Note:

d1= dispatch1.make-a-store.com
d2=dispatch2.make-a-store.com

Install process:

When the order is made and it is inserted in make-a-store.com , a call is made to either d1 or d2 so it installs the new client. Currently it picks randomly which server to be the master. Divinity Free installs are installed on only one server. Paid ones are installed on the two, even if they only choose one cloud.

Initial Part - Preparation of data and adding install to db.
Code - bare_slave::install()
NK: This is run on d1 or d2, right?
This runs on the server on which the install is getting installed. So yes it can be d1 or d2.
- A record in the installs table (internal database on cloud-db1.make-a-store.com) is created for the install. It adds all the info for the install there. Database name, FS name etc..
- Each site is being inserted in the install_sites table. Also the ssl keys paths and info are stored here.
- In install_site_ips are inserted the IP's for each site.

NK: Where do we store the available IPs on each box. What is the process when we run out? We must choose movable IPs from Softlayer, right?
The available IPs are stored in the master plugin on make-a-store.com in the divinity database. In the table server_ips.
When we run out we must order more IPs . We can track the amount we have and have some small notifier to notify us that we need more.
Ips routed to specific VLAN (the one d1/d2 are on) must be ordered from softlayer.
- We choose a codebase for the install. Each install shares a codebase with several others. Info for those codebases can be found in the shared_code_groups table. It will choose the group with smallest amount of sites.

NK: What is the max # of installs per codebase we are currently configured to?
There is no solid max. The table shared_code_groups have a column - limit. That is the limit.
- A queue record is being added which is for the the other part of the install.

Actual Part - The core installation.
Code - slave::install_v2()
NK: Where does this run? On d1 and d2 ?
This runs on the installed master only. (either d1 or d2 depending on the chosen master from above)
- A subdir is created in /usr/home . The dir name will be the first several letters of the first hostname of the install.

- The filesystem is being created. Not all dirs are symlinked to the codebase. In /heavymetal theme and snippets are not symlinked. In /html/mas_assets - theme dir. Everything else is symlinked with the codebase.
- Config file is being populated and saved to /heavymetal/config.php.
- Database and db user/pass are created for the install.
- Mut is being run to process all the sql updates and tmp dir creations. This is the place where 99% of the installation errors happen. If an install fails it is quite possible the problem is in the sql updates.

NK: How do we debug this. If an install fails, how do I find the error message as to why?

If install fails we first check the queue record in the queue table (on cloud1db) for error. If no error is supplied there, we must create a snippet that must run what is in the queue record and debug the stdout and stderr.

NK: Can you create this snippet please and provide instructions?

Sure.sn
/home/internal/heavymetal/snippets/rerun_queue.php
You just need to text edit it. Set the $id to be equal to the queue id in the table and run it... Dont forget to redirect the stderr and stdout to a logfile for later checks.

- The install directory is synced with all the servers.
- Additional configurations are being done - create crons for the servers other than the master , add values to the vars relate to memcache and sphinx.
- An FTP directory structure is created in /heavymeal/tmp/ftp. Most of the directories there are mounted with nullfs or also known as bind mount . The info on those mounts is in the table startup_mounts. There is a cron that regulary checks for new mounts or deleted. It is called cron_mount.

NK: When customers call and say their FTP permissions aren't workin. What is the fix? Chown ftp:ftp on the dir in question? Or is there a snippet to fix a client?

If its a new bug - debug and find the issue.. But most of the times a simple chown will fix it. Also check if the cron_mounts is running or run manualy if there is some error there.

NK: Do you think we need a snippet we can run on d1 or d2 (depending on the master) to reset a user? e.g. reset_permissions www_bioex1 - this will globally reset all permissions for that user and check / fix if the cron is not running?

Can be usefull I guess in some rare situations

If any of the above steps fail the whole install fails and must start from the start.

NK: Does it roll back, undo what it did, etc?

Nope.

NK: OK, can you make a snippet then to clean it? Maybe something like: clean_install id, where id is the installs.id from the internal db on cloud1db?

I think there will be no need for a snippet. Will register a shutdown function to check if the install failed and if it did, it will delete the junk...But I probably need to make smth to clean current junk.. We've got some trash data still do clean up.

- Send mail to customer
- Rebuild nginx config.

NK: What happens if for some reason the nginx config gets broken. How do we force it to rebuild? Is there a backup of “last good nginx.conf” ?
If nginx config is broken we will not load that config at all. We do config file checks before we restart nginx with the new config. If it fails nginx stays with the old config. Forcing nginx rebulding is done by running /home/admin/rebuild_nginx_config on both d1 and d2.

