🚀 Quick Start - Codex Cloud

⚡ 5 Minutos para Começar

1️⃣ Autenticação (Uma Vez)

```
# Login com conta Nexcode
gcloud auth login adm@nexcode.live

# Configurar projeto
gcloud config set project elaihub-prod

# Verificar
gcloud auth list
# Deve mostrar: * adm@nexcode.live
```

2️⃣ Teste Rápido com cURL

```
# Obter token
export TOKEN=$(gcloud auth print-identity-token)

# Fazer primeira requisição
curl -X POST https://codex-wrapper-467992722695.us-
central1.run.app/api/v1/exec/stream \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "prompt": "What is 2+2? Answer with just the number.",
    "model": "gpt-4o-mini"
  }' \
  --no-buffer
```

Resultado esperado: Stream de eventos SSE terminando com resposta "4"

📋 Copiar e Colar - Exemplos Prontos

Exemplo 1: Pergunta Simples

```
export TOKEN=$(gcloud auth print-identity-token)

curl -X POST https://codex-wrapper-467992722695.us-
central1.run.app/api/v1/exec/stream \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
```

```
  -d '{"prompt": "Explain quantum computing in one sentence", "model":
"gpt-4o-mini"}' \
  --no-buffer | grep "agent_message" -A 1
```

Exemplo 2: Criar e Executar Script

```
export TOKEN=$(gcloud auth print-identity-token)

curl -X POST https://codex-wrapper-467992722695.us-
central1.run.app/api/v1/exec/stream \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "prompt": "Create a Python script that prints the current date and
time, then execute it",
    "model": "gpt-4o-mini",
    "timeout_ms": 60000
  }' \
  --no-buffer
```

Exemplo 3: Análise de Dados

```
export TOKEN=$(gcloud auth print-identity-token)

curl -X POST https://codex-wrapper-467992722695.us-
central1.run.app/api/v1/exec/stream \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "prompt": "Create a Python script that generates 10 random numbers and
calculates their average",
    "model": "gpt-4o-mini"
  }' \
  --no-buffer
```

🐍 Python - Script Completo Pronto

Salve como `test_cloud.py`:

```
#!/usr/bin/env python3
import subprocess
import requests
import json
import sys
```

```
def get_token():
    result = subprocess.run(
        ['gcloud', 'auth', 'print-identity-token'],
        capture_output=True, text=True, check=True
    )
    return result.stdout.strip()

def exec_codex(prompt, model="gpt-4o-mini"):
    token = get_token()
    url = "https://codex-wrapper-467992722695.us-
central1.run.app/api/v1/exec/stream"

    headers = {
        "Authorization": f"Bearer {token}",
        "Content-Type": "application/json"
    }

    payload = {
        "prompt": prompt,
        "model": model,
        "timeout_ms": 60000
    }

    response = requests.post(url, json=payload, headers=headers,
stream=True)
    response.raise_for_status()

    final_message = ""
    for line in response.iter_lines(decode_unicode=True):
        if line.startswith('data:'):
            data = line[5:].strip()
            try:
                data_json = json.loads(data)
                if 'message' in data_json:
                    final_message = data_json['message']
                elif 'last_agent_message' in data_json and
data_json['last_agent_message']:
                    final_message = data_json['last_agent_message']
            except:
                pass

    return final_message

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Uso: python3 test_cloud.py 'seu prompt aqui'")
        sys.exit(1)

    prompt = " ".join(sys.argv[1:])
    print(f"Executando: {prompt}\n")

    resultado = exec_codex(prompt)
    print(f"Resposta:\n{resultado}")
```

Uso:

```
chmod +x test_cloud.py
python3 test_cloud.py "What is the capital of France?"
python3 test_cloud.py "Create a hello world in Python and execute it"
```

🟨 JavaScript - Script Completo Pronto

Salve como `test_cloud.js`:

```
#!/usr/bin/env node
const { spawn } = require('child_process');
const https = require('https');

function getToken() {
    return new Promise((resolve, reject) => {
        const gcloud = spawn('gcloud', ['auth', 'print-identity-token']);
        let token = '';

        gcloud.stdout.on('data', (data) => { token += data.toString(); });
        gcloud.on('close', (code) => {
            if (code !== 0) reject(new Error('Falha ao obter token'));
            else resolve(token.trim());
        });
    });
}

async function execCodex(prompt, model = 'gpt-4o-mini') {
    const token = await getToken();
    const url = new URL('https://codex-wrapper-467992722695.us-
central1.run.app/api/v1/exec/stream');

    const postData = JSON.stringify({
        prompt,
        model,
        timeout_ms: 60000
    });

    return new Promise((resolve, reject) => {
        const options = {
            hostname: url.hostname,
            port: 443,
            path: url.pathname,
            method: 'POST',
            headers: {
                'Authorization': `Bearer ${token}`,
                'Content-Type': 'application/json',
                'Content-Length': Buffer.byteLength(postData)
```

```
            }
        };

        const req = https.request(options, (res) => {
            let finalMessage = '';
            let buffer = '';

            res.on('data', (chunk) => {
                buffer += chunk.toString();
                const lines = buffer.split('\n');
                buffer = lines.pop() || '';

                for (const line of lines) {
                    if (line.startsWith('data:')) {
                        try {
                            const data =
JSON.parse(line.substring(5).trim());
                            if (data.message) finalMessage = data.message;
                            if (data.last_agent_message) finalMessage =
data.last_agent_message;
                        } catch {}
                    }
                }
            });

            res.on('end', () => resolve(finalMessage));
        });

        req.on('error', reject);
        req.write(postData);
        req.end();
    });
}

async function main() {
    const prompt = process.argv.slice(2).join(' ');
    if (!prompt) {
        console.log('Uso: node test_cloud.js "seu prompt aqui"');
        process.exit(1);
    }

    console.log(`Executando: ${prompt}\n`);
    const resultado = await execCodex(prompt);
    console.log(`Resposta:\n${resultado}`);
}

main().catch(console.error);
```

