Mit Git entwickeln: vom Benutzer zum Entwickler

Der empfohlene Weg, Websites mit Exponential zu bauen. Installieren Sie den Kernel und jede se7enxweb-Erweiterung als Git-Klone mit ihrer vollständigen History, stellen Sie eine bestehende Site auf Klone um, halten Sie…

Source file: `doc/ger-de/hilfe/mit-git-entwickeln.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/develop-with-git.md

Sie kennen sich in der Administration aus, Sie haben Inhalte veröffentlicht und vielleicht ein Template geändert. Diese Seite hilft Ihnen beim nächsten Schritt: am Code Ihrer Site so zu arbeiten, wie es die Menschen tun, die Exponential entwickeln. Das Werkzeug dafür ist **Git**, und die nützlichste Einzelentscheidung, die Sie treffen können, ist, Exponential **als Git-Klone mit ihrer vollständigen History** zu installieren, den Kernel und jede Erweiterung, statt als Release-Archive.

> **Warnung `.htaccess` bremst jede Anfrage.** Sind `.htaccess`-Dateien erlaubt (`AllowOverride` alles außer `None`), geschweige denn genutzt, sucht Apache sie bei jeder Anfrage in jedem Verzeichnis des Pfads und liest und wertet sie aus, und die Ladezeiten der Seiten steigen drastisch (gemessen). Die einzige wirkliche Abhilfe ist, die Funktion für den virtuellen Host abzuschalten: `AllowOverride None` im virtuellen HTTPS-Host, und die Regeln in der Konfiguration des virtuellen Hosts selbst als mod_rewrite-Regeln. Das ist in Geschwindigkeitstests im Produktivbetrieb weit schneller und die Zeit für die Optimierung einer Produktivkonfiguration unbedingt wert. Stellen Sie sofort um: Übernehmen Sie die Regeln des Beispiels für einen virtuellen HTTPS-Host in Ihren virtuellen Host (mit dem statischen Cache auch `staticcache_offload.conf`), setzen Sie `AllowOverride None`, führen Sie `apachectl configtest` aus, laden Sie Apache neu und testen Sie die Website. Dieselben Regeln für andere Server: nginx, LiteSpeed, Caddy, IIS, Velocity (eingebaut); wohin Regeln für virtuelle Hosts in einem Control Panel gehören: Plesk, cPanel, DirectAdmin, ISPConfig. **Unter LiteSpeed oder OpenLiteSpeed** kann zusätzlich der Seiten-Cache des Servers anonyme Besucher und Crawler ohne PHP bedienen: Seiten-Caching mit LiteSpeed und OpenLiteSpeed.

Nichts hier setzt voraus, dass Sie Experte sind. Jeder Schritt zeigt den einzugebenden Befehl und was Sie sehen sollten, und jeder Befehl auf dieser Seite wurde auf einem frischen Rechner ausprobiert, bevor er aufgeschrieben wurde.

Warum Git-Klone und keine Archive

Composer kann ein Paket auf zwei Arten installieren:

	Release-Archiv („dist“)	**Git-Klon („source“)**
Was Sie erhalten	die Dateien einer Version, sonst nichts	die Dateien **und** die vollständige History des Projekts
Größe und Geschwindigkeit	klein, schnell	größer, bei der ersten Installation einige Minuten länger

	Release-Archiv („dist“)	Git-Klon („source“)
Sehen, was Sie geändert haben	`composer status -v` nennt geänderte Dateien	`git status` und `git diff` zeigen jede Änderung, Zeile für Zeile
Herausfinden, warum eine Zeile da ist	nicht möglich	`git log` und `git blame` zeigen, wer sie wann und warum geändert hat
Herausfinden, welche Änderung etwas kaputt gemacht hat	nicht möglich	`git bisect` findet sie in wenigen Schritten
Eine Korrektur vor ihrem Release nutzen	Dateien von Hand kopieren	`git cherry-pick` oder den Commit auschecken
Eigene Änderungen behalten	beim nächsten Update verloren	auf Ihrem eigenen Branch behalten; Composer weigert sich, sie zu überschreiben
Eine Korrektur an das Projekt zurückschicken	nicht möglich	einen Branch in Ihren Fork pushen und einen Pull-Request öffnen

Archive genügen für einen Server, der nur eine fertige Site betreibt. Zum **Bauen** einer Website sind Git-Klone der empfohlene Weg: Sie machen aus jedem Paket etwas, das Sie lesen, durchsuchen, verstehen und verbessern können.

