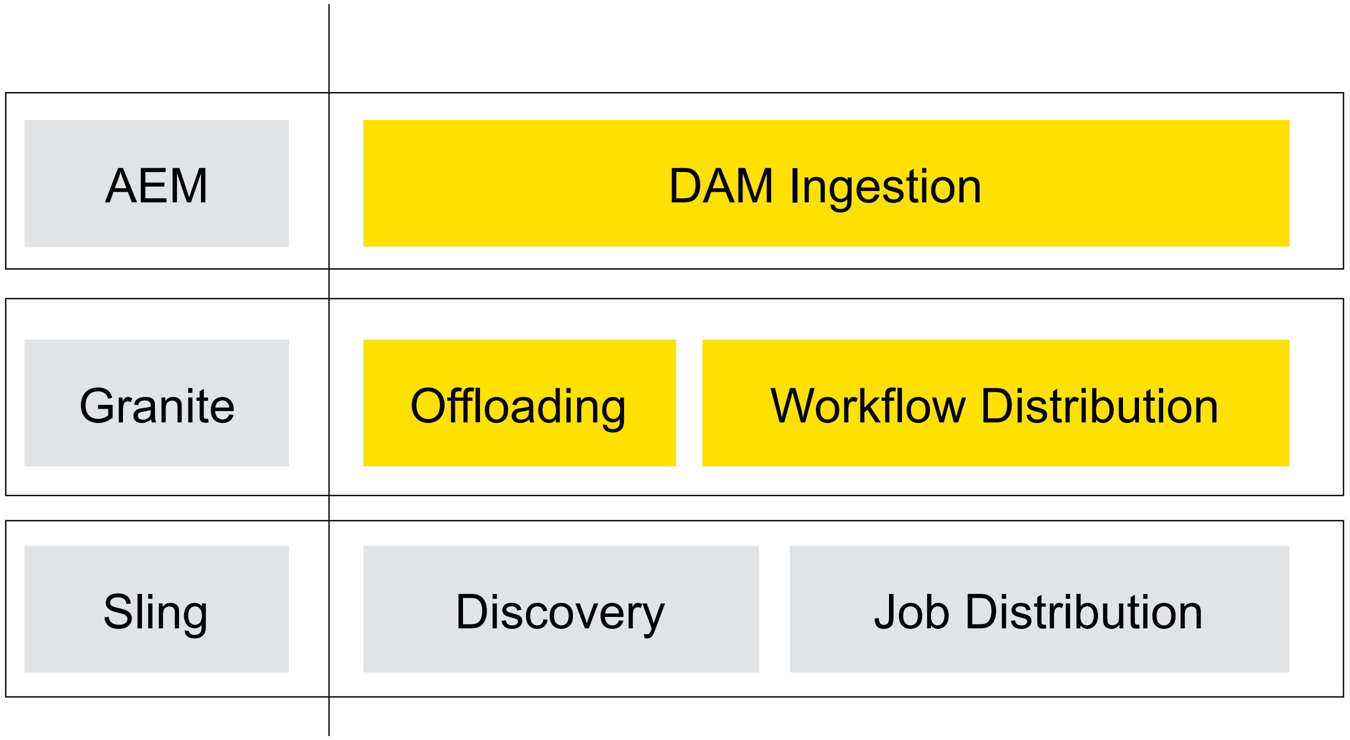
Granite - Offloading

Marc Pfaff | Basel

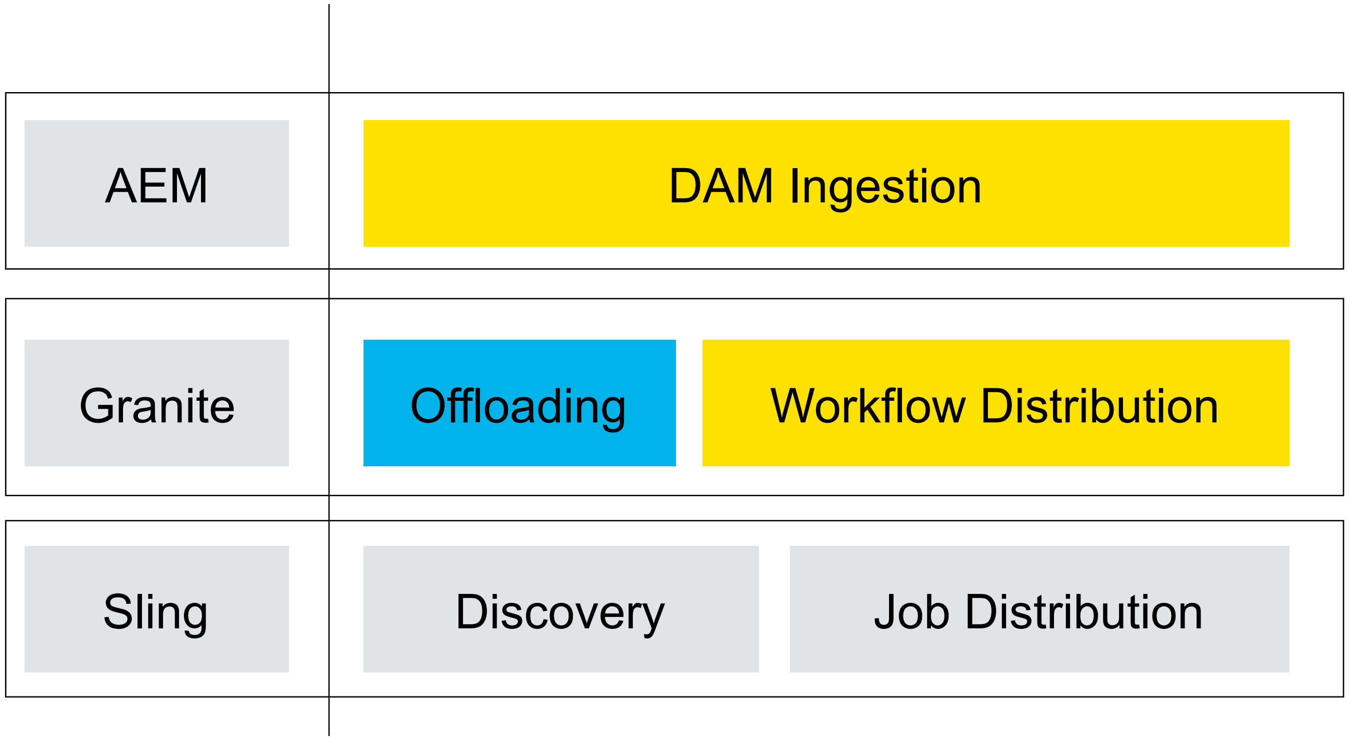
Why Offloading

- Scalability in AEM:
  - DAM Ingestion
  - Non-Clustered installation requirement
- The term Offloading:
  - In AEM used for all things job distribution and topology in clustered and non-clustered installations, e.g. ‘Offloading Browser’
  - More technically it’s ‘only’ a little add-on in Granite to Sling Job Distribution for handling non-clustered installations