Uso:

```
chmod +x test_cloud.js
node test_cloud.js "What is the capital of France?"
```

```
node test_cloud.js "Create a hello world in JavaScript and execute it"
```

🔥 Comandos Úteis do Dia a Dia

Renovar Token (quando expirar)

```
# Token expira em ~1 hora
# Para renovar:
gcloud auth login adm@nexcode.live

# Ou força nova sessão:
gcloud auth application-default login
```

Ver Logs do Serviço

```
# Últimos 50 logs
gcloud run services logs read codex-wrapper --region=us-central1 --
limit=50

# Logs em tempo real
gcloud run services logs tail codex-wrapper --region=us-central1

# Filtrar erros
gcloud run services logs read codex-wrapper --region=us-central1 | grep
ERROR
```

Status do Serviço

```
# Info geral
gcloud run services describe codex-wrapper --region=us-central1

# URL do serviço
gcloud run services describe codex-wrapper --region=us-central1 --
format="value(status.url)"

# Última revisão
gcloud run services describe codex-wrapper --region=us-central1 --
format="value(status.latestReadyRevisionName)"
```

Upload/Download GCS

```
# Upload
gsutil cp meu-arquivo.txt gs://elaistore/uploads/
```

```
# Download
gsutil cp gs://elaistore/sessions/session-123.json ./

# Listar
gsutil ls gs://elaistore/sessions/
```

🪛 Troubleshooting Rápido

Erro: "gcloud: command not found"

```
# macOS
brew install --cask google-cloud-sdk

# Linux
curl https://sdk.cloud.google.com | bash
exec -l $SHELL  # Reiniciar shell
```

Erro: "401 Unauthorized"

```
# Token expirou, renovar:
gcloud auth login adm@nexcode.live
```

Erro: "403 Forbidden"

```
# Verificar se está usando conta @nexcode.live
gcloud auth list

# Se necessário, trocar conta
gcloud config set account adm@nexcode.live
```

Erro: "Timeout"

```
# Aumentar timeout na requisição
# Adicionar: "timeout_ms": 120000  (2 minutos)
```

📱 Atalhos para Shell

Adicione ao seu `~/.bashrc` ou `~/.zshrc`:

```
# Atalho para obter token
alias codex-token='gcloud auth print-identity-token'

# Atalho para executar codex cloud
codex-exec() {
    local TOKEN=$(gcloud auth print-identity-token)
    curl -X POST https://codex-wrapper-467992722695.us-
central1.run.app/api/v1/exec/stream \
      -H "Authorization: Bearer $TOKEN" \
      -H "Content-Type: application/json" \
      -d "{\"prompt\": \"$1\", \"model\": \"gpt-4o-mini\"}" \
      --no-buffer
}

# Uso:
# codex-exec "What is 2+2?"
```

Depois de adicionar:

```
source ~/.bashrc  # ou source ~/.zshrc
codex-exec "test"
```

🎯 Casos de Uso Comuns

1. Gerar Código

```
codex-exec "Create a FastAPI endpoint that returns user data from a
database"
```

2. Análise de Logs

```
codex-exec "Analyze this error log and suggest a fix: [paste error]"
```

3. Documentação

```
codex-exec "Generate API documentation for this Python function: [paste
code]"
```

4. Refatoração

```
codex-exec "Refactor this code to be more Pythonic: [paste code]"
```

5. Testes

```
codex-exec "Create unit tests for this function: [paste code]"
```

📊 Modelos Disponíveis

Modelo	Velocidade	Capacidade	Custo	Quando Usar
`gpt-4o-mini`	⚡ ⚡ ⚡ Rápido	⭐ ⭐ Médio	💰 Barato	Tarefas simples, prototipagem
`gpt-4o`	⚡ ⚡ Médio	⭐ ⭐ ⭐ Alto	💰 💰 Médio	Tarefas complexas, produção
`claude-sonnet-4`	⚡ ⚡ Médio	⭐ ⭐ ⭐ Alto	💰 💰 Médio	Análise profunda, código

Padrão: `gpt-4o-mini` (bom equilíbrio custo/benefício)

🆘 Precisa de Ajuda?

1. **Verificar documentação completa**: `GUIA_COMPLETO_USO.md`
2. **Ver resumo da implementação**: `RESUMO_IMPLEMENTACAO.md`
3. **Contato**: adm@nexcode.live

Pronto para usar! 🎉

Qualquer dúvida, consulte a documentação completa ou entre em contato.