診斷 Kubernetes 環境下 vmalert 寫入 vminsert 時的 503 連線終止錯誤

第1章：引言：問題背景與技術棧總覽

在現代雲原生架構中，監控系統與服務網格的結合是常見的部署模式。當服務網格（例如 Istio）介入微服務之間的通訊時，原有的應用程式間通訊問題可能會轉化為更複雜的服務網格層面問題。使用者在 VictoriaMetrics 監控堆疊中，遇到 vmalert 將警報寫入 vminsert 時出現的 503 錯誤，並附帶了具體的錯誤訊息 'upstream connect error or disconnect/reset before headers. reset reason: connection termination'。本報告旨在對此問題進行深度剖析，並提供一個由淺入深、系統化的診斷與解決方案。

1.1 錯誤訊息：**503 upstream connect error...** 深度剖析

HTTP 503 Service Unavailable 是一個通用狀態碼，表示伺服器暫時無法處理請求。然而，在 Istio/Envoy 服務網格的環境中，這個錯誤通常會附帶一個詳細的重設原因（reset reason），這為診斷提供了關鍵線索。使用者所遇到的錯誤訊息 'upstream connect error or disconnect/reset before headers. reset reason: connection termination' 是一個精確的技術描述，它將問題的範圍從所有可能的 503 錯誤中精準鎖定到一個特定的故障模式 [1]。

這個錯誤訊息可以被拆解為幾個核心部分：

- **upstream connect error**: 這表示客戶端 Envoy Proxy 在嘗試與上游服務（即 vminsert Pod 的 Envoy Sidecar）建立連線時遇到了問題 [2]。
- **disconnect/reset before headers**: 這是此問題最關鍵的特徵。它表明 TCP 連線已經建立成功，但在 HTTP 標頭（Headers）完全傳輸或接收之前，連線就被意外地關閉或重設了。這與 503 NR（No Route）或 503 UF（Upstream Failure）等錯誤有本質區別，因為後者通常表示根本沒有可用的路由或上游服務完全無法回應，而不是在連線的生命週期中途被終止 [4]。

- **reset reason: connection termination**: 這進一步證實了連線被突然終止的事實。這通常是由於上游服務或其代理主動發送了一個 TCP RST（Reset）封包，而非客戶端因逾時而放棄連線。

綜合來看，這個錯誤模式排除了簡單的 Istio 路由配置錯誤，而將診斷重心引導至連線建立後的生命週期管理問題。這強烈暗示了 HTTP Keep-Alive 逾時不匹配的競態條件，或是上游服務 vminsert 在處理請求時發生了瞬時性的資源瓶頸或應用程式崩潰，導致其突然關閉了連線。

1.2 VictoriaMetrics 在 Kubernetes 環境下的寫入流程：vmalert, vminsert, Istio/Envoy

要全面診斷此問題，必須先理解 vmalert 到 vminsert 的資料寫入流程，以及 Istio 在其中扮演的角色。

在 VictoriaMetrics 的分散式集群架構中，vminsert 扮演著資料攝取前端的角色。它負責接收來自各種來源（包括 vmalert）的監控資料，並透過一致性雜湊（consistent hashing）將這些資料分發到多個 vmstorage 節點進行持久化儲存 [7]。

vmalert 則是一個客戶端元件，它會定期執行配置的警報或記錄規則，並透過 -remoteWrite.url 命令行參數指定的遠端寫入協議將結果發送到 vminsert [9]。

在 Kubernetes 且啟用 Istio 服務網格的環境下，這個通訊路徑變得更為複雜。vmalert Pod 和 vminsert Pod 都會被注入一個 Envoy Sidecar 代理。因此，vmalert 應用程式發出的遠端寫入請求實際上並非直接到達 vminsert 應用程式，而是遵循以下路徑：

1. vmalert 應用程式 (客戶端)
2. vmalert Pod 的 Envoy Sidecar (客戶端代理)
3. vminsert Pod 的 Envoy Sidecar (伺服器端代理)
4. vminsert 應用程式 (伺服器端)

