Lab 11 Submission

Task 1: Local IPFS Node Setup and File Publishing

IPFS Node Configuration

Docker Setup Command:

docker run -d --name ipfs_node \ -v ipfs_staging:/export \ -v ipfs_data:/data/ipfs \ -p 4001:4001 -p 8080:8080 -p 5001:5001 \ ipfs/kubo:latest

Node Status:

- Container ID: 4f58a35e7b75
- Image: ipfs/kubo:latest
- Status: Up and healthy
- Exposed Ports:
  - 4001: P2P communication with other IPFS nodes
  - 5001: API endpoint for IPFS commands
  - 8080: HTTP gateway for accessing IPFS content
- Volumes:
  - ipfs_staging:/export - For staging files to add to IPFS
  - ipfs_data:/data/ipfs - For persistent IPFS node data

Network Status

Connected Peers: 55 active IPFS peers

Network Connectivity: Node successfully bootstrapped and connected to the IPFS network
Sample Connected Peers:

/ip4/107.170.200.4/tcp/4001/p2p/12D3KooWFCxTFRdTAGV8m5iX7MVnEyGTp1Fp9LDnQem9FiNAtaDG
/ip4/142.93.22.139/tcp/4001/p2p/12D3KooWRAqnxYYn9SkAZTQb1GP2pgrWauxVkVsfAMcfKFDgJobF
/ip4/144.202.22.34/tcp/4001/p2p/12D3KooWDhersJaGGDUS7E1y8LDC7cofThiMhX9G5oBU7L1Ts5hH /ip4/142.171.168.32/udp/4001/quic-v1/p2p/12D3KooWCftSrYcGQjeJYPfZVHYsNtf3XuwrDxUL5Xd9NUJPMeks
/ip4/45.32.179.61/tcp/4001/p2p/12D3KooWAr5UTq8St5eSUgiTQzxJjWunk3EHmNunUzZHMoyHLCL

Network Protocols in Use:

- TCP (standard TCP connections)
- UDP with QUIC-v1 (modern, faster protocol)
- Multiple geographic locations (US, Europe, Asia)

File Publishing

Test File Details:

- Content: Hello IPFS Lab
- File Size: 15 bytes
- Location in Container: /export/testfile.txt
  Publishing Process:

1. Create test file on Windows echo "Hello IPFS Lab" > testfile.txt #
2. Copy file to IPFS container staging volume docker cp testfile.txt ipfs_node:/export/ #
3. Add file to IPFS docker exec ipfs_node ipfs add //export/testfile.txt

IPFS CID (Content Identifier):

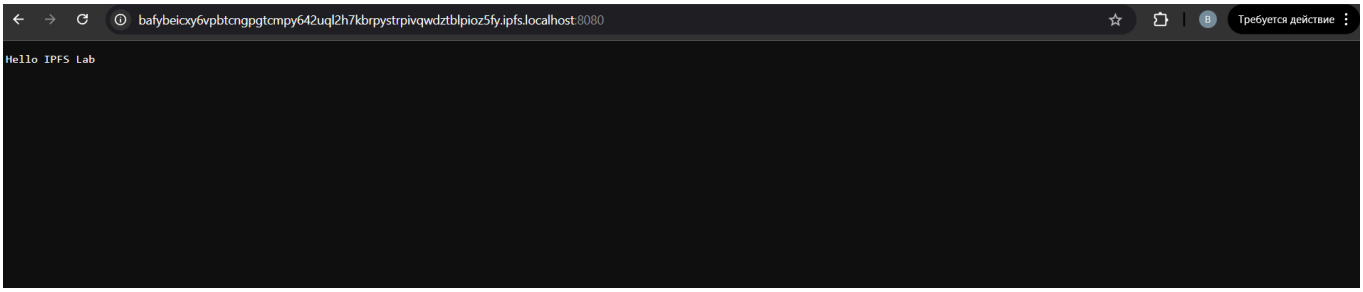
`QmUFJmQRosK4Amzcjwbip8kV3gkJ8jqCURjCNxuv3bWYS1`

Command output:

```
15 B / 15 B 100.00%added QmUFJmQRosK4Amzcjwbip8kV3gkJ8jqCURjCNxuv3bWYS1 testfile.txt
```

Gateway Access Verification

URL: http://localhost:8080/ipfs/QmUFJmQRosK4Amzcjwbip8kV3gkJ8jqCURjCNxuv3bWYS1
Response: Browser displays "Hello IPFS Lab" content

