Deploy Jitsi with Docker + High Quality

Every `eolab.de` has to be changed to your URL.

Prerequisits

- Ubuntu Server 20.04
- Docker and Docker-Compose are installed
- Running NGINX Reverse Proxy
- A SSL certificate for the desired url

Downloading Jitsi

1. `git clone https://github.com/jitsi/docker-jitsi-meet.git`
2. `mv docker-jitsi-meet/ jitsi-meet/`
3. `cd jitsi-meet/`
4. `cp env.example .env`
5. `./gen-passwords.sh`
6. `mkdir -p ~/.jitsi-meet-cfg/{web/letsencrypt,transcripts,prosody,jicofo,jvb,jigasi,jibri}`

Configuring Jitsi

1. `nano .env`
2. make sure it uses an unused port

   `HTTP_PORT=8000`
3. set timezone

   `TZ=Europe/Berlin`
4. set the later used public URL

   `PUBLIC_URL=https://meet.eolab.de`

Configuring NGINX

1. `sudo nano /etc/nginx/sites-available/eolab.de`
2. make sure there is a redirect from HTTP to https (port 80 to 443)
3. add this to the top

```
upstream jitsi {
  server localhost:8000;
}
```

   change the port if needed

4. add this to the bottom

```
server {
  listen 443 ssl http2;
  listen [::]:443 ssl http2;
  server_name meet.eolab.de;

  ssl_certificate /etc/letsencrypt/live/meet.eolab.de/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/meet.eolab.de/privkey.pem;
  include snippets/ssl-params.conf;

  add_header Strict-Transport-Security "max-age=31536000;
includeSubdomains; preload";
  add_header X-Xss-Protection "1; mode=block";
  add_header X-Content-Type-Options nosniff;
  add_header Referrer-Policy same-origin;
  proxy_cookie_path / "/; HTTPOnly; Secure";
  add_header Expect-CT "enforce, max-age=21600";
  add_header Feature-Policy "payment none";

  keepalive_timeout    70;
  sendfile             on;
  client_max_body_size 0;

  gzip on;
  gzip_disable "msie6";
  gzip_vary on;
  gzip_proxied any;
  gzip_comp_level 6;
  gzip_buffers 16 8k;
  gzip_http_version 1.1;
  gzip_types text/plain text/css application/json
application/javascript text/xml application/xml application/xml+rss
text/javascri$

   location / {
        log_not_found off;
```

```
        proxy_cache_valid 200 120m;
        proxy_set_header        Host    $http_host;
        proxy_set_header        X-Real-IP $remote_addr;
        proxy_set_header        X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header        X-Scheme $scheme;
        proxy_pass http://jitsi/;
        }

   location ~ ^/colibri-ws/([a-zA-Z0-9-\.]+)/(.*) {
        tcp_nodelay on;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_pass http://jitsi/colibri-ws/$1/$2$is_args$args;
        }

   location /xmpp-websocket {
        tcp_nodelay on;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_pass http://jitsi/xmpp-websocket;
        }
}
```

make sure to change the `server_name`, `ssl_certificate` and `ssl_certificate_key`

5. Save and quit the file
6. `sudo systemctl restart nginx`

Start Jitsi

1. `docker-compose up -d`

Configure Jitsi with internal authentication for creating rooms

1. make sure Jitsi was run at least once
2. stop it if its running

```
docker-compose stop
```

3. `rm -r ~/.jitsi-meet-cfg`

4. `mkdir -p ~/.jitsi-meet-cfg/{web/letsencrypt,transcripts,prosody,jicofo,jvb,jigasi,jibri}`

5. `nano .env`

6. `ENABLE_AUTH=1`

7. `ENABLE_GUESTS=1`

8. `AUTH_TYPE=internal`

9. Save and quit the file

10. `docker-compose up -d`

11. find the container id of the jitsi_prosody container and copy that

    `docker ps -a`

12. `docker exec -it CONTAINER-ID-HERE /bin/bash`

13. `su`

14. change the `username` and `password` in this command:

    ```
    prosodyctl --config /config/prosody.cfg.lua register username meet.jitsi password
    ```

    These credentials are now needed to create a new room

15. `exit`

16. `exit`

Configure Jitsi for high quality webcams and screenshares

1. make sure Jitsi was run at least once
2. stop it if its running

   `docker-compose stop`

3. `sudo nano ~/.jitsi-meet-cfg/web/config.js`

4.
```
resolution: 1080,
```

5.
```
constraints: {
       video: {
           height: {
               ideal: 1080,
               max: 1440,
               min: 480
           }
       }
     },
```

6.
```
desktopSharingFrameRate: {
       min: 5,
       max: 24
     },
```