Puppet for Dumps maintainers

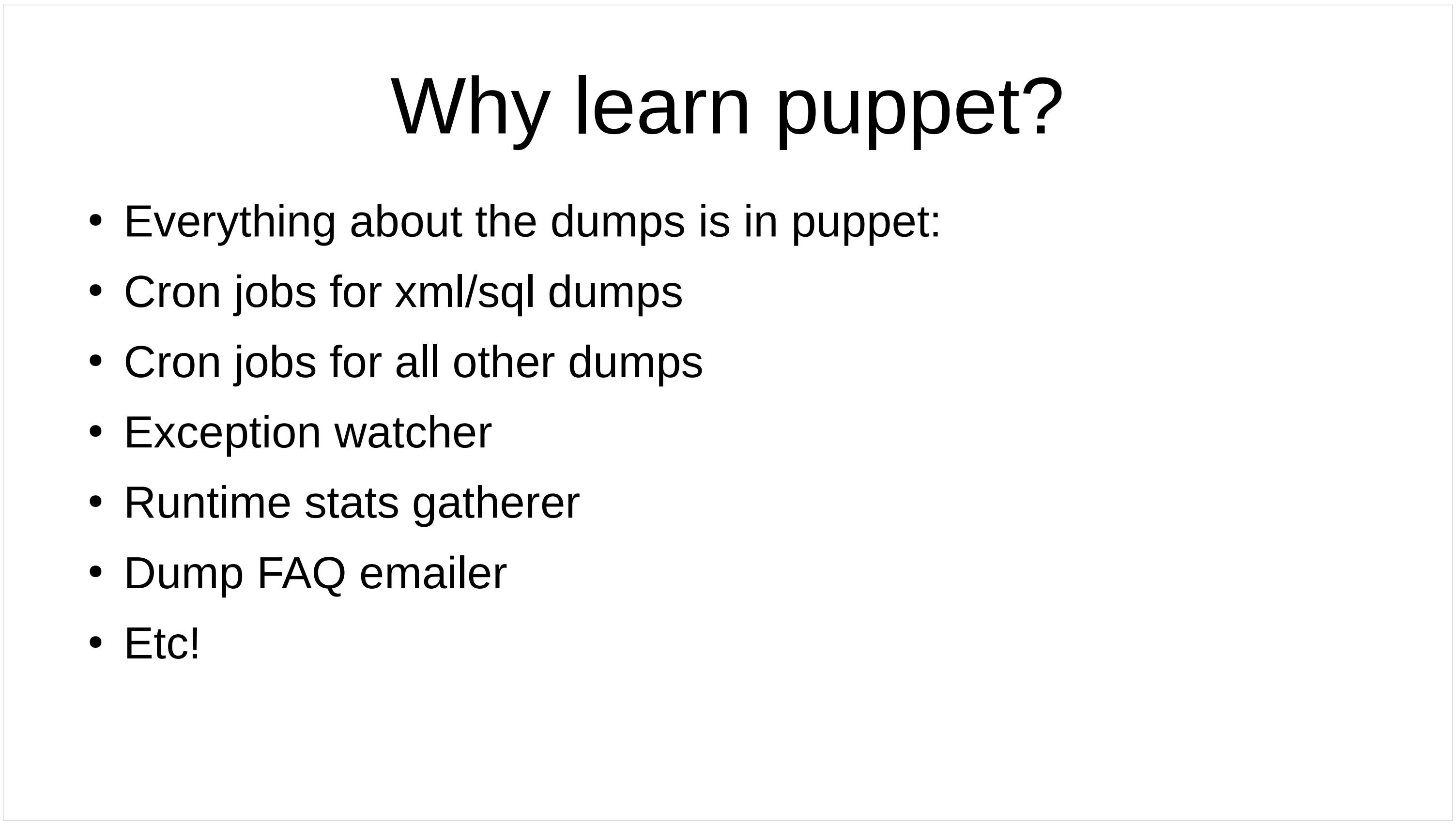
V 0.2

I noticed that only a few people have Puppet skills marked in the skill matrix so it seemed like a good idea to do a talk.

50 (+10) minutes of Puppet

- Why should you care?
- What is Puppet?
- Cabilities
- Approach to code
- Language
- Syntax and style
- Default variable values (Hiera)
- Facter
- Classes
- More classes
- Roles, profiles, modules
- Testing with PCC (puppet compiler)
- Testing in deployment-prep
- Merging changes
- Effects of a change
- Disabling puppet
- The private repo
- Q&A

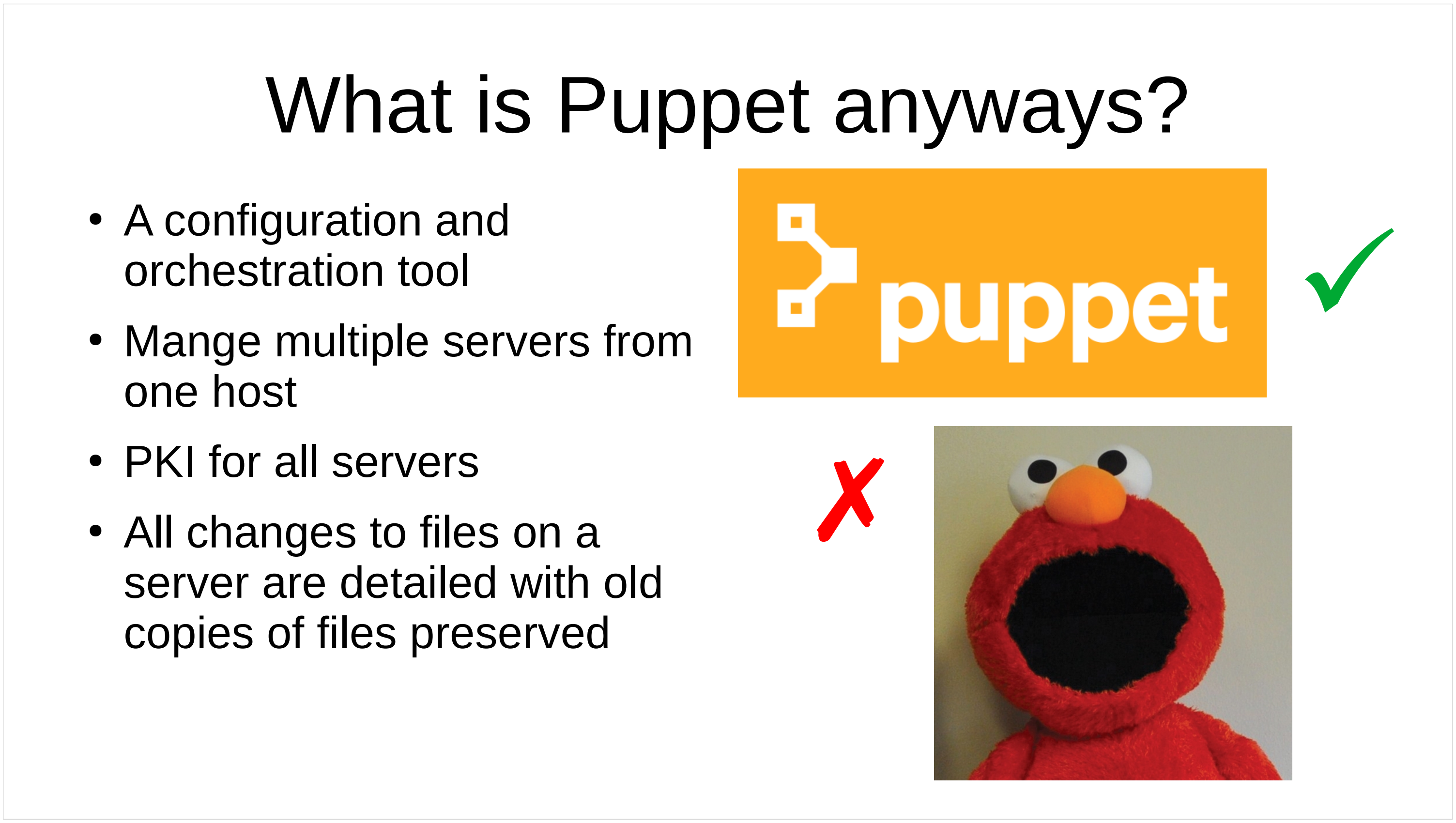
Here’s what we’re going to fly through.



Why learn puppet?

- Everything about the dumps is in puppet:
- Cron jobs for xml/sql dumps
- Cron jobs for all other dumps
- Exception watcher
- Runtime stats gatherer
- Dump FAQ emailer
- Etc!

Besides puppet having All The Things for dumps, it’s also dead useful for any services and other stuff, so once you learn it you’ll never go back.



What is Puppet anyways?

- A configuration and orchestration tool
- Mange multiple servers from one host
- PKI for all servers
- All changes to files on a server are detailed with old copies of files preserved

puppet ✓

✗

It lets you set up things after an initial install, and, with care, make changes to installed configs, packages, and services.

There’s a puppet CA cert and each client gets a cert to keep infiltrators offa yer stuff.

It’s a little inconvenient to find sometimes but old versions of files and such are stashed in a so-called “file bucket” before being updated during a puppet run, and this has saved me more than once.

What Puppet can do

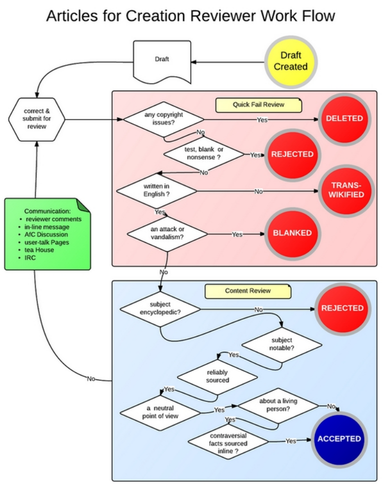
- Create configuration files
- Install packages
- Set up cron jobs and systemd units
- Start and stop services
- Set up firewall rules
- Create users and add their ssh keys
- Make, export or mount NFS shares
- And much much more

Coordinating many systems harmoniously

It’s got a buttload of features, it’s cross platform so all you have to think about are resources, classes, and maybe some library functions if you want that.

