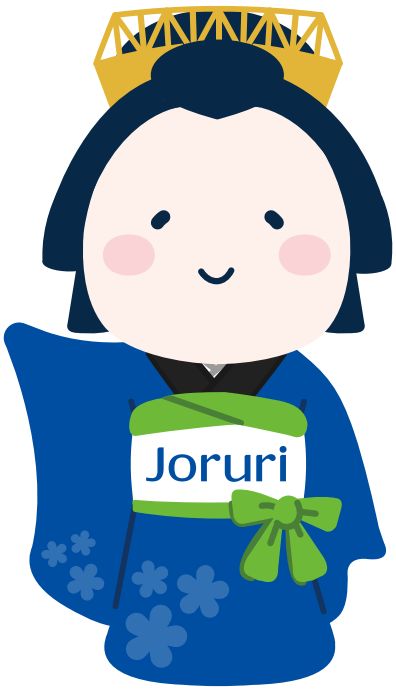
Joruri Mail 2022

JoruriMail2022
Release4
インストール手順



2026-09-25 第四版
サイトブリッジ株式会社

目次

JoruriMail2022は、JoruriPWM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。
JoruriMail2022はRelease3でマルチテナントに対応しました。本手順書ではシングルテナント（primary）のみで稼働させるためのものとして作成しています。

想定環境
- AlmaLinux 9.x（x86_64）
- Nginx 1.30
- PostgreSQL 15
- Valkey 8.0
- Node.js 24
- Ruby 3.4.8
- Rails 8.1.3.1
- jdk25
- Tika 3.3.0

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。

```
rustc 1.98.1 (48a229cea 2026-09-01)
[root@localhost ~]#
```

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 3.4.8
rbenv global 3.4.8
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。

```
ruby 3.4.8 (2025-12-17 revision 995b59f666) +PRISM [x86_64-linux]
[root@localhost ~]#
```

nodenvでNode.jsをインストール

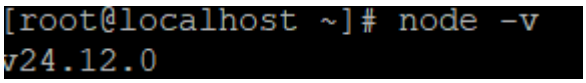
nodenvをインストールします。

```
# 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 24.12.0
# nodenv global 24.12.0
# nodenv rehash
# node -v
```

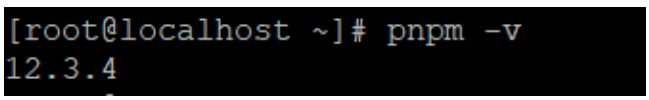
インストールしたnodeのバージョンが表示されます。



pnpmをインストールします。

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

インストールしたpnpmのバージョンが表示されます。



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 stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
---
```

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。

```
[root@localhost ~]# nginx -v
nginx version: nginx/1.30.4
```

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 15.19 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-14), 64-bit
(1 行)
```

Valkeyをインストール

valkeyをインストールします。

dnf -y install valkey hiredis
valkey-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 ~]# valkey-server -v
Valkey server v=8.0.9 sha=00000000:1 malloc=jemalloc-5.3.0 bits=64 build=6913de585e6611e0
```

valkeyを設定します。
vi /etc/valkey/valkey.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。

```
[root@localhost ~]# postconf mail_version
mail_version = 3.5.25
```

JoruriPWM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/joruri-mail-2022-v4.0.0.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 joruri-mail-2022-v4.0.0.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。※¥はバックスラッシュに置き換えてください。

```
$ 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
$ 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
```

インストールしたgemファイルの一覧が表示されれば成功です。

JoruriPWMのインストール（2）

jsライブラリをインストールします。

```
$ pnpm install --production
$ pnpm 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。

```
 $ ./bin/rails 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
```

貼り付けた設定値が出力されるのを確認します。

```
[pwm@localhost pwm]$ ./bin/rails credentials:show --environment production
# aws:
#   access_key_id: 123
#   secret_access_key: 345
secret_key_base:

action_mailbox:
  ingress_password:

pwm:
  secret:
```

JoruriPWMのインストール（3）

DBを作成します。

$ ./bin/rails db:create db:migrate db:seed db:seed:base RAILS_ENV=production
$ ./bin/rails 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。

```
[pwm@localhost pwm]$ ./bin/rails db:version RAILS_ENV=production

database: pwm_production
Current version: 20251121084453
```

アセットをコンパイルします。

$ ./bin/rails 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/**/*

インストールした日付のファイルが存在することを確認します。

周期実行ジョブを登録します。

$ ./bin/rails pwm_core:jobs:schedule RAILS_ENV=production

JoruriPWMのインストール（4）

cronに定期実行処理を追加します。

$ RAILS_ENV=production 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: 2026-0
9-09 10:28:32 +0900
0 0,6,12,18 * * * /bin/bash -l -c 'cd /var/www/pwm && RAILS_ENV=production bundl
e exec rake pwm_core:jobs:schedule --silent'

