Update-Schritte

Schilderung typischer to-dos und Abläufe für Moodle-Updates

Moodle-Updates gliedern sich in folgende typischen Phasen, wobei auf dieser Wiki-Seite vor allem technische Themen dokumentiert werden sollen:

1. Sichten neuer Funktionen und UI-Änderungen
2. Installieren und Konfigurieren auf Testsystem (ggf. mehrfach)
3. Testing auf Testsystem
4. Dokumentieren der Neuerungen
5. Update-Ankündigung
6. Generalprobe auf Testsystem
7. Update-Durchführung Produktiv-System inkl. DB-Backup + Frontend-config
8. Troubleshooting

1. Sichten neuer Funktionen und UI-Änderungen

Es lassen sich drei Kategorien von Änderungen benennen: Konfiguration, Funktionen/Features, User Interface/Layout. Als erstes sollte die Moodle-Dokumentation sorgfältig gelesen werden, zuvorderst die Docs-Seite zu Upgrading und dort der Bereich "Possible issues that may affect you in Moodle", z.B. für Moodle 5.1:

https://docs.moodle.org/501/en/Upgrading#Possible_issues_that_may_affect_you_in_Moodle_5.1

Die neuen Funktionen werden auf der Release-Seite kurz benannt mit Links zu den entsprechenden Tracker-Einträgen, wo mehr Details und meist Screenshots zu finden sind. Die New Features Seite geht auf die (aus Sicht des Headquaters) wichtigsten Änderungen ein. Diese werden meist auch noch in einem Screencast per YouTube vorgestellt (Playlist oben auf New Features Seite verlinkt). Änderungen am Layout und der Umgang damit werden im Bereich Moodle Testing näher beschrieben.

Liste aller Neuerungen (Beispiel Moodle 5.1): https://moodledev.io/general/releases/5.1

Neue Funktionalitäten / New Features: https://docs.moodle.org/de/Neue_Funktionalit%C3%A4ten / https://docs.moodle.org/en/New_features

YouTube-Videos Moodle: https://www.youtube.com/@moodle/playlists (Videos zu Moodle 5.1)

Dag Klimas macht auch oft Videos zu den neuen Features, z.B. Moodle 5.1 - Was hat sich geändert, was ist neu?

2. Installieren und Konfigurieren auf Testsystem

Die Installation wird nach bevorzugter Variante der Administration durchgeführt, z.B. mit Hilfe von Ansible-Playbooks. Dies baut aus verschiedenen Quellen das angepasste Installationspaket. Neben den Moodle core-Dateien sind dies z.B. Moodle Plugins aus github oder modifzierte Plugins aus interner Quelle sowie die config.php Dateien. Das Testsystem (staging) sollte eine möglichst mit dem live-System übereinstimmende Kopie sein, mit gleichen Server-Einstellungen (inkl. php.ini) und Moodle-Konfiguration. Davon ausgehend wird zur Vorbereitung des live-Updates der komplette Update-Prozess auf staging durchlaufen (siehe 6.), meist auch mehrfach, bis zu einer fehlerfrei laufenden Version.
Praxis-Tipp: Beim installieren auffallende Probleme und nötige Anpassungen wie Konfigurationen sollten gut auffindbar und nachvollziehbar dokumentiert werden. Hier kann auch eine Art Vorlage hilfreich sein.

3. Testing auf staging

Die Details sind näher beschrieben auf der Seite Moodle Testing. Ziel ist ein fehlerfrei laufendes System, erfahrungsgemäß treten am ehesten Fehler in nicht-Core Komponenten sowie im Layout auf. Als Nebeneffekt des Testings werden dabei die Änderungen der neuen Version kennengelernt.

4. Dokumentieren der Neuerungen

Als Übersicht für alle user sollte für major-Releases ein Neuerungen-Dokument erstellt werden oder auf andere Weise über die Änderungen informiert werden. Auch angepasste user-Tours können hierbei unterstützen.

