Prelaunch Checklist

A successful product launch sucks. It's full of pain, fires, and not having a second to catch your breath. Instead of enjoying the moment, more often than not, you find yourself dealing with burning servers and pain points you didn't realize existing until you had 1000 people loading your site at the same time.

This checklist gives you a fighting chance. Go through it before your launch. Everything might not apply to you, but even if you remove one hot spot— you've given yourself a better chance of a flawless launch.

Even the most experienced pilots use a checklist before taking off. We should too.

In 2011, the Twitpic team launched Heello. While the product ultimately flopped, we launched with an enormous amount of traffic. Every major tech news site wrote about us. There were 1 million new signups the first week, and something like 500,000 on the first day.

That's impressive, but what's even more impressive— we didn't go down once during that time. In fact, we kept shipping new code even under the enormous amount of load.

What was our secret? We spent days before launch preparing for the traffic. Battle hardening the servers and getting ready to scale. We were ready.

AT YOUR DOMAIN REGISTRAR

```
[ ] Change your DNS Name servers to point to Route53
[ ] Add DNS Load Balancing for Multiple HAProxy Load Balancers
[ ] Lower the TTL for your primary subdomains (60-300s)
[ ] Lower the TTL for your SOA DNS Record (60-600s)
```

ON EVERY SERVER

```
[ ] Create user accounts for yourself and your team (w/ sudo)
[ ] Create a unix account for deployment
[ ] Add your SSH Public Keys
[ ] Disable SSH Password Authentication
[ ] Disable SSH Root Login
[ ] Disable Reverse DNS Lookup on SSH login
[ ] Use non-standard SSH Port (optional)
[ ] Disable non-intranet SSH, use proxy server or VPN to connect (optional)
[ ] Enabled ufw firewall and block unused ports, on EC2 just use security groups (optional)
```

Prelaunch Checklist

ON EVERY SERVER (CONTINUED)

[ ] Configure /etc/resolv.conf to have lower timeouts
[ ] Install nscd (or setup a dnsmasq server and point to it)
[ ] Install fail2ban
[ ] Install htop, sysstat, strace, ack-grep, vim, and other favorite debugging tools

LOAD BALANCER

[ ] Configure and secure HAProxy Web UI
[ ] Verify HAProxy Health Checks are working
[ ] Verify HAProxy starts when the server is rebooted
[ ] Check HAProxy timeout settings for your server pool
[ ] Test HAProxy server failover with keepalived
[ ] Increase net.core.somaxconn to a 999999
[ ] Set net.ipv4.tcp_tw_recycle to 1
[ ] Set net.ipv4.ip_local_port_range to 10240 62000
[ ] Increase maximum number of open file descriptors
[ ] Verify that sysctl changes persist across reboots
[ ] Verify that ulimit changes persist across reboots

APP SERVER

[ ] Verify you are using the latest version of PHP
[ ] Verify your SSL certificates, as well as certificate chain are on the app servers
[ ] Validate your SSL Settings (https://www.ssllabs.com/ssltest/analyze.html)
[ ] Remove PHP Sohosin if installed by your package manager
[ ] Verify your Nginx worker_processes is equal to the number of CPU cores
[ ] Verify Nginx starts when the server is rebooted

Prelaunch Checklist

APP SERVER (CONTINUED)

[ ] Verify PHP-FPM starts when the server is rebooted
[ ] Verify you have your pm.max_children value in PHP-FPM pool config
[ ] Setup Logrotate to rotate Nginx Log Files
[ ] Setup Logrotate to rotate PHP-FPM Slow Log
[ ] Verify APC or Zend OpCache is configured in php.ini
[ ] If using APC, verify apc.stat_ctime=1 is set
[ ] Mount your filesystem with noatime option
[ ] Enabled tmpfs for your /tmp directory
[ ] Increase net.core.somaxconn to 999999
[ ] Increase maximum number of open file descriptors
[ ] Verify sysctl and ulimit changes persist across reboot
[ ] Verify php.ini settings- expose_php=Off, memory_limit=-1, max_execution_time=5, display_errors=Off
[ ] Verify session handing uses Memcached, Redis, or Cookies
[ ] Install igbinary PECL extension
[ ] Install memcached PECL extension
[ ] Install redis PECL extension
[ ] Setup capistrano or chef for quick deploys