Puppet uses a declarative approach

- You declare the state you want your system(s) to be in
- Puppet determines how to get there
- Order of execution of statements in your code is not a given!
- “before”, “after”, “require”
- Example: make sure a package required by some service is installed before the service is started



It’s not quite this bad. Status: it’s complicated.

This is a BFD. When we write code we like to think “Now statement A runs, then statement B, then statement C”. Nope. Puppet does not do that. You have to be explicit about requring things to be done in a certain order. We used to have to do multiple runs in order to get puppet to complete the catalog without errors after initial installs, because we hadn’t figured out how to get that right. For small stuff though, and now that the base infra just works, you get used to it.

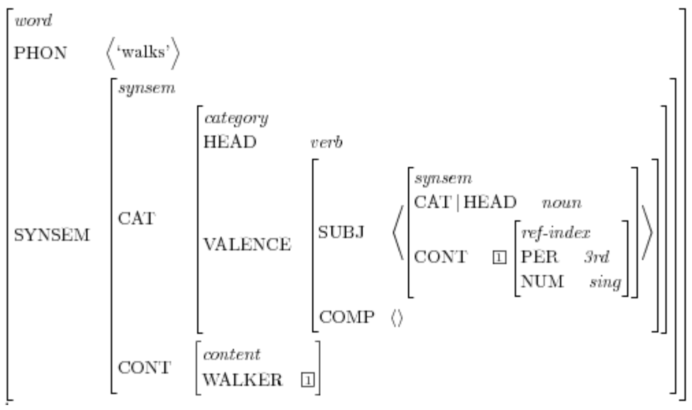
Puppet language

- Ruby-like for manifests
- Snippets of Ruby in templates
- Full Ruby in library functions
- You don’t have to learn Ruby (I didn’t!)

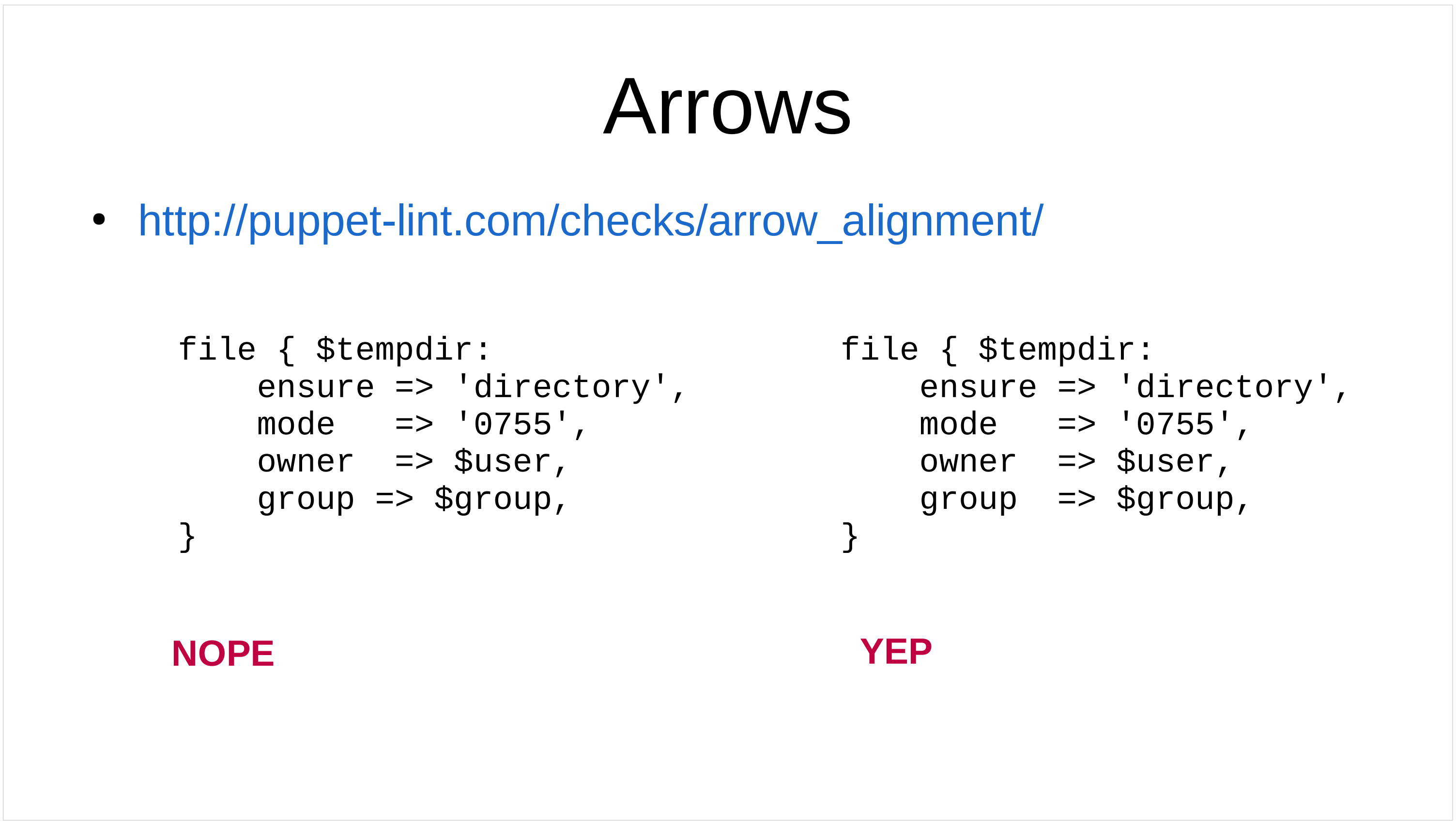
Ruby isn’t all that difficult, but it does make my eyes itch; ymmv, and may folks like it a lot. The puppet manifests (code files) are pretty clean though, and easy to understand.

Syntax, linting

- puppet parser validate
- puppet-lint
- quotes and variables in strings
- arrow spacing
- 4-space indentation
- classes, profiles, roles… later.



Puppet-lint is your best friend, unless you like cleaning up after jenkins a whole lot.



Arrows

- http://puppet-lint.com/checks/arrow_alignment/

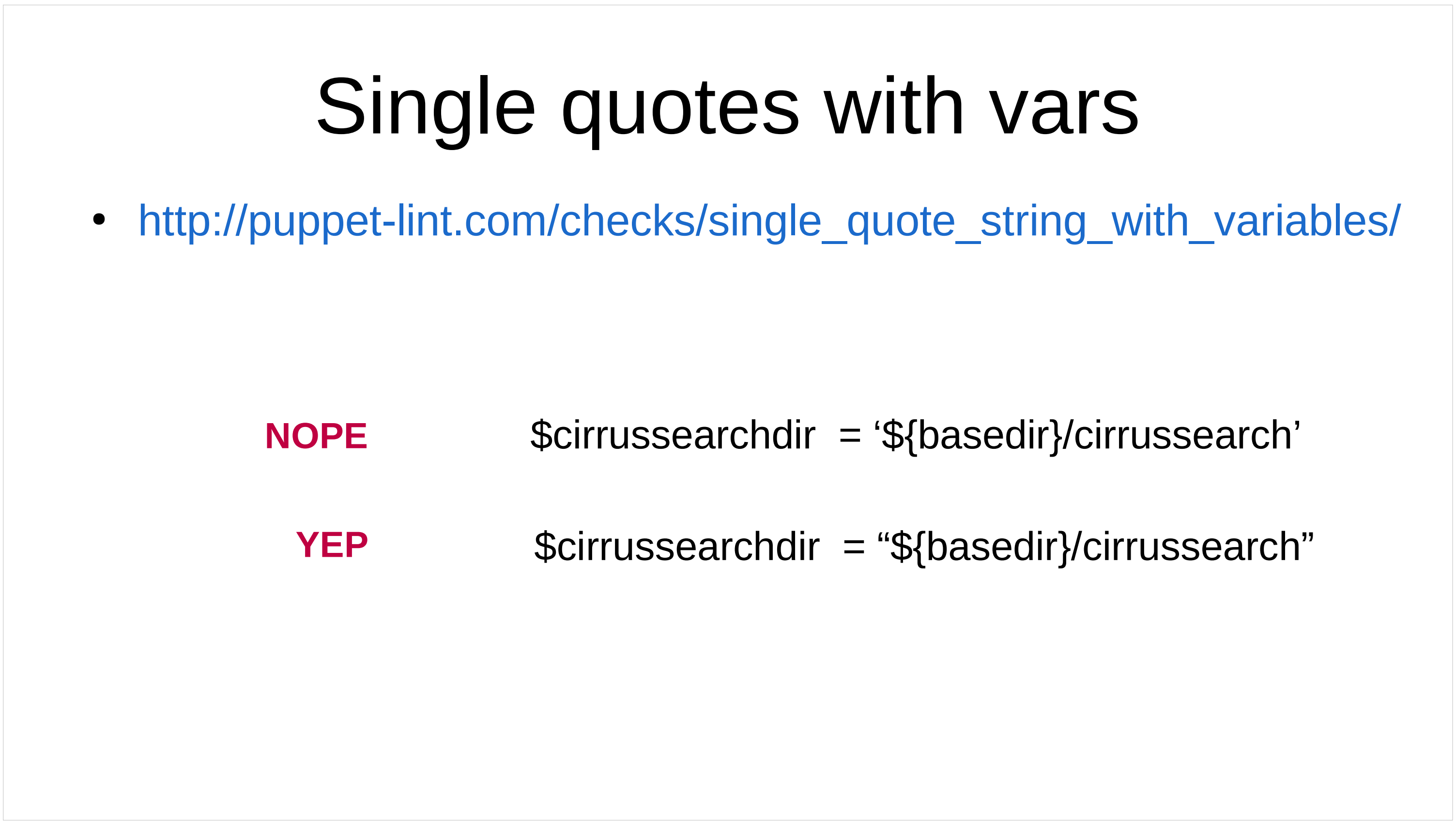
```
file { $tempdir:
    ensure => 'directory',
    mode   => '0755',
    owner  => $user,
    group => $group,
}
```

NOPE

```
file { $tempdir:
    ensure => 'directory',
    mode   => '0755',
    owner  => $user,
    group  => $group,
}
```

YEP

There’s a lot of this kind of thing and it’s irritating, because we wind up making multiple line changes just to add one thing. But, in the long run, it’s better for cognitive load.



