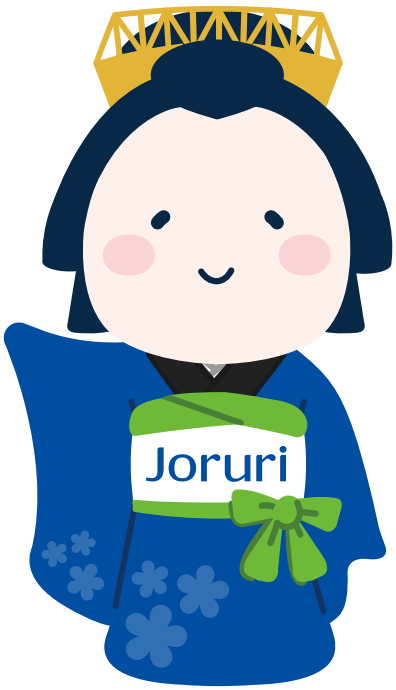
Joruri Mail 2022

JoruriMail2022
Release2
インストール手順



2024-10-21 第二版
サイトブリッジ株式会社

目次

Joruri Mail 2022は、Joruri PWMというプロダクトをベースに、メール関連アプリケーションを追加して稼働しております
インストールは以下の順番で実施します。

想定環境・事前準備

前提条件
下記の構成でAlmaLinux9をインストール済みとします。
- 言語サポート：日本語
- ソフトウェアの選択：最小限のインストール
- ホスト名：pwm.localdomain.jp

また、インターネット接続が可能な環境でのインストールを前提としています。

想定環境
- AlmaLinux 9.x (x86_64)
- Nginx 1.26
- PostgreSQL 15
- Redis 6.2
- Node.js 20
- Ruby 3.1/3.3
- Rails 7.1

インストール
インストールの作業ログを取得する場合は下記を実行します。

```
# script /root/pwm_install.log
```

selinuxを無効にします。

```
# /usr/sbin/setenforce 0
# vi /etc/sysconfig/selinux
---
SELINUX=permissive  # permissiveに変更
---
```

ロケールを確認し、必要に応じて日本語に設定します。

```
# localectl status
# localectl set-locale LANG=ja_JP.UTF-8
```

必要なツールをインストール

crbを有効にします。

```
# dnf config-manager --set-enabled crb
```

必要なツールをインストールします。

```
# dnf -y install git wget patch unzip tar epel-release
```

RbenvでRubyをインストール

必要なパッケージをインストールします。

dnf -y install gcc-c++ libffi-devel libyaml-devel make openssl-devel readline-devel zlib-devel bzip2 jemalloc-devel

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
インストールしたrustcのバージョンが表示されます。

```
[root@localhost ~]# rustc --version
rustc 1.81.0 (eeb90cda1 2024-09-04)
```

rbenvをインストールします。

git clone https://github.com/rbenv/rbenv.git /usr/local/rbenv
git clone https://github.com/rbenv/ruby-build.git /usr/local/rbenv/plugins/ruby-build
git clone https://github.com/rbenv/rbenv-vars.git /usr/local/rbenv/plugins/rbenv-vars
vi /etc/profile.d/rbenv.sh
---
export RBENV_ROOT="/usr/local/rbenv"
export PATH="${RBENV_ROOT}/bin:${PATH}"
eval "$(rbenv init -)"
---
source /etc/profile.d/rbenv.sh

rbenvで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。

```
[root@localhost ~]# ruby -v
ruby 3.3.4 (2024-07-09 revision be1089c8ec) [x86 64-linux]
```

nodenvでNode.jsをインストール

nodenvをインストールします。

```
# git clone https://github.com/nodenv/nodenv.git /usr/local/nodenv
# git clone https://github.com/nodenv/node-build.git
/usr/local/nodenv/plugins/node-build
# vi /etc/profile.d/nodenv.sh
---
export NODENV_ROOT="/usr/local/nodenv"
export PATH="${NODENV_ROOT}/bin:$PATH"
eval "$(nodenv init -)"
---
# source /etc/profile.d/nodenv.sh
```

nodenvでnodejsをインストールします。

```
# nodenv install 20.15.0
# nodenv global 20.15.0
# nodenv rehash
# node -v
```

