

## DevOps for SpringBoot

단계	주제	내용
[1]	리눅스 기초 I (명령어와 파일 시스템)	- 리눅스 디렉토리 구조 이해- 기본 명령어 (ls, cd, mkdir, rm, cp, mv, cat, chmod 등)- 사용자 및 퍼미션 이해- 실습: 파일 조작, 사용자 추가, 권한 변경
[2]	리눅스 기초 II (WSL2 네트워크 실습)	- 네트워크 명령어: ping, curl, ss, netstat, ip- 패키지 설치: apt update, apt install- 프로세스/포트 확인: ps aux, top, ss -tuln- 실습: 네트워크 점검, 패키지 관리, 실행 중인 프로세스 추적
[3]	Docker & Spring Boot 컨테이너화	- Docker 기본 구조 및 명령어 실습- Dockerfile 작성- Spring Boot 컨테이너화- Docker Compose 로 DB 연동 실습
[4]	Jenkins 설치 및 CI 구성	- Docker 로 Jenkins 설치- Jenkins 초기 설정 및 Git 연동- Jenkins Job 생성 및 빌드 자동화- Jenkinsfile 작성: Build + Test
[5]	Kubernetes 설치 및 수동 배포	- Minikube 설치 및 kubectl 설정- Kubernetes 개념 (Pod, Deployment, Service)- Spring Boot 앱 수동 배포 (YAML 작성)- 실습: 서비스 접속 및 로깅 확인
[6]	Jenkins → Kubernetes 자동 배포 (CD)	- Jenkins 에 kubectl 연동 (kubeconfig)- Jenkinsfile 수정: Build → Deploy 자동화- 실습: Git Push → Jenkins → K8s 자동배포 구성
[7]	Ingress-Nginx + 경로 기반 라우팅	- Ingress Controller 설치 (Minikube Addon)- 도메인/경로 기반 서비스 분리- 실습: /api, /admin, /user 등으로 분기 라우팅 구성
[8]	Prometheus + Grafana 모니터링	- Prometheus 설치 및 Spring Boot 와 연동- actuator, micrometer 설정- Grafana 설치 및 대시보드 구성- 실습: JVM 메모리, 요청 수, 응답 속도 시각화
[9]	전체 통합 배포 흐름 구성	- Git → Jenkins → Docker → K8s → Ingress → Grafana- 장애 복구 및 배포 실패 대응 실습- 실습: 실제 시나리오 기반 전체 배포 테스트

## 자동화 작업시 필요한 쉘 스크립트

상황	셸 스크립트 활용
Docker 컨테이너 일괄 빌드 및 실행	for 루프 + docker build, docker run
Kubernetes YAML 배포 자동화	kubectl apply -f 자동 처리
Jenkins Job 실행 후 로그 백업	cp, mv, tar, cron
로그 수집 및 모니터링 통계	grep, awk, sed 조합
서비스 상태 체크 및 재시작	if, systemctl, docker restart 등

## CI/CD 스크립트는 결국 셸 기반

- Jenkins 의 Jenkinsfile 은 기본적으로 **셸 명령을 기반으로 동작**
- GitHub Actions, GitLab CI 도 대부분 **bash 또는 sh** 로 작업 정의
- entrypt.sh 는 Docker 에서 컨테이너 실행 전 필요한 설정을 수행하는데 자주 사용됨

## Kubernetes & Docker 는 CLI 중심 도구

- kubectl, docker, helm 등은 모두 리눅스 CLI 기반
- bash, sh, zsh 를 능숙하게 다룰 수 있어야 문제 해결/자동화 가능

## 꼭 배워야 할 쉘 명령 / 스크립트 요소

분 류	명령어
기본 명령	cd, ls, cp, rm, cat, echo, touch
조건문	if, else, test, [ ], [[ ]]
반복문	for, while, until
함수	my_func() { ... }
변수 사용	VAR=value, \$VAR
입출력	>, >>, <, `
실무 도구	grep, awk, sed, cut, xargs

