Manual MongoDB Installation Using Tarball Method

As a MongoDB DBA with 15 years of experience, this comprehensive guide outlines a secure, production-ready manual installation of MongoDB 8.0 Community Edition using the tarball method on Linux systems. This approach provides complete control over the installation process and is ideal for environments where package managers are unavailable or custom configurations are required.

Installation Overview & Key Benefits

The manual tarball installation method offers maximum flexibility and control for MongoDB deployments. This approach is particularly valuable for production environments with specific security requirements, custom directory structures, or offline installation scenarios.

1 Download and Extract Binaries

Fetch the official MongoDB tarball from the download center and extract binaries to a secure, custom installation path with proper ownership and permissions.

2 Create Dedicated System User

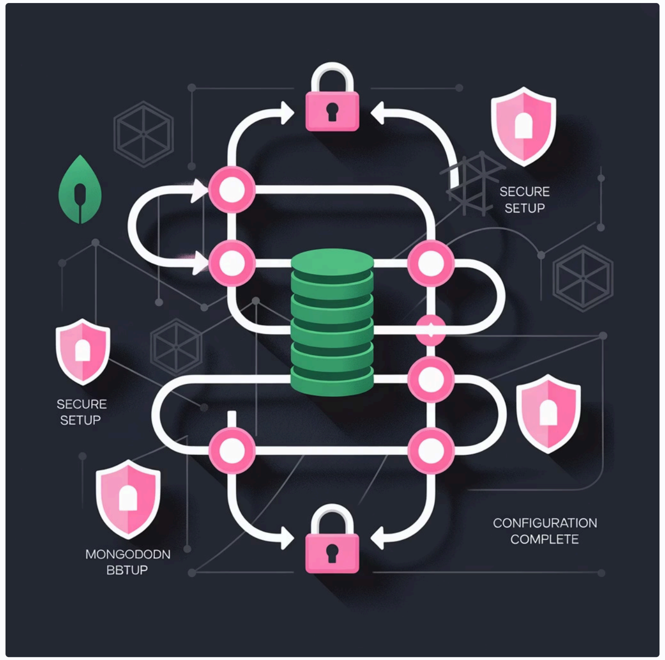
Establish a non-root mongodb user account for process isolation, following security best practices with restricted shell access and dedicated group membership.

3 Configure Directory Structure

Set up data, log, and runtime directories with appropriate permissions, ensuring secure file system access and proper logging capabilities.

4 Enable Security Features

Implement authentication, access controls, and optional TLS encryption to protect against unauthorized access and ensure data security.



This installation method assumes a 64-bit Ubuntu LTS system for demonstration, with specific notes provided for Red Hat/CentOS variants throughout the guide.

System Prerequisites & Platform Requirements

MongoDB 8.0 requires specific system prerequisites to ensure optimal performance and stability. Proper preparation of the target environment is crucial for a successful production deployment.

Operating System

64-bit Architecture Required: x86_64 (Intel Sandy Bridge+ or AMD Bulldozer+) and ARM64 (v8.2-A+)

Supported Distributions:

- Ubuntu 22.04/24.04 LTS
- Red Hat Enterprise Linux 8/9
- CentOS Stream 8/9
- Rocky Linux 8/9

Hardware Specifications

Minimum Requirements:

- 4GB RAM (16GB+ recommended)
- 4 CPU cores minimum
- 10GB disk space minimum
- SSD storage strongly recommended

Production Sizing: Plan for working set to fit in memory, with additional capacity for growth and caching.

System Dependencies

Ubuntu Dependencies:

```
libcurl4 libgssapi-krb5-2 libldap-2.5-0
libwrap0 libsasl2-2 libsasl2-modules
libsasl2-modules-gssapi-mit openssl liblzma5
```

Red Hat Dependencies:

```
libcurl openssl xz-libs krb5-libs
libldap openssl-libs cyrus-sasl
```

System optimization requires setting appropriate ulimits (nofile ≥ 64000, nproc ≥ 32000), disabling Transparent Hugepages for MongoDB 7.0 and earlier, configuring vm.swappiness=1, and enabling NTP for time synchronization. Use XFS filesystem with readahead settings of 8-32 for optimal I/O performance.

