IPFS Constellation

Master Deployment & Operations Guide

Target OS: Ubuntu 24.04 LTS | Version 1.0

1. Master Deployment Script

This script automates the entire provisioning process for a fresh Ubuntu 24.04 LTS server. It handles dependencies, security hardening, IPFS installation, and the Constellation orchestration layer setup.

How to Use This Script

1. Save the code below as `deploy.sh` and make it executable: `chmod +x deploy.sh`

Option A: Deploying the First Node (Primary)

This generates the Cluster Secret and initializes the swarm.

```
sudo ./deploy.sh --role primary
```

Option B: Deploying Subsequent Nodes (Workers)

Use the secret and peer address output from Node 1.

```
sudo ./deploy.sh --role worker \
    --secret "YOUR_GENERATED_SECRET" \
    --bootstrap "/ip4/10.0.0.1/tcp/9096/p2p/12D3K..."
```

The Master Script

```
#!/bin/bash

# IPFS Constellation - Master Deployment Script
# Target OS: Ubuntu 24.04 LTS
# Version: 1.0

# --- CONFIGURATION VARIABLES ---
KUBO_VERSION="v0.29.0"
CONSTELLATION_URL="https://ubitquityx.com/downloads/constellation/v1.0/constell
ation-daemon"
INSTALL_DIR="/usr/local/bin"
CONFIG_DIR="/etc/constellation"
SERVICE_USER="constellation"

# --- COLORS ---
RED='\033[0;31m'
GREEN='\033[0;32m'
CYAN='\033[0;36m'
NC='\033[0m'

# --- HELPER FUNCTIONS ---
log() { echo -e "${CYAN}[INFO]${NC} $1"; }
```

```
error() { echo -e "${RED}[ERROR]${NC} $1"; exit 1; }
success() { echo -e "${GREEN}[SUCCESS]${NC} $1"; }

# --- ROOT CHECK ---
if [ "$EUID" -ne 0 ]; then
  error "Please run as root (sudo ./deploy.sh ...)"
fi

# --- ARGUMENT PARSING ---
ROLE=""
SECRET=""
BOOTSTRAP=""

while [[ "$#" -gt 0 ]]; do
    case $1 in
        --role) ROLE="$2"; shift ;;
        --secret) SECRET="$2"; shift ;;
        --bootstrap) BOOTSTRAP="$2"; shift ;;
        *) error "Unknown parameter: $1" ;;
    esac
    shift
done

if [[ "$ROLE" != "primary" && "$ROLE" != "worker" ]]; then
    error "Usage: ./deploy.sh --role [primary|worker] [--secret <HEX>]
[--bootstrap <MULTIADDR>]"
fi

if [[ "$ROLE" == "worker" && (-z "$SECRET" || -z "$BOOTSTRAP") ]]; then
    error "Workers require --secret and --bootstrap arguments."
fi

# --- STEP 1: SYSTEM PREP ---
log "Updating system and installing dependencies..."
apt-get update -q && apt-get upgrade -y -q
apt-get install -y wget tar jq ufw

# Create Service User
if ! id "$SERVICE_USER" &>/dev/null; then
    useradd -r -s /bin/false $SERVICE_USER
    log "Created service user: $SERVICE_USER"
fi

# --- STEP 2: INSTALL IPFS (KUBO) ---
if ! command -v ipfs &> /dev/null; then
    log "Installing Kubo (IPFS) $KUBO_VERSION..."
    wget -q
"https://dist.ipfs.tech/kubo/${KUBO_VERSION}/kubo_${KUBO_VERSION}_linux-amd64.t
ar.gz" -O kubo.tar.gz
    tar -xvzf kubo.tar.gz
    cd kubo && bash install.sh
    cd .. && rm -rf kubo kubo.tar.gz

    # Initialize IPFS for the service user
    mkdir -p /home/$SERVICE_USER/.ipfs
    chown $SERVICE_USER:$SERVICE_USER /home/$SERVICE_USER/.ipfs
    sudo -u $SERVICE_USER ipfs init --profile server
else
    log "IPFS is already installed."
fi

# Create IPFS Systemd Service
cat <<EOF > /etc/systemd/system/ipfs.service
```

```
[Unit]
Description=IPFS Daemon