Install done.

Important Crons : ***These are all run on d1 and d2 and part of internal database?***
Yes. These run on both.
- cron_installs - This cron runs all the crons for the installs in the installs table.
- cron_mount - It mouns/demounts the clients ftp directories and set permisions
- cron_slave_queue - A cron for internal use . Extension of the queue cron used by the server so each knows his queue records.
- cron_site_accounting - Parsing of web server logs and sending statistical data to us
- cron_slave_sync - Syncs the theme and theme assets for each install.
- sphinx - Rebuild sphinx config and reindex.
- web_server_config - Check for changes in the installs site settings and rebuild web server config when needed.

If any of these lock, it would be bad, right? DO we have something in place to detect?
Yes. It wont be good. Yes we can detect if some of these go rotten.

Updating installs
If an install is to be updates , all installs that use the same shared code group must be updated too. in the installs table , each install has the shared_code_group_id field .
In order to update a group of installs we must execute
$params = array('shared_group_id'=>$group_id,’revision’=>’HEAD’);
slave::update_cloud_users(serialize($params));
from somewhere in the internal install..
We need to run this on d1 AND d2? Or only on the master server for that group_id (according to installs table). Is there a snippet for this already on d1 and d2? If so what is it?
We must run this only on the master server (either d1 or d2) for the specific group_id . It will sync with the other servers.
/home/internal/heavymetal/snippets - update_shared_code_installs.php . **It must be edited** with the correct shared_code_group_id and will be fine.. I personaly run this by having two screens. On one I do php update_shared_code_installs.php &> update.log , on the other tail -f update.log to see what is going on and later on recheck update.log for any issues.

The update process goes with alot of debug info so it will be best the stdout and stderr to be redirected to another file. That file should be reviewed after that for any install issues.
e.g. svn conflicts, sql errors, etc.
Are there any “sanity” checks that run after an update yet?

Svn conflics are for now fixed. After update is done it does svn st to see if there are conflics and auto fixies it. For other errors the contents of the update.log must be seen.
After an update rudolf files are deleted , catalog cache files , files in local/tmp. cache table is truncated, apc cache is deleted too.

Updating specific files
If only few files are to be updated they must be updated on all the servers.

Specific snippet updating.
Copy+Paste the snippet content in open mysql console and add a record for the snippet inside database_updates table for the specific install.
So for each client we are updating, we must insert into database_updates the revision for each database effected.
We insert the filename of the sql snippet and plugin_id for each database we are changing.

Servers Setup.
All freeBSD . Each running nginx, php-fpm, proftpd , memcache, sphinx.
For now dispatch2 is running mysql server used for replication.
Nginx is used as load balancer between the php handlers and for serving static content.
In theory we can have as many php handlers as we wish.
NK: Why can't we add as many php dispatchers as we want too? What are our bottlenecks? The dispatcher only serves static files and hands off to the handler right? Does the handler send back the HTML or does the dispatcher?
We can also have as many dispatchers as we want too. It is not correct to say php dispatcher. The dispatcher is the web server. The handler is the php server. Currently we have both installed on d1 and d2 . By saying dispatcher I mean the web server. The connection goes as follows - Client <-> Dispatcher <-> Handler. For now the bottlenecks are mostly CPU related. Nginx config file is generated by the web_server_config cron in the slave plugin.

Php-fpm is used for php handling on all servers.

Adding new dispatcher server(web server).
Install all the software - nginx ….Copy their config files from any of the other servers.
Copy /home/internal from any of the other servers to the one and add a record for it in the handlers table.Add to crontab its lib/cron/cron.php and there is a new handler.
NK: Add to crontab on the new dispatcher you mean, right?
Yes

When adding a new dispatcher all php-fpm config files on all handlers must be updated so they allow connections from the new dispatcher internal ip. The 10.0..... IP.
NK: WHere are these configs located and can you give an example?
What about IP allocation? What would we update in the master or slave plugins so it knows to start using the new dispatcher?

/usr/local/etc/php-fpm.conf
On the master (make-a-store.com divinity database)- New record into the servers table. Add the available ips of the new server into server_ips table.
On the slave (cloud1db internal database) - Insert new record into the handlers table.

Adding new handler(php).
Install php-fpm with matching version of some of the other servers , proftpd.

