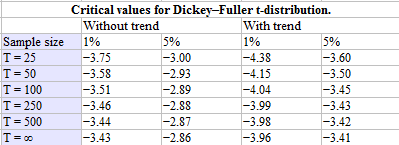
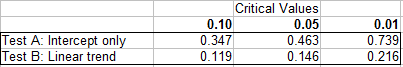
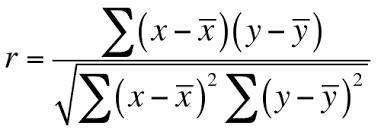
## <https://machinelearningmastery.com/how-to-grid-search-sarima-model-hyperparameters-for-time-series-forecasting-in-python/>

## Exploratory Data Analysis

* Use pandas describe to do initial descriptive + outliers
* Visualization
  + Histogram
    - Using histogram on diff() sometimes more interesting if there is a trend
    - If there is a trend, your raw histogram will spread
  + Scatter plot
    - To measure whether 2 variables are corelated
    - Introducing diff + lag is important with the idea of comparing delta (ti and ti-k)
  + Season plot, box plot
    - Grouped by certain characteristics
  + Heat map
    - Define 2 axes in time perspective, e.g. year and day
* Stationarity
  + Line plot
  + Mean and variance split
  + Histogram plot  
    If histogram is spread (not normal) mostly there is a trend
  + Statistical Test
    - Dickey Fuller Test / Augmented Dickey Fuller Test 🡪 test unit root present  
      **H0 : unit root is present (not stationary)**
      * *from statsmodels.tsa.stattools import adfuller*

1st result output is the test statistic, more negative, the better to reject hypothesis  
2nd result output is the p-value, it has to be generally less than 5% value  


* + - Additional: KPSS (Kwiatkowski-Phillips-Schmidt-Shin) 🡪 test stationarity present. Thus, we need p-values >= 0.05  
      **H0 : process is stationary (unit root is not present)**  
      
  + Transformation
    - Diff() couple of times < 3
    - Log or sqrt transform
    - Box Cox transformation  
      *scipy.stats.boxcox*  
      Transforming non-normal to normal distribution
  + Window function
    - Rolling Window (Mean)
    - Expanding Window
* Autocorrelation Function
  + Pearson correlation  
    
  + ACF 🡪 with raw data time lags  
    ACF is symmetric, only positive lags are considered
  + PACF 🡪 with residual time lags  
    Critical region is +- 1.96 \* sqrt(n)  
    Correlation of the residual from linear modelling of Ti | Ti-1 and Ti-2 | Ti-1  
    Ti-2 can be expanded to Ti-k+1
  + If you have a trend, ACF might not be informative as the value above confidence band will be many. However, the PACF might tell us better.
* Spurious Correlation
  + Cointegration, real relationship between 2 time series

## Decomposition

* Classical:
  + Seasonal components assumed to be constant
  + Type:
  + Additive

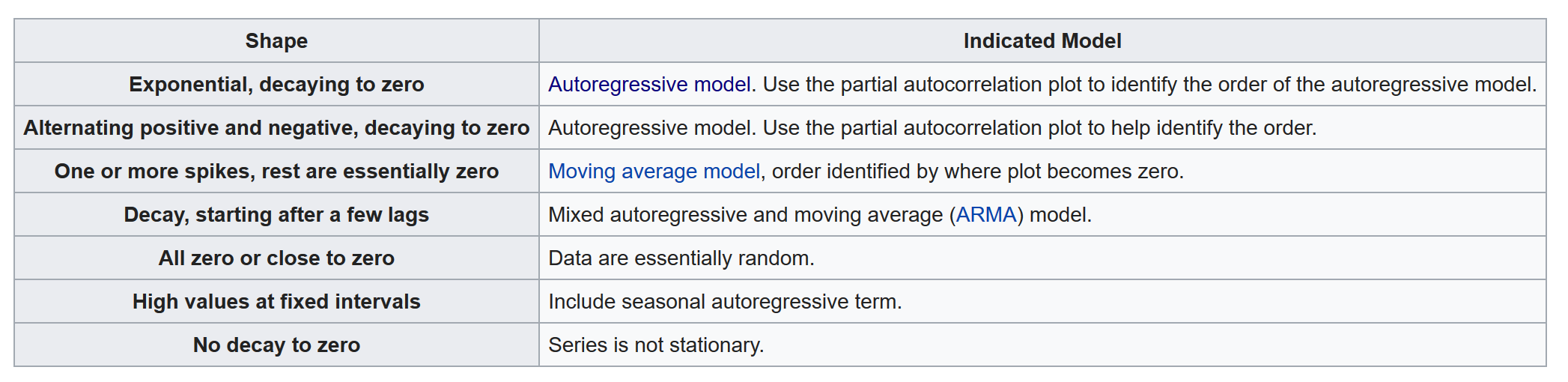
y(t) = Level + Trend + Seasonality + Noise