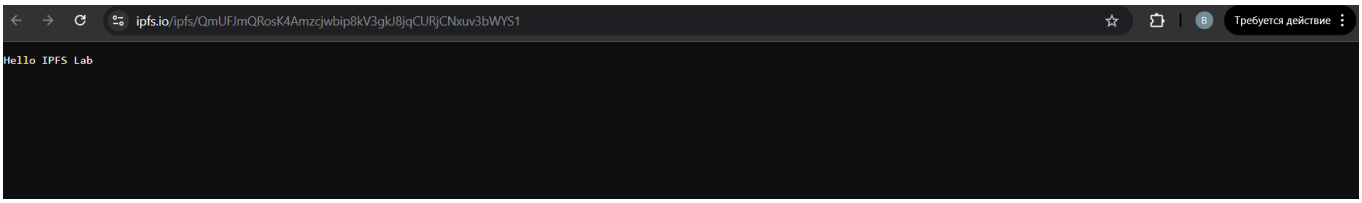


Browser showing "Hello IPFS Lab" accessed via local gateway
Notes:

- Local gateway provides immediate access to content
- No propagation delay
- Relies on local IPFS node running

Public Gateway 1 — ipfs.io

URL: https://ipfs.io/ipfs/QmUFJmQRosK4Amzcjwbip8kV3gkJ8jqCURjCNxuv3bWYS1
Propagation Time: Approximately 2-3 minutes

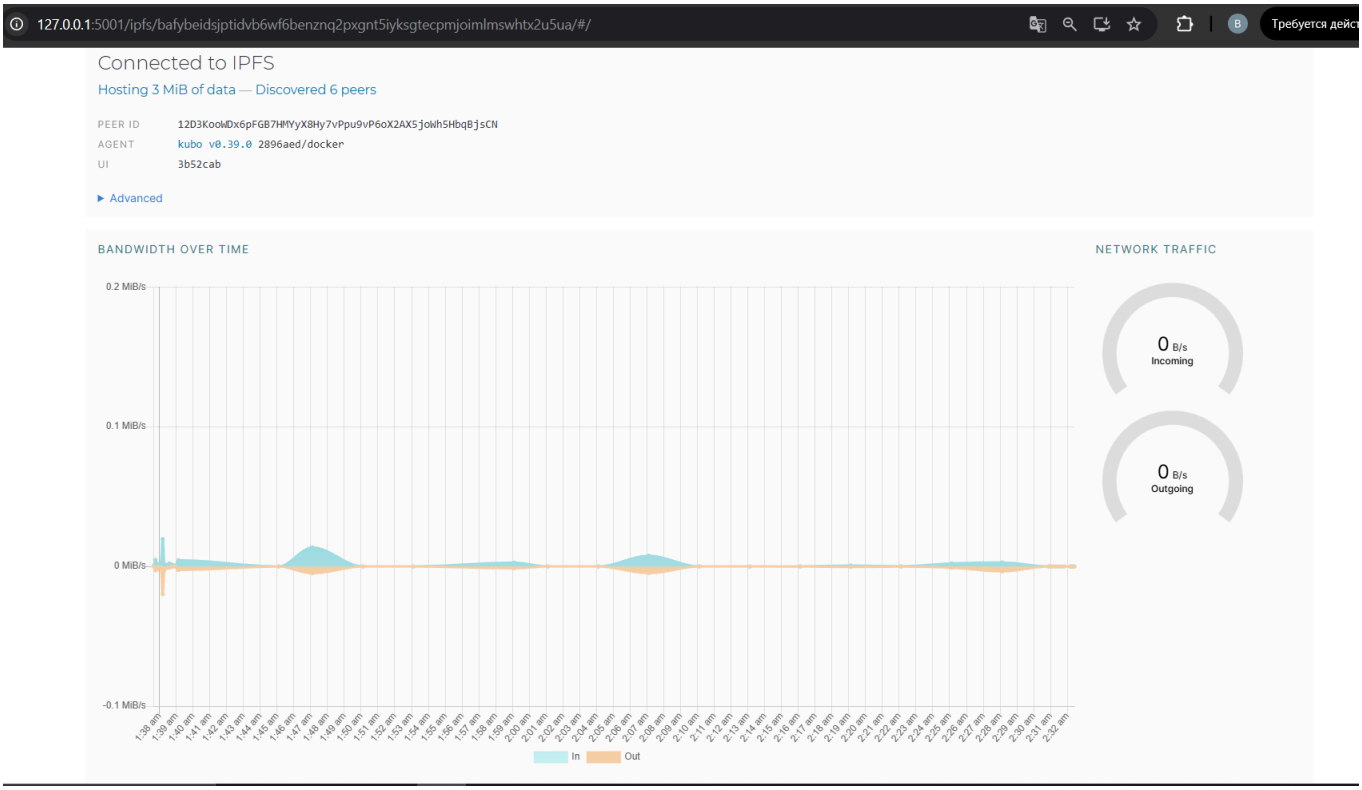


Browser showing content accessed via ipfs.io gateway
Notes:

