Clawn VPN MVP Blueprint

Overview & Intent

This blueprint explains how Clawn VPN will be built as a web-first VPN management platform. The system will run in the browser first, and later a mobile app can be added without rebuilding or replacing the backend. The backend remains the central control system, and every device (web or mobile) will simply connect to it.

The goal is to design the platform in a way that is stable, secure, and scalable from the very beginning, instead of rushing features that will break later.
We prioritize reliability, security, and correct behavior before adding more features or more users. A small system that works correctly is better than a big system that collapses.

1) Control Plane — Web App + API (The "Brain" of the Platform)

This is the central decision-making system. All users, devices, peers, and regions are managed here. Architecture Model (Web → Mobile Ready)

The Clawn VPN platform is built around **three core subsystems**. Each subsystem has a specific role and responsibility. Separating them ensures security, reliability, and clean scaling as more users and regions are added.

The three subsystems are:

1. **Control Plane (Web App + API)**
2. **VPN Data Plane (WireGuard Servers in Each Region)**
3. **Configuration Delivery Layer (How Users Receive VPN Configs & Device Controls)**

Each one is explained below.

It is built using:

- **Next.js (hosted on Vercel)** — for the web dashboard and frontend
- **Supabase (Auth + Postgres database)** — for authentication and data storage
- **Row Level Security (RLS enforced)** — ensures users only access their own data
- **Service-role admin functions** — restricted backend operations for provisioning and management

The control plane handles:

- user accounts
- device registration limits
- assigning servers to users

- creating peer entries
- revoking or rotating keys
- enforcing policies
- maintaining audit logs

Nothing happens directly on the VPN servers without going through this layer.

This prevents:

- uncontrolled device creation
- "rogue" peers
- unsafe key distribution
- cross-user data exposure

The control plane ensures the platform remains **predictable, secure, and manageable at scale**.

2) VPN Data Plane — Regional WireGuard Nodes (The "Network Layer")

This layer consists of **multiple VPN servers deployed across regions**.

Each server runs:

- **WireGuard**
- on a **dedicated VPS per region**

Example regions include:

- USA
- Singapore
- Germany
- Australia
- Japan

Each node must enforce:

- strict firewall rules
- peer limits (prevent overload)
- access isolation
- enforced peer revocation

This ensures:

- unknown peers cannot connect
- revoked users lose access immediately
- stale peers do not remain active
- no cross-region leakage

The VPN data plane is optimized for:

- speed
- stability
- low latency
- predictable performance

3) MVP Functional Scope (What Must Exist)

Payments, freemium tiers, and mobile are deferred until platform stability is validated. The MVP delivers a **secure, controlled, and scalable VPN platform** with a web-based control plane. It prioritizes reliability, lifecycle control, and predictable behavior over aggressive growth.

The MVP must support the following functional capabilities.

1) Account Login & Authentication

Users must authenticate before accessing any VPN features.

Authentication ensures:

- every device + peer is tied to a real account
- no anonymous access exists
- actions are auditable and reversible

Future-ready capabilities:

- billing integration
- device ownership enforcement
- abuse tracking & risk control

Authentication is the foundation of system trust.

2) Device Registration (With Limited Slots Per User)

Each user may register only a limited number of devices.

MVP default:

- **1 device slot per user** (expandable later)

When a device is added:

- it consumes a slot
- it becomes part of the device lifecycle
- its peer belongs only to that user

This prevents:

- unlimited peer creation
- config sharing & account reselling
- uncontrolled growth in active connections

Device caps are critical to:

- performance stability
- cost control
- abuse prevention

3) Region Selection From Available Server Pool

Users may select a VPN region from the list of:

- online
- healthy
- capacity-eligible

servers only.

Unavailable nodes must be:

- hidden OR
- marked degraded

No user should be routed into:

- overloaded nodes
- maintenance nodes
- experimental nodes

4) Auto-Assign Node Based on Load Score

Users are NOT assigned to servers randomly.

