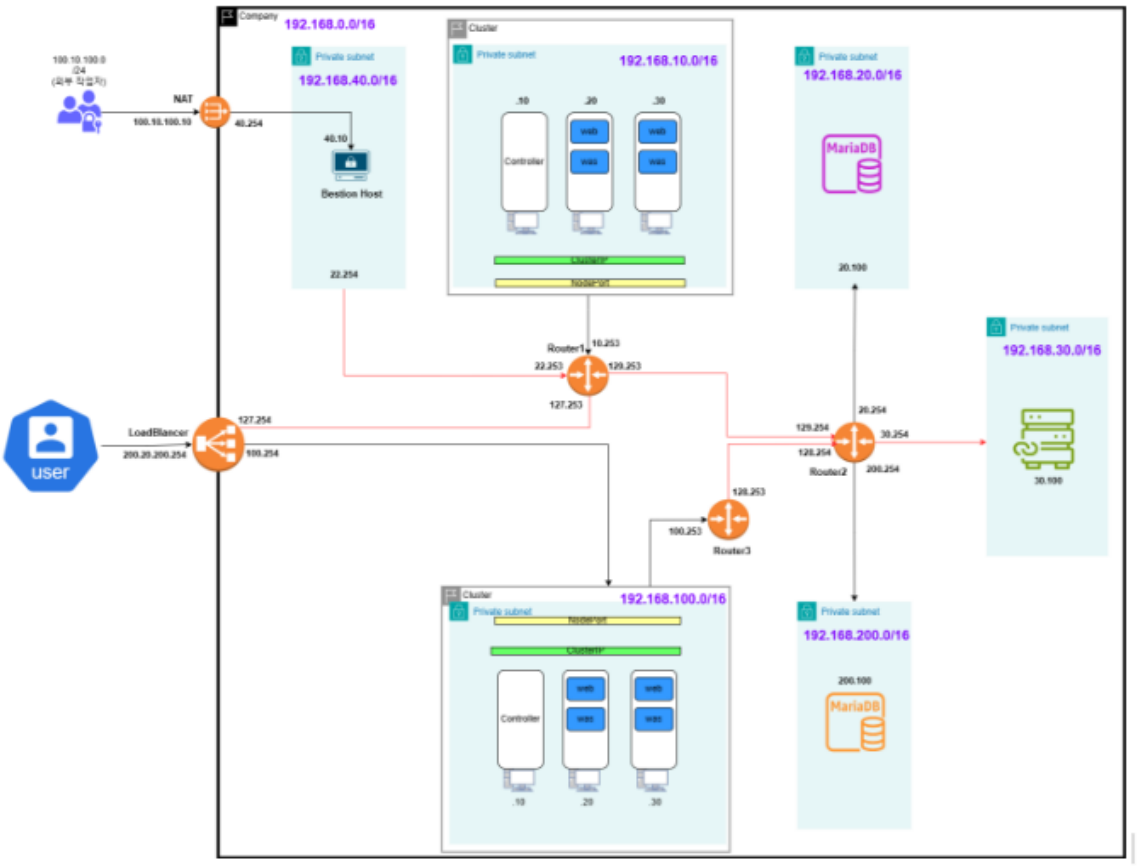
최민영 개인 보고서

구성도



💡 🏗️ 시나리오

기업 A는 온라인 쇼핑몰 운영 사업을 시작한 신규 IT 서비스 사업자로, 신규 고객 유입을 위한 정기 특가 상품 판매 행사의 시작을 앞두고 있습니다. 이 행사는 향후 일정 기간 동안 정기적으로 시행 될 예정이며, 시행될 때 마다 고객들은 단 한 시간 동안만 특가로 상품을 구매할 수 있습니다. 이를 위해, 기업 A의 인프라 팀에서는 원활한 행사 진행 및 활발한 DAU 유입을 위해 서비스 중단 최소화 인프라 구축을 목표로 쇼핑몰 이용 고객들의 행사 간 배송 및 상품 정보에 관한 원활한 문의 등록과 함께 최종 서비스 사용자들이 자사 서비스에 대한 요구사항을 깊이 있게 이해하고, 신속하게 대응할 수 있도록 기존 고객 Q&A 게시판이 필요하다고 저희 LED Company에 인프라 구축을 의뢰하였습니다.행사 진행 중에는 많은 접속이 예상되기 때문에, 다수의 서버를 준비할 계획입니다. 하지만, 하나의 서버에 요청이 몰릴 경우 준비한 의미가 없기 때문에, 이를 분산시키기 위한 방법이 필요하다고 판단하였습니다.

