JS7 JobScheduler Architecture

Implementation Architecture: Components & Services

Information for Interested Parties

Software- und Organisations-Service

JS7 JobScheduler

- Workflows and Orders
  - System Architecture
  - Workflows
  - Orders
- Controller and Agent Implementation Architecture
  - Controller Cluster
  - Controller Journal
  - Controller / Agent
- JOC Cockpit Implementation Architecture
  - JOC Cockpit Cluster
  - JOC Cockpit Services
  - JOC Cockpit Background Services
  - JOC Cockpit Proxy Service

System Architecture

JOC Cockpit

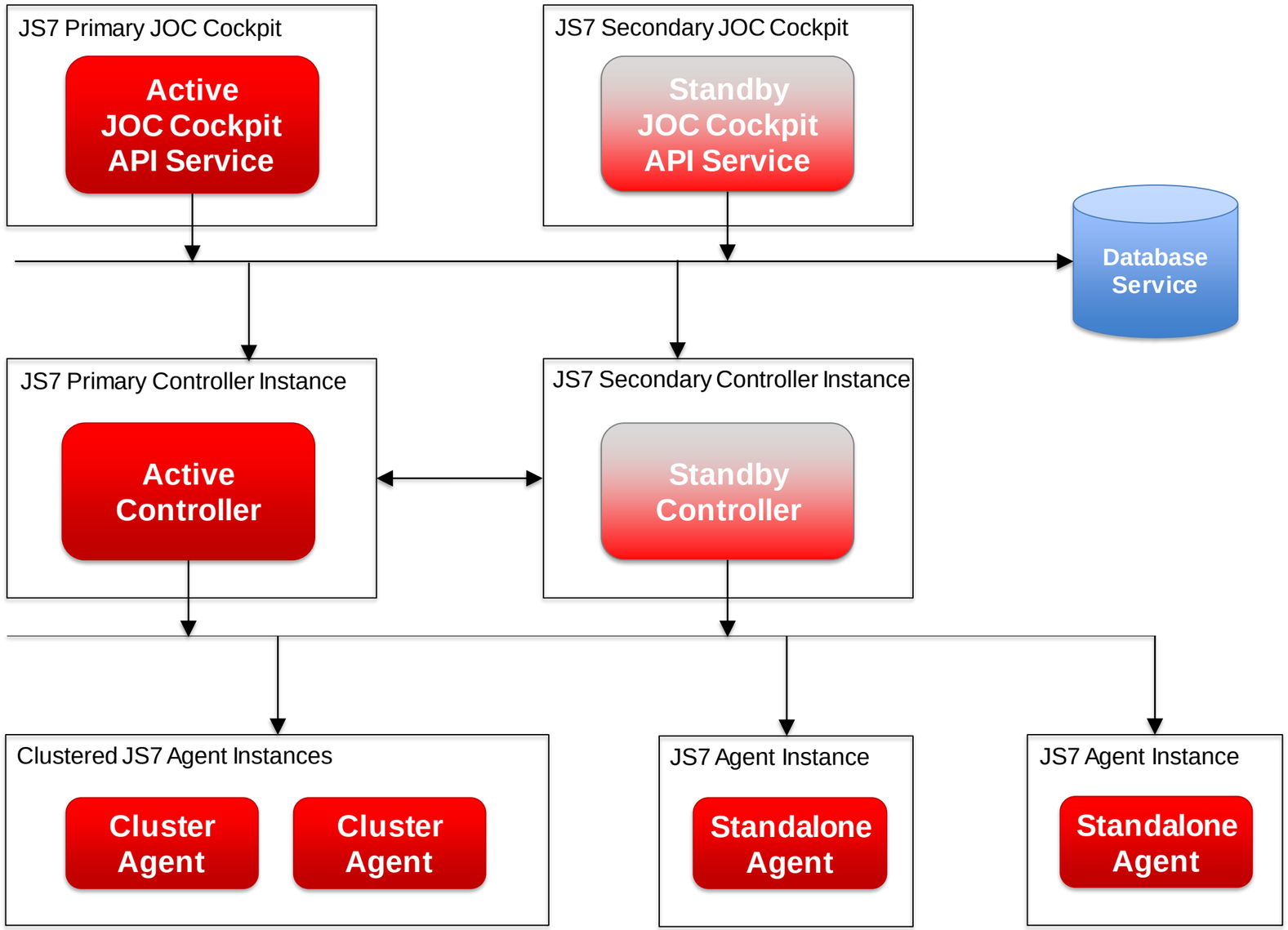
- JOC Cockpit is operated as a passive cluster or standalone and serves the User Interface and REST API Service
- Makes use of a database for persistence and for restart capabilities

Controller / Agents

- A Controller operated as a passive cluster or stand-alone orchestrates Agents
- Agents receive workflow configurations from a Controller, start workflows autonomously and report back execution results
- Agents are operated as a cluster or standalone

Connections

- Communication between components within the indicated direction of network connections