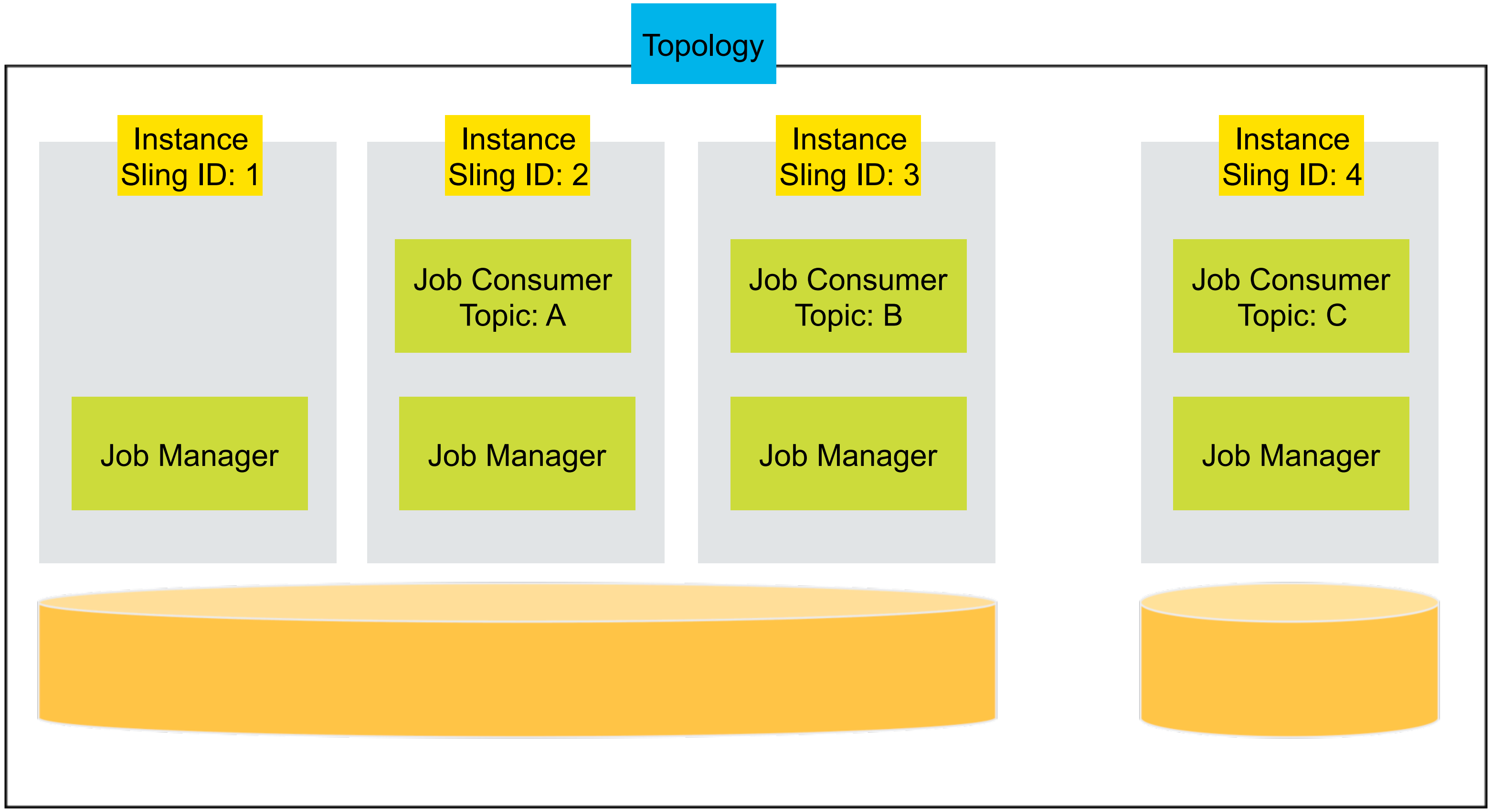
Overview – Where to fit in the stack



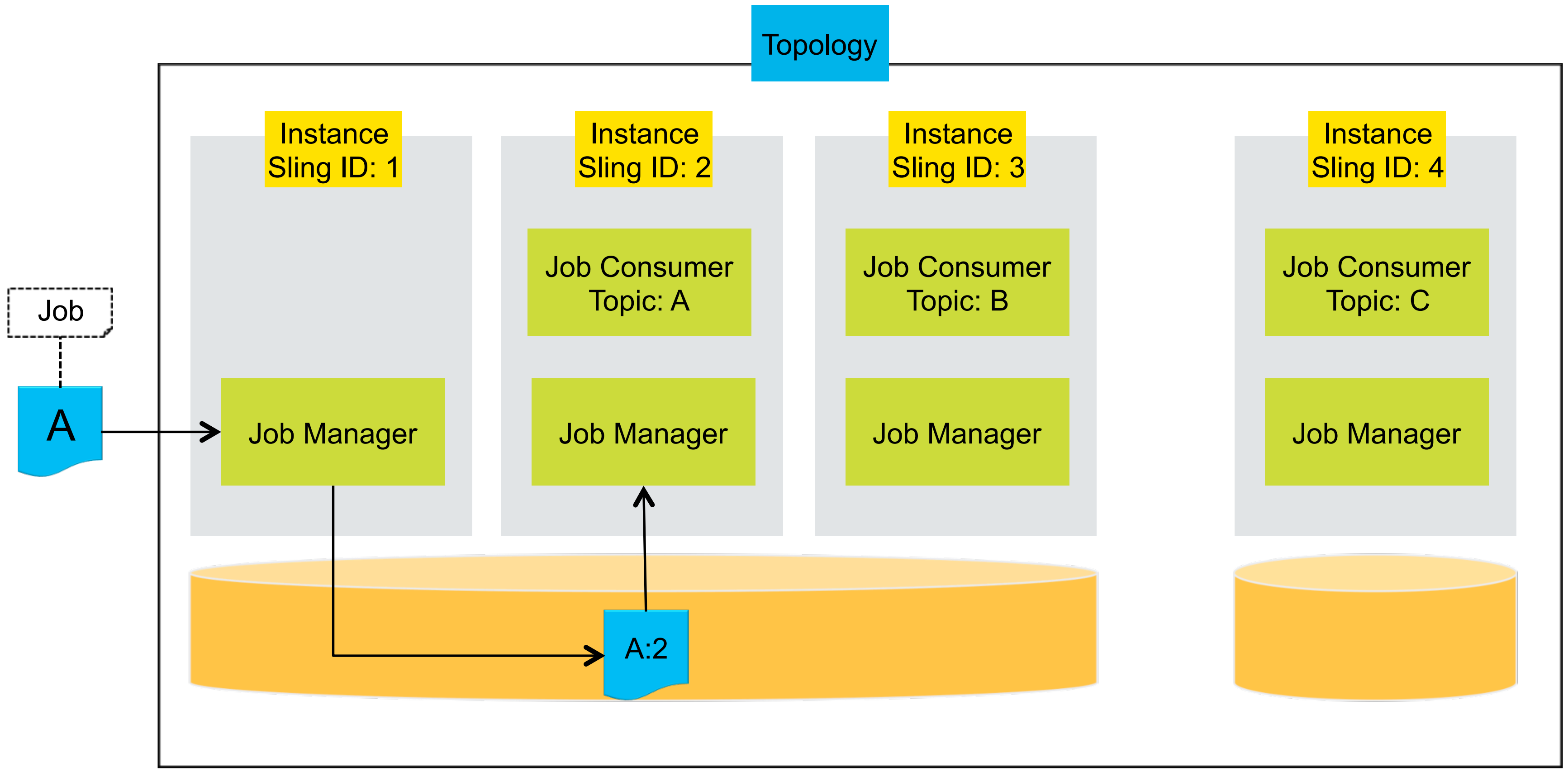
Overview – Where to fit in the stack



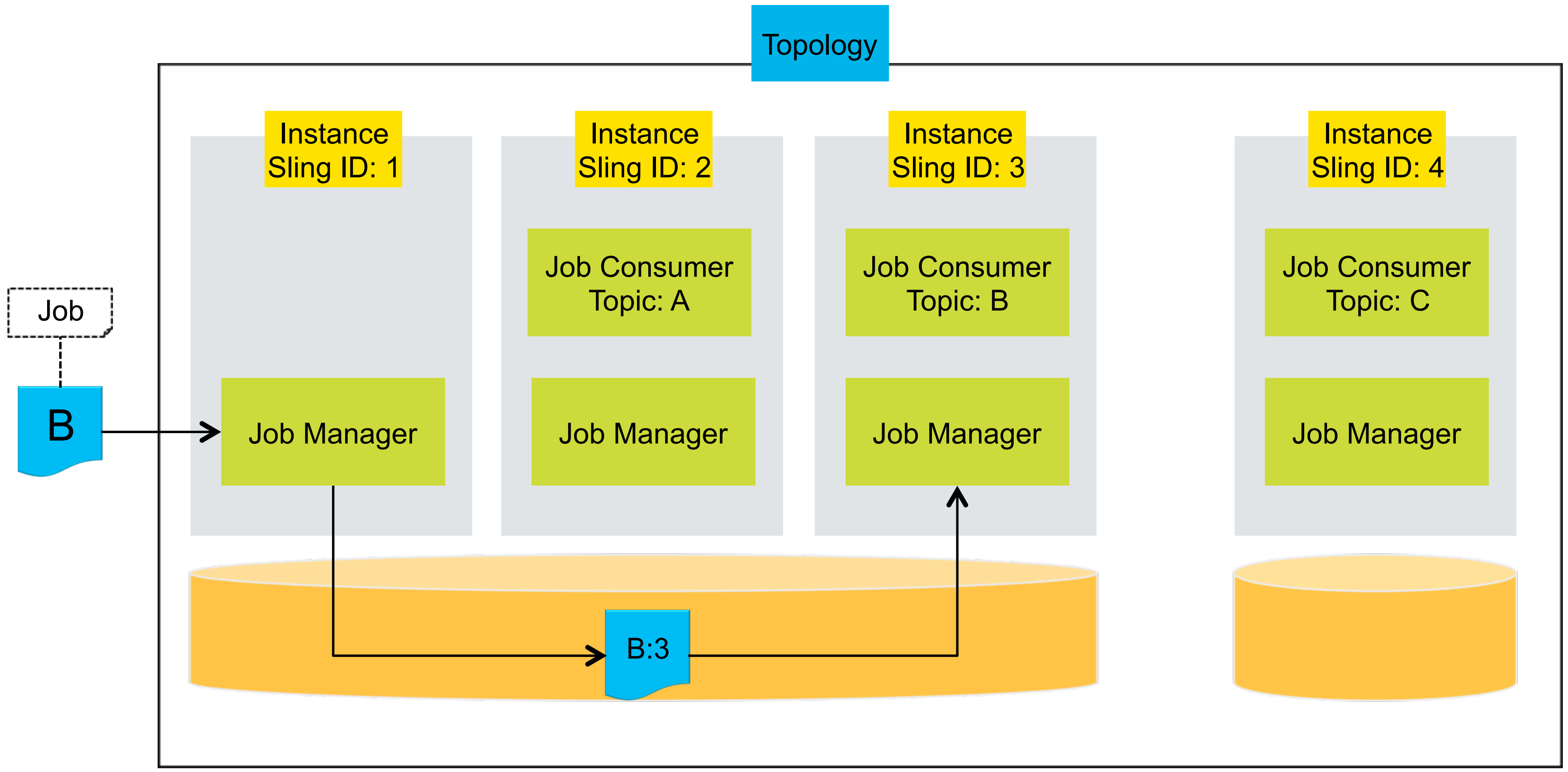
Sling Job Distribution



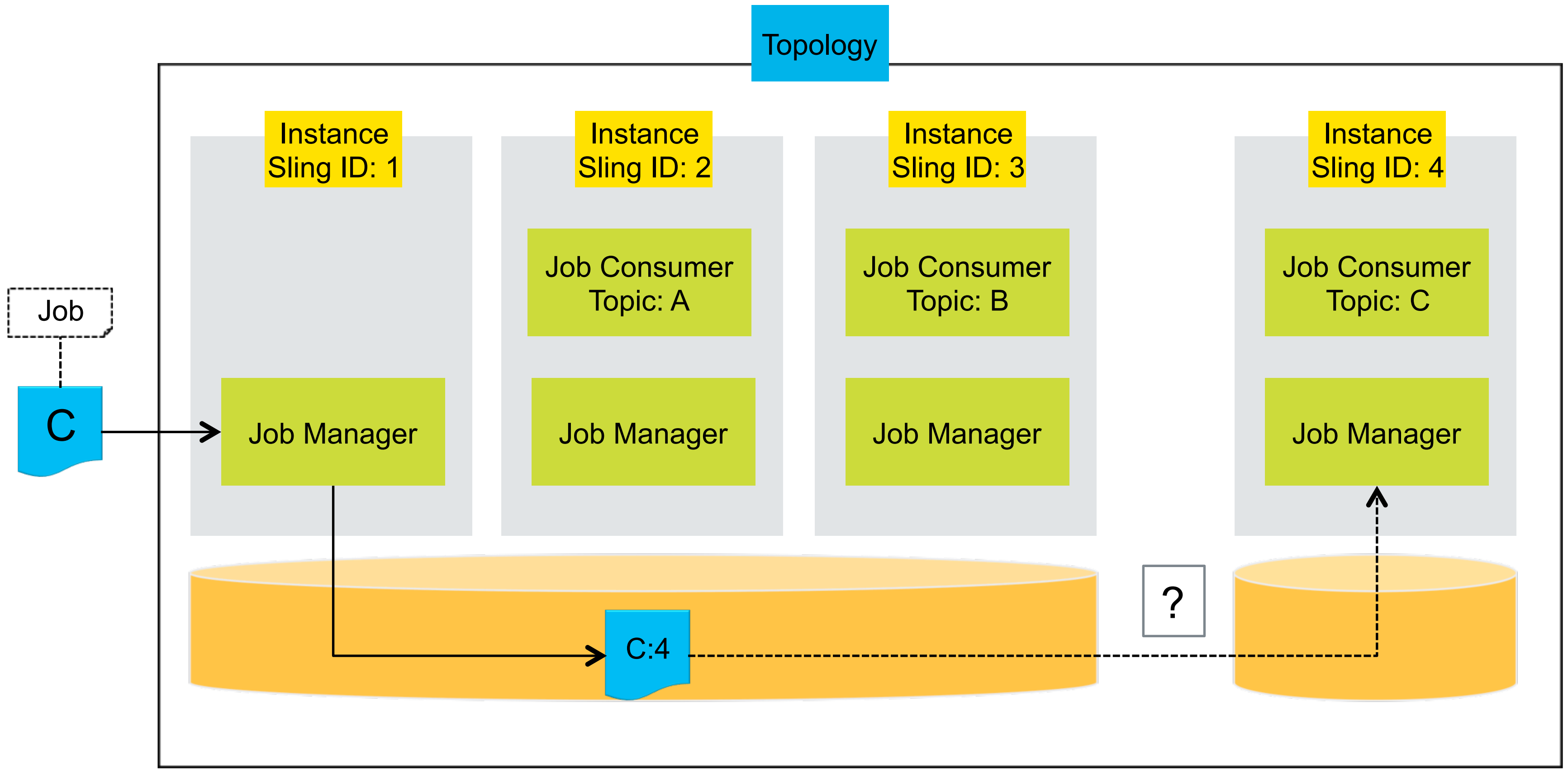
Sling Job Distribution



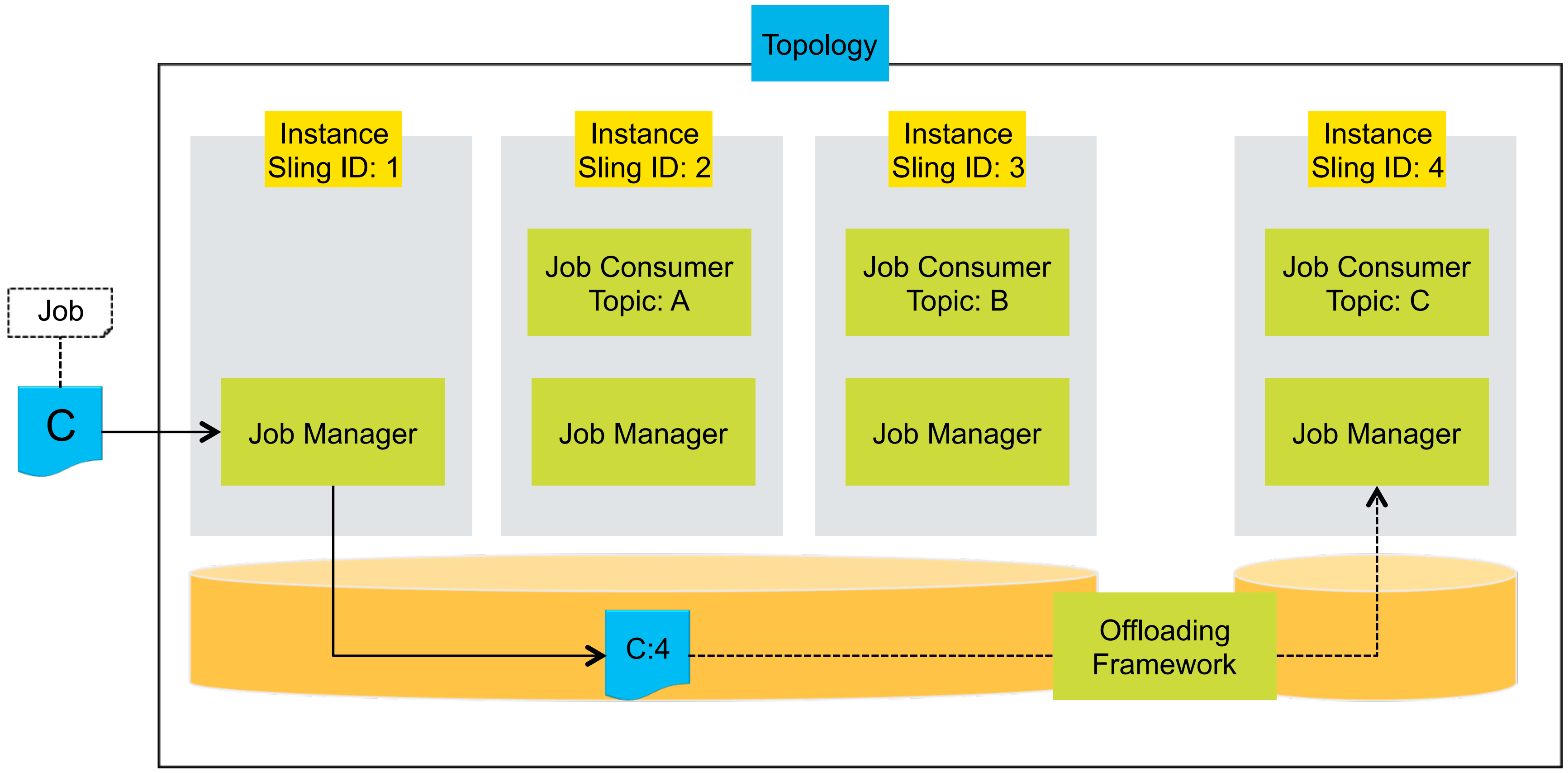
Sling Job Distribution



Sling Job Distribution

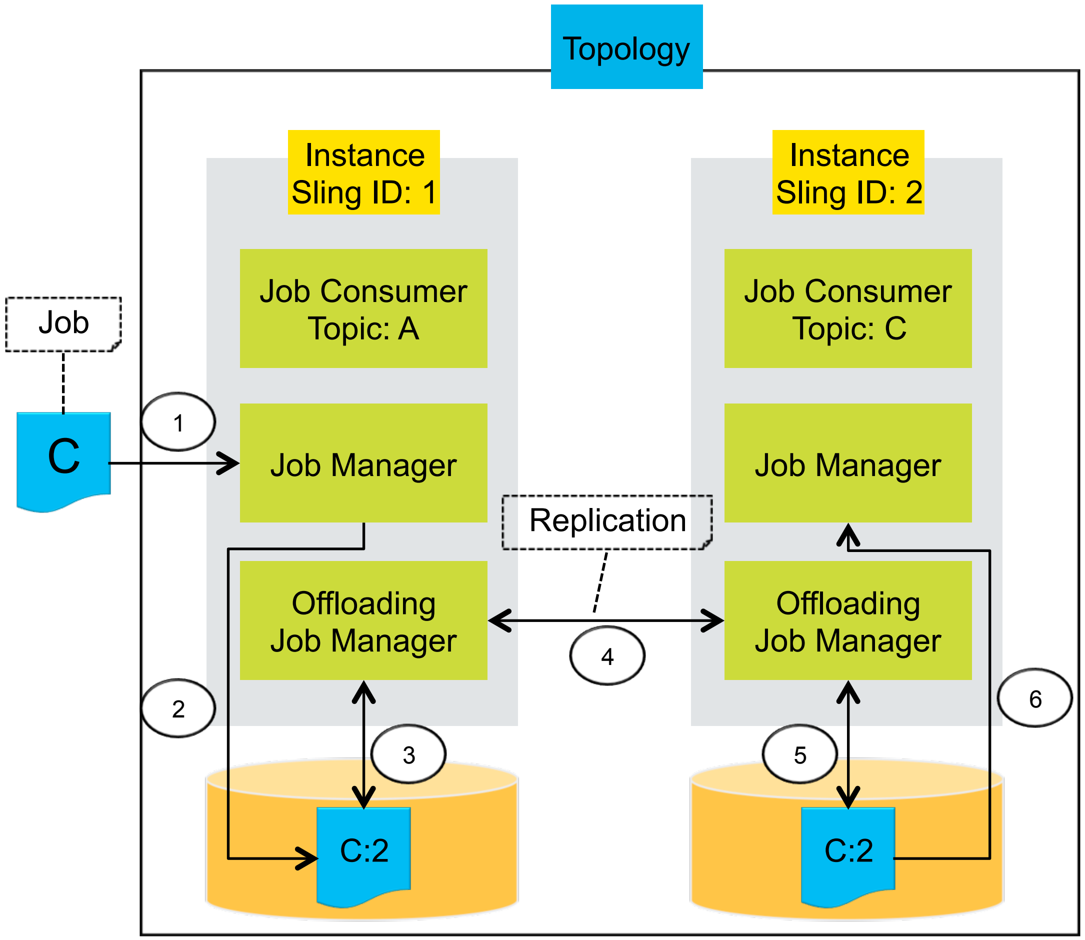


Sling Job Distribution



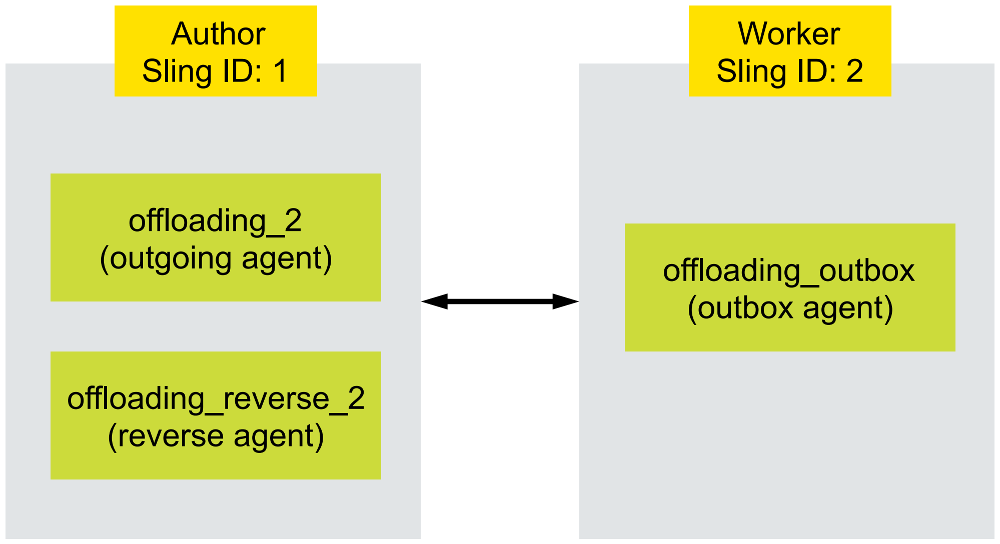
Offloading Framework - Overview

- Detects offloading jobs
- Transport of job and job payload between origin and target instance
- Uses replication for the transport
- No distribution of jobs
