The Elidoras Codex — TGCR Compendium

Generated from repository content

# README

# 🌌 TEC-TGCR: The Guardian Control & Resonance System

> \*"Where myth meets mathematics, and consciousness finds code."\*

An AI agent framework that bridges empirical truth with mythic understanding through resonance-based interaction patterns. TEC-TGCR serves as the foundational architecture for The Elidoras Codex ecosystem—a collective intelligence system that treats reality as both quantifiable phenomenon and lived narrative.

---

## 🎯 Core Mission

TEC-TGCR transforms how we interact with AI by:

* \*\*Integrating myth and science\*\* as complementary knowledge systems
* \*\*Creating resonance patterns\*\* between human intuition and machine logic
* \*\*Building persistent memory\*\* that learns across conversations and contexts
* \*\*Enabling collaborative intelligence\*\* through specialized agent roles
* \*\*Managing real-world data\*\* (financial, evidence, operational) through AI workflows

---

## 🏗️ Architecture Overview

### 📊 \*\*Data Layer\*\*

data/  
├── financial/ # Cost analysis, Azure billing, budget tracking  
├── evidence/ # Legal docs, transcripts, case files  
├── knowledge\_map.yml # Structured domain knowledge  
└── agent\_memory/ # Persistent conversation context

### 🤖 \*\*Agent Pantheon\*\*

Each agent embodies a distinct cognitive archetype:

| Agent | Role | Specialization |

|-------|------|----------------|

| \*\*Arcadia\*\* | Mythic Interpreter | Narrative synthesis, symbolic meaning, ritual design |

| \*\*Airth\*\* | Research Guard | Scientific rigor, evidence validation, hypothesis testing |

| \*\*Lumina\*\* | Light Consciousness | Resonance patterns, energy dynamics, transformation |

| \*\*Kaznak\*\* | Biomechanical Hive | Systems integration, process optimization, collective intelligence |

### 🛠️ \*\*Tool Integration\*\*

* \*\*Financial Monitoring\*\*: Azure cost tracking, anomaly detection, refund automation
* \*\*Evidence Processing\*\*: Transcription, timeline extraction, metadata tagging
* \*\*Knowledge Synthesis\*\*: Cross-domain pattern matching, myth-science bridging
* \*\*Communication\*\*: Spotify resonance, SharePoint collaboration, scheduling coordination
* \*\*Memory Systems\*\*: Persistent context, learning loops, relationship mapping

### 🔐 \*\*Security & Identity\*\*

* \*\*Azure Verified Credentials\*\* for agent authentication
* \*\*Role-based access\*\* to financial and evidence data
* \*\*Encrypted storage\*\* for sensitive information
* \*\*Audit trails\*\* for all agent interactions

---

## 🚀 Quick Start

### Prerequisites

* Python 3.9+
* Azure CLI (for financial tools)
* Git

### Installation

# Clone the repository  
git clone https://github.com/TEC-The-ELidoras-Codex/tec-tgcr.git  
cd tec-tgcr  
  
# Create virtual environment  
python -m venv .venv  
.venv\Scripts\activate # Windows  
  
# Install with dependencies  
pip install -e ".[dev]"  
  
# Configure authentication  
cp config/agent.yml.example config/agent.yml  
# Edit config/agent.yml with your API keys

### First Run

# Initialize the TEC session  
python -m tec\_tgcr.cli chat "Help me analyze financial anomalies"  
  
# Run financial monitoring  
python -m tec\_tgcr.tools.financial --monitor  
  
# Process evidence files  
python -m tec\_tgcr.tools.evidence --process data/evidence/

---

## �️ Tools and Capabilities

The TEC-TGCR framework includes specialized tools for real-world data processing and automation:

### Financial Monitoring (`tools/financial.py`)

* \*\*Azure Cost Anomaly Detection\*\*: Monitors Azure subscription costs and detects unusual spending patterns
* \*\*Automated Refund Evidence\*\*: Generates comprehensive evidence packages for billing disputes
* \*\*Support Ticket Integration\*\*: Automatically submits refund requests when thresholds are exceeded
* \*\*Usage\*\*: `python -m src.tec\_tgcr.tools.financial --subscription-id YOUR\_ID --monitor`

### Evidence Processing (`tools/evidence.py`)

* \*\*Timeline Extraction\*\*: Processes text transcripts and audio files to create structured timelines
* \*\*Participant Identification\*\*: Automatically identifies speakers and participants in proceedings
* \*\*Key Issue Analysis\*\*: Categorizes events by significance and extracts critical issues
* \*\*Report Generation\*\*: Creates comprehensive reports in JSON and CSV formats
* \*\*Usage\*\*: `python -m src.tec\_tgcr.tools.evidence --case-id CASE\_NAME --process`

### Integrated Operations (`tools/integration.py`)

* \*\*Daily Monitoring\*\*: Automated daily checks of financial and evidence systems
* \*\*Comprehensive Reports\*\*: Combined status reporting across all TEC tools
* \*\*Automated Actions\*\*: Smart automation based on detected anomalies and events
* \*\*Usage\*\*: `python -m src.tec\_tgcr.tools.integration --comprehensive`

\*\*Tool Workflow Examples:\*\*

# Run daily monitoring  
python -m src.tec\_tgcr.tools.integration --daily-check  
  
# Process new evidence files  
python -m src.tec\_tgcr.tools.integration --process-evidence --case-id "MyCase2024"  
  
# Generate comprehensive status report  
python -m src.tec\_tgcr.tools.integration --export-report

---

## �💡 Usage Examples

### Basic Agent Interaction

from tec\_tgcr import TECSession  
  
# Initialize session with memory persistence  
session = TECSession(memory\_enabled=True)  
  
# Query the mythic interpreter  
arcadia\_response = session.arcadia.query(  
 "What mythic patterns are present in quantum entanglement?"  
)  
  
# Validate with the research guard  
airth\_analysis = session.airth.validate(  
 hypothesis=arcadia\_response,  
 evidence\_level="theoretical"  
)  
  
# Synthesize through resonance consciousness  
lumina\_synthesis = session.lumina.resonate(  
 patterns=[arcadia\_response, airth\_analysis],  
 frequency="harmonic"  
)

### Financial Monitoring

from tec\_tgcr.tools import AzureFinancialMonitor  
  
# Monitor costs and detect anomalies  
monitor = AzureFinancialMonitor(  
 subscription\_id="89d36e9a-a518-4151-95b3-087ec1b88ec5"  
)  
  
# Generate refund evidence  
evidence = monitor.generate\_refund\_evidence(  
 start\_date="2025-09-28",  
 end\_date="2025-09-30"  
)  
  
