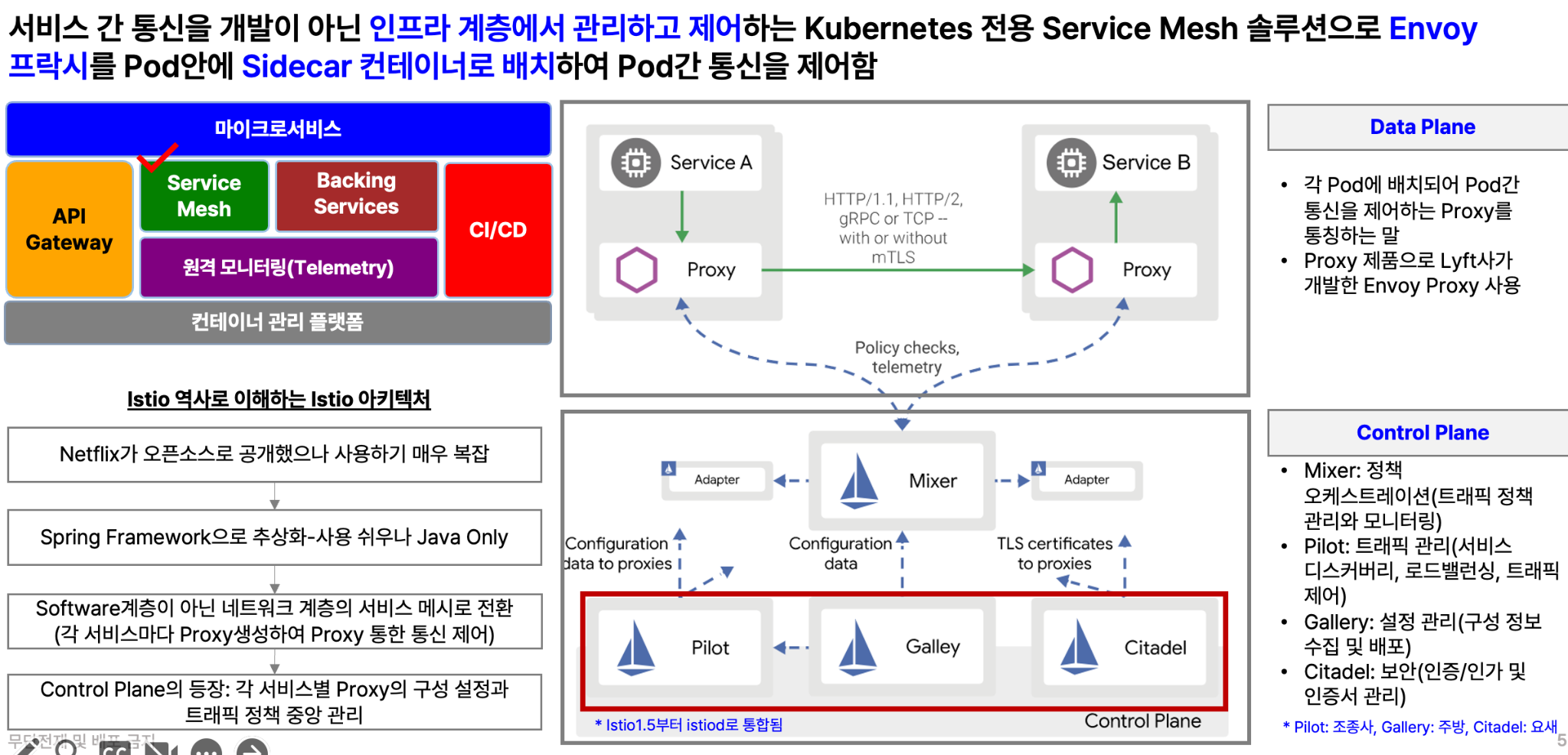
