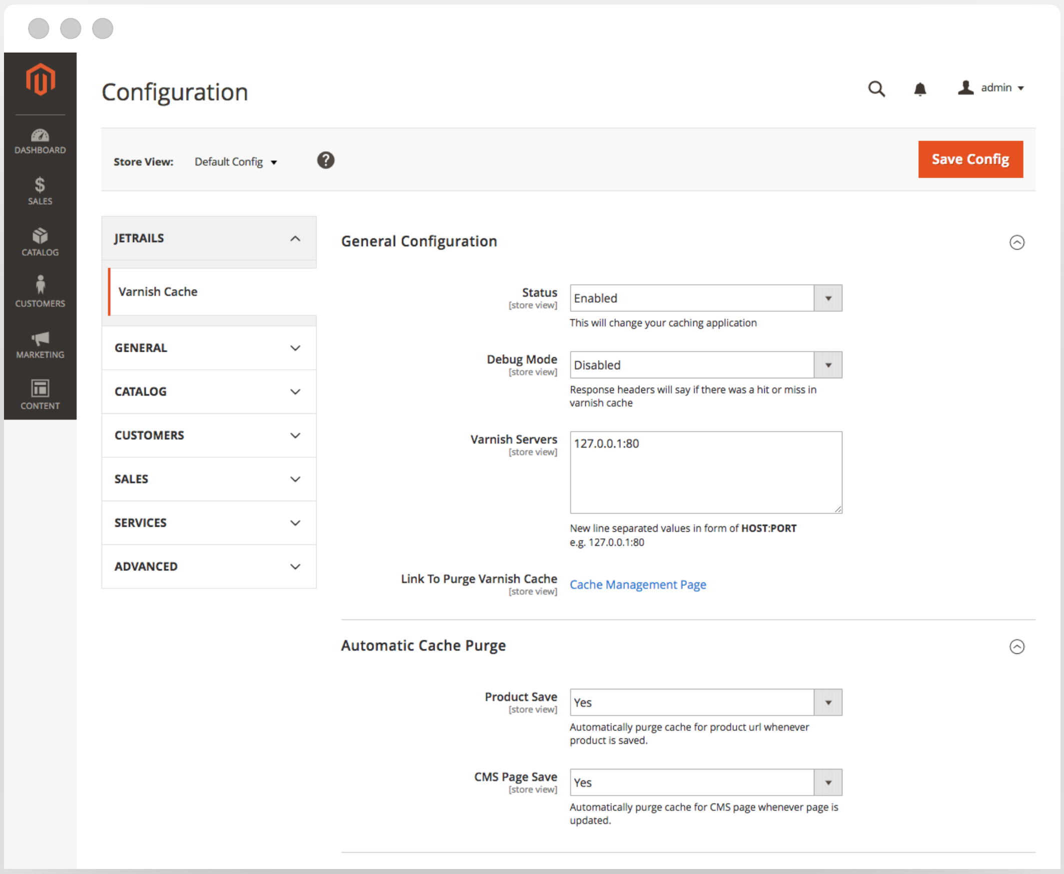
JetRails Extension For Varnish

User Guide



Configuration

Store View: Default Config

General Configuration

Status [store view]: Enabled
This will change your caching application

Debug Mode [store view]: Disabled
Response headers will say if there was a hit or miss in varnish cache

Varnish Servers [store view]: 127.0.0.1:80
New line separated values in form of **HOST:PORT**
e.g. 127.0.0.1:80

Link To Purge Varnish Cache [store view]: Cache Management Page

Automatic Cache Purge

Product Save [store view]: Yes
Automatically purge cache for product url whenever product is saved.

CMS Page Save [store view]: Yes
Automatically purge cache for CMS page whenever page is updated.

