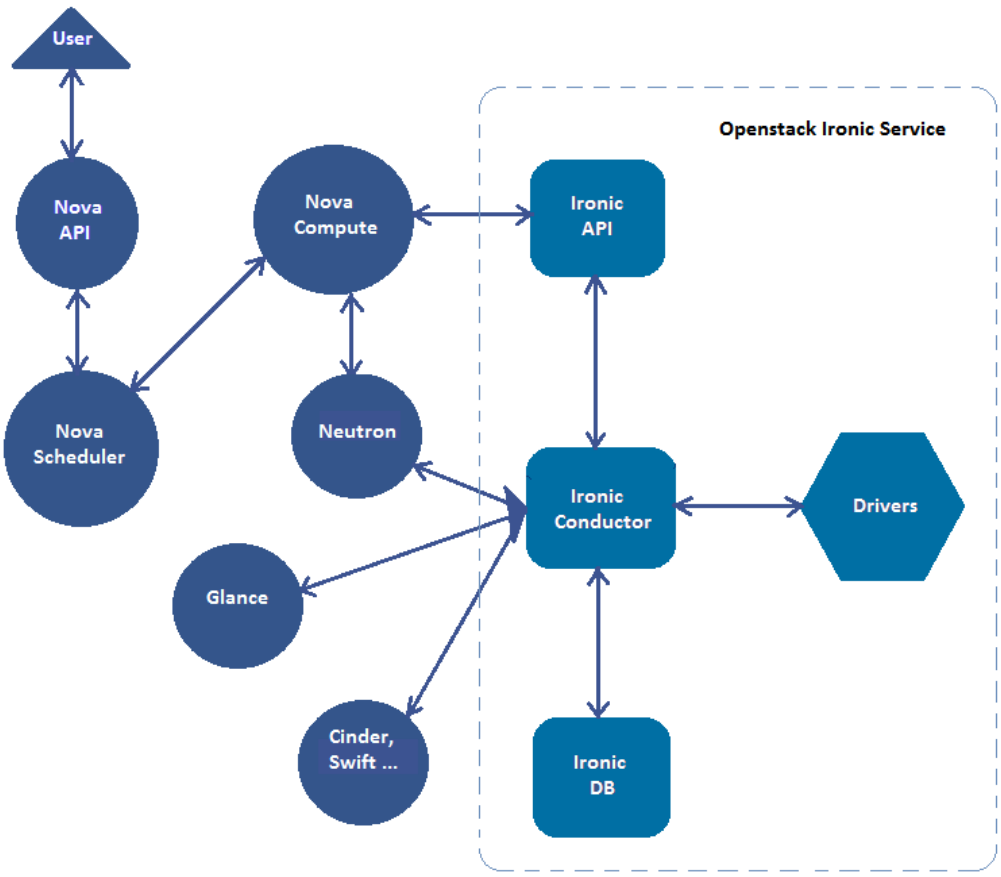
목차

Ironic 베어메탈 인스턴스의 배포 프로세스 3
논리 아키텍처 3
배포 아키텍처 3
전제 조건 4
배포 프로세스 4
iSCSI 배포 UML 6
직접 배포 UML 8
참조링크 12

