TCP, UDP Ingress for Kubernetes
In Kubernetes, an Ingress is an object that allows access to the Kubernetes services from outside the cluster to services within the cluster. However, there are many applications like Databases, Legacy apps, FTP or DNS based apps that require TCP/UDP load balancing. In the case of legacy apps, app owners need to expose specific TCP ports to avoid re-writing dependent apps.

To expose TCP/UDP based apps, the only solution is to use LoadBalancer service type. However, using LoadBalancer service type is a costly proposition as you've to spin up a dedicated cloud load balancer for each service that you want to expose.

Use Citrix Ingress Controller to expose non HTTP applications

Citrix Ingress Controller (CIC) listens to the Kubernetes API server for Ingress resources and configures the Citrix ADC.

TCP Ingress: CIC supports configuration of TCP, UDP and TCP-SSL based apps for load balancing in Citrix ADC by leveraging annotations in Ingress resource.

Example of Ingress resources for TCP load balancing:

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: redis-master-ingress
  annotations:
    kubernetes.io/ingress.class: "guestbook"
    ingress.citrix.com/insecure-service-type: "tcp"
    ingress.citrix.com/insecure-port: "6379"

spec:
  backend:
    serviceName: redis-master-pods
    servicePort: 6379
```
Example of Ingress resources for TCP load balancing:

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: udp-ingress
  annotations:
    ingress.citrix.com/insecure-service-type: "udp"
    ingress.citrix.com/insecure-port: "5084"

spec:
  backend:
    serviceName: frontend
    servicePort: 53
```

**TCP Ingress with TLS:** CIC can configure SSL_TCP load balancing using the ‘ingress.citrix.com/secure-service-type: ssl_tcp’ annotation in Ingress resource.

Example of Ingress resources for TCP load balancing:

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: colddrinks-ingress
  annotations:
    kubernetes.io/ingress.class: "colddrink"
    ingress.citrix.com/insecure-service-type: "ssl_tcp"
    ingress.citrix.com/secure_backend: '{"frontend-colddrinks":"True"}"

spec:
  tls:
  - secretName: "colddrink-secret"
  backend:
    serviceName: frontend-colddrinks
    servicePort: 443
```
Better performance, monitoring of TCP, UDP based apps

Application developers can closely monitor the health of TCP/UDP based apps through rich monitors in Citrix ADC (such as TCP-ECV, UDP-ECV). The ECV (extended content validation) monitors help in checking whether the application is returning expected content or not. Additionally, the application performance can be improved by using persistence methods (such as Source IP). You can use these Citrix ADC features through Smart Annotations in Kubernetes. Here is one such example:

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: mongodb
  annotations:
    NETSCALER_HTTP_PORT: "80"
    NETSCALER_VIP: "192.168.1.1"
    ingress.citrix.com/csvserver: '{"l2conn":"on"}"
    ingress.citrix.com/lbserver: '{"mongodb-svc":{"lbmethod":"SRCIPDESTIPHASH"}}"
    ingress.citrix.com/monitor: '{"mongodb-svc":{"type":"tcp-ecv"}}'
spec:
  rules:
  - host: mongodb.beverages.com
    http:
      paths:
      - path: /
        backend:
          serviceName: mongodb-svc
          servicePort: 80
```