DevOpsSchool
Lets Learn, Share & Practice DevOps

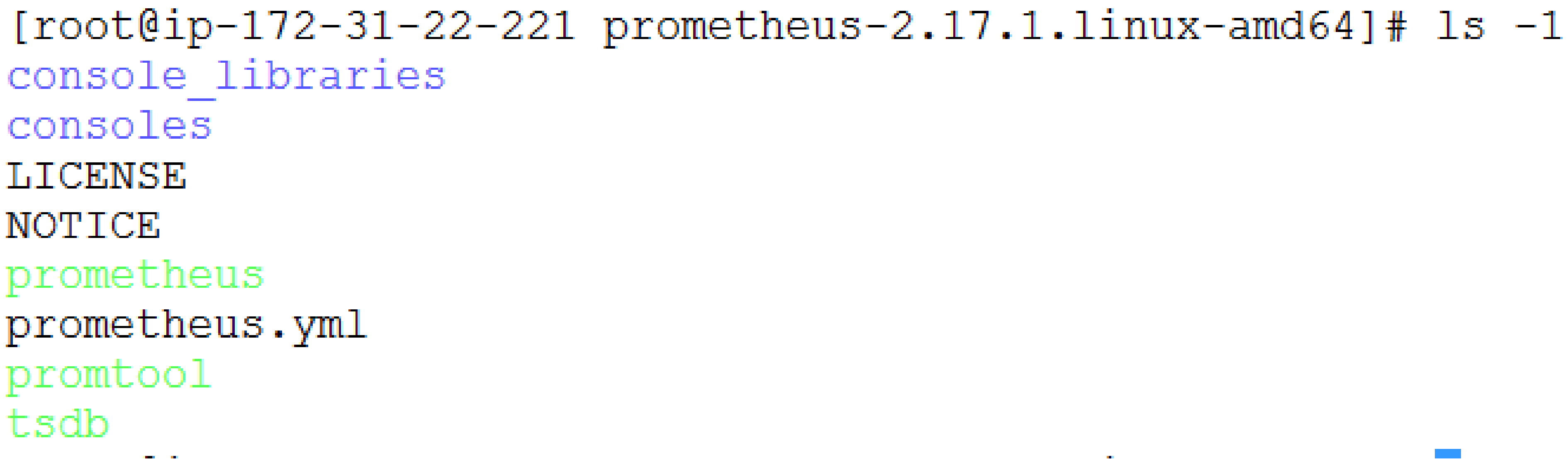
Prometheus Install & Configurations

Prometheus Installation

- You can download the **full distribution** from https://prometheus.io/download/
- **MacOs, Windows, Linux,** and some **Unix** distributions are supported
- After extracting you'll get a **prometheus executable** (Prometheus.exe for windows), which you can use to run Prometheus, for example:
- /prometheus --config.file /path/to/prometheus.yaml

Demo: Installing Prometheus

Installation files



```
[root@ip-172-31-22-221 prometheus-2.17.1.linux-amd64]# ls -1
console_libraries
consoles
LICENSE
NOTICE
prometheus
prometheus.yml
promtool
tsdb
```

Starting Prometheus

To start Prometheus with your newly created configuration file, change to the directory containing the Prometheus binary and run:

```
# Start Prometheus.
# By default, Prometheus stores its database in ./data (flag --storage.tsdb.path).
./prometheus --config.file=prometheus.yml
```

Accessing Prometheus

Prometheus should start up. You should also be able to browse to a status page about itself at localhost:9090. Give it a couple of seconds to collect data about itself from its own HTTP metrics endpoint.

You can also verify that Prometheus is serving metrics about itself by navigating to its metrics endpoint: localhost:9090/metrics

Demo: Node exporter

1. Node Exporter
2. Windows exporter
3. Apache exporter
4. MySQL server exporter
5. Docker Daemon

Demo: Go Instrumentations

Prometheus: Configurations

Prometheus is configured via
- Command-line flags and
- A configuration file.

While the **command-line flags** configure immutable system parameters (such as storage locations, amount of data to keep on disk and in memory, etc.),
The **configuration file** defines everything related to scraping jobs and their instances, as well as which rule files to load.

To view all available command-line flags, run **./prometheus -h.**

Prometheus Reload without command line restart
==

```
$ curl -i -XPOST localhost:9090/-/reload #  (when the --web.enable-lifecycle flag is enabled).
$ killall -HUP prometheus
$ sudo systemctl daemon-reload     #  (when the --web.enable-lifecycle flag is enabled).
$ sudo systemctl restart prometheus
```

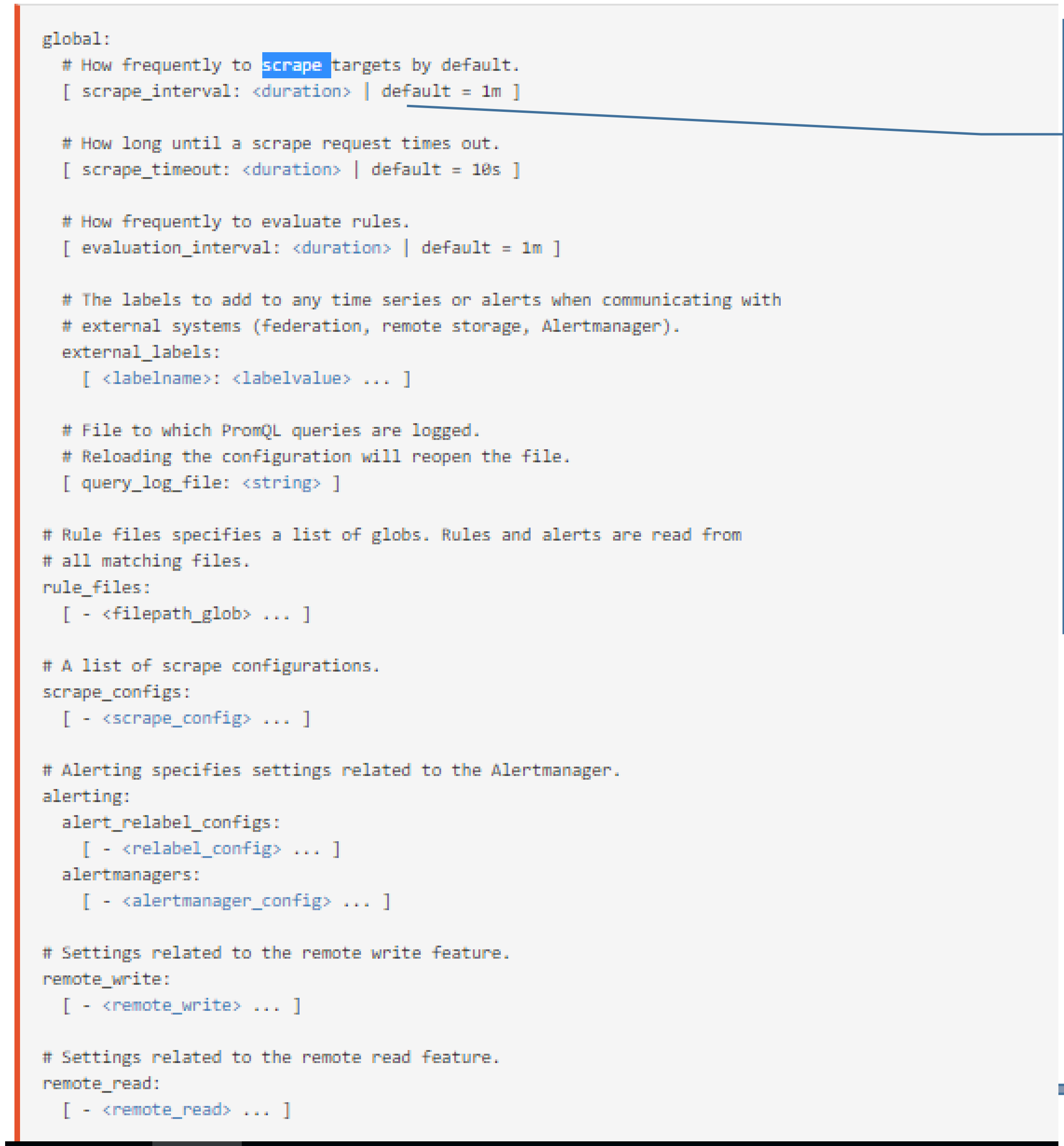
Configuration file

To specify which configuration file to load, use the --config.file flag.

The file is written in YAML format, defined by the scheme described below. Brackets indicate that a parameter is optional. For non-list parameters the value is set to the specified default.