インストールしたnodeのバージョンが表示されます。

```
[root@localhost ~]# node -v
v20.15.0
```

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。

```
[root@localhost ~]# yarn -v
1.22.22
```

Nginxをインストール

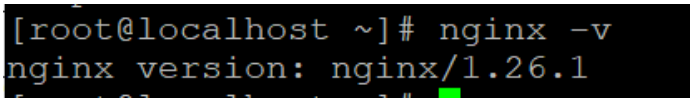
yumリポジトリを追加します。

```
# vi /etc/yum.repos.d/nginx.repo
---
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/9/$basearch/
gpgcheck=0
enabled=1
---
```

nginxをインストールします。

```
# dnf -y install nginx
# nginx -v
```

インストールしたnginxのバージョンが表示されます。



不要な設定ファイルをリネームします。

```
# mv /etc/nginx/conf.d/default.conf /etc/nginx/conf.d/default.conf.org
```

設定を変更します。

```
# vi /etc/nginx/nginx.conf
---
gzip  on;   # コメントを外してgzipを有効にします
---
```

Postgresqlをインストール

postgresqlをインストールします。

dnf -y install
https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/pgdg-redhat-repo-latest.noarch.rpm
dnf -y install postgresql15-server postgresql15-contrib postgresql15-devel

DBを初期化します。

/usr/pgsql-15/bin/postgresql-15-setup initdb

ユーザー認証方法を変更します。

vi /var/lib/pgsql/15/data/pg_hba.conf

---
host　all　　all　　127.0.0.1/32　　scram-sha-256
---

postgresqlを起動します。

systemctl enable postgresql-15 && systemctl start postgresql-15
systemctl status postgresql-15
Active: active(running)と表示されることを確認します。

接続を確認します。

su - postgres -c “psql -c ’select version()’ ”

インストールしたPostgreSQLのバージョンが表示されます。

```
[root@localhost ~]# su - postgres -c "psql -c 'select version()'"
                                                 version

-------------------------------------------------------------------------------
-------------------------
 PostgreSQL 15.7 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 11.4.1 20231218 (
Red Hat 11.4.1-3), 64-bit
(1 row)
```

Redisをインストール

redisをインストールします。

dnf -y install redis hiredis
redis-server –v
インストールしたredisのバージョンが表示されます。

```
[root@localhost ~]# redis-server -v
Redis server v=6.2.7 sha=00000000:0 malloc=jemalloc-5.1.0 bits=64 build=ec192bdd
77ecd321
```

redisを設定します。
vi /etc/redis/redis.conf

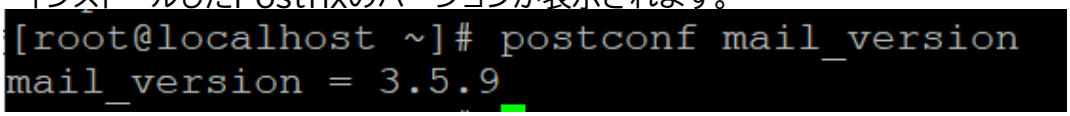
下記の行を追加します。環境毎に適切な値を設定してください。
maxmemory 500mb
maxmemory-policy allkeys-lru

Postfixをインストール

postfixをインストールします。

```
# dnf -y install postfix
# postconf mail_version
```

インストールしたPostfixのバージョンが表示されます。



Joruri PWMをインストール（1）

必要なパッケージをインストールします。

```
# dnf -y install ImageMagick-devel zip libicu-devel
```

pwmユーザーを追加します。

```
# useradd pwm
```

DB接続ユーザーを追加します。

```
# su - postgres
$ createuser --createdb --pwprompt pwm
```
※DB接続ユーザーのパスワードを入力してください。

```
$ createdb pwm
$ exit
```

Joruri Mail 2022ソースコードを下記のパスにアップロードします。

/usr/local/src/pwm-normal-v2.0.0.tar.gz

ソースコードを解凍して/var/www以下に設置します。

```
# cd  /usr/local/src
# tar -xvzf pwm-normal-v2.0.0.tar.gz
# mkdir –p /var/www
# cp -r /usr/local/src/pwm /var/www/pwm
# chown -R pwm:pwm /var/www/pwm
```

pwmユーザーに切り替えます。

```
# su - pwm
```

設置した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 {} ¥;
```

gemライブラリをインストールします。

