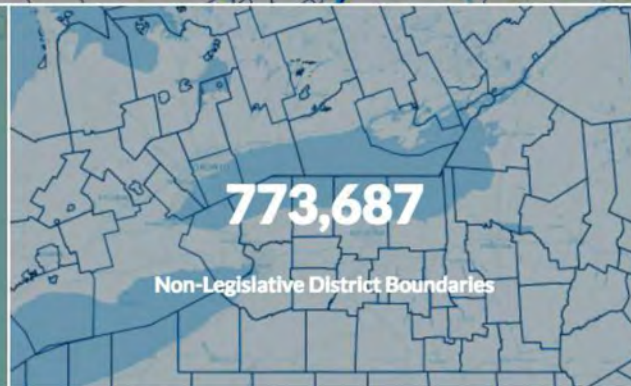
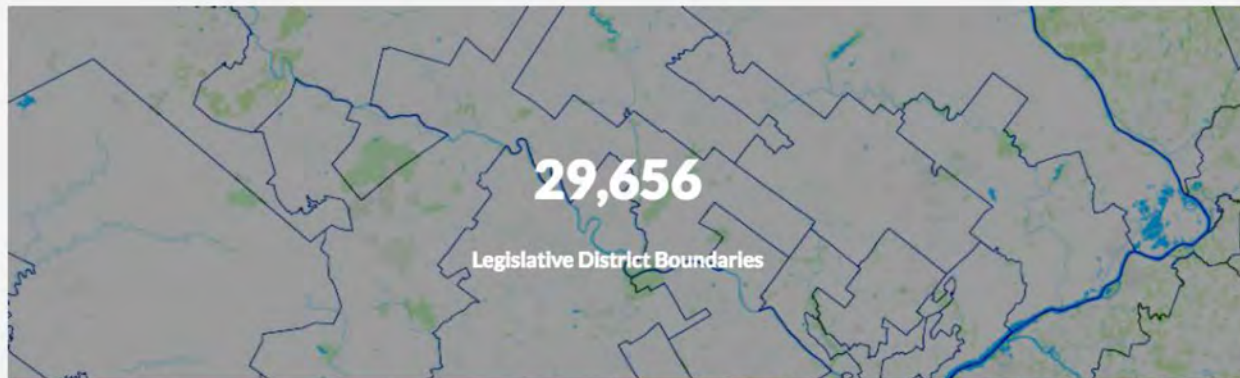


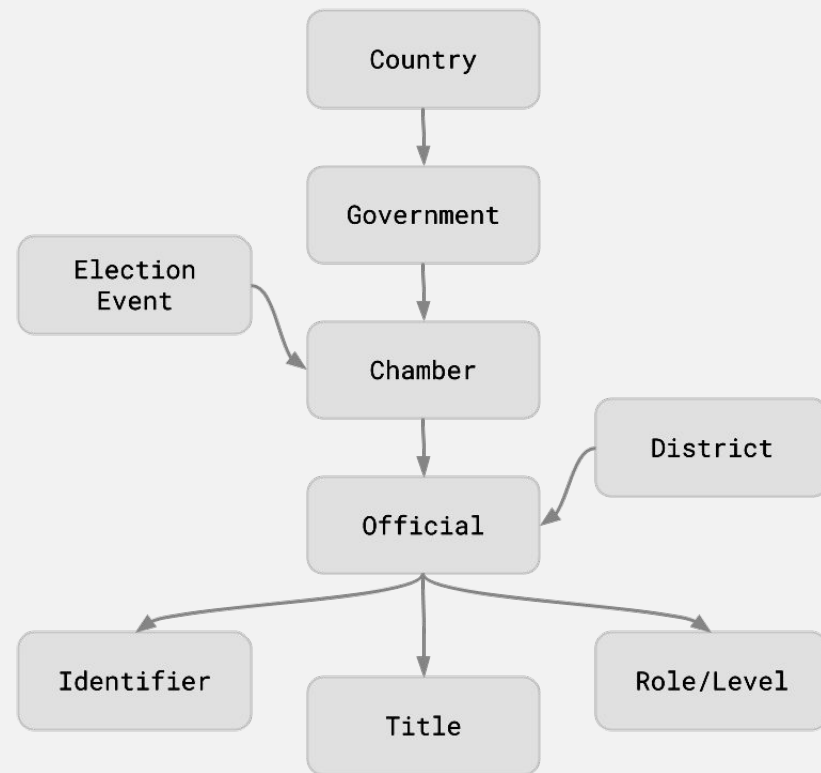
ZIP Code to Legislative District Matching via Cicero



What is Cicero?

Cicero is a database of elected officials and legislative districts spanning nine countries.

www.ciceroata.com



History of Cicero

- Started with local coverage in Philadelphia
- Greater Philadelphia Cultural Alliance
 - wanted an advocacy campaign because funding was going to be cut
 - needed to match members to their representatives

API

The Cicero API matches your addresses to their legislative districts and gives you back a wealth of elected official contact information.



Data Licensing

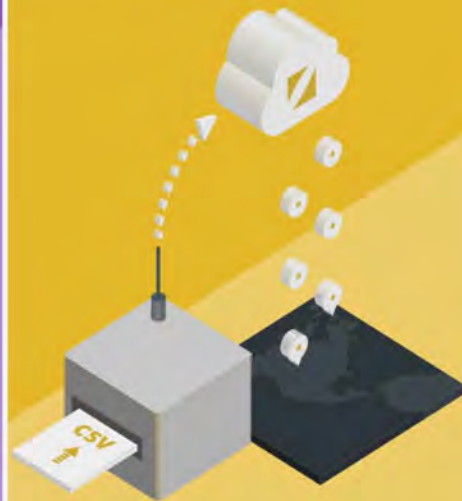
Whether you need an updated database of elected officials or the latest shapefile of legislative districts, we've got you covered.



District Match

A seamless way to match your addresses to districts and elected officials.

