Redesigning Apache Flink's Distributed Architecture

Till Rohrmann
trohrmann@apache.org
@stsffap

dataArtisans


ONE DOES NOT SIMPLY
CHANGE ONE'S CLUSTER ENVIRONMENT
imgflip.com

1001 Deployment Scenarios

- Many different deployment scenarios
  - Yarn
  - Mesos
  - Docker/Kubernetes
  - Standalone
  - Etc.

MESOS

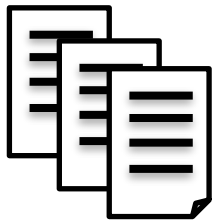
Different Usage Patterns

- Few long running vs. many short running jobs
  - Overhead of starting a Flink cluster
- Job isolation vs. sharing resources
  - Allowing to define per job credentials & secrets
  - Efficient resource utilization by sharing

Job & Session Mode

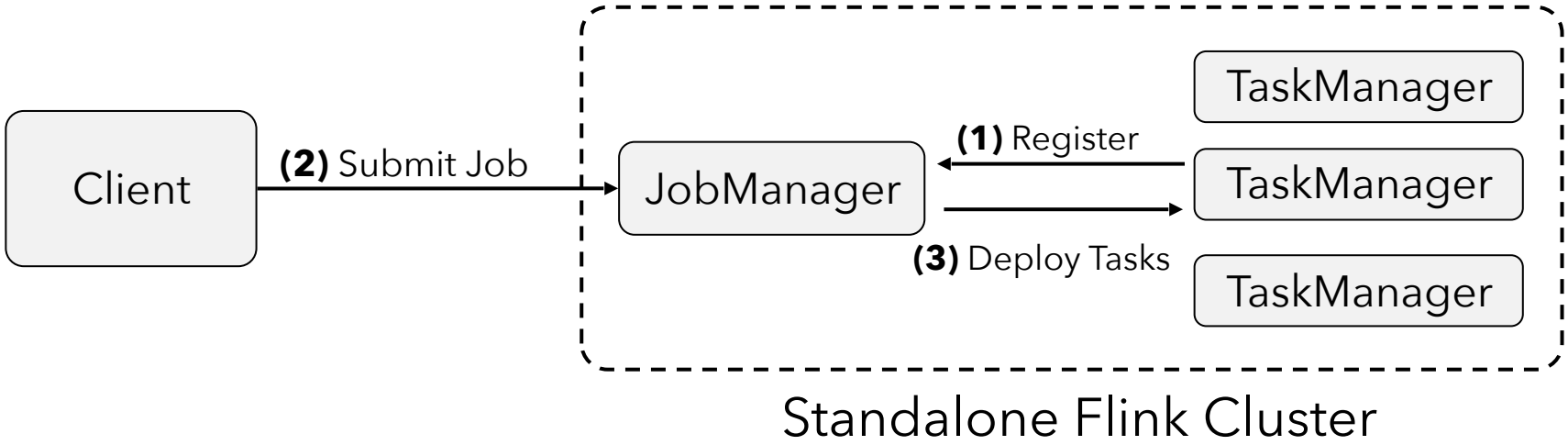
- **Job** mode
  - Dedicated cluster for a single job

- **Session** mode
  - Shared cluster for multiple jobs
  - Resources can be shared across jobs