Version 1.1.2
Magento 2

Table of Contents

Disclaimer

Installation Process

Assumptions:

- This extension in archived form resides in */home/JetRails_Varnish-x.x.x.tgz*
- The Magento installation is located in */var/www/html/*

Instructions:

1. Copy the archived extension into the Magento installation folder.

```
cp /home/JetRails_Varnish-x.x.x.tgz /var/www/html/
```

2. Extract the contents of the archive into the Magento installation.

```
cd /var/www/html/
tar -xvf ./JetRails_Varnish-x.x.x.tgz
```

3. Complete installation by upgrading Magento.

```
php ./bin/magento setup:upgrade
```

4. Remove the extension archive file.

```
rm /var/www/html/JetRails_Varnish-x.x.x.tgz
```

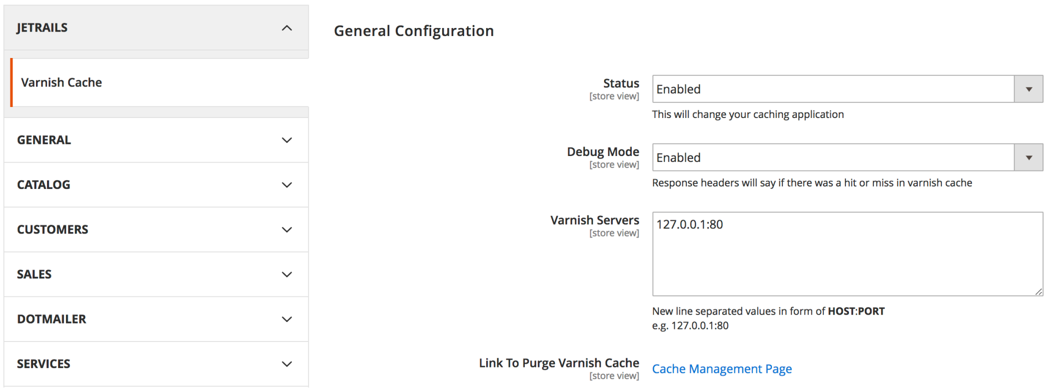
Setup Process

Enabling the Extension

First thing that you need to do is install the Varnish configuration file that came with this extension onto the server that is running Varnish. Once that is complete, log into the Magneto backend. Navigate to ***Stores > Settings > Configuration*** through the menu and click on the ***JETRAILS > Varnish Cache*** tab. Once there you will see a drop down setting for ***Status*** in the ***General Configuration*** panel. Setting this option to *Enabled* will change the Magento's caching application to *Varnish Cache* (thus enabling this extension), otherwise Magento's caching application will be set to *Built-in Cache* (effectively disabling this extension).

Configuring Varnish Servers

This extension supports configuring an array of Varnish servers. This means that you can configure this extension to communicate with multiple Varnish servers and purge cache on all of them at once. In the ***Varnish Servers*** text area in the ***General Configuration*** panel, you can enter as many varnish servers as you want by entering each one in it's own line in the format **HOST:PORT**. An example of such an entry can be **127.0.0.1:80**.



General Configuration

Status [store view]: Enabled
This will change your caching application

Debug Mode [store view]: Enabled
Response headers will say if there was a hit or miss in varnish cache

Varnish Servers [store view]: 127.0.0.1:80
New line separated values in form of **HOST:PORT**
e.g. 127.0.0.1:80

Link To Purge Varnish Cache [store view]: Cache Management Page

Using Debug Mode

When *Debug Mode* is enabled, an extra HTTP parameter will be sent in the response header of every page. This parameter will be identified as *JetRails-Debug-Cache* and it's value will always be either **HIT** or **MISS**. The values signify whether the visit to the page resulted in a HIT in Varnish cache or if it resulted in a MISS in Varnish cache.

Enabling/Disabling Debug Mode:

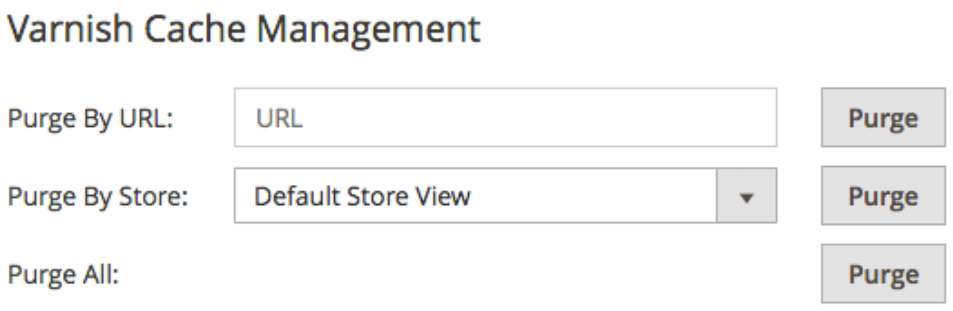
To enable or disable *Debug Mode*, log into your Magento Backend and navigate to ***Store > Settings > Configuration*** using the menu. Under the ***JETRAILS > Varnish Cache*** tab you will see the option ***General Configuration > Debug Mode***. Toggling this setting will either enable or disable *Debug Mode*.

