Titracka::Installation Guide

Wolfgang Barth

2020-11-14

Inhalt

Kapitel 1. Installationsvoraussetzungen für Titracka

Stand 2020-11-15

1.1. System

Hardware oder virtuelle Maschine

- x86_64 Architektur (Standard-Intel oder AMD-Prozessor)
- mind. 8 GByte Memory, besser 16 GByte
- mind. 2 Prozessorkerne
- Storage: >= 20 GByte (abhängig von den zu erwartenden Dateimengen; 50 GByte oder mehr sinnvoll wg. Datensicherung).

Betriebssystem

- Ubuntu LTS x86_64, aktuelle Version (z.Z. 20.04)
- Minimal-Ubuntu-Server-Installation

Datenbank

- Datenbank: MariaDB (MySQL nur nach Absprache)
- Andere Datenbanken auf Anfrage (z.B. PostgreSQL)

Webserver

- Apache 2.4 oder neuer
- Passenger standalone

Benutzer

- Benutzer `wob` mit sudo-Rechten
- Benutzer `deploy` ohne sudo-Rechte; unter diesem Benutzer läuft die Anwendung

1.2. Systemzugang

Für Installation und Betrieb sind zwei Benutzer `wob` und `deploy` erforderlich. Der Benutzer `wob` benötigt während der Installationsphase Root-Rechte (via sudo) Das System muss per SSH vom Standort Waldbreitbach aus über VPN erreichbar sein, entweder über einen festen VPN-Tunnel (ipsec) oder alternativ über OpenVPN.

Das System benötigt Internetzugang für die Installation und Aktualisierung von Systemsoftware (Ubuntu-Paketquellen) und Ruby-Gems (rubygems.org, github.com, u.a.). Die Nutzung eines Proxys mit User/Passwort und die Einschränkung auf nutzbare URLs ist möglich; genutzte Protokolle sind http/https.

1.3. Datensicherung

Die Datensicherung erfolgt auf mehreren Ebenen:

Systemsicherung
: Sicherung des Gesamtsystem

Systemkonfiguration
: Sicherung der Systemkonfigurationsdaten im Verzeichnis /etc

Anwendungsdaten
: alle für die Anwendung notwendige Daten

1.3.1. Systemsicherung

Eine Sicherung des Gesamtsystems wird nicht mit eingerichtet. Bitte sichern Sie das Gesamtsystem regelmäßig wie Ihre anderen Systeme auch (Images, Snapshots, etc).

1.3.2. Systemkonfiguration

Die Sicherung von Änderungen an der Systemkonfiguration im Verzeichnis `/etc` erfolgt mittels `etckeeper` über ein lokales Git-Repository. Die Anbindung an einen vorhandenen Git-Server ist ebenfalls möglich.

1.3.3. Sicherung der Anwendungsdaten

Die Sicherung der Anwendungsdaten (Datenbank, Dateien) erfolgt über das Ruby-Gem „backup“, als Storage sind möglich:

- Remote Server via Netzwerk: (Protokolle: FTP, SFTP, SCP, oder RSync)
- Lokaler Storage, auch gemountete Filesysteme (z.B. Windows-Dateiserver via smbfs).

Gesichert wird die Datenbank als SQL-Dump, die Dateien als `tar`-Archiv. Eine Verschlüsselung des Backups ist möglich, Versionierung nach Absprache.

Kapitel 2. Setup

2.1. Installation benötigter Pakete

Pakete für Debian/Ubuntu

```
# base packages
apt install bzip2 curl screen vim vim-common vim-rails \
  wget pwgen sudo etckeeper ldap-client p7zip-full gdebi aptitude

# net packages
apt install bridge-utils ethtool tcpdump vlan mtr-tiny

# daemons
apt install ntp xinetd

# mail
apt install postfix mutt bsd-mailx

# development
apt install autoconf binutils bison build-essential flex gettext ncurses-dev

# database
apt install mariadb-client-10.0 mariadb-client-core-10.0 mariadb-common \
  mariadb-server mariadb-server-10.0 mariadb-server-core-10.0
apt install libmariadb-client-lgpl-dev
apt install libsqlite3-dev

# webserver
apt install apache2 apache2-bin apache2-data apache2-utils
apt install libapache2-mod-passenger

# ruby
apt install ruby ruby-bundler ruby-dev

# wobrailsapps
apt install libxml2 libxml2-dev zlib1g zlib1g-dev
apt install nodejs
# für closure; should be replaced!
apt install openjdk-9-jre-headless
```

