Kubernetes Interview Cheat Sheet

Q1: What is Kubernetes?	**A container orchestration platform that automates deployment, scaling, and management of containe**
Q2: Difference between Docker Swarm and Kubernetes?	Swarm: simple, limited. K8s: powerful, scaling, self-healing, RBAC.
Q3: What is a Pod?	Smallest deployable unit, runs one/more containers sharing network & storage.
Q4: What is a Node?	Worker machine where pods run; runs kubelet, kube-proxy, container runtime.
Q5: What is a Cluster?	Collection of master and worker nodes managed together.
Q6: Explain Architecture.	Control Plane (API server, etcd, scheduler, controllers) + Nodes (kubelet, kube-proxy).
Q7: What is a ReplicaSet?	Ensures desired number of pod replicas are running.
Q8: What is a Deployment?	Manages ReplicaSets, allows rollouts/rollbacks, scaling.
Q9: Deployment vs StatefulSet?	Deployment: stateless. StatefulSet: stateful, stable IDs, persistent storage.
Q10: What is a DaemonSet?	Ensures one pod runs per node, e.g., monitoring/logging agents.
Q11: What is a Service?	Abstracts pod IPs, provides stable access (ClusterIP, NodePort, LB).
Q12: Service Types?	ClusterIP: internal. NodePort: external via node IP. LB: cloud LB. ExternalName: DNS alias.
Q13: What is Ingress?	Manages HTTP/HTTPS traffic, routes by host/path, SSL termination.
Q14: Ingress Controller?	Implements ingress rules (e.g., NGINX, Traefik).
Q15: Kubernetes DNS?	CoreDNS gives each service a DNS name (<svc>.<ns>.svc.cluster.local).
Q16: CNI Plugins?	Networking plugins (Calico, Flannel, Cilium) for pod networking.
Q17: kube-proxy modes?	iptables: slower. IPVS: kernel-level, faster.
Q18: Persistent Volume?	Cluster-wide storage abstraction (NFS, EBS).
Q19: PVC?	Storage request by apps, binds to PV.
Q20: StorageClass?	Defines dynamic PV provisioning.
Q21: ConfigMap vs Secret?	ConfigMap: non-sensitive. Secret: sensitive, base64 encoded.
Q22: Scheduler?	Assigns pods to nodes based on resources, rules.
Q23: Taints & Tolerations?	Taints repel pods; tolerations allow pods to run on tainted nodes.
Q24: Affinity vs Anti-Affinity?	Affinity: place together. Anti-Affinity: spread apart.
Q25: HPA?	Scales pods based on CPU/memory/custom metrics.
Q26: VPA?	Adjusts pod resource requests/limits dynamically.
Q27: Cluster Autoscaler?	Scales nodes up/down based on workload.
Q28: RBAC?	Controls permissions with Roles/Bindings for users, groups, service accounts.
Q29: Service Accounts?	K8s identities for pods to securely access API.
Q30: PSS?	Pod Security Standards: Privileged, Baseline, Restricted.
Q31: Admission Controllers?	Interceptors that enforce policies before object persistence.
Q32: Job?	Runs pods to completion for batch tasks.
Q33: CronJob?	Runs jobs on a schedule (like cron).
Q34: Init Containers?	Run before app containers for setup tasks.
Q35: Sidecar Containers?	Support containers running alongside main app (logging, proxy).

Q36: Ephemeral Containers?	Temporary containers for debugging running pods.
Q37: CRD?	Custom Resource Definitions to extend API.
Q38: Operator?	Automates app lifecycle via CRDs + controllers.
Q39: Helm?	Kubernetes package manager using charts.
Q40: Kustomize?	Patch-based configuration tool built into kubectl.
Q41: Service Mesh?	Manages service-to-service comms (security, traffic, observability).
Q42: Istio?	Service mesh with mTLS, routing, telemetry, sidecar proxies.
Q43: Monitoring?	Prometheus + Grafana for metrics; ELK/EFK for logs.
Q44: GitOps?	Manage deployments via Git as single source of truth (ArgoCD, Flux).
Q45: CSI?	Container Storage Interface: standard for storage plugins.
Q46: Pod Disruption Budgets?	Ensures min pods remain available during disruptions.
Q47: Canary Deployment?	Gradual rollout to subset of users.
Q48: Blue-Green Deployment?	Two environments; switch traffic to new one when stable.
Q49: Disaster Recovery?	Backup etcd, use Velero, GitOps manifests.
Q50: Troubleshooting CrashLoop?	Check logs, describe pod, probes, configs, resources.