
Software Engineering

ITAO 30210

Source Control with Git

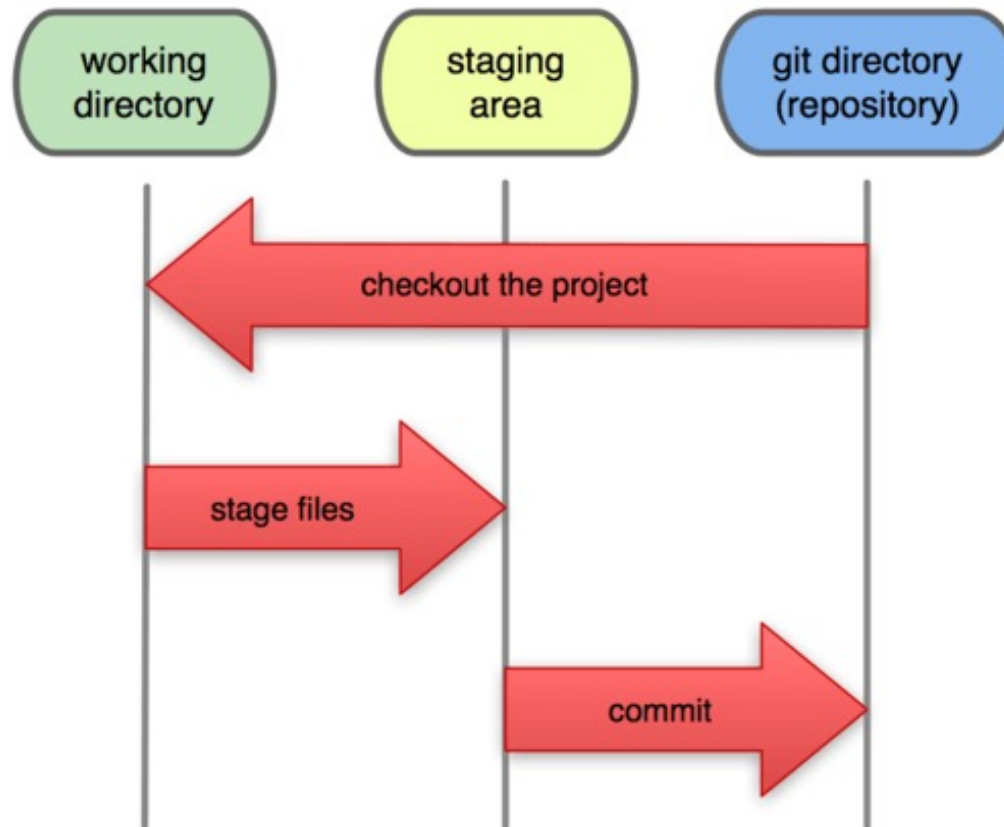




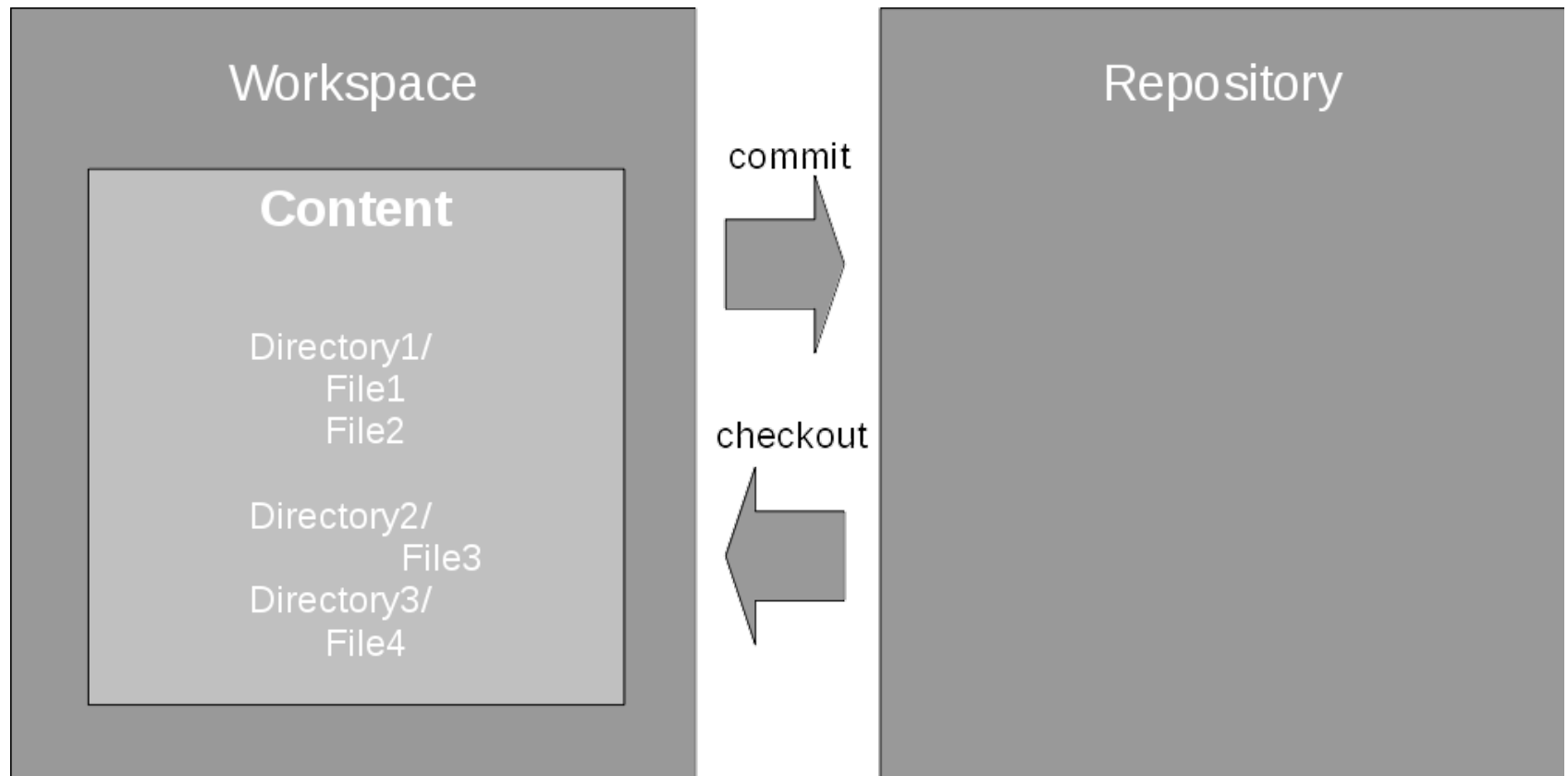
Why Source Control?

