СЛЁРМ

+

Southbridge

Публикация сервисов и приложений

slurm.io

Способы публикации

```
-A KUBE-MARK-MASQ -j MARK --set-xmark 0x4000/0x4000
-A KUBE-NODEPORTS -p tcp -m comment --comment "student000/np2:http"
-m tcp --dport 30029 -j KUBE-MARK-MASQ
```

Service: L3 OSI, NAT, kube-proxy

```
-A KUBE-NODEPORTS -p tcp -m comment --comment "student000/np2:http"
-m tcp --dport 30029 -j KUBE-SVC-2F3FOG2AWAH5Y5PC
```

```
server {
  server_name slurm.io
  ssl on;
```

Ingress: L7 OSI, HTTP и HTTPS, nginx, envoy, traefik, haproxy

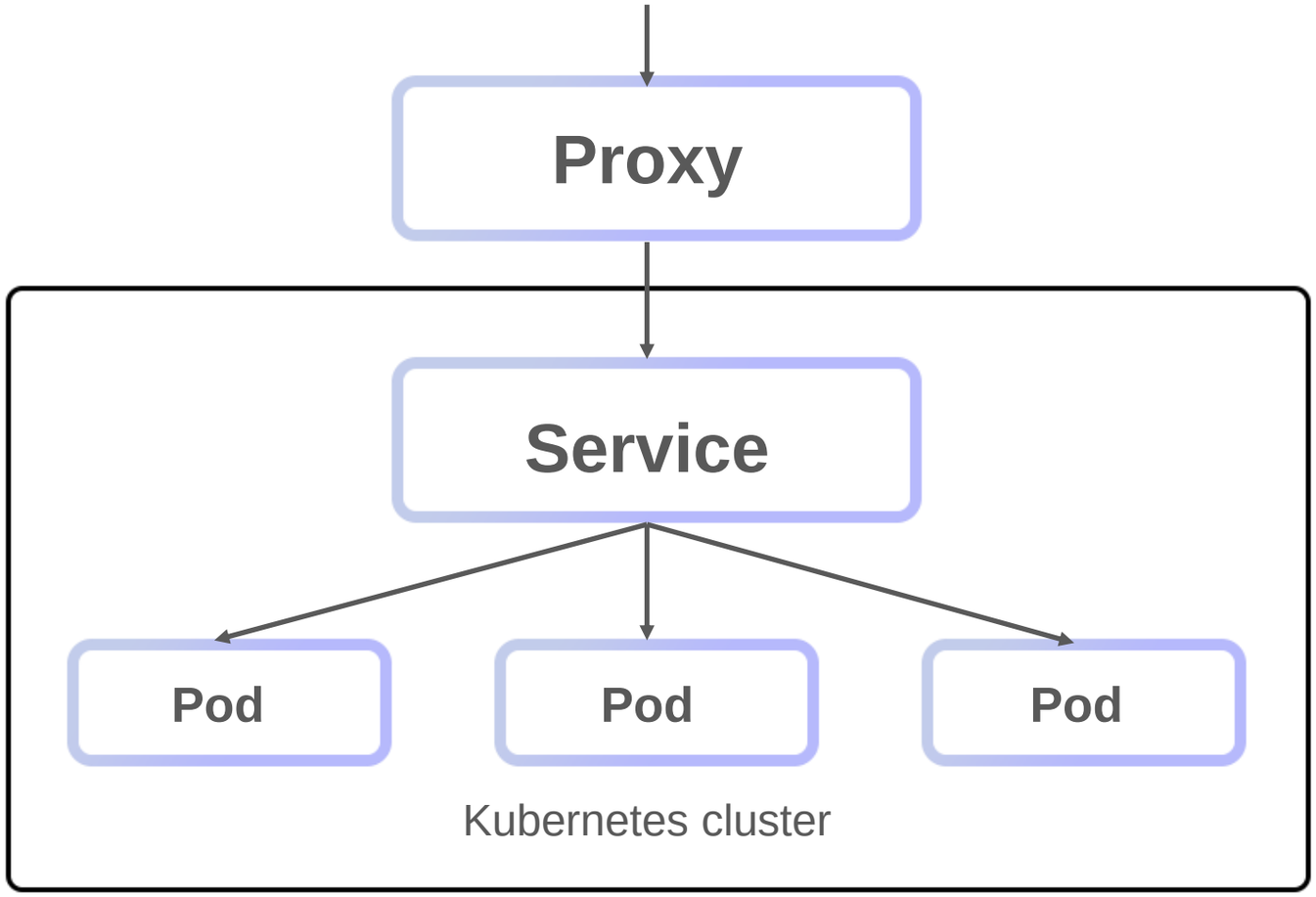
```
  location {
     proxy_pass http://backend;
  }
}
```

