

# Development for Fully-Automated Bare Metal Provisioning in OpenStack

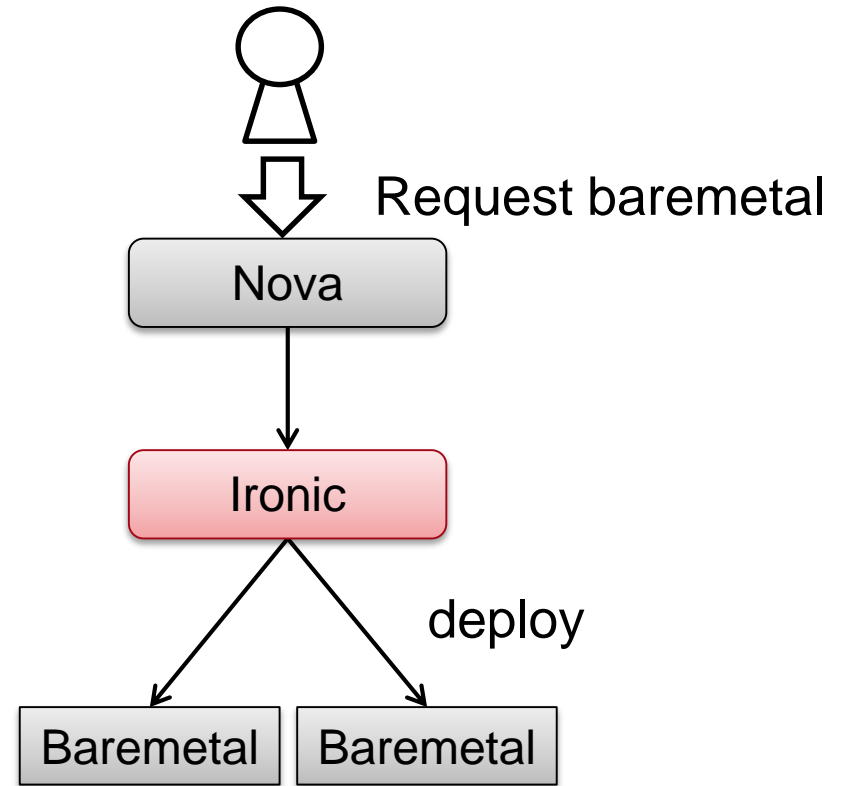
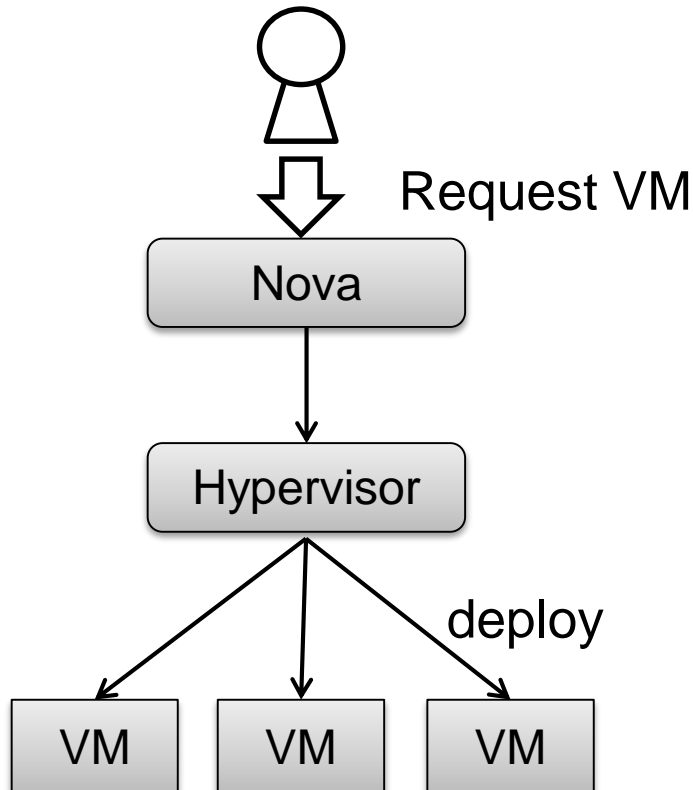
June 5, 2015  
Hironori Shiina  
Fujitsu Limited

- OpenStack Ironic overview
- Our development for baremetal provisioning
  - Automated provisioning with multi tenancy
  - Network isolation with automated network configuration

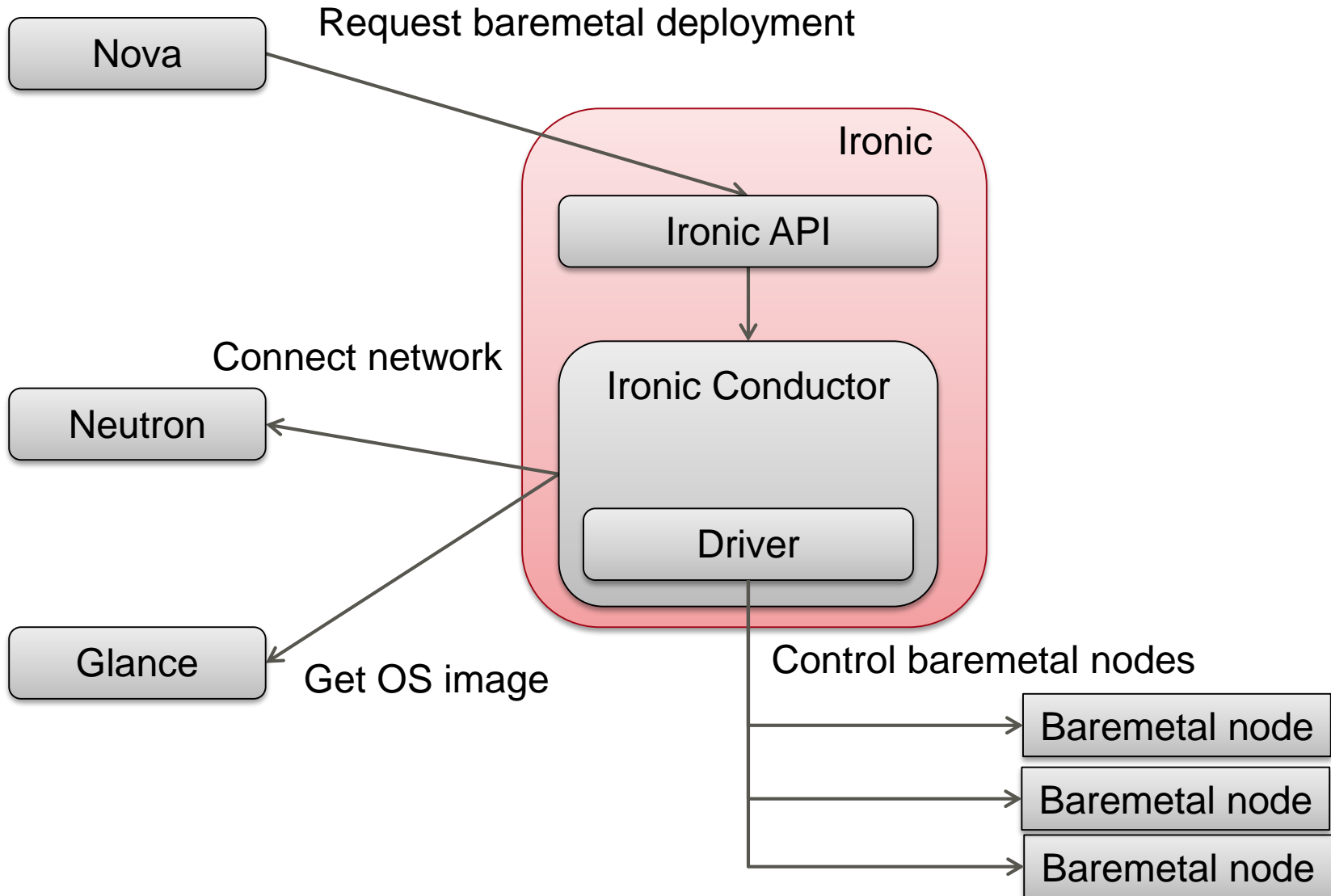
# OpenStack Ironic overview

- Workloads requiring high performance are unsuited for VMs (e.g. Enterprise database)
- Some users cannot allow to be affected by other users
  
- Easy deployment is required for baremetal like VM deployment
  - Automated setup
  - On demand deployment

- Ironic provides baremetal provisioning
- Officially released since OpenStack Kilo (Apr, 2015)
- Users can request baremetal instances with the same interface as VM instances