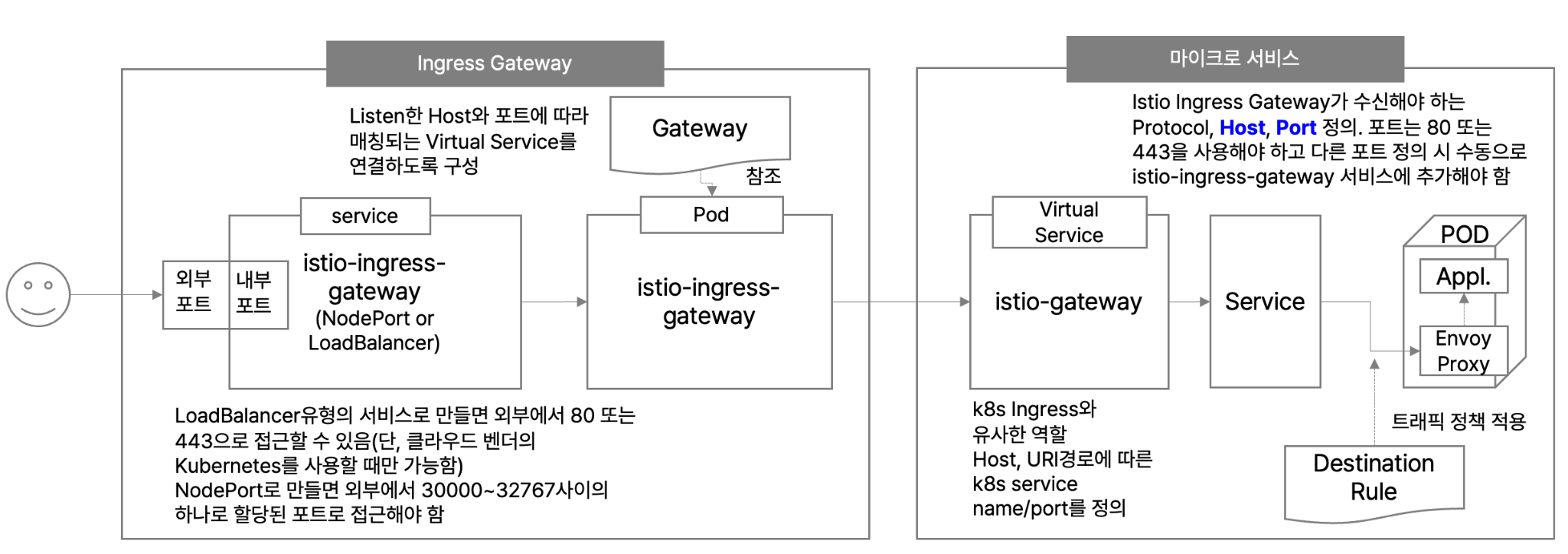
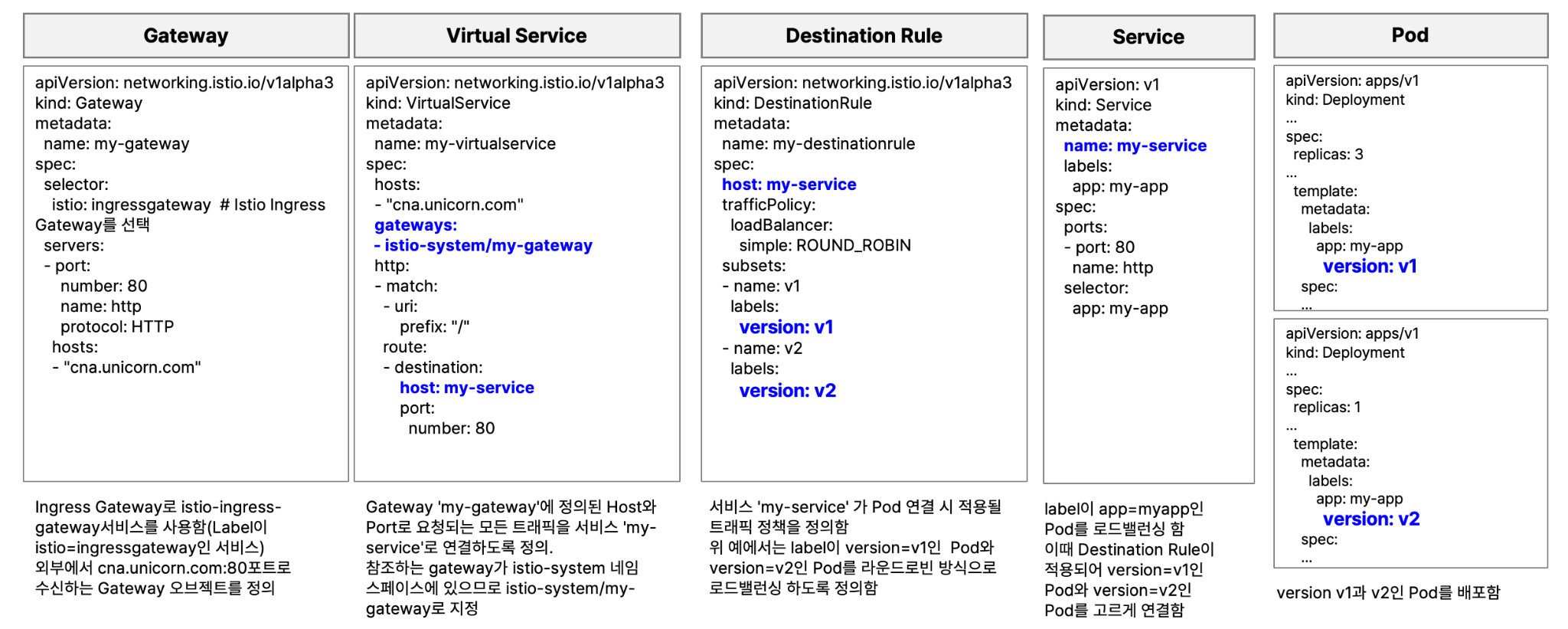