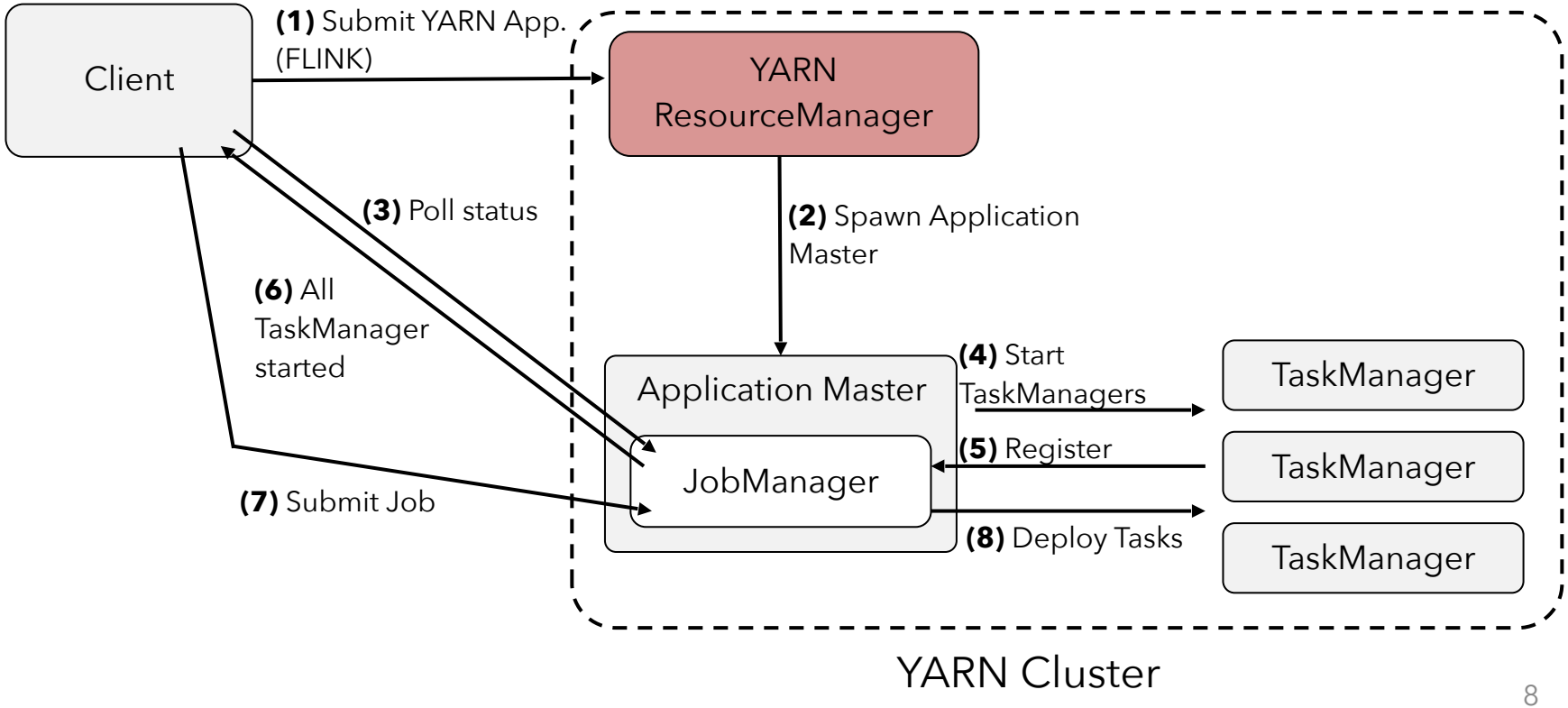


Flink's Current State

As-Is State (Standalone)



As-Is State (YARN)



Problems

- No clear separation of concerns
- No dynamic resource allocation
- No heterogeneous resources
- Not well suited for containerized execution

Flink's New Distributed Architecture

Flink Improvement Proposal 6

- Introduce generic building blocks
- Compose blocks for different scenarios
- Mainly driven by:

Alibaba Group

dataArtisans

Flip-6 design document: https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=65147077

The Building Blocks

ResourceManager

- ClusterManager-specific
- May live across jobs
- Manages available Containers/TaskManagers
- Used to acquire / release resources

Dispatcher

- Lives across jobs
- Touch-point for job submissions
- Spawns JobManagers
- May spawn ResourceManager

