AEM ARCHITECTURE

The six key steps to create a solid AEM solution

The six key steps

Create

An overall picture

Perform

Feature mapping

Simplify

The content structure

Trash out

The integration

Finalize the

Deployment model

Plan for

Routine maintenance

THE OVERALL PICTURE

Get it right…

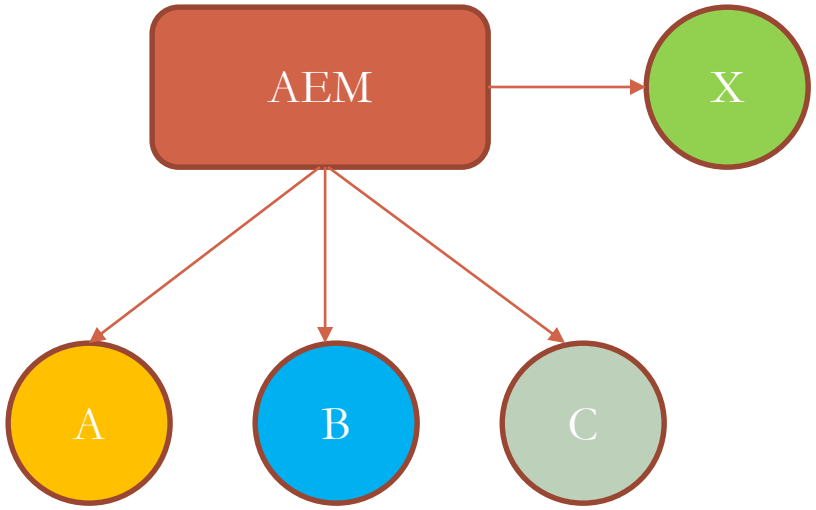
Getting the overall picture right

- Its important to create an overall picture of the solution and finalize where AEM fits in this overall solution before proceeding further
- Decide on the boundary of each of the system in the overall scope and finalize the responsibility of each
- Analyze the system separately from authoring point of view and publish point of view. The overall layout you foresee for author might be very different from that of publish
- The role of AEM in the overall system solution can fall into one of the following three categories

AEM as umbrella system

A standalone AEM implementation or AEM as the wrapper system that integrates with other systems and delivers the site

- AEM most often would be the umbrella system integrating with other systems to serve the functionality
- In this configuration AEM takes the responsibility of integrating with other systems to provide integrated functionality

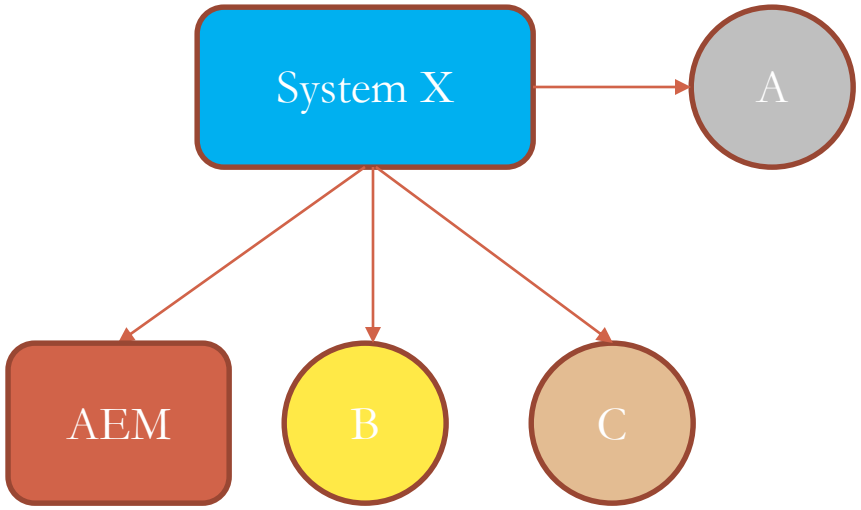


AEM as umbrella system

AEM as provider system

Beware of the impact on AEM internal functionalities when using AEM as provider system

