perplexity

□ CREDITX ECOSYSTEM - ENTERPRISE PRODUCTION BUILD MAP

SYSTEM ARCHITECTURE OVERVIEW

```
┌─────────────────────────────────────────────────────────────────────────────────┐
│                        FRONTEND LAYER                            │
│  Next.js 14 App Router + CopilotKit + TailwindCSS             │
└─────────────────────────┬───────────────────────────────────────────────────────┘
                     │
                     ▼

┌─────────────────────────────────────────────────────────────────────────────────┐
│                    COPILOTKIT ABSTRACTION                        │
│     AG-UI Protocol | Shared State | HITL | Generative UI       │
└─────────────────────────┬───────────────────────────────────────────────────────┘
                     │
        ┌───────────────┼───────────────┐
        ▼             ▼             ▼

   ┌───────────┐   ┌───────────┐  ┌────────────┐
   │LangGraph│  │Agent    │   │Direct   │
   │Backend  │  │Router   │   │LLM      │
   └─────┬─────┘   └─────┬─────┘  └─────┬──────┘
        │            │            │
        └───────────────┼───────────────┘
                     ▼

┌─────────────────────────────────────────────────────────────────────────────────┐
│                     BACKEND SERVICES LAYER                       │
│    5 Module Orchestrators + Shared Services                     │
└─────────────────────────┬───────────────────────────────────────────────────────┘
                     │
        ┌───────────────┼───────────────┬───────────────┐
        ▼             ▼             ▼             ▼

   ┌───────────┐   ┌───────────┐  ┌────────────┐  ┌───────────┐
   │PostgreSQL│  │Redis    │   │S3/Blob  │  │External │
   │Multi-    │  │Cache/   │   │Storage  │  │APIs     │
   │Tenant    │  │Queue    │   │         │  │CRM/Email│
   └───────────┘   └───────────┘  └────────────┘  └───────────┘
                     │
                     ▼

┌─────────────────────────────────────────────────────────────────────────────────┐
│                   INFRASTRUCTURE LAYER                           │
│   Spaceship Starlight VMs | Hyperlift CI/CD | CDN | LB          │
└─────────────────────────────────────────────────────────────────────────────────┘
```

MODULE FUNCTION FLOW MAP

MODULE 1: CREDITX COMPLIANCE AUTOMATION

Feature Specifications

- **KYC Document Generation**: 5-second generation per report
- **AML Sanctions Screening**: 500ms per transaction check
- **Audit Trail Management**: Real-time compliance monitoring
- **Regulatory Reporting**: 10-K, SOX evidence auto-collection

Function Flow

```
USER ACTION → FRONTEND TOOL → AGENT DECISION → BACKEND ACTION → DATABASE → RESPONSE

1. Upload Transaction Data
   ↓
2. useCopilotAction("uploadTransactionData")
   ↓
3. Agent validates & routes to CreditX module
   ↓
4. Backend: sanctionsScreening(transaction)
   ↓
5. External API: WorldCheck/OFAC lookup
   ↓
6. PostgreSQL: INSERT audit_log, UPDATE compliance_status
   ↓
7. Agent generates compliance document
   ↓
8. S3: Store PDF report
   ↓
9. Generative UI: Display results + approval workflow
   ↓
10. HITL: Human approves/rejects
   ↓
11. Email notification via Spacemail
```

Technical Stack

- **Agent**: LangGraph workflow with checkpoints
- **OCR**: TensorFlow custom model for document extraction
- **Database**: `creditx_compliance` schema per tenant
- **APIs**: WorldCheck, OFAC, SWIFT sanctions
- **Storage**: S3 for regulatory documents (AES-256)

Database Schema

```
-- creditx_compliance.transactions
CREATE TABLE transactions (
  id UUID PRIMARY KEY,
  tenant_id INTEGER REFERENCES tenants(id),
  transaction_date TIMESTAMPTZ,
  amount DECIMAL(15,2),
  currency VARCHAR(3),
  counterparty VARCHAR(255),
  sanctions_status VARCHAR(50), -- CLEAR, FLAGGED, BLOCKED
  compliance_score INTEGER, -- 0-100
  kyc_document_url TEXT,
  audit_log_id UUID,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- creditx_compliance.audit_logs
CREATE TABLE audit_logs (
  id UUID PRIMARY KEY,
  tenant_id INTEGER,
  action VARCHAR(100),
  user_id UUID,
  resource_type VARCHAR(50),
  resource_id UUID,
  changes JSONB,
  timestamp TIMESTAMPTZ DEFAULT NOW(),
  ip_address INET
);
```

