Server preparation - Backup and Recovery

When it comes to backup, the CzechIdM deployments consists of three parts:

- Identity manager's data in the PostgreSQL database.
- Application's `idm.war` archive in the Tomcat.
- Configuration folders under `/opt/czechidm`.

Depending on your deployment, there can also be sets of scripts, the Vault and so on. But those are highly deployment-specific.

All backups should be shipped off the production machine. In case the production server fails, it must be possible to rebuild it from scratch (documentation) and to restore the repository and configuration from recent backup.

Repository backups

Using PostgreSQL allows us to do the `pg\_dump` of the repository which is a primary backup strategy. Because identity manager contains company's private data, it is desirable to encrypt the repository backup. We will store all backups in the `/opt/backup/…` location. To create encrypted backups, we will use this shell script.

The script needs a public-private keypair to be set up. When making a backup, a symmetric key will be generated. This symmetric key is used to encrypt the actual backup. Symmetric key is then asymmetrically encrypted by the public key and stored alongside the backup. For recovery, you need the backup and its corresponding symmetric key. When recovering, first, you have to decrypt the symmetric key with the private key generated earlier. Then you will use symmetric key to restore the data backup. For instructions about keypair initialization, backup creation and recovery and also for the actual command to carry out these actions, please refer to the script itself.

When you obtain the repository backup, you can restore the repository:

1. Stop the identity manager container.
2. Backup current repository somewhere else - in case you need to check some data later.
3. Delete all data from the repository / drop the repository itself. This depends on the backup created - if you have database creation statements in there and such.
4. Restore the data from the pgdump with `psql [parameters] < your\_backup\_name.sql`
5. Start the identity manager.

Script deployment

This is a standard deployment scenario. First, create the directory structure and setup the script and keys:

```
mkdir -p /opt/backup/database
chown -Rf postgres:postgres /opt/backup
```

```
chmod 700 /opt/backup/database
# deploy the script above:
vim /opt/backup/enc_backup_database.sh
chown root:postgres /opt/backup/enc_backup_database.sh
chmod 750 /opt/backup/enc_backup_database.sh
# create public-private keypair on YOUR machine, NOT on the CzechIdM server
# copy ONLY the public key to the CzechIdM server, STORE the private key
SAFELY
scp publickey.key czechidm-server:/opt/backup/backup_database-rsa.pub
chown root:postgres /opt/backup/backup_database-rsa.pub
chmod 440 /opt/backup/backup_database-rsa.pub
```

Then, configure the backup script:

```
BACKUP_ROOT="/opt/backup"
BACKUP_LOC="${BACKUP_ROOT}/database"
BACKUP_PREFIX="backup_database."
BACKUP_AES_KEY_PREFIX="backup_database."
RSA_ENC_KEY_FILE="${BACKUP_ROOT}/backup_database-rsa.pub"
```

Plain version of the script does not come with commands for making the actual backup, script serves as a wrapper which will encrypt whatever you need. Therefore, you have to check the script if there is actually a backup command specified and, if not, fill it in. If you want, you can add other things to the encrypted backup. See the script for lines:

```
#do the dump
# say we run the actual backup and create dump1.dmp, dump2.dmp and dump3.dmp
here

#pack the dump
#tar usage "tar [parameters] archive_name file1 [file2 file3 ...]"
tar --remove-files -czf current_backup.tgz dump1.dmp dump2.dmp dump3.dmp
```

And change them to (expected name of the czechidm database is `czechidm`):

```
#do the dump
pg_dump --create --dbname=czechidm > czechidm.sql

#pack the dump
#tar usage "tar [parameters] archive_name file1 [file2 file3 ...]"
tar --remove-files -czf current_backup.tgz czechidm.sql
```

Last thing to do is to set up a cronjob. Here we set up a daily backup to 04:00. It may be useful to change the time the backup is done because there may be batch jobs running inside the identity manager. Set the job to run under the postgres user.

```
crontab -l -u postgres
00 04 * * * /opt/backup/enc_backup_database.sh
```

Configuration backups

Everything about the identity manager lives in `/opt/czechidm`, the configuration itself in the `/opt/czechidm/etc/` directory. Some files there (logging configuration, Quartz configuration and most parts of application property file) can be reconstructed from documentation. Crucial parts (repository password and especially `secret.key`) have to be backed up.

- Losing repository password is not a problem (you can always set the new password in postgres) but it is a complication.
- Losing `secret.key` effectively means **losing all contents of the Confidential Storage**.

For simplicity reasons (and also because all the files count up to few kilobytes in size) we recommend backing up whole configuration folder. **It is vital that this backup is encrypted, as it contains the confidential storage key.**

Implementation

- For implementation, use the encrypted backups script as above.
- Create new instance of the script.
- Ideally generate separate RSA keypair for it.
- Do not pack the `secret.key` and repository backup together as this potentially lowers the security. (However, this somewhat depends where the backups are stored when off machine.)

Application backups

CzechIdM is a Java application distributed as a WAR archive. This archive is deployed inside a Apache Tomcat container. For recovery of the application, only the WAR is needed.

- If you use standard application release without any added modules and customizations, the `idm.war` **does not need to be backed up**. You can always download the application release again.
- If you customized identity manager's binary in any way, you **should rely on your version control system and development lifecycle** and you should be always able to produce the `idm.war` again.
- If you still want to backup the `idm.war`, simply copying it somewhere safe is completely sufficient. This could look something like:

backup_app.sh

```
#!/bin/bash

BACKUP_ROOT="/opt/backup"
BACKUP_DIR="${BACKUP_ROOT}/app"
LOCKFILE="${BACKUP_ROOT}/`basename ${0}`.lock"
BACKUP_KEEP_DAYS="30"

NOW=$(date +"%Y-%m-%d-%H%M%S")

if [ -f "${LOCKFILE}" ]; then
```

```
    echo "Backup is already running, exitting..." >&2
    exit 1
fi
touch "${LOCKFILE}"

tar cpfz "${BACKUP_DIR}/backup_app.${NOW}.tgz"
/opt/tomcat/current/webapps/* 2>/dev/null || echo "Error when
performing backup." >&2
find "$BACKUP_DIR" -name "*tgz" -type f -mtime
"+${BACKUP_KEEP_DAYS}" -delete
rm "${LOCKFILE}"
exit 0
```

When doing the recovery, simply put the CzechIdM WAR archive into the same folder as you would normally do and restart the Tomcat container.

Script deployment

If you really want to use the `backup\_app.sh`, here is an example deployment scenario:

```
mkdir -p /opt/backup/app
# this is because we usually do repository backup to sibling directories in
the tree
chown postgres:postgres /opt/backup
chown root:root /opt/backup/app
chmod 700 /opt/backup/app
# deploy the script above:
vim /opt/backup/backup_app.sh
chown root:root /opt/backup/backup_app.sh
chmod 750 /opt/backup/backup_app.sh
```

At last, set up a cronjob:

```
crontab -l -u root
50 03 * * * /opt/backup/backup_app.sh
```

In some cases, CzechIdM is not deployed with frontend and backend bundled together in the `idm.war`. When backing up such environment, the backend should be backed up the way as was just described. The frontend, which may be deployed somewhere else, should be backed up in a similar way using the same script. For example, when running frontend application from separate Apache HTTPD, you should deploy another backup script which backs up `/var/www/html/\*` directory instead of `/opt/tomcat/current/webapps/\*`.