Puppet + Hiera

by

Olivier Schad


AUTOMATIC
SERVER AG

This is a talk about managing data, especially with Hiera

This is a talk about managing data, especially with Hiera

Context: Web companies

Why?

Don't start with questions, it's friday

Web companies have a very specific requirement set. More heterogeneous environement should use a ENC like foreman or a mix of Hiera and ENC

Why isn't Foreman good enough for everything?

Did I mention, it's friday?

2 things to distinct

- Code/Manifests
- Data
- Welcome to software development topics

What is wrong with old solutions?

- Put your data in some manifests
- in the first place nothing wrong, it works
- But ask your developers what's wrong with putting your customer data into the code
- syntax of your decisions is noisy, you have bracktets, big decision trees, inheritance trees, variable renaming
- potential to do very bad things every day, cause puppet gives you full control and every change means potential full fuckup
- split manifest release cycles and data changes

Split the things

- daily business is managing data
- sometimes manifest development

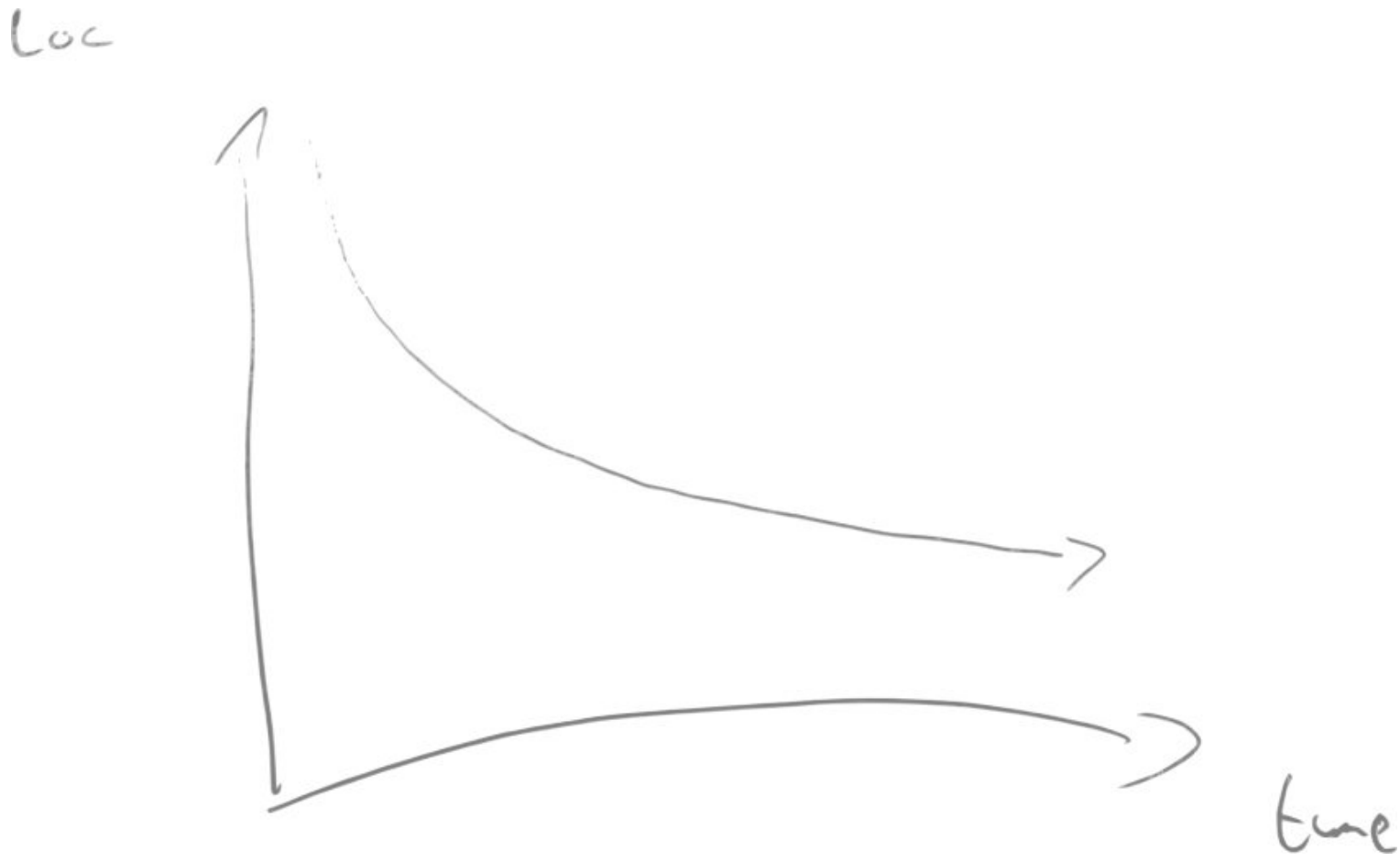


Abbildung: Efforts for puppet manifests over time

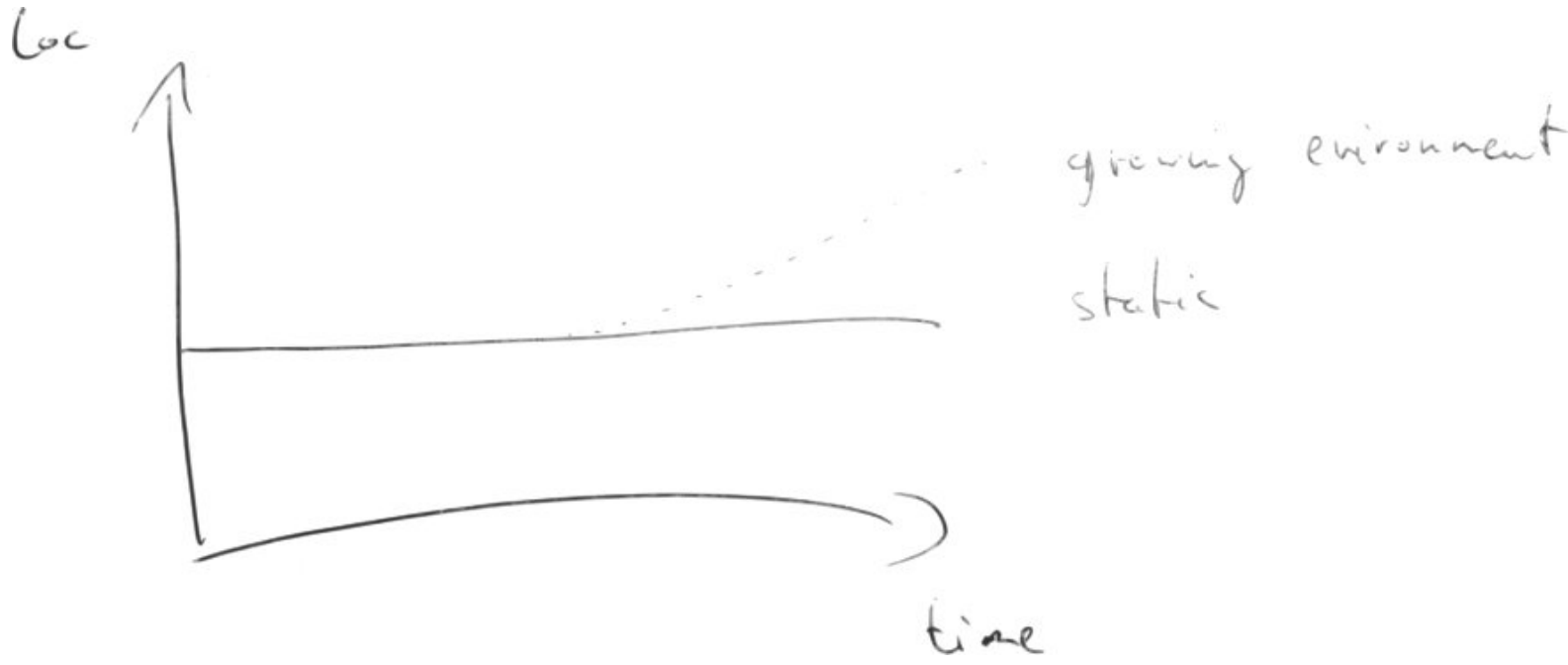


Abbildung: Efforts for managing data

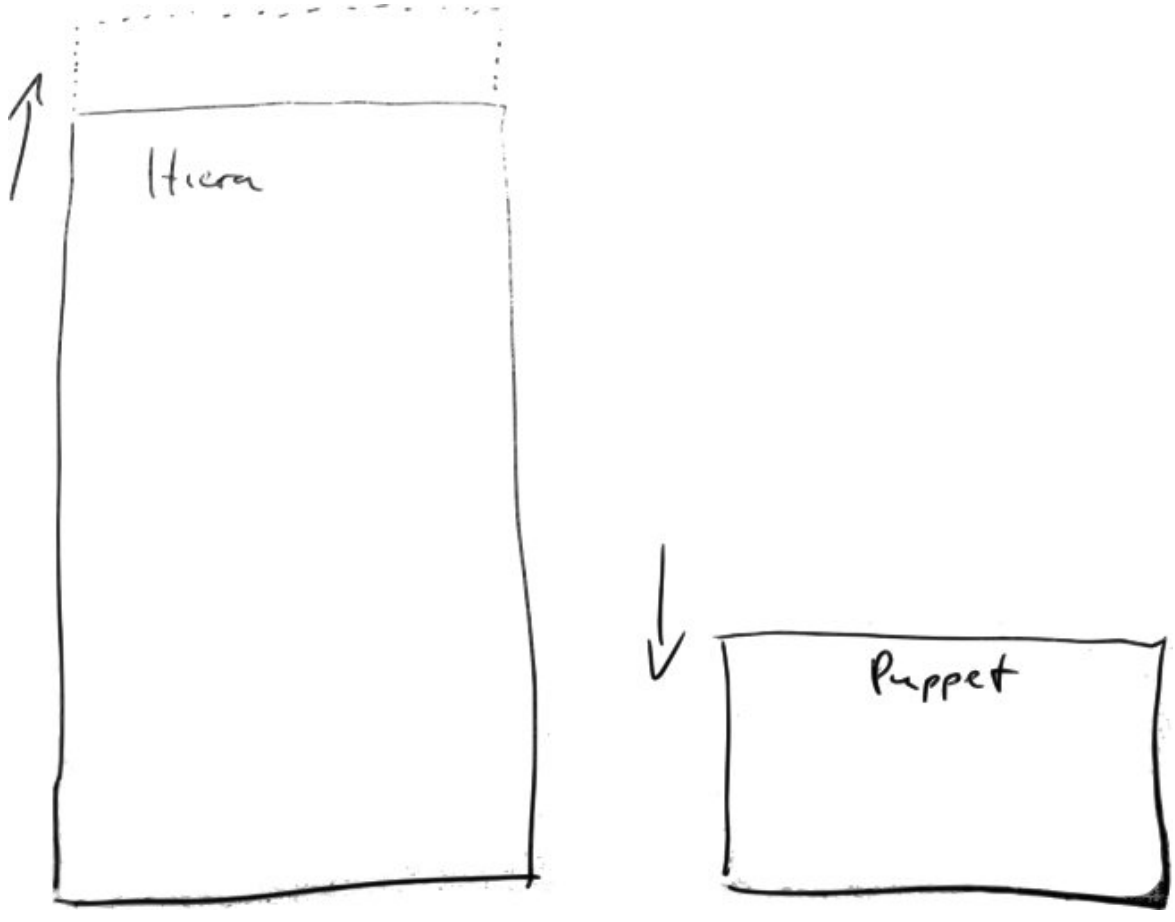


Abbildung: Relative Efforts

So how to make puppet with Hiera working?

```
---
network::nameservers:
  - 10.0.0.10
  - 10.0.0.11

network::search_domains:
  - production.example.com
  - example.com

proxy::http: http://proxy:3128
```

```
class dns (
  $nameservers = hiera('dns::nameservers')
) {

  file { '/etc/resolv.conf':
    owner   => 'root',
    group   => 'root',
    mode    => '0444',
    content => template('dns/resolv.conf.erb'),
  }

}
```

```
class nginx::defines () {
  $nginx_vhost_data = hiera("nginx::vhosts")
  if $nginx_vhost_data {
    create_resources('nginx::vhost', $nginx_vhost_data)
  }
}
```

Puppet 3 did handle Hiera and default parameters wrong

```
data_binding_terminus = none
```

Puppet 3.3.2

Be happy, works again - I think

Clean split

hiera is about managing parameters/data so you have a clean split, so you have a less powerful tool for doing daily business with a clean syntax

What is Hiera?

- Hiera is a simple key value store which uses itselfs several native stores as backends (YAML, JSON, Redis, MySQL, ...)
- Hiera is not that simple, it has a lookup for a key at several places (in the same backend)
- the places where to look up are fact dependent, so it's at the end machine dependent
- first match wins (but we can do more)
- so this is hierarchical lookup
- it is done for every key from scratch, no memoryzation of old lookups and where a key was found before

How is it useful?

- machinename
- environment
- default

Lookup for nameserver data

- mymachine.example.com.yaml - do I find this key? - no
- staging.yaml - do I find this key? - yes, install those nameservers in /etc/resolv.conf
- default.yaml - don't look at it

Which layers could we add?

