Kubernetes 服務完整建置安裝指南

概述

本文檔提供完整的 Kubernetes 服務建置安裝步驟，包含 Ingress 控制器、監控系統、日誌管理、分散式追蹤和管理工具，確保按正確順序部署並配置固定 IP 地址。

環境資訊

- **Kubernetes 叢集**: hcch-k8s
- **節點數量**: 5 個（3 Master + 2 Worker）
- **安裝順序**: Nginx Ingress → Istio → Prometheus → Grafana → Jaeger → Kiali → Elasticsearch/Kibana → K8s Dashboard → Swagger UI

IP 地址分配

服務	IP 地址	端口	用途
Nginx Ingress	172.21.169.73	80/443	HTTP/HTTPS 路由
Istio Ingress	172.21.169.72	80/443	服務網格入口
Prometheus	172.21.169.75	9090	監控數據收集
Grafana	172.21.169.74	3000	監控視覺化
Jaeger UI	172.21.169.82	16686	分散式追蹤
Kiali	172.21.169.77	20001	服務網格視覺化
Kibana	172.21.169.71	5601	日誌搜尋和視覺化
K8s Dashboard	172.21.169.81	8443	叢集管理界面
Swagger UI	172.21.169.79	8080	API 文件和測試

第二階段：Ingress 控制器

1. Nginx Ingress Controller 安裝 (172.21.169.73)

1.1 前置條件檢查

```
# 檢查叢集狀態
kubectl get nodes
kubectl get pods -A | grep -E "(Running|Ready)"

# 檢查 Helm
helm version
```

```
# 檢查目標 IP 可用性
ping -c 3 172.21.169.73  # 應該無法連通
```

1.2 安裝 MetalLB（如未安裝）

```
# 檢查是否已安裝
kubectl get namespace metallb-system

# 如果不存在則安裝
kubectl apply -f
https://raw.githubusercontent.com/metallb/metallb/v0.13.12/config/manifest
s/metallb-native.yaml

# 等待啟動
kubectl wait --namespace metallb-system \
  --for=condition=ready pod \
  --selector=app=metallb \
  --timeout=90s
```

1.3 配置 MetalLB IP 地址池

```
cat > metallb-nginx-config.yaml << 'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: nginx-ingress-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.21.169.73/32
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: nginx-ingress-advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - nginx-ingress-pool
EOF

kubectl apply -f metallb-nginx-config.yaml
```

1.4 新增 Nginx Ingress Helm Repository

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update
```

1.5 創建 Nginx Values 配置

```
cat > nginx-values.yaml << 'EOF'
controller:
  replicaCount: 2

  resources:
    limits:
      cpu: 500m
      memory: 512Mi
    requests:
      cpu: 250m
      memory: 256Mi

  service:
    type: LoadBalancer
    loadBalancerIP: "172.21.169.73"
    externalTrafficPolicy: Local
    annotations:
      metallb.universe.tf/loadBalancerIPs: "172.21.169.73"

  ingressClassResource:
    name: nginx
    enabled: true
    default: true
    controllerValue: "k8s.io/ingress-nginx"

  config:
    log-format-json: "true"
    worker-processes: "auto"
    worker-connections: "2048"
    ssl-protocols: "TLSv1.2 TLSv1.3"
    client-max-body-size: "100m"
    proxy-connect-timeout: "60"
    proxy-send-timeout: "60"
    proxy-read-timeout: "60"
    use-forwarded-headers: "true"
    compute-full-forwarded-for: "true"

  metrics:
    enabled: true
    service:
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "10254"

defaultBackend:
```

```
  enabled: true
  replicaCount: 1
  resources:
    limits:
      cpu: 10m
      memory: 20Mi
    requests:
      cpu: 10m
      memory: 20Mi
EOF
```

1.6 安裝 Nginx Ingress

```
helm install nginx-ingress ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --create-namespace \
  --values nginx-values.yaml \
  --version 4.11.3
```

1.7 驗證 Nginx Ingress 安裝

```
# 檢查 Pods
kubectl get pods -n ingress-nginx

# 檢查 Service 和 IP
kubectl get svc -n ingress-nginx

# 測試連通性
curl -I http://172.21.169.73
```

預期結果：

```
NAME                                 TYPE           EXTERNAL-IP
PORT(S)
nginx-ingress-ingress-nginx-controller   LoadBalancer   172.21.169.73
80:xxxxx/TCP,443:xxxxx/TCP
```

---

2. Istio Ingress Gateway 安裝 (172.21.169.72)

2.1 安裝 Istio CLI

```
# 下載 Istio
curl -L https://istio.io/downloadIstio | sh -
```

```
# 移動到 PATH
sudo mv istio-*/bin/istioctl /usr/local/bin/

# 驗證安裝
istioctl version
```

2.2 安裝 Istio Control Plane

```
# 使用預設配置安裝
istioctl install --set values.defaultRevision=default -y

# 或使用自定義配置
cat > istio-control-plane.yaml << 'EOF'
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
metadata:
  name: control-plane
spec:
  values:
    defaultRevision: default
    pilot:
      traceSampling: 1.0
  components:
    pilot:
      k8s:
        resources:
          requests:
            cpu: 200m
            memory: 256Mi
          limits:
            cpu: 500m
            memory: 512Mi
EOF

istioctl install -f istio-control-plane.yaml -y
```

2.3 配置 MetalLB 為 Istio 分配 IP

```
cat > metallb-istio-config.yaml << 'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: istio-ingress-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.21.169.72/32
---
```

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: istio-ingress-advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - istio-ingress-pool
EOF

kubectl apply -f metallb-istio-config.yaml
```