# End Whenever generated tasks for: /var/www/pwm/config/schedule.rb at: 2026-09-
09 10:28:32 +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

```
[pwm@localhost pwm]$ ./bin/rails console -e production
Loading production environment (Rails 8.1.3.1)
pwm(prod):001>
```

エラーが出力されないことを確認したら、railsコンソールを終了します。

> exit

nginxサンプル設定ファイルをコピーします。

$ cp -r config/samples/nginx/* config/nginx
$ vi config/nginx/pwm.d/primary.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。

サービスの設定

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_include.conf /etc/nginx/conf.d/pwm.conf

ファイルがコピーされていることを確認します。

ls -lh /etc/nginx/conf.d/

```
[root@localhost ~]# ls -lh /etc/nginx/conf.d/
合計 8.0K
-rw-r--r--. 1 root root 1.1K  4月 23 21:58 default.conf.org
-rw-r--r--. 1 root root   44  8月 21 10:04 pwm.conf
```

ログローテートを設定します。

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

ファイルがコピーされていることを確認します。

ls -lh /etc/logrotate.d/

```
[root@localhost ~]# ls -lh /etc/logrotate.d/
合計 40K
-rw-r--r--. 1 root root 130 10月 14  2019 btmp
-rw-r--r--. 1 root root 160 12月  5  2023 chrony
-rw-r--r--. 1 root root  88  9月  9  2022 dnf
-rw-r--r--. 1 root root  93 10月  3  2024 firewalld
-rw-r--r--. 1 root root 162 11月 13  2024 kvm_stat
-rw-r--r--. 1 root root 343  4月 23 21:58 nginx
-rw-r--r--. 1 root root 330  8月 21 10:04 pwm
-rw-r--r--. 1 root root 127  7月 22 01:17 redis
-rw-r--r--. 1 root root 289 11月 12  2024 sssd
-rw-r--r--. 1 root root 145 10月 14  2019 wtmp
```

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)
```

valkeyを起動します。

systemctl enable valkey && systemctl start valkey
systemctl status valkey
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

JoruriMail2022はJoruriPWM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 ~]# 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-25-openjdk java-25-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 ~]# java -version
openjdk version "25.0.4.1" 2026-08-18 LTS
OpenJDK Runtime Environment (Red_Hat-25.0.4.1.1-1) (build 25.0.4.1+1-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-25.0.4.1.1-1) (build 25.0.4.1+1-LTS, mixed mod
e, sharing)
```

古いバージョンが表示される場合、インストールしたバージョンに切り替えます。

```
# update-alternatives --config 'java'
```

```
[root@localhost ~]# update-alternatives --config 'java'

There are 3 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
*  1           java-17-openjdk.x86_64 (/usr/lib/jvm/java-17-openjdk-17.0.20.1.1-1.2.el9.alma.1.x86_64/bin/java)
 + 2           java-21-openjdk.x86_64 (/usr/lib/jvm/java-21-openjdk-21.0.12.1.1-1.2.el9.alma.1.x86_64/bin/java)
   3           java-25-openjdk (/usr/lib/jvm/java-25-openjdk/bin/java)

Enter to keep the current selection[+], or type selection number: 3
```

tika-serverをインストールします。

```
# curl https://archive.apache.org/dist/tika/3.3.0/tika-server-standard-3.3.0.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 ~]# curl http://localhost:9998/tika
This is Tika Server (Apache Tika 3.3.0). 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-9.3.4-
x86_64.rpm
# curl -fsSLO
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-9.3.4-
x86_64.rpm.sha512
# sha512sum -c elasticsearch-9.3.4-x86_64.rpm.sha512
# rpm -ivh elasticsearch-9.3.4-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-1  # 各サーバ―ごとに適宜変更
node.roles: [data, master] # 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
#cluster.initial_master_nodes: [HOSTNAME] # コメントアウト
#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
れているサーバのプライベートIPを指定
cluster.initial_master_nodes: ["node-1"] # 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。

```
---
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
---
```

ICU Analysis Pluginをインストールします。

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

インストール状況を確認します。

```
# bin/elasticsearch-plugin list
```

```
[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
[root@localhost elasticsearch]#  bin/elasticsearch-plugin list
analysis-icu
```

kuromoji(形態素解析プラグイン)をインストールします。

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

インストール状況を確認します。

```
# bin/elasticsearch-plugin list
```

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

Elasticsearchのインストール（3）

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-1"]
---
```