Kubernetes Service

- Cluster IP
- NodePort
- LoadBalancer
- ExternalName
- ExternalIPs

ClusterIP

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  type: ClusterIP
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
```

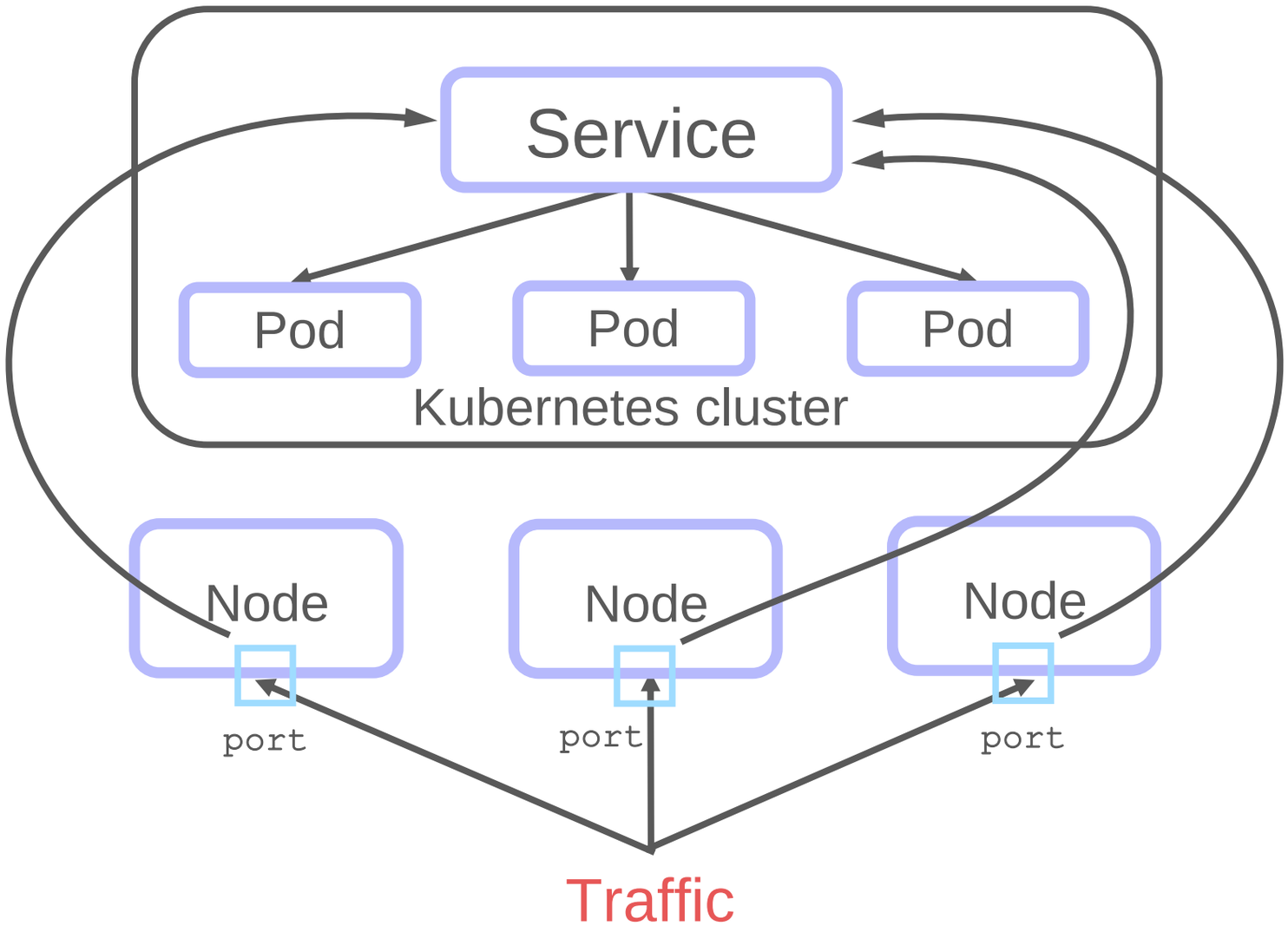