Why do handlers need FTP if the dispatchers are the ones doing the static files. Isnt' that were the clients are FTPing to? Dispatchers not handlers?
This is an issue that we need to address later on. The templates are used by the handlers(php) and the static files are used by the dispatchers(nginx). For now we have both installed on 1 server but if we decide to split them up at some time we need to have probably 2 ftps or think of some other mechanic here.

Copy the config files from the other servers. Always install APC! Also copy its configuration from some of the other servers.
Please give a list of config files and the php ports?
php.ini , exrensions.ini or any other auto loaded .ini file used by php. Those can by easy checked with phpinfo()
The ports depend on the extensions used. phpinfo() will give info on that.

For both cases - look what is inside /etc/sysctl.conf and copy the vars that seem right for the dispatcher or handler server.
If it's the same hardware and freebsd version, we can copy exact?
Yes most of the times... but I would rather check each variable and what it does before copying it. For example - net.inet.carp.preempt=1 . Its related to CARP . If the machine doesnt have carp it will be pointless to copy it but it also wont create any issues... I guess for sanity reasons , best to check what is being copied.

Adding a new cloud server.
Cloud servers are temporary php handlers. They can be also web servers but their I/O is very bad and will not bring alot of improvement.
Their purpose is to manage the spikes in the load since Divinity is very CPU consuming and be removed shortly after the spike.
For now all the cloud servers are Debians. They are brought up from a saved image and will be configured and ready for work as they go live.
When a server goes up . Its php-fpm config file must be checked for allowed ips. Those values must match the IPs of the web servers that will request php handling.
Can we by default just allow all 10.x on our vlan?
I am not sure. It will be alot better for that . Here is what is said for that config param
allowed_clients - Comma separated list of ipv4 addresses of FastCGI clients that allowed to connect. Equivalent to FCGI_WEB_SERVER_ADDRS environment in original php.fcgi (5.2.2+) Makes sense only with AF_INET listening . Must test with smth like 10.0.0.0/8 to see if will

work.

In /home/admin there is a script called cloud_manager. It is a small python app that manages all the things realted to temporary cloud servers.
This is done on dispatch1?
Yes.
Add a new server - cloud_manager --action=add --ip=$prive_server_ip --hostname=$server_hostname
This adds a new server in the handlers table.
$private_server_ip would be the main IP associated w/ the cloud instance (debian) and $server_hostname would be the site we are adding the cloud to? What would $server_hostname match against in installs table? Can you give an example?
Is this script just inserting a record in the database? What actually configures nginx and all that stuff on the cloud instance?
$server_hostname is the hostname of the server we are added . ex - dispatch1.make-a-store.com
ex. cloud_manager --action=add --ip=10.0.4.1 --hostname=tmp-cloud.make-a-store.com
This will just add that tmp cloud at the list of available servers for us to use.
There is no nginx on the cloud instance. There is just php handler.
How can we give weights to the clouds. e.g. let's say Tea wants to do a big traffic spike. And we want all their traffic to go to the new debian clouds, and NOT to d1 and d2, what is the procedure?
It is not automated by anything for now.. It is done manualy by editing nginx config , For sure smth must be done for manipulating this.

Remove server - cloud_manager --action=remove --ip=$private_server_ip
Removes all records related to this server and rebuilds web server config.

Add install to a server - cloud_manager --action=add_install --id=$install_id --ip=$private_server_ip
Add additional php handler for install . Syncs the install files and shared code. Rebuild web server config. Also adds crons for that install on the new server.

So this must always be done after the action=add ? And the $install_id is the installs.id for the site we want this cloud to use?

Yes.

What if we wanted a debian cloud to handle multiple sites. How would that work. Please give examples.

cloud_manager --action=add_install --id=40 --ip=10.0.4.1
(It will do some syncs and db inserts and will rebuild nginx)
cloud_manager --action=add_install --id=41 --ip=10.0.4.1

(The same)
And now we have installs 40 and 41 on the new tmp cloud.

Remove install from a server - cloud_manager --action=remove_install --id=$install_id --ip=$private_server_ip
Removes install from cloud server. Rebuild web server config.

Failsafe.

Failsafe of the dispatchers.
Since we use round robin for dispatchers , if one goes down , 50% of all requests will die.
In order for that not to happen we must quickly migrate the IPs of the server which is down to the server that is alive. This is done by using IPs routed to specific VLAN. So the machine that first adds the IP is the one that has it.
The whole process has few keypoints.

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/carp.html
This is being used for an indicator when a server dies.
Each server that will use this must have the carp kernel module enabled.