항목	awk	sed
주 사용 목적	필드(열) 처리	문자열 치환, 삭제
동작 방식	줄 단위 + 열 분석	줄 단위 문자열 검색/편집
대표 사용 예	특정 열 출력, 조건 검색	문자열 치환, 특정 줄 삭제

웹서버 종류	기본 경로
Apache	/var/log/apache2/access.log
Nginx	/var/log/nginx/access.log

**sudo cat /var/log/nginx/access.log | awk '{print \$1}'**

**# 또는**

**sudo cat /var/log/apache2/access.log | awk '{print \$1}'**

```
cat > access.log <<EOF
192.168.1.1 - - [18/Jun/2025:11:00:00 +0900] "GET /index.html HTTP/1.1" 200 1024
10.0.0.2 - - [18/Jun/2025:11:01:00 +0900] "POST /login HTTP/1.1" 302 512
EOF
awk '{print $1}' access.log
user01@Dominica:~$ cat > access.log <<EOF
/Jun/20> 192.168.1.1 - - [18/Jun/2025:11:00:00 +0900] "GET /index.html HTTP/1.1" 200 :
024
> 10.0.0.2 - - [18/Jun/2025:11:01:00 +0900] "POST /login HTTP/1.1" 302 512
> EOF
user01@Dominica:~$ awk '{print $1}' access.log
192.168.1.1
10.0.0.2
```

---

## [1] DevOps 실무 셸 명령어 20 선

### 1. 파일 및 디렉토리 관리

1. `cd /path/to/dir` # 디렉토리 이동
  2. `ls -al` # 상세 목록 보기
  3. `mkdir -p /path/to/dir` # 상위 디렉토리 포함 생성
  4. `rm -rf /path/to/dir_or_file` # 강제 삭제 (주의!)
- 

### 2. 파일 내용 확인 및 조작

5. `cat file.txt` # 전체 내용 출력
  6. `tail -f app.log` # 실시간 로그 보기
  7. `grep "ERROR" app.log` # 문자열 검색
  8. `awk '{print $1}' file.txt` # 컬럼 추출 (공백 기준)
  9. `sed -i 's/dev/prod/g' *.yaml` # 문자열 치환 (in-place 수정)
- 

### 3. 시스템 및 서비스 상태 확인

10. `top` # 실시간 CPU/메모리 확인
  11. `df -h` # 디스크 사용량
  12. `free -h` # 메모리 사용량
  13. `ps aux | grep nginx` # 실행 중인 프로세스 확인
  14. `netstat -tuln | grep 8080` # 포트 열림 확인
- 

### 4. Docker & Kubernetes 연계

15. `docker ps -a` # 컨테이너 목록
  16. `docker logs myapp` # 도커 로그 확인
  17. `kubectl get pods -A` # 전체 네임스페이스의 파드 상태
  18. `kubectl describe pod POD명` # 파드 상세 상태 확인
  19. `kubectl logs POD명` # 파드 로그 출력
- 

### 5. 자동화/스크립팅에 유용

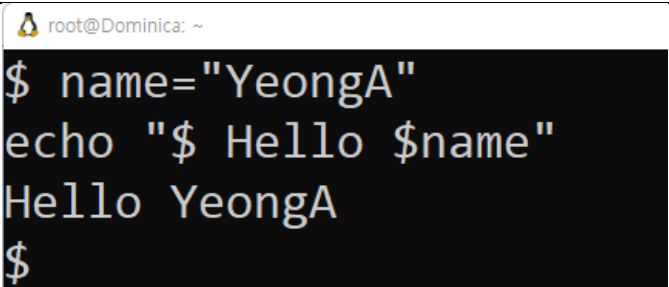
20. `for f in *.log; do grep "fail" "$f"; done` # 반복처리 예시
-

## 추가 팁: DevOps 용 alias 설정 예시

```
alias k='kubectl'
alias d='docker'
alias l='ls -alh'
alias kpods='kubectl get pods -A'
```

