



Red Hat OpenStack Platform 17.1

Red Hat OpenStack Platform 강화

모범 사례, 규정 준수 및 보안 강화

Red Hat OpenStack Platform 17.1 Red Hat OpenStack Platform 강화

모범 사례, 규정 준수 및 보안 강화

OpenStack Team
rhos-docs@redhat.com

법적 공지

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

초록

이 가이드에서는 Red Hat OpenStack Platform 환경의 보안을 강화하는 방법에 대한 모범 사례 및 개념 정보를 제공합니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체	6
RED HAT 문서에 관한 피드백 제공	7
1장. 보안 소개	8
1.1. RED HAT OPENSTACK PLATFORM 보안	8
1.2. RED HAT OPENSTACK PLATFORM 관리자 역할 이해	8
1.3. RED HAT OPENSTACK PLATFORM의 보안 영역 식별	9
1.4. RED HAT OPENSTACK PLATFORM에서 보안 영역 찾기	9
1.5. 보안 영역 연결	10
1.6. 위협 완화	11
2장. 보안 개선 사항	12
2.1. 보안 ROOT 사용자 액세스 사용	12
2.2. 오버클라우드 방화벽에 서비스 추가	12
2.3. 오버클라우드 방화벽에서 서비스 삭제	13
2.4. SNMP(SIMPLE NETWORK MANAGEMENT PROTOCOL) 문자열 변경	14
2.5. OPEN VSWITCH 방화벽 사용	15
3장. RHOSP 환경 기록	17
3.1. 시스템 역할 기록	17
3.2. 하드웨어 인벤토리 생성	18
3.3. 소프트웨어 인벤토리 생성	19
4장. ID 및 액세스 관리	21
4.1. RED HAT OPENSTACK PLATFORM FERNET 토큰	21
4.2. OPENSTACK ID 서비스 엔티티	21
4.3. KEYSTONE으로 인증	21
4.4. ID 서비스 HEAT 매개변수를 사용하여 잘못된 로그인 시도를 중지	22
4.5. 외부 ID 공급자로 인증	23
5장. TLS 및 PKI를 사용하여 RED HAT OPENSTACK 배포 보안	25
5.1. PKI(PUBLIC KEY INFRASTRUCTURE)의 구성 요소	25
5.2. 인증 기관 요구 사항 및 권장 사항	25
5.3. 사용자 환경에서 TLS 버전 확인	26
5.4. OPENSTACK에 대한 IDM(IDENTITY MANAGEMENT) 서버 권장 사항	28
5.5. ANSIBLE을 사용하여 TLS-E 구현	29
5.6. TRIPLEO-IPA 매개변수	31
5.7. 모든 위치에서 TLS(TLS-E)에서 MEMCACHED 트래픽 암호화	32
5.8. 개인 키의 크기 증가	33
5.9. RED HAT OPENSTACK PLATFORM의 IDM 서버를 복제본으로 교체	33
6장. 사용자 지정 SSL/TLS 인증서 설정	35
6.1. 서명 호스트 초기화	35
6.2. 인증 기관 생성	35
6.3. 클라이언트에 인증 기관 추가	36
6.4. SSL/TLS 키 생성	36
6.5. SSL/TLS 인증서 서명 요청 생성	36
6.6. SSL/TLS 인증서 생성	37
6.7. 언더클라우드에 인증서 추가	38
7장. 오버클라우드 공용 끝점에서 SSL/TLS 활성화	40
7.1. SSL/TLS 활성화	40

7.2. 루트 인증서 삽입	41
7.3. DNS 끝점 구성	42
7.4. 오버클라우드 생성 중 환경 파일 추가	43
7.5. 수동으로 SSL/TLS 인증서 업데이트	43
8장. 오버클라우드의 암호화를 위해 CRYOSTAT 키 사용	45
8.1. CRYOSTAT 배포 검토	45
9장. RED HAT OPENSTACK PLATFORM의 연방 정보 처리 표준	47
9.1. FIPS 활성화	47
10장. 사용자 액세스 보안 개선	49
10.1. SRBAC 가상 사용자	49
10.2. 보안 역할 기반 액세스 제어 활성화	50
10.3. RBAC 환경에서 역할 할당	50
11장. POLICIES	52
11.1. 기존 정책 검토	52
11.2. 서비스 정책 이해	52
11.3. 정책 구문	53
11.4. 액세스 제어에 정책 파일 사용	53
11.5. 예: 특성을 기반으로 액세스 제한	55
11.6. HEAT로 정책 수정	56
11.7. 사용자 및 역할 감사	57
11.8. API 액세스 감사	58
12장. 네트워크 시간 프로토콜	59
12.1. 일관된 시간이 중요한 이유	59
12.2. NTP 설계	59
13장. 인프라 및 가상화 강화	60
13.1. RED HAT OPENSTACK PLATFORM용 HARWARE	60
13.2. 클라우드 환경의 소프트웨어 업데이트	60
13.3. OPENSTACK 환경에서 SSH 키 업데이트	60
13.4. 하드웨어 및 소프트웨어 기능 제한	62
13.5. RED HAT OPENSTACK PLATFORM의 SELINUX	62
13.6. 컨테이너화된 서비스 조사	63
13.7. 컨테이너화된 서비스를 임시로 변경	63
13.8. 컨테이너화된 서비스를 영구적으로 변경	64
13.9. 펌웨어 업데이트	64
13.10. SSH 배너 텍스트 사용	65
13.11. 시스템 이벤트 감사	65
13.12. 방화벽 규칙 관리	65
13.13. AIDE를 사용한 침입 탐지	67
13.14. SECURETTY 검토	69
13.15. IDENTITY SERVICE에 대한 CRYOSTATF 감사	69
13.16. LOGIN.DEFS 값을 검토합니다.	70
14장. 대시보드 서비스 강화	71
14.1. 대시보드 서비스 디버깅	71
14.2. 도메인 이름 선택	71
14.3. ALLOWED_HOSTS 구성	71
14.4. 교차 사이트 스크립팅 (XSS)	72
14.5. CSRF(CROSS SITE REQUEST FORGERY)	72
14.6. IFRAME 포함 허용	72

14.7. 대시보드 트래픽에 HTTPS 암호화 사용	72
14.8. HSTS(HTTP STRICT TRANSPORT SECURITY)	73
14.9. 프론트 엔드 캐싱	73
14.10. 세션 백엔드	73
14.11. 시크릿 키 검토	73
14.12. 세션 쿠키 구성	74
14.13. 암호 복잡성 검증	74
14.14. 관리자 암호 확인 적용	74
14.15. 암호 표시 비활성화	75
14.16. 대시보드의 로그인 배너 표시	75
14.17. 파일 업로드 크기 제한	77
15장. 네트워킹 서비스 강화	78
15.1. 프로젝트 네트워크 서비스 워크플로	78
15.2. 네트워킹 리소스 정책 엔진	78
15.3. 보안 그룹	78
15.4. ARP 스누핑 완화	78
15.5. 인증을 위한 보안 프로토콜 사용	79
16장. RED HAT OPENSTACK PLATFORM에서 블록 스토리지 강화	80
16.1. 요청 본문의 최대 크기 설정	80
16.2. 볼륨 암호화 활성화	80
16.3. 볼륨 도핑	80
17장. 공유 파일 시스템 강화 (MANILA)	81
17.1. MANILA의 보안 고려 사항	81
17.2. MANILA의 네트워크 및 보안 모델	82
17.3. 백엔드 모드 공유	82
17.4. MANILA의 네트워킹 요구 사항	83
17.5. MANILA를 사용하는 보안 서비스	84
17.6. 보안 서비스 소개	84
17.7. 보안 서비스 관리	84
17.8. 액세스 제어 공유	85
17.9. 공유 유형 액세스 제어	87
17.10. POLICIES	88
18장. 오브젝트 스토리지	90
18.1. 네트워크 보안	90
18.2. 루트가 아닌 사용자로 서비스 실행	92
18.3. 파일 권한	92
18.4. 스토리지 서비스 보안	92
18.5. OBJECT STORAGE 계정 용어	92
18.6. 프록시 서비스 보안	93
18.7. HTTP 청취 포트	93
18.8. 로드 밸런서	93
18.9. 오브젝트 스토리지 인증	93
18.10. AT-REST SWIFT 오브젝트 암호화	93
18.11. 추가 항목	94
19장. 모니터링 및 로깅	95
19.1. 모니터링 인프라 강화	95
19.2. 모니터링할 이벤트의 예	95
20장. 프로젝트의 데이터 개인 정보	97
20.1. 데이터 레지던스	97

20.2. 데이터 처리	97
20.3. CINDER 볼륨 데이터 암호화	98
20.4. 이미지 서비스 지연 삭제 기능	99
20.5. 컴퓨팅 소프트웨어 삭제 기능	99
20.6. 베어 메탈 프로비저닝의 보안 강화	99
20.7. 하드웨어 식별	99
20.8. 데이터 암호화	99
20.9. 키 관리	101
21장. 인스턴스 보안 관리	102
21.1. 인스턴스에 엔트로피 제공	102
21.2. 노드에 인스턴스 예약	102
21.3. 신뢰할 수 있는 이미지 사용	103
21.4. 이미지 생성	103
21.5. 이미지 서명 확인	104
21.6. 인스턴스 마이그레이션	104
21.7. 모니터링, 경고 및 보고	105
21.8. 업데이트 및 패치	106
21.9. 방화벽 및 인스턴스 프로필	106
21.10. 보안 그룹	106
21.11. 인스턴스 콘솔에 액세스	107
21.12. 인증서 삽입	107
22장. 메시지 대기열	108
22.1. 메시징 전송 보안	108
22.2. 대기열 인증 및 액세스 제어	109
22.3. RABBITMQ의 OPENSTACK 서비스 구성	109
22.4. CRYOSTAT의 OPENSTACK 서비스 구성	109
22.5. 메시지 큐 프로세스 격리 및 정책	110
22.6. 네임스페이스	110
23장. RED HAT OPENSTACK PLATFORM에서 끝점 보안	111
23.1. 내부 API 통신	111
23.2. ID 서비스 카탈로그에서 내부 URL 구성	111
23.3. 내부 URL에 대한 애플리케이션 구성	111
23.4. 붙여넣기 및 미들웨어	111
23.5. 보안 METADEF API	112
23.6. 클라우드 사용자를 위한 METADEF API 액세스 활성화	112
23.7. HAPROXY에 대한 SSL/TLS 암호화 및 규칙 변경	113
23.8. 네트워크 정책	114
23.9. 필수 액세스 제어	114
23.10. API 끝점 속도 제한	114
24장. 페더레이션 구현	115
24.1. RED HAT SINGLE SIGN-ON을 사용하여 IDM과 통합	115
24.2. 페더레이션 워크플로	115

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 [CTO Chris Wright의 메시지](#)를 참조하십시오.

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 어떻게 더 나은지 알려주십시오.

Jira에서 문서 피드백 제공

[Create Issue](#) 양식을 사용하여 OpenShift (RHOSO) 또는 이전 Red Hat OpenStack Platform (RHOSP)의 Red Hat OpenStack Services 문서에 대한 피드백을 제공합니다. RHOSO 또는 RHOSP 문서에 대한 문제를 생성할 때 RHOSO Jira 프로젝트에 문제가 기록되어 피드백의 진행 상황을 추적할 수 있습니다.

문제 생성 양식을 완료하려면 Jira에 로그인해야 합니다. Red Hat Jira 계정이 없는 경우 <https://issues.redhat.com>에서 계정을 생성할 수 있습니다.

1. 다음 링크를 클릭하여 **문제 생성** 페이지를 엽니다.
<https://issues.redhat.com/secure/CreateIssueDetails!init.jspx?pid=12336920&summary=Documentation%20feedback:%20%3CAdd%20summary%20here%3E&i<Include+the+documentation+URL,+the%20chapter+or+section+number,+and+a+detailed+descrip>
2. **요약** 및 **설명** 필드를 작성합니다. **설명** 필드에 문서 URL, 장 또는 섹션 번호, 문제에 대한 자세한 설명을 포함합니다. 양식의 다른 필드를 수정하지 마십시오.
3. **생성**을 클릭합니다.

1장. 보안 소개

RHOSP(Red Hat Openstack Platform)에 제공된 툴을 사용하여 계획 및 운영의 보안을 우선시하여 사용자의 개인 정보 보호 및 데이터 보안을 충족할 수 있습니다. 보안 표준을 구현하지 못하면 다운타임이나 데이터 유출이 발생할 수 있습니다. 사용 사례는 감사 및 규정 준수 프로세스를 전달해야 하는 법률의 대상이 될 수 있습니다.

- Ceph 강화에 대한 자세한 내용은 [데이터 보안 및 강화 가이드를 참조하십시오](#).



참고

이 가이드의 지침에 따라 환경 보안을 강화합니다. 그러나 이러한 권장 사항은 보안 또는 규정 준수를 보장하지 않습니다. 사용자 환경의 고유한 요구 사항의 보안을 평가해야 합니다.

1.1. RED HAT OPENSTACK PLATFORM 보안

기본적으로 RHOSP(Red Hat OpenStack Platform) director는 다음 툴 및 보안에 대한 액세스 제어를 사용하여 오버클라우드를 생성합니다.

SELinux

SELinux는 각 프로세스에 모든 작업에 대한 명시적 권한이 있어야 하는 액세스 제어를 제공하여 RHOSP에 대한 보안 개선 사항을 제공합니다.

Podman

컨테이너 툴로 Podman은 작동하는 루트 프로세스가 필요한 클라이언트/서버 모델을 사용하지 않기 때문에 RHOSP의 보안 옵션입니다.

시스템 액세스 제한

오버클라우드 배포 중에 director가 tripleo-admin에 생성하는 SSH 키 또는 오버클라우드에서 생성한 SSH 키를 사용하여 오버클라우드 노드에 로그인할 수 있습니다. 암호와 함께 SSH를 사용하여 오버클라우드 노드에 로그인하거나 root를 사용하여 오버클라우드 노드에 로그인할 수 없습니다.

조직의 요구 사항 및 신뢰 수준에 따라 다음과 같은 추가 보안 기능을 사용하여 director를 구성할 수 있습니다.

- 퍼블릭 TLS 및 TLS-everywhere
- OpenStack Key Manager(barbican)와 하드웨어 보안 모듈 통합
- 서명된 이미지 및 암호화된 볼륨
- 워크플로우 실행을 사용한 암호 및 fernet 키 교체

1.2. RED HAT OPENSTACK PLATFORM 관리자 역할 이해

사용자에게 **admin** 역할을 할당하면 이 사용자에게 모든 프로젝트의 리소스를 확인, 변경, 생성 또는 삭제할 수 있는 권한이 있습니다. 이 사용자는 공개적으로 사용 가능한 Glance 이미지 또는 공급자 네트워크와 같이 프로젝트 전체에서 액세스할 수 있는 공유 리소스를 생성할 수 있습니다. 또한 **admin** 역할이 있는 사용자는 사용자를 생성하거나 삭제하고 역할을 관리할 수 있습니다.

admin 역할을 사용자에게 할당하는 프로젝트는 **openstack** 명령이 실행되는 기본 프로젝트입니다. 예를 들어 **development** 라는 프로젝트의 **admin** 사용자가 다음 명령을 실행하면 **development** 프로젝트에 **internal-network** 라는 네트워크가 생성됩니다.

```
openstack network create internal-network
```

admin 사용자는 **--project** 매개변수를 사용하여 모든 프로젝트에서 **internal-network**를 생성할 수 있습니다.

```
openstack network create internal-network --project testing
```

1.3. RED HAT OPENSTACK PLATFORM의 보안 영역 식별

보안 영역은 일반적인 보안 문제를 공유하는 리소스, 애플리케이션, 네트워크 및 서버입니다. 일반적인 인증 및 권한 부여 요구 사항과 사용자를 갖도록 보안 영역을 설계합니다. 클라우드 아키텍처, 사용자 환경에 대한 허용 가능한 신뢰 수준, 조직의 표준화된 요구 사항에 따라 필요에 따라 자체 보안 영역을 정의할 수 있습니다. 영역과 신뢰 요구 사항은 클라우드 인스턴스가 퍼블릭, 프라이빗 또는 하이브리드인지에 따라 달라질 수 있습니다.

예를 들어 Red Hat OpenStack Platform의 기본 설치를 다음 영역으로 분할할 수 있습니다.

표 1.1. 보안 영역

영역	네트워크	세부 정보
퍼블릭	external	공용 영역은 인스턴스의 외부 연결을 위해 외부 네트워크, 공용 API 및 유동 IP 주소를 호스팅합니다. 이 영역을 사용하면 관리 제어 외부의 네트워크에서 액세스할 수 있으며 클라우드 인프라의 신뢰할 수 없는 영역입니다.
게스트	tenant	게스트 영역은 프로젝트 네트워크를 호스팅합니다. 인스턴스에 무제한 액세스를 허용하는 퍼블릭 및 프라이빗 클라우드 공급자의 경우 신뢰할 수 없습니다.
스토리지 액세스	storage, storage_mgmt	스토리지 액세스 영역은 스토리지 관리, 모니터링 및 클러스터링, 스토리지 트래픽용입니다.
제어	ctlplane, internal_api, ipmi	제어 영역에는 언더클라우드, 호스트 운영 체제, 서버 하드웨어, 물리적 네트워킹 및 Red Hat OpenStack Platform director 컨트롤 플레인도 포함됩니다.

1.4. RED HAT OPENSTACK PLATFORM에서 보안 영역 찾기

다음 명령을 실행하여 Red Hat OpenStack Platform 배포의 물리적 구성에 대한 정보를 수집합니다.

사전 요구 사항

- Red Hat OpenStack Platform 환경이 설치되어 있어야 합니다.

- stack으로 director에 로그인되어 있습니다.

프로세스

1. 소스 **stackrc**:

```
$ source /home/stack/stackrc
```

2. 할당된 ip 네트워크를 연결된 영역에 일치하도록 **openstack subnet list** 를 실행합니다.

```
openstack subnet list -c Name -c Subnet
+-----+-----+
| Name          | Subnet          |
+-----+-----+
| ctlplane-subnet | 192.168.101.0/24 |
| storage_mgmt_subnet | 172.16.105.0/24 |
| tenant_subnet   | 172.16.102.0/24 |
| external_subnet | 10.94.81.0/24   |
| internal_api_subnet | 172.16.103.0/24 |
| storage_subnet  | 172.16.104.0/24 |
+-----+-----+
```

3. **openstack server list** 를 실행하여 인프라의 물리적 서버를 나열합니다.

```
openstack server list -c Name -c Networks
+-----+-----+
| Name          | Networks          |
+-----+-----+
| overcloud-controller-0 | ctlplane=192.168.101.15 |
| overcloud-controller-1 | ctlplane=192.168.101.19 |
| overcloud-controller-2 | ctlplane=192.168.101.14 |
| overcloud-novacompute-0 | ctlplane=192.168.101.18 |
| overcloud-novacompute-2 | ctlplane=192.168.101.17 |
| overcloud-novacompute-1 | ctlplane=192.168.101.11 |
+-----+-----+
```

4. **openstack server list** 명령의 **ctlplane** 주소를 사용하여 물리적 노드의 구성을 쿼리합니다.

```
ssh tripleo-admin@192.168.101.15 ip addr
```

1.5. 보안 영역 연결

다양한 신뢰 수준 또는 인증 요구 사항으로 여러 보안 영역에 걸쳐 있는 모든 구성 요소를 신중하게 구성해야 합니다. 이러한 연결은 종종 네트워크 아키텍처에서 약한 지점입니다. 연결 중인 영역에서 가장 높은 신뢰 수준의 보안 요구 사항을 충족하도록 이러한 연결을 구성해야 합니다. 대부분의 경우 연결된 영역의 보안 제어는 공격의 가능성으로 인해 주요 관심사입니다. 구역이 발견되면 잠재적인 공격 지점이 추가되고 공격자가 배포의 더 민감한 부분으로 공격을 마이그레이션할 수 있는 기회를 추가합니다.

경우에 따라 OpenStack Operator는 통합 지점을 보유한 영역보다 더 높은 표준으로 보호하는 것을 고려할 수 있습니다. 위의 API 끝점 예제를 고려할 때 적자는 이러한 영역이 완전히 분리되지 않은 경우 관리 영역 내에서 내부 또는 관리자 API를 손상시키거나 액세스할 수 있기를 기대하며, 공용 영역의 공용 API 끝점을 대상으로 할 수 있습니다.

OpenStack의 설계는 보안 영역을 분리하기가 어렵습니다. 핵심 서비스는 일반적으로 두 개 이상의 영역에 확장되므로 보안 제어를 적용할 때 특별한 고려 사항이 제공되어야 합니다.

1.6. 위협 완화

대부분의 유형의 클라우드 배포, 퍼블릭, 프라이빗 또는 하이브리드는 일종의 보안 위협에 노출됩니다. 다음 사례는 보안 위협을 완화하는 데 도움이 됩니다.

- 최소 권한 원칙을 적용합니다.
- 내부 및 외부 인터페이스에서 암호화를 사용합니다.
- 중앙 집중식 ID 관리 사용.
- Red Hat OpenStack Platform을 계속 업데이트하십시오.

컴퓨팅 서비스는 악의적인 행위자에게 DDoS 및 무차별 공격용 도구를 제공할 수 있습니다. 예방 방법은 송신 보안 그룹, 트래픽 검사, 침입 탐지 시스템, 고객 교육 및 인식이 포함됩니다. 공용 네트워크에서 액세스하거나 인터넷과 같은 공용 네트워크에 액세스할 수 있는 배포의 경우 아웃바운드 오용을 탐지하고 해결하기 위해 프로세스 및 인프라가 제 위치에 있는지 확인합니다.

추가 리소스

- [Ansible을 사용하여 TLS-e 구현](#)
- [Red Hat IdM\(Identity Manager\)과 OpenStack Identity\(keystone\) 통합](#)
- [Red Hat OpenStack Platform의 마이너 업데이트 수행](#)

2장. 보안 개선 사항

다음 섹션에서는 오버클라우드 보안을 강화하기 위한 몇 가지 제안 사항을 제공합니다.

2.1. 보안 ROOT 사용자 액세스 사용

오버클라우드 이미지에는 **root** 사용자에게 대한 강화된 보안이 자동으로 포함됩니다. 예를 들어 배포된 각 오버클라우드 노드는 **root** 사용자에게 직접 SSH 액세스를 자동으로 비활성화합니다. 오버클라우드 노드의 **root** 사용자에게 계속 액세스할 수 있습니다. 각 오버클라우드 노드에는 **tripleo-admin** 사용자 계정이 있습니다. 이 사용자 계정에는 언더클라우드의 암호 없이 오버클라우드 노드로 SSH 액세스를 제공하는 언더클라우드 공용 SSH 키가 포함되어 있습니다.

사전 요구 사항

- Red Hat OpenStack Platform director 환경이 설치되어 있어야 합니다.
- stack으로 director에 로그인되어 있습니다.

프로세스

1. 언더클라우드 노드에서 SSH를 통해 **tripleo-admin** 사용자로 오버클라우드 노드에 로그인합니다.
2. **sudo -i** 를 사용하여 **root** 사용자로 전환합니다.

2.2. 오버클라우드 방화벽에 서비스 추가

Red Hat OpenStack Platform을 배포할 때 각 코어 서비스는 각 오버클라우드 노드에 기본 방화벽 규칙 세트와 함께 배포됩니다. **ExtraFirewallRules** 매개변수를 사용하여 추가 서비스의 포트를 여는 규칙을 만들거나 서비스를 제한하는 규칙을 만들 수 있습니다.

각 규칙 이름은 각 **iptables** 규칙에 대한 주석이 됩니다. 각 규칙 이름은 최종 **iptables** 파일에서 Puppet에서 규칙을 정렬하는 데 도움이 되도록 3자리 접두사로 시작합니다. 기본 Red Hat OpenStack Platform 규칙은 000~200 범위의 접두사를 사용합니다. 새 서비스에 대한 규칙을 생성할 때 이름 앞에는 200보다 큰 3자리 숫자의 접두사를 추가합니다.

프로세스

1. 문자열을 사용하여 **ExtraFireWallRules** 매개변수 아래에 각 규칙 이름을 정의합니다. 규칙 이름 아래에 다음 매개변수를 사용하여 규칙을 정의할 수 있습니다.
 - **dport::** 규칙에 연결된 대상 포트입니다.
 - **proto::** 규칙과 연결된 프로토콜입니다. 기본값은 **tcp** 입니다.
 - **action:** 규칙과 연결된 작업 정책입니다. 기본값을 수락합니다.
 - **Source::** 규칙과 연결된 소스 IP 주소입니다.

다음 예제에서는 규칙을 사용하여 사용자 지정 애플리케이션에 대한 추가 포트를 여는 방법을 보여줍니다.

```
cat > ~/templates/firewall.yaml <<EOF
parameter_defaults:
  ExtraFirewallRules:
    '300 allow custom application 1':
```

```
dport: 999
proto: udp
'301 allow custom application 2':
dport: 8081
proto: tcp
EOF
```



참고

action 매개변수를 설정하지 않으면 결과가 허용됩니다. 삭제, 삽입 또는 추가하도록 **action** 매개변수만 설정할 수 있습니다.

2. **openstack overcloud deploy** 명령에 **~/templates/firewall.yaml** 파일을 포함합니다. 배포에 필요한 모든 템플릿을 포함합니다.

```
openstack overcloud deploy --templates /
...
-e /home/stack/templates/firewall.yaml /
....
```

2.3. 오버클라우드 방화벽에서 서비스 삭제

규칙을 사용하여 서비스를 제한할 수 있습니다. 규칙 이름에 사용하는 번호에 따라 **iptables** 에서 규칙이 삽입되는 위치가 결정됩니다. 다음 절차에서는 **rabbitmq** 서비스를 InternalAPI 네트워크로 제한하는 방법을 보여줍니다.

프로세스

1. 컨트롤러 노드에서 **rabbitmq** 의 기본 **iptables** 규칙 수를 찾습니다.

```
[tripleo-admin@overcloud-controller-2 ~]$ sudo iptables -L | grep rabbitmq
ACCEPT tcp -- anywhere anywhere multiport dports vtr-
emulator,epmd,amqp,25672,25673:25683 state NEW /* 109 rabbitmq-bundle ipv4 */
```

2. 환경 파일 **parameter_defaults** 에서 **ExtraFirewallRules** 매개변수를 사용하여 **rabbitmq** 를 InternalApi 네트워크로 제한합니다. 이 규칙에는 기본 **rabbitmq** 규칙 번호 또는 109보다 낮은 수가 제공됩니다.

```
cat > ~/templates/firewall.yaml <<EOF
parameter_defaults:
  ExtraFirewallRules:
    '098 allow rabbit from internalapi network':
      dport:
        - 4369
        - 5672
        - 25672
      proto: tcp
      source: 10.0.0.0/24
    '099 drop other rabbit access':
      dport:
        - 4369
        - 5672
        - 25672
```

```
proto: tcp
action: drop
EOF
```



참고

action 매개변수를 설정하지 않으면 결과가 허용됩니다. **삭제, 삽입** 또는 **추가** 하도록 **action** 매개변수만 설정할 수 있습니다.

3. **openstack overcloud deploy** 명령에 **~/templates/firewall.yaml** 파일을 포함합니다. 배포에 필요한 모든 템플릿을 포함합니다.

```
openstack overcloud deploy --templates /
...
-e /home/stack/templates/firewall.yaml /
....
```

2.4. SNMP(SIMPLE NETWORK MANAGEMENT PROTOCOL) 문자열 변경

director는 오버클라우드에 대한 기본 읽기 전용 SNMP 설정을 제공합니다. 네트워크 장치에 대해 학습하지 않은 사용자의 위험을 완화하기 위해 SNMP 문자열을 변경하는 것이 좋습니다.



참고

string 매개변수를 사용하여 **ExtraConfig** 인터페이스를 구성할 때 heat 및 Hiera가 문자열을 **"<VALUE>"** 로 해석하지 않도록 하려면 다음 구문을 사용해야 합니다.

오버클라우드의 환경 파일에서 **ExtraConfig** 후크를 사용하여 다음 hieradata를 설정합니다.

SNMP 기존 액세스 제어 설정

snmp::ro_community

IPv4 읽기 전용 SNMP 커뮤니티 문자열입니다. 기본값은 **public** 입니다.

snmp::ro_community6

IPv6 읽기 전용 SNMP 커뮤니티 문자열입니다. 기본값은 **public** 입니다.

snmp::ro_network

RO가 데몬을 쿼리할 수 있는 네트워크입니다. 이 값은 문자열 또는 배열일 수 있습니다. 기본값은 **127.0.0.1** 입니다.

snmp::ro_network6

RO로 **IPv6**로 데몬을 쿼리할 수 있는 네트워크입니다. 이 값은 문자열 또는 배열일 수 있습니다. 기본값은 **::1/128** 입니다.

tripleo::profile::base::snmp::snmpd_config

snmpd.conf 파일에 안전으로 추가할 줄의 배열입니다. 기본값은 **[]** 입니다. 사용 가능한 모든 옵션은 [SNMP 구성 파일](#) 웹 페이지를 참조하십시오.

예를 들면 다음과 같습니다.

```
parameter_defaults:
  ExtraConfig:
    snmp::ro_community: mysecurestring
```

```
snmp::ro_community6: myv6securestring
```

이렇게 하면 모든 노드에서 읽기 전용 SNMP 커뮤니티 문자열이 변경됩니다.

SNMP 보기 기반 액세스 제어 설정(VACM)

snmp::com2sec

VACM com2sec 매핑 배열입니다. WaylandNAME, SOURCE 및 COMMUNITY를 제공해야 합니다.

snmp::com2sec6

VACM com2sec6 매핑의 배열입니다. WaylandNAME, SOURCE 및 COMMUNITY를 제공해야 합니다.

예를 들면 다음과 같습니다.

```
parameter_defaults:
  ExtraConfig:
    snmp::com2sec: ["notConfigUser default mysecurestring"]
    snmp::com2sec6: ["notConfigUser default myv6securestring"]
```

이렇게 하면 모든 노드에서 읽기 전용 SNMP 커뮤니티 문자열이 변경됩니다.

자세한 내용은 **snmpd.conf** 도움말 페이지를 참조하십시오.

2.5. OPEN VSWITCH 방화벽 사용

Red Hat OpenStack Platform director에서 OVS(Open vSwitch) 방화벽 드라이버를 사용하도록 보안 그룹을 구성할 수 있습니다. **NeutronOVSFirewallDriver** 매개변수를 사용하여 사용하려는 방화벽 드라이버를 지정합니다.

- **iptables_hybrid** - iptables/hybrid 기반 구현을 사용하도록 네트워킹 서비스(neutron)를 구성합니다.
- **openvswitch** - OVS 방화벽 흐름 기반 드라이버를 사용하도록 네트워킹 서비스를 구성합니다.

openvswitch 방화벽 드라이버는 성능이 뛰어나고 게스트를 프로젝트 네트워크에 연결하는 데 사용되는 인터페이스 및 브리지 수를 줄입니다.



중요

멀티 캐스트 트래픽은 OVS(Open vSwitch) 방화벽 드라이버와 iptables 방화벽 드라이버에 의해 다르게 처리됩니다. iptables를 사용하면 기본적으로 VRRP 트래픽이 거부되고 모든 VRRP 트래픽이 끝점에 도달하기 위해 보안 그룹 규칙에서 VRRP를 활성화해야 합니다. OVS를 사용하면 모든 포트가 동일한 OpenFlow 컨텍스트를 공유하고 멀티캐스트 트래픽은 포트당 개별적으로 처리할 수 없습니다. 보안 그룹은 모든 포트(예: 라우터의 포트)에 적용되지 않으므로 OVS는 **NORMAL** 작업을 사용하고 RFC 4541에서 지정한 모든 포트로 멀티캐스트 트래픽을 전달합니다.



참고

iptables_hybrid 옵션은 OVS-DPDK와 호환되지 않습니다. **openvswitch** 옵션은 OVS Hardware Offload와 호환되지 않습니다.

network-environment.yaml 파일에서 **NeutronOVSFirewallDriver** 매개변수를 구성합니다.

NeutronOVSFirewallDriver: openvswitch

- **NeutronOVSFirewallDriver** : 보안 그룹을 구현할 때 사용할 방화벽 드라이버의 이름을 구성합니다. 가능한 값은 시스템 구성에 따라 다릅니다. 몇 가지 예는 **noop**, **openvswitch**, **iptables_hybrid**입니다. 빈 문자열의 기본값은 구성이 지원됩니다.

3장. RHOSP 환경 기록

시스템 구성 요소, 네트워크, 서비스 및 소프트웨어를 문서화하는 것은 보안 문제, 공격 벡터 및 가능한 보안 영역 브리징 포인트를 식별하는 데 중요합니다. RHOSP(Red Hat OpenStack Platform) 배포에 대한 설명서에는 다음 정보가 포함되어야 합니다.

- RHOSP 프로덕션, 개발 및 테스트 환경의 시스템 구성 요소, 네트워크, 서비스 및 소프트웨어에 대한 설명입니다.
- 가상 머신 또는 가상 디스크 볼륨과 같은 임시 리소스의 인벤토리입니다.

3.1. 시스템 역할 기록

RHOSP(Red Hat OpenStack Platform) 배포의 각 노드는 클라우드 인프라에 기여하거나 클라우드 리소스를 제공하는 특정 역할을 수행합니다.

인프라에 기여하는 노드는 클라우드의 운영 및 프로비저닝을 지원하는 데 필요한 메시지 대기열 서비스, 스토리지 관리, 모니터링, 네트워킹 및 기타 서비스와 같은 클라우드 관련 서비스를 실행합니다. 인프라 역할의 예로는 다음이 포함됩니다.

- 컨트롤러
- Networker
- 데이터베이스
- Telemetry

클라우드 리소스를 제공하는 노드는 클라우드에서 실행되는 인스턴스에 대해 컴퓨팅 또는 스토리지 용량을 제공합니다. 리소스 역할의 예는 다음과 같습니다.

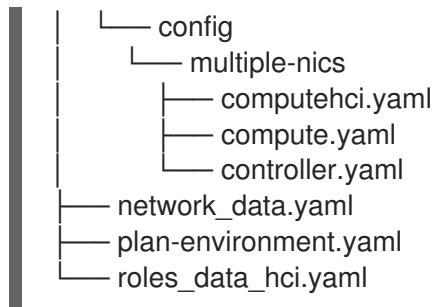
- CephStorage
- Compute
- ComputeOvsDpdk
- ObjectStorage

사용자 환경에서 사용되는 시스템 역할을 문서화합니다. 이러한 역할은 RHOSP를 배포하는 데 사용되는 템플릿 내에서 식별할 수 있습니다. 예를 들어 사용자 환경에서 사용하는 각 역할에 대한 NIC 구성 파일이 있습니다.

절차

1. 현재 사용 중인 역할을 지정하는 파일의 배포에 대한 기존 템플릿을 확인합니다. 사용자 환경에서 사용하는 각 역할에 대한 NIC 구성 파일이 있습니다. 다음 예에서 RHOSP 환경에는 **ComputeHCI** 역할, **Compute** 역할 및 **Controller** 역할이 포함됩니다.

```
$ cd ~/templates
$ tree
.
├── environments
│   └── network-environment.yaml
├── hci.yaml
└── network
```



2. RHOSP 환경의 각 역할은 많은 상호 관련 서비스를 수행합니다. 역할 파일을 검사하여 각 **역할**에 사용하는 서비스를 문서화할 수 있습니다.

- a. 템플릿에 대한 **역할** 파일이 생성된 경우 **~/templates** 디렉터리에서 찾을 수 있습니다.

```
$ cd ~/templates
$ find . -name *role*
> ./templates/roles_data_hci.yaml
```

- b. 템플릿에 대한 **역할** 파일이 생성되지 않은 경우 문서 목적으로 현재 검사하는 역할에 대해 하나를 생성할 수 있습니다.

```
$ openstack overcloud roles generate \
> --roles-path /usr/share/openstack-tripleo-heat-templates/roles \
> -o roles_data.yaml Controller Compute
```

3.2. 하드웨어 인벤토리 생성

인트로스펙션 중 수집된 데이터를 확인하여 Red Hat OpenStack Platform 배포를 통해 하드웨어 정보를 검색할 수 있습니다. 인트로스펙션은 CPU, 메모리, 디스크 등에 대한 노드에서 하드웨어 정보를 수집합니다.

사전 요구 사항

- Red Hat OpenStack Platform director 환경이 설치되어 있어야 합니다.
- Red Hat OpenStack Platform 배포를 위한 노드가 인트로스펙션되어 있습니다.
- stack으로 director에 로그인되어 있습니다.

프로세스

1. 언더클라우드에서 **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 사용자 환경의 노드를 나열합니다.

```
$ openstack baremetal node list -c Name
+-----+
| Name   |
+-----+
| controller-0 |
| controller-1 |
| controller-2 |
```

```
| compute-0 |
| compute-1 |
| compute-2 |
+-----+
```

3. 정보를 수집하고 다음 명령을 실행하여 인트로스펙션 데이터를 검색하는 각 baremetal 노드에 대해 다음을 실행합니다.

```
$ openstack baremetal introspection data save <node> | jq
```

& **lt;node** >를 1단계에서 검색한 목록에서 노드 이름으로 바꿉니다.

