Designing for Optimal Performance in SharePoint Online Lite*

Sean P. McDonough
Microsoft MVP (Office Development, Office Servers and Services)
National Office 365 Solution Manager
Cardinal Solutions Group

@spmcdonough on Twitter (for heckling purposes)

About Cardinal

Founded in 1996
Cincinnati Ohio

350+ FTEs
$50M+ Revenue

Cincinnati
Columbus
Charlotte
Raleigh
Tampa

Mobile
Portals & Collab
UXD
Application Dev
WEM
BI

Agile Coaching
Business Analysis
Project Management

cardinal

Our Agenda

- SharePoint Online (SPO) Implementation
- Acknowledging the Reality of Plumbing
- SharePoint Online Diagnostics and Tools
- Design and Development Guidance
- References

But first ...


An important note

This is
Office 365

hanging and updating it"

Please don

Office 365 is an "evergreen service"

eaning "Microsoft is always changing and updating

An important note

Office 365 is an "evergreen service"

meaning "Microsoft is always changing and updating it"

What I show you today ...

- will probably be true tomorrow

always changing and updating

What I show you today ...

- will probably be true tomorrow
- has a good chance of being true next week
- might be true in month
- probably worth questioning and re-evaluating in a year

Please don't dig this up in five years and then send me hate mail because I presented something that is no longer accurate due to a SharePoint Online service change.

Please don't dig this up in five years and then send me hate mail because I presented something that is no longer accurate due to a SharePoint Online service change.

Dear Sean,

I was reviewing a presentation you put together five years ago, and I found elements that were incorrect. You are a horrible person and you should never touch SharePoint Online again.

Love you lots!

- an attendee

First Stop:

Some basic SharePoint farm architecture

(and why that matters with SPO)


Welcome to the farm!

u might think that SPO is simply an extension of this pat

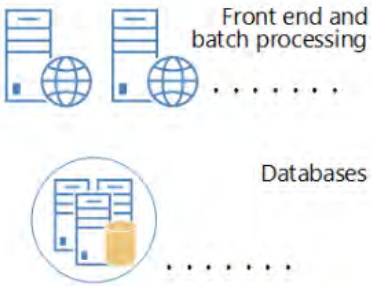
"Small Farm"

<1,000 users

Fault tolerance for simple workloads with small volumes of content

Two tiers:
- Combined front-end and batch processing servers
- Database servers

Scale the number of servers as needed.



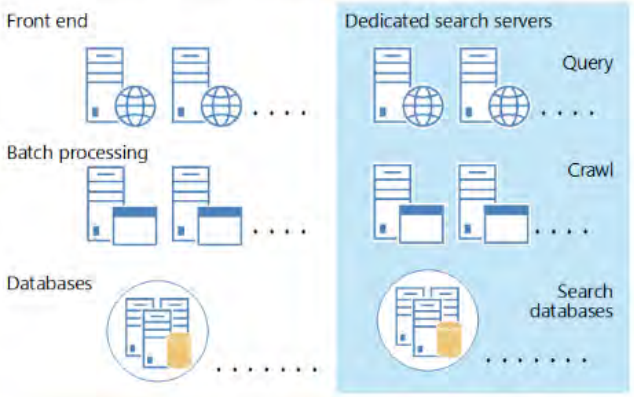
"Medium Farm"

<10,000 users

Dedicated search servers for up to 10 million items.

Three tiers:
- Front-end servers
- Batch processing servers
- Database servers

Scale the number of servers as needed.

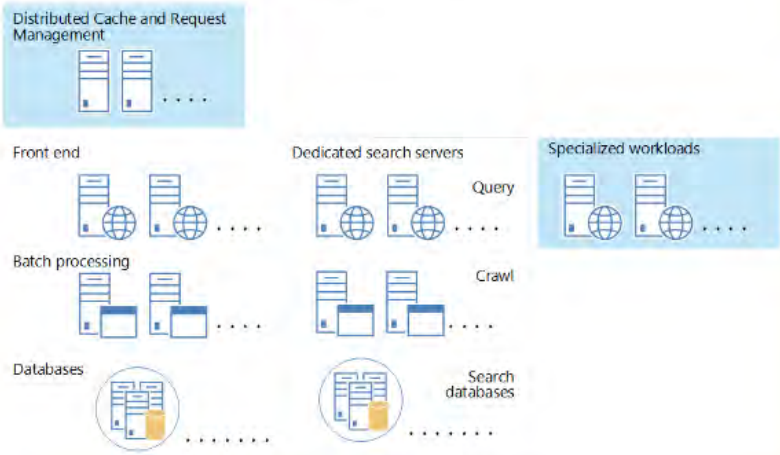


"Large Farm"

More than 10,000 users

Additional server types to support large farms.

This farm represents each of the server roles that are recommended. For each server role the servers are configured identically. Scale each server role independently. Large farms benefit by adding dedicated servers for Distributed Cache and by adding Request Management.



You might think that SPO is simply an extension of this pattern.

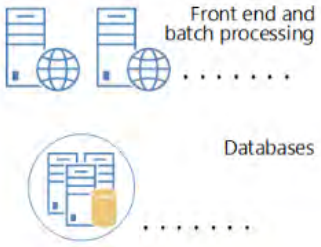
"Small Farm"

<1,000 users

Fault tolerance for simple workloads with small volumes of content

Two tiers:
- Combined front-end and batch processing servers
- Database servers

Scale the number of servers as needed.



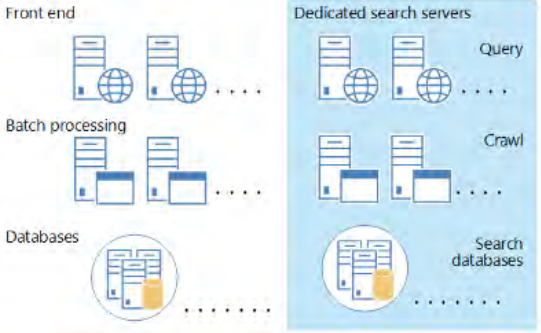
"Medium Farm"

<10,000 users

Dedicated search servers for up to 10 million items.

Three tiers:
- Front-end servers
- Batch processing servers
- Database servers

Scale the number of servers as needed.

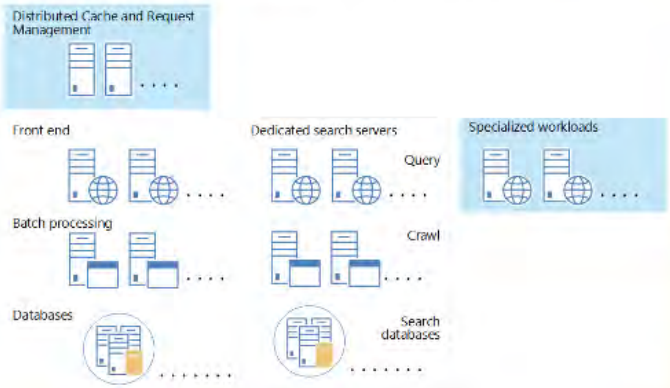


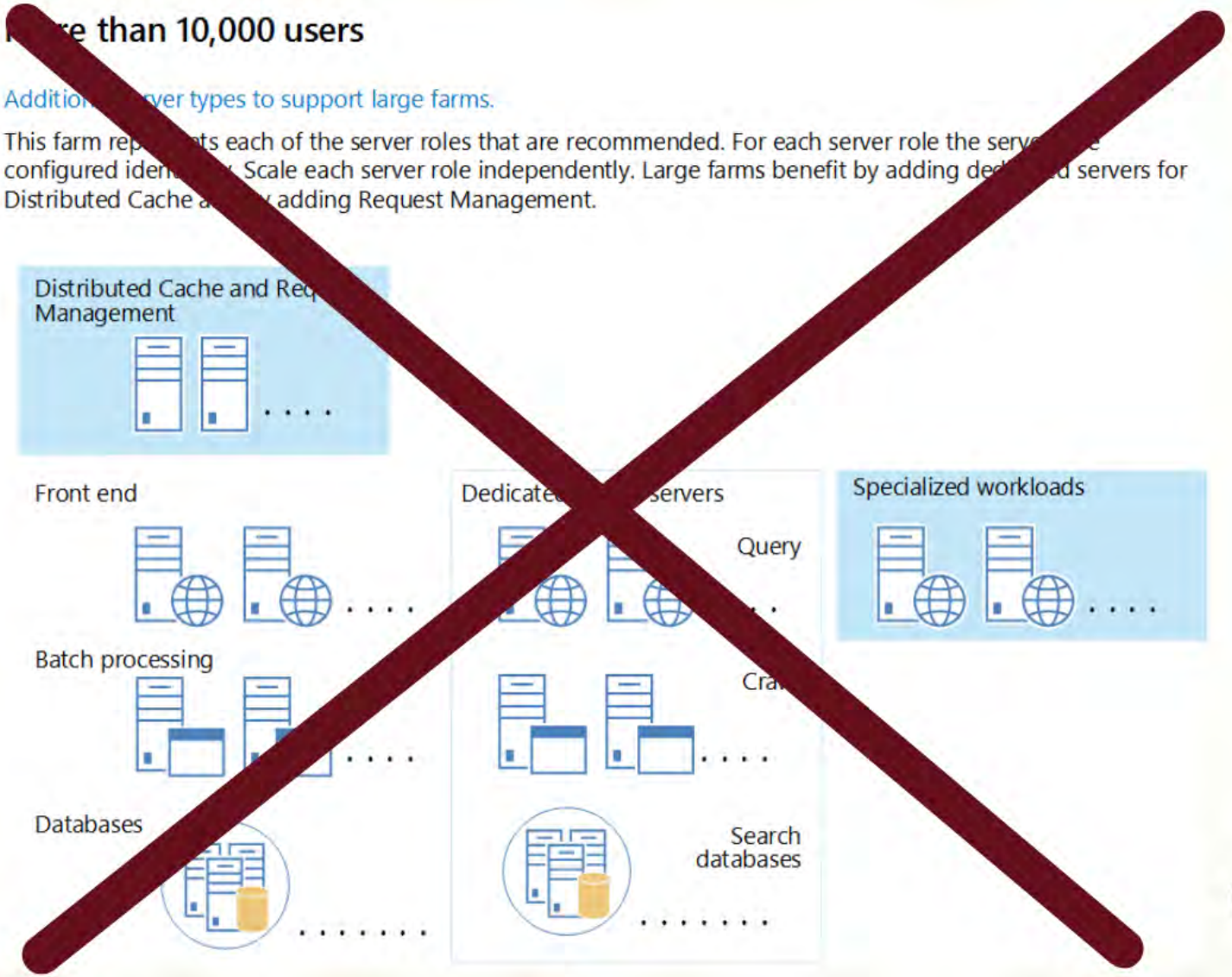
"Large Farm"

More than 10,000 users

Additional server types to support large farms.

This farm represents each of the server roles that are recommended. For each server role the servers are configured identically. Scale each server role independently. Large farms benefit by adding dedicated servers for Distributed Cache and by adding Request Management.





It is not.

well, not really ...

You might think that SPO is simply an extension of this pattern.

This is a stamp
too (well, several of them)

This is a stamp too (well, several of them)

Datacenter 1..N:

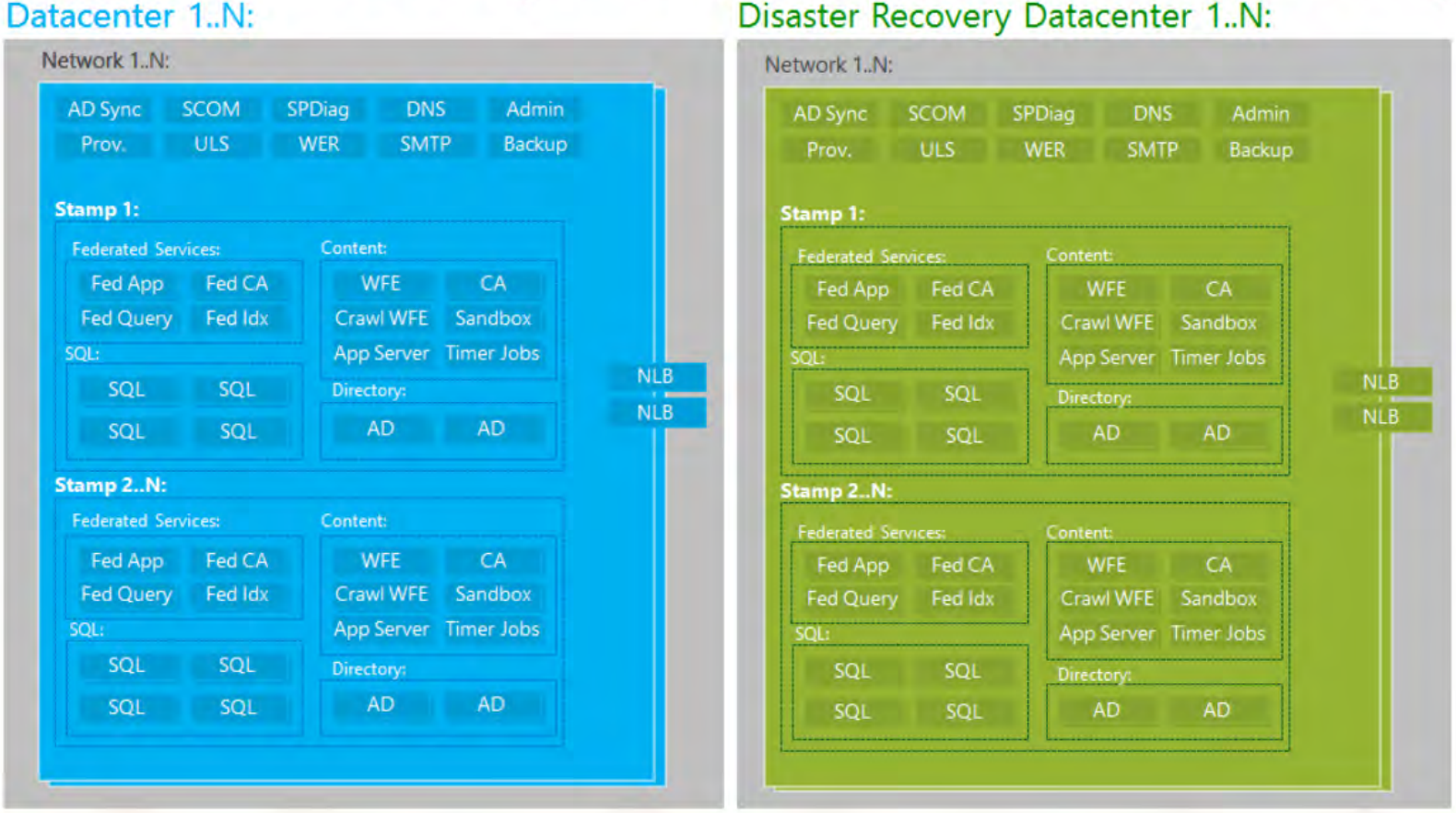
Network 1..N:

Disaster Recovery Datacenter 1..N:

Network 1..N:

Grid Manager

This is a stamp too (well, several of them)

