Runtime Infrastructure - Support #3303

make the automated deployment process fully production ready

06/26/2017 08:48 AM - Greg Shah

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Roger Borrello	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 11/10/2017 02:59 PM - Greg Shah

- Assignee changed from Greg Shah to Constantin Asofiei

We have a documented process for the automated creation of a FWD VM.

The automation is built using Ansible. The existing implementation is quite good, but there is a list of enhancements needed to make this solution suitable for production deployments. This task is meant to refine and enhance the existing process to meet these requirements.

Some of these requirements provide new functional capabilities for the process, others are designed to improve security/reliability of the result and some requirements are there to make the process simpler and more foolproof.

Current Requirements

1. Eliminate the requirement for a CONTROLLER system. The automation should be able to be launched/executed directly on the TARGET system.

The existing process requires 2 systems, which significantly increases the complexity of the process (and the number of failure points). There is a command line helper program called ansible-pull that implements a local execution of a playbook that is checked out using a version control system (e.g. bzr). This will be used OR if ansible-pull can't do what we need, we will implement our own approach that is based on the same concept.

2. Add support for a unified process for VM creation and setup on first boot. Modern cloud environments use OS images that are naturally cloud-aware. In AWS or in an on-premises KVM installation, one can use an Ubuntu server distribution which is a "cloud image": https://cloud-images.ubuntu.com/. This is a pre-installed VM image that has a technology called "cloud-init": http://cloudinit.readthedocs.io/en/latest/index.html built in. cloud-init automates the customization of the VM the first time the VM is booted. This customization includes disk partitioning, networking setup, locale settings, timezone, software installation and the execution of arbitrary scripts/code that can be used as a hook for implementing any other automated setup. We will use this to bootstrap our automated ansible process. The result will allow a new FWD VM to be created in minutes with less effort than before.

3. Support updated versions of software including Ubuntu 18.04 LTS and PostgreSQL 10.0 (if easily possible, fallback is 9.5). The existing process uses 16.04 and PostgreSQL 9.5. To reduce future effort in updating deployments, we want to use the latest code that can be used without problems.

Since we will be using Ubuntu 18.04 LTS, we will design the playbooks, roles etc... for Ansible 2.5 which is the version in the Ubuntu repos for 18.04. That version adds a significant amount of new capabilities which we previously had to work around (the current version of the automation scripting is limited to Ansible 2.0).

4. Implement selectable modes. The idea is to choose a different master playbook (controls the automation) to match the mode being requested.

- runtime
  - "base" (FWD system + no application)
  - "application" (FWD system + application, no per-customer stuff)
  - "full" (FWD system + application + db import and runtime setup)
- conversion (can be used for working on a conversion project, for analytics or for a build system)
  - chui template
  - gui template
  - application
    - required input is a pre-defined project configuration
    - optional inputs for sources/schemata (if the source code and/or schemata are included with the project configuration, then these should be bypassed)
    - provide examples for Hotel ChUI and Hotel GUI
- analytics (this is a conversion system which just runs the front end)

- developer (full runtime and conversion system, add dev tools too)

5. Provide a generic approach to provide application-specific and customer-specific inputs.

Some selectable modes require additional files to be downloaded for the mode to work. This may include resources, code, scripts and maybe even Ansible playbooks.

- When an application is being installed, the code (source/schema or preconverted jars) and project configuration must be supplied. This is the "app files" download.
- When a customer system is being installed, the import data (.d files) and runtime configuration must be supplied. This is the "cust files" download.

The selected mode will need to be bootstrapped. This will require a download mechanism, configuration for the URL, optional authentication and a standard for the contents/naming of what gets downloaded.

What mechanisms will be provided to host files in a secure way?

6. Application-level per-user setup.

This might be some directories created, files placed, permissions set properly and so forth. There should be an optional application-specific script to be run for this.

This will need to be run for each application user created, whether it is during the initial installation or later.

It should allow for a download of files that can be cached as a kind of template user environment that can be copied.

7. Provide a script for user creation.

This would also run the per-user setup processing.

I think that the OS-level user delete, user disable and password reset should be fine as is.

