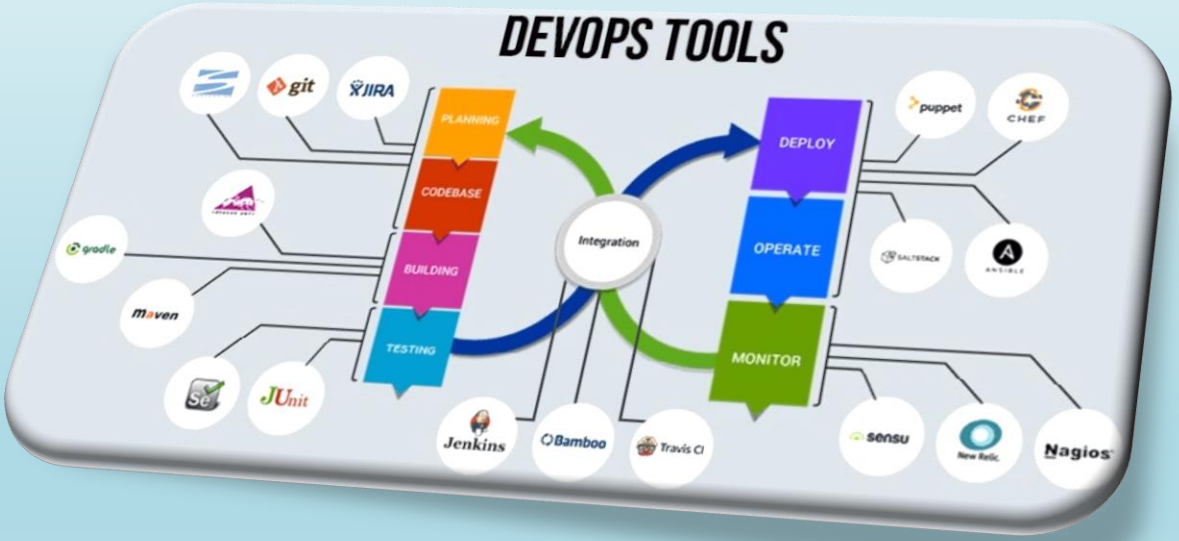
CLOUD TRAIN
ACCELERATE YOUR GROWTH

Continuous Monitoring
[Prometheus + Grafana]



Agenda

- WHAT IS PROMETHEUS
- PROMETHEUS ARCHITECTURE
- PROMETHEUS INSTALLATION & CONFIGURATION
- MONITORING NODES WITH EXPORTERS
- PROMETHEUS METRICS QUERYING
- ALERTING WITH PROMETHEUS
- CLIENT LIBRARIES
- GRAFANA INSTALLATION & CONFIGURATION

What is Prometheus?

What is Prometheus?

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Prometheus is an **Open-source monitoring** solution

- Started at SoundCloud around 2012-2013, and was made public in early 2015
- Prometheus provides **Metrics & Alerting**
- It is inspired by Google's **Borgmon**, which uses time-series data as a datasource, to then send alerts based on this data
- It fits very well in the **cloud native infrastructure**
- Prometheus is also a member of the **CNCF (Cloud Native Foundation)**
- Prometheus includes a Flexible **Query Language - PromQL**
- **Visualizations** can be shown using a built-in expression browser or with integrations like Grafana
- It stores metrics in **memory** and **local disk** in an own **custom, efficient format**
- It is written in **Go** and supports Many **client libraries** and **integrations available**

Prometheus Concepts

CLOUD TRAIN
ACCELERATE YOUR GROWTH

In Prometheus we talk about **Dimensional Data**: time series are identified by metric name and a set of key/value pairs

Prometheus collects metrics from monitored targets by **scraping metrics HTTP endpoints**

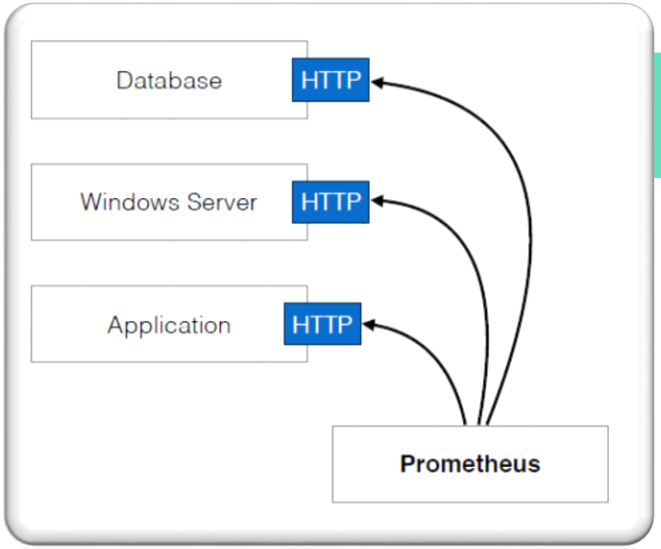
This is fundamentally different than other monitoring and alerting systems, (except this is also how Google's Borgmon works)