Workflows with JOC Cockpit, Controller, Agents

Workflows

JOC Cockpit / API Service

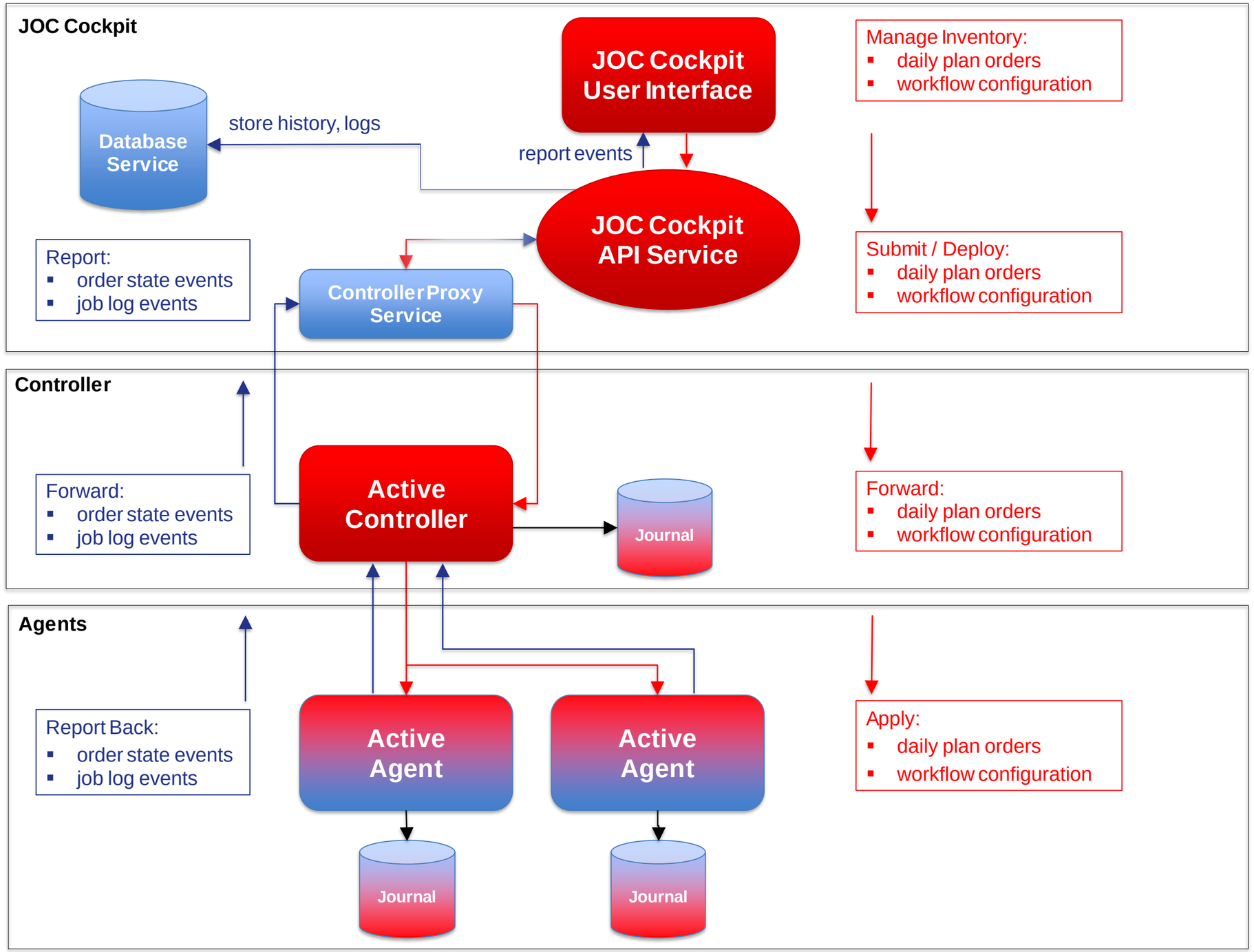
- JOC Cockpit manages the inventory of workflows & jobs and the daily plan which are deployed to a Controller
- During workflow execution JOC Cockpit receives job log output and order state events in near real-time

Controller

- The Controller checks and forwards the daily plan and workflow configuration to related Agents

Agent

- Agents start workflows / jobs autonomously / on demand:
  - jobs in workflows can be executed with mixed Agents
  - Agents execute workflows autonomously within the scope of the daily plan
  - Agents report back to the Controller any log output and events, for example when starting or completing a task



Orders with JOC Cockpit, Controller, Agents

Orders

JOC Cockpit / API Service

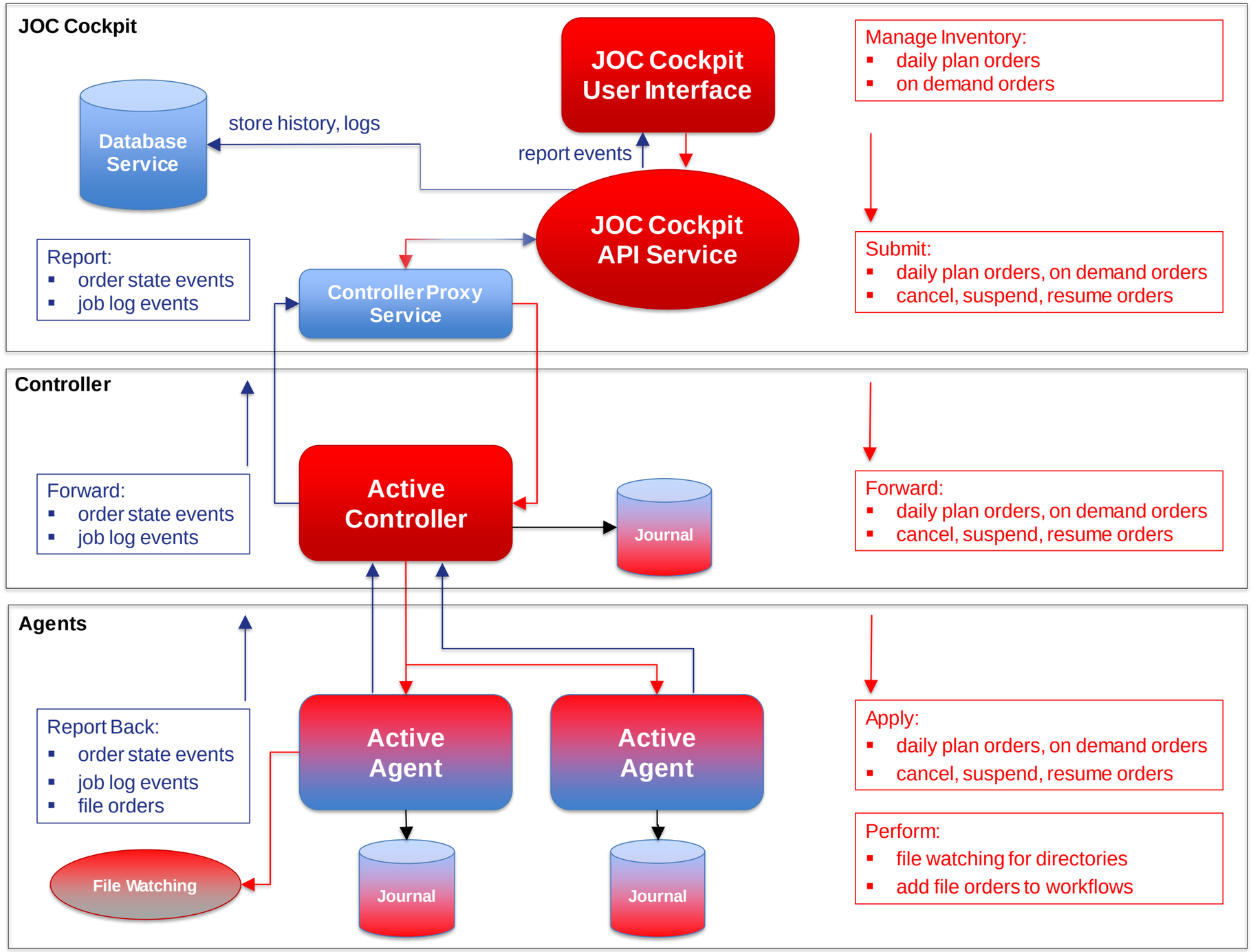
- JOC Cockpit manages orders from the daily plan and orders on demand
- Orders are submitted to a Controller for execution of workflows with Agents

