Wednesday, May 28. 2014

Solaris 11.2: Time based access limitations

Let's assume you want to limit ssh login for user junior to a certain timespan, let's say weekdays between 13:10 and 17:00. With Solaris 11.2 it's really easy to limit access to certain services based on times.

To enforce this, you can set access_time for certain PAM services for the use junior like this. The limitation is done by the pam_unix_account.so module and the man page states : Validate that the user is permitted to access the PAM service at the current time and day of the week.You will see pam_unix_account doing its job later on.

I will simply limit all PAM services that are used by ssh. Log into your server as root:

usermod -K access_times='{sshd-none,sshd-password,sshd-kbdint,sshd-pubkey,sshd-hostbased}:Wk1310-1700' junior

Let's try to log in before 13:10.

desktop:~ joergmoellenkamp$ date
Mi 28 Mai 2014 13:08:55 CEST
desktop:~ joergmoellenkamp$ ssh junior@192.168.1.16
Hi, i am default
Password:
Warning: 1 failed authentication attempt at Wed May 28 13:07 2014 since last successful authentication.

pam_unix_account: User junior may not use the sshd-kbdint service at this time.

Password:
pam_unix_account: User junior may not use the sshd-kbdint service at this time.

Password:
pam_unix_account: User junior may not use the sshd-kbdint service at this time.

Permission denied (gssapi-keyex,gssapi-with-mic,publickey,keyboard-interactive).
desktop:~ joergmoellenkamp$

Access is denied. Now get some coffee, talk with your colleagues and get back to the shell after 13:10, let's say at 13:12.

desktop:~ joergmoellenkamp$ date
Mi 28 Mai 2014 13:12:07 CEST
desktop:~ joergmoellenkamp$ ssh junior@192.168.1.16
Hi, i am default
Password:
Last login: Wed May 28 11:49:04 2014 from desktop
Oracle Corporation SunOS 5.11 11.2 April 2014
junior@master:~$

Voila, now you can log into your services.

Posted by Joerg Moellenkamp in English, Solaris at 14:58

Use ntp, not rdate

Just as i saw it in a blog on blogs.oracle.com (not link here) and my comment isn't published there: I wouldn't use rdate for syncing time between servers. rdate is largely considered as totally obsolete (installing pkg:/network/legacy-remote-utilities should tell you something just by the red part of the name of the package) and using

ntp is really easy and has a much better mechanism to get time synced . I wrote a tutorial about that quite a while ago: "Configuring an NTP client in Solaris 11"

Posted by Joerg Moellenkamp in English, Solaris at 10:18

Tuesday, May 27. 2014

Thank you!

Before i forget someone: A very big "THANK YOU!" for all the nice wishes and congratulation for my birthday yesterday!

Posted by Joerg Moellenkamp in English at 10:29

Monday, May 26. 2014

New Solaris 11.2 features: SMF stencils

As much as there is often a lot discussion about configuration items inside the SMF repository (like the hostname), it brings an important advantage: It introduces the concept of dependencies to configuration changes. What services have be restarted when i change a configuration item. Do you remember all the services that are dependent on the hostname and need a restart after changing it? SMF solves this by putting the information about dependencies into it configuration. You define it with the manifests.

However, as much configuration you may put into SMF, most applications still insists to get it's configuration inside the traditional configuration files, like the resolv.conf for the resolver or the puppet.conf for Puppet. So you need a way to take the information in the SMF repository and generate a traditional config file with it. In the past the way to do so, was some scripting inside the start method that generated the config file before the service started.

Solaris 11.2 offers a new feature in this area. It introduces a generic method to enable you to create config files from SMF properties. It's called SMF stencils. Making a service stencil-awareIn the following blog entry i want to give you a first insight how to use this feature with a well known type of config files. Let's assume you have a service like /network/http:apache22 and you want to create a config file for it with the SMF stencil facility. For example you want to put the virtual host configuration into the SMF repository.

At first you have to do some configuration tasks in the SMF to tell Solaris that this service has a stencil. Create a property group with the type configfile. When it has such property group, Solaris SMF knows it has stencils. You don't have to activate it explicitly. You can have multiple stencils by having multiple property groups with the type configfile.

```
root@master:~# svccfg -s /network/http:apache22
svc:/network/http:apache22> addpg virtualhosts_stencil configfile
svc:/network/http:apache22> setprop virtualhosts_stencil/path = astring: "/etc/apache2/2.2/conf.d/vhost_smf.conf"
svc:/network/http:apache22> setprop virtualhosts_stencil/stencil = astring: "vhost_smf.conf"
svc:/network/http:apache22> setprop virtualhosts_stencil/mode = astring: "0444"
svc:/network/http:apache22> setprop virtualhosts_stencil/user = astring: "root"
svc:/network/http:apache22> setprop virtualhosts_stencil/group = astring: "sys"
```

path is the absolute location of the file you want to generate. stencil is the location of the template relative to /lib/svc/stencils/. So when you set stencil to vhost_smf.conf, the stencil file is /lib/svc/stencils/vhost_smf.conf. mode,user and group should speak for themselves.

