

tee 기능 설명



tee 기능에 대해 알려줘.

예) `kubectl get po -A | tee out_file.txt`



‘tee’는 파이프라인에서 많이 쓰는 유틸리티입니다. 이름 그대로 “삼지창(tee)” 모양처럼 **표준출력(stdout)**을 두 갈래로 흘려보내요.

당신이 예로 든

```
kubectl get po -A | tee out_file.txt
```

를 보면:

1. ‘`kubectl get po -A`’ → 모든 네임스페이스의 파드 목록을 표준출력으로 뿌림.
2. ‘`tee out_file.txt`’ → 그 출력을 **화면(표준출력)**에도 그대로 보여주고, 동시에 ‘`out_file.txt`’라는 파일에도 저장함.

즉, ‘>’ 리다이렉션과 달리 **화면에서도 결과를 확인하면서 파일에도 기록**할 수 있는 게 핵심이에요.

몇 가지 팁도 있어요:

- ‘`tee -a out_file.txt`’ : 덮어쓰기 대신 **append(추가)** 모드로 기록.
- 여러 파일에 동시에 저장 가능:

```
kubectl get po -A | tee out1.txt out2.txt
```

- 다른 명령과 조합해 로그 모으기에도 자주 씬. 예:

```
tail -f /var/log/syslog | tee syslog_copy.log
```

👉 정리하면: ‘tee’는 출력 흐름을 "보면서 동시에 기록"하는 도구.

혹시 이걸 파이프 안에서 중간 결과를 확인하는 용도로 쓰는 방법까지도 보고 싶으세요?



응



좋습니다 😊

`tee`는 파이프 중간에서 “옆길로 분기”할 수 있기 때문에 **중간 결과를 눈으로 확인하거나 따로 저장**하는 용도로 자주 씁니다.

예시 몇 가지 보여드릴게요.

1. 중간 파이프라인 확인

```
kubectl get po -A \  
| tee step1.txt \  
| grep Running \  
| tee step2.txt \  
| wc -l
```

- `step1.txt` → 전체 파드 목록 저장
- `step2.txt` → `Running` 상태만 필터링된 목록 저장
- 마지막에 `wc -l`로 개수 출력

즉, 중간마다 결과를 파일로 떨어뜨리면서 최종 결과까지 이어갈 수 있어요.

2. 로그 분석할 때

```
cat app.log \  
| grep ERROR \  
| tee error.log \  
| cut -d' ' -f1 \  
| sort | uniq -c
```

- 에러 로그 전체는 `error.log`에 저장
- 동시에 시간대별 발생 빈도 집계까지 이어감

3. 실시간 모니터링 + 저장

```
kubectl logs -f mypod \  
| tee live.log
```

→ 화면에서 로그를 보면서 동시에 파일에 기록. (나중에 grep으로 분석 가능)

요약하면, `tee`는 단순히 출력 복제만 하는데도 **파이프라인 중간 체크포인트**를 만들 수 있어서 디버깅이나 로그 수집에 꽤 유용합니다.

혹시 원하세요? 제가 `kubectl` 환경에서 자주 쓰이는 `tee` 활용 패턴만 따로 정리해드릴까요?



응. 좋아, 항상 예제는 kubernetes운영 또는 airflow 등의 예제로 해주면 좋지.



좋습니다 👍

쿠버네티스 운영이나 Airflow 로그/잡 운영을 하면서 `tee`를 쓰는 패턴 위주로 정리해드릴게요.

◆ `tee` 활용 패턴 (Kubernetes & Airflow 예시)

1. 파드 상태 기록 + 필터링

```
kubectl get po -A \  
| tee all_pods.txt \  
| grep CrashLoopBackOff \  
| tee crash_pods.txt
```

- `all_pods.txt` : 전체 파드 스냅샷 저장
- `crash_pods.txt` : CrashLoopBackOff 상태만 따로 저장
👉 운영 중 장애 상황 파악할 때 유용합니다.