The system evaluates:

- active peer count
- bandwidth utilization
- latency band
- node capacity threshold

A **load score** is computed per node.

User is assigned to the **best performing node** in that region.

This prevents:

- congestion
- packet loss
- degraded UX
- server collapse under load

Load balancing is the backbone of scalability

5) Generate Peer + Configuration (Secure Lifecycle)

For each registered device:

- a peer is created
- mapped to a region and node
- configuration metadata generated

Security rules:

- private keys generated on device
- only public keys stored
- peer ownership traceable

This ensures:

- zero-knowledge key handling
- legal privacy safety
- full lifecycle governance

No “ghost peers.”
No orphan configs.
No uncontrolled device access.

6) Config Delivery Options

User must be able to retrieve configuration via:

- .conf file download
- copy-paste raw config
- QR code import

Supports:

- desktop clients
- mobile clients
- routers / gateways

Reduces onboarding friction and support load.

7) Instant Device Revocation

User (or admin) must be able to:

- revoke a device instantly
- disable the associated peer
- terminate access immediately

Used for:

- stolen devices
- leaked configurations
- security incidents

Revocation must:

- not rely on cron jobs
- apply at node level
- take effect immediately

No delayed or partial revocation is acceptable.

8) Device Key Rotation

System must support:

- rotating WireGuard keys
- issuing new configuration safely
- invalidating old keys

Used for:

- compromised devices
- periodic hygiene
- lifecycle maintenance

Peer ownership remains intact.

History is retained for audit\

9) Exit IP & Tunnel Status Visibility

Dashboard must display:

- assigned VPN exit IP
- active / inactive tunnel status
- last seen session timestamp
- current region & node indicator

Purpose:

- reduce support dependency
- enable self-diagnostics
- increase transparency

This is essential for user trust & troubleshooting.

10) NEW — Priority Access Token System (Fast-Lane Mode)

A token grants **temporary priority network performance** within a specific region.

Token unlocks:

- preferred routing to lower-load nodes
- relaxed bandwidth constraints
- short-term premium experience

Token characteristics:

- assigned to a user
- scoped to a specific region
- valid for a fixed time window
- expires automatically
- logged & auditable

Use cases:

- beta rewards
- testing premium behaviour
- temporary access upgrades
- controlled performance trials

Token Behaviour Rules (Non-Negotiable)

Tokens DO:

- bias routing toward better nodes
- increase performance weight
- respect capacity protections

Tokens DO NOT:

- bypass device limits
- allow unlimited connections
- override revocation rules
- ignore overload safeguards

Token Expiry Behaviour

When the token expires:

- routing priority is removed
- bandwidth returns to normal
- no disconnect required
- user continues in standard mode

No manual cleanup required.

No lingering privilege.

4) What Is Deliberately Deferred (Not in MVP)

The following capabilities will not be introduced during the MVP phase:

- payments and subscription plans
- freemium or trial tiers
- referral or viral growth mechanisms
- aggressive user acquisition or scaling rollout
- mobile client deployment

These are powerful growth and monetization drivers, but they also place significant stress on the platform. Adding them too early would increase risk and complexity before the core system has proven that it can operate reliably under real-world usage. The MVP must first demonstrate that the platform can operate as a **stable, secure, predictable network system** before higher-impact features are introduced.

Why these features are intentionally delayed

Scaling amplifies mistakes

If architectural or lifecycle flaws exist:

- growth compounds the impact
- issues multiply across users and devices
- failures become harder to contain

A small bug with 10 users is manageable. The same bug with 500 users can:

- crash nodes
- cause mass outages
- damage brand trust

Delaying scaling ensures that weaknesses are discovered and corrected while the impact radius is still small.

Monetization increases load and user expectations

When users begin paying:

- system reliability expectations increase dramatically
- uptime becomes contractual rather than optional
- support and operational workload increases

At the same time:

- user volume typically rises
- session durations increase
- usage intensity becomes heavier

This creates:

- more pressure on servers
- more edge-case scenarios
- higher reputational risk

Monetization should occur only after the network has proven that it can handle sustained operational stability.