API Endpoints

```
POST   /api/creditx/transactions/upload
POST   /api/creditx/sanctions/screen
GET    /api/creditx/compliance/reports/:reportId
POST   /api/creditx/kyc/generate
PATCH  /api/creditx/transactions/:id/approve
GET    /api/creditx/audit-trail
```

MODULE 2: 91 APPS BUSINESS AUTOMATION

Feature Specifications

- **Lead Scoring**: 100ms per lead, ML-based qualification
- **PO Automation**: 5-second order creation, supplier integration
- **Working Capital Optimization**: 20-30 day cycle time reduction
- **Campaign Orchestration**: Multi-channel automation

Function Flow

```
TRIGGER EVENT → AGENT ANALYZES → AUTOMATION EXECUTES → STATE UPDATE

1. New Lead Enters CRM (Salesforce webhook)
   ↓
2. Event published to Redis queue
   ↓
3. 91Apps Agent consumes event
   ↓
4. useCopilotReadable: Shares lead context with agent
   ↓
5. Agent calls leadScoring(leadData)
   ↓
6. ML Model inference: Score 0-100
   ↓
7. Backend: updateLeadScore(leadId, score)
   ↓
8. Salesforce API: Update lead record
   ↓
9. useCopilotAction: "createFollowUpTask"
   ↓
10. Shared State: UI updates real-time
   ↓
11. Agent generates email draft
   ↓
12. Generative UI: Show email editor
   ↓
13. HITL: User reviews/approves
   ↓
14. Gmail API: Send email
   ↓
15. PostgreSQL: Log activity
```

Technical Stack

- **Agent**: LangGraph + Pydantic AI for type-safe operations
- **ML Models**: Qwen-2.5-7B fine-tuned on sales data
- **Cache**: Redis for sub-millisecond workflow state
- **Queue**: Redis Bull for background jobs
- **Integrations**: Salesforce, SAP, NetSuite, Gmail, LinkedIn

Database Schema

```
-- apps_91.leads
CREATE TABLE leads (
  id UUID PRIMARY KEY,
  tenant_id INTEGER,
  external_id VARCHAR(255), -- Salesforce ID
  name VARCHAR(255),
  email VARCHAR(255),
  company VARCHAR(255),
  status VARCHAR(50), -- new, qualified, engaged, converted
  score INTEGER, -- 0-100
  last_activity_at TIMESTAMPTZ,
  assigned_to UUID,
  metadata JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- apps_91.automation_workflows
CREATE TABLE automation_workflows (
  id UUID PRIMARY KEY,
  tenant_id INTEGER,
  workflow_type VARCHAR(50), -- lead_scoring, po_creation, email_campaign
  trigger_event VARCHAR(100),
  conditions JSONB,
  actions JSONB,
  status VARCHAR(20), -- active, paused, completed
  execution_count INTEGER DEFAULT 0,
  last_executed_at TIMESTAMPTZ,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- apps_91.workflow_executions
CREATE TABLE workflow_executions (
  id UUID PRIMARY KEY,
  workflow_id UUID REFERENCES automation_workflows(id),
  tenant_id INTEGER,
  input_data JSONB,
  output_data JSONB,
  status VARCHAR(20), -- pending, running, completed, failed
  error_message TEXT,
  duration_ms INTEGER,
  executed_at TIMESTAMPTZ DEFAULT NOW()
);
```

API Endpoints

```
POST   /api/91apps/leads/score
POST   /api/91apps/workflows/create
POST   /api/91apps/workflows/:id/execute
GET    /api/91apps/workflows/:id/executions
POST   /api/91apps/purchase-orders/create
PATCH  /api/91apps/leads/:id
GET    /api/91apps/analytics/dashboard
POST   /api/91apps/integrations/salesforce/sync
POST   /api/91apps/emails/send
```

MODULE 3: GLOBAL AI ALERT NETWORK

Feature Specifications

- **Packet Inspection**: 10M packets/second, DNS threat detection
- **Breach Detection**: 7-day avg vs 279-day industry avg
- **Threat Scoring**: ML-based anomaly detection
- **Alert Latency**: <5 seconds from detection to notification