- No execution of jobs



Offloading Framework – Replication

- Replication agents are created automatically
- Uses naming convention
- Needs manual adjustments (replication user)



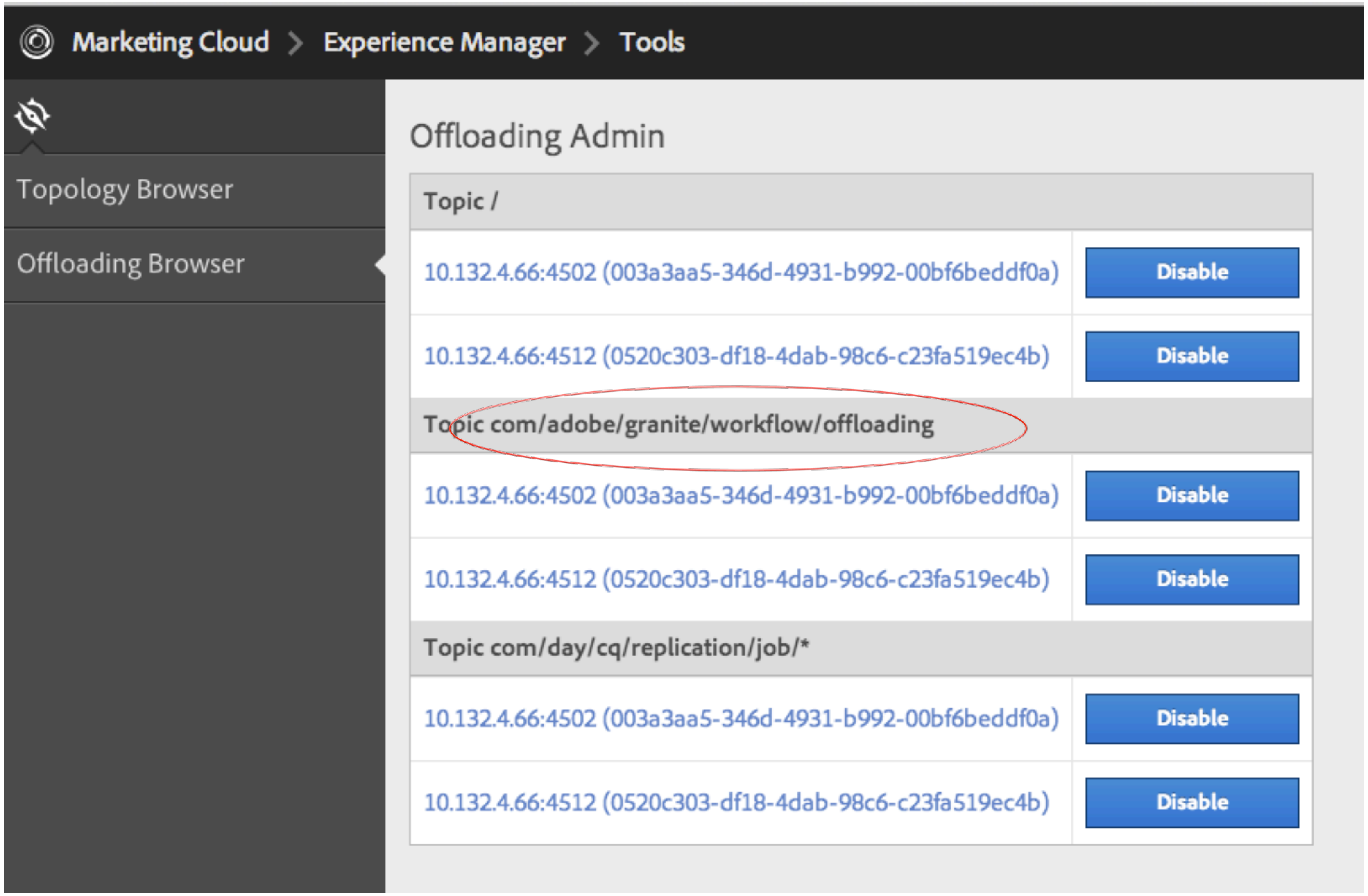
Offloading Framework – Payload

- Defines the content to be transported between the instances
- Properties on the job payload
- Takes comma separated list of paths
- Used to build the replication package
- The job itself is implicitly added by the offloading framework
- Offloading job input
  - Property: OffloadingJobProperties. INPUT_PAYLOAD (offloading.input.payload)
- Offloading job output
  - Property: OffloadingJobProperties. OUTPUT_PAYLOAD (offloading.output.payload)

