Deploying OpenVidu on premises

Deployment instructions (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#deployment-instructions)

OpenVidu is deployed in production as a set of **Docker** containers managed with **Docker Compose**.
You can deploy OpenVidu in **any modern Linux distribution**.

This procedure installs the following services:

- **OpenVidu Server (openvidu-server)**: this is the brain of OpenVidu platform. In charge of the signaling plane.
- **Kurento Media Server (kms)**: this is the hearth of the OpenVidu platform. In charge of media plane.
- **Coturn (coturn)**: server used to allow media communications with browsers in certain special networks.
- **Redis (redis)**: database to manage users in Coturn server.
- **Nginx (nginx)**: a reverse proxy used to configure SSL certificate and to allow both Openvidu Server and the Application to be served in the standard https port (443).
- **Videoconference Application (app)**: OpenVidu Call application or any other application. Can be disabled.

> **NOTE 1** : Docker basic knowledge is not required, but recommended. If you are completely new to Docker and containers, take a few minutes to read the official Docker *Get started* (https://docs.docker.com/get-started/) documentation.
>
> **NOTE 2** : instructions below are intended for production environments. For development environments it is recommended to use this simple Docker image that wraps everything necessary to work inside your network in just a single container: openvidu/openvidu-server-kms (https://hub.docker.com/r/openvidu/openvidu-server-kms)

Explore docs
2.13.0 ▾
What is OpenVidu? (/)
Tutorials ▾
Demos ▾
Cheatsheet ▾
Advanced features ▾
Reference Documentation ▾
Deployment ▴
AWS (deployment/deploying-aws/)
On premises (deployment/deploying-on-premises/)
Upgrading OpenVidu (deployment/upgrading/)
Contribute ▾
Troubleshooting / FAQ (troubleshooting/)
Releases (releases/)
OpenVidu Pro PRO ▾

1) Prerequisites (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#1-prerequisites)

- **Install Docker (https://docs.docker.com/engine/install/#server)**
- **Install Docker Compose (https://docs.docker.com/compose/install/)**. NOTE: install docker-compose from the link (official Docker site) as minimum version `1.24` is required.
- **Configure a domain name**: OpenVidu is deployed using HTTPS because it is mandatory to use WebRTC. Then, if you do not have a domain name, an autogenerated SSL certificate will be used and an ugly warning will appear to your users when enter to your site. And of course you can suffer a man-in-the-middle attack. So it is recommended that you configure a domain name pointing to your machine's public IP. A valid SSL certificate can be automatically generated using Let's Encrypt in the installation process. If you already have a valid SSL certificate of your own, it also can be configured.
- **Port configuration in the server**
  - **Open these ports** (in section Close ports to avoid external attacks (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#close-ports-to-avoid-external-attacks) you have an UFW sample to configure a firewall)
    - **22 TCP**: to connect using SSH to admin OpenVidu.
    - **80 TCP**: if you select Let's Encrypt to generate an SSL certificate this port is used by the generation process.
    - **443 TCP**: OpenVidu server and application are published by default in standard https port.
    - **3478 TCP+UDP**: used by TURN server to resolve clients IPs.
    - **40000 - 57000 TCP+UDP**: used by Kurento Media Server to establish media connections.
    - **57001 - 65535 TCP+UDP**: used by TURN server to establish relayed media connections.
  - **Close all other ports**: this is VERY important to avoid external attacks to OpenVidu internal services. Check troubleshooting section Close ports to avoid external attacks (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#close-ports-to-avoid-external-attacks) to learn more about this.
  - **Free ports inside the server**: OpenVidu platform services will need the following ports to be available in the machine: 80, 443, 3478, 5442, 5443, 6379 and 8888. If some of these ports is used by any process, OpenVidu platform won't work correctly. It is a typical error to have an NGINX process in the system before installing OpenVidu. Please uninstall it.

2) Deployment (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#2-deployment)

You need root permissions to deploy OpenVidu.

```
sudo su
```

The recommended folder to install OpenVidu is **`/opt`**. Every other instruction in the documentation regarding on premises deployment assumes this installation path.

```
cd /opt
```

Now execute the following command to download and run the installation script.

```
curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash
```

This will download all required files into `openvidu` folder and will show this message with basic instructions:

```
Openvidu Platform successfully installed.

1. Go to openvidu folder:
$ cd openvidu

2. Configure OPENVIDU_DOMAIN_OR_PUBLIC_IP and OPENVIDU_SECRET in .env file:
$ nano .env

3. Start OpenVidu
$ ./openvidu start

For more information, check readme.md
```

> To deploy a fixed version, including previous ones, replace `latest` with the desired version number.
> For example: `curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_`**2.13.0**`.sh | bash`

3) Configuration (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#3-configuration)

OpenVidu Platform configuration is specified in the **`.env`** file with environment variables.