Ironic 베어메탈 인스턴스의 배포 프로세스

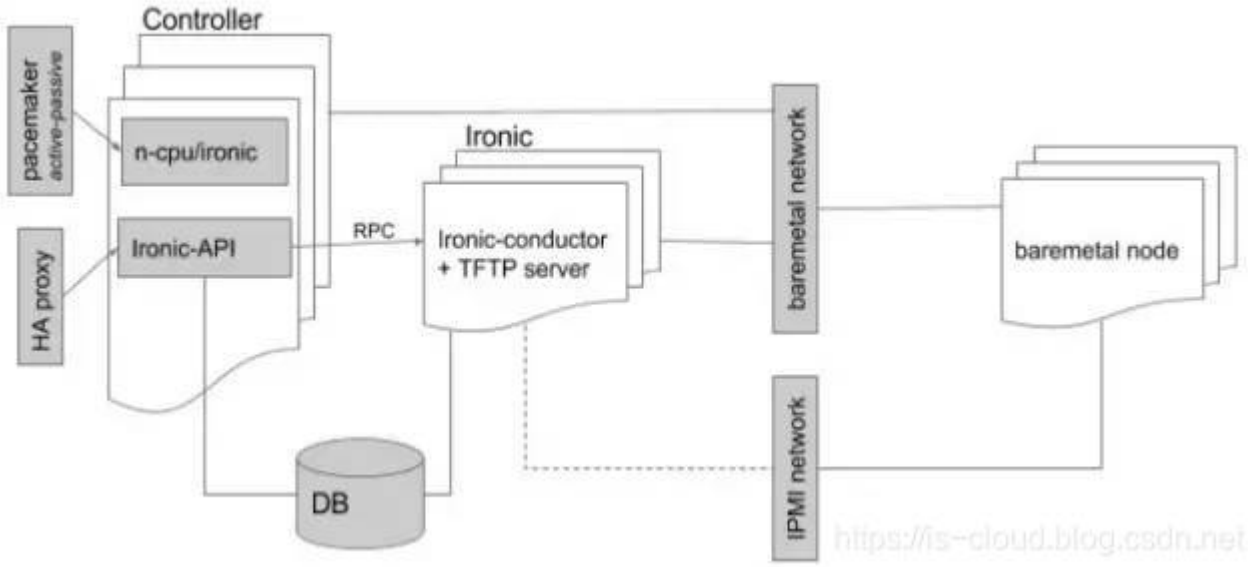
논리 아키텍처

Figure 1.2. Logical Architecture



https://is-cloud.blog.csdn.net

배포 아키텍처

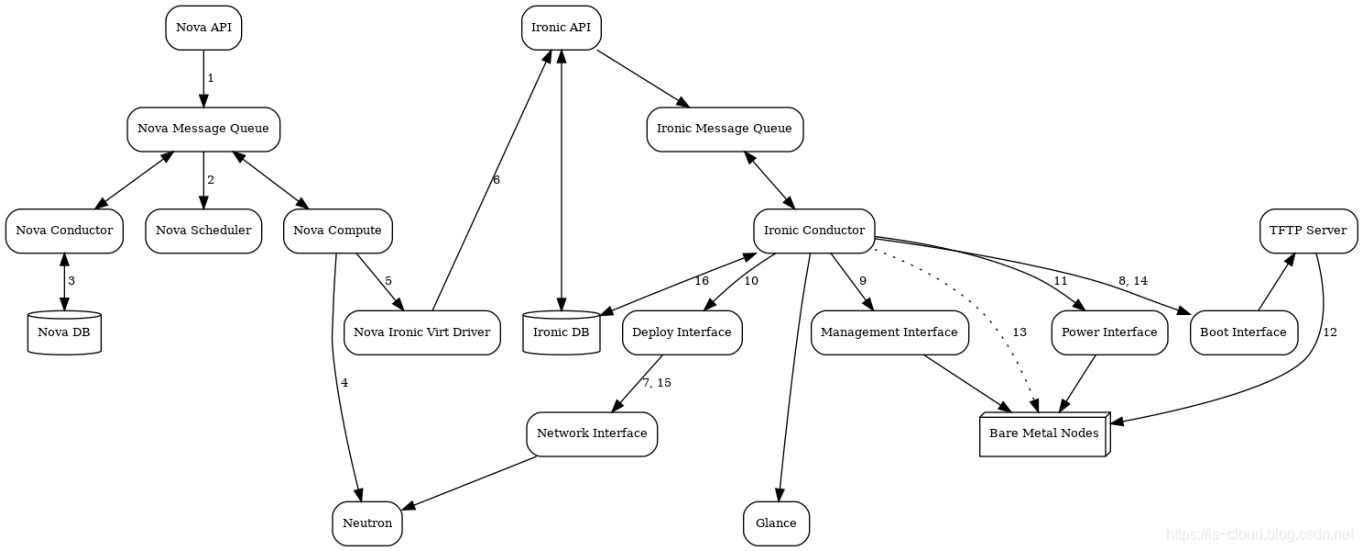


전제 조건

공식 문서: https://docs.openstack.org/ironic/latest/user/index.html

- 베어메탈 프로비저닝을 위해 `ironic-conductor`가 실행되는 베어메탈 서비스 노드에서 `tftp-server`, `ipmi`, `syslinux` 등의 종속 패키지를 구성해야 합니다.
- Nova는 베어메탈 서비스 엔드포인트를 사용하도록 구성되어야 하며, Nova 컴퓨트 노드에서 Ironic 드라이버를 사용하도록 컴퓨트 드라이버가 구성되어야 합니다.
- 사용 가능한 하드웨어를 위한 Flavors를 생성해야 합니다. Nova는 어떤 Flavor로 부팅할지 알아야 합니다.
- 이미지는 Glance에 준비되어야 합니다. 성공적인 베어메탈 배포에 필요한 이미지 유형은 다음과 같습니다:
    - bm-deploy-kernel
    - bm-deploy-ramdisk
    - user-image
    - user-image-vmlinuz
    - user-image-initrd
- 하드웨어는 Ironic RESTful API 서비스를 통해 등록되어야 합니다.