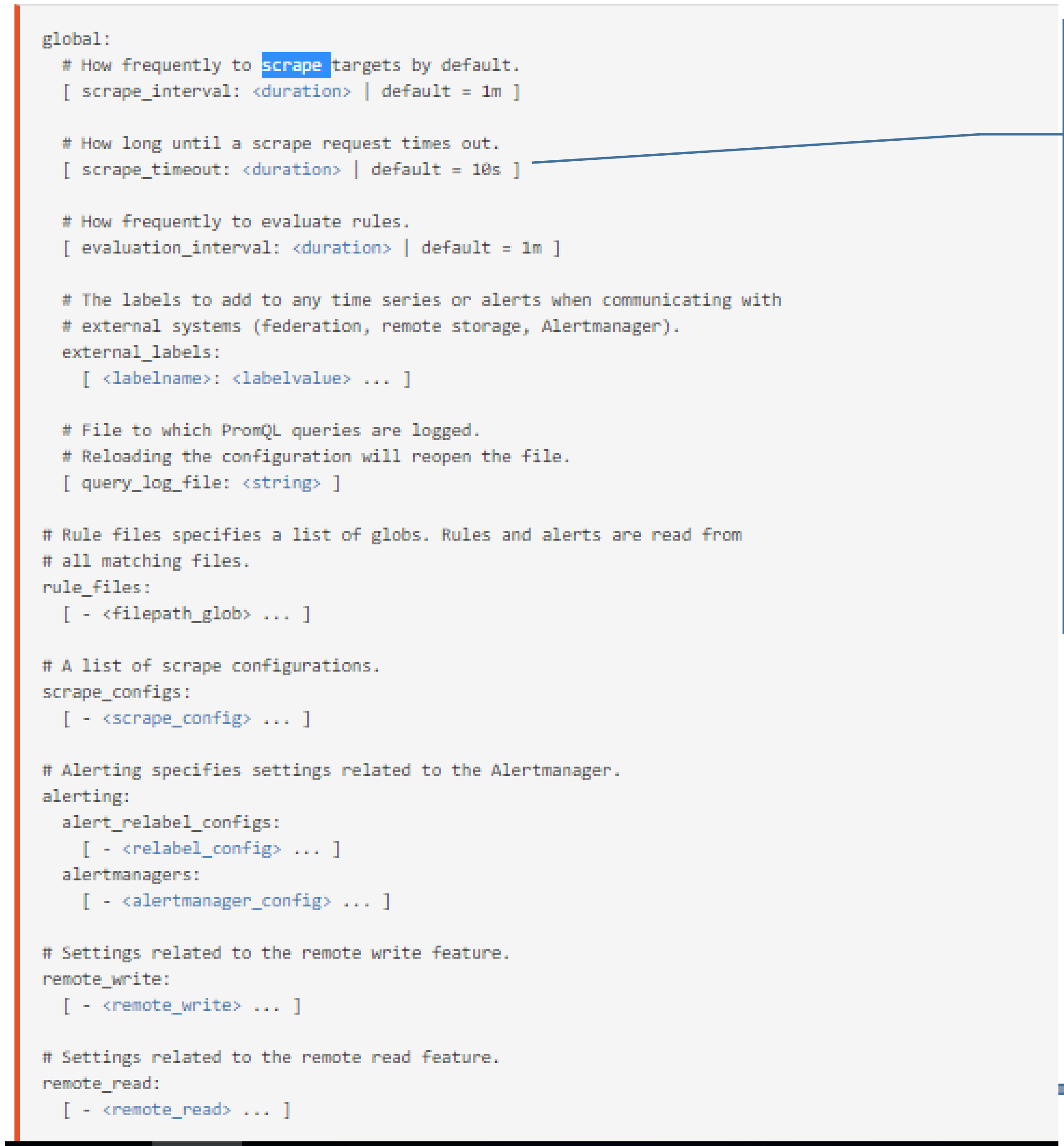
Generic placeholders are defined as follows:

<boolean>: a boolean that can take the values true or false
<duration>: a duration matching the regular expression [0-9]+(ms|[smhdwy])
<labelname>: a string matching the regular expression [a-zA-Z_][a-zA-Z0-9_]*
<labelvalue>: a string of unicode characters
<filename>: a valid path in the current working directory
<host>: a valid string consisting of a hostname or IP followed by an optional port number
<path>: a valid URL path
<scheme>: a string that can take the values http or https
<string>: a regular string
<secret>: a regular string that is a secret, such as a password
<tmpl_string>: a string which is template-expanded before usage



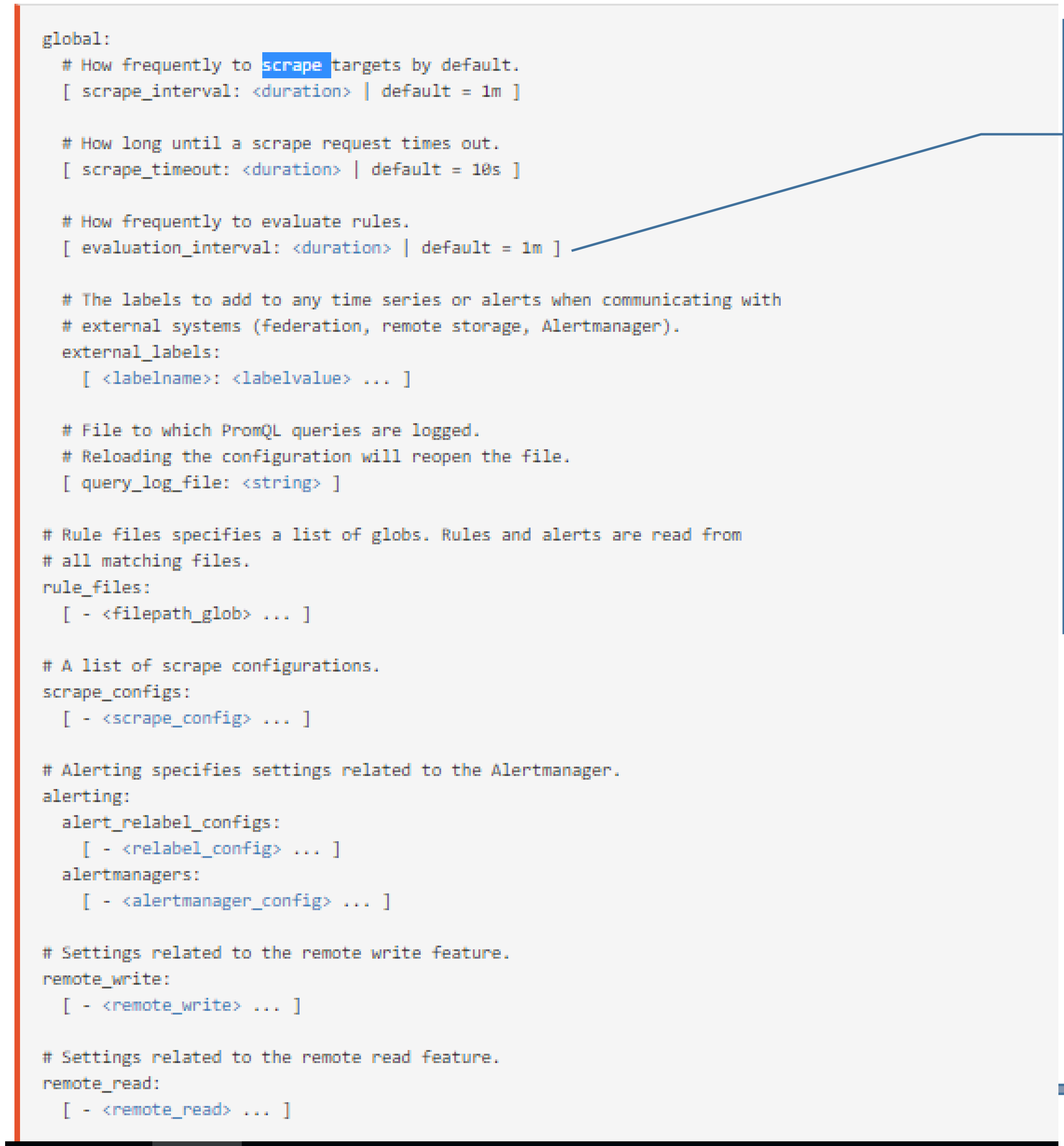
```
global:
  # How frequently to scrape targets by default.
  [ scrape_interval: <duration> | default = 1m ]

  # How long until a scrape request times out.
  [ scrape_timeout: <duration> | default = 10s ]

  # How frequently to evaluate rules.
  [ evaluation_interval: <duration> | default = 1m ]

  # The labels to add to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    [ <labelname>: <labelvalue> ... ]

  # File to which PromQL queries are logged.
  # Reloading the configuration will reopen the file.
  [ query_log_file: <string> ]

# Rule files specifies a list of globs. Rules and alerts are read from
# all matching files.
rule_files:
  [ - <filepath_glob> ... ]

# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]

# Alerting specifies settings related to the Alertmanager.
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]

# Settings related to the remote read feature.
remote_read:
  [ - <remote_read> ... ]
```

scrape_interval
How frequently to scrape targets by default. default = 1m



```
global:
  # How frequently to scrape targets by default.
  [ scrape_interval: <duration> | default = 1m ]

  # How long until a scrape request times out.
  [ scrape_timeout: <duration> | default = 10s ]

  # How frequently to evaluate rules.
  [ evaluation_interval: <duration> | default = 1m ]

  # The labels to add to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    [ <labelname>: <labelvalue> ... ]

  # File to which PromQL queries are logged.
  # Reloading the configuration will reopen the file.
  [ query_log_file: <string> ]

# Rule files specifies a list of globs. Rules and alerts are read from
# all matching files.
rule_files:
  [ - <filepath_glob> ... ]

# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]

# Alerting specifies settings related to the Alertmanager.
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]

# Settings related to the remote read feature.
remote_read:
  [ - <remote_read> ... ]
```

