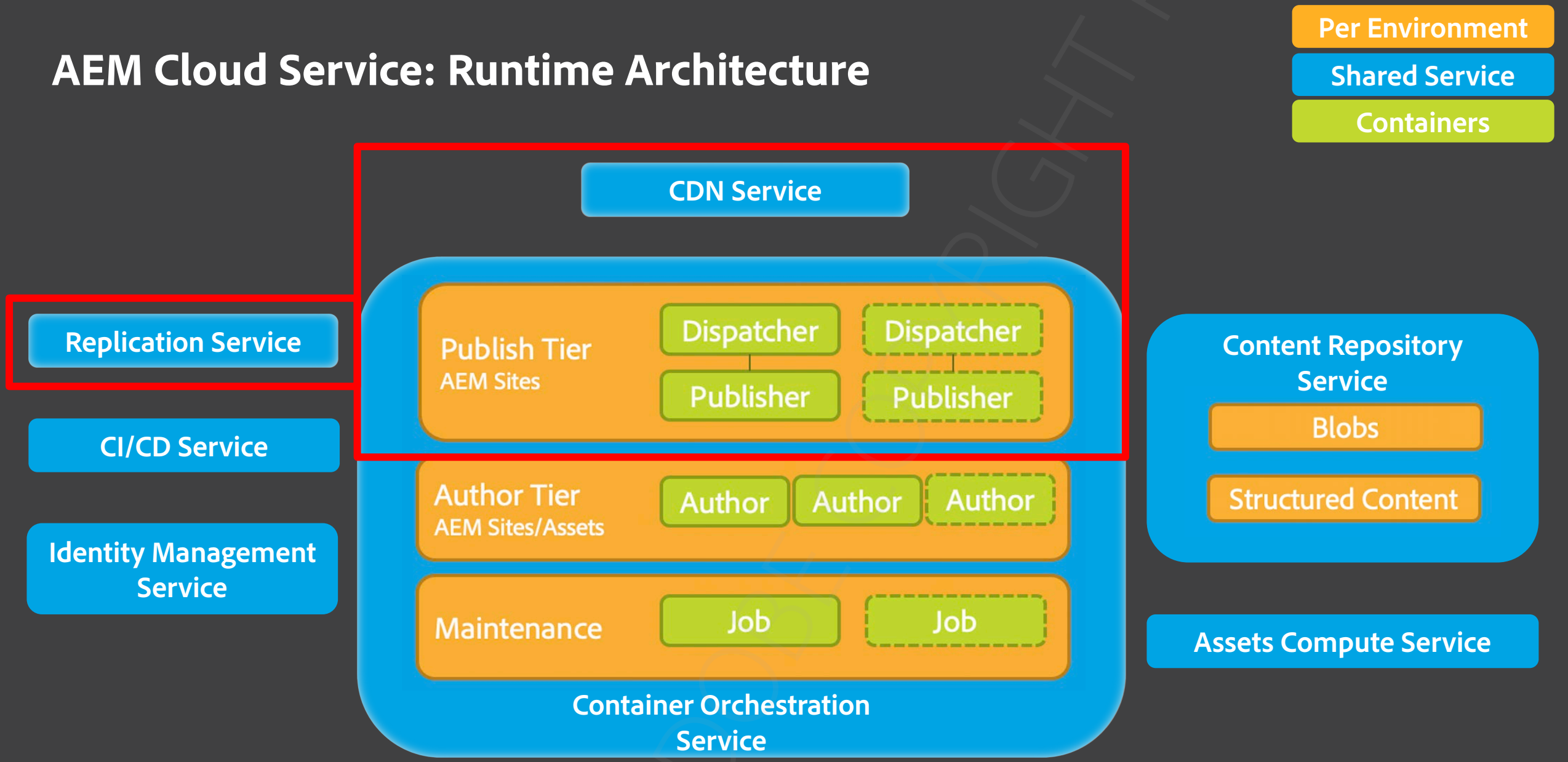
Manage Publishing and Content Delivery

Objectives:

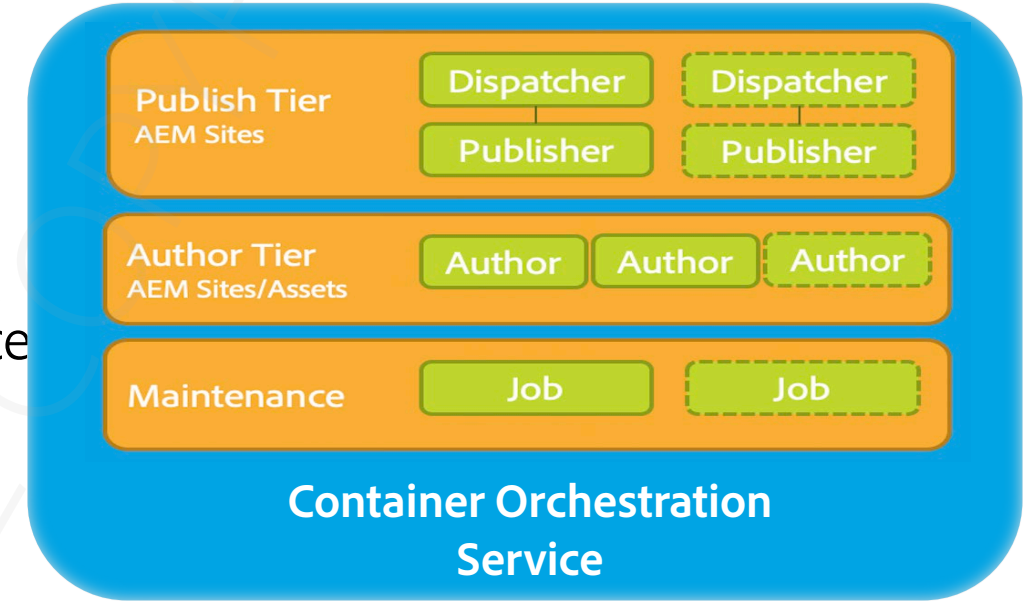
- Publish Service
- Apache Sling Content Distribution and the Golden Master
- Publishing a page
- Publishing a tree of nodes
- Configuring and testing the Dispatcher

AEM Cloud Service: Runtime Architecture

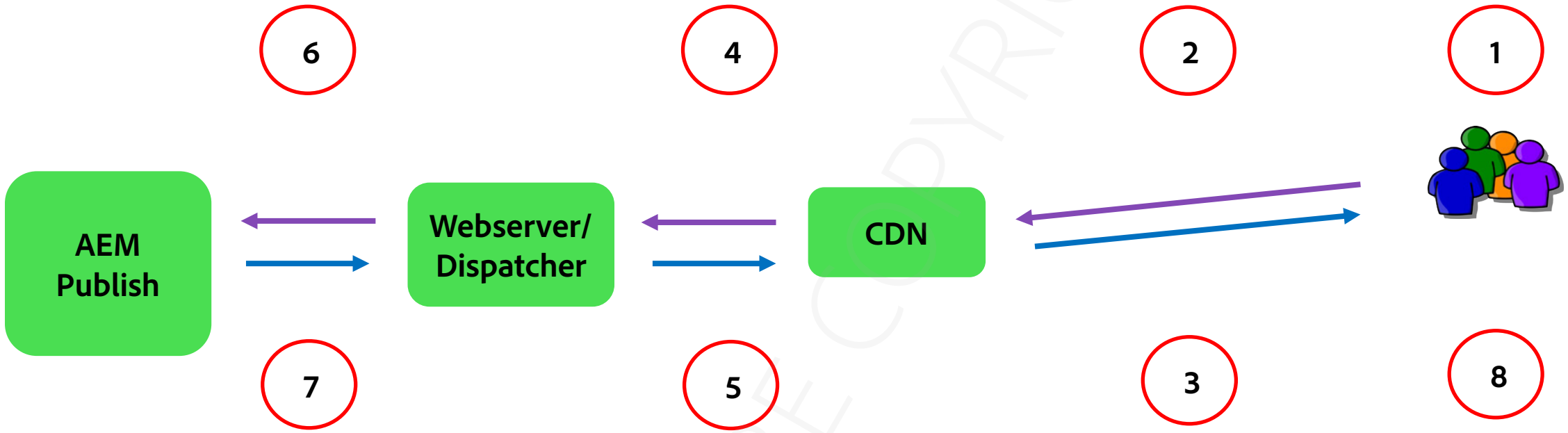


Everything is a service

- Paradigm: Service
  - Don't think of individual instances or nodes
- No need to configure or maintain
  - Replication agents
  - Publish instances
- No need to worry about how many publish instance are there

