

# ArgoCD PreSync Hook 설명



argoCD 의 hook preSync yaml 파일인데, 최대한 기초적인 것부터 설명 부탁해.

```
{{- if .Values.preCheck.enabled }}
apiVersion: batch/v1
kind: Job
metadata:
  #name: {{ include "chart.fullname" . }}-
presync
  name: {{ .Release.Name }}-presync
  annotations:
    argocd.argoproj.io/hook: PreSync
    #argocd.argoproj.io/hook-delete-policy:
HookSucceeded,HookFailed
    # 다음 배포에서 같은 hook을 다시 만들기 직전에만 삭제
    (그 외에는 남겨둠)
    argocd.argoproj.io/hook-delete-policy:
BeforeHookCreation
spec:
  backoffLimit: {{
.Values.preCheck.backoffLimit | default 1 }}
  template:
    spec:
      restartPolicy: Never
      volumes:
        - name: hook-scripts
          configMap:
            name: {{ .Release.Name }}-hook-
scripts
            defaultMode: 0755
        - name: workdir
          emptyDir: {} # <-- 쓰기
가능 작업 디렉토리
      containers:
        - name: presync
          image: {{ .Values.preCheck.image }}
          volumeMounts:
```

```

- name: hook-scripts
  mountPath: /hooks      # 읽기 전용
(ConfigMap)
- name: workdir
  mountPath: /work      # 읽기/쓰기
OK

env:
- name: SCRIPT
  value: {{ required
"preCheck.scriptName is required"
.Values.preCheck.scriptName }}
  command: ["/bin/sh", "-c"]
  args:
  - |
    set -euxo pipefail
    echo "[PreSync] copy scripts to
/work (writable)"
    # 모든 스크립트를 /work로 복사하면서
CRLF 제거

    for f in /hooks/*; do
      [ -f "$f" ] || continue
      base="$(basename "$f")"
      # CRLF -> LF 변환: sed stdout 리
다이렉션으로 새 파일 만들기
      sed 's/\r$//' "$f" >
"/work/$base"
      chmod +x "/work/$base" || true
    done

    echo "[PreSync] running
/work/${SCRIPT}"
    ls -l /work || true
    /work/${SCRIPT}

{{- end }}

```



좋습니다 👍

불여주신 건 ArgoCD **hook** 중에서도 **PreSync** 단계에서 실행되는 Job을 Helm 템플릿으로 만든 예

시예요.

하나씩 아주 기초부터 뜯어볼게요.

## 1. ArgoCD Hook이란?

- ArgoCD는 `Application`을 **sync**할 때 여러 단계(Sync, PreSync, PostSync 등)마다 **추가 리소스**를 실행할 수 있습니다.
- 예: 배포 전에 검사 작업을 하거나(Postgres DB가 준비됐는지, DNS 확인 등) 배포 후 테스트를 돌리는 용도.
- 여기서는 **PreSync Hook** → 애플리케이션 리소스를 배포하기 **직전**에 실행됨.

## 2. YAML 구조 큰 그림

```
apiVersion: batch/v1
kind: Job
metadata:
  name: <릴리즈명>-presync
  annotations:
    argocd.argoproj.io/hook: PreSync
    argocd.argoproj.io/hook-delete-policy: BeforeHookCreation
spec:
  backoffLimit: <재시도 횟수>
  template:
    spec:
      restartPolicy: Never
      volumes: [...]
      containers: [...]
```

- **Job**: 쿠버네티스에서 한번 실행되고 끝나는 작업. 배포 전 확인 같은 일회성 태스크에 딱 맞음.
- **annotations**:
  - `argocd.argoproj.io/hook: PreSync` → 이 Job은 일반 리소스가 아니라 PreSync 책임을 알림.
  - `argocd.argoproj.io/hook-delete-policy: BeforeHookCreation` → 같은 hook을 다시 만들기 직전에만 지움. (성공/실패 결과를 남겨놓고 다음 배포 전까진 보관.)
- **backoffLimit**: 실패했을 때 재시도 횟수 (기본 1번).

