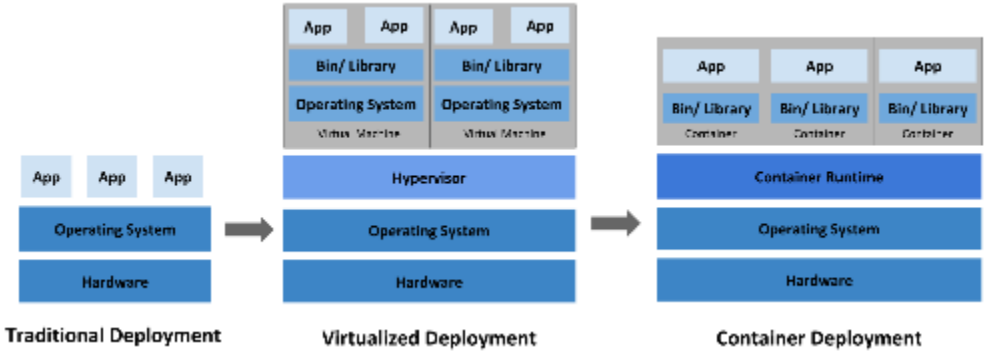
Kubernetes 정리

쿠버네티스(kubernetes:k8s)란 무엇인가?

- **컨테이너화**된 **워크로드**와 서비스를 관리하기 위한 **이식성**이 있고, **확장 가능**한 오픈소스 플랫폼이다. 쿠버네티스는 **선언적 구성과 자동화를 모두 용이**하게 해준다. 쿠버네티스는 **크고, 빠르게 성장하는 생태계**를 가지고 있다.



- 애플리케이션 생성과 배포: VM 이미지를 사용하는 것에 비해 컨테이너 **이미지 생성이 보다 쉽고 효율적**임.
- 지속적인 개발, 통합 및 배포: 안정적으로 컨테이너 이미지를 빌드해서 배포할 수 있고 이미지의 **불변성 덕에 빠르고 효율적으로 롤백**할 수 있다.
- 개발과 운영의 관심사 분리: 배포 시점이 아닌 빌드/릴리스 시점에 애플리케이션 컨테이너 이미지를 만들기 때문에, 애플리케이션이 구성환경에서 의존성이 줄어든다.
- 가시성(observability): OS 수준의 정보와 메트릭에 머무르지 않고, 애플리케이션의 헬스와 그 밖의 시그널을 볼 수 있다.
- 개발, 테스팅 및 운영 환경에 걸친 일관성
- 클라우드 및 OS 배포판 간 이식성: Ubuntu, RHEL, CoreOS, 온-프레미스, 주요 퍼블릭 클라우드와 어디에서든 구동된다.
- 애플리케이션 중심 관리: 가상 하드웨어 상에서 OS를 실행하는 수준에서 논리적인 리소스를 사용하는 OS 상에서 애플리케이션을 실행하는 수준으로 추상화 수준이 높아진다.
- 느슨하게 커플되고, 분산되고, 유연하며, 자유로운 마이크로 서비스: 애플리케이션은 단일 목적의 머신에서 모놀리식 스택으로 구동되지 않고 보다 작고 독립적인 단위로 쪼개져서 동적으로 배포되고 관리될 수 있다.
- 리소스 격리: 애플리케이션 성능을 예측할 수 있다.
- 리소스 사용량: 고효율 고집적.

 # 참고 : 워크로드는 쿠버네티스에서 구동되는 애플리케이션이다.
워크로드가 단일 컴포넌트이거나 함께 작동하는 여러 컴포넌트이든 관계없이,
쿠버네티스에서는 워크로드를 일련의 Pod 집합 내에서 실행한다.
쿠버네티스에서 Pod 는 클러스터에서 실행 중인 컨테이너 집합을 나타낸다.

