Adobe Commerce Cloud

Seamlessly merge commerce with customer experience.

6 tips and tricks to enhance the creation of your ecommerce experiences.

Adobe Commerce Cloud provides merchants the e-commerce platform and business intelligence they need to transform their business and address the full spectrum of commerce challenges.

David Young
Business Solutions Architect,
Adobe Commerce Cloud

David Young is a business solutions architect. He played an instrumental part in developing and growing the e-commerce and marketing departments for a retailer and later took a role as a lead developer at an agency where he became a Magento certified developer and solutions specialist. David leverages his experience and expertise to envision and develop the future for his clients in Adobe Commerce Cloud.

With the benefit of his real-world insight, we've curated the following tips and tricks to simplify and enhance the development and maintenance of your e-commerce cloud platform.

Tip 1: Perform day-to-day development in your own external repository.

Adobe Commerce Cloud provides a GitHub repository for triggering deployments in your environment. However, this repository typically shouldn't be used for day-to-day development. Instead, development activities should be performed in your own external Git or Bitbucket repository. Adobe Commerce Cloud allows you to integrate your external repository with the Adobe provided Git repository and automatically synchronize events between them.

The specific steps and order for performing an integration vary between Bitbucket and GitHub. But generally speaking, you will need to create your external repository by cloning your Adobe Commerce Cloud project from an existing environment and migrating the project branches to a new empty external repository, making sure you preserve the same branch names. You will also need to create a security token that gives you admin access to your repository and write access for pull requests and webhooks. Next, based on your respective environment, you need to enable the integration following the configuration steps specific to Bitbucket or GitHub. After that, add a webhook to your Bitbucket or GitHub repository and then verify that the integration works by creating a test file.

Learn more ›

Important to know: *The only branches that should exist in the Git repository when the integration is initially activated are those from the original Cloud repository. The names of the branches in the external repository must also be identical to those in the original Cloud repository (note that these names are case-sensitive).*

Helpful hint: *More details on repository integration, as well as thorough guidance on other aspects of building your e-commerce platform, can be found in our "Best Practices Guide: Developing & Maintaining your Cloud Project".*

Important to know: *The person setting up your integration must have "admin" access levels for your repository.*

Tip 2: Make sure you use and have the latest version of ece-tools installed.

The metapackage for Adobe Commerce Cloud comes with the ece-tools package. This facilitates the management of your cloud project's infrastructure and code deployment. It gives you a rich set of management features, including wizards to help you configure your environment according to best practices, scripts and commands to manage your code and automatically build and deploy your projects, and direct integrations with your local cloud docker setup to simplify management of services, environments, and deployments.

Since ece-tools provides the essential functionality needed to simplify and effectively deploy and manage your Adobe Commerce Cloud project, make sure it's installed and up-to-date. You can quickly check by running "`cat composer.lock | grep "ece-tools"`" from the root of your Adobe Commerce Cloud project. If you need to upgrade to a more recent version of ece-tools, first make sure that you have a clean working tree and execute the following commands from the root of your local development environment: `composer update magento/ece-tools and git add -A && git commit -m "Update magento/ece-tools" && git push origin <branch name>.`

Learn more ›

Helpful hint: *You can install the Magento Cloud CLI utility to locally manage the environments for your Adobe Commerce Cloud projects in your local repository. More details are available in the "Manage branches with the CLI" section of the online documentation.*

Tip 3: Change static content deployment from deploy phase to build phase.

The static content deployment phase is one of the longest and most resource-consuming processes in deploying your store. It involves assembling all the template files, JavaScript, and theme images needed to render the front end of your site. By default, this occurs during the deploy phase when, unfortunately, all the connections to your website are held until the deploy phase ends. If you instead assemble all your static content as part of the build phase, those tasks no longer happen during deploy, significantly reducing your site downtime.

