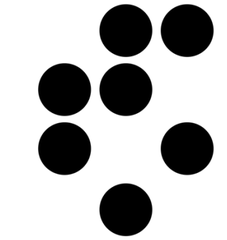
Virtualisation & Cloud Security

Barbara Krašovec



ROADMAP

What we'll cover

01 Virtualisation fundamentals

02 Cloud security foundations (public / private / hybrid)

03 Hot topics: AI, rootless VMs, IAM, secrets

04 Data security & key management

05 Cloud-native tooling & network security

06 Applications, APIs, identity & DevSecOps

Virtualisation Fundamentals

Hypervisors, VMs, images — and how each layer gets attacked

What is virtualisation?

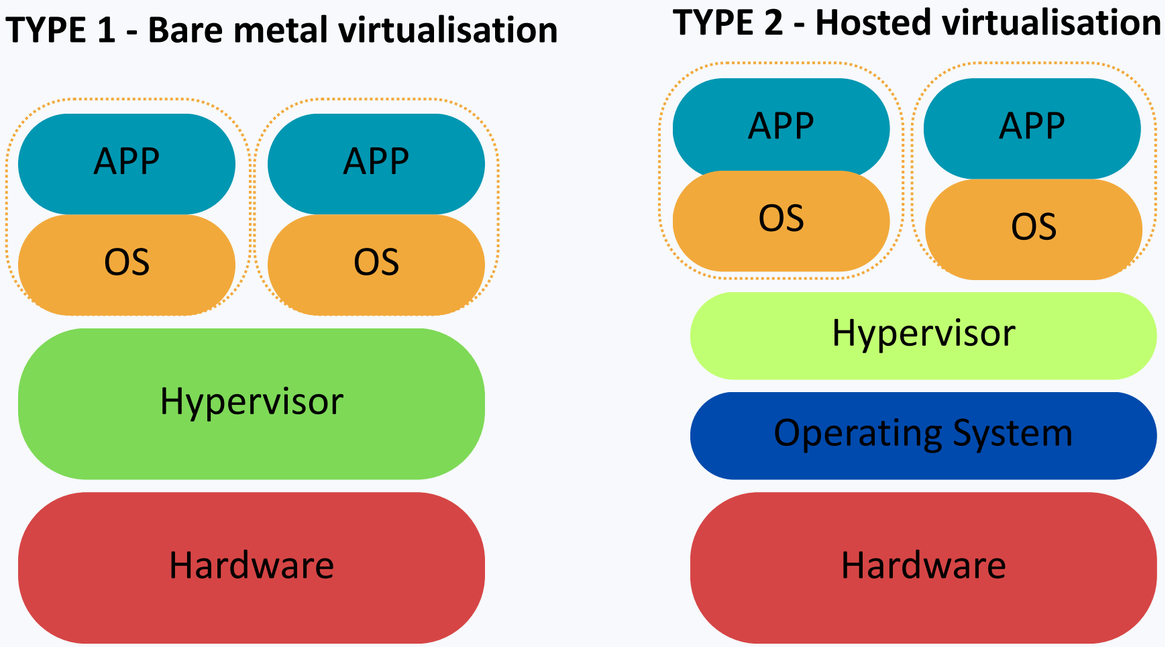
- **Virtualisation is the abstraction of physical resources**
- A hypervisor sits on top of physical hardware and abstracts CPU, memory, storage and network for multiple guests

- **Why virtualise?**
- Efficient use of resources — multiple OSs on one physical system
- Isolation of services, lower operating cost vs. one physical machine per service
- Flexibility and fast (re)deployment

The hypervisor

- **The hypervisor is the OS that provides the virtualisation environment**
- It is the main component in virtualisation — it controls access between virtual guests and host hardware

- **Once a hypervisor is compromised, the attacker owns everything on it**
- Access control and monitoring of virtualisation administrators is critical
- There is no full physical separation between guests like on physical networks — direct memory access and resource sharing remain a risk



Isolation models: VMs vs. containers/namespaces

VM ISOLATION (hardware-enforced)

- Each VM runs its own kernel — the hypervisor's device emulation + a small hypercall interface is the only shared boundary
- Compromising a guest kernel doesn't hand the attacker anything outside that guest
- Escaping to the host means breaking the hypervisor itself

CONTAINER ISOLATION (software-enforced, shared kernel)

- Every container on a node shares one kernel, namespaces (PID, net, mount, user) and cgroups create logical separation, not physical
- Attack surface = the entire host kernel syscall interface, not a narrow hypercall API
- A kernel privilege-escalation bug lets one container reach every other container and the host, there's no hypervisor boundary to stop it
- seccomp/AppArmor/SELinux narrow this, but they're mitigations layered on a shared kernel, not a substitute for a separate one

Kubernetes multi-tenancy: the network problem

"Kubernetes was designed for one organisation's own workloads sharing a cluster, not for mutually distrusting tenants."

SOFT vs. HARD MULTI-TENANCY

- **Soft**: trusted teams inside one org, isolated via RBAC + namespaces + resource quotas - fine, because nobody's actively hostile
- **Hard**: tenants who don't trust each other (e.g., a SaaS platform hosting external customers) - namespaces alone are not a security boundary here

- By default, all pods share one flat network, without NetworkPolicies (and a Container Network Interface - CNI, that actually enforces them), any pod can reach any other pod
- The shared control plane (API server, etcd) is itself a target, one tenant exploiting or DoS-ing it affects everyone on the cluster

Kubernetes multi-tenancy: how clouds solve it

"If tenants are mutually distrusting, the real security boundary is a per-tenant cluster or per-pod microVM — not a namespace."

HOW PUBLIC CLOUDS BRIDGE THE GAP

- **Dedicated node pools** (taints/tolerations): isolate tenants to specific nodes, but they still share a kernel unless pods are sandboxed
- **microVM-per-pod sandboxing**: AWS Fargate, GKE Autopilot/Cloud Run, and Kata Containers wrap each pod in a lightweight VM (Firecracker/QEMU), restoring hypervisor-grade isolation without a full cluster per tenant
- **Confidential Containers (CoCo)**: adds hardware memory encryption on top of microVM isolation, so even the provider can't inspect the workload
- **Dedicated cluster per tenant**: the real gold standard for hard multi-tenancy: separate control plane, nodes, and network; highest isolation, highest cost
- **Virtual clusters (vcluster, etc.)**: give each tenant their own control-plane view for convenience, but nodes/kernel are still shared (don't mistake this for real isolation)

VIRTUALISATION

Hypervisor security checklist

- Keep **hypervisor software and firmware updated** — apply HW firmware updates before OS installation (patch the hypervisor like your edge firewall)
- **Avoid emulated devices** the VM doesn’t need (USB, unused NIC controllers etc)
- Remove or mask services you don't need
- Use the same hardware across hosts where possible (one vendor, one patch stream, secure boot)
- Restrict access; use restrictive SSH configuration
- Do you really need a public IP?
- Use **SELinux** to confine each QEMU process, use seccomp filtering
- Audit, and layer **firewall**, VPN and security monitoring on top
- **Host-side monitoring**, not just guest-side. eBPF-based host intrusion detection or auditd on the hypervisor host catches attempted escapes even against unknown bugs
- **Track CVE pairs**, not single fixes. The Januscape/CVE-2026-46113 case example included two similar-looking shadow-paging bugs, one fix does not imply the other is covered. CVE-2026-46113 was a frame-number fix; CVE-2026-53359 escape fix.