```
$ cd /var/www/pwm
$ 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
```

Joruri PWMのインストール（2）

jsライブラリをインストールします。

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

デフォルト設定ファイルをコピーします。

```
$ cp /var/www/pwm/config/original/*.yml /var/www/pwm/config/
$ cp /var/www/pwm/config/original/credentials/*
/var/www/pwm/config/credentials/
```

database.ymlを設定します。

```
$ vi /var/www/pwm/config/database.yml
---
production:
  primary:
    <<: *default
    database: pwm_production
    username: pwm
    password: [YOUR PASSWORD]
---
```

※[YOUR PASSWORD]はDB接続ユーザーのパスワードに変更してください。

credentialsの設定値を生成します。生成された設定値はテキストファイルにコピーしてください。

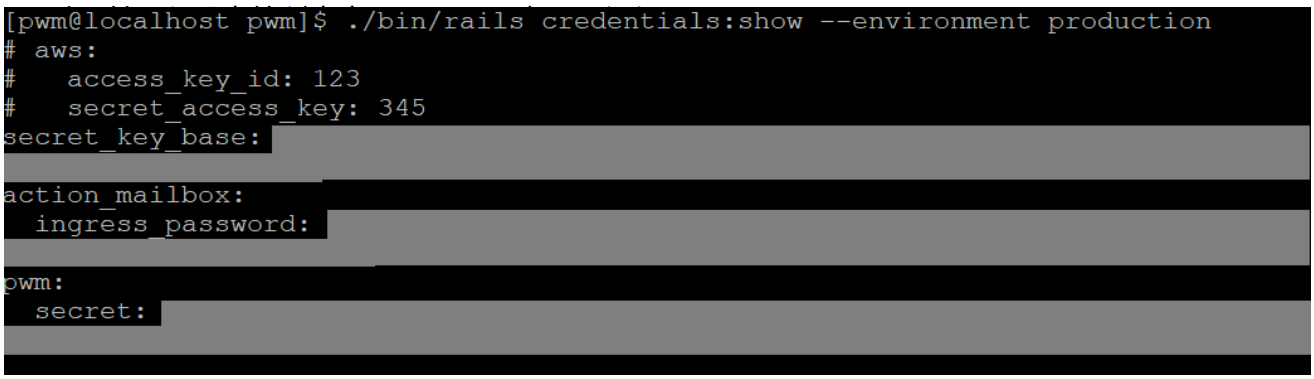
```
 $ bundle exec rake pwm:credentials:generate RAILS_ENV=production
```

credentialsエディターを起動し、コピーした設定値を貼り付けて保存します。

```
$ EDITOR=vi ./bin/rails credentials:edit --environment production
```

credentialsに保存されたことを確認します。

```
$ ./bin/rails credentials:show --environment production
```



Joruri PWMのインストール（3）

DBを作成します。

$ bundle exec rake db:create db:migrate db:seed RAILS_ENV=production
$ bundle exec rake db:version RAILS_ENV=production

作成したデータベースの名称とバージョンが表示されます。

```
database: pwm_production
Current version: 20240918071457
```

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
$ crontab -l

Cronに/var/www/pwmのジョブが追加されていることを確認します。

```
[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

サーバーの設定

rootユーザーに切り替えます。

```
$ su -
```

nginxを設定します。

```
# cp /var/www/pwm/config/samples/nginx/pwm.conf /etc/nginx/conf.d/.
# cp -r /var/www/pwm/config/samples/nginx/pwm.d /etc/nginx/conf.d/.
# vi /etc/nginx/conf.d/pwm.conf
---
server {
  ...
  server_name pwm.localdomain.jp;
  ...
}
---
```

※環境に応じて適切にserver_nameを設定してください。

ログローテートを設定します。

```
# cp /var/www/pwm/config/samples/logrotate/pwm /etc/logrotate.d/.
```