Function Flow

```
NETWORK TRAFFIC → PACKET CAPTURE → ML ANALYSIS → ALERT GENERATION

1. Network packet captured (libpcap)
   ↓
2. DNS query logged
   ↓
3. Packet metadata extracted
   ↓
4. Redis: Publish to analysis queue
   ↓
5. Global AI Alert Agent consumes
   ↓
6. ML Model: Threat scoring (PyTorch)
   ↓
7. PostgreSQL: INSERT threat_intelligence
   ↓
8. If threat_score > 70:
   ↓
9. Agent triggers alerting workflow
   ↓
10. Thunderbolt: E2EE notification to SOC
   ↓
11. useCopilotAction: "showThreatDashboard"
   ↓
12. Generative UI: Real-time threat map
   ↓
13. HITL: Analyst reviews threat
   ↓
14. Backend: executePlaybook(threatId, action)
   ↓
15. Network segmentation enforcement
```

Technical Stack

- **Agent**: LangGraph with streaming responses
- **ML**: PyTorch for threat detection, custom CNN model
- **Packet Capture**: libpcap library (requires root access)
- **Database**: TimescaleDB extension for time-series data

- **Communication**: Thunderbolt E2EE (Signal Protocol)

Database Schema

```
-- global_ai_alert.threat_intelligence
CREATE TABLE threat_intelligence (
  id UUID PRIMARY KEY,
  tenant_id INTEGER,
  source_ip INET,
  dest_ip INET,
  dns_query TEXT,
  packet_metadata JSONB,
  threat_type VARCHAR(50), -- c2_beacon, exfiltration, lateral_movement
  threat_score INTEGER, -- 0-100
  severity VARCHAR(20), -- low, medium, high, critical
  detected_at TIMESTAMPTZ DEFAULT NOW(),
  resolved_at TIMESTAMPTZ,
  resolution VARCHAR(100)
);

-- global_ai_alert.network_devices
CREATE TABLE network_devices (
  id UUID PRIMARY KEY,
  tenant_id INTEGER,
  device_type VARCHAR(50), -- iot, server, workstation, mobile
  mac_address MACADDR,
  ip_address INET,
  hostname VARCHAR(255),
  last_seen_at TIMESTAMPTZ,
  baseline_profile JSONB, -- ML behavioral baseline
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

API Endpoints

```
POST   /api/global-ai-alert/packets/ingest
GET    /api/global-ai-alert/threats/active
POST   /api/global-ai-alert/threats/:id/investigate
PATCH  /api/global-ai-alert/threats/:id/resolve
GET    /api/global-ai-alert/dashboard/real-time
POST   /api/global-ai-alert/playbooks/:id/execute
GET    /api/global-ai-alert/devices/:tenantId
```

MODULE 4: GUARDIAN AI ENDPOINT SECURITY

Feature Specifications

- **Endpoint Monitoring**: Windows, macOS, iOS, Android agents
- **Behavioral Analysis**: TensorFlow Lite on-device ML
- **Isolation Response**: 5-second breach containment
- **Breach Prevention**: 80% of endpoint-origin breaches blocked

Function Flow

```
ENDPOINT TELEMETRY → AGENT ANALYSIS → ANOMALY DETECTION → AUTO-ISOLATION

1. Endpoint agent (installed on device)
   ↓
2. Monitors: Process execution, file changes, network connections
   ↓
3. Telemetry stream: 100 events/sec per device
   ↓
4. HTTPS POST to /api/guardian-ai/telemetry/ingest
   ↓
5. Redis: Buffer events
   ↓
6. Guardian AI Agent batch processes
   ↓
7. TensorFlow Lite: Behavioral analysis
   ↓
8. Compare against baseline_profile
   ↓
9. If anomaly_score > 85:
   ↓
10. PostgreSQL: INSERT alert
   ↓
11. Agent generates isolation command
   ↓
12. useCopilotAction: "isolateEndpoint"
   ↓
13. Generative UI: Show incident response workflow
   ↓
14. HITL: Analyst confirms isolation
   ↓
15. WebSocket: Push isolation command to endpoint
   ↓
16. Endpoint agent: Network isolation enforced
   ↓
17. Spacemail: Incident notification
```

Technical Stack

- **Agent**: CrewAI for multi-agent coordination
- **ML**: TensorFlow Lite (on-device), PyTorch (server-side)
- **Endpoint Agents**: Electron (cross-platform), native Swift/Kotlin

- **Real-time**: WebSocket for command/control
- **CDN**: Spaceship CDN for agent downloads

Database Schema