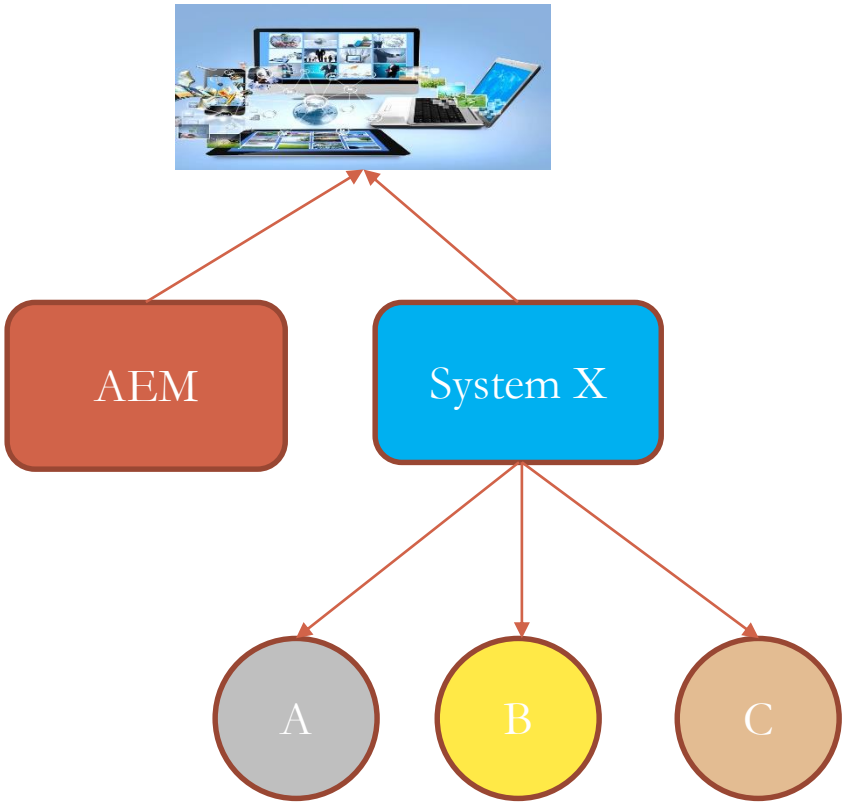
- AEM would very well be one of the backend system providing services to other systems
- Beware of the AEM internal working details when deploying AEM in this mode
- E.g., how can AEM deliver targeted content as service???



AEM as provider system

AEM as independent system in the stack

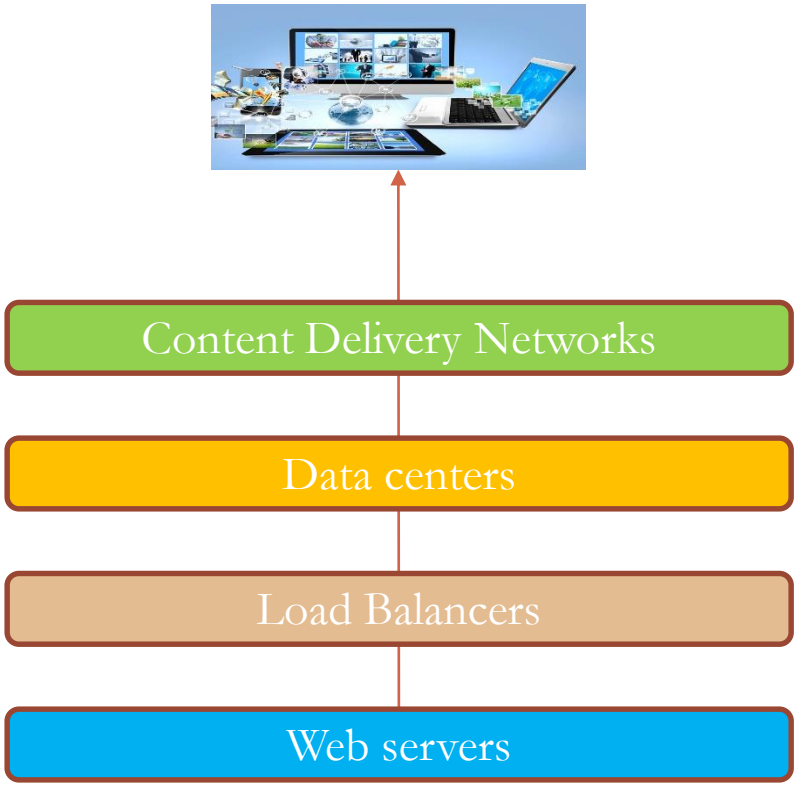
- AEM could also be an independent system providing services to be consumed directly from the client devices
- AJAX calls directly to AEM to pull content (becoming popular in sites implementing client MVC frameworks)
- Beware of the security implications when deploying in this mode



