Git Repo System Admin Guide

Table of Contents

Overview

The intended audience for this Administrative Guide is the Event Management & Mediation (EM&M), (formerly Converged Event Management Platform (CEMP)), Configuration Management (CM) group. The associated user guide for this admin guide is the EM&M Git Users Guide. The Git repo CFX-EMM-GITSystem is maintained with all files used to setup and maintain the EMM GIT Repository System.

The (EM&M) CM Repository System is where version management of EM&M source code and other elements occurs. The DNS names of the servers involved are:

- emm-git1.sys.comcast.net emmputl-po-7p.sys.comcast.net with an IP of 172.28.98.231.
- emm-git2.sys.comcast.net cmputl-ch2-4p.sys.comcast.net with an IP of 172.28.179.157.

This system is based on the Git tool. Administration of the Git repositories is being done with Gitolite. The emm-git1 server is primary server and the emm-git2 server is the fail-over that is fully restored on a daily basis.

Notes:

1. The logon banners for the servers in this system has been adjusted since the git user access will be limited to gitolite controlled ssh sessions. Both emm-git:/etc/issue and emm-git:/etc/issue.net have been adjusted accordingly.
2. The files emm-git1,2:/root/README.txt file is maintained to communicate with any analyst that logs onto this system as "root" the intent and purpose of this system. This wiki document location is also mention in this README.txt file via the main primary index.html server files.
3. Followed instructions for Atlas Client. Also did the following:
    a. cd /etc/yum.repos.d/ && mkdir OLD_REPOS/ && mv atlas* OLD_REPOS/
    b. Add a new the RHEL 6.x x86_64 repo:
    c. atlas-client repo-join --repo-id 92
    d. Clean up Yum:
        i. yum clean all
        ii. yum list # This will download all package definitions

Prerequisites

VM servers where setup using the Service Catalog and entering "vm setup" in the search window, then click on **Go**.

Select "Add New VM" and a window will be opened for setting up a request for a new VM system.

You need to be aware of how the new VM will be added to the iTRC System.

Verify the VM server(s) have been setup in CADA Host Table SEO_Unix_Cemp_CM. If not, open JIRA issue with NSO Access Control: (This is what should be opened by the Service Catalog via Infrastructure Services -> Access Request -> Host Access Requests -> Linux/Unix CADA Kerberos Hosts)

1. Open JIRA AC project Issue, Type is Task
2. In summary, "Add new Host(s) to CADA Host Group"
3. Select "GUI Access - RSA or CADA" in Component/s field pulldown
4. Add text in Description of JIRA issue like this example:
   "Please add host emmutl-po-7p.sys.comcast.net to EMM CADA Host Group SEO_Unix_Cemp_CM."
5. CC EM&M CM Team, and any other appropriate manager(s) on this ticket.
   Note: - Reference example ticket AC-26941.

Verify CM team logon. May need to open Service Catalog ticket for the SAs to install the CADA RPMs on the server.

The access protocols supported for Git on this system are:

- http - read only via **snapshot** of given project branch/release.
- ssh - read write

Access

Access to this system is controlled by CADA, host group SEO_Unix_Cemp_CM via iTRC application EM&M Versioning System. Once conversion from the CEMP CVS repository to EMM-GIT is complete, the CADA host group SEO_Unix_CEMP_CM can have most all user groups removed from it's association other than CADA User group SEO_Unix_Cemp_CM_Admins.

Git user read/write access is via ssh keys. Git user read only access using Gitweb via emm-git1.sys.comcast.net primary server and emm-git2.sys.comcast.net fail-over server.

Operating Systems

The EM&M Git repository system is based two servers with the CentOS GNU/Linux operating system.

1. emm-git1.sys.comcast.net at the Potomac datacenter in Denver.
2. emm-git2.sys.comcast.net at the Chicago datacenter.

```
[root@emmutl-po-7p ~]# cat /etc/*release*
CentOS release 6.4 (Final)
LSB_VERSION=base-4.0-amd64:base-4.0-noarch:core-4.0-amd64:core-4.0-noarch:graphics-4.0-amd64:graphics-4.0-noarch:printing-4.0-amd64:p

cat: /etc/lsb-release.d: Is a directory
CentOS release 6.4 (Final)
CentOS release 6.4 (Final)
cpe:/o:centos:linux:6:GA
[root@emmutl-po-7p ~]# uname -a
Linux emmutl-po-7p.sys.comcast.net 2.6.32-358.23.2.el6.x86_64 #1 SMP Wed Oct 16 18:37:12 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
[root@emmutl-po-7p ~]#

[root@cmputl-ch2-4p ~]# cat /etc/*release*
CentOS release 6.3 (Final)
[root@cmputl-ch2-4p ~]# uname -a
Linux cmputl-ch2-4p.sys.comcast.net 2.6.32-279.22.1.el6.x86_64 #1 SMP Wed Feb 6 03:10:46 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
[root@cmputl-ch2-4p ~]# date
Fri Nov 8 01:29:26 UTC 2013
[root@cmputl-ch2-4p ~]#
```

Application Users

In Comcast, users are considered application users when:

- Access is granted only through ssh keys or sudo'ing from a valid Comcast userid.
- Passwords are not used and password aging is disabled.

The following application users have been setup on cmputl-ch2-4p.sys.comcast.net:

- git
- jenkins
- cmadmin
- apache

The following application users has been setup on emmutl-po-7p.sys.comcast.net:

- git
- cmadmin
- apache

```
cp /etc/passwd /etc/passwd-orig
cp /etc/group /etc/group-orig
cp /etc/shadow /etc/shadow-orig
cp /etc/gshadow /etc/gshadow-orig
useradd [userid]
example useradd -g git cmadmin
cp /etc/group /etc/group-
cp /etc/passwd /etc/passwd-
cp /etc/shadow /etc/shadow-
cp /etc/gshadow /etc/gshadow-
```

Fix password aging for usernames git, jenkins, and cmadmin:
chage -M 99999 <username>

Network DNS Names

DNS names where setup with the IPAM DNS System. Following is an example of an email you get from IPAM DNS after entering a request. You can use the links to see how to enter a request in this system. This example sets up emm-git2 on the Chicago VM system cmputl-ch2-4p.sys.comcast.net. For any new DNS names you setup, add them to the appropriate device records in iTRC as aliases.

> **From:** [CHQ -- IPAM-Requests]
> **Sent:** Friday, June 27, 2014 11:18 AM
> **To:** Wallace, Andrew
> **Subject:** Your request has been submitted: 172912
>
> emailDnsRequest
>
> | | |
> |---|---|
> | Type: | dns_request |
> | ID: | 172912 |
> | Request Status: | Submitted |
> | Requester: | Wallace, Andrew |
> | Requester(Email): | andrew_wallace@cable.comcast.com |
> | Requester(Phone): | (720) 267-2677/null |
> | CR#: | |
> | Assigned To: | |
> | Queue: | DNS_TEAM_QUEUE CHQ--IPAM-Requests@cable.comcast.com |
> | Notes: | |
>
> **Please note there is a 24 hour SLA. Once the ticket is processed, please allow 2-4 hours for propagation. Click here to edit request Click here to view (Read Only) request

You can use the iTRC System for cutting and pasting FQDN names.

