In Kubernetes YAML files, the kind field specifies the type of Kubernetes resource being defined. Here are some common resource types you'll encounter in Kubernetes YAML files:

Each resource type has its own set of fields and specifications for configuring and managing Kubernetes objects.

1. Pod:
    - Defines a single instance of a containerized application.
    - The smallest deployable unit in Kubernetes.
2. ReplicaSet:
    - Ensures that a specified number of pod replicas are running at any given time.
    - A higher-level abstraction over individual pods, providing scaling and self-healing capabilities.
3. Deployment:
    - Manages the deployment and scaling of a set of identical pods, often used for stateless applications.
4. StatefulSet:
    - Manages the deployment and scaling of a set of stateful pods, maintaining a stable identity for each pod.
    - Suitable for stateful applications that require stable storage, network identifiers, and ordered deployment.
5. DaemonSet:
    - Ensures that a copy of a pod runs on all or a subset of nodes in a cluster.
    - Typically used for system-level tasks like logging, monitoring, or networking.
6. Job:
    - Creates one or more pods and ensures that a specified number of them successfully terminate.
    - Useful for batch processing, ETL (Extract, Transform, Load) jobs, or one-time tasks.
7. CronJob:
    - Creates jobs on a recurring schedule, similar to cron jobs in Unix/Linux systems.
    - Automates repetitive tasks like backups, data synchronisation, or periodic maintenance.
8. Service:
    - Defines a logical set of pods and a policy for accessing them.
    - Provides network abstraction to access applications running on Kubernetes, including load balancing and service discovery.
9. Ingress:
    - Manages external access to services within a cluster.
    - Provides HTTP and HTTPS routing, load balancing, and SSL termination for incoming traffic.
10. ConfigMap:
    - Stores configuration data as key-value pairs.
    - Enables decoupling of configuration from application code and simplifies configuration management.
11. Secret:
    - Stores sensitive data like passwords, tokens, or keys.
    - Encrypted at rest and in transit, providing a secure way to manage confidential information.

12. PersistentVolume:
    - Represents a piece of storage in the cluster, provisioned by an administrator.
    - Provides a way for containers to consume durable storage independent of the pod lifecycle.
13. PersistentVolumeClaim:
    - Requests storage resources from a PersistentVolume.
    - Enables dynamic provisioning of storage based on user-defined storage classes.
14. Namespace:
    - Provides a scope for Kubernetes objects, allowing multiple virtual clusters to share the same physical cluster.
    - Helps with resource isolation, access control, and organization of cluster resources.
15. ServiceAccount:
    - Provides an identity for processes running in a pod.
    - Used by Kubernetes to authenticate and authorize actions performed by pods.
16. Role and RoleBinding:
    - Defines a set of permissions (role) and binds them to users or service accounts (role binding).
    - Used for fine-grained access control within a namespace.
17. ClusterRole and ClusterRoleBinding:
    - Similar to Role and RoleBinding but scoped to the entire cluster rather than a specific namespace.
18. HorizontalPodAutoscaler (HPA):
    - Automatically adjusts the number of replicas in a deployment or replica set based on CPU utilization or other metrics.
    - Scales applications up or down to meet changing demand.
19. PodDisruptionBudget (PDB):
    - Ensures that a minimum number of pods in a deployment are available during voluntary disruptions, such as maintenance or updates.
    - Prevents all pods from being simultaneously terminated, ensuring high availability.
20. HorizontalPodAutoscaler (HPA):
    - Automatically adjusts the number of replicas in a deployment or replica set based on CPU utilization or other metrics.
    - Scales applications up or down to meet changing demand.
21. PodDisruptionBudget (PDB):
    - Ensures that a minimum number of pods in a deployment are available during voluntary disruptions, such as maintenance or updates.
    - Prevents all pods from being simultaneously terminated, ensuring high availability.
22. Endpoint:
    - Represents a set of network addresses corresponding to a service.
    - Used by the kube-proxy to route traffic to services.
23. Event:
    - Represents a real-time stream of events emitted by the Kubernetes system components.
    - Provides visibility into the state and changes happening within the cluster.
24. LimitRange:
    - Specifies default resource requests and limits for pods in a namespace.
    - Helps prevent resource contention and ensures fair resource allocation.
25. NetworkPolicy:
    - Defines rules for network traffic within a namespace.
    - Controls which pods can communicate with each other and on which ports.
26. PodSecurityPolicy (PSP):

- Defines a set of conditions that pods must satisfy to be allowed to run in a namespace.
- Enforces security best practices and policies for pod creation and execution.