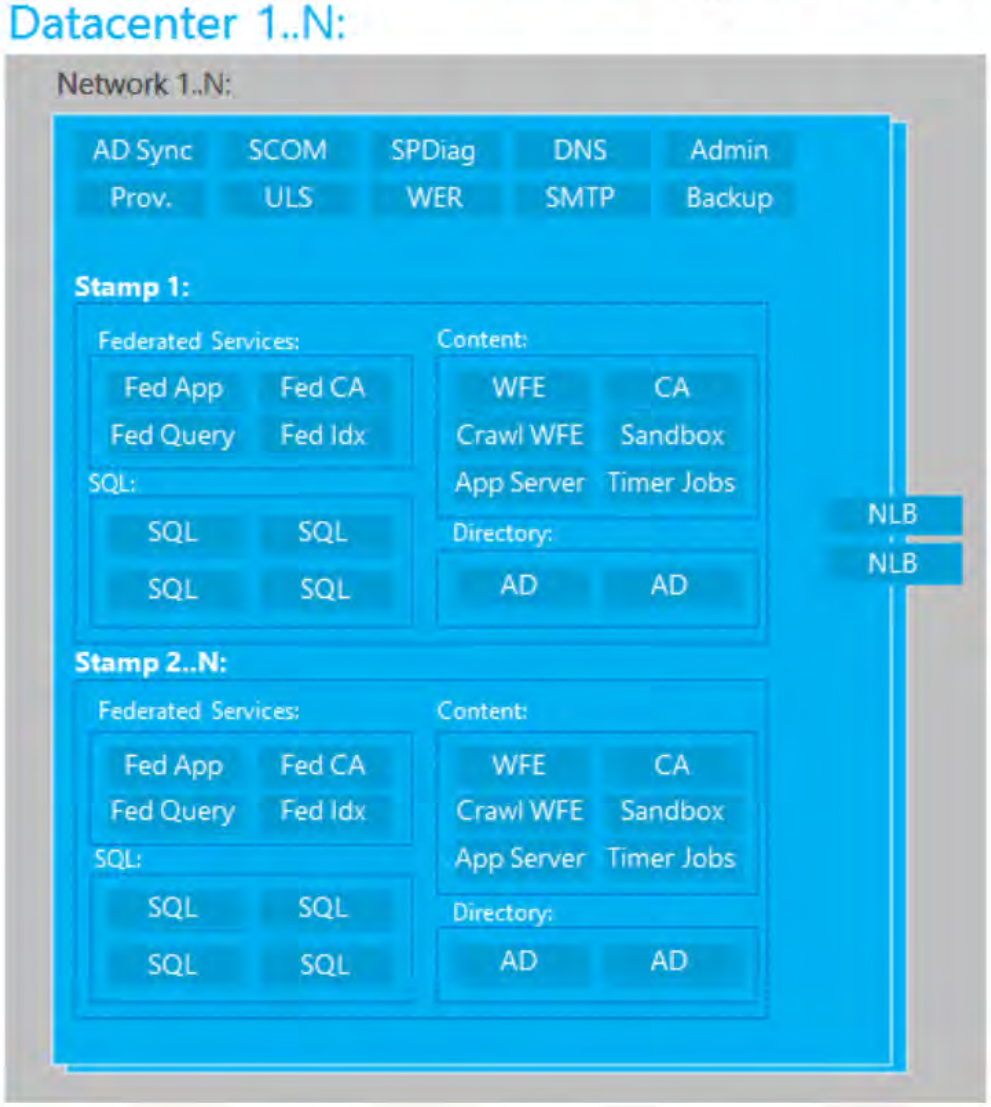


- Grid Manager
- Global Directory
- Tenant Admin (UI)
- Commerce backend
- DNS (multiple)
- OrgID Auth, Svc.
- Incident Management
- Azure (Windows/SQL)
- CDN Services

Looking at the representation of an individual stamp, you might think it's only 16 servers.

This
too (well,

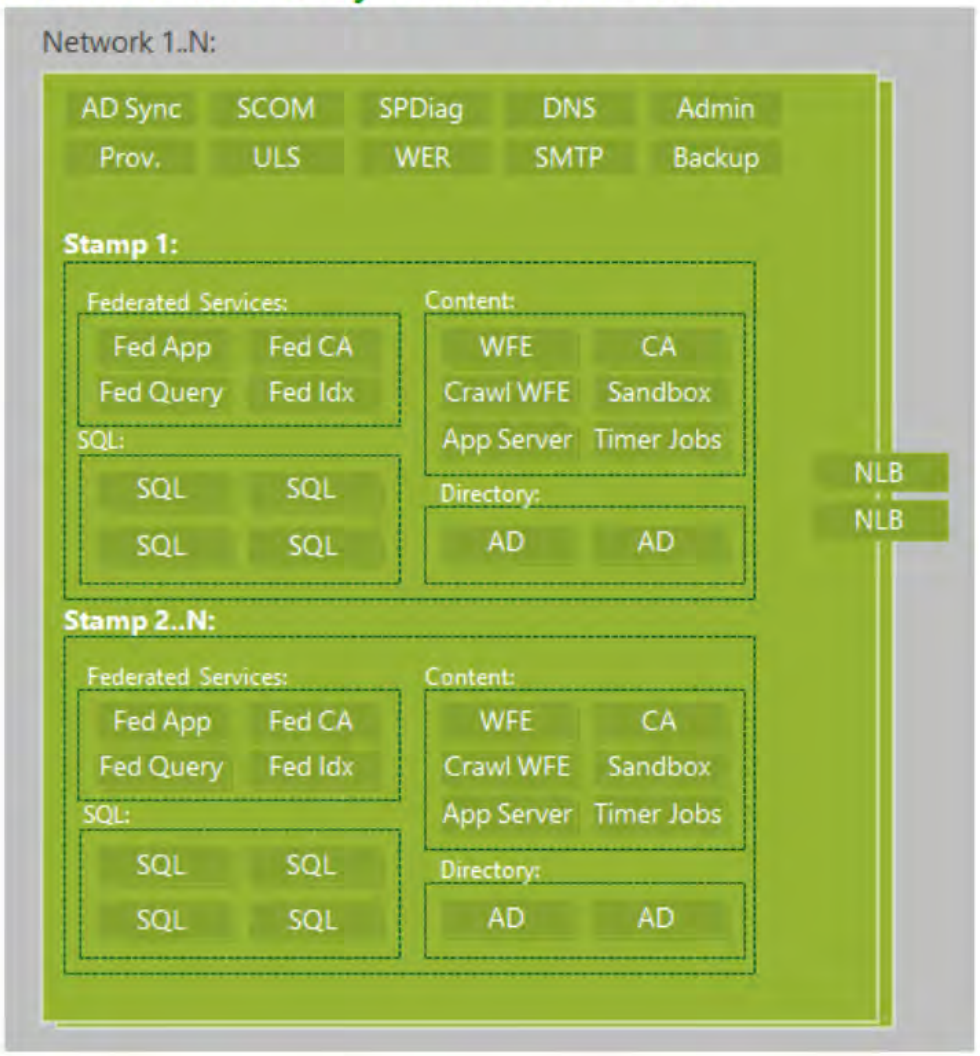
Each datacenter has two or more stamps per SPO environment for high-availability.



Looking at the representation of an individual stamp, you might think it's only 16 servers.

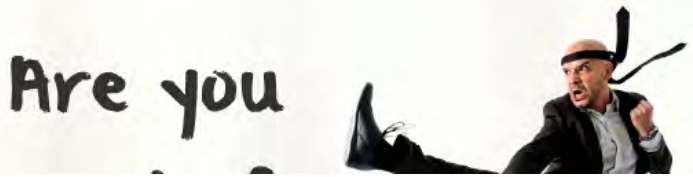
s is a stamp
several of them)

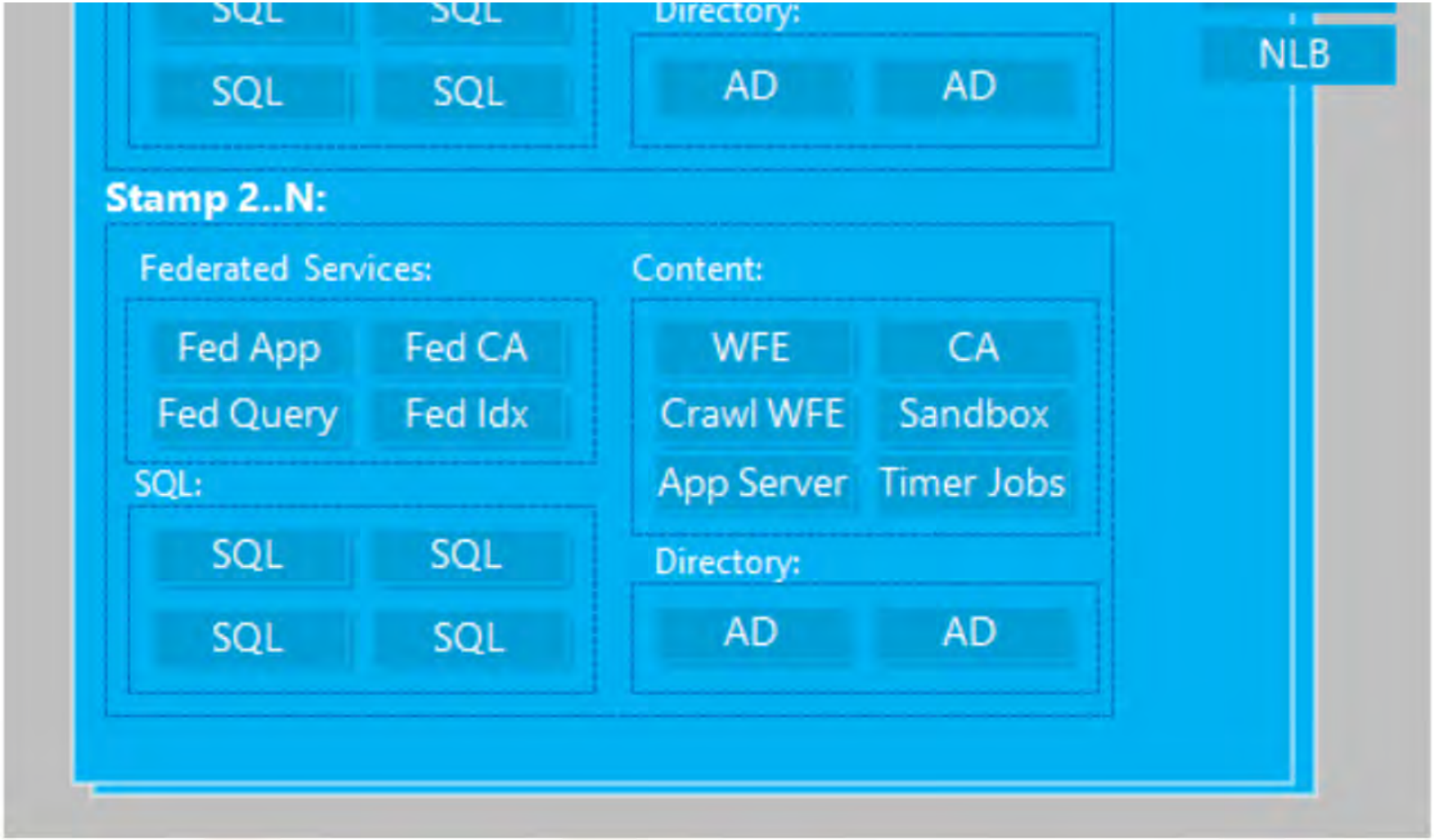
Disaster Recovery Datacenter 1..N:



- Grid Manager
- Global Directory
- Tenant Admin (UI)
- Commerce backend
- DNS (multiple)
- OrgID Auth, Svc.
- Incident Management
- Azure (Windows/SQL)
- CDN Services

Additional stamps exist in a different region for redundancy and failover.



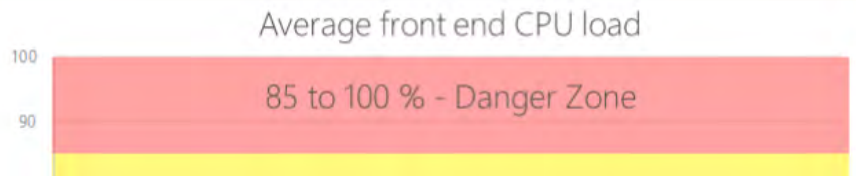


Looking at the representation of an individual stamp, you might think it's only 16 servers.

Are you ready for the kicker?

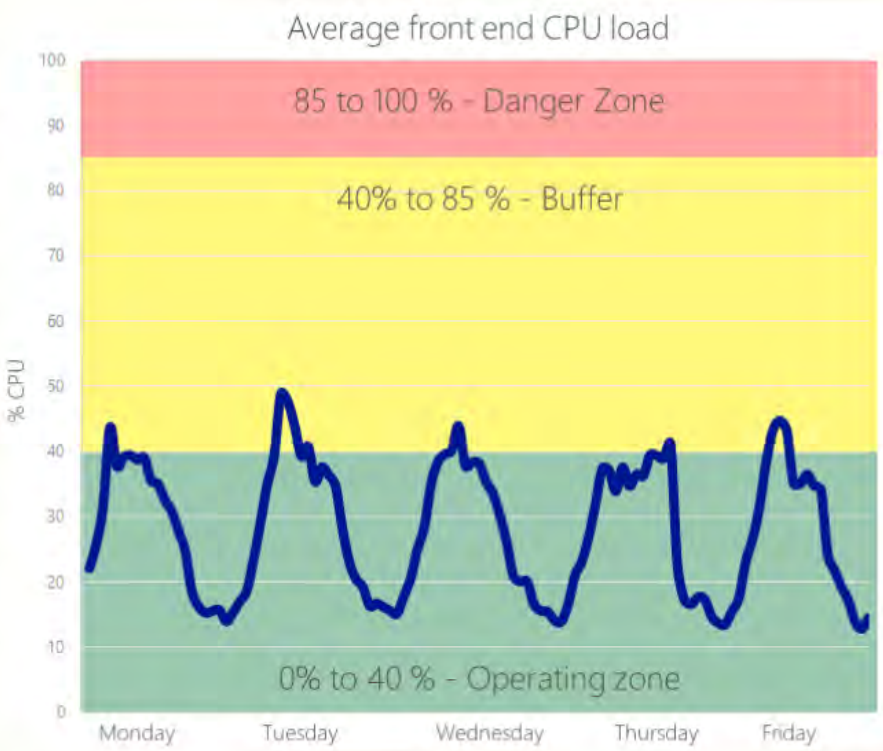
The exact number of servers in a SharePoint Online stamp is variable.

The number of servers per



The exact number of servers in a SharePoint Online stamp is variable.

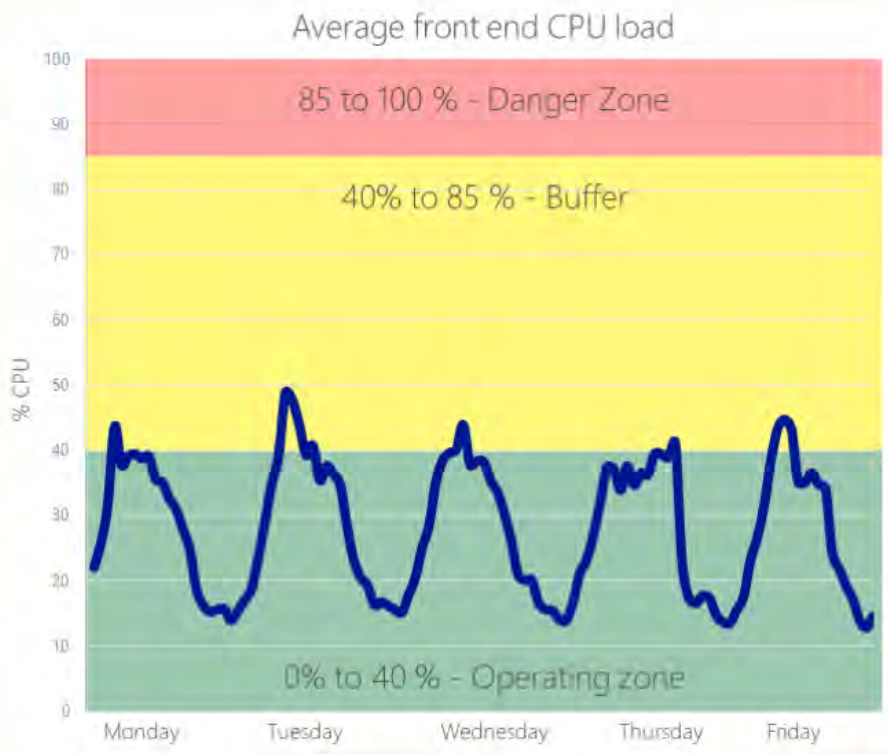
The number of servers per stamp varies because the server count is adjusted based on average front-end CPU load.



