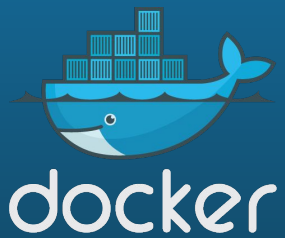
Heart of the SwarmKit: Topology Management

Docker Distributed Systems Summit
10.07.2016

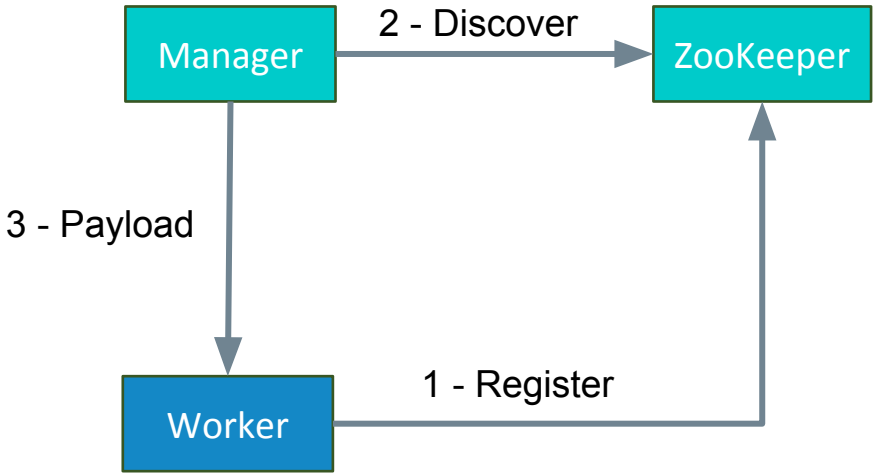
Andrea Luzzardi / al@docker.com / @aluzzardi
Docker Inc.



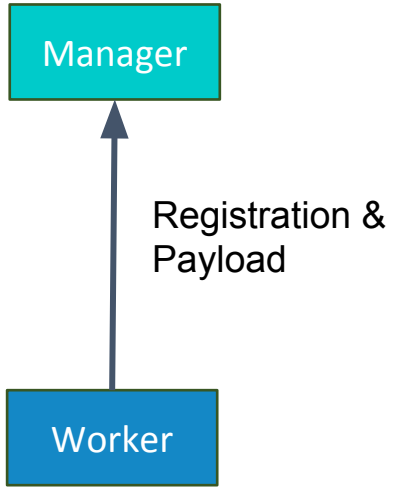
Push vs Pull Model

Push vs Pull

Push



Pull



Push vs Pull

Push

- **Pros**: Provides better control over communication rate
  - Managers decide when to contact Workers
- **Cons**: Requires a discovery mechanism
  - More failure scenarios
  - Harder to troubleshoot

Pull

- **Pros**: Simpler to operate
  - Workers connect to Managers and don't need to bind
  - Can easily traverse networks
  - Easier to secure
  - Less moving parts
- **Cons**: Workers must maintain connection to Managers at all times

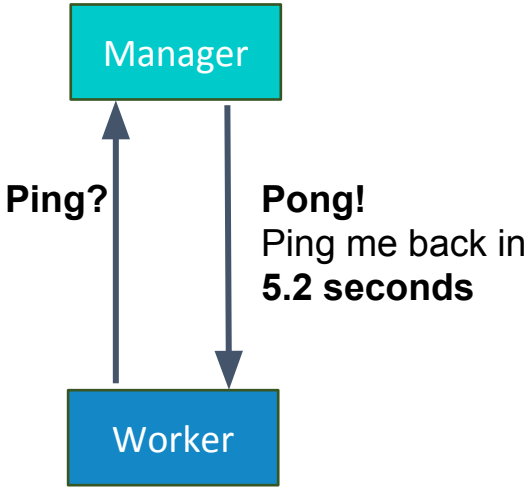
Push vs Pull

- **SwarmKit** adopted the **Pull** model
- Favored **operational simplicity**
- Engineered solutions to provide **rate control in pull mode**

