Govern Application Deployments on Kubernetes with Policy-as-Code using Kyverno

Date - Oct 19, 2024

By - Vinod Kumar Nair
Designation - Principal Engineer | AWS Cloud Expert
Email - vinod827@yahoo.com

Who am I

- Completed B.E. (Information Technology) - 2004-2008
- 16 years of experience in software development across different business domains like Manufacturing, Government sector and Banking (Retail & Investment)
- Currently working as Principal Engineer with Arcesium India Private Limited, Gurgaon
- Certified in AWS and in Kubernetes (CKAD, CKA, CKS, KCNA & KCSA)
- Volunteering work - Tech Blogger, Open Source contributor in CNCF Projects like KEDA and Terraform from HashiCorp and Ambassador of Data on Kubernetes Community

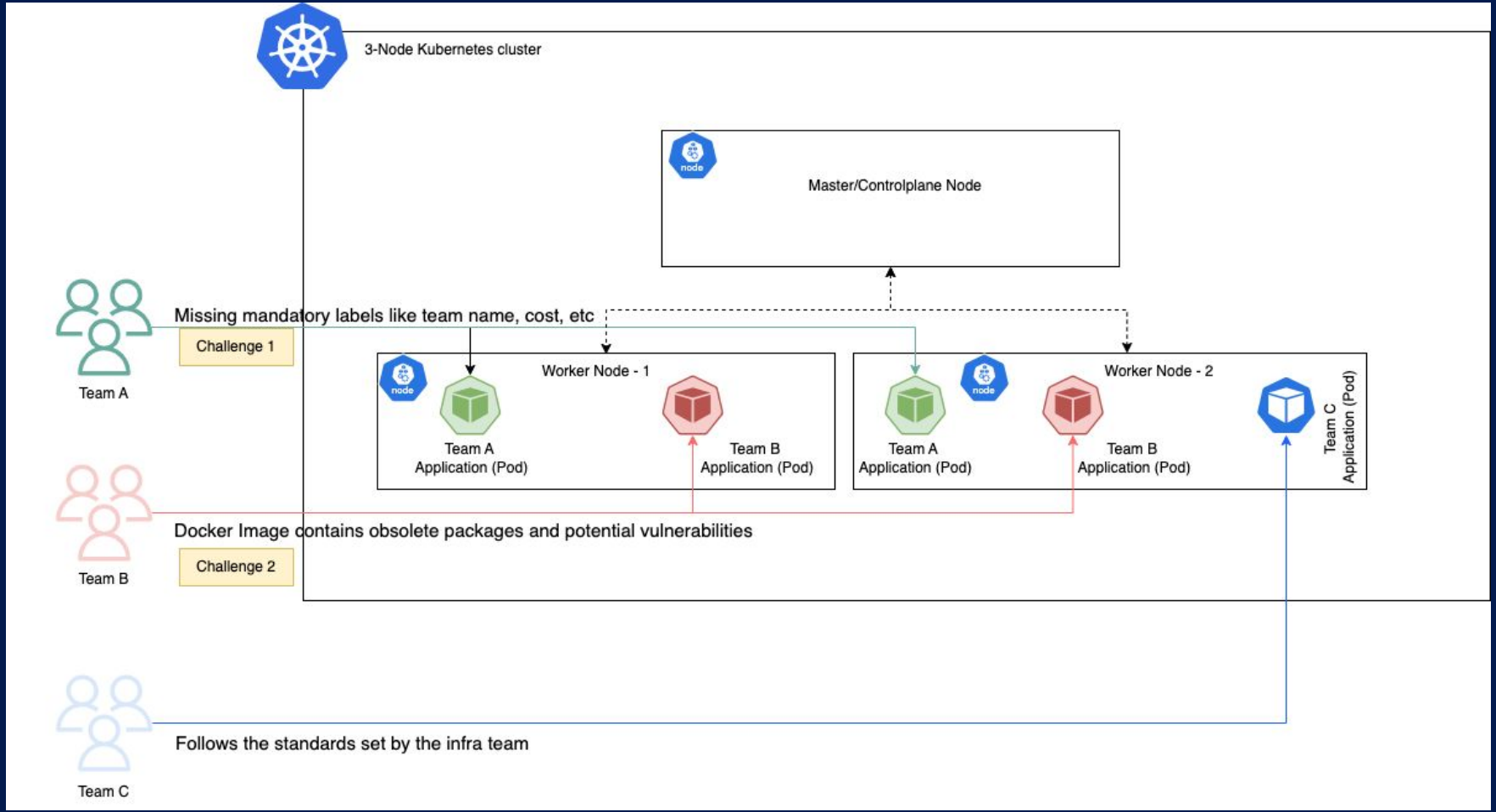
Agenda

- Current challenges in governing application deployments
- Overview of Kyverno
- Tenets of using Kyverno in Kubernetes
- Authorization in Kubernetes
- Kyverno Policies and Rules
- Kyverno demo
- Key Takeaways
- Reference Links & Recommended Learnings
- Q & A

Current challenges in governing application deployments

Problems:-

- Difficult to govern as different teams follows their own deployment approaches
- Potential risk to the production applications due to unknown vulnerabilities & threats