這種代理模式意味著連線問題可能發生在上述四個環節中的任何一個。然而，由於錯誤訊息明確指出 'upstream connect error' 和 'connection termination'，最有可能的故障點發生在 vmalert 的 Envoy Sidecar 試圖與 vminsert 的 Envoy Sidecar 建立和維護連線的過程中。因此，Istio/Envoy 的配置、vminsert Pod 的健康狀態，以及底層 Kubernetes 網路行為都將成為診斷的重點。

第2章：服務網格層面的診斷：聚焦 Istio 與 Envoy

Istio 作為服務網格的核心，通過其數據平面代理 Envoy，接管了所有 Pod 間的網路通訊。因此，當出現連線終止錯誤時，Istio 和 Envoy 的配置是首要的排查對象。

2.1 HTTP Keep-Alive 逾時設定不匹配：核心競態條件分析

基於錯誤訊息 'reset reason: connection termination'，最常見且最具欺騙性的根本原因之一是 HTTP Keep-Alive 逾時（idle timeout）在客戶端（vmalert 的 Envoy Sidecar）和伺服器端（vminsert 的 Envoy Sidecar 或應用程式本身）之間存在不匹配。

HTTP Keep-Alive 旨在通過重用同一個 TCP 連線來發送多個 HTTP 請求，從而提高性能。在連線閒置一段時間後，伺服器或客戶端可以主動關閉它。問題的根源在於一種微妙的競態條件：如果伺服器的閒置連線逾時設定比客戶端的短，伺服器就可能在連線閒置期間先行關閉連線。此時，客戶端（vmalert 的 Envoy）並不知道連線已失效，並可能在伺服器關閉連線的瞬間，嘗試通過該連線發送下一個請求 [11]。

這種情況會導致客戶端收到一個 TCP RST 封包，進而觸發 Envoy 記錄下 'upstream connect error... reset reason: connection termination' 的錯誤。這種問題的診斷難度在於，它往往是間歇性的，只在連線長時間閒置後才發生。

不同的應用程式框架或伺服器，其預設的 Keep-Alive 逾時設定各不相同，例如 Nginx 預設為 75 秒，Tomcat 預設為 60 秒 [11]。因此，一個通用的建議是：在任何使用 HTTP Keep-Alive 的場景中，應始終確保伺服器的 Keep-Alive 逾時設定

嚴格大於客戶端代理的逾時，以避免此類競態條件 [12]。

2.2 Istio 配置檢查：VirtualService, DestinationRule, Sidecar

儘管 connection termination 錯誤不太可能是由 Istio 路由配置的簡單疏忽所致，但檢查相關的 Istio 資源仍是必要的。Istio 的配置是分層級的，高層次的 VirtualService 雖不直接管理連線逾時，但其底層配置可能影響流量行為 [5]。

- **VirtualService 與 DestinationRule**：應檢查 vmalert 服務與 vminsert 服務之間的通

訊路徑是否由 VirtualService 和 DestinationRule 資源正確定義。雖然這些資源通常與 503 NR（No Route）錯誤更相關，但它們的配置錯誤仍可能影響流量的穩定性 [4]。

- **Sidecar** 資源：解決 Keep-Alive 逾時不匹配問題的關鍵在於 Istio 的 Sidecar 資源。Envoy 的連線行為可以通過 Sidecar 資源的 outboundTrafficPolicy 或 inboundTrafficPolicy 中的 httpProtocolOptions 進行調整。Sidecar 資源允許操作者為特定的服務或工作負載，定義 Envoy 的 idleTimeout 設定。例如，可以為 vminsert 服務的 Sidecar 擴展上游連線的 idleTimeout，確保其大於 vmalert 服務端 Envoy 的逾時，以解決上述的競態條件問題 [11]。
- **Istio Pilot** 同步延遲：當 vminsert Pod 由於資源限制或應用程式問題被意外逐出或重啟時，Istio 的控制平面 Pilot 需要時間來更新服務端點列表。雖然 Pod 成功重啟，但如果 Istio 的狀態同步出現延遲，仍可能導致客戶端 Envoy 嘗試連線到一個已經失效的端點，從而引發連線錯誤。在某些情境下，重新建立 VirtualService 可能是一種手動刷新 Istio 狀態的權宜之計 [14]。

