NoOps:
More Dev, Less Ops

Roy Miller, Devellocus

Southern Fried Agile 2014

We want

Discipline

We track what we do for the team's benefit

Focus

We get to spend most of our time making software

Transparency

Everybody can see where things stand at any point

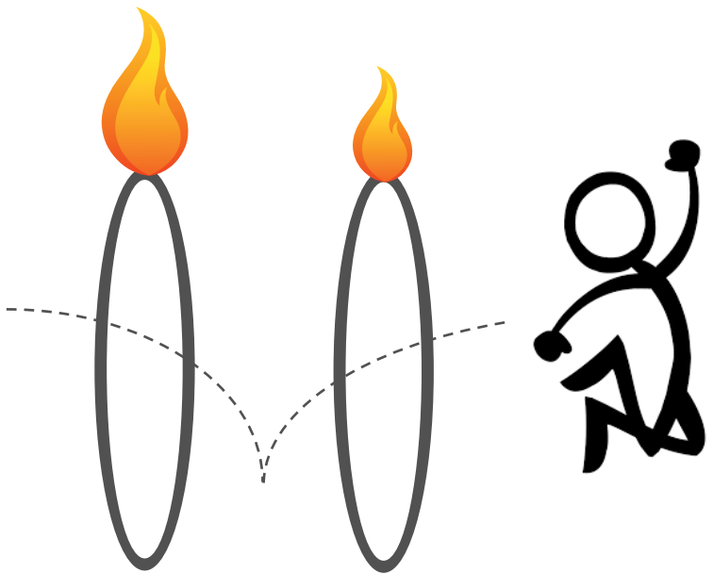
Consistency

We do the same stuff in the same way, every time

Can we please make SOFTWARE?!

```
$ atq | sed 's_\([0-9]\{1,8\}
  \).*_\1_g' | xargs atrm
```