Instability destroys trust early

A VPN is a **security and reliability product**.

If early users experience:

- dropped connections
- bad routing behaviour
- revoked peers not applying instantly
- unpredictable device lifecycle behaviour

They will:

- lose confidence in the platform
- avoid recommending it
- be unwilling to pay later

Early stability is more valuable than early revenue or rapid growth.

A strong reputation compounds — a weak one collapses.

What we must prove first

Before scaling or monetizing, the platform must demonstrate:

Reliability

- consistent connection behaviour
- predictable routing outcomes
- stable server performance under normal load

Reliability proves that the system behaves as expected.

Resilience

- ability to handle errors gracefully
- safe revocation and rotation processes
- operational recovery without disruption

Lifecycle correctness

- every device is traceable
- every peer is governed
- revocation and rotation work instantly
- no orphan or ghost devices exist

Lifecycle integrity is critical for:

- security
- operational hygiene
- long-term maintainability

Only when lifecycle behaviour is fully correct can new features safely sit on top of it.

5) Clawn VPN — Project Folder Architecture

```
clawn-vpn/
├── apps/
│   ├── web/                # Frontend web dashboard (Next.js)
│   ├── admin/              # Internal admin console (optional later)
│   └── api/                # Backend API / server functions
│
├── packages/
│   ├── ui/                 # Shared UI components (buttons, cards, layouts)
│   ├── lib/                # Shared utilities & helpers
│   ├── types/              # Shared TypeScript models & contracts
│   └── security/           # Validation rules, token logic, policies
│
├── infra/
│   ├── vpn-nodes/          # WireGuard node config + automation
│   ├── provisioning/       # Scripts to sync peers → nodes
│   ├── monitoring/         # Alerts, metrics & health checks
│   └── terraform/          # (Future) infra-as-code
│
```

```
├── db/
│   ├── schema/             # SQL schema files (tables, indexes, RLS)
│   ├── migrations/         # Versioned migration scripts
│   ├── functions/          # Supabase edge + SQL functions
│   └── seeds/              # Test data & mock fixtures
│
├── docs/
│   ├── architecture/       # Architecture diagrams & specs
│   ├── security/           # Threat model, abuse mitigation, policies
│   ├── product/            # MVP scope, features & acceptance criteria
│   └── operations/         # Runbooks, incident procedures
│
├── tests/
│   ├── api/                # API contract & integration tests
│   ├── db/                 # Schema + RLS tests
│   └── load/               # Node load + performance scenarios
│
└── .env.example            # Environment template
```

Frontend Structure

```
apps/api/
├── routes/
│   ├── auth/              # session & identity
│   ├── devices/           # add / revoke / rotate
│   ├── peers/             # peer provisioning
│   ├── regions/           # region availability + load score
│   ├── configs/           # config generation + delivery
│   ├── tokens/            # priority token issue + validation
│   └── sessions/          # usage metrics + exit IP
│
├── services/
│   ├── peers.service.ts     # lifecycle + ownership rules
│   ├── routing.service.ts   # load scoring + node assignment
│   ├── tokens.service.ts    # token validation policy
│   ├── configs.service.ts   # config assembly logic
│   └── audits.service.ts    # audit log events
│
```

Backend structure

```
apps/web/
├── app/                   # Next.js app router pages
├── components/            # Local UI components
├── features/
│   ├── auth/              # login, session, account UI
│   ├── devices/           # device list, add, revoke, rotate
│   ├── regions/           # region selector + latency view
│   ├── configs/           # config delivery + QR
│   ├── tokens/            # priority access token UX
│   └── sessions/          # exit IP + tunnel status
├── hooks/                 # client hooks (supabase, session)
├── services/              # API client wrappers
├── providers/             # auth + user context providers
├── styles/                # Tailwind / theme
└── utils/                 # formatting, helpers
```

```
├── adapters/
│   ├── supabase/            # db client + RLS-safe access
│   └── vpn-sync/            # server sync + peer push jobs
│
├── workers/
│   ├── cleanup/             # idle peer cleanup
│   └── rotation/            # scheduled key rotation jobs
│
└── middleware/
    ├── auth.guard.ts        # user auth enforcement
    ├── admin.guard.ts       # service-role gate
    └── token.guard.ts       # token-aware routing
```