VM security & lifecycle

HARDEN & SEGMENT

- Don't use default credentials
- Don't mix production and development VMs on the same hypervisor
- Use different networks / security groups for prod vs. dev
- Group services by risk profile on the same host
- Follow OS hardening guidelines (STIG, CIS controls)
- Harden automated installation & configuration (e.g. Ansible)

MONITOR & MAINTAIN

- Monitor all VMs — production, testing and development
- Shut down VMs you don't need; always patch offline VMs before putting them back online
- Maintain a full inventory of VMs — track OS, software, users, images
- Provide security monitoring (e.g. Prometheus) and implement vulnerability management
- Check regularly for open ports, default passwords, unpatched software

VM/container attack techniques

- **Hypervisor breakout - VM/container escape**
  Exploiting a software vulnerability to break out of a guest — rare, but real (classic examples: CVE-2008-0923, CVE-2021-22960, CVE-2022-20779, CVE-2023-20867, CVE-2026-47876, CVE-2026-53359)

- **Hyperjacking & VM hyper-jumping**
  Exploiting the host/hypervisor to attack other guests (e.g. CVE-2020-8554)

- **Side-channel attacks**
  Spectre / Meltdown-class flaws (CVE-2017-5715, CVE-2018-3646) — unintended side effects of shared CPU execution; largely the provider's responsibility

- **Misconfiguration, code execution & outdated hypervisor package**

VM/container attack examples

Docker "CopyEscape": CVE-2026-17106

On the container side, this vulnerability severely breaks container-to-host isolation boundaries using built-in command mechanisms.

- **The Flaw**: An arbitrary file-write vulnerability found within the native docker cp command.
- **The Impact**: A compromised or malicious container can manipulate a standard file copy routine to break out of its sandbox and write or overwrite files anywhere on the host machine. If the user running the command has administrative rights, an attacker can overwrite host binaries or shell configurations to gain remote code execution on the host machine.
- **Mitigation**: Fixed in Docker Desktop or Docker Engine 29.7.2 or later.

Januscape: CVE-2026-53359

16-year-old flaw hidden deep inside the Linux kernel's KVM subsystem, affecting both Intel and AMD systems.

- **The Flaw**: A use-after-free vulnerability tied to the shadow MMU (Memory Management Unit) paging logic.
- **The Impact**: If a host system has nested virtualisation enabled, a root user inside a guest VM can exploit this page-table manipulation to trigger host kernel panics, denial-of-service, or achieve full guest-to-host system escape.
- **Mitigation**: Patched across stable Linux kernel branches in July. If patching immediately is impossible, the temporary mitigation is to explicitly disable nested virtualisation (kvm_intel.nested=0 or kvm_amd.nested=0)

Securing VM images

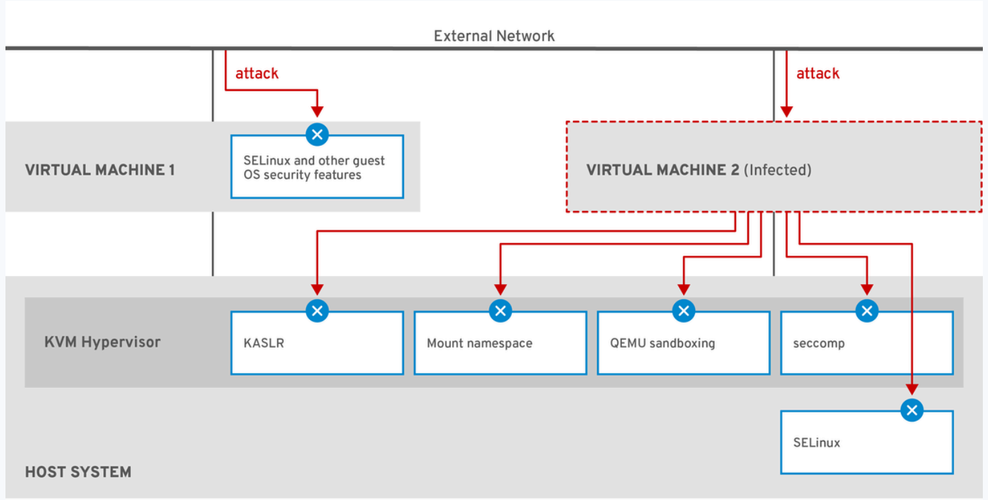
- Never store credentials, certificates or sensitive data inside an image
- Harden the software baked into the image before distribution
- Sign images, and don't use untrusted third-party images
- Scan images for known vulnerabilities before and after deployment
- Keep images updated and track them under asset management
- Create your own hardened images, build and deploy using CI/CD
- Use of private registries

Source: RedHat:

Securing VMs

Crucial VM security controls:

- **Minimal, patched hypervisor** — lean Type-1 hypervisor (KVM/Xen/Hyper-V), aggressive patching (most VM escapes exploit hypervisor bugs)
- **Namespaces & cgroups** — isolate PID/mount/network/user space and limit resources per VM/process
- **seccomp-bpf** — restrict syscalls to the minimum a guest process actually needs
- **Exploit mitigations** — ASLR/KASLR, NX/DEP, SMEP/SMAP, CFI to blunt memory-corruption bugs
- **Mandatory access control — SELinux/AppArmor** + least-privilege capabilities instead of all-or-nothing root
- **Measured/secure boot + attestation** — vTPM per VM, remote attestation to verify integrity before trust
- **Network micro-segmentation & encryption** — virtual firewalls between VMs, encrypted disks and live migration



Source: RedHat:

CLOUD

Primary attack trends and vectors

- **Identity as the Perimeter:** Over 70% to 83% of cloud breaches involve identity compromise. Threat actors routinely leverage adversary-in-the-middle (AitM) phishing and voice phishing (vishing) to bypass multi-factor authentication (MFA) or pressure help desks into credential resets.

- **Vulnerability Exploitation Surge:** unpatched third-party applications and software, with remote code execution (RCE) vulnerabilities aggressively targeted within hours of disclosure.