Was Sie brauchen

- **Git** 2.28 oder neuer (`git --version`).
- **Composer** 2 (`composer --version`) und PHP wie unter Voraussetzungen aufgeführt.
- Ein kostenloses **GitHub-Konto**, nur wenn Sie Änderungen zurückschicken möchten. Für die Installation brauchen Sie keines.

Der Kernel ist github.com/se7enxweb/exponential, und jede Erweiterung hat ihr eigenes Repository in der Organisation se7enxweb. Sie sind unter denselben Namen auf Packagist veröffentlicht, sodass Composer jedes davon entweder als Archiv oder als Klon holen kann.

Schritt 1: Composer anweisen, Klone zu bevorzugen

Eine einzige Einstellung bewirkt, dass Composer jedes `se7enxweb/*`-Paket als Git-Klon installiert und alles andere (die Zeta Components, die Bibliotheken) als schnelles Archiv. Setzen Sie sie einmal für Ihr Benutzerkonto:

Shell

```
composer config --global 'preferred-install.se7enxweb/*' source
```

Behalten Sie die Anführungszeichen um `se7enxweb/*`, damit Ihre Shell nicht versucht, das `*` zu expandieren. So sehen Sie die Einstellung:

Shell

```
composer config --global preferred-install
```

```
{"se7enxweb/*":"source","*":"dist"}
```

Möchten Sie die Einstellung lieber bei einem einzelnen Projekt halten (zum Beispiel, damit jeder

in Ihrem Team sie bekommt), führen Sie denselben Befehl ohne `--global` innerhalb des Projekts aus. Er wird dann in die `composer.json` dieses Projekts geschrieben.

Schritt 2: mit Git installieren

Eine neue Site

Shell

```
git clone https://github.com/se7enxweb/exponential.git exponential
cd exponential
git switch -c my-site
composer install
```

- `git clone` holt den Kernel mit seiner ganzen History: mehr als 27.000 Commits, ein Download von etwa 370 MB. Bei einer normalen Verbindung dauert das zwei bis drei Minuten.
- `git switch -c my-site` beginnt **Ihren eigenen Branch** für die Änderungen Ihrer Site, auf Basis von `main`, der aktuellen Entwicklungslinie. Um stattdessen auf einem Release aufzubauen, nennen Sie dessen Tag: `git switch -c my-site v6.0.14`. Die Release-Tags listet `git tag -l 'v*' --sort=version:refname` auf; lesen Sie Versionen, Tags und Branches, bevor Sie eines wählen, denn ältere Tags brauchen für ihre Testwerkzeuge ein neueres PHP.
- `composer install` klont danach jede Erweiterung, die der Kernel verlangt, nach `extension/<name>`, jede mit ihrer eigenen History (auf `main` sind das 67 Repositories, einige Minuten), und holt die Bibliotheken als Archive nach `vendor/`.

Installieren Sie die Site dann wie gewohnt: Eine Installationsmethode wählen oder der eine Befehl `php bin/php/console exp:install --apply` aus der Konsoleninstallation.

Für eine lokale Entwicklungs-Site unter Apache bringen Sie die Regeln und die Konstanten von der Konsole aus an ihren Platz. Eine Kopie von `config.php-RECOMMENDED` setzt nichts, es gelten also die eingebauten Vorgaben, und die passen zur Entwicklung (Änderungen an INI-Dateien werden sofort gesehen); `exp:config:list` zeigt, was sich setzen ließe:

Shell

```
php bin/php/console exp:setup:status
php bin/php/console exp:setup:htaccess --apply
php bin/php/console exp:setup:config --apply
php bin/php/console exp:config:list
```

Der Klon verfolgt nur die Vorlagen (`.htaccess_root`, `.htaccess_root_static`, `config.php-RECOMMENDED`); die Kopien erscheinen in `git status` als nicht verfolgte Dateien. Committen Sie `config.php` nie: Sie kann ein Passwort enthalten. Tragen Sie in einem Repository Ihrer eigenen Site `/config.php` in dessen `.gitignore` ein. Einzelheiten: .htaccess und config.php von der Konsole aus einrichten.

