DrupalCon
GLOBAL 2020
JULY 14-17

Caching and Performance Deep Dive

Fabian Franz

@fabianfranz

VP of Software Engineering
Tag1 Consulting

Overview

Overview

About me

- Fabian Franz

- VP of Software Engineering @ Tag1 Consulting

- Co-Author of BigPipe and the Drupal 8/9 Caching system + D7 core maintainer + subsystems ...

=> **Motivation: Teach you all I know about Caching!**

Overview

What to expect: Educational Session

- Disclaimer: Absolute beginner session!
- Some concepts from a different angle however.
- Roughly three parts with 10 min each and 5 min for Questions in between parts

Overview

What to expect

- Part 1: General caching and cache invalidation strategies (cache items, cache max-age and tags)

- Part 2: Cache variation, cache hit ratio, placeholders and uncacheable things

Part 3: Caching layers + Common Caching Pitfalls

1. What is Caching?

> In computing, a cache is a hardware or software component that stores data so that future requests for that data can be served faster; the data stored in a cache might be the result of an earlier computation or a copy of data stored elsewhere.

Wikipedia

What is Caching?

Sooo much theory ...

- Example: We have a restaurant and we prepare meals (pages)
- Pizza takes 10 min to prepare
- Takeaway => Pizza is wrapped and given out

What is Caching?

Sooo much theory ...

- Example: We have a restaurant and we prepare meals (pages)
- Pizza takes 10 min to prepare
- Takeaway => Pizza is wrapped and given out -----> THAT IS CACHING!

What is Caching?

Sooo much theory ...

- That’s a cache, performance of pizza delivery is improved
- Finite numbers of pizzas?

What is Caching?

Sooo much theory ...

- We have a magic replicator!
- Customer comes, we replicate the Pizza that we prepared earlier, and give it away

What is Caching?

Sooo much theory ...

- Every item that we cache gets a name: Cache item name or cache address
- In Drupal this is a cache ID or later this is also called “cache keys”
- Cache keys sample -- [‘pizza’, ‘margherita’] => pizza:margherita

Let’s make Pizza! :D

How to cache?

Examples for you :)

```php
$pizza = \Drupal::cache('pizzas')->get('pizza:margherita');
if ($pizza) {
  return $pizza;
}

$pizza = \Drupal::service('pizza.oven')->make('margherita');
\Drupal::cache('pizzas')->set('margherita', $pizza);

return $pizza;
```

Who sees the bug?

How to cache?

Fixed example!

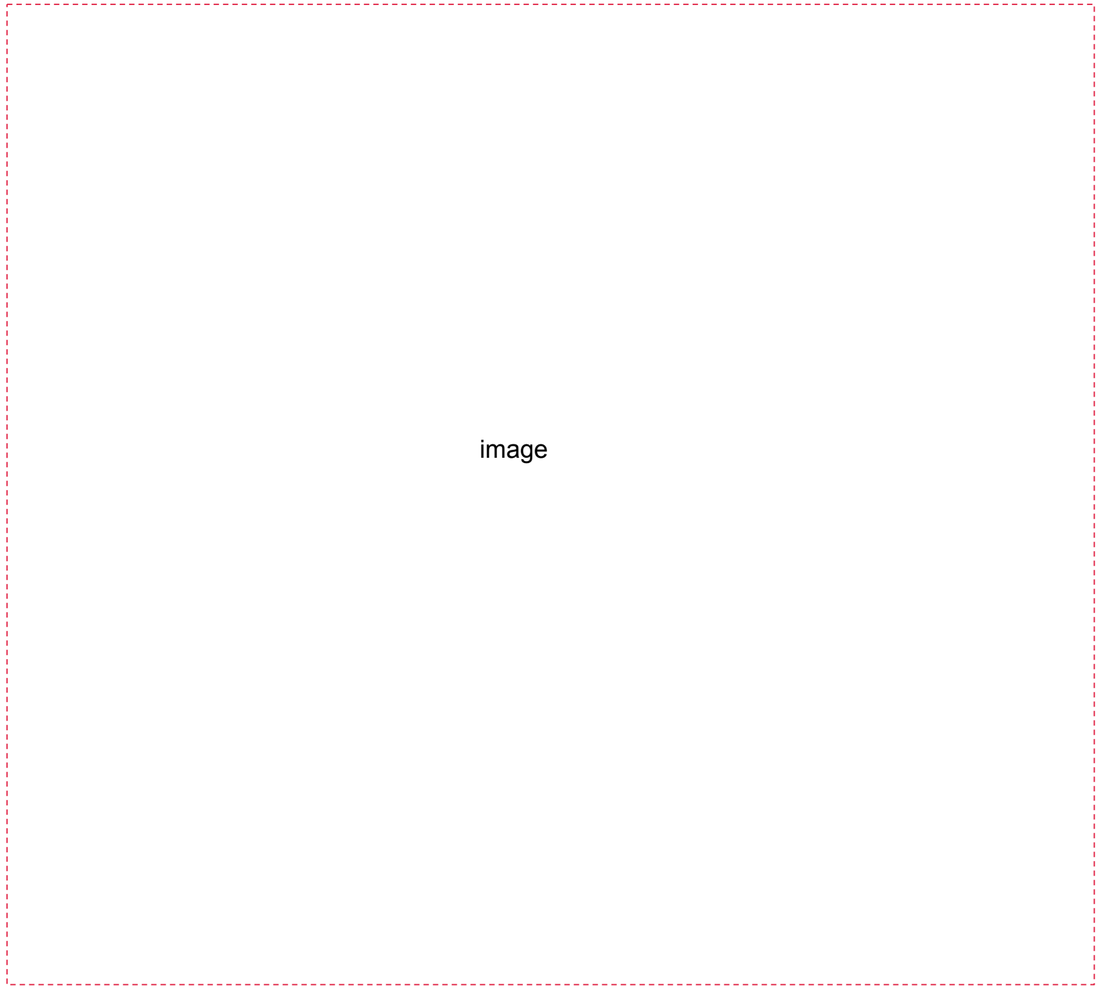
```php
$cid = 'pizza:margherita'; // Cache ID
$pizza = \Drupal::cache('pizzas')->get($cid);
if ($pizza) {
  return $pizza;
}

$pizza = \Drupal::service('pizza.oven')->make('margherita');
\Drupal::cache('pizzas')->set($cid, $pizza);

return $pizza;
```

How long is a product valid?

