ChatGPT

Problem: Gradle Prewarm Phase Fails in Codex Environment

Symptom: During the "Light Gradle prewarm" step of your `maintain_codex_env.sh`, the Gradle commands (like listing tasks) hang or time out, resulting in a **"context canceled"** error. This indicates the Codex environment is aborting the operation before completion. Others have noted similar difficulties getting Android/Gradle code to run in the Codex cloud environment [1].

Likely Causes: - **Network Access Blocked:** The Codex cloud environment **has no direct internet** access for security. Gradle may be trying to download the Gradle distribution or project dependencies (e.g. Android Gradle plugin, libraries) and getting stuck because the network is unreachable [2]. In OpenAI's forum, users found Maven/Gradle needed proxy settings in this environment [2]. - **Time Limits:** The environment imposes a time limit on setup commands. Long-running Gradle configuration (especially an Android project's full task scan) can exceed this limit, causing the context to cancel. - **Android SDK/License Issues:** If the Android SDK isn't fully set up in the container, running tasks that touch Android modules (like your `:app` or `:wear` modules) might pause or fail awaiting SDK components or license approvals.

Solutions and Script Adjustments

To fix these issues without skipping the Gradle phase, we can modify `maintain_codex_env.sh` and environment settings as follows:

1. Enable Proxy for Gradle/Maven (Allow dependency download)

Configure Gradle and Maven to use the Codex environment's proxy so they can fetch needed files. OpenAI's Codex cloud uses a proxy at host `proxy` port **8080** for outbound traffic [2]. We should set the `GRADLE_OPTS` and `MAVEN_OPTS` environment variables accordingly at the start of the script:

```
# Enable proxy for Gradle/Maven in Codex environment
export GRADLE_OPTS="$GRADLE_OPTS -Dhttp.proxyHost=proxy -Dhttp.proxyPort=8080 -
Dhttps.proxyHost=proxy -Dhttps.proxyPort=8080"
export MAVEN_OPTS="$MAVEN_OPTS -Dhttp.proxyHost=proxy -Dhttp.proxyPort=8080 -
Dhttps.proxyHost=proxy -Dhttps.proxyPort=8080"
```

This ensures Gradle/Maven use the provided proxy. (This tip comes from others running Java builds in Codex [2].)

2. Avoid Full Task Scan on Init

Listing **all Gradle tasks** (`gradlew tasks`) can be very slow for multi-module Android projects, as it configures every module. In a limited environment, it's better to do a lighter warm-up: - **Warm up only a single module**: You already run `:core:help` which targets the `core` module. This is smart, as it avoids initializing heavy Android application modules. We can keep this. It will download the Gradle wrapper and

any core module dependencies. - **Skip the full** `gradlew tasks`: Instead of listing all tasks, consider listing tasks only for a lightweight module or just skipping this step. Since the goal is just to pre-warm caches, the `:core:help` (and possibly a simple `gradlew help` or `gradlew projects`) may suffice. Removing the full tasks scan will significantly reduce startup time in the container finalize step. For example, you could replace it with a narrower query or omit it:

```
tlim 90 ./gradlew :core:help --no-daemon --quiet
# (Omit the global "gradlew tasks" to avoid heavy configuration of all modules)
```

This prevents touching the Android-specific modules during initialization. If you still want to list tasks, limit it to a specific subproject (e.g. `./gradlew :core:tasks`) to avoid triggering Android SDK checks.

3. Make Gradle Commands Non-Blocking

Ensure that if a Gradle command does time out or fail, it **doesn't abort the entire script**: - Add `|| true` (or a similar failure guard) after each `tlim ... ./gradlew` invocation. This way, even if the command times out (exit code 124 from `timeout`) or fails, the script will continue instead of exiting on `set -e`. For example:

```
tlim 90 ./gradlew --version || true
tlim 90 ./gradlew :core:help --no-daemon --quiet || true
tlim 90 ./gradlew tasks --no-daemon --quiet || true
```

This will ignore non-zero exit codes from the Gradle calls. The `--version` and `:core:help` are mostly safe, but the `tasks` one is the most likely to exceed time. By appending `|| true`, the script won't terminate even if "context canceled" occurs on that line. - (Alternatively, you could temporarily disable `set -e` around the prewarm section and re-enable it after. But using `|| true` on each line is straightforward.)