```
-- guardian_ai.endpoints
CREATE TABLE endpoints (
  id UUID PRIMARY KEY,
  tenant_id INTEGER,
  device_id VARCHAR(255) UNIQUE,
  device_type VARCHAR(50),
  os_version VARCHAR(100),
  agent_version VARCHAR(20),
  last_checkin_at TIMESTAMPTZ,
  status VARCHAR(20), -- online, offline, isolated
  baseline_established BOOLEAN DEFAULT FALSE,
  baseline_data JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- guardian_ai.endpoint_events
CREATE TABLE endpoint_events (
  id UUID PRIMARY KEY,
  endpoint_id UUID REFERENCES endpoints(id),
  tenant_id INTEGER,
  event_type VARCHAR(50), -- process_start, file_change, network_connect
  event_data JSONB,
  anomaly_score INTEGER, -- 0-100
  flagged BOOLEAN DEFAULT FALSE,
  timestamp TIMESTAMPTZ DEFAULT NOW()
);

-- guardian_ai.incidents
CREATE TABLE incidents (
  id UUID PRIMARY KEY,
  endpoint_id UUID REFERENCES endpoints(id),
  tenant_id INTEGER,
  incident_type VARCHAR(50),
  severity VARCHAR(20),
  description TEXT,
  status VARCHAR(20), -- open, investigating, resolved, false_positive
  isolated_at TIMESTAMPTZ,
  resolved_at TIMESTAMPTZ,
  resolution_notes TEXT,
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

API Endpoints

```
POST   /api/guardian-ai/telemetry/ingest
GET    /api/guardian-ai/endpoints/:tenantId
POST   /api/guardian-ai/endpoints/:id/isolate
POST   /api/guardian-ai/endpoints/:id/restore
GET    /api/guardian-ai/incidents/active
PATCH  /api/guardian-ai/incidents/:id/resolve
GET    /api/guardian-ai/agents/download/:platform
POST   /api/guardian-ai/baselines/:endpointId/establish
```

MODULE 5: STOLEN/LOST PHONES DEVICE RECOVERY

Feature Specifications

- **GPS Tracking**: 5-second location query
- **Recovery Rate**: 70-80% vs 30% industry avg
- **PHI Breach Prevention**: 90% reduction in lost-device breaches
- **Chain-of-Custody**: Immutable audit trail for law enforcement

Function Flow

```
DEVICE STOLEN REPORT → GPS TRACKING → RECOVERY PLAYBOOK → INSURANCE CLAIM

1. User reports device stolen (mobile app or web)
   ↓
2. POST /api/stolen-phones/devices/:id/report-stolen
   ↓
3. PostgreSQL: UPDATE device status = 'stolen'
   ↓
4. Device agent (background service on phone)
   ↓
5. GPS/cellular triangulation activated
   ↓
6. Telemetry stream: Location updates every 30 seconds
   ↓
7. Stolen Phones Agent receives location
   ↓
8. useCopilotReadable: Share device location with agent
   ↓
9. Agent executes recovery playbook:
   - Device lock (biometric enforcement)
   - Data wipe preparation
   - Chain-of-custody logging
   ↓
10. Generative UI: Real-time location map
   ↓
11. useCopilotAction: "notifyAuthorities"
   ↓
12. HITL: User decides to alert law enforcement
```

```
   ↓
13. Thunderbolt E2EE: Notify recovery team
   ↓
14. Chain-of-custody: Immutable audit log
   ↓
15. If device recovered:
   ↓
16. Agent triggers insurance API integration
   ↓
17. Automated claim processing
```

Technical Stack

- **Agent**: Agno (rapid development, simple API)
- **GPS**: Native iOS/Android location services
- **MDM**: Integration with Intune, Jamf, VMware Workspace ONE
- **Database**: `stolen_phones` schema with JSONB for telemetry
- **Communication**: Thunderbolt E2EE for sensitive location data

