🚀 Guía Definitiva: Setup GitHub PCU-TCDS

Tiempo estimado: 30-45 minutos
Nivel: Intermedio

📋 Prerequisitos

- [ ] Cuenta de GitHub activa
- [ ] Git instalado localmente
- [ ] Python 3.8+ instalado
- [ ] Editor de código (VS Code, PyCharm, etc.)

Paso 1: Crear Repositorio en GitHub (5 min)

1.1 En GitHub.com

1. Ve a https://github.com/new
2. Configuración:
   - **Repository name**: `TCDS-PCU-Universal`
   - **Description**: `Puente Causal Universal - Operador isomórfico Q-Σ-φ-χ para validación causal multi-dominio (TCDS)`
   - **Visibility**: ✅ Public (para ciencia abierta)
   - **Initialize**: ❌ NO marcar "Add a README"
   - **License**: None (usaremos licencia dual personalizada)

3. Click **"Create repository"**

1.2 Guardar URL del repo

```bash
# Tu repo estará en:
https://github.com/geozunac3536-jpg/TCDS-PCU-Universal.git
```

Paso 2: Setup Local (10 min)

2.1 Crear directorio de trabajo

```bash
# En tu máquina local
cd ~/Projects  # O donde guardes tus proyectos
mkdir TCDS-PCU-Universal
cd TCDS-PCU-Universal
```

2.2 Inicializar Git

```bash
git init
git branch -M main
```

2.3 Conectar con GitHub

```bash
git remote add origin https://github.com/geozunac3536-jpg/TCDS-PCU-Universal.git
```

Paso 3: Crear Estructura de Directorios (2 min)

3.1 Ejecutar script de estructura

Guarda el siguiente código como `create_structure.py`:

```python
# (Copiar el contenido del artefacto create_repo_structure)
```

Ejecuta:

```bash
python create_structure.py
```

Esto crea:
```
TCDS-PCU-Universal/
├── src/
├── data/
├── config/
├── runs/
├── outputs/
├── docs/
├── notebooks/
├── tests/
└── scripts/
```

Paso 4: Copiar Código Core (15 min)

4.1 Archivos Python (desde artefactos de Claude)

Crea estos archivos en `src/`:

src/eveto.py
```python
# Copiar contenido completo del artefacto "eveto_module"
```

src/metrics_sigma.py
```python
# Copiar contenido completo del artefacto "metrics_sigma"
```

src/pcu_operator.py
```python
# Copiar contenido completo del artefacto "pcu_operator"
```

src/runner.py
```python
# Copiar contenido completo del artefacto "runner_pcu"
```

src/generate_trampa_dataset.py
```python
# Copiar contenido completo del artefacto "generate_trampa"
```

4.2 Script de demo

demo_pcu_completo.py (raíz del proyecto)
```python
# Copiar contenido del artefacto "demo_completo"
```

4.3 Configuración

config/default.yaml
```yaml
# Copiar contenido del artefacto "config_yaml"
```

requirements.txt
```
# Copiar contenido del artefacto "requirements_txt"
```

4.4 Documentación

README.md
```markdown
# Copiar contenido del artefacto "readme_pcu" (actualizado)
```

```

**LICENSE-DUAL.md**
```markdown
# Copiar contenido del artefacto "license_dual"
```

CITATION.cff
```yaml
# Copiar contenido del artefacto "citation_cff"
```

CONTRIBUTING.md
```markdown
# Copiar contenido del artefacto "contributing_md"
```

QUICKSTART.md
```markdown
# Copiar contenido del artefacto "quickstart_md"
```

IMPLEMENTATION.md
```markdown
# Copiar contenido del artefacto "implementacion_guide"
```

.gitignore
```
# Copiar contenido del artefacto "github_structure"
```

setup_repo.sh
```bash
# Copiar contenido del artefacto "repo_setup_script"
chmod +x setup_repo.sh
```

Paso 5: Instalación y Tests (10 min)

5.1 Ejecutar setup