최근, 국내 최대 온라인 전자상거래 플랫폼을 운영하는 기업 B의 상품 데이터 베이스 고장을 일으킨 적이 있습니다. 이에 판매자들이 서비스를 위해 업로드한 이미지 파일이 삭제돼 불편을 겪은 것으로 확인되었습니다. 서버가 하드웨어 장애로 판매자들이 업로드한 이미지가 삭제되는 일이 발생하는 등의 문제에 대비하기 위해 기업 A는 서버 장애 발생 시에도 판매자들이 업로드한 이미지 파일이 안전하게 보호될 수 있도록 대비하기로 결정했습니다. 이를 위해 Secondary DB 서버와 다운된 Primary DB 서버의 복구를 도울 수 있는 NFS(Network File System) 서버를 구축할 계획이며, 이를 통해 기업 A는 서버 장애로 인한 데이터 손실을 최소화하고, 판매자 및 고객들에게 안정적인 서비스를 제공할 수 있게끔 인프라 구축 방향을 설정하였습니다.

기업 A의 인프라 기존 구성과 변경 시 고려해야 할 요소는 다음과 같습니다. 정기 특가 상품 판매 행사의 시작을 앞두고 있는 기업 A는 첫 행사의 시작과 동시에 폭증할 트래픽의 규모를 정확히 예측해 수치로 제시하기 어려운 상황입니다. 기업 A는 현재 온프레미스 환경에서 가상화 기술을 활용해 WEB 서비스용 머신과 WAS용 머신을 별도로 운영하고 있으며, 자원의 효율성과 고가용성, 복구력 강화를 위해 Kubernetes 도입을 원하고 있습니다. 고객들의 서비스 접속이 폭발적으로 증가할 수 있는 특가 상품 판매 이벤트 특성 상 트래픽을 분산 처리해야 합니다.

모든 고객 데이터는 외부에 공개할 수 없는 민감한 정보로 취급하며, 보안을 위해 고객 데이터용 데이터베이스는 독립된 온프레미스 서버에 저장해야 합니다. 고객 데이터 손실을 방지하고 장애에 대비하기 위해 데이터베이스는 동일 데이터센터 내 별도의 서버에 주기적으로 백업을 진행해야 합니다. 웹 어플리케이션의 업데이트가 빈번하게 자주 이루어질 예정이기 때문에 버전 업데이트를 용이하게 해야 하며, 외부에 있는 관리자를 위한 방법을 준비해야 합니다. 보안성 향상 및 검색 엔진 최적화(SEO)를 위해 웹 페이지에 SSL을 적용해야 합니다.

✅ 개인 디벨롭

Database (192.168.20.0/24) (192.168.200.0/24)	MariaDB-A	192.168.20.100	Primary Database
	MariaDB-B	192.168.200.100	Secondary Database
NFS (192.168.30.0/24)	NFS	192.168.30.100	Network File System

1. 특정 호스트 또는 네트워크에 대한 엑세스 설정

💡 NFS 서버의 `/etc/exports` 파일에서 특정 호스트 또는 네트워크에 대한 액세스를 설정하여 해당 호스트 또는 네트워크에 대한 액세스 권한 제한

- NFS 서버는 파일 시스템을 네트워크를 통해 공유하므로 악의적인 클라이언트로부터의 액세스를 방지하기 위해 보안 강화가 필요
- 액세스 제어 설정을 통해 불필요한 클라이언트로부터의 액세스를 차단하고, 허가된 클라이언트만이 서버에 접근할 수 있도록 제어할 수 있음
- 허가되지 않은 클라이언트의 액세스를 차단함으로써 서버 리소스를 보호하고, 네트워크 대역폭을 효율적으로 관리하여 시스템의 안정성과 성능을 유지