Database Schema

```sql
-- stolen_phones.devices
CREATE TABLE devices (
  id UUID PRIMARY KEY,
  tenant_id INTEGER,
  device_id VARCHAR(255) UNIQUE,
  owner_user_id UUID,
  device_type VARCHAR(50),
  os_version VARCHAR(100),
  status VARCHAR(20), -- active, stolen, recovered, wiped
  last_location GEOGRAPHY(POINT, 4326),
  last_location_at TIMESTAMPTZ,
  stolen_at TIMESTAMPTZ,
  recovered_at TIMESTAMPTZ,
  insurance_claim_id VARCHAR(100),
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- stolen_phones.location_history
CREATE TABLE location_history (
  id UUID PRIMARY KEY,
  device_id UUID REFERENCES devices(id),
  tenant_id INTEGER,
  location GEOGRAPHY(POINT, 4326),
  accuracy_meters INTEGER,
  location_method VARCHAR(50), -- gps, cellular, wifi
  timestamp TIMESTAMPTZ DEFAULT NOW()
);

-- stolen_phones.recovery_workflows
CREATE TABLE recovery_workflows (
  id UUID PRIMARY KEY,
  device_id UUID REFERENCES devices(id),
  tenant_id INTEGER,
  workflow_status VARCHAR(50),
  playbook_actions JSONB,
  authorities_notified BOOLEAN DEFAULT FALSE,
  insurance_claim_filed BOOLEAN DEFAULT FALSE,
  chain_of_custody JSONB[], -- Immutable array
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

API Endpoints

```
POST   /api/stolen-phones/devices/:id/report-stolen
POST   /api/stolen-phones/devices/:id/location/update
GET    /api/stolen-phones/devices/:id/location/history
POST   /api/stolen-phones/devices/:id/lock
POST   /api/stolen-phones/devices/:id/wipe
POST   /api/stolen-phones/devices/:id/notify-authorities
POST   /api/stolen-phones/insurance/claim
GET    /api/stolen-phones/workflows/:deviceId
```

SHARED SERVICES ARCHITECTURE

Tenant Management Service

Database Schema

```
-- Core multi-tenancy
CREATE TABLE tenants (
  id SERIAL PRIMARY KEY,
  external_id UUID UNIQUE DEFAULT gen_random_uuid(),
  name VARCHAR(255) NOT NULL, -- "Nuvei", "Revau", etc.
  domain VARCHAR(255) UNIQUE, -- "nuvei.ecosystem.ai"
  schema_name VARCHAR(63) UNIQUE, -- "tenant_001_nuvei"
  status VARCHAR(20) DEFAULT 'active',
  modules_enabled TEXT[], -- ['creditx', '91apps', 'guardian_ai']
  settings JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Tenant users
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id INTEGER REFERENCES tenants(id),
  email VARCHAR(255) UNIQUE NOT NULL,
  name VARCHAR(255),
  role VARCHAR(50), -- admin, manager, user
  auth_provider VARCHAR(50), -- oauth, saml
  auth_provider_id VARCHAR(255),
  permissions JSONB,
  last_login_at TIMESTAMPTZ,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- RLS Policy
ALTER TABLE users ENABLE ROW LEVEL SECURITY;

CREATE POLICY tenant_isolation ON users
  USING (tenant_id = current_setting('app.current_tenant_id')::INTEGER);
```

API Endpoints

```
POST   /api/tenants/create
GET    /api/tenants/:tenantId
PATCH  /api/tenants/:tenantId/settings
POST   /api/tenants/:tenantId/modules/enable
GET    /api/tenants/:tenantId/users
POST   /api/tenants/:tenantId/users/invite
```

Authentication & Authorization Service

OAuth 2.0 Flow

```
1. User navigates to nuvei.ecosystem.ai
   ↓
2. Domain-based routing: Identify tenant
   ↓
3. Redirect to /api/auth/oauth/authorize?tenant_id=001
   ↓
4. OAuth provider (Google, Microsoft, Okta)
   ↓
5. Callback: /api/auth/oauth/callback
   ↓
6. Exchange code for token
   ↓
7. JWT signed with tenant_id claim
   ↓
8. Set session cookie (httpOnly, secure, sameSite)
   ↓
9. Middleware: Extract tenant_id from JWT
   ↓
10. SET app.current_tenant_id = tenant_id
   ↓
11. All queries filtered by RLS policy
```

API Endpoints

