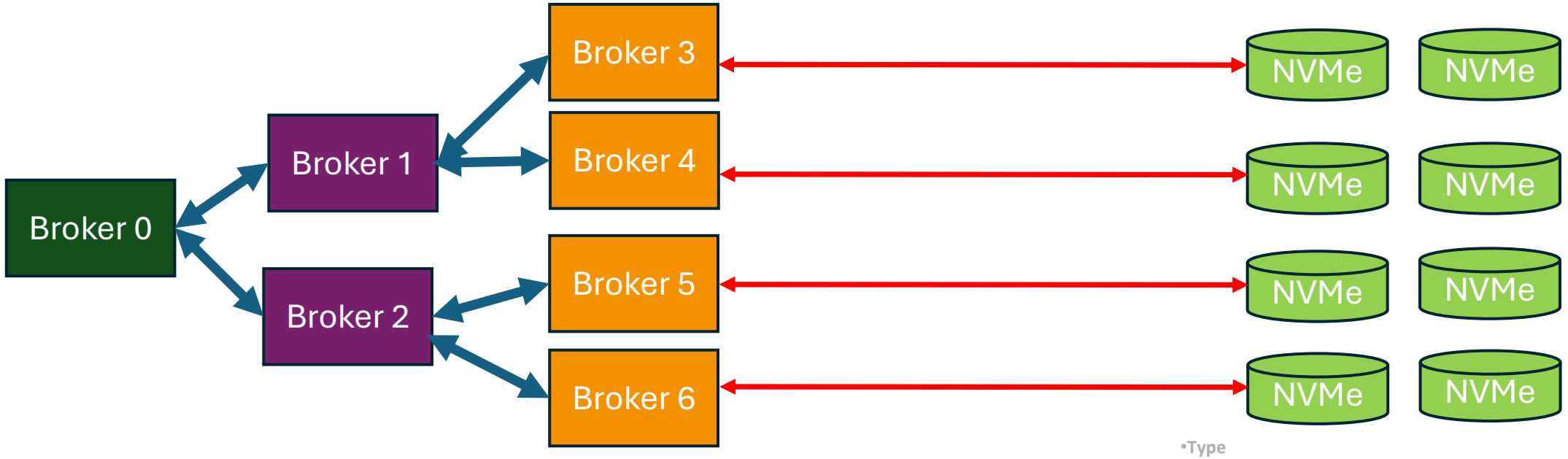
Flux Architecture and Resource Pools

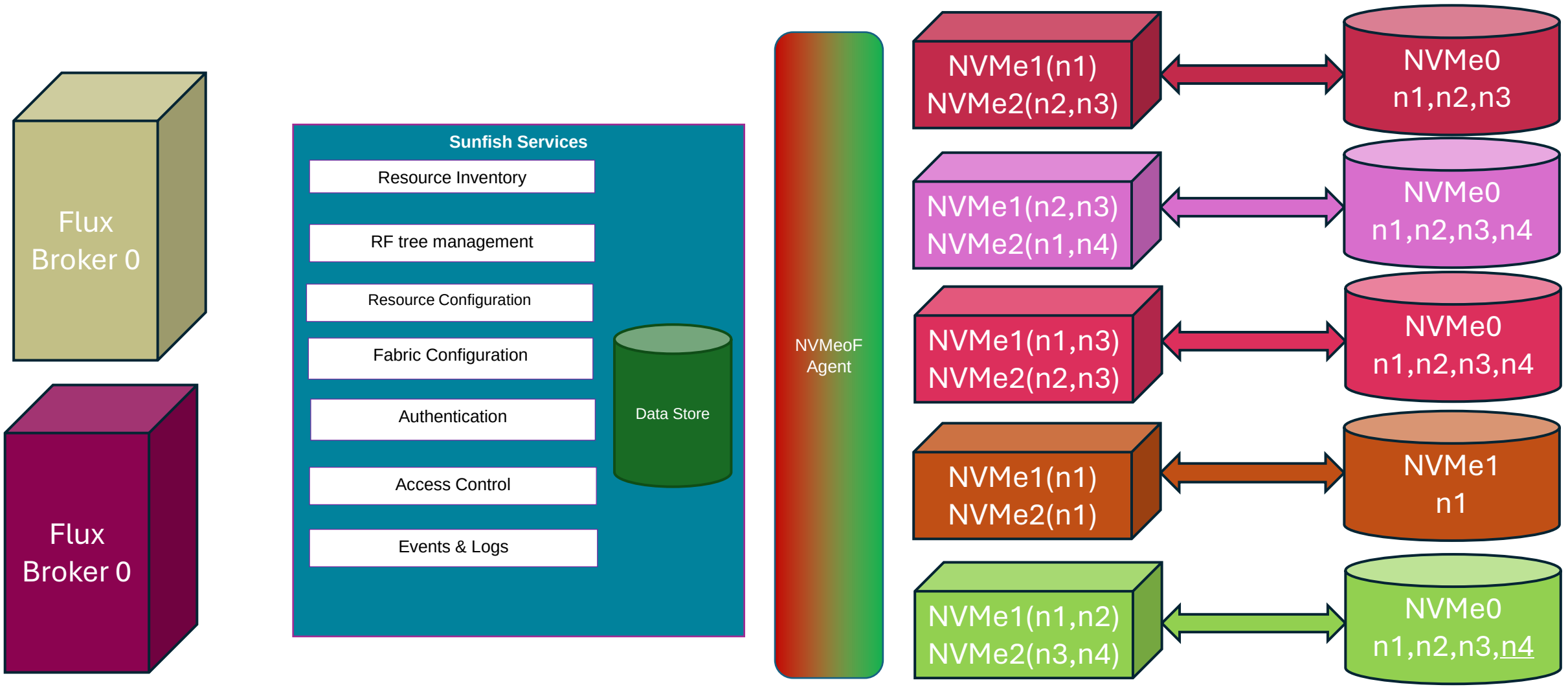


- The concept to describe each identifiable resource is called *resource pool*. A resource pool is a group of one or more *indistinguishable* resources of a same kind.
- When a resource needs to described at coarse granularity, it can be pooled together with other resources of the same type. Conversely, when finer granularity is required, it can be promoted to its own individual pool.