- Difficult to attribute cost at team level due to missing labels



Overview of Kyverno

Kyverno is an open source, Kubernetes-native policy engine designed to validate, mutate, generate and clean up configurations for Kubernetes resources.

Unlike the other popular enforcement engines like OPA (Open Policy Agent) there is no need to learn a different language like Rego to write the policies for Kubernetes.

We can define the policies (as Code) natively just like any other Kubernetes resources like Pods, Deployments, etc in YAML format.
Policies can also be written in JSON format.

```
package kubernetes.admission

import rego.v1

deny contains sprintf("image '%s' comes from untrusted registry", [co
    input.request.kind.kind == "Pod"
    some container in input.request.object.spec.containers
    not startswith(container.image, "hooli.com/")
}
```

Tenets of using Kyverno in Kubernetes

Key tenets:-

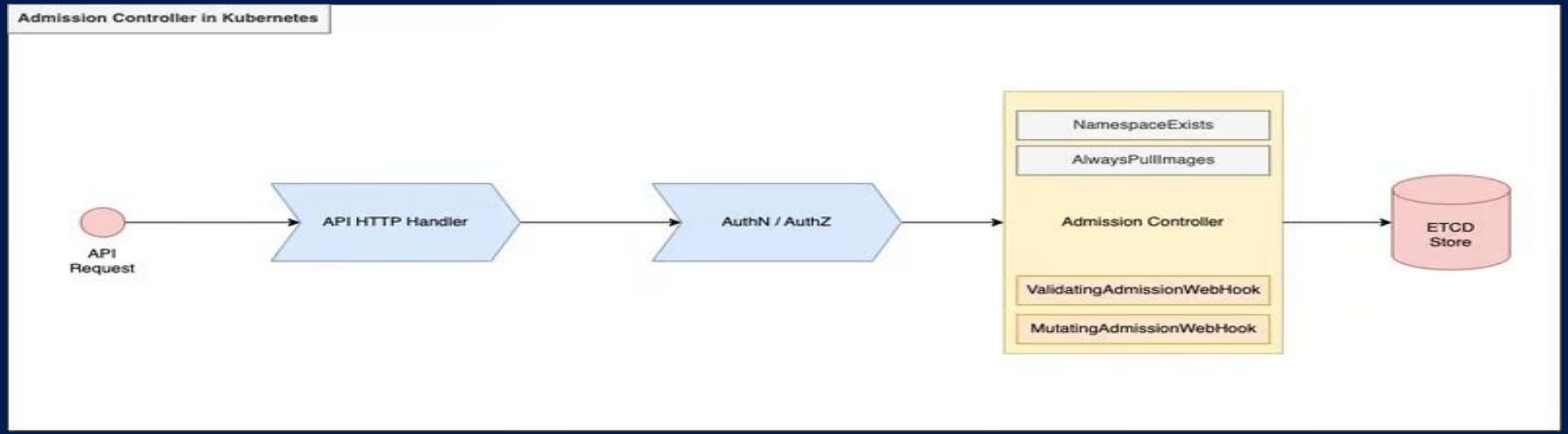
- Kubernetes-native
- Highly customizable using rules to meet the deployment requirements
- Policies can be applied at cluster-wide or namespace level in Kubernetes
- Write the policies in YAML/JSON formats
- Manage the policies like a code on remote source code repositories like GitHub/GitLab

