JupyterHub for Teaching

Release 1.0

Project Jupyter

Apr 01, 2018

Contents

Version: 1.0

Date: Apr 01, 2018

CHAPTER 1

Abstract

This deployment is designed for teaching a small to medium group of trusted users.

As a simple, reusable JupyterHub deployment for your reference, this repository enables installation and deployment of JupyterHub and nbgrader on a single server. The reference deployment follows best practices and has been used by Professor Brian Granger when teaching “Introduction to Data Science”.

CHAPTER 2

Contents

2.1 Design goals

2.1.1 Instructors and maintainers

When using this repository to deploy JupyterHub and nbgrader, individuals should be able to have a deployment that is as simple as possible:

- No Docker use.
- NGINX as a frontend proxy, serving static assets, and a termination point for SSL/TLS.
- A single server.
- Ansible for configuration.
- Optionally, use Let's Encrypt for generating SSL certificates.

JupyterHub

- Start from:
  - An empty Ubuntu latest stable server with SSH key based access.
  - A valid DNS name.
  - A formatted and mounted directory to use for user home directories.
  - The assumption that all users of the system will be "trusted," meaning that you would given them a user-level shell account on the server.
- Always have SSL/TLS enabled.
- Specify local drives to be mounted.
- Manage the running of jupyterhub and nbgrader using supervisor.

- Optionally, monitor the state of the server and set email alerts using NewRelic. The built-in monitoring of your cloud provider may also be used.
- Specify admin users of JupyterHub.
- Add the public SSH keys of GitHub users who need to be able to `ssh` to the server as `root` for administration.
- Manage users and authentication using either:
  - Regular Unix users and PAM (Pluggable authentication modules)
  - GitHub OAuth

nbgrader

- Run nbgrader and configure:
  - The course name.
  - The instructors username.
  - Graders' usernames.
  - The location of the nbgrader config.

2.1.2 Students

End users of this deployment should be able to:

- Use the following Jupyter kernels.
  - Python version 3 using the IPython kernel with the main Python libraries for data science.
  - Bash kernel <https://github.com/takluyver/bash_kernel>
- Sign in using their GitHub or Unix credentials.
- Have a persistent home directory.
- Have outbound network access.

2.2 Installation Guide

2.2.1 Prerequisites

- Start a server running latest Ubuntu version.
- Enable password-less SSH access for **`root`** user.
- Partition and format any local disks you want to mount.
- Verify a valid DNS entry for the server.
- Choose an SSL certificate source. Use either of these options:
  - Let's Encrypt
  - obtain a trusted SSL certificate and key for the server at that FQDN.
- Checkout the latest version of the repository including the `ansible-conda` submodule:

```
$ git clone --recursive https://github.com/jupyterhub/jupyterhub-deploy-teaching.
↪git
```

2.2.2 Create the hosts group

1. Edit the `./hosts` file to lists the FQDN's of the hosts in the `jupyterhub_hosts` group.
2. Create for each host a file in `./host_vars` directory with the name of the host, starting from `./host_vars/hostname.example`.

2.2.3 Secure your deployment

1. Create a cookie secret file, `./security/cookie_secret`, using:

```
$ openssl rand -hex 1024 > ./security/cookie_secret
```

   For additional information, see the cookie secret file section in the JupyterHub documentation.

2. If you are using Let's Encrypt, skip this step. Otherwise, install your SSL private key `./security/ssl.key` and certificate as `./security/ssl.crt`.

2.2.4 Deploy with Ansible

1. Run `ansible-playbook` for the main deployment:

```
$ ansible-playbook -i hosts deploy.yml
```

2.2.5 Verify your deployment

1. SSH into the server:

```
$ ssh root@{hostname}
```

substituting your hostname for {hostname}. For example, `ssh root@jupyter.org`.

2. Reload supervisor:

```
$ supervisorctl reload
```

2.3 Configuring nbgrader

The nbgrader package will be installed with the reference deployment.

To run nbgrader's formgrade application or use its notebook extensions, additional steps are needed.

2.3.1 Deploy formgrade

First, edit the `deploy_formgrade.yml` file with the information for each course you want to start formgrade for. Each course should have a unique *nbgrader_course_id* and *nbgrader_port*.

Second, make sure that each main instructor (the *nbgrader_owner* for each course) has logged into JuptyerHub at least once. This ensures that their home directory has been created. The home directory of the main instructor is used for the main nbgrader course files. It is assumed that the main instructor will be running the nbgrader command line programs.