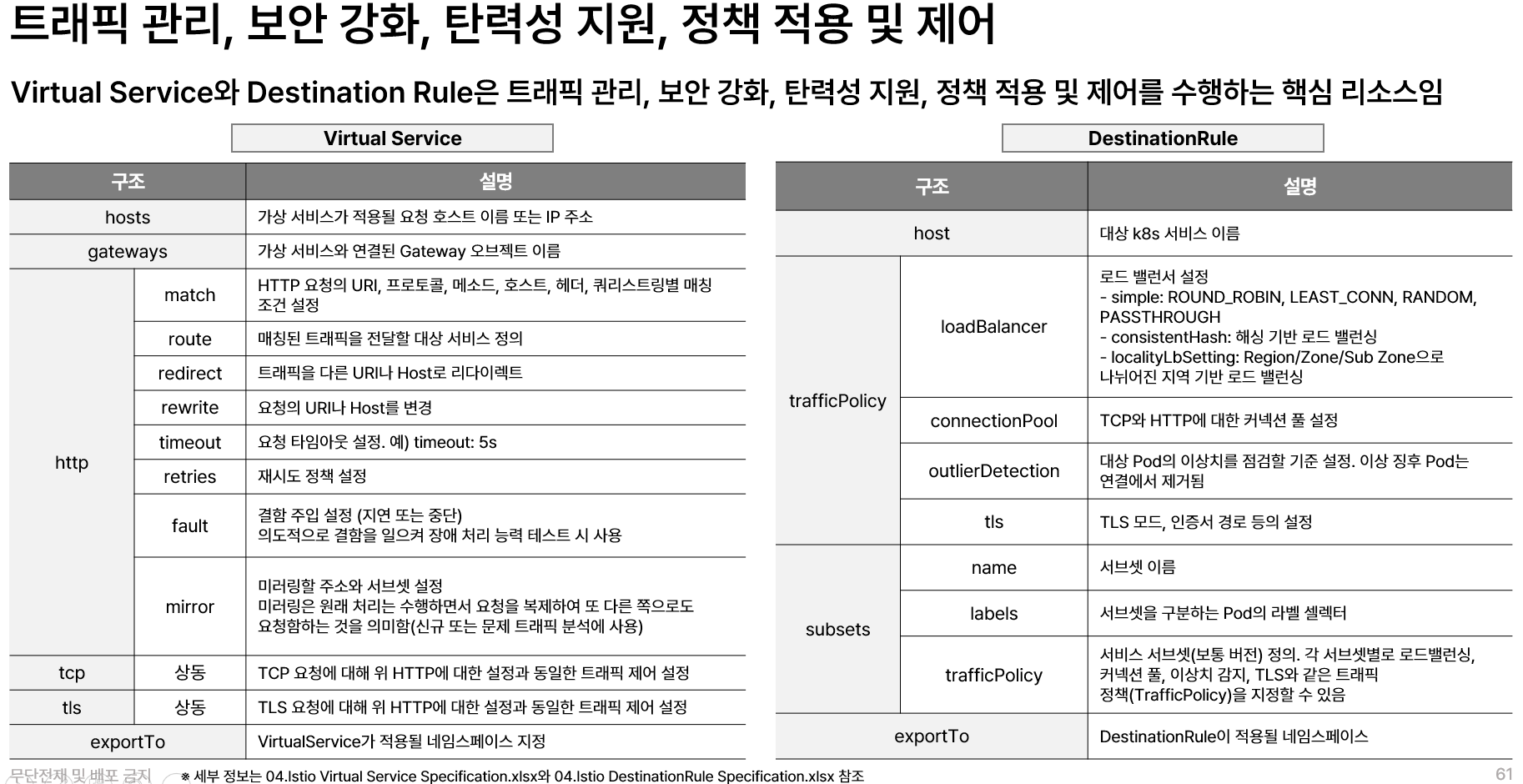
Istio 설치 및 사용 예제

