Red Hat Enterprise Linux 9

Amazon Web Services에 RHEL 9 배포

AWS에서 RHEL 시스템 이미지 가져오기 및 RHEL 인스턴스 생성

Last Updated: 2024-07-19

Red Hat Enterprise Linux 9 Amazon Web Services에 RHEL 9 배포

AWS에서 RHEL 시스템 이미지 가져오기 및 RHEL 인스턴스 생성

법적 공지

초록

퍼블릭 클라우드 환경에서 RHEL(Red Hat Enterprise Linux)을 사용하려면 AWS(Amazon Web Services)를 비롯한 다양한 클라우드 플랫폼에 RHEL 시스템 이미지를 생성하고 배포할 수 있습니다. AWS에서 HA(Red Hat High Availability) 클러스터를 생성하고 구성할 수도 있습니다. 다음 장에서는 AWS에서 클라우드 RHEL 인스턴스 및 HA 클러스터를 생성하는 방법을 설명합니다. 이러한 프로세스에는 필수 패키지 및 에이전트 설치, 펜싱 구성 및 네트워크 리소스 에이전트 설치가 포함됩니다.

차례

RHEL 베타 버전 릴리스

Red Hat은 Red Hat Enterprise Linux 베타 버전을 통해 등록된 모든 Red Hat 계정에 액세스할 수 있습니다. 베타 버전 액세스의 목적은 다음과 같습니다.

- 고객에게 일반 가용성 릴리스 전에 주요 기능을 테스트하고 피드백을 제공하거나 문제를 보고할 수 있는 기회를 제공합니다.
- 베타 제품 설명서를 미리 보기로 제공합니다. 베타 제품 문서는 개발 중이며 상당한 변경이 있을 수 있습니다.

Red Hat은 프로덕션 사용 사례에서는 RHEL 베타 버전 릴리스를 지원하지 않습니다. 자세한 내용은 Red Hat Enterprise Linux에서 베타 버전의 의미는 무엇이며 RHEL 베타 설치를 GA(General Availability) 릴리스로 업그레이드할 수 있습니까?에서 참조하십시오.

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 의견에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

특정 문구에 대한 의견 제출

1. **Multi-page HTML** 형식으로 설명서를 보고 페이지가 완전히 로드된 후 오른쪽 상단 모서리에 **피드백** 버튼이 표시되는지 확인합니다.
2. 커서를 사용하여 주석 처리할 텍스트 부분을 강조 표시합니다.
3. 강조 표시된 텍스트 옆에 표시되는 **피드백 추가** 버튼을 클릭합니다.
4. 의견을 추가하고 **제출**을 클릭합니다.

Bugzilla를 통해 피드백 제출(등록 필요)

1. Bugzilla 웹 사이트에 로그인합니다.
2. **버전** 메뉴에서 올바른 버전을 선택합니다.
3. **요약** 필드에 설명 제목을 입력합니다.
4. **설명** 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. **버그 제출**을 클릭합니다.

1장. 베타의 새로운 기능

보다 집중된 정보 및 개선된 사용자 환경을 제공하기 위해 공용 클라우드 플랫폼에 RHEL 9 배포 문서는 특정 클라우드 환경에 배포하는 데 전하는 여러 개의 문서로 나뉘어져 있습니다.

이 변화가 좋은지 여부에 대한 귀하의 의견을 환영합니다. 자세한 내용은 Red Hat 문서에 대한 피드백 제공 을 참조하십시오.

2장. AWS AMI 이미지 생성 및 업로드

AWS(Amazon Web Services) 클라우드에서 사용자 지정 RHEL 시스템 이미지를 사용하려면 해당 출력 유형을 사용하여 Image Builder로 시스템 이미지를 생성하고, 이미지를 업로드하도록 시스템을 구성하고, AWS 계정에 이미지를 업로드합니다.

2.1. AWS AMI 이미지 업로드 준비

AWS AMI 이미지를 업로드하기 전에 이미지를 업로드할 시스템을 구성해야 합니다.

사전 요구 사항

- AWS IAM 계정 관리자에 Access Key ID가 구성되어 있어야 합니다.
- 쓰기 가능한 S3 버킷이 준비되어 있어야 합니다.

절차

1. Python 3 및 **pip** 툴을 설치합니다.

```
# dnf install python3
# dnf install python3-pip
```

2. **pip** 를 사용하여 AWS 명령행 툴 을 설치합니다.

```
# pip3 install awscli
```

3. 다음 명령을 실행하여 프로필을 설정합니다. 터미널에서 인증 정보, 지역 및 출력 형식을 제공하라는 메시지를 표시합니다.

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

4. 버킷 이름을 정의하고 다음 명령을 사용하여 버킷을 생성합니다.

```
$ BUCKET=bucketname
$ aws s3 mb s3://$BUCKET
```

bucketname 을 실제 버킷 이름으로 교체합니다. 이는 전역적으로 고유한 이름이어야 합니다. 결과적으로 버킷이 생성됩니다.

5. S3 버킷에 액세스할 수 있는 권한을 부여하려면 이전에 수행하지 않은 경우 AWS IAM(Identity and Access Management)에서 **vmimport** S3 역할을 생성합니다.

```
$ printf '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service":
"vmie.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition": { "StringEquals":{
"sts:Externalid": "vmimport" } } } ] }' > trust-policy.json
$ printf '{ "Version":"2012-10-17", "Statement":[ { "Effect":"Allow", "Action":[
"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket" ], "Resource":[ "arn:aws:s3:::%s",
"arn:aws:s3:::%s/*" ] }, { "Effect":"Allow", "Action":[ "ec2:ModifySnapshotAttribute",
```

```
"ec2:CopySnapshot", "ec2:RegisterImage", "ec2:Describe*" ], "Resource":"*" } ] }' $BUCKET
$BUCKET > role-policy.json
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-
policy.json
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file://role-policy.json
```

추가 리소스

- AWS CLI에서 상위 수준(s3) 명령 사용

2.2. CLI를 사용하여 AWS에 AMI 이미지 업로드

CLI를 사용하여 이미지 빌더를 사용하여 **ami** 이미지를 빌드하고, CLI를 사용하여 Amazon AWS Cloud 서비스 공급자로 직접 푸시할 수 있습니다.

사전 요구 사항

- AWS IAM 계정 관리자에 **Access Key ID** 가 구성되어 있습니다.
- 쓰기 가능한 S3 버킷이 준비되어 있습니다.
- 정의된 청사진이 있습니다.

절차

1. 텍스트 편집기를 사용하여 다음 내용이 포함된 구성 파일을 생성합니다.

```
provider = "aws"

[settings]
accessKeyID = "AWS_ACCESS_KEY_ID"
secretAccessKey = "AWS_SECRET_ACCESS_KEY"
bucket = "AWS_BUCKET"
region = "AWS_REGION"
key = "IMAGE_KEY"
```

   필드의 값을 **accessKeyID**,**secretAccessKey**,**버킷**, **리전** 의 자격 증명으로 바꿉니다. **IMAGE_KEY** 값은 EC2에 업로드할 VM 이미지의 이름입니다.

2. 파일을 *CONFIGURATION-FILE*.toml로 저장하고 텍스트 편집기를 종료합니다.

3. 작성을 시작합니다.

```
# composer-cli compose start BLUEPRINT-NAME IMAGE-TYPE IMAGE_KEY
CONFIGURATION-FILE.toml
```

   교체:

   - *BLUEPRINT-NAME* (으)로 생성한 사용자 이름
   - **ami** 이미지 유형의 *IMAGE-TYPE*.
   - EC2에 업로드할 VM 이미지의 이름으로 *IMAGE_KEY*.

- 클라우드 공급자의 구성 파일 이름으로 *CONFIGURATION-FILE*.toml

참고

사용자 지정 이미지를 보낼 버킷에 대한 올바른 IAM 설정이 있어야 합니다. 이미지를 업로드하려면 먼저 버킷에 정책을 설정해야 합니다.

4. 이미지 빌드 상태를 확인하고 AWS에 업로드합니다.

```
# composer-cli compose status
```

이미지 업로드 프로세스가 완료되면 "FINISHED" 상태가 표시됩니다.

검증

이미지 업로드에 성공했는지 확인하려면 다음을 수행합니다.

1. 메뉴에서 EC2 에 액세스하고 AWS 콘솔에서 올바른 리전을 선택합니다. 이미지가 성공적으로 업로드되었음을 나타내기 위해 **사용 가능한** 상태가 있어야 합니다.
2. 대시보드에서 이미지를 선택하고 **시작을** 클릭합니다.

추가 리소스

- VM을 가져오는 데 필요한 서비스 역할

2.3. AWS CLOUD AMI로 이미지 푸시

생성한 출력 이미지를 **Amazon AWS Cloud AMI**서비스 공급자로 직접 푸시할 수 있습니다.

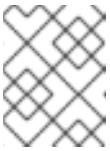
사전 요구 사항

- **root** 또는 git group **사용자가** 시스템에 액세스할 수 있어야 합니다.
- 브라우저에서 RHEL 웹 콘솔의 이미지 빌더 인터페이스를 열었습니다.
- 생성했습니다. 웹 콘솔 인터페이스에서 이미지 빌더 청사진 생성을 참조하십시오.
- AWS IAM 계정 관리자에 Access Key ID가 구성되어 있어야 합니다.
- 쓰기 가능한 S3 버킷이 준비되어 있어야 합니다.

