Dateien direkt über Shell vom Webserver nach GitHub schieben

Einrichten von GIT auf Webserver

1. Wir prüfen die GIT Version mit

```
git --version
```

Als Antwort bekommt, man so etwas wie „git version 2.34.1“

2. Lokales Git-Repository auf dem Server einrichten

```
cd /www/modellbahn-anlage.de/wiki/httpdocs
git init
git remote add origin https://github.com/modellbahn-anlage/Wiki.git
git branch -M master
```

Erster Commit (wenn noch keine .git-Struktur vorhanden ist):

```
git add .
git commit -m "Initial commit Web-Server"
git push -u origin master
```

Git fragt hier nach Benutzername und Passwort – das geht nicht mehr per Klartext. Du brauchst ein GitHub-Personal Access Token (siehe nächster Punkt).

3. GitHub-Zugang per Token (ohne Passwort)

Token erzeugen:

1. Gehe zu https://github.com/settings/tokens

2. Wähle: Fine-grained token
3. Vergib:
    1. Repository-Zugriff: Nur auf modellbahn-anlage/Wiki
    2. Berechtigungen: Contents: Read and Write
4. Speichern und Token kopieren (wird nur 1× angezeigt)

Git so konfigurieren, dass der Token verwendet wird:

```
git config --global user.name "DeinGitHubBenutzername"
git config --global user.email "deine@email.de"
```

Dann die Remote-URL mit Token dauerhaft speichern:

```
git remote set-url origin
https://<TOKEN>@github.com/modellbahn-anlage/Wiki.git
```

Achtung: Diese URL enthält den Token im Klartext. Er sollte nicht öffentlich einsehbar sein (z. B. in HTML-Dateien oder Logs). Git speichert ihn aber nur lokal.

4. Übertragung per Hand anstoßen

Um das ganze aus der Shell heraus zu starten, gibt es diese Befehle:

```
cd /hp/cv/aa/ok/www/modellbahn-anlage.de/wiki/httpdocs  # Ins Verzeichnis
wechseln
git add .  # Alle Änderungen hinzufügen
git commit -m "Manueller Commit via Shell am $(date '+%Y-%m-%d %H:%M:%S')"
# Commit erstellen
git pull --rebase origin master  # Aktuelle Änderungen von GitHub holen und
mit deinen kombinieren
git push origin master  # Änderungen auf GitHub übertragen
```

5. Cronjob für nächtliche Versionierung

1. Skript ertsellen z.B. /hp/cv/aa/ok/www/modellbahn-anlage.de/wiki/git-auto.sh:

mit dem Inhalt:

```
#!/bin/sh
cd /hp/cv/aa/ok/www/modellbahn-anlage.de/wiki/httpdocs || exit 1

# Nur committen, wenn sich etwas geändert hat
if ! git diff --quiet || ! git diff --cached --quiet; then
  TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')
  git add .
  git commit -m "Auto-Commit am $TIMESTAMP"
  git pull --rebase origin master
  git push origin master
fi
```

2. Ausführbar machen

```
chmod +x /hp/cv/aa/ok/www/modellbahn-anlage.de/wiki/git-auto.sh
```

3. Cronjob setzen (alle 10 Minuten für Tests)

```
*/10 * * * * /hp/cv/aa/ok/www/modellbahn-anlage.de/wiki/git-auto.sh >>
/hp/cv/aa/ok/www/git-auto-commit.log 2>&1
```

4. Cronjob für 1x täglich

```
0 3 * * * /hp/cv/aa/ok/www/modellbahn-anlage.de/wiki/git-auto.sh >>
/hp/cv/aa/ok/www/git-auto-commit.log 2>&1
```

Logile im GIT leeren

Cronjob (z. B. jeden 1. des Monats um 03:15 Uhr)

```
15 3 1 * * > /hp/cv/aa/ok/www/git-auto-commit.log
```

Am 1. jedes Monats um 03:15 Uhr

```
15 3 1 * *
```

leert die Datei einfach (ohne sie zu löschen)

```
>
```

Cache nicht mehr übertragen

```
git rm -r --cached data/cache/
git commit -m "Remove cache from tracking"
```

PHP-Datei zum manuellen ausführen (git-backup.php)

```
<?php
// Lade Umgebungsvariablen aus einer .env-Datei im übergeordneten
Verzeichnis
function loadEnv($path) {
    if (!file_exists($path)) {
        echo ".env-Datei nicht gefunden: $path\n";
        exit(1);
    }

    $lines = file($path, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
    foreach ($lines as $line) {
        if (str_starts_with(trim($line), '#')) continue;
        list($name, $value) = explode('=', $line, 2);
        putenv(trim($name) . '=' . trim($value));
    }
}

// .env liegt im übergeordneten Ordner
loadEnv(__DIR__ . '/../.env');

$repoPath = __DIR__;
$user = getenv('GIT_USER');
$token = getenv('GIT_TOKEN');
$remoteUrl = "https://$user:$token@github.com/<color #fff200>NAME DES
VERZEICHNISES/GIT_NAME</color>.git";
$commitMessage = "Automatisches Backup am " . date('Y-m-d H:i:s');

// Git-Befehle ausführen
function git($command, $cwd) {
    $output = [];
```

```
    exec("cd " . escapeshellarg($cwd) . " && $command 2>&1", $output, $ret);
    if ($ret !== 0) {
        echo "? Fehler bei '$command':\n" . implode("\n", $output) . "\n";
        exit(1);
    }
    return $output;
}

// Git init bei Bedarf
if (!is_dir("$repoPath/.git")) {
    git("git init", $repoPath);
    git("git remote add origin \"$remoteUrl\"", $repoPath);
}

// Benutzerkonfiguration für Git setzen
git("git config user.name 'Dokuwiki Bot'", $repoPath);
git("git config user.email '<color #fff200>E-Mail</color>'", $repoPath);

// Submodule löschen
$rii = new RecursiveIteratorIterator(new
RecursiveDirectoryIterator($repoPath));
foreach ($rii as $file) {
    if ($file->isDir() && $file->getFilename() === '.git' &&
$file->getPath() !== $repoPath) {
        exec("rm -rf " . escapeshellarg($file->getPathname()));
    }
}

// Änderungen vorbereiten
git("git add .", $repoPath);
$status = git("git status --porcelain", $repoPath);

if (empty($status)) {
    echo "?? Keine Änderungen gefunden – nichts zu committen.\n";
    exit;
}

// Commit & Push
git("git commit -m \"" . addslashes($commitMessage) . "\"", $repoPath);
git("git push --set-upstream origin master", $repoPath);  // Hier wird das
Token in der Remote-URL bereits verwendet

echo "? Backup erfolgreich auf GitHub übertragen.\n";
?>
```