Applications

The following applications are listed in the order they were installed on this server. If you are setting up a new server for this Git Repo System, it is recommended you review this document in it's entirety before starting this task.

Administrator Communication

For all administrators that may access the servers as "root" for this system, a README.txt file has been placed in the "root" userid home directory of each system server. This file is also maintained in the git repo CFX_EMM-GIT-System for each system server.

Git

The Git Fast Version Control system is being setup on this system to replace The CEMP CVS repository system. It has been installed as follows:

To install Git for the EM&M repository system, the following libraries that Git depends on need to be installed: curl, zlib, openssl, expat, and libiconv.

```
[root@cmputl-ch2-4p ~]# yum install curl-devel expat-devel gettext-devel \
> openssl-devel zlib-devel
...
Complete!
```

Compile and Install

Notes:

- EPEL Does not work in this environment
- Do Compile and Install as root

download the latest git tar.gz from https://code.google.com/p/git-core/downloads/list to /root/downloads
Note: git-1.9.0.tar.gz was downloaded and added to the CFX_EMM-GIT-System git repo.

Verify SHA1 checksum with all git downloads, here an example:

```
[root@cmputl-ch2-4p tmp]# sha1sum git-1.9.0.tar.gz
e60667fc16e5a5f1cde46616b0458cc802707743 git-1.9.0.tar.gz
[root@cmputl-ch2-4p tmp]#
```

(Do the following in /usr/local/temp or src)

```
[root@cmputl-ch2-4p downloads]# tar -zxf git-1.9.0.tar.gz
```

[root@cmputl-ch2-4p downloads]# cd git-1.9.0
[root@cmputl-ch2-4p downloads]# make prefix=/usr/local all
{Compile output ....}
[root@cmputl-ch2-4p git-1.9.0]# make prefix=/usr/local install
{Install output ....}
[root@cmputl-ch2-4p git-1.9.0]#

Install Man Pages

download the latest git-manpages tar.gz from https://code.google.com/p/git-core/downloads/list to /root/downloads
Note: git-manpages-1.9.0.tar.gz was downloaded and added to the CFX_EMM-GIT-System git repo.

[root@emmutl-po-7p downloads]# sha1sum git-manpages-1.9.0.tar.gz
cff590c92b4d1c8a143c078473140b653cc5d56a git-manpages-1.9.0.tar.gz
[root@emmutl-po-7p downloads]#
[root@cmputl-ch2-4p git-1.9.0]# cd /usr/local/share/man
[root@cmputl-ch2-4p man]# tar -zxvf /root/downloads/git-manpages-1.9.0.tar.gz

Verify

[root@cmputl-ch2-4p ~]# git --version
[root@cmputl-ch2-4p ~]# man git

- **NOTE:** Also verified that "yum update" did not change the git install from git version 1.9.0.

Git Upgrade Procedure

As root, follow instructions for Compile and Install, and Install Man Pages.
Update repo CFX-EMM-GITSystem with new downloads, removing old ones after testing.

yum update curl-devel expat-devel gettext-devel openssl-devel zlib-devel
(or more generally "yum update")
download git-[release number].tar.gz from https://code.google.com/p/git-core/downloads/list to git repo CFX_EMM-GIT-System.

Verify SHA1 checksum with all git downloads, here an example:
[root@cmputl-ch2-4p tmp]# sha1sum git-[release number].tar.gz
e60667fc16e5a5f1cde46616b0458cc802707743 git-1.9.0.tar.gz
[root@cmputl-ch2-4p tmp]#

Upload git-[release number].tar.gz from git repo CFX_EMM-GIT-System to upgrade server. (/root/upload maybe)
tar -zxf git-[release number].tar.gz
cd git-[release number]
make prefix=/usr/local all
make prefix=/usr/local install
cd /usr/local/share/man
tar -zxvf /root/downloads/git-manpages-[release number].tar.gz

Apache

Used to support read access to git repositories via the httpd daemon on this server. Also support git documentation.

Install httpd

[awalla5075k@cmputl-ch2-4p ~]$ sudo yum install httpd

Configure Apache

- Set the apache service to start on boot:

[awalla5075k@cmputl-ch2-4p ~]$ sudo chkconfig --levels 235 httpd on

- Enable name-based virtual hosting on port 80:

[awalla5075k@cmputl-ch2-4p ~]$ cd /etc/httpd/conf
[awalla5075k@cmputl-ch2-4p conf]$ sudo cp httpd.conf httpd.conf-orig

- Set permissions so git user can configure apache
  (Enables git user to configure apache for git usage. Removes need to logon as root for future maintenance.)

As root on both primary and fail-over systems:

```
cd /var/www
chmod -R g+w *
chgrp -R git *
cd /etc/httpd
chmod -R g+w conf conf.d
chgrp -R git conf conf.d
```

- As git, open the httpd configuration file located at /etc/httpd/conf/httpd.conf
- Un-comment the line containing the text NameVirtualHost *:80
- Save the file

Configure Virtual Hosts

Reference Virtualhost instructions at http://dev.antoinesolutions.com/apache-server/.

- Prepare CFX_EMM-GIT-System/git1,2-server/repository.conf for new server

```
[git@server] cd /etc/httpd/conf.d
[git@server] cp CFX_EMM-GIT-System/git1,2-server/repository.conf .
```

- Setup the Virtual host directory:
  [root@server]# cd /app
  [root@server]# mkdir git-repos
  [root@server]# chown git:git git-repos
  Note: cmputl-ch2-4p was setup with /db/mysql rather the /app. Using /app on new VMs moving forward.

Configure and Install Index Pages

- Prepare CFX_EMM-GIT-System/git1,2-server/noindex.html file
  - Diff with /var/www/error/noindex.html on new server, account for diffs.

```
[git@server]# cd /var/www/error
[git@server]# cp noindex.html noindex.html-orig
[git@server]# cp CFX_EMM-GIT-System/git1,2-server/noindex.html .
```

- Prepare CFX_EMM-GIT-System/git1,2-server/index.html file

```
[git@server]# cd /var/www/index
[git@server]# cp CFX_EMM-GIT-System/git1,2-server/index.html .
```

- Copy Git logo files to each EMM git1,2 server

```
[git@server]# cd /var/www/icons
[git@server]# cp CFX_EMM-GIT-System/git-logo.png .
[git@server]# cp CFX_EMM-GIT-System/gitweb-logo.png .
```

Restart Apache and Test

```
[awalla5075k@server]$ sudo service httpd restart
```

1. Verify "http://emm-git1.sys.comcast.net" or "http://emm-git2.sys.comcast.net" in browser accordingly.
2. rename /var/www/html/index.html hold.html
3. Verify noindex.html error condition
4. rename /var/www/html/hold.html index.html
5. Verify "http://emm-git1.sys.comcast.net" or "http://emm-git2.sys.comcast.net" in browser accordingly.