Offloading Browser - UI

- Configures Job Consumers
  - Configures the topic white/black listing properties of each instance
  - What jobs to execute on what instance
- Configures the distribution
- Configuration applies for both, clustered and non-clustered installations

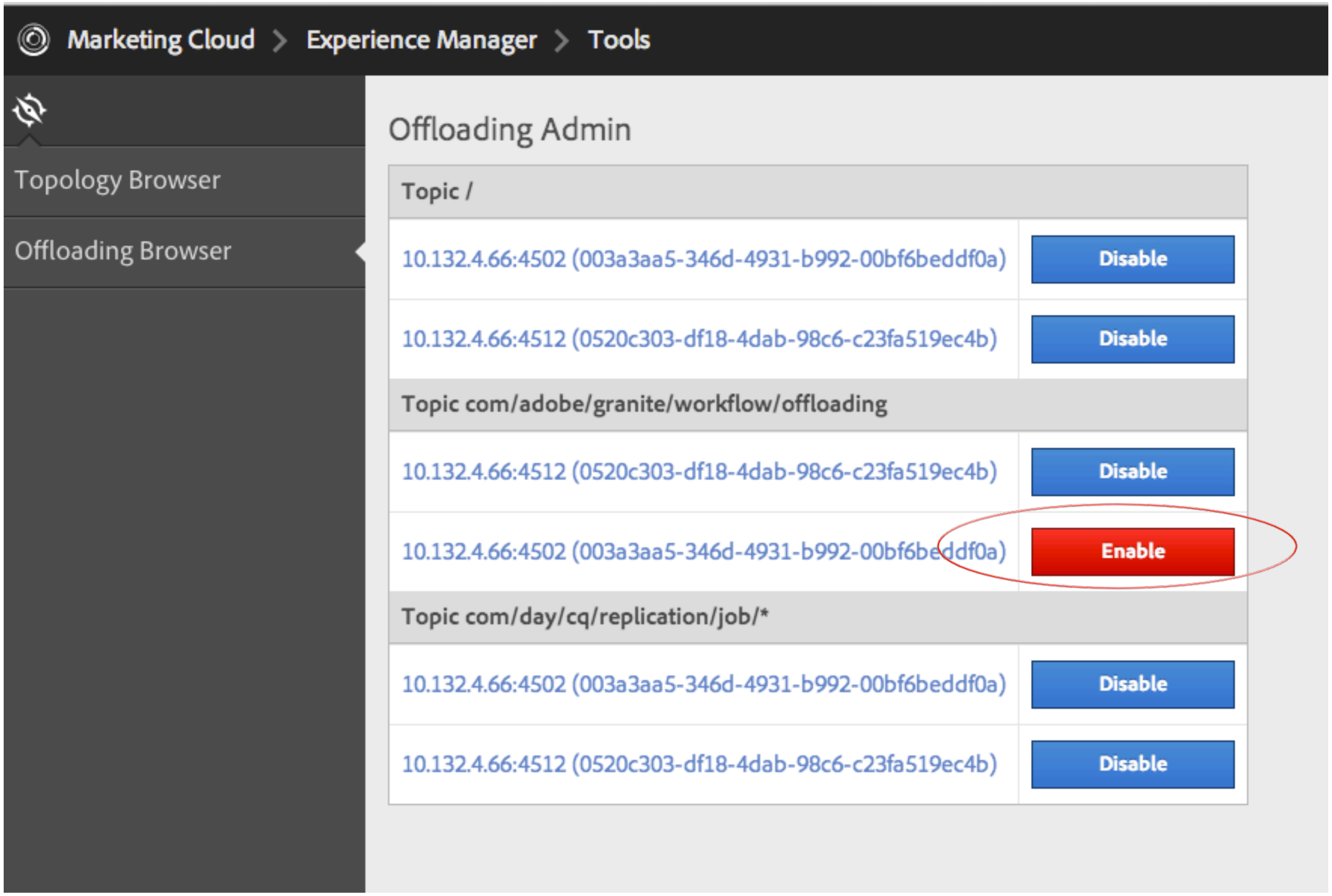
Offloading Browser - UI



Offloading Admin

Topic /	
10.132.4.66:4502 (003a3aa5-346d-4931-b992-00bf6beddf0a)	Disable
10.132.4.66:4512 (0520c303-df18-4dab-98c6-c23fa519ec4b)	Disable
Topic com/adobe/granite/workflow/offloading	
10.132.4.66:4502 (003a3aa5-346d-4931-b992-00bf6beddf0a)	Disable
10.132.4.66:4512 (0520c303-df18-4dab-98c6-c23fa519ec4b)	Disable
Topic com/day/cq/replication/job/*	
10.132.4.66:4502 (003a3aa5-346d-4931-b992-00bf6beddf0a)	Disable
10.132.4.66:4512 (0520c303-df18-4dab-98c6-c23fa519ec4b)	Disable

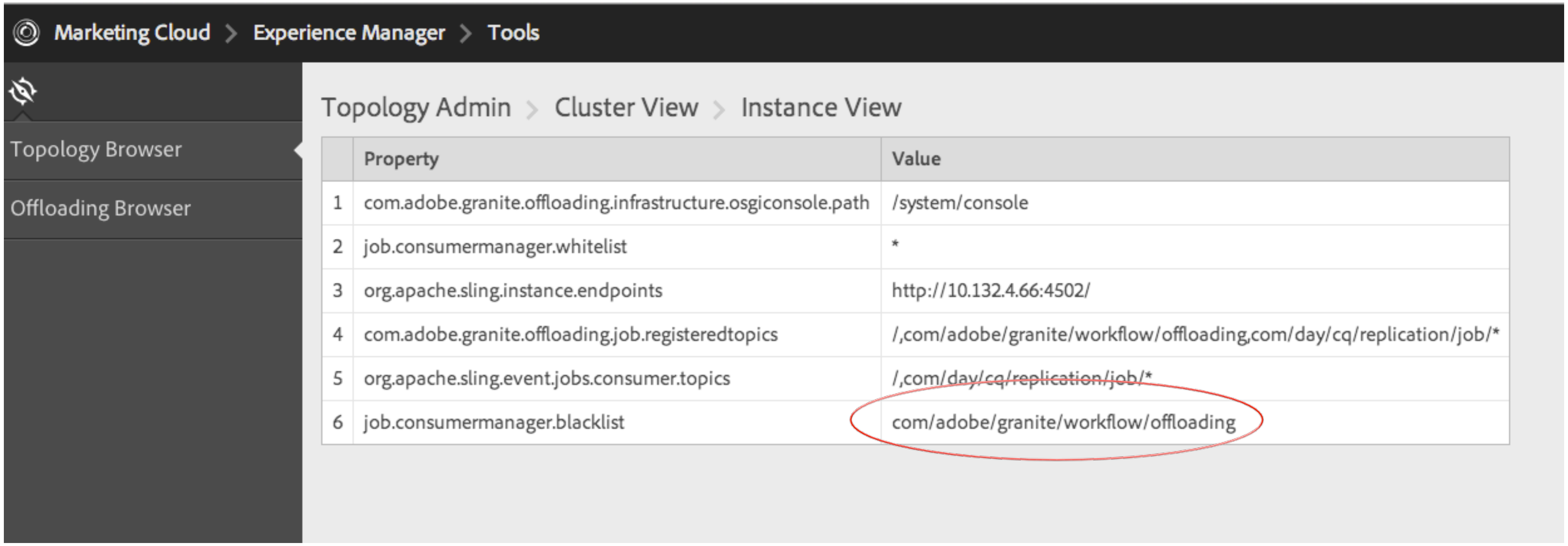
Offloading Browser - UI



Offloading Admin

Topic /	
10.132.4.66:4502 (003a3aa5-346d-4931-b992-00bf6beddf0a)	Disable
10.132.4.66:4512 (0520c303-df18-4dab-98c6-c23fa519ec4b)	Disable
Topic com/adobe/granite/workflow/offloading	
10.132.4.66:4512 (0520c303-df18-4dab-98c6-c23fa519ec4b)	Disable
10.132.4.66:4502 (003a3aa5-346d-4931-b992-00bf6beddf0a)	Enable
Topic com/day/cq/replication/job/*	
10.132.4.66:4502 (003a3aa5-346d-4931-b992-00bf6beddf0a)	Disable
10.132.4.66:4512 (0520c303-df18-4dab-98c6-c23fa519ec4b)	Disable

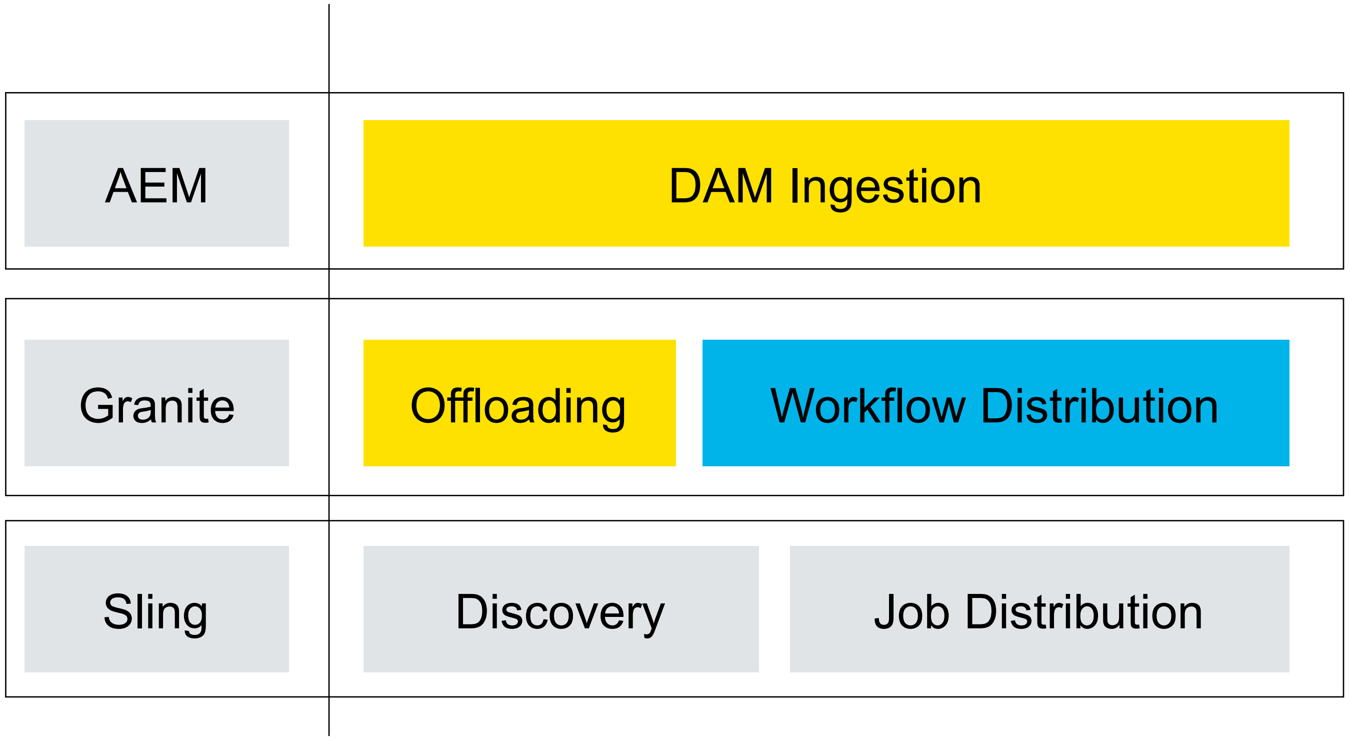
Offloading Browser - UI



Topology Admin > Cluster View > Instance View

	Property	Value
1	com.adobe.granite.offloading.infrastructure.osgiconsole.path	/system/console
2	job.consumermanager.whitelist	*
3	org.apache.sling.instance.endpoints	http://10.132.4.66:4502/
4	com.adobe.granite.offloading.job.registeredtopics	/,com/adobe/granite/workflow/offloading,com/day/cq/replication/job/*
5	org.apache.sling.event.jobs.consumer.topics	/,com/day/cq/replication/job/*
6	job.consumermanager.blacklist	com/adobe/granite/workflow/offloading