## Istio Overview







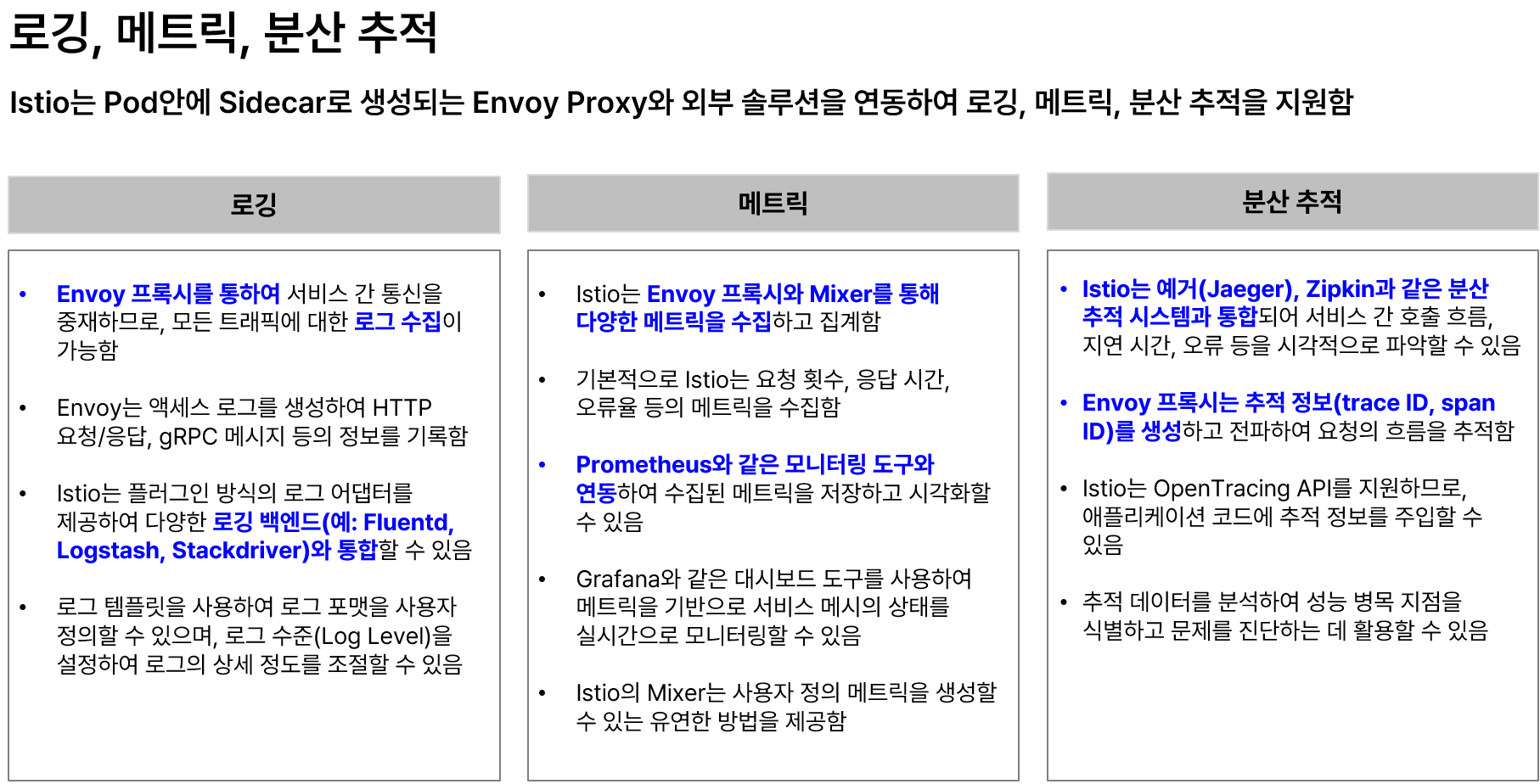


04.Istio DestinationRule Specification.xlsx

<https://docs.google.com/spreadsheets/d/1ml--0_ejjK001-nVPjXF3u4wZP1vePap/edit?usp=drive_link&ouid=110516422607493163049&rtpof=true&sd=true>