Controller

- The Controller forwards orders from the daily plan and orders on demand to related Agents

Agent

- Agents start workflows / jobs autonomously / on demand
- Agents report back resulting order state transition events and log output events
- Agents watch directories for incoming files and create file orders
- Agents handle any number of orders for the same workflow and for different workflows in parallel



Software- und Organisations-Service

JS7 JobScheduler

- Workflows and Orders
  - System Architecture
  - Workflows
  - Orders
- Controller and Agent Implementation Architecture
  - Controller Cluster
  - Controller Journal
  - Controller / Agent
- JOC Cockpit Implementation Architecture
  - JOC Cockpit Cluster
  - JOC Cockpit Services
  - JOC Cockpit Background Services
  - JOC Cockpit Proxy Service

Controller Cluster Management

Controller Cluster using JOC Cockpit as Cluster Watch

Communication

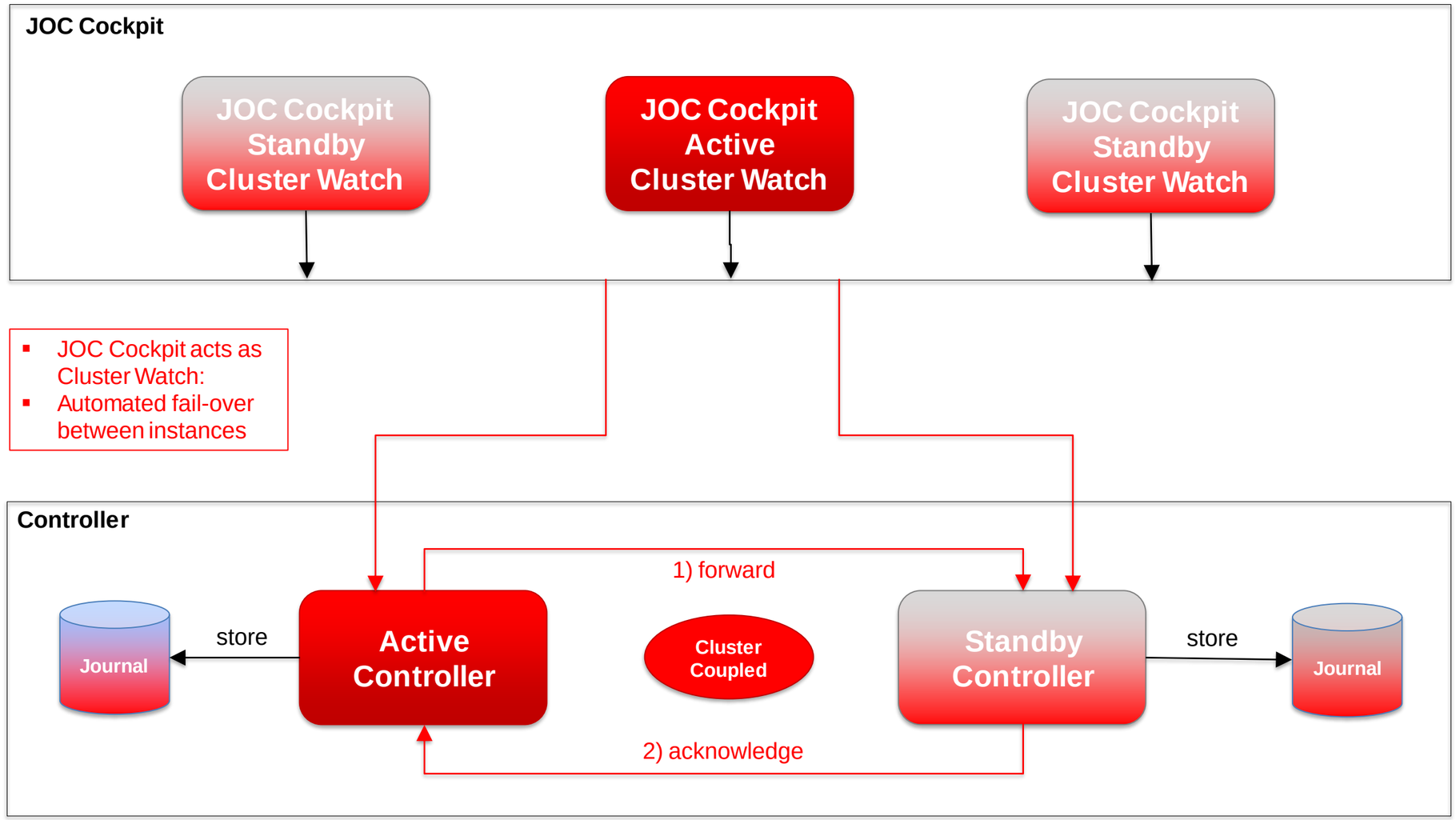
- Both Active/Standby Controller instances establish HTTP(S) connections to each other

Coupling

- The Active Controller adds changes to objects and order state transitions to its journal and synchronizes with the Standby Controller instance
- The Standby Controller adds such information to its journal and acknowledges receipt
- When Active and Standby Controller instances are in sync then the Cluster is considered being coupled
- Recoupling occurs as needed

Fail-over

- In case of failure of a Cont-roller instance or connection the Cluster Watch is consulted to determine which Controller instance should take over the active role
- Fail-over occurs within 15s



