

2.1.4 git pull

2.1.4.1 Description

The *git pull* command updates the local repository with any changes that may have been committed to the remote repository. To ensure the code is kept in sync across the team, a *git pull* should be done frequently and **always** before any new code is developed or edited. The frequency with which to initiate a *git pull* depends on the velocity of changes to a repository, but it is recommended that a *git pull* be done **once a day, at a minimum**.

2.1.4.2 Syntax

```
git pull
```

2.1.5 git add

2.1.5.1 Description

The *git add* command adds new files to your local Git repository. New files must be added before changes can be committed and pushed to the remote repository.

2.1.5.2 Syntax

```
git add {space separated list of files to add}
```

2.1.6 git commit

2.1.6.1 Description

The *git commit* command is used to save changes to your local repository. When committing code to your local repository, the files need to be explicitly referenced in the command. After the files have been committed to your local repository, they can be pushed out to the remote repository, where they will be available for other developers.

2.1.6.2 Syntax

```
git commit -m "{comment describing the changes}" {space separated list of files to commit}
```

2.1.7 git push

2.1.7.1 Description

The *git push* command pushes the changes to the local repository on the server up to the remote Git repository, where the changes will be available for other developers to retrieve.

2.1.7.2 Syntax

```
git push {space separated list of files to commit}
```

2.2 Git Repository for USPS

[Table 2](#) contains a list of Git repositories available for use by USPS.

Note: The <https://gitlab.ondemand.sas.com> repository is included for historical purposes, but it should not be leveraged for new development.

Table 2: USPS Git Repository Locations

Project	Location	Details
VSP hosted projects	https://git.ondemand.sas.com/users/sign_in	<ul style="list-style-type: none">• Authentication Handled via VSP Domain• All Customer Hosted Projects within VSP
ZSP hosted projects	https://zgitlab01.zsp.sas.com	<ul style="list-style-type: none">• Authentication Handled via ZSP Domain• All Customer Hosted Projects within ZSP
SAS	https://gitlab.sas.com	<ul style="list-style-type: none">• Authentication Handled via CARYNT Domain• Non-hosted projects can use this if customer's on-prem environment has access to this website
DECOM – Old VSP / QA GitLab	https://gitlab.ondemand.sas.com/users/sign_in	<ul style="list-style-type: none">• Previously managed by QA, system was used for some projects

3 Environment Setup

To leverage Git in USPS, Technical Leads first request that a repository to be set up by the IT team. After the repository has been established, access to the repository is granted by the Account Team through Jira ticket requests (see Section 3.2). After they have access to the repository, the developers create their playpens on the development server by cloning the Git repository locally. If the project leverages SAS Studio V, then the repository can be further integrated by leveraging the Git User Interface built into SAS Studio V.

3.1 Request a Git Repository

Note: Before requesting a Git repository for the relevant project, ensure that one does not already exist.

To request a new Git repository, use the [Request a New GitLab Subgroup](#) form in the *GHUSPS Request Catalog* in ServiceNow. For most projects, only one Git repository is necessary. When a new Git repository is requested, the name given to the repository is on the Line of Business (LOB) field. This LOB is equivalent to the project TLA.

3.2 Request Access to a Git Repository

After the repository has been created, you need to be added to the *{tla}git* group so that you can access the repository. If you are an existing project team member, create an *Add Entitlement* sub-task under your main ACT in Jira. Within the sub-task, request that you be added to the *{tla}git* group. This ticket needs to be approved by the relevant Project Owner and then assigned to the Account Team for completion.

If you join the project after the repository has been created, then when you create an ACT ticket to gain access to the project itself, ensure that you also request *{tla}git* access.

3.3 Set Up Token Authentication

Leveraging an authentication token to associate your development environment with your Git repository eliminates the necessity of logging into Git each time you need to interact with the repository. This results in a more seamless integration with development environment (eliminates hurdles).

1. To set up token authentication with the Git server, you must first create the RSA token on your development server. Start by logging into your development environment. USPS standard deployment leverages Red Hat Linux as the development environment. The screen shots included in this section were taken from a Red Hat server.

```
> ssh-keygen -t rsa <enter>
```

Generating public/private rsa key pair.

Enter file in which to save the key (/home/{username}/.ssh/id_rsa): <enter>

/home/{username}/.ssh/id_rsa already exists.