```bash
./setup_repo.sh
```

Esto:
- Crea entorno virtual
- Instala dependencias
```

- Genera dataset trampa
- Ejecuta tests básicos

5.2 Activar entorno

```bash
# Linux/Mac
source venv/bin/activate

# Windows
venv\Scripts\activate
```

5.3 Test manual

```bash
python src/eveto.py
python src/metrics_sigma.py
python src/pcu_operator.py
```

Todos deben ejecutar sin errores y mostrar tests de auto-validación.

5.4 Demo completo

```bash
python demo_pcu_completo.py
```

Salida esperada:
```
=== Generando datos multi-dominio ===
=== Test de Reproducibilidad (N=3) ===
--- Corrida 1/3 (seed=42) ---
...
Reproducibilidad: 100.0%
✅ ÉXITO: El PCU rechazó correctamente el dataset trampa.
DESAFÍO SUPERADO ✅
```

Paso 6: Commit Inicial (5 min)

6.1 Verificar archivos

```bash
git status
```

6.2 Añadir archivos

```bash
git add .
```

6.3 Commit

```bash
git commit -m "feat: Initial commit PCU-TCDS v1.0.0

- Implementación completa del operador Q-Σ-φ-χ
- E-Veto con umbrales canónicos
- Σ-metrics (LI, R, RMSE_SL, κΣ, ΔH)
- Runner con trazabilidad forense (seeds, hashes)
- Dataset trampa generador
- Documentación completa (README, QUICKSTART, etc.)
- Tests básicos
- Licencia dual (CC BY-NC-SA 4.0 + Commercial)

Checklist 'Desafío Superado':
✅ Isomorfismo operativo
✅ E-Veto activo
✅ Silencio correcto (trampa rechazada)
✅ Reproducibilidad ≥ 95%
✅ Falsación absoluta
✅ Separación ocupacional"
```

6.4 Push a GitHub

```bash
git push -u origin main
```

Paso 7: Configurar GitHub (5 min)

7.1 Editar descripción del repo

En GitHub:
1. Ve a tu repo
2. Click en ⚙️ (Settings)
3. **About** → Edit
   - Description: `Puente Causal Universal - Operador Q-Σ-φ-χ para validación causal multi-dominio (TCDS)`
   - Website: `https://geozunac3536-jpg.github.io`
   - Topics: `tcds`, `causal-inference`, `reproducibility`, `falsification`, `coherence-metrics`

7.2 Habilitar Issues

Settings → Features → ✅ Issues

7.3 Crear Release

1. Ve a **Releases** → **Create a new release**
2. Tag: `v1.0.0`
3. Title: `PCU-TCDS v1.0.0 - Initial Release`
4. Description:
   ```markdown
   # PCU-TCDS v1.0.0 - Initial Public Release

   Primera implementación completa del Puente Causal Universal bajo el paradigma TCDS.

   ## Highlights

   - ✅ Operador isomórfico Q-Σ-φ-χ ejecutable
   - ✅ E-Veto entrópico (ΔH ≤ -0.20)
   - ✅ Reproducibilidad ≥ 95% verificada
   - ✅ Dataset trampa incluido y rechazado
   - ✅ Documentación completa

   ## Archivos

   - Código fuente completo en `src/`
   - Demo ejecutable: `demo_pcu_completo.py`
   - Setup automático: `setup_repo.sh`

   ## Instalación

   Ver [QUICKSTART.md](QUICKSTART.md)
   ```
5. Click **Publish release**

Paso 8: Conectar con Zenodo (OPCIONAL - 10 min)

