**============================================================**

Author: TGibson  
File: SaaS\_Proposal\_GovCloud.docx  
Repo: AWS EKS CI/CD (Commercial + GovCloud) via CloudFormation  
Version: 1.5  
Date: 2025-08-28

**============================================================**

**Proposal — AWS EKS CI/CD (Commercial + GovCloud) via CloudFormation**

**Table of Contents**

* **Acronyms & Definitions**
* **Chapter 1 — Executive Summary**
* **Chapter 2 — SaaS Principles**
* **Chapter 3 — Infrastructure Layer**
  + 3.1 AWS Accounts, VPCs, Subnets, **Security Groups**, Endpoints
  + 3.2 EKS Clusters (dev/stg/prod)
  + 3.3 Add-ons (ALB Controller, Metrics Server, OIDC Integration)
  + 3.4 **Preflight Validation Scripts (deploy/preflight-vpc.sh)**
  + 3.5 **FIPS Endpoints, VPC-only, Compliance Controls**
* **Chapter 4 — CI/CD Pipeline Layer**
  + 4.1 Full CodePipeline Flow (Source → Build → Scan → Sign/Attest → Approval → Deploy → Mirror)
  + 4.2 Why CodePipeline + CodeBuild (FedRAMP Native)
  + 4.3 Tools Inside CodeBuild (Syft, Trivy, Cosign)
  + 4.4 FIPS Enforcement (VPC Endpoints)
  + 4.5 Repo Proof: Buildspecs + Pipeline Templates
* **Chapter 5 — Security & Compliance**
  + 5.1 Runtime Enforcement (Kyverno verifyImages, PodSecurity)
  + 5.2 IAM Least Privilege
  + 5.3 IRSA ServiceAccounts & OIDC Federation
  + 5.4 Zero Trust Posture
  + 5.5 Compliance Tie-In (FedRAMP, RMF, FIPS baked in)
  + 5.6 **Secrets Management (AWS Secrets Manager + External Secrets Operator)**
* **Chapter 6 — Supply Chain Security**
  + 6.1 SBOM Generation (Syft)
  + 6.2 Vulnerability Scanning (Trivy)
  + 6.3 Signing + Attestation with KMS (Cosign)
  + 6.4 Kyverno Admission Checks (verifyImages, PodSecurity)
  + 6.5 Immutable ECR Repositories
* **Chapter 7 — Tenant Lifecycle**
  + 7.1 Namespace-per-Tenant
  + 7.2 Helm Tenant Templates
  + 7.3 Quotas & Network Policies
  + 7.4 Onboarding Automation (bootstrap-fill-and-deploy.sh)
  + 7.5 Noisy-Neighbor Protection & Isolation Models
* **Chapter 8 — Public vs Private Partition (Commercial vs GovCloud)**
  + 8.1 Commercial → GovCloud Mirroring
  + 8.2 GovCloud Restrictions (Private-only, FIPS Endpoints)
  + 8.3 Why This Dual Setup is Necessary
* **Chapter 9 — Approval & Governance**
  + 9.1 Manual Approvals in CodePipeline (dev → stg → prod)
  + 9.2 Signed Artifact Verification
  + 9.3 Notifications & Event Routing (SNS, EventBridge, Slack/ServiceNow)
  + 9.4 Ops Handoff
* **Chapter 10 — Observability & Operations**
  + 10.1 Logging (CloudWatch, S3, Fluent Bit/ELK optional)
  + 10.2 Metrics (CW Container Insights, AMP, Grafana)
  + 10.3 Tracing (X-Ray / OpenTelemetry)
  + 10.4 Cost Allocation (CUR, Tags, Grafana overlays)
  + 10.5 **Backup, DR & Lifecycle Management**
  + 10.6 **Cost & Resilience Guardrails (ECR lifecycle, DR runbook, tenant tagging)**
* **Chapter 11 — Criteria Compliance Matrix**
* **Chapter 12 — Conclusion**

**Appendices**

* Appendix A — AWS Services by Function
* Appendix B — Environment Setup & Validation Checklist
* Appendix C — Deployment Evidence (Scripts & Outputs)
* Appendix D — File-to-Chapter Mapping (Repo Proof)
* Appendix E — **Optional AWS Services for Scale-Out (Service Catalog, Control Tower)**

**Acronyms & Definitions**

* **ALB** – Application Load Balancer
* **AMG** – Amazon Managed Grafana
* **AMP** – Amazon Managed Prometheus
* **API** – Application Programming Interface
* **ATO** – Authority to Operate
* **AWS** – Amazon Web Services
* **CAC** – Common Access Card
* **CI/CD** – Continuous Integration / Continuous Delivery
* **CJIS** – Criminal Justice Information Services (FBI) security policy
* **CM-8** – NIST 800-53 Control: Information System Component Inventory
* **CodeBuild** – AWS managed build service for CI/CD
* **CodeCommit** – AWS managed Git-based source control service
* **CodePipeline** – AWS managed CI/CD orchestration service
* **CUR** – Cost and Usage Report (AWS Billing)
* **Dev/Stg/Prod** – Development / Staging / Production environments
* **DoD SRG** – Department of Defense Security Requirements Guide
* **DR** – Disaster Recovery
* **EBS** – Elastic Block Store
* **ECR** – Elastic Container Registry
* **EFS** – Elastic File System
* **EKS** – Elastic Kubernetes Service
* **ELK** – Elasticsearch, Logstash, Kibana stack
* **EO 14028** – Executive Order on Improving the Nation’s Cybersecurity (Supply Chain)
* **ESO** – External Secrets Operator
* **FedRAMP** – Federal Risk and Authorization Management Program
* **FIPS** – Federal Information Processing Standard (e.g., 140-2 for cryptography)
* **HPA** – Horizontal Pod Autoscaler
* **IAM** – Identity and Access Management
* **IRSA** – IAM Roles for Service Accounts (Kubernetes → IAM integration)
* **ITAR** – International Traffic in Arms Regulations
* **KMS** – Key Management Service (AWS cryptographic key management)
* **Kyverno** – Kubernetes-native policy engine (admission controller)
* **NIST 800-37 (RMF)** – Risk Management Framework
* **NIST 800-53** – Security and Privacy Controls for Information Systems
* **NLB** – Network Load Balancer
* **OIDC** – OpenID Connect (federated identity standard)
* **OTel** – OpenTelemetry (observability framework)
* **PIV** – Personal Identity Verification (government smart card)
* **RBAC** – Role-Based Access Control
* **RMF** – Risk Management Framework (NIST 800-37)
* **S3** – Simple Storage Service
* **SBOM** – Software Bill of Materials
* **SCP** – Service Control Policy
* **Seccomp** – Secure Computing Mode (Linux kernel sandboxing)
* **SES** – Simple Email Service
* **SNS** – Simple Notification Service
* **SQS** – Simple Queue Service
* **STS** – Security Token Service
* **Syft** – SBOM generation tool
* **Tenant Stamp** – A fully isolated environment (VPC, EKS, pipeline) provisioned per customer
* **Trivy** – Vulnerability scanning tool
* **VPC** – Virtual Private Cloud
* **WAF** – Web Application Firewall
* **Zero Trust** – Security framework requiring continuous verification

**Chapter 1 — Executive Summary**

**1.1 What the SaaS Is**

This solution delivers a **multi-tenant Software-as-a-Service (SaaS) platform** designed to operate across both **AWS Commercial** and **AWS GovCloud** partitions. It provides a unified codebase with configurable deployment models:

* **Pooled:** Shared infrastructure (Aurora PostgreSQL with Row Level Security, shared EKS cluster namespaces).
* **Siloed:** Tenant-specific schemas and namespaces within shared clusters.
* **Dedicated Stamp:** Fully isolated infrastructure stack (separate AWS account, VPC, EKS, and CI/CD pipeline).