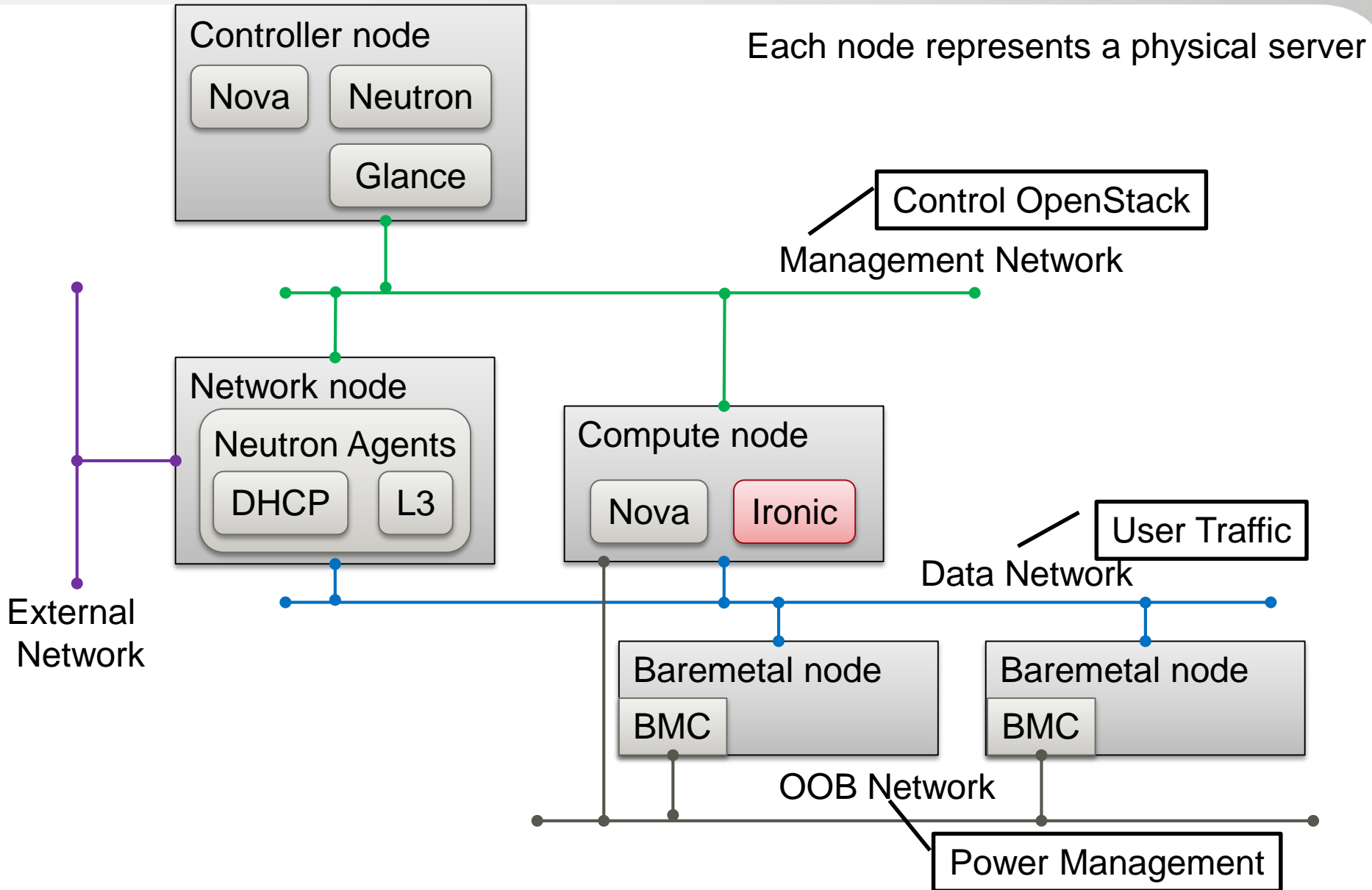


# Architecture of Ironic



- Provides interface between Ironic and baremetal nodes
- Default driver controls nodes with IPMI (commonly available)
- Hardware vendors implement their own drivers to provide improved performance and additional functions
  - iRMC driver (Fujitsu)
  - iLO driver (HP)
  - DRAC driver (Dell)
  - etc...
- Various deployment methods are implemented by drivers
  - PXE boot
  - Ironic Python Agent
  - Virtual media deployment

# System overview

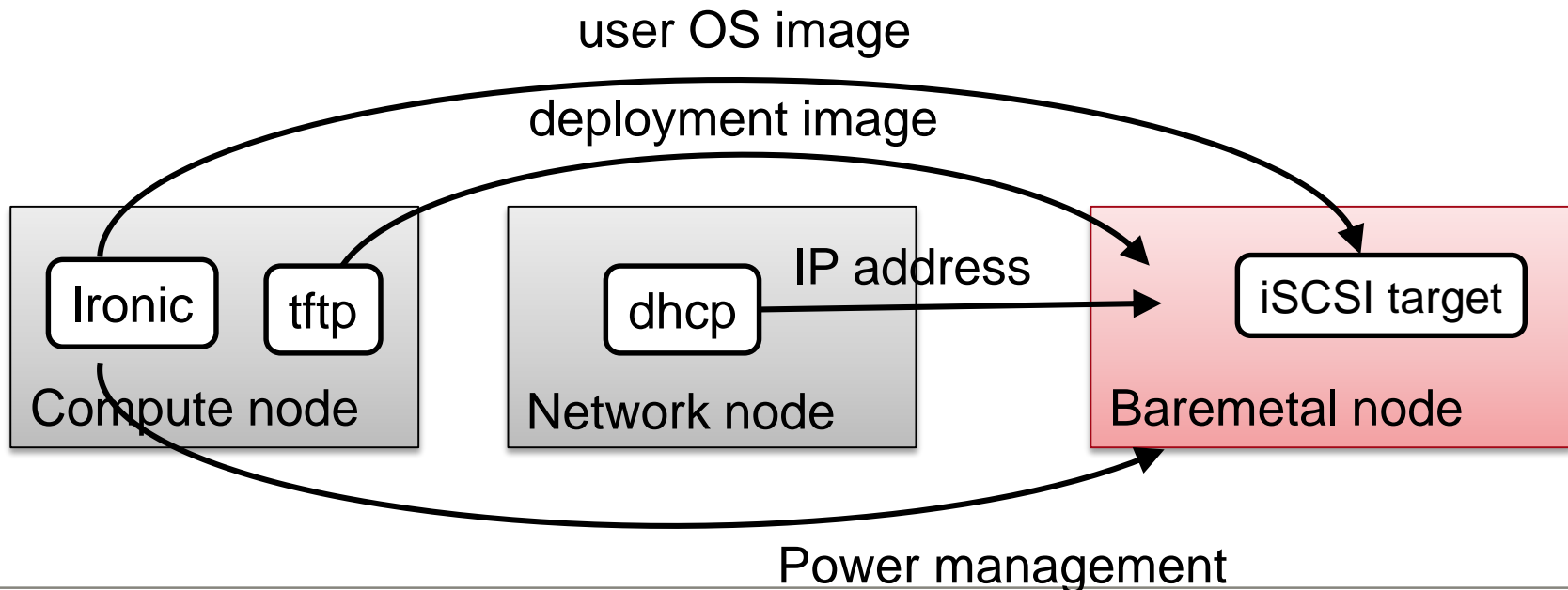




- Before Deployment, administrator needs setup
  
- Enroll baremetal nodes to Ironic
  - Register node specs (CPU number, Memory size, etc...)
  - Register MAC address as a port
  - Register a driver and enroll BMC access information
  
- Create flavors for baremetal deployment  
(User requests a baremetal node by selecting a flavor)
  
- Create disk images for baremetal
  - Deployment image (only used for deployment)
    - bm-deploy-kernel and bm-deploy-ramdisk
  - User OS image
    - user-image, user-image-vmlinuz and user-image-initrd

# Overview of Deployment

1. Ironic powers on a baremetal node using a driver
2. The baremetal node gets a deployment image
3. The deployment image configures iSCSI target
4. Ironic copies a user OS image to the baremetal node
5. Ironic reboots the baremetal node
6. The baremetal is booted by the user image



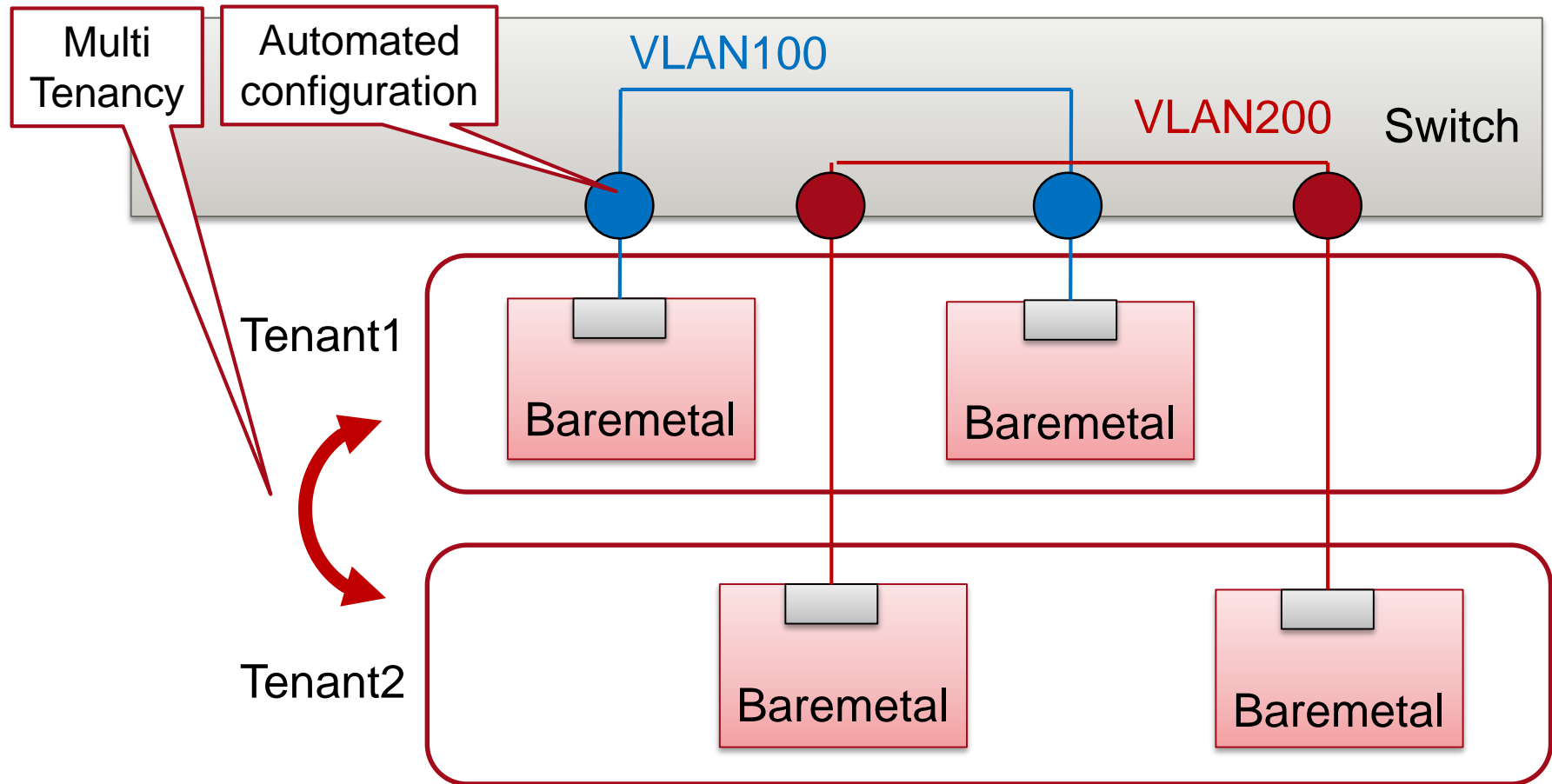
# Our development for automated provisioning

# More advanced features are required

- Current Ironic function
  - Power management
  - Deploy OS
  
- Ironic community continues enhancing Ironic functions
- More features are necessary to use baremetal nodes like VMs
  - Multi tenancy
  - Attach virtual volumes (cinder integration)
  - Security groups
  - Etc...
  