2.4 安裝 Istio Ingress Gateway

```
# 創建 Istio Gateway 配置
cat > istio-gateway.yaml << 'EOF'
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
metadata:
  name: ingress-gateway
spec:
  components:
    ingressGateways:
    - name: istio-ingressgateway
      namespace: istio-system
      enabled: true
      k8s:
        service:
          type: LoadBalancer
          loadBalancerIP: "172.21.169.72"
          annotations:
            metallb.universe.tf/loadBalancerIPs: "172.21.169.72"
        resources:
          requests:
            cpu: 200m
            memory: 256Mi
          limits:
            cpu: 500m
            memory: 512Mi
        hpaSpec:
          minReplicas: 2
          maxReplicas: 5
EOF

istioctl install -f istio-gateway.yaml -y
```

2.5 驗證 Istio 安裝

```
# 檢查 Control Plane
kubectl get pods -n istio-system

# 檢查 Gateway Service
kubectl get svc -n istio-system istio-ingressgateway

# 測試連通性
curl -I http://172.21.169.72
```

2.6 創建測試 Gateway 和 VirtualService

```
cat > istio-test.yaml << 'EOF'
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
  name: test-gateway
  namespace: istio-system
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
---
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: test-vs
  namespace: istio-system
spec:
  hosts:
  - "*"
  gateways:
  - test-gateway
  http:
  - match:
    - uri:
        prefix: /
    route:
    - destination:
        host: nginx-ingress-ingress-nginx-controller.ingress-
nginx.svc.cluster.local
        port:
          number: 80
EOF
```

```
kubectl apply -f istio-test.yaml
```

第三階段：監控和可觀測性

3. Prometheus 安裝 (172.21.169.75:9090)

3.1 新增 Prometheus Helm Repository

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

3.2 配置 MetalLB 為 Prometheus 分配 IP

```
cat > metallb-prometheus-config.yaml << 'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: prometheus-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.21.169.75/32
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: prometheus-advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - prometheus-pool
EOF

kubectl apply -f metallb-prometheus-config.yaml
```

3.3 創建 Prometheus Values 配置

```
cat > prometheus-values.yaml << 'EOF'
# Prometheus Server 配置
prometheus:
  prometheusSpec:
    # 資源配置
```

```
    resources:
      requests:
        memory: 1Gi
        cpu: 500m
      limits:
        memory: 2Gi
        cpu: 1000m

    # 存儲配置
    storageSpec:
      volumeClaimTemplate:
        spec:
          storageClassName: nfs-storage
          accessModes: ["ReadWriteOnce"]
          resources:
            requests:
              storage: 50Gi

    # 保留時間
    retention: 30d
    retentionSize: 45GB

    # 外部 URL
    externalUrl: http://172.21.169.75:9090

    # 服務發現配置
    serviceMonitorSelectorNilUsesHelmValues: false
    podMonitorSelectorNilUsesHelmValues: false
    ruleSelectorNilUsesHelmValues: false

    # 額外的 scrape 配置
    additionalScrapeConfigs:
    - job_name: 'kubernetes-pods'
      kubernetes_sd_configs:
      - role: pod
      relabel_configs:
      - source_labels:
[__meta_kubernetes_pod_annotation_prometheus_io_scrape]
        action: keep
        regex: true
      - source_labels:
[__meta_kubernetes_pod_annotation_prometheus_io_path]
        action: replace
        target_label: __metrics_path__
        regex: (.+)
      - source_labels: [__address__,
__meta_kubernetes_pod_annotation_prometheus_io_port]
        action: replace
        regex: ([^:]+)(?::\d+)?;(\d+)
        replacement: $1:$2
        target_label: __address__

# Prometheus Server Service 配置
server:
```

```
    service:
      type: LoadBalancer
      loadBalancerIP: "172.21.169.75"
      annotations:
        metallb.universe.tf/loadBalancerIPs: "172.21.169.75"
      port: 9090

  # AlertManager 配置
  alertmanager:
    enabled: true
    alertmanagerSpec:
      resources:
        requests:
          memory: 256Mi
          cpu: 100m
        limits:
          memory: 512Mi
          cpu: 200m
      storage:
        volumeClaimTemplate:
          spec:
            storageClassName: nfs-storage
            accessModes: ["ReadWriteOnce"]
            resources:
              requests:
                storage: 5Gi

  # Grafana 整合（暫時禁用，稍後單獨安裝）
  grafana:
    enabled: false

  # Node Exporter 配置
  nodeExporter:
    enabled: true

  # Kube State Metrics 配置
  kubeStateMetrics:
    enabled: true

  # 服務監控器
  defaultRules:
    create: true
    rules:
      alertmanager: true
      etcd: true
      configReloaders: true
      general: true
      k8s: true
      kubeApiserverAvailability: true
      kubeApiserverBurnrate: true
      kubeApiserverHistogram: true
      kubeApiserverSlos: true
      kubelet: true
      kubeProxy: true
```

```
    kubePrometheusGeneral: true
    kubePrometheusNodeRecording: true
    kubernetesApps: true
    kubernetesResources: true
    kubernetesStorage: true
    kubernetesSystem: true
    network: true
    node: true
    nodeExporterAlerting: true
    nodeExporterRecording: true
    prometheus: true
    prometheusOperator: true

# RBAC 配置
rbac:
  create: true
EOF
```

3.4 安裝 Prometheus Stack

```
helm install prometheus prometheus-community/kube-prometheus-stack \
  --namespace monitoring \
  --create-namespace \
  --values prometheus-values.yaml \
  --version 61.1.1
```

3.5 驗證 Prometheus 安裝

```
# 檢查 Pods
kubectl get pods -n monitoring

# 檢查 Service
kubectl get svc -n monitoring | grep prometheus

# 檢查 PVC
kubectl get pvc -n monitoring

# 測試訪問
curl http://172.21.169.75:9090
```