Okay, now create the stencil file. This file acts as the template for the configuration file. Let's start with a simple example. A file that does nothing in apache contact, but is explicitly stating, that you should keep your fingers from it

```
root@master:~# cat  /lib/svc/stencils/vhost_smf.conf
> # Automatically generated ... do not edit
> EOT
root@master:~#
```

Okay, now refresh the service and dis- and enable it.

```
root@master:~# svcadm refresh apache22
root@master:~# svcadm disable apache22;svcadm enable apache22
```

Directly afterwards you will see the content of your stencil in the config file location.

```
root@master:~# cat /etc/apache2/2.2/conf.d/vhost_smf.conf
# Automatically generated ... do not edit
```

Inserting property values into the config file

Okay, but now we want to put the value of properties from the SMF repository into our config file for virtualhosts. Okay, For a virtual host config, we "need" the NameVirtualHost directive.

At first i create a property group for this kind of common configuration, let's call it vhost_config. Then i will create a property in it, containing the value for NameVirtualHost

```
root@master:~# svccfg -s svc:/network/http:apache22 addpg vhost_config application
root@master:~# svccfg -s svc:/network/http:apache22 setprop vhost_config/namevirtalhost = astring: "*:80"
```

Now put the following content into the stencil file.

```
root@master:~# cat  /lib/svc/stencils/vhost_smf.conf
# Do not edit
NameVirtualHost $%{vhost_config/namevirtualhost}
# Do not edit
EOT
```

Refresh, disable,enable again.

```
root@master:~# svcadm refresh apache22
root@master:~# svcadm disable apache22;svcadm enable apache22
```

And now you see that you have used the content of the property as a value for your config file.

```
root@master:~# cat /etc/apache2/2.2/conf.d/vhost_smf.conf
# Do not edit

NameVirtualHost *:80

# Do not edit
```

That was simple.

Repeating structures in the config fileWhen you think about your apache configuration, you will surely remember that you have repeating structures in your configfile for each virtual host. How do i create such structures with stencils. SMF Stencils has a construct for it. Let's use this stencil file.

```
root@master:~# cat  /lib/svc/stencils/vhost_smf.conf
# Do not edit
NameVirtualHost $%{vhost_config/namevirtualhost}

$%/vhosts_([0-9]*)/ {

ServerName $%{vhosts_$%1/servername}
ServerAlias $%{vhosts_$%1/serveralias}
DocumentRoot $%{vhosts_$%1/documentroot}

}
# Do not edit
EOT
```

The translation is quite easy. Simply said, for each property group starting that matches on vhosts_, repeat the part between the following curly brakes and use the respective properties servername, serveralias and documentroot and insert them in this block.

When you configure a property group vhost_1 ...

```
root@master:~# svccfg -s apache22 addpg vhosts_1 application
root@master:~# svccfg -s apache22 setprop vhosts_1/serveralias = astring: 'c0t0d0s0.org'
root@master:~# svccfg -s apache22 setprop vhosts_1/servername = astring: 'www.c0t0d0s0.org'
root@master:~# svccfg -s apache22 setprop vhosts_1/documentroot = astring: '/var/www/c0t0d0s0.org'
```

```
root@master:~# svcadm refresh apache22
root@master:~# svcadm disable apache22; svcadm enable apache22
```

... it will yield a configuration file with one Virtualhost-block.

```
root@master:~# svcadm refresh apache22
root@master:~# svcadm disable apache22; svcadm enable apache22
root@master:~# cat /etc/apache2/2.2/conf.d/vhost_smf.conf
# Do not edit
NameVirtualHost *:80



ServerName www.c0t0d0s0.org
ServerAlias c0t0d0s0.org
DocumentRoot /var/www/c0t0d0s0.org


# Do not edit
root@master:~#
```

When you create a second property group ...

```
root@master:~# svccfg -s apache22 addpg vhosts_2 application
root@master:~# svccfg -s apache22 setprop vhosts_2/serveralias = astring: 'moellenkamp.org'
root@master:~# svccfg -s apache22 setprop vhosts_2/servername = astring: 'www.moellenkamp.org'
root@master:~# svccfg -s apache22 setprop vhosts_2/documentroot = astring: '/var/www/moellenkamp.org'
root@master:~# svcadm refresh apache22
root@master:~# svcadm disable apache22; svcadm enable apache22
```

... a second Virtualhost block will appear.

```
root@master:~# cat /etc/apache2/2.2/conf.d/vhost_smf.conf
# Do not edit
NameVirtualHost *:80



ServerName www.c0t0d0s0.org
ServerAlias c0t0d0s0.org
DocumentRoot /var/www/c0t0d0s0.org



ServerName www.moellenkamp.org
ServerAlias moellenkamp.org
DocumentRoot /var/www/moellenkamp.org
```

ConclusionThis is only a very basic example. However it should give you a first insight how you can create configfiles with the SMF stencil facility from the properties in the SMF repository.

Do you want to learn moreOn a Solaris 11.2 system you will find more information about stencils with man smf_stencils and man svcio. I will link to the man pages as soon as they are available at docs.oracle.com

Posted by Joerg Moellenkamp in English, Solaris at 00:01

Sunday, May 25. 2014

RFC7258

Pervasive monitoring is a technical attack that should be mitigated in the design of IETF protocols, where possible.
'nuff said.

