Hydroponic Automation

A Automize code to automatically bundle dependencies from requirements.txt and make them available in your PYTHONPATH.

Requires Serverless >= v1.12

Install

```
sls plugin install -n serverless-python-requirements
```

This will automatically add the plugin to your project's package.json and the plugins section of its serverless.yml. That's all that's needed for basic use! The plugin will now bundle your python dependencies specified in your requirements.txt or Pipfile when you run sls deploy.

For a more in depth introduction on how to use this plugin, check out this post on the Serverless Blog

If you're on a mac, check out these notes about using python installed by brew.

Cross compiling!

Compiling non-pure-Python modules or fetching their manylinux wheels is supported on non-linux OSs via the use of Docker and the docker-lambda image. To enable docker usage, add the following to your serverless.yml:

```
custom:
  pythonRequirements:
    dockerizePip: true
```

The dockerizePip option supports a special case in addition to booleans of 'non-linux' which makes it dockerize only on non-linux environments.

To utilize your own Docker container instead of the default, add the following to your serverless.yml:

```
custom:
  pythonRequirements:
    dockerImage: <image name>:tag
```

This must be the full image name and tag to use, including the runtime specific tag if applicable.

Alternatively, you can define your Docker image in your own Dockerfile and add the following to your serverless.yml:

```
custom:
  pythonRequirements:
    dockerFile: ./path/to/Dockerfile
```

With Dockerfile the path to the Dockerfile that must be in the current folder (or a subfolder). Please note the dockerImage and the dockerFile are mutually exclusive.

To install requirements from private git repositories, add the following to your serverless.yml:

```
custom:
  pythonRequirements:
    dockerizePip: true
    dockerSsh: true
```

The dockerSsh option will mount your $HOME/.ssh/id_rsa and $HOME/.ssh/known_hosts as a volume in the docker container. If your SSH key is password protected, you can use ssh-agent because $SSH_AUTH_SOCK is also mounted & the env var set. It is important that the host of your private repositories has already been added in your $HOME/.ssh/known_hosts file, as the install process will fail otherwise due to host authenticity failure.

You can also pass environment variables to docker by specifying them in dockerEnv option:

```
custom:
```

```
  pythonRequirements:
    dockerEnv:
      - https_proxy
```

:checkered_flag: Windows notes

Pipenv support :sparkles::cake::sparkles:

If you include a `Pipfile` and have `pipenv` installed instead of a `requirements.txt` this will use `pipenv lock -r` to generate them. It is fully compatible with all options such as `zip` and `dockerizePip`. If you don't want this plugin to generate it for you, set the following option:

```
custom:
  pythonRequirements:
    usePipenv: false
```

Poetry support :sparkles::pencil::sparkles:

NOTE: Only poetry version 1 supports the required `export` command for this feature. As of the point this feature was added, poetry 1.0.0 was in preview and requires that poetry is installed with the --preview flag.

TL;DR Install poetry with the `--preview` flag.

If you include a `pyproject.toml` and have `poetry` installed instead of a `requirements.txt` this will use `poetry export --without-hashes -f requirements.txt` to generate them. It is fully compatible with all options such as `zip` and `dockerizePip`. If you don't want this plugin to generate it for you, set the following option:

```
custom:
  pythonRequirements:
    usePoetry: false
```