8. Ownership/Permissions lock down for the /opt/<app_name>/ directory structure.

We need to minimize world-access to just enough for running the client. Everything else should be invisible and inaccessible.

- If possible, can we eliminate such access?
- Do we need to enhance our spawning to add/improve:
  - Better control over the working directory, in both virtual desktop and embedded mode.
  - Server-specified environment variables.
- The Swing client is not in usage in this VM scenario (at least not for the web users), so access to client.sh is probably not needed.

9. Multiple user account creation.

It should be possible to create more than one user account on the system. I think the best way to handle this is with comma-separated values in user_acct_name and user_acct_pw.

Please also add logic to ensure that a user account can never be the name fwd (which is the server account and thus is security sensitive).

Our web client configuration will need to be set to work with these accounts.

10. Support multiple database instances (creation/import).

11. The admin console account in FWD needs to have its password set to something passed on the command line on install.sh. We don't want to ever use the default admin password in any real system.

12. Make sure that the deploy.dist ant target creates the application artifacts in the form needed by the deployment processes.

13. Make sure that the ncurses patching works. In a customer's initial test in AWS, the ncurses patching failed as part of the ansible install. Manually running it worked.

14. Port range control.

Allow the specification of the size of the port range to use for clients. This will need to change both the firewall as well as the directory configuration.

15. Firewall rules customized to roles.

There is already a firewall implementation, but it should be customized based on the roles implemented for a given host. For example, 9443 should only be open on a host that supports analytics.

16. Let's Encrypt SSL Certificates

The Ansible process should support 3 certificate modes:

- self-signed certs

- ACME/Let's Encrypt real certs
- bypass cert generation

In the first two cases, the keystores and directory.xml should be fully updated and ready to go when complete. A command line parameter should be used to control which mode we use. If I understand correctly, the first and third modes are already present in our current process. #3240 provides details on implementation of the ACME processing. Sergey can answer questions.

17. Backup Hooks

Provide a sample backup script and install it. The customer will have to edit that and provide a real version that meets their needed, but this should provide a good starting point.

18. Add skeletons support to the convert_in_place mode. Actually, any scenario where we are putting the code down (for a build system or dev system) could need this. Deal with it like Possenet. If this is specified, then we also need to generate the class_map.xml if one does not already exist.

19. Test the process in AWS.

20. Update the documentation and ensure that both KVM and AWS are handled. Also make sure that CPU, memory and disk requirements (for the VM) are addressed.

Future Improvements

Reverse Proxy Support

This is needed for deployments which are limited to accessing the application through a single port (most likely 443). **For the short term, this requirement can probably be deferred.**

Implementing this involves Apache server installation and some configuration. #2683 provides details on how to do this and Sergey can help. The core idea is that this will allow a single port to be used and multiplexed for multiple users.

The user should be able to bypass this and it should be bypassed by default.

When proxy mode is added, the firewall setup will have to be different for proxy and non-proxy mode.

Monitoring and Management

Implement an install for a standard monitoring system that should allow monitoring/management of cpu, memory, disk, logs and so forth.

#2 - 06/18/2018 11:52 AM - Greg Shah

- Subject changed from create a production reference VM that can be used for deployment of a full FWD system to make the automated deployment process fully production ready

- Assignee changed from Constantin Asofiei to Greg Shah

Constantin/Eric: Please review #3303-1 and post any suggestions or comments.

#3 - 06/19/2018 04:03 PM - Greg Shah

The following are my findings in regard to using ansible-pull.

How it Works

The command line launcher is /usr/bin/ansible-pull. This is Python code that calls ansible. The code detects the original program name and ultimately it invokes /usr/lib/python2.7/dist-packages/ansible/cli/pull.py.

Ultimately, this code does 2 things:

- Runs ansible using the source control system as the module (e.g. -m 'bzr') to do the checkout in the right location from the right URL.
- Runs ansible-playbook with a specially generated inventory file, --connection=local (no remote connection) and other parameters.

If we find any issues with the behavior, we can do everything ourselves as needed.

BTW, I tracked this code's location using this:
https://stackoverflow.com/questions/122327/how-do-i-find-the-location-of-my-python-site-packages-directory

Local Connection

This is really the key thing that makes the playbook run locally. This can be implemented outselves by passing --connection=local on the ansible or ansible-playbook command line OR it can be specified in an inventory file as ansible_connection=local (on the specific hostname that will be used).