* + Multiplicative

y(t) = Level \* Trend \* Seasonality \* Noise

* + Python
    - statsmodels.tsa.seasonal import seasonal\_decompose
* X11 decomposition
* SEATS decomposition
* STL decomposition

## Statistical Modelling

* ARIMA and friends
  + 
  + Ljung-box method
    - Checking if residuals are iid. Independent identically distributed
      * Basically, checking if the residuals are white noise, i.e. checking mean = 0 and variance constant
      * H0: residuals are rigged
      * H1: residuals are white noise
      * Need p-values to be high to reject this, > 0.05 (chi square test)
    - Many statisticians challenge the concept of ljungbox method
  + Breusch Godfrey method
  + Jarque Bera method
    - Checking if residuals is normal
    - Pvalue < 0.1: then potential not normal
    - Pvalue >= 0.1: residuals are normal
* Conditional Heteroskedasticity
  + Variance is not constant; thus, this is to be used to identify if residuals are white noise
  + To be exact, there is some part of the time series “regime” that has different variance
  + Regime detection, to avoid conditional volatility
  + Might look like white noise but it is not!

## Comparison between metrics for GridSearch

<https://medium.com/@rrfd/sarima-modelling-for-car-sharing-basic-data-pipelines-applications-with-python-pt-1-75de4677c0cd>

# System Design

## URL Shortening

* Assumptions on data:
  + Long URL: 2KB, 2048 Characters
  + Short URL: 20 Bytes, 20 Characters
  + Created: 8 Bytes, 8 Characters
  + Expired: 8 Bytes, 8 Characters
  + Total: 2KB per record
  + 10 million records, could be related to users OR number of URLs from our websites
  + 2,100 \* 10,000,000 = 0.02 TB / month = 0.24 TB / year
* Algorithms:
  + MD5
    - Too long, can take the first n characters
  + B62
    - Using base62 (all alphanumeric chars) as representation
* Database:
  + RDBMS
    - ACID
    - But scalability
  + NOSQL
    - Distributed
    - But eventual consistency
* Technique:
  + Generate random number and encode using base62