4. 선택 사항: 출력을 특정 유형의 하드웨어로 제한하려면 인벤토리 키 목록을 검색하고 특정 키에 대한 인트로스펙션 데이터를 볼 수 있습니다.

- a. 다음 명령을 실행하여 인트로스펙션 데이터에서 최상위 키 목록을 가져옵니다.

```
$ openstack baremetal introspection data save controller-0 | jq '.inventory | keys'

[
  "bmc_address",
  "bmc_v6address",
  "boot",
  "cpu",
  "disks",
  "hostname",
  "interfaces",
  "memory",
  "system_vendor"
]
```

- b. 예를 들어 키(예: 디스크)를 선택하고 다음을 실행하여 자세한 정보를 가져옵니다.

```
$ openstack baremetal introspection data save controller-1 | jq '.inventory.disks'

[
  {
    "name": "/dev/sda",
    "model": "QEMU HARDDISK",
    "size": 85899345920,
    "rotational": true,
    "wwn": null,
    "serial": "QM00001",
    "vendor": "ATA",
    "wwn_with_extension": null,
    "wwn_vendor_extension": null,
    "hctl": "0:0:0:0",
    "by_path": "/dev/disk/by-path/pci-0000:00:01.1-ata-1"
  }
]
```

3.3. 소프트웨어 인벤토리 생성

RHOSP(Red Hat OpenStack Platform) 인프라에 배포된 노드에서 사용 중인 소프트웨어 구성 요소를 문서화합니다. 시스템 데이터베이스, RHOSP 소프트웨어 서비스 및 로드 밸런서, DNS 또는 DHCP 서비스와

같은 지원 구성 요소는 라이브러리, 애플리케이션 또는 소프트웨어 클래스의 손상 또는 취약점을 평가할 때 중요합니다.

- Red Hat OpenStack Platform 환경이 설치되어 있어야 합니다.
- stack으로 director에 로그인되어 있습니다.

프로세스

1. 악의적인 활동을 받을 수 있는 시스템 및 서비스의 진입점을 알고 있어야 합니다. 언더클라우드에서 다음 명령을 실행합니다.

```
$ cat /etc/hosts
$ source stackrc ; openstack endpoint list
$ source overcloudrc ; openstack endpoint list
```

2. RHOSP는 컨테이너화된 서비스에 배포되므로 해당 노드에서 실행 중인 컨테이너를 확인하여 오버클라우드 노드에서 소프트웨어 구성 요소를 볼 수 있습니다. **ssh** 를 사용하여 오버클라우드 노드에 연결하고 실행 중인 컨테이너를 나열합니다. 예를 들어 **compute-0** 에서 오버클라우드 서비스를 보려면 다음과 유사한 명령을 실행합니다.

```
$ ssh tripleo-admin@compute-0 podman ps
```

4장. ID 및 액세스 관리

ID 서비스(keystone)는 Red Hat OpenStack Platform 환경에서 클라우드 사용자에게 인증 및 권한 부여를 제공합니다. 직접 최종 사용자 인증에 ID 서비스를 사용하거나 외부 인증 방법을 사용하여 보안 요구 사항을 충족하거나 현재 인증 인프라와 일치하도록 구성할 수 있습니다.

4.1. RED HAT OPENSTACK PLATFORM FERNET 토큰

Cryostat는 **UUID** 토큰 공급자를 대체하는 기본 토큰 공급자입니다. 각 페넷 토큰은 기본적으로 최대 1시간 동안 유효합니다. 이를 통해 사용자는 재인증할 필요 없이 일련의 작업을 수행할 수 있습니다.

인증 후 ID 서비스(keystone)입니다.

- fernet 토큰이라는 암호화된 전달자 토큰을 발행합니다. 이 토큰은 사용자의 ID를 나타냅니다.
- 역할에 따라 작업을 수행할 수 있도록 권한을 부여합니다.

추가 리소스

- [오버클라우드의 암호화를 위해 Cryostat 키 사용](#)

4.2. OPENSTACK ID 서비스 엔티티

Red Hat OpenStack Identity 서비스(keystone)는 다음 엔티티를 인식합니다.

사용자

OpenStack Identity 서비스(keystone) 사용자는 인증의 원자 단위입니다. 인증하려면 사용자에게 프로젝트에 대한 역할을 할당해야 합니다.

그룹

OpenStack ID 서비스 그룹은 사용자의 논리 그룹입니다. 그룹은 특정 역할에 따라 프로젝트에 대한 액세스 권한을 제공할 수 있습니다. 사용자 대신 그룹을 관리하면 역할 관리를 단순화할 수 있습니다.

역할

OpenStack ID 서비스 역할은 해당 역할이 할당된 사용자 또는 그룹이 액세스할 수 있는 OpenStack API를 정의합니다.

프로젝트

OpenStack ID 서비스 프로젝트는 물리적 리소스의 공유 할당량과 해당 물리적 리소스에서 구축된 가상 인프라에 대한 공통 액세스 권한이 있는 격리된 사용자 그룹입니다.

도메인

OpenStack ID 서비스 도메인은 프로젝트, 사용자 및 그룹의 높은 수준의 보안 경계입니다. OpenStack Identity 도메인을 사용하여 모든 keystone 기반 ID 구성 요소를 중앙에서 관리할 수 있습니다. Red Hat OpenStack Platform은 여러 도메인을 지원합니다. 별도의 인증 백엔드를 사용하여 다른 도메인의 사용자를 나타낼 수 있습니다.

4.3. KEYSTONE으로 인증

OpenStack Identity 서비스(keystone)에 필요한 인증 보안 요구 사항을 조정할 수 있습니다.

표 4.1. ID 서비스 인증 매개변수

매개변수	설명
------	----

KeystoneLockoutDuration	KeystoneLockoutFailureAttempts 에서 지정한 대로 최대 실패한 인증 시도 수를 초과할 때 사용자 계정이 잠긴 시간(초)입니다.
KeystoneLockoutFailureAttempts	KeystoneLockoutDuration 에서 지정한 시간(초) 동안 사용자 계정이 잠기기 전에 사용자가 인증할 수 있는 최대 횟수입니다.
KeystoneMinimumPasswordAge	사용자가 암호를 변경하기 전에 암호를 사용해야 하는 일 수입니다. 이렇게 하면 사용자가 암호 기록을 지우고 이전 암호를 재사용하기 위해 즉시 암호를 변경할 수 없습니다.
KeystoneUniqueLastPasswordCount	이렇게 하면 새로 생성된 암호가 고유한 암호를 강제 적용하기 위해 기록에 보관하기 위해 이전 사용자 암호 반복 횟수가 제어됩니다.

추가 리소스

- [ID\(keystone\) 매개 변수](#).

4.4. ID 서비스 HEAT 매개변수를 사용하여 잘못된 로그인 시도를 중지

반복적으로 실패한 로그인 시도는 시도된 무차별 적용 공격의 표시가 될 수 있습니다. ID 서비스를 사용하여 반복적으로 실패한 로그인 시도 후 계정에 대한 액세스를 제한할 수 있습니다.

사전 요구 사항

- Red Hat OpenStack Platform director 환경이 설치되어 있어야 합니다.
- stack으로 director에 로그인되어 있습니다.

프로세스

1. 사용자가 사용자 계정이 잠기기 전에 인증할 수 없는 최대 횟수를 구성하려면 환경 파일에서 **KeystoneLockoutFailureAttempts** 및 **KeystoneLockoutDuration** heat 매개변수 값을 설정합니다. 다음 예에서는 **KeystoneLockoutDuration** 이 1시간으로 설정됩니다.

```
parameter_defaults
  KeystoneLockoutDuration: 3600
  KeystoneLockoutFailureAttempts: 3
```

2. 배포 스크립트에 환경 파일을 포함합니다. 이전에 배포한 환경에서 배포 스크립트를 실행하면 추가 매개변수로 업데이트됩니다.

```
openstack overcloud deploy --templates \
...
-e keystone_config.yaml
...
```

4.5. 외부 ID 공급자로 인증

외부 ID 공급자(IdP)를 사용하여 OpenStack 서비스 공급자(SP)에 인증할 수 있습니다. SPS는 OpenStack 클라우드에서 제공하는 서비스입니다.

별도의 IdP를 사용하는 경우 외부 인증 자격 증명은 다른 OpenStack 서비스에서 사용하는 데이터베이스와 다릅니다. 이렇게 분리하면 저장된 인증 정보가 손상될 위험이 줄어듭니다.

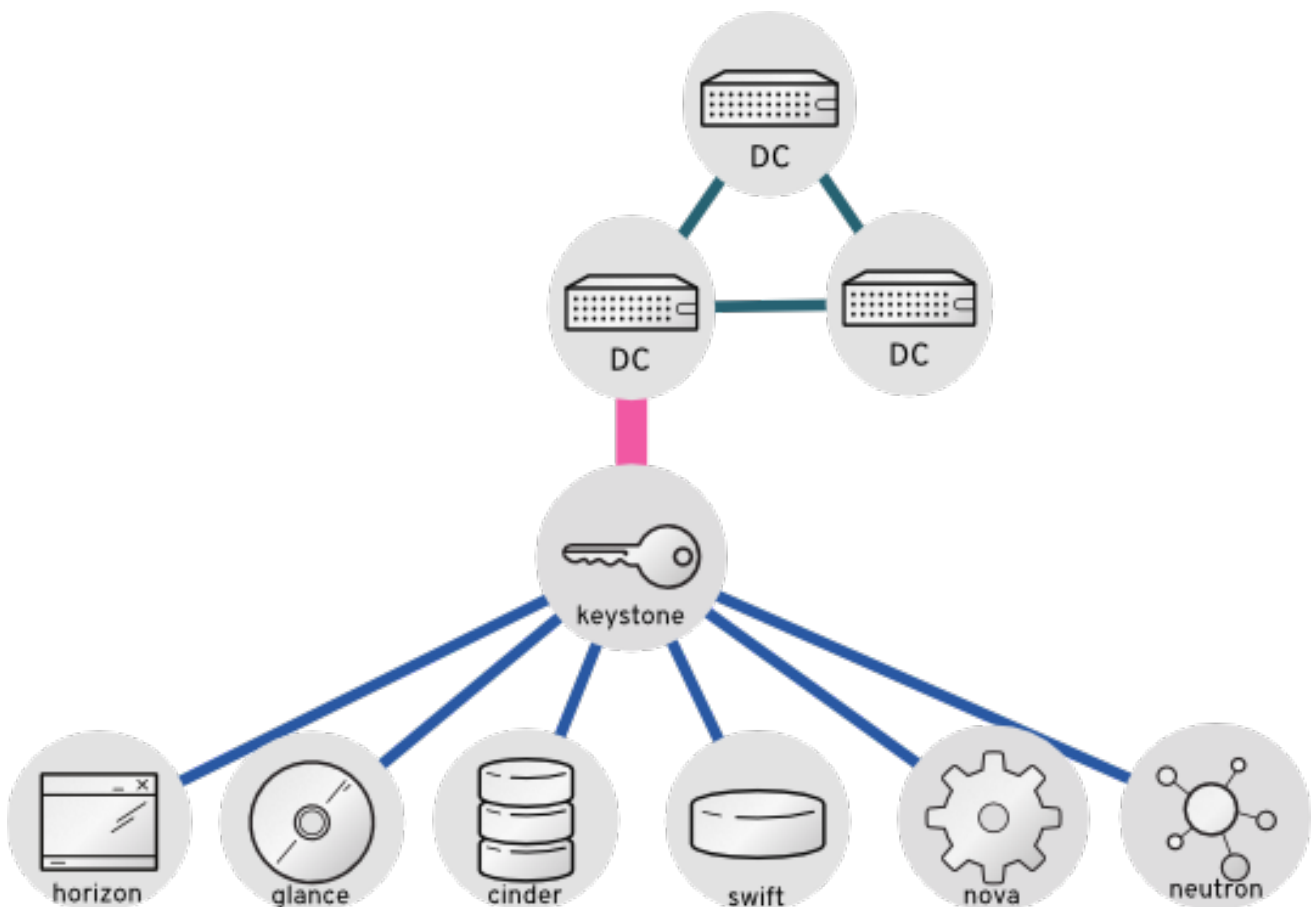
각 외부 IdP에는 OpenStack ID 서비스(keystone) 도메인에 일대일 매핑이 있습니다. Red Hat OpenStack Platform과 공존하는 도메인을 여러 개 보유할 수 있습니다.

외부 인증을 사용하면 추가 ID를 생성하지 않고 기존 인증 정보를 사용하여 Red Hat OpenStack Platform의 리소스에 액세스할 수 있습니다. 인증 정보는 사용자의 IdP에 의해 유지 관리됩니다.

ID 관리를 위해 Red Hat IdM(Identity Management) 및 AD DS(Microsoft Active Directory Domain Services)와 같은 IdP를 사용할 수 있습니다. 이 구성에서 OpenStack Identity 서비스에는 LDAP 사용자 데이터베이스에 대한 읽기 전용 액세스 권한이 있습니다. 사용자 또는 그룹 역할을 기반으로 하는 API 액세스 관리는 keystone에 의해 수행됩니다. 역할은 OpenStack ID 서비스를 사용하여 LDAP 계정에 할당됩니다.

4.5.1. LDAP 통합의 작동 방식

아래 다이어그램에서 keystone은 암호화된 LDAPS 연결을 사용하여 Active Directory 도메인 컨트롤러에 연결합니다. 사용자가 Horizon에 로그인하면 keystone은 제공된 사용자 자격 증명을 수신하여 Active Directory에 전달합니다.



추가 리소스

- [OpenStack Identity\(keystone\)와 Active Directory 통합](#)

- [Red Hat IdM\(Identity Manager\)](#)과 OpenStack Identity(keystone) 통합
- 도메인별 LDAP 백엔드를 사용하도록 director 구성

5장. TLS 및 PKI를 사용하여 RED HAT OPENSTACK 배포 보안

Red Hat OpenStack Platform은 사용자가 보호할 수 있는 민감한 데이터 또는 기밀 데이터를 처리하는 많은 네트워크와 엔드포인트로 구성됩니다. TLS(Transport Layer Security)를 사용하면 대칭 키 암호화를 사용하여 트래픽을 보호합니다. 키와 암호는 TLS 핸드셰이크에서 협상됩니다. TLS 핸드셰이크는 CA(인증 기관)라는 중개자(CA)에서 공유 신뢰를 통해 서버 ID를 검증해야 합니다.

PKI(Public Key Infrastructure)는 인증 기관을 통해 엔티티를 검증하기 위한 프레임워크입니다.

5.1. PKI(PUBLIC KEY INFRASTRUCTURE)의 구성 요소

PKI의 핵심 구성 요소는 다음 표에 표시됩니다.

표 5.1. 주요 용어

용어	정의
최종 엔티티	디지털 인증서를 사용하여 자신을 검증하는 사용자, 프로세스 또는 시스템입니다.
인증 기관(CA)	CA는 최종 엔티티와 최종 엔티티의 유효성을 확인하는 신뢰 당사자 둘 다에서 신뢰하는 엔티티입니다.
고객 지원	신뢰 당사자는 최종 엔티티의 검증으로 디지털 인증서를 수신하며 디지털 인증서를 확인할 수 있습니다.
디지털 인증서	서명된 공개 키 인증서에는 검증 가능한 엔티티와 공개 키가 있으며 CA에서 발행합니다. CA가 인증서에 서명하면 개인 키로 암호화된 인증서에서 메시지 다이제스트를 생성합니다. CA와 연결된 공개 키를 사용하여 서명을 확인할 수 있습니다. X.509 표준은 인증서를 정의하는 데 사용됩니다.
등록 기관 (RA)	RA는 CA에서 인증서를 발급하기 전에 최종 엔티티 인증과 같은 관리 기능을 수행할 수 있는 선택적 전용 기관입니다. RA가 없는 경우 CA는 최종 엔티티를 인증합니다.
CRL(Certificate Revocation List)	CRL은 취소된 인증서 일련 번호 목록입니다. 취소된 일련 번호가 있는 인증서를 제공하는 최종 엔티티는 PKI 모델에서 신뢰할 수 없습니다.
CRL 발행자	CA가 인증서 취소 목록 게시를 위임하는 선택적 시스템입니다.
인증서 리포지토리	최종 엔티티 인증서 및 인증서 취소 목록이 저장되고 쿼리되는 위치입니다.

5.2. 인증 기관 요구 사항 및 권장 사항

공개적으로 사용 가능한 Red Hat OpenStack Platform 대시보드 또는 공개적으로 액세스 가능한 API에 대해 널리 알려진 CA(인증 기관)에서 서명한 인증서를 가져와야 합니다.

TLS로 보안되는 각 끝점에 DNS 도메인 또는 하위 도메인을 지정해야 합니다. 제공하는 도메인은 CA에서 발급한 인증서를 생성하는 데 사용됩니다. 고객은 CA가 엔드포인트를 검증할 수 있도록 DNS 이름을 사용하여 대시보드 또는 API에 액세스합니다.

Red Hat은 별도의 및 내부적으로 관리되는 CA를 사용하여 내부 트래픽을 보호하는 것이 좋습니다. 이를 통해 클라우드 배포자는 개인 키 인프라(PKI) 구현을 유지 관리하고 내부 시스템을 위해 인증서를 요청, 서명 및 배포할 수 있습니다.

오버클라우드 끝점에서 SSL/TLS를 활성화할 수 있습니다. 모든 곳에서 TLS를 설정하는 데 필요한 인증서 수(TLS-e)로 인해 director는 Red Hat IdM(Identity Management) 서버와 통합하여 인증 기관 역할을 수행하고 오버클라우드 인증서를 관리합니다. TLS-e 구성에 대한 자세한 내용은 [Ansible을 사용한 TLS 구현](#) 을 참조하십시오.

OpenStack 구성 요소에서 TLS 지원 상태를 확인하려면 [TLS 사용 상태 매트릭스](#) 를 참조하십시오.

자체 인증 기관이 있는 SSL 인증서를 사용하려면 [오버클라우드 공용 끝점에서 SSL/TLS 활성화](#)를 참조하십시오.



참고

이렇게 하면 공개적으로 액세스 가능한 엔드포인트에서만 SSL/TLS를 사용하여 Red Hat OpenStack Platform이 설정됩니다.

5.3. 사용자 환경에서 TLS 버전 확인



중요

TLS 버전 1.0은 Red Hat OpenStack 플랫폼에서 더 이상 사용되지 않습니다. 또한 NIST-approval에 대해 TLS 1.2를 사용해야 합니다. 자세한 내용은 [선택, 구성 및 TLS\(Transport Layer Security\) 구현에 대한 지침](#) 을 참조하십시오.

cipherscan 을 사용하여 배포에서 제공하는 TLS 버전을 확인할 수 있습니다. Cipherscan은 <https://github.com/mozilla/cipherscan> 에서 복제할 수 있습니다. 이 예제 출력은 **Horizon** 에서 수신한 결과를 보여줍니다.



참고

프로덕션 환경 이외의 시스템에서 **암호를 실행할 수 있으므로** 처음 실행할 때 추가 종속 항목을 설치할 수 있습니다.

프로세스

- 대시보드 서비스의 액세스 가능한 **URL** 에 대해 암호 검사를 실행합니다.

```
$ ./cipherscan https://openstack.lab.local
```

```
.....
```

```
Target: openstack.lab.local:443
```

```
prio ciphersuite          protocols pfs          curves
1  ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2  ECDH,P-256,256bits prime256v1
2  ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2  ECDH,P-256,256bits prime256v1
```

```

3  DHE-RSA-AES128-GCM-SHA256  TLSv1.2  DH,1024bits  None
4  DHE-RSA-AES256-GCM-SHA384  TLSv1.2  DH,1024bits  None
5  ECDHE-RSA-AES128-SHA256    TLSv1.2  ECDH,P-256,256bits prime256v1
6  ECDHE-RSA-AES256-SHA384    TLSv1.2  ECDH,P-256,256bits prime256v1
7  ECDHE-RSA-AES128-SHA       TLSv1.2  ECDH,P-256,256bits prime256v1
8  ECDHE-RSA-AES256-SHA       TLSv1.2  ECDH,P-256,256bits prime256v1
9  DHE-RSA-AES128-SHA256      TLSv1.2  DH,1024bits  None
10 DHE-RSA-AES128-SHA         TLSv1.2  DH,1024bits  None
11 DHE-RSA-AES256-SHA256      TLSv1.2  DH,1024bits  None
12 DHE-RSA-AES256-SHA         TLSv1.2  DH,1024bits  None
13 ECDHE-RSA-DES-CBC3-SHA     TLSv1.2  ECDH,P-256,256bits prime256v1
14 EDH-RSA-DES-CBC3-SHA       TLSv1.2  DH,1024bits  None
15 AES128-GCM-SHA256          TLSv1.2  None         None
16 AES256-GCM-SHA384          TLSv1.2  None         None
17 AES128-SHA256              TLSv1.2  None         None
18 AES256-SHA256              TLSv1.2  None         None
19 AES128-SHA                  TLSv1.2  None         None
20 AES256-SHA                  TLSv1.2  None         None
21 DES-CBC3-SHA               TLSv1.2  None         None

```

Certificate: trusted, 2048 bits, sha256WithRSAEncryption signature

TLS ticket lifetime hint: None

NPN protocols: None

OCSP stapling: not supported

Cipher ordering: server

Curves ordering: server - fallback: no

Server supports secure renegotiation

Server supported compression methods: NONE

TLS Tolerance: yes

Intolerance to:

SSL 3.254 : absent

TLS 1.0 : PRESENT

TLS 1.1 : PRESENT

TLS 1.2 : absent

TLS 1.3 : absent

TLS 1.4 : absent

서버를 스캔할 때 Cipherscan은 협상할 수 있는 가장 높은 TLS 버전인 특정 TLS 버전에 대한 지원을 알립니다. 대상 서버가 TLS 프로토콜을 올바르게 따르는 경우 상호 지원되는 가장 높은 버전으로 응답합니다. 이는 Cipherscan이 처음 공개된 것보다 낮을 수 있습니다. 서버가 해당 특정 버전을 사용하여 클라이언트와의 연결을 설정하는 동안 진행 중인 경우 해당 프로토콜 버전에 영향을 미치지 않는 것으로 간주되지 않습니다. 연결(지정된 버전 또는 더 낮은 버전)이 설정되지 않은 경우 해당 프로토콜 버전에 대한 과민성이 있는 것으로 간주됩니다. 예를 들면 다음과 같습니다.

Intolerance to:

SSL 3.254 : absent

TLS 1.0 : PRESENT

TLS 1.1 : PRESENT

TLS 1.2 : absent

TLS 1.3 : absent

TLS 1.4 : absent

이 출력에서 **TLS 1.0** 및 **TLS 1.1**의 내성이 **PRESENT**로 보고됩니다. 즉, 연결을 설정할 수 없으며 해당 TLS 버전에 대한 지원을 알리는 동안 Cipherscan을 연결할 수 없었습니다. 따라서 이러한 프로토콜 버전 (및 더 낮은) 버전의 프로토콜이 스캔된 서버에서 활성화되지 않았다는 결론을 내는 것이 좋습니다.

5.4. OPENSTACK에 대한 IDM(IDENTITY MANAGEMENT) 서버 권장 사항

Red Hat은 IdM 서버 및 OpenStack 환경을 통합하는 데 도움이 되는 다음 정보를 제공합니다.

IdM 설치를 위한 Red Hat Enterprise Linux 준비에 대한 자세한 내용은 [Identity Management](#) 설치를 참조하십시오.

ipa-server-install 명령을 실행하여 IdM을 설치하고 구성합니다. 명령 매개변수를 사용하여 대화형 프롬프트를 건너뛸 수 있습니다. IdM 서버가 Red Hat OpenStack Platform 환경과 통합할 수 있도록 다음 권장 사항을 사용하십시오.

표 5.2. 매개변수 권장 사항

옵션	권장 사항
--admin-password	사용자가 제공하는 값을 기록해 둡니다. IdM과 작동하도록 Red Hat OpenStack Platform을 구성할 때 이 암호가 필요합니다.
--ip-address	사용자가 제공하는 값을 기록해 둡니다. 언더클라우드 및 오버클라우드 노드에는 이 IP 주소에 대한 네트워크 액세스 권한이 필요합니다.
--setup-dns	IdM 서버에 통합 DNS 서비스를 설치하려면 이 옵션을 사용합니다. 언더클라우드 및 오버클라우드 노드는 도메인 이름 확인에 IdM 서버를 사용합니다.
--auto-forwarders	/etc/resolv.conf 의 주소를 DNS 전달자로 사용하려면 이 옵션을 사용합니다.
--auto-reverse	이 옵션을 사용하여 IdM 서버 IP 주소의 역방향 레코드 및 영역을 확인합니다. 역방향 레코드 또는 영역을 확인할 수 없는 경우 IdM은 역방향 영역을 생성합니다. 이렇게 하면 IdM 배포가 간소화됩니다.
--ntp-server, --ntp-pool	이러한 옵션 중 하나 또는 둘 다를 사용하여 NTP 소스를 구성할 수 있습니다. IdM 서버와 OpenStack 환경 모두 올바르게 동기화된 시간이 있어야 합니다.

Red Hat OpenStack Platform 노드와의 통신을 활성화하려면 IdM에 필요한 방화벽 포트를 열어야 합니다. 자세한 내용은 [IdM에 필요한 포트 열기](#)를 참조하십시오.

추가 리소스

- [Identity Management 구성 및 관리](#)
- [Red Hat Identity Management 문서](#)

5.5. ANSIBLE을 사용하여 TLS-E 구현

새로운 **tripleo-ipa** 방법을 사용하여 TLS(TLS-e)라는 오버클라우드 끝점에서 SSL/TLS를 활성화할 수 있습니다. Red Hat OpenStack Platform은 필요한 인증서 수로 인해 Red Hat IdM(Identity Management)과 통합됩니다. **tripleo-ipa** 를 사용하여 TLS-e를 구성할 때 IdM은 인증 기관입니다.

사전 요구 사항

- stack 사용자 생성과 같은 언더클라우드의 모든 설정 단계가 완료되었는지 확인합니다. 자세한 내용은 [director를 사용하여 Red Hat OpenStack Platform 설치 및 관리](#)를 참조하십시오.
- DNS 서버의 IP 주소는 언더클라우드에 IdM 서버의 IP 주소로 구성됩니다. **undercloud.conf** 파일에 다음 매개변수 중 하나를 구성해야 합니다.
 - **DEFAULT/undercloud_nameservers**
 - **%SUBNET_SECTION%/dns_nameservers**

프로세스

다음 절차에 따라 Red Hat OpenStack Platform의 새 설치 또는 TLS-e로 구성할 기존 배포에 TLS-e를 구현합니다. 사전 프로비저닝된 노드에 TLS-e를 사용하여 Red Hat OpenStack Platform을 배포하는 경우 이 방법을 사용해야 합니다.



참고

기존 환경에 TLS-e를 구현하는 경우 **openstack undercloud install, openstack overcloud deploy** 와 같은 명령을 실행해야 합니다. 이러한 절차는 먹등이며 업데이트된 템플릿 및 구성 파일과 일치하도록 기존 배포 구성만 조정합니다.

1. **/etc/resolv.conf** 파일을 구성합니다.

/etc/resolv.conf 의 언더클라우드에 적절한 검색 도메인과 네임서버를 설정합니다. 예를 들어 배포 도메인이 **example.com** 이고 FreeIPA 서버의 도메인이 **bigcorp.com** 인 경우 **/etc/resolv.conf**에 다음 행을 추가합니다.

```
search example.com bigcorp.com
nameserver $IDM_SERVER_IP_ADDR
```

2. 필요한 소프트웨어를 설치합니다.

```
sudo dnf install -y python3-ipalib python3-ipaclient krb5-devel
```

3. 환경과 관련된 값으로 환경 변수를 내보냅니다.

```
export IPA_DOMAIN=bigcorp.com
export IPA_REALM=BIGCORP.COM
export IPA_ADMIN_USER=$IPA_USER ①
export IPA_ADMIN_PASSWORD=$IPA_PASSWORD ②
export IPA_SERVER_HOSTNAME=ipa.bigcorp.com
export UNDERCLOUD_FQDN=undercloud.example.com ③
export USER=stack
export CLOUD_DOMAIN=example.com
```

① ② IdM 사용자 인증 정보는 새 호스트 및 서비스를 추가할 수 있는 관리자입니다.

- 3 **UNDERCLOUD_FQDN** 매개변수의 값은 **/etc/hosts** 의 첫 번째 hostname-to-IP 주소 매핑과 일치합니다.

4. 언더클라우드에서 **undercloud-ipa-install.yaml** ansible 플레이북을 실행합니다.

```
ansible-playbook \
--ssh-extra-args "-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null" \
/usr/share/ansible/tripleo-playbooks/undercloud-ipa-install.yaml
```

5. **undercloud.conf**에 다음 매개변수를 추가합니다.

```
undercloud_nameservers = $IDM_SERVER_IP_ADDR
overcloud_domain_name = example.com
```

6. [선택 사항] IPA 영역이 IPA 도메인과 일치하지 않는 경우 **certmonger_krb_realm** 매개변수 값을 설정합니다.

- a. **/home/stack/hiera_override.yaml**에서 **certmonger_krb_realm**의 값을 설정합니다.

```
parameter_defaults:
  certmonger_krb_realm: EXAMPLE.COMPANY.COM
```

- b. **undercloud.conf**의 **custom_env_files** 매개변수 값을 **/home/stack/hiera_override.yaml**로 설정합니다.

```
custom_env_files = /home/stack/hiera_override.yaml
```

7. 언더클라우드를 배포합니다.

```
openstack undercloud install
```

검증

다음 단계를 완료하여 언더클라우드가 올바르게 등록되어 있는지 확인합니다.

1. IdM의 호스트를 나열합니다.

```
$ kinit admin
$ ipa host-find
```

2. **/etc/novajoin/krb5.keytab**이 언더클라우드에 있는지 확인합니다.

```
ls /etc/novajoin/krb5.keytab
```



참고

novajoin 디렉터리 이름은 레거시 이름 지정 목적으로만 사용됩니다.

오버클라우드에서 TLS-e 구성

TLS-e(TLS-e)를 사용하여 TLS를 사용하여 오버클라우드를 배포하면 Undercloud 및 Overcloud의 IP 주소가 자동으로 IdM에 등록됩니다.

1. 오버클라우드를 배포하기 전에 다음과 유사한 콘텐츠를 사용하여 YAML 파일 **tls-parameters.yaml** 을 생성합니다. 선택한 값은 환경에 따라 다릅니다.

```
parameter_defaults:
  DnsSearchDomains: ["example.com"]
  CloudDomain: example.com
  CloudName: overcloud.example.com
  CloudNameInternal: overcloud.internalapi.example.com
  CloudNameStorage: overcloud.storage.example.com
  CloudNameStorageManagement: overcloud.storagemgmt.example.com
  CloudNameCtlplane: overcloud.ctlplane.example.com
  IdMServer: freeipa-0.redhat.local
  IdMDomain: redhat.local
  IdMInstallClientPackages: False

resource_registry:
  OS::TripleO::Services::IpaClient: /usr/share/openstack-tripleo-heat-templates/deployment/ipa/ipaservices-baremetal-ansible.yaml
```

- **OS::TripleO::Services::IpaClient** 매개변수의 표시된 값은 **enable-internal-tls.yaml** 파일의 기본 설정을 재정의합니다. **openstack overcloud deploy** 명령에서 **tls-parameters.yaml** 파일이 **enable-internal-tls.yaml** 파일을 사용해야 합니다.
 - TLS-e를 구현하는 데 사용하는 매개변수에 대한 자세한 내용은 [tripleo-ipa](#) 의 매개변수를 참조하십시오.
2. [선택 사항] IPA 영역이 IPA 도메인과 일치하지 않는 경우 **tls-parameters.yaml** 파일에 **CertmongerKerberosRealm** 매개변수 값을 포함해야 합니다.

```
CertmongerKerberosRealm: EXAMPLE.COMPANY.COM
```

3. 오버클라우드를 배포합니다. 배포 명령에 **tls-parameters.yaml**을 포함해야 합니다.

```
DEFAULT_TEMPLATES=/usr/share/openstack-tripleo-heat-templates/
CUSTOM_TEMPLATES=/home/stack/templates

openstack overcloud deploy \
-e ${DEFAULT_TEMPLATES}/environments/ssl/tls-everywhere-endpoints-dns.yaml \
-e ${DEFAULT_TEMPLATES}/environments/services/haproxy-public-tls-certmonger.yaml \
-e ${DEFAULT_TEMPLATES}/environments/ssl/enable-internal-tls.yaml \
-e ${CUSTOM_TEMPLATES}/tls-parameters.yaml \
...
```

4. 끝점 목록에 대해 keystone을 쿼리하여 각 끝점이 HTTPS를 사용하고 있는지 확인합니다.

```
openstack endpoint list
```

5.6. TRIPLEO-IPA 매개변수

클라우드의 FQDN(정규화된 도메인 이름)을 사용하여 **tripleo-ipa** 에 필요한 클라우드 이름 및 클라우드 도메인 매개변수를 정의합니다. 예를 들어 **overcloud.example.com** 의 FQDN을 사용하여 다음 값을 사용합니다.

- CloudDomain: example.com

- `cloudName: overcloud.example.com`
- `CloudNameCtlplane: overcloud.ctlplane.example.com`
- `CloudNameInternal: overcloud.internalapi.example.com`
- `CloudNameStorage: overcloud.storage.example.com`
- `CloudNameStorageManagement: overcloud.storagemgmt.example.com`

환경 요구 사항에 따라 다음 추가 매개변수를 설정합니다.

CertmongerKerberosRealm

CertmongerKerberosRealm 매개변수를 IPA 영역 값으로 설정합니다. IPA 영역이 IPA 도메인과 일치하지 않는 경우 필요합니다.

DnsSearchDomains

DnsSearchDomains 매개변수는 쉼표로 구분된 목록입니다. IdM 서버의 도메인이 클라우드 도메인과 다른 경우 **DnsSearchDomains** 매개변수에 IdM 서버의 도메인을 포함합니다.

EnableEtcInternalTLS

DCN(Distributed Compute Node) 아키텍처에 TLS를 배포하는 경우 **True** 값을 사용하여 **EnableEtcInternalTLS** 매개변수를 추가해야 합니다.

IDMInstallClientPackages

컴퓨팅 노드를 사전 프로비저닝된 경우 **IDMInstallClientPackages** 매개변수를 **True** 값으로 설정합니다. 그렇지 않으면 값을 **False** 로 설정합니다.

IDMModifyDNS

Red Hat Identity Server에서 오버클라우드 노드의 자동 IP 등록을 비활성화하려면 **IDMModifyDNS** 매개변수를 **false** 로 설정합니다.

IdmDomain

IdmDomain 매개변수를 Red Hat Identity 서버의 FQDN의 domain 부분으로 설정합니다. 지정한 값은 IdM 영역의 값으로도 사용됩니다. IdM 도메인과 IdM 영역이 다른 경우 **CertmongerKerberosRealm** 매개변수를 사용하여 명시적으로 영역을 설정합니다.

IdmServer

IdmServer 매개변수를 Red Hat ID 서버의 FQDN으로 설정합니다. 복제된 IdM 환경을 사용하는 경우 쉼표로 구분된 목록을 사용하여 여러 값을 설정합니다. IdM 복제본에 대한 자세한 내용은 [IdM 복제본 설치](#)를 참조하십시오.

5.7. 모든 위치에서 TLS(TLS-E)에서 MEMCACHED 트래픽 암호화

이제 TLS-e를 사용하여 memcached 트래픽을 암호화할 수 있습니다. 이 기능은 novajoin 및 tripleo-ipa 둘 다에서 작동합니다.

1. 다음 내용으로 **memcached.yaml** 이라는 환경 파일을 생성하여 memcached에 대한 TLS 지원을 추가합니다.

```
parameter_defaults:
  MemcachedTLS: true
  MemcachedPort: 11212
```

2. 오버클라우드 배포 프로세스에 **memcached.yaml** 환경 파일을 포함합니다.

```
openstack overcloud deploy --templates \
```

```
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/enable-internal-tls.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-everywhere-endpoints-
dns.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/haproxy-public-tls-
certmonger.yaml \
-e /home/stack/memcached.yaml
...
```

추가 리소스

- tripleo-ipa를 사용하여 TLS를 배포하는 방법에 대한 자세한 내용은 [Ansible을 사용하여 TLS-e 구현을 참조하십시오](#).

5.8. 개인 키의 크기 증가

암호화된 서비스 트래픽에 대한 인증서를 생성하는 데 사용되는 개인 키의 크기를 늘려 보안을 개선할 수 있습니다. 기본 RHOSP 개인 키 크기는 최소로 권장되는 NIST(National Institute of Standards and Technology)와 일치합니다.