04.Istio Virtual Service Specification.xlsx

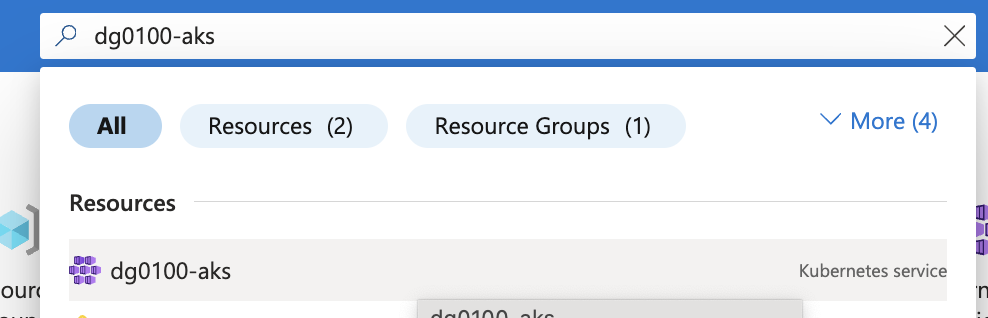
<https://docs.google.com/spreadsheets/d/1nAQJmZmktSc8GNWJpjpDtOEXSbgK-tqA/edit?usp=drive_link&ouid=110516422607493163049&rtpof=true&sd=true>