- **Linux Kernel & Container Escalations:** kernel flaws like CVE-2026-31431 (Copy Fail) and the Dirty Frag vulnerability family (CVE-2026-43284, CVE-2026-43500), which let unprivileged users compromise the page cache to seize root access.

- **AI and LLM Pipeline Exploits:** indirect prompt injection against Retrieval-Augmented Generation (RAG) frameworks, exposed vector databases, and unauthorised orchestration API access leading to data exfiltration.

- **Living-off-the-Cloud (LotC):** Attackers increasingly abuse native cloud management and orchestration tools (like Kubernetes components) for stealthy persistence instead of deploying traditional custom malware

- **Misconfigurations/human error**

- Source:
  - https://cloud.google.com/security/report/resources/cloud-threat-horizons-report-h1-2026
  - https://www.sentinelone.com/cybersecurity-101/cloud-security/cloud-security-statistics/

BRIDGE

Virtualisation vs. cloud

VIRTUALISATION

- A technology: creates multiple environments from a single physical system
- The ability to emulate hardware using software
- Can exist entirely on-prem, with no "cloud" involved at all

CLOUD

- An environment / operating model, not a single technology
- Can include bare metal, virtualisation, or container platforms
- Adds on-demand self-service, elasticity and a shared-responsibility contract with a provider

03

Cloud Security Foundations

Service models, deployment models, zero trust and shared responsibility

Running services in the cloud

- What are your risks — and what are your responsibilities?
- Which domains are under your control, and which are in the hands of the cloud provider?
- Where will you store your data, and how will you transfer and use it?
- Are there regulations governing where and how the data can be stored?

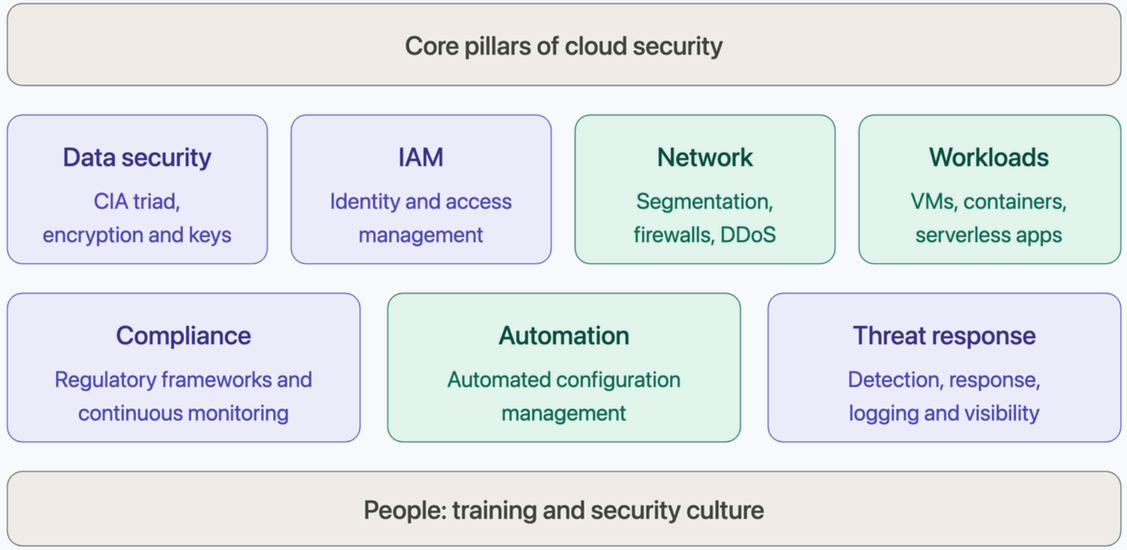
CLOUD SECURITY

What is cloud security? Core pillars

Security controls that protect cloud infrastructure, workflows and data

CORE PILLARS:

- Data security and CIA (confidentiality, integrity, availability)
- Identity and access management (IAM)
- Regulatory compliance — frameworks and continuous monitoring
- Automated configuration management
- Threat detection and response, security architecture, and people
- Network security
- Workload security



CLOUD SECURITY

Zero trust in the cloud

Zero trust assumes breach: every request is verified regardless of where it originates.

- Least-privilege access — grant only what's needed, for as long as it's needed
- Continuous verification of identities, not a one-time login check
- Network segmentation to contain lateral movement
- Device and access controls enforced at every boundary, not just the perimeter

Zero trust in the cloud is a security model built on one core idea: never trust, always verify — regardless of whether a request comes from inside or outside your network perimeter.

Cloud service models

THE THREE MAIN MODELS

- **IaaS — Infrastructure as a Service**
- Raw virtual machines — you manage the OS up
- **PaaS — Platform as a Service**
- Managed databases, Kubernetes — you manage the app
- **SaaS — Software as a Service**
- Ready-to-use applications — you manage your data & users

AND THE GROWING XAAS FAMILY

- CaaS — Container as a Service
- FaaS — Function as a Service
- NaaS — Network as a Service
- DSaaS — Data Storage as a Service
- The more "as-a-Service" layers you consume, the less infrastructure you control — and the more your security depends on the provider's IAM and configuration

CLOUD DEPLOYMENT MODELS

Public vs. private vs. hybrid cloud

PUBLIC CLOUD

- Shared multi-tenant infrastructure owned by a third-party provider (AWS, Azure, GCP)
- No physical/hardware access — trust the provider's controls
- Elastic, pay-as-you-go, fast to provision
- Attack surface: identity, APIs, misconfiguration
- Provider secures the hypervisor; customer secures IAM, data, workloads