This means that the default transport ssh is not used. The code runs "in place" using the current user's account. No ssh key setup is needed for this to work. This reduces the setup steps and it also eliminates the need for the CONTROLLER.

The sudoers still needs NOPASSWD configured OR one must use --ask-become-pass on the command line.

Inventory

By default, ansible-pull adds a -i 'localhost,' to the ansible-playbook command line, which is a way to implement an "inline" inventory file with only a single value. This can be overridden at the ansible-pull command line (using -i).

The ansible-playbook will be run with the following implicit limit option --limit='localhost,current_system_fully_qualified_hostname,127.0.0.1'. This allows a match to localhost default for inventory.

This seems a decent approach. It also works easily using ansible or ansible-playbook directly.

Examples:

```
ansible -i 'localhost,' localhost --connection=local -a 'ls -l'
ansible-playbook -i 'localhost,' --limit localhost --connection=local -b --ask-become-pass dist_upgrade.yml
```

Notice that the ansible-playbook version needs the --limit localhost to be specified.

An equivalent version can be encoded in a simple inventory file. Assume that this file is named hosts:

```
localhost  ansible_connection=local
```

Examples using this hosts file:

```
ansible -i hosts localhost -a 'ls -l'
ansible-playbook -i hosts --limit localhost -b --ask-become-pass dist_upgrade.yml
```

Notice that in this form, host vars can be specified for localhost. We use this here to avoid the need to add the --connection=local to the command

line.

The downside to this usage of localhost is that:

- It will break any groups and/or variables that are normally assigned via an inventory file, unless they are set for localhost.
- It will break any variables that are defined using the group_vars/ and host_vars/ directory structure, unless those are specified for group_vars/localhost/ and host_vars/localhost/.

These limitations may be avoided in local mode by using a more arbitrary name. It turns out that localhost has no specific meaning in the TCP/IP sense. Thus, the following works:

```
ansible -i 'whatever,' whatever --connection=local -a 'ls -l'
ansible-playbook -i 'whatever,' --limit whatever --connection=local -b --ask-become-pass dist_upgrade.yml
```

The forms using the hosts inventory file also works, assuming you have a whatever host in there.

Using the Same Playbooks in Local and Remote Modes

This is possible using either of these techniques:

- If you use localhost as the inventory name, then you must encode all playbooks using hosts: all. This ensures that the playbooks are not specific to a given host name. Using hosts: all allows any host name to be used, not just localhost.
- If you have playbooks that are encoded to specific hosts, then you must use the host name override (see whatever in the Inventory section above) to match the right host to the playbooks that are needed.

The FWD playbooks are all written using hosts: all, so this won't be a problem.

Current Directory

The source control checkout will execute from whatever directory is the current directory when the ansible-pull is executed.

After the code is checked out using the source control module, ansible-pull will change directory to the top of the checkout before running ansible-playbook.

+ansible-pull Command Line

Most of the options are for controlling the checkout. -U repo_url (this is the same as the bzr module name) and -d directory_for_checkout (same as bzr module dest) are required. For using bzr, you must add -m 'bzr'.

-C revision (or --checkout=revision) can be used to check out a specific version number (same as bzr module version).

-f (or --force) can be used to run the playbook even if the bzr checkout fails.

You can override the inventory file using -i.

The playbook can be explicitly specified. If not specified, then it will look for a file named local.yml in the top level directory of the checkout.

Approach for FWD

I don't see any particular reason to use ansible-pull. The syntax limits our choices and we have other bootstrapping needs anyway. This means that we will have some kind of bootstrapping script and thus we can just do an explicit bzr checkout and then our own ansible-playbook using the syntax shown above in the Inventory section.

