Amazon Elastic Container Registry Public (ECR Public)

Amazon Elastic Container Registry Public (ECR Public) 사용해보기

개요

Amazon Elastic Container Registry(Amazon ECR)는 안전하고 확장 가능하고 신뢰할 수 있는 AWS 관리형 컨테이너 이미지 레지스트리 서비스 입니다.
Amazon ECR은 AWS IAM을 사용하여 리소스 기반 권한이 있는 퍼블릭/프라이빗 이미지 리포지토리를 지원 합니다.
따라서 AWS 제공하는 ECS, EKS등에서 ECR을 쉽게 엑세스 할 수 있습니다.

요금

https://aws.amazon.com/ko/ecr/pricing/

프라이빗 리포지토리에서 전송된 데이터:

리전: 아시아 태평양(서울)

	요금
데이터 수신	
수신되는 모든 데이터	GB당 0.00 USD
데이터 송신**	
다음 9.999TB/월	GB당 0.126 USD
다음 40TB/월	GB당 0.122 USD
다음 100TB/월	GB당 0.117 USD
150TB 이상/월	GB당 0.108 USD

퍼블릭 리포지토리에서 전송된 데이터:

	요금
데이터 수신	
모든 데이터 수신	GB당 0.00 USD
데이터 송신***	
AWS 계정을 사용하지 않는 경우	
500GB/월	GB당 0.00 USD
AWS 계정을 사용하는 경우	
5TB/월	GB당 0.00 USD
AWS 리전 외 위치로 5TB/월을 초과하는 경우	GB당 0.09 USD
모든 AWS 리전 금액	GB당 0.00 USD

** 프라이빗 리포지토리에서 송신되는 데이터 요금 티어는 Amazon Elastic Cloud Compute(EC2), Amazon ECR, Amazon Elastic Block Storage(EBS), Amazon Simple Storage Service(S3), Amazon S3 Glacier, Amazon Relational Database Service(RDS), Amazon SimpleDB, Amazon Simple Queue Service(SQS), Amazon Simple Notification Service(SNS), Amazon DynamoDB, AWS Storage Gateway 및 Amazon Virtual Private Cloud(VPC) 전체의 아웃바운드 데이터 전송을 집계합니다.
*** 퍼블릭 리포지토리에서 송신되는 데이터는 AWS 계정을 사용하지 않는 경우 소스 IP로 제한됩니다.

요금 예제:

데이터 "수신" 및 "송신"은 Amazon ECR에서 데이터를 수신하고 송신하는 것을 말합니다. 동일한 리전의 Amazon ECR 및 기타 서비스 사이에서 전송된 데이터(예: Amazon EC2, AWS Lambda, AWS App Runner 또는 AWS Fargate)는 무료입니다(즉, GB당 0.00 USD). Amazon ECR과 다른 리전의 AWS 서비스 사이에 전송된 데이터에 대한 요금은 전송의 양쪽에서 인터넷 데이터 전송 요율로 부과됩니다.

요금 예제 1: Amazon ECR 프라이빗 리포지토리와 리전 내 전송
조직 내에서 비공개로 공유할 총 40GB의 소프트웨어 이미지를 저장합니다. 스토리지에는 GB당 0.10 USD로 한 달에 총 4 USD가 부과되지만 수신 데이터 전송에는 요금이 부과되지 않습니다. 조직의 다른 사용자는 Amazon EC2 또는 AWS Fargate에서 Amazon Elastic Container Service(Amazon ECS)를 사용하여 동일한 리전 내에서 월 1TB의 이미지를 가져오므로 송신 데이터 전송 요금이 부과되지 않습니다. **총 비용 = 4 USD/월**

요금 예제 2: Amazon ECR 프라이빗 리포지토리와 교차 리전 전송
us-east-1 리전에서 총 20GB의 소프트웨어 이미지를 저장합니다. 스토리지에는 한 달에 2 USD가 부과되지만 수신 데이터 전송에는 요금이 부과되지 않습니다. 조직의 다른 사용자는 Amazon EC2 또는 Fargate에서 Amazon ECS를 사용하여 동일한 리전과 us-west-1 리전으로 데이터를 가져옵니다. 리전당 월 50GB를 가져오지만 요금은 us-west-1로의 송신 데이터 전송에만 부과됩니다. 처음 GB는 무료지만 나머지에는 GB당 0.09 USD의 요금이 부과되며 총 데이터 송신 요금은 4.41 USD입니다. **총 비용 = 2 USD + 4.41 USD = 6.41 USD/월**

요금 예제 3: 무료 한도 내의 Amazon ECR 퍼블릭 리포지토리 고객
공개적으로 공유할 총 40GB의 소프트웨어 이미지와 아티팩트를 저장합니다. 무료 한도 내에 있으므로 스토리지 요금이 부과되지 않습니다. 데이터 수신 요금도 부과되지 않습니다. **총 비용 = 0 USD**

요금 예제 4: 무료 한도 내의 Amazon ECR 퍼블릭 리포지토리 익명 개발자
익명의 개발자가 퍼블릭 레지스트리에서 월 300GB를 가져옵니다. 무료 한도 내에 있으므로 데이터 송신 요금이 부과되지 않습니다. **총 비용 = 0 USD**

요금 예제 5: 무료 한도를 초과한 Amazon ECR 퍼블릭 리포지토리 고객
AWS 계정을 사용하여 ECR Public에서 월 6TB의 이미지를 데이터 센터로, 월 8TB의 이미지를 AWS 리전으로 가져옵니다. 데이터 센터로 가져온 처음 5TB는 무료 한도 미만이므로 AWS 이외의 대상으로 전송한 초과 1TB의 데이터(GB당 0.09 USD)에 대한 90 USD만 부과됩니다. AWS 리전으로 전송된 월 8TB는 무료입니다. **총 비용 = 90 USD/월**

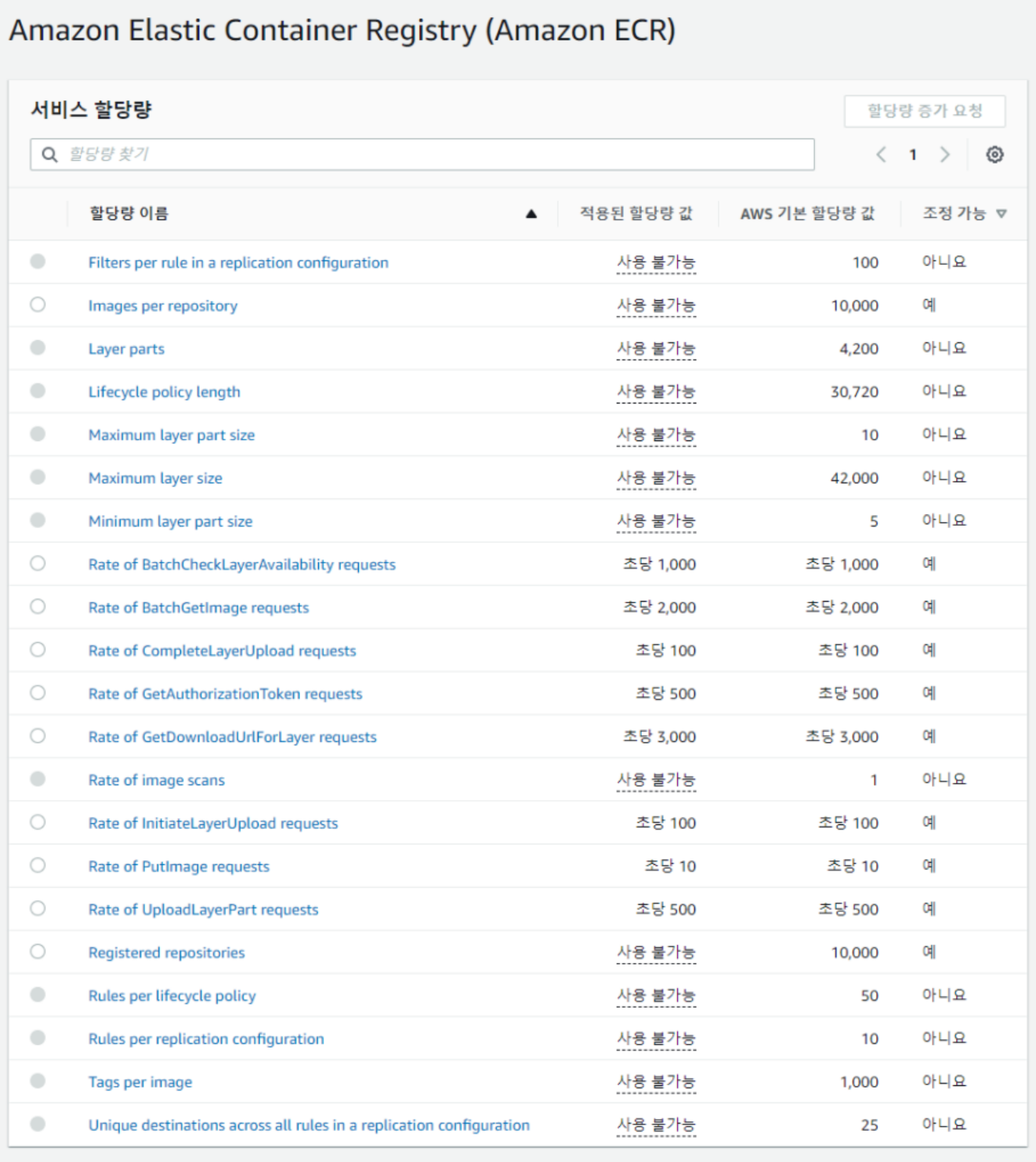
명시된 경우를 제외하고 요금에는 VAT 및 해당 판매세를 비롯한 관련 조세 공과가 포함되지 않습니다. 청구지 주소가 일본으로 되어 있는 고객의 경우 AWS 사용 시 일본 소비세의 적용을 받게 됩니다. 소비세에 대해 자세히 알아보세요.

월 500TB를 초과하는 데이터 전송의 경우 AWS에 문의하세요.

- Docker Hub와 환경 및 비용등 고려 필요
    - https://www.docker.com/pricing/

Service Quotas

https://us-west-1.console.aws.amazon.com/servicequotas/home/services/ecr/quotas



Amazon Elastic Container Registry (Amazon ECR)

서비스 할당량

할당량 증가 요청

할당량 찾기

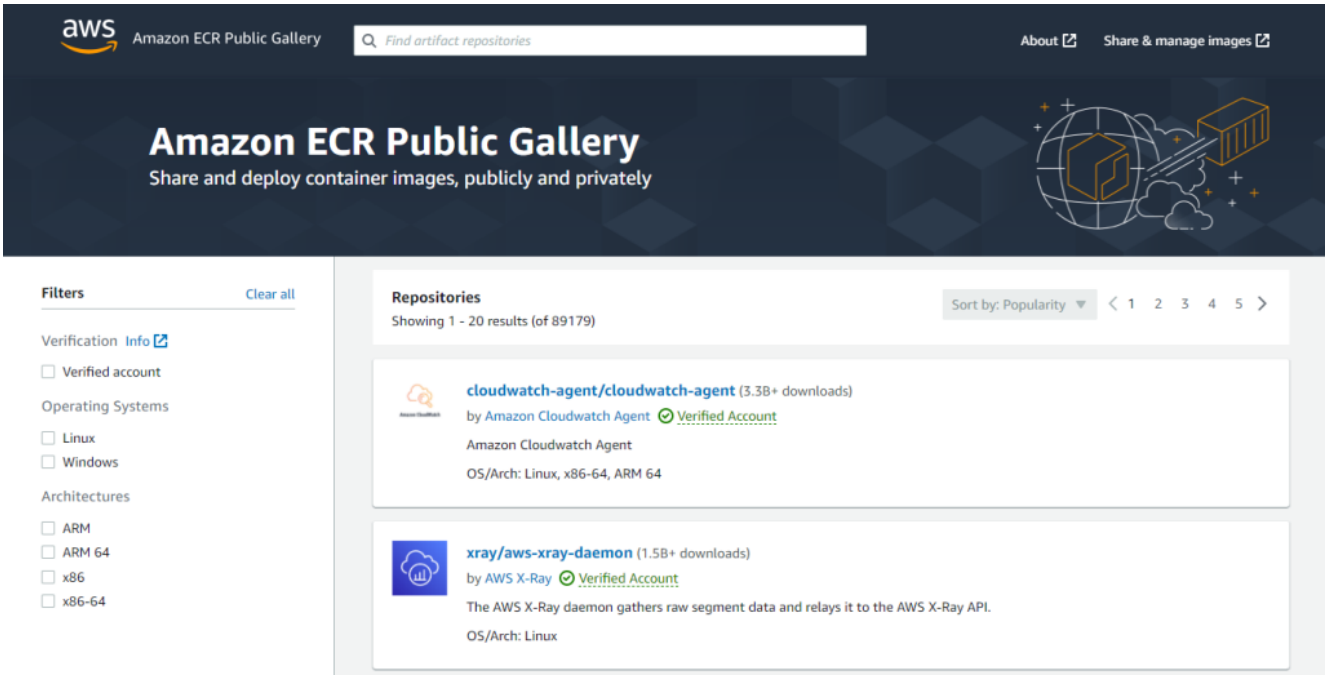
1

할당량 이름	적용된 할당량 값	AWS 기본 할당량 값	조정 가능
Filters per rule in a replication configuration	사용 불가능	100	아니요
Images per repository	사용 불가능	10,000	예
Layer parts	사용 불가능	4,200	아니요
Lifecycle policy length	사용 불가능	30,720	아니요
Maximum layer part size	사용 불가능	10	아니요
Maximum layer size	사용 불가능	42,000	아니요
Minimum layer part size	사용 불가능	5	아니요
Rate of BatchCheckLayerAvailability requests	초당 1,000	초당 1,000	예
Rate of BatchGetImage requests	초당 2,000	초당 2,000	예
Rate of CompleteLayerUpload requests	초당 100	초당 100	예
Rate of GetAuthorizationToken requests	초당 500	초당 500	예
Rate of GetDownloadUrlForLayer requests	초당 3,000	초당 3,000	예
Rate of image scans	사용 불가능	1	아니요
Rate of InitiateLayerUpload requests	초당 100	초당 100	예
Rate of PutImage requests	초당 10	초당 10	예
Rate of UploadLayerPart requests	초당 500	초당 500	예
Registered repositories	사용 불가능	10,000	예
Rules per lifecycle policy	사용 불가능	50	아니요
Rules per replication configuration	사용 불가능	10	아니요
Tags per image	사용 불가능	1,000	아니요
Unique destinations across all rules in a replication configuration	사용 불가능	25	아니요

Amazon ECR Public Gallery

https://gallery.ecr.aws/

Amazon ECR 공개 갤러리는 Amazon ECR 공개 리포지토리에서 호스팅되는 컨테이너 이미지를 찾고 공유할 수 있는 공개 웹 사이트입니다.



사용자 추가

사용자 추가

1 2 3 4 5

사용자 세부 정보 설정

동일한 액세스 유형 및 권한을 사용하여 한 번에 여러 사용자를 추가할 수 있습니다. 자세히 알아보기

사용자 이름* ecr_user_bmt

다른 사용자 추가

AWS 액세스 유형 선택

이러한 사용자가 주로 AWS에 액세스하는 방법을 선택합니다. 프로그래밍 방식의 액세스만 선택하면 사용자가 위임된 역할을 사용하여 콘솔에 액세스하는 것을 방지할 수 없습니다. 액세스 키와 자동 생성된 암호가 마지막 단계에서 제공됩니다. 자세히 알아보기

AWS 자격 증명 유형 선택*

- [x] **액세스 키 – 프로그래밍 방식 액세스**
  AWS API, CLI, SDK 및 기타 개발 도구에 대해 **액세스 키 ID** 및 **비밀 액세스 키** 을(를) 활성화합니다.
- [ ] **암호 – AWS 관리 콘솔 액세스**
  사용자가 AWS Management Console에 로그인할 수 있도록 허용하는 **비밀번호** 을(를) 활성화합니다.

* 필수

취소 다음: 권한

사용자 추가

1 2 3 4 5

권한 설정

그룹에 사용자 추가 | 기존 사용자에서 권한 복사 | 기존 정책 직접 연결

정책 생성

정책 필터 ElasticContainerRegistryPublicFullAccess 1 결과 표시

	정책 이름	유형	사용 용도
[x]	AmazonElasticContainerRegistryPublicFullAccess	AWS 관리형	없음

권한 경계 설정

취소 이전 다음: 태그

사용자 추가

1 2 3 4 5

태그 추가(선택 사항)

IAM 태그는 사용자 사용자에 추가할 수 있는 키-값 페어입니다. 태그는 이메일 주소와 같은 사용자 정보를 포함하거나 직책과 같은 내용일 수 있습니다. 태그를 사용하여 이 사용자에 대한 액세스를 구성, 추적 또는 제어할 수 있습니다. 자세히 알아보기

키	값(선택 사항)	제거
새 키 추가		

50 태그를 더 추가할 수 있습니다.

취소 이전 다음: 검토

사용자 추가

1 2 3 4 5

검토

선택 항목을 검토합니다. 사용자를 생성한 후 자동으로 생성된 비밀번호와 액세스 키를 보고 다운로드할 수 있습니다.