PRIVATE CLOUD

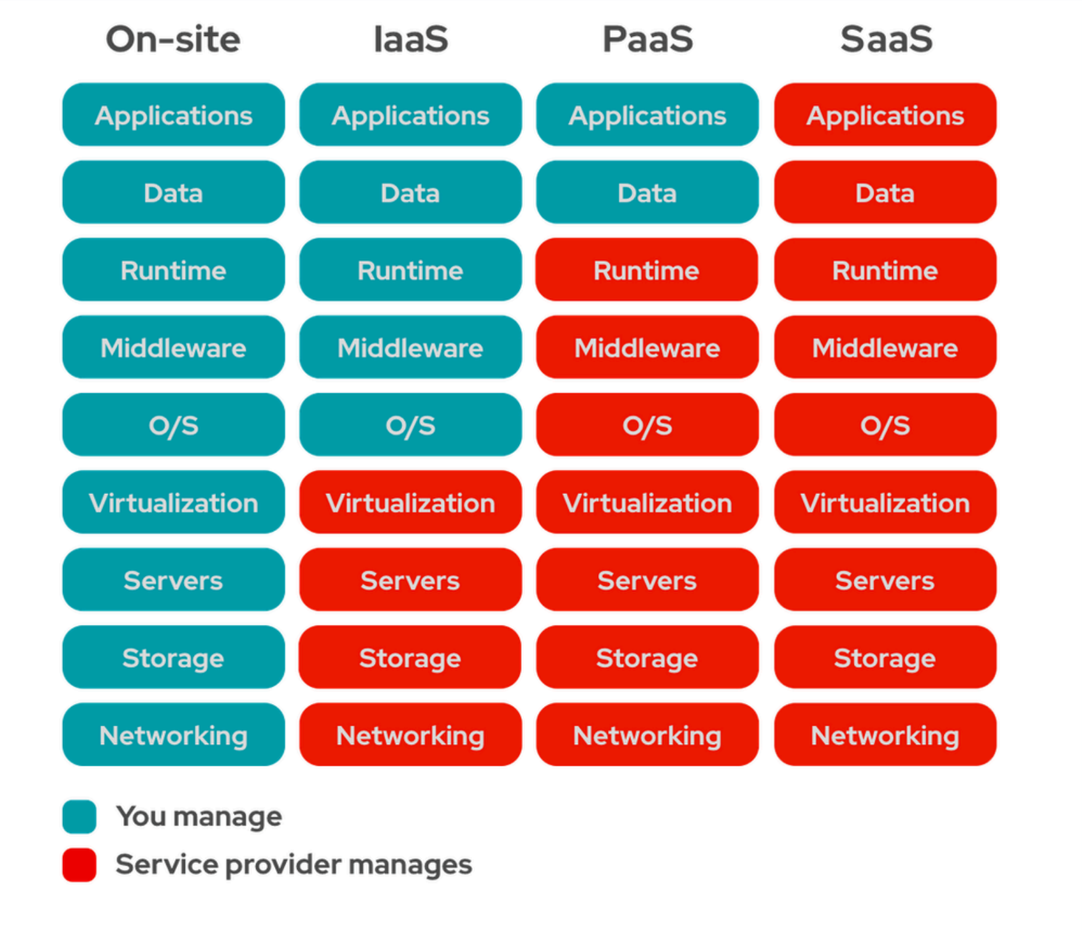
- Dedicated infrastructure, on-prem or hosted, single organisation
- Full control of hypervisor, network design and hardware
- Slower to scale, higher CAPEX, needs in-house expertise
- Attack surface: hypervisor, physical access, internal network design
- Organisation owns the entire stack — and the entire responsibility

HYBRID CLOUD

- Mix of public + private, workloads move between them
- Sensitive data/workloads stay private; burst capacity goes public
- Best of both — but doubles the operational complexity
- Attack surface: the connective tissue (VPN, interconnects, federation)
- Consistent policy across environments is the hard part

Shared responsibility, public vs. private

- **Public cloud**
  - Provider secures the hypervisor, physical hosts and facilities; customer secures IAM, data, OS patching inside the VM, and configuration — most breaches originate on the customer's side of the line
- **Private cloud**
  - The organisation owns the entire stack — no provider to share the burden with, but also no shared blind spots. Full control, full accountability, including physical security
- **Hybrid cloud**
  - Responsibility boundaries must be mapped per-workload; the interconnect between environments is itself part of the attack surface



Cloud security challenges

FOR THE CUSTOMER

- No access to the hypervisor or hardware — physical/host security is gone
- No control over which other tenants share the same host, or how well they protect their VMs
- Users often end up acting as sysadmins for services they don't fully control
- Complex environment → high chance of misconfiguration and human error
- Insufficient IAM, insecure APIs, data loss from outages or accidental deletion

FOR THE PROVIDER

- Complex network designs at massive scale
- No control over the state of customer VMs and services running on them
- Must defend a constantly evolving threat landscape across every tenant simultaneously
- Balancing default-secure settings against customer flexibility

Cloud risks

- **Misconfigurations** (human error) — still the #1 root cause of cloud incidents
- **Leaked credentials/API keys**
- **Insider threats**, especially privileged users; weak identity management; leaked credentials
- **API vulnerabilities** and insecure interfaces
- **Compliance failures**, encryption problems, malware
- **Vulnerable/unpatched VM images or dependencies**
- **Supply chain**
- **Data-related risks:** exfiltration, data in the wrong region, weak access controls

The cloud is the most-targeted environment today, and most "conventional" security tools are insufficient for it

How to prevent them

- Align cloud security with business goals; use a shift-left approach to deployment
- Automate security tooling to catch misconfigurations, vulnerabilities and identity/ permission issues
- IAM configuration is crucial — apply MFA to privileged accounts especially
- Update software regularly; encrypt data; segment networks; limit access to a few trusted locations
- Regular backups and disaster-recovery planning, plus continuous security training

05

Hot Topics in Cloud & Virtualisation Security

The five themes shaping the field right now

AI & SECURITY

AI: strengthening the cloud

WHERE AI IS HELPING

- CNAPP/CSPM* platforms now use AI to correlate millions of signals and prioritise the handful that matter
- Automated remediation — misconfigurations fixed before a human even sees the alert
- AI-assisted SOC triage cuts investigation time on routine alerts
- Anomaly detection on identity and API behaviour catches abuse of legitimate access (the exact pattern behind the Drift breach)

BUT IT'S EARLY DAYS

- AI recommendations still need human validation — false positives and false confidence both carry real cost
- Attackers adopt the same tools faster than most defenders operationalise them
- AI is a force multiplier for existing security practice, not a replacement for it

CSPM (Cloud Security Posture Management)
CNAPP (Cloud-Native Application Protection Platform)

AI & SECURITY

AI: expanding the attack surface

- **LLMjacking**
  - Attackers steal cloud credentials specifically to run LLM/GPU inference at the victim's expense
- **Agentic & "shadow" AI**
  - Autonomous AI agents connected to Slack, Google Workspace and internal systems create non-human identities with elevated privileges that traditional tools don't see. One widely-adopted open-source agent alone was linked to 20,000+ exposed instances after critical flaws were disclosed
- **AI-authored malware**
  - Shai-Hulud's own code showed signs of AI assistance — attackers are using AI to write and iterate malware faster
- **AI-accelerated intrusions**
  - As seen in the 8-minute admin-takeover case — automation compresses the attacker's timeline dramatically

HOT TOPIC

Rootless VMs, containers & confidential computing

ROOTLESS BY DEFAULT

- Podman's rootless mode and containerd 2.0's improved user-namespace support map "root" inside the container to an unprivileged host user
- If the container engine, runtime or orchestrator is compromised, the attacker still doesn't gain root on the host
- Extends to VMs too: least-privilege hypervisor management, dropped Linux capabilities (CAP_SYS_ADMIN, CAP_NET_ADMIN) by default
- Kata Containers / Confidential Containers (CoCo) run each container inside a lightweight, hardware-isolated microVM

