Contents

â• â•

SECTION 6: SELF-HOSTED MODELS & MID-LEVEL SERVICES (v2.2.0)

â• â•

> **Dependencies:** Sections 0-5 **Creates:** 30+ SageMaker models, Thermal state management

Type Imports

```
import { ThermalState, ThermalConfig, ServiceState, MidLevelService } from '@radiant/shared';
```

RADIANT v2.2.0 - Prompt 6: Self-Hosted Models & Mid-Level Services

> **Prompt 6 of 9** | Target Size: ~75KB | Version: 3.7.0 | December 2024

OVERVIEW

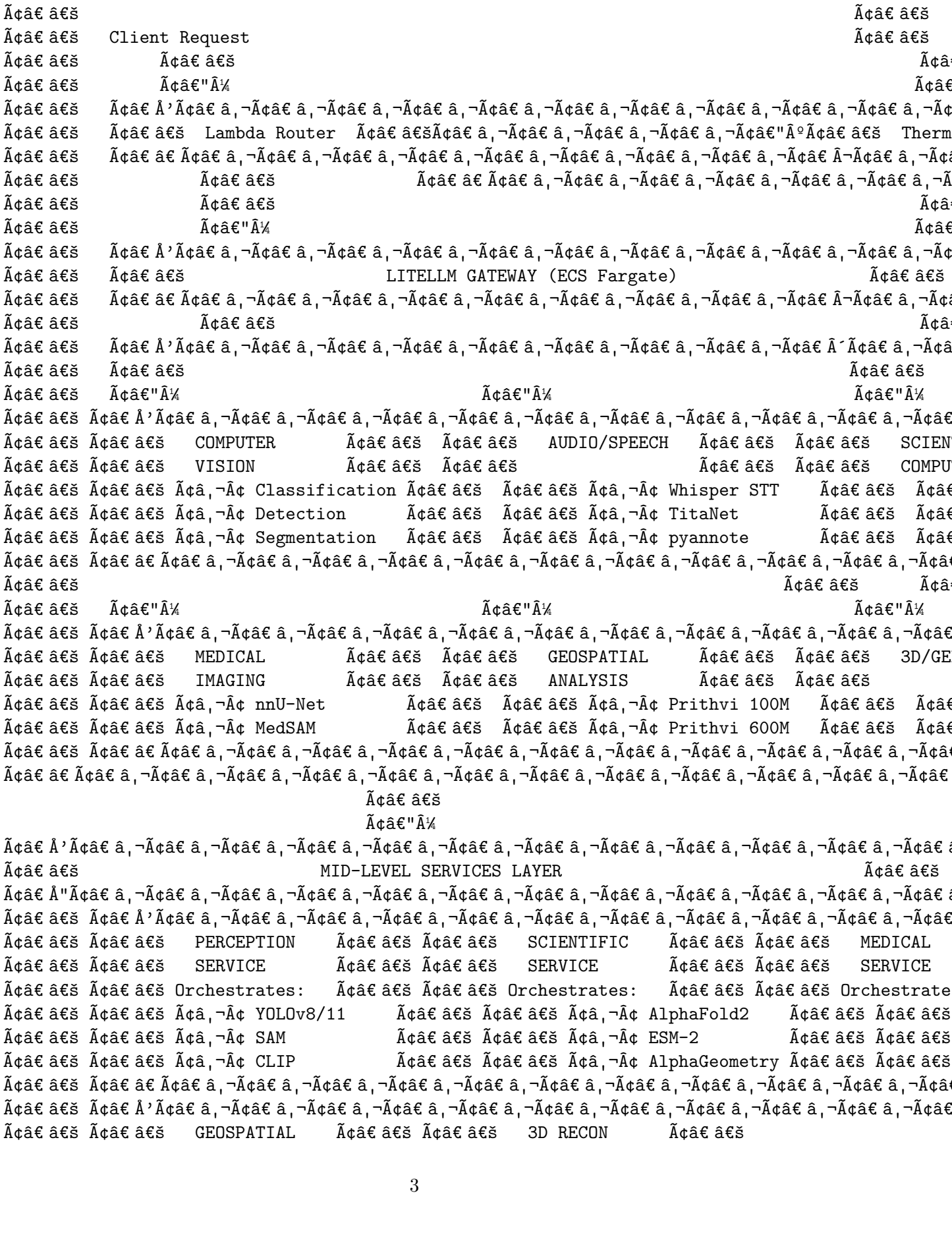
This prompt creates the self-hosted AI model configurations and mid-level orchestration services:

1. **Self-Hosted Models** - 30+ AI models deployed on SageMaker across 9 categories
2. **Thermal State Management** - Lambda functions for OFF/COLD/WARM/HOT/AUTOMATIC states
3. **Mid-Level Services** - 5 domain-specific orchestration services
4. **Service State Management** - RUNNING/DEGRADED/DISABLED/OFFLINE states
5. **LiteLLM Configuration** - Self-hosted model routing configuration
6. **Client Notifications** - Real-time model warm-up notifications via WebSocket

All configurations are tier-aware, with self-hosted models requiring Tier 3+ (GROWTH and above).

ARCHITECTURE

```
Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬
Ã¢â€ â€š                                SELF-HOSTED AI LAYER                                   Ã¢â€ â€š
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬
```


Client Request
LITELLM GATEWAY (ECS Fargate)
COMPUTER VISION
AUDIO/SPEECH
SCIEN
Classification
Detection
Segmentation
Whisper STT
TitaNet
pyannote
MEDICAL IMAGING
GEOSPATIAL ANALYSIS
3D/GE
nnU-Net
MedSAM
Prithvi 100M
Prithvi 600M
MID-LEVEL SERVICES LAYER
PERCEPTION SERVICE
SCIENTIFIC SERVICE
MEDICAL SERVICE
Orchestrates:
YOLOv8/11
SAM
CLIP
AlphaFold2
ESM-2
AlphaGeometry
GEOSPATIAL
3D RECON

```
Ã¢â€ â€š Ã¢â€ â€š   SERVICE       Ã¢â€ â€š Ã¢â€ â€š   SERVICE       Ã¢â€ â€š
Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ Prithvi       Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ Nerfstudio    Ã¢â€ â€š
Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ SAM           Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ 3D Gaussian   Ã¢â€ â€š
Ã¢â€ â€š Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€
Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ 
```

DIRECTORY STRUCTURE

```
packages/infrastructure/
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ lib/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ config/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ models/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts                # All model ex
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ vision.models.ts        # Computer vis
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ audio.models.ts         # Audio/speech
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ scientific.models.ts    # Scientific m
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ medical.models.ts       # Medical imag
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ geospatial.models.ts    # Geospatial m
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ generative.models.ts    # 3D/generativ
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ services/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts                # All service exports
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ perception.service.ts
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ scientific.service.ts
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ medical.service.ts
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ geospatial.service.ts
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ reconstruction.service.ts
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ lambda/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ thermal/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ manager.ts                # Thermal state manag
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ warmer.ts                 # Model warming Lambd
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ notifier.ts               # Client notificatio
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ services/
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ perception.ts             # Perception orchestration
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ scientific.ts             # Scientific orchestration
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ medical.ts                # Medical orchestration
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ geospatial.ts             # Geospatial orchestration
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ reconstruction.ts         # 3D reconstruction
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ litellm/
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ config/
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ self-hosted.yaml          # Self-hosted model config
Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ migrations/
    Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ 006_self_hosted_models.sql     # Schema additions
```