Pre-Installation System Configuration

System Tuning Parameters

Before installing MongoDB, several system-level configurations must be optimized to ensure peak performance and stability in production environments.

Configure System Limits:

```
# Edit /etc/security/limits.conf
mongodb soft nproc 32000
mongodb hard nproc 32000
mongodb soft nofile 64000
mongodb hard nofile 64000
```

Disable Transparent Huge Pages (THP):

```
# For MongoDB 7.0 and earlier
echo 'never' | sudo tee /sys/kernel/mm/transparent_hugepage/enabled
echo 'never' | sudo tee /sys/kernel/mm/transparent_hugepage/defrag
```

Configure Memory Management:

```
# Set swappiness to minimize swap usage
echo 'vm.swappiness = 1' | sudo tee -a /etc/sysctl.conf
sysctl -p
```

Time Synchronization Setup:

```
# Install and configure NTP
sudo apt-get install ntp  # Ubuntu
sudo yum install ntp      # RHEL/CentOS
sudo systemctl enable ntp
sudo systemctl start ntp
```

Filesystem Recommendations:

- Use XFS filesystem for data directories
- Mount with noatime or relatime options
- Configure readahead to 8-32 blocks
- Ensure sufficient disk space for growth

NUMA Considerations:

For multi-socket servers, disable NUMA in BIOS or use numactl --interleave=all when starting mongod to prevent memory distribution issues.

Download and Extract MongoDB Binaries

The first step involves downloading the official MongoDB Community Edition tarball from the MongoDB website and extracting it to the designated installation directory.

01

Download Official Tarball

Visit the MongoDB Community Download Center and select the appropriate package for your system architecture and operating system.

```
# Download MongoDB 8.0 Community Edition
cd /tmp
wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu2204-8.0.0.tgz
```

For Red Hat systems, use the rhel80 or rhel90 variant instead of ubuntu2204.

02

Verify Download Integrity

Always verify the integrity of downloaded files using checksums to ensure the package hasn't been corrupted or tampered with.

```
# Download and verify checksum (optional but recommended)
wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu2204-8.0.0.tgz.sha256
sha256sum -c mongodb-linux-x86_64-ubuntu2204-8.0.0.tgz.sha256
```

03

Extract to Installation Directory

Create the installation directory and extract the tarball contents, removing the top-level directory structure for cleaner organization.

```
# Create installation directory
sudo mkdir -p /opt/mongodb

# Extract with strip-components to remove top directory
sudo tar -zxvf mongodb-linux-x86_64-ubuntu2204-8.0.0.tgz -C /opt/mongodb --strip-components=1

# Set proper ownership
sudo chown -R root:root /opt/mongodb
```

Creating Dedicated MongoDB User and Directories

Security best practices require running MongoDB under a dedicated, non-privileged user account. This approach follows the principle of least privilege and provides process isolation from other system services.

Create System User and Group

```
# Create mongodb group
sudo groupadd mongodb

# Create system user with no login shell
sudo useradd -r -s /bin/false -g mongodb mongodb

# Verify user creation
id mongodb
```

Create Directory Structure

```
# Create essential directories
sudo mkdir -p /var/lib/mongo      # Data directory
sudo mkdir -p /var/log/mongodb    # Log directory
sudo mkdir -p /var/run/mongodb    # Runtime/PID directory

# Set ownership to mongodb user
sudo chown -R mongodb:mongodb /var/lib/mongo
sudo chown -R mongodb:mongodb /var/log/mongodb
sudo chown -R mongodb:mongodb /var/run/mongodb

# Set secure permissions (owner-only access)
sudo chmod -R 700 /var/lib/mongo
sudo chmod -R 700 /var/log/mongodb
sudo chmod -R 755 /var/run/mongodb
```

