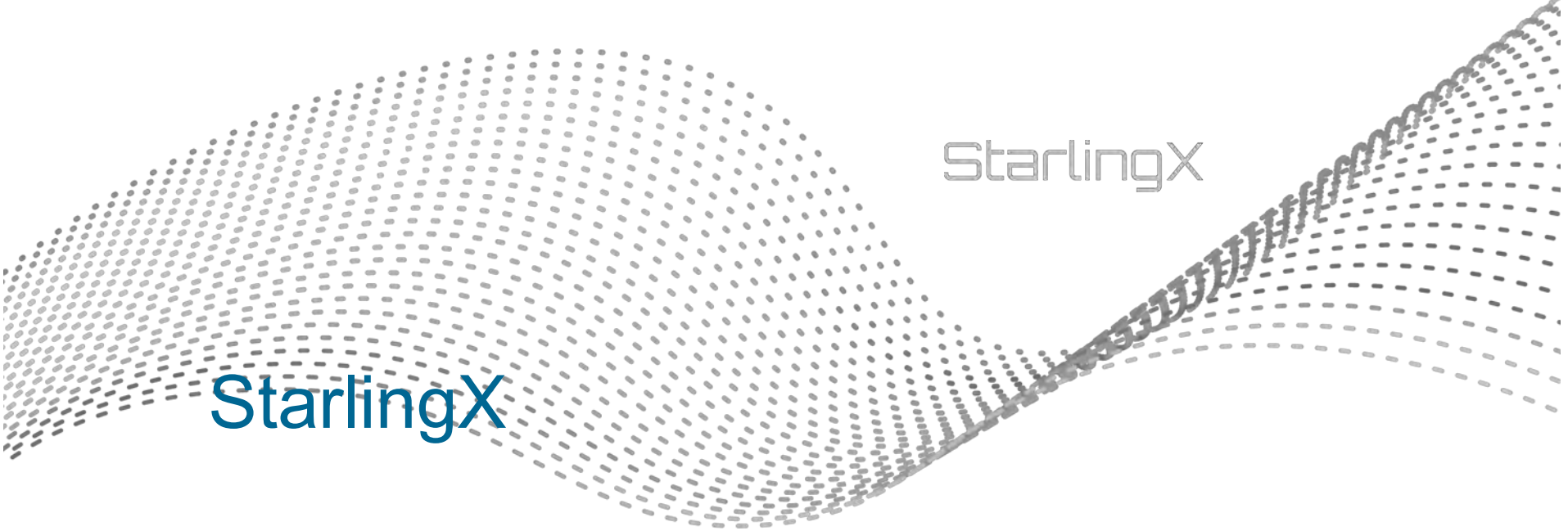


StarlingX

StarlingX

The StarlingX Project provides a fully integrated openstack platform with focus on High Availability, Quality of Service, Performance and Low Latency needed for industrial and telco use cases.

StarlingX is aligned with the OpenStack Foundation Edge Working Group and the Linux Foundation Akraino Edge Stack.

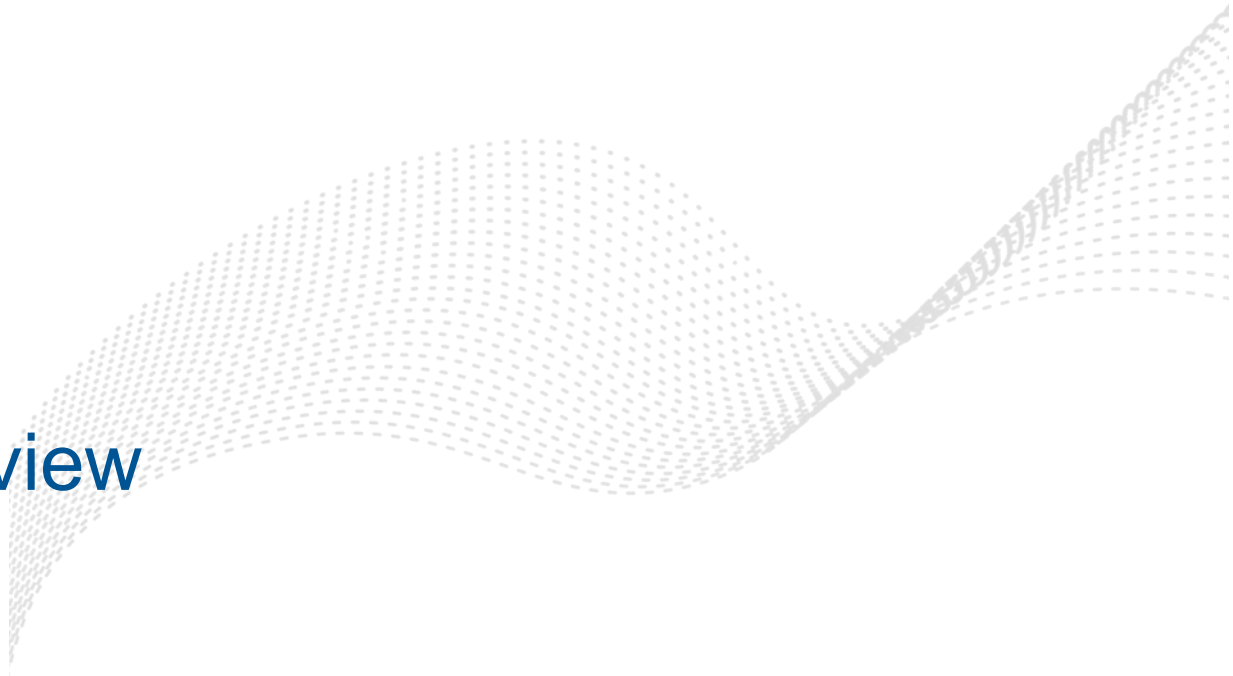
StarlingX – Getting Started Docs

- Guiding Values
- Overview
- Main Components
- Installation Overview
- How to build the StarlingX installation ISO,
- How to install StarlingX,
- Incubation Projects
- In-flight StarlingX Evolution

StarlingX – Guiding Values

- High Availability and Robust Solutions,
- High Performance & Low Latency,
- Highly Secure Solutions,
- Scale from Single Server Deployments to 100+ Server Deployments,
- Ensure Small Footprint of StarlingX infrastructure,
- Meet all Edge Computing Requirements,
- Simplicity & Usability.

