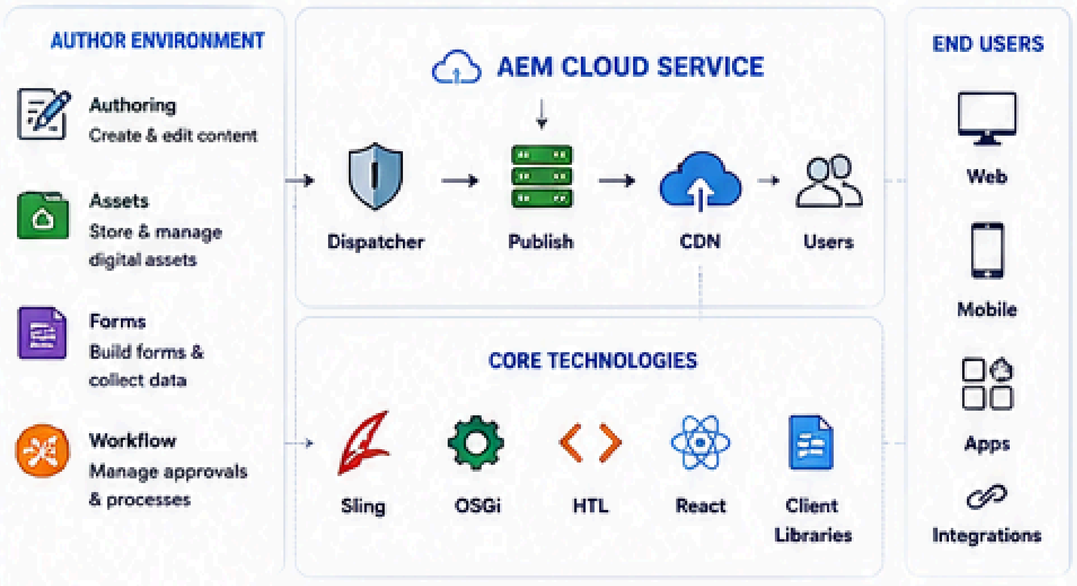
AEM CLOUD SERVICE
DEVELOPER CHEAT SHEET
THE COMPLETE QUICK REFERENCE

Adobe Experience Manager

1. THE BIG PICTURE

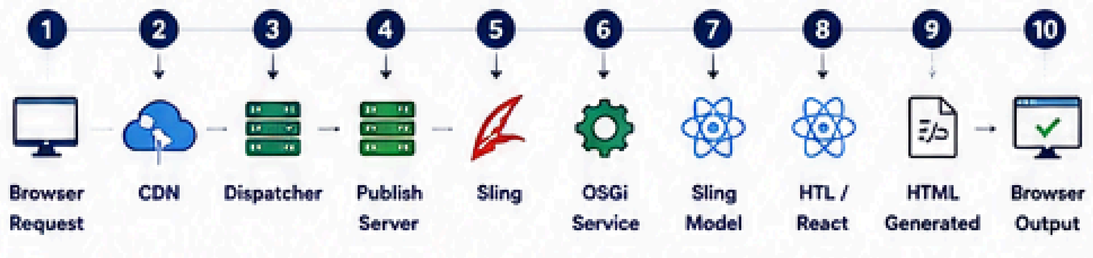
AEM is a complete digital experience platform.



Content is authored on Author, published on Publish, cached by CDN, and delivered to users through Dispatcher with security and performance.

2. THE REQUEST LIFECYCLE (HIGH LEVEL)

From browser to final HTML.



WHAT HAPPENS?

1. User requests a URL in the browser.
2. CDN checks cache for the content.
3. Dispatcher filters the request.
4. Request reaches AEM Publish.
5. Sling resolves the resource.
6. OSGi services provide business logic.
7. Sling Models adapt data for the view.
8. HTL (or React) renders the HTML.
9. HTML is returned and cached.
10. Browser renders the final page.

TIP: Performance and good architecture depend on how well each layer plays its part in this lifecycle.

3. CORE TECHNOLOGIES – WHAT THEY DO

TECHNOLOGY	PRIMARY RESPONSIBILITY
Author (AEM Author)	Create, edit, manage content and assets.
Publish (AEM Publish)	Delivers content to users. No authoring.
Dispatcher	Security, caching, URL filtering, load balancing.
Sling	Request routing, resource resolution.
OSGi Services	Business logic, integrations, reusable services.
Sling Models	Adapt content to Java objects for the view.
HTL (Sightly)	Secure server-side rendering of HTML.
React	Build interactive, dynamic client-side experiences.
Client Libraries	Manage CSS, JavaScript, fonts, and dependencies.
Content Fragments	Structured, reusable content for any channel.
Cloud Manager	Build, test, and deploy to Cloud Service.