SELinux Configuration (Red Hat Systems):

On Red Hat systems with SELinux enabled, additional policy configuration may be required:

```
# Install SELinux policy tools
sudo yum install git make

# Clone MongoDB SELinux policy
git clone https://github.com/mongodb/mongodb-selinux
cd mongodb-selinux

# Build and install policy
make
sudo make install
```

Directory Purpose:

- **/var/lib/mongo:** Database files and indexes
- **/var/log/mongodb:** Server and audit logs
- **/var/run/mongodb:** Process ID files

Binary Path Configuration and Access

Making MongoDB binaries accessible system-wide requires updating the PATH environment variable and creating appropriate symbolic links or direct PATH modifications.

1

Global PATH Update

Add MongoDB binaries to the system-wide PATH for easy command access:

```
# Edit system profile
sudo nano /etc/profile

# Add MongoDB bin directory to PATH
export PATH=/opt/mongodb/bin:$PATH

# Apply changes
source /etc/profile
```

2

Create Symbolic Links

Alternative approach using symbolic links to standard binary locations:

```
# Create links in /usr/local/bin
sudo ln -s /opt/mongodb/bin/mongod
/usr/local/bin/mongod
sudo ln -s /opt/mongodb/bin/mongos
/usr/local/bin/mongos
sudo ln -s /opt/mongodb/bin/mongodump
/usr/local/bin/mongodump
sudo ln -s /opt/mongodb/bin/mongorestore
/usr/local/bin/mongorestore
```

3

Verify Installation

Test that MongoDB binaries are accessible and display version information:

```
# Check mongod version
mongod --version

# Verify all binaries are accessible
which mongod
which mongos
which mongodump
```

Note: The mongosh (MongoDB Shell) is distributed separately and should be downloaded and installed from the MongoDB download center following similar extraction and PATH configuration steps.

MongoDB Configuration File Creation

The MongoDB configuration file (/etc/mongod.conf) defines all operational parameters for the database server. This YAML-formatted file controls storage, networking, logging, and security settings.

Basic Configuration Template

```
# /etc/mongod.conf
storage:
  dbPath: /var/lib/mongo
  journal:
    enabled: true
  engine: wiredTiger
  wiredTiger:
    engineConfig:
      cacheSizeGB: 2

systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log
  logRotate: rename

net:
  port: 27017
  bindIp: 127.0.0.1  # Localhost only initially
  maxIncomingConnections: 1000

processManagement:
  fork: true
  pidFilePath: /var/run/mongodb/mongod.pid
  timeZoneInfo: /usr/share/zoneinfo
```

Configuration Parameters Explained:

- **cacheSizeGB:** WiredTiger cache size (defaults to 50% of RAM minus 1GB, adjust based on available memory)
- **bindIp:** Network interfaces to bind to (127.0.0.1 for localhost only, 0.0.0.0 for all interfaces)
- **maxIncomingConnections:** Maximum concurrent client connections (default 65536, adjust for workload)
- **logRotate:** Log rotation method (rename creates numbered log files, reopen works with external tools)

Security and Performance Settings

```
# Additional configuration sections
security:
  authorization: disabled  # Enable after user creation

operationProfiling:
  slowOpThresholdMs: 100
  mode: slowOp

# For replica sets (add when needed)
#replication:
#  replSetName: "rs0"

# For sharded clusters (add when needed)
#sharding:
#  clusterRole: shardsvr
```

Set Configuration File Permissions:

```
# Create configuration file
sudo nano /etc/mongod.conf

# Set secure ownership and permissions
sudo chown root:mongodb /etc/mongod.conf
sudo chmod 640 /etc/mongod.conf
```