Lieber `composer create-project`? Auch das kann Klone erzeugen, aber Sie brauchen zwei Optionen zusammen:

Shell

```
composer create-project se7enxweb/exponential exponential --prefer-source --keep-vcs
```

Ohne `--keep-vcs` löscht Composer am Ende den `.git`-Ordner des Kernels, und Sie stehen ohne History da. Ohne Angabe einer Version wählt Composer das neueste Release, sodass der Kernel dann ein Klon ist, der auf dem Tag dieses Releases steht („HEAD detached at v6.0.14“); führen Sie darin `git switch -c my-site` aus, bevor Sie etwas ändern. Hier macht `--prefer-source` jedes Paket zum Klon, die Bibliotheken eingeschlossen; mit der Einstellung aus Schritt 1 können Sie es weglassen und erhalten Klone nur für die `se7enxweb/*`-Pakete.

Prüfen, ob es funktioniert hat

Shell

```
composer status -v
ls -d extension/*/.git | wc -l
git -C extension/eztags log --oneline -3
git -C extension/eztags status
```

`composer status -v` sollte `No local changes` melden, der zweite Befehl zählt die Erweiterungen, die Klone sind, und die letzten beiden zeigen die neuesten Commits einer Erweiterung und wo sie steht:

```
HEAD detached at v2.4.21
nothing to commit, working tree clean
```

„HEAD detached“ ist normal. Composer checkt genau das Release-Tag aus, das Ihre `composer.json` verlangt, keinen Branch. In diesem Zustand können Sie beliebig lesen, suchen und testen. Wenn Sie etwas ändern möchten, beginnen Sie zuerst einen Branch (siehe Eine Änderung per Pull-Request beitragen).

`git status` im Wurzelverzeichnis der Installation listet die Erweiterungen als nicht verfolgte Ordner auf: Sie sind eigene Repositories, nicht Teil des Kernel-Repositorys, und das ist so vorgesehen.

Schritt 3: eine bestehende Site auf Klone umstellen

Eine aus Archiven installierte Site lässt sich umstellen, ohne sie neu zu installieren. **Erstellen Sie zuerst ein Backup** (mit dem Backup-Manager von git_manager oder einer Kopie des Ordners und der Datenbank), und probieren Sie es an einer Kopie der Site aus, bevor Sie die Live-Site umstellen.

Der Kernel

Stellen Sie fest, welche Version Sie betreiben, und verbinden Sie dann den Ordner genau bei dieser Version mit dem Repository des Kernels:

Shell

```
php bin/php/console --version
git init -b my-site
git remote add origin https://github.com/se7enxweb/exponential.git
git fetch origin --tags
git reset v6.0.14
git status
```

Verwenden Sie Ihre eigene Version anstelle von `v6.0.14`; eine von `main` installierte Site verwendet `git reset origin/main`. `git reset` ohne `--hard` verschiebt nur den Zeiger von Git: **Es ändert keine einzige Datei.** `git status` zeigt dann genau, was vom Release

abweicht, bei einer aus einem Archiv installierten Site also Ihre eigenen Änderungen und die Dateien, die Ihre Installation angelegt hat (etwa `settings/override/`; der größte Teil von `var/` wird ignoriert):

```
On branch my-site
 M design/standard/templates/pagelayout.tpl
?? settings/override/
```

Committen Sie die Änderungen, die Sie behalten möchten, auf `my-site`. Die Dateien in `settings/override/` und `settings/siteaccess/` enthalten Datenbankpasswörter: Pushen Sie sie nie in ein öffentliches Repository.

Die Erweiterungen

Shell

```
composer config --global 'preferred-install.se7enxweb/*' source
composer status -v
composer reinstall 'se7enxweb/*'
composer status -v
```

`composer reinstall` entfernt jedes `se7enxweb/*`-Paket und installiert es in derselben Version erneut als Klon.

> **Warnung:** `composer reinstall` löscht zuerst den Ordner des Pakets. Listet das erste `composer status -v` geänderte Dateien in einer Erweiterung auf, die noch ein Archiv ist, kopieren Sie diesen Ordner vor der Neuinstallation an einen sicheren Ort, vergleichen Sie ihn danach mit dem Klon (`diff -ru <copy> extension/<name>`) und übernehmen Sie Ihre Änderungen auf einem Branch.

