CKA · CHEAT SHEET · CERTIFYDECK

Certified Kubernetes Administrator (CKA)

Dominios, pesos oficiales y habilidades clave del temario de CKA, para repasar antes del examen.

Nivel: Intermediate (Associate) Duración: 120 min Fuente: guía de estudio oficial de Microsoft Learn

1. Cluster Architecture, Installation and Configuration

25–25%

Manage role based access control (RBAC)

- Un Role concede permisos aditivos dentro de un espacio de nombres y un ClusterRole los define a nivel de clúster; un RoleBinding que referencia un ClusterRole solo lo concede en su propio espacio de nombres. El grupo core se escribe "", los subrecursos como pods/log se indican con barra, roleRef es inmutable y kubectl auth can-i --as permite comprobar permisos de usuarios y ServiceAccounts.

Prepare underlying infrastructure for installing a Kubernetes cluster

- Cada nodo necesita un Linux compatible, hostname, MAC y product_uuid únicos, los puertos requeridos abiertos y un runtime compatible con CRI como containerd o CRI-O, con el mismo driver de cgroup (systemd) que el kubelet. Hay que activar net.ipv4.ip_forward, desactivar el swap de forma persistente o tolerarlo con failSwapOn: false, e instalar kubeadm, kubelet y kubectl desde pkgs.k8s.io fijando sus versiones.

Create and manage Kubernetes clusters using kubeadm

- kubeadm init crea el plano de control con manifiestos de Pods estáticos en /etc/kubernetes/manifests y certificados en /etc/kubernetes/pki; después se copia admin.conf a $HOME/.kube/config y se instala un complemento de red CNI, sin el cual CoreDNS queda en Pending. Los nodos se unen con kubeadm join, un token y --discovery-token-ca-cert-hash; kubeadm token create --print-join-command genera un comando nuevo y kubeadm reset revierte un nodo.

Manage the lifecycle of Kubernetes clusters

- Un clúster de kubeadm se actualiza de una versión menor en una: kubeadm upgrade apply en el primer nodo del plano de control, kubeadm upgrade node en los demás y, en cada nodo, drain, actualización del kubelet, reinicio y uncordon. La copia de seguridad de etcd se hace con etcdctl snapshot save y se restaura con etcdutl snapshot restore; los certificados se revisan con kubeadm certs check-expiration y se renuevan con kubeadm certs renew.

Implement and configure a highly-available control plane

- Un plano de control de alta disponibilidad con kubeadm usa al menos tres nodos tras un equilibrador de carga fijado en --control-plane-endpoint, con etcd apilado (más sencillo) o externo (más hosts). kubeadm init --upload-certs y kubeadm join --control-plane --certificate-key reparten los certificados; etcd necesita quórum, por eso conviene un número impar de miembros, y kube-scheduler y kube-controller-manager eligen un líder mediante Leases.

Use Helm and Kustomize to install cluster components

- Use Helm and Kustomize to install cluster components

Understand extension interfaces (CNI, CSI, CRI, etc.)

- El kubelet habla con el runtime de contenedores (containerd, CRI-O) por gRPC mediante la API v1 de CRI, y crictl permite inspeccionar Pods, contenedores e imágenes en el nodo. Es el runtime, no el kubelet, quien carga los plugins CNI (configuración en /etc/cni/net.d, binarios en /opt/cni/bin); sin red de Pods CoreDNS se queda en Pending. CSI y los device plugins añaden almacenamiento y hardware sin cambiar el código de Kubernetes.

Understand CRDs, install and configure operators

- Un CustomResourceDefinition (apiextensions.k8s.io/v1) se llama <plural>.<group>, fija el ámbito Namespaced o Cluster y declara versiones con served y una sola storage, cada una con un esquema openAPIV3Schema estructural que poda los campos desconocidos. Un operador combina recursos personalizados con un controlador, normalmente un Deployment, cuya cuenta de servicio necesita permisos RBAC sobre esos recursos.

2. Workloads and Scheduling

15–15%

Understand application deployments and how to perform rolling update and rollbacks

- Un Deployment solo inicia un rollout, y crea una revisión nueva, cuando cambia su plantilla de Pod; escalar no lo hace. RollingUpdate limita el proceso con maxSurge y maxUnavailable, mientras que Recreate elimina todos los Pods antes de crear los nuevos. kubectl rollout status, history, undo --to-revision, pause, resume y restart gestionan los cambios, y revisionHistoryLimit decide cuántos ReplicaSets antiguos quedan para revertir.

