puppet

GUERRILLA DEVOPS @ Braintree™

TODAY

TODAY

WHO'S BRAINTREE?

WHY PUPPET?

SUPPLY_DROP

LESSONS LEARNED

Braintree™

MISSION OBJECTIVE

1. ♥ 2. $

DEAD SIMPLE

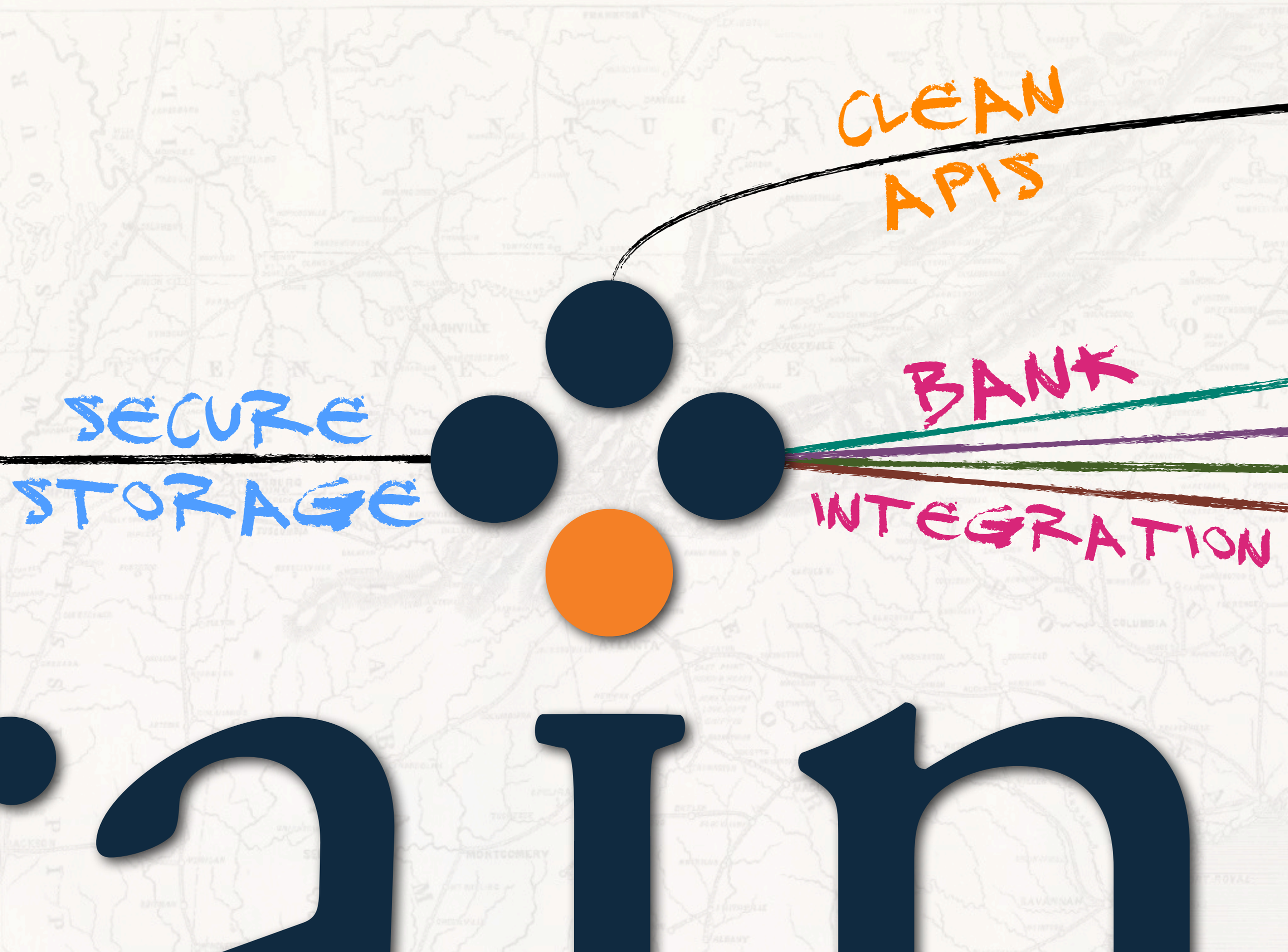
Braintree™

raintree

CLEAN APIS

BANK INTEGRATION

SECURE STORAGE



Braintree™

- SEXY TECH
- KILLER SUPPORT
- ROCK-SOLID PLATFORM

~~WHO'S BRAINTREE?~~

WHY PUPPET?

