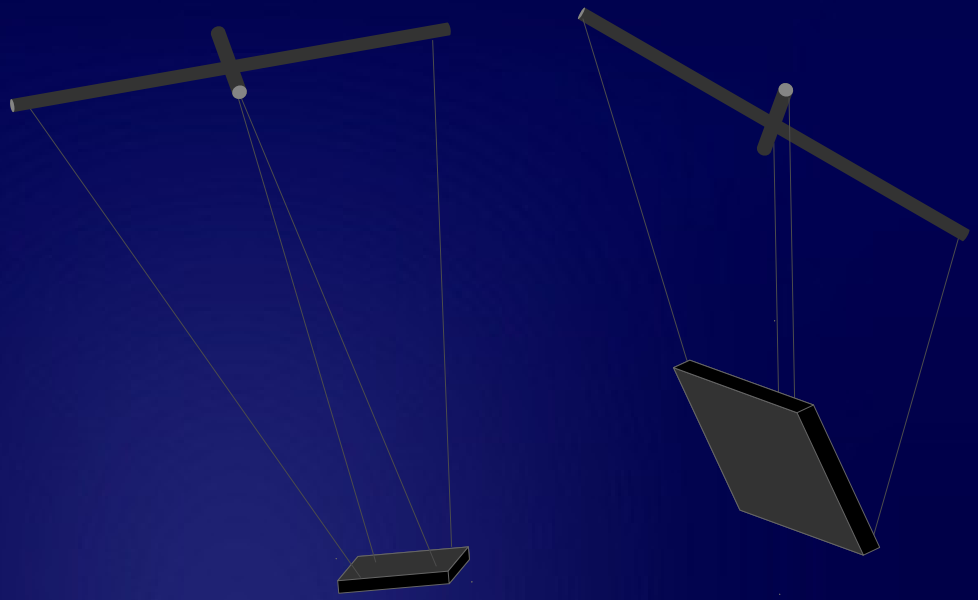
Doodle

Managing your servers with

Puppet



David Gubler

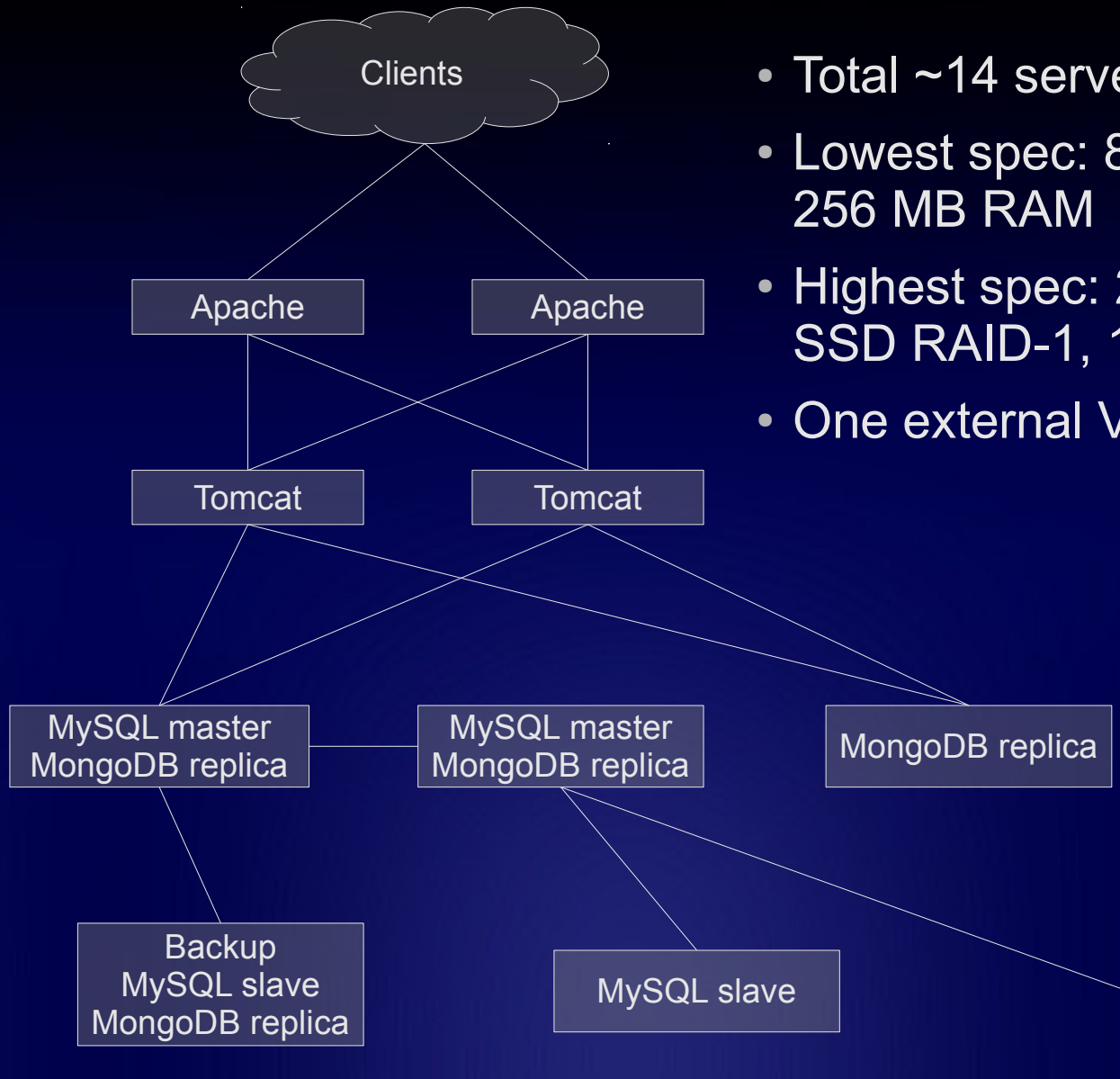
Software & Operations Engineer @ Doodle AG

Doodle

ToC

- Motivation and Introduction
- Authentification
- Examples
- Classes and Modules
- Nodes and Templates
- Facts
- Puppet @ Doodle
- Tips and Tricks
- Pointers

Motivation: Doodle Infrastructure



- Total ~14 servers
- Lowest spec: 800 Mhz Geode, 1 GB CF, 256 MB RAM
- Highest spec: 2x4 Core Xeon, 2x 300 GB SSD RAID-1, 128 GB RAM
- One external VM (not hosted by us)

Motivation: The challenges

- Manage configuration...
  - Keeping all servers in sync?
  - What the heck did I do 1.5 years ago to make X work?
  - Y stopped working. Last changes? Uhm...
  - Backup /etc/*? Not really.
- Manage admin users...
  - LDAP is overkill
- Distribute custom components...
  - Backup scripts, GeoIP database, Munin plugins, whatever

Introducing Puppet

- Declarative language
- Clients connect to server („puppetmaster“) to fetch their configuration
- Re-use configuration
  - Even if servers have subtle (or not-so-subtle) configuration differences
- Ready-to-use constructs for common tasks and software
- Supports many configuration file syntaxes (!)

You get...

- Always up-to-date documentation
- History
  - Keep configuration in version control system
- Centralized backup
  - No need to backup or restore individual files in /etc
  - Implement backup strategies only for your data (DB, user's files), not for your server setup
  - No „precious“ servers with non-reproducible configuration that has grown over years
- Flexibility
  - Easily move services from one server to another
  - Clone services without hassle
- „DevOps“
  - Apply principles of software engineering to operations