Useful Links

Latest Documentation
Connection Type
ansible-pull Syntax

Some tutorials/articles describing the idea:
https://calgaryrhce.ca/blog/2016/02/03/ansible-pull-and-kickstart-for-one-touch-server-provisioning/
https://www.reddit.com/r/devops/comments/6fajam/ansible_in_pull_mode/
https://www.trainingdevops.com/training-material/ansible-workshop/ansible-pull-introduction
https://rbgeek.wordpress.com/2016/05/15/autoscaling-with-aws-instances-using-ansible-pull/

An example playbook for installing ansible-pull as a cron job:
https://github.com/ansible/ansible-examples/blob/master/language_features/ansible_pull.yml

#4 - 01/28/2021 05:51 PM - Greg Shah

- Assignee changed from Greg Shah to Roger Borrello

It is time to take the next steps with Ansible. Roger will be "skilling up" in Ansible and when ready he will execute the following:

1. Our first pass Using Ansible for Management of a FWD Ubuntu System was built for Ubuntu 16.04 and an older version of FWD. We need to:
   - Move to the current Ubuntu LTS (20.04).
   - Use the version of Ansible (2.9) that comes with that version.
   - Ensure that the latest FWD, PostgreSQL etc... all are supported properly.
   - Any changes can be checked into the bzr repo. We do NOT need to continue support for older versions since we can always use the existing revision of the project for that purpose.
2. Eliminate the requirement for a controller system, ensuring that it can be run in local mode on a fresh VM/Ubuntu install. I have done quite a bit of research on this in #3303-3. I've also done some work with "pull node" in internal GCD usage of Ansible. I want this sub-task worked 2nd because I think that using the latest Ansible is going to make the work much easier.
3. Discuss the rest of the requirements for this task, add anything missing, clarify anything that is unclear.
4. Implement the other production deployment features.

#5 - 02/08/2021 03:39 PM - Roger Borrello

One of the "connection points" between our projects and the use of Ansible for deployment is the deploy.dist target. If we bolster this connection, and generate output that can be used directly with the playbooks, it will be a win-win. The Using Ansible for Management of a FWD Ubuntu System wiki shows a set of files to be included in that target.

From a current project, the build.xml contains:

```
   <target name="deploy.dist" description="Create an archive with the application jars and other files require
d for deployment on other systems.">
      <mkdir dir="${dist.home}/"/>
      <zip destfile="${dist.home}/${appname}_jars_deploy_${DSTAMP}.zip" basedir="${basedir}"
           includes="build/lib/*.jar
                     adm_windows.json
                     ddl/*
                     data/namespace/*
                     cfg/registry.xml" />
   </target>
```

I'd like to make sure the adm_windows.json is no longer needed. As well, the build/lib/*.jar should probably be build/*. What about cfg/registry.xml? Is that necessary?

I'd also propose addition deploy.dist targets such as deploy.dist.debug, which would create an archive of the src/* files and perhaps *.cache, although that might help to have a separate target, like deploy.dist.artifacts. The deploy.dist target could build the other 2.

#6 - 02/09/2021 07:26 AM - Greg Shah

> If we bolster this connection, and generate output that can be used directly with the playbooks, it will be a win-win.

Agreed. As long long as the "connection" is simply making the deploy.dist target do exactly what is needed in the Ansible paybooks. I don't want any direct tie other than that.

> adm_windows.json

This is related to embedded mode. It was needed at one time but may no longer be needed. Constantin?

> build/lib/*.jar should probably be build/*

As far as I know, this is not needed.

> cfg/registry.xml

I think it is still needed. I think it is used during dynamic query conversion at runtime. We do need to eliminate this dependency. It is a poor approach to require misc conversion artifacts as part of the file system deployment for an application.

> I'd also propose addition deploy.dist targets such as deploy.dist.debug, which would create an archive of the src/* files and perhaps *.cache, although that might help to have a separate target, like deploy.dist.artifacts.

It is a good idea. You can look inside Eugenie's "abl" and "convert" archives to see what should be automatically generated.

#7 - 02/22/2021 05:50 PM - Roger Borrello

Greg Shah wrote:

> build/lib/*.jar should probably be build/*

As far as I know, this is not needed.

So the build/classes.aop is **not** needed? That is good, because they can be very numerous, depending on the project.

#8 - 02/22/2021 07:03 PM - Roger Borrello

Greg Shah wrote:

> The command line launcher is /usr/bin/ansible-pull. This is Python code that calls ansible. The code detects the original program name and ultimately it invokes /usr/lib/python2.7/dist-packages/ansible/cli/pull.py.