배포 프로세스



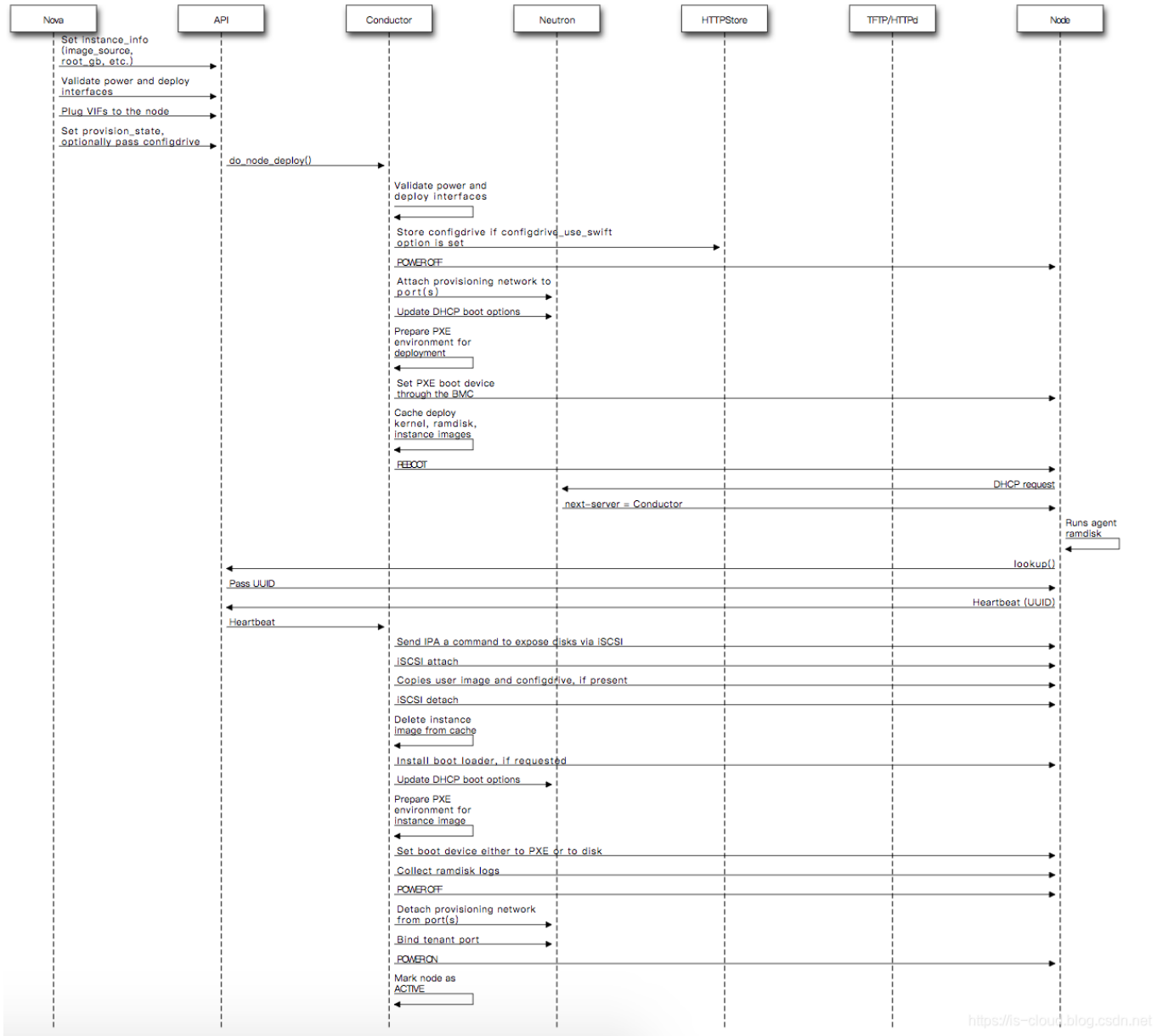
1. 부팅 인스턴스 요청은 Nova API를 통해 메시지 큐로 전달되어 Nova 스케줄러에 도달합니다.
2. Nova 스케줄러는 필터를 적용해 적합한 하이퍼바이저를 찾고, Flavor의 `extra_specs`(예: `cpu_arch`)을 사용해 대상 물리 노드를 매칭합니다.
3. Nova 컴퓨트 매니저는 선택된 하이퍼바이저의 자원을 할당합니다.
4. Nova 컴퓨트 매니저는 Nova 부팅 요청에서 요청된 네트워크 인터페이스에 따라 네트워킹 서비스에서 테넌트 가상 인터페이스(VIF)를 생성합니다. 여기서 포트의 MAC 주소는 랜덤으로 생성되며, VIF가 노드에 연결될 때 해당 노드의 네트워크 인터페이스 카드(NIC) 또는 본드의 MAC으로 업데이트됩니다.
5. Nova 컴퓨트는 부팅할 이미지 등의 정보를 포함한 spawn 작업을 생성합니다. 이 작업은 Nova 컴퓨트의 virt 계층에서 `driver.spawn`을 호출합니다. spawn 프로세스 동안 virt 드라이버는 다음 작업을 수행합니다:
   - 배포 이미지, 인스턴스 UUID, 요청된 기능 및 다양한 Flavor 속성 정보를 포함해 대상 Ironic 노드를 업데이트합니다.
   - Ironic API를 호출해 노드의 전원 및 배포 인터페이스를 검증합니다.
   - 이전에 생성된 VIF를 노드에 연결합니다. 각 Neutron 포트는 어떤 Ironic 포트나 포트 그룹에도 연결될 수 있으며, 포트 그룹이 포트보다 우선순위가 높습니다. Ironic 측에서는 네트워크 인터페이스가 이 작업을 수행하며, 여기서의 연결은 VIF 식별자를 Ironic 포트 또는 포트 그룹에 저장하고, VIF MAC을 해당 포트 또는 포트 그룹의 MAC으로 업데이트하는 것을 의미합니다.
   - 요청 시 config 드라이브를 생성합니다.
6. Nova의 Ironic virt 드라이버는 Ironic API를 통해 베어메탈 노드를 관리하는 Ironic Conductor에 배포 요청을 보냅니다.
7. 가상 인터페이스가 연결되고, Neutron API는 DHCP 포트를 업데이트해 PXE/TFTP 옵션을 설정합니다. Neutron 네트워크 인터페이스를 사용하는 경우 Ironic은 네트워킹 서비스에 별도의 프로비저닝 포트를 생성하며, flat 네트워크 인터페이스를 사용하는 경우 Nova에서 생성된 포트를 프로비저닝 및 배포된 인스턴스 네트워킹 모두에 사용합니다.
8. Ironic 노드의 부팅 인터페이스는 (i)PXE 구성을 준비하고 배포 커널과 램디스크를 캐시합니다.
9. Ironic 노드의 관리 인터페이스는 노드의 네트워크 부팅을 활성화하는 명령을 실행합니다.
10. Ironic 노드의 배포 인터페이스는 인스턴스 이미지를 캐시하고(iscsi 배포 인터페이스의 경우), 필요한 경우 커널과 램디스크도 캐시합니다(예: netboot의 경우).
11. Ironic 노드의 전원 인터페이스는 노드를 켜는 명령을 실행합니다.
12. 노드는 배포 램디스크를 부팅합니다.
13. 사용된 드라이버에 따라, conductor가 iSCSI를 통해 이미지를 물리 노드로 복사하거나(iSCSI 배포), 배포 램디스크가 임시 URL에서 이미지를 다운로드합니다(직접 배포). 임시 URL은 Swift API 호환 객체 저장소(예: Swift 또는 RadosGW)를 통해 생성할 수 있습니다. 이미지 배포가 완료됩니다.
14. 노드의 부팅 인터페이스는 PXE 구성을 인스턴스 이미지로 전환하거나(로컬 부팅의 경우 부팅 장

