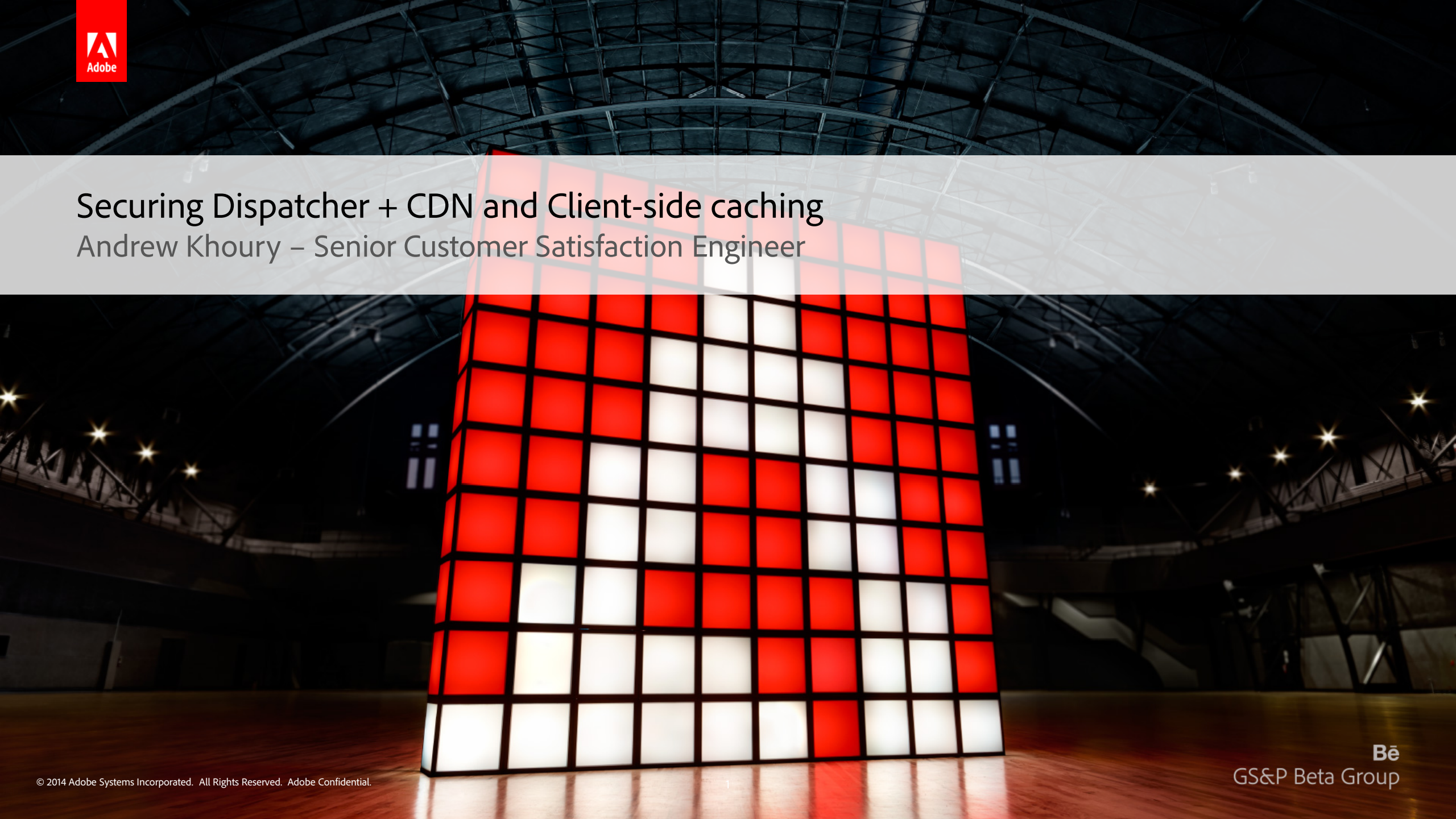


Securing Dispatcher + CDN and Client-side caching

Andrew Khoury – Senior Customer Satisfaction Engineer

Bē
GS&P Beta Group

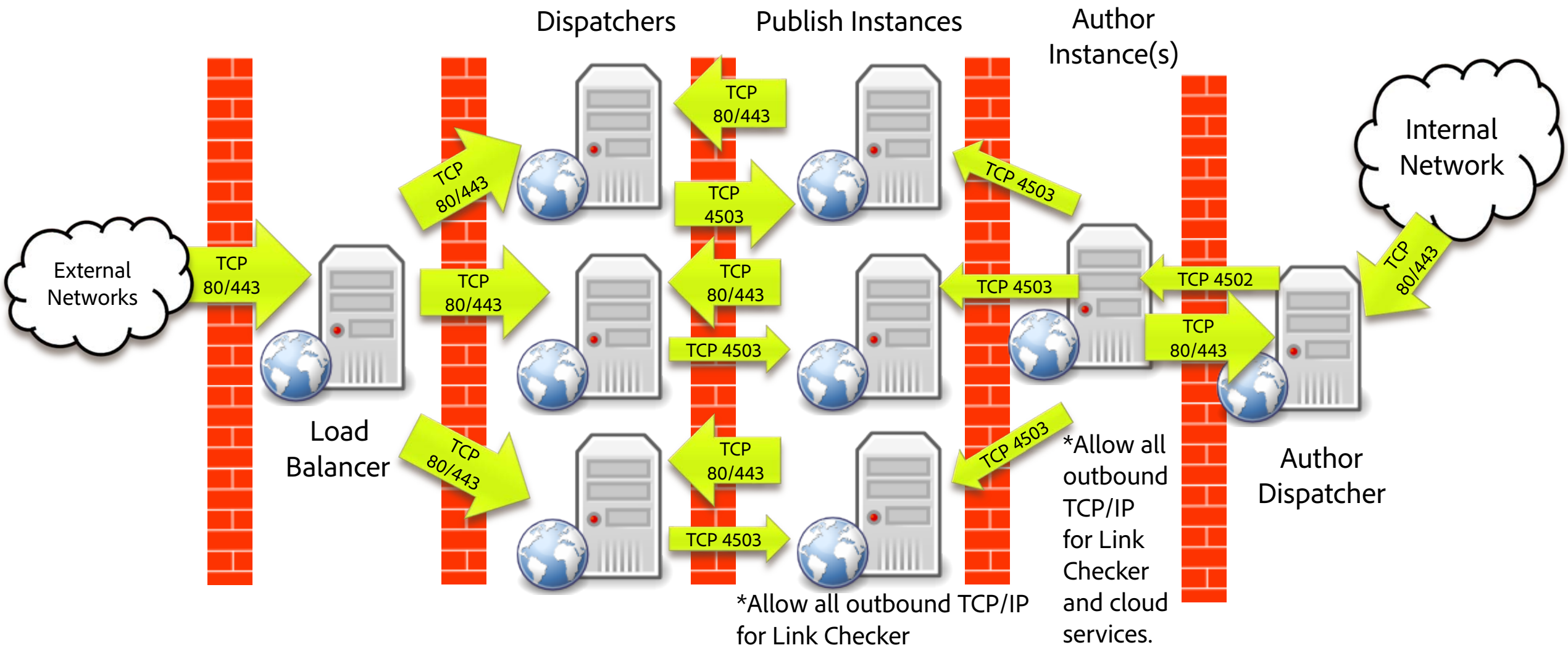
Prerequisite Knowledge

- Before watching this, you need to know:
  - Basics of HTTP protocol
  - Apache HTTP Server configurations
  - AEM Dispatcher
    - Which requests get cached
    - Familiarity with dispatcher.any configurations
    - For review, refer to this past Dispatcher webinar:https://github.com/cqsupport/webinar-dispatchercache