Inspecting Varnish Cache Status While In Debug Mode:

Using Google Chrome, open the *Developer Tools* panel by clicking on the three dots on the top right of the browser window, using the menu, click on ***More Tools > Developer Tools***. Once the developer tools panel opens, visit the ***Network*** tab. Once you refresh the page, you will see every resource that loads for the current page you are viewing. The very first item will be the page itself. If you click on that item, you will see that more information is displayed. Looking under the ***Response Headers*** under the ***Headers*** tab, you will see that the *JetRails-Debug-Cache* parameter will be defined.

Purge Varnish Cache

Once this extension is configured correctly with Varnish, it gives you the ability to purge your Varnish cache. You can get to the ***Varnish Cache Management*** panel by navigating to ***System > Tools > Cache Management*** through your Magento backend. You should see the panel right above the ***Additional Cache Management*** panel. The three different actions that you can take are outlined below:



Purge By URL:
Whatever URL is passed will have to be matched exactly.

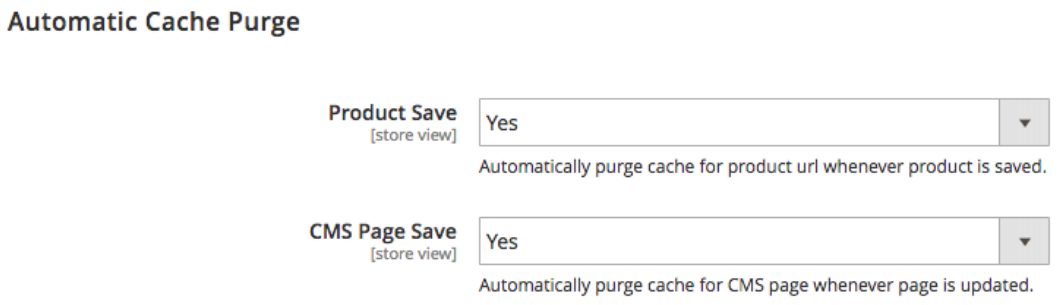
Purge By Store:
This option will be populated with all the different store views that your Magento store has. Each store view is associated with a domain name. This domain name will be used to purge all the urls that contain said domain name.

Purge All:
This option will purge every entry that Varnish cached.

Automatic Varnish Cache Purging

Purging Varnish cache can optionally be triggered whenever a product gets saved or a CMS page gets saved. This extension then purges the FPC within Varnish for all the URLs that are associated with the product or CMS page.

To toggle these setting, you can go to your Magento store's backend and navigate to ***Stores > Settings > Configuration*** through the menu. Next, make sure you are in the ***JetRails > Varnish Cache*** tab. Under the ***Automatic Cache Purge*** section you will see the ***Product Save*** and the ***CMS Page Save*** settings.