4. WHERE DOES MY CODE GO?

TASK / REQUIREMENT	BEST PLACE
Read authored content	Sling Model
Call external API / Integration	OSGi Service
Business logic / Rules / Calculations	OSGi Service
Render HTML on server	HTL (Sightly)
Interactive / Rich UI	React Component
CSS, JS, Images, Fonts	Client Library
Aathor configuration / Dialogs	Component Dialog
Store content / Structure	JCR Repository

5. DISPATCHER AT A GLANCE

WHAT IT DOES

- Caches content for faster response
- Filters and blocks unwanted requests
- Protects Publish from direct access
- Distributes load across instances
- Compresses and optimizes responses

CACHE FLOW

CACHE HIT

CACHE MISS

IMPORTANT FILES

- dispatcher.any – Main configuration
- filters.any – Request filtering rules
- cache.any – Caching rules
- vhosts/*.any – Virtual hules
- rewrite.any – URL rewriting rules

6. HTL VS REACT

	HTL (SERVER-SIDE)	REACT (CLIENT-SIDE)
Purpose	Build structure & content	Add interactivity & dynamic behavior
Where it runs	AEM Publish (server)	Browser (client)
Performance	Fast first paint, SEO friendly	Dynamic updates, rich UX
SEO	Excellent	Good (with SSR / pre-rendering)
Best for	Content, pages, SEO, simple UI	Search, dashboards, filters, apps
Example	Page layout, text, images, forms	Product search, filters, wizards

7. DEBUGGING FLOW (OUTSIDE IN)

1. **Browser** – Check URL, Console, Network
2. **CDN** – Check cache status, headers
3. **Dispatcher** – Check dispatcher.log, rules
4. **Publish** – Check error.log, access.log
5. **Sling** – Resource resolution, selectors
6. **Sling Model** – Check adaptation & data
7. **OSGi Service** – Check service status, logs
8. **HTL / React** – Check rendering, console
9. **Browser Output** – Verify final HTML, JS, CSS

Always debug from the outside in.
Don't guess. Find the layer.

8. CLOUD MANAGER & CI/CD PIPELINE

PIPELINE STEPS

ENVIRONMENTS

Development (dev)
- For developers
- Build & unit test
- Not accessible to end users

Staging (stage)
- Pre-production
- QA, UAT, Integration
- Mirrors production configuration

Production (prod)
- Live environment
- Scalable & secure
- Accessible to end users

Automated · Consistent · Secure · Scalable

9. REMEMBER THESE RULES

- Keep Sling Models thin – only adapt and prepare data.
- Put all business logic in OSGi Services.
- Never call external APIs from HTL or React directly.
- Use HTL for SEO-critical and static content.
- Use React only when interactivity is required.
- Organize Client Libraries and minimize dependencies.
- Follow accessibility (a11y) and performance best practices.
- Cache everything that can be cached.
- Test in Staging as close to Production as possible.

10. COMMON MISTAKES

- Putting business logic in Sling Models or HTL.
- Calling external APIs from HTL / React directly.
- Creating large monolithic React components.
- Duplicate logic between HTL and React.
- Ignoring caching and Dispatcher rules.
- Not testing in Staging before Production.
- Hardcoding paths and configurations.
- Wrong use of Client Libraries (global pollution).
- Not following accessibility and performance rules.

QUICK REFERENCE – KEY FILES & LOGS

KEY FILES

- **/apps** – Application code
- **/content** – Content repository
- **dispatcher.any** – Dispatcher config
- **filters.any** – Filtering rules
- **cache.any** – Cache rules
- **cloudmanager.yaml** – Pipeline config
- **ui.apps** – UI components
- **ui.config** – OSGi configurations

IMPORTANT LOGS

- **error.log** – Errors
- **access.log** – Access requests
- **dispatcher.log** – Dispatcher
- **request.log** – Request details
- **audit.log** – Author actions
- **workflow.log** – Workflows

PRO TIP

Understand the architecture. Follow the flow.
Write clean, secure, and maintainable code. Your future self will thank you.

FREE DOWNLOAD

Get the printable AEM Cloud Service Cheat Sheet (One Page PDF) at the end of this guide.