5. Update-Ankündigung

Updates sollten mit ausreichend Vorlauf angekündigt werden, sodass die user sich auf die maximale Ausfallzeit einstellen können, mögliche Info-Kanäle:

- Moodle-Startseite, z.B. mit Boost Union via Banner
- Websites Störungs- und Wartungsmeldungen (z.B. Rechenzentrum)
- Foren / Lehrenden-Foren
- Info-Chats

6. Generalprobe auf Testsystem

Aufs staging-System wird eine Kopie der live-Datenbank eingespielt und sichergestellt, dass die Versions-Dateien auf den Webservern dazu passen. Dann werdendie neuen Dateien eingespielt und der komplette Update-Prozess simuliert.

7. Update-Durchführung Produktiv-System inkl. DB-Backup + Frontend-config

Zuerst ist ein Datenbank-Backup zu erstellen, z.B. mittels pgadmin. Jetzt wird es ernst: Wartungsmodus per CLI aktivieren: `sudo -u www_mdl /usr/bin/php admin/cli/maintenance.php –enable`

Zugriff für Administratoren während Wartungszeit:

Der Wartungsmodus muss per Frontend gestartet werden (`/admin/settings.php?section=maintenancemode`) oder via `/maintenance.php --enableold`, dann können admin-Accounts Moodle weiter nutzen oder sich auch neu einloggen via /login/index.php?username=[Anmeldename]
Wird `/cli/maintenance.php` aufgerufen, ist die Site auch für admins gesperrt.

Dann wird das live-System mittels vorbereitetem Job aktualisiert. Upgrade per CLI oder im Frontend, dann ist der Aktualisierungsschlüssel einzugeben und der Update-Fortschritt wird verfolgt. Wichtig: auf live dauert es meist deutlich länger als auf den Testsystemen (trotz DB-Kopie), z.B. weil noch mehr Daten im Cache sind. Ist das Update durchgelaufen, sind die vorher definierten Konfigurationsanpassungen im admin-Frontend einzustellen. Zum Abschluss wird der Wartungsmodus wieder deaktiviert (`maintenance.php –disable`).

Wenn gepflegt, kann noch ein "to-dos nach dem Update" abgearbeitet werden. Zudem sollten minor- und Major-Updates in einem Changelog dokumentiert werden, dazu auch wenn neue Plugins hinzu kamen.

8. Troubleshooting (bei Bedarf)

Sollte etwas nicht wie geplant und im Vorfeld getestet funktionieren: don´t panic! In diesem Fall sind die Debbuging-Methoden anzuwenden. Zudem wie immer auch die debugging- und logging-Meldungen sichten. im Normalfall fallen die Probleme bereits im Testing auf, weshalb ein sorgfältiges Testen mit einer möglichst identischen live-System-Kopie wichtig ist.
Hier ein Beispiel.

Aktivieren Debug-Modus mittels config.php

```
//=============================================================================
// 7. SETTINGS FOR DEVELOPMENT SERVERS - not intended for production use!!!
//=============================================================================
```

//
// Force a debugging mode regardless the settings in the site administration
```
@error_reporting(E_ALL | E_STRICT);   // NOT FOR PRODUCTION SERVERS!
@ini_set('display_errors', '1');         // NOT FOR PRODUCTION SERVERS!
$CFG->debug = (E_ALL | E_STRICT);   // === DEBUG_DEVELOPER - NOT FOR PRODUCTION SERVERS!
$CFG->debugdisplay = 1;              // NOT FOR PRODUCTION SERVERS!
```

Hinweis: `E_STRICT` ist ab PHP 8.4 deprecated und wird dereinst nicht mehr setzbar sein.

SQL-Abfragen anzeigen:
$CFG->upgradeshowsql = true;

Dadurch werden alle SQL-Befehle während des Upgrades ausgegeben und wie lange sie geladen haben ("Query took"), das sieht dann in der Konsole z.B. so aus:

```
--------------------------------
DECLARE crs7 NO SCROLL CURSOR WITH HOLD FOR SELECT * FROM mdl_capabilities
[array (
)]
--------------------------------
Query took: 0.00043296813964844 seconds.
--------------------------------
```

Links / Quellen

https://docs.moodle.org/de/Aktualisierung_von_Moodle

Autor: Klaus Steitz, TU Darmstadt

Version #10
Erstellt: 2025-12-01 13:13:24 UTC von Klaus Steitz
Zuletzt aktualisiert: 2026-06-05 13:17:53 UTC von Luca Bösch