Controller Cluster Management

Controller Cluster using an Agent as Cluster Watch

Communication

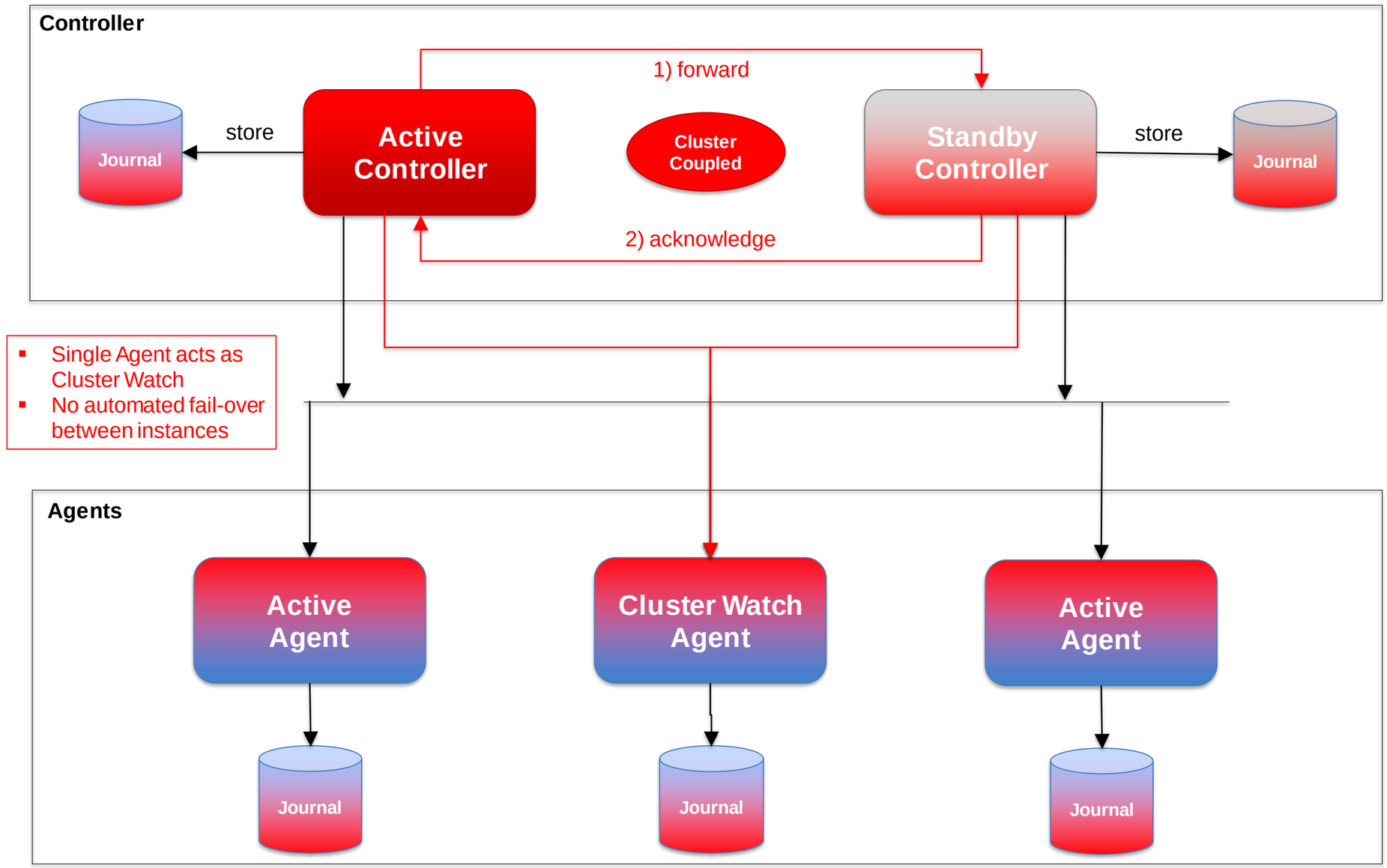
- Both Active/Standby Controller instances establish HTTP(S) connections to each other

Coupling

- The Active Controller adds changes to objects and order state transitions to its journal and synchronizes with the Standby Controller instance
- The Standby Controller adds such information to its journal and acknowledges receipt
- When Active and Standby Controller instances are in sync then the Cluster is considered being coupled
- Recoupling occurs as needed

Fail-over

- In case of failure of a Cont-roller instance or connection the Cluster Watch Agent is consulted to determine which Controller instance should take over the active role
- Fail-over occurs within 15s



