

# CORD CUTTING CALCULATOR

---

DIRECTED STUDIES REPORT

*Advisor: Duncan M (Hank) Walker*

*Student: Dileep Kumar Gunda*

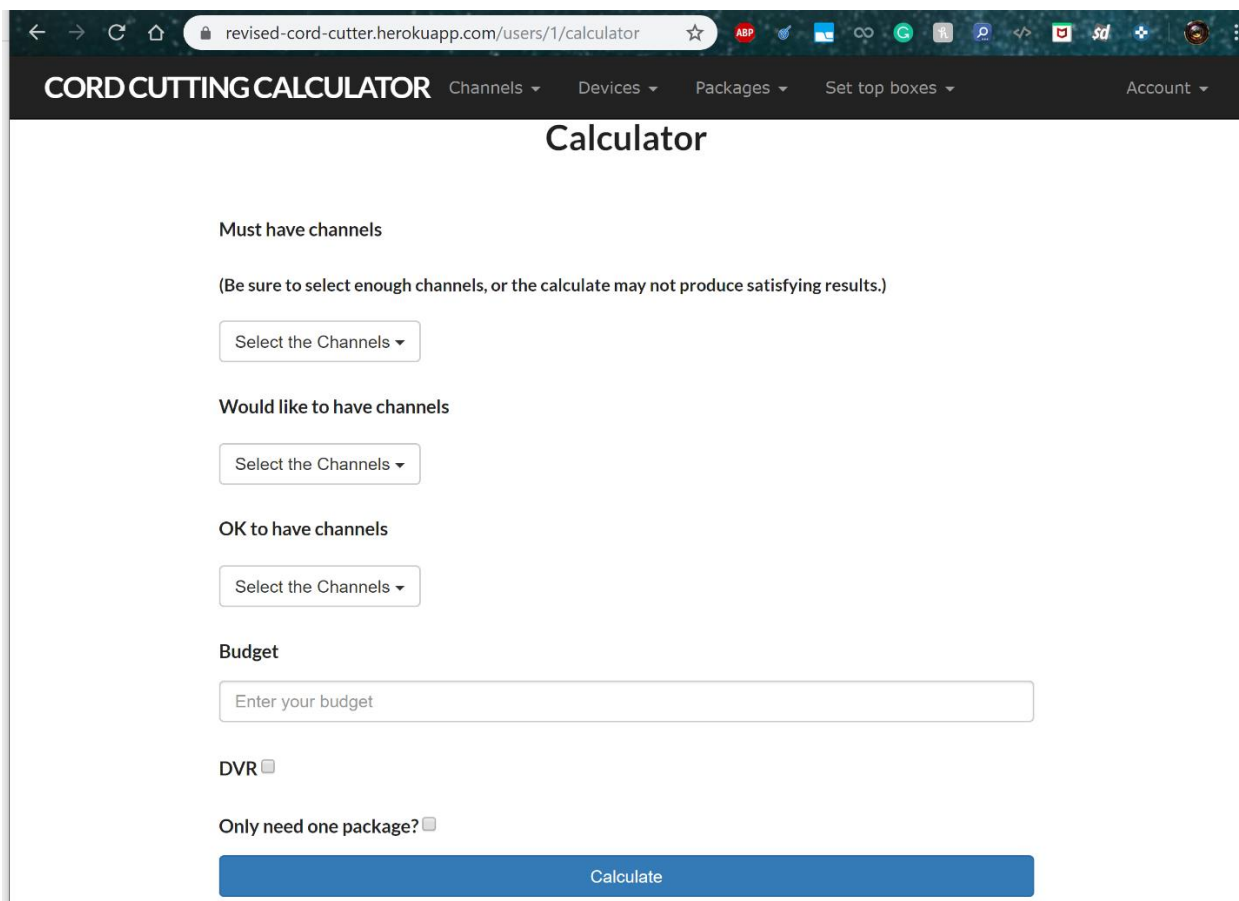
*Date: December 9, 2019*

## INTRODUCTION

Now-a-days there exists many streaming services which provide streaming to live tv channels. This has given rise to multiple options for a user to select to consume entertainment, namely FuboTV, Sling, YouTube TV, Hulu, Philo. The channels bundled into each service provider package is different to each other like YouTube Tv offers 70 channels to live stream for \$49.99 a month whereas Philo offers 58 channels to live stream for \$20 a month and the channels offered by one is not necessarily offered by other. So, there is not one best service provider for all. It differs for each individual based on the content they want to consume and the budget and, the individual preferences changes over time which might result in reconsidering the service provider.

