Longhorn UI 문제 해결

> 쿠버네티스에서 longhorn 웹 ui 가 잘 뜨지 않아. 잘 되다가 안되다가 그래.왜그런거야?

증상이 “가끔 뜨고, 가끔 안 뜨는” 거면 UI 자체 문제라기보다 **경로 / 프록시 / 엔드포인트 상태가 들쭉날쭉**한 경우가 많아요. Longhorn UI는 **`longhorn-frontend`**(SPA 정적파일) ↔ (포드 내부 Nginx 프록시) ↔ **`longhorn-backend/longhorn-manager`** 로 붙는데, 중간 고리가 흔들리면 UI가 랜덤하게 죽습니다. 아래에서 가장 흔한 원인 → 바로 확인할 체크리스트 순서로 정리했어요.

자주 보던 원인 Top 7

1. **Ingress 재작성/타임아웃 문제**
   - `/` → `/#/` 라우팅, WebSocket/장기요청(로그/이벤트 스트림) 타임아웃 미설정.
   - 증상: 첫 로딩은 되는데 클릭하면 빈 화면/회색 화면, 새로고침하면 간헐적으로 열림.
2. **`longhorn-frontend` 단일 레플리카 + 노드 드레인/재시작**
   - 1개 포드만 떠 있고 그 노드가 바쁘거나 옮겨지면 잠깐씩 502/504.
3. **Service Endpoints가 비었다가 생김**
   - **`longhorn-backend/longhorn-manager`** 재시작 중일 때 엔드포인트 0개 → UI 빈 화면.
4. **리소스 부족(특히 manager)**
   - CPU 압박으로 **`longhorn-manager`**의 Readiness가 흔들리면 API 응답 끊김.
5. **클러스터/노드 DNS 지연**
   - Ingress → Service → Pod 해석이 느려져 간헐적 실패.
6. **네트워크 정책/보안장비가 WebSocket 끊음**
   - L7 장비/프록시가 업그레이드 뒤 기본 타임아웃이 짧아짐.
7. **버전 미스매치(프론트는 새 버전, 백엔드는 예전 버전)**
   - Helm/ArgoCD 롤링 중 절반만 바뀌었을 때 발생.

바로 보는 10분 점검(명령 모음)

> 네임스페이스는 보통 **`longhorn-system`**입니다.

A. 포드/이벤트 상태

```
bash


kubectl get po -n longhorn-system -o wide
kubectl describe deploy longhorn-frontend -n longhorn-system | sed -n '1,200p'
```

```
kubectl get events -n longhorn-system --sort-by=.lastTimestamp | tail -n 30
```

B. 서비스와 엔드포인트가 비는지 확인

```
bash

kubectl get svc,endpoints -n longhorn-system | egrep 'longhorn-(front|back)end'
kubectl describe ep longhorn-frontend -n longhorn-system
kubectl describe ep longhorn-backend  -n longhorn-system
```

C. Ingress 라우팅/타임아웃(있다면)

```
bash

kubectl get ing -n longhorn-system -o yaml
```

확인 포인트:

- `/` 경로가 **`longhorn-frontend`** 서비스로 가는지
- 주석(annotations)에 **WebSocket 허용**과 **대기시간 상향** 있는지
  - 예: **`nginx.ingress.kubernetes.io/proxy-read-timeout: "3600"`**
  - 예: **`nginx.ingress.kubernetes.io/websocket-services: longhorn-frontend`**

D. 클러스터 내부에서 직접 프론트엔드/백엔드 헬스 확인

```
bash

# 임시 테스트 파드
kubectl run -n longhorn-system curlbox --image=curlimages/curl:8.10.1 -it --rm -- \
  sh -lc 'apk add --no-cache jq || true; \
          echo "Frontend:";  curl -sSf http://longhorn-frontend.longhorn-
system.svc.cluster.local/ | head -n1; \
          echo "Backend:";   curl -sS  http://longhorn-backend.longhorn-
system.svc.cluster.local/v1 | jq .status 2>/dev/null || true'
```

- Frontend가 200이고 Backend **`/v1`**가 응답하면 UI 경로·프록시는 정상인 편.

