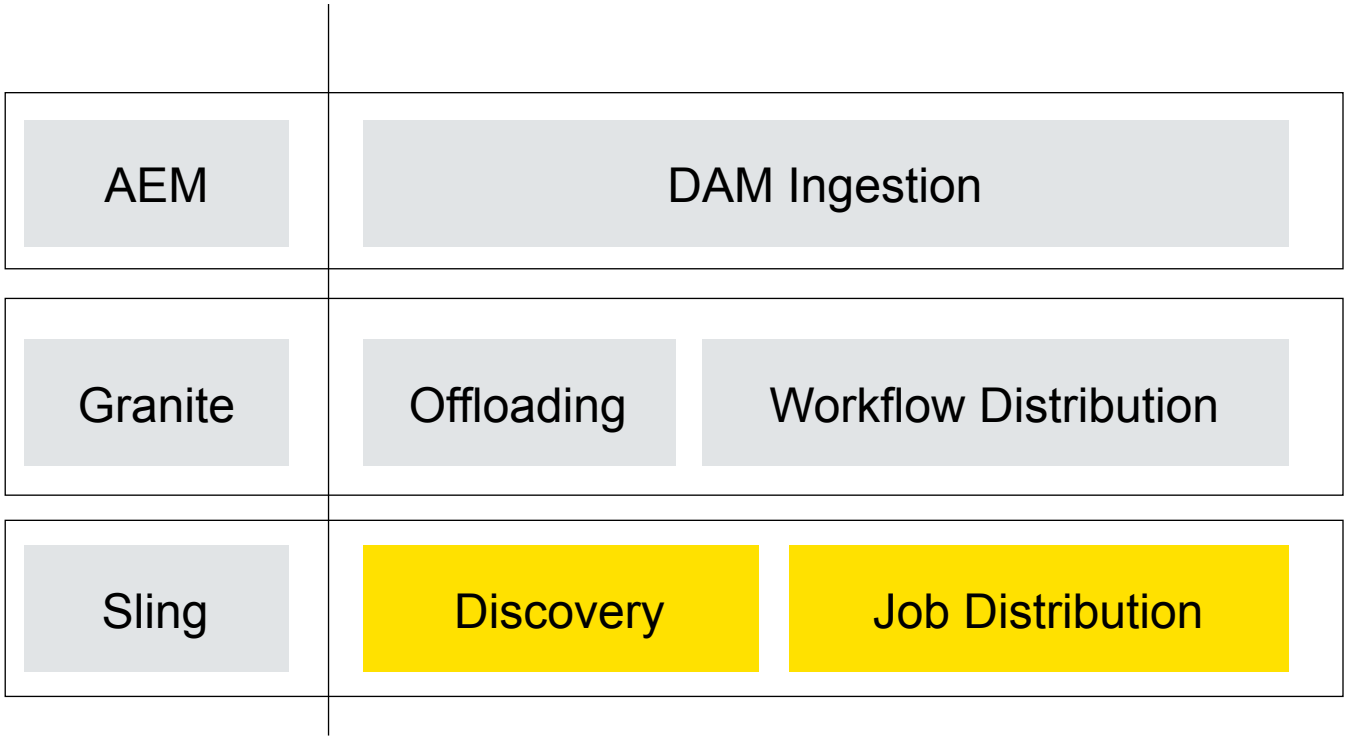
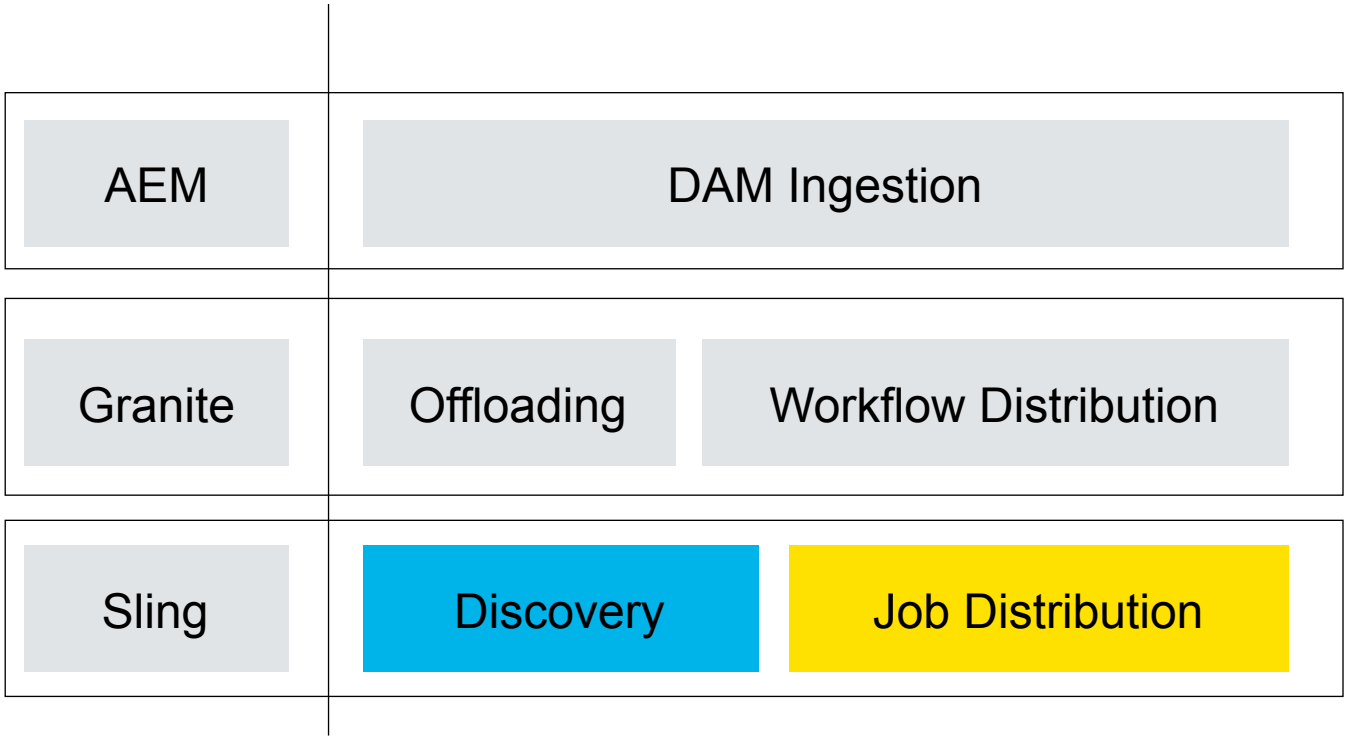
Granite – Discovery and Job Processing