SystemD Service Configuration

Creating a proper systemd service ensures MongoDB starts automatically on system boot, follows system service standards, and provides proper process management and logging integration.

```
# Create /etc/systemd/system/mongod.service
[Unit]
Description=MongoDB Database Server
Documentation=https://docs.mongodb.org/manual
After=network-online.target
Wants=network-online.target

[Service]
User=mongodb
Group=mongodb
Type=forking
PIDFile=/var/run/mongodb/mongod.pid
ExecStart=/usr/local/bin/mongod --config /etc/mongod.conf
ExecReload=/bin/kill -HUP $MAINPID
Restart=on-failure
RestartSec=10
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=mongod

# Resource limits for production
LimitFSIZE=infinity
LimitCPU=infinity
LimitAS=infinity
LimitNOFILE=64000
LimitNPROC=64000
LimitMEMLOCK=infinity
TasksMax=infinity
TasksAccounting=false

[Install]
WantedBy=multi-user.target
```

1

Enable Service

```
# Reload systemd daemon
sudo systemctl daemon-reload

# Enable automatic startup
sudo systemctl enable mongod
```

2

Start MongoDB

```
# Start the service
sudo systemctl start mongod

# Check status
sudo systemctl status mongod
```

3

Verify Operation

```
# Check process
ps aux | grep mongod

# Check port binding
netstat -tlnp | grep 27017
```

Initial Security Configuration

Security configuration is critical for production MongoDB deployments. By default, MongoDB starts without authentication enabled, which is acceptable for development but must be changed for production use.

Create Administrative User

Before enabling authentication, create an administrative user with full privileges:

```
# Connect to MongoDB (no auth required initially)
mongosh

# Switch to admin database
use admin

# Create administrative user
db.createUser({
  user: "mongoAdmin",
  pwd: passwordPrompt(),  // Will prompt for password
  roles: [
    { role: "userAdminAnyDatabase", db: "admin" },
    { role: "readWriteAnyDatabase", db: "admin" },
    { role: "dbAdminAnyDatabase", db: "admin" },
    { role: "clusterAdmin", db: "admin" }
  ]
})

# Exit MongoDB shell
exit
```

Enable Authentication

```
# Edit configuration file
sudo sed -i 's/authorization: disabled/authorization: enabled/' /etc/mongod.conf

# Restart MongoDB service
sudo systemctl restart mongod

# Test authentication
mongosh -u mongoAdmin -p --authenticationDatabase admin
```

Create Application User:

```
# Connect as admin
mongosh -u mongoAdmin -p --authenticationDatabase admin

# Switch to application database
use myappdb

# Create application-specific user
db.createUser({
  user: "appuser",
  pwd: passwordPrompt(),
  roles: [
    { role: "readWrite", db: "myappdb" }
  ]
})
```

Role-Based Security:

- **userAdminAnyDatabase:** Manage users and roles
- **dbAdminAnyDatabase:** Database administration tasks
- **readWriteAnyDatabase:** Read/write data in any database
- **clusterAdmin:** Cluster management operations

Network Security and Firewall Configuration

Proper network security configuration ensures MongoDB is accessible only to authorized systems while protecting against unauthorized access from external networks.

Firewall Rules (firewalld - RHEL/CentOS)

```
# Allow MongoDB port for specific sources
sudo firewall-cmd --permanent --add-rich-rule="rule
family='ipv4' source address='192.168.1.0/24' port
protocol='tcp' port='27017' accept"

# Or allow from all (less secure)
sudo firewall-cmd --permanent --add-port=27017/tcp

# Apply changes
sudo firewall-cmd --reload

# List active rules
sudo firewall-cmd --list-all
```

UFW Configuration (Ubuntu)

```
# Allow from specific subnet
sudo ufw allow from 192.168.1.0/24 to any port 27017

# Or allow from specific IP
sudo ufw allow from 192.168.1.100 to any port 27017

# Enable UFW if not already active
sudo ufw enable

# Check status
sudo ufw status verbose
```

MongoDB Network Binding

```
# Update mongod.conf for network access
net:
  port: 27017
  bindIp: 127.0.0.1,192.168.1.10  # Add server IP
  maxIncomingConnections: 1000

# Restart after changes
sudo systemctl restart mongod
```

Security Note: Never bind to 0.0.0.0 (all interfaces) unless the server is behind a properly configured firewall.