- Track every change you make to a project
 - Track every change everyone else makes to a project
 - Work on new features without breaking the current code base
 - Return to any point in time, at any time
 - Know exactly what code is running in your production environment
-

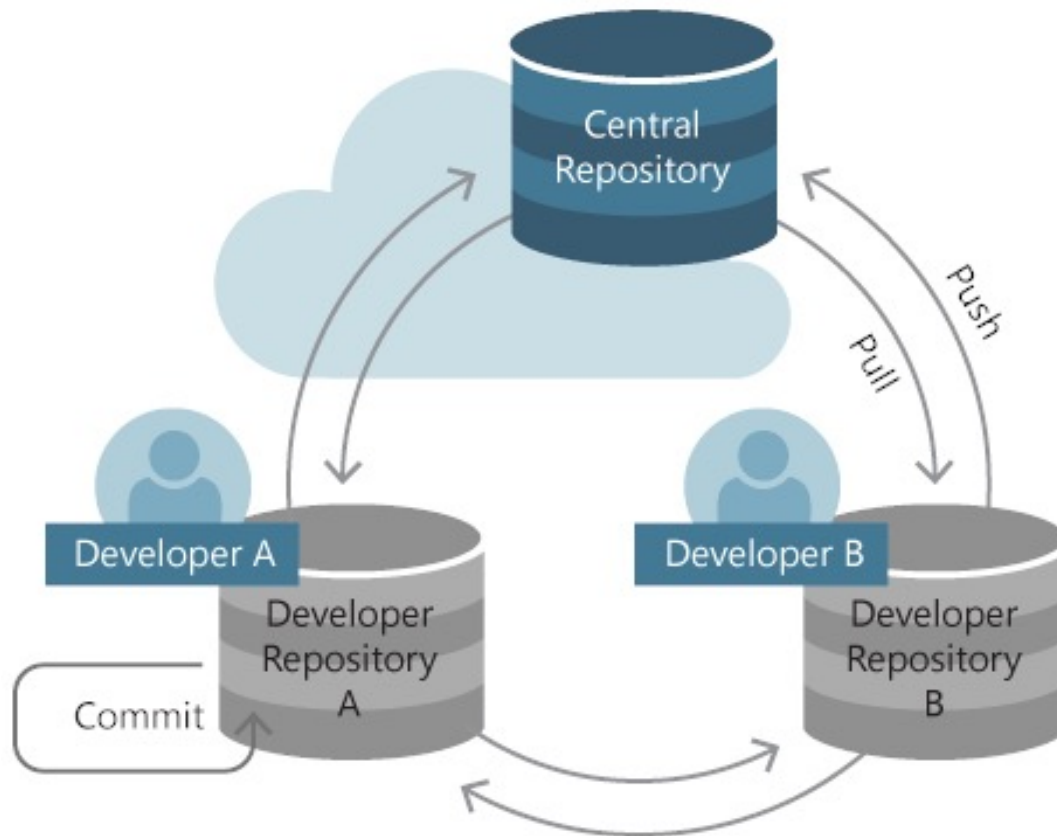
Local Operations



Git vs. Classical Source Control



Git vs. Classical Source Control



Git Terminology and Workflow

■ Git

- ❑ A CLI source control system.
- ❑ Can be installed on your computer or on a remote server for distributed work

■ GitHub

- ❑ Third party service using git but with it's own set of services and features

■ BitBucket

- ❑ Similar to GitHub, but a different service
-

Git Terminology and Workflow

- **Repository (repo)**
 - The code base and all the changes that have occurred
 - The **remote** repo is the code base that all developers share and interact with (GitHub, BitBucket)
 - **Master/Main/Production**
 - The branch that is considered the current working version of the code base. Name can vary but Master has been the convention
-

Git Terminology and Workflow

- `git init`
- `git clone`
 - Create a new repo or retrieve a repo from remote location



Git Terminology and Workflow

- `git status`
 - Lets you know if you
 - have files to be added (untracked files)
 - files not yet committed (not staged)
 - Files updated locally but not on remote (ahead of...)



Git Terminology and Workflow

- `git add`
 - When you create new files, you must tell git that there are new files



Git Terminology and Workflow

- `git commit`
 - When you add or update files, you must tell git about the changes.
 - When you commit, you must add a message.
 - Messages should explain the nature of the change and are very helpful to your future self and other developers
-

Git Terminology and Workflow

- `git push`
 - When you add files and commit changes, it is all done locally on your workstation.
 - Once you are happy with the state of the code, you push your changes to the remote repo
 - Once you do this, anyone else working on the code will be able to incorporate your changes
-