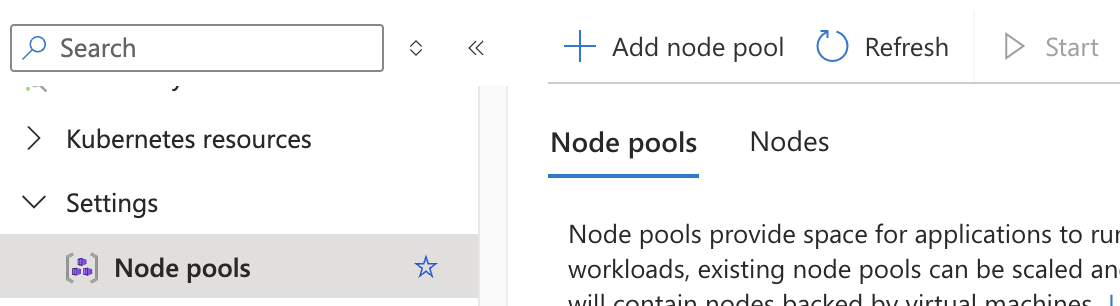
## Istio 설치

1) AKS Node 추가

- AKS 오픈



- Settings > Node pools 선택 후 [Add node pool] 클릭

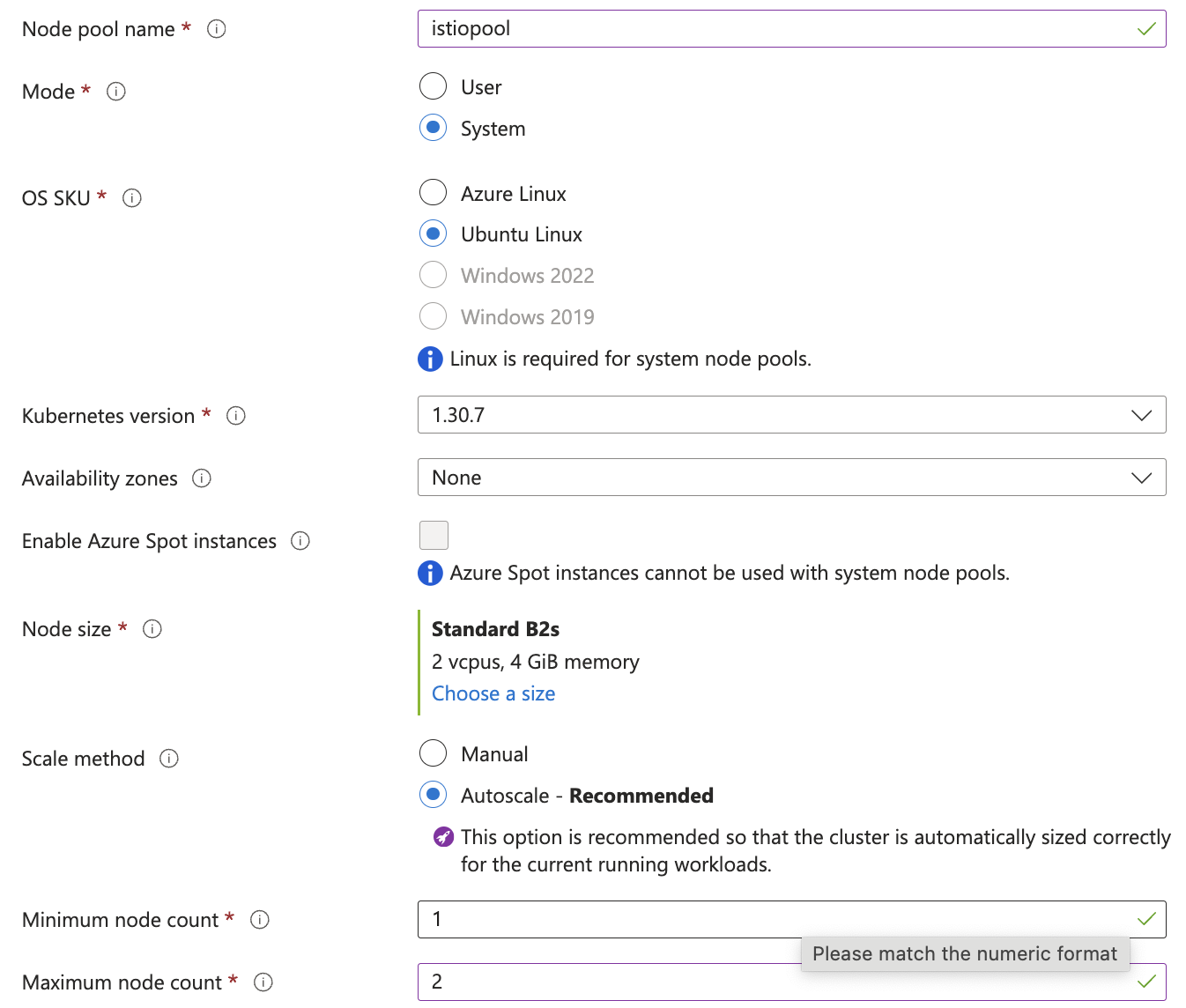


- ‘istiopool’ 추가

- Mode: 시스템 Pod가 배포되므로 ‘System’으로 선택

- Node size: Standard B2s

- Maximum node count: 2



하단의 [Review+Create] 버튼 클릭

그리고 [Create]클릭

2) istioctl 설치

- istioctl 다운로드를 위한 디렉토리 생성:

| mkdir -p ~/install && cd ~/install |
| --- |

- 최신 stable 버전의 Istio 다운로드:

| curl -L https://istio.io/downloadIstio | sh - |
| --- |

- 다운로드된 Istio 디렉토리로 이동 (버전은 다운로드된 버전으로 변경):

| cd {istio 디렉토리}  예) cd istio-1.24.2 |
| --- |

- istioctl을 시스템 PATH에 추가:

| sudo cp bin/istioctl /usr/local/bin/ |
| --- |

- 설치 확인:

| istioctl version |
| --- |

3) Istio 설치

- istiopool에 생성된 Node의 Lable 확인

| ondal@ihaegyeong-ui-MacBookAir gateway % k get nodes --show-labels NAME STATUS ROLES AGE VERSION LABELS aks-istiopool-26390779-vmss000000 Ready <none> 113s v1.30.7 **agentpool=istiopool**,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=Standard\_B2s,**beta.kubernetes.io/os=linux**,... |
| --- |

- 설치: 1 Node에 최소 설치

| istioctl install --set profile=minimal \  --set values.global.defaultNodeSelector."beta.kubernetes.io/os"=linux \  --set values.global.defaultNodeSelector."agentpool"=istiopool |
| --- |

| Profile 종류  - default 프로덕션 환경을 위한 권장 프로필 istiod, ingress gateway만 설치 적절한 리소스 요청/제한이 설정됨  - demo 학습과 평가를 위한 프로필 default와 비슷하지만 더 많은 기능 활성화 Kiali, Prometheus, Jaeger, Grafana 등 추적/모니터링 도구 포함 리소스 요청/제한이 낮게 설정됨  - minimal 가장 가벼운 프로필 istiod만 설치 최소 기능만 제공  - external 컨트롤 플레인이 외부에 있는 경우 사용 게이트웨이 및 사이드카 프록시만 설치  - preview 실험적 기능 포함 프로덕션에는 권장하지 않음 새로운 기능을 미리 테스트해볼 때 사용 |
| --- |

- 확인

| k get po -n istio-system |
| --- |

## istio 적용

- Istio ingress gateway 설치

| cat > istio-minimal-gateway.yaml << EOF apiVersion: install.istio.io/v1alpha1 kind: IstioOperator spec:  profile: minimal  components:  ingressGateways:  - name: istio-ingressgateway  enabled: true  namespace: istio-system EOF |
| --- |

| istioctl install -f istio-minimal-gateway.yaml -y |
| --- |

외부에서는 istio ingressgateway의 Service L/B IP로 접근할 수 있습니다.