- We focus on multi tenancy in this presentation

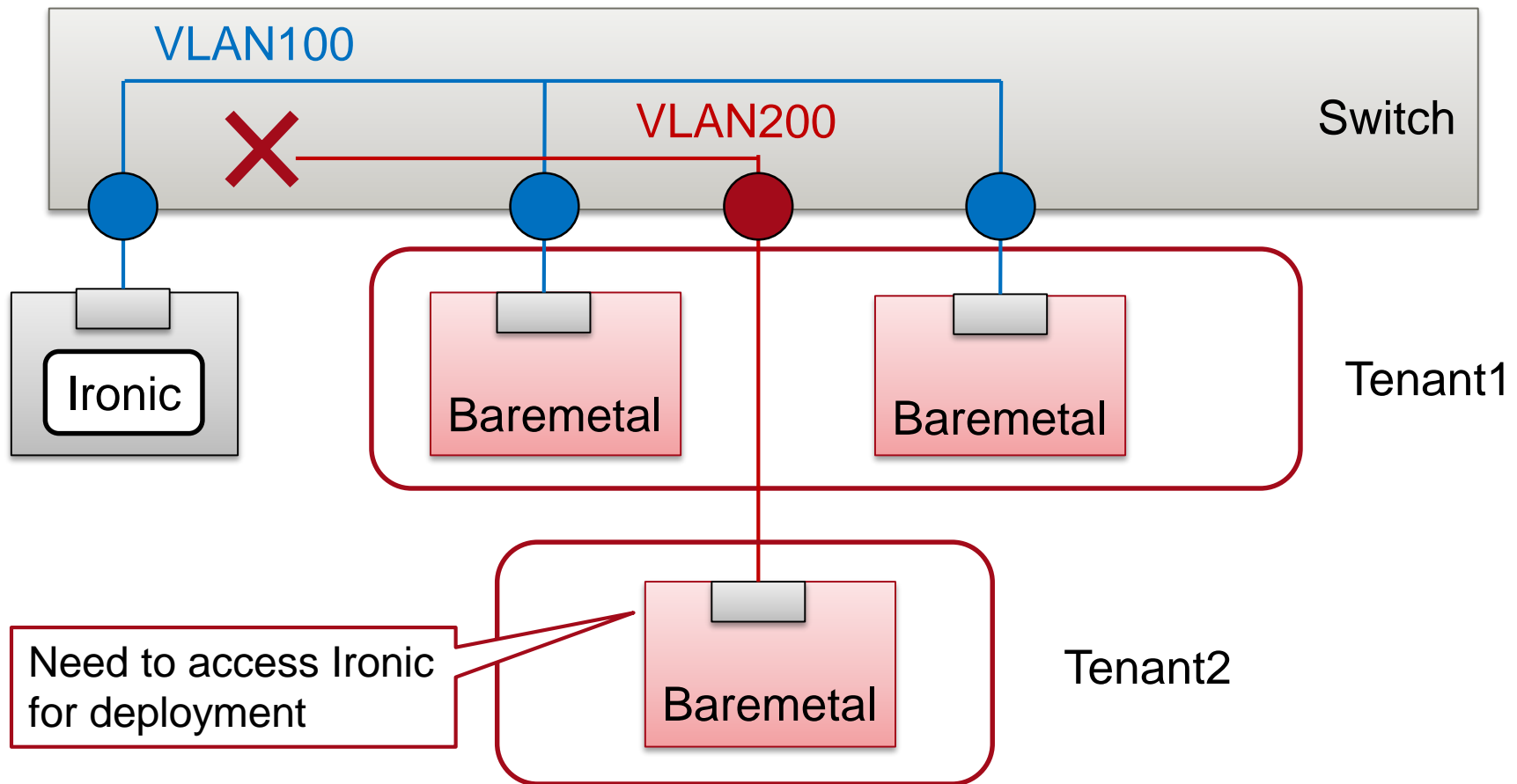
# Our Goal

- For the multi tenancy, network isolation is necessary
- Network isolation requires some configuration to physical switches
- We will automate network configuration for multi tenancy



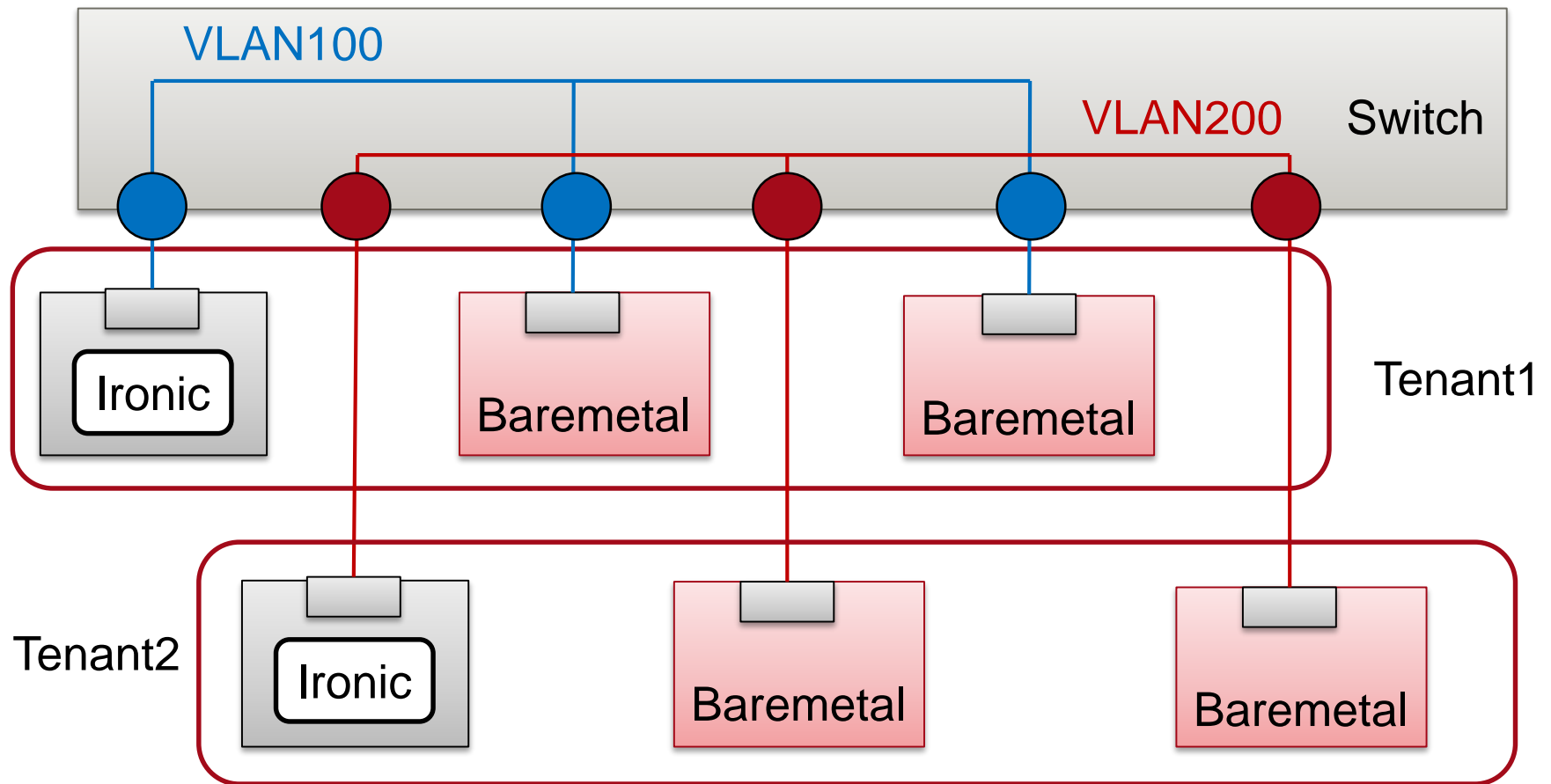
# First idea

- First, we thought of dividing network with VLAN simply
- When deploying, all baremetal nodes need to get OS image from Ironic