DATABASE SERVER

[ ] Verify you are running the latest version of Percona MySQL
[ ] Secure MySQL user permissions
[ ] Verify MySQL slaves are using local network IP addresses (not public internet!)
[ ] Verify MySQL is only listening on local network IP address (not public internet!)
[ ] Put your MySQL slaves behind your HAProxy load balancer
[ ] Setup pt-kill to kill orphaned queries, verify it runs after reboot
[ ] Verify max_connections is set to > # app servers * # php-fpm workers.
[ ] Verify innodb_flush_method=O_DIRECT
[ ] Verify innodb_buffer_pool_size is 90% of your server memory

Prelaunch Checklist

DATABASE SERVER (CONTINUED)

[ ] Verify query_cache is disabled (query_cache_size=0, query_cache_type=0)
[ ] On Master Database, verify innodb_flush_log_at_trx_commit=1
[ ] On Slave Database, verify innodb_flush_log_at_trx_commit=2
[ ] Verify any significant tables are using InnoDB as their storage engine
[ ] Change Linux I/O Scheduler to deadline or noop
[ ] Change vm.swapiness sysctl setting
[ ] Increase max number of open file descriptors
[ ] Disable file system access times (noatime)
[ ] Verify your MySQL database partition is using the XFS filesystem
[ ] Verify MySQL starts on server reboot
[ ] Verify ulimit and sysctl changes persist across reboots
[ ] Verify MySQL Backups are being taken
[ ] Kill a slave server; Make sure your app handles it gracefully.
[ ] Practice creating a new slave server from a MySQL Backup
[ ] Practice a MySQL Master Failover and Slave Promotion
[ ] Verify MySQL is configured with NUMA settings if applicable
[ ] Verify you are running the latest linux kernel
[ ] Bonus: Make sure your MySQL backups are being synced to Amazon S3

REDIS SERVER

[ ] Enable Append-Only File (AOF)
[ ] Tune maxmemory setting
[ ] Verify your using an appropriate maxmemory-policy for your use case
[ ] Change Linux I/O Scheduler to deadline or noop
[ ] Tune vm.swapiness sysctl setting
[ ] Increase number of open file descriptors
[ ] Disable filesystem access times (noatime)
[ ] Make sure Redis is only listening on your internal IP (not public internet!)
[ ] Verify Redis starts on server reboot

Prelaunch Checklist

REDIS SERVER (CONTINUED)

[ ] Restart Redis and see how long it takes to reload your data– big datasets can take minutes.
[ ] Bonus: Setup a Backup for your dump.rdb file

RESQUE & WORKER SERVER

[ ] Verify Resque Web UI is running and locked down with a UN/PW
[ ] Verify your Resque workers are running and have the latest code
[ ] Verify that your Resque workers start on reboot
[ ] Verify that capistrano deploys to your worker server and reloads the workers

IN YOUR CODE

[ ] Make sure you're using a dog pile lock for slow-database queries that you're storing in Memcached
[ ] Memcached all the things! Load as much stuff into your cache as humanely possible
[ ] Move any long running work, API calls, or anything that can be done later into an async queue
[ ] Verify your complex background jobs have MAX_ATTEMPTS logic built in
[ ] Verify your ORM or Database Library can split queries between Master/Slave MySQL Servers
[ ] Verify Capistrano rollbacks work as expected
[ ] Change double quotes (“) to single quotes (‘), they're faster. JOKE!! Don't do this. It's not true.
[ ] Use a tool like xhprof, xdebug, or just microtime() to profile potential hotspots in your code
[ ] Use a tool like airbrake, new relic, or exceptional to log (and alert) exceptions that happen in production
[ ] Setup a tool like pingdom, copperegg, or pager duty to alert you when your site is slow or down