Need to create hierarchy of indexes to quicken search

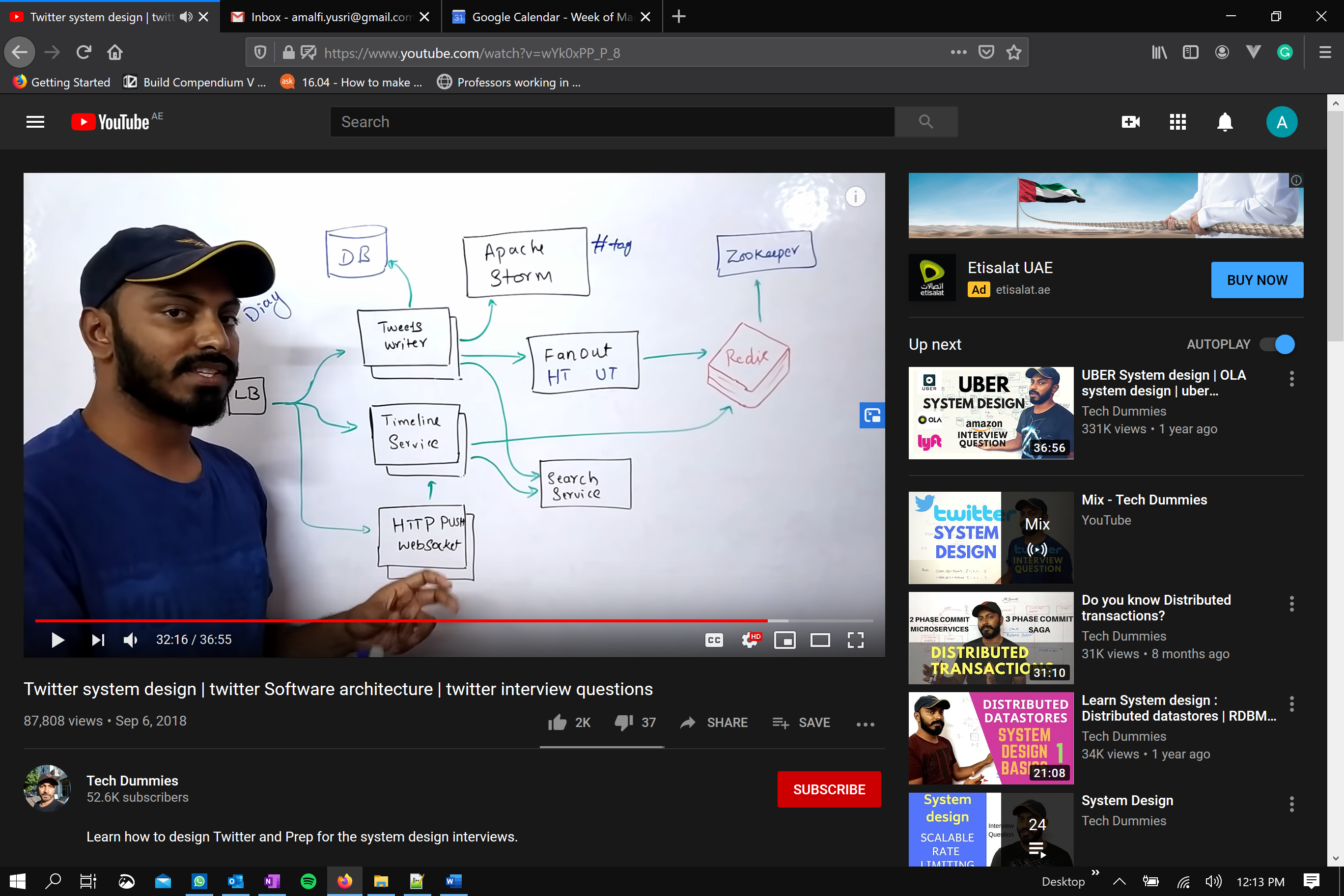
* + MD5 and add random number before hashing
  + Use of counter and convert to base62  
    Zookeeper for counting before converting to base62

## Streaming

* Multitier load balancing scheme, per zone then per services
* Transcoding to accommodate multiple devices, resolution and bitrate (network speed)  
  Break the “movie” into several chunks for transcoding and merge (store on AWS)
* Streaming server and application server are different
* Streaming server application will keep tracking for the best server for streaming