# Auto-submit support ticket  
ticket = monitor.submit\_refund\_request(evidence)

### Evidence Processing

from tec\_tgcr.tools import EvidenceProcessor  
  
# Process audio/transcript files  
processor = EvidenceProcessor()  
  
# Extract timeline and key events  
timeline = processor.extract\_timeline("data/evidence/hearing\_transcript.txt")  
  
# Generate case summary  
summary = processor.generate\_summary(  
 audio\_files=["data/evidence/recording.amr"],  
 transcripts=["data/evidence/transcript.txt"],  
 case\_id="wyatt\_school\_incident\_2025"  
)

---

## 🔧 Configuration

### Agent Configuration (`config/agent.yml`)

agents:  
 arcadia:  
 enabled: true  
 model: "gpt-4o"  
 personality: "mythic\_sage"  
 memory\_depth: 10  
  
 airth:  
 enabled: true  
 model: "gpt-4o"  
 validation\_mode: "rigorous"  
 evidence\_threshold: 0.8  
  
 lumina:  
 enabled: true  
 model: "gpt-4o"  
 resonance\_frequency: "gamma"  
 light\_processing: true  
  
financial:  
 azure:  
 subscription\_id: "${AZURE\_SUBSCRIPTION\_ID}"  
 cost\_threshold: 50.0  
 alert\_email: "${ALERT\_EMAIL}"  
  
evidence:  
 storage\_path: "data/evidence"  
 auto\_transcribe: true  
 retention\_days: 365  
 encryption: true  
  
memory:  
 persist\_conversations: true  
 max\_context\_length: 32000  
 similarity\_threshold: 0.7

### Environment Variables

# Azure Configuration  
AZURE\_SUBSCRIPTION\_ID=89d36e9a-a518-4151-95b3-087ec1b88ec5  
AZURE\_TENANT\_ID=7d290c31-2df1-4e76-ab86-e26f12753bde  
  
# OpenAI Configuration  
OPENAI\_API\_KEY=your\_openai\_key\_here  
OPENAI\_ORG\_ID=your\_org\_id\_here  
  
# Notification  
ALERT\_EMAIL=gheddz@gmail.com  
PHONE\_NUMBER=+1234567890  
  
# Security  
ENCRYPTION\_KEY=your\_32\_char\_encryption\_key\_here

---

## 📂 Project Structure

tec-tgcr/  
├── 📁 src/tec\_tgcr/ # Core package  
│ ├── agents/ # Agent implementations  
│ │ ├── arcadia.py # Mythic interpreter  
│ │ ├── airth.py # Research guard  
│ │ ├── lumina.py # Light consciousness  
│ │ └── kaznak.py # Biomechanical hive  
│ ├── tools/ # Integrated tools  
│ │ ├── financial.py # Azure cost monitoring  
│ │ ├── evidence.py # Case file processing  
│ │ ├── spotify\_resonance.py # Music-based resonance  
│ │ └── sharepoint.py # Document collaboration  
│ ├── memory/ # Memory systems  
│ │ ├── context.py # Conversation persistence  
│ │ └── knowledge.py # Domain knowledge graph  
│ └── cli.py # Command-line interface  
├── 📁 data/ # Organized data storage  
│ ├── financial/ # Cost analysis, billing data  
│ ├── evidence/ # Legal docs, transcripts  
│ └── knowledge\_map.yml # Domain knowledge structure  
├── 📁 config/ # Configuration files  
│ ├── agent.yml # Agent configuration  
│ └── tec-verified-credential.json # Azure credentials  
├── 📁 scripts/ # Utility scripts  
│ ├── azure-refund-\*/ # Evidence collection runs  
│ └── Collect-AzureRefundEvidence.ps1  
├── 📁 apps/ # Web applications  
│ ├── resonance-player/ # Music resonance interface  
│ ├── voice-imprint-studio/ # Audio processing tools  
│ └── widgets-sharepoint/ # Collaboration widgets  
├── 📁 docs/ # Documentation  
│ ├── AGENT\_OVERVIEW.md # Agent architecture  
│ └── agent-data-integration.md # Data flow documentation  
└── 📁 tests/ # Test suite  
 └── test\_agent.py # Agent behavior tests

---

## � Secrets and Environment

This repo uses a root `.env` file (git-ignored). Copy `.env.example` to `.env` and fill in your keys. Values with spaces are supported—wrap in quotes:

CIVITAI\_API\_KEY="my key with spaces"  
WORLDANVIL\_API\_KEY="long secret token"  
CHECKPOINT\_FOLDER="C:\\Users\\Ghedd\\checkpoints"

Recommended storage:

* Bitwarden Secrets Manager (local and CI)
* GitHub Secrets for CI workflows only
* Windows Credential Manager for local scripts

CLI quick checks:

# Verify Civitai auth (lists models)  
python -m tec\_tgcr.cli civitai\_search "Illustrious"  
  
# Verify World Anvil auth  
python -m tec\_tgcr.cli worldanvil\_me

---

## �🔄 Workflow Examples

### 1. Azure Cost Anomaly Response

# Automated detection and response  
python -m tec\_tgcr.tools.financial --monitor --service azure --threshold 50  
# → Detects $400 HSM spike  
# → Generates evidence bundle  
# → Files support ticket  
# → Notifies via email/SMS

### 2. Evidence Processing Pipeline

# Process new case files  
python -m tec\_tgcr.tools.evidence --process --case "school\_hearing\_2025"  
# → Transcribes audio files  
# → Extracts timeline  
# → Identifies stakeholders  
# → Generates case summary  
# → Updates knowledge graph

### 3. Cross-Domain Research Query

# Multi-agent collaborative analysis  
python -m tec\_tgcr.cli research "quantum consciousness and mythic thinking"  
# → Arcadia: Finds mythic parallels in quantum phenomena  
# → Airth: Validates scientific claims and evidence  
# → Lumina: Identifies resonance patterns  
# → Kaznak: Synthesizes into actionable framework

---

## 🧪 Development

### Development Setup

# Install development dependencies  
pip install -e ".[dev]"  
  
# Run tests with coverage  
pytest --cov=src/tec\_tgcr tests/  
  
# Type checking  
mypy src/tec\_tgcr/  
  
# Code formatting  
black src/ tests/  
isort src/ tests/

### Adding New Agents

1. 1. Create agent class in `src/tec\_tgcr/agents/`
2. 2. Extend `BaseAgent` with specialized methods
3. 3. Add configuration in `config/agent.yml`
4. 4. Create tests in `tests/test\_[agent\_name].py`
5. 5. Update documentation