K8s 기본 구성 아키텍처

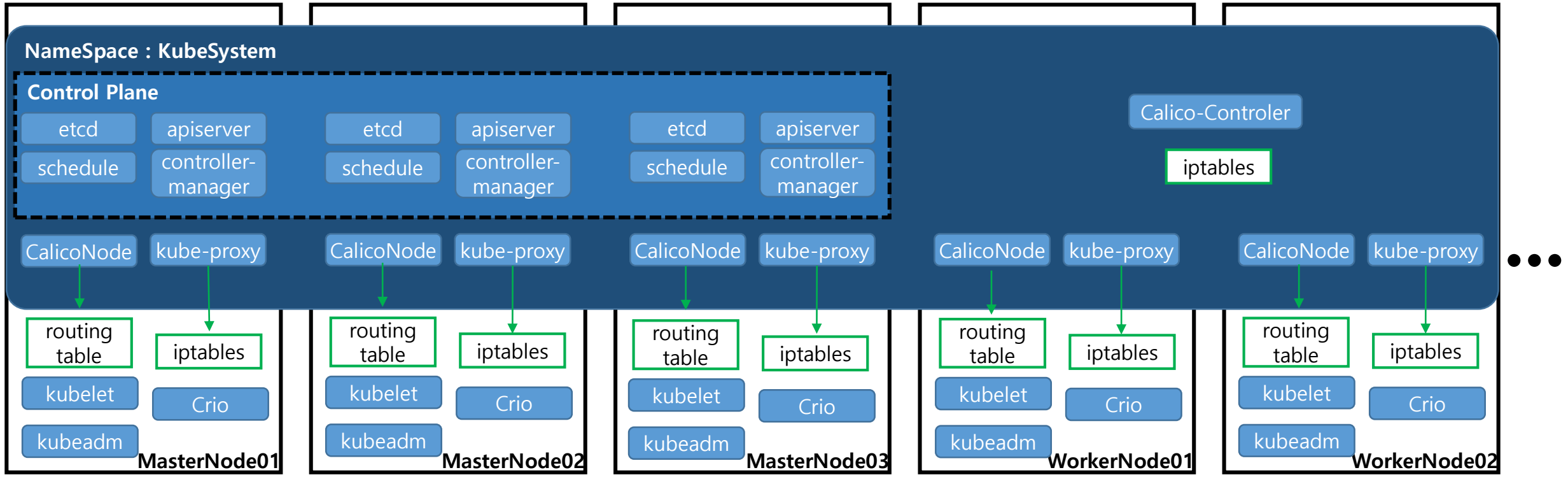
Control Plane

- Etcd : 클러스터 관련 정보들을 담는 저장소 (자세한 내용 https://tech.kakao.com/2021/12/20/kubernetes-etcd/)
- kube-apiserver : k8s 를 조작하고 정보를 제공하는 REST 방식 Server (create, edit, get, delete 등등) (조작방법 https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/)
- kube-scheduler : 노드 선택 스케줄링 해주는 구성요소 (리소스, 정책 등 고려)
- kube-controller-manager : 노드 관리, Job 관리, End Point 관리, 신규 NameSpace의 ServiceAccount 생성 해주는 구성요소