Controller Integration with JOC Cockpit

Controller Integration

Controller Journal

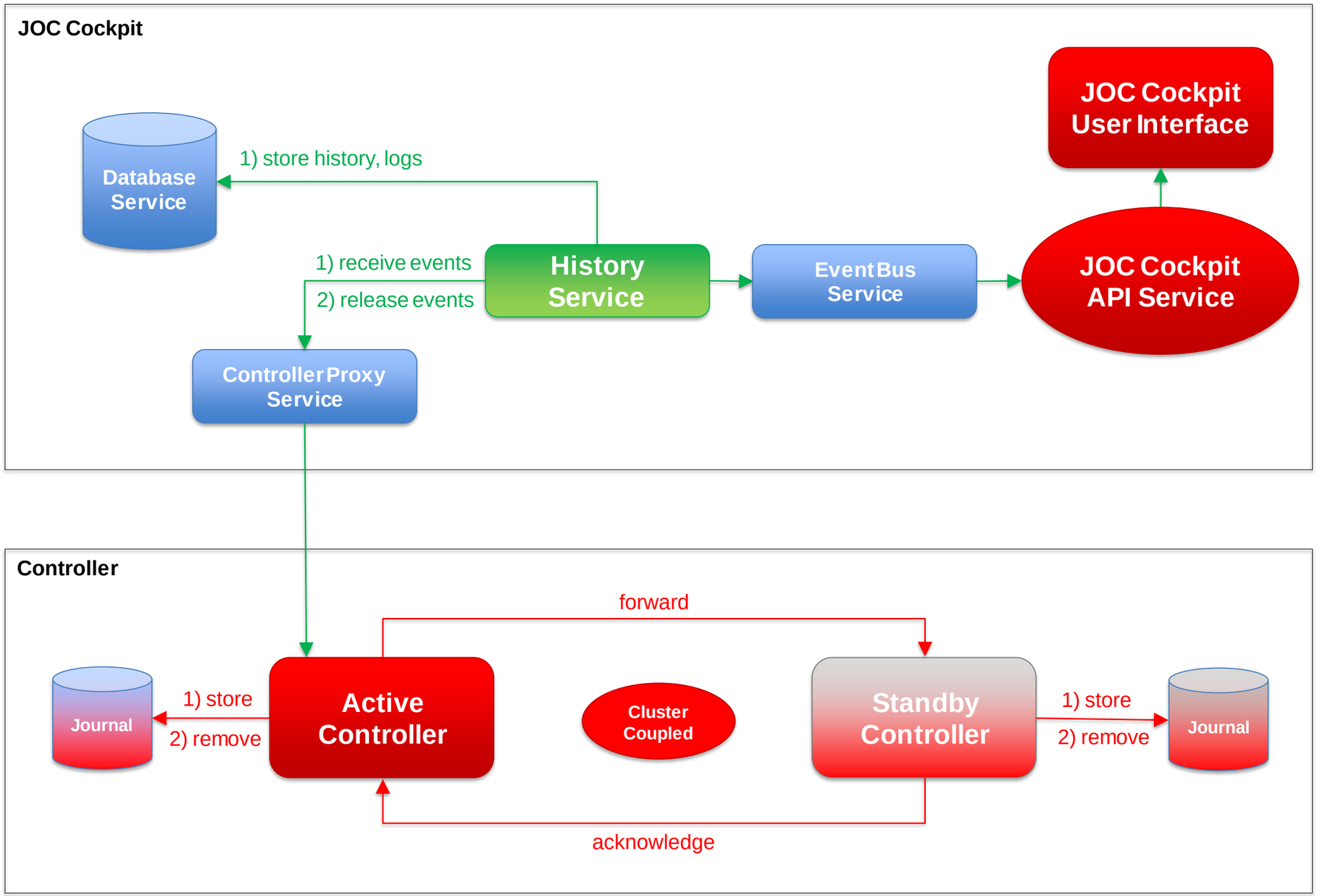
- The Journal holds objects such as order state transition events and log events of a Controller
- Such objects are synchronized with the Standby instance

History Service

- The History Service subscribes to events of the Controller
- Having received events and having stored them to the database the service forwards events to the GUI and instructs the Controller to release events

Controller

- Events are originally stored to the Journal after receipt from an Agent or originating from workflow instructions
- Events are removed from the Journal when released by the History Service
- Journal size can grow with the number of objects, but will shrink when orders are completed and events are released



Controller / Agent Communication

Controller / Agent

Controller

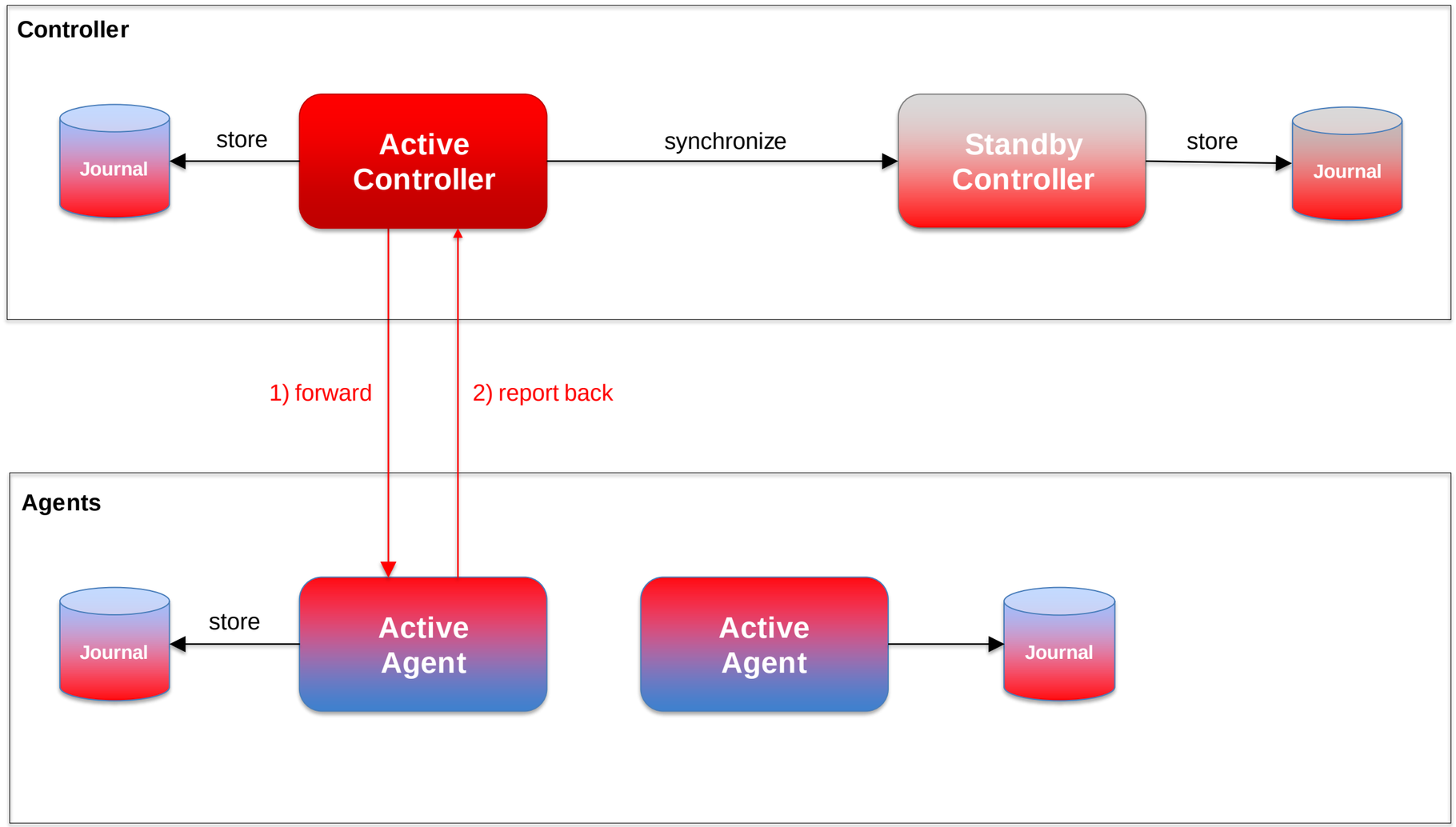
- All Controller instances store workflow configurations and order state transitions in their journals for synchronization
- These objects are passed asynchronously to Agents

