Deep dive into the AEM upgrade process - part 1: AEM itself

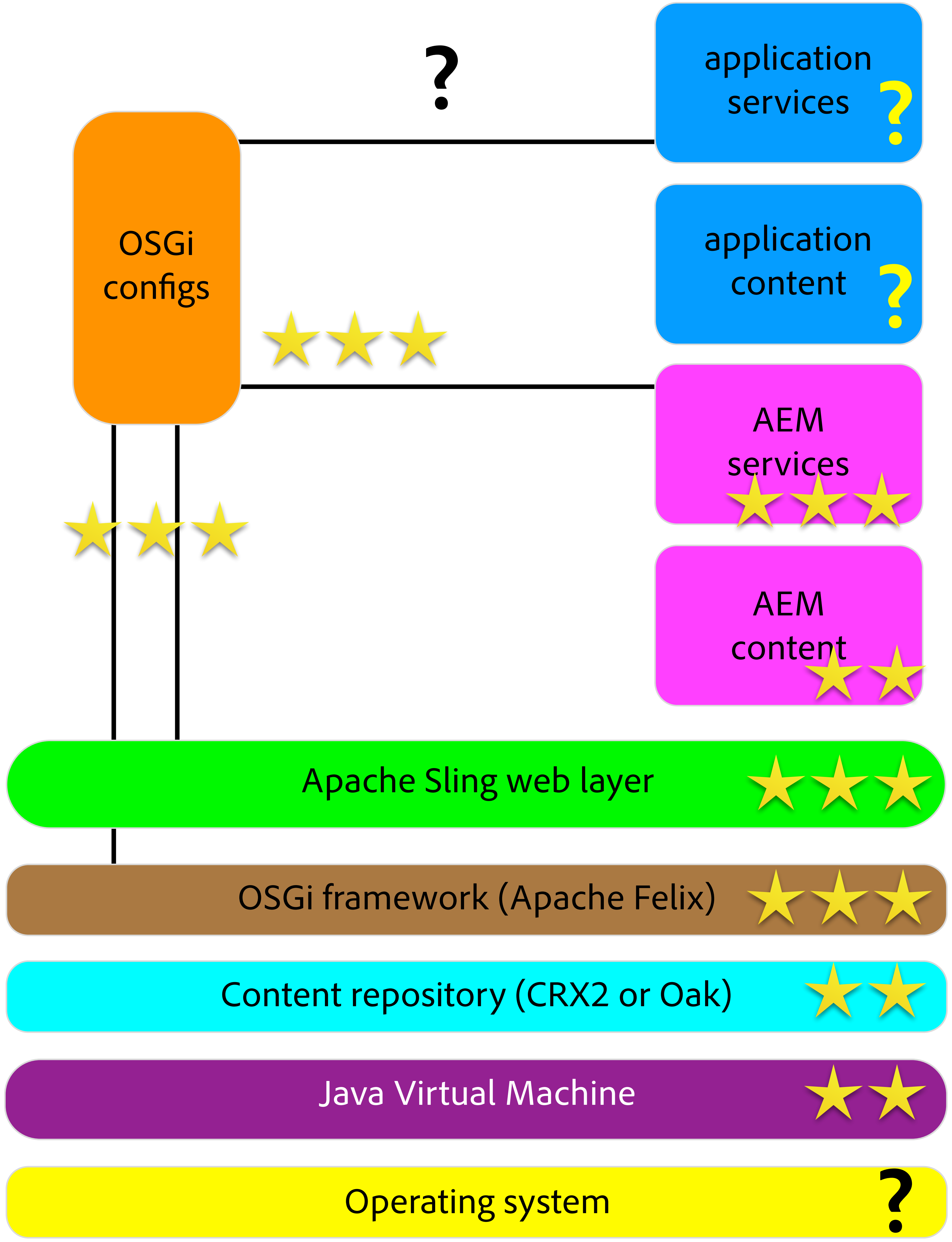
Bertrand Delacretaz | @bdelacretaz | Principal Scientist, AEM R&D team

Gems on Adobe Experience Manager - October 2015

slides rev 2015-10-14

Bē
GS&P Beta Group

Upgrading a modular system (with support for N old versions)



Each module should be self-upgrading and backwards compatible. ★★★

Modules and content that are not upgradeable might be moved aside during the upgrade, and moved back in later.

