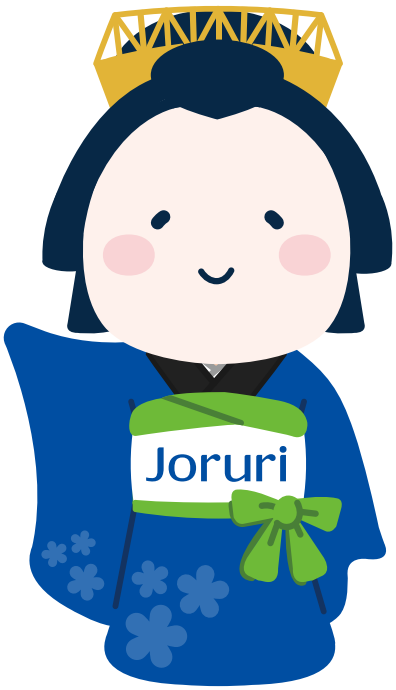
Joruri Mail 2022

Joruri Mail 2022
バージョンアップ手順書



2024-10-21 第一版
サイトブリッジ株式会社

目次

注意事項・事前準備

Joruri Mail 2022のバージョンアップを行う際に関連サービスの停止を行います。
そのため、バージョンアップ中はJoruri Mail 2022にアクセスできません。
また、本手順書は1台構成を想定したアップデート手順です。

事前準備として、Release2のソースコードをサーバーにアップロードします。
アップロード先パス：/usr/local/src/joruri-mail-2022-v2.0.0.tar.gz

Nginxのアップグレード

rootユーザーに変更します。
$ su -

Nginxをアップグレードします。
dnf update nginx

```
[root@localhost src]# dnf update nginx
Last metadata expiration check: 2:35:40 ago on Wed Oct 16 11:07:31 2024.
Dependencies resolved.
================================================================================
 Package       Architecture   Version                         Repository     Size
================================================================================
Upgrading:
 nginx         x86_64         1:1.26.2-1.el9.ngx              nginx         996 k

Transaction Summary
================================================================================
Upgrade  1 Package

Total download size: 996 k
Is this ok [y/N]:
```

アップグレードについて質問されるので「y」を入力して続行します。

Nginxのバージョンが1.26になっていることを確認します。
nginx -v

```
[root@localhost src]# nginx -v
nginx version: nginx/1.26.2
```

Rubyのアップグレード（1）

Rustをインストールします。
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
インストール方法についての質問に「1」と入力します。

```
1) Proceed with standard installation (default - just press enter)
2) Customize installation
3) Cancel installation
>1
```

source ~/.cargo/env
rustc --version

Rbenvを更新します。
cd /usr/local/rbenv/
git pull
cd /usr/local/rbenv/plugins/rbenv-vars/
git pull
cd /usr/local/rbenv/plugins/ruby-build/
git pull

Rubyのアップグレード（2）

Rubyをインストールします。

```
# rbenv install 3.3.4
# rbenv global 3.3.4
# rbenv rehash
# ruby -v
```

インストールしたRubyのバージョンが表示されます。3.3.4と表示されれば成功です。

```
[root@localhost ~]# ruby -v
ruby 3.3.4 (2024-07-09 revision be1089c8ec) [x86_64-linux]
```

MeCab-Rubyを再インストールします。ruby 3.2以降はアプリケーション起動時に警告が表示されるため、MeCab_wrap.cppにパッチをあてます。

```
# cd /usr/local/src
# curl -fsSL
'https://drive.google.com/uc?export=download&id=0B4y35FiV1wh7VUN
lczBWVDZJbE0' -o mecab-ruby-0.996.tar.gz
# tar zxf mecab-ruby-0.996.tar.gz
# cd mecab-ruby-0.996
# cp MeCab_wrap.cpp MeCab_wrap.cpp.bk
# vi MeCab_wrap.cpp
---
  VALUE cl = rb_define_class("swig_runtime_data", rb_cObject);
+ rb_undef_alloc_func(cl);
  /* create and store the structure pointer to a global variable */
---

#ruby extconf.rb && make && make install
# mecab -v
```

インストールしたMeCabのバージョンが表示されます

```
[root@localhost mecab-0.996]# mecab -v
mecab of 0.996
```

