Introductions

605.401: Foundations of Software Engineering

Fall 2017

Phabricator is suite of web-based software development tools, including a bug tracker (Maniphest), code review (Differential), repository browser (Diffusion), and wiki (Phriction), that originated at Facebook. Git is a popular distributed version control system designed for scalability and data integrity. This assignment provides practice using Phabricator and Git by writing some biographical information about yourself and sharing it with the rest of the class.

It is recommended that you use a virtual machine (VM) for your work in this course, which ensures a common environment for everyone. I recommend using VirtualBox with a Debian-based Linux distribution such as Ubuntu or Linux Mint as the operating system (OS). Of course, you are not strictly required to follow these recommendations, but due to the variety of potential system configurations, *you should not expect any special support your particular system*. That is, you are solely responsible for any issues that arise on your system.

1 Phabricator

1.1 Account

The first step is to create an account on Phabricator. Open a web browser and navigate to https://web4.jhuep.com/. You should see a login form on the main page.

Click the "Register New Account" button at the bottom of the form. Enter the required information in the registration form. Please use your Johns Hopkins Enterprise Directory (JHED) ID[1] as your username and JHU-affiliated email as the email address.

After clicking the registration button, an email will be sent to the site administrator(s) to approve your account. After your account is approved, you will receive a welcome message.

1.2 Settings

Phabricator's repositories are hosted using SSH. Users must provide their public key to Phabricator for SSH authentication.

The following steps associate a SSH public key with your Phabricator account:[2]

[1] Your JHED ID is used for single sign-on (SSO) at Johns Hopkins University (JHU), including authenticating to sites like Blackboard.

[2] If you do not have an SSH public key, follow the steps in Appendix A to create one.

1. Click on your user icon in the top right corner of the page and select "Settings" from the drop-down menu to access your personal settings.
2. Click "SSH Public Keys" under "Authentication" in the left menu bar.
3. Click the button labeled "SSH Key Actions."
4. Click the button labeled "Upload Public Key" in the menu.
5. Copy your public key into the text box labeled "Public Key," provide a name for the key, and click the button labeled "Upload Public Key."

2 Git

2.1 Configuration

If you haven't done so previously, now is the time to introduce yourself to Git. Git records the name and email address for every commit—it's part of the permanent record of the repository. Follow the instructions in *Pro Git* [Chacon and Straub, 2014] to set your user name and email address.[3] Please use the same name and email address as your Phabricator account.

2.2 Repository

Phabricator's tool for repository hosting is Diffusion. You should see a link to Diffusion in the left navigation bar of Phabricator's home page. Click the link and look for the "introductions" repository. Click the link on the "introductions" repository.

Clone the repository using the URL shown in Diffusion. The complete command should look like the following:

```
git clone
↪  ssh://git@web4.jhuep.com:2222/diffusion/5/introductions.git
```

If `git clone` returns an error similar to "Permission denied (publickey)," then there is an error with the public key that you uploaded. Try uploading the public key again. If you experience a timeout or cannot connect to the remote machine (`web4.jhuep.com`), the most likely cause is that your network is blocking the port that Phabricator uses (e.g., 2222); try again from another network (e.g., at home). Otherwise, if you experience other issues, please email the instructor for troubleshooting assistance.

2.3 Changes

Copy the provided template to a file named *`JHED`*`.txt` (where *JHED* is replaced with your JHED ID) in the directory corresponding to the current term (e.g., *year.term* where 1 designates the spring term, 2 the summer term, and 3 the fall term). Update the copied file so it includes the following information (see instructions in the template):

- Name
- Employer (or general job description)

[3] https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

- Contact information (email at a minimum)
- Education
  - Prior degrees
  - Prior software engineering courses
- Programming experience: language(s) and number of years for each language
- Development interests: presentation layer (i.e., front end), data access layer (i.e., back end), or algorithms

For the development interests, a short description accompanying your preference may be helpful. This information will be considered when creating teams for the class project, but there is no guarantee that you will be assigned to the team that matches your stated preference(s).

Include a picture of yourself (one that is clearly recognizable and ideally with a file size under 500 KB) in the `figs/` directory; the pictures will help learn everyone's name more quickly. Add these files to the repository and commit your changes.[4]

3 Review

In many cases, the next step would be sharing your change with others by pushing it to back to Phabricator. Best practices in software engineering, however, often include reviews to check for mistakes. It's a bit like asking someone else to proofread a paper prior to its submission in an effort to identify any lingering typographical mistakes.

3.1 Arcanist

Arcanist provides a command-line interface to Phabricator. Arcanist supports many Phabricator tools, including Differential for code reviews, and manages common workflows for these tools.

Arcanist's setup can be challenging, particularly when configuring certificate authoritys (CAs). The dev-tools repository (https://web4.jhuep.com/diffusion/DEV/) contains an installation script to install Arcanist and to configure it. This script automates most of the installation instructions but assumes that you are using a Debian-based Linux distribution. If you are using another OS, you should follow the installation instructions online: https://secure.phabricator.com/book/phabricator/article/arcanist/#installing-arcanist.

