




Packaging ~~Problems~~ Opportunities

Florent Mara
<florent.mara@gmail.com>



Introduction

- Company, software and processes maturing
- A lot of stable code, a lot of new code
- Increasingly complex and fluid environment

Actively promoting easier, faster, safer releases. Aiming for continuous deployment ... for now.

Now is time for our next iteration at improving our development tools. How?

2008 ... 2012 - Prehistory

- 1 Perl project, 2 Java front-ends
 - Perl project tied to Talend IDE
 - 2 svn repositories / silos
 - Deploying to Microsoft servers
-
- 11 releases
 - Slow, tedious, frustrating, documentation based click & point releases

2012 ... 2014 - Step 1

- Second team member to work on Perl project
- Necessity to migrate from Talend IDE

- 18 months migration with 'manual merging'
- Still 2 svn repositories but now both doing version control
- 3rd Java front-end
- Release still involve documentation + click + point + sweat

May & June 2014 - Step 2

- Perl project migration completed
 - 2 repositories moved to git and merged
 - Jenkins integration server setup
 - Perl project developed and released on Linux Debian
-
- Followed by 5 weeks holiday overseas
 - Same release process as before

Since June 2014 - Step 3

- Started tooling: devtools
 - Builds 3 Java projects + Perl backend into a tar file
 - Deploys the Perl backend to staging
- More & less complexity
 - (more) Fluid environment
 - (more) 1 new microservice: Perl backend and Node.JS frontend
 - (less) Phasing out Microsoft Windows
 - (less) One Git repository
- Configuration management with Ansible
 - Instance base setup and configuration.
 - A lot more untapped scope
- 51 releases

Some Perl code

```
my $user = 'worker';
if ( system("scp $file $user@$server:$file") != 0 ) {
    say "Could not SCP $file to $server";
    Return;
} else { say "Success: SCP $file to $server"; }
if ( ssh_cmd( { user => $user, host => $server, command => 'rm -rf backend'} ) ) {
    say "Could not rm backend folder from $server";
    Return;
} else { say "Success: rm backend folder from $server"; }
if ( ssh_cmd( { user => $user, host => $server, command => 'rm -rf workspace/release'} ) ) {
    say "Could not rm release folder from $server";
    Return;
} else { say "Success: rm release folder from $server"; }
if ( ssh_cmd( { user => $user, host => $server, command => "tar -xf $file"} ) ) {
    say "Could not untar $file on $server";
    ....
}
```

Packaging Problems ~~Opportunities~~

- Setup gaps. Eg : Ansible does some things, devtools does other, neither does crontab configuration, secrets, env. configuration.
- Devtools build is bulk
- Devtools is not trusted to be applied in production
- No clean un-install.
- Release documentation still has ~20 steps.

- Too much knowledge required from operator.

Packaging ~~Problems~~ Opportunities

- Started work on emmet (Master Builder)
 - Should accept command line parameters for target git tag and project
 - Should include all current projects
 - Should run some project tests
 - Should be fully tested and trusted to package for production
- Started work on rapt (Regen apt)
 - Should accept command line parameter for target environment
 - Should enable uninstalling in all environment
 - Should be fully tested and trusted to safely deploy to production

Both should behave as normal command line tools (help, man, verbose ...) and be easily maintained, expanded, improved.

Packaging ~~Problems~~ ~~Opportunities~~ Questions

- Ansible playbook wrapping Perl command line(s)
- Or Perl command line wrapping ansible playbook(s)
- Either way, what should each do? Does it matter?
- What about building .deb packages?
- Where to: setup / unset crontab? Add secret / env. specific configuration? User and folders setup? Should dependencies be installed with app deploy or beforehand using a different process?
- Why not containers? (but then how to prepare, deploy, manage, monitor ...)
- Get an intern to do all that boring stuff by hand? (and coffee!)
- None of the above?
- What do people do?

Questions

Florent Mara
<florent.mara@gmail.com>

Bonus slide

Find the error in this list and why.

Node.JS, PHP, Python, Ruby, Tomcat, IIS, Java, Go, Docker, Glassfish