**1: Prove your skills with GitHub**

**Link:**

**Notes:**

# The Tech World Has Adopted GitHub

In 2017, **[GitHub saw](https://octoverse.github.com/" \t "_blank)**:

* 1.5 million organizations using the platform
* 24 million users across 200 countries, from entrepreneurs to open source contributors to company employees
* 25.3 million active repositories
* 47 million pull requests, which means people and organizations are working together

Software engineers, developers, and data scientists at most companies use GitHub daily to house their work, read each other's documentation, and collaborate across teams and organizations.

Recruiters, hiring managers, and senior engineers search through GitHub for potential job candidates. Your GitHub profile - activity, repos, commits, documentation - provides evidence that you're a skilled engineer.

Udacity graduate Ryan Waite was [**hired**](https://blog.udacity.com/2016/03/student-success-ryan-waite-goes-to-nasa-and-beyond.html) based on the strength of his GitHub profile.

# No Resubmission Required

Unlike other Nanodegree projects, you submit to a Career Service once to get feedback and do not need to meet specifications to continue on in your program. We highly encourage you to take advantage of this guidance; most Alumni who report jobs and other opportunities (such as collaboration) attribute their success to networking.

### Career Review Requirement

Your Nanodegree program was designed to ensure your long-term success in the field, and to pursue a successful career you must showcase the skills you’ve learned to employers effectively. ***Therefore, this is a requirement for graduation, unless you opt-out.***

#### Opting Out

If improving your GitHub profile does not currently align with your Nanodegree goals, **you may opt-out of this graduation requirement**.

If you would like to opt-out, please click [***here***](https://forms.udacity.com/forms/career-opt-out).

# Let's Get Started

Submit your GitHub profile and receive in-depth and honest feedback from a Career Reviewer. They'll ensure you're presenting your best self to the community.

[**GET FEEDBACK**](https://review.udacity.com/#!/rubrics/52/submit-link)

## Resources

We recommend you review these resources to ensure your profile is ready for feedback.

1. [**Rubric**](https://review.udacity.com/#!/rubrics/52/view). Your project will be reviewed by a Udacity Career Reviewer against this rubric.
2. [**Checklist**](https://docs.google.com/document/d/1a9AKnNyqfGgdQV5ohPCN5H9ntnEUhMptWMwVBWURCN0/pub?embedded=true). Based on the project rubric, this is a handy checklist on GitHub best practices.
3. [**Career Resource Center**](https://career-resource-center.udacity.com/linkedin-github-profiles). Find additional tips and guides on developing your GitHub Profile.

If you'd like more guidance, continue on to the lesson to learn more about GitHub best practices on commit style, forking and starring, READMEs, and more!

**2: Introduction**

**Link:** [**https://www.youtube.com/watch?v=Vnj2VNQROtI**](https://www.youtube.com/watch?v=Vnj2VNQROtI)

**Notes:**

* [**Version Control with Git**](https://www.udacity.com/course/version-control-with-git--ud123)

**3: GitHub profile important items**

**Link:** [**https://www.youtube.com/watch?v=prvPVTjVkwQ**](https://www.youtube.com/watch?v=prvPVTjVkwQ)

**Notes:**

[**How to use Git and GitHub**](https://www.udacity.com/course/how-to-use-git-and-github--ud775)

**4: Good GitHub repository**

**Link:** [**https://www.youtube.com/watch?v=qBi8Q1EJdfQ**](https://www.youtube.com/watch?v=qBi8Q1EJdfQ)

**Notes:**

**5: Interview with Art-Part 1**

**Link:** [**https://www.youtube.com/watch?v=ClLYamtaO-Q**](https://www.youtube.com/watch?v=ClLYamtaO-Q)

**Notes:**

**6: Identify fixes for example “bad” profile**

**Link:** [**https://www.youtube.com/watch?v=AF07y1oAim0**](https://www.youtube.com/watch?v=AF07y1oAim0)

**Link2:** [**https://www.youtube.com/watch?v=ncFtwW5urHk**](https://www.youtube.com/watch?v=ncFtwW5urHk)

**Notes:**

**7: Quick Fixes #1**

**Link:** [**https://www.youtube.com/watch?v=Lb9e2KemR6I**](https://www.youtube.com/watch?v=Lb9e2KemR6I)

**Notes:**

**8: Quick Fixes #2**

**Link:** [**https://www.youtube.com/watch?v=It6AEuSDQw0**](https://www.youtube.com/watch?v=It6AEuSDQw0)

**Notes:**

<https://docs.google.com/document/d/e/2PACX-1vSkShiwfp8DP79yO1wiGI2cyDqlzMugMdivXzMfr_6CVESvbLAVYeiZowV47VJMRQ_L4F4fdeQjo9qd/pub?embedded=true>

