

Timestamp 추가 방법



쿠버네티스 cm에서 get할때 timestamp 컬럼도 추가하고싶어.

ConfigMap(cm)은 쿠버네티스 오브젝트라서 `kubectl get cm` 하면 **NAME, DATA, AGE** 같은 기본 컬럼만 보여주죠. **timestamp**는 기본 컬럼에 없어서 안 나옵니다. 대신 `--output` 옵션을 써야 해요.

예시 몇 가지 드릴게요.

1. kubectl get cm -o jsonpath

ConfigMap 생성 시간을 찍고 싶으면:

```
bash

kubectl get cm <cm-name> -o jsonpath='{.metadata.creationTimestamp}'
```

2. 목록에 timestamp 컬럼 추가 (custom-columns)

여러 ConfigMap을 표로 보고 싶을 때:

```
bash

kubectl get cm -o custom-columns=NAME:.metadata.name,CREATED:.metadata.creationTimestamp
```

결과 예시:

```
arduino

NAME          CREATED
app-config    2025-09-04T00:20:33Z
db-config     2025-09-01T08:11:52Z
```

3. 전체 리소스 공통 (kubectl get ... --show-labels 같이)

ConfigMap뿐 아니라 Secret, Deployment 등도 같은 방식으로 쓸 수 있습니다:

```
bash

kubectl get secret -o custom-columns=NAME:.metadata.name,TIMESTAMP:.metadata.creationTimestamp
```

👉 질문 하나만 확인 드릴게요. 원하시는 게 **ConfigMap 생성 시각(creationTimestamp)**을 추가하는 건가요, 아니면 ****마지막 수정된 시각(last applied / resourceVersion 기반)****을 보고 싶으신 건가요?

- 생성 시각은 위 방법으로 가능
- 수정 시각은 이벤트/리소스버전 기준이라 별도 확인이 필요해요 (`kubectl get -o yaml | grep resourceVersion` 식)



name, age, timestamp 나오게 해줘

네, `kubectl get cm` 출력에 기본으로 **NAME**, **AGE**는 나오는데 **TIMESTAMP**는 따로 붙여줘야 합니다. `custom-columns` 옵션으로 원하는 컬럼만 정리하면 됩니다.

예시:

```
bash

kubectl get cm -o custom-
columns=NAME:.metadata.name,AGE:.metadata.creationTimestamp,TIMESTAMP:.metadata.creationTimestamp
```

다만 여기서 주의할 점은:

- **AGE**는 원래 `kubectl`이 계산해서 보여주는 값이고, JSONPath로는 바로 못 뽑습니다.
- `creationTimestamp`는 ISO8601 시각 그대로 찍힙니다.

그래서 아래처럼 두 가지 버전을 나눠 보는 게 보통 편해요:

1. NAME + AGE + TIMESTAMP 같이 보기

```
bash

kubectl get cm -o custom-columns=NAME:.metadata.name,TIMESTAMP:.metadata.creationTimestamp --
sort-by=.metadata.creationTimestamp
```

→ 여기서 `kubectl get cm` 기본 출력의 AGE 컬럼도 같이 나옵니다.

2. 완전히 내가 정의한 컬럼만 쓰고 싶을 때

AGE를 계산해주는 기능은 `kubectl get` 기본 테이블에서만 지원돼요. JSONPath로는 구현이 안 되니까, 그 경우는 **TIMESTAMP**만 가져오고 AGE는 못 붙입니다.

👉 정리하면, **NAME / AGE / TIMESTAMP** 세 가지를 한 화면에서 보려면 `kubectl get cm -o custom-columns=...에 TIMESTAMP만 추가`해주시면 됩니다. 그러면 NAME, AGE(기본), TIMESTAMP가 같이 출력됩니다:

```
bash

kubectl get cm -o custom-columns=NAME:.metadata.name,TIMESTAMP:.metadata.creationTimestamp
```

혹시 원하시는 출력 형태가 ****kubectl 기본 AGE(몇일 전, 몇분 전)****로 꼭 같이 나와야 하는 건가요, 아니면 AGE도 타임스탬프로 직접 찍어도 괜찮으신 건가요?



