Complete Script Collection - AI Platform

📦 All Scripts Overview

Script	Purpose	Category	Priority
`setup.sh`	Create directory structure	Setup	Optional
`check.sh`	Verify file structure	Verification	Required
`check-db.sh`	Verify database initialization	Verification	Required
`deploy.sh`	**Master deployment script**	**Deployment**	⭐ **Main**
`init-db-manual.sh`	Manual database initialization	Database	As needed
`backup.sh`	Create database backup	Backup	Required
`restore.sh`	Restore from backup	Restore	Required
`rollback.sh`	Quick rollback to latest	Restore	Required
`list-backups.sh`	View all backups	Management	Useful
`cleanup-backups.sh`	Clean old backups	Management	Useful

🚀 Quick Start Guide

One-Time Setup

```bash
# 1. Create directory structure
./setup.sh

# 2. Copy all code files from artifacts
# (See "Artifact to File Mapping Guide")

# 3. Make all scripts executable
chmod +x *.sh

# 4. Verify file structure
./check.sh

# 5. Deploy everything!
./deploy.sh
```

That's it! The `deploy.sh` script handles everything automatically.

📋 Script Dependency Map

```
deploy.sh (Master Script)
  ├──→ check.sh (Verifies files)
  ├──→ docker-compose up (Starts containers)
  ├──→ check-db.sh (Verifies database)
  └──→ init-db-manual.sh (If DB not initialized)

backup.sh
  └──→ Creates timestamped backup

restore.sh
  ├──→ backup.sh (Creates safety backup first)
  └──→ Restores selected backup

rollback.sh
  └──→ restore.sh (Calls with latest backup)
```

🎯 Common Workflows

Fresh Installation

```bash
chmod +x *.sh
./deploy.sh
```

Daily Operations

```bash
# Create backup before changes
./backup.sh

# Make your changes...

# If something goes wrong
./rollback.sh
```

Regular Maintenance

```bash
# View logs
docker-compose logs -f

# Check database health
./check-db.sh

# List backups
./list-backups.sh

# Clean old backups (weekly)
./cleanup-backups.sh
```

Disaster Recovery

```bash
# Stop everything
docker-compose down -v

# Redeploy
./deploy.sh

# Restore from backup
./restore.sh
```

📝 Script Details

1. setup.sh (Optional)

Purpose: Create directory structure
When to use: First time, before copying files
Output: Creates all necessary folders

```bash
./setup.sh
```

2. check.sh (Required)

Purpose: Verify all files exist
When to use: Before deployment
Output: Lists missing files if any

```bash
./check.sh
```

Example output:

```
✓ docker-compose.yml
✓ backend/Dockerfile
✓ backend/package.json
...
✅ All files and folders are present!
```

3. deploy.sh (⭐ Main Script)

Purpose: Complete deployment automation
When to use: Initial setup, redeployment
What it does:

- Pre-flight checks (Docker installed?)
- File verification (runs check.sh)
- Environment setup (creates .env)
- Container management (stops/starts)
- Database initialization (runs check-db.sh)
- Health checks (tests API)
- Shows next steps

```
bash

./deploy.sh
```

Interactive prompts:

- Create .env? (if missing)
- Stop existing containers?
- Remove volumes? (data loss warning)
- Initialize database manually? (if needed)

4. check-db.sh (Required)

Purpose: Verify database tables and data
When to use: After deployment, troubleshooting
Output: Shows which tables exist

```
bash

./check-db.sh
```

Example output:

```
✓ PostgreSQL container is running
✓ Table 'users' exists
✓ Table 'ai_providers' exists
...
✅ Database is properly initialized!
📋 AI Providers in database: 4
```

5. init-db-manual.sh (As Needed)

Purpose: Manually run init.sql
When to use: If auto-initialization failed
Output: Executes SQL and verifies

```
bash

./init-db-manual.sh
```

6. backup.sh (Required)

Purpose: Create compressed database backup
When to use:

- Before major changes
- Daily (automated)
- Before updates

Features:

- Timestamped files
- Automatic compression
- Auto-cleanup old backups
- Shows backup size

```
bash

./backup.sh
```

Creates: `backups/backup_20241205_143022.sql.gz`

7. restore.sh (Required)

Purpose: Restore database from backup
When to use: Recovery, data migration
Safety: Creates safety backup first

```bash
# Interactive (choose from list)
./restore.sh

# Direct (specify file)
./restore.sh ./backups/backup_20241205_143022.sql.gz
```

Features:

- Interactive backup selection
- Safety backup before restore
- Data loss warnings
- Progress indicators
- Verification after restore

8. rollback.sh (Required)

Purpose: Emergency rollback to latest backup
When to use: Quick recovery needed
Difference: No choosing, uses latest automatically

```bash
./rollback.sh
```

9. list-backups.sh (Useful)

Purpose: View all available backups
When to use: Before restore, checking space
Output: Table of backups with size and date

```bash
./list-backups.sh
```

Shows:

- Total backup count
- Total disk usage
- Regular backups
- Safety backups (pre-restore)
- Latest backup highlighted

10. cleanup-backups.sh (Useful)

Purpose: Remove old backups to free space
When to use: Disk space management
Features:

- Configurable retention
- Shows files before deletion
- Confirmation required

```bash
./cleanup-backups.sh
```

Prompts:

- How many to keep?
- Delete old backups?
- Delete safety backups?

