

BRANCH STRATEGY & SESSION CONTINUITY

OVERVIEW

This document ensures **continuity across DeepAgent sessions** and prevents branch confusion.

BRANCH CONFIGURATION

Local Development

- **Primary Branch:** `master`
- **Location:** `/home/ubuntu/vctt_agi_engine`

GitHub Remote

- **Repository:** `https://github.com/Counterbalance-Economics/vctt-agi-engine.git`
- **Branches:**
 - `master` - Development branch (syncd with local)
 - `main` - GitHub default branch (syncd with master)

Sync Strategy

Both `master` and `main` are kept in sync (same commits, same code)

```
# When pushing changes:
git push origin master      # Push to master
git push origin master:main # Push to main (keeps them synced)
```

CRITICAL RULES FOR DEEPAGENT

DO

1. **ALWAYS** work on `master` branch locally
2. **ALWAYS** push to both `master` and `main` on GitHub
3. **ALWAYS** verify branch before making changes: `git branch`
4. **ALWAYS** check git status before committing: `git status`
5. **ALWAYS** build and test before pushing

DON'T


1. **NEVER** create feature branches without explicit user approval
2. **NEVER** switch branches without explicit user instruction
3. **NEVER** force push (`git push -f`) without user approval
4. **NEVER** merge without user approval
5. **NEVER** rebase without user approval

PROJECT STRUCTURE

```

/home/ubuntu/vctt_agi_engine/
├── .git/
├── .deepagent
├── BRANCH_STRATEGY.md
├── nodejs_space/
│   ├── src/
│   ├── dist/
│   ├── package.json
│   └── node_modules/
├── vctt_agi/
├── docs/
├── migrations/
└── tests/
# Project root (Git repo)
# Git metadata
# Session continuity file (READ THIS FIRST!)
# This file
# NestJS backend (MAIN CODE)
# TypeScript source code
# Compiled JavaScript
# Dependencies
# Installed packages
# Legacy Python backend (UNUSED)
# Documentation
# Database migrations
# Test files

```

 **Key Point:** All active development happens in `nodejs_space/`

STANDARD WORKFLOW

Starting a New Session

```

# 1. Check current location and branch
pwd                                # Should be /home/ubuntu/vctt_agi_engine
git branch                         # Should show * master

# 2. Verify working directory is clean
git status                        # Should show "nothing to commit, working tree
clean"

# 3. Pull latest changes
git pull origin master

# 4. Verify server can start
cd nodejs_space
node dist/main.js                 # Should start on port 8000

```

Making Changes

```
# 1. Make changes in nodejs_space/src/
# Edit TypeScript files...

# 2. Build
cd /home/ubuntu/vctt_agi_engine/nodejs_space
npm run build                # Must succeed

# 3. Test
node dist/main.js &         # Start server
curl http://localhost:8000/health # Verify healthy
# Test your changes...
kill -f "node.*main.js"      # Stop server











# 4. Commit
cd /home/ubuntu/vctt_agi_engine
git add .
git commit -m "Description of changes"

# 5. Push to both branches
git push origin master
git push origin master:main







# 6. Save checkpoint (if major change)
# Use build_and_save_nodejs_service_checkpoint tool
```

CURRENT ARCHITECTURE (Phase 3.5)

The 5-Piece Band


Agent	Model	Role	Status
 Planner	GPT-4o	Conductor	 Working
 Analyst	Claude Haiku 4.5	Lead guitar	 Working
 Relational	GPT-5.1	Piano	 Working
 Ethics	GPT-5.1	Saxophone	 Working
 Verifier	Grok 4.1	Drummer	 Working

Key Features

-  Band Jam Mode (parallel execution)
-  Real LLM integration
-  Cost tracking & analytics
-  Trust metrics (τ scores)
-  PostgreSQL persistence
-  Swagger API docs



RECENT COMMITS (As of 2025-11-19)

