Architecture & Tenancy

VMware Certified Advanced Professional - VMware Cloud Foundation VKS

KNOWLEDGE CHEATSHEET

PLATFORM CONSUMPTION

Supervisor — Platform API → **Namespace** — Tenant boundary → **VKS cluster** — Kubernetes → **Pod** — Application unit

01 Supervisor architecture and tenancy

Supervisor, clusters and pods

- **vSphere Supervisor** exposes a declarative Kubernetes API for consuming vSphere infrastructure and services. Its control plane manages platform resources; a **VKS cluster** has its own Kubernetes control plane and worker nodes.
- **VKS** integrates Kubernetes authentication, persistent storage, pod networking and load balancing with the vSphere platform. **VM Service** provisions virtual machines through declarative resources.
- A **Pod** is Kubernetes' smallest deployable unit. Its containers share networking and can share volumes; a Pod can contain more than one container.
- A **vSphere Pod** uses a lightweight virtualization boundary with its own Linux kernel. Distinguish this Supervisor workload from a conventional Pod running inside a VKS worker VM.

Infrastructure and failure domains

- Prepare compatible **ESX hosts**, shared storage policies, **vSphere HA**, fully automated **DRS** and distributed networking before enabling a Supervisor. Validate requirements against the selected networking architecture.
- **VDS networking** and **NSX networking** have different infrastructure and load-balancer dependencies. Plan management and workload networks, IP pools, routing, DNS, NTP and certificate trust before activation.
- A **vSphere Zone** represents a placement and failure domain backed by vSphere infrastructure. Multi-zone designs spread eligible control-plane and workload instances across failure domains to improve availability.
- Plan **zone topology**, available capacity and storage accessibility together. Assign the intended zones to namespaces and cluster placement; confirm supported topology changes before expanding an existing Supervisor.

Namespaces, capacity and permissions

- A **vSphere Namespace** is a tenancy and resource-consumption boundary. Associate permissions, compute limits, storage policies, VM classes and applicable networking or zone settings with that boundary.
- A **VM Class** defines a VM's CPU and memory sizing and resource guarantees. Make suitable classes available to the namespace; choose node pools for each workload's resource profile.

Architecture & Tenancy / Lifecycle & Access

VMware Certified Advanced Professional - VMware Cloud Foundation VKS

KNOWLEDGE CHEATSHEET

DECLARATIVE LIFECYCLE

Manifest Desired state → **API** Validate → **Controller** Reconcile → **Status** Observe

- Namespace **CPU, memory and storage limits** constrain consumption. Available physical capacity, namespace limits and workload requests all affect whether new nodes or volumes can be provisioned.
- Kubernetes **ResourceQuota** caps aggregate namespace consumption; **LimitRange** supplies or constrains per-object requests and limits. Quotas govern capacity, while RBAC governs allowed actions.

02 Cluster lifecycle and access

Desired state and reconciliation

- **Cluster API** uses Kubernetes custom resources and controllers to create, configure, scale and upgrade clusters. **Cluster**, **Machine** and **MachineDeployment** describe infrastructure lifecycle rather than application replicas.
- A workload **Deployment** controls Pods; a cluster's worker topology controls nodes. More Pods require application scaling, while more workers require node scaling. Neither action automatically supplies the other.
- Keep the **declarative manifest** in sync with the desired configuration. A later apply, GitOps reconciliation or autoscaler can replace a manual replica edit with the value it owns.
- **kubectl edit** updates a live API object after the editor saves successfully. Set **KUBE_EDITOR** or **EDITOR** when needed; for file-based changes, validate the YAML, apply it and inspect resulting status.

Upgrade layers and service lifecycle

- Treat **vCenter**, **Supervisor**, the **VKS service** and each **workload cluster's Kubernetes release** as distinct lifecycle layers. Updating a management layer does not by itself upgrade an application's target cluster.
- Before an upgrade, check the **compatibility matrix**, supported upgrade path, node capacity, storage, image availability and workload health. Review the target API schema for removed fields or deprecated cluster variables.
- **Asynchronous service releases** allow supported service updates outside the bundled platform release. With a private registry, stage the required artifacts and configure registry location, credentials and trusted CA certificates.
- **Core Supervisor services** are platform dependencies. Use their supported upgrade workflow; do not treat deleting a core service as routine upgrade remediation. For optional services, distinguish uninstalling a Supervisor instance from removing its registered service definition.

Identity, contexts and authorization

- Use the **vSphere UI**, supported **VCF CLI** workflows and **kubectl** for their respective platform and Kubernetes tasks. A named CLI context stores connection settings; selecting it does not grant additional permissions.
- A **kubeconfig context** selects a cluster, user and optional namespace. Check **kubectl config current-context**, list contexts and use **kubectl config use-context** before changing resources.