[Get Started](#)



live.cicero.com



Elected Officials & Districts

powered by  cicero

Find your elected officials & districts using our address-based lookup tool

990

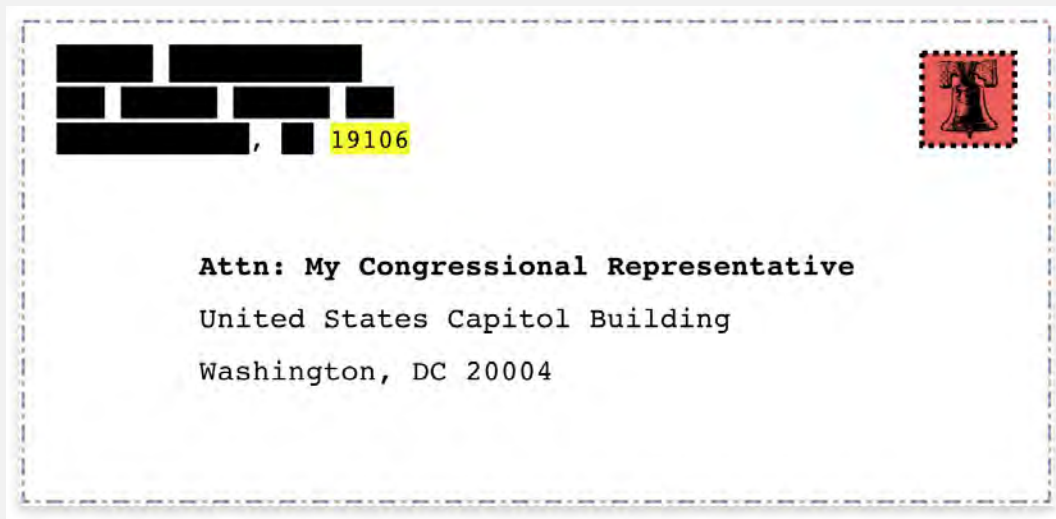
- 990 Spring Garden Street, Philadelphia, PA, United States
- 990 North 2nd Street, Philadelphia, PA, United States
- 990 North 5th Street, Philadelphia, PA, United States
- 990 Buttonwood Street, Philadelphia, PA, United States
- 990 North Randolph Street, Philadelphia, PA, United States

powered by  Google

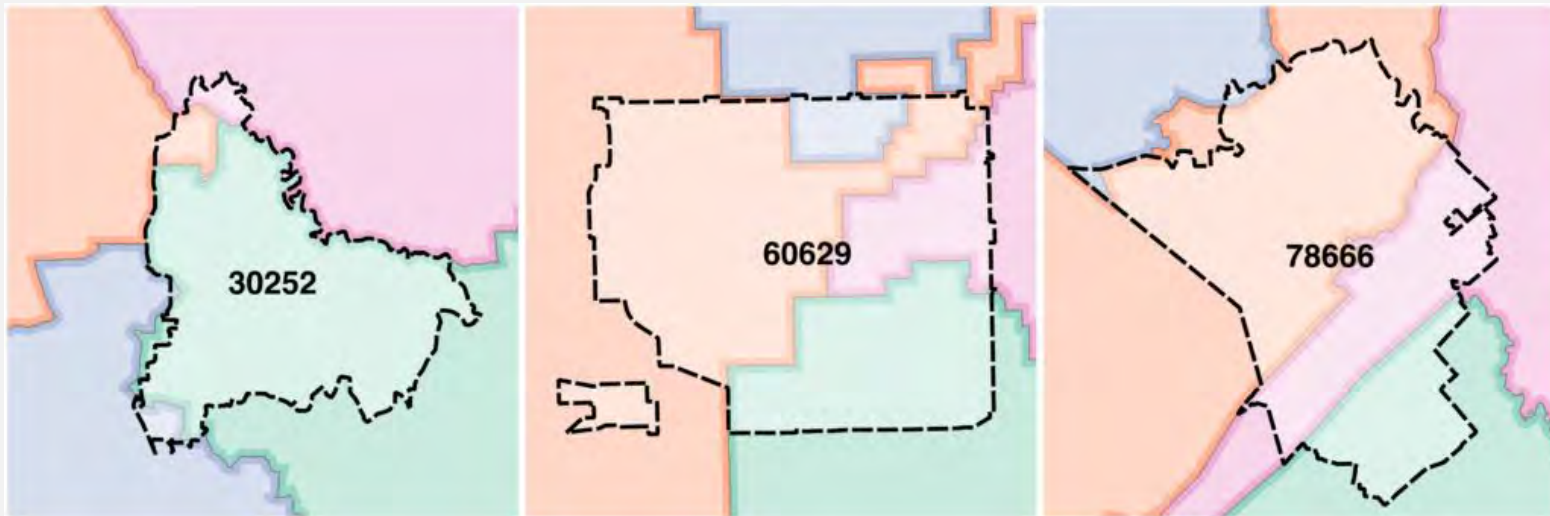
© 2017 Azavea. All Rights Reserved. [Terms of use](#) [Privacy policy](#) [Contact Us](#)

Aaron Dennis • adennis@azavea.com

Challenge: ZIP Code to District matching



Challenge: ZIP Code to District matching



Why use ZIP Codes?

- widely adopted
- commonly understood
- don't require personally identifiable information (like street addresses)
- Census Bureau provides a ZCTA (ZIP Code Tabulation Area) dataset

ZIP Codes don't really exist

- **TRUE** → ZIP Codes are not actually areas/polygons
- **TRUE** → ZIP Codes are collections of postal delivery routes and points
- **FALSE** → ZIP Codes are useless for defining the boundaries of a region

Challenge: ZIP Code to District matching

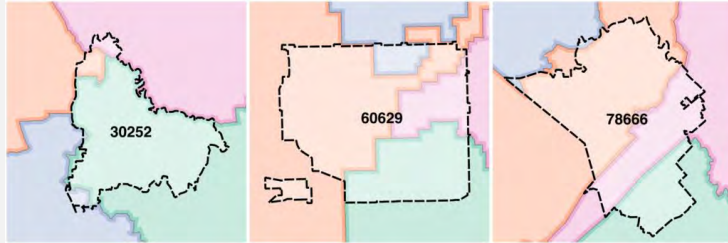
Conventional method:

matched with center point of ZIP Code

Our method:

matched to most likely district based on population distribution

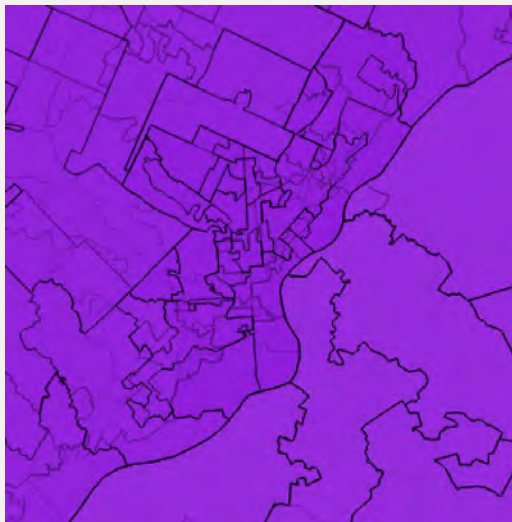
Population based matching:



ZIP Code	Congressional District	Match Probability
30252	Georgia's 10th	90%
	Georgia's 13th	7%
	Georgia's 3rd	2%
	Georgia's 4th	1%
60629	Illinois's 3rd	54%
	Illinois's 7th	20%
	Illinois's 1st	18%
	Illinois's 4th	8%
78666	Texas's 35th	56%
	Texas's 21st	34%
	Texas's 15th	10%
	Texas's 25th	0%

Input Data:

1. ZIP Code Tabulation Areas (US Census)
2. Legislative District Boundaries (Cicero)
3. Census Block Population Counts



Technical Details:

Docker Containers

Ubuntu

- GDAL
- Postgres Client
- Python
- YAML
- Tippecanoe & MBUtil

PostGIS

- SQL
- spatial functions

Output of automated workflow:

Products

Basic

- matched with center point of ZIP Code

ZIP-to-District.csv

Premium

- matched to most likely district based on population distribution

ZIP-to-District-premium.csv

ZIP-to-District-premium.json

Example output JSON:

```
"19123": {  
  "sld1": [  
    {  
      "id": "ocd-division/country:us/state:pa/sld1:181",  
      "pct": 0.638  
    },  
    {  
      "id": "ocd-division/country:us/state:pa/sld1:175",  
      "pct": 0.362  
    }  
  ],  
  ...  
}
```

Open Civic Data
Identifiers!
(OCD ID)

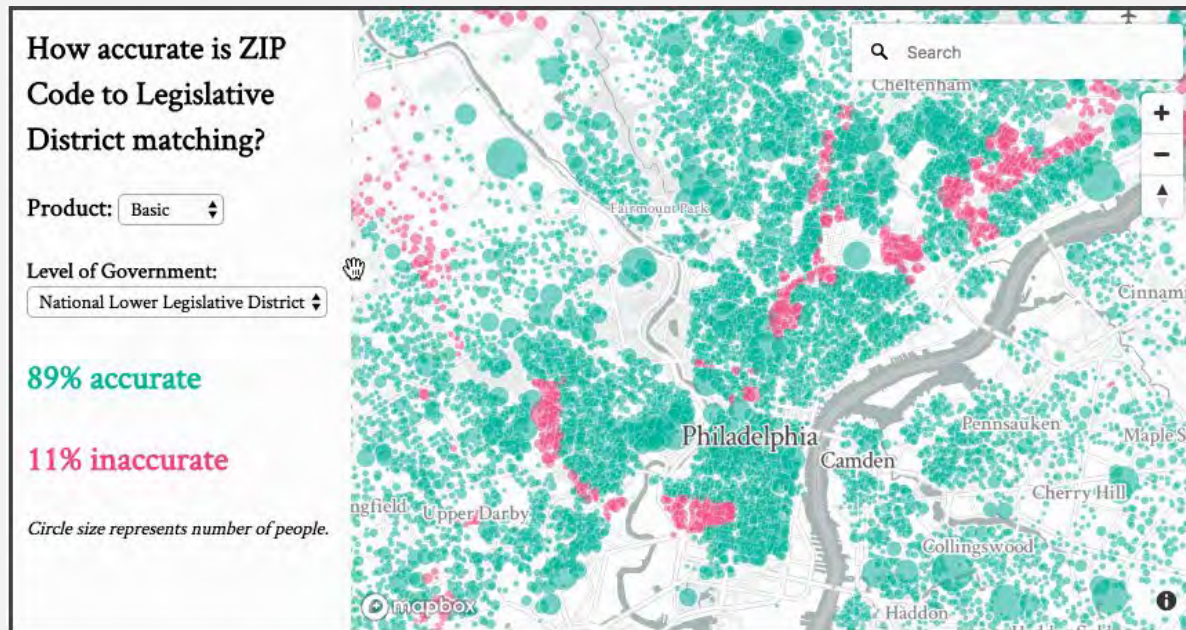


Accuracy of centroid vs. population:

National Accuracy

Level of Government	Basic Product	Premium Product
National Upper Legislative District	99%	99%
National Lower Legislative District	92%	94%
State Upper Legislative District	85%	88%
State Lower Legislative District	75%	80%

Regional Accuracy:



How accurate is ZIP Code to Legislative District matching?

Product: Basic

Level of Government:

National Lower Legislative District

88% accurate

12% inaccurate

Circle size represents number of people.



How accurate is ZIP Code to Legislative District matching?

Product: Premium

Level of Government:

National Lower Legislative District

90% accurate

10% inaccurate

Circle size represents number of people.



How accurate is ZIP Code to Legislative District matching?

Product: Basic

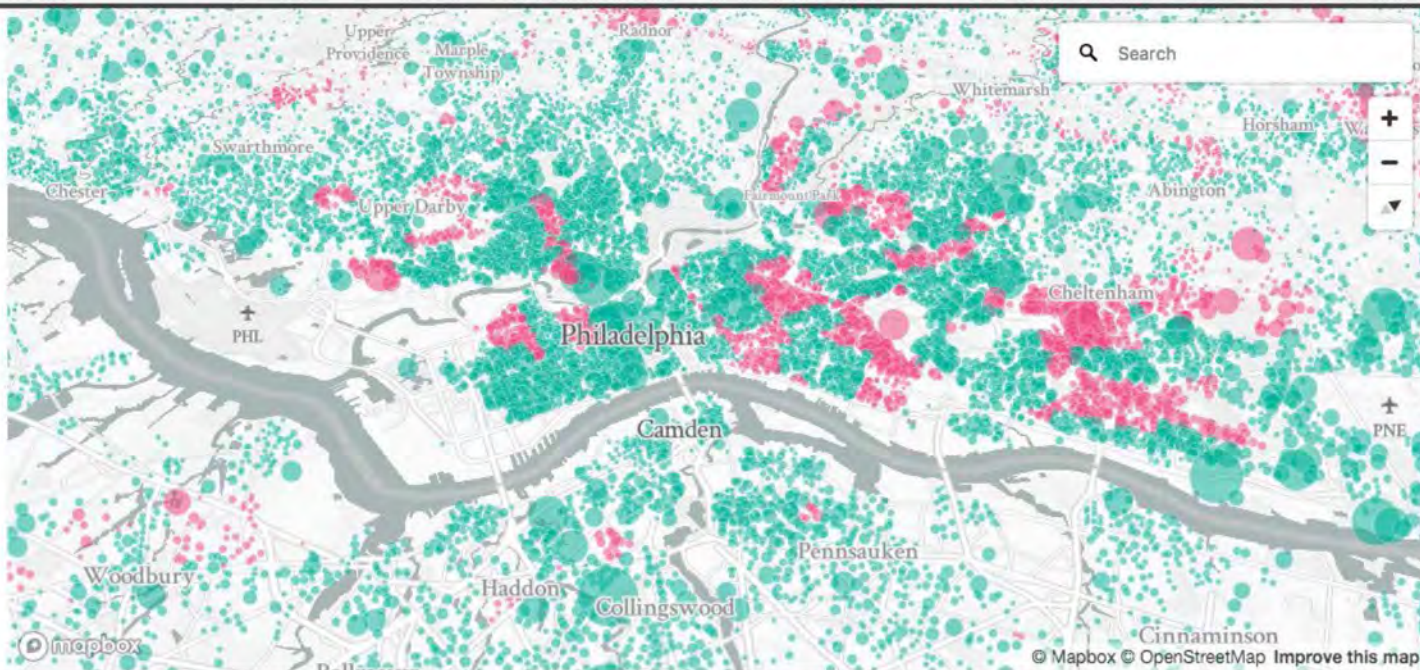
Level of Government:

State Upper Legislative District

79% accurate

21% inaccurate

Circle size represents number of people.