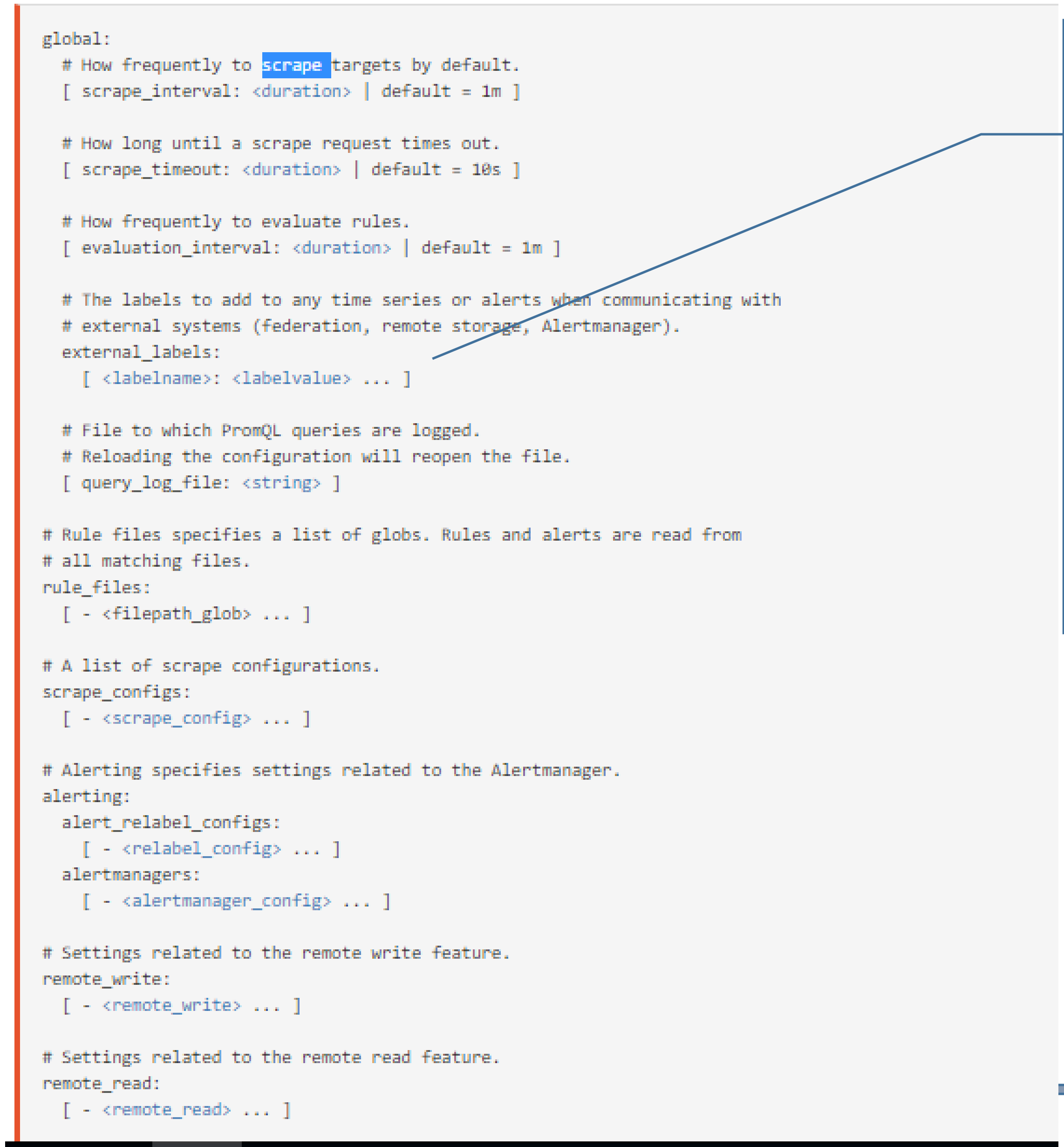
scrape_timeout
How long until a scrape request times out. default = 10s



```
global:
  # How frequently to scrape targets by default.
  [ scrape_interval: <duration> | default = 1m ]

  # How long until a scrape request times out.
  [ scrape_timeout: <duration> | default = 10s ]

  # How frequently to evaluate rules.
  [ evaluation_interval: <duration> | default = 1m ]

  # The labels to add to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    [ <labelname>: <labelvalue> ... ]

  # File to which PromQL queries are logged.
  # Reloading the configuration will reopen the file.
  [ query_log_file: <string> ]

# Rule files specifies a list of globs. Rules and alerts are read from
# all matching files.
rule_files:
  [ - <filepath_glob> ... ]

# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]

# Alerting specifies settings related to the Alertmanager.
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]

# Settings related to the remote read feature.
remote_read:
  [ - <remote_read> ... ]
```

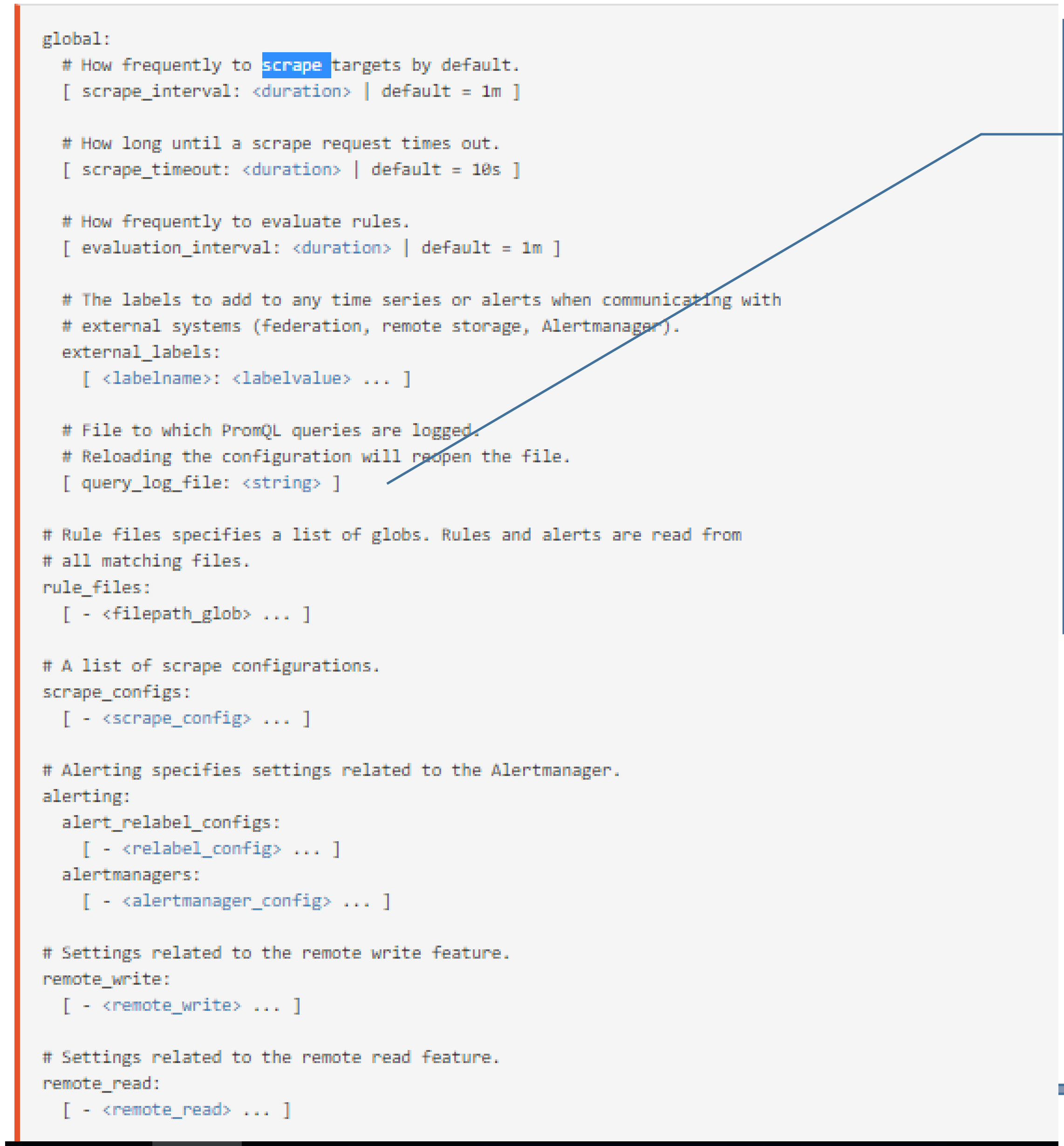
evaluation_interval
How frequently to evaluate rules. default = 1m