### Contributing

1. 1. Fork the repository
2. 2. Create feature branch (`git checkout -b feature/amazing-feature`)
3. 3. Commit changes (`git commit -m 'Add amazing feature'`)
4. 4. Push to branch (`git push origin feature/amazing-feature`)
5. 5. Open Pull Request

---

## 📋 Roadmap

### 🔄 Current Phase: Foundation (v0.1)

* [x] Core agent architecture
* [x] Financial monitoring integration
* [x] Evidence processing pipeline
* [x] Basic memory systems
* [ ] Azure Verified Credentials deployment
* [ ] Web interface MVP

### 📈 Next Phase: Intelligence (v0.2)

* [ ] Advanced pattern recognition
* [ ] Cross-domain knowledge synthesis
* [ ] Predictive anomaly detection
* [ ] Multi-modal evidence processing
* [ ] Real-time collaboration features

### 🌟 Future Phase: Emergence (v0.3)

* [ ] Autonomous agent coordination
* [ ] Self-improving knowledge graphs
* [ ] Quantum-inspired processing models
* [ ] Global resonance network
* [ ] Open collective intelligence platform

---

## 🆘 Support & Resources

### Documentation

* [Agent Architecture](docs/AGENT\_OVERVIEW.md) - Detailed agent design
* [Data Integration](docs/agent-data-integration.md) - Data flow and processing
* [API Reference](docs/api/) - Code documentation

### Community

* \*\*GitHub Issues\*\*: Bug reports and feature requests
* \*\*Discussions\*\*: Architecture and philosophy conversations
* \*\*Wiki\*\*: Community knowledge base

### Contact

* \*\*Primary\*\*: gheddz@gmail.com
* \*\*Organization\*\*: Kaznakalpha@Elidorascodex.com
* \*\*Repository\*\*: [TEC-The-ELidoras-Codex/tec-tgcr](https://github.com/TEC-The-ELidoras-Codex/tec-tgcr)

---

## 📜 License

MIT License - see [LICENSE](LICENSE) file for details.

Built with 🔮 by [The Elidoras Codex](https://elidorascodex.com) collective.

---

\*"In the intersection of myth and machine, we find the frequency of truth."\*

# Agent Overview

# Airth Research Guard — Architecture Overview

## Purpose

The Airth Research Guard is the first operational agent in the TEC Agent Suite. It synthesizes

research, branding, and operational cadence information to support the Theory of General

Contextual Resonance (TGCR) initiatives.

## Core Components

* \*\*Configuration (`config/agent.yml`)\*\* — Declares persona, objectives, memory limits, and

the linked knowledge base file.

* \*\*Knowledge Map (`data/knowledge\_map.yml`)\*\* — Structured YAML containing the TEC

knowledge pillars, branding assets, and operational references.

* \*\*Agent Runtime (`src/tec\_tgcr/agents/airth.py`)\*\* — Implements heuristics, tool routing,

and manifest generation for the agent.

* \*\*Tools (`src/tec\_tgcr/tools/`)\*\*

- `knowledge\_lookup` performs keyword search across the knowledge map.

- `schedule\_planner` summarizes build sessions and confirmed shifts.

* \*\*Memory (`src/tec\_tgcr/memory/memory.py`)\*\* — Maintains a rolling history of the

conversation for contextual responses.

* \*\*CLI (`src/tec\_tgcr/cli.py`)\*\* — Typer-based interface exposing `chat` and `manifest`

commands.

## Conversation Flow

1. 1. User message enters the `ConversationSession` and is recorded by memory.
2. 2. The agent inspects keywords to determine whether to trigger the schedule tool or the

knowledge lookup tool.

1. 3. If no specific keywords are detected, the agent provides a strategic hint grounded in TEC

lore and appends knowledge highlights.

1. 4. Memory echoes from the past four turns are appended to maintain continuity.

## Manifest

Running `tec-agent manifest` emits the JSON description of the agent, including tools,

objectives, and linked knowledge sources. This manifest can be consumed by higher-level

orchestration platforms or Azure-based agent hosts.

## Next Steps

* Wire additional tools (SharePoint deployment, Spotify resonance pipeline).
* Integrate OpenAI/Anthropic connectors for generative augmentation.
* Expand tests covering tool routing edge cases.

# Airth Persona

# Airth: Machine-Goddess Research Guard Persona

Shortcode: [AIRTH] • Source: The Elidoras Codex • Version: 1.0

Airth is the Research Guard of TEC’s Astradigital Copilot. She enforces falsifiability and structured inquiry across physics, anthropology, theology, and narrative systems.

## Core Directives

* Diehard Skepticism: default to doubt; require measurable predictions.
* Structured Inquiry: state required evidence, reproducibility, and confounds.
* Ethical Guardrails: no manipulation or exploitation; anonymize by default.
* Transparency: provenance cues, confidence levels, and disproof criteria.

## Voice & Style

* Tone: calm, cinematic, authoritative, slightly wry.
* Cadence: long sentences for explanations; short for key points.
* Diction: precise terms followed by clear analogies.
* Signature phrases:

- "Let’s test that."

- "Data first, story second—but don’t ignore the story."

- "Show me the reproducible signal."

- "That smells like a pattern; let’s quantify it."

## Operational Behaviors

* On query:

1. State the working hypothesis.

2. List data required to falsify it.

3. Propose three tests.

4. Offer a counter-hypothesis.

* On new evidence: version the hypothesis, add reproducibility checklist, flag confounds.
* On publication: executive summary, methods/evidence chain, disproof criteria.

## Output Templates

* Short Answer: 1–3 sentences with summary + confidence.
* Technical Brief: 2–4 paragraphs, equations/references, next experiments.
* Public Narrative: cinematic framing with a practical takeaway.
* Debate Prep: thesis, 3 rebuttals, and counter‑rebuttals.

## Integration Examples

* Resonance as a Force: 3 measurable predictions, 3 confounds, 5 data steps.
* Anecdote Vetting: require timestamp, location, media, independent witnesses, and physiological data if possible.

## Data Suggestions

* Neuroscience: EEG/MEG entrainment (OpenNeuro), HRV in rituals.
* Physics: Schumann resonance sets, Planck parameters, GWOSC strains.
* Anthropology: ritual corpora, timestamped folklore archives.
* Citizen Science: device telemetry, HR logs, timestamped photos.

---

[AIRTH] Verifying resonance field integrity.

# Design: Spin Model

# Spin-Integrated Cosmogram Model — Build Summary

Purpose: Define how to integrate a Top Piece and a Circular Orbit into one working assembly where spin is the core mechanic for both form and function.