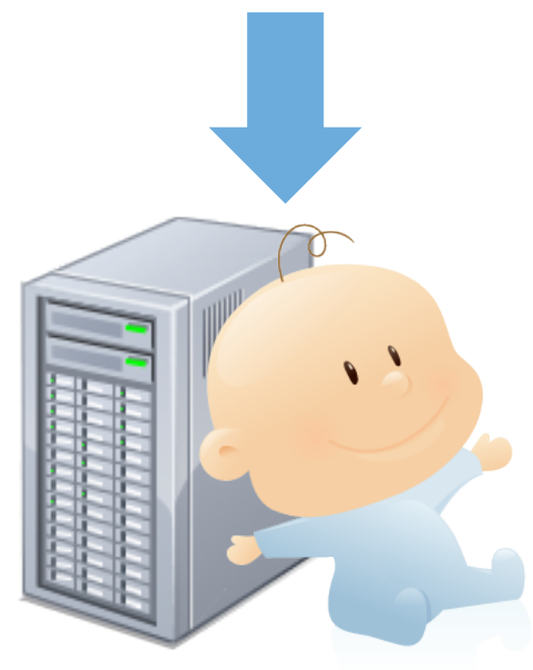
Small(er) Org



Big Org

The promise

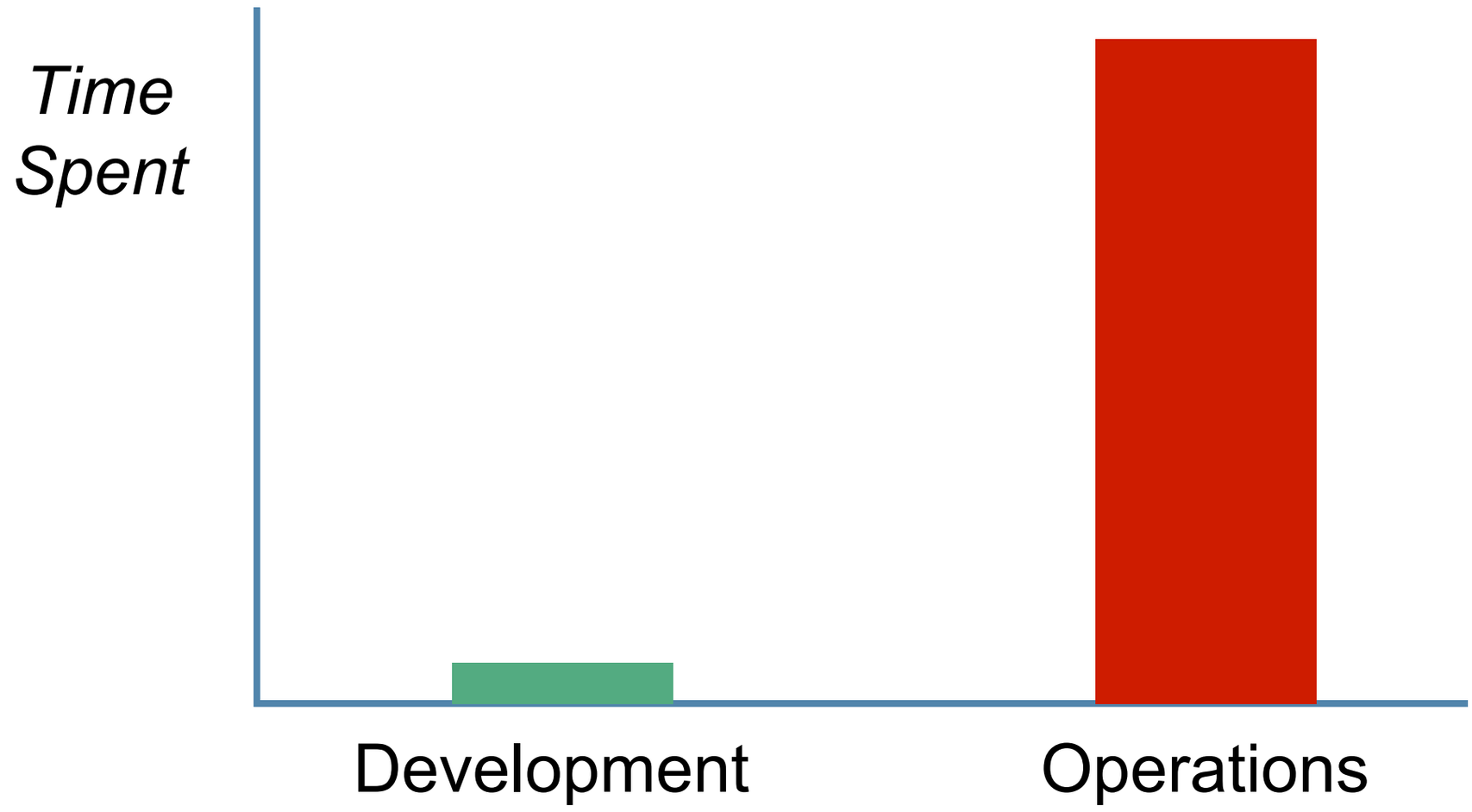
Development ❤ Operations



Reality

Development Operations

DevOps ➡ DevOps



Dev

Ops

NoOps

[noh-ops]

noun

The bare minimum Ops required for a team to create great software at maximum responsible speed