사용자 세부 정보

사용자 이름	ecr_user_bmt
AWS 액세스 유형	프로그래밍 방식 액세스 - 액세스 키 사용
권한 경계	권한 경계가 설정되지 않았습니다

권한 요약

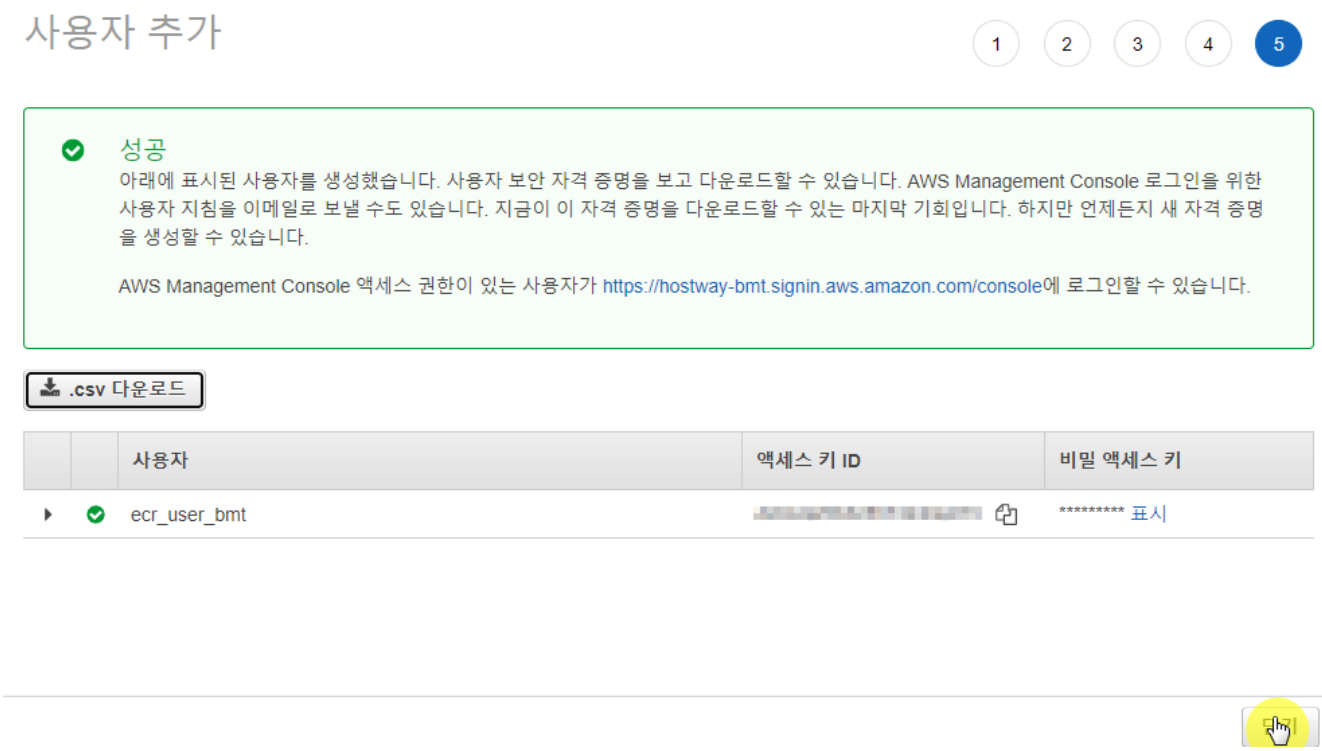
다음 정책이 위에 표시된 사용자에게 연결됩니다.

유형	이름
관리형 정책	AmazonElasticContainerRegistryPublicFullAccess

태그

태그가 추가되지 않았습니다.

취소 이전 사용자 만들기



자격 증명 설정

- *AWS CLI 설치*
  - *https://docs.aws.amazon.com/ko_kr/cli/v1/userguide/cli-chap-install.html*
  - *운영 환경에 맞는 지침 참고*

퍼블릭 리포지토리 생성


Amazon Elastic Container Registry
Private registry
Public registry
Repositories
Getting started
Documentation
Public gallery
Amazon ECR > 리포지토리
Private
Public
퍼블릭 리포지토리
푸시 명령 보기
삭제
편집
리포지토리 생성
리포지토리 찾기
리포지토리 이름
URI
생성 날짜
리포지토리 없음
리포지토리를 찾을 수 없음

Amazon ECR > 리포지토리 > 리포지토리 생성

리포지토리 생성

일반 설정

표시 여부 설정 Info
리포지토리에 대한 가시성 설정을 선택합니다.

- ○ 프라이빗
  액세스는 IAM 및 리포지토리 정책 권한에 의해 관리됩니다.
- ◉ 퍼블릭
  이미지 풀에 대해 공개적으로 표시되고 액세스할 수 있습니다.

ⓘ 리포지토리가 생성되면 해당 리포지토리의 가시성 설정을 변경할 수 없습니다.

세부 정보

리포지토리 이름 Info
리포지토리 이름에 네임스페이스를 넣을 수 있습니다(예: namespace/repo-name).

public.ecr.aws/[redacted]/ ecr-bmt-hostway

최대 205자 중 15자(최소 2자 이상) The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, and forward slashes.

ⓘ 첫 번째 퍼블릭 리포지토리가 생성되면 기본 별칭이 퍼블릭 레지스트리에 연결됩니다. 레지스트리 별칭은 리포지토리 URI의 리포지토리 이름에 접두사로 표시됩니다. 사용자 지정 별칭은 [레지스트리 설정] 페이지에서 요청할 수 있습니다.

리포지토리 로고 - *선택 사항* Info
리포지토리 로고로 사용할 로컬 이미지 파일을 선택합니다.

파일 업로드 재설정

지원되는 파일 형식은 PNG입니다. 지원되는 이미지 크기는 높이와 너비 모두 최소 60픽셀, 최대 2048픽셀이어야 합니다. 최대 파일 크기는 500KB입니다.

간단한 설명 - *선택 사항*
간단한 설명은 검색 결과 및 리포지토리 세부 정보 페이지에 표시됩니다.

ecr-bmt-hostway

최대 255자 중 15자

콘텐츠 유형 - *선택 사항* Info
리포지토리의 이미지와 호환되는 운영 체제와 시스템 아키텍처를 선택합니다.

운영 체제
- ☐ Linux
- ☐ Windows

아키텍처
- ☐ ARM
- ☐ ARM 64
- ☐ x86
- ☐ x86-64

소개 - *선택 사항* Info

예제 보기

리포지토리에 대한 자세한 설명을 제공합니다. 리포지토리에 포함된 항목, 라이선스 세부 정보 또는 기타 관련 정보를 식별합니다.

이 리포지토리 설명

최대 10,240자 중 0자 텍스트에 GitHub Flavored Markdown 형식을 사용합니다. 자세히 알아보기

미리 보기

사용 - *선택 사항* Info

예제 보기

리포지토리에서 이미지를 사용하는 방법에 대한 세부 정보를 제공합니다. 그러면 리포지토리 사용자에 대한 컨텍스트, 지원 정보 및 추가 사용 세부 정보가 제공됩니다.

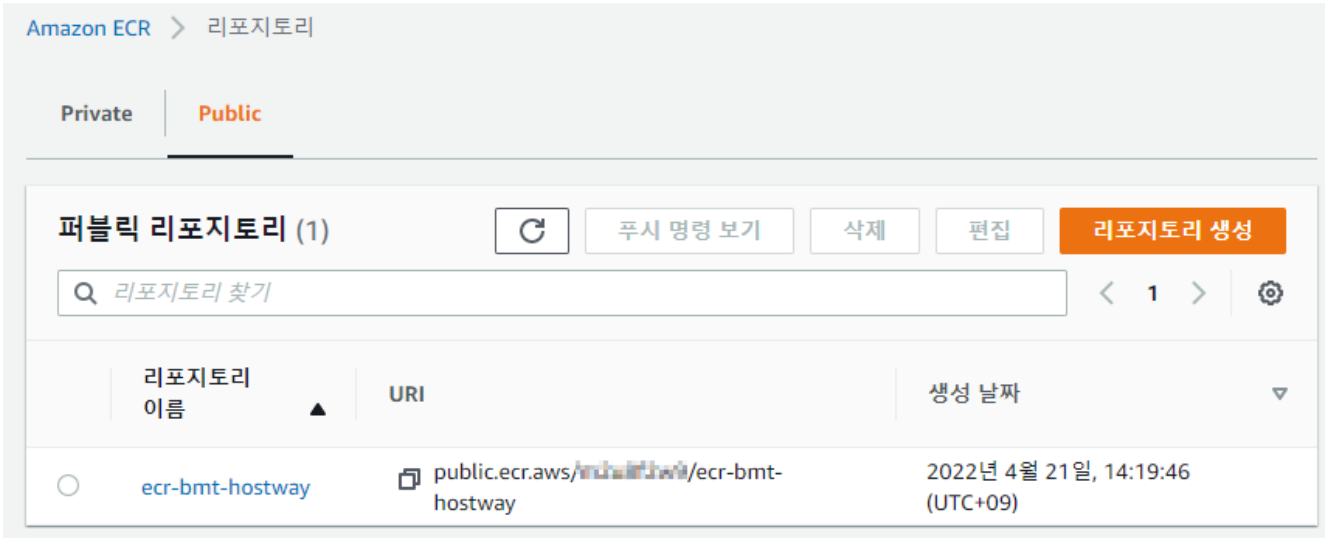
사용 정보

최대 10,240자 중 0자 텍스트에 GitHub Flavored Markdown 형식을 사용합니다. 자세히 알아보기

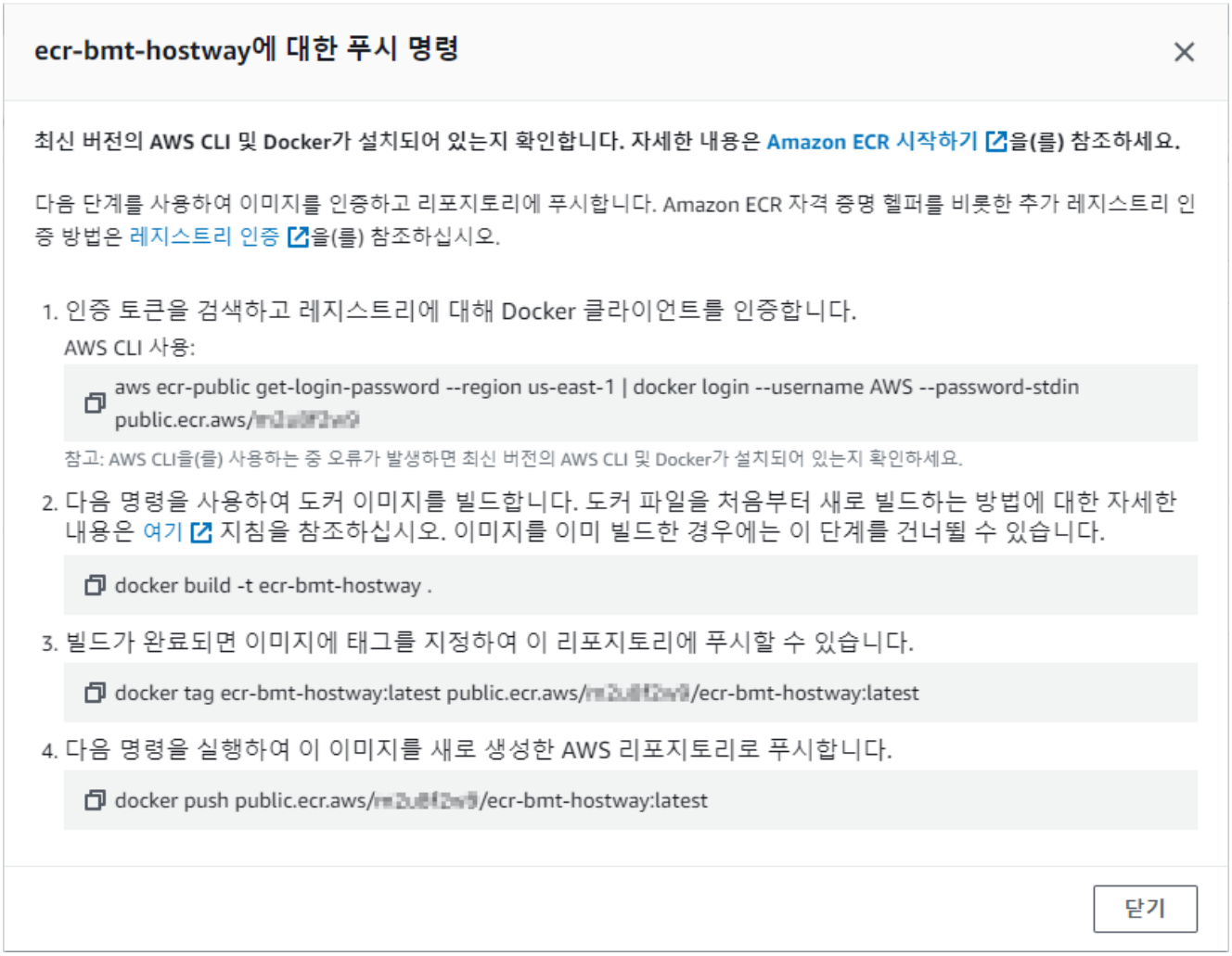
미리 보기

미리 보기

취소 리포지토리 생성


Amazon ECR > 리포지토리
Private
Public
퍼블릭 리포지토리 (1)
푸시 명령 보기
삭제
편집
리포지토리 생성
리포지토리 찾기
1
리포지토리 이름
URI
생성 날짜
ecr-bmt-hostway
public.ecr.aws/[redacted]/ecr-bmt-hostway
2022년 4월 21일, 14:19:46 (UTC+09)

Push


ecr-bmt-hostway에 대한 푸시 명령
최신 버전의 AWS CLI 및 Docker가 설치되어 있는지 확인합니다. 자세한 내용은 Amazon ECR 시작하기 을(를) 참조하세요.
다음 단계를 사용하여 이미지를 인증하고 리포지토리에 푸시합니다. Amazon ECR 자격 증명 헬퍼를 비롯한 추가 레지스트리 인증 방법은 레지스트리 인증 을(를) 참조하십시오.
1. 인증 토큰을 검색하고 레지스트리에 대해 Docker 클라이언트를 인증합니다.
AWS CLI 사용:
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/[redacted]
참고: AWS CLI을(를) 사용하는 중 오류가 발생하면 최신 버전의 AWS CLI 및 Docker가 설치되어 있는지 확인하세요.
2. 다음 명령을 사용하여 도커 이미지를 빌드합니다. 도커 파일을 처음부터 새로 빌드하는 방법에 대한 자세한 내용은 여기 지침을 참조하십시오. 이미지를 이미 빌드한 경우에는 이 단계를 건너뛸 수 있습니다.
docker build -t ecr-bmt-hostway .
3. 빌드가 완료되면 이미지에 태그를 지정하여 이 리포지토리에 푸시할 수 있습니다.
docker tag ecr-bmt-hostway:latest public.ecr.aws/[redacted]/ecr-bmt-hostway:latest
4. 다음 명령을 실행하여 이 이미지를 새로 생성한 AWS 리포지토리로 푸시합니다.
docker push public.ecr.aws/[redacted]/ecr-bmt-hostway:latest
닫기

ecr-public endpoints

- AWS CLI 또는 AWS SDK 사용할 때 권장

지역 이름	지역	끝점	규약
미국 동부(버지니아 북부)	us-east-1	ecr-public.us-east-1.amazonaws.com api.ecr-public.us-east-1.amazonaws.com	HTTPS HTTPS

제공되는 명령어를 참고하여 진행한 예제.

- **인증 토큰을 검색하고 레지스트리에 대해 Docker 클라이언트를 인증**

```
root@hostway-bmt:~# aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/n3n4q5v6
WARNING! Your password will be stored unencrypted in /root/.docker/config.json. Configure a credential helper to remove this warning. See https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

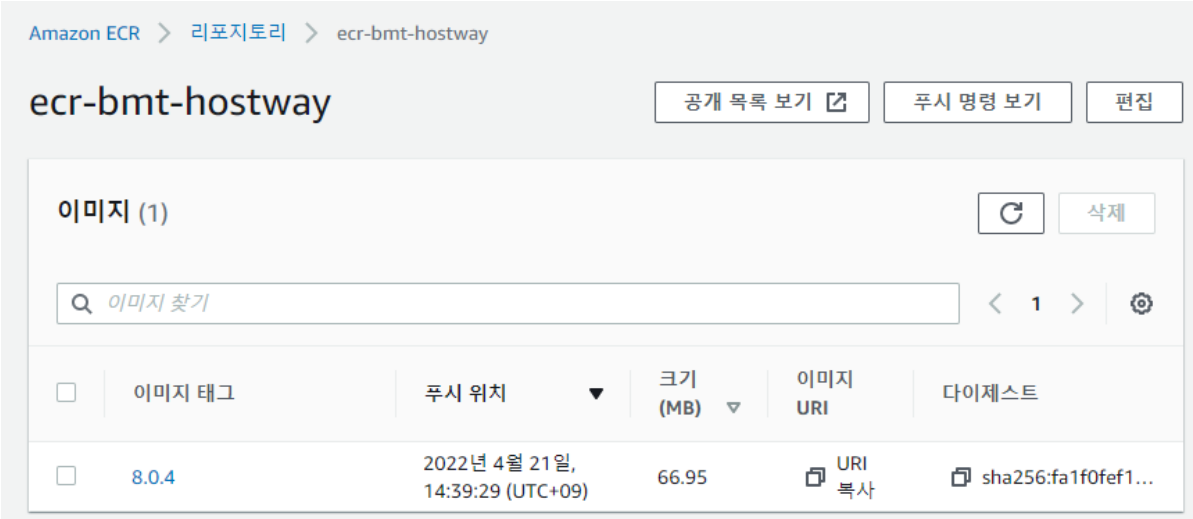
- **이미지에 태그를 지정**

```
root@hostway-bmt:~#  docker  tag  grafana:8.0.4 public.ecr.aws/n3n4q5v6/ecr-bmt-hostway:8.0.4
root@hostway-bmt:~# docker images
REPOSITORY                                    TAG       IMAGE ID       CREATED       SIZE
```

```
grafana                                                 8.0.4
4a61138b1602    2 weeks ago    206MB
public.ecr.aws/n3n4q5v6/ecr-bmt-hostway    8.0.4
4a61138b1602    2 weeks ago    206MB
```

- **AWS 리포지토리로 푸시**