2.3 TLS/mTLS 憑證錯誤排查：Envoy 憑證狀態檢查

在一個啟用了服務網格的生產環境中，mTLS (mutual TLS) 通常是預設的通訊加密機制。在這種情況下，TLS 憑證的有效性對於服務之間的連線至關重要。如果憑證過期或配置不正確，TLS 握手將會失敗，這會直接導致連線被重設。

- 憑證過期：Envoy Sidecar 使用 Istio 簽發的短期葉憑證（leaf certificates）進行 mTLS 通訊。如果由於 Istiod 控制平面或其與 Envoy 之間的通訊問題，導致 Envoy 無法及時輪換憑證，憑證過期將會發生。這將導致任何新的連線嘗試在 TLS 握手階段失敗，產生與使用者所見錯誤訊息一致的連線重設錯誤 [15]。
- 診斷與解決：可以透過 curl http://localhost:19000/certs 命令，在 Pod 內直接檢查 Envoy Sidecar 的憑證狀態。如果 days_until_expiration 顯示為 0，則表示憑證已過期。此時，重啟 Pod 的 Sidecar 容器通常會觸發憑證的重新獲取和更新 [15]。

第3章：應用程式與 Kubernetes 資源層面的診斷

服務網格層面的連線終止問題，其根源往往是底層應用程式或 Kubernetes 資源配置的不當。因此，對 vminsert Pod 的內部狀態進行深入檢查是必不可少的。

3.1 Pod 健康與狀態檢查：readinessProbe 與 CrashLoopBackOff

即使 Pod 顯示為 Running 狀態，也並不保證它能夠正常接收流量。Kubernetes 的服務發現機制依賴於 readinessProbe。

- **readinessProbe 失敗**：如果 vminsert Pod 的 readinessProbe 失敗，即使 Pod 本身沒有終止，kubelet 也會將其從 Service 的端點（Endpoint）列表中移除。這將導致發往 vminsert 服務的流量無法到達任何健康的 Pod，進而觸發客戶端接收到 503 錯誤 [16]。
  應仔細檢查
  vminsert 的 readinessProbe 配置，並透過 kubectl describe pod 觀察其狀態和相關事件。
- **CrashLoopBackOff**：如果 vminsert Pod 處於 CrashLoopBackOff 狀態，這表示應用程式正在持續崩潰並重啟。在每次崩潰重啟的過程中，所有現有連線都會被終止，並且在 Pod 重新恢復服務前，它也無法處理新的連線。這種行為會直接導致客戶端接收到連線終止的錯誤 [16]。

3.2 資源瓶頸分析：CPU 與記憶體限制、OOMKill

資源不足是導致應用程式行為異常或崩潰的常見原因，而這在處理遠端寫入資料量時尤其突出。vmalert 的遠端寫入請求可能會在短時間內產生流量尖峰，這對 vminsert 的資源處理能力構成了嚴峻挑戰。

- 資源耗盡：當 vminsert Pod 的 CPU 或記憶體資源使用率在負載高峰時達到上限，應用程式可能會變得無回應，甚至崩潰 [1]。
- **OOMKill**：如果 vminsert Pod 的記憶體使用量超過了其 limits 配置，Kubernetes 的 kubelet 進程會強制終止該 Pod，標記為 OOMKilled。Pod 的突然終止將立即中斷所有連線，這完美解釋了 connection termination 的錯誤。因此，檢查 vminsert Pod 的資源使用趨勢（kubectl top）以及其 kubectl describe pod 的事件列表，是診斷此類問題的關鍵 [18]。

