

Full Page Caching in Magento 2 for humans





Norbert Nagy

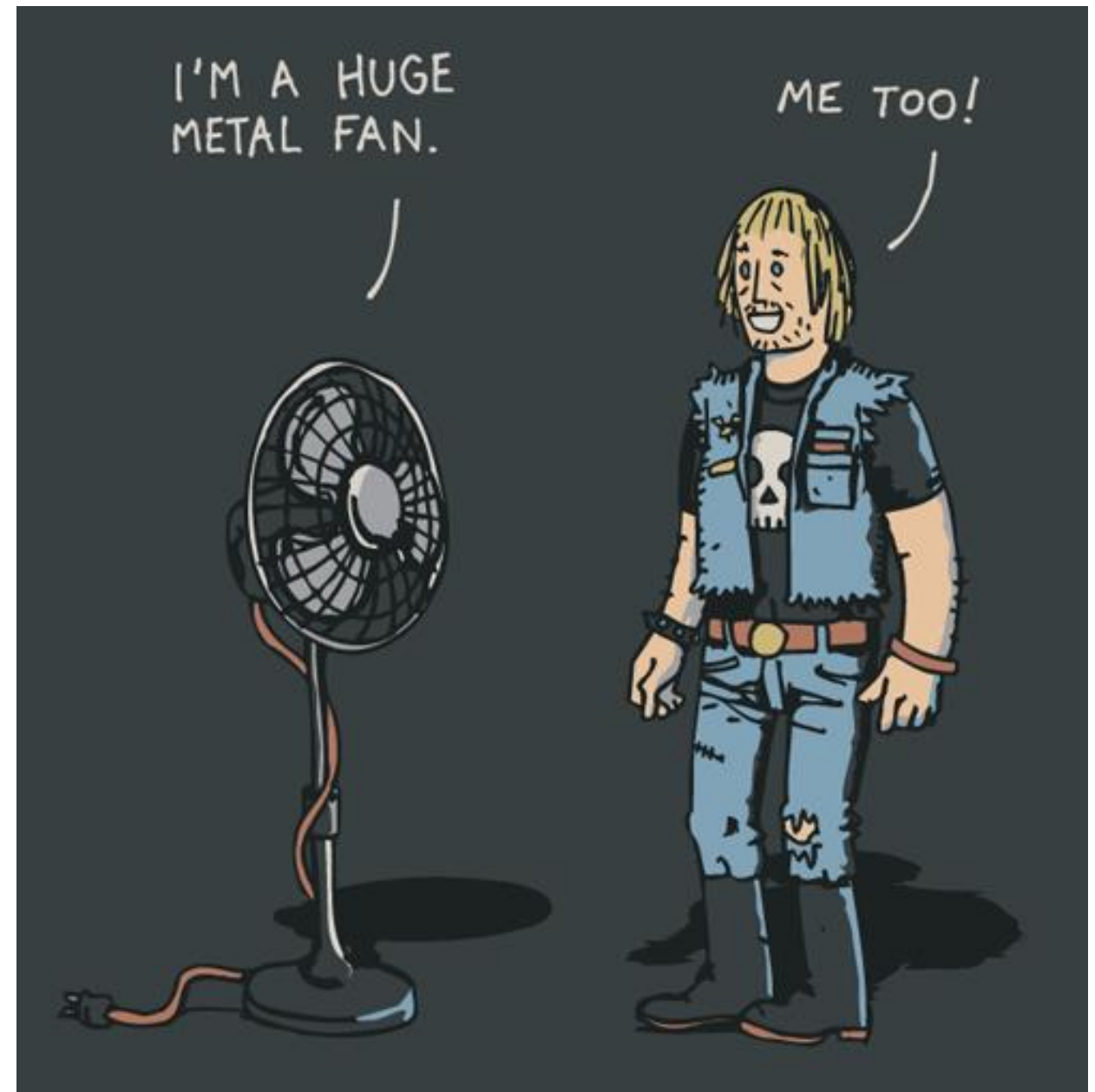
Technical Team Lead – Session Digital

Twitter: @nagno123



Why

Huge fan of Magento 2's page
cache implementation



Content

1. Caching in general
2. Page cache in general
3. Page cache setup in Magento 2
4. Magento 2's page caching system via Varnish
5. Most common issues and how to resolve them



What is cache

“In computing, a cache is a component that stores data so future requests for that data can be served faster” - Wikipedia



What is page cache

- Temporary storage for the result of successful requests
- Further requests for the same URL will be served from the cache
- Shared across all users of the site
- Additional metadata for further functionality



Page cache setup in Magento 2

Cache frontend

- High level gateway/interface to actual cache storage classes
- Provides access to load, save, remove functionality
- Set different cache backend for cache types
- You can add extra layers such as profiling, logging, ...