- If load rises above 40%, additional servers are automatically provisioned and added to the stamp

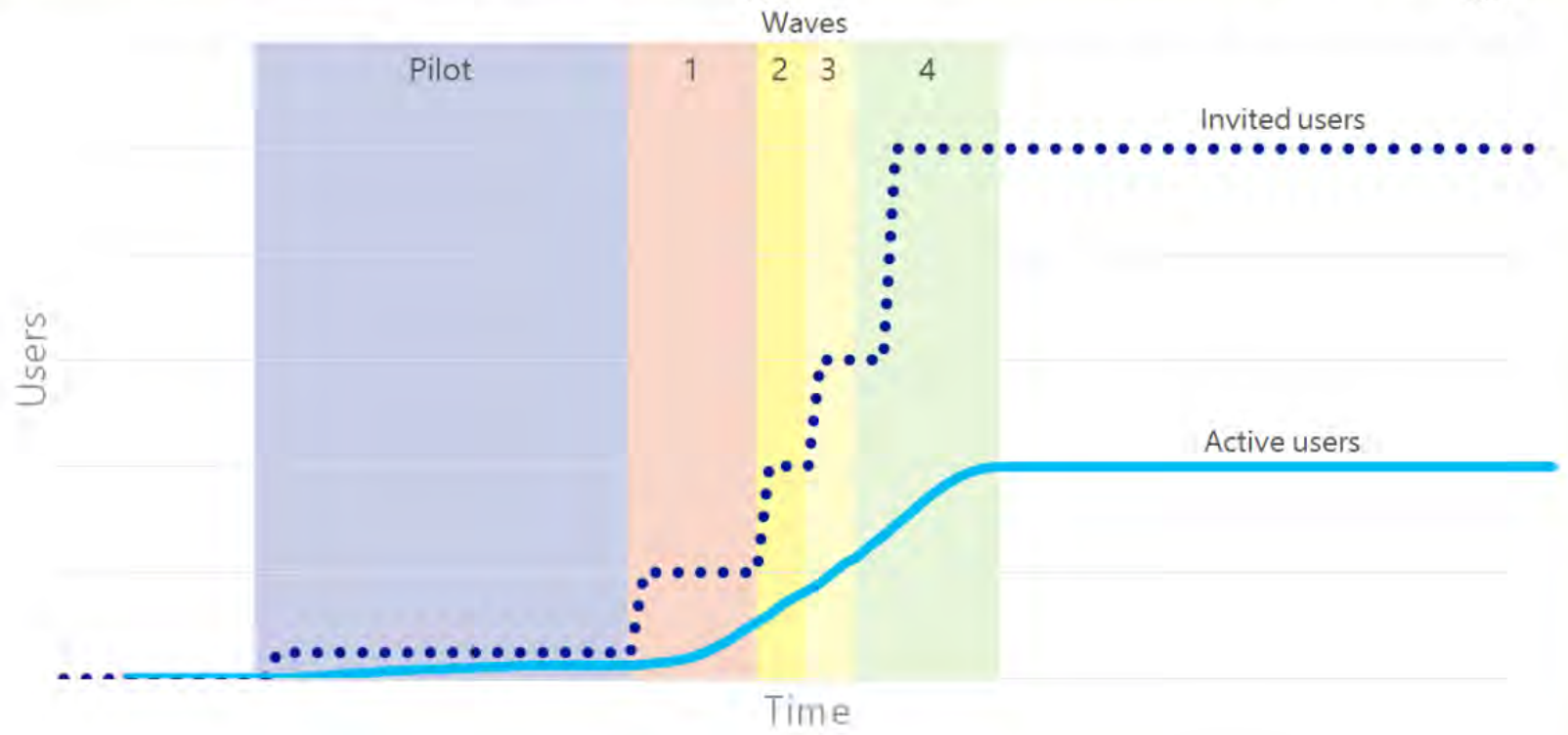
SharePoint Online stamp is variable.

The number of servers per stamp varies because the server count is adjusted based on average front-end CPU load.



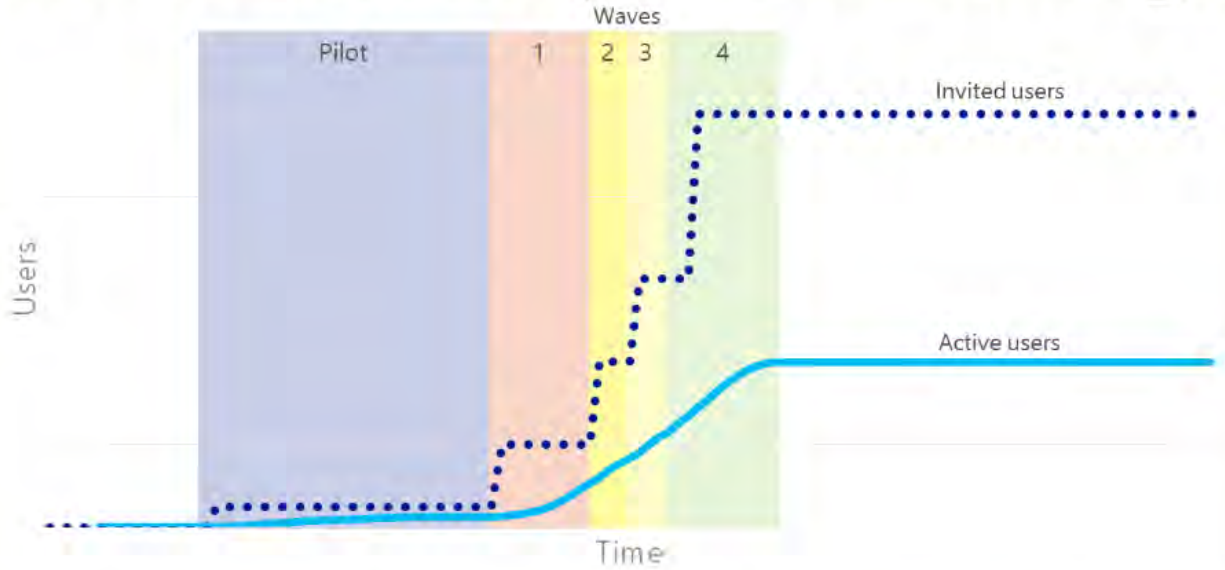
- If load rises above 40%, additional servers are automatically provisioned and added to the stamp.
- If load drops, servers are decommissioned.

Adding and removing is not an instantaneous process, though.



an control load (for example, gradually increasing the nur
users over time when you roll out a site), you should. It w

Adding and removing is not an instantaneous process, though.



If you can control load (for example, gradually increasing the number of invited users over time when you roll out a site), you should. It will give the provisioning system time to adjust/compensate for growing load.

If you remember only one thing in this discussion of stamps and elastic capacity, please let it be this one point ...

Load testing is futile.

astic nature of a stamp, there's really no way to effectively load t

please let it be this one point ...

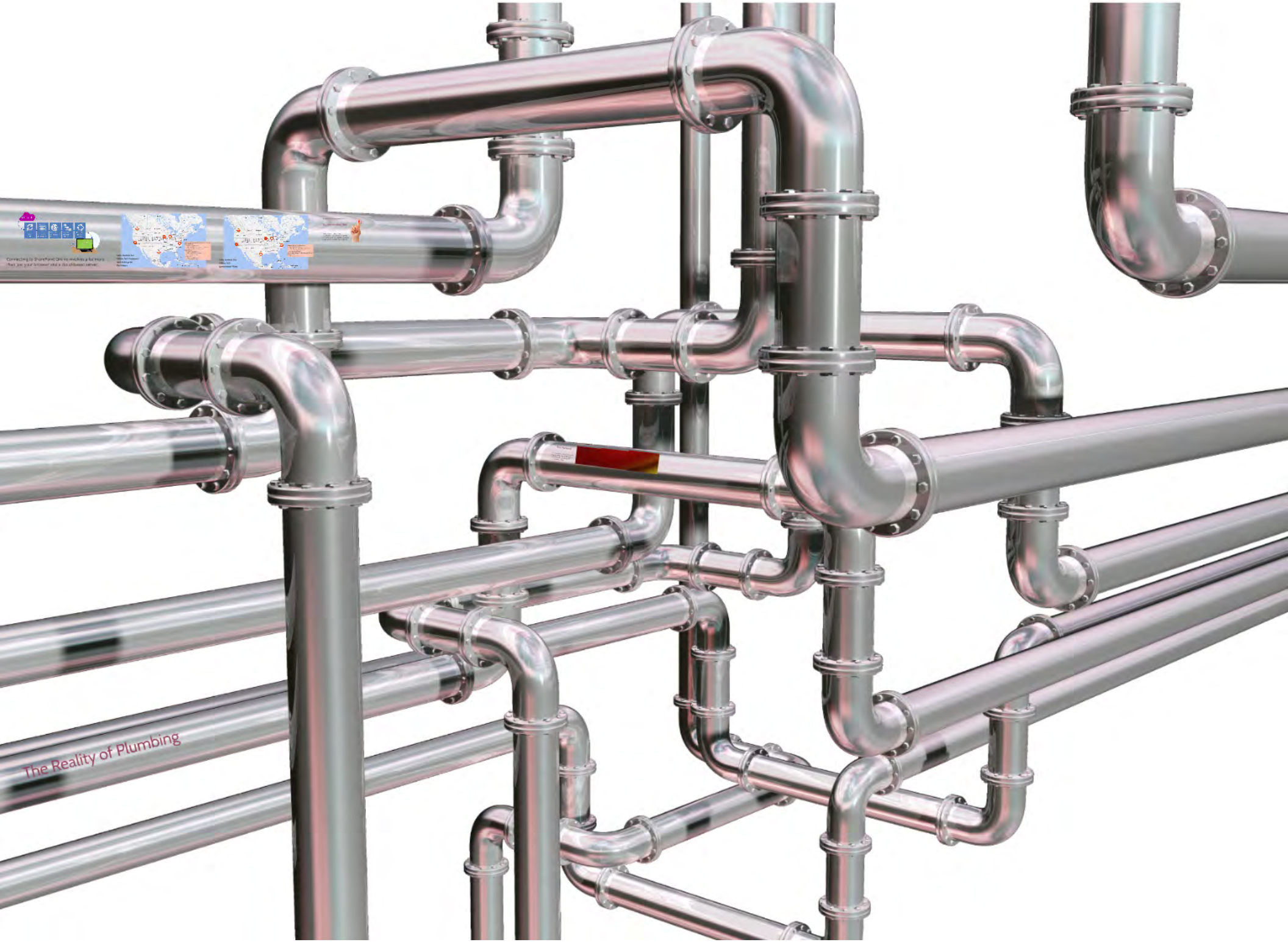
g the number of
uld. It will give
growing load.

Load testing is futile.

- Given the elastic nature of a stamp, there's really no way to effectively load test SPO. Any numbers you get or produce are essentially meaningless in the grand scheme of things.
- Rather than load testing, focus instead on the items we're going to cover in the rest of this presentation. They'll help you avoid poorly performing pages and sites.


Welcome to the farm!


The Reality of Plumbing


The Reality of Plumbing

We don't have time to cover all the plumbing in this session ...

So, remember this

If you've spent a lot of time

We don't have time to cover all the plumbing in this session ...

So, remember this:

If you've spent a lot of time troubleshooting in SharePoint Online (to little or no effect), maybe you should zoom out and consider the network.

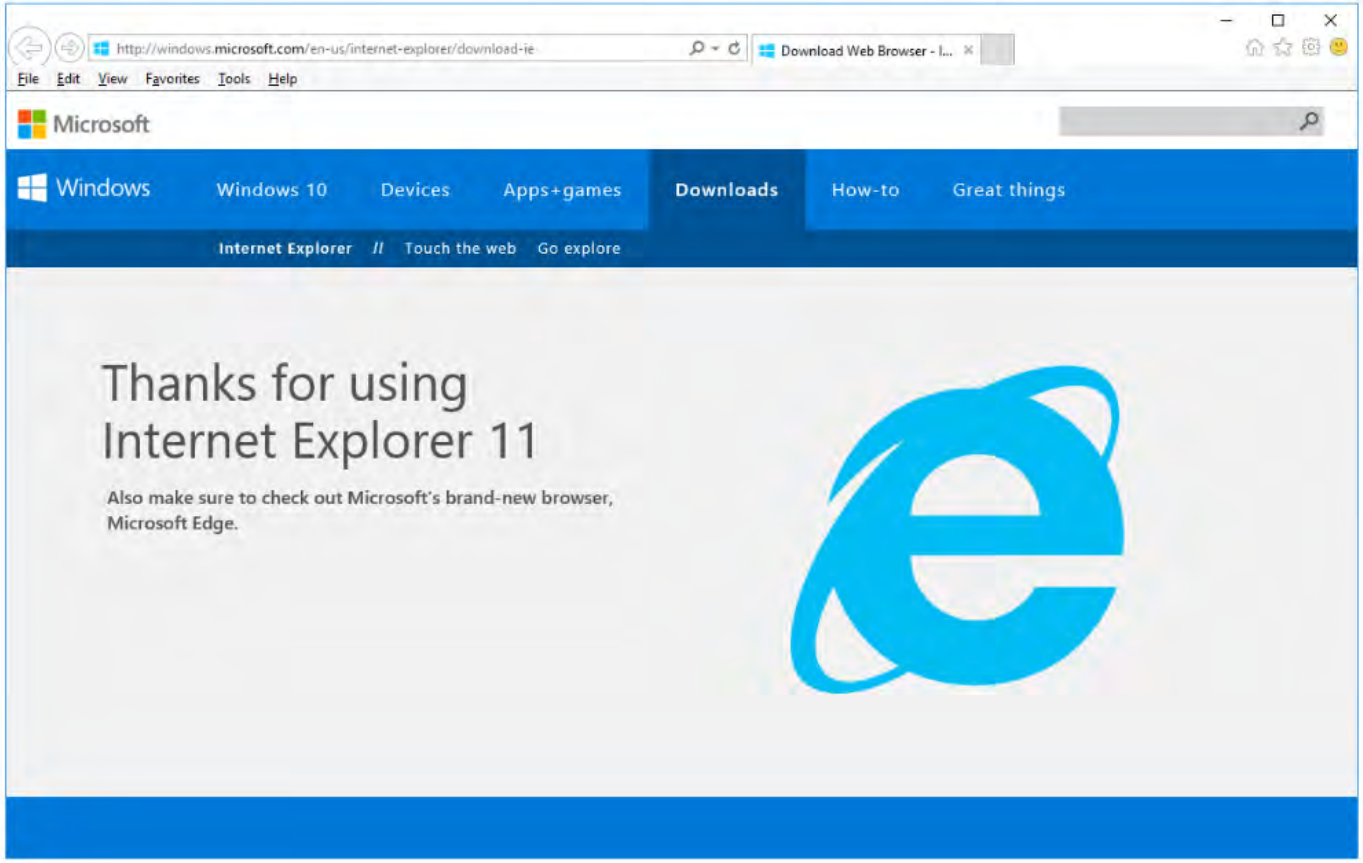
Okay, so it really feels like there is a tangible performance problem.

Okay, so it really feels like there is a tangible performance problem.

How do you prove it objectively?

Meet your primary diagnostic tool.

In all likelihood, you already have it on your system.



