Job Search documentation

Implementing commute search in your UI

You can integrate commute search into your UI to allow job seekers to search for jobs within a geographic area set by commute time. Commute search estimates commute time based on a user's selected transit mode and the time of day they plan to travel.

Implement commute search

1. Before you can implement commute search, Cloud Talent Solution must be hooked up to your UI. Follow the quickstart (https://cloud.google.com/talent-solution/job-search/docs/before-you-begin) guides to set up Cloud Talent Solution.
2. Commute search uses the address data that you uploaded with your jobs during CTS implementation to calculate commute time. To enable this feature in your existing CTS UI, send a `jobs.search` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs/search) request and include a `CommuteFilter` object in the `JobQuery.commuteFilter` field. `commuteMethod`, `travelDuration`, `startCoordinates`, and either `roadTraffic` or `departureTime` are required fields.

GO JAVA PHP PYTHON RUBY

JDPLATFORM/GOLANG-SAMPLES/BLOB/MASTER/JOBS/V4/HOWTO/COMMUTE_SEARCH_SAMPLE.GO)

FEEDBACK (#)

```
import (
        "context"
        "fmt"
        "io"

        talent "cloud.google.com/go/talent/apiv4beta1"
        "github.com/golang/protobuf/ptypes/duration"
        "google.golang.org/api/iterator"
        talentpb "google.golang.org/genproto/googleapis/cloud/talent/v4beta1"
        "google.golang.org/genproto/googleapis/type/latlng"
)

// commuteSearch searches for jobs within commute filter.
func commuteSearch(w io.Writer, projectID, companyID string) error {
        ctx := context.Background()

        // Initialize a jobService client.
        c, err := talent.NewJobClient(ctx)
        if err != nil {
                return fmt.Errorf("talent.NewJobClient: %v", err)
        }

        // Construct a jobQuery request with a commute filter.
        jobQuery := &talentpb.JobQuery{
                CommuteFilter: &talentpb.CommuteFilter{
                        CommuteMethod:  talentpb.CommuteMethod_TRANSIT,
                        TravelDuration: &duration.Duration{Seconds: 1800},
                        StartCoordinates: &latlng.LatLng{
                                Latitude:  37.422408,
                                Longitude: -122.085609,
                        },
                },
        }
        if companyID != "" {
                jobQuery.Companies = []string{fmt.Sprintf("projects/%s/companies/%
        }

        // Construct a searchJobs request with a jobQuery.
        req := &talentpb.SearchJobsRequest{
                Parent: fmt.Sprintf("projects/%s", projectID),
                // Make sure to set the RequestMetadata the same as the associated
                // search request.
                RequestMetadata: &talentpb.RequestMetadata{
                        // Make sure to hash your userID.
                        UserId: "HashedUsrID",
```

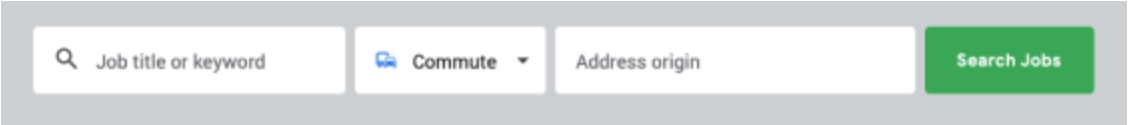
```
                        // Make sure to hash the sessionID.
                        SessionId: "HashedSessionID",
                        // Domain of the website where the search is conducted.
                        Domain: "www.googlesample.com",
                },
                // Set the actual search term as defined in the jobQuery.
                JobQuery: jobQuery,
        }

        it := c.SearchJobs(ctx, req)

        for {
                resp, err := it.Next()
                if err == iterator.Done {
                        return nil
                }
                if err != nil {
                        return fmt.Errorf("it.Next: %v", err)
                }
                fmt.Fprintf(w, "Matcing job: %q\n", resp.GetJob().GetName())
                fmt.Fprintf(w, "Job address: %v\n", resp.GetCommuteInfo().GetJobLo
        }
}
```

UI recommendations

1. Cloud Talent Solution doesn't allow searching by both distance (using the CTS location filter) commute time in the same API call. To allow job seekers to access both options, use a 2-tab approach or similar.
2. Modify the frontend of your application to ensure that the backend automatically populates a job seeker's relevant information into the commute filter (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/JobQuery#FIELDS.commute_filter). The backend should call the API as it would in a regular search request.
3. Include items in your UI:
    - An option to select either a distance search or commute search. For example, your Search UI could look like the sample below:



- A drop-down menu of commute method options.
- An option to adjust traffic conditions.
- The total travel time (the maximum supported travel time is 60 minutes).
- Commute start time.

4. The commute time information returned from the API is used to display information to the job seeker. Only relevant jobs located within the designated commute time area are returned in the results list. See the Job Search Best Practices (https://cloud.google.com/talent-solution/job-search/docs/best-practices) documentation for a discussion of ways to adjust the order and number of jobs returned within this area.
5. Commute search results are based on historical and aggregated data rather than live traffic conditions. The `departureTime` traffic conditions are calculated from average traffic conditions at the specified time of day. The `BUSY_HOUR`/`TRAFFIC_FREE` options under `roadTraffic` are average traffic conditions at morning rush hour and midnight, respectively. Users receive the same commute search results no matter what time of day they send a query.

Generating a map with commute information (Recommended)

You can leverage Google Maps to generate a map based on the commute time information returned from CTS and embed it into the results returned to a job seeker. The Maps API suite has several options for displaying a map. Some Maps API options are more effective than others. For example, the Google Maps JavaScript Heatmap visualization (https://developers.google.com/maps/documentation/javascript/earthquakes#heatmaps) paired with marker clustering (https://developers.google.com/maps/documentation/javascript/marker-clustering) is an effective way to visualize the relevant jobs returned to a job seeker inside the area determined by their set commute preferences. Conversely, Directions Mode (https://developers.google.com/maps/documentation/javascript/examples/directions-travel-modes) does not show all jobs returned in a search request and is not a recommended option.

Note: Maps APIs has a number of different pricing structures. These prices are NOT covered under the usage of Cloud Talent Solution. Only the use of Cloud Talent Solution APIs to request the jobs within the specified

commute times are covered as a part of Cloud Talent Solution pricing.

For more information on implementing a commute-based search, see the Commute Search how-to guide (https://cloud.google.com/talent-solution/job-search/docs/search-commute).

Last updated December 13, 2019.