Page cache setup in Magento 2

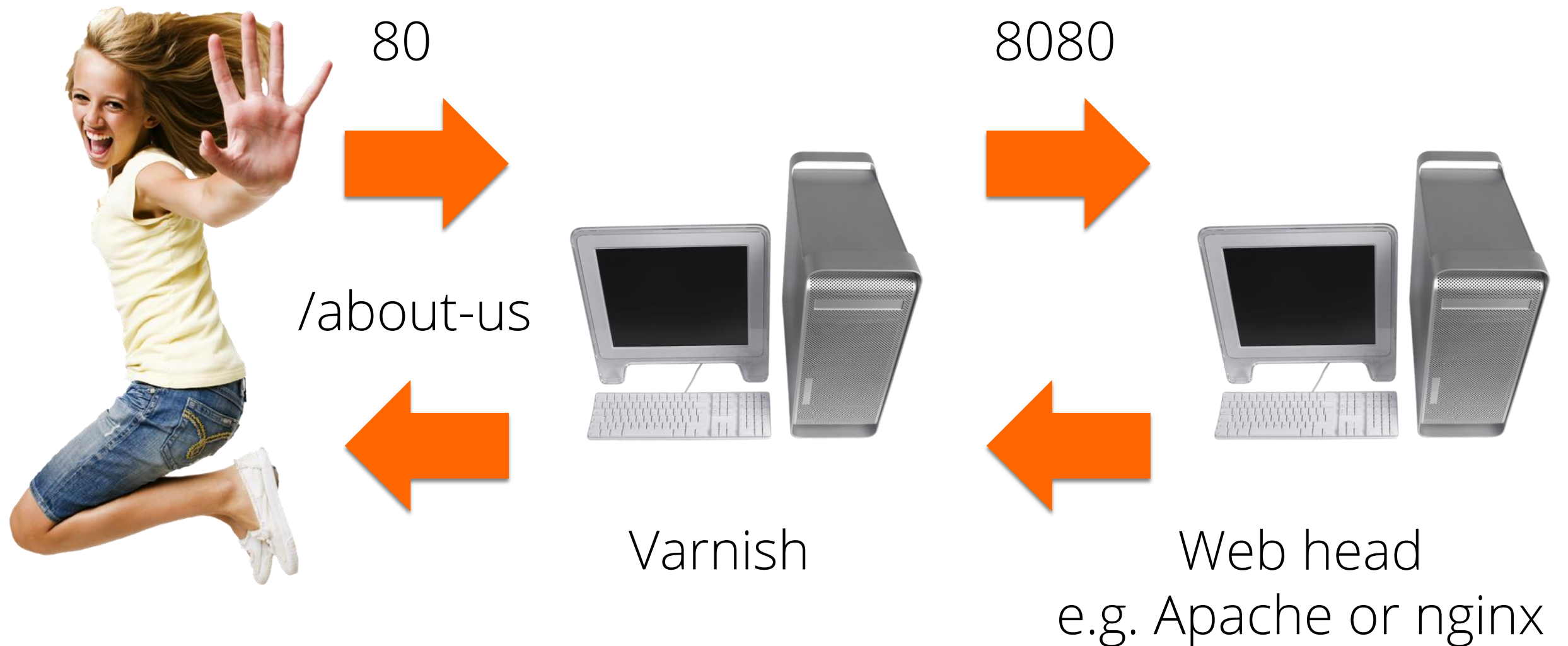
Cache backend

- Implementation details of the cache storage for the framework
- Adapter between cache frontend and actual cache driver classes (e.g. Zend_Cache_* classes)
- Varnish doesn't require a cache backend class!