https://flux-framework.readthedocs.io/projects/flux-rfc/en/latest/spec_4.html

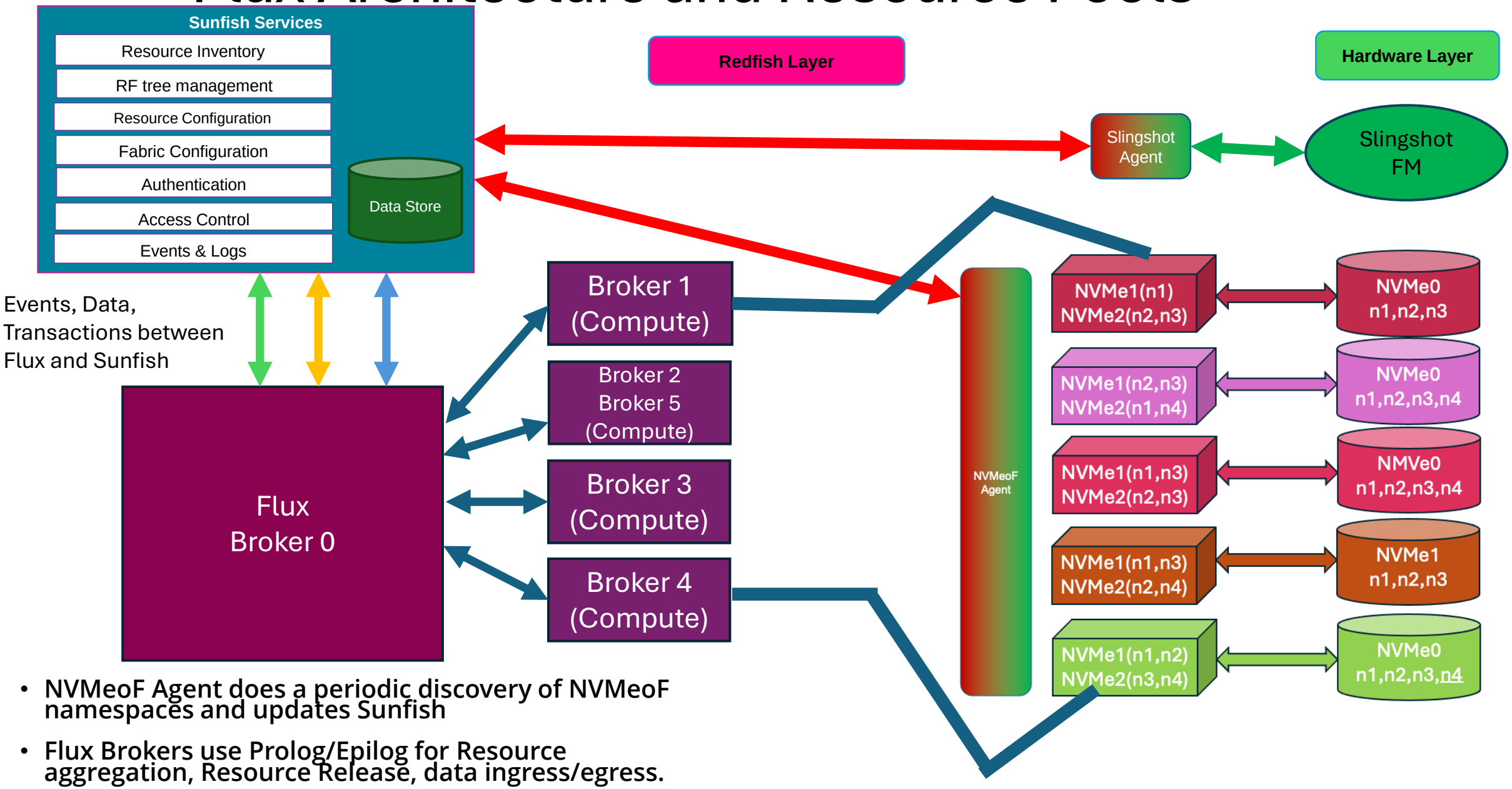
- **Type**
- **UUID** (Unique ID for this resource)
- **Basename**
- **Name**
- **ID** (OPTIONAL numeric ID to be appended to basename to get name)
- **Properties** (static properties associated with this instance)
- **Size** (Total number of resources in this pool)
- **Units** (OPTIONAL units associated with the size value)