TODAY

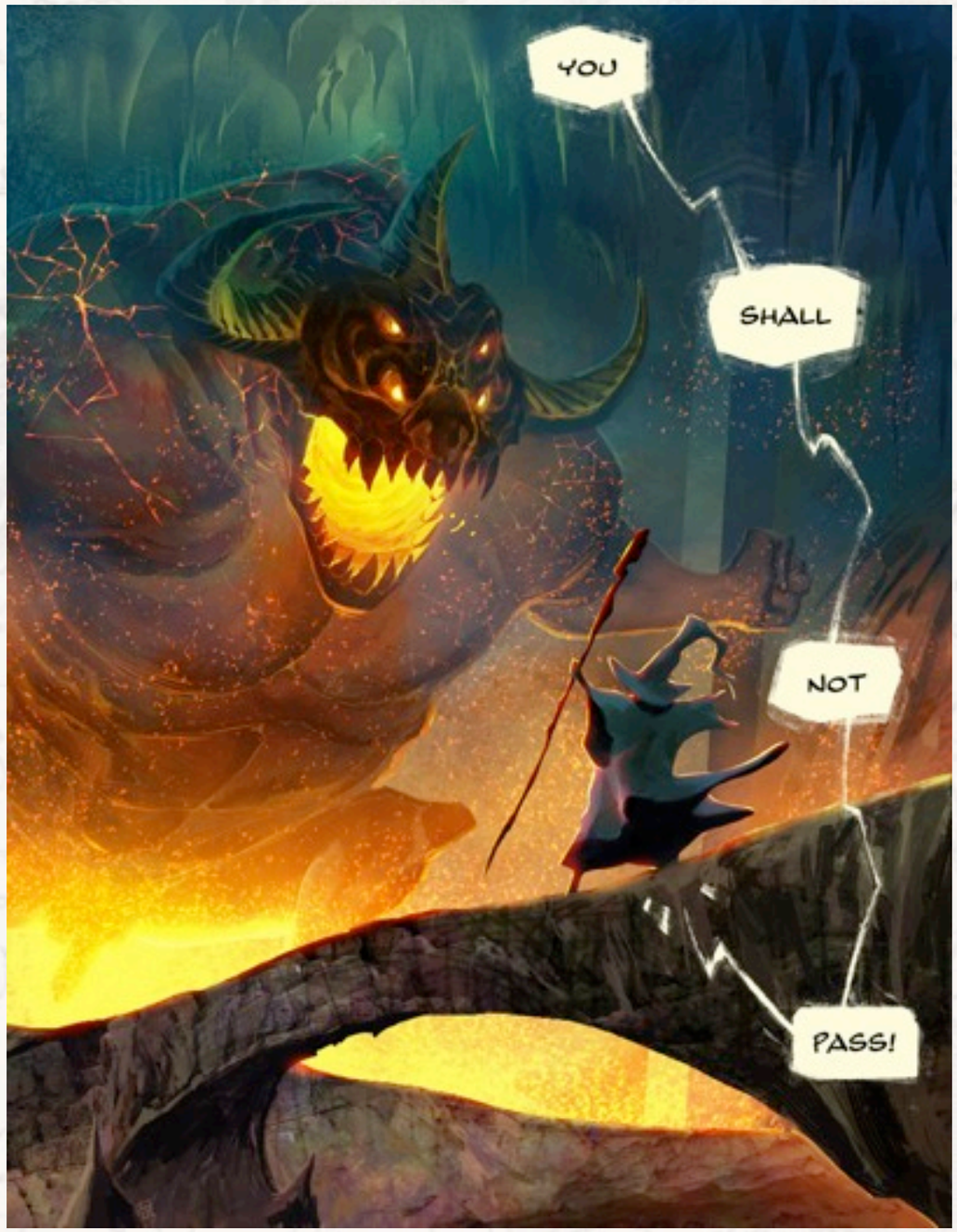
SUPPLY_DROP