27. ResourceQuota:
    - Enforces limits on the amount of compute resources (CPU, memory) and storage that can be consumed within a namespace.
    - Helps prevent resource exhaustion and ensures fair resource usage among users or teams.
28. CustomResourceDefinition (CRD):
    - Extends the Kubernetes API with custom resource types.
    - Allows users to define and manage custom resources using Kubernetes' native API machinery.
29. CustomResource:
    - Instances of custom resources defined by CustomResourceDefinitions (CRDs).
    - Enables the creation and management of custom resources using Kubernetes' API and controllers.
30. VolumeSnapshot:
    - Captures the state of a persistent volume at a specific point in time.
    - Enables data backup, restore, and migration for stateful applications using Kubernetes.
31. VolumeSnapshotContent:
    - Represents the content of a volume snapshot, including the snapshot data and metadata.
    - Used by Kubernetes to manage volume snapshots and restore operations.
32. PodTemplate:
    - Defines a template for creating pods in a replication controller, replica set, or job.
    - Allows for easy creation and management of pods with consistent configurations.
33. ValidatingWebhookConfiguration:
    - Defines a configuration for a validating admission webhook.
    - Enables custom validation of Kubernetes API objects before they are persisted in the cluster.
34. MutatingWebhookConfiguration:
    - Defines a configuration for a mutating admission webhook.
    - Allows for automatic modification of Kubernetes API objects before they are persisted in the cluster.
35. ClusterRole and ClusterRoleBinding:
    - Defines a set of permissions (cluster role) and binds them to users or service accounts across the entire cluster (cluster role binding).
    - Provides cluster-wide access control for managing resources and performing actions within Kubernetes.
36. PodSecurityPolicy:
    - Defines a set of security policies and constraints for pods running in a namespace.
    - Enforces security best practices, such as container runtime security, privilege escalation prevention, and host namespace isolation.
37. StorageClass:
    - Defines storage provisioning requirements and parameters for dynamic volume provisioning.
    - Allows administrators to specify storage types, access modes, and other storage properties for persistent volumes.
38. VolumeAttachment:
    - Attaches a volume to a node in a Kubernetes cluster.
    - Facilitates dynamic volume provisioning and lifecycle management for persistent volumes.
39. PriorityClass:

- Assigns priority levels to pods based on their importance or criticality.
- Allows Kubernetes to prioritize scheduling and resource allocation for high-priority workloads.

40. TokenRequest and TokenRequestProjection:
    - Requests short-lived tokens for accessing the Kubernetes API or other cluster resources.
    - Enables fine-grained authentication and authorization for service accounts and other entities within the cluster.
41. EndpointsSlice:
    - A subset of endpoints associated with a service.
    - Provides more granular control over routing and load balancing for services with large numbers of endpoints.
42. Lease:
    - Represents a time-limited claim on a particular resource or functionality within the cluster.
    - Used by various Kubernetes components for coordination, leader election, and resource ownership.
43. RuntimeClass:
    - Defines a class of container runtimes that can be used to run pods in a Kubernetes cluster.
    - Enables administrators to configure and manage different container runtimes, such as Docker, containerd, or CRI-O.
44. CertificateSigningRequest (CSR):
    - Requests a certificate to be signed by the Kubernetes cluster's certificate authority (CA).
    - Used for securing communication between Kubernetes components, API clients, and other external entities.
45. PriorityLevelConfiguration:
    - Defines priority levels for API requests and enforces admission control policies based on request priority.
    - Helps prioritize and manage API requests within a Kubernetes cluster, ensuring fair resource allocation and responsiveness.
46. RuntimeClass:
    - Defines classes of container runtimes that can be used to run pods in a Kubernetes cluster.
    - Enables administrators to configure and manage different container runtimes, such as Docker, containerd, or CRI-O.
47. MutatingWebhookConfiguration and ValidatingWebhookConfiguration:
    - Define configurations for mutating and validating admission webhooks, respectively.
    - Allow for custom processing of Kubernetes API objects before they are persisted or validated in the cluster.
48. PriorityClass:
    - Assigns priority levels to pods based on their importance or criticality.
    - Enables Kubernetes to prioritize scheduling and resource allocation for high-priority workloads.
49. Event:
    - Represents a real-time stream of events emitted by Kubernetes components.
    - Provides visibility into the state and changes happening within the cluster.
50. Issuer and ClusterIssuer:
    - Define configurations for issuing certificates using the cert-manager tool.
    - Enable automated management of TLS certificates for securing communication within the cluster and with external services.
51. CertificateSigningRequest (CSR):
    - Requests a certificate to be signed by the Kubernetes cluster's certificate authority (CA).