NVMeoF/Sunfish Architecture



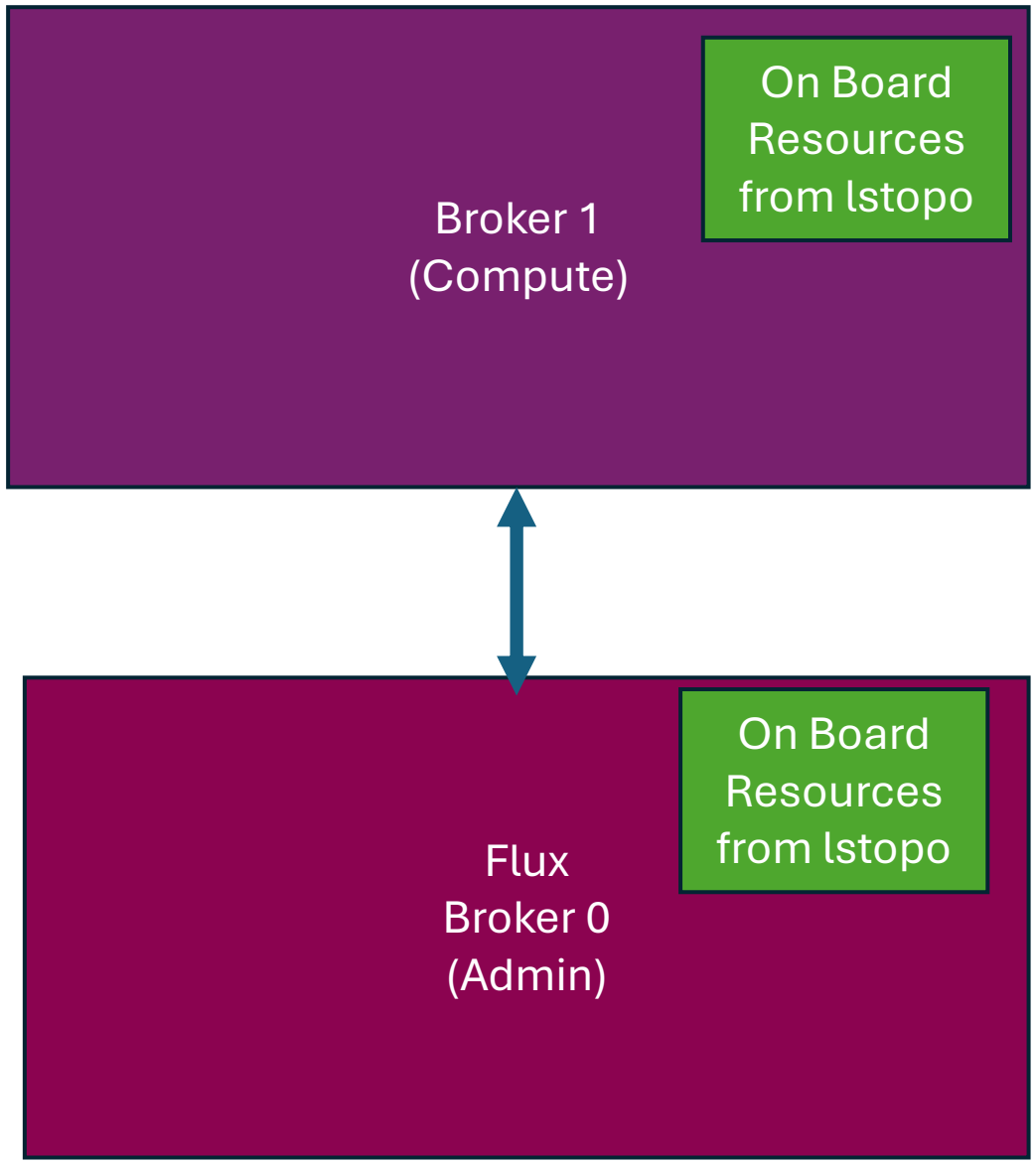


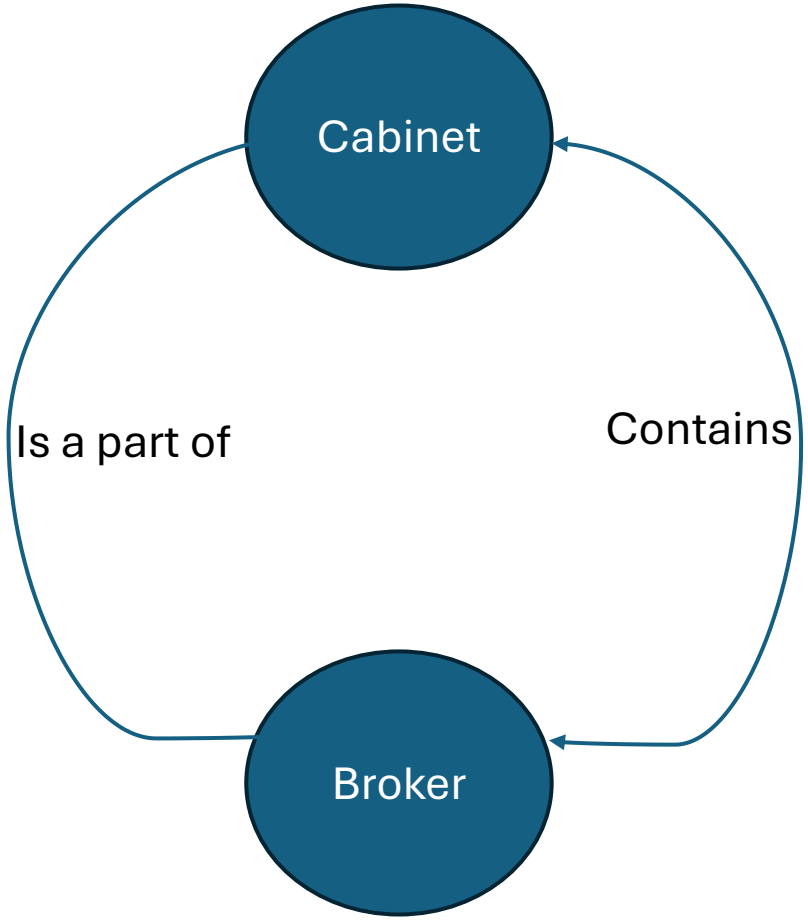

