Annotations

- They are key-value pairs used to attach arbitrary non-identifying metadata to objects
- Labels are used to select/identify objects whereas annotations are used to store additional information that can be used for various purposes
    - tooling and automation
    - Debugging and monitoring
    - Metadata storing
- Consider the following deployment yaml

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deploy
spec:
  minReadySeconds: 2
  replicas: 4
  selector:
    matchLabels:
      app: web
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
  template:
    metadata:
      labels:
        app: web
        version: v1.0
    spec:
      containers:
        - name: app
          image: shaikkhajaibrahim/dmtest:1.0
          resources:
            limits:
              cpu: 250m
              memory: 128Mi
          ports:
            - containerPort: 80
              protocol: "TCP"
          readinessProbe:
            initialDelaySeconds: 3
            periodSeconds: 5
            successThreshold: 3
            failureThreshold: 2
            httpGet:
              path: '/'
              port: 80
```

- Refer Here for well known labels, annotations and taints
- Now lets add an annotation which will add the change cause

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deploy
  annotations:
    kubernetes.io/change-cause: 'First Version'
spec:
  minReadySeconds: 2
  replicas: 4
  selector:
    matchLabels:
      app: web
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
  template:
    metadata:
      labels:
        app: web
        version: v2.0
    spec:
      containers:
        - name: app
          image: shaikkhajaibrahim/dmtest:1.0
          ports:
            - containerPort: 80
              protocol: "TCP"
          readinessProbe:
            initialDelaySeconds: 3
            periodSeconds: 5
            successThreshold: 3
            failureThreshold: 2
            httpGet:
              path: '/'
              port: 80
```

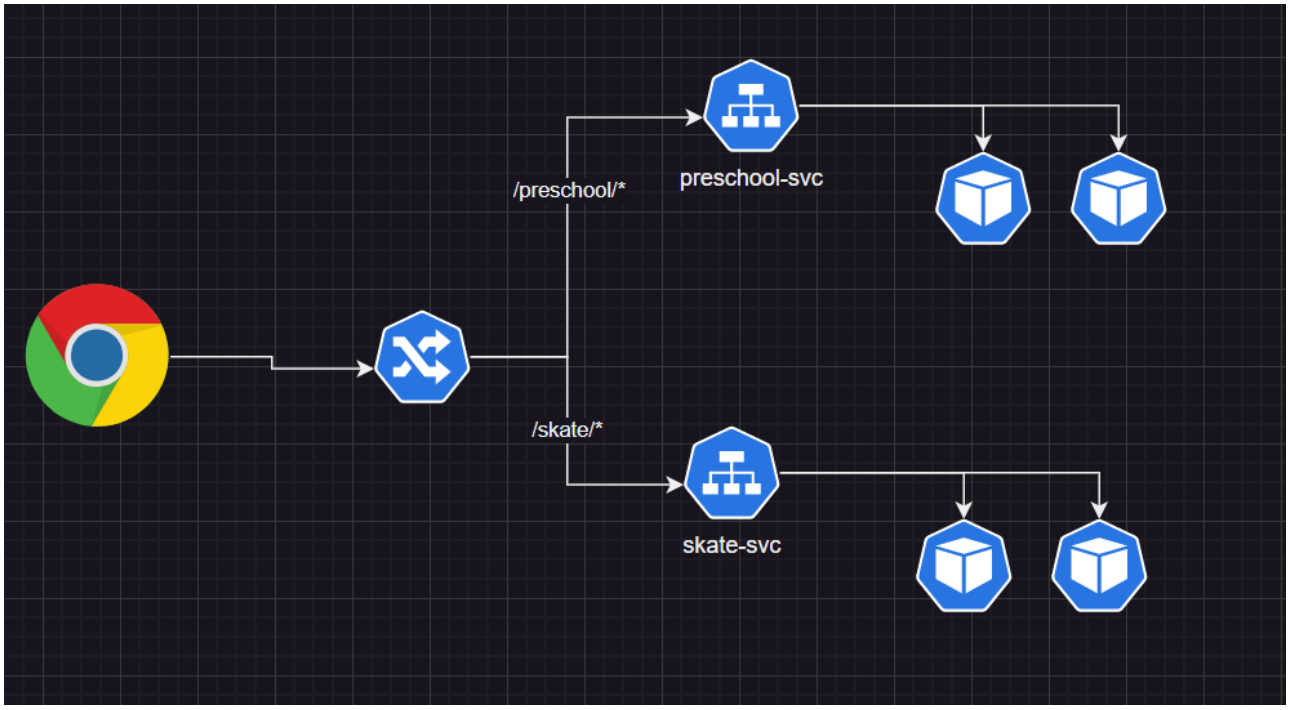
```
controlplane $ cd deploy/
controlplane $ kubectl apply -f deploy.yaml
deployment.apps/app-deploy created
controlplane $ kubectl rollout history deployment app-deploy
deployment.apps/app-deploy
REVISION  CHANGE-CAUSE
1         First Version

controlplane $
```

```
Editor   Tab 1   +
Initialising Kubernetes... done

controlplane $ cd deploy/
controlplane $ kubectl apply -f deploy.yaml
deployment.apps/app-deploy created
controlplane $ kubectl rollout history deployment app-deploy
deployment.apps/app-deploy
REVISION  CHANGE-CAUSE
1         First Version

controlplane $ kubectl apply -f deploy.yaml
deployment.apps/app-deploy configured
controlplane $ kubectl rollout history deployment app-deploy
deployment.apps/app-deploy
REVISION  CHANGE-CAUSE
1         First Version
2         Updated the image to 2.0

controlplane $
```

- Create an internal load balancer in azure/aws with service type loadbalancer
  - aws Refer Here to be very specific Refer Here
  - azure Refer Here

Lets create path based routing

- Scenario:



- images:
  - skate board: `kumar1502/skateboard:1.0`
  - preschool: `panky7700/preschool:1.1`
- Refer Here for deployments and cluster ip svc for preschool and skate

```
Editor   Tab 1   +
controlplane $ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/preschool-deploy-6dbf5fbb8d-n7fh6   1/1     Running   0          41s
pod/preschool-deploy-6dbf5fbb8d-vvmnk   1/1     Running   0          41s
pod/skate-deploy-7db45987b5-xj818       1/1     Running   0          41s

NAME                     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
service/kubernetes       ClusterIP   10.96.0.1        <none>        443/TCP   19d
service/preschool-svc    ClusterIP   10.104.104.181   <none>        80/TCP    41s
service/skate-svc        ClusterIP   10.97.128.3      <none>        80/TCP    41s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/preschool-deploy    2/2     2            2           41s
deployment.apps/skate-deploy        1/1     1            1           41s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/preschool-deploy-6dbf5fbb8d   2         2         2       41s
replicaset.apps/skate-deploy-7db45987b5       1         1         1       41s
controlplane $
```

- Now we need ingress controller, lets use nginx ingress controller Refer Here

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-
nginx/controller-v1.10.1/deploy/static/provider/cloud/deploy.yaml
```

```
kubectl get po --namespace ingress-nginx
```

- nginx ingress annotations Refer Here
- Now we need to write ingress rules Refer Here for the changes.
- Refer Here for using nginx ingress controller in aks
- Refer Here for nginx ingress controller in eks