This app, 'Cord-Cutting-Calculator' recommends users a right service provider based on their requirements. The app home page looks like below. It has options to enter channels one need and the budget in which they are looking for. Channels to input are categorized into three, namely,

- Must have
- Would like to have
- Ok to have



The screenshot shows the web interface of the 'CORD CUTTING CALCULATOR' app. The browser address bar shows the URL 'revised-cord-cutter.herokuapp.com/users/1/calculator'. The app's navigation bar includes links for 'Channels', 'Devices', 'Packages', 'Set top boxes', and 'Account'. The main heading is 'Calculator'. The form contains several sections: 'Must have channels' with a note '(Be sure to select enough channels, or the calculate may not produce satisfying results.)' and a 'Select the Channels' dropdown; 'Would like to have channels' with a 'Select the Channels' dropdown; 'OK to have channels' with a 'Select the Channels' dropdown; 'Budget' with a text input field 'Enter your budget'; 'DVR' with a checkbox; and 'Only need one package?' with a checkbox. A large blue 'Calculate' button is at the bottom.

CORD CUTTING CALCULATOR Channels Devices Packages Set top boxes Account

### Calculator

Must have channels

(Be sure to select enough channels, or the calculate may not produce satisfying results.)

Select the Channels ▼

Would like to have channels

Select the Channels ▼

OK to have channels

Select the Channels ▼

Budget

Enter your budget

DVR ☐

Only need one package? ☐

Calculate

## User Stories

---

### 1. SCRAPE PROVIDER WEBSITE:

As an Admin User

I want to fetch channels from streaming provider programmatically

So that when creating a new Package, I don't have to manually enter channels by looking up on streaming provider website.

### 2. REVIEW & APPROVE NEW SCRAPPED CHANNELS:

As an Admin User

I want to approve new channels the app picks up from the streaming provider

So that I can update the system's global channels collection

### 3. EDIT NEW SCRAPPED CHANNELS:

As an Admin User

I want to Edit the new channel's name that the app picked up from the streaming provider website

So that I can register the channel name appropriately into the system

### 4. CREATE PACKAGE FROM SCRAPPED CHANNELS:

As an Admin User

I want to create the Package with channels extracted from streaming provider website

So that the app can save the Package and recommend it to users

### 5. SCRAPE PROVIDER WEBSITE IN EDIT PACKAGE:

As an Admin User

I want to fetch channels from streaming provider in edit package page

So that I can scrape the channels for existing package and update it

### 6. SETUP HEROKU ENVIRONMENT FOR SCRAPPER LIBRARIES:

As an application owner

I want to migrate to new Heroku

So that a new instance of the app can run using the new framework to extract channel from streaming provider websites

### 7. DATA MIGRATION

As an application owner

I want to migrate data to new instance of the app

So that user can use the application existing data and admin can use new features to create packages with less hassle

## 8. NEW STREAMING DEVICES:

As an Admin user

I want to update streaming devices of the app

So that they accurately represent real data and user can benefit more

### Scrape Provider Website

The main idea of this task is to make a program read and extract the channels off the provider website like Hulu. This would bypass the Admin's task of manually reading the channels off the provider website and entering the application, turning the task of creating new package very efficient and Admin friendly. There are couple of approaches to solve this and all of them involving parsing the webpage.

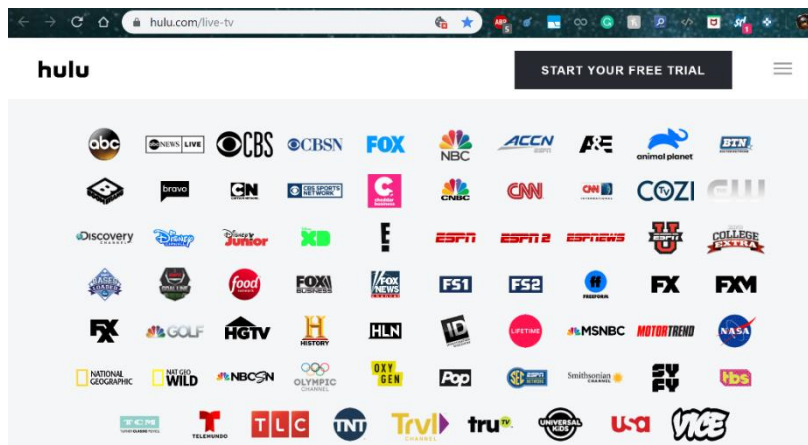
- First approach is to write specific parsers for each of the service provider like, one parser to extract channels from Hulu and another parser to extract channels from Sling. As webpage and their structure for each provider differ, different parsers are needed to parse different providers.
- Second approach is to write a generic parser for all providers. This generic parser does only extract specific information from the page, only channels. This approach still faces the challenges that arise from variant websites that each provider has. Below discussed is the analysis and method of solving these challenges.