Flux Architecture and Resource Pools



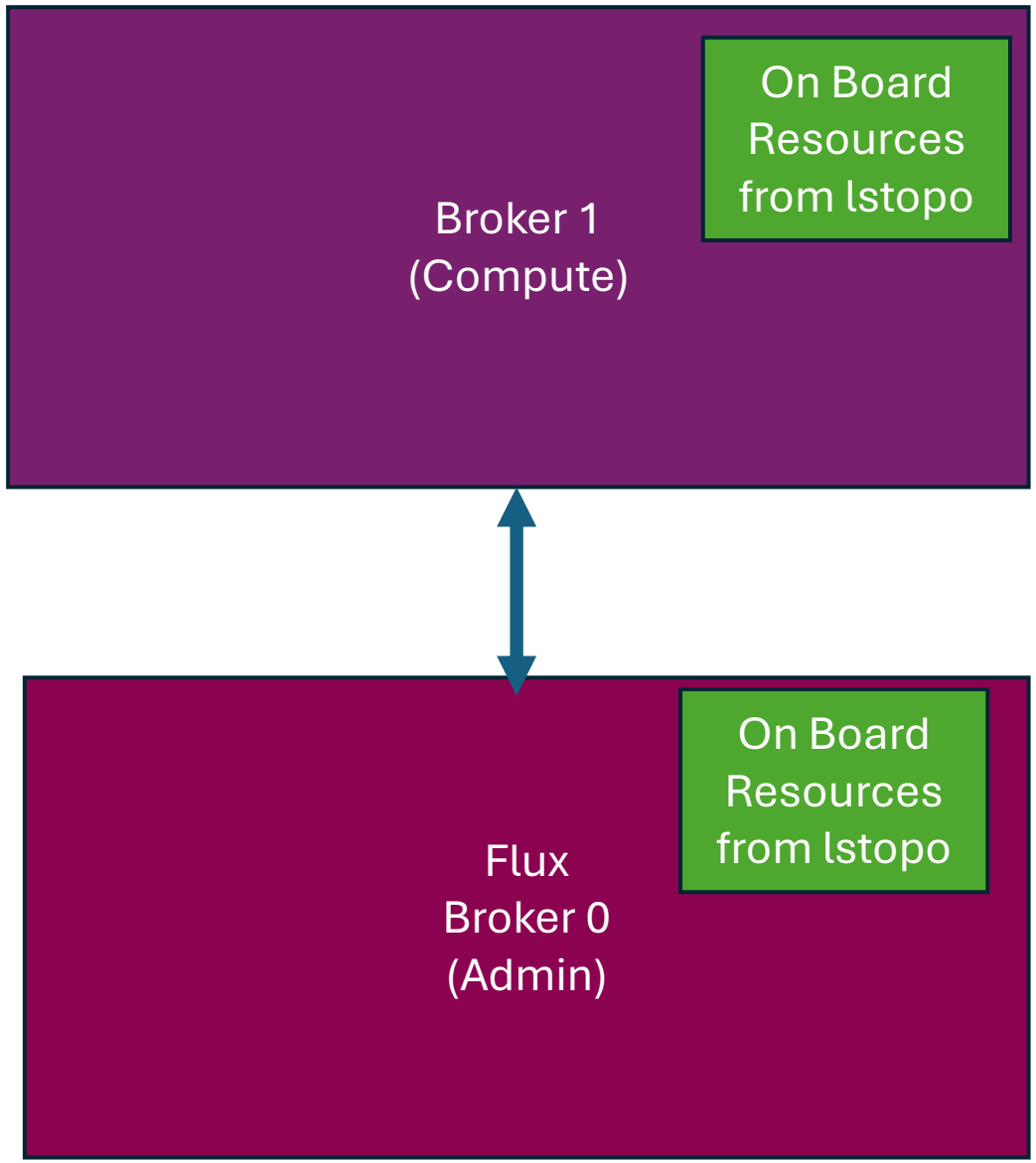
- NVMeoF Agent does a periodic discovery of NVMeoF namespaces and updates Sunfish
- Flux Brokers use Prolog/Epilog for Resource aggregation, Resource Release, data ingress/egress.