```
root@hostway-bmt:~# docker push
public.ecr.aws/n3n4q5v6/ecr-bmt-hostway:8.0.4
The push refers to repository
[public.ecr.aws/n3n4q5v6/ecr-bmt-hostway]
53fb6eb29c9d: Pushed
1fc7a3b8385e: Pushed
271c2e7e76f9: Pushed
5f70bf18a086: Pushed
9f9cedf089aa: Pushing [============================>
                ]  103.6MB/182.1MB
3c960bff45aa: Pushed
efd53ffe2857:                                   Pushing
[==================================================>]
5.707MB
30a972c8cd3c: Pushed
b2d5eeeaba3a:                                   Pushing
[==================================================>]
5.88MB
8.0.4:                                          digest:
sha256:fa1f0fef19321a5p1apcee215febe02ad3fb1a992a4d5e369
b7f651b3epeb13f size: 2199
```



pull

```
root@hostway-bmt:~# docker pull public.ecr.aws/n3n4q5v6/ecr-bmt-hostway:8.0.4
8.0.4: Pulling from n3n4q5v6/ecr-bmt-hostway
Digest: sha256:fa1f0fef19321a5p1apcee215febe02ad3fb1a992a4d5e369b7f651b3epeb13f
Status: Downloaded newer image for public.ecr.aws/n3n4q5v6/ecr-bmt-hostway:8.0.4
public.ecr.aws/n3n4q5v6/ecr-bmt-hostway:8.0.4
```

[Windows] [오류] 윈도우 원격 데스크

톱 연결 인증 오류 CredSSP 암호화 Oracle 수정 때문일 수 있습니다.

원인

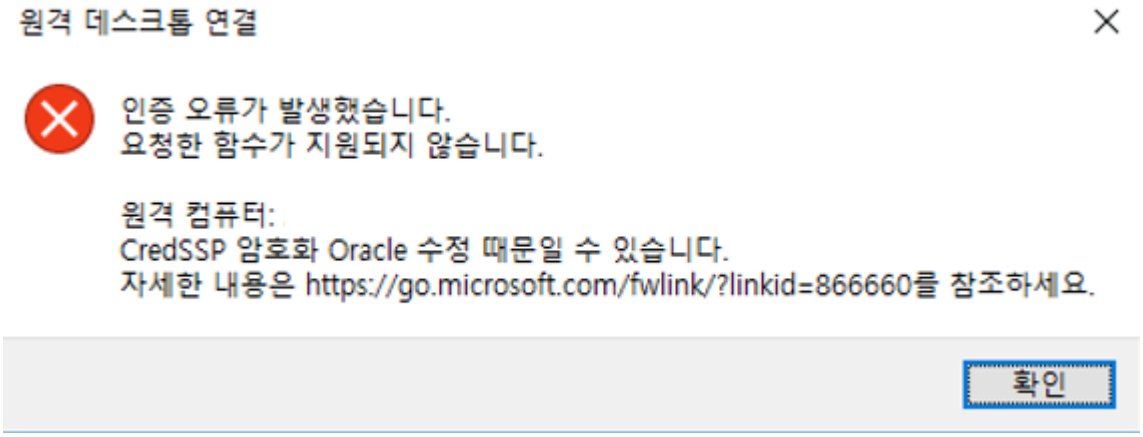
최근 마이크로소프트에서 배포한 보안 업데이트 적용으로 Windows Server에 원격 접근 이슈가 발생
보안 업데이트의 내용은 CredSSP버전에서 원격 코드 실행 취약성이 발견되어 배포되는 부분으로 클라이언트와 서버간의 인증 프로토콜이 강화되는 내용

오류화면

[Window Title]
원격 데스크톱 연결

[Content]
인증 오류가 발생했습니다.
요청한 함수가 지원되지 않습니다.

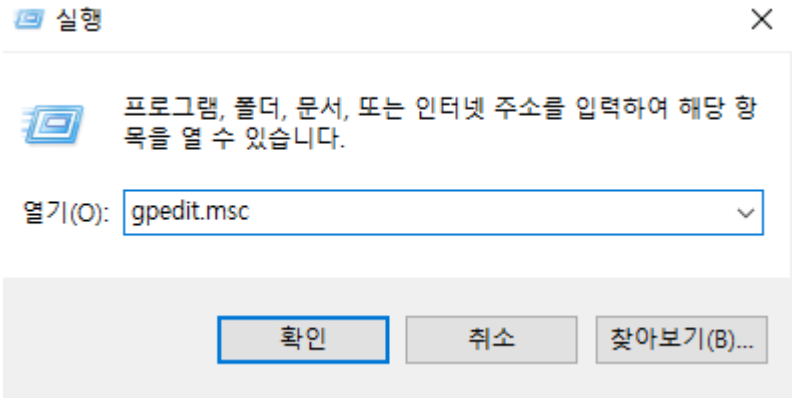
원격 컴퓨터:
CredSSP 암호화 오라클 수정 때문일 수 있습니다.
자세한 내용은 https://go.microsoft.com/fwlink/?linkid=866660를 참조하세요.



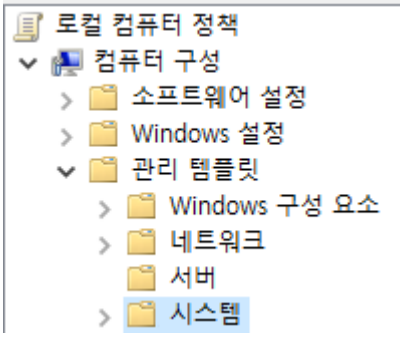
원격 CredSSP 인증 오류 해결 방법 1

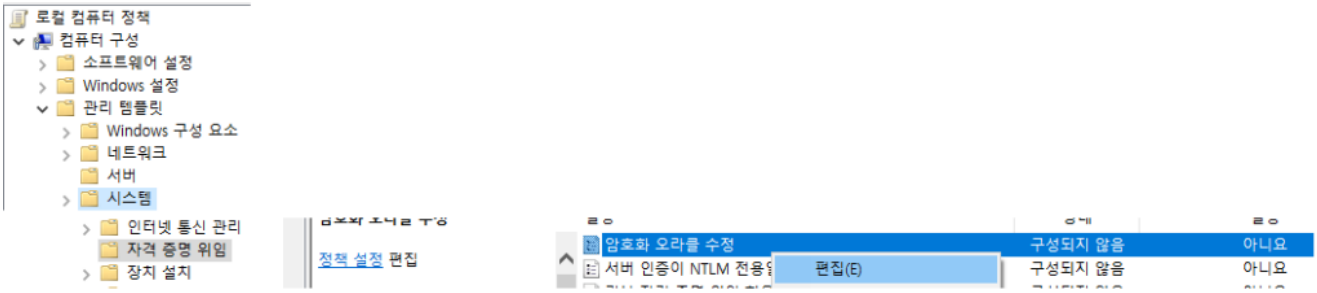
접속하는 Client PC 의 그룹정책 수정으로 강제 적용 방법

Wins + R 단축키를 사용하여 실행 창을 열고 "gpedit.msc"을 입력 후 확인

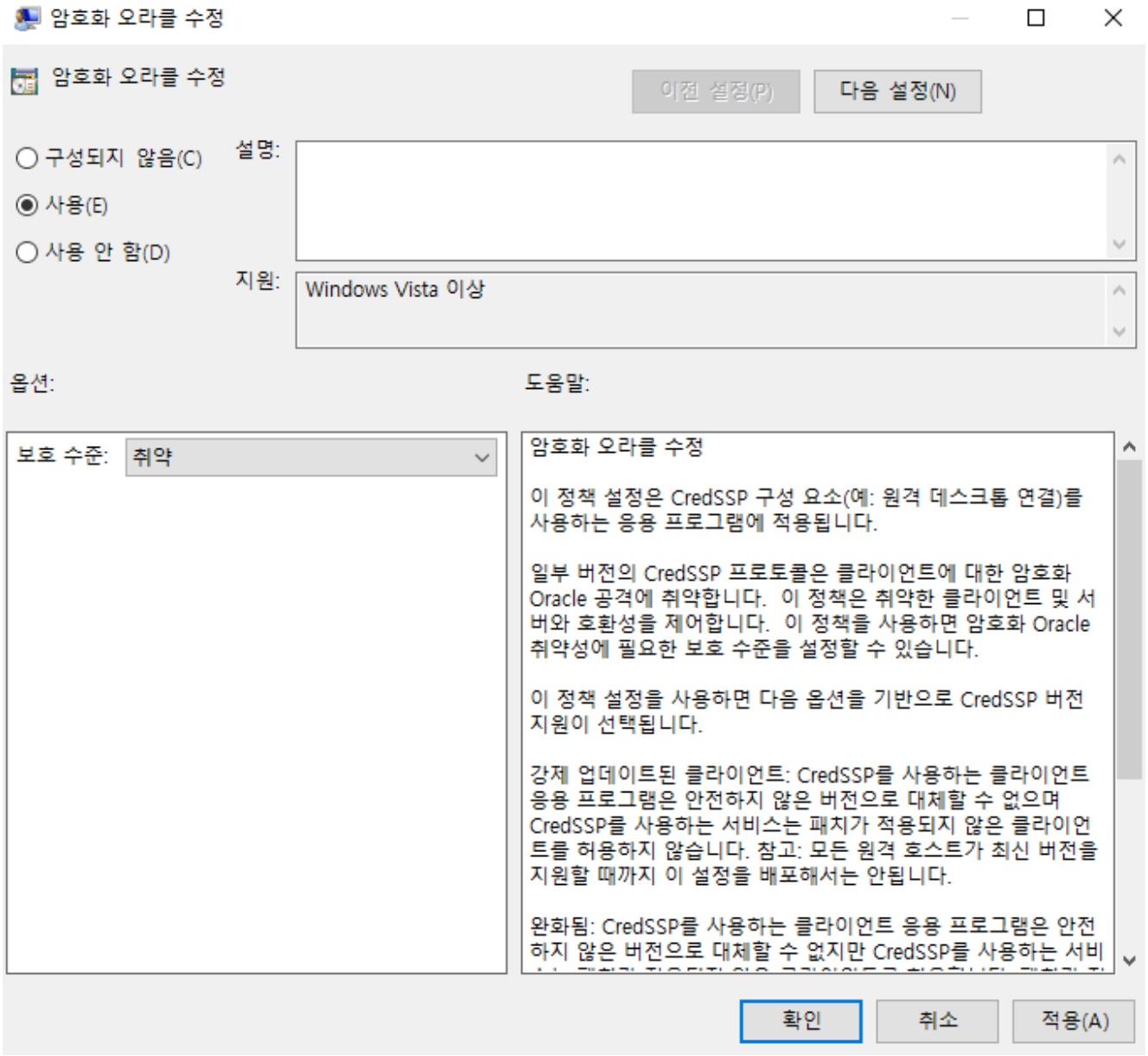


컴퓨터 구성 -> 관리 템플릿 -> 시스템 -> 자격 증명 위임 -> Oracle 수정 암호화 편집 클릭

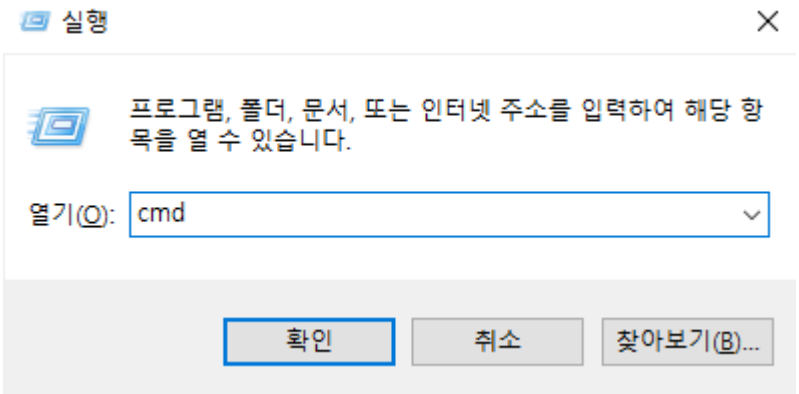




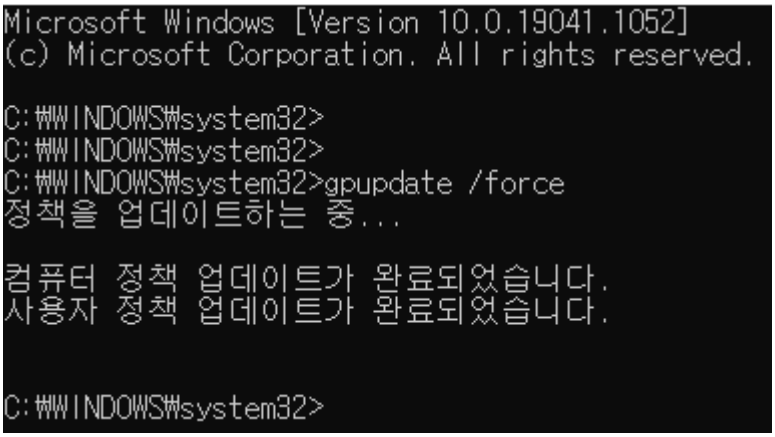
사용 - 보호 수준: "취약" 선택후 확인 클릭



로컬 그룹 정책 편집기 창 종료
윈도우 키 + R 키를 이용하여 실행 창 활성화 => "cmd" 입력 후 엔터



`gpupdate /force` 입력 후 엔터



원격 데스크톱 연결 접속 확인

원격 CredSSP 인증 오류 해결 방법 2

Client: 사용하는 PC 및 노트북 최신 Windows Update 적용
Server: 최신 Windows Update 적용
원격 데스크톱 연결 접속 확인

[Container Runtime] CRI-O 설치 가이드

CRI-O 설치 가이드

CRI-O ?

CRI-O는 OCI 호환 컨테이너 런타임 인터페이스(CRI)다.
가장 일반적으로 사용되는 런타임은 Docker였으나 Kubernetes 1.24 버전부터 Docker용 CRI(컨테이너 런타임 인터페이스) shim이 더 이상 사용되지 않아(Docker Engine은 지원) CRI-O 설치 가이드를 작성한다.

구성 서버 (VM)
Controller Server : 1EA
Worker Server : 1EA

OS
Ubuntu 20.04 Server Minimal

지원되는 버전 확인 : https://github.com/cri-o/cri-o/blob/main/install.md#supported-versions
쿠버네티스 설치 참조 : https://tech.hostway.co.kr/2022/02/06/418/

Traffic Setup

```
# .conf 파일을 만들어 부팅 시 모듈을 로드한다

# Controller / Worker

cat <<EOF | sudo tee /etc/modules-load.d/crio.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# 요구되는 sysctl 파라미터 설정, 이 설정은 재부팅 간에도 유지된다.

# Controller / Worker

cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF

sudo sysctl --system
```

cgroup driver

```
# CRI-O는 기본적으로 systemd cgroup 드라이버를 사용한다.

cgroupfs cgroup 드라이버로 전환하려면, /etc/crio/crio.conf 를 수정하거나 /etc/crio/crio.conf.d/02-cgroup-manager.conf 에 드롭-
```

인(drop-in) 구성을 배치한다.

```
[crio.runtime]
conmon_cgroup = "pod"
cgroup_manager = "cgroupfs"
```

CRI-O 설치

```
# Controller / Worker

sudo -i
export OS=xUbuntu_20.04 # OS 버전
export VERSION=1.19     # cri-o 버전

echo "deb https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/$OS/ /" > /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list
echo "deb http://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-o:/$VERSION/$OS/ /" > /etc/apt/sources.list.d/devel:kubic:libcontainers:stable:cri-o:$VERSION.list

curl -L https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable:cri-o:$VERSION/$OS/Release.key | apt-key add -
curl -L https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/$OS/Release.key | apt-key add -

sudo apt-get update

sudo apt-get install cri-o cri-o-runc

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl status crio
```

Controller Node Initialize

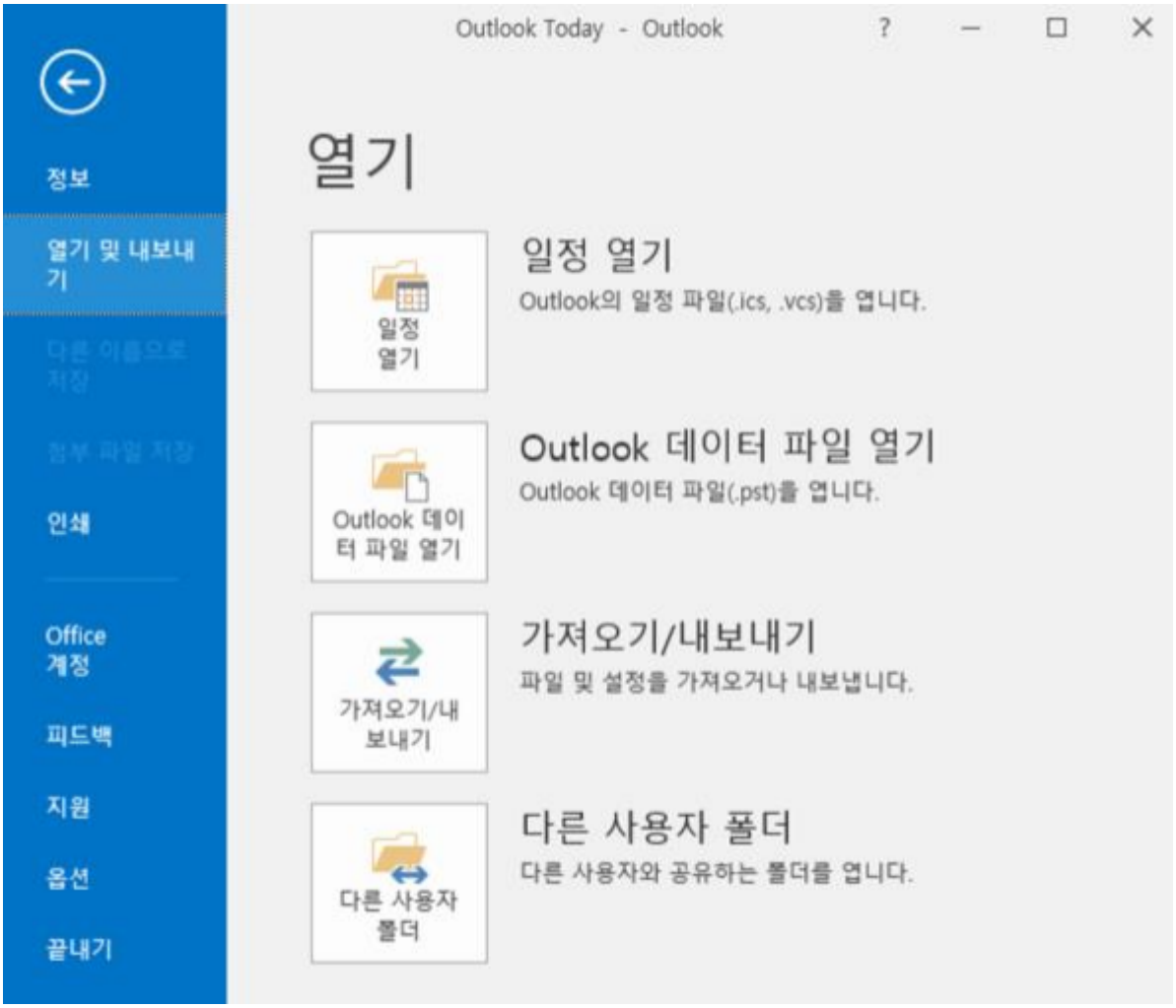
```
# docker의 경우 여러 런타임이 설치되어있어도 1순위로 감지되어 구성하나 CRI-O의 경우 따로 지정해주어야 함

