Ingress Controller: The Ingress Controller is responsible for managing inbound traffic to applications running in Kubernetes clusters. It uses Nginx as the underlying reverse proxy to route incoming requests to the appropriate services based on defined rules.

Controller Manager: The Controller Manager component is responsible for reconciling the desired state of Ingress resources with the actual configuration of Nginx instances. It monitors changes to Ingress resources in the Kubernetes cluster and updates the Nginx configuration accordingly to ensure that routing rules are applied correctly.

Custom Resource Definitions (CRDs): Nginx Controller may define custom resource definitions (CRDs) to extend Kubernetes' native capabilities and provide additional features specific to Nginx configurations. For example, CRDs may define custom resources for configuring advanced Nginx settings such as rate limiting, SSL termination, or caching.

Configuration Management: Nginx Controller for Kubernetes typically includes tools and mechanisms for managing Nginx configurations in a Kubernetes-native way. This may involve using ConfigMaps or other Kubernetes resources to define and store Nginx configuration settings, making it easier to manage and update configurations alongside other Kubernetes resources.

Define the YAML Structure: Start by creating a YAML file with the appropriate structure. An Ingress resource in Kubernetes typically consists of several fields, including metadata, spec, and rules.

Specify Metadata: Define the metadata for the Ingress resource, including the name, namespace, and any labels or annotations you want to apply.

Define the Ingress Spec: The spec field contains the configuration for the Ingress resource. This includes rules for routing traffic to different backend services based on hostnames, paths, or other criteria.

Define Ingress Rules: Within the spec field, define the rules for routing traffic. Each rule typically consists of a host field (optional) and a list of HTTP paths and their corresponding backend services.

Define Backend Services: For each path in the Ingress rules, specify the backend service that should receive traffic for that path. This includes the service name and port.

Add Annotations (Optional): Optionally, you can include annotations in the metadata section to configure additional settings for the Ingress resource, such as SSL certificates, load balancing algorithms, or timeouts.

Save the YAML File: Once you've defined the Ingress resource configuration, save the YAML file with a .yaml extension.

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: example-ingress

namespace: default

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: example.com

http:

paths:

- path: /app1

pathType: Prefix

backend:

service:

name: app1-service

port:

number: 80

- path: /app2

pathType: Prefix

backend:

service:

name: app2-service

port:

number: 80