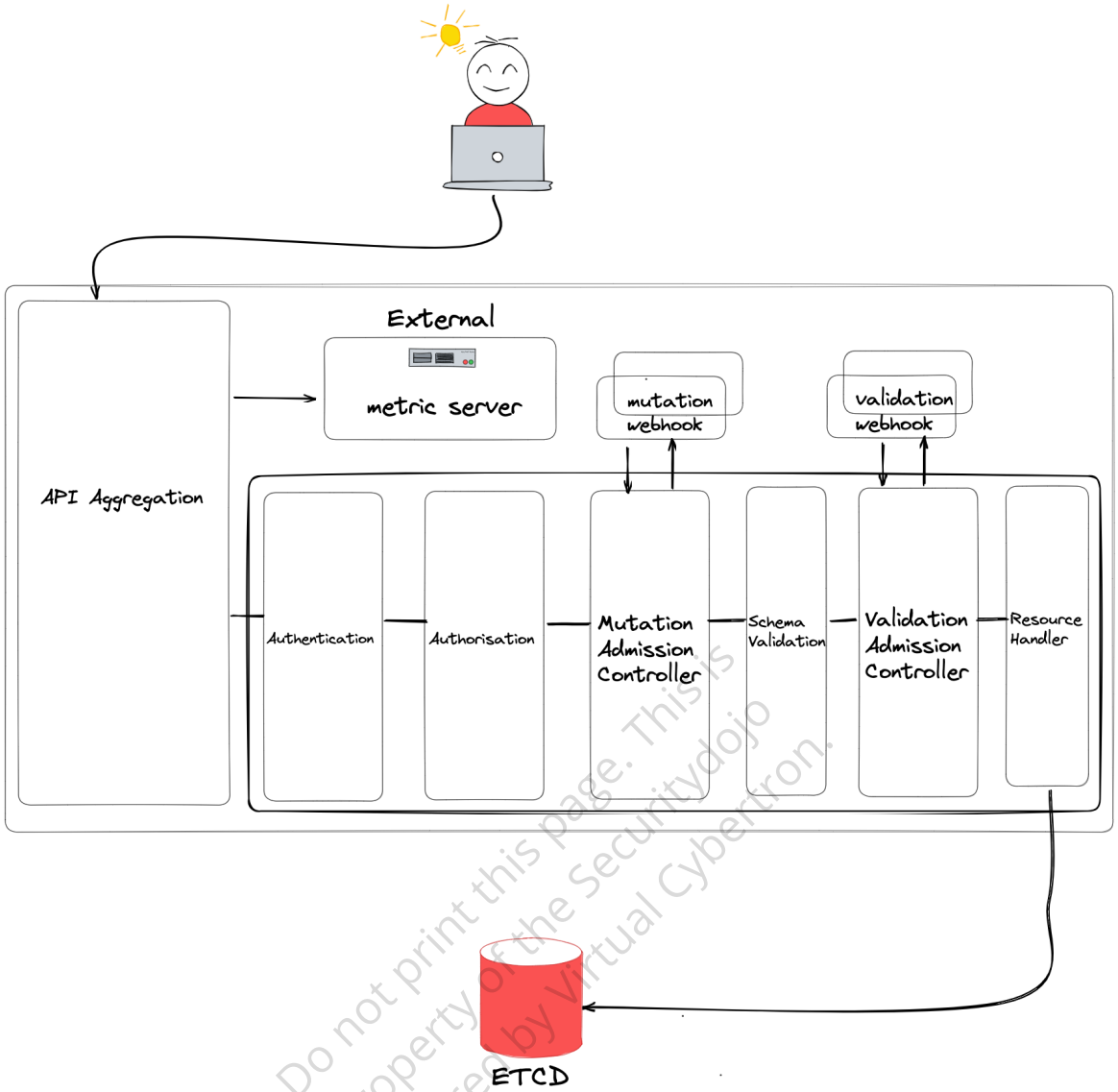
Authorization & Authentication In K8s

Authentication: Who are you? Authentication enables the users to login into the system correctly.

Authorization: What can you do? Authorization grants proper permissions to the users

The Kubernetes API server is responsible for authenticating requests, and it distinguishes between internal and external users. The authentication module consists of various plugins, including static token files, X.509 certificates, Open ID Connect, authentication proxy, and webhook. Once authenticated, the authorization module grants access to resources based on the Role-based Access Control (RBAC) model, where users are assigned permissions through Roles and ClusterRoles, and the link between them is defined through bindings. This process also applies to microservices within the cluster.

- When the kubectl apply is executed, the request is sent to the cluster.
- The API server in the control plane receives that request.
- The API server validates the requests and stores the object in etcd.