PART 1: SELF-HOSTED MODEL DEFINITIONS

AWS SageMaker Instance Pricing (with 75% markup)

Instance	GPUs	VRAM	Base $/hr	Billed $/hr	Best For
ml.g4dn.xlarge	1x T4	16 GB	$0.74	$1.30	Small models, embeddings
ml.g5.xlarge	1x A10G	24 GB	$1.41	$2.47	Medium models
ml.g5.2xlarge	1x A10G	24 GB	$1.52	$2.66	Medium+ models
ml.g5.4xlarge	1x A10G	24 GB	$2.03	$3.55	Large vision models
ml.g5.12xlarge	4x A10G	96 GB	$8.16	$14.28	Large LLMs
ml.g5.48xlarge	8x A10G	192 GB	$20.36	$35.63	Very large models
ml.p4d.24xlarge	8x A100	320 GB	$32.77	$57.35	Scientific computing

packages/infrastructure/lib/config/models/index.ts

```typescript
/**
 * Self-Hosted Model Registry
 * All models available for deployment on SageMaker
 * Pricing includes 75% markup over AWS infrastructure costs
 */

export * from './vision.models';
export * from './audio.models';
export * from './scientific.models';
export * from './medical.models';
export * from './geospatial.models';
export * from './generative.models';

// ============================================================================
// SHARED TYPES
// ============================================================================

export type ThermalState = 'OFF' | 'COLD' | 'WARM' | 'HOT' | 'AUTOMATIC';
export type ServiceState = 'RUNNING' | 'DEGRADED' | 'DISABLED' | 'OFFLINE';

export interface ThermalConfig {
  defaultState: ThermalState;
  scaleToZeroAfterMinutes: number;
  warmupTimeSeconds: number;
  minInstances: number;
  maxInstances: number;
}
```

```typescript
export interface SageMakerModelConfig {
  id: string;
  name: string;
  displayName: string;
  description: string;
  category: ModelCategory;
  specialty: ModelSpecialty;

  // SageMaker Configuration
  image: string;
  instanceType: string;
  environment: Record<string, string>;
  modelDataUrl?: string;

  // Model Capabilities
  parameters: number;
  accuracy?: string;
  benchmark?: string;
  capabilities: string[];
  inputFormats: string[];
  outputFormats: string[];

  // Thermal Management
  thermal: ThermalConfig;

  // Licensing
  license: string;
  licenseUrl?: string;
  commercialUseNotes?: string;

  // Pricing (75% markup over AWS)
  pricing: SelfHostedModelPricing;

  // Requirements
  minTier: number;
  requiresGPU: boolean;
  gpuMemoryGB: number;

  // Status
  status: 'active' | 'beta' | 'deprecated' | 'coming_soon';
}

export type ModelCategory =
  | 'vision_classification'
  | 'vision_detection'
  | 'vision_segmentation'
  | 'audio_stt'
```

```
  | 'audio_speaker'
  | 'scientific_protein'
  | 'scientific_math'
  | 'medical_imaging'
  | 'geospatial'
  | 'generative_3d'
  | 'llm_text';

export type ModelSpecialty =
  | 'image_classification'
  | 'object_detection'
  | 'instance_segmentation'
  | 'speech_to_text'
  | 'speaker_identification'
  | 'protein_folding'
  | 'protein_embeddings'
  | 'geometry_reasoning'
  | 'medical_segmentation'
  | 'satellite_analysis'
  | '3d_reconstruction'
  | 'text_generation';

export interface SelfHostedModelPricing {
  hourlyRate: number;        // Per-hour instance cost (with 75% markup)
  perImage?: number;
  perMinuteAudio?: number;
  perMinuteVideo?: number;
  per3DModel?: number;
  perRequest?: number;
  markup: number;            // 0.75 = 75%
}

export const INSTANCE_PRICING: Record<string, { base: number; billed: number }> = {
  'ml.g4dn.xlarge':   { base: 0.74,  billed: 1.30 },
  'ml.g5.xlarge':     { base: 1.41,  billed: 2.47 },
  'ml.g5.2xlarge':    { base: 1.52,  billed: 2.66 },
  'ml.g5.4xlarge':    { base: 2.03,  billed: 3.55 },
  'ml.g5.12xlarge':   { base: 8.16,  billed: 14.28 },
  'ml.g5.48xlarge':   { base: 20.36, billed: 35.63 },
  'ml.p4d.24xlarge':  { base: 32.77, billed: 57.35 },
};
```