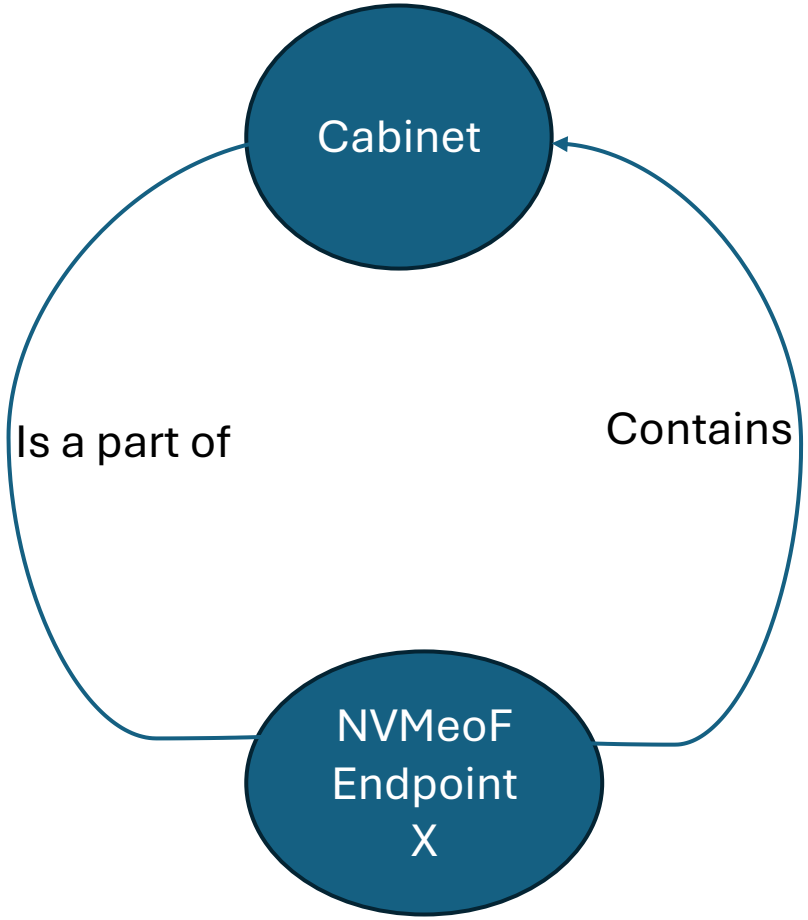
Flux Architecture and Resource Pools NVMeoF