DevOpsSchool
Lets Learn, Share & Practice DevOps



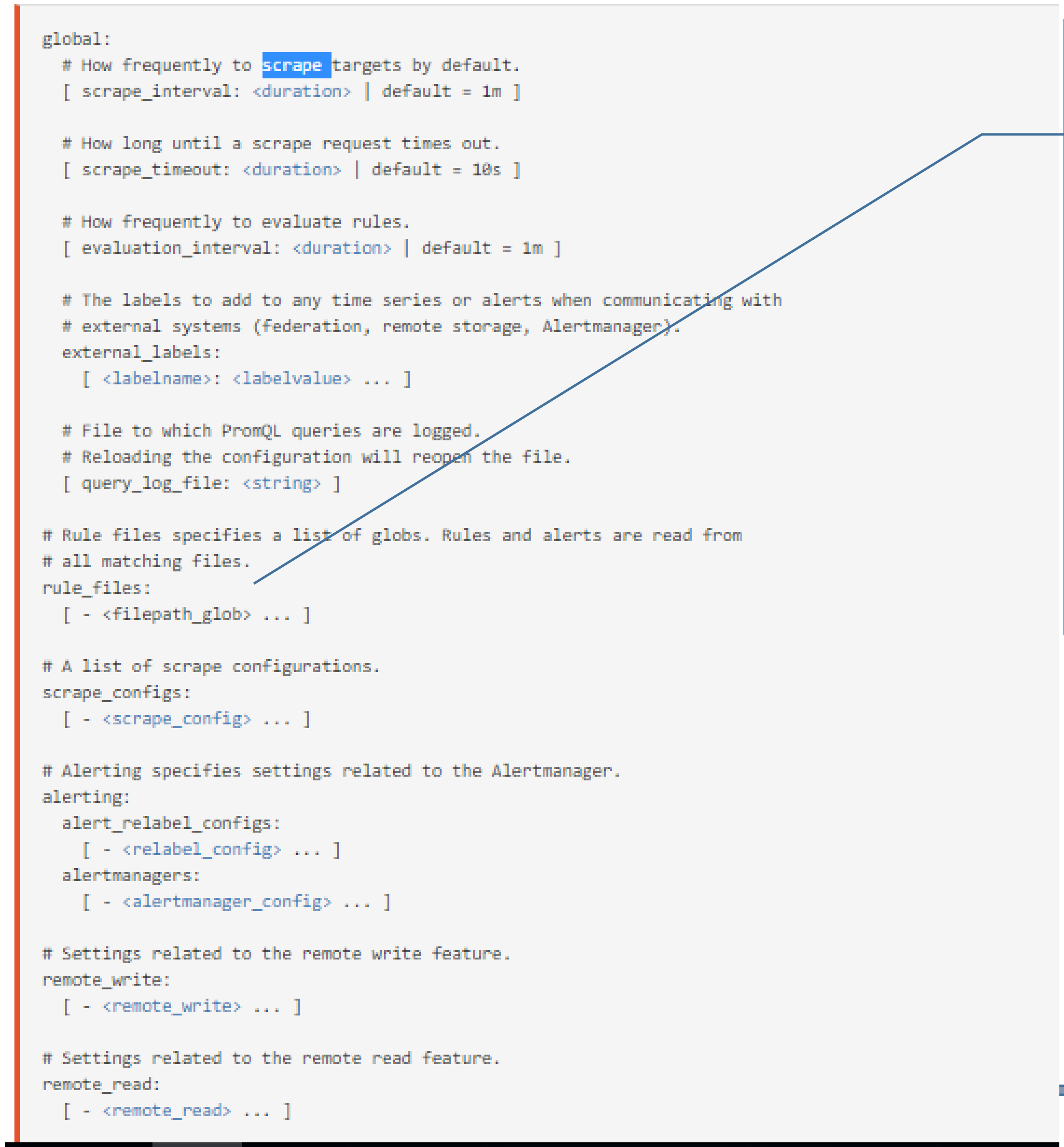
```
global:
  # How frequently to scrape targets by default.
  [ scrape_interval: <duration> | default = 1m ]

  # How long until a scrape request times out.
  [ scrape_timeout: <duration> | default = 10s ]

  # How frequently to evaluate rules.
  [ evaluation_interval: <duration> | default = 1m ]

  # The labels to add to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    [ <labelname>: <labelvalue> ... ]

  # File to which PromQL queries are logged.
  # Reloading the configuration will reopen the file.
  [ query_log_file: <string> ]

# Rule files specifies a list of globs. Rules and alerts are read from
# all matching files.
rule_files:
  [ - <filepath_glob> ... ]

# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]

# Alerting specifies settings related to the Alertmanager.
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]

# Settings related to the remote read feature.
remote_read:
  [ - <remote_read> ... ]
```

external_labels:
The labels to add to any time series or alerts when communicating with external systems (federation, remote storage, Alertmanager).



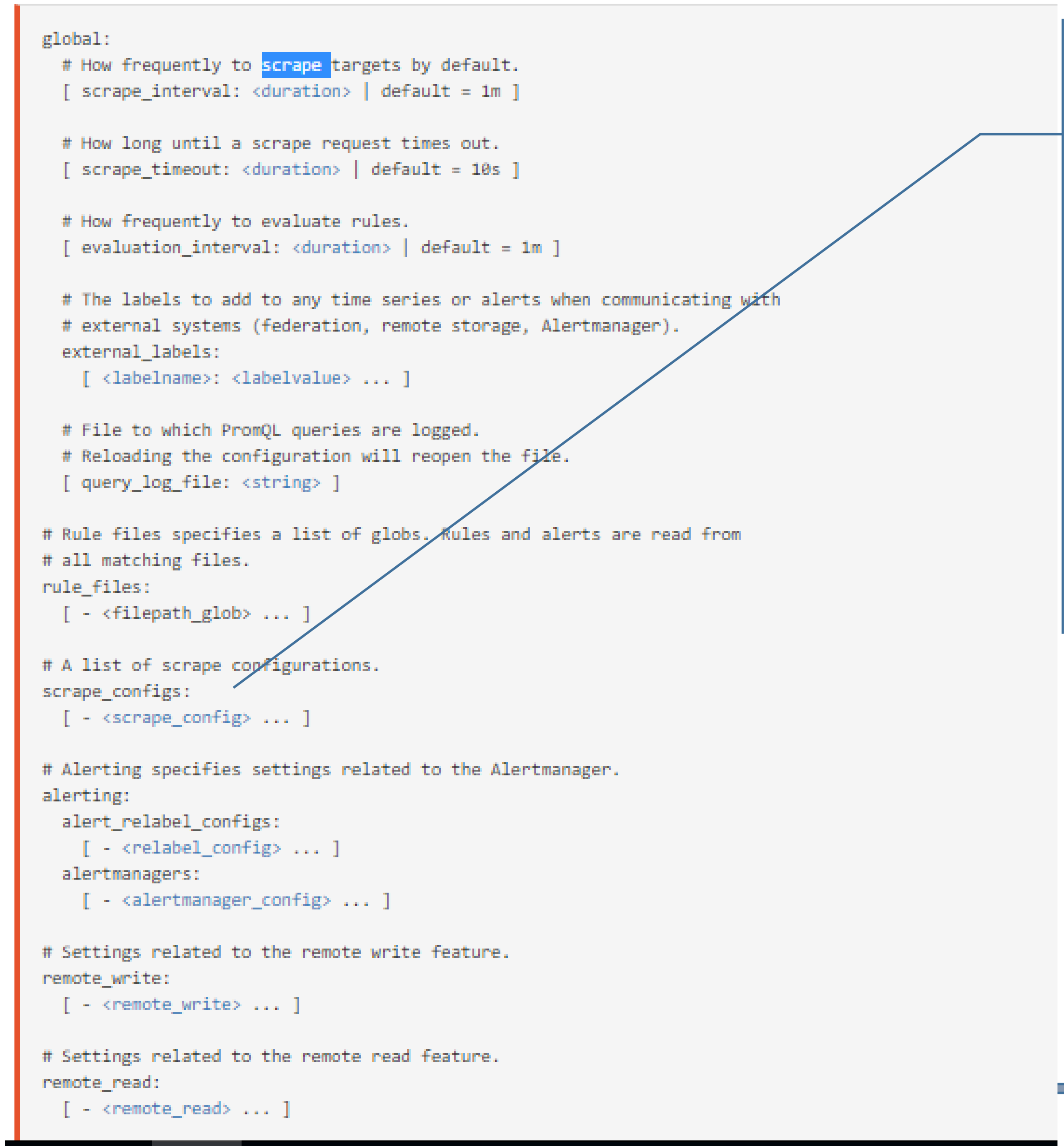
```
global:
  # How frequently to scrape targets by default.
  [ scrape_interval: <duration> | default = 1m ]

  # How long until a scrape request times out.
  [ scrape_timeout: <duration> | default = 10s ]

  # How frequently to evaluate rules.
  [ evaluation_interval: <duration> | default = 1m ]

  # The labels to add to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    [ <labelname>: <labelvalue> ... ]

  # File to which PromQL queries are logged.
  # Reloading the configuration will reopen the file.
  [ query_log_file: <string> ]

# Rule files specifies a list of globs. Rules and alerts are read from
# all matching files.
rule_files:
  [ - <filepath_glob> ... ]

# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]

# Alerting specifies settings related to the Alertmanager.
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]

# Settings related to the remote read feature.
remote_read:
  [ - <remote_read> ... ]
```

query_log_file
File to which PromQL queries are logged. Reloading the configuration will reopen the file.



