Orchestrator - Bug #51817

FAILED tests/test_cephadm.py::TestShell::test_fsid - AttributeError: 'FakePipeWrapper' object has no attribute 'readable'

07/23/2021 12:10 AM - Sebastian Wagner

Status:	Resolved	**% Done:**	0%
Priority:	High		
Assignee:			
Category:			
Target version:			
Source:		**Affected Versions:**	
Tags:		**ceph-qa-suite:**	
Backport:		**Pull request ID:**	
Regression:	No	**Crash signature (v1):**	
Severity:	3 - minor	**Crash signature (v2):**	
Reviewed:			

Description

```
______________________________________________________________________________________________ Test
Shell.test_keyring ______________________________________________________________________________
_______________

self = <tests.test_cephadm.TestShell object at 0x7fcbcaeaa790>, cephadm_fs = <pyfakefs.fake_filesy
stem.FakeFilesystem object at 0x7fcbcaeaabb0>

    def test_keyring(self, cephadm_fs):
        cmd = ['shell']
        with with_cephadm_ctx(cmd) as ctx:
>           retval = cd.command_shell(ctx)

/home/sebastian/Repos/ceph/src/cephadm/tests/test_cephadm.py:1496: 
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
cephadm:1714: in _infer_config
    return func(ctx)
cephadm:1684: in _infer_fsid
    return func(ctx)
cephadm:1714: in _infer_config
    return func(ctx)
cephadm:1742: in _infer_image
    return func(ctx)
cephadm:4536: in command_shell
    mounts = get_container_mounts(ctx, ctx.fsid, daemon_type, daemon_id,
cephadm:2385: in get_container_mounts
    if HostFacts(ctx).selinux_enabled:
cephadm:6378: in __init__
    self.arch: str = platform.processor()
/usr/lib64/python3.9/platform.py:969: in processor
    return uname().processor
/usr/lib64/python3.9/functools.py:969: in __get__
    val = self.func(instance)
/usr/lib64/python3.9/platform.py:793: in processor
    return _unknown_as_blank(_Processor.get())
/usr/lib64/python3.9/platform.py:745: in get
    return func() or ''
/usr/lib64/python3.9/platform.py:764: in from_subprocess
    return subprocess.check_output(
```

```
/usr/lib64/python3.9/subprocess.py:424: in check_output
    return run(*popenargs, stdout=PIPE, timeout=timeout, check=True,
/usr/lib64/python3.9/subprocess.py:505: in run
    with Popen(*popenargs, **kwargs) as process:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _

self = <Popen: returncode: None args: ['uname', '-p']>, args = ['uname', '-p'], bufsize = -1, exec
utable = None, stdin = None, stdout = -1, stderr = -3, preexec_fn = None, close_fds = True, shell
= False
cwd = None, env = None, universal_newlines = None, startupinfo = None, creationflags = 0, restore_
signals = True, start_new_session = False, pass_fds = ()

    def __init__(self, args, bufsize=-1, executable=None,
                 stdin=None, stdout=None, stderr=None,
                 preexec_fn=None, close_fds=True,
                 shell=False, cwd=None, env=None, universal_newlines=None,
                 startupinfo=None, creationflags=0,
                 restore_signals=True, start_new_session=False,
                 pass_fds=(), *, user=None, group=None, extra_groups=None,
                 encoding=None, errors=None, text=None, umask=-1):
        """Create new Popen instance."""
        _cleanup()
        # Held while anything is calling waitpid before returncode has been
        # updated to prevent clobbering returncode if wait() or poll() are
        # called from multiple threads at once.  After acquiring the lock,
        # code must re-check self.returncode to see if another thread just
        # finished a waitpid() call.
        self._waitpid_lock = threading.Lock()

        self._input = None
        self._communication_started = False
        if bufsize is None:
            bufsize = -1  # Restore default
        if not isinstance(bufsize, int):
            raise TypeError("bufsize must be an integer")

        if _mswindows:
            if preexec_fn is not None:
                raise ValueError("preexec_fn is not supported on Windows "
                                 "platforms")
        else:
            # POSIX
            if pass_fds and not close_fds:
                warnings.warn("pass_fds overriding close_fds.", RuntimeWarning)
                close_fds = True
            if startupinfo is not None:
                raise ValueError("startupinfo is only supported on Windows "
                                 "platforms")
            if creationflags != 0:
                raise ValueError("creationflags is only supported on Windows "
                                 "platforms")

        self.args = args
        self.stdin = None
        self.stdout = None
        self.stderr = None
        self.pid = None
        self.returncode = None
        self.encoding = encoding
        self.errors = errors

        # Validate the combinations of text and universal_newlines
        if (text is not None and universal_newlines is not None
            and bool(universal_newlines) != bool(text)):
            raise SubprocessError('Cannot disambiguate when both text '
```

```
                                'and universal_newlines are supplied but '
                                'different. Pass one or the other.')

        # Input and output objects. The general principle is like
        # this:
        #
        # Parent                   Child
        # ------                   -----
        # p2cwrite   ---stdin--->  p2cread
        # c2pread    <--stdout---  c2pwrite
        # errread    <--stderr---  errwrite
        #
        # On POSIX, the child objects are file descriptors.  On
        # Windows, these are Windows file handles.  The parent objects
        # are file descriptors on both platforms.  The parent objects
        # are -1 when not using PIPEs. The child objects are -1