Overview – Where to fit in the stack



Workflow Distribution

- New JobConsumer
  - Class: WorkflowOffloadingJobConsumer
  - Topic: com/adobe/granite/workflow/offloading
  - Can launch new workflows
  - Expects the workflow model on the job payload
  - Expects the workflow payload on the job payload
- For use with clustered and non-clustered installations

Workflow Distribution – Job Consumer (Simplified)

```
@Component
@Service
@Properties({
  @Property(name = JobConsumer.PROPERTY_TOPICS, value = WorkflowOffloadingJobConsumer.TOPIC)
})
public class WorkflowOffloadingJobConsumer implements JobConsumer {

  // the topic for use for this job consumer
  public static final String TOPIC = "com/adobe/granite/workflow/offloading";

  // the job payload properties
  public static final String WORKFLOW_OFFLOADING_MODEL = "offloading.workflow.model";
  public static final String WORKFLOW_OFFLOADING_PAYLOAD = "offloading.workflow.payload”;

  public JobResult process(Job job) {
    // read workflow model and payload from job payload
    String modelPath= job.getProperty(WORKFLOW_OFFLOADING_MODEL , "");
    String payloadPath= job.getProperty(WORKFLOW_OFFLOADING_PAYLOAD , "");

    // get/create WorkflowSession, WorkflowModel and WorkflowData objects
    WorkflowSession wfSession = ..;
    WorkflowModel wfModel = ..;
    WorkflowData wfData = ..;

    // start the workflow
    wfSession.startWorkflow(wfModel, wfData, metaData);

    // all good
    return JobResutl.OK;
  }
}
```

Workflow Distribution – Job Consumer (Simplified)

```
@Component
@Service
@Properties({
  @Property(name = JobConsumer.PROPERTY_TOPICS, value = WorkflowOffloadingJobConsumer.TOPIC)
})
public class WorkflowOffloadingJobConsumer implements JobConsumer {

  // the topic for use for this job consumer
  public static final String TOPIC = "com/adobe/granite/workflow/of    ding";

  // the job payload properties
  public static final String WORKFLOW_OFFLOADING_MODEL = "
  public static final String WORKFLOW_OFFLOADING_PAYLOAD =

  public JobResult process(Job job) {
    // read workflow model and payload from job payload
    String modelPath= job.getProperty(WORKFLOW_OFFLOADING_
    String payloadPath= job.getProperty(WORKFLOW_OFFLOADIN

    // get/create WorkflowSession, WorkflowModel and Workf
    WorkflowSession wfSession = ..;
    WorkflowModel wfModel = ..;
    WorkflowData wfData = ..;

    // start the workflow
    wfSession.startWorkflow(wfModel, wfData, metaData);

    // all good
    return JobResutl.OK;
  }
}
```

- Create service component
- Must register with topic
- Implement new JobConsumer interface

Workflow Distribution – Job Consumer (Simplified)

```
@Component
@Service
@Properties({
  @Property(name = JobConsumer.PROPERTY_TOPICS, value = WorkflowOffloadingJobConsumer.TOPIC)
})
public class WorkflowOffloadingJobConsumer implements JobConsumer {

  // the topic for use for this job consumer
  public static final String TOPIC = "com/adobe/granite/workflow/offloading";

  // the job payload properties
  public static final String WORKFLOW_OFFLOADING_MODEL = "offloadi        low.model";
  public static final String WORKFLOW_OFFLOADING_PAYLOAD =

  public JobResult process(Job job) {
    // read workflow model and payload from job payload
    String modelPath= job.getProperty(WORKFLOW_OFFLOADING_MODEL , "");
    String payloadPath= job.getProperty(WORKFLOW_OFFLOADING_PAYLOAD , "");

    // get/create WorkflowSession, WorkflowModel and WorkflowData objects
    WorkflowSession wfSession = ..;
    WorkflowModel wfModel = ..;
    WorkflowData wfData = ..;

    // start the workflow
    wfSession.startWorkflow(wfModel, wfData, metaData);

    // all good
    return JobResutl.OK;
  }
}
```

- Define the job topic

Workflow Distribution – Job Consumer (Simplified)

```
@Component
@Service
@Properties({
  @Property(name = JobConsumer.PROPERTY_TOPICS, value = WorkflowOffloadingJobConsumer.TOPIC)
})
public class WorkflowOffloadingJobConsumer implements JobConsumer {

  // the topic for use for this job consumer
  public static final String TOPIC = "com/adobe/granite/workflow/offloading";

  // the job payload properties
  public static final String WORKFLOW_OFFLOADING_MODEL = "offloading.workflow.model";
  public static final String WORKFLOW_OFFLOADING_PAYLOAD = "offloading.workflow.payload”;

  public JobResult process(Job job) {
    // read workflow model and payload from job payload
    String modelPath= job.getProperty(WORKFLOW_OFFLOADING_MODEL , "");
    String payloadPath= job.getProperty(WORKFLOW_OFFLOADING_PAYLOAD , "");

    // get/create WorkflowSession, WorkflowModel and WorkflowData objects
    WorkflowSession wfSession = ..;
    WorkflowModel wfModel = ..;
    WorkflowData wfData = ..;

    // start the workflow
    wfSession.startWorkflow(wfModel, wfData, metaData);

    // all good
    return JobResutl.OK;
  }
}
```

- Access job properties (payload)
- Read workflow model and payload from job properties

Workflow Distribution – Job Consumer (Simplified)

```
@Component
@Service
@Properties({
  @Property(name = JobConsumer.PROPERTY_TOPICS, value = WorkflowOffloadingJobConsumer.TOPIC)
})
public class WorkflowOffloadingJobConsumer implements JobConsumer {

  // the topic for use for this job consumer
  public static final String TOPIC = "com/adobe/granite/wo

  // the job payload properties
  public static final String WORKFLOW_OFFLOADING_MODEL = "
  public static final String WORKFLOW_OFFLOADING_PAYLOAD =

  public JobResult process(Job job) {
    // read workflow model and payload from job payload
    String modelPath= job.getProperty(WORKFLOW_OFFLOADING_MODEL ,
    String payloadPath= job.getProperty(WORKFLOW_OFFLOADING_PAYLOAD ,     );

    // get/create WorkflowSession, WorkflowModel and WorkflowData objects
    WorkflowSession wfSession = ..;
    WorkflowModel wfModel = ..;
    WorkflowData wfData = ..;

    // start the workflow
    wfSession.startWorkflow(wfModel, wfData, metaData);

    // all good
    return JobResutl.OK;
  }
}
```

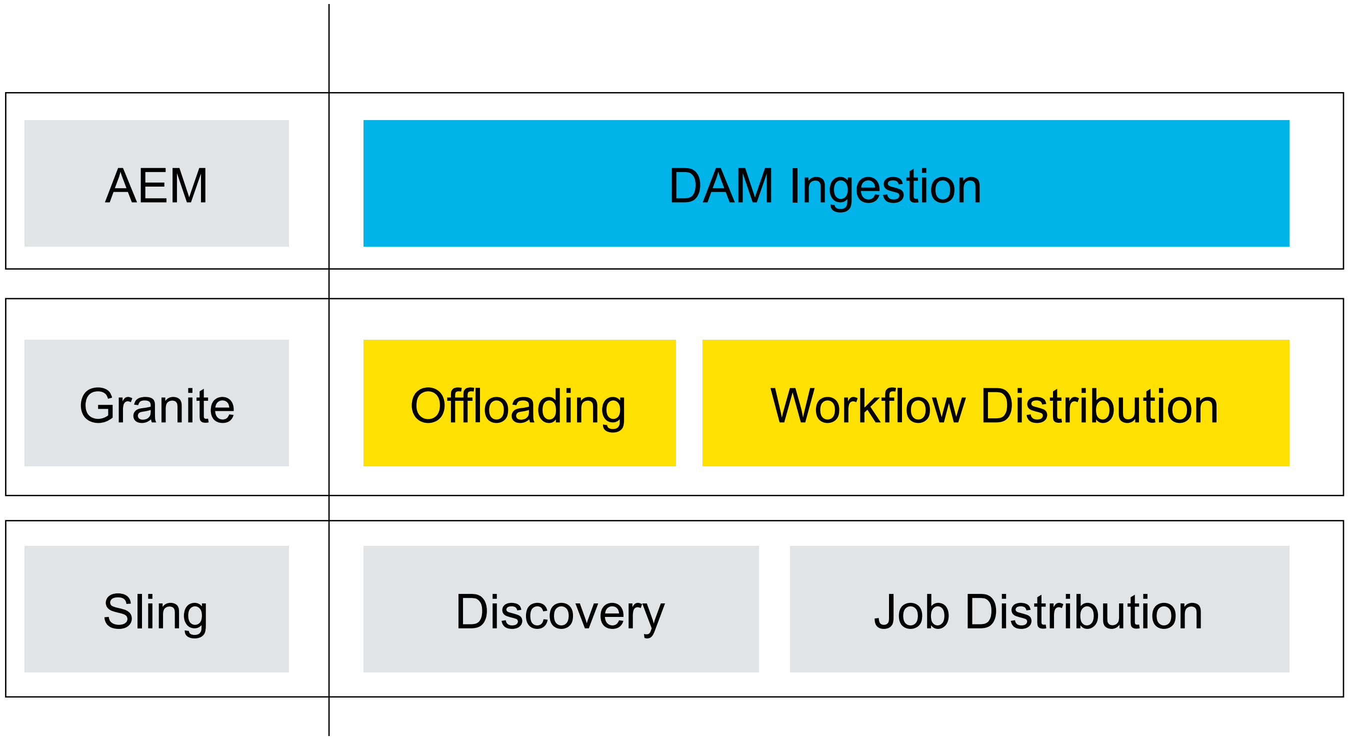
- Workflow specific
- Use workflow API to start workflow for the given model and payload

Workflow Distribution – Job Consumer (Simplified)

```
@Component
@Service
@Properties({
  @Property(name = JobConsumer.PROPERTY_TOPICS, value = WorkflowOffloadingJobConsumer.TOPIC)
})
public class WorkflowOffloadingJobConsumer implements JobConsumer {

  // the topic for use for this job consumer
  public static final String TOPIC = "com/adobe/granite/workflow/offloading";

  // the job payload properties
  public static final String WORKFLOW_OFFLOADING_MODEL = "offloading.workflow.model";
  public static final String WORKFLOW_OFFLOADING_PAYLOAD = "offloading.workflow.payload";

  public JobResult process(Job job) {
    // read workflow model and payload from job payload
    String modelPath= job.getProperty(WORKFLOW_OFFLOADING_
    String payloadPath= job.getProperty(WORKFLOW_OFFLOADIN

    // get/create WorkflowSession, WorkflowModel and Workf
    WorkflowSession wfSession = ..;
    WorkflowModel wfModel = ..;
    WorkflowData wfData = ..;

    // start the workflow
    wfSession.startWorkflow(wfModel, wfData, metaData);

    // all good
    return JobResutl.OK;
  }
}
```

