

Lab (1)

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

HTTP status codes are three-digit codes that are used to indicate the status of an HTTP request. They are divided into five classes:

1xx - Informational responses. These codes indicate that the request was received and is being processed.

2xx - Successful responses. These codes indicate that the request was successfully processed.

3xx - Redirection messages. These codes indicate that the client needs to take further action in order to complete the request.

4xx - Client error responses. These codes indicate that the client made a mistake in its request.

5xx - Server error responses. These codes indicate that the server encountered an error while processing the request.

Here are some of the most common HTTP status codes:

100 Continue - The client should continue with its request.

200 OK - The request was successfully processed.

301 Moved Permanently - The requested resource has been moved permanently to a new location.

400 Bad Request - The client made a mistake in its request.

401 Unauthorized - The client is not authorized to access the requested resource.

403 Forbidden - The client is forbidden from accessing the requested resource.

404 Not Found - The requested resource could not be found.

500 Internal Server Error - The server encountered an unexpected error while processing the request.

503 Service Unavailable - The server is temporarily unavailable.

The meaning of each HTTP status code is defined in the Hypertext Transfer Protocol:
<https://tools.ietf.org/html/rfc7231> specification.

Here are some additional HTTP status codes that you may encounter:

201 Created - The requested resource has been created successfully.

202 Accepted - The request has been accepted for processing, but the processing has not been completed yet.

203 Non-Authoritative Information - The requested resource is available, but the information in the response may not be authoritative.

204 No Content - The request was successful, but the response does not contain any content.

205 Reset Content - The client should reset its view of the resource.

206 Partial Content - The client only needs to download a portion of the requested resource.

302 Found - The requested resource has been moved temporarily to a new location.

303 See Other - The client should get the requested resource from a different URI.

304 Not Modified - The requested resource has not been modified since the client last requested it.

307 Temporary Redirect - The client should temporarily redirect its request to a different URI.

308 Permanent Redirect - The client should permanently redirect its request to a different URI.

402 Payment Required - The client needs to pay in order to access the requested resource.

405 Method Not Allowed - The requested method is not supported by the resource.

406 Not Acceptable - The requested resource is not available in the format that the client requested.

407 Proxy Authentication Required - The client needs to authenticate with a proxy server in order to access the requested resource.

408 Request Timeout - The client's request timed out.

410 Gone - The requested resource is no longer available.

411 Length Required - The request must specify the length of the content.

412 Precondition Failed - One of the preconditions that the client specified in its request failed.

413 Payload Too Large - The request payload is too large.

414 URI Too Long - The URI in the request is too long.

415 Unsupported Media Type - The requested content type is not supported by the resource.

418 I'm a teapot - This is a joke status code that is sometimes used to indicate that the server is not a teapot.

421 Misdirected Request - The request was directed to the wrong server.

422 Unprocessable Entity - The request was well-formed but could not be processed due to semantic errors.

423 Locked - The resource is locked and cannot be modified.

426 Upgrade Required - The client must upgrade its protocol in order to continue the request.

429 Too Many Requests - The client has made too many requests in

2-What database is used by Prometheus?

Prometheus uses a custom time series database called the Prometheus Time Series Database (TSDB). The TSDB is designed to be efficient for storing and querying large amounts of time series data. It uses a variety of techniques to optimize for performance, including:

- In-memory caching of frequently accessed data
- LevelDB for indexing
- Chunked storage of sample data
- A write-ahead log for durability

The TSDB is also designed to be scalable. It can be horizontally scaled by adding more Prometheus servers, and it can be vertically scaled by increasing the amount of memory and storage on each server