LESSONS LEARNED

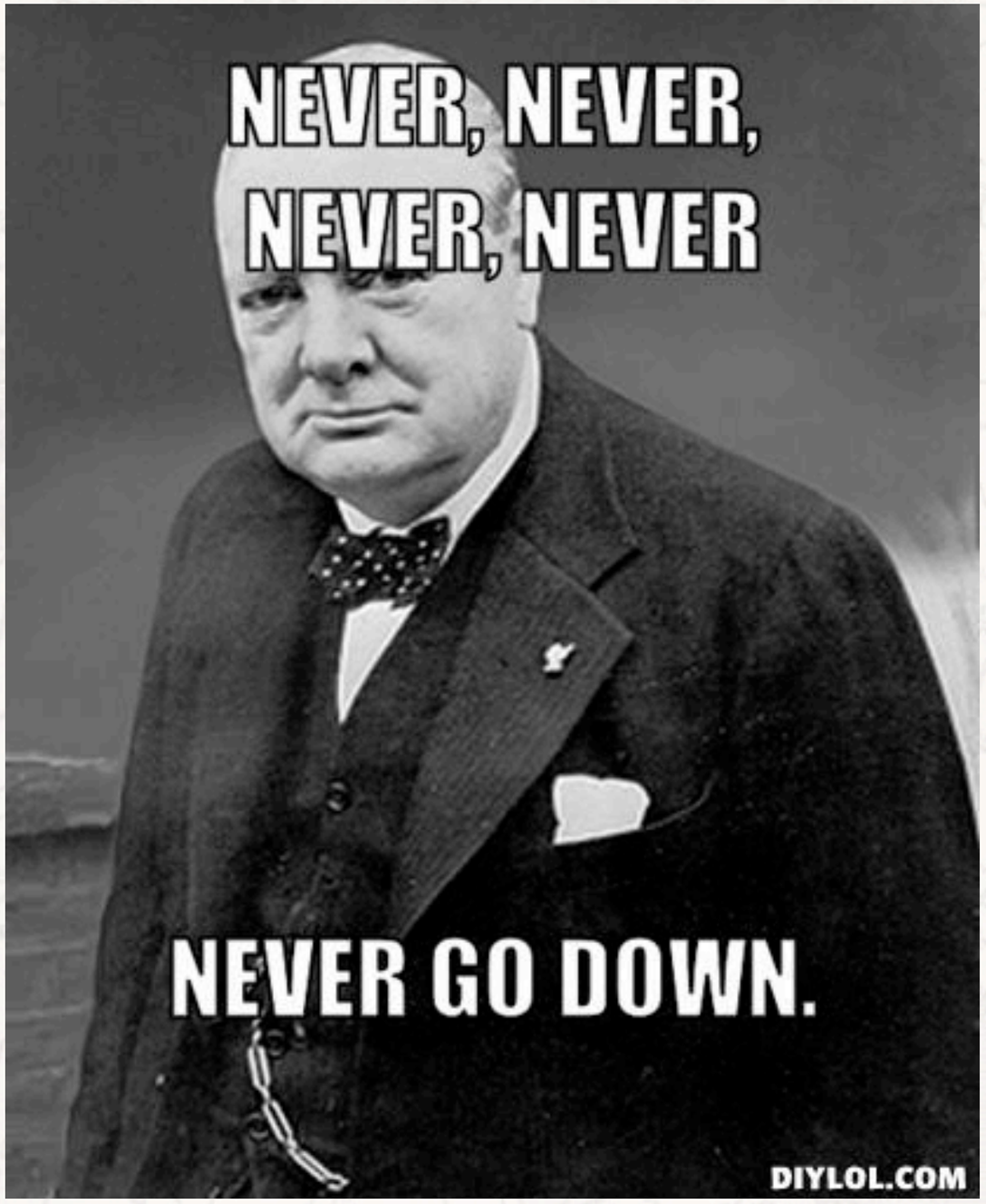
WHY PUPPET?