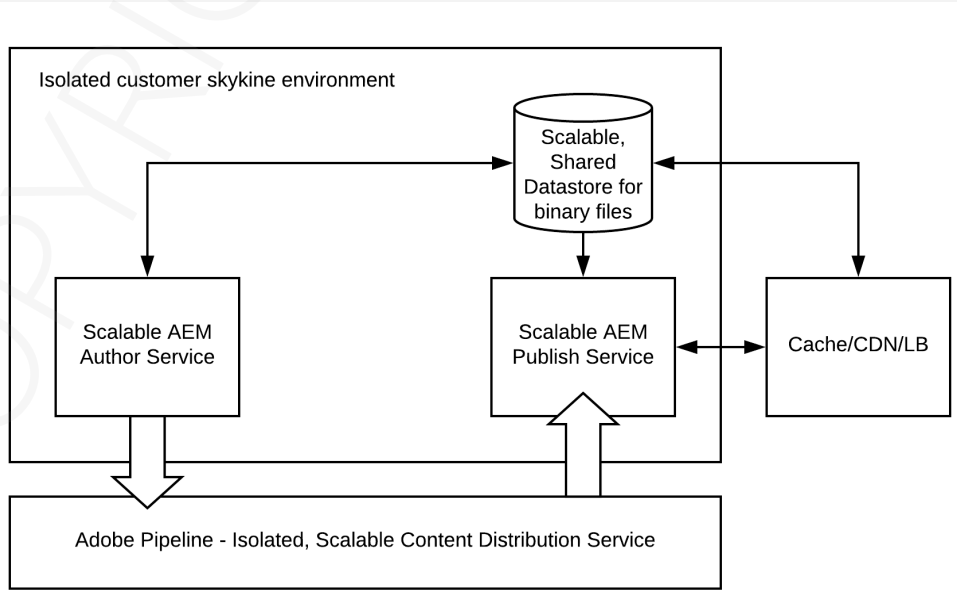


Content Delivery Flow



Content replication – Sling Content Distribution

- Sling Content Distribution is used to move the content to a pipeline service
- Pipeline service runs on Adobe I/O
  - Outside of the AEM runtime
- Publishing Process:
  - Automated
  - Auto-configures itself during runtime when when publish nodes are added or removed
  - Atomic publication/unpublication to all publish instances



Golden master

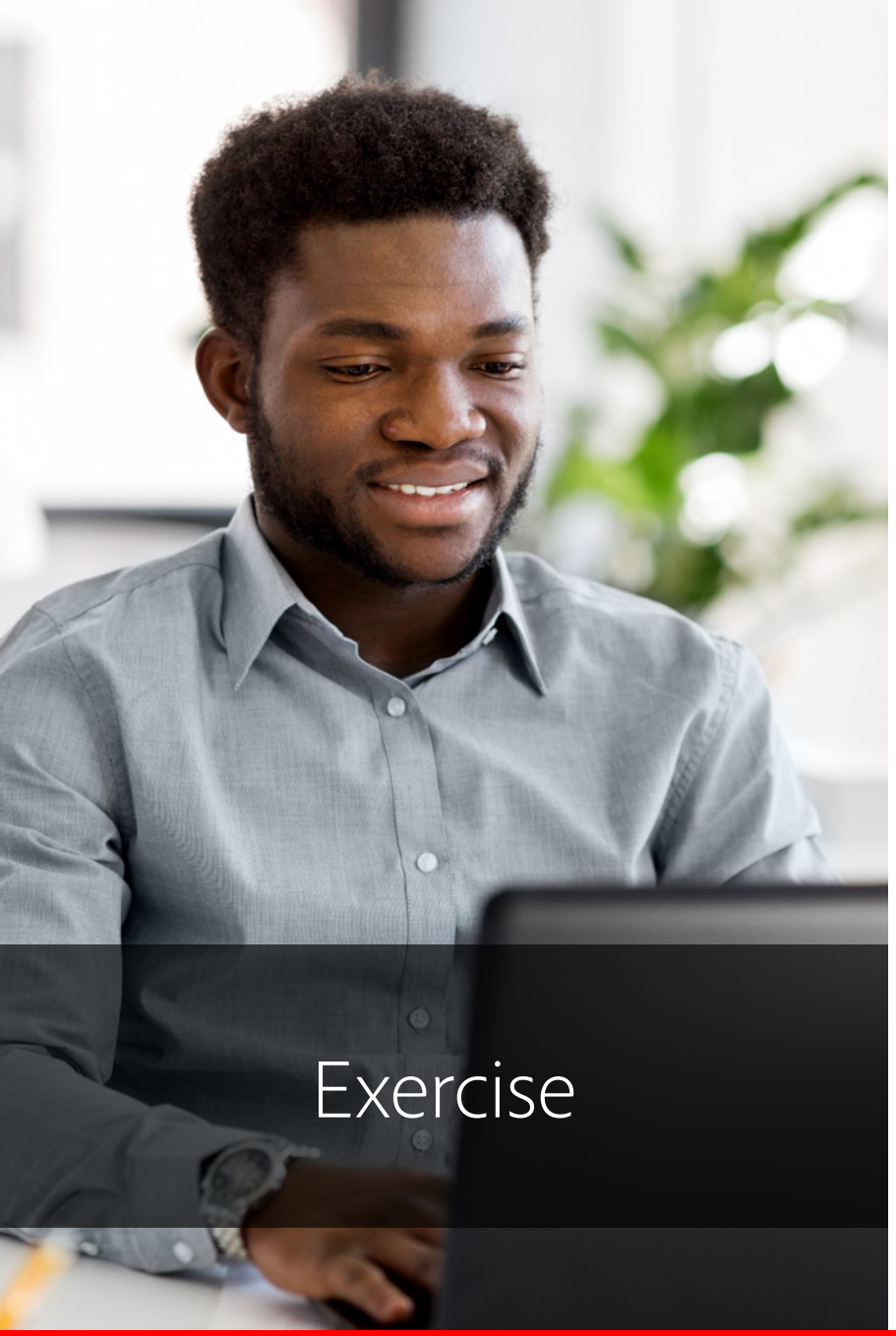
Golden master:

- Special-purpose publish node
- Never accessed by any end user
- Master from which all the nodes of the publish service are created