This architecture enables federal and commercial customers to adopt **zero-trust scoring and compliance coaching** services at the level of isolation their mission requires.

**1.2 Why AWS**

AWS was chosen because it uniquely provides:

* **FedRAMP High authorization** across Commercial and GovCloud services.
* **FIPS 140-2 validated endpoints** for cryptographic operations (KMS, S3, ECR, STS).
* **Scalable Kubernetes runtime (Amazon EKS)** with add-ons for ingress, autoscaling, and observability.
* **Mature security ecosystem** including IAM, VPC endpoints, WAF, and KMS.
* **Operational parity** between Commercial and GovCloud, enabling a mirrored pipeline without duplicate build systems.

**1.3 Why Multi-Tenant**

Delivering as multi-tenant SaaS provides:

* **Cost efficiency:** Shared infrastructure where possible (pooled/silo) with optional full isolation.
* **Flexibility:** Support for commercial, DoD, and federal agencies with one codebase.
* **Consistency:** Standardized CI/CD, governance, and compliance guardrails across all tenants.
* **Speed:** Tenant onboarding via Helm templates and automation scripts (bootstrap-fill-and-deploy.sh).

**1.4 Why FIPS/GovCloud**

Some customers mandate operations in **GovCloud** to satisfy **ITAR, CJIS, and DoD SRG** requirements. This platform supports:

* **GovCloud deployment parity** with Commercial builds (signed images mirrored via ECR).
* **FIPS 140-2 validation** enforced by VPC endpoints and private-only builds.
* **Controlled operations** managed by U.S. persons only in GovCloud.
* **Secure CI/CD handoff** between Commercial and GovCloud partitions without duplicating build pipelines.

**1.5 Key Drivers: Compliance, Scalability, Security, Repeatability**

* **Compliance:** FedRAMP High, FIPS 140-2, and NIST RMF (800-37/800-53) controls are baked into every layer.
* **Scalability:** EKS clusters with Horizontal Pod Autoscaler (HPA) and AWS Load Balancer Controller. Tenants can be deployed in pooled, siloed, or dedicated stamps with minimal configuration.
* **Security:** Zero Trust is enforced at build (SBOM + scans + cosign signatures), deploy (immutable digests), and runtime (Kyverno admission controls).
* **Repeatability:** All resources are provisioned via **Infrastructure-as-Code (CloudFormation + Helm)**, ensuring deterministic deployments and auditability.

**Chapter 2 — SaaS Principles**

**2.1 Single Codebase, Multiple Deployment Models**

**Understanding:**  
The repo is designed around a **single codebase** that supports three levels of tenant isolation: pooled, siloed, and dedicated stamps. This reduces complexity while offering flexibility for different compliance and mission requirements.

**Approach:**

* **Pooled Model:** Tenants share EKS clusters and Aurora PostgreSQL (Serverless v2) with **Row Level Security (RLS)**. Enforced via Cognito JWT claims and DB policies.
* **Silo Model:** Tenants share infrastructure but operate with **dedicated DB schemas, namespaces, IAM roles, and quotas**.
* **Dedicated Stamp Model:** A complete **isolated AWS account + VPC + EKS + pipeline** deployed per tenant, typically for federal or classified workloads.

**Tools & Platforms:**

* Kubernetes namespaces, Helm templating
* Aurora PostgreSQL with RLS policies (ops/migrations/aurora/0001\_init\_tenancy.sql)
* CloudFormation templates for isolated “stamps” (templates/aurora-postgres-slsv2.yaml)

**Proof in Repo:**

* ops/tenant/values-tenant-template.yaml
* ops/k8s/tenant-networkpolicy.yaml
* ops/k8s/tenant-resourcequota.yaml

**Benefit:**  
Delivers **cost efficiency** for commercial tenants, while offering **mission-specific isolation** for government agencies — all without maintaining multiple codebases.

**2.2 Public (Commercial) vs Private (GovCloud) Separation**

**Understanding:**  
Many federal agencies require workloads in GovCloud due to **ITAR, CJIS, and DoD SRG** compliance. The repo supports **pipeline parity** across Commercial and GovCloud to maintain consistency.

**Approach:**

* **Commercial Partition:** Builds, scans, SBOM generation, and artifact signing occur here using the full AWS service catalog.
* **GovCloud Partition:** Receives only **signed + attested artifacts** from Commercial via ECR mirroring (templates/gov-mirror-receiver.yaml).
* **No internet egress** allowed in GovCloud — all API traffic routed through **FIPS VPC endpoints**.

**Tools & Platforms:**

* AWS CodePipeline / CodeBuild
* AWS ECR (cross-partition replication)
* AWS KMS for signing (Cosign integration)

**Proof in Repo:**

* ops/pipeline/buildspec-mirror-gov.yml
* templates/gov-mirror-receiver.yaml

**Benefit:**  
Guarantees **audit parity** across partitions, while ensuring **federal workloads run inside GovCloud** without duplicating build pipelines.

**2.3 Customer Isolation: Namespaces, IAM, Quotas, and Network Segmentation**

**Understanding:**  
Multi-tenancy requires strong isolation at runtime to prevent data leakage and ensure zero trust.

**Approach:**

* **Namespaces:** Each tenant runs in a dedicated Kubernetes namespace.
* **IAM Roles:** Workloads assume scoped roles via IRSA (IAM Roles for Service Accounts).
* **Resource Quotas:** Ensure no tenant can overconsume compute, memory, or storage.
* **Network Policies:** Restrict pod-to-pod and pod-to-service communications, preventing lateral movement.

**Tools & Platforms:**

* Kubernetes ResourceQuotas, NetworkPolicies, RBAC
* AWS IAM + IRSA integration (templates/workload-deploy-iam.yaml)

**Proof in Repo:**

* ops/k8s/tenant-networkpolicy.yaml
* ops/k8s/tenant-resourcequota.yaml

**Benefit:**  
Delivers **noisy-neighbor protection**, **data isolation**, and **audit-ready compliance evidence** for FedRAMP and NIST RMF.

**2.4 Zero Trust by Design**

**Understanding:**  
OMB M-22-09 and CISA Zero Trust Maturity Model mandate continuous verification at every layer. This repo enforces Zero Trust principles in **identity, network, runtime, and pipeline**.

**Approach:**

* **Identity:** Cognito/OIDC federation, IAM least privilege, IRSA.
* **Network:** Private-only VPC subnets, FIPS endpoints, and WAF.
* **Runtime:** Kyverno admission policies enforcing signed images + PodSecurity.
* **Pipeline:** Supply chain integrity enforced via SBOMs, cosign signatures, and manual approvals.

**Proof in Repo:**

* ops/policies/kyverno-verifyimages.yaml
* ops/policies/kyverno-podsecurity.yaml
* deploy/preflight-vpc.sh

**Benefit:**  
Creates a **compliance-first SaaS platform** where every layer is secured, monitored, and auditable.

**Chapter 3 — Infrastructure Layer**

**3.1 AWS Accounts, VPCs, Subnets, Security Groups, Endpoints**

**Understanding:**  
A compliant SaaS must separate workloads across accounts and enforce **least-privilege networking**. The repo uses AWS Organizations and CloudFormation templates to provision accounts and VPCs for tooling (CI/CD), dev, stg, and prod.

**Approach:**

* **Accounts:** Tooling account for CI/CD, workload accounts for dev/stg/prod, optional dedicated accounts for stamps.
* **VPCs:** Dedicated VPCs per environment. Private subnets for workloads, public subnets only for load balancers.
* **Subnets:** Multi-AZ private subnets to ensure HA.
* **Security Groups:** Restrictive ingress/egress, allowing only required service flows (e.g., ALB → EKS, EKS → RDS).
* **Endpoints:** VPC Interface Endpoints enforce FIPS 140-2 service calls for S3, ECR, KMS, STS, CodeBuild, Logs.