절차

1. Blueprint **이름을** 클릭합니다.
2. **Images** (이미지) 탭을 선택합니다.
3. **Create Image** (이미지 만들기)를 클릭하여 사용자 지정된 이미지를 생성합니다.
팝업 창이 열립니다.
   a. **유형** 드롭다운 메뉴 목록에서 **Amazon Machine Image Disk(.raw)** 를 선택합니다.
   b. **AWS에 업로드** 확인란을 선택하여 이미지를 AWS 클라우드에 업로드하고 **다음을** 클릭합니다.

c. AWS에 대한 액세스를 인증하려면 해당 필드에 **AWS 액세스 키 ID** 와 **AWS 시크릿 액세스 키** 를 입력합니다. **다음**을 클릭합니다.

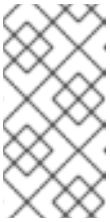


참고

새 액세스 키 ID를 생성할 때만 AWS 시크릿 액세스 키를 볼 수 있습니다. Secret Key를 모르는 경우 새 Access Key ID를 생성합니다.

d. 이미지 이름 필드에 **이미지 이름을** 입력하고 Amazon **S3 버킷 이름 필드에 Amazon 버킷 이름을** 입력하고 사용자 지정 이미지를 추가할 버킷의 **AWS 리전** 필드를 입력합니다. **다음**을 클릭합니다.

e. 정보를 검토하고 **마침** 을 클릭합니다.
필요한 경우 **뒤로** 를 클릭하여 잘못된 세부 정보를 수정할 수 있습니다.



참고

사용자 지정 이미지를 보낼 버킷에 대한 올바른 IAM 설정이 있어야 합니다. 이 절차에서는 IAM 가져오기 및 내보내기를 사용하므로 이미지를 업로드하기 전에 버킷에 **정책을** 설정해야 합니다. 자세한 내용은 IAM 사용자에게 필요한 권한을 참조하십시오.

4. 오른쪽 상단에 있는 작은 팝업이 저장 진행 상황을 알려줍니다. 또한 이미지 생성이 시작되었으며 이 이미지 생성이 진행 중이고 AWS Cloud에 후속 업로드가 있음을 알립니다.
프로세스가 완료되면 **이미지 빌드 완료** 상태를 확인할 수 있습니다.

5. 메뉴에서 ServiceECDHEEC2 를 클릭하고 AWS 콘솔에서 올바른 리전 을 선택합니다. 이미지에는 업로드되었음을 나타내기 위해 **Available** 상태가 있어야 합니다.

6. 대시보드에서 이미지를 선택하고 **시작을** 클릭합니다.

7. 새 창이 열립니다. 이미지를 시작하는 데 필요한 리소스에 따라 인스턴스 유형을 선택합니다. **검토 및 시작을** 클릭합니다.

8. 인스턴스 시작 세부 정보를 확인합니다. 변경할 필요가 있는 경우 각 섹션을 편집할 수 있습니다. **시작을 클릭합니다.**

9. 인스턴스를 시작하기 전에 공개 키를 선택하여 액세스할 수 있습니다.
이미 보유하고 있는 키 쌍을 사용하거나 새 키 쌍을 만들 수 있습니다. 또는 **이미지** 빌더를 사용하여 사전 설정된 공개 키가 있는 이미지에 사용자를 추가할 수 있습니다. 자세한 내용은 SSH 키를 사용하여 사용자 계정 생성을 참조하십시오.

   다음 단계에 따라 EC2에서 새 키 쌍을 생성하고 새 인스턴스에 연결합니다.

   a. 드롭다운 메뉴 목록에서 **새 키 쌍 만들기** 를 선택합니다.

   b. 새 키 쌍의 이름을 입력합니다. 새 키 쌍을 생성합니다.

   c. **Download Key Pair** (키 쌍 다운로드)를 클릭하여 로컬 시스템에 새 키 쌍을 저장합니다.

10. 그러면 **인스턴스 시작을 클릭하여** 인스턴스를 시작할 수 있습니다.
Initializing 으로 표시되는 인스턴스의 상태를 확인할 수 있습니다.

11. 인스턴스 상태가 **실행 중** 이면 **연결** 버튼을 사용할 수 있게 됩니다.

12. **연결**을 클릭합니다. SSH를 사용하여 연결하는 방법에 대한 지침이 포함된 팝업 창이 표시됩니다.

    a. 에 권장되 **는 연결 방법으로 독립 실행형 SSH 클라이언트**를선택하고 터미널을 엽니다.

    b. 개인 키를 저장하는 위치에서 SSH가 작동하도록 키를 공개적으로 볼 수 있는지 확인하십시오. 이렇게 하려면 다음 명령을 실행합니다.

    ```
    $ chmod 400 <your-instance-name.pem>_
    ```

    c. 공용 DNS를 사용하여 인스턴스에 연결합니다.

    ```
    $ ssh -i "<_your-instance-name.pem_"> ec2-user@<_your-instance-IP-address_>
    ```

    d. **yes** 를 입력하여 연결을 계속할지 확인합니다.
    결과적으로 SSH를 사용하여 인스턴스에 연결됩니다.

검증

1. SSH를 사용하여 인스턴스에 연결된 동안 모든 작업을 수행할 수 있는지 확인합니다.

추가 리소스

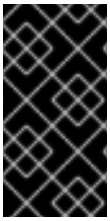
- Red Hat 고객 포털에서 케이스 열기
- SSH를 사용하여 Linux 인스턴스에 연결
- 오픈 지원 사례

3장. RED HAT ENTERPRISE LINUX 이미지를 AMAZON WEB SERVICES에서 EC2 인스턴스로 배포

RHEL(Red Hat Enterprise Linux) 9 이미지를 AWS(Amazon Web Services)에서 EC2 인스턴스로 배포하기 위한 다양한 옵션이 있습니다. 이 장에서는 이미지 및 목록을 선택하는 옵션에 대해 설명하거나 호스트 시스템 및 VM(가상 시스템)의 시스템 요구 사항을 나타냅니다. 이 장에서는 ISO 이미지에서 사용자 지정 VM을 생성하고 EC2에 업로드한 후 EC2 인스턴스를 시작하는 절차를 설명합니다.

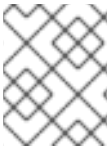
AWS(Amazon Web Services)에서 EC2 인스턴스로 Red Hat Enterprise Linux 9 (RHEL 9)를 배포하려면 다음 정보를 따르십시오. 이 장에서는 다음을 수행합니다.

- 이미지 선택 옵션에 대해 설명합니다.
- 호스트 시스템 및 VM(가상 머신)의 시스템 요구 사항을 나열하거나 참조합니다.
- ISO 이미지에서 사용자 지정 VM을 생성하고 EC2에 업로드 및 EC2 인스턴스를 시작하는 절차를 제공합니다.



중요

ISO 이미지에서 사용자 지정 VM을 생성할 수 있지만 Red Hat Image Builder 제품을 사용하여 특정 클라우드 공급자에서 사용할 사용자 지정 이미지를 생성하는 것이 좋습니다. Image Builder를 사용하면 **ami** 형식으로 AMI(Amazon Machine Image)를 생성하고 업로드할 수 있습니다. 자세한 내용은 사용자 지정된 RHEL 시스템 이미지 구성 을 참조하십시오.



참고

AWS에서 안전하게 사용할 수 있는 Red Hat 제품 목록은 Amazon Web Services에서 Red Hat 을 참조하십시오.

사전 요구 사항

- Red Hat 고객 포털 계정에 등록합니다.
- AWS에 가입하고 AWS 리소스를 설정합니다. 자세한 내용은 Amazon EC2로 설정을 참조하십시오.
- Red Hat Cloud Access 프로그램 에서 서브스크립션을 활성화합니다. Red Hat Cloud Access 프로그램을 사용하면 Red Hat 서브스크립션을 물리적 또는 온프레미스 시스템에서 AWS로 이전할 수 있으며 Red Hat의 완전한 지원을 받을 수 있습니다.

3.1. AWS에서 RED HAT ENTERPRISE LINUX 이미지 옵션

다음 표에는 이미지 선택 사항이 나열되어 이미지 옵션의 차이점을 기록해 둡니다.