E. 매니저 리더/레디니스

```
bash

kubectl get po -n longhorn-system -l longhorn.io/component=manager -o wide
kubectl logs -n longhorn-system -l longhorn.io/component=manager --tail=200 | egrep -i
'error|timeout|leader|health'
```

F. 빠르게 포트포워드로 외부 경유 없이 확인(Ingress 문제 분리)

```
bash

kubectl port-forward -n longhorn-system svc/longhorn-frontend 8080:80
# 브라우저에서 http://localhost:8080
```

- 이 경로로는 항상 잘 뜬다면, Ingress/LB 쪽 설정이 범인인 가능성 ↑

바로 적용 가능한 안정화 팁(안전한 조치부터)

1. **프론트엔드 레플리카 2로 스케일 + 간단한 Anti-Affinity**

```bash
kubectl scale deploy/longhorn-frontend -n longhorn-system --replicas=2
kubectl patch deploy longhorn-frontend -n longhorn-system --type='json' -p='[
  {"op":"add","path":"/spec/template/spec/affinity","value":{
    "podAntiAffinity":{"preferredDuringSchedulingIgnoredDuringExecution":[
      {"weight":100,"podAffinityTerm":{
        "topologyKey":"kubernetes.io/hostname",
        "labelSelector":{"matchLabels":{"app":"longhorn-ui","longhorn.io/component":"frontend"}}
      }}
    ]}
  }}
]'
```

- UI 자체는 상태가 없어서 2개로 돌려도 부담 적습니다.

2. **Ingress에 WebSocket/타임아웃/리라이트 보강**
   (Ingress-NGINX 예시)

```yaml
metadata:
  annotations:
    nginx.ingress.kubernetes.io/proxy-read-timeout: "3600"
    nginx.ingress.kubernetes.io/proxy-send-timeout: "3600"
    nginx.ingress.kubernetes.io/proxy-buffering: "off"
    nginx.ingress.kubernetes.io/websocket-services: "longhorn-frontend"
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
  - host: longhorn.example.com
    http:
      paths:
      - path: /(.*)
        pathType: Prefix
        backend:
          service:
            name: longhorn-frontend
            port:
              number: 80
```

- 컨트롤러 종류별로 키가 다를 수 있어요(HAProxy/Traefik이면 해당 키로).

3. **리소스 요청치 최소 보장**

   - **`longhorn-manager`**, **`longhorn-backend`**(존재 시), **`longhorn-frontend`**에 CPU/메모리 requests를 작게라도 지정.
   - 노드가 바쁠 때도 레디니스가 덜 흔들립니다.

4. **Service 타입 단순화해서 문제 분리**

   - Ingress 전에 **`ClusterIP`** + 포트포워드로 정상 확인 → Ingress만 재검.

5. **버전 일치**

   - Helm/ArgoCD로 업그레이드 중이었다면 **모든 컴포넌트가 같은 릴리스**인지 확인.
   - 앱 삭제-재설치 대신 **Sync 정책(Prune 포함)으로 롤아웃**이 일반적으로 안전합니다.

빠른 원클릭 진단 스크립트(운영용 초안)