2. 이벤트 로그 모니터링 + 저장

```
kubectl get events -A --sort-by='.lastTimestamp' -w \
| tee cluster_events.log
```

- 화면에서 이벤트 흐름을 실시간 확인
- 동시에 `cluster_events.log`에 저장해서 사후 분석 가능

3. Airflow DAG 실행 로그 추적

```
airflow tasks test example_dag task1 2025-09-07 \
| tee task1_test.log
```

- 화면에서 DAG 태스크 로그를 보면서
- 그대로 파일에도 저장 → Slack 업로드, grep 분석 등에 활용

4. Airflow Scheduler/Worker 로그 분석

```
kubectl logs -f deploy/airflow-scheduler -n airflow \
| tee scheduler.log \
| grep -i dag
```

- `scheduler.log` : 전체 로그 저장
- 화면에서는 DAG 관련 로그만 바로 확인 가능

5. Helm 배포 디버깅

```
helm upgrade airflow apache/airflow -f values.yaml \
| tee helm_upgrade.log
```

- Helm 배포 결과를 저장해두면 rollback 할 때 참고하기 좋아요.

💡 요약

- `tee`는 “운영 중 화면 확인 + 사후 분석용 로그 저장”이라는 두 마리 토끼를 잡을 때 최적.
- 특히 **kubectl / airflow cli / helm** 같은 운영 명령어랑 찰떡궁합.

- 파일은 ``-a`` 옵션으로 누적 저장도 가능 (``tee -a file.log``).

혹시 제가 위 예제들을 **운영 환경용 치트시트 (한 페이지 PDF)** 형태로 정리해드리면 더 편하실까요?



내용 보강해서 pdf 로 내가 출력할께. 너는 그냥 text로 내용 정리만 해줘.



좋습니다 😊 운영 환경에서 바로 참고할 수 있도록, ``tee`` 활용 패턴을 정리된 텍스트 매뉴얼 스타일로 보강해드릴게요.

Kubernetes & Airflow 운영에서의 ``tee`` 활용 매뉴얼

1. 기본 개념

- ``tee`` 명령은 **표준출력(stdout)**을 화면에 그대로 보여주면서 동시에 파일에도 기록하는 기능을 제공.
- 단일 파일, 여러 파일, 누적 모드(``-a``) 모두 지원.
- 운영 환경에서 실시간 로그 확인, 중간 파이프라인 결과 저장, 장애 분석 기록에 자주 활용됨.

2. 주요 옵션

- ``tee file.txt`` → 파일에 저장 (덮어쓰기)
- ``tee -a file.txt`` → 파일에 추가 (append)
- ``tee file1.txt file2.txt`` → 여러 파일에 동시에 저장

3. Kubernetes 운영 예제

3.1 파드 전체 스냅샷 + 장애 파드 분리

```
kubectl get po -A \
| tee all_pods.txt \
| grep CrashLoopBackOff \
| tee crash_pods.txt
```

- 전체 파드 목록(`all_pods.txt`) 저장
- 장애 파드만 별도 추출(`crash_pods.txt`)

3.2 이벤트 모니터링 + 저장

```
kubectl get events -A --sort-by='.lastTimestamp' -w \
| tee cluster_events.log
```

- 화면에서 이벤트 실시간 확인
- 동시에 `cluster_events.log`에 기록 → 사후 장애 원인 분석 가능

3.3 로그 확인 및 분석

```
kubectl logs -f deploy/airflow-scheduler -n airflow \
| tee scheduler.log \
| grep -i dag
```