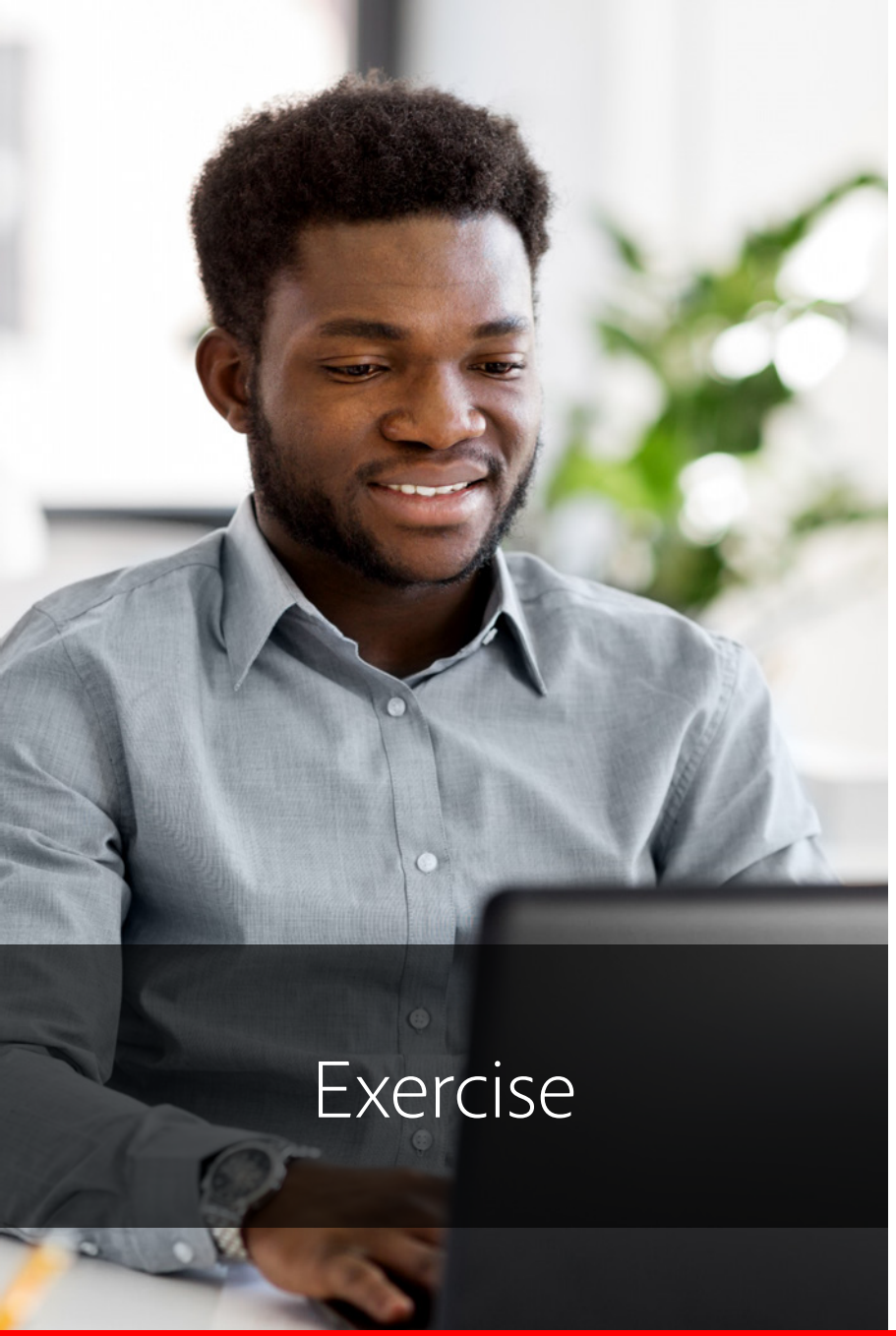
Maintenance operations such as compaction are performed on the content repository attached to the golden master

Quirks of local development

- Adobe Experience Manager (AEM) cloud-ready quickstart uses Replication, not Content Distribution
  - Classic replication capabilities need to be used with an Author/Publish setup
- AEM as a Cloud Service publication mechanism is backwards compatible with the AEM Replication Java APIs
- **Caution**: You cannot troubleshoot replication problems on a local cloud-ready instance



Exercise 1: Publish a page



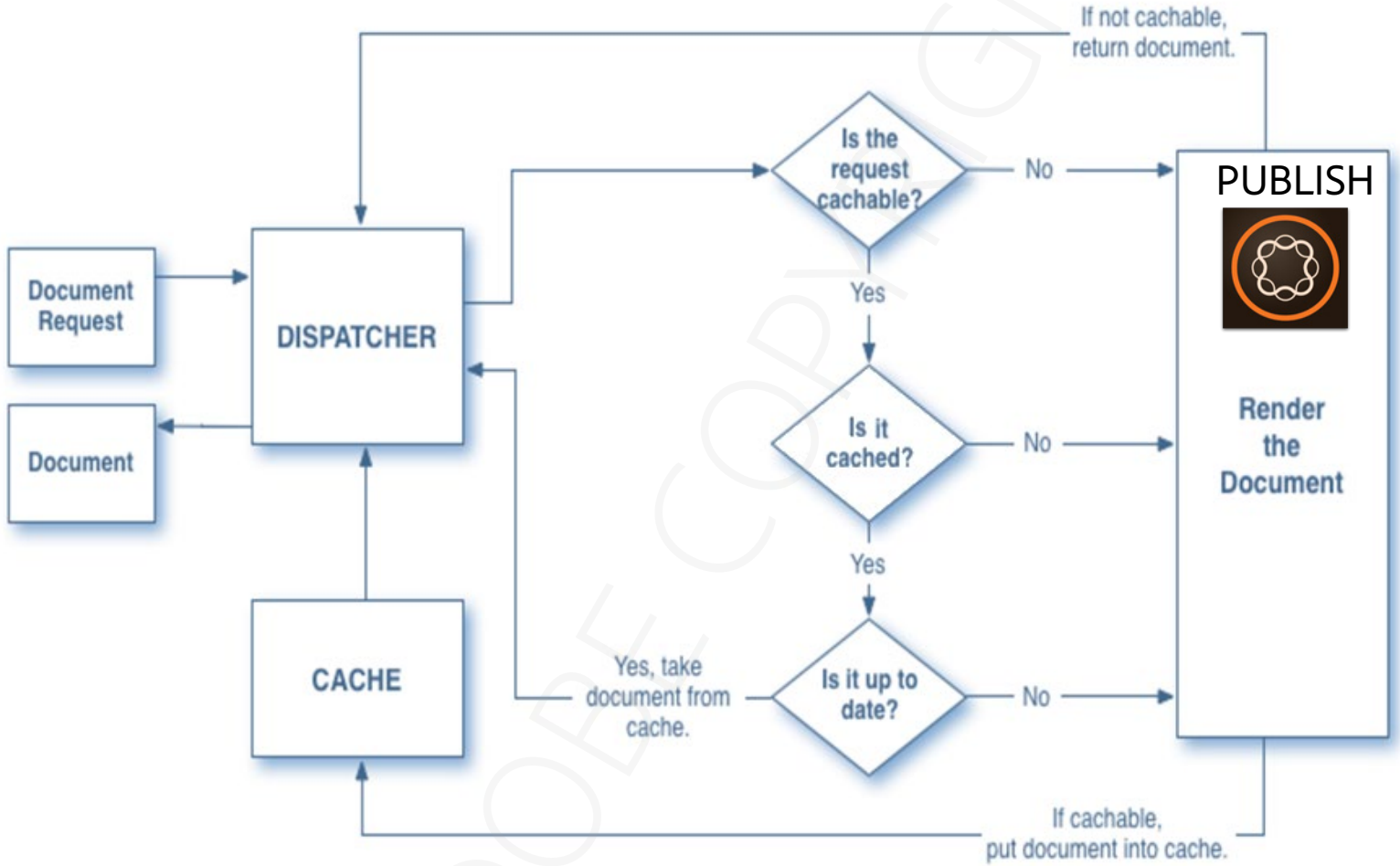
Exercise 2: Publish a tree of nodes from the JCR

Webserver/Dispatcher

- Traffic goes through an apache web server that has the AEM dispatcher configured

- AEM dispatcher
  - Primarily a cache to limit processing on the publish nodes in order to increase performance
  - Provides a security layer to protect the AEM publish service

Dispatcher Caching Algorithm



Caching Parameters

Parameter	Allowed	Not Allowed
The URL must be allowed by the /cache => /rules and /filter sections of dispatcher.any.	N/A	N/A
The URL must have a file extension.	/content/foo.html	/content/foo
The URL must not contain any querystring parameters (no ? in the URL).	/content/foo.html	/content/foo.html?queryparam=1
If the URL has a suffix path, then the suffix path must have a file extension.	/content/foo.html/suff/path.html	/content/foo.html/suffix/path
The HTTP method must be GET or HEAD.	GET /foot.html HTTP/1.1 HEAD /foo.html HTTP/1.1	POST /foo.html HTTP/1.1

Caching Parameters

Parameter	Allowed	Not Allowed
HTTP response status (from AEM) must be 200 OK.	HTTP/1.1 200 OK	HTTP/1.1 500 Internal Server Error HTTP/1.1 404 Not Found
HTTP response (from AEM) must not contain the response header “Dispatcher: no-cache”.	HTTP/1.1 200 OK Content-Type: text/plain Content-Length:42	HTTP/1.1 200 OK Content-Type: text/plain Content-Length: 42 Dispatcher: no-cache
Exception	**Allowed**	**Not Allowed**
If the URL is a cacheable resource path with an extension, and a URL with a suffix path is requested, then it is not cached.	/content/foo.html is already cached; /content/foo.html/suffix/path.html	N/A
When the suffix file already exists in the cache and the resource file is requested, the resource file is cached instead.	/content/foo.html	N/A