Stefan Egli & Carsten Ziegeler | Basel

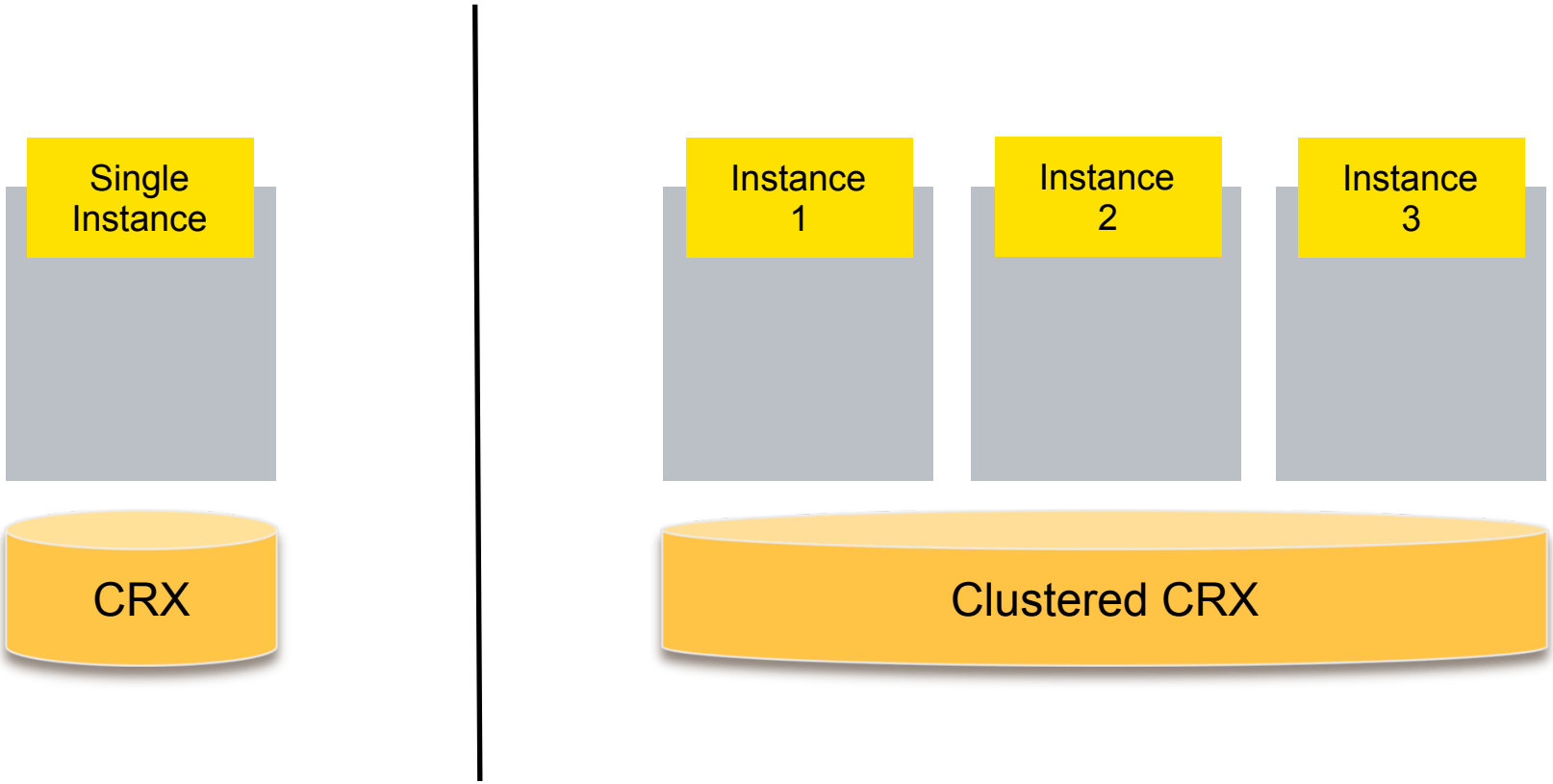
Overview – Where to fit in the stack



