OPENFABRICS ALLIANCE

2024 OFA Virtual Workshop

A DEEP REINFORCEMENT AGENT FOR RESOURCE SCHEDULING WITH SUNFISH IN A COMPOSABLE DISAGGREGATED INFRASTRUCTURE

Catherine Appleby, Computer Scientist, Applied Machine Intelligence

Michael Aguilar, EIT, PI, Senior Computer Scientist, HPC Research and Development

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories in a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2024-04819PE

A Deep Reinforcement Agent for Resource Scheduling with Sunfish in a Composable Disaggregated Infrastructure

Catherine Appleby, Computer Scientist,
Applied Machine Intelligence, Sandia National Labs

Michael Aguilar, EIT, PI, Senior Computer Scientist,
HPC Research and Development, Sandia National Labs

Christian Pinto, Phil Cayton, Russ Herrell, Richelle Ahlvers, Alex Lovell-Troy

OpenFabrics Alliance Workshop

April, 2024

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2024-04819PE

A Deep Reinforcement Agent for Resource Scheduling with Sunfish in a Composable Disaggregated Infrastructure

1. Why Composable Disaggregated Infrastructure (CDI)
2. Design Considerations for a Composability Manager on a Large-Scale HPC system
3. Sunfish
4. Deep Reinforcement Learning for Resource Allocation
5. Integrations
6. Acknowledgements and Questions

Why Composable Disaggregated Infrastructure (CDI)

Current Beowulf architectures

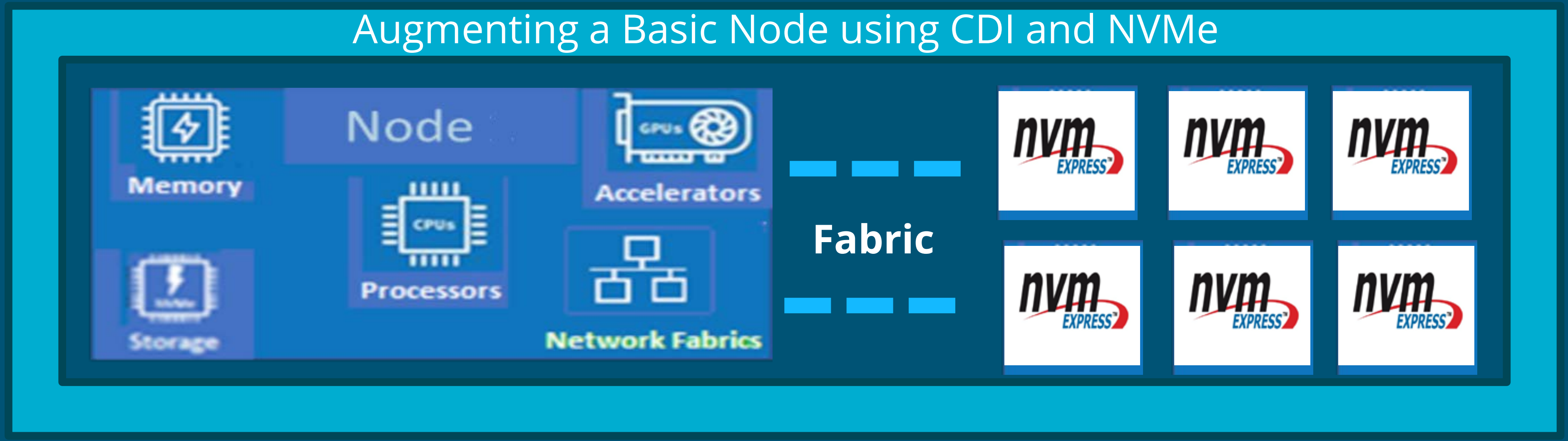
- Larger HPC systems create a larger potential impact of stranded resources
- Resource limits are fixed for each compute node
- Need to build out components to address all possible types of application codes that the HPC must support
- Hardware failures during run-time can kill running batch applications

CDI

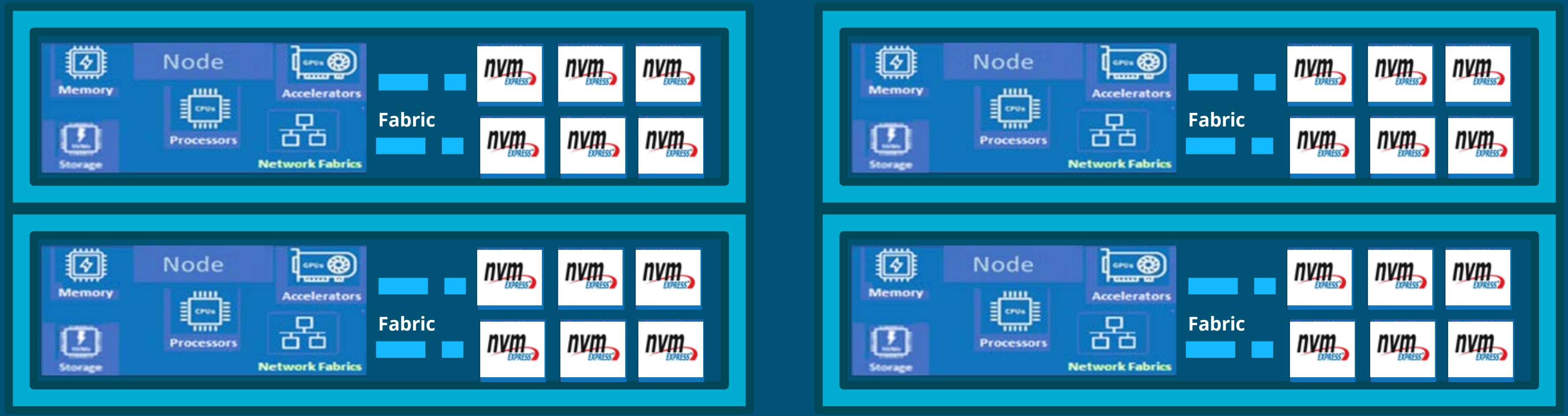
- Mitigate stranded 'wasted' resources
- Dynamically utilize hardware resources such as CPUs, GPUs, and memory
  - Enables better application workload matching to increase application efficiency
- Dynamically apply resources to potentially abate out-of-memory conditions, abate IO thrashing, route around network connection failures, and reduce batch job failures due to hardware failures

Fun fact: 2% of the US's energy consumption is input into Datacenters: https://www.energy.gov/eere/buildings/data-centers-and-servers

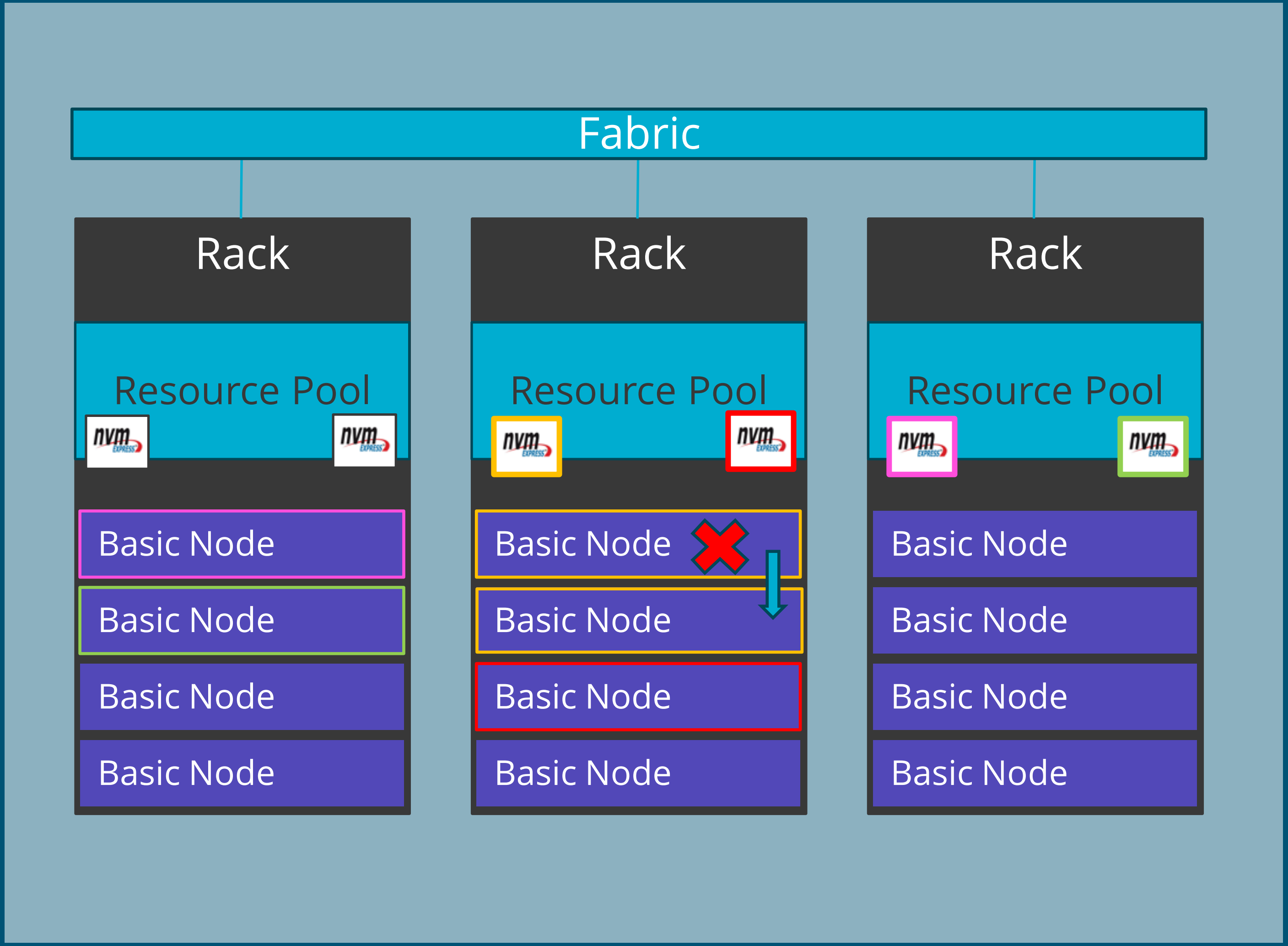
CDI – Software Derived Architectures



If we need more Storage servers to mitigate load issues, we can compose additional servers and automatically add them into the storage pool.