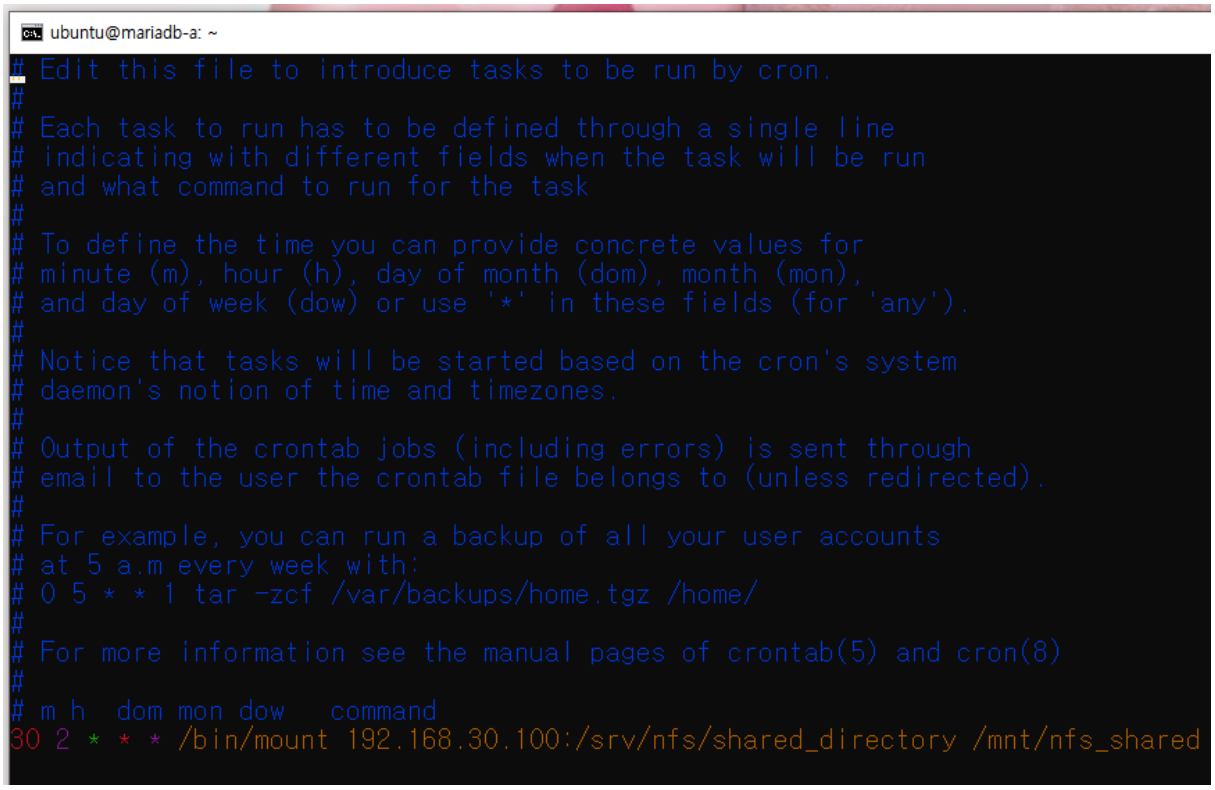
```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
/srv/nfs/shared_directory *(rw, sync,no_subtree_check)
```

→ 기존에는 * 을 사용하여 모든 호스트에 대한 접근을 허용

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
/srv/nfs/shared_directory 192.168.20.0/24(rw,sync,no_subtree_check)
```

→ 특정 서브넷(`192.168.20.0/24`) 내의 호스트로 접근을 제한

2. crontab 설정

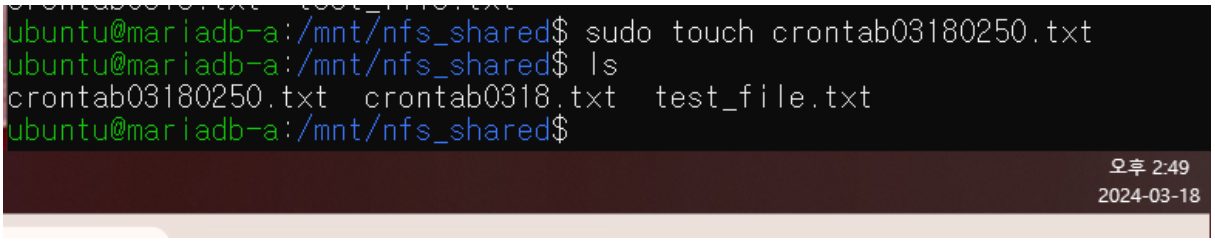



