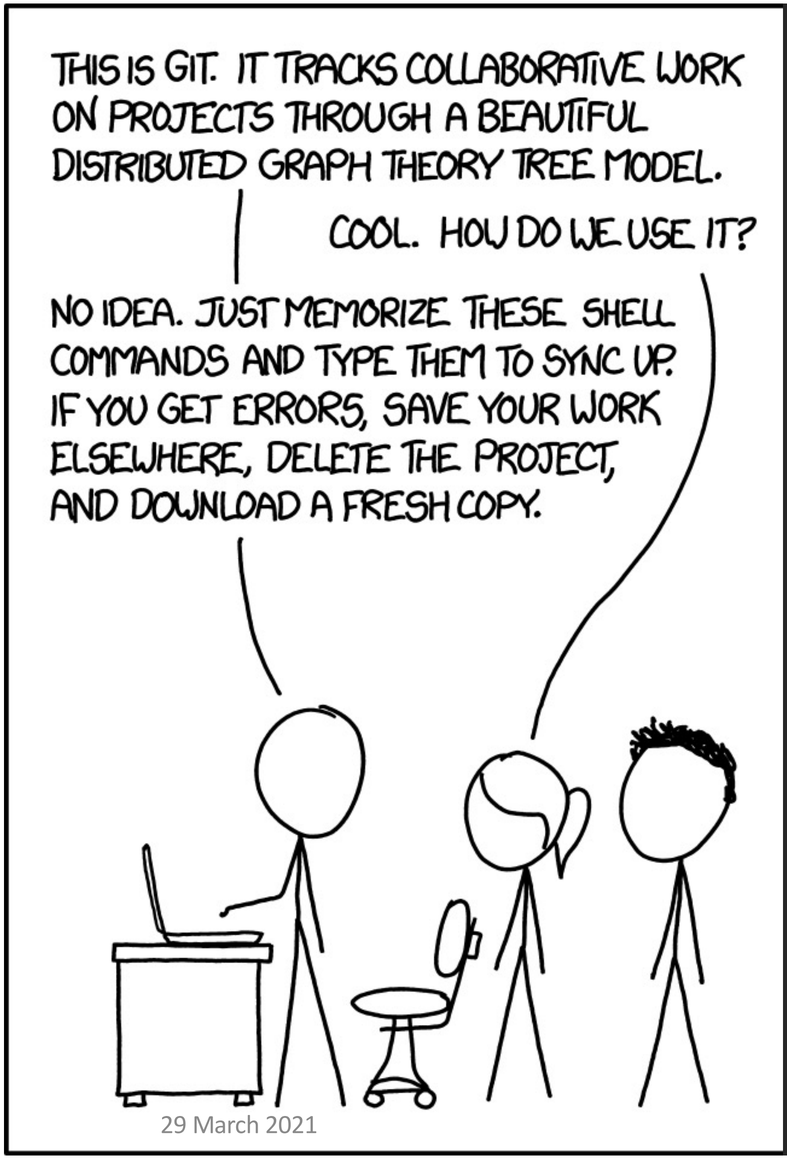

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Source Code Control using `git`

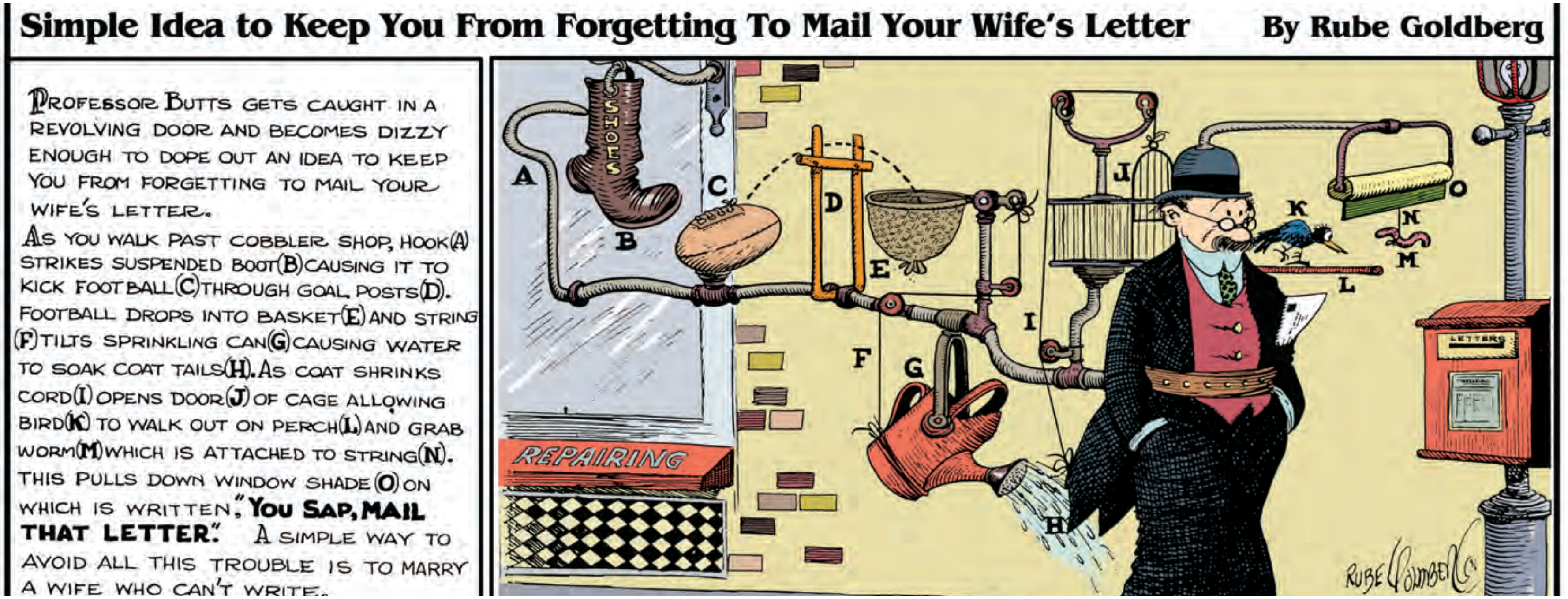
Prof. Darrell Long

CSE 13S

What is `git`?

- It is a *Source Code Control* system.
  - It tracks files and the changes to those files.
- It is a collaboration tool.
  - It allows multiple people to work independently and then merge their work.
- It is your *best friend*
  - When you lose a file, or
  - When you mess things up and want to go back.

Warning: `git` can be complicated!



Simple Idea to Keep You From Forgetting To Mail Your Wife's Letter **By Rube Goldberg**

Professor Butts gets caught in a revolving door and becomes dizzy enough to dope out an idea to keep you from forgetting to mail your wife's letter.