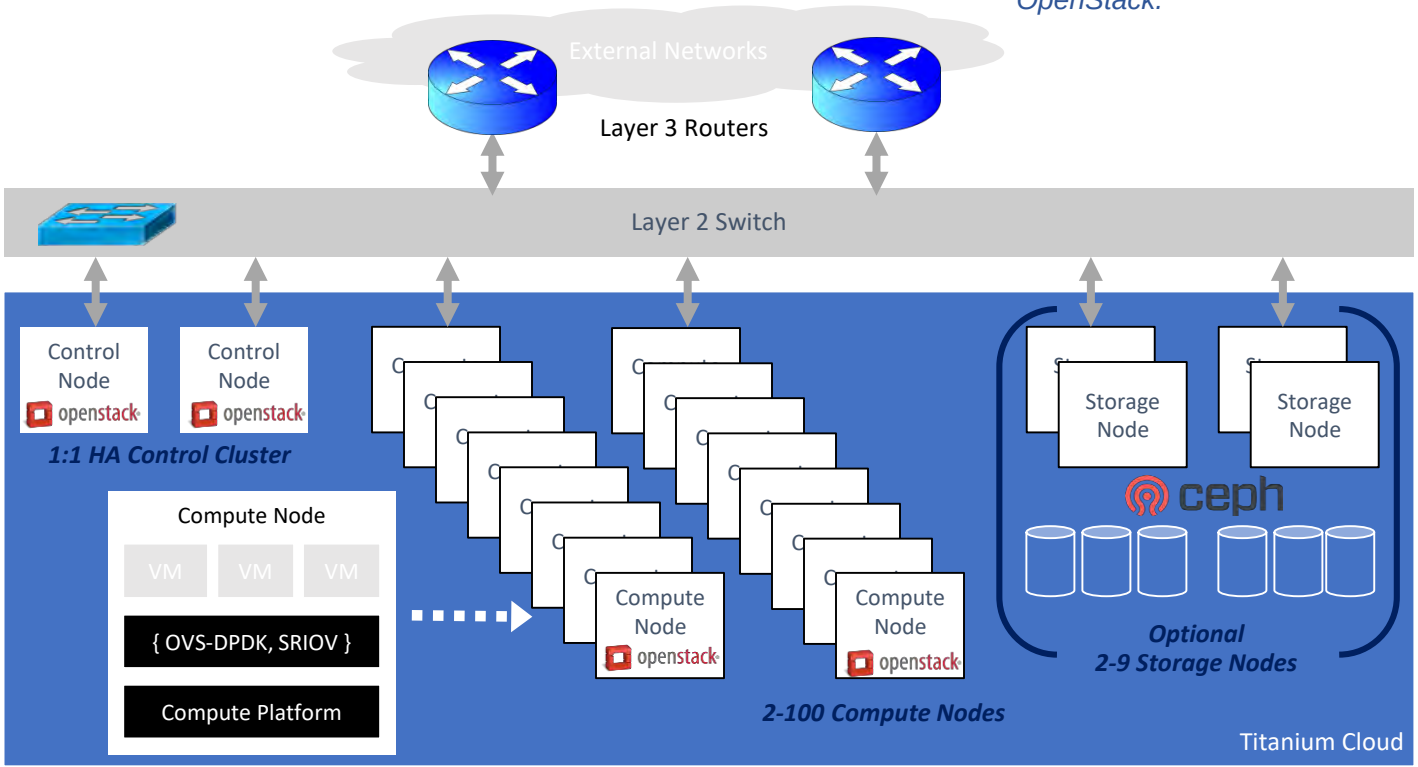
Overview



StarlingX – Physical Solution Architecture

Standard Configuration

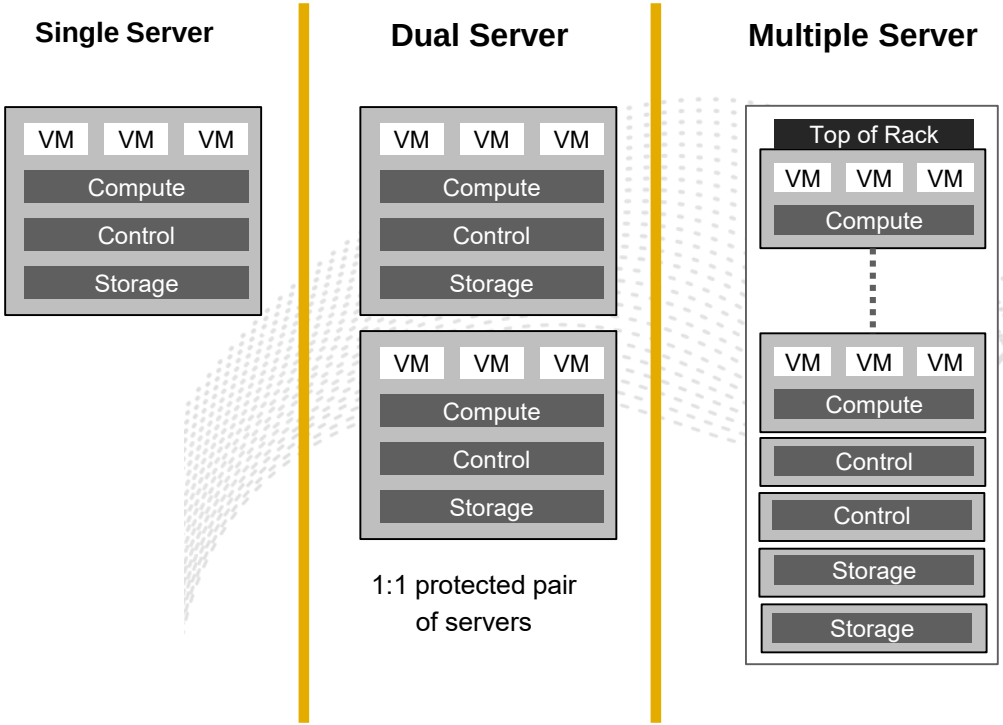
StarlingX provides a bare metal deployment of OpenStack.



- StarlingX's 'Standard Configuration consists of:
  - 2x Node HA Controller Node Cluster
  - 2-100x Compute Node Cluster
    (Note: up to 100 only with CEPH Storage Node Cluster)
  - OPTIONAL 2-9 Node CEPH Storage Node Cluster
- StarlingX Service Manager provides Cluster Management for HA Control Node Cluster
- Neutron's L3 Networking Services are distributed across Compute Nodes,
- Cinder backend can be:
  - LVM on Controllers (DRBD sync'd for HA), and/or
  - On CEPH Storage Nodes.
- Glance backend can be:
  - Filesystem on Controllers (DRBD sync'd for HA), and/or
  - On CEPH Storage Nodes
- Swift backend is only supported on CEPH Storage Nodes,
- CEPH Storage Nodes can be deployed in 'replication groups'
  - Groups of 2x Storage Nodes, for replication factor of 2, or
  - Groups of 3x Storage Nodes for replication factor of 3.