Lifecycle & Access / Networking & Services

VMware Certified Advanced Professional - VMware Cloud Foundation VKS

KNOWLEDGE CHEATSHEET

APPLICATION CONNECTIVITY

- **CNI** — Pod network
- **Ingress** — External routing
- **Mesh** — Service traffic
- **TLS** — Trust and encryption

- **Authentication** establishes identity; **RBAC** authorizes operations. A **Role** grants namespace-scoped permissions through a **RoleBinding**. A **ClusterRole** can be bound in one namespace or cluster-wide through the appropriate binding.
- **OIDC** provides token-based identity federation for Kubernetes through the configured integration. Directory protocols, identity provisioning and API authorization serve separate purposes. Check effective access with **kubectl auth can-i**.

03 Networking and application services

Pod connectivity and traffic policy

- **CNI** implements Pod networking. **Antrea** provides Kubernetes networking and policy enforcement in supported VKS deployments; distinguish a workload cluster's CNI from the Supervisor's infrastructure network choice.
- A **Service** supplies a stable endpoint for selected Pods. **ClusterIP** is internal, **NodePort** exposes a port on nodes, and **LoadBalancer** requests an external endpoint through the configured provider.
- **NetworkPolicy** selects Pods and allows ingress or egress by peer and port. Enforcement requires a supporting network plugin. Selected traffic is isolated in the specified direction; applicable allow rules combine additively.
- For an **NSX-backed load balancer**, Edge capacity and form factor affect available scale. Size against connection and throughput demand as well as platform limits; Pod count alone is not a load-balancer sizing metric.

Ingress, DNS and certificates

- **Contour** is an application-layer ingress controller using **Envoy** as its data plane. It translates supported routing resources, including **HTTPProxy**, into routes to backend Services.
- **Avi Kubernetes Operator (AKO)** reconciles supported Kubernetes exposure resources with Avi load-balancing infrastructure. An ingress controller, external load balancer and application Service have separate responsibilities.
- **ExternalDNS** reconciles DNS records from configured resource sources. Services and Contour HTTPProxy resources can supply names and targets when their corresponding sources and provider permissions are configured.
- **cert-manager** adds **Certificate**, **Issuer** and **ClusterIssuer** resources. An Issuer is namespace-scoped; a ClusterIssuer can be referenced across namespaces. A Certificate requests issuance and renewal and stores key material in a Secret.
- For a **corporate CA**, configure the supported issuer integration, credentials and client trust chain. Automating renewal still requires a functioning issuer and an endpoint that reloads the renewed certificate.

Networking & Services / Storage & Recovery

VMware Certified Advanced Professional - VMware Cloud Foundation VKS

KNOWLEDGE CHEATSHEET

APPLICATION CONNECTIVITY

- **CNI** — Pod network
- **Ingress** — External routing
- **Mesh** — Service traffic
- **TLS** — Trust and encryption

Service mesh and application boundaries

- A **service mesh** adds application communication controls such as service discovery, traffic routing, identity, policy and telemetry. Its control plane configures the data plane that processes service traffic.
- **Istio mutual TLS (mTLS)** encrypts service-to-service connections and authenticates both peers using workload identities. Configure trust and authentication policy; certificate encryption alone does not define authorization.
- **Observability** combines request metrics, traces and logs. Mesh traffic policies can control routing, retries and timeouts; these features complement Kubernetes networking and application monitoring.
- In a federated mesh, a **Global Namespace** can define an application boundary across participating namespaces and clusters and apply consistent policy. It does not merge Kubernetes namespaces or automatically place workloads across regions.
- The **istioctl install** workflow installs Istio for an appropriate profile and environment. Downloading the CLI only obtains tooling; installation, workload enrollment and policy configuration are distinct steps.

04 Persistent storage and recovery

Storage policies, CSI and claims

- Create **vSphere storage policies** and make them available to the **vSphere Namespace**. Those policies govern storage consumption and backing capabilities for supported cluster and workload storage.
- A **StorageClass** describes dynamic provisioning. Its **provisioner** identifies the driver; **csi.vsphere.vmware.com** identifies the vSphere CSI provisioner. Driver-specific **parameters** supply storage configuration.
- A **PersistentVolumeClaim (PVC)** requests capacity and access properties. A **PersistentVolume (PV)** represents provisioned storage. **CSI** handles volume lifecycle and attachment; the **cloud provider implementation** handles separate cloud-integration functions.
- **ReadWriteOnce** means read-write mounting by one node, not necessarily one Pod. **ReadWriteMany** permits multiple nodes when supported. Match access mode, storage class, capacity and topology to the backing storage.