치를 디스크로 설정), 램디스크 에이전트에 노드를 소프트 파워 오프하도록 요청합니다. 램디스크 에이전트의 소프트 파워 오프가 실패하면, 베어메탈 노드는 IPMI/BMC 호출을 통해 전원이 꺼집니다.
15. 배포 인터페이스는 프로비저닝 포트를 제거하고 테넌트 포트를 노드에 바인딩합니다(아직 바인딩되지 않은 경우). 그런 다음 노드를 켭니다.
16. 베어메탈 노드의 프로비저닝 상태가 `active`로 업데이트됩니다.

iSCSI 배포 UML

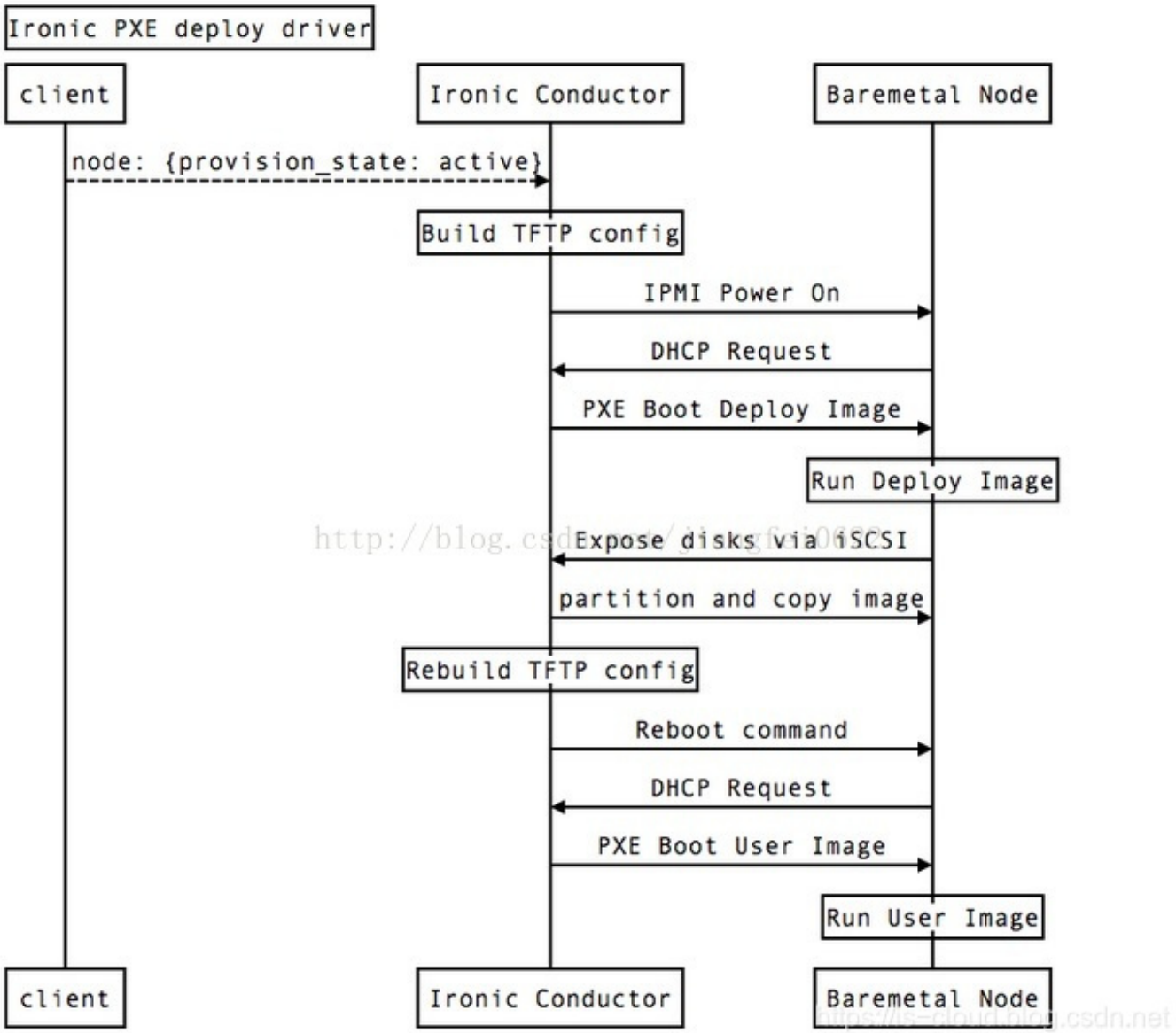
iSCSI 배포 인터페이스를 사용하는 경우, 배포 램디스크는 노드의 하드 드라이브를 iSCSI 공유로 공개합니다. 그런 다음, ironic-conductor가 이미지를 이 공유로 복사합니다. 이 인터페이스는 기본적으로 사용됩니다.

```
openstack baremetal node create --driver ipmi --deploy-interface iscsi
openstack baremetal node set <NODE> --deploy-interface iscsi
```