Authorization in Kubernetes

Kubernetes supports various authorization modes like

- AlwaysAllow
- AlwaysDeny
- ABAC (Attribute Based Access Control)
- RBAC (Role Based Access Control)
- Node
- Webhook

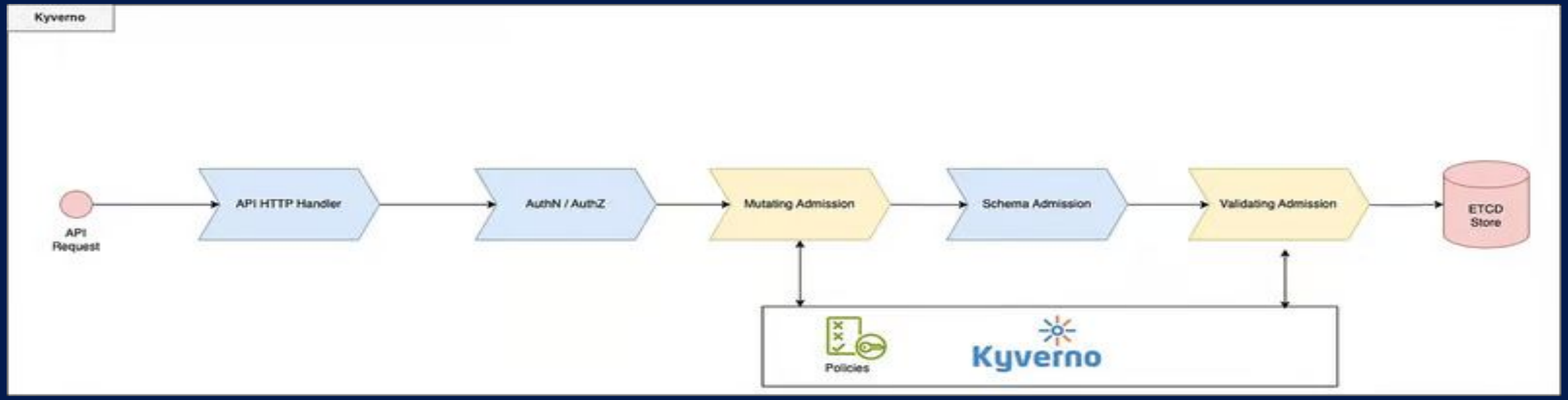
```
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --authorization-mode=AlwaysAllow \\
  --bind-address=0.0.0.0 \\
  --enable-swagger-ui=true \\
  --etcd-cafile=/var/lib/kubernetes/ca.pem \\
  --etcd-certfile=/var/lib/kubernetes/apiserver-etcd-client.crt \\
  --etcd-keyfile=/var/lib/kubernetes/apiserver-etcd-client.key \\
  --etcd-servers=https://127.0.0.1:2379 \\
```