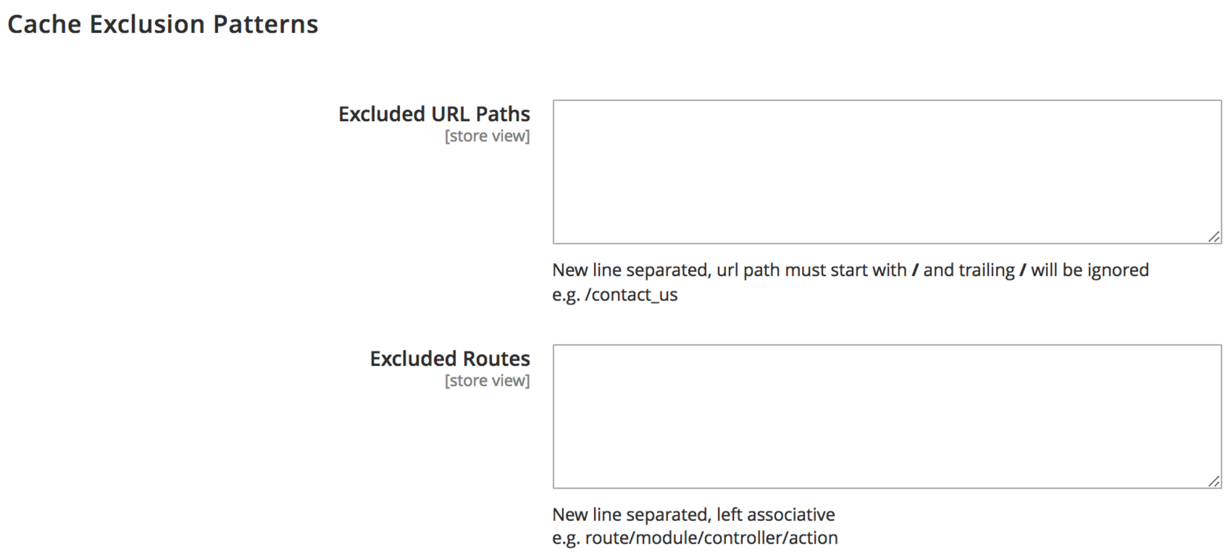
Automatic Cache Purge

Product Save [store view]: Yes
Automatically purge cache for product url whenever product is saved.

CMS Page Save [store view]: Yes
Automatically purge cache for CMS page whenever page is updated.

Excluding Paths & Routes

If you wish to exclude certain URLs and/or routes from being cached by Varnish, you can do so though this extension. After logging into your Magneto backend, navigate to ***Stores > Settings > Configuration***. Under the ***JETRAILS > Varnish Cache*** tab and the ***Cache Exclusion Patterns*** panel, you will find the ***Exclude URL Paths*** and ***Exclude Routes*** settings.


Cache Exclusion Patterns

Excluded URL Paths
[store view]

New line separated, url path must start with **/** and trailing **/** will be ignored
e.g. /contact_us

Excluded Routes
[store view]

New line separated, left associative
e.g. route/module/controller/action

Exclude URL Paths

Paths will point to a single page and will not contain a domain name, therefore they will always begin with a forward-slash (/). Entering in one path per line will enable you to exclude multiple paths from being cached by Varnish. You can confirm that this path exclusion rule is working by enabling debug mode for this extension and opening up the *Chrome Developer Tools* to analyzing the response headers. If this exclusion rule successfully worked, then you should see that *JetRails-Debug-Cache* will be set to *MISS* and *JetRails-No-Cache-Blame-Url* will contain the rule that you set previously.

Exclude Routes

Routes are defined within Magento and consist of the following format: **route/module/controller/action**. Excluding routes are left associative and therefore we are able to exclude a whole route by just specifying the route name. An example is if we enter admin then all requests to the admin route will be excluded from cache. Attaching more information to the route will more granularly help you exclude the routes that you want to exclude. You can confirm that this route exclusion rule is working by enabling debug mode for this extension and opening up the *Chrome Developer Tools* to analyzing the response headers. If this exclusion rule successfully worked, then you should see that JetRails-Debug-Cache will be set to *MISS* and *JetRails-No-Cache-Blame-Route* will contain the rule that you set previously.

CLI Tools

Magento binary can be executed by going to your Magento installation directory and running the following command. Note that <*command*> should be substituted with a command below: **php ./bin/magento <command>**

varnish:status
Shows if the extension is enabled. This extension is enabled when Magento's caching application is set to Varnish, it is disabled otherwise.

varnish:status:set <enable|disable>
When passing *enable*, Magento's caching application is set to *Varnish*. When *disable* is passed, the caching application is set to *Built-in Cache*.

varnish:purge:url <url>
Passing the URL will purge that url from Varnish Cache.

varnish:purge:store [store_id]
If a store id is not passed, then all store views are displayed alongside their ids. This information can be used to pass a desired store view id. If executed with a valid store view id, the store view's domain name will be used to purge Varnish cache.

varnish:purge:all
Purge all cached entries in Varnish.

Additional Support

For additional support or inquiries, please feel free to reach out to us by phone or email. Any found bugs or feature requests can also be communicated to us through email.

email:	**development@jetrails.com**
phone:	**1(888) 997-2457**

This extension is currently in beta and any feedback would be extremely valuable.