Deployment Guide for Research made Readable

This guide provides instructions for deploying the Research made Readable application on an external Virtual Machine (VM).

Prerequisites

- Ubuntu 20.04 LTS or later
- Python 3.8 or higher
- 2GB RAM minimum (4GB recommended)
- 10GB disk space minimum

> **Simplified Deployment**: No external database installation required! The application uses DuckDB with Parquet file storage for a completely self-contained setup.

VM Setup

1. Initial Server Setup

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Install required system packages
sudo apt install -y python3 python3-pip python3-venv git nginx supervisor

# Create application user
sudo useradd -m -s /bin/bash research-made-readable
sudo usermod -aG sudo research-made-readable
```

2. Database Setup

> **No Database Setup Required!** The application uses DuckDB with Parquet files for data storage. Database files are created automatically in the `data/db/` directory when the application first runs.

Benefits of this approach:
- No external database server installation or configuration
- Automatic database initialization on first startup
- Portable data storage - simply copy the `data/db/` directory to backup or migrate
- No database credentials or connection strings to manage

3. Application Deployment

```
# Switch to application user
sudo -u research-made-readable -i

# Clone the application
git clone <repository-url> /home/research-made-readable/research_summary_app
cd /home/research-made-readable/research_summary_app

# Create virtual environment
python3 -m venv venv
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt

# Set up environment configuration (see detailed section below)
cp .env-example .env
nano .env  # Edit with your actual API key

# Set secure permissions
chmod 600 .env

# Create required directories
mkdir -p data/db data/uploads data/exports logs

# Initialize application (database files created automatically)
python setup.py
```

4. Environment Configuration

🔑 API Key Setup (REQUIRED)

The application requires an AbacusAI API key to access all AI models. Here's how to set it up securely in production:

1. **Obtain your AbacusAI API key:**
   - Visit AbacusAI (https://abacus.ai/)
   - Sign up for an account or log in
   - Navigate to your account settings or API section
   - Generate a new API key
   - Copy the key for configuration

2. **Configure environment variables:**
   ```bash
   # Edit the .env file with your actual API key
   nano .env

# Add your API key (replace with your actual key)
ABACUSAI_API_KEY=your_actual_api_key_here

# Optional: Add other configuration settings
# LOG_LEVEL=INFO
# DEBUG=False
```

```
# MAX_UPLOAD_SIZE=10
```

1. **Secure the environment file:**
   ```bash
   # Set restrictive permissions (owner read/write only)
   chmod 600 .env

# Verify permissions
ls -la .env
# Should show: -rw------- 1 research-made-readable research-made-readable
```

🔒 Production Security Considerations

Environment Variables Security:
- Never commit the `.env` file to version control
- Use restrictive file permissions (600 or 640 maximum)
- Consider using system environment variables for enhanced security:
`bash`
`  # Alternative: Set system environment variable`
`  echo 'export ABACUSAI_API_KEY="your_key_here"' >> ~/.bashrc`
`  source ~/.bashrc`

API Key Management:
- Use separate API keys for development and production
- Monitor API usage through your AbacusAI dashboard
- Set up usage alerts to prevent unexpected charges
- Rotate API keys regularly for security

Network Security:
- Restrict API access to your server's IP if possible
- Use HTTPS in production (see SSL Certificate section)
- Consider using a reverse proxy for additional security

💰 Cost Management and Monitoring

API Usage Optimization:
- Monitor API usage through AbacusAI dashboard
- Set up billing alerts and usage limits
- Use lower-cost models (e.g., GPT-4-Mini) for development/testing
- Implement rate limiting to prevent abuse

Usage Tracking:

```
# Monitor application logs for API usage
tail -f logs/streamlit.log | grep -i "api\|error"

# Check supervisor logs
tail -f /var/log/research-made-readable.log
```

🛠️ Troubleshooting Environment Issues

Common API Key Issues:

1. **Invalid API Key Error:**
   ```bash
   # Check if API key is set
   grep ABACUSAI_API_KEY .env