Doodle

Authentication

- Configuration may contain secrets
- Fake puppetmaster could „own“ your servers

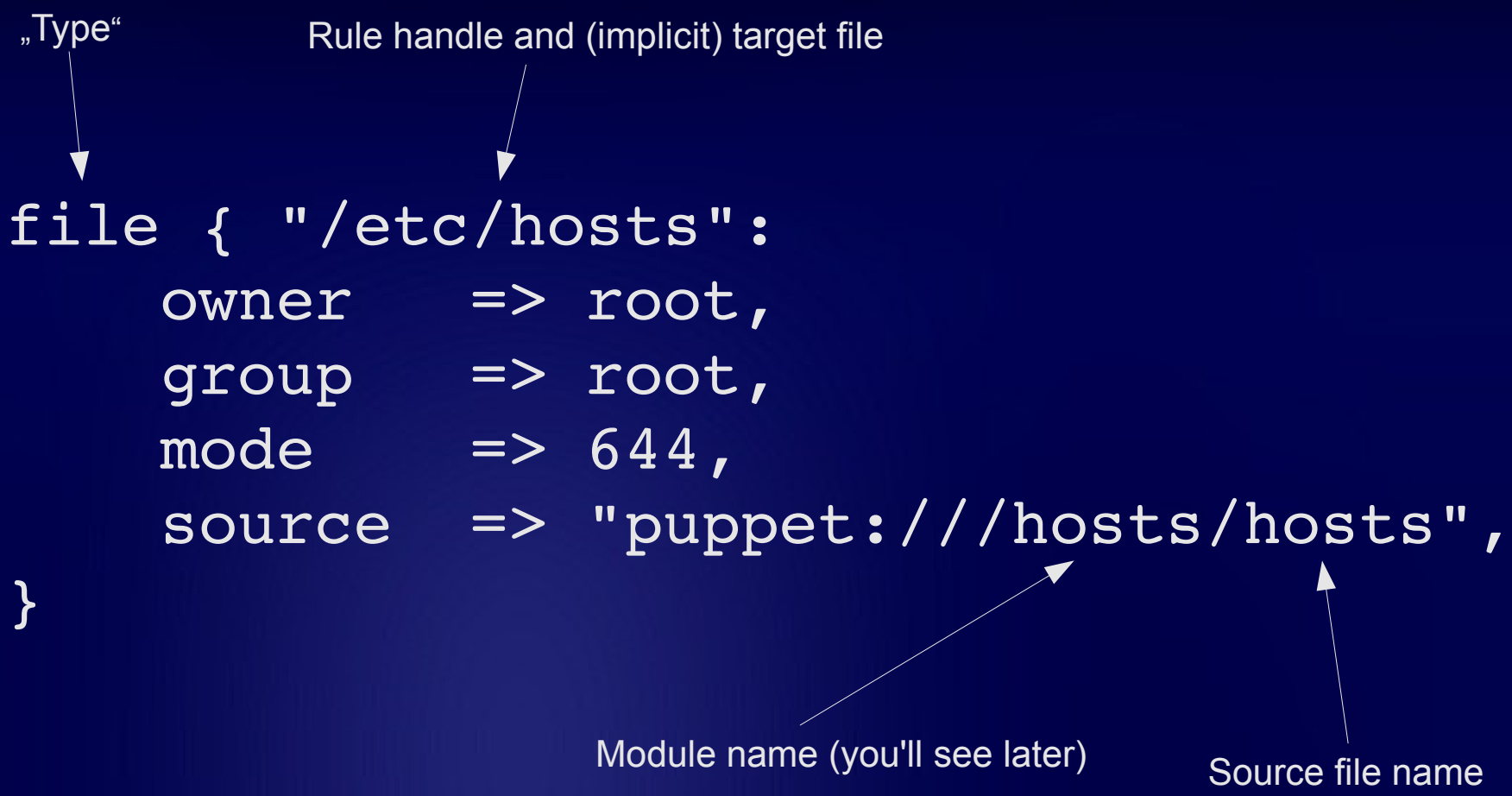
Puppet uses it's own PKI:

- Easy to use (no cryptic openssl commands)

```
puppetmaster:~# puppetca --list
  „worker.doodle.com“ (AB:03:C0:FF:EE:BA:BE:D1:38:97:34:C5:43:E0:57:5D)
puppetmaster:~# puppetca --sign worker.doodle.com
puppetmaster:~#
```

Example 1: Distribute a file

- Problem: Deploy /etc/hosts on all servers
- Solution: Puppet!