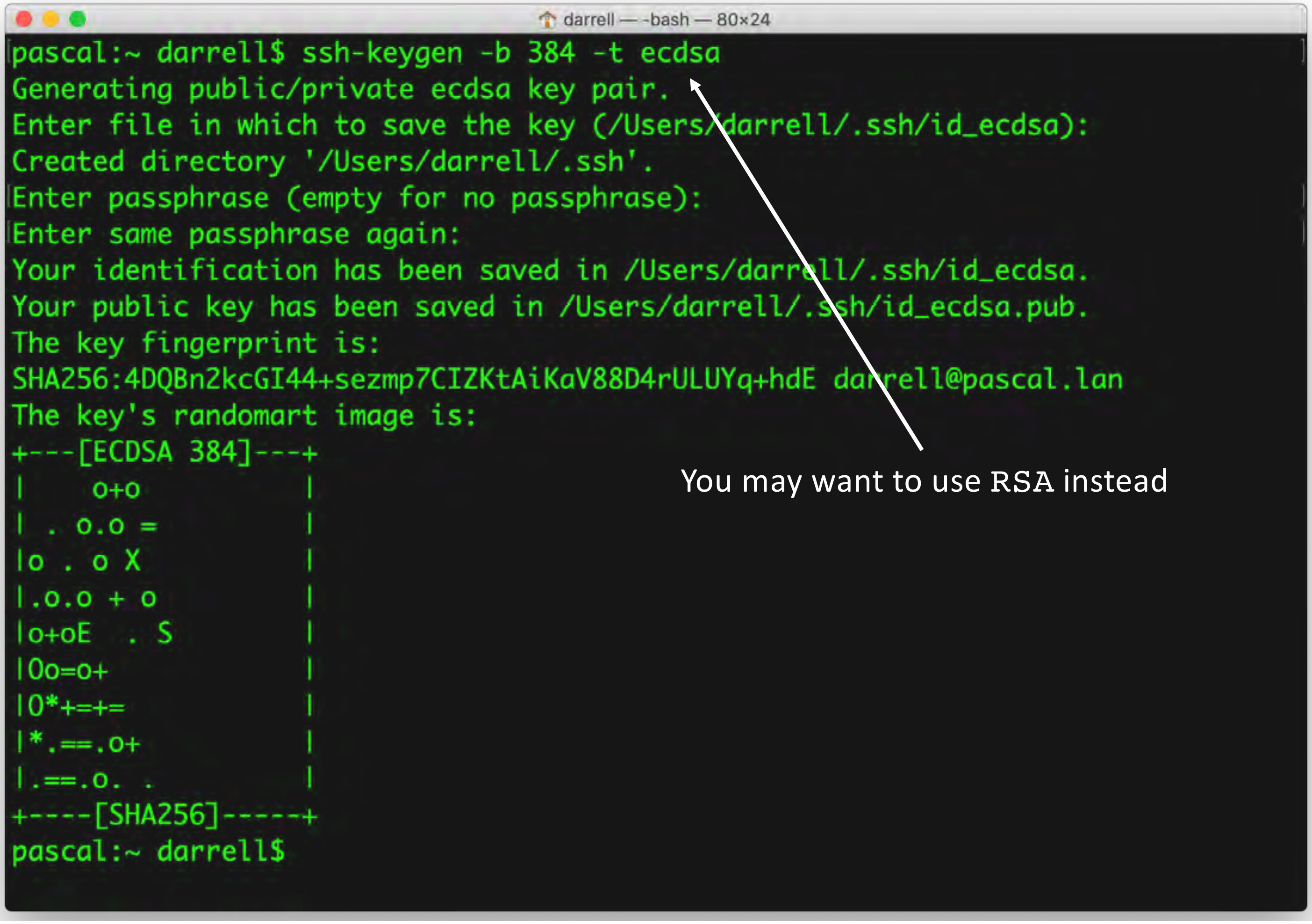
As you walk past cobbler shop, hook (A) strikes suspended boot (B) causing it to kick football (C) through goal posts (D). Football drops into basket (E) and string (F) tilts sprinkling can (G) causing water to soak coat tails (H). As coat shrinks cord (I) opens door (J) of cage allowing bird (K) to walk out on perch (L) and grab worm (M) which is attached to string (N). This pulls down window shade (O) on which is written, "**You sap, mail that letter.**" A simple way to avoid all this trouble is to marry a wife who can't write.

What is `ssh`?

- It is a family of programs that let one computer communicate with another.
  - `ssh` — Secure Shell
  - `scp` — Secure Copy
  - `sftp` — Secure FTP

- Data is compressed and encrypted before sending it over the network.
- Authentication is provided by Public Key Cryptography.

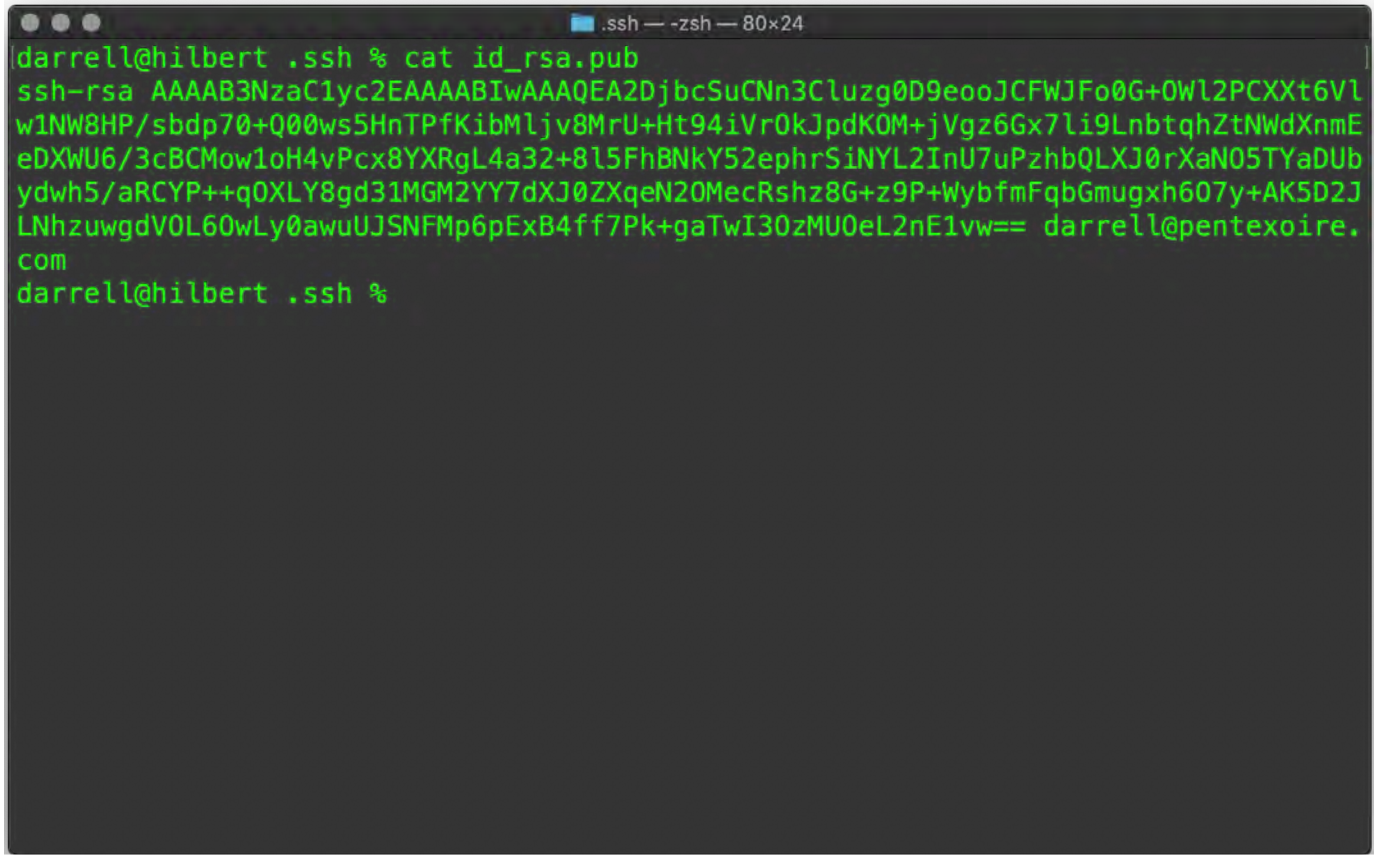
Create an `ssh` key



```
pascal:~ darrell$ ssh-keygen -b 384 -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/Users/darrell/.ssh/id_ecdsa):
Created directory '/Users/darrell/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/darrell/.ssh/id_ecdsa.
Your public key has been saved in /Users/darrell/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:4DQBn2kcGI44+sezmp7CIZKtAiKaV88D4rULUYq+hdE darrell@pascal.lan
The key's randomart image is:
+---[ECDSA 384]---+
|    o+o          |
| . o.o =         |
|o . o X          |
|.o.o + o         |
|o+oE  . S        |
|Oo=o+            |
|O*+=+=           |
|*.==.o+          |
|.==.o. .         |
+----[SHA256]-----+
pascal:~ darrell$
```