After=network.target

[Service]
User=$SERVICE_USER
ExecStart=/usr/local/bin/ipfs daemon --enable-namesys-pubsub
Restart=on-failure
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF

systemctl daemon-reload
systemctl enable ipfs
systemctl start ipfs

# --- STEP 3: INSTALL CONSTELLATION ---
log "Installing Constellation Daemon..."

if [ ! -f "$INSTALL_DIR/constellation-daemon" ]; then
    touch $INSTALL_DIR/constellation-daemon
    chmod +x $INSTALL_DIR/constellation-daemon
    log "Warning: Created placeholder binary. Replace with real binary before
starting!"
fi

# --- STEP 4: CONFIGURE CONSTELLATION ---
mkdir -p $CONFIG_DIR
chown $SERVICE_USER:$SERVICE_USER $CONFIG_DIR

# Generate Secret if Primary
if [[ "$ROLE" == "primary" ]]; then
    SECRET=$(od -vN 32 -An -tx1 /dev/urandom | tr -d ' \n')
    log "Generated New Cluster Secret."
fi

cat <<EOF > $CONFIG_DIR/config.json
{
  "cluster": {
    "secret": "$SECRET",
    "listen_multiaddress": "/ip4/0.0.0.0/tcp/9096",
    "replication_factor_min": 2,
    "replication_factor_max": 3
  },
  "ipfs_connector": {
    "api_multiaddress": "/ip4/127.0.0.1/tcp/5001"
  },
  "consensus": {
    "type": "crdt"
  }
}
EOF
chown $SERVICE_USER:$SERVICE_USER $CONFIG_DIR/config.json

# Create Constellation Systemd Service
cat <<EOF > /etc/systemd/system/constellation.service
[Unit]
Description=IPFS Constellation Daemon
After=ipfs.service
Wants=ipfs.service
```

```
[Service]
User=$SERVICE_USER
ExecStart=$INSTALL_DIR/constellation-daemon start --config
$CONFIG_DIR/config.json
Restart=on-failure
RestartSec=10s
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF

systemctl daemon-reload
systemctl enable constellation

# --- STEP 5: BOOTSTRAPPING (Worker Only) ---
if [[ "$ROLE" == "worker" ]]; then
    log "Configuring bootstrap peer..."
    sed -i "s|start --config|start --bootstrap $BOOTSTRAP --config|" \
        /etc/systemd/system/constellation.service
    systemctl daemon-reload
fi

systemctl start constellation

# --- STEP 6: FIREWALL (UFW) ---
log "Configuring Firewall..."
ufw allow 22/tcp    # SSH
ufw allow 4001/tcp  # IPFS Swarm
ufw allow 4001/udp  # IPFS Swarm
ufw allow 9096/tcp  # Constellation Swarm
ufw allow 8080/tcp  # IPFS Gateway (Optional)
# Note: Port 5001 is NOT exposed publicly
ufw --force enable

# --- FINAL OUTPUT ---
echo ""
success "Deployment Complete!"
echo "-------------------------------------------------"
echo "Role:           $ROLE"
echo "IPFS Status:    $(systemctl is-active ipfs)"
echo "Cluster Status: $(systemctl is-active constellation)"
echo "-------------------------------------------------"

if [[ "$ROLE" == "primary" ]]; then
    echo -e "${RED}IMPORTANT: SAVE THIS SECRET!${NC}"
    echo "Cluster Secret: $SECRET"
    echo ""
    echo "To add a worker node, run:"
    echo "./deploy.sh --role worker --secret $SECRET --bootstrap /ip4/$(curl -s
ifconfig.me)/tcp/9096/p2p/<YOUR_PEER_ID>"
fi
echo "-------------------------------------------------"
```

2. Day 2 Operations Checklist

While "Day 1" is about getting live, "Day 2" is about keeping it alive. This section covers long-term stability, security, and recoverability of your Constellation fleet.

2.1 Backup Strategy (Disaster Recovery)

Since Constellation uses a distributed ledger (CRDT) for state, you don't need to back up every node every day. However, you must protect the Identity and Pinset State to survive a catastrophic "Cluster Wipe" event.

What to Backup

Component	Path	Frequency	Criticality
Node Identity	`~/.ipfs/config`	Once (on creation)	**CRITICAL**
Cluster Secret	`/etc/constellation/config.json`	Once (on creation)	**CRITICAL**
Pinset Snapshot	`constellation export`	Daily	**HIGH**

Automated Backup Script

Place this in /etc/cron.daily/constellation-backup:

```
#!/bin/bash
BACKUP_DIR="/mnt/backups/constellation-$(date +%F)"
mkdir -p $BACKUP_DIR

# 1. Export the current Pinset (List of CIDs)
constellation-cli pin ls > $BACKUP_DIR/pinset_dump.txt

# 2. Backup Identity (Encrypted)
# Assumes you have a GPG key for admin@yourdomain.com
tar -czf - /home/constellation/.ipfs/config | \
    gpg --encrypt -r admin@yourdomain.com > $BACKUP_DIR/identity.tar.gz.gpg

# 3. Push to Off-Site Storage (S3 Example)
aws s3 cp $BACKUP_DIR s3://your-constellation-backups/ --recursive
```

2.2 Log Rotation & Management

IPFS can be chatty. Without rotation, logs will fill the disk, causing a "No Space Left on Device" crash.

Configure Logrotate

File: /etc/logrotate.d/constellation

```
/var/log/constellation.log {
    daily
    rotate 7
    compress
    delaycompress
    missingok
    notifempty
```

```
    create 0640 constellation constellation
    postrotate
        systemctl reload constellation > /dev/null 2>/dev/null || true
    endscript
}
```

Tip: If piping logs to Systemd (journald), limit the journal size in /etc/systemd/journald.conf:

```
SystemMaxUse=500M
```

2.3 Upgrade Procedure (Zero Downtime)

Because your fleet has redundancy (Replication Factor = 3), you can perform Rolling Upgrades without service interruption.

The Workflow

2. **Drain Node 1:** Stop the Constellation daemon. The other 2 nodes continue serving content.
3. **Upgrade Binary:** Download new binary and replace /usr/local/bin/constellation-daemon.
4. **Restart & Verify:** Run systemctl start constellation. Wait for constellation-cli peers ls to show "Synced".
5. **Repeat:** Move to Node 2, then Node 3.

> ⚠️ **WARNING**
> Never upgrade all nodes simultaneously. If the new version has a consensus bug, you risk corrupting the entire fleet's state.

2.4 Security Audits & Key Rotation

Quarterly Audit Checklist

- **Firewall Check:** Ensure only ports 4001 (IPFS), 9096 (Cluster), and 80/443 (Gateway) are open. Port 5001 (API) must be closed to the public.
- **Access Control:** Review SSH keys in ~/.ssh/authorized_keys. Remove former employees immediately.
- **Token Rotation:** If using an "API Key" for the Constellation Proxy, rotate it every 90 days and update client apps.

2.5 Cost Optimization (Storage Tiering)

As your cluster fills up, "Hot" NVMe storage becomes expensive.

- **Metric to Watch:** Disk Usage % via Grafana.
- **Action at 80% Usage:**
  1. **Add Cold Storage:** Add a new node with cheap HDD storage.
  2. **Tagging:** Tag the new node as tier:cold.
  3. **Repin:** Move older content to cold storage:

```
constellation-cli pin update <CID> --allocations "tier:cold"
```

3. Fire Drill Simulation

This procedure trains your team on how to recognize a bad deployment and execute an immediate rollback to the "Last Known Good Configuration" (LKGC).