Infra & VPN Node Automation

```
infra/
├── vpn-nodes/
│   ├── templates/           # base wg configs
│   ├── regions/             # per-region node definitions
│   └── keys/                # server pubkeys (encrypted)
├── provisioning/
│   ├── sync-peers.sh
│   └── revoke-peer.sh
├── monitoring/
│   ├── node-health.sh
│   └── alerts.yaml
```

database schema

This section defines the logical database schema for the Clawn VPN Control Plane. The schema is designed around four core principles:

- strict ownership boundaries
- device-centric lifecycle control
- auditability and operational traceability

- predictable scaling behaviour across regions

The database does not exist merely as storage — it acts as a **policy enforcement layer** for user, device, and peer relationships. Every entity in the system must be traceable back to a single user and must participate in a controlled lifecycle.

Schema Philosophy

The schema is intentionally structured to:

- prevent cross-account resource exposure
- eliminate "orphaned" devices or peers
- guarantee revocation and rotation integrity
- support region-aware routing and capacity planning
- maintain minimal, privacy-aligned session data

Rather than allowing free-form record creation, the schema enforces:

- ownership rules at the data level
- lifecycle constraints at the relationship level
- accountability through audit records

This ensures the control plane remains authoritative and deterministic.

Core Entities & Their Roles

The schema is built around six primary tables, each representing a critical lifecycle domain within the VPN platform.

1) Users

Purpose: Account identity and platform entry point.

Stores:

- unique user identity
- email address
- timestamp of account creation

The **user record is the anchor** from which all other entities — devices, peers, sessions — are derived. Nothing in the system should exist outside of a user context.

This guarantees:

- accountability
- traceability
- clean deletion semantics (no ghost resources)

2) Devices

Purpose: Represents a physical or logical device that connects to Clawn VPN.

Each device:

- belongs to exactly one user
- consumes a limited device slot
- contains a unique public key
- exists in one of two lifecycle states:
    - active
    - revoked

This table enforces:

- user-to-device ownership consistency
- prevention of shared or duplicated devices

- clear revocation lifecycle

Revoked devices remain recorded for:

- historical reference
- security investigation
- lifecycle traceability

No device can be reassigned across accounts.

3) Servers

Purpose: Represents regional VPN exit nodes and routing targets.

Each server record stores:

- region and city metadata
- network address (IP and port)
- public key for WireGuard node identity
- operational status
- capacity score used by routing logic

This table supports:

- node health signaling
- load-aware routing
- controlled regional expansion
- capacity planning analysis

It functions as the **source of truth for the data plane topology**.

4) Peers

Purpose: Logical VPN identity for a user's device on a specific server.

A peer links together three critical relationships:

- the user
- the device
- the assigned server

It also stores:

- the WireGuard public key
- allowed IP ranges
- creation timestamp

This table embodies the **connection lifecycle contract**:

- a peer must belong to the same user as the device
- a peer cannot be created independently of a device
- a peer must always be associated with a server

This prevents:

- cross-account peer assignment
- unauthorized device association
- unmanaged tunnel identities

Peer integrity is central to platform security.

5) Sessions

Purpose: Lightweight usage and activity tracking for each peer.

Tracks for each peer:

- total outbound bytes
- total inbound bytes
- last seen timestamp

Session data is intentionally minimal.

The design deliberately avoids:

- traffic inspection
- URL logging

- packet-level monitoring

The objective is:

- operational visibility
- performance insight
- privacy alignment

If deeper analytics are required in the future, they must be added through **explicit opt-in extensions**, not silent accumulation.