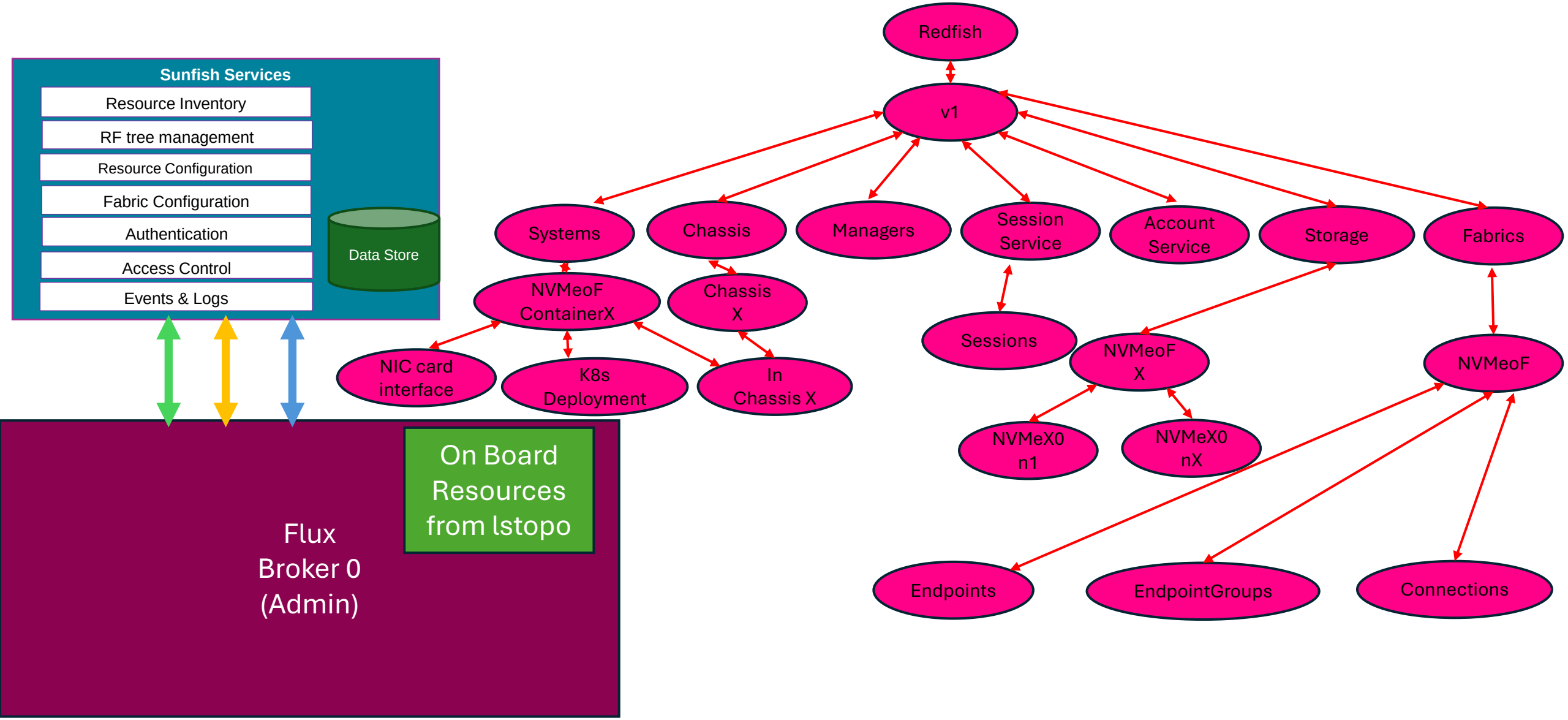


Flux Architecture and Resource Pools NVMeoF





Flux Architecture and Resource Pools NVMeoF



https://redfish.dmtf.org/schemas/DSP2046_2022.2.html#aggregationservice-101

Agent

- **NVMeoF Central Discovery Agent**
- **Sunfish that is geared only to NVMeoF endpoints**
  - Any and all available endpoints of NVMeoF
  - Aggregated components of each NVMe drive and container endpoints
  - Namespaces
  - IP addresses
  - How to access the drive namespaces-NqN (blake14*)*
  - Address of the port that you need to connect to
  - nvme connect -t rdma -n blake14 -a 10.101.46.14 -s 4420
  - Multi-tenant restrictions

Sunfish Agent Discovery and Handoff Walk-Through

- **Assumption is that all of the NVMeoF are up and running—Container start-up**
- **Sunfish Agent acts as a Centralized Discovery Controller**
  - It is querying the NVMeoF endpoints
    - It learns and fills logs:
      - How to access the drive namespaces-NqN (blake14*)*
      - Address of the port that you need to connect to
      - nvme connect -t rdma -n blake14 -a xxx.xxx.xxx.xxx -s 4420
      - Multi-tenant restrictions
    - Send events

