Platforma Server: Runbook for AWS EKS Production Deployment

Advanced Deployment Path — Terraform

OVERVIEW

This document describes the advanced, infrastructure-as-code path for deploying the Platforma backend server on Amazon EKS using Terraform (or OpenTofu). It is intended for IT and DevOps teams that already manage AWS infrastructure declaratively and want a deployment they can read, review, diff, and adapt in source control.

Platforma offers two deployment paths that provision the same architecture — the same EKS cluster, controllers, storage, and Platforma release. The CloudFormation path (document PL-INSTALL-003) is a console-driven single stack and is the recommended starting point for most installations. The Terraform path documented here trades that guided experience for full control over the declarative source. Choose the one that fits how your team operates AWS.

- **Greenfield deployment.** Like the CloudFormation template, these modules create and own a brand-new EKS cluster end to end. They are not intended to be pointed at a cluster you already manage; for integrating the Helm chart into an existing cluster, contact MiLaboratories.
- **Bring-your-own VPC supported.** The cluster is always Terraform-managed, but it can be deployed into existing subnets.
- **Source published on GitHub.** This note intentionally references the published modules rather than embedding code. The Terraform modules, variable reference, and examples live at:

  https://github.com/milaboratory/platforma-helm/tree/main/infrastructure/aws/terraform

DEPLOYMENT ARCHITECTURE

The diagram below summarises the deployed system and which Terraform module owns each component. The deployment is split into two modules applied in order (see “Why two modules”).

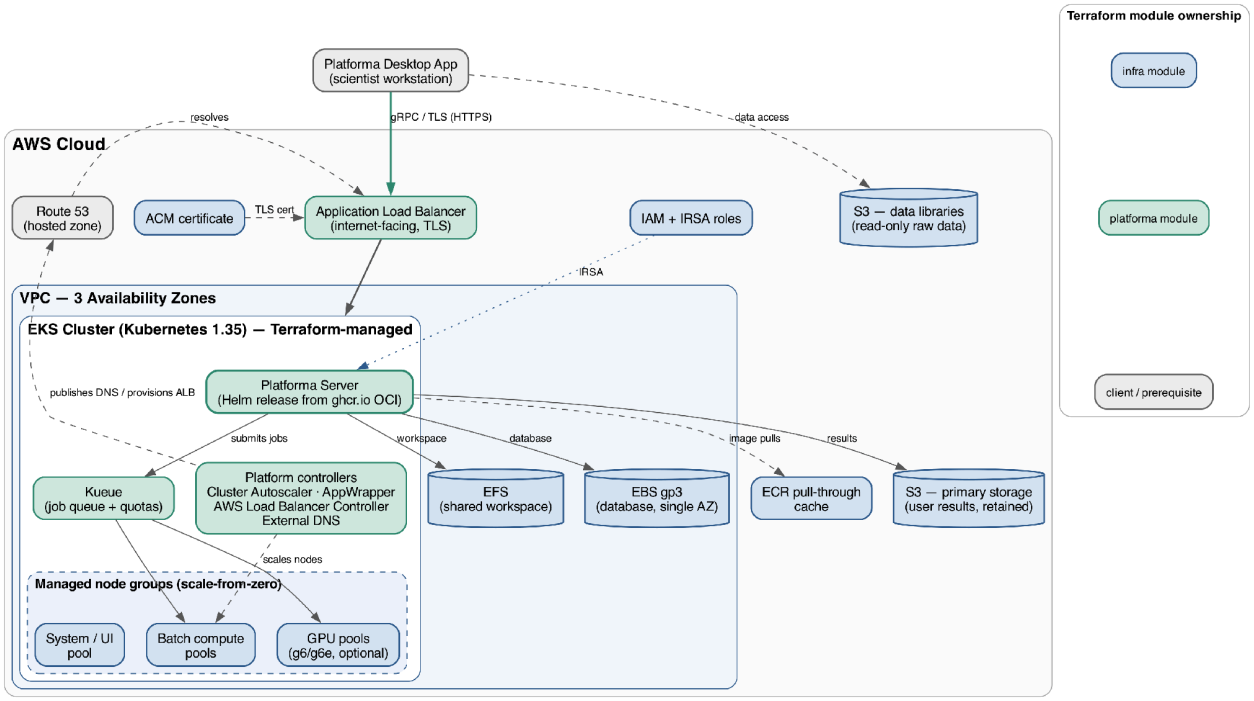

Figure 1: Platforma on AWS EKS — Terraform deployment architecture

The infra module provisions the AWS foundation: a VPC across three Availability Zones (new or bring-your-own), the EKS 1.35 cluster, managed node groups (a system/UI pool, scale-from-zero batch compute pools, and optional scale-from-zero GPU pools), IAM roles with IRSA, an EFS file system for the shared workspace, the primary S3 bucket for user result data, an ECR pull-through cache for container images, and — when ingress is enabled — a DNS-validated ACM certificate.

