CKAD

Certified Kubernetes Application Developer

Domains & Competencies

- Application Design and Build: 20%
- Application Deployment: 20%
- Application Observability and Maintenance: 15%
- Application Environment, Configuration and Security: 25%
- Services and Networking: 20%

Application Design and Build (20%)

- Define, build and modify container images
- Choose and use the right workload resource (Deployment, DaemonSet, CronJob, etc.)
- Understand multi-container Pod design patterns (e.g. sidecar, init and others)
- Utilize persistent and ephemeral volumes

Application Deployment (20%)

- Use Kubernetes primitives to implement common deployment strategies (e.g. blue/green or canary)
- Understand Deployments and how to perform rolling updates
- Use the Helm package manager to deploy existing packages
- Kustomize

Application Observability and Maintenance (15%)

- Understand API deprecations
- Implement probes and health checks
- Use built-in CLI tools to monitor Kubernetes applications
- Utilize container logs
- Debugging in Kubernetes

Application Environment, Configuration and Security (25%)

- Discover and use resources that extend Kubernetes (CRD, Operators)
- Understand authentication, authorization and admission control
- Understand requests, limits, quotas
- Understand ConfigMaps
- Define resource requirements
- Create & consume Secrets
- Understand ServiceAccounts
- Understand Application Security (SecurityContexts, Capabilities, etc.)

Services and Networking (20%)

- Demonstrate basic understanding of NetworkPolicies
- Provide and troubleshoot access to applications via services
- Use Ingress rules to expose applications

Define, build and modify container images

- Container image
- Containerfile (a.k.a. Dockerfile)

- Linux namespaces
- Control groups
- Container engine

Choose and use the right workload resource

- Deployment
- ReplicaSet
- ReplicationController
- Pod
- DaemonSet
- StatefulSet
- CronJob
- Job

Understand multi-container Pod design patterns

- containers
- initContainers

Utilize persistent and ephemeral volumes

- volumes
- emptyDir
- persistentVolumeClaim

Implement common deployment strategies

- kubectl apply
- Deployment
- Service

Deployments and how to perform rolling updates

- kubectl rollout
- strategy.rollingUpdate

Use Helm package manager to deploy existing packages

- helm search repo
- helm repo add
- helm repo update
- helm install
- helm upgrade
- helm list
- helm uninstall

Kustomize

- base/kustomization.yaml
- kubectl apply -k
- resources
- patchesStrategicMerge
- configMapGenerator
- secretGenerator
- images
- kubectl kustomize

Implement probes and health checks

- livenessProbe
- readinessProbe
- startupProbe

Use built-in CLI tools to monitor Kubernetes applications

- kubectl get
- kubectl describe
- kubectl top

Utilize container logs

- kubectl logs

Debugging in Kubernetes

- kubectl exec
- kubectl port-forward
- kubectl debug

Discover and use resources that extend Kubernetes

- Custom Resource Definitions (CRDs)
- Operators

Authentication, authorization and admission control

- Authentication (Who are you?)
  - X.509 client certificates
  - Bearer tokens (e.g., from service accounts)
  - OIDC (via identity providers like Google, Azure AD)
  - Webhook token authentication
- Authorization (What are you allowed to do?)
  - Role: permissions within a namespace
  - ClusterRole: cluster-wide permissions
  - RoleBinding and ClusterRoleBinding: assign roles to users/groups/service accounts
- Admission Control (Should we allow it?)
  - NamespaceLifecycle: prevents deleting system namespaces
  - LimitRanger: enforces resource limits
  - PodSecurity: enforces pod security standards (e.g., no privileged pods)
  - ValidatingAdmissionWebhook / MutatingAdmissionWebhook: custom logic via webhooks

Understand requests, limits, quotas

- Requests and Limits (Per Pod/Container)
- Quota (Per Namespace)
- LimitRange (Default resource policies)

Understand ConfigMaps

- Create a ConfigMap
  - From a file
  - From key-value pairs
  - From a YAML file
- Use ConfigMap in a Pod
  - As environment variables
  - As individual env vars
  - As files in a volume
- Inspect and Manage ConfigMaps

Define resource requirements

- requests: the minimum resources required (used for scheduling)
- limits: the maximum resources a container can use (enforced at runtime)

Create & consume Secrets

- Create a Secret
  - From key-value pairs (base64 encoded by Kubernetes)
  - From a YAML file
- View and Decode a Secret
- Consume Secrets in Pods
  - As Environment Variables
  - As Files in Volumes

Understand ServiceAccounts

- Authenticate pods to the Kubernetes API
- Attach permissions via RBAC
- Separate workloads with different privilege levels
- Avoid embedding credentials into containers

Understand Application Security

- SecurityContext Overview
- Capabilities
- Key SecurityContext Fields
- Avoid Privileged Containers
- Read-Only Root FileSystem
- AppArmor / seccomp / SELinux (Advanced)

Demonstrate basic understanding of NetworkPolicies

- Ingress: Controls incoming traffic to a pod
- Egress: Controls outgoing traffic from a pod
- PodSelector: Selects which pods the policy applies to
- NamespaceSelector: Selects pods in other namespaces

Troubleshoot access to applications via services

- ClusterIP: Internal access only within cluster
- NodePort: Exposes service on each Node’s IP at a static port
- LoadBalancer: Exposes externally using cloud provider's load balancer
- ExternalName: Maps service to an external DNS name

Use Ingress rules to expose applications

- Install an Ingress Controller
- Define Ingress Resource