- **CertificateKeySize** 매개변수를 사용하여 전역적으로 개인 키의 크기를 변경합니다.
- **RedisCertificateKeySize** 와 같은 서비스별 매개변수를 사용하거나 특정 개인 키를 수정하거나 글로벌 **CertificateKeySize** 매개변수를 재정의합니다.

환경 heat 템플릿에서 이러한 매개변수를 사용하고 오버클라우드 배포 명령에 템플릿을 포함합니다. 오버클라우드를 이미 배포한 경우 원래 사용한 템플릿과 동일한 **openstack overcloud deploy** 명령을 재실행하고 변경 사항을 적용하려면 새 매개변수를 포함해야 합니다.

다음 예에서 개인 키에 대한 글로벌 값은 **4096** 입니다. **RedisCertificateKeySize** 는 글로벌 매개변수를 재정의하므로 **redis** 의 개인 키는 **2048** 입니다.

예제

```
parameter_defaults:
  CertificateKeySize: '4096'
  RedisCertificateKeySize: '2048'
```

5.9. RED HAT OPENSTACK PLATFORM의 IDM 서버를 복제본으로 교체

기존 IPA 서버를 복제본으로 교체할 때 필요한 매개 변수를 업데이트해야 합니다. 이로 인해 클러스터 구성을 업데이트할 때 오버클라우드 배포에 실패합니다.

프로세스

1. 각 오버클라우드 노드에서 **/etc/ipa/default.conf** 구성 파일을 편집하여 **server** 및 **xmlrpc_uri** 매개변수에 IdM 서버의 FQDN(정규화된 도메인 이름)이 포함되도록 합니다.

```
#File modified by ipa-client-install

[global]
basedn = dc=redhat,dc=local
realm = REDHAT.LOCAL
domain = redhat.local
server = freeipa-0.redhat.local
```

```
host = undercloud-0.redhat.local
xmlrpc_uri = https://freeipa-0.redhat.local/ipa/xml
enable_ra = True
```

2. 언더클라우드에서 **/home/stack/templates/tls-parameters.yaml** 환경 파일을 편집하고 **IPA_SERVER_HOSTNAME** 매개 변수가 **/etc/ipa/default.conf** 의 **xmlrpc_uri** 및 **server** 매개 변수에 표시된 FQDN과 일치하는지 확인합니다. 모든 매개 변수가 사용자 환경과 일치하는지 확인합니다.

```
export IPA_DOMAIN=bigcorp.com
export IPA_REALM=BIGCORP.COM
export IPA_ADMIN_USER=$IPA_USER
export IPA_ADMIN_PASSWORD=$IPA_PASSWORD
export IPA_SERVER_HOSTNAME=ipa.bigcorp.com
export UNDERCLOUD_FQDN=undercloud.example.com
export USER=stack
export CLOUD_DOMAIN=example.com
```

6장. 사용자 지정 SSL/TLS 인증서 설정

공용 끝점을 통한 통신에 SSL/TLS를 사용하도록 언더클라우드를 수동으로 설정할 수 있습니다. SSL/TLS를 사용하여 언더클라우드 끝점을 수동으로 구성할 때 보안 끝점을 개념 증명으로 생성합니다. Red Hat은 인증 기관 솔루션을 사용하는 것이 좋습니다.

CA(인증 기관) 솔루션을 사용하는 경우 인증서 갱신, CRL(인증 취소 목록) 및 업계에서 허용되는 암호화 와 같은 프로덕션 준비 솔루션이 있습니다. Red Hat IdM(Identity Manager)을 CA로 사용하는 방법에 대한 자세한 내용은 [Ansible을 사용하여 TLS 구현](#) 을 참조하십시오.

자체 인증 기관에서 SSL 인증서를 사용하려면 다음 구성 단계를 완료해야 합니다.

6.1. 서명 호스트 초기화

서명 호스트는 새 인증서를 생성하고 인증 기관을 통해 서명하는 호스트입니다. 선택한 서명 호스트에서 SSL 인증서를 생성한 적이 없는 경우 새 인증서에 서명할 수 있도록 호스트를 초기화해야 할 수 있습니다.

절차

1. **/etc/pki/CA/index.txt** 파일에는 서명된 모든 인증서의 기록이 포함되어 있습니다. 파일 시스템 경로와 **index.txt** 파일이 있는지 확인합니다.

```
$ sudo mkdir -p /etc/pki/CA
$ sudo touch /etc/pki/CA/index.txt
```

2. **/etc/pki/CA/serial** 파일은 서명할 다음 인증서에 사용할 다음 일련번호를 식별합니다. 이 파일이 있는지 확인합니다. 파일이 없는 경우 새 시작 값으로 새 파일을 생성합니다.

```
$ echo '1000' | sudo tee /etc/pki/CA/serial
```

6.2. 인증 기관 생성

일반적으로는 외부 인증 기관을 통해 SSL/TLS 인증서에 서명합니다. 고유한 인증 기관을 사용하려는 경우도 있습니다. 예를 들어 내부 전용 인증 기관을 사용할 수도 있습니다.

절차

1. 인증 기관 역할을 하는 키와 인증서 쌍을 생성합니다.

```
$ openssl genrsa -out ca.key.pem 4096
$ openssl req -key ca.key.pem -new -x509 -days 7300 -extensions v3_ca -out ca.crt.pem
```

2. **openssl req** 명령은 기관에 대한 특정 세부 정보를 요청합니다. 메시지가 나타나면 해당 세부 정보를 입력합니다. 이 명령을 수행하면 **ca.crt.pem**이라는 인증 기관 파일이 생성됩니다.
3. **enable-tls.yaml** 파일에서 인증서 위치를 **PublicTLSCAFile** 매개변수 값으로 설정합니다. 인증서 위치를 **PublicTLSCAFile** 매개변수 값으로 설정하면 CA 인증서 경로가 **clouds.yaml** 인증 파일에 추가되었는지 확인합니다.

```
parameter_defaults:
  PublicTLSCAFile: /etc/pki/ca-trust/source/anchors/cacert.pem
```

6.3. 클라이언트에 인증 기관 추가

외부 클라이언트가 SSL/TLS를 사용하여 통신하려는 경우 Red Hat OpenStack Platform 환경에 액세스해야 하는 각 클라이언트에 인증 기관 파일을 복사합니다.

절차

1. 클라이언트 시스템에 인증 기관을 복사합니다.

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

2. 각 클라이언트에 인증 기관 파일을 복사한 후 각 클라이언트에서 다음 명령을 실행하여 인증 기관 신뢰 번들에 인증서를 추가합니다.

```
$ sudo update-ca-trust extract
```

6.4. SSL/TLS 키 생성

OpenStack 환경에서 SSL/TLS를 활성화하려면 인증서를 생성하기 위한 SSL/TLS 키가 필요합니다.

절차

1. 다음 명령을 실행하여 SSL/TLS 키(**server.key.pem**)를 생성합니다.

```
$ openssl genrsa -out server.key.pem 2048
```

6.5. SSL/TLS 인증서 서명 요청 생성

인증서 서명 요청을 생성하려면 다음 단계를 완료합니다.

절차

1. 기본 OpenSSL 설정 파일을 복사합니다.

```
$ cp /etc/pki/tls/openssl.cnf .
```

2. 새 **openssl.cnf** 파일을 편집하고 director에 사용할 SSL 매개변수를 설정합니다. 수정할 매개변수 유형의 예제는 다음과 같습니다.

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req

[req_distinguished_name]
countryName = Country Name (2 letter code)
countryName_default = AU
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Queensland
localityName = Locality Name (eg, city)
localityName_default = Brisbane
organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = Red Hat
```

```

commonName = Common Name
commonName_default = 192.168.0.1
commonName_max = 64

[ v3_req ]
# Extensions to add to a certificate request
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names

[alt_names]
IP.1 = 192.168.0.1
DNS.1 = instack.localdomain
DNS.2 = vip.localdomain
DNS.3 = 192.168.0.1

```

commonName_default를 다음 항목 중 하나로 설정합니다.

- IP 주소를 사용하여 SSL/TLS를 통해 director에 액세스하는 경우 **undercloud.conf** 파일의 **undercloud_public_host** 매개변수를 사용합니다.
- 정규화된 도메인 이름을 사용하여 SSL/TLS를 통해 director에 액세스하는 경우 도메인 이름을 사용합니다.

alt_names 섹션을 편집하여 다음 항목을 포함합니다.

- **IP** - 클라이언트가 SSL을 통해 director에 액세스하는 데 사용하는 IP 주소 목록입니다.
- **DNS** - 클라이언트가 SSL을 통해 director에 액세스하는 데 사용하는 도메인 이름 목록입니다. 또한 공용 API IP 주소를 **alt_names** 섹션 끝에 DNS 항목으로 추가합니다.



참고

openssl.cnf에 대한 자세한 내용을 보려면 **man openssl.cnf** 명령을 실행합니다.

3. 다음 명령을 실행하여 인증서 서명 요청(**server.csr.pem**)을 생성합니다.

```
$ openssl req -config openssl.cnf -key server.key.pem -new -out server.csr.pem
```

-key 옵션을 사용하여 OpenStack SSL/TLS 키를 지정합니다.

이 명령을 실행하면 인증서 서명 요청인 **server.csr.pem** 파일이 생성됩니다. 이 파일을 사용하여 OpenStack SSL/TLS 인증서를 생성합니다.

6.6. SSL/TLS 인증서 생성

OpenStack 환경에 대한 SSL/TLS 인증서를 생성하려면 다음과 같은 파일이 필요합니다.

openssl.cnf

v3 확장을 지정하는 사용자 지정 설정 파일입니다.

server.csr.pem

인증서를 생성하고 인증 기관을 통해 서명하는 인증서 서명 요청입니다.

ca.crt.pem

인증서에 서명하는 인증 기관입니다.

ca.key.pem

인증 기관 개인 키입니다.

절차

1. **newcerts** 디렉터리가 없는 경우 해당 디렉터리를 생성합니다.

```
sudo mkdir -p /etc/pki/CA/newcerts
```

2. 다음 명령을 실행하여 언더클라우드 또는 오버클라우드에 대한 인증서를 생성합니다.

```
$ sudo openssl ca -config openssl.cnf -extensions v3_req -days 3650 -in server.csr.pem -out server.crt.pem -cert ca.crt.pem -keyfile ca.key.pem
```

이 명령은 다음 옵션을 사용합니다.

-config

사용자 지정 구성 파일, 즉 v3 확장이 포함된 **openssl.cnf** 파일을 사용합니다.

-extensions v3_req

v3 확장을 활성화합니다.

-days

인증서가 만료될 때까지 남은 기간(일)을 정의합니다.

-in

인증서 서명 요청입니다.

-out

생성된 서명된 인증서입니다.

-cert

인증 기관 파일입니다.

-keyfile

인증 기관 개인 키입니다.

이 명령을 실행하면 **server.crt.pem**이라는 새 인증서가 생성됩니다. 이 인증서를 OpenStack SSL/TLS 키와 함께 사용합니다.

6.7. 언더클라우드에 인증서 추가

언더클라우드 신뢰 번들에 OpenStack SSL/TLS 인증서를 추가하려면 다음 단계를 완료합니다.

절차

1. 다음 명령을 실행하여 인증서와 키를 결합합니다.

```
$ cat server.crt.pem server.key.pem > undercloud.pem
```

이 명령을 실행하면 **undercloud.pem** 파일이 생성됩니다.

2. **/etc/pki** 디렉터리 내의 위치에 **undercloud.pem** 파일을 복사하고 HAProxy가 읽을 수 있도록 필요한 SELinux 컨텍스트를 설정합니다.

```
$ sudo mkdir /etc/pki/undercloud-certs
$ sudo cp ~/undercloud.pem /etc/pki/undercloud-certs/.
$ sudo semanage fcontext -a -t etc_t "/etc/pki/undercloud-certs(/.*)?"
$ sudo restorecon -R /etc/pki/undercloud-certs
```

3. **undercloud.conf** 파일의 **undercloud_service_certificate** 옵션에 **undercloud.pem** 파일 위치를 추가합니다.

```
undercloud_service_certificate = /etc/pki/undercloud-certs/undercloud.pem
```

generate_service_certificate 및 **certificate_generation_ca** 매개변수를 설정하거나 활성화하지 마십시오. **director**는 이러한 매개변수를 사용하여 수동으로 생성한 **undercloud.pem** 인증서를 사용하는 대신 인증서를 자동으로 생성합니다.

4. 인증서에 서명한 인증 기관을 언더클라우드의 신뢰할 수 있는 인증 기관 목록에 추가하면 언더클라우드 내의 다른 서비스에서 해당 인증 기관에 액세스할 수 있습니다.

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

언더클라우드에 인증 기관이 추가되었는지 확인하려면 **openssl**을 사용하여 신뢰 번들을 확인합니다.

```
$ openssl crl2pkcs7 -nocrl -certfile /etc/pki/tls/certs/ca-bundle.crt | openssl pkcs7 -print_certs
-text | grep <CN of the CA issuer> -A 10 -B 10
```

<CN of the CA issuer>를 CA 발행자의 일반 이름으로 바꿉니다. 이 명령은 유효 날짜를 포함하여 주요 인증서 세부 정보를 출력합니다.

7장. 오버클라우드 공용 끝점에서 SSL/TLS 활성화

기본적으로 오버클라우드는 오버클라우드 서비스에 암호화되지 않은 엔드포인트를 사용합니다. 오버클라우드에서 SSL/TLS를 활성화하려면 CA(인증 기관) 솔루션을 사용하는 것이 좋습니다.

CA 솔루션을 사용하는 경우 인증서 갱신, CRL(인증 취소 목록) 및 업계에서 허용되는 암호화와 같은 프로덕션 준비 솔루션이 있습니다. Red Hat IdM(Identity Manager)을 CA로 사용하는 방법에 대한 자세한 내용은 [Ansible을 사용하여 TLS 구현](#) 을 참조하십시오.

다음 수동 프로세스를 사용하여 공용 API 엔드포인트에만 SSL/TLS를 활성화할 수 있으며 Internal 및 Admin API는 암호화되지 않은 상태로 유지됩니다. CA를 사용하지 않는 경우 SSL/TLS 인증서를 수동으로 업데이트해야 합니다. 자세한 내용은 [수동으로 SSL/TLS 인증서 업데이트](#)를 참조하십시오.

사전 요구 사항

- 공용 API의 끝점을 정의하는 네트워크 격리입니다.
- **openssl-perl** 패키지가 설치되어 있습니다.
- SSL/TLS 인증서가 있습니다. 자세한 내용은 [사용자 정의 SSL/TLS 인증서 구성](#)을 참조하십시오.

7.1. SSL/TLS 활성화

오버클라우드에서 SSL/TLS를 활성화하려면 SSL/TLS 인증서 및 개인 키에 대한 매개변수가 포함된 환경 파일을 생성해야 합니다.

절차

1. heat 템플릿 컬렉션에서 **enable-tls.yaml** 환경 파일을 복사합니다.

```
$ cp -r /usr/share/openstack-tripleo-heat-templates/environments/ssl/enable-tls.yaml
~/templates/.
```

2. 이 파일을 편집하고 이러한 매개변수에 대해 다음과 같이 변경합니다.

SSLCertificate

인증서 파일(**server.crt.pem**)의 콘텐츠를 **SSLCertificate** 매개변수에 복사합니다.

```
parameter_defaults:
  SSLCertificate: |
    -----BEGIN CERTIFICATE-----
    MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGS
    ...
    sFW3S2roS4X0Af/kSSD8mIBBTFTCMBAj6rtLBKLaQ
    -----END CERTIFICATE-----
```



중요

인증서 콘텐츠에는 모든 새 행에 대해 동일한 들여쓰기 수준이 필요합니다.

SSLIntermediateCertificate

중간 인증서가 있는 경우 중간 인증서의 내용을 **SSLIntermediateCertificate** 매개변수에 복사합니다.

```
parameter_defaults:
  SSLIntermediateCertificate: |
    -----BEGIN CERTIFICATE-----
    sFW3S2roS4X0Af/kSSD8mlBBTFTCMBAj6rtLBKLaQblxEplzrgvpBCwUAMFgx CzAJB
    ...
    MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGSIb3DQE
    -----END CERTIFICATE-----
```



중요

인증서 콘텐츠에는 모든 새 행에 대해 동일한 들여쓰기 수준이 필요합니다.

SSLKey

개인 키(**server.key.pem**)의 콘텐츠를 **SSLKey** 매개변수에 복사합니다.

```
parameter_defaults:
  ...
  SSLKey: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIIEowIBAAKCAQEAqVw8lnQ9Rbel1EdLN5PJP0IVO
    ...
    ctiKn3rAAadyumi4JDjESAXHIKfjJNOLrBmpQyES4X
    -----END RSA PRIVATE KEY-----
```



중요

개인 키 콘텐츠에는 모든 새 행에 대해 동일한 들여쓰기 수준이 필요합니다.

7.2. 루트 인증서 삽입

인증서 서명자가 오버클라우드 이미지의 기본 신뢰 저장소에 없는 경우 오버클라우드 이미지에 인증 기관을 삽입해야 합니다.

절차

1. heat 템플릿 컬렉션에서 **inject-trust-anchor-hiera.yaml** 환경 파일을 복사합니다.

```
$ cp -r /usr/share/openstack-tripleo-heat-templates/environments/ssl/inject-trust-anchor-hiera.yaml ~/templates/.
```

이 파일을 편집하고 이러한 매개변수에 대해 다음과 같이 변경합니다.

CAMap

오버클라우드에 삽입할 각 CA(인증 기관 콘텐츠)를 나열합니다. 오버클라우드에는 언더클라우드와 오버클라우드의 인증서에 서명하는 데 사용되는 CA 파일이 필요합니다. 루트 인증 기관 파일(**ca.crt.pem**)의 콘텐츠를 항목으로 복사합니다. 예를 들어 **CAMap** 매개변수는 다음과 같을 수 있습니다.

```
parameter_defaults:
  CAMap:
    ...
    undercloud-ca:
      content: |
        -----BEGIN CERTIFICATE-----
        MIIDITCCAn2gAwIBAgIJAOOnPtx2hHEhrMA0GCS
        BAYTAIVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBw
        UmVkiEhhdDELMAkGA1UECwwCUUUuFDASBgNVBA
        -----END CERTIFICATE-----
    overcloud-ca:
      content: |
        -----BEGIN CERTIFICATE-----
        MIIDBzCCAe+gAwIBAgIJAlc75A7FD++DMA0GCS
        BAMMD3d3dy5leGFtcGxlLmNvbTAeFw0xOTAxMz
        Um54yGCARyp3LpkxvyfMXX1DokpS1uKi7s6CkF
        -----END CERTIFICATE-----
```



중요

인증 기관 콘텐츠에는 모든 새 행에 대해 동일한 들여쓰기 수준이 필요합니다.

CAMap 매개변수를 사용하여 추가 CA를 삽입할 수도 있습니다.

7.3. DNS 끝점 구성

DNS 호스트 이름을 사용하여 SSL/TLS를 통해 오버클라우드에 액세스하는 경우 **/usr/share/openstack-tripleo-heat-templates/environments/predictable-placement/custom-domain.yaml** 파일을 **/home/stack/templates** 디렉터리에 복사합니다.



참고

이 환경 파일이 초기 배포에 포함되지 않은 경우 TLS-everywhere 아키텍처로 재배포할 수 없습니다.

필요한 경우 사용자 지정 네트워크에 대한 매개변수를 추가하여 모든 필드의 호스트 및 도메인 이름을 구성합니다.

CloudDomain

호스트의 DNS 도메인입니다.

CloudName

오버클라우드 끝점의 DNS 호스트 이름입니다.

CloudNameCtlplane

provisioning 네트워크 끝점의 DNS 이름입니다.

CloudNameInternal

내부 API 끝점의 DNS 이름입니다.

CloudNameStorage

스토리지 끝점의 DNS 이름입니다.

CloudNameStorageManagement

스토리지 관리 끝점의 DNS 이름입니다.

프로세스

- 다음 매개변수 중 하나를 사용하여 사용할 DNS 서버를 추가합니다.
 - **DEFAULT/undercloud_nameservers**
 - **%SUBNET_SECTION%/dns_nameservers**

작은 정보

CloudName{network.name} 정의를 사용하여 가상 IP를 사용하는 구성 가능 네트워크에서 API 끝점의 DNS 이름을 설정할 수 있습니다.

자세한 내용은 *director* 를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 구성 가능 네트워크 추가 를 참조하십시오.

7.4. 오버클라우드 생성 중 환경 파일 추가

배포 프로세스에 환경 파일을 포함하려면 배포 명령 **openstack overcloud deploy** 와 함께 **-e** 옵션을 사용합니다. 다음 순서로 이 섹션의 환경 파일을 추가합니다.

- SSL/TLS를 활성화하는 환경 파일(**enable-tls.yaml**)
- DNS 호스트 이름을 설정하는 환경 파일 (**custom-domain.yaml**)
- 루트 인증 기관을 삽입할 환경 파일(**inject-trust-anchor-hiera.yaml**)
- 공용 끝점 매핑을 설정하는 환경 파일입니다.
 - 공용 끝점에 액세스하는 데 DNS 이름을 사용하는 경우 **/usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-dns.yaml**을 사용합니다.
 - 공용 끝점에 액세스하는 데 IP 주소를 사용하는 경우 **/usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-ip.yaml**을 사용합니다.

절차

- SSL/TLS 환경 파일을 포함하는 방법의 예로 다음 배포 명령 스니펫을 사용합니다.

```
$ openstack overcloud deploy --templates \
[...]\
-e /home/stack/templates/enable-tls.yaml \
-e ~/templates/custom-domain.yaml \
-e ~/templates/inject-trust-anchor-hiera.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-dns.yaml
```

7.5. 수동으로 SSL/TLS 인증서 업데이트

TLS-e(TLS-e) 프로세스에서 TLS에서 자동 생성되지 않는 자체 SSL/TLS 인증서를 사용하는 경우 다음 단계를 완료합니다.

절차

1. 다음 콘텐츠를 사용하여 heat 템플릿을 편집합니다.

- **enable-tls.yaml** 파일을 편집하고 **SSLCertificate**, **SSLKey** 및 **SSLIntermediateCertificate** 매개변수를 업데이트합니다.
- 인증 기관이 변경된 경우 **inject-trust-anchor-hiera.yaml** 파일을 편집하고 **CAMap** 매개변수를 업데이트합니다.

2. 배포 명령을 다시 실행합니다.

```
$ openstack overcloud deploy --templates \  
[...]  
-e /home/stack/templates/enable-tls.yaml \  
-e ~/templates/custom-domain.yaml \  
-e ~/templates/inject-trust-anchor-hiera.yaml \  
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-  
dns.yaml
```

3. director에서 각 컨트롤러에 대해 다음 명령을 실행합니다.

```
ssh tripleo-admin@<controller> sudo podman \  
restart $(podman ps --format="{{.Names}}" | grep -w -E 'haproxy(-bundle-.*-[0-9]+)?')
```

8장. 오버클라우드의 암호화를 위해 CRYOSTAT 키 사용

Cryostat는 **uuid** 를 대체하는 기본 토큰 공급자입니다. Cryostat 배포를 검토하고 토큰이 올바르게 작동하는지 테스트할 수 있습니다. Cryostat는 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/fernet-keys** 에 저장된 세 가지 유형의 키를 사용합니다. 번호가 가장 높은 디렉터리에는 새 토큰을 생성하고 기존 토큰을 해독하는 기본 키가 포함되어 있습니다.

8.1. CRYOSTAT 배포 검토

Cryostat 토큰이 올바르게 작동하는지 테스트하려면 컨트롤러 노드의 IP 주소를 검색하고 컨트롤러 노드에 SSH를 연결하고 토큰 드라이버 및 공급자 설정을 검토합니다.

절차

1. 컨트롤러 노드의 IP 주소를 검색합니다.

```
[stack@director ~]$ source ~/stackrc
[stack@director ~]$ openstack server list
```

ID	Name	Status	Networks
756bd73-e47b-46e6-959c-e24d7fb71328	overcloud-controller-0	ACTIVE	ctlplane=192.0.2.16
62b869df-1203-4d58-8e45-fac6cd4cfbee	overcloud-novacompute-0	ACTIVE	ctlplane=192.0.2.8

2. 컨트롤러 노드에 SSH를 실행합니다.

```
[tripleo-admin@overcloud-controller-0 ~]$ ssh tripleo-admin@192.0.2.16
```

3. 토큰 드라이버 및 공급자 설정의 값을 검색합니다.

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo crudini --get /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf token driver
sql
[tripleo-admin@overcloud-controller-0 ~]$ sudo crudini --get /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf token provider
fernet
```

4. Cryostat 공급자를 테스트합니다.

```
[tripleo-admin@overcloud-controller-0 ~]$ exit
[stack@director ~]$ source ~/overcloudrc
[stack@director ~]$ openstack token issue
```

Field	Value
expires	2016-09-20 05:26:17+00:00
id	gAAAAABX4LppE8vaiFZ992eah2i3edpO1aDFxIKZq6a_RJzxUx56QVKORrmW0-oZK3-Xuu2wcnpyq_eeK2SGLz250eLpZOzxKBR0GsoMfxJU8mEFF8NzfLNcbuS-iz7SV-N1re3XEywSDG90JcgwjQfXW-8jtCm-n3LL5laZexAYlw059T_-cd8

```
| project_id | 26156621d0d54fc39bf3adb98e63b63d |  
| user_id | 397daf32cadd490a8f3ac23a626ac06c |
```

```
-----  
-----+
```

결과에는 long Cryostat 토큰이 포함됩니다.

9장. RED HAT OPENSTACK PLATFORM의 연방 정보 처리 표준

FIPS(Federal Information Processing Standards)는 NIST(National Institute of Standards and Technology)에서 개발한 일련의 보안 요구 사항입니다. Red Hat Enterprise Linux 9에서 지원되는 표준은 FIPS 게시 140-3: 암호화 모듈에 대한 보안 요구 사항입니다. 지원되는 표준에 대한 자세한 내용은 [연방 정보 처리 표준 발행 140-3](#)를 참조하십시오.

이러한 보안 요구 사항은 허용되는 암호화 알고리즘과 보안 모듈을 포함하여 암호화 알고리즘을 정의합니다.

- FIPS 140-3 검증은 FIPS를 통해 승인되는 암호화 알고리즘, 규정된 방식으로 및 검증된 모듈을 통해 승인되는 암호화 알고리즘만 사용하여 수행됩니다.
- FIPS 140-3 호환성은 FIPS를 통해 승인된 암호화 알고리즘만 사용하여 수행됩니다.

Red Hat OpenStack Platform 17은 **FIPS 140-3와 호환됩니다**. Red Hat에서 제공하는 이미지를 사용하여 오버클라우드를 배포하여 FIPS 호환성을 활용할 수 있습니다.



참고

OpenStack 17.1은 Red Hat Enterprise Linux (RHEL) 9.2를 기반으로 합니다. FIPS 검증을 위해 RHEL 9.2가 아직 제출되지 않았습니다. Red Hat은 RHEL 9.0 및 RHEL 9.2 모듈에 대한 FIPS 검증을 받기 위해 특정 기간 동안 커밋할 수는 없지만 RHEL 9.x의 마이너 릴리스도 필요할 것으로 예상합니다. 업데이트는 [규정 준수 활동 및 정부 표준에서 사용할 수 있습니다](#).

9.1. FIPS 활성화

FIPS를 활성화하는 경우 언더클라우드 및 오버클라우드를 설치하는 동안 일련의 단계를 완료해야 합니다.

사전 요구 사항

- Red Hat Enterprise Linux를 설치했으며 Red Hat OpenStack Platform director 설치를 시작할 준비가 되어 있습니다.
- Red Hat Ceph Storage 6 이상에서 Red Hat Ceph Storage를 스토리지 백엔드로 사용하는 경우 배포되었습니다.

프로세스

1. 언더클라우드에서 FIPS를 활성화합니다.
 - a. 언더클라우드를 설치하려는 시스템에서 FIPS를 활성화합니다.

```
fips-mode-setup --enable
```



참고

이 단계에서는 **fips=1** 커널 매개 변수가 GRUB 설정 파일에 추가됩니다. 결과적으로 Red Hat Enterprise Linux에서 사용하는 암호화 알고리즘 모듈만 FIPS 모드에 있으며 표준에 의해 승인되는 암호화 알고리즘만 사용됩니다.

- b. 시스템을 재부팅합니다.

- c. FIPS가 활성화되었는지 확인합니다.

```
fips-mode-setup --check
```

- d. Red Hat OpenStack Platform director를 설치하고 구성합니다. 자세한 내용은 [언더클라우드에 director 설치](#)를 참조하십시오.

2. 오버클라우드에 대한 FIPS 지원 이미지를 준비합니다.

- a. 오버클라우드 이미지를 설치합니다.

```
sudo dnf -y install rhosp-director-images-uefi-fips-x86_64
```

- b. **stack** 사용자의 홈 디렉터리에 **images** 디렉터리를 생성합니다.

```
$ mkdir /home/stack/images
$ cd /home/stack/images
```

- c. 이미지를 홈 디렉터리에 추출합니다.

```
for i in /usr/share/rhosp-director-images/*fips*.tar; do tar -xvf $i; done
```

- d. 이미지를 업로드하기 전에 심볼릭 링크를 생성해야 합니다.

```
ln -s ironic-python-agent-fips.initramfs ironic-python-agent.initramfs
ln -s ironic-python-agent-fips.kernel ironic-python-agent.kernel
ln -s overcloud-hardened-uefi-full-fips.qcow2 overcloud-hardened-uefi-full.qcow2
```

- e. FIPS 지원 오버클라우드 이미지를 이미지 서비스에 업로드합니다.

```
openstack overcloud image upload --update-existing --whole-disk
```



참고

현재 OpenStack 이미지 서비스에 이미지가 없는 경우에도 **--update-existing** 플래그를 사용해야 합니다.

3. 오버클라우드에서 FIPS를 활성화합니다.

사용자 환경과 관련된 오버클라우드 배포에 맞게 템플릿을 구성합니다. `fips.yaml`을 포함하여 배포 명령에 모든 구성 템플릿을 포함합니다.

```
openstack overcloud deploy
...
-e /usr/share/openstack-tripleo-heat-templates/environments/fips.yaml
```

10장. 사용자 액세스 보안 개선

Red Hat OpenStack Platform 17에서 보안 역할 기반 액세스 제어(SRBAC)를 활성화할 수 있습니다. SRBAC 모델에는 프로젝트 범위에 있는 세 가지 역할을 기반으로 하는 세 개의 가상 사용자가 있습니다.

10.1. SRBAC 가상 사용자

가상 사용자는 역할의 조합과 해당 역할이 속한 범위입니다. Red Hat OpenStack Platform 17을 배포할 때 프로젝트 범위에서 가상 사용자를 할당할 수 있습니다.

10.1.1. Red Hat OpenStack Platform SRBAC 역할

현재 프로젝트 범위 내에서 세 가지 다른 역할을 사용할 수 있습니다.

admin

admin 역할에는 리소스 또는 API에 대한 모든 생성, 읽기, 업데이트 또는 삭제 작업이 포함됩니다.

멤버

member 역할은 멤버가 속한 범위에 속하는 리소스를 생성, 읽기, 업데이트 및 삭제할 수 있습니다.

reader

reader 역할은 적용되는 범위에 관계없이 읽기 전용 작업에 사용됩니다. 이 역할은 적용되는 범위 전체에서 리소스를 볼 수 있습니다.

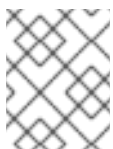
10.1.2. Red Hat OpenStack Platform SRBAC 범위

범위는 작업이 수행되는 컨텍스트입니다. Red Hat OpenStack Platform 17에서는 **프로젝트** 범위만 사용할 수 있습니다. **프로젝트** 범위는 OpenStack 내에서 격리된 셀프 서비스 리소스를 위한 API의 포함된 하위 집합입니다.

10.1.3. Red Hat OpenStack Platform SRBAC 가상 사용자

프로젝트 관리자

프로젝트 관리자 가상 사용자는 사용 가능한 유일한 관리자 개인이므로 Red Hat OpenStack Platform 17에는 프로젝트 관리자에게 최고 수준의 권한 부여를 부여하는 수정된 정책이 포함되어 있습니다. 이 가상 사용자는 프로젝트 간에 리소스에 대한 생성, 읽기, 업데이트 및 삭제 작업을 포함하며, 여기에는 사용자 및 기타 프로젝트 추가 및 제거가 포함됩니다.



참고

이 개인은 향후 개발 범위에 따라 변경될 것으로 예상됩니다. 이 역할은 프로젝트 멤버 및 프로젝트 리더에게 부여된 모든 권한을 의미합니다.

프로젝트 멤버

프로젝트 멤버 persona는 프로젝트 범위 내에서 리소스를 사용할 수 있는 권한이 부여된 사용자를 위한 것입니다. 이 사용자는 할당된 프로젝트 내에서 리소스를 생성, 나열, 업데이트 및 삭제할 수 있습니다. 이 사용자는 프로젝트 리더에게 부여된 모든 권한을 의미합니다.

프로젝트 리더

프로젝트 reader persona는 프로젝트에서 중요하지 않은 리소스를 볼 수 있는 권한이 부여된 사용자를 위한 것입니다. 프로젝트에서 리소스를 검사하거나 확인해야 하는 최종 사용자에게 reader 역할을 할당합니다. 감사자는 감사 목적으로 단일 프로젝트 내에서 프로젝트별 리소스만 볼 필요가 있는 감사 사용자에게 reader 역할을 할당하면 프로젝트 읽기 사용자가 모든 감사 사용 사례를 다루지는 않습니다.

시스템 또는 도메인 범위를 기반으로 하는 추가 가상 사용자는 개발 중이며 사용할 수 없습니다.



참고

Image 서비스(glance)는 metadef API에 대한 SRBAC 권한을 지원하지 않습니다. 이미지 서비스 metadef API의 RHOSP 17.1의 기본 정책은 관리자 전용입니다.

10.2. 보안 역할 기반 액세스 제어 활성화

보안 역할 기반 인증을 활성화하면 Red Hat OpenStack Platform 환경에서 사용자에게 할당된 권한 범위를 정의하는 새로운 정책 파일 세트를 활성화합니다.

사전 요구 사항

- Red Hat OpenStack Platform director 환경이 설치되어 있어야 합니다.

프로세스

- Red Hat OpenStack Platform을 배포할 때 배포 스크립트에 **enable-secure-rbac.yaml** 환경 파일을 포함합니다.

```
openstack overcloud deploy --templates
...
-e /usr/share/openstack-tripleo-heat-templates/environments/enable-secure-rbac.yaml
```

10.3. RBAC 환경에서 역할 할당

Red Hat OpenStack Platform에서 SRBAC를 사용하면 사용자를 **project-admin**, **project-member** 또는 **project-reader** 역할에 할당할 수 있습니다.

사전 요구 사항

- RBAC(보안 역할 기반 인증)를 통해 Red Hat OpenStack Platform을 배포했습니다.

프로세스

- 다음 구문을 사용하여 **openstack role add** 명령을 사용합니다.

- **admin** 역할을 할당합니다.

```
$ openstack role add --user <user> --user-domain <domain> --project <project> --
project-domain <project-domain> admin
```

- **멤버** 역할을 할당합니다.

```
$ openstack role add --user <user> --user-domain <domain> --project <project> --
project-domain <project-domain> member
```

- **reader** 역할을 할당합니다.

```
$ openstack role add --user <user> --user-domain <domain> --project <project> --
project-domain <project-domain> reader
```

- `<user>`를 기존 사용자로 교체하여 역할을 적용합니다.
- `<domain>`을 역할이 적용되는 도메인으로 바꿉니다.
- `<project>`를 사용자에게 역할을 부여하는 프로젝트로 바꿉니다.
- `<project-domain>`을 프로젝트가 있는 도메인으로 바꿉니다.

11장. POLICIES

각 OpenStack 서비스에는 액세스 정책에서 관리하는 리소스가 포함되어 있습니다. 예를 들어 리소스에는 다음 기능이 포함될 수 있습니다.

- 인스턴스 생성 및 시작 권한
- 인스턴스에 볼륨을 연결하는 기능

RHOSP(Red Hat OpenStack Platform) 관리자인 경우 사용자 지정 정책을 생성하여 다양한 수준의 액세스 권한이 있는 새 역할을 도입하거나 기존 역할의 기본 동작을 변경할 수 있습니다.



중요

Red Hat은 사용자 지정 역할 또는 정책을 지원하지 않습니다. 구문 오류 또는 잘못된 권한 부여는 보안 또는 유용성에 부정적인 영향을 미칠 수 있습니다. 프로덕션 환경에서 사용자 지정 역할 또는 정책이 필요한 경우 지원 예외는 Red Hat 지원팀에 문의하십시오.

11.1. 기존 정책 검토

서비스의 정책 파일은 일반적으로 **/etc/\$service** 디렉터리에 존재합니다. 예를 들어 Compute(nova)에 대한 **policy.json** 파일의 전체 경로는 **/etc/nova/policy.json** 입니다.

기존 정책을 찾는 방법에 영향을 미치는 두 가지 중요한 아키텍처 변경 사항이 있습니다.