3.6 配置 Nginx Ingress 規則（可選）

```
cat > prometheus-ingress.yaml << 'EOF'
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
```

```
  name: prometheus-ingress
  namespace: monitoring
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: prometheus.172.21.169.73.nip.io
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: prometheus-kube-prometheus-prometheus
            port:
              number: 9090
EOF

kubectl apply -f prometheus-ingress.yaml
```

---

4. Grafana 安裝 (172.21.169.74)

4.1 配置 MetalLB 為 Grafana 分配 IP

```
cat > metallb-grafana-config.yaml << 'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: grafana-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.21.169.74/32
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: grafana-advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - grafana-pool
EOF

kubectl apply -f metallb-grafana-config.yaml
```

4.2 新增 Grafana Helm Repository

```
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update
```

4.3 創建 Grafana Values 配置

```
cat > grafana-values.yaml << 'EOF'
# 副本配置
replicas: 1

# 資源配置
resources:
  limits:
    cpu: 500m
    memory: 512Mi
  requests:
    cpu: 250m
    memory: 256Mi

# 持久化存儲
persistence:
  enabled: true
  storageClassName: nfs-storage
  size: 10Gi
  accessModes:
    - ReadWriteOnce

# Service 配置
service:
  type: LoadBalancer
  loadBalancerIP: "172.21.169.74"
  port: 80
  targetPort: 3000
  annotations:
    metallb.universe.tf/loadBalancerIPs: "172.21.169.74"

# 管理員配置
adminUser: admin
adminPassword: Grafana123!

# Grafana 配置
grafana.ini:
  server:
    root_url: http://172.21.169.74
    serve_from_sub_path: false
  security:
    allow_embedding: true
    cookie_secure: false
  auth.anonymous:
    enabled: true
    org_role: Viewer
```

```
  dashboards:
    default_home_dashboard_path: /var/lib/grafana/dashboards/kubernetes-
cluster-monitoring.json

# 資料源配置
datasources:
  datasources.yaml:
    apiVersion: 1
    datasources:
    - name: Prometheus
      type: prometheus
      url: http://prometheus-kube-prometheus-
prometheus.monitoring.svc.cluster.local:9090
      access: proxy
      isDefault: true
      jsonData:
        timeInterval: 30s
        queryTimeout: 60s
        httpMethod: POST
    - name: Prometheus-AlertManager
      type: alertmanager
      url: http://prometheus-kube-prometheus-
alertmanager.monitoring.svc.cluster.local:9093
      access: proxy

# 儀表板提供者
dashboardProviders:
  dashboardproviders.yaml:
    apiVersion: 1
    providers:
    - name: 'kubernetes'
      orgId: 1
      folder: 'Kubernetes'
      type: file
      disableDeletion: false
      editable: true
      allowUiUpdates: true
      options:
        path: /var/lib/grafana/dashboards/kubernetes
    - name: 'istio'
      orgId: 1
      folder: 'Istio'
      type: file
      disableDeletion: false
      editable: true
      allowUiUpdates: true
      options:
        path: /var/lib/grafana/dashboards/istio

# 預設儀表板
dashboards:
  kubernetes:
    # Kubernetes 叢集監控
    kubernetes-cluster-monitoring:
```

```
      gnetId: 7249
      revision: 1
      datasource: Prometheus
    # Node Exporter 完整
    node-exporter-full:
      gnetId: 1860
      revision: 31
      datasource: Prometheus
    # Kubernetes 部署監控
    kubernetes-deployment:
      gnetId: 8588
      revision: 1
      datasource: Prometheus
    # Kubernetes Pod 監控
    kubernetes-pods:
      gnetId: 6336
      revision: 1
      datasource: Prometheus
  istio:
    # Istio 控制平面儀表板
    istio-control-plane:
      gnetId: 7645
      revision: 75
      datasource: Prometheus
    # Istio 服務儀表板
    istio-service:
      gnetId: 7636
      revision: 75
      datasource: Prometheus
    # Istio 工作負載儀表板
    istio-workload:
      gnetId: 7630
      revision: 75
      datasource: Prometheus

# 插件配置
plugins:
  - grafana-piechart-panel
  - grafana-worldmap-panel
  - grafana-clock-panel

# 環境變數
env:
  GF_EXPLORE_ENABLED: true
  GF_PANELS_DISABLE_SANITIZE_HTML: true
  GF_LOG_FILTERS: rendering:debug

# RBAC 配置
rbac:
  create: true
  pspEnabled: false

# ServiceAccount 配置
serviceAccount:
```

```
  create: true

# 安全上下文
securityContext:
  runAsUser: 472
  runAsGroup: 472
  fsGroup: 472

# 初始化容器（預載儀表板）
initChownData:
  enabled: true
  resources:
    limits:
      cpu: 100m
      memory: 128Mi
    requests:
      cpu: 50m
      memory: 64Mi
EOF
```

4.4 安裝 Grafana

```
helm install grafana grafana/grafana \
  --namespace monitoring \
  --values grafana-values.yaml \
  --version 8.5.1
```

4.5 驗證 Grafana 安裝

```
# 檢查 Pod
kubectl get pods -n monitoring | grep grafana

# 檢查 Service
kubectl get svc -n monitoring | grep grafana

# 檢查 PVC
kubectl get pvc -n monitoring | grep grafana

# 測試訪問
curl -I http://172.21.169.74
```

4.6 獲取 Grafana 登入資訊

```
# 獲取管理員密碼（如果使用自動生成）
kubectl get secret --namespace monitoring grafana -o jsonpath="
{.data.admin-password}" | base64 --decode ; echo
```

```
# 或使用配置中設定的密碼
echo "Username: admin"
echo "Password: Grafana123!"
```