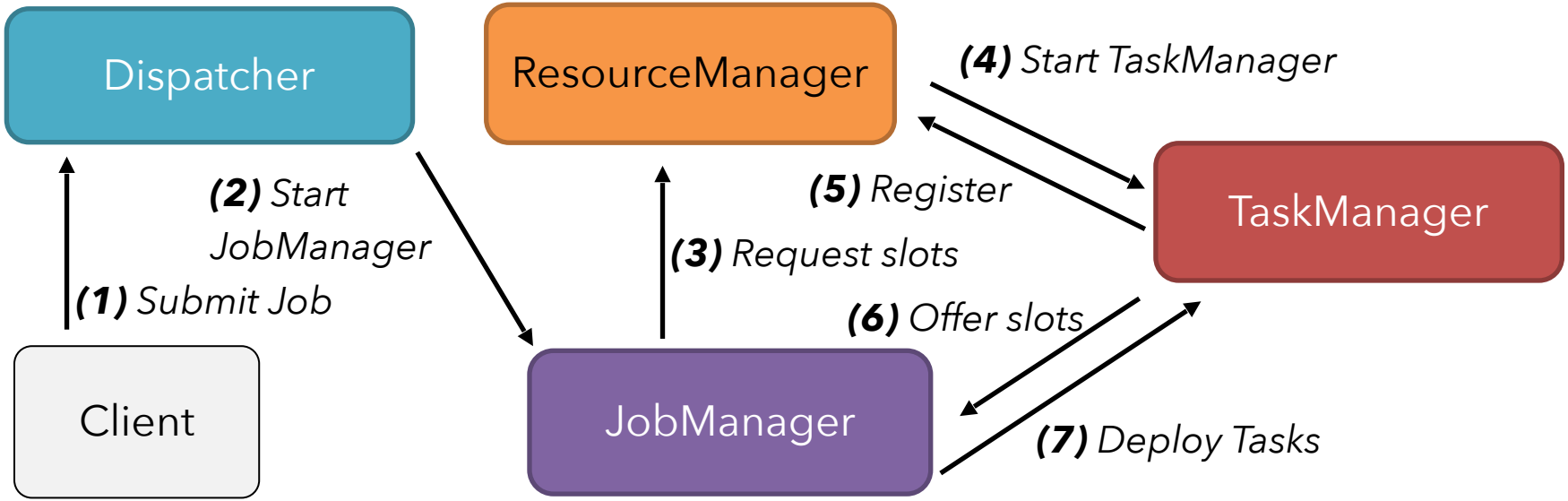
JobManager

- Single job only, started per job
- Thinks in terms of "task slots"
- Deploys and monitors job/task execution