- Red Hat OpenStack Platform은 이제 컨테이너화되었습니다.
 - 서비스 컨테이너 내부에서 이를 확인하는 경우 정책 파일이 기존 경로에 있습니다.
/etc/\$service/policy.json
 - 서비스 컨테이너 외부에서 해당 파일을 보면 정책 파일이 다음 경로에 있습니다.
/var/lib/config-data/puppet-generated/\$service/etc/\$service/policy.json
- 각 서비스에는 코드로 제공되는 기본 정책, 수동으로 생성한 경우에만 사용 가능한 파일이 있거나 **oslopolicy** 툴링을 사용하여 생성되는 경우 사용할 수 있습니다. 정책 파일을 생성하려면 다음 예와 같이 컨테이너 내에서 **oslopolicy-policy-generator** 를 사용합니다.

```
podman exec -it keystone oslopolicy-policy-generator --namespace keystone
```

기본적으로 생성된 정책은 oslo.policy CLI 툴에 의해 stdout으로 푸시됩니다.

11.2. 서비스 정책 이해

서비스 정책 파일 설명은 별칭 정의 또는 규칙입니다. 별칭 정의는 파일 상단에 있습니다. 다음 목록에는 Compute(nova)에 대해 생성된 **policy.json** 파일의 별칭 정의에 대한 설명이 포함되어 있습니다.

- "context_is_admin": "role:admin"
rule:context_is_admin 이 대상 뒤에 표시되면 해당 작업을 허용하기 전에 사용자가 관리 컨텍스트로 작동하는지 확인합니다.
- "admin_or_owner": "is_admin:True or project_id:\$(project_id)s"
대상 뒤에 **admin_or_owner** 가 표시되면 정책에서 사용자가 admin인지 또는 해당 작업을 허용하기 전에 프로젝트 ID가 대상 오브젝트의 소유 프로젝트 ID와 일치하는지 확인합니다.

- "admin_api": "is_admin:True"

대상 뒤에 **admin_api**가 표시되면 해당 작업을 허용하기 전에 사용자가 admin인지 확인합니다.

11.3. 정책 구문

policy.json 파일은 특정 연산자를 지원하므로 이러한 설정의 대상 범위를 제어할 수 있습니다. 예를 들어 다음 keystone 설정에는 관리자만 사용자를 생성할 수 있는 규칙이 포함되어 있습니다.

```
"identity:create_user": "rule:admin_required"
```

: 문자 왼쪽에 있는 섹션은 권한을 설명하고, 오른쪽 섹션은 권한을 사용할 수 있는 사람을 정의합니다. 또한 오른쪽에 연산자를 사용하여 범위를 추가로 제어할 수도 있습니다.

- **!** - 관리자(관리자 포함)는 이 작업을 수행할 수 없습니다.
- **@** 및 **""** - 모든 사용자가 이 작업을 수행할 수 있습니다.
- **not, and, 또는** - 표준 연산자 함수를 사용할 수 있습니다.

예를 들어 다음 설정은 사용자가 새 사용자를 생성할 수 있는 권한이 없음을 의미합니다.

```
"identity:create_user": "!"
```

11.4. 액세스 제어에 정책 파일 사용



중요

Red Hat은 사용자 지정 역할 또는 정책을 지원하지 않습니다. 구문 오류 또는 잘못된 권한 부여는 보안 또는 유용성에 부정적인 영향을 미칠 수 있습니다. 프로덕션 환경에서 사용자 지정 역할 또는 정책이 필요한 경우 지원 예외는 Red Hat 지원팀에 문의하십시오.

기본 규칙을 재정의하려면 적절한 OpenStack 서비스에 대한 **policy.json** 파일을 편집합니다. 예를 들어 Compute 서비스의 nova 디렉터리에는 컨테이너 내부에서 볼 때 컨테이너화된 서비스에 대한 파일의 올바른 위치인 **policy.json**이 있습니다.



참고

- 프로덕션 환경에서 구현하기 전에 스테이징 환경에서 정책 파일에 대한 변경 사항을 철저히 테스트해야 합니다.
- 액세스 제어 정책에 대한 변경 사항이 리소스의 보안을 의도하지 않게 손상시키지 않는지 확인해야 합니다. 또한 **policy.json** 파일에 대한 모든 변경 사항은 즉시 적용되며 서비스를 다시 시작할 필요가 없습니다.

예: 고급 사용자 역할 생성

keystone 역할의 권한을 사용자 지정하려면 서비스의 **policy.json** 파일을 업데이트합니다. 즉, 사용자 클래스에 할당하는 권한을 보다 세밀하게 정의할 수 있습니다. 이 예제에서는 다음 권한으로 배포에 대한 고급 사용자 역할을 생성합니다.

- 인스턴스를 시작합니다.
- 인스턴스를 중지합니다.

- 인스턴스에 연결된 볼륨을 관리합니다.

이 역할의 목적은 특정 사용자에게 **관리자** 액세스 권한을 부여하지 않고 추가 권한을 부여하는 것입니다. 이러한 권한을 사용하려면 사용자 지정 역할에 다음 권한을 부여해야 합니다.

- 인스턴스를 시작합니다. **"os_compute_api:servers:start": "role:PowerUsers"**
- 인스턴스를 중지: **"os_compute_api:servers:stop": "role:PowerUsers"**
- 특정 볼륨을 사용하도록 인스턴스를 구성합니다.
"os_compute_api:servers:create:attach_volume": "role:PowerUsers"
- 인스턴스에 연결된 볼륨을 나열합니다. **"os_compute_api:os-volumes-attachments:index": "role:PowerUsers"**
- 볼륨 연결: **"os_compute_api:os-volumes-attachments:create": "role:PowerUsers"**
- 연결된 볼륨의 세부 정보 보기: **"os_compute_api:os-volumes-attachments:show": "role:PowerUsers"**
- 인스턴스에 연결된 볼륨 변경: **"os_compute_api:os-volumes-attachments:update": "role:PowerUsers"**
- 인스턴스에 연결된 볼륨 삭제: **"os_compute_api:os-volumes-attachments:delete": "role:PowerUsers"**



참고

policy.json 파일을 수정할 때 기본 정책을 재정의합니다. 결과적으로 **PowerUsers**의 멤버는 이러한 작업을 수행할 수 있는 유일한 사용자입니다. **admin** 사용자가 이러한 권한을 보유할 수 있도록 하려면 *admin_or_power_user*에 대한 규칙을 만들 수 있습니다. 일부 기본 조건부 논리를 사용하여 **역할:PowerUsers** 또는 **role:Admin**을 정의할 수도 있습니다.

1. 사용자 지정 keystone 역할을 생성합니다.

```
$ openstack role create PowerUsers
+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | None |
| id | 7061a395af43455e9057ab631ad49449 |
| name | PowerUsers |
+-----+-----+
```

2. 기존 사용자를 역할에 추가하고 프로젝트에 역할을 할당합니다.

```
$ openstack role add --project [PROJECT_NAME] --user [USER_ID] [PowerUsers-ROLE_ID]
```



참고

역할 할당은 하나의 프로젝트와만 페어링됩니다. 즉, 사용자에게 역할을 할당할 때 동시에 대상 프로젝트도 정의합니다. 사용자가 동일한 역할을 수신하도록 하려면 다른 프로젝트에 대해 역할을 다시 할당해야 하지만 다른 프로젝트를 대상으로 하는 경우 역할을 다시 할당해야 합니다.

3. 기본 nova 정책 설정을 확인합니다.

```
$ oslopolicy-policy-generator --namespace nova
```

4. `/var/lib/config-data/puppet-generated/nova/etc/nova/policy.json`에 다음 항목을 추가하여 새 **PowerUsers** 역할에 대한 사용자 지정 권한을 생성합니다.



참고

배포 전에 정책 변경 사항을 테스트하여 예상대로 작동하는지 확인합니다.

```
{
  "os_compute_api:servers:start": "role:PowerUsers",
  "os_compute_api:servers:stop": "role:PowerUsers",
  "os_compute_api:servers:create:attach_volume": "role:PowerUsers",
  "os_compute_api:os-volumes-attachments:index": "role:PowerUsers",
  "os_compute_api:os-volumes-attachments:create": "role:PowerUsers",
  "os_compute_api:os-volumes-attachments:show": "role:PowerUsers",
  "os_compute_api:os-volumes-attachments:update": "role:PowerUsers",
  "os_compute_api:os-volumes-attachments:delete": "role:PowerUsers"
}
```

이 파일을 저장할 때 변경 사항을 구현하고 nova 컨테이너를 다시 시작합니다. **PowerUsers** keystone 역할에 추가된 사용자에게는 이러한 권한이 부여됩니다.

11.5. 예: 특성을 기반으로 액세스 제한



중요

Red Hat은 사용자 지정 역할 또는 정책을 지원하지 않습니다. 구문 오류 또는 잘못된 권한 부여는 보안 또는 유용성에 부정적인 영향을 미칠 수 있습니다. 프로덕션 환경에서 사용자 지정 역할 또는 정책이 필요한 경우 지원 예외는 Red Hat 지원팀에 문의하십시오.

해당 API 호출을 수행하는 사용자의 특성에 따라 API 호출에 대한 액세스를 제한하는 정책을 생성할 수 있습니다. 예를 들어 다음 기본 규칙은 관리 컨텍스트에서 실행되는 경우 또는 토큰의 사용자 ID가 대상과 연결된 사용자 ID와 일치하는 경우 키 쌍 삭제가 허용됨을 나타냅니다.

```
"os_compute_api:os-keypairs:delete": "rule:admin_api or user_id:%(user_id)s"
```

참고: * 새로 구현된 기능은 각 릴리스마다 모든 서비스에 보장되는 것은 아닙니다. 따라서 대상 서비스의 기존 정책의 규칙을 사용하여 규칙을 작성하는 것이 중요합니다. 이러한 정책 보기에 대한 자세한 내용은 기존 정책 검토를 참조하십시오. * 모든 정책은 릴리스 간 호환성을 보장하지 않으므로 배포될 모든 버전에 대해 프로덕션이 아닌 환경에서 엄격하게 테스트해야 합니다.

위의 예제를 기반으로 리소스를 소유하고 있는지 여부에 따라 사용자에게 액세스를 확장하거나 제한하도록 API 규칙을 만들 수 있습니다. 또한 아래 예제와 같이 속성을 다른 제한 사항과 결합하여 규칙을 형성할 수 있습니다.

```
"admin_or_owner": "is_admin:True or project_id:%(project_id)s"
```

위의 예제를 고려할 때 관리자와 사용자로 제한되는 고유한 규칙을 생성한 다음 해당 규칙을 사용하여 추가 작업을 제한할 수 있습니다.

```
"admin_or_user": "is_admin:True or user_id:%(user_id)s"
"os_compute_api:os-instance-actions": "rule:admin_or_user"
```

추가 리소스

- [정책 구문](#).

11.6. HEAT로 정책 수정



중요

Red Hat은 사용자 지정 역할 또는 정책을 지원하지 않습니다. 구문 오류 또는 잘못된 권한 부여는 보안 또는 유용성에 부정적인 영향을 미칠 수 있습니다. 프로덕션 환경에서 사용자 지정 역할 또는 정책이 필요한 경우 지원 예외는 Red Hat 지원팀에 문의하십시오.

heat를 사용하여 오버클라우드의 특정 서비스에 대한 액세스 정책을 구성할 수 있습니다. 다음 매개변수를 사용하여 해당 서비스에 정책을 설정합니다.

표 11.1. 정책 매개변수

매개변수	설명
KeystonePolicies	OpenStack Identity(keystone)에 대해 구성할 정책의 해시입니다.
IronicApiPolicies	OpenStack Bare Metal(ironic) API에 대해 구성할 정책의 해시입니다.
BarbicanPolicies	OpenStack Key Manager(barbican)에 대해 구성할 정책의 해시입니다.
NeutronApiPolicies	OpenStack Networking(neutron) API에 대해 구성할 정책의 해시입니다.
SaharaApiPolicies	OpenStack Cryostat(sahara) API에 대해 구성할 정책의 해시입니다.
NovaApiPolicies	OpenStack Compute(nova) API에 대해 구성할 정책의 해시입니다.
CinderApiPolicies	OpenStack Block Storage(cinder) API에 대해 구성할 정책의 해시입니다.
GlanceApiPolicies	OpenStack Image Storage(glance) API에 대해 구성할 정책의 해시입니다.
HeatApiPolicies	OpenStack Orchestration(heat) API에 대해 구성할 정책의 해시입니다.

서비스에 대한 정책을 구성하려면 정책 매개변수에 서비스의 정책이 포함된 해시 값을 지정합니다. 예를 들면 다음과 같습니다.

- OpenStack Identity(keystone)는 **KeystonePolicies** 매개 변수를 사용합니다. 환경 파일의 **parameter_defaults** 섹션에서 이 매개변수를 설정합니다.

```
parameter_defaults:
  KeystonePolicies: { keystone-context_is_admin: { key: context_is_admin, value: 'role:admin'
  } }
```

- OpenStack Compute(nova)는 **NovaApiPolicies** 매개변수를 사용합니다. 환경 파일의 **parameter_defaults** 섹션에서 이 매개변수를 설정합니다.

```
parameter_defaults:
  NovaApiPolicies: { nova-context_is_admin: { key: 'compute:get_all', value: '@' } }
```

11.7. 사용자 및 역할 감사

Red Hat OpenStack Platform에서 사용할 수 있는 툴을 사용하여 사용자 및 관련 권한당 역할 할당 보고서를 작성할 수 있습니다.

사전 요구 사항

- Red Hat OpenStack Platform 환경이 설치되어 있어야 합니다.
- stack으로 director에 로그인되어 있습니다.

프로세스

1. **openstack role list** 명령을 실행하여 현재 환경에 있는 역할을 확인합니다.

```
openstack role list -c Name -f value

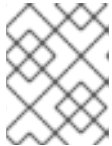
swiftoperator
ResellerAdmin
admin
_member_
heat_stack_user
```

2. **openstack role assignment list** 명령을 실행하여 특정 역할의 멤버인 모든 사용자를 나열합니다. 예를 들어 admin 역할이 있는 모든 사용자를 보려면 다음을 실행합니다.

```
$ openstack role assignment list --names --role admin

+-----+-----+-----+-----+-----+-----+-----+
| Role | User                | Group | Project    | Domain  | System | Inherited |
+-----+-----+-----+-----+-----+-----+-----+
| admin | heat-cfn@Default    |      | service@Default |        |        | False     |
| admin | placement@Default   |      | service@Default |        |        | False     |
| admin | neutron@Default     |      | service@Default |        |        | False     |
| admin | zaqar@Default        |      | service@Default |        |        | False     |
| admin | swift@Default        |      | service@Default |        |        | False     |
| admin | admin@Default        |      | admin@Default   |        |        | False     |
| admin | zaqar-websocket@Default |    | service@Default |        |        | False     |
| admin | heat@Default         |      | service@Default |        |        | False     |
```

```
| admin | ironic-inspector@Default | | service@Default | | | False |
| admin | nova@Default | | service@Default | | | False |
| admin | ironic@Default | | service@Default | | | False |
| admin | glance@Default | | service@Default | | | False |
| admin | mistral@Default | | service@Default | | | False |
| admin | heat_stack_domain_admin@heat_stack | | | heat_stack | | False |
|
| admin | admin@Default | | | | all | False |
+-----+-----+-----+-----+-----+-----+-----+-----+
```



참고

-f {csv,json,table,value,yaml} 매개변수를 사용하여 이러한 결과를 내보낼 수 있습니다.

11.8. API 액세스 감사

지정된 역할에서 액세스할 수 있는 API 호출을 감사할 수 있습니다. 각 역할에 대해 이 프로세스를 반복하면 각 역할에 대해 액세스 가능한 API에 대한 포괄적인 보고서가 생성됩니다.

사전 요구 사항

- 대상 역할에서 사용자로 소싱할 인증 파일입니다.
- JSON 형식의 액세스 토큰입니다.
- 감사하려는 각 서비스의 API에 대한 정책 파일입니다.

절차

1. 원하는 역할에서 사용자의 인증 파일을 가져와 시작합니다.
2. Keystone에서 생성된 토큰을 캡처하고 파일에 저장합니다. `openstack-cli` 명령을 실행하고 제공된 토큰을 stdout에 출력하는 `--debug` 옵션을 사용하여 이 작업을 수행할 수 있습니다. 이 토큰을 복사하여 액세스 파일에 저장할 수 있습니다. 다음 명령을 사용하여 이를 단일 단계로 수행합니다.

```
openstack token issue --debug 2>&1 | egrep ^{"token\":" > access.file.json
```

3. 정책 파일을 생성합니다. 이는 관심 있는 컨테이너화된 서비스를 호스팅하는 오버클라우드 노드에서 수행할 수 있습니다. 다음 예제에서는 cinder 서비스에 대한 정책 파일을 생성합니다.

```
ssh tripleo-admin@CONTROLLER-1 sudo podman exec cinder_api \
oslopolicy-policy-generator \
--config-file /etc/cinder/cinder.conf \
--namespace cinder > cinderpolicy.json
```

4. 이러한 파일을 사용하여 cinder의 API에 액세스하기 위해 해당 역할을 감사할 수 있습니다.

```
oslopolicy-checker --policy cinderpolicy.json --access access.file.json
```

12장. 네트워크 시간 프로토콜

Red Hat OpenStack Platform 클러스터 내의 시스템이 시스템 간 정확하고 일관된 타임스탬프를 갖도록 해야 합니다.

Red Hat Enterprise Linux 9의 Red Hat OpenStack Platform은 시간 관리를 위해 Chrony를 지원합니다. 자세한 내용은 [Chrony 모음을 사용하여 NTP 구성을](#) 참조하십시오.

12.1. 일관된 시간이 중요한 이유

조직 전체의 일관된 시간은 운영 및 보안 요구 사항 모두에 중요합니다.

보안 이벤트 식별

일관된 시간 유지를 통해 영향을 받는 시스템의 이벤트 타임스탬프를 연결하여 이벤트 순서를 파악할 수 있습니다.

인증 및 보안 시스템

보안 시스템은 시간 스큐에 민감할 수 있습니다. 예를 들면 다음과 같습니다.

- kerberos 기반 인증 시스템은 몇 초의 클럭 스큐에 의해 영향을 받는 클라이언트 인증을 거부할 수 있습니다.
- TLS(Transport Layer Security) 인증서는 유효한 시간 소스에 따라 달라집니다. 클라이언트와 서버 시스템 시간 간의 차이가 *Valid From* 날짜 범위를 초과하면 클라이언트의 TLS 연결이 실패합니다.

Red Hat OpenStack Platform 서비스

일부 핵심 OpenStack 서비스는 특히 HA(고가용성) 및 Ceph를 포함한 정확한 시간 관리에 따라 달라집니다.

12.2. NTP 설계

NTP(Network Time Protocol)는 계층 구조 설계로 구성됩니다. 각 레이어를 stratum이라고 합니다. 계층 구조 맨 위에는 원자 클럭과 같은 계층 0 장치가 있습니다. NTP 계층 구조에서 stratum 0 장치는 공개적으로 사용 가능한 계층 1 및 계층 2 NTP 시간 서버에 대한 참조를 제공합니다.

데이터 센터 클라이언트를 공개적으로 사용 가능한 NTP 계층 1 또는 2 서버에 직접 연결하지 마십시오. 직접 연결 수가 공용 NTP 리소스에 불필요한 부담을 초래했습니다. 대신 데이터 센터에 전용 시간 서버를 할당하고 클라이언트를 해당 전용 서버에 연결합니다.

인스턴스가 상주하는 호스트가 아니라 전용 시간 서버에서 시간을 수신하도록 구성합니다.



참고

Red Hat OpenStack Platform 환경에서 실행되는 서비스 컨테이너는 여전히 해당 환경에서 실행되는 호스트에서 시간을 수신합니다.

13장. 인프라 및 가상화 강화

물리적 및 가상 환경을 강화하여 내부 및 외부 위협으로부터 효과적으로 보호할 수 있습니다.

13.1. RED HAT OPENSTACK PLATFORM용 HARWARE

클라우드 환경에서 사용할 하드웨어를 추가할 때 하드웨어 가상화가 지원되는지 확인하십시오. 사용하지 않는 하드웨어 기능을 비활성화합니다.

프로세스

1. [Red Hat OpenStack의 인증된 하드웨어](#)를 확인하여 하드웨어가 지원되는지 확인합니다.
2. 하드웨어 가상화가 사용 가능하고 활성화되어 있는지 확인합니다.

```
cat /proc/cpuinfo | egrep "vmx|svm"
```

3. 하드웨어 플랫폼에서 모든 펌웨어가 최신 상태인지 확인합니다. 자세한 내용은 하드웨어 벤더 설명서를 참조하십시오.

13.2. 클라우드 환경의 소프트웨어 업데이트

보안, 성능 및 지원 가능성을 위해 RHOSP(Red Hat OpenStack Platform)를 업데이트하십시오.

- 업데이트할 때 커널 업데이트가 포함된 경우 업데이트된 물리적 시스템 또는 인스턴스를 재부팅해야 합니다.
- OpenStack Image(glance) 이미지를 업데이트하여 새로 생성된 인스턴스에 최신 업데이트가 있는지 확인합니다.
- RHOSP에서 패키지를 선택적으로 업데이트하는 경우 모든 보안 업데이트가 포함되어 있는지 확인합니다. 최신 취약점 및 보안 업데이트에 대한 자세한 내용은 다음을 참조하십시오.
 - [RHSA-announce](#)
 - [에라타 알림](#)
 - [보안 권고](#)

13.3. OPENSTACK 환경에서 SSH 키 업데이트

또한 다음 시나리오에서 SSH 키가 2048비트 미만인 경우 업데이트해야 합니다.

- RHOSP 16.2에서 17.1로 업그레이드하는 동안 RHOSP(Red Hat OpenStack Platform) 클러스터를 RHEL 9.2로 업그레이드하고 있습니다.
- RHOSP 17.0 또는 17.1 마이너 릴리스에서 최신 RHOSP 17.1 마이너 릴리스로 업데이트 중입니다.

`ssh_key_rotation.yaml` Ansible 플레이북을 실행하여 SSH 키를 순환하는 프로세스를 안전하게 자동화합니다. 오버클라우드에서 백업 키는 다음 디렉터리에 저장됩니다.

```
/home/{{ ansible_user_id }}/backup_keys/{{ ansible_date_time.epoch }}/authorized_keys"
```

사전 요구 사항

- 완전히 설치된 RHOSP 환경이 있습니다.

프로세스

1. RHOSP director에 로그인합니다.

```
ssh stack@director
```

2. ansible-playbook **ssh_key_rotation.yaml**을 실행합니다.

```
$ ansible-playbook \
-i /home/stack/overcloud-deploy/<stack_name>/tripleo-ansible-inventory.yaml \ ❶
-e undercloud_backup_folder=/home/stack/overcloud_backup_keys \ ❷
-e stack_name=<stack_name> \ ❸
/usr/share/ansible/tripleo-playbooks/ssh_key_rotation.yaml
```

- ❶ <stack_name>을 오버클라우드 스택의 이름으로 바꿉니다.
- ❷ **stack** 홈 디렉터리에서 선택한 백업 디렉터를 지정합니다.
- ❸ SSH 키의 경로가 포함된 파일을 찾습니다. 플레이북에서 SSH 키를 찾을 수 없는 경우 <stack_name> 매개변수를 설정하지 **않고** **keys_folder** 및 **keys_name** 매개변수를 설정하여 SSH 키 위치를 지정할 수 있습니다. 예를 들어 SSH 키 위치가 **/home/stack/.ssh/id_rsa** 인 경우 다음 값을 설정합니다.

- **keys_folder=/home/stack/.ssh**
- **keys_name=id_rsa**



참고

단일 셀링 배포가 있는 경우 이 절차를 완료하십시오. 둘 이상의 셀이 있는 경우 계속해야 합니다.

3. 다음을 포함하는 각 추가 셀에 대해 플레이북을 재실행합니다.

```
ansible-playbook \
-i /home/stack/overcloud-deploy/<stack_name>/tripleo-ansible-inventory.yaml \ ❶
-e stack_name=<stack_name> \ ❷
-e rotate_undercloud_key=false \ ❸
-e ansible_ssh_private_key_file=/home/stack/overcloud_backup_keys/id_rsa ❹
tripleo-ansible/playbooks/ssh_key_rotation.yaml
```

- ❶ <stack_name>을 셀의 스택 이름으로 바꿉니다. 각 셀마다 다른 이름이 있습니다.
- ❷ SSH 키의 경로가 포함된 파일을 찾습니다. 플레이북에서 SSH 키를 찾을 수 없는 경우 <stack_name> 매개변수를 설정하지 **않고** **keys_folder** 및 **keys_name** 매개변수를 설정하여 SSH 키 위치를 지정할 수 있습니다. 예를 들어 SSH 키 위치가 **/home/stack/.ssh/id_rsa** 인 경우 다음 값을 설정합니다.

- **keys_folder=/home/stack/.ssh**
- **keys_name=id_rsa**

- 3 **rotate_undercloud_key** 매개변수를 **false** 로 설정하여 언더클라우드 키가 교체 되지 않았는지 확인해야 합니다.
- 4 이전 SSH 키가 순환된 후 다른 셀의 Compute 호스트에 인증할 수 있도록 SSH 백업 키를 가리킵니다.

13.4. 하드웨어 및 소프트웨어 기능 제한

공격 가능성에 더 적은 코드가 노출되도록 사용하는 하드웨어 및 소프트웨어 기능만 활성화합니다. 신뢰할 수 있는 환경에서만 활성화해야 하는 몇 가지 기능이 있습니다.

PCI 패스스루

PCI 통과를 사용하면 인스턴스가 노드의 PCI 장치에 직접 액세스할 수 있습니다. PCI 장치 액세스가 가능한 인스턴스는 악의적인 행위자가 펌웨어를 수정할 수 있도록 허용할 수 있습니다. 또한 일부 PCI 장치에는 직접 메모리 액세스(DMA)가 있습니다. Cryostat를 사용하여 인스턴스를 제어하면 임의의 물리적 메모리 액세스를 얻을 수 있습니다.

NFV(Network Functions Virtualization)와 같은 특정 사용 사례에 대해 PCI 패스스루를 활성화해야 합니다. 배포에 필요하지 않는 한 PCI 패스스루를 활성화하지 마십시오.

커널 동일 페이지 병합

KSM(커널 동일 페이지 병합)은 메모리 페이지의 중복 제거 및 공유를 통해 메모리 사용을 줄이는 기능입니다. 메모리에 두 개 이상의 가상 머신이 있는 경우 해당 페이지를 공유하여 더 높은 밀도를 허용할 수 있습니다. 메모리 중복 제거 전략은 사이드 채널 공격에 취약하며 신뢰할 수 있는 환경에서만 사용해야 합니다. Red Hat OpenStack Platform에서 KSM은 기본적으로 비활성화되어 있습니다.

13.5. RED HAT OPENSTACK PLATFORM의 SELINUX

SELinux(Security-Enhanced Linux)는 MAC(필수 액세스 제어) 구현입니다. MAC은 시스템에서 프로세스 또는 애플리케이션을 수행할 수 있는 작업을 제한하여 공격의 영향을 제한합니다. SELinux에 대한 자세한 내용은 [SELinux란 무엇입니까?](#).

RHOSP(Red Hat OpenStack Platform) 서비스에 대해 SELinux 정책이 미리 구성되어 있습니다. RHOSP에서 SELinux는 별도의 보안 컨텍스트에서 각 QEMU 프로세스를 실행하도록 구성됩니다. RHOSP에서 SELinux 정책은 다음과 같은 위협으로부터 하이퍼바이저 호스트 및 인스턴스를 보호하는 데 도움이 됩니다.

하이퍼바이저 위협

인스턴스 내에서 실행되는 손상된 애플리케이션은 기본 리소스에 액세스하도록 하이퍼바이저를 공격합니다. 인스턴스가 하이퍼바이저 OS에 액세스할 수 있는 경우 물리적 장치 및 기타 애플리케이션이 대상이 될 수 있습니다. 이 위협은 상당한 위험을 나타냅니다. 하이퍼바이저의 손상으로 인해 펌웨어, 기타 인스턴스 및 네트워크 리소스도 손상될 수 있습니다.

인스턴스 위협

인스턴스 내에서 실행되는 손상된 애플리케이션은 하이퍼바이저를 공격하여 다른 인스턴스 및 해당 리소스 또는 인스턴스 파일 이미지에 액세스하거나 제어합니다. 실제 네트워크를 보호하기 위한 관리 전략은 가상 환경에 직접 적용되지 않습니다. 모든 인스턴스는 SELinux에서 레이블이 지정된 프로세스이므로 Linux 커널에서 적용되는 각 인스턴스 주위에 보안 경계가 있습니다.

RHOSP에서 디스크의 인스턴스 이미지 파일은 SELinux 데이터 유형 **svirt_image_t** 로 레이블이 지정됩니다. 인스턴스의 전원이 켜지면 SELinux는 임의의 숫자 식별자를 이미지에 추가합니다. 임의의 숫자 식별자로 인해 손상된 OpenStack 인스턴스가 다른 컨테이너에 대한 무단 액세스 권한을 얻지 못할 수 있습니다. SELinux는 각 하이퍼바이저 노드에 최대 524,288개의 숫자 식별자를 할당할 수 있습니다.

13.6. 컨테이너화된 서비스 조사

Red Hat OpenStack Platform과 함께 제공되는 OpenStack 서비스는 컨테이너 내에서 실행됩니다. 컨테이너화를 사용하면 종속성 관련 충돌 없이 서비스를 개발 및 업그레이드할 수 있습니다. 서비스가 컨테이너 내에서 실행되면 해당 서비스에 대한 잠재적인 취약점도 포함됩니다.

다음 단계를 사용하여 환경에서 실행 중인 서비스에 대한 정보를 가져올 수 있습니다.

프로세스

- 'podman inspect'를 사용하여 마운트된 호스트 디렉터리 바인딩과 같은 정보를 가져옵니다.
예제:

```
$ sudo podman inspect <container_name> | less
```

<container_name>을 컨테이너 이름으로 바꿉니다. 예를 들면 **nova compute** 입니다.

- **/var/log/containers:**에 있는 서비스의 로그를 확인합니다.
예제:

```
sudo less /var/log/containers/nova/nova-compute.log
```

- 컨테이너 내에서 대화형 CLI 세션을 실행합니다.
예제:

```
podman exec -it nova_compute /bin/bash
```



참고

컨테이너 내에서 직접 테스트 목적으로 서비스를 변경할 수 있습니다. 컨테이너를 다시 시작하면 모든 변경 사항이 손실됩니다.

13.7. 컨테이너화된 서비스를 임시로 변경

컨테이너를 다시 시작할 때 지속되는 컨테이너화된 서비스를 변경할 수 있지만 RHOSP(Red Hat OpenStack Platform) 클러스터의 영구 구성에는 영향을 미치지 않습니다. 이 기능은 구성 변경 사항을 테스트하거나 문제 해결 시 디버그 수준 로그를 활성화하는 데 유용합니다. 변경 사항을 수동으로 되돌릴 수 있습니다. 또는 RHOSP 클러스터에서 재배포를 실행하면 모든 매개변수가 영구 구성으로 재설정됩니다.

/var/lib/config-data/puppet-generated/[service]에 있는 구성 파일을 사용하여 서비스를 일시적으로 변경합니다. 다음 예제에서는 nova 서비스에서 디버깅을 활성화합니다.

프로세스

1. **nova_compute** 컨테이너에 마운트된 **nova.conf** 구성 파일을 편집합니다. **debug** 매개변수의 값을 **True**로 설정합니다.

```
$ sudo sed -i 's/^debug=.*debug=True'\n/var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf'
```



주의

OpenStack 파일의 구성 파일은 **[DEFAULT]** 및 **[database]** 와 같은 여러 섹션이 있는 **ini** 파일입니다. 각 섹션에 고유한 매개 변수는 전체 파일에서 고유하지 않을 수 있습니다. **sed** 를 주의해서 사용합니다. **egrep -v "^\$|^#"** **[configuration_file] | grep [parameter]**를 실행하여 구성 파일에서 매개 변수가 두 번 이상 표시되는지 확인할 수 있습니다.

2. nova 컨테이너를 다시 시작합니다.

```
sudo podman restart nova_compute
```

13.8. 컨테이너화된 서비스를 영구적으로 변경

heat를 사용하여 RHOSP(Red Hat OpenStack Platform) 서비스에서 컨테이너화된 서비스를 영구적으로 변경할 수 있습니다. RHOSP를 처음 배포할 때 사용한 기존 템플릿을 사용하거나 배포 스크립트에 추가할 새 템플릿을 생성합니다. 다음 예제에서는 libvirt의 개인 키 크기가 **4096** 으로 증가했습니다.

프로세스

1. **libvirt-keysize.yaml** 이라는 새 yaml 템플릿을 생성하고 **LibvirtCertificateKeySize** 매개 변수를 사용하여 기본값을 **2048** 에서 **4096** 으로 늘립니다.

```
cat > /home/stack/templates/libvirt-keysize.yaml
parameter_defaults:
  LibvirtCertificateKeySize: 4096
EOF
```

2. **libvirt-keysize.yaml** 구성 파일을 배포 스크립트에 추가합니다.

```
openstack overcloud deploy --templates \
...
-e /home/stack/templates/libvirt-keysize.yaml
...
```

3. 배포 스크립트를 다시 실행합니다.

```
./deploy.sh
```

13.9. 펌웨어 업데이트

물리 서버는 복잡한 펌웨어를 사용하여 서버 하드웨어 및 표시 관리 카드를 활성화하고 운영하므로 자체 보안 취약점이 있어 시스템 액세스 및 중단을 허용할 수 있습니다. 이를 해결하기 위해 하드웨어 벤더는 운영 체제 업데이트와 별도로 설치된 펌웨어 업데이트를 발행합니다. 이러한 업데이트를 정기적으로 검색, 테스트 및 구현하는 운영 보안 프로세스가 필요합니다. 펌웨어 업데이트를 적용하려면 물리적 호스트를 재부팅해야 하는 경우가 많습니다.

13.10. SSH 배너 텍스트 사용

SSH를 통해 연결하는 모든 사용자에게 콘솔 메시지를 표시하는 배너를 설정할 수 있습니다. 환경 파일에서 다음 매개변수를 사용하여 배너 텍스트를 **/etc/issue**에 추가할 수 있습니다. 요구 사항에 맞게 이 샘플 텍스트를 사용자 정의하는 것이 좋습니다.

```
resource_registry:
  OS::TripleO::Services::Sshd:
    /usr/share/openstack-tripleo-heat-templates/deployment/ssh/ssh-baremetal-puppet.yaml

parameter_defaults:
  BannerText: |
    *****
    * This system is for the use of authorized users only. Usage of *
    * this system may be monitored and recorded by system personnel. *
    * Anyone using this system expressly consents to such monitoring *
    * and is advised that if such monitoring reveals possible *
    * evidence of criminal activity, system personnel may provide *
    * the evidence from such monitoring to law enforcement officials.*
    *****
```

이 변경 사항을 배포에 적용하려면 설정을 **ssh_banner.yaml**이라는 파일로 저장한 다음 다음과 같이 **overcloud deploy** 명령에 전달합니다. **<full environment>**는 원래 배포 매개변수를 모두 포함해야 함을 나타냅니다. 예를 들면 다음과 같습니다.

```
openstack overcloud deploy --templates \
  -e <full environment> -e ssh_banner.yaml
```

13.11. 시스템 이벤트 감사

모든 감사 이벤트의 레코드를 유지 관리하면 시스템 기준선을 설정하거나 문제 해결을 수행하거나 특정 결과를 초래하는 이벤트 순서를 분석하는 데 도움이 됩니다. 감사 시스템은 시스템 시간 변경, 중간/거부 액세스 제어 변경, 사용자 또는 그룹 생성/삭제와 같은 많은 유형의 이벤트를 로깅할 수 있습니다.

환경 파일을 사용하여 규칙을 생성할 수 있습니다. 이 파일은 director에서 **/etc/audit/audit.rules**에 삽입합니다. 예를 들면 다음과 같습니다.

```
resource_registry:
  OS::TripleO::Services::AuditD: /usr/share/openstack-tripleo-heat-
    templates/deployment/auditd/auditd-baremetal-puppet.yaml
parameter_defaults:
  AuditdRules:
    'Record Events that Modify User/Group Information':
      content: '-w /etc/group -p wa -k audit_rules_usergroup_modification'
      order : 1
    'Collects System Administrator Actions':
      content: '-w /etc/sudoers -p wa -k actions'
      order : 2
    'Record Events that Modify the Systems Mandatory Access Controls':
      content: '-w /etc/selinux/ -p wa -k MAC-policy'
      order : 3
```

13.12. 방화벽 규칙 관리

방화벽 규칙은 배포 중에 오버클라우드 노드에 자동으로 적용되며, OpenStack 작동에 필요한 포트만 노출하기 위한 것입니다. 필요에 따라 추가 방화벽 규칙을 지정할 수 있습니다. 예를 들어 Zabbix 모니터링 시스템에 대한 규칙을 추가하려면 다음을 수행합니다.

```
parameter_defaults:
  ControllerExtraConfig:
    ExtraFirewallRules:
      '301 allow zabbix':
        dport: 10050
        proto: tcp
        source: 10.0.0.8
```



참고

action 매개변수를 설정하지 않으면 결과가 허용됩니다. **삭제,삽입** 또는 **추가** 하도록 **action** 매개변수만 설정할 수 있습니다.