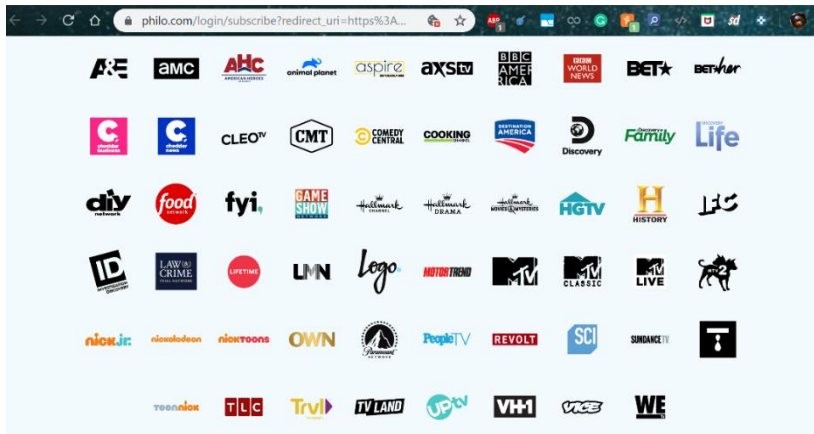
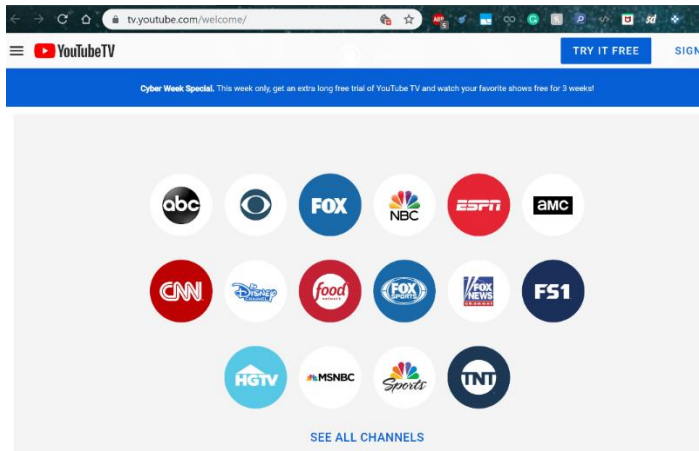
### GENERIC PARSER:

In addition to the above-mentioned challenge of different structured webpages for each provider, there is a challenge of parsing dynamic webpage. All the provider webpages are dynamic i.e. the html has only the skeleton of the webpage without of most of the content and the content itself is loaded by executing JS scripts during page loads. This dynamic nature of the webpage makes the simple static parsers like famous Nokogiri obsolete without support of other library that can run JS code.

Watir is one such library that helps in crawling and parsing dynamic webpages. But water requires a browser to run in the background and AWS Cloud9 environment doesn't support any heavy visually heavy applications to run in it, even headless browsers. It is extremely restrictive in this aspect. To make Watir work, the whole application setup was migrated to Google Cloud.

After analysis of webpages of Hulu, Youtube TV, Philo, Fubo TV, Sling a pattern is detected. Every provider displays these channels together as below.





All these channels are segregated into a single html element like div. A sample such div structure for Hulu is mentioned below. Div 'channels firestick-offer' is the main element which holds all the channels and each channel is wrapped in an additional div and holds few elements in it. The original element, responsible for the channel logo is img and it has attributes 'alt' and 'title' which holds the channel's name. The most commonly observed pattern in multiple providers webpage is this behavior that all channels are wrapped together and the html element an attribute mostly 'alt' or 'title' that holds the channel's name.

