**SaaS zero-trust scoring & coaching platform**. Call sheet of what we should **use** and **how it fits**—I rolled these choices into the repo so it’s production-ready end-to-end.

**Final decisions (and why)**

**1) Compute & scale**

* **EKS + Karpenter** (replace Cluster Autoscaler): fastest scale-out, right-sizes nodes per workload, cheaper at steady state.
* **AWS Load Balancer Controller** with **ALB** Ingress; **CloudFront** sits in front for global latency, caching, and extra WAF layer.

**2) Identity, authN/Z, and tenancy**

* **Amazon Cognito** (Hosted UI, OIDC/OAuth2) for B2B tenants (supports SAML/OIDC federation later).
* **Multi-tenant model options** (all supported day-1; pick per customer SKU):
  + **Pooled**: **Aurora PostgreSQL Serverless v2** + **Row Level Security (RLS)** with tenant\_id claim from Cognito JWT enforced in DB policies.
  + **Silo**: separate **schema per tenant** in the same Aurora cluster.
  + **Dedicated**: separate **deployment stamp** (you already support this stamp concept).
* **External Secrets Operator** pulls Cognito, DB, SMTP, and third-party creds from **Secrets Manager/SSM** to K8s.

**3) Data & analytics**

* **Aurora PostgreSQL Serverless v2** for main OLTP.
* **S3** data lake for evidence uploads (policies: TLS-only, bucket keys, object lock optional).
* **Glue + Athena** optional later for analytics; **QuickSight** for internal dashboards.

**4) Zero-Trust scoring engine**

* **Scoring service** (EKS microservice) with:
  + Canonical control catalog mapped to **NIST 800-207** and **CISA Zero Trust Maturity Model 2.0** (pillars: Identity, Devices, Network, Application, Data, Visibility/Analytics, Automation).
  + Weighted scoring + maturity tiers (Traditional → Initial → Advanced → Optimal).
  + Evidence attachments (S3), automated checks (integrations below), and **Action Plans** (Step Functions).
* **EventBridge** for async scoring jobs and webhooks from scanners.

**5) Workflows & automation**

* **Step Functions** to orchestrate: assess → collect evidence → run checks → compute score → generate roadmap → notify.
* **Amazon SES** (or SNS + email) for notifications to tenant admins.

**6) Edge, DNS, TLS, and WAF**

* **CloudFront** → **ALB** (origin).
* **WAFv2**: Managed rule sets + custom rate limits at **CloudFront** and optional ALB association.
* **Route 53** + **ACM** (DNS-validated certs).

**7) Observability & SLOs**

* **OTel Collector** → **CloudWatch metrics/logs** + **X-Ray**.
* **AMP/AMG** (managed Prometheus/Grafana) already scaffolded—keep enabled.
* **SLOs**: availability, p95 latency, error rate for API and scoring jobs—**CloudWatch alarms** wired to SNS/PagerDuty.

**8) CI/CD (prod-safe)**

* **CodePipeline** with:
  + Source → Build (CodeBuild) → Staging deploy → **Manual Approval** → **Prod deploy** (Helm to EKS).
  + **Image signing** (cosign) verification gate (Kyverno policy already present).
  + Separate **gov mirror** step you already have (for GovCloud push).

**9) Security, backup, DR**

* **AWS Backup** (accounts, tag-based for RDS/EFS) + **Velero** for cluster state/app data (we added both).
* **ECR cross-region replication** for images.
* **Customer data isolation** via RLS / schema; **KMS CMKs** for Aurora, S3, Velero bucket.

**10) Cost guardrails**

* **AWS Budgets** (alerts added) + tagging (Project, Env, Tenant, CostCenter).
* Karpenter consolidation for idle burn reduction.

**What I will add to your repo (concrete modules)**

I’ll generate these **ready-to-deploy** pieces to match the decisions above and wire them to your existing structure:

**Identity & tenancy**

* templates/cognito-userpool.yaml – User Pool, App Client, hosted domain, groups (TenantAdmin/Analyst).
* ops/migrations/aurora/0001\_init\_tenancy.sql – tables (tenants, users, assessments, evidence, scores) + **RLS policies**.
* ops/migrations/aurora/0002\_seed\_controls.sql – control catalog mapped to NIST 800-207 + CISA ZTMM 2.0.
* templates/aurora-postgres-slsv2.yaml – Aurora PostgreSQL Serverless v2 (encrypted, DB subnet group, secrets).
* ops/app/env/example.env – TENANT\_CLAIM=custom:tenant\_id etc.

**Edge, DNS, TLS, WAF**

* templates/cloudfront-alb.yaml – CF distro with ALB origin, WAF at edge, OAC enabled.
* templates/waf-managed-rules.yaml – AWS Managed Rules + rate limit + IP allow/deny lists.

**Karpenter (scale)**

* templates/karpenter-controller-iam.yaml – Controller role & policy.
* templates/karpenter-node-iam.yaml – Node role + instance profile (ties to Karpenter NodeClass).
* deploy/deploy-karpenter.sh – installs Karpenter chart + NodeClass/NodePool (we prepped this).

**Scoring engine & workflows**

* services/scoring/ – containerized microservice scaffold (FastAPI or Node—defaults to FastAPI), Helm chart, buildspec.
* templates/eventbridge-bus-rules.yaml – scoring job events;
* templates/stepfunctions-scoring.yaml – state machine: gather → run checks → compute → persist → notify.
* ops/policies/app-iam.yaml – least-privilege for scoring service (S3 evidence read, Secrets, RDS access via SG).

**Observability & SLOs**

* templates/cloudwatch-slos.yaml – alarms for API p95, 5xx %; Karpenter queue backlog; DB CPU/conn; Velero failures.
* dashboards/ – JSON dashboards (CW + AMG) for “SaaS Zero-Trust Overview”.

**CI/CD (prod approval & signing)**

* templates/pipeline-workload-prod.yaml – CodePipeline with Staging → **ManualApproval** → Prod; cosign verify in buildspec.
* ops/pipeline/buildspec-verify-signature.yml – enforces sig verification before deploy.

**Docs & samples**

* docs/tenant-model.md – how JWT → RLS works + schema vs pooled.
* docs/scoring-model.md – pillars, weights, tiers, and example calculation.

**How it looks in production (flow)**

1. User authenticates via **Cognito** → app gets JWT with tenant\_id.
2. App calls **Scoring API** (EKS). DB calls enforce **RLS** using tenant\_id.
3. Evidence uploads stored in **S3** (KMS, TLS-only); async checks emit **EventBridge** events.
4. **Step Functions** runs the assessment → computes score → stores results → **SNS/SES** notify tenant admins.
5. **CloudFront** + **WAF** protect the edge; **ALB** terminates to EKS Ingress.
6. **OTel** → **CloudWatch/AMP/X-Ray**, with **SLO alarms** to on-call.
7. **Backups** via AWS Backup + **Velero**; images replicated cross-region via **ECR**.
8. Releases ship through **CodePipeline** with **manual prod approval** and **image-sig verification**.