StarlingX – Scaling Small or Scaling Large

Scalability for the Edge

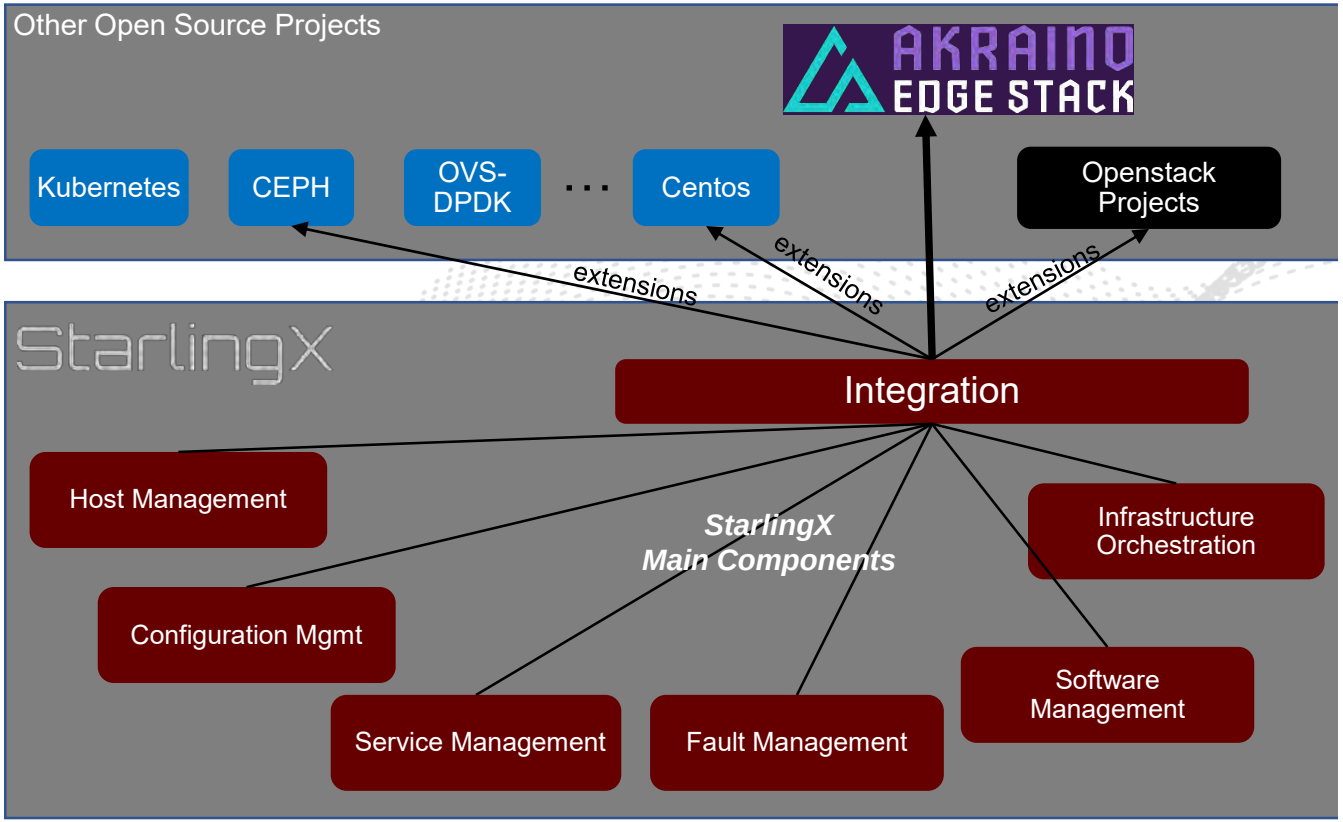


- StarlingX solution scales both small and large,
- Single Server
  - Single server,
  - Running Controller, Compute, Networking and Storage functions,
  - Supported on small hardware form factor (e.g. 8-core XeonD boards)
- Dual Server
  - Two Servers,
  - Each Server running Controller, Compute, Networking and Storage functions,
  - Controller, Networking and Storage functions run HA across two servers,
  - Supported on small hardware form factor (e.g. 8-core XeonD boards)
- Multiple Server
  - Controller, Compute and Storage functions all running on independent servers,
  - 2x Node HA Controller Node Cluster
  - 2-100x Compute Node Cluster
  - OPTIONAL 2-9 Node CEPH Storage Node Cluster
  - And L3 Networking services distributed across compute nodes.

Physical Server

StarlingX – Main Components

And Interworkings w/other Open Source Projects



- StarlingX consists of 6x main components for managing the StarlingX Bare Metal Deployment of OpenStack:
  - Configuration Management,
  - Host Management,
  - Service Management,
  - Software Management,
  - Fault Management, and
  - Infrastructure Management.

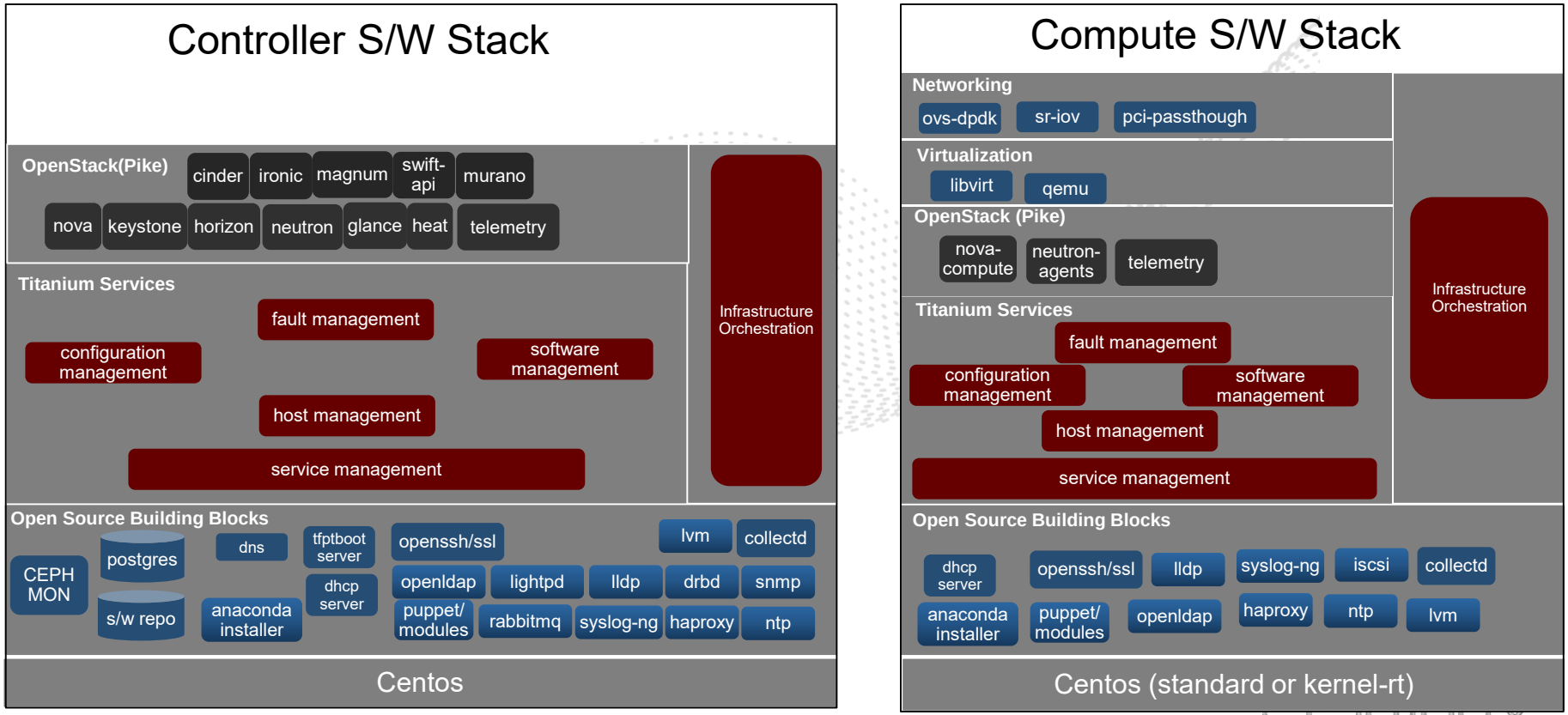
  (explained in detail in following slides)