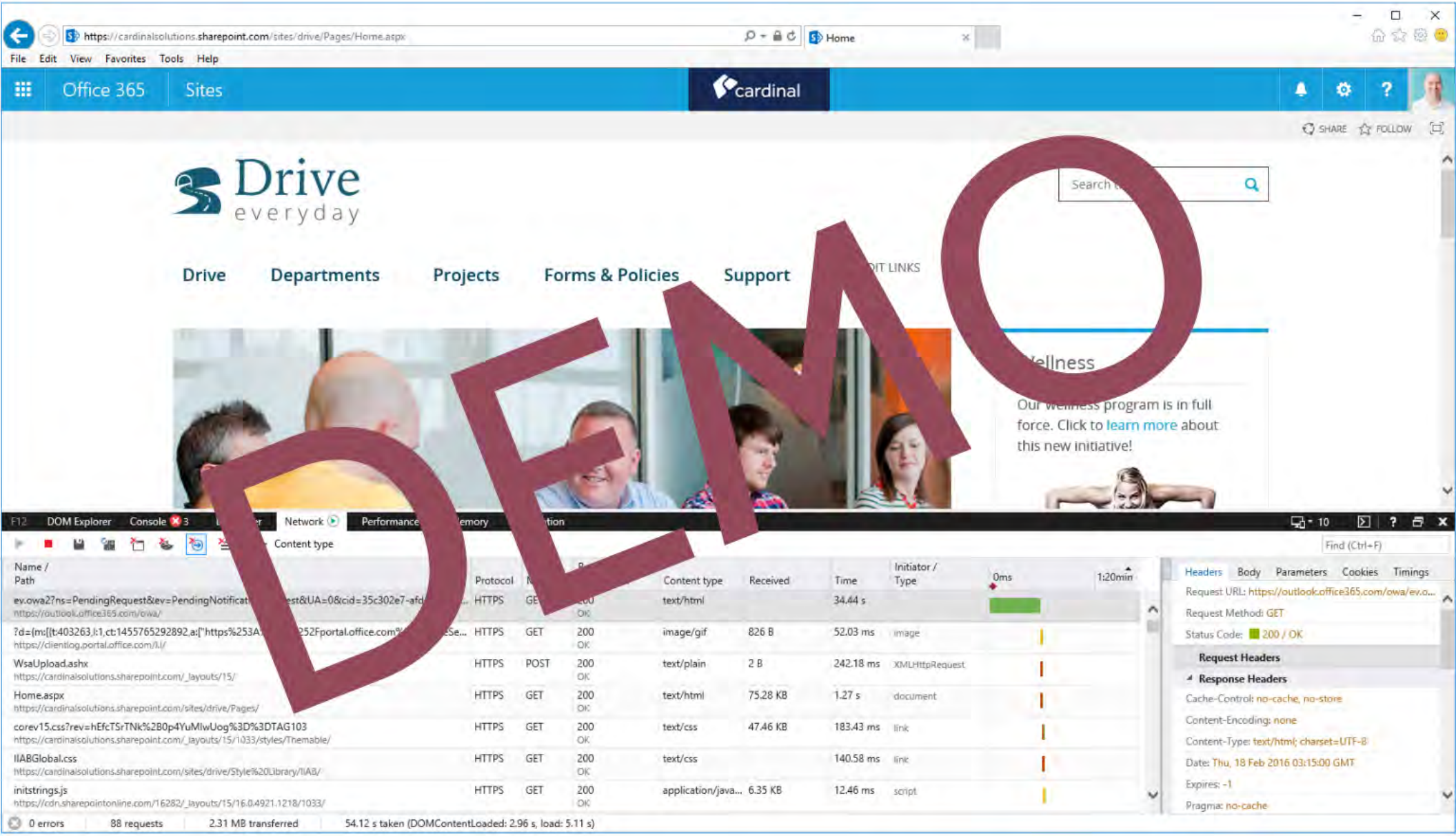
Hello, Internet Explorer!

Um ... you're kidding, right?

None!

Um ... you're kidding, right?

Nope!


https://cardinalsolutions.sharepoint.com/sites/drive/Pages/Home.aspx
Home
File Edit View Favorites Tools Help
Office 365
Sites
cardinal
SHARE
FOLLOW
Drive
everyday
Drive
Departments
Projects
Forms & Policies
Support
Our wellness program is in full force. Click to learn more about this new initiative!
DEMO
F12 DOM Explorer Console Network Performance
Content type
Name / Path
Protocol
Content type
Received
Time
Initiator / Type
Headers Body Parameters Cookies Timings
Request URL: https://outlook.office365.com/owa/ev.o...
Request Method: GET
Status Code: 200 / OK
Request Headers
Response Headers
Cache-Control: no-cache, no-store
Content-Encoding: none
Content-Type: text/html; charset=UTF-8
Date: Thu, 18 Feb 2016 03:15:00 GMT
Expires: -1
Pragma: no-cache
0 errors
88 requests
2.31 MB transferred
54.12 s taken (DOMContentLoaded: 2.96 s, load: 5.11 s)

* may be due to routing issues (as in "number of hops")
* plenty of other possibilities

Demo Takeaways

HTTP Response Headers

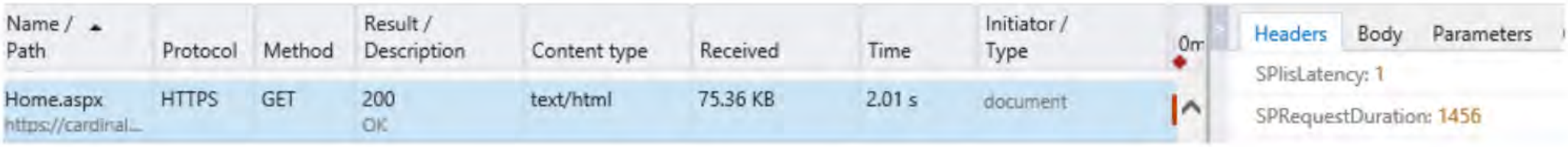
- SPIisLatency
- SPRequestDuration
- X-SharePointHealthScore

waiting on server - generally zero or near zero

time spent processing on server (in ms) - ideally low

0 to 10 (you want 0)

Generally speaking ...



Name / Path	Protocol	Method	Result / Description	Content type	Received	Time	Initiator / Type
Home.aspx https://cardinal...	HTTPS	GET	200 OK	text/html	75.36 KB	2.01 s	document

Headers | Body | Parameters

SPIisLatency: 1

SPRequestDuration: 1456

Time - (SPRequestDuration + SPIisLatency) = "time lost elsewhere"

* potential network latency
* may be due to routing issues (as in "number of hops")
* plenty of other possibilities

Demo Takeaways

HTTP Response

waiting on server - generally zero or

time spent processing on

So, you've concluded that your pages are slow and you have the data to prove it!

SPIisLatency is low, and
X-SharePointHealthScore is low,

but ...

So, you've concluded that your pages are slow and you have the data to prove it!

SPIisLatency is low, and
X-SharePointHealthScore is low,

but ...

SPRequestDuration is crazy high (e.g., 9000 ms)!

Repeat after me ...

"The problem probably isn't

Repeat after me ...

"The problem probably isn't SharePoint Online. It's my site."

So, who's to blame?

In all likelihood

So, who's to blame?

In all likelihood:
blame the
lousy* devs.

*Note: not all devs are lousy devs. Just the ones who cause performance problems and knee-jerk into blaming Microsoft and SharePoint Online.

So, who's to blame?

In all likelihood:
blame the
lousy* devs.

*Note: not all devs are lousy devs. Just the ones who cause performance problems and knee-jerk into blaming Microsoft and SharePoint Online.

ones who cause performance
ft and SharePoint Online.

- Compare processing and respon
  SharePoint site or page.

- In the majority of poor performance scenarios, a combination of UI/UX , client-side code additions, and questionable customization/deployment mechanisms are to blame.

- Microsoft has indicated that the slowest 1% of pages in SPO take more than 5,000ms to load - again, usually due to customizations.

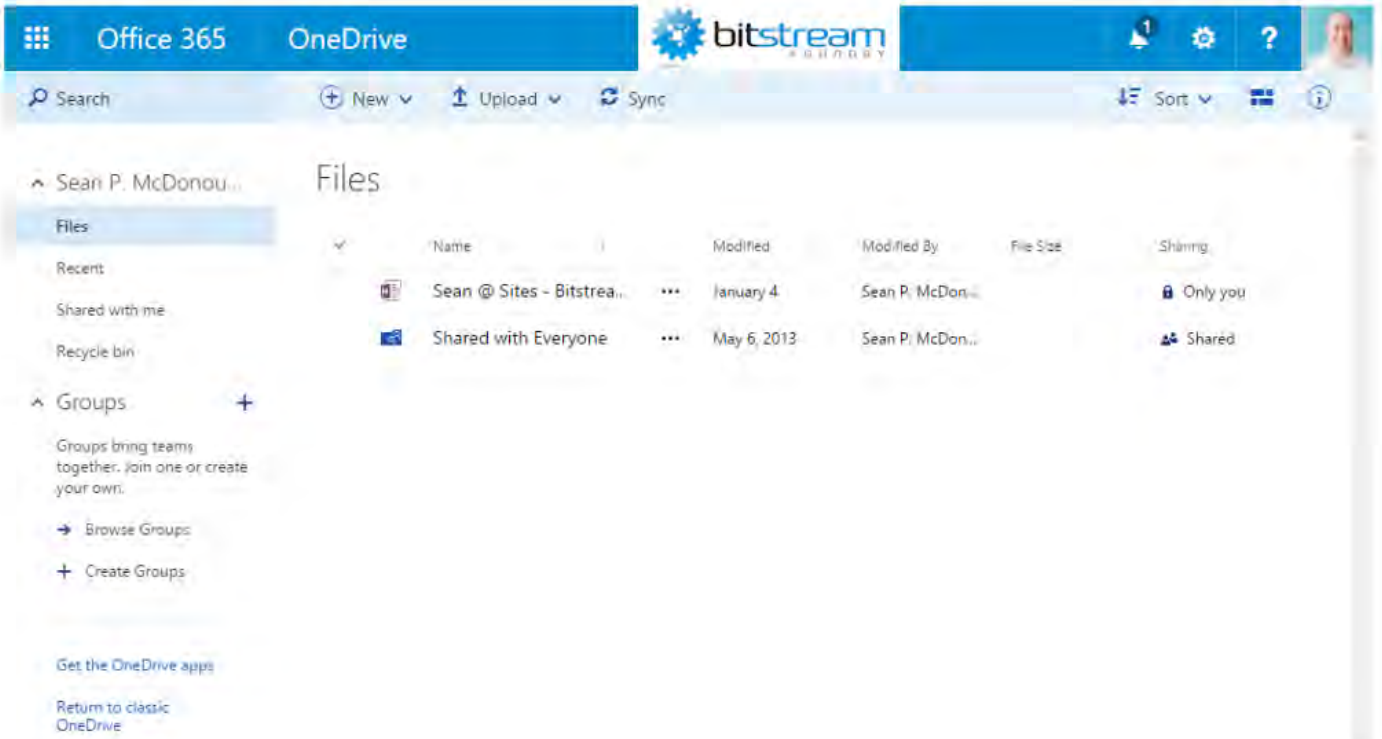
- Compare processing and response times to your problematic SharePoint site or page.

- In the majority of poor performance scenarios, a combination of UI/UX , client-side code additions, and questionable customization/deployment mechanisms are to blame.

- Microsoft has indicated that the slowest 1% of pages in SPO take more than 5,000ms to load - again, usually due to customizations.

Don't believe me?

Collect the data and validate for yourself!



- Profile your OneDrive for Business page (it's in your MySite).
- Compare processing and response times to your problematic SharePoint site or page.

- In the majority of poor performance scenarios, a

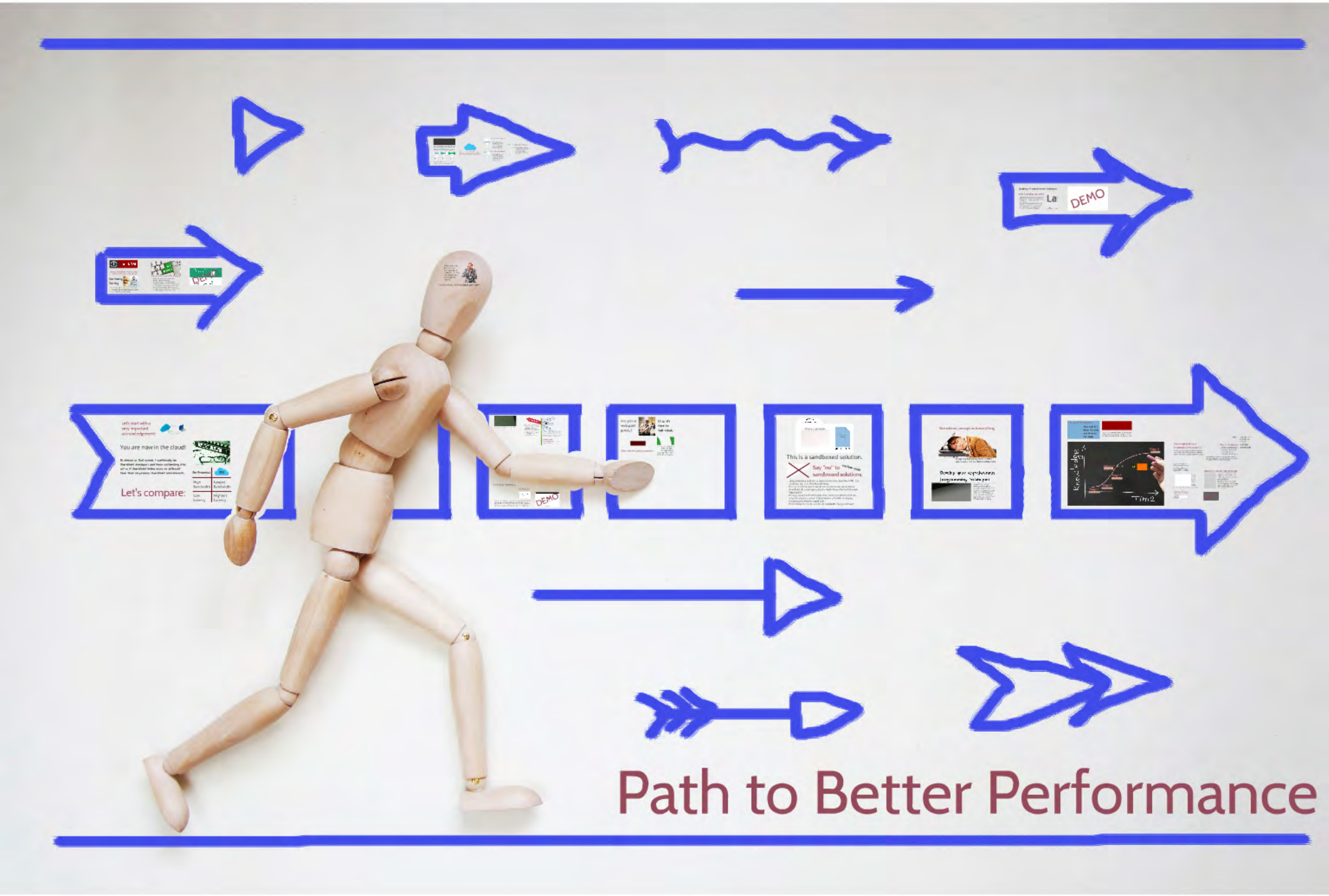
Don't believe me?

"Okay, yeah - my OneDrive for Business page is really fast ... but my SharePoint pages are completely choking."

u're probably thinking "What can I do

"Okay, yeah - my OneDrive for Business page is really fast ... but my SharePoint pages are completely choking."

You're probably thinking **"What can I do?"**


DEMO
You are now in the cloud!
Let's compare:
DEMO
This is a sandboxed solution.
Say 'no' to
sandboxed solutions
Path to Better Performance

Let's start with a very important acknowledgement:

Let's start with a very important acknowledgement:

You are now in the cloud!

As obvious as that sounds, I continually see

acknowledgement:

You are now in the cloud!

As obvious as that sounds, I continually see SharePoint developers and those customizing sites act as if SharePoint Online were no different than their on-premises SharePoint environments.

very important
acknowledgement:

You are now in the cloud!

As obvious as that sounds, I continually see SharePoint developers and those customizing sites act as if SharePoint Online were no different than their on-premises SharePoint environments.

Let's compare:

On-Premises	SPO
High Bandwidth	Low(er) Bandwidth
Low Latency	High(er) Latency

Failing to acknowledge the "we're in the cloud now" reality leads to a problem I simply call ...

Too Many

Failing to acknowledge the "we're in the cloud now" reality leads to a problem I simply call ...