```
kubectl proxy --port=8080
http://localhost:8080/api/v1/proxy/
namespacesdefault/services/my-service:http/

kubectl port-forward service/
my-service 10000:80
```

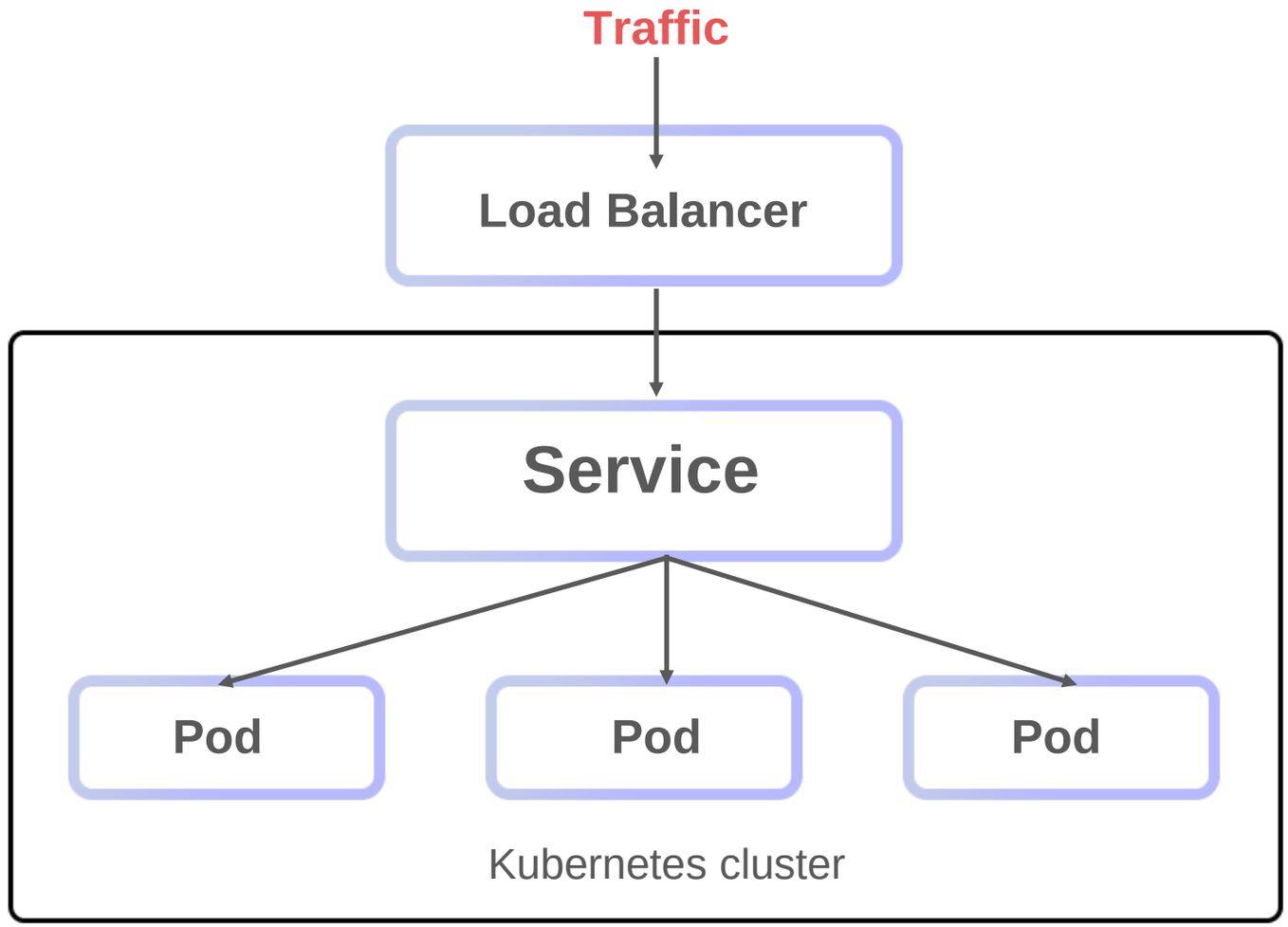
NodePort

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-np
spec:
  selector:
    app: my-app
  type: NodePort
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
```