Rather than using custom scripts that check on particular services and systems, the **monitoring data itself is used**

Scraping endpoints is much more efficient than other mechanisms, like 3rd party agents

A single prometheus server is able to ingest up to one million samples per second as several million time series

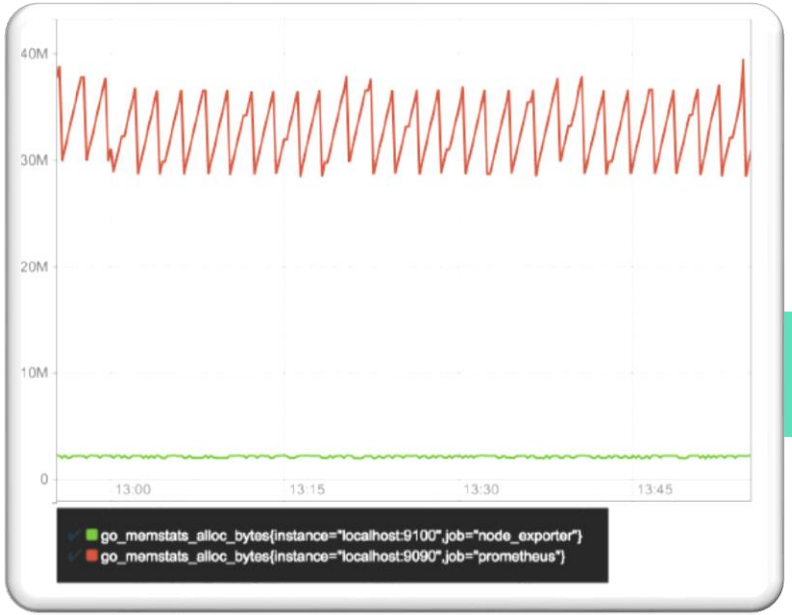
Metric name	Label	Sample
Temperature	location=outside	90



Prometheus Concepts

- All data is stored as time series
  - Every time series is identified by the "**metric name**" and a set of **key-value pairs,** called **labels**
    - **metric:** go_memstat_alloc_bytes
      - instance=localhost:9090
      - job=prometheus

- The time series data also consists of the **actual data**, called **Samples**:
  - It can be a **float64** value, or
  - a **millisecond-precision timestamp**



