Setting up GitLab on IntelliJ

This document will walk you through the steps required to set up GitLab on IntelliJ. It assumes that you are on a Windows machine with IntelliJ already installed. If you already have Git installed on your machine (regardless of operating system), then skip to step 2.

Contents

Git for Windows

1. Git for Windows is available for installation at https://gitforwindows.org/. Download and run the setup executable.
    1.1. It will prompt you to accept the GNU General Public License, then to choose which components to install.
        1.1.1. Git Bash is recommended, as its commands follow the same conventions as Linux, instead of those normally followed by the command line on Windows.
    1.2. The setup wizard will then prompt you for multiple preferences. The default options have been tested and proven to work without issue, but you should be able to select otherwise without much issue.
    1.3. Select your preferred default Git editor.
    1.4. Select how you would like your PATH environment to be changed.
        1.4.1. “Git from the command line and also from 3rd party software” is recommended, as it allows you to use Git from within the Windows Command Prompt as well as from Git Bash.
    1.5. Select “Use the OpenSSL library” as your HTTPS transport backend.
    1.6. Select “Checkout Windows-style, commit Unix-style line endings”.
        1.6.1. This ensures that files appear correctly on your machine, while also ensuring that they are still accessible from Unix-style machines.
    1.7. Select a terminal emulator to use with Git Bash.
    1.8. Enable the Git Credential Manager.
    1.9. The selected features should now be installed on your machine, pending admin installation rights.
2. Set up Git via the Git Bash prompt.
    2.1. Open Git Bash through your start menu.
    2.2. Globally set your user name and email for future commits on this machine.
        2.2.1. Do this by executing the following commands:
            2.2.1.1. git config --global user.name “Your name”
            2.2.1.2. git config --global user.email youremail@kent.ac.uk
        2.2.2. If your global configuration is not your name and kent email address, please configure them locally as such for projects you undertake within Untitled Solutions.

Setting Up GitLab

1. The following section is based on https://docs.gitlab.com/ee/ssh/README.html. For a more detailed explanation, see the linked page.
2. The SSH protocol should be included in Git Bash.
3. The SSH protocol allows you to “push” and “pull” from the gitlab server remotely, and it authenticates to the GitLab remote server without supplying your username or password each time. To use it, you will need to generate a public/private key pair, and copy the public key to the appropriate page on GitLab.
4. Generating a key pair
    - 4.1. Open Git Bash (or equivalent if using another operating system).
    - 4.2. The Kent School of Computing GitLab accepts RSA keys, so generate one in Git Bash with the following command:
        - 4.2.1. ssh-keygen -o -t rsa -b 4096 -C "email@example.com"
        - 4.2.2. (Replacing the email with your kent email, though this is an optional comment for telling keys apart.)
        - 4.2.3. Update: The aforementioned GitLab also accepts ED25519 SSH keys, which are more secure. To generate an ED25519 SSH key instead, type in the following command:
            - 4.2.3.1. ssh-keygen -t ed25519 -C "email@example.com"
    - 4.3. Git Bash will prompt you to input a file path to save the SSH key to. The default path is recommended, as it will allow you to use it with no additional configuration.
    - 4.4. Git Bash will then prompt you to input a password to secure the SSH key pair. This is optional, though recommended.
5. Adding an SSH key to your GitLab account
    - 5.1. Copy your public SSH key to the clipboard by using the command below (See linked GitLab page above for non-Windows operating systems):
        - 5.1.1. cat ~/.ssh/id_rsa.pub | clip
        - 5.1.2. If you generated an ED25519 SSH key, use the following command instead:
            - 5.1.2.1. cat ~/.ssh/id_ed25519.pub | clip
    - 5.2. On your GitLab account, click your avatar in the upper right corner and select Settings, then navigate to SSH Keys.
    - 5.3. Paste your public key in the “key” section, and name your individual key via a title in the field provided.
    - 5.4. Click Add key.
6. Test whether you can access GitLab
    - 6.1. To test whether your SSH key was added correctly, run the following command in your terminal:
        - 6.1.1. ssh -T git@git.cs.kent.ac.uk
    - 6.2. The first time you connect to GitLab via SSH, you will be asked to verify the authenticity of the GitLab host you are connecting to. Please ensure the SSH host key fingerprint is correct, to make sure you’re connecting to the right server.
        - 6.2.1. For the appropriate SSH host key fingerprint, see https://git.cs.kent.ac.uk/help/instance_configuration.