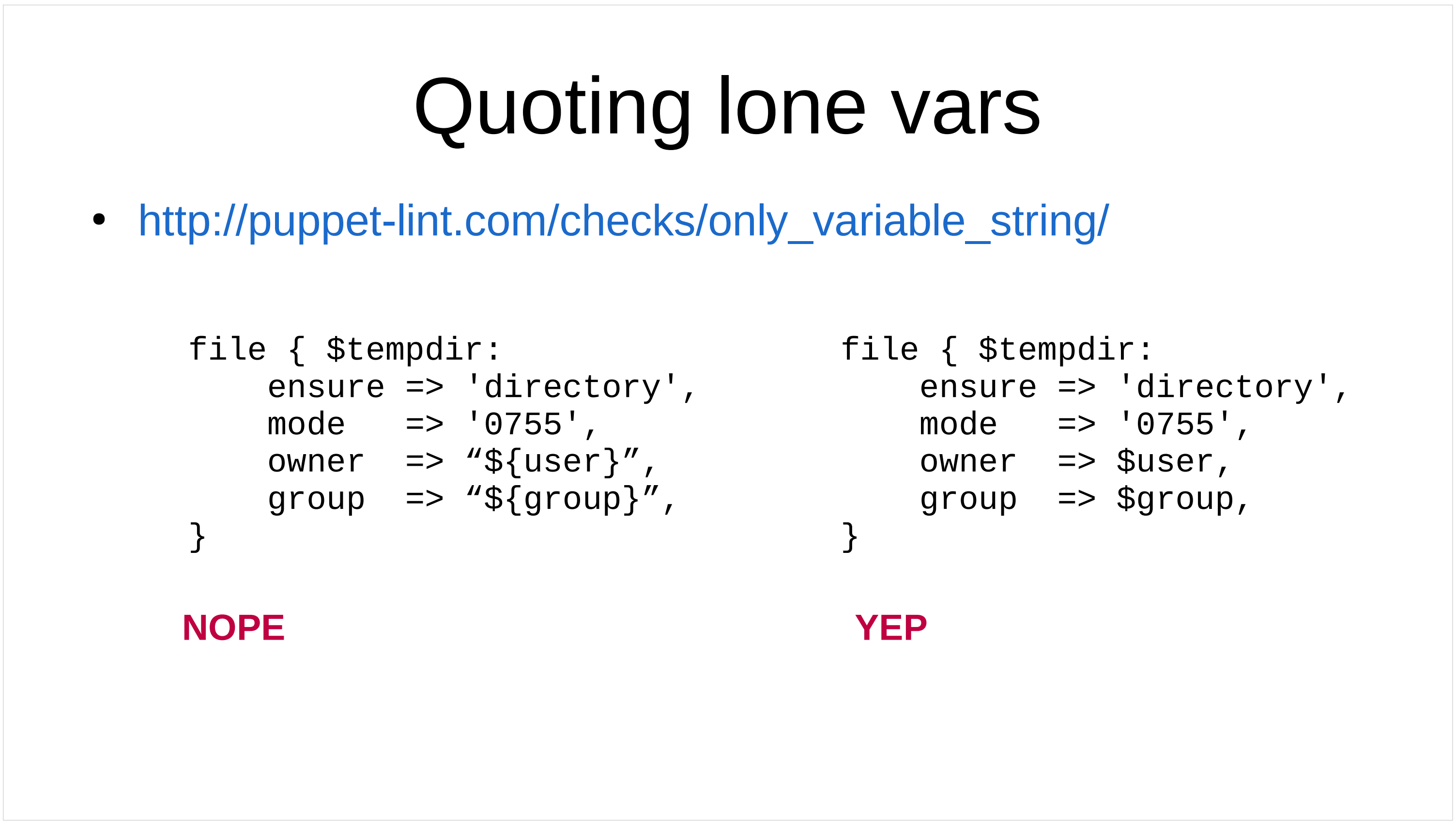
Single quotes with vars

- http://puppet-lint.com/checks/single_quote_string_with_variables/

NOPE $cirrussearchdir = ‘${basedir}/cirrussearch’

YEP $cirrussearchdir = “${basedir}/cirrussearch”

Quoting rules remind me of Bash to some degree.



Quoting lone vars

- http://puppet-lint.com/checks/only_variable_string/

```
file { $tempdir:
    ensure => 'directory',
    mode   => '0755',
    owner  => “${user}”,
    group  => “${group}”,
}
```

NOPE

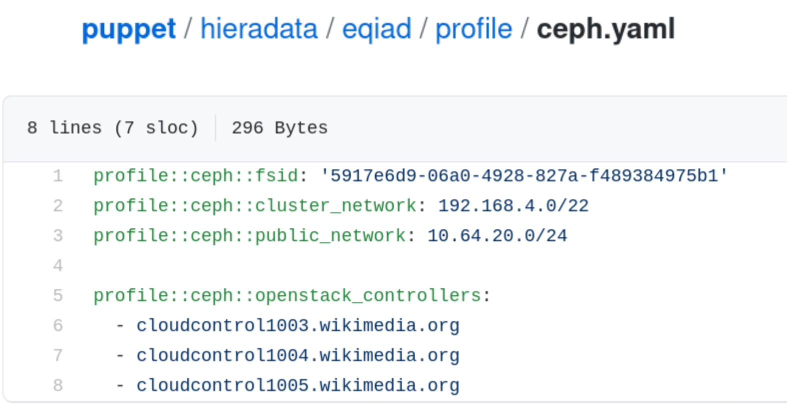
```
file { $tempdir:
    ensure => 'directory',
    mode   => '0755',
    owner  => $user,
    group  => $group,
}
```

YEP

More of that.

Hiera

- Set values for vars for one or a group of servers
- Puppet/hieradata
- Yaml syntax
- Scope explicit in name



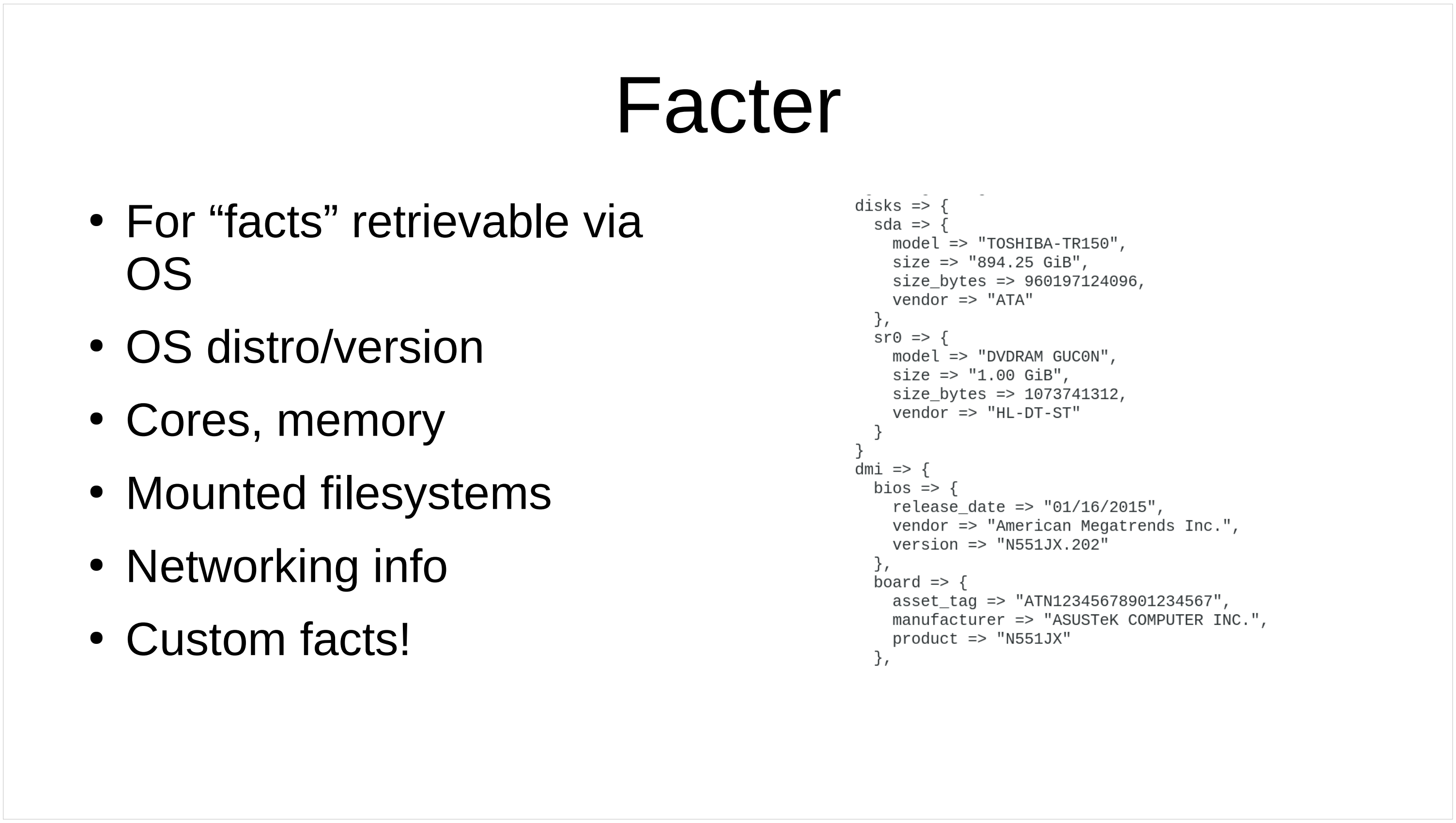
puppet / hieradata / eqiad / profile / ceph.yaml

8 lines (7 sloc) | 296 Bytes

```yaml
profile::ceph::fsid: '5917e6d9-06a0-4928-827a-f489384975b1'
profile::ceph::cluster_network: 192.168.4.0/22
profile::ceph::public_network: 10.64.20.0/24

profile::ceph::openstack_controllers:
  - cloudcontrol1003.wikimedia.org
  - cloudcontrol1004.wikimedia.org
  - cloudcontrol1005.wikimedia.org
```

Restrictions on use, info about lookup scope:
https://wikitech.wikimedia.org/wiki/Puppet_coding#Hiera

This is where you change values of variables around on a per host or cluster or dc basis. The only place you can trip up is getting the scope (namespace) right. I screw this up all the time; other people get it right every time. YMMV. See the link for how that works.