- Use JobResult enumeration to report back the job status

Overview – Where to fit in the stack



DAM Ingestion

- Default ingestion workflow: “DAM Update Asset”
  - Load is put on the instance where the workflow is started, usually the author
- New ingestion workflow: “DAM Update Asset Offloading”
  - Needs to be manually enabled by changing the workflow launcher
  - New workflow model with a single step: AssetOffloadingProcess
  - Uses WorkflowExternalProcess API
  - Creates a new job on topic: com/adobe/granite/workflow/offloading
  - Allows distributing the default ingestion workflow
  - Load is put on the instance where the job is distributed to
- Can be used to distribute in clustered and non-clustered installations

DAM Ingestion – Create Job (from workflow step)

```
@Component
@Service
public class AssetOffloadingProcess implements WorkflowExternalProcess {
  @Reference
  private JobManager jobManager;

  private static final String TOPIC = "com/adobe/granite/workflow/offloading";

  public Serializable execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap){
    Asset asset = ..;
    String workflowModel = “/etc/workflow/models/dam/update_asset/jcr:content/model”;
    String workflowPayload = “/content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg”;

    ValueMap jobProperties = new ValueMapDecorator(new HashMap<String, Object>());
    jobProperties.put(WORKFLOW_OFFLOADING_MODEL, workflowModel);
    jobProperties.put(WORKFLOW_OFFLOADING_PAYLOAD, workflowPayload);

    String offloadingInput = “/etc/workflow/models/dam/update_asset/jcr:content/model,
        /content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg” ;
    String offloadingOutput = “/etc/workflow/models/dam/update_asset/jcr:content/model,
        /content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg” ;

    jobProperties.put(OffloadingJobProperties.INPUT_PAYLOAD.propertyName(), offloadingInput);
    jobProperties.put(OffloadingJobProperties.OUTPUT_PAYLOAD.propertyName(), offloadingOutput);

    Job offloadingJob = jobManager.addJob(TOPIC, null, jobProperties);
    return offloadingJob.getId();
  }

  public boolean hasFinished(Serializable externalProcessId, ..){
    // returns null, if job is finished
    Job offloadingJob = jobManager.getJobById((String) externalProcessId);
    return offloadingJob == null;
  }
}
```

DAM Ingestion – Create Job (from workflow step)

```
@Component
@Service
public class AssetOffloadingProcess implements WorkflowExternalProcess {
  @Reference
  private JobManager jobManager;

  private static final String TOPIC = "com/adobe/granite/workflow/offloadi

  public Serializable execute(WorkItem workItem, WorkflowSession workflo        MetaDataMap metaDataMap){
    Asset asset = ..;
    String workflowModel = "/etc/workflow/models/dam/update_asse
    String workflowPayload = "/content/dam/geometrixx-outdoors/a

    ValueMap jobProperties = new ValueMapDecorator(new HashMap<S
    jobProperties.put(WORKFLOW_OFFLOADING_MODEL, workflowModel);
    jobProperties.put(WORKFLOW_OFFLOADING_PAYLOAD, workflowPaylo

    String offloadingInput = "/etc/workflow/models/dam/update_as
        /content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg" ;
    String offloadingOutput = "/etc/workflow/models/dam/update_asset/jcr:content/model,
        /content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg" ;

    jobProperties.put(OffloadingJobProperties.INPUT_PAYLOAD.propertyName(), offloadingInput);
    jobProperties.put(OffloadingJobProperties.OUTPUT_PAYLOAD.propertyName(), offloadingOutput);

    Job offloadingJob = jobManager.addJob(TOPIC, null, jobProperties);
    return offloadingJob.getId();
  }

  public boolean hasFinished(Serializable externalProcessId, ..){
    // returns null, if job is finished
    Job offloadingJob = jobManager.getJobById((String) externalProcessId);
    return offloadingJob == null;
  }
}
```

- Create service component
- Implement WorkflowExternalProcess interface
- Reference JobManager service

DAM Ingestion – Create Job (from workflow step)

```
@Component
@Service
public class AssetOffloadingProcess implements WorkflowExternalProcess {
  @Reference
  private JobManager jobManager;

  private static final String TOPIC = "com/adobe/granite/workflow/offloading";

  public Serializable execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap){
    Asset asset = ..;
    String workflowModel = “/etc/workflow/models/dam/update_asset/jcr:content/model”;
    String workflowPayload = “/content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg”;

    ValueMap jobProperties = new ValueMapDecorator(new HashMap<String, Obj   >());
    jobProperties.put(WORKFLOW_OFFLOADING_MODEL, workflowModel);
    jobProperties.put(WORKFLOW_OFFLOADING_PAYLOAD, workflowPayload);

    String offloadingInput = “/etc/workflow/models/dam/update_as
        /content/dam/geometrixx-outdoors/articles/downhill-ski-c
    String offloadingOutput = “/etc/workflow/models/dam/update_a
        /content/dam/geometrixx-outdoors/articles/downhill-ski-c

    jobProperties.put(OffloadingJobProperties.INPUT_PAYLOAD.prop
    jobProperties.put(OffloadingJobProperties.OUTPUT_PAYLOAD.pro

    Job offloadingJob = jobManager.addJob(TOPIC, null, jobProper   );
    return offloadingJob.getId();
  }

  public boolean hasFinished(Serializable externalProcessId, ..){
    // returns null, if job is finished
    Job offloadingJob = jobManager.getJobById((String) externalProcessId);
    return offloadingJob == null;
  }
}
```

- DAM and Workflow specific
- Resolve to Asset
- Read model from meta data
- Read workflow payload from Asset path

Adobe

DAM Ingestion – Create Job (from workflow step)

```
@Component
@Service
public class AssetOffloadingProcess implements WorkflowExternalProcess {
  @Reference
  private JobManager jobManager;

  private static final String TOPIC = "com/adobe/granite/workflow/offloading";

  public Serializable execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap){
    Asset asset = ..;
    String workflowModel = "/etc/workflow/models/dam/update_asset/jcr:content/model";
    String workflowPayload = "/content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg";

    ValueMap jobProperties = new ValueMapDecorator(new HashMap<String, Object>());
    jobProperties.put(WORKFLOW_OFFLOADING_MODEL, workflowModel);
    jobProperties.put(WORKFLOW_OFFLOADING_PAYLOAD, workflowPayload);

    String offloadingInput = "/etc/workflow/models/dam/update_asset/jcr:co    nt/model,
        /content/dam/geometrixx-outdoors/articles/downhill-ski-conditioni       " ;
    String offloadingOutput = "/etc/workflow/models/dam/update_asset/jcr:        /model,
        /content/dam/geometrixx-outdoors/articles/downhill-ski-condition

    jobProperties.put(OffloadingJobProperties.INPUT_PAYLOAD.prop
    jobProperties.put(OffloadingJobProperties.OUTPUT_PAYLOAD.pro

    Job offloadingJob = jobManager.addJob(TOPIC, null, jobProper
    return offloadingJob.getId();
  }

  public boolean hasFinished(Serializable externalProcessId, ..)
    // returns null, if job is finished
    Job offloadingJob = jobManager.getJobById((String) externalProcessId);
    return offloadingJob == null;
  }
}
```

- ValueMap for job properties
- Put model and payload on job properties
- Used by the JobConsumer

DAM Ingestion – Create Job (from workflow step)

```
@Component
@Service
public class AssetOffloadingProcess implements WorkflowExternalProcess {
  @Reference
  private JobManager jobManager;

  private static final String TOPIC = "com/adobe/granite/workflo

  public Serializable execute(WorkItem workItem, WorkflowSessio
    Asset asset = ..;
    String workflowModel = “/etc/workflow/models/dam/update_asse
    String workflowPayload = “/content/dam/geometrixx-outdoors/a

    ValueMap jobProperties = new ValueMapDecorator(new HashMap<S
    jobProperties.put(WORKFLOW_OFFLOADING_MODEL, workflowModel);
    jobProperties.put(WORKFLOW_OFFLOADING_PAYLOAD, workflowPayload);

    String offloadingInput = “/etc/workflow/models/dam/update_asset/jcr:content/model,
        /content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg” ;
    String offloadingOutput = “/etc/workflow/models/dam/update_asset/jcr:content/model,
        /content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg” ;

    jobProperties.put(OffloadingJobProperties.INPUT_PAYLOAD.propertyName(), offloadingInput);
    jobProperties.put(OffloadingJobProperties.OUTPUT_PAYLOAD.propertyName(), offloadingOutput);

    Job offloadingJob = jobManager.addJob(TOPIC, null, jobProperties);
    return offloadingJob.getId();
  }

  public boolean hasFinished(Serializable externalProcessId, ..){
    // returns null, if job is finished
    Job offloadingJob = jobManager.getJobById((String) externalProcessId);
    return offloadingJob == null;
  }
}
```

- Build offloading payload properties
- Comma separated list of paths
- Put them on the job payload as well
- Only used for non-clustered distribution

DAM Ingestion – Create Job (from workflow step)

```
@Component
@Service
public class AssetOffloadingProcess implements WorkflowExternalProcess {
  @Reference
  private JobManager jobManager;

  private static final String TOPIC = "com/adobe/granite/workflow/offloading";

  public Serializable execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap){
    Asset asset = ..;
    String workflowModel = "/etc/workflow/models/dam/update_asset/jcr:content/model";
    String workflowPayload = "/content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg";

    ValueMap jobProperties = new ValueMapDecorator(new HashMap<S
    jobProperties.put(WORKFLOW_OFFLOADING_MODEL, workflowModel);
    jobProperties.put(WORKFLOW_OFFLOADING_PAYLOAD, workflowPaylo

    String offloadingInput = "/etc/workflow/models/dam/update_as
        /content/dam/geometrixx-outdoors/articles/downhill-ski-c
    String offloadingOutput = "/etc/workflow/models/dam/update_a
        /content/dam/geometrixx-outdoors/articles/downhill-ski-c

    jobProperties.put(OffloadingJobProperties.INPUT_PAYLOAD.propertyName()    dingInput);
    jobProperties.put(OffloadingJobProperties.OUTPUT_PAYLOAD.propertyName(),  loadingOutput);

    Job offloadingJob = jobManager.addJob(TOPIC, null, jobProperties);
    return offloadingJob.getId();
  }

  public boolean hasFinished(Serializable externalProcessId, ..){
    // returns null, if job is finished
    Job offloadingJob = jobManager.getJobById((String) externalProcessId);
    return offloadingJob == null;
  }
}
```

- Create job using JobManager service
- Use topic from job consumer
- Put job payload properties
- Return the jobId as the workflow process id (workflow specific)

DAM Ingestion – Create Job (from workflow step)