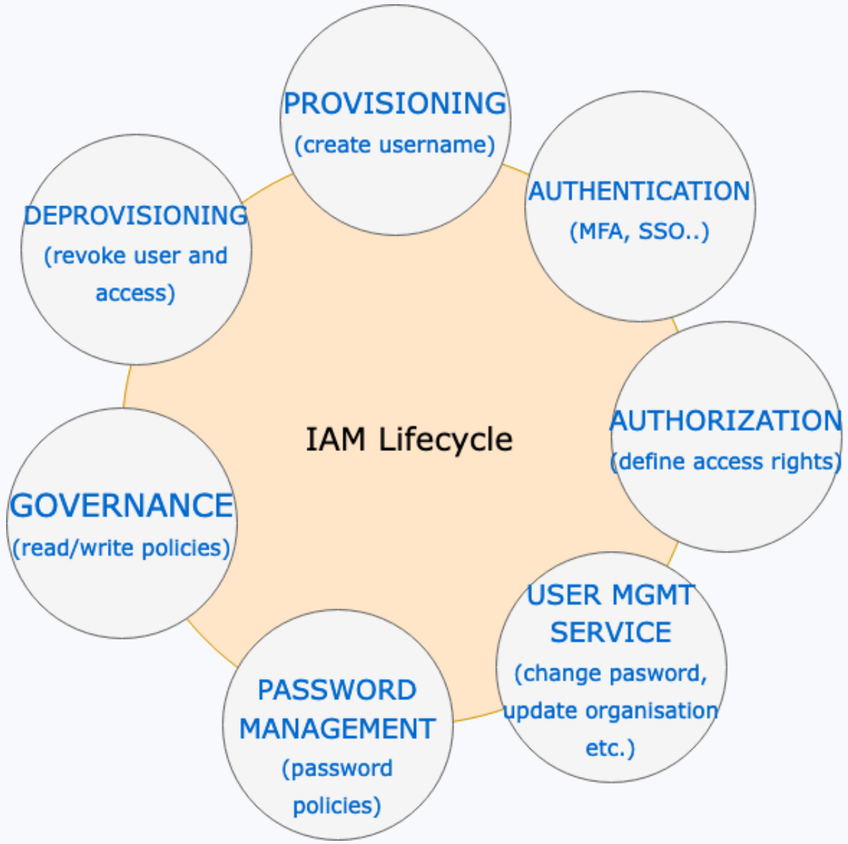
CONFIDENTIAL COMPUTING GOES MAINSTREAM

- Gartner ranks confidential computing among its top-3 technologies to watch in 2026
- Hardware-enforced trusted execution environments (Intel TDX, AMD SEV) protect data in use — even from a compromised hypervisor or cloud admin
- Now integrates with existing VM and container runtimes — no app redesign required
- Growing interest in confidential AI: running models inside a TEE so prompts and weights stay protected

HOT TOPIC

IAM: growing complexity

- **Public cloud IAM**
  - Users, groups, roles and policies defined per provider; console access needs MFA and no remote root; identity federation bridges corporate directories into the cloud without duplicating accounts
- **Private cloud IAM**
  - Typically integrates with existing enterprise directories (LDAP/AD); fewer moving parts, but the organisation owns every misconfiguration itself
- **The shared trend**
  - IAM is now consistently named the top named driver of cloud risk — and it's getting harder, not easier, as multi-cloud, federation and non-human identities pile on top of each other



HOT TOPIC

Secret stores, tokens & non-human identity

45–144×

non-human identities now outnumber human accounts across enterprise and cloud-native environments

28.6M

new hardcoded secrets found in public GitHub commits in 2025 alone — a 34% year-over-year jump

96%

of organisations still store some secrets outside a proper secrets manager — in code, configs or CI/CD tools

- Use a centralised secrets manager / vault for tokens, keys and certificates — never hardcode them
- Short-lived, scoped credentials beat long-lived static tokens; rotate aggressively
- MCP configs and AI-coding-assistant workflows are a new, fast-growing leak vector — treat them like any other secret store

Source: GitGuardian State of Secrets Sprawl 2026; Cloud Security Alliance / Entro Labs

06

Cloud Data Security

Storage, lifecycle, protection strategies and key management

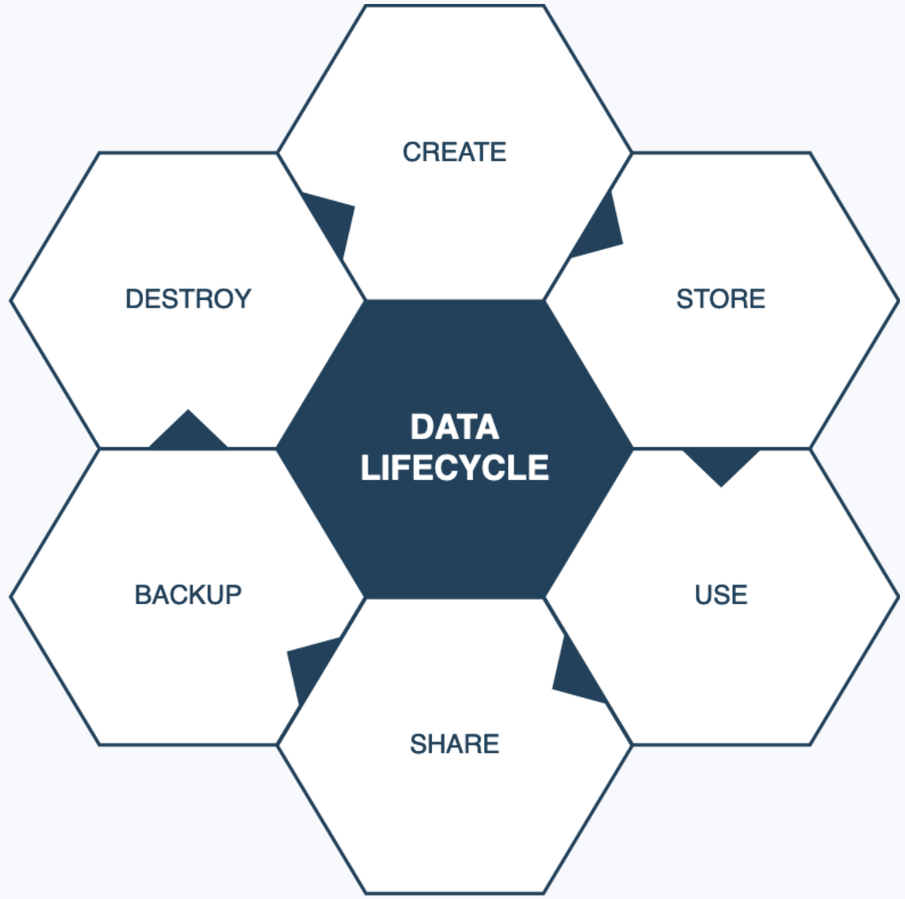
Cloud data storage

- **Object storage**
  - Like valet parking — you hand over a file, the application manages where and how it's stored (S3, Blob, etc.)
- **Block storage**
  - Traditional hard-disk model (FC, iSCSI)
- **File storage**
  - Presents itself as a filesystem (NFS, CephFS)

