Architecture Document: Multi-Tenant SaaS on AWS (Commercial & GovCloud) with EKS and FIPS Posture

Author: Ted Gibson | Version: 1.4 | Date: 2025-08-26

1. Executive Summary

This document defines a single-codebase, multi-tenant SaaS architecture that operates in two deployment stamps: AWS Commercial (private sector) and AWS GovCloud (public sector). The design begins with the strictest requirements and provides configurable isolation levels by customer SKU (pooled, silo, and dedicated stamp). It standardizes CI/CD to deploy to Amazon EKS with explicit, prescriptive steps to meet FIPS expectations in AWS Commercial and a FedRAMP-aligned posture in AWS GovCloud. Every decision includes a short rationale so you can understand why each step is necessary.

2. Goals and Non-Goals

2.1 Goals

- Single codebase serving both commercial and government customers.
- Configurable multi-tenant isolation: pooled (row/schema), silo (DB-per-tenant), and dedicated stamp (per-agency deployment).
- EKS-based runtime with hardened, FIPS-capable worker nodes.
- CI/CD with security gates (secrets, IaC lint/policy, image scanning, SBOM, optional SAST).
- Evidence generation suitable for audits (centralized logging, posture dashboards, conformance rules).

2.2 Non-Goals

- Achieving a formal FedRAMP authorization in this phase (the design aligns to it but authorization is a separate program).
- Supporting non-AWS clouds in this iteration.

3. Requirements

3.1 Functional

- Serve multiple tenants with logical isolation and configurable features by SKU.
- Automate tenant onboarding, configuration, and lifecycle.
- Deploy reliably through **sec** → **dev** → **stg** → **prod** in each partition (Commercial and GovCloud).

3.2 Non-Functional

- Security by default (encryption, least privilege, vulnerability management).

- High availability (multi-AZ), clear RTO/RPO targets, and repeatable disaster recovery procedures.
- Operational efficiency via automation and consistent guardrails.

3.3 Compliance & FIPS Expectations (Commercial)

- **Mandatory:** route all AWS SDK/CLI calls to **FIPS endpoints** in build and runtime where available (set environment/config globally).
- **Mandatory:** use **FIPS-capable OS crypto** wherever cryptography occurs (Bottlerocket FIPS for EKS workers; Amazon Linux 2023 in FIPS mode for CI runners and other hosts).
- **Mandatory:** encrypt data at rest and in transit; **envelope-encrypt Kubernetes Secrets** with customer-managed KMS keys.
- Provide demonstrable evidence through continuous compliance checks and centralized logs.

Reason: These settings ensure TLS and cryptographic operations use FIPS-validated modules and endpoints end-to-end.

4. Multi-Tenant Model and Deployment Stamps

We operate a single platform with configurable tenancy:

- **Pooled:** Shared database with row-level or schema isolation for cost efficiency.
- **Silo:** Database-per-tenant for stronger isolation and easier data residency guarantees.
- **Stamp:** Dedicated, isolated environment (for example, an agency in GovCloud) for the highest assurance.

Reason: Starting with the strictest options lets us serve government needs without forking the code and preserves operational efficiency for commercial tenants.

5. Organization, Accounts, and Separation of Duties

We create four accounts per partition (Commercial and GovCloud): **sec, dev, stg, prod**.
Reason: Clear blast-radius boundaries, controlled promotion paths, and a dedicated security vantage point.

5.1 Accounts

- **sec:** Security and audit tooling, delegated admin for security services, centralized log archive.
- **dev:** CI/CD services, developer EKS cluster, artifact and container registries.
- **stg:** Pre-production EKS cluster and data stores mirroring prod at lower scale.
- **prod:** Production EKS cluster and all customer-facing workloads.