- StarlingX Solution also consists of extensions to Integrated Open Source Projects (e.g. CEPH, CentOS, OpenStack)
  - Some of these have been contributed back into the Open Source Projects,
  - Some of these are currently supported within StarlingX, outside of the source Open Source Project.

  HOWEVER, as part of the open-sourcing of StarlingX, the intent is to contribute ALL of these extensions back into the source OpenSource Project ... or remove the extension.

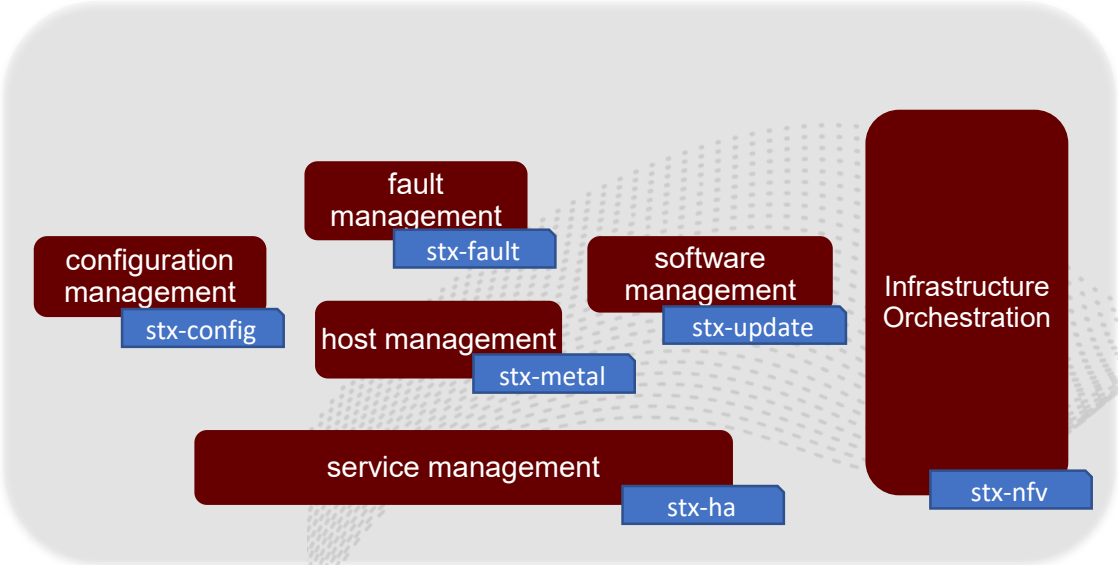
- StarlingX Solution implements a core blueprint / use case within the Akraino Edge Stack Open Source Project.

StarlingX – Controller/Compute S/W Stacks



Main Components

StarlingX – Main Components and Repos



- Configuration Management
  - Provides Host Installation, Inventory Discovery and Host Configuration, *(NOTE will move to Host Management in future)*
  - Provides System-level Configuration and Configuration of StarlingX Platform Services
  - Provides API, Horizon and CLI services for all Main Components.
- Host Management
  - Manages lifecycle of Host; operational and administrative,
  - Provides Host fault monitoring, alarming and initiates fault recovery,
- Service Management
  - High Availability Cluster Management for StarlingX and OpenStack Services on Controllers,
- Software Management
  - Software Patch (bug fix) management and deployment,
  - Software Release Upgrade management.
- Infrastructure Orchestration
  - High Availability management of VMs,
  - Software Patching & Upgrade Orchestrator Services.
- Fault Management
  - Alarm and Log Reporting Services for other StarlingX Components.

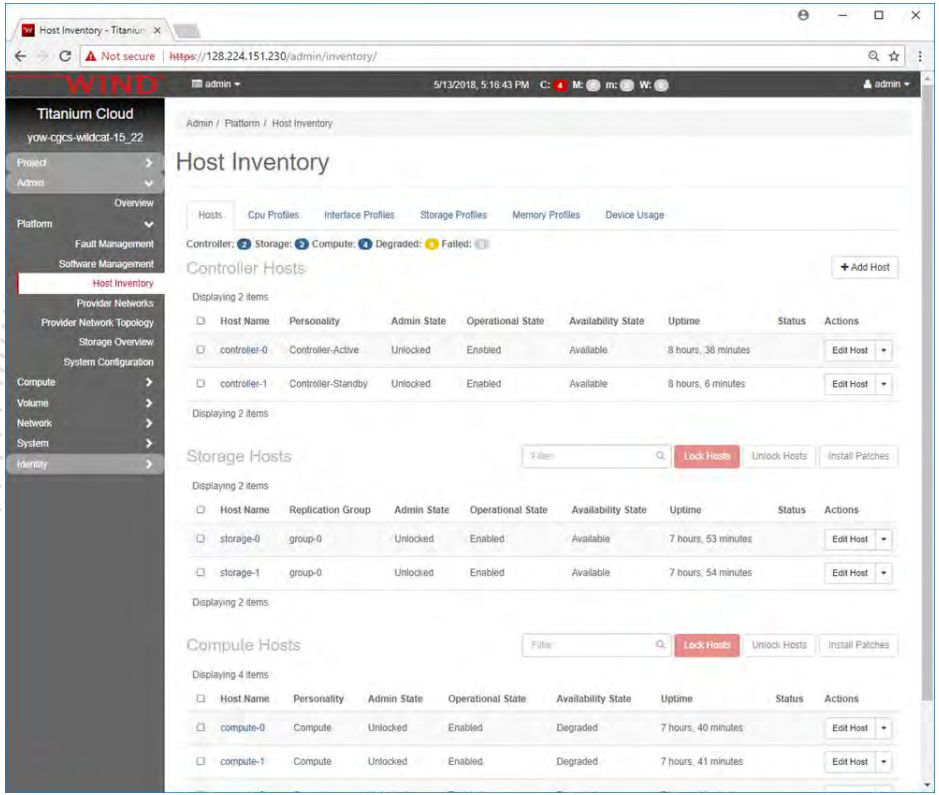
REPO: stx-config

StarlingX – Configuration Management

Inventory and Nodal Configuration

(NOTE: will move to Host Management in future)