How long is a product valid?



- Supermarket: Best before [DATE]
- Pizza after a while looks like this =>

  Don’t want to eat it anymore …
- Solution: Expiration date

Best before:
09/2022

```php
$cid = 'pizza:margherita'; // Cache ID
$time_to_live = 10*60; // 10 min valid

$pizza = \Drupal::service('pizza.oven')->make('margherita');
\Drupal::cache('pizzas')->set($cid, $pizza, $time_to_live);

return $pizza;
```

Best before: 09/2022

- Page cache in Drupal 3-6
- Still a perfect pattern => EASY!
- Cache for 10 min unconditionally, great for high traffic sites

Weekend - let’s clean up!

Weekend!

Let’s clean-up!

```php
$cid = 'pizza:margherita'; // Cache ID
\Drupal::cache('pizzas')->delete($cid);
```

Let’s offer Frozen Margherita!

- Dough with 00-flour, pint of salt + water
- Custom made Tomato Sauce
- Mozzarella
- Basil

Let's keep it for longer

```php
$cid = 'pizza:margherita'; // Cache ID
$bin = 'frozen_pizzas';
$time_to_live = 30*24*60*60; // 30 days valid!

$pizza = \Drupal::service('pizza.maker')->makeFrozen('margherita');
\Drupal::cache($bin)->set($cid, $pizza, $time_to_live);

return $pizza;
```

Recap - How our Shop works!

Recap (Slides)

- [Customer] drives to our Pizza Shop
- [Customer] orders a frozen [Pizza Margherita]
- [Waiter] gets the [Pizza] from the fridge at the counter
- [Waiter] checks the expiration date, if it’s expired he gets one from central storage in the cellar
- [Waiter] replicates and delivers the pizza to the customer

Let’s offer Marinara as well!

- Dough with 00-flour, pint of salt + water
- Custom made Tomato Sauce
- Extra virgin olive oil
- Oregano + Garlic

It’s a vegan pizza!

Mooore Pizza!

Completely new pizza! Not a variation. Now on offer!

```php
$cid = 'pizza:marinara'; // Cache ID
$bin = 'frozen_pizzas';
$time_to_live = 30*24*60*60; // 30 days valid!

$pizza = \Drupal::service('pizza.maker')->makeFrozen('marinara');
\Drupal::cache($bin)->set($cid, $pizza, $time_to_live);

return $pizza;
```

Success! We are growing!

A better recipe for the dough!

After super-secret expedition to Italy!

Pizza-Dough 2.0

Pizza-Dough 2.0

We are lovin’ it!

- Invalidate all the (cached) old pizzas
- Not wait for 30 days
- How do we know if they are new or old?

Pizza-Dough 2.0

Naive solution

```
$pizza = \Drupal::cache('frozen_pizzas')->get('pizza:margherita:dough_version=2');
if ($pizza) {
  return $pizza;
}
```

- This does not scale :(
- All old versions are kept around

pizza:marinara:dough_version=10

pizza:marinara:dough_version=4

pizza:margherita:dough_version=4

What a Mess!

pizza:margherita:dough_version=10

pizza:marinara:dough_version=2

pizza:marinara:dough_version=3

pizza:margherita:dough_version=3

pizza:marinara:dough_version=10

Pizza-Dough 2.0

Let’s tag it!

```
name: Margherita
expires: 08/2020
tags:
  - dough_version: 2
```

```
name: Marinara
expires: 08/2020
tags:
  - dough_version: 2
```

Pizza-Dough 2.0

Let’s tag it!

```php
$cid = 'pizza:marinara'; // Cache ID
$bin = 'frozen_pizzas';
$time_to_live = 30*24*60*60; // 30 days valid!

$pizza = \Drupal::service('pizza.maker')->makeFrozen('marinara');
\Drupal::cache($bin)->set($cid, $pizza, $time_to_live, ['dough_version']);
```

Pizza-Dough 2.0

Let's tag it!

```php
$cid = 'pizza:marinara'; // Cache ID
$bin = 'frozen_pizzas';
$time_to_live = 30*24*60*60; // 30 days valid!
$cache_tags = ['dough_version'];

$pizza = \Drupal::service('pizza.maker')->makeFrozen('marinara');
\Drupal::cache($bin)->set($cid, $pizza, $time_to_live, $cache_tags);
```

Release a new dough version, do that:

```php
\Drupal::cache($bin)->invalidateTags(['dough_version']);
```

Pizza-Dough 2.0

Tagging is versioning!

- Drupal versions the tags automatically
- cachetags table: `tag, invalidations`
- It’s a version number conceptually!

Pizza-Dough 2.0

Ways of Tagging

- v3.1.0 (versions)
- 2020-07-15 (timestamps)
- Snow Leopard (names)
- 1..10000 (counters)

Pizza-Dough 2.0

This ain’t easy

- node:1 is saved and cache tag is invalidated (v42 -> v43)
- node:1 cache tag now SHOULD BE v43
- Anything tagged with node:1 must have value of v43, else it’s invalid

Pizza-Dough 2.0

This ain’t easy

- Complex, but once mastered this is so powerful:

  Cache Item = {Name, tag=v42}

  Canonical Store = {Current Version of tag = v43}

Pizza-Dough 2.0

This ain’t easy

Hint: **Everything in the same request always uses the same current version.**

In other words: The waiter just checks the list of dough versions e.g. once a day and not every minute.

Recap - How our Shop works - now with tagging!

