Container 'cloud'

Ressourcen

- 1 GB RAM
- 2 Cores
- 5 GB HDD (root-fs)
- 35 GB HDD (/var/www)

System

- interne IPs
  - 10.2.0.14, fd00:10:2:0::14
  - 10.3.0.14, fd00:10:3:0::14

Dienste

- Nextcloud (via NGINX)

Betrieb

Nextcloud aktualisieren (auf der Konsole)

1. Updater ausführen
   - **sudo -u www-data php /var/www/public_html/updater/updater.phar**
     - Start update? [y/N] **y**
     - Should the „occ upgrade“ command be executed? [Y/n] **y**
     - Keep maintenance mode active? [y/N] **n**
2. In Nextcloud einloggen und unter Einstellungen - Verwaltung - Übersicht prüfen. Folgende Fehler sind „normal“:
   - Dein Web-Server ist nicht richtig eingerichtet um „/.well-known/caldav“ aufzulösen.
   - Dein Web-Server ist nicht richtig eingerichtet um „/.well-known/carddav“ aufzulösen.
   - Es wurde kein PHP-Memory-Cache konfiguriert.
3. Angezeigte Fehler prüfen und korrigieren
   1. Nutzung von OCC: **sudo -u www-data /var/www/public_html/occ ...**

Backup

```
Dateien per Shell sichern
  - cd /var/www/public_html
  - sudo -u www-data php occ maintenance:mode --on
  - sudo tar -cpzf //nc_backup_`date +"%Y%m%d"`.tar.gz -C /var/www/public_html/ .
```

Installation

- Standard-Template mit Benutzern

NGINX / PHP

1. NGINX und PHP-FPM installieren
   - **sudo apt-get install php7.3 php7.3-cli php7.3-fpm php7.3-curl php7.3-gd php7.3-json php7.3-xml php7.3-mbstring php7.3-zip php7.3-mysql php7.3-bz2 php7.3-intl php7.3-bcmath php7.3-gmp php-imagick nginx**
2. Default-Konfiguration anpassen

   /etc/nginx/sites-available/default

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

```
    root /var/www/public_html;

    # Add index.php to the list if you are using PHP
        index index.php index.html /index.php$request_uri;

    server_name _;

        # set max upload size
        client_max_body_size 512M;
        fastcgi_buffers 64 4K;

        # Enable gzip but do not remove ETag headers
        gzip on;
        gzip_vary on;
        gzip_comp_level 4;
        gzip_min_length 256;
        gzip_proxied expired no-cache no-store private no_last_modified no_etag auth;
        gzip_types application/atom+xml application/javascript application/json
application/ld+json application/manifest+json application/rss+xml
application/vnd.geo+json application/vnd.ms-fontobject application/x-font-ttf
application/x-web-app-manifest+json application/xhtml+xml application/xml font/opentype
image/bmp image/svg+xml image/x-icon text/cache-manifest text/css text/plain text/vcard
text/vnd.rim.location.xloc text/vtt text/x-component text/x-cross-domain-policy;

        # HTTP response headers borrowed from Nextcloud `.htaccess`
        add_header Referrer-Policy                      "no-referrer"   always;
        add_header X-Content-Type-Options               "nosniff"       always;
        add_header X-Download-Options                   "noopen"        always;
        add_header X-Frame-Options                      "SAMEORIGIN"    always;
        add_header X-Permitted-Cross-Domain-Policies    "none"          always;
        add_header X-Robots-Tag                         "none"          always;
        add_header X-XSS-Protection                     "1; mode=block" always;

        # Remove X-Powered-By, which is an information leak
        fastcgi_hide_header X-Powered-By;

        # Rule borrowed from `.htaccess` to handle Microsoft DAV clients
        location = / {
            if ( $http_user_agent ~ ^DavClnt ) {
                return 302 /remote.php/webdav/$is_args$args;
            }
        }

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

        # Make a regex exception for `/.well-known` so that clients can still
        # access it despite the existence of the regex rule
        # `location ~ /(\.|autotest|...)` which would otherwise handle requests
        # for `/.well-known`.
        location ^~ /.well-known {
            # The following 6 rules are borrowed from `.htaccess`
            rewrite ^/\.well-known/host-meta\.json  /public.php?service=host-meta-json
last;
            rewrite ^/\.well-known/host-meta        /public.php?service=host-meta
last;
            rewrite ^/\.well-known/webfinger        /public.php?service=webfinger
last;
```

```
            rewrite ^/\.well-known/nodeinfo         /public.php?service=nodeinfo
last;
            location = /.well-known/carddav     { return 301 /remote.php/dav/; }
            location = /.well-known/caldav      { return 301 /remote.php/dav/; }
            try_files $uri $uri/ =404;
        }

        # Rules borrowed from `.htaccess` to hide certain paths from clients
        location ~ ^/(?:build|tests|config|lib|3rdparty|templates|data)(?:$|/)  {
return 404; }
        location ~ ^/(?:\.|autotest|occ|issue|indie|db_|console)              { return
404; }

        # Ensure this block, which passes PHP files to the PHP process, is above the
blocks
        # which handle static assets (as seen below). If this block is not declared
first,
        # then Nginx will encounter an infinite rewriting loop when it prepends
`/index.php`
        # to the URI, resulting in a HTTP 500 error response.
        location ~ \.php(?:$|/) {
            fastcgi_split_path_info ^(.+?\.php)(/.*)$;
            set $path_info $fastcgi_path_info;

            try_files $fastcgi_script_name =404;

            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $path_info;
            fastcgi_param HTTPS on;

            fastcgi_param modHeadersAvailable true;         # Avoid sending the
security headers twice
            fastcgi_param front_controller_active true;     # Enable pretty urls
            #fastcgi_pass php-handler;
        fastcgi_pass unix:/run/php/php7.3-fpm.sock;

            fastcgi_intercept_errors on;
            fastcgi_request_buffering off;
        }

        location ~ \.(?:css|js|svg|gif)$ {
            try_files $uri /index.php$request_uri;
            expires 6M;         # Cache-Control policy borrowed from `.htaccess`
            access_log off;     # Optional: Don't log access to assets
        }

        location ~ \.woff2?$ {
            try_files $uri /index.php$request_uri;
            expires 7d;         # Cache-Control policy borrowed from `.htaccess`
            access_log off;     # Optional: Don't log access to assets
        }

    location / {
                try_files $uri $uri/ /index.php$request_uri;
    }

}
```