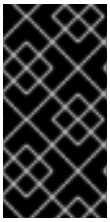
표 3.1. 이미지 옵션

이미지 옵션	서브스크립션	샘플 시나리오	고려 사항

이미지 옵션	서브스크립션	샘플 시나리오	고려 사항
Red Hat Gold Image를 배포하도록 선택합니다.	기존 Red Hat 서브스크립션을 사용합니다.	Red Hat Cloud Access 프로그램을 통해 서브스크립션을 활성화한 다음 AWS에서 Red Hat Gold Image를 선택합니다.	서브스크립션에는 Red Hat 제품 비용이 포함되며, 다른 모든 인스턴스 비용에 대해 Amazon에 비용을 지불합니다. 기존 Red Hat 서브스크립션을 활용하기 때문에 Red Hat Gold Images는 "Cloud Access" 이미지라고 합니다. Red Hat은 Cloud Access 이미지에 대한 직접 지원을 제공합니다.
AWS로 이동하는 사용자 정의 이미지를 배포하도록 선택합니다.	기존 Red Hat 서브스크립션을 사용합니다.	Red Hat Cloud Access 프로그램을 통해 서브스크립션을 활성화하고 사용자 지정 이미지를 업로드하고 서브스크립션을 첨부하십시오.	서브스크립션에는 Red Hat 제품 비용이 포함되며, 다른 모든 인스턴스 비용에 대해 Amazon에 비용을 지불합니다. 기존 Red Hat 서브스크립션을 활용하기 때문에 AWS로 이동하는 사용자 지정 이미지는 "Cloud Access" 이미지입니다. Red Hat은 Cloud Access 이미지에 대한 직접 지원을 제공합니다.
RHEL이 포함된 기존 Amazon 이미지를 배포하도록 선택합니다.	AWS EC2 이미지에는 Red Hat 제품이 포함되어 있습니다.	AWS Management Console 에서 인스턴스를 시작할 때 RHEL 이미지를 선택하거나 AWS Marketplace 에서 이미지를 선택합니다.	사용량에 따라 사용량에 따라 사용량에 따라 Amazon을 사용량에 따라 책정할 수 있습니다. 이러한 이미지를 " 온 디맨드" 이미지라고 합니다. Amazon은 온디맨드 이미지에 대한 지원을 제공합니다. Red Hat은 이미지에 대한 업데이트를 제공합니다. AWS는 RHUI(Red Hat Update Infrastructure)를 통해 업데이트를 제공합니다.

참고

Red Hat Image Builder를 사용하여 AWS의 사용자 지정 이미지를 생성할 수 있습니다. 자세한 내용은 사용자 지정된 RHEL 시스템 이미지 구성 을 참조하십시오.



중요

온디맨드 인스턴스를 Red Hat Cloud Access 인스턴스로 변환할 수 없습니다. 온 디맨드 이미지에서 BYOS(your-your-own-subscription) 이미지로 변경하려면 새 Red Hat Cloud Access 인스턴스를 생성하고 온 디맨드 인스턴스에서 데이터를 마이그레이션합니다. 데이터를 마이그레이션한 후 이중 청구를 방지하기 위해 온 디맨드 인스턴스를 취소합니다.

다음 섹션에서는 사용자 지정 이미지와 관련된 정보 및 절차를 제공합니다.

추가 리소스

- Red Hat Cloud Access 프로그램
- 사용자 지정된 RHEL 시스템 이미지 구성
- AWS 관리 콘솔
- AWS Marketplace

3.2. 기본 이미지 이해

이 섹션에는 사전 구성된 기본 이미지 및 해당 구성 설정 사용에 대한 정보가 포함되어 있습니다.

3.2.1. 사용자 정의 기본 이미지 사용

VM(가상 시스템)을 수동으로 구성하려면 먼저 기본(시작자) VM 이미지를 생성합니다. 다음으로 구성 설정을 수정하고 VM이 클라우드에서 작동하는 데 필요한 패키지를 추가할 수 있습니다. 이미지를 업로드한 후 특정 애플리케이션에 대한 추가 구성을 변경할 수 있습니다.

추가 리소스

- Red Hat Enterprise Linux

3.2.2. 가상 머신 구성 설정

클라우드 VM에는 다음과 같은 구성 설정이 있어야 합니다.

표 3.2. VM 구성 설정

설정	권장 사항
ssh	VM에 대한 원격 액세스를 제공하려면 SSH를 활성화해야 합니다.
dhcp	기본 가상 어댑터는 dhcp에 대해 구성해야 합니다.

3.3. ISO 이미지에서 기본 VM 생성

이 섹션의 절차에 따라 ISO 이미지에서 RHEL 9 기본 이미지를 생성합니다.

사전 요구 사항

- 호스트 머신에서 가상화가 활성화되어 있습니다.
- Red Hat 고객 포털에서 최신 Red Hat Enterprise Linux ISO 이미지를 다운로드하여 이미지를 **/var/lib/libvirt/images** 로 이동했습니다.

3.3.1. RHEL ISO 이미지에서 VM 생성

절차

1. 가상화를 위해 호스트 시스템을 활성화했는지 확인합니다. 자세한 내용은 RHEL 9에서 가상화 활성화를 참조하십시오.
2. 기본 Red Hat Enterprise Linux VM을 만들고 시작합니다. 자세한 내용은 가상 머신 생성을 참조하십시오.
   a. 명령줄을 사용하여 VM을 생성하는 경우 기본 메모리 및 CPU를 VM에 필요한 용량으로 설정해야 합니다. 가상 네트워크 인터페이스를 **virtio**로 설정합니다.
   예를 들어 다음 명령은 **/home/username/Downloads/rhel9.iso** 이미지를 사용하여 **kvmtest** VM을 생성합니다.

```
# virt-install \
    --name kvmtest --memory 2048 --vcpus 2 \
    --cdrom /home/username/Downloads/rhel9.iso,bus=virtio \
    --os-variant=rhel9.0
```

   b. 웹 콘솔을 사용하여 VM을 생성하는 경우 다음 주의 사항과 함께 웹 콘솔을 사용하여 가상 머신 생성 절차를 따르십시오.
      - **즉시 VM 시작**을 선택하지 마십시오.
      - **메모리** 크기를 원하는 설정으로 변경합니다.
      - 설치를 시작하기 전에 **가상 네트워크 인터페이스 설정**에서 **모델**을 **virtio**로 변경하고 **vCPU**를 VM에 필요한 용량 설정으로 변경했는지 확인합니다.

3.3.2. RHEL 설치 완료

다음 단계를 수행하여 설치를 완료하고 VM이 시작되면 루트 액세스를 활성화합니다.

절차

1. 설치 프로세스 중에 사용할 언어를 선택합니다.
2. **설치 요약** 보기에서 다음을 실행합니다.
   a. **소프트웨어 선택** 사항을 클릭하고 **최소 설치를** 확인합니다.
   b. **Done** 을 클릭합니다.
   c. **Installation Destination** (설치 대상)을 클릭하고 스토리지 구성에서 **Custom** (사용자 **지정**)을 선택합니다.
      - **/boot**에 대해 최소 500MB인지 확인합니다. root에 남은 공간을 사용할 수 있습니다. /.
      - 표준 파티션은 권장되지만 LVM(Logical Volume Management)을 사용할 수 있습니다.

- 파일 시스템에 xfs, ext4 또는 ext3을 사용할 수 있습니다.
- 변경 사항이 완료되면 **완료**를 클릭합니다.

3. **설치 시작**을 클릭합니다.
4. **Root 암호**를 설정합니다. 해당되는 경우 다른 사용자를 만듭니다.
5. VM을 재부팅하고 설치가 완료되면 **root**로 로그인합니다.
6. 이미지를 구성합니다.
   a. VM을 등록하고 Red Hat Enterprise Linux 9 리포지토리를 활성화합니다.

```
# subscription-manager register --auto-attach
```

   b. **cloud-init** 패키지가 설치되어 활성화되어 있는지 확인합니다.

```
# dnf install cloud-init
# systemctl enable --now cloud-init.service
```

7. **중요: 이 단계는 AWS에 업로드하려는 VM에만 해당합니다.**
   a. AMD64 또는 Intel 64(x86_64) VM의 경우 **nvme**,**xen-netfront** 및 **xen-blkfront** 드라이버를 설치합니다.

```
# dracut -f --add-drivers "nvme xen-netfront xen-blkfront"
```

   b. ARM 64(aarch64) VM의 경우 **nvme** 드라이버를 설치합니다.

```
# dracut -f --add-drivers "nvme"
```

   이러한 드라이버를 포함하면 시간 초과 가능성이 제거됩니다.

   또는 **/etc/dracut.conf.d/** 에 드라이버를 추가한 다음 **dracut -f** 를 입력하여 기존 **initramfs** 파일을 덮어쓸 수 있습니다.
8. VM의 전원을 끕니다.

추가 리소스

- 고객 포털에서 자동 연결 서브스크립션 이해
- cloud-init 소개

3.4. RED HAT ENTERPRISE LINUX 이미지를 AWS에 업로드

이 섹션의 절차에 따라 이미지를 AWS에 업로드합니다.

3.4.1. AWS CLI 설치

AWS에서 HA 클러스터를 관리하는 데 필요한 대부분의 절차에는 AWS CLI 사용이 포함됩니다. AWS CLI를 설치하려면 다음 단계를 완료합니다.

사전 요구 사항

- AWS 액세스 키 ID와 AWS Secret Access Key를 생성하고 액세스할 수 있습니다. 지침 및 자세한 내용은 AWS CLI 구성을 빠르게 참조하십시오.