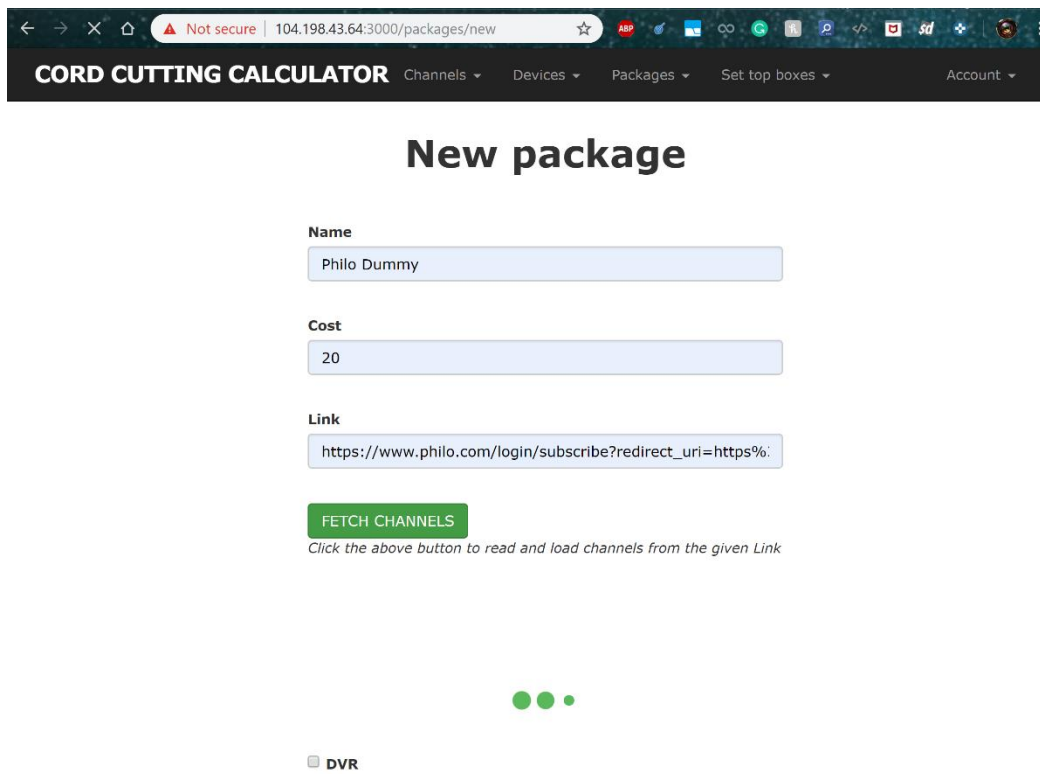
```
<head>_</head>
<div id="slickdeals" data-initialized="1">_</div>
<body data-gr-c-s-loaded="true">
  <div id="mount">
    <div>
      <div class="two-step-container">
        <div class="two-step-register-header firestick-offer">_</div>
        <div class="lineups">
          <div class="lineup">
            <div class="base">_</div>
            <div class="channels firestick-offer"> == $0
              <div class="channel">
                
                <div class="name">A&E</div>
              </div>
                <div class="channel">
                  
                  <div class="name">AMC</div>
                </div>
              <div class="channel">_</div>
              <div class="channel">_</div>
              <div class="channel">_</div>
              <div class="channel">_</div>
              <div class="channel">_</div>
              <div class="channel">_</div>
              <div class="channel">_</div>
              <div class="channel">_</div>
              <div class="channel">_</div>
              <div class="channel">_</div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>
```

The generic parser is built based on this observed pattern to extract channels. It uses simple approach of iterating through entire DOM of page and identifies the class-name of the div that holds any channel-name, its 'channel' from the example above. The generic parser leverages the channels already existing in the database to find this html element that encloses the given channel.

After discovering the class-name of this enclosing element, the algorithm looks for all html elements with the same class-name and extracts the channels present in attributes 'alt' and 'title' of each element. This might lead to duplicate channel-names and so are filtered.

Next step after extracting the list of unique channel names found on the website is to segregate these channels into two sets. One set is channels that the system already aware of, call it Known set. Cord-Cutter has a global list of channels in its database. Another set is the list of new channels that are now included into the global list, call it Unknown set.

The above algorithm concludes the generic parse which crawls the provider webpage and extracts all the channels into Known and Unknown Set, which are used in the later stages. Below is the app page that facilitates Admin to call this Generic parser upon clicking the button 'Fetch Channels'.



The screenshot shows a web browser window with the address bar displaying '104.198.43.64:3000/packages/new'. The page title is 'CORD CUTTING CALCULATOR'. The navigation bar includes links for 'Channels', 'Devices', 'Packages', 'Set top boxes', and 'Account'. The main heading is 'New package'. Below this, there are three input fields: 'Name' with the value 'Philo Dummy', 'Cost' with the value '20', and 'Link' with the value 'https://www.philo.com/login/subscribe?redirect\_uri=https%'. A green button labeled 'FETCH CHANNELS' is positioned below the link field. A note below the button reads: 'Click the above button to read and load channels from the given Link'. At the bottom of the form, there are three green dots and a checkbox labeled 'DVR'.

