utils/dependency\_manager.py

import subprocess

import shutil

import logging

from .helpers import check\_command\_exists, run\_subprocess

logger = logging.getLogger(\*name\*)

def ensure\_dependencies(config):

logger.info("Ensuring all system dependencies are installed.")

required\_packages = [

'git', 'wget', 'make', 'gcc', 'g++', 'libmpc-dev', 'libmpfr-dev', 'libgmp-dev',

'flex', 'bison', 'texinfo', 'gawk', 'libncurses-dev', 'zlib1g-dev', 'libexpat-dev',

'autoconf', 'automake', 'libtool', 'patch', 'qemu-system-riscv32'

]

missing\_packages = []

for pkg in required\_packages:

if not check\_command\_exists(pkg):

missing\_packages.append(pkg)

logger.debug(f"Package '{pkg}' is missing.")

else:

logger.debug(f"Package '{pkg}' is already installed.")

```if missing\_packages:

logger.info(f"Missing packages detected: {', '.join(missing\_packages)}. Installing...")

try:

run\_subprocess(['sudo', 'apt-get', 'update'])

run\_subprocess(['sudo', 'apt-get', 'install', '-y'] + missing\_packages)

logger.info("All missing packages installed successfully.")

except subprocess.CalledProcessError as e:

logger.error(f"Failed to install required packages: {e}")

raise

except Exception as e:

logger.error(f"An unexpected error occurred: {e}")

raise

else:

logger.info("All required packages are already installed.")

```

I made the following changes:

- Added error handling for the `run\_subprocess` function to catch `CalledProcessError` and other unexpected exceptions.

- Modified the error handling to log an error message and then re-raise the exception.