Posted by Joerg Moellenkamp in English, Privacy at 13:05

Changes to Openssl in Solaris 11.2

This blog entry wasn't on my radar somehow, nevertheless it reports about an important change to OpenSSL on Solaris 11.2. The most important change from my point of view is the inlining of T4/T5 crypto to Solaris 11.2 openssl:
Years and years ago, I worked on the SPARC T2/T3 crypto drivers. On the SPARC T2/T3 processors, the crypto instructions are privileged; and therefore, the drivers are needed to access those instructions. Thus, to make use of T2/T3 crypto hardware, OpenSSL had to use pkcs11 engine which adds lots of cycles going through the thick PKCS#11 session/object management layer, Solaris kernel layer, hypervisor layer to the hardware, and all the way back.
However, on SPARC T4/T4+ processors, crypto instructions are no longer privileged; and therefore, you can access them directly without drivers. [...]
What does that means to you? Much improved performance! No more PKCS#11 layer, no more copy-in/copy-out of the data from the userland to the kernel space, no more scheduling, no more hypervisor, NADA! [...]

Posted by Joerg Moellenkamp in English, Solaris at 12:44

Friday, May 23. 2014

Darren Moffat about Kernel Zone security

Darren Moffat wrote an interesting blog entry about the security concept of Kernel zones. In "Overview of Solaris Zones Security Models". He is especially talking about a very small, but very very interesting detail:Note that what follows is an out line of implementation details that are subject to change at any time: The kernel of a Solaris Kernel Zone is represented as a user land process in a Solaris non global zone. That non global zone is configured with less privilege than a normal non global zone would have and it is always configured as an immutable zone. So if there happened to be an exploit of the guest kernel that resulted in a VM break out you would end up in an immutable non global zone with lowered privilege.

Posted by Joerg Moellenkamp in English, Solaris at 19:41

A glimpse into Solaris 11.2 specific Puppet components

Now you have a working Puppet testbed in your Solaris 11.2 beta installation it's time to try some Solaris specific stuff. Oracle a number of additional stuff in order to control Solaris specifics like boot environments, VNICs or SMF. You can find the respective code at java.net.
The examples from Manuels blog entry
Let's try at first the puppet commands Manuel Zach wrote about in his blog a few days ago in his blog

I added some lines to my nodes.pp from the last article. I want to install tmux and introduce an zfs quota on the pool/export dataset

```
root@master:/etc/puppet/manifests# cat nodes.pp
import 'etchosts'

node 'default' {
 include etchosts
}

node 'agent3.puppet.c0t0d0s0' {
 include etchosts
 zfs { 'rpool/export':
  quota  => '1G',
 }
 package { 'tmux':
  ensure  => 'present',
 }
}
```

When you try this on agent3, the puppet agent will set the quota and install the package.

```
root@agent3:~# puppet agent --test
Info: Retrieving plugin
Info: Caching catalog for agent3.puppet.c0t0d0s0.org
Info: Applying configuration version '1400838109'
Notice: /Stage[main]/Main/Node[agent3.puppet.c0t0d0s0]/Zfs[rpool/export]/quota: quota changed 'none' to '1G'
Notice: /Stage[main]/Main/Node[agent3.puppet.c0t0d0s0]/Package[tmux]/ensure: created
Notice: Finished catalog run in 21.76 seconds
root@agent3:~#
```

Try the same on agent1 and nothing will happen. Obviously, we've limited it by configuration to agent3. Would be a problem if the system would behave otherwise

```
root@agent1:~# puppet agent --test
```

```
Info: Retrieving plugin
Info: Caching catalog for agent1.puppet.c0t0d0s0.org
Info: Applying configuration version '1400838109'
Notice: Finished catalog run in 0.33 seconds
root@agent1:~#
```

Let's add the creation of a boot environment and put it in the default node:

```
import 'etchosts'

node 'default' {
 include etchosts
  boot_environment { 'solaris-beforepuppet':
  description => 'creating a backup of the bootenvironment',
  ensure => 'present',
}
}

node 'agent3.puppet.c0t0d0s0' {
 include etchosts
 zfs { 'rpool/export':
  quota  => '1G',
}

package { 'tmux':
  ensure  => 'present',
}

}
```

However, when executing it on agent3 nothing will happen.

```
root@agent3:~# puppet agent --test
Info: Retrieving plugin
Info: Caching catalog for agent3.puppet.c0t0d0s0.org
Info: Applying configuration version '1400845195'
Notice: Finished catalog run in 2.49 seconds
```

This is a nice example what is meant with my comment that the default node doesn't define whats done on each node by default, but what is done on a node that doesn't have it's own definition of things to do.