Three stars under /libs and /apps but more work needed under /etc for example.

More work if migrating between storage options (CRX2, Oak, storage backends) but we have tools for that.

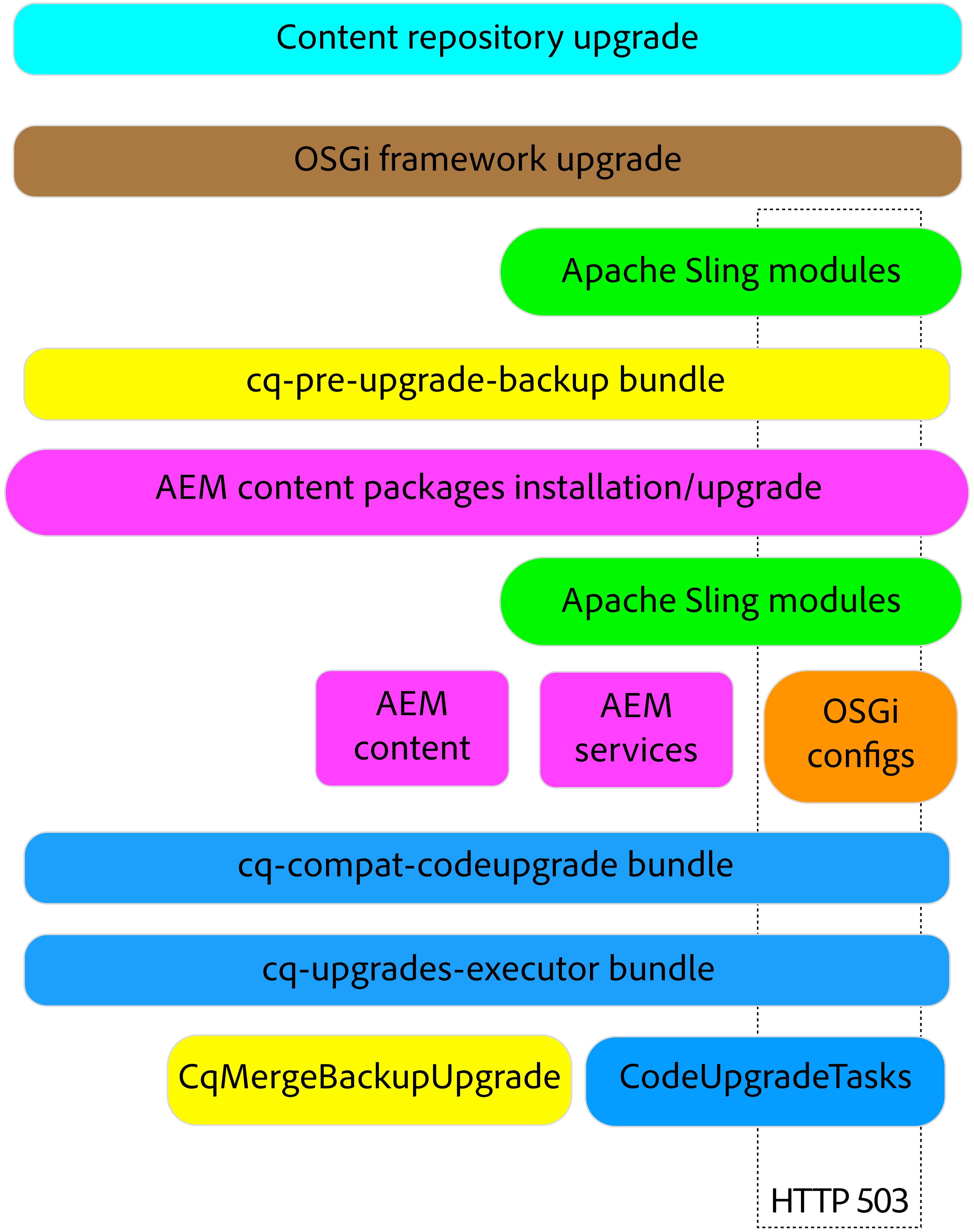
Might imply changes to old code

Might have subtle effects

The Upgrade Scenario

Bē
GMUNK

AEM upgrade scenario


Content repository upgrade
OSGi framework upgrade
Apache Sling modules
cq-pre-upgrade-backup bundle
AEM content packages installation/upgrade
Apache Sling modules
AEM content
AEM services
OSGi configs
cq-compat-codeupgrade bundle
cq-upgrades-executor bundle
CqMergeBackupUpgrade