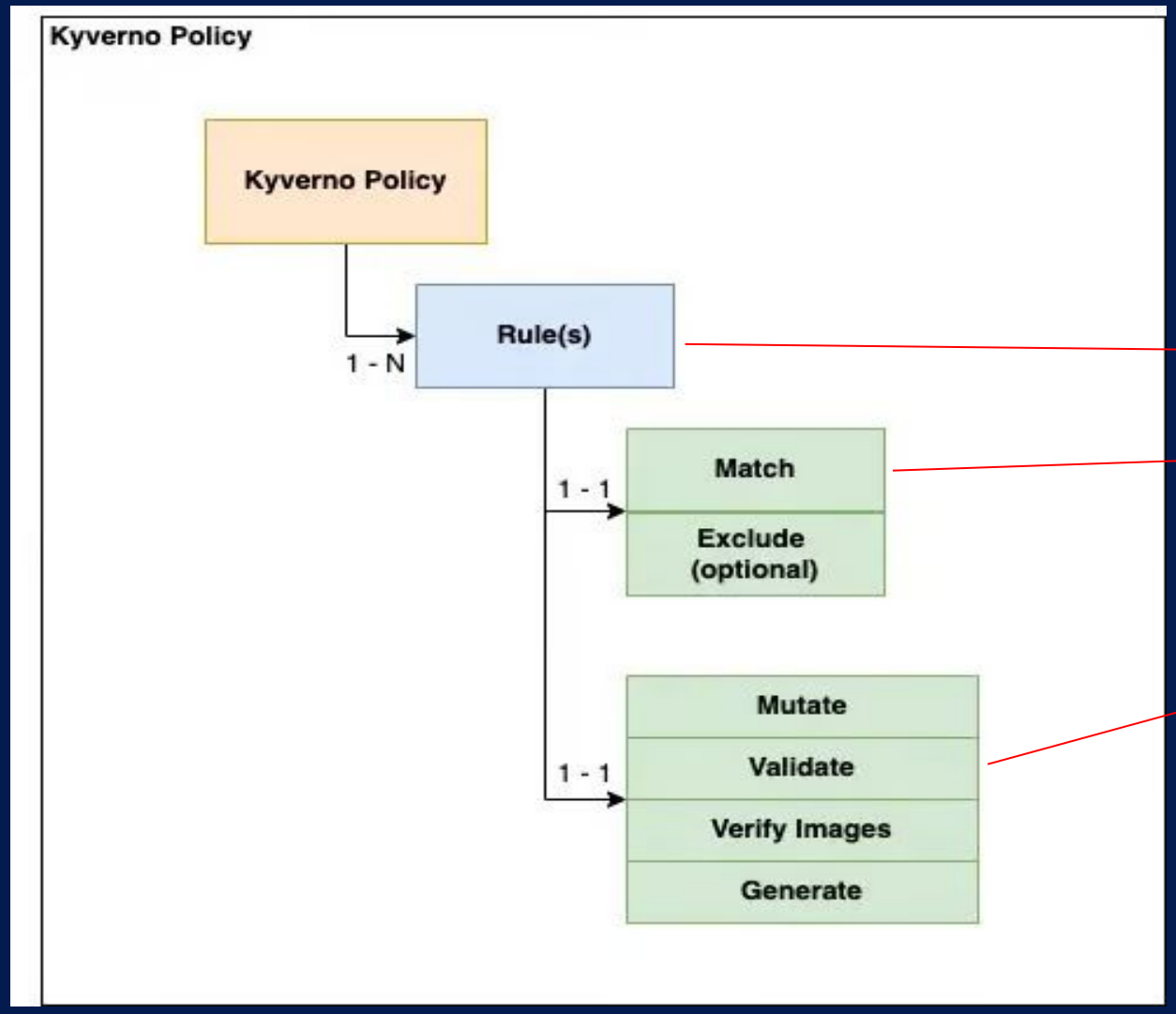
Authorization Mode (Webhook) in Kubernetes

Webhook - A synchronous based HTTP call is made to the remote service (like Kyverno) and the request waits till a response is received.

When Kyverno is used in the cluster then it operates as a dynamic admission controller, enforcing custom policies when the resources (like Pod, Deployment) are created, updated, or deleted. Now during the Mutating & Validating phases of Admission Controller, the incoming request is reviewed by Kyverno as per the rules (or policies) defined.