Recap (Slides)

- [Customer] drives to our Pizza Shop
- [Customer] orders a frozen [Pizza Margherita]
- [Waiter] gets the [Pizza] from the fridge at the counter
- [Waiter] checks the expiration date and tags
- [Waiter] marks the pizza as valid or invalid
- If the pizza is not valid, he gets one from central storage in the cellar
- [Waiter] replicates and delivers the pizza to the customer

Recap

All that we learned so far!

We now know how to:

- Get an item from the cache
- Set an item into the cache

Recap

Three ways to expire the cache! *sing*

- Direct deletion / invalidation by name of item [cache id - name]
- Time based (TTL - time to live) invalidation [cache - max-age]
- Tag based invalidation [cache - tags]

Recap

Core is cheating :p

We also implicitly created a new cache:

- The list of versions for the tags (we store it for the time of the request)

Hence: Cache tags DON’T solve the problem of cache invalidation, they just move it to somewhere else.

1. What is Caching?

Question Time!

2. What should you cache?

2 years later

Grown even more!

Success is great!

Ready for new products!

Pizza-Shop 2.0

Gluten-free dough, vegan mozzarella, pizza spinacci, ...

- New pizza variations
- Gluten free offering
- Vegan Margherita offering (Marinara was always vegan!)

Quick Recap
(now with 100% more variation)

Recap (Slides)

- [Customer] comes and orders a pizza
- [Waiter] asks for the preferences (vegan/gluten free) (cache context)
- [Waiter] checks the fridge for the wanted variation
- [Waiter] gives the wanted variation to the customer (cache hit) or produces it (cache miss) and then stores it in the fridge

Pizza-Shop 2.0

Let’s add it to the name (again?!)

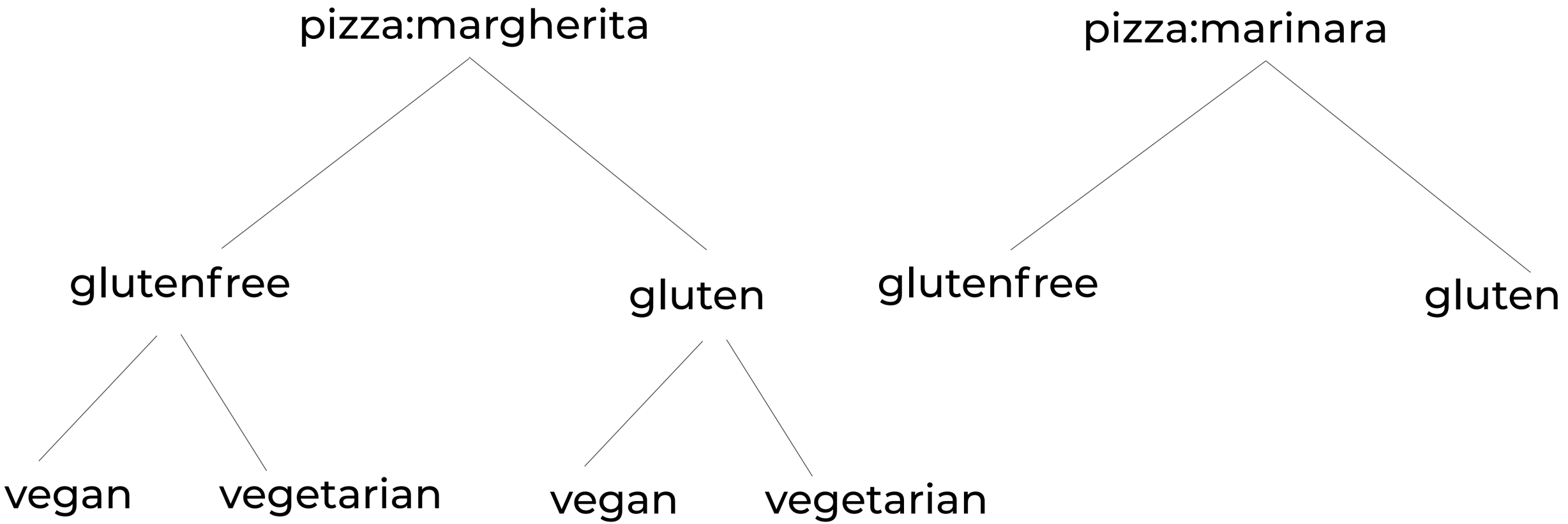
- - pizza:margherita:vegan:glutenfree
- - pizza:margherita:vegan:gluten
- - pizza:margherita:vegetarian:glutenfree
- - pizza:margherita:vegetarian:gluten
- - pizza:marinara:vegan:glutenfree
- - pizza:marinara:vegan:gluten

Hmm, nope!

- - pizza:marinara:vegetarian:glutenfree
- - pizza:marinara:vegetarian:gluten

Pizza-Shop 2.0

What we would like:



Cache Contexts

Vary me if you can!

- … are used for variation in Drupal 8/9
- … are computed on demand
- … internally adds the cache context values to the Cache ID name

Cache Contexts

Pizza-Shop 2.0

Name: pizza:margherita

Cache Contexts:

- vegan=yes|no
- gluten_free=yes|no

Name: pizza:margherita:vegan=yes|no:glutenfree=yes|no

Expires: 09/2020

Tag:

- dough_version=2

Cache Contexts

Pizza-Shop 2.0

Name: pizza:marinara

Cache Contexts:

- gluten_free=yes|no

Name: pizza:marinara:glutenfree=yes|no

Expires: 09/2020

Tag:

- dough_version=2

Quick Recap
(now with intelligent variation)

Recap (Slides)

- [Customer] drives to our Pizza Shop
- [Customer] orders a frozen [Pizza Margherita] (Cache ID)
- **[Waiter] looks at the [Pizza] variations for Margherita (Cache Context Router)**
- **[Waiter] asks the [Customer] for his preferences (vegan and/or gluten-free?) (Cache Context Execution)**
- [Waiter] gets the preferred [Pizza] from the fridge at the counter (Cache Retrieval)
- [Waiter] checks the expiration date and tags (Cache validation)
- [Waiter] marks the pizza as valid or invalid
- If the pizza is not valid, he gets one from central storage in the cellar (Cache miss)
- [Waiter] replicates and delivers the pizza to the customer (Cache hit)