Agent

- Agents receive objects and store them in a journal
- Agents execute jobs independently from an active connection to a Controller
- Agents report back the resulting order state events and log events, e.g. after job completion

Communication

- If Controller, Agent or the connection between them fail then they will reconnect
- Communication recovers in case of longer outages for hours and days



Software- und Organisations-Service

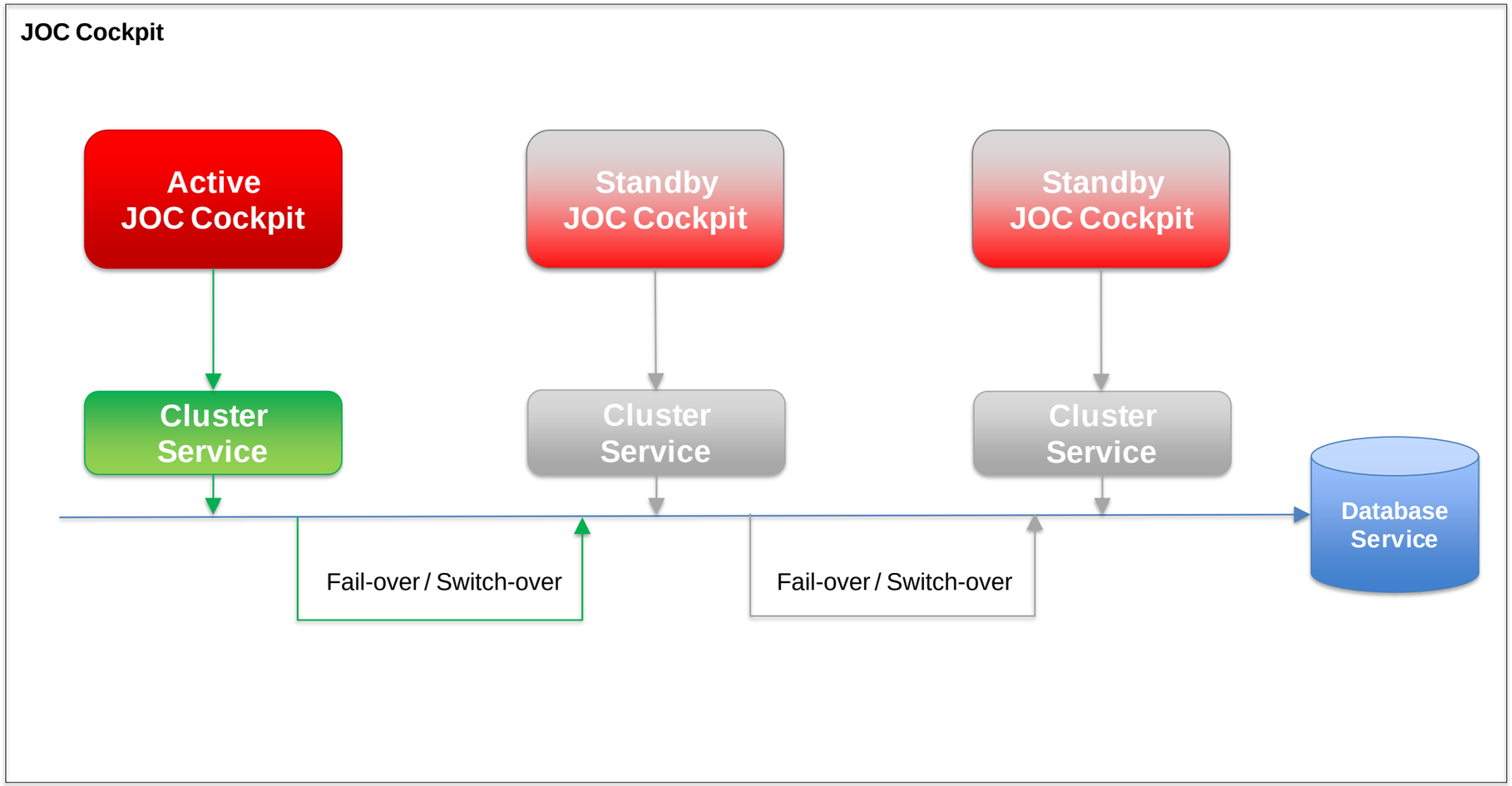
JS7 JobScheduler

- Workflows and Orders
  - System Architecture
  - Workflows
  - Orders
- Controller and Agent Implementation Architecture
  - Controller Cluster
  - Controller Journal
  - Controller / Agent
- JOC Cockpit Implementation Architecture
  - JOC Cockpit Cluster
  - JOC Cockpit Services
  - JOC Cockpit Background Services
  - JOC Cockpit Proxy Service

JOC Cockpit Cluster

JOC Cockpit Cluster fail-over and switch-over

- Cluster Service instances are synchronized by use of the database to which they send heartbeats and check availability of each other instance
- In case of failure one of the remaining instances will perform a cluster fail-over operation
- Users can perform a switch-over operation by selecting the next active JOC Cockpit instance
- In case of switch-over the Cluster Service will stop any running Background Services normally
- For fail-over / switch-over the Background Services are started from the Cluster Service of the next active JOC Cockpit instance