packages/infrastructure/lib/config/models/vision.models.ts

```
/**
 * Computer Vision Models - Classification, Detection, Segmentation
 */
```

```typescript
import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

// ============================================================================
// IMAGE CLASSIFICATION MODELS (8 models)
// ============================================================================

export const CLASSIFICATION_MODELS: SageMakerModelConfig[] = [
  {
    id: 'efficientnet-b0',
    name: 'efficientnet-b0',
    displayName: 'EfficientNet-B0',
    description: 'Lightweight image classification model',
    category: 'vision_classification',
    specialty: 'image_classification',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g4dn.xlarge',
    environment: { HF_MODEL_ID: 'google/efficientnet-b0', HF_TASK: 'image-classification' },
    parameters: 5_300_000,
    accuracy: '77.1% ImageNet Top-1',
    capabilities: ['image_classification', 'feature_extraction'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 45, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 1.30, perImage: 0.001, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 2, status: 'active',
  },
  {
    id: 'efficientnetv2-l',
    name: 'efficientnetv2-l',
    displayName: 'EfficientNetV2-L',
    description: 'State-of-the-art classification with improved training efficiency',
    category: 'vision_classification',
    specialty: 'image_classification',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.2xlarge',
    environment: { HF_MODEL_ID: 'google/efficientnetv2-l', HF_TASK: 'image-classification' },
    parameters: 118_000_000,
    accuracy: '85.7% ImageNet Top-1',
    capabilities: ['image_classification', 'feature_extraction'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.66, perImage: 0.003, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
  },
  {
```

```
    id: 'swin-large',
    name: 'swin-large',
    displayName: 'Swin Transformer Large',
    description: 'Vision Transformer with shifted window attention',
    category: 'vision_classification',
    specialty: 'image_classification',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.2xlarge',
    environment: { HF_MODEL_ID: 'microsoft/swin-large-patch4-window7-224', HF_TASK: 'image-clas
    parameters: 197_000_000,
    accuracy: '87.3% ImageNet Top-1',
    capabilities: ['image_classification', 'feature_extraction'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, minIns
    license: 'MIT',
    pricing: { hourlyRate: 2.66, perImage: 0.004, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
  {
    id: 'clip-vit-l14',
    name: 'clip-vit-l14',
    displayName: 'CLIP ViT-L/14',
    description: 'Multimodal vision-language model for zero-shot classification',
    category: 'vision_classification',
    specialty: 'image_classification',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.2xlarge',
    environment: { HF_MODEL_ID: 'openai/clip-vit-large-patch14', HF_TASK: 'zero-shot-image-clas
    parameters: 428_000_000,
    accuracy: '76.2% ImageNet Zero-Shot',
    capabilities: ['zero_shot_classification', 'image_text_matching'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, minIns
    license: 'MIT',
    pricing: { hourlyRate: 2.66, perImage: 0.005, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
];

// ============================================================================
// OBJECT DETECTION MODELS (7 models)
// ============================================================================

export const DETECTION_MODELS: SageMakerModelConfig[] = [
  {
    id: 'yolov8n',
```

```
  name: 'yolov8n',
  displayName: 'YOLOv8 Nano',
  description: 'Ultra-lightweight real-time object detection',
  category: 'vision_detection',
  specialty: 'object_detection',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g4dn.xlarge',
  environment: { MODEL_NAME: 'yolov8n' },
  parameters: 3_200_000,
  benchmark: '37.3 COCO mAP',
  capabilities: ['object_detection', 'real_time'],
  inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 30, minIns
  license: 'AGPL-3.0',
  commercialUseNotes: 'Requires Ultralytics Enterprise License for commercial use',
  pricing: { hourlyRate: 1.30, perImage: 0.002, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 2, status: 'active',
},
{
  id: 'yolov8m',
  name: 'yolov8m',
  displayName: 'YOLOv8 Medium',
  description: 'Medium model for balanced performance',
  category: 'vision_detection',
  specialty: 'object_detection',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.xlarge',
  environment: { MODEL_NAME: 'yolov8m' },
  parameters: 25_900_000,
  benchmark: '50.2 COCO mAP',
  capabilities: ['object_detection', 'real_time'],
  inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 45, minIns
  license: 'AGPL-3.0',
  commercialUseNotes: 'Requires Ultralytics Enterprise License for commercial use',
  pricing: { hourlyRate: 2.47, perImage: 0.004, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 8, status: 'active',
},
{
  id: 'yolov8x',
  name: 'yolov8x',
  displayName: 'YOLOv8 Extra Large',
  description: 'Largest YOLOv8 for maximum accuracy',
  category: 'vision_detection',
  specialty: 'object_detection',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
```

```
  instanceType: 'ml.g5.xlarge',
  environment: { MODEL_NAME: 'yolov8x' },
  parameters: 68_200_000,
  benchmark: '53.9 COCO mAP',
  capabilities: ['object_detection', 'high_accuracy'],
  inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, minIns
  license: 'AGPL-3.0',
  commercialUseNotes: 'Requires Ultralytics Enterprise License for commercial use',
  pricing: { hourlyRate: 2.47, perImage: 0.006, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 12, status: 'active',
},
{
  id: 'yolov11x',
  name: 'yolov11x',
  displayName: 'YOLOv11 Extra Large',
  description: 'Latest YOLO with improved architecture',
  category: 'vision_detection',
  specialty: 'object_detection',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.xlarge',
  environment: { MODEL_NAME: 'yolo11x' },
  parameters: 40_000_000,
  benchmark: '54.7 COCO mAP',
  capabilities: ['object_detection', 'real_time', 'high_accuracy'],
  inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, minIns
  license: 'AGPL-3.0',
  commercialUseNotes: 'Requires Ultralytics Enterprise License for commercial use',
  pricing: { hourlyRate: 2.47, perImage: 0.006, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
},
{
  id: 'rt-detr-x',
  name: 'rt-detr-x',
  displayName: 'RT-DETR-X',
  description: 'Real-time Detection Transformer (no NMS)',
  category: 'vision_detection',
  specialty: 'object_detection',
  image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.xlarge',
  environment: { HF_MODEL_ID: 'PekingU/rtdetr_r101vd', HF_TASK: 'object-detection' },
  parameters: 40_000_000,
  benchmark: '54.8 COCO mAP',
  capabilities: ['object_detection', 'real_time', 'end_to_end'],
  inputFormats: ['image/jpeg', 'image/png'],
```

```
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.47, perImage: 0.005, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
  },
  {
    id: 'grounding-dino',
    name: 'grounding-dino',
    displayName: 'Grounding DINO',
    description: 'Open-vocabulary object detection with text prompts',
    category: 'vision_detection',
    specialty: 'object_detection',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.2xlarge',
    environment: { HF_MODEL_ID: 'IDEA-Research/grounding-dino-base', HF_TASK: 'zero-shot-object
    parameters: 172_000_000,
    benchmark: '52.5 COCO Zero-Shot',
    capabilities: ['zero_shot_detection', 'text_prompted', 'open_vocabulary'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.66, perImage: 0.008, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
];

// ============================================================================
// SEGMENTATION MODELS (4 models)
// ============================================================================

export const SEGMENTATION_MODELS: SageMakerModelConfig[] = [
  {
    id: 'sam-vit-h',
    name: 'sam-vit-h',
    displayName: 'SAM ViT-H',
    description: 'Segment Anything Model - largest variant',
    category: 'vision_segmentation',
    specialty: 'instance_segmentation',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.4xlarge',
    environment: { HF_MODEL_ID: 'facebook/sam-vit-huge', HF_TASK: 'mask-generation' },
    parameters: 636_000_000,
    capabilities: ['instance_segmentation', 'interactive', 'zero_shot'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json', 'image/png'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 120, minI
```

```
  license: 'Apache-2.0',
  pricing: { hourlyRate: 3.55, perImage: 0.015, markup: 0.75 },
  minTier: 4, requiresGPU: true, gpuMemoryGB: 20, status: 'active',
},
{
  id: 'sam2',
  name: 'sam2',
  displayName: 'SAM 2',
  description: 'Segment Anything Model 2 with video support',
  category: 'vision_segmentation',
  specialty: 'instance_segmentation',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.4xlarge',
  environment: { MODEL_NAME: 'sam2_hiera_large' },
  parameters: 224_000_000,
  capabilities: ['instance_segmentation', 'video_segmentation', 'interactive', 'zero_shot'],
  inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
  outputFormats: ['application/json', 'image/png', 'video/mp4'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 120, minIn
  license: 'Apache-2.0',
  pricing: { hourlyRate: 3.55, perImage: 0.012, perMinuteVideo: 0.50, markup: 0.75 },
  minTier: 4, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
},
{
  id: 'mobilesam',
  name: 'mobilesam',
  displayName: 'MobileSAM',
  description: 'Lightweight SAM for fast inference',
  category: 'vision_segmentation',
  specialty: 'instance_segmentation',
  image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g4dn.xlarge',
  environment: { HF_MODEL_ID: 'dhkim2810/mobilesam', HF_TASK: 'mask-generation' },
  parameters: 10_000_000,
  capabilities: ['instance_segmentation', 'interactive', 'fast'],
  inputFormats: ['image/jpeg', 'image/png'],
  outputFormats: ['application/json', 'image/png'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 30, minIns
  license: 'Apache-2.0',
  pricing: { hourlyRate: 1.30, perImage: 0.004, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 4, status: 'active',
},
{
  id: 'mask-rcnn',
  name: 'mask-rcnn',
  displayName: 'Mask R-CNN',
  description: 'Classic instance segmentation model',
  category: 'vision_segmentation',
```

```
    specialty: 'instance_segmentation',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { MODEL_NAME: 'mask_rcnn_R_101_FPN_3x', DETECTRON2_CONFIG: 'COCO-InstanceSegme
    parameters: 63_000_000,
    benchmark: '42.9 COCO AP',
    capabilities: ['instance_segmentation', 'object_detection'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json', 'image/png'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.47, perImage: 0.006, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 8, status: 'active',
  },
];

// Export all vision models
export const ALL_VISION_MODELS = [...CLASSIFICATION_MODELS, ...DETECTION_MODELS, ...SEGMENTATIO
```