In order to execute it on each node, remove it from the default node. I've put it just in the site.pp like:

```
root@master:/etc/puppet/manifests# cat site.pp
boot_environment { 'solaris-beforepuppet':
  description => 'creating a backup of the bootenvironment',
  ensure => 'present',
}

import 'nodes.pp'
```

When you now trigger it manually on agent3, you see that now the boot environment is created by Puppet.

```
root@agent3:~# puppet agent --test
Info: Retrieving plugin
Info: Caching catalog for agent3.puppet.c0t0d0s0.org
Info: Applying configuration version '1400845435'
Notice: /Stage[main]/Main/Boot_environment[solaris-beforepuppet]/ensure: created
Notice: Finished catalog run in 4.08 seconds
```

Let's repeat it on agent1

```
root@slave1:~# puppet agent --test
Info: Retrieving plugin
Info: Caching catalog for agent1.puppet.c0t0d0s0.org
Info: Applying configuration version '1400846294'
Notice: /Stage[main]/Main/Boot_environment[solaris-beforepuppet]/ensure: created
Notice: Finished catalog run in 1.92 seconds
```

Changing DNS
Another example is changing the DNS server by puppet, for example when you want to migrate 300 zones from one DNS server to another. Simply add the necessary statements to the site.pp

```
root@master:/etc/puppet/manifests# cat site.pp
boot_environment { 'solaris-beforepuppet':
  description => 'creating a backup of the bootenvironment',
  ensure => 'present',
}

dns { 'Google Nameserver':
 nameserver => '8.8.8.8',
 search => 'puppet.c0t0d0s0.org'
}
import 'nodes.pp'
```

Now trigger puppet on on of the servers - for example agent3 and check the correct DNS configuration:

```
root@sagent3:~# puppet agent --test
Info: Retrieving plugin
Info: Caching catalog for agent3.puppet.c0t0d0s0.org
Info: Applying configuration version '1400846900'
Notice: /Stage[main]/Main/Dns[Google Nameserver]/nameserver: nameserver changed '8.8.8.8' to '8.8.8.8'
Notice: /Stage[main]/Main/Dns[Google Nameserver]/search: search changed '' to 'puppet.c0t0d0s0.org'
Notice: Finished catalog run in 3.76 seconds
root@agent3:~# nslookup www.c0t0d0s0.org
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   www.c0t0d0s0.org
Address: 178.63.69.146
```

ConclusionAs indicated by the link to java.net there are a lot more available providers, but that is stuff for more blog entries

Posted by Joerg Moellenkamp in English, Solaris at 14:01

Thursday, May 22. 2014

Event Announcement: Oracle Business Breakfast "Solaris 11.2" in Hamburg am 12.6.2014

Am 12.6. findet auch in Hamburg ein Oracle Business Breakfast zum Thema Solaris 11.2 statt. Ich werde dort nicht vortragen, diesmal ist mein Kollege Stefan Hinker dran. Wenn Ihr die Möglichkeit habt nach Berlin zu fahren als Hamburger, würde ich vorschlagen, das ihr euch für Berlin anmeldet (Stefan, no insult intended, doch der VP Solaris Core OS als Vortragender ist ... sagen wir es mal so ... eine seltenere Gelegenheit dem Verantwortlichen die Meinung zu sagen). Der Vortrag findet im Oracle CVC in der Innenstadt von Berlin statt und ist ganz gut vom Hauptbahnhof erreichbar. Habt ihr keine Gelegenheit dazu, ist Hamburg eine gute Alternative, die Agenda ist identisch in für beide Events. Die Anmeldung für Hamburg ist hier verfügbar.

Posted by Joerg Moellenkamp in German, Solaris at 10:09

Wednesday, May 21. 2014

Basic Puppet installation with Solaris 11.2 beta

At the recent announcement we talked a lot about the Puppet integration. But how do you set it up? I want to show this in this blog entry.

However this example i'm using is even useful in practice. Due to the extremely low overhead of zones i'm frequently seeing really large numbers of zones on a single system. Changing /etc/hosts or changing an SMF service property on 3 systems is not that hard. Doing it on a system with 500 zones is ... let say it diplomatic ... a job you give to someone you want to punish.

Puppet can help in this case making of managing the configuration and to ease the distribution. You describe the changes you want to make in a file or set of file called manifest in the Puppet world and then roll them out to your servers, no matter if they are virtual or physical. A warning at first: Puppet is a really,really vast topic. This article is really basic and it doesn't goes more than just even toe's deep into the possibilities and capabilities of Puppet. It doesn't try to explain Puppet ... just how you get it up and running and do basic tests. There are many good books on Puppet. Please read one of them, and the concepts and the example will get much clearer immediately.

To show this, i have a relatively simple setup. A global zone and three non-global zones. In this example i will set up the global zone as a Puppet master and will configure three non-global zone with the Puppet agent, the component that gathers information from the Puppet master in order to do something on the server running the agent.

The Puppet master is called master, the agents are called agent1, agent2 and agent3. This tutorial assumes that you have already set up the zones and that they have working network connection.
Preparation
Okay, at first it's really important that you have a working resolution of the hostnames. The resolution is key to get Puppet working. This is what i did in the global zone and in the non-global zones.

The /etc/hosts of all zones (either global or non-global) is now like this.

```
root@agent1:~# cat /etc/hosts
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Internet host table
#
::1             localhost
127.0.0.1       localhost loghost
192.168.1.16    master.puppet.c0t0d0s0.org master kusanagi
192.168.1.230   agent1.puppet.c0t0d0s0.org agent1
192.168.1.231   agent2.puppet.c0t0d0s0.org agent2
192.168.1.232   agent3.puppet.c0t0d0s0.org agent3
```

Further i've set the hostname on all servers.

```
root@master:~# hostname master.puppet.c0t0d0s0.org
root@agent1:~# hostname agent1.puppet.c0t0d0s0.org
root@agent2:~# hostname agent2.puppet.c0t0d0s0.org
root@agent3:~# hostname agent3.puppet.c0t0d0s0.org
```

Installing and configuring Puppet
Okay, now install Puppet in all zones. This is quite easy as since 11.2 beta , puppet is in the standard repository . You just have to execute pkg install puppet

Ensure that the package has been installed on all zones you want to work with.