You may want to use RSA instead

ssh Public Key

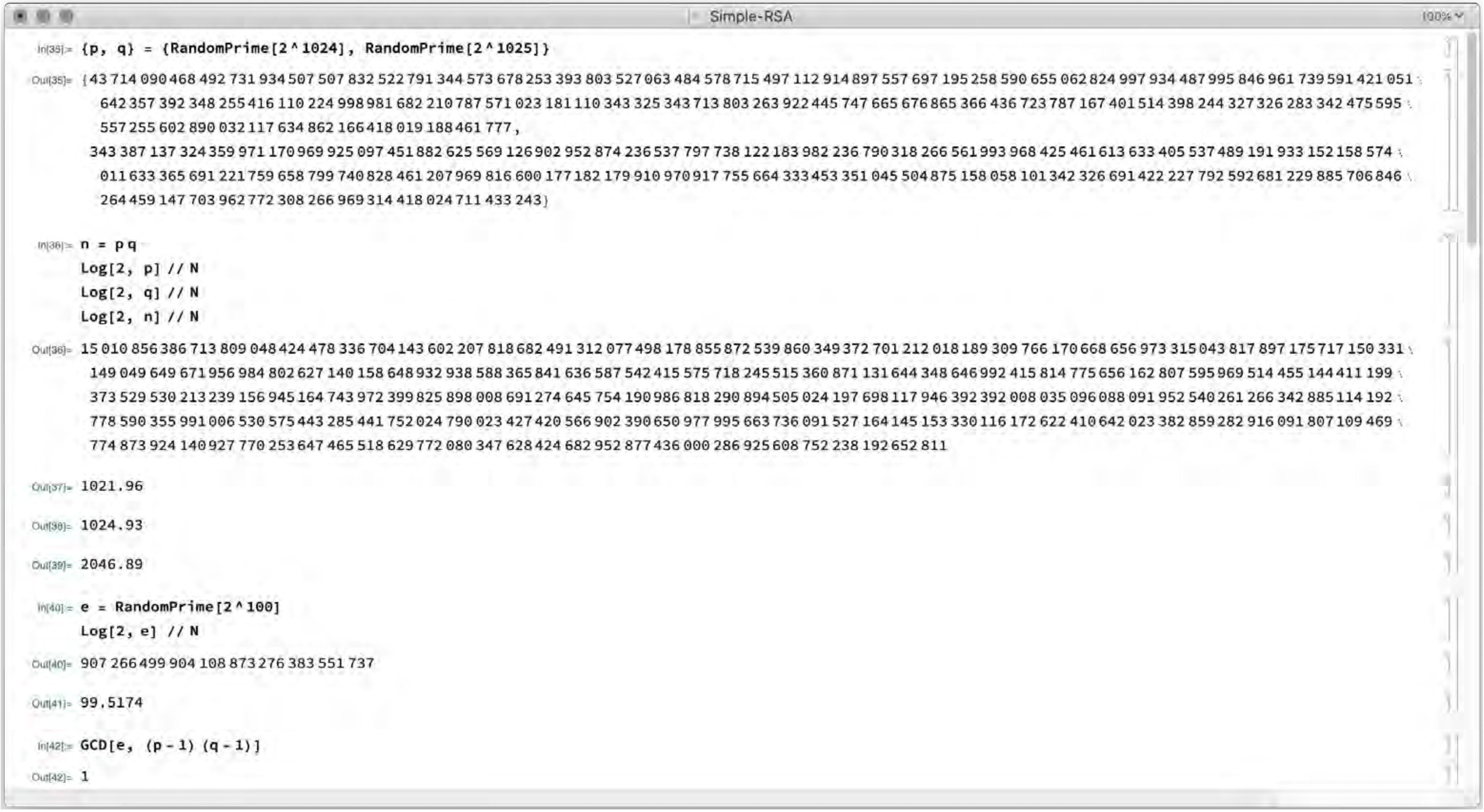


```
darrell@hilbert .ssh % cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA2DjbcSuCNn3Cluzg0D9eooJCFWJFo0G+OWl2PCXXt6Vl
w1NW8HP/sbdp70+Q00ws5HnTPfKibMljv8MrU+Ht94iVrOkJpdKOM+jVgz6Gx7li9LnbtqhZtNWdXnmE
eDXWU6/3cBCMow1oH4vPcx8YXRgL4a32+8l5FhBNkY52ephrSiNYL2InU7uPzhbQLXJ0rXaNO5TYaDUb
ydwh5/aRCYP++qOXLY8gd31MGM2YY7dXJ0ZXqeN2OMecRshz8G+z9P+WybfmFqbGmugxh6O7y+AK5D2J
LNhzuwgdVOL6OwLy0awuUJSNFMp6pExB4ff7Pk+gaTwI3OzMUOeL2nE1vw== darrell@pentexoire.
com
darrell@hilbert .ssh %
```

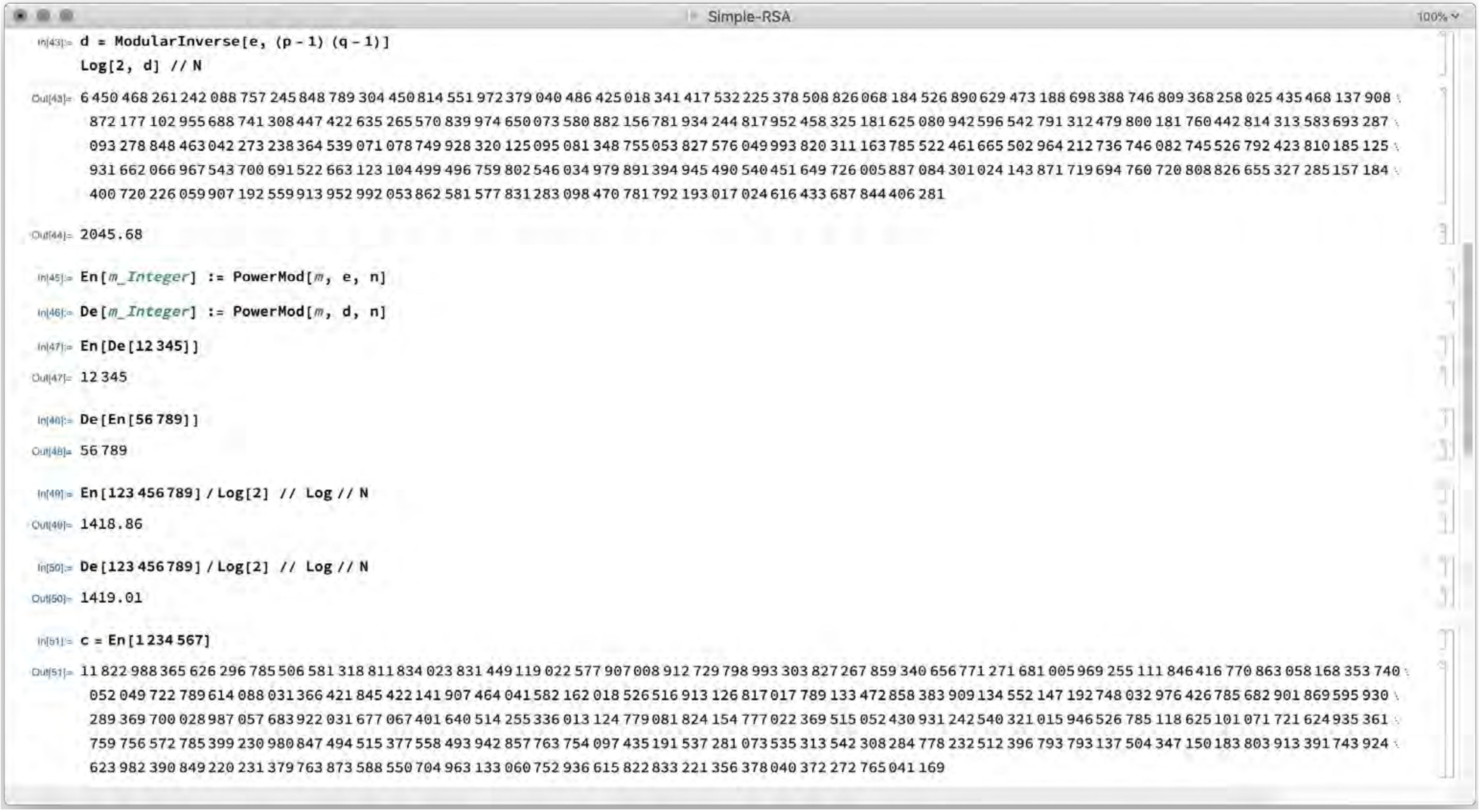
What is RSA?

- Public Key cryptography
  - There is a public key and a private key
- Choose very large primes p & q
  - $n = p \times q$
  - e is a random prime
  - $e \times d \ \equiv 1 \ (\mathrm{mod})(p-1)(q-1)$
- Encryption: $E(m) = m^e \bmod n = c$
- Decryption: $D(c) = c^d \bmod n = m$
- $E\big(D(x)\big) = D\big(E(x)\big) = x$