- Cloudflare-operated gateway
- Global CDN with regional caching
- Most widely used public IPFS gateway

IPFS Web UI Status

URL: http://127.0.0.1:5001/webui/



IPFS Web UI dashboard showing node information and connected peers

---

Task 2: Static Site Deployment with 4EVERLAND

4EVERLAND Project Configuration

Platform Settings:

- Hosting Platform: IPFS/Filecoin
- Repository: https://github.com/NoNesmer/F25-DevOps-Intro
- Branch: main
- Framework: Other (static site)
- Output Directory: ./ (root directory)

Deployment Information

4EVERLAND Project URL:
https://f25-devops-intro-llguvaur-nonesmer.ipfs.4everland.app/

IPFS CID (from 4EVERLAND Dashboard):
bafybeiecykdsvmmnhye4vzt2rk6eo7hhts4aoonprhruu7yjjv4ogj5k44

IPFS Protocol URL:
ipfs://bafybeiecykdsvmmnhye4vzt2rk6eo7hhts4aoonprhruu7yjjv4ogj5k44

Deployment Time: Approximately 2-3 minutes from GitHub connection to IPFS deployment

Site Verification

Via 4EVERLAND Subdomain

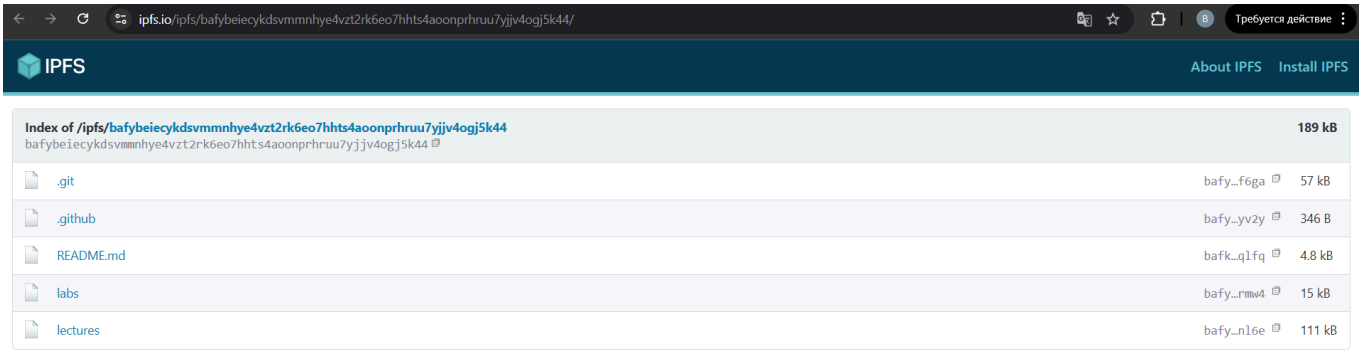
URL: https://f25-devops-intro-llguvaur-nonesmer.ipfs.4everland.app/

Browser showing deployed site accessed via 4EVERLAND subdomain
Features:

- Easy-to-remember subdomain URL
- Served through 4EVERLAND's infrastructure
- Automatic content pinning ensures availability
- IPFS hash embedded in 4EVERLAND's CDN
- Continuous deployment: Updates trigger automatically on GitHub pushes

Via Public IPFS Gateway

URL: https://ipfs.io/ipfs/bafybeiecykdsvmmnhye4vzt2rk6eo7hhts4aoonprhruu7yjjv4ogj5k44

