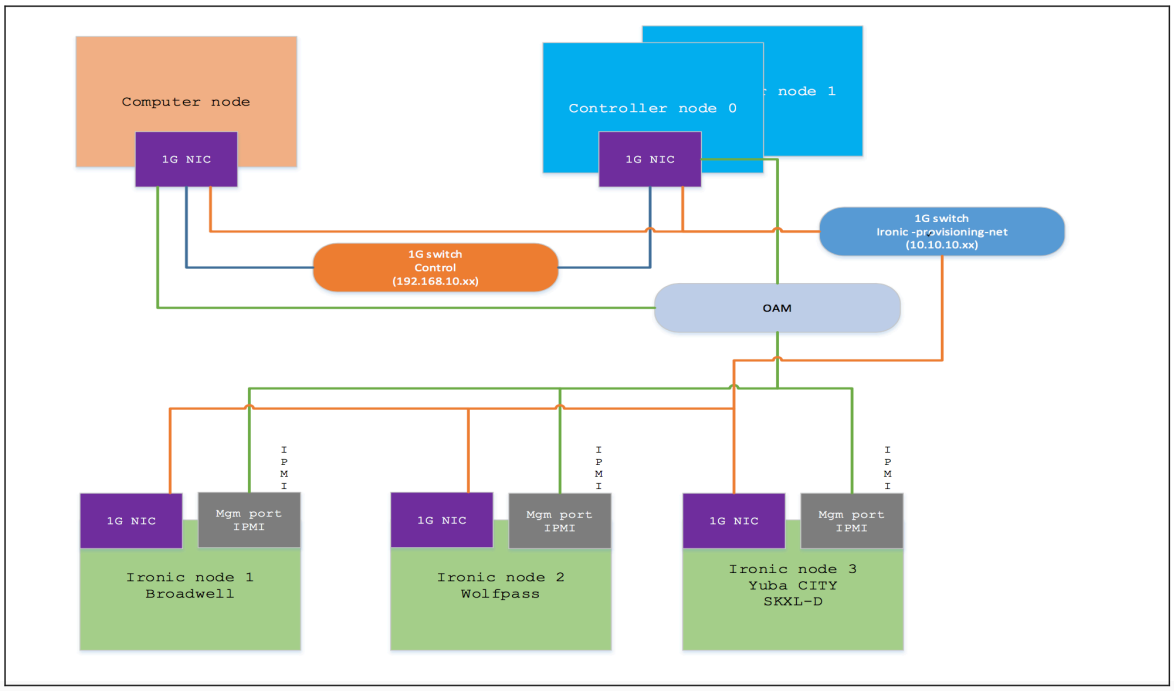
Steps to setup a bare metal host by Openstack Ironic

version: 1.0

[System Topology]



[Ironic service enablement on Controller]

mandatory - create a providernet, named ironic-provisioning-providernet
openstack providernet create --type flat ironic-provisioning-providernet

mandatory - create a network for ironic, named ironic-provisioning-net
openstack network create --provider-physical-network=ironic-provisioning-providernet --provider-network-type=flat ironic-provisioning-net

mandatory - create a subnetwork, named ironic-provisioning-subnet which is attached to ironic-provisioning-net
openstack subnet create ironic-provisioning-subnet --network ironic-

provisioning-net --gateway 18.10.10.1 --allocation-pool start=18.10.10.11,end=18.10.10.99 --subnet-range 18.10.10.0/24

mandatory - modify interface on Controller Node
NOTE: this interface is different from Titanium to StarlingX (it was changed by StarlingX patches)
“--networks pxeboot” is actually a workaround to enable later we can set IP for enp2s0 *manually*
system host-if-modify -n enp2s0 --networks pxeboot -c platform controller-0 enp2s0

mandatory - manually set IP for interface on Controller-0
//NOTE: 18.10.10.3/24 is out of the scope of allocation-pool as explained above
//sudo ip addr add 18.10.10.3/24 dev enp2s0
// **NO NEED any more — because** 18.10.10.2 as a floating TFTP server address will be automatically set to enp2s0

 # mandatory - delete that address allocated by system network “pxeboot” (as a Work-Around)
 # we do NOT need this IP
 # // sudo ip addr del 169.254.202.3/24 dev enp2s0
// **NO NEED any more — because** we can still keep this IP automatically allocated by “pxeboot” network

mandatory - modify interface on Compute Node, to make ironic-provisioning-providernet allocate
IP for enp2s0 on Compute-node
system host-if-modify -n enp2s0 -p ironic-provisioning-providernet -c data $Compute-Node-Name enp2s0

mandatory - add a few service parameters
mandatory - for ironic node, we need ironic_provisioning_network as the provider, for both deploy stage
and user image (**NOTE**: very likely, ethernet interfaces running in user image might NOT be up by default,
so we have to log in the user image by console first, and then enable the network interface by **ifconfig**)
system service-parameter-add ironic neutron provisioning_network=<IRONIC_PROVISIONING_NET_UUID>

NOTE: $Floating_IP_from_ironic-provisioning-net could be any IP which is NOT used yet by controllers
and OUT OF scope of DHCP allocation pool of ironic-provisioning-net, for example, 18.10.10.2
system service-parameter-add ironic pxe tftp_server=$Floating_IP_from_ironic-provisioning-net
system service-parameter-add ironic pxe netmask=24
system service-parameter-add ironic pxe controller_0_if=<CONTROLLER_0_INTERFACE>
system service-parameter-add ironic pxe controller_1_if=<CONTROLLER_1_INTERFACE>

mandatory - enable ironic service
system service-enable ironic

if there are alarms something like, data out-of-sync, from “$ fm alarm-list”, it is normal!
mandatory - lock and unlock controllers to make system take effects with changes done above
DO NOT forget this!!!

