

0Z1 JP-LINK Adapter Server インストールガイド

バージョン 1.2



Adapter Server

コンテンツ

はじめに	3
環境条件	3
ネットワーク要件	3
アプリケーションのインストール	3
パッケージのインストール	3
タイムゾーンの設定(任意)	4
アプリケーションの構築と実行	4
アプリケーションの設定と実行	4
SSL サーバー証明書	4
Dockerコンテナの設定	6
Docker コンテナの起動確認	7
ファーストユーザーの追加	8
アプリケーションへのアクセス	8
アプリケーションの管理	9
アプリケーションの更新	9
インストールバージョンを指定してインストールする方法	9
ログの表示	10
アダプターサーバの管理DBのパスワード変更方法	11
インストール後のタイムゾーン変更方法	11
未使用のDockerイメージの削除	11
[TIPS]Dockerコマンド	11

はじめに

本書は、OZ1 Universal JP-LINK Adapter サーバーのインストール、アップデート、設定について説明しています。

インストールは大きく、以下の手順で行います。

- Ubuntuのパッケージアップデート
- Dockerのインストール
- アダプターサーバDockerイメージ取得のためのgitリポジトリクローン
- SSL証明書の設定
- Dockerイメージの構築

環境条件

- オペレーションシステム：[Ubuntu 20.04.3 LTS \(64-bit server\)](#)※セキュリティサーバとは異なりますのでご注意ください
- 推奨ストレージドライブ容量：50GB以上
- 推奨物理メモリ： 4GB RAM以上

ネットワーク要件

- アダプターサーバのユーザーインターフェース接続のためにTCP:443のインバウンド
- セキュリティサーバとアダプターサーバ間の通信のためにTCP:80/8085/8003のインバウンド
- データベースとの接続・通信のためにデータベース毎に設定されたTCPポートのアウトバウンド(PostgreSQLの場合、デフォルトポートは5432です)

アプリケーションのインストール

すべてのコマンドは、sudo ユーザーとして実行する必要があります。sudo モードを有効にするには、Ubuntu のターミナルで以下のコマンドを実行してください。

```
sudo -i
```

パッケージのインストール

最新のパッケージに更新する。

```
apt update
apt upgrade
```

docker/docker-compose/gitをインストールします。

```
apt install docker docker-compose git
```

タイムゾーンの設定(任意)

タイムゾーンの設定を行わなくても、アダプターサーバは稼働しますが、タイムゾーンは、Webアプリケーションやログに正しい時刻を表示するために重要な設定です。
次のコマンドで、利用可能なタイムゾーンを一覧表示し、タイムゾーンを確認してください。

```
timedatectl list-timezones
```

リストをフィルタリングするために、grepを使用することもできます。

例) アジア圏のタイムゾーンのみ表示

```
timedatectl list-timezones | grep Asia
```

タイムゾーンをリストアップされたものから選択して、設定します。

例) 以下はタイムゾーンを日本標準時に設定する場合のコマンドになります。

```
timedatectl set-timezone Asia/Tokyo
```

タイムゾーンが正しく設定されているか確認します。

```
timedatectl
```

“Time zone:” と表示されている行に、設定したタイムゾーンが表示されます。

出力例:

```
$ timedatectl
          Local time: Tue 2022-04-12 11:19:57 JST
          Universal time: Tue 2022-04-12 02:19:57 UTC
             RTC time: Tue 2022-04-12 02:19:56
        Time zone: Asia/Tokyo (JST, +0900)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

アプリケーションの構築と実行

ソースコードや SSL 証明書を格納するディレクトリを作成します。