packages/infrastructure/lib/config/models/audio.models.ts

```
/**
 * Audio & Speech Models - STT, Speaker Recognition
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

// ============================================================================
// SPEECH-TO-TEXT MODELS (3 models)
// ============================================================================

export const STT_MODELS: SageMakerModelConfig[] = [
  {
    id: 'parakeet-tdt-1b',
    name: 'parakeet-tdt-1b',
    displayName: 'NVIDIA Parakeet TDT 1.1B',
    description: 'State-of-the-art ASR with 4.4% WER on LibriSpeech',
    category: 'audio_stt',
    specialty: 'speech_to_text',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { MODEL_NAME: 'nvidia/parakeet-tdt-1.1b' },
    parameters: 1_100_000_000,
    accuracy: '4.4% WER LibriSpeech test-clean',
    capabilities: ['speech_to_text', 'english', 'real_time'],
    inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, minIns
```

```
    license: 'CC-BY-4.0',
    pricing: { hourlyRate: 2.47, perMinuteAudio: 0.03, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
  },
  {
    id: 'whisper-large-v3',
    name: 'whisper-large-v3',
    displayName: 'OpenAI Whisper Large V3',
    description: 'Multilingual speech recognition and translation',
    category: 'audio_stt',
    specialty: 'speech_to_text',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { HF_MODEL_ID: 'openai/whisper-large-v3', HF_TASK: 'automatic-speech-recogniti
    parameters: 1_550_000_000,
    accuracy: '5.0% WER LibriSpeech test-clean',
    capabilities: ['speech_to_text', 'multilingual', 'translation', 'timestamps'],
    inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac', 'audio/m4a'],
    outputFormats: ['application/json', 'text/plain', 'text/vtt', 'text/srt'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, minIns
    license: 'MIT',
    pricing: { hourlyRate: 2.47, perMinuteAudio: 0.04, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
  },
  {
    id: 'whisper-turbo',
    name: 'whisper-turbo',
    displayName: 'OpenAI Whisper Turbo',
    description: 'Fast multilingual ASR with 8x speed improvement',
    category: 'audio_stt',
    specialty: 'speech_to_text',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g4dn.xlarge',
    environment: { HF_MODEL_ID: 'openai/whisper-large-v3-turbo', HF_TASK: 'automatic-speech-rec
    parameters: 809_000_000,
    accuracy: '5.5% WER LibriSpeech test-clean',
    capabilities: ['speech_to_text', 'multilingual', 'fast', 'real_time'],
    inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac', 'audio/m4a'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, minIns
    license: 'MIT',
    pricing: { hourlyRate: 1.30, perMinuteAudio: 0.02, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 6, status: 'active',
  },
];

// ============================================================================
// SPEAKER RECOGNITION MODELS (3 models)
```

```typescript
// ============================================================================

export const SPEAKER_MODELS: SageMakerModelConfig[] = [
  {
    id: 'titanet-large',
    name: 'titanet-large',
    displayName: 'NVIDIA TitaNet-Large',
    description: 'Speaker verification and embedding extraction',
    category: 'audio_speaker',
    specialty: 'speaker_identification',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g4dn.xlarge',
    environment: { MODEL_NAME: 'nvidia/speakerverification_en_titanet_large' },
    parameters: 23_000_000,
    accuracy: '0.68% EER VoxCeleb1',
    capabilities: ['speaker_verification', 'speaker_embedding', 'speaker_identification'],
    inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 45, minIns
    license: 'CC-BY-4.0',
    pricing: { hourlyRate: 1.30, perMinuteAudio: 0.02, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 4, status: 'active',
  },
  {
    id: 'ecapa-tdnn',
    name: 'ecapa-tdnn',
    displayName: 'ECAPA-TDNN',
    description: 'Emphasized channel attention for speaker verification',
    category: 'audio_speaker',
    specialty: 'speaker_identification',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g4dn.xlarge',
    environment: { MODEL_NAME: 'speechbrain/spkrec-ecapa-voxceleb' },
    parameters: 20_800_000,
    accuracy: '0.80% EER VoxCeleb1',
    capabilities: ['speaker_verification', 'speaker_embedding'],
    inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 45, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 1.30, perMinuteAudio: 0.02, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 4, status: 'active',
  },
  {
    id: 'pyannote-speaker-diarization',
    name: 'pyannote-speaker-diarization',
    displayName: 'pyannote Speaker Diarization 3.1',
    description: 'Who spoke when - speaker diarization pipeline',
```

```
    category: 'audio_speaker',
    specialty: 'speaker_identification',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { MODEL_NAME: 'pyannote/speaker-diarization-3.1' },
    parameters: 0,
    accuracy: '18.4% DER AMI Mix-Headset',
    capabilities: ['speaker_diarization', 'speaker_counting', 'overlap_detection'],
    inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac'],
    outputFormats: ['application/json', 'text/rttm'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, minIns
    license: 'MIT',
    commercialUseNotes: 'Requires pyannote/speaker-diarization access grant',
    pricing: { hourlyRate: 2.47, perMinuteAudio: 0.05, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 8, status: 'active',
  },
];

export const ALL_AUDIO_MODELS = [...STT_MODELS, ...SPEAKER_MODELS];
```