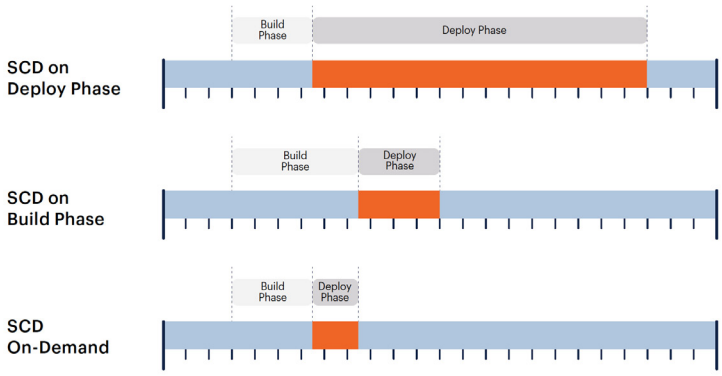
To allow static content deployment to take place during the build phase, you simply need to update your config.php file with the CLI utility by using the command `php vendor/bin/m2-ece-scd-dump` (**Recommended**: Exports only modified configuration settings) or `php ./vendor/bin/ece-tools config:dump` (Exports every configuration setting, including modified and default settings). .

Learn more ›

Good to know: *You also have the option to deploy static content on-demand, which reduces site downtime even more. But in most cases this option is not recommended since the first users to access that content can experience significant response lags.*

Helpful hint: *Other ways you can decrease deployment downtime include enabling the following: HTML minification on the fly, compact SCD strategy, and post deploy hook.*

Deployment time with SCD on different phases.



Tip 4: Use the Ideal State Wizard before deployment to ensure best practice configurations.

One of the smart wizards included with ece-tools is the Ideal State Wizard, which tells you whether configuration settings are in the ideal state for deployment. For example, it checks your static content generation settings, HTML minification settings, and the presence of post_deploy hooks. Then if any of those settings are not in the ideal state, it gives you feedback and direction on how to optimize your configuration. To run the Ideal State Wizard, SSH into your cloud environment and execute the command `./vendor/bin/ece-tools wizard:ideal-state` from your project's root.

Learn more ›

Helpful hint: *The other smart wizards ece-tools let you check your database load balancing configurations and whether your static content deployment is set for on-demand, the build stage, or the deploy stage.*

Tip 5: Verify your configuration files and branching strategy before deployment.

It's important to understand that the configuration files for your deployment are housed in your project environment and will determine what happens when you trigger your deployment. Before you deploy your project, verify that you have a valid branching strategy in your development environment and that you've set up your GitHub integration properly. For example, you should know which of your branches are active and tied to environments and have a process for closing out non-active branches. To help you create healthy development and deployment workflows, read our guidance for Pro and Starter environments.

Also, if you're having issues with a service that didn't initiate correctly or has some aspect that isn't working properly, you can do a container refresh by pushing an empty commit. Normally, when you push a change it triggers a redeployment of your environment. But an empty commit only rebuilds the container. To trigger a container refresh with an empty commit, use the command `git commit --allow-empty -m "redeploy" && git push <BRANCH_NAME>`.

Learn more ›

Important to know: *Deployments are synchronous. So, when you push changes separately, they'll queue up in linear fashion. That makes it important to prioritize your changes, since you don't want less urgent changes to delay more urgent ones from being deployed.*

Tip 6: Know the location of your applications and log files.

Since the Adobe Commerce Cloud server environment might not be structured the same as other hosting providers, when you drop into the root of your project you might not know where to get your applications and system logs. Depending on your connected environment, the paths vary slightly.

In the integration environment, the following paths are used:

Application: `/app`
Staging: `/app/<project_id>_stg/`
Production: `/app/<project_id>/`

The location for standard Magento logs and cloud-specific logs is relative to the project root. For example, the following lobs are found in the following locations:

Environment logs in integration environments: `/app/var/log`
Error reports in integration environments: `/app/var/report`
System logs: `/var/log` (relative to the system root)
Troubleshooting log files for both build and deploy errors: `~/var/log/cloud.log` and `~/var/log/install_upgrade.log`

Learn more ›

Helpful hint: *You can use the CLI utility to pull logs without having to log into the server. Learn more about this capability in the Project Structure — Logs section of the developer documentation.*

Important to know: *Using the Fastly CDN is mandatory for production staging. If you're not familiar with this service, refer to the Fastly section of our Best Practices Guide and our developer documentation on Fastly troubleshooting.*

Important to know: *Don't use the "`cacheable=false`" block attribute to prevent the caching and redisplay of customer information to the wrong user. It will disable caching for the entire page. Instead, handle it as private content on the client-side using our customer-data JS library to store private data in local storage.*

Got unanswered questions? We've got unlimited resources.

Visit our Learn & Support page, professional services, or adobe.com to learn more about how to use helpful features within Adobe Commerce Cloud. You can also access tailored learning paths, community forums, and feature request forms in Experience League.