Overview – Where to fit in the stack

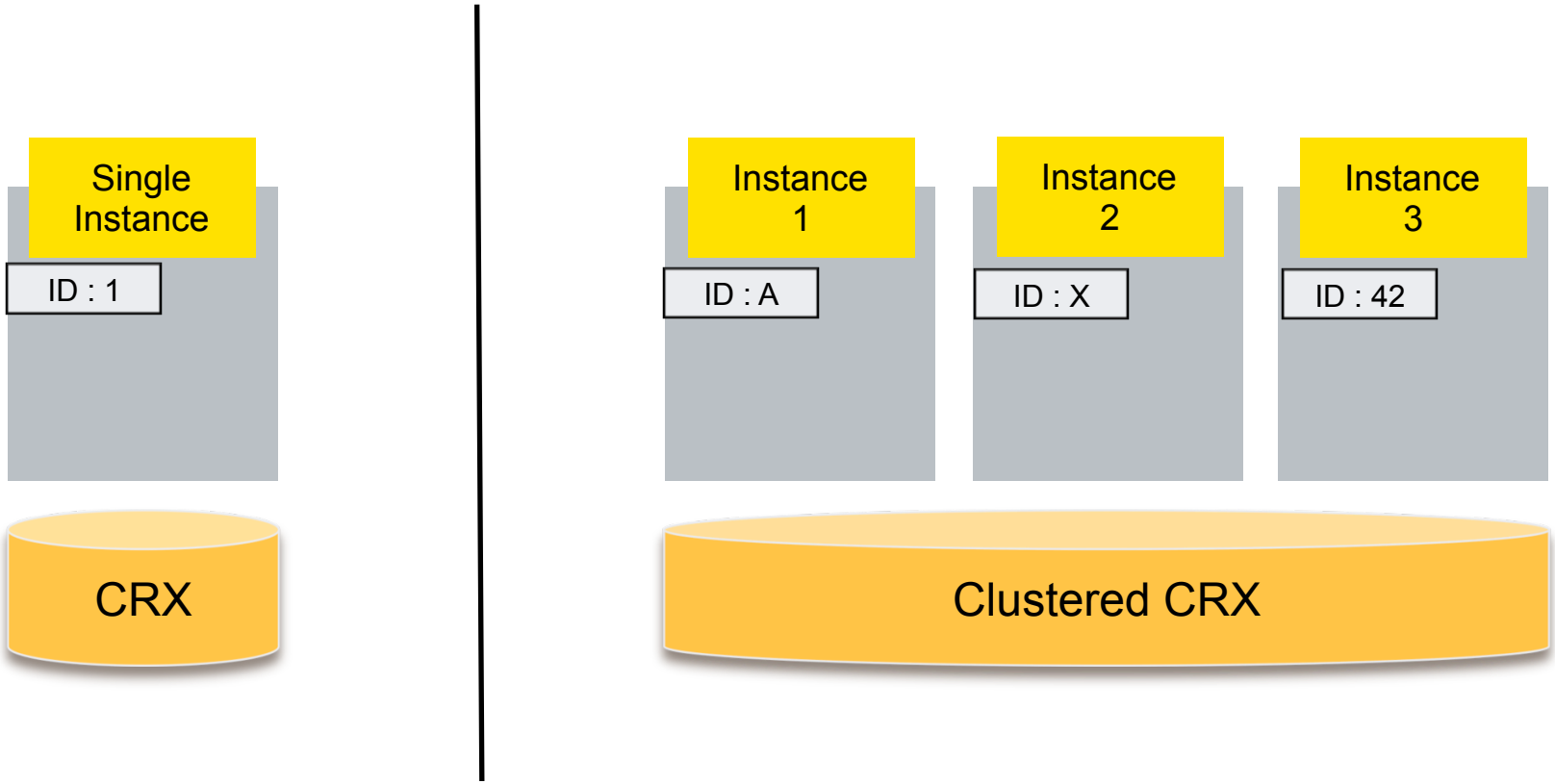


Installation Scenarios