Cache Contexts

Practical Example

- Only works with Render Arrays
- Took us quite some time to understand in depth
- RenderCache could provide it as Service in the future

Cache Contexts

Direct vs. Render Array - Compare:

```php
$cid = 'pizza:marinara'; // Cache ID
$bin = 'frozen_pizzas';
$time_to_live = 30*24*60*60; // 30 days valid!
$cache_tags = ['dough_version'];

$pizza = \Drupal::service('pizza.maker')->makeFrozen('marinara');
\Drupal::cache($bin)->set($cid, $pizza, $time_to_live, $cache_tags);
```

Cache Contexts

Direct vs. **Render Array** - Compare:

```php
$build = [
  '#cache' => [
    'bin' => 'frozen_pizzas',
    'keys' => ['pizza','marinara'],
    'max-age' => $time_to_live,
    'tags' => $cache_tags,
  ],
];

$build['#pre_render'][] = function($elements) {
  $elements['pizza'] = \Drupal::service('pizza.maker')->makeFrozen('marinara');
  return $elements;
};
```

Cache Contexts

Practical Example using Render Array

- Provide the Cache metadata via #cache
- Provide the Cache miss function (#pre_render)

Cache Contexts

Render Array with Cache Contexts added

```
$build = [
  '#cache' => [
    'contexts' => ['user.vegan', 'user.glutenfree'],
    'keys' => ['pizza', $pizza_name],
    'max-age' => $time_to_live,
    'tags' => $cache_tags,
  ],
];

$build['#pre_render'][] = function($elements) use ($pizza_name) {
  $elements['pizza'] = \Drupal::service('pizza.maker')->makeFrozen($pizza_name);
  return $elements;
};
```

Cache Contexts

Render Array with Cache Contexts added

```php
$build = [
  '#cache' => [
    'keys' => ['pizza', $pizza_name],
    'max-age' => $time_to_live,
    'tags' => $cache_tags,
  ],
];

$build['#pre_render'][] = function($elements) use ($pizza_name) {
  $elements['pizza'] = \Drupal::service('pizza.maker')->makeFrozen($pizza_name);
  return $elements;
};

$build['#cache']['contexts'] = ['user.vegan', 'user.glutenfree'];
```

Cache Contexts

Render Array with dynamic cache contexts

```php
$build = [
  '#cache' => [
    'keys' => ['pizza', $pizza_name],
    'max-age' => $time_to_live,
    'tags' => $cache_tags,
  ],
];

$build['#pre_render'][] = function($elements) use ($pizza_name) {
  $elements['pizza'] = \Drupal::service('pizza.maker')->makeFrozen($pizza_name);

  $elements['#cache']['contexts'][] = 'user.glutenfree';
  if ($pizza_name == 'margherita') {
    $elements['#cache']['contexts'][] = 'user.vegan';
  }
  return $elements;
};
```

Cache Contexts

Creating a Cache Context: src/UserVeganCacheContext.php

```
class UserVeganCacheContext extends UserCacheContext {

  /**
   * {@inheritdoc}
   */
  public static function getLabel() {
    return t('Vegan User');
  }

  /**
   * {@inheritdoc}
   */
  public function getContext() {
    return $this->user
      ->field_vegan
      ->value() ? 'yes' : 'no';
  }

}
```

Cache Contexts

Creating a Cache Context: pizza.services.yml

```
services:
  cache_context.user.vegan:
    class: Drupal\pizza\UserVeganCacheContext
    arguments: ['@current_user']
    tags:
      - { name: cache.context}
```

TADA! That works great!

Alert: Fridge is full!

So many variations ...

Help!

Soooo many variations ...

- Pizza Spinacci is bought way less
- Custom pizza is “uncacheable”
- Check your cache hit ratio and invalidations: https://www.drupal.org/project/cache_metrics

Help!

Soooo many variations ...

- Let’s disable the cache
- Easiest: Not cache at all

Disable cache

Max-Age = 0

```
$build['#cache']['max-age'] = 0;
```

Disable cache

For cacheable objects

```
$cacheable_object->setCacheMaxAge(0);
```

Disable Cache

Full example

```php
<?php

$build['#pre_render'][] = function($elements) use ($pizza_name, $ingredients) {
  if ($pizza_name == 'custom') {
    $pizza = \Drupal::service('pizza.maker')->makeCustomPizza($ingredients);
    $elements['#cache']['max-age'] = 0;
    return $elements; // We early return ...
  }

  if ($pizza_name == 'spinacci') {
    $elements['#cache']['max-age'] = 0; // We fall through ...
  }

  // [...] The rest of the callback

  return $elements;
};
```

Disable Cache

Practical Example using Render Array

- Cache max-age=0 set after function has been rendered

- **Pitfall**: Clear your cache (drush cr) after making such a change during local development

  -> Happened to me more often than I'd like to admit ...

Disable Cache

Practical Example using Render Array

- **Pitfall**: Clear your cache (drush cr) after making such a change during local development

- 3 ways:
  - `drupal cache:tag:invalidate rendered`
  - `drush cache:tag rendered`
  - `\Drupal\Core\Cache::invalidateTags(['rendered']);`

Disable Cache

Before it is retrieved from the Cache

```
$build = [
  '#cache' => [
    'keys' => ['pizza', $pizza_name],
    'max-age' => $time_to_live,
    'tags' => $tags,
  ],
  '#pizza_name' => $pizza_name,
  '#pre_render' => [$this, 'makePizza'],
];

if (in_array($pizza_name, ['custom', 'spinacci'])) {
  $build['#cache']['max-age'] = 0;
}
```

Disable Cache

Practical Example using Render Array

- It’s always more efficient to disable the cache before the item is retrieved from the Cache
- Similar to: Request based Cache Policy

Cache Chains

No Pizza-Shop creates the Pizza always from Scratch

Pizza is made from pre-prepared things:

Dough (12-24 hrs till ready), Tomato sauce, Ingredients

Composing Sites

Pages consist of different cached and uncached parts

- Main page response (need to custom cache)

- Blocks, Menus, Header, Footer, ...