Doodle

Example 2a: Create a user

- Problem: Create user accounts on all servers
- Solution: Puppet!

```
group { "dg":
    ensure => 'present',
    gid => 1009,
}
user { "dg":
    ensure => 'present',
    managehome => 'true',
    shell => '/bin/bash',
    uid => 1009,
    comment => 'David Gubler',
    gid => 'dg',
    password => '$1$eZsBI1E2$NR8bHTmRSnXaMigwQO2y48'; *
}
```

* Don't bother trying to crack the hash, it's completely fake

Example 2b: Remove a user

- Puppet makes sure that the server conforms to whatever you specify
- It does not care about things you don't specify! Thus, removing group{} and user{} does not help: Puppet will just stop caring!
- Solution: ensure => 'absent'

```
group { "dg":
    ensure => 'absent',
}
user { "dg":
    ensure => 'absent',
}
```

Example 3: Enable syncookies

- Problem: Enable syncookies in /etc/sysctl.conf, do not modify other settings, apply without reboot
- Solution: Puppet! (Duh)

```
augeas { "syncookies":
    context => "/files/etc/sysctl.conf",
    changes => [
        "set net.ipv4.tcp_syncookies 1",
    ],
    notify => Exec[ "setsyncookies" ],
}
exec { "setsyncookies":
    command => "/sbin/sysctl net.ipv4.tcp_syncookies=1",
    refreshonly => true,
}
```

Doodle

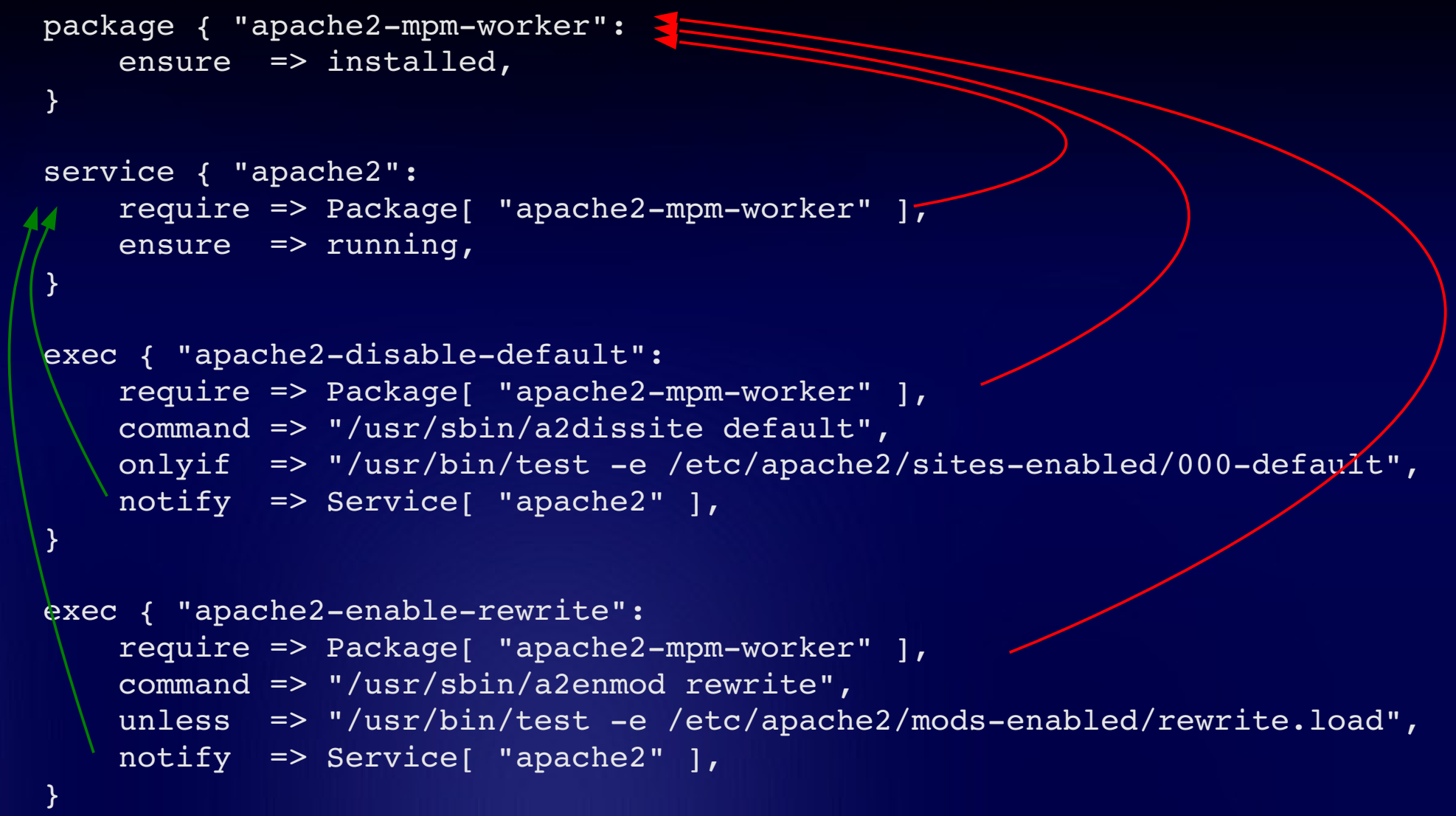
Example 4: Dependencies

```
package { "apache2-mpm-worker":
    ensure  => installed,
}

service { "apache2":
    require => Package[ "apache2-mpm-worker" ],
    ensure  => running,
}

exec { "apache2-disable-default":
    require => Package[ "apache2-mpm-worker" ],
    command => "/usr/sbin/a2dissite default",
    onlyif  => "/usr/bin/test -e /etc/apache2/sites-enabled/000-default",
    notify  => Service[ "apache2" ],
}

exec { "apache2-enable-rewrite":
    require => Package[ "apache2-mpm-worker" ],
    command => "/usr/sbin/a2enmod rewrite",
    unless  => "/usr/bin/test -e /etc/apache2/mods-enabled/rewrite.load",
    notify  => Service[ "apache2" ],
}
```



