# Nextcloud Backup

Ordner zu sichern

- Nextcloud Datenverzeichnis
- Nextclud Config-Verzeichnis
- Nextcloud Datenbank Dump

# Backup-Skript erstellen

`nano nc-backup.sh`

### Zugriffsrechte anpassen

```
chown root nc-backup.sh
chmod 0700 nc-backup.sh
```

### Skript ausführen (wenn erstellt)

`./nc-backup.sh`

# NC-restore.sh erstellen

`nano nc-resetore.sh`

### Zugriffsrechte anpassen

```
chown root nc-restore.sh
chmod 0700 nc-restore.sh
```

### Nextcloud aus Backup zurückspielen

`./nc-restore.sh ordnerbezeichnung`

# Skripte

# Backup Skript

Bei # TODO sind Anpassungen zu tätigen

```
#!/bin/bash

#
# Bash script for creating backups of Nextcloud.
#
# Version 2.0.0
#
# Usage:
#  - With backup directory specified in the script:  ./NextcloudBackup.sh
#  - With backup directory specified by parameter: ./NextcloudBackup.sh <BackupDirectory> (e.g. ./NextcloudBackup.sh /media/hdd/nextcloud_backup)
#
# The script is based on an installation of Nextcloud using nginx and MariaDB, see https://decatec.de/home-server/nextcloud-auf-ubuntu-server-18-04-lts-mit-nginx-mariadb-php-lets-encrypt-redis-und-fail2ban/
#

#
# IMPORTANT
# You have to customize this script (directories, users, etc.) for your actual environment.
# All entries which need to be customized are tagged with "TODO".
#

# Variables
backupMainDir=$1

if [ -z "$backupMainDir" ]; then
    # TODO: The directory where you store the Nextcloud backups (when not specified by args)
    backupMainDir='/media/hdd/nextcloud_backup'
fi

currentDate=$(date +"%Y%m%d_%H%M%S")

# The actual directory of the current backup - this is a subdirectory of the main directory above with a timestamp
backupdir="${backupMainDir}/${currentDate}/"
```

```
# TODO: The directory of your Nextcloud installation (this is a directory under your web root)
nextcloudFileDir='/var/www/nextcloud'

# TODO: The directory of your Nextcloud data directory (outside the Nextcloud file directory)
# If your data directory is located under Nextcloud's file directory (somewhere in the web root), the data directory should not be a separate part of the backup
nextcloudDataDir='/var/nextcloud_data'

# TODO: The directory of your Nextcloud's local external storage.
# Uncomment if you use local external storage.
#nextcloudLocalExternalDataDir='/var/nextcloud_external_data'

# TODO: The service name of the web server. Used to start/stop web server (e.g. 'systemctl start <webserverServiceName>')
webserverServiceName='nginx'

# TODO: Your web server user
webserverUser='www-data'

# TODO: The name of the database system (one of: mysql, mariadb, postgresql)
databaseSystem='mariadb'

# TODO: Your Nextcloud database name
nextcloudDatabase='nextcloud_db'

# TODO: Your Nextcloud database user
dbUser='nextcloud_db_user'

# TODO: The password of the Nextcloud database user
dbPassword='mYpAsSw0rd'

# TODO: The maximum number of backups to keep (when set to 0, all backups are kept)
maxNrOfBackups=0

# TODO: Ignore updater's backup directory in the data directory to save space
# Set to true to ignore the backup directory
ignoreUpdaterBackups=false

# File names for backup files
# If you prefer other file names, you'll also have to change the NextcloudRestore.sh script.
```

```
fileNameBackupFileDir='nextcloud-filedir.tar.gz'
fileNameBackupDataDir='nextcloud-datadir.tar.gz'

# TODO: Uncomment if you use local external storage
#fileNameBackupExternalDataDir='nextcloud-external-datadir.tar.gz'

fileNameBackupDb='nextcloud-db.sql'

# Function for error messages
errorecho() { cat <<< "$@" 1>&2; }

function DisableMaintenanceMode() {
    echo "Switching off maintenance mode..."
    sudo -u "${webserverUser}" php ${nextcloudFileDir}/occ maintenance:mode --off