1. Nova API를 통해 베어메탈 인스턴스 배포 요청을 발행합니다.
2. Nova 스케줄러는 요청에 포함된 파라미터 정보(예: 하드웨어 템플릿, 이미지 등)를 바탕으로 적합한 Ironic 노드를 필터링합니다. 예를 들어, 스케줄링 요소는 보통 Flavor의 `extra_spec` 속성에 포함됩니다(예: `cpu_arch`, `baremetal:deploy_kernel_id`, `baremetal:deploy_ramdisk_id`).
3. Nova Compute는 베어메탈 배포에 필요한 정보를 모두 정리한 후, Build 작업을 `spawn`합니다. 이 작업은 Ironic API를 호출하고 매개변수를 전달하여 실제로 실행되며, Ironic은 이 작업에 필요한 하드웨어 자원 정보를 데이터베이스에 지속적으로 저장합니다.
4. Ironic은 OpenStack의 다른 서비스와 상호작용하여 Glance 서비스에서 이미지(Deploy Images 및 User Images)를 가져오고, Neutron 서비스를 호출해 베어메탈을 위한 포트를 생성하며, Cinder 서비스를 호출해 볼륨을 가져옵니다.
5. Ironic은 실제 베어메탈 배포를 시작하며, PXE 드라이버는 TFTP와 부트로더를 준비하고 IPMI 드라이버는 베어메탈의 부팅 모드를 PXE로 설정한 후 전원을 켭니다.
6. 베어메탈이 전원이 켜지면 DHCP를 통해 Ironic Conductor의 주소를 얻고, TFTP를 통해 Conductor에서 Deploy Images를 가져오려고 시도합니다. Conductor가 RAMDisk를 배포한 후, IPA는 iSCSI 프로토콜을 통해 베어메탈의 하드 디스크를 노출하고, Conductor는 User Images를 베어메탈 디스크에 주입하여 루트 디스크로 사용합니다.
7. 루트 디스크 배포가 완료된 후, IPMI 드라이버는 베어메탈의 부팅 순서를 조정하고 배포를 완료합니다.