サービスの起動

postgresqlを起動します。

systemctl enable postgresql-15 && systemctl start postgresql-15
systemctl status postgresql-15
Active: active(running)と表示されることを確認します。

```
[root@localhost src]# systemctl status postgresql-15
● postgresql-15.service - PostgreSQL 15 database server
     Loaded: loaded (/usr/lib/systemd/system/postgresql-15.service; enabled; pr>
     Active: active (running) since Wed 2024-08-07 13:40:31 JST; 1h 24min ago
       Docs: https://www.postgresql.org/docs/15/static/
   Main PID: 35892 (postmaster)
      Tasks: 7 (limit: 23083)
```

redisを起動します。

systemctl enable redis && systemctl start redis
systemctl status redis
postgresql と同様にActive: active(running)と表示されることを確認します。

postfixを起動します。

systemctl enable postfix && systemctl start postfix
systemctl status postfix
postgresql と同様に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 enable nginx && systemctl start nginx
systemctl status nginx
postgresql と同様にActive: active(running)と表示されることを確認します。

pumaを起動します。

cp /var/www/pwm/config/samples/systemd/puma.service
/etc/systemd/system/pwm_puma.service
systemctl enable pwm_puma && systemctl start pwm_puma
systemctl status pwm_puma
postgresql と同様に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。

cp /var/www/pwm/config/samples/systemd/delayed_job.service
/etc/systemd/system/pwm_delayed_job.service
systemctl enable pwm_delayed_job && systemctl start pwm_delayed_job
systemctl status pwm_delayed_job
postgresql と同様にActive: active(running)と表示されることを確認します。

ポートの開放

httpおよびhttpsポートを開放します。
※ファイアウォールは環境に応じて適切に設定してください。

```
# firewall-cmd --add-service=http --zone=public --permanent
# firewall-cmd --add-service=https --zone=public --permanent
```

firewallをリロードします。

```
# firewall-cmd --reload
# firewall-cmd --list-all
```

servicesにhttp, httpsが表示されることを確認します。

```
[root@localhost src]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3 enp0s8
  sources:
  services: cockpit dhcpv6-client http https ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

ログインの確認

ブラウザで下記のURL(「サーバーの設定」で指定したドメイン)にアクセスしログインを確認します。

https://pwm.localdomain.jp/

* ユーザーID: pwm
* パスワード: pwm

※ドメインをDNSに登録していない場合は接続エラーになります。
hostsファイルにIPアドレスとドメインの組み合わせを追加してください。

IPアドレスを確認します。

```
# ip a
```

hostsファイルを編集します。

Linux:

```
# vi /etc/hosts
---
[IP ADDRESS]    pwm.localdomain.jp
---
```

Windows:

```
> notepad C:¥Windows¥System32¥drivers¥etc¥hosts
---
[IP ADDRESS]    pwm.localdomain.jp
---
```

Libreofficeのインストール

Joruri Mail 2022はJoruri PWMというプロダクトにメール関連アプリケーションを追加インストールして稼働しています。
以後の手順ではメール関連アプリケーションに必要なライブラリをインストールしていきます。

メールの添付ファイル解説のためにLibre Officeをインストールします。

libreofficeをインストールします。
$ su -
# dnf -y install libreoffice libreoffice-langpack-ja
# soffice –version

インストールしたlibreofficeのバージョンが表示されます。

```
[root@localhost src]# soffice --version
LibreOffice 7.1.8.1 10(Build:1)
```

sofficeのサービスファイルを設置します。

cp /var/www/pwm/vendor/engines/pwm-core-libre/config/samples/systemd/soffice.service /etc/systemd/system/soffice.service

sofficeを起動します。

systemctl start soffice && systemctl enable soffice
systemctl status soffice

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

Tika-serverのインストール

PDF等の添付ファイルからテキストデータを抽出し検索可能にするために、Tika-serverをインストールします。

Javaをインストールします。

```
# dnf -y install java-21-openjdk java-21-openjdk-devel
# java -version
```

インストールしたJavaのバージョンが表示されます。

```
[root@localhost src]#  java -version
openjdk version "11.0.24" 2024-07-16 LTS
OpenJDK Runtime Environment (Red_Hat-11.0.24.0.8-2) (build 11.0.24+8-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-11.0.24.0.8-2) (build 11.0.24+8-LTS, mixed mod
e, sharing)
```

tika-serverをインストールします。