엑세스를 제한하는 규칙을 추가할 수도 있습니다. 규칙 정의 중에 사용되는 숫자는 규칙의 우선 순위를 결정합니다. 예를 들어 RabbitMQ의 규칙 번호는 기본적으로 **109** 입니다. If you want to restrain it, you switch it to use a lower value:

```
parameter_defaults:
  ControllerParameters:
    ExtraFirewallRules:
      '098 allow rabbit from internalapi network':
        dport: [4369,5672,25672]
        proto: tcp
        source: 10.0.0.0/24
      '099 drop other rabbit access':
        dport: [4369,5672,25672]
        proto: tcp
        action: drop
```

이 예에서 **098** 및 **099** 는 RabbitMQ의 규칙 번호 **109** 보다 낮은 임의로 선택됩니다. 규칙 번호를 확인하려면 적절한 노드에서 iptables 규칙을 검사할 수 있습니다. RabbitMQ의 경우 컨트롤러를 확인합니다.

```
iptables-save
[...]
-A INPUT -p tcp -m multiport --dports 4369,5672,25672 -m comment --comment "109 rabbitmq" -m state --state NEW -j ACCEPT
```

또는 puppet 정의에서 포트 요구 사항을 추출할 수 있습니다. 예를 들어 RabbitMQ의 규칙은 **puppet/services/rabbitmq.yaml** 에 저장됩니다.

```
ExtraFirewallRules:
  '109 rabbitmq':
    dport:
      - 4369
      - 5672
      - 25672
```

규칙에 대해 다음 매개변수를 설정할 수 있습니다.

- **dport**: 규칙에 연결된 대상 포트입니다.
- **활동 활동**: 규칙과 관련된 소스 포트입니다.
- **Proto**: 규칙과 연결된 프로토콜입니다. 기본값은 *tcp*입니다.
- **action**: 규칙과 연결된 작업 정책입니다. 기본값은 *Cryo stat*이며 점프를 *ACCEPTS* 로 설정합니다.
- **State**: 규칙과 관련된 상태의 배열입니다. 기본값 [새로 설정]
- **Source**: 규칙과 연결된 소스 IP 주소입니다.
- **interface**: 규칙과 연결된 네트워크 인터페이스입니다.
- **chain**: 규칙과 연결된 체인입니다. 기본값은 *INPUT*
- **destination**: 규칙과 연결된 대상 cidr입니다.

13.13. AIDE를 사용한 침입 탐지

AIDE(Advanced Intrusion Detection Environment)는 파일 및 디렉터리 무결성 검사기입니다. 무단 파일 변조 또는 변경 사항을 감지하는 데 사용됩니다. 예를 들어 AIDE에서는 시스템 암호 파일이 변경되는 경우 이를 경고할 수 있습니다.

AIDE는 시스템 파일을 분석한 다음 파일 해시의 무결성 데이터베이스를 컴파일하여 작동합니다. 그런 다음 데이터베이스는 파일 및 디렉터리의 무결성을 확인하고 변경 사항을 감지하기 위한 비교 지점 역할을 합니다.

director에는 AIDE 서비스가 포함되어 있으므로 AIDE 구성에 항목을 추가할 수 있으며, AIDE 서비스에서 무결성 데이터베이스를 생성하는 데 사용됩니다. 예를 들면 다음과 같습니다.

```
resource_registry:
  OS::TripleO::Services::Aide:
    /usr/share/openstack-tripleo-heat-templates/deployment/aide/aide-baremetal-ansible.yaml

parameter_defaults:
  AideRules:
    'TripleORules':
      content: 'TripleORules = p+sha256'
      order: 1
    'etc':
      content: '/etc/ TripleORules'
      order: 2
    'boot':
      content: '/boot/ TripleORules'
      order: 3
    'sbin':
      content: '/sbin/ TripleORules'
      order: 4
    'var':
      content: '/var/ TripleORules'
      order: 5
    'not var/log':
      content: '!/var/log.*'
      order: 6
```

```
'not var/spool':
  content: '!/var/spool.*'
  order: 7
'not nova instances':
  content: '!/var/lib/nova/instances.*'
  order: 8
```



참고

위의 예제는 적극적으로 유지 관리되거나 벤치마킹되지 않으므로 요구 사항에 맞는 AIDE 값을 선택해야 합니다.

1. 매번 동일한 속성을 반복적으로 사용하지 않도록 **TripleORules** 라는 별칭이 선언됩니다.
2. 별칭은 **p+sha256** 의 속성을 수신합니다. AIDE 용어로 다음과 같은 지침이 표시됩니다. **sha256** 의 무결성 체크섬을 사용하여 모든 파일 권한 **p** 를 모니터링합니다.

AIDE의 구성 파일에 사용 가능한 전체 속성 목록은 <https://aide.github.io/> 에서 AIDE MAN 페이지를 참조하십시오.

배포에 변경 사항을 적용하려면 다음을 완료합니다.

1. 설정을 **/home/stack/templates/** 디렉터리에 **aide.yaml** 이라는 파일로 저장합니다.
2. **aide.yaml** 환경 파일을 편집하여 환경에 적합한 매개변수와 값을 지정합니다.
3. 환경과 관련된 기타 필요한 모든 heat 템플릿 및 환경 파일과 함께 **openstack overcloud deploy** 명령에 **/home/stack/templates/aide.yaml** 환경 파일을 포함합니다.

```
openstack overcloud deploy --templates
...
-e /home/stack/templates/aide.yaml
```

13.13.1. 복잡한 AIDE 규칙 사용

이전에 설명한 형식을 사용하여 복잡한 규칙을 생성할 수 있습니다. 예를 들면 다음과 같습니다.

```
MyAlias = p+i+n+u+g+s+b+m+c+sha512
```

이 명령은 체크섬 생성에 sha256을 사용하여 권한, inode, 링크 수, 사용자, 그룹, 크기, 블록 수, mtime, ctime 등의 명령어로 변환됩니다.

별칭에는 항상 **1** 의 순서 위치가 있어야 합니다. 즉, AIDE 규칙 상단에 배치되고 아래의 모든 값에 재귀적으로 적용됩니다.

별칭 뒤에는 모니터링할 디렉터리가 있습니다. 정규 표현식을 사용할 수 있습니다. 예를 들어 **var** 디렉터리에 대한 모니터링을 설정했지만 **!/var/log.*** 및 **!/var/spool.***를 사용하여 not 절으로 덮어씁니다.

13.13.2. 추가 AIDE 값

다음 AIDE 값도 사용할 수 있습니다.

AideConfPath: aide 구성 파일의 전체 POSIX 경로이며 기본값은 **/etc/aide.conf** 입니다. 파일 위치를 변경할 필요가 없는 경우 기본 경로를 사용하는 것이 좋습니다.

AideDBPath: AIDE 무결성 데이터베이스의 전체 POSIX 경로입니다. AIDE 데이터베이스 파일은 노드에서 읽기 전용 파일 마운트에 저장되므로 Operator가 자체 전체 경로를 선언할 수 있도록 이 값을 구성할 수 있습니다.

AideDBTempPath: AIDE 무결성 임시 데이터베이스의 전체 POSIX 경로입니다. 이 임시 파일은 AIDE에서 새 데이터베이스를 초기화할 때 생성됩니다.

AideHour: 이 값은 hour 속성을 AIDE cron 구성의 일부로 설정하는 것입니다.

AideMinute: 이 값은 minute 속성을 AIDE cron 구성의 일부로 설정하는 것입니다.

AideCronUser: 이 값은 linux 사용자를 AIDE cron 구성의 일부로 설정하는 것입니다.

AideEmail: 이 값은 cron 실행이 수행될 때마다 AIDE를 수신하는 이메일 주소를 설정합니다.

AideMuaPath: 이 값은 AIDE 보고서를 **AideEmail** 내에 설정된 이메일 주소로 보내는 데 사용되는 메일 사용자 에이전트의 경로를 설정합니다.

13.13.3. AIDE의 Cron 구성

AIDE director 서비스를 사용하면 cron 작업을 구성할 수 있습니다. 기본적으로 보고서를 `/var/log/audit/`로 보냅니다. 이메일 경고를 사용하려는 경우 **AideEmail** 매개변수를 활성화하여 구성된 이메일 주소로 경고를 보냅니다. 중요한 경고에 대한 이메일에 대한 의존은 시스템 중단 및 의도하지 않은 메시지 필터링에 취약할 수 있습니다.

13.13.4. 시스템 업그레이드의 영향 고려

업그레이드가 완료되면 AIDE 서비스가 새 무결성 데이터베이스를 자동으로 다시 생성하여 업그레이드된 모든 파일이 업데이트된 체크섬을 보유하도록 올바르게 다시 계산되도록 합니다.

openstack overcloud deploy를 초기 배포에 대한 후속 실행으로 호출하고 AIDE 구성 규칙이 변경되면 director AIDE 서비스에서 새 구성 속성이 무결성 데이터베이스에 캡슐화되도록 데이터베이스를 다시 빌드합니다.

13.14. SECURETTY 검토

securetty를 사용하면 모든 콘솔 장치(tty)에 대해 root 액세스를 비활성화할 수 있습니다. 이 동작은 `/etc/securetty` 파일의 항목에서 관리합니다. 예를 들면 다음과 같습니다.

```
resource_registry:
  OS::TripleO::Services::Securetty: ../puppet/services/securetty.yaml

parameter_defaults:
  TtyValues:
    - console
    - tty1
    - tty2
    - tty3
    - tty4
    - tty5
    - tty6
```

13.15. IDENTITY SERVICE에 대한 CRYOSTATF 감사

철저하게 감사 프로세스는 OpenStack 배포에 대한 지속적인 보안 상태를 검토하는 데 도움이 될 수 있습니다. 이는 보안 모델에서의 역할 때문에 keystone에 특히 중요합니다.

Red Hat OpenStack Platform은 감사 이벤트의 데이터 형식으로 Cloud Auditing Data Federation(CADF)을 채택했으며, keystone 서비스는 ID 및 토큰 작업에 대한 CryostatF 이벤트를 생성했습니다.

KeystoneNotificationFormat 을 사용하여 keystone에 대해 CryostatF 감사를 활성화할 수 있습니다.

```
parameter_defaults:
  KeystoneNotificationFormat: cadf
```

13.16. LOGIN.DEFS 값을 검토합니다.

새 시스템 사용자(non-keystone)에 대한 암호 요구 사항을 적용하기 위해 director는 다음 예제 매개변수에 따라 **/etc/login.defs** 에 항목을 추가할 수 있습니다.

```
resource_registry:
  OS::TripleO::Services::LoginDefs: ../puppet/services/login-defs.yaml

parameter_defaults:
  PasswordMaxDays: 60
  PasswordMinDays: 1
  PasswordMinLen: 5
  PasswordWarnAge: 7
  FailDelay: 4
```

14장. 대시보드 서비스 강화

대시보드 서비스(horizon)는 관리자가 설정한 제한 내에서 자체 리소스를 프로비저닝할 수 있는 셀프 서비스 포털을 제공합니다. OpenStack API와 동일한 민감도로 대시보드 서비스의 보안을 관리합니다.

14.1. 대시보드 서비스 디버깅

DEBUG 매개변수의 기본값은 **False** 입니다. 프로덕션 환경에 기본값을 유지합니다. 이 설정은 조사 중에만 변경합니다. **DEBUG** 매개변수의 값을 **True** 로 변경하면 Django는 중요한 웹 서버 상태 정보가 포함된 브라우저 사용자에게 스택 trace를 출력할 수 있습니다.

DEBUG 매개변수의 값이 **True** 이면 **ALLOWED_HOSTS** 설정도 비활성화됩니다. ALLOWED_HOSTS 구성에 대한 자세한 내용은 **ALLOWED_HOSTS 구성** 을 참조하십시오.

14.2. 도메인 이름 선택

모든 수준에서 공유 도메인과 달리 Dashboard 서비스(horizon)를 두 번째 수준 도메인에 배포하는 것이 좋습니다. 각각에 대한 예는 다음과 같습니다.

- 두 번째 수준 도메인: **https://example.com**
- 공유 하위 도메인: **https://example.public-url.com**

전용 두 번째 수준 도메인에 대시보드 서비스를 배포하면 브라우저의 **동일한 원본** 정책을 기반으로 다른 도메인에서 쿠키 및 보안 토큰이 격리됩니다. 하위 도메인에 배포하면 대시보드 서비스의 보안은 동일한 두 번째 수준 도메인에 배포된 가장 안전한 애플리케이션과 동일합니다.

쿠키 지원 세션 저장소를 피하고 HSTS(HTTP Strict Transport Security)를 구성하여 이러한 위험을 추가로 완화할 수 있습니다.



참고

https://example/ 과 같은 베어 도메인에 대시보드 서비스를 배포하는 것은 지원되지 않습니다.

14.3. ALLOWED_HOSTS 구성

Horizon은 python Django 웹 프레임워크를 기반으로 구축되었으며 잘못된 HTTP 호스트 헤더와 관련된 보안 위험을 보호해야 합니다. 이 보호를 적용하려면 OpenStack 대시보드에서 제공하는 FQDN을 사용하도록 **ALLOWED_HOSTS** 설정을 구성합니다.

ALLOWED_HOSTS 설정을 구성하면 이 목록의 값과 일치하지 않는 Host 헤더가 있는 HTTP 요청이 거부되고 오류가 발생합니다.

프로세스

1. 템플릿의 **parameter_defaults** 아래에 **HorizonAllowedHosts** 매개변수 값을 설정합니다.

```
parameter_defaults:
  HorizonAllowedHosts: <value>
```

& **lt;value** >를 OpenStack 대시보드에서 제공하는 FQDN으로 바꿉니다.

- 수정된 템플릿과 사용자 환경에 필요한 기타 모든 템플릿을 사용하여 오버클라우드를 배포합니다.

14.4. 교차 사이트 스크립팅 (XSS)

OpenStack 대시보드는 대부분의 필드에서 전체 유니코드 문자 세트를 허용합니다. 악의적인 행위자는 이 확장성을 사용하여 XSS(Cross-site scripting) 취약점을 테스트할 수 있습니다. OpenStack Dashboard 서비스(horizon)에는 XSS vulnerabilities에 대해 강화되는 도구가 있습니다. 사용자 지정 대시보드에서 이러한 툴을 올바르게 사용하는지 확인하는 것이 중요합니다. 사용자 정의 대시보드에 대한 감사를 수행할 때 다음 사항에 주의하십시오.

- **mark_safe** 함수입니다.
- **is_safe** - 사용자 지정 템플릿 태그와 함께 사용하는 경우
- **safe** 템플릿 태그입니다.
- 어디에서나 자동 이스케이프가 해제되고, JavaScript가 잘못 이스케이프된 데이터를 평가할 수 있습니다.

14.5. CSRF(CROSS SITE REQUEST FORGERY)

여러 JavaScript 인스턴스를 사용하는 대시보드는 **@csrf_exempt** 데코레이터 사용과 같은 취약점에 대해 감사해야 합니다. CORS(Cross Origin Resource Sharing) 제한을 낮추기 전에 권장 보안 설정을 따르지 않는 대시보드를 평가합니다. 각 응답과 함께 제한적인 CORS 헤더를 보내도록 웹 서버를 구성합니다. 대시보드 도메인 및 프로토콜만 허용합니다(예: **Access-Control-Allow-Origin: https://example.com/**). 와일드카드의 출처를 절대 허용하지 않아야 합니다.

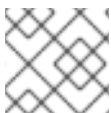
14.6. IFRAME 포함 허용

DISALLOW_IFRAME_EMBED 설정은 Dashboard가 iframe 내에 포함될 수 없습니다. 레거시 브라우저는 여전히 XFS(Cross-Frame Scripting) 취약점에 취약할 수 있으므로 이 옵션은 iframe이 필요하지 않은 배포에 보안 강화를 추가합니다. 설정은 기본적으로 **True** 로 설정되지만 필요한 경우 환경 파일을 사용하여 비활성화할 수 있습니다.

프로세스

- 다음 매개변수를 사용하여 iframe 포함을 허용할 수 있습니다.

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disallow_iframe_embed: false
```



참고

이러한 설정은 잠재적인 보안 영향을 완전히 이해한 후에만 **False** 로 설정되어야 합니다.

14.7. 대시보드 트래픽에 HTTPS 암호화 사용

HTTPS를 사용하여 대시보드 트래픽을 암호화하는 것이 좋습니다. 인식된 CA(인증 기관)에서 신뢰할 수 있는 유효한 인증서를 사용하도록 구성하여 이 작업을 수행할 수 있습니다. 개인 조직 발행 인증서는 신뢰의 루트가 모든 사용자 브라우저에 사전 설치된 경우에만 적합합니다.

정규화된 HTTPS URL로 리디렉션하도록 대시보드 도메인에 HTTP 요청을 구성합니다.

자세한 내용은 [7장. 오버클라우드 공용 끝점에서 SSL/TLS 활성화](#) 을 참조하십시오.

14.8. HSTS(HTTP STRICT TRANSPORT SECURITY)

HSTS(HTTP Strict Transport Security)는 처음에 보안 연결을 만든 후 이후 안전하지 않은 연결을 수행하지 못하도록 합니다. HTTP 서비스를 퍼블릭 또는 신뢰할 수 없는 영역에 배포한 경우 HSTS가 특히 중요합니다.

director 기반 배포의 경우 이 설정은 기본적으로 `/usr/share/openstack-tripleo-heat-templates/deployment/horizon/horizon-container-puppet.yaml` 파일에서 활성화됩니다.

```
horizon::enable_secure_proxy_ssl_header: true
```

검증

오버클라우드가 배포되면 Red Hat OpenStack Dashboard(horizon)의 `local_settings` 파일을 확인하여 확인합니다.

1. **ssh** 를 사용하여 컨트롤러에 연결합니다.

```
$ ssh tripleo-admin@controller-0
```

2. **SECURE_PROXY_SSL_HEADER** 매개변수의 값이 ('HTTP_X_FORWARDED_PROTO')인 'https' 인지 확인합니다.

```
sudo egrep ^SECURE_PROXY_SSL_HEADER /var/lib/config-data/puppet-generated/horizon/etc/openstack-dashboard/local_settings
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
```

14.9. 프런트 엔드 캐싱

OpenStack API 요청에서 직접 생성되는 동적 콘텐츠를 렌더링하므로 대시보드에 프런트 엔드 캐싱 툴을 사용하지 않는 것이 좋습니다. 결과적으로 **varnish** 와 같은 프런트 엔드 캐싱 계층을 사용하면 올바른 콘텐츠가 표시되지 않을 수 있습니다. 대시보드는 웹 서비스에서 직접 제공되는 정적 미디어를 제공하고 이미 웹 호스트 캐싱의 이점을 제공하는 Django를 사용합니다.

14.10. 세션 백엔드

director 기반 배포의 경우 horizon의 기본 세션 백엔드는 memcached와 결합된 **django.contrib.sessions.cache** 입니다. 이 접근 방식은 성능상의 이유로 로컬 메모리 캐시를 선호하며 고가용성 및 부하 분산 설치에서 더 안전하며 여러 서버를 통해 캐시를 공유할 수 있는 기능을 제공하는 동시에 단일 캐시로 처리합니다.

director의 **horizon.yaml** 파일에서 이러한 설정을 검토할 수 있습니다.

```
horizon::cache_backend: django.core.cache.backends.memcached.MemcachedCache
horizon::django_session_engine: 'django.contrib.sessions.backends.cache'
```

14.11. 시크릿 키 검토

대시보드는 일부 보안 기능의 공유 **SECRET_KEY** 설정에 따라 다릅니다. 시크릿 키는 임의로 생성된 문자열 64자 이상이어야 하며, 이 문자열은 모든 활성 대시보드 인스턴스에서 공유해야 합니다. 이 키를 손상시키면 원격 공격자가 임의의 코드를 실행할 수 있습니다. 이 키를 교체하면 기존 사용자 세션 및 캐싱이 무효화됩니다. 이 키를 공개 리포지토리에 커밋하지 마십시오.

director 배포의 경우 이 설정은 **HorizonSecret** 값으로 관리됩니다.

14.12. 세션 쿠키 구성

대시보드 세션 쿠키는 JavaScript와 같은 브라우저 기술의 상호 작용에 열려 있을 수 있습니다. 모든 곳에서 TLS를 사용한 director 배포의 경우 **HorizonSecureCookies** 설정을 사용하여 이 동작을 강화할 수 있습니다.



참고

선행 점이 있는 와일드카드 도메인을 사용하도록 CSRF 또는 세션 쿠키를 설정하지 마십시오.

14.13. 암호 복잡성 검증

OpenStack 대시보드(horizon)는 암호 유효성 검사 검사를 사용하여 암호 복잡성을 적용할 수 있습니다.

프로세스

1. 암호 검증을 위해 정규식과 실패한 테스트에 표시할 도움말 텍스트를 지정합니다. 다음 예제에서는 사용자가 8~18자 사이의 암호를 생성해야 합니다.

parameter_defaults:

HorizonPasswordValidator: '^.{8,18}\$'

HorizonPasswordValidatorHelp: 'Password must be between 8 and 18 characters.'

1. 이 변경 사항을 배포에 적용합니다. 설정을 **horizon_password.yaml** 이라는 파일로 저장한 다음 다음과 같이 **overcloud deploy** 명령에 전달합니다. <full environment> 는 원래 배포 매개변수를 모두 포함해야 함을 나타냅니다. 예를 들면 다음과 같습니다.

```
openstack overcloud deploy --templates \
-e <full environment> -e horizon_password.yaml
```

14.14. 관리자 암호 확인 적용

다음 설정은 기본적으로 **True** 로 설정되지만 필요한 경우 환경 파일을 사용하여 비활성화할 수 있습니다.



참고

이러한 설정은 잠재적인 보안 영향을 완전히 이해한 후에만 **False** 로 설정되어야 합니다.

프로세스

대시보드의 **local_settings.py** 파일의 **ENFORCE_PASSWORD_CHECK** 설정에 암호 변경 양식의 **Admin Password** 필드가 표시되어 관리자가 암호 변경을 시작하고 있는지 확인하는 데 도움이 됩니다.

- 환경 파일을 사용하여 **ENFORCE_PASSWORD_CHECK** 를 비활성화할 수 있습니다.

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::enforce_password_check: false
```

14.15. 암호 표시 비활성화

disable_password_reveal 매개변수는 기본적으로 **True** 로 설정되지만, 필요한 경우 환경 파일을 사용하여 비활성화할 수 있습니다. 암호 공개 버튼을 사용하면 대시보드의 사용자가 입력할 암호를 볼 수 있습니다.

프로세스

- **ControllerExtraConfig** 매개변수 아래에 **horizon::disable_password_reveal: false** 를 포함합니다. 이 값을 heat 환경 파일에 저장하고 배포 명령을 사용하여 포함합니다.

예제

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disable_password_reveal: false
```



참고

이러한 설정은 잠재적인 보안 영향을 완전히 이해한 후에만 **False** 로 설정되어야 합니다.

14.16. 대시보드의 로그인 배너 표시

HIPAA, PCI-DSS 및 US Government와 같은 규제된 업계에서는 사용자 로그인 배너를 표시해야 합니다. RHOSP(Red Hat OpenStack Platform) 대시보드(horizon)는 Horizon 컨테이너 내부에 저장된 기본 주제(RCUE)를 사용합니다.

사용자 정의 대시보드 컨테이너 내에서 **/usr/share/openstack-dashboard/openstack_dashboard/themes/rcue/templates/auth/login.html** 파일을 수동으로 편집하여 로그인 배너를 생성할 수 있습니다.

프로세스

1. {% 가 'auth/_login.html' %} 섹션 바로 앞에 필요한 로그인 배너를 입력합니다. HTML 태그를 사용할 수 있습니다.

```
<snip>
<div class="container">
  <div class="row-fluid">
    <div class="span12">
      <div id="brand">
        
      </div><!--#brand-->
    </div><!--.span*-->

    <!-- Start of Logon Banner -->
    <p>Authentication to this information system reflects acceptance of user monitoring
    agreement.</p>
    <!-- End of Logon Banner -->
```

```

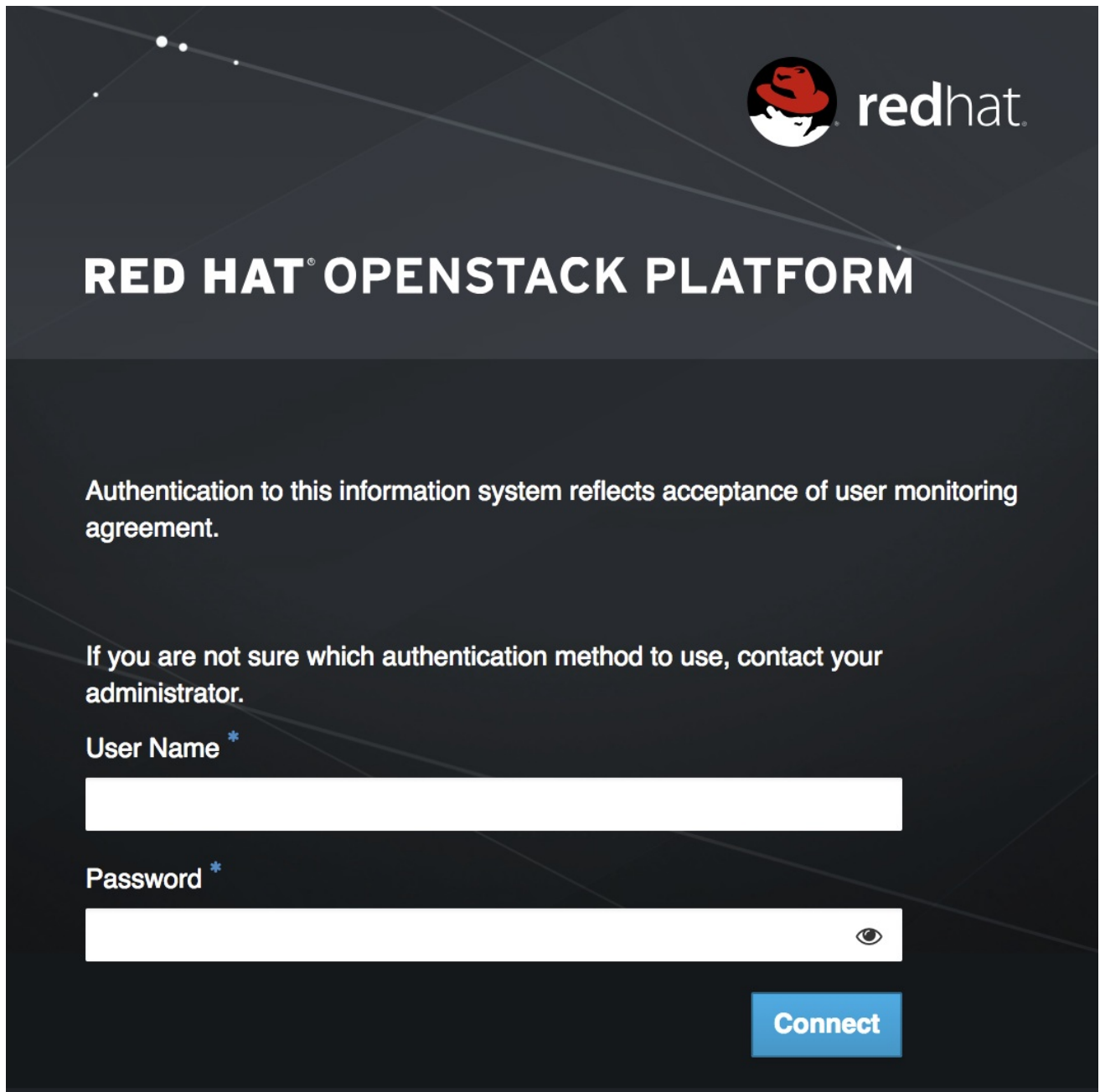
    {% include 'auth/_login.html' %}
  </div><!--/.row-fluid →
</div><!--/.container-->

{% block js %}
  {% include "horizon/_scripts.html" %}
{% endblock %}

</body>
</html>

```

위 예제에서는 다음과 유사한 대시보드를 생성합니다.



The image shows a login dashboard for Red Hat OpenStack Platform. At the top right is the Red Hat logo. The main heading is "RED HAT® OPENSTACK PLATFORM". Below this is a message: "Authentication to this information system reflects acceptance of user monitoring agreement." followed by "If you are not sure which authentication method to use, contact your administrator." The login form includes a "User Name" field with an asterisk, a "Password" field with an asterisk and an eye icon for toggling visibility, and a blue "Connect" button at the bottom right.

추가 리소스

- [대시보드 사용자 정의](#)

14.17. 파일 업로드 크기 제한

선택적으로 파일 업로드 크기를 제한하도록 대시보드를 구성할 수 있습니다. 이 설정은 다양한 보안 강화 정책의 요구 사항이 될 수 있습니다.

LimitRequestBody - 이 값(바이트)은 이미지 및 기타 대용량 파일과 같이 대시보드를 사용하여 전송할 수 있는 파일의 최대 크기를 제한합니다.



중요

이 설정은 Red Hat에서 공식적으로 테스트하지 않았습니다. 프로덕션 환경에 배포하기 전에 이 설정의 효과를 철저히 테스트하는 것이 좋습니다.



참고

값이 너무 작으면 파일 업로드가 실패합니다.

예를 들어 이 설정은 각 파일 업로드를 최대 10GB(**10737418240**)로 제한합니다. 배포에 맞게 이 값을 조정해야 합니다.

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf/httpd.conf**

```
<Directory />
  LimitRequestBody 10737418240
</Directory>
```

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/10-horizon_vhost.conf**

```
<Directory "/var/www">
  LimitRequestBody 10737418240
</Directory>
```

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/15-horizon_ssl_vhost.conf**

```
<Directory "/var/www">
  LimitRequestBody 10737418240
</Directory>
```



참고

이러한 구성 파일은 Puppet에서 관리하므로 **openstack overcloud deploy** 프로세스를 실행할 때마다 관리되지 않는 변경 사항을 덮어씁니다.

15장. 네트워킹 서비스 강화

Networking 서비스(neutron)는 RHOSP(Red Hat OpenStack Platform)의 소프트웨어 정의 네트워킹(SDN) 구성 요소입니다. RHOSP 네트워킹 서비스는 가상 머신 인스턴스로의 내부 및 외부 트래픽을 관리하고 라우팅, 분할, DHCP 및 메타데이터와 같은 핵심 서비스를 제공합니다. 가상 네트워킹 기능 및 스위치, 라우터, 포트 및 방화벽 관리를 위한 API를 제공합니다.

Red Hat OpenStack Platform Networking 서비스에 대한 자세한 내용은 [Red Hat OpenStack Platform 네트워킹 구성](#)을 참조하십시오.

이 섹션에서는 OpenStack Networking 구성 모범 사례에 OpenStack 배포 내의 프로젝트 네트워크 보안에 적용되는 방법에 대해 설명합니다.

15.1. 프로젝트 네트워크 서비스 워크플로

OpenStack Networking에서는 네트워크 리소스의 셀프 서비스 구성을 사용자에게 제공합니다. 클라우드 아키텍트와 운영자가 사용 가능한 네트워크 리소스를 생성, 업데이트 및 제거할 수 있는 기능을 사용자에게 제공하는 설계 사용 사례를 평가하는 것이 중요합니다.

15.2. 네트워킹 리소스 정책 엔진

OpenStack Networking 내의 정책 엔진 및 해당 구성 파일(**policy.json**)은 프로젝트 네트워킹 방법 및 오브젝트에 대해 사용자를 세분화하여 권한을 부여하는 방법을 제공합니다. OpenStack Networking 정책 정의는 네트워크 가용성, 네트워크 보안 및 전체 OpenStack 보안에 영향을 미칩니다. 클라우드 아키텍트와 운영자는 네트워크 리소스 관리에 대한 사용자 및 프로젝트 액세스에 대한 정책을 신중하게 평가해야 합니다.



참고

보안 상태에 맞게 이 정책을 수정할 수 있으므로 기본 네트워킹 리소스 정책을 검토하는 것이 중요합니다.

OpenStack을 배포하면 다양한 보안 영역에 여러 외부 액세스 지점을 제공하는 것이 중요합니다. 즉, 여러 개의 vNIC를 여러 개의 외부 액세스 포인트에 연결할 수 있는 프로젝트의 기능을 제한하는 것이 중요합니다. 즉, 이러한 보안 영역을 브리지할 수 있으므로 보안 손상이 발생할 수 있습니다. Compute에서 제공하는 호스트 집계 기능을 사용하거나 다른 가상 네트워킹 구성을 사용하여 프로젝트 인스턴스를 여러 프로젝트로 분할하여 이 위험을 완화할 수 있습니다. 호스트 집계에 대한 자세한 내용은 호스트 집계 [생성 및 관리](#)를 참조하십시오.

15.3. 보안 그룹

보안 그룹은 보안 그룹 규칙 컬렉션입니다. 보안 그룹 및 해당 규칙을 통해 관리자 및 프로젝트에서 가상 인터페이스 포트를 통과할 수 있는 트래픽 및 방향(ingress/egress) 유형을 지정할 수 있습니다. OpenStack Networking에서 가상 인터페이스 포트가 생성되면 보안 그룹과 연결됩니다. 배포별로 동작을 변경하기 위해 기본 보안 그룹에 규칙을 추가할 수 있습니다.

Compute API를 사용하여 보안 그룹을 수정하는 경우 업데이트된 보안 그룹이 인스턴스의 모든 가상 인터페이스 포트에 적용됩니다. 이는 neutron에서 찾은 것처럼 Compute 보안 그룹 API가 포트 기반이 아닌 인스턴스 기반 API이기 때문입니다.

15.4. ARP 스푸핑 완화

OpenStack Networking에는 인스턴스의 ARP 스푸핑 위협을 완화하는 데 도움이 되는 기본 기능이 있습니다. 이는 결과적인 위협에 주의를 기울이지 않는 한 비활성화해서는 안 됩니다.

15.5. 인증을 위한 보안 프로토콜 사용

`/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf` 에서 **[keystone_authtoken]** 섹션 아래의 **auth_uri** 값이 'https'로 시작하는 ID API 끝점으로 설정되어 있는지 확인합니다.

16장. RED HAT OPENSTACK PLATFORM에서 블록 스토리지 강화

OpenStack Block Storage(cinder)는 영구 블록 수준 스토리지 장치를 셀프 서비스에 소프트웨어(서비스 및 라이브러리)를 제공하는 서비스입니다. 이렇게 하면 Compute(nova) 인스턴스에 사용할 블록 스토리지 리소스에 대한 온디맨드 액세스가 생성됩니다. 이렇게 하면 소프트웨어 구현 또는 기존 하드웨어 스토리지 제품일 수 있는 다양한 백엔드 스토리지 장치로 블록 스토리지 풀을 가상화하여 소프트웨어 정의 스토리지가 생성됩니다. 이 문제의 주요 기능은 블록 장치의 생성, 연결 및 분리를 관리하는 것입니다. 소비자는 백엔드 스토리지 장비 유형 또는 위치에 대한 지식이 필요하지 않습니다.

컴퓨팅 인스턴스는 iSCSI, ATA over Ethernet 또는 Fibre-Channel과 같은 업계 표준 스토리지 프로토콜을 사용하여 블록 스토리지를 저장하고 검색합니다. 이러한 리소스는 OpenStack 기본 HTTP RESTful API를 사용하여 관리 및 구성됩니다.

16.1. 요청 본문의 최대 크기 설정

요청당 최대 본문 크기가 정의되지 않은 경우 공격자는 대규모의 임의의 OSAPI 요청을 생성하여 서비스가 충돌하고 마지막으로 서비스 거부 공격을 유발할 수 있습니다. t 값을 할당하면 악의적인 초과 요청이 차단되어 서비스의 지속적인 가용성을 보장합니다.

cinder.conf의 **[oslo_middleware]** 섹션 아래의 **max_request_body_size**가 **114688**로 설정되어 있는지 검토합니다.

16.2. 볼륨 암호화 활성화

암호화되지 않은 볼륨 데이터는 공격자가 여러 다른 VM에 대한 데이터를 읽을 수 있으므로 볼륨 호스팅 플랫폼을 특히 높은 값의 대상으로 만듭니다. 또한 물리적 저장 매체를 도난, 다시 마운트 및 다른 시스템에서 액세스할 수 있습니다. 볼륨 데이터 및 볼륨 백업을 암호화하면 이러한 위험을 완화하는 데 도움이 되며 볼륨 호스팅 플랫폼에 대한 심층 방어 기능을 제공할 수 있습니다. Block Storage(cinder)는 디스크에 쓰기 전에 볼륨 데이터를 암호화할 수 있으므로 볼륨 암호화를 활성화하고 개인 키 스토리지에 Barbican을 사용하는 것이 좋습니다.

16.3. 볼륨 도핑

블록 스토리지 장치를 지우는 방법은 여러 가지가 있습니다. 기존 접근 방식은 **lvm_type**을 **thin**으로 설정한 다음 **volume_clear** 매개 변수를 사용하는 것입니다. 또는 볼륨 암호화 기능이 사용되는 경우 볼륨 암호화 키가 삭제되면 볼륨 wiping이 필요하지 않습니다.



참고

이전에는 **lvm_type=default**가 wipe를 나타내는 데 사용되었습니다. 이 방법은 계속 작동하지만 secure delete를 설정하는 데 **lvm_type=default**를 사용하지 않는 것이 좋습니다.

volume_clear 매개 변수는 **0** 또는 **shred**를 인수로 사용할 수 있습니다. **0**은 장치에 대한 0의 단일 패스를 작성합니다. **shred** 작업은 사전 정의된 비트 패턴의 세 가지 패스를 작성합니다.

17장. 공유 파일 시스템 강화 (MANILA)

Shared File Systems 서비스(manila)는 다중 프로젝트 클라우드 환경에서 공유 파일 시스템을 관리하기 위한 일련의 서비스를 제공합니다. manila를 사용하면 공유 파일 시스템을 생성하고 가시성, 접근성, 할당량과 같은 해당 속성을 관리할 수 있습니다.

Shared File Systems 서비스(manila)에 대한 자세한 내용은 [영구 스토리지 가이드 구성](#) 을 참조하십시오.

17.1. MANILA의 보안 고려 사항

Manila는 keystone에 등록되므로 **manila endpoint** 명령을 사용하여 **API**를 찾을 수 있습니다. 예를 들면 다음과 같습니다.