A SMALL STARTUP...

...WITH A BIG JOB

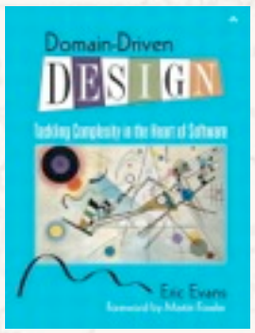


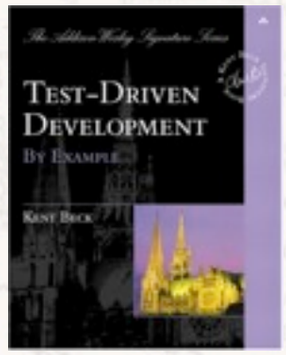
...WITH A BIG JOB

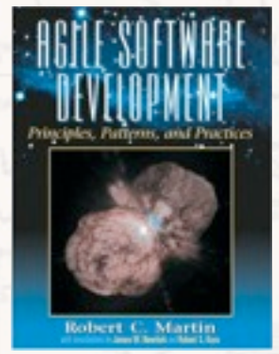


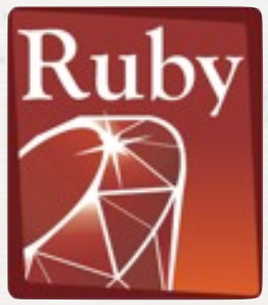
USE WHAT YOU'VE GOT

<3









USE WHAT YOU'VE GOT

puppet

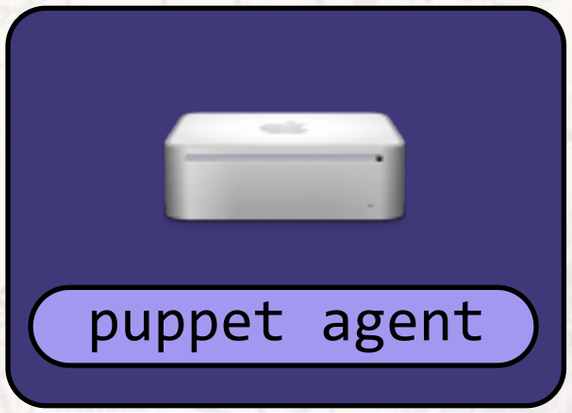
- AUTOMATED CONSISTENCY
- VERSION CONTROLLED
- LAYER(S) OF ABSTRACTION

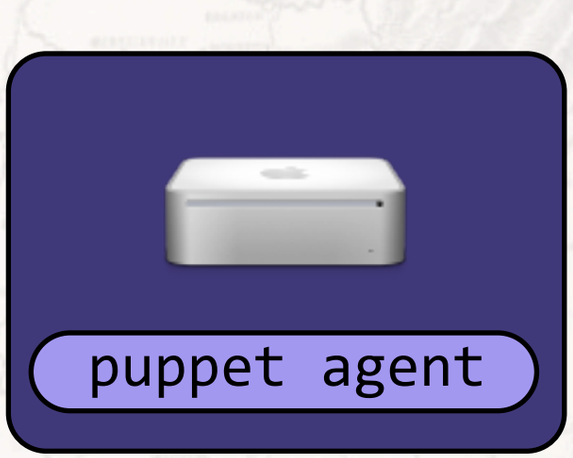
puppet !!