```
# curl https://archive.apache.org/dist/tika/2.9.2/tika-server-standard-2.9.2.jar -o /usr/local/sbin/tika-server.jar
```

tika-serverの起動設定を取得します。

```
# cp /var/www/pwm/vendor/engines/pwm-core-tika/config/samples/systemd/tika.service /etc/systemd/system/tika.service
```

tika-serverを起動します。

```
# systemctl enable tika && systemctl start tika
# systemctl status tika
```

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

Tika-serverの稼働を確認します。

```
# curl http://localhost:9998/tika
```

インストールしたtika-serverのバージョンが表示されます。

```
[root@localhost src]# curl http://localhost:9998/tika
This is Tika Server (Apache Tika 2.9.2). Please PUT
```

Elasticsearchのインストール（１）

メール検索エンジンとしてElasticsearchをインストールします。
複数台のサーバーにElasticsearchをインストールしてクラスタを構成する場合は
TCPを対象として各サーバ―の9300ポートを開けてください。

```
$ su -
# curl -fsSLO
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.9.1-
x86_64.rpm
# sha1sum elasticsearch-8.9.1-x86_64.rpm
# rpm -ivh elasticsearch-8.9.1-x86_64.rpm
```

Elasticsearchの設定ファイルを変更します。

```
# vi /etc/elasticsearch/elasticsearch.yml
```

```
---
cluster.name: pwm  # アプリケーションに応じて適宜変更
node.name: node-001  # 各サーバ―ごとに適宜変更
node.roles: [data, master, ingest, ml]  # 複数台構成の場合は最低1つはmasterを指定
network.host: xxx.xxx.xxx.xxx  # Elasticsearchがインストールされているサーバのプライ
ベートIPを指定(ノードが1つのときはlocalhostでもよい)
http.port: 9200
xpack.security.enabled: false  # プライベートネットワークで稼働させることを想定するた
めfalseを指定
xpack.security.enrollment.enabled: false
xpack.security.http.ssl:
  enabled: false
xpack.security.transport.ssl:
  enabled: false
#http.host: 0.0.0.0  # コメントアウト
---
```

Elasticsearchのクラスタのノードが複数の場合は下記のように設定します。

```
---
discovery.seed_hosts: ["xxx.xxx.xxx.xxx:9300"]  # Elasticsearchがインストールさ
れているサーバのIPを指定
cluster.initial_master_nodes: ["node-001"]  # masterノードを指定
---
```

Elasticsearchのクラスタのノードが1つの場合は下記のように設定します。

```
---
# cluster.initial_master_nodes: ["node-1", "node-2"] #コメントアウト
discovery.type: single-node
---
```

Elasticsearchのインストール（2）

Elasticsearchが使用するJavaヒープメモリ量を設定します。

```
# touch /etc/elasticsearch/jvm.options.d/heap.options
# vi /etc/elasticsearch/jvm.options.d/heap.options
---
 -Xms1g # 推奨は物理メモリの半分
 -Xmx1g # -Xmsと同じメモリ量とする
---
```

Elasticsearchのインストール（3）

ICU Analysis Pluginをインストールします。

```
# cd /usr/share/elasticsearch
# bin/elasticsearch-plugin install analysis-icu
```

```
[root@localhost elasticsearch]# bin/elasticsearch-plugin install analysis-icu
-> Installing analysis-icu
-> Downloading analysis-icu from elastic
[=================================================] 100%
-> Installed analysis-icu
-> Please restart Elasticsearch to activate any plugins installed
```

```
# cd /usr/share/elasticsearch
# bin/elasticsearch-plugin install analysis-kuromoji
```

```
[root@localhost elasticsearch]# bin/elasticsearch-plugin install analysis-kuromo
ji
-> Installing analysis-kuromoji
-> Downloading analysis-kuromoji from elastic
[=================================================] 100%
-> Installed analysis-kuromoji
-> Please restart Elasticsearch to activate any plugins installed
```

elasticsearch.keystoreが/etc/elasticsearch以下になければ作成します。