CDI – Software Derived Architectures



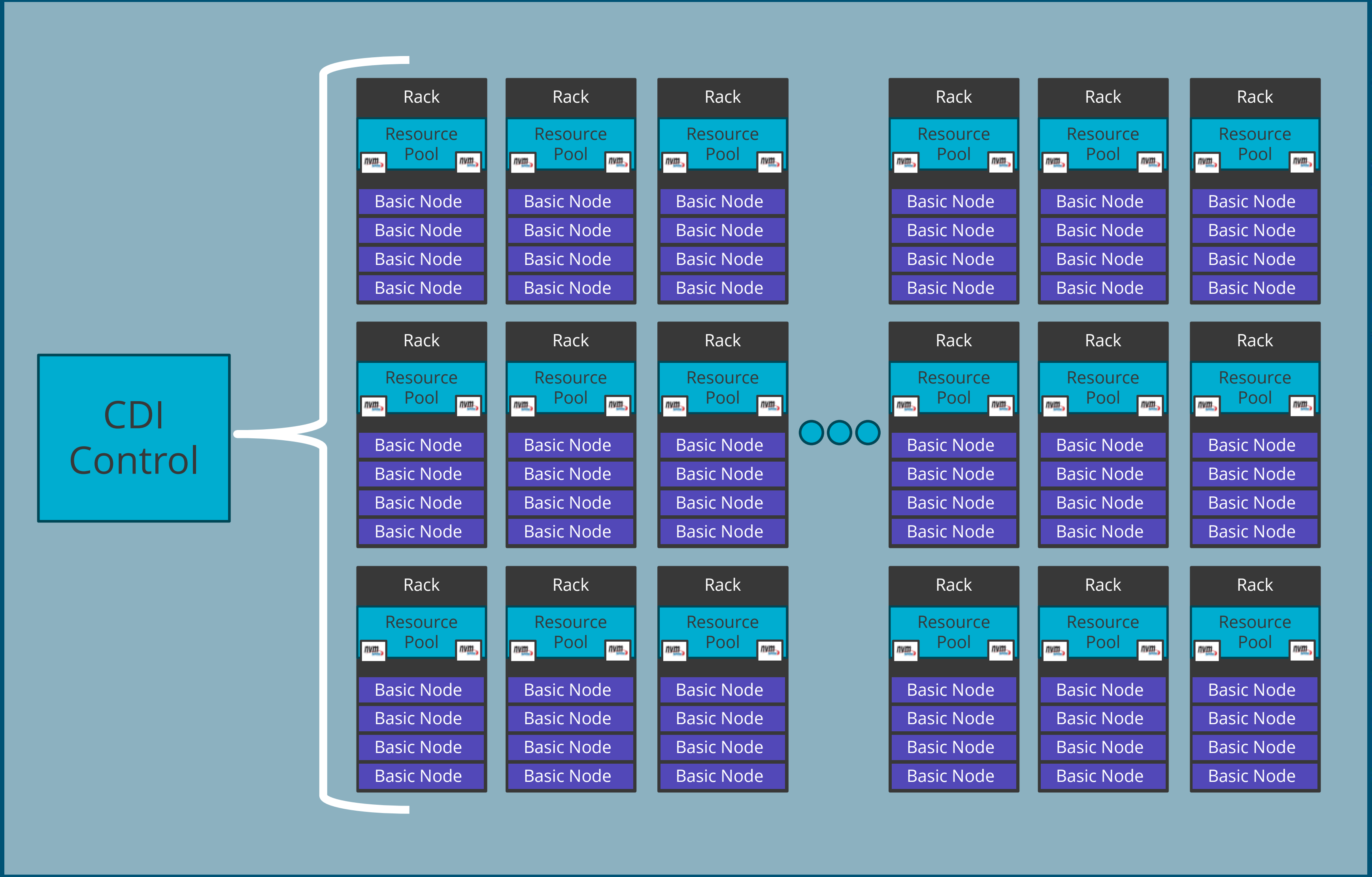
If there is node failure, dynamically swap it out

- We can leave a malfunctioning node behind, allocate another node, and **utilize the memory**, from the pool, that the other node was using, **seamlessly**.

Scaling the Control Structure to Very Large Systems

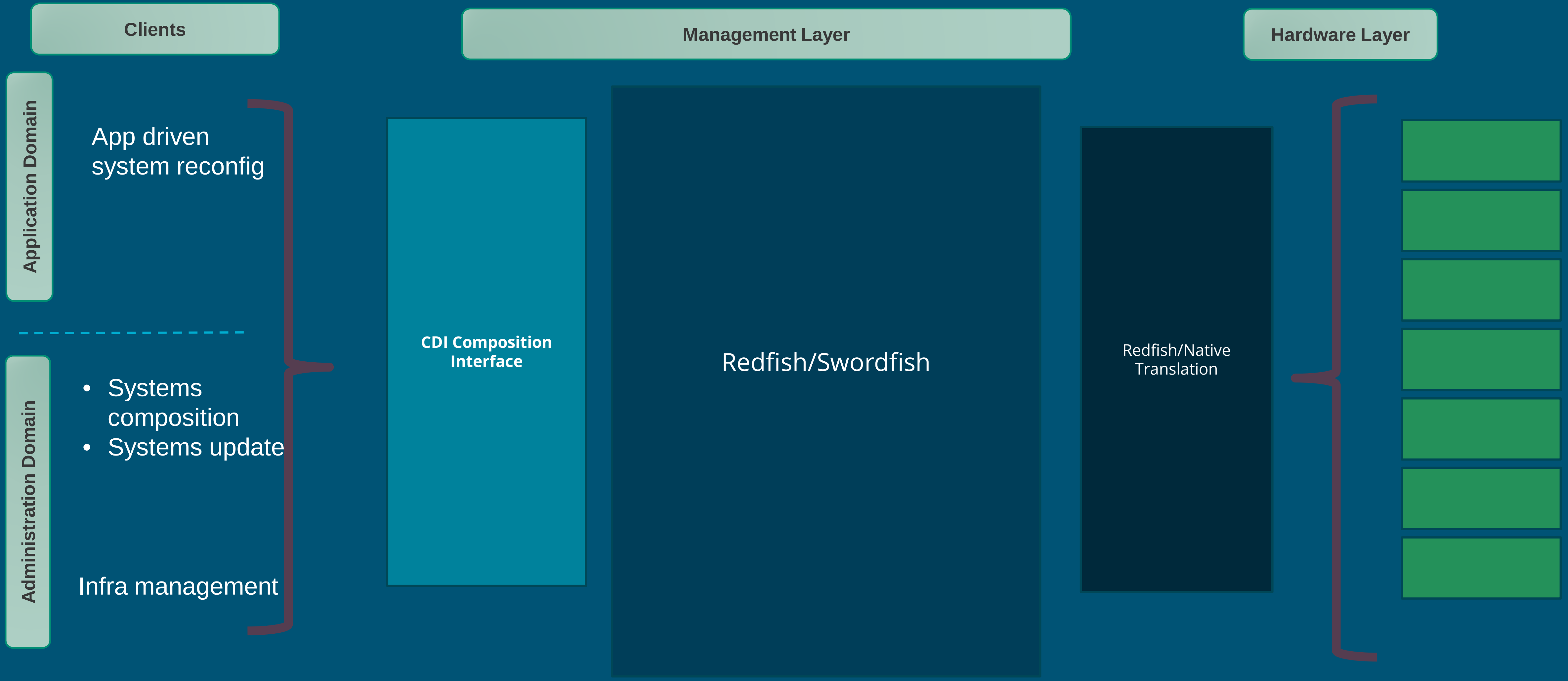
CDI Control

- Need to keep track of a huge number of concurrent resources
- Need to keep management and query communications down to a reasonable quantity
- Need to be able to execute timely changes to the HPC system as those changes are requested