Facter

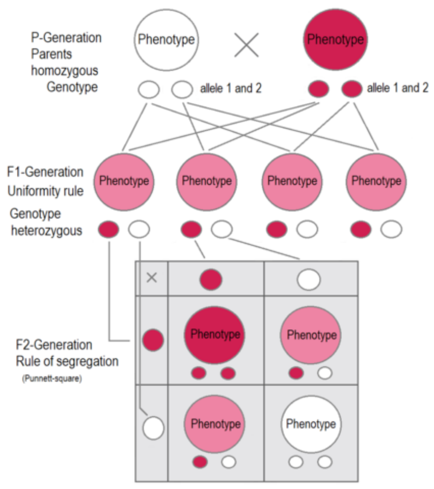
- For “facts” retrievable via OS
- OS distro/version
- Cores, memory
- Mounted filesystems
- Networking info
- Custom facts!

```
disks => {
  sda => {
    model => "TOSHIBA-TR150",
    size => "894.25 GiB",
    size_bytes => 960197124096,
    vendor => "ATA"
  },
  sr0 => {
    model => "DVDRAM GUC0N",
    size => "1.00 GiB",
    size_bytes => 1073741312,
    vendor => "HL-DT-ST"
  }
}
dmi => {
  bios => {
    release_date => "01/16/2015",
    vendor => "American Megatrends Inc.",
    version => "N551JX.202"
  },
  board => {
    asset_tag => "ATN12345678901234567",
    manufacturer => "ASUSTeK COMPUTER INC.",
    product => "N551JX"
  },
```

Facter is one of the most awesome things about puppet. And its extensibility. Want to run more processes on a host because there’s more cores? Here ya go. Need a new fact that puppet doesn’t provide by default? Write your own!

Classes and inheritance

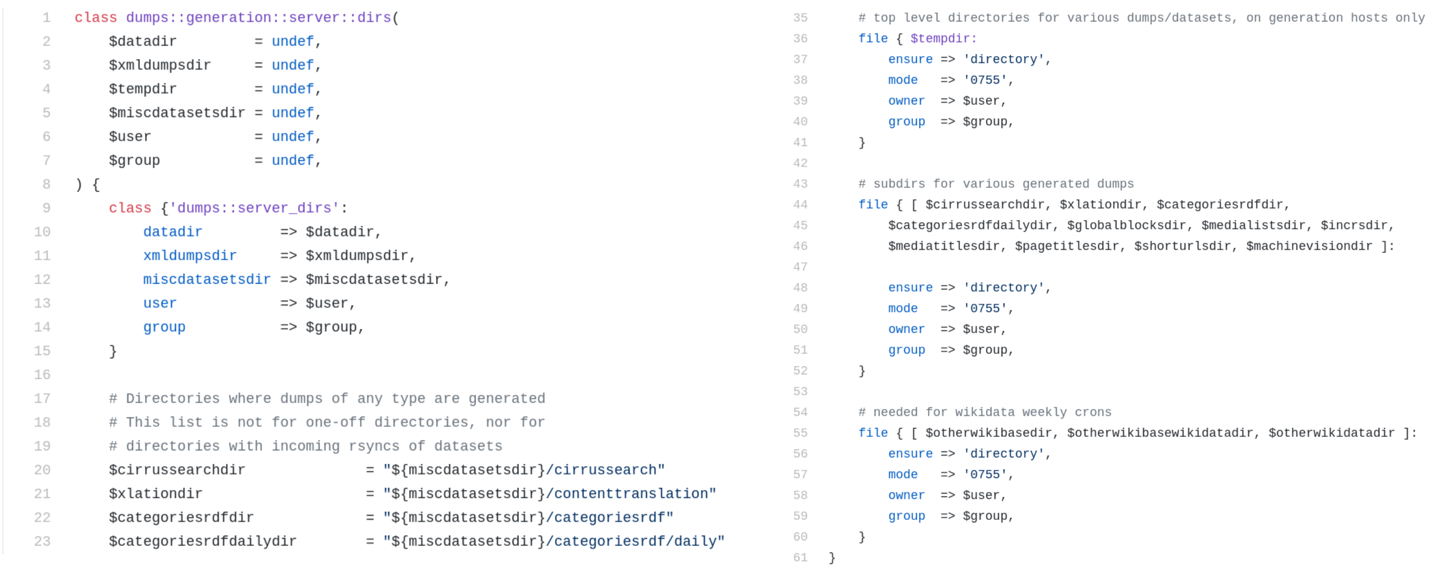
- Short version:
  don’t.
- Long version:
  still don’t.



Even Puppet says don’t:
https://puppet.com/docs/puppet/5.5/lang_classes.html#inheritance

So the base unit of a puppet manifest is the class. And so you might think right away “oh cool, inheritance!” In fact, oh so not cool. Trust me on this. Use classes only as a unit to have a bunch of related statements and resources together, NOTHING ELSE.

Classes as building blocks



```
class dumps::generation::server::dirs(
    $datadir         = undef,
    $xmldumpsdir     = undef,
    $tempdir         = undef,
    $miscdatasetsdir = undef,
    $user            = undef,
    $group           = undef,
) {
    class {'dumps::server_dirs':
        datadir         => $datadir,
        xmldumpsdir     => $xmldumpsdir,
        miscdatasetsdir => $miscdatasetsdir,
        user            => $user,
        group           => $group,
    }

    # Directories where dumps of any type are generated
    # This list is not for one-off directories, nor for
    # directories with incoming rsyncs of datasets
    $cirrussearchdir              = "${miscdatasetsdir}/cirrussearch"
    $xlationdir                   = "${miscdatasetsdir}/contenttranslation"
    $categoriesrdfdir             = "${miscdatasetsdir}/categoriesrdf"
    $categoriesrdfdailydir        = "${miscdatasetsdir}/categoriesrdf/daily"
```

```
    # top level directories for various dumps/datasets, on generation hosts only
    file { $tempdir:
        ensure => 'directory',
        mode   => '0755',
        owner  => $user,
        group  => $group,
    }

    # subdirs for various generated dumps
    file { [ $cirrussearchdir, $xlationdir, $categoriesrdfdir,
        $categoriesrdfdailydir, $globalblocksdir, $medialistsdir, $incrsdir,
        $mediatitlesdir, $pagetitlesdir, $shorturlsdir, $machinevisiondir ]:

        ensure => 'directory',
        mode   => '0755',
        owner  => $user,
        group  => $group,
    }

    # needed for wikidata weekly crons
    file { [ $otherwikibasedir, $otherwikibasewikidatadir, $otherwikidatadir ]:
        ensure => 'directory',
        mode   => '0755',
        owner  => $user,
        group  => $group,
    }
}
```

https://github.com/wikimedia/puppet/blob/production/modules/dumps/manifests/generation/server/dirs.pp

We can look at excerpts from an actual class, so you get the idea of the sort of thing that goes on in here.

Classes are built from other classes

- ...but not from profiles or roles
- ...and only from classes in the same module
- Profiles are built from classes in multiple modules
- Roles are built from multiple profiles

We have a three tier system in production: classes, profiles, roles. Classes are th only thing native to puppet; profiles and roles are just names we use for larger units of things, and names of directories where we put them.

Building with classes

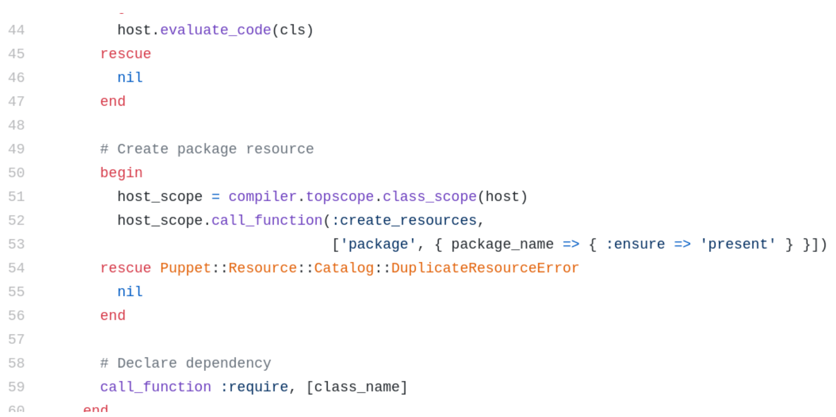
- Include: no params, as often as you want
- Require: no params, as often as you want
- Declare: params, only once

Yes, this is built entirely from Legos.

There's a few different ways to add your class into another one. You can "include classname-here" and do that as many times as you like in the mnifests that will make the catalog for your target hosts. BUT no parameters can be passed. The same is true for "require". Why is this? Because if puppet permitted params, you could include with one set of values in one place and another set in another place and puppet, being declarative and trying to reach your desired state, wouldn't know what to do. What state is that actually? Heh. If you need to pass in specific params, you can only do that ONCE in a catalog and you declare the class with the values for the args, as we've seen earlier.