# Controller

kubeadm init --cri-socket=/var/run/crio/crio.sock --ignore-preflight-errors=all --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=203.248.23.192

# Worker

kubeadm join 203.248.23.192:6443 --token 9oy7rn.qtw04nfd8ga417p4 --discovery-token-ca-cert-hash sha256:26218e0c80320b7c23735916c130fc48f644b26314212d917969553ec0651256
```

나머지 설치 단계는 쿠버네티스 설치 링크를 참조한다.

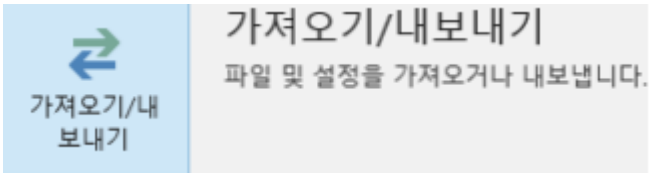
Outlook 2016 PST 백업 및 복원 가이드

Outlook PST 백업하기

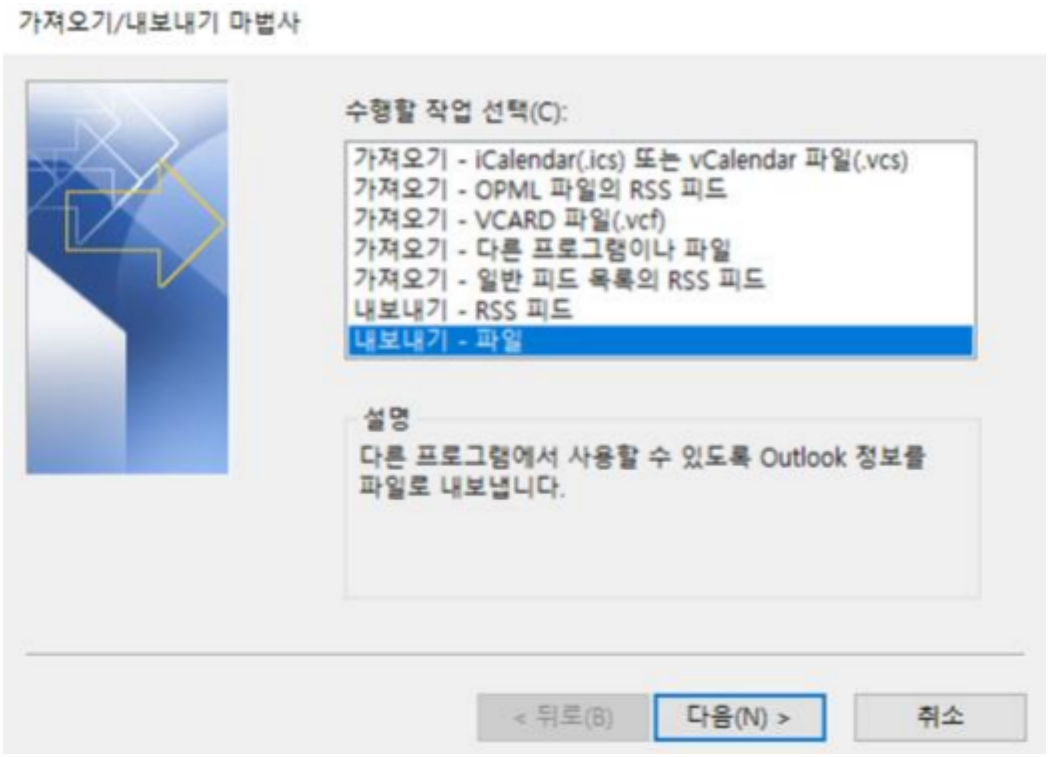
1. Outlook 파일 -> 열기 및 내보내기를 클릭합니다.



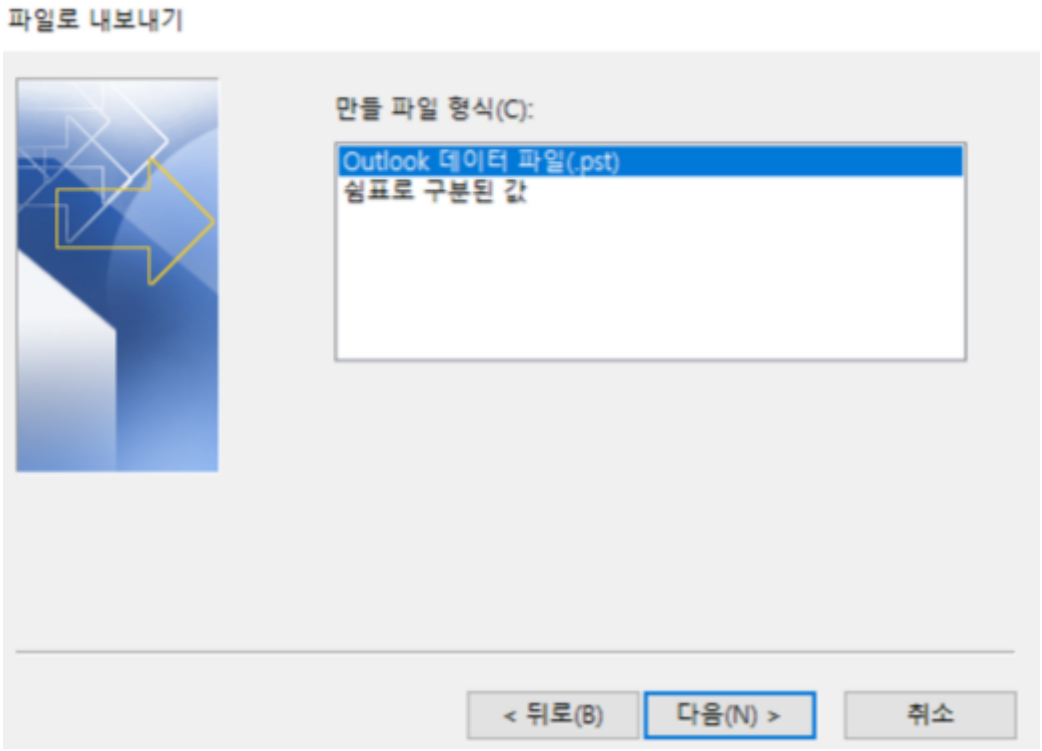
2. 가져오기/내보내기를 클릭합니다.



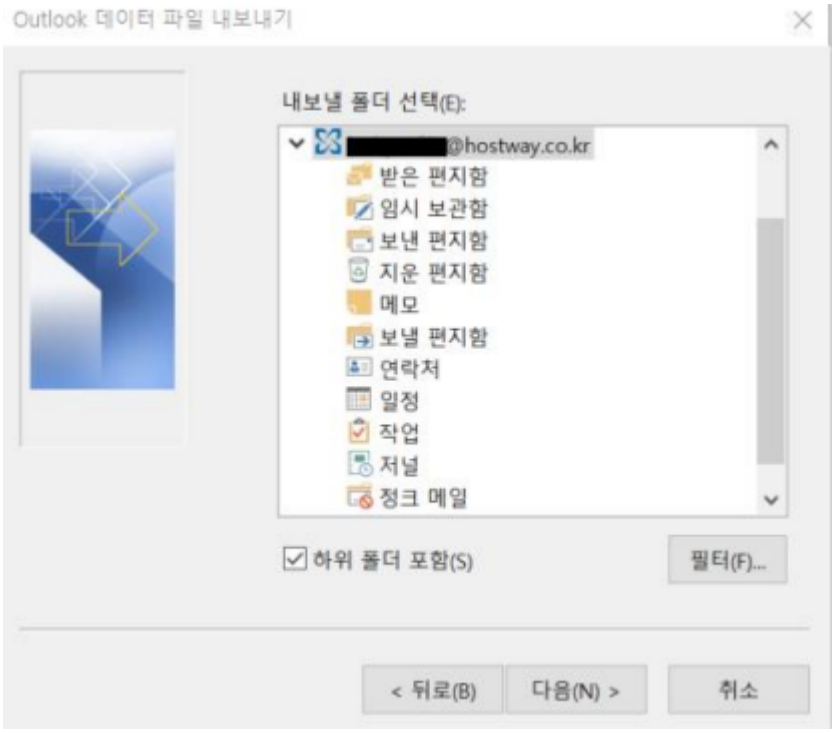
3. 내보내기 – 파일을 선택하고 다음(N)을 클릭합니다.



4. Outlook 데이터 파일(.pst)를 선택하고 다음(N)을 클릭합니다.

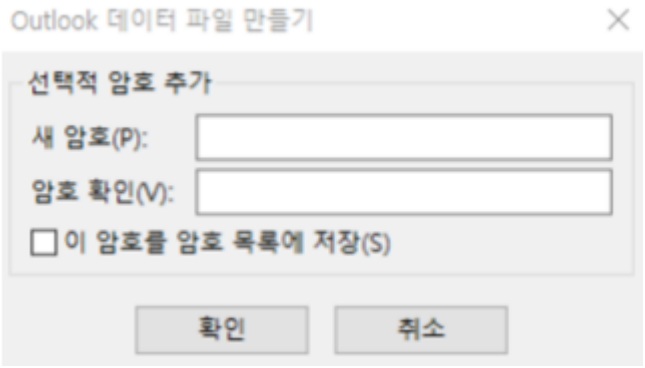


5. 백업하고 싶은 상위 디렉토리를 선택하고 다음(N)을 클릭합니다.
(하위 폴더 포함(S)을 체크하도록 하며, 메일 계정을 선택해야 계정의 전체 데이터가 백업됩니다.)

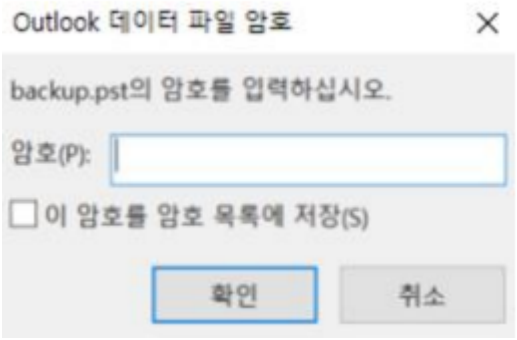


6. 저장할 위치를 선택하고 마침을 클릭합니다.

7. 저장할 파일의 암호를 지정하거나 필요 없다면 미 입력 후 확인을 클릭합니다.

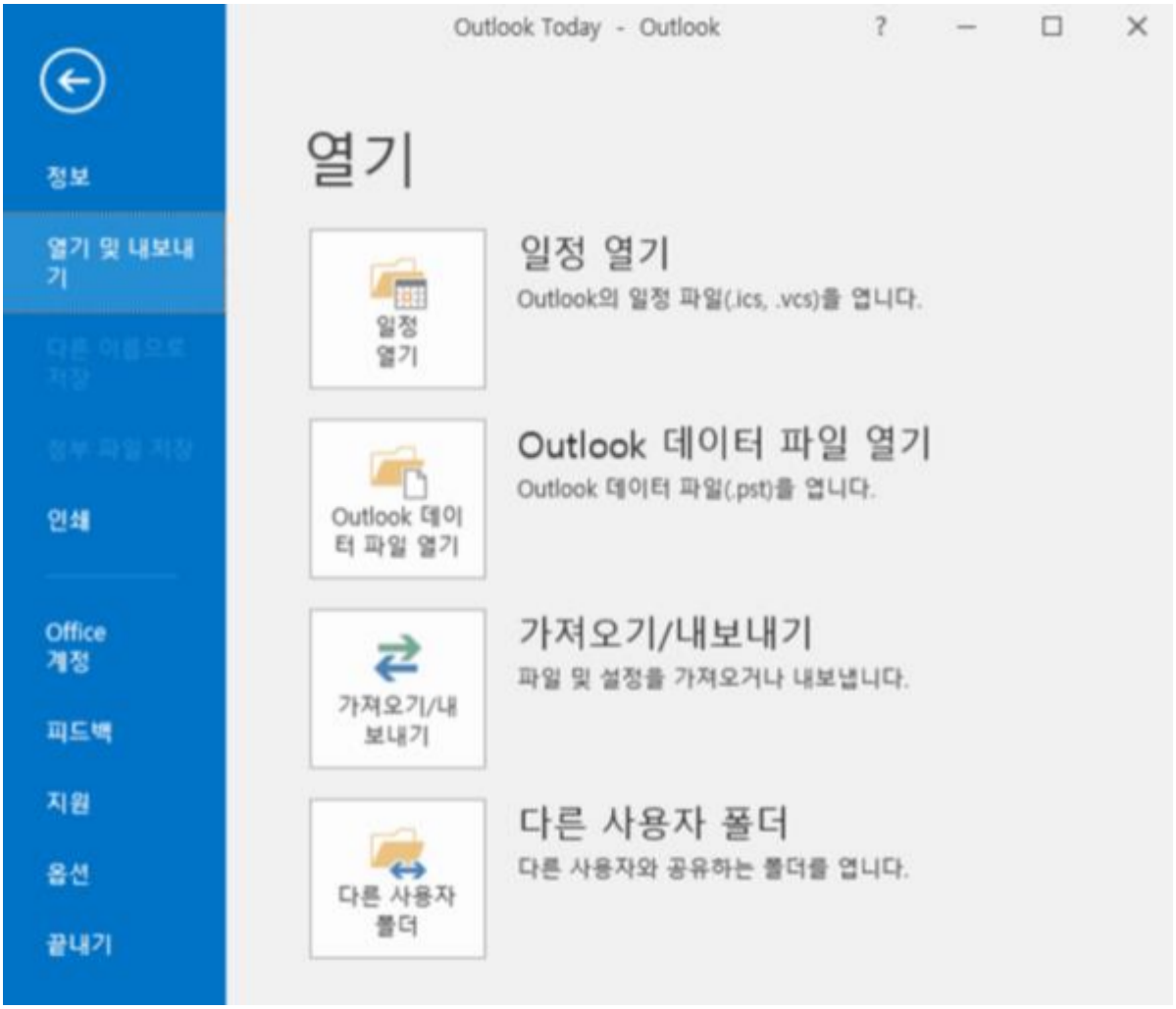


8. 암호를 입력하였다면 한번 더 입력하여 줍니다.

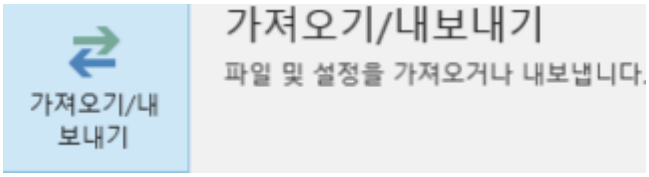


Outlook PST 복원하기

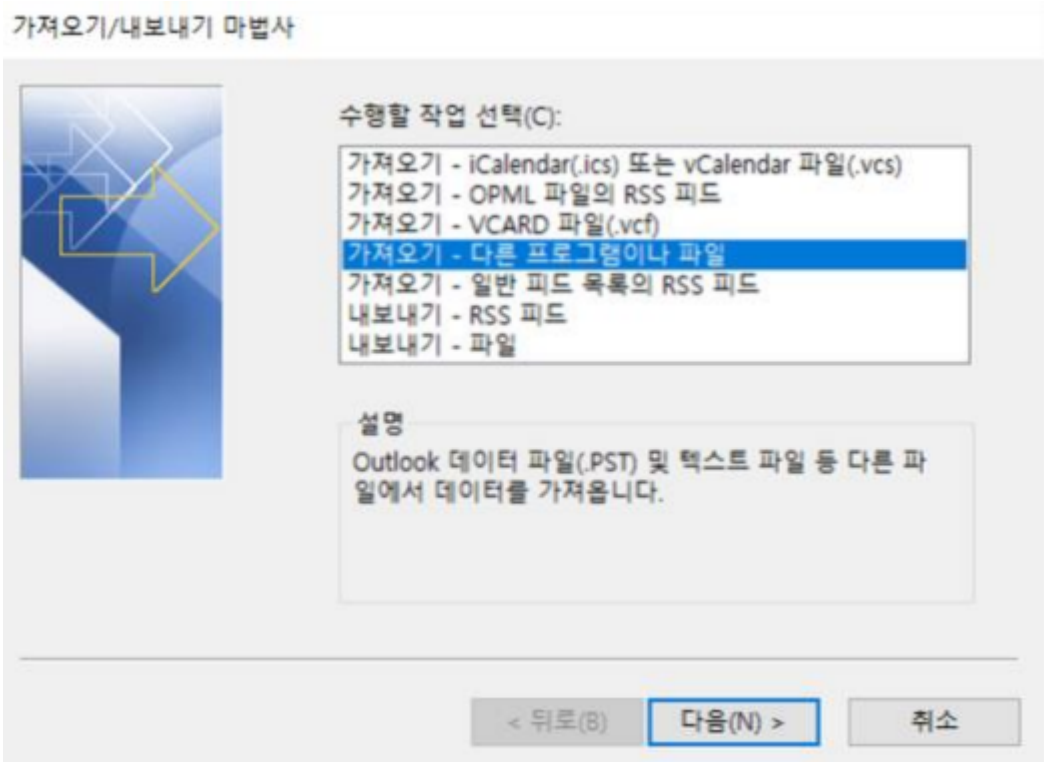
1. Outlook 파일 -> 열기 및 내보내기를 클릭합니다.



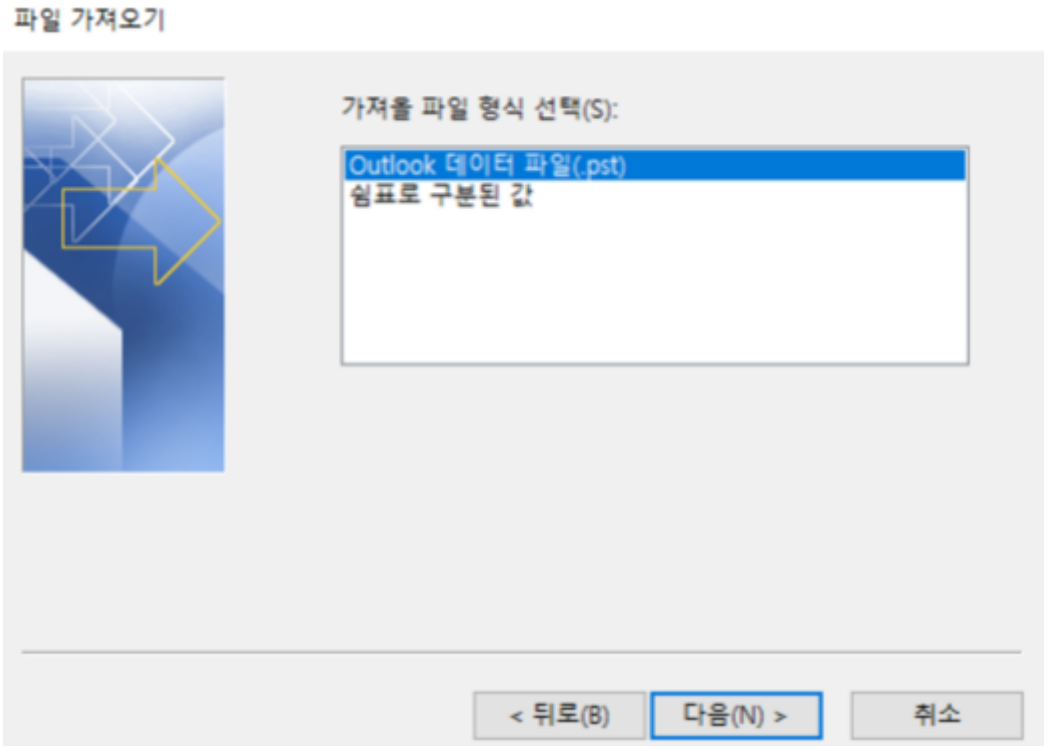
2. 가져오기/내보내기를 클릭합니다.



3. 가져오기 – 다른 프로그램이나 파일을 선택하고 다음(N)을 클릭합니다.



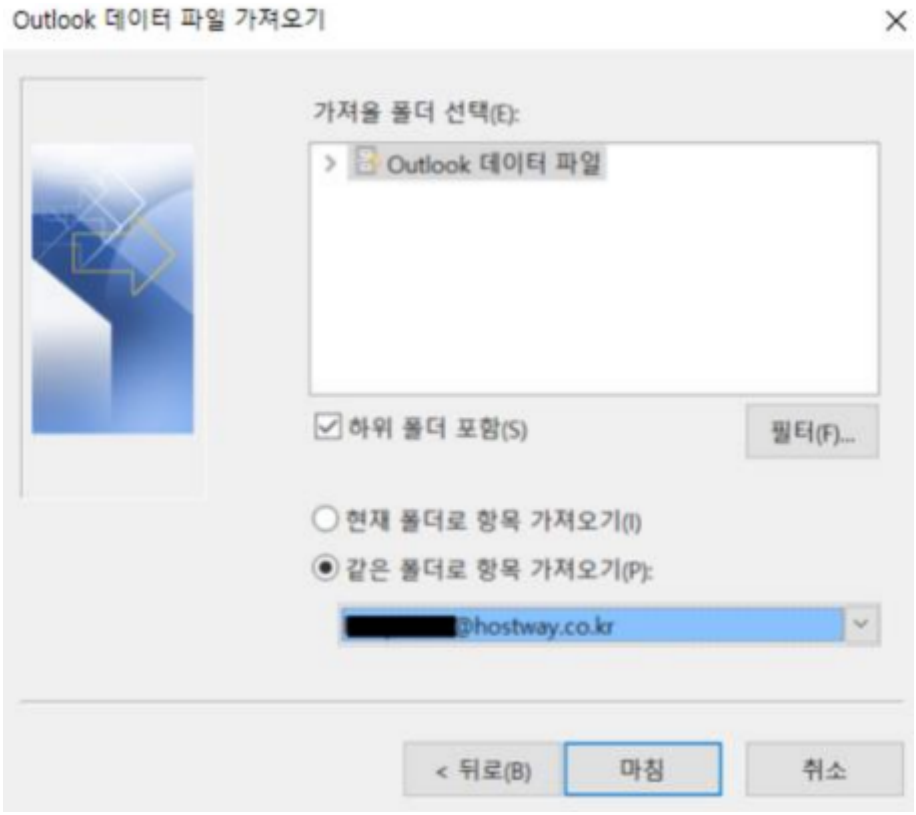
4. Outlook 데이터 파일(.pst)를 선택하고 다음(N)을 클릭합니다.



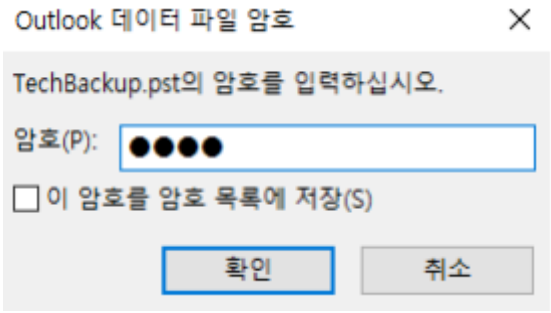
5. 백업 받았던 파일(.pst)을 지정하고 다음(N)을 클릭합니다.

6. 가져올 계정 및 폴더를 선택하고 마침을 클릭합니다.

7. 백업 시 암호를 지정하였다면 암호를 입력합니다.



8. Outlook에서 복원된 데이터를 확인합니다.



기타

※ 해당 문서는 Outlook 2016을 기준으로 작성 되었으며
Outlook 하위 버전에서도 백업 및 복원 방법이 크게 다르지 않습니다.

[Windows] Windows 랜섬웨어 &

SMB 취약점 포트 차단

[Windows] Windows 랜섬웨어 & SMB 취약점 포트 차단

SMB

네트워크에 연결된 컴퓨터끼리 파일, 프린터, 포트 또는 기타 메시지를 전달하는데 사용된다.
SMB 프로토콜 원격코드 실행 취약점은 랜섬웨어 공격으로 많이 사용되고 있다.

OS 최신 보안 패치

MS에서 SMB 관련 취약점에 대한 패치는 보안 업데이트로 제공된다.
업데이트가 어려운 서버는 아래 SMB 포트 차단으로 조치 가능하다.

SMB 포트 차단

Powershell ISE에서 아래 내용 복사 후 실행