I don't see /usr/bin/ansible-pull in Ubuntu 20.04. I do see it on my Linux Mint, which corresponds to 18.04 Ubuntu.

I see Ansible documentation (https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#installing-ansible-on-ubuntu) has instructions for installing on Ubuntu:

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo apt-add-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

#9 - 02/23/2021 07:19 AM - Greg Shah

> So the build/classes.aop is not needed?

I don'tknow for sure.

Hynek: is this something from AspectJ? Is it useful for a deploy.dist step?

> I see Ansible documentation (https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#installing-ansible-on-ubuntu) has instructions for installing on Ubuntu:

This is the process to use their PPA (custom repo). They like to document this approach so that you can stay bleeding edge with their builds.

We don't need this. It can be as simple as sudo apt-get instal ansible. On 20.04 Ansible is at 2.9 which is very recent. There is no need for a special PPA.

#10 - 02/23/2021 07:43 AM - Hynek Cihlar

Greg Shah wrote:

> So the build/classes.aop is not needed?

> I don'tknow for sure.
>
> Hynek: is this something from AspectJ? Is it useful for a deploy.dist step?

build/classes.aop is the output dir of AspectJ compiler (ajc). The dir is used as the source when building p2j*.jar files, but it is otherwise not used for the deployment build targets.

#12 - 12/07/2023 08:13 PM - Roger Borrello

I have been looking closer at mgmt since I'd like to use it to setup a customer's VM for supporting Docker images. It seems the mgmt isn't really written to use a pull of the playbooks, which is what I really need in this use case, since the system base OS is Ubuntu 22.04, and the only system I can push from is Ubuntu 18.04, and seems to be incompatible. The host is reporting:

```
fwd@fwd-dev:~/projects/mgmt$ ./doit_docker.sh -v
Running (1): ansible-playbook -v -i fwd_inventory_file.txt --limit docker -u fwd -b --ask-become-pass --extra-
vars "fwd_srv_hostname=docker  os_language=en_GB os_encoding=UTF-8 os_timezone=Etc/UTC" docker_host.yml
Using /etc/ansible/ansible.cfg as config file
SUDO password:

PLAY [Docker Host 1] ***********************************************************************************
**********************

TASK [Gathering Facts] *********************************************************************************
**********************
fatal: [docker]: FAILED! => {"changed": false, "module_stderr": "Shared connection to docker closed.\r\n", "mo
dule_stdout": "\r\nTraceback (most recent call last):\r\n  File \"/tmp/ansible_90gu5avh/ansible_module_setup.p
y\", line 125, in <module>\r\n    from ansible.module_utils.basic import AnsibleModule\r\n  File \"/tmp/ansibl
e_90gu5avh/ansible_modlib.zip/ansible/module_utils/basic.py\", line 80, in <module>\r\nImportError: cannot imp
ort name 'Mapping' from 'collections' (/usr/lib/python3.10/collections/__init__.py)\r\n", "msg": "MODULE FAILU
RE", "rc": 1}
        to retry, use: --limit @/home/fwd/projects/mgmt/docker_host.retry
```

So the host is passing a python program that isn't compatible with the newer python3 3.10. If I could get the target system to 3.9, it might work, but then that would defeat the whole purpose of using Ansible.

#13 - 12/08/2023 08:13 AM - Greg Shah

The playbooks are 3 years old, were never 100% complete and we never got local execution running (even though we do very much want it to work).

So: please fix it and make it work locally.

#14 - 12/12/2023 06:22 PM - Roger Borrello

Branch 3303 was created from trunk_14870. I checked in an update to allow for locales to be included in the convert archive:

```
=== modified file 'build.xml'
--- old/build.xml    2023-10-11 13:56:42 +0000
+++ new/build.xml    2023-12-12 23:19:31 +0000
@@ -258,6 +258,7 @@
 ** 109 OM  20230510          Included schema/import.xml in jar to be accessible for at import time.
 **     RFB 20230831          Include new import.sh tool in the convert archive. Reformatted this header.
 ** 110 GBB 20231023          Adding new login and logout html resources.
+** 111 RFB 20231212          Include locales directory in the convert archive. Ref. #3303.
 -->

 <!--
@@ -1085,7 +1086,8 @@
                     prefix="build/lib/spi"
```

```
                     includes="fwdspi.jar" />
        <zipfileset dir="${basedir}"
-                    includes="rules/**
+                    includes="locale/**
+                              rules/**
                               src/4gl/embedded/**
                               src/com/goldencode/**" />
        <zipfileset dir="${basedir}"
```

Would be great to merge this to trunk soon.