Configure dispatcher

Configurations are stored in a file called **dispatcher.any**

- dispatcher.any has 1 to many *farms* of configurations

A *farm* is a set of common configurations for a website, brand, or other logical grouping

farm contain several major sections that configure

- Caching rules
- Client headers
- Request filtering
- Virtual hostnames

Major sections of dispatcher.any

/available | enabled farms section

- Configures a set of load-balanced renderers
- List of farms or websites, each has a name. For example, /website

/clientheaders section

- Client headers are passed to renderers

/virtualhosts section

- Virtual host names [can use an asterisk (*)]
- Lets the webserver handle virtual hosting

/renders section

- Hosts IPs and ports corresponding to the publish instances of AEM

/filter section

- Filters to allow or deny access to specific paths

/cache section

- /docroot: Cache location

Configure what gets cached

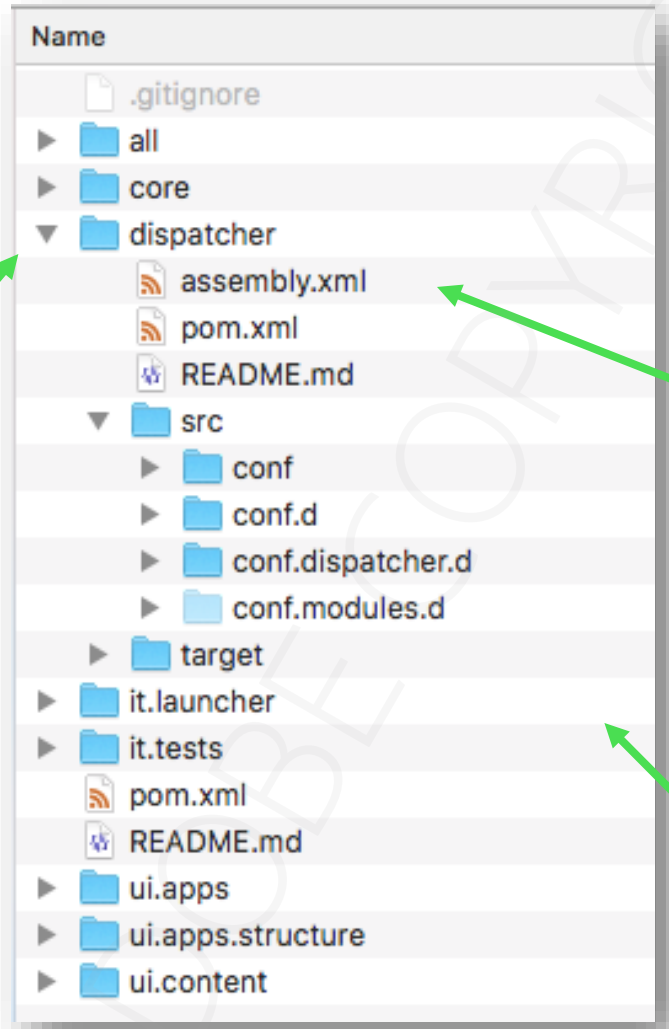
Within the /cache section you can configure:

- /statfileslevel: Defines the levels down to which the .stat files will be written.
  - Controls how much of the tree will be invalidated on publish
- /rules: Specifies the paths and documents to be cached
- /invalidate: Defines the documents that are invalid after content update

AEM Cloud service | Deploy dispatcher configuration with Code

```
<modules>
    <module>all</module>
    <module>core</module>
    <module>ui.apps</module>
    <module>ui.apps.structure</module>
    <module>ui.content</module>
    <module>it.tests</module>
    <module>it.launcher</module>
    <module>dispatcher</module>
</modules>
```

dispatcher module



assembly instructions for dispatcher module

AEM project (from Archetype)

AEM Cloud service | Dispatcher configuration file structure

```
conf.d
├── available_vhosts
│   └── default.vhost
├── dispatcher_vhost.conf
├── enabled_vhosts
│   ├── README
│   └── default.vhost -> ../available_vhosts/default.vhost
└── rewrites
│   ├── default_rewrite.rules
│   └── rewrite.rules
└── variables
    ├── custom.vars
    └── global.vars
```

Configuration of the Apache Web Server ***httpd.conf***

```
conf.dispatcher.d
├── available_farms
│   └── default.farm
├── cache
│   ├── default_invalidate.any
│   ├── default_rules.any
│   └── rules.any
├── clientheaders
│   ├── clientheaders.any
│   └── default_clientheaders.any
├── dispatcher.any
├── enabled_farms
│   ├── README
│   └── default.farm -> ../available_farms/default.farm
├── filters
│   ├── default_filters.any
│   └── filters.any
├── renders
│   └── default_renders.any
└── virtualhosts
    ├── default_virtualhosts.any
    └── virtualhosts.any
```

Configuration of ***dispatcher.any***

conf.d folder – generates httpd.conf

available_vhosts/default.vhost

- Contains a sample virtual host.

available_vhosts/<CUSTOMER_CHOICE>.vhost

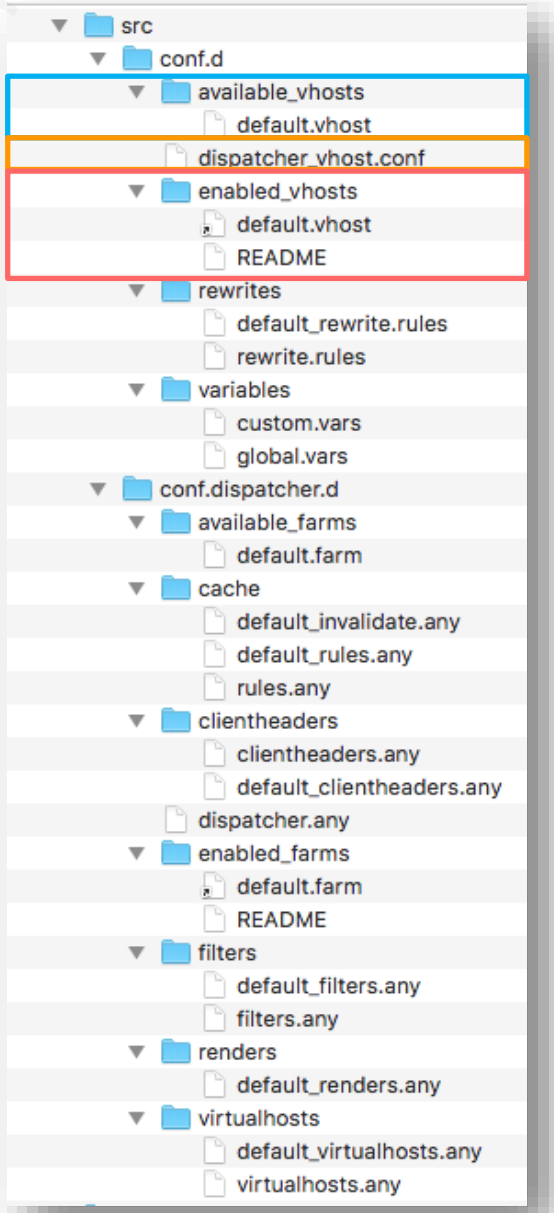
- One or more of these files can exist and they contain the <VirtualHost> entries to match host names and allow Apache to handle each domain traffic with different rules

dispatcher_vhost.conf

- Used to illustrate how your virtual hosts and global variables are included.

enabled_vhosts/default_vhost

- Symbolic link to custom virtual host definition. Enables the custom available hosts.



conf.d folder – generates httpd.conf

rewrites/default_rewrite.rules

- Default rewrite rules suitable for a standard project.

