**PromQL**

* PromQL is the Prometheus Query Language
* Labels are key part of PromQL and you can use them not only do arbitrary aggregations but also to join different metrics together for arthimetic operations against them.

**Aggregation Basics**

* Gauge: These are snapshots of state and usually when you are aggregating them you want to take a sum, average, minimum or maximum.
  + Consider the metric node\_filesystem\_size\_bytes (Node Exporter) which reports the size of each of your mounted filesystems and has device, fstype and mountpoint labels
  + Consider this query

sum without(device, fstype, mountpoint)(node\_filesystem\_size\_bytes)

* This works as without tells the sum aggregator to sum everything up with the same labels and ignoring these three
* Consider this query

max without(device, fstype, mountpoint)(node\_filesystem\_size\_bytes)

* This would return the biggest mounted filesystem on each device.
* Consider the expression avg without(instance, job)(process\_open\_fds)
* Counter: Counter tracks the number or size of events and the value your applications expose on their metrics.
  + When we use counter we would usually want to know how counter is increasing/decreasing over time
  + This can be done by rate function
* rate(node\_network\_receive\_bytes\_total[5m])
  + The above expression/query calculates amount of network traffic received per second and [5m] provides the rate function with 5 minutes of data
  + The output of rate function is a gauge, so we can use aggregations
* sum without(device)(rate(node\_network\_receive\_bytes\_total[5m]))
* Summary: Summary metric usually contains both \_sum and \_count and sometimes a time series with no suffix with a quantile lablel. \_sum and \_count are both counters
  + Prometheus exposes http\_response\_size\_bytes summary and http\_response\_size\_bytes\_count tracks number of user requests
  + Consider the expression sum without(handler)(rate(http\_response\_size\_bytes\_count[5m]))
* Histogram: Histogram metrics allows you to track the distribution of the size of the events, which allows you to calculate quantiles
  + Prometheus exposes a histogram prometheus\_tsdb\_compaction\_duration\_seconds that tracks how many seconds compaction takes for time series database
  + histogram\_quantile function takes catre of calculating quantiles
* histogram\_quantile(0.9, rate(prometheus\_tsdb\_compaction\_duration\_seconds[1d]))
* Selectors: working with all the different time series with different label values for a metric can be overwhelming and confusing. Usually you will want to narrow down which time series you are working on
  + process\_resident\_memory\_bytes{job="node"}
  + `job="node" is called a matcher and we have many matcher
  + Matchers: There are four matchers
    - =: this is equality matcher
    - !=: this is negative equality matcher
    - =~: This is regular expression mathcher job=~"n.\*"
    - !~: This is negative regular expression matcher instance!~"prod\*"
* Durations:
  + ms: Milliseconds
  + s: seconds
  + m: minutes
  + h: hours
  + d: days
  + w: weeks
  + y: year
  + While using durations write duration as 1 unit
* 100m (valid)
* 1h40m (invalid)
* Offset: There is a modifier we can use called as offset, which allows you take evaluation time for a query on a per-selector basis
  + process\_resident\_memory\_bytes{job="node"} offset 1h this would get memory usage an hour before the query evaluation time.
  + `rate(process\_cpu\_seconds\_total{job="node"}[5m] offset 1h )
* by: In addition to without ther s also a by clause. Where without specifies the labels to remove by specifies labesls to keep. you cannot use both by and without in same aggregation
  + sum by(job, instane, device)(node\_filesystem\_size\_bytes)
  + count by(release)(node\_uname\_info)
* Operators:
  + sum
  + count
  + avg
  + stddev
  + stdvar
  + min
  + max
  + topk
  + bottomk
  + quantile
  + count\_values
* Arithmetic Operators:
  + / devision
  + % : modulation
  + ^: exponentiation
* Comparision Operators
  + == equals
  + != not equals
  + <
  + >
  + >=
  + <=