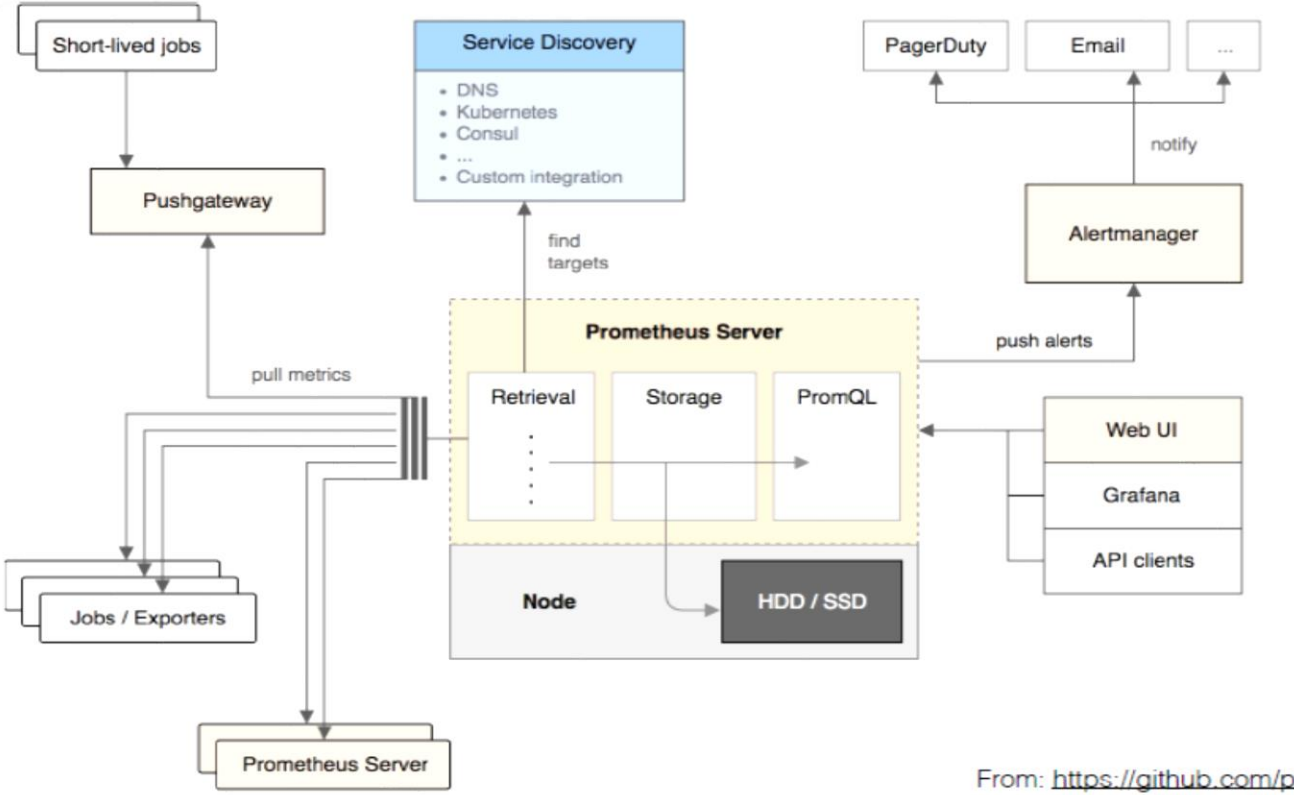
Graph Console

Element	Value
go_memstats_alloc_bytes{instance="localhost:9090",job="prometheus"}	30950392
go_memstats_alloc_bytes{instance="localhost:9100",job="node_exporter"}	2251192

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Prometheus Architecture

Prometheus Architecture



From: https://github.com/prometheus/prometheus

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Prometheus Installation

Installing Prometheus

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Download the **full distribution** from below URL:

- https://github.com/prometheus/prometheus/releases

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

Not secure | 34.125.4.246:9090/graph?g0.expr=up%7Bjob%3D"prometheus"%7D&g0.tab=1&g0.stacked=0&g0.show_exemplars=0&g0.r...

Prometheus Alerts Graph Status Help

Use local time | Enable query history | Enable autocomplete | Enable highlighting | Enable linter

```
up{job="prometheus"}
```

Execute

Load time: 190ms Resolution: 14s Result series: 1

Table Graph

Evaluation time

up{instance="localhost:9090", job="prometheus"} 1

Remove Panel

Add Panel

Prometheus Alerts Graph Status Help

Use local time | Enable query history | Enable autocomplete | Enable highlighting | Enable linter

```
http_requests_total
```

Execute

Table Graph

1h End time Res. (s) Show Exemplars

Prometheus Configuration

Prometheus Configuration

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Prometheus Configuration file

The **configuration** is stored in the Prometheus configuration file, in yaml format

- The configuration file can be **changed and applied**, without having to restart Prometheus
  - A **reload** can be done by executing kill -SIGHUP <pid>

You can also pass parameters (flags) at **startup time** to ./Prometheus

- Those parameters cannot be changed without restarting Prometheus

The configuration file is passed using the flag --config.file

Prometheus Configuration

CLOUD TRAIN
ACCELERATE YOUR GROWTH

The default configuration looks like this:

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          - localhost:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - "/etc/prometheus/alert.rules"
  # - "second_rules.yml"
```

Prometheus Configuration

CLOUD TRAIN
ACCELERATE YOUR GROWTH

To scrape metrics, you need to add configuration to the Prometheus config file

For example, to scrape metrics from prometheus itself, the following code block is added by default