- Installation
  - Auto-discover new nodes
  - Manage installation parameters (i.e. console, root disks)
  - Bulk provisioning of nodes through xml file
  - Installation progress indicators
- Inventory Discovery
  - CPU/cores, SMT, processors, memory, huge pages
  - Storage, ports
  - GPUs, crypto/compression H/W, LLDP neighbor info
- Nodal Configuration
  - Node role, role profiles
  - Core, memory (including huge page) assignments
  - Network Interfaces and storage assignments
  - Bulk configuration of nodes via system profiles
- User Interface
  - REST API, Horizon GUI and “system” CLI command-suite

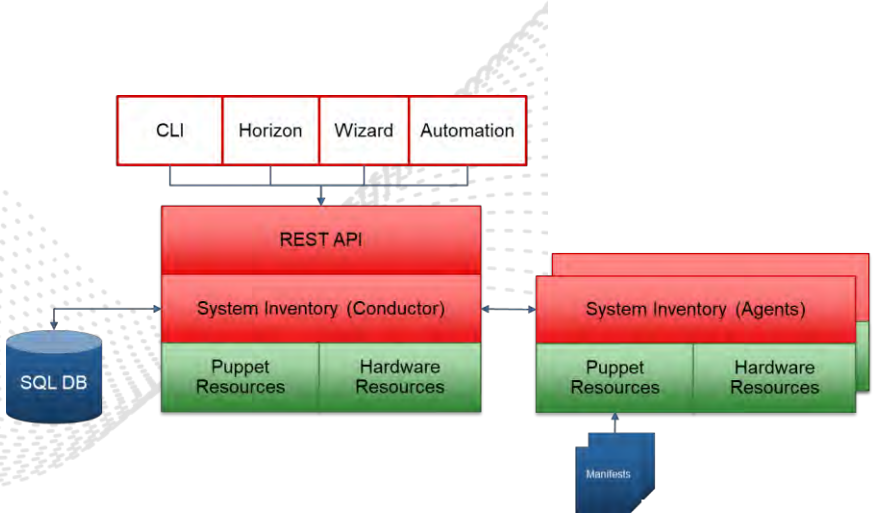


REPO: stx-config

StarlingX – Configuration Management

Puppet-based Configuration

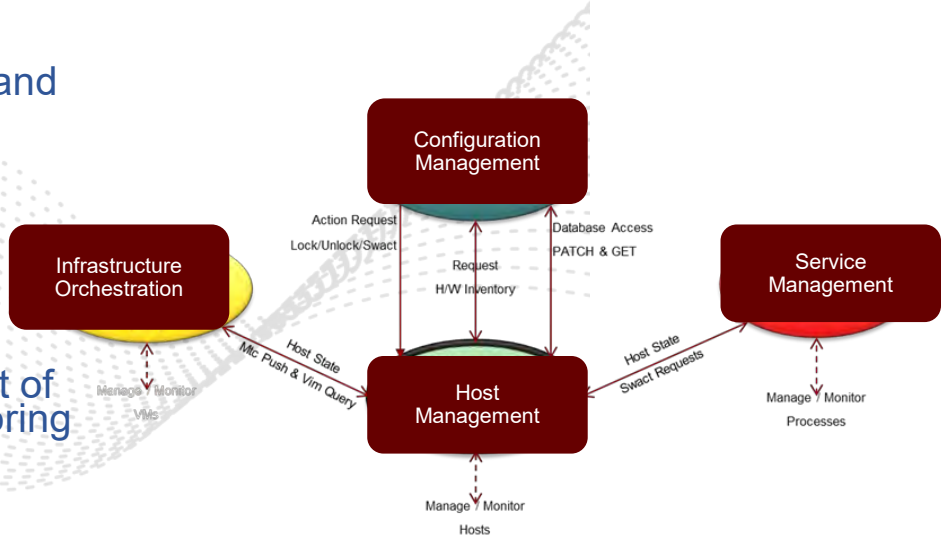
- System configuration files and system setup managed by puppet
- Hosts are configured according to the puppet hiera yaml data files which are either static defaults or populated by system inventory based on the system configuration data
- System Inventory is responsible for managing the puppet hiera yaml files and will regenerate them during configuration operations
- Configuration changes supported post initial deployment



REPO: stx-metal

StarlingX – Host Management

- Manages the life cycle of the host
- Detects and automatically handles host failures and initiates recovery
- Monitor & Alarm & Recover
  - Cluster connectivity, critical process failures
  - Resource utilization thresholds, interface states
  - H/W fault / sensors, host watchdog
  - Activity progress reporting (e.g. booting, testing)
- Interfaces with board management (BMC) for out of band reset, power-on/off and H/W sensor monitoring
- Publishes host state with other StarlingX Components,
- Exposes a set of administrative commands for the user to manage the host via REST API
  - Lock/unlock, reboot, reset, re-install, SWACT
  - Out-of-Band via BMC: Reboot, reset, power on/off

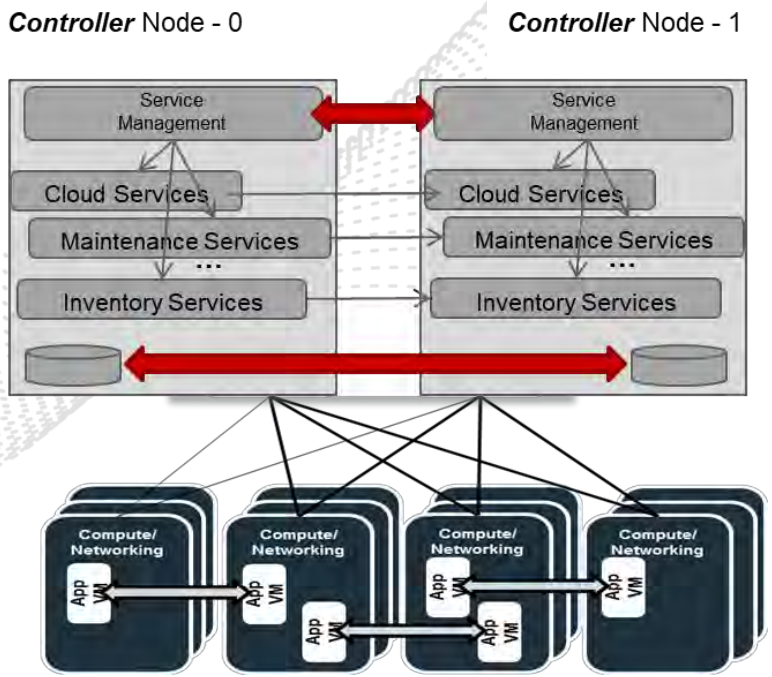


Vendor Neutral Host Management

REPO: stx-ha

StarlingX – Service Management

- High availability manager to manage the life cycle of services where the redundancy model can be N+M or N across multiple nodes
  - Currently used in StarlingX to provide 1+1 HA Controller Cluster,
- Configured to use multiple messaging paths to avoid split-brain communication failures
  - Up to 3 independent communication paths
  - LAG can also be configured for multi-link protection of each path
  - Messages are authenticated using HMAC SHA-512 if configured / enabled on an interface-by-interface basis
- Active or passive monitoring of services
- Allows for specifying the impact of a service failure
- Completely data driven through the configuration of a SQL database