- You *must* give a value to properties **`OPENVIDU_DOMAIN_OR_PUBLIC_IP`** and **`OPENVIDU_SECRET`**. Default empty values will fail.
- You can change the **`CERTIFICATE_TYPE`** if you have a valid domain name. Setting this property to `letsencrypt` will automatically generate a valid certificate for you (it is required to set property `LETSENCRYPT_EMAIL`). Or if for any unknown reason you prefer to use your own certificate, set the property to `owncert` and place the certificate files as explained.
- All other configuration properties come with sane defaults. You can go through them and change whatever you want. Visit OpenVidu Server configuration (reference-docs/openvidu-config/) for further information.

The **`.env`** file is pretty self-explanatory. It looks like this:

```
# OpenVidu configuration
# ----------------------
# Documentation: https://docs.openvidu.io/en/stable/reference-docs/openvidu-config/

# NOTE: This file doesn't need to quote assignment values, like most shells do.
# All values are stored as-is, even if they contain spaces, so don't quote them.

# Domain name. If you do not have one, the public IP of the machine.
# For example: 198.51.100.1, or openvidu.example.com
OPENVIDU_DOMAIN_OR_PUBLIC_IP=

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=

# Certificate type:
# - selfsigned:  Self signed certificate. Not recommended for production use.
#                Users will see an ERROR when connected to web page.
# - owncert:     Valid certificate purchased in a Internet services company.
#                Please put the certificates files inside folder ./owncert
#                with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#                required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#                variable.
CERTIFICATE_TYPE=selfsigned

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL=user@example.com

...
```

Videoconference application (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#videoconference-application)

By default, the OpenVidu Call (demos/openvidu-call/) application is deployed alongside OpenVidu Platform. It is accessible in the URL:

```
https://OPENVIDU_DOMAIN_OR_PUBLIC_IP/
```

This application is defined in file `docker-compose.override.yml`. To disable OpenVidu Call application, you can delete the file `docker-compose.override.yml` (or just rename it in case you want to enable it again in the future).

You can configure any other application updating the content of `docker-compose.override.yml` to use any other Docker container, with the following requirements:

- Application server port must be binded to 5442 in the host, as this port is used by NGINX to publish your app in the default HTTPS port (443).
- The app must be served in plain HTTP as NGINX is the responsible of managing SSL certificate, so disable HTTPS and SSL in your app.
- Application has to know OpenVidu Server URL. You can use the environment variables ${OPENVIDU_DOMAIN_OR_PUBLIC_IP} and ${OPENVIDU_SECRET} in `docker-compose.override.yml` file.
- The application and OpenVidu platform are deployed in the same domain. For that reason, the following URLs are reserved for OpenVidu and you cannot use them in the application:
    - `/api/`
    - `/openvidu/`
    - `/dashboard/`
    - `/recordings/`

---

4) Execution (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#4-execution)

To start OpenVidu Platform (and the application if enabled) you can execute this command:

```
./openvidu start
```

All docker images for services will be downloaded (only the first time) and executed.

The first part of the log shows how docker-compose command executes all services:

```
Creating openvidu-docker-compose_coturn_1          ... done
Creating openvidu-docker-compose_app_1             ... done
Creating openvidu-docker-compose_kms_1             ... done
Creating openvidu-docker-compose_nginx_1           ... done
Creating openvidu-docker-compose_redis_1           ... done
Creating openvidu-docker-compose_openvidu-server_1 ... done
```

Then, `openvidu-server` service logs are shown. When OpenVidu Platform is ready you will see this message:

```
----------------------------------------------------

   OpenVidu Platform is ready!
   ---------------------------

   * OpenVidu Server: https://server/

   * OpenVidu Dashboard: https://server/dashboard/

----------------------------------------------------
```

You can press `Ctrl+C` to come back to the shell and OpenVidu will be executed in the background.

Available services (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#available-services)

- Consume OpenVidu REST API (reference-docs/REST-API/) through `https://server/`
- If the application (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#videoconference-application) is enabled, it will also be available at `https://server/`
- You can open OpenVidu Dashboard to verify everything is working as expected at `https://server/dashboard/` with credentials:
  - user: OPENVIDUAPP
  - pass: the value of OPENVIDU_SECRET in `.env` file

---

5) Administration (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#5-administration)

Run the following commands to manage OpenVidu Platform service:

- Start OpenVidu

  ```
  ./openvidu start
  ```

- Stop OpenVidu

  ```
  ./openvidu stop
  ```

- Restart OpenVidu

  ```
  ./openvidu restart
  ```

- Show logs of OpenVidu

  ```
  ./openvidu logs
  ```

> To change current configuration, you just need to update `.env` configuration file with the new desired values and run `./openvidu restart` command.