```
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
```

Monitoring with Exporters

Exporters

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Build for exporting prometheus metrics from existing third-party metrics

When Prometheus is not able to pull metrics directly(Linux sys stats, haproxy, …)

- Examples:
  - MySQL server exporter
  - Memcached exporter
  - Consul exporter
  - Node/system metrics exporter
  - MongoDB Redis Many more….

Refer https://prometheus.io/docs/instrumenting/exporters/ for details

Monitoring Nodes

CLOUD TRAIN
ACCELERATE YOUR GROWTH

To monitor nodes, you need to install the node-exporter

The node exporter will expose machine metrics of Linux / *Nix machines

- For example: cpu usage, memory usage

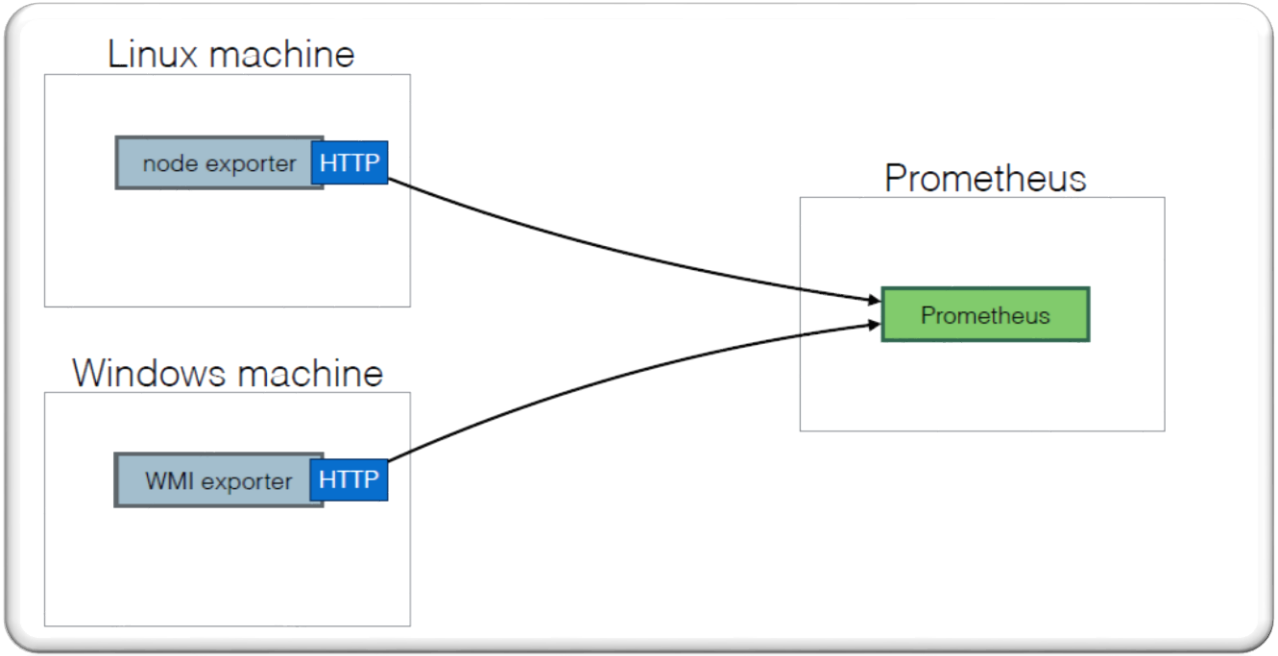
The node exporter can be used to monitor machines, and later on, you can **create alerts based on these ingested metrics**

For Windows, there's a WMI exporter (see https://github.com/martinlindhe/ wmi_exporter)

We are already using one in */etc/prometheus/prometheus.*

```
  - job_name: 'node_exporter'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9100']
```

Node Exporters



CLOUD TRAIN
ACCELERATE YOUR GROWTH

Prometheus Metrics Querying

Types of Metrics

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Counter

- A value that only goes up (e.g. Visits to a website)

Gauge

- Single numeric value that can go up and down (e.g. CPU load, temperature)

Histogram

- Samples observations (e.g. request durations or response sizes) and these observations get counted into buckets. Includes (_count and _sum)
- Main purpose is calculating quantiles

Summary

- Similar to a histogram, a summary samples observations (e.g. request durations or response sizes). A summary also provides a total count of observations and a sum of all observed values, it calculates configurable quantiles over a sliding time window.
- Example: You need 2 counters for calculating the latency
  - 1) total request(_count)
  - 2) the total latency of those requests (_sum)
- Take the rate() and divide = average latency

Querying Metrics

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Prometheus provides a functional expression language called PromQL

- Provides built in operators and functions
- Vector-based calculations 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)`

Querying Expressions

CLOUD TRAIN
ACCELERATE YOUR GROWTH

- **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`

Querying Operators

CLOUD TRAIN
ACCELERATE YOUR GROWTH

- **Arithmetic binary operators**
  Example: - (subtraction), * (multiplication), / (division), % (modulo), ^ (power/exponentiation)

- **Comparison binary operators**
  Example: == (equal), != (not-equal), > (greater-than), < (less-than) ,>= (greater-or-equal), <= (less-or-equal)

- **Logical/set binary operators**
  Example: and (intersection), or (union), unless (complement)

- **Aggregation operators**
  Example:**sum** (calculate sum over dimensions), **min** (select minimum over dimensions) ,**max** (select maximum over dimensions), **avg** (calculate the average over dimensions), **stddev** (calculate population standard deviation over dimensions), **stdvar** (calculate population standard variance over dimensions), **count** (count number of elements in the vector), **count_values** (count number of elements with the same value), **bottomk** (smallest k elements by sample value), **topk** (largest k elements by sample value), **quantile** (calculate ϕ-quantile ($0 \leq \phi \leq 1$) over dimensions)

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Alerting with Prometheus

Alerting

Alerting in Prometheus is separated into 2 parts

- **Alerting rules** in Prometheus server
- **Alertmanager**

Setting up alerts

Install Alertmanager

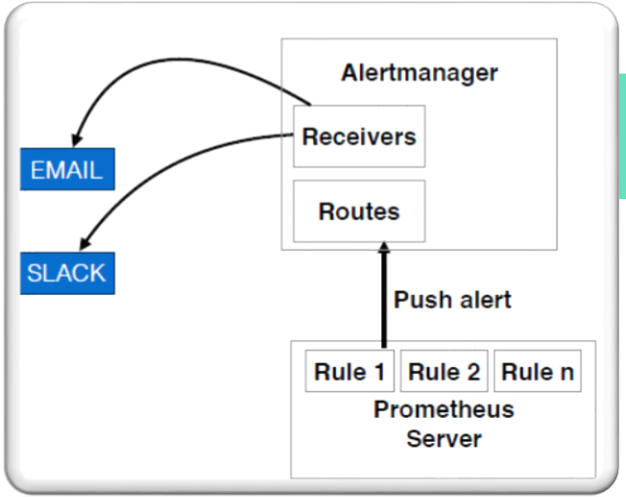
Create config for the Alertmanager

- Mail
- Slack

Alter prometheus config

Setup an alert

See the notification coming in when an alert is fired



Alerting Rules

CLOUD TRAIN
ACCELERATE YOUR GROWTH

- Rules live in Prometheus server config
- Best practice to separate the alerts from the prometheus config
  - Add an include to */etc/prometheus/prometheus.yml*

```
rule_files:
- "/etc/prometheus/alert.rules"
```

- Alert format:

```
ALERT <alert name>
  IF <expression>
  [ FOR <duration> ]
  [ LABELS <label set> ]
  [ ANNOTATIONS <label set> ]
```

- Alert example:

```
groups:
- name: example
  rules:
  - alert: cpuUsge
    expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{job='node_exporter',mode="idle"}[5m])) * 100) >
95
    for: 1m
    labels:
      severity: critical
    annotations:
      summary: Machine under healvy load
```

Alerting Rules

CLOUD TRAIN
ACCELERATE YOUR GROWTH

- Alerting rules allow you to define the alert conditions
- Alerting rules sent the alerts being fired to an external service
- The format of these alerts is in the Prometheus expression language
- Example:

```
groups:
- name: Important instance
  rules:

  # Alert for any instance that is unreachable for >5 minutes.
  - alert: InstanceDown
    expr: up == 0
    for: 5m
    labels:
      severity: page
    annotations:
      summary: "Instance {{ $labels.instance }} down"
      description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 minutes."
```

Alert Manager

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Alertmanager handles the alerts fired by the prometheus server

Handles **deduplication**, **grouping** and **routing** of alerts

Routes alerts to **receivers** (Pagerduty, Opsgenie, email, Slack,…)

Concepts:

- **Grouping**: Groups similar alerts into 1 notification
- **Inhibition**: Silence other alerts if one specified alert is already fired
- **Silences**: A simple way to mute certain notifications

Alert states:

- Inactive - No rule is met
- Pending - Rule is met but can be suppressed due to validations
- Firing - Alert is sent to the configured channel(mail,Slack,…)

Runs on port **:9093**

Alert Manager

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Alertmanager Configuration (*/etc/alertmanager/alertmanager.yml*):