Third, run the ansible-playbook to deploy formgrade:

```
$ ansible-playbook -i hosts deploy_formgrade.yml
```

Fourth, SSH into the JupyterHub server:

```
$ ssh {user}@{hostname}
```

Finally, restart jupyterhub and nbgrader by doing:

```
$ supervisorctl reload
```

2.3.2 Configuration notes

- To limit the deployment to certain hosts, add the `-l hostname` to the commands:

  ```
  $ ansible-playbook -i hosts -l hostname deploy.yml
  ```

- The logs for *jupyterhub* are in `/var/log/jupyterhub`.
- The logs for *nbgrader* are in `/var/log/nbgrader`.
- If you are not using GitHub OAuth, you will need to manually create users using *adduser*:

  ```
  $ adduser --gecos "" username
  ```

- Change the ansible configuration by editing `./ansible_cfg`.
- To manage the jupyterhub and nbgrader services by SSH to the server and run:

  ```
  $ supervisorctl jupyterhub { start, stop, restart }
  ```

2.3.3 Troubleshooting: Saving and restoring users

In some situations, you may remount your user's home directories into a new instance that doesn't have their user accounts, but has their home directories. When recreating the same users it is important that they all have the same uids so the new users have ownership of the home directories.

This is only relevant when using GitHub OAuth for users and authentication.

To save the list of usernames and uids in *{{homedir}}/saved_users.txt*:

```
$ ansible-playbook -i hosts saveusers.yml
```

Then, when you run deploy.yml, it will look for this file and if it exists, will create those users with those exact uids and home directories.

You can also manually create the users by running:

```
$ python3 create_users.py
```

in the home directory.

2.4 Using nbgrader

With the reference deployment, instructors can start to use nbgrader. This section contains a rough sketch of what that looks like. For full details see the nbgrader documentation.

2.4.1 Preparing class assignments - Instructor

To use nbgrader, an instructor will primarily use the nbgrader command line program.

Create a list of students and assignments

Before doing this, the instructor will need to edit the `nbgrader_config.py` file with a list of students and assignments as follows:

```
c.NbGrader.db_assignments = [dict(name="ps1")]
c.NbGrader.db_students = [
    dict(id="bitdiddle", first_name="Ben", last_name="Bitdiddle"),
    dict(id="hacker", first_name="Alyssa", last_name="Hacker"),
    dict(id="reasoner", first_name="Louis", last_name="Reasoner")
]
```

You can also add an `email` field to each student and a `duedate` field to each assignment.

Remember to add new assignments to the `nbgrader_config.py` file as the assignments are created.

Create an assignment directory

Create a directory for each assignment's source:

```
$ cd ~/nbgrader/<course>
$ mkdir source/<assignment>
```

Copy notebooks into assignment directory

Copy notebooks into the assignment directory:

```
$ cp ~/Problem1.ipynb ~/nbgrader/<course>/source/<assignment>
$ cp ~/Problem2.ipynb ~/nbgrader/<course>/source/<assignment>
```

Create a student version of an assignment

These notebooks should be prepared using the nbgrader "Create Assignment Cell toolbar". Now create the assignment:

```
$ nbgrader assign <assignment>
```

After creating the student versions of the notebooks, put them into the `~/nbgrader/<course>/release/<assignment>` directory, and remember to remove your solutions.

Release the assignment

Next, release the assignment to students:

```
$ nbgrader release <assignment>
```

2.4.2 Working with an assignment - Students

Fetch the assignment

At this point, students can fetch the assignment by doing:

```
$ nbgrader fetch --course <course> <assignment>
```

That will give students a copy of the assignment directory with all of the notebooks.

Submit an assignment solution

When students are done working the notebooks, they can submit the assignment by doing:

```
$ nbgrader submit --course <course> <assignment>
```

2.4.3 Grading the assignments - Instructor

Collect student assignments

You can collect submitted assignments by doing:

```
$ nbgrader collect <assignment>
```

This puts the students submitted work into the `~/nbgrader/<course>/submitted/<assignment>` directory.

Grade the assignments

To enter those notebooks into the nbgrader web grading system, run:

```
$ nbgrader autograde <assignment>
```

By default, this will rerun all of the students notebooks.