```
mkdir -p /var/lib/uxa/{git,certs}
```

UXA リポジトリをクローンします。

```
git clone https://UXA:MxVzwHG4Ud1imzsLyk7x@git.aktors.ee/root/UXA.git
/var/lib/uxa/git
```

* この git URL は Universal JP-LINK AdapterServerの git リポジトリへの read アクセスを許可するもので、本書に記載されている用途以外では利用しないでください。

アプリケーションの設定と実行

SSL サーバー証明書

SSL証明書を設定するには、2 つのオプションがあります。

1. [証明書からPKCS12 キーストアを作成する](#)
2. [既存のテスト証明書を使用する](#)

1. 証明書から PKCS12 キーストアを作成する

CAから提供された証明書ファイル、または自己署名証明書を事前に準備してください。

以下のパラメータを置き換えてコマンドを実行すると、PKCS12 keystore ファイルが作成されます。

- <CERTIFICATE> - サーバ証明書ファイルのパス（例：server.crt）
- <CERTIFICATE_KEY> - 証明書キー・ファイルのパス（例：server.key）
- <ALIAS> - 鍵ストアに格納する証明書の名前（任意の値）
- <CA_CRERT> - CAのCrtファイルのパス（例：ca.crt）
- <CA_NAME> - CAの名称（例：root）

このコマンドを実行すると、カレントディレクトリに *keystore.p12* ファイルが作成されます。パスワードの入力を求められますので、パスワードを入力してください。このパスワードは、後続のDocker設定にて<SSL_KEYSTORE_PASSWORD>に設定します。またアップデートの際にも利用するため、失念・紛失しないよう保管ください。

```
openssl pkcs12 -export \  
-in <CERTIFICATE> -inkey <CERTIFICATE_KEY> \  
-out keystore.p12 -name <ALIAS> \  
-CAfile <CA_CERT> -caname <CA_NAME>
```

作成されたPKCS12ファイルを所定のフォルダに移動してください。

```
mv keystore.p12 /var/lib/uxa/certs/keystore.p12
```

PKCS12ファイルの権限を変更し、全ユーザーに読み取り権限を付与してください。

```
chmod a+r /var/lib/uxa/certs/keystore.p12
```

新しく作成したPKCS12 キーストアを使用するには、Docker の設定時に以下のプロパティ値を指定してください。

- <SSL_KEYSTORE> - /var/lib/uxa/certs/keystore.p12
- <SSL_KEYSTORE_ALIAS> - キーストアの作成時に指定したエイリアス名
- <SSL_KEYSTORE_PASSWORD> - キーストアの作成時に入力したパスワード

2. 既存のテスト証明書の使用する

リポジトリ内にある自己署名付きの PKCS12 鍵ストアを使用するには、[Docker の設定](#)時に以下のプロパティ値を指定してください。

- <SSL_KEYSTORE> - /var/lib/uxa/git/backend/src/main/resources/uxa_ssl.p12
- <SSL_KEYSTORE_ALIAS> - uxa_oz1
- <SSL_KEYSTORE_PASSWORD> - uxauxauxa

Dockerコンテナの設定

プレースホルダーを実際の値に置き換えて、コマンドを実行します。

初回のインストールには最大 30 分程度の時間がかかる場合があります。

- <APPLICATION_NAME> - 表示するアプリケーション名を入力します。アダプタサーバのインターフェースへアクセスした際、ブラウザのメニューバーやタブに表示する名前を指定します（例：adServer）。
- <POSTGRES_PASSWORD> - データベースのパスワードを指定します（初めてコマンドを実行したときに、指定したパスワードを持つ新しいデータベースユーザーが作成されます）
- <SSL_KEYSTORE> - SSLキーストアの絶対パス（例：/var/lib/uxa/certs/keystore.p12）
- <SSL_KEYSTORE_ALIAS> - SSLに使用されるキーストアの名前です。
- <SSL_KEYSTORE_PASSWORD> -SSLキーストアのパスワード

```
APPLICATION_NAME=<APPLICATION_NAME> \  
POSTGRES_PASSWORD=<POSTGRES_PASSWORD> \  
SSL_KEYSTORE=<SSL_KEYSTORE> \  
SSL_KEYSTORE_ALIAS=<SSL_KEYSTORE_ALIAS> \  
SSL_KEYSTORE_PASSWORD=<SSL_KEYSTORE_PASSWORD> \  
docker-compose \  
-f /var/lib/uxa/git/docker-compose.yml \  
--env-file /var/lib/uxa/git/.env \  
up -d --build webapp
```

【例】2. 既存のテスト証明書を使用する場合の記載例