**Tools & Platforms:**

* AWS Organizations, VPC, SGs, VPC Endpoints
* IaC via templates/\*.yaml, params/\*.json

**Proof in Repo:**

* deploy/preflight-vpc.sh (validation of endpoints)
* templates/tools-pipeline.yaml
* params/tools-\*.json

**Benefit:**  
Network isolation, FIPS enforcement, and account separation ensure **Zero Trust**, prevent data exfiltration, and provide audit-ready evidence.

**3.2 EKS Clusters (dev/stg/prod)**

**Understanding:**  
All workloads must move through gated environments with **parity across clusters** to ensure consistency.

**Approach:**

* **EKS Clusters:** One per environment (eks-dev, eks-stg, eks-prod).
* **Parity:** Configured identically using Helm and CloudFormation.
* **IAM Governance:** Deploy roles scoped per environment and mapped into EKS aws-auth ConfigMaps.

**Tools & Platforms:**

* Amazon EKS
* IAM + IRSA
* Helm

**Proof in Repo:**

* ops/helm/values-dev.yaml
* ops/helm/values-stg.yaml
* ops/helm/values-prod.yaml
* templates/workload-deploy-iam.yaml

**Benefit:**  
Supports **promotion gates (dev → stg → prod)**, ensures consistency, and enforces least privilege per environment.

**3.3 Add-ons (ALB Controller, Metrics Server, OIDC Integration)**

**Understanding:**  
Certain add-ons are required for scaling, ingress, and secure identity integration.

**Approach:**

* **AWS Load Balancer Controller:** Provides ALBs for tenant ingress.
* **Metrics Server:** Required for HPA to scale workloads.
* **OIDC Integration:** Associates each EKS cluster with an OIDC provider, enabling IRSA.

**Tools & Platforms:**

* ALB Controller Helm chart
* Kubernetes Metrics Server
* AWS OIDC Provider

**Proof in Repo:**

* ops/helm/templates/\*
* Cluster OIDC setup documented in deploy/bootstrap-fill-and-deploy.sh

**Benefit:**  
Ensures **elastic scale**, **secure ingress**, and **tenant-aware IAM integration**.

**3.4 Preflight Validation Scripts (deploy/preflight-vpc.sh)**

**Understanding:**  
Before deploying workloads, the architecture must validate network and FIPS readiness.

**Approach:**

* Script checks for required VPC endpoints: ecr.api, ecr.dkr, s3, logs, sts, kms, codecommit, codebuild, events, ec2, eks.
* Fails early if any endpoint is missing.
* Enforces private-only builds (no NAT/public egress).

**Tools & Platforms:**

* Shell script in deploy/preflight-vpc.sh

**Proof in Repo:**

* deploy/preflight-vpc.sh

**Benefit:**  
Prevents misconfigurations, enforces **Zero Trust networking**, and guarantees **FIPS-compliant API calls** before workloads ever launch.

**3.5 FIPS Endpoints, VPC-only, Compliance Controls**

**Understanding:**  
Ongoing compliance requires **all cryptographic operations and service calls** to run through FIPS 140-2 validated endpoints.

**Approach:**

* VPC Endpoints configured with FIPS DNS suffixes.
* CodeBuild/CodePipeline run in private subnets, no internet egress.
* Security Groups and Network ACLs ensure least-privilege flows.

**Tools & Platforms:**

* AWS KMS, ECR, S3, STS with FIPS endpoints
* IaC templates + validation scripts

**Proof in Repo:**

* deploy/preflight-vpc.sh
* templates/tools-pipeline.yaml

**Benefit:**  
Delivers **compliance-first assurance** (FedRAMP High, FIPS 140-2, NIST RMF), prevents audit failures, and reduces ATO timelines.

**Chapter 4 — CI/CD Pipeline Layer**

**4.1 Full CodePipeline Flow (Source → Build → Scan → Sign/Attest → Approval → Deploy → Mirror)**

**Understanding:**  
Section 10 of the architecture requires a **secure, auditable CI/CD pipeline** that enforces supply chain security, gated approvals, and parity between Commercial and GovCloud.

**Approach:**

* **Source:** CodeCommit repository serves as source of truth.
* **Build:** CodeBuild compiles workloads into container images.
* **Scan:** SBOMs generated (Syft), vulnerabilities scanned (Trivy).
* **Sign/Attest:** Cosign signs artifacts with AWS KMS asymmetric keys.
* **Approval:** Manual approvals inserted between dev → stg → prod.
* **Deploy:** Helm deploys immutable digests into EKS clusters.
* **Mirror:** Signed + attested artifacts mirrored to GovCloud ECR.

**Tools & Platforms:**

* AWS CodePipeline, CodeCommit, CodeBuild, KMS, ECR, Helm

**Proof in Repo:**

* templates/tools-pipeline.yaml
* ops/pipeline/buildspec-build.yml
* ops/pipeline/buildspec-deploy.yml
* ops/pipeline/buildspec-mirror-gov.yml

**Benefit:**  
Provides an **end-to-end immutable chain of custody**, enforcing **compliance-first deployments**.

**4.2 Why CodePipeline + CodeBuild (FedRAMP Native)**

**Understanding:**  
Only AWS-native CI/CD services are FedRAMP High authorized in both Commercial and GovCloud.

**Approach:**

* **CodePipeline:** Orchestrates CI/CD flow, approvals, notifications.
* **CodeBuild:** Executes builds in private subnets, ensuring all service calls use **FIPS endpoints**.
* **Native Integration:** CloudWatch, IAM, KMS, and SNS integrated out-of-the-box.

**Proof in Repo:**

* templates/tools-pipeline.yaml

**Benefit:**  
Reduces compliance burden, eliminates third-party tools, and ensures **ATO alignment**.

**4.3 Tools Inside CodeBuild (Syft, Trivy, Cosign)**

**Understanding:**  
Pipeline security depends on **artifact transparency and cryptographic assurance**.

**Approach:**

* **Syft:** Generates SBOMs in CycloneDX format.
* **Trivy:** Scans for vulnerabilities, failing builds on HIGH/CRITICAL.
* **Cosign:** Signs images + SBOMs with AWS KMS asymmetric key; attestations stored alongside images.

**Proof in Repo:**

* ops/pipeline/buildspec-build.yml

**Benefit:**  
Ensures **supply chain integrity**, satisfies EO 14028, NIST 800-53, and FedRAMP vulnerability management.

**4.4 FIPS Enforcement (VPC Endpoints)**

**Understanding:**  
FIPS 140-2 requires validated cryptographic modules for all service calls.

**Approach:**

* CodeBuild jobs run inside private subnets, no internet egress.
* Required VPC endpoints validated via deploy/preflight-vpc.sh.
* All API calls routed through FIPS DNS suffixes for S3, ECR, KMS, STS, CodeBuild, CodeCommit.

**Proof in Repo:**

* deploy/preflight-vpc.sh
* params/tools-\*.json

**Benefit:**  
Guarantees **federal-grade compliance** while preventing data exfiltration.

**4.5 Repo Proof: Buildspecs + Pipeline Templates**

**Understanding:**  
Auditors need **traceability from requirements → repo artifacts**.

**Approach:**

* **Infrastructure:** Defined in templates/tools-pipeline.yaml.
* **Buildspecs:**
  + ops/pipeline/buildspec-build.yml → Build, scan, sign, attest
  + ops/pipeline/buildspec-deploy.yml → Deploy to EKS clusters by digest
  + ops/pipeline/buildspec-mirror-gov.yml → Mirror signed artifacts into GovCloud
* **IAM Roles:** Defined in templates/workload-deploy-iam.yaml

**Benefit:**  
Delivers **deterministic, auditable pipelines**, with compliance evidence stored in S3 (SBOMs, vulnerability reports, attestations).

