pingpong - A metrics and alerts cloning and generation Service

A Go-based HTTP/HTTPS service that provides dynamic metric creation, tracking, and webhook handling with support for both Prometheus and Victoria Metrics.

Overview

This service acts as a bridge between your application metrics and monitoring systems, providing endpoints for dynamic metric creation, updates, and exposition. It supports both standard HTTP and HTTPS protocols, with built-in support for Prometheus metric scraping and webhook handling for various notification services.

Key Features

- **Dynamic Metric Management**
    - Create new metrics via API
    - Update existing metrics
    - Support for counter-type metrics with labels
    - Compatible with Prometheus Client Data Exposition Format
- **Dual Protocol Support**
    - HTTP (default: 8090)
    - HTTPS (default: 8443) with configurable TLS
    - Optional TLS enforcement
- **Monitoring Integration**
    - Prometheus metrics exposition
    - Victoria Metrics support
    - Custom metric creation and tracking
    - Label-based metric segregation
- **Webhook Support**
    - Generic webhook endpoint
    - Specialized endpoints for:
        - Slack
        - PagerDuty
    - JSON payload validation and pretty printing

Endpoints

GET /metrics - Prometheus metrics endpoint

POST /create - Create new metrics

POST /update - Update existing metrics

GET /ping - Health check endpoint

POST /webhook - Generic webhook handler

POST /slack - Slack webhook handler

POST /pagerduty - PagerDuty webhook handler

Configuration

Flag	Default	Description
`--port`	8090	HTTP port
`--port-tls`	8443	HTTPS port
`--disable-tls`	false	Disable HTTPS listener
`--key`	server.key	Server certificate key file
`--cert`	server.crt	Server certificate file
`--disable-insecure`	false	Enforce server certificate validation

Metric Creation

Metrics can be created via POST requests to `/create` using either:

- Prometheus Client Data Exposition Format
- JSON format

Example JSON payload:

```
{
  "name": "my_metric",
  "help": "Description of metric",
  "type": "counter",
  "metrics": [
    {
      "labels": {
        "path": "/some/path",
        "receiver": "slack"
      },
      "value": "1"
    }
  ]
}
```

Technical Notes

- Uses goroutines for concurrent HTTP/HTTPS servers
- Implements custom Prometheus collector registration

- Supports metric label validation
- Includes request content-type validation
- Provides metric value caching
- Supports metric re-registration handling

Dependencies

- github.com/VictoriaMetrics/metrics
- github.com/prometheus/client_golang
- github.com/golang/gddo/httputil/header

Usage

Build and run:

```
go build
./metrics-service --port=8090 --port-tls=8443
```

Enable TLS with custom certificates:

```
./metrics-service --cert=path/to/cert.pem --key=path/to/key.pem
```

Development Notes

- Metric names must be valid Prometheus metric names
- Label names must be valid Prometheus label names
- TLS configuration is optional but recommended for production
- Webhook endpoints expect JSON payloads
- Metric creation validates label consistency

Security Considerations

- TLS support for encrypted communications
- Content-type validation on all endpoints
- Optional certificate validation enforcement = Request method validation