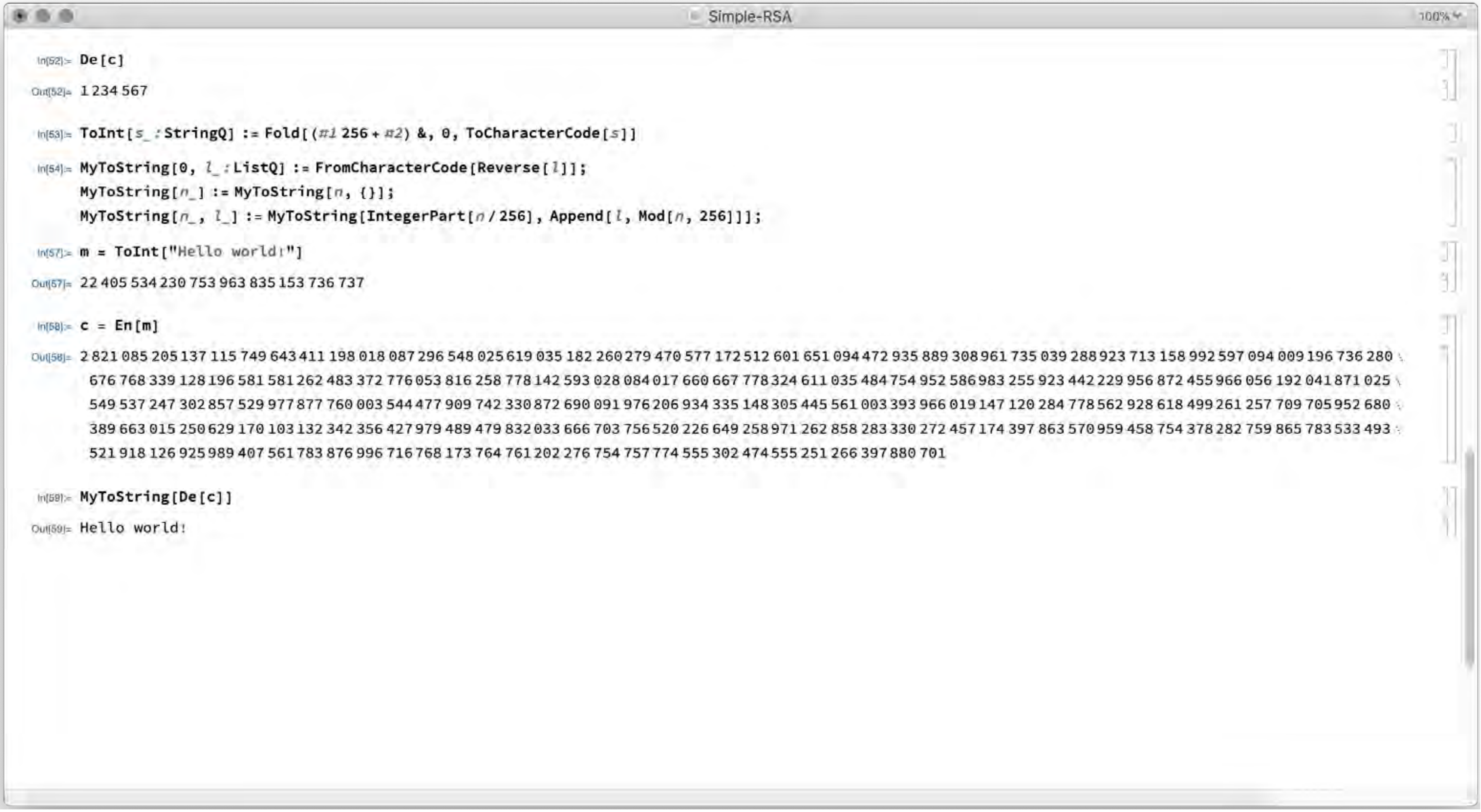
Ron Rivest



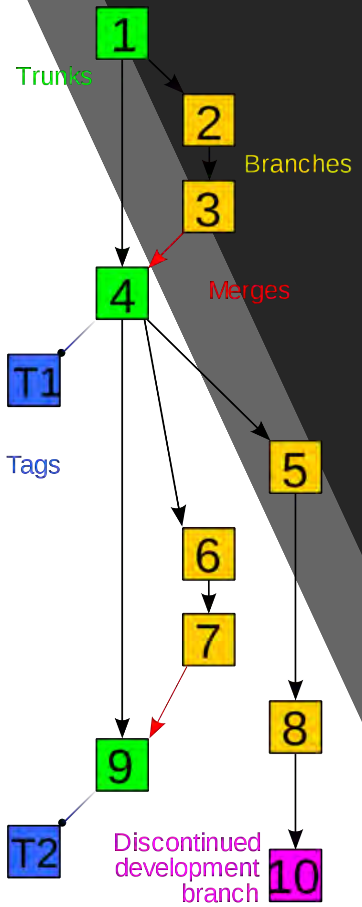
```
In[35]:= {p, q} = {RandomPrime[2^1024], RandomPrime[2^1025]}

Out[35]= {43 714 090 468 492 731 934 507 507 832 522 791 344 573 678 253 393 803 527 063 484 578 715 497 112 914 897 557 697 195 258 590 655 062 824 997 934 487 995 846 961 739 591 421 051 \
           642 357 392 348 255 416 110 224 998 981 682 210 787 571 023 181 110 343 325 343 713 803 263 922 445 747 665 676 865 366 436 723 787 167 401 514 398 244 327 326 283 342 475 595 \
           557 255 602 890 032 117 634 862 166 418 019 188 461 777,
          343 387 137 324 359 971 170 969 925 097 451 882 625 569 126 902 952 874 236 537 797 738 122 183 982 236 790 318 266 561 993 968 425 461 613 633 405 537 489 191 933 152 158 574 \
           011 633 365 691 221 759 658 799 740 828 461 207 969 816 600 177 182 179 910 970 917 755 664 333 453 351 045 504 875 158 058 101 342 326 691 422 227 792 592 681 229 885 706 846 \
           264 459 147 703 962 772 308 266 969 314 418 024 711 433 243}

In[36]:= n = p q
         Log[2, p] // N
         Log[2, q] // N
         Log[2, n] // N

Out[36]= 15 010 856 386 713 809 048 424 478 336 704 143 602 207 818 682 491 312 077 498 178 855 872 539 860 349 372 701 212 018 189 309 766 170 668 656 973 315 043 817 897 175 717 150 331 \
          149 049 649 671 956 984 802 627 140 158 648 932 938 588 365 841 636 587 542 415 575 718 245 515 360 871 131 644 348 646 992 415 814 775 656 162 807 595 969 514 455 144 411 199 \
          373 529 530 213 239 156 945 164 743 972 399 825 898 008 691 274 645 754 190 986 818 290 894 505 024 197 698 117 946 392 392 008 035 096 088 091 952 540 261 266 342 885 114 192 \
          778 590 355 991 006 530 575 443 285 441 752 024 790 023 427 420 566 902 390 650 977 995 663 736 091 527 164 145 153 330 116 172 622 410 642 023 382 859 282 916 091 807 109 469 \
          774 873 924 140 927 770 253 647 465 518 629 772 080 347 628 424 682 952 877 436 000 286 925 608 752 238 192 652 811

Out[37]= 1021.96

Out[38]= 1024.93

Out[39]= 2046.89

In[40]:= e = RandomPrime[2^100]
         Log[2, e] // N

Out[40]= 907 266 499 904 108 873 276 383 551 737

Out[41]= 99.5174

In[42]:= GCD[e, (p - 1) (q - 1)]

Out[42]= 1
```



```
In[43]:= d = ModularInverse[e, (p - 1) (q - 1)]
         Log[2, d] // N

Out[43]= 6 450 468 261 242 088 757 245 848 789 304 450 814 551 972 379 040 486 425 018 341 417 532 225 378 508 826 068 184 526 890 629 473 188 698 388 746 809 368 258 025 435 468 137 908 \
          872 177 102 955 688 741 308 447 422 635 265 570 839 974 650 073 580 882 156 781 934 244 817 952 458 325 181 625 080 942 596 542 791 312 479 800 181 760 442 814 313 583 693 287 \
          093 278 848 463 042 273 238 364 539 071 078 749 928 320 125 095 081 348 755 053 827 576 049 993 820 311 163 785 522 461 665 502 964 212 736 746 082 745 526 792 423 810 185 125 \
          931 662 066 967 543 700 691 522 663 123 104 499 496 759 802 546 034 979 891 394 945 490 540 451 649 726 005 887 084 301 024 143 871 719 694 760 720 808 826 655 327 285 157 184 \
          400 720 226 059 907 192 559 913 952 992 053 862 581 577 831 283 098 470 781 792 193 017 024 616 433 687 844 406 281

Out[44]= 2045.68

In[45]:= En[m_Integer] := PowerMod[m, e, n]

In[46]:= De[m_Integer] := PowerMod[m, d, n]

In[47]:= En[De[12 345]]

Out[47]= 12 345

In[48]:= De[En[56 789]]

Out[48]= 56 789

In[49]:= En[123 456 789] / Log[2] // Log // N

Out[49]= 1418.86

In[50]:= De[123 456 789] / Log[2] // Log // N

Out[50]= 1419.01

In[51]:= c = En[1 234 567]

Out[51]= 11 822 988 365 626 296 785 506 581 318 811 834 023 831 449 119 022 577 907 008 912 729 798 993 303 827 267 859 340 656 771 271 681 005 969 255 111 846 416 770 863 058 168 353 740 \
          052 049 722 789 614 088 031 366 421 845 422 141 907 464 041 582 162 018 526 516 913 126 817 017 789 133 472 858 383 909 134 552 147 192 748 032 976 426 785 682 901 869 595 930 \
          289 369 700 028 987 057 683 922 031 677 067 401 640 514 255 336 013 124 779 081 824 154 777 022 369 515 052 430 931 242 540 321 015 946 526 785 118 625 101 071 721 624 935 361 \
          759 756 572 785 399 230 980 847 494 515 377 558 493 942 857 763 754 097 435 191 537 281 073 535 313 542 308 284 778 232 512 396 793 793 137 504 347 150 183 803 913 391 743 924 \
          623 982 390 849 220 231 379 763 873 588 550 704 963 133 060 752 936 615 822 833 221 356 378 040 372 272 765 041 169
```



```
In[52]:= De[c]

Out[52]= 1 234 567

In[53]:= ToInt[s_?StringQ] := Fold[(#1 256 + #2) &, 0, ToCharacterCode[s]]

In[54]:= MyToString[0, l_?ListQ] := FromCharacterCode[Reverse[l]];
         MyToString[n_] := MyToString[n, {}];
         MyToString[n_, l_] := MyToString[IntegerPart[n / 256], Append[l, Mod[n, 256]]];

In[57]:= m = ToInt["Hello world!"]

Out[57]= 22 405 534 230 753 963 835 153 736 737

In[58]:= c = En[m]

Out[58]= 2 821 085 205 137 115 749 643 411 198 018 087 296 548 025 619 035 182 260 279 470 577 172 512 601 651 094 472 935 889 308 961 735 039 288 923 713 158 992 597 094 009 196 736 280 \
          676 768 339 128 196 581 581 262 483 372 776 053 816 258 778 142 593 028 084 017 660 667 778 324 611 035 484 754 952 586 983 255 923 442 229 956 872 455 966 056 192 041 871 025 \
          549 537 247 302 857 529 977 877 760 003 544 477 909 742 330 872 690 091 976 206 934 335 148 305 445 561 003 393 966 019 147 120 284 778 562 928 618 499 261 257 709 705 952 680 \
          389 663 015 250 629 170 103 132 342 356 427 979 489 479 832 033 666 703 756 520 226 649 258 971 262 858 283 330 272 457 174 397 863 570 959 458 754 378 282 759 865 783 533 493 \
          521 918 126 925 989 407 561 783 876 996 716 768 173 764 761 202 276 754 757 774 555 302 474 555 251 266 397 880 701

In[59]:= MyToString[De[c]]

Out[59]= Hello world!
```