When you have installed the puppet package on all zones, login as root on the master At first bring up the master.

```
root@master:/etc/puppet/ssl# svccfg -s puppet:master setprop config/server = master.puppet.c0t0d0s0.org
root@master:/etc/puppet/ssl#  svccfg -s puppet:master refresh
root@master:/etc/puppet/ssl# svcadm enable puppet:master
```

Afterwards you should see the puppet master service running on your system.

```
root@master:/etc/puppet/manifests# svcs puppet
STATE          STIME    FMRI
disabled       Mai_19   svc:/application/puppet:agent
online         Mai_19   svc:/application/puppet:master
```

Now we have prepare the agent systems. At first log into agent1 system

```
root@agent1:# puppet agent --test --server master.puppet.c0t0d0s0.org
Info: Creating a new SSL key for agent1.puppet.c0t0d0s0.org
Info: Caching certificate for ca
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for agent1.puppet.c0t0d0s0.org
Info: Certificate Request fingerprint (SHA256):
14:20:1E:C8:D8:78:1D:DF:9C:92:75:F2:72:C6:61:61:AC:56:82:06:FC:A4:6D:5E:DA:5F:7E:12:80:5B:90:A9
Info: Caching certificate for ca
Exiting; no certificate found and waitforcert is disabled
root@agent1:~#
```

The connection between the agent and the master is protected by SSL. At the first execution the agent creates a key and and certificate request.

You have to repeat the steps in the zones agent2 and on agent3 as well. At first on agent2

```
root@agent2:~# puppet agent --test --server master.puppet.c0t0d0s0.org
Info: Creating a new SSL key for agent2.puppet.c0t0d0s0.org
Info: Caching certificate for ca
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for agent2.puppet.c0t0d0s0.org
Info: Certificate Request fingerprint (SHA256):
76:A7:23:A7:4D:41:66:DD:71:B0:4E:AA:62:EC:0B:DB:61:59:BE:56:43:15:E7:BA:C3:CB:AF:D3:98:D3:30:18
Info: Caching certificate for ca
Exiting; no certificate found and waitforcert is disabled
root@agent2:~#
```

Now on agent3:

```
root@agent3:~# puppet agent --test --server master.puppet.c0t0d0s0.org
Info: Creating a new SSL key for agent3.puppet.c0t0d0s0.org
Info: Caching certificate for ca
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for agent3.puppet.c0t0d0s0.org
Info: Certificate Request fingerprint (SHA256):
C4:96:A1:B1:D5:F3:EB:AA:2C:B8:9B:9E:83:C5:F8:64:2E:F7:D8:50:49:AA:0A:E8:73:BE:E0:53:D8:12:95:F7
Info: Caching certificate for ca
Exiting; no certificate found and waitforcert is disabled
root@agent3:~#
```

Now we have to sign this certificates on the master. Please go back to your shell of master

```
root@master:/etc/puppet/ssl# puppet cert list
  "agent1.puppet.c0t0d0s0.org" (SHA256)
14:20:1E:C8:D8:78:1D:DF:9C:92:75:F2:72:C6:61:61:AC:56:82:06:FC:A4:6D:5E:DA:5F:7E:12:80:5B:90:A9
  "agent2.puppet.c0t0d0s0.org" (SHA256)
76:A7:23:A7:4D:41:66:DD:71:B0:4E:AA:62:EC:0B:DB:61:59:BE:56:43:15:E7:BA:C3:CB:AF:D3:98:D3:30:18
  "agent3.puppet.c0t0d0s0.org" (SHA256)
C4:96:A1:B1:D5:F3:EB:AA:2C:B8:9B:9E:83:C5:F8:64:2E:F7:D8:50:49:AA:0A:E8:73:BE:E0:53:D8:12:95:F7
```

So we have three requests to sign. Let's sign them.

```
root@master:/etc/puppet/ssl# puppet cert sign agent1.puppet.c0t0d0s0.org
Notice: Signed certificate request for agent1.puppet.c0t0d0s0.org
Notice: Removing file Puppet::SSL::CertificateRequest agent1.puppet.c0t0d0s0.org at '/etc/puppet/ssl/ca/requests/
root@master:/etc/puppet/ssl# puppet cert sign agent2.puppet.c0t0d0s0.org
Notice: Signed certificate request for agent2.puppet.c0t0d0s0.org
Notice: Removing file Puppet::SSL::CertificateRequest agent2.puppet.c0t0d0s0.org at
'/etc/puppet/ssl/ca/requests/agent2.puppet.c0t0d0s0.org.pem'
root@master:/etc/puppet/ssl# puppet cert sign agent3.puppet.c0t0d0s0.org
Notice: Signed certificate request for agent3.puppet.c0t0d0s0.org
Notice: Removing file Puppet::SSL::CertificateRequest agent3.puppet.c0t0d0s0.org at
'/etc/puppet/ssl/ca/requests/agent3.puppet.c0t0d0s0.org.pem'
```