Nodejsのアップグレード

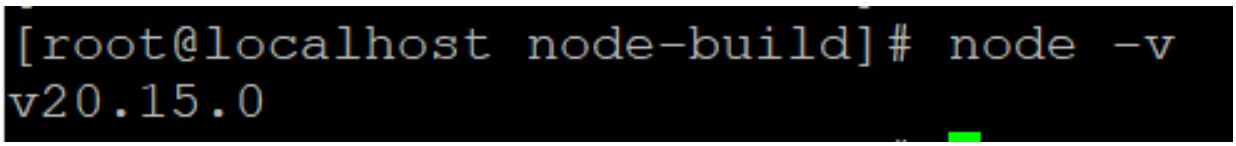
nodenvをアップグレードします。

```
# cd /usr/local/nodenv/
# git pull
```

nodejsをアップグレードします。

```
# cd /usr/local/nodenv/plugins/node-build/
# git pull
# nodenv install 20.15.0
# nodenv global 20.15.0
# nodenv rehash
# node -v
```

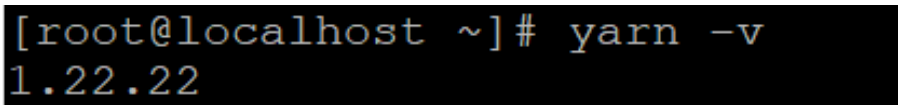
インストールしたnodejsのバージョンが表示されます。20.15.0と表示されれば成功です。



yarnをインストールします。

```
# npm install -g yarn
# nodenv rehash
# yarn -v
```

インストールしたyarnのバージョンが表示されます。



サービスの停止

nginxを停止します。

systemctl stop nginx
systemctl status nginx
Active: inactive と表示されることを確認します。

pumaを停止します。

systemctl stop pwm_puma
systemctl status pwm_puma
Active: inactive と表示されることを確認します。

delayed_jobを停止します。

systemctl stop pwm_delayed_job
systemctl status pwm_delayed_job
Active: inactive と表示されることを確認します。

crondを停止します。

systemctl stop crond
systemctl status crond
Active: inactive と表示されることを確認します。

ソースコード差し替え（1）

既存のJoruri Mail 2022ソースコードとRelease2のソースコードを差し替えます。

前回インストールしたpwmのソースコードが/usr/local/src配下に存在するか確認します。
```
# ls –lh /usr/local/src/pwm
```

存在する場合はリネームします。
```
# mv /usr/local/src/pwm /usr/local/src/pwm_v1
```

Release2のソースコードを解凍します。
```
# cd /usr/local/src
# tar -xvzf joruri-mail-2022-v2.0.0.tar.gz
```

既存のソースコードをリネームして退避します。
```
# mv /var/www/pwm /var/www/pwm_v1
```

Release2のソースコードを設置します。
```
# cp -r /usr/local/src/pwm /var/www/pwm
# chown -R pwm:pwm /var/www/pwm
```

pwmユーザーに変更します。
```
# su - pwm
```

ソースコードのパーミッションを変更します。
```
$ find /var/www/pwm -type d -exec chmod 755 {} ¥;
```

既存のソースコードの設定ファイルをRelease2にコピーします。
```
$ cd /var/www/pwm
$ cp /var/www/pwm_v1/config/*yml /var/www/pwm/config/.
$ cp /var/www/pwm_v1/config/credentials/*
/var/www/pwm/config/credentials/.
```

ソースコード差し替え（2）

gemライブラリをインストールします。

```
$ cp config/samples/bundler/Gemfile.engines .
$ bundle config build.pg --with-pg-config=/usr/pgsql-15/bin/pg_config
$ bundle config set --local path 'vendor/bundle'
$ bundle config set --local without 'development test'
$ bundle install
$ bundle list
```

DBを更新します。

```
$ bundle exec rake db:migrate RAILS_ENV=production
$ bundle exec rake db:version RAILS_ENV=production
```

sample.Gemfile.enginesをリネームします。

```
$ mv Gemfile.engines bk.Gemfile.engines
$ mv sample.Gemfile.engines Gemfile.engines
```

追加アプリケーションをインストールします。

```
$ bundle install
```

追加アプリケーション分のDB更新を反映します。

```
$ bundle exec rake db:migrate RAILS_ENV=production
$ bundle exec rake db:version RAILS_ENV=production
```

jsライブラリを更新します。

```
$ yarn install --production
$ yarn list
$ bin/install/assets.sh
```

