# Executive Summary

Throughout the past few weeks I tested and experimented with two different federated learning platforms, namely NVFlare and OpenFL on the cnvrg platform. OpenFL came within the MONAI toolkit developed by intel and NVFlare was Nvidia's federated learning solution. OpenFL offers the most ease of use compared to NVFlare and Swarm learning, this allows it to run out of docker containers and be simulated that way on the cnvrg platform. It also doesn't require a license server like HPE Swarm leaning. This, however, means that it lacks some functionality that the other two have. NVFlare has both a simulation and proof of concept mode in addition to its production mode. These are meant for developers to test their models before running them on the federated setup. Because of the simplicity of OpenFL, however, a simulation mode is not really necessary. Another feature that HPE Swarm learning has that OpenFL doesn't is that the aggregation server is choses probabilistically. This allows all participating servers to have a chance at aggregation.

Below I've outlined how to set up the OpenFL demo on cnvrg using available docker containers.

First determine open ports, I've assigned them to the DIRECTOR environment variable

```
DIRECTOR=$(ss -tulpn | grep LISTEN | grep -Po -m 1 '(?<=127.0.0.1:)\d{5}')
```

Then I can run the image intel/openfl

```
docker run -it --rm --network host -e DIRECTOR=$DIRECTOR intel/openfl bash
```

Within the container, there is a straightforward CLI to begin a director in OpenFL, which is called with

```
fx director start --disable-tls -c director_config.yaml
```

The syntax is similar for the envoys like such.

```
fx envoy start -n envoy1 --disable-tls --envoy-config-path envoy_config.yaml -dh localhost -dp 50051
```

The director*config.yaml contains information about the training, the host and port should be changed to reflect the ip of the machine that's hosting the director server.*
*Similarly, the envoy_config.yaml contains information about the rank and worldsize of the server, as well as what is called the shard . The _shard* describes what the dataset looks like and provides a template for what the structure of the dataset should look like from server to server.

Once the director and envoys are running, the federation can be run interactively through a jupyter notebook or python file.

```
from openfl.interface.interactive_api.federation
client_id = 'api'
cert_dir = 'cert'
director_node_fqdn = 'localhost'
director_port = 50051

federation = Federation(
    client_id=client_id,
    director_node_fqdn=director_node_fqdn,
    director_port=director_port,
    tls=False
)
```