Securing Apache HTTP Server

- Keep Apache HTTP Server binaries up to date - http://httpd.apache.org/
- Be aware of the latest Apache security reports:
  - http://httpd.apache.org/security_report.html
- Limit Apache user access –http://httpd.apache.org/docs/2.2/misc/security_tips.html
- Disable .htaccess files:
  AllowOverride None
- If using SSI, set:
  Options +IncludesNOEXEC
  not:
  Options +Includes
- Disable UserDir or don't load mod_userdir
  UserDir disabled
- Disable directory listing in Apache
  Options -Indexes
- Disable Apache modules you are not using
- Consider using a Web Application Firewall (WAF) such as mod_security or using a WAF appliance (not covered in this presentation)

Firewall Rules



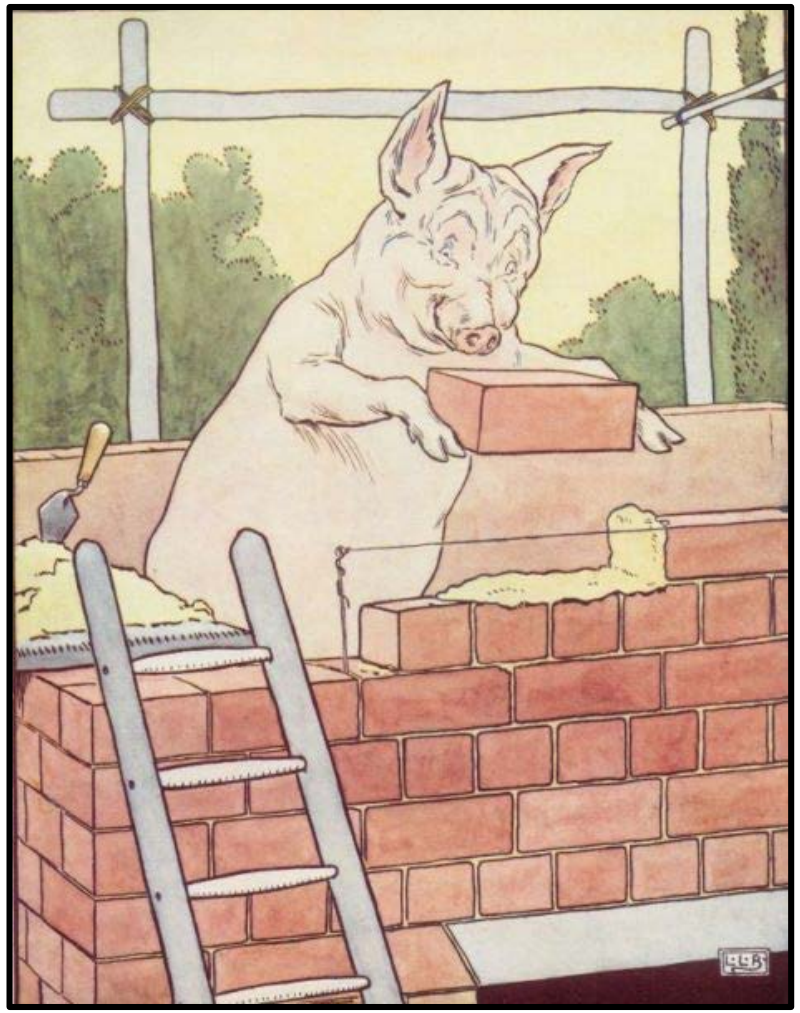
Adobe

Firewall Rules

- If you are using the Link Checker then allow all outbound TCP/IP connections
- If you are not using the Link Checker, but plan to use some Cloud Services then implement outbound firewall rules as mentioned here: http://helpx.adobe.com/analytics/kb/adobe-ip-addresses.html

Keep bad traffic out!