```
global:
  smtp_smarthost: 'localhost:25'
  smtp_from: 'alertmanager@prometheus.com'
  smtp_auth_username: ''
  smtp_auth_password: ''
  smtp_require_tls: false

templates:
- '/etc/alertmanager/template/*.tmpl'

route:
  repeat_interval: 1h
  receiver: operations-team

receivers:
- name: 'operations-team'
  email_configs:
  - to: 'vista.sunil@gmail.com'
  slack_configs:
  - api_url: https://hooks.slack.com/services/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
    channel: '#devops'
    send_resolved: true
```

Alert Manager

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Prometheus Configuration (*/etc/prometheus/prometheus.yml*):

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          - localhost:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - "/etc/prometheus/alert.rules"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
```

Client Code Libraries

Client Libraries

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Instrumenting your code

Libraries

- Official: Go, Java/Scala, Python, Ruby
- Unofficial: Bash, C++, Common Lisp, Elixir, Erlang, Haskell, Lua for Nginx, Lua for Tarantool, .NET / C#, Node.js, PHP, Rust

No client library available?

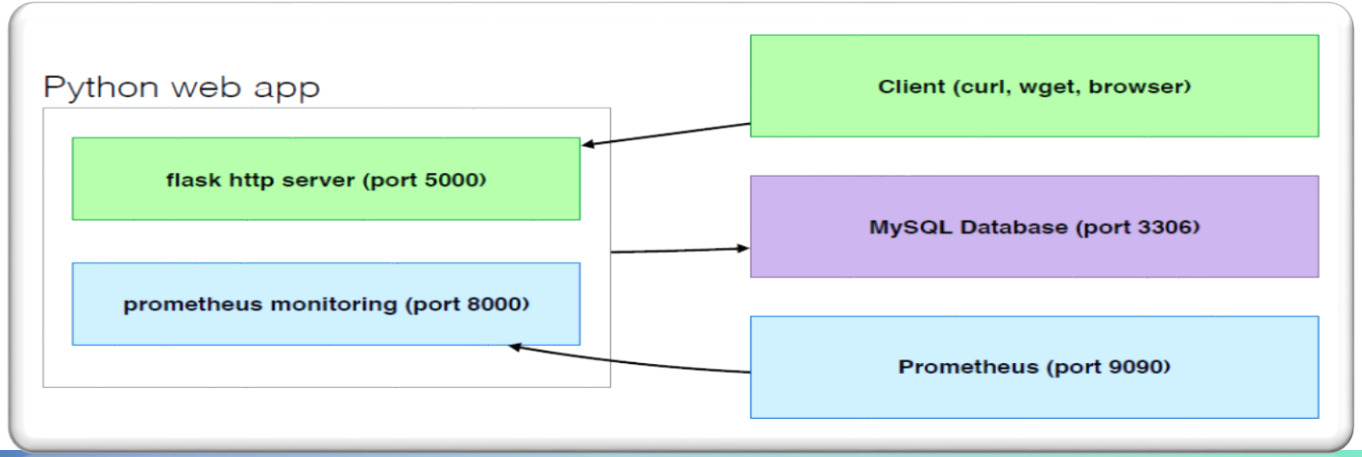
- Implement it yourself in one of the supported exposition formats

Monitoring a Python Web Application

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Let's integrate **prometheus monitoring** with a **web application** based on **python**

- **We**'ll use the official **prometheus_client** library for Python
- **Flask** is the web framework I'm going to use
  - It will create an **http server** and I'll able to configure routes (e.g. /query)
- We'll use **mysqlclient** for python to query a **MySQL** database
  - Will include one normal query and one **"bad behaving" query** that will take between 0 and 10 seconds to execute



CLOUD TRAIN
ACCELERATE YOUR GROWTH

Grafana Introduction

Introduction to Grafana

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Grafana is a multi-platform open-source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources.