절차

1. **dnf** 명령을 사용하여 AWS 명령행 툴 을 설치합니다.

```
# dnf install awscli
```

2. **aws --version** 명령을 사용하여 AWS CLI를 설치했는지 확인합니다.

```
$ aws --version
aws-cli/1.19.77 Python/3.6.15 Linux/5.14.16-201.fc34.x86_64 botocore/1.20.77
```

3. AWS 액세스 세부 정보에 따라 AWS 명령줄 클라이언트를 구성합니다.

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

추가 리소스

- AWS CLI 빠른 구성
- AWS 명령줄 툴

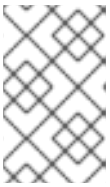
3.4.2. S3 버킷 생성

AWS로 가져오려면 Amazon S3 버킷이 필요합니다. Amazon S3 버킷은 오브젝트를 저장하는 Amazon 리소스입니다. 이미지를 업로드하는 프로세스의 일부로 S3 버킷을 생성한 다음 이미지를 버킷으로 이동합니다. 버킷을 생성하려면 다음 단계를 완료합니다.

절차

1. Amazon S3 콘솔을 시작합니다.
2. **Create Bucket** 을 클릭합니다. **버킷 생성** 대화 상자가 표시됩니다.
3. **이름 및 지역** 보기에서 다음을 수행합니다.
   a. **버킷 이름을 Bucket** 으로 입력합니다.
   b. **리전** 을 입력합니다.
   c. **다음**을 클릭합니다.
4. **Configure options** view에서 원하는 옵션을 선택하고 **다음을** 클릭합니다.
5. **권한 설정** 보기에서 기본 옵션을 변경하거나 수락하고 **다음을** 클릭합니다.
6. 버킷 구성을 검토합니다.

7. **버킷 생성을** 클릭합니다.



참고

또는 AWS CLI를 사용하여 버킷을 생성할 수 있습니다. 예를 들어 **aws s3 mb s3://my-new-bucket** 명령은 **my-new-bucket** 이라는 S3 버킷을 생성합니다. **mb** 명령에 대한 자세한 내용은 AWS CLI 명령 참조 를 참조하십시오.

추가 리소스

- Amazon S3 Console
- AWS CLI 명령 참조

3.4.3. vmimport 역할 생성

다음 절차에 따라 VM 가져오기에 필요한 **vmimport** 역할을 생성합니다. 자세한 내용은 Amazon 문서의 VM Import Service Role 을 참조하십시오.

절차

1. **trust-policy.json** 이라는 파일을 생성하고 다음 정책을 포함합니다. 파일을 시스템에 저장하고 위치를 기록해 둡니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": { "Service": "vmie.amazonaws.com" },
         "Action": "sts:AssumeRole",
         "Condition": {
            "StringEquals":{
               "sts:Externalid": "vmimport"
            }
         }
      }
   ]
}
```

2. **create role** 명령을 사용하여 **vmimport** 역할을 생성합니다. **trust-policy.json** 파일의 전체 경로를 지정합니다. **file://** 를 경로에 접두사로 지정합니다. 예를 들면 다음과 같습니다.

```
$ aws iam create-role --role-name vmimport --assume-role-policy-document
file:///home/sample/ImportService/trust-policy.json
```

3. **role-policy.json** 이라는 파일을 생성하고 다음 정책을 포함합니다. **s3-bucket-name** 을 S3 버킷 이름으로 바꿉니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
```

```
            "Action":[
              "s3:GetBucketLocation",
              "s3:GetObject",
              "s3:ListBucket"
            ],
            "Resource":[
              "arn:aws:s3:::s3-bucket-name",
              "arn:aws:s3:::s3-bucket-name/*"
            ]
          },
          {
            "Effect":"Allow",
            "Action":[
              "ec2:ModifySnapshotAttribute",
              "ec2:CopySnapshot",
              "ec2:RegisterImage",
              "ec2:Describe*"
            ],
            "Resource":"*"
          }
        ]
      }
```

4. **put-role-policy** 명령을 사용하여 정책을 생성한 역할에 연결합니다. **role-policy.json** 파일의 전체 경로를 지정합니다. 예를 들면 다음과 같습니다.

```
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document file:///home/sample/ImportService/role-policy.json
```

추가 리소스

- VM 가져오기 서비스 역할
- 필수 서비스 역할

3.4.4. 이미지를 S3로 변환 및 푸시

이미지를 변환하고 S3로 푸시하려면 다음 절차를 완료합니다. 샘플은 대표적인 것입니다. **qcow2** 파일 형식으로 포맷된 이미지를 **원시** 형식으로 변환합니다. Amazon은 **OVA**,**VHD**,**VHDX**,**VMDK**, **raw** 형식의 이미지를 허용합니다. Amazon에서 허용하는 이미지 형식에 대한 자세한 내용은 VM Import/Export Works 를 참조하십시오.

절차

1. **qemu-img** 명령을 실행하여 이미지를 변환합니다. 예를 들면 다음과 같습니다.

```
# qemu-img convert -f qcow2 -O raw rhel-9.0-sample.qcow2 rhel-9.0-sample.raw
```

2. 이미지를 S3로 푸시합니다.

```
$ aws s3 cp rhel-9.0-sample.raw s3://s3-bucket-name
```

참고

이 절차는 몇 분 정도 걸릴 수 있습니다. 완료되면 AWS S3 콘솔을 사용하여 이미지가 S3 버킷에 성공적으로 업로드되었는지 확인할 수 있습니다.

추가 리소스

- VM Import/Export Works
- AWS S3 Console

3.4.5. 스냅샷으로 이미지 가져오기

다음 절차에 따라 이미지를 스냅샷으로 가져옵니다.

절차

1. 파일을 생성하여 이미지의 버킷과 경로를 지정합니다. 파일 이름을 **containers.json** 으로 지정합니다. 다음 샘플에서 **s3-bucket-name** 을 버킷 이름으로 바꾸고 **s3-key** 를 키로 바꿉니다. Amazon S3 콘솔을 사용하여 이미지의 키를 가져올 수 있습니다.

```
{
    "Description": "rhel-9.0-sample.raw",
    "Format": "raw",
    "UserBucket": {
        "S3Bucket": "s3-bucket-name",
        "S3Key": "s3-key"
    }
}
```

2. 이미지를 스냅샷으로 가져옵니다. 이 예제에서는 공용 Amazon S3 파일을 사용합니다. Amazon S3 콘솔 을 사용하여 버킷의 권한 설정을 변경할 수 있습니다.

```
aws ec2 import-snapshot --disk-container file://containers.json
```

터미널에 다음과 같은 메시지가 표시됩니다. 메시지 내에 **ImportTaskID** 를 기록해 둡니다.

```
{
    "SnapshotTaskDetail": {
        "Status": "active",
        "Format": "RAW",
        "DiskImageSize": 0.0,
        "UserBucket": {
            "S3Bucket": "s3-bucket-name",
            "S3Key": "rhel-9.0-sample.raw"
        },
        "Progress": "3",
        "StatusMessage": "pending"
    },
    "ImportTaskId": "import-snap-06cea01fa0f1166a8"
}
```

3. **describe-import-snapshot-tasks** 명령을 사용하여 가져오기 진행 상황을 추적합니다. **ImportTaskID** 를 포함합니다.

```
$ aws ec2 describe-import-snapshot-tasks --import-task-ids import-snap-
06cea01fa0f1166a8
```

반환된 메시지는 작업의 현재 상태를 표시합니다. 완료되면 **Status** (상태)가 **완료된** 것으로 표시됩니다. 상태 내에서 스냅샷 ID를 기록하십시오.

추가 리소스

- Amazon S3 Console
- VM Import/Export를 사용하여 스냅샷으로 디스크 가져오기

3.4.6. 업로드된 스냅샷에서 AMI 생성

EC2 내에서 인스턴스를 시작할 때 AMI(Amazon Machine Image)를 선택해야 합니다. 업로드된 스냅샷에서 AMI를 생성하려면 다음 절차를 수행합니다.

절차

1. AWS EC2 대시보드로 이동합니다.
2. **Elastic Block Store** 에서 **스냅샷** 을 선택합니다.
3. 스냅샷 ID(예: **snap-0e718930bd72bcda0**)를 검색합니다.
4. 스냅샷을 마우스 오른쪽 버튼으로 클릭하고 **Create image** 를 선택합니다.
5. 이미지의 이름을 지정합니다.
6. 가상화 **유형에서** 하드웨어 **지원 가상화를** 선택합니다.
7. **생성**을 클릭합니다. 이미지 생성에 대한 노트에는 이미지에 대한 링크가 있습니다.
8. 이미지 링크를 클릭합니다. 이미지는 **Images>AMI** 아래에 표시됩니다.

참고

또는 AWS CLI **register-image** 명령을 사용하여 스냅샷에서 AMI를 생성할 수 있습니다. 자세한 내용은 register-image 를 참조하십시오. 예는 다음과 같습니다.

```
$ aws ec2 register-image \
   --name "myimagename" --description "myimagedescription" --
architecture x86_64 \
   --virtualization-type hvm --root-device-name "/dev/sda1" --ena-
support \
   --block-device-mappings "{\"DeviceName\": \"/dev/sda1\",\"Ebs\":
{\"SnapshotId\": \"snap-0ce7f009b69ab274d\"}}"
```

루트 장치 볼륨 **/dev/sda1** 을 **root-device-name** 으로 지정해야 합니다. AWS의 장치 매핑에 대한 개념 정보는 블록 장치 매핑 예제 를 참조하십시오.