    echo "Done"
    echo
}

# Capture CTRL+C
trap CtrlC INT

function CtrlC() {
    read -p "Backup cancelled. Keep maintenance mode? [y/n] " -n 1 -r
    echo

    if ! [[ $REPLY =~ ^[Yy]$ ]]
    then
        DisableMaintenanceMode
    else
        echo "Maintenance mode still enabled."
    fi

    echo "Starting web server..."
    systemctl start "${webserverServiceName}"
    echo "Done"
    echo

    exit 1
}
```

```
#
# Print information
#
echo "Backup directory: ${backupMainDir}"

#
# Check for root
#
if [ "$(id -u)" != "0" ]
then
    errorecho "ERROR: This script has to be run as root!"
    exit 1
fi

#
# Check if backup dir already exists
#
if [ ! -d "${backupdir}" ]
then
    mkdir -p "${backupdir}"
else
    errorecho "ERROR: The backup directory ${backupdir} already exists!"
    exit 1
fi

#
# Set maintenance mode
#
echo "Set maintenance mode for Nextcloud..."
sudo -u "${webserverUser}" php ${nextcloudFileDir}/occ maintenance:mode --on
echo "Done"
echo

#
# Stop web server
#
echo "Stopping web server..."
systemctl stop "${webserverServiceName}"
echo "Done"
echo
```

```
#
# Backup file directory
#
echo "Creating backup of Nextcloud file directory..."
tar -cpzf "${backupdir}/${fileNameBackupFileDir}" -C "${nextcloudFileDir}" .
echo "Done"
echo

#
# Backup data directory
#
echo "Creating backup of Nextcloud data directory..."

if [ "$ignoreUpdaterBackups" = true ] ; then
        echo "Ignoring updater backup directory"
        tar -cpzf "${backupdir}/${fileNameBackupDataDir}"  --exclude="updater-*/backups/*" -C
"${nextcloudDataDir}" .
else
        tar -cpzf "${backupdir}/${fileNameBackupDataDir}"  -C "${nextcloudDataDir}" .
fi

echo "Done"
echo

# Backup local external storage.
# Uncomment if you use local external storage
#echo "Creating backup of Nextcloud local external storage directory..."
#tar -cpzf "${backupdir}/${fileNameBackupExternalDataDir}"  -C "${nextcloudLocalExternalDataDir}" .
#echo "Done"
#echo

#
# Backup DB
#
if [ "${databaseSystem,,}" = "mysql" ] || [ "${databaseSystem,,}" = "mariadb" ]; then
  echo "Backup Nextcloud database (MySQL/MariaDB)..."

if ! [ -x "$(command -v mysqldump)" ]; then
errorecho "ERROR: MySQL/MariaDB not installed (command mysqldump not found)."
```

```
    errorecho "ERROR: No backup of database possible!"
  else
    mysqldump --single-transaction -h localhost -u "${dbUser}" -p"${dbPassword}" "${nextcloudDatabase}" >
"${backupdir}/${fileNameBackupDb}"
  fi

  echo "Done"
  echo
elif [ "${databaseSystem,,}" = "postgresql" ] || [ "${databaseSystem,,}" = "pgsql" ]; then
  echo "Backup Nextcloud database (PostgreSQL)..."

  if ! [ -x "$(command -v pg_dump)" ]; then
    errorecho "ERROR: PostgreSQL not installed (command pg_dump not found)."
    errorecho "ERROR: No backup of database possible!"
  else
    PGPASSWORD="${dbPassword}" pg_dump "${nextcloudDatabase}" -h localhost -U "${dbUser}" -f
"${backupdir}/${fileNameBackupDb}"
  fi
  
  echo "Done"
  echo
fi

#
# Start web server
#
echo "Starting web server..."
systemctl start "${webserverServiceName}"
echo "Done"
echo

#
# Disable maintenance mode
#
DisableMaintenanceMode

#
# Delete old backups
#
if [ ${maxNrOfBackups} != 0 ]
```

```
then
    nrOfBackups=$(ls -l ${backupMainDir} | grep -c ^d)

    if [[ ${nrOfBackups} > ${maxNrOfBackups} ]]
    then
        echo "Removing old backups..."
        ls -t ${backupMainDir} | tail -$(( nrOfBackups - maxNrOfBackups )) | while read -r dirToRemove; do
            echo "${dirToRemove}"
            rm -r "${backupMainDir}/${dirToRemove:?}"
            echo "Done"
            echo
        done
    fi
fi

echo
echo "DONE!"
echo "Backup created: ${backupdir}"
```