4.7 配置 Nginx Ingress 規則（可選）

```
cat > grafana-ingress.yaml << 'EOF'
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: grafana-ingress
  namespace: monitoring
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: grafana.172.21.169.73.nip.io
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: grafana
            port:
              number: 80
EOF

kubectl apply -f grafana-ingress.yaml
```

整體驗證和測試

1. 檢查所有服務狀態

```
# 檢查所有命名空間的 Pods
kubectl get pods -A | grep -E "(nginx|istio|prometheus|grafana)"

# 檢查所有 LoadBalancer Services
kubectl get svc -A | grep LoadBalancer

# 檢查 MetalLB IP 分配
kubectl get svc -A -o wide | grep -E "(172.21.169.7[2-5])"
```

2. 連通性測試

```
# 測試所有服務
echo "Testing Nginx Ingress..."
curl -I http://172.21.169.73

echo "Testing Istio Ingress..."
curl -I http://172.21.169.72

echo "Testing Prometheus..."
curl -I http://172.21.169.75:9090

echo "Testing Grafana..."
curl -I http://172.21.169.74
```

3. 功能驗證

```
# 檢查 Prometheus targets
curl -s http://172.21.169.75:9090/api/v1/targets | jq
'.data.activeTargets[].labels.job' | sort | uniq

# 檢查 Grafana datasources
curl -s -u admin:Grafana123! http://172.21.169.74/api/datasources | jq '.
[].name'
```

4. 部署測試應用

```
cat > test-microservice.yaml << 'EOF'
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-app
  namespace: default
  labels:
    app: test-app
    version: v1
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test-app
      version: v1
  template:
    metadata:
      labels:
        app: test-app
        version: v1
```

```
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "8080"
        prometheus.io/path: "/metrics"
    spec:
      containers:
      - name: test-app
        image: nginx:1.21
        ports:
        - containerPort: 80
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: test-app-service
  namespace: default
  labels:
    app: test-app
spec:
  selector:
    app: test-app
  ports:
  - name: http
    port: 80
    targetPort: 80
  - name: metrics
    port: 8080
    targetPort: 8080
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-app-nginx-ingress
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: test-nginx.172.21.169.73.nip.io
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: test-app-service
            port:
              number: 80
---
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
```

```
  name: test-app-istio-vs
  namespace: default
spec:
  hosts:
  - test-istio.172.21.169.72.nip.io
  gateways:
  - istio-system/test-gateway
  http:
  - route:
    - destination:
        host: test-app-service
        port:
          number: 80
EOF

kubectl apply -f test-microservice.yaml
```

5. 測試完整流程

```
# 測試 Nginx Ingress 路由
curl -H "Host: test-nginx.172.21.169.73.nip.io" http://172.21.169.73

# 測試 Istio Ingress 路由
curl -H "Host: test-istio.172.21.169.72.nip.io" http://172.21.169.72

# 檢查 Prometheus 中的目標
curl -s http://172.21.169.75:9090/api/v1/targets | jq
'.data.activeTargets[] | select(.labels.job=="kubernetes-pods")'

# 訪問 Grafana 儀表板
echo "Open http://172.21.169.74 in browser"
echo "Login: admin / Grafana123!"
```

故障排除

常見問題及解決方案

1. IP 地址無法分配

```
# 檢查 MetalLB 狀態
kubectl get pods -n metallb-system
kubectl logs -n metallb-system deployment/controller

# 檢查 IP 地址池
kubectl get ipaddresspool -n metallb-system
```

2. Pod 無法啟動

```
# 檢查 Pod 事件
kubectl describe pod <pod-name> -n <namespace>

# 檢查節點資源
kubectl top nodes
kubectl describe nodes
```

3. 持久化存儲問題

```
# 檢查 PVC 狀態
kubectl get pvc -A

# 檢查 StorageClass
kubectl get storageclass

# 檢查 NFS Provisioner
kubectl get pods -n nfs
```

4. 服務間連通性問題

```
# 檢查 DNS 解析
kubectl run test-pod --image=busybox --rm -it -- nslookup prometheus-kube-
prometheus-prometheus.monitoring.svc.cluster.local

# 檢查網路策略
kubectl get networkpolicy -A
```

清理指令

如需移除所有服務：

```
# 刪除測試應用
kubectl delete -f test-microservice.yaml

# 刪除 Grafana
helm uninstall grafana -n monitoring

# 刪除 Prometheus
helm uninstall prometheus -n monitoring

# 刪除 Istio
```

```
istioctl uninstall --purge -y

# 刪除 Nginx Ingress
helm uninstall nginx-ingress -n ingress-nginx

# 刪除命名空間
kubectl delete namespace monitoring ingress-nginx istio-system

# 清理 MetalLB 配置
kubectl delete -f metallb-*-config.yaml
```

第四階段：分散式追蹤

5. Jaeger 安裝 (172.21.169.82)

5.1 配置 MetalLB 為 Jaeger 分配 IP

```
cat > metallb-jaeger-config.yaml << 'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: jaeger-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.21.169.82/32
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: jaeger-advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - jaeger-pool
EOF

kubectl apply -f metallb-jaeger-config.yaml
```

5.2 安裝 Jaeger Operator

```
# 創建命名空間
kubectl create namespace observability

# 安裝 Jaeger Operator
kubectl create -f https://github.com/jaegertracing/jaeger-
operator/releases/download/v1.57.0/jaeger-operator.yaml -n observability
```

```
# 等待 Operator 啟動
kubectl wait --for=condition=available deployment/jaeger-operator -n
observability --timeout=300s
```