How you can …

Automate Everything™

The NoOps Principle

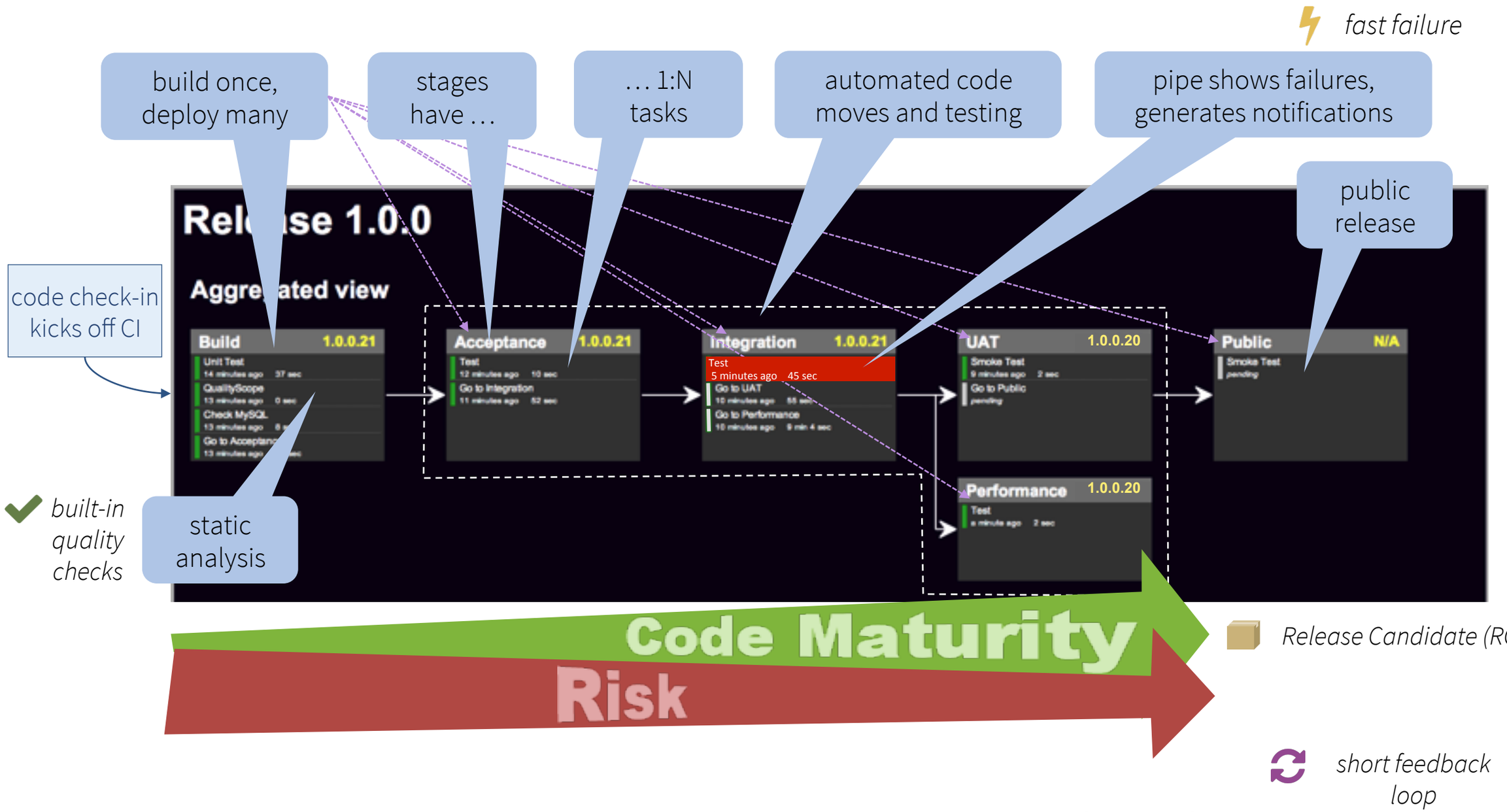
An app knows its stuff, the platform doesn't

App	Platform
Knows build file locations, deploy targets, etc.	Moves things to environments, runs things

