Dev Buildout Artifacts

Salt Master

Salt maintains a downloadable install script which should work in just about every distribution of Linus. When I ran salt master install script, it didn't actually install the salt master. In the future `aptitude install salt-master` (etc...) works just fine.

Installing and starting the Salt Minion (which is described in its own section) will send keys to the Master which require acceptance before the minion can be manipulated by it. The command `salt-key -L` will list all the salt keys, including an unaccepted section with keys from minions asking to be accepted. The command `salt-key -a Salt_Key" will accept an individual key called "Salt_Key" that shows up in the previous list, and `salt-key -A` will accept all keys.

Keys are based on the `id` established in the '/etc/salt/minion' configuration file. If you change the id you will need to accept the key again for the new id.

The salt state files are in /srv/salt, and in that directory you'll find we recently we subdivided that space by environments, dev, stage, prod, and poc. The first three are self-explanitory, the 'poc' is a production environment for the testing product.
Using grains to set up nodegroups for resin and ldap.

The Salt Master requires an EIP, but amazon doesn't automatically associate it after bootup. To solve this problem we put a script in /etc/init.d/elastic-ip that associates the EIP on bootup. It uses an IAM user with specific and limited access to associate EIP's only. But that required the latest version of the ec2-api-tools, which I eventually just downloaded and installed in /opt (the ones available in apt were all to old).

To facilitate the use of salt in the ec2 environment, we deployed a salt ec2 grain plugin at _grains in the salt directories, which can be found here https://github.com/saltstack/salt-contrib/blob/master/grains/ec2_info.py

Salt Minion

Installing Salt Minion on Redhat Servers...

```
rpm -Uvh http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpmyum install salt-minion
```

Once installed, and before you start it, be sure to edit the `/etc/salt/minion` file for the following details...

id: Salt Keys are based on this value. And by default this value is based on the hostname of this machine. AWS sets a hostname that is based on the IP, and that can change each time you reboot. So it is best to set the ID yourself.

For the LDAP servers, I've been using a convention of "ldapdev1" which specifies the role, then the environment, and then a sequence to make sure the ID is unique among servers.

For the Jetty servers, we've been using a convention of "jetty-090983204" where the series of numbers actually represents the time the machine was started up as seconds since the UNIX Epoch as found by `date+"%s"`. Note that at some point, we anticipate establishing the `id` will be automated through a utility called `salt-cloud` with the base pattern 'role'-'environment'_'Epoch_Seconds'.

You will also want to establish the role for the machine, which will tell the Salt Master what should be installed and enabled on that machine. The 'roles' list is found by searching the /etc/salt/minion file.

LDAP

Installing 389 on Centos or Redhat is actually very easy once Salt Minion is installed, because they both require the EPEL yum repository to be added. From there the salt file that installs 389-DS is very easy and accomplished with the package module which already knows how to install and uninstall rpm's from yum...

The 389-ds/init.sls file simply creates an ldap user, and installs 389-ds.

Commit Directory

It is when you want to install a directory into 389-ds where installation gets more involved. Fortunately, we can do much of it from the command line.

Essentially there are three parts to this, the setup perl script which is included in the 389-ds packages installed on the client, the setup.inf file which it uses to configure the new directory, and multi-master.ldif which initiates a dual multi-master connection between two nodes (one assumed to be ldap-dev1 and ldap-dev2).

Mullti-master.ldif uses Jinja templates to make slight alterations for whether it is going to ldap-dev1 or ldap-dev2.

In this way we can give our servers a sense of architecture, or where they fit in a much greater whole. The challenge, which is a challenge with any configuration management program, is how to describe that architecture in a dynamic way. How does the installation change if we add another ldap server so that they are all mutli-master? We can bypass this conundrum if we simplify the multi-master role to simply be two servers, and make any extension to them be copy-only. But only slightly, as the replication agreement is still requires actions on both members of that replication scheme.

Also perplexing at the moment is how to handle SSL files. We are using a script which will generate new self-signed certs for each installation. This would certainly be required for any additional ldap server iterated into the cluster. However those certs will need to be added all over the cluster, and to the java servers authenticating to ldap, once created. how to do that is an issue that we don't have a good answer to at the moment. One solution would be to create our own certificate authority, which would be the only cert required to add to every server.

Jetty

Like LDAP, Jetty comprises two installations, the server itself and then the Shibboleth IDP installation.
The Jetty installation is a bit involved because it does not come in a handy rpm from yum. Instead the tar file is downloaded, and untarred, and then Jetty is configured in the salt process. We had to write our own init.d file, which is also installed during that process, which better handles the forking process for starting Jetty.

We also configured Jetty to listen on a port to know when to be shut down, rather than using the typical init.d technique of saving and looking for a pid. To test if Jetty is up, it does a quick connection to the expected open Jetty port.

Right now we don't have SSL installed, or a way to add the certs to the keystore. For more information on the complexity of ssl certs in general, please review the LDAP section above.

IDP

We simplified this installation by creating a repository in Salt to handle all of the IDP installation. Salt then manages the IDP directories on all of the Jetty Servers, meaning that any change to the local copy on Salt will cause the minion to have the same change when the state is propigated.

Another interesting note is that IDP is configuring its portion of the Jetty xml file. This means that for each state enforcement, Jetty re-sets its configuration files to default, and then IDP re-writes its changes each time. There are many ways to solve this, however right now we are working on differentiating an installation and an update or enforcement state change through appropriate "unless" messages.

Current Machines...

The most important machine to know is, 'ec2-54-244-127-183.us-west-2.compute.amazonaws.com' which is the Salt Master [EIP]. All of the other systems are dynamic, and change more than can be updated in this document. However, the following command will give a list of all the available servers and the DNS names they can be reached at...

```
salt * grains.get ec2_public-hostname
```

This relies on the ec2 grains module described in the Salt Master section of this document.

ELB Entries:

CommITIdPELB-1232715940.us-west-2.elb.amazonaws.com (A Record)
ipv6.CommITIdPELB-1232715940.us-west-2.elb.amazonaws.com (AAAA Record)
dualstack.CommITIdPELB-1232715940.us-west-2.elb.amazonaws.com (A or AAAA Record)