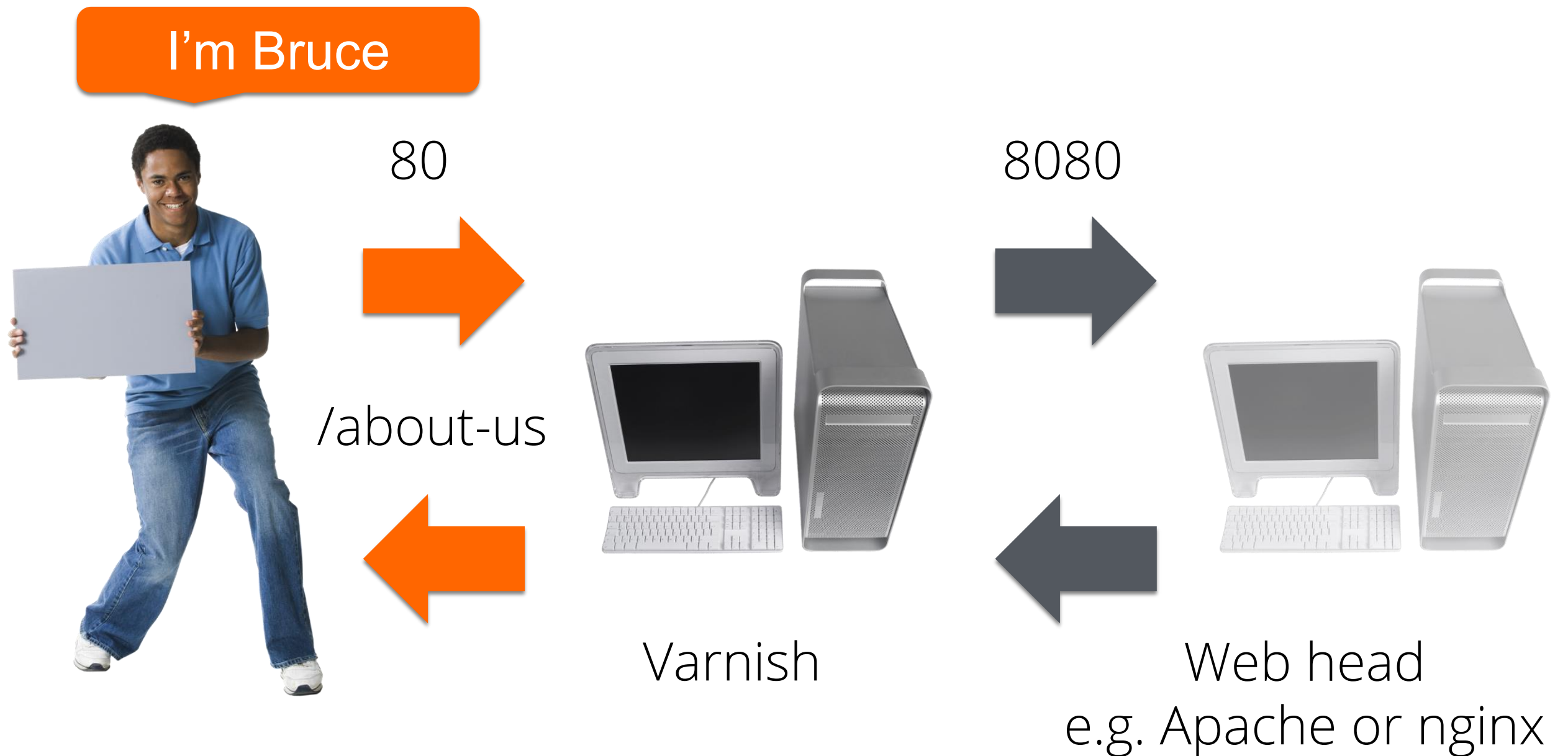


Request flow via varnish - MISS

I'm Veronica



Request flow via varnish - HIT



Privacy

- Hey! What about my privacy?
- Hole punching
- Javascript based loaders
- ESI
- HTTP headers!



Privacy – non cacheable – page

- Sensitive, nobody should cache it at all
- I really meant nobody, not even your browser!
- Layout XML file controls it in Magento 2
- The entire response will be non-cacheable!

```
<block class="Magento\Sales\Block\Order\View"  
name="sales.order.view" cacheable="false"  
after="sales.order.info.links" />
```



Privacy – private data – hole punching

- Specific to the customer
- No intermediary caching is allowed. This might be saved by your browser depending from your browser's implementation
- You can handle session data within the block
- Define your block as private

```
In \Magento\Checkout\Block\Cart:  
$this->_isScopePrivate = true;
```



Privacy – private data – hole punching

- core_layout_render_element event dispatched
- Cacheable, block not contains any TTL value in the layout file
- BLOCK placeholder generated (see below an example)
- Javascript loads the content via a separate private AJAX call

```
<!-- BLOCK notification_window --><!-- /BLOCK  
notification_window -->
```