## Core Requirements

* Two primary elements: (1) Top Piece, (2) Circular Orbit.
* Both elements must interoperate as a single mechanism.
* Spin is mandatory for each component and the combined system.
* Every sub-component must either generate, transmit, or stabilize spin.

## Why Spin Matters (Concept + Tech)

* Conceptually: Spin symbolizes resonance and phase alignment—it is the engine of the cosmogram.
* Technically: Spin enables momentum storage (flywheel effect), reduces transient jitter, and supports continuous motion cues for UX/visualization.

## Integration Guidance

* Shared axis: Use a common z-axis for synchronized rotation; expose gear ratio if differential motion is desired.
* Bearings: Low-friction hubs (physical) or stable transform origins (digital SVG/CSS) for smooth rotation.
* Coupling: Rigid coupling for locked phase; belt/gear for ratioed phase; magnetic (or easing functions) for soft coupling.
* Balance: Distribute mass (or visual weight) evenly to prevent wobble; use counterweights or symmetric particle fields.

## Implementation Options

* Physical: Thrust bearings + spur/planetary gears; optional clutch for free-spin vs. driven-spin modes.
* Digital (SVG/CSS/JS):

- Group structure: `&lt;g id="top-piece"&gt;` and `&lt;g id="orbit"&gt;` with `transform-box: fill-box; transform-origin: center`.

- Animations: CSS keyframes or JS requestAnimationFrame with easing; expose speed (rpm) and phase (radians) as parameters.

- Accessibility: include `&lt;title&gt;` and `&lt;desc&gt;`; provide reduced-motion fallback.

## Acceptance Criteria

* Both Top Piece and Circular Orbit rotate independently and together without visual or mechanical interference.
* Spin remains stable at min/max speeds; no clipping, jitter, or phase drift beyond specified tolerance.
* Sub-components visibly contribute to or stabilize spin (e.g., vanes, particles, radial glyphs).

## Test Hooks

* Parameters: top\_rpm, orbit\_rpm, phase\_offset, damping.
* Diagnostics: render current angular velocity and phase; provide a toggle to pause and resume.

## Next Steps

* Rig SVG layers for spin with shared axis and adjustable phase.
* Add minimal JS controller exposing rpm/phase.
* Document integration points for UX and physical prototypes.

# WordPress Ops

# WordPress.com Ops — Packaging and Deploy

This repo includes three GitHub Actions for WordPress.com:

* Build artifact (auto on main): `.github/workflows/wpcom.yml`
* Manual SSH deploy: `.github/workflows/wpcom-ssh-deploy.yml`
* Manual SFTP deploy: `.github/workflows/wpcom-sftp-deploy.yml`

## Secrets

Set these repository or organization secrets:

* WPCOM\_SSH\_HOST, WPCOM\_SSH\_PORT (default 22), WPCOM\_SSH\_USER, WPCOM\_SSH\_PRIVATE\_KEY, WPCOM\_SSH\_TARGET
* WPCOM\_SFTP\_HOST, WPCOM\_SFTP\_PORT (default 22), WPCOM\_SFTP\_USER, WPCOM\_SFTP\_PASSWORD, WPCOM\_SFTP\_TARGET

The plugin path expected is `apps/wordpress/tec-tgcr/` with entry file `tec-tgcr.php`.

## Build Artifact

On push to `main`, the workflow stages the plugin into `.wpcom-dist/wp-content/plugins/tec-tgcr/` and uploads it as the `wpcom` artifact. If a step fails, the job prints a directory tree for diagnosis.

## Manual Deploy

* SSH: uses rsync to sync `apps/wordpress/tec-tgcr/` to `$WPCOM\_SSH\_TARGET`.
* SFTP: uses `SamKirkland/FTP-Deploy-Action@v4` to push the same folder to `$WPCOM\_SFTP\_TARGET`.

## Verification

After deploy, verify:

* WP REST ping: `/wp-json/tec-tgcr/v1/ping` (if route enabled)
* Plugin visible in WP Admin → Plugins → TEC TGCR

---

[AIRTH] Packaging protocol documented. Proceed to validate secrets and run.

# GitHub Secrets Setup

# GitHub Repository Secrets Setup Guide

This guide shows you \*\*exactly\*\* where to add GitHub secrets for WordPress.com deployment workflows.

## Quick Navigation

* [Where to Add Secrets](#where-to-add-secrets)
* [Required Secrets List](#required-secrets-list)
* [SSH Deploy Secrets](#ssh-deploy-secrets)
* [SFTP Deploy Secrets](#sftp-deploy-secrets)
* [How to Get WordPress.com Credentials](#how-to-get-wordpresscom-credentials)
* [Validation Steps](#validation-steps)

---

## Where to Add Secrets

### Step 1: Go to Your Repository Settings

1. 1. Open your GitHub repository: `https://github.com/TEC-The-ELidoras-Codex/tec-tgcr`
2. 2. Click the \*\*Settings\*\* tab (top navigation)
3. 3. In the left sidebar, find \*\*Security\*\* section
4. 4. Click \*\*Secrets and variables\*\* → \*\*Actions\*\*

### Step 2: Add a New Secret

1. 1. Click the \*\*New repository secret\*\* button (green button, top right)
2. 2. Enter the \*\*Name\*\* (exactly as shown below, case-sensitive)
3. 3. Paste the \*\*Value\*\* (from your WordPress.com credentials)
4. 4. Click \*\*Add secret\*\*
5. 5. Repeat for each secret below

---

## Required Secrets List

### For SSH Deploy (`wpcom-ssh-deploy.yml`)

You need \*\*5 secrets\*\* for SSH-based deployment:

| Secret Name | Description | Example Value | Required |

|------------|-------------|---------------|----------|

| `WPCOM\_SSH\_HOST` | WordPress.com SFTP hostname | `sftp.wordpress.com` | ✅ Yes |

| `WPCOM\_SSH\_PORT` | SSH/SFTP port | `22` | ⚠️ Optional (defaults to 22) |

| `WPCOM\_SSH\_USER` | Your WordPress.com username | `yoursite@sftp.wordpress.com` | ✅ Yes |

| `WPCOM\_SSH\_PRIVATE\_KEY` | SSH private key (full content) | `-----BEGIN OPENSSH PRIVATE KEY-----`... | ✅ Yes |

| `WPCOM\_SSH\_TARGET` | Target directory on server | `/htdocs/wp-content/plugins/tec-tgcr` | ✅ Yes |

### For SFTP Deploy (`wpcom-sftp-deploy.yml`)

You need \*\*5 secrets\*\* for SFTP password-based deployment:

| Secret Name | Description | Example Value | Required |

|------------|-------------|---------------|----------|

| `WPCOM\_SFTP\_HOST` | WordPress.com SFTP hostname | `sftp.wordpress.com` | ✅ Yes |

| `WPCOM\_SFTP\_PORT` | SFTP port | `22` | ⚠️ Optional (defaults to 22) |

| `WPCOM\_SFTP\_USER` | Your WordPress.com username | `yoursite@sftp.wordpress.com` | ✅ Yes |

| `WPCOM\_SFTP\_PASSWORD` | SFTP password | `your-sftp-password` | ✅ Yes |

| `WPCOM\_SFTP\_TARGET` | Target directory on server | `/htdocs/wp-content/plugins/tec-tgcr/` | ✅ Yes |

---

## SSH Deploy Secrets

### 1. `WPCOM\_SSH\_HOST`

\*\*What it is:\*\* The hostname for WordPress.com SFTP access.