packages/infrastructure/lib/config/models/scientific.models.ts

```
/**
 * Scientific Computing Models - Protein folding, embeddings, math reasoning
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

export const SCIENTIFIC_MODELS: SageMakerModelConfig[] = [
  {
    id: 'alphafold2',
    name: 'alphafold2',
    displayName: 'AlphaFold 2',
    description: 'Protein structure prediction with near-experimental accuracy',
    category: 'scientific_protein',
    specialty: 'protein_folding',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.12xlarge',
    environment: { MODEL_NAME: 'alphafold2_ptm', JAX_PLATFORM_NAME: 'gpu' },
    parameters: 93_000_000,
    accuracy: 'GDT > 90 on CASP14',
    capabilities: ['protein_folding', 'structure_prediction', 'confidence_estimation'],
    inputFormats: ['text/fasta', 'text/plain'],
    outputFormats: ['application/pdb', 'application/mmcif', 'application/json'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 180, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 14.28, perRequest: 2.00, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 96, status: 'active',
```

```
},
{
  id: 'esm2-3b',
  name: 'esm2-3b',
  displayName: 'ESM-2 3B',
  description: 'Protein language model for embeddings and analysis',
  category: 'scientific_protein',
  specialty: 'protein_embeddings',
  image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.4xlarge',
  environment: { HF_MODEL_ID: 'facebook/esm2_t36_3B_UR50D' },
  parameters: 3_000_000_000,
  capabilities: ['protein_embedding', 'sequence_analysis', 'structure_prediction'],
  inputFormats: ['text/fasta', 'text/plain'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, minIns
  license: 'MIT',
  pricing: { hourlyRate: 3.55, perRequest: 0.50, markup: 0.75 },
  minTier: 4, requiresGPU: true, gpuMemoryGB: 20, status: 'active',
},
{
  id: 'alphageometry',
  name: 'alphageometry',
  displayName: 'AlphaGeometry',
  description: 'Olympiad-level geometry problem solver',
  category: 'scientific_math',
  specialty: 'geometry_reasoning',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.2xlarge',
  environment: { MODEL_NAME: 'alphageometry' },
  parameters: 150_000_000,
  accuracy: '25/30 IMO geometry problems',
  capabilities: ['geometry_solving', 'proof_generation', 'theorem_proving'],
  inputFormats: ['application/json', 'text/plain'],
  outputFormats: ['application/json', 'text/plain'],
  thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 90, minInst
  license: 'Apache-2.0',
  pricing: { hourlyRate: 2.66, perRequest: 0.30, markup: 0.75 },
  minTier: 4, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
},
{
  id: 'protenix',
  name: 'protenix',
  displayName: 'Protenix',
  description: 'Open-source AlphaFold3-like structure prediction',
  category: 'scientific_protein',
  specialty: 'protein_folding',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
```

```
    instanceType: 'ml.g5.12xlarge',
    environment: { MODEL_NAME: 'protenix' },
    parameters: 0,
    capabilities: ['protein_folding', 'dna_rna_binding', 'ligand_binding', 'multi_chain'],
    inputFormats: ['text/fasta', 'application/json'],
    outputFormats: ['application/pdb', 'application/mmcif', 'application/json'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 180, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 14.28, perRequest: 2.50, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 96, status: 'beta',
  },
];
```

packages/infrastructure/lib/config/models/medical.models.ts

```
/**
 * Medical Imaging Models - HIPAA-compliant medical image analysis
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

export const MEDICAL_MODELS: SageMakerModelConfig[] = [
  {
    id: 'nnunet',
    name: 'nnunet',
    displayName: 'nnU-Net',
    description: 'Self-configuring medical image segmentation',
    category: 'medical_imaging',
    specialty: 'medical_segmentation',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.2xlarge',
    environment: { MODEL_NAME: 'nnunet', NNUNET_DATASET: 'generic' },
    parameters: 31_000_000,
    accuracy: 'State-of-the-art on 23/23 Medical Segmentation Decathlon tasks',
    capabilities: ['medical_segmentation', 'tumor_detection', 'organ_segmentation', '3d_imaging
    inputFormats: ['application/dicom', 'application/nifti', 'image/png'],
    outputFormats: ['application/nifti', 'application/json', 'image/png'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, minIns
    license: 'Apache-2.0',
    commercialUseNotes: 'HIPAA compliant when deployed in compliant AWS environment',
    pricing: { hourlyRate: 2.66, perImage: 0.10, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
  {
    id: 'medsam',
    name: 'medsam',
    displayName: 'MedSAM',
    description: 'Segment Anything Model fine-tuned for medical images',
```

```
    category: 'medical_imaging',
    specialty: 'medical_segmentation',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.2xlarge',
    environment: { HF_MODEL_ID: 'wanglab/medsam-vit-base', HF_TASK: 'mask-generation' },
    parameters: 93_000_000,
    capabilities: ['medical_segmentation', 'interactive', 'multi_modality'],
    inputFormats: ['application/dicom', 'application/nifti', 'image/png', 'image/jpeg'],
    outputFormats: ['application/json', 'image/png'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 90, minInst
    license: 'Apache-2.0',
    commercialUseNotes: 'HIPAA compliant when deployed in compliant AWS environment',
    pricing: { hourlyRate: 2.66, perImage: 0.08, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 12, status: 'active',
  },
];
```

packages/infrastructure/lib/config/models/geospatial.models.ts

```
/**
 * Geospatial Analysis Models - Satellite imagery and earth observation
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

export const GEOSPATIAL_MODELS: SageMakerModelConfig[] = [
  {
    id: 'prithvi-100m',
    name: 'prithvi-100m',
    displayName: 'Prithvi 100M',
    description: 'NASA/IBM geospatial foundation model (100M parameters)',
    category: 'geospatial',
    specialty: 'satellite_analysis',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { HF_MODEL_ID: 'ibm-nasa-geospatial/Prithvi-100M' },
    parameters: 100_000_000,
    capabilities: ['satellite_analysis', 'land_use', 'flood_detection', 'crop_mapping'],
    inputFormats: ['image/tiff', 'image/geotiff', 'image/png'],
    outputFormats: ['application/json', 'image/geotiff'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 90, minInst
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.47, perImage: 0.05, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
  },
  {
    id: 'prithvi-600m',
    name: 'prithvi-600m',
```

```
    displayName: 'Prithvi 600M',
    description: 'NASA/IBM geospatial foundation model (600M parameters)',
    category: 'geospatial',
    specialty: 'satellite_analysis',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.4xlarge',
    environment: { HF_MODEL_ID: 'ibm-nasa-geospatial/Prithvi-EO-2.0-600M' },
    parameters: 600_000_000,
    capabilities: ['satellite_analysis', 'land_use', 'flood_detection', 'crop_mapping', 'change
    inputFormats: ['image/tiff', 'image/geotiff', 'image/png'],
    outputFormats: ['application/json', 'image/geotiff'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 3.55, perImage: 0.10, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 20, status: 'active',
  },
];
```