CodeUpgradeTasks
HTTP 503

Upgrade Low-level data structures, indexes.
Migrate from CRX2 to Oak? Migrate storage back-ends? Part 2 of this session.

Replace bootstrap bundles, upgrade framework data structures
Start replacing (tons of) bundles

Backup / move aside selected content to be restored after upgrade.

The bulk of the upgrade

CodeUpgradeTasks are doing less work in newer AEM versions, the goal is to need less and less of them.

Register *CodeUpgradeTask* services

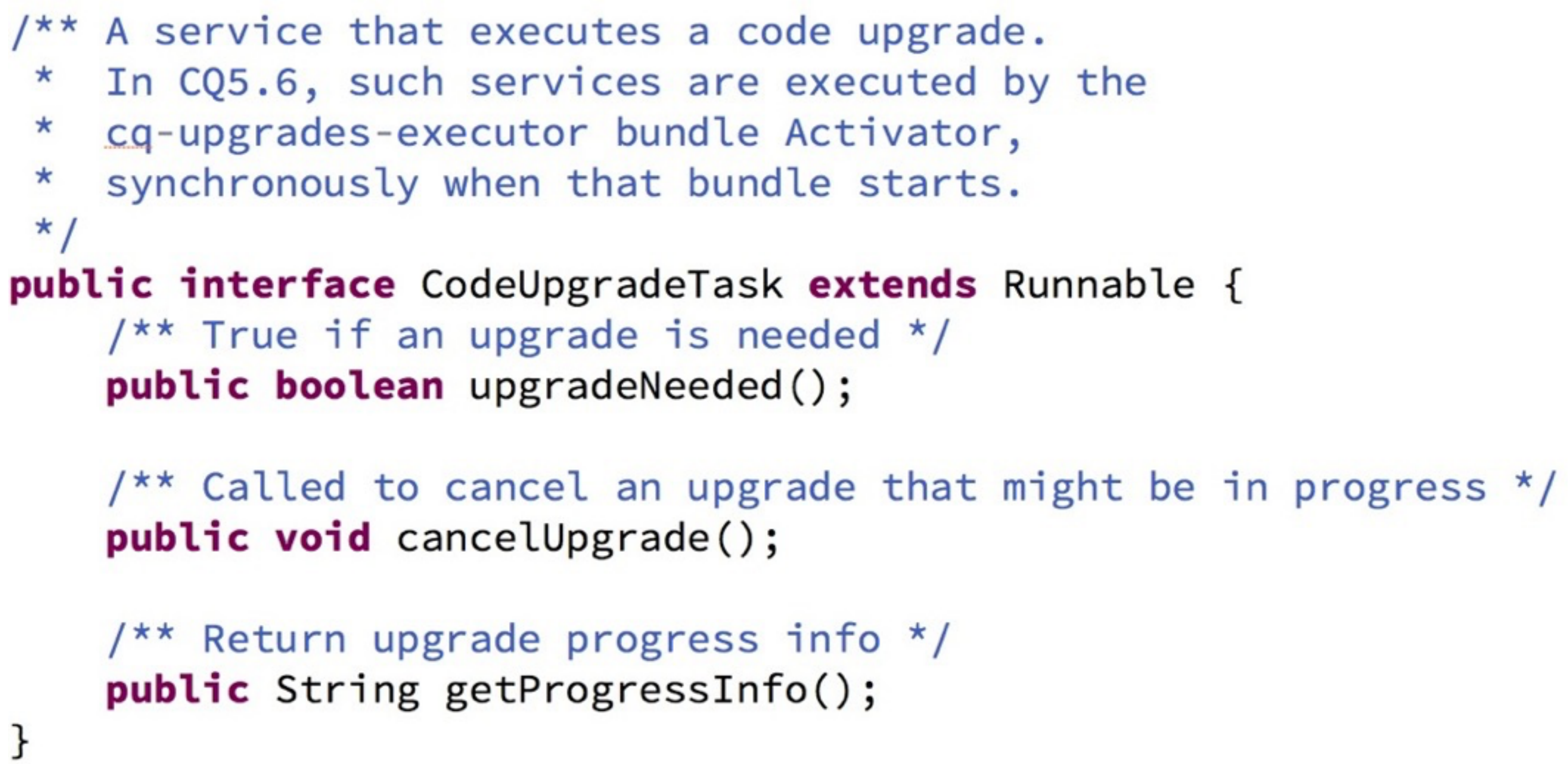
Execute *CodeUpgradeTask* services:
Merge and upgrade content, run custom upgrade tasks if registered.

