**Dream Dinners Linux Server Developer Guide**

**General Strategy**

Account privileges

Developers are given non-root accounts but are listed in the wheel group so they can use the sudo command when elevated permissions are required. If the nature of task requires it the developer can run “sudo -s” and supply their password at which point they are essentially logged in as root. It is safest to only use sudo when necessary.

SE Linux

SELinux is a powerful security tool but requires time and expertise to use and maintain. It is currently disabled.

Groups

In addition to belonging to the wheel group developers are added to the webdev group. This group is assigned to the files and folders of the Dream Dinners eCommerce Web Application.

Access

Only ports 443 and 80 are exposed and only on web servers. All other access requires a VPN Connection. See the VPN access guide for more information.

**Safety**

It goes without saying that extreme caution should be exercised when executing commands on the live servers. Here are some tips for staying safe:

* Use color to distinguish the importance of the server. For example, when using Putty each saved server connection can set the background color of the console. You could use red for Live Web and DB servers, purple for Replication and CHORS servers and green for test servers.
* Proceed slowly on the Live servers and verify assumptions prior to executing scripts: run ‘pwd’ to be sure you are in the directory you are expecting to be in.
* If possible do dry runs of any script making large scale changes. On approach is to have a script create SQL commands rather than altering the database directly. This way you can examine the SQL to be sure it is doing what you expected.
* Changes to the database can also be wrapped in the a transaction if the affected tables use the InnoDB engine. This way if something fails the entire transaction fails and you are not left with half updated data.
* Add commands to your .bash\_profile file that erase your command line history. This way if someone impersonates you they cannot back up through your history and execute sensitive commands. This can also be done for the mysql client.
  + TBD: Provide instructions

**Topography**

WE currently manage 6 CentOS 7-based Linux Servers

**Armor Dallas/Fort Worth**

DREA02VMW01 or W01 or Web Front End

* Main processes:
  + Apache – main eCommerce web service
  + Cron- runs many critical – shell and PHP scripts

DREA02VMD02 or D02 or Live

* Main processes:
  + Mariadb – main live eCommerce database

**Armor Chicago (was Phoenix)**

DREA02VMD01 or D01 or Rep

* Main processes
  + Mariadb – replication server
  + Cron – scripts for monitoring and backing up rep server
    - Hourly backups

CHORS (Cloud-based Home Office Reporting Server)

Main processes:

* Mariadb – redundant replication server and host of dreamdigest database. This database is a simplified version of the main ecommerce database and is an intermediary between Dream Dinners eCommerce and Salesforce.

**Microsoft Azure**

Sandbox

* Main processes – test server
  + Apache
  + Mariadb

Branch

* Main processes – test server
  + Apache
  + Mariadb

**Common Tasks**

**Deploying Code**

There are 2 methods

1. Utilize shell scripts found at the application root
2. Move file manually using SCP or SFTP

**Running Maintenance Scripts**

The are usually PHP scripts written for a specific and often one-time maintenance task. Scripts are copied (usually to the WFE) and executed from the command line.

Examples are

* Unloading a gift card
* Correcting data in the database
* Mass un-enrolling or enrolling guests into a marketing program
* Running an ad hoc report

**Altering Database**

There are several methods:

1. Logging into mariadb from the l;inux command line and executing commands there.
2. Running an sql script as input to the mysql command line tool (“mysql -udd\_root -p dreamsite < somescript.sql”
3. Running DDL as a query using Navicat or another database GUI tool.

**Maintaining Replication**

Replication is based on statement-based binary logging. That is, the queries themselves, including DDL, are what is logged by the master and read by the slave.

Common Scenarios

1. Re-establish replication after a database restore (note this is a complex subject – please consult more detailed explanations before attempting)
   1. A simple method when both the live and rep servers are rebuilt with the same backup is to initialize binary logging. Essentially starting replication from zero. (It is still necessary to run “Show Master Status” on the live server and set the coordinates on the replication server accordingly.
   2. If the Live server is stopped in mid process and the rep server is rebuilt it will be necessary to get the coordinates from the master and update the slave. Binary logs should be flushed so that the backup file dumped from the master is consistent with the logs.
2. Replication is stopped on an error
   1. Often a DDL command can sneak into the logs for a table that is listed as ignored. There are other harmless situations that can cause replication to stop on an error. In these situations You can follow this sequence:
      1. On the slave run the mysql client and issue these commands
         1. Show Slave Status\G - The results will allow you to confirm replication state and see the error
         2. Stop Slave; - this stops both the io and sql threads if either is running
         3. SET GLOBAL sql\_slave\_skip\_counter = 1; - this will cause the slave to skip the offending line.
         4. Start Slave;
         5. Then run Show Slave Status\G to see if replication started up. If not repeat steps 3 through 5.

**Initial Configuration**

This is beyond the scope of this document. See the separate Dream Dinners Linux Server Configuration Guide.

**Onboarding new Developer**

See the Configuration Guide

**Freeing Space**

You can use the following command to find large files that you may be able to delete:

find / -type f -size +100000k -exec ls -lh {} \; | awk '{ print $9 ": " $5 }'

The first “/” is the location to begin the search. The size parameter can be changed as needed. It’s best to run as sudo to cover the complete disk.

**Important Locations**

/etc/crontab – specify cron jobs here; alternatively you can place shell scripts in /etc/cron.hourly or /etc/cron.daily

/etc/cron.custom – a good place to locate shell scripts executed by virtue of a listing in /etc/crontab

/etc/my.cnf – mariadb config file [Note: the data directory may be in a non-standard location. Confirm the location by viewing my.cnf and look for the ‘datadir’ directive.

/etc/php.ini

/etc/httpd/conf – main httpd.conf is located here

/etc/httpd/conf.d – auxiallary conf files

/var/log/httpd – error and access files for apache

/DreamSite on the WFE (or DreamReports or DreamDigest on CHORS) – The web application root.

**Useful Commands**

**Services**

* Restarting
  + systemctl restart httpd
  + systemctl restart mariadb
* Stopping
  + Systemctl stop httpd
  + systemctl stop mariadb
* Etc.
  + Use start or status in place of stop

**Building the database from a gzip file** (often needed due to tight drive space)

gzip -dc < /DreamSite/www/dd.sql.gz | mysql -usomeuser -p dreamsite

**Building database from sql file**

Mysql -uroot -p dreamsite < dd.sql

**Dumping a database to a gzipped backup sql file**

mysqldump --routines --single-transaction --quick -usomeuser -p --host=localhost dreamsite | gzip > /home/carls/dd\_full.sql.gz

**List active Internet connections**

netstat -tulpn

**Set the machine clock**

date -s "28 AUG 2012 11:21:00"

**Follow a log file in realtime**

tail --lines=100 -f /var/log/messages

tail -f <file> | grep pattern location

**Find file larger than a number of bytes**

find / -type f -size +100000k -exec ls -lh {} \; | awk '{ print $9 ": " $5 }'

**Blocking IPs**

[‎11/‎23/‎2015 10:28 AM] Josh Thayer:

If you ever need to block and IP like that, it's really easy.

Edit this file: /etc/rc.d/init.d/rc.firewall

Find this section, and copy one of the DROP lines, and replace the IP:

# these guys are a real hassle...  so no server for them!

$IPT -A INPUT -s 162.213.31.65 -j DROP #jerk running login scripts

$IPT -A INPUT -s 91.200.12.81 -j DROP

Then do a: /etc/rc.d/init.d/rc.firewall restart

Validate the changes with: iptables -L