Use ConfigMaps and Secrets to configure applications

- Los ConfigMaps y Secrets se crean con kubectl create configmap o secret (generic, tls, docker-registry) y se consumen como variables (env.valueFrom, envFrom) o como volúmenes del mismo espacio de nombres. Los volúmenes se actualizan solos; las variables y los montajes con subPath no. base64 no es cifrado: el cifrado en reposo requiere una EncryptionConfiguration en kube-apiserver.

Configure workload autoscaling

- Un HorizontalPodAutoscaler autoscaling/v2 ajusta las réplicas de un Deployment o StatefulSet según métricas como la utilización de CPU, que se calcula sobre las solicitudes y requiere metrics-server. kubectl autoscale lo crea; behavior controla la ventana de estabilización y las políticas. VerticalPodAutoscaler es un complemento aparte que ajusta solicitudes, y el redimensionado en caliente usa el subrecurso resize.

Understand the primitives used to create robust, self-healing, application deployments

- Los controladores (ReplicaSet, Deployment, StatefulSet, DaemonSet, Job, CronJob) recrean Pods para mantener el estado deseado. Los sondeos de actividad reinician contenedores bloqueados, los de preparación retiran el Pod de los Services y los de inicio protegen arranques lentos. Los contenedores de inicio se ejecutan en orden, los sidecars son initContainers con restartPolicy: Always y un PodDisruptionBudget limita los desalojos voluntarios.

Configure Pod admission and scheduling (limits, node affinity, etc.)

- El programador coloca un Pod solo donde la suma de solicitudes cabe en la capacidad asignable; nodeSelector y la afinidad de nodo (términos con OR, expresiones con AND) eligen nodos, y las manchas con tolerancias los repelen. LimitRange inyecta valores por defecto y ResourceQuota limita totales por espacio de nombres; Pod Security Admission aplica enforce a los Pods, no a los Deployments.

3. Services and Networking

20–20%

Understand connectivity between Pods

- Cada Pod recibe su propia IP única en el clúster y los contenedores de un Pod comparten el espacio de red, así que se hablan por localhost y no pueden usar el mismo puerto. Los Pods se comunican entre nodos sin NAT gracias al plugin CNI; kube-proxy (iptables o nftables) solo implementa las IP virtuales de los Service, y los rangos de Pods y Service no deben solaparse.

Define and enforce Network Policies

- Las NetworkPolicy (networking.k8s.io/v1) solo surten efecto con un plugin de red que las implemente y son reglas de permiso aditivas: un Pod queda aislado en un sentido cuando una política lo selecciona con ese policyType. podSelector: {} crea una denegación por defecto; namespaceSelector y podSelector en el mismo elemento se combinan con AND, y tras denegar la salida hay que permitir DNS por UDP y TCP 53.

Use ClusterIP, NodePort, LoadBalancer service types and endpoints

- ClusterIP es el tipo por defecto y solo es accesible dentro del clúster; NodePort abre el mismo puerto (del rango --service-node-port-range) en cada nodo y LoadBalancer depende de una integración externa que aprovisione el equilibrador. Los EndpointSlices, enlazados por la etiqueta kubernetes.io/service-name, registran los Pods listos; externalTrafficPolicy: Local conserva la IP del cliente.

Use the Gateway API to manage Ingress traffic

- Gateway API se instala como CRDs junto con un controlador: GatewayClass nombra el controlador (controllerName), Gateway define los listeners y allowedRoutes, y HTTPRoute se adjunta con parentRefs y enruta por ruta, cabeceras o pesos con filtros como RequestRedirect o URLRewrite. Las referencias entre espacios de nombres necesitan un ReferenceGrant en el destino; revisa siempre status.conditions.

Know how to use Ingress controllers and Ingress resources

- Un Ingress networking.k8s.io/v1 no hace nada sin un controlador de Ingress en ejecución; spec.ingressClassName elige la IngressClass y la anotación is-default-class marca la predeterminada. Cada ruta exige pathType (Exact, Prefix o ImplementationSpecific), Prefix compara elemento a elemento y gana la ruta más larga; TLS usa un Secret kubernetes.io/tls cuyo host coincide con la regla.