# Restore Skript

Bei # Todo sind Anpassungen zu tätigen

```
#!/bin/bash

#
# Bash script for restoring backups of Nextcloud.
#
# Version 2.0.0
#
# Usage:
#   - With backup directory specified in the script: ./NextcloudRestore.sh <BackupName> (e.g. ./NextcloudRestore.sh 20170910_132703)
#   - With backup directory specified by parameter: ./NextcloudRestore.sh <BackupName> <BackupDirectory> (e.g. ./NextcloudRestore.sh 20170910_132703 /media/hdd/nextcloud_backup)



#
# IMPORTANT
# You have to customize this script (directories, users, etc.) for your actual environment.
# All entries which need to be customized are tagged with "TODO".
```

```
#

# Variables
restore=$1
backupMainDir=$2

if [ -z "$backupMainDir" ]; then
    # TODO: The directory where you store the Nextcloud backups (when not specified by args)
    backupMainDir='/media/hdd/nextcloud_backup'
fi

echo "Backup directory: $backupMainDir"

currentRestoreDir="${backupMainDir}/${restore}"

# TODO: The directory of your Nextcloud installation (this is a directory under your web root)
nextcloudFileDir='/var/www/nextcloud'

# TODO: The directory of your Nextcloud data directory (outside the Nextcloud file directory)
# If your data directory is located under Nextcloud's file directory (somewhere in the web root), the data directory should not be restored separately
nextcloudDataDir='/var/nextcloud_data'

# TODO: The directory of your Nextcloud's local external storage.
# Uncomment if you use local external storage.
#nextcloudLocalExternalDataDir='/var/nextcloud_external_data'

# TODO: The service name of the web server. Used to start/stop web server (e.g. 'systemctl start <webserverServiceName>')
webserverServiceName='nginx'

# TODO: Your web server user
webserverUser='www-data'

# TODO: The name of the database system (one of: mysql, mariadb, postgresql)
databaseSystem='mariadb'

# TODO: Your Nextcloud database name
nextcloudDatabase='nextcloud_db'

# TODO: Your Nextcloud database user
```

```
dbUser='nextcloud_db_user'

# TODO: The password of the Nextcloud database user
dbPassword='mYpAsSw0rd'

# File names for backup files
# If you prefer other file names, you'll also have to change the NextcloudBackup.sh script.
fileNameBackupFileDir='nextcloud-filedir.tar.gz'
fileNameBackupDataDir='nextcloud-datadir.tar.gz'

# TODO: Uncomment if you use local external storage
#fileNameBackupExternalDataDir='nextcloud-external-datadir.tar.gz'

fileNameBackupDb='nextcloud-db.sql'

# Function for error messages
errorecho() { cat <<< "$@" 1>&2; }

#
# Check if parameter(s) given
#
if [ $# != "1" ] && [ $# != "2" ]
then
    errorecho "ERROR: No backup name to restore given, or wrong number of parameters!"
    errorecho "Usage: NextcloudRestore.sh 'BackupDate' ['BackupDirectory']"
    exit 1
fi

#
# Check for root
#
if [ "$(id -u)" != "0" ]
then
    errorecho "ERROR: This script has to be run as root!"
    exit 1
fi

#
# Check if backup dir exists
#
if [ ! -d "${currentRestoreDir}" ]
```

```
then
    errorecho "ERROR: Backup ${restore} not found!"
    exit 1
fi

#
# Check if the commands for restoring the database are available
#
if [ "${databaseSystem,,}" = "mysql" ] || [ "${databaseSystem,,}" = "mariadb" ]; then
    if ! [ -x "$(command -v mysql)" ]; then
        errorecho "ERROR: MySQL/MariaDB not installed (command mysql not found)."
        errorecho "ERROR: No restore of database possible!"
        errorecho "Cancel restore"
        exit 1
    fi
elif [ "${databaseSystem,,}" = "postgresql" ] || [ "${databaseSystem,,}" = "pgsql" ]; then
    if ! [ -x "$(command -v psql)" ]; then
        errorecho "ERROR: PostgreSQL not installed (command psql not found)."
        errorecho "ERROR: No restore of database possible!"
        errorecho "Cancel restore"
        exit 1
    fi
fi

#
# Set maintenance mode
#
echo "Set maintenance mode for Nextcloud..."
sudo -u "${webserverUser}" php ${nextcloudFileDir}/occ maintenance:mode --on
echo "Done"
echo

#
# Stop web server
#
echo "Stopping web server..."
systemctl stop "${webserverServiceName}"
