Packaging and Deployment

Paul O'Fallon

@paulofallon

Overview

Publish our command line app!

Automatically check for updates

Automate our deployments

Create a Docker image for our CLI

Publishing Twine to npm

Two simple commands:

```
npm login
```

```
npm publish
```

However...

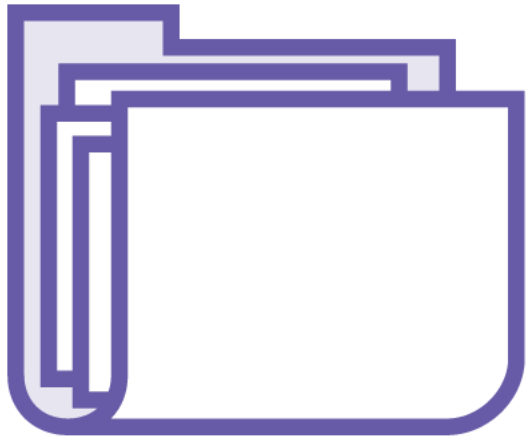
we need a scoped package:

```
@pofallon/twine
```

“scope”

Scoped Packages in npm

Private packages



Grouping

Namespacing

Executing Binaries Directly with npx

Typical module usage

```
npm install -g @pofallon/twine
twine lookup users paulofallon
```

With npx

```
npx @pofallon/twine lookup users paulofallon
```

Demo

Define twine as a scoped package

Fix our commands

Publish our application!

Checking for Updates

Automatically check for a new version

Notify the user periodically, but...

Don't annoy them!

Automating Our Releases

Demo

Add support for an update check

Setup TravisCI integration

Shipping Our CLI as a Docker Image

1. **Build (and publish) a Docker image**

```
docker build -t pofallon/twine .
```

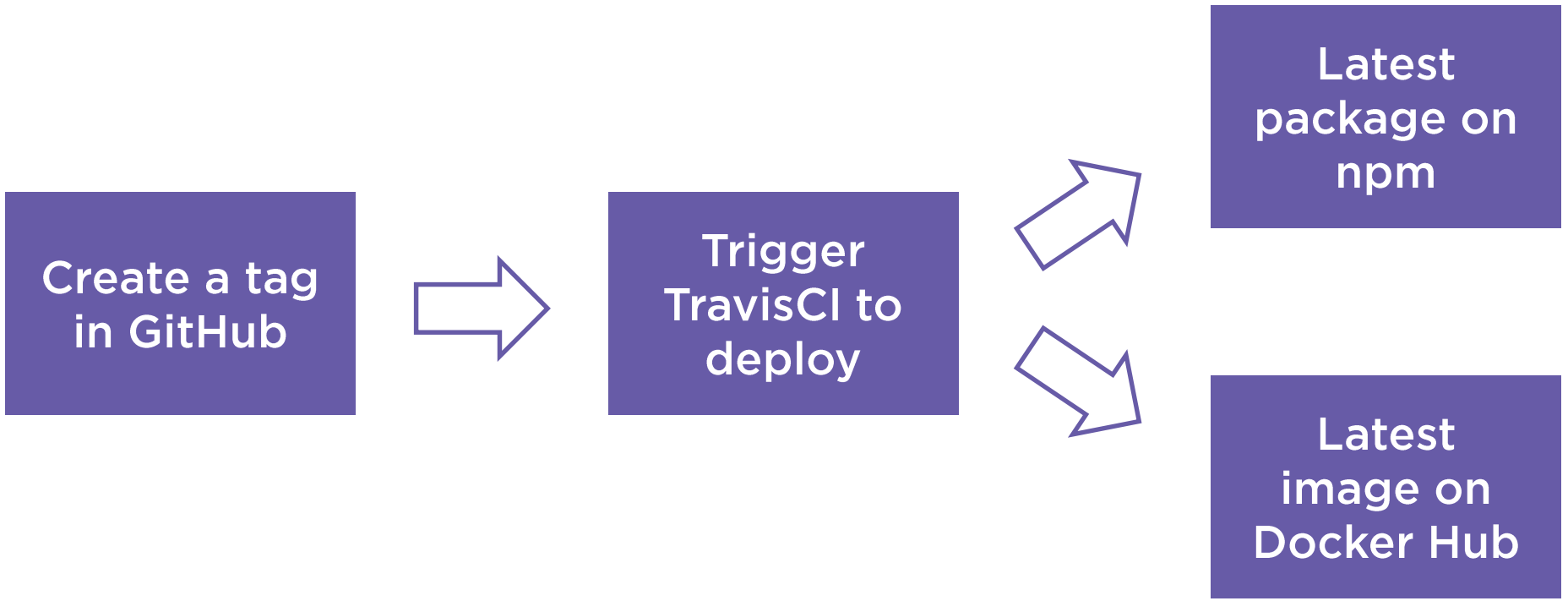
2. **Create an environment file (`env.list`) with our credentials**

```
TWINE_CONSUMER_KEY=0tq…
TWINE_CONSUMER_SECRET=O5M…
TWINE_ACCOUNT_KEY=242…
TWINE_ACCOUNT_SECRET=4YS…
```

3. **Invoke the Docker image with these credentials**

```
docker run --env-file env.list pofallon/twine lookup users paulofallon
```

Updating our Deployment Process



Demo

Create a Docker image for our CLI

Add this to our deployment automation

Summary

- Updated to a scoped package name
- Published Twine to npm
- ... and automated it with TravisCI!
- Enabled user update notifications
- Added a Docker-ized version of our CLI