```
$ manila endpoints
+-----+-----+
| manila | Value |
+-----+-----+
| adminURL | http://172.18.198.55:8786/v1/20787a7b...|
| region   | RegionOne |
| publicURL | http://172.18.198.55:8786/v1/20787a7b...|
| internalURL | http://172.18.198.55:8786/v1/20787a7b...|
| id       | 82cc5535aa444632b64585f138cb9b61 |
+-----+-----+

+-----+-----+
| manilav2 | Value |
+-----+-----+
| adminURL | http://172.18.198.55:8786/v2/20787a7b...|
| region   | RegionOne |
| publicURL | http://172.18.198.55:8786/v2/20787a7b...|
| internalURL | http://172.18.198.55:8786/v2/20787a7b...|
| id       | 2e8591bfcac4405fa7e5dc3fd61a2b85 |
+-----+-----+
```

기본적으로 manila API 서비스는 IPv4 및 IPv6를 모두 지원하는 **tcp6** 이 있는 포트 **8786** 에서만 수신 대기합니다.

Manila는 여러 구성 파일을 사용합니다. 이러한 파일은 **/var/lib/config-data/puppet-generated/manila/**에 저장됩니다.

```
api-paste.ini
manila.conf
policy.json
rootwrap.conf
rootwrap.d

./rootwrap.d:
share.filters
```

시스템 관리자만 수정할 수 있도록 루트가 아닌 서비스 계정에서 실행하도록 manila를 구성하고 파일 권한을 변경하는 것이 좋습니다. Manila는 관리자만 구성 파일에 쓸 수 있을 것으로 예상하고, 서비스는 **manila** 그룹의 그룹 멤버십을 통해서만 읽을 수 있습니다. 다른 사용자는 서비스 계정 암호가 포함되어 있으므로 이러한 파일을 읽을 수 없어야 합니다.



참고

root 사용자만 **rootwrap.conf** 에서 **manila-rootwrap** 의 구성에 쓸 수 있어야 하며 **rootwrap.d/share.filters** 의 공유 노드에 대한 **manila-rootwrap** 명령 필터를 작성할 수 있어야 합니다.

17.2. MANILA의 네트워크 및 보안 모델

manila의 공유 드라이버는 백엔드에서 공유 작업을 관리하도록 설정할 수 있는 Python 클래스이며, 일부는 벤더에 따라 다릅니다. 백엔드는 **manila-share** 서비스의 인스턴스입니다. Manila는 다양한 스토리지 시스템을 위한 드라이버를 공유하여 상용 벤더와 오픈 소스 솔루션을 모두 지원합니다. 각 공유 드라이버는 하나 이상의 백엔드 모드(**공유 서버 및 공유 서버 없음**)를 지원합니다. 관리자가 **driver_handles_share_servers** 를 사용하여 **manila.conf** 에 지정하여 모드를 선택합니다.

공유 서버는 공유 파일 시스템을 내보내는 논리 네트워크 연결 스토리지(NAS) 서버입니다. 현재 백엔드 스토리지 시스템은 정교하며 서로 다른 OpenStack 프로젝트 간에 데이터 경로와 네트워크 경로를 분리할 수 있습니다.

manila 공유 드라이버에서 프로비저닝한 공유 서버는 프로젝트 사용자가 생성하는 격리된 네트워크에서 생성됩니다. **공유 서버** 모드는 네트워크 공급자에 따라 플랫 네트워크 또는 분할된 네트워크를 사용하여 구성할 수 있습니다.

서로 다른 모드에 대해 별도의 드라이버가 동일한 하드웨어를 사용할 수 있습니다. 선택한 모드에 따라 구성 파일을 통해 더 많은 구성 세부 정보를 제공해야 할 수 있습니다.

17.3. 백엔드 모드 공유

각 공유 드라이버는 사용 가능한 드라이버 모드 중 하나 이상을 지원합니다.

- 공유 서버 - **driver_handles_share_servers = True** - 공유 드라이버는 공유 서버를 생성하고 공유 서버 라이프사이클을 관리합니다.
- 공유 서버 없음 - **driver_handles_share_servers = False** - 공유 드라이버가 아닌 관리자(공유 드라이버 대신)는 공유 서버가 있는 대신 네트워크 인터페이스로 베어 메탈 스토리지를 관리합니다.

공유 서버 모드 없음 - 이 모드에서는 드라이버가 공유 서버를 설정하지 않으므로 새 네트워크 인터페이스를 설정할 필요가 없습니다. 드라이버에서 관리하는 스토리지 컨트롤러에는 필요한 모든 네트워크 인터페이스가 있다고 가정합니다. 드라이버는 이전에 공유 서버를 생성하지 않고 직접 공유를 생성합니다. 이 모드에서 작동하는 드라이버를 사용하여 공유를 생성하려면 사용자가 개인 공유 네트워크를 생성할 필요가 없습니다.



참고

공유 서버 모드가 없는 manila는 모든 프로젝트에서 공유한 공유 영역에 이미 연결할 수 있는 네트워크 인터페이스가 있다고 가정합니다.

공유 서버 모드 없이 공유 드라이버는 공유 서버 라이프사이클을 처리하지 않습니다. 관리자는 프로젝트 격리를 제공하는 데 필요할 수 있는 스토리지, 네트워킹 및 기타 호스트 측 구성을 처리해야 합니다. 이 모드에서 관리자는 공유를 내보내는 호스트로 스토리지를 설정할 수 있습니다. OpenStack 클라우드 내의 모든 프로젝트는 공통 네트워크 파이프를 공유합니다. 격리 부족으로 인해 보안 및 서비스 품질에 영향을 미칠 수 있습니다. 공유 서버를 처리하지 않는 공유 드라이버를 사용하는 경우 트리에서 파일 시스템의 최상위 디렉토리를 통과하여 신뢰할 수 없는 사용자가 해당 공유에 액세스할 수 있는지 확인할 수 없습니다.

퍼블릭 클라우드에서는 한 클라이언트에서 모든 네트워크 대역폭을 사용할 수 있으므로 관리자가 이 문제가 발생하지 않도록 주의해야 합니다. 네트워크 밸런싱은 OpenStack 툴뿐만 아니라 모든 방법으로 수행할 수 있습니다.

공유 서버 모드 - 이 모드에서 드라이버는 공유 서버를 생성하여 기존 OpenStack 네트워크에 연결할 수 있습니다. Manila는 새 공유 서버가 필요한지 여부를 확인하고 공유 드라이버에서 필수 공유 서버를 생성하는 데 필요한 모든 네트워킹 정보를 제공합니다.

공유 서버를 처리하는 드라이버 모드에서 공유를 생성할 때 사용자는 공유를 내보낼 수 있는 공유 네트워크를 제공해야 합니다. Manila는 이 네트워크를 사용하여 이 네트워크에서 공유 서버에 대한 네트워크 포트를 생성합니다.

사용자는 **공유 서버 및 공유 서버 백엔드 모드 없이 보안 서비스**를 구성할 수 있습니다. 그러나 **공유 서버 백엔드 모드**가 없는 상태에서 관리자는 필요한 인증 서비스를 호스트에서 수동으로 설정해야 합니다. 공유 서버 모드에서 manila는 생성된 공유 서버에서 사용자가 식별한 보안 서비스를 구성할 수 있습니다.

17.4. MANILA의 네트워킹 요구 사항

Manila는 **flat,GRE,VLAN,VXLAN** 등 다양한 네트워크 유형과 통합할 수 있습니다.



참고

Manila는 네트워크 공급자가 제공하는 실제 네트워크가 있는 데이터베이스에만 네트워크 정보를 저장합니다. Manila는 OpenStack Networking 서비스(neutron) 및 "독립 실행형" 사전 구성된 네트워킹 사용을 지원합니다.

공유 서버 백엔드 모드에서 공유 드라이버는 각 공유 네트워크에 대한 공유 서버를 생성하고 관리합니다. 이 모드는 두 가지 변형으로 나눌 수 있습니다.

- 공유 서버 백엔드 모드의 플랫폼 네트워크
- 공유 서버 백엔드 모드의 분할된 네트워크

사용자는 OpenStack Networking(neutron) 서비스의 네트워크 및 서브넷을 사용하여 공유 네트워크를 생성할 수 있습니다. 관리자가 **StandAloneNetworkPlugin**을 사용하기로 결정한 경우 관리자는 구성 파일에서 이를 사전 구성하므로 사용자가 네트워킹 정보를 제공할 필요가 없습니다.



참고

일부 공유 드라이버에서 생성한 공유 서버는 Compute 서비스를 사용하여 생성된 Compute 서버입니다. 이러한 드라이버 중 일부는 네트워크 플러그인을 지원하지 않습니다.

공유 네트워크가 생성되면 manila는 네트워크 공급자, 네트워크 유형, 세그먼트 식별자(네트워크가 세그먼트를 사용하는 경우) 및 네트워크를 할당할 CIDR 표기법의 IP 블록을 검색합니다.

사용자는 AD 또는 LDAP 도메인 또는 Kerberos 영역과 같은 보안 요구 사항을 지정하는 보안 서비스를 생성할 수 있습니다. Manila는 보안 서비스에서 참조하는 모든 호스트에 공유 서버가 생성되는 서브넷에서 연결할 수 있다고 가정하여 이 모드를 사용할 수 있는 사례 수를 제한합니다.



참고

일부 공유 드라이버는 모든 유형의 분할을 지원하지 않을 수 있습니다. 자세한 내용은 사용 중인 드라이버의 사양을 참조하십시오.

17.5. MANILA를 사용하는 보안 서비스

Manila는 네트워크 인증 프로토콜과 통합하여 파일 공유에 대한 액세스를 제한할 수 있습니다. 각 프로젝트에는 클라우드의 keystone 인증 도메인과 별도로 작동하는 고유한 인증 도메인이 있을 수 있습니다. 이 프로젝트 도메인은 manila를 포함하여 OpenStack 클라우드 내에서 실행되는 애플리케이션에 권한 부여 (AuthZ) 서비스를 제공하는 데 사용할 수 있습니다. 사용 가능한 인증 프로토콜에는 LDAP, Kerberos 및 Microsoft Active Directory 인증 서비스가 포함됩니다.

17.6. 보안 서비스 소개

공유를 생성하고 내보내기 위치를 가져온 후 사용자는 마운트하고 파일과 함께 작동할 수 있는 권한이 없습니다. 사용자는 새 공유에 대한 액세스 권한을 명시적으로 부여해야 합니다.

클라이언트 인증 및 권한 부여(authN/authZ)는 보안 서비스와 함께 수행할 수 있습니다. 공유 드라이버 및 백엔드에서 지원하는 경우 Manila는 LDAP, Kerberos 또는 Microsoft Active 디렉터리를 사용할 수 있습니다.



참고

경우에 따라 보안 서비스 중 하나를 명시적으로 지정해야 합니다(예: NetApp, EMC 및 Windows 드라이버)에는 CIFS 프로토콜과의 공유를 생성하기 위해 Active Directory가 필요합니다.

17.7. 보안 서비스 관리

보안 서비스는 Active Directory 도메인 또는 Kerberos 도메인과 같은 특정 공유 파일 시스템 프로토콜의 보안 영역을 정의하는 일련의 옵션을 추상화하는 manila 엔티티입니다. 보안 서비스에는 manila가 지정된 도메인에 조인하는 서버를 생성하는 데 필요한 모든 정보가 포함되어 있습니다.

사용자는 API를 사용하여 보안 서비스를 생성, 업데이트, 보기, 삭제할 수 있습니다. Security Services는 다음과 같은 가정에 따라 설계되었습니다.

- 프로젝트는 보안 서비스에 대한 세부 정보를 제공합니다.
- 관리자는 보안 서비스에 대한 관심: 이러한 보안 서비스의 서버 측면을 구성합니다.
- manila API 내에서 **security_service** 는 **share_networks** 와 연결됩니다.
- 공유 드라이버는 보안 서비스의 데이터를 사용하여 새로 생성된 공유 서버를 구성합니다.

보안 서비스를 생성할 때 다음 인증 서비스 중 하나를 선택할 수 있습니다.

- **LDAP** - Lightweight Directory Access Protocol IP 네트워크를 통해 분산 디렉터리 정보 서비스에 액세스하고 유지 관리하기 위한 애플리케이션 프로토콜입니다.
- **Kerberos** - 비보안 네트워크를 통해 통신하는 노드가 안전한 방식으로 서로 신원을 증명할 수 있도록 티켓에 따라 작동하는 네트워크 인증 프로토콜입니다.
- **Active Directory** - Microsoft가 Windows 도메인 네트워크용으로 개발한 디렉터리 서비스입니다. LDAP, Microsoft 버전의 Kerberos 및 DNS를 사용합니다.

Manila를 사용하면 다음 옵션을 사용하여 보안 서비스를 구성할 수 있습니다.

- 프로젝트 네트워크 내에서 사용되는 DNS IP 주소입니다.

- 보안 서비스의 IP 주소 또는 호스트 이름입니다.
- 보안 서비스의 도메인입니다.
- 프로젝트에서 사용하는 사용자 또는 그룹 이름입니다.
- 사용자 이름을 지정하는 경우 사용자의 암호입니다.

기존 보안 서비스 엔티티는 manila에 공유 그룹의 보안 및 네트워크 구성에 대해 알려주는 공유 네트워크 엔티티와 연결할 수 있습니다. 지정된 공유 네트워크의 모든 보안 서비스 목록을 확인하고 공유 네트워크에서 연결을 해제할 수도 있습니다.

공유 소유자인 관리자와 사용자는 IP 주소, 사용자, 그룹 또는 TLS 인증서를 통해 인증을 사용하여 액세스 규칙을 생성하여 공유에 대한 액세스를 관리할 수 있습니다. 인증 방법은 구성 및 사용하는 공유 드라이버 및 보안 서비스에 따라 다릅니다. 그런 다음 manila 및 keystone 없이 클라이언트와 작동할 수 있는 특정 인증 서비스를 사용하도록 백엔드를 구성할 수 있습니다.



참고

다양한 공유 드라이버에서 다양한 인증 서비스를 지원합니다. 다른 드라이버의 기능 지원에 대한 자세한 내용은

https://docs.openstack.org/manila/latest/admin/share_back_ends_feature_support_mappings.html 참조하십시오.

드라이버의 특정 인증 서비스에 대한 지원은 공유 파일 시스템 프로토콜로 구성할 수 있음을 의미하지는 않습니다. 지원되는 공유 파일 시스템 프로토콜은 NFS, CEPHFS, CIFS, GlusterFS 및 HDFS입니다. 특정 드라이버 및 보안 서비스에 대한 구성에 대한 정보는 드라이버 공급 업체의 설명서를 참조하십시오.

일부 드라이버는 보안 서비스를 지원하며 다른 드라이버는 위에서 언급한 보안 서비스를 지원하지 않습니다. 예를 들어 NFS를 사용하는 일반 드라이버 또는 CIFS 공유 파일 시스템 프로토콜은 IP 주소를 통한 인증 방법만 지원합니다.



참고

대부분의 경우 CIFS 공유 파일 시스템 프로토콜을 지원하는 드라이버는 Active Directory를 사용하고 사용자 인증을 통해 액세스를 관리하도록 구성할 수 있습니다.

- GlusterFS 프로토콜을 지원하는 드라이버는 TLS 인증서를 사용하여 인증과 함께 사용할 수 있습니다.
- IP 주소를 사용하여 NFS 프로토콜 인증을 지원하는 드라이버는 지원되는 유일한 옵션입니다.
- HDFS 공유 파일 시스템 프로토콜은 NFS 액세스를 사용하므로 IP 주소로 인증하도록 구성할 수도 있습니다.

프로덕션 manila 배포에 권장되는 구성은 CIFS 공유 프로토콜로 공유를 생성하고 이를 Microsoft Active Directory 디렉터리 서비스에 추가하는 것입니다. 이 구성을 사용하면 Kerberos 및 LDAP 접근 방식을 통합하는 중앙 집중식 데이터베이스 및 서비스를 얻을 수 있습니다.

17.8. 액세스 제어 공유

사용자는 생성하는 공유에 액세스할 수 있는 특정 클라이언트를 지정할 수 있습니다. keystone 서비스로 인해 개별 사용자가 생성한 공유는 동일한 프로젝트 내의 자체 및 기타 사용자에게만 표시됩니다. Manila를 사용하면 "공개"가 표시되는 공유를 생성할 수 있습니다. 이러한 공유는 소유자가 액세스 권한을 부여

하는 경우 다른 OpenStack 프로젝트에 속하는 사용자의 대시보드에 표시됩니다. 네트워크에서 액세스할 수 있게 되면 이러한 공유를 마운트할 수도 있습니다.

공유를 생성하는 동안 key **--public** 을 사용하여 다른 프로젝트에서 공유 공개를 공유 목록에서 확인하고 자세한 정보를 확인합니다.

policy.json 파일에 따라 공유 소유자인 관리자와 사용자는 액세스 규칙을 생성하여 공유에 대한 액세스 권한을 관리할 수 있습니다. **manila access-allow**, **manila access-deny**, **manila access-list** 명령을 사용하면 지정된 공유에 대한 액세스 권한을 적절하게 부여, 거부 및 나열할 수 있습니다.



참고

Manila는 스토리지 시스템의 엔드 투 엔드 관리를 제공하지 않습니다. 백엔드 시스템을 무단 액세스로부터 별도로 보호해야 합니다. 결과적으로 누군가가 백엔드 스토리지 장치를 손상시키는 경우에도 manila API에서 제공하는 보호 기능을 우회하여 대역 액세스를 확보할 수 있습니다.

공유가 생성되면 해당 공유와 연결된 기본 액세스 규칙과 마운트 권한이 없습니다. 이는 사용 중인 내보내기 프로토콜의 마운트 구성에서 확인할 수 있습니다. 예를 들어 각 원격 공유를 제어하고 액세스할 수 있는 호스트를 정의하는 스토리지에 NFS 명령 **exportfs** 또는 **/etc/exports** 파일이 있습니다. 누가 공유를 마운트할 수 없는 경우 비어 있습니다. 원격 CIFS 서버의 경우 설정을 보여주는 **net conf list** 명령이 있습니다. **hosts** 거부 매개 변수는 공유 드라이버에서 **0.0.0.0/0** 으로 설정해야 합니다. 즉, 모든 호스트가 공유 마운트를 거부함을 의미합니다.

manila를 사용하면 지원되는 공유 액세스 수준 중 하나를 지정하여 공유에 대한 액세스 권한을 부여하거나 거부할 수 있습니다.

- **RW** - 읽기 및 쓰기(RW) 액세스. 이는 기본값입니다.
- **ro**- 읽기 전용(RO) 액세스.



참고

RO 액세스 수준은 관리자가 일부 특정 편집기 또는 기여자에 대해 읽기 및 쓰기(RW) 액세스를 제공하고 나머지 사용자(viewer)에 대해 읽기 전용(RO) 액세스 권한을 부여하는 경우 공개 공유에 유용할 수 있습니다.

다음과 같은 지원되는 인증 방법 중 하나를 지정해야 합니다.

- **IP** - IP 주소를 사용하여 인스턴스를 인증합니다. CIDR 표기법으로 잘 구성된 IPv4 또는 IPv6 주소 또는 서브넷으로 주소를 지정할 수 있는 클라이언트에 IP 액세스를 제공할 수 있습니다.
- **cert** - TLS 인증서를 사용하여 인스턴스를 인증합니다. TLS ID를 **IDENTKEY** 로 지정합니다. 유효한 값은 인증서의 CN(일반 이름)에 있는 최대 64자까지의 문자열입니다.
- **user** - 지정된 사용자 또는 그룹 이름으로 인증합니다. 유효한 값은 일부 특수 문자를 포함할 수 있는 영숫자 문자열이며 4~32자 길이입니다.



참고

지원되는 인증 방법은 사용하는 공유 드라이버, 보안 서비스 및 공유 파일 시스템 프로토콜에 따라 달라집니다. 지원되는 공유 파일 시스템 프로토콜은 MapRFS, CEPHFS, NFS, CIFS, GlusterFS 및 HDFS입니다. 지원되는 보안 서비스는 LDAP, Kerberos 프로토콜 또는 Microsoft Active Directory 서비스입니다.

액세스 규칙(ACL)이 공유에 대해 올바르게 구성되었는지 확인하려면 해당 권한을 나열하면 됩니다.



참고

공유에 대한 보안 서비스를 선택할 때 공유 드라이버가 사용 가능한 인증 방법을 사용하여 액세스 규칙을 생성할 수 있는지 여부를 고려해야 합니다. 지원되는 보안 서비스는 LDAP, Kerberos 및 Microsoft Active Directory입니다.

17.9. 공유 유형 액세스 제어

공유 유형은 프로젝트 표시 설명과 추가 사양이라는 프로젝트에서 볼 수 없는 키-값 쌍 목록으로 구성된 관리자 정의 서비스 유형입니다. **manila-scheduler** 는 추가 사양을 사용하여 스케줄링 결정을 내리고 드라이버는 공유 생성을 제어합니다.

관리자는 공유 유형을 생성 및 삭제할 수 있으며 manila 내에서 의미를 부여하는 추가 사양도 관리할 수 있습니다. 프로젝트는 공유 유형을 나열하고 이를 사용하여 새 공유를 생성할 수 있습니다. 공유 유형은 **공용** 및 **비공개**로 생성할 수 있습니다. 공유 유형 목록에서 다른 프로젝트에서 다른 프로젝트를 볼 수 있는지 여부를 정의하고 이를 사용하여 새 공유를 생성하는 데 사용할 수 있는 공유 유형의 가시성 수준입니다.

기본적으로 공유 유형은 공용으로 생성됩니다. 공유 유형을 생성하는 동안 **--is_public** 매개변수를 **False** 로 설정하여 공유 유형을 비공개로 설정하여 다른 프로젝트에서 공유 유형 목록에 보고 새 공유를 생성하지 못하게 합니다. 반면 **공용** 공유 유형은 클라우드의 모든 프로젝트에서 사용할 수 있습니다.

Manila를 사용하면 관리자가 프로젝트의 **개인** 공유 유형에 대한 액세스 권한을 부여하거나 거부할 수 있습니다. 지정된 개인 공유 유형의 액세스에 대한 정보를 가져올 수도 있습니다.



참고

추가 사양으로 인한 공유 유형은 사용자가 공유를 생성하기 전에 백엔드를 필터링하거나 선택하는 데 도움이 되므로 공유 유형에 대한 액세스를 사용하면 특정 백엔드 중에서 클라이언트를 제한할 수 있습니다.

예를 들어 관리자 프로젝트의 관리자는 **my_type** 이라는 개인 공유 유형을 생성하여 목록에서 확인할 수 있습니다. 아래 콘솔 예제에서는 로그인 및 out이 생략되고 현재 로그인한 사용자를 표시하기 위해 환경 변수가 제공됩니다.

```
$ env | grep OS_
...
OS_USERNAME=admin
OS_TENANT_NAME=admin
...
$ manila type-list --all
+-----+-----+-----+-----+-----+-----+
| ID | Name | Visibility | is_default | required_extra_specs | optional_extra_specs |
+-----+-----+-----+-----+-----+-----+
| 4.. | my_type | private | - | driver_handles_share_servers:False | snapshot_support:True |
| 5.. | default | public | YES | driver_handles_share_servers:True | snapshot_support:True |
+-----+-----+-----+-----+-----+-----+
```

데모 프로젝트의 데모 사용자는 유형을 나열할 수 있으며 **my_type** 이라는 개인 공유 유형은 표시되지 않습니다.

```
$ env | grep OS_
...
```

```
OS_USERNAME=demo
OS_TENANT_NAME=demo
...
$ manila type-list --all
```

ID	Name	Visibility	is_default	required_extra_specs	optional_extra_specs
5..	default	public	YES	driver_handles_share_servers:True	snapshot_support:True

관리자는 **df29a37db5ae48d19b349fe947fada46** 과 동일한 프로젝트 ID를 사용하여 데모 프로젝트의 개인 공유 유형에 대한 액세스 권한을 부여할 수 있습니다.

```
$ env | grep OS_
...
OS_USERNAME=admin
OS_TENANT_NAME=admin
...
$ openstack project list
```

ID	Name
df29a37db5ae48d19b349fe947fada46	demo

```
$ manila type-access-add my_type df29a37db5ae48d19b349fe947fada46
```

결과적으로 데모 프로젝트의 사용자는 개인 공유 유형을 보고 공유 생성에 사용할 수 있습니다.

```
$ env | grep OS_
...
OS_USERNAME=demo
OS_TENANT_NAME=demo
...
$ manila type-list --all
```

ID	Name	Visibility	is_default	required_extra_specs	optional_extra_specs
4..	my_type	private	-	driver_handles_share_servers:False	snapshot_support:True
5..	default	public	YES	driver_handles_share_servers:True	snapshot_support:True

지정된 프로젝트에 대한 액세스를 거부하려면 **manila type-access-remove <share_type> <project_id>**를 사용합니다.



참고

공유 유형의 용도를 설명하는 예에서는 두 개의 백엔드인 LVM을 공용 스토리지로, Ceph를 프라이빗 스토리지로 사용하는 경우를 고려하십시오. 이 경우 특정 프로젝트에 대한 액세스 권한을 부여하고 **사용자/그룹** 인증 방법으로 액세스를 제어할 수 있습니다.

17.10. POLICIES

공유 파일 시스템 서비스 API는 역할 기반 액세스 제어 정책과 함께 제공됩니다. 이러한 정책은 특정 방식으로 특정 API에 액세스할 수 있는 사용자가 결정하며 서비스의 **policy.json** 파일에 정의됩니다.



참고

설정 파일 **policy.json** 은 어디에서나 배치할 수 있습니다. 기본적으로 **/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 경로가 예상됩니다.

manila에 API 호출이 수행될 때마다 정책 엔진에서는 적절한 정책 정의를 사용하여 호출을 수락할 수 있는지 결정합니다. 정책 규칙은 API 호출이 허용되는 상황에 따라 결정됩니다. **/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 파일에는 규칙이 항상 허용되는 규칙인 **""**, 사용자 역할 또는 규칙을 기반으로 하는 규칙, 부울 표현식을 사용하는 규칙입니다. 다음은 manila에 대한 **policy.json** 파일의 스니펫입니다. OpenStack 릴리스 간에 변경될 수 있습니다.

```
{
  "context_is_admin": "role:admin",
  "admin_or_owner": "is_admin:True or project_id:%(project_id)s",
  "default": "rule:admin_or_owner",
  "share_extension:quotas:show": "",
  "share_extension:quotas:update": "rule:admin_api",
  "share_extension:quotas:delete": "rule:admin_api",
  "share_extension:quota_classes": "",
}
```

사용자는 정책에서 참조하는 그룹 및 역할에 할당되어야 합니다. 이 작업은 사용자 관리 명령을 사용할 때 서비스에서 자동으로 수행됩니다.



참고

/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json 에 대한 모든 변경 사항은 즉시 적용되므로 manila가 실행되는 동안 새 정책을 구현할 수 있습니다. 정책을 수동으로 수정하면 예기치 않은 부작용이 발생할 수 있으며 권장되지 않습니다. Manila는 기본 정책 파일을 제공하지 않습니다. 모든 기본 정책은 코드 기반 내에 있습니다. 다음을 실행하여 manila 코드에서 기본 정책을 생성할 수 있습니다. **oslopolicy-sample-generator --config-file=var/lib/config-data/puppet-generated/manila/etc/manila/manila-policy-generator.conf**

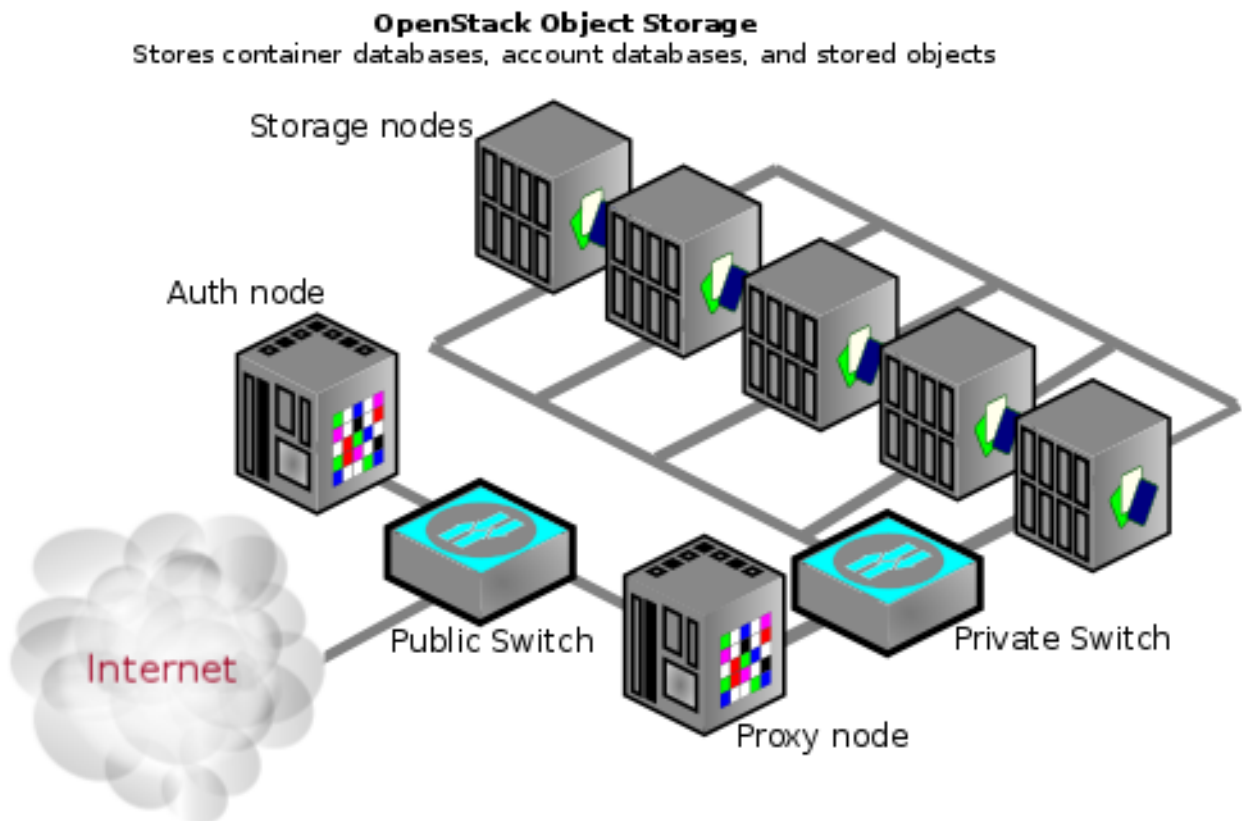
18장. 오브젝트 스토리지

Object Storage(swift) 서비스는 HTTP를 통해 데이터를 저장하고 검색합니다. 오브젝트(데이터 블록)는 익명 읽기 전용 액세스, ACL 정의 액세스 또는 임시 액세스를 제공하도록 구성할 수 있는 조직 계층 구조에 저장됩니다. Swift는 미들웨어를 통해 구현된 여러 토큰 기반 인증 메커니즘을 지원합니다.

애플리케이션은 업계 표준 HTTP RESTful API를 사용하여 오브젝트 스토리지에 데이터를 저장하고 검색합니다. 백엔드 swift 구성 요소는 동일한 RESTful 모델을 따릅니다. 일부 API(예: 지속성 관리)는 클러스터에 비공개로 유지됩니다.

swift의 구성 요소는 다음과 같은 기본 그룹에 속합니다.

- 프록시 서비스
- 인증 서비스
- 스토리지 서비스
 - 계정 서비스
 - 컨테이너 서비스
 - 오브젝트 서비스



참고

오브젝트 스토리지 설치에 인터넷에 연결할 필요가 없으며 조직의 내부 네트워크 인프라의 일부를 공용으로 전환하는 프라이빗 클라우드일 수도 있습니다.

18.1. 네트워크 보안

swift를 위한 보안 강화는 네트워킹 구성 요소 보안부터 시작됩니다. 자세한 내용은 네트워킹 장을 참조하십시오.

고가용성을 위해 rsync 프로토콜은 스토리지 서비스 노드 간에 데이터를 복제하는 데 사용됩니다. 또한 프록시 서비스는 클라이언트 엔드포인트와 클라우드 환경 간의 데이터를 릴레이할 때 스토리지 서비스와 통신합니다.



참고

Swift는 노드 간 통신과 함께 암호화 또는 인증을 사용하지 않습니다. 이는 Swift는 성능상의 이유로 네이티브 rsync 프로토콜을 사용하고 rsync 통신에 SSH를 사용하지 않기 때문입니다. 따라서 아키텍처 다이어그램에 프라이빗 스위치 또는 프라이빗 네트워크([V]LAN)가 표시됩니다. 이 데이터 영역은 다른 OpenStack 데이터 네트워크와 분리되어 있어야 합니다.



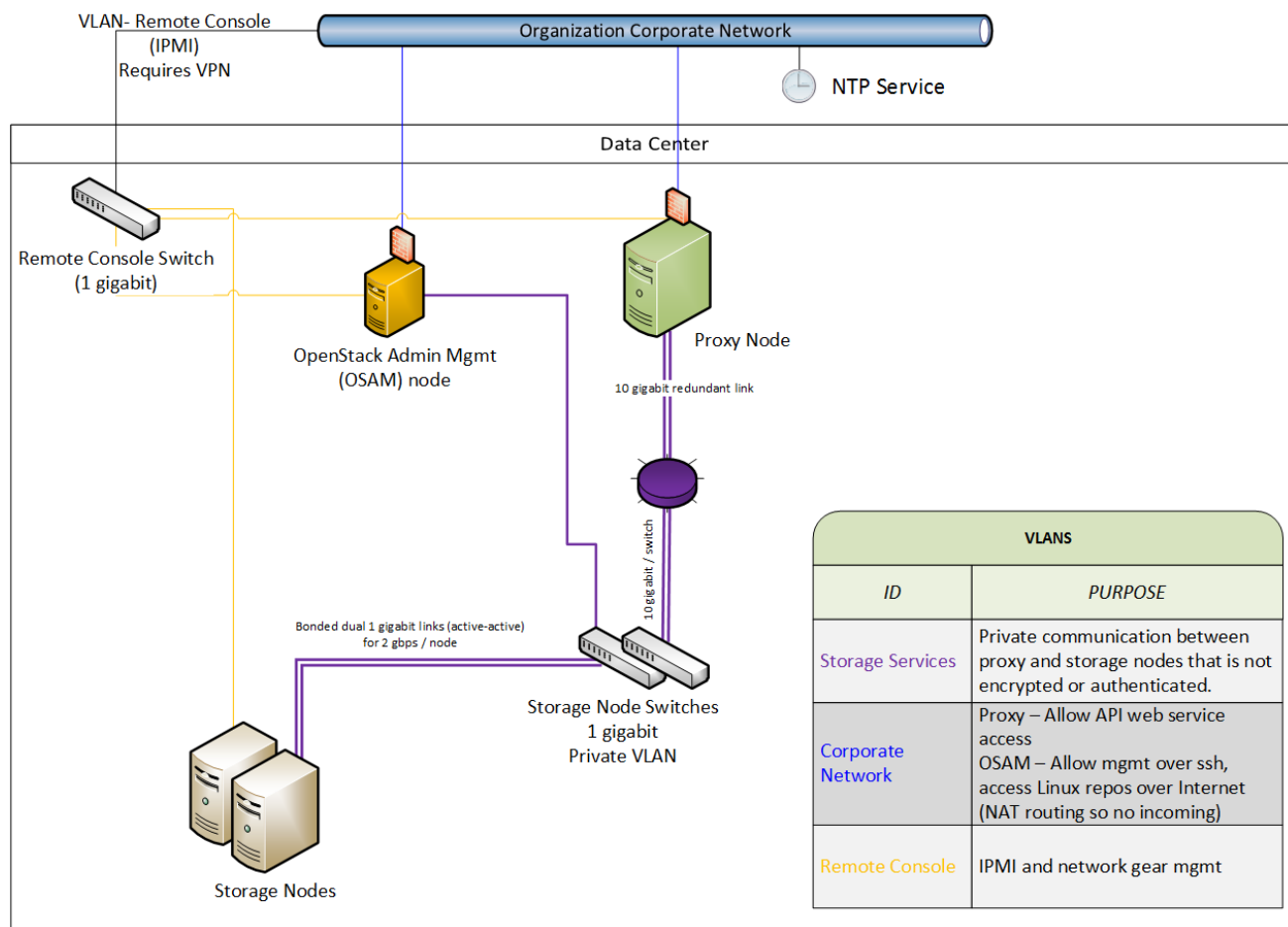
참고

데이터 영역의 스토리지 노드에 대해 사설(V)LAN 네트워크 세그먼트를 사용합니다.

이를 위해서는 프록시 노드에 이중 인터페이스(실제 또는 가상)가 있어야 합니다.

- 소비자가 도달할 수 있는 공용 인터페이스 중 하나입니다.
- 스토리지 노드에 액세스할 수 있는 개인 인터페이스로 또 다른 인터페이스입니다.

다음 그림은 OSAM(관리 노드)과 함께 Object Storage 네트워크 아키텍처를 사용하는 하나의 네트워크 아키텍처를 보여줍니다.



18.2. 루트가 아닌 사용자로 서비스 실행

root가 아닌 (**UID 0**) 서비스 계정에서 실행하도록 swift를 구성하는 것이 좋습니다. 한 가지 권장 사항은 director에서 배포한 대로 기본 그룹 **swift** 를 사용하는 사용자 이름 **swift** 입니다. 오브젝트 스토리지 서비스에는 **proxy-server,container-server,account-server** 가 포함됩니다.

18.3. 파일 권한

/var/lib/config-data/puppet-generated/swift/etc/swift/ 디렉터리에는 링 토폴로지 및 환경 구성에 대한 정보가 포함되어 있습니다. 다음 권한을 사용하는 것이 좋습니다.