```
crontab -e
```

→ NFS Client Server 에서 crontab 수정

```
EX) 매일 2시 50분에 자동 MOUNT 설정

50 2 * * * /bin/mount 192.168.30.100:/srv/nfs/shared_directory /mnt/nfs_share
```

📂 crontab 실행 확인



일반 파일로 crontab 정상 실행 확인

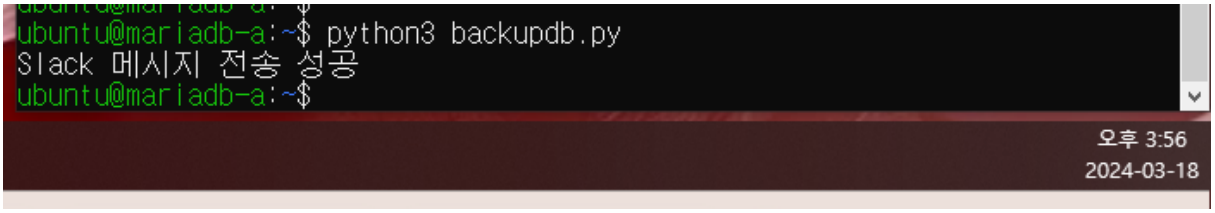
3. 실제 DB 백업 파일로 mount 확인

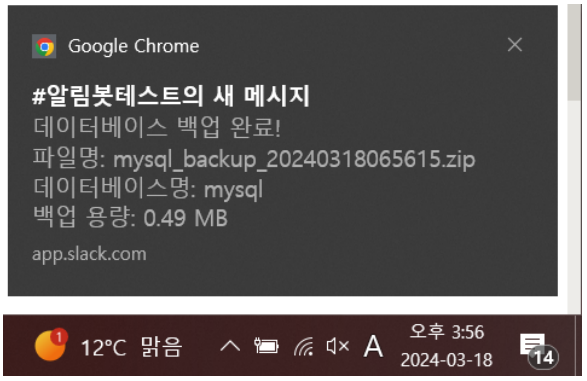
- 데이터베이스 백업 도구

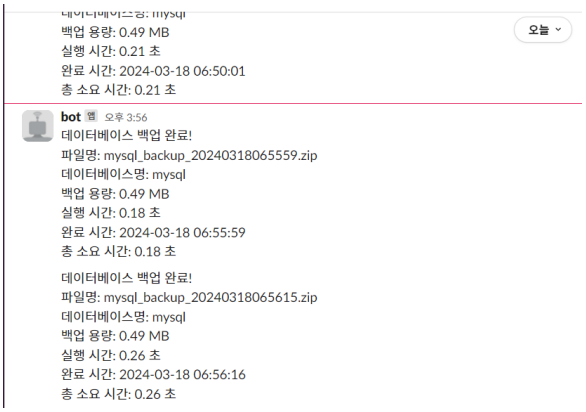
```
import subprocess
import os
import time
import datetime
import requests
import shutil
import zipfile

def backup_database(db_name, backup_folder):
    timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
    backup_file = f"{db_name}_backup_{timestamp}.sql"
    backup_path = os.path.join(backup_folder, backup_file)

    try:
        subprocess.run(["mysqldump", "-u", "root", "-proot", db_name, "--resu
        return backup_path
    except subprocess.CalledProcessError as e:
        print(f"백업 실패: {e}")
        return None

def compress_and_store(backup_path, storage_folder):
    compressed_file = f"{os.path.splitext(os.path.basename(backup_path))[0]}.
    compressed_path = os.path.join(storage_folder, compressed_file)
    with zipfile.ZipFile(compressed_path, 'w', zipfile.ZIP_DEFLATED) as zipf:
        zipf.write(backup_path, os.path.basename(backup_path))
    os.remove(backup_path)
    return compressed_path

def send_slack_message(token, channel, message):
    url = "https://slack.com/api/chat.postMessage"
    headers = {
        "Content-Type": "application/json",
```

```
        "Authorization": f"Bearer {token}"
    }
    payload = {
        "channel": channel,
        "text": message
    }
    response = requests.post(url, headers=headers, json=payload)
    return response


def main():
    slack_token = os.environ.get("slack_token")
    slack_channel = "C06G735J4CA"
    db_name = "mysql"
    backup_folder = "/mnt/nfs_shared"
    storage_folder = "/mnt/nfs_shared"
    start_time = time.time()
    backup_path = backup_database(db_name, backup_folder)
    if backup_path:
        compressed_path = compress_and_store(backup_path, storage_folder)
        end_time = time.time()
        duration = end_time - start_time
        message = f"데이터베이스 백업 완료!\n파일명: {os.path.basename(compressed_p
        response = send_slack_message(slack_token, slack_channel, message)
        if response.status_code == 200 and response.json()["ok"]:
            print("Slack 메시지 전송 성공")
        else:
            print("Slack 메시지 전송 실패")
    else:
         print("백업 실패")
if __name__ == "__main__":
        main()
```

