SOLVING THE DISTRIBUTED NAMING PROBLEM (!) IN APPLIED PACKAGE MANAGEMENT

IPFS Camp

Eric Myhre
@warpfork

I have software I want to build!

use!

I have software I want to ~~build~~!

compile!

okay, *patch*!

then compile and use!

and share..!

(there are many different ways to go about this)

What’s tricky isn’t building.

We have build tools.

What’s tricky is
integrating.

What’s tricky is connecting
different build processes together.

What’s tricky is composing systems.

What’s tricky is
integrating.

The best way to figure out how to *integrate* something is… to break it apart, first.

What is a “build”?

Files go in…

```
{
  "inputs": {
    "/": "ipfs:Qm6q7G4hWr26q7G4hWTa5Lf89p"
    "/app/gcc": "ipfs:Qmr283FpTaSL883FpL8"
    “/task/src”: “git:7c554c3ae7bee8e3b42”
  }
}
```

What is a “build”?

Files go in…

Files go out…

```
{
  "inputs": {
    "/": "ipfs:Qm6q7G4hWr26q7G4hWTa5Lf89p"
    "/app/gcc": "ipfs:Qmr283FpTaSL883FpL8"
    “/task/src”: “git:7c554c3ae7bee8e3b42”
  },

  "outputs": {
    "/task/out": {"packtype": "ipfs"}
  }
}
```

What is a “build”?

Files go in…

Files go out…

An action in the middle.

```
{
  "inputs": {
    "/": "ipfs:Qm6q7G4hWr26q7G4hWTa5Lf89p"
    "/app/gcc": "ipfs:Qmr283FpTaSL883FpL8"
    “/task/src”: “git:7c554c3ae7bee8e3b42”
  },
  "action": {
    "exec": [
      "/app/gcc/bin/gcc",
      "/task/src/cool.c",
      "-o", "/task/out/cool.exe" ]
  },
  "outputs": {
    "/task/out": {"packtype": "ipfs"}
  }
}
```

```
{
  "inputs": {
    "/": "ipfs:Qm6q7G4hWr26q7G4hWTa5Lf89p"
    "/app/gcc": "ipfs:Qmr283FpTaSL883FpL8"
    “/task/src”: “git:7c554c3ae7bee8e3b42”
  },
  "action": {
    "exec": [
      "/app/gcc/bin/gcc",
      "/task/src/cool.c",
      "-o", "/task/out/cool.exe" ]
  },
  "outputs": {
    "/task/out": {"packtype": "ipfs"}
  }
}
```

What can we do if we…

Do this with IPLD?


{
"inputs": {
"/": "ipfs:Qm6q7G4hWr26q7G4hWTa5Lf89p"
"/app/gcc": "ipfs:Qmr283FpTaSL883FpL8"
“/task/src”: “git:7c554c3ae7bee8e3b42”
},
"action": {
"exec": [
"/app/gcc/bin/gcc",
"/task/src/cool.c",
"-o", "/task/out/cool.exe"]
},
"outputs": {
"/task/out": {"packtype": "ipfs"}
}
}
“Computation-addressable”
{
"formulaID": "QmVdAD3v91JWPjZXGkTKD6",
"exitCode": 0,
"results": {
"/task/out": "ipfs:QmD9AjmrdCY4QUquhiQ"
}
}

This is a useful building block.

- Share a complete build instruction easily.
- Rebuild later, or use in bug reports, etc.
- Audit logs; explanations of builds that happened previously.
- Check output hashes to easily check reproducibility.
- General forcing-function for sanity!

Now let’s try to integrate.

Now let’s try to integrate.

- Cool, we can address builds we’ve already drafted.

  ...

Now let's try to integrate.

- Cool, we can address builds we've already drafted.
- But nobody likes copy-pasting hashes;
- We'll need a way to describe multiple steps and their relationships in advance to be productive.

Now let's try to integrate.

"describe multiple steps and relate them in advance".....

We have a **naming problem**.

So break it down again.

Distributed naming problems are hard.

So let’s try to break down the problem into component parts…

(until suddenly none of them look like “naming” anymore!)

and then integrate again afterwards.

So break it down again.

- Naming for labeling (still immutable!)
- Naming for making a graph (can be locally-scoped!)
- Naming for looking for ‘updates’
- Naming for grouping and categorization
- Naming for reputation
- ...

So break it down again.

- **Naming for labeling (still immutable!)**
- **Naming for making a graph (can be locally-scoped!)**
- Naming for looking for ‘updates’
- Naming for grouping and categorization
- Naming for reputation
- …

<u>Some of these use-cases are easier than others.</u>

So break it down again.

