****

**Trickster**

**Introduction**

Trickster is an HTTP reverse proxy/cache for http applications and a dashboard query accelerator for time series databases.

**Project Summary**

|  |
| --- |
| Website :- https://trickstercache.org/ |
| Organization/Foundation Name:- Cloud Native Computing Foundation (CNCF) |
| License :-Apache-2.0 license |
| Open/Proprietary:- Open |
| Source Path(if open source):- https://github.com/trickstercache/trickster.git |

Brief Description An Open Source HTTP Reverse Proxy Cache and Time Series Dashboard

Accelerator.

Trickster is a fully-featured HTTP Reverse Proxy Cache for HTTP applications like static file servers and web API's.

Trickster dramatically improves dashboard chart rendering times for end users by eliminating redundant computations on the TSDBs it fronts. In short, Trickster makes read-heavy Dashboard/TSDB environments, as well as those with highly-cardinalized datasets, significantly more performant and scalable.

**Project Details**

**Key Features**

● A unique and powerful Application Load Balancer for Time Series and generic HTTP endpoints

● Supports TLS and HTTP/2 for frontend termination and backend origination

● Offers several options for a caching layer, including in-memory, filesystem, Redis and bolt

● Highly customizable, using simple yaml configuration settings, down to the HTTP Path

● Built-in Prometheus metrics and customizable Health Check Endpoints for end-to-end monitoring

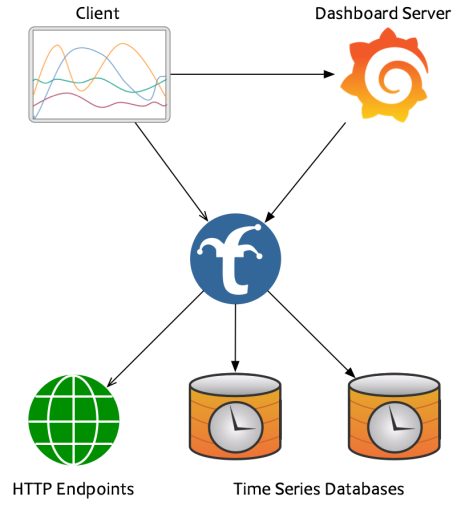
● Negative Caching to prevent domino effect outages

● High-performance Collapsed Forwarding

● Best-in-class Byte Range Request caching and acceleration.

● Distributed Tracing via OpenTelemetry, supporting Jaeger and Zipkin ● Rules engine for custom request routing and rewriting

**Architecture**



**Current Usage**

Technical DTrickster is an open-source project that provides a fast, lightweight, and scalable HTTP/Reverse Proxy Cache for Kubernetes, OpenShift, and other cloud-native applications. Here are some organizations and products that are used by the Trickster project:

* Prometheus: Trickster is designed to work with Prometheus, a popular open-source monitoring system and time-series database. Trickster can act as a reverse proxy cache for Prometheus, accelerating query response times and reducing load on Prometheus.
* Kubernetes: Trickster is built for Kubernetes, a popular container orchestration system. Trickster can be deployed as a sidecar container alongside applications in Kubernetes, providing a performant caching layer that accelerates application response times.
* Grafana: Trickster is compatible with Grafana, an open-source visualization and analytics platform. Grafana can be used to visualize data stored in Prometheus, which Trickster can cache and accelerate.
* OpenShift: Trickster is also designed to work with OpenShift, a container application platform based on Kubernetes. Trickster can be deployed in OpenShift environments to provide a caching layer that speeds up application response times and reduces load on upstream resources.
* Cloud-native applications: Trickster is built for cloud-native applications, which are designed to run in containerized environments like Kubernetes and OpenShift. Trickster can provide a caching layer that improves the performance of these applications, making them more responsive and scalable.

Overall, Trickster integrates with a range of popular open-source projects and cloud-native technologies to provide a performant caching layer for Kubernetes, OpenShift, and other cloud-native applications.