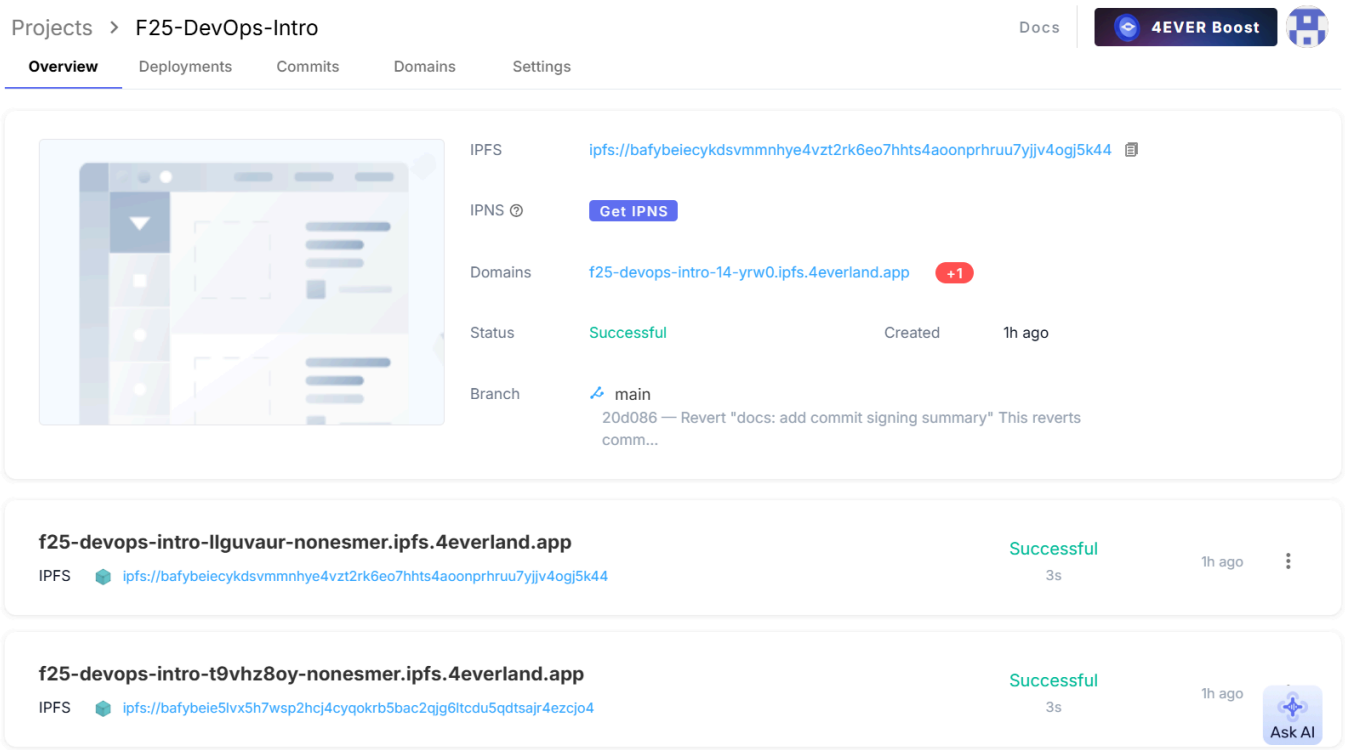


Browser showing deployed site accessed via IPFS gateway

Notes:

- Same content accessible through any public IPFS gateway
- Content addressable by IPFS CID, not location
- Alternative gateways: cloudflare-ipfs.com, gateway.pinata.cloud, etc.
- Demonstrates true decentralization: multiple gateways, same content

4EVERLAND Dashboard



4EVERLAND deployment dashboard showing deployment status and IPFS CID

Analysis: IPFS vs. Traditional Web Hosting

Q1: How does IPFS's content addressing differ from traditional URLs?

Traditional Web (Location-Based Addressing):

- Content identified by where it is stored (server location)
- URL format: https://example.com/path/to/file
- Dependencies: Server location, domain, path structure
- Fragility: Content becomes unavailable if server goes down
- Change management: Moving content to new server breaks old URLs

IPFS (Content-Based Addressing):

- Content identified by what it is (cryptographic hash)
- CID format: QmXxxx... (content hash) or bafyxxx... (newer format)
- Independence: Same file produces same CID regardless of storage location
- Resilience: Content remains accessible from any node storing it
- Immutability: CID changes only if content changes, ensuring authenticity
- Content Addressing: ipfs://Qm... or gateway: https://ipfs.io/ipfs/Qm...

Key Differences:

Aspect	Traditional URLs	IPFS CIDs
Addressing	Location-based (where)	Content-based (what)
Mutability	URL stable, content may change	CID stable iff content stable
Server Dependency	Single origin server	Any node with content
Censorship	Can be removed by server owner	Censorship-resistant
Verification	Trust server authenticity	Cryptographically verified
Availability	One failure point	Multi-node resilience

Q2: What are the advantages of decentralized storage?