6) Audit Logs

Purpose: System-wide accountability and event tracking.

Each audit entry records:

- who performed an action
    - user
    - admin
    - system process
- what action occurred
- which entity was affected
- contextual metadata
- timestamp of execution

Audit logs are critical for:

- security incident response
- lifecycle traceability
- administrative oversight
- compliance and accountability

No sensitive data is stored — only operational context.

6) Security Model & RLS Enforcement

.

The Clawn VPN control plane is designed with a **least-privilege, ownership-bound security model**. The database is not only a persistence layer — it also acts as an enforcement boundary that prevents unauthorized data exposure, cross-account leakage, and uncontrolled resource creation.

The security posture is built around the following core principles.

User Data Isolation

Users must only be able to access:

- their own devices
- their own peers
- their own session data

No user should have the ability to:

- view another user's devices
- query peers belonging to another account
- access shared or public datasets

All user-facing queries are ownership-scoped.

This prevents:

- accidental cross-tenant exposure
- enumeration of foreign devices
- insight into network usage patterns of others

Zero-Retention Private Key Policy

Private keys must never be stored, logged, or transmitted to the server.

Key security rules:

- private keys are generated on the client device
- only public keys are stored in the database
- key lifecycle actions operate without private key material

This ensures:

- zero-knowledge security posture
- cryptographic privacy integrity
- compliance with best-practice VPN safety standards

Even administrative operators should never have access to user private keys.

If keys are compromised, the mitigation path is **rotation, not retrieval**.

Controlled Administrative Privileges

Administrative functionality is separated from normal user behavior.

Admin-level actions require:

- use of a service-role key
- execution through controlled backend functions

Administrative operations include:

- node assignment
- peer provisioning
- device revocation
- key rotation
- token issuance
- routing overrides

No administrative operation may occur:

- directly through client-side requests
- from user-originated queries
- outside approved control paths

This prevents privilege escalation and operational misuse.

All administrative actions must also be traceable.

Comprehensive Administrative Audit Logging

Every administrative action must create a corresponding audit log entry.

Each log records:

- the actor who performed the action (admin, user, or automated system)
- the target entity affected
- the nature of the operation
- contextual metadata
- timestamp of execution

Audit logs exist to support:

- incident investigation
- abuse detection
- platform governance
- internal accountability

Administrative control without auditability presents unacceptable risk — therefore, auditing is mandatory.

Row-Level Security (RLS) Enforcement Model

RLS is treated as a **core architectural safeguard**, not an optional feature.

It ensures that:

- resource visibility is tied to identity context
- unauthorized relationships cannot be formed
- access violations cannot occur via raw queries

The following enforcement rules apply.

User-Scoped Access Policies

Users may only:

- query peers that belong to their own account
- query devices they own

Effective rule logic:

- SELECT peers WHERE peers.user_id = auth.uid()
- SELECT devices WHERE devices.user_id = auth.uid()

This guarantees that:

- every selection is identity-bound
- data exposure cannot occur through query misuse
- ownership boundaries remain strict

Users cannot read or infer the existence of other users' resources.

Restricted Peer Creation Path

Peers may not be inserted directly through client actions.

Peer creation is only permitted when:

- executed through a validated provisioning function
- evaluated under lifecycle integrity rules
- linked to the correct device → user → server relationships

This prevents:

- rogue peer injection
- unauthorized tunnel generation
- device assignment hijacking

The provisioning layer becomes the enforcement checkpoint.

Default Denial of Cross-User Visibility

The default access posture is:

deny cross-user resource visibility

Any table that contains user-bound data is:

- non-readable across accounts
- non-writable without integrity checks
- restricted by default unless explicitly allowed

This eliminates accidental exposure through:

- joins
- misconfigured queries

- future feature additions

The system must fail closed, not fail open.