Rate Control

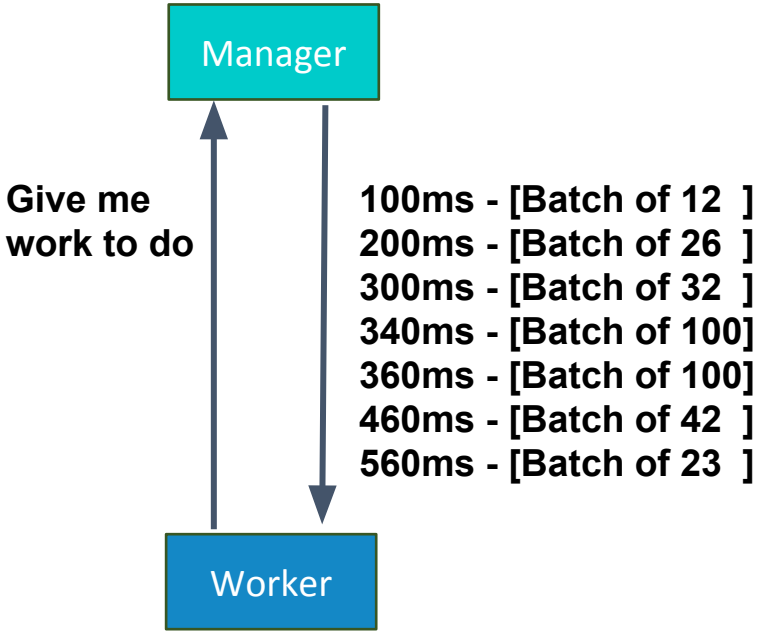
Controlling communication rate in a Pull model

Rate Control: Heartbeats



- Manager dictates heartbeat rate to Workers
- Rate is Configurable
- Managers agree on same Rate by Consensus (Raft)
- Managers add jitter so pings are spread over time (avoid bursts)

Rate Control: Workloads

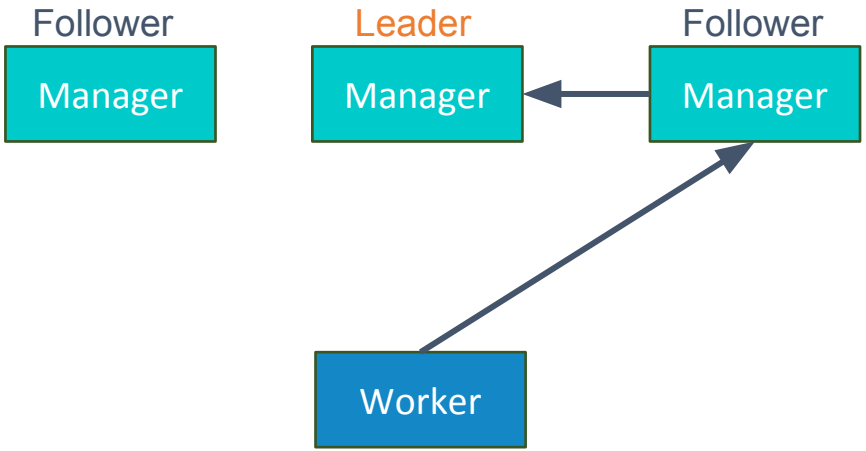


- Worker opens a **gRPC stream** to receive workloads
- Manager can send data whenever it wants to
- Manager will **send data in batches**
- **Changes are buffered** and sent in batches of 100 or every 100 ms, whichever occurs first
- **Adds little delay** (at most 100ms) but drastically **reduces amount of communication**

Replication

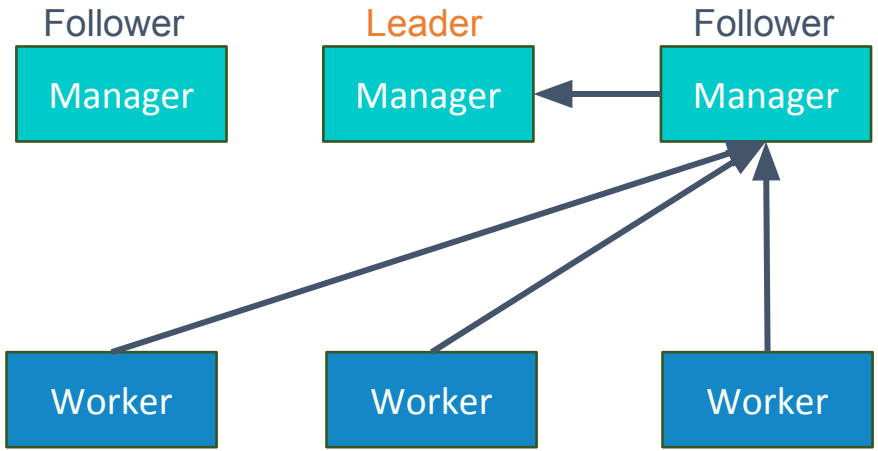
Running multiple managers for high availability

Replication



- **Worker** can connect to any **Manager**
- **Followers** will forward traffic to the **Leader**