Carp config from dispatch1 -
```
cloned_interfaces="carp1 carp2"
ifconfig_carp1="vhid 1 pass d1d2 advskew 10 172.16.102.21/24"
ifconfig_carp2="vhid 2 pass d1d2 advskew 20 172.16.102.22/24"
```

Carp config from dispatch2 -
```
cloned_interfaces="carp1 carp2"
ifconfig_carp1="vhid 1 pass d1d2 advskew 20 172.16.102.21/24"
ifconfig_carp2="vhid 2 pass d1d2 advskew 10 172.16.102.21/24""
```

In basic human words this does this - Creates carp1 interface on both machines. On dispatch1 it is with advskew 10 and on dispatch2 it is with advskew 20 which means - If dispatch1 is up that interface will be up there and will be backup on dispatch2. When dispatch1 goes down, dispatch2 will take its role. When dispatch1 comes up, it will become master and dispatch2 will release become backup. The same logic is for carp2 only there dispatch2 is the master and dispatch1 is the backup.
These ips - 172.16.102.21/24 and 172.16.102.21/24 are there simply because we must have some IPs there. The actual IPs are in another place.
When carp2 goes up on dispatch1 an event is being triggered by the devd daemon.
http://nixdoc.net/man-pages/FreeBSD/devd.8.html
It is event for an interface going UP or DOWN.
This is what is appened to /etc/devd.conf

```
notify 10 {
        match "system"          "IFNET";
        match "subsystem"       "(carp1|carp2)";
        match "type"            "(LINK_UP|LINK_DOWN)";
        action "/home/admin/ip_manager --carp=$subsystem --action=$type";
};
```

It tells the devd daemon when carp1 or carp2 go UP or DOWN /home/admin/ip_manager to be executed.
All the magic is there. Several checks are made there and depending on the event it manupulates the IPs the server have. That script uses a config file in which we specify the IPs for each interface. The config file is here /usr/local/etc/mas_ip_manager.cfg.

```
[carp1]
interface = igb1
ips : 208.43.48.2
        208.43.48.3
        208.43.48.4
[carp2]
interface = igb1
ips : 208.43.48.5
        208.43.48.6
```

So when dispatch2(carp2) goes down - on dispatch1 an event will be received and will be run /home/admin/ip_manager --carp=carp2 --action=LINK_DOWN . The script will read from the config file this:

```
[carp2]
interface = igb1
ips : 208.43.48.5
        208.43.48.6
```

and will add to interface igb1 208.43.48.5 and 208.43.48.6 IPs.
When dispatch2 goes UP home/admin/ip_manager --carp=carp2 --action=LINK_UP will be executed and that same two IPs will be removed.

NK: This is a bit over my head ;) What could go wrong here and what would be done to fix it. Let's say a box is down but we notice IPs aren't moved? What do we do when we get new IPs on d1 and d2? This is for both public and private IPs?

If we have d1 down and d2 must take its ips, before they are taken the script does a ping to those ips. If the ping is succesful , the IPs are not taken. As u recall we had few cases when we could ping a server but no other service was responding.

If d1 is down but no ips are moved we check ifconfig. And we check carp1(d1 carp interface). If its state is active master and the ips are not moved that means the ips are still allocated on d1.

The easiest thing to do is ifconfig carp1 down; ifconfig carp1 up; But again if those ips are pingable they wont be taken.

Private ips we just add to rc.conf .
If we get ips that are routed to vlan(movable ips). We add them /usr/local/etc/mas_ip_manager.cfg.
Example . we get 211.212.213.1 and 211.212.213.2 . We decide we want 211.212.213.1 to be on d1 and 211.212.213.2 to be on d2.
We open the config file /usr/local/etc/mas_ip_manager.cfg and under carp1 section we add 211.212.213.1 and under carp2 section we add 211.212.213.2.
The config param interface = igb1 tells us which nic to use.. We then do on d1 ifconfig carp1 down; ifconfig carp1 up; and on d2 ifconfig carp2 down; ifconfig carp2 up;

Installs that use the floating ips must have their column dynamic_ips set to 1. All the new installs that are paid have this.

Failsafe of the handlers.
It is done by nginx . It checks for certain amount of errors during amount of time and stops sending requests to that handler for some amount of time.

Failsafe of DB.
Still nothing.