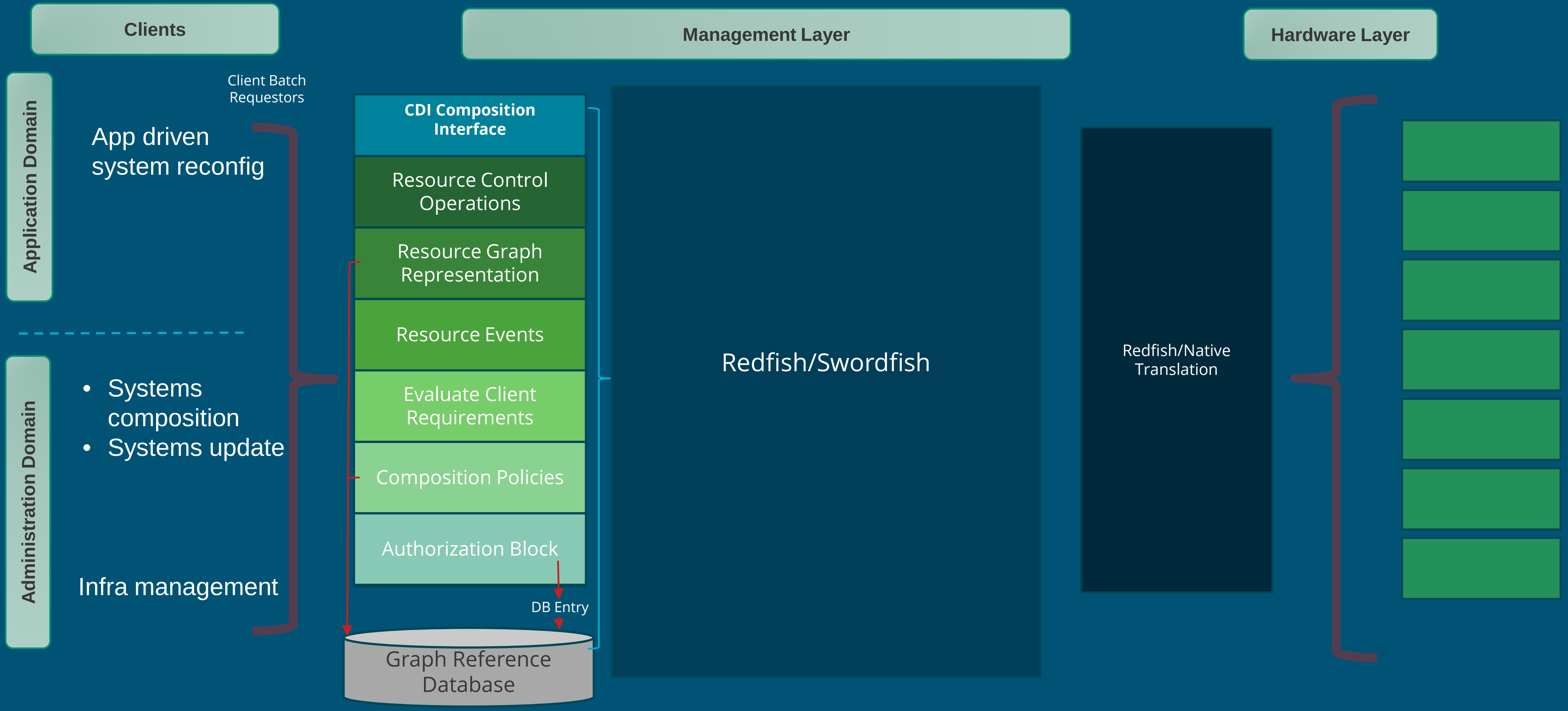


Introducing Sunfish

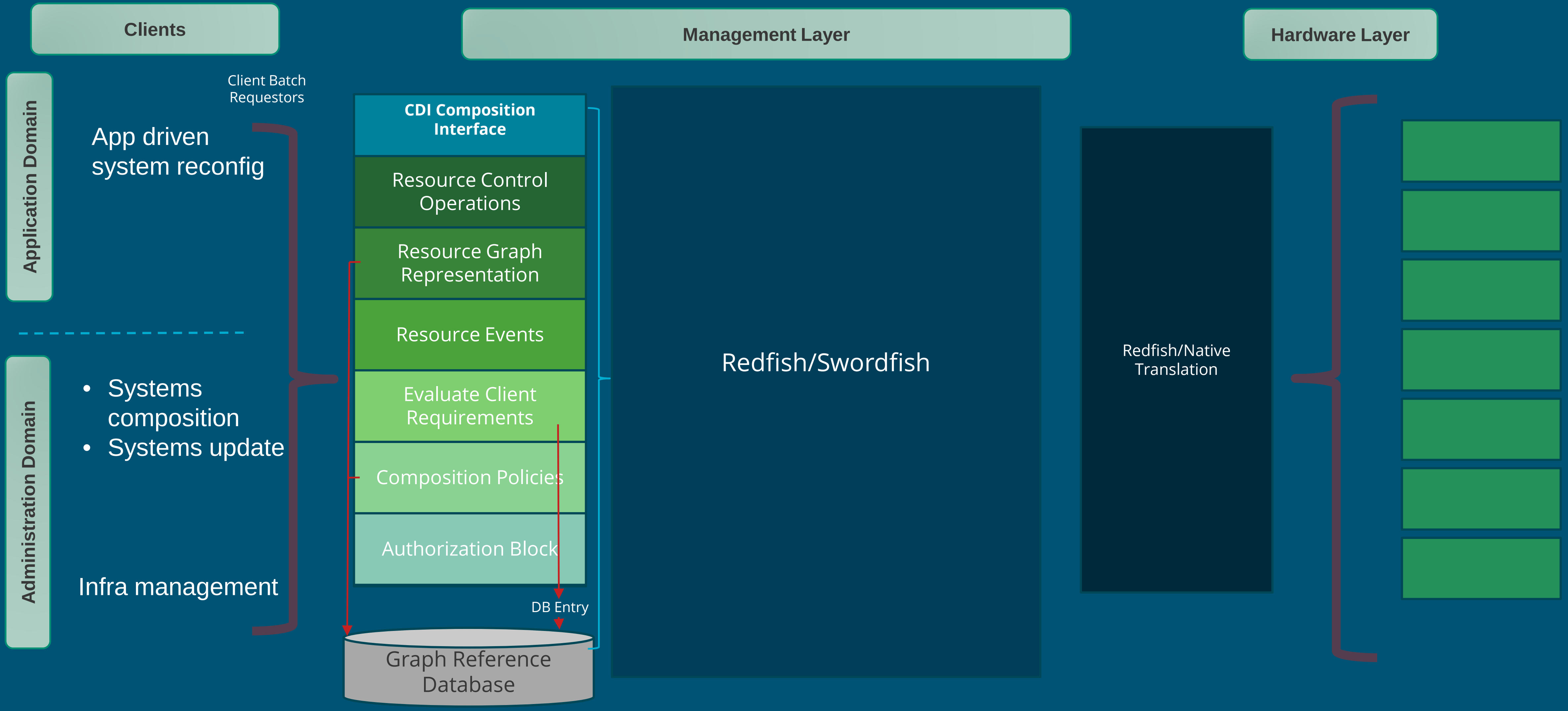
Introducing Sunfish



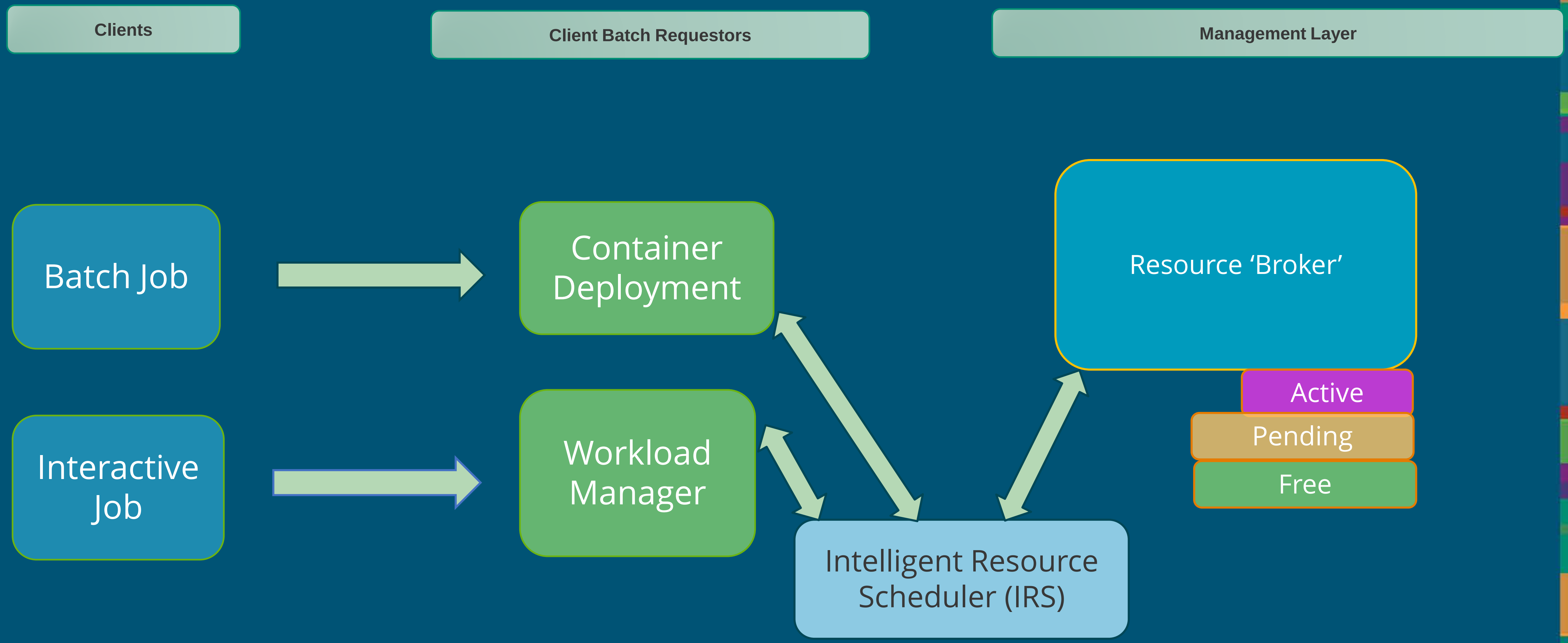
Introducing Sunfish



