DIFO432180: Digital Forensic
Week 8: Cloud-Native Artifact Collection for Docker-based Applications

Build-time, Runtime, Orchestration, Cloud, and Compliance

PhD. Nguyen Ngoc Tu

October 9, 2025

Outline

Purpose of Artifact Collection

- Preserve **high-value** evidence across build → ship → run.
- Cover **all planes**: workload (containers), orchestration (K8s), cloud control plane, and network.
- Ensure **integrity**: hash all artifacts, record tool versions, UTC timestamps, and custodians.
- Minimize **perturbation**: prefer exports, snapshots, and **read-only** mounts.
- Deliver **outcomes**: reproducible timelines, root cause, blast radius, and containment steps.

Build-time: What to Collect (1/2)

- **Dockerfiles**: include `ARG`/`ENV`, multistage lineage, and base image digests.
- **Build logs**: CI system and local logs; cache provenance; compiler flags and build args.
- **Image identity**: repository, tag, and **immutable digest** (`sha256:...`).

Thuật ngữ: Immutable digest (sha256) — Băm bất biến; Build args — Tham số build; Cache provenance — Nguồn gốc bộ nhớ đệm; Dockerfile — Tệp mô tả dựng ảnh

Build-time: What to Collect (2/2)

- **SBOMs**: SPDX or CycloneDX formats; generated from the final image.
- **Attestations**: SLSA/Sigstore (Cosign) provenance and signatures.
- **Secret exposure audit**: hard-coded keys, `.env`, build args, and layer history.

Thuật ngữ: SBOM (Software Bill of Materials) — Danh mục thành phần phần mềm; Attestation (Sigstore/Cosign provenance) — Bản xác thực nguồn gốc; Secret exposure — Rò rỉ bí mật; .env (environment file) — Tệp biến môi trường; Layer history — Lịch sử lớp ảnh; SPDX — Software Package Data Exchange

Build-time: Collection Commands (Examples)

Scenario (build pipeline context). A service image is built in CI and pushed to the registry. We need a verifiable record of *how* it was built and *what* was shipped. Run the prep steps *before* the build; run SBOM/signature checks *after* the image is pushed and you have its immutable digest.

When to run

- **Pre-build:** copy `Dockerfile`, capture repo state, archive CI workflows.
- **Post-build:** record image digest, generate SBOM, verify signatures/attestations.

```
# From local repo / CI workspace (pre-build)
cp Dockerfile evidence/
docker images --digests --no-trunc > image_index.txt

# SBOM (CycloneDX) + signature verification (post-build, use DIGEST)
syft myrepo/app:1.2.3 -o cyclonedx-json > sbom.cdx.json
cosign verify myrepo/app@sha256:<digest> > cosign_verify.txt

# Export CI pipeline definitions (pre-build)
tar -C .github -cpf github_workflows.tar workflows # GitHub Actions
tar -C . -cpf gitlab_ci.tar .gitlab # GitLab CI
```

Runtime: What to Collect (1/2)

Why runtime artifacts matter in DFIR. They reveal *what actually executed* at incident time: attacker TTPs, data access, and command paths—evidence you won"t see from build-time alone.

- **Container logs** (stdout/stderr), supervisor logs, rotated chunks.
  DFIR value: reconstruct actions and errors; align events to UTC; spot suspicious retries/outbound calls.
- **Process view**: `ps`/`top` per namespace; env/config; entrypoint/cmd.
  DFIR value: attribute rogue processes, abused env vars/API keys, and living-off-the-land binaries.
- **RW layer** (filesystem deltas), bind mounts, secret mounts.
  DFIR value: identify persistence (dropped scripts, modified configs), data staging, or credential theft.

Thuật ngữ: STDOUT/STDERR — Luồng ra/lỗi tiêu chuẩn; TTP (Tactics, Techniques, Procedures) — Chiến thuật/Kỹ thuật/Quy trình; RW layer — Lớp ghi-đọc; Namespace — Không gian tên

Runtime: What to Collect (2/2)

- **Metrics & traces**: scrape metrics endpoints; trace spans (Jaeger/OTel).
  DFIR value: timing and dependency graph; detects latency spikes around exfil/crypto-mining; links services touched by the attacker.
- **Core dumps** (targeted) & **CRIU** checkpoints (if enabled).
  DFIR value: last-known memory state for credential/artifact recovery; reproduce execution state safely offline.