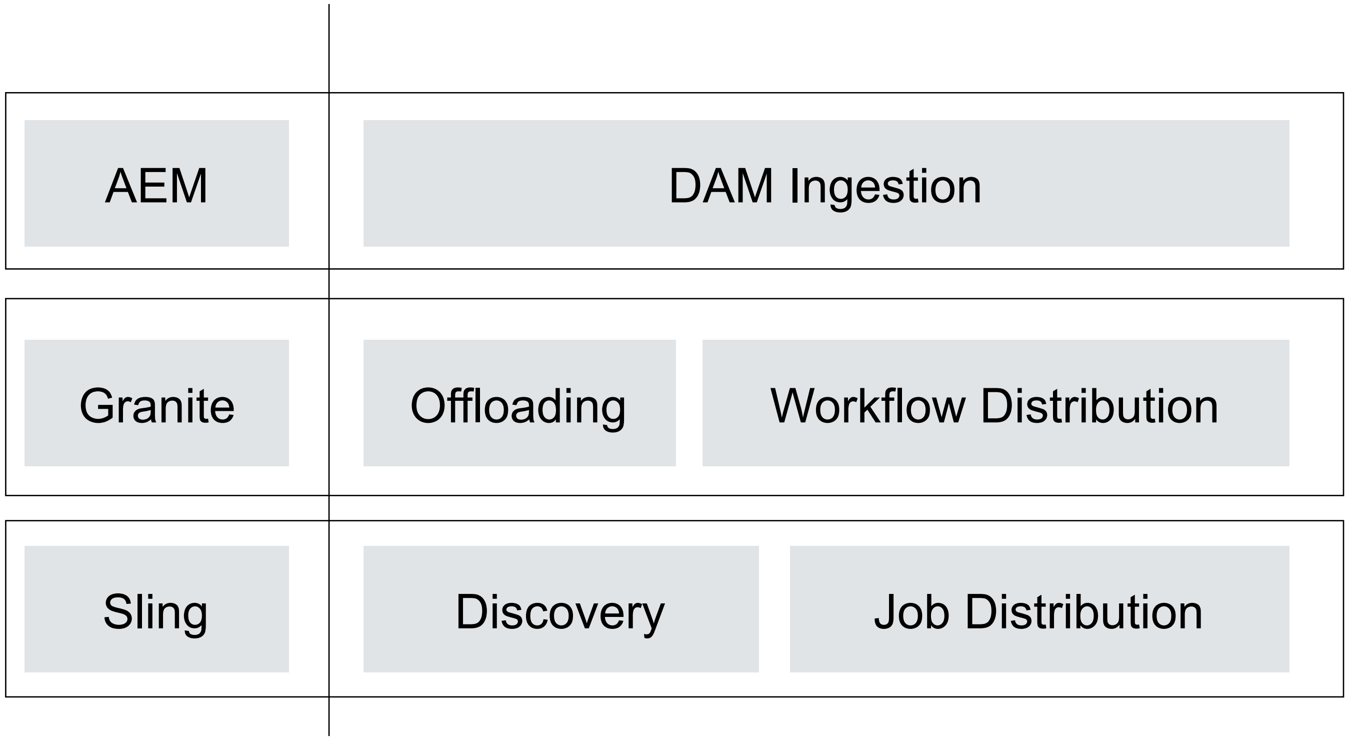
```
@Component
@Service
public class AssetOffloadingProcess implements WorkflowExternalProcess {
  @Reference
  private JobManager jobManager;

  private static final String TOPIC = "com/adobe/granite/workflow/offloading";

  public Serializable execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap){
    Asset asset = ..;
    String workflowModel = "/etc/workflow/models/dam/update_asset/jcr:content/model";
    String workflowPayload = "/content/dam/geometrixx-outdoors/articles/downhill-ski-conditioning.jpg";

    ValueMap jobProperties = new ValueMapDecorator(new HashMap<String, Object>());
    jobProperties.put(WORKFLOW_OFFLOADING_MODEL, workflowModel);
    jobProperties.put(WORKFLOW_OFFLOADING_PAYLOAD, workflowPayload);

    String offloadingInput = "/etc/workflow/models/dam/update_as
        /content/dam/geometrixx-outdoors/articles/downhill-ski-c
    String offloadingOutput = "/etc/workflow/models/dam/update_a
        /content/dam/geometrixx-outdoors/articles/downhill-ski-c

    jobProperties.put(OffloadingJobProperties.INPUT_PAYLOAD.prop
    jobProperties.put(OffloadingJobProperties.OUTPUT_PAYLOAD.pro

    Job offloadingJob = jobManager.addJob(TOPIC, null, jobProperties);
    return offloadingJob.getId();
  }

  public boolean hasFinished(Serializable externalProcessId, ..){
    // returns null, if job is finished
    Job offloadingJob = jobManager.getJobById((String) externalProcessId);
    return offloadingJob == null;
  }
}
```

- Workflow API specific callback
- Process id = jobId, from execute()
- Query job by jobId
- Workflow step finished when job is finished

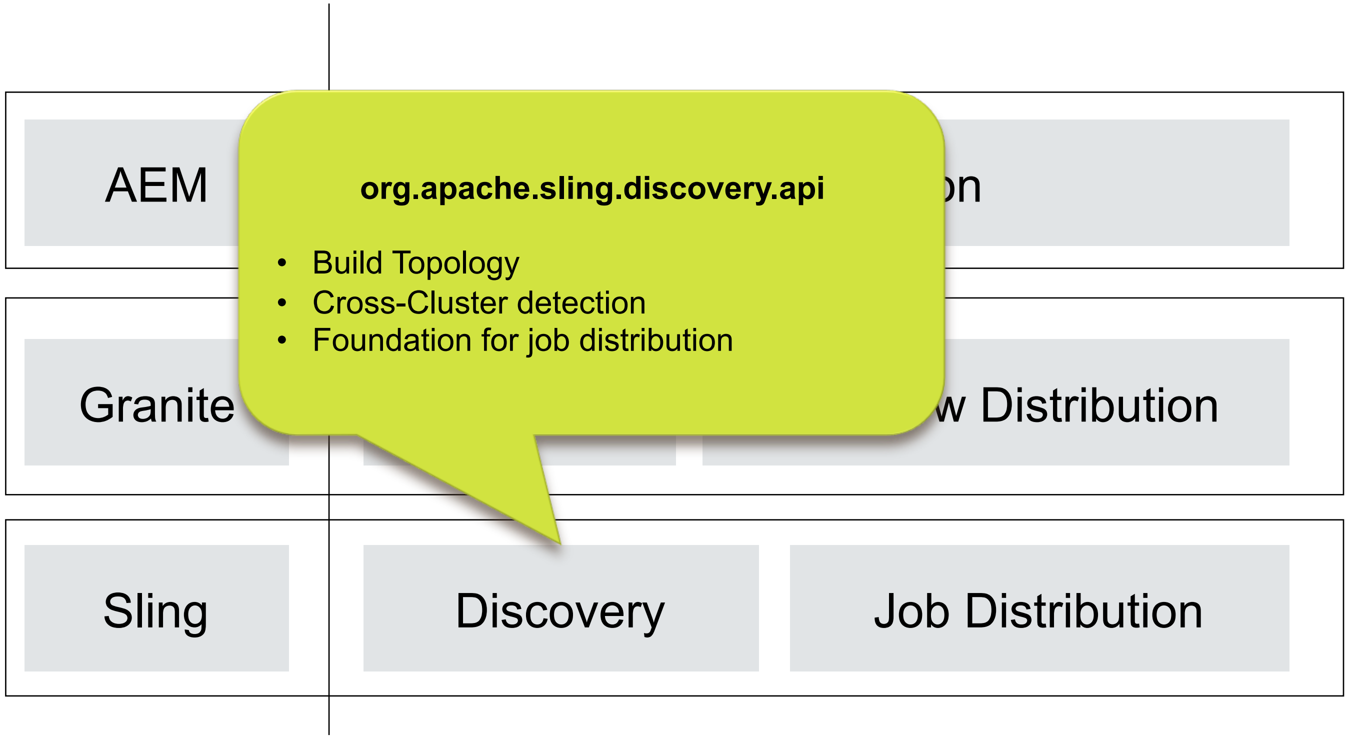
Development - Recipe

1. Choose a job topic
2. Create `JobConsumer` component and register with topic chosen
3. To create a new job use new `JobManager.addJob()` API with the topic chosen and the job payload
4. Add offloading payload to job payload
5. Bundle and deploy `JobConsumer` on topology instances
6. Enable/Disable the new topic on the instances, using Offloading Browser

Take away


AEM
DAM Ingestion
Granite
Offloading
Workflow Distribution
Sling
Discovery
Job Distribution

