|  |  |
| --- | --- |
| Use Case Description | Kubernetes Multi-node Deployment (interactive mode application launch) |
| Actors | Fabric Manager, Administrator, Master Node, Worker nodes, Ethernet switches |
| Description | Create template and deploy multiple K8s Pods on multiple nodes |
| Initial State | * Master Node
	+ Running K8s Services
	+ Running etcd data base mgr
* Worker Nodes
	+ Running Kubelet service
	+ Running container runtime
		- Docker, rkt, runc
* FM
	+ Running Redfish services
	+ Running CNI daemon
	+ Clusters configured, CIDR blocks (IP address pools) assigned
* Ethernet Network
	+ ***Master Node and Worker Nodes connected (cluster already created)***
 |
| Normal Flow | * Admin: Create YAML Deployment and Services files describing micro-service containers, Pods, replications, and connections (internal and external)
* Admin: invoke ‘Kubectl apply ‘on Master node with YAML files as arg’s
* MN: Parse YAML, update etcd data base, select worker nodes
* MN: launch appropriate numbers of Pods on target worker nodes via kubelet
* WN: extract Pod descriptions from etcd data base
* MN: assign each Pod an Ethernet namespace, construct port connections according to YAML template
* WN: invoke CNI plugin, which will contact FM CNI daemon and obtain IP address for Pod
* WN: update etcd data base with Pod’s IP address, query IP address for other Pods in this deployment
* MN: parse YAML Services file and establish external IP address through which the deployment converses with clients
* MN: update etcd data base to include Services IP
* ??: Set up event monitoring for Deployment & the Service
* MN: monitor etcd data base status of deployment, adjust resource allocations as necessary
* Admin: invoke ‘Kubectl destroy‘ on Master node with YAML files as arg’s
* All: tear down connections, shut down processes on worker nodes, update etcd data base

Missing: * Security key management
* Authentication and authorization steps
 |
| Alternate Flow 1 | * Admin: Create YAML Deployment and Services files describing micro-service containers, Pods, replications, and connections
* Admin: invoke ‘Kubectl apply ‘on Master node with YAML files as arg’s
* MN: Parse YAML, update etcd data base, select worker nodes
* MN: launch appropriate numbers of Pods on target worker nodes via kubelet
* WN: extract Pod descriptions from etcd data base
* MN: assign each Pod an Ethernet namespace, construct port connections according to YAML template
* WN: invoke CNI plugin, which will contact FM CNI daemon and obtain IP address for Pod
* WN: update etcd data base with Pod’s IP address, query IP address for other Pods in this deployment
* MN: parse YAML Services file and establish external IP address through which the deployment converses with clients
* MN: update etcd data base to include Services IP
* ??: Set up event monitoring for Deployment & the Service
* MN: monitor etcd data base status of deployment, adjust resource allocations as necessary
* MN: Error – K8s controller cannot match Status of executing Pods to desired deployment because containing cluster has insufficient resources
* MN: ??? -need some mechanism to increase cluster resources, still researching how K8s might already handle this
 |

## Create a cluster

|  |  |
| --- | --- |
| Use Case Description | Create a K8s cluster within a composable DC fabric |
| Actors | Fabric Manager, Resource manager, Composer, Administrator, Master Node, Worker nodes, Ethernet switches |
| Description | Instantiate a K8s cluster designed to run ML Ops as a priority |
| Initial State | * Diverse free pools of compute, memory, GPU, HSN, and storage resources are in power savings mode (offline)
* Diverse pools of compute, memory, GPU, HSN, and storage resources are available in existing clusters currently in service (online)
* Ethernet and online high speed Networks are running
* Other virtual clusters (K8s and others) running on the ‘online’ machines
 |
| Normal Flow | Composing Manager:* Parse the cluster requirements
* Find potential cluster elements
	+ Consult Resource Managers for candidate elements
		- RMs obtain inventory from various OFMFs directly or through aggregators, probably ahead of time
		- RMs responsible for tracking logical resources
	+ Note: We don’t have a ‘logical resource model’ for memory in Redfish.
		- How do we represent an aggregated memory ‘object’?
		- MPI and shmem libraries will need to malloc shared data objects from FAM, not private DIMMs.
	+ Note: We don’t have a fabric agnostic method specified for hosts to share (map) local resources to the fabric.
* Craft potential cluster implementations
	+ Validate feasibility
	+ Extract specific attributes about the configuration
	+ Analyze acceptability
	+ Iterate until happy
* Create the cluster using the best candidate template
	+ Lock down the endpoint resources via RMs
	+ Create the fabric zone via the OFMF
* Activate the cluster
	+ Create required connections for cluster administration
	+ Launch cluster master (Master Node) and Worker Nodes
* MN and WNs create RPC services to communicate
	+ (Ethernet based?) RPC comms created
* Master Node creates etcd data base in shared storage
	+ MN maintains consistency via RAFT algorithm, which uses RPC communications

MN gives WNs access to etcd |
| Alternate Flow 1 | *
 |

## Create a zone

|  |  |
| --- | --- |
| Use Case Description | Create a zone to host a K8s cluster within a composable DC fabric |
| Actors | Fabric Manager, Resource manager, Composer, Administrator,  |
| Description | Use Redfish ‘zone’ object to define a virtual, private network within the larger fabric  |
| Initial State | * Diverse free pools of compute, memory, GPU, HSN, and storage resources are in power savings mode (offline)
* Diverse pools of compute, memory, GPU, HSN, and storage resources are available in existing clusters currently in service (online)
* Ethernet and online high speed Networks are running
* Other virtual clusters (K8s and others) running on the ‘online’ machines
* List of cluster members defined. Resources reserved by Composing Manager
 |
| Normal Flow | Composing Manager:* Parse the list of cluster members
* Query cluster member endpoints for membership in existing ‘zones’
	+ Validate isolation
	+ What about multi-zone membership?
* Create a Redfish fabric zone object based on Redfish schema
	+ Do we need an OFMF utility to do this?
	+ Fill in the endpoints (resources) to be contained in the zone
	+ How do we indicate address pool restrictions?
	+ Do we indicate an optional ‘make symmetric connections’ task at the same call? <not needed if endpoint groups>
* Post the Redfish zone object to the OFMF’s resource tree

OFMF:* OFMF: parse the HTTP request and post a new zone
* OFMF: calculate the new route table entries and patch appropriate switch or router table entries, if enabled
	+ policy may not enable routes until connections are enabled
* OFMF: update (patch) appropriate endpoint objects
* OFMF: update hardware as appropriate
* OFMF: respond to client with success
 |
| Alternate Flow 1 | *
 |