AEM as independent system

The de facto layers in the eco system

- Always consider the de facto layers in the eco system
- These are almost always there and has to be planned for in the overall layout
- With AEM, these layers impacts some of the key decisions made
- Remember AEM has dispatcher and make attempt to cache everything that goes out of AEM
- Anything that's cached in dispatcher can be cached in CDN



The de facto layers

FEATURE MAPPING

Product selection and mapping the features needed to the capabilities

AEM only vs. AEM + others

List out all the features needed and map it to the capabilities

- Consider all the features required for the overall solution. Features include capabilities like Security, Multi-lingual, Analytics, Search, Targeting, so on…
- AEM provides capabilities for most of these features which may or may not suffice for the given requirement
- Beware leveraging some of these features from AEM might require additional licensing

Validate objectively

Adobe would try to goad you into using their stack. While using products from the same suite has its advantages, it might have serious limitations especially when your overall solution is a complex mix of multiple systems

Validate with POC's

You might need few proof of concepts if you are choosing different products to complete the feature requirements

- Some of the functionalities within AEM are tightly integrated
- For example, AEM targeting depends on campaign management and Profiles. In case you make a choice to use Profile from a different system and targeting from AEM, be sure that the fitment is correct
- Work of the details to the most granular level possible at this stage
- Its wise to plan few proof of concepts to validate any assumptions that are taken

CONTENT STRUCTURE

Almost all design aspects link back to the content structure

Content structure

Almost all minor design aspects link back to the content structure

- The content structure impacts security design, template finalization, component visibility, content flow, …
- Remember content stored in JCR in hierarchical
- Keep the content structure as simple as possible and contain cross linking of content to the minimal
- Keep the structure flexible… provide to maintain more languages, companies, departments…

Some technical stuff to remember

- There are no rigid primary keys in JCR storage. Do not depend on them
- You cannot store null values. Property not present means value is null or empty or non-existent
- Design to keep the queries to run on nodes nested as deep as possible in the JCR structure

INTEGRATION

Correct integration approach is the key to the success of the implementation

Integration approaches

Move the integration as close to the browser as possible

Integrate in the browser: Possible when the services to be integrated with are exposed over http and available for consumption from the browser