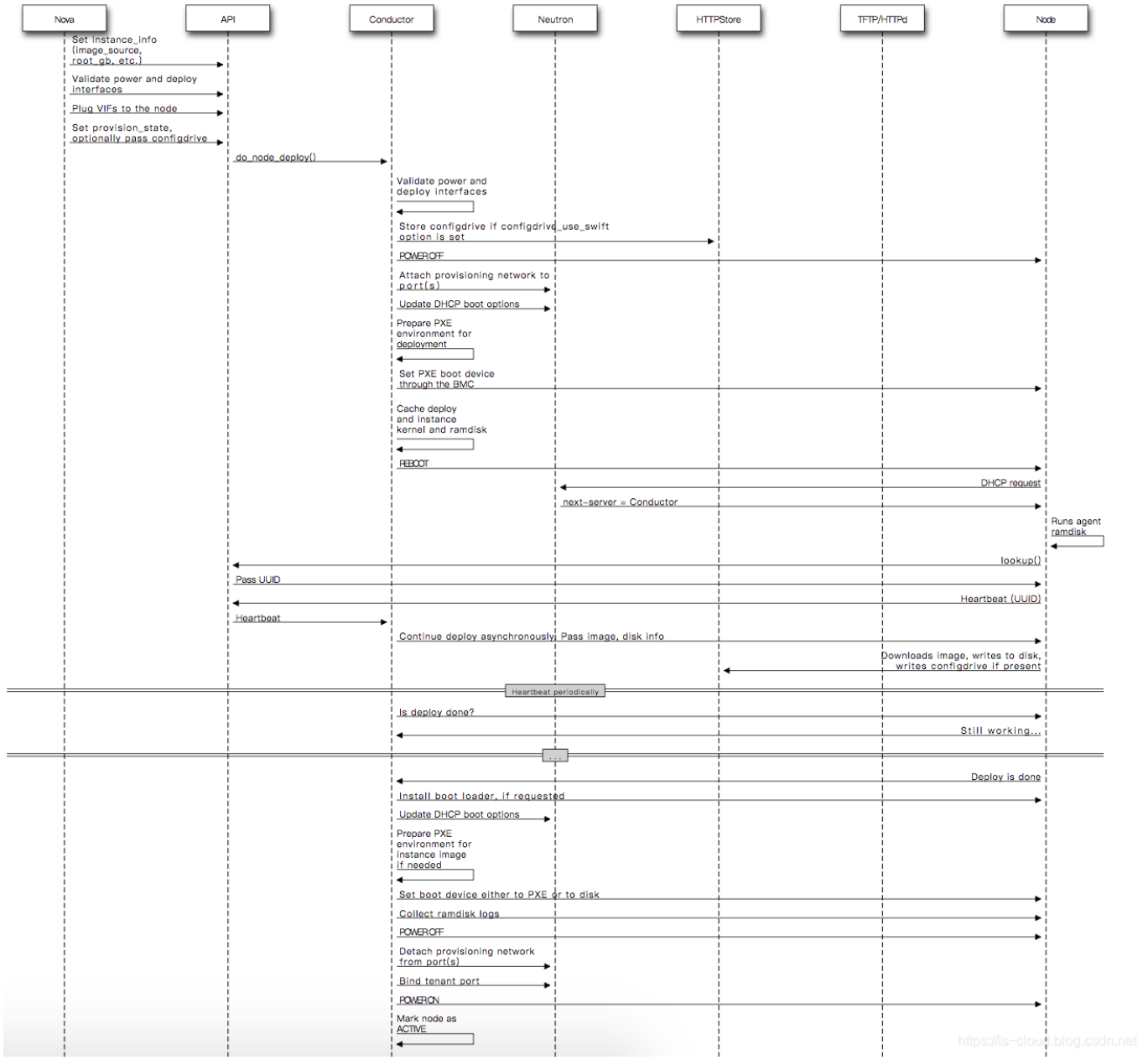
PXE 배포 드라이버

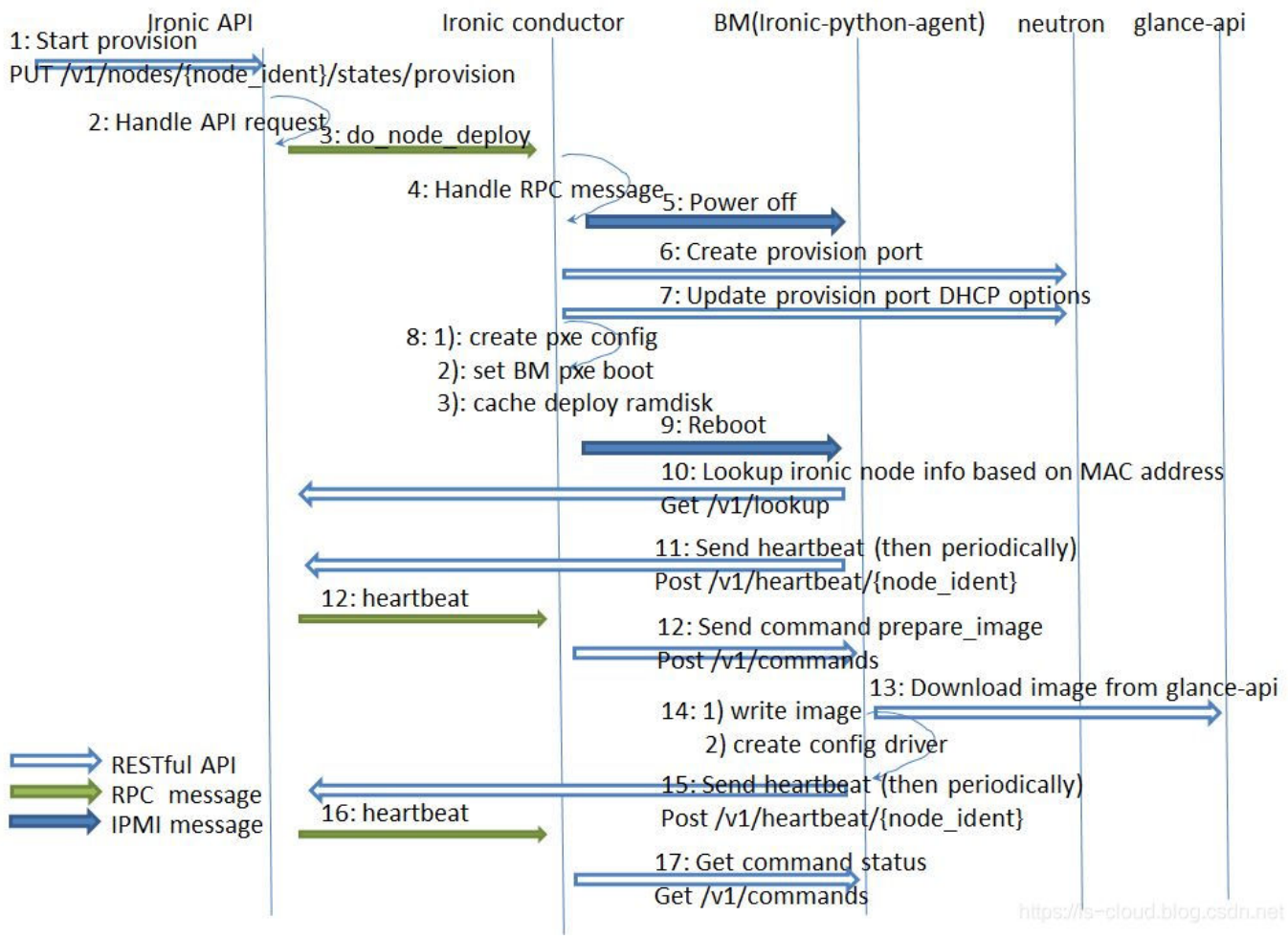


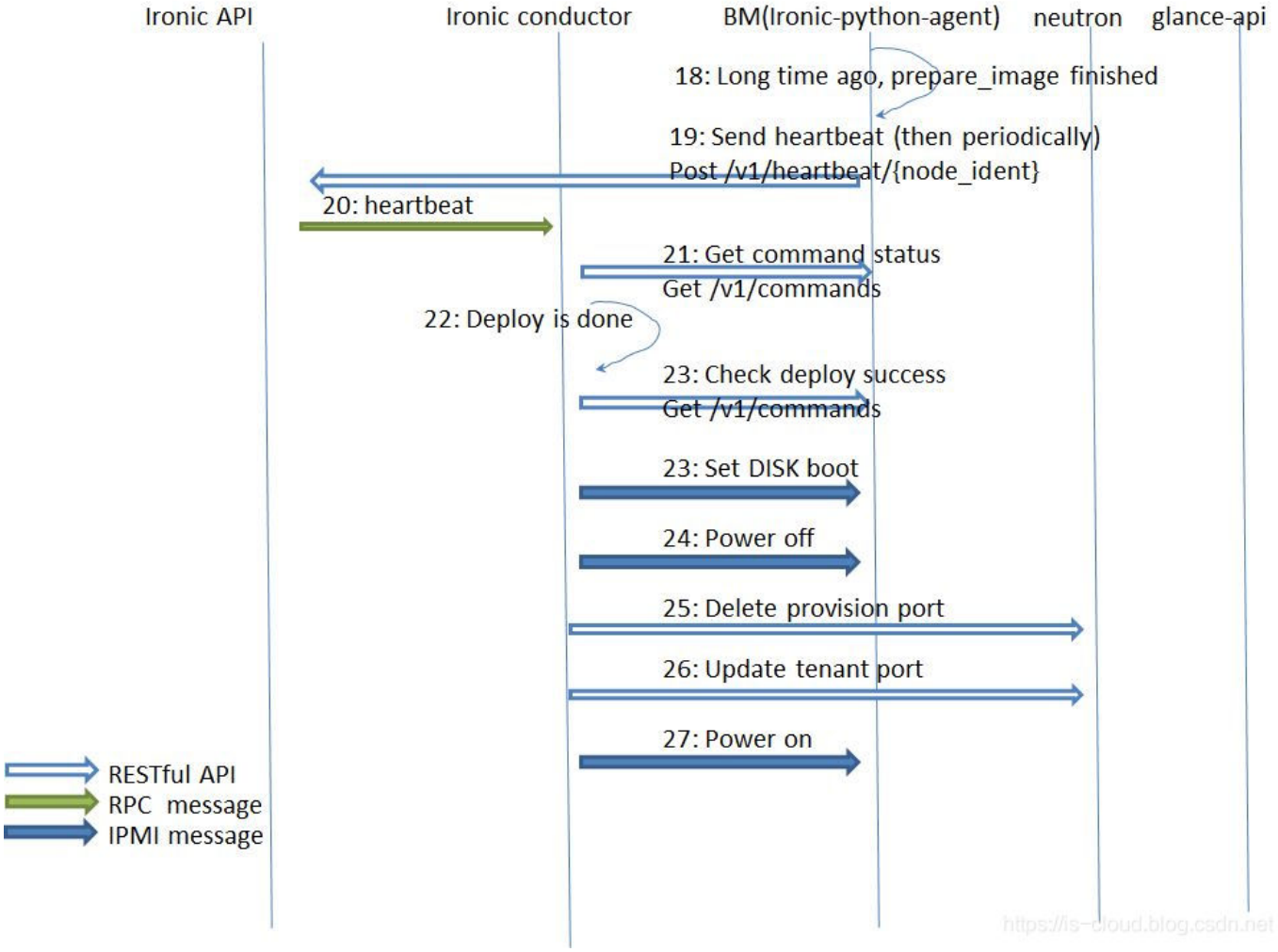
직접 배포 UML

Direct 배포 인터페이스를 사용할 경우, 배포 램디스크는 이미지를 HTTP 위치에서 가져옵니다. 이 HTTP 위치는 객체 저장소(Swift 또는 RadosGW)의 임시 URL일 수도 있고, 사용자가 제공한 HTTP URL일 수도 있습니다. 배포 램디스크는 이후 이미지를 대상 디스크로 복사합니다.

```
openstack baremetal node create --driver ipmi --deploy-interface direct
openstack baremetal node set <NODE> --deploy-interface direct
```