**Chapter 5 — Security & Compliance**

**5.1 Runtime Enforcement (Kyverno verifyImages, PodSecurity)**

**Understanding:**  
Section 9.2 of the architecture requires **runtime enforcement of supply chain integrity and baseline pod security**.

**Approach:**

* **verifyImages Policy:** Kyverno admission controller ensures only container images signed with KMS-backed Cosign keys are admitted.
* **PodSecurity Policies:** Enforce non-root users, read-only root filesystem, seccomp profiles, and dropped Linux capabilities.
* **Fail Fast:** Unsigned or non-compliant pods are blocked before runtime.

**Proof in Repo:**

* ops/policies/kyverno-verifyimages.yaml
* ops/policies/kyverno-podsecurity.yaml

**Benefit:**  
Guarantees **Zero Trust at runtime**, provides auditors with **evidence of enforcement**, and blocks risky workloads.

**5.2 IAM Least Privilege**

**Understanding:**  
Strict IAM role separation prevents privilege escalation and aligns with NIST AC family controls.

**Approach:**

* **Environment-Specific Roles:** Separate deploy roles for dev, stg, prod.
* **Scoped Permissions:** Deploy roles grant only EKS + Helm privileges.
* **No Long-Lived Keys:** All role assumption done via IRSA/OIDC federation.

**Proof in Repo:**

* templates/workload-deploy-iam.yaml

**Benefit:**  
Minimizes blast radius, enforces least privilege, and aligns with **FedRAMP/NIST 800-53 AC controls**.

**5.3 IRSA ServiceAccounts & OIDC Federation**

**Understanding:**  
Static credentials are not compliant in a multi-tenant SaaS. Workloads must assume scoped IAM roles dynamically.

**Approach:**

* **Cluster OIDC Provider:** Each EKS cluster is integrated with OIDC.
* **IRSA:** ServiceAccounts in Kubernetes map to scoped IAM roles.
* **Tenant Workloads:** Each tenant namespace uses its own ServiceAccount + IAM role.

**Proof in Repo:**

* templates/workload-deploy-iam.yaml
* OIDC configuration in deploy/bootstrap-fill-and-deploy.sh

**Benefit:**  
Eliminates hardcoded secrets, enables **per-tenant IAM isolation**, and supports CAC/PIV SAML federation.

**5.4 Zero Trust Posture**

**Understanding:**  
No workload, user, or service is implicitly trusted. Continuous verification is required across the stack.

**Approach:**

* **Network:** Private-only subnets + VPC endpoints enforce FIPS traffic.
* **Identity:** OIDC federation + IAM least privilege.
* **Runtime:** Kyverno enforces signed artifacts and pod security baselines.
* **Pipeline:** CodePipeline approvals provide human-in-the-loop governance.

**Proof in Repo:**

* deploy/preflight-vpc.sh
* ops/policies/\*
* templates/tools-pipeline.yaml

**Benefit:**  
Aligns with **OMB M-22-09 Zero Trust strategy**, ensures constant verification, and prevents lateral compromise.

**5.5 Compliance Tie-In (FedRAMP, RMF, FIPS baked in)**

**Understanding:**  
Compliance is baked into the repo — not bolted on.

**Approach:**

* **FedRAMP High:** Only authorized AWS services (EKS, CodePipeline, CodeBuild, S3, ECR).
* **FIPS 140-2:** All service calls route through validated endpoints.
* **RMF (NIST 800-37):** Controls mapped across identity, access, monitoring, backup.
* **Evidence:** SBOMs, vulnerability scans, cosign attestations stored in encrypted S3.

**Proof in Repo:**

* ops/pipeline/buildspec-build.yml
* ops/policies/kyverno-verifyimages.yaml
* templates/tools-pipeline.yaml

**Benefit:**  
Provides a **compliance-first platform**, reducing ATO timelines and delivering audit evidence with every build.

**5.6 Secrets Management (Secrets Manager + External Secrets Operator)**

**Understanding:**  
Applications require access to DB creds, API tokens, and SMTP keys. Hardcoding or storing in ConfigMaps violates compliance.

**Approach:**

* **AWS Secrets Manager / SSM Parameter Store:** Stores sensitive values with KMS encryption.
* **External Secrets Operator (ESO):** Syncs secrets from AWS into Kubernetes namespaces dynamically.
* **IAM Controls:** IRSA restricts access so only the right tenant workloads pull their secrets.

**Proof in Repo:**

* ops/policies/app-iam.yaml (workload access policies)
* ESO references in deploy/bootstrap-fill-and-deploy.sh

**Benefit:**  
Eliminates secret sprawl, provides rotation capabilities, and ensures **audit logs for every secret access**.

**Chapter 6 — Supply Chain Security**

**6.1 SBOM Generation (Syft)**

**Understanding:**  
Executive Order 14028 and NIST controls require full transparency into software supply chains via **Software Bill of Materials (SBOMs)**.

**Approach:**

* Every build generates a CycloneDX-format SBOM using **Syft**.
* SBOM artifacts stored in S3 alongside logs and images.
* SBOMs also published to OCI registries as attestations.

**Tools & Platforms:**

* Syft (via CodeBuild container)
* AWS CodeBuild
* Amazon S3

**Proof in Repo:**

* ops/pipeline/buildspec-build.yml (SBOM generation step)

**Benefit:**  
Provides visibility into dependencies, accelerates vulnerability management, and satisfies **FedRAMP/NIST SBOM mandates**.

**6.2 Vulnerability Scanning (Trivy)**

**Understanding:**  
FedRAMP requires continuous monitoring and vulnerability management. Builds must fail if HIGH/CRITICAL vulnerabilities exist.

**Approach:**

* Container images scanned in CodeBuild using **Trivy** against CVE databases.
* Policy: fail build if HIGH/CRITICAL issues are detected.
* Scan reports archived in encrypted S3.

**Tools & Platforms:**

* Trivy
* AWS CodeBuild
* Amazon S3

**Proof in Repo:**

* ops/pipeline/buildspec-build.yml (scan stage)

**Benefit:**  
Prevents insecure workloads from reaching runtime, aligns with **NIST 800-53 CM-8** (System Component Inventory).

**6.3 Signing + Attestation with KMS (Cosign)**

**Understanding:**  
Unsigned artifacts can be tampered with in transit. Compliance requires **cryptographic verification** of builds.

**Approach:**

* **Cosign** signs container images and SBOMs using **AWS KMS asymmetric keys**.
* Signatures stored in ECR alongside images.
* Attestations prove the image was built, scanned, and signed in a controlled environment.

**Tools & Platforms:**

* Cosign
* AWS KMS (asymmetric keys)
* Amazon ECR

**Proof in Repo:**

* ops/pipeline/buildspec-build.yml (sign + attest step)

**Benefit:**  
Guarantees artifact provenance, integrity, and compliance with **NIST 800-53 SC-12** and FedRAMP requirements.

**6.4 Kyverno Admission Checks (verifyImages, PodSecurity)**

**Understanding:**  
Even with signatures, enforcement must occur at runtime to prevent unsigned workloads from bypassing the pipeline.

**Approach:**

* Kyverno admission controller enforces **verifyImages** — rejecting pods that lack valid cosign signatures.
* Additional PodSecurity rules enforce non-root, seccomp, and read-only root FS.
* Policies applied cluster-wide at bootstrap.