Gotolite

Application for managing ssh keys and general repository access. Reference section 4.8 of the Git Documentation.

Pre-Install

1. Read through all steps in this section before actually doing the install.
2. Instructions referenced for this initial install at http://gitolite.com/gitolite/qi.html.
3. Adhere to http://gitolite.com/gitolite/WARNINGS.html.
4. Since this system has both "Primary" and "Fail-Over" servers, be familiar with Moving Servers content (including links).
5. Download gitolite-master.zip from https://github.com/sitaramc/gitolite to Git repo CFX_EMM-Git-System.
6. Upload gitolite-master.zip to server:/tmp. Git user will copy from there.
   a. NOTE: Since we can't execute "git clone git://github.com/sitaramc/gitolite" as indicated in ~/gitolite/README.txt file, The previous steps were done.
7. Logon to server and sudo to git.

8. Follow instructions for "Installation and Setup" in ~/gitolite/README.txt.
    a. Note: While executing, gitolite/install -ln, got message "git describe failed; cannot deduce version number". To correct, see "Fix Gitolite version indicator" section below.
9. Go back to Moving Servers and make sure the "Primary" and "Fail-Over" servers are setup correctly.

Install

- **On windows workstation:**

1. Download zip from https://github.com/sitaramc/gitolite after setting site for desired release. (started with master)
    a. This was done by selecting branch under blue bar and identifying Master branch
2. Copy gitolite-master.zip to git repo CFX_EMM-GIT-System
    a. Keeps actual files used available for future reference.
3. cp gitolite-master.zip to [server:/tmp]
4. cp awalla5075.pub to [server:/tmp]

- **On server logged on as git:**
  [git@server]$ pwd
  /home/git
  [git@server]$ mkdir bin
  [git@server]$ cp /tmp/awalla5075k.pub .
  [git@server]$ cp /tmp/gitolite-master.zip .
  [git@server]$ unzip gitolite-master.zip
  [git@server]$ rm gitolite-master.zip
  [git@server]$ cd gitolite-master
  [git@server]$ ./install -to $HOME/bin
  [git@server]$ gitolite setup -pk awalla5075k.pub
  Initialized empty Git repository in /home/git/repositories/gitolite-admin.git/
  Initialized empty Git repository in /home/git/repositories/testing.git/
  WARNING: /home/git/.ssh missing; creating a new one
  (this is normal on a brand new install)
  WARNING: /home/git/.ssh/authorized_keys missing; creating a new one
  (this is normal on a brand new install)

- **In /home/git, mv contents of repositories to /app/git-repos. Create softlink repositories -> /app/git-repos**
- **Need to sudo chown git:git /app/git-repos**
  [git@server]$ cd repositories
  [git@server]$ mv * /app/git-repos
  [git@server]$ ln -s /app/git-repos repositories
- Update .gitolite.rc as follows:
    - Add Following line to ROLES definition under READERS and WRITERS:
      OWNERS => 1,
    - Follow instructions in .gitolite.rc for adding more roles.

- **Setup servers for backup/restore**
  Logged on fail-over (or primary as required) server as git:
  $ cd ~/.ssh
  $ ssh-keygen -t rsa -f git (create on one server, move to the other)
- Copy git and git.pub to git repo CFX-EMM-GitSystem
- Copy git and git.pub to primary server /home/git/.ssh directory
- Copy following files from CFX-EMM-GitSystem repo to primary and fail-over servers:
    - CFX-EMM-GITSystem/[primary-server|failover-server]/ssh_config-cmadmin to [primary/failover]:/home/git/.ssh/config
- On both primary and failover servers:
    - set permissions of following files:
      -rw-r----- 1 git git 1820 Jul 30 21:57 config
      -rw------- 1 git git 1675 Jul 30 22:03 git
      -rw-r----- 1 git git 415 Jul 30 22:04 git.pub
    - Update git.pub key as follows: (git ssh key should be after "# gitolite end" line)
      $ cat git.pub >> authorized_keys

- **On your workstation:**
  # clone the admin repo so you can start adding stuff
  git clone emm-git:gitolite-admin.git
  # Note 1: clone path must not include "repositories/"
  # Note 2: it may include the ".git" at the end but it is optional

- While testing:
  Found that the wrong information is in ~/.ssh/known_hosts file. Reference Appendix 4: ssh host aliases:

Adjust the system logon banner

Since gitolite controlled ssh sessions are the only access allowed, the following was done:

[root@server]# pwd
/etc
[root@server]# cp issue issue-orig
[root@server]# cp issue.net issue.net-orig

- copy issue and issue.net from appropriate server directory in git repo CFX-EMM-GIT-System to server:/etc

Gitolite New System and Restore Notes

- Upgrades, system changes, and fixes are tested on the fail-over system first (emm-git2); then scheduled for the primary system (emm-git1).

System Restore

After a new system has been setup, following all install related instructions in this guide to this point, using following steps as required:

On the new server logged on as git:

1. Copy all repositories from active EM&M Git System server to this new server's repository directory (should be /app/git-repos).
    a. Note: Left the new repos gitolite-admin.git and testing.git alone. Copied all others over:
        i. cfx-branch.git
        ii. cfx-emm-cm.git
        iii. CFX-MANILLA.git
        iv. DATA-XfinityWifi.git
        v. foo.git
2. Shut off "git push" on "fail-over" server. Run gitolite writable -h for more info.
   [git@server]$ gitolite writable @all off Git push disabled, fail-over server.
3. Copy the $HOME/.gitolite.rc file from the old server, overwriting the one on the new primary server.
4. Run gitolite setup.
5. Initialize backup functionality on this new server and restore functionality on current fail-over server. **Not yet done until testing of new system occurs.**

On your workstation, in the git rep gitolite-admin:

1. Initialize the gitolite.conf file with the currently active EMM Git System. This will ensure the new primary server is in sync with the access policies established and documented in the EMM Git User Guide:
    a. copy gitolite.conf to gitolite.conf-orig
    b. overwrite gitolite.conf with currently active gitolite.conf
2. Initialize the gitolite-admin repo's key directory with the currently active EMM Git System. This will ensure the correct set of user keys for the currently active EMM Git System. **Be Careful with the initial admin key for the new gitolite-admin repo**.
    a. copy ??/gitolite-admin/keydir/*.pub .
3. Check in changes to new gitolite-admin repo and push to new server.
4. Verify new Gitolite setup is functioning correctly by running the following:
    a. ssh git@emm-git info lc

This section will take more passes before complete.

Upgrading Gitolite

This section based on Gitolite Upgrading documentation. Update version number in this section when it is exercised. This indicates the current version installed in the EMM Git System.

This needs to be done on both EMM Git System servers starting with fail-over server for testing purposes.

Review Gitolite documentation that may be specific to the Gitolite release you are upgrading from to the release you are upgrading to. Be sure to review the "CHANGELOG" file at github sitaramc/gitolite.

