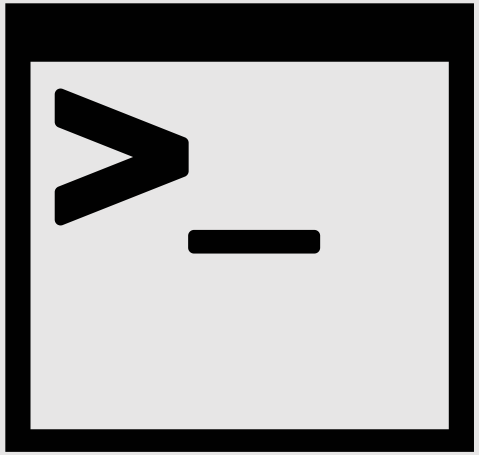
Packaging your D app

DConf’19 – Lionello Lunesu – Enuma Technologies

@LioLunesu

So you want to package your D app?



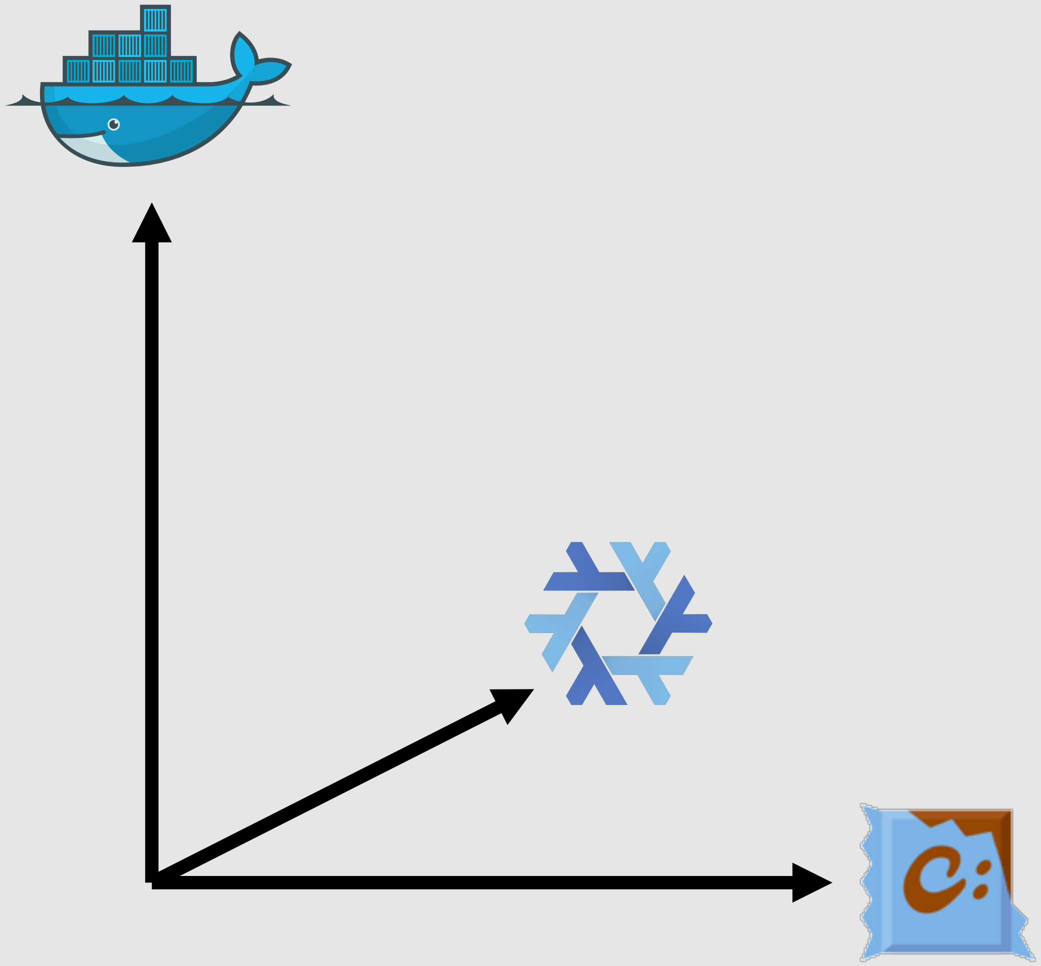
> Cloud

Google Cloud Platform
amazon web services
Microsoft Azure
IBM Bluemix™
DigitalOcean
OPENSHIFT® by Red Hat®
Alibaba Cloud aliyun.com

> CLI / GUI

macOS
DUB
yum
yellowdog updater modified
dpkg
Windows

After Principal Component Analysis:



> Docker

- One *container* for local testing and cloud deployments
- Contains *all* dependencies
- Convenient for web services
  - and CLI tutorials
- Less convenient for local CLI tools that access state
- Containers can be *pushed* to Docker Hub registry
- Supported by most cloud providers
- Stateless containers are ready to be auto-scaled

>> Dockerfile

```
FROM dlanguage/dmd
COPY hello.d .
RUN dmd hello.d
CMD ["./hello"]
```

```
$ docker build .
Successfully built 62525b23b4f5
```

```
$ docker run 62525b23b4f5
Hello world
```

>> Dependency hell

- Container is ~560MiB. Does it matter?
- Built container still has all build tools; not needed for running
- What else does the base image contain?
- What base image to use instead of `dlanguage/dmd` ?

- Best base image is `scratch`, 0-bytes
    - But must copy all dependencies too
    - Recursive `ldd`?