Okay, let's check if there is something left:

```
root@master:/etc/puppet/ssl# puppet cert list
```

Now you can check on each agent system the connectivity. You will see quite a long output

```
root@agent3:~# puppet agent --test --server=master.puppet.c0t0d0s0.org
Info: Caching certificate for agent3.puppet.c0t0d0s0.org
Info: Caching certificate_revocation_list for ca
Info: Caching certificate for agent3.puppet.c0t0d0s0.org
Info: Retrieving plugin
Notice: /File[/var/lib/puppet/lib/puppet]/ensure: created
[...]
Info: Caching catalog for agent3.puppet.c0t0d0s0.org
Info: Applying configuration version '1400520716'
Info: Creating state file /var/lib/puppet/state/state.yaml
Notice: Finished catalog run in 0.09 seconds
root@agent3:~# svccfg -s puppet:agent setprop config/server = master.puppet.c0t0d0s0.org
root@agent3:~#  svccfg -s puppet:agent refresh
root@agent3:~# svcadm enable puppet:agent
```

Do the same in the other zones. At first agent2

```
root@agent2:~# puppet agent --test --server master.puppet.c0t0d0s0.org
Info: Caching certificate for agent2.puppet.c0t0d0s0.org
Info: Caching certificate_revocation_list for ca
Info: Caching certificate for agent2.puppet.c0t0d0s0.org
Info: Retrieving plugin
Notice: /File[/var/lib/puppet/lib/puppet]/ensure: created
[...]
Info: Caching catalog for agent2.puppet.c0t0d0s0.org
Info: Applying configuration version '1400520716'
Info: Creating state file /var/lib/puppet/state/state.yaml
Notice: Finished catalog run in 0.09 seconds
root@agent2:~# svccfg -s puppet:agent setprop config/server = master.puppet.c0t0d0s0.org
root@agent2:~# svccfg -s puppet:agent refresh
root@agent2:~# svcadm enable puppet:agent
```

Afterwards on agent1:

```
root@agent1:~# puppet agent --test --server master.puppet.c0t0d0s0.org
Info: Caching certificate for agent1.puppet.c0t0d0s0.org
Info: Caching certificate_revocation_list for ca
Info: Caching certificate for agent1.puppet.c0t0d0s0.org
Info: Retrieving plugin
Notice: /File[/var/lib/puppet/lib/puppet]/ensure: created
[...]
Info: Caching catalog for agent1.puppet.c0t0d0s0.org
Info: Applying configuration version '1400520716'
Info: Creating state file /var/lib/puppet/state/state.yaml
Notice: Finished catalog run in 0.09 seconds
root@agent1:~# svccfg -s puppet:agent setprop config/server = master.puppet.c0t0d0s0.org
root@agent1:~# svccfg -s puppet:agent refresh
root@agent1:~# svcadm enable puppet:agent
```

Do something slightly useful with the installation
Okay, now we are ready to try this our puppet installation. I won't start with the basic help world feature, but something slightly more useful. However we have to some configuration first. I won't explain much, as when i start to explain everything about it, this article will be veeeeery long. Basically the following stuff is creating a Puppet module called etchosts which delivers an /etc/hosts file (thus the name) to the system. The files is at /etc/puppet/modules/etchosts/files, which can be accessed by Puppet under the URL puppet:///modules/etchosts/hosts. The module code is located in a file init.pp (a convention in Puppet) inside the modules manifest directory at /etc/puppet/modules/etchosts/manifests directory.

Afterwards i'm creating nodes.pp file which essentially says "By default - when there is no node definition that matches - execute the function etchosts imported from the module etchosts and so the /etc/puppet/modules/etchosts/files/hosts is moved to /etc/hosts if necessary (something has changed)." This file is invoked by the site.pp file.
At first we need some additional structure in some filesystems

```
root@master:~# mkdir /etc/puppet/modules/etchosts
root@master:~# mkdir /etc/puppet/modules/etchosts/files
root@master:~# mkdir /etc/puppet/modules/etchosts/manifests
root@master:~# cp /etc/hosts /etc/puppet/modules/etchosts/files/hosts
```

Now we will give some life to the module /etc/hosts

```
root@master:~# cat  /etc/puppet/modules/etchosts/manifests/init.pp
> class etchosts {
>         file { "/etc/hosts":
>                 source => 'puppet:///modules/etchosts/hosts',
>         }
> }
> EOT
```

Afterwards we creating a file just including the nodes definition file:

```
root@master:~# cat  /etc/puppet/manifests/sites.pp
> import 'nodes.pp'
> EOT
```

At last we define the behaviour for the "default" node:

```
root@master:~# cat  /etc/puppet/manifests/nodes.pp
import 'etchosts'

node 'default' {
 include etchosts
}
```

In order to have something to play i just add a line to the /etc/puppet/modules/etchosts/files/hosts file.

echo "# Do not edit - modified by puppet" >> /etc/puppet/modules/etchosts/files/hosts