Collection guidance (DFIR-safe).

Prefer *read* operations first (logs, inspect, metrics) → minimal perturbation.

Scope dumps narrowly (specific PID/container; strict timebox) to avoid impact and PII spillage.

Hash and timestamp every artifact; record tool versions and container IDs/digests.

Map evidence to identity: container ID → Pod/Task → host/node → account.

Thuật ngữ: Jaeger/OTel (OpenTelemetry) — Hệ theo dõi phân tán; Core dump — Tệp bộ nhớ lõi; CRIU (Checkpoint/Restore In Userspace) — Chụp/khôi phục tiến trình; Minimal perturbation — Tối thiểu xâm động

Runtime: Collection Scenario & Order (1/2)

Scenario. An app container is suspected of outbound data exfiltration and is still running. **Goal:** preserve *what actually executed* with *minimal perturbation* in DFIR-safe order.

Order of operations (DFIR-safe).

- Identify the container and freeze identity (`docker ps` → container ID, image digest).
- Snapshot immutable metadata (`docker inspect`) for env, mounts, entrypoint, network, timestamps.
- Pull bounded logs (`docker logs -since 24h`) to reconstruct actions in UTC.
- Snapshot runtime filesystem without execing in (`docker export`) — layerless, point-in-time view.

Thuật ngữ: Order of volatility — Thứ tự biến mất của bằng chứng; Minimal perturbation — Tối thiểu xâm động

Runtime: Commands & Notes (2/2)

```
docker ps --no-trunc > docker_ps.txt
docker inspect <cid> > <cid>.inspect.json
docker logs --since 24h <cid> > <cid>.logs.txt
docker export <cid> -o <cid>.fs.tar # layerless snapshot

# Optional / targeted memory capture (host PID of the container)
pid=$(docker inspect -f "{{.State.Pid}}" <cid>)
sudo gcore -o /ev/p$pid.core $pid
```

Notes.

- Avoid `docker exec`; record tool versions and UTC; hash outputs (`sha256sum`).
- Map identity: `CID` → host PID → node/namespace.
- Scope memory capture narrowly; timebox and document potential impact/PII.

Thuật ngữ: Layerless snapshot — Ảnh không lớp; Core dump — Tệp bộ nhớ lõi; PID (Process ID) — Mã tiến trình

Networking (SDN view): Control, Policy, State (1/3)

Goal. Reconstruct packet fate across *SDN control plane* (intents/policies) and *data plane* (actual forwarding).

- **Host firewall state**: `iptables/nft`, `conntrack` tables (NAT, states), local routes.
- **Cloud network policy**: Security Groups/NSGs (L3/4 allow-lists), NACLs, route tables, IGW/NATGW egress.
- **K8s network policy & CNI**: Calico/Cilium policies, identities, and datapath programs (eBPF/iptables).
- **L4 sockets & proc map**: `ss -tpna` with PID/command attribution.
- **K8s service graph**: Services ↔ Endpoints ↔ Pods; Ingress/Egress controllers.

Thuật ngữ: SDN (Software-Defined Networking) — Mạng định nghĩa bằng phần mềm; Security Group / NSG — Nhóm bảo mật (đám mây); NACL (Network ACL) — Danh sách kiểm soát mạng; CNI (Container Network Interface) — Giao diện mạng cho container

Networking (SDN view): Telemetry & Overlay/Underlay (2/3)

Telemetry sources.

- **Flow logs**: VPC/VNet Flow Logs (cloud fabric), CNI flow/identity logs (e.g., Hubble), mesh telemetry (mTLS/SNI).
- **Packet capture**: bounded `tcpdump` at host or pod netns; mirror/traffic capture at LB or TAP if available.
- **LB/Proxy logs**: ALB/NLB/ELB, Envoy/Istio access logs, Ingress controller logs.
- **DNS & egress**: resolver/query logs, NAT gateway translation logs, egress firewall appliances.

Overlay vs Underlay.

- *Overlay* (pod-to-pod): encapsulation/IDs (VXLAN/Geneve/eBPF) $\rightarrow$ service mesh policy, identities.
- *Underlay* (VPC/VNet): ENI/IP, SG/NSG checks, NACL/route decision, NAT/Egress path.

