Lokale Git Umgebung

Repository Struktur: c:\freeding\tbot052025\
├── .git/ (Git Metadaten)
├── notebooks/
│ └── Lokale Python-Umgebung.ipynb ✅
├── src/
│ ├── trading_bot.py
│ ├── strategies/
│ └── utils/
├── configs/
│ └── .env.example
├── requirements.txt
├── README.md
└── .gitignore

```
# Git Status prüfen
git status
# Änderungen hinzufügen
git add .
# Commit erstellen
git commit -m "Update trading strategy"
# Zum Server pushen
git push origin main
```

Server Git Integration

Server Repository: /home/trading/ada-trading/
├── .git/ (Remote Repository)
├── python_bot/
│ └── src/ (von Git synchronisiert)
├── r_analysis/
│ └── strategies/ (von Git synchronisiert)
├── shared_data/ (nicht in Git)
├── logs/ (nicht in Git)
└── backups/ (nicht in Git)

```
# Aktuelle Änderungen holen
git pull origin main
# Service neustarten
sudo systemctl restart ada-trading-bot
# Deployment Status
git log --oneline -5
```

Kompletter Git Workflow

1

Lokale Entwicklung

Code in VS Code/Jupyter schreiben
Lokale Tests durchführen
Bitget API Testing

→

2

Git Add & Commit

Änderungen hinzufügen
Aussagekräftige Commit-Message
Lokale Versionskontrolle

→

3

Push zu Remote

Code zum Repository pushen
GitHub/GitLab Synchronisation
Backup in der Cloud

→

4

Server Deployment

SSH zum Server
Git Pull ausführen
Services neustarten

Initial Setup (Einmalig)

```
# Git Repository initialisieren
cd c:\freeding\tbot052025\
git init
git remote add origin https://
github.com/username/ada-trading-bot.git
# .gitignore erstellen
echo "*.log" > .gitignore
echo ".env" >> .gitignore
echo "__pycache__/" >> .gitignore
echo "venv/" >> .gitignore
# Erste Commit
git add .
git commit -m "Initial ADA trading bot
setup"
git push -u origin main
```

Täglicher Workflow

```
# Aktuellen Status prüfen
git status
git diff
# Änderungen committen
git add src/trading_bot.py
git commit -m "Improve RSI signal
logic"
# Notebook hinzufügen
git add notebooks/
backtest_analysis.ipynb
git commit -m "Add backtesting
analysis"
# Alles pushen
git push origin main
```

Server Deployment

```
# SSH zum Server
ssh trading@91.99.11.170 -p 2222
# Zum Repository Verzeichnis
cd ~/ada-trading/
# Updates holen
git pull origin main
# Bot neustarten
sudo systemctl restart ada-trading-bot
sudo systemctl status ada-trading-bot
# Logs prüfen
tail -f logs/python_bot/bot.log
```

Git History & Debugging

```
# Commit History anzeigen
git log --oneline -10
git log --graph --oneline
# Änderungen zwischen Commits
git diff HEAD~1 HEAD
# Bestimmte Datei wiederherstellen
git checkout HEAD~1 -- src/
trading_bot.py
# Branch erstellen für neue Features
git checkout -b feature/new-strategy
git checkout main
git merge feature/new-strategy
```

Praktische Git Szenarien

Szenario 1: Neue Trading-Strategie entwickeln

- Lokale Entwicklung in Jupyter Notebook
- Strategie in Python Bot implementieren

- Lokale Tests mit Bitget Sandbox
- Git Add + Commit mit aussagekräftiger Message
- Push zur Remote Repository
- SSH zum Server, Git Pull ausführen
- Trading Bot Service neu starten
- Live-Performance überwachen

🚨 Szenario 2: Hotfix für kritischen Bug

- Bug im Live-System identifizieren
- Sofortiger Fix im lokalen Code
- Schneller Commit: "Hotfix: Fix order execution bug"
- Sofortiger Push zur Remote
- SSH zum Server
- Git Pull + Service Restart
- Überwachung der Lösung

📊 Szenario 3: R-Analyse Ergebnisse integrieren

- R-Analyse in Positron IDE durchführen
- Signale in shared_data/ speichern
- Python Bot für neue Signale anpassen
- Beide R-Scripts und Python-Code committen

- Integration auf Server deployen
- R-Analyse via Cron Job automatisieren

💡 Git Best Practices für Trading Bot

📝 Aussagekräftige Commits

"Add RSI oversold signal"
"Fix position sizing calculation"
"Update Bitget API endpoints"

🔒 Sensitive Daten ausschließen

API Keys in .env files
.gitignore für credentials
Logs nicht versionieren

🎯 Feature Branches

Neue Strategien in separaten Branches
Testing vor Merge zu main
Production bleibt stabil

⏰ Regelmäßige Commits

Kleine, häufige Commits
Ende jeder Session committen
Backup durch Git History

🔄 Automated Deployment

Git Hooks für auto-deployment
Service restart nach Pull
Rollback bei Fehlern

📋 Code Review

Pull Requests bei Teams
Code-Qualität prüfen
Trading-Logic validieren

⚠️ Häufige Git Probleme & Lösungen

Problem: "Permission denied (publickey)"
Lösung: SSH Keys prüfen: `ssh-add ~/.ssh/id_rsa`

Problem: Merge Konflikte
Lösung: `git status` → Konflikte manuell lösen → `git add .` → `git commit`

Problem: Versehentlich .env committed
Lösung: `git rm --cached .env` → `.env` zu .gitignore hinzufügen

🎯 Aktuelle Git Integration Status

Repository: c:\freeding\tbot052025\	**Remote:** GitHub/GitLab Ready	**Sync Status:** ✅ Lokal ↔ Server
Notebook: ✅ Versioniert	**Deployment:** Git Pull → Auto-Restart	**Backup:** Git History + Cloud