5.2 Guardrails (Service Control Policies and Baselines)

- Deny public S3 by default and require account-level Public Access Block.
- Require encryption at rest for EBS, EFS, RDS, and S3; deny disabling EBS-encryption-by-default.
- Disallow RDS instances with public exposure in workload accounts.
- Enforce use of TLS 1.2+ where supported.
- Restrict wildcard administrator policies and require approvals for break-glass roles.

Reason: SCPs prevent classes of misconfiguration regardless of individual team choices and reduce audit burden.

6. Networking and Isolation

- VPC per account/environment with isolated private subnets for nodes and data tiers.
- Ingress via load balancers; egress through NAT with restrictive network ACLs and, where appropriate, VPC endpoints.
- Strict security groups; no direct internet exposure for nodes or databases.

Reason: Minimizes exposure, supports zero-trust controls, and simplifies boundary definitions for audits.

7. Identity and Access Management

- External SSO for workforce access with MFA and short session durations.
- Role-based access; least-privilege IAM policies; no long-lived credentials for CI/CD.
- Cross-account deployment roles assumed by pipelines to target stg and prod.
- Break-glass role with logging, approvals, and expiry.

Reason: Centralized identity and ephemeral access reduce credential risk and simplify reviews.

8. Data, Encryption, and Residency

- **KMS keys** per environment and partition; separate keys for application data, logs, and backups.
- **EKS Secrets envelope-encrypted** with customer-managed KMS keys; periodic key rotation and re-encryption.
- Backups are encrypted and kept in-partition; no cross-partition movement for regulated data.
- Tenant isolation enforced at app and data layers; silo tenants receive dedicated databases and keys when required.

Reason: Clear key boundaries and encrypted flows support confidentiality, residency, and incident containment.

9. EKS Runtime Architecture

9.1 Control Plane & Nodes

- Managed Amazon EKS control plane per environment.
- **Managed node groups pinned to Bottlerocket FIPS AMIs** (Commercial). For EC2/runners, **enable AL2023 FIPS mode**.
- Auto scaling with disruption budgets; multi-AZ node groups.

Reason: Reduces patching surface, meets cryptographic module expectations, and provides resilient capacity.

9.2 Platform Security Controls

- Pod Security Admission set to **restricted** on all namespaces (enforce/warn/audit).
- Admission control (Gatekeeper or Kyverno) to enforce: no privileged pods, read-only root FS, no host network/paths, required NetworkPolicies, and **only signed images**.
- Image signing with an organizational signing profile; verification enforced at admission.
- Private image pulls from ECR; avoid public registries for production.

Reason: Defense in depth; image provenance becomes enforceable policy.

9.3 Secrets and Configuration

- Kubernetes secrets are envelope-encrypted; sensitive app config in Secrets Manager or Parameter Store.
- RBAC mapped to IAM roles; CI/CD deploy role is scoped to apply manifests only.

Reason: Limits blast radius and keeps credentials off developer machines.

9.4 ECR and Private Connectivity Caveat

- ECR FIPS endpoints are not available through PrivateLink. Fully private clusters may require **NAT egress** for FIPS endpoint access, or an approved, time-boxed exception for image pulls during bootstrap only.

Reason: Avoids deployment failures in isolated subnets while preserving FIPS endpoint usage elsewhere.

10. CI/CD Pipeline (sec → dev → stg → prod in each partition)

10.1 Stages

1. **Source:** commit triggers pipeline; branch policies enforced by code review.
2. **Build:** container built on hardened runner; **FIPS endpoint usage is mandatory** via environment or SDK config.
3. **Security Scan:** secrets scan, IaC lint, policy checks; image scanning on push; SBOM generated.
4. **Sign:** container image signed; attestations stored with artifacts.

5. **Deploy-dev:** automatic; smoke and integration tests.
6. **Approve:** manual approval before stg and before prod.
7. **Deploy-stg:** apply Helm/manifests; run load and failover tests.
8. **Deploy-prod:** progressive rollout with canaries and automated rollback on SLO breach.