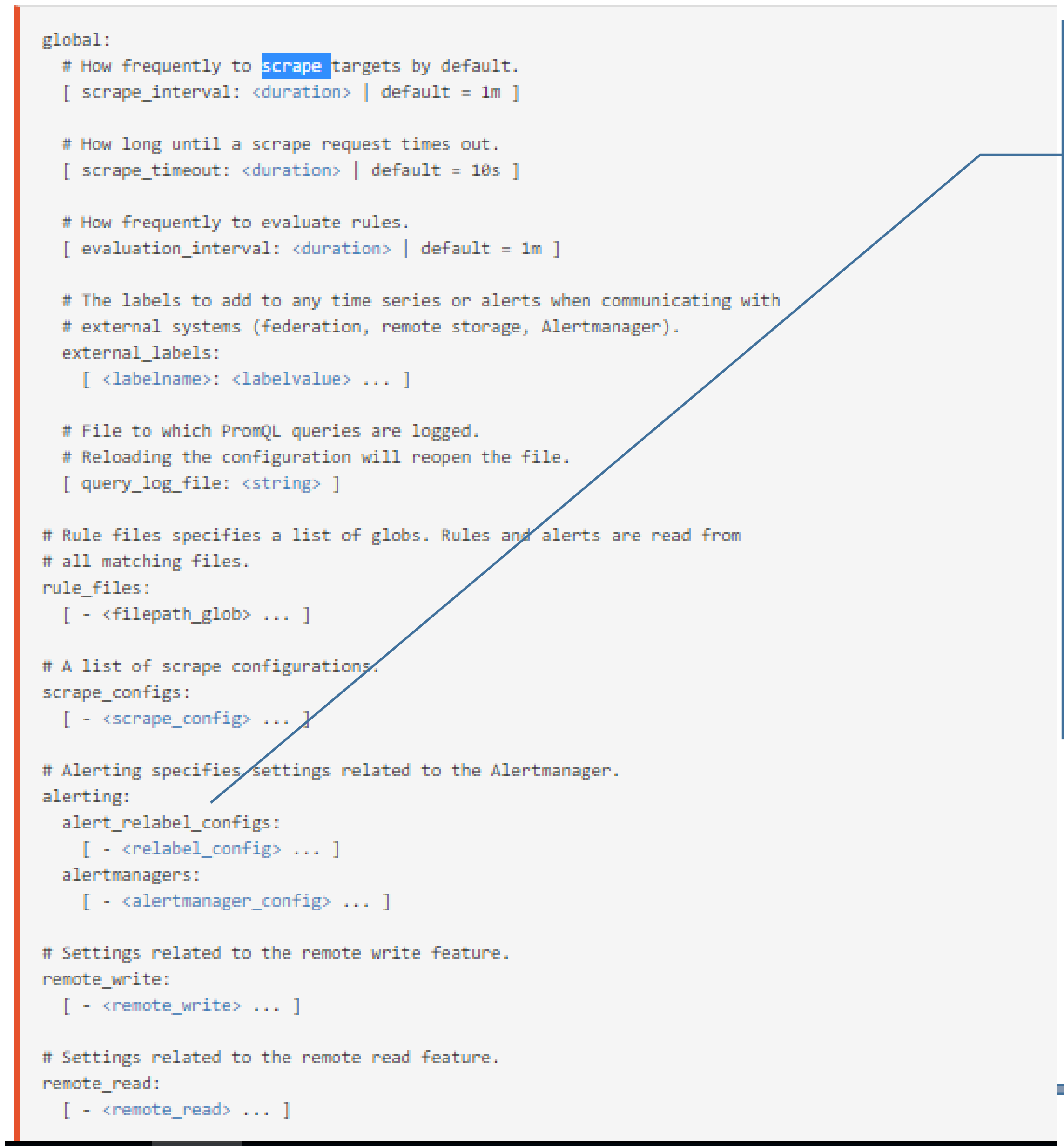
```
global:
  # How frequently to scrape targets by default.
  [ scrape_interval: <duration> | default = 1m ]

  # How long until a scrape request times out.
  [ scrape_timeout: <duration> | default = 10s ]

  # How frequently to evaluate rules.
  [ evaluation_interval: <duration> | default = 1m ]

  # The labels to add to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    [ <labelname>: <labelvalue> ... ]

  # File to which PromQL queries are logged.
  # Reloading the configuration will reopen the file.
  [ query_log_file: <string> ]

# Rule files specifies a list of globs. Rules and alerts are read from
# all matching files.
rule_files:
  [ - <filepath_glob> ... ]

# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]

# Alerting specifies settings related to the Alertmanager.
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]

# Settings related to the remote read feature.
remote_read:
  [ - <remote_read> ... ]
```

rule_files
Rule files specifies a list of globs. Rules and alerts are read from all matching files.



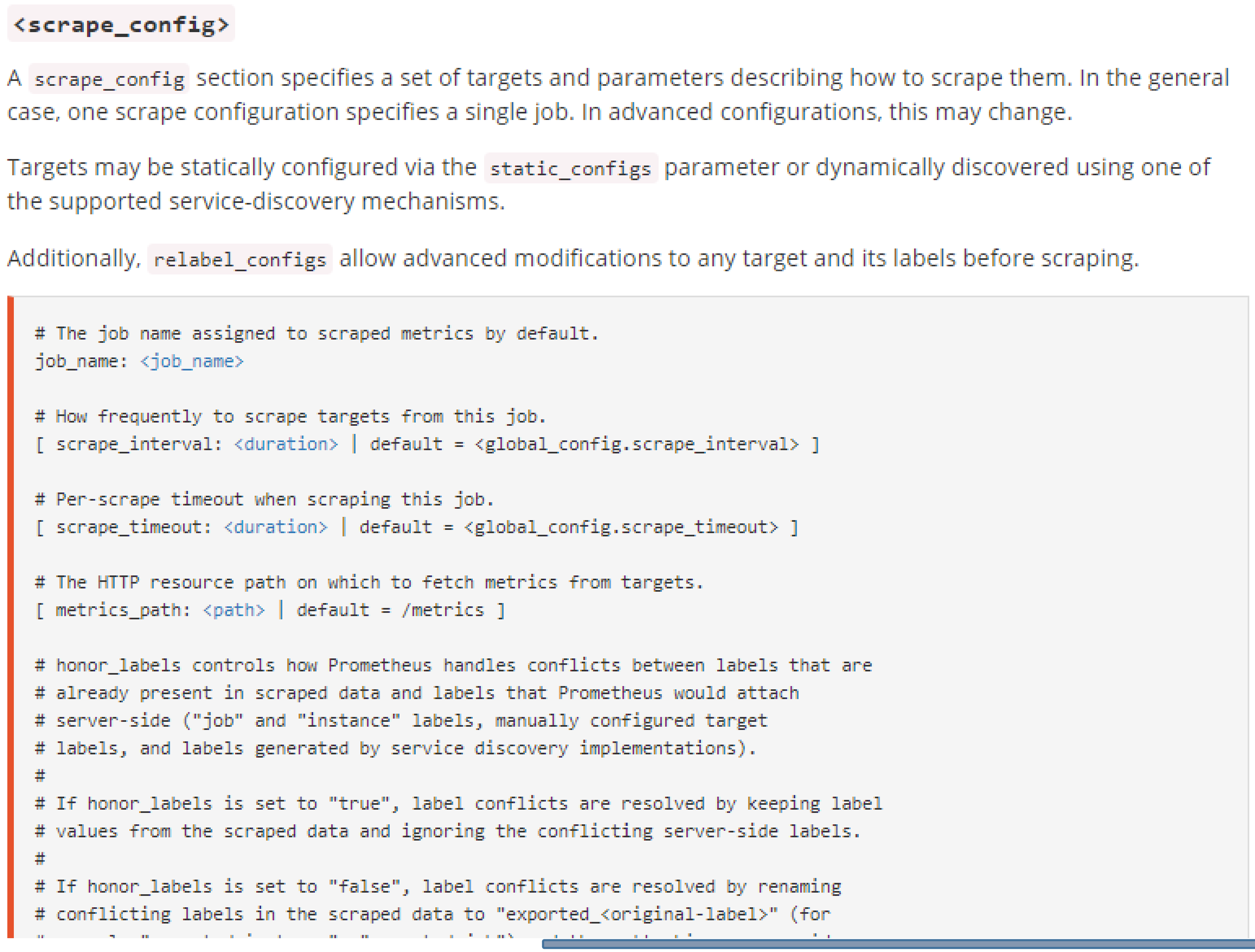
```
global:
  # How frequently to scrape targets by default.
  [ scrape_interval: <duration> | default = 1m ]

  # How long until a scrape request times out.
  [ scrape_timeout: <duration> | default = 10s ]

  # How frequently to evaluate rules.
  [ evaluation_interval: <duration> | default = 1m ]

  # The labels to add to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    [ <labelname>: <labelvalue> ... ]

  # File to which PromQL queries are logged.
  # Reloading the configuration will reopen the file.
  [ query_log_file: <string> ]

# Rule files specifies a list of globs. Rules and alerts are read from
# all matching files.
rule_files:
  [ - <filepath_glob> ... ]

# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]

# Alerting specifies settings related to the Alertmanager.
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]

# Settings related to the remote read feature.
remote_read:
  [ - <remote_read> ... ]
```