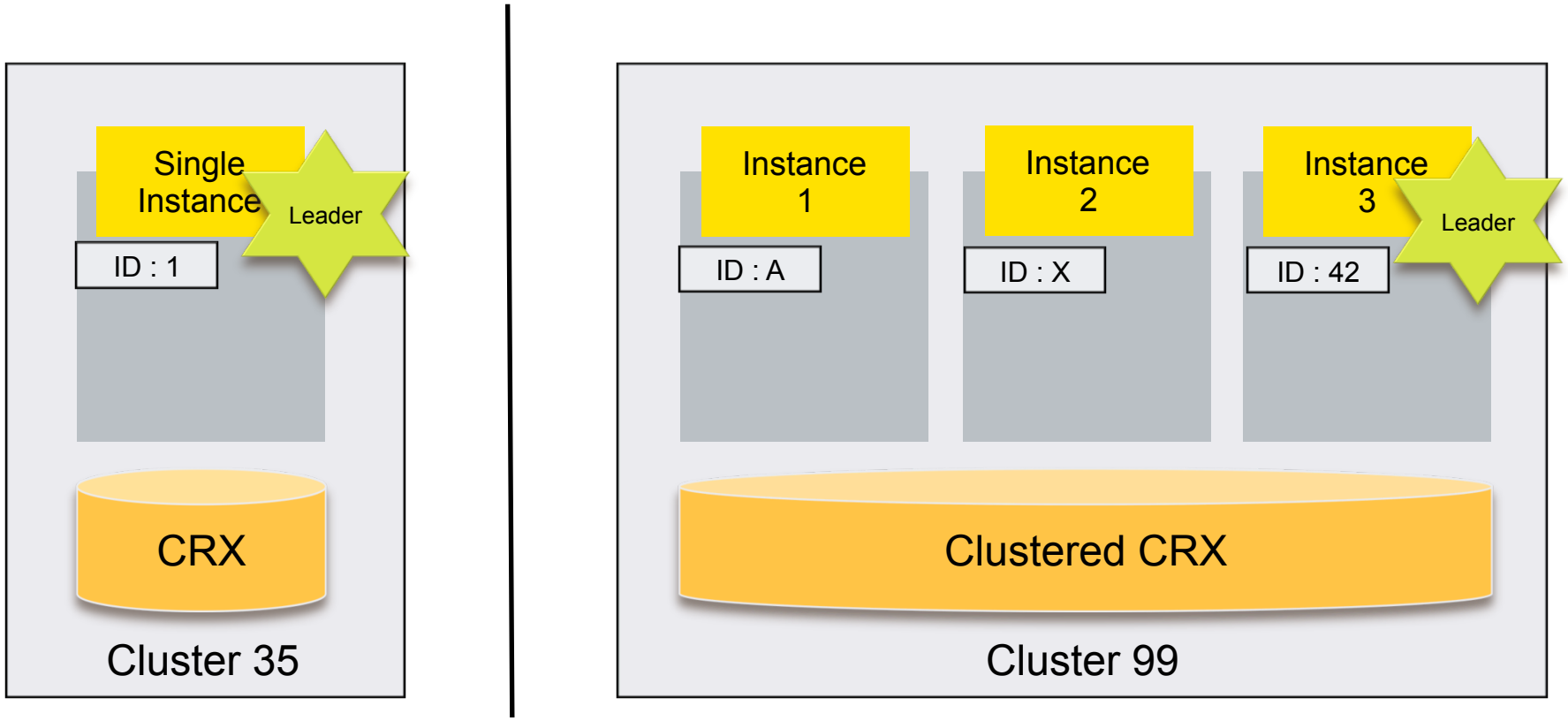
Topologies I – (Apache Sling Discovery)

- Instance: Unique Id (Sling ID)