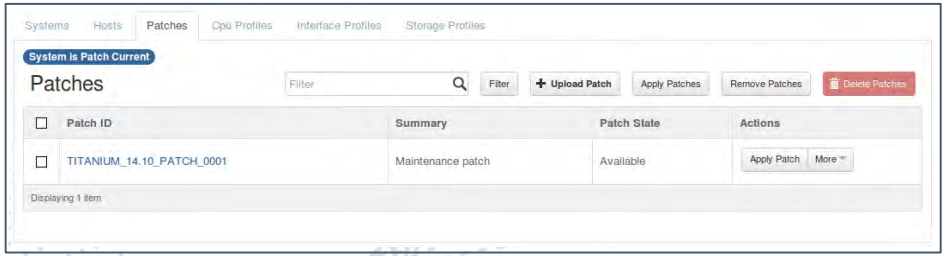
High Availability for Critical Infrastructure

REPO: stx-update

StarlingX – Software Management

Software Patching

- Provides the ability to deploy software updates for corrective content and/or new functionality
- Consistent mechanism for patching all layers of the infrastructure stack -- kernel all the way up to the openstack services layer
- Rolling update strategy across nodes
  - Nodes can be updated in parallel when constraints are not violated
- In-service and reboot required patches supported
  - Reboot required for kernel replacement etc.
  - For patches that require a reboot, VM's are live migrated off of node
- Comprehensive patch status at node and system level
- REST API, Horizon GUI and "sw-patch" CLI command-suite



Hitless Software Patching

REPO: stx-update

StarlingX – Software Management

Software Upgrades

Cloud Running on Rel N From Side

Cloud Running on Rel N+1 To Side

- Integrated end-to-end rolling upgrade solution
  - Automated, low number of steps
  - No additional hardware required for upgrade
- Rolling Upgrade across Nodes
  - Managing API compatibility between nodes at N and N+1 release
  - Migrating of Hosted Applications;
  - No requirement for Hosted Applications to be rebuilt/stopped
- Manages Upgrades of all Software
  - Host OS changes,
  - New / upgraded OS packages,
  - New / upgraded StarlingX Service Software,
  - New / Upgraded Cloud Software.
- Transparently deals with database schema changes and database data conversion.

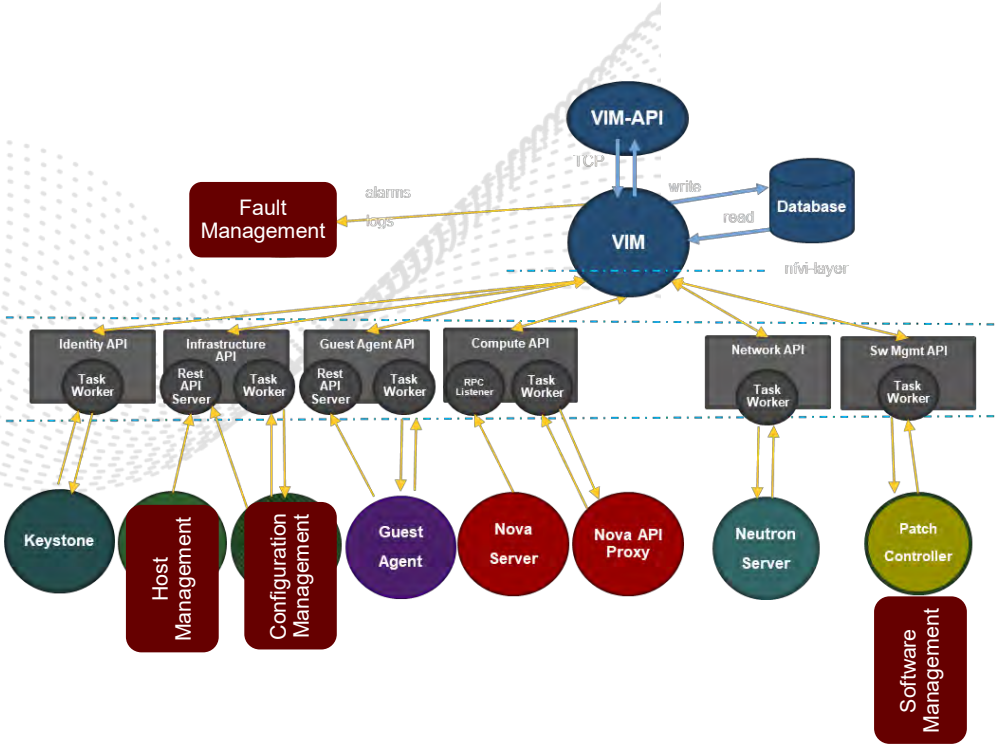
Hitless Software Upgrades

REPO: stx-nfv

StarlingX – Infrastructure Orchestration

VM High Availability

- Responsible for managing and orchestrating the VM carrier grade and high availability capabilities,
- Auto-healing of failed instances
- Orchestrates guest API actions
- Raising and clearing operator alarms about instances. Generating operator logs about instances
- Orchestrates the migration of instances off compute hosts being taken administratively out of service
- Orchestrates software patching and upgrades of hosts



REPO: stx-nfv

StarlingX – Infrastructure Orchestration

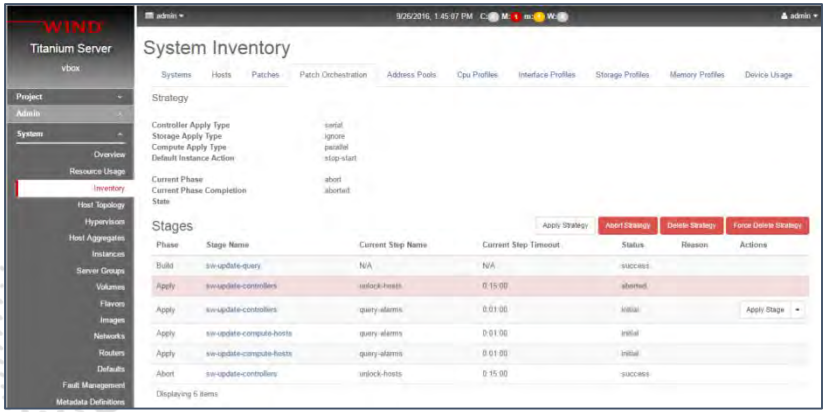
Patching / Upgrades Orchestration

Patching Orchestration:

- Automates the installation of a patch across a deployment leveraging the S/W patching API's
- Automatically iterates through all nodes of system and installs patch(s)
  - Can patch nodes in parallel if criteria met
- Automatically migrates VMs through procedure
- Exposed via REST API, Horizon, and CLI