此外，即使是滾動升級這樣看似無害的操作，也可能因為負載轉移到剩餘的健康節點上，導致這些節點的資源負載激增，從而引發集群範圍的不穩定性 [19]。

3.3 服務與端點連通性驗證：Service Selector, Endpoint, NetworkPolicy

在服務網格之上，底層的 Kubernetes 網路仍然是所有通訊的基石。檢查其基本機制是否正常運作是診斷的最後一道防線。

- **Service Selector 與 Endpoint**：確保 vminsert Service 的 selector 標籤與 vminsert Pod 的 label 精確匹配。如果兩者不匹配，Service 將無法發現任何 Pod，導致其 Endpoint 列表為空，客戶端將無法連線 [16]。可以使用 kubectl describe service 和 kubectl get pods -l "[label]" 命令來驗證。
- **NetworkPolicy**：在啟用了 NetworkPolicy 的集群中，一個過於嚴格的策略可能會無意中阻擋 vmalert Pod 與 vminsert Pod 之間的流量，導致連線失敗 [2]。
- **kube-proxy 與 conntrack**：在某些極端情況下，尤其是在處理大流量時，底層 Linux 內核的 conntrack（connection tracking）機制可能出現問題。kube-proxy 依賴此機制進行服務代理。conntrack 表滿載或處理錯誤可能導致間歇性的連線重設，從而引發 connection termination 錯誤。這種情況通常需要更深層次的網路診斷和內核參數調整 [21]。

第4章：根本原因綜合診斷與結構化解決方案

本章將上述所有診斷路徑整合成一個結構化的故障排除流程，並針對最常見的根本原因提供具體的解決方案。

4.1 結構化故障排除流程

1. 檢查 **vminsert Pod** 狀態：首先使用 kubectl get pods -n <namespace> 和 kubectl describe pod <vminsert-pod-name> -n <namespace> 檢查 vminsert Pod 是否處於 Running 且沒有 CrashLoopBackOff 或 OOMKilled 等異常狀態。
2. 檢查 **vminsert Pod** 資源使用：使用 kubectl top pod <vminsert-pod-name> -n <namespace> 監控 Pod 的 CPU 和記憶體使用率。觀察是否有資源使用率飆升導致 Pod 被殺死的跡象。
3. 檢查 **vmalert** 和 **vminsert** 的日誌：使用 kubectl logs <pod-name> -n <namespace> 檢查兩個 Pod 的日誌。在 vminsert 的日誌中尋找任何錯誤、崩潰或超載的跡象。

4. 驗證 **Pod** 間連通性：在 vmalert Pod 內部使用 kubectl exec 執行 curl 命令，以測試到 vminsert 服務的連通性：kubectl exec -it <vmalert-pod-name> -n <namespace> -- curl -I <vminsert-service-name>:<port>。
5. 檢查 **Istio** 配置：使用 kubectl get virtualservice 和 kubectl get destinationrule 檢查 Istio 的路由配置。如果懷疑是 Keep-Alive 逾時問題，檢查是否有針對 vminsert 的 Sidecar 資源配置了逾時策略。
6. 檢查 **TLS** 憑證狀態：如果 Istio 啟用了 mTLS，在 vminsert Pod 內檢查 Envoy 的憑證狀態：kubectl exec -it <vminsert-pod-name> -n <namespace> -- curl http://localhost:19000/certs。

4.2 針對常見根本原因的解決方案

解決方案一：調整 **Keep-Alive** 逾時設定

這是最有可能的根本原因。解決方案是為 vminsert 服務的 Envoy Sidecar 配置一個較長的上游閒置逾時，以避免與 vmalert 客戶端 Envoy 的逾時發生競態條件。

YAML

```yaml
# 範例 Istio Sidecar 配置
apiVersion: networking.istio.io/v1beta1
kind: Sidecar
metadata:
  name: vminsert-sidecar
  namespace: <vminsert-namespace>
spec:
  workloadSelector:
    labels:
      app: vminsert
  outboundTrafficPolicy:
    mode: REGISTRY_ONLY
  egress:
    - hosts:
        - "istio-system/*" # Allow egress to Istio control plane
    - hosts:
```

```
      - "./*" # Allow egress to services in the same namespace
  inbound:
    port:
      number: 8480 # Port of vminsert
      protocol: HTTP
    defaultEndpoint: 127.0.0.1:8480
    httpProtocolOptions:
      idleTimeout: 180s # 調整此值，確保大於 vmalert 端 Envoy 的逾時
```

