Domain 8 Cloud Workload Security

- Cloud Workload Security key differences:
  - **Dynamic and Expansive** / **Complexity and Diversity** / **New types of workloads**.
- **Types of Cloud Workloads**:
  - Virtual Machines (VMs): The security of guest OS is the CSC's responsibility.
  - Containers: **Security** relies on OS-level controls and container image management.
  - FaaS: run without infrastructure management. / CSP handles most security.
  - AI Workloads : Process large datasets.

Securing Virtual Machines:

- Cloud's **auto-scaling feature** enables the **use of immutable workloads**, improving efficiency and adapting to changing demand.
- Challenges and mitigation:
  - **Use secure base images:** Enforce from a managed catalog, versioned and immutable.
  - **Patch management:** Regularly update images with security patches.
  - **Minimize attack surface**: Remove unnecessary OS components and harden configurations
  - **Access controls**: Implement least privilege principles
  - **Automation**:Utilize for scanning, patching, and reporting
  - **Configuration management:** Use Infrastructure as Code (IaC)
  - **Monitoring** and logging: Centralize and track activities
  - **Network security:** Harden Secure Shell (SSH) and use host-based firewalls
  - **Secure boot:** Protect against pre-boot malware attacks
  - **Specialized security tools:** Monitor hypervisors continuously.

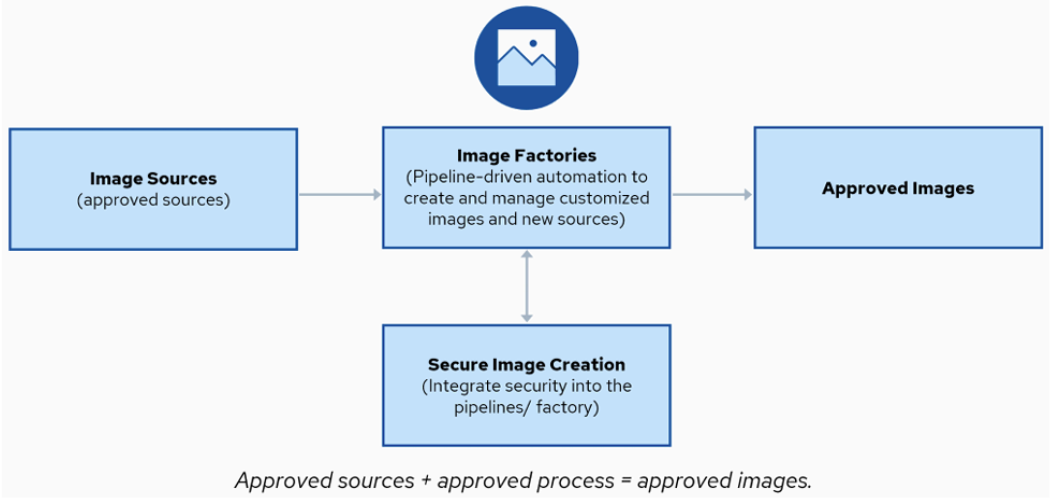

Figure 18: Secure Virtual Machine Image Creation Process

Image factories :
● Building, testing, and fine-tuning VM images to ensure consistency across deployments
● Minimizing discrepancies that could lead to security vulnerabilities
● Streamlining the integration of security updates and configuration changes.

Best practices:
● **Hardening**: To minimize potential vulnerabilities, set up VM images with only the essential software, services, and access rights.
● **Patch management:** Regularly update the VM images with the latest security improvements to protect against new threats.
● **Configuration management:** Use standardized templates and scripts to ensure all VM images meet the required security standards to automate the image creation workflow and reduce manual errors.
● **Validation and testing**: Before using a VM image, thoroughly check it for security weaknesses and operational issues to ensure it is safe and functions correctly. VM images must always come from trusted sources.