내가 원하는건 파드에서 사용하지 않는 cm을 지우고 싶어. 루프로 cm조회해서 하루 이상된 cm을 대상으로 파드에서 사용하지 않으면 삭제하는 쉘

좋아요. "하루 이상 된 ConfigMap 중에서, 현재 어떤 파드에서도 참조하지 않는 것만 삭제"하는 안전한 쉘 스크립트 드릴게요.

기본은 드라이런(dry-run)이고, **--apply**를 붙여야 실제 삭제합니다. 네임스페이스는 **-n**으로 지정(기본은 **default**).

```
bash

#!/usr/bin/env bash
# cm-gc.sh - Garbage-collect unused ConfigMaps older than a day
# 요구: kubectl, jq, GNU date(리눅스 기본). macOS는 coreutils 설치 후 gdate 사용 또는 아래 DATE_CMD 수정.

set -euo pipefail

NS="default"
OLDER_THAN_SECONDS=$((24*60*60)) # 1 day
APPLY=false
INCLUDE_CONTROLLERS=false # Deployment/STS/DS/Job/CronJob 템플릿까지 검사하려면 true 또는 -include-controllers
DATE_CMD="date" # macOS면: DATE_CMD="gdate"

usage() {
    cat <<EOF
사용법: $0 [-n NAMESPACE] [--older-than-seconds SEC] [--apply] [--include-controllers]
예시 : $0 -n data --apply
EOF
}

while [[ $# -gt 0 ]]; do
    case "$1" in
        -n|--namespace) NS="$2"; shift 2 ;;
        --older-than-seconds) OLDER_THAN_SECONDS="$2"; shift 2 ;;
        --apply) APPLY=true; shift ;;
        --include-controllers) INCLUDE_CONTROLLERS=true; shift ;;
        -h|--help) usage; exit 0 ;;
        *) echo "알 수 없는 옵션: $1"; usage; exit 1 ;;
    esac
done

# 날짜 → epoch(sec)
to_epoch() {
    # RFC3339 예: 2025-09-04T00:20:33Z
    $DATE_CMD -u -d "$1" +%s
}

now_epoch=$(to_epoch "$DATE_CMD -u +%s")
cutoff_epoch=$(( now_epoch - OLDER_THAN_SECONDS ))

echo "# 네임스페이스: $NS"
echo "# 기준: $(to_epoch "$DATE_CMD -u -d @$cutoff_epoch" +%Y-%m-%dT%H:%M:%SZ)) 이전(older-than ${OLDER_THAN_SECONDS}s)"
echo "# 컨트롤러 템플릿 검사: $INCLUDE_CONTROLLERS"
echo "# 실행 모드: $([[ "$APPLY" == true ]] && echo APPLY || echo DRY-RUN)"
echo

pods_json=$(kubectl get pods -n "$NS" -o json)
controllers_json=""
if [[ "$INCLUDE_CONTROLLERS" == true ]]; then
    controllers_json=$(kubectl get deploy,rs,statefulset,daemonset,job,cronjob -n "$NS" -o json 2>/dev/null || echo '{"items":[]}')
fi
```

```

is_used_by_pods() {
    local cm="$1"
    echo "$pods_json" | jq -e --arg cm "$cm" '
        .items[]? |
        (
            # volumes: configMap / projected.sources[].configMap
            ( .spec.volumes[]? | select(.configMap?.name==$cm) ) or
            ( .spec.volumes[]? | .projected?.sources[]? | select(.configMap?.name==$cm) ) or
            # containers/initContainers: envFrom.configMapRef / env.valueFrom.configMapKeyRef
            ( [ .spec.initContainers[]?, .spec.containers[]? ][]?
              | ( .envFrom[]? | select(.configMapRef?.name==$cm) )
                or ( .env[]? | select(.valueFrom?.configMapKeyRef?.name==$cm) )
            )
        )
    ' >/dev/null
}

is_used_by_controllers() {
    local cm="$1"
    [[ "$INCLUDE_CONTROLLERS" != true ]] && return 1
    echo "$controllers_json" | jq -e --arg cm "$cm" '
        .items[]? |
        .spec.template as $t |
        (
            ( $t.spec.volumes[]? | select(.configMap?.name==$cm) ) or
            ( $t.spec.volumes[]? | .projected?.sources[]? | select(.configMap?.name==$cm) ) or
            ( [ $t.spec.initContainers[]?, $t.spec.containers[]? ][]?
              | ( .envFrom[]? | select(.configMapRef?.name==$cm) )
                or ( .env[]? | select(.valueFrom?.configMapKeyRef?.name==$cm) )
            )
        )
    ' >/dev/null
}

# 하루 이상된 CM 목록 뽑기
mapfile -t candidates <<(
    kubectl get cm -n "$NS" -o json \
    | jq -r '
        .items[]
        | [ .metadata.name, .metadata.creationTimestamp ]
        | @tsv
    ' \
    | while IFS=$'\t' read -r name ts; do
        [[ -z "$ts" ]] && continue
        epoch=$(to_epoch "$ts")
        if (( epoch <= cutoff_epoch )); then
            printf "%s\t%s\t%s\n" "$name" "$ts" "$epoch"
        fi
    done \
    | sort -k3,3n
)

if [[ ${#candidates[@]} -eq 0 ]]; then
    echo "대상 없음: 하루 이상된 ConfigMap이 없습니다."
    exit 0
fi

printf "검사 대상(%d개):\n" "${#candidates[@]}"
printf " %s %s %s\n" "NAME" "CREATED" "STATUS"
for line in "${candidates[@]}; do
    name=$(cut -f1 <<<"$line")
    ts=$(cut -f2 <<<"$line")

    status="UNUSED"
    if is_used_by_pods "$name"; then
        status="IN-USE:pod"
    elif is_used_by_controllers "$name"; then
        status="IN-USE:controller"
    fi

    printf " %s %s %s\n" "$name" "$ts" "$status"

    if [[ "$status" == UNUSED ]]; then