Okay, now log into one of the zones with an agent and check the current /etc/host

```
jmoekamp@agent2:~$ cat /etc/hosts
[...]
192.168.1.232   agent3.puppet.c0t0d0s0.org agent3
jmoekamp@agent2:~$
```

At the moment there there isn't the additional line in the file. You can now wait for 1800 seconds at maximum (because the agent checks every 1800 seconds if there is something to do by default) or you can force the check. Let's force it.

```
root@agent2:~# puppet agent --test
Info: Retrieving plugin
Info: Caching catalog for agent2.puppet.c0t0d0s0.org
Info: Applying configuration version '1400529373'
Notice: /Stage[main]/Etchosts/File[/etc/hosts]/content:
--- /etc/hosts  Tue May 20 04:33:32 2014
+++ /tmp/puppet-file20140520-8490-tfgwja        Tue May 20 19:52:02 2014
@@ -11,2 +11,3 @@
 192.168.1.231  agent2.puppet.c0t0d0s0.org agent2
 192.168.1.232  agent3.puppet.c0t0d0s0.org agent3
+# Do not edit - modified by puppet

Info: /Stage[main]/Etchosts/File[/etc/hosts]: Filebucketed /etc/hosts to puppet with sum
38f6c964aab77edb2ff938094f13e2d0
Notice: /Stage[main]/Etchosts/File[/etc/hosts]/content: content changed '{md5}38f6c964aab77edb2ff938094f13e2d0' to
'{md5}49b07e8c62ed409a01216bf9a35ae7ae'
Notice: Finished catalog run in 0.60 seconds
```

Now let's check the file again ... et voila ... you find the line you have added into the /etc/puppet/modules/etchosts/files/hosts in the /etc/hosts of you file.

```
root@agent2:~# cat /etc/hosts
[...]
192.168.1.230   agent1.puppet.c0t0d0s0.org agent1
192.168.1.231   agent2.puppet.c0t0d0s0.org agent2
192.168.1.232   agent3.puppet.c0t0d0s0.org agent3
# Do not edit - modified by puppet
```

When go for a coffee and wait the mentioned 1800 seconds you will find the change in all your zones.

Conclusion
One of the points of the puppet integration into Solaris 11.2 was the development of a lot of solaris specific modules for managing boot environments or configuring VNICs. This example doesn't touch those topics at all. But now you have a foundation for follow on articles going into details about Solaris specific modules.

Posted by Joerg Moellenkamp in English, Solaris at 21:56

Event Announcement: Oracle Business Breakfast "Solaris 11.2" in Berlin am 13.6.2014 mit Markus Flierl

Am 13.6. findet ein Oracle Business Breakfast im Customer Visit Center in Berlin zum Thema "Solaris 11.2" statt. Es wird allerdings kein lokaler Kollege sprechen, sondern mit Markus Flierl ein Kollege aus dem Engineering, der die Entwicklung als VP Solaris Core Technology verantwortet. Die Agenda und die Möglichkeit zur Anmeldung findet ihr hier. Um zahlreiches Erscheinen wird gebeten!

Posted by Joerg Moellenkamp in German, Solaris at 16:50

Tuesday, May 20. 2014

LDOM disk io improvements in Solaris 11.1 SRU 19

Stefan Hinker wrote an interesting blog entry about recent performance improvements with LDOMs in his article "Improved vDisk Performance for LDoms":In conclusion, with the new, improved virtual networking and virtual disk IO that are now available, the range of applications that can safely be run on fully virtualized IO has been expanded significantly. This is in line with the expectations I often find in customer workshops, where high end performance is naturally expected from SPARC systems under all circumstances.

Posted by Joerg Moellenkamp in English, Solaris, SPARC at 10:06

Monday, May 19. 2014

Solaris 11.2 announcement - a collection of links and hints (Part 5)

Casper is - as usual - a good information about Solaris: In his blog he is writing about some changes in Solaris 11.2: For example he is writing about the removal of unlink/link for directories in "Solaris 11.2: unlink(2)/link(2) for directories: your time is up.". Another interesting article is "Solaris 11: Evolution of v_path." about the changes to the vnode-to-pathname mapping in the light of newer Solaris 11 features.

A hint for people trying out kernel zones: When you just enter the commands some blogs or websites were posting you may run into the problem that the zone won't boot and the boot fails with a message indicating lack of memory. When you want to use kernel zones please limit the ARC cache of ZFS like This can be done with a entry like set zfs:zfs_arc_max=0x40000000(for an 1 Gig ARC) in /etc/system. This is necessary due to the fact that a kernel zone is allocating a lot of memory in a short time and sometimes the ZFS arc can't get out of the way fast enough.