rewrites/rewrite.rules

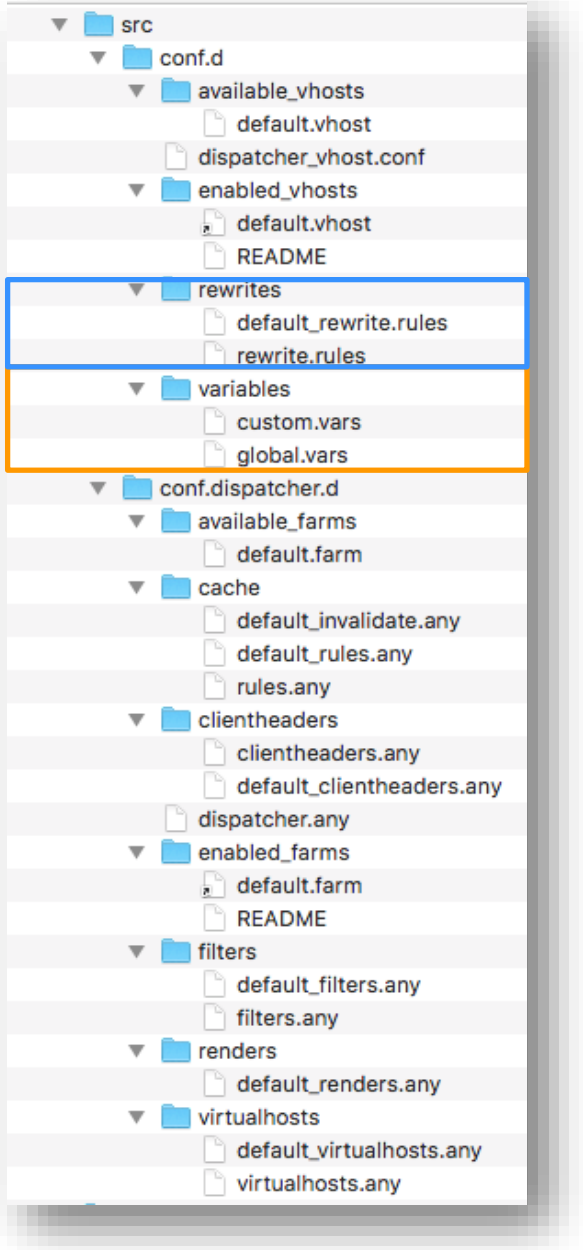
- Included from inside your .vhost files. Contains a set of rewrite rules for mod_rewrite

variables/custom.vars

- Included from inside your .vhost files. Contain Apache variables definitions

variables/global.vars

- Included from inside the dispatcher_vhost.conf file. Controls dispatcher and rewrite log level



conf.dispatcher.d folder – generates dispatcher.any

available_farms/default.farm

- Contains a sample dispatcher farm.

available_farms/<CUSTOMER_CHOICE>.farm

- One or more of these files can exist and they contain farms to match host names and allow the dispatcher module to handle each farm with different rules

cache/default_invalidate.any

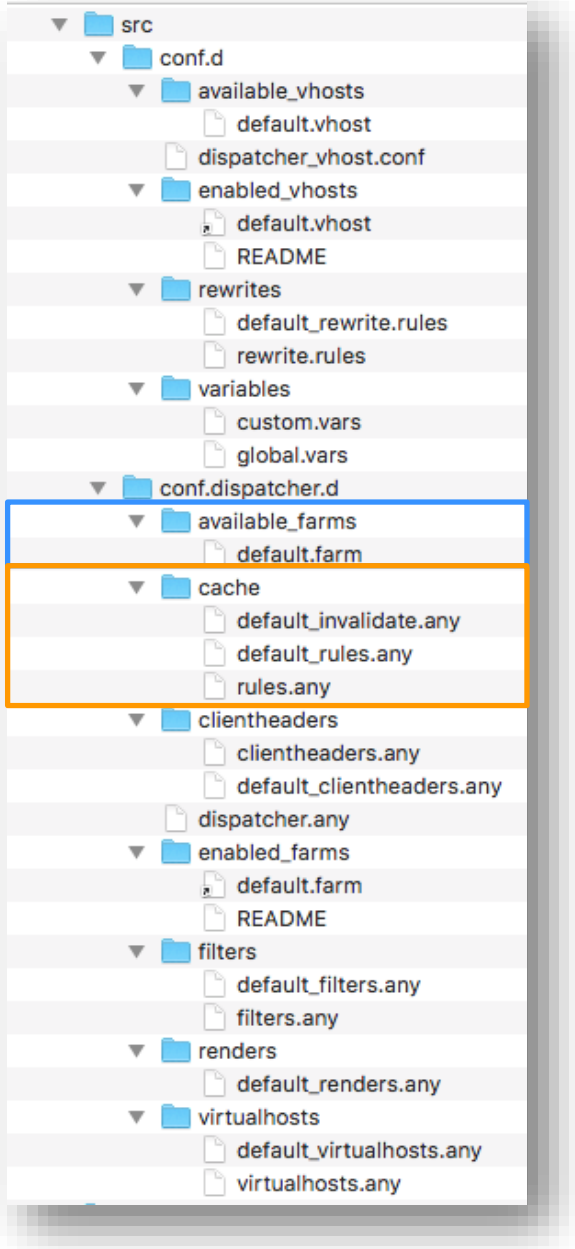
- Part of base framework, gets generated on startup. You are required to include this file in every farm you define, in the cache/allowedClients section

cache/default_rules.any

- Default cache rules suitable for a standard project

cache/rules.any

- Custom caching rules



conf.dispatcher.d folder – generates dispatcher.any (cont)

dispatcher.any

- Part of base framework, used to illustrate how your dispatcher farms are included

clientheaders/default_clientheaders.any

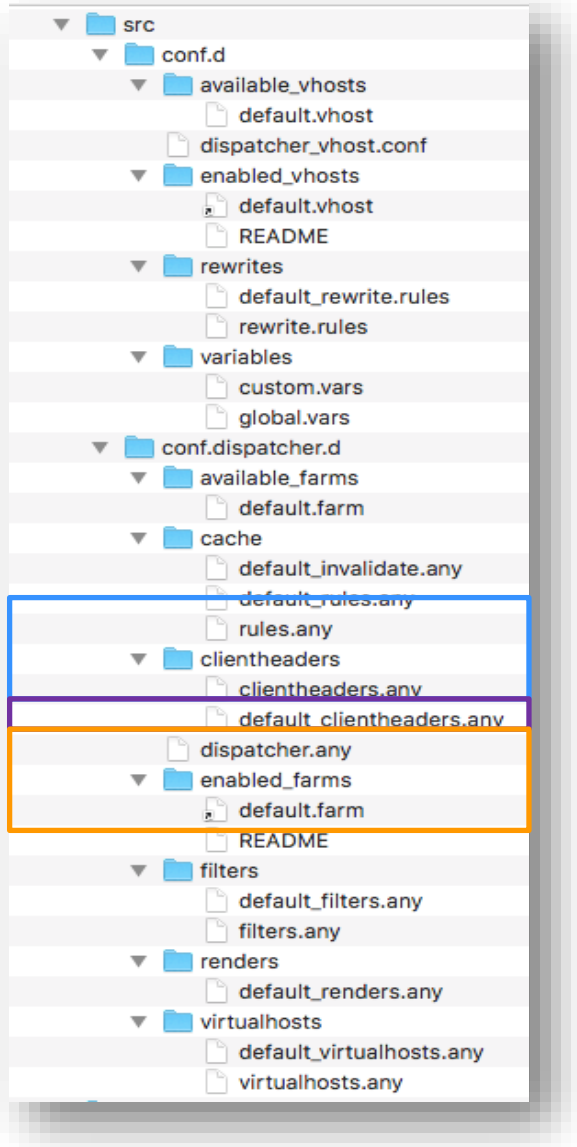
- Default request headers to forward to backend, suitable for a standard project.

clientheaders/clientheaders.any

- Which request headers should be forwarded to the backend.