此處的 httpProtocolOptions.idleTimeout 應設定為一個足夠長的時間，例如 3 分鐘（180s），以確保 vminsert 端在處理完請求後不會過早關閉連線 [11]。

解決方案二：修正 **Istio** 配置或憑證問題

如果問題源於 Istio 的配置或憑證過期，則需要修正相關的 Istio 資源或重啟 Pod。

- 修正 **VirtualService**：確保 VirtualService 的 hosts 欄位與 Service 名稱匹配，且 route 指向正確的 host [4]。
- 重啟 **Pod**：如果 istioctl proxy-config 顯示配置異常或憑證過期，重啟 vminsert Pod 通常能強制 Envoy Sidecar 重新從 Istiod 獲取最新配置和憑證。

解決方案三：優化 **Kubernetes** 資源與配置

如果問題是由資源瓶頸引起的，則需要調整 Pod 的資源配置。

- 調整資源限制：增加 vminsert Deployment 的 requests 和 limits，為其提供足夠的 CPU 和記憶體來處理高峰負載 [18]。

YAML

```
# 範例 Deployment 資源配置
spec:
  template:
    spec:
      containers:
      - name: vminsert
        resources:
          requests:
            memory: "128Mi"
```

```
        cpu: "250m"
      limits:
        memory: "512Mi"
        cpu: "1000m"
```

- 調整 **readinessProbe**：確保 readinessProbe 的配置能夠準確反映 vminsert 應用程式的健康狀態，且其 periodSeconds 和 failureThreshold 設定合理，以避免 Pod 在短暫過載期間被錯誤地移除端點列表.[16]

表1：503 錯誤與潛在原因對應表

錯誤代碼/訊息	可能原因	診斷命令	相關來源
503 UC (connection termination)	HTTP Keep-Alive 逾時不匹配	istioctl proxy-config listeners <pod-name> -o json 檢查 Envoy 逾時設定	11
503 UC (connection termination)	vminsert Pod 資源耗盡 (OOMKill)	kubectl top pod <vminsert-pod-name> kubectl describe pod <vminsert-pod-name>	1
503 UC (connection termination)	mTLS 憑證過期或不匹配	kubectl exec <pod-name> -- curl http://localhost:15000/certs	15
503 NR (No Route Configured)	VirtualService 或 DestinationRule 配置錯誤	kubectl get virtualservice kubectl get destinationrule	4
503 Service Unavailable	readinessProbe 失敗導致 Pod 不在端點列表中	kubectl describe pod <pod-name> kubectl get endpoints <service-name>	16
503 UF (Upstream	上游服務崩潰或無法回	kubectl logs	6

Failure)	應	<pod-name>	

第5章：長期預防與最佳實踐建議

診斷並解決當前問題是重要的，但更關鍵的是建立一套健全的機制來預防此類問題再次發生。

5.1 在服務網格中設定合理的逾時策略

確保服務網格中的所有服務都遵循一致的連線逾時管理策略是預防競態條件的根本方法。

- 統一標準：建立一個組織層面的標準，規定所有應用程式的 HTTP Keep-Alive 逾時都必須大於 Envoy Sidecar 的上游閒置逾時，例如統一將伺服器端應用程式的逾時設定為 300 秒，而 Envoy 的 idleTimeout 設定為 240 秒 [12]。
- 利用 **Istio** 資源：透過在 Istio 的 Sidecar 或 DestinationRule 資源中定義統一的逾時策略，並將其應用於整個服務網格，可以實現全局的一致性管理 [11]。