```
chown -R root:swift /var/lib/config-data/puppet-generated/swift/etc/swift/*
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type f -exec chmod 640 {} \;
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type d -exec chmod 750 {} \;
```

이 제한에서는 root로만 구성 파일을 수정할 수 있지만 **swift** 그룹의 멤버십으로 인해 서비스를 읽을 수 있습니다.

18.4. 스토리지 서비스 보안

다음은 다양한 스토리지 서비스의 기본 수신 대기 포트입니다.

- 계정 서비스 - **TCP/6002**
- 컨테이너 서비스 - **TCP/6001**
- 오브젝트 서비스 - **TCP/6000**
- Rsync - **TCP/873**



참고

rsync 대신 ssync를 사용하는 경우 오브젝트 서비스 포트가 지속성 유지 관리에 사용됩니다.



참고

스토리지 노드에서 인증이 발생하지 않습니다. 이러한 포트 중 하나에서 스토리지 노드에 연결할 수 있는 경우 인증 없이 데이터에 액세스하거나 수정할 수 있습니다. 이 문제를 완화하려면 이전에 프라이빗 스토리지 네트워크 사용에 대한 권장 사항을 따라야 합니다.

18.5. OBJECT STORAGE 계정 용어

swift 계정은 사용자 계정 또는 인증 정보가 아닙니다. 다음과 같은 차이점이 있습니다.

- Swift 계정 - 컨테이너 컬렉션(사용자 계정 또는 인증 아님). 사용하는 인증 시스템은 계정에 연결된 사용자 및 해당 계정에 액세스하는 방법을 결정합니다.
- Swift 컨테이너 - 오브젝트의 컬렉션입니다. 컨테이너의 메타데이터는 ACL에 사용할 수 있습니다. ACL 사용은 사용된 인증 시스템에 따라 다릅니다.
- Swift 오브젝트 - 실제 데이터 오브젝트입니다. 오브젝트 수준의 ACL도 메타데이터와 함께 사용할 수 있으며 사용되는 인증 시스템에 따라 다릅니다.

각 수준에서는 사용자 액세스를 제어하는 ACL이 있습니다. ACL은 사용 중인 인증 시스템에 따라 해석됩니다. 가장 일반적인 인증 공급자는 ID 서비스(keystone)입니다. 사용자 지정 인증 공급자도 사용할 수 있습니다.

18.6. 프록시 서비스 보안

프록시 노드에는 하나 이상의 공용 및 개인 인터페이스(실제 또는 가상)가 두 개 이상 있어야 합니다. 방화벽 또는 서비스 바인딩을 사용하여 공용 인터페이스를 보호할 수 있습니다. 공용 서비스는 엔드 포인트 클라이언트 요청을 처리하고 인증하고 적절한 작업을 수행하는 HTTP 웹 서버입니다. 개인 인터페이스에는 수신 대기 서비스가 필요하지 않지만 대신 프라이빗 스토리지 네트워크에서 스토리지 노드에 대한 발신 연결을 설정하는 데 사용됩니다.

18.7. HTTP 청취 포트

director는 루트가 아닌(UID 0) 사용자에서 실행되도록 웹 서비스를 설정합니다. **1024** 보다 큰 포트 번호를 사용하면 웹 컨테이너의 일부를 root로 실행하지 않도록 합니다. 일반적으로 HTTP REST API(및 자동 인증을 수행)를 사용하는 클라이언트는 인증 응답에서 필요한 전체 REST API URL을 검색합니다.

OpenStack REST API를 사용하면 클라이언트가 하나의 URL로 인증한 다음 실제 서비스에 대해 완전히 다른 URL을 사용하도록 리디렉션할 수 있습니다. 예를 들어 클라이언트는

https://identity.cloud.example.org:55443/v1/auth 에 인증하고

https://swift.cloud.example.org:44443/v1/AUTH_8980 의 인증 키 및 스토리지 URL(프록시 노드 또는 로드 밸런서의 URL)을 사용하여 응답을 가져올 수 있습니다.

18.8. 로드 밸런서

Apache 사용 옵션이 가능하지 않거나 TLS 작업을 오프로드하려는 성능을 위해 전용 네트워크 장치 로드 밸런서를 사용할 수 있습니다. 이는 여러 프록시 노드를 사용할 때 중복성과 로드 밸런싱을 제공하는 일반적인 방법입니다.

TLS를 오프로드하도록 선택하는 경우 로드 밸런서와 프록시 노드 간의 네트워크 링크가 사설(V)LAN 세그먼트에 있는지 확인합니다. 따라서 네트워크의 다른 노드(아마도 손상될 수 있음)는 암호화되지 않은 트래픽(sniff)을 수행할 수 없습니다. 이러한 위반이 발생한 경우 공격자는 엔드포인트 클라이언트 또는 클라우드 관리자 자격 증명에 액세스하여 클라우드 데이터에 액세스할 수 있습니다.

사용하는 인증 서비스는 끝점 클라이언트에 대한 응답에 다른 URL을 구성하는 방법을 결정하므로 개별 프록시 노드 대신 로드 밸런서를 사용할 수 있습니다.

18.9. 오브젝트 스토리지 인증

Object Storage(swift)는 WSGI 모델을 사용하여 일반 확장성을 제공할 뿐만 아니라 엔드포인트 클라이언트의 인증에도 사용되는 미들웨어 기능을 제공합니다. 인증 공급자는 어떤 역할 및 사용자 유형이 있는지 정의합니다. 일부는 기존 사용자 이름과 암호 자격 증명을 사용하지만 다른 사용자는 API 키 토큰 또는 클라이언트 측 x.509 인증서를 활용할 수 있습니다. 사용자 정의 미들웨어를 사용하여 사용자 정의 공급자를 통합할 수 있습니다.

오브젝트 스토리지에는 기본적으로 두 가지 인증 미들웨어 모듈이 포함되어 있으며, 이 모듈 중 하나는 사용자 정의 인증 미들웨어를 개발하기 위한 샘플 코드로 사용할 수 있습니다.

18.10. AT-REST SWIFT 오브젝트 암호화

Swift는 Barbican과 통합하여 저장된 오브젝트를 투명하게 암호화하고 암호 해독할 수 있습니다. At-rest 암호화는 전송 중 암호화와 다르며 디스크에 저장되는 동안 암호화되는 오브젝트를 나타냅니다.

Swift는 swift에 업로드할 때 오브젝트가 자동으로 암호화되고 사용자에게 제공되면 자동으로 암호 해독 되므로 이러한 암호화 작업을 투명하게 수행합니다. 이 암호화 및 암호 해독은 Barbican에 저장된 동일한 (symmetric) 키를 사용하여 수행됩니다.

추가 리소스

- [OpenStack Key Manager를 사용하여 시크릿 관리](#)

18.11. 추가 항목

모든 노드의 `/var/lib/config-data/puppet-generated/swift/etc/swift/swift.conf`에는 **swift_hash_path_prefix** 설정과 **swift_hash_path_suffix** 설정이 있습니다. 이는 저장되는 오브젝트에 대한 해시 충돌 가능성을 줄이고 한 사용자가 다른 사용자의 데이터를 덮어쓰는 것을 방지하도록 제공됩니다.

이 값은 처음에 암호화 보안 임의 번호 생성기로 설정되어야 하며 모든 노드에서 일관되게 유지되어야 합니다. 적절한 ACL로 보호되고 데이터 손실을 방지하기 위해 백업 사본이 있는지 확인합니다.

19장. 모니터링 및 로깅

로그 관리는 OpenStack 배포의 보안 상태를 모니터링하는 중요한 구성 요소입니다. 로그는 관리자, 프로젝트 및 인스턴스의 BAU 작업과 OpenStack 배포를 구성하는 구성 요소 활동에 대한 인사이트를 제공합니다.

로그는 사전 예방적 보안 및 지속적인 규정 준수 활동에 중요할 뿐만 아니라 조사 및 사고 대응에 중요한 정보 소스이기도 합니다. 예를 들어 keystone 액세스 로그를 분석하면 로그인 실패, 빈도, 원본 IP 및 이벤트가 다른 관련 정보 중에서 계정을 선택할 수 있는지 여부를 알릴 수 있습니다.

director에는 AIDE를 사용한 침입 탐지 기능과 keystone에 대한 CryostatF 감사 기능이 포함되어 있습니다. 자세한 내용은 조정 [인프라](#) 및 [가상화](#)를 참조하십시오.

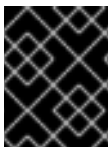
19.1. 모니터링 인프라 강화

중앙 집중식 로깅 시스템은 성공적인 위반으로 인해 이벤트 기록을 지우거나 변조할 수 있으므로 중앙 집중식 로깅 시스템은 침입자의 높은 가치의 대상입니다. 이를 염두에 두고 모니터링 플랫폼을 강화할 것을 권장합니다. 또한 중단 또는 DoS 발생 시 장애 조치 계획을 사용하여 이러한 시스템을 정기적으로 백업하는 것이 좋습니다.

19.2. 모니터링할 이벤트의 예

이벤트 모니터링은 환경 보안에 대한 보다 적극적인 접근 방식으로 실시간 감지 및 대응을 제공합니다. 모니터링에 도움이 될 수 있는 여러 톨이 있습니다. OpenStack 배포의 경우 하드웨어, OpenStack 서비스 및 클라우드 리소스 사용량을 모니터링해야 합니다.

이 섹션에서는 알아야 할 몇 가지 예제 이벤트에 대해 설명합니다.



중요

이 목록은 완전하지 않습니다. 특정 네트워크에 적용할 수 있고 비정상적인 동작을 고려할 수 있는 추가 사용 사례를 고려해야 합니다.

- 로그 생성이 없음을 감지하는 것은 높은 값의 이벤트입니다. 이러한 격차는 서비스 실패 또는 일시적으로 로깅을 해제하거나 추적을 숨기기 위해 로그 수준을 수정한 침입자도 나타낼 수 있습니다.
- 예약되지 않은 시작 또는 중지 이벤트와 같은 애플리케이션 이벤트가 보안에 영향을 미칠 수 있습니다.
- 사용자 로그인 또는 재시작과 같은 OpenStack 노드의 운영 체제 이벤트입니다. 이를 통해 시스템의 적절하고 부적절한 사용을 구분하는 데 중요한 통찰력을 제공할 수 있습니다.
- 네트워킹 브리지가 다운됩니다. 이는 서비스 중단의 위험 때문에 실행 가능한 이벤트입니다.
- iptables가 컴퓨팅 노드에서 이벤트를 플러시하고 이로 인해 인스턴스에 대한 액세스 권한이 손실됩니다.

ID 서비스의 사용자, 프로젝트 또는 도메인 삭제에서 분리된 인스턴스에서 보안 위험을 줄이기 위해 시스템에서 알람을 생성하고 OpenStack 구성 요소가 인스턴스 종료, 연결된 볼륨 연결 해제, CPU 및 스토리지 리소스 회수 등 이러한 이벤트에 적절하게 응답하도록 논의가 있습니다.

침입 탐지 소프트웨어, 안티바이러스 소프트웨어 및 제거 유틸리티와 같은 보안 모니터링 제어는 공격 또는 침입의 시기와 방법을 보여주는 로그를 생성할 수 있습니다. 이러한 톨은 OpenStack 노드에 배포할 때

보호 계층을 제공할 수 있습니다. 프로젝트 사용자는 인스턴스에서 이러한 도구를 실행하려고 할 수도 있습니다.

20장. 프로젝트의 데이터 개인 정보

OpenStack은 다양한 데이터 요구 사항이 있는 프로젝트 간에 멀티 테넌시를 지원하도록 설계되었습니다. 클라우드 운영자는 해당 데이터 개인 정보 보호 문제 및 규정을 고려해야 합니다. 이 장에서는 OpenStack 배포에 대한 데이터 레지던스 및 유지 관리의 측면을 다룹니다.

20.1. 데이터 레지던스

개인 정보 보호 및 격리는 지난 몇 년 동안 클라우드 채택의 주요 장벽으로 지속적으로 인용되었습니다. 클라우드에서 데이터를 소유하고 있고 클라우드 Operator가 궁극적으로 이러한 데이터의 보호자로 신뢰할 수 있는지에 대한 우려는 과거의 중요한 문제였습니다.

특정 OpenStack 서비스는 프로젝트 또는 프로젝트 정보에 속하는 데이터 및 메타데이터에 액세스할 수 있습니다. 예를 들어 OpenStack 클라우드에 저장된 프로젝트 데이터에는 다음 항목이 포함될 수 있습니다.

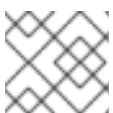
- 오브젝트 스토리지 오브젝트.
- 컴퓨팅 인스턴스 임시 파일 시스템 스토리지.
- 컴퓨팅 인스턴스 메모리.
- 블록 스토리지 볼륨 데이터.
- Compute 액세스를 위한 공개 키입니다.
- 이미지 서비스의 가상 머신 이미지.
- 인스턴스 스냅샷.
- 컴퓨팅의 구성 드라이브 확장에 전달된 데이터입니다.

OpenStack 클라우드에 의해 저장된 메타데이터에는 다음 항목이 포함됩니다(이 목록은 필수가 아님).

- 조직 이름.
- 사용자의 "실제 이름".
- 실행 중인 인스턴스, 버킷, 오브젝트, 볼륨 및 기타 할당량 관련 항목의 수 또는 크기입니다.
- 인스턴스 실행 또는 데이터 저장 시간 수입니다.
- 사용자의 IP 주소입니다.
- 컴퓨팅 이미지 번들링을 위해 내부적으로 생성된 개인 키입니다.

20.2. 데이터 처리

모범 사례에 따라 운영자는 조직 제어가 해제되기 전 또는 재사용을 위해 릴리스하기 전에 클라우드 시스템 미디어(digital 및 non-digital)를 포기해야 합니다. Sanitization 방법은 특정 보안 도메인과 정보의 민감도를 고려하여 적절한 수준의 강도와 무결성을 구현해야 합니다.



참고

NIST Special Publication 800-53 버전 4는 이 주제에 대해 구체적으로 설명합니다.

The sanitization process removes information from the media such that the information cannot be retrieved or reconstructed. Sanitization techniques, including clearing, purging, cryptographic erase, and destruction, prevent the disclosure of information to unauthorized individuals when such media is reused or released for disposal.

클라우드 운영자는 일반 데이터 처리 및 시행 지침을 개발할 때 (NIST 권장 보안 제어)를 고려해야 합니다.

- 미디어 활성화 및 삭제 작업을 추적, 문서화 및 확인합니다.
- 적절한 성능을 확인하기 위해 장비 및 절차를 테스트하십시오.
- 이러한 장치를 클라우드 인프라에 연결하기 전에 이식 가능한 이동식 스토리지 장치를 삭제합니다.
- 폐기할 수 없는 클라우드 시스템 미디어를 삭제합니다.

따라서 OpenStack 배포에서는 다음과 같은 사례를 처리해야 합니다(다른 배포).

- 안전한 데이터 삭제
- 인스턴스 메모리 스크러빙
- 블록 스토리지 볼륨 데이터
- 컴퓨팅 인스턴스 임시 스토리지
- 베어 메탈 서버 종료

20.2.1. 데이터가 안전하게 삭제되지 않음

OpenStack에서는 일부 데이터가 삭제될 수 있지만 위에 설명된 NIST 표준의 컨텍스트에서는 안전하게 삭제되지 않을 수 있습니다. 이는 일반적으로 데이터베이스에 저장된 대부분의 또는 모든 메타데이터 및 정보에 적용할 수 있습니다. 이는 자동 비우기 및 주기적인 자유 공간(free-space wiping)을 위해 데이터베이스 및/또는 시스템 구성으로 수정될 수 있습니다.

20.2.2. 인스턴스 메모리 스크러빙

다양한 하이퍼바이저에 특정한 것은 인스턴스 메모리 처리입니다. 이 동작은 Compute에 정의되지 않지만 일반적으로 하이퍼바이저는 인스턴스를 삭제할 때 또는 둘 다 또는 둘 다 인스턴스를 삭제할 때 메모리를 스크럽하기 위해 최선의 노력을 기울입니다.

20.3. CINDER 볼륨 데이터 암호화

OpenStack 볼륨 암호화 기능을 사용하는 것이 좋습니다. 이 내용은 볼륨 암호화의 데이터 암호화 섹션에서 자세히 설명합니다. 이 기능을 사용하면 암호화 키를 안전하게 삭제하여 데이터 삭제를 수행합니다. 최종 사용자는 볼륨을 생성하는 동안 이 기능을 선택할 수 있지만 관리자는 먼저 볼륨 암호화 기능을 일회성으로 설정해야 합니다.

OpenStack 볼륨 암호화 기능을 사용하지 않으면 일반적으로 다른 접근 방식을 활성화하기가 더 어렵습니다. 백엔드 플러그인을 사용하는 경우 암호화 또는 비표준 덮어쓰기 솔루션을 수행하는 독립적인 방법이 있을 수 있습니다. OpenStack Block Storage에 대한 플러그인은 다양한 방식으로 데이터를 저장합니다. 많은 플러그인은 공급 업체 또는 기술에 고유하지만 다른 플러그인은 파일 시스템 (예: LVM 또는 ZFS)에 관한 더 많은 Cryostat 솔루션입니다. 데이터를 안전하게 삭제하는 방법은 플러그인, 벤더 및 파일 시스템마다 다릅니다.

일부 백엔드(예: ZFS)는 데이터 노출을 방지하기 위해 COW(Copy-On-Write)를 지원합니다. 이 경우 작성된 블록에서 읽기는 항상 0을 반환합니다. 기타 백엔드(예: LVM)는 기본적으로 이를 지원하지 않을 수 있으므로 cinder 플러그인은 사용자에게 전달하기 전에 이전에 작성된 블록을 재정의해야 합니다. 선택한 블록 백엔드에서 제공하는 보장을 검토하고 제공되지 않는 보장에 사용할 수 있는 수정을 확인하는 것이 중요합니다.

20.4. 이미지 서비스 지연 삭제 기능

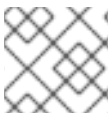
이미지 서비스에는 지연된 삭제 기능이 있으며, 이 기능은 정의된 기간 동안 이미지 삭제를 일시 중지합니다. 이 동작이 보안에 관련된 경우 이 기능을 비활성화하는 것이 좋습니다. **glance-api.conf** 파일을 편집하고 **delayed_delete** 옵션을 **False** 로 설정하여 이 작업을 수행할 수 있습니다.

20.5. 컴퓨팅 소프트 삭제 기능

컴퓨팅에는 정의된 기간 동안 삭제된 인스턴스가 소프트 삭제 상태가 되도록 하는 소프트 삭제 기능이 있습니다. 이 기간 동안 인스턴스를 복원할 수 있습니다. **soft-delete** 기능을 비활성화하려면 **/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf** 파일을 편집하고 **reclaim_instance_interval** 옵션을 비워 둡니다.

20.6. 베어 메탈 프로비저닝의 보안 강화

베어 메탈 프로비저닝 인프라의 경우 일반적으로 BMC(Baseboard Management Controller) 및 IPMI를 강화하는 것이 좋습니다. 예를 들어 프로비저닝 네트워크 내에서 이러한 시스템을 분리하고, 기본이 아닌 강력한 암호를 구성하고, 원하지 않는 관리 기능을 비활성화할 수 있습니다. 자세한 내용은 이러한 구성 요소 보안 강화에 대한 벤더의 지침을 참조하십시오.



참고

가능한 경우 기존의 BMC보다 Redfish 기반 BMC를 평가하는 것이 좋습니다.

20.7. 하드웨어 식별

서버를 배포할 때 항상 공격자의 서버와 구별할 수 있는 안정적인 방법이 없을 수 있습니다. 이 기능은 어느 정도 하드웨어/BMC에 따라 달라질 수 있지만 일반적으로 서버에 구축된 검증 가능한 식별 수단이 없는 것처럼 보입니다.

20.8. 데이터 암호화

이 옵션은 구현자가 디스크에 저장되거나 네트워크를 통해 전송되는 경우(예: 아래 설명된 OpenStack 볼륨 암호화 기능)를 통해 프로젝트 데이터를 암호화하는 데 있습니다. 이는 사용자가 공급자에게 전송하기 전에 자신의 데이터를 암호화하는 일반적인 권장 사항 이상입니다.

프로젝트를 대신하여 데이터를 암호화하는 것의 중요성은 공격자가 프로젝트 데이터에 액세스할 수 있는 공급자가 가정하는 위험과 관련이 있습니다. 여기에는 정부 요구 사항, 정책별 요구 사항, 프라이빗 계약 또는 퍼블릭 클라우드 공급자의 프라이빗 계약 관련 법률이 있을 수 있습니다. 프로젝트 암호화 정책을 선택하기 전에 위험 평가 및 법적 조언을 얻는 것이 좋습니다.

인스턴스별 또는 개체별 암호화는 내림차순, 프로젝트별, 호스트별 및 클라우드별 집계에 더 적합합니다. 이 권장 사항은 구현의 복잡성과 어려움에 반하지 않습니다. 현재 일부 프로젝트에서는 프로젝트별로도 암호화를 느슨하게 세밀하게 구현하기가 어렵거나 불가능합니다. 구현자는 프로젝트 데이터를 암호화할 때 고려해야 합니다.

종종 데이터 암호화는 단순히 키를 던지고 프로젝트 및 인스턴스별 데이터를 안정적으로 삭제하는 기능과 관련이 있습니다. 이렇게 하면 안정적이고 안전한 방식으로 키를 파괴하는 것이 중요합니다.

사용자에 대한 데이터를 암호화할 수 있는 기회가 있습니다.

- Object Storage 오브젝트
- 네트워크 데이터

20.8.1. 볼륨 암호화

OpenStack의 볼륨 암호화 기능은 프로젝트별로 개인 정보 보호를 지원합니다. 지원되는 기능은 다음과 같습니다.

- 대시보드 또는 명령줄 인터페이스를 통해 시작된 암호화된 볼륨 유형 생성 및 사용
- 암호화 알고리즘 및 키 크기와 같은 선택 매개변수 활성화
- iSCSI 패킷에 포함된 볼륨 데이터가 암호화됨
- 원래 볼륨이 암호화된 경우 암호화된 백업 지원
- 볼륨 암호화 상태의 대시보드 표시입니다. 볼륨이 암호화되었음을 나타내고 알고리즘 및 키 크기와 같은 암호화 매개변수를 포함합니다.
- 키 관리 서비스와 인터페이스

20.8.2. Object Storage 오브젝트

Object Storage(swift)는 스토리지 노드에서 미사용 오브젝트 데이터의 선택적 암호화를 지원합니다. 오브젝트 데이터의 암호화는 권한이 없는 당사자가 디스크에 대한 물리적 액세스 권한을 얻는 경우 사용자의 데이터를 읽을 위험을 완화하기 위한 것입니다.

미사용 데이터의 암호화는 프록시 서버 WSGI 파이프라인에 포함될 수 있는 미들웨어에 의해 구현됩니다. 기능은 swift 클러스터 내부이며 API를 통해 노출되지 않습니다. 클라이언트는 이 기능을 통해 이 기능을 통해 swift 서비스로 암호화됨을 인식하지 못합니다. 내부적으로 암호화된 데이터는 swift API를 통해 클라이언트에 반환되지 않아야 합니다.

다음 데이터는 swift에 있는 동안 암호화됩니다.

- 예를 들어 오브젝트 콘텐츠(예: 오브젝트 **PUT** 요청 본문)입니다.
- 콘텐츠가 0이 아닌 오브젝트의 entity 태그(**ETag**)입니다.
- 모든 사용자 정의 사용자 오브젝트 메타데이터 값입니다. 예를 들어 **X-Object-Meta-** 접두사가 지정된 헤더와 **PUT** 또는 **POST** 요청을 사용하여 전송된 메타데이터입니다.

위 목록에 포함되지 않은 데이터 또는 메타데이터는 다음을 포함하여 암호화되지 않습니다.

- 계정, 컨테이너 및 오브젝트 이름
- 계정 및 컨테이너 사용자 정의 사용자 메타데이터 값
- 모든 사용자 정의 사용자 메타데이터 이름
- Object Content-Type values

- 오브젝트 크기
- 시스템 메타데이터

20.8.3. 블록 스토리지 성능 및 백엔드

운영 체제를 활성화할 때 Intel 및 AMD 프로세서 모두에서 사용 가능한 하드웨어 가속 기능을 사용하여 OpenStack 블록 암호화 성능을 향상시킬 수 있습니다.

OpenStack 블록 암호화 기능은 호스트에서 **dm-crypt** 또는 기본 **QEMU** 암호화 지원을 사용하여 블록 데이터를 보호합니다. 암호화된 블록을 생성할 때 **LUKS** 블록 암호화 유형을 사용하는 것이 좋습니다.

20.8.4. 네트워크 데이터

컴퓨팅 노드의 프로젝트 데이터는 IPsec 또는 기타 터널을 통해 암호화할 수 있습니다. 이 방법은 OpenStack에서 일반 또는 표준이 아니지만 의도하고 관심 있는 구현자에게 사용할 수 있는 옵션입니다. 마찬가지로 암호화된 데이터는 네트워크를 통해 전송될 때 암호화됩니다.

20.9. 키 관리

프로젝트 데이터 개인 정보 보호에 대해 자주 언급되는 문제를 해결하기 위해 OpenStack 커뮤니티 내에서 데이터 암호화를 보다 유비쿼터스로 만드는 것이 중요합니다. 최종 사용자가 데이터를 저장하기 전에 데이터를 암호화하는 것은 비교적 쉽습니다. 이는 미디어 파일, 데이터베이스 아카이브와 같은 프로젝트 오브젝트에 대한 실행 가능한 경로입니다. 경우에 따라 클라이언트 측 암호화는 향후 사용하기 위해 데이터를 암호 해독하기 위해 클라이언트 상호 작용(예: 현재 키)이 필요한 가상화 기술이 보유한 데이터를 암호화하는 데 사용됩니다.

Barbican은 프로젝트에서 데이터를 보다 원활하게 암호화하고 키 관리로 사용자에게 부담을 주지 않고도 액세스할 수 있도록 지원합니다. OpenStack의 일환으로 암호화 및 키 관리 서비스를 제공하면 데이터 채택이 쉬워지고 데이터 개인 정보 보호 또는 오용에 대한 고객의 우려를 해결할 수 있습니다.

블록 암호화 기능은 키 관리자 서비스(barbican)와 같은 키 관리 서비스를 사용하여 키 관리 서비스를 사용하여 키의 생성 및 보안 강화 스토리지를 사용합니다.

21장. 인스턴스 보안 관리

가상화된 환경에서 인스턴스 실행의 이점 중 하나는 일반적으로 베어 메탈에 배포할 때 사용할 수 없는 보안 제어의 새로운 기회입니다. 특정 기술을 가상화 스택에 적용하여 OpenStack 배포에 대한 정보 보장을 개선할 수 있습니다. 강력한 보안 요구 사항이 있는 Operator는 이러한 기술을 배포하는 것을 고려할 수 있지만 모든 상황에 적용되는 것은 아닙니다. 경우에 따라 규범적인 비즈니스 요구 사항으로 인해 클라우드에서 기술 사용을 배제할 수 있습니다. 마찬가지로 일부 기술은 시스템 사용자에게 바람직하지 않을 수 있는 실행 상태와 같은 인스턴스 데이터를 검사합니다.

이 장에서는 이러한 기술과 인스턴스 또는 기본 노드의 보안을 개선하는 데 사용할 수 있는 상황을 설명합니다. 데이터 통과, 인트로스펙션 또는 엔트로피 소스를 포함할 수 있는 가능한 개인 정보 보호 문제도 강조되어 있습니다.

21.1. 인스턴스에 엔트로피 제공

엔트로피는 인스턴스에서 사용할 수 있는 임의의 데이터의 품질 및 소스를 나타냅니다. 암호화 기술은 일반적으로 엔트로피 풀의 드로잉이 필요한 무작위성에 의존합니다. 엔트로피 주의를 인스턴스가 암호화 기술에 필요한 무작위성을 지원하기에 충분한 엔트로피를 얻을 수 없을 때 발생합니다. 엔트로피 부족은 경우에 따라 관련이 없는 것처럼 보일 수 있습니다. 예를 들어 부팅 시간이 느리면 인스턴스가 SSH 키 생성을 대기하기 때문에 발생할 수 있습니다. 엔트로피 부족의 가능성은 클라우드 사용자가 인스턴스 내에서 품질이 좋지 않은 엔트로피 소스를 사용하도록 유도하여 클라우드에서 실행되는 애플리케이션을 덜 안전하게 만들 수 있습니다.

인스턴스에 높은 엔트로피 소스를 제공하려면 인스턴스를 지원하기 위해 클라우드에 충분한 하드웨어 임의 번호 생성기(HRNG)가 필요합니다. 일상적인 작업을 위해 최신 HRNG는 50-100 컴퓨팅 노드를 지원하기에 충분한 엔트로피를 생성할 수 있습니다. 높은 대역폭 HRNG는 더 많은 노드를 처리할 수 있습니다. 충분한 엔트로피를 사용할 수 있도록 클라우드에 대한 애플리케이션 요구 사항을 식별해야 합니다.

VirtIO RNG는 `/dev/urandom` 을 기본적으로 엔트로피 소스로 사용하여 인스턴스가 부팅 시 엔트로피를 사용하지 않도록 하는 임의 번호 생성기입니다. 또한 HRNG를 사용하거나 엔트로피 수집 데몬(EGD)과 같은 툴을 구성하여 배포를 통해 엔트로피를 배포할 수 있습니다. virtio RNG 장치는 기본적으로 인스턴스에 대해 활성화됩니다. 인스턴스의 Virtio RNG 장치를 비활성화하려면 인스턴스 플레이어에서 `hw_rng:allowed` 를 **False** 로 설정해야 합니다.

21.2. 노드에 인스턴스 예약

인스턴스를 생성하기 전에 이미지 인스턴스화를 위한 호스트를 선택해야 합니다. 이 선택은 **nova-scheduler** 에서 컴퓨팅 및 볼륨 요청을 디스패치하는 방법을 결정합니다.

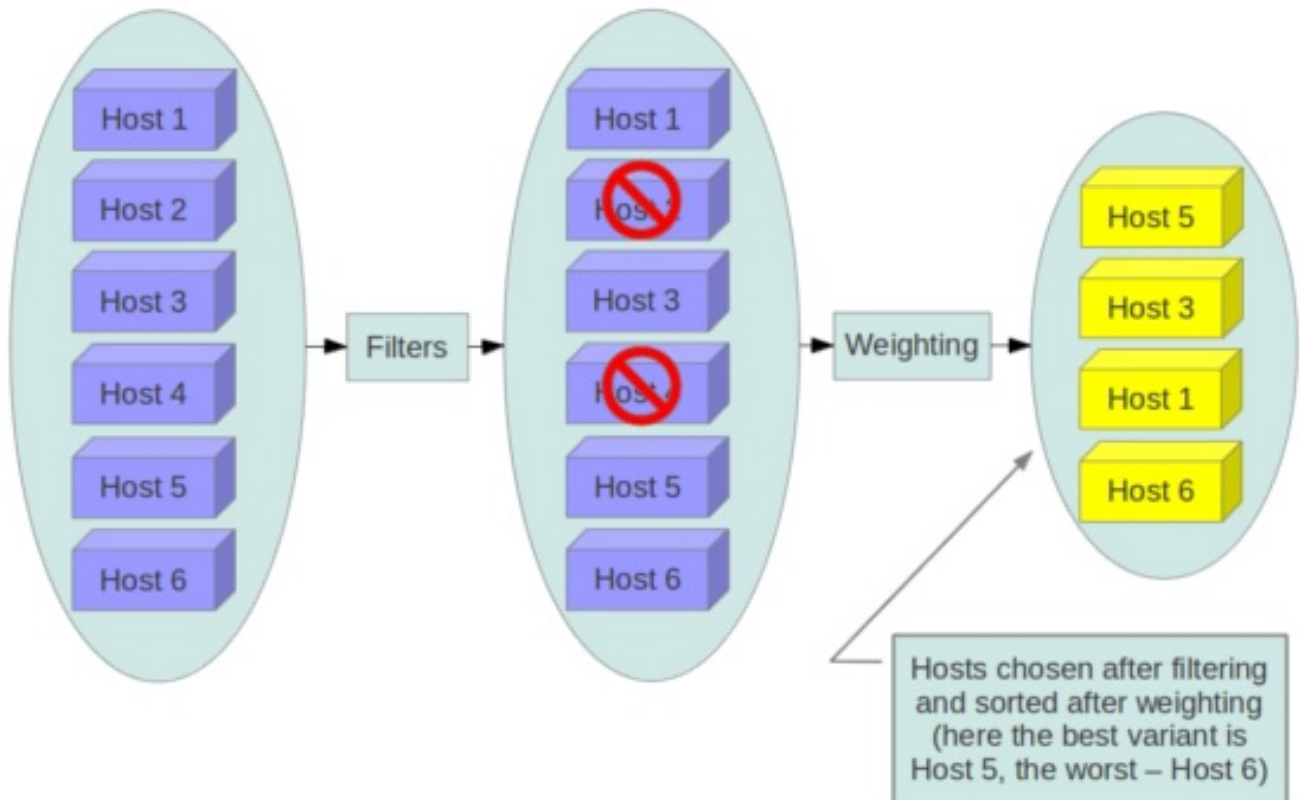
FilterScheduler 는 다른 스케줄러가 있지만 Compute의 기본 스케줄러입니다. 이 기능은 *필터 팁과 함께* 작동하여 인스턴스를 시작해야 하는 위치를 결정합니다. 이 호스트 선택 프로세스를 통해 관리자는 다양한 보안 및 규정 준수 요구 사항을 충족할 수 있습니다. 데이터 격리가 주요 관심사인 경우 가능한 한 프로젝트 인스턴스가 동일한 호스트에 상주하도록 선택할 수 있습니다. 반대로 가용성 또는 내결함성을 위해 인스턴스가 가능한 한 많은 다른 호스트에 상주하도록 시도할 수 있습니다.

필터 스케줄러는 다음과 같은 주요 카테고리에 속합니다.

- **리소스 기반 필터** - 하이퍼바이저 호스트 세트의 시스템 리소스 사용량에 따라 인스턴스의 배치를 결정하고 RAM, IO 또는 CPU 사용률과 같은 무료 또는 사용 속성을 트리거할 수 있습니다.
- **이미지 기반 필터** - VM의 운영 체제 또는 사용된 이미지 유형과 같이 사용되는 이미지 메타데이터를 기반으로 인스턴스 생성을 위임합니다.
- **환경 기반 필터** - 특정 IP 범위 내, 가용성 영역 또는 다른 인스턴스와 동일한 호스트와 같은 외부 세부 정보를 기반으로 인스턴스의 배치를 결정합니다.

- **사용자 지정 기준** - 신뢰 또는 메타데이터 구문 분석과 같은 관리자 제공 기준을 기반으로 인스턴스 생성을 위임합니다.

여러 필터를 한 번에 적용할 수 있습니다. 예를 들어 **ServerGroupAffinity** 필터는 특정 호스트 세트 멤버에서 인스턴스가 생성되었는지 확인하고 **ServerGroupAntiAffinity** 필터는 동일한 인스턴스가 다른 특정 호스트 세트에서 생성되지 않는지 확인합니다. 이 두 필터는 일반적으로 동시에 활성화되며 지정된 속성의 값에 대해 각 점검이 동시에 true일 수 없으므로 서로 충돌할 수 없습니다.



중요

사용자가 제공하거나 조작(예: 메타데이터)할 수 있는 오브젝트를 구문 분석하는 필터를 비활성화하는 것이 좋습니다.

21.3. 신뢰할 수 있는 이미지 사용

클라우드 환경에서 사용자는 사전 설치된 이미지 또는 자체적으로 업로드하는 이미지와 함께 작동합니다. 두 경우 모두 사용자가 사용하는 이미지가 변조되지 않았는지 확인할 수 있어야 합니다. 이미지를 확인하는 기능은 보안을 위해 기본적인 필수 요소입니다. 이미지 소스에서 사용되는 대상까지 신뢰 체인이 필요합니다. 이 작업은 신뢰할 수 있는 소스에서 가져온 이미지에 서명하고 사용하기 전에 서명을 확인하여 수행할 수 있습니다. 확인된 이미지를 확보하고 생성하는 다양한 방법에 대해서는 아래에서 설명하고, 이미지 서명 확인 기능에 대한 설명은 다음과 같습니다.

21.4. 이미지 생성

Red Hat OpenStack Image 서비스(glance)에 이미지를 생성하고 업로드하는 방법에 대한 지침은 [이미지 생성 및 관리](#)를 참조하십시오. 보안 강화를 위해 환경에 대해 신뢰할 수 있는 이미지를 사용하고, 추가 보호를 위해 조직의 강화 지침을 사용하십시오. 다음과 같은 몇 가지 방법 중 하나로 환경에 대한 이미지를

가져올 수 있습니다.

인스턴스 미디어 다운로드

신뢰할 수 있는 소스에서 부팅 미디어를 얻으려면 공식 Red Hat 소스에서 이미지를 다운로드하고 검증 위해 SHA256SUM을 사용합니다.

ISO에서 이미지 생성

설치 프로세스에서 이미지를 생성하는 방법에 대한 자세한 내용은 [Red Hat Enterprise Linux 9 이미지 생성](#) 을 참조하십시오.

이미지 빌더 사용

disk-image-builder 를 사용하여 OpenStack 내에서 목적에 필요한 구성 요소만 있는 최소 시스템을 생성할 수 있습니다. **disk-image-builder** 를 사용하여 사용자 지정 이미지를 생성하는 방법에 대한 자세한 내용은 사용자 [지정된 RHEL 시스템 이미지](#) 구성을 참조하십시오.

21.5. 이미지 서명 확인

이미지 서명 확인을 활성화하여 Compute 서비스(nova)가 인스턴스를 시작하기 전에 Image 서비스(glance) 이미지에 무단 변경 사항이 포함되어 있지 않은지 확인할 수 있습니다. 이 기능을 사용하면 악성 코드 또는 보안 취약점이 포함될 수 있는 새 인스턴스가 시작되지 않도록 합니다.

사전 요구 사항

- Red Hat OpenStack Platform director 환경이 설치되어 있어야 합니다.
- stack으로 director에 로그인되어 있습니다.

프로세스

1. heat 템플릿에서 **True** 값을 **VerifyGlanceSignatures** 매개변수로 설정하여 인스턴스 서명 검증 기능을 활성화합니다.