5.3 創建 Jaeger 實例配置

```
cat > jaeger-instance.yaml << 'EOF'
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: jaeger-production
  namespace: observability
spec:
  strategy: production

  # Collector 配置
  collector:
    replicas: 2
    resources:
      requests:
        cpu: 200m
        memory: 256Mi
      limits:
        cpu: 500m
        memory: 512Mi
    options:
      collector:
        zipkin:
          host-port: ":9411"

  # Query 服務配置 (UI)
  query:
    replicas: 2
    resources:
      requests:
        cpu: 100m
        memory: 128Mi
      limits:
        cpu: 200m
        memory: 256Mi
    options:
      query:
        base-path: /
    service:
      type: LoadBalancer
      loadBalancerIP: "172.21.169.82"
      annotations:
        metallb.universe.tf/loadBalancerIPs: "172.21.169.82"
      ports:
      - name: query-http
```

```
        port: 16686
        targetPort: 16686

  # Agent 配置
  agent:
    strategy: DaemonSet  # 在每個節點運行
    resources:
      requests:
        cpu: 50m
        memory: 64Mi
      limits:
        cpu: 100m
        memory: 128Mi

  # 存儲配置
  storage:
    type: elasticsearch
    elasticsearch:
      nodeCount: 1
      redundancyPolicy: ZeroRedundancy
      resources:
        requests:
          cpu: 500m
          memory: 1Gi
        limits:
          cpu: 1000m
          memory: 2Gi
      storage:
        storageClassName: nfs-storage
        size: 20Gi
    options:
      es:
        server-urls: http://elasticsearch:9200
        index-prefix: jaeger
EOF

kubectl apply -f jaeger-instance.yaml
```

5.4 配置 Istio 與 Jaeger 整合

```
# 更新 Istio 配置啟用 Jaeger 追蹤
kubectl patch configmap istio -n istio-system --type merge -p '{
  "data": {
    "mesh": "defaultConfig:\n  proxyStatsMatcher:\n    inclusionRegexps:\n
- \".*outlier_detection.*\"\n    - \".*circuit_breakers.*\"\n    -
\".*upstream_rq_retry.*\"\n    - \".*_cx_.*\"\ndefaultProviders:\n
tracing:\n  - jaeger\nextensionProviders:\n- name: jaeger\n
envoyExtAuthzHttp:\n    service: jaeger-production-
collector.observability.svc.cluster.local\n    port: 14268\n  zipkin:\n
service: jaeger-production-collector.observability.svc.cluster.local\n
port: 9411"
```

```
    }
  }'

  # 重啟 Istio 組件以應用配置
  kubectl rollout restart deployment/istiod -n istio-system
  kubectl rollout restart daemonset/istio-proxy -n istio-system || true
```

5.5 驗證 Jaeger 安裝

```
# 檢查 Jaeger 組件
kubectl get pods -n observability

# 檢查 Jaeger 服務
kubectl get svc -n observability

# 測試 Jaeger UI
curl -I http://172.21.169.82:16686

# 檢查 Elasticsearch
kubectl get pods -n observability | grep elasticsearch
```

5.6 配置 Nginx Ingress 規則（可選）

```
cat > jaeger-ingress.yaml << 'EOF'
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: jaeger-ingress
  namespace: observability
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: jaeger.172.21.169.73.nip.io
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: jaeger-production-query
            port:
              number: 16686
EOF

kubectl apply -f jaeger-ingress.yaml
```

第五階段：服務網格可觀測性

6. Kiali 安裝 (172.21.169.77:20001)

6.1 配置 MetalLB 為 Kiali 分配 IP

```
cat > metallb-kiali-config.yaml << 'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: kiali-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.21.169.77/32
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: kiali-advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - kiali-pool
EOF

kubectl apply -f metallb-kiali-config.yaml
```

6.2 安裝 Kiali Operator

```
# 新增 Kiali Helm Repository
helm repo add kiali https://kiali.org/helm-charts
helm repo update

# 安裝 Kiali Operator
helm install \
  --namespace kiali-operator \
  --create-namespace \
  kiali-operator \
  kiali/kiali-operator \
  --version 1.86.0
```

6.3 創建 Kiali 實例配置

```
cat > kiali-instance.yaml << 'EOF'
apiVersion: kiali.io/v1alpha1
```

```
kind: Kiali
metadata:
  name: kiali
  namespace: istio-system
spec:
  # 安裝配置
  installation_tag: "v1.86.0"

  # 認證配置
  auth:
    strategy: "anonymous"  # 生產環境建議使用 "openshift" 或 "token"

  # 部署配置
  deployment:
    replicas: 1
    resources:
      requests:
        cpu: 200m
        memory: 256Mi
      limits:
        cpu: 500m
        memory: 512Mi

    # Service 配置
    service_type: "LoadBalancer"
    service_annotations:
      metallb.universe.tf/loadBalancerIPs: "172.21.169.77"
    load_balancer_ip: "172.21.169.77"

    # 自定義端口
    http_port: 20001

  # 外部服務配置
  external_services:
    # Prometheus 配置
    prometheus:
      url: "http://prometheus-kube-prometheus-
prometheus.monitoring.svc.cluster.local:9090"

    # Grafana 配置
    grafana:
      enabled: true
      url: "http://grafana.monitoring.svc.cluster.local"
      in_cluster_url: "http://grafana.monitoring.svc.cluster.local"

    # Jaeger 配置
    tracing:
      enabled: true
      in_cluster_url: "http://jaeger-production-
query.observability.svc.cluster.local:16686"
      url: "http://172.21.169.82:16686"

  # Istio 配置
  istio_namespace: "istio-system"
```

```
  # API 配置
  api:
    namespaces:
      exclude:
      - "kube-.*"
      - "openshift.*"
      - "metallb-system"
      - "nfs"

  # 伺服器配置
  server:
    web_root: "/"
    web_fqdn: "172.21.169.77"
    web_port: 20001

  # 額外配置
  kiali_feature_flags:
    certificates_information_indicators:
      enabled: true
    clustering:
      enabled: false
    disabled_features: []
    validations:
      ignore: ["KIA1301"]
EOF

kubectl apply -f kiali-instance.yaml
```