Developers own their app

Setting up new pipes is easy

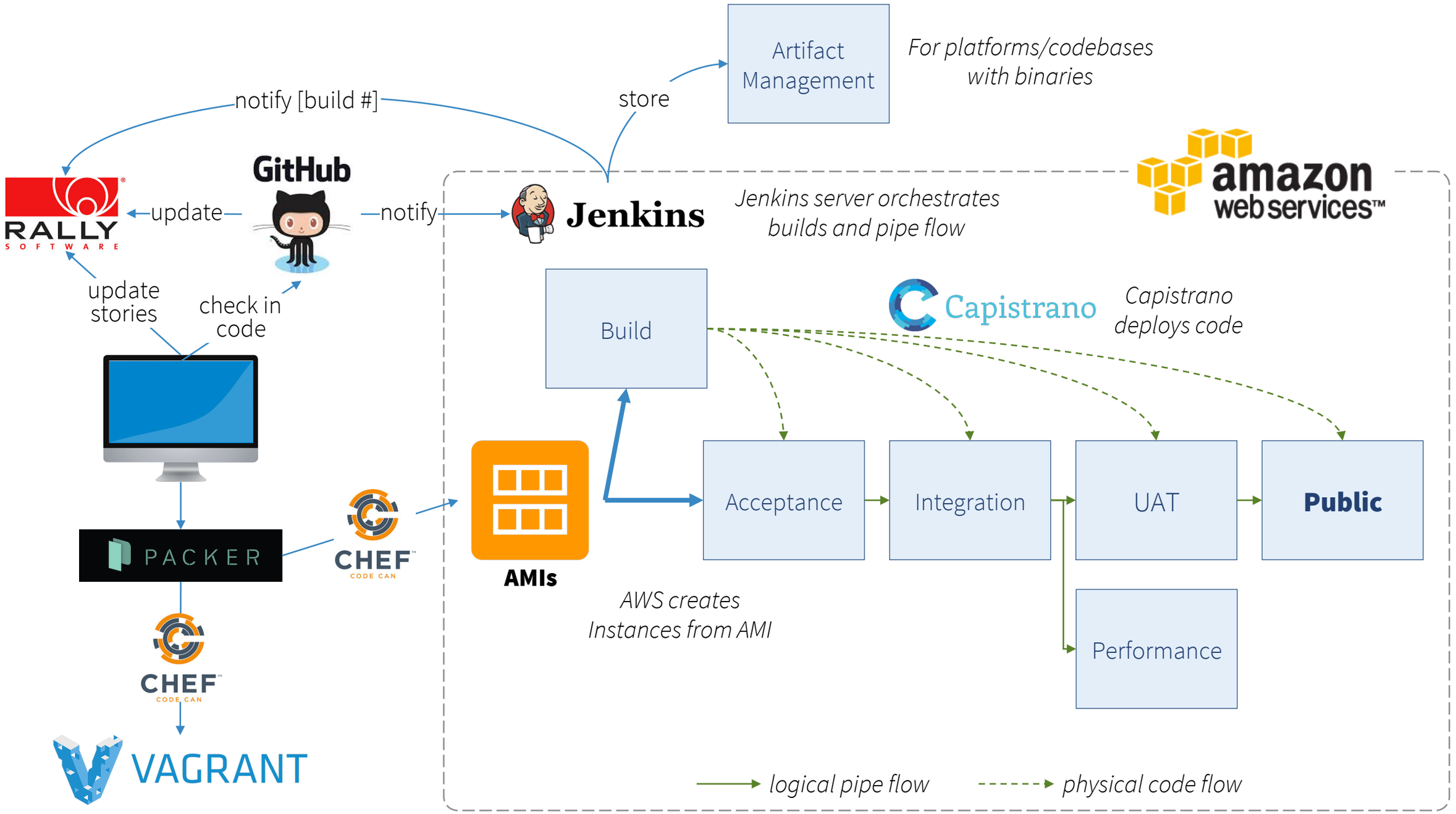
Continuous Delivery



*A **clear, visible information radiator** to show the state of the software at any point*

The TechStew

*Initially targeting **Node/Rails on Linux** … but we need Windows, too*



Before we dive in ...

I'm no sysadmin ...
I'm sure I've done unsavory things

This isn't a fancy Rails setup...
The app is a basic blog app

It's as secure as <u>we</u> need it ...
We're maturing as we go

I'm not a Chef ...
There might be better ways to bake

It's command line for now ...
Ultimately a Rails app, most likely

You may have limits ...
You might not be able to do some (or any) of this

Before we dive in ...

No Animals

Were Harmed

One pipe definition

`stages [...]`

additional installs via Chef recipes

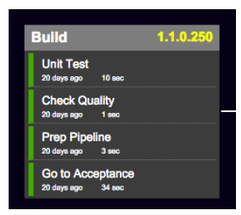
```
{
   :name=>:build,
   :display_name=>”Build",
   :role=>:ci,
   :servers=>[
     { :roles=>[:ci,:web,:app,:db], :instance_type=>"m3.medium", :installs=>[:nginx,:mysql] }
   ],
   :jobs=>[
     { :sequence=>1, :name=>"unit-test", :task_name=>"Unit Test",
       :build_on_scm_push=>true, :notify=>true,
       :update_scm_status=>true, :publish_to_github=>true,
       :create_delivery_pipeline_version=>true,
       :notify_rally=>true },
     { :sequence=>2, :name=>"check-quality", :task_name=>"Check Quality",
       :commands=>"echo '[placeholder] run static analysis'" },
     { :sequence=>3, :name=>"check-pipeline", :task_name=>"Check Pipeline" },
     { :sequence=>4, :name=>"deploy-uat", :task_name=>"Go to UAT", :notify=>tru
   ],
   :next_stage=>[:uat]
}
```