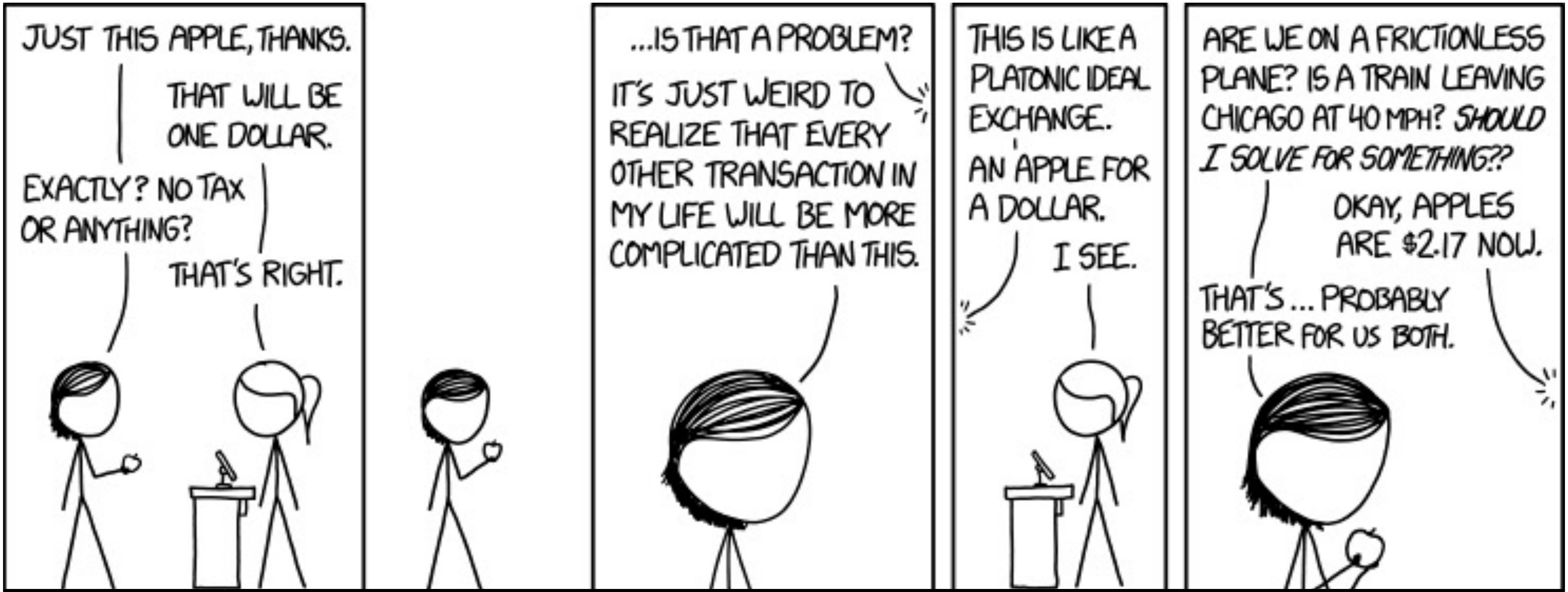


Source Code Control

- `git` is the source control system that you will use during your education.
    - It is among the most widely used.
- Keeps multiple versions of your files.
    - That means if you make a mistake you can go back to an earlier version.
- You can share your files with others in a controlled fashion, and
    - You can reconcile divergent versions of your files.
- Files are stored in a repository.
    - Stores are opaque chunks (*do not* mess with them).
    - The repository represents the history of each file.
- You have a local repository that you can synchronize with the one stored on the server.

A little more about `git`

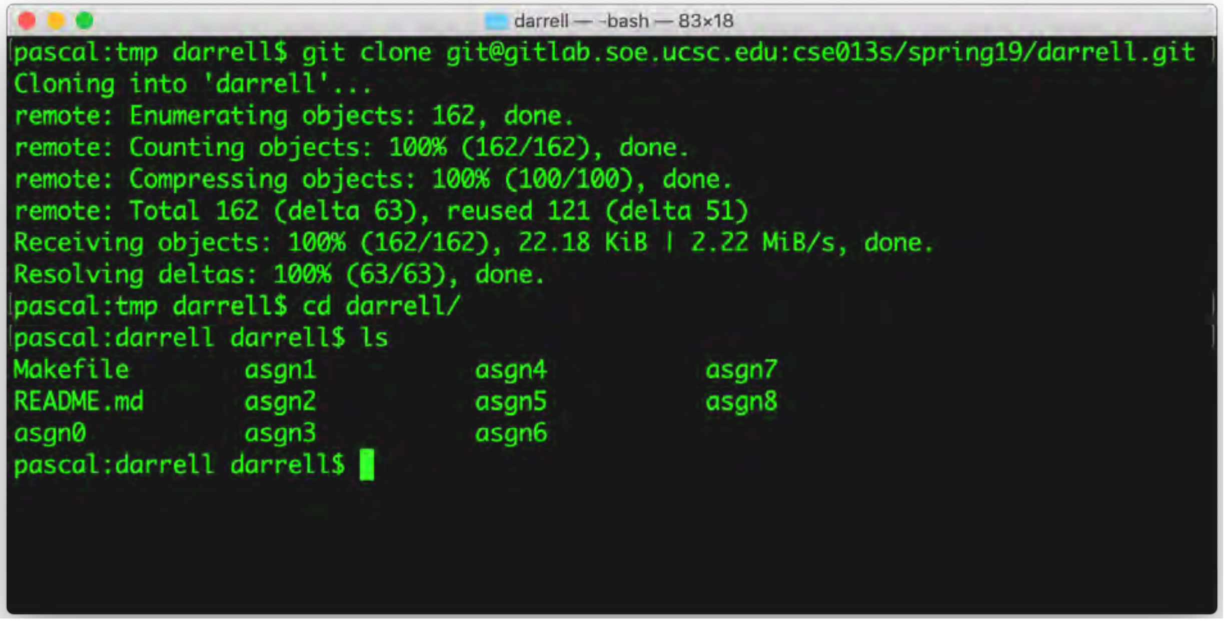
- Files are stored in a repository
  - `The .git` directory contains indexing information and a content-addressable store of file chunks.
- A set of `git` commands for manipulating the repository:
  - Use only the git commands,
  - Do not mess with the contents of `.git` (bad things will happen).
- If you do then you can clone a new copy from the server,
  - *If* you `commit` and `push` *regularly*.


