1. Risks of Community Datacenters

Even if data centers are community run and all the apps running on the servers are open source and E2EE then there are still risks associated for its users. In this chapter we will discuss these risks.

- 1. Datacenters vs Self-Hosting
- 2. How to minimize the community datacenter risks ?
- 3. Mass Surveillance

1. Datacenters vs Self-Hosting

This is a thoughtful question about security and privacy trade-offs. Let me break down the remaining risks users would face in your scenario compared to complete self-hosting.

Key Remaining Risks in Community Datacenters

Risk Category	Community Datacenter	Self-Hosted
Physical infrastructure access	Datacenter staff or attackers with physical access could compromise hardware, extract data, or plant backdoors	Only the user controls physical access to their own hardware
Operational security	Depends on datacenter's competence in patching, backups, disaster recovery, and incident response	User bears full responsibility but has complete control
Credential management	Datacenter must secure admin credentials; potential for insider threats or compromised credentials	User controls all credentials; insider threat is eliminated
Network-level attacks	Shared infrastructure means potential for lateral movement between tenants' encrypted traffic or infrastructure compromises	Isolated network with only user's devices; harder to target
Supply chain vulnerabilities	Datacenter's hardware/software suppliers could be compromised (affects many users at once)	User's supply chain is smaller and more controllable
Audit and verification	Users must trust datacenter's security claims; open source helps but doesn't guarantee honest operation	User can directly verify their system's state and logs
Regulatory/legal risks	Datacenter could face legal orders, seizures, or regulatory changes affecting service availability	User's self-hosted data generally has stronger legal protections in their own jurisdiction

The Core Issues with Community Datacenters

Trust in operators and infrastructure: Even with open-source code and e2ee, users must trust that the datacenter is actually running the code they see, not a modified version. This is the "software supply chain" problem—you can verify the code is secure, but you can't verify it's actually being executed as written without running it yourself.

Insider threats: Datacenter employees with system access could potentially:

- Monitor unencrypted metadata (IP addresses, connection patterns, timing information)
- Access encryption keys during key rotation or backup operations
- Exploit zero-days before patches are deployed
- Manipulate logs or audit trails

Physical security: A compromised datacenter is a single point of failure for many users. An attacker with physical access could install persistent backdoors on hardware, clone hard drives, or extract encryption keys from memory.

Operational security complexity: Running secure infrastructure at scale is harder than securing a single user's machine. More complexity = more potential failure points. The datacenter must handle:

- Secure key management across many services
- Safe secret rotation and distribution
- Proper isolation between applications
- Secure backups without leaking data

Metadata leakage: E2ee protects message content, but datacenters still see:

- When users access services (timing attacks)
- Which users communicate with which users
- Data volume and traffic patterns
- IP addresses and geolocation
- Account activity patterns

A sophisticated attacker can often extract sensitive information from metadata alone.

Zero-day exposure: A single vulnerability discovered in a shared component affects all users simultaneously. Self-hosted users are more isolated unless they use the exact same software version.

The Practical Tradeoff

The community datacenter model trades **user security control** for **operational convenience**. Self-hosting gives you:

- Complete verification of what's running
- Isolated attack surface (attacker can't easily pivot to other users)
- Control over all physical security
- No insider threat risk
- Stronger legal protections in your jurisdiction

But requires you to:

- Maintain systems and apply patches regularly
- Handle backups and disaster recovery
- Manage your own infrastructure costs
- Possess technical expertise or hire it

Your scenario where apps are **open source and e2ee** significantly reduces risks compared to proprietary, encrypted cloud services. The real question isn't whether any risks remain (they do), but whether the convenience of managed infrastructure is worth accepting operational risks you can't directly control.

2. How to minimize the community datacenter risks ?

Here are practical strategies to significantly reduce the remaining risks in your community datacenter model:

Architectural & Operational Solutions

Strategy	How It Reduces Risk	Trade-offs
Decentralized node operators	Spread datacenter across many independent operators; no single entity controls all infrastructure	Requires coordination, harder to manage consistency
Hardware security modules (HSMs)	Encryption keys stored in tamper-resistant hardware; even datacenter staff can't access them	Adds cost and complexity; potential performance impact
Threshold cryptography	Split encryption keys among multiple independent parties; no single operator has complete key	Complex coordination needed; slower operations
Attestation & proof systems	Use trusted execution environments (TEEs) or zero-knowledge proofs to verify datacenter is running correct code	Can be circumvented if TEE is compromised; adds overhead
Distributed backups	Replicate encrypted data across geographically dispersed datacenters run by different operators	Increases complexity and latency; coordination overhead
Immutable audit logs	Append-only, cryptographically signed logs stored separately; impossible to cover up suspicious activity	Doesn't prevent attacks, only detects them after the fact
User-controlled key derivation	Users derive their own encryption keys locally; datacenter never sees master keys	More complex UX; key recovery becomes harder

Strategy	How It Reduces Risk	Trade-offs
Red team audits	Regular independent security audits with adversarial testing of datacenter operations	Expensive; doesn't catch everything; only periodic

---

Technical Controls to Implement

Eliminate metadata exposure:

- **Onion routing** (like Tor): Route all traffic through multiple independent relays so datacenter never sees user's IP
- **Metadata-resistant protocols**: Use systems that hide communication patterns (Signal's Sealed Sender, Briar's approach)
- **Noise padding**: Add fake traffic to obscure real activity patterns and timing
- **Private information retrieval (PIR)**: Users retrieve data without revealing *which* data they accessed

Verify code execution:

- **Remote attestation**: Datacenter cryptographically proves it's running unmodified code on trusted hardware (Intel SGX, ARM TrustZone, or open alternatives like Keystone)
- **Reproducible builds**: Anyone can independently compile the source code and verify the binary matches what's deployed
- **Frequent code audits**: Independent security researchers audit and sign off on code before deployment
- **User-side verification**: Random users periodically download and verify datacenter's running code matches published sources

Reduce insider threats:

- **No single person holds credentials**: Require multiple datacenter staff to authenticate for sensitive operations (multi-signature scheme)
- **Microservice isolation**: Each app runs in isolated containers; compromise of one doesn't affect others
- **Role-based access control with logging**: Every admin action is logged and attributed; creates accountability
- **Regular staff rotation**: Reduce time any individual has access to critical systems
- **Employee vetting & compartmentalization**: Sensitive roles have minimal information about each other's operations

---

Governance & Accountability

Make operators transparent:

- **Public infrastructure reports**: Regular published reports on security incidents, patches applied, hardware replaced
- **Community oversight board**: Users elect representatives to audit operations and approve major changes
- **Bug bounty programs**: Reward researchers for finding vulnerabilities
- **Regulatory compliance**: Subject to independent audits (SOC 2, ISO 27001 equivalent)

Legal & jurisdictional safeguards:

- **Distributed legal entities**: Run operations in multiple jurisdictions so no single government can seize everything
- **Data minimization by design**: Don't collect metadata that could be demanded by legal orders
- **Encrypted-by-default policies**: Make it technically impossible to comply with orders to decrypt user data
- **User bill of rights**: Legal commitment to specific security practices; users can sue if violated

---

Hybrid Approaches

Federated model: Instead of one community datacenter, users can choose from many independent operators. If you don't trust one, switch to another. Standardized protocols mean apps aren't locked into a single datacenter.

Trusted Hardware Enclaves: Deploy applications in secure enclaves (SGX, SEV) where even datacenter operators can't read running memory. Users get operational convenience without trusting the datacenter.

Client-side caching: Users keep frequently-accessed data locally. Datacenter is only contacted occasionally, reducing exposure and metadata leakage.

Hybrid self-hosted + community: Critical apps (email, messaging) self-hosted; convenience apps (notes, file sync) use community datacenters. Limits damage from any single compromise.

The Reality

The most effective approach combines several of these:

1. **Decentralize operators** so no single entity is the target
2. **Use hardware attestation** to verify code execution
3. **Implement threshold cryptography** for keys
4. **Add metadata-resistant protocols** (onion routing, PIR)
5. **Create accountability** through audits and oversight boards
6. **Use reproducible builds** for user verification

This still doesn't match self-hosting's security posture—you're trading some control for convenience—but it dramatically reduces the gap. The remaining risks are mostly about trusting operators' competence and legal jurisdiction, not fundamental security flaws.

The key insight: **you can't eliminate all risk in a managed system, but you can distribute it across multiple independent operators and make attacks detectable and expensive.**

3. Mass Surveillance

I don't have specific recent statements from Edward Snowden directly comparing mass surveillance in China to happiness or quality of life there. Let me search for what he's actually said about surveillance states and China specifically. Based on my search, I can now provide you with a more informed answer. While Snowden hasn't published a detailed critique specifically addressing your point about "people living happily" in China despite surveillance, his views on mass surveillance states come through clearly in his public statements.

Snowden's Concern: Control, Not Comfort

Snowden's warning about mass surveillance states like China centers on political control and loss of freedom, not immediate suffering. In 2019, he described China's surveillance apparatus as **"utterly mind-boggling,"** referring to their system of over 700 million cameras, facial recognition, artificial intelligence monitoring, and the social credit system.

The key issue isn't whether people *feel* happy—it's about what mass surveillance enables governments to do with unprecedented power. Snowden's core argument is this: **"Do we want to live in a society where we live totally naked in front of government, and they are totally opaque to us?"**

How Mass Surveillance Enables Authoritarianism

China exemplifies exactly what Snowden warns against. Consider what the surveillance infrastructure is *actually used for*:

Social Control Through Arbitrariness. Journalist Liu Hu in China was prevented from buying plane tickets and credits based on court decisions he couldn't contest. **He later told the BBC: "There have been occasions when I have met some friends and soon after someone from the government contacts me. They warned me, 'Don't see that person, don't do this and that'. With artificial intelligence we have nowhere to hide."**

Targeting of Specific Groups. China deploys surveillance against Uyghurs in Xinjiang using what's called "racial AI"—cameras programmed to identify people by ethnicity. During a nine-month period, the state conducted 11 million phone searches on nearly half of Urumqi's population. The result: **millions of Uyghurs were monitored, then imprisoned in work camps on fabricated grounds.**

Why Happiness Now Doesn't Guarantee Freedom Later

Snowden's argument is essentially about power imbalance and future risk. People in China may feel relatively safe and content now, but that doesn't address the fundamental problem: **once a surveillance infrastructure exists, it can be turned against anyone at any moment.** A journalist, dissident, ethnic minority, or religious group can suddenly become a target—not because they did anything wrong, but because those in power decide they're inconvenient.

His famous framing of the dilemma applies here: **"If you create an irresistible power, whether it's held by Facebook or whether it's held by any government, the question is: how will you police the expression of that power when it is used against the public rather than for it?"** The infrastructure itself becomes the weapon.

Snowden emphasizes that **"mass surveillance equals control,"** and this control doesn't announce itself loudly—it operates through quiet pressure, arbitrary punishment, and the chilling effect of knowing you're always watched. People may adapt and seem content, but they've lost the ability to dissent, organize, or think freely without fear.