  [Decoration around the main page response]

Pizza Funghi

2 ways to create a Pizza with Mushrooms!

- Start with the empty pan, add the dough, add the tomato sauce add the mozzarella cheese and then add the mushrooms.

- Start with a finished pizza margherita and just add the mushrooms.

Pizza Funghi

+ Dynamic Page Cache

That is what the true power of dynamic page cache is:

- We cache the response
- We add flavor / placeholders afterwards

Pizza Funghi

+ Dynamic Page Cache

Drupal 8+9 with two ways for really dynamic things:

- Disable the (dynamic) page cache; just cache all the inner parts (blank pan, create from scratch)

- Cache the whole response in dynamic page cache and just add some placeholders for dynamic data

Pizza Funghi

+ Dynamic Page Cache

- Glutenfree cannot be a placeholder

- It’s the foundation of our pizza

- Both are needed:

  - Variation (varies all cache entries)

  - Placeholders (out of band)

  => Decide case-by-case

Placeholders

Pizza M+X

Margherita + Placeholders

- A placeholder in Drupal: Can be independently rendered. Must not depend on anything that has been executed before.

For example:

- It’s not possible to add more wheat to the dough after the pizza is finished already.

Pizza M+X

Classified - Top Secret - Placeholders internal structure

```
$elem['#attached']['placeholders']['%%ingredients_placeholder%%'] = $build;
$elem['#markup'] = '%%ingredients_placeholder%%';
```

Pizza M+X

+ Placeholders

Contract:

- Executed after all other parts have been rendered
- #pre_render => #lazy_builder (stronger contract)

Placeholders

LazyBuilder vs. **#pre_render**

```php
$build = [
  '#cache' => [
    'keys' => ['pizza', $pizza_name],
    'max-age' => $time_to_live,
    'tags' => $cache_tags,
  ],
];
$build['#pre_render'][] = function($elements) use ($pizza_name) {
  $elements['pizza'] = \Drupal::service('pizza.maker')->makeFrozen($pizza_name);
  return $elements;
};
```

Code

Lazy Builder - Auto Placeholdering

```
$build = [
  '#cache' => [
    'keys' => ['pizza', $pizza_name],
    'max-age' => $time_to_live,
    'tags' => $cache_tags,
  ],
];
$build['#lazy_builder'] = [
  '\Drupal\pizza\PizzaLazyBuilder::build',
  [$pizza_name],
];
```

Code

Lazy Builder - Explicit Placeholder

```
$build = [
  '#cache' => [
    'keys' => ['pizza', $pizza_name],
    'max-age' => $time_to_live,
    'tags' => $cache_tags,
  ],
];
$build['#lazy_builder'] = [
  '\Drupal\pizza\PizzaLazyBuilder::build',
  [$pizza_name],
];


$build['#create_placeholder'] = TRUE;
```

Pizza M+X

+ LBs + Placeholders - Pitfalls (!)

Lazy Builders:

- Must not contain complex data (enforced!)
- Must not depend on the main page request

Pizza M+X

+ LBs + Placeholders

Lazy Builders + Placeholders allows to:

- Use **big_pipe** (in Core, enable and good to go!)
- Cache the uncacheable
- Break up variation: per-page/per-user => per-page + per-user

2. What should you cache?

Question Time!

3. Where should you cache?

Shop is even more successful!

But Customers
need to drive to us :(

Many drive for 2 hours and more

Can’t we do something about that?

Solution: We offer our pizza in supermarkets around the world!

Solution:

Content Delivery Network (CDN)

CDN

Pizza Delivery Network (PDN!)

Drupal 8/9 makes it easy:

- Choose CDN (Akamai, Cloudflare, Fastly) or Varnish
- Enable module
- Profit!

CDN

Pizza Delivery Network (PDN!)

CDN does the checks:

- Has the pizza expired?
- Is the dough_version still matching?
- dough_version changes => Give CDN a heads up!

CDN

Pizza Delivery Network (PDN!)

See headers for yourself:

- X-Drupal-Cache-Tags
- Debug option

Code

Title

```
parameters:
  # Cacheability debugging:
  #
  # Responses with cacheability metadata (CacheableResponseInterface instances)
  # get X-Drupal-Cache-Tags and X-Drupal-Cache-Contexts headers.
  #
  # For more information about debugging cacheable responses, see
  # https://www.drupal.org/developing/api/8/response/cacheable-response-interface
  #
  # Not recommended in production environments
  # @default false
  http.response.debug_cacheability_headers: true
```

CDN

Pizza Delivery Network (PDN!)

And this is the result:

- X-Drupal-Cache-Tags: dough_version
- Expires: 09/2022

Great - but what about the dough itself?

Need to get it from warehouse 10 miles away.

Let’s put it in a fridge under the counter

Efficiency 3.0

Dough near the counter

Drupal has ChainedFast:

- ACPu (shared memory within PHP process)

Main rule of thumb:

- **If you have things that are seldom changing, put it into a special bin and connect that bin to “chained fast”. (mostly read only cache traffic)**

Efficiency 3.0

The dough is always near the counter - yeah!

```
$settings['cache']['bins']['pizza_dough'] = 'cache.backend.chainedfast';
```

No Efficiency 3.0

The custom made pizzas should NOT be stored near the counter

Second rule:

Never put chained fast on things that are often changing or have lots of variations:

- You can get serious write lock problems and performance will decrease!
- If the cache is full it can lead to lock-ups as a full garbage collection needs to be performed.

Efficiency 3.0

APCu is really cool :D

APCu is ideal (and used in Drupal) for:

- FileCache (depends only if the file has changed)
- ClassCache (depends only on where the class sits on the filesystem)
- Config cache (is invalidated only if config changes)

This shows now also the importance of ‘bins’ as those can have different cache backends associated with them.

