**What is kubeconfig?**

* kubeconfig is a configuration file (usually named config) that stores information about how to connect to one or more Kubernetes clusters.
* It is used by tools like kubectl, helm, k9s, or any Kubernetes client library (Go, Python, etc.) to authenticate and interact with the cluster.

By default, Kubernetes looks for this file at:

~/.kube/config

But you can also explicitly pass it with:

kubectl --kubeconfig=/path/to/config get pods

**Why kubeconfig is important?**

Without kubeconfig, kubectl doesn’t know:

* Which cluster API server to talk to
* Which credentials to use for authentication
* Which namespace/context to default to

So kubeconfig acts as the bridge between you and the Kubernetes cluster.

**Structure of a kubeconfig file:-**

It’s a YAML file with 3 main sections:

1. clusters → Info about Kubernetes API servers  
   Example:

clusters:

- name: dev-cluster

cluster:

server: https://123.45.67.89

certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t...

* server: The endpoint of the API server
* certificate-authority-data: Base64-encoded CA cert to verify server’s TLS

1. users → Credentials to authenticate with the cluster  
   Example:

users:

- name: dev-user

user:

client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0...

client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURS...

* Could also be a token, IAM role, OIDC identity provider, etc.

**3.** contexts → Combination of cluster + user + namespace  
 Example:

contexts:

- name: dev-context

context:

cluster: dev-cluster

user: dev-user

namespace: dev

* Lets you quickly switch between different clusters and users

**4.** current-context → Active context being used  
 Example:

current-context: dev-context

* This is the default context kubectl will use unless you override with --context

**How kubeconfig works in practice:-**

1. When you run:

kubectl get pods

* kubectl checks ~/.kube/config (or the path you specify).
* Reads the current-context.
* Finds the associated cluster and user.
* Uses the credentials to authenticate with the API server.
* Sends the API request.

1. If you have multiple clusters, you can switch with:

kubectl config use-context dev-context

**Example of a full kubeconfig:-**

apiVersion: v1

kind: Config

clusters:

- name: dev-cluster

cluster:

server: https://api.dev-cluster.k8s.local

certificate-authority-data: <CA\_DATA>

users:

- name: dev-user

user:

client-certificate-data: <CERT\_DATA>

client-key-data: <KEY\_DATA>

contexts:

- name: dev-context

context:

cluster: dev-cluster

user: dev-user

namespace: default

current-context: dev-context

**Key Points to Remember:-**

* kubeconfig can manage multiple clusters and users.
* It’s just a YAML file → easy to merge or share.
* For EKS/GKE/AKS, cloud CLIs (aws eks update-kubeconfig, gcloud container clusters get-credentials, etc.) automatically generate/update this file.
* You can override it temporarily with environment variable:

export KUBECONFIG=/path/to/other/config