JOC Cockpit Services

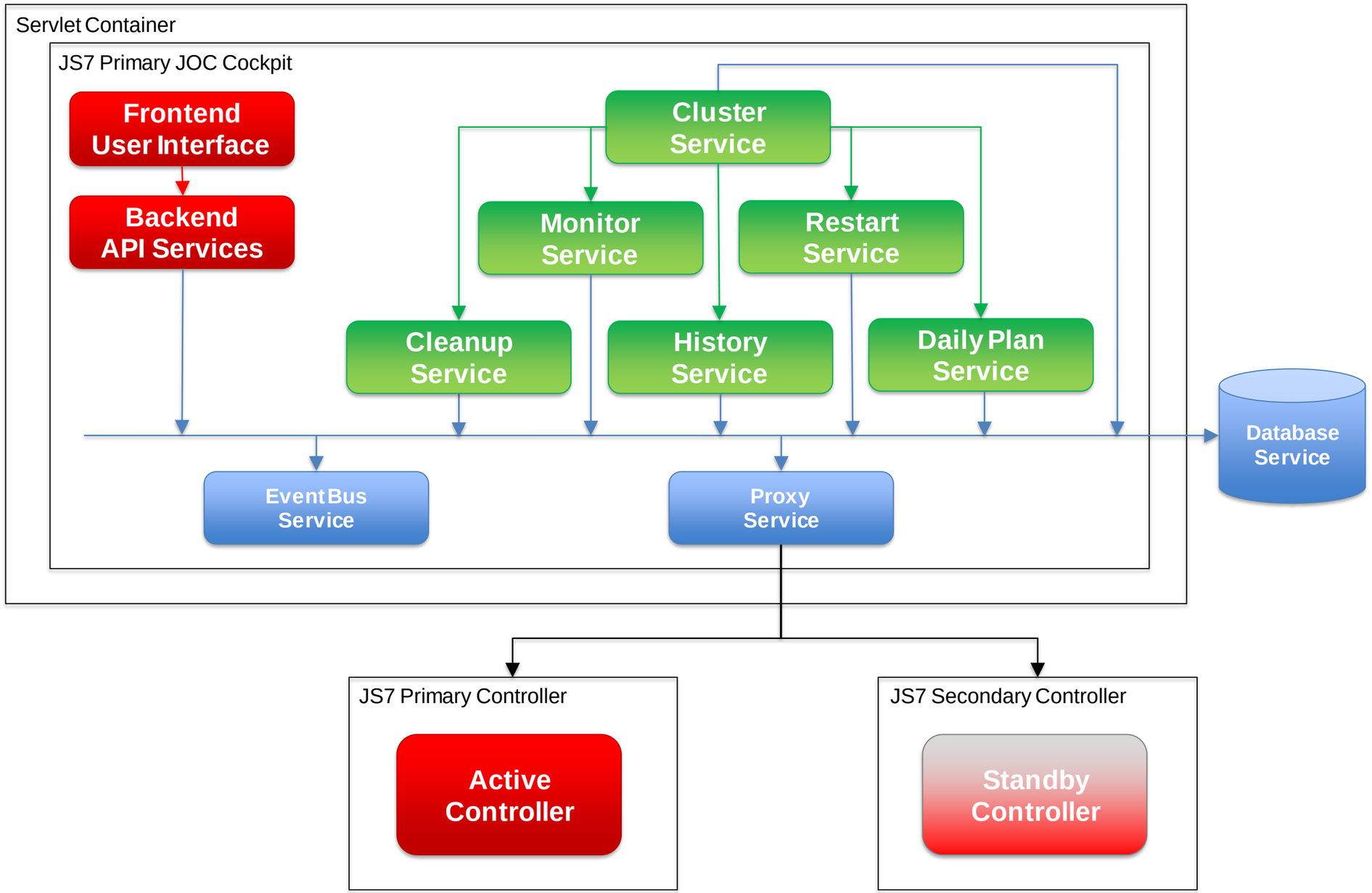
JOC Cockpit Frontend/Backend Services, Background Services, Event Bus and Proxy

- JOC Cockpit is operated in a servlet container
- Frontend User Interface for browser access
- Backend API Services provide information to the GUI frontend and to clients using the REST API

- The Cluster Service manages a number of Background Services for housekeeping, history and daily plan management

- Communication between Backend API Services and Background Services is based on an Event Bus
- The Proxy Service reports order state transitions occurring in a Controller or Agent

- Any JOC Cockpit service can access the database service to store and to retrieve information



JOC Cockpit Background Services

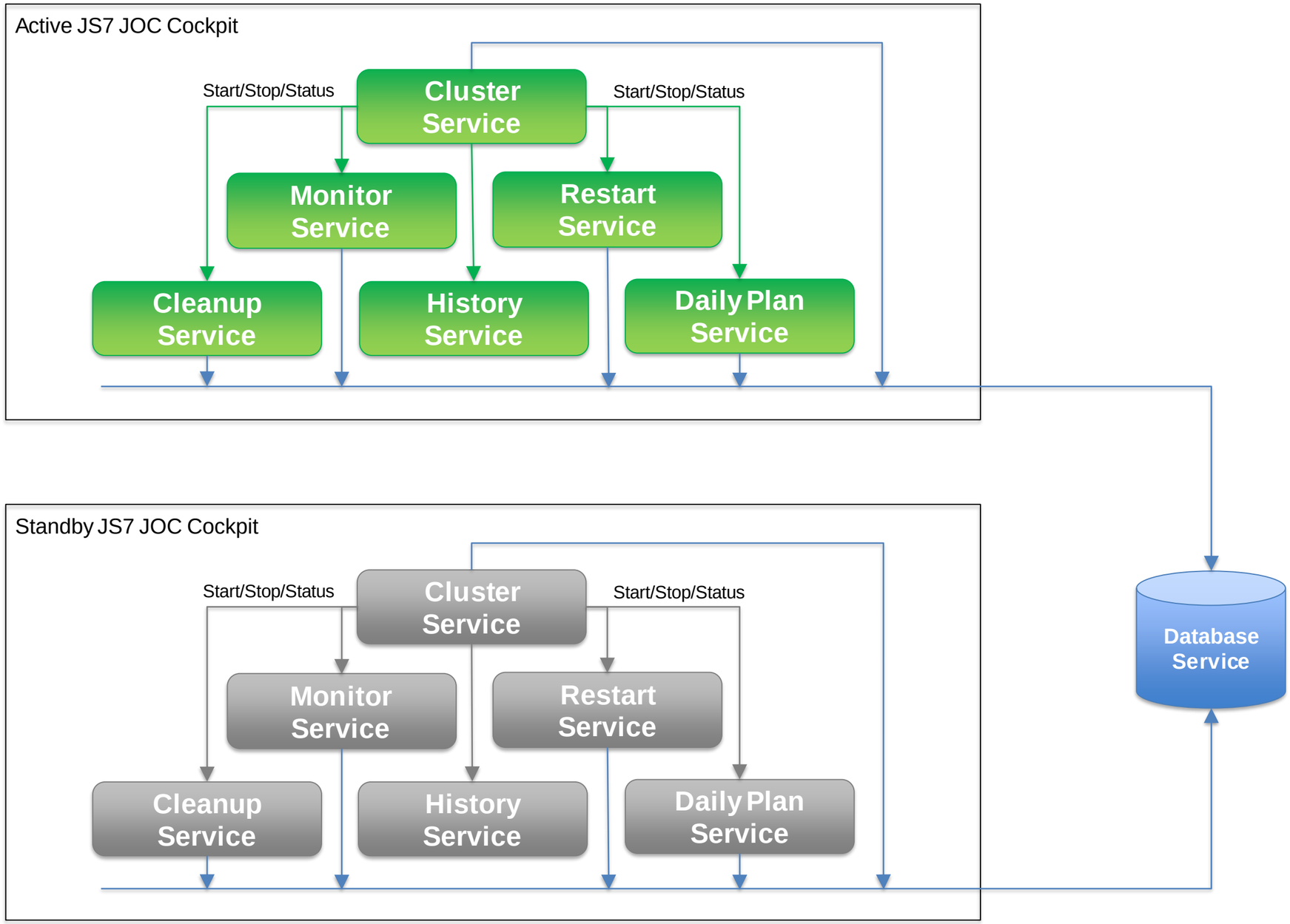
JOC Cockpit clustered Background Services