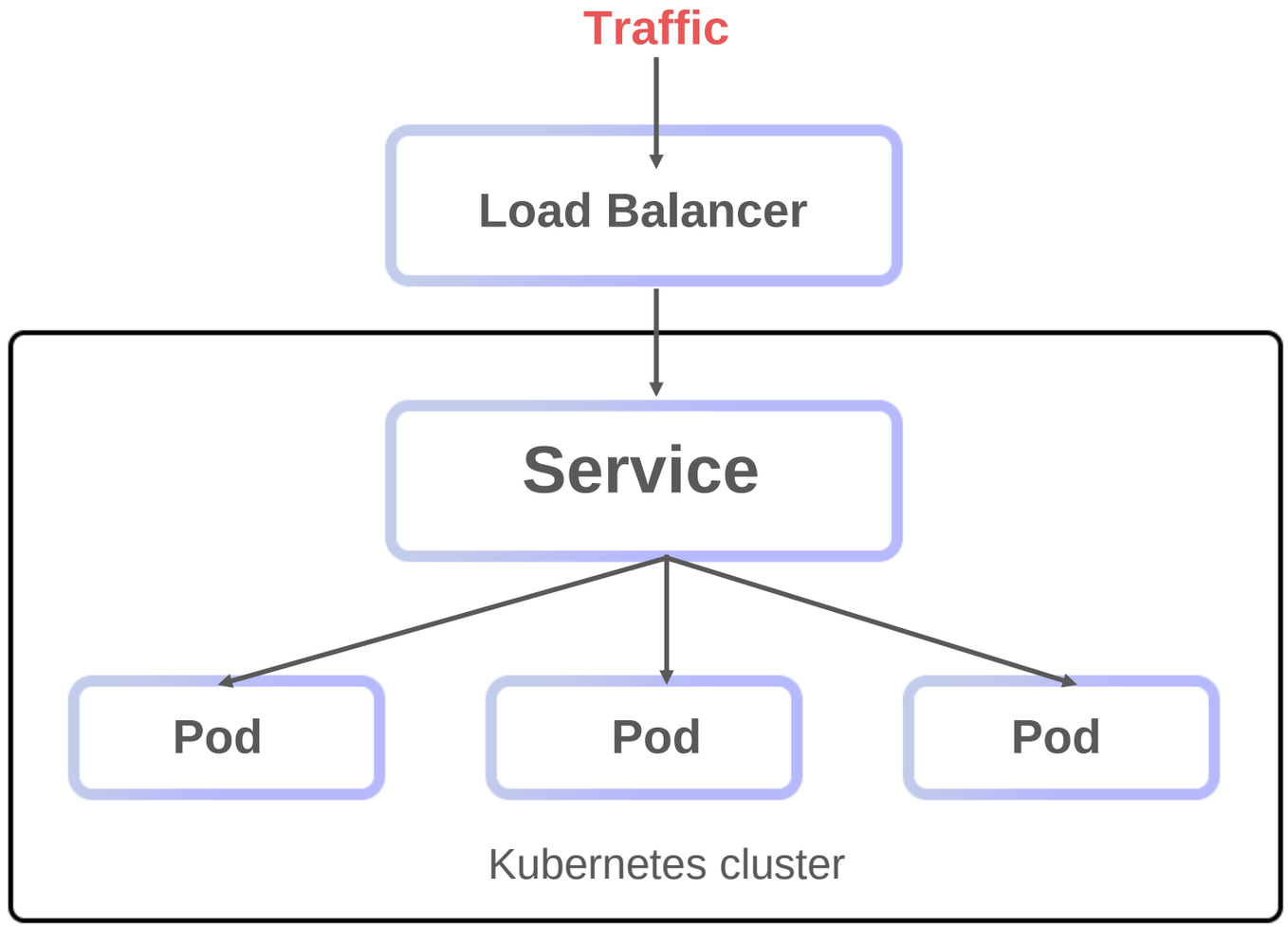
LoadBalancer

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-lb
spec:
  selector:
    app: my-app
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
```



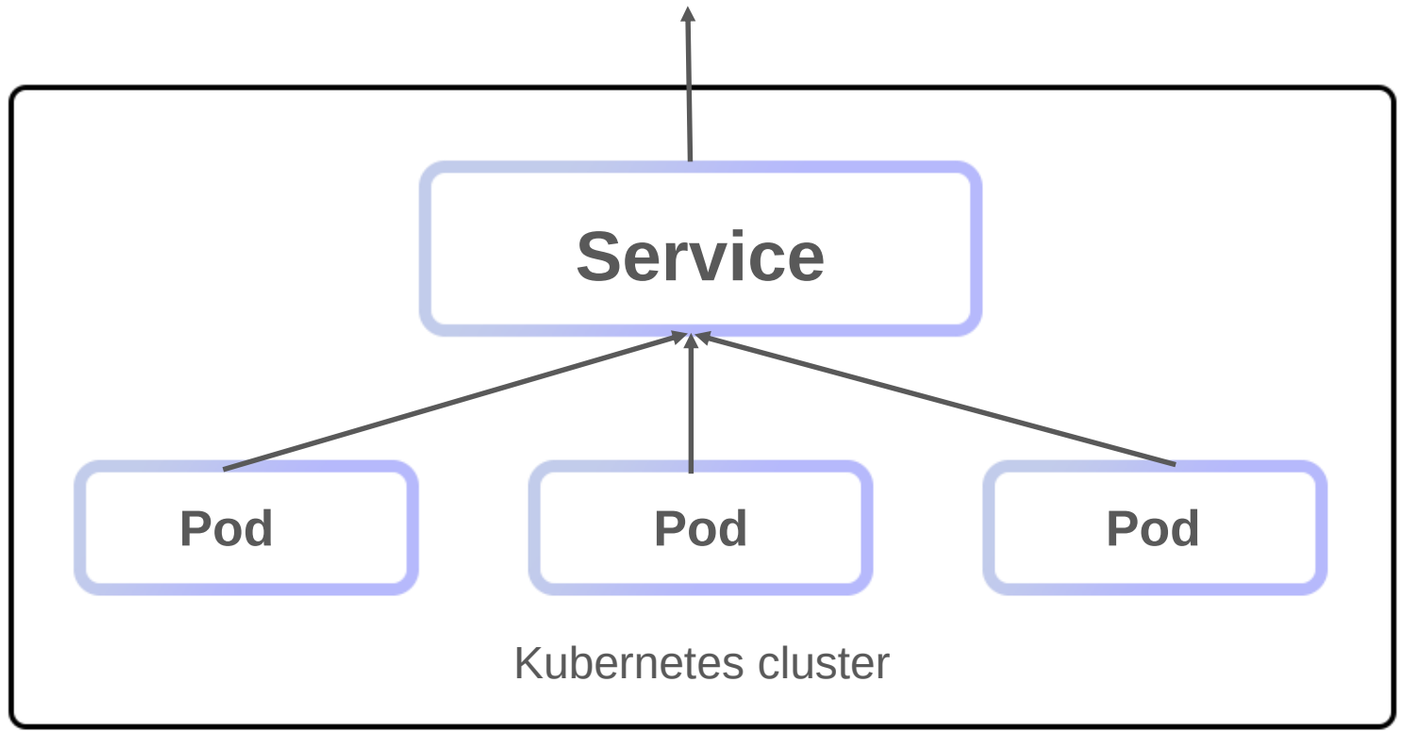
LoadBalancer static IP

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-lb
spec:
  selector:
    app: my-app
  type: LoadBalancer
  loadBalancerIP:
   "1.1.1.1"
  ports:
  - port: 80
    targetPort: 80
```



