Betriebsdokumentation

Experimenteverwaltung - Erweiterung Buchungssystem (I2)

Richard Böhme

2021-07-01 11:58:18 +0200

Inhaltsverzeichnis

1. Einführung

Das Experimente Management System wird verwendet, um Physik Experimente zu verwalten und zu buchen. Es handelt sich bei der Anwendung um einen klassischen Rails Monolithen. Die nachfolgende Dokumentation soll der Administration der Anwendung dienen.

2. Systeme

Die Anwendung besitzt ein Test- und ein Produktivsystem, welche beide auf je einer VM laufen. Die virtuellen Maschinen sind von der Fakultät Maschinenbau bereitgestellt und werden von Herr Hornoff betreut.

Tabelle 1. Systemübersicht

Stage	**IP**
Testsystem	test.system.ip
Produktivsystem	prod.system.ip

Da das Testsystem ein Klon des ehemaligen Produktivsystems ist, ist der Zugang für beide Systeme identisch. Verbunden werden kann sich mit dem Nutzer *user* und dem Passwort *passwort*.

3. Systemvoraussetzungen

3.1. Mindestanforderungen an der Hardware

Systemvoraussetzungen für eine Rails Anwendung sind abhängig von der Anwendung zu wählen. Im Falle des Experimente Management Systems, bei welchem es sich um eine relativ kleine Anwendung handelt, sollten folgende Voraussetzungen ausreichend sein:

- mindestens 2 GB RAM
- mindestens 2 vCPU-Kerne
- mindestens 50 GB Speicherplatz*

(*) *Speicherplatz sollte erhöht werden, desto länger das System online ist. Mediendateien (wie Bilder oder Videos) sowie die Datenbank liegen im Dateispeicher. Empfohlen wird deshalb mehr Speicherplatz zu wählen.*

Hierbei handelt es sich jedoch um Mindestanforderungen. Sollten deutlich mehr Benutzer das System nutzen, kann es zu langsamen Antwortzeiten o.Ä. kommen. Man sollte dann die Hardware der virtuellen Maschine erweiteren. Mit Befehlen wie `top` (bzw. `htop`) und `df -h` kann geprüft werden, wo der Flaschenhals des Systems ist.

3.2. Softwareanforderungen

Es sollte ein debianbasierendes Betriebssystem wie Ubuntu 18.04 oder 20.04 verwendet werden.

Um die Ruby Versions verwalten zu können sollte die aktuellste Version des Ruby Version Managers (RVM) installiert werden. Eine Installationsanleitung ist hier zu finden. Danach sollte mittels `rvm install 3.0.1` Ruby 3.0.1 installiert werden.

Als Web- und Anwendungsserver wird NGINX mit Passenger verwendet. Eine Installationsanleitung sowie eine Dokumentation der Konfiguration ist hier zu finden.

Um eine verschlüsselte Verbindung via SSL zu ermöglichen wird Let's Encrypt verwendet. Die dazugehörige Software `certbot` wird benötigt um die Zertifikate zu verwalten und automatisch zu verlängern. Eine Anleitung dazu ist hier zu finden.

Folgende Bibliotheken müssen zudem installiert werden, um einen reibungslosen Betrieb der Anwendung zu garantieren:

- build-essential
- libssl-dev
- libyaml-dev
- libreadline-dev
- openssl
- curl

- git-core
- zlib1g-dev
- bison
- libxml2-dev
- libxslt1-dev
- libcurl4-openssl-dev
- nodejs
- libsqlite3-dev
- sqlite3
- imagemagick
- sendmail

4. Einrichtung des Systems

Die Anwendung wird über ein Tool namens Capistrano deployt. Dabei wird der komplette Anwendungscode in einer gewissen Verzeichnisstruktur auf dem Server abgelegt. Sobald das geschehen ist, wird der Anwendungsserver (in diesem Fall Passenger) neu gestartet. Wurde NGINX korrekt konfiguriert kann dann die aktualisierte Anwendung aufgerufen werden.