Thuật ngữ: Flow logs — Nhật ký luồng; Overlay/Underlay — Lớp phủ / lớp nền; mTLS (mutual TLS) — TLS hai chiều; SNI (Server Name Indication) — Chỉ báo tên máy chủ

Networking (SDN view): Attribution & DFIR Playbook (3/3)

Attribution path.

- Map **Pod/Task** → **IP/ENI** → **SG/NSG+NACL** → **LB/NAT** → **Flow log record**.
- Join with **process/socket** owners (PID, cmdline) and **K8s** metadata (namespace, labels).
- Correlate with **mesh/ingress logs** to recover HTTP targets (path/authority) even when TLS hides payload.

DFIR checklist.

- Capture host rules (`iptables`/`nft`), `conntrack`, sockets; dump K8s services/endpoints.
- Export cloud **flow logs**, **SG/NSG** config, **route tables**, **LB** access logs within the UTC window.
- Bound PCAP to seconds; avoid payload where policy/metadata suffices; hash all artifacts.
- Document policy *drift*: GitOps intent vs live SG/NSG/CNI state.

Thuật ngữ: Attribution — Quy chiếu nguồn gốc; ENI (Elastic Network Interface) — Giao diện mạng đàn hồi; Drift — Sai lệch cấu hình sống; LB (Load Balancer) — Cân bằng tải

Other Orchestrators & Runtimes (DFIR scope)

Other platforms and runtimes also appear in the wild and must be considered in evidence plans.
Orchestrators (non-K8s).

- **Docker Swarm** (Swarm mode services, tasks, overlay networks)
- **HashiCorp Nomad** (jobs, allocations; integrates with Consul)
- **AWS ECS** (clusters, services, tasks) & **Fargate** (serverless tasks)
- **Apache Mesos/Marathon** (legacy but still encountered)

Container runtimes (under any orchestrator).

- **containerd** (ctr/crictl; images, namespaces, tasks)
- **CRI-O** (K8s-focused CRI runtime; crictl/ocic)
- **dockerd/Moby** (legacy daemon & logs)
- **runc** (low-level OCI runtime), **gVisor**, **Kata** (sandboxed)

Why list them now? Your DFIR playbooks for logs, tasks, and network paths must map to the *actual* control plane & runtime in use—even when it's not Kubernetes.

Thuật ngữ: Orchestrator — Trình điều phối; Runtime — Môi trường chạy; CRI (Container Runtime Interface) — Giao diện runtime container; ECS (Elastic Container Service) — Dịch vụ Container mềm dẻo

Networking (SDN): Host Data Plane (1/4)

Goal. Baseline links, routes, filters, and connection state.

```
# Links / addrs / routes / policy routing
ip -d link show
ip addr
ip route
ip rule

# Host firewall (iptables or nftables)
iptables-save > iptables_rules.txt
nft list ruleset > nft_ruleset.txt 2>/dev/null || true

# Conntrack state (stats + sample)
conntrack -S > conntrack_stats.txt
conntrack -L -o extended | head -200 > conntrack_sample.txt

# Sockets with process owners
ss -tpna > sockets.txt
```

Thuật ngữ: SDN — Mạng định nghĩa bằng phần mềm; conntrack — Theo dõi trạng thái kết nối

Networking (SDN): Bounded Capture & Queues (2/4)

Goal. Get a short PCAP and queue stats without perturbation.

```
# UTC stamp for correlation
TZ=UTC date -u +"%Y-%m-%dT%H:%M:%SZ" | tee capture_utc.txt

# Bounded PCAP (no DNS resolution, full snaplen)
timeout 60 tcpdump -i any -nn -s 0 -U -w host_60s.pcap
# or ring buffer:
# tcpdump -i any -nn -s 0 -G 30 -W 2 -w ring_%Y%m%d%H%M%S.pcap

# NIC/queue snapshots (best-effort per IF)
for IF in $(ip -o -4 addr | awk "{print $2}" | sort -u); do
  ethtool -i $IF >> ethtool_info.txt 2>/dev/null || true
  ethtool -S $IF >> ethtool_stats.txt 2>/dev/null || true
  tc -s qdisc show dev $IF >> tc_qdisc.txt 2>/dev/null || true
done
```

Thuật ngữ: pcap — Gói tin đã bắt; tc (Traffic Control) — Điều khiển lưu lượng

Networking (SDN): Cloud Fabric (3/4)

Goal. Export SDN objects (NIC/ENI, policy, routes, flow logs).