1. Download zip from https://github.com/sitaramc/gitolite after setting site for desired release. (v3.6.1 currently)
    a. This was done by selecting tag under blue bar and identifying v3.6.1 tag
2. Copy gitolite-3.6.1.zip to git repo CFX-EMM-GITSystem
    a. Keeps actual files used available for future reference.
    b. After successful install and test, remove current gitolite-[release].zip from CFX_EMM-GITSystem
3. cp gitolite-3.6.1.zip to [server:/tmp]
4. cp awalla5075.pub to [server:/tmp]

- Update your clone of the gitolite source.
  On server logged on as git:
  [git@server]$ pwd
  /home/git
  [git@server]$ cp /tmp/awalla5075k.pub . (This should be the original admin key.)
  [git@server]$ cp /tmp/gitolite-3.6.1.zip .
  [git@server]$ unzip gitolite-3.6.1.zip
  [git@server]$ rm gitolite-3.6.1.zip

[git@server]$ rm -fr gitolite-master
[git@server]$ cd gitolite-3.6.1

- Repeat the install command you used earlier (make sure you use the same arguments as before).
  [git@server]$ ./install -to $HOME/bin
  git describe failed; cannot deduce version number
  [git@server]$
  (Due to previous "git describe failed" note, updated ~HOME/bin/VERSION with (v3.6.1))

- Run gitolite setup.
  [git@server]$ gitolite setup

Gitolite Admininstration

Link used for administrating Gitolite reference.

TODO:

http://gitolite.com/gitolite/emergencies.html#lost-key - Need to following section 2 Lost Admin Key to setup gitadmin for Bob, Bruce, and myself. Then test with Bob and Bruce having gitadmin private in their .ssh directories.

Be sure to gitadmin ssh key on both Primary and Failover. Don't forget to setup both primary and failover remotes on your client repo for gitolite-admin.

GitWeb

As root:

[root@cmputl-ch2-4p git]# pwd
/opt/git
[root@cmputl-ch2-4p git]# yum install gitweb
...
[root@cmputl-ch2-4p git]#

Update /etc/gitweb.conf as follows:
[root@server]$ cp gitweb.conf gitweb.conf-orig

- For emm-git1 set "our $projectroot =/app/git-repos";
- For emm-git2 set "our $projectroot = "/db/mysql/git-repo";

[root@cmputl-ch2-4p git]# service httpd restart

GitWeb Maintenance

GitWeb is configured from /etc/gitweb.conf. By default, the GitWeb view is controlled by file /home/git/projects.list. by adjusting /etc/gitweb.conf, (comment out $projects_list variable), the $projectroot variable setting (and file permissions) will dictate what is view-able by gitweb.

By default, the temporary repos created by analysts are not view-able from gitweb. Since each individual analyst can use "ssh git@emm-git info" to see what repos they have access to, it was determined the temporary repos do not need to be view-able from gitweb.

Initial DevLead repos are treated like temporary repos until they are moved to the permanent repository location and updated for GitWeb usage. **Currently documented this procedure in a gitolite.conf preamble while it is being tested. Once testing completed, both this doc and user guide will need to be updated accordingly.**

Note that further documentation on gitweb is available at directory /usr/share/doc/gitweb-1.7.1 of the cmputl-ch2-4p.sys.comcast.net server.

Gitolite Admin Updates Performed

To setup project or repo-name for http read access:

$ cd project.git
$ mv hooks/post-update.sample hooks/post-update
$ chmod a+x hooks/post-update
$ ./post-update

Fix Gitolite version indicator

- Logon to server as git

[git@server]$ cd bin
[git@server]$ ls -l
total 40
drwxrwxr-x 2 git git 4096 Jan 14 15:16 commands
-rwxr-xr-x 1 git git 3083 Jan 14 15:16 gitolite
-rwxr-xr-x 1 git git 8363 Jan 14 15:16 gitolite-shell
drwxrwxr-x 3 git git 4096 Jan 14 15:16 lib
drwxrwxr-x 2 git git 4096 Jan 14 15:16 syntactic-sugar
drwxrwxr-x 3 git git 4096 Jan 14 15:16 triggers
~~rw-rw-r-~~ 1 git git 9 Jan 20 19:45 VERSION
drwxrwxr-x 2 git git 4096 Jan 14 15:16 VREF
[git@cmputl-ch2-4p src]$ vi VERSION
(v3.x)
`
`
"VERSION" 1L, 7C written
[git@server]$

Add Description for gitolite-admin repo:

awalla5075k@CO183LCETENG08 ~/git-server-admin/gitolite-admin (master)
$ ssh git@emm-git info -lc -ld
hello awalla5075k, this is git@cmputl-ch2-4p running gitolite3 (v3.x) on git 1.8.4.4

R W cfx_emm_cm Cross Functional EMM Configuration Management
R W foo
R W gitolite-admin Unnamed repository; edit this file 'description' to name the reposit
R W testing

awalla5075k@CO183LCETENG08 ~/git-server-admin/gitolite-admin (master)
$ ssh git@emm-git desc gitolite-admin "Git repo used to administer the repos on the emm-git server."

awalla5075k@CO183LCETENG08 ~/git-server-admin/gitolite-admin (master)
$ ssh git@emm-git info -lc -ld
hello awalla5075k, this is git@cmputl-ch2-4p running gitolite3 (v3.x) on git 1.8.4.4

R W cfx_emm_cm Cross Functional EMM Configuration Management
R W foo
R W gitolite-admin Git repo used to administer the repos on the emm-git server.
R W testing

awalla5075k@CO183LCETENG08 ~/git-server-admin/gitolite-admin (master)
$

MagNETO and puppet study

Nov 25th:
Link from Eric Hoopes -> http://community.teamcomcast.com/i/b6/MagNETO/default.aspx

Setup EMM group on GitLab

Setup project CFX-EMM-GITSystem repo as project in EMM Group on GitLab

- Added Bob Sell and Hitesh as members of EMM Group on GitLab.

Nov 26th:

Setup ajwalla2014.wordpress.com for gravatar.com in support of magneto gitlab profile setting

Review MagNETO Overview:

Puppet Docs – http://docs.puppetlabs.com
Git Docs – http://git-scm.com/book/en
Gitlab Docs - http://gitlab.org/gitlab-ce/

MagNETO Wiki Docs https://wiki.io.comcast.net/display/IDEA/MagNETO+-+IDEA+Automation+Puppet+Infrastructure

MagNETO Comcast Community http://community.teamcomcast.com/i/b6/MagNETO/default.aspx

APS IDEA Automation Team
NETO-AutomationTeam@cable.comcast.com

EMM Usage:

Single Repo - Multiple branches (as opposed to Multi Repo - Single branch)

- develop branch - automated build, automated develop deploy (main branch)
- Release branch - automated deploy (supporting branch)
- Production branch - automated production deploy (main master branch)
    - NOTE: Gitolite allows access control at the branch level, the magneto implementation does not

Setup VM learn_puppet_centos-6.5-pe-3.3.2
Completed:

- Welcome Quest
- Power of Puppet Quest

Nov 28th

Review Welcome Quest and Power of Puppet Quest 1.5h

Review Resources and Manifests Quests 1.5h

Resources Quest

- Understand how resources on the system are modeled in Puppet's Domain Specific Language (DSL)
- Learn about the Resource Abstraction Layer (RAL)
- Use Puppet to inspect resources on your system

Manifest Quest

- Understand the concept of a Puppet manifest
- Construct and apply manifests to manage resources

"best practice" methods to creating, checking, applying your manifest:

1. Open or create a manifest with the .pp extension
2. Add or edit your Puppet code
3. Use the puppet parser tool to check for syntax errors (validate filename) (recommended)
4. Simulate your manifest using puppet apply --noop (recommended)
5. Enfore your manifest using puppet apply
6. Check to make sure everything is working correctly (recommended)

Variables Quest: 2h

- Learn how to make Puppet manifests more flexible using variables
- Learn how to interpolate variables in manifests
- Understand how facter facts can be used

Conditional Statements Quest: 1h

- Learn how to use conditional logic to make your manifests adaptable.
- Understand the syntax and function of the if, unless, case, and selector statements.

Ordering Quest: 1h

- Understand why some resources must be managed in a specific order.
- Use the before, require, notify, and subscribe metaparameters to effectively manage the order that Puppet applies resource declarations.

Classes Quest: 30m

- Understand what a class means in Puppet's Language
- Learn how to use a class definition
- Understand the difference between defining and declaring a class

Modules Quest: 45m

- Understand the purpose of Puppet modules
- Learn the basic structure and layout of modules
- Write and test a simple module

Puppet module tool (forge) Quest: 30m

- Confidently use the puppet module tool in association with Forge modules

Dec 23

Verified new VM for puppet testing, emmutl-qc-7d.ula.comcast.net.

- Configured in SSH Secure Shell

- Reviewed ULA CADA and iTRC records
- Locked red userid (as root)
- Use puppet to:
    - Add git and cmadmin userids
    - Use this guide to configure emmutl-qc-7d just like cmmutl-po-7p.
    - puppet conclusions:
        - The Stand Alone Architecture
        - Puppet agents run environment
        - Directory Environments vs. Config File Environments - Using directory environments
        - Enabling Directory Environments in Open Source Puppet

Look at http://projects.puppetlabs.com/projects/puppet/wiki/Strict_RPM_Package_Version_Management - May need to get to this level on each system.

https://wiki.io.comcast.net/display/IDEA/Puppet+Tutorial+-+Delivery+a-la-cart - From Hitesh

https://magneto.sys.comcast.net/se_ops_archs/dev - Should be one for dev, test, and production. Check with Eric Sunberg

https://docs.puppetlabs.com/puppet/3.7/reference/ Puppet Manual

Admin Tasks

Setup New Users

New users will sent the our team their public ssh key per section SSH Public Key of the EM&M Git User Guide.

When setting up new ssh keys, Updates will be required in the gitolite-admin repo in the keydir directory and the conf/gitolite.conf file.

1. Be aware of the current crontab schedule for cmputl-ch2-4p:/git/bin/gitsys-failover-mirror.bsh. Time the following "git push" runs accordingly.
2. Update both the primary and fail-over servers for this repo per the following Updates for gitolite-admin repo section.
3. Verify the public keys have been added to the following directories on the EM&M Git System servers. You will need to check the fail-over server after a cmputl-ch2-4p:/git/bin/gitsys-failover-mirror.bsh script run. If the gitolote.conf/keys have not been add/updated in these directories, the user ssh keys will not function correctly:
    - emmutl-po-7p:/git/.gitolite/keydir
    - emmutl-po-7p:/git/.gitolite/conf/gitolite.conf
    - cmputl-ch2-4p:/git/.gitolite/keydir
    - cmputl-ch2-4p:/git/.gitolite/conf/gitolite.conf
4. Send an email back to the User and their manager based on the following example:

User Name,

You have been setup in the "general|devleads|developers" group of the EM&M Git System.
Reference the Access Management section of the Version Management Implementation document for more details.
Regards,

Fail-over System

The EM&M Git System consists of a "primary" and "fail-over" system. The fail-over system is designed to be updated automatically as changes occur on the primary system using Git clone and fetch commands.

The script gitsys-failover-mirror.bsh is run from the git userid crontab on the failover system. This keeps the load off of the primary system. This script is maintained in the git repo CFX-EMM-GITSystem. The git crontab entry on the fail-over server is also maintained in the CFX-EMM-GITSystem repo at app/fail-over-server/crontab-file.txt.

Please review the preamble in the gitsys-failover-mirror.bsh script, installed at fail-over:/home/git/bin, for daily Usage and Monitoring tasks.

A dependency of the gitsys-failover-mirror.bsh script is the ssh config file for each git .ssh directory on both the failover and primary servers. Copies of this config file for both servers are maintained in the CFX-EMM-GITSystem repo. They are named ssh_config-git in the repo and installed as ~/.ssh/config on both the primary and fail-over servers under the git userid. There values vary for each server so two copies are maintained.

The git user is setup as a gitolite controlled userid in the gitolite-admin repo so Git functionality can be used in the gitsys-backuprestore.bsh

Updates for gitolite-admin repo

You **MUST** update both the primary and fail-over servers with separate git push commands for any changes made to this repo. This is the EM&M Git System management repo. It is duplicated on both the primary and fail-over servers for fail-over functionality. Be sure to setup two remote settings for your local copy of this repo. **Verify your changes you are about to push with all who push for this repo.**

Git push capability is off by default on the fail-over server via the gitolite writable utility (ssh git@fail-over writable -h).
You must turn this on and then when updating the fail-over server with gitolite-admin repo pushes. The default text-string for turning off writable

for the fail-over server is in the following "writable @all off" command. The following commands are executed from your gitolite-admin repo for any updates to this repo when pushing them to the EM&M Git System servers:

```
$ git push                                  ---
Updates the primary server (default remote)
(output from git push ... )
$
$ ssh git@fail-over writable @all on    ---
Makes all repos writable on the fail-over
server
WARNING: This system is solely for the use
of authorized Comcast employees and
contractors.
$
$ git push failover                         ---
Git push to failover remote defined earlier
(output from git push ... )
$
$ ssh git@fail-over writable @all off "This
is the fail-over server. The git push option
is disabled."
WARNING: This system is solely for the use
of authorized Comcast employees and
contractors.
$
```

NOTE: When managing the gitolite-admin repo, you MUST push to both the primary and fail-over servers for any updates. Set up two remote entries in your gitolite-admin repo to do this.

Still working on full backup for daily, weekly, and monthly requirements. Following is an example script that may be useful.