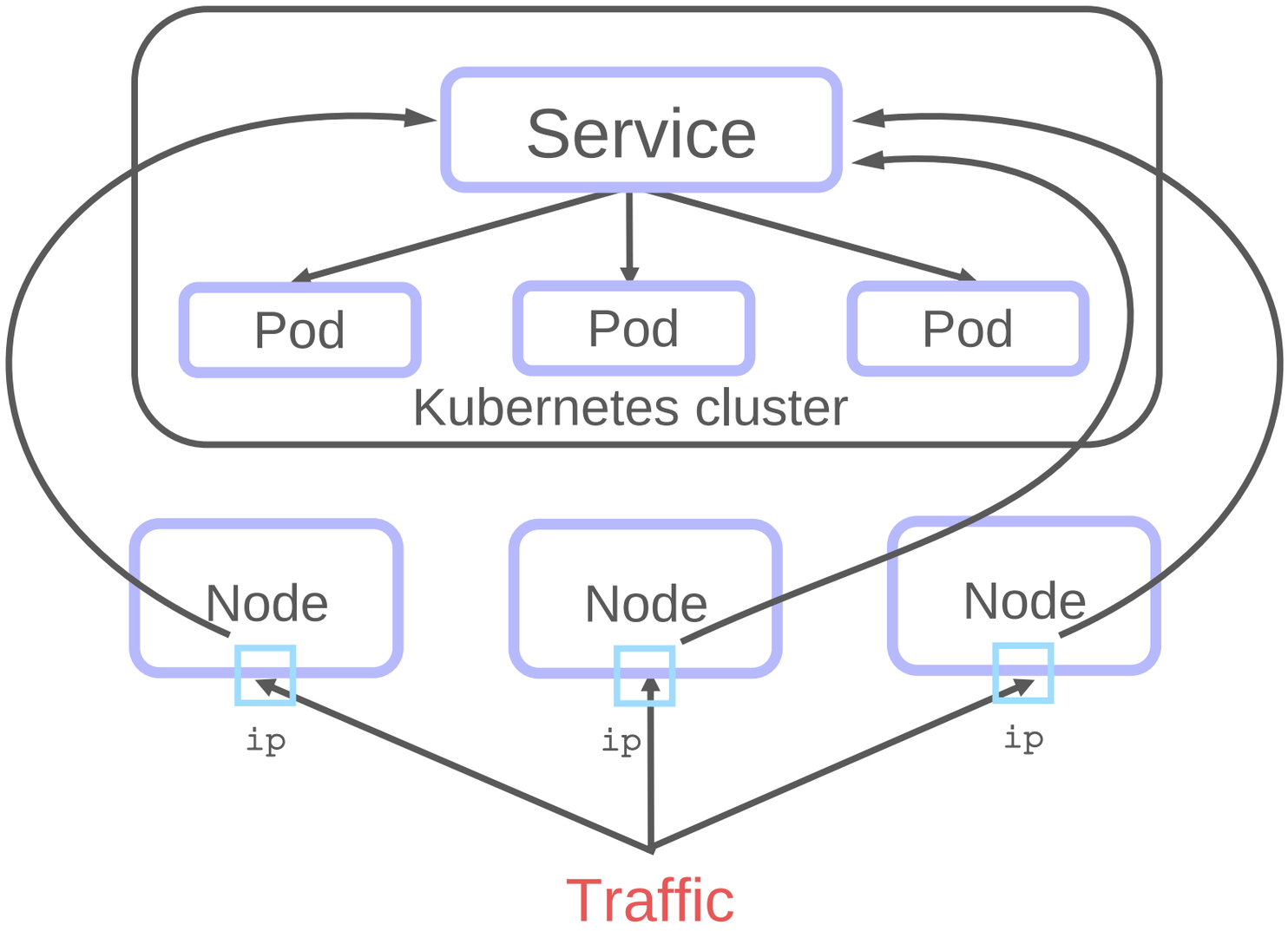
ExternalName

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  type: ExternalName