5.2 增強 **VictoriaMetrics** 元件的資源彈性與監控

資源瓶頸是服務不穩定性的主要來源。對 vmalert 和 vminsert 進行主動式資源管理至關重要。

- 配置合理的資源：為 vminsert 和 vmalert Pod 設定基於實際負載的 requests 和 limits，以確保它們有足夠的資源來處理資料尖峰，並防止被 kubelet 意外終止 [18]。
- 考慮自動擴展：針對 vminsert 服務，可以考慮實施 Horizontal Pod Autoscaler（HPA），根據 CPU 或自定義指標，自動擴展 Pod 副本數量，以應對 vmalert 遠端寫入資料量的不規則尖峰 [19]。
- 了解集群行為：在進行滾動升級等維護操作時，應意識到剩餘節點的負載會增加。適當增加集群中 vmstorage 和 vminsert 節點的數量，可以顯著降低升級期間的單節點負載，從而提高集群的穩定性 [19]。

5.3 建立全面的可觀察性，集成 Istio 與 VictoriaMetrics 指標

單獨監控應用程式或服務網格是不夠的，必須將兩者的指標整合起來，才能獲得一個完整的系統視圖。

- 指標整合：確保 Prometheus Operator (ServiceMonitor) 或其他機制能夠從 vmalert 和 vminsert 收集原生指標，同時也能從 Istio Sidecar 收集流量、錯誤和延遲等相關指標 [23]。
- 建立預警：針對關鍵指標設定預警，例如 vminsert 的遠端寫入錯誤率、Pod 的記憶體或 CPU 使用率、以及 readinessProbe 失敗等，以便在問題演變成服務中斷前主動介入 [6]。
- 利用診斷工具：將本報告中提及的診斷命令內化為日常運維 SOP 的一部分，以便在問題發生時能夠快速、準確地定位根本原因。

附錄：常用診斷命令速查表

命令	描述
kubectl get pods -n <namespace>	檢查指定命名空間下所有 Pod 的狀態。
kubectl logs <pod-name> -n <namespace>	獲取指定 Pod 的應用程式日誌。
kubectl top pod <pod-name> -n <namespace>	監控指定 Pod 的 CPU 和記憶體使用情況。
kubectl describe pod <pod-name> -n <namespace>	獲取 Pod 的詳細資訊和事件，有助於診斷 OOMKilled、readinessProbe 失敗等問題。
kubectl get virtualservice -n <namespace>	列出指定命名空間下所有 VirtualService 資源。
kubectl get destinationrule -n <namespace>	列出指定命名空間下所有 DestinationRule 資源。
kubectl get service <service-name> -n <namespace>	檢查服務的基本配置和狀態。

kubectl get endpoints <service-name> -n <namespace>	檢查服務後端是否有健康的端點。
kubectl exec -it <pod-name> -n <namespace> -- curl <service-name>:<port>	在 Pod 內部執行 curl 測試與其他服務的連通性。
istioctl proxy-config listeners <pod-name> -o json	檢視指定 Pod 內部 Envoy Proxy 的監聽器配置，包括逾時設定。
kubectl exec -it <pod-name> -n <namespace> -- curl http://localhost:19000/certs	檢查 Envoy Sidecar 的 mTLS 憑證狀態。