Additional Security Measures:

- Use VPN or private networks for database access
- Implement IP whitelisting in application configuration
- Consider using MongoDB's built-in IP binding restrictions
- Monitor connection logs for unauthorized access attempts

TLS/SSL Encryption Configuration

Implementing TLS encryption ensures data in transit between MongoDB clients and the server is protected from eavesdropping and man-in-the-middle attacks.

Generate SSL Certificates

```
# Create certificate directory
sudo mkdir -p /etc/ssl/mongodb

# Generate private key
sudo openssl genrsa -out /etc/ssl/mongodb/mongodb.key 2048

# Generate certificate signing request
sudo openssl req -new -key /etc/ssl/mongodb/mongodb.key -out /etc/ssl/mongodb/mongodb.csr

# Generate self-signed certificate (for testing)
sudo openssl x509 -req -days 365 -in /etc/ssl/mongodb/mongodb.csr -signkey /etc/ssl/mongodb/mongodb.key -
out /etc/ssl/mongodb/mongodb.crt

# Combine key and certificate for MongoDB
sudo cat /etc/ssl/mongodb/mongodb.key /etc/ssl/mongodb/mongodb.crt > /etc/ssl/mongodb/mongodb.pem

# Set permissions
sudo chown mongodb:mongodb /etc/ssl/mongodb/mongodb.pem
sudo chmod 600 /etc/ssl/mongodb/mongodb.pem
```

Update MongoDB Configuration

```
# Add to mongod.conf
net:
  tls:
    mode: requireTLS
    certificateKeyFile: /etc/ssl/mongodb/mongodb.pem
    allowConnectionsWithoutCertificates: true
```

TLS Mode Options:

- **disabled:** No TLS (default, not recommended for production)
- **allowTLS:** Accept both TLS and non-TLS connections
- **preferTLS:** Use TLS for client connections if available
- **requireTLS:** Require TLS for all connections

Client Connection with TLS:

```
# Connect using TLS
mongosh --tls --host localhost:27017

# With authentication
mongosh --tls --host localhost:27017 -u mongoAdmin -p --authenticationDatabase admin
```

Production Considerations:

- Use certificates from a trusted CA in production
- Configure certificate validation on clients
- Set up certificate rotation procedures
- Monitor certificate expiration dates

Log Management and Rotation

Proper log management ensures MongoDB operational logs don't consume excessive disk space while maintaining adequate historical data for troubleshooting and audit purposes.

1 Configure Log Rotation

```
# Create /etc/logrotate.d/mongod
/var/log/mongodb/mongod.log {
daily
missingok
rotate 52
compress
delaycompress
notifempty
create 640 mongodb mongodb
postrotate
/bin/kill -SIGUSR1 $(cat /var/run/mongodb/mongod.pid 2>/dev/null) 2>/dev/null || true
endscript
}
```

2 MongoDB Log Settings

```
# Enhanced logging configuration
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log
  logRotate: reopen  # Use with logrotate
  component:
    accessControl:
      verbosity: 1
    command:
      verbosity: 1
```

3 Log Analysis Tools

```
# View recent logs
sudo tail -f /var/log/mongodb/mongod.log

# Search for errors
sudo grep -i error /var/log/mongodb/mongod.log

# Monitor slow operations
sudo grep "slow operation" /var/log/mongodb/mongod.log
```

Log Monitoring Best Practices:

- Set up automated alerts for ERROR level messages
- Monitor disk space in log directories
- Implement centralized logging for multiple instances
- Configure appropriate verbosity levels for different components
- Regular log analysis to identify performance issues