6.4 等待 Kiali 部署完成

```
# 等待 Kiali Pod 啟動
kubectl wait --for=condition=Ready pod -l app=kiali -n istio-system --
timeout=300s

# 檢查 Kiali 狀態
kubectl get pods -n istio-system | grep kiali
kubectl get svc -n istio-system | grep kiali
```

6.5 驗證 Kiali 安裝

```
# 測試 Kiali UI
curl -I http://172.21.169.77:20001

# 檢查 Kiali 配置
kubectl get kiali -n istio-system kiali -o yaml
```

6.6 配置 Istio Sidecar 注入

```
# 為測試命名空間啟用 Istio 注入
kubectl label namespace default istio-injection=enabled

# 重啟現有的 Pod 以注入 Sidecar
kubectl rollout restart deployment -n default
```

第六階段：日誌管理

7. Elasticsearch 和 Kibana 安裝 (172.21.169.71:5601)

7.1 配置 MetalLB 為 Kibana 分配 IP

```
cat > metallb-kibana-config.yaml << 'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: kibana-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.21.169.71/32
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: kibana-advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - kibana-pool
EOF

kubectl apply -f metallb-kibana-config.yaml
```

7.2 新增 Elastic Helm Repository

```
helm repo add elastic https://helm.elastic.co
helm repo update
```

7.3 安裝 Elasticsearch

```
cat > elasticsearch-values.yaml << 'EOF'
# Elasticsearch 配置
```

```
replicas: 1
minimumMasterNodes: 1

# 資源配置
resources:
  requests:
    cpu: 500m
    memory: 1Gi
  limits:
    cpu: 1000m
    memory: 2Gi

# JVM 設定
esJavaOpts: "-Xmx1g -Xms1g"

# 持久化存儲
persistence:
  enabled: true
  storageClass: "nfs-storage"
  accessModes:
    - ReadWriteOnce
  size: 30Gi

# 叢集設定
clusterName: "elasticsearch"
nodeGroup: "master"

# 安全設定
esConfig:
  elasticsearch.yml: |
    xpack.security.enabled: false
    xpack.security.transport.ssl.enabled: false
    xpack.security.http.ssl.enabled: false

# Service 配置
service:
  type: ClusterIP
  port: 9200

# 健康檢查
readinessProbe:
  failureThreshold: 3
  initialDelaySeconds: 10
  periodSeconds: 10
  successThreshold: 3
  timeoutSeconds: 5
EOF

helm install elasticsearch elastic/elasticsearch \
  --namespace logging \
  --create-namespace \
  --values elasticsearch-values.yaml \
  --version 8.5.1
```

7.4 安裝 Kibana

```
cat > kibana-values.yaml << 'EOF'
# Elasticsearch 連接配置
elasticsearchHosts: "http://elasticsearch-
master.logging.svc.cluster.local:9200"

# 副本配置
replicas: 1

# 資源配置
resources:
  requests:
    cpu: 500m
    memory: 1Gi
  limits:
    cpu: 1000m
    memory: 2Gi

# Service 配置
service:
  type: LoadBalancer
  loadBalancerIP: "172.21.169.71"
  port: 5601
  annotations:
    metallb.universe.tf/loadBalancerIPs: "172.21.169.71"

# Kibana 配置
kibanaConfig:
  kibana.yml: |
    server.host: "0.0.0.0"
    server.port: 5601
    elasticsearch.hosts: ["http://elasticsearch-
master.logging.svc.cluster.local:9200"]
    server.publicBaseUrl: "http://172.21.169.71:5601"

    # 安全設定
    xpack.security.enabled: false
    xpack.encryptedSavedObjects.encryptionKey:
"fhjskloppd678ehkdfdlliverpoolfcr"

    # 預設索引模式
    kibana.index: ".kibana"

    # 日誌設定
    logging.dest: stdout
    logging.silent: false
    logging.quiet: false
    logging.verbose: true

# 環境變數
```

```
extraEnvs:
  - name: "NODE_OPTIONS"
    value: "--max-old-space-size=1800"
  - name: "KIBANA_SYSTEM_PASSWORD"
    value: "kibana123"

# 健康檢查
healthCheckPath: "/app/kibana"

# 額外配置
serverHost: "0.0.0.0"

# 生命週期鉤子
lifecycle:
  preStop:
    exec:
      command: ["/bin/bash", "-c", "sleep 20"]
EOF

helm install kibana elastic/kibana \
  --namespace logging \
  --values kibana-values.yaml \
  --version 8.5.1
```

7.5 安裝 Filebeat（日誌收集器）

```
cat > filebeat-values.yaml << 'EOF'
# DaemonSet 配置
deployment:
  replicas: 1

# 資源配置
resources:
  requests:
    cpu: 100m
    memory: 100Mi
  limits:
    cpu: 200m
    memory: 200Mi

# Filebeat 配置
filebeatConfig:
  filebeat.yml: |
    filebeat.inputs:
    - type: container
      paths:
        - /var/log/containers/*.log
      processors:
        - add_kubernetes_metadata:
            host: ${NODE_NAME}
            matchers:
```

```
            - logs_path:
                logs_path: "/var/log/containers/"

    output.elasticsearch:
      host: '${NODE_NAME}'
      hosts: ["http://elasticsearch-
master.logging.svc.cluster.local:9200"]
      indices:
        - index: "filebeat-kubernetes-%{+yyyy.MM.dd}"

    setup.template.name: "filebeat-kubernetes"
    setup.template.pattern: "filebeat-kubernetes-*"
    setup.template.settings:
      index.number_of_shards: 1
      index.number_of_replicas: 0

    setup.kibana:
      host: "http://kibana-kibana.logging.svc.cluster.local:5601"

# 掛載配置
extraVolumes:
  - name: varlog
    hostPath:
      path: /var/log
  - name: varlibdockercontainers
    hostPath:
      path: /var/lib/docker/containers

extraVolumeMounts:
  - name: varlog
    mountPath: /var/log
    readOnly: true
  - name: varlibdockercontainers
    mountPath: /var/lib/docker/containers
    readOnly: true

# 環境變數
extraEnvs:
  - name: NODE_NAME
    valueFrom:
      fieldRef:
        fieldPath: spec.nodeName
EOF

helm install filebeat elastic/filebeat \
  --namespace logging \
  --values filebeat-values.yaml \
  --version 8.5.1
```