2.1.1. Yarn für das Deployment von Javascript

Pakete für Debian/Ubuntu

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
sudo apt-get update && sudo apt-get install yarn
```

Pakete für Fedora

```
sudo wget https://dl.yarnpkg.com/rpm/yarn.repo -O /etc/yum.repos.d/yarn.repo
curl --silent --location https://rpm.nodesource.com/setup_6.x | bash -
dnf install -y yarn
```

2.2. User `deploy`

Die Installation der Anwendung erfolgt unter dem User `deploy`. Bei der Installation und bei Updates werden Pakete aus dem Internet gezogen. In durch einen Proxy geschützten Netzwerken setzt man die Variablen `http_proxy` und `https_proxy` (`.bashrc`). Git greift nicht auf die Shellvariablen zu, sondern benötigt eine eigene Konfiguration.

bashrc: Proxy-Einstellungen

```
export http_proxy=http://<proxy>:<proxyport>/
export https_proxy=http://<proxy>:<proxyport>/
```

Git-Konfiguration User `deploy` für Updates der Anwendung

```
git config --global http.proxy http://<proxy>:<proxyport>/
git config --global https.proxy http://<proxy>:<proxyport>/
```

2.3. Verzeichnisse für Titracka

Vor dem ersten Deploy legt man die im Listing Verzeichnisse für Titracka angegebenen Verzeichnisse an. Die Verzeichnisse müssen alle dem User `deploy` gehören.

Verzeichnisse für Titracka

```
export BASE=/srv/www/titracka
mkdir -p ${BASE}/titracka/releases
mkdir -p ${BASE}/titracka/shared/config
mkdir -p ${BASE}/titracka/shared/log
mkdir -p ${BASE}/titracka/shared/files
mkdir -p ${BASE}/titracka/shared/public/images
mkdir -p ${BASE}/titracka/shared/Backup/models
mkdir -p ${BASE}/titracka/shared/pids
chown deploy:deploy -R ${BASE}/titracka
```

`./releases/`	Verzeichnis für die Installation des Codes. Normalerweise befinden sich hier die 5 letzten Updates, um bei Bedarf einen Rollback durchführen zu können. Ältere Versionen werden beim Deployment automatisch gelöscht.
`./current/`	Ein Symlink auf das jeweils aktuelle Release.
`./shared/`	Zentrales Verzeichnis für die statische Konfiguration und alle Dateien, die bei Updates der Anwendung unverändert bleiben.
`./shared/config/`	Konfigurationsdateien `application.yml`, `database.yml` und `secrets.yml` [2]
`./shared/log/`	Logdateien
`./shared/files/`	Dateiablage für Anlagen
`./shared/images/`	Dateiablage für Logos zu den OUs
`./shared/Backup/`	Konfigurationdateien für die Anwendungssicherung
`./shared/pids/`	wird von `delayed_job` für die Hintergrund-Jobs benötigt

[1] secrets.yml ab ab Rails 4.2
[2] secrets.yml ab ab Rails 4.2

Kapitel 3. Apache-Setup

Die Kombination Apache 2.4 + Passenger unterstützt leider keine Websockets. Das Setup ist daher etwas aufwendiger: Apache dient als Frontend (wg. Authentifikation u.a.) und Proxy-Server, Passenger Standalone bedient die Anwendung. Der Apache-Server bedient den Anwender per SSL

3.1. Apache

/etc/apache2/sites-available/titracka

```
<VirtualHost *:80>
  ServerName titracka.example.com
  ServerAdmin max.mustermann@example.com
  DocumentRoot /var/srv/www/titracka/public
  LimitRequestFieldSize 32768