AWS

```
aws ec2 describe-network-interfaces > aws_enis.json
aws ec2 describe-security-groups > aws_sg.json
aws ec2 describe-route-tables > aws_routes.json
aws ec2 describe-flow-logs > aws_flowlogs_cfg.json
aws elbv2 describe-load-balancers > aws_lbs.json
```

GCP

```
gcloud compute networks list --format=json > gcp_vpc.json
gcloud compute firewall-rules list --format=json > gcp_fw.json
gcloud compute routes list --format=json > gcp_routes.json
gcloud logging read "resource.type="gce_subnetwork"" --format=json \
  > gcp_vpc_flowlogs.json
```

Azure

```
az network nic list --output json > az_nics.json
az network nsg list --output json > az_nsg.json
az network route-table list --output json > az_routes.json
az network watcher flow-log show --location <region> --nsg <name> \
```

Networking (SDN): Attribution & Joins (4/4)

Goal. Tie packets to processes and cloud policy decisions.

- **Join path:** Process/Socket → IP:Port → ENI/NIC → SG/NSG + NACL/UDR → LB/NAT/Egress → FlowLog.
- **Keys to join:** local IP, container/host PID, interface name, ENI ID, 5-tuple + UTC window.
- **Minimal payload:** prefer metadata (flow logs, access logs) over deep payload captures.
- **Drift check:** compare exported SG/NSG/NACL with IaC/Git to spot unauthorized changes.

Thuật ngữ: Attribution — Quy chiếu nguồn gốc; Drift — Sai lệch cấu hình; 5-tuple — Bộ 5 tham số (src/dst IP, src/dst port, proto)

Storage in Cloud/Containers vs Physical (1/2)

Why it's different.

- **Copy-on-Write layers** (image → RW layer) rather than a single disk; container FS is *ephemeral* unless data is on a volume.
- **Volumes via CSI/PV/PVC**: storage is *abstracted & dynamic* (provisioned and detached by the platform), not a fixed drive.
- **Snapshots** are *fabric features* (EBS/PD/Azure Disk) with crash- or app-consistent options; speed and scope differ from imaging a physical disk.
- **Network/storage fabrics**: EBS/EFS/NFS/Ceph, encryption via KMS, multi-tenant isolation—policy/keys matter as much as blocks.
- **Lifecycle semantics**: K8s *reclaimPolicy* (Retain/Delete), PVC/PV binding, and rapid churn change evidence availability.

Thuật ngữ: Copy-on-Write (CoW) — Sao chép-khi-ghi; PV/PVC (Persistent Volume/Claim) — Ổ bền/phiếu yêu cầu; CSI (Container Storage Interface) — Giao diện lưu trữ cho container; Crash-/App-consistent snapshot — Ảnh chụp nhất quán khi sập/ứng dụng

DFIR Actions for Storage (2/2)

What to do (order of volatility aware).

- **Enumerate & label:** list Docker volumes, RW layers, and (if K8s) PVC↔PV↔StorageClass mappings.
- **Preserve RW layer:** export container FS (no exec) and tar volumes from the host path; avoid modifying mount options.
- **Snapshot in cloud:** use provider snapshots; **attach read-only** to a forensic instance; mount `ro`, compute hashes, and export.
- **Respect DB modes:** capture WAL/journals and record snapshot timing; if possible, quiesce app or note it's crash-consistent.
- **Record control plane:** save PVC/PV/StorageClass objects, reclaimPolicy, encryption/KMS keys/aliases, and attachment history.

Thuật ngữ: Reclaim policy (Retain/Delete) — Chính sách thu hồi; WAL (Write-Ahead Log) — Nhật ký ghi trước; Read-only attach — Gắn chỉ-đọc; Hash (SHA-256) — Băm kiểm chứng

Storage: Host & K8s (1/2)

Goal. Snapshot RW layers/volumes and capture K8s storage intent/bindings.

```
# UTC stamp
TZ=UTC date -u +"%Y-%m-%dT%H:%M:%SZ" | tee collected_at_utc.txt
# Docker: identify & snapshot
sudo docker inspect <cid> > <cid>.inspect.json
sudo docker export <cid> -o <cid>.fs.tar # note: no xattrs
sudo docker volume ls > volumes.txt
VOL=<name>; MP=$(sudo docker volume inspect -f "{{.Mountpoint}}" "$VOL")
sudo tar --xattrs -C "$MP" -cpf "${VOL}.tar" .
# If CRI (containerd/CRI-O)
sudo crictl ps -a > crictl_ps.txt 2>/dev/null || true
# K8s storage objects (+ CSI snapshots if present)
kubectl get pv,pvc,storageclass -A -o json > k8s_storage.json
kubectl get volumesnapshotclass,volumesnapshot -A -o json \
  > k8s_csi_snapshots.json 2>/dev/null || true
```