## [2] - 리눅스 셸 기초 문법

### 2.1 변수 (Variables)

<pre>name="YeongA" echo "Hello \$name"</pre>	 <pre>root@Dominica: ~ \$ name="YeongA" echo "\$ Hello \$name" Hello YeongA \$</pre>
--	--

- 규칙: 공백 없이 변수명=값 으로 설정
- echo \$변수명 으로 출력

## 연산자

### 산술 연산자 (정수 계산만 가능)

a=10	
b=3	
sum=\$((a + b))	# 덧셈
sub=\$((a - b))	# 뺄셈
mul=\$((a * b))	# 곱셈
div=\$((a / b))	# 나눗셈
mod=\$((a % b))	# 나머지

## ▶ 비교 연산자 (숫자 비교)

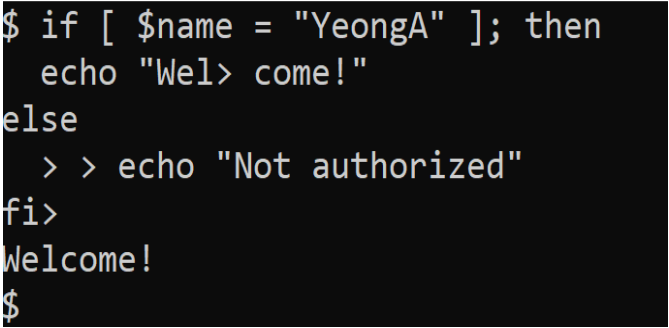
연산자	의미
-eq	같다 (equal)
-ne	같지 않다 (not equal)
-gt	크다 (greater than)
-lt	작다 (less than)
-ge	크거나 같다 (>=)
-le	작거나 같다 (<=)

## ▶ 문자열 비교

연산자	의미
=	같다
!=	다르다
-z	문자열 길이가 0 인가?
-n	문자열 길이가 0 이 아닌가?

## 2.2 조건문 (조건문 (if, elif, else) )

<pre>if [ 조건 ]; then     명령 1 elif [ 조건 ]; then     명령 2 else     명령 3 fi</pre>	<pre>score=85 if [ \$score -ge 90 ]; then     echo "A" elif [ \$score -ge 80 ]; then     echo "B" else     echo "C" fi</pre>
---	--

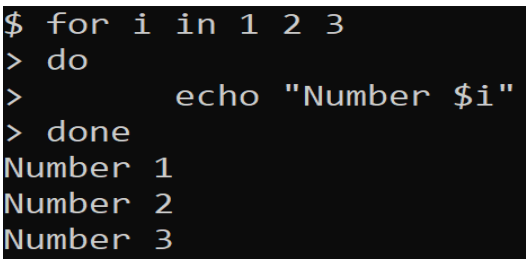
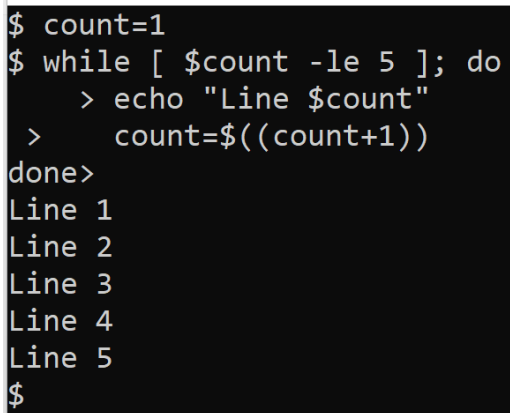
<pre>if [ \$name = "YeongA" ]; then     echo "Welcome!" else     echo "Not authorized" fi</pre>	 <pre>root@Dominica: ~ \$ if [ \$name = "YeongA" ]; then   echo "Welcome!" else   echo "Not authorized" fi Welcome! \$</pre>
---	--

## 2.3 반복문 (for / while)

```
for 변수 in 값 1 값 2 값 3 ...; do
    명령어
done
```

```
while [ 조건 ]; do
    명령어
done
```

```
until [ 조건 ]; do
    명령어
done
```

