ssheepdog Documentation

Release

2012, SheepDogInc

February 03, 2012

CONTENTS

ssheepdog is a public ssh key management tool for teams of programmers. Different people need different privileges to different servers. Ssheepdog allows you to specify these relationships and then sync the keys to the appropriate servers.

ssheepdog is a django app and it's contained in the `src/` directory. You should be able to run it locally without much trouble.

ssheepdog is developed and maintained by SheepDogInc, based in Halifax, Canada.

> **Warning:** This is alpha software. Please proceed with caution.

Contents:

CHAPTER

ONE

INSTALLATION

Ssheepdog doesn't yet come as a reusable Django app, so we have provided a starter Django project for you.

```
$ git clone git://github.com/SheepDogInc/ssheepdog.git
$ cd ssheepdog
$ pip install -r requirements.txt
$ cd src
$ python manage.py syncdb
$ python manage.py migrate
$ python manage.py runserver
```

And the application will be running on `http://localhost:8000`.

Note: We **strongly** recommend using virtualenv when installing ssheepdog.

1.1 Celery tasks

If you have a lot of servers to sync, you may wish to run the sync process in the background. This is the recommended setup for production use since a real web server will timeout with such a long running request. There is a celery task that you can use to accomplish this.

In addition to your development server, you will need to run a celery worker.

```
$ python manage.py celeryd -l info
```

Then, you can run a background task like this:

```
from ssheepdog.tasks import sync

sync.delay()
```

CHAPTER

TWO

DEVELOPING

We have provided a vagrant VM configuration for testing the ssh syncing.

Boot it up:

```
$ vagrant up
```

Log in:

```
$ ssh ssheepdog@127.0.0.1 -p 2222 -i deploy/cookbooks/ssheepdog/files/default/id_rsa
```

CHAPTER

THREE

HOW IT WORKS

This section will give you an overview of how ssheepdog works. You should understand this model before you use this application in a production environment.

3.1 Models

There are three main models:

- User
- Machine
- Login

The User model represents a member of your team. It's a real person who can log into the application. This uses the built-in Django User model along with a custom `UserProfile` class. The User has the ability to access and edit their own account. They are required to specify their public ssh key once their account is created.

The Machine model represents a server. A Machine will typically have an IP address or a hostname. You can specify a client for each machine, if it's active, etc.

Each Machine will have one or more Logins associated with it. A Login is the unix username that you log in as on the Machine. Each Login knows what Users can log in with that account.

3.2 Administration

Once the application is deployed, you will be able to specify which physical Users can log into what Machines via what Logins. Once your changes are made, you will start the syncing process.

3.3 Syncing

The basic idea behind synchronization is to update the `authorized_keys` file for each login on the remote server. This file is based on the list of authorized Users for a given Login. Furthermore, the ssheepdog application itself uses an ssh key pair to log into each machine. The application's ssh key is renewed each time a sync is run.

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*