This can be done using these tools:

- **Cloud Workload Protection Platforms (CWPP)** : in-depth vulnerability scans across cloud workloads
- **Traditional vulnerability scanners**: not to be as effective in the cloud.
- **Configuration management tools**
- **Endpoint Detection and Response (EDR)**
- **SIEM**

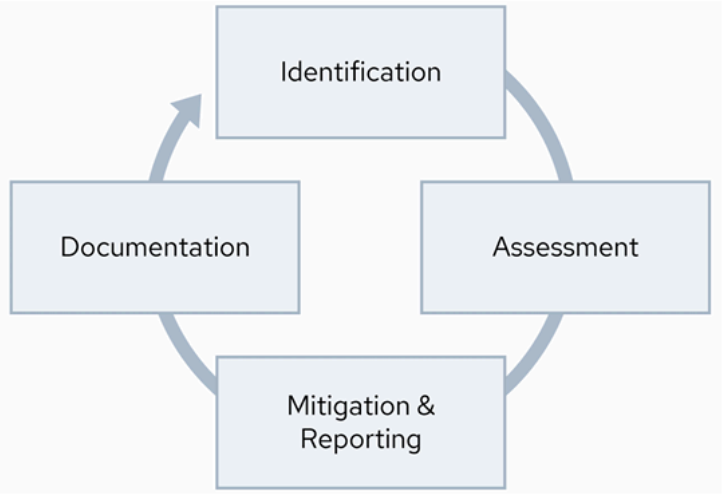

Figure 19: Vulnerability Management Lifecycle for VMs

- Snapshots are crucial for managing VM lifecycles.

Securing Containers:

- **Self-hosting Docker containers** on a VM to consuming fully managed container orchestration services **based on Kubernetes**.
- **Container Image Creation**:
  - Created from instructions in a Docker file = consistency and portability across environments.
  - Once built and deployed, they are not modified but replaced with new images for updates.
- **Container Networking**:
  - An extension of the host operating system (often Linux) networking.
  - **Kubernetes** (“**K8s**") networking, and therefore network isolation, happens on multiple levels ranging from individual containers to application-aware load balancers (e.g., Ingress Controller).
- **Container Orchestration & Management Systems**:
  - Amazon (EKS), Microsoft (AKS), and Google (GKE) have adopted Kubernetes, tailoring it to their platforms with proprietary enhancements.