- Airflow 스케줄러 로그 전체는 `scheduler.log`에 저장
- 화면에는 `DAG` 관련 로그만 실시간 표시

3.4 Helm 배포 결과 저장

```
helm upgrade airflow apache/airflow -f values.yaml \
| tee helm_upgrade.log
```

- 배포 결과를 `helm_upgrade.log`에 저장 → 롤백/장애 대응 시 참고 자료 확보

4. Airflow 운영 예제

4.1 DAG Task 단위 테스트

```
airflow tasks test example_dag task1 2025-09-07 \  
| tee task1_test.log
```

- 화면에서 테스트 로그 확인
- `task1_test.log`에 저장 → 나중에 grep/분석 가능

4.2 Worker 로그 추적

```
kubectl logs -f pod/airflow-worker-0 -n airflow \  
| tee worker0.log
```

- 워커 로그를 보면서 동시에 파일에도 기록
- 장애 시 해당 로그를 팀 공유 가능

5. 실무 활용 팁

- 중간 파이프라인 체크포인트

```
kubectl get po -A \  
| tee step1.txt \  
| grep Running \  
| tee step2.txt \  
| wc -l
```

→ 단계별 결과 저장하면서 최종 카운트 계산

- 실시간 모니터링 + 장기 보관

```
kubectl logs -f mypod | tee -a mypod_full.log
```

→ 화면에서 계속 확인하면서, 파일에는 누적 저장

- 분석/리포트 자동화 연계

저장된 로그를 `grep`, `awk`, `wc`, `jq` 등과 함께 후처리 가능

6. 정리

- `tee` = “보면서 기록하기”
- 쿠버네티스/에어플로우 운영에서 특히 유용한 순간:
 - 장애 상황 → 로그 확인 + 저장
 - 배포/업그레이드 → 결과 아카이빙
 - DAG/Task 테스트 → 증적 확보
- 다른 명령과 조합해 **실시간 디버깅 + 사후분석** 모두 가능

이렇게 정리된 텍스트를 바로 PDF로 출력하시면 될 겁니다.

👉 원하시면, 위 내용을 **표 형식(명령어 / 설명 / 활용처)** 으로 다시 재구성해드릴 수도 있습니다. 그렇게 하면 매뉴얼 느낌이 더 깔끔해지거든요. 원하실까요?



응.



좋습니다 😊 이번엔 표 형식으로 깔끔하게 정리해드릴게요. 그대로 PDF로 출력하면 매뉴얼 느낌이 날 겁니다.

📖 Kubernetes & Airflow 운영에서의 `tee` 활용 매뉴얼 (표 정리)

명령어 예제	설명	활용처
<code>`kubectl get po -A tee all_pods.txt`</code>	모든 네임스페이스 파드 목록을 화면에 출력하면서 <code>`all_pods.txt`</code> 에 저장	전체 파드 스냅샷 확보
<code>`kubectl get po -A tee all_pods.txt grep CrashLoopBackOff tee crash_pods.txt`</code>	전체 목록 저장 + CrashLoopBackOff 파드만 따로 저장	장애 파드 추출 및 분석
<code>`kubectl get events -A --sort-by='.lastTimestamp' -w tee cluster_events.log`</code>	이벤트를 실시간 모니터링하면서 파일에도 기록	장애 원인 추적, 이벤트 히스토리 저장

명령어 예제	설명	활용처
<code>`kubectl logs -f deploy/airflow-scheduler -n airflow tee scheduler.log grep -i dag`</code>	Airflow 스케줄러 로그 전체는 파일 저장, 화면엔 DAG 관련 로그만 출력	DAG 디버깅 및 로그 아카이빙
<code>`kubectl logs -f pod/airflow-worker-0 -n airflow tee worker0.log`</code>	워커 로그를 실시간 확인하며 파일에도 저장	워커 장애 분석, 팀 공유 로그
<code>`helm upgrade airflow apache/airflow -f values.yaml tee helm_upgrade.log`</code>	Helm 배포 결과를 화면 확인 + 파일 저장	배포 결과 기록, 롤백 시 참고 자료
<code>`airflow tasks test example_dag task1 2025-09-07 tee task1_test.log`</code>	Airflow DAG 태스크 단위 테스트 로그 저장	DAG 디버깅, 테스트 결과 보관
<code>`kubectl get po -A tee step1.txt grep Running tee step2.txt wc -l`</code>	중간 결과를 단계별 파일로 저장, 최종적으로 Running 파드 수 계산	디버깅, 점진적 필터링
<code>`kubectl logs -f mypod tee -a mypod_full.log`</code>	실시간 로그를 확인하면서 파일에 누적 저장	장기 로그 수집, 장애 재현용

👉 이 표 하나면 운영 매뉴얼 느낌이 딱 납니다.