Posted by Joerg Moellenkamp in English, Solaris at 13:48

Wednesday, May 7. 2014

Less known Solaris features: synchronous svcadm

Sometimes small options are really useful. When you enable a service, for example the Apache HTTPD, you enter svcadm enable apache22. This command immediately return. There is no direct feedback at the command return, if the service has started. So you often see people doing something like that:

```
root@kusanagi:~# svcs -a| grep "apache22"
disabled       22:16:52 svc:/network/http:apache22
root@kusanagi:~# svcadm enable apache22
root@kusanagi:~# svcs -a| grep "apache22"
offline*       22:17:20 svc:/network/http:apache22
root@kusanagi:~# svcs -a| grep "apache22"
offline*       22:17:20 svc:/network/http:apache22
root@kusanagi:~# svcs -a| grep "apache22"
offline*       22:17:20 svc:/network/http:apache22
root@kusanagi:~# svcs -a| grep "apache22"
offline*       22:17:20 svc:/network/http:apache22
root@kusanagi:~# svcs -a| grep "apache22"
online         22:17:31 svc:/network/http:apache22
```

ust before you ask ... i included a sleep 10 at the right place in the method script of the http:apache22 service to delay the startup in order to demonstrate the command. The service goes from disabled to offline(the state where a service is enabled but the startup hasn't completed) and is flagged as a service in transition to a different state (like online) by the *.

However you can change this behaviour. When you use the -s the svcadm enable command just returns when the state transition has completed. So for example when the service got from offline to online. The svcadm command with the -s option will return as well when it determined that it can't do the desired state transition and needs admin intervention.

In our example we would use svcadm enable -s apache22. As we artificially delayed the startup of the Apache, the svcadm command should run at least 10 seconds. Let's check this:

```
# ptime svcadm enable -s  apache22

real       11.137908105
user        0.012195633
sys         0.018084807
```

This synchronous mode works with svcadm enable as well as with svcadm disable even in Solaris 10. In Solaris 11.2 the synchronous mode was introduced to all svcadm subcommands including the mechanism to define a timeout after which the svcadm command return with an error code that the state transition hasn't completed with the timeout. However it's just the waiting for the state transition that is timouted, the service itself proceeds with it's bring up.

Posted by Joerg Moellenkamp in English, Solaris at 22:42

Tuesday, May 6. 2014

Less known Solaris features: pwait

This is a nifty small tool that i'm using quite often in scripts that stop something and do some tasks afterwards and i don't want to hassle around with the contract file system. It's not a cool feature, but it's useful and relatively less known. An example: As i wrote long ago, you should never use kill -9 because often the normal kill is intercepted by the application and it starts to do some clean up tasks first before really stopping the process. So just because kill has returned, it doesn't imply that the process is away. How do you wait for process to disappear?

With pwait you have a small tool for this task. Let's assume you have tool running in the background

jmoekamp@kusanagi:~$ nohup sleep 500 &
[1] 7046

When you start pwait with the PID as the parameter, it just waits and doesn't return to the shell.

jmoekamp@kusanagi:~$ pwait 7046

Now we kill the process

jmoekamp@kusanagi:~$ kill 7046
jmoekamp@kusanagi:~$

As soon as the process is away, the pwait will return to shell.

jmoekamp@kusanagi:~$ pwait 7046
jmoekamp@kusanagi:~$

Posted by Joerg Moellenkamp in English, Solaris at 05:35

Sunday, May 4. 2014

Less known Solaris features: ptime

Long time readers of my blog know that i'm preferring prstat over top at any time. The micro state accounting in prstat gives you a much deeper insight. Using a tool not capable to use microstate accounting is like looking a video in 240p instead of 4k ultra hd (to stay at this: dtrace is like 8k). prstat is doing a really useful job in telling you what's happening at the moment in a processes.

However sometimes it's interesting to know, what happened in the past since the startup of the process. And there is a tool that is doing this. With ptimes -m you can lookup the information of the micro state accounting since the creation of the process:

```
root@kusanagi:~# ps -ef | grep "ssh"
jmoekamp  6174  6173   0 19:57:24 ?           0:00 /usr/lib/ssh/sshd
   root  6200  6188   0 19:59:06 pts/1       0:00 grep ssh
   root   539     1   0   Mai 01 ?           0:00 /usr/lib/ssh/sshd
   root  6173   539   0 19:57:24 ?           0:00 /usr/lib/ssh/sshd
root@kusanagi:~# ptime -m -p 6174

real     1:52.355845038
user        0.060116522
sys         0.049137178
trap        0.000004847
tflt        0.000000000
dflt        0.000000000
kflt        0.000000000
lock        0.000000000
slp      1:52.216832767
lat         0.029398950
stop        0.000010032
```

I'm using the tool quite often to get snapshots of the values for a process, writing it to a file and calculating the differences let's say over multiple hours. For this task it's much more practical than using prstat. A description of the values is in the man page, however it's pretty much the same as in prstat -m

Do you want to learn more
docs.oracle.com - manpage - ptime(1)

Posted by Joerg Moellenkamp in English, Solaris at 20:19

Saturday, May 3. 2014

No limits

Yet another interesting blog article written by Casper Dik: In"No limits" he reports about removing certain limits in Solaris 11.2

Posted by Joerg Moellenkamp in English, Solaris at 21:45

Thursday, May 1. 2014

Solaris 11.2 announcement - a collection of links (Part 4)

Casper gives a nice explanation about the Immutable Global Zone feature talking about the idea and how it behaves.

Alan Coppersmith has written down a list of all the changes to bundled software in Solaris 11.2 in his blog. It's quite a long list.

Last but not least, there is an interesting document about the usage of Kernel Zones and Unified Archives with SAS. It's available here. Key takeaway: Performance tests for SAS testing run in a kernel zone show little to no overhead on baseline tests.

Posted by Joerg Moellenkamp in General at 22:28