Node

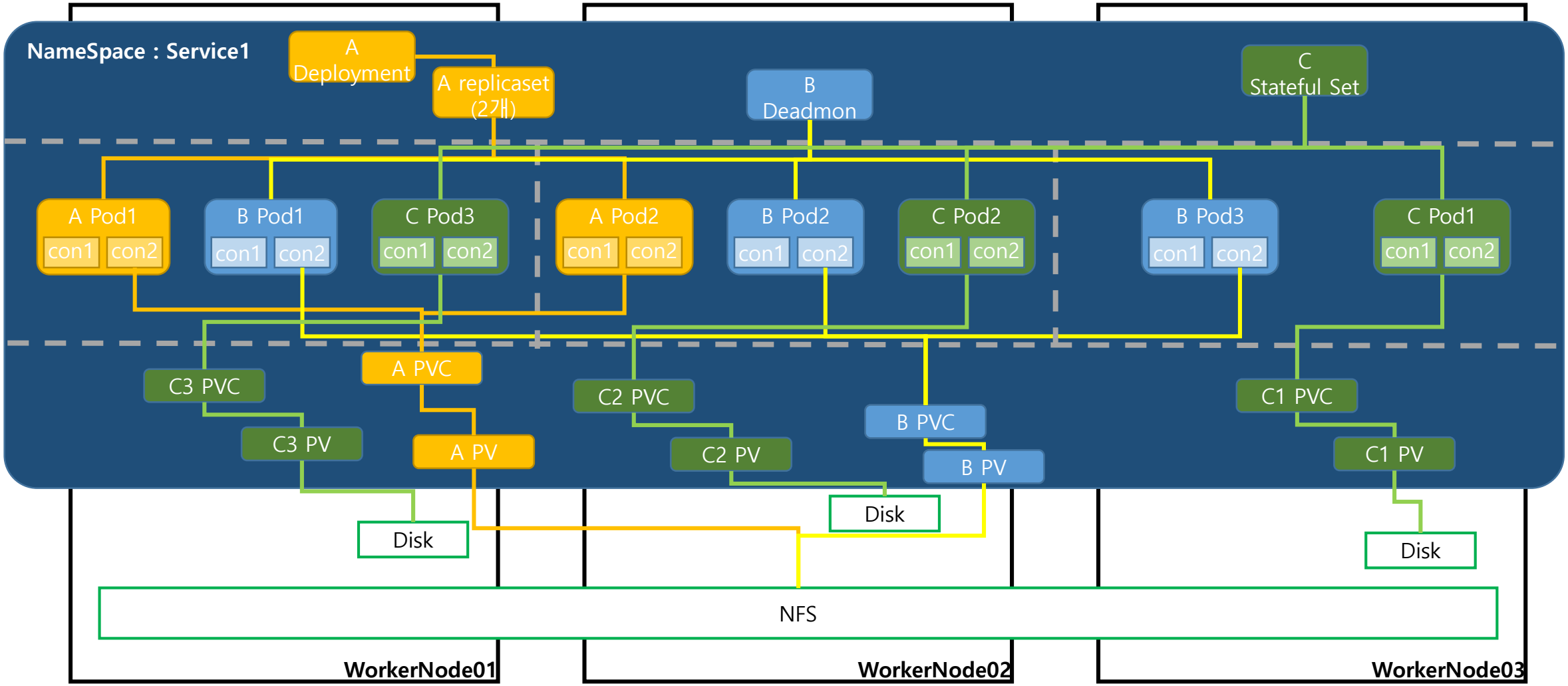
- Crio : Container를 생성하고 수행하는 엔진으로 Container Runtime 이라 함(그 외 containerd, Docker Engine 등이 있음)
- kubelet : 각 노드에서 컨테이너 동작을 관리하는 에이전트 방식의 서비스
- Kubeadm : k8s 클러스터 구축을 위한 도구 (kubeadm init, kubeadm join 등 지원)
- Kube-proxy : kube-proxy는 서비스의 IP를 Pod IP로 변환해주는 역할
- Calico : 파드의 중복되지 않은 IP 부여, Pod 간의 통신을 위한 Routing Table 관리 해주는 역할