Resources

- Files, cron jobs, services, packages
- Declare any given resource only once
- require_package() to save headaches

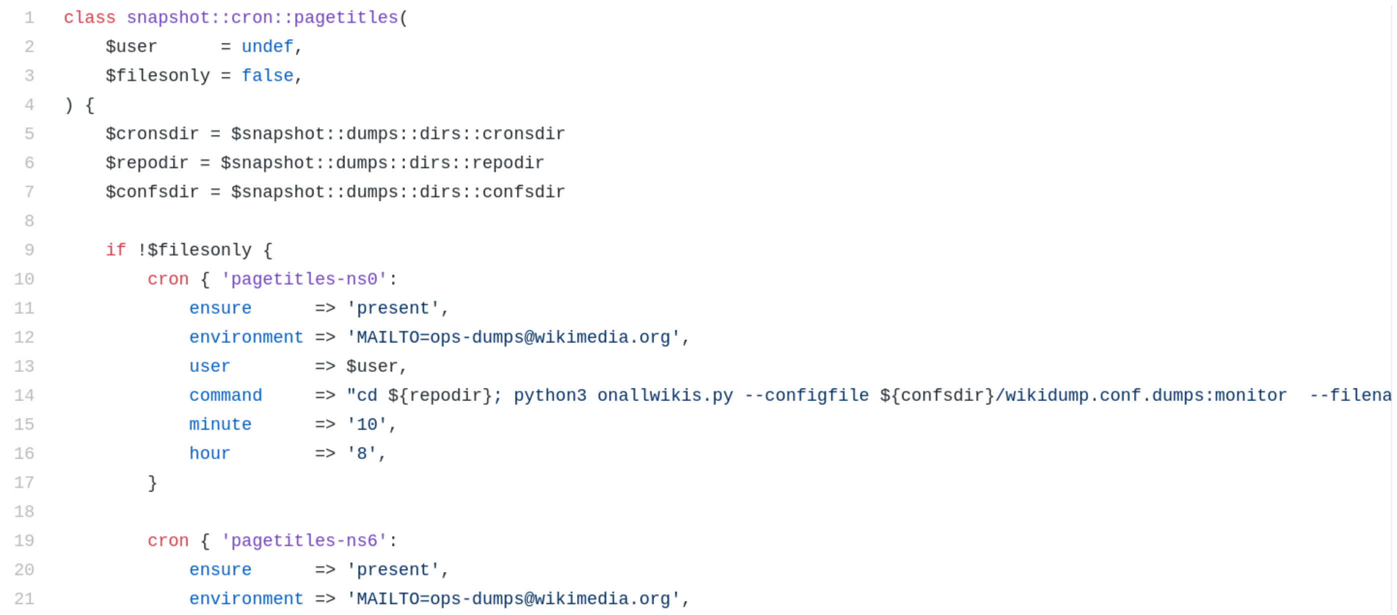


```
      host.evaluate_code(cls)
    rescue
      nil
    end

    # Create package resource
    begin
      host_scope = compiler.topscope.class_scope(host)
      host_scope.call_function(:create_resources,
                               ['package', { package_name => { :ensure => 'present' } }])
    rescue Puppet::Resource::Catalog::DuplicateResourceError
      nil
    end

    # Declare dependency
    call_function :require, [class_name]
  end
```

https://github.com/wikimedia/puppet/blob/production/modules/wmflib/lib/puppet/parser/functions/require_package.rb

Resources also go into classes. These can be all kinds of things, some are built in to puppet like the ones named here, but you can also define your own.

Cron jobs as resources



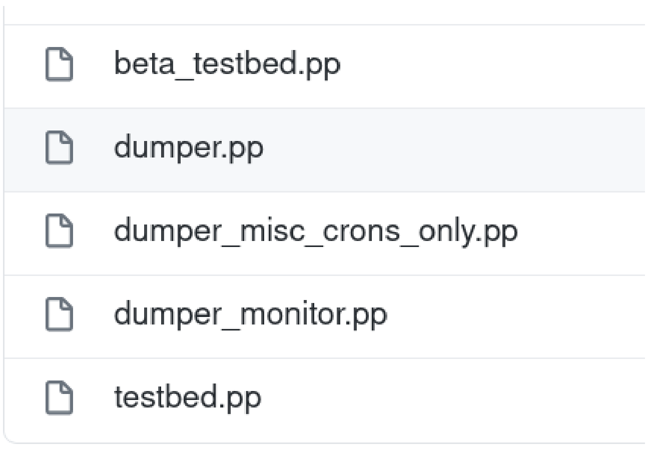
```
class snapshot::cron::pagetitles(
    $user      = undef,
    $filesonly = false,
) {
    $cronsdir = $snapshot::dumps::dirs::cronsdir
    $repodir = $snapshot::dumps::dirs::repodir
    $confsdir = $snapshot::dumps::dirs::confsdir

    if !$filesonly {
        cron { 'pagetitles-ns0':
            ensure      => 'present',
            environment => 'MAILTO=ops-dumps@wikimedia.org',
            user        => $user,
            command     => "cd ${repodir}; python3 onallwikis.py --configfile ${confsdir}/wikidump.conf.dumps:monitor  --filena
            minute      => '10',
            hour        => '8',
        }

        cron { 'pagetitles-ns6':
            ensure      => 'present',
            environment => 'MAILTO=ops-dumps@wikimedia.org',
```

https://github.com/wikimedia/puppet/blob/production/modules/snapshot/manifests/cron/pagetitles.pp

Here's an example of a cron job resource I use.

Roles, profiles, classes

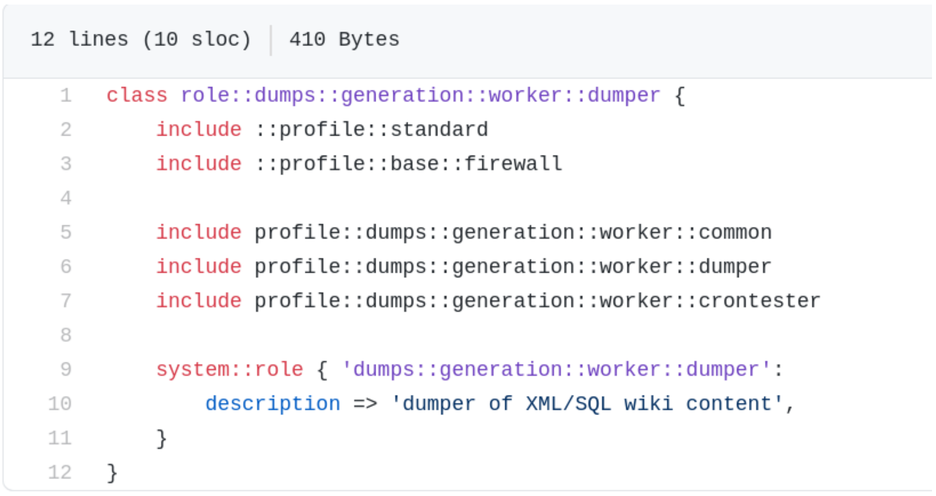
- Classes build other classes and profiles
- Profiles build other profiles and roles
- Roles don't build anything
- ONE ROLE PER HOST



beta_testbed.pp
dumper.pp
dumper_misc_crons_only.pp
dumper_monitor.pp
testbed.pp

So more about our classes/profiles/roles: you want one role per host. Them's the rules, I didn't make 'em. For me in practice this means 4 roles for seven hosts (plus one more for the instance in deployment-prep) but most of these roles are just a few lines of "include" and some profile name. Not a big deal.

Role for xml/sql dumps workers



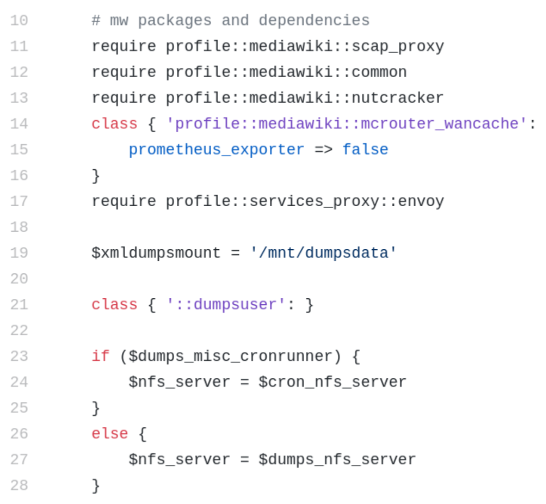
```
12 lines (10 sloc) | 410 Bytes

class role::dumps::generation::worker::dumper {
    include ::profile::standard
    include ::profile::base::firewall

    include profile::dumps::generation::worker::common
    include profile::dumps::generation::worker::dumper
    include profile::dumps::generation::worker::crontester

    system::role { 'dumps::generation::worker::dumper':
        description => 'dumper of XML/SQL wiki content',
    }
}
```

https://github.com/wikimedia/puppet/blob/production/modules/role/manifests/dumps/generation/worker/dumper.pp

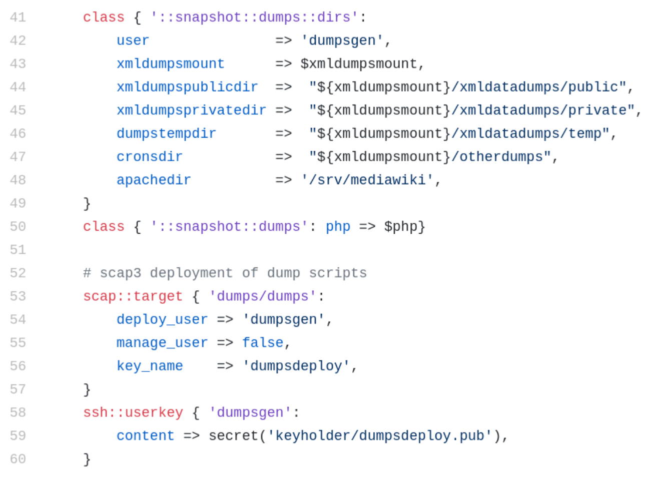
Here’s one of those so you can see.

The first two are standard profiles, that wind up on virtually all hosts. They do things like set up the firewall, set up basic users and ssh keys, set up ntp and promethus, make it a target for cumin for remote commands via ssh, and so on. Only the last three are specific to the dumps.

