Hierarchical Systems Policy Management in a Puppet/LDAP Environment

Patrick Paul

Puppet Camp LA – February 22, 2013

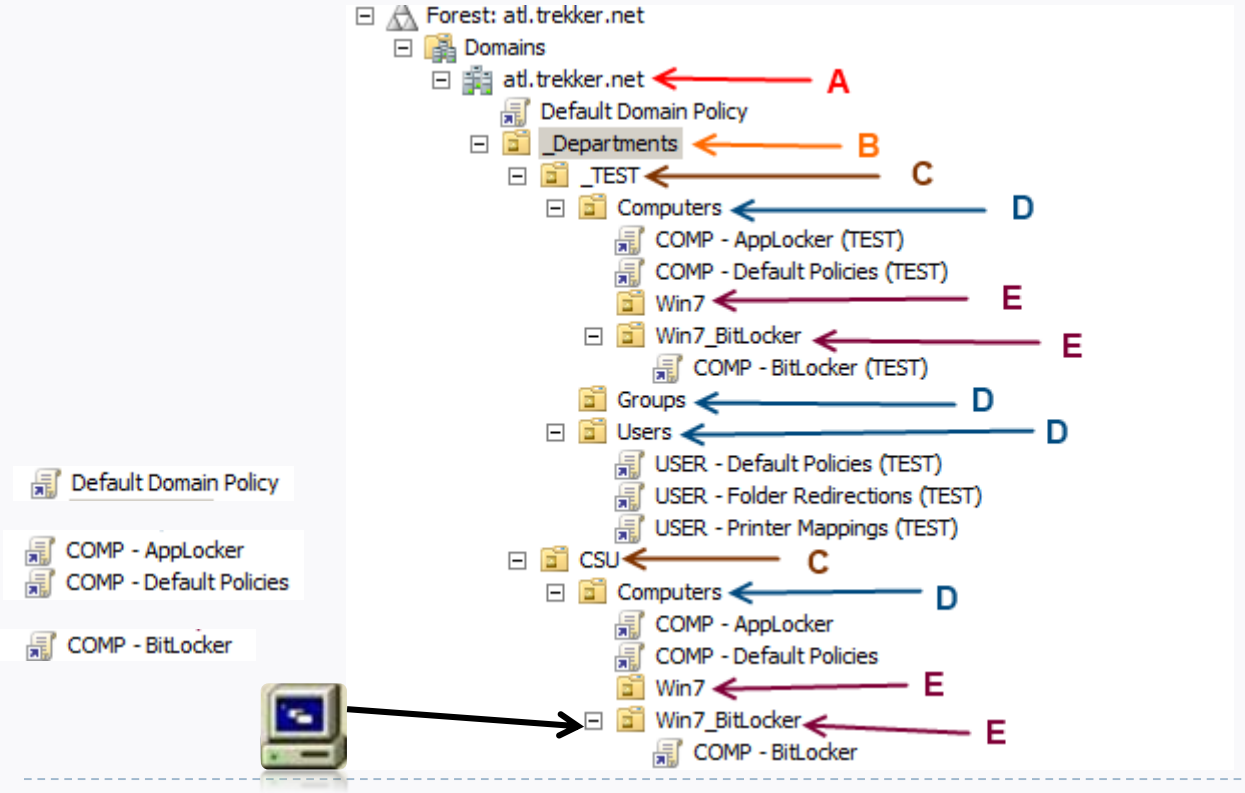
Format

- The Then:
  Design inspirations from hierarchical management (AD)
- The Now:
  Taking those same methods while using Puppet
- The How:
  How this was implemented

- Q&A, Nuts and Bolts, Where it could go next
- Code at https://github.com/jaxon6/inheritable-puppet-ldap

What Came Before

- Active Directory, WinAthena (win.mit.edu)
- Computer objects are managed in an LDAP tree
- Policies are applied to the computer objects



Introducing Puppet

- Policies Defined with Puppet Classes

Define: With Puppet's declarative language you design a graph of relationships between resources within reusable modules. These modules define your infrastructure in its desired state.

```
class tomcat6::tomcat6 {

    if defined(Class["java::java"]) != true {
        include java::java
    } else {
        #notify { "java::java for tomcat is already defined": }
    }

    case $lsbdistid {
        'Ubuntu':  {
```