```
APPLICATION_NAME=adServer \  
POSTGRES_PASSWORD=uxa \  
SSL_KEYSTORE=/var/lib/uxa/git/backend/src/main/resources/uxa_ssl.p12 \  
SSL_KEYSTORE_ALIAS=uxa_oz1 \  
SSL_KEYSTORE_PASSWORD=uxauxauxa \  
docker-compose \  
-f /var/lib/uxa/git/docker-compose.yml \  
--env-file /var/lib/uxa/git/.env \  
up -d --build webapp
```

*PostgreSQLのパスワードは任意の値に変更可能です。

以下のコマンドで、Dockerコンテナの起動を確認してください。
起動完了までには1～2分程度かかる場合があります。

```
docker ps
```

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	PORTS	NAMES
c3acfdc9c752	uxa_webapp	"java -jar -Duser.ti..."	15	
seconds ago	Up 14 seconds	0.0.0.0:80->8080/tcp, :::80->8080/tcp, 0.0.0.0:443->8443/tcp, :::443->8443/tcp	webapp	
373bad06db90	postgres:12.9-alpine3.15	"docker-entrypoint.s..."	16	
seconds ago	Up 15 seconds (healthy)	0.0.0.0:5432->5432/tcp, :::5432->5432/tcp	database	

```
docker logs webapp
```

```
$ docker logs webapp
```

[illegible]

```
2022-02-16 01:30:33.115 INFO 1 --- [main]
ee.aktors.uxa.UxaApplication : Starting UxaApplication on
c3acfdc9c752 with PID 1 (/uxa.jar started by spring in /)
2022-02-16 01:30:33.130 INFO 1 --- [main]
ee.aktors.uxa.UxaApplication : The following profiles are
active: prod
2022-02-16 01:31:32.610 INFO 1 --- [main]
ee.aktors.uxa.UxaApplication : Started UxaApplication in
62.448 seconds (JVM running for 65.272)
2022-02-16 01:31:32.613 INFO 1 --- [main]
e.a.u.c.bootstrap.HttpsConfiguration : Initializing HTTPS
configuration.
2022-02-16 01:31:32.613 INFO 1 --- [main]
e.a.u.c.bootstrap.HttpsConfiguration : Initializing a socket factory
that trusts all HTTPS certificates.
```

ファーストユーザーの追加

アプリケーションサーバーが起動したことを確認したら、ターミナルで以下のコマンドを実行し、最初のユーザーを追加します。

<USERNAME>と<PASSWORD>を実際の値に置き換えて、以下のコマンドを実行します。

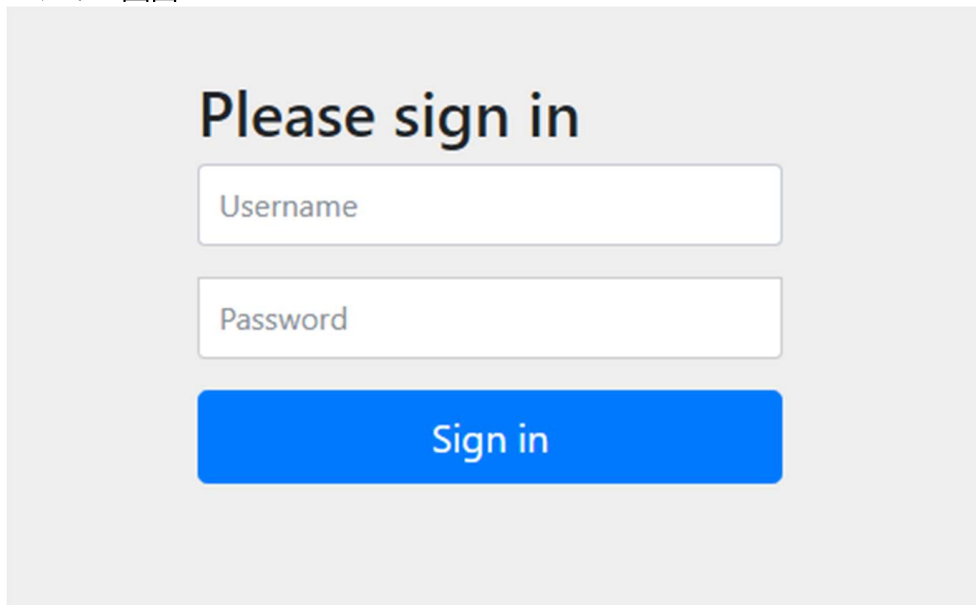
```
sudo docker exec -it webapp curl -k -X PUT -w '\n' \
https://localhost:8443/api/local/user/save --header \
'Content-Type:application/json' --data-raw \
'{"username": "<USERNAME>", "password": "<PASSWORD>"}
```

ユーザーの追加に成功した場合は、"Success!"と表示されます。

アプリケーションへのアクセス

これで、アプリケーションは [<https://<IPADDRESS>/>] という URL からアクセスできるようになりました。<IPADDRESS>は、アプリケーションが動作しているサーバーの IP アドレスです。[<https://<IPADDRESS>/>]をWebブラウザに入力し、実行するとログイン画面が表示されます。

ログイン画面



「ファーストユーザーの追加」で作成したユーザー名とパスワードでログインしてください。

アプリケーションが稼働しているかを確認する場合、下記コマンドを実行ください