Overwrite (y/n)? y <enter>

Enter passphrase (empty for no passphrase):<enter>

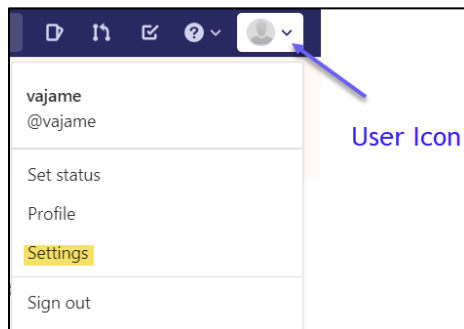
Enter same passphrase again:<enter>

Your identification has been saved in `/home/{username}/.ssh/id_rsa`.

Your public key has been saved in `/home/{username}/.ssh/id_rsa.pub`.

2. After you have completed the steps above, open the `/home/{username}/.ssh/id_rsa.pub` file and copy the entire contents of this file to your clipboard.
3. With the contents still in your clipboard, open your web browser and navigate to the GitLab instance associated with your project.
4. Log on to GitLab.
5. Identify and click the user icon in the upper right corner of the user interface ([Figure 1](#)).
6. From the resulting menu, select **Settings** ([Figure 1](#)).

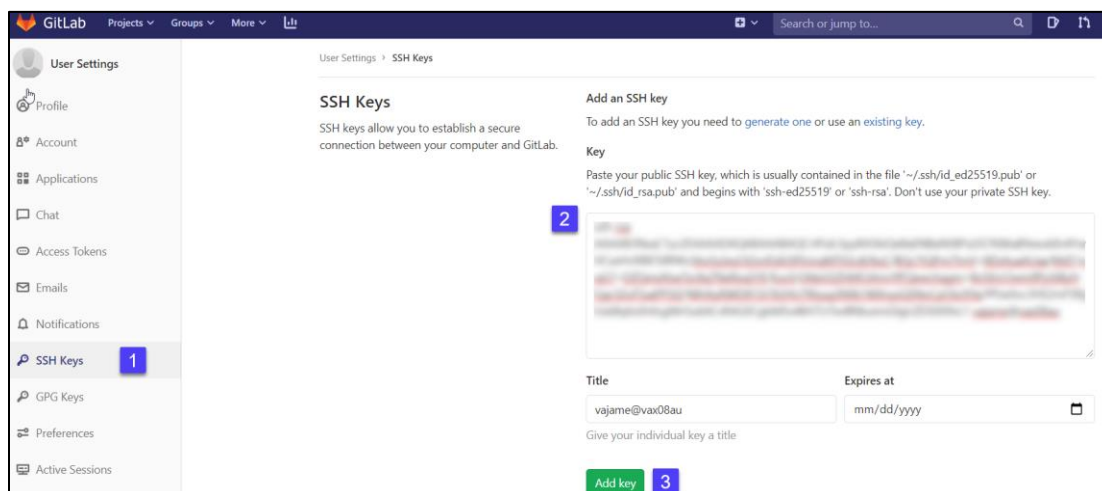
Figure 1: GitLab User Icon



7. On the **User Settings** screen
 - a. Select the **SSH Keys** link on the left menu (identified in [Figure 2](#) with a **1**)
 - b. Paste your RSA key into the **Key** field (identified in [Figure 2](#) with a **2**)
 - c. Select **Add key** (in [Figure 2](#) with a **3**).

At this point, your development server should now be able to communicate with GitLab without the need to enter your credentials each time.

Figure 2: Adding SSH Key to GitLab



3.4 Set Up Playpen

Now that you have opened the lines of communication between the development server and the remote repository, you can pull the project repository down to the server to begin your development work.

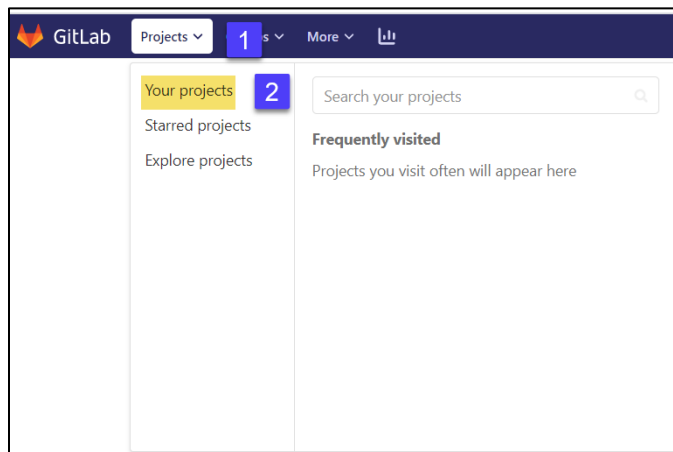
Using either MobaXterm or PuTTY, log into your development server and navigate to the developer playpen location for your project.