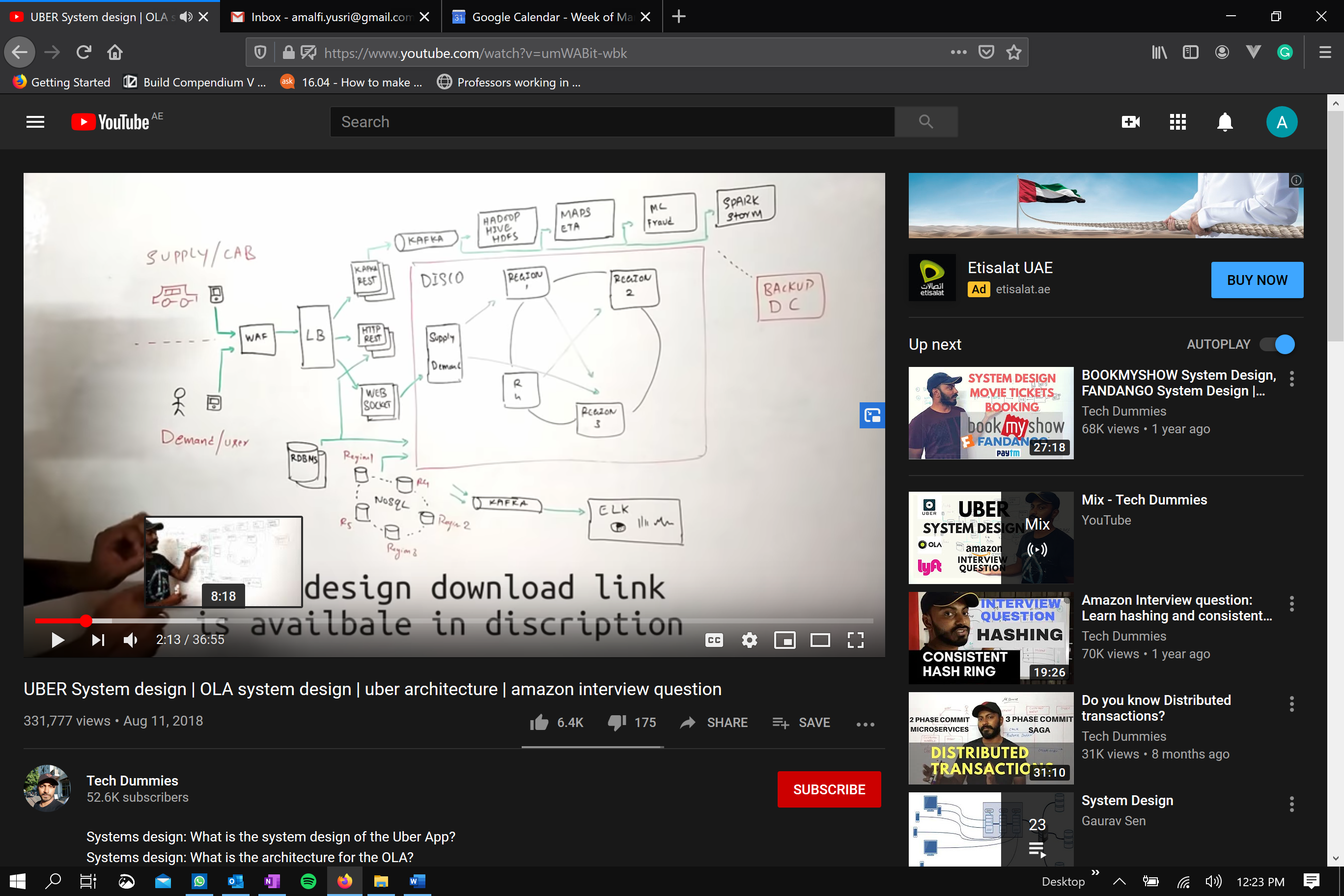
## Twitter

* Concept of data:
  + User
  + Tweet
  + Follower
* User timeline
  + Need to utilize cache for faster “recent” tweets
    - Redis use dictionary like access
* Home timeline (from different users)
  + Get following
  + Get their recent tweets
  + Merge into timeline
  + Use fanout approach:
    - Push to the follower “timeline” instead of them pulling
    - Don’t wait for pull, but push when there is a new tweet
    - However, if the user is super popular, pull mechanism is better
      * Creating new cache for celebrity cache, trigger will be eventual frequency
  + Operators (decoupling) scaling utilizing apache kafka ended up in redis
  + Inverted full text index
    - Keyed on words, rank the tweet based on hashtags / words
    - Access via scatter and gather