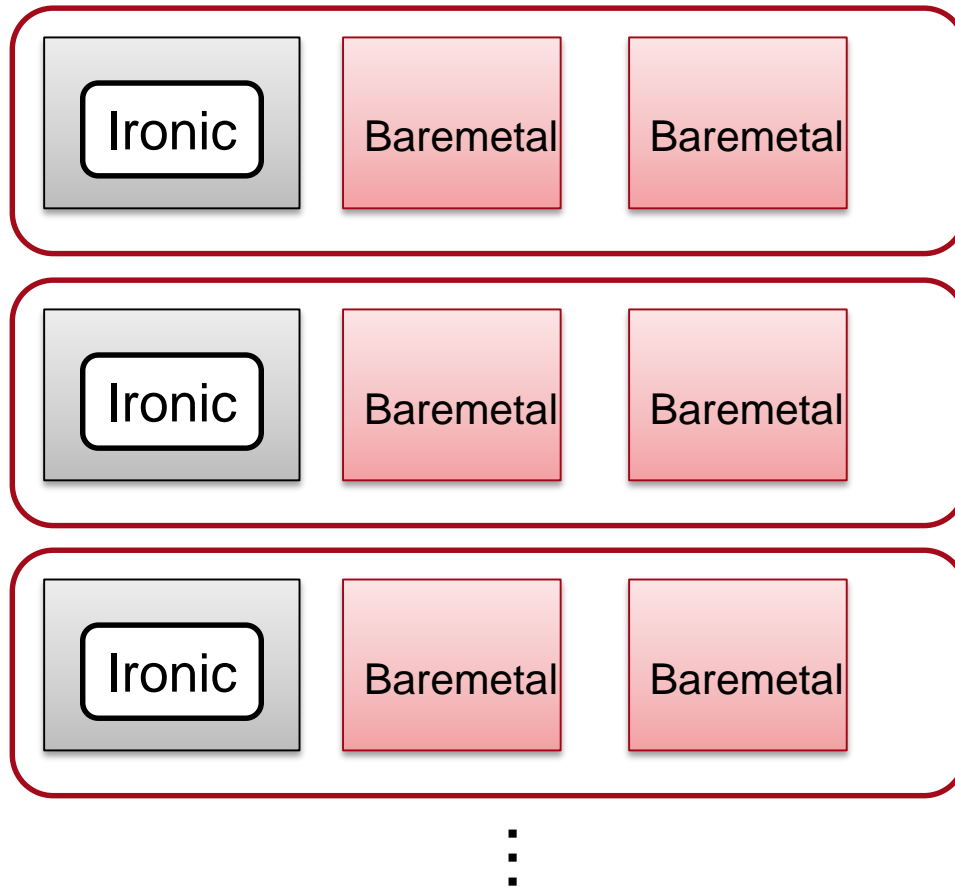
# Next idea

- Could we add a Ironic node to another tenant?



# Should each tenant have a node for Ironic?

- The idea is not practical because it consumes too many nodes for Ironic

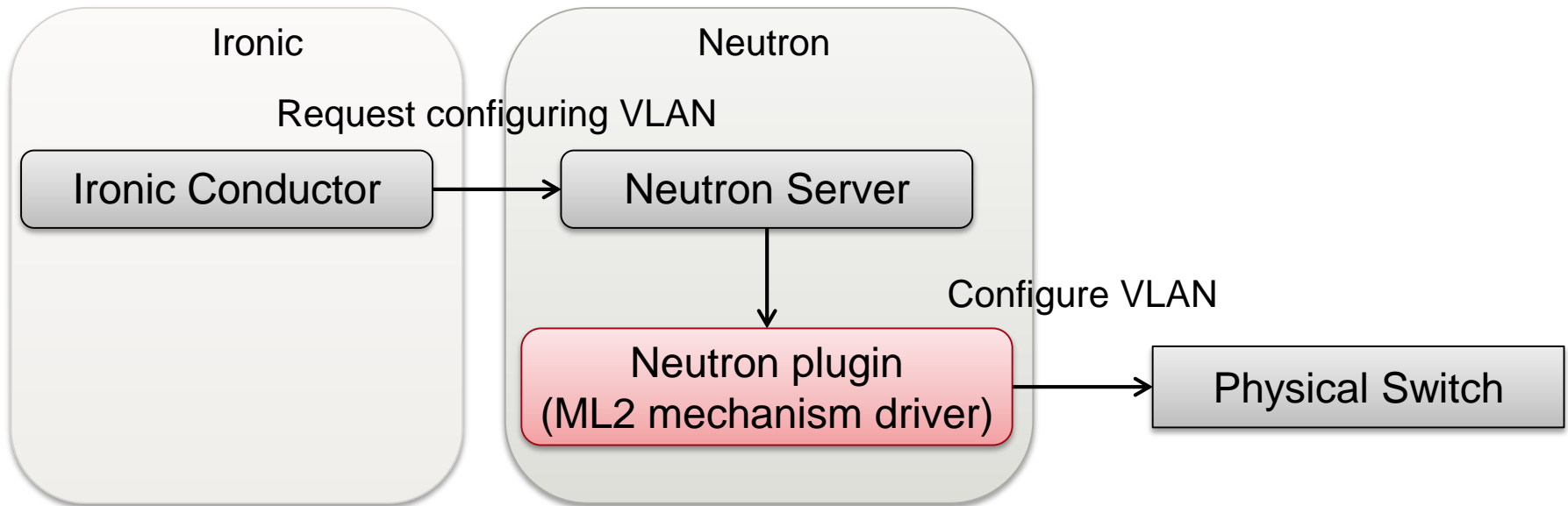




- One Ironic manages multiple tenants
  
- Use two types of VLAN
  - Deployment VLAN
    - Created by administrator as a Neutron network
    - Ironic compute node is connected to this VLAN
    - Each baremetal node connects to this VLAN only when deployment
  - Tenant VLAN
    - Created by a tenant user as a Neutron network
    - Baremetal nodes in a tenant connect to this type of VLAN after deployment
  
- Switch VLAN types before and after deployment

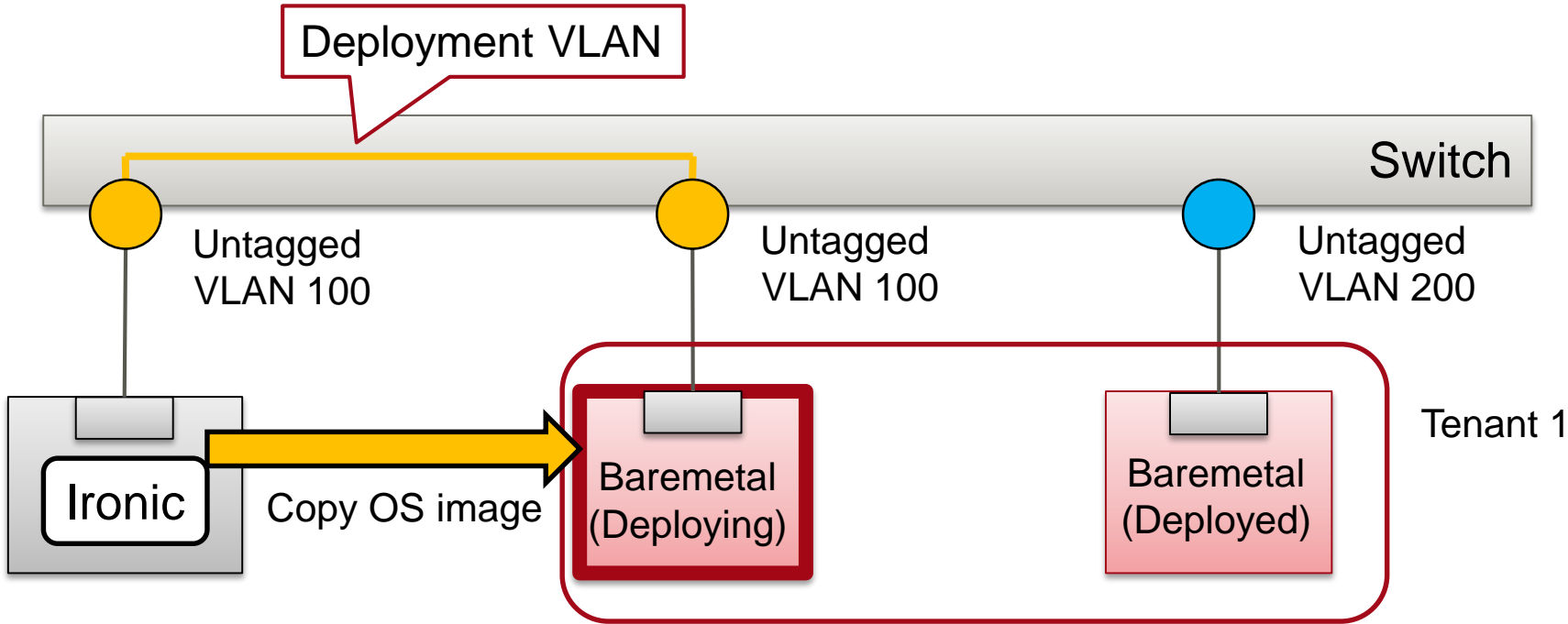
# Control physical switches

- Control switches by Neutron plugin
  - Configure VLAN of a port in our solution
- We're planning to implement this plugin as a ML2 mechanism driver