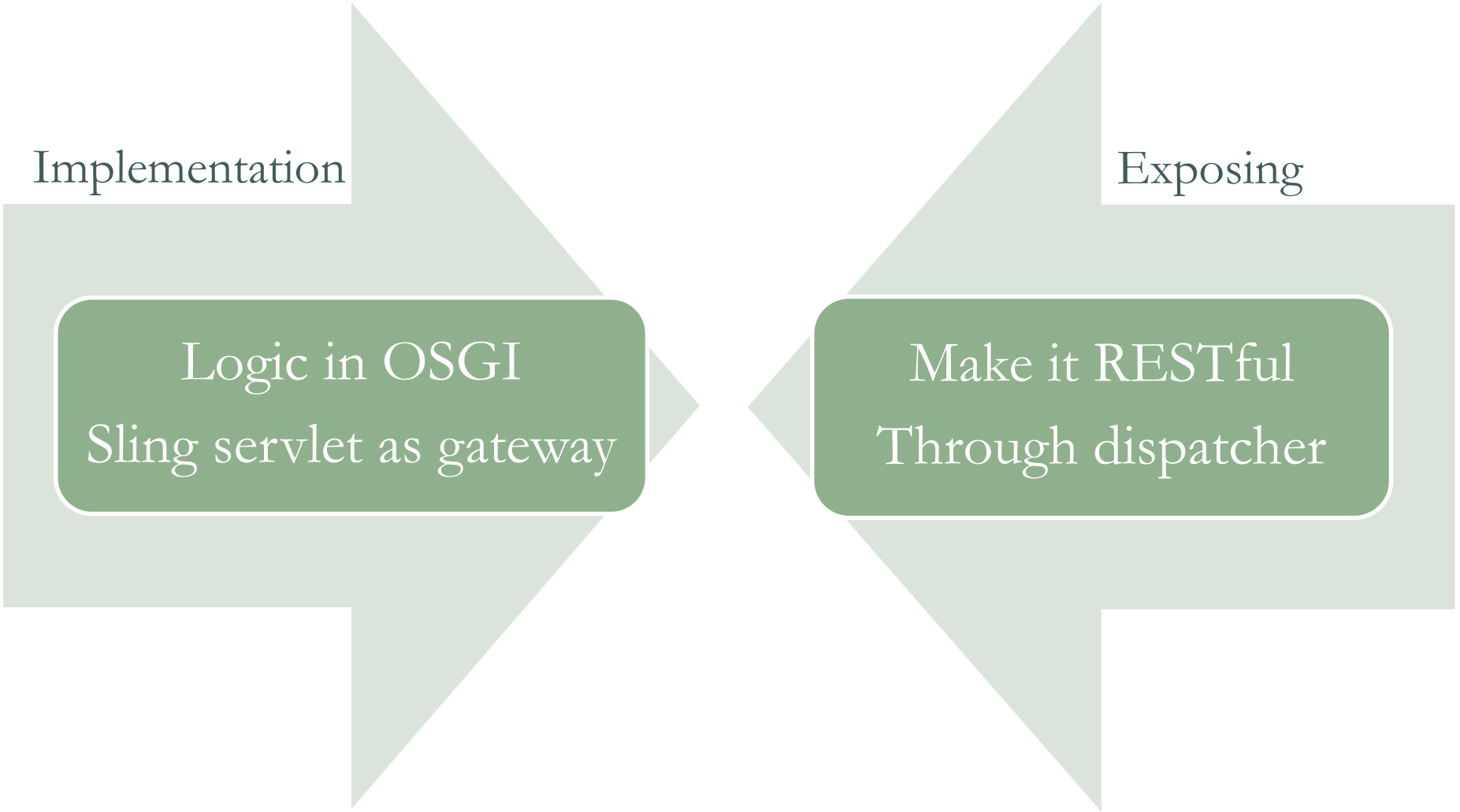
Integrate at the dispatcher: Possible when the services to be integrated with are available over http; but are not exposed to the browser

Integrate at the Sling layer: Maintain caution to avoid making this layer logic intensive. Simple REST end points can be consumed at sling layer

Integration at the OSGI: Use this for logic intensive integration. Create wrapper bundles to be consumed from multiple places