- Used for securing communication between Kubernetes components, API clients, and external entities.

52. EndpointSlice:
    - Represents a subset of endpoints associated with a Kubernetes service.
    - Provides more granular control over routing and load balancing for services with large numbers of endpoints.
53. Lease:
    - Represents a time-limited claim on a particular resource or functionality within the cluster.
    - Used for coordination, leader election, and resource ownership by various Kubernetes components.
54. NetworkPolicy:
    - Defines rules for network traffic within a namespace.
    - Controls which pods can communicate with each other and on which ports, enhancing network security within the cluster.
55. PodSecurityPolicy (PSP):
    - Defines a set of conditions that pods must satisfy to be allowed to run in a namespace.
    - Enforces security best practices and policies for pod creation and execution.
56. ResourceQuota:
    - Enforces limits on the amount of compute resources (CPU, memory) and storage that can be consumed within a namespace.
    - Helps prevent resource exhaustion and ensures fair resource usage among users or teams.
57. CustomResourceDefinition (CRD):
    - Extends the Kubernetes API with custom resource types.
    - Allows users to define and manage custom resources using Kubernetes' native API machinery.
58. CustomResource:
    - Instances of custom resources defined by CustomResourceDefinitions (CRDs).
    - Enables the creation and management of custom resources using Kubernetes' API and controllers.
59. RuntimeClass:
    - Defines classes of container runtimes that can be used to run pods in a Kubernetes cluster.
    - Enables administrators to configure and manage different container runtimes, such as Docker, containerd, or CRI-O.
60. APIService:
    - Represents an external API server serving resources that Kubernetes consumes.
    - Used for aggregating and serving resources from multiple API servers within a cluster.
61. ComponentStatus:
    - Reports the health status of various Kubernetes cluster components, such as the scheduler, controller manager, and etcd.
    - Provides insights into the operational state of critical components within the cluster.
62. ControllerRevision:
    - Represents a specific revision of a controller-managed object, such as a deployment or stateful set.
    - Enables rollback and historical tracking of changes made to controller-managed resources.
63. Event:
    - Represents a real-time stream of events emitted by Kubernetes components.
    - Provides visibility into the state and changes happening within the cluster.
64. Issuer and ClusterIssuer:

- Define configurations for issuing certificates using the cert-manager tool.
- Enable automated management of TLS certificates for securing communication within the cluster and with external services.

65. NodeMetrics and PodMetrics:
    - Provide metrics data for nodes and pods, respectively.
    - Used for monitoring and resource utilisation analysis within the cluster.
66. PriorityLevelConfiguration:
    - Defines priority levels for API requests and enforces admission control policies based on request priority.
    - Helps prioritise and manage API requests within a Kubernetes cluster, ensuring fair resource allocation and responsiveness.
67. StorageVersion:
    - Represents the current storage version used by the Kubernetes API server.
    - Enables clients to determine the supported API versions and features for interacting with storage resources.
68. EndpointSlice:
    - Represents a subset of endpoints associated with a Kubernetes service.
    - Provides more granular control over routing and load balancing for services with large numbers of endpoints.
69. PodSecurityPolicy (PSP):
    - Defines a set of conditions that pods must satisfy to be allowed to run in a namespace.
    - Enforces security best practices and policies for pod creation and execution.
70. ResourceQuota:
    - Enforces limits on the amount of compute resources (CPU, memory) and storage that can be consumed within a namespace.
    - Helps prevent resource exhaustion and ensures fair resource usage among users or teams.
71. ValidatingWebhookConfiguration:
    - Defines configurations for validating admission webhooks.
    - Allows for custom validation of Kubernetes API objects before they are persisted in the cluster.
72. MutatingWebhookConfiguration:
    - Defines configurations for mutating admission webhooks.
    - Allows for automatic modification of Kubernetes API objects before they are persisted in the cluster.
73. PriorityLevelConfiguration:
    - Defines priority levels for API requests and enforces admission control policies based on request priority.
    - Helps prioritise and manage API requests within a Kubernetes cluster, ensuring fair resource allocation and responsiveness.
74. PodTemplate:
    - Defines a template for creating pods in a replication controller, replica set, or job.
    - Allows for easy creation and management of pods with consistent configurations.
75. NetworkPolicy:
    - Defines rules for network traffic within a namespace.
    - Controls which pods can communicate with each other and on which ports, enhancing network security within the cluster.
76. ClusterRole and ClusterRoleBinding:
    - Define permissions (cluster role) and bind them to users or service accounts across the entire cluster.