JUST THIS APPLE, THANKS.
THAT WILL BE ONE DOLLAR.
EXACTLY? NO TAX OR ANYTHING?
THAT'S RIGHT.
...IS THAT A PROBLEM?
IT'S JUST WEIRD TO REALIZE THAT EVERY OTHER TRANSACTION IN MY LIFE WILL BE MORE COMPLICATED THAN THIS.
THIS IS LIKE A PLATONIC IDEAL EXCHANGE.
AN APPLE FOR A DOLLAR.
I SEE.
ARE WE ON A FRICTIONLESS PLANE? IS A TRAIN LEAVING CHICAGO AT 40 MPH? SHOULD I SOLVE FOR SOMETHING??
OKAY, APPLES ARE $2.17 NOW.
THAT'S ... PROBABLY BETTER FOR US BOTH.

Good News! Most of the time it is not complicated at all.

git clone

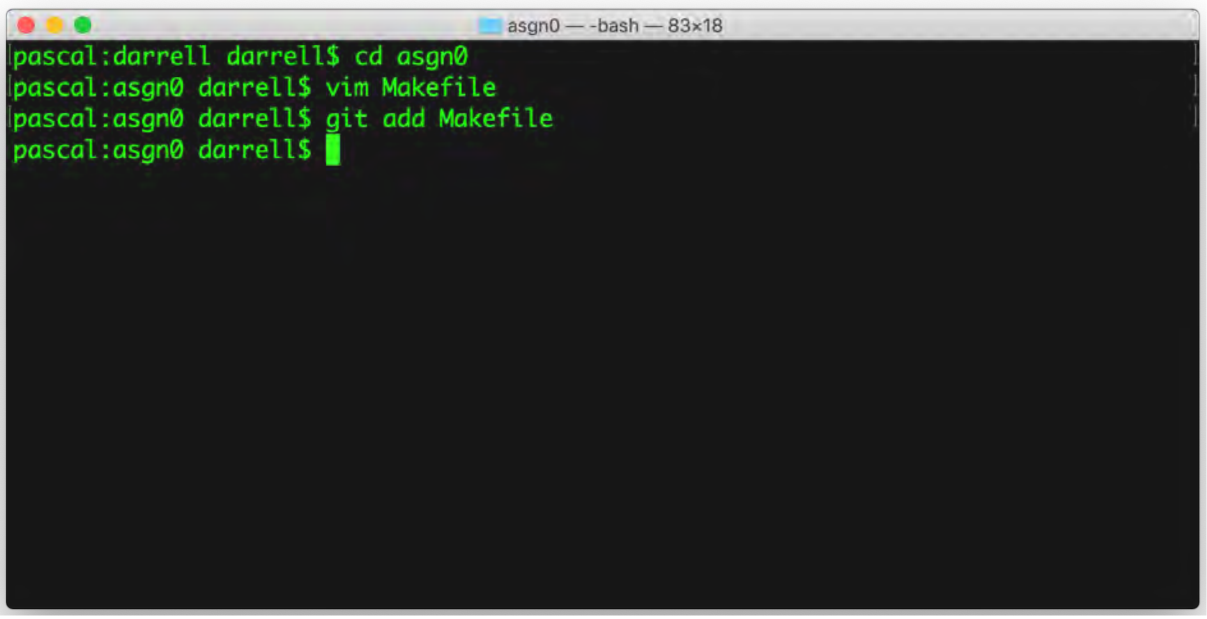
- This command makes a copy of the remote repository stored on the server.
- You can clone this directory in any place you like, as many times as you like.
- It will be prepopulated with the directories that you need for this class.



```
pascal:tmp darrell$ git clone git@gitlab.soe.ucsc.edu:cse013s/spring19/darrell.git
Cloning into 'darrell'...
remote: Enumerating objects: 162, done.
remote: Counting objects: 100% (162/162), done.
remote: Compressing objects: 100% (100/100), done.
remote: Total 162 (delta 63), reused 121 (delta 51)
Receiving objects: 100% (162/162), 22.18 KiB | 2.22 MiB/s, done.
Resolving deltas: 100% (63/63), done.
pascal:tmp darrell$ cd darrell/
pascal:darrell darrell$ ls
Makefile        asgn1           asgn4           asgn7
README.md       asgn2           asgn5           asgn8
asgn0           asgn3           asgn6
pascal:darrell darrell$ 
```

`git add`

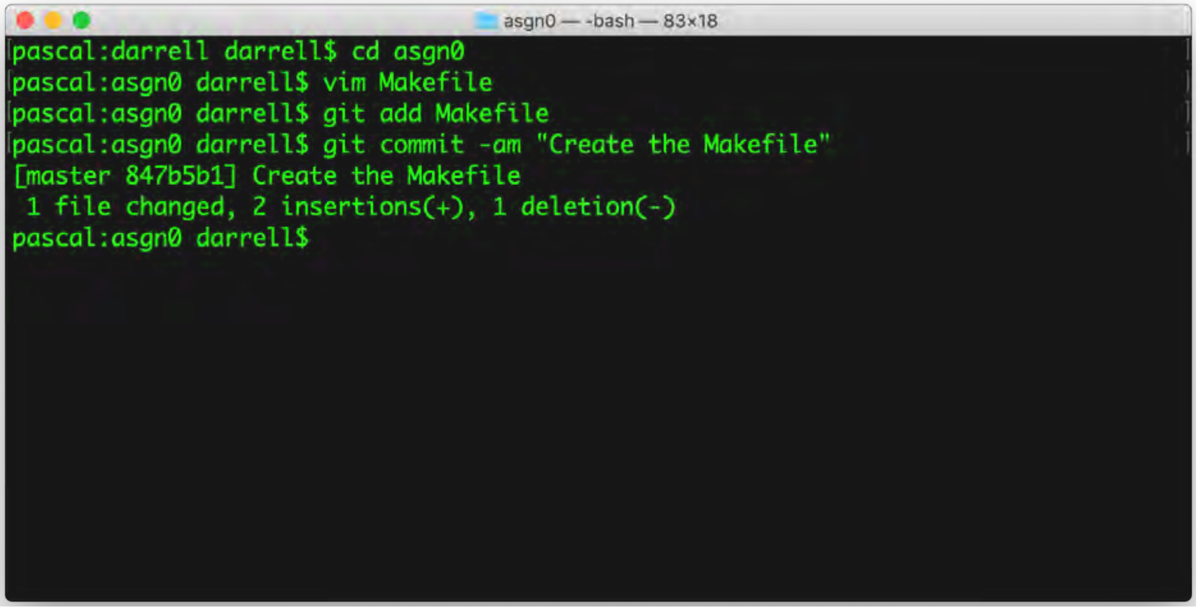
- `git add` *file$_1$ file$_2$ file$_3$ …*
  - Adds files to your repository and causes `git` to track changes to those files.



```
pascal:darrell darrell$ cd asgn0
pascal:asgn0 darrell$ vim Makefile
pascal:asgn0 darrell$ git add Makefile
pascal:asgn0 darrell$ 
```

