Upgrading DFS cluster to 5.4

This only applies to installations that are configured with DFS clustering. Native support is added in eZ Publish 5.4, and does not use legacy kernel callbacks anymore. As a consequence, you need to configure DFS on the new stack (no migration of data is required).

Assuming a typical dfs configuration, in `ezpublish_legacy/settings/override/file.ini.append.php`, like the following.

```
[ClusteringSettings]
FileHandler=eZDFSFileHandler

[eZDFSClusteringSettings]
MountPointPath=/var/nfs
DBBackend=eZDFSFileHandlerMySQLiBackend
DBHost=clusterhost
DBPort=3306
DBName=ezpublish_cluster
DBUser=clusteruser
DBPassword=clusterpassword
MetaDataTableNameCache=ezdfsfile_cache
```

Where should configuration be placed

Either `ezpublish/config/ezpublish.yml`, `ezpublish/config/config.yml` or any equivalent file that you are using.

Cluster doctrine connection

First, if the cluster database is different from the content database (and it should), you need to create a new doctrine dbal connection.

```
doctrine:
   dbal:
       connections:
           cluster:
               driver: pdo_mysql
               host: clusterhost
               port: 3306
               dbname: ezpublish_cluster
               user: clusteruser
               password: clusterpassword
               charset: UTF8
```

This connection will be made available as `doctrine.dbal.cluster_connection`.

Metadata handler configuration

Handing of file metadata in the `ezdfs` tables is handled by the `legacy_dfs_cluster` IO metadata handler. You need to declare a new one that uses the doctrine connection created above.

```
ez_io:
   metadata_handlers:
       dfs:
           legacy_dfs_cluster:
               connection: doctrine.dbal.cluster_connection
```

`dfs` is the name of our custom metadata handler, and `legacy_dfs_cluster` its type.

Flysystem adapter

In order to read and write files to the NFS mount point `/var/nfs`, you need to add a flysystem adapter. One important note is that the var storage directories will not be added when writing files, meaning that they need to be specified them in the configuration.

```
oneup_flysystem:
   adapters:
       nfs_adapter:
           local:
               directory: "/var/nfs/$var_dir$/$storage_dir$"
```

`$var_dir$` and `$storage_dir$` will be replaced by the matching configuration values, and should be used as is for legacy compatibility. The value of "directory" will be set depending on the configuration, for instance to `"/var/nfs/var/ezdemo_site/storage"`.

DFS binary data handler

The next step is to configure a binary data handler that uses the flysystem adapter we created above. It is very similar to what was done for the metadata one:

```
ez_io:
   binarydata_handlers:
       nfs:
           flysystem:
               adapter: nfs_adapter
```

Pre-Final step: configuring the metadata and binarydata handlers

The last thing to do is set eZ Publish to use the binarydata and metadata handlers we created above, in the siteaccess aware configuration:

```
ezpublish:
   system:
       default:
           io:
               metadata_handler: dfs
               binarydata_handler: nfs
```