- More Defined Types = More Classes

```
find /etc/puppet/modules/ -name *\.pp|grep -v init\.pp|wc -l
187
```

Tomcat	HBase	SNMP Agent
Glassfish	ZooKeeper	HAProxy
Apache2	NFS	Solr
Redis	S3FS	Vertica
MySQL	Java	logstash
PostgreSQL	MCollective	PHP
ActiveMQ	mod_jk	Apt/Yum Repo
Hadoop	Nagios	OS Settings
		Many more...

Associate Classes With Servers

- Site.pp/Nodes.pp

```
node 'ldapmaster.domain.com' inherits basenode {
    include s_ldap::master
}

node 'humanresources.domain.com' inherits basenode {
    include c_humanresources
}
```

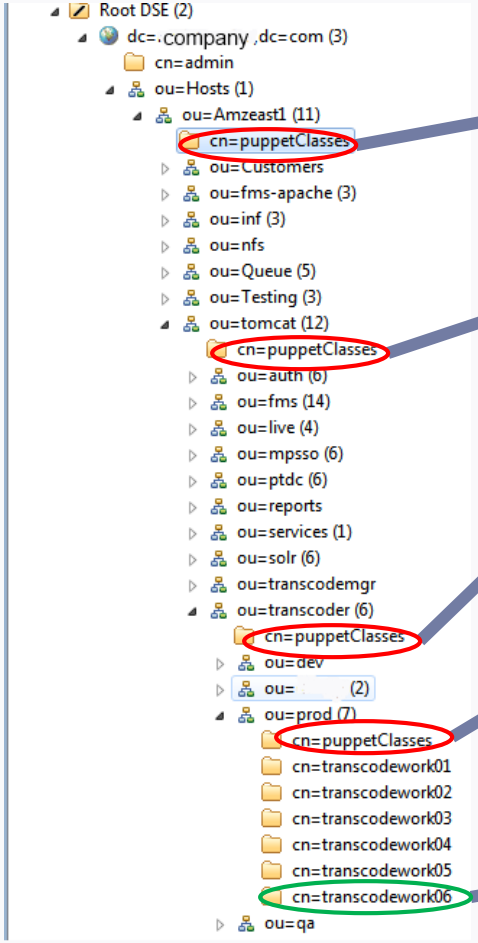
- External Node Classifier
  - Puppet Master calls executable, executable looks up node, returns YAML
    Node data stored in database or other object store
- LDAP
  - http://projects.puppetlabs.com/projects/puppet/wiki/LDAP_Nodes

Puppet And LDAP

- Apache LDAP Browser makes it easy
- Manually applying classes to servers is painful
  - As is implementing logic to automatically apply classes
- Using LDAP, you can make it intrinsic and inherited
  - It is intrinsic, because classes are based on where the object lives
  - It is inherited, because all child objects inherit the classes of the parent objects

LDAP Tree, With Puppet

- puppetHierarchify.pl and cn=puppetClasses



Attribute Description	Value
objectClass	***device (structural)***
objectClass	***puppetClient (auxiliary)***
objectClass	***top (abstract)***
cn	**puppetClasses**
puppetClass	basic::basic
puppetClass	basic::mcollective

Attribute Description	Value
objectClass	***device (structural)***
objectClass	***puppetClient (auxiliary)***
objectClass	***top (abstract)***
cn	**puppetClasses**
puppetClass	tomcat6::tomcat6

Attribute Description	Value
objectClass	***device (structural)***
objectClass	***puppetClient (auxiliary)***
objectClass	***top (abstract)***
cn	**puppetClasses**
puppetClass	transcoder::binaries11

