Load Balancer Overview

Project Zeus Load Balancer: Technical Documentation

Overview

This documentation provides detailed information on the various components of the Project Zeus Load Balancer, a system designed to distribute traffic across multiple backend servers for high availability and performance. The following scripts and configuration files are integral to the load balancer's operation and management. Each section provides a breakdown of the file's purpose, key functions, and potential for further customization.

1. `watch.php`

Purpose

The `watch.php` script serves as a monitoring tool for the Project Zeus load balancer, responsible for tracking load balancer activities, such as server health, incoming traffic, and error rates. Its primary function is to ensure the load balancer operates efficiently and to provide real-time updates or logs for system administrators.

Key Functions and Components

Initialization

- The script likely starts with initialization routines that set up required variables, such as API keys, paths to log files, or external dependencies (libraries like cURL for making HTTP requests).
- It may load configuration settings, such as thresholds for server loads or alert criteria, from external configuration files or databases.

Monitoring Logic

- **Server Health Checks**: The script might periodically ping backend servers to verify they are online and responsive. This could include checking response times, CPU usage, memory utilization, and disk space.
- **Load Balancer Metrics**: It could track traffic distribution among backend servers, ensuring an even load spread and preventing any single server from becoming overwhelmed.
- **Error Tracking**: Monitoring for errors, such as HTTP 500/502/503 errors, which could indicate server issues or application failures.

Output/Logging

- **Log Generation**: The script likely writes logs to files or external logging systems (e.g., ELK stack or Graylog). These logs can contain server status, traffic metrics, and timestamps of key events.
- **Alerts**: The script may trigger alerts when certain conditions are met, such as high CPU usage or server downtime. Alerts could be sent via email, SMS, or third-party services like Slack or PagerDuty.

Possible Interactions

- **Nginx**: The script might interact with Nginx to check the status of servers behind the load balancer or to pull access logs for traffic analysis.
- **Databases**: The script could log metrics or server statuses to a database (e.g., MySQL, InfluxDB) for long-term storage and analysis.
- **Auto-scaling/Failover**: It might trigger automatic scaling actions or redirect traffic to backup servers when predefined thresholds are breached, improving fault tolerance.

2. `install.sh`

Purpose

The `install.sh` script automates the setup and deployment of the Project Zeus load balancer environment. This ensures that the necessary services, configurations, and dependencies are properly installed and configured for optimal performance.

Key Functions and Components

Environment Setup

- **Dependency Checks**: The script likely begins by verifying the presence of required system packages (e.g., `nginx`, `php`, `php-fpm`, `curl`) and installs them if they are missing.
- **System Configuration**: It may modify system settings like network interfaces (e.g., enabling IP forwarding) or update firewall rules (e.g., configuring `iptables` or `firewalld`) to allow traffic through the load balancer.

File Deployment

- **Configuration Files**: The script could copy configuration files such as `nginx.conf` to appropriate directories, ensuring they are in place before starting the services.
- **Directory Setup**: It may create necessary directory structures (e.g., `/opt/zeus/`, `/var/www/html/`) to store web assets, logs, or custom scripts.

Service Initialization

- **Starting Services**: The script is responsible for starting key services like Nginx or monitoring scripts (`watch.php`). It might also check that the services are running correctly post-startup.
- **Registering Services**: It may configure services to start automatically on system boot using tools like `systemctl` (for SystemD) or `chkconfig` (for older init systems).

Possible Customization

- **Custom Installation Paths**: Variables at the beginning of the script (e.g., `$INSTALL_PATH`, `$NGINX_CONF_PATH`) may allow for installation in custom directories or systems.
- **Load Balancer Options**: The script could allow customization of load balancing algorithms (e.g., round-robin, least connections) or server pools based on the project's needs.

3. `crontab.txt`

Purpose

The `crontab.txt` file defines scheduled tasks that automate the maintenance and monitoring processes within the Project Zeus cluster. These cron jobs ensure periodic execution of scripts or tasks vital to system operations.

Key Functions and Components

CMS Agent:

```
@reboot php -q /opt/zeus/agent.php
```

- This cron job ensures that the `agent.php` script is run on system reboot. This script could register the server with a central load balancer, perform initial configuration tasks, or run health checks on system startup.

Git Update:

```
# 1 * * * * sh /var/www/html/update.sh
```

- This (currently commented) cron job is set to run every hour. It might pull code updates from a Git repository and redeploy them. The job is commented out, possibly due to ongoing development or testing.

Important Notes

- **Job Timing**: Cron job intervals should be carefully chosen to avoid overloading the system while ensuring tasks run often enough for the load balancer to remain in sync.
- **Operational Integrity**: The cron jobs help maintain the uptime and functionality of the Zeus environment by running critical scripts at regular intervals.

4. `nginx.conf`

Purpose

The `nginx.conf` file configures the Nginx web server, which functions as the load balancer for Project Zeus. It directs incoming traffic and handles SSL/TLS encryption, logging, and error handling.

Key Functions and Components

Load Balancing Configuration

- **Upstream Directives**: Defines a list of backend servers that receive traffic from Nginx. Load balancing methods such as round-robin, least connections, or IP hash may be used to distribute traffic.

```
upstream backend {
    server backend1.example.com;
    server backend2.example.com;
    least_conn; # Load balancing method
}
```

SSL/TLS Settings

- **SSL Configuration**: The file may contain paths to SSL certificates and configuration for handling secure connections.

```
ssl_certificate /etc/ssl/certs/zeus.crt;
ssl_certificate_key /etc/ssl/private/zeus.key;
ssl_protocols TLSv1.2 TLSv1.3;
```

Server Blocks

- **Multiple Server Blocks**: Nginx can serve multiple domains or applications through different server blocks. Each block could route traffic to specific backends based on domain name or request path.

Logging and Error Handling

- **Access and Error Logs**: Configuration for logging requests and errors. These logs are crucial for debugging and performance tuning.

```
access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log;
```

Advanced Configurations

- **Caching**: Nginx might be configured to cache responses for certain types of content, reducing the load on backend servers.
- **Rate Limiting**: Configuration to limit the number of requests from individual IP addresses to mitigate DDoS attacks.

5. `subtitles.php`

Purpose

The `subtitles.php` script is responsible for managing subtitle files, possibly in conjunction with media content distributed through the Project Zeus infrastructure. It could handle tasks such as subtitle file uploads, formatting, and synchronization with video streams.

Key Functions and Components

File Handling

- **Subtitle File Manipulation**: This script might read and write subtitle files, convert between different formats (e.g., SRT, VTT), and adjust timestamps to synchronize subtitles with video content.

Integration with Load Balancer

- **Media Distribution**: If subtitles are part of media content being load balanced, the script may work in conjunction with the load balancer to ensure subtitles are served from the correct backend server.

Error Handling

- **File Validation**: The script likely checks for issues such as missing files or incorrect formats and handles errors gracefully, preventing corrupted files from being served.

Potential for Expansion

- **Additional Subtitle Formats**: The script could be extended to support more subtitle formats, or it could integrate with third-party subtitle services (e.g., automatic translation or synchronization services).

Overall Project Integration

The files and scripts discussed above are essential for the smooth operation of the Project Zeus Load Balancer. From installation (`install.sh`) and monitoring (`watch.php`), to periodic maintenance (`crontab.txt`) and service configuration (`nginx.conf`), these components collectively ensure a high-performing, reliable load balancing infrastructure. Understanding each component's role and interaction is crucial for further development, troubleshooting, and optimization of the load balancer environment.