3.4.7. AMI에서 인스턴스 시작

AMI에서 인스턴스를 시작하고 구성하려면 다음 절차를 수행합니다.

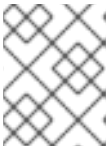
절차

1. AWS EC2 대시보드에서 **이미지** 및 **AMI**를 선택합니다.
2. 이미지를 마우스 오른쪽 버튼으로 클릭하고 **시작을** 선택합니다.
3. 워크로드 요구 사항을 충족하거나 초과하는 **인스턴스 유형을** 선택합니다.
   인스턴스 유형에 대한 자세한 내용은 Amazon EC2 인스턴스 유형을 참조하십시오.
4. **Next: Configure Instance Details** 를 클릭합니다.
   a. 생성할 **인스턴스 수를** 입력합니다.
   b. **Network** 는 AWS 환경을 설정할 때 생성한 VPC를 선택합니다. 인스턴스의 서브넷을 선택하거나 새 서브넷을 만듭니다.
   c. **Enable** for Auto-assign Public IP를 선택합니다.

참고

기본 인스턴스를 생성하는 데 필요한 최소 구성 옵션입니다. 애플리케이션 요구 사항에 따라 추가 옵션을 검토합니다.

5. **Next: Add Storage** 를 클릭합니다. 기본 스토리지가 충분한지 확인합니다.
6. **Next: 태그 추가를** 클릭합니다.



참고

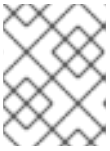
태그는 AWS 리소스를 관리하는 데 도움이 될 수 있습니다. 태그 지정에 대한 정보는 Amazon EC2 리소스 태그 지정을 참조하십시오.

7. **Next: Configure Security Group** 을 클릭합니다. AWS 환경을 설정할 때 생성한 보안 그룹을 선택합니다.
8. **검토 및 시작을** 클릭합니다. 선택 항목을 확인합니다.
9. **시작을** 클릭합니다. 기존 키 쌍을 선택하거나 새 키 쌍을 만들라는 메시지가 표시됩니다. AWS 환경을 설정할 때 생성한 키 쌍을 선택합니다.

참고

개인 키에 대한 권한이 올바른지 확인합니다. 필요한 경우 명령 옵션>-< **400 <keyname>.pem** 을 사용하여 권한을 변경합니다.

10. **인스턴스 시작을** 클릭합니다.
11. **인스턴스 보기** 를 클릭합니다. 인스턴스의 이름을 지정할 수 있습니다.
    인스턴스를 선택하고 **연결을 클릭하여 인스턴스에 대한 SSH 세션을 시작할 수 있습니다 독립 실행형 SSH 클라이언트에** 제공된 예제를 사용합니다.



참고

또는 AWS CLI를 사용하여 인스턴스를 시작할 수 있습니다. 자세한 내용은 Amazon 문서의 Amazon EC2 인스턴스 시작, 나열 및 종료 를 참조하십시오.

추가 리소스

- AWS 관리 콘솔
- Amazon EC2로 설정
- Amazon EC2 인스턴스
- Amazon EC2 인스턴스 유형

3.4.8. Red Hat 서브스크립션 첨부

Red Hat Cloud Access 프로그램을 통해 이전에 활성화한 서브스크립션을 연결하려면 다음 단계를 완료합니다.

사전 요구 사항

- 서브스크립션을 활성화해야 합니다.

절차

1. 시스템을 등록합니다.

```
# subscription-manager register --auto-attach
```

2. 서브스크립션을 첨부합니다.
   - 활성화 키를 사용하여 서브스크립션을 연결할 수 있습니다. 자세한 내용은 Red Hat 고객 포털 활성화 키 생성을 참조하십시오.
   - 또는 서브스크립션 풀 ID(Pool ID)를 사용하여 서브스크립션을 수동으로 연결할 수 있습니다. 명령줄을 통해 서브스크립션 연결 및 제거 를 참조하십시오.

추가 리소스

- Red Hat 고객 포털 활성화 키 생성
- 명령줄을 통해 서브스크립션 연결 및 제거
- Red Hat 서브스크립션 관리자 사용 및 구성

3.4.9. AWS Gold Images에서 자동 등록 설정

AWS(Amazon Web Services)에 RHEL 8 가상 머신을 더 빠르고 편리하게 배포하기 위해 Red Hat Subscription Manager(RHSM)에 RHEL 8의 Gold 이미지를 자동으로 등록하도록 설정할 수 있습니다.

사전 요구 사항

- AWS의 최신 RHEL 8 Gold 이미지를 다운로드했습니다. 자세한 내용은 AWS에서 골드 이미지 사용을 참조하십시오.

참고

AWS 계정은 한 번에 하나의 Red Hat 계정에만 연결할 수 있습니다. 따라서 다른 사용자가 Red Hat 계정에 연결하기 전에 AWS 계정에 액세스할 필요가 없는지 확인하십시오.

절차

1. AWS에 골드 이미지를 업로드합니다. 자세한 내용은 AWS에 Red Hat Enterprise Linux 이미지 업로드를 참조하십시오.
2. 업로드한 이미지를 사용하여 VM을 생성합니다. 자동으로 RHSM에 등록됩니다.

검증

- 위 지침을 사용하여 생성된 RHEL 9 VM에서 **subscription-manager identity** 명령을 실행하여 시스템이 RHSM에 등록되어 있는지 확인합니다. 성공적으로 등록된 시스템에서 시스템의 UUID가 표시됩니다. 예를 들면 다음과 같습니다.

```
# subscription-manager identity
system identity: fdc46662-c536-43fb-a18a-bbcb283102b7
name: 192.168.122.222
org name: 6340056
org ID: 6340056
```

추가 리소스

- AWS 관리 콘솔
- Red Hat 서비스의 클라우드 소스 구성

3.5. 추가 리소스

- Red Hat Cloud Access 참조 가이드
- 퍼블릭 클라우드의 Red Hat
- Amazon EC2 기반 Red Hat Enterprise Linux - FAQ
- Amazon EC2로 설정
- Amazon Web Services의 Red Hat

4장. AWS에서 RED HAT HIGH AVAILABILITY 클러스터 구성

이 장에서는 EC2 인스턴스를 클러스터 노드로 사용하여 AWS(Amazon Web Services)에서 Red Hat High Availability(HA) 클러스터를 구성하는 데 필요한 정보 및 절차를 설명합니다. 클러스터에 사용하는 RHEL(Red Hat Enterprise Linux) 이미지를 가져올 수 있는 다양한 옵션이 있습니다. AWS의 이미지 옵션에 대한 자세한 내용은 AWS의 Red Hat Enterprise Linux 이미지 옵션을 참조하십시오.

이 장에서는 다음을 설명합니다.

- AWS 환경을 설정하기 위한 사전 요구 사항 절차 환경을 설정한 후에는 EC2 인스턴스를 생성하고 구성할 수 있습니다.
- 개별 노드를 AWS의 HA 노드 클러스터로 변환하는 HA 클러스터 생성과 관련된 절차입니다. 여기에는 각 클러스터 노드에 High Availability 패키지 및 에이전트를 설치하고, 펜싱을 구성하며, AWS 네트워크 리소스 에이전트를 설치하는 절차가 포함됩니다.

사전 요구 사항

- Red Hat 고객 포털 계정에 등록합니다.
- AWS에 가입하고 AWS 리소스를 설정합니다. 자세한 내용은 Amazon EC2로 설정을 참조하십시오.
- Red Hat Cloud Access 프로그램 에서 서브스크립션을 활성화합니다. Red Hat Cloud Access 프로그램을 사용하면 Red Hat 서브스크립션을 물리적 또는 온프레미스 시스템에서 AWS로 이전할 수 있으며 Red Hat의 완전한 지원을 받을 수 있습니다.

4.1. AWS ACCESS KEY 및 AWS SECRET ACCESS KEY 생성

AWS CLI를 설치하기 전에 AWS Access Key 및 AWS Secret Access Key를 생성해야 합니다. 펜싱 및 리소스 에이전트 API는 AWS Access Key 및 Secret Access Key를 사용하여 클러스터의 각 노드에 연결합니다.

이러한 키를 생성하려면 다음 단계를 완료합니다.

사전 요구 사항

- IAM 사용자 계정에 프로그래밍 방식의 액세스 권한이 있어야 합니다. 자세한 내용은 AWS 환경 설정을 참조하십시오.

절차

1. AWS 콘솔 을 시작합니다.
2. AWS 계정 ID를 클릭하여 드롭다운 메뉴를 표시하고 **My Security Credentials** 를 선택합니다.
3. **사용자를** 클릭합니다.
4. 사용자를 선택하고 **Summary** (요약) 화면을 엽니다.
5. **보안 자격 증명** 탭을 클릭합니다.
6. **액세스 키 생성을** 클릭합니다.
7. **.csv** 파일을 다운로드하거나 두 키를 모두 저장합니다. 펜싱 장치를 생성할 때 이 키를 입력해야 합니다.

4.2. AWS CLI 설치

AWS에서 HA 클러스터를 관리하는 데 필요한 대부분의 절차에는 AWS CLI 사용이 포함됩니다. AWS CLI를 설치하려면 다음 단계를 완료합니다.