        # when not redirecting.

        (p2cread, p2cwrite,
         c2pread, c2pwrite,
         errread, errwrite) = self._get_handles(stdin, stdout, stderr)

        # We wrap OS handles *before* launching the child, otherwise a
        # quickly terminating child could make our fds unwrappable
        # (see #8458).

        if _mswindows:
            if p2cwrite != -1:
                p2cwrite = msvcrt.open_osfhandle(p2cwrite.Detach(), 0)
            if c2pread != -1:
                c2pread = msvcrt.open_osfhandle(c2pread.Detach(), 0)
            if errread != -1:
                errread = msvcrt.open_osfhandle(errread.Detach(), 0)

        self.text_mode = encoding or errors or text or universal_newlines

        # How long to resume waiting on a child after the first ^C.
        # There is no right value for this.  The purpose is to be polite
        # yet remain good for interactive users trying to exit a tool.
        self._sigint_wait_secs = 0.25  # 1/xkcd221.getRandomNumber()

        self._closed_child_pipe_fds = False

        if self.text_mode:
            if bufsize == 1:
                line_buffering = True
                # Use the default buffer size for the underlying binary streams
                # since they don't support line buffering.
                bufsize = -1
            else:
                line_buffering = False

        gid = None
        if group is not None:
            if not hasattr(os, 'setregid'):
                raise ValueError("The 'group' parameter is not supported on the "
                                 "current platform")

            elif isinstance(group, str):
                if grp is None:
                    raise ValueError("The group parameter cannot be a string "
                                     "on systems without the grp module")

                gid = grp.getgrnam(group).gr_gid
            elif isinstance(group, int):
                gid = group
            else:
```

```
                raise TypeError("Group must be a string or an integer, not {}"
                                .format(type(group)))

            if gid < 0:
                raise ValueError(f"Group ID cannot be negative, got {gid}")

        gids = None
        if extra_groups is not None:
            if not hasattr(os, 'setgroups'):
                raise ValueError("The 'extra_groups' parameter is not "
                                 "supported on the current platform")

            elif isinstance(extra_groups, str):
                raise ValueError("Groups must be a list, not a string")

            gids = []
            for extra_group in extra_groups:
                if isinstance(extra_group, str):
                    if grp is None:
                        raise ValueError("Items in extra_groups cannot be "
                                         "strings on systems without the "
                                         "grp module")

                    gids.append(grp.getgrnam(extra_group).gr_gid)
                elif isinstance(extra_group, int):
                    gids.append(extra_group)
                else:
                    raise TypeError("Items in extra_groups must be a string "
                                    "or integer, not {}"
                                    .format(type(extra_group)))

            # make sure that the gids are all positive here so we can do less
            # checking in the C code
            for gid_check in gids:
                if gid_check < 0:
                    raise ValueError(f"Group ID cannot be negative, got {gid_check}")

        uid = None
        if user is not None:
            if not hasattr(os, 'setreuid'):
                raise ValueError("The 'user' parameter is not supported on "
                                 "the current platform")

            elif isinstance(user, str):
                if pwd is None:
                    raise ValueError("The user parameter cannot be a string "
                                     "on systems without the pwd module")

                uid = pwd.getpwnam(user).pw_uid
            elif isinstance(user, int):
                uid = user
            else:
                raise TypeError("User must be a string or an integer")

            if uid < 0:
                raise ValueError(f"User ID cannot be negative, got {uid}")

        try:
            if p2cwrite != -1:
                self.stdin = io.open(p2cwrite, 'wb', bufsize)
                if self.text_mode:
                    self.stdin = io.TextIOWrapper(self.stdin, write_through=True,
                            line_buffering=line_buffering,
                            encoding=encoding, errors=errors)
            if c2pread != -1:
                self.stdout = io.open(c2pread, 'rb', bufsize)
                if self.text_mode:
```

```
>                   self.stdout = io.TextIOWrapper(self.stdout,
                                  encoding=encoding, errors=errors)
E                                 AttributeError: 'FakePipeWrapper' object has no attribute 'readable'

/usr/lib64/python3.9/subprocess.py:943: AttributeError
```

```
(py3) 
 cephadm git:(master) 
 pip list
Package    Version
---------- -------
attrs      21.2.0
iniconfig  1.1.1
mock       4.0.3
packaging  21.0
pip        21.1.2
pluggy     0.13.1
py         1.10.0
pyfakefs   4.5.0
pyparsing  2.4.7
pytest     6.2.4
setuptools 57.0.0
toml       0.10.2
wheel      0.36.2
WARNING: You are using pip version 21.1.2; however, version 21.1.3 is available.
You should consider upgrading via the '/home/sebastian/Repos/ceph/src/cephadm/.tox/py3/bin/python
-m pip install --upgrade pip' command.
```

History

#1 - 08/16/2021 12:49 PM - Sebastian Wagner

- Status changed from New to Resolved

https://github.com/ceph/ceph/pull/42664