저장: **`lh-ui-diagnose.sh`**, 실행: **`bash lh-ui-diagnose.sh`**

```
bash


#!/usr/bin/env bash
set -euo pipefail

NS="${NS:-longhorn-system}"

echo "== Pods (wide) =="
kubectl get po -n "$NS" -o wide

echo -e "\n== Services & Endpoints (frontend/backend) =="
kubectl get svc,endpoints -n "$NS" | egrep 'longhorn-(front|back)end' || true
echo -e "\n-- EP(frontend) --"
kubectl describe ep longhorn-frontend  -n "$NS" || true
echo -e "\n-- EP(backend) --"
kubectl describe ep longhorn-backend   -n "$NS" || true

echo -e "\n== Manager pods =="
kubectl get po -n "$NS" -l longhorn.io/component=manager -o wide || true

echo -e "\n== Recent events =="
kubectl get events -n "$NS" --sort-by=.lastTimestamp | tail -n 40 || true

echo -e "\n== Frontend Deploy describe (first 150 lines) =="
kubectl describe deploy longhorn-frontend -n "$NS" | sed -n '1,150p' || true

echo -e "\n== Ingress (if any) =="
kubectl get ing -n "$NS" -o yaml || true

echo -e "\n== In-cluster curl test =="
kubectl run -n "$NS" curlbox --image=curlimages/curl:8.10.1 -it --rm -- \
  sh -lc 'apk add --no-cache jq >/dev/null 2>&1 || true; \
          echo "[frontend]"; curl -sS http://longhorn-frontend.'"$NS"'.svc.cluster.local/ | head
-n1 || true; \
          echo "[backend /v1]"; curl -sS http://longhorn-backend.'"$NS"'.svc.cluster.local/v1 |
jq .status 2>/dev/null || true'
```

- 출력에서 **EP가 0개**로 나오는 순간이 보이면 그 타이밍이 끊김 원인에 가깝습니다.
- Ingress가 있는데 포트포워드는 항상 정상 → Ingress 설정/외부 LB 의심.
- 반대로 내부도 불안정 → manager/backend 리소스/재시작/리더 변경 쪽을 보세요.

마무리 힌트

- ArgoCD로 관리 중이면, Ingress/Deployment의 **requests/annotations**가 값파일에서 덮이는지 확인하세요. Sync 때마다 좋아졌다 나빠지는 패턴이면 Git 값이 원인일 확률이 높습니다.
- 노드 교체/라벨 이동을 자주 하셨다면, **`longhorn-manager`**가 그 타이밍에 재조정하며 잠깐 API가 끊길 수 있어요. 그 구간을 이벤트 타임라인과 맞춰보면 깔끔하게 보입니다.

필요하면 당신 환경(Helm values/Ingress 컨트롤러 종류) 기준으로 **안정화용 values.yaml 패치**까지 바로 만들어드릴게요.

> 브라우저에서 주의요함이라고 느낌표가 떠거든. 수십번 새로고침하면 화면이 렌더링 될때도 있고 그래

말씀해주신 **브라우저의 "주의 요함(느낌표)" 아이콘**은 대체로 TLS/HTTPS 경고일 때 많이 뜹니다. Longhorn UI 자체가 SPA(단일 페이지 앱)이라 HTML은 잘 로드돼도, 백엔드 API 호출이 HTTPS → HTTP(혼합 콘텐츠)로 가거나, 인증서가 유효하지 않으면 저런 경고가 뜨고 화면도 들쭉날쭉합니다.

보통 이런 케이스가 많습니다:

1. **Self-signed 인증서 사용**
   - **`kubectl port-forward`**로 열거나, Ingress에서 자체 서명 인증서를 쓸 때 브라우저가 "신뢰할 수 없음" 띄움.
   - 새로고침하다가 캐시가 섞이면 UI가 한 번씩 뜨기도 하고, 백엔드 호출은 막히기도 함.
2. **Mixed Content (혼합 콘텐츠)**
   - 브라우저에서 **`https://`** 주소로 접속했는데, UI 자바스크립트가 내부적으로 **`http://`** API를 호출.
   - 크롬/엣지에서는 이런 요청을 기본 차단 → UI는 비어 있거나 랜덤하게 동작.
3. **Ingress/프록시 설정 누락**
   - **`nginx.ingress.kubernetes.io/force-ssl-redirect: "true"`** 같은 HTTPS 강제 리다이렉트 미설정.
   - **`longhorn-backend`**로 프록시 패스 할 때 **`/v1`** 경로를 제대로 못 넘김.
4. **인증서 체인 불완전**
   - 사내 CA 발급 인증서인데 중간 CA 체인을 Ingress에 안 넣은 경우.
   - 일부 브라우저/OS는 그냥 막아버리고, 일부는 허용 → "되다 안 되다" 현상.