If you don't want to run them:

```
$ nbgrader autograde --no-execute <assignment>
```

2.4.4 Next steps

To see the full command line options for nbgrader, run:

```
$ nbgrader <subcommand> --help
```

Some other things you can do with nbgrader:

- Run **`collect`** and **`autograde`** commands for a single student or notebook.
- Collect a single assignment multiple times and regrade all or parts selectively.

2.5 Checklist for a JupyterHub teaching deployment

Documentation for teaching deployment: https://jupyterhub-deploy-teaching.readthedocs.io

Documentation for JupyterHub: https://jupyterhub.readthedocs.io

2.5.1 Notes

- Does **not** use Docker.
- NGINX as a frontend proxy, for serving static assets, and a termination point for SSL/TLS.
- Single Ubuntu server
- Ansible for configuration.

2.5.2 1. Prepare the server

- [ ] **Server:** running latest Ubuntu version
- [ ] **SSH:** enable password-less SSH for `ubuntu` user
- [ ] **Local disks:** partition and format
- [ ] **DNS (domain name):** valid entry for server

2.5.3 2. Install JupyterHub source

- [ ] **Source:** Clone latest `jupyterhub-deploy-teaching` repo using `--recursive` (needed for `ansible-conda`) submodule

```
$ git clone --recursive https://github.com/jupyterhub/jupyterhub-deploy-teaching.
↪git
```

2.5.4 3. Secure before deployment

- [ ] **cookie secret file:** Create `./security/cookie_secret`

```
$ openssl rand -hex 1024 > ./security/cookie_secret
```

- [ ] **SSL:**

  - Let's Encrypt: No additional steps as Ansible will install for you.
  - Third Party SSL trusted source: Install SSL private key `./security/ssl.key` and certificate as `./security/ssl.crt`.

2.5.5 4. Create JupyterHub hosts group

- [ ] **`./hosts` file:** Edit file to lists the FQDN's of the hosts in the `jupyterhub_hosts` group.
- [ ] **hostname files:** Use `./host_vars/hostname.example` as a template for creating and editing a hostname file for each host and place hostname files in `./host_vars` directory.

2.5.6 5. Configure admins

- [ ] List of admins is configured in `jupyterhub_admin_users` in the config file. Public SSH keys will be retrieved from GitHub.

2.5.7 6. Configure users

- [ ] If using PAM (Pluggable authentication modules), you will need to manually create users using adduser: `adduser --gecos "" username`.
- [ ] If using GitHub OAuth, add usernames to `jupyterhub_users` list.

2.5.8 7. Add optional services

- [ ] **Monitoring:** New Relic
- [ ] **Analytics:** Google Analytics
- [ ] **Assignment distribution and collection:** nbgrader
- [ ] **Grading:** nbgrader

2.5.9 8. Deploy with Ansible

- [ ] **Deploy:** Run `ansible-playbook` for the main deployment.

```
$ ansible-playbook -i hosts deploy.yml
```

2.5.10 9. Verify deployment and reload supervisor

- [ ] **Verify:** SSH into the server:

```
$ ssh root@{hostname}
```

substituting your hostname for {hostname}. For example, `ssh root@jupyter.org`.

2.5.11 10. JupyterLab

2.6 Acknowledgment

Prof. Brian Granger, Cal Poly San Luis Obispo, authored this repository's code to deploy JupyterHub for the course, DATA 301, "Introduction to Data Science."

Thank you Brian Granger and Jonathan Fredric, co-author of an earlier code prototype, for sharing their work.

2.7 Repository Contents

2.7.1 Ansible application

ansible.cfg

Custom configuration settings for the Ansible application

- We use to customize root access, root privileges, and ssh connection length.

ansible-conda

Git submodule for `ansible-conda` application

2.7.2 Inventory (Ansible)

hosts.inventory

Inventory file of servers (hosts) being managed by Ansible

2.7.3 Playbooks (Ansible)

deploy.yml (a.k.a. site.yml in Ansible jargon)

deploy_formgrade.yml

saveusers.yml

2.7.4 Variables (Ansible)

group_vars

host_vars

2.7.5 Roles (Ansible)

bash

common

cull_idle

formgrade

jupyterhub

nbgrader

newrelic

nginx

python

r

saveusers

supervisor

2.7.6 Development

.gitignore

.gitmodules

LICENSE

README.md

2.7.7 Documentation

readthedocs.yml

Settings for readthedocs services

docs

Directory containing sphinx documentation for the reference deployment.