ソースコード差し替え（3）

アセットをコンパイルします。

$ bundle exec rake assets:precompile RAILS_ENV=production

最新のassetsファイルが作成されていることを確認します。

$ ls -l public/assets/**/*

cronに定期実行処理を追加します。

$ bundle exec whenever --update-crontab

```
[pwm@localhost pwm]$ crontab -l


# Begin Whenever generated tasks for: /var/www/pwm/config/schedule.rb at: 2024-0
8-07 14:46:49 +0900
0 * * * * /bin/bash -l -c 'cd /var/www/pwm && RAILS_ENV=production bundle exec r
ake pwm_core:jobs:schedule --silent'

0 5 * * * /bin/bash -l -c 'cd /var/www/pwm && RAILS_ENV=production bundle exec r
ake pwm_core:vacuum --silent'

0 6 * * * /bin/bash -l -c 'cd /var/www/pwm && RAILS_ENV=production bundle exec r
ake pwm_core:reindex --silent'

0 0 * * 0 /bin/bash -l -c 'cd /var/www/pwm && RAILS_ENV=production bundle exec r
ake pwm_core:clean --silent'

# End Whenever generated tasks for: /var/www/pwm/config/schedule.rb at: 2024-08-
07 14:46:49 +0900
```

railsコンソールを起動できることを確認します。

$ ./bin/rails console -e production

> exit

メールのスレッドデータ再構築

メールのスレッドデータを再構築します。

※全メールのスレッドデータを再構築します。負荷が高く時間がかかりますので注意してください。

```
$ bundle exec rake pwm_wmail:threadings:refresh FORCE=true
RAILS_ENV=production
```

サービスの再起動

rootユーザーに切り替えます。

$ su -

pumaを起動します。

systemctl start pwm_puma
systemctl status pwm_puma
Active: active (running)と表示されることを確認します。

delayed_jobを起動します。

systemctl start pwm_delayed_job
systemctl status pwm_delayed_job
Active: active (running)と表示されることを確認します。

crondを起動します。

systemctl start crond
systemctl status crond
Active: active (running)と表示されることを確認します。

nginxを起動します。

systemctl start nginx
systemctl status nginx
Active: active (running)と表示されることを確認します。

ログインの確認

ブラウザでJoruri Mail 2022にアクセスしログインを確認します。

https://pwm.localdomain.jp/
※標準インストール時のURLです。インストール時の設定に合わせて適宜読み替えてください。

- ユーザーID: pwm
- パスワード: pwm

ログイン画面にVer2.0.0と表示されることを確認します。

検索インデックス再作成

全アカウントの検索インデックスを再作成します。

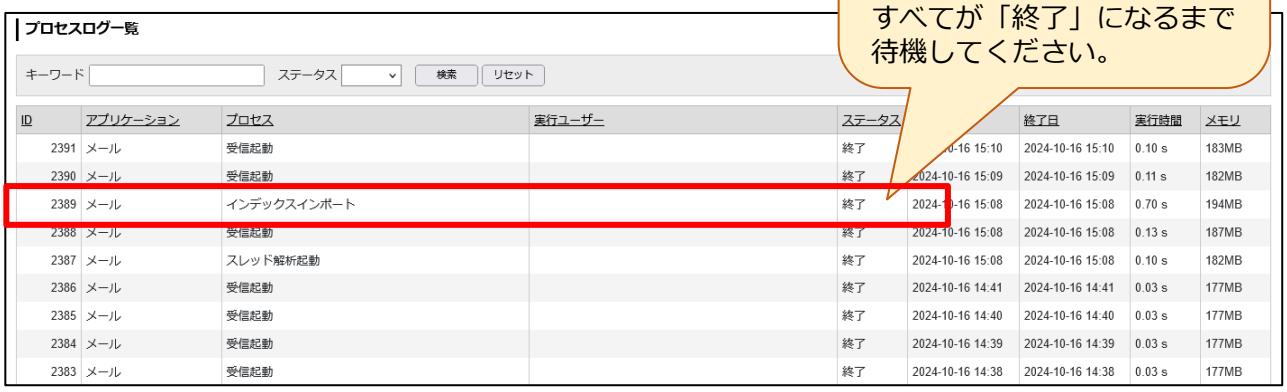
pwmユーザーに切り替えます。

```
# su - pwm
$ cd /var/www/pwm
```

インデックスインポート(import_index)タスクを実行すると、プロセスログ画面から進行状況を確認できるので、完了まで待機します。

```
$ bundle exec rake pwm_wmail_search:search:create_index RAILS_ENV=production FORCE=true
$ bundle exec rake pwm_wmail_search:search:set_refresh_interval RAILS_ENV=production INTERVAL=60s
$ bundle exec rake pwm_wmail_search:search:import_index RAILS_ENV=production TARGET_ALL_ACCOUNT_CHECK=1 BATCH_SIZE=100
```

プロセスログ画面はシステム管理者でPWMにログインし、左上メニューを展開して「プロセスログ」を選択してください。



「インデックスインポート」プロセスのステータスを確認します。
ユーザー数やデータ規模に応じして複数回実行されます。
すべてが「終了」になるまで待機してください。

プロセスログ画面を確認して、インデックスインポートが完了したら下記コマンドを実行します。

```
$ bundle exec rake pwm_wmail_search:search:set_refresh_interval RAILS_ENV=production INTERVAL=1s
```