`studylog/api` Docker Deployment Strategy

Install Docker

Mac & PC

Docker Desktop is the recommended solution for individual workstations and servers running Windows or MacOS.

We recommend following the official installation instructions on the Docker website.

> ⚠ *Instructions and information about the latest version of Docker Desktop can be found at:*
>
> *For Windows (https://docs.docker.com/desktop/windows/install/)*
>
> *For Mac (https://docs.docker.com/desktop/mac/install/)*

Linux

Docker daemon is the recommended solution for *nix machines.

We recommend following the official installation instructions on the Docker website.

> ⚠ *Instructions and information about the latest version of Docker for Linux can be found at:*
> *Docker for Linux (https://docs.docker.com/desktop/linux/install/)*

Install the AWS Command Line Interface

The AWS command-line interface is used in the deployment process for authenticating your docker installation to our docker container registry.

> ⚠ *Instructions and information about the latest version of AWS Command Line Interface can be found at:*
> *AWS CLI (https://aws.amazon.com/cli/)*

Acquiring AWS Credentials

Before you are able to download docker images from our private container registry, you will need to authenticate.

For your convenience, we offer the following options for authentication:

With Your AWS Account

If you would like to use your own AWS account to access our container registry, please provide your Studyog support representative with your AWS Account ID.

> ⚠ *More information about finding your AWS Account ID can be found at: AWS Account ID (https://docs.aws.amazon.com/IAM/latest/UserGuide/console_account-alias.html)*

With Studylog Credentials

If you do not have an AWS account, Studylog can create an AWS IAM user for you. Please submit a request to your Studylog support representative to have have these credentials created.

Once your AWS IAM user is created, you will receive an AWS Access Key ID, and AWS Secret Key from your Studylog support representative.

Container Registry Authentication

AWS CLI Configuration

The following command will configure your AWS CLI.

Once this step is completed, future **aws** commands will run as the configured account.

```
aws configure
```

> ⚠ *Instructions and information about authenticating AWS Command Line Interface can be found at:*
> *AWS CLI Authentication (https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html)*

Private AWS Registry Authentication

The following command will use the **`aws ecr`** command to download docker credentials, and then pass those credentials to the **`docker login`** command.
Once this step is completed, the docker daemon will be authenticated for one month.

```
aws ecr get-login-password --region us-west-1 | docker login --username AWS --password-stdin
823477618550.dkr.ecr.us-west-1.amazonaws.com
```

> ⚠ *Instructions and information about authenticating Private registry authentication can be found at:*
> *AWS ECR Authentication (https://docs.aws.amazon.com/AmazonECR/latest/userguide/registry_auth.html)*

Pulling Docker Images

The following command will download the latest **`studylog/api`** docker image from our container registry.

```
docker pull 823477618550.dkr.ecr.us-west-1.amazonaws.com/studylog/api:latest
```

> ⚠ *Information about pulling docker images can be found at:*
> ***`docker pull`*** *(https://docs.docker.com/engine/reference/commandline/pull/)*

Deploying Docker Images

Docker Container Variables

When you create a docker container, you can use the `--env-file` argument to specify the path to a custom environment file for your container.

As an example:

```
docker run -d --rm \
  --name studylog-api \
  --env-file /path/to/your/file.env \
  823477618550.dkr.ecr.us-west-1.amazonaws.com/studylog/api:latest
```

> ⚠ *Information about running docker images can be found at:*
> ***`docker run`*** *(https://docs.docker.com/engine/reference/commandline/run/)*

Environment Variables

Required Variables

Please modify and add the following environment variables to your docker container, replacing the **`<default values>`** with the correct ones for your specific installation.

```
APP_MODELS_VERSION=<423>

APP_JWT_SECRET=<long string here>
APP_SESSION_SECRET=<long string here>

APP_DB_HOST=<database host>
APP_DB_NAME=<database name>
APP_DB_USER=<database user>
APP_DB_PASS=<database pass>
```

Optional Variables

```
APP_HOST=<0.0.0.0>
APP_PORT=<3000>
APP_ROOT=<http://localhost:3000>

APP_PATH=</api>

APP_ENABLE_CORS=<false>
APP_MAX_REQUEST_SIZE=<100MB>
APP_JWT_EXPIRATION=<1d>

APP_DB_PORT=<1433>
APP_DB_SCHEMA=<dbo>
```

Database Encryption

If your mssql database is configured to support encryption, you can enable it with the following environment variables:

```
APP_DB_ENCRYPT="<true>"
APP_DB_TRUST_CERT="<true>"
```

SSO/SAML Authentication

If you want to use SAML to authenticate your users, you can enable it with the following environment variables:

```
APP_SAML_SP_REDIRECT="<http://localhost:3000>/api/auth/saml/redirect?"
APP_SAML_SP_ASSERT_ENDPOINT="<http://localhost:3000>/api/auth/saml/assert"

APP_SAML_IDP_SSO_LOGIN_URL="<saml_sso_assert_url>"
APP_SAML_IDP_SSO_LOGOUT_URL="<saml_sso_logout_url>"
APP_SAML_IDP_RESPONSE_EMAIL_ATTR="<attributes.email.0>"

APP_SAML_SP_CERTIFICATE="</path/to/docker/volume/server.crt>"
APP_SAML_SP_PRIVATE_KEY="</path/to/docker/volume/server.pem>"
APP_SAML_IDP_CERTIFICATE="</path/to/docker/volume/server.crt>"
```