```
# updated by Hostway System Team


$Language = Get-WinUserLanguageList
$Lang = $Language.LanguageTag

Switch($Lang)
{
‘ko’
{
Write ‘OS Version’
[Environment]::OSVersion
Write-Verbose -Message “취약점 포트 제거 UDP 137, UDP 138, TCP 139, TCP 445″ -Verbose

netsh advfirewall firewall set rule name=”네트워크 검색(NB-Name-In)” profile=public dir=in localport=137 protocol=udp new enable=no
```

```
netsh advfirewall firewall set rule name="네트워크 검색(NB-Name-In)" profile=private dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Name-In)" profile=public,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Name-In)" profile=private,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Name-In)" profile=private,public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Name-In)" profile=any dir=in localport=137 protocol=udp new enable=no

netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Name-In)" profile=public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Name-In)" profile=private dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Name-In)" profile=public,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Name-In)" profile=private,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Name-In)" profile=private,public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Name-In)" profile=any dir=in localport=137 protocol=udp new enable=no

netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=private dir=in localport=137 protocol=udp new enable=no
```

```
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=public,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=private,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=private,public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=any dir=in localport=137 protocol=udp new enable=no

netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Datagram-In)" profile=public dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Datagram-In)" profile=private dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Datagram-In)" profile=public,domain dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Datagram-In)" profile=private,domain dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Datagram-In)" profile=private,public dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Datagram-In)" profile=any dir=in localport=138 protocol=udp new enable=no

netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=public dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=private dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=public,domain dir=in localport=138 protocol=udp new enable=no
```

```
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=private,domain dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=private,public dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name="네트워크 검색(NB-Datagram-In)" profile=any dir=in localport=138 protocol=udp new enable=no

netsh advfirewall firewall set rule name="가상 컴퓨터 모니터링(NB-Session-In)" profile=public dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="가상 컴퓨터 모니터링(NB-Session-In)" profile=private dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="가상 컴퓨터 모니터링(NB-Session-In)" profile=public,domain dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="가상 컴퓨터 모니터링(NB-Session-In)" profile=private,domain dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="가상 컴퓨터 모니터링(NB-Session-In)" profile=private,public dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="가상 컴퓨터 모니터링(NB-Session-In)" profile=any dir=in localport=139 protocol=tcp new enable=no

netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Session-In)" profile=public dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Session-In)" profile=private dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Session-In)" profile=public,domain dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="파일 및 프린터 공유(NB-Session-In)" profile=private,domain dir=in localport=139 protocol=tcp new enable=no
```

```
netsh advfirewall firewall set rule name=”파일 및 프린터 공유(NB-Session-In)” profile=private,public dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”파일 및 프린터 공유(NB-Session-In)” profile=any dir=in localport=139 protocol=tcp new enable=no

netsh advfirewall firewall set rule name=”파일 및 프린터 공유(SMB-In)” profile=public dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”파일 및 프린터 공유(SMB-In)” profile=private dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”파일 및 프린터 공유(SMB-In)” profile=public,domain dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”파일 및 프린터 공유(SMB-In)” profile=private,domain dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”파일 및 프린터 공유(SMB-In)” profile=private,public dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”파일 및 프린터 공유(SMB-In)” profile=any dir=in localport=445 protocol=tcp new enable=no
}
‘en-US’
{
Write ‘OS Version’
[Environment]::OSVersion
Write-Verbose -Message “Vulnerability Port Removal UDP 137, UDP 138, TCP 139, TCP 445″ -Verbose

netsh advfirewall firewall set rule name=”Network Discovery (NB-Name-In)” profile=public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Name-In)” profile=private dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Name-In)” profile=public,domain dir=in localport=137
```

```
protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Name-In)” profile=private,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Name-In)” profile=private,public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Name-In)” profile=any dir=in localport=137 protocol=udp new enable=no

netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Name-In)” profile=public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Name-In)” profile=private dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Name-In)” profile=public,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Name-In)” profile=private,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Name-In)” profile=private,public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Name-In)” profile=any dir=in localport=137 protocol=udp new enable=no

netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=private dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=public,domain dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=private,domain dir=in localport=137
```

```
protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=private,public dir=in localport=137 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=any dir=in localport=137 protocol=udp new enable=no

netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Datagram-In)” profile=public dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Datagram-In)” profile=private dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Datagram-In)” profile=public,domain dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Datagram-In)” profile=private,domain dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Datagram-In)” profile=private,public dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”File and Printer Sharing (NB-Datagram-In)” profile=any dir=in localport=138 protocol=udp new enable=no

netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=public dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=private dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=public,domain dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=private,domain dir=in localport=138 protocol=udp new enable=no
netsh advfirewall firewall set rule name=”Network Discovery (NB-Datagram-In)” profile=private,public dir=in localport=138
```

```
protocol=udp new enable=no
netsh advfirewall firewall set rule name="Network Discovery (NB-Datagram-In)" profile=any dir=in localport=138 protocol=udp new enable=no

netsh advfirewall firewall set rule name="Virtual Machine Monitoring (NB-Session-In)" profile=public dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="Virtual Machine Monitoring (NB-Session-In)" profile=private dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="Virtual Machine Monitoring (NB-Session-In)" profile=public,domain dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="Virtual Machine Monitoring (NB-Session-In)" profile=private,domain dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="Virtual Machine Monitoring (NB-Session-In)" profile=private,public dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="Virtual Machine Monitoring (NB-Session-In)" profile=any dir=in localport=139 protocol=tcp new enable=no

netsh advfirewall firewall set rule name="File and Printer Sharing (NB-Session-In)" profile=public dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="File and Printer Sharing (NB-Session-In)" profile=private dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="File and Printer Sharing (NB-Session-In)" profile=public,domain dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="File and Printer Sharing (NB-Session-In)" profile=private,domain dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="File and Printer Sharing (NB-Session-In)" profile=private,public dir=in localport=139 protocol=tcp new enable=no
netsh advfirewall firewall set rule name="File and Printer Sharing (NB-Session-In)" profile=any dir=in localport=139
```

```
protocol=tcp new enable=no

netsh advfirewall firewall set rule name=”File Server Remote Management (SMB-In)” profile=public dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”File Server Remote Management (SMB-In)” profile=private dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”File Server Remote Management (SMB-In)” profile=public,domain dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”File Server Remote Management (SMB-In)” profile=private,domain dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”File Server Remote Management (SMB-In)” profile=private,public dir=in localport=445 protocol=tcp new enable=no
netsh advfirewall firewall set rule name=”File Server Remote Management (SMB-In)” profile=any dir=in localport=445 protocol=tcp new enable=no
}
}
```

[Network] K8S Overlay Network 변경 실습 (IPIP -> VXLAN)

K8S Overlay Network 의 변경

IPIP -> VXLAN 변경 하기

주의) 다른 방식의 네트워크에서 생성된 서비스 POD가 있을 경우 프로덕션 레벨에서는 변경이 사실상 어렵다.
(pod 통신의 문제 발생에 대한 트러블슈팅이나 보장을 할 수가 없다)

Calico IP-IP Network 제거 후 VXLAN 으로 재 설치

Node : Controller / Worker01 / Worker02

```
## Controller
# Mode 확인 시 IPIPMODE 인 것을 확인할 수 있다
calicoctl get ippool -o wide
NAME                  CIDR              NAT    IPIPMODE   VXLANMODE   DISABLED   DISABLEBGPEXPORT   SELECTOR
default-ipv4-ippool   192.168.0.0/16   true   Always     Never       false      false
 all()

# 설치 Manifest YAML 을 이용하여 삭제한다.
kubectl delete -f calico.yml

## Contoller / Worker 공통
# 가비지 삭제 후 리부팅하면 tunl0 인터페이스가 사라진 것을 확인할 수 있다.
sudo rm -rf /var/run/calico/
sudo rm -rf /var/lib/calico/
sudo rm -rf /etc/cni/net.d/
sudo rm -rf /var/lib/cni/
sudo reboot


## Controller
# Manifest. calico.yaml 파일을 VXLAN 설치 타입으로 수정한다.

 livenessProbe:
            exec:
              command:
              - /bin/calico-node
              - -felix-live
              # - -bird-live        // VXLAN 은 bird(BGP) 를 사용하지 않으므로 주석 처리 필수
            periodSeconds: 10
            initialDelaySeconds: 10
            failureThreshold: 6
            timeoutSeconds: 10
          readinessProbe:
            exec:
```

```
              command:
              - /bin/calico-node
              - -felix-ready
             # - -bird-ready    // 주석 필수


            # Enable IPIP
            - name: CALICO_IPV4POOL_IPIP
              value: "Never"       // Always --> Never 로 변경한
다
            # Enable or Disable VXLAN on the default IP pool.
            - name: CALICO_IPV4POOL_VXLAN
              value: "Always"         // Never --> Always 로 변
경한다

kind: ConfigMap
apiVersion: v1
metadata:
  name: calico-config
  namespace: kube-system
data:
  # Typha is disabled.
  typha_service_name: "none"
  # Configure the backend to use.
  calico_backend: "vxlan"        // "bird" --> "vxlan" 으로 변경
한다.