- The Cluster Service manages Background Services running in the servlet container
- Background Services are started, stopped etc.

- Cluster Service manages fail-over to the next JOC Cockpit instance in case of service failure

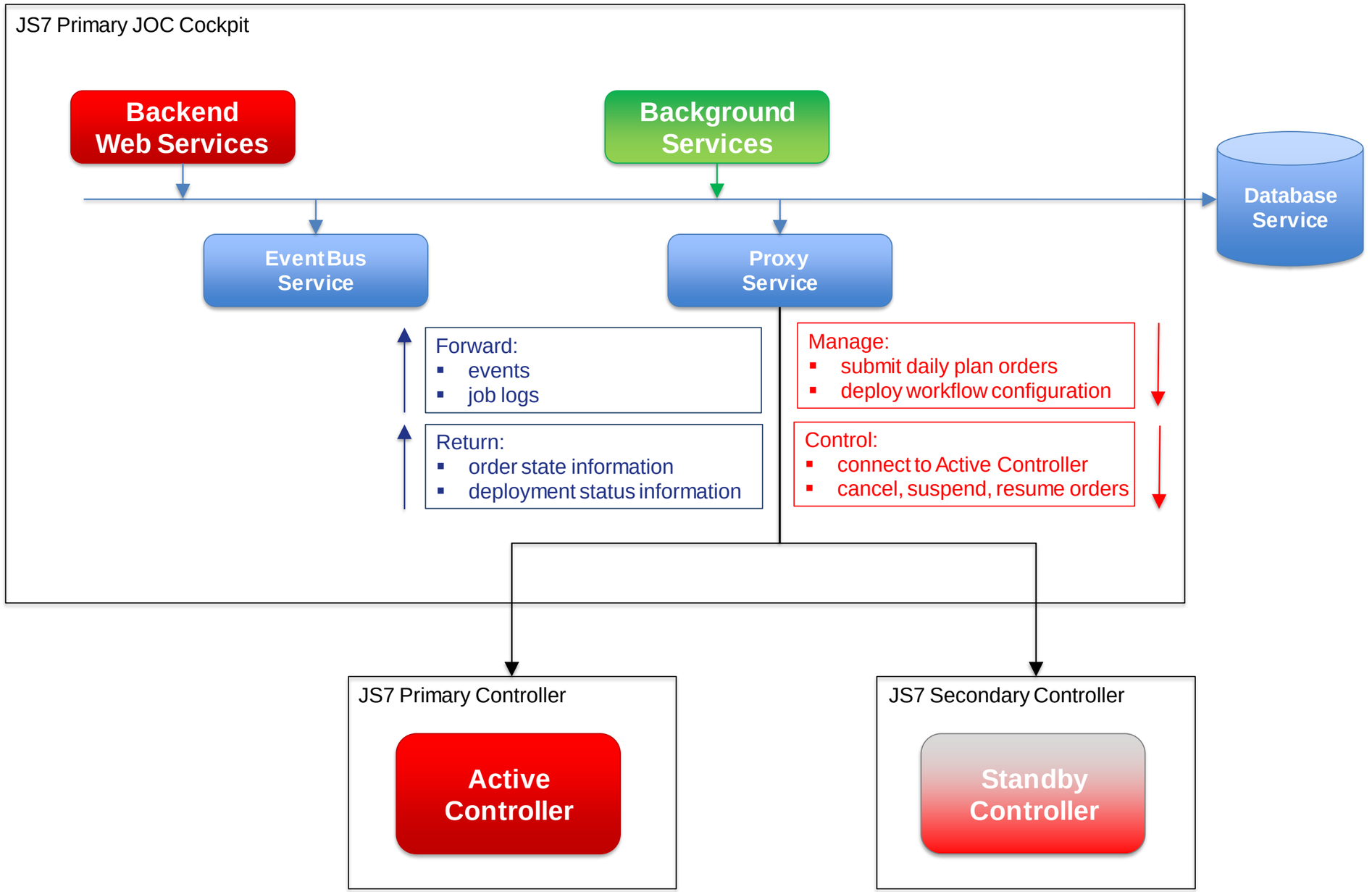
- Monitor Service notifies about failed jobs and component failures etc.
- Restart Service reruns pending deployments and performs synchronization with a Controller

- Cleanup Service purges the database, e.g. to limit the size of the history
- History Service retrieves execution results and logs from a Controller instance
- Daily Plan Service creates and submits orders to connected Controllers



JOC Cockpit Proxy Service

- The Proxy connects to the active Controller instance, supports fail-over and manages asynchronous messages
- The Proxy deploys configuration objects, submits orders to the Controller
- The Proxy handles asynchronous operations such as cancel, suspend, resume etc. for orders with the Controller
- The Proxy returns the order state and deployment status of objects
- The Proxy forwards asynchronous events including order state transitions and log output of jobs from the Controller
- Information returned or forwarded by the Proxy is added to the Event Bus



Questions?

Comments?

Feedback?

Software- und
Organisations-
Service GmbH

Giesebrechtstr. 15
D-10629 Berlin

info@sos-berlin.com
https://www.sos-berlin.com