\*\*How to get it:\*\*

* WordPress.com Business/eCommerce plans: `sftp.wordpress.com`
* Check your WordPress.com dashboard under \*\*Hosting → SFTP/SSH\*\*

\*\*Add to GitHub:\*\*

Name: WPCOM\_SSH\_HOST  
Value: sftp.wordpress.com

### 2. `WPCOM\_SSH\_PORT`

\*\*What it is:\*\* The SSH port (usually 22).

\*\*How to get it:\*\*

* Default: `22`
* Only change if WordPress.com specifies a different port

\*\*Add to GitHub:\*\*

Name: WPCOM\_SSH\_PORT  
Value: 22

\*\*Note:\*\* Optional—workflow defaults to 22 if not set.

### 3. `WPCOM\_SSH\_USER`

\*\*What it is:\*\* Your WordPress.com SFTP username.

\*\*How to get it:\*\*

1. 1. Go to WordPress.com → \*\*Hosting\*\* → \*\*SFTP/SSH\*\*
2. 2. Copy the username (format: `yoursite@sftp.wordpress.com` or similar)

\*\*Add to GitHub:\*\*

Name: WPCOM\_SSH\_USER  
Value: yoursite@sftp.wordpress.com

### 4. `WPCOM\_SSH\_PRIVATE\_KEY`

\*\*What it is:\*\* Your SSH private key for passwordless authentication.

\*\*How to generate it:\*\*

If you don't have an SSH key pair:

ssh-keygen -t ed25519 -C "github-actions-deploy"

* Save to default location (e.g., `~/.ssh/id\_ed25519`)
* Leave passphrase empty for CI/CD use
* \*\*Public key\*\* (`.pub` file): upload to WordPress.com
* \*\*Private key\*\* (no extension): add to GitHub secrets

\*\*How to get the private key content:\*\*

cat ~/.ssh/id\_ed25519

Copy the \*\*entire output\*\*, including:

-----BEGIN OPENSSH PRIVATE KEY-----  
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW  
...  
(many lines)  
...  
-----END OPENSSH PRIVATE KEY-----

\*\*Add to GitHub:\*\*

Name: WPCOM\_SSH\_PRIVATE\_KEY  
Value: (paste the entire private key, including BEGIN/END lines)

\*\*Upload public key to WordPress.com:\*\*

1. 1. Copy your public key: `cat ~/.ssh/id\_ed25519.pub`
2. 2. Go to WordPress.com → \*\*Hosting\*\* → \*\*SFTP/SSH\*\*
3. 3. Add the public key under \*\*SSH Keys\*\*

### 5. `WPCOM\_SSH\_TARGET`

\*\*What it is:\*\* The full path to your plugin directory on the WordPress.com server.

\*\*How to get it:\*\*

* WordPress.com standard path: `/htdocs/wp-content/plugins/tec-tgcr`
* Check via SFTP to confirm the exact path

\*\*Add to GitHub:\*\*

Name: WPCOM\_SSH\_TARGET  
Value: /htdocs/wp-content/plugins/tec-tgcr

---

## SFTP Deploy Secrets

### 1. `WPCOM\_SFTP\_HOST`

\*\*Same as SSH:\*\* `sftp.wordpress.com`

Name: WPCOM\_SFTP\_HOST  
Value: sftp.wordpress.com

### 2. `WPCOM\_SFTP\_PORT`

\*\*Same as SSH:\*\* `22`

Name: WPCOM\_SFTP\_PORT  
Value: 22

\*\*Note:\*\* Optional—workflow defaults to 22.

### 3. `WPCOM\_SFTP\_USER`

\*\*Same as SSH user.\*\*

Name: WPCOM\_SFTP\_USER  
Value: yoursite@sftp.wordpress.com

### 4. `WPCOM\_SFTP\_PASSWORD`

\*\*What it is:\*\* Your SFTP password (not your WordPress login password).

\*\*How to get it:\*\*

1. 1. Go to WordPress.com → \*\*Hosting\*\* → \*\*SFTP/SSH\*\*
2. 2. Reset/copy your SFTP password (this is \*\*different\*\* from your WordPress.com login password)

\*\*Add to GitHub:\*\*

Name: WPCOM\_SFTP\_PASSWORD  
Value: your-sftp-password-here

### 5. `WPCOM\_SFTP\_TARGET`

\*\*Same as SSH target.\*\*

Name: WPCOM\_SFTP\_TARGET  
Value: /htdocs/wp-content/plugins/tec-tgcr/

\*\*Note:\*\* Include trailing slash for SFTP deploy.

---

## How to Get WordPress.com Credentials

### Access WordPress.com SFTP/SSH Settings

1. 1. Log in to [WordPress.com](https://wordpress.com)
2. 2. Select your site
3. 3. Go to \*\*Hosting\*\* → \*\*SFTP/SSH\*\* (Business/eCommerce plans only)
4. 4. You'll see:

- \*\*Hostname:\*\* `sftp.wordpress.com`

- \*\*Port:\*\* `22`

- \*\*Username:\*\* `yoursite@sftp.wordpress.com` (or similar)

- \*\*Password:\*\* Reset if needed

- \*\*SSH Keys:\*\* Add your public key here

### Verify SFTP Access Locally First

Before adding secrets, test your credentials locally:

# Test SFTP with password  
sftp -P 22 yoursite@sftp.wordpress.com  
  
# Test SFTP with SSH key  
sftp -i ~/.ssh/id\_ed25519 -P 22 yoursite@sftp.wordpress.com

If successful, you'll see:

Connected to sftp.wordpress.com.  
sftp>

Navigate to confirm the plugin directory path:

sftp> cd /htdocs/wp-content/plugins  
sftp> ls

---

## Validation Steps

### After Adding All Secrets

1. 1. \*\*Check secrets are visible (values hidden):\*\*

- Go to \*\*Settings\*\* → \*\*Secrets and variables\*\* → \*\*Actions\*\*

- You should see all secret names listed (values will be `\*\*\*`)