Thuật ngữ: RW layer (Read-Write layer) — Lớp ghi-đọc; CSI (Container Storage Interface) — Giao diện lưu trữ container

Storage: Cloud Snapshots & Integrity (2/2)

Goal. Use provider snapshots, attach read-only, and hash artifacts.

```
# AWS EBS snapshot (crash-consistent)
aws ec2 create-snapshot --volume-id <vol-...> --description "IR-snap" \
  > ebs_snapshot.json
aws ec2 describe-snapshots --snapshot-ids \
  $(jq -r ".SnapshotId" ebs_snapshot.json) > ebs_snapshot_desc.json
# After attaching to forensics host (examples)
sudo mkdir -p /mnt/ir
# ext4: no journal replay
sudo mount -o ro,noload /dev/xvdf1 /mnt/ir
# XFS: avoid UUID clash
# sudo mount -t xfs -o ro,nouuid /dev/xvdf1 /mnt/ir
# Integrity
sha256sum *.json *.txt *.tar 2>/dev/null | tee SHA256SUMS.txt
```

Thuật ngữ: Read-only mount — Gắn chỉ-đọc; ext4 noload — Không phát lại nhật ký; XFS nouuid — Bỏ kiểm tra UUID; SHA-256 hash — Băm kiểm chứng

Security (Supply Chain): What to Collect (1/2)

Why (DFIR). Proves *what was shipped* and whether it was *trusted*.

- **Signatures & attestations**: verify image *by digest* (Cosign/Sigstore); capture verification output and transparency log refs.
- **Provenance**: SLSA/Cosign attestations (builder, inputs, timestamps); ties image to CI run and commit.
- **SBOMs**: SPDX/CycloneDX from the *pushed image*; retain file hashes and generator version.
- **Vulnerability scans**: Grype/Trivy reports aligned to the *same digest*; record DB timestamps/severities.

Thuật ngữ: Signature (Sigstore/Cosign) — Chữ ký số; Provenance (SLSA/Cosign) — Bản xác thực nguồn gốc; SBOM (Software Bill of Materials) — Danh mục thành phần phần mềm; Vulnerability scan — Quét lỗ hổng

Security (Policies & Runtime): What to Collect (2/2)

Why (DFIR). Explains *who could do what* and *what actually tripped* during execution.

- **Admission policy logs**: Kyverno/Gatekeeper decisions (allow/deny/audit), webhook configs, policy versions.
- **Runtime alerts**: Falco/Tetragon findings (rules, PIDs, container IDs, syscall details, UTC times).
- **Secret managers (access logs)**: KMS/Secrets Manager/Key Vault requests (caller identity, key/secret IDs, result).
- **Correlate to workload**: join alerts/logs with image digest, pod/container ID, and node to attribute actions.

Thuật ngữ: Admission policy — Chính sách tiếp nhận; Runtime alert — Cảnh báo thời gian chạy; KMS (Key Management Service) — Dịch vụ quản lý khoá; Secret manager — Trình quản lý bí mật

Security: Supply Chain (1/2)

DFIR-safe: use the *immutable digest*, stamp UTC, hash outputs.

```
# UTC + image digest
TZ=UTC date -u +"%Y-%m-%dT%H:%M:%SZ" | tee collected_at_utc.txt
IMG=repo/image@sha256:<digest>
# Signatures + provenance
cosign verify "$IMG" > cosign_verify.txt 2>&1 || true
cosign verify-attestation --type slsaprovenance "$IMG" \
  > cosign_attest.txt 2>&1 || true
# SBOM + vuln scan (same digest)
syft "$IMG" -o cyclonedx-json > sbom.cdx.json
grype "$IMG" -o json > vuln.json
grype version | tee grype_version.txt
# Integrity
sha256sum *.txt *.json 2>/dev/null | tee SHA256SUMS_supply.txt
```

