tcp
ubuntu@instance-20220502-1505:~/app$ docker service scale myapp_node-app=4
myapp_node-app scaled to 4
overall progress: 4 out of 4 tasks
1/4: running [=====>]
2/4: running [=====>]
3/4: running [=====>]
4/4: running [=====>]
verify: Service converged
ubuntu@instance-20220502-1505:~/app$ docker service scale myapp_nginx=2
myapp_nginx scaled to 2
overall progress: 2 out of 2 tasks
1/2: running [=====>]
2/2: running [=====>]
verify: Service converged
ubuntu@instance-20220502-1505:~/app$
```

VI. Conclusion



Never in the field of human
conflict was so much owed
by so many to so few.

Winston Churchill

“ duoxifancy

```
Imagine there's no VM  
It's easy if you try  
No hell below us  
Above us only sky  
Imagine all web sites  
Moving for today...
```

```
Imagine there's no VM  
It isn't hard to do  
Nothing to lose or die for  
And no hassle too  
Imagine all web sites  
Running em in peace...
```

```
You may say I'm a dreamer  
But I'm not the only one  
I hope someday you'll join us  
And the world will be as one
```

VII. Reference

1. [Learn Docker - DevOps with Node.js & Express](#)
2. [Share Compose configurations between files and projects](#)
3. [How to generate and use a SSL certificate in NodeJS](#)
4. [Quick Tip: Configuring NGINX and SSL with Node.js](#)
5. [How to Use SSL/TLS with Node.js](#)
6. [\[v1.25.0\] "Only pull images that can't be built" should be optional #7103](#)
7. [Markdown Cheat Sheet](#)

VIII. Appendix

Generate SSL/TLS certificates (valid for 365 days):

```
openssl genrsa -out key.pem

openssl req -new -key key.pem -out csr.pem

openssl x509 -req -days 365 -in csr.pem -signkey key.pem -out cert.pem
```

Configuring NGINX by adding three more lines to server block in default.conf:

```
listen 443 ssl;

ssl_certificate /etc/nginx/ssl/cert.pem;
ssl_certificate_key /etc/nginx/ssl/key.pem;
```

default.conf

```
server {
    listen 80;
    listen 443 ssl;

    ssl_certificate /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;

    location /api {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;
        proxy_pass http://node-app:3000;
        proxy_redirect off;
    }
}
```

EOF (2022/06/25)