Sunfish NVMeoF Agent

- sudo nvme discover -t TRANSPORT -a DISCOVERY_CONTROLLER_ADDRESS -s SERVICE_ID

  Replace *TRANSPORT* with the underlying transport medium: loop, rdma, tcp, or fc. Replace *DISCOVERY_CONTROLLER_ADDRESS* with the address of the discovery controller. For RDMA and TCP, this should be an IPv4 address. Replace *SERVICE_ID* with the transport service ID. If the service is IP based, like RDMA or TCP, service ID specifies the port number. For Fibre Channel, the service ID is not required.
  https://manpages.debian.org/testing/nvme-cli/nvme-discover.1.en.html#:~:text=The%20NVMe-over-Fabrics%20specification%20defines%20the%20concept%20of%20a,which%20it%20can%20connect%20to%20on%20the%20network.

  The NVMe hosts only see the subsystems they are allowed to connect to.

- sudo nvme discover -t tcp -a 10.0.0.3 -s 4420
- nvme discover --transport=tcp --host-traddr=192.168.101.154 --trsvcid=8009

  nvme list-subsysPrints the layout of the multipath devices.
  multipath -ll

Client

- dnf install nvme-cli
- modprobe nvmet-rdma
- modprobe nvmet
- modprobe nvme-rdma
- lsmod | grep nvme
- dnf -y install xfsprogs
- nvme discover -t rdma -a 10.101.46.14 -s 4420
- nvme connect -t rdma -n blake14 -a 10.101.46.14 -s 4420
- lsblk
- dmesg
- nvme list
- nvme **disconnect** -n blake14

Target

- dnf install nvme-cli
- modprobe nvmet
- modprobe nvmet-rdma
- modprobe nvme-rdma
- lsmod | grep nvme
- mkdir /sys/kernel/config/nvmet/subsystems/blake14
- cd /sys/kernel/config/nvmet/subsystems/blake14
- echo 1 > attr_allow_any_host
- mkdir namespaces/10
- cd namespaces/10/
- nvme list
- echo -n /dev/nvme1n1 > device_path
- echo 1 > enable
- mkdir /sys/kernel/config/nvmet/ports/1
- pushd /sys/kernel/config/nvmet/ports/1
- ip addr show
- echo 10.101.46.14 > addr_traddr. ; # IPADDR address for ib0
- echo rdma > addr_trtype
- echo 4420 > addr_trsvcid
- echo ipv4 > addr_adrfam
- ln -s /sys/kernel/config/nvmet/subsystems/blake14 /sys/kernel/config/nvmet/ports/1/subsystems/blake14
- dmesg | grep "enabling port"
- exit

- Assumptions
  - Starting at time 0
  - Static endpoints
  - Static NVMeoF resources
  - All endpoints are already online
  - Fabric endpoints have an IP address
  - Flux Brokers have static endpoints included in the /etc/flux/resource folder---they stay drained, initially

Sunfish Library
Sunfish Agent Server
Read the Configuration file for endpoints
For each available endpoint, then retrieve Discover Log Page and register for keep alive timeout
Populate the Redfish Registry
Register Agent with Sunfish Server
For all endpoints {
Crawl through endpoint registry
Send event to Sunfish Server
Push discovery log page }

- Assumptions
  - Agents periodically poll endpoints

Sunfish NVMeoF Agent

Keep alive from each endpoint
If keep alive is missing for endpoint {
Update registry,
Send Event to Sunfish server,
Update Sunfish server}

- Assumptions
  - Resources in Flux Broker 0 are listed as name-jbod, name-host-jbod, name-endpoint
  - Resources remain drained in Flux until Sunfish signals that the resources are active
  - HPC headnode can turn on resources via Redfish commands
  - Communication between Sunfish Server and Flux Broker 0 is done via Events and JSON.

Sunfish Server

Sunfish Server registers with Flux Broker 0
Sunfish Server populates Flux Broker 0 dynamic resource list
Flux NVMeoF resources are listed as Active is Sunfish has the resource available
Flux NVMeoF resources are drained, initially
Flux Broker 0 handles the CDI aggregation

- https://www.ibm.com/docs/en/storage-ceph/7?topic=target-defining-nvme-subsystem
- https://www.ibm.com/docs/en/storage-ceph/7?topic=target-defining-nvme-subsystem