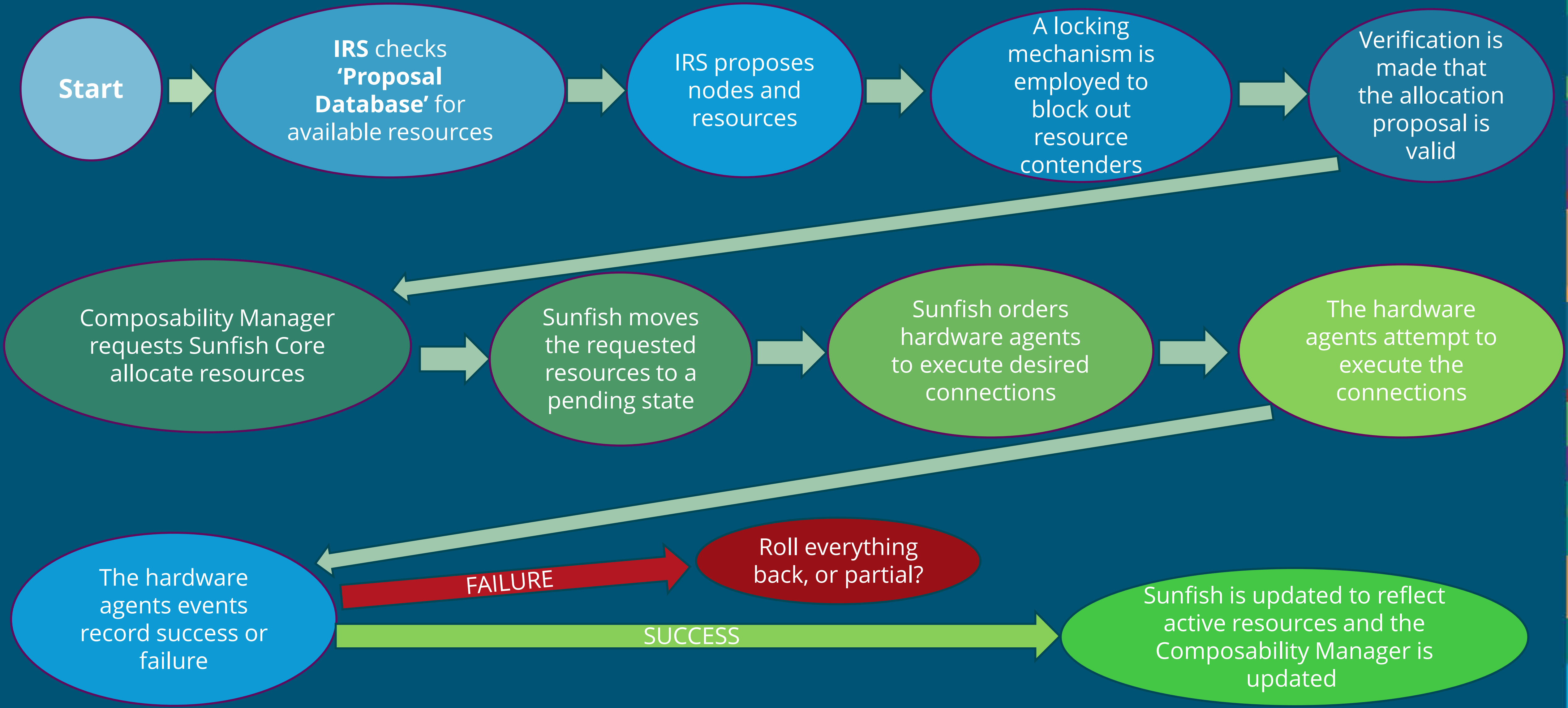
Introducing Sunfish



Introducing Sunfish

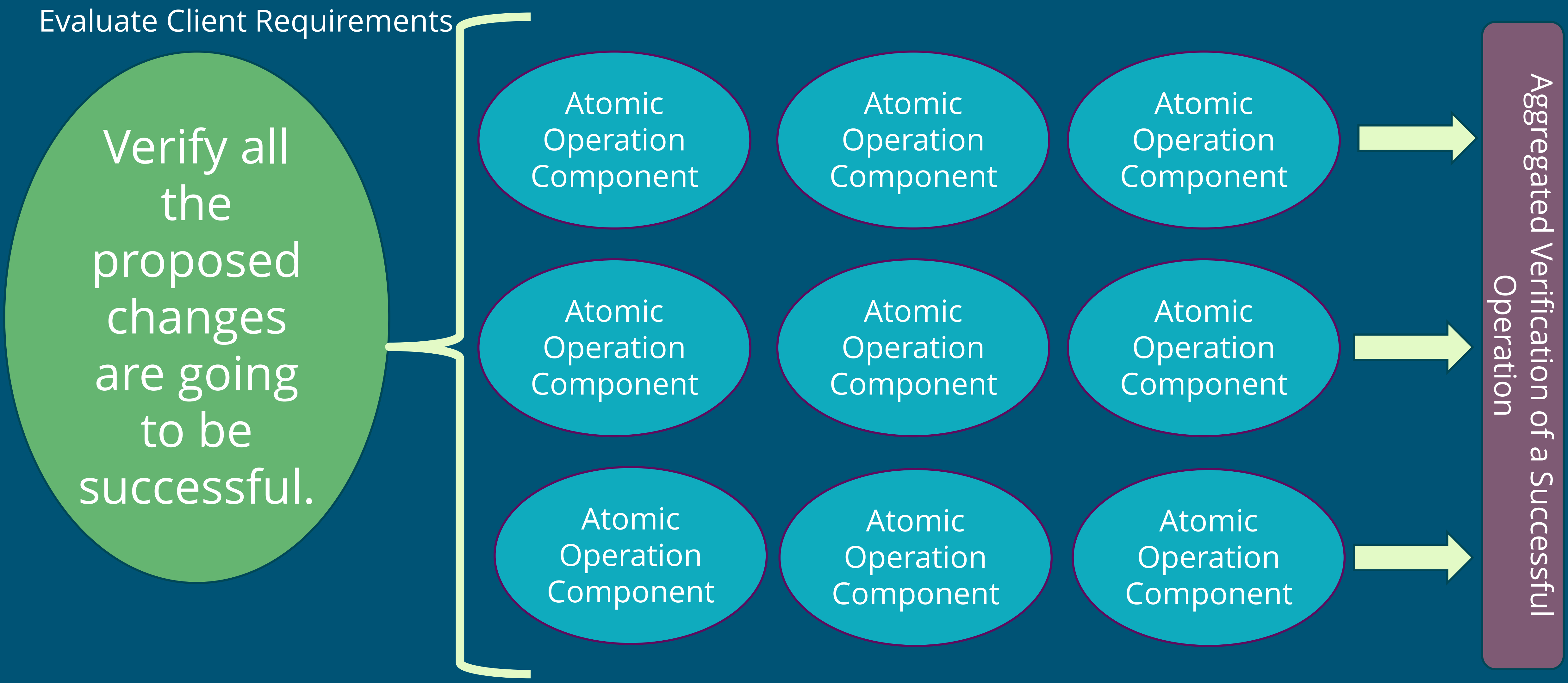


Proposed Atomic Operation Component

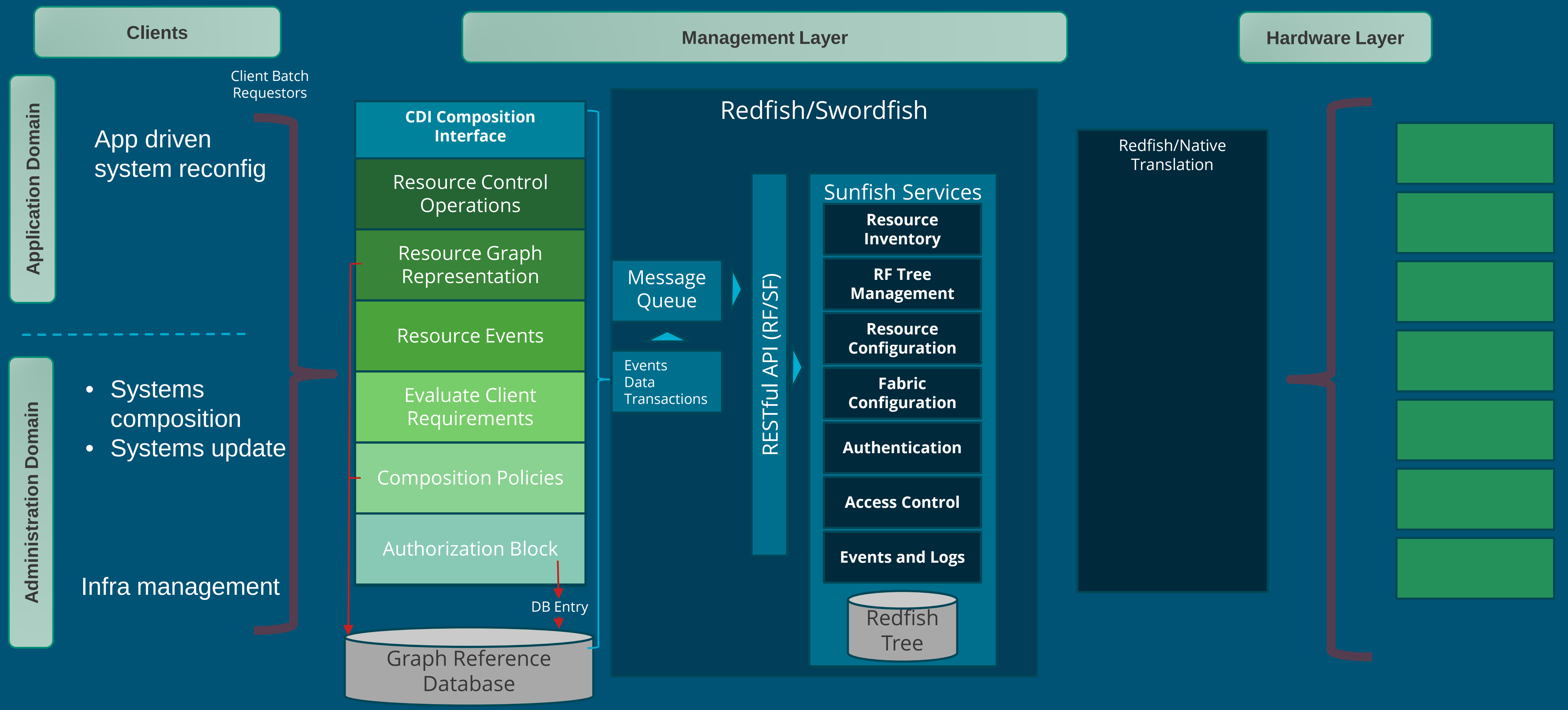