**9: Writing READMEs with Walter**

**Link:** [**https://www.youtube.com/watch?v=DQEfT2Zq5\_o**](https://www.youtube.com/watch?v=DQEfT2Zq5_o)

**Notes:**

This is the link for [Writing READMEs with Walter](https://classroom.udacity.com/courses/ud777).

https://classroom.udacity.com/courses/ud777

**10: Interview with Art-Part 2**

**Link:** [**https://www.youtube.com/watch?v=Vvzl2J5K7-Y**](https://www.youtube.com/watch?v=Vvzl2J5K7-Y)

**Notes:**

**11: Commit messages best practice**

**Link:**

**Notes:**

As Art said, commit messages are essential for communicating why your code was changed. This is for your coworkers or collaborators, and also for your future self. Let’s go through some best practices for writing commit messages.

Here is the message format we use here at Udacity. You can find it [here](http://udacity.github.io/git-styleguide/) for future reference.

type: subject

body

footer

The first line is the subject. This should be a short description of what changed. Nothing like “fixed it” or “did something,” these need to be clear and informative, and try to avoid profanity. The subject should be 50 characters or less, with the first letter capitalized, and end without a period. At Udacity, we also include a short annotation about the type of the commit, if it is a bug fix, a feature, change to the documentation, etc.

The body is next, this is where you give a more detailed description of why you made the change. The body should typically have around 72 characters per line. This is to ensure that the message fits into a terminal window when using git on the command line. You’ll also need to make sure there is a blank line between the subject line and the body. You can also add bullet points, using asterisks or dashes, when you need to make a list.

Some commits don’t require a body in the message. If you fix a typo for example, it’s okay to only have a subject line.

You can also include a footer, typically this will be used to indicate which issues or bugs the commit addresses.

A more fleshed out example looks like this:

feat: Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72

characters or so. In some contexts, the first line is treated as the

subject of the commit and the rest of the text as the body. The

blank line separating the summary from the body is critical (unless

you omit the body entirely); various tools like `log`, `shortlog`

and `rebase` can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you

are making this change as opposed to how (the code explains that).

Are there side effects or other unintuitive consequences of this

change? Here's the place to explain them.

Further paragraphs come after blank lines.

- Bullet points are okay, too

- Typically a hyphen or asterisk is used for the bullet, preceded

by a single space, with blank lines in between, but conventions

vary here

If you use an issue tracker, put references to them at the bottom,

like this:

Resolves: #123

See also: #456, #789

This does come with an exception of course. If you are working on an open source project, be sure to follow the message format for that project. This will make the maintainers happy and increase the chance your pull request is accepted.

**12: Reflect on your commit messages**

**Link:** [**https://www.youtube.com/watch?v=\_0AHmKkfjTo**](https://www.youtube.com/watch?v=_0AHmKkfjTo)

**Notes:**

13: Participating in open source projects

Link: <https://www.youtube.com/watch?v=OxL-gMTizUA>

Notes:

[Get Started With Open Source Projects](https://blog.udacity.com/2013/10/get-started-with-open-source-projects.html)

14: Interview with Art-Part 3

Link: <https://www.youtube.com/watch?v=M6PKr3S1rPg>

Notes:

15: Participating in open source projects 2

Link: <https://www.youtube.com/watch?v=elZCLxVvJrY>

Notes:

16: Starring interesting repositories

Link: <https://www.youtube.com/watch?v=U3FUxkm1MxI>

Link2: <https://www.youtube.com/watch?v=ZwMY5rAAd7Q>

Notes:

17: Next steps

Link:

Notes:



**Apply these best GitHub practices moving forward and attract the attention of recruiters and senior software engineers.**

You'll want to:

* Commit every week, at minimum
* Get 2-4 weeks streaks

A personal side-project may grow into something bigger. You can't always predict what will catch the eye of a recruiter or senior software engineer, but when it does, you want to showcase your best work.

Continue building your knowledge of using GitHub and improving your profile, be sure to check out the "GitHub Professional Profile Resources" found in the Classroom folder under Lesson Resources.

Once you've implemented the necessary changes to make your profile more professional, remember to maintain activity by regularly making commits on GitHub.

Good luck!

18: Project