Converting from docker-compose to Kubernetes for AWS

To convert from docker-compose to kubernetes, one must first translate the docker-compose.yaml into the various containers for Kubernetes. For this task, kompose is applied to a copy of the original docker-compose. Before that, some adjustments are applied. Note that we are assuming that a proper enviroment for Kubernetes is already set up and containers are able to see the same files with a shared file system

- Dependencies between containers are removed; one cannot be sure if a given container is running on the same host of another container, since it is up to Kubernetes to decide
- Some volumes are not appropriately suited and are therefore fixed
- The ports for the iotapps are all returned to be 1880
- Ports in are set to be the same as the ports out
- Have all services be exposed, as long as a port was specified: it means to generate an additional file that exposes a port to access what a pod/container provides
- Fix the ports of servicemap, dashboard-builder and the wsserver

After that, the kubernetes files are generated with the volumes being converted to `persistentVolumeClaim`; additional adjustments are still needed:

- All the volumes are properly linked to the correct folder, as their generation is bound to where the `kompose` command was ran
- The persistent volumes are generated

After these steps, the initial conversion is completed; the following operations are meant to be executed if one decides to use aws with Kubernetes: * Have properly set up the ebs/efs volumes with aws. This operation cannot be performed with the tools provided by Snap4City and must be executed manually

- Set the security context for the access to the disk; docker deals with disk access without supervision, but Kubernetes on aws does not allow this and will prevent the software from accessing its data if the user running the software doesn't have the permission. The user is set to `0,0,0`, which usually means root and is therefore acceptable. Should it not be as such, you may always fix it manually later
- For commodity, some startup scripts are added as readyness probe
- The volumes are compressed into one singular volume, as all of them will read the data from the same "disk"

- The volume mounts are fixed to be compliant with nfs requirements, where applicable

All but the first point are dealt with by running the script `compatibility4nfs.py`. Services and exposure are not regenerated but the previous are still functional

You might need to install a python library before executing the script properly: `pip3 install pyyaml`

Some pods, like dashboard-db (which runs a mariadb), do not function properly with efs and the older ebs file system is used instead. This problem is beyond us and must be fixed by Amazon

The name of the volume claim is generated to be "claimnamereplaceme". You might change it one go by running the following command

```
find /home/debian/k8s-aws-script-fix/127.0.0.1/kubernetes_eks
-type f -name "*.yaml" -exec sed -i "s/claimnamereplaceme/whatever-you-like/g"
{} +
```

Be sure to replace "whatever-you-like" with the intended name

Some additional files are provided, but require additional manual fixing, such as the definition of the volume used