The platforma module installs the in-cluster controllers and the Platforma application. Kueue schedules batch jobs against per-size quotas, AppWrapper groups multi-pod jobs, and the Cluster Autoscaler scales node groups from zero on demand. With ingress enabled, the AWS Load Balancer Controller provisions an internet-facing ALB and External DNS publishes the endpoint record into Route 53. The Platforma server itself is a Helm release pulled from the MiLaboratories OCI registry. The Desktop App connects to the endpoint over gRPC secured by TLS, and reads sample data directly from the configured S3 data libraries.

Why two modules

The Kubernetes and Helm providers must be configured against a cluster that already exists. Splitting the deployment lets the platforma module resolve the cluster endpoint, IAM role ARNs, EFS id, and ACM certificate through plan-time data sources — so its providers receive concrete values instead of unknowns. Applying everything in one state would force the providers to depend on resources created in the same run. Each module is independently plan/apply-able and keeps its own state.

Module	Creates	Providers
`infra`	VPC (optional), EKS cluster + node groups, IAM/IRSA roles, EFS, S3, ECR pull-through cache, ACM certificate	aws

`platforma`	In-cluster controllers (Kueue, AppWrapper, Cluster Autoscaler, and — with ingress — ALB Controller + External DNS) and the Platforma Helm release	`aws, kubernetes, helm, kubectl`

PREREQUISITES

- **Terraform ≥ 1.5 or OpenTofu ≥ 1.6.** Commands below use terraform; substitute tofu for OpenTofu.
- **AWS CLI v2 on the machine running Terraform and on PATH.** The kubernetes/helm/kubectl providers authenticate to EKS via the aws eks get-token exec plugin.
- **AWS credentials** with permissions to create EKS, EC2/VPC, EFS, S3, IAM, ACM, and ECR resources.
- **A Route 53 hosted zone for your domain.** Required when ingress is enabled (the default): the ACM certificate is DNS-validated through this zone and External DNS manages the endpoint record.
- **A Platforma license key** (provided separately by MiLaboratories).
- **The Platforma Desktop App** installed on each end user's workstation. kubectl is optional but useful for retrieving credentials and inspecting the cluster after apply.

REPOSITORY AND STATE LAYOUT

The published repository contains two module directories applied in order:

- `infra/` — the AWS foundation (apply first).
- `platforma/` — controllers and the Platforma release (apply second).

Neither module declares a state backend — wire your own before the first apply (an S3 bucket with DynamoDB locking is typical), one state per module. State holds secrets (the license key, the htpasswd hash, any data-library credentials), so use an encrypted, access-controlled backend.

A set of shared identifiers must be identical in both modules — the IRSA trust policies, ACM certificate, ECR cache, and Kueue quotas created by infra are keyed to these exact values:

`region, cluster_name, platforma_namespace, helm_release_name, deployment_size, enable_gpu, ingress_enabled, domain_name, route53_zone_id, data_libraries`

STEP 1: PROVISION INFRASTRUCTURE (infra MODULE)

The infra module uses only the AWS provider. Set the deployment variables (in a terraform.tfvars file or your preferred mechanism), then initialise and apply. Provisioning the EKS cluster and node groups takes roughly 15–20 minutes.

Key variables:

- `region` — AWS region (GPU node groups require g6/g6e availability).
- `cluster_name` — EKS cluster name (lowercase, ≤25 chars; the S3 and ECR names derive from it).

- `domain_name, route53_zone_id` — the Platforma endpoint and its hosted zone (required with ingress).
- `deployment_size` — small / medium / large / xlarge (see Deployment Sizes below).
- `enable_gpu` — provision scale-from-zero GPU node groups (set false where GPUs are unavailable).
- `vpc_id` — leave empty to create a new VPC; set it for bring-your-own VPC (see Networking).

After apply, capture the generated bucket name for Step 2:

```
terraform output -raw s3_bucket_name
```

Networking: new VPC or bring-your-own

- **New VPC (default):** leave vpc_id empty. A VPC (default CIDR 10.0.0.0/16) with public and private subnets across 3 AZs is created.
- **Existing VPC:** set vpc_id and supply exactly 3 private subnet IDs (nodes + EFS mount targets) and 3 public subnet IDs (the internet-facing ALB), one per AZ. Three are required because EFS places one mount target per AZ and EKS spreads nodes across AZs.

STEP 2: DEPLOY CONTROLLERS AND PLATFORMA (platforma MODULE)

The platforma module uses the aws, kubernetes, helm, and kubectl providers, authenticating to the cluster created in Step 1. It discovers the cluster, IAM roles, EFS, and ACM certificate by name and tag; the only value passed in by hand is the S3 bucket name captured above (its auto-generated name carries a random suffix). Set the shared identifiers to match Step 1, then apply (roughly 5–10 minutes).

Key variables (beyond the shared identifiers):

- `s3_bucket_name` — the value from Step 1's output.
- `license_key` — your Platforma license (stored in a Kubernetes Secret; mark sensitive).
- `auth_method` — htpasswd (default) or ldap (see Authentication).
- `chart_version` — the Platforma chart version to pull (see Chart Source).
- `deploy_platforma` — set false to install only the controllers, useful for staging the cluster before the first application rollout; re-apply with true (and a license_key) to add Platforma.