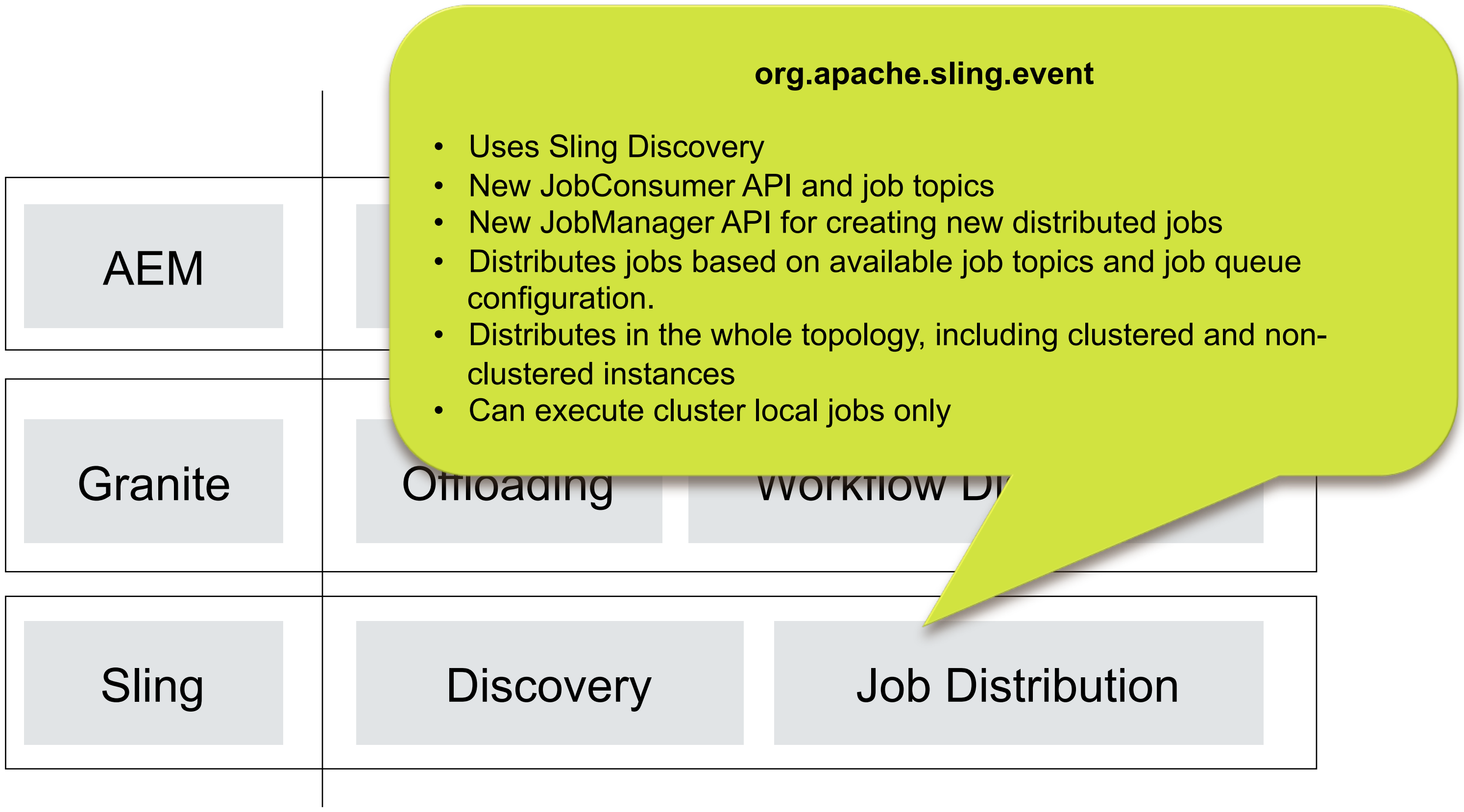
Take away



org.apache.sling.discovery.api

- Build Topology
- Cross-Cluster detection
- Foundation for job distribution

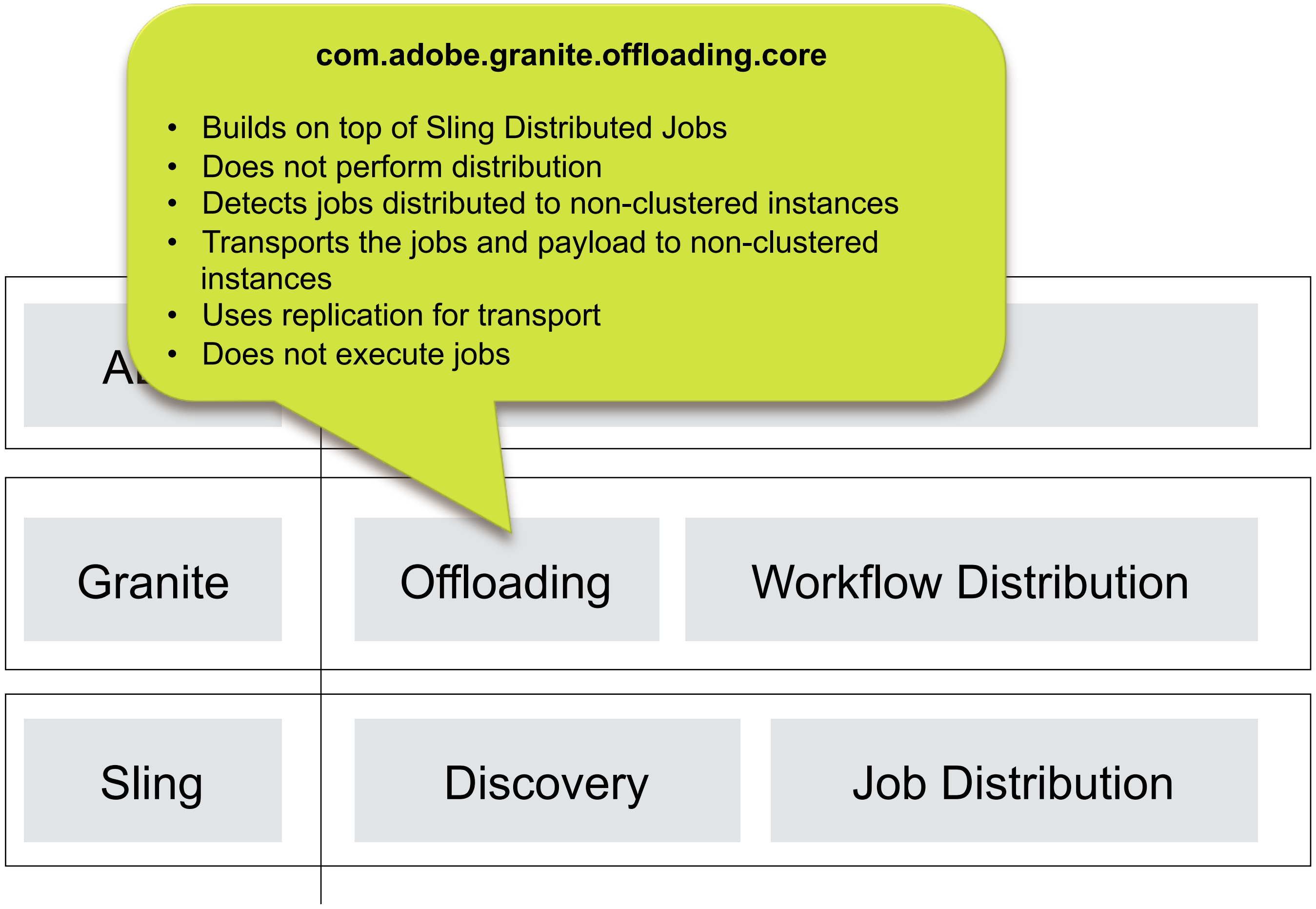
Take away



org.apache.sling.event

- Uses Sling Discovery
- New JobConsumer API and job topics
- New JobManager API for creating new distributed jobs
- Distributes jobs based on available job topics and job queue configuration.
- Distributes in the whole topology, including clustered and non-clustered instances
- Can execute cluster local jobs only

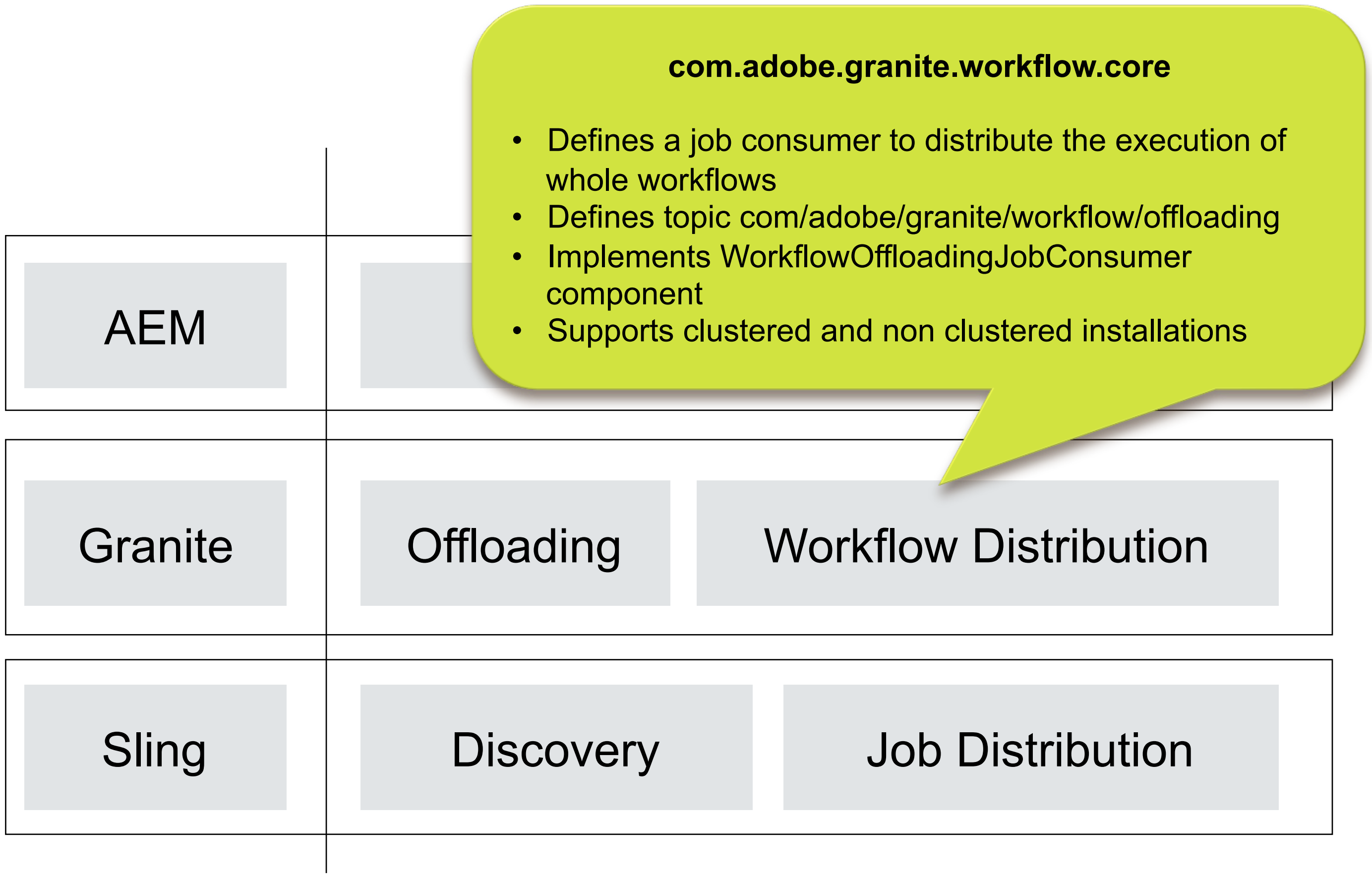
Take away



com.adobe.granite.offloading.core

- Builds on top of Sling Distributed Jobs
- Does not perform distribution
- Detects jobs distributed to non-clustered instances
- Transports the jobs and payload to non-clustered instances
- Uses replication for transport
- Does not execute jobs

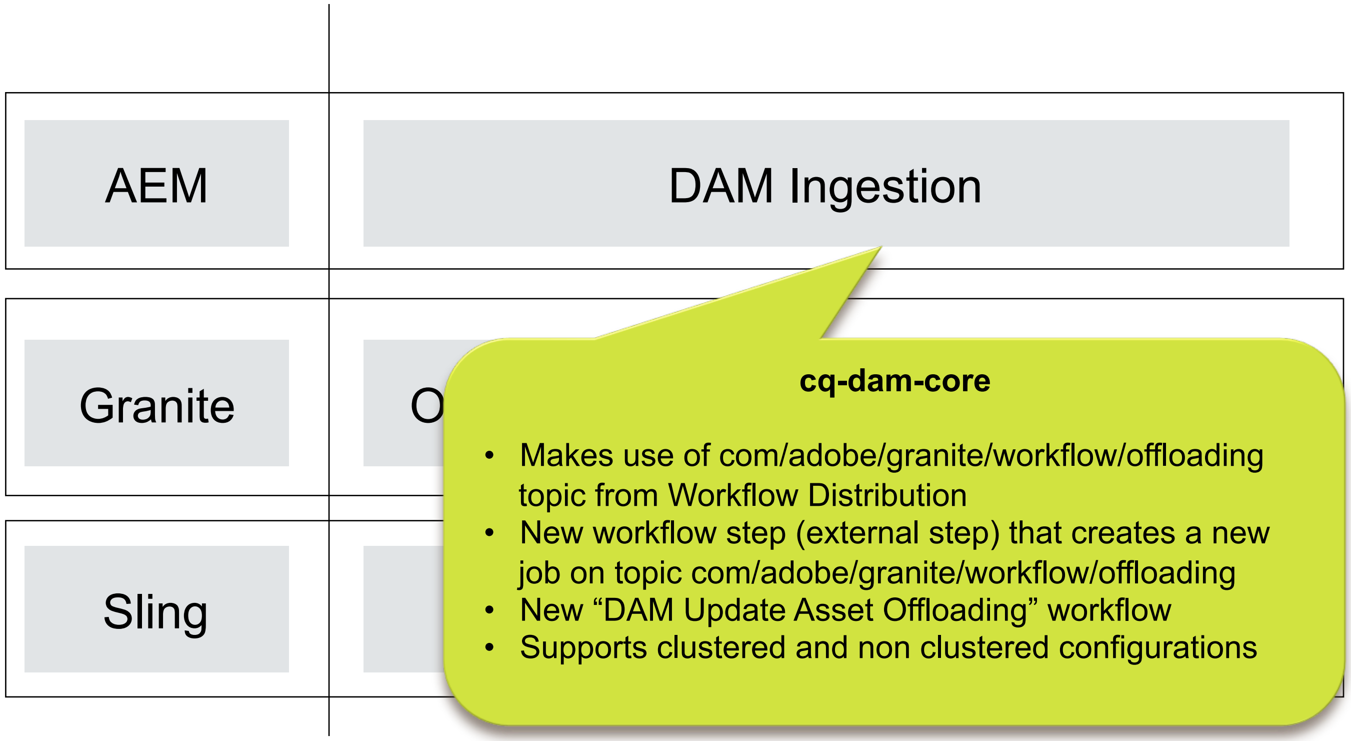
Take away



com.adobe.granite.workflow.core

- Defines a job consumer to distribute the execution of whole workflows
- Defines topic com/adobe/granite/workflow/offloading
- Implements WorkflowOffloadingJobConsumer component
- Supports clustered and non clustered installations

Take away



cq-dam-core

- Makes use of com/adobe/granite/workflow/offloading topic from Workflow Distribution
- New workflow step (external step) that creates a new job on topic com/adobe/granite/workflow/offloading
- New “DAM Update Asset Offloading” workflow
- Supports clustered and non clustered configurations

Adobe®