Attribute Description	Value
objectClass	***device (structural)***
objectClass	***puppetClient (auxiliary)***
objectClass	***top (abstract)***
cn	**puppetClasses**
puppetClass	basic::prod_prompt
puppetClass	s3fs::transcoder_prod
puppetClass	tomcat6::transcoder_prod
puppetClass	transcoder::transcoder_prod

info attribute, how puppetHierarchify.pl tracks changes

Attribute Description	Value
objectClass	***device (structural)***
objectClass	***extensibleObject (auxiliary)***
objectClass	***puppetClient (auxiliary)***
objectClass	***top (abstract)***
cn	**transcodework06**
info	basic::basic
info	basic::mcollective
info	basic::prod_prompt
info	s3fs::transcoder_prod
info	tomcat6::tomcat6
info	tomcat6::transcoder_prod
info	transcoder::binaries11
info	transcoder::transcoder_prod
puppetClass	basic::basic
puppetClass	basic::mcollective
puppetClass	basic::prod_prompt
puppetClass	s3fs::transcoder_prod
puppetClass	tomcat6::tomcat6
puppetClass	tomcat6::transcoder_prod
puppetClass	transcoder::binaries11
puppetClass	transcoder::transcoder_prod

Easy To Use

- Graphical tool for easy in/easy out work
- If you move an object, classes automatically add/remove themselves
- To create a new role, create a new OU and specify cn=puppetClasses value
- Add your servers and the classes apply automatically
- Different OUs with different permissions

From First Boot To Fully Installed

- For bare-metal:
  - Add DHCP/DNS entries for host(s)
  - PXE-boot server, install OS and auto-enable Puppet
  - Puppet takes it from there
- For EC2 (using un-modified vendor AMIs):
  - UserData bash script - sets hostname then runurl script
    - script installs puppet, sets domain-name
  - Puppet takes it from there
- Shell script queries LDAP, appending to autosign.conf

puppetHierarchify.pl And Next Steps

```
sub getPuppetClasses {
undef @puppetClassesArray;
$ouArrayCount = @ouArray;
$puppetTemplatesCounter = 0;

for ( $ouArrayLoop = 0; $ouArrayLoop < $ouArrayCount; $ouArrayLoop++ ) {
    $puppetTemplate = $ouArray[$ouArrayLoop];
    $puppetTemplatesMatch =  "(objectClass=puppetClient)";
    $puppetTemplatesFilter = Net::LDAP::Filter->new($puppetTemplatesMatch) or die "Bad bindFilter $puppetTemplatesMatch";
    $puppetTemplatesMesg = $ldap->search ( #do the search
                           base   => $puppetTemplate,
                           filter => $puppetTemplatesFilter,
                           scope  => "base",
                           attrs  => ['puppetClass'],
    ) or die $@;


    #if error, spit out error - don't die, as some hosts are in OUs without puppetClasses objects
    #$puppetTemplatesMesg->code && print $puppetTemplatesMesg->error;

    #print $puppetTemplate;
    #print "\n";
    #we only search for one object, so no looping, just an if for not 0
    if ( $puppetTemplatesMesg->count != 0) {
        #we only return one host, so just search for entry 0
        $puppetTemplatesEntry = $puppetTemplatesMesg->entry ( 0 );
        if ($puppetTemplatesEntry->exists( 'puppetClass' )) {
            $puppetTemplatesAttrValue = join( "\n", $puppetTemplatesEntry->get_value( 'puppetClass' ) ), ;
            @ptmplArray = (split(/\n/, $puppetTemplatesAttrValue));
            $ptmplac = @ptmplArray;
            #print $ptmplac;
            #print "\n";
            for ( $ptmplLoop = 0; $ptmplLoop < $ptmplac; $ptmplLoop++ ) {
                $puppetClassesArray[$puppetTemplatesCounter] = $ptmplArray[$ptmplLoop];
                #print $puppetClassesArray[$puppetTemplatesCounter];
                #print "\n";
                $puppetTemplatesCounter++;
            }
        }
    }
```

- Make puppetVars inherited
  - puppetVar nagioscheck (e.g., nagioscheck=tomcat6)
  - All nagioscheck puppetVars automatically apply to server
  - Script on Nagios server queries LDAP and automatically applies those checks as defined by nagioscheck
- Other AD features, like inheritance blocking

Thanks / Q&A

- Mediaplatform.com
  - Puppet DevOps person
  - QA Lead

- Puppet Labs

- SCaLE
- Questions and Answers