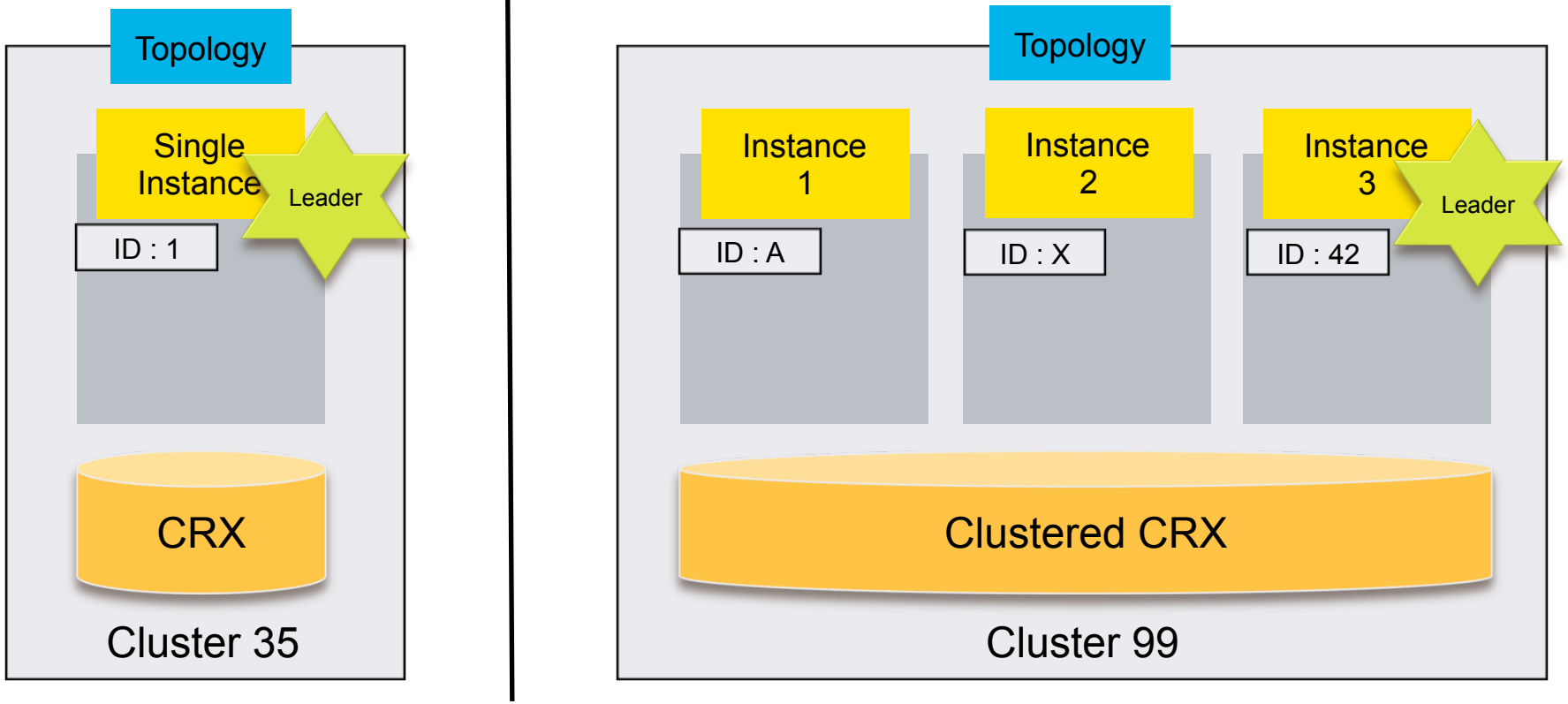
Topologies I – (Apache Sling Discovery)