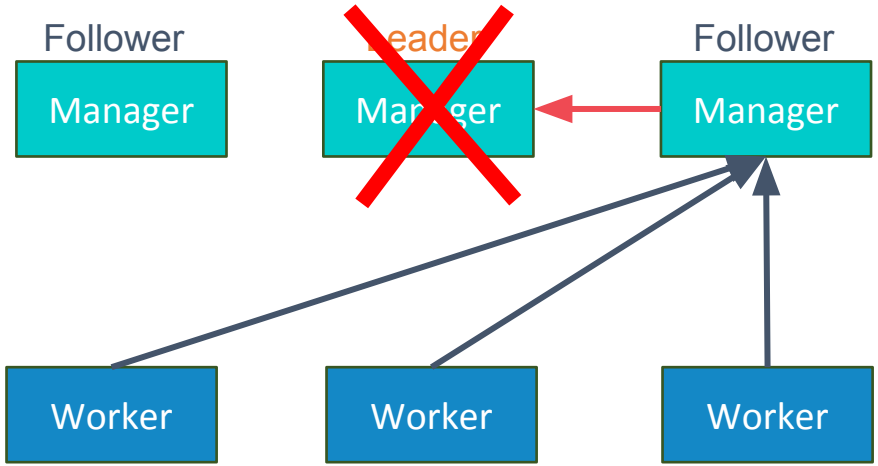
Replication



- **Followers** multiplex all **workers** to the **Leader** using a single connection
- Backed by **gRPC** channels (HTTP/2 streams)
- Reduces **Leader** networking load by spreading the connections evenly

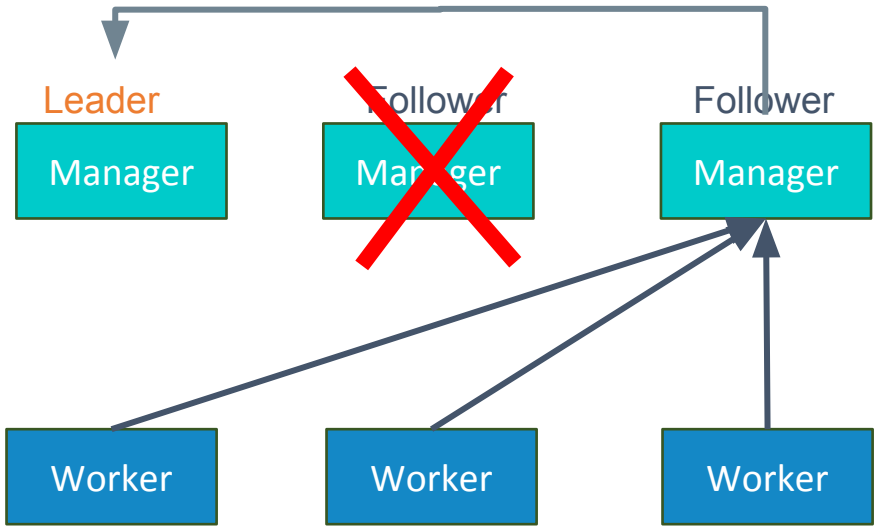
Example: On a cluster with 10,000 workers and 5 managers, each will only have to handle about 2,000 connections. Each follower will forward its 2,000 workers using a single socket to the leader.

Replication



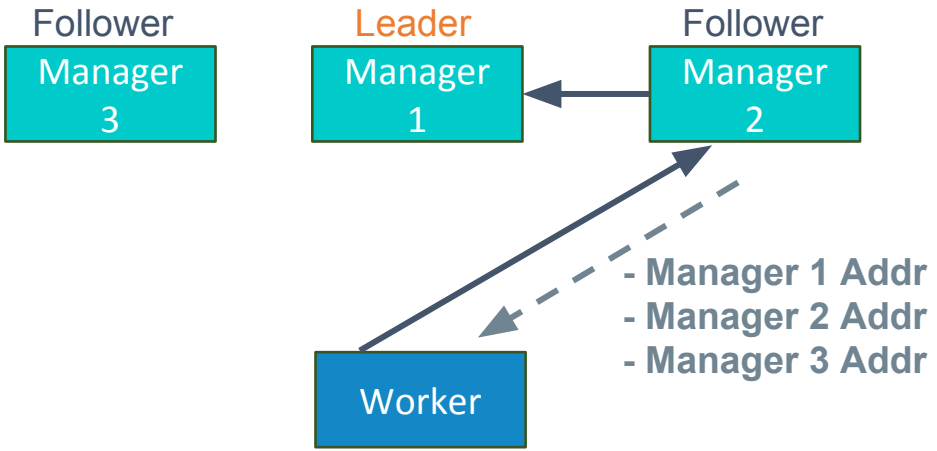
- Upon **Leader** failure, a new one is elected
- All managers start redirecting worker traffic to the new one
- Transparent to workers