Erzeugen Sie nach der Neuinstallation die Klassen-Maps neu und leeren Sie die Caches wie nach jeder Codeänderung:

Shell

```
php bin/php/ezpgenerateautoloads.php -e
php bin/php/ezcache.php --clear-all
```

Die Administration kann dasselbe ohne Shell: Mit der Erweiterung ezupdate (**Setup** > **Aktualisierungen und Pakete**) wählen Sie für jeden Installations- oder Update-Lauf „dist oder source“.

Eine Änderung per Pull-Request beitragen

Einen Fehler gefunden und behoben? Schicken Sie die Korrektur zurück, damit sie im nächsten Release enthalten ist und Sie sie nicht selbst mitschleppen müssen. Das funktioniert für den Kernel und für jede Erweiterung gleich; das Beispiel verwendet `eztags`.

1. Öffnen Sie auf GitHub das Repository (github.com/se7enxweb/eztags) und klicken Sie auf **Fork**. Sie haben nun Ihre eigene Kopie, `github.com/<you>/eztags`.
2. Fügen Sie Ihren Fork als zweites Remote hinzu und beginnen Sie einen Branch:

Shell

```
cd extension/eztags
git remote add fork https://github.com/<you>/eztags.git
git switch -c fix-tag-search
```

3. Nehmen Sie die Änderung vor, testen Sie sie auf Ihrer Site und committen Sie sie. Beginnen Sie die Nachricht mit dem, was die Änderung tut: `Fixed:`, `Added:`, `Updated:`, `Removed:` oder `Renamed:`.

   Shell

   ```
   git commit -am "Fixed: The tag search finds tags with an apostrophe"
   git push -u fork fix-tag-search
   ```

4. GitHub gibt einen Link zum Öffnen eines **Pull-Requests** aus; folgen Sie ihm, beschreiben Sie, was Sie behoben und wie Sie es getestet haben, und schicken Sie ihn ab. (Mit dem Kommandozeilenwerkzeug von GitHub: `gh pr create`.)

Ein von Composer erstellter Klon hat bereits ein Remote namens `origin`, das auf das Projekt zeigt; Pushes gehen stattdessen in Ihren Fork, sodass Sie keinen Schreibzugriff auf das Projekt brauchen. Lesen Sie für den Kernel selbst zuerst `CONTRIBUTING.md` im Wurzelverzeichnis der Installation: Es erklärt den Commit-Hook (`make hooks`) und das Dateimanifest (`make manifest-fix`), die jeder Pull-Request für den Kernel aktuell hält.

Rechnen Sie mit vielen Prüfrunden. 7x prüft einen Commit oder Pull-Request in viel mehr Runden als üblich, manchmal in Hunderten von Überarbeitungen; seien Sie bereit, Ihre Änderung zu überarbeiten, bis sie sich einfach übernehmen lässt. Eine Korrektur für Verhalten, das sich als beabsichtigt herausstellt, eine negative Funktion, auf die sich andere Sites verlassen, braucht zusätzlich eine klare Begründung, und eine Option lässt sich leichter übernehmen als ein neuer Standard. Solange der Pull-Request offen ist, fassen Sie ihn zusammen (squash) und setzen ihn mit rebase auf `main` neu auf, wie die Prüfungen es verlangen; 7x setzt das voraus:

Shell

```
git fetch origin
git rebase -i origin/main
git push --force-with-lease
```

Pushen Sie mit Force nur in den Branch Ihres eigenen Pull-Requests. Nach dem Merge wird `main` nie umgeschrieben (Code reviews).