Upgrades Orchestration:

- Upgrades orchestration capability, similar to patch orchestration leveraging the upgrades API's enabling automated S/W upgrade of the deployment

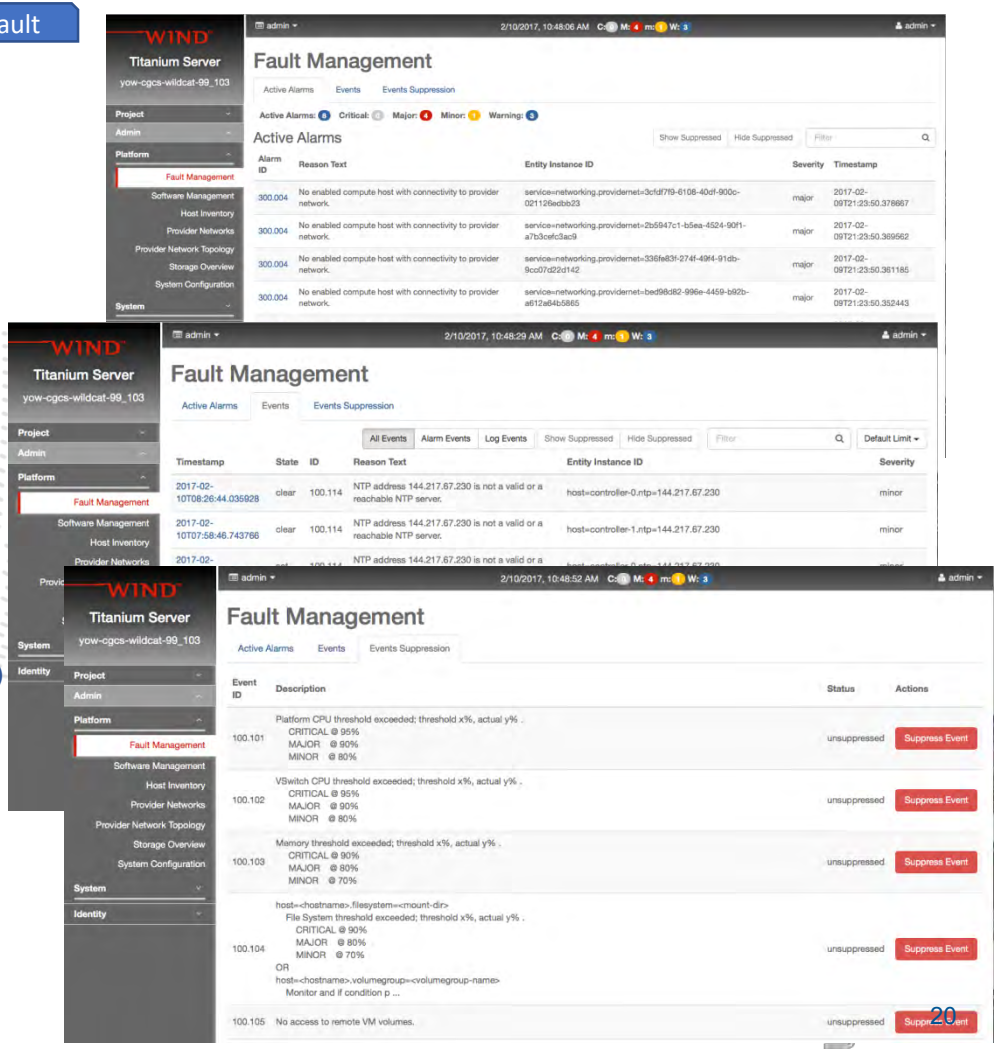



```
// create patch orchestration plan;
// specifying parameters like VM-MODE
% sw-manager patch-strategy create … <options>
%

// display patch orchestration plan details
// including order node actions, vm actions, etc.
% sw-manager patch-strategy show
%

// run the patch orchestration
% sw-manager patch-strategy apply [stage]
%
```

REPO: stx-fault

StarlingX – Fault Mgmt

- Framework for infrastructure services via a client API to:
  - Set, clear and query customer alarms
  - Generate customer logs for significant events
- Framework maintains an Active Alarm List
  - Active Alarm Counts Banner on Horizon GUI
- Framework provides REST API to query alarms & events and also publish through SNMPv2c Traps
- Support for Alarm suppression
- Operator Alarms
  - On Platform Nodes & Resources
  - On Hosted Virtual Resources (i.e. VMs, Volumes, Networks)
- Operator Logs - Event List
  - Logging of Set/Clears of Alarms,
  - Related to Platform Nodes & Resources
  - Related to Hosted Virtual Resources (i.e. VMs, Volumes, Networks)



Installation Overview

StarlingX – Installation Concepts

- StarlingX supports an install and deploy of a bare metal openstack cloud,
- StarlingX does a "*full*" install (OS+Applications) on "*dedicated*" physical servers,
- To perform a StarlingX installation, you must build a StarlingX installation ISO,
  - i.e. a StarlingX installation ISO can be built using the build-tools and build instructions found in the StarlingX Repos,
  - This ISO contains the images for Controller Nodes, Compute Nodes and Storage Nodes.

- StarlingX does NOT require a separate installer / genesis server
  - The initial server installed becomes controller-0,
  - Any controller in StarlingX deployment can provide "Installation Services" for itself and the rest of the deployment
    - i.e. modify system-wide or host configuration, add/install new host, remove existing host, etc.

StarlingX – Installation Procedure Overview

1. Install initial server from USB or an External PXE Boot Server
   - NOTE full server install of OS + StarlingX Integrated Software
   - NOTE this initial server becomes ‘controller-0’

2. Run bootstrap command to bring up system and external network connectivity to APIs
   - After completion of this step, Horizon and Remote APIs / CLI are enabled.

3. Now ‘controller-0’ provides the ‘Installation Services’ for the other servers.
   For each remaining server,
   a) Power on server w/wiped disk & BIOS for PXE Boot ➔ controller-0 detects DHCP Requests & discovers server,
   b) Select ‘personality’, { controller, compute or storage } ➔ controller-0 INSTALLS SOFTWARE on server,
   c) controller-0 runs inventory discovery on new server to learn of hardware and resources,
   d) User configures details of Host; i.e. interfaces, disks, memory pages, etc.
   e) User UNLOCKS server ➔ controller-0 applies puppet manifest to configure server and bring it into service.

   ... Rinse & Repeat for all initial servers in system.

Low touch deployment without Installer Node

How to Build the StarlingX Installation ISO

StarlingX – How to build Installation ISO

- See README file @ https://github.com/openstack/stx-tools

How to Install StarlingX ‘Single Server’

StarlingX – How to Install ‘Single Server’ *(1 of #)*

- Create a bootable USB flash drive with the StarlingX installation ISO,
- Insert USB flash drive into Hardware Server,
- Power on Hardware Server and force it to boot from USB,
- Select the following options from the Installer Welcome Screen
  - All-in-One Controller Configuration
  - Serial or Graphical Console *... as appropriate*
  - Standard Security Boot Profile
- Installation completes and Host resets automatically.