4.1. Verzeichnisstruktur

Die Anwendung wird unter `/var/www/app` installiert. In diesem Verzeichnis werden folgende Verzeichnisse von Capistrano erzeugt und verwaltet:

- *releases* - Enthält den Anwendungscode der letzten fünf Installationen auf dem System (inklusive der aktuell Laufenden).
- *current* - Ist ein symbolischer Link auf das letzte und somit aktuelle Release im `releases` Ordner.
- *shared* - Enthält Dateien die sich über alle Installationen hinweg nicht ändern sollen. Dazu gehören Backups (siehe Backups), die Datenbank, Log-Dateien und Mediendateien (hochgeladene Bilder und Videos).

Innerhalb eines Releases im `releases` Ordner (oder wenn man in `current` navigiert) befindet sich die normale Verzeichnisstruktur von Rails. Der Anwendungscode besteht hauptsächlich aus Ruby- und HTML-Dateien (bzw. Slim-Dateien). Die Assets (Javascript, CSS und Bilder) werden bei der Installation automatisch minifiziert und im Verzeichnis `/var/www/app/current/public/assets` abgelegt.

4.2. Konfiguration von Rails

Rails kann umfassend konfiguriert werden. Alle Konfigurationsdaten befinden sich im Ordner `config` einer aktuellen Installation (z.B. in `/var/www/app/current`). Folgende Dateien sind besonders relevant:

- *environments/production.rb*: Hier werden alle generellen Einstellungen der Rails Anwendung verwaltet. Die Dokumentation dazu findet man hier.
- *database.yml*: Enthält die Zugangsdaten und den Dateipfad zur SQLite Datenbank die die Anwendung nutzt.
- *roles.yml*: Enthält die Zugangsrechte pro Rolle der Benutzer.

Wichtig ist bei Änderungen jedoch, dass diese ggf. überschrieben werden, sobald ein Entwickler eine neue Installation vornimmt. Es sollte also jede Änderung mit einem Entwickler abgesprochen werden.

Eventuell muss die Anwendung neu gestartet werden, damit Änderungen an der Konfiguration einen Effekt zeigen.

4.3. Anwendung neu starten

Die Anwendung kann mit folgendem Befehl neugestartet werden:

```
touch /var/www/app/current/tmp/restart.txt
```

Der Webserver NGINX kann mit folgendem Befehl neugestartet werden:

```
sudo service nginx restart
```

4.4. Konfiguration von Anwendungs- und Webserver

Um Passenger gemeinsam mit NGINX zu verwenden, stellt Passenger umfassende Installations- und Konfigurationsdokumentation bereit, welche hier aufgerufen werden können.

Zudem muss Passenger noch so konfiguriert werden, dass es RVM als Ruby Versions Manager nutzt. Eine Anleitung dazu kann hier gefunden werden.

5. Systembetreuung

5.1. Backups

Jeden Tag um 5 Uhr wird ein Backup der aktuellen Datenbank und der Mediendateien vorgenommen. Der Crontab dazu sollte folgendermaßen eingestellt werden:

```
0 5 * * * /var/www/app/current/bin/backup
```

Um ein altes Backup einzuspielen, sollten folgende Schritte vorgenommen werden:

- Anwendung stoppen:

```
sudo service nginx stop
```

- Backup entpacken:

```
cd /var/app/shared/backups
unzip <Dateiname>
```

- Alten Stand der Datenbank nutzen:

```
cp production.sqlite3 /var/www/app/shared/db
```

- Aktuelle Mediendateien sichern:

```
mv /var/www/app/shared/public/uploads /var/www/app/shared/public/uploads-bak
```

- Alte Mediendateien nutzen:

```
cp -R uploads /var/www/app/shared/public
```

- Anwendung starten:

```
sudo service nginx start
```

- Änderungen verifizieren
- Entpackte Dateien entfernen

```
cd /var/www/shared/backups
rm backup_readme.md
rm production.sqlite3
rm -r uploads
```

- ggf. gesicherte Mediendateien entfernen:

```
cd /var/www/app/shared/public
rm -r uploads-bak
```

5.2. Fehlerdiagnose

Im Folgenden sind Log-Dateien aufgeführt, welche in einem Fehlerfall hilfreich sind. Aufgrund der Komplexität der Anwendung kann nicht jeder auftretbarer Fehler dokumentiert werden. Häufig ist es hilfreich nach Fehlernachrichten oder Fehlercodes im Internet zu recherchieren.