#15 - 12/12/2023 06:33 PM - Roger Borrello

Greg Shah wrote:

> The playbooks are 3 years old, were never 100% complete and we never got local execution running (even though we do very much want it to work).
>
> So: please fix it and make it work locally.

I checked in an update to mgmt which allows install.sh to receive -local option, and will allow this to run locally. The docker_host role was added. I can base other roles off this. Perhaps we focus on:

1. Role with FWD (can be used for runtime or build). This would pull the latest trunk (or a specific branch if specified) using **bzr** off xfer.goldencode.com. It will prompt for the xfer ID/password for the pull.
2. Role with FWD with PostgreSQL. This would be based off the FWD role, and add pg14. It could be used for runtime.
3. Role with hotel and hotel_gui in place for building either.

Any of those would include docker, so that it minimizes the variation. The size difference would be very minimal. The only reason I could think of for not including base items like **ant** and **gcc** (JDK would be in the base) would be for some sort of security issues.

#16 - 12/13/2023 07:07 AM - Greg Shah

I generally like this. Let's consider the idea that roles can be additive. By splitting these into "lego blocks" we can mix and match them to build systems as needed.

> Role with FWD (can be used for runtime or build). This would pull the latest trunk (or a specific branch if specified) using bzr off xfer.goldencode.com. It will prompt for the xfer ID/password for the pull.

There are 2 levels of "build": A) building FWD itself and B) converting/compiling a 4GL application. We must support both. Considering that we will certainly be creating a FWD build system (where it just creates binaries of FWD), we probably should split these into 2 roles.

And the runtime part is another role.

> Role with FWD with PostgreSQL. This would be based off the FWD role, and add pg14. It could be used for runtime.

This is just a pg14 role. We can add it to the FWD runtime role.

> Role with hotel and hotel_gui in place for building either.

This is 2 roles (one for each project).

#17 - 12/14/2023 10:19 AM - Roger Borrello

May I suggest Docker should be the main basis for the build and runtime roles? This would allow us to pull "code" from the goldencode docker.io repository for conversion or runtime. (Another reason why 3303a needs to be merged to trunk :-)

Note how easily it is to setup FWD so it can be made available:

```
rfb@rfb:~/projects/fwd$ docker run -v /tmp:/sample --user root -it --rm goldencode/fwd_4.0_ubuntu_22.04:4722b_
latest sh -c "(cd /home/fwd/repo; ./deploy_fwd.sh /sample/fwd-deploy convert -f)"
ERROR: Cannot locate p2j.jar (./build/lib/p2j.jar) looking for version.properties in current directory...
Using version: major=4 minor=0 release=0 repo=p2j branch=4722b rev=14876
Deploying: convert
rfb@rfb:~/projects/fwd$ ls -l /tmp/fwd-deploy/convert/
total 32
drwxr-xr-x  4 root root  4096 Dec 14 08:02 build
-rwxr-xr-x  1 root root 17112 Nov 20 21:23 import.sh
drwxr-xr-x 15 root root  4096 Nov  3 00:30 rules
drwxr-xr-x  4 root root  4096 Nov  3 00:30 src
rfb@rfb:~/projects/fwd$
```

Then it's just a matter of linking up to the location where FWD was place. Obviously you'd use something more useful then /tmp

#18 - 12/14/2023 08:38 PM - Roger Borrello

The latest revision builds a FWD reference system utilizing Docker image to retrieve the FWD deployment. Right now it will only work with H2, since we don't have the locales necessary to build the postgres cluster. That would need 3303a.

One thing missing is the positioning of the /opt/spawner directory for runtime usage.