**Tools & Platforms:**

* Kyverno Admission Controller

**Proof in Repo:**

* ops/policies/kyverno-verifyimages.yaml
* ops/policies/kyverno-podsecurity.yaml

**Benefit:**  
Ensures **Zero Trust for workloads**, blocks unsigned or tampered containers, and provides runtime compliance proof.

**6.5 Immutable ECR Repositories**

**Understanding:**  
Images must be immutable to prevent post-build tampering.

**Approach:**

* ECR repos configured as immutable — tags cannot be overwritten.
* Lifecycle policies remove stale images, reducing attack surface.
* Scan-on-push enabled for early vulnerability detection.

**Tools & Platforms:**

* Amazon ECR
* IaC via CloudFormation

**Proof in Repo:**

* templates/tools-pipeline.yaml (ECR config)

**Benefit:**  
Guarantees a **tamper-proof chain of custody** from build → deploy, supporting audit requirements for supply chain integrity.

**Chapter 7 — Tenant Lifecycle**

**7.1 Namespace-per-Tenant**

**Understanding:**  
To support **multi-tenancy** without data leakage, each tenant must have its own namespace boundary inside EKS.

**Approach:**

* A namespace is provisioned for each tenant.
* Tenant-specific objects (ConfigMaps, Secrets, Deployments, Services) live inside the namespace.
* Labels and annotations applied for governance, billing, and observability.

**Tools & Platforms:**

* Amazon EKS
* Kubernetes Namespaces
* Helm templates

**Proof in Repo:**

* ops/tenant/values-tenant-template.yaml

**Benefit:**  
Provides **logical isolation**, supports **tenant tagging for billing**, and meets FedRAMP multi-tenancy requirements.

**7.2 Helm Tenant Templates**

**Understanding:**  
Onboarding must be **repeatable and automated**. Helm templates standardize tenant configuration.

**Approach:**

* values-tenant-template.yaml defines tenant defaults (quotas, ingress, policies).
* Operators fill in tenant-specific values (TenantID, quotas, RBAC).
* Helm deploys workloads consistently across dev, stg, prod.

**Tools & Platforms:**

* Helm
* Kubernetes
* Amazon EKS

**Proof in Repo:**

* ops/tenant/values-tenant-template.yaml

**Benefit:**  
Reduces onboarding time, eliminates misconfigurations, and ensures **consistency across environments**.

**7.3 Quotas & Policies**

**Understanding:**  
Without guardrails, tenants could overconsume resources or impact neighbors (“noisy neighbor problem”).

**Approach:**

* **ResourceQuotas:** Cap CPU, memory, and storage per tenant.
* **LimitRanges:** Enforce per-container request/limit ranges.
* **NetworkPolicies:** Restrict pod-to-pod and pod-to-service traffic, preventing lateral movement.
* **RBAC:** Scopes tenant admin roles to their namespace only.

**Tools & Platforms:**

* Kubernetes ResourceQuotas, LimitRanges, NetworkPolicies, RBAC

**Proof in Repo:**

* ops/k8s/tenant-resourcequota.yaml
* ops/k8s/tenant-networkpolicy.yaml

**Benefit:**  
Prevents noisy-neighbor risk, enforces **data and traffic isolation**, and satisfies Zero Trust requirements.

**7.4 Onboarding Automation (bootstrap-fill-and-deploy.sh)**

**Understanding:**  
Onboarding must be **fast, auditable, and low-risk**. Manual steps introduce errors.

**Approach:**

* bootstrap-fill-and-deploy.sh provisions namespaces, applies Helm tenant templates, and injects IAM roles.
* Guardrails (NetworkPolicies, ResourceQuotas) applied automatically at Day 1.
* Outputs (Helm values, IAM ARNs, policies) logged to S3 for audit evidence.

**Tools & Platforms:**

* Shell automation
* Helm
* IAM + IRSA

**Proof in Repo:**

* deploy/bootstrap-fill-and-deploy.sh

**Benefit:**  
Enables **rapid, repeatable tenant onboarding** while ensuring compliance guardrails are applied immediately.

**7.5 Noisy-Neighbor Protection & Isolation Models**

**Understanding:**  
The SaaS must serve both **commercial tenants** (cost-optimized pooled/siloed) and **federal tenants** (isolated dedicated stamps).

**Approach:**

* **Pooled/Silo Models:** Use quotas, network policies, and RBAC to enforce logical isolation.
* **Dedicated Stamps:** Provision full stacks (VPC, EKS, CI/CD) for agency workloads requiring maximum assurance.
* **Noisy Neighbor Protection:** LimitRanges + Quotas enforce fairness in pooled models.

**Proof in Repo:**

* ops/tenant/values-tenant-template.yaml
* ops/k8s/tenant-networkpolicy.yaml
* ops/k8s/tenant-resourcequota.yaml

**Benefit:**  
Provides **mission-appropriate isolation**, prevents tenant conflicts, and supports both **commercial efficiency** and **federal compliance requirements**.

**Chapter 8 — Public vs Private Partition (Commercial vs GovCloud)**

**8.1 Commercial → GovCloud Mirroring**

**Understanding:**  
GovCloud has a **restricted AWS service catalog** and cannot run advanced build/scanning stages. To remain compliant, all artifacts must originate from a trusted pipeline in Commercial, then be mirrored into GovCloud.

**Approach:**

* **Builds in Commercial:** CodePipeline + CodeBuild generate SBOMs, run Trivy scans, and sign artifacts with Cosign/KMS.
* **Mirroring:** Signed artifacts mirrored into GovCloud ECR via gov-mirror-receiver.yaml.
* **Trust Boundary:** GovCloud workloads only consume artifacts that are **already built, scanned, signed, and attested**.

**Tools & Platforms:**

* AWS CodePipeline (Commercial)
* AWS ECR cross-partition replication
* AWS KMS signing (Cosign integration)

**Proof in Repo:**

* ops/pipeline/buildspec-mirror-gov.yml
* templates/gov-mirror-receiver.yaml

**Benefit:**  
Maintains a **single trusted build pipeline**, enforces artifact provenance, and prevents duplicate infrastructure in GovCloud.

**8.2 GovCloud Restrictions (No Public Endpoints)**

**Understanding:**  
GovCloud mandates **FIPS endpoints and private-only execution**. Public endpoints and internet egress are prohibited.

**Approach:**

* All GovCloud builds run in private subnets.
* Required VPC endpoints created for ECR, S3, STS, KMS, Logs, CodeBuild.
* **No NAT Gateway** configured — workloads cannot reach public internet.
* Mirrored artifacts from Commercial provide the only ingress into GovCloud pipelines.

**Tools & Platforms:**

* AWS GovCloud (US) partition
* VPC endpoints + private subnets
* CodeBuild (VPC-only mode)

**Proof in Repo:**

* deploy/preflight-vpc.sh (validates endpoints)
* templates/tools-pipeline.yaml

**Benefit:**  
Guarantees **FIPS 140-2 compliance**, prevents data exfiltration, and meets ITAR/CJIS/DoD SRG restrictions.

**8.3 Why This Dual Setup is Necessary**

**Understanding:**  
Federal workloads cannot operate solely in AWS Commercial due to regulatory constraints. At the same time, GovCloud lacks the service richness needed for SBOMs, scanning, and signing. A dual-partition design satisfies both.

**Approach:**

* **Commercial:** Full build + scan + sign with SBOM generation.
* **GovCloud:** Restricted runtime environment that consumes only pre-validated artifacts.
* **Parity:** Same Helm + IaC deployed in both partitions for consistency.

**Benefit:**

* Meets **federal compliance requirements** (ITAR, DoD SRG, CJIS).
* Leverages **Commercial flexibility** for advanced CI/CD.
* Provides **audit-ready consistency** between Commercial and GovCloud.