# Our solution overview (1/4)

- A baremetal node is deployed by using the deployment VLAN

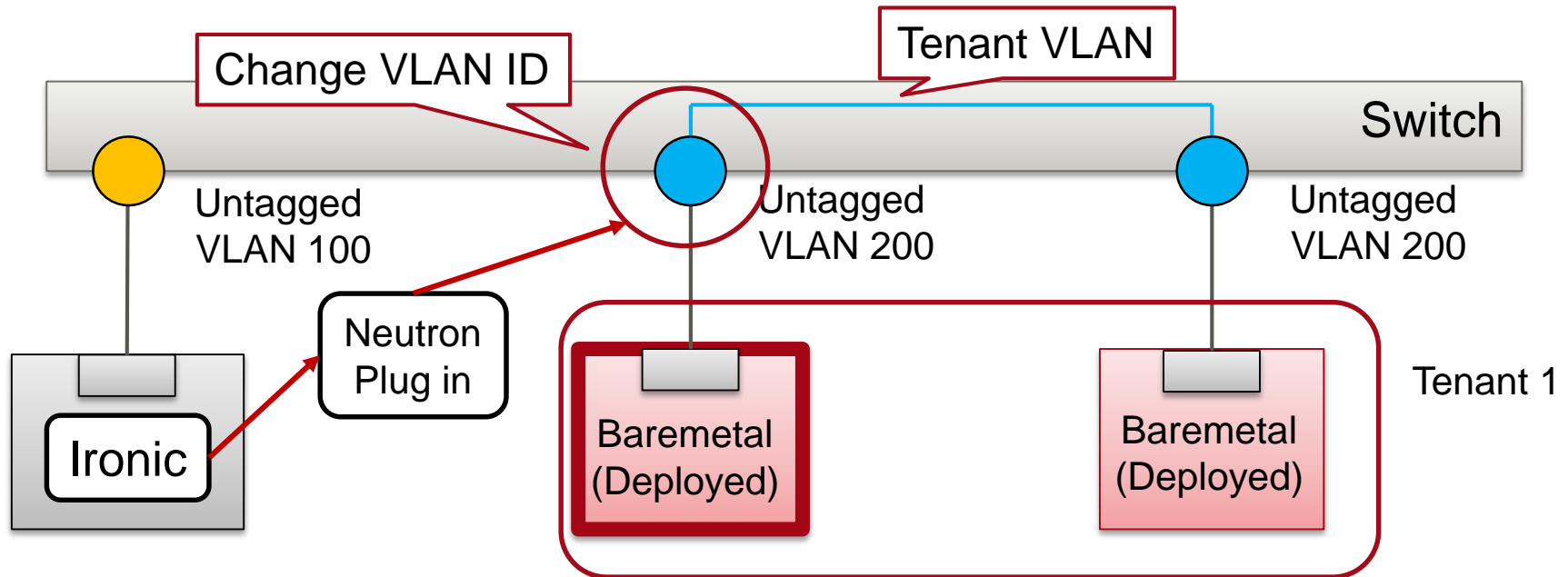


VLAN ID

- 100:Deployment
- 200:Tenant1

# Our solution overview (2/4)

- After deployment, Ironic changes the VLAN ID so that the baremetal node connects to the tenant VLAN



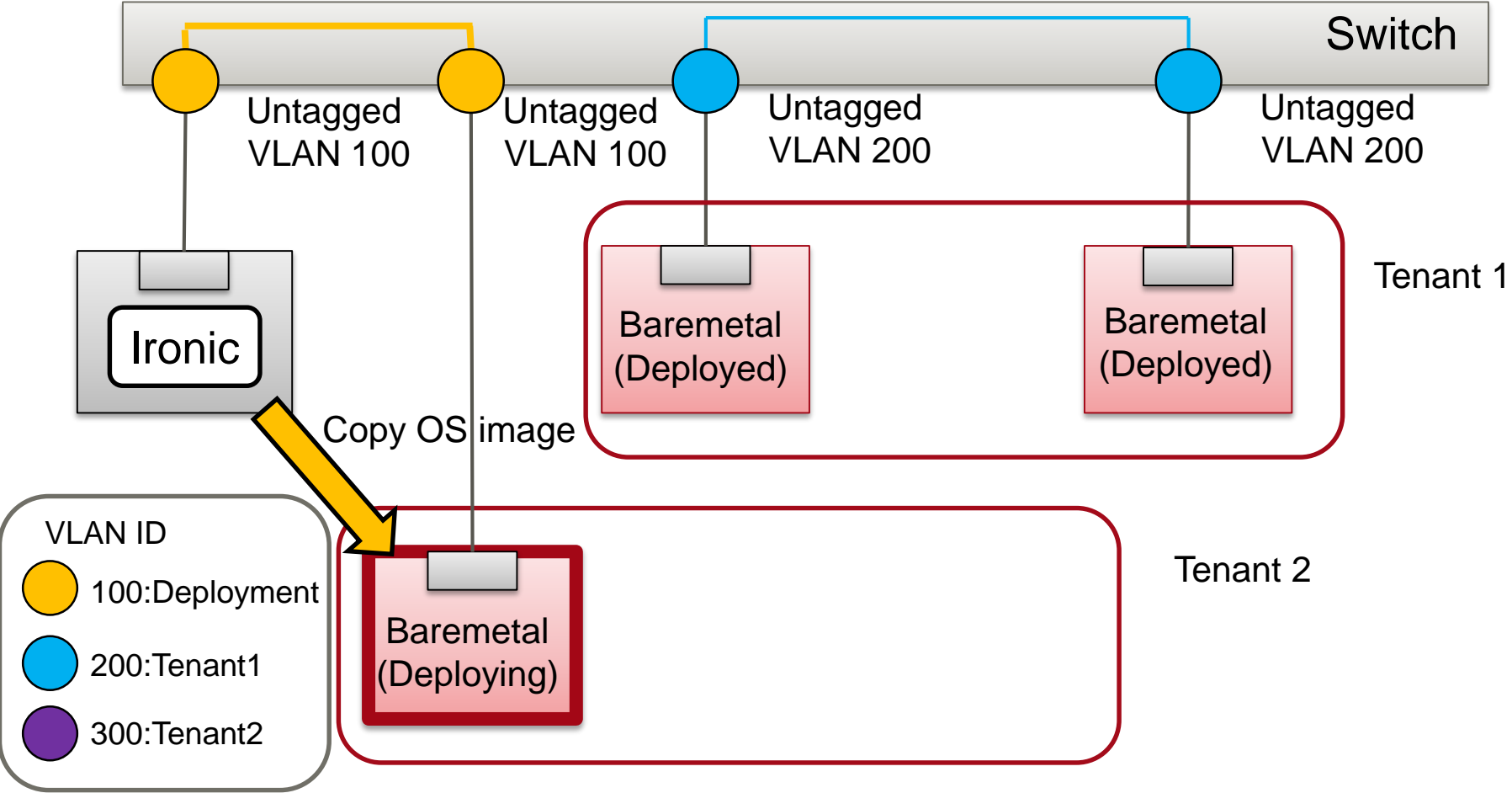
VLAN ID

100:Deployment

200:Tenant1

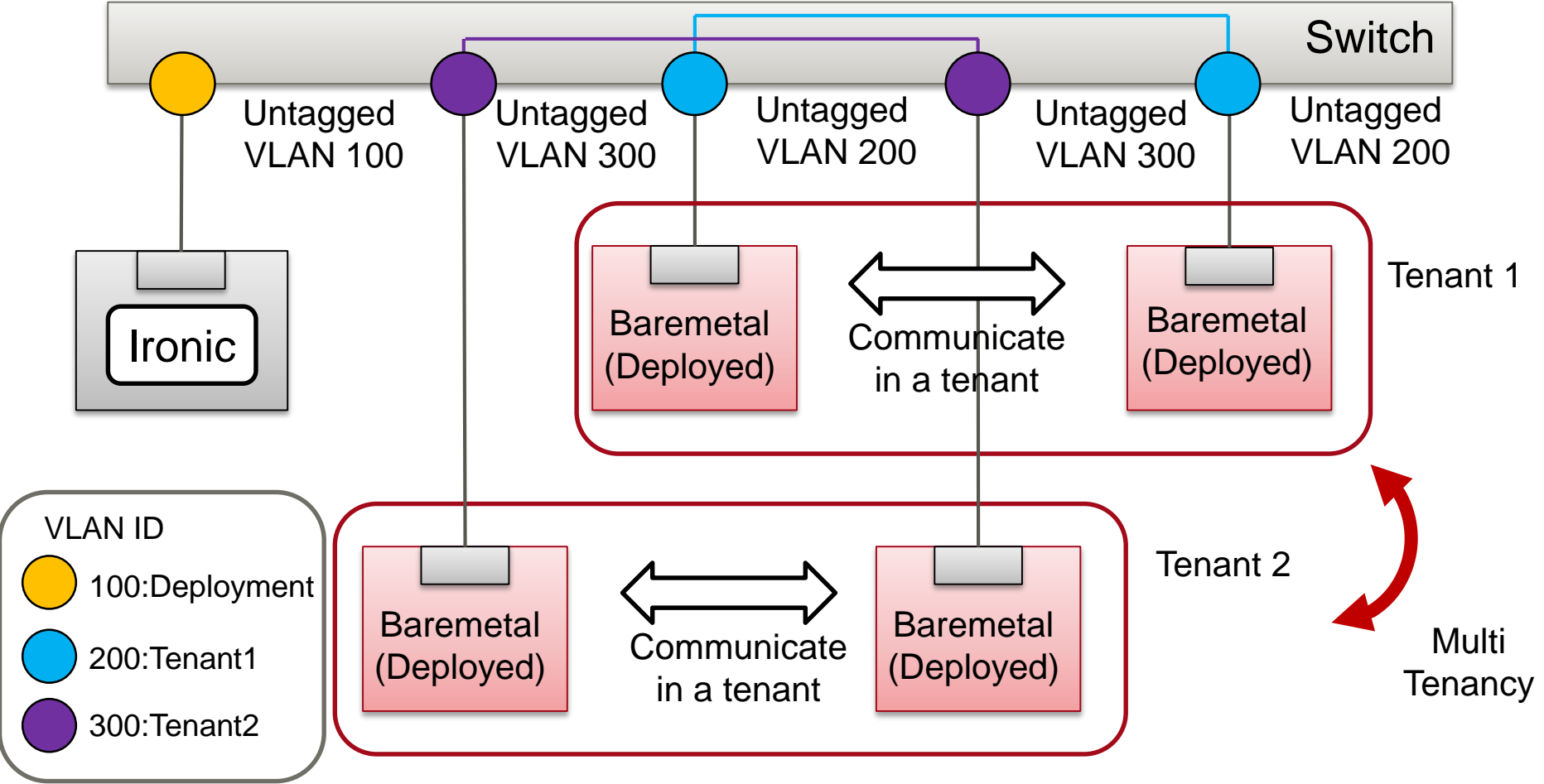
# Our solution overview (3/4)

