# Cloud Computing e-Portfolio

Module: Cloud Operations and Management

## Introduction

## This e-Portfolio brings together learning from the cloud module, covering service models (IaaS, PaaS, SaaS) and deployment types (public, private, hybrid, multi-cloud). It focuses on how governance and automation (IaC), cloud-native tooling, security/compliance, disaster recovery, serverless, and AI turn into practical, auditable operations. The sections that follow offer focused reflections on key units, applied skills, an industry application (housing/social care), and clear goals for future development.

## Final Reflective Essay. Focus on Units 2, 5, 6, 7, 8, 10

### Overview

The most valuable shift over the module was moving from feature comparison to practical design. Six units shaped that change: Unit 2 (architecture and frameworks), Unit 5 (cloud‑native & Kubernetes), Unit 6 (hybrid/fog/edge), Unit 7 (security & compliance), Unit 8 (disaster recovery), and Unit 10 (serverless). Together, they shifted cloud as automation with guardrails, with explicit trade‑offs rather than a catalogue of services.

### Unit 2. Architecture & frameworks that run

TOGAF provided a backbone for alignment between business goals and technical choices, the Software Development Life Cycle (SDLC) kept change disciplined, and Infrastructure as Code (IaC) turned architecture into executable artefacts. The key insight was to treat policies, pipelines, and templates as architecture. When landing‑zone rules, regions, SKUs, encryption, and mandatory tags live as policy‑as‑code and are enforced via CI/CD, compliance becomes a build step rather than a gatekeeping meeting (The Open Group, 2022. Microsoft, 2024a). Two tensions were examined. First, rigour versus speed: over‑engineering governance slows delivery if artefacts are written for review, not execution. Thin, enforceable guardrails, regional restrictions, encryption requirements, and budget alerts kept delivery fast and safe. Second, platform coupling versus portability: native ARM/Bicep speeds Azure governance, while Terraform modules preserve leverage across providers. The practical settlement was to anchor mandatory controls in the platform, then use portable modules for workload teams.

### Unit 5. Cloud‑native & Kubernetes: power with a price

A hands‑on Kubernetes deployment made Twelve‑Factor principles concrete: configuration in the environment, stateless processes, and disposability that enables rolling updates. Readiness and liveness probes stopped traffic from hitting a half‑awake pod. immutable image tags prevented “works on my machine” drift (Kubernetes Authors, 2024. Wiggins, n.d.). The reflection was not that Kubernetes is always right, but that it has a real operational cost, cluster upgrades, CNI choices, admission controllers, sidecars, and security baselines all add cognitive load. The practical rule was to default to PaaS for routine web/API workloads and reserve Kubernetes for problems that benefit from orchestration, multi‑component systems, spiky traffic with strict release cadence, or advanced networking needs. Minimal manifests, namespacing, and early visibility kept complexity in check.

### Unit 6. Hybrid, fog, and edge: fit to constraint

The hybrid design was framed as a response to constraints, latency, data residency, contracts, and available skills, rather than a trend. Fog and edge computing made sense where round‑trip delay harms user experience or safety. The major risk was hybrid creep if every exception path created a one‑off rule. Management‑group hierarchies, role‑based access, and policies for regions, SKUs, encryption, and budgets ensured new workloads inherited safe defaults (Microsoft, 2024a). Identity stitched the estate together: conditional access, device compliance, and least privilege replaced boundary‑only assumptions. The evaluation concluded that hybrid should be the default shape of the platform—governed at the platform edge, so teams do not renegotiate controls project‑by‑project.

### Unit 7. Security & compliance: operating rhythm, not a scramble

Security moved from periodic audits to a measurable operating rhythm. Mapping scanner findings to ISO/IEC 27001 controls turned scanning into structured work with owners and due dates (ISO, 2022). Most risks were mundane but serious: permissive security groups, weak TLS, stale images, and over‑privileged identities. The habits that paid off were identity‑first security, MFA, conditional access, least privilege, and evidence‑based assurance with central logs/metrics feeding a risk register and change tickets that reference control objectives (Greenbone, 2025). The result was a programme with cadence, not a quarterly fire drill.

### Unit 8. Disaster recovery: design for bad days

Simulated failure forced a reality check. Backup/restore drills using open‑source tools worked but exposed manual friction, naming inconsistencies, undocumented dependencies, and longer‑than‑planned sequences. For tier‑1 workloads, orchestrated failover with continuous replication was justified to meet realistic RTO/RPO. for everything else, robust backups and tested restore paths sufficed (Microsoft, 2024b). The durable lesson was simple: recovery is a design input. Network addressing, secrets management, and storage classes must be recoverable by default. Regular game days convert static runbooks into operational readiness.