Understanding User Management in Kubernetes

- Kubernetes, the popular container orchestration system, distinguishes between two types of users: internal and external. Internal users are managed by Kubernetes and are typically used by applications within the cluster. These are known as workload identities. On the other hand, external users are not managed by Kubernetes and include users authenticated through external identity providers like Keystone, Google account, and LDAP.

- In simpler terms, Kubernetes has a way of managing user accounts. There are two types of users - those that Kubernetes manages internally and those that are external to the system. Internal users are created by Kubernetes itself and are typically used by applications running inside the cluster. External users, on the other hand, come from

outside the system and are authenticated through third-party providers.

Authentication

- In Kubernetes, authentication is the process of verifying the identity of a user, process, or system before granting access to the Kubernetes cluster or its resources.

K8s Authentication Methods

Kubernetes supports various authentication methods such as X.509 Client Certificates, HTTP Basic Authentication, and Bearer Tokens, among others.

- X.509 Client Certificates: This is a commonly used authentication method in Kubernetes. It involves using a digital certificate, typically issued by a Certificate Authority (CA), to authenticate a client to the Kubernetes cluster.
- HTTP Basic Authentication: This is a simple authentication method that involves sending a username and password in clear text over the network. It is not recommended for use in production environments.
- Bearer Tokens: A bearer token is an encoded string that is typically issued by an authentication server and used to authenticate requests to a Kubernetes API server.
- Web Hook: A webhook is a mechanism that allows external systems to be notified of events in the Kubernetes cluster, such as when a user attempts to authenticate.

Working Of Certificate Based Auth

Lab is based on certificate based authentication.

- A user or administrator creates a private key.
- The user or administrator generates a certificate signing request (CSR).
- The administrator approves the request and signs it with their CA.
- The administrator provides the resulting certificate back to the user, who can then use it - to authenticate to the Kubernetes cluster.

Authorization

- Node: Node authorization is a special-purpose authorization mode that specifically authorizes API requests made by kubelets.
- ABAC: Attribute-Based Access Control is a deprecated authorization method in k8s, where access to resources is based on user attributes such as username, group, or role.
- RBAC: Role-Based Access Control is the recommended authorization method in k8s. It uses roles and role bindings to grant access to resources based on user roles, which are defined as sets of permissions.
- Webhook: This refers to an external service that can be used to authenticate and authorize users in k8s, instead of relying solely on k8s internal mechanisms.
- Kubernetes authorization scopes: This refers to the two scopes in which authorization can be applied in k8s, at the cluster level or at the namespace level.
    - Cluster
    - Namespace
- Roles in Kubernetes.
    - Role and RoleBinding: This allows granting permissions to a user or a group of users within a specific namespace.
    - ClusterRole and ClusterRoleBinding: This allows granting permissions across all namespaces in the cluster.

Role In RBAC

In Kubernetes RBAC (Role-Based Access Control), a role is a set of permissions that define what actions can be performed on a specific group of resources.

These roles are used to grant specific privileges to users, groups, or service accounts, ensuring that they can only access and perform actions on the resources necessary for their tasks. This fine-grained access control mechanism helps maintain security and reduce the risk of unauthorized actions within a Kubernetes cluster.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: student-role
rules:
- apiGroups: ['']
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

- This YAML snippet defines a Kubernetes Role in the RBAC system. The role is named "student-role" and is assigned to the "default" namespace.
- It grants permissions to perform three specific actions on pods within the namespace: "get", "watch", and "list".
- The apiGroups field with an empty string ("") indicates that these permissions apply to the core API group, which includes fundamental Kubernetes resources such as pods.
- By assigning this role to a user, group, or service account, you provide them with read-only access to the pod resources within the default namespace.

Role Binding In RBAC

In Kubernetes RBAC (Role-Based Access Control), a RoleBinding is a crucial component that links a role to users, groups, or service accounts.

The purpose of a RoleBinding is to grant the permissions defined in the role to the specified subjects, allowing them to perform the permitted actions on the associated resources.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: student-rolebinding
  namespace: default
subjects:
- kind: User
  name: student
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: student-role
  apiGroup: rbac.authorization.k8s.io
```

- The RoleBinding is named "student-rolebinding" and is scoped to the "default" namespace.
- The "subjects" field specifies that this RoleBinding applies to a user with the name

"student".
- The "apiGroup" field within the "subjects" section is set to "rbac.authorization.k8s.io", indicating the RBAC API group.
- The "roleRef" field references the role that will be bound to the specified subject (the "student" user in this case).
- The "kind" field within "roleRef" is set to "Role", indicating that the binding refers to a namespace-scoped role.
- The "name" field within "roleRef" is set to "student-role", which is the name of the Role to be assigned to the "student" user.

Cluster Role In RBAC

A ClusterRole is a cluster-wide role in Kubernetes RBAC that defines permissions on resources across all namespaces.

It contains a set of rules specifying the allowed actions (e.g., "get", "list", "watch") on resources (e.g., "pods", "services") within any namespace.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: student-clusterrole
rules:
- apiGroups: ['']
  resources: ["nodes"]
  verbs: ["get", "list", "watch"]
```