  <Directory /var/srv/www/titracka/public>
    Require all granted
  </Directory>

  RequestHeader set X-Proxy-Secure-USER %{REMOTE_USER}s ①

  <Location "/">
    ProxyPass http://127.0.0.1:4000/ ②
    ProxyPassReverse http://127.0.0.1:4000/ ②
  </Location>

  <Location /cable>
    ProxyPass ws://127.0.0.1:4000/cable ②
    ProxyPassReverse ws://127.0.0.1:4000/cable ②
  </Location>

  <Proxy *>
    Order deny,allow
    Allow from all
  </Proxy>

  <LocationMatch "/auth/login">
    AuthName "Titracka"
    require valid-user
    AuthType CAS
    CASScope /titracka/
  </LocationMatch>

</VirtualHost>
```

```
<VirtualHost *:443>
  ServerName titracka.example.com
  ServerAdmin max.mustermann@example.com
  DocumentRoot /var/srv/www/titracka/public
  LimitRequestFieldSize 32768

  <Directory /var/srv/www/titracka/public>
    Require all granted
  </Directory>

  RequestHeader set X-Proxy-Secure-USER %{REMOTE_USER}s ①
  RequestHeader set X-Forwarded-Proto https ③

  <Location "/">
    ProxyPass http://127.0.0.1:4000/ ②
    ProxyPassReverse http://127.0.0.1:4000/ ②
  </Location>

  <Location /cable>
    ProxyPass ws://127.0.0.1:4000/cable ②
    ProxyPassReverse ws://127.0.0.1:4000/cable ②
  </Location>

  <Proxy *>
    Order deny,allow
    Allow from all
  </Proxy>

  <LocationMatch "/auth/login">
    AuthName "Titracka"
    require valid-user
    AuthType CAS
    CASScope /titracka/
  </LocationMatch>

  SSLEngine On
  SSLCertificateFile /etc/apache2/ssl/****.crt
  SSLCertificateKeyFile /etc/apache2/ssl/****.key
  SSLCipherSuite HIGH:!aNULL:!MD5
  SSLProtocol all -SSLv3 -TLSv1

</VirtualHost>
```

① Ein via Apache authentifizierter User wird in der Header-Variable X_PROXY_SECURE_USER an die Anwendung für Single-Sign-On weitergereicht

② **SECURITY**: bei Single-Sign-On darf die Anwendung nie direkt erreichbar sein, daher immer `localhost` oder `127.0.0.1` verwenden!

③ Verhindert bei `force_ssl=true` eine Endlosschleife, weil die Anwendung über den Proxy

ohne https aufgerufen wird und Passenger nicht erkennt, dass SSL verwendet werden soll.

3.2. Passenger Standalone

/var/srv/www/titracka/shared/config/Passengerfile.json

```
{
  "address": "127.0.0.1", ①
  "environment": "production",
  "daemonize": true, ②
  "port": "4000", ③
  "user": "deploy", ④
  "envvars": {
   }
}
```

① **SECURITY:** immer localhost oder 127.0.0.1 verwenden.

② Start als Daemon

③ Verwendeter Port, identisch zur Apache-Proxy-Konfiguration

④ User, unter dem die Anwendung läuft. Sollte identisch sein mit dem User, dem das Anwendungsverzeichnis `/var/srv/www/titracka` gehört.

Kapitel 4. Konfiguration

4.1. Zentrale Konfigurationdatei titracka.yml

Die Konfigurationsdatei titracka.yml ist reserviert für eine spätere Konfiguration via Datei. Die Datei muss existieren, kann aber auch zunächst leer sein.

titracka.yml