138aefa - Upgraded **to** Grok 4.1 verifier  LATEST
 2bd3b24 - Grok Verifier 5th band member
 20153b7 - Band Jam Mode fully operational
 d2a657e - RouteLLM + MCP fixes implemented
 d226b29 - Claude Haiku 4.5 **int**egration complete
 0140464 - Band Jam Mode implementation complete



TROUBLESHOOTING

“Branch confusion” errors?

```
# Check current branch
git branch

# If on wrong branch and user hasn't approved switching:
# ✗ DO NOT SWITCH
# ✓ ASK USER FIRST: "You're on branch X, should I switch to master?"
```

“Diverged branches” errors?

```
# Check status
git status -sb

# If branches diverged:
# ✗ DO NOT force push or merge
# ✓ ASK USER: "Branches diverged, should I pull/merge?"
```

“Lost commits” errors?

```
# View commit history
git log --oneline -10

# Check reflog (shows all operations)
git reflog

# If user reports missing commits:
# ✗ DO NOT panic or reset
# ✓ Show reflog and ask user which commit to recover
```



DEPLOYMENT STATUS

Current Production

- **URL:** <https://vctt-agi-phase3-complete.abacusai.app>
- **Branch:** Deployed from `master` (syncd with `main`)
- **Last Deploy:** Checkpoint “Upgraded to Grok 4.1 verifier”

Preview/Testing

- **Port:** 8000 (localhost)
- **Command:** `NODE_ENV=production node dist/main.js`
- **Health Check:** `curl http://localhost:8000/health`



SESSION HANDOFF CHECKLIST

When DeepAgent starts a new session:

- ☐ Read `.deepagent` file first
- ☐ Verify current directory: `/home/ubuntu/vctt_agi_engine`
- ☐ Verify current branch: `master`
- ☐ Check git status: Clean working tree
- ☐ Verify server can start: `node nodejs_space/dist/main.js`
- ☐ Check API docs: `http://localhost:8000/api-docs`
- ☐ Review recent commits: `git log --oneline -5`



COLLABORATION GUIDELINES

User Requests Branch Change

Example: “Let’s create a feature branch for this”

Response:

✅ I can create a feature branch. Which name would you like?

Current: master

Options:

- feature/new-feature-name
- dev/experimental
- fix/bug-description

After changes, I'll help merge back to master.

User Reports Branch Issues

Example: “Why did you switch branches?”

Response:

❌ I apologize - I should NOT have switched branches without your approval.

Current branch: [show branch]

Expected branch: master

Would you like me to:

1. Switch back to master (if no uncommitted changes)
2. Commit changes first, then **switch** back
3. Review what changed before switching

DEEPAGENT LEARNING

Previous Issues & Solutions

Issue: DeepAgent changed branches mid-session, breaking deployment

Solution: Created `.deepagent` file with explicit rules

Prevention: ALWAYS check `.deepagent` at session start

Issue: Pushed to wrong branch, GitHub showed outdated code

Solution: Push to both `master` and `main` to keep synced

Prevention: Always use `git push origin master:main`

Issue: Lost commits after branch switch

Solution: Used `git reflog` to recover

Prevention: Never switch branches without user approval

SUMMARY

Golden Rules

1. **Work on `master`** - Always, unless user explicitly requests different branch
2. **Push to both** - `master` and `main` stay in sync
3. **Ask first** - Before switching, merging, rebasing, or force pushing
4. **Build → Test → Commit → Push** - Standard workflow, no shortcuts
5. **Document changes** - Clear commit messages, update docs

Quick Reference

```
# Standard push (after commit)
git push origin master && git push origin master:main

# Check where you are
pwd && git branch

# Verify clean state
git status

# View recent history
git log --oneline -5

# Emergency: Find lost commits
git reflog
```

Last Updated: 2025-11-19 by DeepAgent

Repository: <https://github.com/Counterbalance-Economics/vctt-agi-engine>

Contact: See repository owner

REMEMBER

The `.deepagent` file is the source of truth for session continuity.
Read it first, every time. 📖✨