scrape_configs
A list of scrape configurations.


```
global:
  # How frequently to scrape targets by default.
  [ scrape_interval: <duration> | default = 1m ]

  # How long until a scrape request times out.
  [ scrape_timeout: <duration> | default = 10s ]

  # How frequently to evaluate rules.
  [ evaluation_interval: <duration> | default = 1m ]

  # The labels to add to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    [ <labelname>: <labelvalue> ... ]

  # File to which PromQL queries are logged.
  # Reloading the configuration will reopen the file.
  [ query_log_file: <string> ]

# Rule files specifies a list of globs. Rules and alerts are read from
# all matching files.
rule_files:
  [ - <filepath_glob> ... ]

# A list of scrape configurations.
scrape_configs:
  [ - <scrape_config> ... ]

# Alerting specifies settings related to the Alertmanager.
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

# Settings related to the remote write feature.
remote_write:
  [ - <remote_write> ... ]

# Settings related to the remote read feature.
remote_read:
  [ - <remote_read> ... ]
```

```
alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]
```

Alerting specifies settings related to the Alertmanager.



`<scrape_config>`

A `scrape_config` section specifies a set of targets and parameters describing how to scrape them. In the general case, one scrape configuration specifies a single job. In advanced configurations, this may change.

Targets may be statically configured via the `static_configs` parameter or dynamically discovered using one of the supported service-discovery mechanisms.

Additionally, `relabel_configs` allow advanced modifications to any target and its labels before scraping.

```
# The job name assigned to scraped metrics by default.
job_name: <job_name>

# How frequently to scrape targets from this job.
[ scrape_interval: <duration> | default = <global_config.scrape_interval> ]

# Per-scrape timeout when scraping this job.
[ scrape_timeout: <duration> | default = <global_config.scrape_timeout> ]

# The HTTP resource path on which to fetch metrics from targets.
[ metrics_path: <path> | default = /metrics ]

# honor_labels controls how Prometheus handles conflicts between labels that are
# already present in scraped data and labels that Prometheus would attach
# server-side ("job" and "instance" labels, manually configured target
# labels, and labels generated by service discovery implementations).
#
# If honor_labels is set to "true", label conflicts are resolved by keeping label
# values from the scraped data and ignoring the conflicting server-side labels.
#
# If honor_labels is set to "false", label conflicts are resolved by renaming
# conflicting labels in the scraped data to "exported_<original-label>" (for
Flags:
  -h, --help                 Show context-sensitive help (also try --help-long and --help-man).
      --version              Show application version.
      --config.file="prometheus.yml"
                             Prometheus configuration file path.
      --web.listen-address="0.0.0.0:9090"
                             Address to listen on for UI, API, and telemetry.
      --web.read-timeout=5m      Maximum duration before timing out read of the request, and closing idle connections.
      --web.max-connections=512  Maximum number of simultaneous connections.
      --web.external-url=<URL>   The URL under which Prometheus is externally reachable (for example, if Prometheus is
                             served via a reverse proxy). Used for generating relative and absolute links back to
                             Prometheus itself. If the URL has a path portion, it will be used to prefix all HTTP
                             endpoints served by Prometheus. If omitted, relevant URL components will be derived
                             automatically.
      --web.route-prefix=<path>  Prefix for the internal routes of web endpoints. Defaults to path of --web.external-url.
      --web.user-assets=<path>   Path to static asset directory, available at /user.
      --web.enable-lifecycle     Enable shutdown and reload via HTTP request.
      --web.enable-admin-api     Enable API endpoints for admin control actions.
```

```
--web.console.templates="consoles"
                         Path to the console template directory, available at /consoles.
    --web.console.libraries="console_libraries"
                         Path to the console library directory.
    --web.page-title="Prometheus Time Series Collection and Processing Server"
                         Document title of Prometheus instance.
    --web.cors.origin=".*"     Regex for CORS origin. It is fully anchored. Example: 'https?://(domain1|domain2)\.com'
    --storage.tsdb.path="data/"
                         Base path for metrics storage.
    --storage.tsdb.retention=STORAGE.TSDB.RETENTION
                         [DEPRECATED] How long to retain samples in storage. This flag has been deprecated, use
                         "storage.tsdb.retention.time" instead.
    --storage.tsdb.retention.time=STORAGE.TSDB.RETENTION.TIME
                         How long to retain samples in storage. When this flag is set it overrides
                         "storage.tsdb.retention". If neither this flag nor "storage.tsdb.retention" nor
                         "storage.tsdb.retention.size" is set, the retention time defaults to 15d. Units
                         Supported: y, w, d, h, m, s, ms.
```

```
--storage.tsdb.retention.size=STORAGE.TSDB.RETENTION.SIZE
                          [EXPERIMENTAL] Maximum number of bytes that can be stored for blocks. Units supported:
                          KB, MB, GB, TB, PB. This flag is experimental and can be changed in future releases.
    --storage.tsdb.no-lockfile
                          Do not create lockfile in data directory.
    --storage.tsdb.allow-overlapping-blocks
                          [EXPERIMENTAL] Allow overlapping blocks, which in turn enables vertical compaction and
                          vertical query merge.
    --storage.tsdb.wal-compression
                          Compress the tsdb WAL.
    --storage.remote.flush-deadline=<duration>
                          How long to wait flushing sample on shutdown or config reload.
    --storage.remote.read-sample-limit=5e7
                          Maximum overall number of samples to return via the remote read interface, in a single
                          query. 0 means no limit. This limit is ignored for streamed response types.
    --storage.remote.read-concurrent-limit=10
```

```
--storage.remote.read-concurrent-limit=10
                          Maximum number of concurrent remote read calls. 0 means no limit.
    --storage.remote.read-max-bytes-in-frame=1048576
                          Maximum number of bytes in a single frame for streaming remote read response types before
                          marshalling. Note that client might have limit on frame size as well. 1MB as recommended
                          by protobuf by default.
    --rules.alert.for-outage-tolerance=1h
                          Max time to tolerate prometheus outage for restoring "for" state of alert.
    --rules.alert.for-grace-period=10m
                          Minimum duration between alert and restored "for" state. This is maintained only for
                          alerts with configured "for" time greater than grace period.
    --rules.alert.resend-delay=1m
                          Minimum amount of time to wait before resending an alert to Alertmanager.
```

```
--alertmanager.notification-queue-capacity=10000
                          The capacity of the queue for pending Alertmanager notifications.
    --alertmanager.timeout=10s
                          Timeout for sending alerts to Alertmanager.
    --query.lookback-delta=5m  The maximum lookback duration for retrieving metrics during expression evaluations and
                          federation.
    --query.timeout=2m        Maximum time a query may take before being aborted.
    --query.max-concurrency=20
                          Maximum number of queries executed concurrently.
    --query.max-samples=50000000
                          Maximum number of samples a single query can load into memory. Note that queries will
                          fail if they try to load more samples than this into memory, so this also limits the
                          number of samples a query can return.
    --log.level=info          Only log messages with the given severity or above. One of: [debug, info, warn, error]
    --log.format=logfmt       Output format of log messages. One of: [logfmt, json]
```