```
# config/titracka.yml
remote_user: HTTP_X_PROXY_SECURE_USER ①
use_ssl: true ②
action_cable_allowed_request_origins: ③
  - http://titracka.example.com
  - https://titracka.example.com
```

① Nur bei Single-Sign-On: User, der die REMOTE_USER-Variable des Apaches bereithält

② Erzwinge SSL; empfohlene Einstellung, sofern Apache2 für SSL konfiguriert ist

③ http:// und https:// Site-Urls für den Websocket

4.2. LDAP-Anbindung zur Übernahme von Benutzern und Kontakten

Bei der Anlage von Benutzern und Kontakten besteht die Möglichkeit, Daten aus einem vorhandenen Active Directory zu importieren. In der Anwendung wird das mit *Benutzer mit Vorlage erstellen* oder *Kontakt mit Vorlage erstellen* bezeichnet.

titracka.yml

```
# config/titracka.yml
ldap_options:
  host: 192.0.2.71 ①
  port: 3269 ②
  base: dc=example,dc=com ③
  encryption: :simple_tls
  auth:
    method: :simple
    username: ldapuser ④
    password: ldappasswd ⑤
```

① IP-Adresse des Domaincontrollers

② Port für den globalen Katalog: 3268=unverschlüsselt, 3269=via https

③ BaseDN des Active Directory

④ Benutzer für die LDAP-Abfrage

⑤ Passwort des LDAP-Benutzers

Tipp: der AD-Benutzer sollte ein Domänengast sein und sonst über keine weiteren Rechte im Active Directory verfügen.

Kapitel 5. Setup der Datenbank

5.1. Die Rails-Konfigurationsdatei database.yml

Die Rails-seitige Konfiguration der Datenbank besteht aus einer YAML-Datei, die Einträge für jede Umgebung enthält (Environment: `production`, `development`, `test`). Für die Produktivumgebung wird nur `production` benötigt, für eine Entwicklungsumgebung nur `development` und `test`.

Die Konfiguration ist abhängig vom Datenbanktreiber (`adapter`) und wird hier beschrieben für MySQL/MariaDB. Für andere Datenbanken bitte in der Rails-Dokumentation nachsehen.[3]

database.yml

```
# -- mysql2
production:
  adapter: mysql2 ①
  encoding: utf8
  database: titracka_production ②
  pool: 5
  username: ###PLEASE_EDIT###  ③
  password: ###PLEASE_EDIT###  ④
  socket: /var/run/mysql/mysql.sock ⑤
  # host:  localhost # alternativ zu ⑤
```

① gilt auch für MariaDB

② Name der Datenbank. Zur besseren Unterscheidung sollte man immer <name>_<environment> verwenden

③ Username für den Datenbankzugriff

④ Passwort für den Datenbankzugriff

⑤ Abhängig von der Datenbankkonfiguration wahlweise über IP/Host oder UNIX-Socket.

5.2. Initialize Database (MariaDB)

Set a root password

```
mysql -u root mysql
update user set password=PASSWORD('******') where User='root';
```

Create databases

```
mysql -u root mysql
CREATE DATABASE IF NOT EXISTS titracka_development
  CHARACTER SET = 'utf8'
  COLLATE       = 'utf8_general_ci';
CREATE DATABASE IF NOT EXISTS titracka_test
  CHARACTER SET = 'utf8'
  COLLATE       = 'utf8_general_ci';
CREATE DATABASE IF NOT EXISTS titracka_production
  CHARACTER SET = 'utf8'
  COLLATE       = 'utf8_general_ci'; ①

select * from information_schema.schemata;
```

① Für eine Produktivumgebung wird nur `titracka_production` benötigt

Create users

```
mysql -u root mysql
CREATE USER IF NOT EXISTS titracka_dev@localhost  IDENTIFIED BY '*****';
CREATE USER IF NOT EXISTS titracka_test@localhost IDENTIFIED BY '*****';
CREATE USER IF NOT EXISTS titracka_prod@localhost IDENTIFIED BY '*****';①

GRANT ALL on titracka_development.* to 'titracka_dev'@'localhost';
GRANT ALL on titracka_test.*        to 'titracka_test'@'localhost';
GRANT ALL on titracka_production.*  to 'titracka_prod'@'localhost';①
FLUSH PRIVILEGES;
```

① Für eine Produktivumgebung wird nur `titracka_prod` benötigt

[3] http://api.rubyonrails.org/classes/ActiveRecord/ConnectionAdapters.html