Python repository setup

Nexus Installation

1. Install docker:
2. create a directory for nexus "mkdir -p /data/2/nexus-repo/"
3. provide full permission to /data/2 directory "sudo chmod -R 777 /data/2/"
4. pull nexus docker image and run run it.

```
sudo docker run -itd -p 8081:8081 -p 8082:8082 -p 8083:8083 -p 8080:
8080 --name nexus3 -v /data/2/nexus-repo/:/nexus-data sonatype/nexus3""
docker ps -a
```

configure nexus container to restart automatically when it is stopped because of error.

```
docker update --restart unless-stopped nexus3
```

to check the logs

```
docker logs nexus3
```

To view the initial admin password

```
docker exec -it nexus3 cat /nexus-data/admin.password
```

Configuring the python pip proxy repository

1. Create a blob store for python repository.

- Go to Server Administration and configuration page
- Click on create blob store
- Provide following details
  Type : file
  Name : Name of the repository
  Path : give the relative path for repository

2. in Administration page click on manage repository

- Click on create repository
- Select pypi (proxy)
  provide following details
  Name : argoid_pypi_proxy
  Remote Storage : https://pypi.org
  blob Storage : argoid_py_blob

Configuring python hosted repository

a. in Administration page click on manage repository

- click on create repository
- select pypi (hosted)
  provide following details
  Name : argoid_pypi_hosted
  blob Storage : argoid_py_blob

Set global environment variables in Jenkins

1. Manage Jenkins > Configure System > Global properties > Environment variables
2. Set below variables
    a. NEXUS_PYPI_REPO = argoid_pypi_hosted

Edit the file `/etc/pip.conf` add the artifactory urls as bellow

```
[global]
index=http://repo1.internal.argoid.com:8081/repository/argoid_pypi_proxy
index-url=http://repo1.internal.argoid.com:8081/repository
/argoid_pypi_proxy/simple
trusted-host=repo1.internal.argoid.com
```

- index-url is used by pip install
- index is used by pip search

Create file .pypirc in user home directory with following content

[distutils]
index-servers =
pypi
[pypi]
repository: http://repo1.internal.argoid.com:8081/repository/argoid_pypi_hosted/
username: admin
password: Argoid@2021

reference url : https://help.sonatype.com/repomanager3/formats/pypi-repositories

Create file .netrc in user home directory with following content

machine repo1.internal.argoid.com

login argoid

password <argoid_user_password>

To view config file

pip config -v list

creating python package

Structure of the package for python3

packaging_tutorial/
LICENSE
pyproject.toml
README.md
setup.cfg
src/
example_package/
init.py
example.py
tests/

Structure of the package for python2

helloworld/

helloworld/
init.py
helloworld.py
helpers.py

tests/
helloworld_tests.py
helpers_tests.py

.gitignore
LICENSE
README.md
requirements.txt
setup.py

Example for **setup.cfg** file

[metadata]
name = example-pkg-YOUR-USERNAME-HERE
version = 0.0.1
author = Example Author
author_email = author@example.com
description = A small example package
long_description = file: README.md
long_description_content_type = text/markdown
url = https://github.com/pypa/sampleproject
project_urls =
Bug Tracker = https://github.com/pypa/sampleproject/issues
classifiers =
Programming Language :: Python :: 3
License :: OSI Approved :: MIT License
Operating System :: OS Independent

[options]
package_dir =
= src
packages = find:
python_requires = >=3.6

[options.packages.find]
where = src

Please refer to following link for more details
https://packaging.python.org/tutorials/packaging-projects/

Fetching the packages from repository

pip install <package name>

or

pip install -r requirements.txt

ex: requirement.txt

```
BeautifulSoup==3.2.0
Django==1.3
Fabric==1.2.0
Jinja2==2.5.5
PyYAML==3.09
Pygments==1.4
SQLAlchemy==0.7.1
South==0.7.3
amqplib==0.6.1
```

Building and uploading the package to hosted repository for both python2 and python3

1. python -m pip install --upgrade pip
2. python -m pip install --upgrade build
3. python -m build
4. python -m pip install --upgrade twine
5. python -m twine upload --repository testpypi dist/*

Managing Python Virtual environment

To create venv run the bellow command

python3.8 -m venv virtual/environment

source virtual/environment/bin/activate

Priority of config file

```
[bala@argoid-infra-jenkins-1 ~]$ source virtual/environment/bin/activate
(environment) [bala@argoid-infra-jenkins-1 ~]$ pip config -v list
For variant 'global', will try loading '/etc/xdg/pip/pip.conf'
For variant 'global', will try loading '/etc/pip.conf'
For variant 'user', will try loading '/home/bala/.pip/pip.conf'
For variant 'user', will try loading '/home/bala/.config/pip/pip.conf'
For variant 'site', will try loading '/home/bala/virtual/environment
/pip.conf'
```

reference link : https://pip-python3.readthedocs.io/en/latest/user_guide.html

some useful pip syntaxes

1. pip list
2. pip list --outdated
3. pip show setuptools
4. pip search "query"