<pre>for i in 1 2 3 do     echo "Number \$i" done</pre>	 <pre>\$ for i in 1 2 3 &gt; do &gt;     echo "Number \$i" &gt; done Number 1 Number 2 Number 3</pre>
<pre>count=1 while [ \$count -le 5 ] do     echo "Line \$count"     count=\$((count+1)) done</pre>	 <pre>\$ count=1 \$ while [ \$count -le 5 ]; do &gt; echo "Line \$count" &gt;     count=\$((count+1)) done&gt; Line 1 Line 2 Line 3 Line 4 Line 5 \$</pre>

대괄호 안과 밖은 반드시 공백 있어야 함

= 앞뒤 공백 없음, 쉘에서는 공백이 명령어 구분이므로 오류 발생

## 2.4 break, continue

```
for i in 1 2 3 4 5; do
    if [ $i -eq 3 ]; then
        break
    fi
    echo "Loop $i"
done
```

```
for i in 1 2 3 4 5; do
    if [ $i -eq 3 ]; then
        continue
    fi
    echo "Loop $i"
done
```

## 2.5 함수 (function)

```
greet() {
    echo "Hi $1"
}
greet "YeongA"
```

```
$ greet(){
> echo "Hi $1"
> }
$ greet "Yeong A"
Hi Yeong A
$
```

- \$1, \$2, ... → 함수에 넘긴 인자
- return 문으로 종료할 수 있음 (return 0 등)

## 2-6 실행 권한 부여 및 실행

- chmod +x script.sh → 실행 권한 부여
- ./script.sh → 스크립트 실행
- bash -x script.sh → 디버깅 모드 실행



**nano loop.sh** 작업 후 **Ctrl + O → Enter → Ctrl + X**

```
root@Dominica: ~  
GNU nano 7.2  
#!/bin/bash  
  
count=1  
while [ $count -le 5 ]; do  
    echo "Line $count"  
    count=$((count+1))  
done
```

**실행 권한 부여**

**chmod +x loop.sh**

**./loop.sh**

```
$ chmod +x loop.sh  
$ ./loop.sh  
$ Line 1  
Line 2  
Line 3  
Line 4  
Line 5  
$
```

```
192.168.1.1 - - [18/Jun/2025:11:00:00 +0900] "GET /index.html HTTP/1.1" 200 1024  
10.0.0.2 - - [18/Jun/2025:11:01:00 +0900] "POST /login HTTP/1.1" 302 512  
172.16.0.3 - - [18/Jun/2025:11:02:00 +0900] "GET /dashboard HTTP/1.1" 200 1536  
192.168.1.1 - - [18/Jun/2025:11:03:00 +0900] "GET /about HTTP/1.1" 404 256
```

[AWK 실습 문제]

Q1) access.log 에서 접속한 IP 주소만 모두 출력하시오.

Q2) 접속한 요청 메서드(GET/POST 등)만 출력하시오.

Q3) 상태 코드가 200 인 줄만 출력하시오.

Q4) 각 상태 코드별 등장 횟수를 세어 출력하시오.

[SED 실습 문제]

Q5) access.log 에서 모든 GET 을 POST 로 변경하여 출력하시오.

Q6) access.log 에서 "192.168.1.1"이 등장하는 줄만 삭제하여 출력하시오.

Q7) 로그에서 HTTP 버전을 모두 "HTTP/2.0"으로 바꾸시오.

Q8) access.log 에서 3 번째 줄만 출력하시오.

Q9) access.log 에서 2~3 번째 줄을 삭제하여 출력하시오.

Q10) 모든 IP 주소를 "HIDDEN\_IP"로 마스킹하여 출력하시오.