Don’t forget Redis / Memcached

Efficiency 4.0

Memcached/Redis is also cool

- MySQL is a warehouse that’s across the street
- Memcached / Redis is a fridge that is in the room next door
- ACPu is the fridge below the counter.

Efficiency 4.0

Advantages and Disadvantages, hmmm - what to do ...

- MySQL: Large Storage space / Slow: 2-5 ms response times usually
- Memcached / Redis: Medium storage space / Fast: 0.5 - 1 ms response times usually
- APCu: Small storage space / Fastest: 0.05 ms usually

Efficiency 4.0

Create Pizza + Deliver Pizza are different cache paths

It is important to distinguish two cases:

- Caches used for creating the pizza (MySQL, APCu, Memcached) [from parts]

- Caches used for delivering the pizza to the customer (MySQL, Memcached, CDN, Browser Cache)

Lot’s of customers at once

=> Pizza with Spring Onions

Spring Onions

Only seconds TTL

- The spring onions can only be cached for a very short while (micro-caching)
- Potential bottleneck

  => Stampede protection (build into most CDNs)

  SHIELD!

Stampede Protection

Microcaching + Stampede Protection

- Inefficient: Prepare lot’s of pizzas in parallel
- Instead: Prepare one spring onion pizza and then just replicate it.

Stampede Protection

```php
public function stampedeProtect($cid) {
  $item = $this->cache->get($cid);
  if ($item) {
    return $item;
  }

  $acquired_lock = $this->lock->acquire('stampede:' . $cid);
  if (!$acquired_lock) {
    sleep(1);
    return $this->stampedeProtect($cid); // Let's try that again.
  }

  // Rebuild cache
  $item = $this->rebuild();

  $this->cache->set($cid, $item, 30); // Cache for only 30 seconds
  $this->lock->release($acquired_lock);

  return $item;
}
```

Common Caching Pitfalls

Common Caching Pitfalls

Planning to fail is better than failing to plan

Plan your caching strategy:

- Know what depends on what
- Known when something needs to be invalidated
- https://drupal.org/project/renderviz module can be a really nice help here.

Have fun and I'll
make a Pizza now ;)

yummy

The End!

More Questions?

