xFans V2 setup

I. Server requirement

xFans supports all platforms: Windows, MacOS, Linux.

xFans needs a VPS server with at least:

- 2GB of RAM - recommend 4GB
- 40GB of HDD
- 1 CPU core - recommend 2 cores at least
- 3 domains / sub domains
  - api.[your-domain] and point to server IP address
  - admin.[your-domain] and point to server IP address
  - [your-domain] and point to server IP address

II. Software requirements

xFans architecture needs these softwares

- NodeJS v12.x
  - To install please download NodeJS here
- MongoDB >= v3.6
  - To install please download mongoDB
- Redis server >= v2.8
  - To install please download and setup redis
- FFMPEG
- Nginx >= v1.3
- PM2 is a daemon process manager that will help you manage and keep your application online 24/7
- Yarn or npm to manage nodeJS package

Ensure all softwares above are running before setup source code

Helpful links

- Install nodejs with nvm
- Install nodejs on Ubuntu
- Install Yarn
- Install mongoDB on Ubuntu
- Install FFMPEG on Ubuntu
- Install nginx on Ubuntu

Testing

From command line / terminal please run these commands to check

1. NodeJS

```
$ node -v
v12.16.2
```

2. MongoDB

```
$ mongo --version
MongoDB shell version v4.0.7
git version: 1b82c812a9c0bbf6dc79d5400de9ea99e6ffa025
allocator: system
modules: none
build environment:
    distarch: x86_64
    target_arch: x86_64
```

3. Redis server

```
$ redis-server --version
Redis server v=5.0.4 sha=00000000:0 malloc=libc bits=64
build=d4ba11298acbb366
```

4. FFMPEG

```
$ ffmpeg -version
fmpeg version 2.8.15-0ubuntu0.16.04.1 Copyright (c) 2000-2018 the
FFmpeg developers
built with gcc 5.4.0 (Ubuntu 5.4.0-6ubuntu1~16.04.10) 20160609
configuration: --prefix=/usr --extra-version=0ubuntu0.16.04.1 --build-
suffix=-ffmpeg --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu
--incdir=/usr/include/x86_64-linux-gnu --cc=cc --cxx=g++ --enable-gpl --
enable-shared --disable-stripping --disable-decoder=libopenjpeg --
disable-decoder=libschroedinger --enable-avresample --enable-avisynth --
enable-gnutls --enable-ladspa --enable-libass --enable-libbluray --
enable-libbs2b --enable-libcaca --enable-libcdio --enable-libflite --
enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-
libgme --enable-libgsm --enable-libmodplug --enable-libmp3lame --enable-
libopenjpeg --enable-libopus --enable-libpulse --enable-librtmp --
enable-libschroedinger --enable-libshine --enable-libsnappy --enable-
libsoxr --enable-libspeex --enable-libssh --enable-libtheora --enable-
libtwolame --enable-libvorbis --enable-libvpx --enable-libwavpack --
enable-libwebp --enable-libx265 --enable-libxvid --enable-libzvbi --
enable-openal --enable-opengl --enable-x11grab --enable-libdc1394 --
enable-libiec61883 --enable-libzmq --enable-frei0r --enable-libx264 --
enable-libopencv
libavutil      54. 31.100 / 54. 31.100
libavcodec     56. 60.100 / 56. 60.100
libavformat    56. 40.101 / 56. 40.101
libavdevice    56.  4.100 / 56.  4.100
libavfilter     5. 40.101 /  5. 40.101
libavresample   2.  1.  0 /  2.  1.  0
libswscale      3.  1.101 /  3.  1.101
libswresample   1.  2.101 /  1.  2.101
libpostproc    53.  3.100 / 53.  3.100
```