>> Multi-stage Dockerfile

```
FROM dlanguage/dmd AS build-stage
RUN apt-get update && apt-get -y install \
  git \
  python-setuptools

# Install dockerize from GIT (or COPY from local submodule)
RUN git clone --depth=1 https://github.com/larsks/dockerize.git
WORKDIR dockerize
RUN python setup.py install

WORKDIR /
COPY hello.d .
RUN dmd hello.d
RUN dockerize -o /output -n /hello

FROM scratch
COPY --from=build-stage /output /
CMD ["./hello"]
```

>> Multi-stage Dockerfile with dependencies

```
FROM dlanguage/dmd AS build-stage
RUN apt-get update && apt-get -y install \
  git \
  python-setuptools

RUN git clone --depth=1 https://github.com/larsks/dockerize.git
WORKDIR dockerize
RUN python setup.py install

WORKDIR /
COPY dub.json .
RUN dub build ; true
COPY hello.d source/
RUN dub build
RUN dockerize -o /output -n /hello

FROM scratch
COPY --from=build-stage /output /
CMD ["./hello"]
```

> Nix

- The Purely Functional Package Manager
- Cross-platform: *nix, macOS, FreeBSD
  - Even works in WSL!*
- One *package* for local testing and cloud deployments
- Nix expressions describe how to build a package from *source*
- All dependencies in `/nix/store`, **no global install**
- Paths based on hash of their *inputs*, not name or version
- Packages can be merged into the *nixpkgs* repository
- Automatic runtime dependencies

```
$ curl https://nixos.org/nix/install | sh
```

>> Nix derivation

```
with (import <nixpkgs> {});

stdenv.mkDerivation {
  name = "hellod";
  nativeBuildInputs = [ dmd ];
  src = ./.;
  buildPhase = ’’
    dmd hello.d
  ’’;
  installPhase = ’’
    mkdir -p $out/bin
    cp hello $out/bin
  ’’;
}
```

```
$ nix-env -if ./hellod.nix
```

>> Dub with Nix

noun.ly/boy

- Nix builds are pure, deterministic
  - no arbitrary internet access
- Need derivations for all dependencies
- For each dependency ...
  - Checkout the required version from its GIT repository
  - `dub build or dub add-local`

- Use `dub2nix` to generate `dub.selections.nix` from json

>> Dub-aware Nix derivation

```
with (import ./mkDub.nix {});       # from dub2nix repo

mkDubDerivation {
  version = "0.1.0";
  src = ./.;
  deps = [];                        # defaults to dub.selections.nix
}

$ nix-env -if https://github.com/lionello/dub2nix/archive/master.zip

$ dub2nix save
```

>> dub.selections.nix

```nix
# This file was generated by https://github.com/lionello/dub2nix v0.1.0
[ {
  fetch = {
    type = "git";
    url = "https://github.com/vibe-d/vibe.d.git";
    rev = "v0.8.5";
    sha256 = "0s1caxqmq2497j5x8h06f44nr597h9zac8qxxml953lkaqkhbzgy";
  };
} {
  fetch = {
    type = "git";
    url = "https://github.com/wilzbach/stdx-allocator.git";
    rev = "v2.77.5";
    sha256 = "03av8zp5p6vf6fg005xbmwnjfw96jyrr4dcj4m56c4a3vx7v72pk";
  };
} ]
```

>> Docker with Nix

```
with (import <nixpkgs> {});

let
  hellod = (import ./hellod.nix);
in
  dockerTools.buildImage {
    name = hellod.name;
    config = {
      Cmd = ["${hellod}/bin/hello"];
    };
  }
```

> Chocolatey

- The package manager for Windows
- Uses Nuget packages
  - optional install/update/uninstall powershell scripts
- Packages can be uploaded to the Chocolatey Community Feed

```
> Set-ExecutionPolicy Bypass -Scope Process -Force;
iex ((New-Object
System.Net.WebClient).DownloadString('https://chocol
atey.org/install.ps1'))
```

>> Chocolatey Nuspec

```
<?xml version="1.0" encoding="utf-8"?>
<package xmlns="http://schemas.microsoft.com/packaging/2015/06/nuspec.xsd">
  <metadata>
    <id>hellod</id>
    <version>0.1.0</version>
    <authors>Lionello Lunesu</authors>
    <description>Simple package for hello world in D</description>
    <!--<dependencies>
      <dependency id="chocolatey-core.extension" version="1.1.0" />
    </dependencies>-->
  </metadata>
  <files>
    <file src="tools\hello.exe" target="tools" />
  </files>
</package>
```

>> Chocolatey overview

```
> choco new

> choco pack

> choco install hellod

> choco uninstall hellod
```

> Summary

	Docker	Nix	Chocolatey
Build OS	Linux, macOS	Linux, FreeBSD, macOS	Windows, Linux (Mono)
Target OS	Linux	Linux, FreeBSD, macOS	Windows
Target	Cloud service	Cloud, CLI, GUI	CLI, GUI
Input	Dockerfile	Nix derivation	Nuspec
Artifact	Image (.tar.gz)	Nix store path (.nar)	.nupkg
Runtime Dependencies	Explicit	Automatic	Explicit
Package registry	Docker Hub	Nixpkgs repo	Community Feed
Build command	docker build	nix-build	choco pack
Install/run command	docker pull / docker run	nix-env / nix-shell	choco install

References

- https://docs.docker.com/develop/develop-images/multistage-build/
- https://github.com/larsks/dockerize
- https://nixos.org/nix/
- https://github.com/lionello/dub2nix
- https://chocolatey.org

noun.ly/file

> Thank you!

Lionello Lunesu – Enuma Technologies

lio@enuma.io

@LioLunesu