7.6 驗證 ELK Stack 安裝

```
# 檢查所有組件
kubectl get pods -n logging

# 檢查 Elasticsearch 叢集健康狀態
kubectl exec -n logging elasticsearch-master-0 -- curl -s
http://localhost:9200/_cluster/health?pretty

# 檢查 Kibana 服務
kubectl get svc -n logging kibana-kibana

# 測試 Kibana UI
curl -I http://172.21.169.71:5601
```

第七階段：管理工具

8. Kubernetes Dashboard 安裝 (172.21.169.81)

8.1 配置 MetalLB 為 Dashboard 分配 IP

```
cat > metallb-dashboard-config.yaml << 'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: dashboard-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.21.169.81/32
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: dashboard-advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - dashboard-pool
EOF

kubectl apply -f metallb-dashboard-config.yaml
```

8.2 安裝 Kubernetes Dashboard

```
# 安裝 Dashboard
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/r
```

```
ecommended.yaml

# 修改 Service 為 LoadBalancer
kubectl patch svc kubernetes-dashboard -n kubernetes-dashboard -p
'{"spec":{"type":"LoadBalancer","loadBalancerIP":"172.21.169.81"}}'

# 添加 MetalLB 註解
kubectl annotate svc kubernetes-dashboard -n kubernetes-dashboard
metallb.universe.tf/loadBalancerIPs="172.21.169.81"
```

8.3 創建管理員用戶

```
cat > dashboard-admin.yaml << 'EOF'
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: v1
kind: Secret
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
  annotations:
    kubernetes.io/service-account.name: "admin-user"
type: kubernetes.io/service-account-token
EOF

kubectl apply -f dashboard-admin.yaml
```

8.4 獲取訪問 Token

```
# 獲取訪問 Token
kubectl get secret admin-user -n kubernetes-dashboard -o jsonpath=
```

```
{".data.token"} | base64 -d
echo
```

8.5 配置 Nginx Ingress（HTTPS 訪問）

```
cat > dashboard-ingress.yaml << 'EOF'
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kubernetes-dashboard-ingress
  namespace: kubernetes-dashboard
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: dashboard.172.21.169.73.nip.io
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: kubernetes-dashboard
            port:
              number: 443
EOF

kubectl apply -f dashboard-ingress.yaml
```

8.6 驗證 Dashboard 安裝

```
# 檢查 Dashboard 組件
kubectl get pods -n kubernetes-dashboard

# 檢查服務
kubectl get svc -n kubernetes-dashboard

# 測試 HTTPS 訪問
curl -k -I https://172.21.169.81:8443
```

---

9. Swagger UI 安裝 (172.21.169.79)

9.1 配置 MetalLB 為 Swagger UI 分配 IP

```
cat > metallb-swagger-config.yaml << 'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: swagger-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.21.169.79/32
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: swagger-advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - swagger-pool
EOF

kubectl apply -f metallb-swagger-config.yaml
```

9.2 創建 Swagger UI 部署

```
cat > swagger-ui.yaml << 'EOF'
apiVersion: v1
kind: ConfigMap
metadata:
  name: swagger-config
  namespace: default
data:
  swagger.yaml: |
    openapi: 3.0.0
    info:
      title: Kubernetes Cluster APIs
      description: API documentation for the Kubernetes cluster services
      version: 1.0.0
    servers:
      - url: http://172.21.169.73
        description: Nginx Ingress Gateway
      - url: http://172.21.169.72
        description: Istio Ingress Gateway
    paths:
      /health:
        get:
          summary: Health check endpoint
          responses:
            '200':
              description: Service is healthy
      /metrics:
```

```
        get:
          summary: Prometheus metrics endpoint
          responses:
            '200':
              description: Metrics data
    components:
      schemas:
        HealthCheck:
          type: object
          properties:
            status:
              type: string
              example: "healthy"
            timestamp:
              type: string
              format: date-time
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: swagger-ui
  namespace: default
  labels:
    app: swagger-ui
spec:
  replicas: 1
  selector:
    matchLabels:
      app: swagger-ui
  template:
    metadata:
      labels:
        app: swagger-ui
    spec:
      containers:
      - name: swagger-ui
        image: swaggerapi/swagger-ui:v5.9.0
        ports:
        - containerPort: 8080
        env:
        - name: SWAGGER_JSON_URL
          value: "/swagger.yaml"
        - name: BASE_URL
          value: "/"
        resources:
          requests:
            cpu: 50m
            memory: 64Mi
          limits:
            cpu: 100m
            memory: 128Mi
        volumeMounts:
        - name: swagger-config
          mountPath: /usr/share/nginx/html/swagger.yaml
```

```
          subPath: swagger.yaml
      volumes:
      - name: swagger-config
        configMap:
          name: swagger-config
---
apiVersion: v1
kind: Service
metadata:
  name: swagger-ui-service
  namespace: default
  labels:
    app: swagger-ui
spec:
  type: LoadBalancer
  loadBalancerIP: "172.21.169.79"
  selector:
    app: swagger-ui
  ports:
  - name: http
    port: 8080
    targetPort: 8080
  annotations:
    metallb.universe.tf/loadBalancerIPs: "172.21.169.79"
EOF

kubectl apply -f swagger-ui.yaml
```