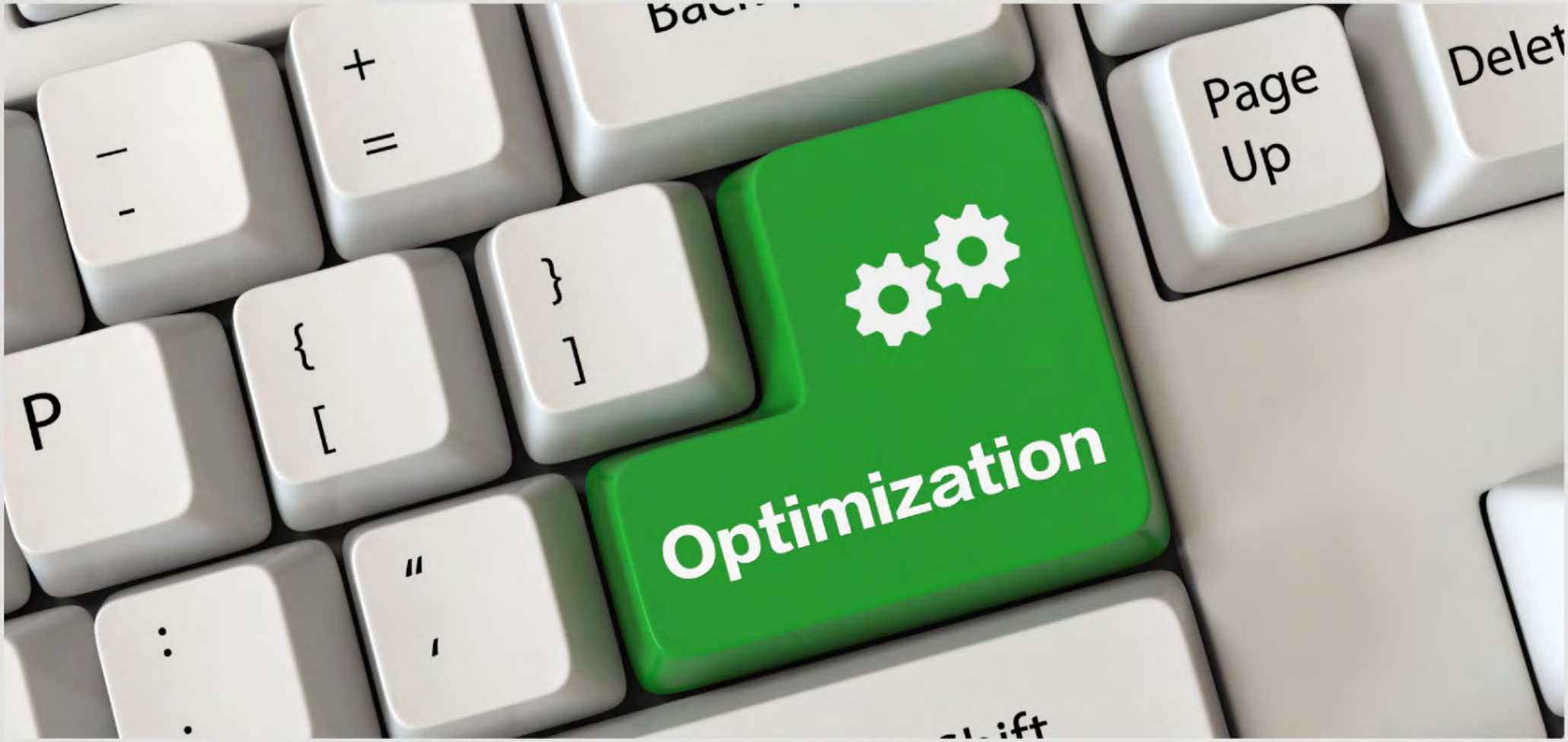
Too Many, Too Big

- Too many calls are made to the server.

now" reality leads to a problem I simply call ...

Too Many, Too Big

- Too many calls are made to the server.
- Too many files are referenced on pages.
- The files in-use are too large.

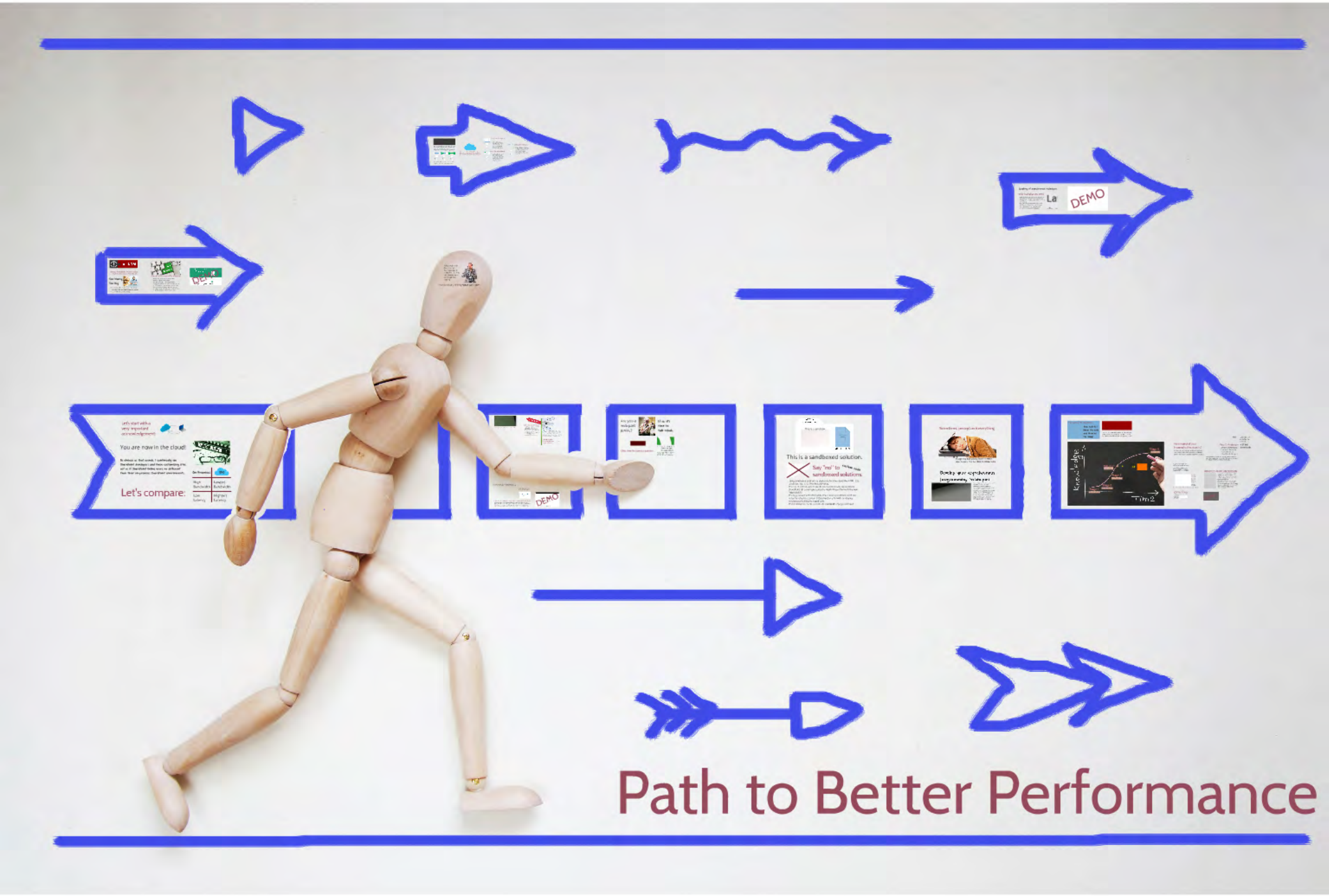


Consider one or more of the following:

- Minify files, especially JavaScript files.
- Resize images to usage sizes

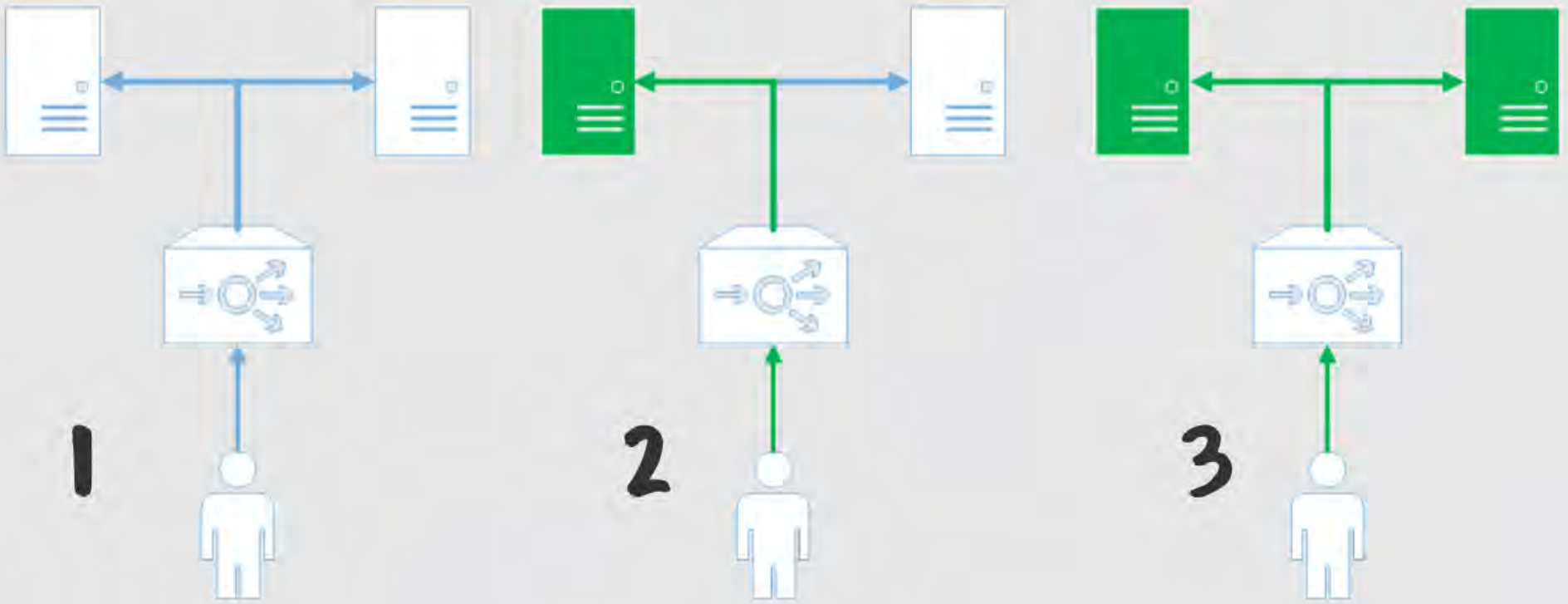
Consider one or more of the following:

- Minify files, especially JavaScript files.
- Resize images to usage sizes.
- Compress images (more) aggressively.
- Use sprite sheets to reduce the actual number of HTTP requests needed to retrieve images.
- Use SharePoint's Image Rendition service.
- Leverage a toolkit like Font Awesome in place of individual icons and associated files.

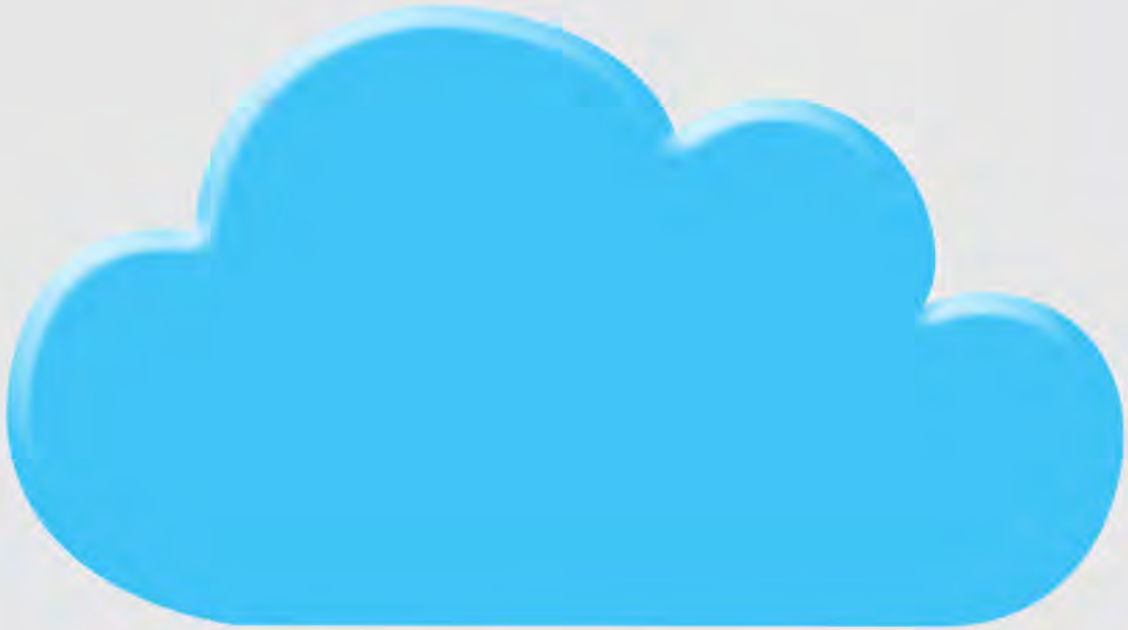

DEMO
You are now in the cloud!
Let's compare:
DEMO
This is a sandboxed solution.
Say 'no' to
sandboxed solutions
Path to Better Performance

Conventional wisdom says caching is good.

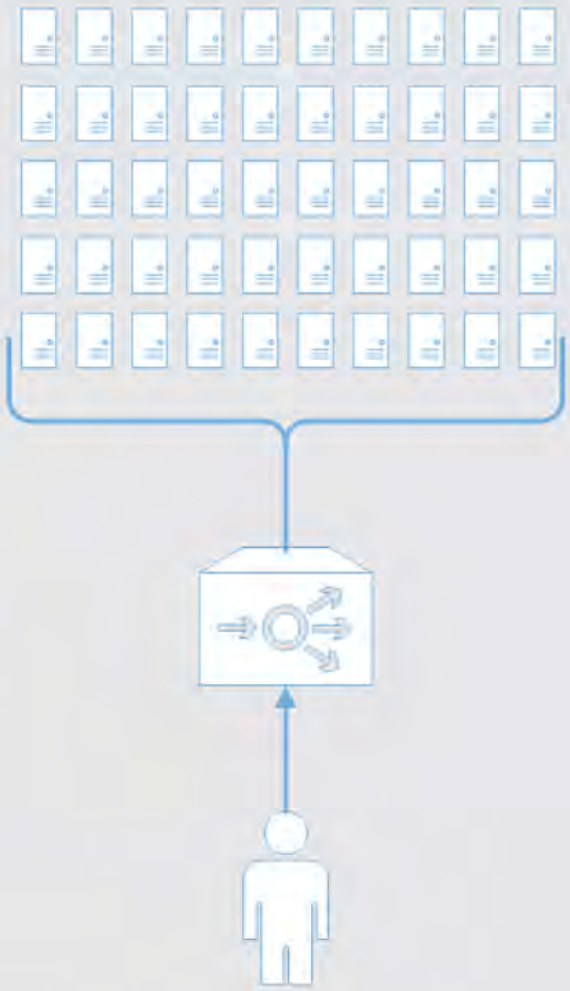
says caching is good.



After just a few requests, the on-premises Object Cache can be "ready for action."



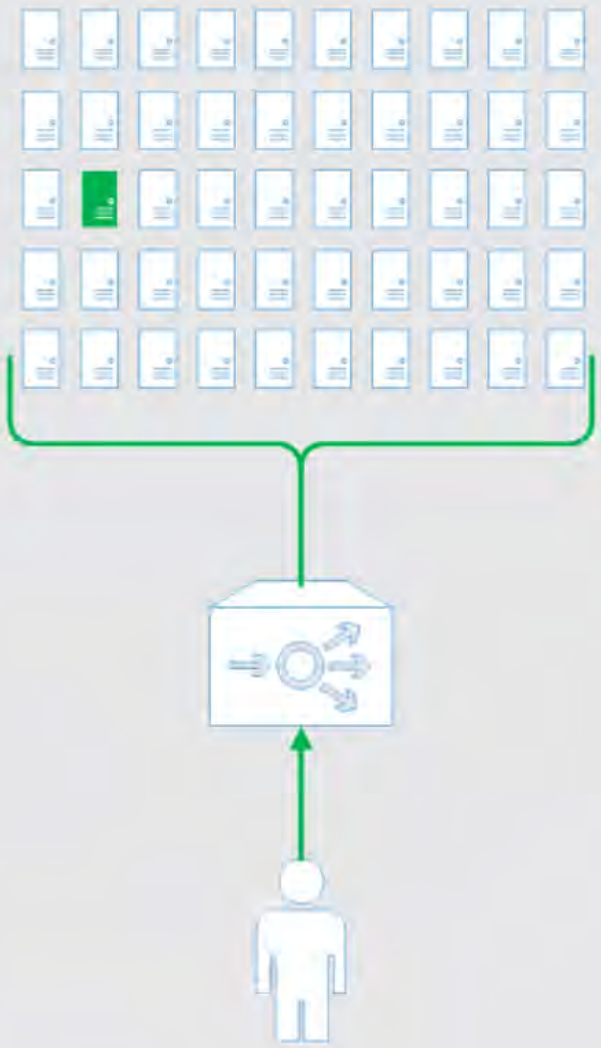
In the cloud, the caching equation (for per-server memory-based caches like the Object Cache) works out a bit differently.