Thuật ngữ: Immutable digest — Băm bất biến; Attestation — Bản xác thực nguồn gốc; SBOM — Danh mục thành phần phần mềm

Security: Policies & Runtime (2/2)

DFIR-safe: collect decisions/alerts, keep UTC, and hash outputs.

```
# Admission policies (Kyverno/Gatekeeper)
kubectl get cpol,policy -A -o yaml > admission_policies.yaml 2>/dev/null || true
# Runtime alerts (Falco/Tetragon) - best effort
kubectl -n falco logs deploy/falco --since=24h \
  > falco_24h.log 2>/dev/null || true
kubectl -n kube-system logs deploy/tetragon --since=24h \
  > tetragon_24h.log 2>/dev/null || true
# Integrity
sha256sum *.yaml *.log 2>/dev/null | tee SHA256SUMS_runtime.txt
```

Thuật ngữ: Admission policy — Chính sách tiếp nhận; Runtime alert — Cảnh báo thời gian chạy

Why CI/CD & GitOps Matter (DFIR Minimal)

- **Prove intent:** Git state (HEAD, commit, diffs), pipeline defs, runner images.
- **Trace the artifact:** build ID → image digest → SBOM/signature.
- **Detect drift:** GitOps desired state vs live cluster (who/what changed).
- **Rollout history:** Helm/Argo release status, values, sync history.

Thuật ngữ: GitOps (Git + Operations) — Vận hành dựa trên Git; Drift — Sai lệch cấu hình

CI/CD: Commands (Examples)

Goal. Prove intent (Git), capture pipeline config, and snapshot deploy state.

```
# UTC stamp for chain-of-custody
TZ=UTC date -u +"%Y-%m-%dT%H:%M:%SZ" | tee collected_at_utc.txt
# --- Git intent (commit, remotes, diff) ---
git rev-parse HEAD > git_head.txt
git remote -v > git_remotes.txt
git log -1 --pretty=fuller > git_last_commit.txt
git diff --name-status HEAD~1..HEAD > git_changed_files.txt
# --- Pipeline definitions (GitHub/GitLab, best-effort) ---
[ -d .github/workflows ] && tar -C .github -cpf github_workflows.tar workflows
[ -d .gitlab ] && tar -C . -cpf gitlab_ci.tar .gitlab
# If running inside CI, capture CI env snapshot (best-effort)
env | grep -E "^(GITHUB_|GITLAB_|CI_)" > ci_env.txt 2>/dev/null || true
# --- Deploy state (GitOps tools) ---
helm ls -A > helm_releases.txt 2>/dev/null || true
argocd app list -o json > argocd_apps.json 2>/dev/null || true
# Integrity
sha256sum *.txt *.json *.tar 2>/dev/null | tee SHA256SUMS_cicd.txt
```

Cloud Infrastructure Artifacts

Why (DFIR). Explains *who did what, from where, and how it traversed the cloud fabric* during the incident.

- **API audit logs** (AWS CloudTrail / Azure Activity Logs / GCP Cloud Audit Logs): capture *principal*, *action*, *resource ID*, *requestId*, *srcIP/userAgent*, *errorCode*; scope by UTC window.
- **IAM changes & access**: role assumptions (STS), policy attach/detach, access key status/last-used; tie to accounts, roles, and federated identities.
- **Registry activity**: image digests/tags, push/pull/delete events (ECR/GCR/ACR) to link runtime images to CI/CD runs.
- **Load balancers**: listeners, target groups/backends, health checks, *access logs* (request path, TLS details, bytes, target status).
- **Network fabric**: VPC/VNet flow logs; ENI/NIC mappings; SG/NSG + NACL/UDR; route tables; NAT/egress gateways; public IP associations.

Thuật ngữ: API Audit Logs (CloudTrail / Activity Logs / Cloud Audit Logs) — Nhật ký kiểm toán API; IAM (Identity and Access Management) — Quản lý danh tính và quyền; Registry events — Sự kiện kho ảnh; ENI/NIC (Elastic/Network Interface) — Giao diện mạng; SG/NSG (Security Group / Network Security Group) — Nhóm bảo mật; Flow logs — Nhật ký luồng

Cloud: Control Plane — AWS (1/4)