```
GET    /api/auth/oauth/authorize
POST   /api/auth/oauth/callback
POST   /api/auth/logout
GET    /api/auth/session
POST   /api/auth/refresh
```

Integration Hub Service

Salesforce Integration

```
// OAuth flow for Salesforce
POST   /api/integrations/salesforce/connect
GET    /api/integrations/salesforce/callback
POST   /api/integrations/salesforce/sync
GET    /api/integrations/salesforce/leads
POST   /api/integrations/salesforce/leads/:id/update
```

Database Schema

```
CREATE TABLE integration_connections (
  id UUID PRIMARY KEY,
  tenant_id INTEGER REFERENCES tenants(id),
  integration_type VARCHAR(50), -- salesforce, sap, netsuites
  credentials JSONB, -- Encrypted access tokens
  settings JSONB,
  last_sync_at TIMESTAMPTZ,
  status VARCHAR(20), -- active, error, disconnected
  created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE integration_sync_logs (
  id UUID PRIMARY KEY,
  connection_id UUID REFERENCES integration_connections(id),
  sync_type VARCHAR(50), -- full, incremental
  records_processed INTEGER,
  errors INTEGER,
  duration_ms INTEGER,
  started_at TIMESTAMPTZ,
  completed_at TIMESTAMPTZ
);
```

CI/CD PIPELINE ARCHITECTURE

GitHub Actions Workflow

```
# .github/workflows/deploy.yml

name: Production Deploy to Spaceship

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm ci
      - run: npm run lint
      - run: npm run type-check
      - run: npm run test
      - run: npm run test:e2e

  build:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: docker/setup-buildx-action@v3
      - uses: docker/login-action@v3
        with:
          registry: registry.spaceship.com
          username: ${{ secrets.SPACESHIP_USERNAME }}
          password: ${{ secrets.SPACESHIP_TOKEN }}

      - name: Build Images
        run: |
          docker build -t creditx-frontend:${{ github.sha }} -f docker/Dockerfile.fronter
          docker build -t creditx-agent:${{ github.sha }} -f docker/Dockerfile.agent .
          docker build -t creditx-api:${{ github.sha }} -f docker/Dockerfile.api .

      - name: Push to Registry
        run: |
          docker push registry.spaceship.com/creditx-frontend:${{ github.sha }}
          docker push registry.spaceship.com/creditx-agent:${{ github.sha }}
          docker push registry.spaceship.com/creditx-api:${{ github.sha }}

  deploy:
    needs: build
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'
    steps:
      - name: Deploy to Spaceship Hyperlift
        run: |
          curl -X POST https://hyperlift.spaceship.com/deploy \
            -H "Authorization: Bearer ${{ secrets.HYPERLIFT_TOKEN }}" \
            -d '{
              "project": "creditx-ecosystem",
              "environment": "production",
              "images": {
                "frontend": "creditx-frontend:${{ github.sha }}",
                "agent": "creditx-agent:${{ github.sha }}",
                "api": "creditx-api:${{ github.sha }}"
              },
              "strategy": "blue-green",
              "healthCheck": "/api/health",
              "rollbackOnFailure": true
            }'
```

DOCKER CONTAINERIZATION STRATEGY

Frontend Container (Dockerfile.frontend)

```
FROM node:20-alpine AS base
WORKDIR /app

# Dependencies
COPY package*.json ./
RUN npm ci --only=production

# Build
COPY . .
RUN npm run build

# Runtime
FROM node:20-alpine AS runner
WORKDIR /app
ENV NODE_ENV=production
COPY --from=base /app/.next/standalone ./
COPY --from=base /app/.next/static ./.next/static
COPY --from=base /app/public ./public

EXPOSE 3000
CMD ["node", "server.js"]
```

Agent Container (Dockerfile.agent)

```
FROM python:3.12-slim
WORKDIR /app

# System dependencies
```

```
RUN apt-get update && apt-get install -y \
    libpcap-dev \
    build-essential \
    && rm -rf /var/lib/apt/lists/*

# Python dependencies
COPY agent/requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Application code
COPY agent/ .

EXPOSE 8000
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

API Container (Dockerfile.api)

```
FROM node:20-alpine
WORKDIR /app

COPY package*.json ./
RUN npm ci --only=production

COPY api/ .

EXPOSE 4000
CMD ["node", "index.js"]
```