**Chapter 9 — Approval & Governance**

**9.1 Manual Approval Stages (dev → stg → prod)**

**Understanding:**  
In regulated SaaS environments, **human-in-the-loop approvals** are mandatory before workloads advance into higher environments.

**Approach:**

* Manual approval actions inserted between **dev → stg** and **stg → prod** in CodePipeline.
* Approvers restricted by IAM RBAC (e.g., release managers, security officers).
* Approvals require review of SBOMs, Trivy reports, and cosign attestations before release.

**Tools & Platforms:**

* AWS CodePipeline Manual Approval
* AWS IAM

**Proof in Repo:**

* templates/tools-pipeline.yaml

**Benefit:**  
Adds a **compliance gate** that prevents unverified artifacts from being promoted into production.

**9.2 Signed Artifact Verification**

**Understanding:**  
Auditors require cryptographic proof that only **signed, attested artifacts** are deployed.

**Approach:**

* **Pipeline Gate:** CodePipeline buildspecs validate cosign signatures before Helm deploy.
* **Runtime Admission:** Kyverno verifyImages policy blocks unsigned workloads from entering the cluster.
* **Dual Enforcement:** Both pipeline and runtime checks required.

**Tools & Platforms:**

* Cosign + AWS KMS
* Kyverno Admission Controller

**Proof in Repo:**

* ops/pipeline/buildspec-deploy.yml
* ops/policies/kyverno-verifyimages.yaml

**Benefit:**  
Provides **end-to-end provenance**, from pipeline to runtime, ensuring **supply chain integrity**.

**9.3 Notifications & Event Routing (SNS, EventBridge, Slack/ServiceNow)**

**Understanding:**  
Governance requires **visibility into pipeline events** and approvals for stakeholders.

**Approach:**

* **SNS Topics:** Send notifications on build success/failure, approval needed, and deploy complete.
* **EventBridge Rules:** Capture pipeline state changes, enabling downstream workflows (e.g., ServiceNow ticket creation, Slack alerts).
* **Email / ChatOps Integration:** Optional subscription for direct stakeholder alerts.

**Tools & Platforms:**

* Amazon SNS
* Amazon EventBridge
* Third-party integrations (Slack, ServiceNow)

**Proof in Repo:**

* templates/pipeline-notifications.yaml

**Benefit:**  
Provides **real-time visibility** into pipeline events, ensures accountability, and maintains **audit trails**.

**9.4 Ops Handoff**

**Understanding:**  
Once workloads hit production, **operations teams** assume responsibility for monitoring, patching, and incident response.

**Approach:**

* Signed artifact metadata (SBOMs, signatures, vulnerability reports) delivered to Ops via S3 + SNS.
* Ops dashboards (Grafana, CloudWatch) provide observability and cost overlays.
* CI/CD pipeline ensures all evidence is preserved for compliance audits.

**Tools & Platforms:**

* Amazon S3
* SNS handoff notifications
* CloudWatch Dashboards

**Proof in Repo:**

* ops/pipeline/buildspec-deploy.yml
* templates/pipeline-notifications.yaml

**Benefit:**  
Ensures a **clear separation of responsibilities**, provides **immutable compliance evidence** for Ops, and reduces deployment risk.

**Chapter 10 — Observability & Operations**

**10.1 Logging (CloudWatch, S3, Fluent Bit, ELK optional)**

**Understanding:**  
Compliance requires **centralized, immutable log storage** with retention policies. Logs must be queryable for audits and incident response.

**Approach:**

* **CloudWatch Logs:** EKS cluster logs, application logs, and pipeline logs ingested by default.
* **S3 Archival:** Logs shipped to encrypted S3 with lifecycle policies for long-term retention.
* **Fluent Bit/ELK (Optional):** Forward logs to Elasticsearch for advanced analysis if required.

**Tools & Platforms:**

* Amazon CloudWatch Logs
* Amazon S3
* Fluent Bit / ELK (optional)

**Proof in Repo:**

* ops/helm/templates/fluent-bit.yaml

**Benefit:**  
Provides **immutable audit trails**, supports RMF evidence requirements, and enables forensic investigation.

**10.2 Metrics (CloudWatch Container Insights, AMP/Grafana)**

**Understanding:**  
Metrics provide visibility into tenant workloads, cluster health, and SLA compliance.

**Approach:**

* **CloudWatch Container Insights:** Captures CPU, memory, disk, and network metrics.
* **Amazon Managed Prometheus (AMP):** Scrapes Prometheus exporters at scale.
* **Amazon Managed Grafana (AMG):** Provides dashboards with per-tenant and system-wide visibility.

**Tools & Platforms:**

* CloudWatch Container Insights
* AMP + AMG
* Grafana dashboards (dashboards/\*.json)

**Proof in Repo:**

* ops/helm/templates/prometheus-exporter.yaml
* dashboards/\*

**Benefit:**  
Supports **proactive monitoring**, SLA validation, and cost/performance optimization.

**10.3 Tracing (X-Ray / OpenTelemetry)**

**Understanding:**  
Multi-tenant SaaS requires tracing across services for troubleshooting and compliance monitoring.

**Approach:**

* **AWS X-Ray:** Provides request tracing across APIs and microservices.
* **OpenTelemetry (OTel):** Standardizes traces, integrates with X-Ray and external observability platforms.
* **Tenant-Aware Tags:** Traces labeled with TenantID for per-customer visibility.

**Tools & Platforms:**

* AWS X-Ray
* OpenTelemetry SDK + Collector

**Proof in Repo:**

* ops/k8s/otelsidecar.yaml

**Benefit:**  
Improves **MTTD/MTTR**, enables tenant-specific performance reporting, and supports compliance investigations.

**10.4 Cost Allocation (Tags, CUR, Grafana Overlays)**

**Understanding:**  
Accurate cost allocation per tenant is required for billing, chargeback, and compliance reporting.

**Approach:**

* **Resource Tagging:** TenantID, Environment, CostCenter applied to all resources.
* **AWS CUR (Cost & Usage Report):** Filtered by tags to generate per-tenant breakdowns.
* **Grafana Dashboards:** Display cost overlays alongside performance metrics.

**Tools & Platforms:**

* AWS Resource Tagging
* AWS CUR + Cost Explorer
* Grafana

**Proof in Repo:**

* templates/tools-pipeline.yaml (tag enforcement)
* ops/tenant/values-tenant-template.yaml

**Benefit:**  
Provides **transparent cost reporting**, ensures fairness, and enables compliance-friendly billing models.

**10.5 Backup, DR & Lifecycle Management**

**Understanding:**  
Disaster Recovery (DR) is required under **FedRAMP contingency planning controls**.

**Approach:**

* **EBS/EFS Backups:** Scheduled backups via AWS Backup, encrypted with KMS.
* **Cross-Region Replication:** Critical S3 buckets and ECR images replicated for redundancy.
* **Velero:** Backups Kubernetes cluster state and tenant namespaces.
* **Lifecycle Policies:** Automatically expire old images and logs.

**Tools & Platforms:**

* AWS Backup
* Velero
* Amazon S3 / ECR lifecycle policies

**Proof in Repo:**

* ops/k8s/backup-policies.yaml
* templates/tools-pipeline.yaml

**Benefit:**  
Ensures **continuity of operations**, validates RTO/RPO objectives, and delivers audit-ready recovery evidence.

**10.6 Cost & Resilience Guardrails**

**Understanding:**  
Multi-tenant SaaS must balance **efficiency and resilience** to avoid unnecessary spend.

**Approach:**

* **AWS Budgets:** Monitors tenant-level costs and triggers alerts.
* **ECR Lifecycle Policies:** Prevent registry bloat by retaining ~100 images.
* **Resilience Drills:** Regular failover testing for DR validation.
* **Karpenter Replacement (if GovCloud compliant):** For Commercial, Karpenter consolidates workloads; in GovCloud, use managed node groups.