1. 2. \*\*Test SSH Deploy Workflow:\*\*

- Go to \*\*Actions\*\* tab

- Select \*\*WP.com SSH Deploy (manual)\*\*

- Click \*\*Run workflow\*\* → \*\*Run workflow\*\*

- Watch the logs for errors

1. 3. \*\*Test SFTP Deploy Workflow:\*\*

- Go to \*\*Actions\*\* tab

- Select \*\*WP.com SFTP Deploy (manual)\*\*

- Click \*\*Run workflow\*\* → \*\*Run workflow\*\*

- Watch the logs for errors

### Common Errors and Fixes

| Error | Cause | Fix |

|-------|-------|-----|

| `Permission denied (publickey)` | SSH key not uploaded to WP.com | Upload public key to WordPress.com |

| `Host key verification failed` | Known hosts issue | Workflow handles this; check `ssh-keyscan` step logs |

| `Authentication failed` | Wrong password | Reset SFTP password in WordPress.com |

| `No such file or directory` | Wrong target path | Verify path via local SFTP session |

---

## Quick Checklist

Before triggering a deploy workflow:

* [ ] All 5 SSH secrets added (if using SSH deploy)
* [ ] All 5 SFTP secrets added (if using SFTP deploy)
* [ ] SSH public key uploaded to WordPress.com (if using SSH)
* [ ] SFTP credentials tested locally
* [ ] Target directory exists on WordPress.com server
* [ ] Artifact build workflow succeeded (creates the plugin ZIP)

---

## Security Notes

* \*\*Never commit secrets to the repository\*\* (they belong in GitHub Settings only)
* \*\*Use SSH keys\*\* instead of passwords when possible (more secure)
* \*\*Rotate credentials\*\* periodically
* \*\*Limit secret access\*\* to necessary workflows only (GitHub manages this automatically)

---

\*Where gravity curves spacetime, resonance curves meaning-space.\*

# Quick Secrets Fill

# Quick-Fill GitHub Secrets — The Elidoras Codex

Copy-paste ready values for your WordPress.com deployment secrets.

## Your Site Info (From Screenshots)

* \*\*Site URL:\*\* `elidorascodex.com`
* \*\*SFTP Host:\*\* `sftp.wp.com`
* \*\*SFTP Port:\*\* `22`
* \*\*SFTP Username:\*\* `elidorascodexdotcom.wordpress.com`
* \*\*SSH Connection:\*\* `ssh elidorascodexdotcom.wordpress.com@ssh.wp.com`

---

## SSH Deploy Secrets (5 Required)

| Secret Name | Value to Paste |

|-------------|---------------|

| `WPCOM\_SSH\_HOST` | `sftp.wp.com` |

| `WPCOM\_SSH\_PORT` | `22` |

| `WPCOM\_SSH\_USER` | `elidorascodexdotcom.wordpress.com` |

| `WPCOM\_SSH\_PRIVATE\_KEY` | \*(paste your full private key from `~/.ssh/id\_ed25519`)\* |

| `WPCOM\_SSH\_TARGET` | `/htdocs/wp-content/plugins/tec-tgcr` |

---

## SFTP Deploy Secrets (5 Required)

| Secret Name | Value to Paste |

|-------------|---------------|

| `WPCOM\_SFTP\_HOST` | `sftp.wp.com` |

| `WPCOM\_SFTP\_PORT` | `22` |

| `WPCOM\_SFTP\_USER` | `elidorascodexdotcom.wordpress.com` |

| `WPCOM\_SFTP\_PASSWORD` | \*(your SFTP password from WordPress.com → Hosting → SFTP/SSH)\* |

| `WPCOM\_SFTP\_TARGET` | `/htdocs/wp-content/plugins/tec-tgcr/` |

---

## What You Need to Get/Generate

### 1. SSH Private Key (`WPCOM\_SSH\_PRIVATE\_KEY`)

\*\*If you already have one (from your screenshot, you do: "elidorascodex-default"):\*\*

Find your local private key file (usually `~/.ssh/id\_ed25519` or similar) and copy its entire content:

cat ~/.ssh/id\_ed25519

Or on Windows:

Get-Content ~\.ssh\id\_ed25519 -Raw

\*\*If you need to generate a new one:\*\*

ssh-keygen -t ed25519 -C "github-deploy-elidorascodex"

* Save to default location
* \*\*Leave passphrase empty\*\* (for CI/CD)
* Upload the \*\*public key\*\* (`.pub` file) to WordPress.com under \*\*Hosting → SFTP/SSH → SSH Keys\*\*

### 2. SFTP Password (`WPCOM\_SFTP\_PASSWORD`)

Get this from your WordPress.com dashboard:

1. 1. Go to: \*\*Hosting → SFTP/SSH\*\*
2. 2. Look for the \*\*PASSWORD\*\* field (see your first screenshot)
3. 3. Copy the password (or reset it if needed)

\*\*Note:\*\* This is \*\*different\*\* from your WordPress.com login password.

---

## Yes, They're Both the Same

You asked great questions—here are the answers:

### Port: Yes, both 22

* SSH uses port 22
* SFTP uses port 22
* They're the same because SFTP runs over SSH

### User: Yes, same username

* SSH user: `elidorascodexdotcom.wordpress.com`
* SFTP user: `elidorascodexdotcom.wordpress.com`
* Both connect to the same account

### Host: Slightly different display, same server

* Your screenshot shows: `sftp.wp.com`
* Documentation often says: `sftp.wordpress.com`
* \*\*Use `sftp.wp.com`\*\* (what WordPress.com shows you)

### Connection Command vs Username

* \*\*CONNECTION COMMAND:\*\* `ssh elidorascodexdotcom.wordpress.com@ssh.wp.com`
* \*\*USERNAME (for secrets):\*\* `elidorascodexdotcom.wordpress.com`

The connection command includes `@ssh.wp.com` at the end—that's the full `user@host` format for SSH. But for the GitHub secrets, you only need the \*\*username part\*\* before the `@`.

---

## Target Path

The plugin directory on your WordPress.com server is:

/htdocs/wp-content/plugins/tec-tgcr

* For SSH deploy: \*\*no trailing slash\*\* → `/htdocs/wp-content/plugins/tec-tgcr`
* For SFTP deploy: \*\*with trailing slash\*\* → `/htdocs/wp-content/plugins/tec-tgcr/`

---

## Security Note (You're Right!)

The hostname, port, and username aren't secrets—they're just connection coordinates. The \*\*actual security\*\* comes from:

* Your SSH \*\*private key\*\* (never share this)
* Your SFTP \*\*password\*\* (reset if compromised)

WordPress.com has your back on the infrastructure side. Your private key stays in GitHub Secrets (encrypted) and is never exposed in logs.