Rather than using the UI, you can also **use yaml and json files** to provision Grafana with datasources and dashboards

This is a much more **powerful way** of using Grafana, as you can test new dashboards first on a **dev / test server**, then **import** the newly created dashboards to **production**

- You can do the import manually through the UI, or **using yaml and json files**
- When using files, you can keep files within **version control** to keep changes, revisions and backups

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Grafana
Installation & Configuration

Grafana Configuration

CLOUD TRAIN
ACCELERATE YOUR GROWTH

- The configuration of Grafana is all kept in /etc/grafana:

```
/etc/grafana/:
-rw-r----- 1 root grafana  14K Jul 17 12:30 grafana.ini
-rw-r----- 1 root grafana 3.4K Jul 17 12:30 ldap.toml
drwxr-xr-x 4 root grafana 4.0K Jul 17 13:15 provisioning/
/etc/grafana/provisioning/:
drwxr-xr-x 2 root grafana 4.0K Jul 17 14:56 dashboards/
drwxr-xr-x 2 root grafana 4.0K Jul 17 15:34 datasources/
```

- The data is kept in /var/lib/grafana:

```
/var/lib/grafana:
drwxr-xr-x 2 root    root    4.0K Jul 17 15:47 dashboards/
-rw-r----- 1 grafana grafana 500K Jul 17 15:48 grafana.db
drwxr-x--- 2 grafana grafana 4.0K Jul 17 12:31 plugins/
drwx------ 5 grafana grafana 4.0K Jul 17 12:40 sessions/
```

Grafana Configuration

CLOUD TRAIN
ACCELERATE YOUR GROWTH

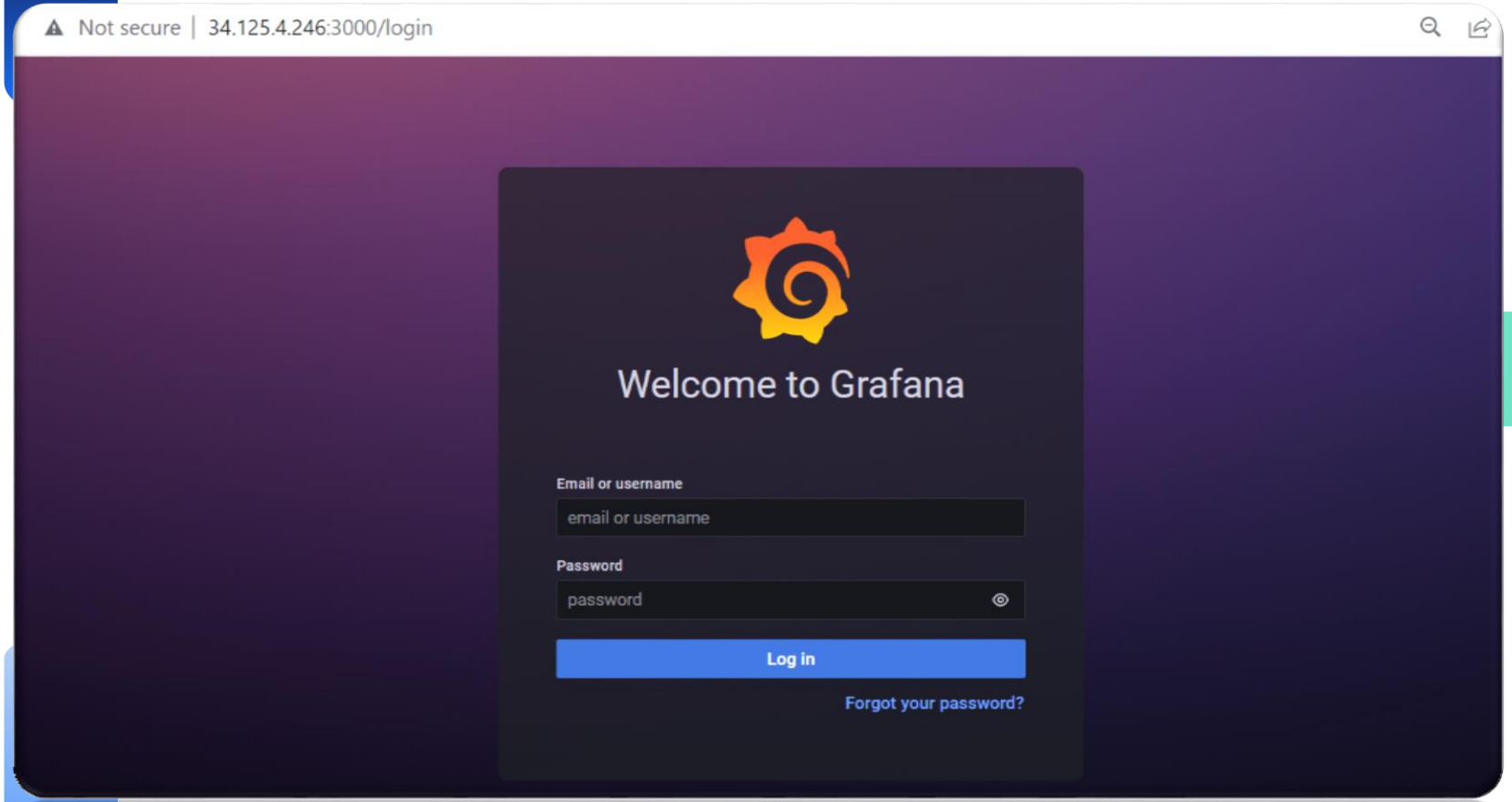
- You can change the database & paths in /etc/grafana/grafana.ini