Git Terminology and Workflow

■ git push

```
(virtualenv) (base) jarp@jarp-mbp16 Python-Examples % g po main
To github.com:ITA0-40540/Python-Examples.git
 ! [rejected]          main -> main (fetch first)
error: failed to push some refs to 'git@github.com:ITA0-40540/Python-Examples.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Git Terminology and Workflow

- `git pull`
 - Once a developer has pushed their changes to the remote repo, you still will not see the changes
 - To incorporate those changes, you pull them from the remote repo to your local repo
-

Git Terminology and Workflow

■ git branch

- ❑ Most of the time when you are working on a new feature (or even a bug fix), you will want to create a branch off the current code base
 - ❑ You will name the branch something that explains what you are working on
 - image-uploads
 - duplicate-username-bug
 - ❑ Branches live in their own universe and all the previous steps (add, commit, push, pull) will be used on the branch
-

Git Terminology and Workflow

- `git merge`
 - Once the work is done on the branch, you merge the changes there into the main branch
 - Post-merge, you are free to delete the branch
 - Post-merge, you push the updated main branch
 - Others then can pull your changes
-

Git Terminology and Workflow

■ Conflicts!!!!

- ❑ If two people have worked on the same file (usually the same lines of code) sometimes git cannot merge the changes and will notify you of the conflict
 - ❑ In cases of conflicts, git will show you the differences and you must resolve the conflict.
 - ❑ Then you commit and push
-

Git Terminology and Workflow

■ git tag

- ❑ Along with branches, there are also tags
 - ❑ Tags are labels given to the code base at a specific place in time
 - ❑ Unlike branches tags generally should never change
 - ❑ Often used to label the code base at time of deploy but can be used for any reason
 - ❑ Once you create a tag, you push it to the remote repo
-

Git Terminology and Workflow

- Repeat for the life of the project



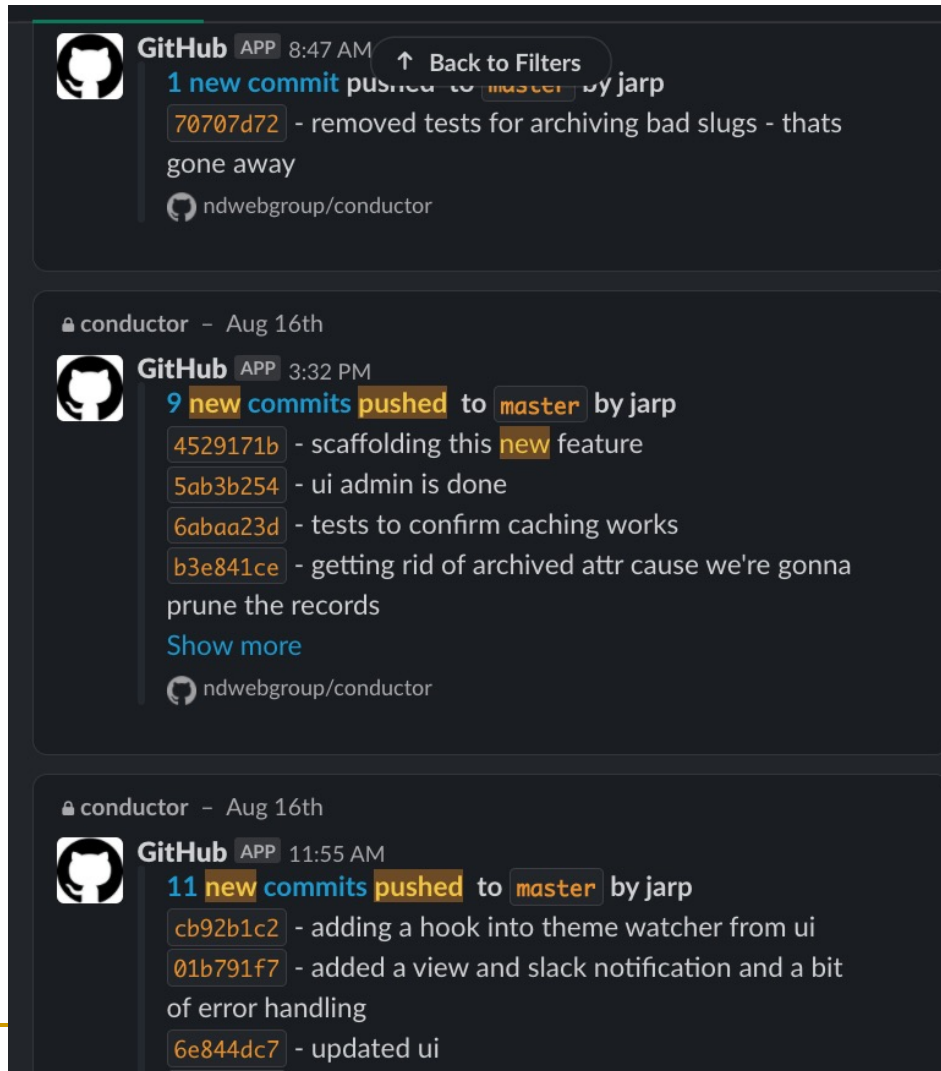
Git Recommendations and Best Practices

- Commit all the time
 - Like, every change... go ahead and commit it
 - Seriously, you can't commit too much
 - Take the time to enter useful messages
 - There is a good chance you will go back to a commit and try to understand what you were thinking last year
 - So tell your future self with a good commit message
 - Same applies to co-workers
-

Git Recommendations and Best Practices

- Make use of branching for most of your development
 - This keeps your working code base clean and bug free
 - You may be working on a new feature and a bug (easy to fix) gets reported. If you are on a branch, you can easily switch over to master, make a new branch, and fix the bug
 - master should always be deployable
-

Useful Commit Messages



The screenshot displays three GitHub notification cards for the repository `ndwebgroup/conductor`. Each card shows a commit push event by user `jarp` to the `master` branch. The first card shows 1 commit at 8:47 AM. The second card shows 9 commits at 3:32 PM. The third card shows 11 commits at 11:55 AM. Each commit message includes a hash and a description of the change.

Notification 1:
GitHub APP 8:47 AM ↑ Back to Filters
1 new commit pushed to master by jarp
70707d72 - removed tests for archiving bad slugs - thats gone away
ndwebgroup/conductor

Notification 2:
🔒 conductor - Aug 16th
GitHub APP 3:32 PM
9 new commits pushed to master by jarp