Note: In standard USPS implementations, the playpen area is located at `/{customer TLA}/projects/{workstream}/`, where `{customer TLA}` is the project code associated with the project and `{workstream}` is the specific workstream associated with the development work. If there is only one workstream, it is usually labeled as *default*.

At this point, you need to know the location of the Git repository that you are trying to clone. You can find this information by

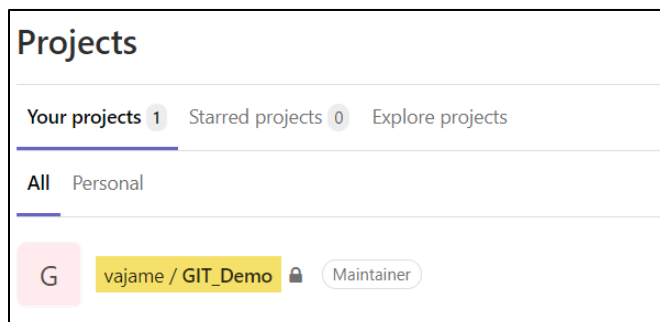
1. Open a web browser and log on to GitLab.
2. Select **Projects** on the top navigation bar (shown in [Figure 3](#) with a **1**).
3. Then select **Your projects** from the drop-down menu (shown in [Figure 3](#) with a **2**).

Figure 3: GitLab Your Projects Link



4. From the **Projects** page, select the repository that you wish to clone from the list of available repositories. In the example shown in [Figure 4](#), the repository is entitled *GIT_Demo*.

Figure 4: GitLab List of Projects




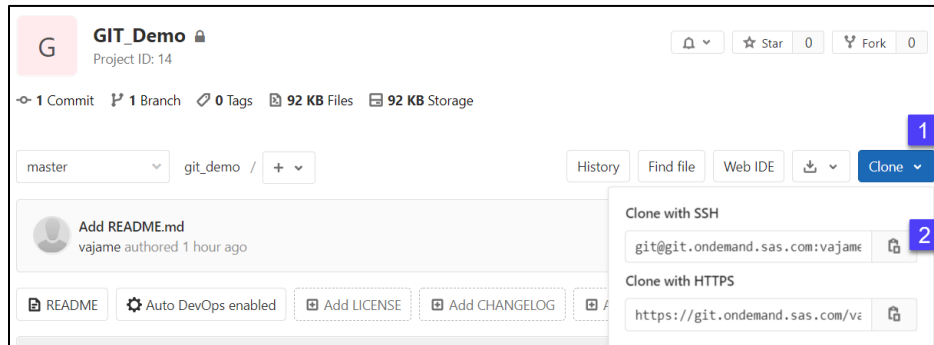
- The main repository page contains a **Clone** button (identified in [Figure 5](#) with a **1**). Select the **Clone** button, then use the  button (identified with a **2** in [Figure 5](#)) to copy the text that appears in the **Clone with SSH** field to your clipboard.

Figure 5: GitLab Clone Repository Link



- With the command still in your clipboard, return to the playpen of your development server and clone the repository using the following syntax:
`git clone {Clone with SSH command} {username}`
 where *{Clone with SSH command}* is the command copied to your clipboard and *{username}* is your userID.
 This clones the repository into a directory identified by your userID, which signifies your playpen where you are able to develop code.
- If you receive the prompt that reads *Are you sure you want to continue connecting (yes/no)?*, type **yes**.

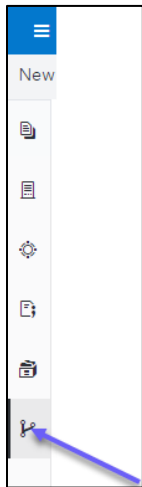
3.5 Integrate with SAS Studio V

If your project leverages SAS Studio V, then you can leverage its built-in capabilities to engage with the Git repository. To facilitate this interaction, you must first clone the Git repository to the development server, as outlined previously.

- After the repository has been cloned to the development server, open a web browser and navigate to the SAS Studio V development instance for your project.
Note: If you are unsure of the location, this information can be found in Confluence on the *Port Information* page for your project. Ensure that you are using the port information for the development server.

- When logged into SAS Studio V, select the Git Repository icon on the left navigation panel (shown with a purple arrow in [Figure 6](#)).

Figure 6: SAS Studio V Git User Interface Icon



- From the **Git Repositories** page, select **Add Repository** ([Figure 7](#)).

Figure 7: SAS Studio V Add Repository Link