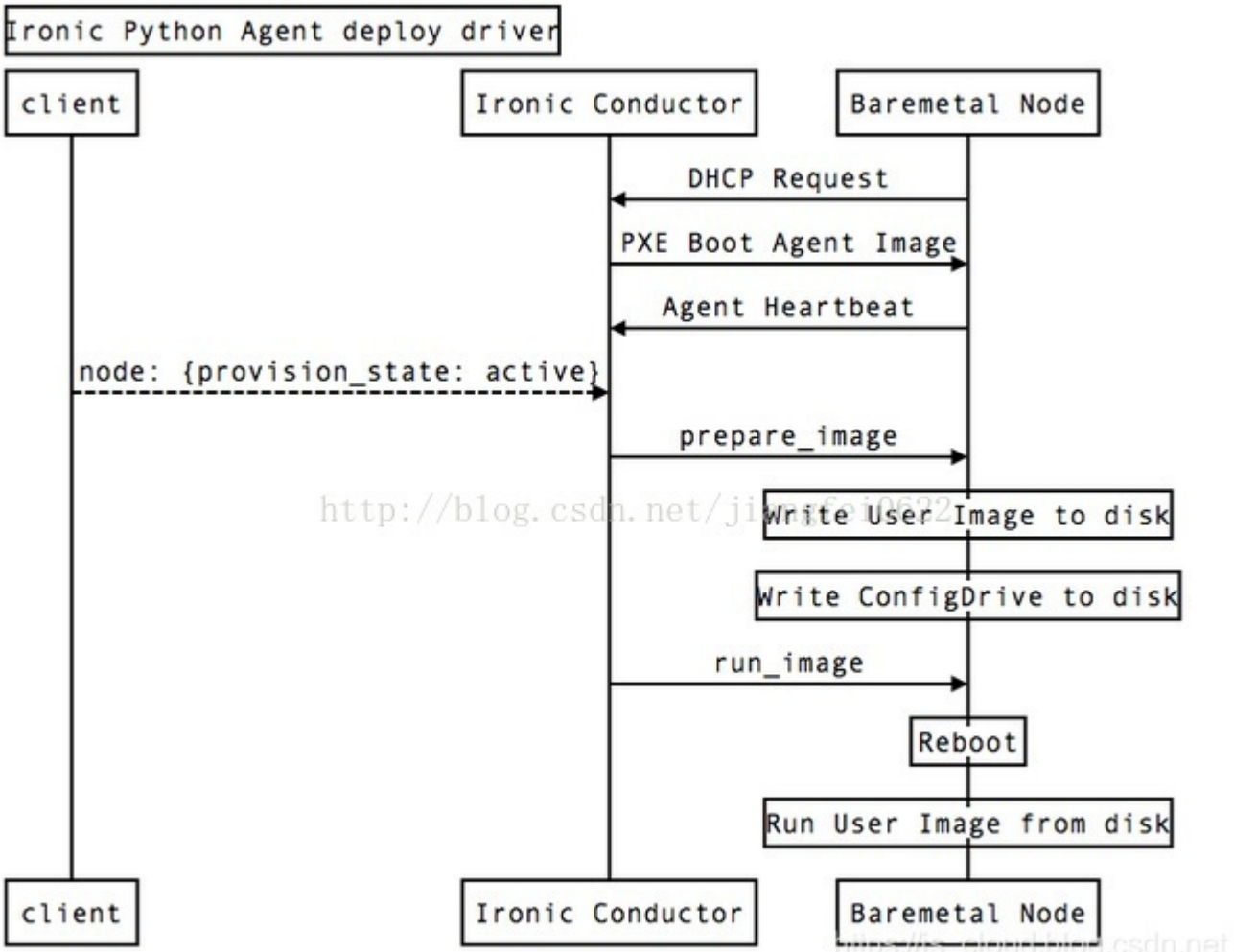

Nova
API
Conductor
Neutron
HTTPStore
TFTP/HTTPd
Node
Set instance_info (image_source, root_gb, etc.)
Validate power and deploy interfaces
Plug VIFs to the node
Set provision_state, optionally pass configdrive
do_node_deploy()
Validate power and deploy interfaces
Store configdrive if configdrive_use_swift option is set
POWER OFF
Attach provisioning network to port(s)
Update DHCP boot options
Prepare PXE environment for deployment
Set PXE boot device through the BMC
Cache deploy and instance kernel and ramdisk
REBOOT
DHCP request
next-server = Conductor
Runs agent ramdisk
lookup()
Pass UUID
Heartbeat (UUID)
Heartbeat
Continue deploy asynchronously. Pass image, disk info
Downloads image, writes to disk, writes configdrive if present
Heartbeat periodically
Is deploy done?
Still working...
...
Deploy is done
Install boot loader, if requested
Update DHCP boot options
Prepare PXE environment for instance image if needed
Set boot device either to PXE or to disk
Collect ramdisk logs
POWER OFF
Detach provisioning network from port(s)
Bind tenant port
POWER ON
Mark node as ACTIVE


Jronic API
Ironic conductor
BM(Ironic-python-agent)
neutron
glance-api
1: Start provision
PUT /v1/nodes/{node_ident}/states/provision
2: Handle API request
3: do_node_deploy
4: Handle RPC message
5: Power off
6: Create provision port
7: Update provision port DHCP options
8: 1): create pxe config
2): set BM pxe boot
3): cache deploy ramdisk
9: Reboot
10: Lookup ironic node info based on MAC address
Get /v1/lookup
11: Send heartbeat (then periodically)
Post /v1/heartbeat/{node_ident}
12: heartbeat
12: Send command prepare_image
Post /v1/commands
13: Download image from glance-api
14: 1) write image
2) create config driver
15: Send heartbeat (then periodically)
Post /v1/heartbeat/{node_ident}
16: heartbeat
17: Get command status
Get /v1/commands
RESTful API
RPC message
IPMI message



1. Provision 네트워크에 연결합니다.
2. 베어메탈 서버를 재부팅합니다.
3. IPA는 `lookup`을 통해 Ironic Conductor에서 해당하는 `ironic_node_uuid`를 가져옵니다.
4. IPA는 `Heartbeat`를 통해 Ironic과 통신하며, 이 과정에서 배포 프로세스가 시작되고, Ironic Conductor가 `prepare_image` 명령을 IPA로 전송합니다. 그러면 IPA는 직접(Direct)으로 사용자 이미지를 다운로드하여 로컬 디스크에 주입합니다. 이미지 쓰기는 시간이 오래 걸리는 작업으로, 보통 300GB 크기의 이미지를 쓰는 데 약 10분 정도 걸립니다.
5. `ironic-conductor`는 `Heartbeat` 메시지를 받은 후(이 메시지는 `ironic-api`로부터 RPC로 전달됩니다), 현재 상태에 따라 적절한 처리를 진행합니다. 첫 번째 `Heartbeat` 메시지는 `ironic-conductor`가 IPA로 `prepare_image` 명령을 내리게 하며, 이후의 `Heartbeat` 메시지는 `ironic-conductor`가 IPA의 명령 상태를 조회하도록 하여 `prepare_image` 등의 명령이 아직 실행 중인지 확인합니다. 명령 상태에서 작업이 완료되었다고 나타나면, 다음 배포 단계로 진행하여 물리 서버의 부팅 장치를 디스크로 설정하고, 서버를 종료한 후 네트워크를 전환하고, 다시 서버를 부팅합니다. 이 과정을 통해 배포가 완료됩니다.

IPA 배포 드라이버



참조링크

- https://www.cnblogs.com/jmilkfan-fanguiju/p/11825062.html

From:
https://atl.kr/dokuwiki/ - **AllThatLinux!**

Permanent link:
https://atl.kr/dokuwiki/doku.php/ironic_%EB%B2%A0%EC%96%B4%EB%A9%94%ED%83%88_%EC%9D%B8%EC%8A%A4%ED%84%B4%EC%8A%A4%EC%9D%98_%EB%B0%B0%ED%8F%AC_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4?rev=1728628218

Last update: **2024/10/11 06:30**