5. NGINX

```
$ nginx -v
nginx version: nginx/1.10.3
```

6. Pm2

```
$ pm2 -v
[PM2] Spawning PM2 daemon with pm2_home=/Users/xxx/.pm2
[PM2] PM2 Successfully daemonized
4.4.0
```

7. Yarn

```
$ yarn -v
1.16.0
```

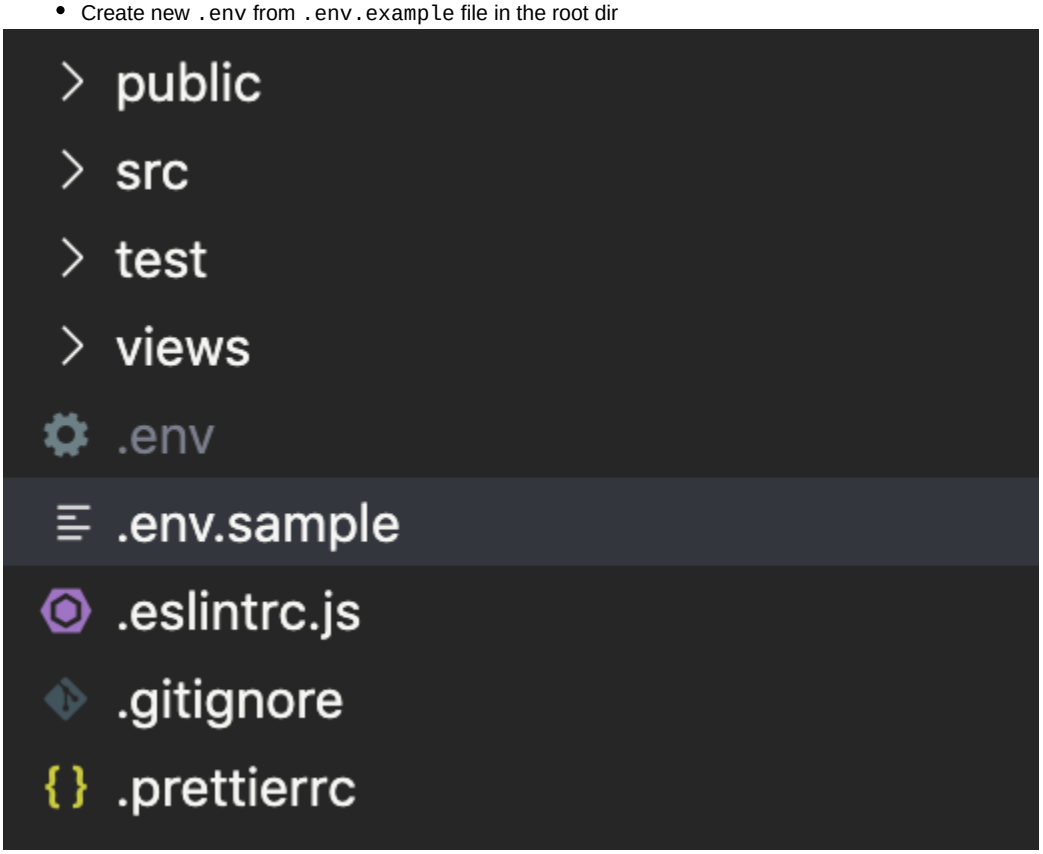
III. Setup and test project locally

1. API

- Ensure `node v12.x`, `MongoDB`, `Redis server`, `FFMPEG` and `yarn` are running
- CD to your API folder
- Run `yarn` to install nodeJS dependencies

```
$ yarn
yarn install v1.16.0
[1/4]   Resolving packages...
[2/4]   Fetching packages...
[3/4]   Linking dependencies...
```

- Create new `.env` from `.env.example` file in the root dir



- Open file `.env` and change config

```
# Change with your custom HTTP port
HTTP_PORT=3000
# Change secret with your key
TOKEN_SECRET=1213456
# Change mongo uri correctly. Read more about connection string here
https://docs.mongodb.com/manual/reference/connection-string/
MONGO_URI=mongodb://localhost/test

# Use for seed and some features, change with your domain without http
(s)://
DOMAIN=example.com
```

- Run `yarn start:dev` to start development env

```
$ yarn start:dev
:48 PM] Starting compilation in watch mode...

[2:49:13 PM] Found 0 errors. Watching for file changes.

[Nest] 60068   - 08/03/2020, 2:49:17 PM   [NestFactory] Starting Nest
application...
[Nest] 60068   - 08/03/2020, 2:49:17 PM   [InstanceLoader] QueueModule
dependencies initialized +134ms
[Nest] 60068   - 08/03/2020, 2:49:17 PM   [InstanceLoader] RedisModule
dependencies initialized +0ms
[Nest] 60068   - 08/03/2020, 2:49:17 PM   [InstanceLoader] HttpModule
dependencies initialized +1ms
[Nest] 60068   - 08/03/2020, 2:49:17 PM   [InstanceLoader] QueueModule
dependencies initialized +0ms
[Nest] 60068   - 08/03/2020, 2:49:17 PM   [InstanceLoader]
ServeStaticModule dependencies initialized +4ms
....
```