3. PHP-Konfiguration für Upload anpassen

   /etc/php/7.3/fpm/pool.d/www.conf

```
...
clear_env = no
...
env[HOSTNAME] = $HOSTNAME
env[PATH] = /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
env[TMP] = /tmp
env[TMPDIR] = /tmp
env[TEMP] = /tmp
...
```

4. PHP-Konfiguration für Upload anpassen

   /etc/php/7.3/fpm/php.ini

```
...
; Output buffering is a mechanism for controlling how much output data
; (excluding headers and cookies) PHP should keep internally before pushing that
; data to the client. If your application's output exceeds this setting, PHP
; will send that data in chunks of roughly the size you specify.
; Turning on this setting and managing its maximum buffer size can yield some
; interesting side-effects depending on your application and web server.
; You may be able to send headers and cookies after you've already sent output
; through print or echo. You also may see performance benefits if your server is
; emitting less packets due to buffered output versus PHP streaming the output
; as it gets it. On production servers, 4096 bytes is a good setting for performance
; reasons.
output_buffering = 0
...
; Maximum execution time of each script, in seconds
; http://php.net/max-execution-time
; Note: This directive is hardcoded to 0 for the CLI SAPI
max_execution_time = 1800
...
; Maximum amount of time each script may spend parsing request data. It's a good
; idea to limit this time on productions servers in order to eliminate unexpectedly
; long running scripts.
max_input_time = 1800
...
; Maximum amount of memory a script may consume (128MB)
; http://php.net/memory-limit
memory_limit = 512M
...
; Temporary directory for HTTP uploaded files (will use system default if not
; specified).
; http://php.net/upload-tmp-dir
upload_tmp_dir = /var/www/tmp
...
; Maximum size of POST data that PHP will accept.
; Its value may be 0 to disable the limit. It is ignored if POST data reading
; is disabled through enable_post_data_reading.
; http://php.net/post-max-size
post_max_size = 512M
...
; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize = 512M
...
```

5. Dienste aktivieren und neustarten
   - **sudo systemctl enable nginx.service php7.3-fpm.service**
   - **sudo systemctl restart nginx.service php7.3-fpm.service**

Nextcloud

1. Nextcloud-Verzeichnisse erstellen
    - **sudo mkdir /var/www/{public_html,data,tmp}**
2. Cronjob einrichten
    - **sudo crontab -u www-data -e**

```
*/5 * * * * php -f /var/www/public_html/cron.php
```

Es wurde eine bereits bestehende Installation übernommen.

Backup mit Borgmatic

1. Installation siehe mariadb
2. Konfiguration

/etc/borgmatic/config.yaml

```
...
location:
    # List of source directories to backup (required). Globs and
    # tildes are expanded.
    source_directories:
        - /etc
        - /home
        - /root
        - /var/log
        - /var/www/data
        - /var/www/public_html
...
```