1. Censorship Resistance

- No single authority can remove content
- Content persists as long as at least one node stores it

- Governments and organizations cannot easily suppress information
- Example: Decentralized publishing of sensitive documents

2. No Single Point of Failure

- Traditional web: Server down means content inaccessible
- IPFS: Content available from any node that has it
- Network becomes more resilient with more participants
- Example: News site on IPFS remains accessible during DDoS attacks

3. Content Permanence

- Pinned content persists indefinitely
- No concern about domain expiration or server shutdown
- Historical content remains accessible
- Example: Archive websites, legal records, historical documents

4. Bandwidth Efficiency

- Content served from nearest nodes (local optimization)
- Reduces global bandwidth usage
- Faster content delivery through geographic distribution
- Example: Large files retrieved from local peers instead of distant servers

5. User Ownership and Control

- Users run own nodes, control their data
- No reliance on cloud providers or hosting companies
- Personal data remains under user control
- Example: Personal websites, decentralized applications

6. Cost Reduction

- No centralized infrastructure costs
- Community-contributed bandwidth
- Services like 4EVERLAND offer affordable pinning
- Example: Hosting can be free or minimal cost versus traditional servers

7. Content Integrity

- Cryptographic hashing ensures content authenticity
- Detection of tampering through CID mismatch

- No server-side modification of content
- Example: Ensure downloaded files are not corrupted

Q3: What are the disadvantages of decentralized storage?

1. Propagation Delays

- New content takes time to spread through network (2-5 minutes typical)
- Not suitable for real-time updates
- Public gateways may have variable latency
- Workaround: Pre-pinning content, using dedicated gateways

2. Gateway Dependence

- Users often rely on public gateways instead of running nodes
- Public gateways can be bottlenecks or single points of failure
- Gateway availability affects user experience
- Workaround: Run personal IPFS node for guaranteed access

3. Pinning Requirements

- Content must be actively pinned to persist
- Unpinned content subject to garbage collection
- Requires understanding of pinning services and costs
- Workaround: Use pinning services like 4EVERLAND, Pinata

4. Limited Maturity

- Fewer tools and standards compared to traditional web
- Less existing infrastructure and ecosystem
- Steep learning curve for new developers
- Example: Limited analytics, monitoring, debugging tools

5. Network Complexity

- Requires understanding of peer-to-peer networking concepts
- Distributed Hash Table (DHT) can be difficult to debug
- Configuration and optimization more complex
- Example: Port forwarding, firewall configuration, peer discovery

6. Performance Variability

- Speed depends on network conditions and node distribution
- No guaranteed uptime Service Level Agreements like traditional hosting
- Large file downloads may be slower than centralized CDN
- Workaround: Use pinning services with SLA guarantees

7. Dynamic Content Limitations

- IPFS excels at static content
- Dynamic applications (databases, real-time updates) require additional layers
- Cannot directly serve server-side code
- Workaround: Use hybrid approach (serverless functions + IPFS)

8. Privacy Concerns

- Content on IPFS is globally accessible once published
- No privacy control like traditional hosting can provide
- Content cannot be deleted (only unpinned)
- Workaround: Encrypt sensitive data before publishing

Trade-offs: Traditional Web Hosting vs. IPFS Hosting

Traditional Web vs IPFS Hosting Comparison:

Criterion	Traditional Web	IPFS Hosting
Speed	Fast with CDN optimization	Varies by gateway and location
Uptime Guarantee	99.9%+ SLA available	Community-dependent
Scalability	Well-established	Peer-contributed resources
Cost	Monthly fees (5-100+ dollars)	Often free or minimal
Censorship Resistance	Subject to takedowns	Highly resistant
Single Point of Failure	Yes (central server)	No (distributed network)
Technical Complexity	Simple to deploy	Moderate to complex
Maturity	Highly mature ecosystem	Emerging technology
Analytics Tools	Rich feature set	Limited options
User Control	Provider-dependent	Full user control
Dynamic Content	Native support	Requires workarounds

Best Use Cases for Traditional Web Hosting:

- Mission-critical applications requiring guaranteed uptime
- Applications needing advanced analytics and monitoring
- Content requiring guaranteed speed and performance
- Large-scale operations with stable revenue
- Applications needing privacy controls and access restrictions
- Dynamic content with frequent updates and real-time data
- Enterprise applications with Service Level Agreement requirements

Examples: Banking, SaaS platforms, e-commerce, social media, real-time dashboards

Best Use Cases for IPFS Hosting:

- Static content (blogs, documentation, portfolios)
- Portfolio websites, galleries, showcases
- Archival content, historical documents, research papers
- Censorship-resistant publishing
- Decentralized applications, smart contracts
- Content needing global resilience without central authority
- Projects with minimal budget constraints
- Software releases, open-source artifacts

Examples: Lab deployments, personal blogs, documentation sites, decentralized websites, blockchain-based applications`