```
sudo vi backupdb.py

db_name = "mysql": mysql db 백업

backup_folder = "/mnt/nfs_shared": 이 위치에 db 백업본 저장

zipfile.ZipFile: 모듈을 사용하여 백업 파일을 ZIP 형식으로 압축

send_slack_message(token, channel, message): 이 함수는 Slack API를 사용하여 메시지
```

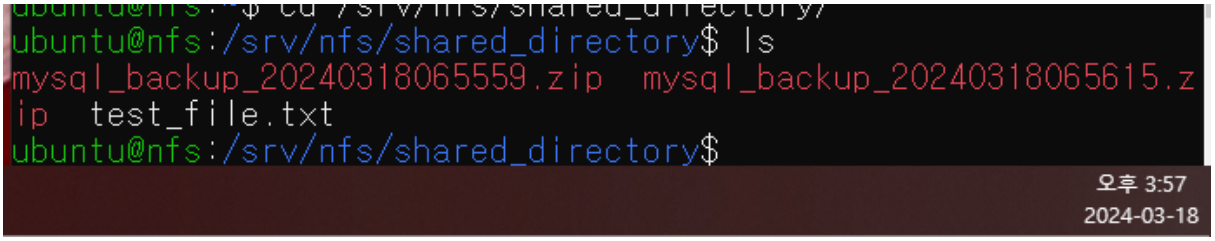






slack 알림 성공

→ NFS Client Server 에 백업 성공 확인



→ NFS Server 에 백업 파일 mount 성공

💬느낀점

💡 이번 프로젝트를 통해 여러 가지 기술과 도구를 활용하여 가상의 기업 A의 고객 데이터 보안과 서비스 신뢰성을 강화하고, 향후 성장에 대비할 수 있는 기반을 마련하는 데에 좋은 경험이었습니다. 또한, 각자가 맡은 역할을 성실히 수행하고 서로의 의견을 존중하며 함께 문제를 해결하여 프로젝트를 성공적으로 마칠 수 있었습니다.

네트워크 설정은 이번 프로젝트에서 가장 중요하면서도 기본적인 부분이었습니다. 특히, 서로 다른 서버 간의 통신을 원활히 구성하기 위해서는 그동안 배웠던 IP 주소 할당, 서브넷 마스크 설정, 라우팅 등에 대한 이해가 필수적으로 필요했습니다. 처음에는 예상치 못한 문제들도 많이 발생하여 어려움이 있었지만 팀원들과 함께 자료를 찾아가며 이해하는 과정을 통해 다양한 네트워크 설정 사례를 경험할 수 있었습니다.

이 프로젝트의 핵심이었던 가상화 기술과 쿠버네티스 클러스터 구축에 대한 실제 경험이 부족하여 복잡한 구성도를 그리는 데에 어려움이 있었습니다. 그러나 팀원들과 다양한 문제들에 대해 논의하며 쿠버네티스를 활용하여 자원의 효율성과 고가용성을 확보하면서도 버전 업데이트와 서비스 확장이 용이하도록 인프라를 구축할 수 있었습니다. 또한, Replica DB와 온프레미스 DB 백업을 통해 고객 데이터의 안전성을 보장하고 장애 대응 능력을 향상시킬 수 있었고, NFS를 활용한 서버 복구 기능은 데이터 손실을 최소화하고 서비스 지속성을 보장하는 등 실제 업무에서도 유용하게 활용될 수 있는 기술등을 경험할 수 있었습니다.

이번 프로젝트를 통해 단순히 기술적인 측면뿐만 아니라 팀원 간의 협업과 의사 소통의 중요성을 다시 한번 깨달았습니다. 또한, 새로운 기술에 도전하고 문제를 해결하는 과정에서의 즐거움과 성취감을 느낄 수 있었습니다. 앞으로도 이러한 경험을 살려 더 나은 프로젝트를 수행하고 발전하는 데에 기여하고 싶습니다.