```
# cd /usr/share/elasticsearch
# bin/elasticsearch-keystore create
```

Elasticsearchを起動します。

```
# systemctl enable elasticsearch && systemctl start elasticsearch
# systemctl status elasticsearch
```

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

cluster.initial_master_nodesを設定した場合は起動後にコメントアウトします。

```
# vi /etc/elasticsearch/elasticsearch.yml
---
#cluster.initial_master_nodes: [”node-001”]
---
```

Elasticsearchのインストール（4）

Elasticsearchが正常に応答するか確認します。

curl -XGET ”xxx.xxx.xxx.xxx:9200”
※「xxx.xxx.xxx.xxx」にはelasticsearch.ymlのnetwork.hostと同じ内容を指定
下記のようにバージョン情報が表示されていれば正常に応答しています。

```
[root@localhost elasticsearch]# curl -XGET "localhost:9200"
{
  "name" : "node-001",
  "cluster_name" : "pwm",
  "cluster_uuid" : "7gsgiEGiSjWw33X34f4jDw",
  "version" : {
    "number" : "8.9.1",
    "build_flavor" : "default",
    "build_type" : "rpm",
    "build_hash" : "a813d015ef1826148d9d389bd1c0d781c6e349f0",
    "build_date" : "2023-08-10T05:02:32.517455352Z",
    "build_snapshot" : false,
    "lucene_version" : "9.7.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Elasticsearch連携設定

Elasticsearch連携設定用ファイルを作成します。

```
# su - pwm
$ cd /var/www/pwm
$ touch config/pwm_elasticclt.yml
$ vi config/pwm_elasticclt.yml

---
elasticclt:
  elasticsearch:
    host: xxx.xxx.xxx.xxx # elasticsearch.ymlのnetwork.hostと同じ内容を指定
    port: 9200
---


# su - pwm
$ cd /var/www/pwm
$ touch config/pwm_wmail_search.yml
$ vi config/pwm_wmail_search.yml
---
wmail_search:
  elasticsearch:
    index_name: pwm_wmail_production
---
```

スパムメール対策ライブラリをインストール

スパムメール対策のため形態素解析エンジンとしてMeCabをインストールします。

```
$ su -
# cd /usr/local/src
# curl -fsSL
'https://drive.google.com/uc?export=download&id=0B4y35FiV1wh7cENtOXlicTFaRUE' -o mecab-0.996.tar.gz
# tar zxf mecab-0.996.tar.gz && cd mecab-0.996 && ./configure --enable-utf8-only && 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
```

MeCab-IPAdicをインストールします。

```
# cd /usr/local/src
# curl -fsSL
'https://drive.google.com/uc?export=download&id=0B4y35FiV1wh7MWVlSDBCSXZMTXM' -o mecab-ipadic-2.7.0-20070801.tar.gz
# tar zxf mecab-ipadic-2.7.0-20070801.tar.gz && cd mecab-ipadic-2.7.0-20070801 && ./configure --with-charset=utf8 && make && make install
```

MeCab-Rubyを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=0B4y35FiV1wh7VUNlczBWVDZJbE0' -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
```

libmecabのパスを設定します。

```
# echo '/usr/local/lib' >> /etc/ld.so.conf.d/usrlocal.conf
# ldconfig
# ldconfig -p | grep "/usr/local/lib"
```

メールアプリケーションをインストール（1）

メール関連ライブラリのインストールの次は、実際にPWMに追加アプリケーションをインストールします。
追加アプリケーションは「/var/www/pwm/Gemfile.engines」というファイルに記載します。配布コードにサンプルのリストが同梱されているのでリネームして利用します。
Sample.Gemfile.enginesをリネームします。