# 설치
kubectl apply -f calico.yaml

# Calico Node 체크. 모두 Ready 여야 한다. 조금 기다려보자.
kubectl get nodes -o wide -A

# Calico Pod 체크. kube-system PoD 에 에러가 있으면 안된다.
kubectl get pod -o wide -A

# Calico Type 체크. BIRD 는 꺼져야 정상
sudo calicoctl node status
Calico process is running.
The BGP backend process (BIRD) is not running.
```

```
# Network 체크 VXLANMODE 가 활성화되었다.
calicoctl get ippool -o wide
NAME                  CIDR              NAT    IPIPMODE
VXLANMODE   DISABLED   DISABLEBGPEXPORT   SELECTOR
default-ipv4-ippool    192.168.0.0/16    true    Never
Always        false        false
 all()

# 각 노드들의 tunl0 인터페이스가 사라지고 vxlan 인터페이스가 생성되었다.
# 노드의 vxlan 인터페이스 간 라우팅이 연결된 부분을 확인 가능하다.

hostway@controller:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric
Ref    Use Iface
0.0.0.0         10.10.10.1      0.0.0.0         UG    0      0
0 ens18
10.10.10.0      0.0.0.0         255.255.255.0   U     0      0
0 ens18  // External (SNAT)
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0
0 docker0 // Container Runtime Bridge
192.168.5.0     192.168.5.0     255.255.255.192 UG    0      0
0 vxlan.calico  // Worker01의 인터페이스 정보
192.168.30.64   192.168.30.64   255.255.255.192 UG    0      0
0 vxlan.calico  // Worker02의 인터페이스 정보
192.168.49.0    0.0.0.0         255.255.255.192 U     0      0
0 *              // Controller 의 vxlan 대역
192.168.49.1    0.0.0.0         255.255.255.255 UH    0      0
0 cali09ae4a7064b   // Node(Worker01)가 도달할 GW
192.168.49.2    0.0.0.0         255.255.255.255 UH    0      0
0 cali1fdac863dc5  // Node(Worker02)가 도달할 GW

# Worker 의 정보를 알고 있다.
hostway@controller:~$ ip nei | grep vxlan
192.168.5.0  dev  vxlan.calico  lladdr  66:8c:33:86:44:ce
PERMANENT
192.168.30.64  dev  vxlan.calico  lladdr  66:fb:72:20:22:a1
PERMANENT

# VXLAN Traffic Port 는 UDP 이다.
udp        0      0 0.0.0.0:4789            0.0.0.0:*
```

PoD 생성 시 특이사항 없음

```
hostway@controller:~$ kubectl create deployment sampleos --
image=gcr.io/google-samples/kubernetes-bootcamp:v1    --
replicas=3
deployment.apps/sampleos created
hostway@controller:~$ kubectl get pod -o wide
NAME                          READY   STATUS    RESTARTS   AGE
IP              NODE       NOMINATED NODE   READINESS GATES
sampleos-646dc9654b-8xjw9   1/1     Running   0          45s
192.168.5.11    worker01   <none>           <none>
sampleos-646dc9654b-gxn75   1/1     Running   0          45s
192.168.5.10    worker01   <none>           <none>
sampleos-646dc9654b-snkxg   1/1     Running   0          45s
192.168.30.75   worker02   <none>           <none>
```

VXLAN 패킷 확인하기

// Controller

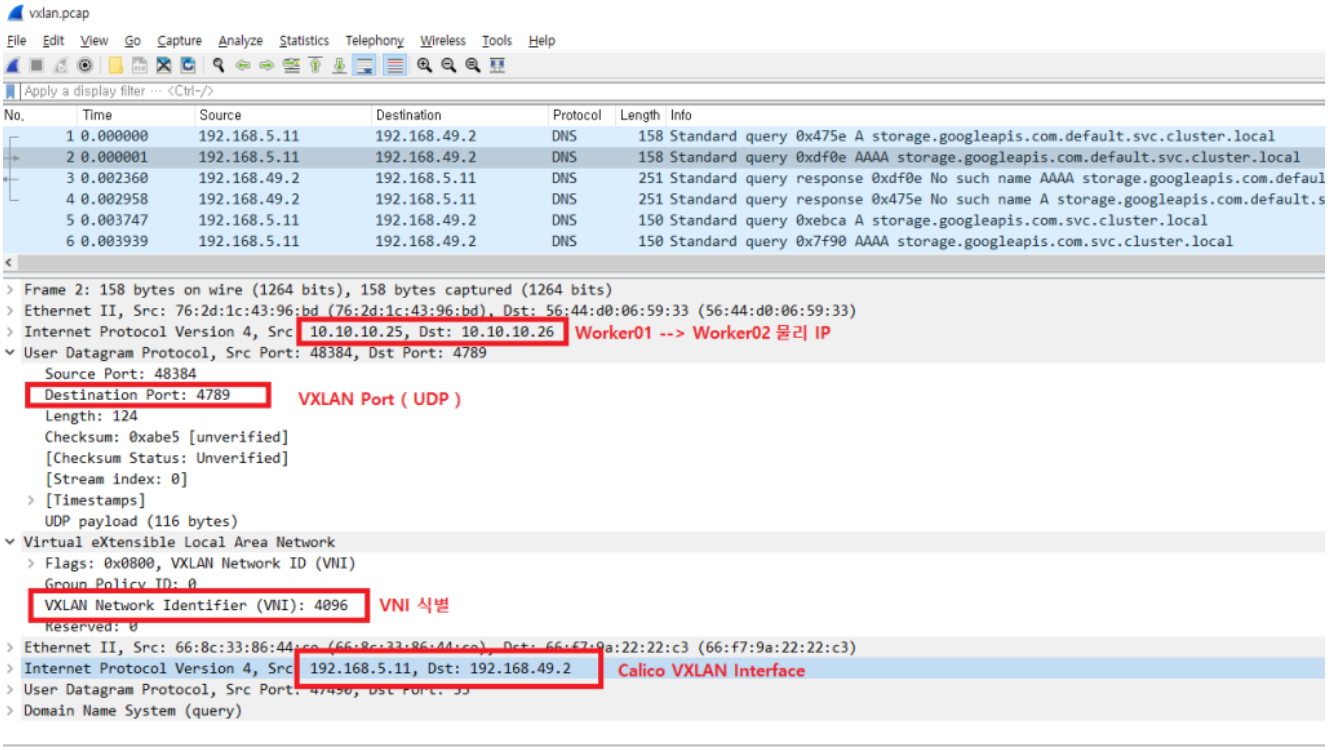
1) worker01 과 worker02 에 위치하는 POD 간 Ping 을 날려본다.

```
hostway@controller:~$      kubectl      exec      -it
sampleos-646dc9654b-8xjw9 -- ping 192.168.30.75
PING 192.168.30.75: 56 data bytes
64 bytes from 192.168.30.75: icmp_seq=0 ttl=115 time=92.124 ms
64 bytes from 192.168.30.75: icmp_seq=1 ttl=115 time=79.735 ms
64 bytes from 192.168.30.75: icmp_seq=2 ttl=115 time=79.233 ms
```

2) 실제 통신은 물리적 인터페이스에서 진행되므로 실제 인터페이스에서 tcpdump 를 추출한다.

```
sudo tcpdump -i ens18 -w vxlan.pcap
```

3) Wireshark 로 확인한다. UDP 패킷을 확인할 수 있다.



[기타] CentOS 7 에 Kubernetes 기본 Install

CentOS 7 에 Kubernetes 기본 설치하기

```
OS : CentOS 7.6.1810 Minimal
Account : root
- 퍼블릭 SNAT IP 세팅
Controller : 10.10.10.237 SSH:4223
Worker-01 : 10.10.10.204 SSH:4224
Worker-02 : 10.10.10.190 SSH:4225
```

전 노드 공통

```
# root 로 초기 로그인. sudo 를 사용하기 위해 일반유저 생성 및 패스워드 지정
useradd -d /home/username username
echo "password" | passwd username --stdin
```

```
# 일반유저는 su 사용하지 못하도록 권한 조정
chmod 700 /usr/bin/su

# sudoer 의 기본 wheel 그룹에 계정 추가
sed -ie '/wheel/s/$/\:username/' /etc/group

# Timezone 설정
sudo timedatectl set-timezone Asia/Seoul

# SWAP OFF
sudo swapoff -a
sudo sed -i -e '/swap/d' /etc/fstab

# firewalld off
sudo systemctl stop firewalld && sudo systemctl disable
firewalld

# Selinux
setenforce 0
sudo sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/selinux/config

# Hostname
sudo hostnamectl set-hostname controller
sudo hostnamectl set-hostname worker-01
sudo hostnamectl set-hostname worker-02

## Controller / Worker
#curl -s https://get.docker.com | sudo sh
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

## Check
sudo docker -v
sudo docker ps -a

## Controller / Worker
sudo mkdir /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
```

```
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

## Docker enable && restart
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

## Packages Repo
sudo cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes
-el7-x86_64
enabled=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF

## Install
sudo yum install -y kubelet kubeadm kubectl --
disableexcludes=kubernetes
```

Controller Init

```
# Controller. 일반적으로 컨트롤러 자신의 공인IP 를 통해 API 서버임을 알
린다 (Advertise)
sudo kubeadm init --ignore-preflight-errors=all --pod-network-
cidr=192.168.0.0/16 --apiserver-advertise-address=10.10.10.237

# Regular User Privileges
  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
# Network Plugin Setting ( Calico )
curl
https://projectcalico.docs.tigera.io/manifests/calico.yaml -O
kubectl apply -f calico.yaml

# System Namespace ( kube-system ) check. CoreDNS 가 올라오면 정상
kubectl get pods -o wide -A

NAMESPACE     NAME                                       READY
STATUS     RESTARTS    AGE       IP                NODE
NOMINATED NODE   READINESS GATES
kube-system   calico-kube-controllers-7c845d499-p85pm   1/1
Running   0           3m6s    192.168.49.3   controller
<none>           <none>
kube-system   calico-node-fnm2q                         1/1
Running   0           3m6s    10.10.10.237   controller
<none>           <none>
kube-system   coredns-64897985d-cgvml                   1/1
Running   0           5m41s   192.168.49.2   controller
<none>           <none>
kube-system   coredns-64897985d-vdckf                   1/1
Running   0           5m42s   192.168.49.1   controller
<none>           <none>
kube-system   etcd-controller                           1/1
Running   0           5m54s   10.10.10.237   controller
<none>           <none>
kube-system   kube-apiserver-controller                 1/1
Running   0           5m54s   10.10.10.237   controller
<none>           <none>
kube-system   kube-controller-manager-controller        1/1
Running   0           6m      10.10.10.237   controller
<none>           <none>
kube-system   kube-proxy-nn5zn                          1/1
Running   0           5m42s   10.10.10.237   controller
<none>           <none>
kube-system   kube-scheduler-controller                 1/1
Running   0           5m54s   10.10.10.237   controller
<none>           <none>


# (공통) Multi NIC 를 가지는 환경에서 INTERNAL-IP 설정
```

다수의 네트워크가 존재하는 서버의 경우 K8S 에서 첫 번째 NIC 의 IP 가 INTERNAL-IP 로 자동 설정된다.
해당 INTERNAL-IP 를 Init 시 설정한
kubeadm --apiserver-advertise-address 와 동일한 IP 대역을 사용하도록 수동 설정한다.

```
cat << EOF | sudo tee /etc/default/kubelet
KUBELET_EXTRA_ARGS='--node-ip $(hostname -I | cut -d ' ' -f2)'
EOF
sudo systemctl daemon-reload
sudo systemctl restart kubelet
kubectl cluster-info
```

Worker Join

```
# Worker-01과 Woker-02 Node 에도 User Privileges 설정 복사 및 적용한다
sudo scp /etc/kubernetes//admin.conf username@10.10.10.204:/home/username/admin.conf
sudo scp /etc/kubernetes//admin.conf username@10.10.10.190:/home/username/admin.conf

# Worker
mkdir -p $HOME/.kube
sudo cp -i ./admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# 각 Worker 에서 kubeadm 를 사용해서 Join 한다.
sudo kubeadm join 10.10.10.237:6443 --token jgocer.fu65ql39kdod5qi0 \
                        --discovery-token-ca-cert-hash sha256:3cb85267e89913d7865d219922daaa8fc6e788dd2be0e2f80fae27176e2dfe3b

# 토큰값을 나중에 재 확인해야 할 경우
kubeadm token create --print-join-command

# Check
kubectl get nodes -o wide
NAME         STATUS     ROLES                  AGE   VERSION
INTERNAL-IP    EXTERNAL-IP   OS-IMAGE                KERNEL-
```

```
VERSION           CONTAINER-RUNTIME
controller   Ready      control-plane,master   16m   v1.23.5
10.10.10.237    <none>          CentOS Linux 7 (Core)
3.10.0-1062.el7.x86_64   docker://20.10.14
worker-01    Ready      <none>                 55s   v1.23.5
10.10.10.204    <none>          CentOS Linux 7 (Core)
3.10.0-1062.el7.x86_64   docker://20.10.14
worker-02    NotReady   <none>                 38s   v1.23.5
10.10.10.190    <none>          CentOS Linux 7 (Core)
3.10.0-1062.el7.x86_64   docker://20.10.14

# Check Pod Create
kubectl run hello --image=nginx --dry-run=client -o yaml |
kubectl apply -f-
pod/hello created
[myungin.baek@controller ~]$ kubectl get pods -o wide
NAME      READY    STATUS      RESTARTS    AGE    IP
NODE          NOMINATED NODE    READINESS GATES
hello   1/1     Running   0          42s   192.168.171.1
worker-01   <none>           <none>
```

[OS] CentOS 7 에서 iptables 로 초기 접근 제어하기

iptables 로 초기 접속 제어 하기

CentOS 7 기본 설치 후, 공인 네트워크 환경에 연결해야 할 때 우선 최소 SSH 접근만 허용하기

(Pre) CentOS 7 에서 firewalld 대신 iptables 사용하기

firewalld 사용이 권장되지만, iptables 에 익숙할 경우 iptables.target 을 service 에 연결하는 패키지를 운영할 수 있다.

```
# firewalld disable
systemctl stop firewalld && systemctl disable firewalld

# firewalld 대신 service 를 통한 운영.
# 설치 후 서비스 재시작하면 /etc/sysconfig/iptables 에 실시간 룰이 저장된다.
yum install iptables-services
service iptables reload
service iptables status

# 룰 저장
service iptables save

# 룰 리로드
service iptables reload

# 룰 형식으로 텍스트 저장. 반드시 -c ( ALL Rule ) 옵션을 지정해서 저장하여야 방화벽 정책과 ROUTE(NAT) 룰이 같이 저장된다.
iptables-save -c > rules.txt

# 룰 복원
iptables-restore < rules.txt
```

iptables 초기 설정 (모두 차단 후 특정 IP만 허용하기)

```
# 모든 룰 초기화
iptables -F

# lo 인터페이스 모든 ACCEPT
iptables -A INPUT -i lo -j ACCEPT

# 모든 접근 허용할 소스 IP 지정. 포트(SSH) 까지 지정할 경우 -p tcp (-m tcp 생략 가능) --dport 22 를 추가한다
iptables -A INPUT -s 1.2.3.4/32 -m comment --comment "회사" -j ACCEPT

# state 상태를 지정한 ACCEPT.
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# (선택) Ping request 요청에 대해 호스트 접근 불가 응답 설정. 서버가 존재한
```

다는 힌트를 주는 것이다.
```
iptables -A INPUT -j REJECT --reject-with icmp-host-prohibited

# (선택) Ping request 요청에 거절 하고 싶다면.
iptables -A INPUT -p icmp --icmp-type echo-request -j REJECT

# (선택) Ping 접근 자체를 막고 싶다면 DROP. 허용하려면 ACCEPT 를 주면 된다.
iptables -A INPUT -p icmp -j DROP

# 그 외 모든 TCP 접근 DROP
iptables -A INPUT -p tcp -j DROP

# 저장
service iptables save
```

추가 접근 정책 추가

```
# 기본 설정 이후에 -A 로 추가하면 DROP 이후의 Line 에 들어가므로 Line 순서 상 제대로 허용되지 않는다
# -I INPUT [DROP Line] 을 기입하여 DROP 라인 위로 추가되도록 한다.

iptables -nL --line-number
------------------------------------------------------------------------------
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  1.2.3.4                    0.0.0.0/0
/* 회사 */
2    DROP       tcp  --  0.0.0.0/0            0.0.0.0/0
------------------------------------------------------------------------------

# 2번 DROP 정책 위로 추가한다.
iptables -I INPUT 2 -s 5.6.7.8 -j ACCEPT -m comment --comment "허용 추가"
iptables -nL --line-number
------------------------------------------------------------------------------
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
```

```
1    ACCEPT     all  --  1.2.3.4                 0.0.0.0/0
/* 회사 */
2    ACCEPT     all  --  5.6.7.8                 0.0.0.0/0
/* 허용 추가 */
3    DROP       tcp  --  0.0.0.0/0            0.0.0.0/0
--------------------------------------------------------------
-----------------
```

라인 정렬이 어려울 경우 /etc/sysconfig/iptables 파일을 직접 수정 후 reload 가능하다.

라인 정책 삭제

```
iptables -D INPUT [Number]
```

[토막] CNI – Calico Plugin

제목 : CNI 와 Calico Network

본문의 내용은 이후에도 업데이트 될 수 있음
내용의 결과값은 #1 에서 초기 구성한 (controller , worker) 시스템에서 진행

CNI (Container Network Interface)

CNCF 에서 정의한 컨테이너 네트워크 인터페이스.
Kubernetes는 기본적으로 Kubenet 이라는 CNI 를 준수하는 Network Plugin 을 갖고 있다.
하지만 복합적이고 다양한 구조의 네트워킹 사용 용도로는 다소 부족하여, 외부 네트워크 플러그인의 필요성이 대두되었다.

Calico Network?

vRouter 기반의 라우터(L3) 기반의 네트워크 관리 모델이다.
Kubernetes 의 서비스 Network 를 구성하는 CNI 를 준수하는 Network

Plugin 중 하나이다.
쿠버네티스의 기본 네트워크 서비스는 기능적으로 부족한 부분이 있는데 자주 사용되는 외부 플러그인이다.

프로젝트 공식 Document URL : https://projectcalico.docs.tigera.io/reference/

Non-overlay 방식의 라우팅

Direct 방식
- BGP(Border Gateway Protocol) 인 BIRD 를 통해 다른 노드 간 라우팅 정보를 동적으로 공유하여 통신할 수 있다
즉 Pod 에서 다른 Pod 로 직접 네트워킹하는 방식으로 성능적으로 가장 뛰어나다.
각 Node 에 존재하는 Calico Pod 는 BGP를 처리하는 Peer 라고 볼 수 있다.
보통 같은 서브넷에 존재하는 노드들을 묶는 방법으로 사용된다. (ex: 라우터 밑의 여러 서버)

Overlay Network 방식의 라우팅

Workload IP(ex: 서비스 컨테이너) 를 모르는 다른 네트워크와의 통신 기법을 의미한다.
여러 노드의 네트워크 패킷을 캡슐화(Encaptulation) 하여 논리적인 공통(L2) 네트워크를 구성하는 방식이다.
캡슐화 패킷 패턴 : 외부 패킷에는 Node 간 IP 정보가 있고, 내부에는 POD 간의 IP 정보가 있다.

IP in IP 방식(Default)
- 각 노드의 네트워크 위치가 달라 Direct 방식으로 통신하기 어려운 경우
IP 패킷 정보를 캡슐화하여 tunl0(tunneling) 인터페이스를 거쳐 노드 간 통신을 가능하게 연결한다.
이후 Direct 와 마찬가지로 BGP (BIRD) 를 통해 노드 간 라우팅 정보를 다른 Node 와 동적(IPVS)으로 전달 및 갱신한다.
Calico Routing 기본 방식이다.

VXLAN 방식
- 가령 물리적 네트워크의 노드와 특정 퍼블릭 클라우드의 다른 네트워크를 연결할 경우
퍼블릭 클라우드 쪽의 라우터 설정 차이 등으로 IP in IP 방식을 쓰지 못할 수 있다. (ex: Azure)
이럴 경우 Calico 에서는 라우터를 거치는 BGP 프로토콜을 사용하지 않고 가상의 VXLAN 인터페이스를 Node 간에 생성하여
L2 캡슐화된 UDP 패킷을 서로 통신할 수 있게 하는 방법이다.

IP in IP 방식에 비해 패킷의 크기가 조금 더 크지만 대규모의 워크로드 시스템이 아닌 한 큰 체감은 없다.

Cross-subnet 방식
캡슐화가 필요한 경우(네트워크가 다를 경우)에서만 선택적인 캡슐화 구성도 가능하다.
(일부 구간/전체 구간)

패킷 암호화 (옵션)

WireGuard
Calico 는 모든 유형의 라우팅 방식에서 패킷 암호화 기능을 별도로 추가할 수 있다.
모든 네트워크 성능의 저하가 발생하게 되므로 적절한 계획 수립이 필요하다.

Calicoctl

Controller 에서 Calico Network 정보를 확인할 수 있는 유틸리티이다.
Host의 단독 명령어 또는 kubectl 에 plugin 타입으로도 설치할 수 있다.