How accurate is ZIP Code to Legislative District matching?

Product: Premium

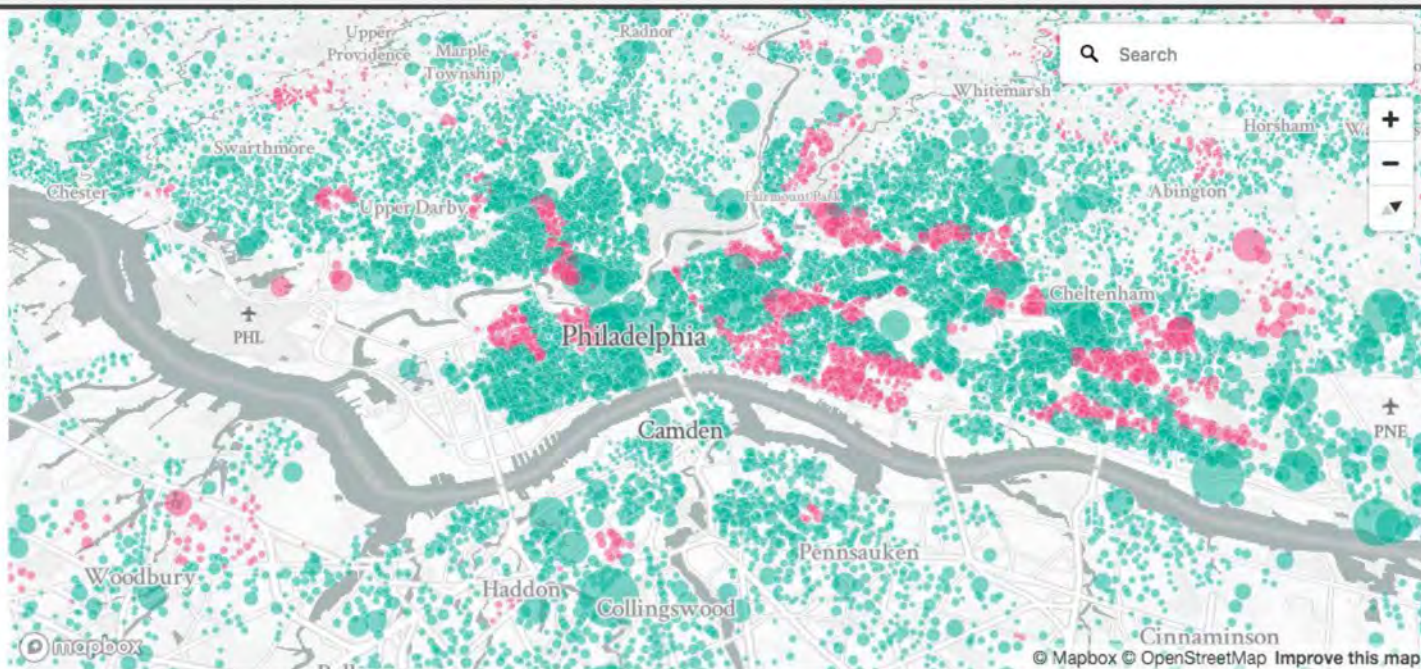
Level of Government:

State Upper Legislative District

84% accurate

16% inaccurate

Circle size represents number of people.



How accurate is ZIP Code to Legislative District matching?

Product: Basic

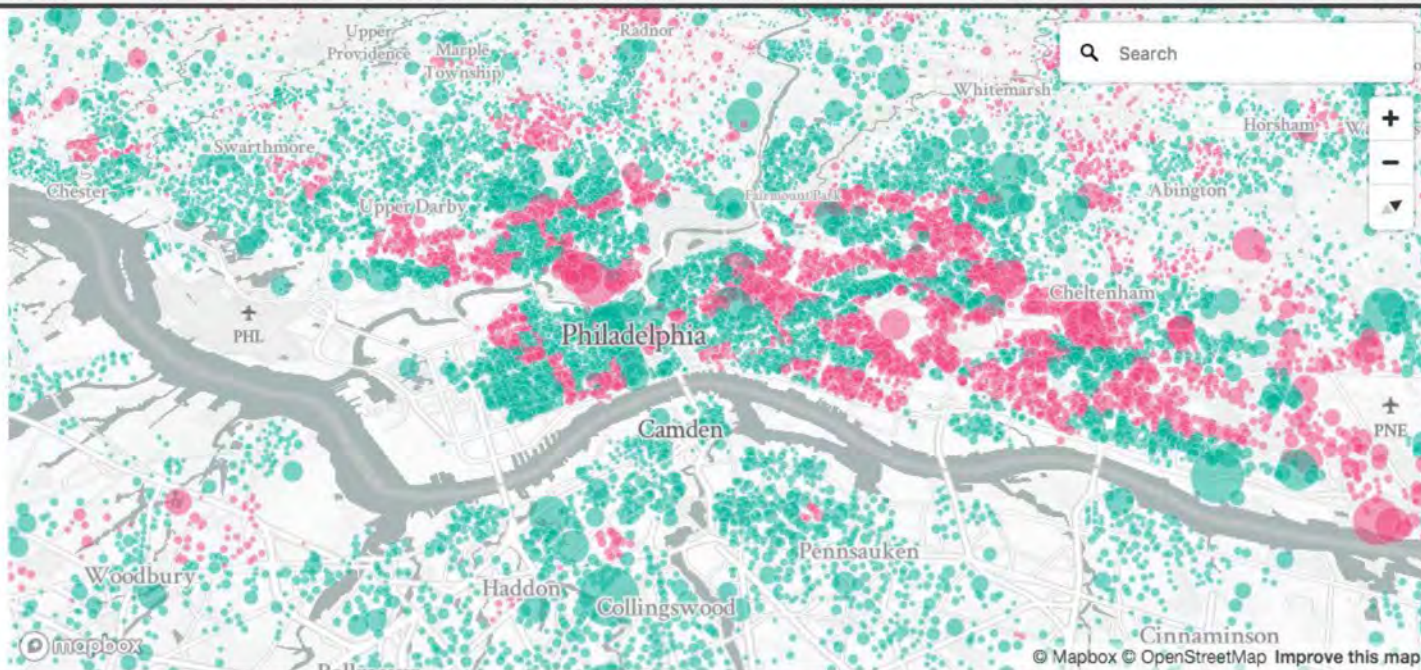
Level of Government:

State Lower Legislative District

66% accurate

34% inaccurate

Circle size represents number of people.