- Open browsers and run `http://localhost:3000` it should show success message or `Hello World!`

2. Frontend web

- Ensure `nodejs` and `yarn` are running
- From frontend root directory run `yarn` command to install nodejs dependencies

```
$ yarn
yarn install v1.16.0
[1/4]   Resolving packages...
[2/4]   Fetching packages...
[3/4]   Linking dependencies...
```

- Open `config > client.ts` file and edit information. Change with your url in `development` or `production` API url

```
export default {
  # In production, please change with your production url like
https://api.yourdomain.com/v1
  apiEndpoint: 'http://localhost:9000/v1',
  # In production, please change with your production url like
https://api.yourdomain.com
  socketUrl: 'http://localhost:9000',
  maxProfilePhotoNum: 5,
  completedProfilePhotoNum: 5
};
```

- Run `yarn run dev` to start development env

```
$ yarn run dev
yarn run v1.16.0
$ ts-node --project tsconfig.server.json server/index.ts
Warning: Built-in CSS support is being disabled due to custom CSS
configuration being detected.
See here for more info: https://err.sh/next.js/built-in-css-disabled

> Using external babel configuration
```

- Then you can open browser `http://localhost:3000` to see the web

3. Admin web

- Similar Frontend web, same steps.
- You should change default port with `yarn run dev -p 9001` then open `http://localhost:9001` to see your admin app

IV. Setup and setup production environment with nginx

1. API

- Make sure you have installed nodeJS dependencies by `yarn` command
- In API root directory, run `yarn build`

```
$ yarn build
yarn run v1.16.0
$ rimraf dist
$ nest build && yarn copy-template
$ cp -r ./src/templates ./dist/templates
  Done in 22.62s
```

- Testing by run command `yarn start:prod` or `node dist/main.js`

```
$ yarn start
yarn run v1.16.0
$ node dist/main
[Nest] 60205   - 08/03/2020, 2:56:11 PM   [NestFactory] Starting Nest
application...
....
```

- Run `pm2 start dist/main.js --name=api` to run API under background

2. Frontend web

- Make sure you have installed nodeJS dependencies by `yarn` command
- Run `yarn build`

```
$ yarn build
yarn run v1.16.0
$ next build && tsc && tsc --project tsconfig.server.json
Warning: Built-in CSS support is being disabled due to custom CSS
configuration being detected.
See here for more info: https://err.sh/next.js/built-in-css-disabled

> Using external babel configuration
> Location: "/Users/tuong/Projects/frontend-base-reactjs/.babelrc"
Creating an optimized production build

Compiled with warnings.

chunk styles [mini-css-extract-plugin]
Conflicting order between:
 * css ./node_modules/css-loader??ref--6-1!./node_modules/less-loader
/dist/cjs.js??ref--6-2!./src/components/video/video.less
 * css ./node_modules/css-loader??ref--6-1!./node_modules/less-loader
/dist/cjs.js??ref--6-2!./src/components/common/layout/page.less
 * css ./node_modules/css-loader??ref--6-1!./node_modules/less-loader
/dist/cjs.js??ref--6-2!./src/components/common/base/loader.less
 ....
```

- Testing by run `yarn start` or `node dist/index.js` it should show success message

```
$ yarn start
yarn run v1.16.0
next start
ready - started server on http://localhost:3000
```

- Run `pm2 start yarn --interpreter bash --name xfans-web -- start -p 8081` to run application under port 8081

3. Admin web

- Similar Frontend web, same steps.
- You should change default port with `pm2 start yarn --interpreter bash --name xfans-web -- start -p 8082` to run app under port 8082

