pfsense – Boot failure after upgrade to 2.4.0

Scenario

Upgraded from pfsense 2.3x to 2.4.0

Upon reboot, I was unable to ssh to the box.

Once at the physical console, I noticed pfsense had encountered a panic condition,
barking about not being able to mount /dev/ad0s1a

Troubleshooting

At the prompt, I typed in "?" to review the available block devices (disks and the like)

I saw in the output the device /dev/ada0s1a, a slightly different device path from what the error message referred to.

I then entered in: ufs:/dev/ada0s1a, and boom, pfsense kicked off its regular routines (although it did keep barking about this or that package needing to be cleaned and such)

The permanent fix was to correct the mount references in /etc/fstab.

I changed any reference to ad0 to ada0, rebooted, and voila.

Next time I upgrade pfsense, I'll read up on any known issues and the like.

Hint Hint:
2.4 New Features and Changes:

https://doc.pfsense.org/index.php/2.4_New_Features_and_Changes#Known_Issues

---

Kubernetes Deployment Error – PodToleratesNodeTaints

Scenario

You have a single node (master) kubernetes deployment and you want to schedule standard pods.

The master name is your hostname: $(hostname).

Upon your attempt at deploying a service, you notice the state of the resulting pod remains in *Pending*.

Further investigation via kubectl describe pod {{ YOUR_POD_NAME }} reveals an error similar to
No nodes are available that match all of the following predicates:: PodToleratesNodeTaints

Due Diligence:

- All kubernetes nodes are in a 'Ready' status: kubectl get nodes
- All kubernetes nodes have sufficient resources for pod deployment: kubectl describe nodes
- Your image is available on the docker registry you've specified in your kubernetes manifest (.yaml)

Troubleshooting

According to this post:

“No nodes are available that match all of the following predicates:: PodFitsHostPorts (1), PodToleratesNodeTaints” https://github.com/kubernetes/kubernetes/issues/49440

The troubleshooting methodology was to review the kubernetes codebase:

- Navigate to the kubernetes github repo
- Search the repository for the relevant function
- Kubernetes is written in golang, so search for “func PodToleratesNodeTaints”

As such, the following block of code:

```
if v1helper.TolerationsTolerateTaintsWithFilter(pod.Spec.Tolerations, taints, filter) {
return true, nil, nil
}
```

Will not be executed, which will trigger the next line of code:

```
return false, []algorithm.PredicateFailureReason{ErrTaintsTolerationsNotMatch}, nil
```

Effectively returning false, hence the original error

Further investigation on your master:

```
kubectl describe node $(hostname) | grep -i taint
```

If the command returns something similar to:

```
Taints: node-role.kubernetes.io/master:NoSchedule
```

Then your node is unschedulable.

The fix would be to remove this taint, as follows:

kubectl taint nodes $(hostname) node-role.kubernetes.io/master:NoSchedule-

You should see a confirmation similar to:

node {{ NODE_NAME }} untainted

You should now be able to schedule pods on this node

Notes

I came across the github issue description by Googling the following search term:

gls*"No nodes are available that match all of the following predicates" "PodToleratesNodeTaints"

Kubernetes, Docker volume mounts, and autofs

Environment details

- Machine_Type: Virtual
- OS: Oracle Enterprise Linux 7.x
- Software: Docker 1.12.6, Kubernetes 1.7.1

Scenario: Can’t Login via ssh public key

Unable to login to docker host using public key authentication
Able to login to the host using my password
Once at the console, I observed an error similar to:

```
Could not chdir to home directory /home/myuser: Too many levels of symbolic links
-bash: /home/myuser/.bash_profile: Too many levels of symbolic links
```

Hmm wtf …

Troubleshooting Steps

A fellow admin suggested I check for docker mapped volumes that point to /home

Here’s the command I used to query for that:

```
sudo docker ps --filter volume=/opt --format "Name:\n\t{{.Names}}\nID:\n\t{{.ID}}\nMounts:\n\t{{.Mounts}}\n"
```

Boom, looks like the kubernetes weaver container is using that mapping:

```
Name:
k8s_weave_weave-net-ljzn9_kube-system_740c10c5-d6b8-11e7-838f-005056b5384e_0
ID:
dc95801e4442
Mounts:
/opt/kubernetes,/lib/modules,/run/xtables.lo,
/var/lib/kubele,/var/lib/weave,/etc,/var/lib/dbus,/var/lib/kubele,/opt
```

Ok, so why would a docker volume mapped to /home induce such a problem?

Turns out that in some cases, binding autofs-mounted paths to docker containers can cause problems on the docker host.

This is due to the way in which kubernetes performs the volume mapping, which utilizes docker volume binds under the hood.

And, depending on how you map a volume to a docker container, you might conflict with autofs volume mounting.

For insight into a similar issue, see:

- Issue with AutoFS mounts and docker 1.11.2: https://github.com/moby/moby/issues/24303

According to the above issue description, the problem we're seeing might be fixed by adjusting the bind propagation for the volume mount in question,
see:
https://docs.docker.com/engine/admin/volumes/bind-mounts/#choosing-the–v-or-mount-flag

However, there's no way to control that setting via a kubernetes manifest, not at present at least, since HostPath bind propagation is currently a proposed feature in kubernetes,
see:
https://github.com/kubernetes/community/blob/master/contributors/design-proposals/node/propagation.md

So the best course of action is to simply change hostPath setting in the weave-kube manifest, e.g.

- Change:
  hostPath:
  path: /home
- To:
  hostPath:
  path: /opt/kubernetes/bind-mounts/weave-kube/home

You can then simply redeploy the offending container (sudo docker stop ecfa204283d3 && sudo docker rm ecfa204283d3 && kubectl apply -f net.yaml)

Note: You’ll have to perform similar changes to the weave manifest according to whatever other autofs mounts its hostPath(s) might conflict with.

Ensure you review your autofs settings!