- Leverage dispatcher to keep bad traffic out
- Web Server + Dispatcher
  - Last line of defense before AEM
  - Prevent extra load by
    - Blocking bad requests
    - Caching valid requests (whenever possible)



Dispatcher Security

- Keep dispatcher up to date
  - Bug Fix Listing: http://www.aemstuff.com/tools/dispatcheronlinetracker.html
  - Latest Dispatcher Download: https://www.adobeaemcloud.com/content/companies/public/adobe/dispatcher/dispatcher.html

Dispatcher Security – Implementing /filter in dispatcher.any

- Keeping bad traffic out using **/filter** rules
- Use the new dispatcher rule format (covered earlier)
- Use a whitelist style **/filter** section
  - **Deny everything first**
  - **Then only allow what you need**
  - For allow rules, be specific
    - For example, specify the "method" ("GET", "POST", "HEAD", etc.)
  - For deny rules, don't be specific
- Use the new vanity URL feature

  (After dispatcher 4.1.9 is released)

Dispatcher Security

- If your site doesn't allow user logins then
  - Block HTTP basic auth
    - List all allowed headers in **/clientheaders** in **dispatcher.any**
    - Omit header "Authorization"
  - Block AEM token authentication (/filter section)
    /0091 { /type "deny" /url "*/j_security_check" }
  - Block unused request methods (Apache httpd.conf)

```
<LimitExcept HEAD GET POST>
        deny from all
</LimitExcept>
```

Dispatcher Security – Cache Flooding and Flushing

- Error pages
  - 4xx and 5xx responses
    - Set correct HTTP status (in response from publish)
    - Cache custom error pages
      - use **DispatcherPassError** (httpd.conf)
- Return 403 or 404 for bad requests
  - Block unused selectors (in AEM publish)
  - Block unused querystrings (in AEM publish)
  - How?
    - Use cq-urlfilter – https://github.com/justinedelson/cq-urlfilter
    - Or implement a solution (javax.servlet.Filter) in your application
- Set /serveStaleOnError "1"
- Block unwanted cache flushes
  - **/allowedClients** - restrict which hosts can flush the cache

Dispatcher Security – Preventing Against DoS Attacks

- Implement a periodic refreshing script to cache expensive requests .
  - RSS feed
  - Site map
- Sample script:

```
#!/bin/bash
#recache_file.sh usage: recache_file.sh /content/geometrixx/en.html
PUBLISH_SERVER=http://host:4503
CACHE_ROOT=/var/www/html
filename=$(basename "$1")
tmpfilepath=/tmp/tmp_cache_$filename
if [ -f $tmpfilepath$1 ] ; then
  echo "Not running recache_file.sh - File exists: $tmpfilepath$1"
  exit 0
fi
status=`curl -o $tmpfilepath --silent --write-out '%{http_code}\n'
$PUBLISH_SERVER$1`
if [ $status -eq 200 ]; then
  mv $tmpfilepath $CACHE_ROOT$1;
  #chown apache:apache $CACHE_ROOT$1;
Else
  rm -f $tmpfilepath
fi
```