git commit

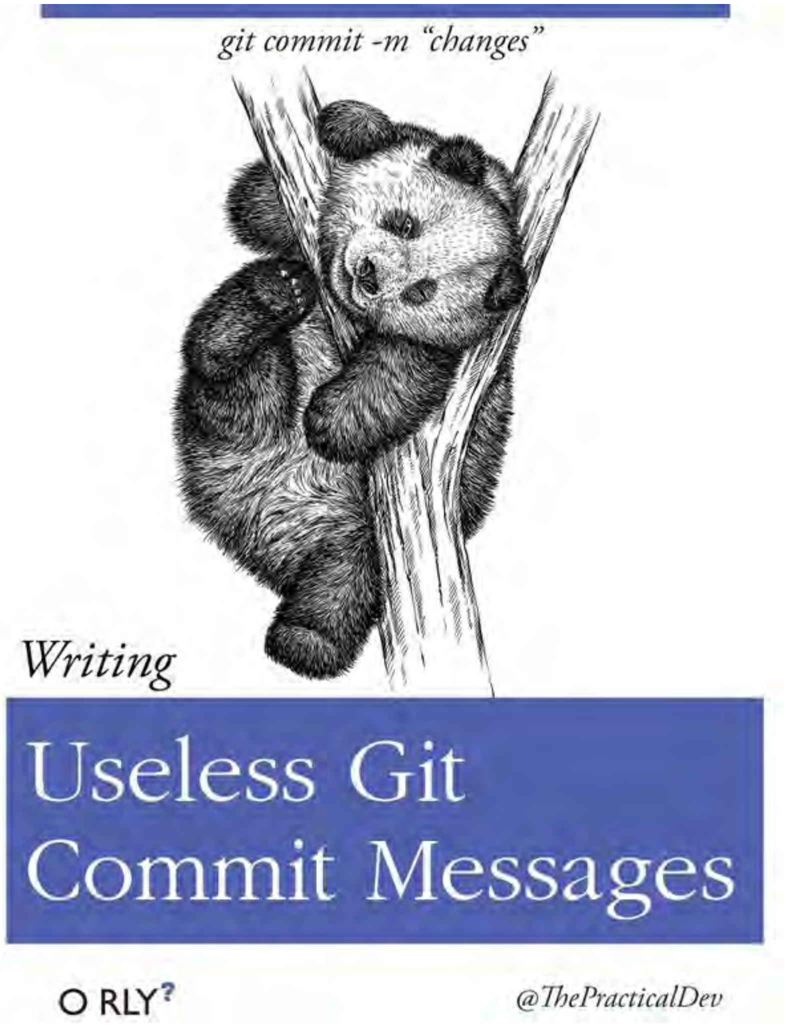
- Issuing a commit sets a point in time (a version) of your file.
- git commit –am “msg”
  - –a means all files
  - –m specifies a commit message



```
pascal:darrell darrell$ cd asgn0
pascal:asgn0 darrell$ vim Makefile
pascal:asgn0 darrell$ git add Makefile
pascal:asgn0 darrell$ git commit -am "Create the Makefile"
[master 847b5b1] Create the Makefile
 1 file changed, 2 insertions(+), 1 deletion(-)
pascal:asgn0 darrell$
```

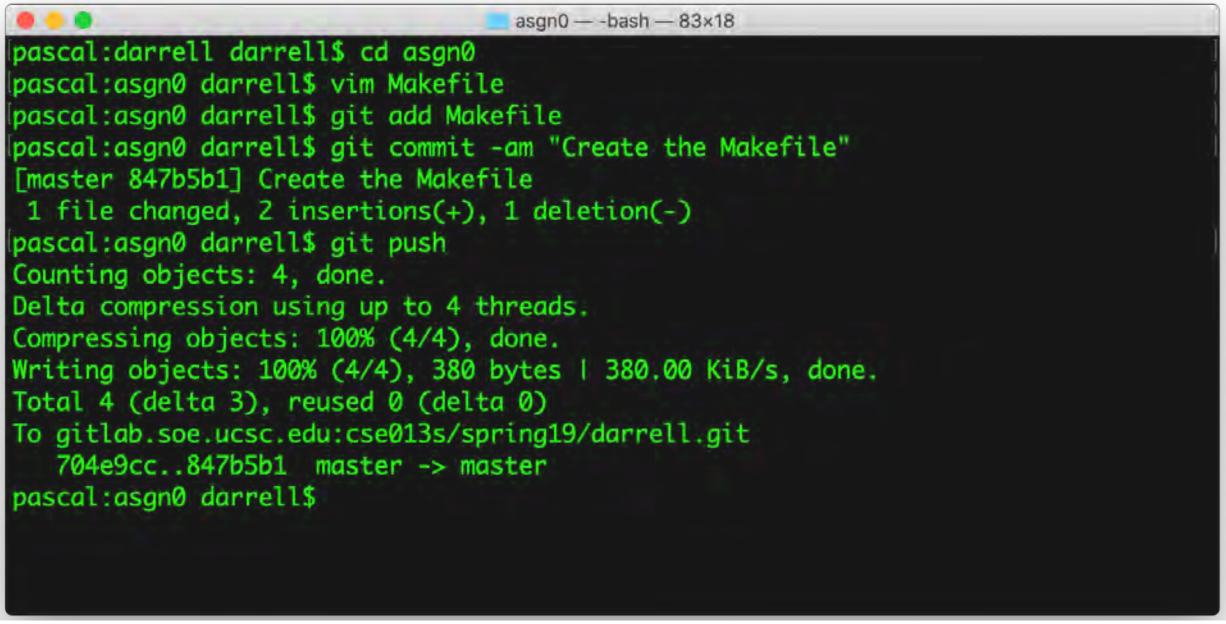
This should not be you!

- The commit message should describe what you did since the last commit.
- It will help you find a stable version when you mess things up!



git push

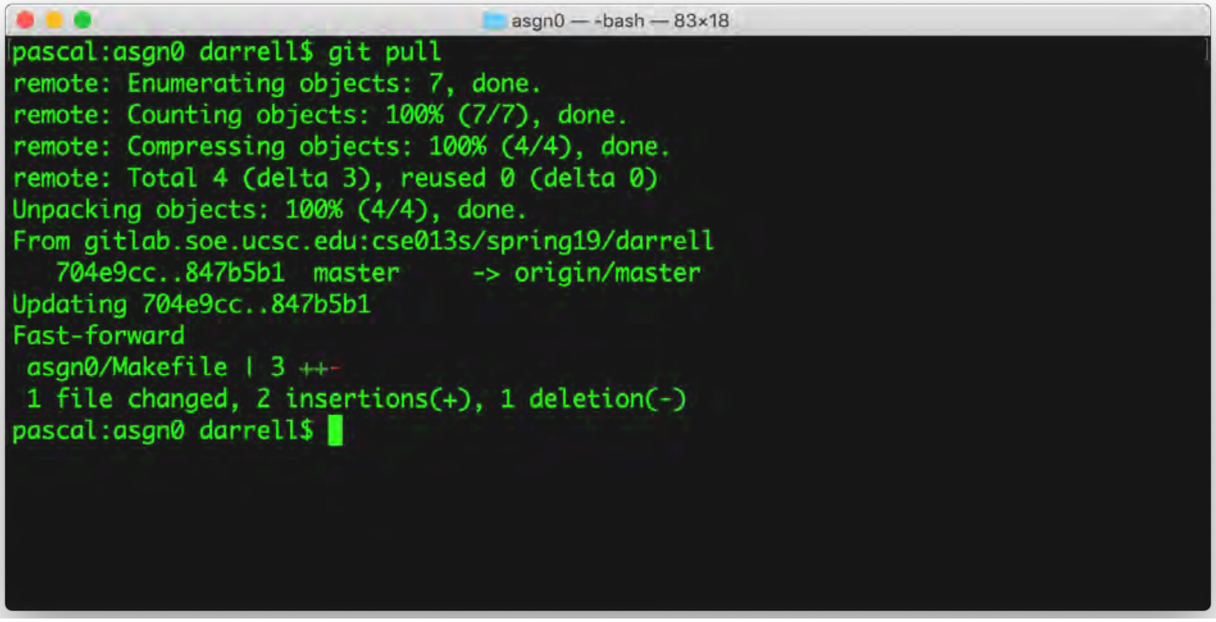
- Sends your local changes to the remote repository.
- It's a common mistake to forget to push, but
  - If you do not push then we cannot grade your code.



```
pascal:darrell darrell$ cd asgn0
pascal:asgn0 darrell$ vim Makefile
pascal:asgn0 darrell$ git add Makefile
pascal:asgn0 darrell$ git commit -am "Create the Makefile"
[master 847b5b1] Create the Makefile
 1 file changed, 2 insertions(+), 1 deletion(-)
pascal:asgn0 darrell$ git push
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 380 bytes | 380.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0)
To gitlab.soe.ucsc.edu:cse013s/spring19/darrell.git
   704e9cc..847b5b1  master -> master
pascal:asgn0 darrell$
```

git pull

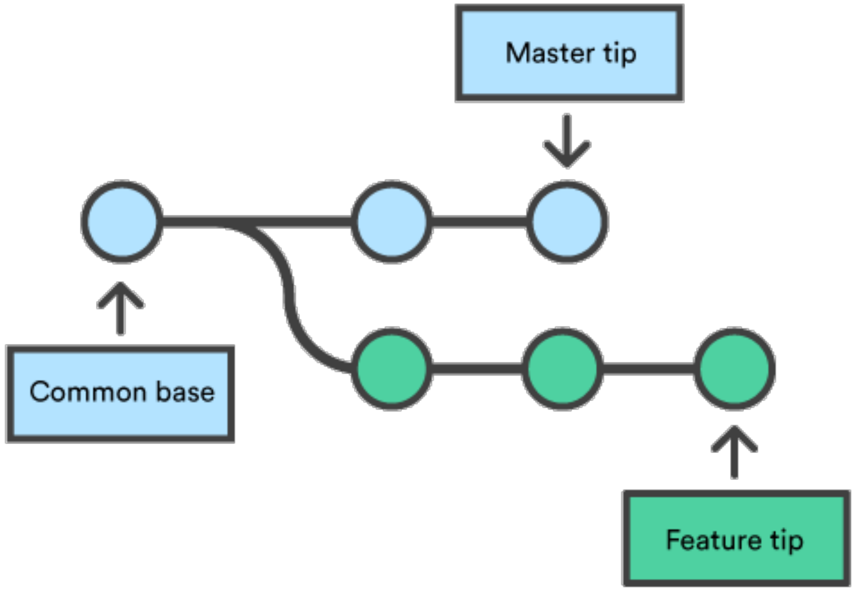
- Brings your local repository up-to-date from the remote repository.
  - Will merge in changes that appear in the other branch (perhaps made by your collaborator).
- There are many options that deal with tags, branches, and other details.
- You are strongly advised at this point in your education to *keep it simple*.



```
pascal:asgn0 darrell$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
From gitlab.soe.ucsc.edu:cse013s/spring19/darrell
   704e9cc..847b5b1  master     -> origin/master
Updating 704e9cc..847b5b1
Fast-forward
 asgn0/Makefile | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
pascal:asgn0 darrell$ 
```