9.3 配置 Nginx Ingress 規則

```
cat > swagger-ingress.yaml << 'EOF'
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: swagger-ui-ingress
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: swagger.172.21.169.73.nip.io
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: swagger-ui-service
            port:
              number: 8080
```

```
EOF

kubectl apply -f swagger-ingress.yaml
```

9.4 驗證 Swagger UI 安裝

```
# 檢查 Swagger UI Pod
kubectl get pods -n default | grep swagger

# 檢查服務
kubectl get svc -n default swagger-ui-service

# 測試訪問
curl -I http://172.21.169.79:8080
```

更新後的整體驗證

1. 檢查所有服務狀態

```
# 檢查所有命名空間的 Pods
kubectl get pods -A | grep -E "
(nginx|istio|prometheus|grafana|jaeger|kiali|elasticsearch|kibana|dashboar
d|swagger)"

# 檢查所有 LoadBalancer Services
kubectl get svc -A | grep LoadBalancer

# 檢查 MetalLB IP 分配
kubectl get svc -A -o wide | grep -E "(172.21.169.7[1-9]|172.21.169.8[0-
2])"
```

2. 完整連通性測試

```
echo "=== 連通性測試 ==="
echo "Testing Nginx Ingress (172.21.169.73)..."
curl -I http://172.21.169.73

echo "Testing Istio Ingress (172.21.169.72)..."
curl -I http://172.21.169.72

echo "Testing Prometheus (172.21.169.75:9090)..."
curl -I http://172.21.169.75:9090

echo "Testing Grafana (172.21.169.74)..."
```

```
curl -I http://172.21.169.74

echo "Testing Jaeger UI (172.21.169.82:16686)..."
curl -I http://172.21.169.82:16686

echo "Testing Kiali (172.21.169.77:20001)..."
curl -I http://172.21.169.77:20001

echo "Testing Kibana (172.21.169.71:5601)..."
curl -I http://172.21.169.71:5601

echo "Testing K8s Dashboard (172.21.169.81:8443)..."
curl -k -I https://172.21.169.81:8443

echo "Testing Swagger UI (172.21.169.79:8080)..."
curl -I http://172.21.169.79:8080
```

3. 功能整合測試

```
# 測試 Jaeger 追蹤功能
kubectl exec -n istio-system $(kubectl get pod -n istio-system -l
app=istiod -o jsonpath='{.items[0].metadata.name}') -- \
  curl -s http://jaeger-production-
collector.observability.svc.cluster.local:14268/api/traces

# 測試 Kiali 服務網格視覺化
curl -s http://172.21.169.77:20001/api/namespaces | jq '.[] | .name'

# 測試 Elasticsearch 索引
kubectl exec -n logging elasticsearch-master-0 -- curl -s
http://localhost:9200/_cat/indices

# 獲取 Dashboard Token
echo "Kubernetes Dashboard Token:"
kubectl get secret admin-user -n kubernetes-dashboard -o jsonpath=
{".data.token"} | base64 -d
echo
```

更新後的清理指令

如需移除所有服務：

```
# 刪除應用服務
kubectl delete -f swagger-ui.yaml
kubectl delete -f dashboard-admin.yaml
kubectl delete -f test-microservice.yaml
```

```
# 卸載 Helm releases
helm uninstall swagger-ui -n default || true
helm uninstall filebeat -n logging
helm uninstall kibana -n logging
helm uninstall elasticsearch -n logging
helm uninstall kiali-operator -n kiali-operator
helm uninstall grafana -n monitoring
helm uninstall prometheus -n monitoring
helm uninstall nginx-ingress -n ingress-nginx

# 刪除 Jaeger 和 Kiali 實例
kubectl delete -f jaeger-instance.yaml
kubectl delete -f kiali-instance.yaml

# 刪除 Kubernetes Dashboard
kubectl delete -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/r
ecommended.yaml

# 刪除 Istio
istioctl uninstall --purge -y

# 刪除 Jaeger Operator
kubectl delete -f https://github.com/jaegertracing/jaeger-
operator/releases/download/v1.57.0/jaeger-operator.yaml -n observability

# 刪除命名空間
kubectl delete namespace monitoring logging observability istio-system
ingress-nginx kubernetes-dashboard kiali-operator

# 清理 MetalLB 配置
kubectl delete -f metallb-*-config.yaml

# 清理配置文件
rm -f *.yaml
```

總結

完成以上步驟後，你將擁有：

✅ **Nginx Ingress Controller** (172.21.169.73) - HTTP/HTTPS 路由 ✅ **Istio Ingress Gateway** (172.21.169.72) - 服務網格入口
✅ **Prometheus** (172.21.169.75:9090) - 監控數據收集 ✅ **Grafana** (172.21.169.74) - 監控視覺化 ✅ **Jaeger UI** (172.21.169.82:16686) - 分散式追蹤 ✅ **Kiali** (172.21.169.77:20001) - 服務網格視覺化 ✅ **Kibana** (172.21.169.71:5601) - 日誌搜尋和視覺化 ✅ **Kubernetes Dashboard** (172.21.169.81:8443) - 叢集管理界面
✅ **Swagger UI** (172.21.169.79:8080) - API 文件和測試

所有服務都配置了固定 IP 地址，並且整合了完整的監控、日誌、追蹤和管理功能，構成了一個完整的微服務可觀測性平台。