4529171b - scaffolding this new feature
5ab3b254 - ui admin is done
6abaa23d - tests to confirm caching works
b3e841ce - getting rid of archived attr cause we're gonna prune the records
Show more
ndwebgroup/conductor

Notification 3:
🔒 conductor - Aug 16th
GitHub APP 11:55 AM
11 new commits pushed to master by jarp
cb92b1c2 - adding a hook into theme watcher from ui
01b791f7 - added a view and slack notification and a bit of error handling
6e844dc7 - updated ui

Useful Commit Messages

The image displays three overlapping screenshots of GitHub commit messages as they appear in a Slack channel. Each message is from the 'GitHub (Legacy)' app and is posted by a user named 'jarp'. The messages are:

- Top-left screenshot (dated Jul 16th, 2020):** Shows a commit message: "1 new commit pushed to devel" with commit hash "1b84b253" and a message fragment "- ugh".
- Top-right screenshot (dated Aug 11th):** Shows a commit message: "to restart-themes by jarp" with a message fragment "ng dumb".
- Bottom screenshot (dated Mar 9th):** Shows a commit message: "1 new commit pushed to directory" with commit hash "4f7665d2" and a message: "wow... forgot to save files before committing the changes. it's been a good day".

Each screenshot also shows the channel name "conductor" and the GitHub logo.

Git Recommendations and Best Practices

■ Pull Requests

- ❑ Not part of git the system, but introduced by GitHub and has become a standard feature of most git based systems
 - ❑ Allows you to:
 - push all the changes in a branch
 - Comment on the changes
 - Create a workflow for feedback and approvals
-

Git Recommendations and Best Practices

■ .gitignore

- ❑ Tells git which files should not be added to repo
 - Workstation files like .DS_Store on macs
 - Files with sensitive information and passwords
 - ❑ Uses patterns
 - Exclude specific files
 - Exclude directories
 - Exclude file extensions
 - ❑ There are templates available
-

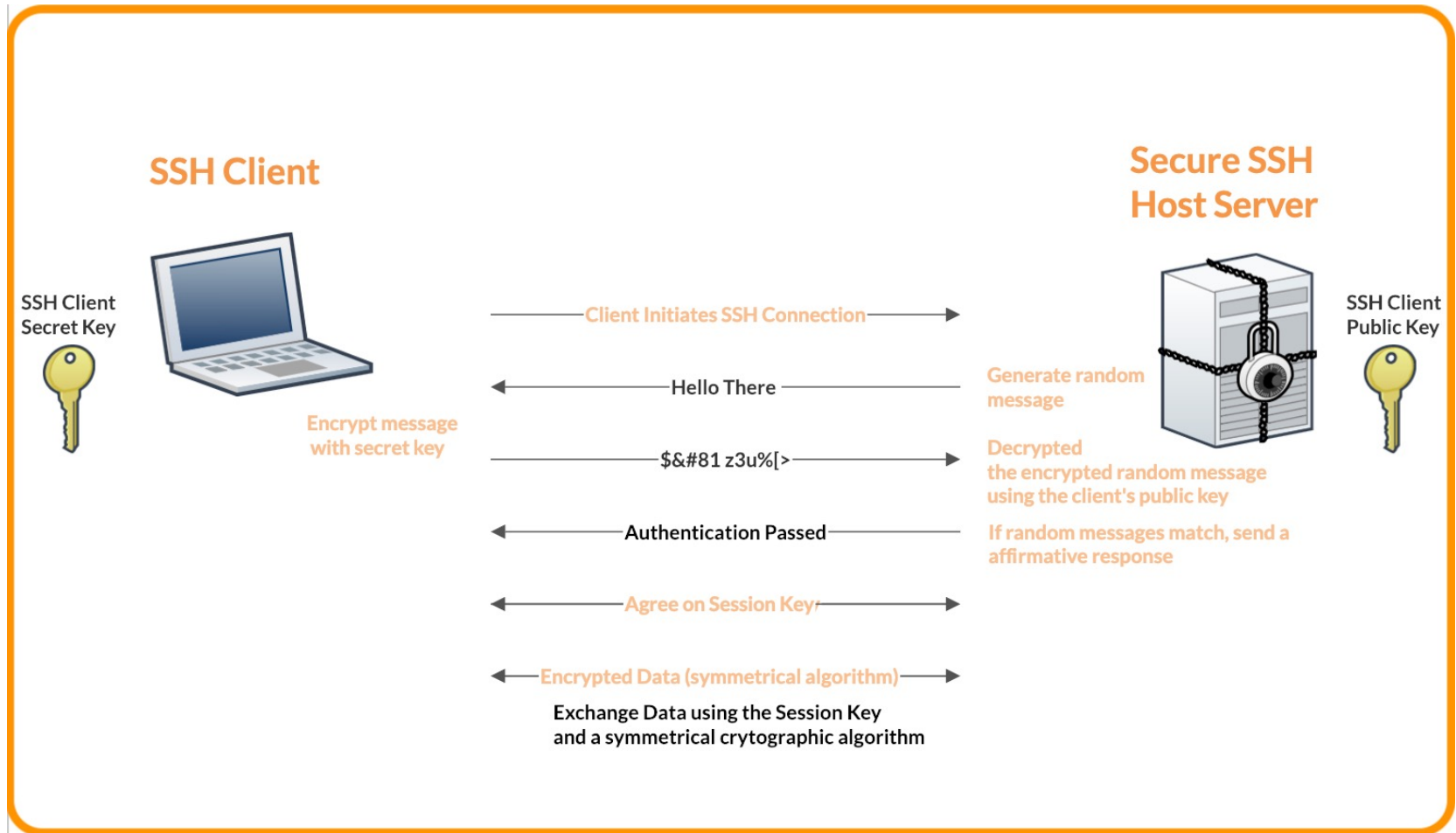
SSH Commands

- ❑ Secure Shell is a network communication protocol that enables two computers to communicate. Similar to HTTP (web browsers)
- ❑ Common uses of SSH
 - Server Administration
 - Code deployments to servers
 - GitHub and other Server based services
- ❑ Syntax:
 - `ssh user@ip-address (ssh deploy@192.345.45.8)`
 - `git@github.com:jarp/Farkle-Cli-40540.git`

SSH Commands

- ❑ Communication between servers can be “key based” or login based
 - ❑ Create a Key and Secret pair
 - ❑ Let the target server know your “public key”
 - ❑ Many Uses:
 - SSH into servers
 - Connect to git remote servers (GitHub)
-

SSH Commands



Setting up SSH with Key Gen

- Enter:
`ssh-keygen`
 - Leave password blank (maybe not?)
 - Creates two files in a `~/.ssh` directory:
 - `~/.ssh/id_rsa`
 - `~/.ssh/id_rsa.pub`
-

Setting up SSH with Key Gen

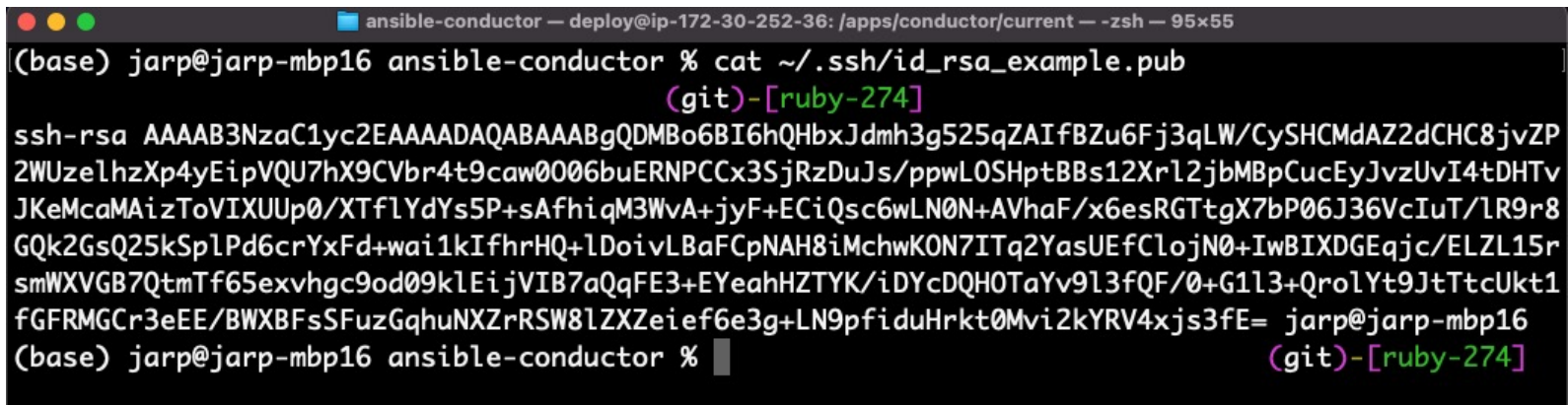
```
Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\jarp> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\jarp/.ssh/id_rsa):
Created directory 'C:\Users\jarp/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\jarp/.ssh/id_rsa.
Your public key has been saved in C:\Users\jarp/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:9+aiq2eWzRwR6f/lgatmj0QoU/HcF6ApJDVIKczPkok adnd\jarp@MCOB004-00-W19
The key's randomart image is:
+---[RSA 2048]-----+
|
| o ..+o .
| + + .. o.
| . * o .o..
| E + o o oo .
| . S oo o
| . =.o.o ..
| . +=o+.o.
| o .o=* ..
| +=oooo
+---[SHA256]-----+
(base) PS C:\Users\jarp> ls .ssh