5.2.1. Passenger und NGINX Logs überprüfen

Passenger loggt alle Fehler in das Standard-NGINX Fehler-Log. Dieses liegt im Normalfall in `/var/log/nginx/error.log`. Dabei handelt es sich meistens um Fehler beim Starten der Anwendung. Gegebenenfalls ist dieser Fehler schon bekannt und Passenger oder NGINX müssen nur aktualisiert werden um den Fehler zu verhindern.

5.2.2. Logs der Anwendung

Auch während des Betriebs der Anwendung können Fehler auftauchen. Rails loggt alle Requests und auch Fehler in `/var/www/app/current/log/production.log`.

5.3. FAQ

5.3.1. Wie kann Ruby oder Rails aktualisiert werden?

Ruby sowie Rails folgen beide dem Semantic Versioning. Kommt eine neue Minor-Version heraus, ist ein Update meist ohne Hilfe eines Entwicklers möglich. Trotzdem sollte ein Entwickler baldmöglichst informiert werden, um zu verhindern, dass bei einer neuen Installation die Version zurückgesetzt wird.

Größere Versionssprünge sollten nur durch Entwickler vorgenommen werden, da ggf. Änderungen an der Anwendung vorzunehmen sind.

Changelogs:

- Rails
- Ruby

Ruby

Um Ruby zu aktualisieren müssen folgende Befehle ausgeführt werden:

```
rvm install <version>
rvm default <version>
```

Danach muss in der Datei `/var/www/app/current/Gemfile` in der Zeile:

```
ruby '3.0.1'
```

die Version angepasst werden. Nun muss im selben Verzeichnis der Befehl

```
bundle install
```

ausgeführt werden. Danach muss die Anwendung neu gestartet werden.

Rails

Um Rails zu aktualisieren muss die Datei `Gemfile` im Verzeichnis `/var/www/app/current` bearbeitet werden. Dabei sollte in der Zeile:

```
gem 'rails', '~> 6.1.3.1'
```

die Version aktualisiert werden. Nun muss im selben Verzeichnis der Befehl

```
bundle install
```

ausgeführt werden. Danach muss die Anwendung neu gestartet werden.

5.3.2. Wie können neue Studiengänge in die Datenbank eingetragen werden?

Generell können Änderungen in der Datenbank direkt über das Command-Line-Tool von SQLite vorgenommen werden.

Dafür navigiert man in das Verzeichnis `/var/www/app/shared/db` und führt folgenden Befehl aus:

```
sqlite3 production.sqlite3
```

Es öffnet sich eine separate Konsole von SQLite. Hier können alle möglichen Operationen mit der Datenbank vorgenommen werden. Die Dokumentation von SQLite findet man hier.

Es sollte beachtet werden, dass damit direkt Änderungen an der Produktivdatenbank vorgenom-

men werden. Gegebenenfalls sollte man Änderungen erst auf dem Testsystem testen. Falls trotzdem ein Fehler passiert, kann mit Hilfe der Backups der alte Stand der Datenbank wieder hergestellt werden.

Um einen neuen Studiengang hinzuzufügen muss folgender Befehl in der Kommandozeile von SQLite ausgeführt werden:

```
INSERT INTO courses (name, created_at, updated_at) VALUES ('Name', datetime('now'),
datetime('now'));
```

Wobei `'Name'` mit dem jeweiligen Namen des Studiengangs ersetzt werden sollte. Danach kann mittels

```
SELECT * FROM courses;
```

geprüft werden, dass des Einfügen des Studienganges funktioniert hat.

5.3.3. Wie öffne ich die Rails Konsole?

Rails bringt eine Möglichkeit mit während der Laufzeit der Anwendung Ruby Code im Kontext der Rails Anwendung auszuführen. So kann die Rails Konsole genutzt werden, um z.B. Nutzer, Buchungen oder Experimente händisch hinzuzufügen.

Allerdings gilt auch hier, dass sie direkt auf den Produktivdaten arbeitet. Ein falscher Befehl kann zum Löschen von wichtigen Daten führen. Eine Nutzung sollte bei Nichtkenntnis von Ruby und/oder Rails mit einem Entwickler abgesprochen werden.

```
cd /var/ww/app/current
RAILS_ENV=production bin/rails c
```