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-1",
  "cluster_name" : "pwm",
  "cluster_uuid" : "lk6dlnQOTDGO0WrF9byFJg",
  "version" : {
    "number" : "9.3.4",
    "build_flavor" : "default",
    "build_type" : "rpm",
    "build_hash" : "69a3e6c50ebb57a1fdbf3f235be9f11061ac7d86",
    "build_date" : "2026-04-22T22:10:58.242616321Z",
    "build_snapshot" : false,
    "lucene_version" : "10.3.2",
    "minimum_wire_compatibility_version" : "8.19.0",
    "minimum_index_compatibility_version" : "8.0.0"
  },
  "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

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

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
# git clone https://github.com/taku910/mecab
# cd mecab/mecab && ./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/mecab/mecab-ipadic && ./configure --with-charset=utf8
&& make && make install
```

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/mecab/mecab/ruby
# 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); #追加
/* create and store the structure pointer to a global variable */
---
```

MeCab-Rubyをインストールします。

```
# 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"
```

```
[root@localhost ruby]# ldconfig -p | grep "/usr/local/lib"
        libmecab.so.2 (libc6,x86-64) => /usr/local/lib/libmecab.so.2
        libmecab.so (libc6,x86-64) => /usr/local/lib/libmecab.so
```

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。

```
# su – pwm
$ cd /var/www/pwm
$ mv Gemfile.engines
/var/www/pwm/config/samples/bundler/Gemfile.engines.bk_yyyymmdd
$ mv /var/www/pwm/config/samples/bundler/Gemfile.mail.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

JoruriMail2022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。

$ ./bin/rails db:migrate RAILS_ENV=production

assetsを更新します。

$ ./bin/rails assets:precompile RAILS_ENV=production

cronを更新します。

$ RAILS_ENV=production bundle exec whenever --update-crontab

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。TENANT変数に各テナント名を指定して作成します。
本手順書はシングルテナントのインストール手順として作成しているため、TENANT=primaryとしています。
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。

Elasticsearchへの接続情報を設定します。

```
# su - pwm
$ cd /var/www/pwm
$ ./bin/rails pwm_elasticclt:setting:change NODE_NAME=localhost
TENANT=primary RAILS_ENV=production
$ ./bin/rails pwm_elasticclt:setting:show RAILS_ENV=production
```

```
[pwm@localhost pwm]$ ./bin/rails pwm_elasticclt:setting:show RAILS_ENV=production
primary: localhost
```

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。

```
$ ./bin/rails pwm_elasticclt:elasticsearch:sample:create_index
TENANT=primary 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greenと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 rep docs.count d
ocs.deleted store.size pri.store.size dataset.size
green  open   pwm_elasticclt_sample_ V2veI2bNSRCyLX7BIME6IA   1   0          0
            0       227b           227b         227b
```

サンプルのインデックスを削除します。

```
$ ./bin/rails pwm_elasticclt:elasticsearch:sample:delete_index
TENANT=primary 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。TENANT変数に各テナント名を指定して作成します。
本手順書はシングルテナントのインストール手順として作成しているため、TENANT=primaryとしています。
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
$ ./bin/rails pwm_wmail_search:search:set_index_name TENANT=primary
RAILS_ENV=production
$ ./bin/rails pwm_wmail_search:search:create_index FORCE=true
TENANT=primary 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greenと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 rep docs.c
ount docs.deleted store.size pri.store.size dataset.size
green  open   pwm_wmail_primary_production UeW2VBDvTuWKlj2FdCJcBw   1   0
   0            0       227b           227b         227b
```

PWMのサービスを再起動します。

pumaを再起動します。

```
# su -
# systemctl restart 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 restart 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。

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
$ ./bin/rails pwm_wmail_search:email:update_sanitized_body
TARGET_ALL_ACCOUNT_CHECK=1 TENANT=primary RAILS_ENV=production
$ ./bin/rails pwm_wmail_search:search:import_index
TARGET_ALL_ACCOUNT_CHECK=1 BATCH_SIZE=100 TENANT=primary
RAILS_ENV=production
```

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。

```
[pwm@localhost pwm]$ curl -XGET "localhost:9200/_cat/indices?v"
health status index                          uuid                   pri rep docs.c
ount docs.deleted store.size pri.store.size dataset.size
green  open   pwm_wmail_primary_production UeW2VBDvTuWKlj2FdCJcBw   1   0
   0            0       227b           227b         227b
```

また、インデックス作成の進行状況は、PWM管理画面の「プロセスログ」から確認できます。

サービスの再起動

pumaを再起動します。

```
$ su -
# systemctl restart 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 restart 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。

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。