### Unit 10. Serverless: small pieces, governed well

Serverless functions delivered clear wins for short‑lived, event‑driven tasks. Scale‑to‑zero avoided paying for idle, and deployment friction was low (OpenFaaS Ltd, 2024). The risk was function creep without shared logging, naming, and budget alarms. The fix was a standard function template with logging, input validation, and retries, plus central metrics. Serverless is not a fit for long‑running or stateful services. combined with PaaS/containers, it keeps the overall system coherent and cost‑aware.

### Cross‑unit habits that stuck

Several habits cut across units. Govern in the path of change: policy and budget guardrails in code beat slide decks (Microsoft, 2024a). Choose the simplest viable model: PaaS and SaaS are often ‘boringly right’. operate by numbers: RTO 4h/RPO 15m, probe timings, budget ceilings, and SLOs keep systems honest (Kubernetes Authors, 2024. Microsoft, 2024b). Identity first: modern identity reduces boundary complexity (ISO, 2022). And practice recovery: backups are a plan. Gamedays are proof.

## Skills Development

Practical capability strengthened across automation, cloud‑native delivery, governance, security, DR, serverless, and migration. Automation: idempotent Bash/Python scripts provisioned OpenStack networks, security groups, and instances with parameters and structured logs, habits that transfer to Azure CLI/PowerShell. Cloud‑native: Docker images were built from minimal bases. Kubernetes Deployments and Services exposed a small API. configuration moved to ConfigMaps and secrets. health checks used readinessProbe and livenessProbe (e.g., initialDelaySeconds=5, periodSeconds=10) to prevent premature routing (Kubernetes Authors, 2024). CI fundamentals (build - test - deploy) reduced manual steps and improved repeatability.

Governance & IaC: Azure Resource Manager and Bicep templates modelled a landing‑zone hierarchy (management groups - subscriptions - resource groups) with policy enforcing regions, SKUs, encryption, and mandatory tags. What‑if previews caught unintended changes. Terraform modules offered portability with remote state and review gates (Microsoft, 2024a). Security: OpenVAS scanning outputs were triaged and mapped to ISO/IEC 27001 controls. recurring tasks included patching cadence, TLS hardening, and access reviews (ISO, 2022. Greenbone, 2025). DR & continuity: backup/restore procedures were tested. a target RTO of 4 hours and RPO of 15 minutes were met in lab conditions for a tier‑1 scenario. Runbooks captured dependencies and failback (Microsoft, 2024b). Serverless: an OpenFaaS function was implemented with resource limits, scale‑to‑zero, and gateway metrics, supporting a pattern for short‑duration utilities (OpenFaaS Ltd, 2024). Migration: Percona XtraBackup supported low‑downtime database moves. managed services like AWS DMS and Azure Migrate were reviewed for larger estates (Percona, 2024. AWS, 2024).

Additional practice focused on container image hygiene (multi‑stage builds, minimal bases, provenance checks) and release safety: blue/green swaps for PaaS apps and canary rollouts for Kubernetes, with automated rollback on health‑signal regression. Observability was standardised: structured logs with correlation IDs, RED metrics for services, and SLO dashboards that highlighted error‑budget burn. These practices linked day‑to‑day engineering with measurable outcomes, reinforcing the habit of shipping small, reversible changes (Kubernetes Authors, 2024. Microsoft, 2024a).

## Application to Industry (Housing & Social Care)

Housing and social care providers operate under strict governance for personal data and budget controls, while supporting distributed teams and 24/7 services. A practical architecture begins with a governed Azure landing zone: management groups separating environments (Prod/UAT/Dev), subscriptions with budget alerts, and policy enforcing UK data residency, encryption at rest and in transit, diagnostic logging, and mandatory tags (service, owner, cost centre) (Microsoft, 2024a, 2024e). Role‑based access, conditional access, and device compliance reduce attack surface while preserving usability. Workload placement prioritises simplicity and value: SaaS for commodity needs. PaaS (App Service + managed database) for most web/APIs. containers where orchestration clearly improves resilience or release velocity. serverless for short‑lived, event‑driven tasks (OpenFaaS Ltd, 2024). Observability aggregates logs and metrics with SLOs that surface error‑budget burn early.

Scenario 1. Repairs intake: a light API on App Service receives requests. images uploaded by residents are resized and redacted by a serverless function. events and metadata are written to central logs with retention labels to support GDPR auditability. A modest ML model prioritises urgent cases based on recent history, helping schedulers focus on high‑risk residents. Region and encryption policies block risky deployments. mandatory tags make costs visible by service. DR differentiates tiers: immutable backups for all. orchestrated failover with Azure Site Recovery for the single critical workflow to meet RTO 4h / RPO 15m (Microsoft, 2024b).

