promus Documentation

Release 0.1.0

Manuel Lopez

August 22, 2014

Contents

Promus is a remote manager designed to create and manage git repositories in a remote server without the need of admin privileges.

CHAPTER 1

Basic Usage

1.1 What is Promus?

Promus is a remote manager designed to create and manage git repositories in a remote server without the need of administrator privileges. It was designed with the goal of creating your own private repositories in a server to which you have access and allows you to create connections to remote servers between you and your collaborators via ssh keys.

Promus is your personal butler. You let promus send invitations to your collaborators so that they may access your account via ssh keys and you specify in each of the repositories an access control list (acl) in which you write down the names of the administrators for the repositories (who have more control over the project) and the users.

1.2 Installing Promus

The easiest way to install promus is to use `pip`. If you wish to perform a global installation and you have admin rights then do

```
sudo pip install promus
```

or to install in some directory under your user account

```
pip install --user promus
```

If you prefer to do the installation manually then from the command line you may do the following (where x.y is the version number):

```
wget https://pypi.python.org/packages/source/p/promus/promus-x.y.tar.gz
tar xvzf promus-x.y.tar.gz
cd promus-x.y/
sudo python setup.py install
```

The last command can be replaced by `python setup.py install --user`. See PyPI for all available versions.

Once you have finished do the installation you need to set up your `$PATH` so that your shell may look for it. If you are using `bash` you can call the `install` command from promus:

```
python -m promus install
```

1.2.1 Git

Promus was designed with one goal in mind: to make `git` available in your own server without administrator rights. For this reason, before you even think about using promus you must obtain a copy of `git` which you can obtain at http://git-scm.com/downloads.

Make sure that `git` is installed in your system before proceding with the next section.

1.2.2 Setting up promus

If you completed the previous sections then you are well on your way to creating your first private repositories or to connect to one. First you need to let promus and git some information about yourself.

```
$ promus setup
Full name:
E-mail address:
Hostname alias:
Host e-mail:
Password:
```

Keep in mind that if you are not going to use your personal computer to create repositories then there is no need to provide a `Host e-mail` nor a `Password`. What is important however, is that you provide a `Hostname alias`. The alias will be helpful to identify the machine from where you provided commands.

If you are setting up promus to create repositories and your server provides you with an e-mail address then you are allowed to ommit entering your password since the system can authenticate you when you `ssh` to the server.

To make sure that the email you provided for the host is working you can use the `verify` command to make promus send you an email.

```
$ promus verify
sending email ... done
```

At this point you should be ready to start using promus.

Note: Make sure to use only one e-mail address in all the machines you are using. This will help with the identification of users even if you have different usernames in different machines.

1.3 Getting Started

Promus allows you to create repositories in a remote server to which you and your collaborators may have access to. There are several scenarios in which you can use promus.

1.3.1 Password-less Connection To Server

We assume that you wish to create a repository in some server to which you have access via `ssh` from your personal computer. We also assume that the alias given to your personal computer is `mac` and shall henceforth refer to the personal computer by the alias.

To create a passwordless access to a remote machine we need to let the remote machine get a hold of a public key that may have been created by promus or may have already existed in the `mac` machine.

To do this use the promus `connect` command:

```
promus connect username@server-name server-alias
```

You will be prompted for your password the remote machine has not been set up with passwordless access. Once this is done you will be able to `ssh` as you always do minus the password, or you can use the `server-alias` you provided to promus:

```
ssh server-alias
```

Once you connect to the server you can install promus there so that you may start creating your first repositories.

1.3.2 Send a Collaboration Request

If you wish for collaborators to be able to access your repositories you need to let promus know what their public keys are so that they may access your account with a limited amount of privileges. These privileges are minimum and they only include `git` commands.

To let promus send an email in your behalf you can execute the following command:

```
promus send request email@hostname 'First Last'
```

The last argument, namely `'First Last'` is not required but if you provide it, then promus will address your collaborator by that name. The email sent will contain contain a link to this documentation and the command that needs to be executed to accept the request.

1.3.3 Accepting a Collaboration Request

By accepting a collaboration request you will simply send your public key so that your host can recognize you. The instructions on how to do this are stated on the email sent by your host. The typical command to execute is

```
promus add host username@hostname
```

This has to be done from a working directory which contains the file `username@hostname`. This file contains a temporary private key which was set up just so that you can send your public key. After this, you will be able to connect to a repository to which your collaborator has given you permission to access.

Note: The email address that you used during the promus setup must be the same email address from which you retrieved the private key. This is done for security purposes. If your promus is set up with a different email address you may ask your collaborator to send you another invitation to the correct email address.

1.3.4 Initializing a Repository

To create a repository in a remote server you will have to first access the server and execute the following command:

```
promus init <name_of_repository>
```

Where `<name_of_repository>` should be replaced by any name you desire.

Note: This will create the repository in `~/git`. You may specify the directory in which you want the repository to be created by specifying the option `--dir`.

1.3.5 Cloning a Repository

To clone a repository you can use the command:

```
promus clone <server>:/path/to/repository.git
```

Here `<server>` should be replaced by the host you are trying to obtain the repository from. The path to the repository is usually `~/git/<repository_name>.git`.

Note: If you are a host trying to initialize a repository you must clone the repository as a git user. That is, you must send a request to yourself and add yourself as a host.