Client Libraries – Golang Example

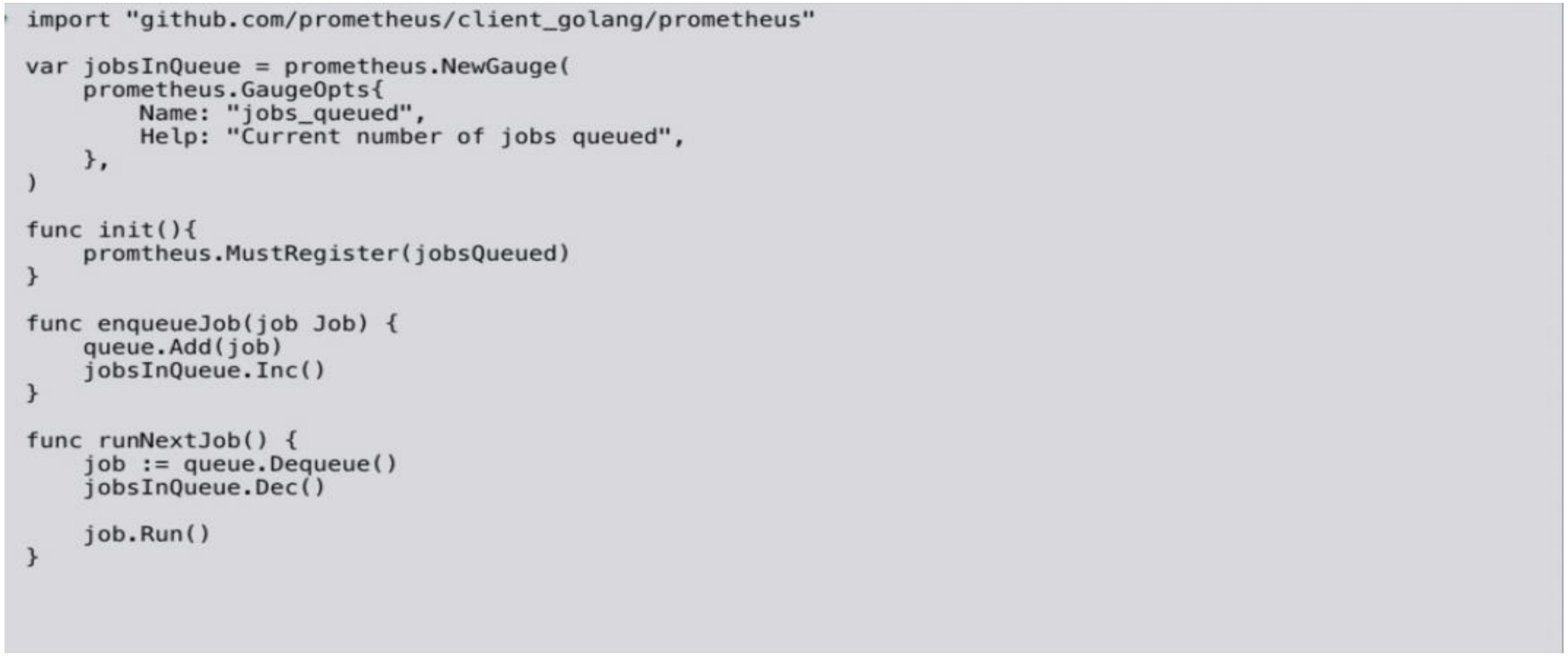
- https://github.com/prometheus/client_golang
- Officially supported language
- Easy to implement:

```
package main

import (
        "github.com/prometheus/client_golang/prometheus/promhttp"
        "net/http"
)

func main() {
        http.Handle("/metrics", promhttp.Handler())
        panic(http.ListenAndServe(":8080", nil))
}
```

- Supported metric: Counter, Ganga, Summary and Histogram

Client Libraries – Golang Example

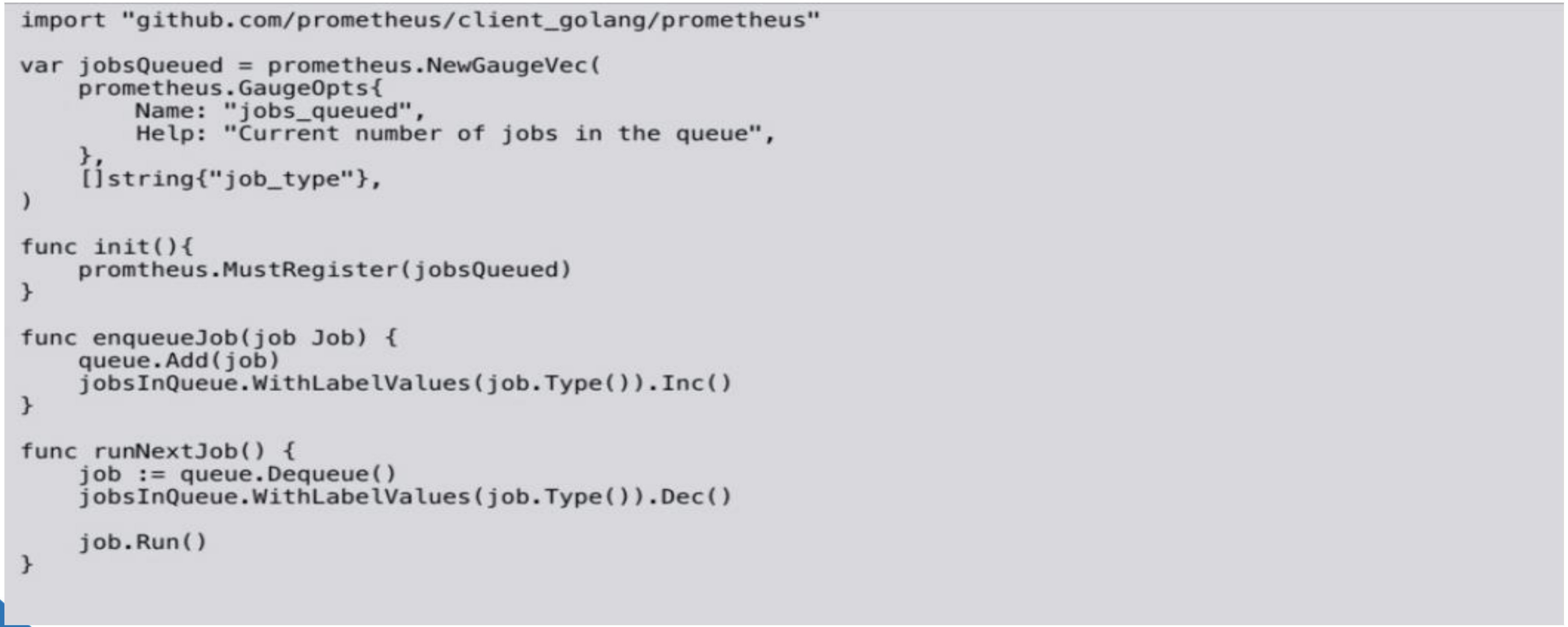
- Gauge



```
import "github.com/prometheus/client_golang/prometheus"

var jobsInQueue = prometheus.NewGauge(
    prometheus.GaugeOpts{
        Name: "jobs_queued",
        Help: "Current number of jobs queued",
    },
)

func init(){
    promtheus.MustRegister(jobsQueued)
}

func enqueueJob(job Job) {
    queue.Add(job)
    jobsInQueue.Inc()
}

func runNextJob() {
    job := queue.Dequeue()
    jobsInQueue.Dec()

    job.Run()
}
```

Client Libraries – Golang Example

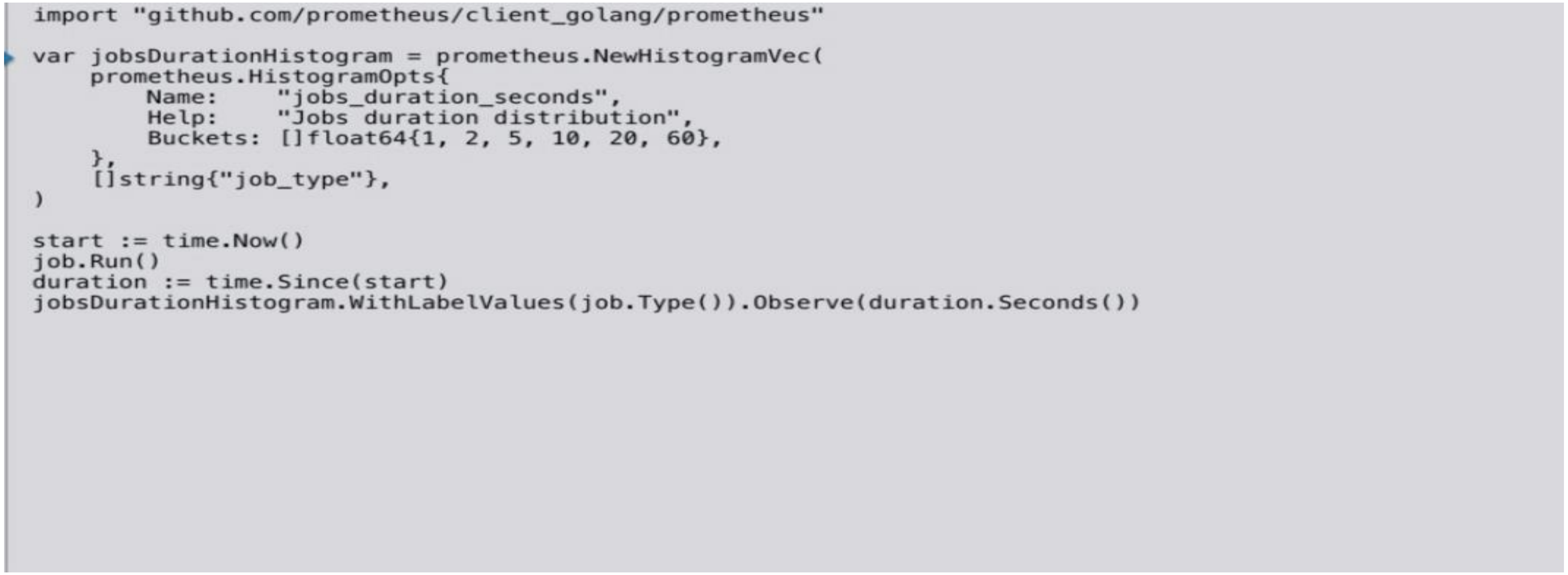
- Adding labels



```
import "github.com/prometheus/client_golang/prometheus"

var jobsQueued = prometheus.NewGaugeVec(
    prometheus.GaugeOpts{
        Name: "jobs_queued",
        Help: "Current number of jobs in the queue",
    },
    []string{"job_type"},
)

func init(){
    promtheus.MustRegister(jobsQueued)
}

func enqueueJob(job Job) {
    queue.Add(job)
    jobsInQueue.WithLabelValues(job.Type()).Inc()
}

func runNextJob() {
    job := queue.Dequeue()
    jobsInQueue.WithLabelValues(job.Type()).Dec()

    job.Run()
}
```

