The ultimate Magento 2 build & deploy commands workflow

1 | Download code

- Clone Git repository and all its submodules. Then, copy Magento 2 code from the Git repository `($CODE_ROOT)` to the web server document root.

2 | Run composer

- If `composer.lock` file is present, we run the composer install command without dev dependencies. Each build is isolated, so we don't use cache.
- If you need credentials, add an `auth.json` file at the same location as the `composer.lock` file.

3 | Compile code

- Compile all Magento files and classes into the generated directory.

4 | Dump an optimized composer autoloader

- The `-optimize` option converts PSR-0/4 autoloading to classmap to get a faster autoloader, which is recommended, especially in production.

5 | Deploy static view files

- Write all static files to the Magento file system.

6 | Mount persistent directories

- Share persistent data and directories across all these web servers. directories to make sure your production environment is highly resilient.

7 | Define the `app/etc/env.php` configuration file

- Copy the Magento `app/etc/env.php` configuration file related to the current environment `($ENV_CODE)` . Use as many environment variables as possible to make your release flexible and reliable.

8 | Enable maintenance mode

- In order to avoid downtime, we put Magento under maintenance only if the new release needs to update configuration and/or database schema/data.

9 | Update database

- Magento 2 uses a database schema and data versioning. Make sure to upgrade modules, if needed.

10 | Update configuration

- Make sure to update the Magento 2 configuration if you change the files `env.php` or `config.php`

11 | Disable maintenance mode

- Make sure to disable `magento maintenance` if it was already enabled.

artifakt