### [3]. 추가적인 유용한 쉘 문법 6 가지

#### 3-1. 배열(Array)

```
fruits=("apple" "banana" "cherry")
echo ${fruits[0]}      # apple
echo ${#fruits[@]}     # 배열 길이
```

#### 3-2. case 문 (조건 분기 switch-case 스타일)

```
read -p "Enter a grade: " grade
case $grade in
  A) echo "Excellent" ;;
  B) echo "Good" ;;
  C) echo "Pass" ;;
  *) echo "Fail" ;;
esac
```

#### 3-3. 백틱(`) 또는 \$(...) 를 이용한 명령어 치환

```
now=$(date)
echo "Current time: $now"

# 또는
echo "Today is: `date`"
```

#### 3-4. 입력/출력 리디렉션

문법	설명
>	출력 덮어쓰기
>>	출력 이어쓰기 (append)
<	파일에서 입력 받기
`	`

```
echo "hello" > file.txt
cat file.txt | grep "hello"
```

---

### 3-5. 조건 표현식 확장 ([[ ... ]])

보다 복잡한 조건 처리 가능 (문자열 비교, 정규표현식 포함):

```
if [[ "$name" == Y* ]]; then
    echo "Starts with Y"
fi
```

---

### 3-6. 명령 실행 결과 확인 (\$?)

```
command
if [ $? -eq 0 ]; then
    echo "성공"
else
    echo "실패"
fi
```

\$? : 직전 명령의 **종료 상태코드** (0 이면 성공, 1 이상이면 오류)

---

Q1) 변수와 출력   파일명: q01\_hello\_name.sh

→ 본인의 이름을 변수에 저장하고, echo 를 이용해 "Hello, [이름]님!"을 출력하는 스크립트를 작성하시오.

Q2) 조건문   파일명: q02\_age\_check.sh

→ 사용자에게 나이를 입력받고, 20 세 이상이면 "성인", 미만이면 "미성년자"를 출력하시오.

Q3) 반복문 (for)   파일명: q03\_for\_loop.sh

→ 1 부터 5 까지 숫자를 출력하는 for 반복문을 작성하시오.

---

Q4) while 반복문 파일명: q04\_while\_even.sh

→ 숫자 1 부터 10 까지 짝수만 출력하는 while 문을 작성하시오.

Q5) 함수 파일명: q05\_greet\_function.sh

→ 이름을 입력받으면 "Hello, [이름]"을 출력하는 함수를 작성하고 호출하시오.

Q6) case 문 파일명: q06\_grade\_case.sh

→ 사용자로부터 성적(A, B, C, F)을 입력받아 다음과 같이 출력되도록 case 문을 작성하시오.

- A: Excellent
- B: Good
- C: Pass
- F: Fail
- 그 외: Invalid input

Q7) 파일 출력 리디렉션 파일명: q07\_redirect\_output.sh

→ date 명령어로 오늘 날짜를 today.txt 에 저장하고, 이어서 whoami 명령어 결과를 today.txt 에 추가하시오.

Q8) grep 과 pipe 파일명: q08\_passwd\_grep.sh

→ /etc/passwd 파일에서 "bash"라는 문자열을 포함한 줄만 출력하는 명령을 작성하시오.

Q9) 명령어 치환 파일명: q09\_pwd\_variable.sh

→ 현재 디렉토리 경로를 변수에 저장하고, "현재 위치는 [경로]입니다"라고 출력하시오.

Q10) 배열 파일명: q10\_array\_fruits.sh

→ "apple", "banana", "cherry"라는 과일을 배열로 선언하고, for 문을 사용하여 모든 요소를 한 줄씩 출력하시오.