Client Libraries – Golang Example

- Histogram



```
import "github.com/prometheus/client_golang/prometheus"

var jobsDurationHistogram = prometheus.NewHistogramVec(
    prometheus.HistogramOpts{
        Name:    "jobs_duration_seconds",
        Help:    "Jobs duration distribution",
        Buckets: []float64{1, 2, 5, 10, 20, 60},
    },
    []string{"job_type"},
)

start := time.Now()
job.Run()
duration := time.Since(start)
jobsDurationHistogram.WithLabelValues(job.Type()).Observe(duration.Seconds())
```

Client Libraries – Golang Example

- Summary

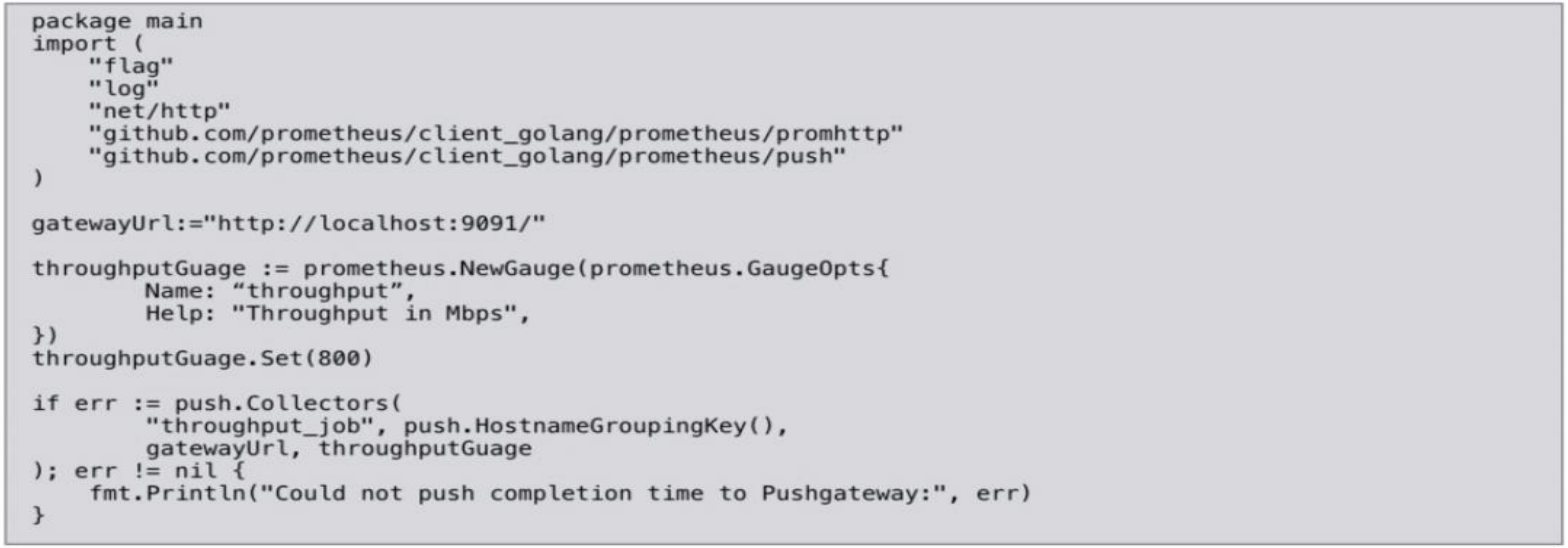
```
prometheus.NewSummary()
```

Prometheus

Pushing Metrics - Go

Pushing Metrics – Go Example

- Go example:



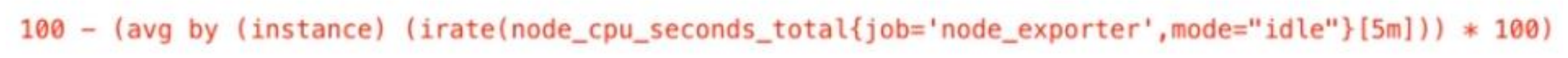
```
package main
import (
    "flag"
    "log"
    "net/http"
    "github.com/prometheus/client_golang/prometheus/promhttp"
    "github.com/prometheus/client_golang/prometheus/push"
)

gatewayUrl:="http://localhost:9091/"

throughputGuage := prometheus.NewGauge(prometheus.GaugeOpts{
        Name: “throughput”,
        Help: "Throughput in Mbps",
})
throughputGuage.Set(800)

if err := push.Collectors(
        "throughput_job", push.HostnameGroupingKey(),
        gatewayUrl, throughputGuage
); err != nil {
    fmt.Println("Could not push completion time to Pushgateway:", err)
}
```

Prometheus

Querying

Querying Metrics – Introduction

- Prometheus provides a functional expression language called PromQL
  - Provides built in operators and functions
  - Vector-based calculation like Excel
  - Expressions over time-series vectors
  - PromQL is read-only
  - Example:

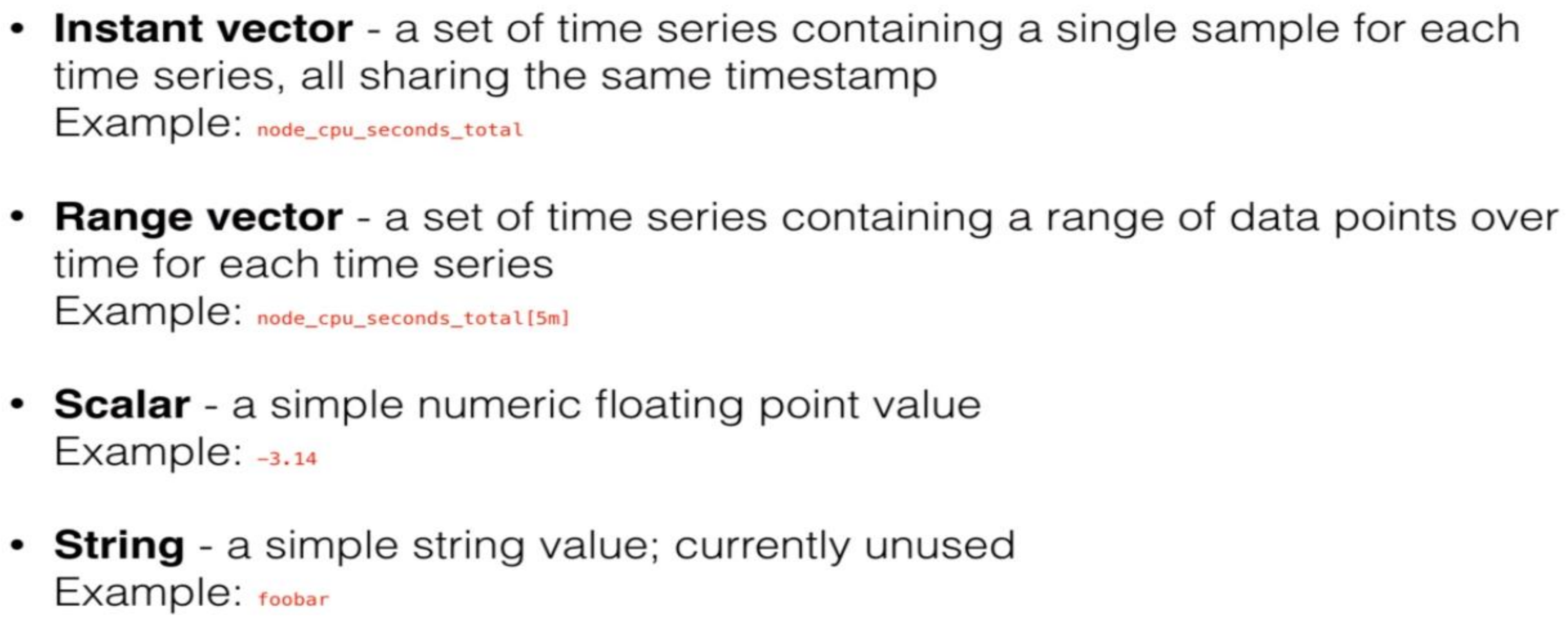


```
100 - (avg by (instance) (irate(node_cpu_seconds_total{job='node_exporter',mode="idle"}[5m])) * 100)
```

Prometheus

Querying - Expressions

Querying Metrics – Introduction



- **Instant vector** - a set of time series containing a single sample for each time series, all sharing the same timestamp
  Example: `node_cpu_seconds_total`

- **Range vector** - a set of time series containing a range of data points over time for each time series
  Example: `node_cpu_seconds_total[5m]`

- **Scalar** - a simple numeric floating point value
  Example: `-3.14`

- **String** - a simple string value; currently unused
  Example: `foobar`

Demo

Querying