Security Risk Warning

If RLS is bypassed, relaxed, or partially applied:

- confidentiality risk increases at scale
- users may gain visibility into unrelated devices or peers
- cross-tenant inference becomes possible
- incident blast radius grows with user volume

7) Server Infrastructure — MVP Deployment Baseline

Device & Peer Lifecycle Flow (Correct Model)
Device onboarding defines how a new device is registered, provisioned, and authorized to establish a VPN connection within the Clawn VPN control plane. The process is intentionally structured, auditable, and ownership-bound to prevent uncontrolled peer creation, credential duplication, and cross-account misuse.

The onboarding flow ensures that every device:

- belongs to a single user
- is uniquely identifiable
- has a verifiable lifecycle state
- can be revoked or rotated instantly
- cannot persist beyond its authorized scope

This lifecycle discipline is critical to platform integrity and long-term operational safety.

Device Onboarding Workflow (End-to-End Sequence)

The onboarding sequence follows the steps below.

1. **User logs into their account**
   The user must be authenticated before interacting with device or peer management functions.
   All lifecycle actions are tied to a verified user identity.

2. **User creates a new device entry and assigns a label**
   The user adds a device with a descriptive name (e.g., “Laptop”, “Work PC”, “Tablet”).
   This device now consumes one of the user’s available device slots.

3. **The private key is generated locally in the browser**
   Cryptographic key generation occurs entirely on the client side.
   The private key is never stored, transmitted, or exposed to the control plane.

4. **Only the public key is sent to the backend**
   The database stores the public key and associates it with:
   - the user
   - the device
   - future peer records

This enforces a **zero-retention private key policy**.

5. **The control plane assigns a region and node**
   Based on:
   - region selection
   - load score
   - node availability
   - capacity limits

The system determines the appropriate gateway node for the device.

6. **A peer entry is created in the database**
   The peer binds together:
   - user → device → server node

Peer creation becomes the canonical connection identity.

7. **A configuration file is generated and becomes downloadable**
   The user can retrieve configuration via:
   - direct file download
   - copy-paste view
   - QR import

The configuration references the assigned peer identity and routing metadata.

At this point, the device becomes:

- identifiable
- governed
- revocable
- auditable

and formally part of the network lifecycle.

Required Device Management Actions (User Dashboard)

The device dashboard must support the following lifecycle management operations.

Revoke device

Immediately disables:

- the device entry
- associated peer identity
- active tunnel access

Used for:

- lost devices
- leaked configurations
- unauthorized access risk

Revocation must be:

- immediate
- deterministic
- irreversible without explicit re-provisioning

Rotate device keys

Generates a new key pair and issues a new configuration.

Used for:

- key hygiene
- suspected compromise
- periodic rotation policy

Old keys become invalid at the node level.

Rotation preserves:

- device ownership

- lifecycle continuity
- audit history

Reassign region

Allows migration of a device peer to another eligible node or region.

Reassignment must:

- respect load balancing rules
- prevent abusive relocation frequency
- ensure stale mappings are cleaned up

This supports:

- latency optimization
- performance preference
- regional failover

View exit IP & tunnel status

The dashboard must display:

- assigned exit IP
- current connection state
- last-seen session timestamp

This enables:

- self-service diagnostics
- clarity of tunnel behavior
- reduced support dependency

Revocation Policy — Immediate & Final

Device revocation is treated as a **security-critical operation**.

Once a device is revoked:

- **its peer must be rejected immediately at the node level**
- **existing active sessions must be invalidated**
- **configuration reuse must not be possible**
- **no delayed or "soft revoke" state is permitted**

There must be no condition under which a revoked device:

- **reconnects successfully**
- **remains partially active**
- **continues operating under stale mapping**

Revocation is final unless the device is explicitly re-provisioned through the onboarding process again.

This reinforces:

- **lifecycle discipline**
- **security integrity**
- **operational trust**

8) Monitoring, Failover & Reliability Controls

The Clawn VPN platform must demonstrate operational resilience before user growth is allowed. Stability is treated as a prerequisite for scaling. Monitoring, failure detection, automated cleanup, and controlled recovery mechanisms are required to ensure that the system continues to operate predictably under stress, load variation, and fault conditions.