🔧 Configuration Variables

All Scripts

```bash
# Colors (can customize)
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
```

Backup Scripts

```bash
# In backup.sh
BACKUP_DIR="./backups"      # Where to store backups
KEEP_BACKUPS=10             # How many to keep
```

Database Scripts

```bash
DB_USER="aiplatform"
DB_NAME="aiplatform"
```

🎨 Output Examples

Success Message

✓ PostgreSQL is running
✓ Backup created successfully
✓ All files present

Warning Message

⚠️ WARNING: This will OVERWRITE the current database!
⚠️ Some files are missing

Error Message

✗ PostgreSQL container is not running
✗ Backup file not found

Section Header

Database Backup

🔒 Security Best Practices

Script Permissions

```bash
# Scripts should be executable by owner only
chmod 700 *.sh

# Or if multiple users need access
chmod 750 *.sh
```

Backup Security

```bash
# Restrict backup directory
chmod 700 backups/
chmod 600 backups/*.gz

# Encrypt backups (optional)
gpg --symmetric backups/backup_*.sql.gz
```

Environment File

```bash
# Protect .env file
chmod 600 backend/.env

# Never commit to git
echo "backend/.env" >> .gitignore
```

Automation Examples

Cron Jobs

```bash
# Edit crontab
crontab -e

# Add jobs
0 2 * * * cd /path/to/ai-platform && ./backup.sh
0 3 * * 0 cd /path/to/ai-platform && ./cleanup-backups.sh
```

Systemd Timer

```bash
# Create timer for daily backup
sudo nano /etc/systemd/system/ai-platform-backup.timer

[Unit]
Description=AI Platform Daily Backup

[Timer]
OnCalendar=daily
OnCalendar=02:00

[Install]
WantedBy=timers.target
```

CI/CD Integration

```yaml
# GitHub Actions example
- name: Backup Database
  run: |
    cd ai-platform
    ./backup.sh

- name: Deploy Changes
  run: |
    cd ai-platform
    ./deploy.sh

- name: Rollback on Failure
  if: failure()
  run: |
    cd ai-platform
    ./rollback.sh
```

📊 Monitoring & Logging

Check Script Success

```bash
# All scripts use exit codes
./backup.sh
if [ $? -eq 0 ]; then
    echo "Backup successful"
else
    echo "Backup failed" | mail -s "Backup Failed" admin@example.com
fi
```

Log Output

```bash
# Redirect to log file
./backup.sh >> /var/log/ai-platform-backup.log 2>&1

# Rotate logs
logrotate /etc/logrotate.d/ai-platform
```

Monitor Backup Size

```bash
# Check backup growth
du -sh backups/

# Alert if too large
BACKUP_SIZE=$(du -s backups/ | cut -f1)
if [ $BACKUP_SIZE -gt 10000000 ]; then
    echo "Backups exceed 10GB" | mail -s "Large Backups" admin@example.com
fi
```

🐛 Troubleshooting

"Permission denied"

```bash
chmod +x *.sh
```

"Docker not found"

```bash
# Install Docker
curl -fsSL https://get.docker.com | sh
```

"Database connection failed"

```bash
# Check if container is running
docker-compose ps postgres

# Check logs
docker-compose logs postgres
```

"Backup failed"

```bash
# Check disk space
df -h

# Check permissions
ls -la backups/
```

🎓 Learning the Scripts

Beginner Path

1. Run `./check.sh` to understand file structure
2. Run `./deploy.sh` for automated setup
3. Try `./backup.sh` and `./list-backups.sh`
4. Read the output messages carefully

Advanced Path

1. Open each script in a text editor
2. Read the comments
3. Understand the functions
4. Customize for your needs
5. Add your own features

Script Template

All scripts follow this pattern:

```bash
#!/bin/bash
# Colors
# Functions (print_header, print_success, etc.)
# Configuration variables
# Main logic
# Exit with code
```

📚 Further Reading

- PostgreSQL Backup Documentation
- Docker Compose Documentation
- Bash Scripting Guide
- Database Initialization Guide (in artifacts)
- Backup & Restore System Guide (in artifacts)

✅ Checklist

- [ ] All scripts created and executable
- [ ] deploy.sh runs successfully
- [ ] Database initialized and verified
- [ ] First backup created
- [ ] Restore tested (in test environment)
- [ ] Cron jobs configured (optional)
- [ ] Documentation reviewed
- [ ] Team trained on procedures