- git-archive-all.sh

Access Check Email Example

```
From: Wallace, Andrew
Sent: Wednesday, September 24, 2014 8:07 AM
To: Kora, Varalakshmi
Cc: Galambos, Robert; Woodcock, Bruce (Contractor); Sell, Robert; Nair, Abhilash; Smith, Dan E; Schulz,
Mike; Hoopes, Eric; Gray, Eric (Contractor); Frazier, Chris
Subject: RE: GIT - ssh key

Lakshmi,

You access looks correct and is functioning as follows:

As a devlead, you have:
 • Creator access to devlead and developer tmp wild repos.
 • Read access to the gitolite-admin repo
 • Read/Write access to the testing repo
You have been setup by repo owners for the following repos:
 • Read/Write CFX-OBIEE
 • Read/Write rgalam200/CFX-Git-Training

NOTE: Repos in tmp and/or userid directories have not been finalized for EM&M production deployments.

Let me know if you have further questions.

Regards,

Andy Wallace
t+p        SE       APS
EM&M Configuration Management Portal
EM&M Git System
183 Inverness Drive
Suite 100 North 1-075
Englewood, CO 80112
720-267-2677
Andrew_Wallace@cable.comcast.com

From: Kora, Varalakshmi
Sent: Tuesday, September 23, 2014 3:49 PM
To: Wallace, Andrew
Cc: Galambos, Robert
Subject: GIT - ssh key

$ ssh git@emm-git info
WARNING: This system is solely for the use of authorized Comcast employees and contractors.
hello vkora200, this is git@emmutl-po-7p running gitolite3 (v3.6.1) on git 1.9.0

     C  CREATOR/\b(CFX|VIDEO|VOICE|DATA)\b\-[A-Za-z0-9]+\-?[A-Za-z0-9]+
     C  tmp/CREATOR/\b(CFX|VIDEO|VOICE|DATA)\b\-[A-Za-z0-9]+\-?[A-Za-z0-9]+\_[0-9]+\_[0-9]+
 R W    CFX-OBIEE
 R      gitolite-admin
 R W    rgalam200/CFX-Git-Training
 R W    rgalam200/VOICE-TaxRevenue
 R W    testing
```

Finalize DevLead Repo

Verify the Development lead has created a "develop" branch. This can be done by reviewing the EMM GitWeb.

1. You will/may need to initiate (prime the pump) for the "develop" main branch. Reference the "git branch" command for details. The "master" branch will be empty on a new repo and the "develop" branch may or may not be empty. Use "git clone" to bring the repo to your workstation so you can run the appropriate git commands to clean up the repo.
   Specifics:
    a. The command **git clone emm-git:[repo name]** will bring the entire repo to your local work area.
        i. **git branch -v** works only for branches you have checked out at least once locally (**git check [branch name]**). Use EMM GitWeb to determine available branch heads for your repo. Then **git check [branch name]** each head, which are branch heads.
        ii. If required, make sure the latest Development Deployment Branch, a Feature Branch, has been merged into the "develop" branch. It should be the the head of the "develop" branch:
            1. Have the "develop" branch checked out and execute commands similar to the following example:
                a. $ **git merge VIDEO-Apache-Spark_1_0_0_0** (VIDEO-Apache-Spark_1_0_0_0 is the name of a developer feature branch.)
                b. $ **git commit -m "Merged branch VIDEO-Apache-Spark_1_0_0_0."** (This is how the branch being merged in the the develop main branch is referenced in the future.)
            2. Verify the existing branches on EMM Git Repo System, are valid, removed any there are not. Valid branches include the Main Branches "develop" and "master" once established and any feature, release, or hotfix branches in progress. All others should be merged into one of the main branches and removed from the EMM

Git Repos System for the repo in question.
    a. To remove a temporary repo branch, execute commands similar to the following:
        i. $ Remove branch on local repo and run **git push origin :VIDEO-Apache-Spark_1_0_0_0**
b. If there is no "develop" branch for the EMM Git Repo system, the following command will establish it:
    i. Have the "develop" branch checked out on your local repo ($ **git checkout develop**) before running the following command.
    ii. $ **git push --set-upstream origin develop**
    iii. All initial files for this repo should be on this initial "develop" main branch.
c. If you need to change an existing git commit comment, it must be the head of a branch. **Note:** The comment is part of the commit and will create a new hash.
    i. The head commit of a branch, git checkout the branch and execute the following:
        1. $ **git commit --amend -m "Merge VIDEO-Apache-Spark_1_0_0_0 branch."**
    ii. **Modifying existing git history on the EMM Git System is prohibited.** It causes future collaboration issues that are best avoided. There are ways developers and other contributors to the EMM Git System can change git history in their own git repos before doing "git push" to the EMM Git System. Refer to Git Tools - Rewriting History for details.
d. Keep a list of everything you have done in this step so it can be included in the reply email back to the devlead.
e. Update the EMM Git Repo System with **git push**. Verify all updates made via EMM GitWeb.

2. The gitolite updates:
    a. Log on to the EM&M Git primary server directly as git, cd $REPO_BASE (default: cd ~/repositories)
    b. Move repo from CREATOR directory to main repository directory
    c. Verify the [repo].git/gl-creator file is intact with the CREATOR userid.
        i. Notes:
            1. Needed for OWNERS, WRITERS, READERS, MSLEADS, and MSDEVS role perms
            2. Reference the "rc" file ($HOME/.gitolite.rc) for details
    d. Verify the [repo].git/gl-perms file exists and contains the following:

```
OWNERS [creator's userid]
```

- **NOTE:** If creator is added as owner to gl-perms, the gl-creator can be a reference to the original repo creator.

    e. Verify the [repo].git/config file exists containing the following:

```
[core]
        repositoryformatversion = 0
        filemode = true
        bare = true
[gitweb]
        owner = [repo creator's First/last Name]
        category = [Appropriate Category value]
```

    f. Back on your gitolite-admin clone:
        i. Edit conf/gitolite.conf
        ii. Remove all occurrences of old-name. (If matched wild-repo (tmp or CREATOR dir, this step not needed)
        iii. Create new entry for repo definition. Use existing repo definition if gl-creator has other repos.
        iv. Then add, commit, and push as usual. **Be sure to merge and push for both origin (primary) and failover remote master branches.**
            1. NOTE: The fail-over server has "git push" disabled for all (@all) repos. Reference "ssh git@fail-over writable -h" for details. Procedure documented at Updates for gitolite admin repo.
    g. On your client system, checkout the new finalized repo and create a master branch:
        i. $ git clone emm-git:[repo]
        ii. $ git checkout develop
        iii. $ git checkout -b master
        iv. $ git push --set-upstream origin master (Note, there will be no release tag on this branch initially.)
        v. $ git remote -v show origin (Are both master and develop pushes are up to date?)
        vi. $ git remote -v show origin (Verify devlead branch "develop" merges and pushes with remote develop branch and that it's up to date.)
    h. Verify you can see the new repo via "ssh git@emm-git info".
    i. Add a repo description (ssh git@emm-git desc -h)
    j. Verify the following files on primary server for repo (Needed for gitweb and gitolite):
        i. config
        ii. description
        iii. gl-conf
        iv. gl-creator
        v. gl-perms
3. Send reply email back to devlead. Here is a template:

Devlead,

The EMM Git System module VOICE-CDV-GATEWAY-1503 has been initiated for EMM GitWeb and gitolite configurations.