Profile common to dumps workers



```
# mw packages and dependencies
require profile::mediawiki::scap_proxy
require profile::mediawiki::common
require profile::mediawiki::nutcracker
class { 'profile::mediawiki::mcrouter_wancache':
    prometheus_exporter => false
}
require profile::services_proxy::envoy

$xmldumpsmount = '/mnt/dumpsdata'

class { '::dumpsuser': }

if ($dumps_misc_cronrunner) {
    $nfs_server = $cron_nfs_server
}
else {
    $nfs_server = $dumps_nfs_server
}
```



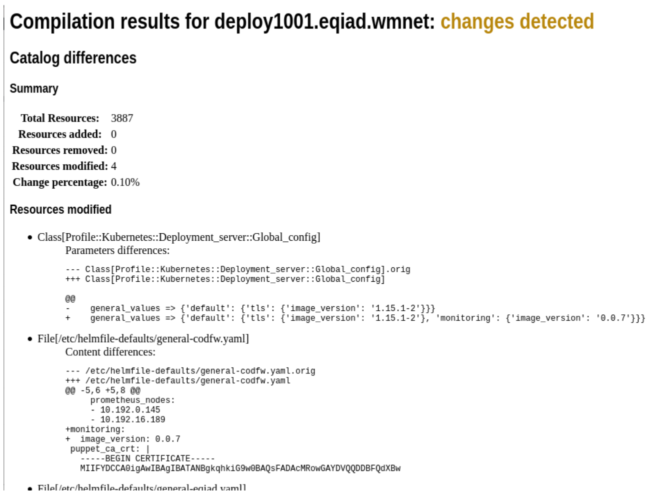
```
class { '::snapshot::dumps::dirs':
    user               => 'dumpsgen',
    xmldumpsmount      => $xmldumpsmount,
    xmldumpspublicdir  =>  "${xmldumpsmount}/xmldatadumps/public",
    xmldumpsprivatedir =>  "${xmldumpsmount}/xmldatadumps/private",
    dumpstempdir       =>  "${xmldumpsmount}/xmldatadumps/temp",
    cronsdir           =>  "${xmldumpsmount}/otherdumps",
    apachedir          => '/srv/mediawiki',
}
class { '::snapshot::dumps': php => $php}

# scap3 deployment of dump scripts
scap::target { 'dumps/dumps':
    deploy_user => 'dumpsgen',
    manage_user => false,
    key_name    => 'dumpsdeploy',
}
ssh::userkey { 'dumpsgen':
    content => secret('keyholder/dumpsdeploy.pub'),
}
```

https://github.com/wikimedia/puppet/blob/production/modules/profile/manifests/dumps/generation/worker/common.pp

Here’s a profile common to all the dump workers.

Puppet Compiler

- Build puppet catalog for gerrit change
- Target specific hosts
- Shows the diff, errors
- Not foolproof but pretty awesome
- Use Wikitech creds to log in



Compilation results for deploy1001.eqiad.wmnet: changes detected

Catalog differences

Summary

Total Resources:	3887
Resources added:	0
Resources removed:	0
Resources modified:	4
Change percentage:	0.10%

Resources modified

- Class[Profile::Kubernetes::Deployment_server::Global_config]
  Parameters differences:

```
--- Class[Profile::Kubernetes::Deployment_server::Global_config].orig
+++ Class[Profile::Kubernetes::Deployment_server::Global_config]

@@
-    general_values => {'default': {'tls': {'image_version': '1.15.1-2'}}}
+    general_values => {'default': {'tls': {'image_version': '1.15.1-2'}, 'monitoring': {'image_version': '0.0.7'}}}
```

- File[/etc/helmfile-defaults/general-codfw.yaml]
  Content differences:

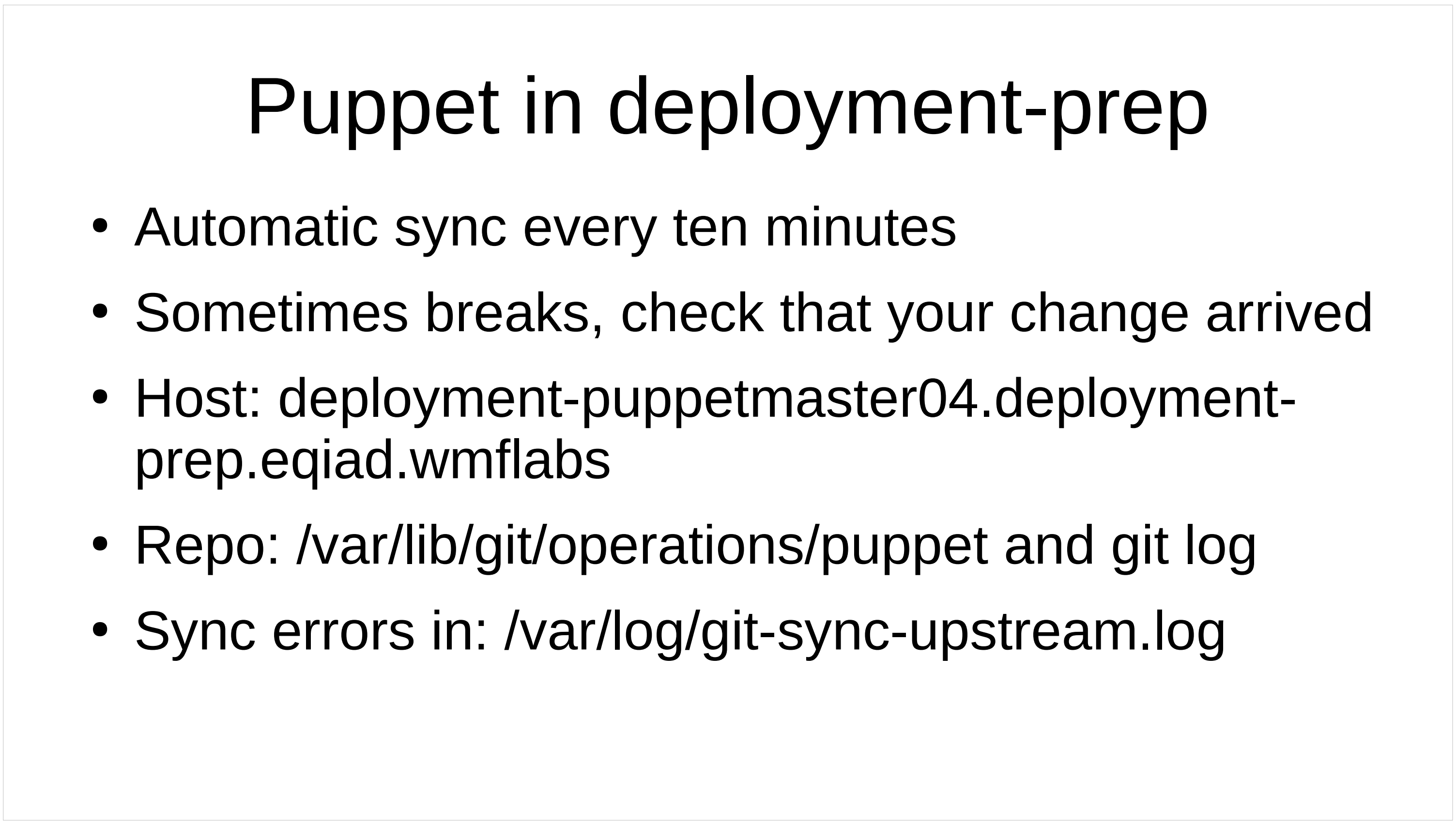
```
--- /etc/helmfile-defaults/general-codfw.yaml.orig
+++ /etc/helmfile-defaults/general-codfw.yaml
@@ -5,6 +5,8 @@
     prometheus_nodes:
     - 10.192.0.145
     - 10.192.16.189
+monitoring:
+  image_version: 0.0.7
 puppet_ca_crt: |
   -----BEGIN CERTIFICATE-----
   MIIFYDCCA0igAwIBAgIBATANBgkqhkiG9w0BAQsFADAcMRowGAYDVQQDDBFQdXBw
```

- File[/etc/helmfile-defaults/general-eqiad.yaml]

https://integration.wikimedia.org/ci/job/operations-puppet-catalog-compiler/build?delay=0sec

This is seriously the awesomest of the awesome right here. If you do much puppet you will use it a lot and love it a lot.

One thing is that the diffs aren't perfect, sometimes if will say "a new resource was added" and the name of the resource (such as a file) but if you want the contents of the file you have to go look at the gerrit change. Still.. awesome!



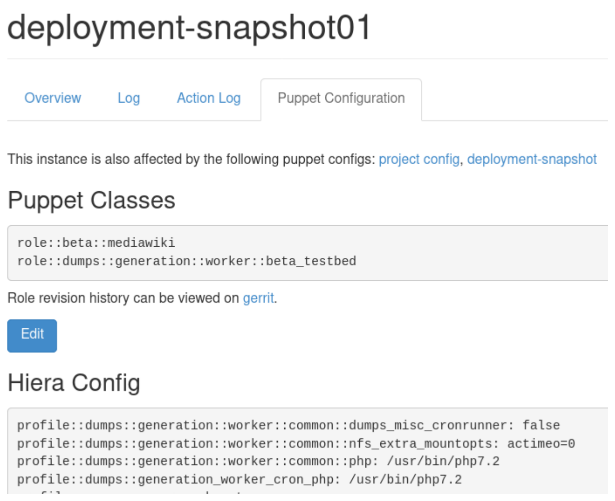
Puppet in deployment-prep

- Automatic sync every ten minutes
- Sometimes breaks, check that your change arrived
- Host: deployment-puppetmaster04.deployment-prep.eqiad.wmflabs
- Repo: /var/lib/git/operations/puppet and git log
- Sync errors in: /var/log/git-sync-upstream.log

By this time you really should have had other people +1 and if you haven’t got +2 in the puppet repo, someone else will have done that too. Nonetheless, testing in deployment-prep is a good thing when you can do it.

Hiera in deployment-prep

- Use Horizon interface
- Check your instance, your prefix-puppet and your project-puppet
- Wikitech creds to log in



This instance is also affected by the following puppet configs: project config, deployment-snapshot

Puppet Classes

```
role::beta::mediawiki
role::dumps::generation::worker::beta_testbed
```

Role revision history can be viewed on gerrit.

Hiera Config

```
profile::dumps::generation::worker::common::dumps_misc_cronrunner: false
profile::dumps::generation::worker::common::nfs_extra_mountopts: actimeo=0
profile::dumps::generation::worker::common::php: /usr/bin/php7.2
profile::dumps::generation_worker_cron_php: /usr/bin/php7.2
```

https://horizon.wikimedia.org/project/instances/898fe007-ec78-465e-b298-d9c57aee7aad/?prev_marker=a82c5f27-a92b-42a6-9f3a-530f9feae621

So there’s an intereface specifically for managing your instances in WMCS, called “horizon”. Hiera settings that aren’t in the production puppet repo will get set here. You can set them at the instance level, or define a “prefix” such that all hosts with that prefix get the setting (example: deployment-snapshot” would cover snapshot0nnn hosts, instead of just one instance). You can also add settings to the deployment-prep project as a whole but in this case be very careful about the namespace. You don’t want that setting to land on an appserver and have some weird impact.