# Test API key validity
curl -H “Authorization: Bearer your_api_key_here” \
https://apps.abacus.ai/v1/chat/completions
```

1. **Permission Errors:**
   `bash`
   `  # Fix file permissions`
   `  sudo chown research-made-readable:research-made-readable .env`
   `  chmod 600 .env`

2. **Environment Not Loading:**
   ```bash
   # Verify python-dotenv is installed
   pip list | grep python-dotenv

# Check if .env file exists in correct location
ls -la .env
```

Environment Validation Script:

```
# Create a simple validation script
cat > validate_env.py << 'EOF'
import os
from dotenv import load_dotenv

load_dotenv()

api_key = os.getenv('ABACUSAI_API_KEY')
if api_key:
    print(f"✅ API key found: {api_key[:10]}...")
else:
    print("❌ API key not found")

print(f"✅ Working directory: {os.getcwd()}")
print(f"✅ .env file exists: {os.path.exists('.env')}")
EOF

python validate_env.py
```

5. Nginx Configuration

```
# Create Nginx configuration
sudo nano /etc/nginx/sites-available/research-made-readable

# Add configuration:
server {
    listen 80;
    server_name your_domain.com;  # Replace with your domain

    location / {
        proxy_pass http://localhost:8501;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
        proxy_read_timeout 86400;
    }
}

# Enable site
sudo ln -s /etc/nginx/sites-available/research-made-readable /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

6. Supervisor Configuration

```
# Create supervisor configuration
sudo nano /etc/supervisor/conf.d/research-made-readable.conf

# Add configuration:
[program:research-made-readable]
command=/home/research-made-readable/research_summary_app/venv/bin/streamlit run
app.py --server.port=8501 --server.address=localhost
directory=/home/research-made-readable/research_summary_app
user=research-made-readable
autostart=true
autorestart=true
redirect_stderr=true
stdout_logfile=/var/log/research-made-readable.log
environment=PATH="/home/research-made-readable/research_summary_app/venv/bin"

# Update supervisor
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl start research-made-readable
```

7. SSL Certificate (Optional but Recommended)

```
# Install Certbot
sudo apt install certbot python3-certbot-nginx

# Get SSL certificate
sudo certbot --nginx -d your_domain.com

# Auto-renewal
sudo crontab -e
# Add: 0 12 * * * /usr/bin/certbot renew --quiet
```

Monitoring and Maintenance

1. Application Monitoring

```
# Check application status
sudo supervisorctl status research-made-readable

# View logs
sudo tail -f /var/log/research-made-readable.log

# Restart application
sudo supervisorctl restart research-made-readable
```

2. Data Backup and Maintenance

```
# Backup all data including environment configuration
cd /home/research-made-readable/research_summary_app

# Create comprehensive backup (data + configuration)
tar -czf backup_$(date +%Y%m%d).tar.gz data/db/ logs/ .env-example
# Note: .env file is excluded for security - document API key separately

# Alternative: Copy data directory only
cp -r data/db/ ../backups/backup_$(date +%Y%m%d)/

# Restore data (simple file copy)
cd /home/research-made-readable/research_summary_app
tar -xzf backup_20240101.tar.gz

# Verify data integrity (optional)
python -c "
import duckdb
conn = duckdb.connect('data/db/research_app.duckdb')
print('Tables:', conn.execute('SHOW TABLES').fetchall())
conn.close()
print('Data verification complete')
"
```

Environment Configuration Backup:
- **Do NOT backup the `.env` file** in version control or regular backups
- **Securely document your API key** in a password manager or secure note
- **Keep a copy of `.env-example`** for reference when setting up new environments
- **Test environment restoration** by validating API key access after restore