Please review the Standards section of the Version Management Implementation document.

Let me know if you have any questions or concerns.

Regards,

Your Signature

The order of these steps is important; do not change order 

NOTE: Review related preamble in gitolite.conf and adjust this section for any variance before starting this procedure.

Git CM Build and Deployment Procedures

As build and deployment automation procedures are implemented, the need for this procedure should diminish and change considerably. It is anticipated that build and deployment automation will be created and tested by development before Release Branches are establish for EMM Git System repos.

Release Branches are Supporting Branches. They are created, maintained, and removed by the EMM CM Team. Refer to the EMM Branching and Merging Workflow for details.

Create Release Branch (Establish CM WorkArea and process QA Install)

1. Verify Development Deploy Branch Merged into current head of develop main branch.
    a. The summary of the JIRA CEMPCM project ticket consists of the [Change Record] and [Module ID] of the Development Deployment Delivery. The Module ID should be merged into the current head of the Git repo develop branch.
    b. Establish/Update CM Work Area
        i. This area is maintained on server cmputl-po-4p at /data/Release_elements/git. Logon to cmputl-po-4p and sudo -i -u git. cd to cmwa, a soft link established in /home/git.
        ii. mkdir and or cd [Change Record]
        iii. **git clone emm-git:[Module Name]** - (Module Name is first part of Module ID in JIRA ticket summary field.)
        iv. **git checkout develop**
        v. Verify CHANGELOG file exists and has been constructed per CHANGELOG section of EM&M Git Users Guide.
        vi. Review GitWeb (or git log) to Verify the Development Deploy Branch has been merged to head of the develop branch. If not, refer to "Finalize Devlead Repo" procedure.
2. Establish new release branch. Using Major (M) and minor (m) increments of Module ID in JIRA Ticket summary field.
    a. **git checkout -b release-[M.m]**
3. Process QA Build/Deployment

Process Developer Re-Work

1. Verify JIRA CEMPCM ticket replaced, Fixed by Clone, Clone incremented and assigned per JIRA EM&M Workflow.
2. Logon/get local to appropriate CM Work Area
    a. This area is maintained on server cmputl-po-4p at /data/Release_elements/git. Logon to cmputl-po-4p and sudo -i -u git. cd to cmwa, a soft link established in /home/git.
    b. **cd [Change Record]/[Module Name]**
3. Create Developer Deployment Branch from current Release Branch based on new JIRA Ticket Clone increment:
    a. **git checkout -b [Module Name]_M_m_d_t**
    b. **git merge release-M.m**
    c. **git push**
4. Update JIRA ticket with comment for checking out new Developer Deployment Branch.

The developer:

1. does the required re-work,
2. merges to the latest **develop** branch,
3. merges to the latest **master** branch
4. commits to the **develop** branch

When developer delivers Re-Work:

1. **git fetch origin**
2. Verify CHANGELOG file exists and has been constructed per CHANGELOG section of EM&M Git Users Guide.
3. Review GitWeb (or git log) to Verify the Development Deploy Branch has been merged to head of the develop branch.
4. Merge Development Deploy Branch, that was pushed to the EMM Git System by developer to the existing **release** branch
5. Removed current Development Deploy Branch after previous step.
6. Update existing release branch CHANGELOG entry:
    a. R for release branch
    b. Build number from maven for BUILD ID
7. Process QA build/deployment.

Complete Release Branch

Once QA Certification and Production Deployment and Verification occurs, document these steps.

Misc

Checkout a Single file

For latest production:

- git clone -n emm-git:[repo name] –b master --depth 1
- cd [repo name]
- git checkout master – [filename]

For latest development:

- git clone -n emm-git:[repo name] -b develop --depth 1
- cd [repo name]
- git checkout develop – [filename]

The “git clone” command depends on the settings in the remote repo .git/config file for the branch identification. Since the “git clone” initializes the .git/config for the new repo being created, the branch should be specified in the “git clone” for clarity and error checking.

The “-n” argument indicates no checkout of the branch HEAD is performed after the clone is complete.

The “git clone” --depth [number] option limits the size of the newly created by the number of Git snapshots for the identified branch. The SHA-1 hash for the number of parents back from the HEAD is identified and stored in .git/shallow. This limits the size of the new repo being created.

The EM&M Git System promotes and supports two main (or primary) branches, “develop” for latest development, and “master” for latest production. If it is required to get one file For any given repo in this system, it should be acquired as indicated above.

For any repo you are cloning from, the branches it supports should be known and understood. For the EM&M Git System, reference the Branching and Merging section of the EM&M Git User Guide for details.

Notes:

1. The “git clone” --[branch] argument can also take tags and detaches the HEAD at the commit in the resulting repository.
2. The “git clone”--depth argument can be used to go back to previous snapshots in the identified branch.
3. Even though the above two options are available, it is best to manage to the branch HEAD and reference earlier snapshots on a “as needed” basis.
4. The need for getting to “single files” in a given Git snapshot should be limited to patch deployments. Patch deployments should be avoided since that can contribute to variance in “deployed to” environments.
5. Remote tracking branches for any Git repo should be known by the user before executing “git fetch” or “git pull”. Execute “git branch -r” or reference .git/config of your repo for details.
6. Filename requires full path.

Sub-Modules in Git

The biggest contribution Git brings to the table is the ability to make branching and merging a part of the natural flow of development. When it comes to modules and sub-modules, the concept or goal of development code "Tightly bound and loosely coupled" should be considered and explored before implementing a solution for modules and sub-modules with Git.

There are two primary components that contribute to "Tightly bound and loosely coupled" they are:

- Divide and Conquer - Break a problem down to smaller components (or modules) that can be implemented more easily.
- Morph and Separate - when a solution grows too large, remove stable unchanging elements into utilities (other modules or libraries) that can be referenced by the original solution and other solutions.

When a module or group of modules grows too large for a given Git repo, it is expected that new Git repos will be created. This may simply be a new Git repo, or the build system will need to be deployed to from stable unchanging Git repos that deploys libraries for build systems.

The EMM Git Repository System has been setup with the above thoughts in mind. It is best to allow developers to develop. Two primary concepts used by developers are "Divide and Conquer" and "Morph and Separate". Multiple modules can eventually become part of any solution. Their definitions and relationships are managed by development managers and teams. Just as there are limits to how many modules can be loaded into memory, the size of the Git repo is limited (approximately 1 gigabyte). For the EMM Git Repository System, developers can manage a single module, or module sets within a Git repo given the limitation for size. Git repos can be related to each other by naming conventions and release documentation within the Git repo. Modules can be represented with many repos so that deployments can be repo baseline deployments.

Other thoughts on using Git to manage module and their sub-modules exist. here is a good reference for consideration pointed out to me by Jim Haun:
Git Submodules: Core Concept, Workflows And Tips
Here's another older reference to sub-modules and Git - Git Submodules, adding, using, removing, and updating

Test with sample index.html file in /opt/git.