enabled_farms/default.farm

- Default filters suitable for a standard project



conf.dispatcher.d folder – generates dispatcher.any (cont)

filters/default_filters.any

- Default filters suitable for a standard project

filters/filters.any

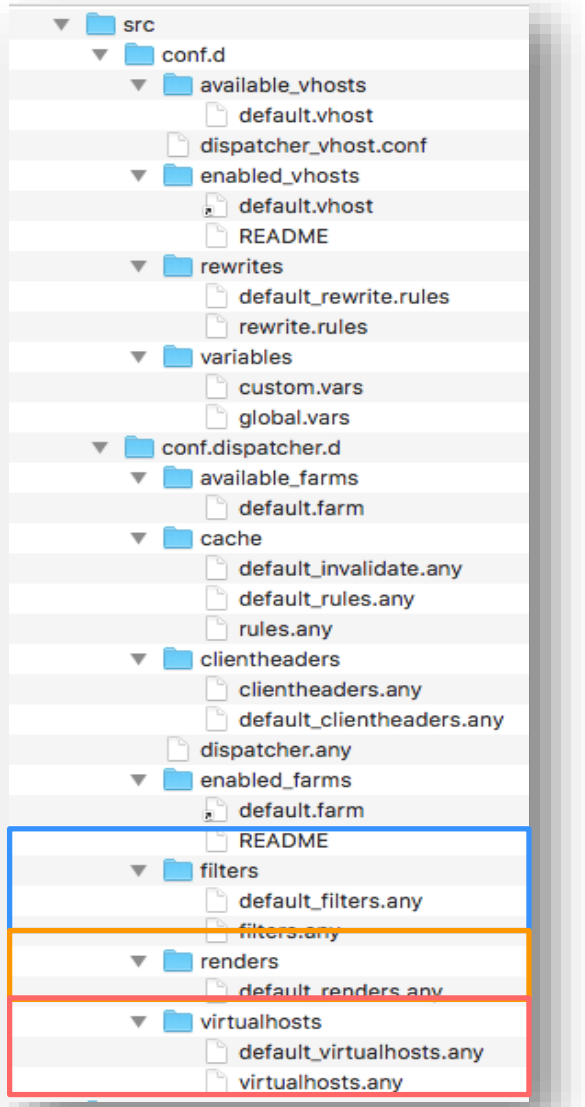
- Custom filter definitions.

renders/default_renders.any

- Part of base framework, gets generated on startup. You are required to include this file in every farm you define, in the renders section

virtualhosts/default_virtualhosts.any

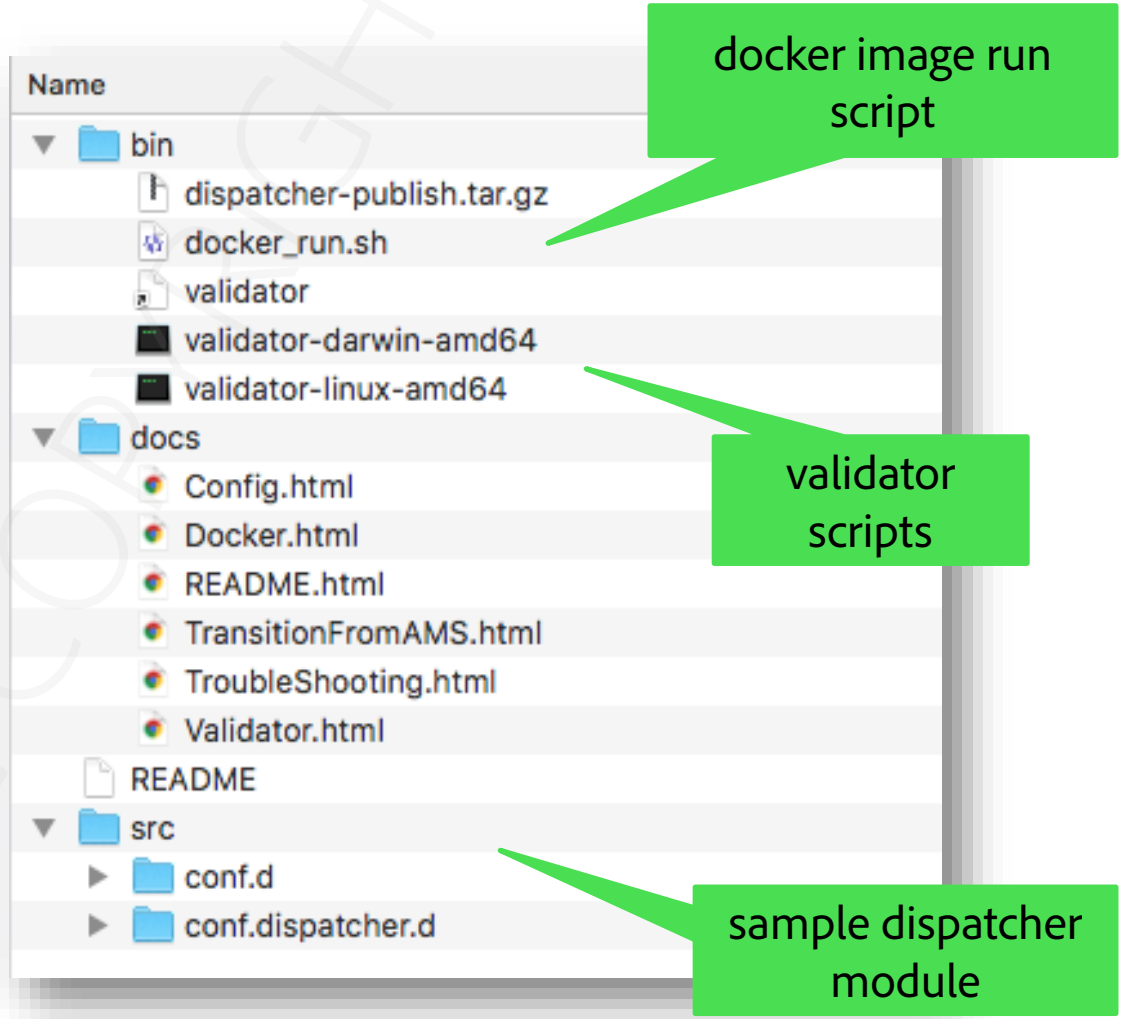
- Default host globbing suitable for a standard project



AEM Cloud service SDK dispatcher tools

The AEM SDK Dispatcher tools provide:

- Vanilla file structure containing the configuration files to include in a maven project for dispatcher
- Tooling for customers to validate a dispatcher configuration locally
- Docker image that brings up the dispatcher locally



Dispatcher SDK – validator utility

```
[/AdobeInstructor/6.xSkyline/dispatcher/DispatcherSDKv2.0.5/bin $ ./validator
Cloud manager validator 2.0.17

    Subcommands are:

    - any:        dump dispatcher configuration files
    - collect:    collect all Apache directive used in multiple configuration files
    - dispatcher: validate one or more dispatcher configuration files
    - httpd:      validate one or more httpd configuration files
    - full:       validate both httpd and dispatcher configuration files
    - whitelist:  prints the whitelisted Apache directives
```

AEM Cloud Service SDK – validator Utility – *full -d*

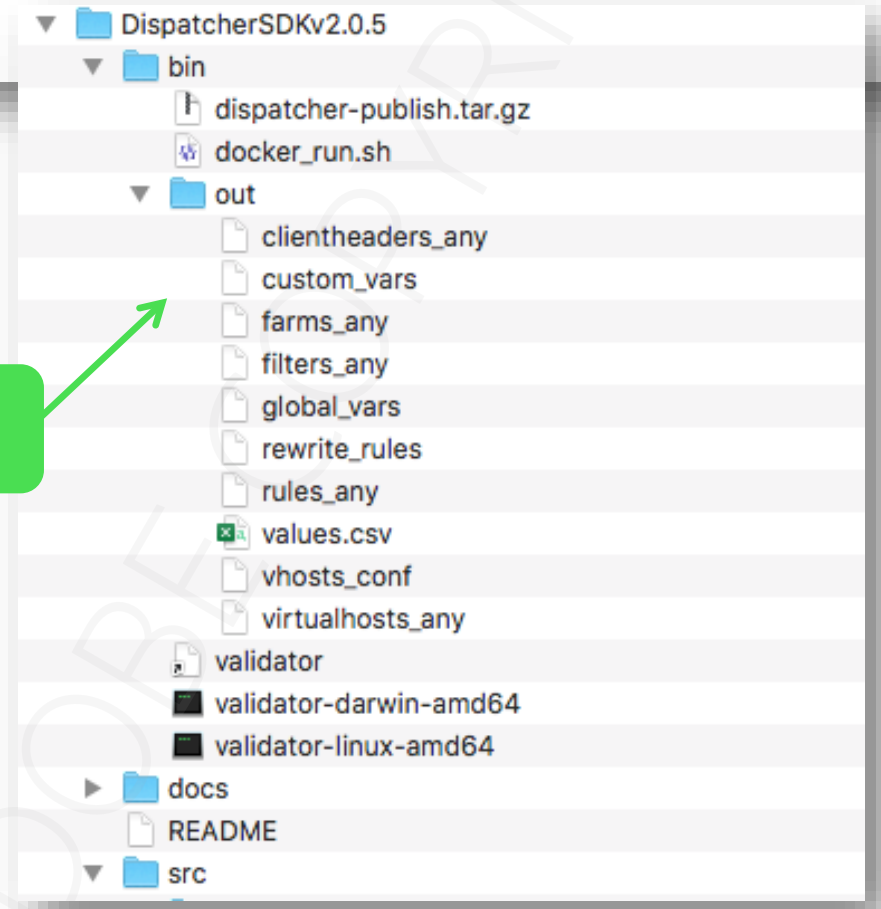
```
$ ./validator full -d <output-foldername> <location-of-dispatcher-package>/<dispatcher-package-version.zip>
      OR
$ ./validator full -d <output-foldername> <location-of-dispatcher-folder>/src
```