---

---

#### [4]. builtin command , type -a echo

echo \$SHELL -> Bash 환경인지 확인하는 방법

```
$ bash
user01@Dominica:~$ type -a echo
echo is a shell builtin
echo is /usr/bin/echo
echo is /bin/echo
user01@Dominica:~$
```

**Bash(Bourne Again SHell)란?** 리눅스/유닉스에서 사용하는 대표적인 셸로 사용자가 입력한 명령을 해석하고 실행하는 "명령어 해석기(shell)" 중 하나이다

명령어	설명
echo	문자열 출력
read	사용자 입력 받기
cd	디렉토리 이동
pwd	현재 경로 출력
test / [ ]	조건 검사
export	환경 변수 설정
let, (( ))	산술 연산
eval	명령어 재해석 후 실행
alias, unalias	별칭 설정 및 해제
set, unset	셸 옵션 설정 및 변수 제거
type	명령의 종류 확인
source 또는 .	다른 스크립트 불러오기

항목	예시
기본 셸	/bin/bash
프롬프트	\$ 또는 [user@host ~]\$
프로파일	~/.bashrc, ~/.bash_profile
문법 지원	변수, 조건문, 반복문, 함수, 내장 명령 등

### .bashrc vs .bash\_profile 차이

파일명	실행 시점	용도
.bashrc	비로그인 셸 (터미널 창 열 때)	alias, 변수, 프롬프트 설정 등 자주 실행되는 명령
.bash_profile	로그인 셸 (SSH, TTY 로그인 등)	.bashrc 를 호출하거나 초기 실행 환경 설정용

~/.bashrc 예시 -> ※ ~는 현재 사용자 홈 디렉토리를 의미 (/home/사용자명)

<pre># ~/.bashrc  # 사용자 정의 alias alias ll='ls -alF' alias gs='git status'  # 환경 변수 export EDITOR=nano export JAVA_HOME=/usr/lib/jvm/java-21-openjdk-amd64  # 프롬프트 설정 PS1='Wu@Wh:WwW\$ '</pre>
---



## ~/.bash\_profile 예시 설정

```
# ~/.bash_profile

# 로그인 시 실행될 내용
echo "Welcome, $USER"

# bashrc 실행 연결
if [ -f ~/.bashrc ]; then
    source ~/.bashrc
fi
```

## 설정 적용하기 (즉시 반영)

`source ~/.bashrc` # 또는 `. ~/.bashrc`

```
-rw-r--r-- 1 user01 user01 3771 Mar 31 2024 .bashrc
drwxr-xr-x 2 user01 user01 4096 Jun 18 13:26 .landscape
-rw----- 1 user01 user01 20 Jun 18 15:24 .lessht
drwxrwxr-x 3 user01 user01 4096 Jun 18 17:37 .local
-rw-rw-r-- 1 user01 user01 0 Jun 18 13:26 .motd_shown
-rw-r--r-- 1 user01 user01 807 Mar 31 2024 .profile
-rw-r--r-- 1 user01 user01 0 Jun 18 14:14 .sudo_as_admin_s
-rw----- 1 user01 user01 4 Jun 18 15:06 a.txt
-rw-rw-r-- 1 user01 user01 46 Jun 18 15:26 log.txt
-rwxrwxr-x 1 user01 user01 99 Jun 18 17:38 loop.sh
user01@Dominica:~$ nano ~/.bashrc
user01@Dominica:~$ source ~/.bashrc
user01@Dominica:~$ hi
Hello, user01
```

`alias today='echo Today is $(date +"%Y-%m-%d")'` 추가해보자

## 등록된 alias 확인

```
user01@Dominica:~$ alias
alias alert='notify-send --urgency=low -i "${ $? = 0 } && echo terminal || echo error
)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\''")"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias hi='echo Hello, $USER'
alias ll='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

## Bash vs 다른 셸 비교

셸 종류	특징
bash	가장 널리 쓰이는 표준 셸, 문법이 풍부
sh (Bourne Shell)	오래된 기본 셸, bash 보다 기능 적음
zsh	자동 완성/하이라이트 기능 강력
fish	직관적인 문법과 GUI-like 환경
dash	아주 빠르지만 기능 최소화 (기본 /bin/sh 로 설정된 경우 많음)

## DevOps 실무용 alias 모음

# 시스템 모니터링	
alias dfh='df -h'	# 디스크 용량 확인
alias duh='du -sh *'	# 디렉토리별 용량
alias mem='free -h'	# 메모리 사용량
alias cpu='top'	# CPU 실시간 모니터링
alias ports='netstat -tulnp'	# 열려 있는 포트 보기
# 사용자 및 프로세스	
alias psg='ps aux   grep'	# 특정 프로세스 검색
alias who='w'	# 로그인 사용자 목록
# 로그 확인	
alias lsys='journalctl -xe'	# 시스템 로그
alias ltail='tail -n 100 -f /var/log/syslog' # 실시간 로그 모니터링	
# Bash 설정	
alias reload='source ~/.bashrc'	# 설정 재적용
alias bashrc='nano ~/.bashrc'	# .bashrc 바로 열기

## Docker 전용 alias 모음