```

```
if [[ "$APPLY" == true ]]; then
    kubectl delete cm "$name" -n "$NS"
else
    echo "    (dry-run) kubectl delete cm $name -n $NS"
fi
fi
done
```

쓰는 법

```
bash

chmod +x cm-gc.sh

# 1) 드라이런으로 확인
./cm-gc.sh -n my-namespace
# 또는 36시간 기준
./cm-gc.sh -n my-namespace --older-than-seconds $((36*3600))

# 2) 실제 삭제 실행
./cm-gc.sh -n my-namespace --apply

# 3) 디플로이먼트/잡 등의 템플릿까지(미래 파드 위험까지) 검사
./cm-gc.sh -n my-namespace --include-controllers --apply
```

안전장치 팁

- **라벨로 보호:** 중요한 CM에 **keep=true** 라벨 달고, 스크립트에서 **kubectl get cm -l 'keep!=true'** 식으로 필터링해도 좋아요.
- **리포트 모드:** 운영이라면 처음엔 **--apply** 없이 결과만 슬랙/메일로 공유하고 다음 원도에 적용하세요.
- **크론 등록:** 주간/야간 배치로 돌릴 땐 **--include-controllers**를 켜 상태로 드라이런 리포트 → 수동 승인 후 실행, 이 루틴이 깔끔합니다.

필요하면 **AGE**(몇 분/시간/일 전) 계산 칼럼도 출력해 드릴게요. 원하는 출력 포맷 있으면 말만 해요.



다시 작성해보자.dry run은 빠고 심플하게 하지. 모든 pod정보는 cat으로 podinfo.yaml 로 생성하고 그 파일을 grep해서 사용여부 판단하는걸로

좋습니다. **심플 + 바로 삭제** 버전으로 정리했어요.