Directory: C:\Users\jarp\.ssh

Mode                LastWriteTime         Length Name
----                -
-a----             8/24/2021   3:31 PM         1679 id_rsa
-a----             8/24/2021   3:31 PM          407 id_rsa.pub
```

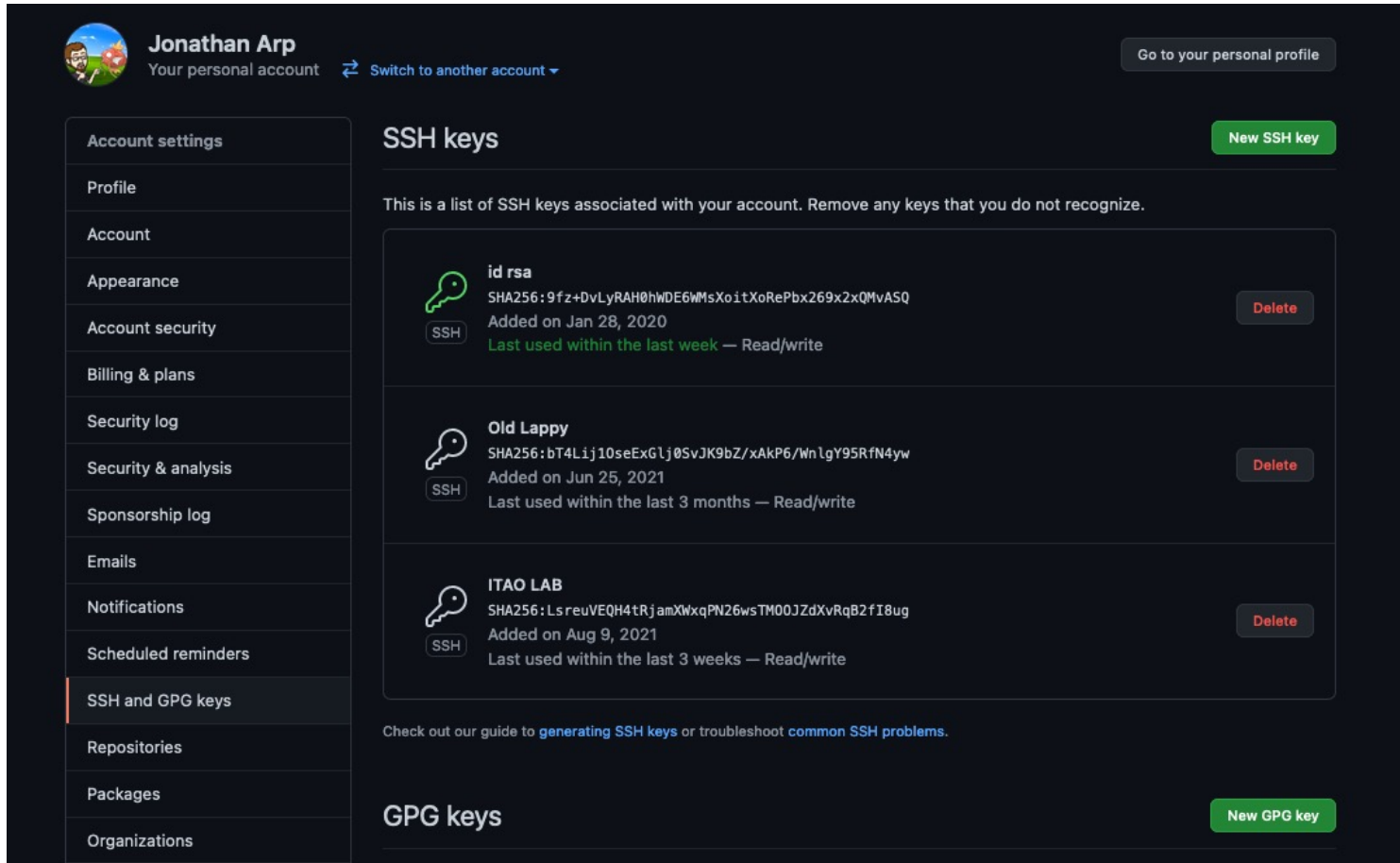
Setting up SSH with Key Gen