6.2.2. Once added to the list of known hosts, you won't be asked to validate the authenticity of GitLab's host again.

6.3. Run the above command once more, and you should only receive a Welcome to GitLab, @username! message.

Configuring IntelliJ Git integration

1. The following section is based off of the jetbrains IntelliJ help, available at the following links for more details:
    1.1. https://www.jetbrains.com/help/idea/using-git-integration.html
    1.2. https://www.jetbrains.com/help/idea/set-up-a-git-repository.html#clone-repo.
2. In the Settings/Preferences dialog in IntelliJ, select Version Control | Git in the left pane and specify the path to the Git executable.
    2.1. This should have been detected automatically by IntelliJ as C:\Users\[yourUserName]\AppData\Local\Programs\Git\cmd\git.exe
        2.1.1. Note: If Git was not installed in this location, you should check the "Program Files" folder or similar folder on your machine.
3. In the same pane, you can choose whether you want to use a native or built-in SSH executable by selecting from the SSH executable drop-down list.
    3.1. If you select built-in, all authorisation is performed on the IDE side (IntelliJ).
    3.2. If you select native, all authorisation is performed on Git side, using Git Bash.
4. You can configure IntelliJ to remember your passwords, so that you do not have to specify your credentials each time authorisation is required. For further details, see Link 1.1.
5. Next, set up a local Git repository.
    5.1. This will be done in one of two ways:
    5.2. If a project in the Untitled Solutions repository has already been set up, you should clone it to your local repository.
        5.2.1. From the main menu, choose VCS | Checkout from Version Control | Git, or, if no project is currently opened, choose Checkout from Version Control | Git on the Welcome screen.
        5.2.2. In the Clone Repository dialog, specify the URL of the remote repository you want to clone
            5.2.2.1. You can click Test beforehand to make sure that connection to the remote can be established.
            5.2.2.2. If a connection cannot be established, ensure that Git works correctly on your machine, and the path is correct.
            5.2.2.3. If a connection still cannot be established, ensure the latest version of IntelliJ is installed, as some past versions are known to have faulty Git integration.
            5.2.2.4. The URL can be found on the GitLab page for that project by clicking "clone", then copying the "Clone with SSH" link.
        5.2.3. In the Directory field, specify the path where the folder for your local Git repository will be created, into which the remote repository will be cloned.
        5.2.4. Click Clone. If you want to create a IntelliJ IDEA project based on the sources you have cloned, click Yes in the confirmation dialog. Git root mapping will be automatically set to the project root directory.

5.3. If a project does not already exist in the Untitled Solutions repository, you can put an existing project under Git version control.
   5.3.1. Open the project that you want to put under Git
   5.3.2. From the main menu, choose VCS | Enable Version Control Integration.
   5.3.3. In the dialog that opens, select Git from the drop-down list and click OK.
   5.3.4. You will then have to push your project to the Untitled Solutions repository so others have access to it.
      5.3.4.1. Invoke the Push dialog when you are ready to push your commits by selecting VCS | Git | Push from the main menu.
      5.3.4.2. If you haven't added any remotes so far, the Define remote link will appear instead of a remote name. Click it to add a remote.
      5.3.4.3. In the dialog that opens, specify the remote name and the URL where it will be hosted, and click OK.

6. You are now able to commit changes to your local repository, as well as pull and push from the Untitled Solutions repository from within IntelliJ.
   6.1. To do so, open the Version Control window in IntelliJ or select the corresponding options from the Version Control dialog.
   6.2. To commit to your local repository, select VCS | Commit.
      6.2.1. This will prompt you to select which files to commit, as well as to input a Commit Message.
      6.2.2. Commit Messages should follow the guidelines set out in the Quality Assurance document.
   6.3. To pull from the Untitled Solutions repository, select VCS | Git | Pull.
      6.3.1. IntelliJ will prompt for a Git Root and Remote. The defaults should suffice.
      6.3.2. The Git Root should be the folder on your local machine where the project is found.
      6.3.3. The Remote should be the URL which can be found on the GitLab page for that project by clicking “clone”, then copying the “Clone with SSH” link.
   6.4. To push a committed version to the Untitled Solutions repository, select VCS | Git | Push.
      6.4.1. IntelliJ will prompt for the repository to commit.
      6.4.2. For details, see https://www.jetbrains.com/help/idea/commit-and-push-changes.html.