#19 - 12/18/2023 04:39 PM - Roger Borrello

Right now I have the roles for building a Docker host as:

- base
- fwd_acct
- dev_tools
- jdk
- ncurses
- interactive_users
- docker
- reboot_here

After testing, I think I should remove at least **ncurses**, perhaps **jdk** as well. It all depends on how "thin" we want the docker host to be.

#20 - 12/19/2023 07:41 AM - Greg Shah

> After testing, I think I should remove at least **ncurses**, perhaps **jdk** as well. It all depends on how "thin" we want the docker host to be.

It should be anorexic. Strip out anything not needed on the docker host.

#21 - 12/19/2023 08:38 AM - Roger Borrello

The repository for ~/secure/code/p2j_repo/mgmt/ is now also available on **xfer.goldencode.com** as /opt/fwd/mgmt

#22 - 12/20/2023 12:00 AM - Roger Borrello

Branch 3303a was rebased to trunk_14886 and is at revision 14889.

#23 - 12/21/2023 01:49 PM - Roger Borrello

Branch 3303a was rebased to trunk_14896 and is at revision 14899.

#24 - 12/21/2023 01:57 PM - Greg Shah

There is no branch at ~/secure/code/p2j_repo/p2j/active/3303a/. Could you have misnamed the branch on creation?

#25 - 12/21/2023 02:00 PM - Roger Borrello

Greg Shah wrote:

> Could you have misnamed the branch on creation?

Ugh... I must dyslexic be... I created it as **3033a**. Is that something easily fixed by a mv?

#26 - 12/21/2023 02:02 PM - Constantin Asofiei

Roger Borrello wrote:

> Greg Shah wrote:
>
> > Could you have misnamed the branch on creation?
>
> Ugh... I must dyslexic be... I created it as **3033a**. Is that something easily fixed by a mv?

Yes, mv, then unbind/bind again.

#27 - 12/21/2023 02:25 PM - Roger Borrello

The incorrect branch name has been corrected to be **3303a**

#28 - 12/21/2023 02:32 PM - Greg Shah

Code Review Task Branch 3303a Revision(s?) 14897 through 14899

I'm not sure why it shows 3 revisions but the change is fine.

You can merge it to trunk after 7026f.

#29 - 12/21/2023 02:42 PM - Roger Borrello

Greg Shah wrote:

> Code Review Task Branch 3303a Revision(s?) 14897 through 14899
>
> I'm not sure why it shows 3 revisions but the change is fine.
>
> You can merge it to trunk after 7026f.

OK.

#30 - 01/02/2024 08:09 AM - Roger Borrello

Branch 3303a was merged into trunk revision 14898 and archived.

#31 - 01/09/2024 08:54 AM - Greg Shah

> May I suggest Docker should be the main basis for the build and runtime roles?

I do want to leverage Docker to make many things easier. I can see it as a very common way for people to run conversions, run analytics , run import and to run test or production execution environments.

It must not be the only way and it probably can't be the preferred way but it should be a first class way.

> Note how easily it is to setup FWD so it can be made available:
> [...]
> Then it's just a matter of linking up to the location where FWD was place. Obviously you'd use something more useful then /tmp

I'm not sure exactly what you are showing here. In this example is deploying FWD just a matter of unzipping? The real use cases actually need to run FWD inside the container for it to be useful.

#32 - 01/09/2024 09:21 AM - Roger Borrello

Greg Shah wrote:

> > Note how easily it is to setup FWD so it can be made available:
> > [...]
> > Then it's just a matter of linking up to the location where FWD was place. Obviously you'd use something more useful then /tmp
>
> I'm not sure exactly what you are showing here. In this example is deploying FWD just a matter of unzipping? The real use cases actually need to run FWD inside the container for it to be useful.

That was showing how easily you can place FWD somewhere for use as runtime or conversion. In that use case, you wouldn't need to gather a

project or have a development environment... just access to the goldencode repository and viola!

```
docker run -v /tmp:/sample --user root -it --rm goldencode/fwd_4.0_ubuntu_22.04:latest sh -c "(cd /home/fwd/re
po; ./deploy_fwd.sh /sample/fwd-deploy convert -f)"
ln -f /tmp/fwd-deploy/convert p2j
```

It's just illustrating the benefit to having the FWD Docker image fully capable of deploying itself.