NOTE, before running “system host-swact”, MUST delete ironic related service parameters above FIRST!!!
system service-parameter-list
FYI: check ironic services status from /var/log/sm.log

[Import images into Glance]

mandatory - import images into Glance
import ramdisk image for deployment system
glance image-create --name deploy-initrd-0 --visibility public --disk-format ari --container-format ari < coreos_production_pxe_image-oem-stable-pike.cpio.gz

import kernel image for deployment system
glance image-create --name deploy-vmlinuz-0 --visibility public --disk-format aki --container-format aki < coreos_production_pxe-stable-pike.vmlinuz

import user image for user system
glance image-create --name “centos-root-img” --visibility public --

disk-format=qcow2 --container-format=bare --file CentOS-7-x86_64-GenericCloud-root.qcow2 --progress

modify /etc/ironic/ironic.conf based on needs:

# optional - enable debug info:
debug=true
# mandatory - workaround “erase_devices” in [deploy] session, by:
erase_devices_priority = 0
# optional - append sshkey in pxe so that we can log into deploy ramdisk by SSH for debugging:
pxe_append_params = nofb nomodeset vga=normal console=ttyS0,115200n8 sshkey=“your_id_rsa.pub key content”

# mandatory - whenever ironic.conf is changed, need to restart ironic services:
sudo sm-restart service ironic-api
sudo sm-restart service ironic-conductor

[use Ironic to manage nodes on Controller]

use ironic client to config the node
export IRONIC_API_VERSION=latest

# NOTE: mandatory - need to do some settings in BIOS in Bare Metal host, to make IPMI work
# test IPMI works on bare metal host.
sudo ipmitool -I lan -H 10.10.10.11 -p 623 -L ADMINISTRATOR -U root -P "test123" sdr list

mandatory - create an ironic node
specify a few important items, such as ipmi_address, ipmi user name, password (all were set in BIOS in advance)
ironic --ironic-api-version latest node-create -d pxe_ipmitool_socat -i ipmi_address=10.10.10.10 -i ipmi_username=root -i ipmi_password=test123

#// ironic --ironic-api-version latest node-create -d pxe_ipmitool_socat -i ipmi_address=10.10.10.11 -i ipmi_username=root -i ipmi_password=test123

```
#// remove ipmi_username and ipmi_password
#//ironic --ironic-api-version latest node-update $NODE_ID remove
driver_info/ipmi_username driver_info/ipmi_password

# //ironic --ironic-api-version latest node-update $NODE_ID add
driver_info/ipmi_address=10.10.10.10 driver_info/ipmi_username=root
driver_info/ipmi_password=test123

# set the name for the newly created ironic node
ironic --ironic-api-version latest node-update $NODE_ID add
name=bm_node_10

# optional - check this node
ironic node-list
ironic node-show $NODE_ID

# mandatory - check the images in Glance for ramdisk and kernel
image
glance image-list

# mandatory - set ramdisk and kernel in driver_info for the node
ironic --ironic-api-version latest node-update $NODE_ID add
driver_info/deploy_kernel=e704ed97-2dba-42f3-88cb-25169d991afe
driver_info/deploy_ramdisk=f4dd5231-a588-4eae-
a22a-0a59b1d2a70b

# mandatory - set ipmi terminal port  in driver_info for the node
ironic --ironic-api-version latest node-update $NODE_ID add
driver_info/ipmi_terminal_port=623

# mandatory - set properties for the node
ironic --ironic-api-version latest node-update $NODE_ID add
properties/cpu_arch=x86_64 properties/cpus=4 properties/
capabilities="boot_option:local" properties/memory_mb=8192
properties/local_gb=400

# optional - enable the console mode for the node, for debugging
purpose mainly
ironic  --ironic-api-version latest node-set-console-mode $NODE_ID
true
```

mandatory - create an ethernet port by adding MAC (a4:bf:01:2b:3b:c9) for the node.
NOTE: this port is *different* from the BMC management port which is set in BIOS for IPMI
This port connects to ironic-provisioning-net and it will be used for IPA at deploy stage
ironic --ironic-api-version latest port-create -n $NODE_ID -a a4:bf:01:2b:3b:c9 --pxe-enabled True

#// ironic --ironic-api-version latest port-create -n <node-2> -a a4:bf:01:2b:34:3f --pxe-enabled true

mandatory - change the provision state to “manageable” from “enroll” state, by “manage” event
details refer to https://docs.openstack.org/ironic/pike/contributor/states.html
ironic --ironic-api-version latest node-set-provision-state $NODE_ID manage

mandatory - change the provision state to “available” from “manageable” state, by “provide” event.
there could be a few intermediate states, such as “cleaning”, “clean-wait”, or “clean-fail” if something bad happened.
details refer to https://docs.openstack.org/ironic/pike/contributor/states.html
to debug the failure, check /var/log/ironic/ironic-conduct.log.
ironic --ironic-api-version latest node-set-provision-state $NODE_ID provide

ironic node-show $NODE_ID

by here ironic node provisioning state should be “available” if everything went well.
then you should be able to create system on this ironic (bare metal) node, similar to VM creating procedures.

use openstack create server or nova client to create an instance with the flavor (in which baremetal is set 'true’ as one of properties)
for example, here is my command line: *openstack server create* --

flavor bmc-flavor --image my-user-image-name --nic net-id=ironic-provisioning-net --key-name baremetalsshkey BMC-linux-11

It takes a few minutes to build instance. (BTW: it's much slower than creating a VM instance.)
once the instance is running (you can check its state by *openstack server list/show),* you have to log in the system by console for the 1st time. To do so, you need to know its user and password, otherwise, you will have no way to log in and make further operations, such as network configurations!!

Enjoy! ALL set!!

#
[———————————— the end ————————————]