```
# su – pwm
$ cd /var/www/pwm
$ mv Gemfile.engines Gemfile.engines.bk
$ mv sample.Gemfile.engines Gemfile.engines
```

サンプルのGemfile.enginesには様々な追加アプリケーションを記載しています。
機能が不要であればコメントアウトした上でアプリケーションをインストールしてください。

コメントアウトの例

- メール検索機能

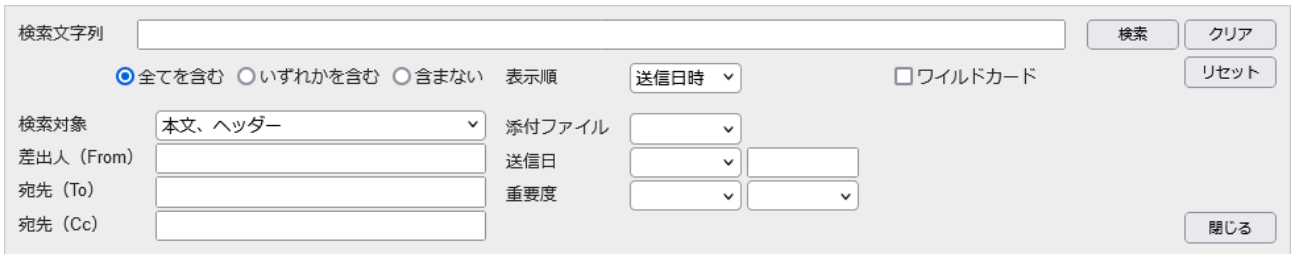
Joruri Mail 2022では、データベース上のメールデータをElasticsearchに連携させることで高度な検索を実現しています。
メール検索機能を利用しない場合は「Elasticsearchのインストール」「Elasticsearch連携設定」の手順は不要です。検索機能が不要な場合は「 /var/www/pem/Gemfile.engines 」ファイルから以下の3つのアプリケーションをコメントアウト（行頭に「＃（半角シャープ）」を記入してください。除外することでメール検索フォームが簡易な文字列検索のみのフォームとなります。

```
$ vi Gemfile.engines

#------
# gem ‘pwm-elasticclt’, path: ‘/var/www/pwm/vendor/engines/pwm-elasticclt’
# gem ‘pwm-elasticman’, path: ‘/var/www/pwm/vendor/engines/pwm-elasticman’
# gem ‘pwm-wmail-search’, path: ‘/var/www/pwm/vendor/engines/pwm-wmail-search’
#------
```

デフォルトの検索フォーム

デフォルトの検索フォームはインデックス検索を実施します。複数の条件を指定して検索できます。



簡易版検索フォーム

簡易版検索フォームはSQLによるメール検索を行うため、単純な部分一致検索を実施します。利用ユーザー数やメール件数によってパフォーマンスに影響が出る可能性があります。

メールアプリケーションをインストール（2）

- 各種アドレス帳

デフォルトの設定では、社員名簿等の他アプリケーションと連携する状態になっています。不要な場合は連携用アプリケーションを削除してください。

$ vi Gemfile.engines

#------
#社員名簿を除外
gem 'pwm-wmail-address-staff', path:
'/var/www/pwm/vendor/engines/pwm-wmail-address-staff'
#------

不要な連携アプリケーションを削除するとメール一覧の左ツリーに関連アプリケーションの名称が表示されなくなります。

メールアプリケーションをインストール（3）

インストールするアプリケーションの指定が終わったら追加インストールを実施します。

$ bundle install

DBを更新します。

$ bundle exec rake db:migrate RAILS_ENV=production

assetsを更新します。

$ bundle exec rake assets:precompile RAILS_ENV=production

cronを更新します。

$ bundle exec whenever --update-crontab

サービスの再起動

pumaとdelayed_jobを再起動します。

$ su -
# systemctl restart pwm_puma
# systemctl restart pwm_delayed_job

Active: active(running)となっていることを確認します。
systemctl status pwm_puma
systemctl status pwm_delayed_job

メール検索設定（1）

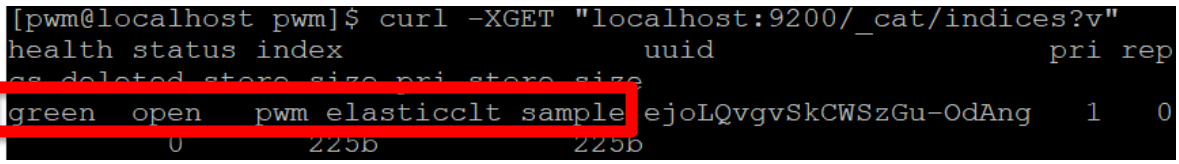
Elasticsearchのインデックスが正常に作成できるか確認するため、サンプルのインデックスを作成します。