### 3. 볼륨과 마운트

```
volumes:
  - name: hook-scripts
    configMap:
      name: <릴리즈명>-hook-scripts
      defaultMode: 0755
  - name: workdir
    emptyDir: {}
```

- ``hook-scripts``: ConfigMap에서 제공하는 스크립트 파일들. (읽기 전용)
- ``workdir``: ``emptyDir`` → Job 실행 동안만 존재하는 임시 디렉토리. 쓰기 가능.

```
volumeMounts:
  - name: hook-scripts
    mountPath: /hooks # 읽기 전용
  - name: workdir
    mountPath: /work # 읽기/쓰기 가능
```

→ 즉, ``/hooks``에는 원본 스크립트, ``/work``는 실행할 복사본.

### 4. 컨테이너 실행 방식

```
containers:
  - name: presync
    image: <preCheck.image>
    env:
      - name: SCRIPT
        value: <실행할 스크립트 이름>
    command: ["/bin/sh", "-c"]
    args:
      - |
        set -euxo pipefail
        echo "[PreSync] copy scripts to /work (writable)"
        for f in /hooks/*; do
          ...
        done
        echo "[PreSync] running /work/${SCRIPT}"
        /work/${SCRIPT}
```

- ``image``: 어떤 컨테이너 이미지를 쓸지 (예: busybox, alpine, etc).
- ``SCRIPT`` 환경변수에 실행할 스크립트 이름을 넣어줌. (``preCheck.scriptName``)
- ``/bin/sh -c``로 셸 스크립트 실행.
- 실제 실행 순서:
  1. ``/hooks``에 있는 모든 스크립트를 ``/work``로 복사하면서 ``CRLF`` → ``LF`` 변환 (윈도우 줄바꿈 제거).
  2. 실행 권한(``chmod +x``) 부여.
  3. ``/work/${SCRIPT}`` 실행.

## 5. Helm 템플릿 부분

```
{{- if .Values.preCheck.enabled }}
...
  backoffLimit: {{ .Values.preCheck.backoffLimit | default 1 }}
  image: {{ .Values.preCheck.image }}
  value: {{ required "preCheck.scriptName is required"
.Values.preCheck.scriptName }}
{{- end }}
```

- ``Values.preCheck.enabled``: values.yaml에서 ``true``일 때만 생성.
- ``backoffLimit``, ``image``, ``scriptName`` → 전부 values.yaml에서 지정.
- ``required`` → scriptName이 반드시 설정되어 있어야 함. 안하면 helm template 단계에서 에러.

## 6. 실제 동작 예시

values.yaml 예:

```
preCheck:
  enabled: true
  image: busybox:1.36
  backoffLimit: 1
  scriptName: precheck.sh
```

ConfigMap (스크립트 저장):

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: myapp-hook-scripts
data:
  precheck.sh: |
    #!/bin/sh
    nslookup kubernetes.default.svc
    echo "PreCheck OK"
```

실행 결과:

1. Job이 만들어지고,
2. ConfigMap에서 `precheck.sh`를 `/work`로 복사,
3. `/work/precheck.sh` 실행 → DNS 확인,
4. 성공 시 Hook Job은 `Completed`로 표시,
5. 이후 애플리케이션 리소스들이 Sync됨.

---

👉 요약:

이 YAML은 Helm Chart에 포함된 **ArgoCD PreSync Hook Job**입니다.

배포 전에 미리 정의한 스크립트를 실행해서, 환경이 준비됐는지 확인하거나 전처리를 수행하는 구조예요.

---

혹시 여기서 더 깊게 보고 싶은 게 **Hook 동작 방식(ArgoCD 입장)** 인지, 아니면 **Helm 템플릿 문법** 쪽일까요?