전제: 먼저 파드 정보를 파일로 만들어 둡니다.

```
bash

# 네임스페이스 예: data
kubectl get pods -n data -o yaml > podinfo.yaml
```

이제 아래 스크립트로 **“하루(24h) 이상 된 CM 중에서, 현재 파드에서 사용 흔적이 없는 것만 삭제”**합니다.

요구하신 대로 **podinfo.yaml**만 **grep**해서 사용여부 판단합니다.

```
bash

#!/usr/bin/env bash
# cm-gc-simple.sh
# 요구 도구: kubectl, jq, GNU date

set -euo pipefail

NS="${1:-default}"          # 첫 번째 인자로 네임스페이스 받음 (기본 default)
PODINFO_FILE="podinfo.yaml"
CUTOFF_HOURS="${CUTOFF_HOURS:-24}"    # 환경변수로 조정 가능 (기본 24h)
OLDER_THAN_SEC=$(( CUTOFF_HOURS * 3600 ))

if [[ ! -f "$PODINFO_FILE" ]]; then
    echo "[${PODINFO_FILE}] 파일이 없습니다. 먼저: kubectl get pods -n $NS -o yaml > $PODINFO_FILE"
    exit 1
fi

# configMap 관련 라인 주변만 추출(간단 grep용 버퍼)
# - configMap / configMapRef / configMapKeyRef / projected 근처 3줄씩
awk '
/configMap|configMapRef|configMapKeyRef|projected/ {for(i=0;i<4;i++){print;
if(getline<=0)break}}
' "$PODINFO_FILE" > cmrefs.txt

now_epoch=$(date -u +%s)
cutoff_epoch=$(( now_epoch - OLDER_THAN_SEC ))

# 하루 이상된 CM 리스트(name \t timestamp)
kubectl get cm -n "$NS" -o json \
| jq -r '.items[] | [.metadata.name, .metadata.creationTimestamp] | @tsv' \
| while IFS=$'\t' read -r name ts; do
    # 보호: 시스템/자동 CM는 건너뛰기(원하면 목록에 추가)
    case "$name" in
        kube-root-ca.crt|kube-public-configmap) continue ;;
    esac

    # timestamp 없는 CM는 스킵
    [[ -z "${ts:-}" ]] && continue

    cm_epoch=$(date -u -d "$ts" +%s)
    # 하루 이상만 대상
    if (( cm_epoch <= cutoff_epoch )); then
        # podinfo.yaml에서 사용 흔적 grep
        # name: <cm> 형태만 검사(간단 매칭). false positive 줄이려 cmrefs.txt로 한정
        if grep -Eq "^[:space:]*name[:space:]*$name$" cmrefs.txt; then
            echo "[KEEP] $name (in-use) created=$ts"
        else
            echo "[DELETE] $name (unused) created=$ts"
            kubectl delete cm "$name" -n "$NS"
        fi
    fi
done
```

사용법

```
bash

# 1) 파드 스냅샷 만들기
kubectl get pods -n my-namespace -o yaml > podinfo.yaml

# 2) 스크립트 실행 (실제 삭제함)
chmod +x cm-gc-simple.sh
./cm-gc-simple.sh my-namespace
```

```
# 3) 36시간 기준으로 바꾸고 싶으면  
CUTOFF_HOURS=36 ./cm-gc-simple.sh my-namespace
```

메모 한 줄

- 이 스크립트는 **파드 스냅샷 시점만** 봅니다. 새로 뜬 파드는 반영 안 되니, 실행 직전에 **podinfo.yaml** 다시 만드는 게 안전합니다.
- **grep** 단순 매칭이라 **동명이인(다른 필드의 name)** 잡음이 드물게 섞일 수 있어요. 그래서 **cmrefs.txt**(configMap 관련 블록만 추림)로 범위를 줄였습니다.
- 꼭 지키고 싶은 CM은 라벨링 말고도, 위 **case** 문에 이름을 추가하면 확실합니다.