echo "Done"
echo

#
```

```
# Delete old Nextcloud directories
#

# File directory
echo "Deleting old Nextcloud file directory..."
rm -r "${nextcloudFileDir}"
mkdir -p "${nextcloudFileDir}"
echo "Done"
echo

# Data directory
echo "Deleting old Nextcloud data directory..."
rm -r "${nextcloudDataDir}"
mkdir -p "${nextcloudDataDir}"
echo "Done"
echo

# Local external storage
# TODO: Uncomment if you use local external storage
#echo "Deleting old Nextcloud local external storage directory..."
#rm -r "${nextcloudLocalExternalDataDir}"
#mkdir -p "${nextcloudLocalExternalDataDir}"
#echo "Done"
#echo

#
# Restore file and data directory
#

# File directory
echo "Restoring Nextcloud file directory..."
tar -xmpzf "${currentRestoreDir}/${fileNameBackupFileDir}" -C "${nextcloudFileDir}"
echo "Done"
echo

# Data directory
echo "Restoring Nextcloud data directory..."
tar -xmpzf "${currentRestoreDir}/${fileNameBackupDataDir}" -C "${nextcloudDataDir}"
echo "Done"
echo
```

```
# Local external storage
# TODO: Uncomment if you use local external storage
#echo "Restoring Nextcloud data directory..."
#tar -xmpzf "${currentRestoreDir}/${fileNameBackupExternalDataDir}" -C "${nextcloudLocalExternalDataDir}"
#echo "Done"
#echo

#
# Restore database
#
echo "Dropping old Nextcloud DB..."

if [ "${databaseSystem,,}" = "mysql" ] || [ "${databaseSystem,,}" = "mariadb" ]; then
    mysql -h localhost -u "${dbUser}" -p"${dbPassword}" -e "DROP DATABASE ${nextcloudDatabase}"
elif [ "${databaseSystem,,}" = "postgresql" ]; then
sudo -u postgres psql -c "DROP DATABASE ${nextcloudDatabase};"
fi

echo "Done"
echo

echo "Creating new DB for Nextcloud..."

if [ "${databaseSystem,,}" = "mysql" ] || [ "${databaseSystem,,}" = "mariadb" ]; then
    # Use this if the databse from the backup uses UTF8 with multibyte support (e.g. for emoijs in filenames):
    mysql -h localhost -u "${dbUser}" -p"${dbPassword}" -e "CREATE DATABASE ${nextcloudDatabase}
CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci"
    # TODO: Use this if the database from the backup DOES NOT use UTF8 with multibyte support (e.g. for emoijs
in filenames):
    #mysql -h localhost -u "${dbUser}" -p"${dbPassword}" -e "CREATE DATABASE ${nextcloudDatabase}"
elif [ "${databaseSystem,,}" = "postgresql" ] || [ "${databaseSystem,,}" = "pgsql" ]; then
    sudo -u postgres psql -c "CREATE DATABASE ${nextcloudDatabase} WITH OWNER ${dbUser} TEMPLATE
template0 ENCODING \"UTF8\";"
fi

echo "Done"
echo

echo "Restoring backup DB..."

if [ "${databaseSystem,,}" = "mysql" ] || [ "${databaseSystem,,}" = "mariadb" ]; then
```

```
    mysql -h localhost -u "${dbUser}" -p"${dbPassword}" "${nextcloudDatabase}" <
"${currentRestoreDir}/${fileNameBackupDb}"
elif [ "${databaseSystem,,}" = "postgresql" ] || [ "${databaseSystem,,}" = "pgsql" ]; then
    sudo -u postgres psql "${nextcloudDatabase}" < "${currentRestoreDir}/${fileNameBackupDb}"
fi

echo "Done"
echo

#
# Start web server
#
echo "Starting web server..."
systemctl start "${webserverServiceName}"
echo "Done"
echo

#
# Set directory permissions
#
echo "Setting directory permissions..."
chown -R "${webserverUser}":"${webserverUser}" "${nextcloudFileDir}"
chown -R "${webserverUser}":"${webserverUser}" "${nextcloudDataDir}"
# TODO: Uncomment if you use local external storage
#chown -R "${webserverUser}":"${webserverUser}" "${nextcloudLocalExternalDataDir}"
echo "Done"
echo

#
# Update the system data-fingerprint (see
https://docs.nextcloud.com/server/latest/admin_manual/configuration_server/occ_command.html#maintenance-
commands-label)
#
echo "Updating the system data-fingerprint..."
sudo -u "${webserverUser}" php ${nextcloudFileDir}/occ maintenance:data-fingerprint
echo "Done"
echo

#
# Disbale maintenance mode
#
```

```
echo "Switching off maintenance mode..."
sudo -u "${webserverUser}" php ${nextcloudFileDir}/occ maintenance:mode --off
echo "Done"
echo


echo
echo "DONE!"
echo "Backup ${restore} successfully restored."
```

---

Revision #1
Created 21 March 2023 10:14:30 by Hermann
Updated 21 March 2023 10:14:59 by Hermann