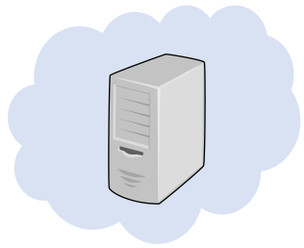
stages have 1:N servers with 1:N roles

chain stages together so Jenkins can chain jobs

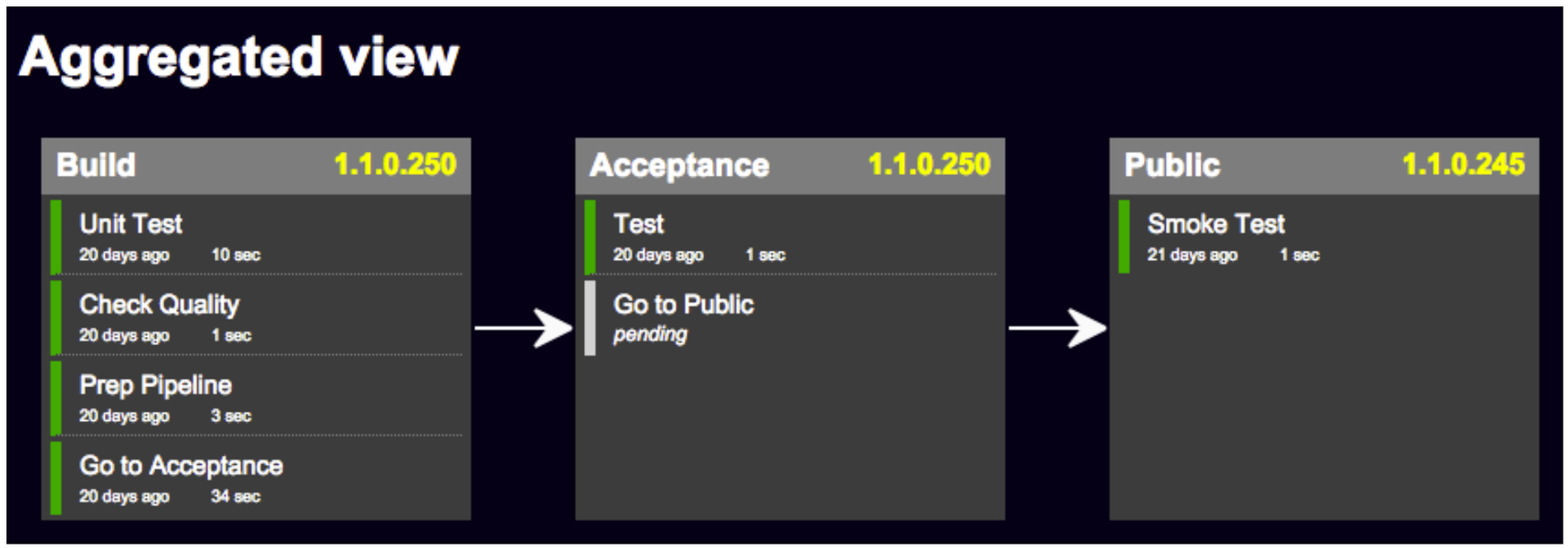
define Jenkins jobs, potentially with custom commands