Introducing Sunfish

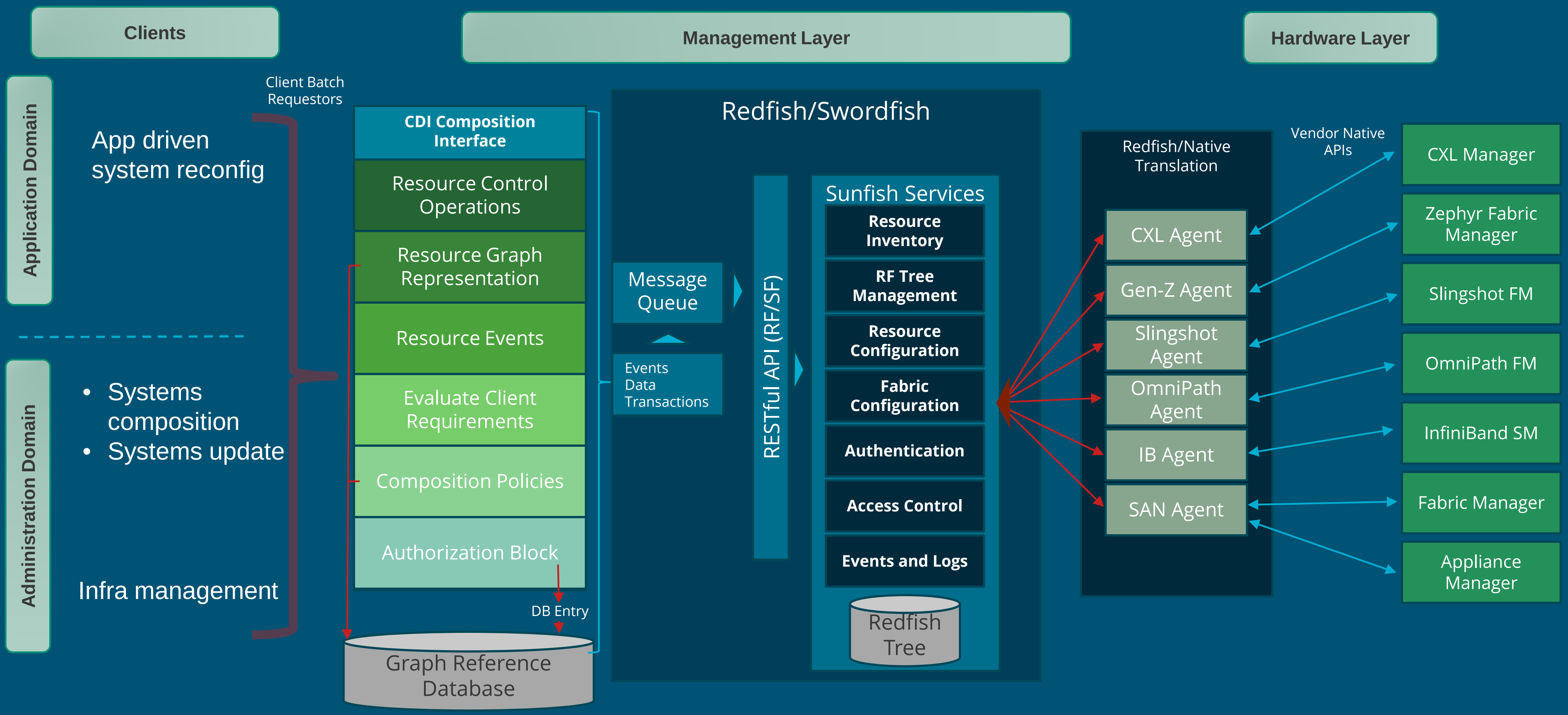
Each batch request for Software Defined Nodes is a single imperative operation
Do we have a partial success or failure?