| ondal@ihaegyeong-ui-MacBookAir istio-1.24.2 % k get po -n istio-system NAME READY STATUS RESTARTS AGE istio-ingressgateway-7f57549c9f-g7ccj 1/1 Running 0 4m57s istiod-ddcf4fdd9-vh966 1/1 Running 0 21m ondal@ihaegyeong-ui-MacBookAir istio-1.24.2 %  ondal@ihaegyeong-ui-MacBookAir istio-1.24.2 % **k get svc -n istio-system** NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE istio-ingressgateway LoadBalancer 10.0.177.77 **20.249.197.158** 15021:32630/TCP,80:30601/TCP,443:31905/TCP 5m9s istiod ClusterIP 10.0.178.128 <none> 15010/TCP,15012/TCP,443/TCP,15014/TCP 21m |
| --- |

- Istio node에 taint 설정

Istio node에 다른 Pod가 생성 안되도록 taint 설정을 합니다.

| k taint nodes -l agentpool=istiopool dedicated=istio:NoSchedule |
| --- |

Taint 설정 확인

| k get nodes -o custom-columns=NAME:.metadata.name,TAINTS:.spec.taints  NAME TAINTS aks-istiopool-21409540-vmss000000 [map[effect:NoSchedule key:dedicated value:istio]] aks-nodepool1-17263721-vmss000002 <none> aks-nodepool1-17263721-vmss000003 <none> |
| --- |

Istio 설치가 끝났습니다.

‘Routing 패턴’실습 중이었다면 **더 이상 진행하지 말고** 가이드 문서로 돌아가세요.

## 라우팅 예시

istio를 적용할 namespace의 label에 'istio-injection=enabled' 추가

| k label namespace {적용할 namespace명} istio-injection=enabled |
| --- |

작업할 namespace 로 이동하여 오브젝트를 생성합니다.

| # Gateway  apiVersion: networking.istio.io/v1alpha3 kind: Gateway metadata:  name: usage-gateway  namespace: istio-system #gateway는 istio-system에 생성 spec:  selector:  istio: ingressgateway  servers:  - port:  number: 80  name: http  protocol: HTTP  hosts:  - "*"  ---  # VirtualService apiVersion: networking.istio.io/v1alpha3 kind: VirtualService metadata:  name: usage-routing spec:  hosts:  - "*\*"  gateways:  - istio-system/usage-gateway #istio-system의 gateway 사용   http:  - match:  - uri:  prefix: "/api/manage"  route:  - destination:  host: usage-management-service  port:  number: 80  - match:  - uri:  prefix: "/api/usage"  route:  - destination:  host: usage-query-service  port:  number: 80  - match:  - uri:  prefix: "/api/monitor"  route:  - destination:  host: usage-update-service  port:  number: 80 |
| --- |

## Circuit Breaker 예시

Destination Rule을 이용

기본적인 Circuit Breaker는 구현 가능하나 fallback처리(C/B 오픈 시 처리)는 못합니다.

| apiVersion: networking.istio.io/v1alpha3 kind: DestinationRule metadata:  name: usage-query-circuit-breaker spec:  host: usage-query-service  trafficPolicy:  connectionPool:  tcp:  # 허용되는 동시 TCP 연결의 최대 수  # 이 값을 초과하면 추가 연결은 큐잉되거나 실패 처리됨  maxConnections: 100  # TCP 연결 설정 시도의 제한 시간  # 이 시간을 초과하면 연결 시도가 실패로 처리됨  connectTimeout: 3s  http:  # 백엔드로 전달되지 못하고 대기 중인 HTTP 요청의 최대 수  # 이 값을 초과하면 추가 요청은 429(Too Many Requests) 오류로 응답됨  http1MaxPendingRequests: 10  # 단일 HTTP/1.1 연결을 통해 처리될 수 있는 최대 요청 수  # 이 값에 도달하면 연결이 닫히고 새로운 연결이 생성됨  maxRequestsPerConnection: 10  # 요청 실패 시 최대 재시도 횟수  # 이 횟수를 초과하면 요청은 최종적으로 실패로 처리됨  maxRetries: 3  outlierDetection:  # 연속된 5xx 오류(500-599)가 이 횟수를 초과하면 해당 Pod를 로드밸런서 풀에서 제외  consecutive5xxErrors: 5   # Envoy가 모든 업스트림 호스트(Pod)의 상태를 검사하는 시간 간격  # 이 간격으로 연속 오류 수를 확인하며, 임계값 초과 시 해당 호스트(Pod)를 로드밸런서 풀에서 제외함  interval: 10s   # 서비스 인스턴스가 제외된 후 다시 포함될 때까지의 기본 시간  # 반복적인 실패가 발생하면 이 시간은 점진적으로 증가될 수 있음  baseEjectionTime: 30s   # 한 번에 Circuit Breaker에 의해 제외될 수 있는 서비스 인스턴스(Pod)의 최대 비율  # 전체 서비스의 중단을 방지하기 위해 일부 인스턴스만 제외하도록 함   maxEjectionPercent: 50 |
| --- |

## BulkHead 예시

고객의 유형에 따라 다른 Pod로 연결하면서 다른 설정이 적용되게 함

아래는 리소스 크기가 다른 Pod를 배포하는 예시입니다.