Privacy – private data – JS + local storage

- E.g. Customer data, messages handled like this
- update_section AJAX call (unfortunately the request is using public caching headers and it makes me a sad panda)
- Sends the response in JSON
- Javascript to place the content to the correct container element



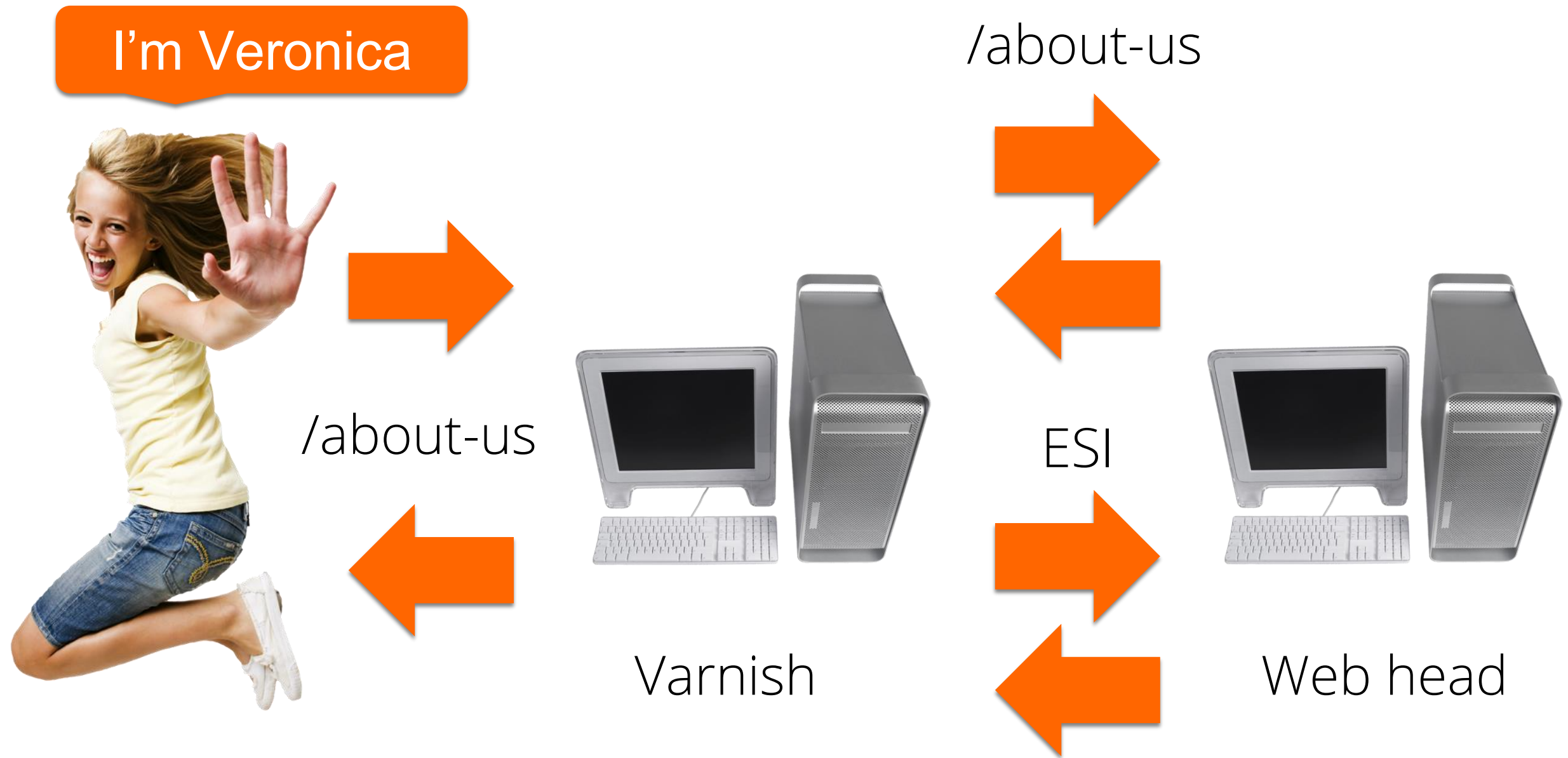
Privacy – private data – JS + local storage