DEPLOYMENT SIZES AND AWS QUOTAS

deployment_size sets the node-group maximum sizes and the Kueue ClusterQueue quotas that control parallelism. Before applying, open the Service Quotas console and verify your Running On-Demand Standard (A, C, D, H, I, M, R, T, Z) instances vCPU quota meets the selected size; request an increase if needed.

Deployment size	Approx. vCPU quota	Parallel jobs (large / small)
`small`	~400	~4 / ~16

`medium`	~700	~8 / ~32
`large`	~1400	~16 / ~64
`xlarge`	~2700	~32 / ~128

All sizes share the same maximum single-job capacity (62 vCPU / 484 GiB); deployment_size controls how many jobs run in parallel, not how large a single job can be.

AUTHENTICATION

Platforma supports two authentication methods, selected with auth_method.

- **htpasswd (default).** With htpasswd_content left empty, the deployment auto-generates a single platforma user with a random password stored in AWS SSM Parameter Store at /<cluster>/platforma/users-password. To manage credentials yourself, supply htpasswd_content with one user:hash line per user (bcrypt hashes, e.g. from htpasswd -nB user).

- **ldap.** Delegates authentication to an external directory. Configure ldap_server (use ldaps:// over port 636), ldap_bind_dn, ldap_search_user, ldap_search_password, and ldap_search_rules. LDAP ensures access follows your organization's joiner/leaver processes for sensitive genomic data.

DATA LIBRARIES

External S3 data libraries expose read-only raw data in the Desktop App. Configure them with data_libraries and pass the same list to both modules.

- **Same-account buckets (no credentials):** the infra module grants the Platforma IRSA roles read access; no keys are stored.

- **Cross-account buckets (with credentials):** the platforma module materialises the access/secret keys as Kubernetes Secrets.

- `enable_demo_data_library` (default true) adds MiLaboratories' read-only demo dataset.

CHART SOURCE AND IMAGES

- `chart_version` — the chart version pulled from the OCI registry.

- `chart_repository` — defaults to oci://ghcr.io/milaboratory/platforma-helm/platforma; override only to pull from a mirror or private ECR.

- `chart_local_path` — install from a local chart directory or .tgz (development / air-gapped).

- `platforma_image` — override the container image (repository:tag); empty uses the chart default.

OUTPUTS, CREDENTIALS, AND VERIFICATION

After the platforma module applies, read the outputs:

```
terraform output
```

When htpasswd auto-generated the user, fetch its password with the helper command exposed as an output (the command's output is the secret):

```
eval "$(terraform output -raw htpasswd_password_command)"
```

Output (module)	Purpose
`platforma_url (platforma)`	The HTTPS endpoint to enter in the Desktop App.
`htpasswd_password_command (platforma)`	Command to fetch the auto-generated platforma user password from SSM.
`s3_bucket_name (infra)`	Primary storage bucket; the value passed into Step 2.
`cluster_name / region (infra)`	Configure kubectl: aws eks update-kubeconfig --name <name> --region <region>.
`acm_certificate_arn (infra)`	The validated ACM certificate backing the ALB ingress.

To verify: open the Platforma Desktop App, add a connection to platforma_url, and authenticate with the platforma user (or your LDAP credentials). Initialise a test workspace and confirm the configured read-only data libraries are visible. ALB provisioning and Route 53 propagation may take a few minutes after apply; if the first connection fails, wait briefly and retry.

DAY-2 OPERATIONS

- **Upgrade Platforma:** bump chart_version and apply the platforma module. The Helm release uses --atomic, so a failed upgrade rolls back.
- **Resize:** change deployment_size in both modules and re-apply (infra adjusts node-group maximums, platforma adjusts the Kueue quotas).
- **Teardown:** destroy in reverse order — the platforma module first, then infra. The primary S3 bucket holds user result data and is retained by default (s3_force_destroy = false, mirroring CloudFormation's Retain policy); empty it manually or set s3_force_destroy = true before destroy to remove it.

RELATIONSHIP TO THE CLOUDFORMATION TEMPLATE

These modules deliberately mirror the CloudFormation template (cloudformation-eks-1-35.yaml): the same EKS version, controller versions, node-group shapes, GPU scale-from-zero configuration, Kueue quotas, and Platforma values. The differences are mechanical, not behavioral: there is no CodeBuild or Lambda (Helm and manifests are applied by Terraform's providers), htpasswd hashes use Terraform's built-in bcrypt(), and the AppWrapper manifest is integrity-checked against a pinned SHA-256 before apply.

Legal Notice

Trademarks

Platforma™, MiXCR™, and the MiLaboratories logo are trademarks or registered trademarks of MiLaboratories Inc. in the United States and other countries. All other product names, trademarks, and registered trademarks are property of their respective owners.

Disclaimer

This Technical Note is for informational purposes only. The information contained herein is subject to change without notice. MiLaboratories Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. MiLaboratories Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Research Use Only

The software and workflows described herein are intended for Research Use Only (RUO) and are not intended for diagnostic or therapeutic use in humans or animals unless explicitly stated otherwise.