Reason: Security gates catch risks pre-deployment, and staged rollouts reduce production risk.

10.2 Mandatory FIPS Settings in CI and App Runtimes

- Set an environment/config value to direct **all** SDK/CLI calls to FIPS endpoints in AWS Commercial and GovCloud.
- Build runners and any component performing cryptography must run on **FIPS-capable OS images**; enable AL2023 **FIPS mode** where policy requires.

Reason: Ensures TLS and crypto use FIPS-validated modules and endpoints throughout.

11. Observability, Security Operations, and Evidence

- Organization-wide CloudTrail with logs delivered to a central, encrypted log archive in **sec**.
- Resource configuration recorded by AWS Config; aggregated in **sec** for a single source of truth.
- Security Hub enabled with a NIST-aligned standard; findings from GuardDuty, Inspector, and others aggregated.
- **CloudWatch metrics, logs, and traces with alarms; app-level dashboards and SLOs.**

Reason: Central visibility enables rapid detection/response and supports audit evidence requests.

12. Tenant Onboarding, SKUs, and Lifecycle

- Feature flags and SKU definitions determine isolation level and default security settings.
- Automated onboarding creates tenant records, allocates data resources (schema/DB), assigns keys, and sets quotas.
- Automated offboarding disables access, exports customer data per contract, and schedules secure deletion after retention.

Reason: Configuration-driven variations avoid code forks and make compliance behavior consistent per customer class.

13. Reliability, DR, and Business Continuity

- Multi-AZ for all critical components; workload auto-healing.
- Backups for databases and object storage with periodic restore tests.

- Target RTO and RPO documented; playbooks for region impairment and tenant-level incidents.

Reason: Ensures availability commitments are met and auditable.

14. Cost Management and Tagging

- Mandatory tags: Project, Environment, TenantId (where applicable), DataClass, Owner, CostCenter.
- Budgets and cost anomaly detection per environment; right-size node groups and **autoscaling policies.**

Reason: Enables chargeback/showback and early detection of cost regressions.

15. Operational Runbooks and SLAs

- Incident response runbook with severity classification and communications.
- Patch management schedule and process for OS images, containers, and dependencies.
- Vulnerability management SLA tied to severity; emergency patching process.

Reason: Codifies expectations and reduces mean time to recover.

16. FIPS Compliance in AWS Commercial — Prescriptive Checklist

1. **Force FIPS endpoints everywhere:** set a global environment/config in CI jobs, app containers, and ops hosts so all AWS API calls use FIPS service endpoints.
2. **Use FIPS-validated OS crypto where cryptography occurs:** pin EKS node groups to **Bottlerocket FIPS** AMIs; enable **AL2023 FIPS mode** on CI runners and other hosts that perform crypto.
3. **Encrypt all data at rest with CMKs:** S3, EBS, EFS, RDS, and application data stores; maintain a key inventory per environment and tenant where applicable.
4. **Enable EKS Secrets envelope encryption** using customer-managed KMS keys; re-encrypt secrets following key rotation.
5. **Enforce TLS 1.2+** on all ingress listeners and client libraries; record selected TLS policies in infrastructure code.
6. **Make the pipeline produce evidence:** retain build logs showing FIPS endpoint selection; store SBOMs and scan results in a KMS-encrypted artifacts bucket.
7. **Centralize posture and evidence:** enable posture services, aggregate findings in **sec**, and export reports for audits.
8. **Document the ECR FIPS/PrivateLink limitation:** plan NAT egress for FIPS pulls, or define a controlled exception for bootstrap only.

17. Operator Procedures — Verifying FIPS Posture

- **Confirm endpoint routing:** run a CLI command with debug from CI or a pod and verify the hostname includes a FIPS endpoint; archive the log snippet.
- **Verify OS FIPS mode:** check the system FIPS flag on hosts and containers and store results with AMI/update change tickets.
- **Validate KMS usage:** export inventories of keys and their usage for each environment and attach to release records.
- **Check EKS secrets encryption:** show the cluster's encryption config and periodic re-encryption logs after key rotation.
- **Check TLS edge:** record configured TLS policies for load balancers and keep a periodic scan result in evidence storage.
- **Archive SBOMs and scan outputs:** store per-release with retention and immutability policies.