Goal. Pull API audit events, IAM changes, and registry (ECR) activity *within a UTC window*.

```
# UTC window (example: last 24h)
START=$(date -u -d "-24 hours" +%Y-%m-%dT%H:%M:%SZ)
END=$(date -u +%Y-%m-%dT%H:%M:%SZ)

# CloudTrail (API audit)
aws cloudtrail lookup-events --start-time "$START" --end-time "$END" \
  > aws_cloudtrail.json

# IAM changes (best-effort; refine with Access Analyzer if needed)
aws iam list-roles > aws_iam_roles.json
# Optional: last-used for keys / role trust policies can be dumped separately.

# ECR registry activity (images & digests)
aws ecr describe-images --repository-name <repo> > aws_ecr_images.json
```

Thuật ngữ: CloudTrail — Nhật ký kiểm toán API (AWS); IAM — Quản lý danh tính và quyền; ECR — Kho ảnh AWS

Cloud: Control Plane — GCP & Azure (2/4)

Goal. Pull audit logs and registry metadata in the same UTC window.

GCP

```
# Reuse START/END
gcloud logging read \
  "logName:"cloudaudit.googleapis.com" \
   timestamp>=""$START"" timestamp<""$END""" \
  --format=json > gcp_audit.json
# Optional narrow IAM admin:
# protoPayload.serviceName="iam.googleapis.com"
# GAR (registry) examples vary by setup; collect via Cloud Logging or gcloud artifacts.
```

Azure

```
az monitor activity-log list --start-time "$START" --end-time "$END" \
  --offset 0 --status All --output json > az_activity.json
# ACR repository metadata (example):
# az acr repository show --name <acrName> --repository <repo> -o json
```

Thuật ngữ: GCP Cloud Audit Logs — Nhật ký kiểm toán (GCP); Azure Activity Logs — Nhật ký hoạt động (Azure); ACR — Kho ảnh Azure

Cloud: Network Fabric — AWS (3/4)

Goal. Map flows to ENI, SG, routes; pull ALB inventory and VPC Flow Logs in the UTC window.

```
# Reuse START/END from the audit slide
aws ec2 describe-network-interfaces > aws_enis.json
aws ec2 describe-security-groups > aws_sg.json
aws ec2 describe-route-tables > aws_routes.json

# VPC Flow Log events (CloudWatch Logs)
aws logs filter-log-events --log-group-name /aws/vpc/flow-logs \
  --start-time $(date -u -d "$START" +%s000) \
  --end-time $(date -u -d "$END" +%s000) \
  > aws_vpc_flowlog_events.json

# Load balancers (ALB/NLB); access logs usually in S3
aws elbv2 describe-load-balancers > aws_lbs.json
```

Thuật ngữ: ENI — Giao diện mạng đàn hồi; SG — Nhóm bảo mật; VPC Flow Logs — Nhật ký luồng VPC

Cloud: Network Fabric — GCP & Azure (4/4)

Goal. Export firewall/routes and VPC/NSG Flow Logs within the same UTC window.

GCP

```
gcloud compute firewall-rules list --format=json > gcp_fw.json
gcloud compute routes list --format=json > gcp_routes.json
gcloud logging read \
  "resource.type="gce_subnetwork" "VPC_FLOW" \
   timestamp>=""$START"" timestamp<""$END""" \
  --format=json > gcp_vpc_flow.json
```

Azure

```
az network nic list --output json > az_nics.json
az network nsg list --output json > az_nsg.json
az network route-table list --output json > az_routes.json
# NSG Flow Logs (Network Watcher):
az network watcher flow-log show --location <region> --nsg <nsgName> \
  --output json > az_flowlogs_meta.json
```

Thuật ngữ: GCP VPC Flow Logs — Nhật ký luồng VPC (GCP); NSG Flow Logs — Nhật ký luồng NSG; NIC — Giao diện mạng; Route table — Bảng tuyến

K8s: Cluster Metadata

- Version (API server, Kubelet), nodes, labels, taints.
- Namespaces, quotas, limit ranges.
- **Command**: `kubectl version -o yaml`, `kubectl get nodes -o json`, `kubectl get ns -o json`.

Thuật ngữ: taint= vết bẩn

K8s: Control Plane

- API server, controller-manager, scheduler logs (managed clusters vary).
- etcd snapshot (requires certs and endpoints); rate-limit and size planning.
- **Command**: `kubectl cluster-info dump`; `etcdctl snapshot save`.

