# Container basiertes Webhosting - unser Container-Hafen

- Ältere Dokumentation auf altem Server mit Webdav-Zugang

Mit dem „Container-Haven“ bietet das Datenkollektiv ein Container-basiertes Webhosting an. Container sind eine Art der Virutalisierung, die sich gegenüber einer kompletten Virtualisierung durch deutlich bessere Performance auszeichnet und sehr wenig Ressourcen benötigt. Dies ermöglicht, eine Vielzahl von Containern parallel auf einem Server zu betreiben. Die einzelnen Container verhalten sich dabei wie eigenständige Server.

Gleichzeitig sorgen wir im Hintergrund für die nötigen Betriebssystem-Updates. Sie müssen sich als Nutzer_in also nur um Ihre Web-Applikationen kümmern - sofern Sie nicht auch dort z.B. das vorinstallierte Wordpress nutzen. Auch dort kümmern wir uns um die Updates.

Dies ist sozusagen ein **Mittelweg** zwischen **„einfachem Webspace“** und einem **V-Server** mit den folgenden Vorteilen:

- Die Dateien in einer Instanz, sind vollständig gegenüber anderen Instanzen isoliert
- Bei hoher Sicherheit haben die einzelnen User große individuelle Konfigurationsmöglichkeiten
- Wir kümmern uns um Sicherheitsupdates
- Ein tägliches Backup der Daten erfolgt automatisch

Wenn Sie sich für Container-basiertes Webhosting interessieren, sprechen Sie uns an.

## Zugangsdaten

Als Zugangsdaten sind notwendig:

- ein Username für den Container (in der Regel „webcXXX“)
- ein Passwort, das bei der Einrichtung übermittelt wird. (Passwort-Änderung unter: https://admin.datenkollektiv.net/)
- URL, unter der der Container verwaltet wird - üblicherweise der Domain-Name wie z.B. USERNAME@hosting.dknuser.de
- der Username und ein Passwort für die Mysql-Datenbanken ist identisch mit dem Usernamen des Containers. Das Passwort wird bei der Einrichtung übermittelt.

## Zugang zum Container

Es gibt mehrere Möglichkeiten des Zugangs zum Container. Datei-Up- und Download lässt sich am einfachsten über sftp vornehmen.

Alle wichtigen Informationen dazu gibt es in der initialen E-Mail nach der Erstellung eines neuen Containers.

## sftp

Der SFTP-Zugang ist:

```
USERNAME@USERNAME.hosting.dknuser.de
```

## Datenbanken konfigurieren

Die Datenbanken sind unter folgender URL zu erreichen:

```
http://sqladmin.dknuser.de/
```

Das Login ist zweistufig. Aus Sicherheitsgründen ist ein ht-Passwort vorgeschaltet, Usernamen und Passwort sind identisch mit denen des Zugangs (sftp-User/Passwort)

Login mit Datenbank-Username und Passwort. Dort können beliebige Datenbanken erstellt werden, deren Namen mit dem Usernamen gefolgt von einem „_“ beginnen müssen. Also z.B.

```
USERNAME_wordpress
```

Als Datenbank-Server muss in den eigenen Instanzen dann folgendes eingetragen werden:

```
sql0.datenkollektiv.net
```

## Wordpress

In den Containern ist evtl. eine Wordpress-Instanz vorinstalliert. Diese Wordpress-Instanz kommt aus den Debian-Repositories. Es ist immer etwas älter als das aktuelle Release. Dafür ist es gut ins System integriert - und bekommt über Debian Sicherheitsupdates - allerdings nicht immer sofort.

Das Daten-Verzeichnis für diese Installation, auf das per sftp zugegriffen werden kann liegt direkt unter `/wordpress/wp-content/`.

## Eigenes Wordpress installieren

Alternativ lässt sich aber auch leicht ein eigenes Worpress im Webroot installieren. Das SFTP-Verzeichnis ist dann `/www/` - dort kann dann z.B. ein Verzeichnis `wordpress` angelegt werden.