Troubleshooting (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#troubleshooting)

Configuration errors (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#configuration-errors)

If there's any error with the configuration, a report detailing which configuration property has issues and a step-by-step guide to fix it will be immediately shown by OpenVidu. The report will be similar to this:

```
Configuration errors
--------------------

* Property OPENVIDU_SECRET is not set. Cannot be empty.
* Property OPENVIDU_DOMAIN_OR_PUBLIC_IP is not set. Cannot be empty


Fix config errors
---------------

1) Return to shell pressing Ctrl+C
2) Set correct values in '.env' configuration file
3) Restart OpenVidu with:

    $ ./openvidu restart
```

Docker compose (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#docker-compose)

To solve any other issue, it is important to understand how is OpenVidu executed.

OpenVidu is executed as a docker-compose file. The commands executed by the script are the standard docker-compose commands, so internally they just do:

- start
  - `$ docker-compose up -d`
  - `$ docker-compose logs -f openvidu-server`
- stop
  - `$ docker-compose down`
- restart
  - `$ docker-compose down`
  - `$ docker-compose up -d`
  - `$ docker-compose logs -f openvidu-server`
- logs
  - `$ docker-compose logs -f openvidu-server`

As you can see, logs of `openvidu-server` service are shown when platform is started or restarted. This log contains the most important information for the OpenVidu execution.

Show service logs (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#show-service-logs)

Take a look to service logs to see what happened. First, see openvidu-server logs:

```
./openvidu logs
```

You can also see all service logs together:

```
docker-compose logs -f
```

Or you can inspect one by one the other services:

```
docker-compose logs -f kms
docker-compose logs -f nginx
docker-compose logs -f coturn
docker-compose logs -f redis
docker-compose logs -f app
```

Review the configuration (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#review-the-configuration)

Sometimes we may have a typo when writing a property name. For this reason, openvidu-server prints in the log all the configuration properties you have set in `.env` file and the default values for all other configuration properties. In that way, you can double check what openvidu-server actually *sees*.

```
Configuration properties
---------------------
* CERTIFICATE_TYPE=selfsigned
* OPENVIDU_CDR=false
* OPENVIDU_CDR_PATH=/opt/openvidu/cdr
* OPENVIDU_DOMAIN_OR_PUBLIC_IP=my.domain.com
* OPENVIDU_RECORDING=false
* OPENVIDU_RECORDING_AUTOSTOP-TIMEOUT=120
* OPENVIDU_RECORDING_COMPOSED-URL=

...
```

Change log level of the services (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#change-log-level-of-the-services)

To change the level of OpenVidu logs (***openvidu-server*** docker service) change the property `OV_CE_DEBUG_LEVEL` in configuration file `.env`.

To change the level of Kurento Media Server logs (***kms*** docker service) change the property `KMS_DEBUG_LEVEL` in configuration file `.env`. For more information about possible values visit Kurento Debug Logging (https://doc-kurento.readthedocs.io/en/stable/features/logging.html).

Change Kurento Media Server docker image (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#change-kurento-media-server-docker-image)

OpenVidu and Kurento Media Server evolve at a different pace. Sometimes, it is possible that a new KMS is released but OpenVidu is not still updated. In that case, if you hit a bug that might be solved in the last KMS version, you can test if just updating KMS fixes your issue. `KMS_IMAGE` property allows you to specify the new KMS image in configuration file `.env` .

Close ports to avoid external attacks (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#close-ports-to-avoid-external-attacks)

Closing all non-necessary ports in your server is very important to avoid external attacks. Some administrators using OpenVidu have reported attacks because their ports weren't properly closed. Of course, all of the opened ports stated in Prerequisites (https://docs.openvidu.io/en/2.13.0/deployment/deploying-on-premises/#1-prerequisites) section must be accessible from the exterior, but the rest must be closed to grant proper protection.

Typically, cloud providers initiate their machines with all ports closed by default, so usually it is only necessary to open the required ones. For example, you can configure public/private ports in in OpenStack with OpenStack security groups (https://docs.openstack.org/horizon/latest/user/configure-access-and-security-for-instances.html#), in AWS with AWS security groups (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html), etc...

If your only choice is to manually configure a firewall, you can for example install in any GNU/LInux system the great UFW (Uncomplicated Firewall) (https://manpages.ubuntu.com/manpages/bionic/en/man8/ufw.8.html) (`sudo apt install ufw`) with the following configuration to only allow the required ports:

```
ufw allow ssh
ufw allow 80/tcp
ufw allow 443/tcp
ufw allow 3478/tcp
ufw allow 3478/udp
ufw allow 40000:57000/tcp
ufw allow 40000:57000/udp
ufw allow 57001:65535/tcp
ufw allow 57001:65535/udp
ufw enable
```

« PREV (DEPLOYMENT/DEPLOYING-AWS/) NEXT » (DEPLOYMENT/UPGRADING/)

Edit Documentation on GitHub (https://github.com/OpenVidu/openvidu.io-docs//blob/master/docs/)