- *<output-foldername>* - location where the runtime web server/dispatcher deployment config files will be written
- *<location of dispatcher-package>* - location of dispatcher module target in project build directory
- *<location of dispatcher folder/src>* - location of dispatcher module src folder in project build directory

Dispatcher tools - validator Utility – *create local dispatcher config files*

```
/AdobeInstructor/6.xSkyline/dispatcher/DispatcherSDKv2.0.5/bin $ ./validator full -d out /AdobeInstructor/6.xSkyline/WKND/aem-guides-wknd-develop/dispatcher/src
Cloud manager validator 2.0.17
2019/10/22 14:34:47 No issues found
/AdobeInstructor/6.xSkyline/dispatcher/DispatcherSDKv2.0.5/bin $
```

output (dump) folder

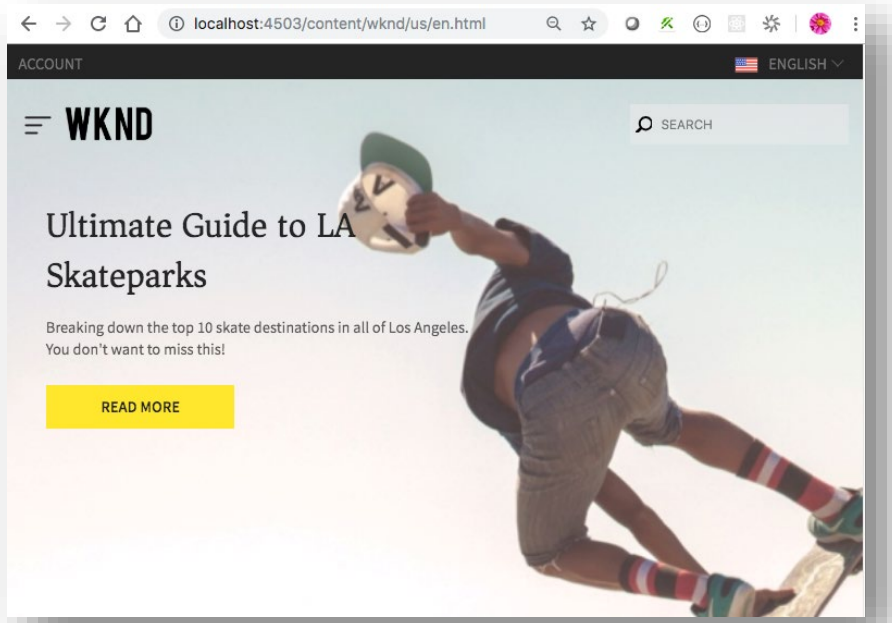


Demo 1: Validate the dispatcher configuration

Set up AEM publish instance

1. Start AEM Cloud Service local publish instance
2. Deploy application to publish instance
   - Profiles: *–Padobe-public –PautoInstallPackagePublish*
3. Test deployment by accessing application on publish instance

Remember AEM cloud-ready quickstart jar does not come with sample content



Run Dispatcher configuration test – *start docker image w/ Apache*

Execute *docker_run.sh* to start the docker image with the apache web server and the dispatcher in it.