引用的著作

1. Istio 503 UC (Upstream Connection Termination) - Doctor Droid, 檢索日期：8月 12, 2025, https://drdroid.io/stack-diagnosis/istio-503-uc--upstream-connection-termination
2. How to Fix "Upstream Connect Error" in Spring Boot with Java 11 | SigNoz, 檢索日期：8月 12, 2025, https://signoz.io/guides/upstream-connect-error-or-disconnect-reset-before-headers-reset-reason-connection-failure-spring-boot-and-java-11/
3. upstream connect error or disconnect/reset before headers. retried and the latest reset reason: protocol error - Microsoft Learn, 檢索日期：8月 12, 2025, https://learn.microsoft.com/en-us/answers/questions/1195197/upstream-connect-error-or-disconnect-reset-before
4. Istio 503 NR (No Route) - Doctor Droid, 檢索日期：8月 12, 2025, https://drdroid.io/stack-diagnosis/istio-503-nr--no-route-8a684
5. Istio 503 NR (No Route Configured) - Doctor Droid, 檢索日期：8月 12, 2025, https://drdroid.io/stack-diagnosis/istio-503-nr--no-route-configured-a6123
6. Istio 503 UF (Upstream Failure) - Doctor Droid, 檢索日期：8月 12, 2025, https://drdroid.io/stack-diagnosis/istio-503-uf--upstream-failure
7. Cluster version - VictoriaMetrics, 檢索日期：8月 12, 2025, https://docs.victoriametrics.com/victoriametrics/cluster-victoriametrics/
8. Getting Started - VictoriaMetrics Components, 檢索日期：8月 12, 2025, https://victoriametrics.com/blog/victoriametrics-getting-started/
9. vmalert - VictoriaMetrics, 檢索日期：8月 12, 2025, https://docs.victoriametrics.com/victoriametrics/vmalert/
10. vmalert - VictoriaMetrics, 檢索日期：8月 12, 2025, https://docs.victoriametrics.com/vmalert/
11. Fixing 503 Errors When Using Istio/Envoy - Medium, 檢索日期：8月 12, 2025, https://medium.com/@kburjack/fixing-503-errors-when-using-istio-envoy-bf63aa720826

12. 503 upstream connect error or disconnect/reset before headers. reset reason: connection termination #55138 - GitHub, 検索日期：8月 12, 2025, https://github.com/istio/istio/issues/55138
13. Troubleshooting Common 503 Errors in Kubernetes Applications | by SQANA - Medium, 検索日期：8月 12, 2025, https://medium.com/@sqanafrica/troubleshooting-common-503-errors-in-kubernetes-applications-1db3998e4071
14. Istio Give 503 error with no healthy upstream when pods get evicted - Networking, 検索日期：8月 12, 2025, https://discuss.istio.io/t/istio-give-503-error-with-no-healthy-upstream-when-pods-get-evicted/6069
15. Resolving Common Errors in Envoy Proxy: A Troubleshooting Guide - HashiCorp Support, 検索日期：8月 12, 2025, https://support.hashicorp.com/hc/en-us/articles/5295078989075-Resolving-Common-Errors-in-Envoy-Proxy-A-Troubleshooting-Guide
16. How to Fix Kubernetes 'Service 503' (Service Unavailable) Error - Komodor, 検索日期：8月 12, 2025, https://komodor.com/learn/how-to-fix-kubernetes-service-503-service-unavailable-error/
17. How to Troubleshoot Kubernetes Service 503 error(Service Unavailable) | by FoxuTech, 検索日期：8月 12, 2025, https://foxutech.medium.com/how-to-troubleshoot-kubernetes-service-503-error-service-unavailable-5ab4760d4d03
18. Kubernetes Operator: Custom resources: VMAlert - VictoriaMetrics, 検索日期：8月 12, 2025, https://docs.victoriametrics.com/operator/resources/vmalert/
19. Troubleshooting - VictoriaMetrics, 検索日期：8月 12, 2025, https://docs.victoriametrics.com/victoriametrics/troubleshooting/
20. Getting connection reset issue in pods | AWS re:Post, 検索日期：8月 12, 2025, https://repost.aws/questions/QU9u92HLUaSwuPm_y5SNQUCg/getting-connection-reset-issue-in-pods
21. kube-proxy Subtleties: Debugging an Intermittent Connection Reset | Kubernetes, 検索日期：8月 12, 2025, https://kubernetes.io/blog/2019/03/29/kube-proxy-subtleties-debugging-an-intermittent-connection-reset/
22. Istio / Virtual Service, 検索日期：8月 12, 2025, https://istio.io/latest/docs/reference/config/networking/virtual-service/
23. victoriametrics 0.1.28 · bitnami/bitnami - Artifact Hub, 検索日期：8月 12, 2025, https://artifacthub.io/packages/helm/bitnami/victoriametrics