Blog

Installation manual

To install the extension follow the instructions below:

1) Backup your web directory and store database

- Get the Blog name and version in your Marketplace account:
- Login to Magento Marketplace with the username and password you used to purchase the Blog extension;
- Enter the My Purchases section: **Your name (top-right corner of the page) > My Profile > My Purchases;**
- Find Blog and click **Technical details.**

2) Copy the provided component name and version in order to update your composer.json file.
Note: *We advise you to use the latest version of our extension.*

3) Update your composer.json file:

- Open the root Magento directory on your server and send Composer the following command:

```
Composer require <Blog>:<version>
```

4) At the Composer request, enter your Magento marketplace credentials (public key - username, private key - password).

5) Make sure that Composer finished the installation without errors.

6) Flush store cache, log out and log in the backend again.
The Blog extension is now installed and ready for work. Use Magento Dev Docs for more details.

PWA Studio for Magento 2

For getting PWA Studio free add-on please contact awsupport@aheadworks.com.

Pre-Installation Requirements

Make sure, the following requirements are met prior to the installation of PWA:

- A *Linux* kernel operating system (OS). Ubuntu distributions are preferred.

  Windows OS is not supported.

- NodeJS (version 10.14.1 and higher) and Yarn (version 1.13.0 and higher), preinstalled on the OS.
- Magento (version 2.4.0-2.4.1), preinstalled on the Server. Access via https enabled.
- Aheadworks Blog (version 2.8.0) and Aheadworks BlogGraphQL extensions, pre-installed on the Server. Though part of the Aheadworks Blog installation package, BlogGraphQl is installed separately.

Installing and Configuring PWA Studio

1. Use the following command to install PWA Studio:

```
yarn create @magento/pwa
```

2. Enter any name (no spaces) in response to the "**Project root directory"** query, For example, "*Blog*". This will create a directory to host all the files of the installation.

3. Choose "*Other*" for the "**Magento instance to use as a backend"** query

4. Enter the address of the Server in response to the **"URL of a Magento to use as a backend"** query. The address must start https://

5. Leave default replies for all the remaining queries.

Reply "y" to the **"Install package dependencies with yarn..."**query.

6. Go to the project folder:

```
cd <project_name>
```

7. Run the following command:

```
yarn run buildpack create-custom-origin .
```

The command is to end with a space and dot.
This will ensure that no error will be ignored on running the project.

8. Go to the project root directory and find the *.env* file. Open the file and change the value of the IMAGE_OPTIMIZING_ORIGIN parameter from "*auto*" into "*backend*".

Employing Aheadworks Blog

1. Copy the */awBlog/* folder into */src/*

2. Go to the project root directory and find the *.local-intercept.js* file. Insert the following code into the file:

```
module.exports = targets => {
    targets.of('@magento/venia-ui').routes.tap(routes => {
        routes.push({
            name: 'Blog',
            pattern: '/blog/:name?/:key?',
            path: require.resolve('./src/awBlog/components/Blog/blog.js')
        });
        return routes;
    });
};
```

3. Add the following line to */src/store.js*

```
import { awReducerBlog } from './awBlog/redux/reducers/awReducerBlog';
```

4. Change the following line:

```
const rootReducer = combineReducers(reducers);
```

To

```
const rootReducer = combineReducers({...reducers, awReducerBlog});
```

Running PWA Studio Project

Run the following command to test the Development version:

```
yarn run watch
```

This will start the Development Server. See the console output for the address of the server.
Run the following command to test the Production version:

```
yarn build && yarn start
```

This will start the Production Server. See the console output for the address of the server.

Important Notes on Compatibility with PWA

PWA addon to M2 Blog from Aheadworks works independently from your M2 Blog from Aheadworks. The add-on fetches data from the main extension. It is important to note, that the following data is not fetched:

- Related posts on product pages
- Disqus comments (not supported by PWA)
- Category descriptions
- Authors

It is furthermore essential to know the following:

- Posts imported from WordPress may not be displayed correctly
- Custom styles employed to posts may not be displayed correctly
- *Blog_title* is not used
- The route to blog is not subject to change (the route to blog is to be known before the build of the project)

The above nuances may well be solved within the upcoming versions of M2 Blog from Aheadworks (higher than 2.8.0) and M2 PWA Studio (higher than 8.x).