folder with dispatcher config files

AEM publish instance IP address and port

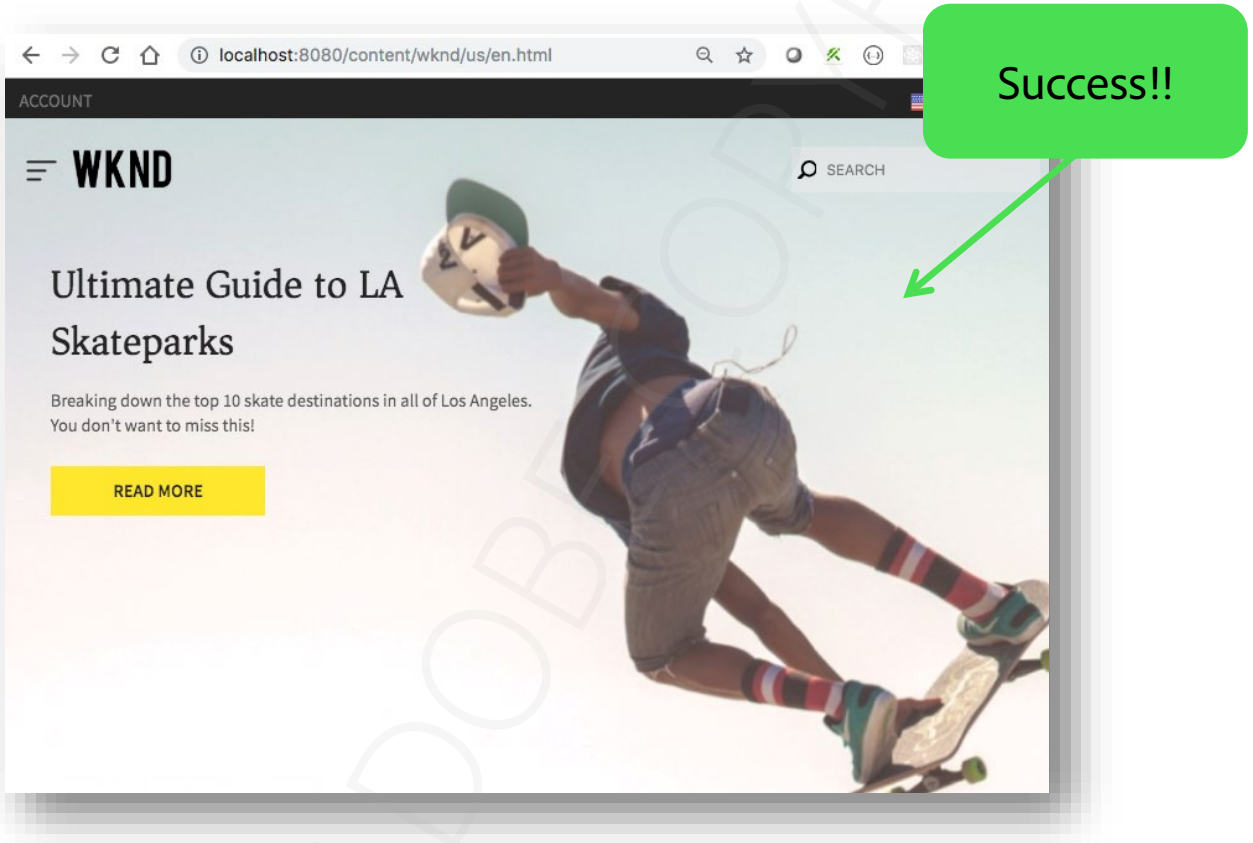
Apache port

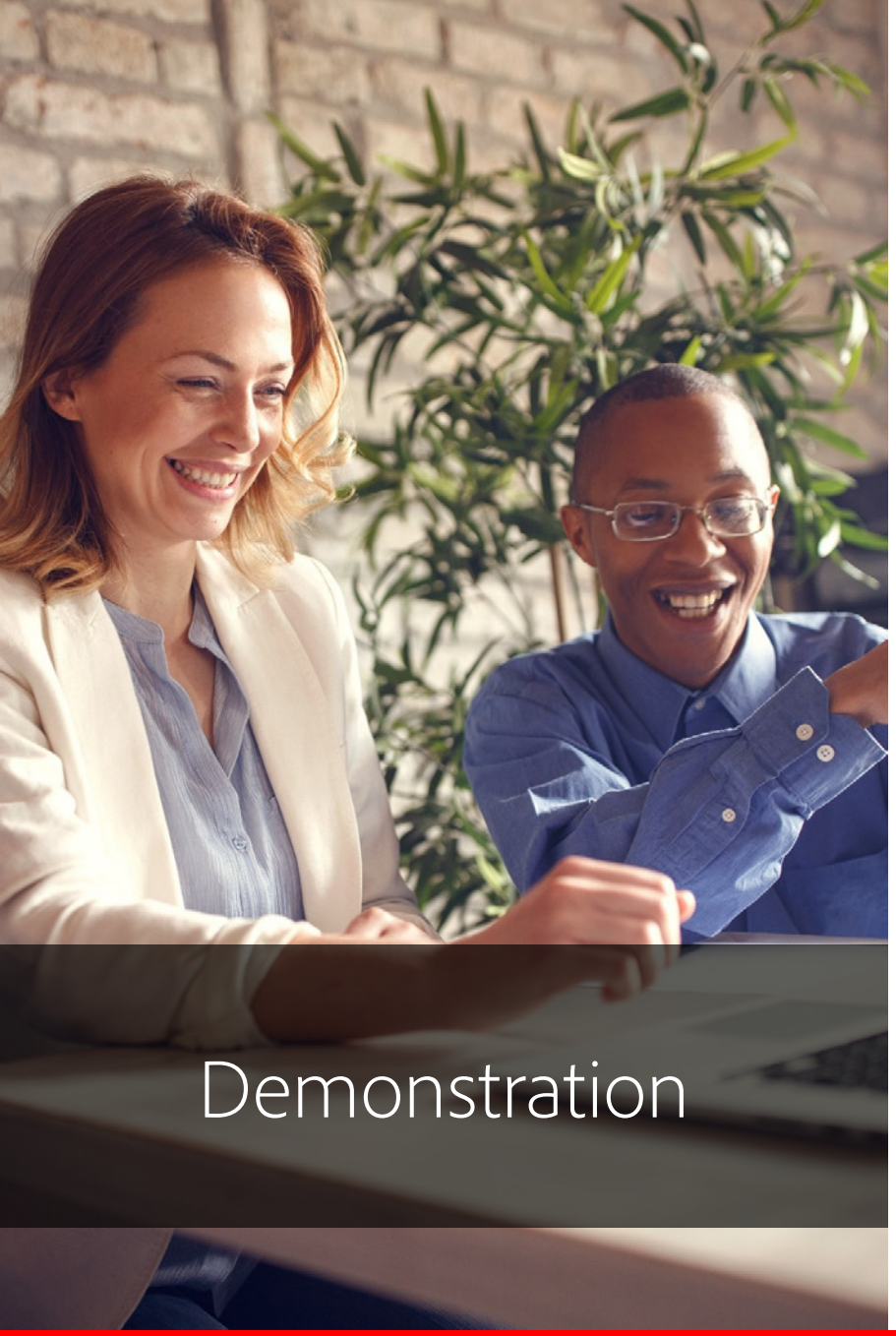
```
./docker_run.sh out 192.168.0.14:4503 8080
```

```
/AdobeInstructor/6.xSkyline/dispatcher/DispatcherSDKv2.0.5/bin $ ./docker_run.sh out 192.168.0.14:4503 8080
Running script /docker_entrypoint.d/10-check-environment.sh
Running script /docker_entrypoint.d/20-create-docroots.sh
Running script /docker_entrypoint.d/30-wait-for-backend.sh
Waiting until 192.168.0.14 is available
192.168.0.14 resolves to 192.168.0.14
Running script /docker_entrypoint.d/40-allowed-clients.sh
Running script /docker_entrypoint.d/50-check-expiration.sh
Running script /docker_entrypoint.d/60-check-loglevel.sh
Starting httpd server
[Wed Oct 23 19:13:39.944249 2019] [:notice] [pid 1:tid 140185461353352] ModSecurity for Apache/2.9.2 (http://www.modsecurity.org/) configured.
[Wed Oct 23 19:13:39.944295 2019] [:notice] [pid 1:tid 140185461353352] ModSecurity: APR compiled version="1.6.3"; loaded version="1.6.3"
[Wed Oct 23 19:13:39.944301 2019] [:notice] [pid 1:tid 140185461353352] ModSecurity: PCRE compiled version="8.42 "; loaded version="8.42 2018-03-20"
[Wed Oct 23 19:13:39.944304 2019] [:notice] [pid 1:tid 140185461353352] ModSecurity: LIBXML compiled version="2.9.8"
[Wed Oct 23 19:13:39.944306 2019] [:notice] [pid 1:tid 140185461353352] ModSecurity: Status engine is currently disabled, enable it by set SecStatusEngine to On.
[Wed Oct 23 19:13:40.021136 2019] [mpm_worker:notice] [pid 1:tid 140185461353352] AH00292: Apache/2.4.41 (Unix) mod_qos/11.59 Communique/4.3.3-20190911 configured -- resuming normal operations
[Wed Oct 23 19:13:40.021327 2019] [core:notice] [pid 1:tid 140185461353352] AH00094: Command line: 'httpd -d /etc/httpd -f /etc/httpd/conf/httpd.conf -D FOREGROUND -D ENVIRONMENT_DEV'
```

Test dispatcher configuration

Test access through the apache web server (in the docker image) by using the dispatcher URL, for example: http://localhost:8080/content/wknd/us/en.html





Demo 2: Test dispatcher configuration

CDN | Content freshness and version consistency

- Pages are made up of HTML, Javascript, CSS, and images
- Customers are encouraged to leverage the clientlibs framework
  - Provides automatic version management
  - Referencing HTML pages are updated with new links to updated library versions

CDN | Client Library Caching

The AEM Client library framework

- Provides automatic version management
  - Clientlibs are stored on a unique versioned paths
  - This feature is called Strict clientlib versioning and is enabled by default in CS environments
- Css and javascript are cached indefinitely in the client browser cache
  - Browsers that don't support this 'immutable' feature, store these clientlibs for 30 days
  - Allows clientlibs to be sent once for an entire site
- Code changes to clientlibs automatically create a new unique path for caching
  - Pages are re-rendered from publish, cached on dispatcher and CDN.
  - Requests for pages forces a request for the new unique path of the updated clientlibs
  - Updated clientibs are sent to the client

CDN | Dispatcher and CDN Caching

No need to manually invalidate dispatcher cache or CDN

- Dispatcher is flushed from the 1 to 1 publish node based on caching rules set
- CDN respects TTL which means there's no need for a CDN flush.
- In general, HTML content is cached in the CDN for 5 minutes

Recommendation: Prior to accepting live traffic, customers should validate with Adobe customer support that the end-to-end traffic routing is functioning correctly

Directly calling the invalidate.cache API is no longer allowed

- ex: `POST http://dispatcher.hostname.com/invalidate.cache`

CDN | Content Delivery Network (CDN) for AEM as a Cloud Service

AEM Managed CDN

- Recommended approach
- AEM's out-of-the-box CDN
- Tightly integrated
- Satisfies most customers need for a CDN

Customer CDN pointed to AEM Managed CDN

- Allowed on a case-by-case basis
- Customer uses their own CDN and is responsible for managing it

CDN | AEM Managed CDN

1. Customer will provide the signed SSL certificate and secret key to Adobe
2. Inform customer support which domain should be associated with each environment
3. Inform customer support if any IP allowlisting is needed for restricting traffic
4. Customer support will then coordinate with customer to create a CNAME DNS record pointing their FQDN to adobe-aem.map.fastly.net
5. Customer will be notified when SSL certificates are expiring so they can resubmit the new SSL certificate

CDN | Using a Customer managed CDN

Requirements if approved to use a customer managed CDN

- Must be able to configure CDN to work with AEM
- Must have engineering CDN experts on call
- Must pass load testing before going to production

The customer managed CDN

- Points to the AEM CDN
- There is potential for a small performance hit due to the extra request hop
- Customer CDN is only supported for the publish tier and not for the author tier.

Key takeaways

- Properly authenticated user can publish either a page or a tree of nodes
- AEMaaCS publish service includes:
  - Content Delivery Network (CDN) Service
  - Publish Tier
  - Replication Service
- The replication service is now a subscription service, using Apache Sling Content Distribution
- Publishing operations are atomic across all publish nodes.
- AEMaaCS is shipped with a built-in CDN

Key takeaways

- AEM dispatcher is an Apache HTTP Web server module that provides a security and performance layer between the CDN and AEM publisher.
- AEM dispatcher contains mechanisms to generate, and update, static HTML based on the content of the dynamic site