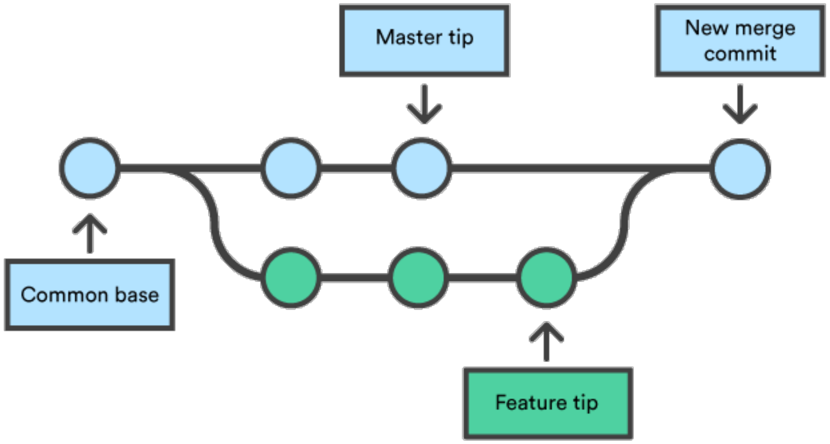
Branching

- `git` stores your repository as a sequence of commits:
  - Each commit created a new node.
- These commits form a directed acyclic graph (DAG).
- You can have a local development branch separate from the master branch.



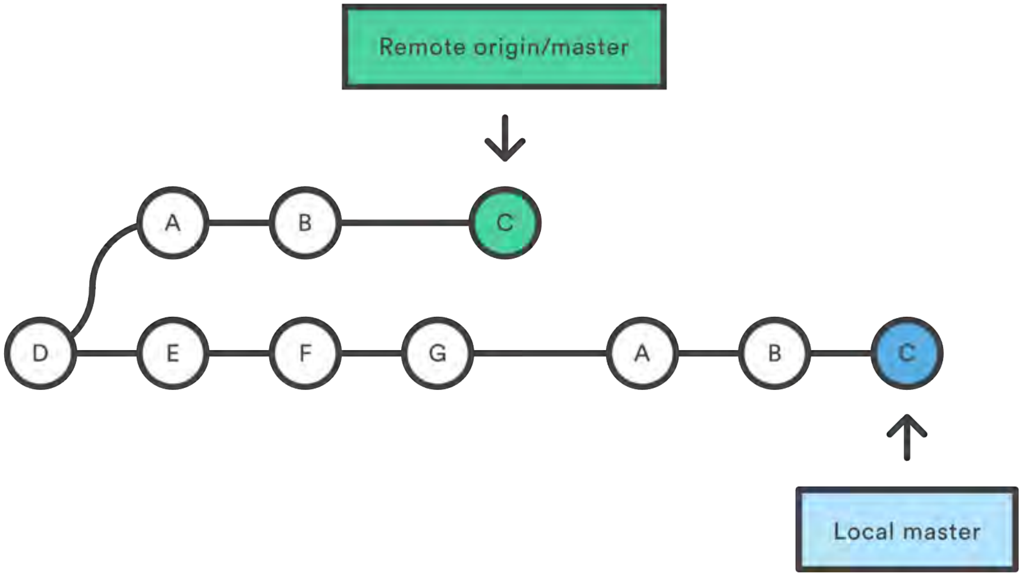
`git pull --merge`

- Merging allows you to combine two or more branches.
  - Trivial changes will be automatically applied.
- `git` is not magic, and if there are conflicts *you* will have to resolve them.



`git pull --rebase`

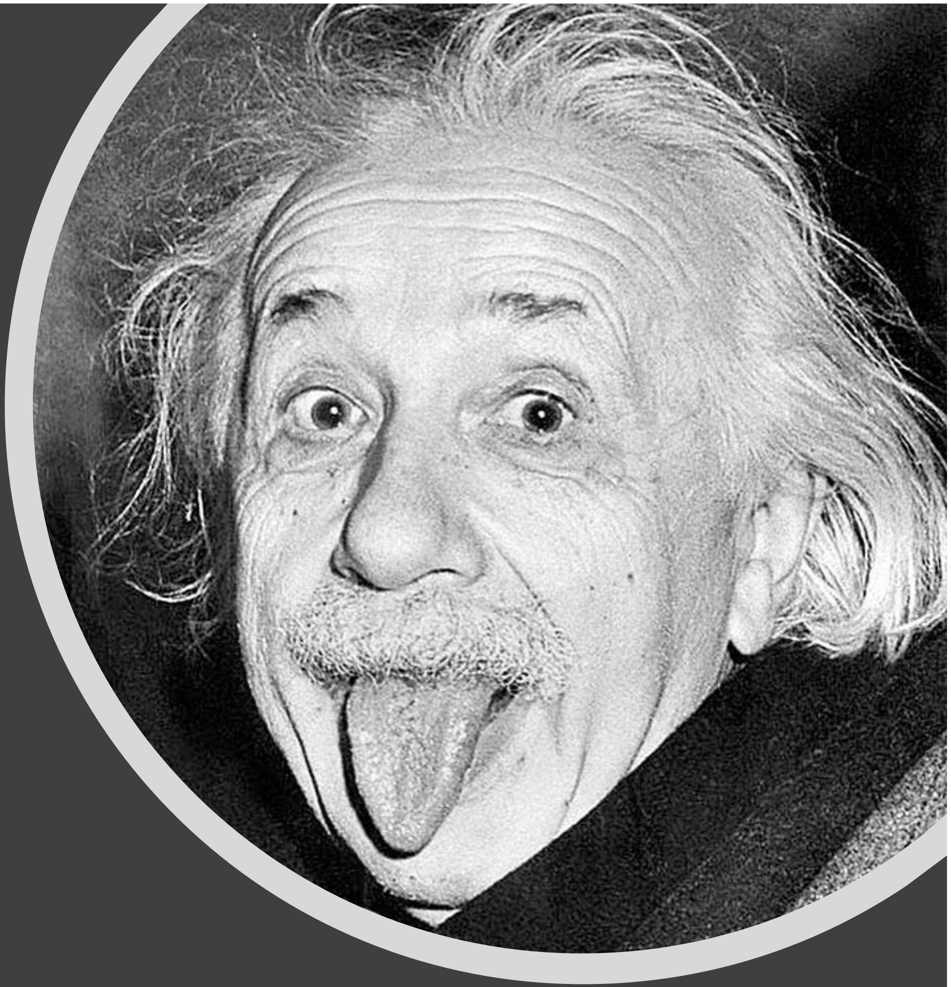
- Brings in the history from the remote repository.
- Applies them to your local respository.



Make it as simple as possible, but no simpler…

- Clone your repository on your virtual machine
- Clone your repository on unix.ucsc.edu
- Clone is as many times as you like!
- Use git push and git pull to transfer files

- Do no make things unnecessarily complicated.
  - Avoid FileZilla and programs like it.



Summary

- `git` is a powerful but complex tool.
  - If you use it correctly, it will save you from many unpleasant situations.
  - `git` tracks each commit, so you can always go back to a previous commit.
- If you only use it to submit your assignments then you are missing the point and lose all the benefits.
- Take the time to work through a tutorial:
  - https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud

It’s really not that hard

—

But it can be if you try…

The popular approach to version control
Memorizing Six Git Commands
A Practical Guide
O RLY?
@ThePracticalDev

Finally

Commit *regularly*, and

Do not forget to push!