- Response stored in your browser's local storage
- Invalidation via version cookie which changes during each POST request
- If your data is invalid or not available, it will be loaded from the server

mage-banners-cache-storage	{"data":{"items":{"salesrule":[],"catalogrule":[],"fixed":[]}}}
mage-banners-cache-timeout	"2016-04-03T15:25:47.908Z"
mage-cache-storage	{"directory-data":{"AD":{"name":"Andorra"},"AE":{"name":"United ,
mage-cache-storage-section-invalidation	{}
mage-cache-timeout	"2016-04-03T16:25:17.890Z"
mage-translation-file-version	{"version":"cc8aba8167908d7ea0b96b1adbd573383842e6c5"}
mage-translation-storage	[]

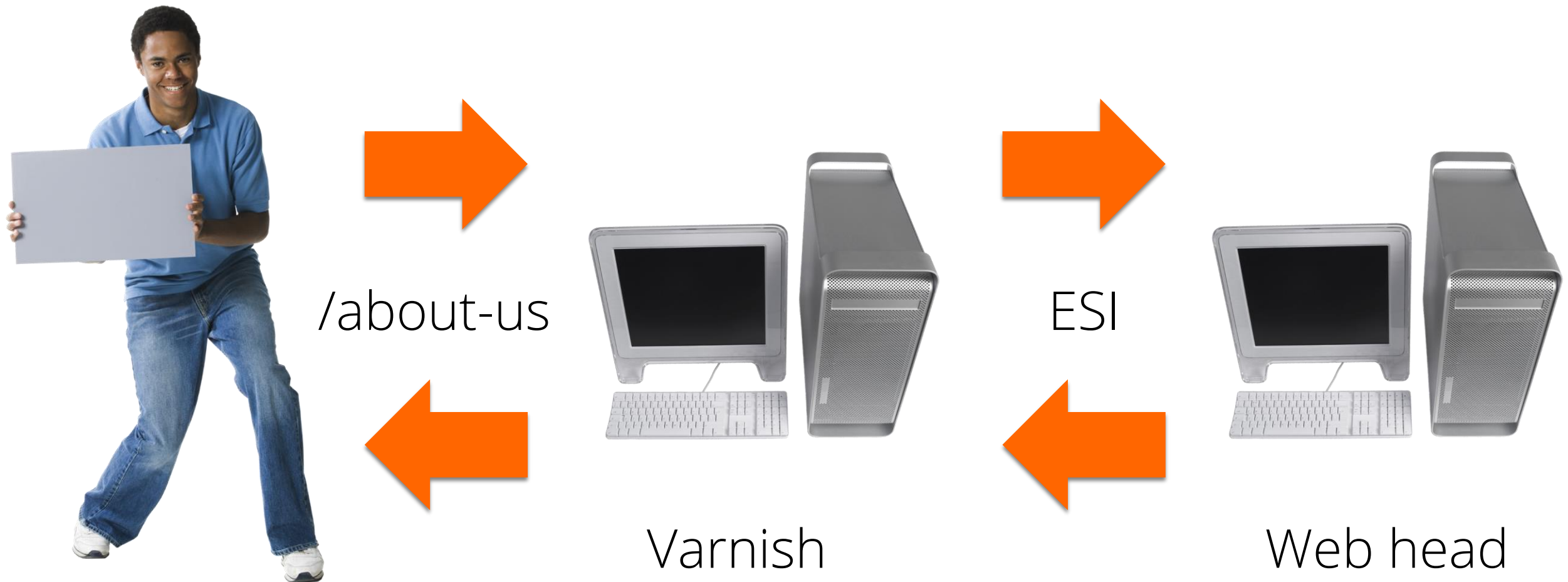


ESI Request flow via varnish - MISS



ESI Request flow via varnish - HIT

I'm Bruce



Privacy – ESI – kind of hole punching

- Magento 2 uses it for often changing public elements, which is allowed to cache
- Don't handle session data within the block => public!
- Think about performance before you add multiple ESI blocks
- Add TTL to your block definition in the layout file

```
<block class="Magento\Theme\Block\Html\Topmenu"  
name="catalog.topnav" template="html/topmenu.phtml"  
ttl="3600"/>
```



Privacy – ESI – kind of hole punching

- core_layout_render_element event dispatched
- Cacheable, block contains TTL value in the layout file
- ESI placeholder generated
- Varnish will automatically handle the ESI placeholder

```
<esi:include  
src="http://example.dev/page_cache/block/esi/blocks/[%22catalog.topnav%22]/handles/[%22default%22,%22cms_index_index%22,%22cms_page_view%22,%22cms_index_index_id_home%22]/" />
```