K8s: Workload Objects

- Deployments, StatefulSets, DaemonSets, ReplicaSets, Pods.
- Jobs and CronJobs; initContainers, sidecars, volumes.
- **Command**: `kubectl get deploy,ds,rs,pod,job,cronjob -A -o json`.

Thuật ngữ: StatefulSets = Tập hợp lưu trạng thái, DaemonSets = Tập hợp tiến trình nền, Pod = Đơn vị triển khai nhỏ nhất trong K8s

K8s: Service Mesh & CNI

- Mesh control plane (Istio/Linkerd) logs and objects; mTLS policies.
- CNI state (Cilium/Calico/AWS VPC CNI); identity-aware flows (Hubble).
- **Command**: Istio `kubectl -n istio-system get all -o json`; Cilium `cilium status`.

K8s: Scheduler & Kubelet Events

- Global events timeline (`kubectl get events -A -sort-by=.lastTimestamp`).
- Node-level journal (if accessible) for kubelet/device/CNI issues.
- CrashLoopBackOff and `-previous` logs per container.

K8s: RBAC, Admission, Audit

- **RBAC** graph: ClusterRoles/Bindings, Roles/Bindings.
- Admission webhooks (validating/mutating); policy engines (Kyverno/Gatekeeper).
- Audit policy and backend; retention and redaction.

Thuật ngữ: RBAC(Role-based access control) = điều khiển truy cập theo vai trò

K8s: Helm/ArgoCD & Disaster Recovery

- Helm: releases, charts, rendered manifests, values history.
- ArgoCD: app list/status, commit SHAs, sync history, live diff.
- DR: cluster object snapshot (`kubectl get all -A -o yaml`), etcd snapshot (if allowed).

Monitoring & Observability

- Prometheus: TSDB snapshots, rule evaluations, alert history.
- Loki: label catalog, log streams (scoped queries).
- Jaeger/OTel: service list, trace exports (PII-aware filtering).

Incident Response & Forensic Artifacts

- Evidence layout per plane; **SHA256SUMS.txt** per bundle.
- Chain-of-custody log; tool versions; UTC; handlers.
- Timeline correlation: container logs, K8s events, cloud audit, runtime alerts.
- Containment: image revocation, secret rotation, RBAC tightening, network policy.

Archival, Retention, Compliance

- **Immutability**: S3 Object Lock / GCS Bucket Lock / Azure Immutable Storage.
- **Encryption**: at-rest + in-transit; KMS CMK segregation.
- **Retention matrix**: build logs (short), audit/flow (long), evidence (case-defined).
- **Access governance**: least privilege, break-glass, access logs, legal hold.

Thuật ngữ: WORM: Ché độ ghi một lần, đọc nhiều lần

Tooling Checklist

- CLIs: `docker`, `kubectl`, `ctr`/`crictl`, `aws`/`gcloud`/`az`, `helm`, `argocd` (opt), `cosign`, `syft`, `grype`, `jq`, `tcpdump`, `conntrack`, `iptables`/`nft`.
- Evidence structure conventions and naming (UTC, host, plane, scope).
- Error handling and timeboxing (pcap sizes, API quotas).

Automation Pack: Quick Start

- Unpack: `tar -xzf week8_artifact_collectors.tar.gz && cd week8-pack`
- Host: `sudo ./scripts/host_collect.sh /tmp/evidence_host 45`
- Orchestration: `./scripts/k8s_orch_collect.sh /tmp/evidence_k8s`
- Cloud (AWS example): `./scripts/aws_collect.sh ap-southeast-1 2025-10-09T00:00:00Z 2025-10-09T23:59:59Z "/ecs/" /tmp/evidence_aws`
- Merge hashes: `cat /tmp/evidence_*/SHA256SUMS.txt > /tmp/ALL_SHA256.txt`
- Build a timeline (jq): `jq -f scripts/timeline_merge.jq *.json > timeline.csv`

Appendix: Evidence Manifest (JSON Schema)

- Required: `collected_at_utc`, `artifacts[].{path, sha256, type}`.
- Optional: `case_id`, `collector_version`, `source`, `notes`.
- See: `schema/artifact_manifest_schema.json`.

Appendix: Chain of Custody Template

- Template: `templates/chain_of_custody.md`.
- Log every transfer; verify SHA256 at each hop; preserve UTC.
- Separate sensitive artifacts; apply legal hold as required.

Questions & Discussion