```
curl -v -X GET -k https://localhost/login
```

HTMLが表示されれば成功です。何らかのエラーメッセージが表示されている場合には、エラーメッセージの内容を確認の上、ここまで手順を見直してみてください。

以上でインストール作業が終了です。

Adapter Serverの設定方法や操作については、Adapter Serverユーザーガイドを参照ください。

アプリケーションの管理

アプリケーションの更新

git から更新情報を取得します。

```
sudo git -C /var/lib/uxa/git pull
```

プレースホルダーを実際の値に置き換え、コマンドで Docker コンテナを更新します（root 権限で実行する必要があります）。

- <APPLICATION_NAME> - 表示するアプリケーション名を入力します。アダプターサーバへアクセスした際、ブラウザのメニューバーやタブに表示する名前を指定します（例：adServer）。
- <POSTGRES_PASSWORD> - [インストール](#)時に設定されたデータベースのパスワードです。
- <SSL_KEYSTORE> - SSLキーストアの絶対パス（例：/var/lib/uxa/certs/keystore.p12）
- <SSL_KEYSTORE_ALIAS> - SSLに使用されるキーストアの名前です。
- <SSL_KEYSTORE_PASSWORD> - SSLキーストアのパスワード

```
APPLICATION_NAME=<APPLICATION_NAME> \  
POSTGRES_PASSWORD=<POSTGRES_PASSWORD> \  
SSL_KEYSTORE=<SSL_KEYSTORE> \  
SSL_KEYSTORE_ALIAS=<SSL_KEYSTORE_ALIAS> \  
SSL_KEYSTORE_PASSWORD=<SSL_KEYSTORE_PASSWORD> \  
docker-compose \  
-f /var/lib/uxa/git/docker-compose.yml \  
--env-file /var/lib/uxa/git/.env \  
up -d --build webapp
```

特定のバージョンをインストールする

Gitコマンドにより、利用可能なタグを検索し、所定のバージョンのリソースを取得することで、特定のバージョンのアダプターサーバを利用可能です（root権限で実行する必要があります）。

特定のバージョンのアプリケーションを使用するには、そのバージョンのソースコードを git から取得する必要があります。アプリケーションのビルドと実行で説明したように、まずリポジトリをクローンします（まだクローンしていない場合）。

※強い理由がない限りは、最新のバージョンのアダプターサーバの利用を推奨しております。

gitから更新情報を取得します

```
git -C /var/lib/uxa/git pull
```

利用可能なバージョンを表示するには、以下のコマンドを使用します。

```
git -C /var/lib/uxa/git tag -l -n
```

※タグの一覧を表示した場合、V1.0以前の状態も表示される場合があります。しかし、これらの リリース以前のベータバージョンは利用しないようお願いします。

リポジトリの状態を所定のタグのバージョンに切り替えます。

<TAG_NAME> - インストールしたいバージョンのタグ名

```
git -C /var/lib/uxa/git checkout tags/<TAG_NAME>
```

上記コマンドで指定したバージョンに切り替わっていることを確認します。