**Tools & Platforms:**

* AWS Budgets
* AWS Backup + DR runbooks
* ECR lifecycle management

**Proof in Repo:**

* ops/k8s/backup-policies.yaml
* templates/tools-pipeline.yaml

**Benefit:**  
Delivers **predictable costs**, enforces **tenant fairness**, and strengthens **resilience posture** across environments.

**Chapter 11 — Criteria Compliance Matrix**

**Understanding:**  
Section 18 of the AWS\_MultiTenant\_SaaS\_EKS\_FIPS\_Architecture requires a **repeatable, auditable compliance mapping**. This matrix shows how repo artifacts fulfill FedRAMP, FIPS 140-2, NIST 800-53, and RMF controls.

| **Control Requirement** | **Architecture Doc Section** | **Repo File(s)** | **Compliance Justification** |
| --- | --- | --- | --- |
| **Supply Chain Integrity** | Sec 9.2 (Platform Security Controls) | ops/pipeline/buildspec-build.yml, ops/policies/kyverno-verifyimages.yaml | All images scanned, signed, attested, and enforced at runtime. Ensures EO 14028 + FedRAMP compliance. |
| **Tenant Isolation** | Sec 4 (Multi-Tenant Isolation Models) | ops/tenant/values-tenant-template.yaml, ops/k8s/tenant-networkpolicy.yaml, ops/k8s/tenant-resourcequota.yaml | Enforces namespace, network, and resource boundaries. Prevents cross-tenant access. |
| **FIPS Endpoint Enforcement** | Sec 3.3 & 10.2 | deploy/preflight-vpc.sh, templates/tools-pipeline.yaml, params/tools-\*.json | All builds + API calls restricted to private subnets and FIPS 140-2 validated endpoints. |
| **Least Privilege IAM** | Sec 5.2 (IAM Guardrails) | templates/workload-deploy-iam.yaml | Deploy roles scoped per environment; workloads assume scoped IRSA roles. |
| **Runtime Security** | Sec 9.2 | ops/policies/kyverno-podsecurity.yaml | Pods run non-root, with seccomp, read-only FS, and dropped capabilities. |
| **CI/CD Pipeline Security** | Sec 10 (CI/CD Requirements) | ops/pipeline/buildspec-deploy.yml, ops/pipeline/buildspec-mirror-gov.yml, templates/tools-pipeline.yaml | Pipeline enforces immutable digests, manual approvals, and GovCloud mirroring. |
| **Identity Federation** | Sec 10.2 (Identity & Access) | templates/workload-deploy-iam.yaml, OIDC configs | Cognito/OIDC federation for users; IRSA for workloads; no static credentials. |
| **Approvals & Governance** | Sec 10 (Approvals) | templates/tools-pipeline.yaml, templates/pipeline-notifications.yaml | Manual approvals inserted in pipeline; SNS/EventBridge notify stakeholders. |
| **Observability** | Sec 12 (Operational Guardrails) | ops/helm/templates/fluent-bit.yaml, ops/k8s/otelsidecar.yaml, dashboards/\* | Logs, metrics, and traces captured via CW/AMP/AMG/X-Ray. |
| **Backup & DR** | Sec 18 (Implementation Plan) | ops/k8s/backup-policies.yaml, templates/tools-pipeline.yaml | Automated snapshots, cross-region replication, and recovery drills ensure continuity. |
| **Immutable Repos** | Sec 10.2 (Artifact Control) | templates/tools-pipeline.yaml (ECR config) | ECR repos immutable, lifecycle policies enabled, prevents overwrite tampering. |
| **Compliance Evidence** | Sec 18 (Audit Readiness) | S3 artifact storage, ops/pipeline/buildspec-build.yml | SBOMs, scan reports, signatures archived securely for auditors. |
| **Secrets Management** | Sec 9.2 (Security Controls) | ops/policies/app-iam.yaml, deploy/bootstrap-fill-and-deploy.sh | Secrets in AWS Secrets Manager, synced with External Secrets Operator; per-tenant IRSA access. |
| **Zero Trust Enforcement** | OMB M-22-09, Sec 9.2 | ops/policies/kyverno-verifyimages.yaml, deploy/preflight-vpc.sh | Continuous verification enforced across network, identity, runtime, and pipeline. |

**Benefit:**  
This matrix makes the repo an **audit-ready package**: every requirement maps to a file and justification, allowing compliance teams to validate FedRAMP, FIPS, and RMF quickly.

**Chapter 12 — Conclusion**

**12.1 How the Repo + Process Fulfill Every Requirement**

The **AWS EKS CI/CD (Commercial + GovCloud) via CloudFormation** repository is not just infrastructure code — it is a **compliance-enforcing SaaS delivery framework**. Every repo artifact maps directly to architecture requirements and federal mandates.

* **Multi-Tenant SaaS Principles (Section 4):**  
  Implemented via Kubernetes namespaces, Helm templates, ResourceQuotas, and NetworkPolicies (ops/tenant/\*, ops/k8s/\*). Supports pooled, siloed, and dedicated tenant models from a single codebase.
* **Infrastructure and FIPS Controls (Sections 3.3 & 10.2):**  
  VPC-only subnets, private endpoints, and FIPS-validated service calls enforced via preflight validation (deploy/preflight-vpc.sh, templates/tools-pipeline.yaml).
* **CI/CD Pipeline Security (Section 10):**  
  End-to-end CodePipeline stages enforce build → scan → sign → attest → approval → deploy → mirror. Repo-defined buildspecs (ops/pipeline/\*) and IAM templates (templates/workload-deploy-iam.yaml) ensure controlled promotion and GovCloud parity.
* **Supply Chain Security (Section 9.2):**  
  SBOM generation (Syft), vulnerability scanning (Trivy), signing + attestations (Cosign), and runtime enforcement (Kyverno policies) guarantee provenance and integrity.
* **Observability and Ops Guardrails (Section 12):**  
  Logging (CloudWatch/Fluent Bit), metrics (AMP/AMG), tracing (X-Ray/OTel), and dashboards (dashboards/\*) deliver full tenant-aware visibility and SLA monitoring.
* **Compliance Evidence and RMF Tie-In (Section 18):**  
  All SBOMs, scan reports, signatures, and pipeline logs stored in encrypted S3, producing an immutable audit trail for FedRAMP, FIPS, and RMF validation.

**12.2 Why This Design Is Production-Ready, Auditable, and Extensible**

**Production-Ready:**

* Runs on AWS-native FedRAMP High services (EKS, CodePipeline, CodeBuild, GovCloud).
* Enforces Zero Trust across network, identity, runtime, and CI/CD layers.
* Proven onboarding automation (deploy/bootstrap-fill-and-deploy.sh) for tenants.

**Auditable:**

* Cryptographic signatures and SBOMs provide **tamper-proof provenance**.
* Runtime policies enforce compliance continuously, not just at build time.
* Criteria Compliance Matrix (Chapter 11) provides **direct audit mapping**.

**Extensible:**

* Single repo supports pooled, siloed, and dedicated stamps.
* Helm + CloudFormation modularity allows adding new tenants, policies, and services without architectural redesign.
* Dual-partition (Commercial + GovCloud) model ensures consistent operations across sensitive and regulated environments.

**Final Statement**

This repository and proposal provide a **blueprint SaaS platform** that is:

* **Secure**: Zero Trust enforced across compute, identity, and runtime.
* **Compliant**: FedRAMP, FIPS 140-2, and RMF controls built into every layer.
* **Scalable**: Supports elastic onboarding across pooled, siloed, and dedicated models.
* **Auditable**: Immutable evidence stored in encrypted S3, traceable to repo files.
* **Extensible**: Modular architecture ensures future-proof growth and customer adoption.