Follow me: @fabianfranz

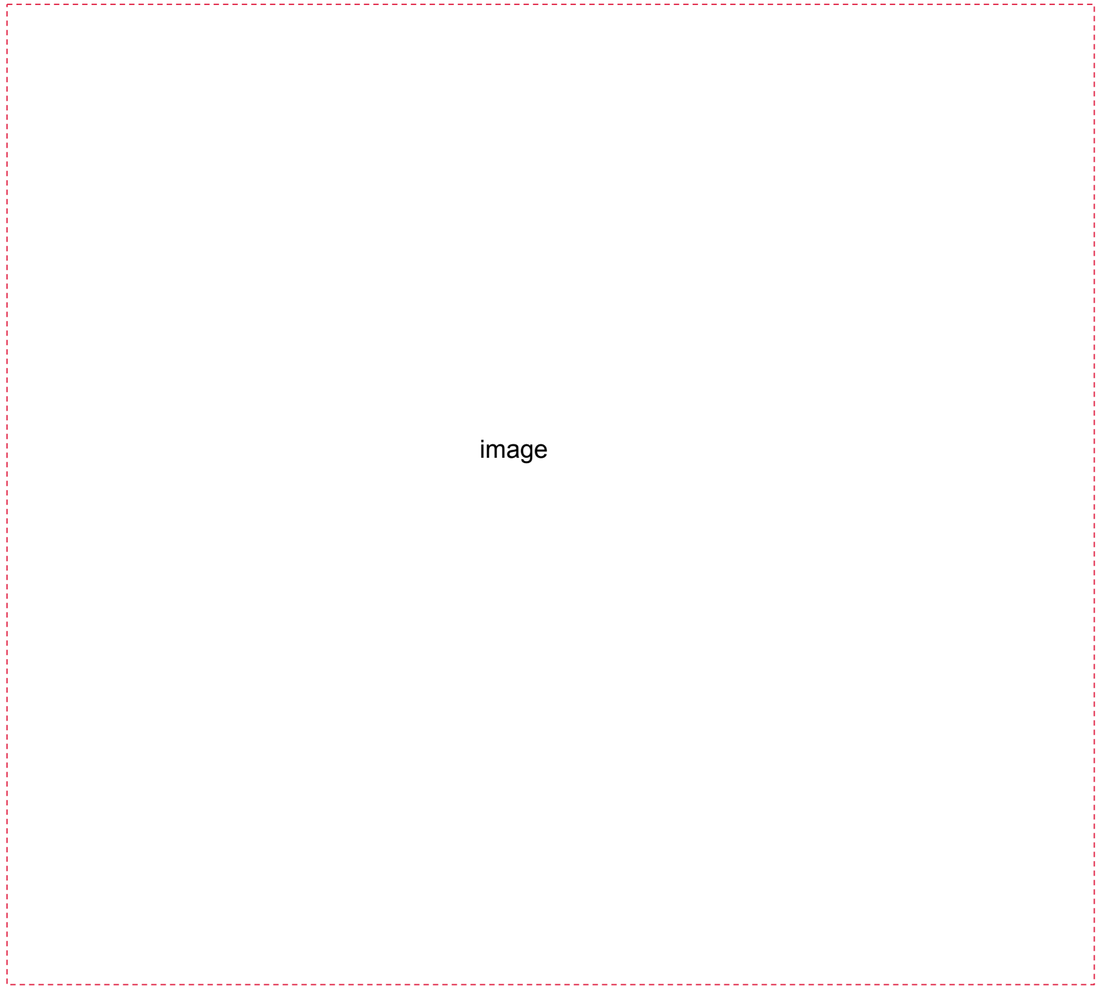
Code

Title

Title slide
Additional title

Title

Second line

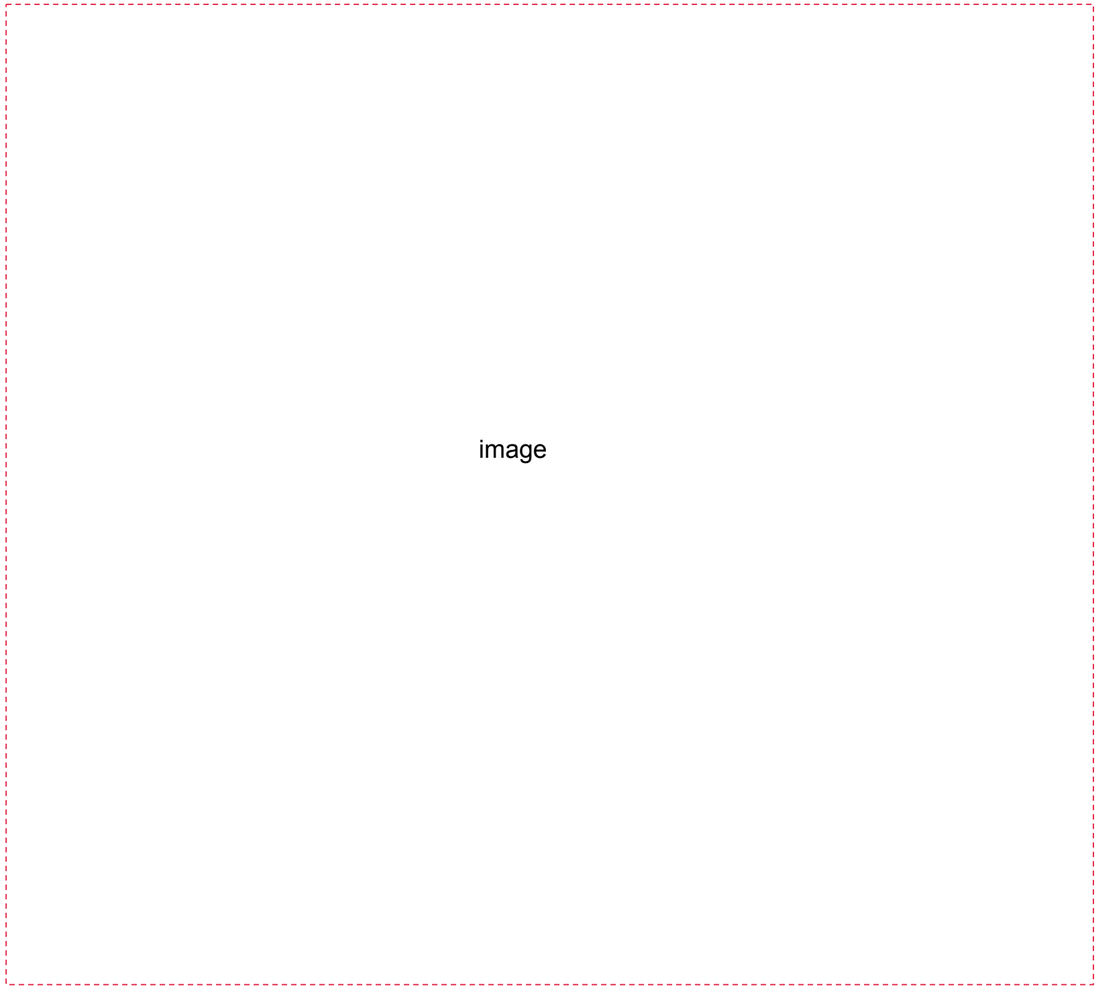


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque ultricies dolor id mi auctor.

Title

Second line

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque ultricies dolor id mi auctor.