- data center
- hostgroup
- application name

How?

- add custom facts to your system

So what data can we put in Hiera?

- which classes should be included?
- which parameteres should they have?
- which defines should be included?
- which parameters should be included?
- everything else

So how could your hiera.conf look like?

```
- env/%{aa_env}/app/%{aa_app}/hosts/%{fqdn}
- env/%{aa_env}/app/%{aa_app}/hostgroups/%{aa_hostgroup}
- env/%{aa_env}/app/%{aa_app}/default
- env/%{aa_env}/app/default
- env/%{aa_env}/default
- loc/%{aa_loc}/app/%{aa_app}/hosts/%{fqdn}
- loc/%{aa_loc}/app/%{aa_app}/hostgroups/%{aa_hostgroup}
- loc/%{aa_loc}/app/%{aa_app}/default
- loc/%{aa_loc}/app/default
- loc/%{aa_loc}/default
- default
```

Anti-Pattern

- Put everything in one file - use a few pseudo layers instead in the worst case or hack your include (suggestion: something like conf.d pattern)
- Make your environments totally unique - you have to merge changes through your stages and that is brain driven

Workflows and Tools

Use something like Git

- Can I use subversion?

NO

- But ...

NO!!!!!!!11111

But

Shut up beavis

Every environment should be one branch

Which environments should I have?

- Your personal development environment, play and test there
- A puppet integration environment, a small picture of your production
- Your development pipeline (testing/staging/production) of the devs

So what I use to handle that?

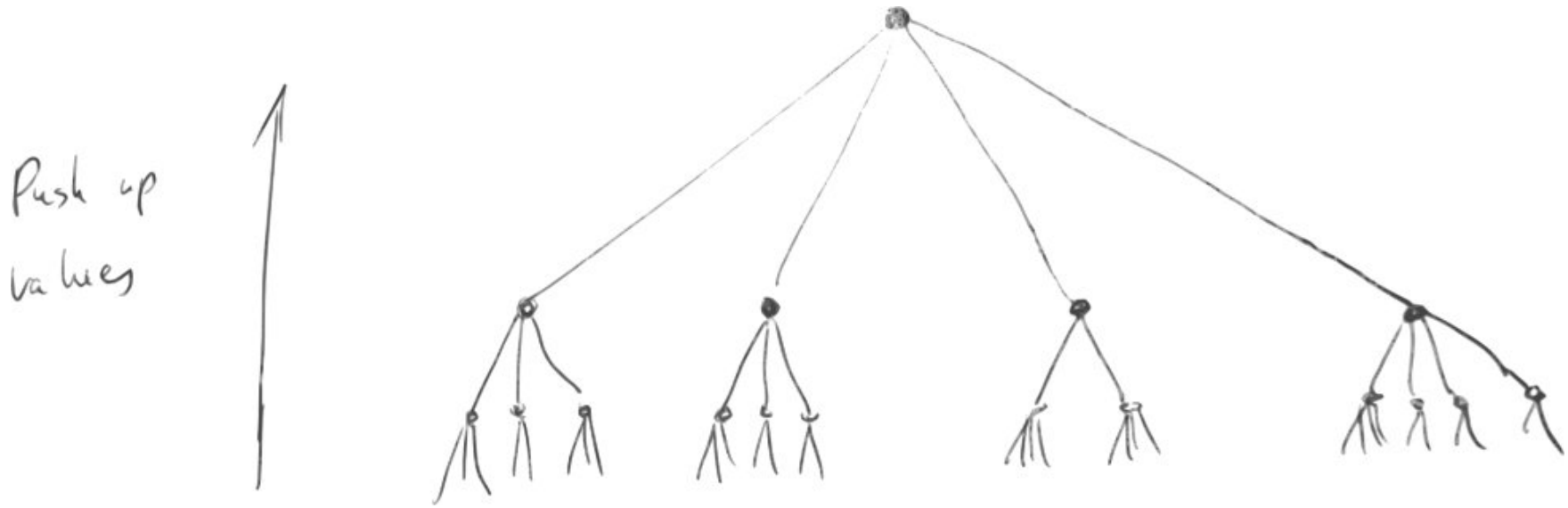
- Git
- CI-Server of your choice
- Deploy-Tool like Capistrano, triggered by CI-Server
- Ask your developers for integration tests, run it
- Ask your developers how to build up a software developing chain

Or better not

New development cycle

- Data
- Manifests
- Your data has environment specific parts so you have to handle it in parts manually - brain driven

review cycle - what's going on in your data?



Questions?