```
cat ~/.ssh/id_rsa_example.pub
```



```
ansible-conductor — deploy@ip-172-30-252-36: /apps/conductor/current — zsh — 95x55
(base) jarp@jarp-mbp16 ansible-conductor % cat ~/.ssh/id_rsa_example.pub
(git)-[ruby-274]
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDMBo6BI6hQHbxJdmh3g525qZAIfBZu6Fj3qLW/CySHCMdAZ2dCHC8jvZP
2WUzelhzXp4yEipVQU7hX9CVbr4t9caw0006buERNPCCx3SjRzDuJs/ppwL0SHptBBs12Xr12jbMBpCucEyJvzUvI4tDHTv
JKeMcaMAizToVIXUUp0/XTfLYdYs5P+sAfhiqM3WvA+jyF+ECiQsc6wLN0N+AVhaF/x6esRGTtgX7bP06J36VcIuT/LR9r8
GQk2GsQ25kSplPd6crYxFd+wai1kIfhrHQ+lDoivLBaFCpNAH8iMchwKON7ITq2YasUEfClojN0+IwBIXDGEqjc/ELZL15r
smWXVGB7Qtmtf65exvhgc9od09klEijVIB7aQqFE3+EYeahHZTYK/iDYcDQH0TaYv9l3fQF/0+G1l3+QroLYt9JtTtcUkt1
fGFRMGCr3eEE/BWXBFSfuzGqhuNXZrRSW8lZXZeief6e3g+LN9pfiduHrkt0Mvi2kYRV4xjs3fE= jarp@jarp-mbp16
(base) jarp@jarp-mbp16 ansible-conductor % (git)-[ruby-274]
```

Using Key Pairs with GitHub



The screenshot shows the GitHub account settings page for Jonathan Arp. The left sidebar contains a list of settings: Account settings, Profile, Account, Appearance, Account security, Billing & plans, Security log, Security & analysis, Sponsorship log, Emails, Notifications, Scheduled reminders, SSH and GPG keys (highlighted), Repositories, Packages, and Organizations. The main content area is titled 'SSH keys' and includes a 'New SSH key' button. Below the title is a list of three SSH keys, each with a key icon, name, SHA256 fingerprint, addition date, and last used date. Each key has a 'Delete' button. At the bottom of the SSH keys section, there is a link to a guide on generating SSH keys and troubleshooting common problems. Below this is the 'GPG keys' section, which includes a 'New GPG key' button.