4. Setup nginx

a. Api

- Let say our frontend code is in `/var/www/api.xfans.info` folder and api is running under port 8080, add new file in `/etc/nginx/sites-enabled/api.xfans.info` and content as bellow
- You have to change with our server name and source code path

```
server {
    listen 80 ;
        listen [::]:80 ;
        root /var/www/api.xfans.info;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
    server_name api.xfans.info;
    gzip on;
    gzip_proxied any;
    gzip_comp_level 4;
    gzip_types text/css application/javascript image/svg+xml;

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    sendfile_max_chunk 512;
    client_max_body_size 200M;

        location / {
      proxy_set_header   X-Forwarded-For $remote_addr;
      proxy_set_header   Host $http_host;
      proxy_pass         http://localhost:8080;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection 'upgrade';
      proxy_cache_bypass $http_upgrade;
        }

        location /videos/protected/ {
          auth_request /authvideo;
          root /var/www/api.xfans.info/public/;
        }

        location = /authvideo {
          internal;
          set $query '';
          if ($request_uri ~* "[^\?]+\?(.*)$") {
              set $query $1;
          }
          proxy_pass http://localhost:8080/user/performer-assets/videos
/auth/check?$query;
          proxy_pass_request_body   off;
          proxy_set_header Content-Length "";
        }
        location /photos/protected/ {
          auth_request /authphoto;
          root /var/www/api.xfans.info/public/;
        }
```

```
        location = /authphoto {
          internal;
          set $query '';
          if ($request_uri ~* "[^\?]+\?(.*)$") {
              set $query $1;
          }
          proxy_pass http://localhost:8080/performer/performer-assets
/photos/auth/check?$query;
          proxy_pass_request_body   off;
          proxy_set_header Content-Length "";
        }
}
```

b. Frontend

- Let say our frontend code is in `/var/www/xfans.xscripts.info` folder and Frontend app is running under port 8081, add new file in `/etc/nginx/sites-enabled/xfans.info` and content as bellow
- You have to change with our server name and source code path

```
server {
    listen 80;
        server_name xfans.info;

        root /var/www/xfans.info;

    gzip on;
    gzip_proxied any;
    gzip_comp_level 4;
    gzip_types text/css application/javascript image/svg+xml;

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    sendfile_max_chunk 512;
    client_max_body_size 200M;

        location / {
        proxy_pass http://localhost:8080/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_redirect        off;
        proxy_set_header      Host $host;
```

```
        proxy_set_header      X-Real-IP $remote_addr;
        proxy_set_header      X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header      Proxy "";

        # WebSocket support
        proxy_connect_timeout 7d;
        proxy_send_timeout 7d;
        proxy_read_timeout 7d;
    }

    location /_next/static/ {
        alias /var/www/xfans.info/.next/static/$1;
        access_log off;
        expires max;
    }

    location /static/ {
        alias /var/www/xfans.info/static/$1;
        expires max;
        autoindex off;
    }
}
```

c. Admin

- Let say our frontend code is in `/var/www/admin.xfans.info` folder and Frontend app is running under port 8082, add new file in `/etc/nginx/sites-enabled/admin.xfans.info` and content as bellow
- You have to change with our server name and source code path

```
server {
    listen 80;
        server_name admin.xfans.info;
        root /var/www/admin.xfans.info;

    gzip on;
    gzip_proxied any;
    gzip_comp_level 4;
    gzip_types text/css application/javascript image/svg+xml;

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    sendfile_max_chunk 512;
    client_max_body_size 200M;

        location / {
        proxy_set_header   X-Forwarded-For $remote_addr;
        proxy_set_header   Host $http_host;
        proxy_pass         http://localhost:8082;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_cache_bypass $http_upgrade;
    }

    location /_next/static/ {
        alias /var/www/admin.xfans.info/.next/static/$1;
        access_log off;
        expires max;
    }

    location /static/ {
        alias /var/www/admin.xfans.info/static/$1;
        expires max;
        autoindex off;
    }
}
```

d. Testing

- Open browser and access: http://yourdomain.com to access web frontend
- Open browser and access: http://admin.yourdomain.com to access web admin

5. Check applications are running under pm2

- Run `pm2 ls` to see your applications

- Run `pm2 stop [id]` to stop app or `pm2 reload [id]` to reload
- Run applications in start up by `pm2 startup` then `pm2 save`

6. Set up nginx with https

- Check here to setup https with nginx
- Setup https with Certbot nginx

7. Migration

- From API root directory run `yarn run script:dev` to seed default values for settings and users
    - Default admin: admin@[yourdomain] with domain is in .env file
    - Default password: adminadmin
- You can change default values in src > scripts folder