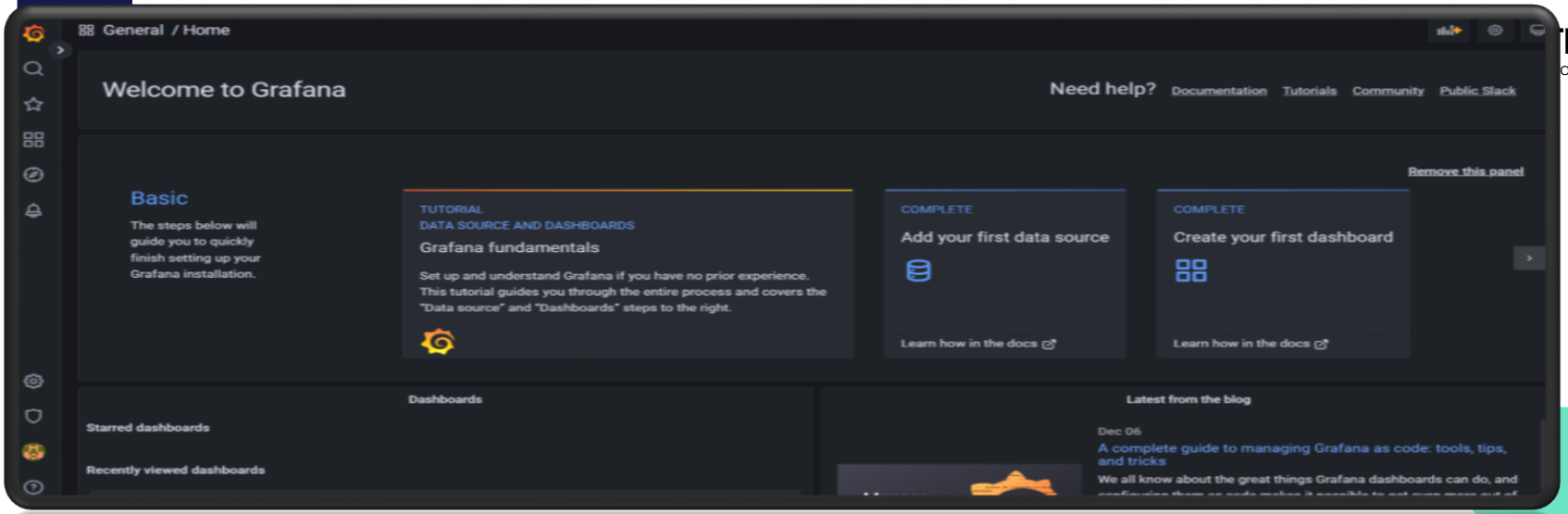
```
[paths]
# Path to where grafana can store temp files, sessions, and the sqlite3 db (if that is used)
;data = /var/lib/grafana

# Directory where grafana can store logs
;logs = /var/log/grafana

# Directory where grafana will automatically scan and look for plugins
;plugins = /var/lib/grafana/plugins

# folder that contains provisioning config files that grafana will apply on startup and while running.
;provisioning = conf/provisioning
...
[database]
# Either "mysql", "postgres" or "sqlite3", it's your choice
;type = sqlite3
;host = 127.0.0.1:3306
;name = grafana
;user = root
# If the password contains # or ; you have to wrap it with triple quotes. Ex """#password;"""
;password =
```





CLOUD TRAIN
ACCELERATE YOUR GROWTH

Got queries or need more info?

Contact us

TO ACCELERATE YOUR CAREER GROWTH

For questions and more details:

please call @ +91 98712 72900 or

visit https://www.thecloudtrain.com/ or

email at join@thecloudtrain.com or

WhatsApp us >>