Development Workflow and Resources (Out of date)

This is the gateway documentation to get all your development-related questions answered.

Before you begin development...

As a best practice (and so that you have the most up-to-date code on your local machine), make sure to update your local repos first thing everyday. To accomplish that, in your terminal, type:
cd ~/Documents/new-projects/postmedia-infrastructure/
and then, once you have postmedia-infrastructure in your path, type:
scripts/git-pull

TL; DR. *

Follow the link : ^

Beginning development

The entire development process - from entering a ticket into the sprint to starting work on it and getting it through production - has been detailed at the following link: Deployment Pipeline.
Below are links to resources which will aid in the process.

Resources needed for your local environment

- LDAP
- Setting up local Virtual Machines (VMs)
- Tools available within VMs
- Connecting to local Redis/MySQL database
- Git branching strategy
- PHP Coding Standards/Unit tests
- Debugging PHP code
- Common gotchas when running VMs
- Accessing VMs locally

Resources for Remote Dev environment

- How do I create snapshots?
- Where do I see my recently built snapshot?

Getting stuff through Staging(QA/UAT) to Production

- How do I build an artifact (for Stage servers)?
- Accessing Websites on Dev/Stage
- Grafana/Kibana Metrics and Tutorial

If the above sections did not answer your questions...

It is very likely that your question has already been addressed before. Please check out the link below:

- How-tos / Frequently Asked Questions (FAQs) about Virtual Machines
- Frequently asked Questions (FAQs) about the deployment pipeline

If all else fails...

Please feel free to post a message in the #operations channel on Slack.

Note: Try and give complete detail about the issue you are facing. For e.g.: Add a snapshot of the error you are seeing on the terminal, add info about the VM you are trying to run etc.)

*

Local

Git

Fix merge conflicts locally (assuming merging YOUR_BRANCH_NAME into qa branch):
git fetch origin
git checkout qa
git merge --no-ff YOUR_BRANCH_NAME
...
(Fix the merge conflicts)
...
git commit -a
...
(Press Esc, then enter :wq, then press Enter)
...
git push
Revert and remove the last commit from the commit history in the current branch:
git reset HEAD^ --hard
git push -f
Undo a merge if you haven't committed the changes yet:
git reset --merge

Dev environment (deprecated)

Restart NGINX:
sudo brew services restart nginx
Restart PHP locally:
brew services restart php71
Restart PHP in VM, after vagrant ssh:
systemctl restart php7.2-fpm
Restart PHP-FPM:
pkill php-fpm; php-fpm -y /usr/local/etc/php/7.1/php-fpm.d/www.conf
Restart MySQL:
brew services restart mysql
Clear WordPress cache and transients:
wp transient delete --all; wp cache flush

Etc

Show hidden files:
defaults write com.apple.finder AppleShowAllFiles YES

VM

Note: These commands can only be run after running:
cd ~/Documents/new-projects/postmedia-infrastructure/vagrant/
communities.local
vagrant ssh

Cache

Clear Redis cache:
redis-cli flushall

Clear nginx caches:
sudo rm -rf /var/cache/nginx/nexus_microcache
sudo rm -rf /var/cache/nginx/thewhig_microcache
sudo rm -rf /var/cache/nginx/thesudburystar_microcache
sudo rm -rf /var/cache/nginx/edmontonexaminer_microcache
sudo rm -rf /var/cache/nginx/kingstonthisweek_microcache
Clear WordPress transients/object caches:
REDIS_DB=0 wp transient delete --all --url=http://nexus.local --path=/opt/
wordpress --debug=false
REDIS_DB=0 wp cache flush --url=http://nexus.local --path=/opt/wordpress --
debug=false
REDIS_DB=0 wp transient delete --all --url=http://thewhig.local --path=/opt/
wordpress --debug=false
REDIS_DB=0 wp cache flush --url=http://thewhig.local --path=/opt/wordpress --
debug=false
REDIS_DB=0 wp transient delete --all --url=http://thesudburystar.local --path=/
opt/wordpress --debug=false
REDIS_DB=0 wp cache flush --url=http://thesudburystar.local --path=/opt/
wordpress --debug=false
REDIS_DB=0 wp transient delete --all --url=http://edmontonexaminer.local --
path=/opt/wordpress --debug=false
REDIS_DB=0 wp cache flush --url=http://edmontonexaminer.local --path=/opt/
wordpress --debug=false
REDIS_DB=0 wp transient delete --all --url=http://kingstonthisweek.local --path=/
opt/wordpress --debug=false
REDIS_DB=0 wp cache flush --url=http://kingstonthisweek.local --path=/opt/
wordpress --debug=false

Etc

Check error log:
tail -100 /var/log/php/php-fpm-error.log
Clear error log:
echo '' > /var/log/php/php-fpm-error.log
Making terminal functions to make life easier

VM

To enter vagrant up on ~/Documents/new-projects/postmedia-infrastructure/

vagrant/communities.local/ could be annoying. I wrote shell functions for this. However, if you used Vagrant Manager, you wouldn't need this functions.

```
vvv() {
	dir=$(pwd)
	cd ~/Documents/new-projects/postmedia-infrastructure/vagrant/communities.local
	vagrant up
	cd ${dir}
}

xxx() {
	dir=$(pwd)
	cd ~/Documents/new-projects/postmedia-infrastructure/vagrant/communities.local
	vagrant halt
	cd ${dir}
}
```

Put this on your ~/.bash_profile or ~/.zshrc (defends on which shell you are using. See Oh-My-Zsh). Now you can turn VM on and off by typing vvv and xxx on terminal, on any directory.

Unit test

It's similar with PHPCS.

```
unittest() {

	 dir=$(pwd)

	 if [ -z "$1" ]; then

		set -- ${postmediaLibraryDir}

	 fi

	cd $1

	phpunit5 -c tests/phpunit.xml --report-useless-tests --disallow-test-output --verbose

	cd ${dir}

}
```

Syntax for VI

For vi editor to show the syntax highlight, put this on your ~/.vimrc, and run vim on

your terminal.

```
colo desert

syntax on
```