  externalName:
   my.database.
   example.com
```

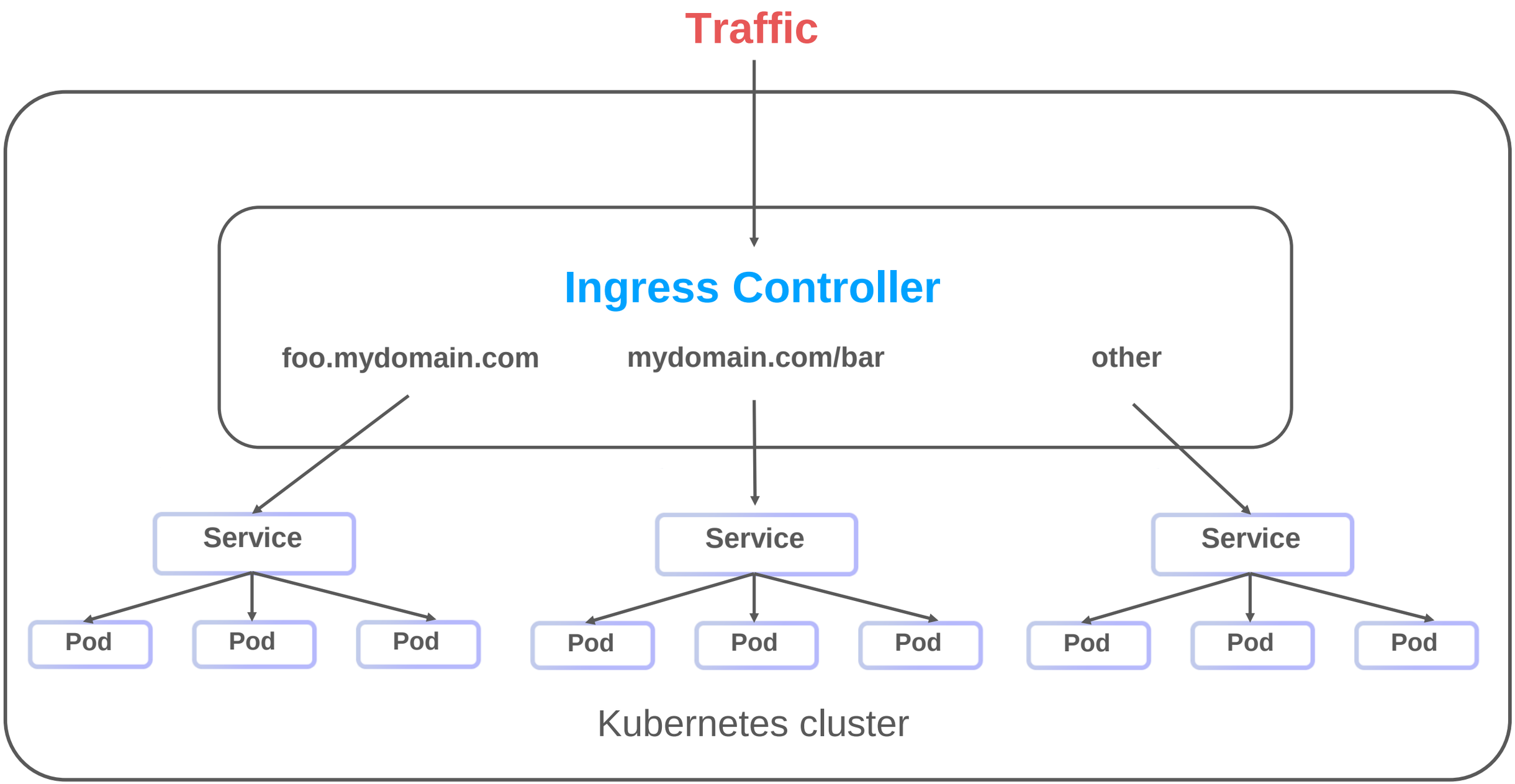


ExternalIPs