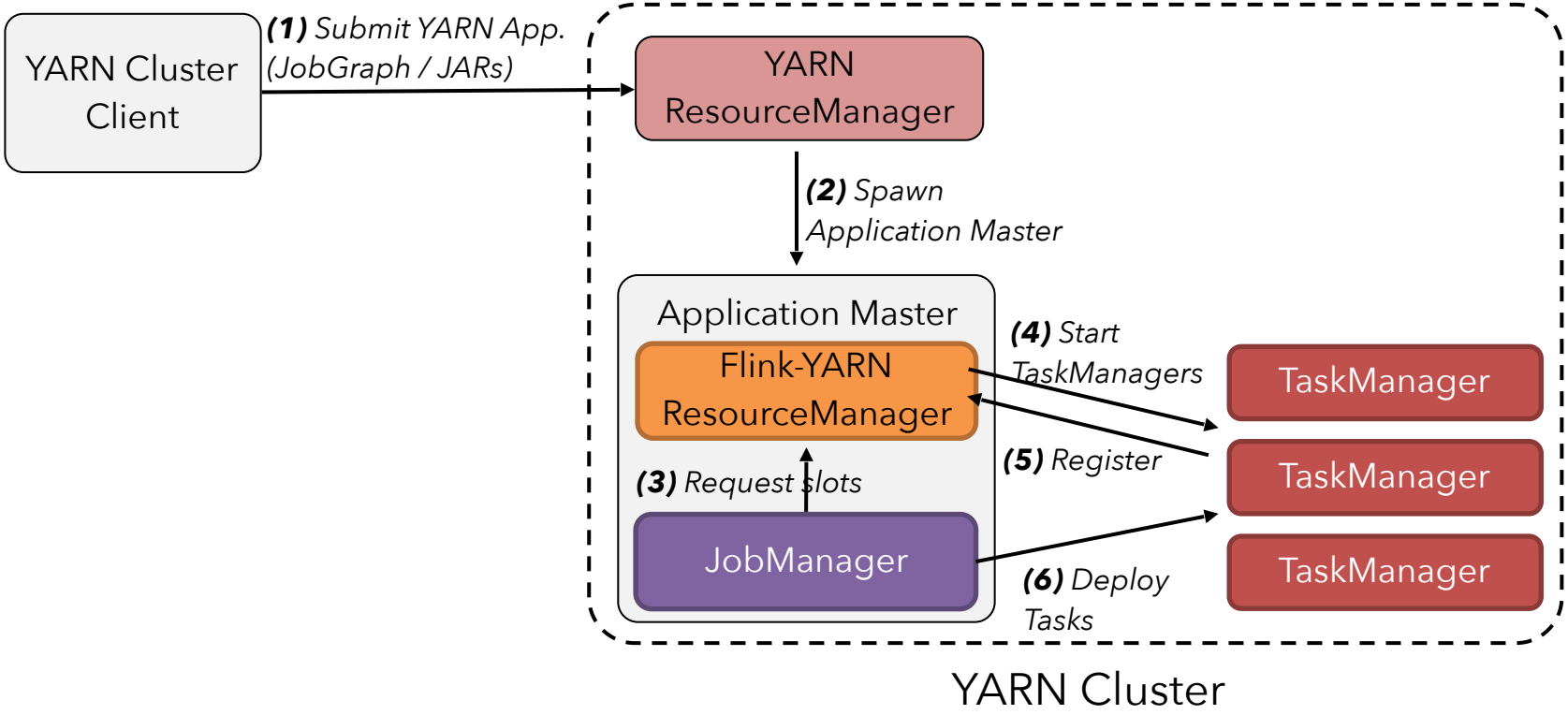
TaskManager

- Registers at ResourceManager
- Gets tasks from one or more JobManagers

The Building Blocks



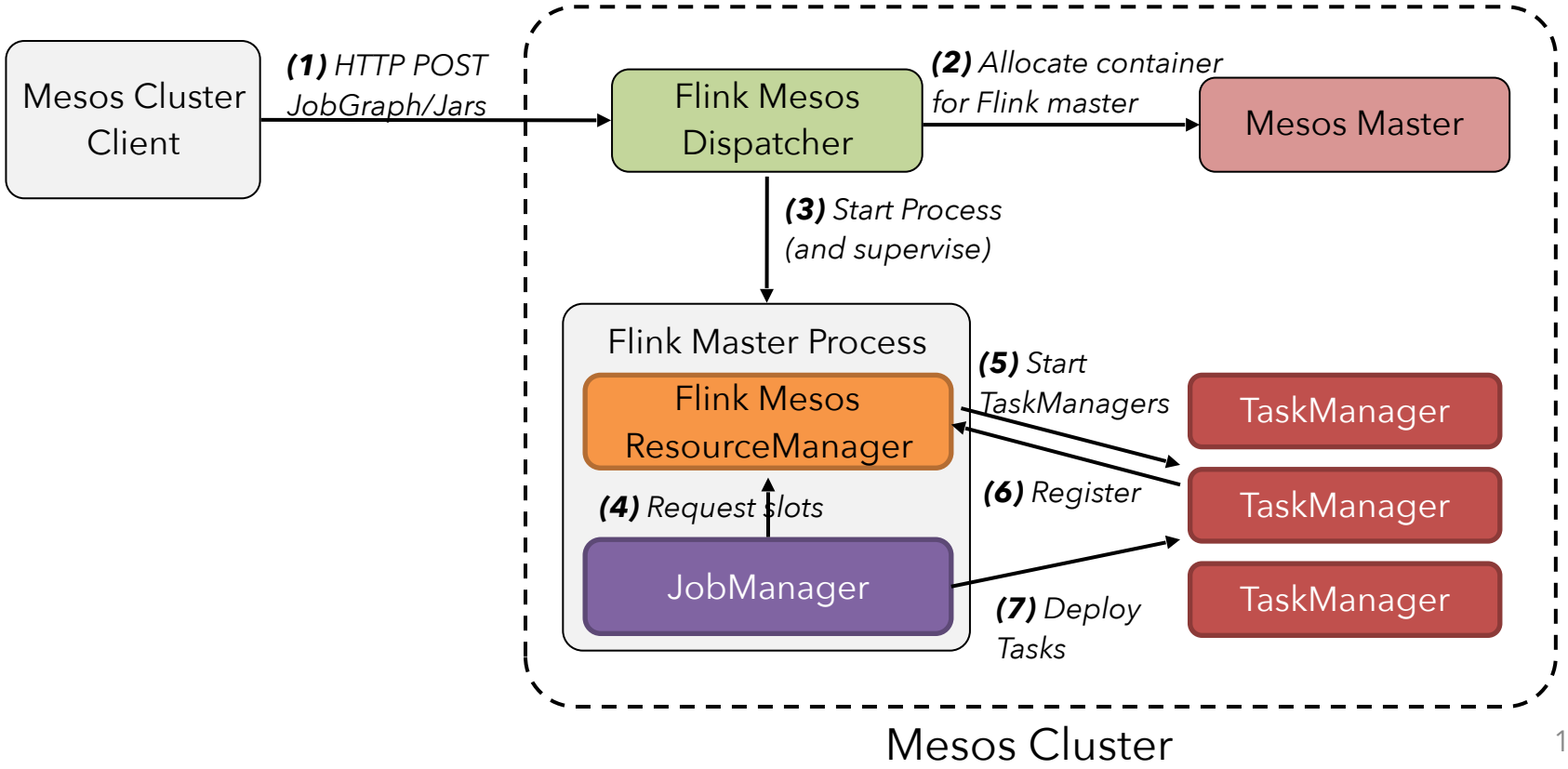