The objective of this reliability framework is to:

- detect failure conditions early
- prevent overload cascades
- maintain node isolation
- preserve predictable user experience
- ensure recoverability without disruption

Growth will only occur after the platform proves its capability to withstand sustained operational pressure.

Operational Monitoring Requirements

The platform must implement monitoring across multiple dimensions of infrastructure and lifecycle behaviour. Monitoring is not limited to uptime; it must reflect **network health, capacity stress, and anomalous usage patterns**.

Required monitoring domains include:

Node availability and health

Each regional node must continuously report:

- online / offline status
- resource state (CPU, memory, load)
- latency band and routing stability

If a node enters a degraded or unstable state:

- it must be excluded from routing selection
- connected peers may be migrated or fail closed
- administrative review must be triggered

Node availability monitoring prevents users from being routed into unstable gateways.

Peer cleanup and idle pruning

Stale, abandoned, or long-idle peers must be:

- identified
- flagged
- removed or rotated where appropriate

This prevents:

- key accumulation over time
- inflated peer counts
- unnecessary capacity reservation

Lifecycle cleanup also strengthens:

- security posture
- routing efficiency
- operational hygiene

Suspicious activity detection

Behavioural signals must be monitored for potential abuse patterns, including:

- unusual connection frequency
- repeated cross-region switching
- excessive bandwidth bursts
- abnormal session churn

Suspicious patterns must trigger:

- soft restriction
- quarantine review
- admin visibility

The purpose is not user surveillance, but **platform protection and abuse mitigation**.

Abuse quarantine controls

When behaviour indicates potential misuse:

- the affected peer may be rate-limited
- token or priority access may be suspended
- routing privileges may be temporarily restricted

Quarantine serves as a safety buffer between:

- legitimate heavy usage
- intentional abuse or automated exploitation

The goal is to preserve network stability without immediately removing the user unless required.

Alert Thresholds & Trigger Conditions

Alerts must be tied to meaningful performance and stability boundaries rather than raw metrics. The following thresholds are treated as early indicators of stress conditions.

Alerts must trigger when:

- CPU usage exceeds **75% sustained**
- packet loss exceeds **2%**
- latency increases to **2× baseline**
- active peers exceed defined capacity limits

These thresholds signal that:

- the node is approaching saturation
- user experience is at risk
- routing fairness may degrade

Alerts are intended to prompt:

- pre-emptive capacity evaluation
- load redistribution
- performance assessment

Not reactive firefighting after failure.

Autoscaling Policy — Capacity Expansion Criteria

Capacity expansion is not based on short traffic spikes, but on **sustained operational load**.

A new node should be added when:

load exceeds 70% on a sustained basis measured at the **95th percentile traffic window**

This ensures that:

- scale decisions are data-driven
- capacity buffers are preserved
- growth does not destabilize the network

The guiding principle is:

- do not scale reactively
- do not over-extend prematurely
- expand when operational evidence requires it

Scaling should improve stability — not introduce new fragility.

Resilience Before Growth

Before user growth is intentionally increased, the platform must demonstrate:

- sustained uptime under real usage
- predictable routing performance
- controlled peer lifecycle behaviour

- safe failover and remediation handling
- visibility into operational health

Capacity Risk — Excessive User Load on a Single Node

Risk Description
A disproportionate number of active peers connecting to the same node can cause:

- resource saturation
- degraded performance and latency
- session drops and connection instability
- increased failure propagation risk

This also creates a **single-point stress hotspot** in the network.

Mitigation Controls

- **Peer allocation caps per node** to prevent overload conditions
- **Dynamic autoscaling rules** to spin up additional nodes on demand
- **Load-aware connection routing** to distribute peers intelligently
- **Health-triggered failover** to protect against cascading degradation

This ensures traffic distribution remains balanced under growth or spike conditions.