Übertragung des Urheberrechts (CLA). 7x und die Exponential Foundation übernehmen einen Code-Beitrag zum Kernel (`se7enxweb/exponential`), gleich in welcher Form er kommt (Pull-Request, Patch, Issue oder Mail), erst, nachdem Sie das Contributor Licensing Agreement unterschrieben haben. **Mit Unterschrift und Einreichung übertragen Sie das Urheberrecht an Ihrem Beitrag auf 7x, das sein Eigentümer wird, und geben die Kontrolle darüber auf:** 7x darf ihn auf jede Weise verwenden, ändern, neu lizenzieren und verbreiten, auch unter anderen Lizenzen und in kommerziellen Produkten, ohne Sie zu fragen oder zu bezahlen, und Sie können ihn weder zurückziehen noch Bedingungen daran knüpfen. Ihre eigenen Erweiterungen und der Branch Ihrer eigenen Site sind davon nicht betroffen. Siehe Copyright assignment (CLA) und die Seite zum CLA.

Eine eigene Erweiterung gebaut? Erweiterungen zurückgeben zeigt, wie Sie sie auf GitHub und Packagist veröffentlichen, damit jede Site sie installieren kann, und was die GPL verlangt, wenn

Sie eine Erweiterung an andere weitergeben.

Die Dokumentation zu verbessern funktioniert genauso: Diese Hilfeseiten sind die Dateien in `doc/` des Kernels. Was Sie damit tun dürfen, steht unter Freie Dokumentation.

Tägliche Arbeit mit Klonen

Sehen, was sich geändert hat

Shell

```
git status
git diff
composer status -v
```

`composer status -v` schaut in jedes Paket auf einmal und nennt diejenigen mit nicht committeten Änderungen, nicht gepushten Commits oder einer anderen Version als in der Lock-Datei.

Herausfinden, warum und wann

Shell

```
git log --oneline -10 -- kernel/content/view.php
git blame kernel/content/view.php
git log -S "CheckValidity" --oneline
```

Der erste Befehl zeigt die letzten zehn Änderungen an einer Datei, der zweite, wer jede Zeile zuletzt geändert hat, der dritte jeden Commit, der ein Stück Text hinzugefügt oder entfernt hat. Führen Sie in einer Erweiterung dieselben Befehle innerhalb von `extension/<name>` oder mit `git -C extension/<name> ...` aus.

Die Änderung finden, die etwas kaputt gemacht hat

Shell

```
git bisect start
git bisect bad
git bisect good v6.0.13
```

Git checkt eine Version in der Mitte aus; testen Sie sie und antworten Sie mit `git bisect good` oder `git bisect bad`, bis Git den Commit nennt. `git bisect reset` bringt Sie dorthin zurück, wo Sie begonnen haben.

Eine Korrektur vor ihrem Release nutzen

Wenn eine Korrektur schon in der History des Projekts steht, aber noch in keinem Release:

Shell

```
git -C extension/eztags fetch origin
git -C extension/eztags switch -c with-fix
git -C extension/eztags cherry-pick <commit>
```

Solange dieser Branch nur auf Ihrem Rechner existiert, weigert sich Composer, das Paket zu aktualisieren (siehe Wenn Composer sich weigert): Pushen Sie den Branch in Ihren Fork, oder kehren Sie mit `git -C extension/eztags switch --detach v2.4.21` (Ihr Release-Tag) zum

Release zurück, sobald ein Release die Korrektur enthält.

Auf dem neuesten Stand bleiben

Zuerst der Kernel auf Ihrem Branch, dann die passenden Erweiterungen:

Shell

```
git fetch origin --tags
git merge origin/main
composer update
```

(oder mergen Sie statt `origin/main` das neueste Release-Tag). Das Mergen behält Ihre eigenen Commits auf `my-site`; `composer update` und nicht `composer install`, weil sich die `composer.json` des Kernels geändert hat und Ihre Lock-Datei ihr folgen muss. Immer zuerst Git (der Kernel), dann Composer (die Erweiterungen). Mit Git aktualisieren beschreibt die ganze Routine, auf der Kommandozeile (Weg A) oder in der Administration mit Git Manager und „Aktualisierungen und Pakete“ (Weg B), für den Rhythmus Ihrer Wahl (stündlich, täglich, wöchentlich, monatlich oder nur bei jedem Release), wie Sie sehen, was sich geändert hat, und ein automatisches Update. Den Code nachzuziehen ist die eine Hälfte eines Upgrades; Upgrade beschreibt die andere, die Datenbank.

Eine einzelne Erweiterung, nachdem Sie ihre Release-Notes gelesen haben:

Shell

```
composer update se7enxweb/eztags
```

Composer setzt den Klon auf das neue Tag. Seine History bleibt erhalten, ebenso Ihre Branches.