참고(kube-proxy 와 Calico(CNI)의 차이) : https://computing-jhson.tistory.com/110 , https://stackoverflow.com/questions/53534553/kubernetes-cni-vs-kube-proxy)



K8s App Pod 구성 아키텍처

Deployment : 컨테이너 템플릿을 작성하여 복제 수량만큼 복제하여 배포할수 있는 컨트롤러
DaemonSet : 모든 노드에서 수행되어야 하는 Container 구성에 사용되는 컨트롤러
Stateful Set : 컨테이너 애플리케이션의 상태를 관리하는 데 사용하는 컨트롤러, 파드의 상태에 따라 동작하며, 순서 및 고유성이 보장된다. (ex. Active / Stanby 구조의 App 구성) https://nearhome.tistory.com/107

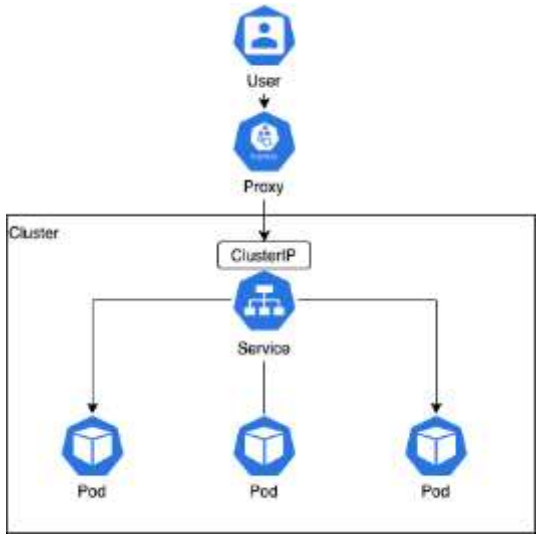


K8s App 서비스 구성 아키텍처

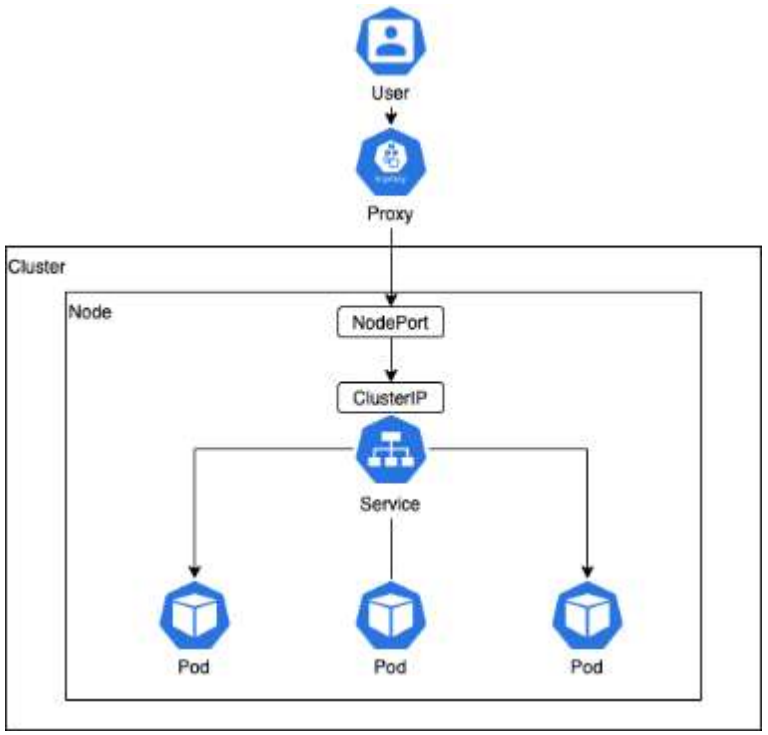
참고 : https://kim-dragon.tistory.com/52

ClusterIP : 클러스터 내부에서만 접근할 수 있는 IP를 할당
NodePort : 노드의 특정 포트를 사용하여 접근하는 방식, 포트당 하나의 서비스만 사용 가능
LoadBalancer : 노드포트 앞단에 특정 LoadBalancer를 사용하여 접근하는 방식
ExternalName : DNS이름에 대한 서비스를 매핑함

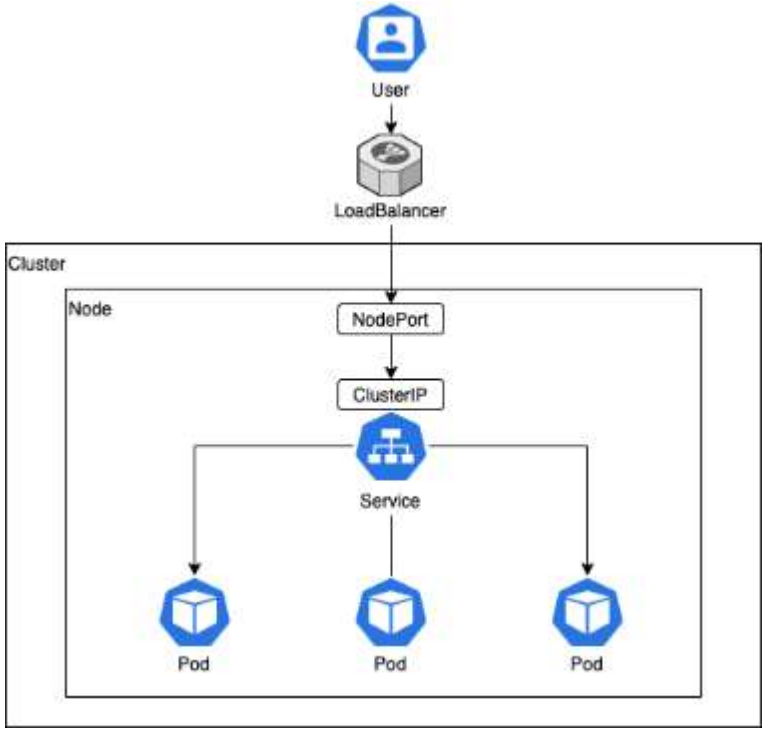
ClusterIP 방식 Service



NodePort 방식 Service



LoadBalancer 방식 Service



K8s 서비스 흐름도

참고 : https://coffeewhale.com/packet-network2

 # CalicoNode 구성

BIRD : Bird는 오픈소스 라우팅 데몬 프로그램이다. calico pod 안 에서 프로세스로 동작하며 각 노드의 pod 정보를 BIRD가 전파하고, 전파 받는다.

Felix : 각 노드에 할당된 pod의 IP 대역이 Brid로 전달되면 그 대역과 정상적으로 통신이 이루어질 수 있도록 iptables와 라우팅테이블 등을 조정한다.

 # tunnel (tunneling)

터널링은 해당 네트워크에서 지원하지 않는 프로토콜을 사용하여 네트워크를 통해 데이터를 전송하는 방법

출발 노드 Tunnel를 통해서 가상 출발, 목적 IP를 담아서 목적 노드에 도착하면 Tunnel 에서 물리 목적 IP를 벗겨내고 가상 목적 IP로 접근 하는 방식이다.