8.1 Vincular repo con Zenodo

1. Ve a https://zenodo.org/account/settings/github/
2. Login con GitHub
3. Busca `TCDS-PCU-Universal`
4. Toggle **ON**

8.2 Obtener DOI

1. Ve a tu repo en Zenodo
2. Copia el DOI (ej: `10.5281/zenodo.XXXXX`)

8.3 Actualizar archivos con DOI

Edita estos archivos y reemplaza `XXXXX` con tu DOI real:

- `README.md` (badge)
- `CITATION.cff`
- `LICENSE-DUAL.md`

Commit:

```bash
git add README.md CITATION.cff LICENSE-DUAL.md
git commit -m "docs: Update with Zenodo DOI"
git push
```

Paso 9: Actualizar Portal TCDS (5 min)

9.1 Editar geozunac3536-jpg.github.io

En tu sitio web principal, añade una sección:

```markdown
## 🔬 PCU - Puente Causal Universal

**Repositorio:** [GitHub](https://github.com/geozunac3536-jpg/TCDS-PCU-Universal)
**DOI:** [10.5281/zenodo.XXXXX](https://doi.org/10.5281/zenodo.XXXXX)

El PCU es el operador ejecutable que unifica todos los dominios TCDS
bajo criterios de validez reproducibles y falsables.

- Reproducibilidad ≥ 95% con trazabilidad forense
- E-Veto entrópico obligatorio
- Dataset trampa (anti-apofenia)
- Multi-dominio (física, geofísica, biología, cognición)

[Ver demo →](https://github.com/geozunac3536-jpg/TCDS-PCU-Universal#-inicio-rápido)
```

✅ Checklist Final

Antes de anunciar públicamente, verifica:

- [ ] Repositorio en GitHub público
- [ ] README.md completo y con badges
- [ ] Tests básicos pasan
- [ ] `demo_pcu_completo.py` ejecuta sin errores
- [ ] Reproducibilidad ≥ 95% verificada
- [ ] Dataset trampa rechazado (0 ACCEPT)
- [ ] Licencia dual clara

- [ ] CITATION.cff presente
- [ ] Release v1.0.0 publicado
- [ ] DOI Zenodo (opcional pero recomendado)
- [ ] Portal TCDS actualizado con enlace

🎉 ¡Listo para Publicar!

Tu repositorio está completo y listo para:

1. **Compartir en redes sociales/académicas**
2. **Recibir contribuciones**
3. **Ser citado en papers**
4. **Usarse en investigación reproducible**

📊 Siguientes Pasos (Opcional)

Semana 1-2: Integración de Datos Reales

```bash
# Migrar datos sísmicos de CENAPRED_COLLAP
cp ../TCDS_CENAPRED_COLLAP_REPO/data/*.csv data/domain_sismo/

# Actualizar demo_pcu_completo.py para usar datos reales
```

Semana 3: Jupyter Notebook

Crear `notebooks/demo_pcu.ipynb` con visualizaciones interactivas.

Semana 4: Paper Técnico

Escribir preprint: "PCU: A Falsifiable Bridge Operator for Cross-Domain Causal Validation"

Mes 2: Comunidad

- Compartir en r/MachineLearning, r/datascience
- Presentar en conferencias
- Buscar colaboradores para auditoría

🆘 Troubleshooting

Error: "Permission denied (publickey)"

```bash
# Usar HTTPS en lugar de SSH
```

git remote set-url origin https://github.com/geozunac3536-jpg/TCDS-PCU-Universal.git
```

### Error: "Module not found" al ejecutar scripts

```bash
# Asegúrate de estar en el directorio raíz
cd TCDS-PCU-Universal

# Y con el entorno activado
source venv/bin/activate
```

Error: Push rechazado

```bash
# Si GitHub tiene commits que no tienes local
git pull origin main --rebase
git push
```

📧 Soporte

- **Issues GitHub**: https://github.com/geozunac3536-jpg/TCDS-PCU-Universal/issues
- **Email**: geozunac3536@gmail.com

¡Felicidades! Tu PCU-TCDS está en GitHub y listo para el mundo. 🚀
```