Security Risk — Key Compromise or Credential Leakage

Risk Description
Exposure of platform keys or credentials could enable:

- unauthorized peer access
- malicious session creation
- traffic hijacking or impersonation
- regulatory / trust breach scenarios

Mitigation Controls

- **Instant key revocation pipeline** with zero-delay kill switch
- **Automated key rotation policy** across all dependent services
- **Signed fallback migration tokens** for uninterrupted peer continuity
- **Scope-limited credentials** to minimize blast radius

This approach prioritizes **containment speed + continuity of service**.

3) Trust & Abuse Risk — Fraudulent or Malicious Traffic

Risk Description
Bad-actor usage patterns may include:

- automated abuse traffic
- country-based fraud rings
- spam / scraping / tunneling misuse
- service masking or evasion behavior

Unchecked, this can affect platform reputation and partner networks.

Mitigation Controls

- **Per-country adaptive throttling policies**
- **Behavior-based peer quarantine pipeline**
- **Silent rate-limit enforcement** instead of hard blocks (reduces adversarial adaptation)
- **Anomaly-score monitoring** for escalation to manual review

This allows us to suppress malicious activity while **minimizing disruption to legitimate users**.

Operational Risk — Support Load & Incident Complexity

Risk Description
As platform scale increases, support complexity may spike due to:

- unclear root-cause visibility
- noisy user-reported symptoms
- dependencies across nodes / routes / sessions

This increases mean-time-to-resolution (MTTR).

Mitigation Controls

Deployment of a **diagnostics & observability dashboard** exposing:

- exit IP + route trace
- latency and RTT trends
- node health state
- packet loss and throughput stats
- recent failover & scaling events

9) Mobile Migration Strategy (Future Phase)

The mobile client will be introduced as a future expansion layer without altering or weakening the existing backend control plane. The guiding principle is that the **backend remains the single source of truth** for identity, device lifecycle, access control, and security enforcement.

Mobile Client Responsibilities (Local Only)

The mobile application will perform strictly client-side operations:

- **Authenticate** the user against the existing identity layer
- **Fetch the assigned peer** from the backend authority
- **Retrieve configuration + policy artifacts** only
- **Initialize and run the tunnel locally** on device

Backend Authority Guarantees

The backend continues to own all platform-critical decisions:

- **No peer creation from the mobile client**
  (all provisioning remains server-side)
- **No bypass of device limits, peer caps, or licensing rules**
  (mobile must conform to the same constraints as desktop clients)
- **No direct key lifecycle manipulation from client devices**
  (revocation, rotation, and issue workflows remain centrally enforced)

Security, Consistency, and Scale Outcomes

This design ensures:

- **Security is preserved** — keys, peers, and entitlements are never client-controlled
- **Consistency is guaranteed** — every device observes the same lifecycle rules
- **Scalability remains predictable** — capacity planning happens centrally, not per-device

Mobile becomes an **interaction surface**, not a governance layer.

10) MVP Completion Criteria

The MVP is considered complete only when platform stability, lifecycle correctness, and

operational readiness have been proven in a constrained but realistic environment.

MVP Readiness Conditions

The MVP must demonstrate:

- **Reliable operation across two regions** with stable routing and failover behavior
- **Fully functional device lifecycle** — register, update, revoke, recover
- **Revocation and key rotation validated** under live usage
- **Consistent and deterministic configuration delivery** to all peers
- **Monitoring, diagnostics, and alert pipelines active**
- **A controlled closed-beta cohort (10–25 users)** onboarded and operating successfully

The definition of done is **operational confidence**, not feature completeness.

failure at scale.

Post-MVP Expansion Sequence

Only after the above criteria are satisfied:

- Expand footprint to **five fully supported regions**
- Introduce **pricing + billing flows**
- Launch the **mobile client release**
- Begin controlled external rollout

Growth follows **proven stability**, not optimism.

Execution Principle

Delaying expansion until MVP discipline is met:

- prevents cascading outages at scale
- reduces rollback risk and customer impact
- ensures operational maturity keeps pace with adoption