- A baremetal node of another tenant also can be deployed by using the deployment VLAN



# Our solution overview (4/4)

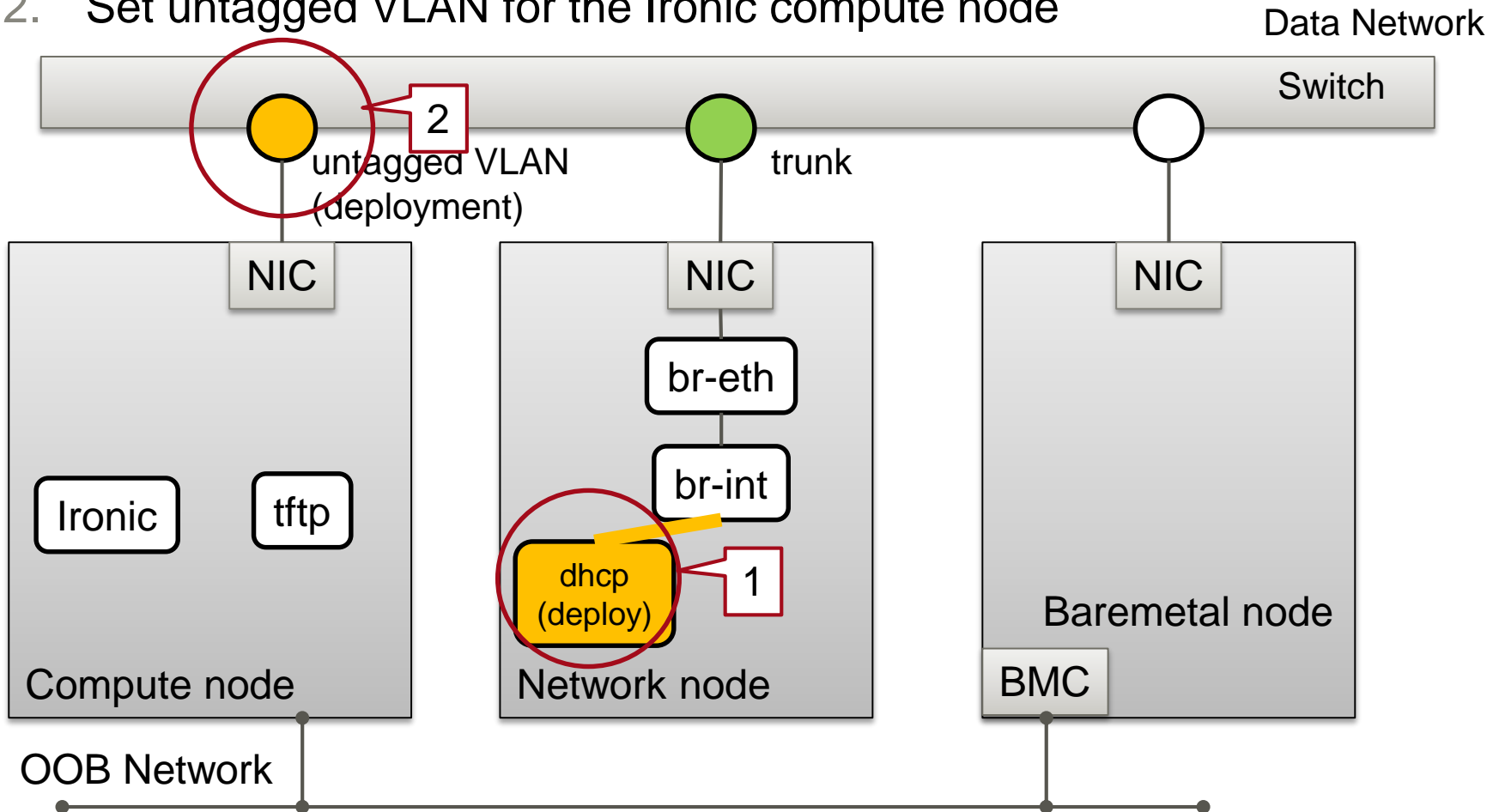
■ By switching VLANs, Ironic can manage all tenants



# Deployment flow (preparation)

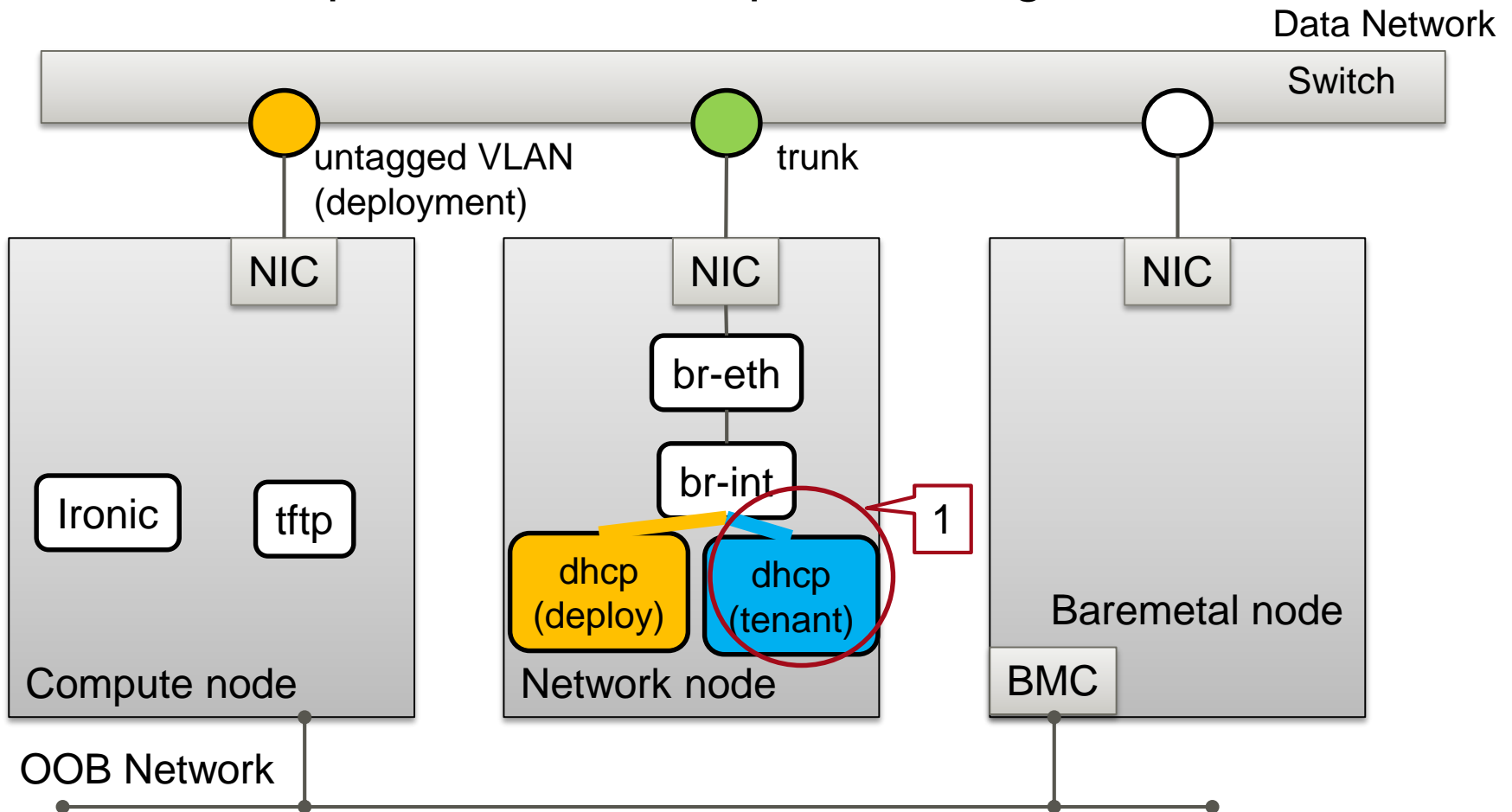
## ■ Administrator operations

1. Create a Neutron network (Deployment VLAN)  
Then, Neutron creates a DHCP server on the network
2. Set untagged VLAN for the Ironic compute node