How accurate is ZIP Code to Legislative District matching?

Product: Premium

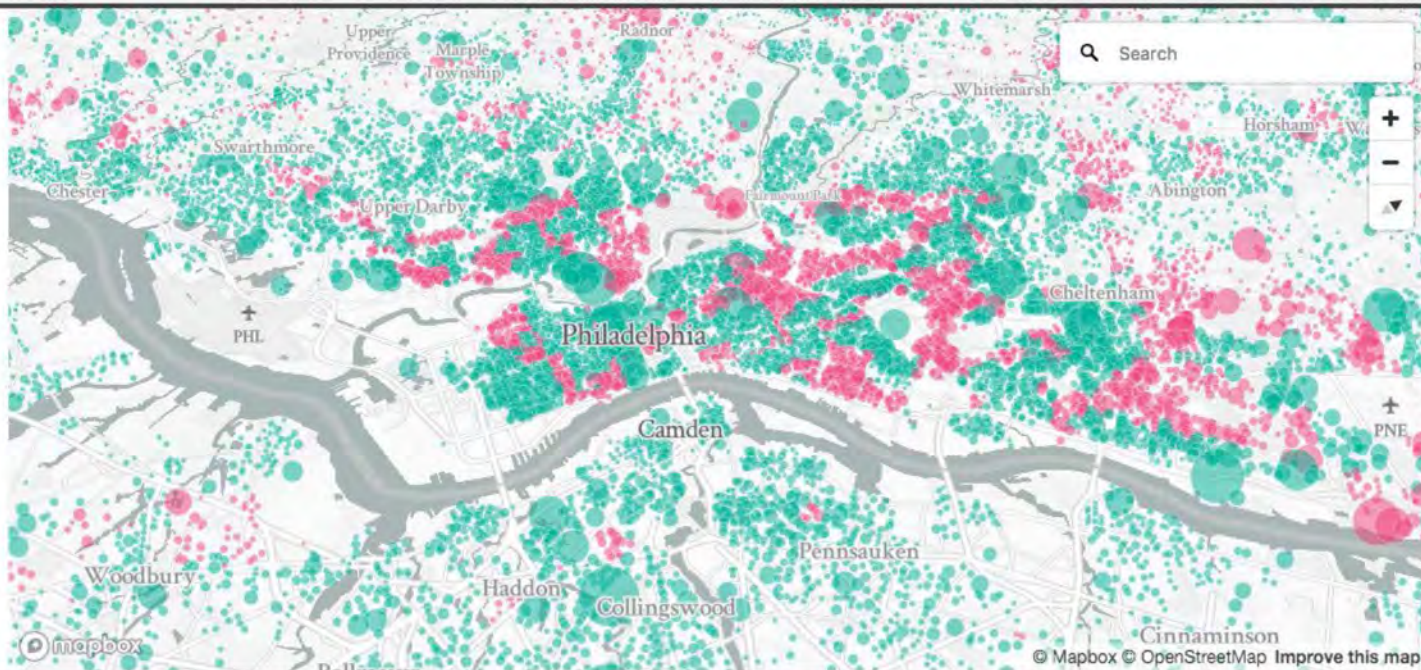
Level of Government:

State Lower Legislative District

72% accurate

28% inaccurate

Circle size represents number of people.



`cicero.com`

`@CiceroAPI`

`github.com/cicero-data`

Deriving Value from Aerial Imagery

Improving Deep Learning Workflows

With Open Source Software



Source:
[DailyOverview](#)

How to derive value from aerial imagery?

Aerial and satellite imagery gives us the unique ability to look down and see the earth from above. It is being used to [measure deforestation](#), [map damaged areas after natural disasters](#), [spot looted archaeological sites](#), and has many more current and untapped use cases. At Azavea, we understand the potential impact that imagery can have on our understanding of the world. We also understand that the enormous and ever-growing amount of imagery presents a significant challenge: how can we derive value and insights from all of this data? There are not enough people to look at all of the images all of the time. That's why we are building tools and techniques to allow technology to see what we cannot.

Source: <https://www.azavea.com/blog/2017/05/30/deep-learning-on-aerial-imagery/>

How to help governments and local stakeholders respond to deforestation?

Every minute, the world loses an area of forest the size of 48 football fields. And deforestation in the Amazon Basin accounts for the largest share, contributing to reduced biodiversity, habitat loss, climate change, and other devastating effects. But better data about the location of deforestation and human encroachment on forests can help governments and local stakeholders respond more quickly and effectively.

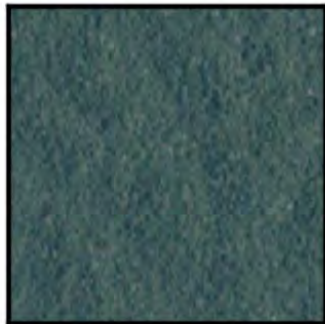


Source: <https://www.kaggle.com/c/planet-understanding-the-amazon-from-space>

Raster Vision: 23rd of 938, 93.4% F-Score



agriculture, clear,
habitation, primary,
road, cultivation



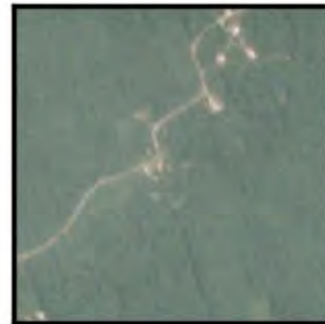
clear, primary



agriculture, primary,
water, cultivation,
habitation, partly_cloudy
missing: clear, road



agriculture, clear,
primary, road, haze



habitation, primary, road
agriculture, clear, water
missing: haze

Source: <https://github.com/azavea/raster-vision>

1. Result demonstration: [Kaggle BE](#)


Identifying riff-raff in the Amazon with machine learning

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer sagittis, est ac condimentum feugiat, libero lacus euismod leo, eu laoreet massa ante nec mauris. Donec vehicula ornare elit id porttitor.