18. Step-by-Step Implementation Plan (What and Why)

1. Create OUs and accounts: **sec, dev, stg, prod** in both partitions. *Why:* establishes hard boundaries and a consistent promotion path.
2. Enable centralized logging and config recording. *Why:* tamper-evident records and configuration history are the basis for audits.
3. Apply baseline SCPs. *Why:* prevent risky configurations at the control plane regardless of team choices.
4. Stand up ECR and enable image scanning. *Why:* detect vulnerabilities early and keep supply-chain inventory.
5. Create EKS clusters and **pin node groups to Bottlerocket FIPS**. *Why:* hardened runtime with validated crypto support.
6. Enable secrets envelope encryption with KMS. *Why:* strengthen confidentiality and key separation per environment.
7. Install admission controls and baseline policies. *Why:* enforce least privilege and provenance at workload admission.
8. Build CI/CD with security gates and **enforce FIPS endpoints**. *Why:* make every change pass the same security bar before it ships.
9. Implement image signing and verification. *Why:* ensure only trusted images run in the cluster.
10. Automate tenant onboarding and feature flags. *Why:* consistency, speed, and avoidance of custom forks.
11. Define DR playbooks and test restores. *Why:* validate RTO/RPO and avoid surprises in an outage.

12. Stand up posture dashboards and evidence exports. *Why:* continuous readiness for assessments and executive reporting.

19. Key Risks and Mitigations

- **Service parity gaps between Commercial and GovCloud.** Mitigation: maintain a compatibility matrix; avoid non-portable services in shared code paths.
- **Egress to non-FIPS endpoints.** Mitigation: egress control with network rules and enforced build-time environment.
- **Policy sprawl in admission controllers.** Mitigation: start with a baseline set, version policies, and stage rollouts.
- **Tenant data mix-ups in pooled mode.** Mitigation: strict tenancy guards in code, automated tests, and query-level enforcement (for example, row-level security).

20. Appendices

20.1 Sample SCP Themes

- Deny S3 without encryption and with public ACLs/policies.
- Deny RDS with public exposure and unencrypted storage.
- Require EBS encryption by default and deny disabling it.
- Deny IAM policies that grant full administrative access outside designated roles.

20.2 Example Admission Policy Set

- No privileged containers, no host network, no hostPath, read-only root filesystem.
- Require NetworkPolicies in production namespaces.
- Only images signed by the organization's signing profile.

20.3 CI Failure Criteria Examples

- Secrets scan findings: fail build.
- IaC lint or policy violations: fail build.
- Container critical vulnerabilities: block promotion; require exception approval.

20.4 Data Classification and Residency

- Public, Internal, Confidential, Regulated. Regulated data remains in-partition; cross-partition movement is prohibited.

20.5 Glossary

A. **sec:** Security and audit account.
B. **dev:** Development and CI/CD account.
C. **stg:** Staging (pre-production) account.
D. **prod:** Production account.
E. **Partition:** AWS Commercial or AWS GovCloud deployment stamp.
F. **SKU:** A product tier that toggles features and isolation.

20.6 Change Log

v1.4

- Made use of FIPS endpoints mandatory across CI and runtime.
- Pinned EKS worker nodes to Bottlerocket FIPS AMIs and clarified AL2023 FIPS mode for hosts.
- Clarified that KMS uses FIPS-validated HSMs and that key inventories must be maintained as evidence.
- Added explicit ECR FIPS/PrivateLink limitation and mitigation.
- Added a prescriptive checklist and operator verification procedures for FIPS in Commercial.