Dispatcher Security – Protect Against DoS Attacks

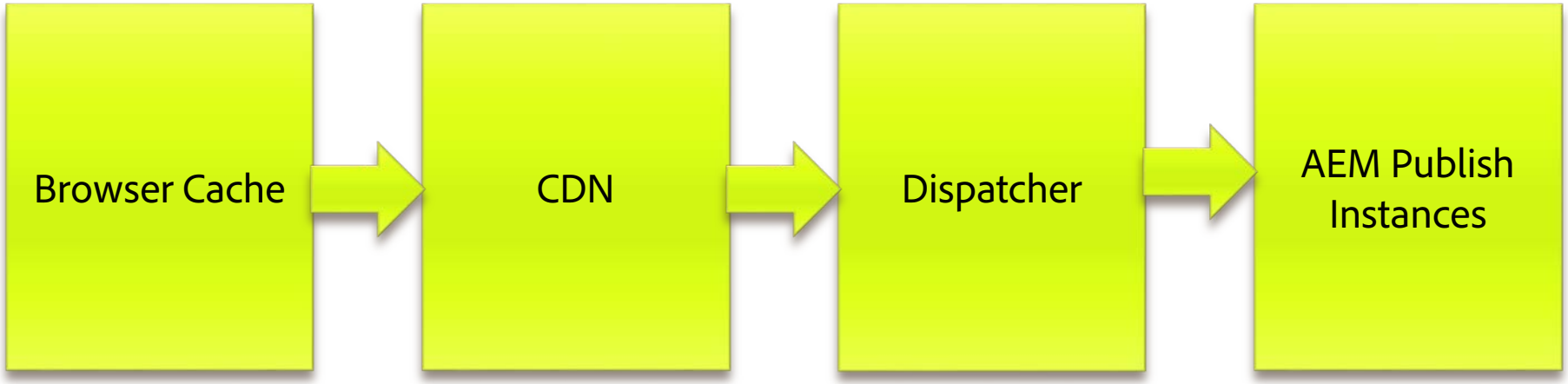
- Configure **/ignoreUrlParams**
  - Allowing requests with querystrings to get cached
  - Allow rules to "ignore" querystring parameters
- Set request timeout per AEM instance (in **/renders** section)
  - Set **/timeout** so that you don't run out of threads in a apache when the back end is unresponsive.
  - 5-10 minutes is usually long enough.