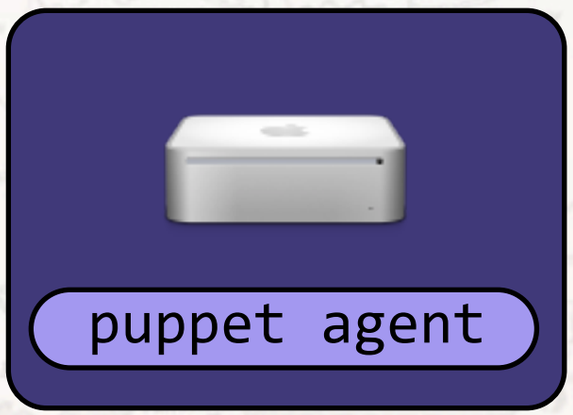
MASTERLESS

WHY PUPPET?

puppet MASTER





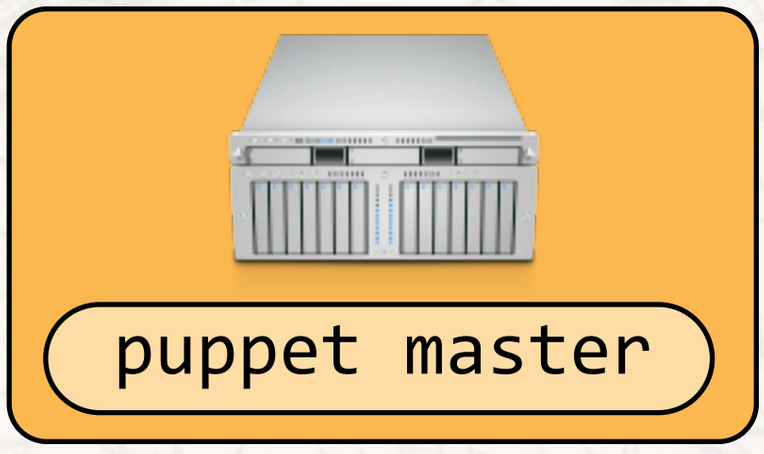


WHO AM I?

a proxy server!

a db box!

an app node!



MASTERLESS puppet

YOU TWO:

MONGODB CLUSTER!

APP NODE ASAP!

MASTERLESS puppet

FINE-GRAINED CONTROL

PARALLELIZATION

DISTRIBUTED (NO S.P.O.F.)

~~WHO'S BRAINTREE?~~

~~MASTERLESS~~

~~WHY, PUPPET?~~

TODAY

SUPPLY_DROP

LESSONS LEARNED

SUPPLY_DROP

SUPPLY_DROP

- CAP TASKS
- PARALLEL RSYNC
- COLLISION DETECTION

LET'S GET SUPPLY_DROP

```
# console
$ mkdir infrastructure
$ cd infrastructure
$ cat <<GEMFILE > Gemfile
source :rubygems

gem 'capistrano'
gem 'supply_drop'
GEMFILE
$ bundle install
...
Your bundle is complete!
$ vim config/deploy.rb
```

LET'S USE SUPPLY_DROP

```
# config/deploy.rb...
def tasks_for_datacenter(datacenter, servers)
  task datacenter do
    role :server, *servers
  end

  servers.each do |server|
    task server do
      role :server, server
    end
  end
end

tasks_for_datacenter :qa, %w(app1.qa db.qa)
tasks_for_datacenter :production, %w(app1.prod db.prod)
```

LET'S USE SUPPLY_DROP

```
# ...config/deploy.rb
require 'rubygems'
require 'supply_drop'
```

```
# puppet.pp
node 'app01.qa' {
  package { 'python': ensure => installed }
}
```

```
# console
$ cap app01.qa puppet:noop
notice: /Stage[main]/Package[python]/ensure:
current_value absent, should be present (noop)
$ cap app01.qa puppet:apply
```