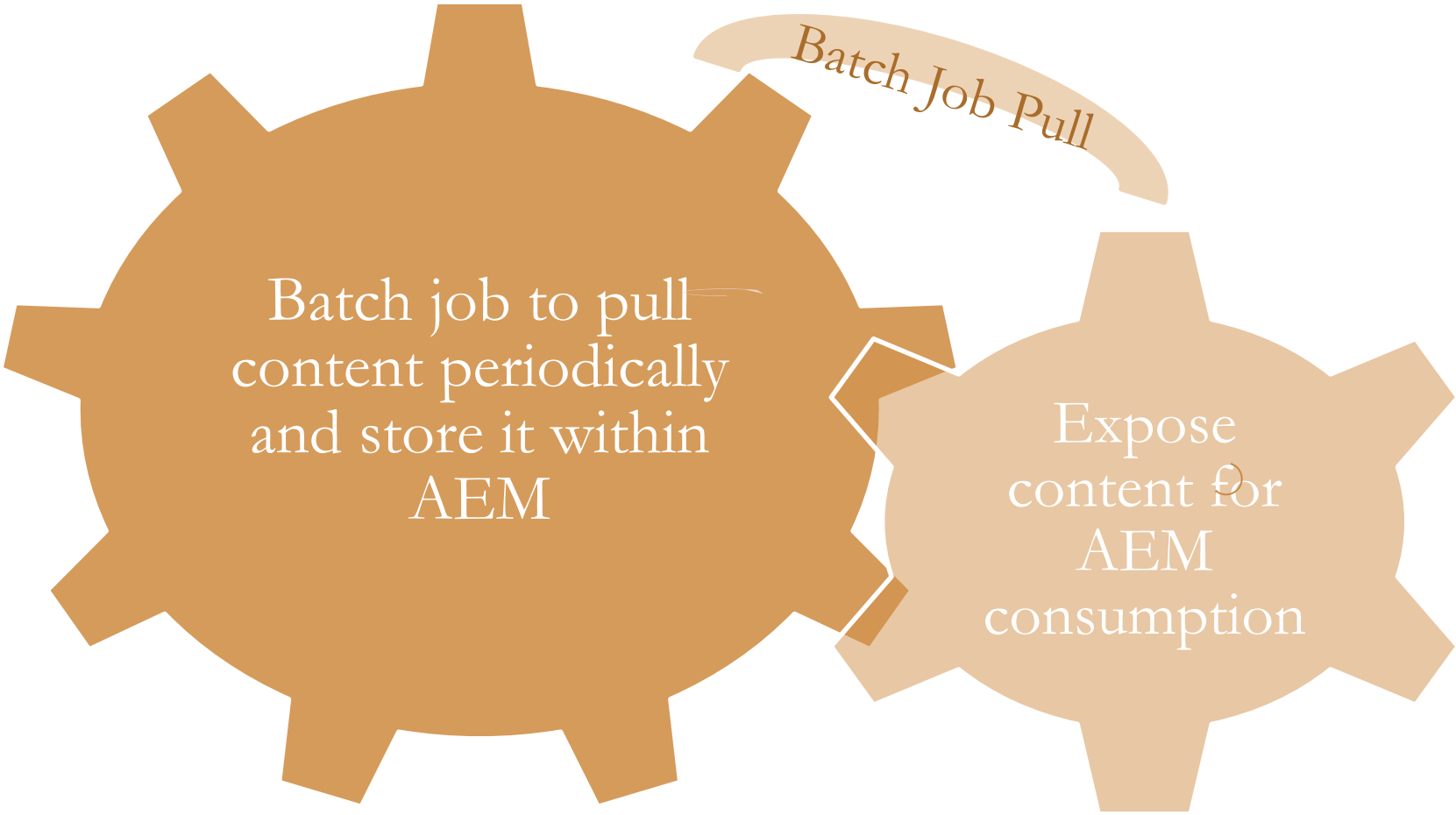
When AEM exposes services

Expose them as REST Services



Less frequently changing content

Use batch jobs to periodically pull content into AEM



DEPLOYMENT MODEL

Plan the deployment model well in advance…

Deployment drives some design choices

The big Mongo vs. Tar war in AEM 6.1… and Tar wins hands down

The real war is between DB Admins & AEM Architects...

Deployment differs between Author and Publish

Deployment drives dispatcher to publish mapping & replication flows

Is author preview good enough? Do you need a real-time preview before publish

User generated content needs special attention

Have pre-prod exactly the same as configuration as prod

MAINTENANCE

Remember: AEM will breakdown if maintenance activities are not carried out periodically

Backup, Purge, Compact

A lot happens inside AEM and is prone to corruption if not maintained properly

- AEM backup is copying the file system
- Repository structure to align with backup policy
- Provision additional systems to run backup on if preferred
- Plan separate backups for author and publish

Backup

- Version, workflows and audit logs needs to be purged
- Finalize purge policy and configure purge activities
- The repository would quickly grow to a huge size if not for purging

Purge

- This requires AEM downtime.
- Best approach is to do it on one instance and clone that instance to other instances (publish)
- Keep the publish instance homogenous (identical clones)

Compact

THANK YOU

Feedback and suggestions welcome. Please write to ashokkumar_ta / ashokkumar.ta@gmail.com