Building Flink-on-YARN



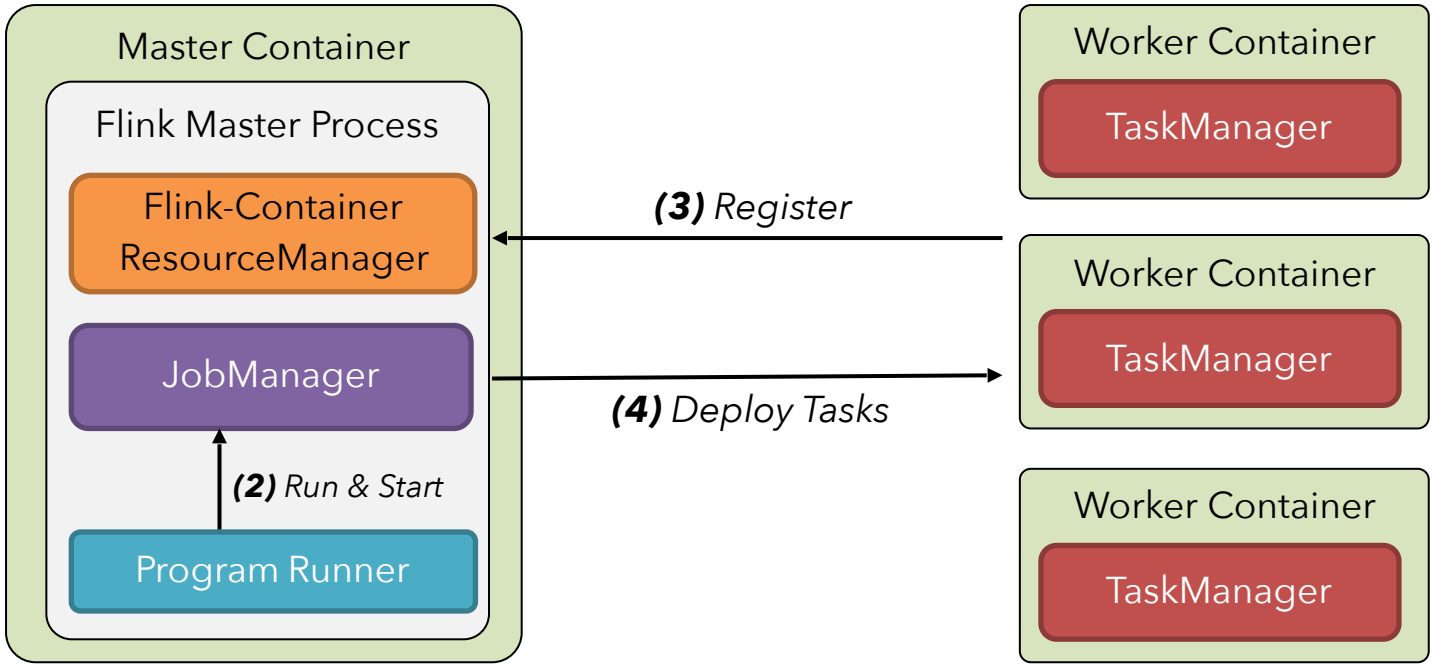
Differences to old YARN mode

- JARs in classpath of all components
- Dynamic resources allocation