[root@cmputl-ch2-4p git]# pwd
/opt/git

```
[root@cmputl-ch2-4p git]# cat index.html
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>

[root@cmputl-ch2-4p git]#
```

User and Admin Guide Maintenance

These guides are maintained in the Confluence CEMPCM Space. The Git Repo System Admin Guide is located under the **Administration** tab and the EMM Git User Guide is located under the **Auto Deploy** tab.

This is done to maintain backups and quick links for both guides from the primary and fail-overs server web pages.

In order to make them available for the EMM Git Repo System server web pages, the following is done:

1. **Export to PDF** from the Confluence page tools pulldown.
2. Save the PDF file in the Git repo CFX-EMM-GITSystem.
3. Copy to the primary and fail-over server directory /var/www/html.
4. On both the primary and fail-over servers update link in /var/www/index.html.
5. Copy updated /var/www/index.html to the Git repo CFX-EMM-GITSystem.
6. Remove previous copies of files (*.pdf, and index.html) from server and Git repo as required.

Verify Apache syntax:

```
[root@cmputl-ch2-4p conf.d]# /usr/sbin/httpd -S
VirtualHost configuration:
wildcard NameVirtualHosts and *default* servers:
*:80 is a NameVirtualHost
default server www.repository (/etc/httpd/conf.d/repository.conf:2)
port 80 namevhost www.repository (/etc/httpd/conf.d/repository.conf:2)
alias repository
Syntax OK
[root@cmputl-ch2-4p conf.d]#
```

Verify Gitolite.conf User Access

When you are logged on the server as git, you have the following gitolite access command to verify user access. This lets you know how Gitolite configuration is working for a given user after successful ssh authorization. Helps trouble-shoot access issues determining and ssh key issue or Gitolite configuration issue. For details enter the following:
[git@emmutl-po-7p ~]$ gitolite access -h

- Example:
  [git@emmutl-po-7p ~]$ gitolite access -s foo jdimme0431c W any
  legend:
  d => skipped deny rule due to ref unknown or 'any',
  r => skipped due to refex not matching,
  p => skipped due to perm (W, +, etc) not matching,
  D => explicitly denied,
  A => explicitly allowed,
  F => denied due to fallthru (no rules matched)

```
d gitolite.conf:156 - master = @developers @leaddevs @devleads # @genuser
d gitolite.conf:157 - develop = @developers # @genuser
A gitolite.conf:159 RW = @developers # @genuser

refs/.*
[git@emmutl-po-7p ~]$
```

welcome.conf adjustment in /etc/httpd/conf.d

```
[root@cmputl-ch2-4p conf.d]# pwd
/etc/httpd/conf.d
[root@cmputl-ch2-4p conf.d]# diff welcome.conf-orig welcome.conf
7,10c7,10
```

```
< <LocationMatch "^/+$">
< Options -Indexes
< ErrorDocument 403 /error/noindex.html
< </LocationMatch>
—
> #<LocationMatch "^/+$">
> # Options -Indexes
> # ErrorDocument 403 /error/noindex.html
> #</LocationMatch>
[root@cmputl-ch2-4p conf.d]#

[root@cmputl-ch2-4p conf.d]# service httpd restart
Stopping httpd: [ OK ]
Starting httpd: [ OK ]
[root@cmputl-ch2-4p conf.d]#
```

Errors Encountered

Errors that have been encountered during setup/testing activites are documented here for future support issues.

For new wild-repos, if you find the CREATOR value (/tmp/CREATOR/new repo and /CREATOR/new repo),
check the gl-creator value and let the user know how to use the command correctly.

The ssh key is working but unable to clone repo

If the following error is encountered:
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Check to see if the analyst's workstation has a variable GIT_SSH that is set to plink.exe. If it is reset it to null.
After setting to null open another git console and enter: echo $GIT_SSH.
If the GIT_SSH variable issue re-appears the GIT_SSH variable is probably a windows registry variable must be removed or you will not be able to checkout/clone.
To remove it permanently open a windows console and enter the following:
REG delete HKCU\Environment /F /V GIT_SSH
Now reboot and the variable should disappear.

"Denied by FallThru"

What the following directory and file when adding and removing ssh keys, they may not be updating correctly:

- git@server:/home/git.ssh/authorized_keys
- git@server;/home/git/.gitolite/keydir (List of *.pub ssh public key files)

You may need to add and remove things. This occurred when updating a ssh key for a given user from mixed case to all lower case in the name.

Also note the Emergencies section of the gitolite documentation. Need to use the "Lost Admin/key Access" section.

Related "Admin Task" section of this document -> Verify gitolite.conf User Access.

GitWeb projects.list Not Working Correctly

If the following errors are encountered: (Font color of error lines changed to red.)

```
awalla5075k@CO183LCETENG08 ~/git-server-admin/gitolite-admin/conf (master)
$ git push
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 501 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: FATAL: git config 'gitweb.owner' not allowed
remote: check GIT_CONFIG_KEYS in the rc file
To git@cemp-git:gitolite-admin
d44ba8d..cf6c002 master -> master

awalla5075k@CO183LCETENG08 ~/git-server-admin/gitolite-admin/conf (master)
$
```

Make sure your .gitolite.rc contains: (logon to emm-git and sudo to git)

GIT_CONFIG_KEYS => '.*',
GITWEB_PROJECTS_LIST => '/home/git/projects.list',

Install python Setup Tools

This was done is support of installing Gitosis testing on cmputl-ch2-4p. Decided to go with Gitolite to better support GitFlow. Left the Python tools on chputl-ch2-4p. Currently not install on emmutl-po-7p.

$ install python-setuptools

[root@cmputl-ch2-4p ~]# yum install python-setuptools

...

[root@cmputl-ch2-4p ~]#

Apache service:

[root@cmputl-ch2-4p git]# service httpd [restart|stop|start]

Branching Model

The branch module followed for successful branching and merging tasks is based on A Successful Git Branching Model by Vincent Driessen. The complete approach followed is drafted in the EM&M Git Users Guide.

Useful Links

Admin's Choice
Gitolite Troubleshooting Checklist
Gitolite Emergencies
Hosting Git Repositories
Git Software
Git Operations
Git Cheat Sheet by Jan Krueger.
Git Cheat Sheet by Zack Rusin.
Regular Expressions
Linux Command Line
Linux Standards Base
Wikipedia
HTML Character Entities
Commit Often, Perfect Later, Publish Once: Git Best Practices

Supporting Documents

Name	Size	Creator	Creation Date	Comment
PNG File SC-VMsetup.PNG	67 kB	Andrew Wallace	Jun 27, 2014 11:37	
PNG File SC-addnewVM.PNG	62 kB	Andrew Wallace	Jun 27, 2014 11:43	
HTML File RE VM Builds for Request #297555-68...	85 kB	Andrew Wallace	Jul 01, 2014 13:37	Issues setting up emmutl-po-7p.

File uploaded successfully