Automated Backup Script:

```
#!/bin/bash
# Create automated backup script
cat > backup_app.sh << 'EOF'
#!/bin/bash
BACKUP_DIR="/home/research-made-readable/backups"
APP_DIR="/home/research-made-readable/research_summary_app"
DATE=$(date +%Y%m%d_%H%M%S)

mkdir -p $BACKUP_DIR
cd $APP_DIR

# Backup data and logs (excluding sensitive .env file)
tar -czf $BACKUP_DIR/research_app_backup_$DATE.tar.gz \
    data/db/ \
    logs/ \
    .env-example \
    --exclude='.env'

echo "Backup completed: $BACKUP_DIR/research_app_backup_$DATE.tar.gz"

# Keep only last 7 days of backups
find $BACKUP_DIR -name "research_app_backup_*.tar.gz" -mtime +7 -delete
EOF

chmod +x backup_app.sh

# Add to crontab for daily backup
crontab -e
# Add: 0 2 * * * /home/research-made-readable/research_summary_app/backup_app.sh
```

3. System Updates

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Update Python packages
cd /home/research-made-readable/research_summary_app
source venv/bin/activate
pip install --upgrade -r requirements.txt

# Restart services
sudo supervisorctl restart research-made-readable
```

Security Considerations

1. Firewall Configuration

```
# Install UFW
sudo apt install ufw

# Configure firewall
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow ssh
sudo ufw allow 'Nginx Full'
sudo ufw enable
```

2. Application Security

```
# Set proper file permissions
sudo chown -R research-made-readable:research-made-readable /home/research-made-read-
able/research_summary_app
sudo chmod -R 755 /home/research-made-readable/research_summary_app
sudo chmod 600 /home/research-made-readable/research_summary_app/.env

# Configure secure headers in Nginx
sudo nano /etc/nginx/sites-available/research-made-readable
# Add to server block:
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";
```

3. Data Security

```
# Secure data directory permissions
sudo chown -R research-made-readable:research-made-readable /home/research-made-read-
able/research_summary_app/data/
sudo chmod -R 750 /home/research-made-readable/research_summary_app/data/db/

# Secure environment file
sudo chmod 600 /home/research-made-readable/research_summary_app/.env

# Note: DuckDB files are local to the application - no network security concerns
```

Backup and Recovery

1. Automated Backups

```
# Create backup script
sudo nano /home/research-made-readable/backup.sh

#!/bin/bash
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR="/home/research-made-readable/backups"
APP_DIR="/home/research-made-readable/research_summary_app"
mkdir -p $BACKUP_DIR

# Data backup (DuckDB and Parquet files)
tar -czf $BACKUP_DIR/data_backup_$DATE.tar.gz -C $APP_DIR data/db/

# Full application backup (including data)
tar -czf $BACKUP_DIR/app_backup_$DATE.tar.gz $APP_DIR

# Clean old backups (keep last 30 days)
find $BACKUP_DIR -name "*_backup_*.tar.gz" -mtime +30 -delete

echo "Backup completed: $DATE"

# Make executable
sudo chmod +x /home/research-made-readable/backup.sh

# Add to crontab for daily backups
sudo crontab -e
# Add: 0 2 * * * /home/research-made-readable/backup.sh
```

2. Recovery Procedures

```
# Stop application first
sudo supervisorctl stop research-made-readable

# Restore data only (DuckDB and Parquet files)
cd /home/research-made-readable/research_summary_app
tar -xzf /home/research-made-readable/backups/data_backup_YYYYMMDD_HHMMSS.tar.gz

# OR restore entire application
tar -xzf /home/research-made-readable/backups/app_backup_YYYYMMDD_HHMMSS.tar.gz -C /
home/research-made-readable/

# Set proper permissions
sudo chown -R research-made-readable:research-made-readable /home/research-made-read-
able/research_summary_app
sudo chmod -R 750 /home/research-made-readable/research_summary_app/data/db/

# Restart application
sudo supervisorctl start research-made-readable
```

Performance Optimization

1. Data Storage Optimization

```
# DuckDB automatically optimizes performance, but you can:

# Monitor disk space usage
df -h /home/research-made-readable/research_summary_app/data/db/

# Check Parquet file sizes
ls -lah /home/research-made-readable/research_summary_app/data/db/*.parquet

# Ensure sufficient disk space for data growth
# DuckDB compresses data efficiently with Parquet format
# Typical compression ratio: 5-10x compared to raw CSV data

# For large datasets, consider SSD storage for better I/O performance
```

2. Application Optimization

```
# Configure Streamlit for production
nano /home/research-made-readable/research_summary_app/.streamlit/config.toml

[server]
maxUploadSize = 200
maxMessageSize = 200
enableCORS = false
enableXsrfProtection = true

[browser]
gatherUsageStats = false

[theme]
primaryColor = "#2563EB"
backgroundColor = "#FFFFFF"
secondaryBackgroundColor = "#F8F9FA"
textColor = "#1F2937"
```

Troubleshooting

Common Issues

1. **Application won't start**
   - Check supervisor logs: `sudo tail -f /var/log/researchlens.log`
   - Verify environment variables in `.env`
   - Check database connectivity

2. **Data storage issues**
   - Check if `data/db/` directory exists and is writable
   - Verify Parquet files are not corrupted: `python -c "import duckdb; duckdb.connect('data/db/research_app.duckdb').execute('SHOW TABLES')"`
   - Check disk space: `df -h /home/research-made-readable/research_summary_app/data/db/`

3. **Nginx errors**
   - Check Nginx configuration: `sudo nginx -t`
   - Review Nginx logs: `sudo tail -f /var/log/nginx/error.log`

4. **SSL certificate issues**
   - Check certificate status: `sudo certbot certificates`
   - Renew certificate: `sudo certbot renew`

Performance Issues

1. **Slow data operations**
   - DuckDB automatically optimizes queries
   - Check available disk space for temporary operations
   - Consider SSD storage for better I/O performance
   - Monitor Parquet file sizes for unexpected growth

2. **High memory usage**
   - Monitor with `htop` or `top`
   - Adjust Streamlit configuration
   - DuckDB has efficient memory management for Parquet files
   - Consider upgrading server resources for large datasets

Maintenance Schedule

Daily

- Monitor application logs
- Check system resources
- Verify backup completion
- Monitor disk space in `data/db/` directory

Weekly

- Review security logs
- Update system packages
- Check Parquet file sizes and growth trends
- Verify data directory permissions

Monthly

- Analyze application performance
- Review DuckDB storage efficiency
- Update application dependencies
- Security audit
- Test backup and recovery procedures

🚀 DuckDB/Parquet Architecture Benefits

Simplified Deployment

- **No Database Server**: Eliminated PostgreSQL installation and configuration
- **Self-Contained**: All data stored in portable Parquet files
- **Reduced Dependencies**: Fewer system packages and services to manage

- **Faster Setup**: Database initialization happens automatically

Operational Advantages

- **Simple Backups**: Copy the `data/db/` directory - that's it!
- **Easy Migration**: Transfer the entire application directory to any server
- **No Database Credentials**: No connection strings or passwords to manage
- **Portable Development**: Identical setup for development and production

Performance & Reliability

- **Optimized Storage**: Parquet format provides excellent compression and query performance
- **ACID Compliance**: DuckDB ensures data integrity
- **Automatic Optimization**: No manual database tuning required
- **Efficient Memory Usage**: DuckDB designed for analytical workloads

Deployment Comparison

Aspect	PostgreSQL (Before)	DuckDB + Parquet (After)
System Packages	8+ packages including PostgreSQL	5 basic packages
Database Setup	Manual configuration, users, permissions	Automatic initialization
Backup Method	`pg_dump` + application files	Simple directory copy
Migration	Database dump/restore + files	Copy entire directory
Security	Database passwords, network config	File permissions only
Dependencies	External database service	Self-contained

This deployment guide provides a comprehensive setup for production deployment of Research made Readable on an external VM with the new simplified DuckDB/Parquet architecture. The deployment process is now significantly simpler and more reliable than the previous PostgreSQL-based setup.