- No two phase job submission

Building Flink-on-Mesos



Building Flink-on-Docker/K8S



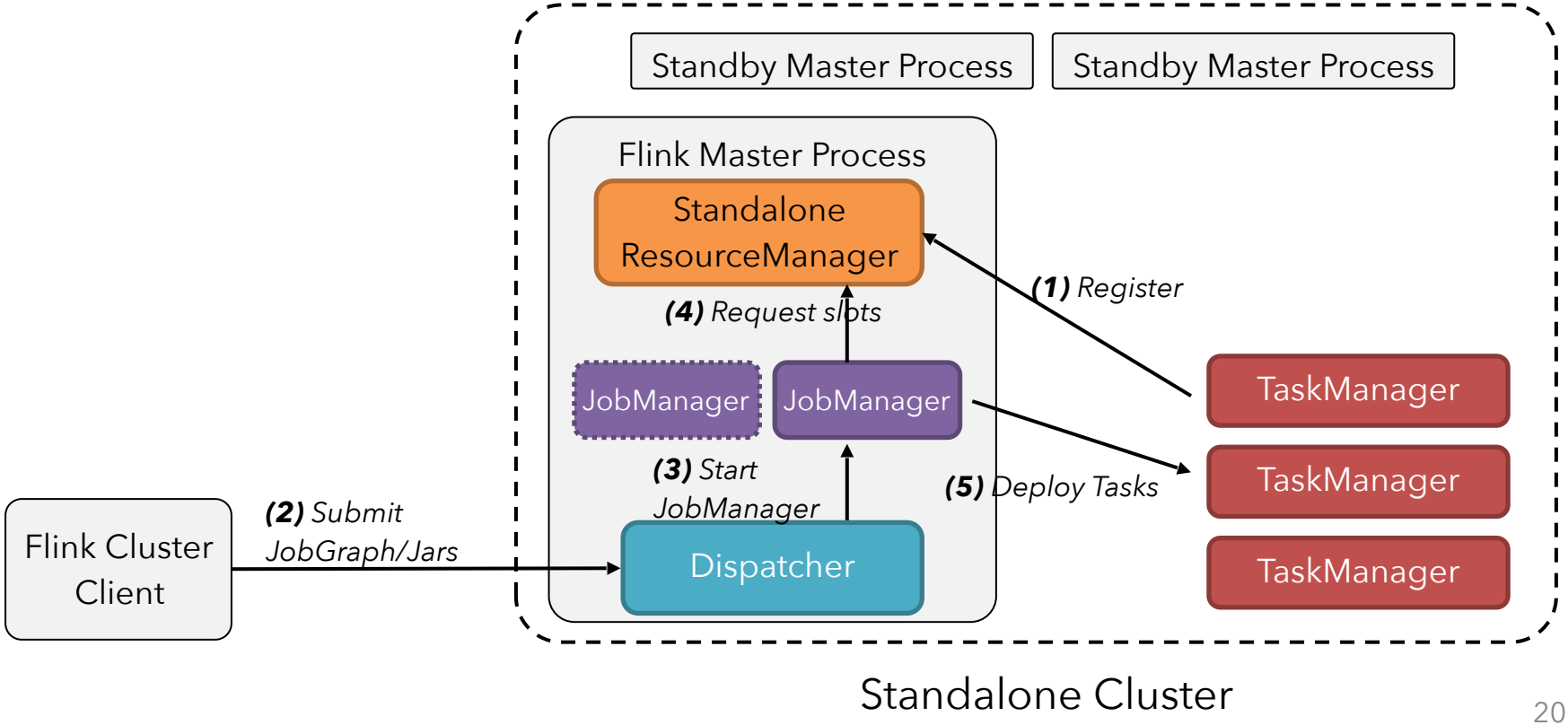
(1) *Container framework starts Master & Worker Containers*