- Provide cluster-wide access control for managing resources and performing actions within Kubernetes.

77. PodDisruptionBudget (PDB):
    - Ensures that a minimum number of pods in a deployment are available during voluntary disruptions.
    - Prevents all pods from being simultaneously terminated, ensuring high availability.
78. StorageClass:
    - Defines storage provisioning requirements and parameters for dynamic volume provisioning.
    - Allows administrators to specify storage types, access modes, and other storage properties for persistent volumes.
79. VolumeAttachment:
    - Attaches a volume to a node in a Kubernetes cluster.
    - Facilitates dynamic volume provisioning and lifecycle management for persistent volumes.
80. APIService:
    - Represents an external API server serving resources that Kubernetes consumes.
    - Used for aggregating and serving resources from multiple API servers within a cluster.
81. CertificateSigningRequest (CSR):
    - Requests a certificate to be signed by the Kubernetes cluster's certificate authority (CA).
    - Used for securing communication between Kubernetes components, API clients, and external entities.
82. ComponentStatus:
    - Reports the health status of various Kubernetes cluster components, such as the scheduler, controller manager, and etcd.
    - Provides insights into the operational state of critical components within the cluster.
83. ControllerRevision:
    - Represents a specific revision of a controller-managed object, such as a deployment or stateful set.
    - Enables rollback and historical tracking of changes made to controller-managed resources.
84. Issuer and ClusterIssuer:
    - Define configurations for issuing certificates using the cert-manager tool.
    - Enable automated management of TLS certificates for securing communication within the cluster and with external services.
85. NodeMetrics and PodMetrics:
    - Provide metrics data for nodes and pods, respectively.
    - Used for monitoring and resource utilisation analysis within the cluster.
86. PriorityLevelConfiguration:
    - Defines priority levels for API requests and enforces admission control policies based on request priority.
    - Helps prioritise and manage API requests within a Kubernetes cluster, ensuring fair resource allocation and responsiveness.
87. StorageVersion:
    - Represents the current storage version used by the Kubernetes API server.
    - Enables clients to determine the supported API versions and features for interacting with storage resources.
88. VolumeSnapshot:
    - Captures the state of a persistent volume at a specific point in time.
    - Enables data backup, restore, and migration for stateful applications using Kubernetes.

89. VolumeSnapshotContent:
    - Represents the content of a volume snapshot, including the snapshot data and metadata.
    - Used by Kubernetes to manage volume snapshots and restore operations.
90. NetworkPolicy:
    - Defines rules for network traffic within a namespace.
    - Controls which pods can communicate with each other and on which ports, enhancing network security within the cluster.
91. RuntimeClass:
    - Defines classes of container runtimes that can be used to run pods in a Kubernetes cluster.
    - Enables administrators to configure and manage different container runtimes, such as Docker, containerd, or CRI-O.
92. APIService:
    - Represents an external API server serving resources that Kubernetes consumes.
    - Used for aggregating and serving resources from multiple API servers within a cluster.
93. CertificateSigningRequest (CSR):
    - Requests a certificate to be signed by the Kubernetes cluster's certificate authority (CA).
    - Used for securing communication between Kubernetes components, API clients, and external entities.
94. ComponentStatus:
    - Reports the health status of various Kubernetes cluster components, such as the scheduler, controller manager, and etcd.
    - Provides insights into the operational state of critical components within the cluster.
95. ControllerRevision:
    - Represents a specific revision of a controller-managed object, such as a deployment or stateful set.
    - Enables rollback and historical tracking of changes made to controller-managed resources.
96. Issuer and ClusterIssuer:
    - Define configurations for issuing certificates using the cert-manager tool.
    - Enable automated management of TLS certificates for securing communication within the cluster and with external services.
97. NodeMetrics and PodMetrics:
    - Provide metrics data for nodes and pods, respectively.
    - Used for monitoring and resource utilisation analysis within the cluster.
98. PriorityLevelConfiguration:
    - Defines priority levels for API requests and enforces admission control policies based on request priority.
    - Helps prioritise and manage API requests within a Kubernetes cluster, ensuring fair resource allocation and responsiveness.
99. StorageVersion:
    - Represents the current storage version used by the Kubernetes API server.
    - Enables clients to determine the supported API versions and features for interacting with storage resources.
100. PodSecurityPolicy (PSP):
    - Defines a set of conditions that pods must satisfy to be allowed to run in a namespace.
    - Enforces security best practices and policies for pod creation and execution.