Pipes, Stages, and Boxes

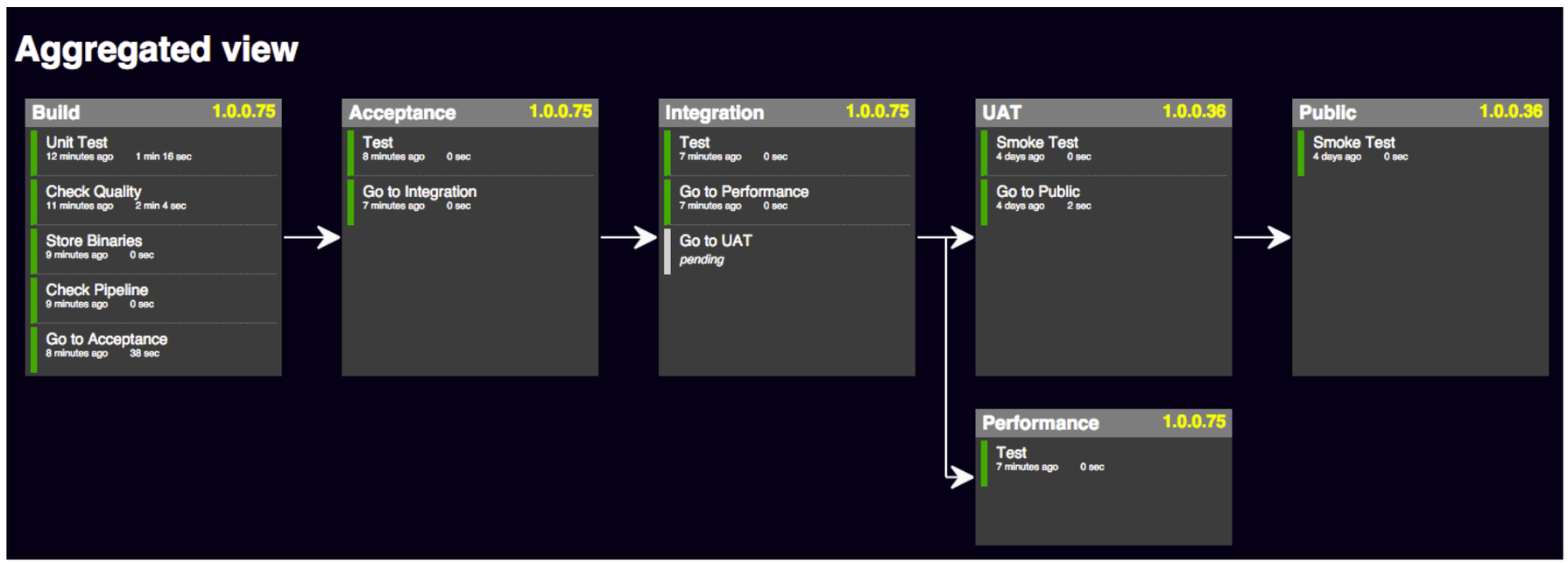




Pipes can be short ...



or long ...



Create and converge machines with Chef in **local mode** (i.e., Chef Metal)

```
23    def create_stages
24      descriptor = ClusterDescriptor.new(:ready)
25      descriptor_file_name = descriptor.create_for_stages
26
27      project_root = File.join(File.dirname(File.dirname(File.expand_path(__FILE__))))
28      pipeline_path = File.join(project_root)
29      metal_command = "cd #{pipeline_path} && chef-client -z #{descriptor_file_name}"
30      status = self.system_execute(metal_command)
31      status
32    end
```

ChefMetalClusterMaker.rb

Running is quick and painless

```
[10-18-2014 9:51] roymiller@darkcastle: ~/Workspaces/pipefitter on metal * [9:51:36] L:1 N:4
± bundle exec bin/pipefitter generate pipe sample

--------------------------------------------------------------------------------

[Pipefitter] 09:52:12 -> Started at 2014/10/18 09:52:12
[Pipefitter] 09:52:12 -> Defining stages
[Pipefitter] 09:52:12 -> All stages defined: build, uat, public
[Pipefitter] 09:52:12 -> Create stages
Starting Chef Client, version 11.16.0
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2014-10-18T09:52:21-04:00] WARN: Node darkcastle.local has an empty run list.
Converging 1 resources
Recipe: @recipe_files::/Users/roymiller/Workspaces/pipefitter/cluster_create.rb
  * machine_batch[default] action ready
    - creating machine sample-build-ci-web-app-db on fog:AWS:298725748436:us-east-1
    -   flavor_id: "m3.medium"
    -   key_name: "pipeline"
    - creating machines sample-uat-web-app-db, sample-public-web-app, sample-public-db on fog:AWS
:us-east-1
    -   flavor_id: "t2.small"
    -   key_name: "pipeline"
    -   groups: ["pipeline"]
    -   groups: ["pipeline"]
    -   image_id: "ami-80bc6ee8"
    -   image_id: "ami-4aa27022"

It's ready to use, once you copy some files and configure your app.
Pipefitter saved you some work by creating these Capistrano stage files:

  * output/build.rb
  * output/public.rb
  * output/uat.rb

Here's what you need to do:

  * Copy those files to the config/deploy dir inside your app
  * Copy the Pipeline file to your app root (if you haven't already)
  * Set build 'host' in config->database.yml to: ip-172-31-29-195.ec2.internal
  * Set uat 'host' in config->database.yml to: ip-172-31-29-185.ec2.internal
  * Set public 'host' in config->database.yml to: ip-172-31-29-184.ec2.internal
  * Set your existing GitHub webhook to: http://jenkins:dvl_123@54.172.116.2:8080/github-webhook/

[Pipefitter] 10:01:37 -> Finished at 2014/10/18 10:01:37
[Pipefitter] 10:01:37 -> Total runtime was 9 minutes and 24 seconds
```