사전 요구 사항

- AWS 액세스 키 ID와 AWS Secret Access Key를 생성하고 액세스할 수 있습니다. 지침 및 자세한 내용은 AWS CLI 구성을 빠르게 참조하십시오.

절차

1. **dnf** 명령을 사용하여 AWS 명령행 툴 을 설치합니다.

```
# dnf install awscli
```

2. **aws --version** 명령을 사용하여 AWS CLI를 설치했는지 확인합니다.

```
$ aws --version
aws-cli/1.19.77 Python/3.6.15 Linux/5.14.16-201.fc34.x86_64 botocore/1.20.77
```

3. AWS 액세스 세부 정보에 따라 AWS 명령줄 클라이언트를 구성합니다.

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

추가 리소스

- AWS CLI 빠른 구성
- AWS 명령줄 툴

4.3. HA EC2 인스턴스 생성

HA 클러스터 노드로 사용하는 인스턴스를 생성하려면 다음 단계를 완료합니다. 클러스터에 사용하는 RHEL 이미지를 가져올 수 있는 여러 옵션이 있습니다. AWS의 이미지 옵션에 대한 자세한 내용은 AWS의 Red Hat Enterprise Linux 이미지 옵션을 참조하십시오.

클러스터 노드에 사용하는 사용자 지정 이미지를 생성 및 업로드하거나 Gold Image(Cloud Access) 이미지 또는 온 디맨드 이미지를 선택할 수 있습니다.

사전 요구 사항

- AWS 환경을 설정해야 합니다. 자세한 내용은 Amazon EC2로 설정을 참조하십시오.

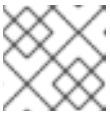
절차

1. AWS EC2 대시보드에서 **이미지** 및 **AMI**를 선택합니다.
2. 이미지를 마우스 오른쪽 버튼으로 클릭하고 **시작을** 선택합니다.

3. 워크로드 요구 사항을 충족하거나 초과하는 **인스턴스 유형을** 선택합니다. HA 애플리케이션에 따라 각 인스턴스에 더 높은 용량이 있어야 할 수 있습니다.
   인스턴스 유형에 대한 자세한 내용은 Amazon EC2 인스턴스 유형을 참조하십시오.

4. **Next: Configure Instance Details** 를 클릭합니다.

   a. 클러스터에 생성할 **인스턴스** 수를 입력합니다. 이 예제 절차에서는 세 개의 클러스터 노드를 사용합니다.

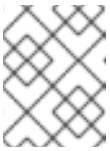
   

   참고

   자동 확장 그룹으로 시작하지 마십시오.

   b. **Network**, select the VPC you created in Set up the AWS environment. 인스턴스의 서브넷을 선택하여 새 서브넷을 만듭니다.

   c. **Enable** for Auto-assign Public IP를 선택합니다. **인스턴스 세부** 정보에 대해 수행해야 하는 최소 선택 사항입니다. 특정 HA 애플리케이션에 따라 추가 선택 항목이 필요할 수 있습니다.

   참고

   기본 인스턴스를 생성하는 데 필요한 최소 구성 옵션입니다. HA 애플리케이션 요구 사항에 따라 추가 옵션을 검토합니다.

5. **Next: Add Storage** 를 클릭하고 기본 스토리지가 충분한지 확인합니다. HA 애플리케이션에 다른 스토리지 옵션이 필요한 경우를 제외하고 이러한 설정은 수정할 필요가 없습니다.

6. **Next: 태그 추가를** 클릭합니다.

   참고

   태그는 AWS 리소스를 관리하는 데 도움이 될 수 있습니다. 태그 지정에 대한 정보는 Amazon EC2 리소스 태그 지정을 참조하십시오.

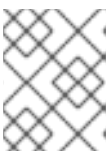
7. **Next: Configure Security Group** 을 클릭합니다. AWS 환경 설정에서 생성한 기존 보안 그룹을 선택합니다.

8. **검토 및 시작을** 클릭하고 선택 항목을 확인합니다.

9. **시작을** 클릭합니다. 기존 키 쌍을 선택하거나 새 키 쌍을 만들라는 메시지가 표시됩니다. AWS 환경을 설정할 때 생성한 키 쌍을 선택합니다.

10. **인스턴스 시작을** 클릭합니다.

11. **인스턴스 보기** 를 클릭합니다. 인스턴스의 이름을 지정할 수 있습니다.

    참고

    또는 AWS CLI를 사용하여 인스턴스를 시작할 수 있습니다. 자세한 내용은 Amazon 문서의 Amazon EC2 인스턴스 시작, 나열 및 종료 를 참조하십시오.

추가 리소스

- AWS 관리 콘솔

- Amazon EC2로 설정
- Amazon EC2 인스턴스
- Amazon EC2 인스턴스 유형

4.4. 개인 키 구성

SSH 세션에서 사용하려면 개인 SSH 키 파일(**.pem**)을 사용하도록 다음 설정 작업을 완료합니다.

절차

1. **Downloads** 디렉토리에서 **홈** 디렉토리 또는 **~/.ssh 디렉토리로** 키 파일을 이동합니다.
2. root 사용자만 읽을 수 있도록 키 파일의 권한을 변경합니다.

```
# chmod 400 KeyName.pem
```

4.5. EC2 인스턴스에 연결

EC2 인스턴스에 연결하려면 모든 노드에서 다음 단계를 완료합니다.

절차

1. AWS 콘솔 을 시작하고 EC2 인스턴스를 선택합니다.
2. **연결을** 클릭하고 **독립 실행형 SSH 클라이언트를** 선택합니다.
3. SSH 터미널 세션에서 팝업 창에 제공된 AWS 예제를 사용하여 인스턴스에 연결합니다. 예제에 경로가 표시되지 않는 경우 **KeyName.pem** 파일에 올바른 경로를 추가합니다.

4.6. 고가용성 패키지 및 에이전트 설치

고가용성 패키지 및 에이전트를 설치하려면 모든 노드에서 다음 단계를 완료합니다.

절차

1. AWS RHUI(Red Hat Update Infrastructure) 클라이언트를 제거합니다. Red Hat Cloud Access 서브스크립션을 사용할 예정이므로 서브스크립션 외에 AWS RHUI를 사용해서는 안 됩니다.

```
$ sudo -i
# dnf -y remove rh-amazon-rhui-client*
```

2. VM을 Red Hat에 등록합니다.

```
# subscription-manager register --auto-attach
```

3. 모든 리포지토리를 비활성화합니다.

```
# subscription-manager repos --disable=*
```

4. RHEL 9 Server HA 리포지토리를 활성화합니다.

```
# subscription-manager repos --enable=rhel-9-for-x86_64-highavailability-rpms
```

5. RHEL AWS 인스턴스를 업데이트합니다.

```
# dnf update -y
```

6. 고가용성 채널에서 사용 가능한 모든 펜싱 에이전트와 함께 Red Hat High Availability Add-On 소프트웨어 패키지를 설치합니다.

```
# dnf install pcs pacemaker fence-agents-aws
```

7. 사용자 **hacluster** 는 이전 단계에서 **pcs** 및 **pacemaker** 설치 중에 생성되었습니다. 모든 클러스터 노드에서 **hacluster**의 암호를 생성합니다. 모든 노드에 동일한 암호를 사용합니다.

```
# passwd hacluster
```

8. **firewalld.service**가 설치된 경우 RHEL 방화벽에 **high availability** 서비스를 추가합니다.

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --reload
```

9. **pcs** 서비스를 시작하고 부팅 시 시작되도록 활성화합니다.

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

10. **/etc/hosts** 를 편집하고 RHEL 호스트 이름 및 내부 IP 주소를 추가합니다. 자세한 내용은 RHEL 클러스터 노드에 /etc/hosts 파일을 어떻게 설정해야 합니까?

검증

- **pcs** 서비스가 실행 중인지 확인합니다.

```
# systemctl status pcsd.service

pcsd.service - PCS GUI and remote configuration interface
Loaded: loaded (/usr/lib/systemd/system/pcsd.service; enabled; vendor preset: disabled)
Active: active (running) since Thu 2018-03-01 14:53:28 UTC; 28min ago
Docs: man:pcsd(8)
man:pcs(8)
Main PID: 5437 (pcsd)
CGroup: /system.slice/pcsd.service
    └─5437 /usr/bin/ruby /usr/lib/pcsd/pcsd > /dev/null &
Mar 01 14:53:27 ip-10-0-0-48.ec2.internal systemd[1]: Starting PCS GUI and remote
configuration interface…
Mar 01 14:53:28 ip-10-0-0-48.ec2.internal systemd[1]: Started PCS GUI and remote
configuration interface.
```

4.7. 클러스터 생성

노드 클러스터를 생성하려면 다음 단계를 완료합니다.

절차

1. 노드 중 하나에서 다음 명령을 입력하여 pcs 사용자 **hacluster**를 인증합니다. 명령에서 클러스터의 각 노드의 이름을 지정합니다.

```
# pcs host auth <hostname1> <hostname2> <hostname3>
```

예제:

```
[root@node01 clouduser]# pcs host auth node01 node02 node03
Username: hacluster
Password:
node01: Authorized
node02: Authorized
node03: Authorized
```

2. 클러스터를 생성합니다.