- Instance: Unique Id (Sling ID)
- Cluster: Unique Id and leader



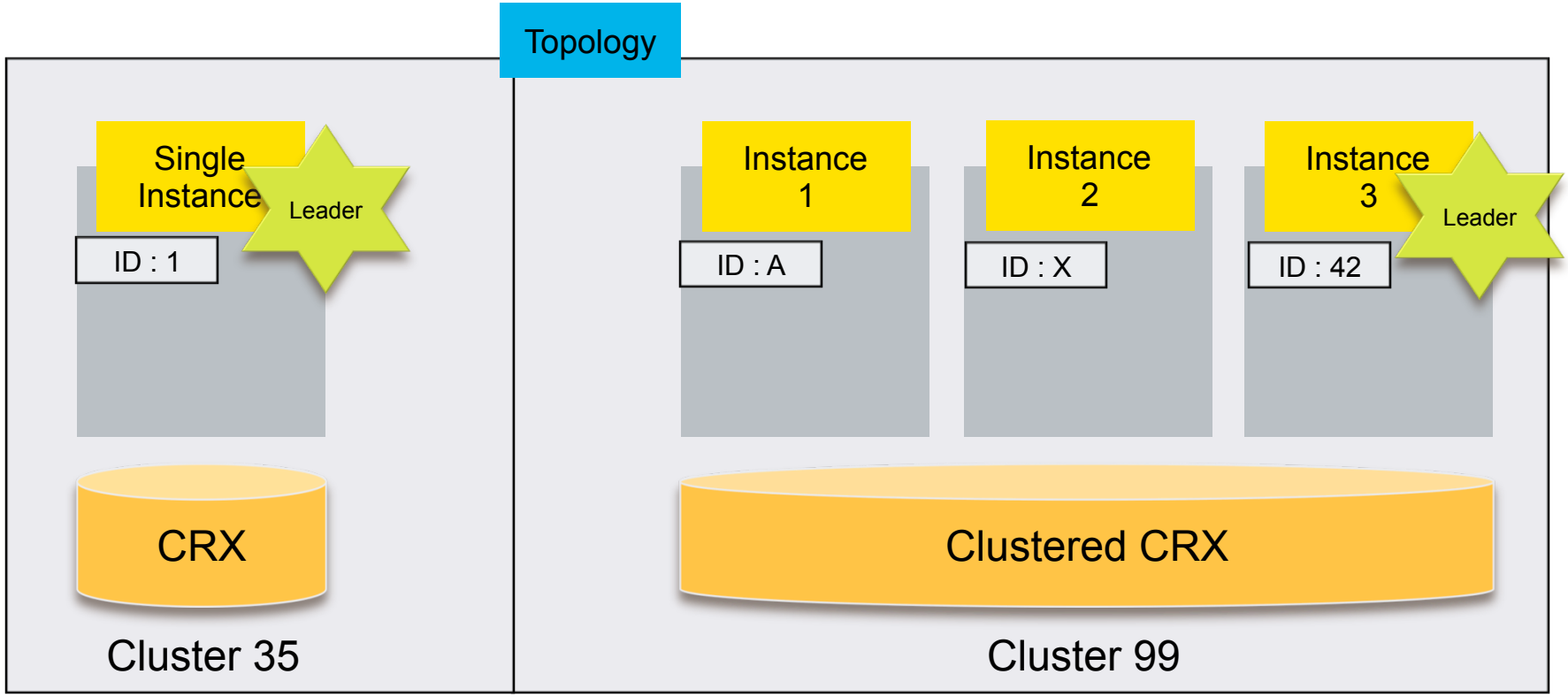
Topologies I – (Apache Sling Discovery)

- Instance: Unique Id (Sling ID)
- Cluster: Unique Id and leader
- Topology: Set of clusters



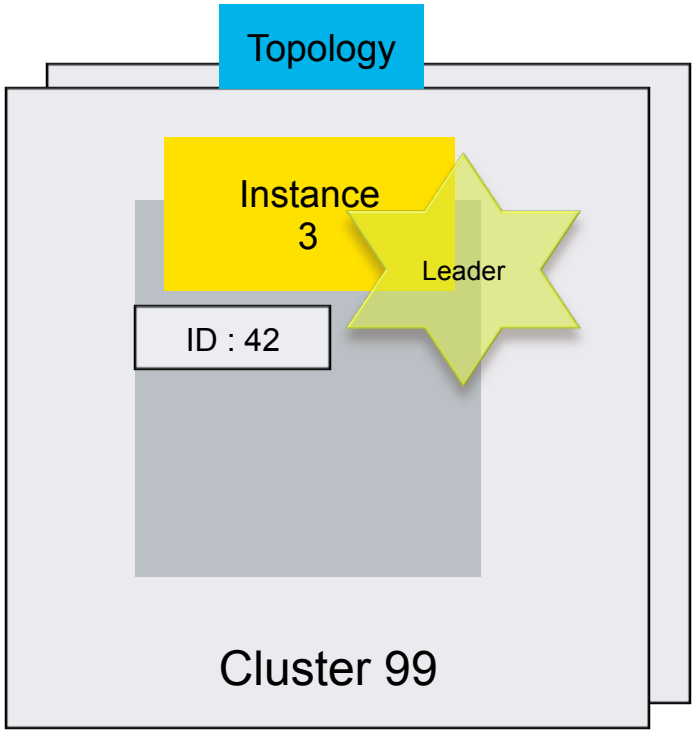
Topologies I – (Apache Sling Discovery)

- Instance: Unique Id (Sling ID)
- Cluster: Unique Id and leader
- Topology: Set of clusters