Understand and use CoreDNS

- CoreDNS se configura con el Corefile del ConfigMap coredns de kube-system y se expone mediante el Service kube-dns; plugins como kubernetes, forward, cache y log definen su comportamiento, y los dominios stub se añaden como bloques de servidor con forward. Los Services se resuelven como nombre.espacio.svc.cluster.local, y dnsPolicy (ClusterFirst, ClusterFirstWithHostNet, Default, None) y dnsConfig controlan el /etc/resolv.conf de cada Pod.

4. Storage

10–10%

Implement storage classes and dynamic volume provisioning

- Una StorageClass define provisioner, parameters, reclaimPolicy (Delete si se omite), volumeBindingMode (Immediate o WaitForFirstConsumer), allowVolumeExpansion y mountOptions. La anotación storageclass.kubernetes.io/is-default-class marca la clase predeterminada que el controlador DefaultStorageClass asigna a los PVC sin storageClassName; un PVC con storageClassName "" no usa aprovisionamiento dinámico.

Configure volume types, access modes and reclaim policies

- emptyDir (medium Memory, sizeLimit), configMap, secret y projected viven con el Pod; hostPath, local (con nodeAffinity obligatorio en el PV) y nfs usan almacenamiento externo, y persistentVolumeClaim da almacenamiento duradero. Los modos ReadWriteOnce, ReadOnlyMany, ReadWriteMany y ReadWriteOncePod (solo CSI) sirven para emparejar, y la política Retain o Delete se cambia con kubectl patch pv en persistentVolumeReclaimPolicy.

Manage persistent volumes and persistent volume claims

- Un PersistentVolumeClaim se vincula, uno a uno, a un PV que cumpla capacidad, modos de acceso, storageClassName, volumeMode y selector; si no lo hay, queda en Pending. Con Retain, al borrar la reclamación el PV pasa a Released y conserva los datos hasta que se quita su claimRef. Para ampliar se edita la reclamación, nunca el PV, y la protección pvc-protection impide borrar una reclamación que usa un Pod.

5. Troubleshooting

30–30%

Troubleshoot clusters and nodes

- Un nodo NotReady se diagnostica con kubectl describe node (condiciones y taints como not-ready o unreachable) y, en el propio nodo, con systemctl status kubelet y journalctl -u kubelet. Causas habituales: el runtime detenido, la swap activa, la falta de red CNI, certificados del kubelet caducados o un kubeconfig de otro clúster; la configuración está en /var/lib/kubelet/config.yaml y las credenciales, en /etc/kubernetes/kubelet.conf.

Troubleshoot cluster components

- En un clúster de kubeadm, kube-apiserver, etcd, kube-scheduler y kube-controller-manager son Pods estáticos que el kubelet crea desde /etc/kubernetes/manifests; un flag erróneo impide que arranquen. Si el servidor de API cae, se diagnostica en el nodo con crictl ps -a y crictl logs; /readyz y /livez indican si acepta tráfico o necesita reinicio.

Monitor cluster and application resource usage

- kubectl top node y kubectl top pod (con --containers, --sort-by=cpu o memory y -A) leen la Metrics API que sirve metrics-server a partir del kubelet. La CPU se mide en milicores y la memoria como working set; compárala con solicitudes y límites: superar el límite de memoria provoca OOMKilled (código 137) y la presión del nodo provoca desalojos.

Manage and evaluate container output streams

- Los contenedores deben escribir en stdout y stderr: el kubelet guarda esa salida en /var/log/pods, la rota según containerLogMaxSize y containerLogMaxFiles y la sirve con kubectl logs (-c, --previous, --tail, --since, -l). Si la aplicación escribe en archivos, un sidecar de streaming los pasa a su stdout; para depurar se usan kubectl exec y kubectl debug con un contenedor efímero.

Troubleshoot services and networking

- Si un Service no responde, comprueba que existe, que su selector coincide con las etiquetas de Pods listos (kubectl get endpointslices -l kubernetes.io/service-name=web) y que targetPort apunta al puerto del contenedor. Después prueba la resolución DNS desde un Pod temporal, el estado de CoreDNS y del Service kube-dns, las NetworkPolicy aplicadas y que kube-proxy funcione en el nodo.