packages/infrastructure/lib/config/models/generative.models.ts

```
/**
 * 3D & Generative Models - 3D reconstruction, self-hosted LLMs
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

export const GENERATIVE_3D_MODELS: SageMakerModelConfig[] = [
  {
    id: 'nerfstudio',
    name: 'nerfstudio',
    displayName: 'Nerfstudio',
    description: 'Neural Radiance Fields for 3D scene reconstruction',
    category: 'generative_3d',
    specialty: '3d_reconstruction',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.4xlarge',
    environment: { MODEL_NAME: 'nerfstudio', NERFSTUDIO_METHOD: 'nerfacto' },
    parameters: 0,
    capabilities: ['3d_reconstruction', 'novel_view_synthesis', 'scene_capture'],
    inputFormats: ['video/mp4', 'image/jpeg', 'image/png'],
    outputFormats: ['model/gltf+json', 'model/obj', 'video/mp4'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 3.55, per3DModel: 5.00, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 24, status: 'active',
  },
  {
    id: '3d-gaussian-splatting',
```

```
    name: '3d-gaussian-splatting',
    displayName: '3D Gaussian Splatting',
    description: 'Real-time radiance field rendering with 3D Gaussians',
    category: 'generative_3d',
    specialty: '3d_reconstruction',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.4xlarge',
    environment: { MODEL_NAME: '3dgs' },
    parameters: 0,
    capabilities: ['3d_reconstruction', 'real_time_rendering', 'novel_view_synthesis'],
    inputFormats: ['video/mp4', 'image/jpeg', 'image/png'],
    outputFormats: ['model/ply', 'model/gltf+json', 'video/mp4'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, minIns
    license: 'Custom',
    commercialUseNotes: 'Check license for commercial use terms',
    pricing: { hourlyRate: 3.55, per3DModel: 4.00, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 24, status: 'active',
  },
];

export const TEXT_GENERATION_MODELS: SageMakerModelConfig[] = [
  {
    id: 'mistral-7b-instruct',
    name: 'mistral-7b-instruct',
    displayName: 'Mistral 7B Instruct',
    description: 'Efficient instruction-following model',
    category: 'llm_text',
    specialty: 'text_generation',
    image: 'huggingface-pytorch-tgi-inference:2.1-tgi1.4-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { HF_MODEL_ID: 'mistralai/Mistral-7B-Instruct-v0.3', MAX_INPUT_LENGTH: '4096'
    parameters: 7_000_000_000,
    capabilities: ['text_generation', 'instruction_following', 'function_calling'],
    inputFormats: ['application/json', 'text/plain'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.47, perRequest: 0.005, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
  {
    id: 'llama-3-70b-instruct',
    name: 'llama-3-70b-instruct',
    displayName: 'Llama 3 70B Instruct',
    description: 'Large open-weight model for complex tasks',
    category: 'llm_text',
    specialty: 'text_generation',
    image: 'huggingface-pytorch-tgi-inference:2.1-tgi1.4-gpu-py310-cu121-ubuntu22.04',
```

```
    instanceType: 'ml.g5.48xlarge',
    environment: { HF_MODEL_ID: 'meta-llama/Meta-Llama-3-70B-Instruct', MAX_INPUT_LENGTH: '819
    parameters: 70_000_000_000,
    capabilities: ['text_generation', 'instruction_following', 'reasoning', 'code_generation']
    inputFormats: ['application/json', 'text/plain'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 300, minIns
    license: 'Llama 3 Community License',
    commercialUseNotes: 'Free for commercial use, attribution required',
    pricing: { hourlyRate: 35.63, perRequest: 0.05, markup: 0.75 },
    minTier: 5, requiresGPU: true, gpuMemoryGB: 160, status: 'active',
  },
  {
    id: 'qwen-72b-instruct',
    name: 'qwen-72b-instruct',
    displayName: 'Qwen 2.5 72B Instruct',
    description: 'State-of-the-art multilingual model',
    category: 'llm_text',
    specialty: 'text_generation',
    image: 'huggingface-pytorch-tgi-inference:2.1-tgi1.4-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.48xlarge',
    environment: { HF_MODEL_ID: 'Qwen/Qwen2.5-72B-Instruct', MAX_INPUT_LENGTH: '32768', MAX_TO
    parameters: 72_000_000_000,
    capabilities: ['text_generation', 'instruction_following', 'multilingual', 'long_context',
    inputFormats: ['application/json', 'text/plain'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 300, minIns
    license: 'Apache-2.0',
    pricing: { hourlyRate: 35.63, perRequest: 0.05, markup: 0.75 },
    minTier: 5, requiresGPU: true, gpuMemoryGB: 160, status: 'active',
  },
];

export const ALL_GENERATIVE_MODELS = [...GENERATIVE_3D_MODELS, ...TEXT_GENERATION_MODELS];
```

PART 2: MID-LEVEL SERVICES

packages/infrastructure/lib/config/services/index.ts

```
/**
 * Mid-Level Service Definitions
 * Services that orchestrate multiple AI models for domain-specific tasks
 */

export type ServiceState = 'RUNNING' | 'DEGRADED' | 'DISABLED' | 'OFFLINE';

export interface MidLevelServiceConfig {
```

```
  id: string;
  name: string;
  displayName: string;
  description: string;
  requiredModels: string[];
  optionalModels: string[];
  defaultState: ServiceState;
  gracefulDegradation: boolean;
  pricing: ServicePricing;
  minTier: number;
  endpoints: ServiceEndpoint[];
}

export interface ServicePricing {
  perRequest?: number;
  perMinuteAudio?: number;
  perMinuteVideo?: number;
  perImage?: number;
  per3DModel?: number;
  markup: number;
}

export interface ServiceEndpoint {
  path: string;
  method: 'GET' | 'POST' | 'PUT' | 'DELETE';
  description: string;
  requiredModels: string[];
  inputFormats: string[];
  outputFormats: string[];
}

export const SERVICE_STATE_COLORS: Record<ServiceState, string> = {
  RUNNING: '#22c55e',    // green
  DEGRADED: '#f59e0b',   // yellow
  DISABLED: '#6b7280',   // gray
  OFFLINE: '#ef4444',    // red
};
```

packages/infrastructure/lib/config/services/perception.service.ts

```
/**
 * Perception Service - Unified computer vision pipeline
 */

import { MidLevelServiceConfig } from './index';

export const PERCEPTION_SERVICE: MidLevelServiceConfig = {
  id: 'perception',
```

```
  name: 'perception',
  displayName: 'Perception Service',
  description: 'Unified computer vision pipeline for detection, segmentation, and classificati
  requiredModels: ['yolov8m', 'mobilesam'],
  optionalModels: ['yolov8x', 'yolov11x', 'sam-vit-h', 'sam2', 'clip-vit-l14', 'grounding-dino
  defaultState: 'DISABLED',
  gracefulDegradation: true,
  pricing: { perImage: 0.02, perMinuteVideo: 0.50, markup: 0.40 },
  minTier: 3,
  endpoints: [
    { path: '/perception/detect', method: 'POST', description: 'Detect objects in images', requ
    { path: '/perception/segment', method: 'POST', description: 'Segment objects or regions', r
    { path: '/perception/classify', method: 'POST', description: 'Classify images', requiredMod
    { path: '/perception/analyze', method: 'POST', description: 'Full perception pipeline', req
  ],
};
```

packages/infrastructure/lib/config/services/scientific.service.ts

```
/**
 * Scientific Computing Service - Protein analysis and math reasoning
 */

import { MidLevelServiceConfig } from './index';

export const SCIENTIFIC_SERVICE: MidLevelServiceConfig = {
  id: 'scientific',
  name: 'scientific',
  displayName: 'Scientific Computing Service',
  description: 'Protein folding, embeddings, and computational biology pipelines',
  requiredModels: ['esm2-3b'],
  optionalModels: ['alphafold2', 'alphageometry', 'protenix'],
  defaultState: 'DISABLED',
  gracefulDegradation: true,
  pricing: { perRequest: 0.50, markup: 0.40 },
  minTier: 4,
  endpoints: [
    { path: '/scientific/protein/embed', method: 'POST', description: 'Generate protein embeddi
    { path: '/scientific/protein/fold', method: 'POST', description: 'Predict protein 3D struct
    { path: '/scientific/geometry/solve', method: 'POST', description: 'Solve geometry problems
  ],
};
```