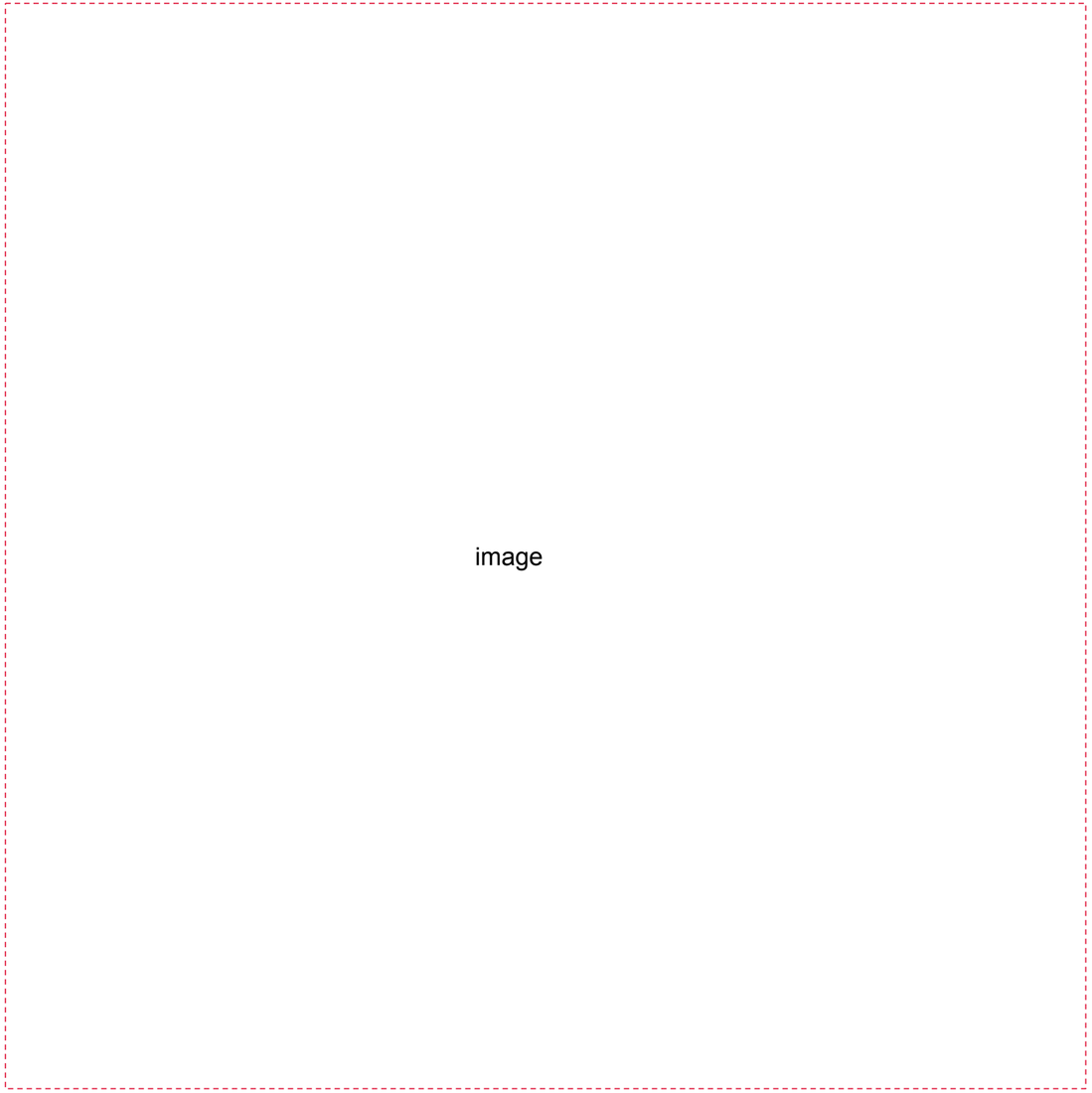
image

Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque ultricies dolor id mi auctor.

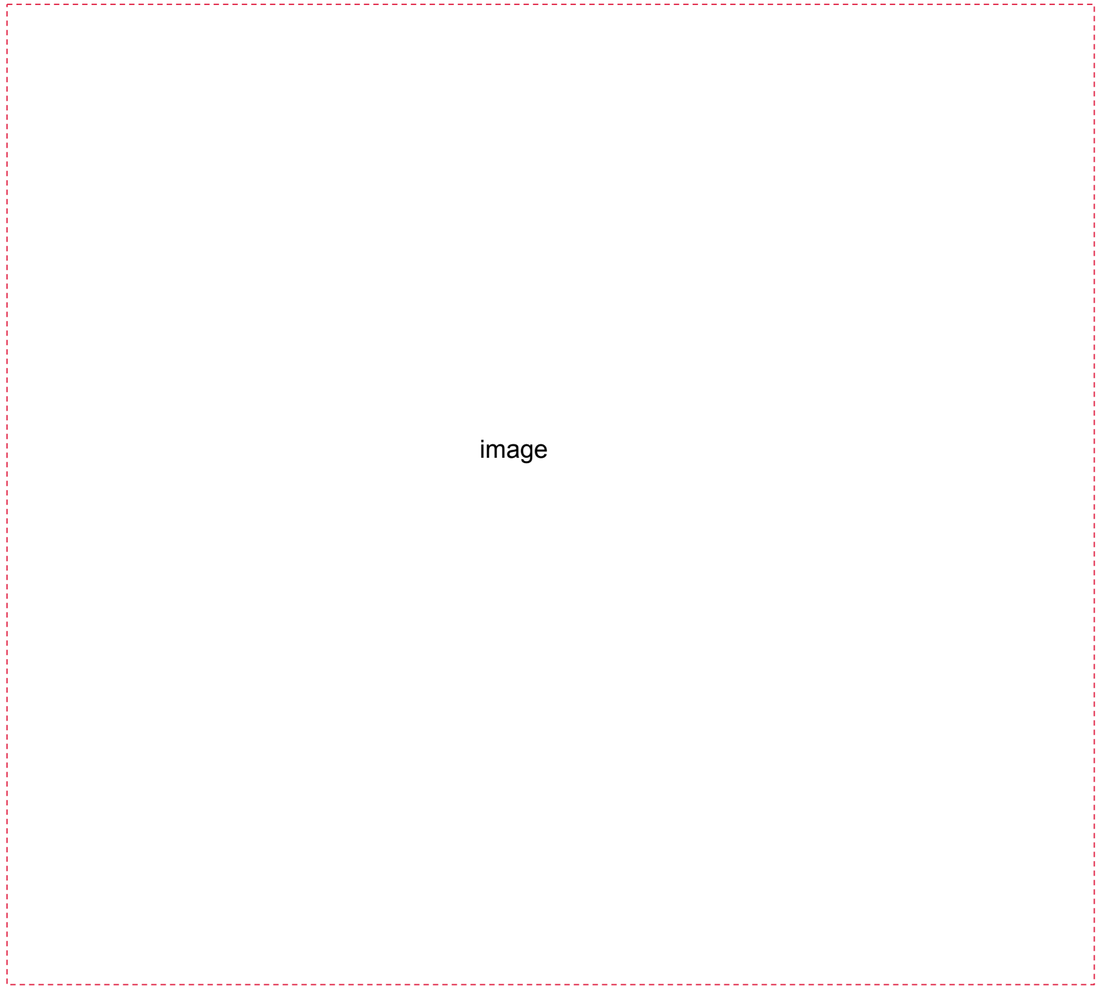
Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque ultricies dolor id mi auctor.



Title

Second line



- List Item 1
- List Item 1
- List Item 1

Some Section header
Second Line

Title

Second line

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque ultricies dolor id mi auctor, vel rutrum diam sodales. Duis nulla justo, commodo

Title

Second line

- List Item 1
- List Item 1
- List Item 1
- List Item 1
- List Item 1

- List Item 1
- List Item 1
- List Item 1
- List Item 1
- List Item 1

“

This will be a quote about something or someone

”

AUTHOR


GLOBAL
DrupalCon
The Open Source Digital Experience Conference
ONLINE EVERYWHERE
2020 JULY 14-17