Jonathan Arp
Your personal account [Switch to another account](#) Go to your personal profile

SSH keys New SSH key

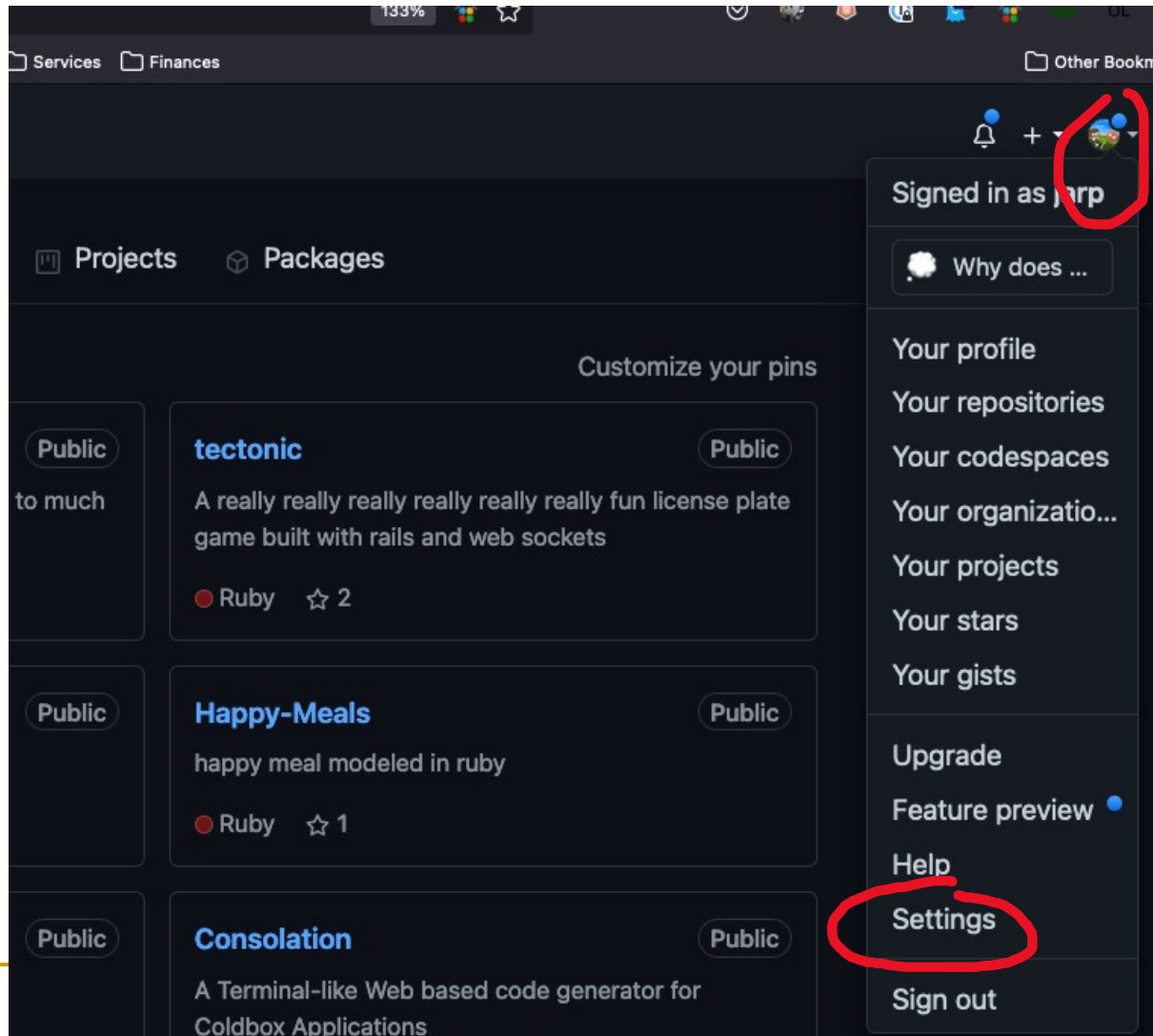
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Name	SHA256 Fingerprint	Added	Last Used	Permissions	Action
id rsa	9fz+DvLyRAH0hwDE6wMsXoitXoRePbx269x2xQMVASQ	Jan 28, 2020	within the last week	Read/write	Delete
Old Lappy	bT4Lij10seExGlj0SvJK9bZ/xAkP6/WnLgY95RfN4yw	Jun 25, 2021	within the last 3 months	Read/write	Delete
ITAO LAB	Ls reuVEQH4tRjamXwxqPN26wsTM00JZdXvRqB2fI8ug	Aug 9, 2021	within the last 3 weeks	Read/write	Delete

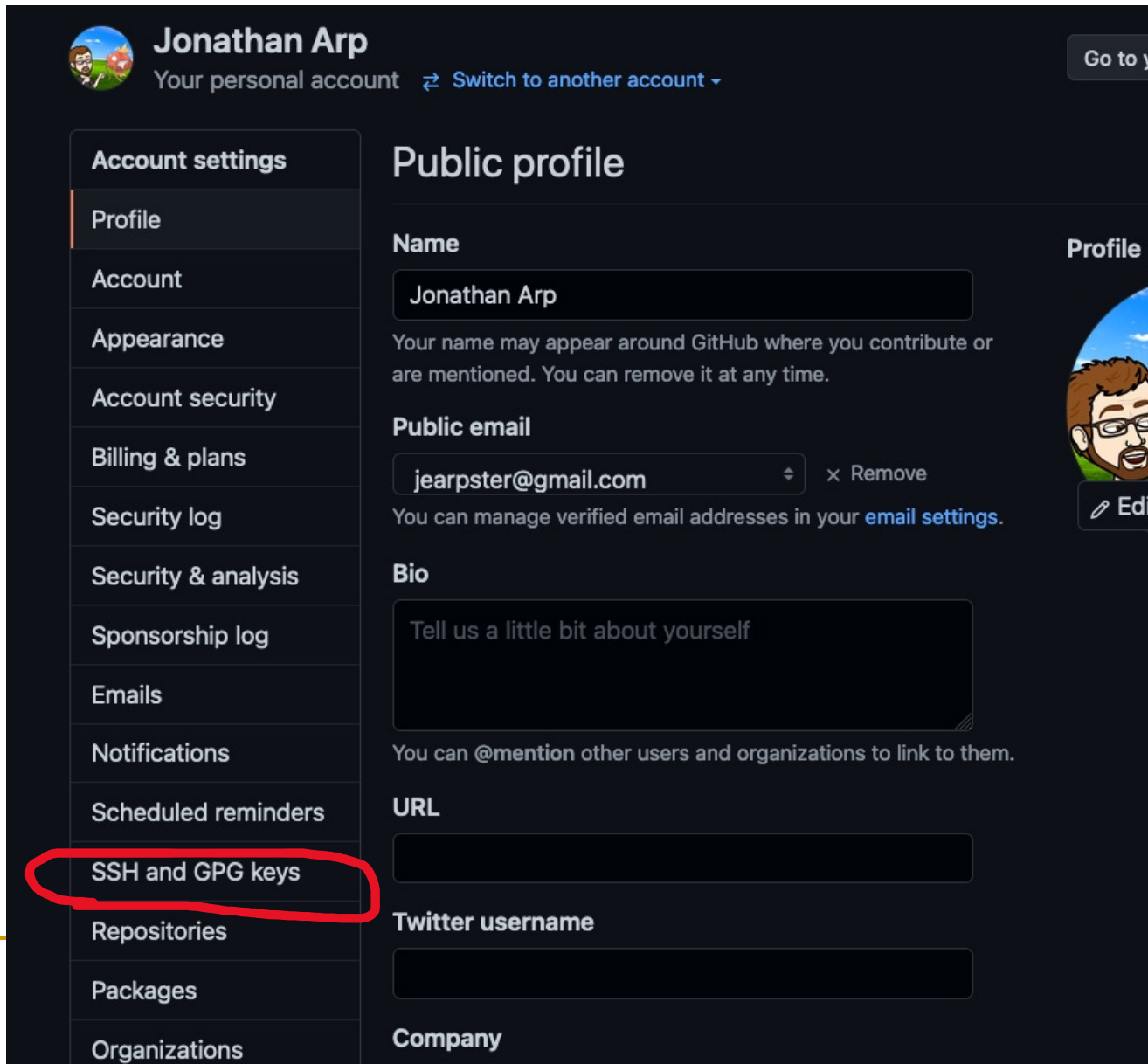
[Check out our guide to generating SSH keys](#) or [troubleshoot common SSH problems](#).

GPG keys New GPG key

Using Key Pairs with GitHub



Using Key Pairs with GitHub



Jonathan Arp
Your personal account [Switch to another account](#)

Account settings

- Profile
- Account
- Appearance
- Account security
- Billing & plans
- Security log
- Security & analysis
- Sponsorship log
- Emails
- Notifications
- Scheduled reminders
- SSH and GPG keys**
- Repositories
- Packages
- Organizations

Public profile

Name
Jonathan Arp
Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

Public email
jearpster@gmail.com [Remove](#)
You can manage verified email addresses in your [email settings](#).