Containerized Execution

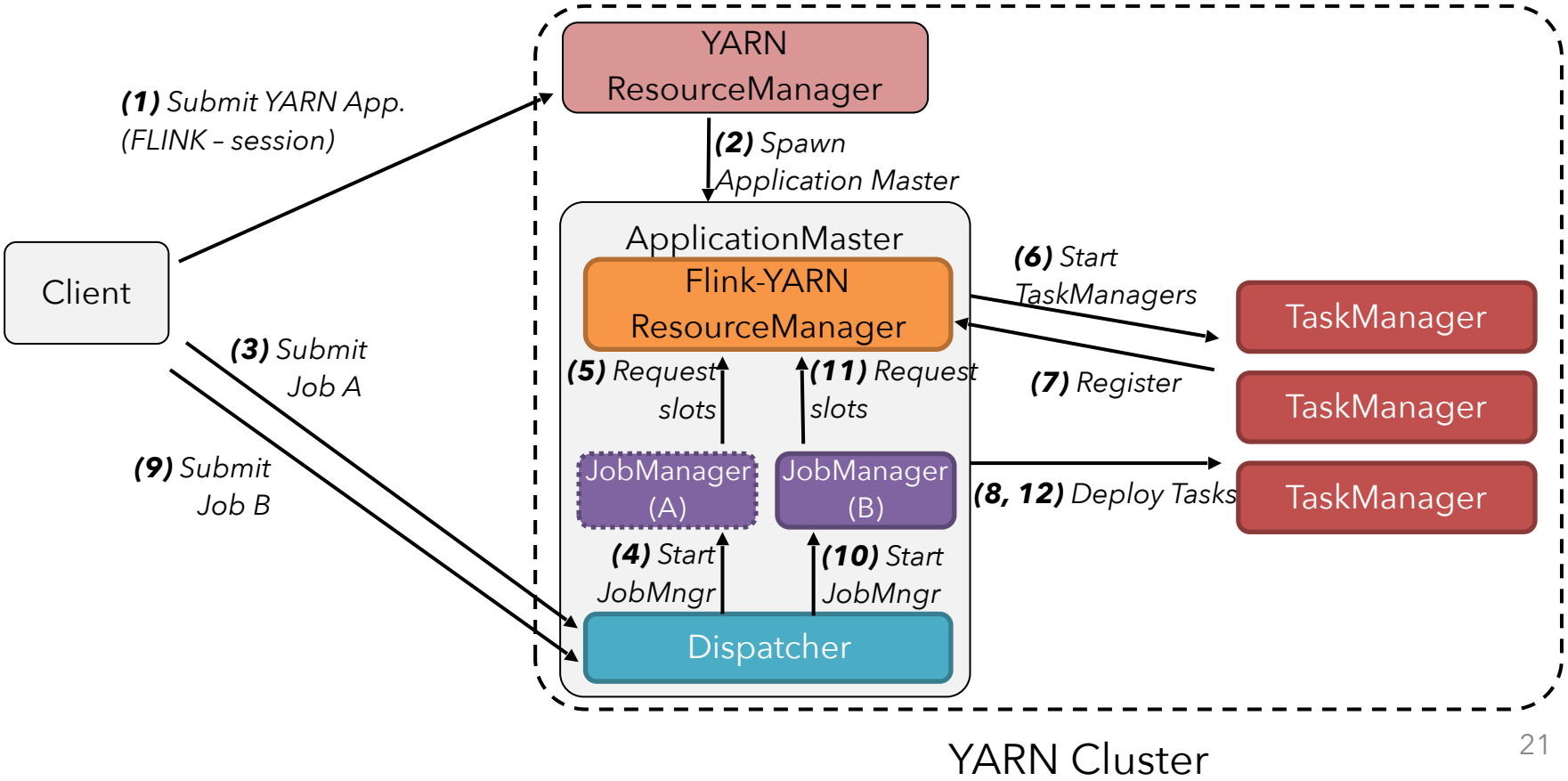
- Single dedicated Resource- and JobManager container and multiple TaskManager containers
- Generalization
  - Start N containers
  - Use leader election to determine JobManager role; remainder TaskManager role
- Enabling auto-scaling groups by rescaling job to fill all available slots