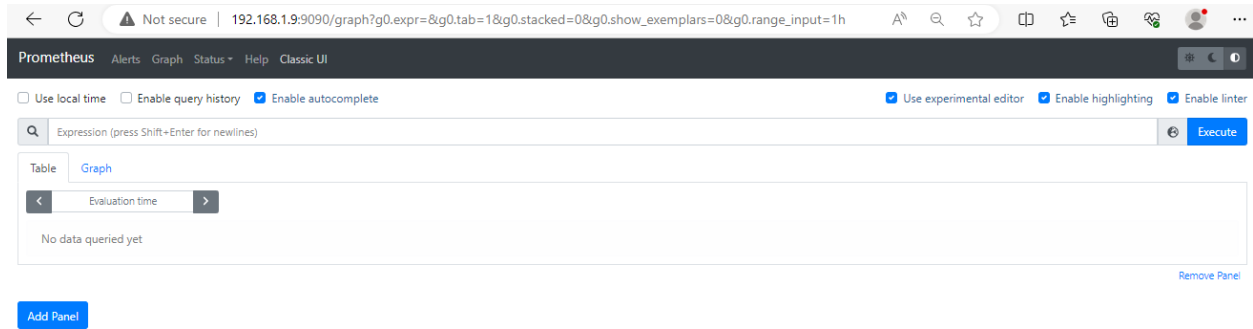
In addition to the TSDB, Prometheus can also integrate with other time series databases, such as InfluxDB and OpenTSDB. This allows Prometheus to scale to even larger datasets by offloading storage to a dedicated database.

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

- **Counter** metrics track the cumulative sum of events. For example, you could use a counter to track the number of requests that have been processed by a server. Counters can only increase, and they are typically reset to zero when the application restarts.
- **Gauge** metrics track the current value of a metric. For example, you could use a gauge to track the current CPU usage of a server. Gauges can increase or decrease, and they are not reset to zero when the application restarts.
- **Histogram** metrics track the distribution of values of a metric. For example, you could use a histogram to track the latency of requests that have been processed by a server. Histograms divide the values of the metric into buckets, and they track the number of values that fall into each bucket.

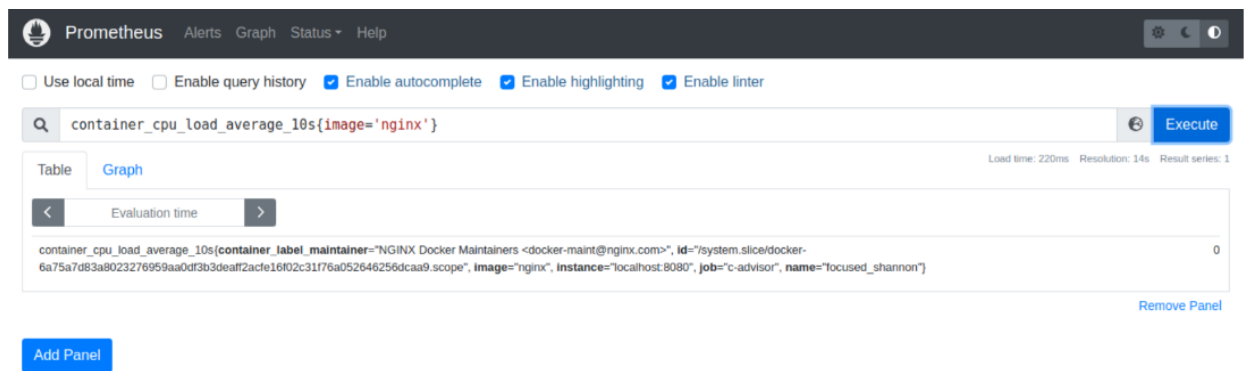
4-Install Prometheus on your localhost or on server in any cloud provider

Installed locally as per below



The screenshot shows the Prometheus web interface in a browser. The address bar shows the URL `192.168.1.9:9090/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.show_exemplars=0&g0.range_input=1h`. The interface includes a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below the navigation bar, there are checkboxes for 'Use local time', 'Enable query history', 'Enable autocomplete', 'Use experimental editor', 'Enable highlighting', and 'Enable linter'. The main query editor contains the text 'Expression (press Shift+Enter for newlines)'. Below the editor, there are tabs for 'Table' and 'Graph', and a 'Remove Panel' button. The 'Graph' tab is selected, and the 'Evaluation time' is set to 'No data queried yet'. An 'Add Panel' button is located at the bottom left.

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



The screenshot shows the Prometheus web interface with a query for container CPU load. The query editor contains the text `container_cpu_load_average_10s{image='nginx'}`. The 'Graph' tab is selected, and the 'Evaluation time' is set to '0'. The graph shows a single data series for the container `container_cpu_load_average_10s{container_label_maintainer="NGINX Docker Maintainers <docker-maint@nginx.com>", id="/system.slice/docker-6a75a7d83a8023276959aa0df3b3deaf2acfe16f02c31176a052646256dcaa9.scope", image="nginx", instance="localhost:8080", job="c-advisor", name="focused_shannon"}`. The 'Add Panel' button is located at the bottom left.