3.2 Differential

Phabricator's tool for code reviews is Differential. The simplest way to interact with Differential is using Arcanist. Quite simply, executing `arc diff` inside a properly-configured repository will send your changes to Differential for review. An independent reviewer will examine your proposed change and either accept it or request further

[4] Don't forget to write an informative commit message! If you need a refresher on what comprises a good commit message, then review the rules in "How to Write a Git Commit Message" (http://chris.beams.io/posts/git-commit/).

changes. For example, if your commit omits a picture of yourself, then your proposed change should be rejected by the reviewer, which gives you an opportunity to add the missing picture.

Use `arc diff` to submit your change for review. This terminal command may be entered from any directory within the project repository; in general, it's simplest to enter this command immediately after committing to the repository.[5] Follow the prompts to submit your change. Note that the "Test Plan" field is required although "N/A" is the best response for this particular change. Arcanist will submit the proposed change to Differential and provide a link to the pending review.

```
Created a new Differential revision:
        Revision URI: https://web4.jhuep.com/D42
```

If you open your web browser and navigate to the provided link, you should see your review. Phabricator is configured to add "Blocking Reviewers" automatically; these individuals will receive an email notifying them about the change. Likewise, if changes are requested, then you'll receive an email notification from Phabricator.[6]

You should verify that "Maintainers" is listed under the "Group Reviewers" for your revision. If not, please notify the instructor and, separately, use the form at the bottom of the page to add this group manually. At this point, you've completed this assignment and your only responsibility is to be responsive to any changes requested by the reviewers. If you happen to disagree with any requests, the comments in Differential are the best way to discuss the matter further. Such comments provide a permanent record of the conversation, which often proves valuable in the context of a software engineering project. When your proposed revision is accepted, you should receive an email notification from Phabricator.

References

[Chacon and Straub, 2014] Chacon, S. and Straub, B. (2014). *Pro Git*. Apress, 2nd edition.

A Generating an SSH Key Pair

Use the steps below to generate an SSH key pair and display the public key.

A.1 Instructions for Unix-like Operating Systems

Unix-like operating systems, including Linux and OS X, have a variety of command-line tools to generate and manage SSH keys.

Use the following steps to generate an SSH key pair and display the public key.

1. Open a terminal.

[5] If you have uncommitted changes, Arcanist will prompt you to commit them. Other code review tools do not perform this step automatically so committing first provides the most consistent workflow.

[6] *You must verify your email address before Phabricator will send you notifications!* Otherwise, you will not be notified when comments are posted, your revision is accepted, etc.

2. Check for existing SSH keys:

   ```
   ls -l ~/.ssh/*.pub
   ```

   If any files are listed, then you already have an SSH key. Skip the following step.
3. Create a key pair:

   ```
   ssh-keygen -t rsa
   ```

   You may use the defaults in response to the prompts—just press "Enter" to proceed.[7]
4. Display the public key:

   ```
   cat ~/.ssh/id_rsa.pub
   ```

   Your public key is printed on the terminal. It will look similar to the following:

   ```
   ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC+9mWFaa4nxinfs/SuYfP
   ↪ U1LC2LLqAsRS2g3oeGKky3iLAKwRi/SBcNGncOkzq2EAIsvuoHOk9T/
   ↪ bo7EaErPHPLxo+wpBPZ54zuJUW60h/GEMGMk+0xNbT1iyWsYt9N8HJm
   ↪ BMWYcm1i8eIxr8iwSeS7/go1GymiMCJu0w2dOelDGqemNWXdaRNRqQB
   ↪ aJmrcAJJC186sGf2709lb2Sq5ON5lkNFfD3+iUh5P4ZQphQSTOo1nGz
   ↪ 76GvrUOUc3rRvJR+rTN7WKuUZUJL9hHxc812w2/22/5SNlnn/MRuh5D
   ↪ F5k2rdxyYsemTj0J7mup4LKR0V6ZzmutxTFZqeUY3xvkFJ
   ↪ jcoffma2@web4
   ```

   You may copy and paste the public key from the terminal into Phabricator.

A.2 Instructions for Windows

PuTTY is an open source implementation of SSH for Windows. PuTTY includes utilities to generate key pairs and an authentication agent to manage keys used for SSH connections.

The following steps outline how to create an SSH key pair and load it into PuTTY's authentication agent.[8]

1. Download PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. You may install the MSI package or download the individual binaries (`puttygen.exe` and `pageant.exe`).
2. Open PuTTYgen.
3. Click the "Generate" button. You may use the defaults in response to the prompts.[9]
4. Save the private key and the public key using the appropriate buttons.
5. The "Public key for pasting into OpenSSH authorized_keys" field provides the public key is the format required by Phabricator. You may copy-paste from this field directly into Phabricator's "Upload Public Key" dialog box.

[7] You may provide a passphrase to encrypt the private key. While a passphrase does provide additional security, it requires you to type it each time that the key is used.

[8] This information is compiled from various online sources, including the PuTTY documentation (http://www.chiark.greenend.org.uk/~sgtatham/putty/docs.html). If you experience an issue, please check PuTTY's documentation to see if it addresses the problem.

[9] You may provide a passphrase to encrypt the private key. While a passphrase does provide additional security, it requires you to type it each time that the key is used or loaded by Pageant.

6. Load the private key into Pageant.
    (a) Open Pageant.
    (b) Click the “Add Key” button.
    (c) Use the file dialog box to navigate to and select the private key file (`*.ppk`) saved previously from PuTTYgen.
    (d) If the private key is encrypted via a passphrase, Pageant will prompt you to enter the passphrase to decrypt the key.