Für diese Wordpress-Instanz ist dann entsprechend dieses Verzeichnis zuständig - und nicht das der systemweiten Installation (siehe oben), z.B.:

```
/www/wordpress/wp-content
```

Bei einer selbst installierten Wordpress-Instanz im Webroot muss ggf. die Webserver-Konfigurtion angepasst werden, die sich unter

```
/nginx/sites-user
```

befindet.

Siehe auch unten: Webserver konfigurieren

Die Zeile

```
server_name webc08.hosting.dknuser.de;
```

muss an den tatsächlichen Seitennamen angepasst werden, also z.B.:

```
server_name example.org;
```

Da der Webserver im Container hinter einem Webproxy liegt, muss ggf. in der Wordpress-Config

```
wp-config.php
```

folgendes vor der Zeile

```
require_once(ABSPATH . 'wp-settings.php');
```

hinzugefügt werden:

```
/**
 *  * Handle SSL reverse proxy
**/
if ($_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https')
            $_SERVER['HTTPS']='on';

if (isset($_SERVER['HTTP_X_FORWARDED_HOST'])) {
            $_SERVER['HTTP_HOST'] = $_SERVER['HTTP_X_FORWARDED_HOST'];
}
```

- vgl. auch: https://www.variantweb.net/blog/wordpress-behind-an-nginx-ssl-reverse-proxy/

Sinnvoll ist es auch Wordpress so zu konfigurieren, dass die Instanz selbst Dateien hochladen kann - sonst klappt z.B. das automatische Installieren von Plugins nicht. Dazu bedarf es folgender Zeile in der wp-config.php (wiederum vor der letzten Zeile `require_once(ABSPATH . 'wp-settings.php');`:

```
define('FS_METHOD','direct');
```

# Eigene html-Seiten / Php-Skripte

Der Webroot liegt unter `/www/html/` - und ist per sftp zugänglich. Alle Dateien und Skripte können dort hochgeladen werden.

# Webserver konfigurieren

Fortgeschrittene User_innen können auch die Webserver-Konfiguration selbst verändern. Dazu sind Kenntnisse in der Konfiguration des Nginx-Webservers nötig. Außerdem muss beachtet werden, dass der Webserver hinter einem Webproxy liegt - und z.B. nicht auf alle Header-Variablen direkt zugegriffen werden kann. Außerdem geschieht die Verbindung zum Webproxy grundsätzlich über Port 80 und http. Sämtliche Konfiguration von Zertifikaten geschieht auf dem Proxy-Server durch die Administratoren des datenkollektiv.

Der Basis-Webroot-Pfad für das Verzeichnis `www` ist:

```
/var/www/
```

Um Änderungen am Webserver wirksam zu machen, muss dieser neu gestartet werden. Dazu bitte in der Datei

```
/www/nginx_reload.txt
```

Die Werte entsprechend setzen:

```
reload=0
restart=1
```

Nach einer Minute wird der Webserver automatisch neu gestartet (die Werte in der ngnix_reload.txt sind dann wieder zurück auf `0` gesetzt).

Beispiele:

## Eigene Wordpress-Installation

wordpress.conf

```
server {
    listen  80;
    listen [::]:80;

    server_name          example.org;
    root                        /var/www/wordpress;
    index                       index.php;
```

```
    location = /favicon.ico {
        log_not_found            off;
        access_log               off;
    }

    location = /robots.txt {
        allow                    all;
        log_not_found            off;
        access_log               off;
    }

    location / {
        # This is cool because no php is touched for static content.
        # include the "?$args" part so non-default permalinks doesn't
break when using query string
        try_files                $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include                  /etc/nginx/fastcgi_params;
        fastcgi_param            SCRIPT_FILENAME
$request_filename;
        fastcgi_index            index.php;
        fastcgi_intercept_errors on;
        fastcgi_pass             unix:/var/run/php7.0-fpm-www-
data.sock;

    }

    location ~* \.(js|css|png|jpg|jpeg|gif|ico)$ {
        expires                  max;
        log_not_found            off;
    }
}
```

## Statische Seite unter /www/html/