- Naming for labeling (still immutable!)
- Naming for making a graph (can be locally-scoped!)
- **Naming for looking for ‘updates’**
- Naming for grouping and categorization
- Naming for reputation
- …

Some of these use-cases entangle application logic.

locally scoped naming is easy

Lesson for any distributed system design:

If you can turn a *distributed* naming problem into a *locally scoped* naming problem, do it. It's easier. (Then try to make it composable.)

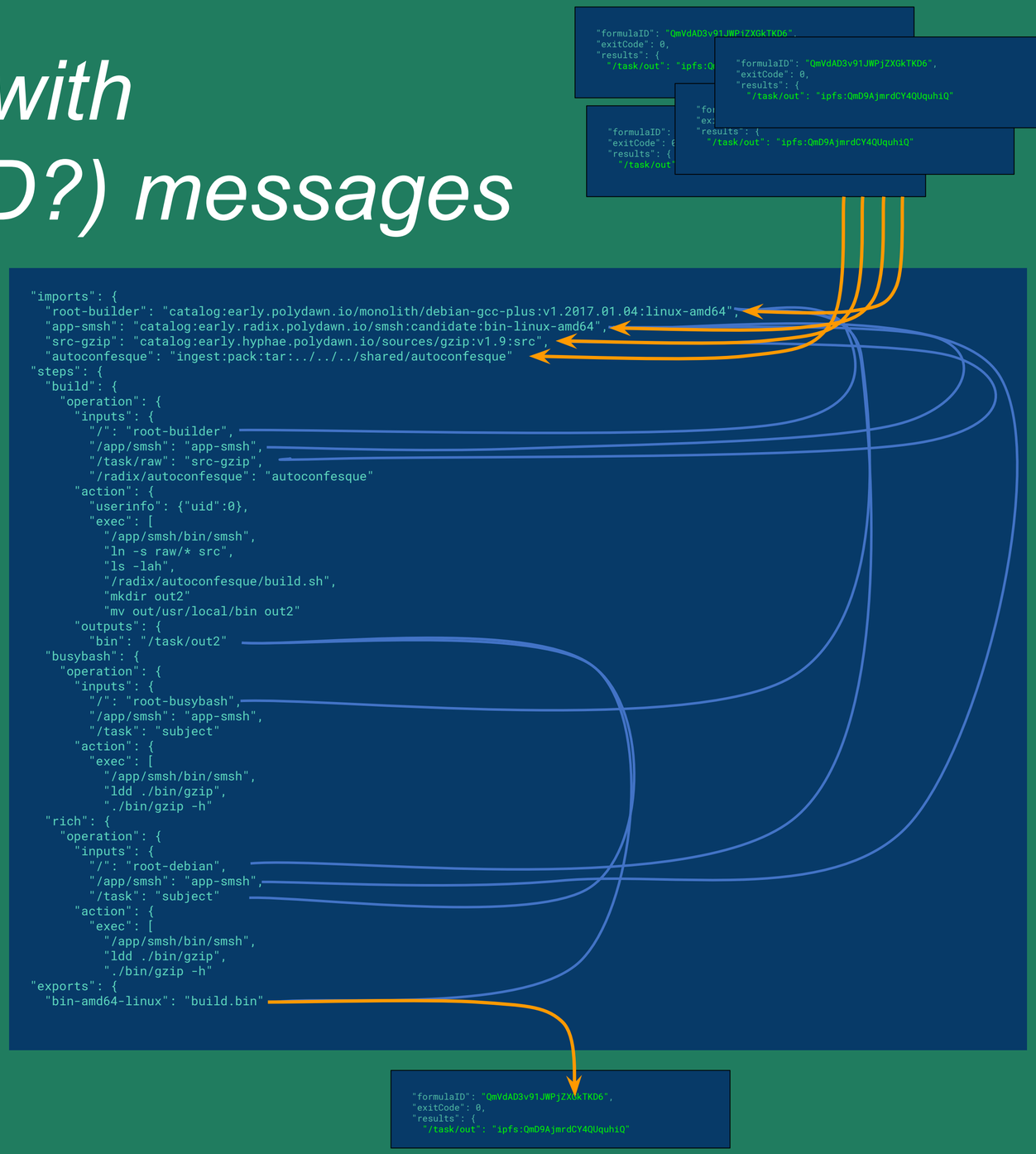
locally scoped naming is easy

… so what do we do when we want to collaborate and integrate with other people and other authors?

connect things with content-addressable (IPLD?) messages

… so what do we do when we want to collaborate and integrate with other people and other authors?

Can we keep extending the defn of “locally scoped” gradually?



Docs, more docs, and prototypes

repeatr.io

github.com/polydawn/timeless