StarlingX – How to Install ‘Single Server’ *(2 of #)*

- Login as wrsroot with password of wrsroot
  - You are forced to change password

- Run bootstrap configuration script

```
% sudo config_controller
```

  - And answer the prompts appropriately;
    - System date and time: <current date and time if incorrect>
    - System timezone: <current timezone or leave it as UTC>
    - System mode: simplex
    - External OAM Network (Interface)
      - i.e. the External OpenStack API Interface
      - Prompts for physical interface, vlan and/or lag configurations, MTU, IP Subnet, Gateway address, etc.

StarlingX – How to Install 'Single Server' *(3 of #)*

- Create a Provider Network

```
% openstack providernet create pnet-0 --type vlan
```

- Create Segmentation Ranges within Provider Network

```
% openstack providernet range create pnet-0 \
                        --name pnet-0-Rexternal --range 10-10
% openstack providernet range create pnet-0 \
                        --name pnet-0-Rinternal --range 600-700
```

- Configure Host Interface for Provider Network (i.e. Data Interface)
  - List all interfaces

```
% system host-if-list -a controller-0
```

  - Configure the appropriate interface as the Data Interface

```
% system host-if-modify –nt data controller-0 <ethname> -p pnet-0
```

StarlingX – How to Install 'Single Server' *(4 of #)*

- Configure Cinder Storage on Controller-0
  - List all disks

```
% system host-disk-list controller-0
```

  - Create cinder-volumes local volume group

```
% system host-lvg-add controller-0 cinder-volumes
```

  - Create disk partition to add to the volume group

```
% system host-disk-partition-add controller-0 <disk-uuid> -t lvm_phys_vol
```

  - Wait for new partition to be created and status=Ready

```
% system host-disk-partition-list controller-0 --disk <disk-uuid>
```

  - Add partition to the volume group

```
% system host-pv-add controller-0 cinder-volumes <partition-uuid>
```

  - Add (controller) LVM storage backend for cinder

```
% system storage-backend-add lvm -s cinder
```

  - Wait for storage backend to finish configuring ... Monitor by:

```
% system storage-backend-list
```

StarlingX – How to Install 'Single Server' *(5 of #)*

- Configure Nova Ephemeral Storage on Controller-0
  - List all disks

```
% system host-disk-list controller-0
```

  - Create nova-local local volume group

```
% system host-lvg-add controller-0 nova-local
```

  - Create disk partition to add to the volume group

```
% system host-disk-partition-add controller-0 <disk-uuid> -t lvm_phys_vol
```

  - Wait for new partition to be created and status=Ready

```
% system host-disk-partition-list controller-0 --disk <disk-uuid>
```

  - Add partition to the volume group

```
% system host-pv-add controller-0 nova-local <partition-uuid>
```

  - Specify the nova-local storage space

```
% system hostd-lvg-modify -s 100000 controller-0 nova-local
```

StarlingX – How to Install ‘Single Server’ *(6 of #)*

- Unlock Controller-0
  - Bring controller-0 into service

```
% system host-unlock controller-0
```

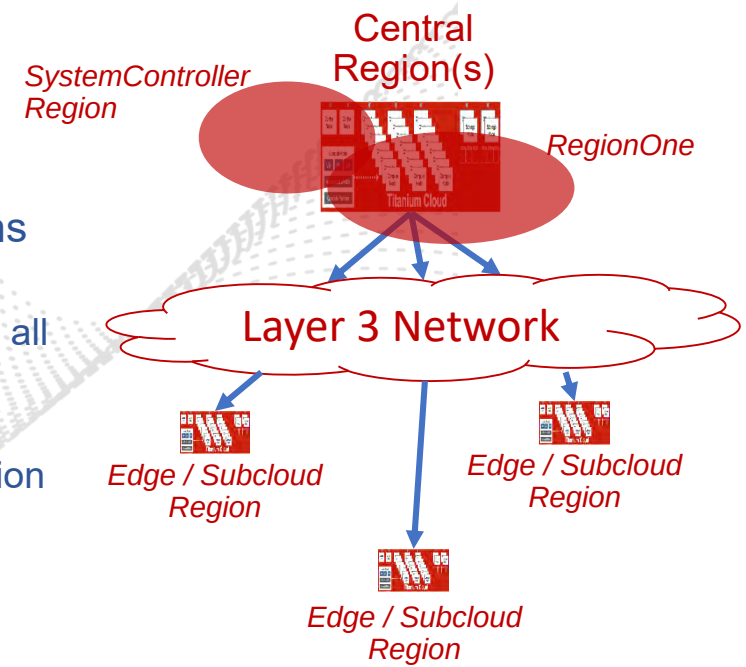
  - Host will reboot and come back up unlocked and in-service.
  - Log back in

```
% source /etc/nova/openrc
~(keystone_admin)$ nova service-list
+-------+------------------+--------------+----------+---------+-------...
|  Id   | Binary           | Host         | Zone     | Status  | State ...
+-------+------------------+--------------+----------+---------+-------...
| 45... | nova-conductor   | controller-0 | internal | enabled | up    ...
| 87... | nova-scheduler   | controller-0 | internal | enabled | up    ...
| fe... | nova-consoleauth | controller-0 | internal | enabled | up    ...
| db... | nova-compute     | controller-0 | nova     | enabled | up    ...
+-------+------------------+--------------+----------+---------+-------...

~(keystone_admin)$ system host-list
+----+--------------+-------------+----------------+-------------+--------------+
| id | hostname     | personality | administrative | operational | availability |
+----+--------------+-------------+----------------+-------------+--------------+
| 1  | controller-0 | controller  | unlocked       | enabled     | available    |
+----+--------------+-------------+----------------+-------------+--------------+
```

Incubation Projects

StarlingX – Distributed Cloud

- Based on OpenStack Regions
- Central Region (System Controller)
  - Hosting shared services
  - System wide infrastructure orchestration functions
    - Deployment and management of Edge clouds
    - Configuration portal for shared configuration across all Edges (host and openstack)
    - Fault aggregation
    - Portal for system wide patch (s/w updates) application
- Geographically dispersed remote Edge regions
  - Connected to the system controller via L3 network
- Inter-region communications via REST APIs
- Edge clouds run a reduced control plane



In-flight StarlingX Evolution

StarlingX – In Flight Evolution

- StarlingX is evolving to
  - Running OpenStack containerized,
  - On top of a Bare Metal Kubernetes Cluster,
  - With OpenStack Helm managing the Lifecycle of the OpenStack Cluster.
- With Kubernetes Cluster initially supporting
  - Docker runtime
  - Calico CNI plugin
  - CEPH as persistent storage backend
  - HELM as the package manager
  - Local Docker Image Registry.
- And Kubernetes cluster being available for non-OpenStack end user applications.