여기서 더 다듬자면, “**쿠버네티스 운영**”, “**Airflow 운영**”, “**Helm 배포**” 이렇게 섹션을 나눠서 표를 2~3개로 쪼개도 깔끔할 수 있어요.

원하실까요, 아니면 지금처럼 한 장짜리 요약표가 나을까요?



구분해서 정리해보자.



좋습니다 👍 이번엔 **구분별(쿠버네티스 / Airflow / Helm)** 로 표를 나눠 정리해드릴게요. PDF로 출력하면 섹션별 매뉴얼처럼 보일 겁니다.



Kubernetes & Airflow 운영에서의 `tee` 활용 매뉴얼



◆ Kubernetes 운영에서 `tee`

명령어 예제	설명	활용처
<code>`kubectl get po -A tee all_pods.txt`</code>	모든 네임스페이스 파드 목록을 저장 하면서 화면 출력	전체 파드 스냅 샷 확보
<code>`kubectl get po -A tee all_pods.txt grep CrashLoopBackOff tee crash_pods.txt`</code>	전체 목록 저장 + CrashLoopBackOff 파드만 따로 저 장	장애 파드 추출 및 분석
<code>`kubectl get events -A --sort- by='.lastTimestamp' -w tee cluster_events.log`</code>	이벤트를 실시간 모니터링 + 파일 저 장	장애 원인 추적, 이벤트 히스토 리
<code>`kubectl logs -f pod/mypod -n default tee -a mypod.log`</code>	파드 로그를 실시간 확인 + 파일 누적 저장	장기 로그 수집, 장애 재현

◆ Airflow 운영에서 `tee`

명령어 예제	설명	활용처
<code>`airflow tasks test example_dag task1 2025-09-07 tee task1_test.log`</code>	DAG 태스크 단위 테스트 로그 저 장	DAG 디버깅, 테 스트 결과 보관
<code>`kubectl logs -f deploy/airflow-scheduler -n airflow tee scheduler.log grep -i dag`</code>	Airflow 스케줄러 로그 저장 + 화 면에선 DAG 관련 로그만 출력	DAG 모니터링 및 로그 아카이빙
<code>`kubectl logs -f pod/airflow-worker-0 -n airflow tee worker0.log`</code>	워커 로그 실시간 확인 + 파일 저 장	워커 장애 분석, 팀 공유 로그

◆ Helm 배포/업그레이드에서 `tee`

명령어 예제	설명	활용처
<code>`helm upgrade airflow apache/airflow -f values.yaml tee helm_upgrade.log`</code>	Helm 배포 결과를 화면 확 인 + 파일 저장	배포 증적 확보, 롤 백 참고 자료
<code>`helm install myapp ./chart tee install_result.log`</code>	신규 설치 결과 기록	설치 이력 관리, 디 버깅

◆ 파이프라인 중간 디버깅 패턴

명령어 예제	설명	활용처
<code>`kubect1 get po -A tee step1.txt grep Running tee step2.txt wc -l`</code>	중간 결과를 단계별 저장, 최종적으로 Running 파드 개수 확인	디버깅, 점진적 필터링
<code>`tail -f /var/log/syslog tee syslog_copy.log`</code>	시스템 로그 실시간 확인 + 복사 저장	운영 서버 로그 아카이빙

👉 이렇게 나누면 섹션별로 찾기 쉽고, 운영 매뉴얼처럼 깔끔합니다.

혹시 이 표에다가 추가로 “옵션 요약(예: **-a append**)” 같은 작은 부록도 붙여드릴까요?