packages/infrastructure/lib/config/services/medical.service.ts

```
/**
 * Medical Imaging Service - HIPAA-compliant medical analysis
 */
```

```typescript
import { MidLevelServiceConfig } from './index';

export const MEDICAL_SERVICE: MidLevelServiceConfig = {
  id: 'medical',
  name: 'medical',
  displayName: 'Medical Imaging Service',
  description: 'HIPAA-compliant medical image segmentation and analysis',
  requiredModels: ['medsam'],
  optionalModels: ['nnunet', 'whisper-large-v3'],
  defaultState: 'DISABLED',
  gracefulDegradation: true,
  pricing: { perImage: 0.15, perMinuteAudio: 0.08, markup: 0.40 },
  minTier: 4,
  endpoints: [
    { path: '/medical/segment', method: 'POST', description: 'Segment anatomical structures', r
    { path: '/medical/segment/3d', method: 'POST', description: 'Volumetric 3D segmentation', r
    { path: '/medical/transcribe', method: 'POST', description: 'Transcribe medical dictation'
  ],
};
```

packages/infrastructure/lib/config/services/geospatial.service.ts

```typescript
/**
 * Geospatial Analysis Service - Satellite imagery analysis
 */

import { MidLevelServiceConfig } from './index';

export const GEOSPATIAL_SERVICE: MidLevelServiceConfig = {
  id: 'geospatial',
  name: 'geospatial',
  displayName: 'Geospatial Analysis Service',
  description: 'Satellite imagery analysis for land use, flood detection, and crop mapping',
  requiredModels: ['prithvi-100m'],
  optionalModels: ['prithvi-600m', 'mobilesam'],
  defaultState: 'DISABLED',
  gracefulDegradation: true,
  pricing: { perImage: 0.08, markup: 0.40 },
  minTier: 4,
  endpoints: [
    { path: '/geospatial/classify', method: 'POST', description: 'Classify land use', requiredM
    { path: '/geospatial/detect/floods', method: 'POST', description: 'Detect flood extent', re
    { path: '/geospatial/change', method: 'POST', description: 'Detect changes between images'
  ],
};
```

packages/infrastructure/lib/config/services/reconstruction.service.ts

```typescript
/**
 * 3D Reconstruction Service - NeRF and 3D Gaussian Splatting
 */

import { MidLevelServiceConfig } from './index';

export const RECONSTRUCTION_SERVICE: MidLevelServiceConfig = {
  id: 'reconstruction',
  name: 'reconstruction',
  displayName: '3D Reconstruction Service',
  description: 'Generate 3D models from images and videos using neural radiance fields',
  requiredModels: ['nerfstudio'],
  optionalModels: ['3d-gaussian-splatting'],
  defaultState: 'DISABLED',
  gracefulDegradation: true,
  pricing: { per3DModel: 8.00, markup: 0.40 },
  minTier: 4,
  endpoints: [
    { path: '/reconstruction/nerf', method: 'POST', description: 'Create 3D model using NeRF',
    { path: '/reconstruction/gaussian', method: 'POST', description: 'Fast 3D using Gaussian sp
    { path: '/reconstruction/render', method: 'POST', description: 'Render novel views', requi
  ],
};
```

PART 3: THERMAL STATE MANAGEMENT

Thermal State Definitions

State	Description	Instance Count	Use Case
OFF	Completely disabled	0	Not needed, cost savings
COLD	Scales to zero when idle	0 (on-demand)	Infrequent use
WARM	Minimum instances always running	min (1+)	Regular use
HOT	Pre-scaled for high traffic	min + buffer	High demand
AUTOMATIC	AI-managed scaling	dynamic	Variable workloads

packages/infrastructure/lambda/thermal/manager.ts (Summary)

```typescript
/**
 * Thermal State Manager Lambda
 *
 * Key Functions:
 * - handleListModels(): List all models with thermal states
 * - handleGetModel(): Get specific model thermal state + metrics
 * - handleUpdateState(): Admin update thermal state
```

```
 * - handleBulkUpdate(): Bulk update multiple models
 * - handleWarmModel(): Request model warm-up (API users)
 * - handleScheduledCheck(): Auto-scaling and scale-to-zero logic
 *
 * Scheduled Tasks (every 5 minutes):
 * - Check transition status
 * - Handle AUTOMATIC scaling based on metrics
 * - Scale to zero inactive WARM/HOT models
 */

// API Routes
// GET  /thermal/models          - List all models
// GET  /thermal/models/:id      - Get model details
// PUT  /thermal/models/:id      - Update thermal state
// POST /thermal/bulk            - Bulk update
// POST /thermal/warm/:id        - Request warm-up
// GET  /thermal/metrics         - Get summary metrics
```

packages/infrastructure/lambda/thermal/notifier.ts (Summary)

```
/**
 * Client Notification Lambda
 *
 * Sends real-time notifications to clients via WebSocket when:
 * - Model warm-up starts
 * - Model becomes ready
 * - Model goes cold/offline
 * - Service state changes
 */
```