Example 5: XEN VM

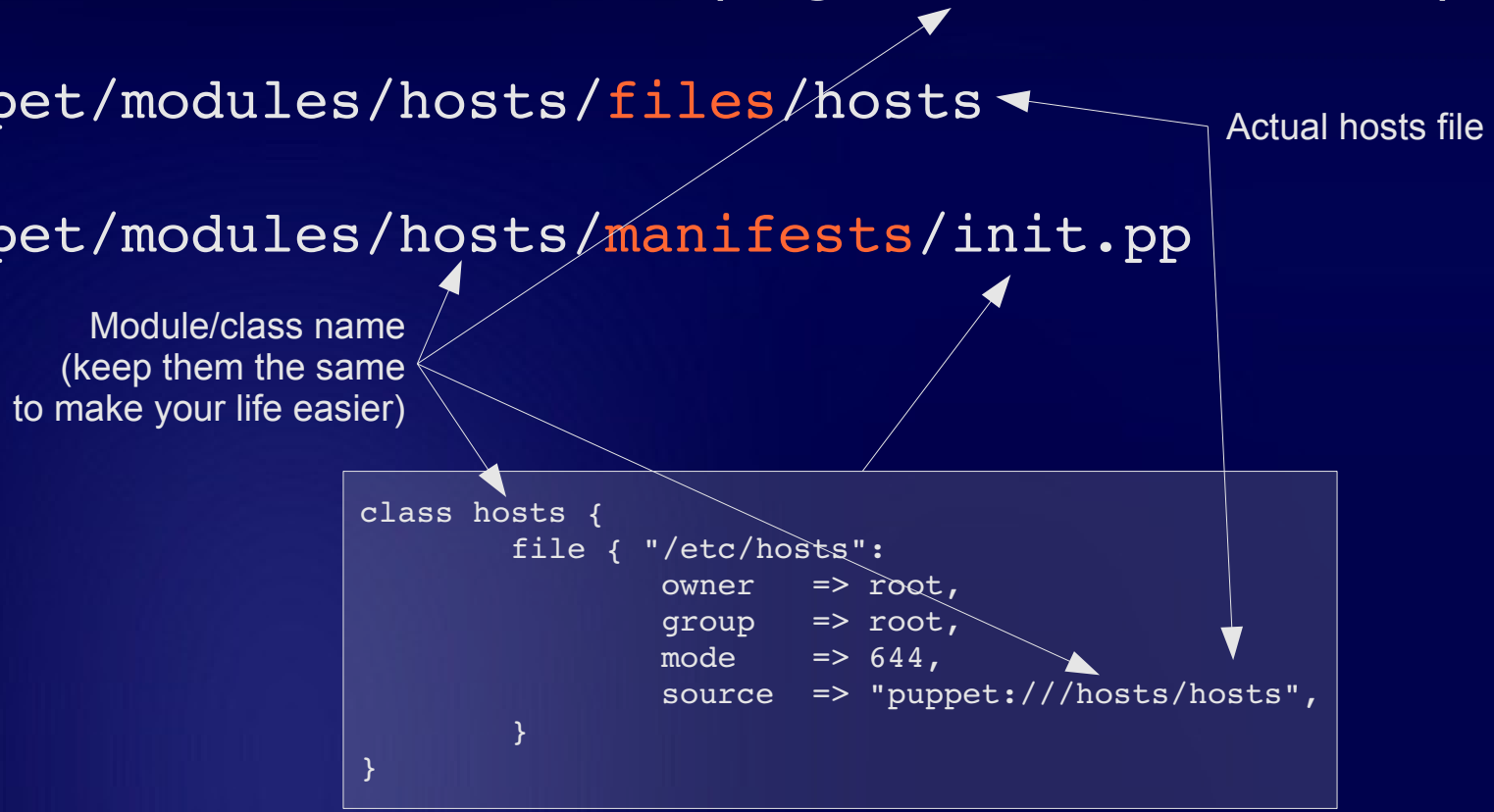
Client certificate and private key for Puppet, hook copies them into new VM

Hook to install Puppet on the new VM before booting it for the first time

```
class xen-dom0::build {
        file { "/etc/xen-puppet/certs/build.doodle.com.pem":
                require => File[ "/etc/xen-puppet/certs" ],
                source  => "puppet:///xen-dom0/certs/build.doodle.com.pem",
        }
        file { "/etc/xen-puppet/private_keys/build.doodle.com.pem":
                require => File[ "/etc/xen-puppet/private_keys" ],
                source  => "puppet:///xen-dom0/private_keys/build.doodle.com.pem",
                mode    => 600,
        }
        exec { "xen-create-build":
                command => "/usr/bin/xen-create-image --hostname=build.doodle.com
                                --ip=188.92.145.101 --gateway=188.92.145.65
                                --netmask=255.255.255.192 --vcpus=6 --pygrub
                                --dist=wheezy --lvm=vg0 --memory=8Gb --size=100Gb
                                --noswap --fs=ext4 --nohosts --role=doodlePuppet --boot",
                unless => "/bin/uname -a | /bin/grep -vq xen || ! /usr/sbin/brctl show
                        | /bin/grep -q xenbr || /usr/bin/test -e
                        /dev/vg0/build.doodle.com-disk || /usr/bin/test -e
                        /etc/xen/build.doodle.com.cfg",
                require => [ Package[ "xen-tools" ], Package[ "bridge-utils" ],
                        File[ "/etc/xen-tools/role.d/doodlePuppet" ],
                        File[ "/etc/xen-puppet/certs/build.doodle.com.pem" ],
                        File[ "/etc/xen-puppet/private_keys/build.doodle.com.pem" ] ],
        }
}
```