```
# 컨테이너 관리
alias dps='docker ps'
alias dpsa='docker ps -a'
alias dstart='docker start'
alias dstop='docker stop'
alias drm='docker rm'
alias drmi='docker rmi'

# 이미지 관리
alias dimages='docker images'
alias dbuild='docker build -t'
alias dexec='docker exec -it'

# 전체 정리
alias dclean='docker system prune -af'

# 로그 보기
alias dlogs='docker logs -f'

# 실행 예시
# drun() 함수로 빠르게 실행
alias drun='docker run -it --rm'
```

## Kubernetes (kubectl) 전용 alias 모음

```
# 기본 명령
alias k='kubectl'
alias kgp='kubectl get pods'
alias kgs='kubectl get svc'
alias kgn='kubectl get nodes'
alias kga='kubectl get all'
alias kgns='kubectl get namespaces'

# 설명 및 로그
alias kd='kubectl describe'
alias kl='kubectl logs'
alias klf='kubectl logs -f'

# 배포 및 삭제
alias kap='kubectl apply -f'
alias kdelf='kubectl delete -f'
alias kdelp='kubectl delete pod'

# 네임스페이스 설정
alias kns='kubectl config set-context --current --namespace'

# 자동 완성 지원 (bash-completion 설치 후)
source <(kubectl completion bash)
```

## DevOps 업무에서 네트워크 확인

확인 목적	설명
서버 접속 가능 여부	포트가 열렸는지, 방화벽 설정이 맞는지
컨테이너/VM 통신 확인	Docker, Kubernetes 내 컨테이너 간 연결 가능 여부
서비스 상태 점검	nginx, DB, API 등 서비스가 외부 요청을 받는지
DNS / 이름 해석 문제 확인	ping, nslookup, dig 등을 통해 도메인 이름 해석 문제 점검
지연 시간 / 속도 분석	ping, traceroute, mtr 등을 통해 병목 구간 파악
방화벽 / 보안 설정 검증	nmap 등을 통해 외부 접근 가능 여부 점검

## 자주 쓰는 네트워크 확인 명령어

명령어	설명
ping [주소]	대상 서버와 연결 가능한지 확인
curl [주소:포트]	웹/API 포트 확인 (예: curl localhost:8080)
netstat -tulpn	열려 있는 포트 목록 확인
ss -tulpn	netstat 보다 빠르고 현대적인 대체 명령
telnet [IP] [PORT]	특정 포트로 연결 가능한지 확인
nc -zv [IP] [PORT]	netcat 으로 포트 열림 여부 테스트
traceroute [주소]	네트워크 경로 추적
nslookup [도메인]	도메인 이름 → IP 해석 확인
dig [도메인]	nslookup 보다 더 상세한 DNS 확인
ip a 또는 ifconfig	서버의 IP 주소 및 인터페이스 확인

```

user01@Dominica:~$ ping google.com
curl -I http://localhost:3000
ss -tulpn | grep 8080
telnet 192.168.0.10 22
nc -zv 192.168.0.10 80
PING google.com (142.251.42.174) 56(84) bytes of data.
64 bytes from nrt12s46-in-f14.1e100.net (142.251.42.174): icmp_seq=1 ttl=112 time=36.6
ms

```