```
git -C /var/lib/uxa/git status
```

※ “HEAD detached at <TAG_NAME>” というようなメッセージが表示される筈です

この後は[アプリケーションの設定と実行]のセクションから再開してください。
ただし、git関連の操作は既に完了しているため、実行せず、スキップしてください。

特定のバージョン指定インストール後、最新のバージョンに差し戻す

バージョンを指定した方法でインストールした場合、最新のバージョンへ差し戻す場合は以下の手順に従ってください（root権限で実行する必要があります）。

Gitからmasterをチェックアウトします。

```
git -C /var/lib/uxa/git checkout master
```

masterに切り替わっていることを確認します。

```
git -C /var/lib/uxa/git status
```

※ “On branch master. Your branch is up to date with 'origin/master'.” というようなメッセージが表示される筈です

この後は[アプリケーションの設定と実行]のセクションから再開してください。
ただし、git関連の操作は既に完了しているため、実行せず、スキップしてください。

ログの表示

root に切り替える。

```
sudo -i
```

ログは以下のディレクトリに保存されます。

```
cd /var/lib/docker/containers/${docker inspect --format="{{.Id}}" webapp}
```

最新のログファイルは<container-id>.json.log という名称で保存されています。

<container-id>は重要な意味を持たないランダムな英数字です。

古いログファイルは圧縮され、<container-id>.json.log.<order-nr>.gz という名前が付けられます。

<order-nr>が小さいほど、最新のアーカイブされたログを示します。<order-nr>が大きいほど、ログが古いことを示します。

アダプターサーバの管理DBのパスワード変更方法

データベースのパスワードを変更する必要がある場合には次の方法でパスワードの変更をおこなってください。

1. databaseコンテナでデータベースに接続する。

```
sudo docker exec -it database psql -U uxa
```

2. デフォルトユーザー "uxa"のパスワードを変更する。

<NEW_POSTGRESQL_PASSWORD> - 変更後のパスワードを指定してください

```
ALTER USER uxa WITH PASSWORD '<NEW_POSTGRESQL_PASSWORD>';
```

インストール後のタイムゾーン変更方法

インストール完了後にタイムゾーンの設定を行う場合には、すでに作成されたコンテナを削除する必要があります。

これは既に保存されたデータ（データソース、サブシステム、サービス、ユーザー等）を削除するものではありません。

稼働中のコンテナを停止し、コンテナを削除します。

```
docker stop database webapp liquibase  
docker rm database webapp liquibase
```

コンテナ削除後、[Dockerコンテナの構築コマンド](#)を再実行します。

次にデータベースのタイムゾーンを変更するため、databaseコンテナのPostgreSQLへアクセスします。

```
sudo docker exec -it database psql -U uxa <POSTGRES_PASSWORD>
```

PostgreSQLのタイムゾーンの設定を変更します。

設定するタイムゾーンはサーバに設定したタイムゾーンと同じタイムゾーンとしてください。

例）以下はタイムゾーンを日本標準時に設定する場合のコマンドになります。

```
SET TIMEZONE='Asia/Tokyo';
```

タイムゾーンが正しく設定されたことを確認します。

```
SHOW TIMEZONE;
```

未使用のDockerイメージの削除

アプリケーションの新しいイメージを構築した後、古いDockerイメージはディスク上に残り、ディスクスペースを占有してしまいます。

ディスクスペースがひっ迫している場合などは、以下のコマンドで未使用のDockerイメージを削除してください。

```
sudo docker image prune -a
```

[TIPS]Dockerコマンド

Dockerコンテナの操作にあたって、よく使われるコマンドの一例

※Dockerコマンドの実行にあたってはsudoが必要になります

- docker ps - 現在実行中のコンテナ一覧を表示します。
- docker start <コンテナ名> - コンテナを起動します。
- docker stop <コンテナ名> - コンテナを停止します。
- docker logs <コンテナ名> - コンテナのログを表示します。

- `docker exec -it <コンテナ名> bash` - コンテナ端末にアクセスします。
- `docker images` - Dockerに現在登録されているイメージファイルの一覧です。過去

現在定義されているコンテナは次の通りです

- `database`: アダプターサーバの各種設定情報が格納されたDBコンテナです。
- `webapp`: アダプターサーバ本体が稼働しているコンテナです。
- `Liquibase`: DBバージョン管理ツール。