Pod의 tier 라벨과 리소스 크기를 다르게 배포합니다.

| # VIP 고객용 Deployment apiVersion: apps/v1 kind: Deployment metadata:  name: usage-query-service-vip spec:  replicas: 2  selector:  matchLabels:  app: usage-query-service  tier: vip  template:  metadata:  labels:  app: usage-query-service  tier: vip  spec:  containers:  - name: usage-query-service  image: unicorncr.azurecr.io/telecom/usage-query-service:latest  resources:  requests:  cpu: "500m"  memory: "1Gi"  limits:  cpu: "1"  memory: "2Gi"  ---  # 일반 고객용 Deployment apiVersion: apps/v1 kind: Deployment metadata:  name: usage-query-service-regular spec:  replicas: 3  selector:  matchLabels:  app: usage-query-service  tier: regular  template:  metadata:  labels:  app: usage-query-service  tier: regular  spec:  containers:  - name: usage-query-service  image: unicorncr.azurecr.io/telecom/usage-query-service:latest  resources:  requests:  cpu: "250m"  memory: "512Mi"  limits:  cpu: "500m"  memory: "1Gi" |
| --- |

Virtual Service와 DestinationRule로 BulkHead 패턴을 적용합니다.

| # VirtualService apiVersion: networking.istio.io/v1alpha3 kind: VirtualService metadata:  name: usage-query-routing spec:  hosts:  - usage-query-service  http:  - match:  - headers:  x-user-type:  exact: "vip"  route:  - destination:  host: usage-query-service  subset: vip  - route:  - destination:  host: usage-query-service  subset: regular  ---   # DestinationRule apiVersion: networking.istio.io/v1alpha3 kind: DestinationRule metadata:  name: usage-query-bulkhead spec:  host: usage-query-service  subsets:  - name: vip  labels:  tier: vip  trafficPolicy:  connectionPool:  tcp:  # VIP용 최대 TCP 연결 수  maxConnections: 100  connectTimeout: 3s  http:  # VIP용 최대 대기 요청 수  http1MaxPendingRequests: 10  maxRequestsPerConnection: 10  maxRetries: 3  outlierDetection:  # 연속된 게이트웨이 오류(502,503,504) 임계값 - VIP  consecutiveGatewayErrors: 10  # 연속된 5xx 오류 임계값 - VIP  consecutive5xxErrors: 10  # 오류 검사 주기  interval: 10s  # 제외 시간  baseEjectionTime: 30s  # 최대 제외 비율  maxEjectionPercent: 50  - name: regular  labels:  tier: regular  trafficPolicy:  connectionPool:  tcp:  # 일반용 최대 TCP 연결 수  maxConnections: 50  connectTimeout: 3s  http:  # 일반용 최대 대기 요청 수  http1MaxPendingRequests: 5  maxRequestsPerConnection: 5  maxRetries: 2  outlierDetection:  # 연속된 게이트웨이 오류(502,503,504) 임계값 - 일반  consecutiveGatewayErrors: 5  # 연속된 5xx 오류 임계값 - 일반  consecutive5xxErrors: 5  # 오류 검사 주기  interval: 10s  # 제외 시간  baseEjectionTime: 30s  # 최대 제외 비율  maxEjectionPercent: 50 |
| --- |

# Istio 삭제

- 먼저 사이드카가 주입된 애플리케이션 삭제

Deployment 와 StatefulSet 등 삭제

| kubectl delete deploy {deployment 객체} namespace {namespace} |
| --- |

- Istio Ingress Gateway 삭제

| kubectl delete gateway --all -n istio-system |
| --- |

- istioctl로 Istio 삭제

| istioctl uninstall --purge -y |
| --- |

- istio-system 네임스페이스 삭제

| kubectl delete namespace istio-system |
| --- |

- 네임스페이스의 istio-injection 라벨 제거 (필요한 경우)

| kubectl label namespace dg0100-gateway istio-injection- |
| --- |

- 삭제 확인

| # istio-system 네임스페이스가 없어야 함 kubectl get ns | grep istio  # Istio 관련 CRD가 없어야 함 kubectl get crd | grep istio |
| --- |