```
# pcs cluster setup <cluster_name> <hostname1> <hostname2> <hostname3>
```

예제:

```
[root@node01 clouduser]# pcs cluster setup new_cluster node01 node02 node03

[...]

Synchronizing pcsd certificates on nodes node01, node02, node03...
node02: Success
node03: Success
node01: Success
Restarting pcsd on the nodes in order to reload the certificates...
node02: Success
node03: Success
node01: Success
```

검증

1. 클러스터를 활성화합니다.

```
[root@node01 clouduser]# pcs cluster enable --all
node02: Cluster Enabled
node03: Cluster Enabled
node01: Cluster Enabled
```

2. 클러스터를 시작합니다.

```
[root@node01 clouduser]# pcs cluster start --all
node02: Starting Cluster...
node03: Starting Cluster...
node01: Starting Cluster...
```

4.8. 펜싱 구성

펜싱 구성을 사용하면 AWS 클러스터의 노드가 자동으로 분리되므로 노드가 클러스터 리소스를 사용하거나 클러스터 기능을 손상시키지 않습니다.

여러 방법을 사용하여 AWS 클러스터에서 펜싱을 구성할 수 있습니다. 이 섹션에서는 다음을 제공합니다.

- 기본 구성에 대한 표준 절차입니다.
- 자동화에 중점을 둔 고급 구성을 위한 대체 구성 절차입니다.

표준 절차

1. 다음 AWS 메타데이터 쿼리를 입력하여 각 노드의 인스턴스 ID를 가져옵니다. 차단 장치를 구성하려면 이러한 ID가 필요합니다. 자세한 내용은 인스턴스 메타데이터 및 사용자 데이터를 참조하십시오.

```
# echo $(curl -s http://169.254.169.254/latest/meta-data/instance-id)
```

예제:

```
[root@ip-10-0-0-48 ~]# echo $(curl -s http://169.254.169.254/latest/meta-data/instance-id) i-07f1ac63af0ec0ac6
```

2. 다음 명령을 입력하여 펜스 장치를 구성합니다. gRPC **mk_host_map** 명령을 사용하여 RHEL 호스트 이름을 인스턴스 ID에 매핑합니다. 이전에 설정한 AWS Access Key 및 AWS Secret Access Key를 사용합니다.

```
# pcs stonith \
    create <name> fence_aws access_key=access-key secret_key=<secret-access-key> \
    region=<region> pcmk_host_map="rhel-hostname-1:Instance-ID-1;rhel-hostname-2:Instance-ID-2;rhel-hostname-3:Instance-ID-3" \
    power_timeout=240 pcmk_reboot_timeout=480 pcmk_reboot_retries=4
```

예제:

```
[root@ip-10-0-0-48 ~]# pcs stonith \
create clusterfence fence_aws access_key=AKIAI123456MRMJA
secret_key=a75EYIG4RVL3hdsdAslK7koQ8dzaDyn5yoIZ/ \
region=us-east-1 pcmk_host_map="ip-10-0-0-48:i-07f1ac63af0ec0ac6;ip-10-0-0-46:i-063fc5fe93b4167b2;ip-10-0-0-58:i-08bd39eb03a6fd2c7" \
power_timeout=240 pcmk_reboot_timeout=480 pcmk_reboot_retries=4
```

대체 절차

1. 클러스터의 VPC ID를 가져옵니다.

```
# aws ec2 describe-vpcs --output text --filters "Name=tag:Name,Values=clustername-vpc" --query 'Vpcs[*].VpcId'
vpc-06bc10ac8f6006664
```

2. 클러스터의 VPC ID를 사용하여 VPC 인스턴스를 가져옵니다.

```
$ aws ec2 describe-instances --output text --filters "Name=vpc-id,Values=vpc-06bc10ac8f6006664" --query 'Reservations[*].Instances[*].{Name:Tags[? Key==Name]|[0].Value,Instance:InstanceId}' | grep "\-node[a-c]"
```

```
i-0b02af8927a895137     clustername-nodea-vm
i-0cceb4ba8ab743b69     clustername-nodeb-vm
i-0502291ab38c762a5     clustername-nodec-vm
```

3. 가져온 인스턴스 ID를 사용하여 클러스터의 각 노드에서 펜싱을 구성합니다. 예를 들면 다음과 같습니다.

```
[root@nodea ~]# CLUSTER=clustername && pcs stonith create fence${CLUSTER}
fence_aws access_key=XXXXXXXXXXXXXXXXXXXX pcmk_host_map=$(for NODE \
in node{a..c}; do ssh ${NODE} "echo -n \${HOSTNAME}:\$(curl -s
http://169.254.169.254/latest/meta-data/instance-id)\;"; done) \
pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240 region=xx-xxxx-x
secret_key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

[root@nodea ~]# pcs stonith config fence${CLUSTER}

Resource: clustername (class=stonith type=fence_aws)
Attributes: access_key=XXXXXXXXXXXXXXXXXXXX pcmk_host_map=nodea:i-
0b02af8927a895137;nodeb:i-0cceb4ba8ab743b69;nodec:i-0502291ab38c762a5;
pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240 region=xx-xxxx-x
secret_key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Operations: monitor interval=60s (clustername-monitor-interval-60s)
```

검증

1. 클러스터 노드 중 하나에 대해 펜싱 에이전트를 테스트합니다.

```
# pcs stonith fence awsnodename
```

참고

명령 응답이 표시되는 데 몇 분이 걸릴 수 있습니다. 펜싱 중인 노드의 활성 터미널 세션을 확인하는 경우 fence 명령을 입력한 후 터미널 연결이 즉시 종료되는 것을 확인할 수 있습니다.

예제:

```
[root@ip-10-0-0-48 ~]# pcs stonith fence ip-10-0-0-58

Node: ip-10-0-0-58 fenced
```

2. 상태를 확인하여 노드가 펜싱되었는지 확인합니다.

```
# pcs status
```

예제:

```
[root@ip-10-0-0-48 ~]# pcs status

Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-46 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Mar  2 19:55:41 2018
```

```
Last change: Fri Mar  2 19:24:59 2018 by root via cibadmin on ip-10-0-0-46

3 nodes configured
1 resource configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ]
OFFLINE: [ ip-10-0-0-58 ]

Full list of resources:
clusterfence  (stonith:fence_aws):    Started ip-10-0-0-46

Daemon Status:
corosync: active/disabled
pacemaker: active/disabled
pcsd: active/enabled
```

3. 이전 단계에서 차단된 노드를 시작합니다.

```
# pcs cluster start awshostname
```

4. 상태를 확인하여 노드가 시작되었는지 확인합니다.

```
# pcs status
```

예제:

```
[root@ip-10-0-0-48 ~]# pcs status

Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-46 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Mar  2 20:01:31 2018
Last change: Fri Mar  2 19:24:59 2018 by root via cibadmin on ip-10-0-0-48

3 nodes configured
1 resource configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ip-10-0-0-58 ]

Full list of resources:

 clusterfence  (stonith:fence_aws):    Started ip-10-0-0-46

Daemon Status:
 corosync: active/disabled
 pacemaker: active/disabled
 pcsd: active/enabled
```

4.9. 클러스터 노드에 AWS CLI 설치

이전에는 호스트 시스템에 AWS CLI를 설치했습니다. 네트워크 리소스 에이전트를 구성하기 전에 클러스터 노드에 AWS CLI를 설치해야 합니다.

각 클러스터 노드에서 다음 절차를 완료합니다.

사전 요구 사항

- AWS Access Key 및 AWS Secret Access Key를 생성해야 합니다. 자세한 내용은 Creating the AWS Access Key and AWS Secret Access Key 를 참조하십시오.

절차

1. AWS CLI를 설치합니다. 자세한 내용은 AWS CLI 설치를 참조하십시오.
2. AWS CLI가 올바르게 구성되었는지 확인합니다. 인스턴스 ID와 인스턴스 이름이 표시되어야 합니다.
   예제:

```
[root@ip-10-0-0-48 ~]# aws ec2 describe-instances --output text --query
'Reservations[].Instances[].[InstanceId,Tags[?Key==Name].Value]'

i-07f1ac63af0ec0ac6
ip-10-0-0-48
i-063fc5fe93b4167b2
ip-10-0-0-46
i-08bd39eb03a6fd2c7
ip-10-0-0-58
```

4.10. 네트워크 리소스 에이전트 설치

HA 작업이 작동하려면 클러스터에서 AWS 네트워킹 리소스 에이전트를 사용하여 페일오버 기능을 활성화합니다. 노드가 정해진 시간 내에 하트비트 검사에 응답하지 않으면 노드가 펜싱되고 클러스터의 추가 노드로 작업이 실패합니다. 네트워크 리소스 에이전트가 작동하도록 구성해야 합니다.

두 리소스를 동일한 그룹에 추가하여 **순서** 및 **공동 배치** 제약 조건을 적용합니다.

보조 개인 IP 리소스 및 가상 IP 리소스 생성

보조 개인 IP 주소를 추가하고 가상 IP를 생성하려면 다음 절차를 완료합니다. 클러스터의 모든 노드에서 이 프로세스를 완료할 수 있습니다.

절차

1. **AWS Secondary Private IP Address** 리소스 에이전트(awsvip) 설명을 확인합니다. 여기에는 이 에이전트의 옵션 및 기본 작업이 표시됩니다.