```
# su - pwm
$ cd /var/www/pwm
$ bundle exec rake pwm_elasticclt:elasticsearch:sample:create_index
RAILS_ENV=production
```

Elasticsearchのインデックスが正常に作成されているか確認します。
※「xxx.xxx.xxx.xxx」にはelasticsearch.ymlのnetwork.hostと同じ内容を指定します

```
$ curl -XGET "xxx.xxx.xxx.xxx:9200/_cat/indices?v"
```

下記のように表示されていれば正常に作成されています。



サンプルのインデックスを削除します。

```
$ bundle exec rake pwm_elasticclt:elasticsearch:sample:delete_index
RAILS_ENV=production
```

サンプルのインデックスが削除されていることを確認します。
※「xxx.xxx.xxx.xxx」にはelasticsearch.ymlのnetwork.hostと同じ内容を指定します

```
$ curl -XGET "xxx.xxx.xxx.xxx:9200/_cat/indices?v"
```

「pwm_elasticclt_sample」が表示されていなければ、正常に削除されています。

メール検索設定（２）

メール検索のためにElasticsearchのインデックスを作成します。
アプリケーションインストール時にメール検索アプリケーションを除外した場合はこの手順は不要です。

```
# su - pwm
$ cd /var/www/pwm
$ bundle exec rake pwm_wmail_search:search:create_index
RAILS_ENV=production FORCE=true
```

Elasticsearchのインデックスが正常に作成されているか確認します。
※「xxx.xxx.xxx.xxx」にはelasticsearch.ymlのnetwork.hostと同じ内容を指定します

```
$ curl -XGET ”xxx.xxx.xxx.xxx:9200/_cat/indices?v“
```

下記のように表示されていれば正常に作成されています。

```
[pwm@localhost pwm]$ curl -XGET "localhost:9200/_cat/indices?v
health status index                uuid                   pri
s.deleted store.size pri.store.size
green  open   pwm_wmail_production Ins4PbZeSpyzY64KBxgTmA   1
        0       225b           225b
```

PWMを再起動します。

```
$ su -
# systemctl restart pwm_puma
# systemctl restart pwm_delayed_job
```

Active: active(running)となっていることを確認します。

```
# systemctl status pwm_puma
# systemctl status pwm_delayed_job
```

メール検索設定（3）

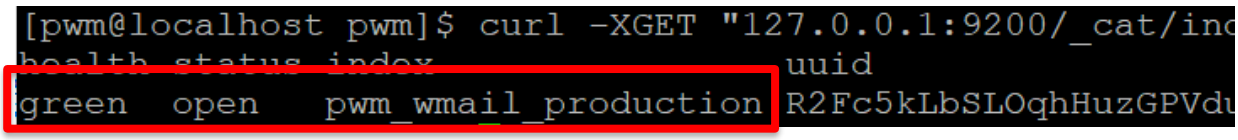
検索インデックスを作成します。
署名・引用削除(update_sanitized_body)タスク、および
インデックスインポート(import_index)タスクを実行すると、
プロセスログ画面から進行状況を確認できます。

```
# su - pwm
$ cd /var/www/pwm
$ bundle exec rake pwm_wmail_search:email:update_sanitized_body
RAILS_ENV=production TARGET_ALL_ACCOUNT_CHECK=1
$ bundle exec rake pwm_wmail_search:search:import_index
RAILS_ENV=production TARGET_ALL_ACCOUNT_CHECK=1 BATCH_SIZE=100
```

インデックスの状況を確認します。
※「xxx.xxx.xxx.xxx」にはelasticsearch.ymlのnetwork.hostと同じ内容を指定します

```
$ $ curl -XGET ”xxx.xxx.xxx.xxx:9200/_cat/indices?v“
```

greenまたはyellowと表示されていれば問題ありません。



ログインの確認

ブラウザで下記のURLにアクセスしログインを確認します。

https://pwm.localdomain.jp/

* ユーザーID: pwm
* パスワード: pwm

画面左上のプルダウンメニューをクリックし、「メール」メニューが表示されていればインストール完了です。