- One constant fact: your data will move between different physical nodes, regardless of which model you use

Cloud data lifecycle

- **Create**
  - define sensitivity; encrypt before sending externally-created data to the cloud
- **Store**
  - DB, object store, or files, secured with ACLs, encryption, auditing, redundancy and backup
- **Use**
  - access, processing and viewing require decryption; log and audit every action; use DLP
- **Share**
  - once shared, security is no longer entirely in your domain
- **Archive & Destroy**
  - move to long-term storage, then apply a clear deletion policy: who can delete data, and when



Key management in the public cloud

- **2021**
  - Some providers offered HSM (Hardware Security Module) — expensive — or a KMS connected to an external HSM
- **2025/26**
  - Every major public cloud provider now offers HSM-backed key management as a standard option

The bar has moved: hardware-backed key protection is now table stakes, not a premium add-on

KSM = key management service, is a cloud service for creating, storing, controlling, and using cryptographic keys
HSM = hardware security module, specialized cryptographic hardware

Key management

CHALLENGES

- Access to the keys, and their replication/ backup
- Key storage and full lifecycle management
- Integrating key stores with other systems is rarely trivial
- No keys = no data — keys should never leave a trusted environment
- Public cloud software-based keystores often don't meet NIST physical-security requirements on their own

BEST PRACTICE

- Use a centralised Key Management Solution (KMS)
- Enforce strong access controls
- Practice regular key rotation
- Audit every access to the keys
- Maintain a tested backup/recovery plan

Data security strategies

PROTECT THE DATA

- Restrict access to data; enforce MFA
- **Tokenisation** — replace sensitive data with a randomly generated token
- **Encrypt** data at rest, in transit and in use
- **Obscure** data via masking, obfuscation and anonymisation
- **Backup your data**

WATCH THE DATA

- **Key management:** creation, distribution, storage, recovery and revocation
  - Where keys are stored materially changes the overall risk
- **Monitor data**-related activity and detect unusual events
- Use a SIEM to collect, manage, analyse and display log data

DATA SECURITY

SaaS Backup: The Shared-Responsibility Blind Spot

MISCONCEPTION

- Most SaaS providers (Microsoft 365, Google Workspace, Salesforce, GitHub) guarantee **infrastructure availability, not your data's recoverability** — that split is explicit in most providers' own terms of service
- Native **"recycle bin" / version history** has short retention windows (often 30–90 days) and doesn't protect against admin-level or automated deletion at scale
- **Ransomware** increasingly targets SaaS data directly (mass file encryption via a compromised OAuth app, mass mailbox deletion) — no server to "restore from an image" in the traditional sense

TO COVER THE GAP

- **Third-party SaaS backup tools** (Veeam, Druva, Rewind, or provider-native options like Google Vault) — independent, provider-agnostic copies, not just longer retention
- **Test restores, not just backups** — an untested SaaS backup is a hope, not a plan
- **Immutable/air-gapped backup copies** — so a compromised admin account can't delete the backup along with the primary data
- **Define recovery point/time objectives (**RPO/RTO) per SaaS app: For each SaaS application, decide how much data you can afford to lose and how quickly the application needs to be restored after a failure.

07

Cloud-Native Tooling & Network Security

CSPM, CIEM, CWPP, CNAPP, CDR — and public vs. private network design

SOLUTIONS

Cloud-native security tooling

CSPM & CIEM

- **CSPM — Cloud Security Posture Management**
  - Continuous monitoring for misconfigurations, vulnerabilities and compliance; can auto-remediate (Wiz, Prisma Cloud, Trivy, Sysdig)
- **CIEM — Cloud Infrastructure Entitlement Management**
  - Analyses user/service access, enforces least privilege, right-sizes permissions and flags inactive identities

CWPP & CDR

- **CWPP — Cloud Workload Protection Platform**
  - Continuous monitoring across VMs, containers and serverless functions; detects malware and unauthorised access
- **CDR — Cloud Detection & Response**
  - Real-time behavioural analytics for credential theft or lateral movement; can auto-isolate compromised workloads (Wazuh, Trivy, Falco)

SOLUTIONS

CNAPP — tying it together

Cloud-Native Application Protection Platform

- Protects cloud-native applications and infrastructure across the whole lifecycle — code to runtime
- Bundles CSPM, CIEM and CWPP capabilities into a single platform
- Real-time threat detection and automated response
- Active monitoring focused on what's actually exploitable in production, not just theoretically vulnerable
- Examples: Wiz, Orca, ThreatMapper

SOLUTIONS

Securing Kubernetes & Container Orchestration

ATTACK SURFACE

- **Exposed/misconfigured API server or kubelet —** anonymous access left enabled, dashboards reachable from the internet with no auth
- **Overly permissive RBAC —** cluster-admin bound too broadly, default service-account tokens auto-mounted into every pod
- **Privileged/host-breakout pods —** privileged: true, hostPath, hostNetwork, or hostPID let a compromised pod reach the node
- **Supply chain** — untrusted images from public registries, no signing or provenance checks, :latest tags in production
- **Secrets exposure** — etcd unencrypted at rest, secrets mounted as plain env vars instead of a secrets manager
- **Flat pod network** — no NetworkPolicy means one compromised pod can reach everything else **in the cluster**

CORE CONTROLS

- **Least-privilege RBAC** — avoid cluster-admin, audit role bindings regularly, disable auto-mounted service-account tokens where not needed
- **Pod Security Admission** — enforce restricted/baseline profiles (the built-in successor to deprecated PodSecurityPolicy)
- **Default-deny NetworkPolicies (Calico, Cilium) —** explicit allow-lists between namespaces/services
- **Policy-as-code admission control** — OPA/Gatekeeper or Kyverno to block non-compliant manifests at deploy time, not after
- **Image security** — signed images (cosign/sigstore), scanning in CI, private registries only
- **Runtime detection** — Falco or equivalent for anomalous syscalls/exec activity inside pods
- **Encrypt etcd at rest; integrate secrets with Vault or your cloud KMS instead of native Secrets objects**
- **Harden the control plane and kubelet** — disable anonymous auth, keep patched, enable audit logging