```
# Host 의 명령어 타입으로 설치
$ cd /usr/local/bin
$ sudo curl -L https://github.com/projectcalico/calico/releases/download/v3.22.1/calicoctl-linux-amd64 -o calicoctl
$ sudo chmod +x calicoctl

# Check
Calico 가 관리하는 Network 의 Pool과 Block 을 확인할 수 있다.
$ sudo calicoctl ipam show --show-blocks
+----------+-------------------+-----------+------------+--------------+
| GROUPING |        CIDR       | IPS TOTAL | IPS IN USE |   IPS FREE   |
+----------+-------------------+-----------+------------+--------------+
| IP Pool  | 192.168.0.0/16    |     65536 | 5 (0%)     | 65531 (100%) |
| Block    | 192.168.136.0/26  |        64 | 4 (6%)     | 60 (94%)     |
| Block    | 192.168.153.192/26 |        64 | 1 (2%)     | 63 (98%)     |
```

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

BGP 프로토콜을 사용해 라우팅 정보를 전달중인 노드를 확인할 수 있다

```
$ sudo calicoctl node status
Calico process is running.
IPv4 BGP status
+---------------+-------------------+-------+----------+-----
--------+
|  PEER ADDRESS  |     PEER TYPE     | STATE |  SINCE   |
INFO     |
+---------------+-------------------+-------+----------+-----
--------+
| 203.248.23.215 | node-to-node mesh | up    | 05:27:05 |
Established |
+---------------+-------------------+-------+----------+-----
--------+
```

Block 에서 사용중인 관련 인터페이스들

```
$ route -n | egrep "tun|cali|\*"
192.168.136.0   0.0.0.0         255.255.255.192 U     0
0         0 *
192.168.136.1   0.0.0.0         255.255.255.255 UH    0      0
0 calibc6c3028870
192.168.136.2   0.0.0.0         255.255.255.255 UH    0      0
0 calid6edae09645
192.168.136.3   0.0.0.0         255.255.255.255 UH    0      0
0 calic6bfd11bfbe
192.168.153.192 203.248.23.215  255.255.255.192 UG    0      0
0 tunl0
```

어떤 Pod가 어떤 calicxxxxxx 인터페이스를 사용하는지 확인하려면 다음의 명령어를 사용한다
System(default) Namespace 까지 보이게 하려면 -A 옵션을 추가한다.

```
$ calicoctl get workloadendpoint -A
NAMESPACE     WORKLOAD                                  NODE
NETWORKS           INTERFACE
kube-system   calico-kube-controllers-56fcbf9d6b-nlqg2   user-
controller   192.168.136.2/32   calid6edae09645
kube-system   coredns-64897985d-jgj5s                   user-
controller   192.168.136.3/32   calic6bfd11bfbe
```

```
kube-system   coredns-64897985d-vbpn4                    user-
controller   192.168.136.1/32   calibc6c3028870
```

Calico 인터페이스는 Veth type(Pair) 이다.

```
$ ip -br -c link show type veth
calibc6c3028870@if3 UP                ee:ee:ee:ee:ee:ee
<BROADCAST,MULTICAST,UP,LOWER_UP>
calid6edae09645@if4 UP                ee:ee:ee:ee:ee:ee
<BROADCAST,MULTICAST,UP,LOWER_UP>
calic6bfd11bfbe@if4 UP                ee:ee:ee:ee:ee:ee
<BROADCAST,MULTICAST,UP,LOWER_UP>
```

Calico Management Pod

특정 Daemon이 아닌 Pod 형태로 존재한다.
Controller 와 Worker Node 에 각각 실행되는 Pod 가 다름을 체크한다.

```
$ kubectl get pods -o wide -n kube-system
NAME                                       READY   STATUS
RESTARTS   AGE   IP               NODE
calico-kube-controllers-56fcbf9d6b-nlqg2   1/1     Running   0
30m   192.168.136.2   user-controller
calico-node-8cts6                          1/1     Running   0
30m   10.0.2.15       user-controller
calico-node-mb9n6                          1/1     Running   0
29m   10.0.2.15       user-worker
```

Calico의 설정 정보 등의 DB 값은 etcd 라는 서비스의 datastore 에 저장된다.

```
$ kubectl get pods -o wide -n kube-system | grep -i etcd
etcd-user-controller                       1/1     Running   0
39m   10.0.2.15       user-controller
```

Calico Felix

쿠버네티스 Pod 네트워크에서 kube-proxy 상위 역할을 해 주는 컴포넌트이다.
etcd 에서 정보들을 읽어와서 Pod Network 의 라우팅 테이블과 룰을 관리한다.
kube-proxy 가 iptables / ipvs Mode 일경우 그에 맞춰 관리해준다.
iptables 라인이 너무 클 경우 성능에 영향이 있을 수 있어 플러그인에 의한 ipvs 방식이 선호된다.
 IPVS = Hash 값으로 동적으로 라우팅 테이블을 관리하는 로드밸런서 알고리즘

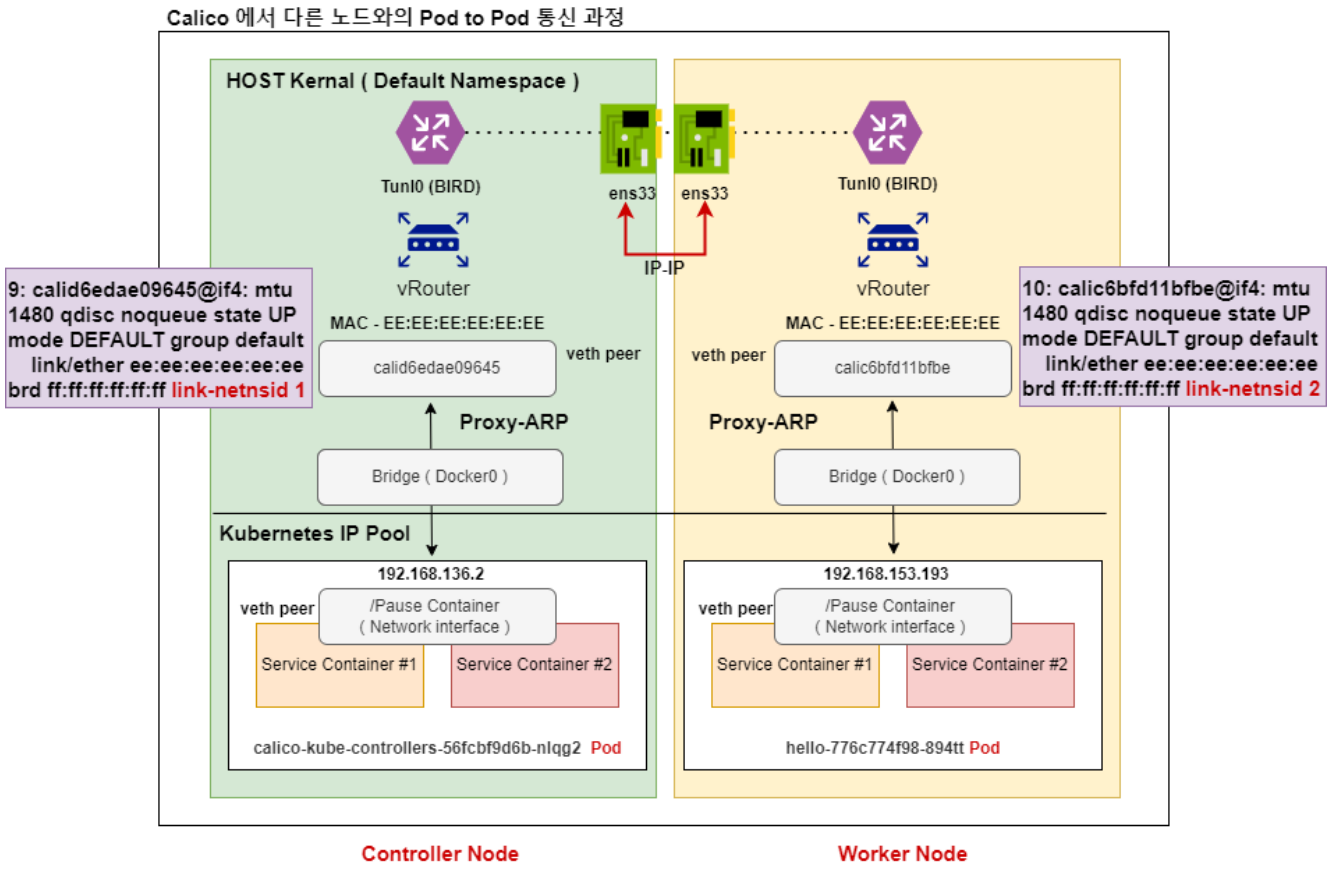
```
$ sudo iptables -t nat -S  | grep -i cali
```

```
$ sudo iptables -t filter -S  | grep -i cali
```

Networking 과정

IP in IP Networking

Controller Node 에서 Worker Node 에 있는 Pod 네트워크로 통신을 시도한다



1) Controller의 192.168.136.2 Pod 에서 Worker의 192.168.153.193 Pod 로 통신
2) Controller의 Pod 인터페이스(veth) 는 Pair 인 Host의 calico (veth) 인터페이스로 ARP 요청
3) Host 의 Calico 인터페이스는 Worker Pod 의 네트워크 정보를 모르므로 ARP 응답 할 수 없어야 함
4) 하지만 Calico 인터페이스는 link-local 타입(내부 브로드캐스트 학습)으로써 HOST 가 BIRD 에 의해 학습한 Worker 의 라우팅 정보를 알고 있음
5) Controller의 Calico 인터페이스는 vRouter에 설정된 ARP_Proxy 기능을 활용하여 Worker 의 ARP 응답을 대신 해 줌
6) BIRD는 Tunl0 --> Host 의 실제 인터페이스를 통해 상대 노드와 Pod 정보를 추가한 캡슐화 된 패킷으로 통신
7) 그림에는 보이지 않지만 만약 다른 노드와 통신이 아닌 퍼블릭 대역과 통신을 한다면
Felix 를 통한 SNAT (MASQUERADE) 를 설정하여 tunl0 을 거치지 않고

HOST 의 ens33 으로 바로 연결.

Packet Check

다른 네트워크 (Controllor POD <---> Worker POD) 간 Ping 체크를 진행한다

```
$ kubectl get pod -o wide
NAME                      READY   STATUS    RESTARTS   AGE   IP
NODE              NOMINATED NODE   READINESS GATES
hello-776c774f98-894tt    1/1      Running   0          13d
192.168.153.193   user-worker      <none>           <none>
hi                        1/1      Running   0          13d
192.168.136.5     user-controller  <none>           <none>
```

Worker POD --> Container POD. 컨테이너에 Ping 명령이 없어 Pod 밖의 Host 에서 컨테이너 PID 를 따서 진행했다.

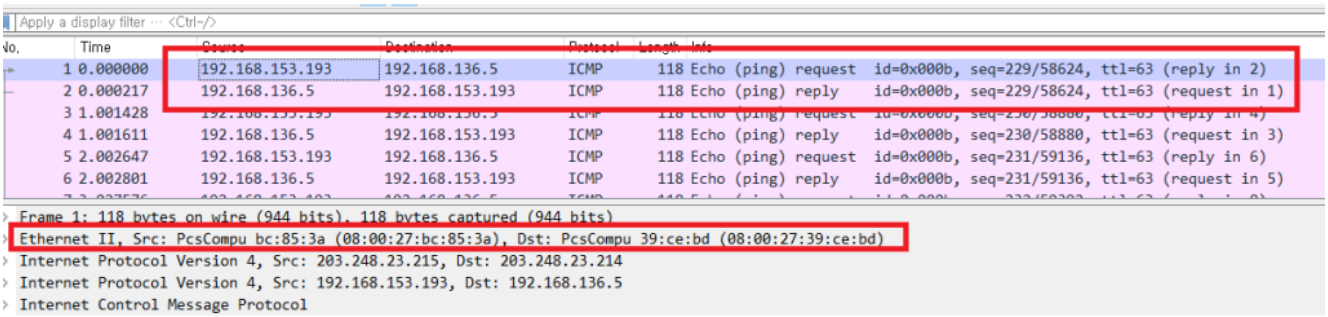
```
$ sudo nsenter -t 225201 -n ping 192.168.136.5
64 bytes from 192.168.136.5: icmp_seq=627 ttl=62 time=0.709 ms
64 bytes from 192.168.136.5: icmp_seq=628 ttl=62 time=0.675 ms
64 bytes from 192.168.136.5: icmp_seq=629 ttl=62 time=0.727 ms
64 bytes from 192.168.136.5: icmp_seq=630 ttl=62 time=0.797 ms
64 bytes from 192.168.136.5: icmp_seq=631 ttl=62 time=0.887 ms
```

Controller 에서 패킷 캡쳐를 한다. IPIP 구조는 API 공인 인터페이스를 통해 터널링하므로, API 인터페이스든 터널링 인터페이스든 상관없다.