**CORD CUTTING CALCULATOR** Channels ▾ Devices ▾ Packages ▾ Set top boxes ▾ Account ▾

## New package

**Name**

**Cost**

**Link**

**FETCH CHANNELS**

*Click the above button to read and load channels from the given Link*

● ● ●

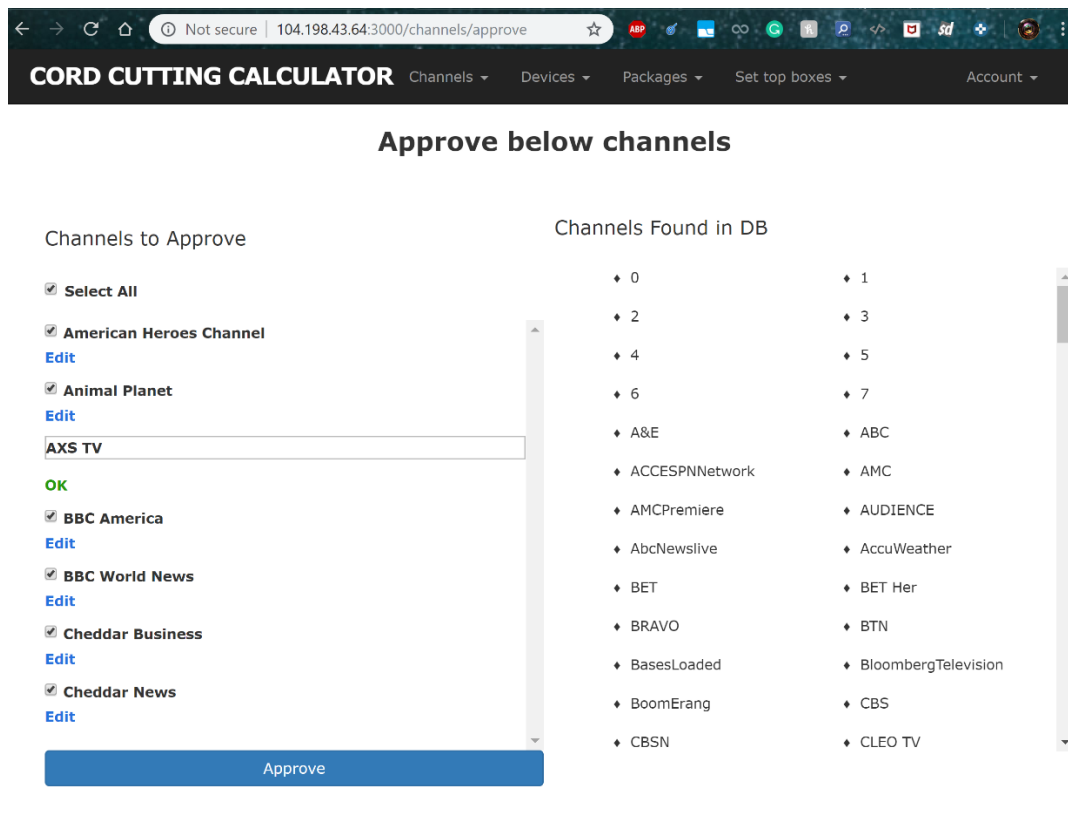
☐ DVR

The screenshot shows a web browser window with the address bar displaying '104.198.43.64:3000/package/parse\_chann...'. The application header is 'CORD CUTTING CALCULATOR' with navigation links for 'Channels', 'Devices', 'Packages', 'Set top boxes', and 'Account'. The main heading is 'New package'. The form contains three input fields: 'Name' with the value 'Philo Dummy', 'Cost' with the value '20', and 'Link' with the value 'https://www.philo.com/login/subscribe?redirect\_uri=https://'. Below the 'Link' field is a green 'FETCH CHANNELS' button with the instruction 'Click the above button to read and load channels from the given Link'. Below that is an orange 'REVIEW NEW CHANNELS (#7)' button with the instruction 'Click the above button to review new channels and approve them to add to application Database. It wouldn't add those new channels to package without reviewing and approving them.' At the bottom left, there is a checkbox labeled 'DVR'.

## Review & Approve New Channels

The Generic Parser extracts the channels from provider page and segregates them into two sets Known and Unknown set as explained in the previous sections. All the channels in the unknown set are not included in the package because they doesn't yet exist in the database. First these channels has to be added to database and then they can be attached with the package Admin wants to create. After the generic parse finishes scrapping the provider webpage and segregating the channels, it provides with an option to review and approve the new channels. This option is shown in the previous pic, orangish button with name 'REVIEW NEW CHANNELS'. On the button it has list of new channels like #7 in the above example.

Clicking on the 'REVIEW NEW CHANNELS' would redirect the app to approve screen. It looks like as presented below. It displays two columns, left column is the scrollable list of new channels the parser scrapped off the provider webpage and it includes an 'Approve' button at the end of column. Upon clicking this, these new channels are persisted into database. Right column is a scrollable list of existing channels in the database. This column is for the reference of existing channels in the database and Admin can refer to these when approving the new channels. Admin is provided with the flexibility to uncheck any of these new channels and they are not added to the database. It also has a feature to check all or uncheck all channels with 'Select All' option.