FINE GRAINED CONTROL

```
$ cap db01.qa puppet:noop          # single boxes
$ cap app{01..05}.qa puppet:noop # classes of boxes
$ cap qa puppet:noop               # entire environment
```

WE ALWAYS READ THE DIFF

```
$ git branch --list
* master       # maps to QA environments
* staging      # maps to pre-prod environments
* production   # maps to production environments
```

PARALLELIZATION

ANYBODY USING QA? WORKING ON A NAGIOS CHECK

I'M TWEAKING THE POSTGRES CONFIGS, BUT WE SHOULD BE FINE

```
$ cap monitor01.qa puppet:noop
$ cap monitor01.qa puppet:apply
$ git pull --rebase
$ git commit
```

```
$ cap db01.qa puppet:noop
$ cap db01.qa puppet:apply
$ git pull --rebase
$ git commit
```

PARALLELIZATION

```
$ git co master && git pull
$ git co staging && git pull
$ git merge master
$ git log origin/staging..HEAD
$ git diff origin/staging..HEAD
# review git log and diff
$ cap staging puppet:noop
# review puppet noop output
$ cap db01.stg puppet:apply
# selectively apply high-risk changes
$ cap staging puppet:apply
# flush out environment
$ cap staging puppet:noop
# double check "clean noop"
$ git push
```

DISTRIBUTED

```
.
├── config                  # capistrano tasks & supply_drop config
├── datacenters             # variable farms inherited by all nodes
|   ├── qa.pp
|   ├── staging.pp
|   └── production.pp
├── manifests               # legacy manifests; here be dragons
├── modules                 # new-style modules
├── nodes                   # node definitions
|   ├── qa
|   |   ├── app.pp          # divided into per-class files
|   |   ├── db.pp
|   |   └── proxy.pp
|   ├── staging
|   └── production
├── puppet.pp               # top-level puppet file
├── templates               # legacy templates; here too be dragons
└── vendor                  # vendored puppet pushed to machines
    ├── facter-1.6.4
    ├── puppet-2.6.13
    └── supply_drop-0.10.2
```

~~WHO'S BRAINTREE?~~

~~MASTERLESS~~

~~WHY, PUPPET?~~

TODAY

~~SUPPLY DROP~~

LESSONS LEARNED

LESSONS LEARNED

AND UNANSWERED QUESTIONS

puppet

IS A

DANGEROUS

WEAPON

DECEPTIVELY SIMPLE

```
node "app1.qa" {
  $rails_env = "qa"
  include users
  include apache
  include rails
}
```

```
node "app1.prod" {
  $rails_env = "prod"
  include users
  include apache
  include rails
}
```

```
<VirtualHost *:443>                          # apache2.conf.erb
<% if rails_env == 'prod' -%>
  HostName www.example.com
<% else -%>
  Hostname qa.example.com
<% end -%>
</VirtualHost>
```

MIND YOUR ABSTRACTIONS

```
node "app1.qa" {
  include apache
  apache::site { 'example':
    hostname => 'qa.example.com'
  }
}
```

```
node "app1.prod" {
  include apache
  apache::site { 'example':
    hostname => 'www.example.com'
  }
}
```

MIND YOUR ABSTRACTIONS

- ☑ 0..1 OR 0..N?
- ☑ GENERIC OR UNIQUE?
- ☑ WHAT CHANGES?
  - ☑ REQUIRED VARIABLES
  - ☑ OPTIONAL VARIABLES
  - ☑ DEFAULTS

MIND YOUR ABSTRACTIONS

MODULES. USE THEM.

MIND YOUR SCOPE

location

node

class/type

template/file

WATCH YOUR SIX!

WORKING THESIS

puppet DOES NOT MANAGE TRUTH

puppet MAPS TRUTH TO COMPLEXITY

HOW SHOULD WE MANAGE TRUTH?

puppet

GUERRILLA DEVOPS @ Braintree™