4. Pre-Install Android SDK or Bypass Android Requirements

If not already done, consider installing a minimal Android SDK or suppressing SDK checks for the Codex environment: - If your environment script can run apt or SDK manager, you might install the Android SDK command-line tools and set `ANDROID_HOME`. However, without internet this is tricky. If the Codex base image doesn't include an SDK, Gradle will complain when touching Android projects. - To bypass SDK license prompts, you can pre-create a dummy `licenses` folder in `~/.android` with an accepted licenses file, or use Gradle's `android.builder.sdkDownload=false` to skip auto-downloads. But since we're avoiding the Android modules in prewarm, this might not be necessary until actual builds. - In summary, focus on running **only non-Android parts** of the build in the Codex environment (e.g. pure JVM unit tests in the core module). According to one user's experience, full Android builds didn't run in Codex due to these limitations [1], so using your CI (GitHub Actions) for full builds might be the fallback [3].

5. Increase Timeouts if Possible *(Optional)*

If the environment allows, you might increase the `tlim 90` to a slightly higher value (e.g. 120 seconds) for the task listing, to give it more chance to finish. However, be cautious – the environment itself may have an upper bound (and a longer wait increases the risk of hitting global timeout). Try the proxy and skipping steps first, as they reduce actual time needed.

Revised Script Snippet

Implementing the above changes, your `maintain_codex_env.sh` "Light Gradle prewarm" section might look like this:

```
log '  Light Gradle prewarm (no heavy tasks)…'
echo -e '\033[0;34m  Light Gradle prewarm (no heavy tasks)…\033[0m'

# Setup proxy for Gradle/Maven in Codex environment
export GRADLE_OPTS="$GRADLE_OPTS -Dhttp.proxyHost=proxy -Dhttp.proxyPort=8080 -
Dhttps.proxyHost=proxy -Dhttps.proxyPort=8080"
export MAVEN_OPTS="$MAVEN_OPTS -Dhttp.proxyHost=proxy -Dhttp.proxyPort=8080 -
Dhttps.proxyHost=proxy -Dhttps.proxyPort=8080"

# Light prewarm only (avoid long tasks in container finalize)
tlim 90 ./gradlew --version || true
tlim 90 ./gradlew :core:help --no-daemon --quiet || true
# Optional: prewarm core module tasks (avoid full project tasks to save time)
tlim 90 ./gradlew :core:tasks --no-daemon --quiet || true

# (Removed the full `./gradlew tasks` to prevent timeout)
```

With this adjusted script:

- Gradle will use the proxy for any downloads (preventing hang on network calls).
- The script won't exit on a timeout – it logs the attempt and moves on.
- We only warm up the lightweight module (`core`); the heavy Android app modules are left for the CI pipeline to handle [3] or for later, once the container is fully initialized.

Conclusion

By applying these changes, the Codex environment setup should no longer get stuck in the Gradle prewarm phase. The environment will initialize faster and continue even if Gradle tasks exceed the time limit. This solution follows patterns observed by others using Codex with Java projects – using proxies for build tools and minimizing on-start workload [2] [1].

Now, you can keep the Gradle warm-up phase without it crashing the container, allowing Codex to proceed with running tests or analyses on your project code. Good luck, and happy coding!

Sources:

- Stephen Siapno, *"Boosting Android Development with OpenAI Codex: First Impressions"* – noted limitations in Codex environment for Android builds [1] [3].
- OpenAI Community Forum – guidance on Maven/Gradle proxy settings in Codex cloud (set `MAVEN_OPTS` and similar) [2].

---

[1] [3] Boosting Android Development with OpenAI Codex: First Impressions | by Stephen Siapno | ProAndroidDev
https://proandroiddev.com/boosting-android-development-with-openai-codex-first-impressions-4ec6e8dcb83a?gi=5902b34199f8

[2] Custom environment variables are ignored in codex
https://community.openai.com/t/custom-environment-variables-are-ignored-in-codex/1264251