## Uber/Grap

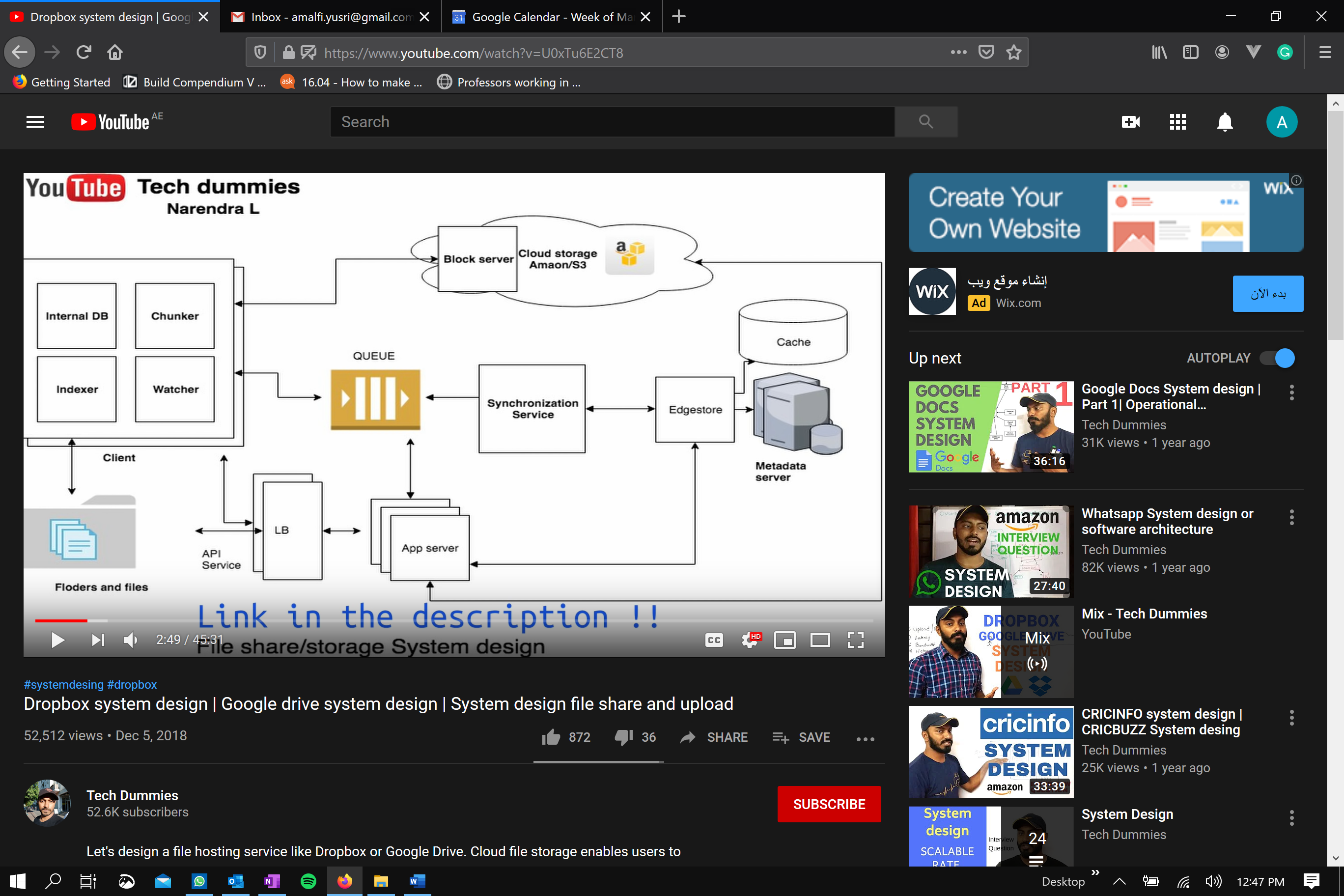
* Microservices are used (service oriented architecture)



* How dispatch optimization works
  + Creating cells in a map, with unique ID to distribute the problem
  + Keep updating the supply / cab location
  + WAF to block unwanted IP or addresses
  + LB to load balance to the components

## Dropbox / Google Drive

* Features
  + Upload and download
  + Sync folder
  + History of updates



* Problem:
  + Upload / concurrency
  + Latency
  + Bandwidth
  + History
* Divide the file into chunks
  + Upload only the updated ones
  + Hash the chunk to store in the metadata (which also be saved in the cloud)

# Zalando Corporate

* Founded October 2008
* Platform connecting customers, brands and partners
* 17 countries, 15 for partners
* 31 million monthly active customers
* 340 million monthly traffic
* Founder: Robert and David
* Innovation
  + Retail to technology company
  + Research lab
  + Hack week
  + Sustainability, worked as data scientist for energy consultancy
* Platform:
  + Frontend
  + Fulfillment services
  + Marketing services
  + Smart logistics
  + Gax system for retailers (manage and track orders)
* Tech
  + Redis
  + Akka
  + Apache flink
  + Kubernetes and docker
  + Open sources FLAIR? NLP: NER and POS, understand fashion concepts
* Why Zalando
  + Technology company
  + Before was retail focused, tells me about innovation is driver
  + Data availability, 31 mio active customers, 340 mio traffic
  + Size of the company, while startup minded
  + Tech perks, hack week, etc.
  + Potentially continuing education in Europe
  + Personal reasons:
    - Family education etc.
* Questions:
  + How often employee interact with CEO / C-exec?
  + Career ladder, clear definition between level
  + Self-development, rotating engineers?
  + Tradebyte and ZMS?
  + Position? Lead / Junior?