# Deployment flow (1/6)

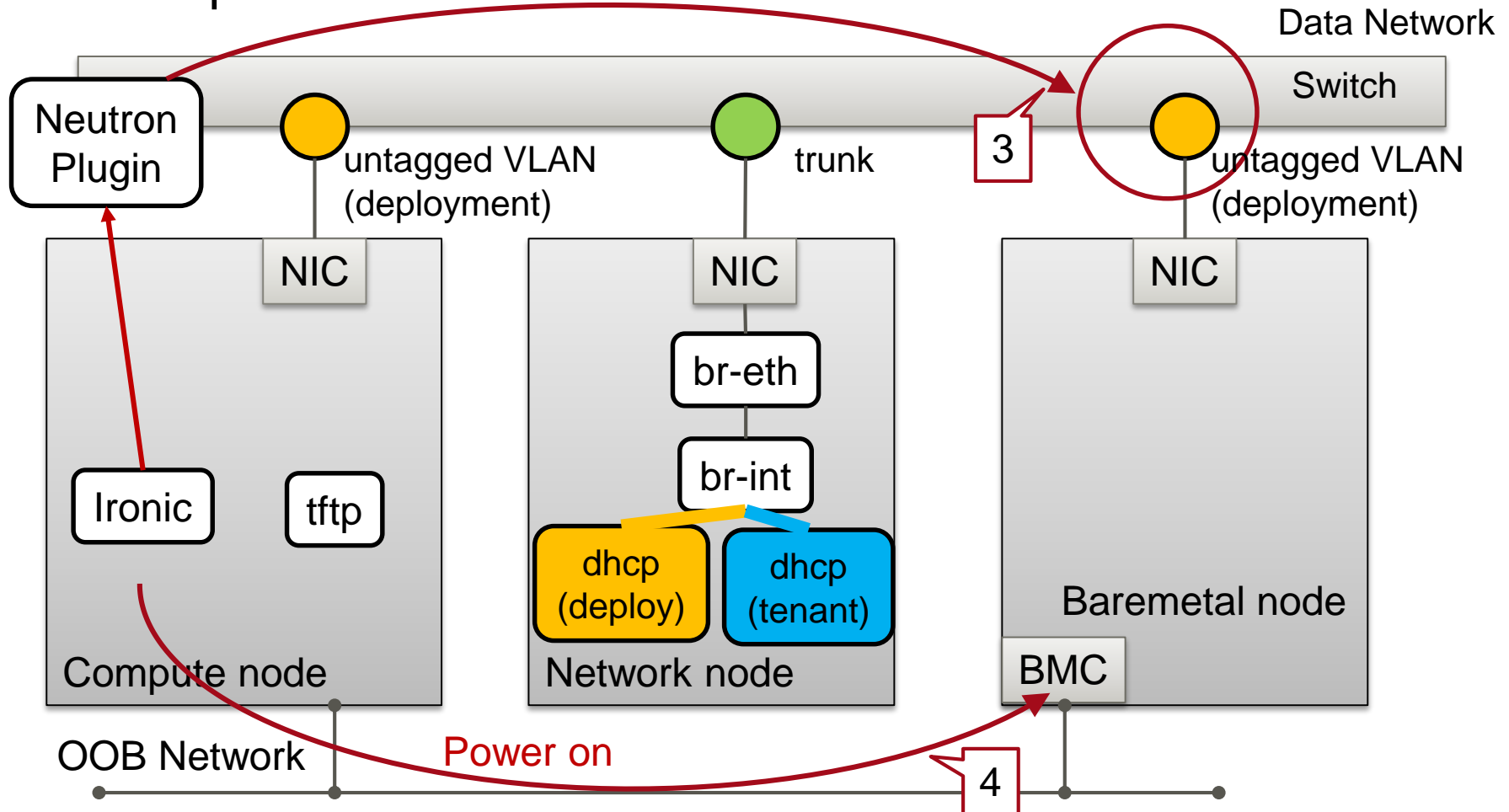
1. Tenant user creates a network (tenant VLAN)  
Then, Neutron creates a DHCP server for the network
2. The user requests baremetal provisioning on the network





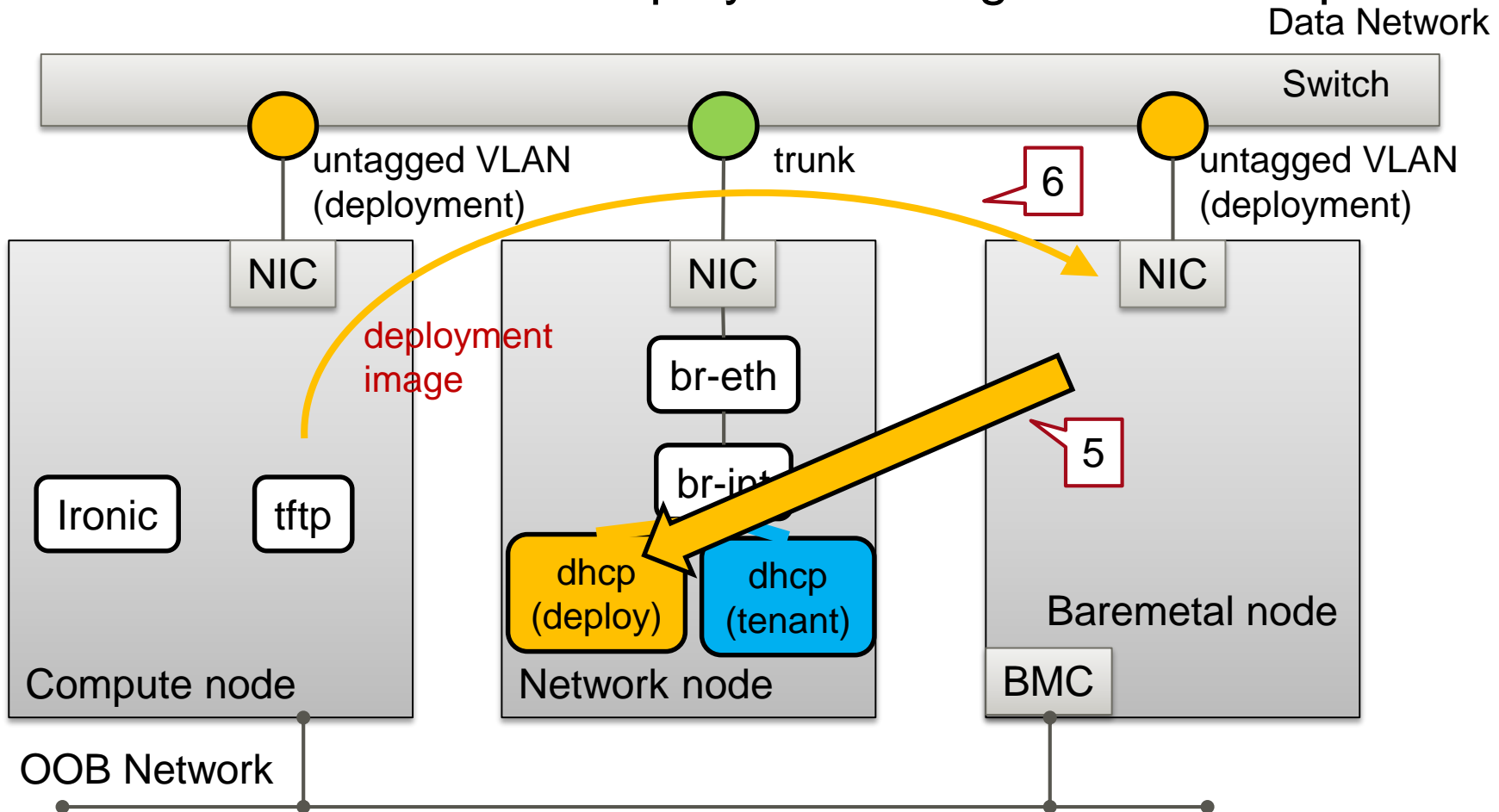
# Deployment flow (2/6)

3. Ironic configures an untagged VLAN ID of deployment VLAN to the port connected to the baremetal node
4. Ironic powers on the baremetal node



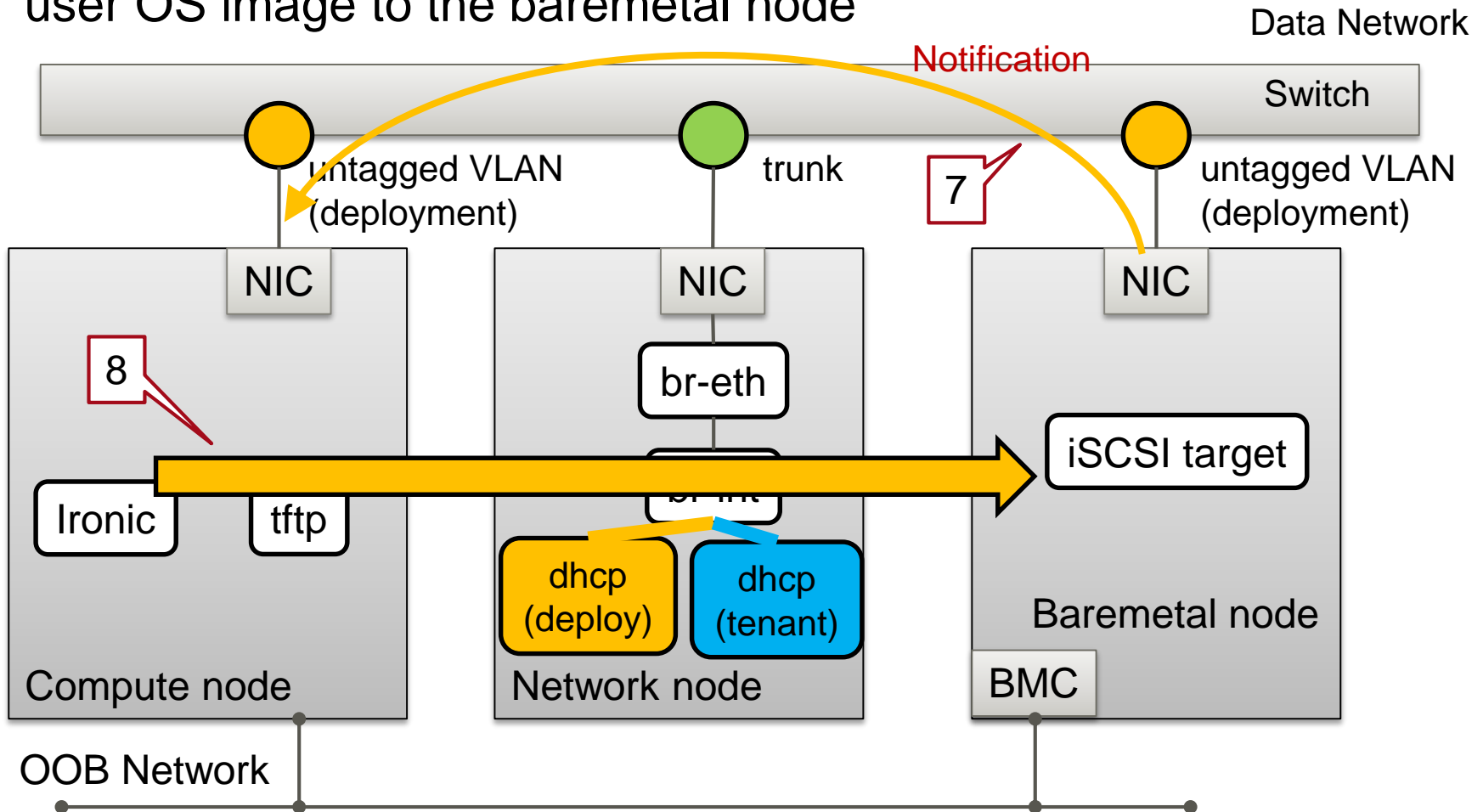
# Deployment flow (3/6)