PART 4: DATABASE SCHEMA

packages/infrastructure/migrations/006_self_hosted_models.sql

```
-- Self-Hosted Model Registry
CREATE TABLE IF NOT EXISTS self_hosted_models (
    id VARCHAR(100) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    display_name VARCHAR(255) NOT NULL,
    description TEXT,
    category VARCHAR(50) NOT NULL,
    specialty VARCHAR(50) NOT NULL,
    sagemaker_image VARCHAR(500) NOT NULL,
    instance_type VARCHAR(50) NOT NULL,
    environment JSONB NOT NULL DEFAULT '{}',
    parameters BIGINT,
    accuracy VARCHAR(100),
```

```
    capabilities TEXT[] NOT NULL DEFAULT '{}',
    input_formats TEXT[] NOT NULL DEFAULT '{}',
    output_formats TEXT[] NOT NULL DEFAULT '{}',
    license VARCHAR(100) NOT NULL,
    commercial_use_notes TEXT,
    hourly_rate DECIMAL(10,4) NOT NULL,
    per_image DECIMAL(10,6),
    per_minute_audio DECIMAL(10,6),
    per_3d_model DECIMAL(10,4),
    markup_percent DECIMAL(5,2) NOT NULL DEFAULT 75.00,
    min_tier INTEGER NOT NULL DEFAULT 3,
    requires_gpu BOOLEAN NOT NULL DEFAULT true,
    gpu_memory_gb INTEGER NOT NULL,
    status VARCHAR(20) NOT NULL DEFAULT 'active',
    created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
);

-- Thermal States per app/environment
CREATE TABLE IF NOT EXISTS thermal_states (
    app_id VARCHAR(100) NOT NULL,
    environment VARCHAR(20) NOT NULL,
    model_id VARCHAR(100) NOT NULL REFERENCES self_hosted_models(id),
    current_state VARCHAR(20) NOT NULL DEFAULT 'COLD',
    target_state VARCHAR(20) NOT NULL DEFAULT 'COLD',
    instance_count INTEGER NOT NULL DEFAULT 0,
    min_instances INTEGER NOT NULL DEFAULT 0,
    max_instances INTEGER NOT NULL DEFAULT 3,
    last_activity TIMESTAMP WITH TIME ZONE,
    last_state_change TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    scale_to_zero_after_minutes INTEGER NOT NULL DEFAULT 15,
    warmup_time_seconds INTEGER NOT NULL DEFAULT 60,
    is_transitioning BOOLEAN NOT NULL DEFAULT false,
    error_message TEXT,
    updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    updated_by VARCHAR(100),
    PRIMARY KEY (app_id, environment, model_id),
    CONSTRAINT valid_thermal_state CHECK (current_state IN ('OFF', 'COLD', 'WARM', 'HOT', 'AUT
);

-- Mid-Level Services
CREATE TABLE IF NOT EXISTS mid_level_services (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    display_name VARCHAR(255) NOT NULL,
    description TEXT,
    required_models TEXT[] NOT NULL DEFAULT '{}',
    optional_models TEXT[] NOT NULL DEFAULT '{}',
```

```
    default_state VARCHAR(20) NOT NULL DEFAULT 'DISABLED',
    graceful_degradation BOOLEAN NOT NULL DEFAULT true,
    per_request DECIMAL(10,6),
    per_image DECIMAL(10,6),
    per_3d_model DECIMAL(10,4),
    markup_percent DECIMAL(5,2) NOT NULL DEFAULT 40.00,
    min_tier INTEGER NOT NULL DEFAULT 3,
    created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
);

-- Service States per app/environment
CREATE TABLE IF NOT EXISTS service_states (
    app_id VARCHAR(100) NOT NULL,
    environment VARCHAR(20) NOT NULL,
    service_id VARCHAR(50) NOT NULL REFERENCES mid_level_services(id),
    current_state VARCHAR(20) NOT NULL DEFAULT 'DISABLED',
    degraded_reason TEXT,
    available_models TEXT[] NOT NULL DEFAULT '{}',
    unavailable_models TEXT[] NOT NULL DEFAULT '{}',
    updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    PRIMARY KEY (app_id, environment, service_id),
    CONSTRAINT valid_service_state CHECK (current_state IN ('RUNNING', 'DEGRADED', 'DISABLED',
);

-- Model Usage Tracking (for billing)
CREATE TABLE IF NOT EXISTS model_usage (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    app_id VARCHAR(100) NOT NULL,
    environment VARCHAR(20) NOT NULL,
    tenant_id VARCHAR(100) NOT NULL,
    user_id VARCHAR(100),
    model_id VARCHAR(100) NOT NULL,
    service_id VARCHAR(50),
    operation VARCHAR(100) NOT NULL,
    processing_time_ms INTEGER NOT NULL,
    input_size_bytes INTEGER,
    output_size_bytes INTEGER,
    computed_cost DECIMAL(12,8) NOT NULL,
    request_id VARCHAR(100),
    metadata JSONB,
    created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_model_usage_tenant ON model_usage(tenant_id);
CREATE INDEX idx_model_usage_model ON model_usage(model_id);
CREATE INDEX idx_model_usage_created ON model_usage(created_at);
```

PART 5: ESTIMATED COSTS BY TIER

Self-Hosted Model Costs (Monthly)

Component	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
SageMaker (Vision)	N/A	N/A	~$150	~$400	~$1,000
SageMaker (Audio)	N/A	N/A	~$100	~$250	~$600
SageMaker (Scientific)	N/A	N/A	N/A	~$300	~$800
SageMaker (Medical)	N/A	N/A	N/A	~$200	~$500
SageMaker (Geospatial)	N/A	N/A	N/A	~$150	~$400
SageMaker (3D)	N/A	N/A	N/A	~$200	~$500
SageMaker (LLMs)	N/A	N/A	~$100	~$500	~$2,000
Lambda (Thermal/Services)	N/A	N/A	~$30	~$80	~$250
DynamoDB (States)	N/A	N/A	~$5	~$15	~$50
Prompt 6 Total	**N/A**	**N/A**	**~$385**	**~$2,095**	**~$6,100**

Notes: - Self-hosted models require Tier 3+ (GROWTH tier or above) - Costs assume models in COLD state (scale to zero when idle) - 75% markup on SageMaker costs included for billing

---

MODEL SUMMARY BY CATEGORY

Computer Vision (19 models)

Category	Count	Models
Classification	8	EfficientNet-B0/B5/V2-L, Swin-T/B/L, CLIP-B32/L14
Detection	7	YOLOv8n/s/m/x, YOLOv11x, RT-DETR-X, Grounding DINO
Segmentation	4	SAM ViT-H, SAM 2, MobileSAM, Mask R-CNN

Audio/Speech (6 models)

Category	Count	Models
Speech-to-Text	3	Parakeet TDT 1.1B, Whisper Large V3, Whisper Turbo
Speaker	3	TitaNet-Large, ECAPA-TDNN, pyannote Diarization

Scientific (4 models)

Category	Count	Models
Protein	3	AlphaFold2, ESM-2 3B, Protenix
Math	1	AlphaGeometry

Medical (2 models)

Category	Count	Models
Imaging	2	nnU-Net, MedSAM

Geospatial (2 models)

Category	Count	Models
Satellite	2	Prithvi 100M, Prithvi 600M

3D/Generative (2 models)

Category	Count	Models
Reconstruction	2	Nerfstudio, 3D Gaussian Splatting

Text Generation (3 models)

Category	Count	Models
LLMs	3	Mistral 7B, Llama 3 70B, Qwen 72B

Total: 38 self-hosted models

API ROUTES SUMMARY

Thermal Management

Method	Path	Description	Permission
GET	/thermal/models	List all models with states	settings:read
GET	/thermal/models/:id	Get model thermal state	settings:read
PUT	/thermal/models/:id	Update thermal state	settings:write
POST	/thermal/bulk	Bulk update states	settings:write
POST	/thermal/warm/:id	Request model warm-up	(API users)

Mid-Level Services

Method	Path	Description
POST	/perception/detect	Object detection
POST	/perception/segment	Image segmentation
POST	/perception/classify	Image classification
POST	/perception/analyze	Full pipeline
POST	/scientific/protein/embed	Protein embeddings
POST	/scientific/protein/fold	Protein structure
POST	/medical/segment	Medical segmentation
POST	/geospatial/classify	Land use classification
POST	/reconstruction/nerf	3D reconstruction

DEPLOYMENT VERIFICATION

```
# 1. Check thermal states
curl -H "Authorization: Bearer $ADMIN_TOKEN" \
  https://admin-api.YOUR_DOMAIN.com/api/v2/thermal/models

# 2. Update thermal state
curl -X PUT -H "Authorization: Bearer $ADMIN_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"targetState": "WARM"}' \
  https://admin-api.YOUR_DOMAIN.com/api/v2/thermal/models/yolov8m

# 3. Test object detection
curl -X POST -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"imageUrl": "https://YOUR_DOMAIN.com/image.jpg"}' \
  https://api.YOUR_DOMAIN.com/api/v2/perception/detect

# 4. Check service status
curl -H "Authorization: Bearer $TOKEN" \
  https://api.YOUR_DOMAIN.com/api/v2/perception/status
```

NEXT PROMPTS

Continue with: - **Prompt 7**: External Providers & Database Schema/Migrations - **Prompt 8**: Admin Web Dashboard (Next.js) - **Prompt 9**: Assembly & Deployment Guide

---

End of Prompt 6: Self-Hosted Models & Mid-Level Services RADIANT v2.2.0 - December 2024

â• â

END OF SECTION 6

â• â

â• â