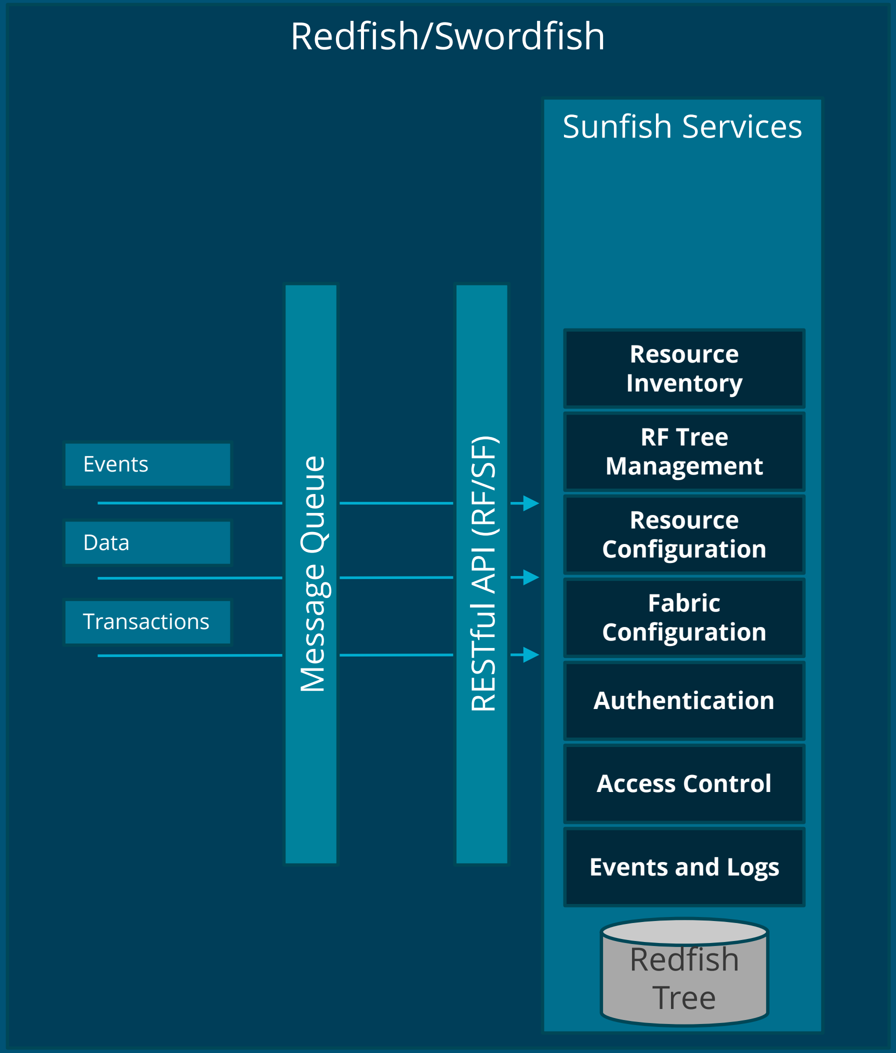
Introducing Sunfish



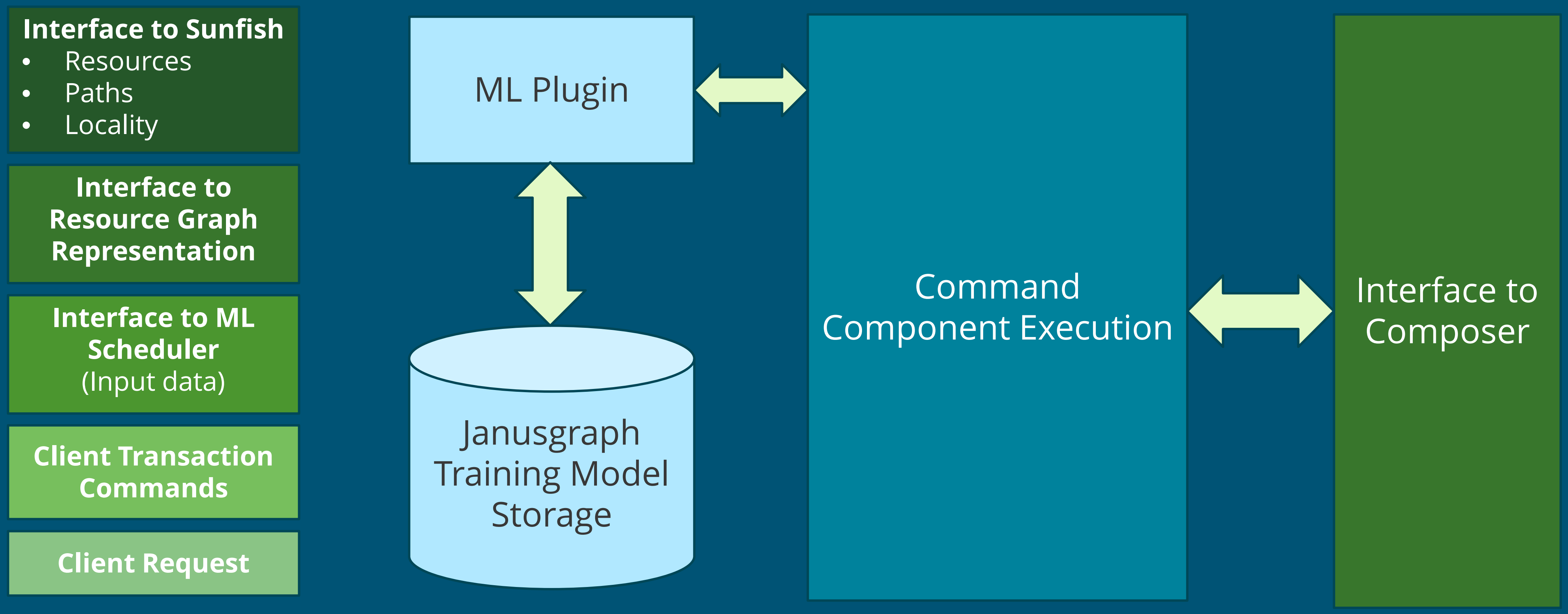
Introducing Sunfish



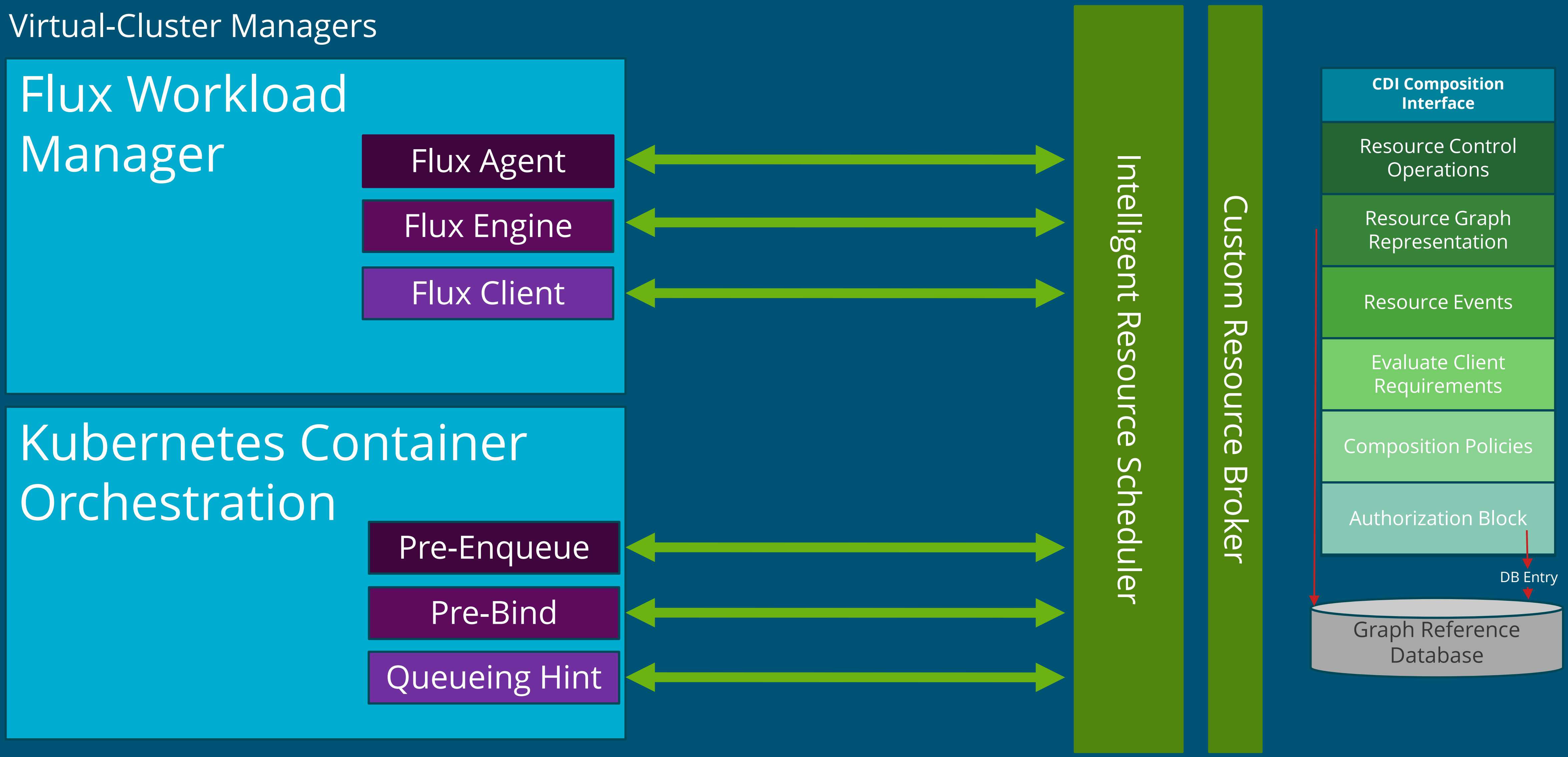
Sunfish Core Services



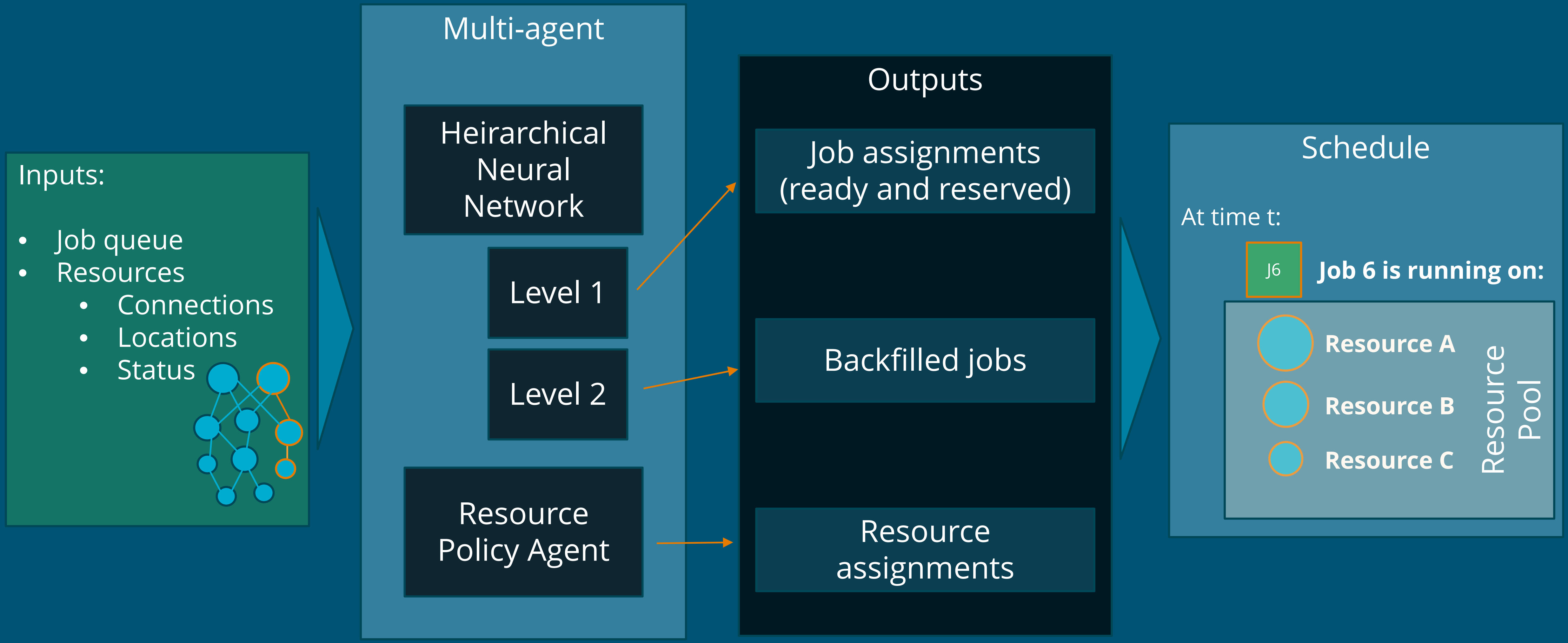
The Sunfish Composability Management Framework



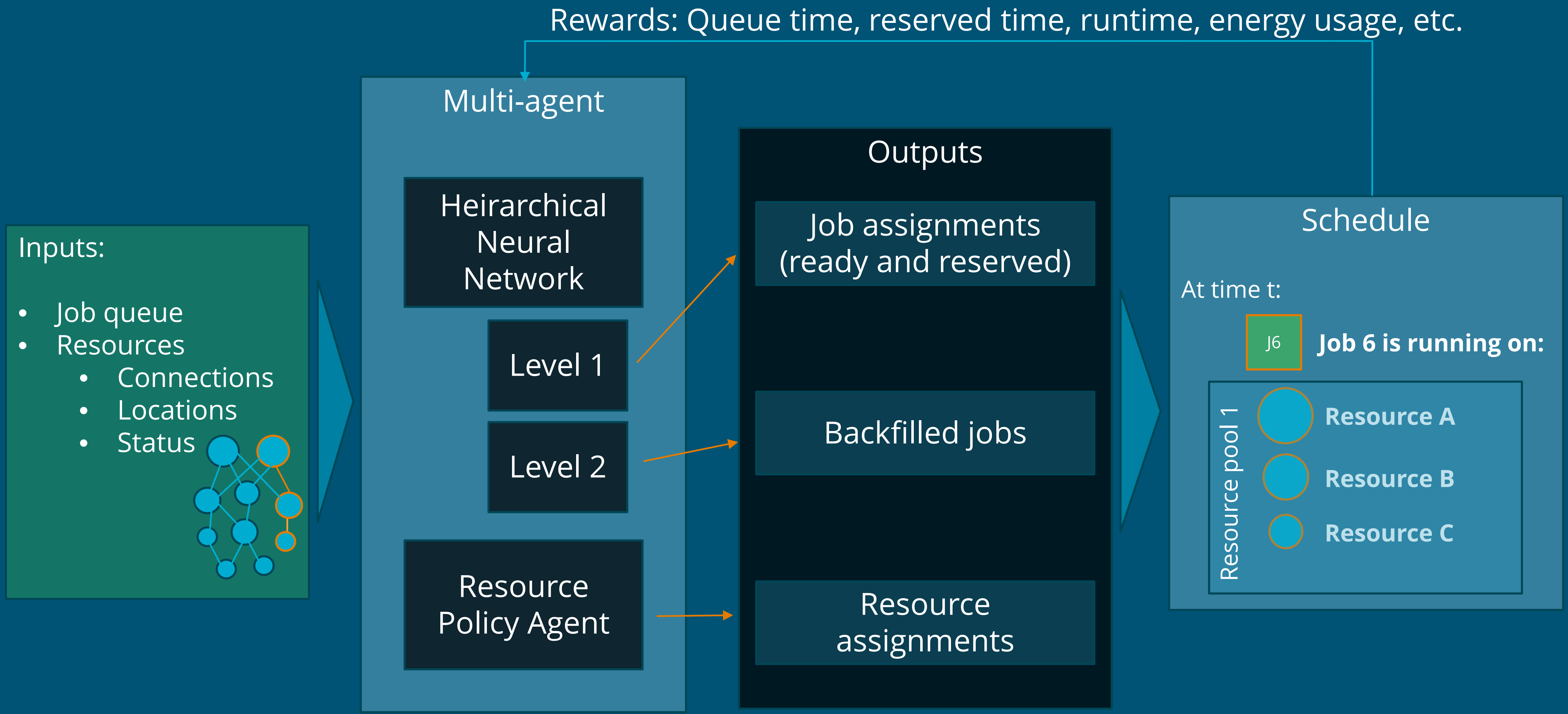
4-Dimensional Software Defined Node Allocation



ML Plugin: Intelligent Resource Scheduler



ML Plugin: Intelligent Resource Scheduler



Why Reinforcement Learning?

Heuristics