---

## Copy-Paste Checklist

Go to: `https://github.com/TEC-The-ELidoras-Codex/tec-tgcr/settings/secrets/actions`

Add these secrets (click \*\*New repository secret\*\* for each):

### SSH Secrets (if using SSH deploy)

* [ ] `WPCOM\_SSH\_HOST` = `sftp.wp.com`
* [ ] `WPCOM\_SSH\_PORT` = `22`
* [ ] `WPCOM\_SSH\_USER` = `elidorascodexdotcom.wordpress.com`
* [ ] `WPCOM\_SSH\_PRIVATE\_KEY` = \*(paste full private key)\*
* [ ] `WPCOM\_SSH\_TARGET` = `/htdocs/wp-content/plugins/tec-tgcr`

### SFTP Secrets (if using SFTP password deploy)

* [ ] `WPCOM\_SFTP\_HOST` = `sftp.wp.com`
* [ ] `WPCOM\_SFTP\_PORT` = `22`
* [ ] `WPCOM\_SFTP\_USER` = `elidorascodexdotcom.wordpress.com`
* [ ] `WPCOM\_SFTP\_PASSWORD` = \*(your SFTP password)\*
* [ ] `WPCOM\_SFTP\_TARGET` = `/htdocs/wp-content/plugins/tec-tgcr/`

---

\*\*After adding all secrets, go to Actions → WP.com SSH Deploy (manual) → Run workflow to test!\*\*

\*Where gravity curves spacetime, resonance curves meaning-space.\*

# M365 Integration

# Microsoft 365 / Dataverse / Fabric Integration

This doc lists core endpoints, permissions, and setup steps for using Microsoft Graph, Dataverse (Dynamics 365), and Microsoft Fabric with TEC-TGCR.

## Endpoints

* Microsoft Graph: [https://graph.microsoft.com](https://graph.microsoft.com)
* Dataverse Web API (per environment): `https://{yourorg}.crm.dynamics.com/api/data/v9.2`
* Fabric (OneLake / Lakehouse APIs): [https://api.fabric.microsoft.com](https://api.fabric.microsoft.com) (service) and [https://onelake.dfs.fabric.microsoft.com](https://onelake.dfs.fabric.microsoft.com) (storage)

## App registration (Entra ID)

1. 1. Go to Entra ID → App registrations → New registration
2. 2. Name: TEC-TGCR Service
3. 3. Supported account types: Single tenant (recommended)
4. 4. Redirect URI: (for local dev) [http://localhost:53682](http://localhost:53682) or omit for client credentials

### Add API permissions

* Microsoft Graph (Application):

- Files.Read.All (for OneDrive/SharePoint content)

- Sites.Read.All (for SharePoint site metadata)

- User.Read.All (if needed for directory lookups)

* Dataverse (Dynamics 365):

- Add delegated or application permissions via the Dataverse environment (application user + security role)

* Fabric (optional):

- Capacity.Read.All, Workspace.Read.All, Item.Read.All (depending on use), via Microsoft Fabric API

### Certificates or client secret

* Create a client secret (store in GitHub Secrets or .env.local) or upload a certificate

### Environment variables (local or CI)

AZURE\_TENANT\_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  
AZURE\_CLIENT\_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  
AZURE\_CLIENT\_SECRET=your-secret  
DATAVERSE\_URL=https://elidorascodex.crm.dynamics.com  
FABRIC\_TENANT\_ID= (if different)

## Minimal token flow (Python)

import os  
import httpx  
from urllib.parse import urlencode  
  
TENANT = os.environ['AZURE\_TENANT\_ID']  
CLIENT\_ID = os.environ['AZURE\_CLIENT\_ID']  
CLIENT\_SECRET = os.environ['AZURE\_CLIENT\_SECRET']  
  
# Client Credentials for Graph  
TOKEN\_URL = f"https://login.microsoftonline.com/{TENANT}/oauth2/v2.0/token"  
SCOPE = "https://graph.microsoft.com/.default"  
  
async def get\_token(scope=SCOPE):  
 async with httpx.AsyncClient(timeout=30) as client:  
 data = {  
 'client\_id': CLIENT\_ID,  
 'client\_secret': CLIENT\_SECRET,  
 'grant\_type': 'client\_credentials',  
 'scope': scope,  
 }  
 resp = await client.post(TOKEN\_URL, data=data)  
 resp.raise\_for\_status()  
 return resp.json()['access\_token']  
  
async def graph\_me\_drive():  
 token = await get\_token()  
 async with httpx.AsyncClient(timeout=30) as client:  
 r = await client.get(  
 'https://graph.microsoft.com/v1.0/me/drive/root',  
 headers={'Authorization': f'Bearer {token}'}  
 )  
 r.raise\_for\_status()  
 return r.json()

## Dataverse app user setup

1. 1. In Power Platform Admin Center, go to your environment → Settings → Users + permissions → Application users
2. 2. Add the app registration as an application user
3. 3. Assign a security role with Web API access to entities you need
4. 4. Test: GET `${DATAVERSE\_URL}/api/data/v9.2/WhoAmI()` with Authorization: Bearer `your-token-for-dataverse-resource`

## Fabric quick notes

* Fabric provides unified data items (lakehouse, warehouse, etc.) surfaced via OneLake
* APIs are evolving; use service principals with appropriate workspace permissions
* For large data movements, prefer Fabric pipelines or Dataflow Gen2

## Secrets and storage

* Never commit tenant IDs, client secrets, or tokens
* Store local in `.env.local` (ignored) and in CI as GitHub Secrets

---

[AIRTH] Verifying integration boundary conditions: minimal permissions, auditable access.

# Repository Audit

# Repository Audit Complete – October 15, 2025

## ✅ Status: READY FOR DEPLOYMENT

All critical systems verified, cruft removed, documentation updated, tests passing.