User's Initial Request

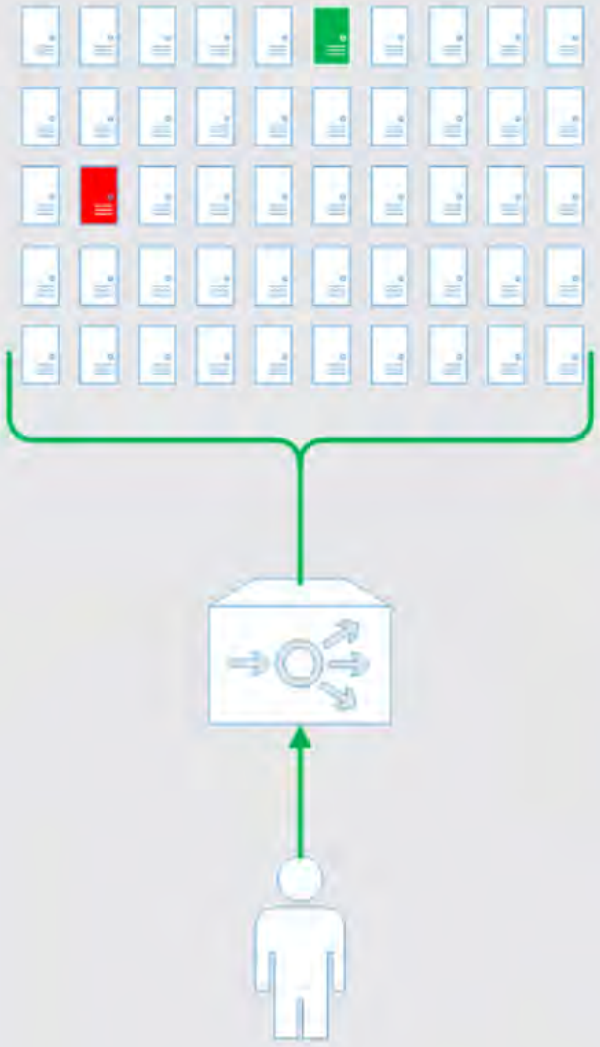
- First thing to note: the number of WFEs tends to be *much* higher in the cloud versus on-premises.

User's Second Request



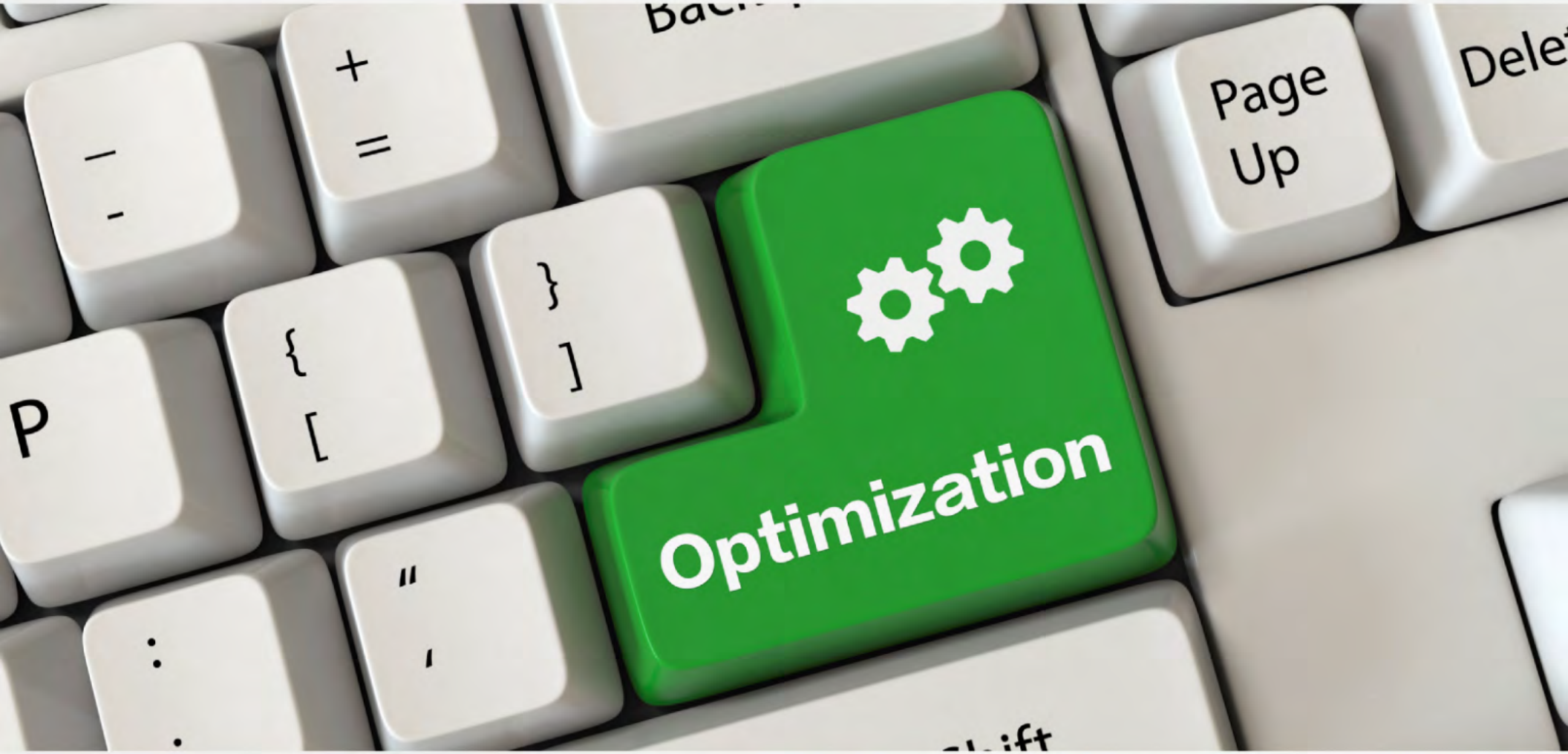
User's Second Request

- No affinity is in use, so the chance of a user hitting the same server again is dramatically less than the on-premises scenario.



Subsequent Requests

- Same reduced chance of hitting the WFE last visited
- Memory pressure causes much more frequent cache ejections versus on-premises.



Two significant adjustments can be made.

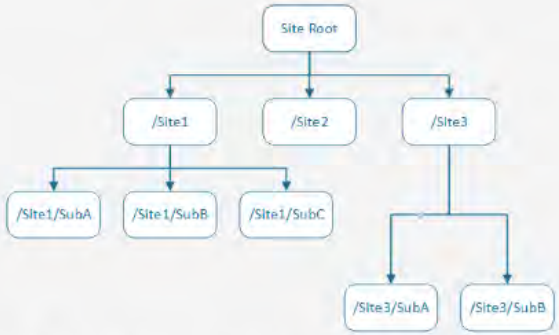
* These sitemaps are then stored in the Object

Navigation style has a huge impact on performance.

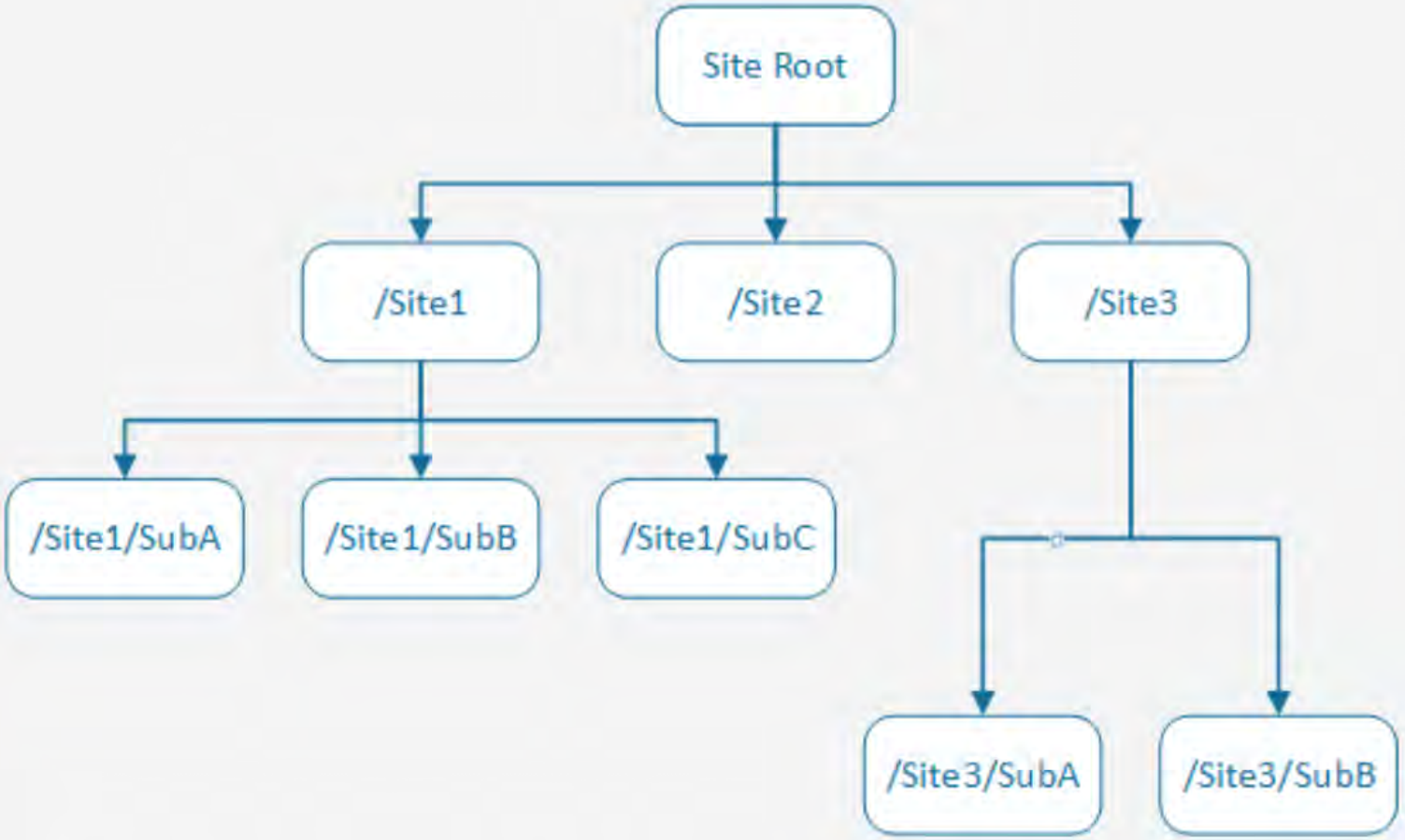
Using structural navigation is the default (but poor) choice for complex site hierarchies in the cloud.

* building each site node generates roughly 8 SQL Server round trips
* These sitemaps are then stored in the Object Cache on WFEs

Navigation style has a huge impact on performance.



8 site nodes/~64 SQL calls



8 site nodes/~64 SQL calls

Better Options for Navigation

- Managed navigation (i.e., using a term set to drive navigational structures) can significantly improve page performance.
  note: the SharePoint Server Publishing Infrastructure site collection Feature must be enabled to use managed navigation
- Search-driven navigation leverages SharePoint's Search index and the process of client-side navigational rendering to dramatically speed things up.
  note: implementation is non-trivial and less customizable

Using structural navigation is the default (but poor) choice for complex site hierarchies in the cloud.

* building each site node generates roughly 8 SQL Server round trips
* These sitemaps are then stored in the Object Cache on WFEs

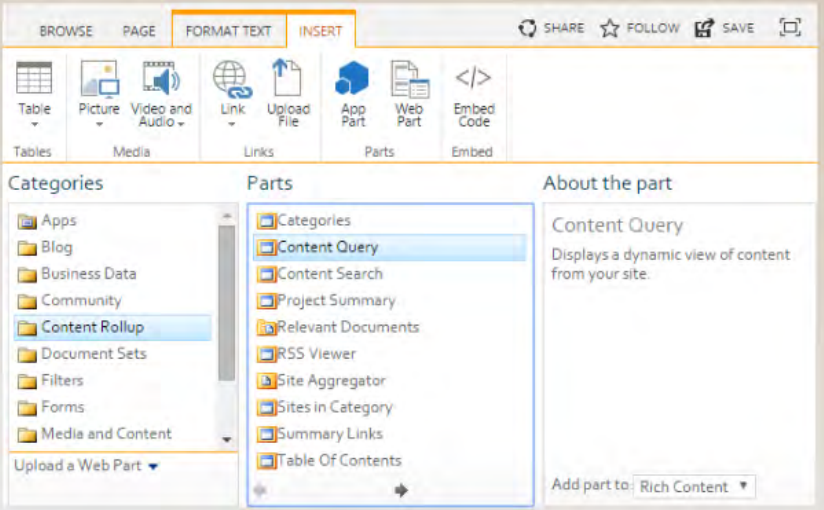
Navigation style

As was pointed-out in the navigational scenario,
Search can be used to boost performance significantly.

In the cloud, the CQWP can cause some signif

ut in the navigational scenario,
ed to boost performance significantly.

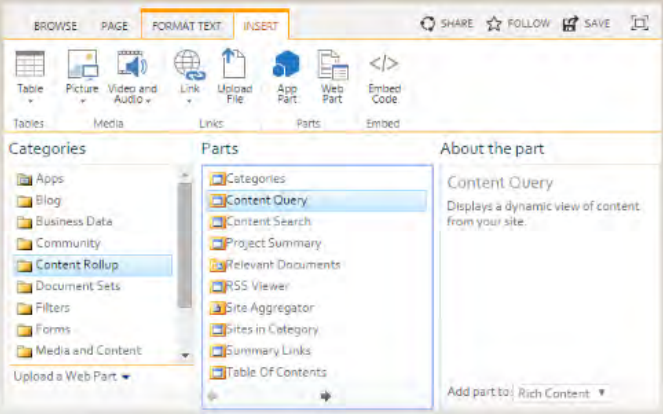
Do you like the Content Query Web Part (CQWP)?



the CQWP can cause some significant performance hits.