Scenario 2. Rent‑arrears outreach: a timed workflow aggregates case events, enriches with recent payments, and triggers a function to create tailored letters through private endpoints. Access is time‑limited via role assignments. all actions are logged for audit. A small forecasting model flags vulnerability indicators (e.g., repeated small payments) for human review rather than automatic escalation. Tags and budget alerts keep spending predictable. where patterns exceed thresholds, autoscaling policies or instance sizing is reviewed (Microsoft, 2024a. OpenFaaS Ltd, 2024). These examples show how simple cloud patterns, paired with responsible AI and policy‑as‑code, improve outcomes without adding operational fragility.

## Future Learning Goals

1- Governed landing zone: publish a reusable ARM/Bicep + Terraform baseline (management groups, policies, RBAC, budgets) with CI gates (what‑if/plan/apply) and zero policy violations in validation within 8 weeks (Microsoft, 2024a, 2024e).   
2- Operational resilience: run quarterly DR game‑days targeting RTO 4h and RPO 15m for one tier‑1 service. capture evidence of failover and failback, and fold lessons into runbooks (Microsoft, 2024b).   
3- Serverless toolkit: deliver two production‑ready utilities (document processing, nightly export) with central logs, metrics, and per‑function budget alarms to prove scale‑to‑zero efficacy (OpenFaaS Ltd, 2024).   
4- Continuous security: implement scheduled scanning mapped to ISO/IEC 27001 with MTTP 14 days for high‑risk findings and no critical 30 days, including access review cadence and certificate rotation (ISO, 2022. Greenbone, 2025).   
5- Responsible AI: build a governed ML workflow using a managed stack (Azure AI or SageMaker) with dataset lineage, model monitoring, and explainability. pilot one analytics use case that improves an operational KPI without introducing decision bias.

## Conclusion

Across six focal units, the module shifted cloud practice as disciplined, measurable engineering: guardrails in code, automation as default, visibility first, identity‑led security, and recovery rehearsed until predictable. Practical skills, scripting, Kubernetes, IaC, security scanning, DR, serverless, and migration, combining with governance to deliver reliable, auditable services. In housing and social care, that translates to safer outcomes for residents and steadier operations for staff. The forward plan keeps the posture light but strong, adapting as technology and regulatory expectations evolve, and keeps learning anchored to measurable outcomes, so that resilience, security, and cost control are designed in, not bolted on.

## References (Harvard style)

* Amazon Web Services (2024) AWS Database Migration Service (DMS) User Guide. Available at: <https://docs.aws.amazon.com/dms/latest/userguide/Welcome.html>
* Greenbone (2025) OpenVAS (Greenbone Community Edition). Available at: <https://www.openvas.org/>
* International Organisation for Standardisation (ISO) (2022) ISO/IEC 27001:2022 — Information security, cybersecurity and privacy protection — ISMS — Requirements. Available at: <https://www.iso.org/standard/82875.html>
* Kubernetes Authors (2024) Kubernetes documentation. Available at: <https://kubernetes.io/docs/home/>
* Microsoft (2024a) Azure Resource Manager overview. Microsoft Learn. Available at: <https://learn.microsoft.com/azure/azure-resource-manager/management/overview>
* Microsoft (2024b) What is Azure Site Recovery? Microsoft Learn. Available at: <https://learn.microsoft.com/azure/site-recovery/site-recovery-overview>
* Microsoft (2024c) Azure Migrate overview. Microsoft Learn. Available at: <https://learn.microsoft.com/azure/migrate/migrate-services-overview>
* Microsoft (2024d) Azure App Service overview. Microsoft Learn. Available at: <https://learn.microsoft.com/azure/app-service/overview>
* Microsoft (2024e) Azure Policy overview. Microsoft Learn. Available at: <https://learn.microsoft.com/azure/governance/policy/overview>
* Mell, P. and Grance, T. (2011) The NIST Definition of Cloud Computing (SP 800‑145). Gaithersburg, MD: National Institute of Standards and Technology. Available at: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>
* OpenFaaS Ltd (2024) OpenFaaS Documentation. Available at: <https://docs.openfaas.com/>
* Percona (2024) Percona XtraBackup 8.0 Documentation. Available at: <https://docs.percona.com/percona-xtrabackup/8.0/>
* The Open Group (2022) TOGAF® Standard, 10th Edition. Available at: <https://www.opengroup.org/togaf>
* Wiggins, A. (n.d.) The Twelve‑Factor App. Available at: <https://12factor.net/>