Sling startup filter + disabler cause 503 status during startup.

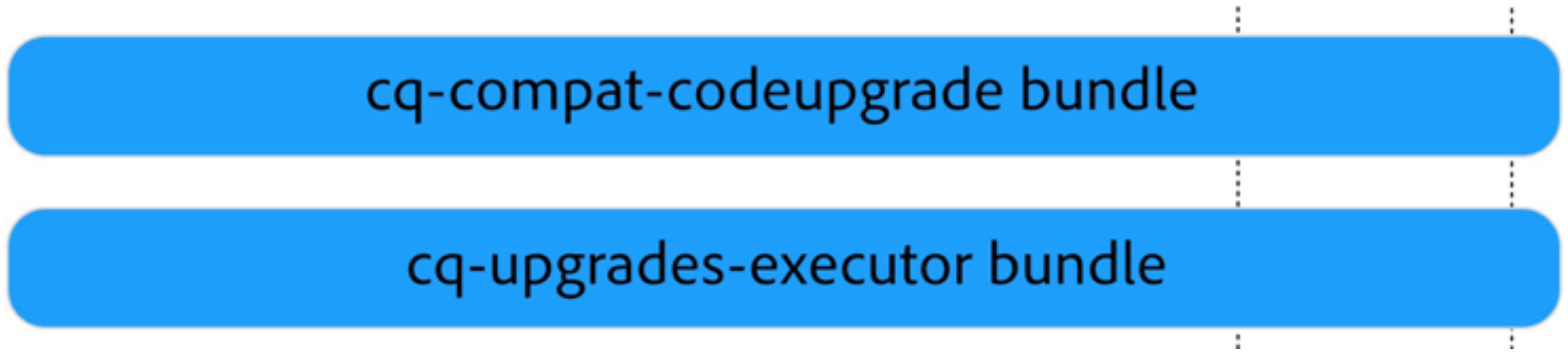
Customizing the upgrade process

Be
Sagmeister & Walsh

CodeUpgradeTask service



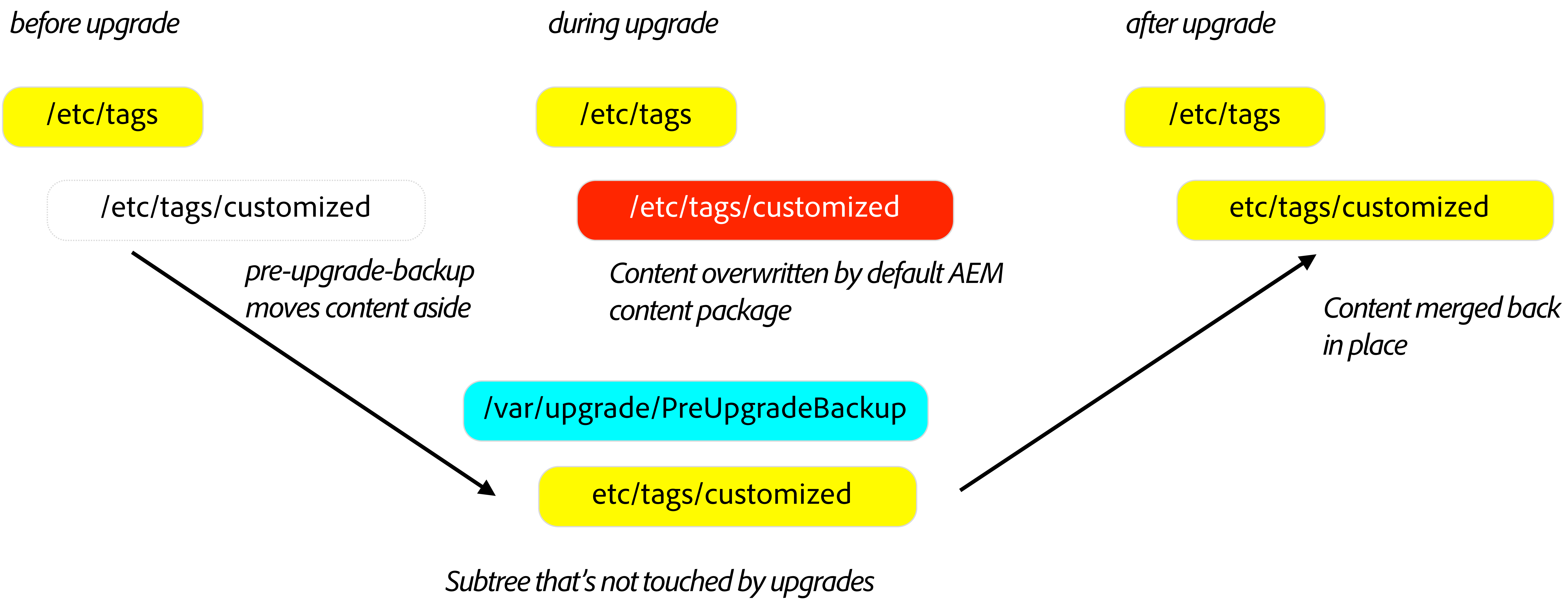
```java
/** A service that executes a code upgrade.
 *  In CQ5.6, such services are executed by the
 *  cq-upgrades-executor bundle Activator,
 *  synchronously when that bundle starts.
 */
public interface CodeUpgradeTask extends Runnable {
    /** True if an upgrade is needed */
    public boolean upgradeNeeded();

    /** Called to cancel an upgrade that might be in progress */
    public void cancelUpgrade();

    /** Return upgrade progress info */
    public String getProgressInfo();
}
```



Executed via their run() method by the cq-upgrades-executor bundle's Activator, synchronously during that bundle's startup, in order of their service.ranking (540 for CQ5.4, 561 for CQ5.6.1 etc.)

The cq-upgrades-executor bundle runs late in the startup process, all services should be available.

Pre-upgrade backup / post-upgrade merge


before upgrade
during upgrade
after upgrade
/etc/tags
/etc/tags
/etc/tags
/etc/tags/customized
/etc/tags/customized
etc/tags/customized
pre-upgrade-backup moves content aside
Content overwritten by default AEM content package
Content merged back in place
/var/upgrade/PreUpgradeBackup
etc/tags/customized
Subtree that's not touched by upgrades

Configured under /var/upgrade/PreUpgradeBackup/configuration - a config there replaces the default one. See appendix for configuration example.