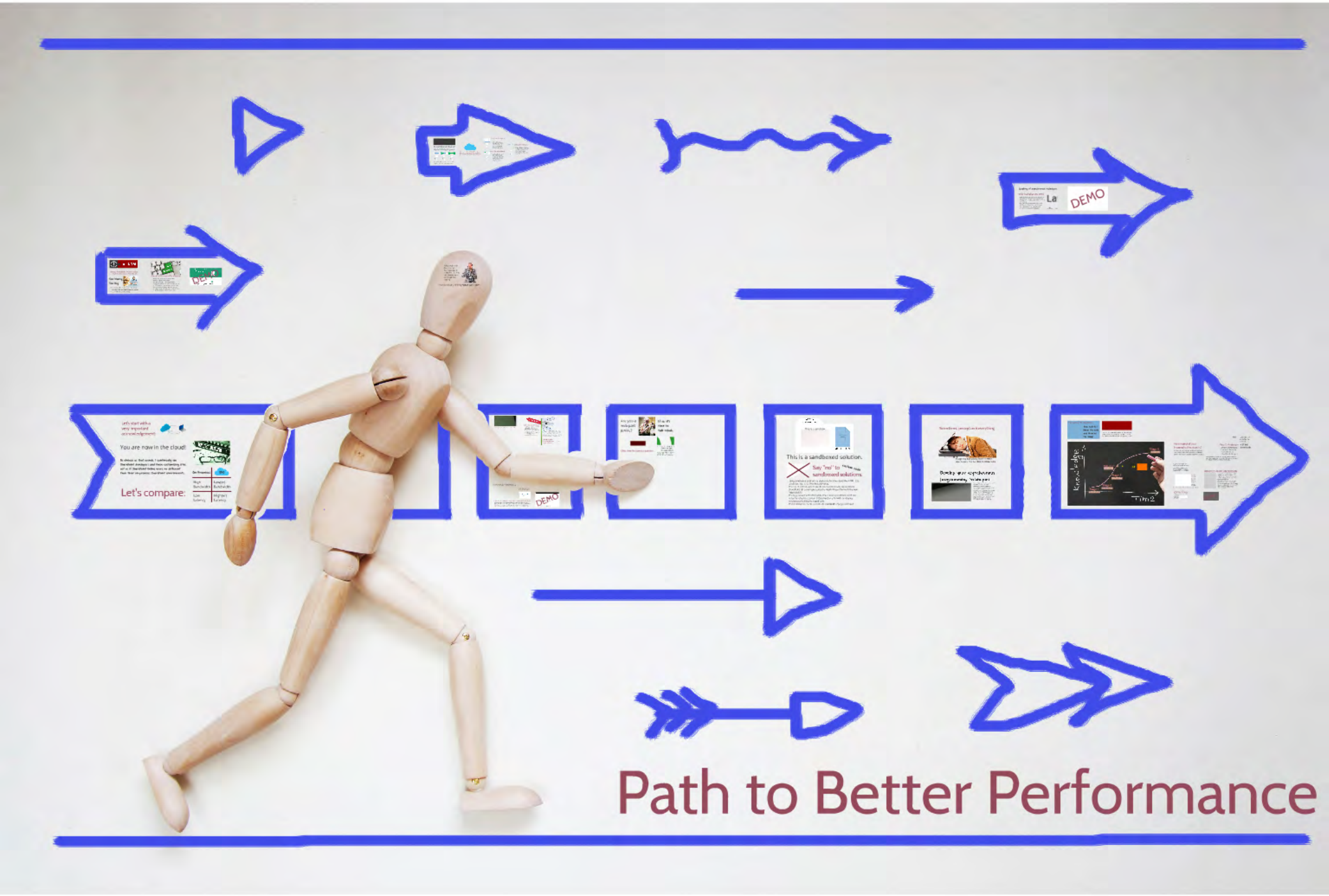
As was pointed-out in the navigational scenario,
Search can be used to boost performance significantly.

Do you like the Content Query Web Part (CQWP)?



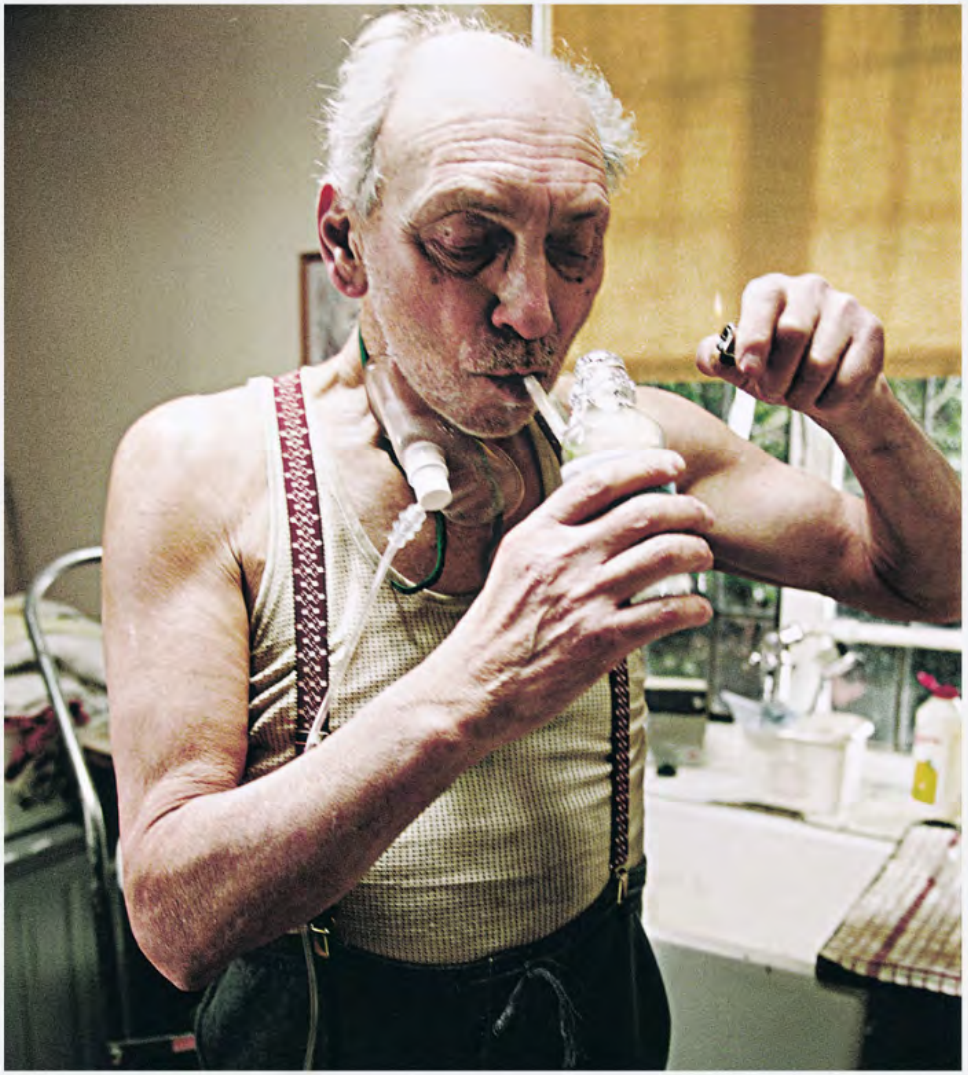
In the cloud, the CQWP can cause some significant performance hits.

- The CQWP performs expensive cross-list and cross-site queries at run-time.
- The CQWP relies on the Object Cache to store results for acceptable performance.
- The Content Search Web Part (CSWP) provides options that are similar to the CQWP (and in a number of ways, more powerful) and uses Search so it's FAST!

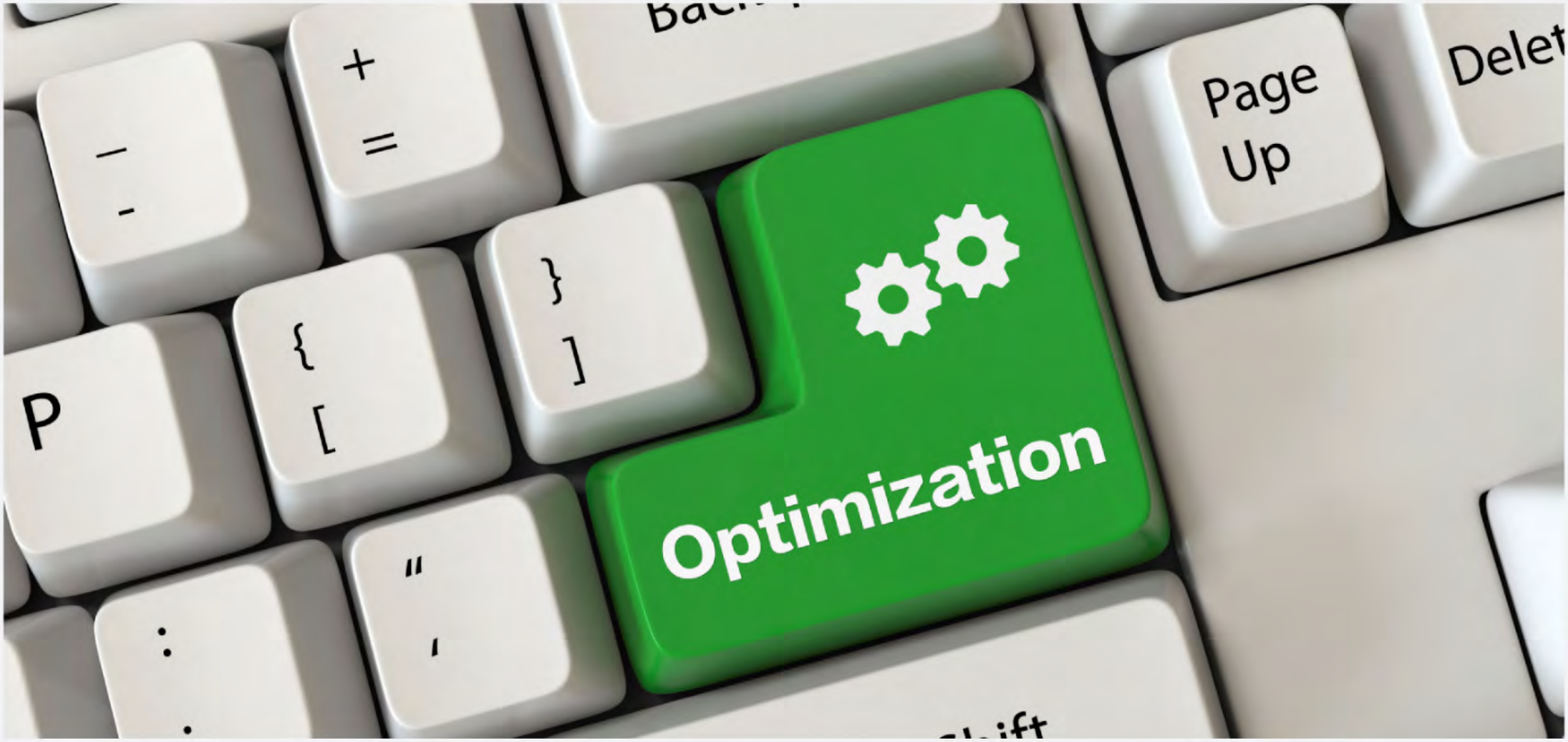

DEMO
You are now in the cloud!
Let's compare:
DEMO
This is a sandboxed solution.
Say 'no' to
sandboxed solutions
Path to Better Performance

Okay, time for a serious question ...

Are you a web part junkie?

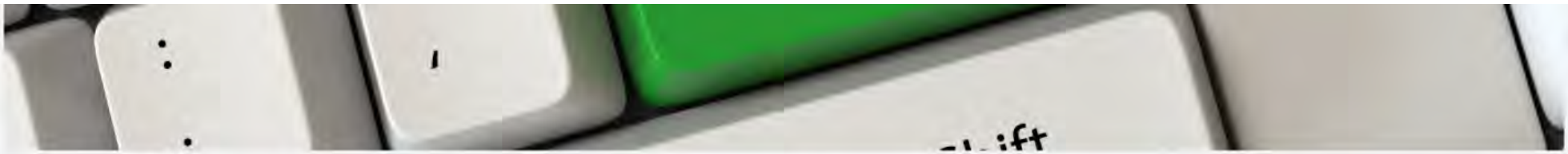


If so, it's time to talk rehab.



What's the alternative to freebasing web parts?

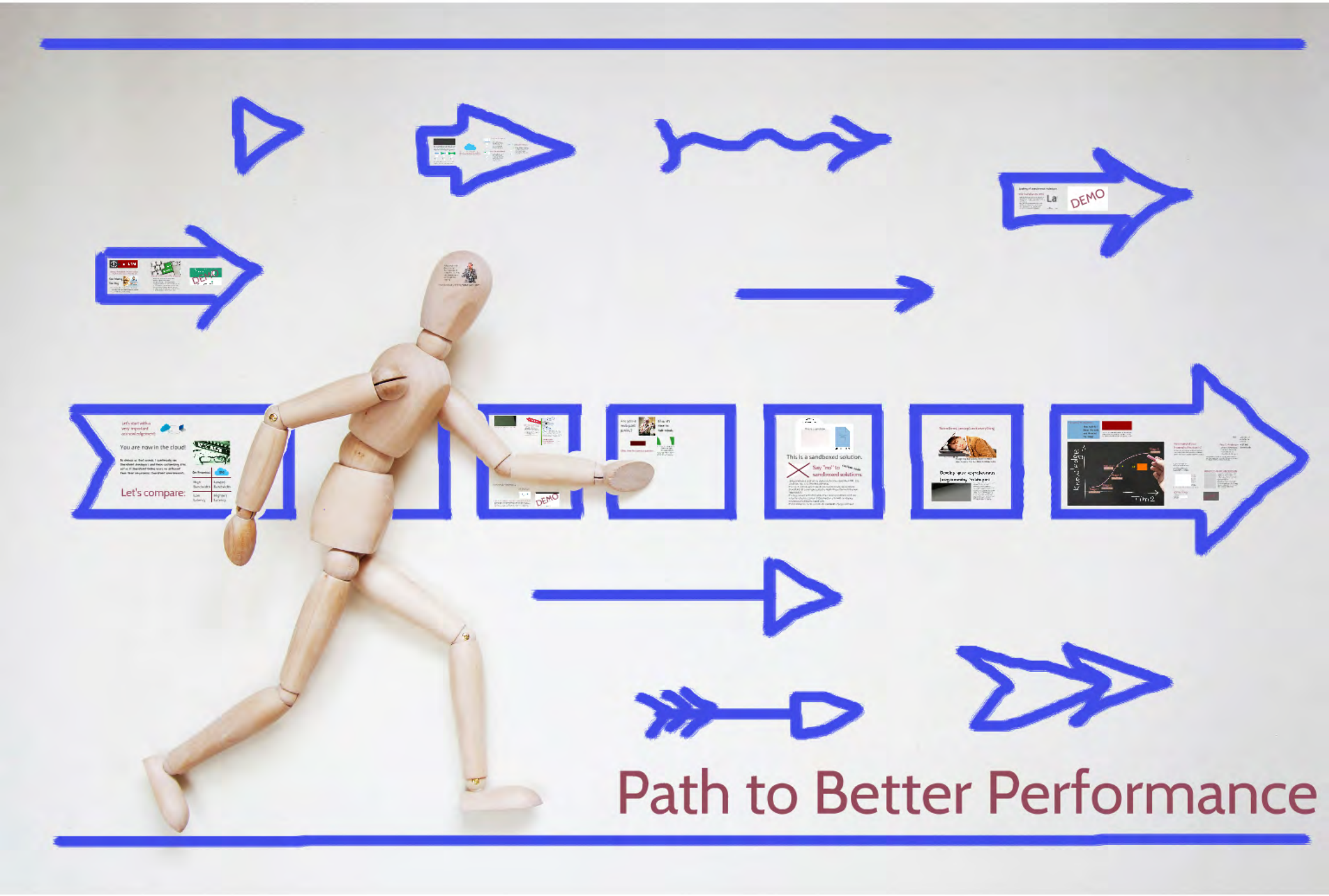
There's no single



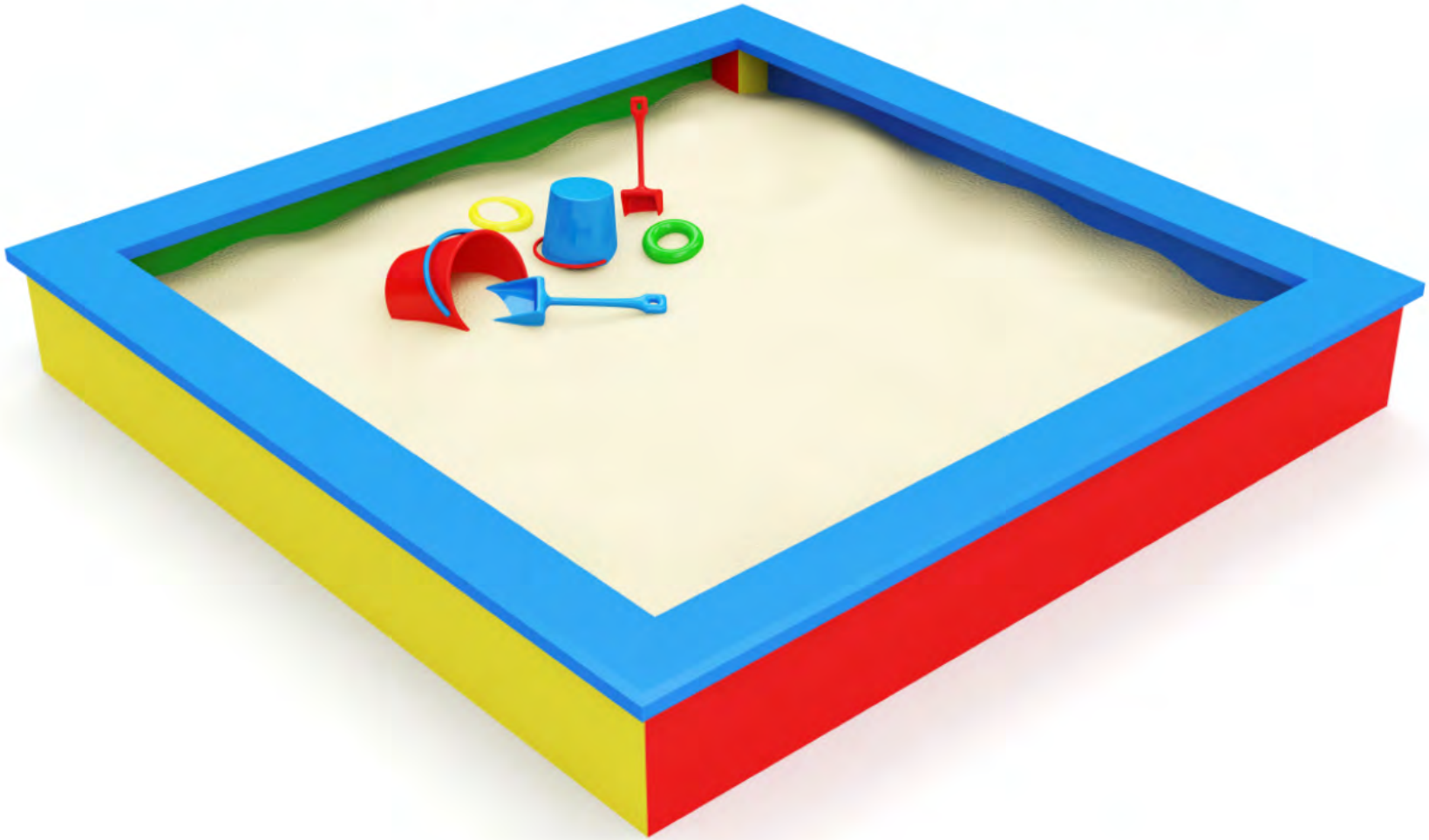
What's the alternative to freebasing web parts?

