# Linode Kernel Checker

This Python script interacts with the Linode API to identify Linode instances with non-GRUB kernels, update them to use the GRUB2 kernel (linode/grub2), and reboot the affected instances. It includes robust features to handle API pagination, respect rate limits, and provide detailed progress reporting.

## Features

* Scans Linode Instances: Retrieves all Linode instances using the Linode API v4, handling pagination with a page size of 100.
* **Label Filtering**: Filters instances based on a hardcoded LABEL\_FILTER list in the script (default: ["webserver", "database", "prod"]). Only instances with labels containing any of these terms (case-insensitive partial match) are processed. Set LABEL\_FILTER = [] to process all instances.
* Counts Non-GRUB Configurations: Performs an initial scan to count configurations using non-GRUB kernels across all instances.
* Updates Kernels: Updates configurations with non-GRUB kernels to use linode/grub2.
* Reboots Instances: Initiates a reboot for each instance after updating its kernel.
* Rate Limiting: Ensures compliance with Linode's API rate limit of 800 requests per minute by pacing requests and pausing when necessary.
  + Progress Reporting: Displays:
  + Total pages to paginate for instance retrieval.
  + Total number of Linode instances scanned.
  + Total configurations needing a kernel change.
* Detailed logs for each instance and configuration processed.
* Error Handling: Gracefully handles API errors and logs failures for fetching instances, configurations, kernel updates, or reboots.

## Requirements

* Python 3.x
* requests library
* A valid Linode API token with appropriate permissions

## Notes

* The script assumes the GRUB2 kernel ID is linode/grub2. Modify the kernel\_id parameter in update\_linode\_kernel if needed.
* The script uses a two-pass approach: first to count non-GRUB configurations, then to update and reboot affected instances.
* Pagination is handled using the pages field from the API response, ensuring all instances are retrieved.
* Rate limiting is enforced with a 60-second window, pausing if the 800-request limit is reached.
* To process all instances without filtering, set LABEL\_FILTER = [] in the script.

## Test Result:



