LAB1

1-what is different http status code and explain meaning of each of them?

- **100s: Informational codes**: the server acknowledges the request initiated by the browser and that it is being processed (100–199).
- **200s: Success codes**: request received, understood, processed and expected info relayed to browser (200–299).
- **300s: Redirection codes**: a different destination has been substituted for the requested resource; further action by the browser may be required (300–399).
- **400s: Client error codes**: website or page not reached; page unavailable or there was a technical problem with the request (400–499).
- **500s: Server error codes**: request was accepted, but due to an error the server could not fulfil the request (500–599).

2-What database is used by Prometheus?

- Prometheus includes a local on-disk **time series database**, but also optionally integrates with remote storage systems.

3-what is the difference between different metrics types (counter, gauge, histogram)

- **Counter**

  A counter is a cumulative metric that represents a single monotonically increasing counter whose value can only increase or be reset to zero on restart. For example, you can use a counter to represent the number of requests served, tasks completed, or errors.

  Do not use a counter to expose a value that can decrease. For example, do not use a counter for the number of currently running processes; instead use a gauge.

  Client library usage documentation for counters:

  - Go
  - Java
  - Python
  - Ruby
  - .Net

- **Gauge**
  A gauge is a metric that represents a single numerical value that can arbitrarily go up and down.

  Gauges are typically used for measured values like temperatures or current memory usage, but also "counts" that can go up and down, like the number of concurrent requests.

  Client library usage documentation for gauges:

  - Go
  - Java
  - Python
  - Ruby
  - .Net

- **Histogram**
  A histogram samples observations (usually things like request durations or response sizes) and counts them in configurable buckets. It also provides a sum of all observed values.
  A histogram with a base metric name of <basename> exposes multiple time series during a scrape:
  - cumulative counters for the observation buckets, exposed as <basename>_bucket{le="<upper inclusive bound>"}
  - the total sum of all observed values, exposed as <basename>_sum
  - the count of events that have been observed, exposed as <basename>_count (identical to <basename>_bucket{le="+Inf"} above)

  Use the histogram_quantile() function to calculate quantiles from histograms or even aggregations of histograms. A histogram is also suitable to calculate an Apdex score. When operating on buckets, remember that the histogram is cumulative.

  Client library usage documentation for histograms:

  - Go
  - Java
  - Python
  - Ruby
  - .Net

- **Summary**
  Similar to a histogram, a summary samples observations (usually things like request durations and response sizes). While it also provides a total count of observations and a

sum of all observed values, it calculates configurable quantiles over a sliding time window.

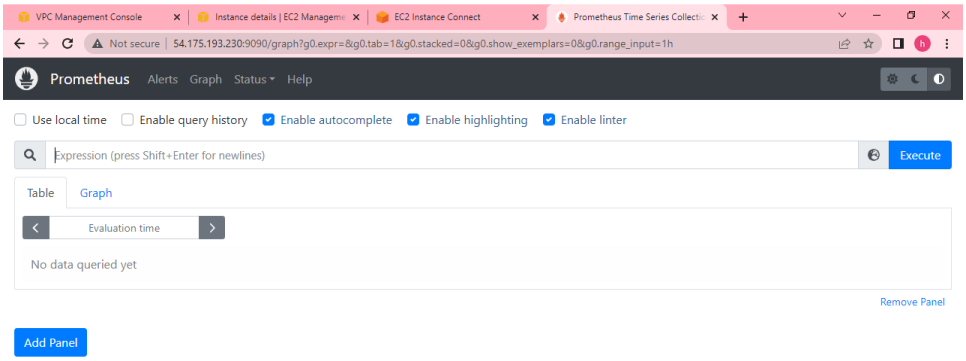
A summary with a base metric name of <basename> exposes multiple time series during a scrape:

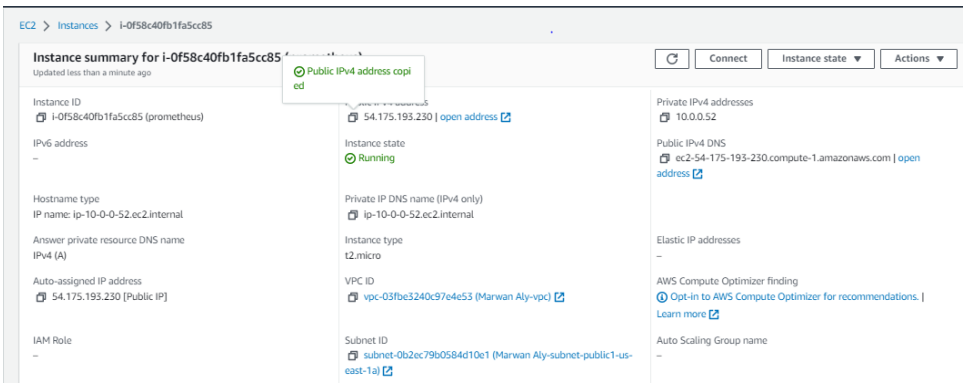
- streaming ϕ-quantiles (0 ≤ ϕ ≤ 1) of observed events, exposed as <basename>{quantile="<ϕ>"}
- the total sum of all observed values, exposed as <basename>_sum
- the count of events that have been observed, exposed as <basename>_count

Client library usage documentation for summaries:

- Go
- Java
- Python
- Ruby
- .Net

4-install Prometheus on your localhost or on server in any cloud provider





```
● prometheus.service - Prometheus
     Loaded: loaded (/etc/systemd/system/prometheus.service; disabled; vendor preset: enabled)
     Active: active (running) since Sat 2023-02-25 15:34:43 UTC; 8s ago
   Main PID: 7338 (prometheus)
      Tasks: 6 (limit: 1143)
     Memory: 15.6M
        CPU: 57ms
     CGroup: /system.slice/prometheus.service
             └─7338 /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --stora
```

5-add any new target to Prometheus.yaml file and apply any query on it using promql language