Classes and Modules

- Puppet rules are combined to a „class“
- Classes and some files make up a „module“
- Subclasses are supported (e.g. hosts::external)



Subclasses

- Use them if a module is shared between multiple servers, but rules differ

modules/apache2/manifests/install.pp
modules/apache2/manifests/php.pp
modules/apache2/manifests/dev.pp

```
class apache2::install {
        package { "apache2-mpm-worker":
        [...]
}
```

```
class apache2::php {
    include apache2::install
    package { "libapache2-mod-fcgid":
        require => Package[ "apache2-mpm-worker" ],
        ensure  => installed,
        notify  => Service[ "apache2" ],
    }
    package { "php5-cgi":
        require => Package[ "apache2-mpm-worker" ],
        ensure  => installed,
        notify  => Service[ "apache2" ],
    }
}
```

```
class apache2::dev {
    include apache2::install
    include apache2::php
    include ssl::production
    file { "/etc/apache2/sites-available/dev.doodle.com":
        owner   => root,
        group   => root,
        mode    => 644,
        source  => "puppet:///apache2/site.dev",
        notify => Service[ "apache2" ],
    }
    exec { "apache2-enable-dev.doodle.com":
        require => File[ "/etc/apache2/sites-available/dev.doodle.com" ],
        command => "/usr/sbin/a2ensite dev.doodle.com",
        unless  => "/usr/bin/test -e /etc/apache2/sites-enabled/[...]",
        notify => Service[ "apache2" ],
    }
}
```

Nodes and Templates

- Wait... which server uses which modules?

```
/etc/puppet/modules/apache2/manifests/dev.pp
/etc/puppet/manifests/nodes.pp
/etc/puppet/manifests/templates.pp
```

```
class basic {
    include ntp
    include mc
    include screen
    include less
    include sysstat
    include vim
    include users::admin
    include users::absent
    include sudo
    include ssh
    include hosts
    include mdadm
    include curl
    include cacertificates
    include postfix::install
    [...]
}
```

```
node „build“ {
    include basic
    include apache2::dev
    include jenkins
}
```

} Template
} Modules

... but templates and modules are actually both just classes, only stored in different places

Facts: Example usage

- Adapt manifests to different environments

Destination file name may differ from source file name

```
file { "/etc/mysql/conf.d/mysqld_replication.cnf":
    require => Package[ "mysql-server" ],
    owner   => root,
    group   => root,
    mode    => 644,
    source  => [ "puppet:///mysql/mysqld_replication.cnf.$hostname",
                 "puppet:///mysql/mysqld_replication.cnf" ],
}
```

Puppet inserts „hostname“ fact here

Fallback in case the more specific config file was not found (a feature of the „file“ type, does not have anything to do with facts)

Doodle

Facts: Some Facts

```
worker:~# facter
architecture => amd64
domain => doodle.com
facterversion => 1.5.7
hardwaremodel => x86_64
hostname => worker
is_virtual => false
kernel => Linux
kernelversion => 2.6.32
lsbdistcodename => squeeze
physicalprocessorcount => 2
processorcount => 8
puppetversion => 2.6.2
timezone => CET
uptime_days => 81
[...]
worker:~#
```

- ... or write your own (easy)
- Bonus: Puppetmaster provides easy-to-use JSON/REST API to access facts about all clients!