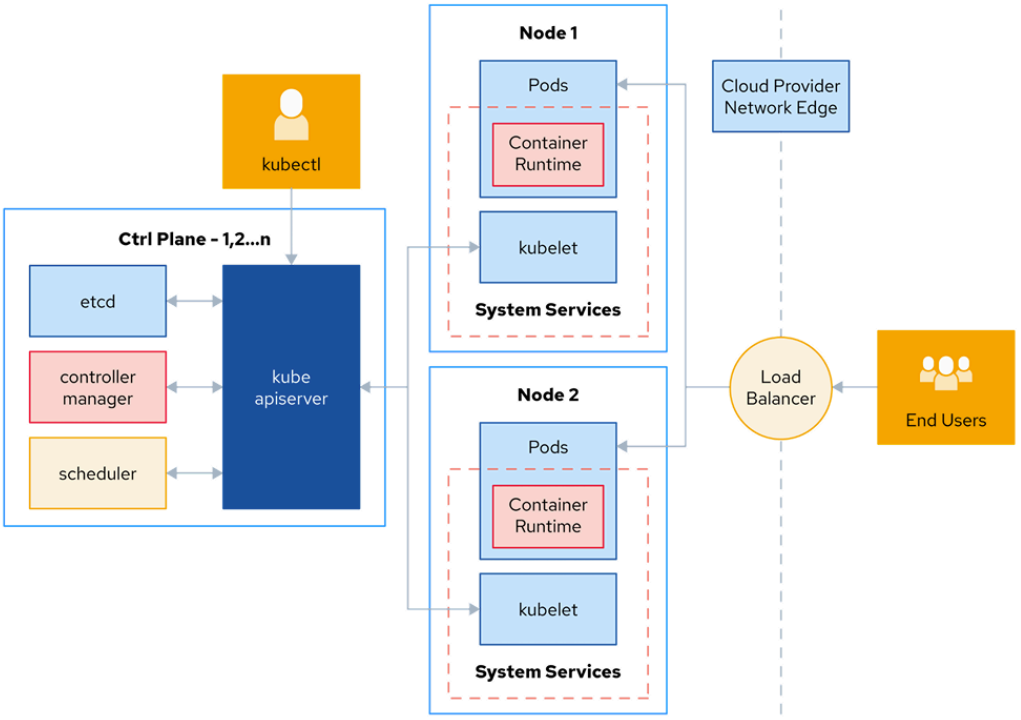

Figure 20: Basic Kubernetes Setup with Load Balancer and Pods

- **Container Orchestration Security**: Securing Kubernetes involves:
  - **Use CSP services** / **Harden services** / **Patch/update** / **Security policies** / **Benchmark san tools** (Follow Center for Internet Security (CIS) benchmarks and use standardization tools). / **Secure image repository** :Use private repositories with rRBAC, image scanning, and **signing**. / **Configuration** / **Host security**: Harden host operating systems. / **Storage security** (Encrypt data, use access controls, and monitor access) / **Network security** (Implement segmentation and firewall rules) / **Image validation**: Validate and sign images in the CI/CD pipeline.

- **Secure Artifact Repositories**:
  - Digital signatures and verification.
  - Only verified users to push or pull images.
  - Images are routinely scanned for vulnerabilities and Container images should be immutable.
  - The provenance of images is thoroughly documented and protected.
  - Audit trails.
- **Runtime Protection for Containers**.

Securing Serverless and Function as a Service:

- A way for developers to write and deploy code without handling the underlying infrastructure (focus purely on coding).
- Cloud Security Alliance (CSA) Serverless Working Group:
  - a cloud-computing execution model. /
  - Serverless computing is offered under a "**Pay as you go**"paradigm where payment is generally made on actual physical resources like central processing unit (CPU) usage time.
- **Operational simplicity**: system automatically adjusts computing resources based on the application's needs.
- subset of serverless: Functions as a Service (FaaS) : cloud consumer **defines a function** and the events through which it will be triggered.
- Single-use container that resides on a single-use virtual machine = distinctly isolated environment and preventing any interference between functions.
- Reduces the attack surface as there is no persistent operating system for the cloud consumer to manage.
- Use environment variables instead of hardcoding secrets like passwords or API keys into the code.
- Amazon Web Services (AWS) Secrets Manager or Microsoft Azure Key Vault offer reliable mechanisms for secrets management, ensuring secure storage, retrieval, and rotation of credentials. (rotating secrets minimizes the risk of compromised credentials).
- This aligns with a Zero Trust (ZT) security model, where trust is continually verified rather than assumed.

Securing AI Workloads:

1. **Model manipulation**: direct or indirect prompt injection (adversarial inputs)
2. **Data poisoning:** model becomes untrustworthy
3. **Sensitive data disclosure**
4. **Model theft : (distillation)**
5. **Model Failure / malfunctioning :** software bugs, hardware failures, hallucinations, or operational errors

6. **Insecure supply chain** : vulnerabilities introduced through third-party components, dependencies, or services integrated into the LLM
7. **Insecure apps/plugins**
8. **Denial of Service (DoS)**
9. **Loss of governance /compliance.**
   Can be extended to generative AI (GenAI) in general.

In its **Threat Taxonomy** CSA defines the following categories of Assets:

- Data Assets
- LLM Ops Environment
- Model
- Orchestrated Service
- AI Applications
  Lifecycle:
- Preparation
- Development
- Evaluation/Validation
- Deployment
- Delivery
- Service Retirement