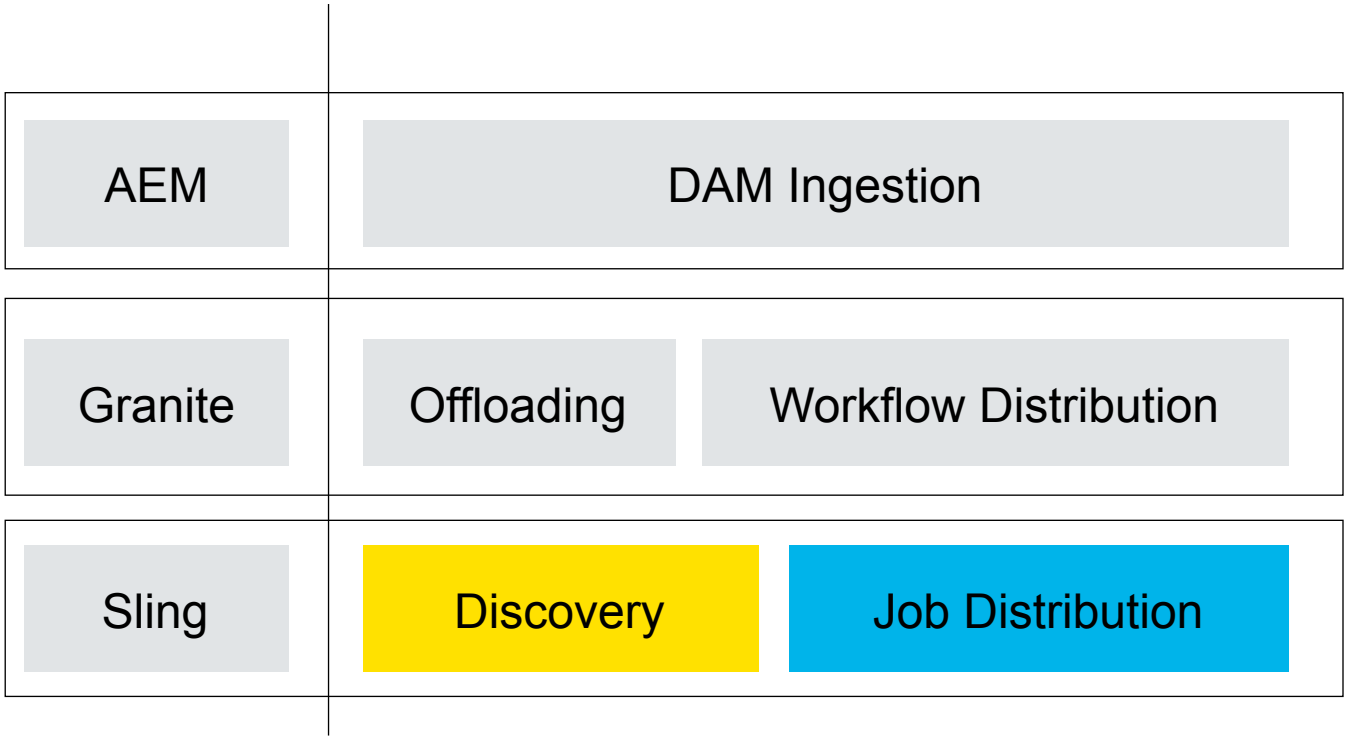


Topologies II – (Apache Sling Discovery)

- Instance
  - Sling ID
  - Optional: Name and description
  - Belongs to a cluster
  - Might be the cluster leader
  - Additional properties which are distributed
    - Extensible through own services (PropertyProvider)
    - E.g. data center, region or *enabled job topics*
- Cluster
  - Elects (stable) leader
  - Stable instance ordering
- TopologyEventListener: receives events on topology changes



Overview – Where to fit in the stack



Job Handling I – (Apache Sling Job Handling)

- Job : Guaranteed processing, exactly once
  - Exactly one job consumer
- Started by client code, e.g. for replication, workflow...
  - Job topic
  - Payload is a serializable map
- Sling Job Manager handles and distributes them
  - Sends out payload events to a consumer…
  - …and waits for response
  - Retry and failover
- Notification listeners (fail, retry, success)

Starting / Processing a job (Apache Sling Job Handling)

Starting a job

```
public interface JobManager {

    Job addJob(String topic, String optionalName, Map<String, Object> properties);
    …
}
```

Processing a job

```
public interface JobConsumer {

    String PROPERTY_TOPICS = "job.topics";

    enum JobResult {
        OK,
        FAILED,
        CANCEL,
        ASYNC
    }

    JobResult process(Job job);
}
```

Note: Starting/processing of jobs through Event Admin is ~~deprecated~~ but still supported

Job Handling II - (Apache Sling Job Handling)

- New jobs are immediately persisted (resource tree / repository)
- Jobs are “pushed” to the processing instance
- Processing instances use different queues
  - Associated with job topic(s)
  - Main queue
  - 0..n custom queues
- For example: replication agent queue or workflow queue