Wenn Composer sich weigert

Composer schützt Ihre Arbeit in einem Klon. Das sind die Meldungen, denen Sie begegnen werden, und was zu tun ist:

Composer meldet	Warum	Was zu tun ist
`Source directory ./extension/eztags has uncommitted changes.`	Sie haben Dateien geändert und nicht committet	committen Sie sie auf einem Branch, oder `git -C extension/eztags stash`, führen Sie das Update aus, dann `git -C extension/eztags stash pop`
`Source directory ./extension/eztags has unpushed changes on the current branch`	Sie haben auf einem Branch committet, der nur auf Ihrem Rechner existiert	`git push -u fork <branch>` in Ihren Fork, dann aktualisieren; Ihr Branch bleibt erhalten
`You have version variations in the following dependencies` (von `composer status`)	der Klon steht auf einem Branch, nicht auf dem Tag aus der Lock-Datei	erwartet, solange Sie an einer Korrektur arbeiten; `composer update <package>` setzt ihn zurück auf das Tag
das Paket ist noch aus einem Archiv installiert	es wurde vor der Einstellung aus Schritt 1 installiert: Composer wandelt bereits installierte Pakete nicht um	`composer reinstall <package>` (siehe Schritt 3)

Drei weitere wissenswerte Dinge:

- **Klonen Sie nie mit `git clone` von Hand in einen Ordner, den Composer als Archiv installiert hat.** Composer hält den Ordner weiterhin für ein Archiv und ersetzt ihn beim nächsten Update dieses Pakets ohne Rückfrage: Ihr Klon, seine History und Ihre Commits sind weg. Lassen Sie Composer den Klon mit `composer reinstall <package>` erstellen (siehe Schritt 3).
- Ein Update kann auf halbem Weg stehen bleiben, wenn es sich weigert: Die Lock-Datei nennt dann womöglich schon die neue Version, während der Ordner noch die alte enthält. Beheben Sie die Ursache und führen Sie dann `composer install` aus, um es abzuschließen.
- `composer update` setzt **jedes** Paket auf die neueste Version, die seine Einschränkung erlaubt. Aktualisieren Sie auf einer Live-Site ein Paket nach dem anderen (`composer update se7enxweb/<package>`), wie Den Code beziehen erklärt.

Wie es weitergeht

- Mit Git aktualisieren: im Rhythmus Ihrer Wahl mit GitHub synchron bleiben.
- Den Code beziehen: jeder Installationsweg, Versionen und Tags, Erweiterungen als Composer-Pakete.
- Erweiterungen und die Seiten zu den Erweiterungen.
- .htaccess und config.php von der Konsole aus einrichten: die lokale `.htaccess` und `config.php`.
- git_manager: den Zustand eines Git-Checkouts in der Administration ansehen und ändern und Backups erstellen, bevor Sie etwas ändern.
- Wenn etwas schiefgeht: welcher Issue-Tracker zu verwenden ist und wie Sie einen guten Bericht schreiben. Die Seite **Fehler melden** dieser Hilfe bereitet das Issue für Sie vor.
- Das Projekt auf GitHub: se7enxweb/exponential, seine Issues und jedes se7enxweb-Repository.
- Freie Dokumentation: Diese Seiten dürfen frei kopiert, übersetzt und verbessert werden.
- Git selbst: das freie Buch Pro Git und die Anleitung von GitHub zu Pull-Requests.

Siehe auch: Ökosystem, PHP-Entwickler, KI, Agenten und agentisch, Software-Ingenieure, Web-Framework-Entwickler, CMS-Entwickler, DXP-Entwickler, Integratoren.

Verwandte Seiten

- Migrationen: Teamarbeit mit Git, Migrationen

> **Werden Sie Exponential-Partner**
>
> Partner teilen ihre Informationen, stellen Mittel bereit, geben Beiträge zurück und entwickeln die Produkte mit uns, und so wächst Exponential. Eine Partner-Website und ein Dashboard sind für Mitte 2027 geplant: Werden Sie noch heute Partner, damit wir schneller wachsen und neue Funktionen früher liefern.
>
> Partner: hier beginnen | Wie Sie sich bewerben

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.