Multi Job Sessions

Building Standalone



YARN Session



Multi Job Sessions

- Dispatcher spawns for each job a dedicated JobManager
- Jobs run under session user credentials
- ResourceManager holds on to resources
  - Reuse of allocated resources
  - Quicker response for successive jobs

Miscellaneous

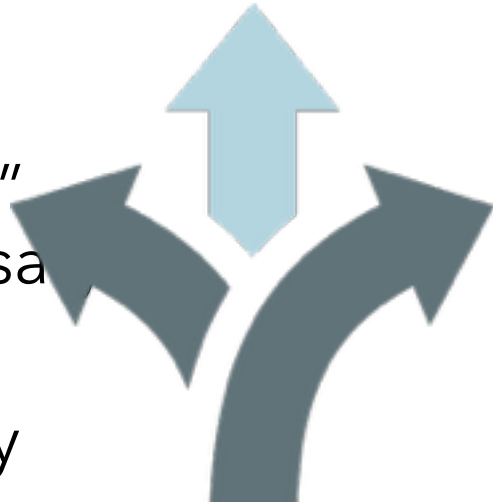
- Resource profiles
  - Specify CPU & memory requirements for individual operators
  - ResourceManager allocates containers according to resource profiles

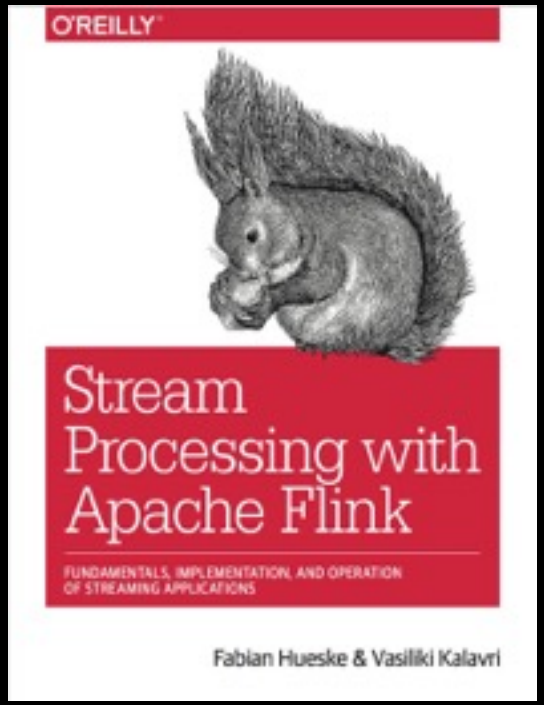
- New RPC abstraction similar to Akka's typed actors
  - Properly defined interface eases development
  - No longer locked in on Akka

Conclusion

Conclusion

- Different cluster environments have different deployment paradigms
- Support for “**Job**” as well as “**Session**” mode in various environments necessa
- Flip-6 architecture provides necessary flexibility to achieve both





Thank you!

@stsffap

@ApacheFlink

@dataArtisans

We are hiring!

data-artisans.com/careers