Tags

- Artisan Mining
- Artisan Mining
- Artisan Mining
- Artisan Mining
- Artisan Mining
- Artisan Mining
- Artisan Mining
- Artisan Mining


6 results



Mining 6.5 Roads 2.5
Really cloudy 8.4



Slash & Burn 8.4



This chip is from 12/4/17

Source: Jeff Frankl


2. Capacity building: GeoTensorFlow


 yoninachmany / geotensorflow

forked from [geotrellis/geotrellis-sbt-template](#)



 Code

 Pull requests 2

 Projects 0

 Wiki

 Settings

Insights ▾

Predict labels on rasters processed by GeoTrellis using pre-trained Raster Vision models

Source: <https://github.com/yoninachmany/geotensorflow>

2. Capacity building: GeoTensorFlow

The `LabelImage` example demonstrates use of this API to classify images using a pre-trained `Inception` architecture convolutional neural network. It demonstrates:

- Graph construction: using the `OperationBuilder` class to construct a graph to decode, resize and normalize a JPEG image.
- Model loading: Using `Graph.importGraphDef()` to load a pre-trained Inception model.
- Graph execution: Using a `Session` to execute the graphs and find the best label for an image.

Source: https://www.tensorflow.org/api_docs/java/reference/org/tensorflow/package-summary

2. Capacity building: GeoTensorFlow

<input type="checkbox"/>	🚨 3 Open ✓ 0 Closed	Author ▾	Labels ▾
<input type="checkbox"/>	🚨 Initial Tensor => Tensor predictions based on frozen models #3 opened 7 days ago by lossyrob		
<input type="checkbox"/>	🚨 Example notebook for how to freeze model architecture and weights #2 opened 7 days ago by lossyrob		
<input type="checkbox"/>	🚨 Create initial repo structure #1 opened 7 days ago by lossyrob		

Source: <https://github.com/azavea/geotensorflow/issues>

3. Dataset exploring: SpaceNet Notebook



Source: <http://blog.digitalglobe.com/developers/the-spacenet-challenge-help-us-to-harness-machine-learning-to-make-maps-more-current-and-complete/>

3. Dataset exploring: [SpaceNet Notebook](#)

Using a [GeoDocker container with Jupyter and GeoPySpark](#), a [single notebook](#) can access the SpaceNet data on [AWS](#), download Rio raster and vector data with [Boto](#), wrangle imagery with [GDAL](#), ingest imagery for fast viewing with [GeoPySpark](#), and show Rio's outline, imagery, and building footprints on a map with [GeoNotebook](#).

Source: Draft of "Exploring SpaceNet Data with GeoPySpark"

3. Dataset exploring: SpaceNet Notebook

Showing Rio's outline, imagery, and building footprints on a map with GeoNotebook

```
In [4]: from geonotebook.wrappers import VectorData
outline_vector = VectorData(outline_filename)
outline_polygons = [polygon for polygon in outline_vector.polygons]
outline_polygon = outline_polygons[0]
outline_centroid = outline_polygon.centroid
x = outline_centroid.x
y = outline_centroid.y
z = 12
M.set_center(x, y, z);
M.add_layer(outline_vector, name=outline_key);
```



Source: <https://github.com/geodocker/geodocker-jupyter-geopyspark/pull/27>



Think Aerial and Open Source

This Overview captures rowing shells on the Schuylkill River, which runs through the center of the city. (Source: [Daily Overview](#))

Thanks!

Git-ting Started With GitHub Desktop

tnation@azavea.com



Acknowledgements

This training and the images used therein are from <https://git-scm.com/book/en/v2/> under the [Creative Commons License](#)

Overview

- VCS & Git
 - VCS overview
 - What is Git?
- Setting up GitHub & GitHub Desktop
- GitHub Pages
- Git Concepts
 - Structure of a Git project
 - Making Changes to a File
 - Branching/Forking
- GitHub Collaboration tools
 - Issues
 - Milestones
 - Pull Requests
 - Git Flow

Commonly Used Git Terms

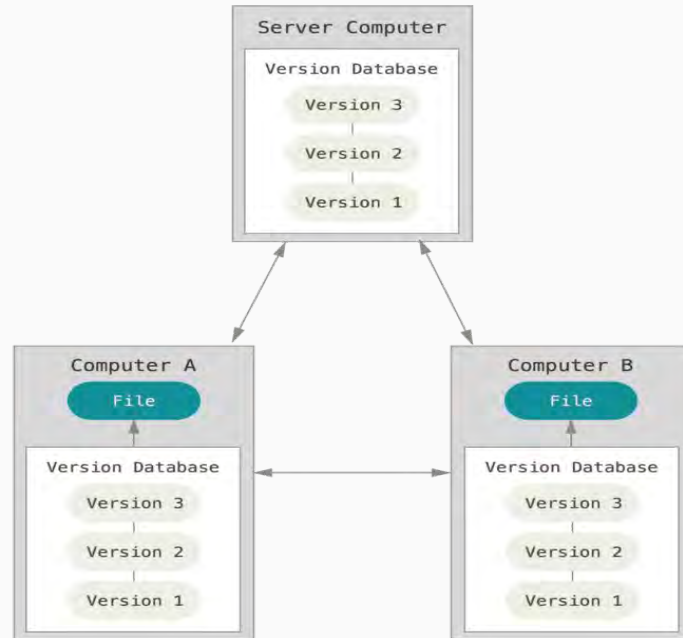
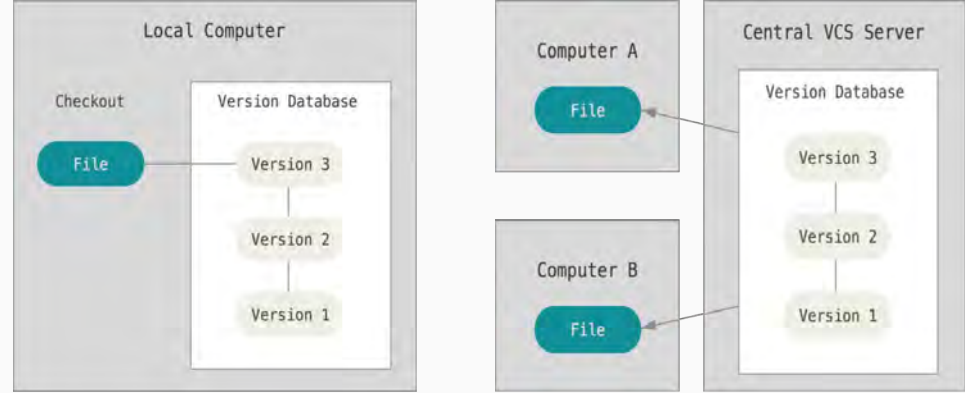
- This Presentation is light on actual git commands, but git-scm.com provides a comprehensive cheat sheet [here](#).
- Repository: Data structure containing a history of file changes. A project **contains** a git repository, but project and repository are often used interchangeably.
- Pull: Retrieve changes from a remote repository, and merge them into the local one.
- Push: Update the remote server's version history with your local changes
- Commit: Record changes to a repository

Commonly Used Git Terms (cont'd)

- Branch: A copy of the revision history, stored in the repository. Generally used for short term feature development, and merged back into the main tree.
- Fork: A copy of revision history, stored in another repository. Generally used to track changes that will not be merged back into the main repository.
- Pull Request/Merge Request/Changelist: the nomenclature varies by organization, but these are requests to merge changes into the main repository.