## Edit New Channels

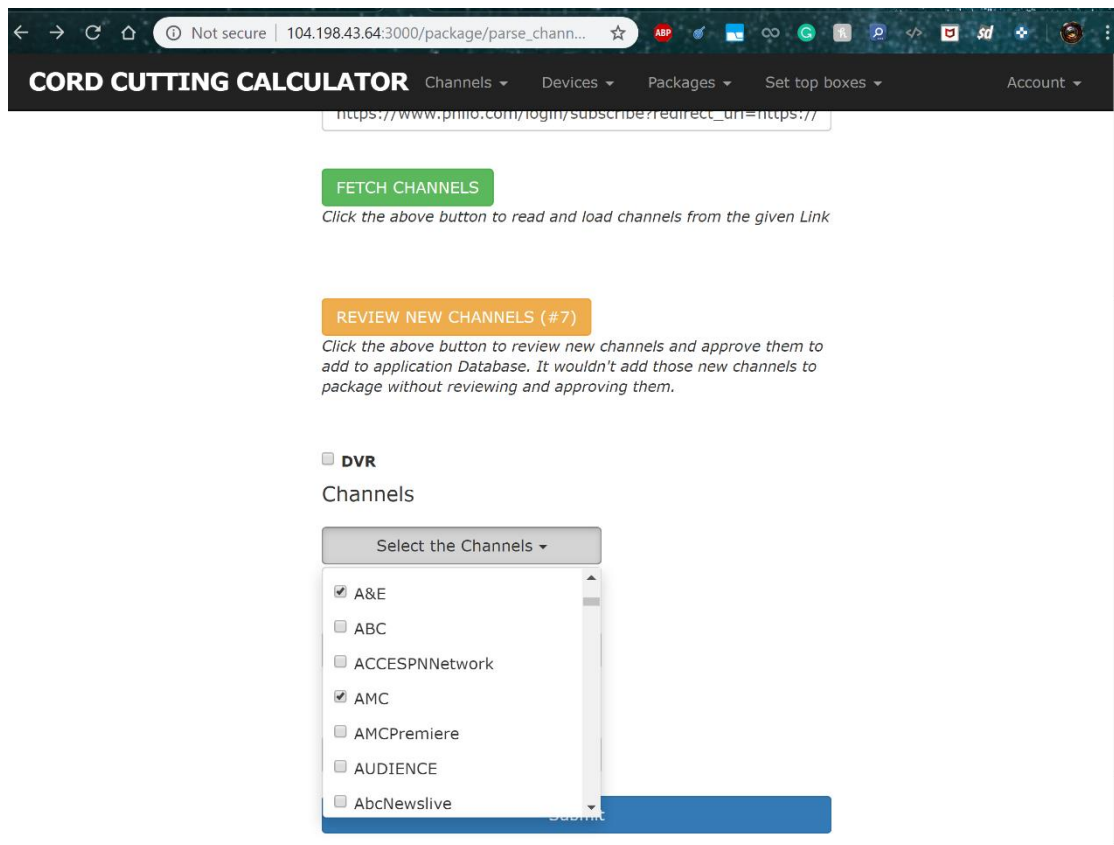
As presented in the above screenshot, approve page has Blue edit buttons below each channel which provides an option to edit any channel name Admin wishes to change. In the above example 'AXS TV' is in edit mode and displays Green 'OK' button below the channel. Clicking 'OK' would display channel back in check box mode with modified channel name as present in the image below.



## Create Package From Scrapped Channels

The task in this story is to create new package with all the channels scrapped off the provider website including the new channels the Admin has approved in the previous step. All the scrapped channels are pre-checked in the Channels listings. Admin has the flexibility to uncheck channels before submitting.



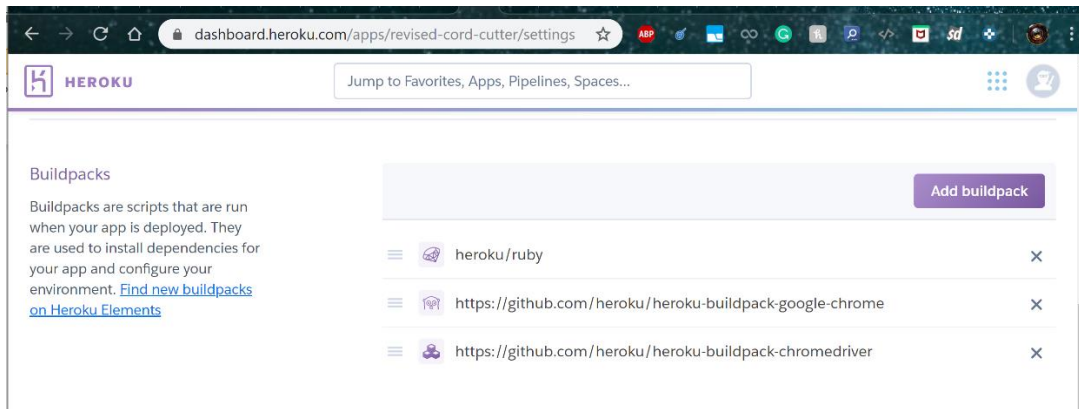


## Scarpe Provider Website In Edit Package

Edit package is very similar to new package page except that information about package is pre-filled including the channels. Edit package page has 'Fetch Channels' that would scrape the website and override the existing the checked list of channels with channels from provider website. If there are any new channels added to provider website and application isn't aware of them, application would show up an option to review and approve the new channels, similar to as mentioned in the previous section.