Storage & Recovery / Security & Operations

VMware Certified Advanced Professional - VMware Cloud Foundation VKS

KNOWLEDGE CHEATSHEET

PERSISTENT DATA LIFECYCLE

Policy Capabilities → **Claim** Request storage → **CSI** Provision volume → **Recovery** Snapshot and backup

Snapshots and restoration

- A **VolumeSnapshot** is a namespaced request or reference to a volume snapshot. **VolumeSnapshotContent** is the cluster-scoped representation of the underlying snapshot; **VolumeSnapshotClass** supplies driver and deletion-policy settings.
- **Dynamic snapshots** request a new backing snapshot from a PVC. **Pre-provisioned snapshots** bind to existing snapshot content using the correct driver, snapshot handle and matching object references.
- Check binding and **readyToUse** before consuming a snapshot. Restore through a new PVC referencing the snapshot as its data source, subject to the driver's supported capacity and topology constraints.
- A **volume snapshot** protects volume state, not all cluster objects. Coordinate applications when consistency matters, and distinguish storage snapshots from an independently retained backup.

Velero and workload protection

- **Velero** backs up and restores Kubernetes resources and integrates with supported volume-protection mechanisms. Configure a **BackupStorageLocation** backed by supported object storage, commonly an S3-compatible endpoint, with credentials and trust.
- Scope backups by **namespace**, resource type and **label selector**. Cluster-scoped resources need explicit consideration. An application backup must include its objects, dependencies and required persistent data.
- Volume protection may use **CSI snapshots**, a platform plugin or a file-system backup mechanism. Select the supported workflow for the target: VKS workload clusters and Supervisor-hosted vSphere Pods have different integration requirements.
- A **VM snapshot**, vCenter configuration backup or infrastructure replication does not by itself provide coordinated Kubernetes application recovery. Test restoration, namespace mapping, storage provisioning and application health against the recovery objectives.

05 Security and day-two operations

Harbor and artifact trust

- **Harbor** is a container and artifact registry with projects, role-based access, vulnerability scanning and policy controls. Separate tenant repositories and grant push or pull permissions according to operational responsibility.
- **Vulnerability scanning** identifies known issues; **image signing** supplies evidence of origin and integrity when verified. Neither patches an image. Rebuild affected images and enforce the chosen deployment policy.

Security & Operations

VMware Certified Advanced Professional - VMware Cloud Foundation VKS

KNOWLEDGE CHEATSHEET

OPERATE WITH EVIDENCE

- **Harbor** — Artifact trust
- **Metrics** — Capacity signals
- **Audit** — API activity
- **Events and logs** — Diagnosis

- For **private image pulls**, configure the correct registry address, repository permissions, trusted CA and **imagePullSecrets** where required. Pinning an image digest makes the selected artifact explicit.
- Before removing an optional **Harbor service**, protect required images, metadata and persistent data and identify consumers. Uninstall service instances through the supported lifecycle before removing an unused registered definition.

Metrics, scaling and audit evidence

- **Prometheus** scrapes and stores metrics; **Grafana** visualizes them. Track node capacity, Pod requests, saturation, errors and latency to distinguish resource pressure from application defects.
- **Horizontal Pod Autoscaler (HPA)** changes workload replica counts from metrics. **Cluster Autoscaler** changes node capacity for scheduling demand within configured bounds. Accurate requests, compatible node pools and namespace capacity are prerequisites for useful scaling.
- For bursts, identify whether the bottleneck is **replica count**, node capacity, a quota or a downstream dependency. Adding nodes does not increase a fixed Deployment's replicas, and a load balancer does not create application capacity.
- **API audit logging** records API request activity according to audit policy, including identity, action, target and outcome. **ENABLE_AUDIT_LOGGING**, where exposed by the cluster configuration, enables this capability; log forwarding and application STDOUT collection are separate controls.

Diagnose from status and events

- Use **kubectl get** for status, **kubectl describe** for conditions and events, and **kubectl logs** for container STDOUT and STDERR. For **CrashLoopBackOff**, inspect exit reasons and use **logs --previous** for the prior container instance; select the container with **-c**.
- For API discovery errors, verify the **endpoint**, network path, certificate trust, authentication and current context. Renew the supported login when credentials expire. A YAML parsing error instead calls for checking syntax and indentation.
- For a Pending Pod with persistent storage, inspect **PVC binding**, events, access modes, policy availability, quotas and topology. Successful service installation does not prove that every dependent database volume is ready.
- For a failed cluster upgrade or a service stuck in **Configuring**, inspect controller conditions, package reconciliation and logs. Check registry access, compatibility and certificates; AppPlatform-to-vCenter authentication can be a cause when its specific errors are present.