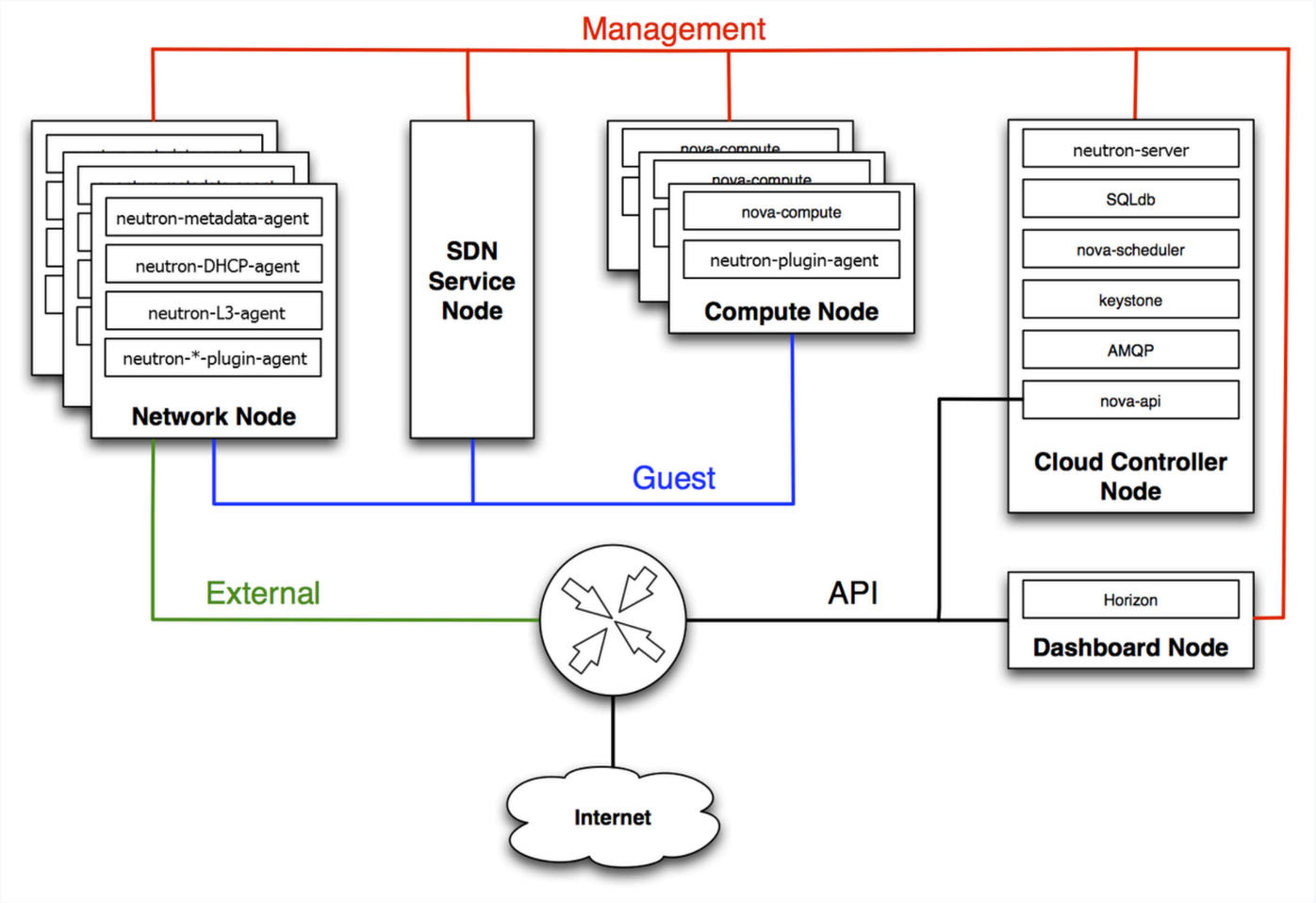
https://www.cisecurity.org/benchmark/kubernetes
https://www.cisa.gov/news-events/alerts/2021/08/02/cisa-and-nsa-release-kubernetes-hardening-guidance

Private cloud network security — OpenStack example

- **In a private cloud, network design is entirely in the organisation's hands — and needs careful planning**
- OpenStack uses SDN, which adds complexity to both the physical and virtual network design
- Which networks need to be physical and which VLAN?

COMMON OPENSTACK NETWORKS:

- **API network**
  - Used to access APIs, reachable from the internet
- **Management network**
  - Communication between OpenStack components (e.g. databases); not routed in or out
- **Guest/tenant network**
  - VM-to-VM data communication within the deployment
- **External/public network**
  - Reachable from the internet



Fundamental network security measures

SEGMENT & FILTER

- Network isolation / segmentation
- Firewalls — whitelisting between zones, perimeter and internal segmentation
- ACLs — rules per network; security groups — rules per instance
- IDS/IPS and continuous network monitoring

Old approach:

- *“Which IP address is allowed to talk to which IP address?”*

MODERN CONSTRUCTS

- Virtual Cloud Servers provide IaaS-style networking in the public cloud
- Service meshes (AWS App Mesh, Istio, Consul) enforce service-to-service policy and authentication
- These constructs matter most where segmentation alone can't keep up with dynamic, ephemeral workloads

Modern approach:

- *“Which application/service is allowed to talk to which other application/service, under what conditions?”*

Public cloud network security

- New infrastructure can be added instantly by anyone — autoscaling and serverless make it hard to maintain overall control
- The customer shares network-security responsibility with the provider — but the split is easy to misunderstand
- Baseline: educate dev-ops and users, and clearly establish who owns which aspect of security
- Use a SIEM or dedicated threat-detection solution
- Partition with VPCs and subnetting — smaller segments limit how far an attacker can move laterally