It is a **production-ready SaaS framework** for Commercial and Government customers, designed to withstand compliance scrutiny, deliver operational excellence, and enable rapid, secure adoption of Zero Trust principles.

**Appendices**

**Appendix A — AWS Services by Function**

**1. SaaS Platform / Application Layer**

* **Amazon EKS** — Multi-tenant Kubernetes clusters (dev, stg, prod).
* **Amazon ECR** — Immutable container registry (Commercial + GovCloud).
* **Amazon EC2 / Managed Node Groups** — Compute capacity (GovCloud-approved).
* **AWS Fargate (optional)** — Serverless workloads (Commercial only).
* **Amazon VPC** — Networking foundation (multi-AZ private subnets).
* **Amazon Route 53** — DNS + failover routing.
* **AWS Load Balancer Controller (ALB/NLB)** — Ingress for tenant workloads.

**2. CI/CD & Supply Chain**

* **AWS CodeCommit** — Git repo for source control.
* **AWS CodeBuild** — Builds, scans, and signs workloads.
* **AWS CodePipeline** — Orchestration for build → scan → sign → deploy → mirror.
* **AWS KMS** — Asymmetric keys for Cosign signing.
* **Amazon S3** — Storage for SBOMs, logs, pipeline evidence.
* **Cross-partition ECR** — Artifact mirroring to GovCloud.
* *3rd Party Tools in Build Containers:* Syft (SBOM), Trivy (scanning), Cosign (signing).

**3. Security & Compliance**

* **AWS IAM** — Role-based access control.
* **IRSA (IAM Roles for Service Accounts)** — Pod-level IAM integration.
* **Amazon Cognito** — End-user authentication with OIDC/SAML federation.
* **AWS WAF** — Web Application Firewall at CloudFront/ALB.
* **AWS CloudTrail** — Account/pipeline activity logging.
* **Kyverno** — Runtime policy enforcement (verifyImages, pod security).
* **AWS PrivateLink (VPC Endpoints)** — FIPS-only API calls.

**4. Observability & Operations**

* **Amazon CloudWatch Logs** — Centralized logs.
* **CloudWatch Container Insights** — EKS metrics.
* **Amazon Managed Prometheus (AMP)** — Scraping metrics.
* **Amazon Managed Grafana (AMG)** — Dashboards + SLO visibility.
* **AWS X-Ray** — Distributed tracing.
* **OpenTelemetry (OTel)** — Tracing pipeline integration.

**5. Governance & Notifications**

* **Amazon SNS** — Notifications for approvals/events.
* **Amazon EventBridge** — Routing pipeline state events.
* **Slack/ServiceNow (optional)** — Approval/alert integrations.

**6. Data & Backup / DR**

* **AWS Backup** — Backup policies for EBS/EFS.
* **Velero** — Cluster/namespace backup.
* **Amazon S3 Cross-Region Replication** — DR redundancy.
* **Amazon DynamoDB / Aurora** — For app data if required.
* **Amazon ElastiCache** — Caching tier.
* **Amazon SQS** — Event decoupling for microservices.

**Appendix B — Environment Setup & Validation Checklist**

**0) Prereqs**

* AWS CLI authenticated to Commercial (and GovCloud if used).
* kubectl + cluster access (eks-dev, eks-stg, eks-prod).
* Helm 3 installed.
* Kyverno installed in all clusters.

**1) Config Setup**

* Populate deploy/bootstrap.conf (APP\_NAME, REPO\_NAME, VPCs, Subnets, etc.).
* Define Commercial + GovCloud account IDs and regions.
* Provide KMS signing alias (alias/myservice-cosign).
* Configure Cognito or OIDC values.

**2) KMS Signing Key**

* Create asymmetric ECC key in KMS.
* Alias must match COSIGN\_KMS\_ALIAS.
* Allow codebuild-build-role to sign.

**3) Preflight VPC Validation**

* Run deploy/preflight-vpc.sh to validate endpoints (ECR, S3, STS, KMS, Logs, CodeBuild).
* Fix missing endpoints before bootstrap.

**4) Bootstrap & Deploy**

* Run deploy/bootstrap-fill-and-deploy.sh.
* Confirm: params/\*.json, Helm values, tenant templates, stacks created.

**5) Cluster Prep**

* Confirm EKS clusters exist (dev/stg/prod).
* Add-ons: Metrics Server, ALB Controller, OIDC provider.

**6) Kyverno Setup**

* Fetch cosign public key with deploy/fetch-cosign-pubkey.sh.
* Apply verifyImages + pod security policies (ops/policies/\*).

**7) Pipeline Setup**

* Push repo to CodeCommit.
* Validate pipeline stages: build → scan → sign → attest → deploy → mirror.

**8) Ingress & DNS**

* Confirm ALB, ACM cert, WAF association, Route53 DNS entry.

**9) Notifications**

* Deploy templates/pipeline-notifications.yaml to enable SNS/EventBridge notifications.

**10) Tenant Onboarding**

* Use ops/tenant/values-tenant-template.yaml.
* Apply quotas and network policies.
* Deploy tenant workloads by image digest.

**11) Ops Hardening**

* Deploy External Secrets Operator.
* Enable observability (CW, AMP, Grafana, OTel).
* Validate backups (AWS Backup + Velero).

**Appendix C — Deployment Evidence (Scripts & Outputs)**

**Scripts**

* deploy/bootstrap-fill-and-deploy.sh — Automates namespace + tenant onboarding.
* deploy/preflight-vpc.sh — Validates FIPS endpoint compliance.
* deploy/fetch-cosign-pubkey.sh — Extracts public key for Kyverno verifyImages policy.
* deploy/apply-kyverno.sh — Applies admission policies.

**Outputs**

* **Pipeline Outputs:** Immutable image digests, SBOMs, Trivy reports, cosign attestations.
* **Cluster Evidence:** Namespace manifests, applied quotas, applied Kyverno policies.
* **Compliance Evidence:** S3 storage of all reports/artifacts.

**Appendix D — File-to-Chapter Mapping (Repo Proof)**

* **Chapter 3 Infrastructure** → templates/tools-pipeline.yaml, deploy/preflight-vpc.sh
* **Chapter 4 CI/CD** → ops/pipeline/buildspec-build.yml, ops/pipeline/buildspec-deploy.yml
* **Chapter 5 Security** → ops/policies/kyverno-verifyimages.yaml, templates/workload-deploy-iam.yaml
* **Chapter 6 Supply Chain** → ops/pipeline/buildspec-build.yml, templates/tools-pipeline.yaml
* **Chapter 7 Tenancy** → ops/tenant/values-tenant-template.yaml, ops/k8s/tenant-networkpolicy.yaml
* **Chapter 8 Partitioning** → ops/pipeline/buildspec-mirror-gov.yml, templates/gov-mirror-receiver.yaml
* **Chapter 9 Governance** → templates/pipeline-notifications.yaml
* **Chapter 10 Observability** → ops/k8s/otelsidecar.yaml, ops/k8s/backup-policies.yaml, dashboards/\*

**Appendix E — Optional AWS Services for Scale-Out (Optional Enhancements)**

* **AWS Control Tower** — Multi-account baseline for governance at scale.
* **AWS Service Catalog** — Self-service tenant onboarding templates.
* **Amazon GuardDuty + Security Hub** — Centralized threat detection.
* **AWS Config + Conformance Packs** — Continuous compliance monitoring.
* **Amazon OpenSearch Service** — Enterprise search/log analytics.