MONITORING & OBSERVABILITY

Launchpad Dashboard Configuration

```
// lib/observability/config.ts

export const observabilityConfig = {
  metrics: {
    latency: {
      p50: 100, // ms
      p95: 500,
      p99: 2000
    },
    throughput: {
      target: 1000, // req/sec
      alert: 500
    },
    errorRate: {
      threshold: 0.02 // 2%
    }
  },

  alerts: {
    channels: ['slack', 'email', 'pagerduty'],
    conditions: [
      { metric: 'latency.p95', operator: '>', value: 10000, severity: 'critical' },
      { metric: 'errorRate', operator: '>', value: 0.05, severity: 'high' },
      { metric: 'cpu', operator: '>', value: 80, severity: 'medium' }
    ]
  },

  tracing: {
    enabled: true,
    sampleRate: 0.1,
    exportInterval: 30000
  }
};
```

DEPLOYMENT CHECKLIST

Pre-Deployment (Jan 16-17, 2026)

- [ ] Provision 15 Starlight VMs (Phoenix + Singapore)
- [ ] Configure 5 Load Balancers
- [ ] Attach 15 Starlight Volumes
- [ ] Setup CDN: cdn.ecosystem.ai
- [ ] Configure Thunderbolt: team.ecosystem.ai
- [ ] Provision Spacemail: support@creditx.ai
- [ ] Configure FastVPN (20 team members)
- [ ] Connect GitHub repos to Hyperlift
- [ ] Setup PostgreSQL multi-tenant schemas (45 tenants)
- [ ] Configure Redis clusters
- [ ] Setup S3 buckets with encryption

Go-Live (Jan 18, 2026)

- [ ] DNS cutover to Starlight VMs
- [ ] Deploy all 5 module containers
- [ ] Run integration tests
- [ ] Validate 99.99% uptime monitoring
- [ ] Load test: 10,000 concurrent users
- [ ] Security scan: OWASP Top 10
- [ ] Performance benchmark: <5s module response
- [ ] Notify NovaCap: Production ready

NEXT STEP

Ready to generate all production code files? This build map provides the complete blueprint. I'll now create:

1. ✅ **45 code files** covering all modules
2. ✅ **Dockerfiles** for each service
3. ✅ **GitHub Actions** CI/CD pipeline
4. ✅ **Database migration scripts**
5. ✅ **API route handlers**
6. ✅ **Agent orchestration code**
7. ✅ **Integration connectors**
8. ✅ **Monitoring configuration**

Confirm to proceed with code generation?[1] [2]

⁂

1. creditX_ecosystem.pdf
2. CopilotKit-Documentation-Complete-Reference-for.pdf
3. creditX-and-quifax-comparisons.pdf
4. Nova-Cap-Headwinds.pdf
5. Nova-Cap-The-Ecosystem-Alignment.pdf
6. The-ICP-Market-Size-The-Ecosystem.pdf
7. creditX_ecosystem.pdf
8. AI-Signals-creditX.pdf
9. 9-Phase-Build-History-Map_creditX.pdf
10. creditX-Platform-Consumer-Agent-Catologue-1-10-of-agents-1-22.pdf
11. system-flow-visual.jpg
12. novacap-dply.jpg
13. visual-image.jpg
14. cicd_pipeline.jpg
15. creditx_architecture-1.jpg
16. creditx_architecture.jpg
17. creditX-Ecosystem-Portfolio-Integration-Architecture.pdf
18. creditX-7Day-Build-ClientPackages.md.pdf
19. CopilotKit-Implementation-Guide.md.pdf
20. AgentKit-Full-Documentation.md.pdf
21. CopilotKit-Documentation-Complete-Reference-for.pdf
22. creditX-Ecosystem-Portfolio-Integration-Architecture.pdf
23. creditX-7Day-Build-ClientPackages.md.pdf

24. CopilotKit-Documentation-Complete-Reference-for.pdf
25. Spaceship-Host-and-Deploy.pdf
26. AgentKit-Implementation-Guide.md.pdf
27. AgentKit-Full-Documentation.md.pdf
28. CopilotKit-Documentation-Complete-Reference-for.pdf
29. Spaceship-Host-and-Deploy.pdf
30. Terms-NOVACAP-ECOSYSTEM-PARTNER.md.pdf
31. under-review-offerTHE-ECOSYSTEM-x-NOVACAP.pdf