```
apiVersion: v1
kind: Service
metadata:
  name: myservice
spec:
  selector:
    app: my-app
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
  externalIPs:
  - 80.11.12.10
```



Ingress


Traffic
Ingress Controller
foo.mydomain.com
mydomain.com/bar
other
Service
Service
Service
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Kubernetes cluster

Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  rules:
  - host: foo.mydomain.com
    http:
      paths:
      - backend:
          serviceName: foo
          servicePort: 8080
  - host: mydomain.com
    http:
      paths:
      - path: /bar/
        backend:
          serviceName: bar
          servicePort: 8080
```

HOST: foo.mydomain.com

HOST: mydomain.com
URL: /bar/

Шифрование SSL/TLS

Указываем сертификат в Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: tls-ingress
spec:
  tls:
  - hosts:
    - sslfoo.com
    secretName: secret-tls
```

Создаем секрет с сертификатом

```
apiVersion: v1
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
kind: Secret
metadata:
  name: secret-tls
  namespace: default
type: kubernetes.io/tls
```

```
kubectl create secret tls ${CERT_NAME} --key ${KEY_FILE} --cert ${CERT_FILE}
```

Cert-manager

- Начинался как способ получить сертификат от LetsEncrypt
- Автоматизирует получение SSL/TLS-сертификатов от различных удостоверяющих центров (LetsEncrypt, selfhosted, selfsigned)
- Интегрируется с ингресс-контроллером
- Автоматизирует продление сертификатов
- CRD: Issuer, ClusterIssuer, Certificate
- RBAC: certmanager.k8s.io

Cert-manager

```
kubectl apply -f https://raw.githubusercontent.com/jetstack/cert-manager/release-0.7/deploy/manifests/00-crds.yaml

kubectl create namespace cert-manager

helm repo add jetstack https://charts.jetstack.io
helm repo update

helm install \
  --name cert-manager \
  --namespace cert-manager \
  --version v0.7.2 \
  --set ingressShim.defaultIssuerName=letsencrypt \
  --set ingressShim.defaultIssuerKind=ClusterIssuer \
  jetstack/cert-manager
```

Cert-manager

```
apiVersion: certmanager.k8s.io/v1alpha1
kind: ClusterIssuer
metadata:
  name: letsencrypt
  namespace: kube-system
spec:
  acme:
# The ACME server URL
server: https://acme-v02.api.letsencrypt.org/directory
# Email address used for ACME registration
email: letsencrypt@slurm.io
# Name of a secret used to store the ACME account private key
privateKeySecretRef:
  name: letsencrypt
# Enable the HTTP-01 challenge provider
http01: {}
```

Подключаем в Ingress

```
apiVersion:
certmanager.k8s.io/v1alpha1
kind: Certificate
metadata:
  name: hostname-ru
  namespace: default
spec:
 acme:
    config:
    - domains:
      - hostname.ru
      - www.hostname.ru
      http01:
        ingress: ""
        ingressClass: nginx
  secretName: hostname-ru-tls
  commonName: hostname.ru
  dnsNames:
  - hostname.ru
  - www.hostname.ru
```

```
 issuerRef:
     name: letsencrypt
     kind: ClusterIssuer
```

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: tls-ingress
  annotations:
    kubernetes.io/tls-acme:
"true"
```

ИЛИ

```
  certmanager.k8s.io/cluster-
issuer:  letsencrypt
```

СЛЁРМ

+

Southbridge