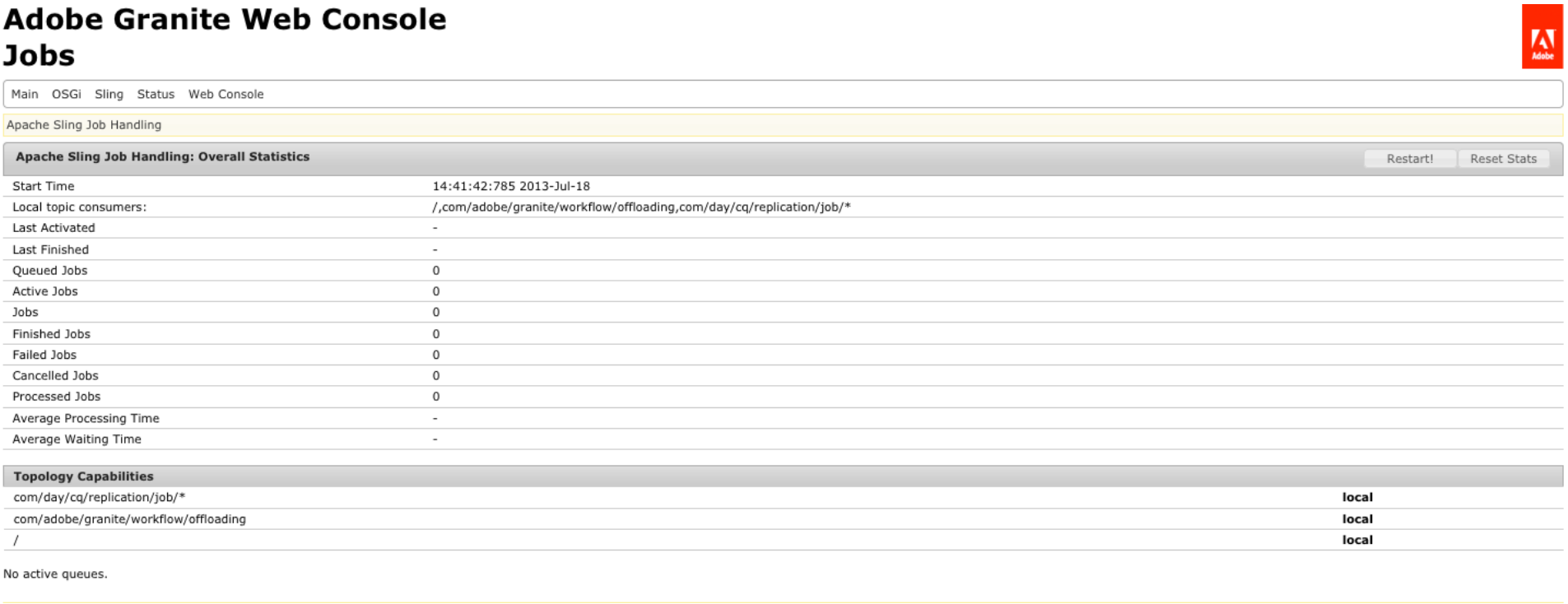
Job Queue – (Apache Sling Job Handling)

- Queue is configurable
- Queue is started on demand in own thread
  - And stopped if unused for some time
- Types
  - Ordered queue (eg replication)
  - Parallel queue: Topic Round Robin (eg workflow)
- Limit for parallel threads per queue
- Number of retries (-1 = endless)
- Retry delay
- Thread priority

Additional Configurations - (Apache Sling Job Handling)

- Job Manager Configuration = Main Queue Configuration
  - Maximum parallel jobs (15)
  - Retries (10)
  - Retry Delay
- Eventing Thread Pool Configuration
  - Used by all queues
  - Pool size (35) = Maximum parallel jobs for a single instance

Monitoring – Web Console - (Apache Sling Job Handling)



Adobe Granite Web Console
Jobs

Main OSGi Sling Status Web Console

Apache Sling Job Handling

Apache Sling Job Handling: Overall Statistics Restart! Reset Stats

Start Time	14:41:42:785 2013-Jul-18
Local topic consumers:	/,com/adobe/granite/workflow/offloading,com/day/cq/replication/job/*
Last Activated	-
Last Finished	-
Queued Jobs	0
Active Jobs	0
Jobs	0
Finished Jobs	0
Failed Jobs	0
Cancelled Jobs	0
Processed Jobs	0
Average Processing Time	-
Average Waiting Time	-

Topology Capabilities

com/day/cq/replication/job/*	**local**
com/adobe/granite/workflow/offloading	**local**
/	**local**

No active queues.

- Now also contained in configuration zip : JSON

Job Distribution - (Apache Sling Job Handling)

- Each instance determines *enabled job topics*
  - Derived from Job Consumers (new API required)
  - Can be whitelist/blacklisted (in Job Consumer Manager)
  - Announced through Topology (used eg by Offloading)
- Job Distribution depends on enabled job topics and queue type
  - Potential set of instances derived from topology (enabled job topics)
  - Ordered : processing on leader only, one job after the other
  - Parallel: Round robin distribution on all potential instances
    - Local cluster instances have preference
- Failover
  - Instance crash: leader redistributes jobs to available instances
    - Leader change taken into account
- On *enabled job topics* changes: potential redistribution


Adobe