TESTING your upgrade is critical.
You do have automated tests, right?

picture: yang yu on fotolia.com

Appendix

Appendix: testing upgrade tasks

WARNING all of this is potentially DANGEROUS for your system - use on throwaway instances only.

Upgrade mode can be forced with
java -Dorg.apache.sling.launchpad.startupmode=update
(use with -nofork) on the quickstart command line.

Watch the **upgrade.log and error.log** to see these tasks' activity.

The upgrade tasks **store state under /var/upgrade/status**, deleting properties of that node causes the corresponding tasks to be rerun next time the cq-upgrades-executor bundle starts.

```
$ curl -u admin:admin http://localhost:4502/var/upgrade/status.tidy.json
{
  "Cq61StartupCodeUpgrade": "Wed Oct 14 2015 10:20:48 GMT+0200",
  "Cq60StartupCodeUpgrade": "Wed Oct 14 2015 10:20:48 GMT+0200",
}
```

To run the upgrade tasks **without restarting**, set the /var/upgrade/status/upgradesExecutor.forceUpgrades property to true and restart the cq-upgrades-executor. Don't forget to delete that property after your tests.

Appendix: testing pre-upgrade backup

WARNING all of this is potentially DANGEROUS for your system - use on throwaway instances only.

To **test the pre-upgrade backup**, set the /var/upgrade/status/preUpgradeBackup.force property to true, restart the com.adobe.cq.cq-pre-upgrade-backup bundle and watch the upgrade.log.

```
UpgradeBackup Startup mode is UPDATE, backup will be done
BackupConfigImpl Creating default configuration at /var/upgrade/PreUpgradeBackup/configuration
BackupConfigImpl Only paths found under /var/upgrade/PreUpgradeBackup/configuration will be backed up
BackupConfigImpl /etc/tags added to paths to backup due to config node /var/upgrade/PreUpgradeBackup/configuration/etc/tags
BackupConfigImpl /etc/cloudservices added to paths to backup due to config node /var/upgrade/PreUpgradeBackup/configuration/etc/cloudservices
BackupConfigImpl path to overwrite '/etc/cloudservices/proxy' set for /etc/cloudservices
UpgradeBackup Backing up content before upgrade, destination=/var/upgrade/PreUpgradeBackup/20151014_102950_793/to-process
UpgradeBackup Moving /etc/tags to /var/upgrade/PreUpgradeBackup/20151014_102950_793/to-process/etc/tags
UpgradeBackup Source path not found, nothing to backup: /etc/cloudservices
UpgradeBackup Backup done
```

To test **the merging** of that backed up content, run the upgrade services as described on the previous slide, after deleting the /var/upgrade/status/CqMergeBackupUpgrade property.

Appendix: example pre-upgrade backup configuration

```
"configuration": {
  "ignoreDefaultConfig" : "true",
  "etc": {
   "workflow": {
      "models": {
        "dam": {
          "update_asset": {
            "overwritePaths": ["/etc/workflow/models/dam/update_asset"]
          }
        }
      }
    }
  }
}
```

This configuration node at /var/upgrade/PreUpgradeBackup/configuration defines that /etc/workflow/models/dam/update_asset is backed up, and when merging the old content overwrites the new one.

For paths that do not have this overwritePaths setting, the merge is a tree merge where the smallest unit is a node, properties are not merged but nodes that were present before the upgrade and absent after it are moved back in place.

Appendix: useful links

Upgrading to AEM 6.1:
https://docs.adobe.com/docs/en/aem/6-1/deploy/upgrade.html
Watch that page for additional backup/merge/troubleshooting information, coming soon.

CQ 5.6 upgrades questions:
http://dev.day.com/content/ddc/en/gems/upgrading-to-56.html
Most of that is still valid, will be updated for 6.1 and linked from the above page.

CodeUpgradeTask javadocs:
https://docs.adobe.com/docs/en/aem/6-1/ref/javadoc/com/day/cq/compat/codeupgrade/CodeUpgradeTask.html
The name "code upgrade" is a bit misleading, those tasks can upgrade anything they need to.

Adobe