https://dreamlab.net/en/blog/post/kubernetes-api-authentication-using-tls-certificates-1/

**❓ 4. What is Security in Kubernetes?**

**🧠 Simple Definition:**

Security in Kubernetes is about **protecting your cluster** — just like locking your house 🏠:

* Who can enter? (**Authentication**)
* What are they allowed to do? (**Authorization**)
* Is anyone watching and recording what's happening? (**Auditing**)
* Are communications safe? (**Encryption and TLS**)

At the center of all this is the **Kubernetes API Server** — it’s the **main gate** to the entire cluster. Everything (kubectl, controllers, users) talks to the API server.

So protecting the **API server** = protecting your cluster.

**🧩 Kubernetes API Server – Why It’s Important**

Think of the API server as:

📞 A receptionist in a secure office.

Everyone — you, your apps, the scheduler, and other parts of Kubernetes — must **talk to the API server** to get anything done.

So it handles:

* Who are you? (Authentication)
* Are you allowed to do this? (Authorization)
* Let me log this action (Auditing)

If the API server is open and unprotected — your entire cluster is at risk.

**🔐 Kubernetes Security Pillar 1: Authentication**

**✅ What is Authentication?**

Authentication is **proving your identity**. It's the **login step**.

"Who are you?"

Types of authentication in Kubernetes:

| **Type** | **Example** | **Who uses it?** |
| --- | --- | --- |
| 📜 Certificate-based | X.509 client cert | Admins, tools |
| 🔑 Token-based | Bearer token | ServiceAccounts |
| 🔒 Username/Password | Basic auth (deprecated) | Testing (not recommended) |
| ☁️ Cloud IAM | GCP, AWS, Azure IAM | Cloud providers |
| 🔌 External | LDAP, OIDC, Dex | Enterprises |

**🔐 Example:**

You try to run kubectl get pods — the API server checks:

“Who is this person? Do they have a valid token or certificate?”

If yes — move to **authorization**.

**✅ Kubernetes Security Pillar 2: Authorization**

**❓ What is Authorization?**

Authorization means:

“Now that I know who you are, **what are you allowed to do**?”

It's like a list of permissions:

* Can you create pods?
* Can you delete services?
* Can you see secrets?

**🧾 Types of Authorization in Kubernetes**

| **Type** | **Description** | **Common Use** |
| --- | --- | --- |
| **RBAC** (Role-Based Access Control) | Uses **roles** and **role bindings** to define access | Most commonly used |
| ABAC (Attribute-Based Access Control) | Uses user labels and attributes (advanced, rare) | Legacy |
| Webhook Authorization | Custom logic via an external server | For custom rules or integrations |
| Node Authorization | Used by kubelet (each node) to limit what it can do | System-level |
| AlwaysAllow / AlwaysDeny | For testing or lockdown mode | Not for production |

**✅ Summary Table: Authentication vs Authorization**

| **Feature** | **Authentication 🔐** | **Authorization ✅** |
| --- | --- | --- |
| Question asked | "Who are you?" | "What can you do?" |
| Output | Identity (user/service) | Allowed actions |
| Examples | Certificates, tokens | RBAC, Webhooks |
| Managed by | API server + kubeconfig | RBAC rules & bindings |

**🧒 Kid-Level Analogy:**

"Think of Kubernetes like a video game arcade 🎮:

* **Authentication** is the game card you scan to enter. It proves it's **your card**.
* **Authorization** is how many tokens/points you have on the card — that tells you **which games you’re allowed to play.**"