DEMO

Adobe

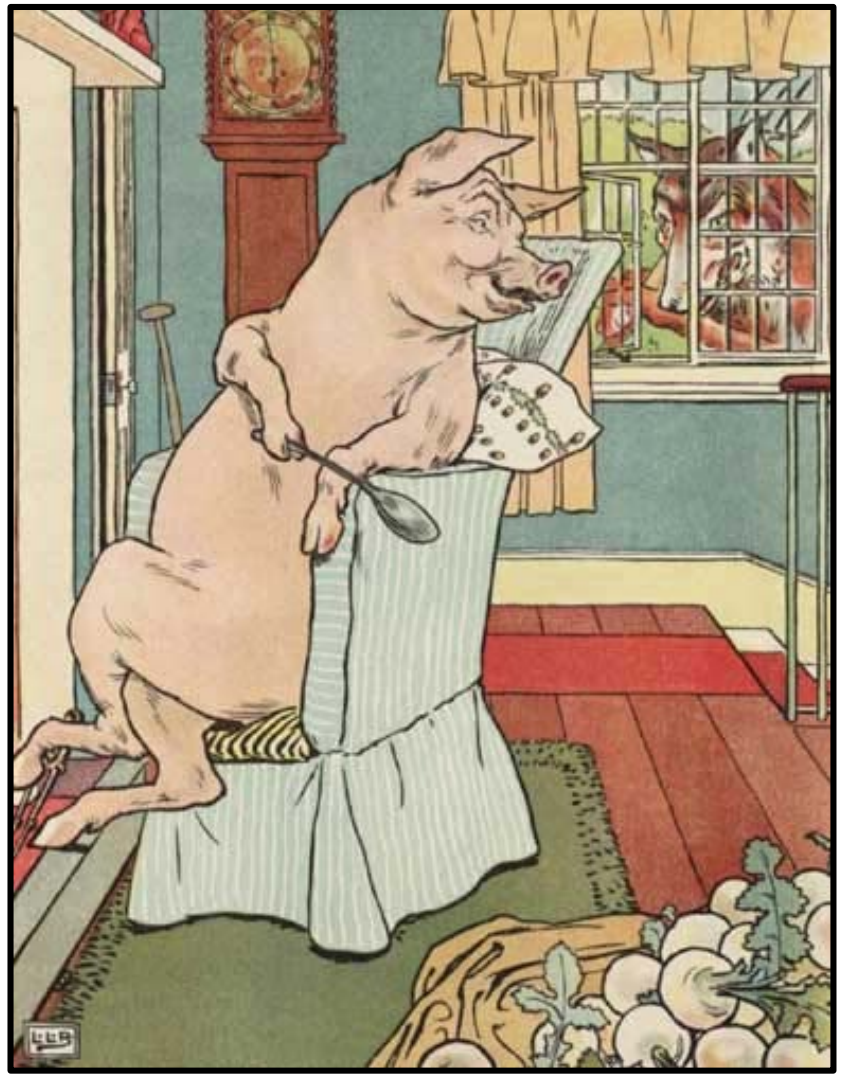
Using a CDN

- Use a CDN
  - A **content delivery network** (**CDN**) is a large distributed system of cache servers that optimize content delivery using geographical proximity
  - CDNs leverage the Cache-Control header or manually configured TTL values to decide when the cached item is stale.
  - Some CDNs support purge requests where you can flush items from the cache on demand.
  - Popular CDN providers – Akamai, Amazon, Rackspace, etc…



Using a CDN

- Use a CDN
  - Another way of reducing traffic from reaching the back-end
  - Multiple Deployment Options
    1. Use short TTLs and serve all URLs through CDN
    2. Serve assets (images, videos, etc.), clientlibs and static resources such as js, css, etc. only through CDN
    3. Implement a custom flush agent that can purge the CDN and use long TTLs (serve everything through CDN)



Using a CDN

1. Whole site through CDN
   - Pros
     - Html pages benefit from CDN edge cache performance.
     - Optimal page load performance
   - Cons
     - Potentially expensive as you are serving everything out of the CDN
     - User waits for TTL expiration before receiving latest content
2. Assets, clientlibs and static resources in CDN
   - Pros
     - Save money on CDN charges
     - Immediate content updates
   - Cons
     - More traffic going to Dispatcher servers

Using a CDN

3. Custom flush agent + long TTLs
   - Pros
     - Gives ability to deliver content on-demand and cache for long periods
     - Reduces dispatcher traffic
   - Cons
     - In real practice not effective or worth the effort:
       - CDN purges are generally slow anyway (so I have been told)
       - Development and maintenance costs on a custom flush agent
   - Possibly makes sense if you have a very large high traffic site .

Using a CDN

- For non-cacheable URLs (with querystring, POST requests, etc) Dispatcher will **not** process the response through the Apache handler.
- So the headers returned will match those coming from AEM.
- **Some CDNs (e.g. Cloudfront) will cache responses that don't have a "Cache-Control: max-age" set.**
- Solutions
  - Set headers to tell the CDN and browser not to cache:
    Cache-Control: no-cache
    Pragma: no-cache
  - Or if relevant, set a short expiration like 30 seconds, for example:
    Cache-Control: max-age=30
  - Or allow the browser to cache, but not the CDN, for example:
    Cache-Control: private, max-age=30

Using a CDN

- For all approaches it would be nice to:
  - Cache js, css and other “static” files for a very long time
  - Be able to use domain sharding
- How do we do that?
  - Use solution developed by Adobe Consulting
    - http://adobe-consulting-services.github.io/acs-aem-commons/#features
    - Versioned Clientlibs – adds md5 hash to clientlib urls
    - Static Reference Rewriter – rewrites the domain of resources included in the page.
      - Used to point clientlibs and other static resources to the CDN
      - and handles domain sharding

Using a CDN

- Tips for using a CDN
  - Use mod_deflate in apache to save money on CDN charges.
  - If your site has personalization then consider leveraging ESI

Client-side Browser Caching

- Using the Client-side Browser Cache
  - Often overlooked
  - Easy to implement
  - Saves you money on CDN charges
  - Use **mod_expires**
  - Leverage **Etags** and **Last Modified Since** headers
    - When not using a CDN, use Sticky Sessions on your load balancer

Client-side Browser Caching

- If using SSI in Apache
  - **Last Modified Since** will not be sent
  - Use **mod_expires** to set an expiration on those html files
- ***No cache related headers are sent for files not cached by dispatcher
- Note: If user clicks refresh it will re -request the URL (bypassing cache)

DEMO


Adobe