```
parameter_defaults:
  VerifyGlanceSignatures: True
```

2. **VerifyGlanceSignatures** 매개변수를 수정하는 데 사용하는 템플릿이 **openstack overcloud deploy** 스크립트에 포함되어 있고 배포 스크립트를 재실행합니다.



참고

서명하지 않은 이미지로 인스턴스를 생성하면 이미지 확인이 실패하고 인스턴스가 시작되지 않습니다. 이미지에 서명하는 방법에 대한 자세한 내용은 [이미지 서비스 이미지 서명](#) 을 참조하십시오.

21.6. 인스턴스 마이그레이션

OpenStack과 기본 가상화 계층은 OpenStack 노드 간에 이미지를 실시간 마이그레이션을 제공하므로 인스턴스 다운타임 없이 컴퓨팅 노드의 롤링 업그레이드를 원활하게 수행할 수 있습니다. 그러나 실시간 마이그레이션에도 상당한 위험이 있습니다. 다음은 실시간 마이그레이션 중에 수행되는 높은 수준의 단계입니다.

1. 대상 호스트에서 인스턴스 시작
2. 전송 메모리

3. 게스트 및 동기화 디스크 중지
4. 상태 전송
5. 게스트 시작



참고

콜드 마이그레이션, 크기 조정 및 보류와 같은 특정 작업은 모두 인스턴스의 데이터를 네트워크를 통해 다른 서비스로 전송할 수 있습니다.

21.6.1. 실시간 마이그레이션 위험

실시간 마이그레이션 프로세스의 다양한 단계에서 인스턴스의 런타임 메모리 및 디스크 내용이 일반 텍스트로 네트워크를 통해 전송됩니다. 따라서 실시간 마이그레이션을 사용할 때 처리해야 하는 여러 가지 위험이 있습니다. 다음의 비-기타적인 목록은 이러한 위험 중 일부를 자세히 설명합니다.

- 서비스 거부(DoS): 마이그레이션 프로세스 중에 문제가 발생하면 인스턴스가 손실될 수 있습니다.
- 데이터 노출: 메모리 또는 디스크 전송은 안전하게 처리되어야 합니다.
- 데이터 조작: 메모리 또는 디스크 전송이 안전하게 처리되지 않으면 공격자는 마이그레이션 중에 사용자 데이터를 조작할 수 있습니다.
- 코드 삽입: 메모리 또는 디스크 전송이 안전하게 처리되지 않으면 공격자는 디스크 또는 메모리에 실행 파일을 조작할 수 있습니다.

21.6.2. 암호화된 실시간 마이그레이션

실시간 마이그레이션 트래픽은 실행 중인 인스턴스의 디스크 및 메모리 내용을 일반 텍스트로 전송하며 현재 기본적으로 내부 API 네트워크에서 호스팅됩니다.

실시간 마이그레이션을 활성화하기 위해 충분한 업그레이드(예: 업그레이드)가 있는 경우 libvirt에서 실시간 마이그레이션에 암호화된 터널을 제공할 수 있습니다. 그러나 이 기능은 OpenStack Dashboard 또는 nova-client 명령에는 노출되지 않으며 libvirt의 수동 구성을 통해서만 액세스할 수 있습니다. 그러면 실시간 마이그레이션 프로세스가 다음과 같은 상위 수준 단계로 변경됩니다.

1. 인스턴스 데이터는 하이퍼바이저에서 libvirt로 복사됩니다.
2. 암호화된 터널은 소스 및 대상 호스트의 libvirt 프로세스 간에 생성됩니다.
3. 대상 libvirt 호스트는 인스턴스를 기본 하이퍼바이저에 다시 복사합니다.



참고

Red Hat OpenStack Platform 13의 경우 권장되는 접근 방식은 Ceph를 백엔드로 사용할 때 기본적으로 활성화되어 있는 터널링된 마이그레이션을 사용하는 것입니다. 자세한 내용은 https://docs.openstack.org/nova/queens/configuration/config.html#libvirt.live_migration_t를 참조하십시오.

21.7. 모니터링, 경고 및 보고

인스턴스는 호스트 간에 복제할 수 있는 서버 이미지입니다. 결과적으로 물리적 호스트와 가상 호스트 간

에 유사한 로깅을 적용하는 것이 좋습니다. 호스트 및 데이터에 대한 액세스 이벤트, 사용자 추가 및 제거, 권한 변경 및 요구 사항에 따라 기타 이벤트를 포함하여 운영 체제 및 애플리케이션 이벤트를 기록해야 합니다. 로그 이벤트를 수집하고 분석을 위해 상호 연관되며 참조 또는 추가 작업을 위해 해당 결과를 저장하는 로그 수집기로 결과를 내보내는 것이 좋습니다. 이를 수행하는 일반적인 도구 중 하나는 ELK 스택 또는 Elasticsearch, Logstash 및 Kibana입니다.



참고

이러한 로그를 정기적으로 검토하거나 NOC(네트워크 운영 센터)에서 수행하는 실시간 보기 내에서 모니터링해야 합니다.

이후에 조치를 위해 응답자에게 전송되는 경고를 트리거할 이벤트를 추가로 결정해야 합니다.

21.8. 업데이트 및 패치

하이퍼바이저는 독립적인 가상 머신을 실행합니다. 이 하이퍼바이저는 운영 체제에서 실행하거나 하드웨어에서 직접 실행할 수 있습니다(bare metal라고 함). 하이퍼바이저 업데이트는 가상 머신에 전파되지 않습니다. 예를 들어 배포에서 KVM을 사용하고 있고 CentOS 가상 머신 세트가 있는 경우 KVM에 대한 업데이트는 CentOS 가상 머신에서 실행되는 모든 항목을 업데이트하지 않습니다.

가상 머신의 강화, 배포 및 지속적인 기능을 담당하는 가상 머신의 명확한 소유권을 소유자에게 할당하는 것이 좋습니다. 또한 프로덕션과 유사한 환경에서 먼저 업데이트를 테스트하면서 정기적으로 업데이트를 배포할 계획이 있어야 합니다.

21.9. 방화벽 및 인스턴스 프로필

대부분의 일반적인 운영 체제에는 추가 보안 계층을 위한 호스트 기반 방화벽이 포함됩니다. 인스턴스는 가능한 한 적은 수의 애플리케이션(사용 가능한 경우 단일 용도의 인스턴스)을 실행해야 하지만, 인스턴스에서 실행 중인 모든 애플리케이션은 애플리케이션에 액세스해야 하는 시스템 리소스, 실행을 위해 필요한 가장 낮은 수준의 권한, 가상 시스템에서 들어오고 들어오는 예상 네트워크 트래픽을 결정하도록 프로파일링해야 합니다. 이 예상 트래픽은 SSH 또는 RDP와 같은 필요한 로깅 및 관리 통신과 함께 허용된 트래픽으로 호스트 기반 방화벽에 추가해야 합니다. 다른 모든 트래픽은 방화벽 구성에서 명시적으로 거부되어야 합니다.

Linux 인스턴스에서 위의 애플리케이션 프로필을 **audit2allow** 와 같은 툴과 함께 사용하여 대부분의 Linux 배포판에서 중요한 시스템 정보를 추가로 보호하는 SELinux 정책을 구축할 수 있습니다. SELinux는 사용자, 정책 및 보안 컨텍스트를 결합하여 애플리케이션을 실행하는 데 필요한 리소스를 구분하고 필요하지 않은 다른 시스템 리소스에서 분할합니다.



참고

Red Hat OpenStack Platform에는 OpenStack 서비스에 맞게 사용자 지정된 정책과 함께 SELinux가 기본적으로 활성화되어 있습니다. 필요에 따라 정기적으로 이러한 정책을 검토하는 것이 좋습니다.

21.10. 보안 그룹

OpenStack은 지정된 프로젝트의 인스턴스에 보안 그룹을 추가하여 호스트와 네트워크 모두에 보안 그룹을 제공합니다. 포트, 프로토콜 및 주소를 기반으로 들어오는 트래픽을 허용하거나 거부하는 호스트 기반 방화벽과 유사합니다. 그러나 보안 그룹 규칙은 들어오는 트래픽에만 적용되지만 호스트 기반 방화벽 규칙은 들어오고 나가는 트래픽 모두에 적용할 수 있습니다. 호스트 및 네트워크 보안 그룹 규칙이 합법적인 트래픽을 충돌하고 거부할 수도 있습니다. 사용 중인 네트워킹에 대해 보안 그룹이 올바르게 구성되었는지 확인하는 것이 좋습니다. 자세한 내용은 이 가이드의 보안 그룹을 참조하십시오.



참고

특히 비활성화해야 하는 경우를 제외하고 보안 그룹 및 포트 보안이 활성화되어 있어야 합니다. 침입 방어 접근 방식을 기반으로 구축하려면 인스턴스에 세분화된 규칙을 적용하는 것이 좋습니다.

21.11. 인스턴스 콘솔에 액세스

기본적으로 인스턴스의 콘솔은 가상 콘솔을 통해 원격으로 액세스할 수 있습니다. 이 기능은 문제 해결에 유용할 수 있습니다. Red Hat OpenStack Platform은 원격 콘솔 액세스에 VNC를 사용합니다.

- 방화벽 규칙을 사용하여 VNC 포트를 잠그는 것이 좋습니다. 기본적으로 **nova_vnc_proxy** 는 **6080** 및 **ctrlplane80** 을 사용합니다.
- VNC 트래픽이 TLS로 암호화되었는지 확인합니다. director 기반 배포의 경우 **UseTLSTransportForVnc** 로 시작합니다.

21.12. 인증서 삽입

인스턴스에 SSH를 실행해야 하는 경우 생성 시 필요한 SSH 키를 인스턴스에 자동으로 삽입하도록 Compute를 구성할 수 있습니다.

추가 리소스

- [Red Hat OpenStack Platform에서 KVM 게스트 이미지 사용](#)

22장. 메시지 대기열

메시지 대기열 서비스는 OpenStack에서 프로세스 간 통신을 용이하게 합니다. 이 작업은 다음과 같은 메시지 큐 서비스 백엔드를 사용하여 수행됩니다.

- RabbitMQ - Red Hat OpenStack Platform은 기본적으로 RabbitMQ를 사용합니다.
- Qpid

RabbitMQ와 Cryostat는 모두 피어 간 통신에 메시지 큐를 제공하는 AMQP(Advanced Message Queuing Protocol) 프레임워크입니다. 큐 구현은 일반적으로 중앙 집중식 또는 분산 대기열 서버 풀로 배포됩니다.

메시지 대기열을 사용하면 OpenStack 배포 전반에 걸쳐 명령 및 제어 기능을 효과적으로 수행할 수 있습니다. 큐에 대한 액세스가 허용되면 추가 권한 부여 검사가 수행되지 않습니다. 큐를 통해 액세스할 수 있는 서비스는 실제 메시지 페이로드 내에서 컨텍스트와 토큰의 유효성을 검사합니다. 그러나 토큰이 잠재적으로 다시 재생되고 인프라에서 다른 서비스를 승인할 수 있으므로 토큰 만료일을 기록해야 합니다.

OpenStack에서는 메시지 서명과 같은 메시지 수준 신뢰를 지원하지 않습니다. 따라서 메시지 전송 자체를 보호하고 인증해야 합니다. HA(고가용성) 구성의 경우 대기열 간 인증 및 암호화를 수행해야 합니다.

22.1. 메시징 전송 보안

AMQP 기반 솔루션(Qpid 및 RabbitMQ)은 TLS를 사용하여 전송 수준 보안을 지원합니다.

메시지 큐에 대해 전송 수준 암호화를 활성화하는 것이 좋습니다. 메시징 클라이언트 연결에 TLS를 사용하면 변조 및 전환에서 메시징 서버로의 통신을 보호할 수 있습니다. 다음은 널리 사용되는 두 개의 메시징 서버에 대해 TLS를 구성하는 방법에 대한 지침이 포함되어 있습니다: Cryostat 및 RabbitMQ. 메시징 서버가 클라이언트 연결을 확인하는 데 사용하는 신뢰할 수 있는 CA(인증 기관) 번들을 구성할 때 노드에 사용되는 CA(인증 기관)만 사용하는 것이 좋습니다. 신뢰할 수 있는 CA의 번들이 인증될 클라이언트 인증서를 결정하고 TLS 연결 설정의 클라이언트-서버 확인 단계를 전달합니다.



참고

인증서 및 키 파일을 설치할 때 파일 권한(예: **chmod 0600** 사용)이 제한되고 메시징 서버의 다른 프로세스 및 사용자가 무단 액세스를 방지하기 위해 메시징 서버 데몬 사용자로 소유권이 제한되어 있는지 확인합니다.

22.1.1. RabbitMQ 서버 SSL 구성

시스템 전체 RabbitMQ 구성 파일(일반적으로 `/etc/rabbitmq/rabbitmq.config`):에 다음 행을 추가해야 합니다.

```
[
  {rabbit, [
    {tcp_listeners, []},
    {ssl_listeners, [{"<IP address or hostname of management network interface>", 5671}]},
    {ssl_options, [{cacertfile, "/etc/ssl/cacert.pem"},
                   {certfile, "/etc/ssl/rabbit-server-cert.pem"},
                   {keyfile, "/etc/ssl/rabbit-server-key.pem"},
                   {verify, verify_peer},
                   {fail_if_no_peer_cert, true}]}
  ]}
].
```



참고

SSL이 아닌 포트에서 수신 대기하지 못하도록 **tcp_listeners** 옵션이 **[]** 로 설정됩니다. **ssl_listeners** 옵션은 서비스의 관리 네트워크에서만 수신 대기하도록 제한해야 합니다.

22.2. 대기열 인증 및 액세스 제어

RabbitMQ 및 Cryostat는 큐에 대한 액세스를 제어하기 위한 인증 및 액세스 제어 메커니즘을 제공합니다.

SASL(Simple Authentication and Security Layer)은 인터넷 프로토콜의 인증 및 데이터 보안을 위한 프레임워크입니다. RabbitMQ와 Cryostat 모두 단순한 사용자 이름과 암호 이외의 SASL 및 기타 플러그형 인증 메커니즘을 제공하여 인증 보안을 강화할 수 있습니다. RabbitMQ는 SASL을 지원하지만 OpenStack의 지원은 현재 특정 SASL 인증 메커니즘을 요청할 수 없습니다. OpenStack의 RabbitMQ 지원은 X.509 클라이언트 인증서와 함께 암호화되지 않은 연결 또는 사용자 이름 및 암호를 통해 사용자 이름과 암호 인증을 사용하여 보안 TLS 연결을 설정할 수 있습니다.

메시징 큐에 대한 클라이언트 연결을 위해 모든 OpenStack 서비스 노드에 X.509 클라이언트 인증서를 구성하는 것이 좋습니다. 가능한 경우 (현재만) X.509 클라이언트 인증서로 인증을 수행하는 것이 좋습니다. 사용자 이름과 암호를 사용하는 경우 큐에 대한 액세스 권한을 세밀하게 감사할 수 있도록 서비스 및 노드 별로 계정을 생성해야 합니다.

배포하기 전에 대기열 서버가 사용하는 TLS 라이브러리를 고려하십시오. Cryostat는 Mozilla의 NSS 라이브러리를 사용하지만 RabbitMQ는 OpenSSL을 사용하는 Erlang의 TLS 모듈을 사용합니다.

22.3. RABBITMQ의 OPENSTACK 서비스 구성

이 섹션에서는 OpenStack 서비스에 대한 일반적인 RabbitMQ 구성에 대해 설명합니다.

```
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_kombu
rabbit_use_ssl = True
rabbit_host = RABBIT_HOST
rabbit_port = 5671
rabbit_user = compute01
rabbit_password = RABBIT_PASS
kombu_ssl_keyfile = /etc/ssl/node-key.pem
kombu_ssl_certfile = /etc/ssl/node-cert.pem
kombu_ssl_ca_certs = /etc/ssl/cacert.pem
```



참고

RA **CryostatIT_PASS** 를 적절한 암호로 바꿉니다.

22.4. CRYOSTAT의 OPENSTACK 서비스 구성

이 섹션에서는 OpenStack 서비스에 대한 일반적인 Cryostat 구성에 대해 설명합니다.

```
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_qpid
qpid_protocol = ssl
qpid_hostname = <IP or hostname of management network interface of messaging server>
```

```
qpid_port = 5671
qpid_username = compute01
qpid_password = QPID_PASS
```



참고

QPID_PASS 를 적절한 암호로 바꿉니다.

선택적으로 Cryostat와 함께 SASL을 사용하는 경우 다음을 추가하여 사용 중인 SASL 메커니즘을 지정합니다.

```
qpid_sasl_mechanisms = <space separated list of SASL mechanisms to use for auth>
```

22.5. 메시지 큐 프로세스 격리 및 정책

각 프로젝트는 메시지를 보내고 사용하는 여러 서비스를 제공합니다. 메시지를 전송하는 각 바이너리는 대기열에서 응답만 하는 메시지를 사용해야 합니다.

메시지 큐 서비스 프로세스는 시스템의 서로 다른 프로세스와 분리되어야 합니다.

22.6. 네임스페이스

Linux는 네임스페이스를 사용하여 프로세스를 독립 도메인에 할당합니다. 이 가이드의 다른 부분에서는 시스템 구분을 자세히 다룹니다.

23장. RED HAT OPENSTACK PLATFORM에서 끝점 보안

OpenStack 클라우드에 참여하는 프로세스는 API 엔드포인트를 쿼리하여 시작합니다. 퍼블릭 및 프라이빗 엔드포인트에는 다른 문제가 있지만 손상된 경우 상당한 위험을 초래할 수 있는 높은 가치 자산입니다.

이 장에서는 공용 및 개인 API 끝점 모두에 대해 보안 개선 사항을 권장합니다.

23.1. 내부 API 통신

OpenStack에서는 공용, 내부 관리자 및 프라이빗 API 엔드포인트를 모두 제공합니다. 기본적으로 OpenStack 구성 요소는 공개적으로 정의된 엔드포인트를 사용합니다. 적절한 보안 도메인 내에서 API 끝점을 사용하도록 이러한 구성 요소를 구성하는 것이 좋습니다. 내부 관리자 엔드포인트는 keystone에 대한 추가 액세스를 허용하므로 이를 추가로 격리하는 것이 바람직할 수 있습니다.

서비스는 OpenStack 서비스 카탈로그를 기반으로 각 API 끝점을 선택합니다. 이러한 서비스는 나열된 공용 또는 내부 API 엔드포인트 값을 따르지 않을 수 있습니다. 이로 인해 내부 관리 트래픽이 외부 API 엔드포인트로 라우팅될 수 있습니다.

23.2. ID 서비스 카탈로그에서 내부 URL 구성

ID 서비스 카탈로그는 내부 URL을 알고 있어야 합니다. 이 기능은 기본적으로 사용되지 않지만 구성을 통해 사용할 수 있습니다. 또한 이 동작이 기본값이 되면 예상되는 변경 사항과 순방향 호환되어야 합니다.

구성된 엔드포인트를 서로 다른 수준의 액세스 권한이 있는 경우 네트워크 수준에서 격리하는 것이 좋습니다. 관리자 엔드포인트는 내부 또는 공용 엔드포인트에서 사용할 수 없는 keystone 작업에 대한 높은 액세스를 제공하므로 클라우드 관리자가 액세스하기 위한 것입니다. 내부 엔드포인트는 클라우드 내부(예: OpenStack 서비스)를 사용하기 위한 것이며, 일반적으로 배포 네트워크 외부에서 액세스할 수 없습니다. 공용 끝점은 TLS를 사용할 수 있어야 하며 클라우드 사용자가 작동할 수 있도록 배포 외부에서 액세스할 수 있는 유일한 API 끝점입니다.

끝점의 내부 URL 등록은 director에 의해 자동화됩니다. 자세한 내용은

[https://github.com/openstack/tripleo-heat-](https://github.com/openstack/tripleo-heat-templates/blob/a7857d6dfcc875eb2bc611dd9334104c18fe8ac6/network/endpoints/build_endpoint_map)

[templates/blob/a7857d6dfcc875eb2bc611dd9334104c18fe8ac6/network/endpoints/build_endpoint_map](https://github.com/openstack/tripleo-heat-templates/blob/a7857d6dfcc875eb2bc611dd9334104c18fe8ac6/network/endpoints/build_endpoint_map)을 참조하십시오.

23.3. 내부 URL에 대한 애플리케이션 구성

일부 서비스가 특정 API 끝점을 사용하도록 강제 적용할 수 있습니다. 따라서 다른 서비스의 API에 연결하는 모든 OpenStack 서비스를 적절한 내부 API 엔드포인트에 액세스하도록 명시적으로 구성해야 합니다.

각 프로젝트는 대상 API 엔드포인트를 정의하는 일관되지 않은 방법을 제공할 수 있습니다. 향후 OpenStack 릴리스에서는 ID 서비스 카탈로그를 일관되게 사용하여 이러한 불일치를 해결하려고 합니다.

23.4. 붙여넣기 및 미들웨어

OpenStack의 대부분의 API 끝점 및 기타 HTTP 서비스는 Python Paste Deploy 라이브러리를 사용합니다. 이 라이브러리는 보안 관점에서 애플리케이션의 구성을 통해 요청 필터 파이프라인을 조작할 수 있습니다. 이 체인의 각 요소를 미들웨어라고 합니다. 파이프라인의 필터 순서를 변경하거나 추가 미들웨어를 추가하면 보안에 영향을 줄 수 없습니다.

구현자는 일반적으로 OpenStack의 기본 기능을 확장하기 위해 미들웨어를 추가합니다. 비표준 소프트웨어 구성 요소를 HTTP 요청 파이프라인에 추가하여 도입된 잠재적 노출을 신중하게 고려하는 것이 좋습니다.

23.5. 보안 METADEF API

RHOSP(Red Hat OpenStack Platform)에서 클라우드 관리자는 메타데이터 정의(metadef) API를 사용하여 키 값 쌍과 태그 메타데이터를 정의할 수 있습니다. 클라우드 관리자가 생성할 수 있는 metadef 네임스페이스, 오브젝트, 속성, 리소스 또는 태그 수에 제한이 없습니다.

이미지 서비스 정책은 metadef API를 제어합니다. 기본적으로 클라우드 관리자만 metadef API를 생성, 업데이트 또는 삭제할 수 있습니다. 이러한 제한은 metadef API가 무단 사용자에게 정보를 노출하지 못하도록 하며, 이미지 서비스(glance) 데이터베이스를 무제한 리소스로 채우는 악의적인 사용자의 위험을 완화하여 DoS(Denial of Service) 스타일 공격을 생성할 수 있습니다. 그러나 클라우드 관리자는 기본 정책을 재정의할 수 있습니다.

23.6. 클라우드 사용자를 위한 METADEF API 액세스 활성화

메타데이터 정의(metadef) API에 대한 쓰기 액세스 권한을 사용하는 클라우드 관리자는 기본 관리자 전용 정책을 재정의하여 모든 사용자가 해당 API에 액세스할 수 있도록 할 수 있습니다. 그러나 이러한 유형의 구성에서는 고객 이름 및 내부 프로젝트와 같은 실수로 민감한 리소스 이름을 누출시킬 가능성이 있습니다. 관리자는 모든 사용자에게 읽기 액세스만 활성화된 경우에도 이전에 생성된 리소스를 식별하기 위해 시스템을 감사해야 합니다.

절차

1. 클라우드 관리자로 언더클라우드에 로그인하여 정책 재정의에 대한 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
$ cat open-up-glance-api-metadef.yaml
```

2. 모든 사용자에게 metadef API 읽기-쓰기 액세스를 허용하도록 정책 덮어쓰기 파일을 구성합니다.

```
GlanceApiPolicies: {
  glance-metadef_default: { key: 'metadef_default', value: " },
  glance-get_metadef_namespace: { key: 'get_metadef_namespace', value:
'rule:metadef_default' },
  glance-get_metadef_namespaces: { key: 'get_metadef_namespaces', value:
'rule:metadef_default' },
  glance-modify_metadef_namespace: { key: 'modify_metadef_namespace', value:
'rule:metadef_default' },
  glance-add_metadef_namespace: { key: 'add_metadef_namespace', value:
'rule:metadef_default' },
  glance-delete_metadef_namespace: { key: 'delete_metadef_namespace', value:
'rule:metadef_default' },
  glance-get_metadef_object: { key: 'get_metadef_object', value: 'rule:metadef_default' },
  glance-get_metadef_objects: { key: 'get_metadef_objects', value: 'rule:metadef_default' },
  glance-modify_metadef_object: { key: 'modify_metadef_object', value:
'rule:metadef_default' },
  glance-add_metadef_object: { key: 'add_metadef_object', value: 'rule:metadef_default' },
  glance-delete_metadef_object: { key: 'delete_metadef_object', value: 'rule:metadef_default'
},
  glance-list_metadef_resource_types: { key: 'list_metadef_resource_types', value:
'rule:metadef_default' },
  glance-get_metadef_resource_type: { key: 'get_metadef_resource_type', value:
'rule:metadef_default' },
  glance-add_metadef_resource_type_association: { key:
'add_metadef_resource_type_association', value: 'rule:metadef_default' },
  glance-remove_metadef_resource_type_association: { key:
```

```
'remove_metadef_resource_type_association', value: 'rule:metadef_default' },
  glance-get_metadef_property: { key: 'get_metadef_property', value: 'rule:metadef_default'
},
  glance-get_metadef_properties: { key: 'get_metadef_properties', value:
'rule:metadef_default' },
  glance-modify_metadef_property: { key: 'modify_metadef_property', value:
'rule:metadef_default' },
  glance-add_metadef_property: { key: 'add_metadef_property', value: 'rule:metadef_default'
},
  glance-remove_metadef_property: { key: 'remove_metadef_property', value:
'rule:metadef_default' },
  glance-get_metadef_tag: { key: 'get_metadef_tag', value: 'rule:metadef_default' },
  glance-get_metadef_tags: { key: 'get_metadef_tags', value: 'rule:metadef_default' },
  glance-modify_metadef_tag: { key: 'modify_metadef_tag', value: 'rule:metadef_default' },
  glance-add_metadef_tag: { key: 'add_metadef_tag', value: 'rule:metadef_default' },
  glance-add_metadef_tags: { key: 'add_metadef_tags', value: 'rule:metadef_default' },
  glance-delete_metadef_tag: { key: 'delete_metadef_tag', value: 'rule:metadef_default' },
  glance-delete_metadef_tags: { key: 'delete_metadef_tags', value: 'rule:metadef_default' }
}
```



참고

rule:metadef_default 를 사용하도록 모든 metadef 정책을 구성해야 합니다. 정책 및 정책 구문에 대한 자세한 내용은 이 [정책](#) 장을 참조하십시오.

- 오버클라우드를 배포할 때 **-e** 옵션을 사용하여 배포 명령에 새 정책 파일을 추가합니다.

```
$ openstack overcloud deploy -e open-up-glance-api-metadef.yaml
```

23.7. HAPROXY에 대한 SSL/TLS 암호화 및 규칙 변경

오버클라우드에서 SSL/TLS를 활성화한 경우 HAProxy 구성에 사용되는 SSL/TLS 암호화 방식 및 규칙을 강화하는 것이 좋습니다. SSL/TLS 암호를 강화하면 [POODLE](#) 취약점과 같은 SSL/TLS 취약점을 방지할 수 있습니다.

- tls-ciphers.yaml** 이라는 heat 템플릿 환경 파일을 생성합니다.

```
touch ~/templates/tls-ciphers.yaml
```

- 환경 파일에서 **ExtraConfig** 후크를 사용하여 **tripleo::haproxy::ssl_cipher_suite** 및 **tripleo::haproxy::ssl_options** hieradata에 값을 적용합니다.

```
parameter_defaults:
  ExtraConfig:
    tripleo::haproxy::ssl_cipher_suite: `DHE-RSA-AES128-CCM:DHE-RSA-AES256-
CCM:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-
AES128-CCM:ECDHE-ECDSA-AES256-CCM:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305`

    tripleo::haproxy::ssl_options: no-sslv3 no-tls-tickets
```



참고

암호화 방식 컬렉션은 연속된 한 행으로 되어 있습니다.

3. 오버클라우드를 배포할 때 `overcloud deploy` 명령을 사용하여 **tls-ciphers.yaml** 환경 파일을 포함합니다.

```
openstack overcloud deploy --templates \
...
-e /home/stack/templates/tls-ciphers.yaml
...
```

23.8. 네트워크 정책

API 끝점은 일반적으로 여러 보안 영역에 걸쳐 있으므로 API 프로세스 분리에 특히 주의해야 합니다. 예를 들어 네트워크 설계 수준에서는 지정된 시스템에 대한 액세스만 제한하는 것을 고려할 수 있습니다. 자세한 내용은 보안 영역에 대한 지침을 참조하십시오.

신중하게 모델링하면 네트워크 ACL 및 IDS 기술을 사용하여 네트워크 서비스 간에 명확한 지점 간 통신을 적용할 수 있습니다. 이러한 유형의 명시적 시행은 중요한 도메인 서비스로서 OpenStack의 메시지 큐 서비스에 적합합니다.

정책을 적용하려면 서비스, 호스트 기반 방화벽(예: iptables), 로컬 정책(SELinux) 및 선택적으로 글로벌 네트워크 정책을 구성할 수 있습니다.

23.9. 필수 액세스 제어

API 끝점 프로세스를 시스템의 서로 및 기타 프로세스로부터 격리해야 합니다. DAC(Discutary Access Controls) 및 MAC(Mandatory Access Controls)을 통해 이러한 프로세스에 대한 구성을 제한해야 합니다. 이러한 향상된 액세스 제어의 목표는 API 엔드포인트 보안 위반을 지원하는 것입니다.

23.10. API 끝점 속도 제한

속도 제한은 네트워크 기반 애플리케이션에서 수신한 이벤트의 빈도를 제어하는 수단입니다. 강력한 속도 제한이 없으면 애플리케이션이 다양한 서비스 거부 공격에 취약해질 수 있습니다. 이는 특히 유사한 요청 유형 및 작업의 높은 빈도를 허용하도록 설계된 API에 특히 해당됩니다.

모든 끝점(특히 공개)은 물리적 네트워크 설계, 속도 제한 프록시 또는 웹 애플리케이션 방화벽을 사용하여 추가 보호 계층을 제공하는 것이 좋습니다.

운영자가 속도 제한 기능을 구성하고 구현할 때 사용자와 서비스의 개별 성능 요구 사항을 신중하게 계획하는 것이 중요합니다.



참고

Red Hat OpenStack Platform 배포의 경우 모든 서비스는 로드 밸런싱 프록시 뒤에 배치됩니다.

24장. 페더레이션 구현

Red Hat은 RHOSP(Red Hat OpenStack Platform)와 함께 Red Hat Single Sign-on(RH-SSO) 또는 Microsoft AD FS(Microsoft Active Directory Federation Services)를 사용한 페더레이션을 지원합니다.

24.1. RED HAT SINGLE SIGN-ON을 사용하여 IDM과 통합

RH-SSO(Red Hat Single Sign-On)를 사용하여 IdM 사용자를 OpenStack 인증(authN)을 통합할 수 있습니다. 페더레이션을 통해 IdM 사용자는 OpenStack 서비스에 인증 정보를 표시하지 않고 OpenStack 대시보드에 로그인할 수 있습니다. 대신 대시보드에 사용자의 인증 정보가 필요한 경우 사용자를 RH-SSO로 전달하여 IdM 자격 증명을 입력할 수 있습니다. 결과적으로 RH-SSO는 사용자가 성공적으로 인증한 대시보드로 다시 어설션하고 Dashboard를 통해 사용자가 프로젝트에 액세스할 수 있습니다.

24.2. 페더레이션 워크플로

이 섹션에서는 ID 서비스(keystone), RH-SSO 및 IdM이 서로 상호 작용하는 방법을 설명합니다. OpenStack의 Federation은 ID 공급자 및 서비스 공급자의 개념을 사용합니다.

IdM(ID 공급자) - 사용자 계정을 저장하는 서비스입니다. 이 경우 IdM에 보관된 사용자 계정은 RH-SSO를 사용하여 Keystone에 제공됩니다.

서비스 공급자 (SP) - IdP의 사용자의 인증이 필요한 서비스입니다. 이 경우 keystone은 IdM 사용자에게 대시보드 액세스 권한을 부여하는 서비스 공급자입니다.

다른 IdP의 범용 어댑터 역할을 할 수 있는 RH-SSO(IdP)와 통신하도록 ID 서비스(SP)를 구성할 수 있습니다. 이 구성에서는 RH-SSO에서 keystone을 가리킬 수 있으며 RH-SSO는 현재 IdM 및 Active Directory를 지원하는 ID 공급자에게 요청을 전달합니다. 이 작업은 서비스 공급자(SP) 및 IdM(Identity Provider) 교환 메타데이터를 사용하면 각 sysadmin이 신뢰 결정을 내립니다. 결과적으로 IdP는 어설션을 안정적으로 만들 수 있으며 SP는 이러한 어설션을 수신할 수 있습니다.

추가 리소스

- [Red Hat OpenStack Platform 및 Red Hat Single Sign-On을 사용한 페더레이션](#)