- 5. The baremetal node gets an IP address from the DHCP server on the deployment VLAN
- 6. The baremetal loads a deployment image from the tftp server



# Deployment flow (4/6)

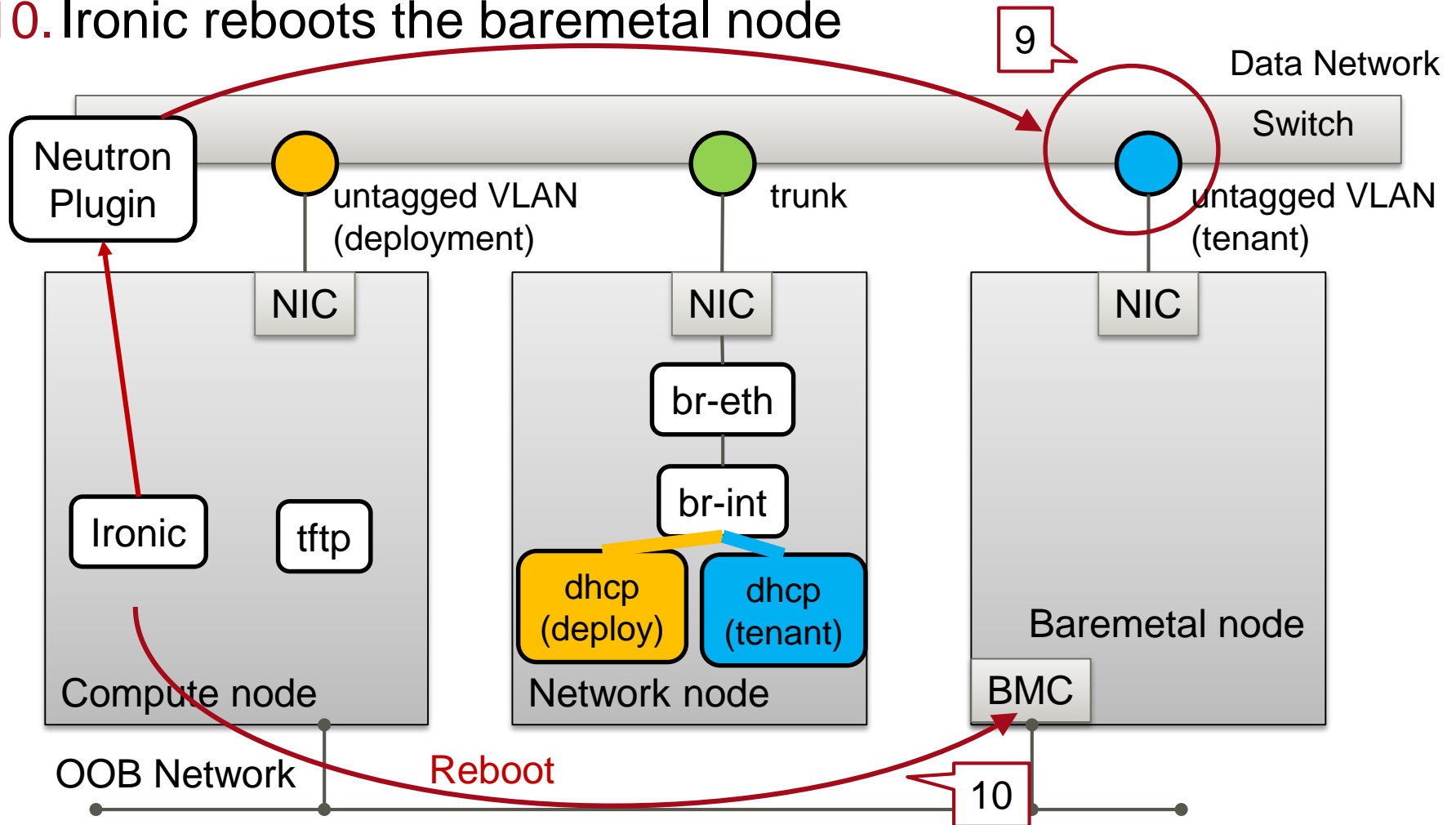
7. The deployment image prepares an iSCSI target and sends a notification to Ironic
8. Receiving notification from the baremetal node, Ironic copies a user OS image to the baremetal node



# Deployment flow (5/6)

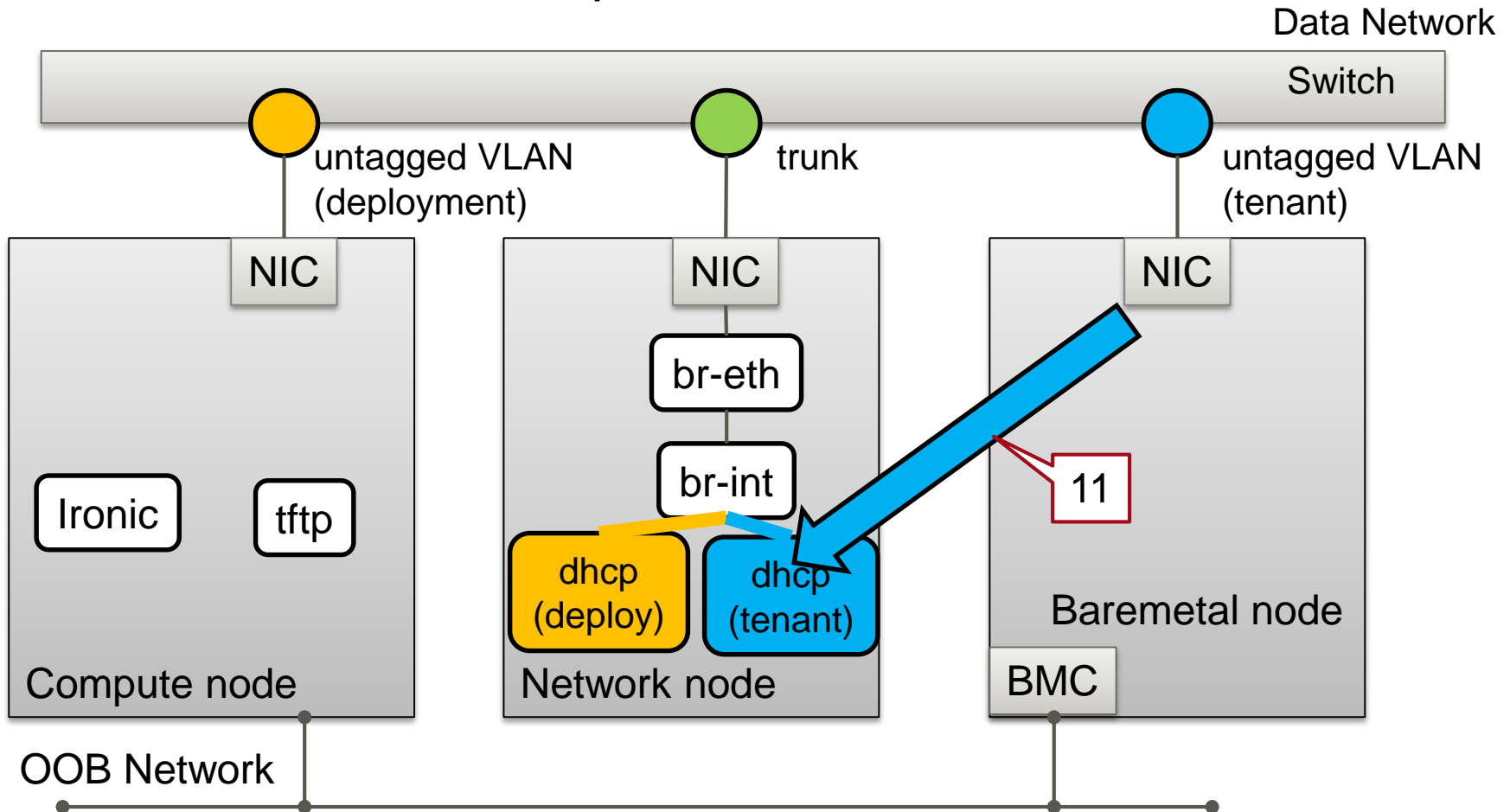
9. Ironic configures an untagged VLAN ID of the tenant VLAN to the port connected to the baremetal node

10. Ironic reboots the baremetal node




# Deployment flow (6/6)

- 11. After rebooted, the baremetal node gets an IP address from DHCP server on the tenant VLAN
- 12. The baremetal node is provided to the user



- OpenStack Summit was held
  - May 18–22 in Vancouver
  
- The issue was discussed at Design Summit
  - Neutron/Ironic integration
  
- We confirmed our solution does not conflict with the approach of the community
  
  
- We will contribute to the community by discussing the design and implementing our plugin
  - We proposed a blueprint  
<https://blueprints.launchpad.net/neutron/+spec/fujitsu-ism-ml2-mechanism-driver>



**FUJITSU**

shaping tomorrow with you