So if cloud1-db goes down, and it will be down more than 10 min, we would need to manually move the internal IP on cloud1 to dispatch2 (the slave master)? Perhaps it would be bettter to have a snippet on dispatch1 we can run like:

move_database --destination master
or
move_database --destination slave

This would update everyone's config.php .. so we don't have to move the IP address.

Or is it better to move the iP?

I also have thought alot on this. I think the editing of config.php is the better solution.

NK2: Can you create a snippet for this that we would run on d1 and d2. Just in case it ever happens, we'd be ready in advance.
Yes. I think it will be best , that script to be on both d1 and d2 and when run to change all installs' config files on both machines so it wont be needed to run it on d1 and d2 , just on one of those to change globally.

Can you talk about what kind of crons we have to make sure sites run, automated clearing of rudolf metadata if a site doesn't load, etc?

This is done by a cron in the internal plugin - cron_slave_widgets . It checks all sites and if a site is dead it mails us and deletes rudolf , catalg cache and vars cache...

Change install hostname.
- change the hostname in install_sites table in the internal database . There needs to be changed hostname,secure_hostname and ssl info(if has its own)
- if there is some ip change , change install_site_ips records for that install_site
- if we manage their dns settings , go and change all the dns records related to the last hostname with the new. Example if the client was something.com and he wants to be newthing.com , change only the domain record. Mostly here will be domains and subdomains changes.
- rebuild nginx config /home/admin/rebuild_nginx_config

Rsync setup   --- NEEDS UPDATING
Divinity is syncing files between servers with rsync. This is very important for images and files syncing. We need to have on each server rsync in daemon mode. After that all the files in /usr/local/etc/rsyncd.* must be copied and added to the new server. Also ownership must be modified to match the ones on d1 or d2.
There are two rsyncd*.pwd files . One is called rsynced_www.pwd and the other rsynced_root.pwd. The www is used when rsynced is triggered from a web page.. - image creation etc. The root one for crons or other scripts called form the root user.

Change ssl.
Put the files in /home/$install/certs on all servers . In install_sites for that install insert two fields ssl_pem and ssl_key , which are the absolute paths to those files and rebuild nginx on all servers.

Downgrade paid to free.
1. Change symlinks of the install to point a shared code that is for free installs. The shared code group that we are moving the install to must be the same revision which is the paid one.
2.Edit install_site_ips for the site and remove the uneeded ips. We must leve just one since we wont be serving the site from many servers anymore.
3.Change install license to match a free one.
4.Unmount all bind mounts related to that install(ftp mostly)
5.Stop ftp acccess. I just rename the username to username_del
6.Edit dns records for that install so only one is left for the site and will point the master dispatcher. Delete all else.
7.Rebuild nginx.l

Custom rewrite rules for install
Add all the rewrite rules in an array and serialize it. Then put it into the custom_rewrites column in installs table.

Adding a new site.

```
$site  = new site();
$site->hostname="http://wholesale.teacollection.com";
$site->secure_hostname="https://wholesale.teacollection.com";
$site->rel_url = "/";
$site->secure_rel_url="/";
$site->insert();
```

After that add a record to install_sites with the hostnames info and after that add the ips that are used for that site in install_site_ips.

Change codebase.
The current codebase and the new must be the same revisions. Loop trough all the files and directories in the install and if there is a symlink for example to /home/shared_codes/code_share_paid_1/heavymetal/lib , change it to /home/shared_codes/code_share_paid_$num/heavymetal/lib.
Also change in the installs table the shared_code_group_id to the new group id

Sign up a customer to their own codebase
Create a codebase with limit=1 to only allow one site to be there. In the master plugin where the master handler is being choosed , choose the server which is the master for a codebase. Choosing this for now is made with if's . Smth like if($customer_email=='smth@smth.com') - > use server_id 6
example code from the master

```
if($this->params['email']=='tea@make-a-store.com') {
          $servers->orderby('id asc'); //this will give u server_id 6 which is d1
 }
 else {
          $servers->orderby('RAND()');
 }
```

At the slave - d1 in the install method in bare_slave here is what is used agian

```
if($this->data['email']=='tea@make-a-store.com') {
          $install->shared_code_group_id = 29;
}
else {
          $install->shared_code_group_id =    slave::get_shared_code_dir($install,$this-
>handler_id);
}
```

Add a new codebase.
Make a snippet and run create_new_shared_code_group($install->free,$handler_id)
$install->free - 0 if is paid or 1 if is for free. handler_id is the handler on which to install the shared code. It will add a queue record for this in queue table. Normaly it takes few to several mintues to checkout divinity