Doodle

Facts: Pending Updates

Some JavaScript...

```
$.ajax({
    url: "/puppet/facts_search/search?facts.needsreboot=true",
    headers: {
        Accept : "pson"
    },
    dataType: "json",
    success : function(data) {
        data.sort();
        $.each(data, function(index, value) {
            $("#needreboot").append($("<tr/>").append($("<td/>").text(value)));
        });
    }
});
$.ajax({
    url: "/puppet/facts_search/search?facts.pendingupdates.ne=",
    headers: {
        Accept : "pson"
    },
    dataType: "json",
    success : function(data) {
        data.sort();
        $.each(data, function(index, value) {
            var row = $("<tr/>");
            row.append($("<td/>").text(value));
            $("#pendingupdates").append(row);

            $.ajax({
                url: "/puppet/facts/"+value,
                headers: {
                    Accept : "pson"
                },
                dataType: "json",
                success : function(data) {
                    row.append($("<td/>").text(data.values.pendingupdates));
                }
            });
        });
    }
});
```

... some Ruby and some Bash scripts...

```
Facter.add("pendingupdates") do
  setcode do
    Facter::Util::Resolution.exec(
                  '/opt/doodle/pendingupdates.sh')
  end
end
```

```
#!/bin/bash
# the daily apt-get update is handled
# by the /etc/cron.daily/apt cronjob
echo `aptitude -q -F%p --disable-columns search ~U`
```

... put mod_proxy with client cert auth in front of your puppetmaster, add a puppet manifest and you get:

Template:Serveractions

Host(s) to be rebooted
foo1.doodle.com
foo2.doodle-bar.com
foo3.doodle.com

Host	Pending updates
foo1.doodle.com	linux-image-3.2.0-0.bpo.4-amd64
foo4.doodle.com	jenkins
foo5.doodle.com	linux-image-3.2.0-0.bpo.4-amd64
foo7.doodle.com	linux-image-3.2.0-0.bpo.4-amd64
foo8.doodle.com	linux-image-3.2.0-0.bpo.4-amd64
foo11.doodle.com	libxenstore3.0 linux-image-3.2.0-0.bpo.4-amd64 xen-hypervisor-4.0-amd64 xen-utils-4.0 xenstore-utils
foo12.doodle.com	linux-image-3.2.0-0.bpo.4-amd64 xen-hypervisor-4.0-amd64

Doodle

Puppet @ Doodle

- Started 2010
- Had to re-arrange modules at some point (but for the better!) - Puppet still evolves
- Covers entire base installation (users, tools, SSH settings, firewall, backup, Munin, ...)
- Covers 50-90% of special configuration, depending on service
- Can install XEN dom0 (incl. kernel, reconfiguring network interfaces) and bootstrap domU (incl. Puppet in domU!)

... too bad the Debian XEN packages are quite broken atm :(

Some Random Tips and Tricks

- Properly configure your DNS (both host name and domain)
- Make sure your servers can access your puppetmaster at puppet.yourdomain.com
- Use conf.d directories (together with file{}) whenever possible
  - Use augeas{} if not
  - Use exec{} with grep/sed for really nasty config file issues
  - DO NOT just overwrite complex config files
- Do not create more than one module per service
  - Use subclasses or facts if your servers need different configuration
- Try to use the same manifests/configuration on all of your servers
- Take your time to do the depencencies properly
- On Debian, you need to manually enable the puppet daemon in /etc/default/puppet. The output goes to /var/log/daemon.log.

Doodle

TMTOWTDI

Even though Puppet uses Ruby, always keep in mind:

There's More Than One Way To Do It™

Which, in this context, is actually just an euphemism for „The examples used in this presentation may not use the latest or best techniques to achieve their goal (or may even be downright wrong), but that's the way we do it at the moment and it kinda works“.

Doodle

Your servers... and beyond

- EC2 has images with puppet pre-installed: Easily duplicate parts of your infrastructure in the cloud!
- What about your workstations and laptops? Even Windows is supported! (I haven't tried that, though)
- VPN gateways, office routers, ...

Doodle

Pointers

http://www.puppetlabs.com/

Their web site.

http://docs.puppetlabs.com/references/latest/type.html

Type reference. You'll need it.

http://www.vagrantup.com/

Vagrant, a virtual environment that simplifies Puppet manifest testing. You may want to use it (I don't).

http://blog.wikimedia.org/2011/09/19/ever-wondered-how-the-wikimedia-servers-are-configured/

How Wikimedia does it.

mailto:dg@doodle.com

If you want more information about what we do.