Version Control Systems (VCS)

- “Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later” (git-scm.com)
- Particularly useful to engineers because it allows us to do quick rollouts/rollbacks, and makes it easy to pinpoint the source of software bugs.
- VCS types: Localized, Centralized, and Distributed
- Git is a distributed VCS.



What is Git?

- From <https://git-scm.com> : “Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.”
- Created in 2005, after the VCS used by Linux Kernel developers revoked their free-use license.
- Used by Facebook, Twitter, Google

What is GitHub?

- GitHub is a project collaboration service that hosts git repositories and handles other aspects of project management.
 - Works with the Git CLI; all you need is an account.
- The GitHub API is a useful tool for organizing projects and team collaboration.
 - Azavea operations collaborates with all teams in the organization. GitHub issues allows us to keep track of tasks across multiple projects.

GitHub Desktop (and other Git GUI tools)

- Graphical User Interfaces (GUIs) make it easy to interact with Git without remembering any commands.
- Visualizing branches makes it easier to do git operations.
- GitHub desktop is developed by GitHub Engineering, and enables users to do the basic Git commands: branching, pulling, pushing, and committing.
 - This will be enough for most Git/GitHub users, but it appears that some of the more complex commands are missing
- A similar tool, SourceTree by Atlassian, has a more comprehensive set of options available (including Git Flow integration) for advanced users.

GitHub Pages

- Host your website on a GitHub Repository
- Great for blogs, and marketing sites
- More info at <https://pages.github.com/>

Exercise: Setting Up GitHub Desktop

- Create a GitHub account at <https://github.com/join>
- Download Git Desktop from <https://git-scm.com/download/guis/>
 - Sign in and configure your Name and Email address.

Structure of a Git Project

- Git directory (repository): contains the revision database and metadata.
 - Previous commits to files are stored here.
- Working tree: A snapshot of one version of the project, stored on disk. Usually, but not necessarily, the latest revision.
 - Created from the changes stored in the Git version database.
 - Ephemeral: changes made in the working tree are not tracked.
- Staging area: a section of the git repo that contains information about files that have been modified, but not committed.

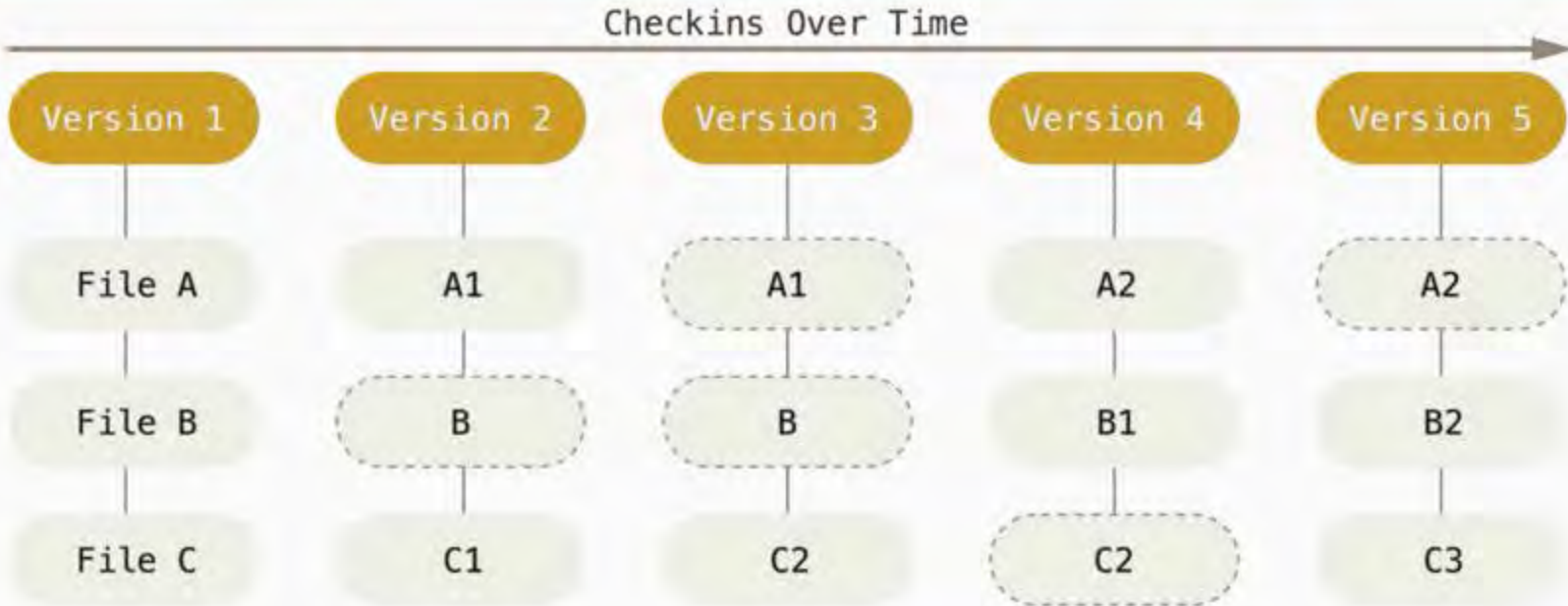
Making Changes to a File

- **Clone** the project you'd like to make edits to
 - **git clone <repository-uri>**
- **Checkout** a version of the project into the working tree
 - **git checkout -b test-branch**
- Make the necessary edits, then **add** them to the Staging area
 - This is ***VERY important***: if you don't add your changes to the staging area, they won't be saved to the version database.
 - **git add edited-file.txt**

Making Changes to a File (cont'd)