Using the IdentityInterface

- Implement
`\Magento\Framework\DataObject\IdentityInterface`
- Add `getIdentities()` to your class and report the identity of your objects
- Use `X-Magento-Tags` header for GET requests to store metadata
- Use `X-Magento-Tags` header for PURGE requests to clean the cache entries related to that entity



Location of X-Magento-Tags

I'm Veronica



/about-us

Varnish



Web head

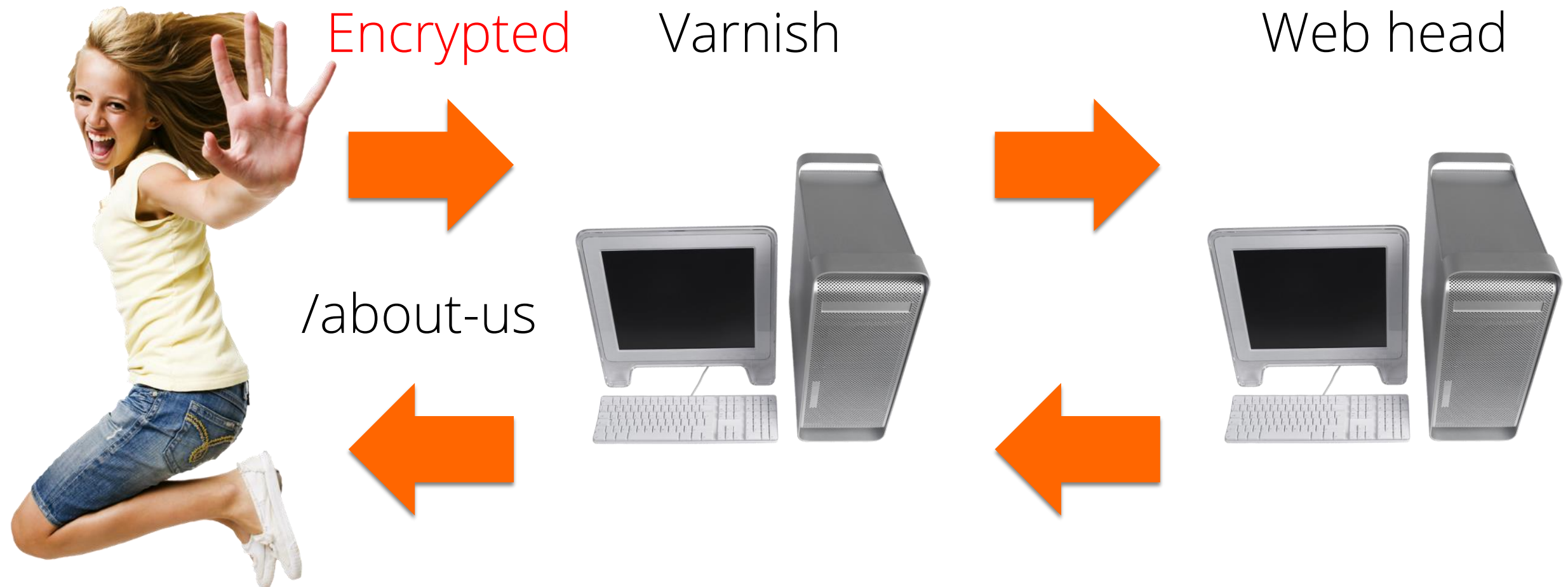


X-Magento-Tags:
cms_block_1, category_product_1



Problem of HTTPS requests with Varnish

I'm Veronica



Caching HTTPS requests with Varnish

I'm Bruce



E (443)



E

SSL/TLS
termination

X-Forwarded-Proto: https

D (80)



D

Varnish

D (8080)



D

Web head



HTTP Authentication vs. Varnish

- HTTP Authentication will be cached by Varnish
- Ignoring requests with the Authorization header in Varnish
=> you will lose the benefit of having Varnish



HTTP Authentication vs. Varnish

I'm Bruce

Remove
Authorization
header



80



8080



8081



HTTP Auth
proxy



Varnish



Web head



Magento 2's Page Cache implementation is amazing, make sure you use it correctly!



Questions?

Twitter: @nagno123



Thank You!



DevelopersParadise
2016 / Opatija / Croatia