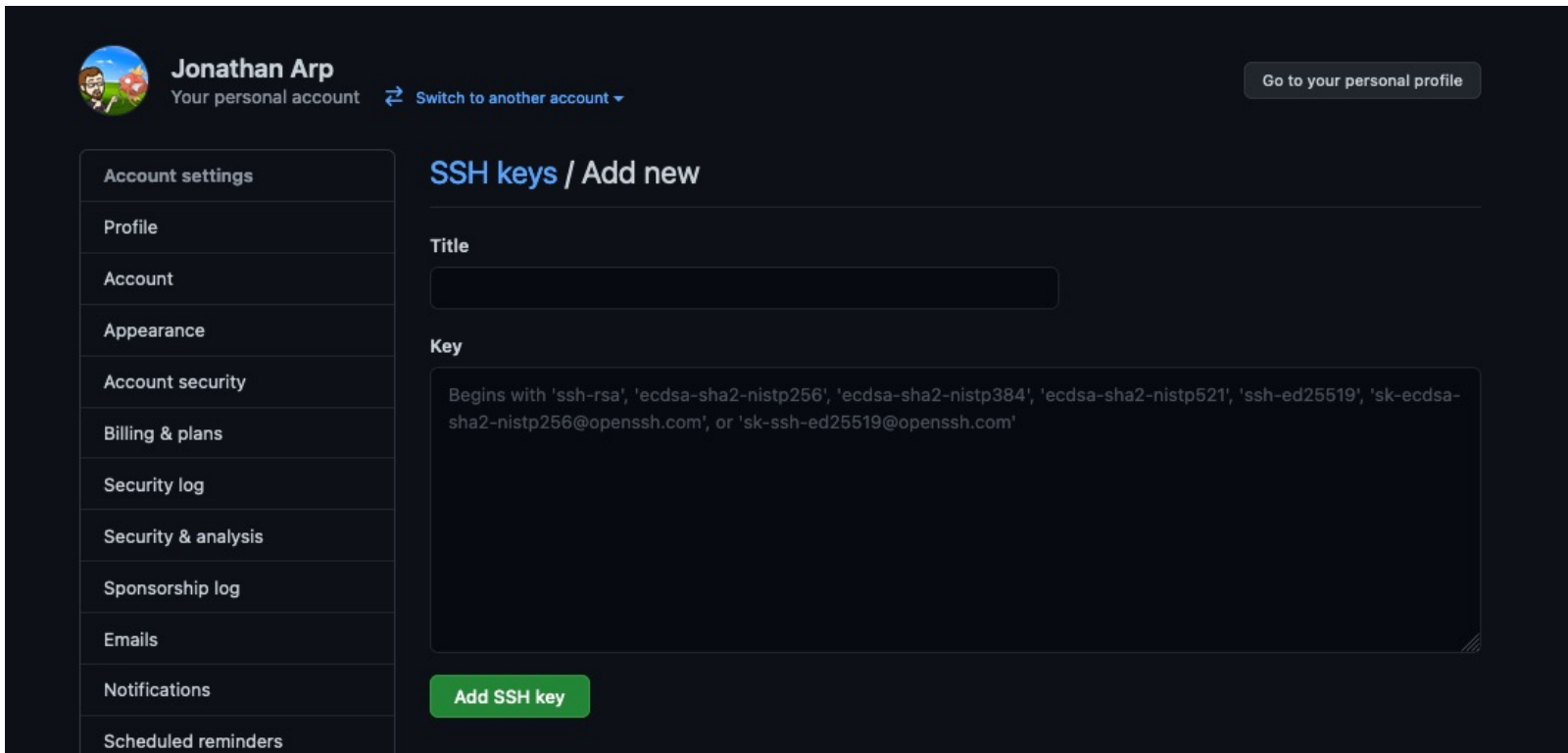
Bio
Tell us a little bit about yourself
You can @mention other users and organizations to link to them.

URL

Twitter username

Company

Using Key Pairs with GitHub



The screenshot shows the GitHub account settings page for Jonathan Arp. The user's name and profile picture are visible at the top left. A navigation menu on the left lists various settings categories. The main content area is titled "SSH keys / Add new" and contains a form with a "Title" field and a "Key" field. The "Key" field has a text box with a placeholder message: "Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'". A green "Add SSH key" button is located at the bottom of the form.

Jonathan Arp
Your personal account [Switch to another account](#)

[Go to your personal profile](#)

- Account settings
- Profile
- Account
- Appearance
- Account security
- Billing & plans
- Security log
- Security & analysis
- Sponsorship log
- Emails
- Notifications
- Scheduled reminders

SSH keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

[Add SSH key](#)

Getting Started

- `cd ~/`
- `mkdir software_engineering`
- `cd software_engineering`
- `git clone git@github.com:ITA0-40540/Python-Examples.git`
- `cd Python-Examples`

Coding Prompt: Create Coding Journal Repo

- Brush up on your basic python skills
 - Lists, looping, if/else, dictionaries
 - Do it in context of git and source control
 - Add files
 - Commit files
 - Create a branch
 - Add and commit files
 - Merge branch
 - Push to remote
-