**Technical details**

Trickster is designed to provide a fast, lightweight, and scalable HTTP/Reverse Proxy Cache for Kubernetes, OpenShift, and other cloud-native applications. Here are some technical details about Trickster's scalability, performance, and other features:

1. Caching: Trickster provides a caching layer that can significantly improve the performance of cloud-native applications. Trickster can cache data from Prometheus, reducing the number of queries that need to be made to Prometheus and speeding up response times.
2. Scaling: Trickster is designed to be horizontally scalable, meaning that additional instances of Trickster can be added to a deployment to handle increasing traffic loads. Trickster can be deployed as a sidecar container alongside application pods in Kubernetes or OpenShift, making it easy to scale alongside the applications it's caching data for.
3. Load balancing: Trickster can act as a load balancer, distributing traffic across multiple instances of an application. Trickster can also distribute traffic across multiple Prometheus instances, reducing the load on any one instance and improving overall performance.
4. Compression: Trickster supports data compression, reducing the amount of data that needs to be transmitted between clients and servers. This can significantly improve performance, particularly in high-latency environments.
5. Prometheus compatibility: Trickster is designed to be fully compatible with Prometheus, supporting all of Prometheus's query language and data formats. This means that Trickster can be used as a drop-in replacement for Prometheus, without requiring any changes to existing Prometheus clients or configurations.
6. Metrics: Trickster provides its own metrics, which can be used to monitor and optimize its performance. Trickster also supports integration with popular monitoring systems like Prometheus and Grafana.

Overall, Trickster is designed to provide high performance and scalability for cloud-native applications. Its caching layer, horizontal scalability, load balancing, compression, and Prometheus compatibility make it a powerful tool for optimizing the performance of Kubernetes, OpenShift, and other cloud-native applications.

**Project comparison**

Trickster is an open-source reverse proxy cache that is designed to accelerate web applications by caching frequently accessed resources. While there are several other open-source projects that offer similar functionality, some of the most popular ones include Varnish, Squid, and Nginx.

1. **Varnish** is a popular HTTP accelerator and cache that is designed to speed up web applications by caching frequently accessed resources. It is widely used in high-traffic websites and offers a powerful configuration language that allows developers to customize its behavior extensively. Varnish can be used as a standalone caching server or as a reverse proxy cache in front of web servers.
2. **Squid** is another open-source caching proxy that is widely used in the industry. It is designed to improve web performance by caching frequently accessed resources and reducing network traffic. Squid supports a variety of protocols, including HTTP, HTTPS, FTP, and others, and can be configured to perform various caching and filtering functions.
3. **Nginx** is a high-performance web server and reverse proxy that is designed to handle high volumes of traffic. It is often used as a load balancer, caching server, or reverse proxy cache to improve web performance. Nginx offers a simple configuration language and is known for its speed and scalability.

**Any other information**

1. Trickster is written in Go, a programming language that is known for its simplicity,concurrency support,and fast compilation times.
2. Trickster is designed to work with popular caching systems like Prometheus,Graphite,and InfluxDB.It can query these systems for metrics data and use the results to determine what data to cache and how long to cache it.
3. Trickster offers several advanced caching features, including stale-while-revalidate (SWR) and time-to-live (TTL) based caching. These features can help improve application performance by reducing the amount of time it takes to retrieve data from the cache.
4. Trickster is actively maintained by the Cloud Native Computing Foundation (CNCF), which is a vendor-neutral organization that supports the development and adoption of cloud-native technologies.

**Reference / Acknowledgments**

1. **Trickster GitHub repository:** [**https://github.com/Comcast/trickster**](https://github.com/Comcast/trickster)
2. **Trickster documentation: https://trickster.io/docs/**
3. **Trickster blog:** [**https://trickster.io/blog/**](https://trickster.io/blog/)
4. **Trickster on the CNCF website: https://www.cncf.io/projects/trickster/**

**-----------------------------End of the template--------------------------------**