The Scenario

Element	Description
Objective	Upgrade Node 1 from v1.0 to v1.1
The Failure	The v1.1 binary is corrupt or incompatible (simulated)
The Safety Net	Nodes 2 & 3 remain live, serving traffic while Node 1 is broken
The Fix	Rapid rollback to v1.0

Step 1: The "Bad" Upgrade (Simulation)

Run this on Node 1. We will manually "break" the node to simulate a faulty binary installation.

```
# 1. Check Pre-Upgrade Status (Should be 'active')
systemctl status constellation

# 2. BACKUP THE CURRENT BINARY (Crucial Step!)
sudo cp /usr/local/bin/constellation-daemon
/usr/local/bin/constellation-daemon.bak
echo "Backup created at /usr/local/bin/constellation-daemon.bak"

# 3. Stop the Service
sudo systemctl stop constellation

# 4. Install the "Corrupt" Update
# We simulate a broken binary by creating a script that exits with an error
code
sudo rm /usr/local/bin/constellation-daemon
echo -e '#!/bin/bash\nexit 1' | sudo tee /usr/local/bin/constellation-daemon
sudo chmod +x /usr/local/bin/constellation-daemon

# 5. Attempt to Start the New Version
echo "Attempting to start v1.1..."
sudo systemctl start constellation
```

Step 2: Recognize the Failure

At this point, your monitoring dashboard (Grafana) should show Node 1 as DOWN. Run the following diagnostics:

```
# A. Check Service Status
systemctl status constellation
# Expected Output: Active: failed (Result: exit-code)

# B. Check Logs for "Panic" or "Fatal" errors
journalctl -u constellation -n 20 --no-pager
# Expected Output: "Process exited with status 1"
```

Cluster Impact

- Users can still fetch data (served by Node 2 & 3)
- Users can still upload data (saved to Node 2 & 3)
- Data remains available — confirms "Leaderless Resilience"

Step 3: The Rollback Procedure

Once the failure is confirmed, execute the rollback immediately. Do not try to "debug" the live node while it is down; restore service first, debug later.

```
echo "⚠️ Upgrade Failed. Initiating Rollback..."

# 1. Ensure Service is Stopped (Clean Slate)
sudo systemctl stop constellation

# 2. Restore the Backup Binary
sudo cp /usr/local/bin/constellation-daemon.bak
/usr/local/bin/constellation-daemon
sudo chmod +x /usr/local/bin/constellation-daemon
echo "Binary restored to Previous Version."

# 3. Restart the Service
sudo systemctl start constellation

# 4. Verify Recovery
sleep 5
STATUS=$(systemctl is-active constellation)

if [ "$STATUS" == "active" ]; then
    echo "✅ ROLLBACK SUCCESSFUL. Node 1 is back online."
else
    echo "❌ CRITICAL: Rollback failed. Manual intervention required."
fi
```

Step 4: Post-Mortem Verification

After the rollback, ensure Node 1 has re-joined the swarm and isn't "zombie" (running but disconnected).

```
# Run this on the recovered Node 1
constellation-cli info

# Expected Output:
# ID: 12D3K...
# Version: 1.0.0 (The old version)
# Peers: 2 (Connected to Node 2 & 3)
# Sync State: SYNCED
```

Summary of Drill Results

Metric	Result
Downtime	~30 seconds (for Node 1 only)
Customer Impact	**Zero** (handled by Nodes 2 & 3)
Data Loss	**Zero**

Conclusion

This document provides the complete end-to-end package for IPFS Constellation:

- Whitepaper Analysis
- MVC Architecture Design
- Deployment Scripts
- Developer Integration Guide
- Day 2 Ops & Disaster Recovery Procedures