Deploying to environments with Puppet

Use Puppet master/agent

I want to deploy a simple application that installs a file in /tmp on a single box.

Here is the puppet module definition:

/etc/puppetlabs/puppet/modules/myapp/manifests/init.pp:

*class myapp {*

*file { 'testfile' : path => '/tmp/testfile-local', ensure => 'present', content => 'Test', mode => 0640}*

*}*

Typically, the puppet master uses a site.pp file for node definitions:

/etc/puppet/manifests/site.pp:

*node development {*

*include "myapp"*

*}*

*node staging {*

*include "myapp"*

*}*

Start the puppet master:

*puppet master --no-daemonize --verbose*

*notice: Starting puppet master version 2.6.4*

This is telling puppet master that myapp needs to be installed on the development box. If I have a puppet agent running on the client (which in this case happens to be the same box), then it will apply the latest configuration from the server automatically when it polls the next time.

I could run the agent manually onetime by ssh'ing into the box and invoking the agent:

*[root@learn ~]# puppet agent --no-daemonize --onetime --server puppet --verbose*

*info: Retrieving plugin*

*info: Caching catalog for puppet*

*info: Applying configuration version '1310559260'*

*notice: /Stage[main]/Myapp/File[testfile-local]/ensure: created*

*notice: Finished catalog run in 0.02 seconds*

CONS:

The main problem here is that I am not sure how you to achieve this via puppet master itself .. have not checked the UI - but surely there must be a way to update specific nodes - i.e. dev nodes only ?

You can always automate this using a capistrano script:

*set :user, "root"*

*task :deploy*

*role :app, ‘development’*

*run 'puppet agent --no-daemonize --onetime --server puppet --verbose'*

*end*

The problem with using a script like this is that the environment information has to be maintained in two places – in the script and the site.pp file. One way to avoid this is to generate this script from the site.pp file.

PROS:

You are using Puppet to install the application, just like a sysadmin would to ensure that machines were setup correctly. Easy to sell to ops.

Server-less puppet

Use capistrano to run puppet “apply” on the relevant nodes. Here is a Capfile for the app. It assumes that you have manifests checked out on the clients.

*set :user, "root"*

*task :development do*

*role :app, “development”*

*end*

*task :staging do*

*role :app, "staging"*

*end*

*task :deploy do*

*# TODO: checkout manifests to module path*

*run "puppet apply -e \"include myapp\""*

*end*

To deploy to development environment, you would run cap for that environment:

*rg6977:puppet Thoughtworks$ cap development deploy*

*\* executing `development'*

*\* executing `deploy'*

*\* executing "puppet apply -e \"include myapp\""*

*servers: ["192.168.56.101"]*

*Password:*

*[192.168.56.101] executing command*

*command finished*

PROS:

Node definitions are now in Capistrano and not in puppet. Puppet is used only to install the application in a given environment. Puppet does a poor job of managing environment specific information. See http://docs.puppetlabs.com/guides/environment.html.

CONS:

Node definitions are now in Capistrano and not in puppet. Puppet is used only to install the application in a given environment.

This is more code than the previous approach. Puppet “apply” will only apply manifests to the local machine. So, if your application must be installed on a web server and db server, your Capistrano script needs to invoke puppet apply on the appropriate manifest (db vs web).