```
# pcs resource describe awsvip
```

2. **VPC CIDR** 블록에서 사용되지 않는 개인 IP 주소를 사용하여 보조 개인 IP 주소를 만듭니다.

```
# pcs resource create privip awsvip secondary_private_ip=Unused-IP-Address --group
group-name
```

   예제:

```
[root@ip-10-0-0-48 ~]# pcs resource create privip awsvip
secondary_private_ip=10.0.0.68 --group networking-group
```

3. 가상 IP 리소스를 생성합니다. 이는 펜싱된 노드에서 장애 조치 노드로 빠르게 다시 매핑할 수 있는 VPC IP 주소입니다. 서브넷 내에서 펜싱된 노드의 장애를 마스킹합니다.

```
# pcs resource create vip IPaddr2 ip=secondary-private-IP --group group-name
```

예제:

```
root@ip-10-0-0-48 ~]# pcs resource create vip IPaddr2 ip=10.0.0.68 --group networking-group
```

검증

- 리소스가 실행 중인지 확인합니다.

```
# pcs status
```

예제:

```
[root@ip-10-0-0-48 ~]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-46 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Mar  2 22:34:24 2018
Last change: Fri Mar  2 22:14:58 2018 by root via cibadmin on ip-10-0-0-46

3 nodes configured
3 resources configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ip-10-0-0-58 ]

Full list of resources:

clusterfence    (stonith:fence_aws):    Started ip-10-0-0-46
 Resource Group: networking-group
     privip (ocf::heartbeat:awsvip):    Started ip-10-0-0-48
     vip    (ocf::heartbeat:IPaddr2):   Started ip-10-0-0-58

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

탄력적 IP 주소 생성

탄력적 IP 주소는 펜싱된 노드에서 장애 조치 노드로 빠르게 다시 매핑할 수 있는 공용 IP 주소로, 펜싱된 노드의 오류를 마스킹합니다.

이 리소스는 이전에 생성된 가상 IP 리소스와 다릅니다. 탄력적 IP 주소는 서브넷 연결 대신 공용 인터넷 연결에 사용됩니다.

1. 이전에 생성된 두 리소스를 동일한 그룹에 추가하여 **순서** 와 **공동 배치** 제약 조건을 적용합니다.

2. 다음 AWS CLI 명령을 입력하여 탄력적 IP 주소를 생성합니다.

```
[root@ip-10-0-0-48 ~]# aws ec2 allocate-address --domain vpc --output text
eipalloc-4c4a2c45   vpc 35.169.153.122
```

3. AWS Secondary Elastic IP Address 리소스 에이전트(awseip) 설명을 확인합니다. 다음 명령은 이 에이전트의 옵션 및 기본 작업을 보여줍니다.

```
# pcs resource describe awseip
```

4. 1단계에서 생성된 할당된 IP 주소를 사용하여 Secondary Elastic IP 주소 리소스를 생성합니다.

```
# pcs resource create elastic awseip elastic_ip=Elastic-IP-Address
allocation_id=Elastic-IP-Association-ID --group networking-group
```

예제:

```
# pcs resource create elastic awseip elastic_ip=35.169.153.122 allocation_id=eipalloc-
4c4a2c45 --group networking-group
```

검증

- **pcs status** 명령을 입력하여 리소스가 실행 중인지 확인합니다.

```
# pcs status
```

예제:

```
[root@ip-10-0-0-58 ~]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-58 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Mon Mar  5 16:27:55 2018
Last change: Mon Mar  5 15:57:51 2018 by root via cibadmin on ip-10-0-0-46

3 nodes configured
4 resources configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ip-10-0-0-58 ]

Full list of resources:

 clusterfence   (stonith:fence_aws):    Started ip-10-0-0-46
 Resource Group: networking-group
     privip (ocf::heartbeat:awsvip):  Started ip-10-0-0-48
     vip    (ocf::heartbeat:IPaddr2):    Started ip-10-0-0-48
     elastic (ocf::heartbeat:awseip):    Started ip-10-0-0-48

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

탄력적 IP 주소 테스트

다음 명령을 입력하여 가상 IP(awsvip) 및 탄력적 IP(awseip) 리소스가 작동하는지 확인합니다.

절차

1. 로컬 워크스테이션에서 이전에 생성된 탄력적 IP 주소로 SSH 세션을 시작합니다.

```
$ ssh -l ec2-user -i ~/.ssh/<KeyName>.pem elastic-IP
```

예제:

```
$ ssh -l ec2-user -i ~/.ssh/cluster-admin.pem 35.169.153.122
```

2. SSH를 통해 연결된 호스트가 생성된 탄력적 리소스와 연결된 호스트인지 확인합니다.

추가 리소스

- 고가용성 애드온 개요
- 고가용성 클러스터 구성 및 관리

4.11. 공유 블록 스토리지 구성

Amazon EBS(Elastic Block Storage) 다중 연결 볼륨을 사용하여 Red Hat High Availability 클러스터의 공유 블록 스토리지를 구성하려면 다음 절차를 사용하십시오. 이 절차는 선택 사항이며 아래 단계에서는 1TB 공유 디스크가 있는 세 개의 인스턴스(세 개의 노드 클러스터)를 가정합니다.

사전 요구 사항

- AWS Nitro System 기반 Amazon EC2 인스턴스를 사용해야 합니다.

절차

1. AWS 명령 create-volume 을 사용하여 공유 블록 볼륨을 생성합니다.

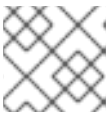
```
$ aws ec2 create-volume --availability-zone <availability_zone> --no-encrypted --size
1024 --volume-type io1 --iops 51200 --multi-attach-enabled
```

예를 들어 다음 명령은 **us-east-1a** 가용 영역에 볼륨을 생성합니다.

```
$ aws ec2 create-volume --availability-zone us-east-1a --no-encrypted --size 1024 --
volume-type io1 --iops 51200 --multi-attach-enabled

{
    "AvailabilityZone": "us-east-1a",
    "CreateTime": "2020-08-27T19:16:42.000Z",
    "Encrypted": false,
    "Size": 1024,
    "SnapshotId": "",
    "State": "creating",
    "VolumeId": "vol-042a5652867304f09",
    "Iops": 51200,
```

```
    "Tags": [ ],
    "VolumeType": "io1"
}
```



참고

다음 단계에서 **VolumeId** 가 필요합니다.

2. 클러스터의 각 인스턴스에 대해 AWS 명령 attach-volume 을 사용하여 공유 블록 볼륨을 연결합니다. < **instance_id> 및 &** lt; **volume_id>를** 사용합니다.

```
$ aws ec2 attach-volume --device /dev/xvdd --instance-id <instance_id> --volume-id
<volume_id>
```

예를 들어 다음 명령은 공유 블록 볼륨 **vol-042a5652867304f09** 를 **i-0eb803361c2c887f2 인스턴스에** 연결합니다.

```
$ aws ec2 attach-volume --device /dev/xvdd --instance-id i-0eb803361c2c887f2 --
volume-id vol-042a5652867304f09

{
    "AttachTime": "2020-08-27T19:26:16.086Z",
    "Device": "/dev/xvdd",
    "InstanceId": "i-0eb803361c2c887f2",
    "State": "attaching",
    "VolumeId": "vol-042a5652867304f09"
}
```

검증

1. 클러스터의 각 인스턴스에 대해 **ssh** 명령을 < **ip_address> 인스턴스와 함께 사용하여 블록 장치를 사용할 수 있는지 확인합니다**.

```
# ssh <ip_address> "hostname ; lsblk -d | grep ' 1T '"
```

예를 들어 다음 명령은 인스턴스 IP **198.51.100.3** 의 호스트 이름 및 블록 장치를 포함한 세부 정보를 나열합니다.

```
# ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T '"

nodea
nvme2n1 259:1    0   1T  0 disk
```

2. **ssh** 명령을 사용하여 클러스터의 각 인스턴스가 동일한 공유 디스크를 사용하는지 확인합니다.

```
# ssh <ip_address> "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i
udevadm info --query=all --name=/dev/{} | grep '^E: ID_SERIAL='"
```

예를 들어 다음 명령은 인스턴스 IP 주소 **198.51.100.3**의 호스트 이름 및 공유 디스크 볼륨 ID를 포함한 세부 정보를 나열합니다.

```
# ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm
```

```
info --query=all --name=/dev/{} | grep '^E: ID_SERIAL="'

nodea
E: ID_SERIAL=Amazon Elastic Block Store_vol0fa5342e7aedf09f7
```

추가 리소스

- 클러스터에서 ScanSetting2 파일 시스템 구성
- Chrony2 파일 시스템 구성

4.12. 추가 리소스

- Red Hat Cloud Access 참조 가이드
- 퍼블릭 클라우드의 Red Hat
- Amazon EC2 기반 Red Hat Enterprise Linux - FAQ
- Amazon EC2로 설정
- Amazon Web Services의 Red Hat