Replication



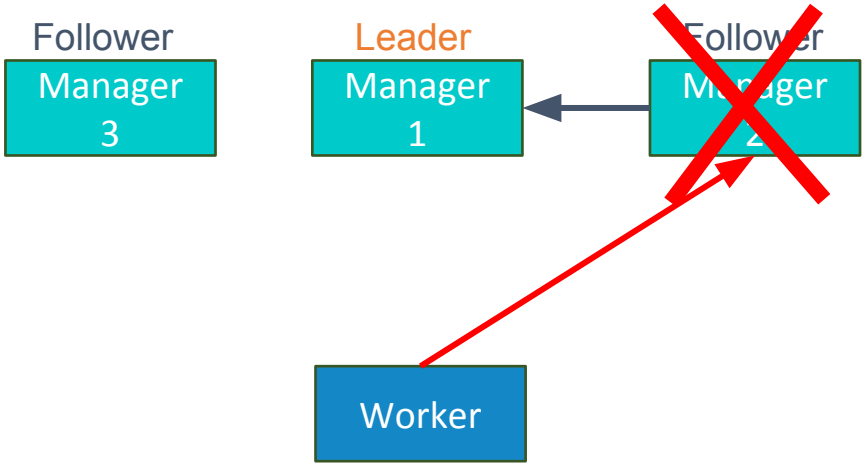
- Upon **Leader** failure, a new one is elected
- All managers start redirecting worker traffic to the new one
- Transparent to workers

Replication



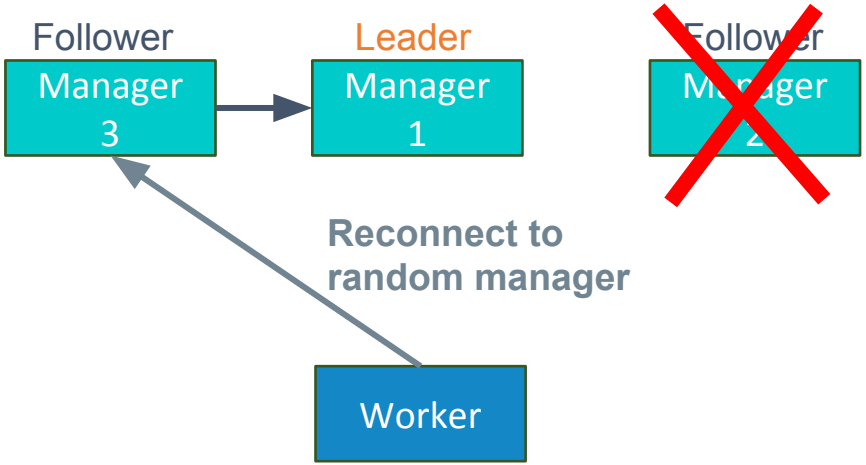
- **Manager** sends list of all managers' addresses to Workers
- When a new manager joins, all workers are notified
- Upon manager failure, workers will reconnect to a different manager

Replication



- **Manager** sends list of all managers' addresses to Workers
- When a new manager joins, all workers are notified
- Upon manager failure, workers will reconnect to a different manager

Replication



- **Manager** sends list of all managers' addresses to Workers
- When a new manager joins, all workers are notified
- Upon manager failure, workers will reconnect to a different manager

Replication

- **gRPC** handles connection management
  - Exponential backoff, reconnection jitter, …
  - Avoids flooding managers on failover
  - Connections evenly spread across Managers
- Manager Weights
  - Allows Manager prioritization / de-prioritization
  - Gracefully remove Manager from rotation

Presence

Scalable presence in a distributed environment

Presence

- Leader commits Worker state (Up vs Down) into Raft
  - Propagates to all managers
  - Recoverable in case of leader re-election

- Heartbeat TTLs kept in Leader memory
  - Too expensive to store “last ping time” in Raft
    - Every ping would result in a quorum write
  - Leader keeps worker<->TTL in a heap (time.AfterFunc)
  - Upon leader failover workers are given a grace period to reconnect
    - Workers considered **Unknown** until they reconnect
    - If they do they move back to **Up**
    - If they don’t they move to **Down**

docker

Andrea Luzzardi
al@docker.com / @aluzzardi