- The "kind" is set to "ClusterRole", indicating that this is a cluster-wide role.
- The ClusterRole is named "student-clusterrole".
- The "rules" field grants permissions to perform "get", "list", and "watch" actions on "nodes" resources.
- The "apiGroups" field with an empty string ('') indicates that the permissions apply to the core API group, which includes fundamental Kubernetes resources such as nodes.

Cluster Role Binding In RBAC

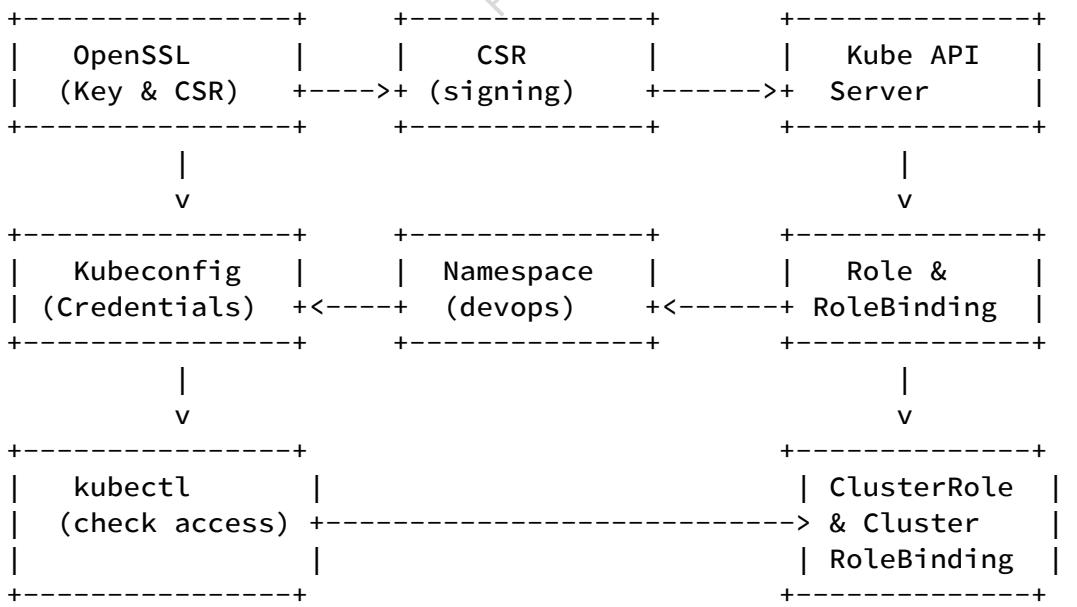
A ClusterRoleBinding is a cluster-wide binding in Kubernetes RBAC that associates a ClusterRole with users, groups, or service accounts.

It grants the permissions defined in the ClusterRole to the specified subjects, enabling them to perform the permitted actions on resources in all namespaces.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: student-clusterrolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: student-clusterrole
subjects:
- kind: User
  name: student
  apiGroup: rbac.authorization.k8s.io
```

- The "kind" is set to "ClusterRoleBinding", indicating that this is a cluster-wide role binding.
- The ClusterRoleBinding is named "student-clusterrolebinding".
- The "roleRef" field references the ClusterRole "student-clusterrole" to be bound to the specified subject.
- The "subjects" field specifies that this ClusterRoleBinding applies to a user named "student".
- The "apiGroup" field within the "subjects" and "roleRef" sections is set to "rbac.authorization.k8s.io", indicating the RBAC API group.

Simplified Diagram of K8s Authentication & Authorization Lab



- Generating a key pair and a Certificate Signing Request (CSR) using OpenSSL.
- Creating a CertificateSigningRequest object in Kubernetes and sending it to the API server.
- The API server approves the CSR, and a certificate is issued.
- Configuring the kubeconfig file with the new user's credentials.
- Creating a namespace (devops) and checking if the user has access to list pods in that namespace.
- Creating a Role, RoleBinding, ClusterRole, and ClusterRoleBinding for the user.
- Using kubectl to check if the user has access to list nodes, as specified in the ClusterRole.