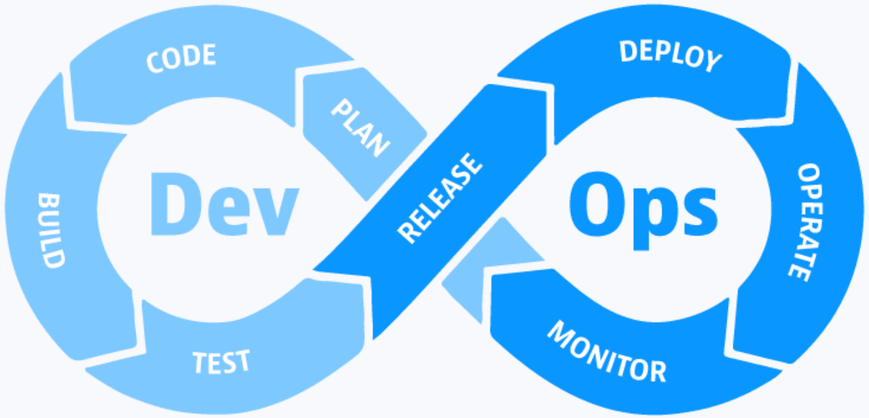
08

Applications, APIs, Identity & DevSecOps

Securing the software layer, its interfaces, and the people who ship it

Shift-left: security across the dev cycle

- **DevSecOps builds security into every step of development, before release**
- Unit tests for cross-browser, API and integration coverage; optimised, temporary test environments
- Functional tests and continuous code-quality checks
- Comply with OWASP secure-coding standards; apply automatic vulnerability scanning
- Provide documentation checked by CI, and a Software Bill of Materials (SBOM)



Secure Coding Practice Quick-reference Guide

1. Introduction
2. Checklist
2.1 Input validation
2.2 Output encoding
2.3 Authentication and password management
2.4 Session management
2.5 Access control
2.6 Cryptographic practices
2.7 Error handling and logging
2.8 Data protection
2.9 Communication security
2.10 System configuration
2.11 Database security
2.12 File management
2.13 Memory management
2.14 General coding practices

OWASP®

DevSecOps toolchain

- **SAST**
  - Static Application Security Testing: scans source code for vulnerabilities
- **DAST**
  - Dynamic Application Security Testing: black-box attacks against a running app
- **IAST**
  - Interactive testing: a hybrid of manual and automated SAST + DAST
- **RASP**
  - Runtime Application Self-Protection, blocks attacks live (OpenRASP, Sqreen)
- **Secret detection**
- **Dependency scanning** (WhiteSource, FOSSA),
- **Application hardening** (WAF)
- **Continuous security monitoring**

APIs: private cloud vs. public cloud

PRIVATE CLOUD — OPENSTACK

- OpenStack exposes both private and public APIs — by default, all are public, but they don't have to be
- Specify allowed API operations explicitly
- Isolate API endpoints on separate hosts; use SELinux, host-based firewall rules and network ACLs/IDS
- Use a reverse proxy and WAF in front of REST endpoints

PUBLIC CLOUD — AWS EXAMPLE

- In AWS, all APIs are public by default
- Service Control Policies limit what APIs can do — whitelists, regional limits, and more
- API Gateway protects against malicious users and traffic bursts
- Protect with SSL certificates, a WAF, throttling targets, and VPC-only access where possible

IDENTITY

Authentication, authorization & token lifecycle

- **Authentication (identity) vs authorisation (access) — two different questions**
  - Traditional IT: identity means email, VPN and service access, all tied together
  - Cloud: deleting an identity stops login — but long-lived authentication tokens can keep working even after the identity behind them is gone
  - This is exactly the gap non-human-identity sprawl exploits — token lifecycle management is now a first-class IAM problem, not an afterthought

Infrastructure security in the public cloud

- **Regions** (physical location) and availability zones (independent power, cooling, physical security)
- Protecting the network means applying zero trust across both network and account boundaries
- At scale, connectivity spans thousands of VPCs, accounts and on-prem networks — hub constructs like AWS Transit Gateway route and control that traffic
- Infrastructure as Code (Terraform, CloudFormation, Bicep) makes resource creation repeatable and auditable
- Define hardening, minimisation and patching as code — including OS-level authentication and access levels

PUBLIC CLOUD

Serverless computing

- Removes the need to manage resources or infrastructure — provider handles provisioning, scaling and maintenance
- You pay only for resources used during execution — suits short-lived tasks like image resizing or IoT data processing
- **FaaS (Function as a Service)** and **BaaS (Backend as a Service)** are the two common flavours
- Shared security responsibility still applies — and a large number of dispersed functions is genuinely hard to monitor and trace

***Serverless**: "I don't want to manage servers."*
***FaaS**: "I want to deploy individual pieces of code that run when triggered."*
***BaaS**: "I want the cloud provider to give me ready-made backend capabilities instead of building them myself."*

PUBLIC CLOUD

Incident Response in the Cloud: Why It's Different

- **Evidence disappears fast** — auto-scaling terminates the compromised instance before you can collect the evidence; ephemeral containers leave no forensic artefact once the pod is gone
- **You don't own the infrastructure** — no physical access, limited log retention windows set by the provider's default tier, and pulling deeper logs (e.g., AWS CloudTrail data events, GCP Access Transparency) often needs to be enabled before the incident, not after
- **Blast radius is identity, not network** — a stolen token can act from anywhere; "isolate the segment" doesn't work the way it does on-prem
- **Multi-tenant constraints** — you can't just seize a shared host, containment means revoking credentials/roles, not powering off hardware

HOW TO ADDRESS THESE CHALLENGES:

- **Pre-authorise logging**: enable CloudTrail/Activity Logs/Audit Logs by default, with a retention period long enough to investigate (90+ days), shipped to storage you control
- **Know your provider's IR process** — who do you call, what's the SLA, what will they hand over
- **Snapshot-before-terminate policy** for any suspicious instance/volume, baked into automation
- **Tabletop exercises** specific to cloud scenarios — practice a token-theft/OAuth-abuse scenario

CASE STUDY

AWS: security tools & IAM

SECURITY TOOLS

- Avoid root, enforce MFA, rotate access keys every 90 days or less
- Automate resource builds and configuration; use a multilayered network, back up regularly
- **AWS has a lot of security tools:**
  - **AWS Config:** asses, and audit of configuration of AWS resources
  - **AWS CloudTrail:** checks changes, records AWS API calls
  - **AWS Config:** configuration management of supported AWS resources in your account
  - **AWS CloudWatch:** applications monitoring (visualisation of metrics, logs, events)
  - **AWS Guard Duty:** threat detection system
  - **AWS Shield:** DDos protection
  - **AWS Identity Manager**

IAM & IDENTITY

- Users, groups, roles and allow/deny policies per service
- Cross-account access and temporary credentials (e.g. scoped S3 access)
- Identity federation brokers corporate users into AWS without duplicating accounts
- **Amazon Macie** (ML-based sensitive-data discovery)
- **Cognito** (SSO / social IdP for web apps)

WRAP-UP

Trends for 2027

- Zero trust remains the security model of the present and the future
- DevSecOps: security embedded across production and development
- Automated security and AI integration on both defence and offence
- Growing IAM complexity, and a sharp rise in CSPM/CNAPP tool adoption
- Multi-cloud security consolidation, and confidential computing moving into the mainstream
- Non-human identity and secrets governance becomes its own discipline

QUESTIONS?

Barbara Krašovec