# Architectural Design

## System Architecture

* High-level physical systems involved:
  + Client’s machine
  + Database server
  + Any Third Party Service(s)
* Connections between systems:
  + HTTP/HTTPS connection from the client to the web server
  + PostgreSQL connection between the web server and the database server
  + Any connection between the web server and Third Party Service(s)
* Identify dependencies:
  + Augur Server, which we can access with API calls

## Data Architecture

* High-level (Entity) descriptions of data:

Repo{

repo\_id,

#PR(week, month, year, ytd),

#Issues(week, month, year, ytd),

average issue closing time,

average comments per issue,

average merge probability per PR,

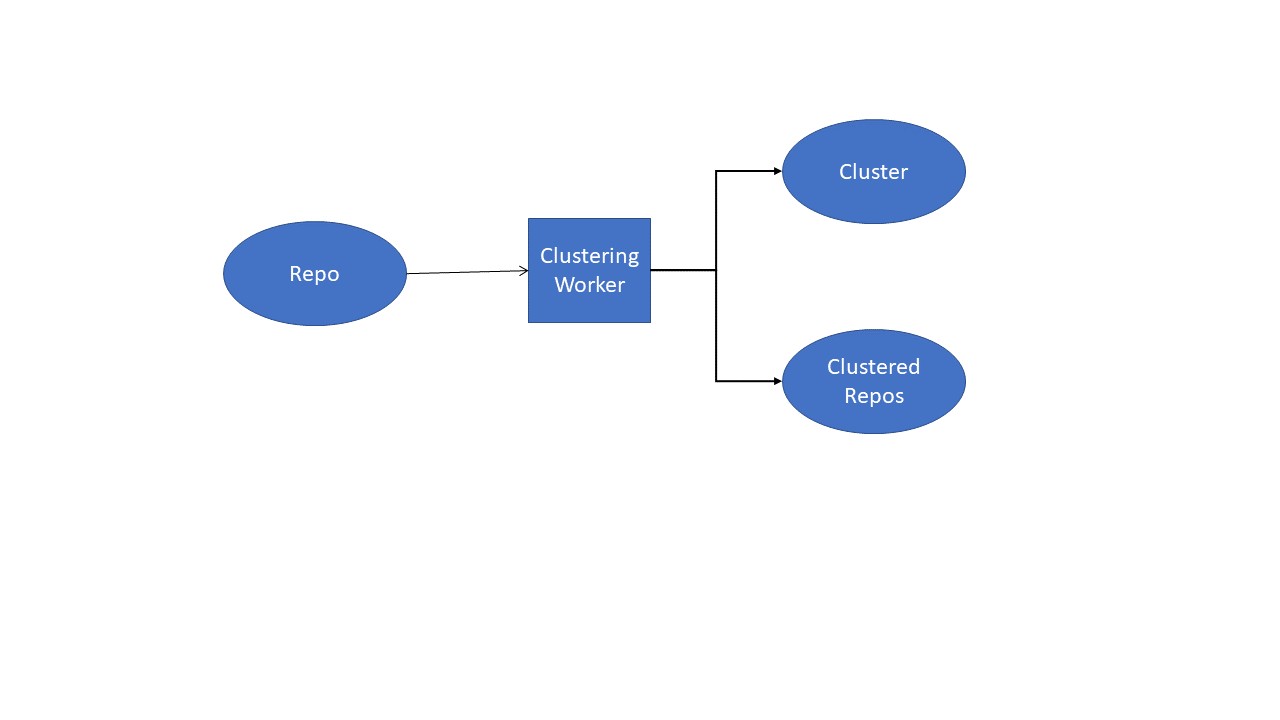
new contributor probability

}

Cluster{Cluster\_id, #repos, centroid}

Clustered repos {repo\_id, Cluster\_id}

* Show the data structure (conceptual) at the entity level Illustrate critical data flows:



## Process Architecture

* Integrate business or user process practices into the architecture
  + The user should be able to submit a set number of repos to cluster
  + The user should be able to change the metrics that the clusters are displayed as, though the data will be clustered on all metrics
  + The user should be able to submit a query to the server
* Show the touchpoints between process and technology
  + There should be a list of repos the user can select from
    - default to the top *n* repositories on Augur
  + There should be drop-down boxes containing the metrics that we display on the front-end
  + There should be a submit button on the front end to submit a query

## Software Architecture

## Software components:

* + Augur
  + IDE/Text Editor
* Connections between components:
  + SSH Protocol
  + TCP/IP with SSL
* Key Software Subsystems:
  + APIs
* Some of these components may also show up in the system architecture

# Detailed Design

1. Software components
   * Augur
     + This is the software the whole project is based around
   * IDE/Text Editor
     + This is what we will use to make changes to augur and test them.
2. Connections between them
   * SSH Connection
   * HTTP/HTTPS Connection
   * TCP/IP with SSL
3. Describe behavior down to methods
   * Cluster( Repos[] ) ← Back-end
     + A method that will cluster the *N* input repositories into *K* different clusters. We will likely use the K-Means clustering algorithm for this but will decide this in the future. If less than *N* repositories, return error code 404 and a message describing the error. Return a 2D array of metrics for each Repository.
   * parse\_data( Data[], Metrics[] ) ← Front-end
     + A method that will parse the data passed in for the entries in Metrics. If either array is NIL, return an error.
   * render( Parsed\_Data[] ) ← Front-end
     + A method that will take an input of parsed data and render a plot in a specified HTML element.
   * on\_submit() ← Front-end, attached to Button element
     + A method that will call the Cluster() method, getting the input Repositories from the web page and sending them to the Cluster() method. Ensure that there are *N* repositories being sent. If an error code is returned, relay the error message to the user and don’t render the plot. If the correct data is returned, get the metrics input by the user and send both the data and metrics to parse\_data(). Assuming no error is returned from parse\_data(), render the data with render(parsed\_data[]).

Notes from meeting:

* Can either use existing worker or make a new one for ML
  + Sounds like we are leaning towards creating a new one
* Current Clustering worker:
  + Has a pre-set number of repos as input
  + Clusters are generated with computational linguistics
  + Semisupervised topic modeling
* Work on individual branches, then merge them into augur-new branch