- Can be fair, but often aren't the most efficient

Optimization Algorithms
- Must be highly tailored to specific machines

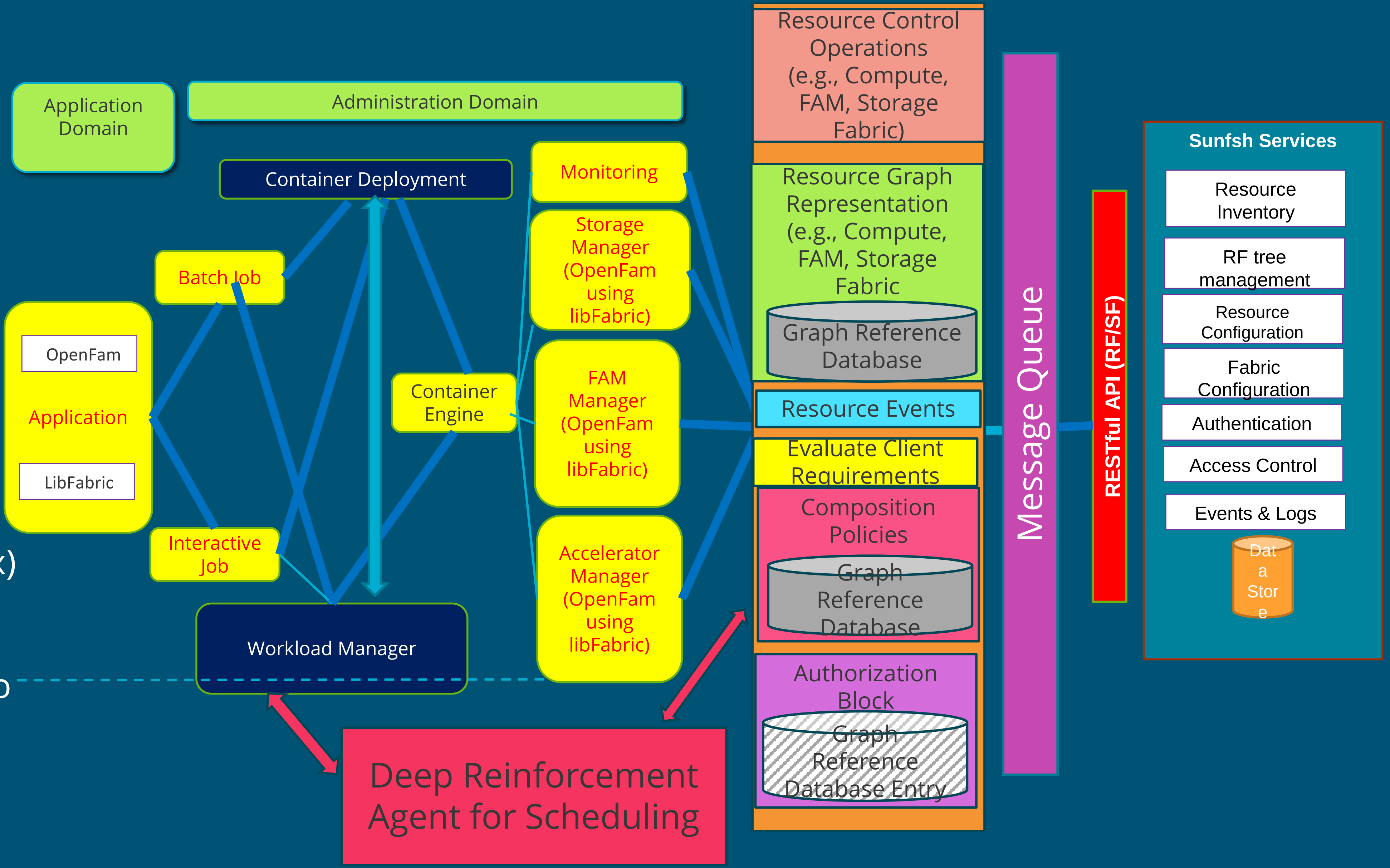
Reinforcement Learning
1. Customized rewards function
   1. Prioritize fairness
   2. Penalize undesirable scheduling
2. Machine agnostic
   1. Adapts to changing resources
   2. Adapts to different traffic volumes
   3. Learns a better algorithm over time
3. Potential cons
   1. Prone to job starvation
   2. May need lots of compute/time