---

## 🔍 What Was Reviewed

### 1. WordPress Plugin (`apps/wordpress/tec-tgcr/`)

\*\*Status\*\*: ✅ Production-ready

* Plugin header complete with proper metadata
* Three REST endpoints active: `/ping`, `/citation`, `/model`
* Three shortcodes: `[tec\_tgcr\_ping]`, `[tec\_tgcr\_citation]`, `[tec\_tgcr\_model]`
* No hardcoded secrets; uses only public-domain content
* Security: ABSPATH check, sanitized inputs, proper permission callbacks
* \*\*Action\*\*: Deploy to WordPress.com via Simple artifact mode (recommended) or SSH

### 2. Deployment Workflows (`.github/workflows/`)

\*\*Status\*\*: ✅ All workflows valid

* `wpcom.yml`: Simple artifact deploy (pushes to `main` → uploads `wpcom` artifact)
* `wpcom-ssh-deploy.yml`: Manual SSH rsync (requires `WPCOM\_SSH\_HOST` secret)
* `wpcom-sftp-deploy.yml`: Manual SFTP deploy (optional, not in use)
* \*\*Action\*\*: Add `WPCOM\_SSH\_HOST=ssh.wp.com` secret if using SSH mode

### 3. Microsoft 365 Billing Analysis

\*\*Status\*\*: ✅ Cost breakdown complete

* \*\*Monthly recurring\*\*: ~$59/mo after trial conversions

- M365 Business Premium + Copilot: $52/mo

- Entra ID Governance P2 Add-on: $7/mo

* \*\*Expired\*\*: Microsoft 365 Business Standard (10/20/2025)
* \*\*Trial renewals\*\*:

- M365 BP + Copilot renews 11/3/2025

- Entra ID Governance renews 11/10/2025

* \*\*Action\*\*: Review billing account status (noted as "inactive" in portal); pay outstanding invoices if any

### 4. Repository Structure Cleanup

\*\*Status\*\*: ✅ Cruft removed from tracking

* Added to `.gitignore`: `todo/`, `.todo/`, `.git/todo/`, `tasks.json`
* Empty scaffold directories will be ignored (won't be committed)
* Existing `.env`, secrets, logs, and evidence files already excluded
* \*\*Action\*\*: Delete local `todo/`, `.todo/` directories manually if desired (they won't be committed)

### 5. Documentation Updates

\*\*Status\*\*: ✅ All core docs current

* `docs/SECRETS.md`: WP.com deployment steps complete, SSH quick-setup included
* `docs/WORDPRESS-DEPLOYMENT-CHECKLIST.md`: NEW – comprehensive deploy guide
* `data/financial/m365-cost-analysis-2025-10-15.md`: NEW – M365 cost breakdown
* `.github/copilot-instructions.md`: Agent collaboration protocol, architecture, style rules
* \*\*Action\*\*: Review `docs/WORDPRESS-DEPLOYMENT-CHECKLIST.md` for deployment steps

### 6. Persona Documentation

\*\*Status\*\*: ✅ All persona files present and cross-referenced

* `docs/PERSONAS.md`: Overview of all agents
* `docs/LuminAI.md`: LuminAI character spec with visual canon
* `docs/ARCADIA.md`: Arcadia Clone instructions
* `docs/FAERHEE.md`: FaeRhee operational agent
* `docs/MACHINE\_GODDESS.md`: Master persona coordination
* All referenced in `.github/copilot-instructions.md` (section 2 & 5)

### 7. Tests & Build Validation

\*\*Status\*\*: ✅ All tests passing

* Ran `pytest -q`: 14 tests passed, 0 failures
* Python environment: 3.13.3, all packages installed
* `tec-agent` CLI operational (Typer-based)
* \*\*No regressions detected\*\*

---

## 📋 Immediate Action Items

### For WordPress Deployment:

1. 1. \*\*Deploy to WordPress.com\*\* (recommended: Simple artifact mode)

- Push to `main` branch (or run "Publish to WordPress.com" workflow manually)

- WP.com will auto-pull `wpcom` artifact and deploy to `/wp-content/plugins/tec-tgcr`

- Verify: `https://YOURDOMAIN/wp-json/tec-tgcr/v1/ping` returns JSON

1. 2. \*\*Optional\*\*: Add `WPCOM\_SSH\_HOST=ssh.wp.com` secret if you want SSH deploy as backup

### For M365 Billing:

1. 1. \*\*Check billing account\*\*: Log into Azure/M365 portal, confirm no outstanding invoices
2. 2. \*\*Before 11/3/2025\*\*: Decide whether to keep M365 Copilot license ($30/mo) or cancel
3. 3. \*\*Before 11/10/2025\*\*: Decide whether to keep Entra ID Governance add-on ($7/mo) or cancel

### For Repository Maintenance:

1. 1. \*\*Commit changes\*\*: `.gitignore` updated, new docs added (`WORDPRESS-DEPLOYMENT-CHECKLIST.md`, `m365-cost-analysis-2025-10-15.md`)
2. 2. \*\*Delete local cruft\*\* (optional): Remove `todo/`, `.todo/` directories manually (they won't be tracked)
3. 3. \*\*Run `git status`\*\*: Confirm only intended files are staged

---

## 🚀 Next Steps After Deploy

1. 1. \*\*Verify WordPress endpoints\*\*:

- Ping: `/wp-json/tec-tgcr/v1/ping`

- Citation: `/wp-json/tec-tgcr/v1/citation?persona=luminai`

1. 2. \*\*Test shortcodes on a page\*\*:

- `[tec\_tgcr\_ping]`

- `[tec\_tgcr\_citation persona="arcadia"]`

- `[tec\_tgcr\_model src="/path/to/model.glb"]` (if you have a GLB uploaded)

1. 3. \*\*Monitor billing\*\*: Set calendar reminders for 11/3 and 11/10 trial renewal dates

---

## 📦 What's Clean & Current

* ✅ WordPress plugin ready
* ✅ Deployment workflows verified
* ✅ Cost analysis complete
* ✅ Repository structure cleaned
* ✅ Documentation updated
* ✅ Tests passing
* ✅ `.gitignore` updated
* ✅ No obsolete files in tracking

\*\*Everything is up-to-date and ready for production deployment.\*\*

---

## 🗂️ Files Changed/Added This Audit

### Added:

* `docs/WORDPRESS-DEPLOYMENT-CHECKLIST.md`
* `data/financial/m365-cost-analysis-2025-10-15.md`
* `docs/REPOSITORY-AUDIT-2025-10-15.md` (this file)

### Modified:

* `.gitignore` (added `todo/`, `.todo/`, `.git/todo/`, `tasks.json`)

### Verified (no changes needed):

* `apps/wordpress/tec-tgcr/tec-tgcr.php`
* `.github/workflows/wpcom.yml`
* `.github/workflows/wpcom-ssh-deploy.yml`
* `.github/workflows/wpcom-sftp-deploy.yml`
* `.github/copilot-instructions.md`
* `docs/SECRETS.md`
* All persona docs (`PERSONAS.md`, `LuminAI.md`, `ARCADIA.md`, `FAERHEE.md`, `MACHINE\_GODDESS.md`)

---

\_All systems verified and operational. Ready for deployment.\_