Puppet-merge in production

- Last chance after +2 and submit to abort
- Check the diff to be sure it’s what you added
- “Multiple merge?” = someone else trying at the same time, GET APPROVAL

Really this will have been done before going to deployment prep. But anyways.

After +2 and submission you still need to get on the puppetmaster in production, whichever is active, and puppet-merge. That’s literally the command. Sudo -i gets you root to do this.

You’ll get a diff of your changes; make sure it really is your diff (this is a last security protection).

If you have bad timing, someone else will have just submitted too (or will have forgotten to puppet-merge their change). Ask in -operations or one of the sre channels and find out who, verify BEFORE you just merge their stuff too.

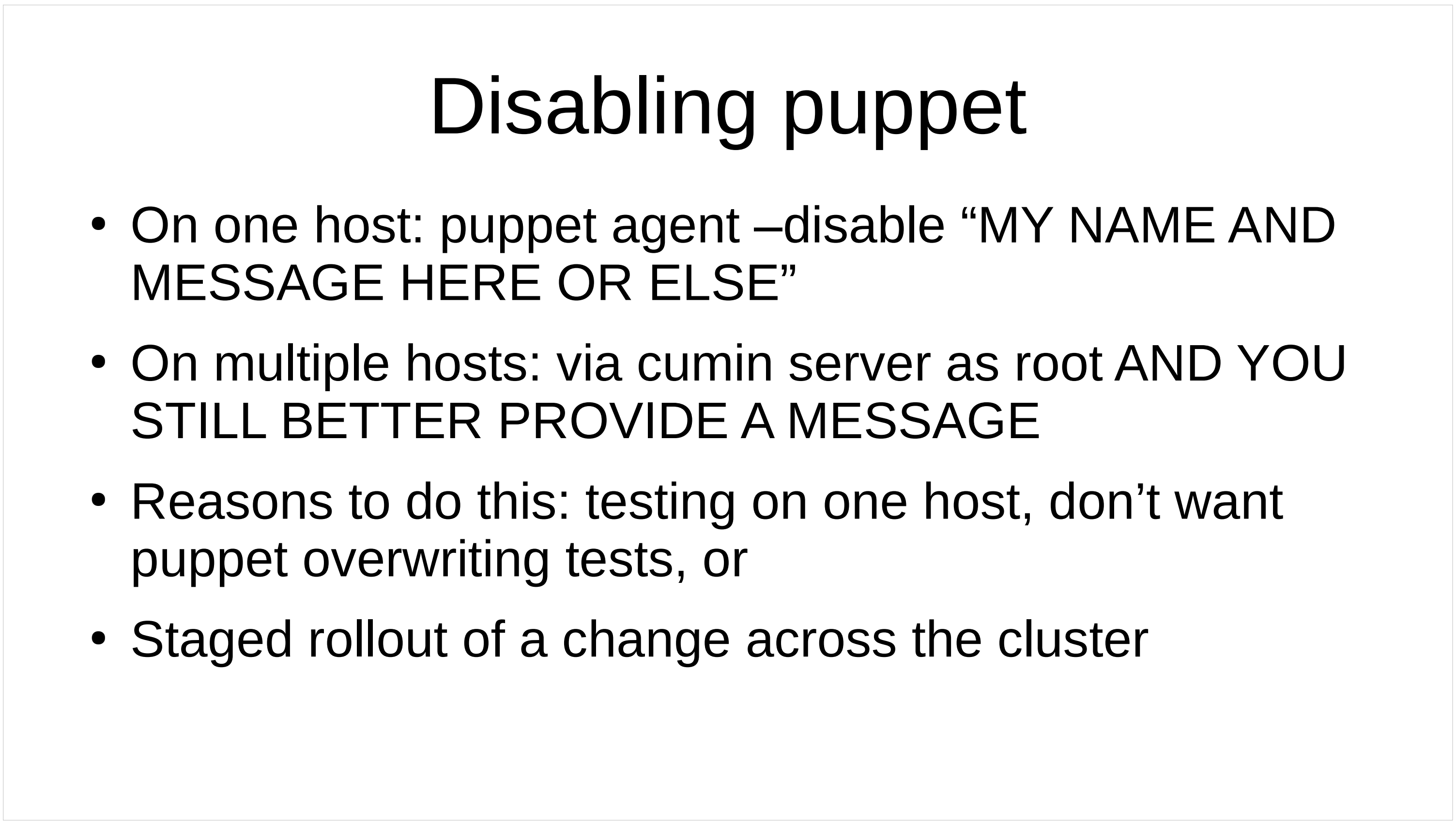
Testing in production (heh)

- Do a live run on a target
- (root) puppet agent --test
- Output: /var/log/puppet.log
- Trival changes aren’t
- No-ops aren’t either

Yes, Virginia, we test in production. Heavy sigh.

Once your change is merged, it’s very good to hop onto a target host and do a live puppet run right there. If there’s an error you can catch it right away, instead of having someone poke you later when icinga alerts about it. And especially if your change affects many hosts, this is only polite. Even the most trivial changes can turn out to not have been quite that trivial :-)

ALSO ALSO ALSO:
Not all changes you make are cleanly applied. Example: change the name of a puppetized cron job and see what happens. Spoiler: old one runs, NEW ONE ALSO RUNS, your stuff breaks. CHECK EVERYTHING.

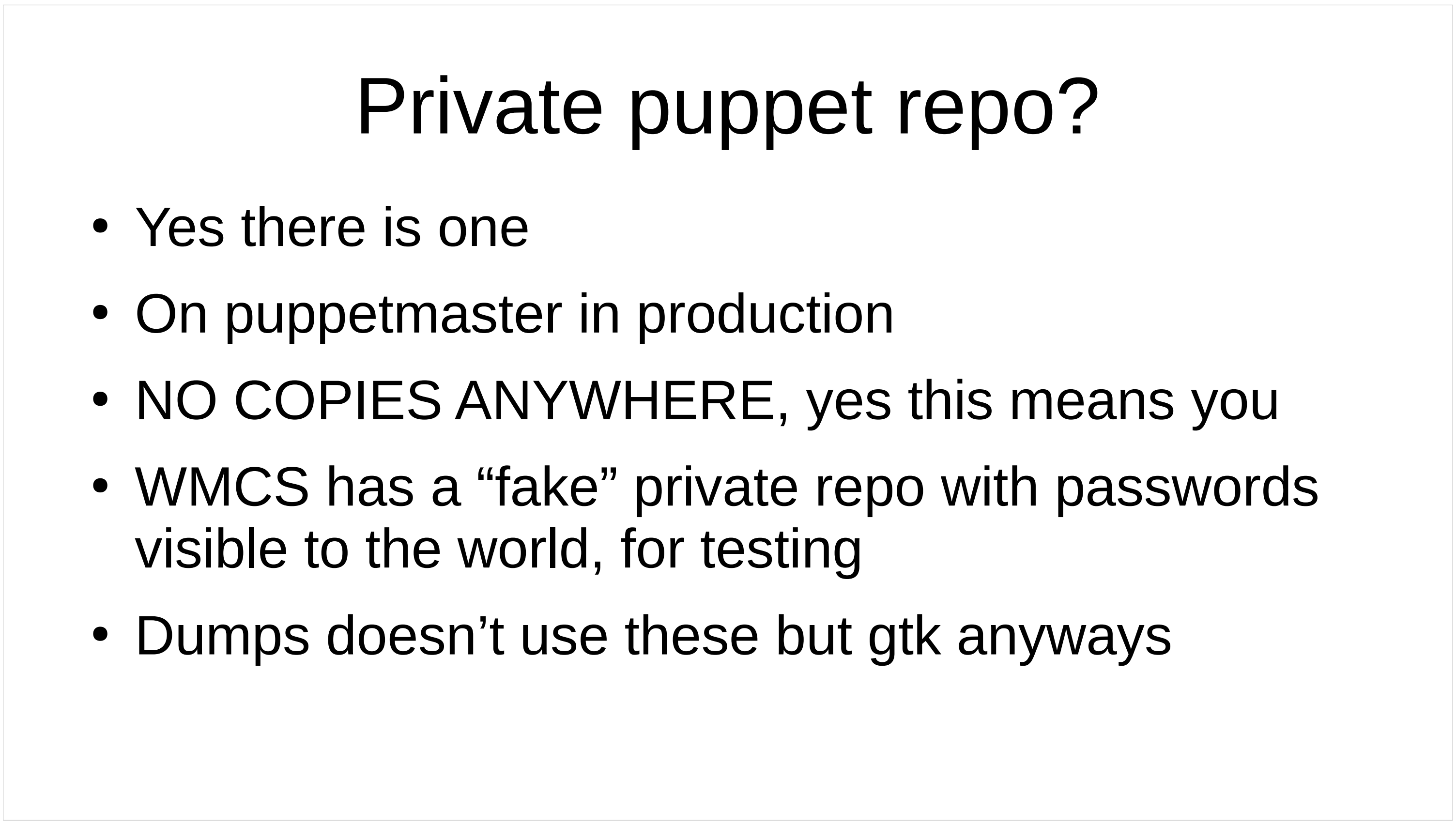


Disabling puppet

- On one host: puppet agent –disable “MY NAME AND MESSAGE HERE OR ELSE”
- On multiple hosts: via cumin server as root AND YOU STILL BETTER PROVIDE A MESSAGE
- Reasons to do this: testing on one host, don’t want puppet overwriting tests, or
- Staged rollout of a change across the cluster

Puppet can be disabled when you are doing a staged rollout of a change that should not go to all hosts at once, or when you are doing a local test or investigation and puppet would otherwise overwrite your changes.

