Authorization

→ Using client key, client certificate and ca certificate authenticates an user.

→ This allows access to the cluster

→ Authorization means once a user gains access to the cluster, what they are authorized to do

→ There are different ways to enforce authorization

→ Node Authorization:

- Node Authorizer
- for access within cluster
- for example kubelet lies in Node users group and authorizer kubelet to perform many actions

→ Attribute Based Auth control (ABAC)

- External access
- For example user or a set of users with permissions
- Done by creating a policy file for each user group or user
- After adding each user or groups, you must edit the policy file manually and restart the api server
- Thus, they are bit difficult to manage

→ Role Based Access control (RBAC)

- instead of associating each user or set of users with a set of permissions, we define a Role
- A Role has all the permissions defined
- Any new user or entity is assigned to appropriate Role

- Easy to manage

→ webhook

- agents of 3rd party
- outsourcing the authorization
- api server makes request to the agent who then authorizes or not

→ Always allow

- allows all requests w/o any checks (default)

→ Always deny

- Deny all

HOW And What is used !

→ specify in the argument of api server as --authorization-mode

Role Based Access Control

→ RBAC

→ How do we create a role?

- By creating a role object
- Similar to any object creation, create a yaml file with kind: Role
- This yaml will have rules field that specifies the resources and different verbs that the role is authorized to

→ How to link a user to a role?

- create another object of type RoleBinding
- yaml definition of this has field subjects
- this can have a list of items each with fields like kind, name, apiGroup

which can be used to specify users

- this yaml can have another field roleRef where we can specify what role to assign to

→ How can i check access?

- kubectl auth can-i delete nodes
- the answer is either yes or no
- we can also check access of another user using --as (user) option.

Cluster Role And Bindings

→ Role and role bindings are namespaced

→ Some resources are not namespaced, like nodes

→ To authorize users for cluster wide resources, we use cluster role and cluster role bindings

→ can be created similar to the role binding that we saw above

→ you can also create cluster role and cluster role binding to namespaced object

→ this will give access to resources across all namespaces