Integrating BeeOND, Sunfish, and the Intelligent Resource Scheduler

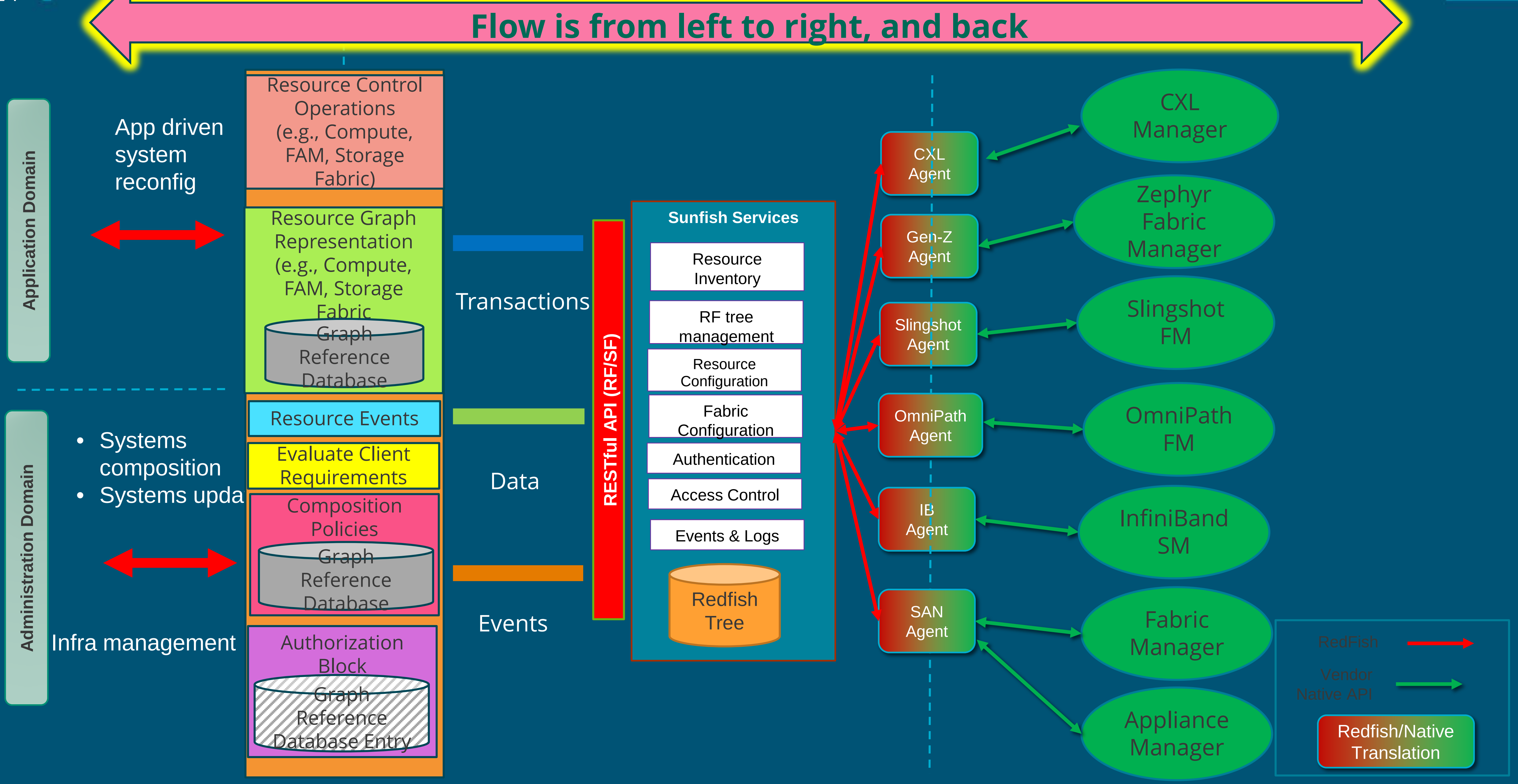
Hardware execution is performed using Sunfish connected hardware Agents

Management of the HPC System is performed by the Sunfish core services.

A Workload Manager (example Slurm or Flux) allocates nodes and requests hardware Resources as a client to the Composability Manager.

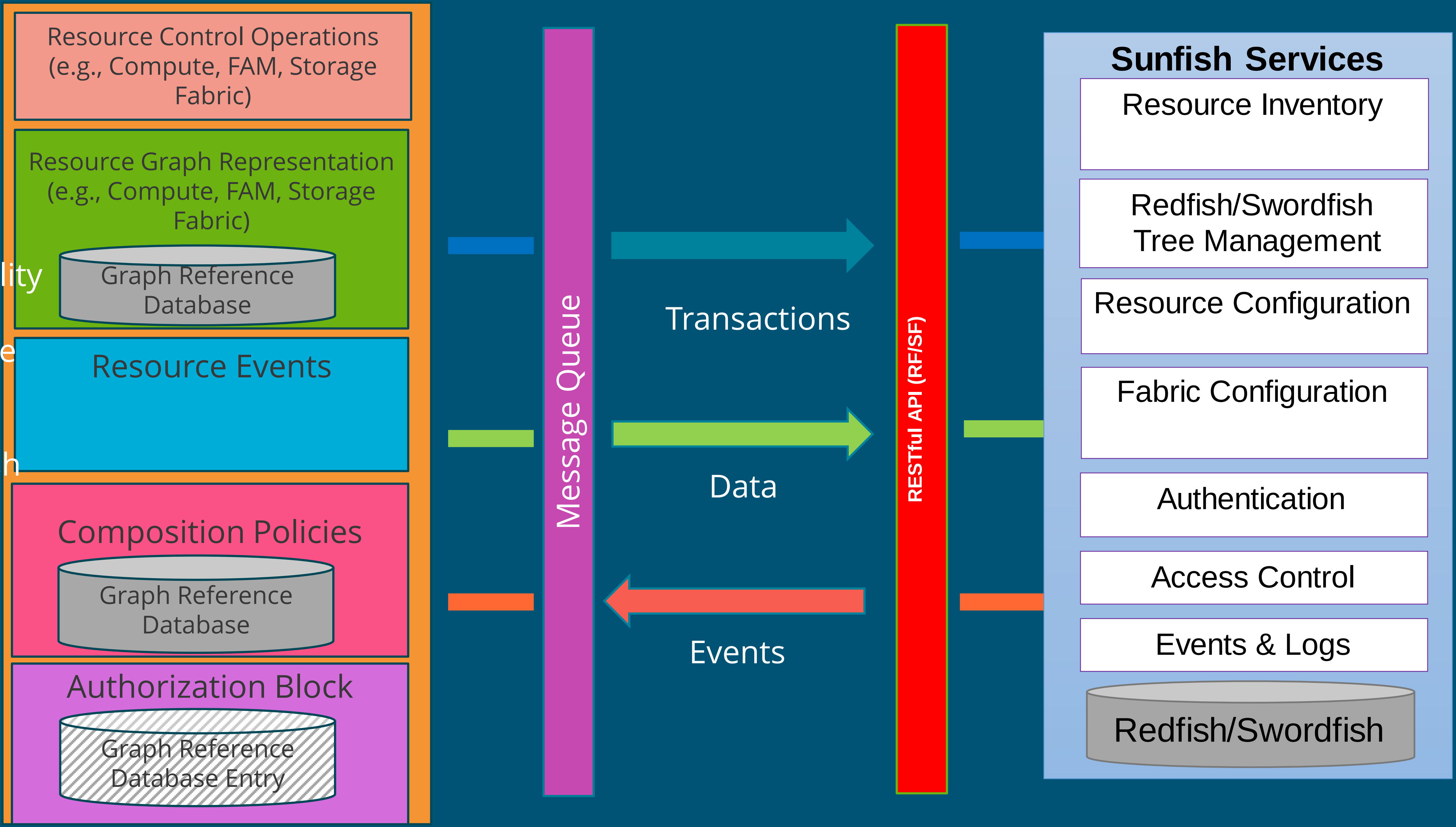


Integrating BeeOND, Sunfish, and the Intelligent Resource Scheduler



Integrating BeeOND, Sunfish, and our ML Algorithm

The Composability Manager requests the hardware Resources from Sunfish



Integrating BeeOND, Sunfish, and our ML Algorithm

Sunfish orders the hardware Agents to aggregate fabric routes and endpoints to fulfill a request for NVMeoF.

Events are generated and propagated back through the Composability Manager to the Workload Manager