Kyverno Policies and Rules




```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: enforce-app-deployment-label
spec:
  validationFailureAction: Enforce
  rules:
    - name: check-for-label
      match:
        resources:
          kinds:
            - Deployment
      validate:
        message: "You must have the label, 'app' for all deployments."
        pattern:
          metadata:
            labels:
              app: "?*"
```

Kyverno Policies and Rules (Cont…)

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: enforce-app-deployment-label
spec:
  validationFailureAction: Enforce
  rules:
    - name: check-for-label
      match:
        resources:
          kinds:
            - Deployment
      validate:
        message: "You must have the label, 'app' for all deployments."
        pattern:
          metadata:
            labels:
              app: "?*"
```

- A Kyverno policy is Kubernetes(k8s) native
- It is written in YAML (Policy-as-Code) like any other k8s manifest files objects like Pods, Deployments and can be pushed to source code repositories like GitHub or GitLab
- A policy can be applied both at the cluster level (kind as ClusterPolicy) or at the namespace level (kind as Policy)
- Policy can be applied to enforce or to warn the violations if the application deployments does not match the rules using, validationFailureAction as Enforce or Audit

Kyverno demo

Let's assume a best practices scenario where your company wants every team not to deploy more than 2 replicas of their respective applications within the development namespace and it should be labelled as team_name to attribute the infra cost

```yaml
apiVersion: kyverno.io/v1
kind: Policy
metadata:
  name: enforce-deployment-label-replica-count
  namespace: development
spec:
  validationFailureAction: Enforce
  rules:
    - name: check-for-label
      match:
        resources:
          kinds:
            - Deployment
      validate:
        message: "You must have the label, team_name for all deployments."
        pattern:
          metadata:
            labels:
              team_name: "?*"

    - name: create-max-two
      match:
        any:
        - resources:
            kinds:
            - Deployment
      validate:
        message: The replica count for this Deployment may not exceed 2.
        pattern:
          spec:
            replicas: <= 2
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: my-nginx
  name: my-nginx
  namespace: development
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-nginx
  strategy: {}
  template:
    metadata:
      labels:
        app: my-nginx
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

Kyverno demo (Cont…)

1. Install Kyverno using Helm (recommended), refer to this link here
2. Apply the Kyverno policy on cluster: kubectl apply -f 3-kyverno-policy.yml
3. Verify the policy: kubectl get policy -n development

```
└─ kubectl get policy -n development
NAME                                    ADMISSION   BACKGROUND   VALIDATE ACTION   READY   AGE   MESSAGE
enforce-deployment-label-replica-count  true        true         Enforce           True    10m   Ready
```

4. Deploy the sample application (refer to YAML here) using kubectl on cluster and see its output

```
└─ kubectl create -f 4-sample-app-invalid.yml
Error from server: error when creating "4-sample-app-invalid.yml": admission webhook "validate.kyverno.svc-fail" denied the request:

resource Deployment/development/my-nginx was blocked due to the following policies

enforce-deployment-label-replica-count:
  check-for-label: 'validation error: You must have the label, team_name for all deployments.
    rule check-for-label failed at path /metadata/labels/team_name/'
  create-max-two: 'validation error: The replica count for this Deployment may not
    exceed 2. rule create-max-two failed at path /spec/replicas/'
```

Key Takeaways

- Learned the importance of governing the application deployments on Kubernetes
- Authorization in Kubernetes
- Deep dived into Kyverno, tenets and its authorization flow
- Learned what Kyverno Policy is and how to write one natively on Kubernetes in YAML
- Demo of an application deployment in Kubernetes using Kyverno

Reference Links & Recommended Learnings

https://kyverno.io/

https://kyverno.io/docs/

https://dev.to/vinod827/enforcing-kubernetes-deployments-a-deep-dive-into-policy-as-code-with-kyverno-5ch6

https://github.com/vinod827/k8s-nest/tree/main/iac/demo/kyverno

https://dev.to/vinod827/enforcing-kubernetes-deployments-a-deep-dive-into-policy-as-code-with-kyverno-5ch6

Thank you for attending the session

Question & Answers