Troubleshooting Central

Reading container logs

If Central is behaving in an unexpected way, it is often helpful to read the Docker container logs with the docker logs command for hints as to what has gone wrong.

The containers that are most likely to be helpful for troubleshooting are the *service* container and the *nginx* container. Use the *–tail* and *–since* options to help filter the logs. For example:

- `docker logs --tail 100 central-service-1` to see the last 100 lines of the service logs.
- `docker logs --since 5m central-nginx-1` to see the last 5 minutes of the nginx logs.

Other commands that are helpful are:

- `docker ps` to see the status of all containers.
- `docker stats` to see the CPU and RAM usage of all containers.

Users aren't receiving emails

Central uses email as a way to verify user identity when setting or changing passwords. This helps ensure that only the intended user has access to their Central account.

Email sounds like a simple technology but in practice there are many things that can cause message delivery issues. By default, Central is installed with a mail server which can be used without configuration. However, it will not work in every environment. For example:

- Many cloud providers restrict the usage of simple mail servers such as Central's as a spam-prevention strategy
- You can be assigned an IP address that was previously used for sending spam and is therefore blocked by many mail recipients
- Your domain may not be recognized by mail recipients and therefore messages from it may be discarded or marked as spam

To address delivery issues, consider using a dedicated email service such as Mailjet. Because Central doesn't send very many emails, using such a service will generally be a cost-effective way of ensuring email delivery. Once you have an account set up, you will need to configure Central to use it.

If you want to directly send emails from your Central installation, the mail-tester service can help you identify what barriers to email delivery you might have. Create a Central account with the email address that it provides, retrieve your results, and then delete the user. Typically, the first thing you will need to do is configure DKIM which will provide email recipients confidence that emails were actually sent by your Central server rather than by a spammer pretending to be your server.

Preview could not connect with server

You may run into a "Could not connect with Server" error message when previewing forms. This error message is typically because your host machine has not made its DNS servers available to Central.

To resolve this problem, first identify your upstream DNS servers. Run `cat /run/systemd/resolve/resolv.conf` to see your current list of nameservers with their IP addresses. They will look like this:

```
nameserver 1.2.3.4
nameserver 9.8.7.6
```

Now, run `nano /etc/docker/daemon.json` to make those nameservers and, optionally, the Google DNS (8.8.8.8) as a fallback available to Docker. Your `daemon.json` file will look like the snippet below, but you will need to replace `1.2.3.4` and `9.8.7.6` with your nameservers' IP addresses.

```
    "dns": ["1.2.3.4", "9.8.7.6", "8.8.8.8"]
}
```

Finally, stop the containers, restart Docker, and bring the containers back up with `docker compose stop`, `systemctl restart docker` and `docker compose up -d`.

Migration fails due to out-of-memory error

During upgrades, some versions of Central may perform complex database migrations that need more memory than the 2GB typically allocated to Central.

If you get an error suggesting that the JavaScript heap is out of memory, try increasing allocated memory.

Export produces corrupt zip

If you have installed Central on a 1GB server or your forms collect many large media files, you may encounter problems exporting submission .zip files. Usually, the .zip file will end up being empty, or much smaller than expected and possibly corrupt.

If you are expecting to collect media files, we recommend having at least 2GB of memory. When collecting images, we recommend specifying a maximum size in form design.

If you still run into problems, try increasing allocated memory.

File upload fails with 413

If you get an error *413* when trying to upload a submission or when trying to upload a form attachment, the file you are trying to upload is too large. By default, files up to 100MB are accepted. We typically recommend reducing the size of the files to upload if possible. For example, images can be scaled down in form design.

If you absolutely must upload files over 100MB, you can change the *client_max_body_size nginx* directive:

```
$ cd central
$ docker compose stop
$ nano files/nginx/odk.conf.template
<modify the nginx conf value for client_max_body_size>
$ docker compose up -d
```

Database reset after running Docker command

> ⚠ Warning
>
> If you are experiencing data loss, the most important thing to do first is to stop and think through your next steps (ideally with a colleague, who can review those steps). Rushing to act without a plan will most certainly make the situation worse.
>
> If you do not have a backup, do not reboot or restart the machine. Instead, take a live, full disk backup of the machine so you have a fallback. If you are using DigitalOcean, see how to create snapshots.

In Central v2023.1 or earlier, it is possible to accidentally reset the database by running the `down` command with `docker-compose`. This no longer happens in Central v2023.2 or later because the default database is stored on a named volume. If you are running an older Central version, you have run this command and your database has reset, follow these steps to restore your data.

The instructions below assume you installed Central on an Ubuntu LTS server. If you did not, or do not feel confident following the steps below, email support@getodk.org for assistance.

1. Capture the location of the new (and empty) database.

```
$ CENTRAL_NEW_DB=$(docker inspect --type container central_postgres_1 \
  -f '{{(index .Mounts 0).Source}}' | cut -d / -f 6)
```

2. Next, find any additional databases you have. You should get the number one (`1`) back. If you get anything else, stop and email support@getodk.org for assistance.

```
$ find /var/lib/docker/volumes/ -name pg_hba.conf \
  | grep -v "$CENTRAL_NEW_DB" | wc -l
```

3. Now that you've confirmed you have only one additional database, capture the location of the old database you wish to restore.

```
$ CENTRAL_OLD_DB=$(find /var/lib/docker/volumes/ -name pg_hba.conf \
  | grep -v "$CENTRAL_NEW_DB" | cut -d / -f 6)
```

4. Stop the Docker containers to prepare for restoration.

```
$ cd central
$ docker-compose stop
```

5. Backup the new database and restore the old database.

```
$ cd /var/lib/docker/volumes/
$ mv "$CENTRAL_NEW_DB" "$CENTRAL_NEW_DB"-backup
$ mv "$CENTRAL_OLD_DB" "$CENTRAL_NEW_DB"
```

6. Now rebuild and restart the containers.

```
$ cd central
$ docker-compose build
$ docker-compose up -d
```

7. Go to your site in a browser and try to log in with an account that previously existed. If everything works as expected, consider deleting the backup of the new database. You can find it with the following command.

```
$ find /var/lib/docker/volumes/ -name *-backup
```