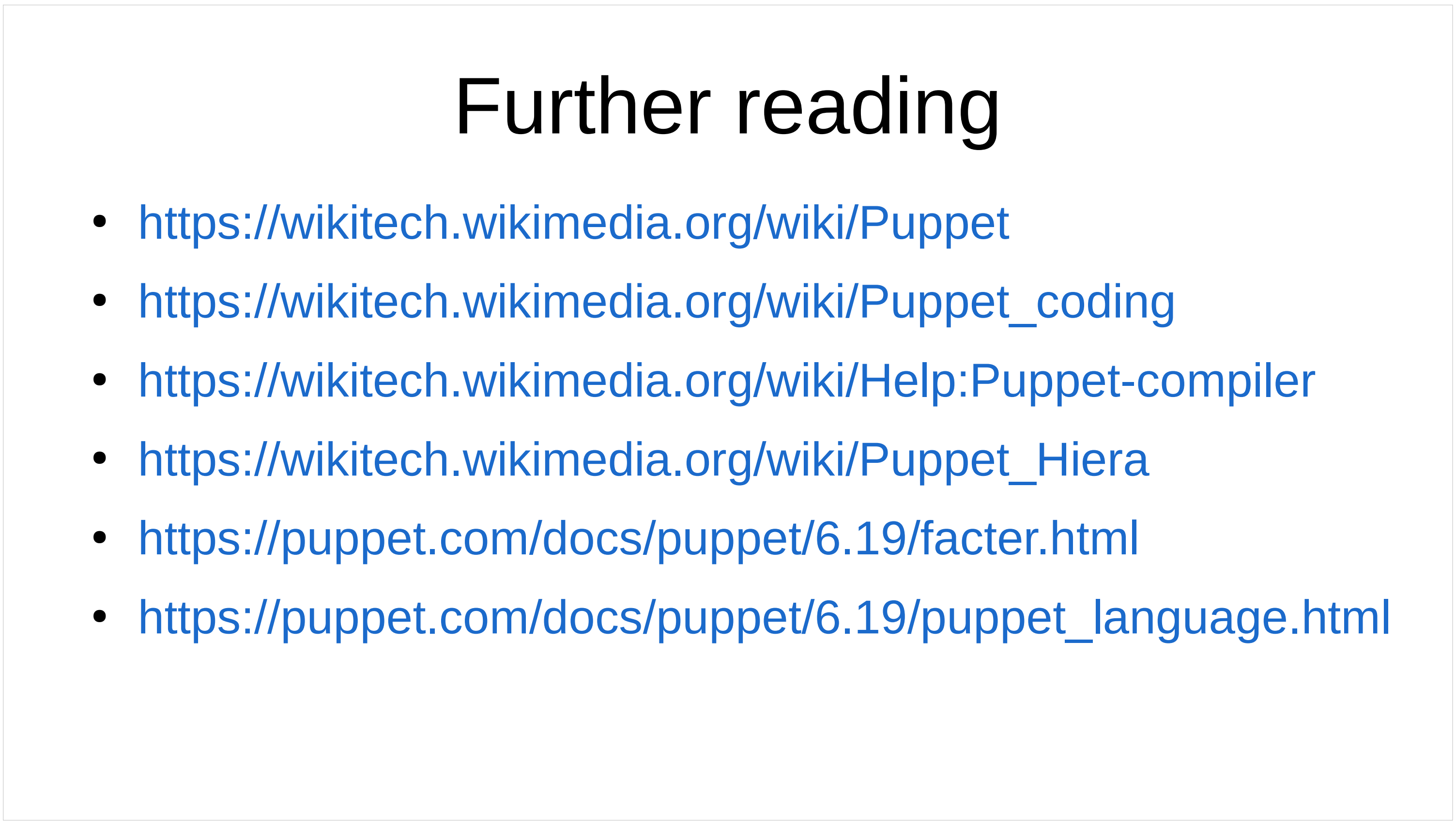
If you do this, IT IS EXTREMELY IMPORTANT for you to add your name and a message about WHY you are disabling puppet. Otherwise when icinga alerts about puppet not having run, or the nxt time puppet needs to go around with an important fix and your host(s) don’t get it, people will be unhappy and rightly so.



Private puppet repo?

- Yes there is one
- On puppetmaster in production
- NO COPIES ANYWHERE, yes this means you
- WMCS has a “fake” private repo with passwords visible to the world, for testing
- Dumps doesn’t use these but gtk anyways

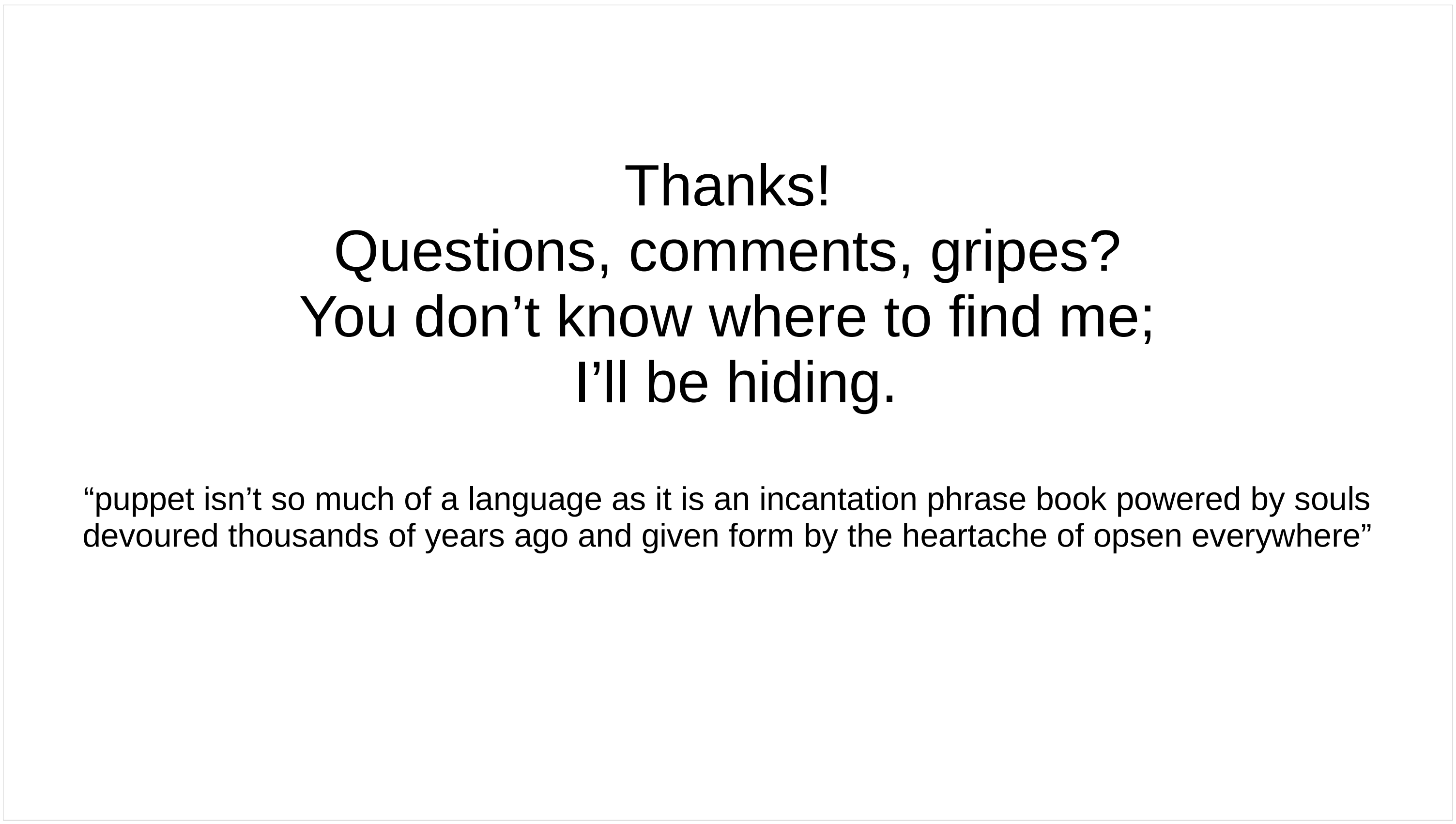
Some stuff is secret. That’s not in our public puppet repo. It is in a private one that lives only on the production puppetmaster. You probably won’t ever need this but it’s nice to know this exists. WMCS does have an equivalent, but all the stuff in there is public. NEVER PUT SOMETHING SECRET inthe WMCS repo!



Further reading

- https://wikitech.wikimedia.org/wiki/Puppet
- https://wikitech.wikimedia.org/wiki/Puppet_coding
- https://wikitech.wikimedia.org/wiki/Help:Puppet-compiler
- https://wikitech.wikimedia.org/wiki/Puppet_Hiera
- https://puppet.com/docs/puppet/6.19/facter.html
- https://puppet.com/docs/puppet/6.19/puppet_language.html

I feel like I jsut barely scratched the surface. Oof! Anyways, here is more reading, don’t get overwhelmed though. You can get started on small things easily enough, copy-pasta works wonders and I am always available for questions or reviews. And there are some other folks on the team who have puppet knowlege too.



Thanks!
Questions, comments, gripes?
You don’t know where to find me;
I’ll be hiding.

“puppet isn’t so much of a language as it is an incantation phrase book powered by souls devoured thousands of years ago and given form by the heartache of opsen everywhere”

Yeah so I started doing these slides at 11 am and now it’s 3:40 pm and I’m almost done and there is no time for a dry run let alone get credits in there… AAAUUUGGHH.

More seriously you do know where to fin me, all the usual channels and phab. So… see you there!

Credits

All images from commons.wikimedia.org or derived from them, or my own screenshots.

- File:Dof_blocks_f22.jpg
- File:Flow_chart_for_flow_in_AfC_on_english_wikipedia_(2).png
- File:Hajrullah_SYLA,_Composer_and_Conductor.jpeg
- File:Heinold%u2019s_First_and_Last_Chance_2007.jpg
- File:Intermediate_inheritance_P_-_F1_-_F2.png
- File:LEGO_Notre_Dame_de_Paris_1.jpg
- File:Puppet_Logo.svg
- File:Ruby_logo.svg
- File:Tux_Paint_t-shirt_01.svg
- File:USMC-17547_cropped.jpg
- File:Walks-avm.png
- https://bash.toolforge.org/quip/AU_3cMwj1oXzWjit5obk