## Setup Heroku Environment

Revised Cord cutter uses Watir library which require Selenium drivers and chrome drivers. These libraries are provided in the buildpacks of the Heroku application as mentioned below. Buildpacks can be accessed from settings. More details about the steps are explain in this article, <https://www.simon-neutert.de/2018/watir-chrome-heroku/>.



## Data Migration

New application, 'revised-cord-cutter' is deployed at Heroku but it's database is empty and for the application to be useful the data existing in the current application has to be migrated to this new application. This data migration is necessary as the existing application has a huge data about streaming providers and the channels they support. This task is handled using Heroku Postgres tools.

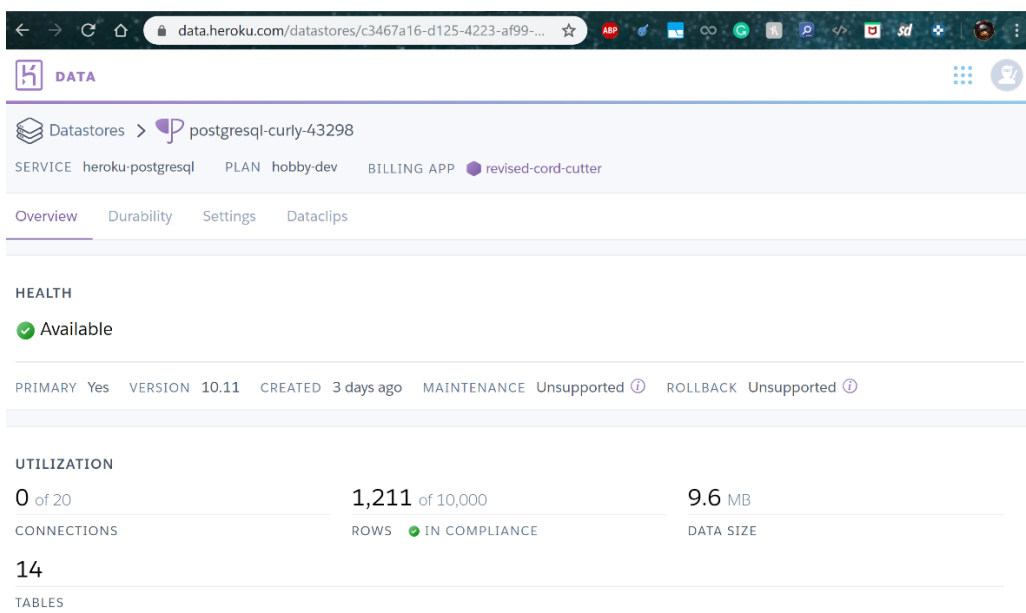
Heroku provides command line support for most of Postgres operations and one of them is for restores and backups. Heroku provides direct support for transferring data from another app to current app using below command.

```
$ heroku pg:copy 'ANOTHER-APP-DATABASE-URL' DB-NAME --app APP-NAME
```

Here DB-NAME and APP-NAME are Database name and App name as in Heroku console. After finding the database url for current app, data is transferred using following command,

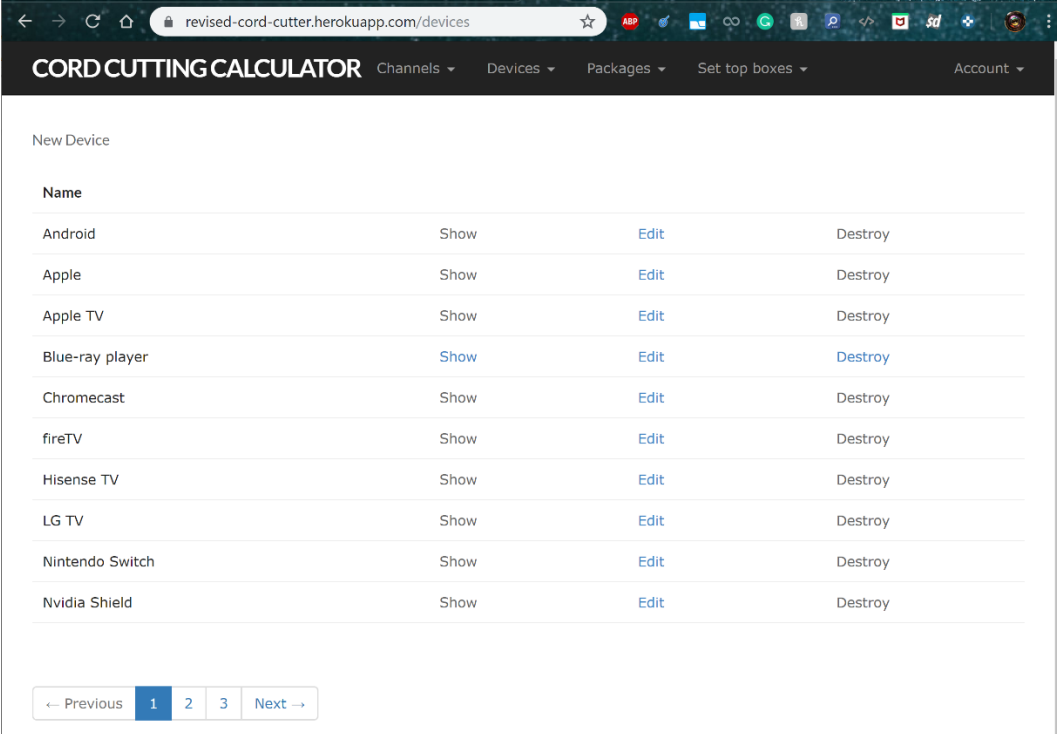
```
$ heroku pg:copy
'postgres://ioksixzxtgclid:9f3c14b1179a97d14a18c6100c1bbed559a091e873c298bdd94e510ab6f73
c4e@ec2-54-221-236-144.compute-1.amazonaws.com:5432/d8cbp9vdkfclpc' postgresql-curly-
43298 --app revised-cord-cutter
```