The **tool orchestrates** pipe creation

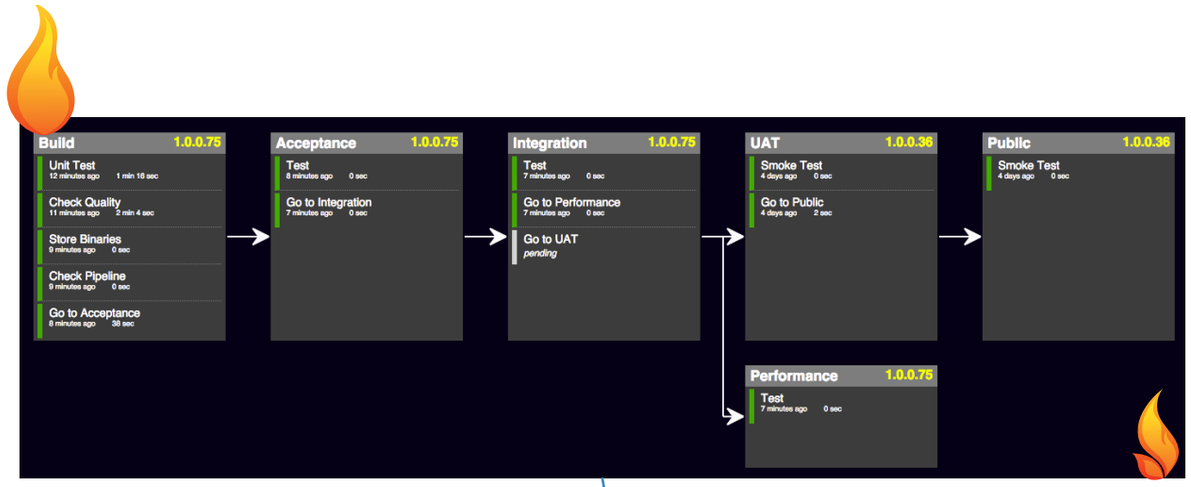
Chef Metal **cooks**

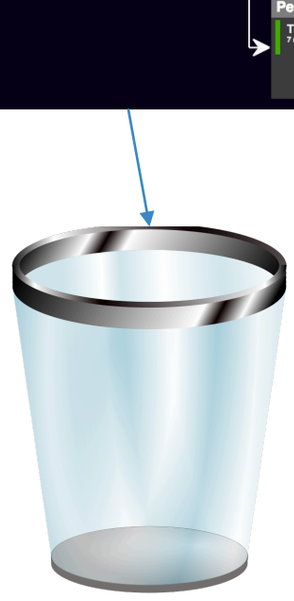
You **see results and TODOs** when it's done

< 10 min

Phoenix infrastructure is <u>real</u>

Martin Fowler coined the term here: http://martinfowler.com/bliki/PhoenixServer.html

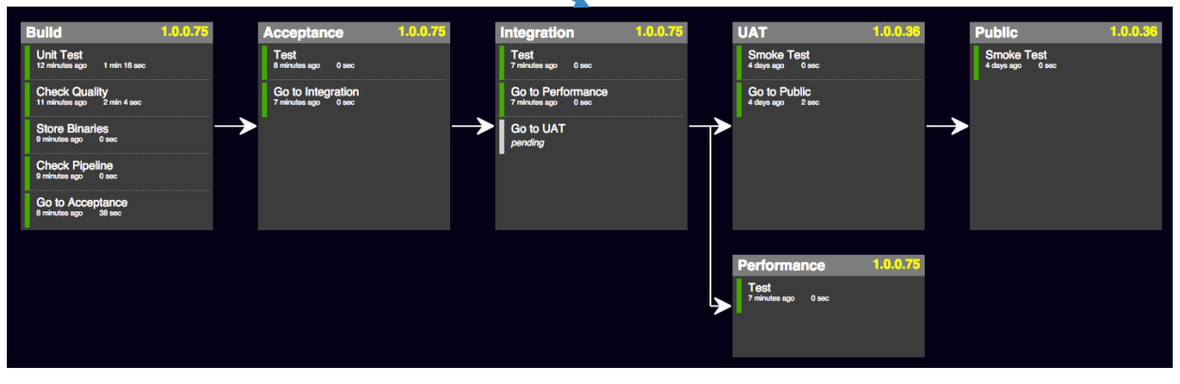




All **automated**

All **“in code”**

Takes only **minutes**



An example ...