확인해보실 포인트

```
bash

kubectl get ing -n longhorn-system -o yaml
```

- **`tls:`** 섹션이 있는지
- **`secretName:`**이 신뢰할 수 있는 인증서인지

```
bash

kubectl describe svc longhorn-frontend -n longhorn-system
kubectl describe svc longhorn-backend  -n longhorn-system
```

- Ingress가 올바른 Service 포트(80 → 80)로 연결되는지

빠른 실험

1. **포트포워드로 접속**

```
bash

kubectl port-forward -n longhorn-system svc/longhorn-frontend 8080:80
```

브라우저에서 **`http://localhost:8080`** 접속했을 때는 경고가 안 뜨고 항상 잘 열리면 → Ingress/TLS 설정 문제 확정.

2. **브라우저 개발자 도구 → Console/Network 탭**
   - **`Mixed Content: The page at 'https://...' was loaded over HTTPS, but requested an insecure resource 'http://...'`** 이런 메시지가 나오면 → 혼합 콘텐츠 문제.

해결 방향

- 테스트 용도: 그냥 HTTP로 열어보세요 (port-forward).
- 운영/공유 환경:
  - Ingress에 **정상 TLS 인증서(사내 CA or Let's Encrypt)** 적용.
  - **`nginx.ingress.kubernetes.io/backend-protocol: "HTTP"`** / **`proxy-read-timeout`** / **`websocket-services`** 같은 어노테이션도 확인.
  - Helm values에서 **`service.ui.type: ClusterIP`** + Ingress를 안전하게 붙이는 게 권장 경로.

👉 혹시 지금 Longhorn UI를 Ingress로 https 붙여서 접속 중이세요, 아니면 **`kubectl port-forward`**나 NodePort로 바로 열고 계세요?

> longhorn cli도 있나?

네, Longhorn도 CLI가 있습니다. 다만 독립적인 "longhorn" 바이너리라기보다는, 두 가지 경로가 있어요:

---

1. `longhornctl` (커뮤니티 도구)

- Rancher 쪽에서 만든 **Longhorn CLI(베타 수준)** 이 있었는데, 지금은 거의 사용되지 않고 문서에서도 잘 다루지 않습니다.
- 대신 공식적으로는 **`kubectl`** 기반 접근을 권장합니다.

---

2. `kubectl longhorn` 플러그인

- Longhorn 프로젝트가 제공하는 **kubectl 플러그인**이 있어요.
- 설치하면 **`kubectl longhorn …`** 형태로 명령어를 쓸 수 있습니다.

```
bash

# 설치 예시
kubectl krew install longhorn
```

```
# 또는 직접 바이너리 받아서 PATH에 추가
```

- 이걸로 볼륨, 노드, 엔진, 백업 등 Longhorn 리소스를 확인하고 조작할 수 있습니다.
- 사용 예:

```
bash

kubectl longhorn volume list
kubectl longhorn volume inspect pvc-xxxx
kubectl longhorn node list
kubectl longhorn backup list
```

---

3. 그냥 `kubectl`로 CRD 다루기

- Longhorn은 사실상 Kubernetes CRD(Custom Resource Definition) 집합이라, 별도 CLI 없이도 다 제어할 수 있습니다.
- 주요 CRD:
  - **`volumes.longhorn.io`**
  - **`engines.longhorn.io`**
  - **`replicas.longhorn.io`**
  - **`nodes.longhorn.io`**
  - **`backups.longhorn.io`**
- 예시:

```
bash

# 볼륨 목록
kubectl get volumes.longhorn.io -n longhorn-system
# 특정 볼륨 상세
kubectl describe volume.longhorn.io pvc-12345 -n longhorn-system
```

👉 정리하면:

- **공식적으로는 `kubectl longhorn` 플러그인**을 추천.
- 없더라도 CRD를 직접 조회/수정 가능.
- 옛날 독립형 **`longhornctl`**은 사실상 쓰이지 않음.

원하시면 제가 **`kubectl longhorn`** 플러그인 설치법이랑 자주 쓰는 명령어 치트시트 만들어드릴까요?