```
$ sudo tcpdump -i enp0s8 -nn proto 4 -w test.pcap
```

Wireshark 로 확인한다.

1) 두 POD IP의 다른 네트워트 간 정상 ICMP 통신을 하고 있는 부분을 확인할 수 있다.

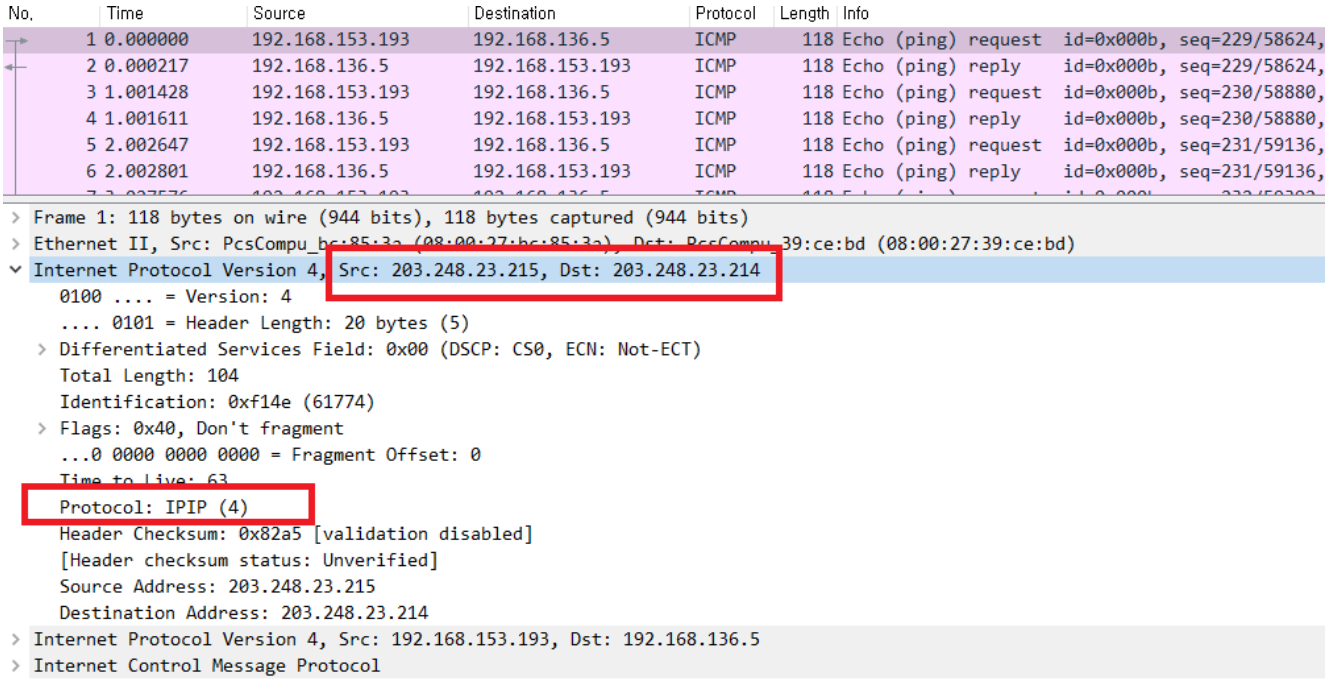


2) MAC 확인시 Controller 와 Worker Node 의 API 공인IP 인터페이스 간 통신임을 확인했다.

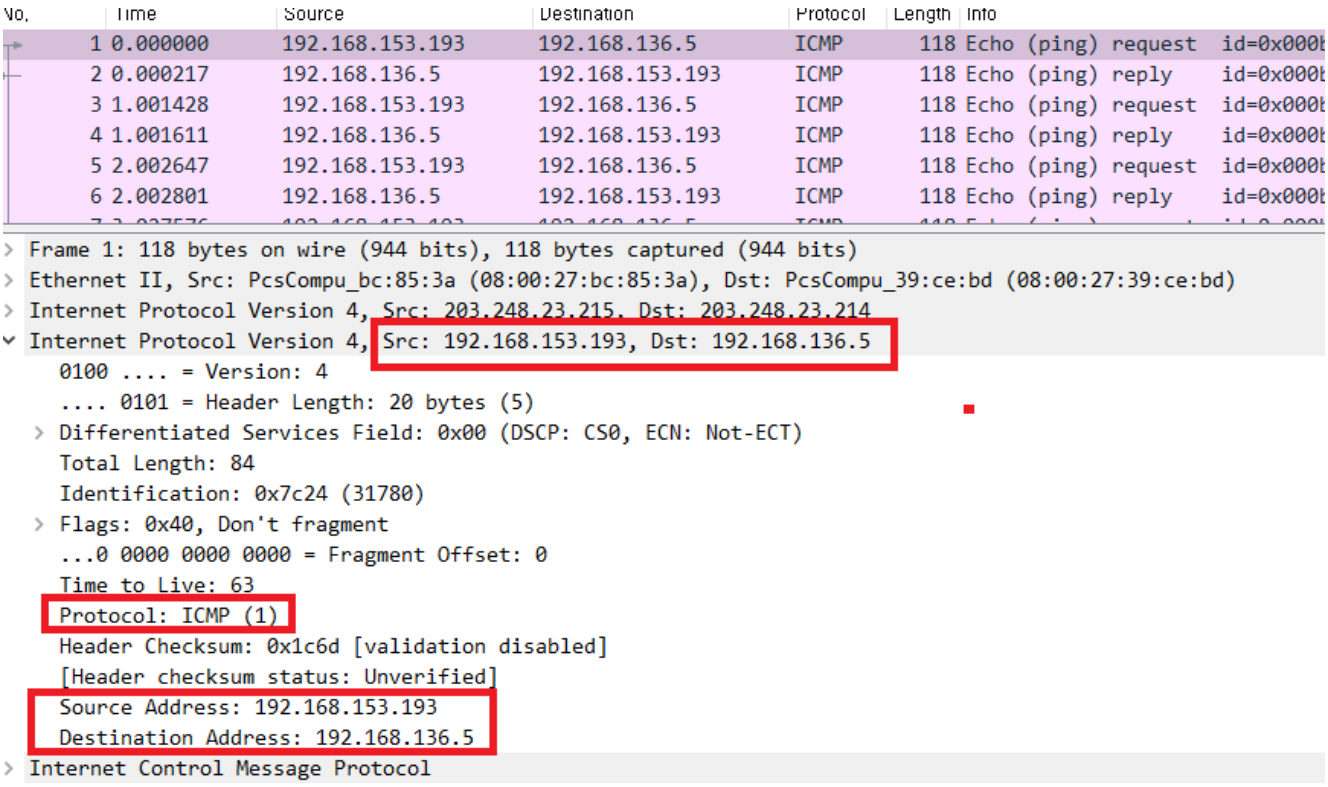
```
# Controller
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    link/ether 08:00:27:39:ce:bd brd ff:ff:ff:ff:ff:ff
     inet 203.248.23.214/25 brd 203.248.23.255 scope global
enp0s8

# Worker
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    link/ether 08:00:27:bc:85:3a brd ff:ff:ff:ff:ff:ff
     inet 203.248.23.215/25 brd 203.248.23.255 scope global
enp0s8
```

3) IPv4 Protocol 확인 시 2개의 헤더를 확인할 수 있다.

공인망의 Outer IP 와 POD 네트워크인 Inner IP 2개를 확인할 수 있다.



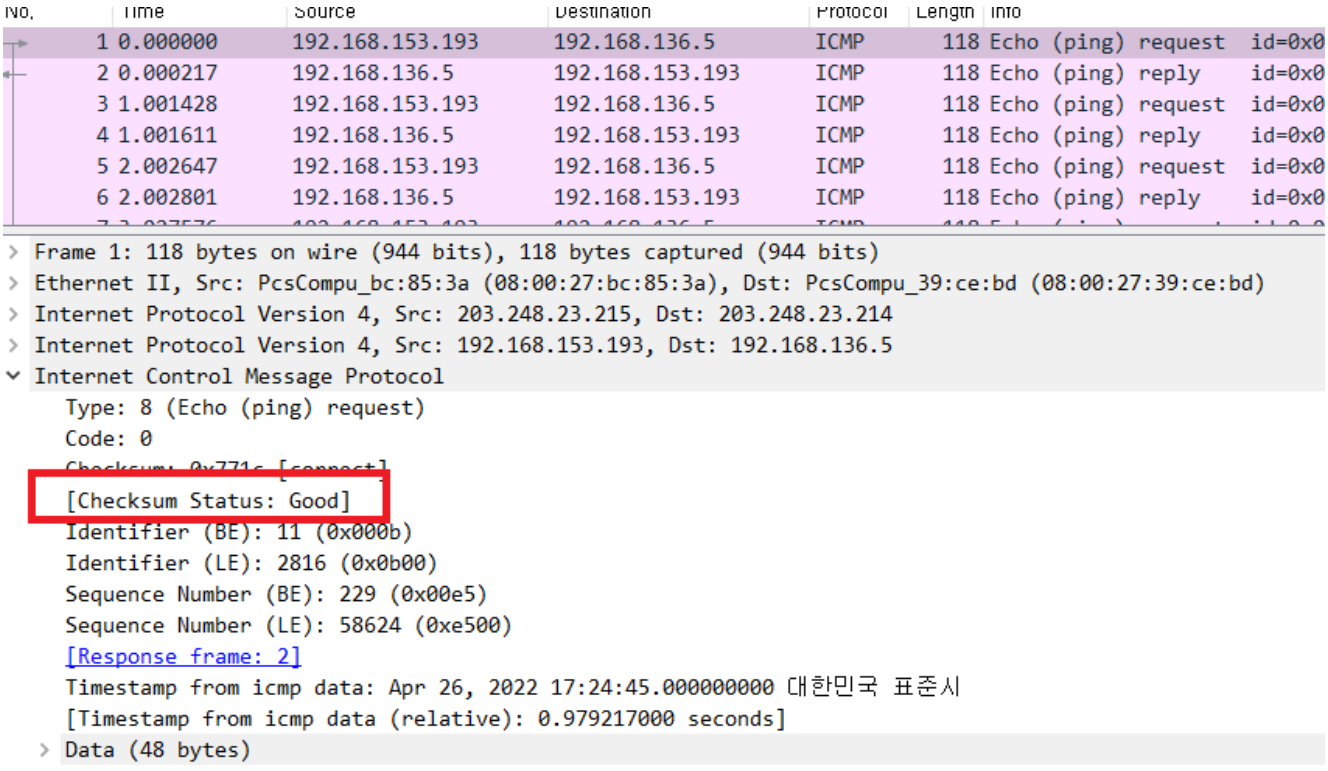
Outer IP 헤더가 InnerIP 를 밖에서 캡슐로 감싼 IP-IP Protocol 임을 확인할 수 있다.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.153.193	192.168.136.5	ICMP	118	Echo (ping) request id=0x000b
2	0.000217	192.168.136.5	192.168.153.193	ICMP	118	Echo (ping) reply id=0x000b
3	1.001428	192.168.153.193	192.168.136.5	ICMP	118	Echo (ping) request id=0x000b
4	1.001611	192.168.136.5	192.168.153.193	ICMP	118	Echo (ping) reply id=0x000b
5	2.002647	192.168.153.193	192.168.136.5	ICMP	118	Echo (ping) request id=0x000b
6	2.002801	192.168.136.5	192.168.153.193	ICMP	118	Echo (ping) reply id=0x000b

```
> Frame 1: 118 bytes on wire (944 bits), 118 bytes captured (944 bits)
> Ethernet II, Src: PcsCompu_bc:85:3a (08:00:27:bc:85:3a), Dst: PcsCompu_39:ce:bd (08:00:27:39:ce:bd)
> Internet Protocol Version 4, Src: 203.248.23.215, Dst: 203.248.23.214
v Internet Protocol Version 4, Src: 192.168.153.193, Dst: 192.168.136.5
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x7c24 (31780)
  > Flags: 0x40, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 63
    Protocol: ICMP (1)
    Header Checksum: 0x1c6d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.153.193
    Destination Address: 192.168.136.5
> Internet Control Message Protocol
```

4) Messages 에서 통신에 정상 문제가 없음을 확인했다.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.153.193	192.168.136.5	ICMP	118	Echo (ping) request id=0x0
2	0.000217	192.168.136.5	192.168.153.193	ICMP	118	Echo (ping) reply id=0x0
3	1.001428	192.168.153.193	192.168.136.5	ICMP	118	Echo (ping) request id=0x0
4	1.001611	192.168.136.5	192.168.153.193	ICMP	118	Echo (ping) reply id=0x0
5	2.002647	192.168.153.193	192.168.136.5	ICMP	118	Echo (ping) request id=0x0
6	2.002801	192.168.136.5	192.168.153.193	ICMP	118	Echo (ping) reply id=0x0

```
> Frame 1: 118 bytes on wire (944 bits), 118 bytes captured (944 bits)
> Ethernet II, Src: PcsCompu_bc:85:3a (08:00:27:bc:85:3a), Dst: PcsCompu_39:ce:bd (08:00:27:39:ce:bd)
> Internet Protocol Version 4, Src: 203.248.23.215, Dst: 203.248.23.214
> Internet Protocol Version 4, Src: 192.168.153.193, Dst: 192.168.136.5
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x771c [correct]
    [Checksum Status: Good]
    Identifier (BE): 11 (0x000b)
    Identifier (LE): 2816 (0x0b00)
    Sequence Number (BE): 229 (0x00e5)
    Sequence Number (LE): 58624 (0xe500)
    [Response frame: 2]
    Timestamp from icmp data: Apr 26, 2022 17:24:45.000000000 대한민국 표준시
    [Timestamp from icmp data (relative): 0.979217000 seconds]
  > Data (48 bytes)
```

참고자료

Network Overlay : https://ikcoo.tistory.com/117

원격으로 CentOS 7 에서 내부 Windows 서버에 RDP 연결

CentOS 7 에서 Windows RDP 연결하기

진행 순서

```
# OS
CentOS 7.9 x86_64 minimal
로컬--> XRDP 를 통한 리눅스 GUI 접속 --> 리눅스 GUI 에서 내부 Windows RDP 접속
```

Linux GUI 환경 세팅

```
# GUI GroupInstall
root@localhost ~]# yum groups list | grep -i desktop
   Cinnamon Desktop
   MATE Desktop
   GNOME Desktop
   General Purpose Desktop
   LXQt Desktop
# GNOME 이 제대로 설치되지 않는다면 "Server with GUI" 그룹을 설치한다.
root@localhost ~] yum groupinstall "GNOME Desktop"

# GUI init
[root@localhost ~]# systemctl get-default
multi-user.target
[root@localhost ~]# systemctl set-default graphical.target
[root@localhost ~]# systemctl get-default
graphical.target
# Reboot 후 자동으로 GUI 모드로 부팅되는지 확인한다.
[root@localhost ~]# reboot
```

Linux 서버에 원격 접속 설정

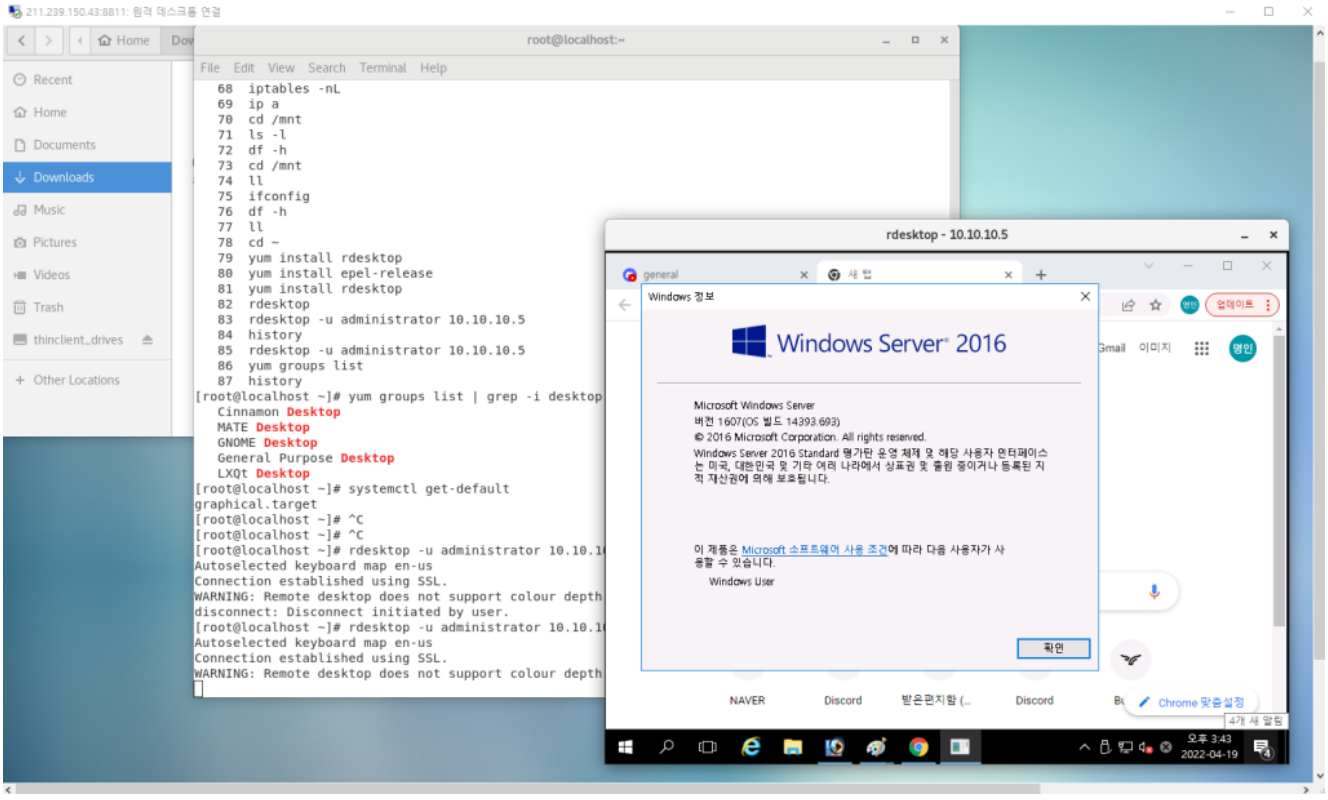
```
# XRDP Install.
[root@localhost ~]# yum install epel-release
[root@localhost ~]# yum install xrdp
[root@localhost ~]# systemctl enable xrdp && systemctl start xrdp

# selinux disable 및 iptables -F or  방화벽 정책에 tcp/3389 를 추가 해준다.
```

rdesktop 설치

```
# 컴파일러와 openssl-devel 이 패키지 컴파일 선행조건이다.
 yum -y install gcc openssl-devel
                                                                wget https://github.com/rdesktop/rdesktop/releases/download/v1.8.6/rdesktop-1.8.6.tar.gz
 tar xvzf rdesktop-1.8.6.tar.gz
 cd rdesktop-1.8.6/
 ./configure --disable-credssp --disable-smartcard
 make
 make install
```

Check

```
# 리눅스 서버에 RDP 로 접속 후, rdesktop -u [User] [ip]  입력한다.
root@localhost ~]# rdesktop -u administrator 10.10.10.5
Autoselected keyboard map en-us
Connection established using SSL.
WARNING: Remote desktop does not support colour depth 24; falling back to 16
```


211.239.150.43:8811: 원격 데스크톱 연결
Home
Recent
Home
Documents
Downloads
Music
Pictures
Videos
Trash
thinclient_drives
Other Locations
root@localhost:~
File Edit View Search Terminal Help
68 iptables -nL
69 ip a
70 cd /mnt
71 ls -l
72 df -h
73 cd /mnt
74 ll
75 ifconfig
76 df -h
77 ll
78 cd ~
79 yum install rdesktop
80 yum install epel-release
81 yum install rdesktop
82 rdesktop
83 rdesktop -u administrator 10.10.10.5
84 history
85 rdesktop -u administrator 10.10.10.5
86 yum groups list
87 history
[root@localhost ~]# yum groups list | grep -i desktop
Cinnamon Desktop
MATE Desktop
GNOME Desktop
General Purpose Desktop
LXQt Desktop
[root@localhost ~]# systemctl get-default
graphical.target
[root@localhost ~]# ^C
[root@localhost ~]# ^C
[root@localhost ~]# rdesktop -u administrator 10.10.1
Autoselected keyboard map en-us
Connection established using SSL.
WARNING: Remote desktop does not support colour depth
disconnect: Disconnect initiated by user.
[root@localhost ~]# rdesktop -u administrator 10.10.1
Autoselected keyboard map en-us
Connection established using SSL.
WARNING: Remote desktop does not support colour depth
rdesktop - 10.10.10.5
general
새 탭
Windows 정보
Windows Server 2016
Microsoft Windows Server
버전 1607(OS 빌드 14393.693)
© 2016 Microsoft Corporation. All rights reserved.
Windows Server 2016 Standard 평가판 운영 체제 및 해당 사용자 인터페이스는 미국, 대한민국 및 기타 여러 나라에서 상표권 및 출원 중이거나 등록된 지적 재산권에 의해 보호됩니다.
이 제품은 Microsoft 소프트웨어 사용 조건에 따라 다음 사용자가 사용할 수 있습니다.
Windows User
확인
업데이트
Gmail
이미지
NAVER
Discord
받은편지함 (...
Discord
Chrome 맞춤설정
4개 새 알림
오후 3:43
2022-04-19