This migrated all 14 Tables of data from current app to new revised app.



## New Streaming Devices

Streaming devices in the existing application are too generic like 'Streaming media players' instead of Apple TV, Roku. Specific streaming devices will be more beneficial to user and this task is to create them. Created around 21 of them and below is the list them.



The screenshot shows a web browser at the URL `revised-cord-cutter.herokuapp.com/devices`. The application header is 'CORD CUTTING CALCULATOR' with navigation links for Channels, Devices, Packages, Set top boxes, and Account. The main content area is titled 'New Device' and contains a table of streaming devices. Each device has a 'Show', 'Edit', and 'Destroy' action link. At the bottom, there is a pagination control showing 'Previous', '1', '2', '3', and 'Next'.

Name	Show	Edit	Destroy
Android	Show	Edit	Destroy
Apple	Show	Edit	Destroy
Apple TV	Show	Edit	Destroy
Blue-ray player	Show	Edit	Destroy
Chromecast	Show	Edit	Destroy
fireTV	Show	Edit	Destroy
Hisense TV	Show	Edit	Destroy
LG TV	Show	Edit	Destroy
Nintendo Switch	Show	Edit	Destroy
Nvidia Shield	Show	Edit	Destroy

## Future Work

### Parser

The simple and smart parser designed in this implementation works great, but it has its limitations.

- It wouldn't capture channels mentioned under different class-named html element. This can be handled by finding class names of multiple channels instead of halting the algorithm after finding one channel in the page.
- On other side, there can be more than channels with same class name, like in case of Youtube TV it extracts information more than channels like devices and more.
- The scope of the parser can be improved by including extracting price of the package, devices supported by provider and there are providers with multiple packages like Sling Blue, Orange. Complex parsers can be designed to help capture all these data.

### Parsehub

Instead of building complex parsers, one can exporting the logic of parsing provider webpages to third party applications like ParseHub. ParseHub is UI friendly, simple to use parsing tool. User can select the elements he wants to parse from a webpage and set conditions and rules on the elements to parse. Simple demo is presented here, <https://www.parsehub.com/intro>. Data extracted can be saved as JSON or CSV formats on AWS. This extracted data can be accessed programmatically. ParseHub provides functionality of scheduled jobs

too which will help in exporting the scrapped data of these providers periodically by the application. It is worth trying if Admin can create a job for every service provider on ParseHub and the extracted data is ingested into system. It is worth trial and evaluate for our use cases.

## Application Details

---

- ✚ Revised Cord Cutter:  
<https://revised-cord-cutter.herokuapp.com/>
- ✚ Github: Branch – auto\_load\_channels  
[https://github.com/Ziyi1215/Cord-Cutting-Calculator/tree/auto\\_load\\_channels](https://github.com/Ziyi1215/Cord-Cutting-Calculator/tree/auto_load_channels)  
<https://github.com/dileep-g/Cord-Cutter> - forked standalone repo for this code  
*Not merged with master, cause master has a different code.*
- ✚ Current Cord Cutter:  
<https://cord-cutting-calculator.herokuapp.com/>

## Refereneces

---

- ✚ WATIR:  
<http://watir.com/>
- ✚ POSTGRES BACKUP & RESTORE:  
<https://devcenter.heroku.com/articles/heroku-postgres-backups>
- ✚ Heroku Environment setup:  
<https://www.simon-neutert.de/2018/watir-chrome-heroku/>
- ✚ ParseHub:  
<https://www.parsehub.com/intro>