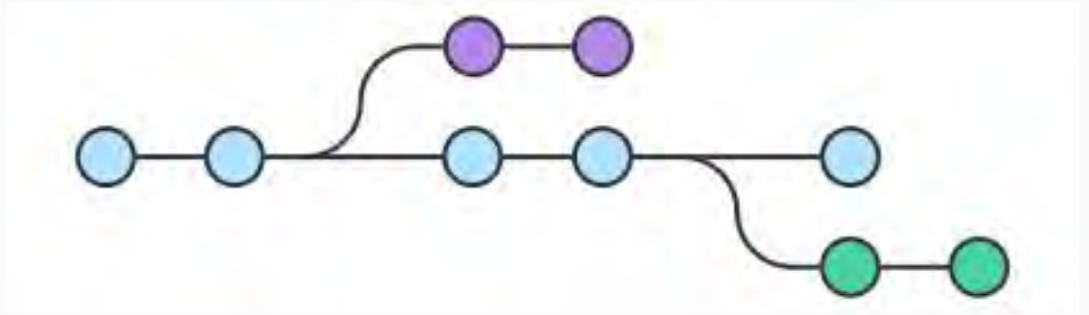
- Review your changes; once you're satisfied, **commit** them to save your revisions to the database.
 - Inspect changes with **git diff**
 - Save your revisions with **git commit**
- **Push** your changes to the remote
 - **git push origin test-branch**

How Git Views Version History



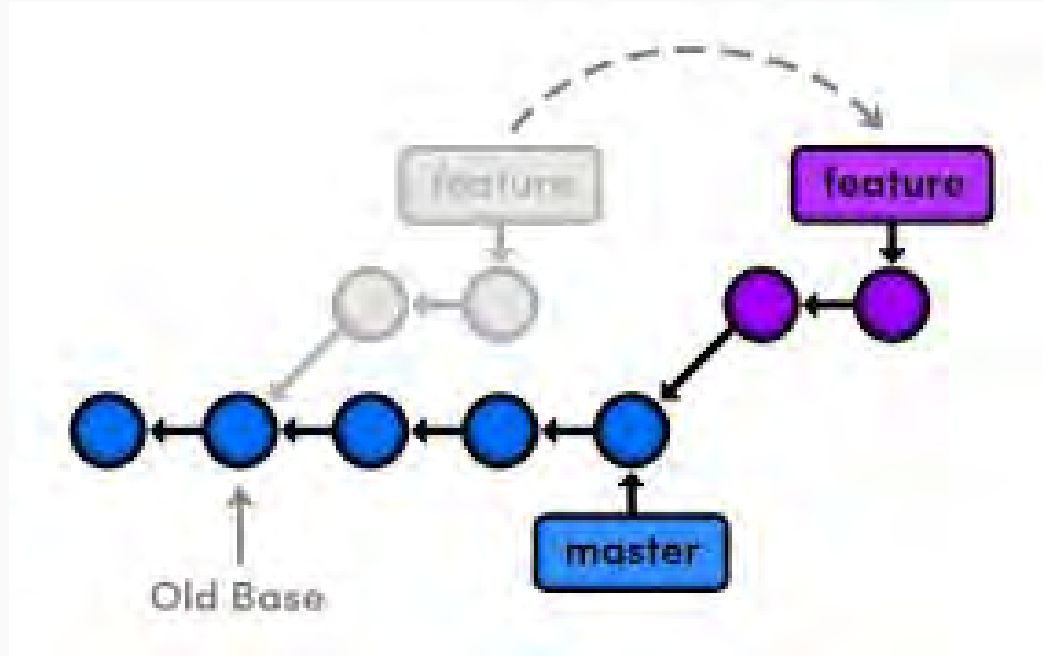
Making Major Changes to a Project

- **Branching** creates a copy of the main revision history that runs parallel to the main history. You can make changes to this copy of the repo without impacting the main line of development.
- **Forking** a repository does the same thing as branching, but it grants full ownership to the user that creates the fork. Forking is useful for creating an idea based on another project, or contributing to a project where you don't have Push access.
- Allows people to experiment with different versions of the source code at the same time.



(Not) N'Sync: Branching/Forking Caveats

- Merge conflicts happens when you try to merge your branch into the main branch, but they're too far out of sync.
 - Your branch contains changes to a file that's deleted
 - Different changes to the same file
- To limit the amount of conflicts, **git pull** changes into the main branch, then use **git rebase** to add your commits to the top of the newly-updated main branch.
- Sometimes, merge conflicts have to be fixed manually.



Git Workflow Exercise: Creating a GitHub Page

- Download a zip file of <https://github.com/tnation14/sample-github-pages-template>
- Create a repository in GitHub Desktop: <https://github.com/<your username>/<your username>.github.io> (i.e. <https://github.com/tnation14/tnation14.github.io>)
 - Take note of the local folder where the repo will live.
- Create a new branch:
 - Click **Current Branch > New**
 - Name the branch **master**
- **Copy** the contents of the zip file into your repository.
- **Edit** index.html to say “Hello, <your name>”.
- Open index.html in a web browser
- Add the new files to the staging area, commit changes, then push them to a remote branch
- Merge your changes to the master branch.

GitHub Collaboration Tools

- Issues: GitHub's way of tracking bugs
- Milestones: Grouping issues by deadline. Also helps break issues down into more manageable chunks
- Pull requests/Merge Requests (PRs/MRs): a request to merge changes from a branch or fork into the main branch.

GitHub Issues: Because There's No Such Thing as Perfect Software

- Useful tool for iterative development
 - Plan, Build, Test, Deploy, Create Issues
- Issue comments are great places for planning between multiple team-members
- Break down complex tasks into more manageable steps
- Good to reference as context for a Pull Request
- Nice feature of GitHub API is that you can link a PR to an issue, so that when you merge the PR the issue is closed as well.
- Example: GeoTrellis [Issue](#) and [Pull Request](#)
- <https://github.com/user/repo/issues>

Milestones

- Organizes disparate issues into a single deliverable for time tracking purposes.
 - Multiple kinds of customer issues can be grouped under a single release milestone
- Set timelines for features
- Used in conjunction with Issues, Milestones are a great way to measure how your team's expected efficiency matches up with its actual efficiency.
- <https://github.com/user/repo/milestones>

Pull Requests

- Asking permission to merge my branch revision history into the main branch
- Place to peer-review code for correctness and readability.
- PR Etiquette
 - Give your PR a thorough self-review before submitting it to someone else.
 - <https://github.com/user/repo/compare/base-branch...your-branch>
 - Describe of the issue you're trying to fix (or link back to an issue)
 - Include good testing instructions
 - When adding changes, avoid rebasing. This can cause merge conflicts for the person testing your commit locally.
 - Before merging, do one final rebase to put your work into logical commit steps
- <https://github.com/user/repo/pulls>

Git Flow

- Branching pattern for teams that do rapid iterative development
- Active development happens on "feature" branches
- Completed work is merged from the Feature branch into develop.
- Release branches contain work that's ready to be deployed
- Keeps in-progress work separate in progress work from completed work, and allows multiple teams to work on projects without getting in each other's way.