Verification, Testing, and Troubleshooting

Installation Verification Tests

```
# 1. Service status verification
sudo systemctl status mongod
sudo systemctl is-active mongod
sudo systemctl is-enabled mongod

# 2. Process and port verification
ps aux | grep mongod
sudo netstat -tlnp | grep 27017
sudo ss -tlnp | grep 27017

# 3. Log file verification
sudo tail -20 /var/log/mongodb/mongod.log
sudo grep "waiting for connections" /var/log/mongodb/mongod.log

# 4. Database connectivity test
mongosh --eval "db.runCommand({connectionStatus: 1})"

# 5. Authentication test
mongosh -u mongoAdmin -p --authenticationDatabase admin --eval "db.adminCommand('ismaster')"
```

Functional Database Tests

```
# Write/Read test with authentication
mongosh -u mongoAdmin -p --authenticationDatabase admin --eval "
use testdb;
db.testcol.insertOne({test: 'data', timestamp: new Date()});
db.testcol.findOne();
db.testcol.drop();
"

# Performance test
mongosh --eval "
for(var i=0; i<1000; i++) {
  db.perftest.insertOne({counter: i, data: 'test data'});
}
print('Inserted 1000 documents');
db.perftest.countDocuments();
db.perftest.drop();"
```

Common Issues and Solutions

Symptom	Possible Cause	Solution
Service fails to start	Configuration errors, permission issues	Check journalctl -u mongod -f, verify config syntax
Permission denied errors	Incorrect file ownership	sudo chown -R mongodb:mongodb /var/lib/mongo
Port binding failed	Port already in use	sudo lsof -i :27017, kill conflicting process
Authentication failures	User/password issues	Verify credentials, check user roles
High memory usage	WiredTiger cache size	Adjust cacheSizeGB in configuration

Log Analysis Commands

```
# System service logs
sudo journalctl -u mongod --since "1 hour ago"
sudo journalctl -u mongod --follow

# MongoDB-specific log analysis
sudo grep -E "(ERROR|SEVERE)" /var/log/mongodb/mongod.log
sudo tail -f /var/log/mongodb/mongod.log | grep -i "slow"
```

Production Hardening and Next Steps

With the basic installation complete, implementing production hardening measures and planning for operational excellence ensures your MongoDB deployment is ready for enterprise workloads.

✓

Security Checklist

- Authentication enabled with strong passwords
- TLS encryption configured for data in transit
- Firewall rules restricting access to authorized hosts
- Regular security updates scheduled
- Audit logging enabled (Enterprise feature)

✓

Operational Excellence

- Monitoring and alerting configured
- Backup strategy implemented and tested
- Log rotation and management configured
- Documentation maintained and accessible
- Change management procedures established

→

Scalability Planning

- Replica set configuration for high availability
- Sharding strategy for horizontal scaling
- Performance monitoring and optimization
- Capacity planning based on growth projections
- Migration to MongoDB Atlas consideration

Essential Maintenance Commands:

```
# Service management
sudo systemctl start mongod
sudo systemctl stop mongod
sudo systemctl restart mongod
sudo systemctl status mongod

# Configuration backup
sudo cp /etc/mongod.conf /etc/mongod.conf.backup.$(date +%Y%m%d)

# Database backup
mongodump --uri="mongodb://mongoAdmin:password@localhost:27017/?authSource=admin" --out /backup/$(date +%Y%m%d)

# Disk usage monitoring
du -sh /var/lib/mongo/
df -h /var/lib/mongo
```

Next Steps for Production Deployment:

1. **High Availability:** Configure a 3-node replica set for automatic failover
2. **Monitoring:** Implement MongoDB Compass, Ops Manager, or third-party monitoring
3. **Backup Strategy:** Set up automated backups with point-in-time recovery
4. **Performance Tuning:** Optimize based on workload patterns and resource utilization
5. **Security Hardening:** Implement additional security measures like IP whitelisting and role-based access

This manual installation provides complete control over the MongoDB deployment and serves as a solid foundation for production workloads. Regular maintenance, monitoring, and security updates will ensure optimal performance and reliability.