There's no single (or simple) answer.

Generally speaking, consider leveraging client-side code (JavaScript) and asynchronous techniques - both of which we'll discuss soon.

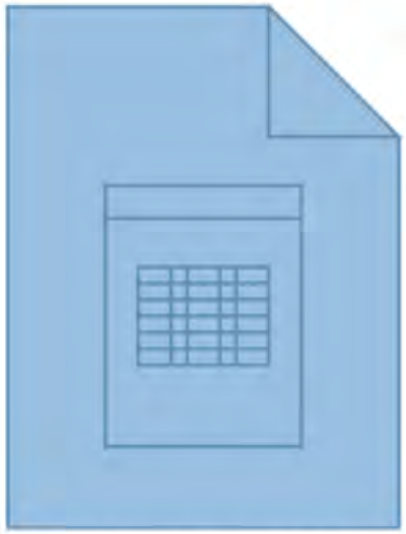

DEMO
You are now in the cloud!
Let's compare:
DEMO
This is a sandboxed solution.
Say 'no' to
sandboxed solutions
Path to Better Performance

This is a sandbox.



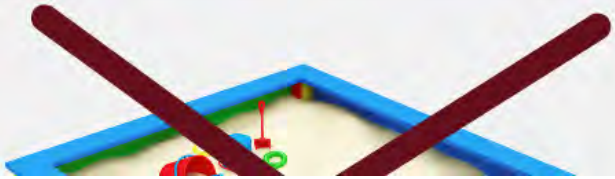
WSP File
(CAB structure)

This is a sandbox.



MANIFEST.XML

This is a sandboxed solution.

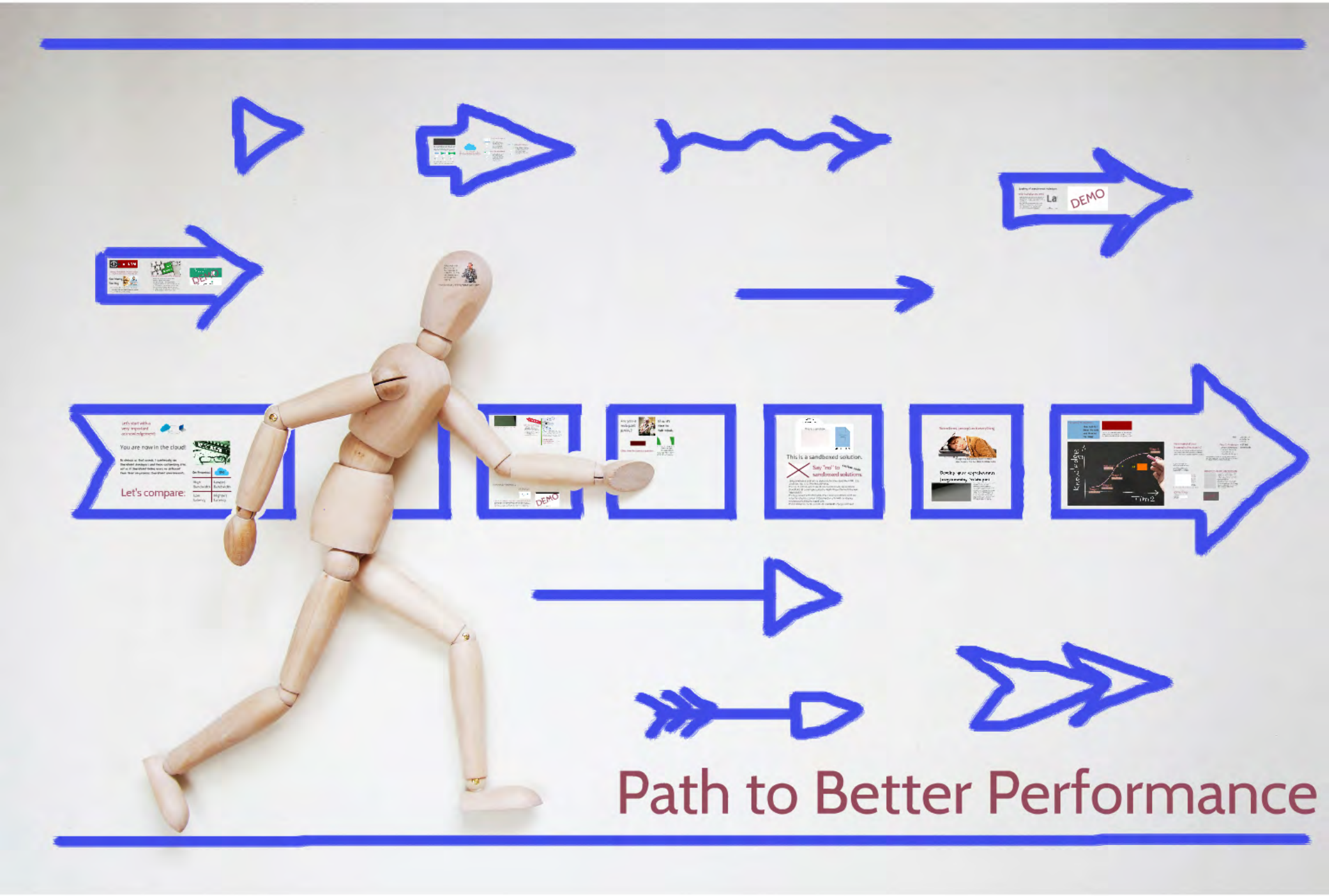


Say "no" to custom code

This is a sandboxed solution.

Say "no" to custom code sandboxed solutions.

- Using sandboxed solutions to deploy declarative assets like HTML, CSS, JavaScript, etc., is still *technically* okay.
- The use of custom code in sandboxed solutions was deprecated in SharePoint 2013 and replaced by the Add-In Model (formerly the new "App Model").
- The big problem with SPO: behind the scenes, (publisher) certificate validation checks occur (on SPO servers) and fail with sandboxed solutions containing managed code.
- These validation checks can introduce ***seconds*** of page overhead!


DEMO
You are now in the cloud!
Let's compare:
DEMO
This is a sandboxed solution.
Say 'no' to
sandboxed solutions
Path to Better Performance

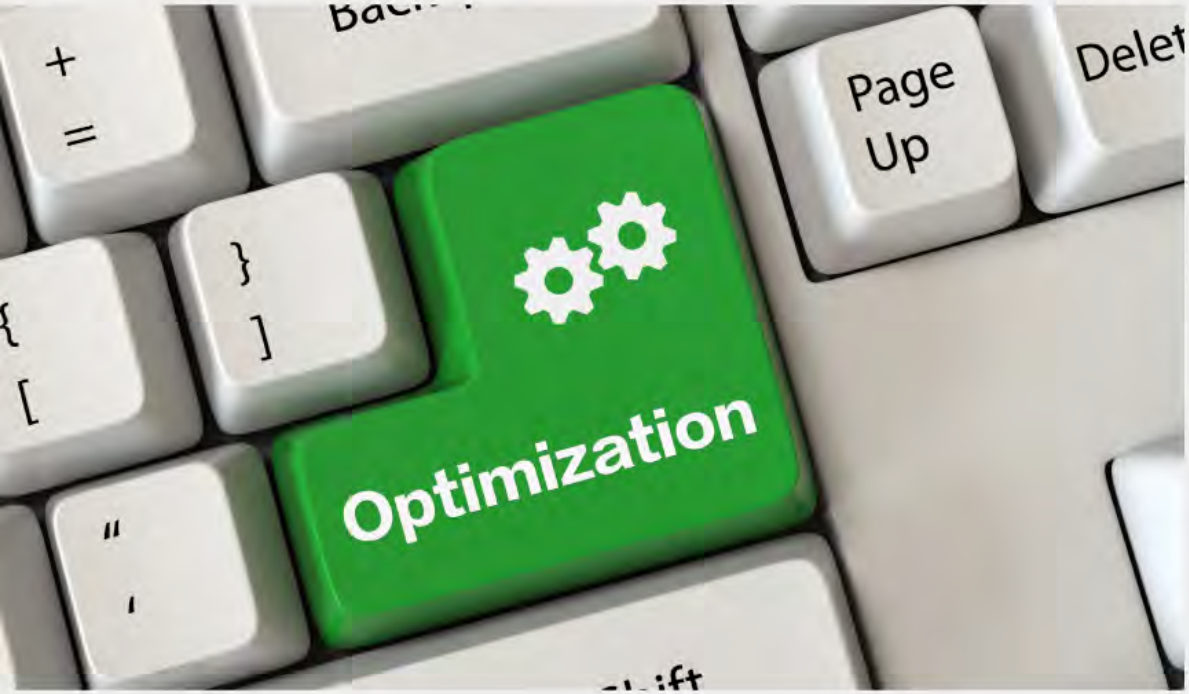
Sometimes, perception is everything.

A page may load quickly, but if it FEELS slow to users, it is the SAME AS BEING SLOW.

Sometimes, perception is everything.

A page may load quickly, but if it FEELS slow to users, it is the SAME AS BEING SLOW.

Develop your asynchronous programming techniques

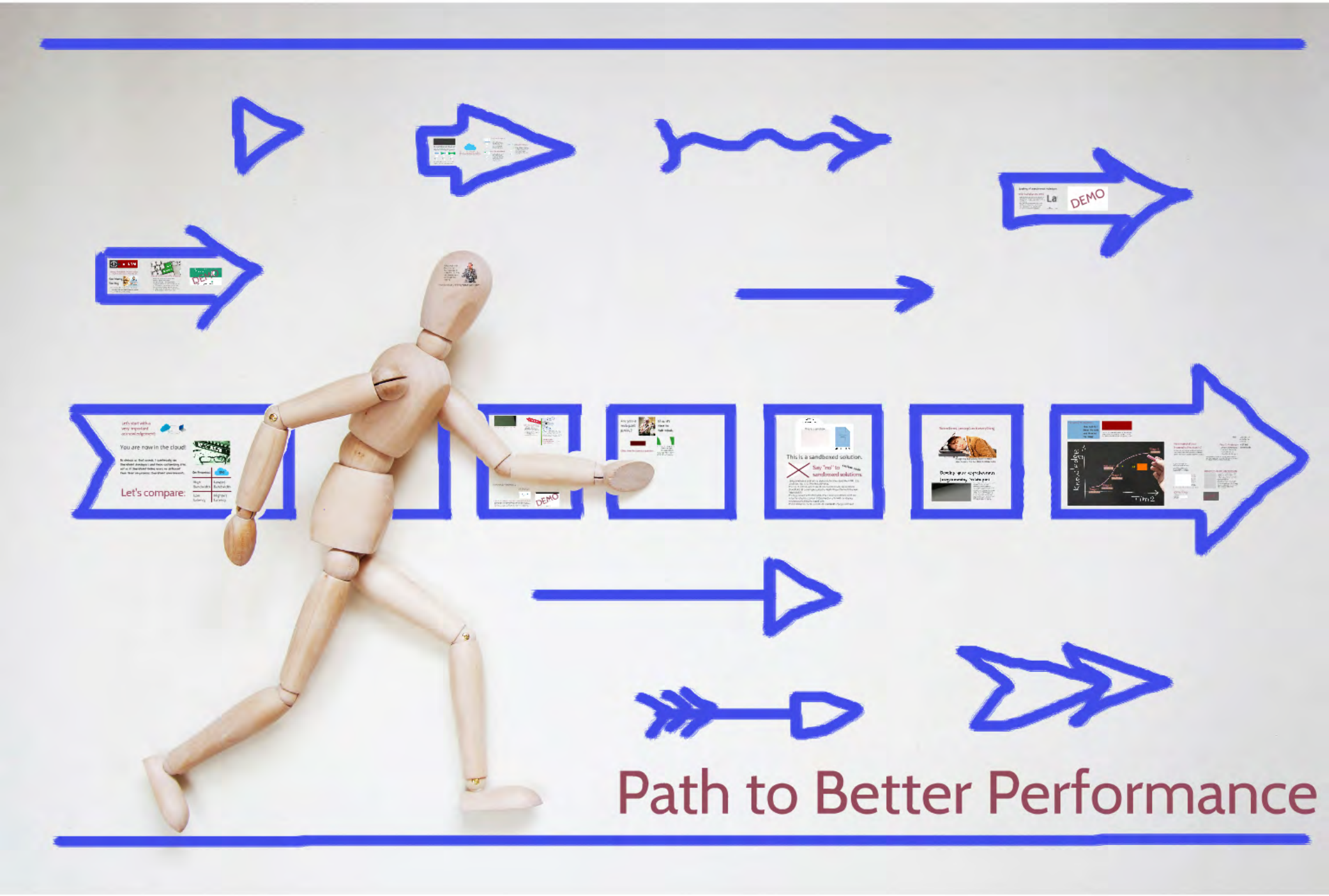


- You can't block a browser's main thread of execution, so leveraging async development patterns is essential.
- Async programming is made much easier in jQuery using promises. Promises approximate a synchronous programming model under asynchronous conditions.
- Certain web parts (e.g., the CSWP) also allow you to set their (a)sync behavior.
- Good use of async techniques make pages ***appear*** to load faster … and as we discussed, perception is everything.

Speaking of asynchronous techniques:

Only load what you need.

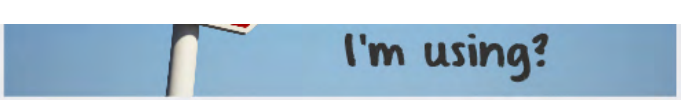
- Instead of fetching everything at once within the context of the initial page load, retrieve the page with only the payload that's needed immediately.
- (Lazy) load images and other items "below the fold" only if users start scrolling down and will see them (e.g., Facebook and LinkedIn's "forever-scrolling" pages).


DEMO
You are now in the cloud!
Let's compare:
DEMO
This is a sandboxed solution.
Say 'no' to
sandboxed solutions
Path to Better Performance

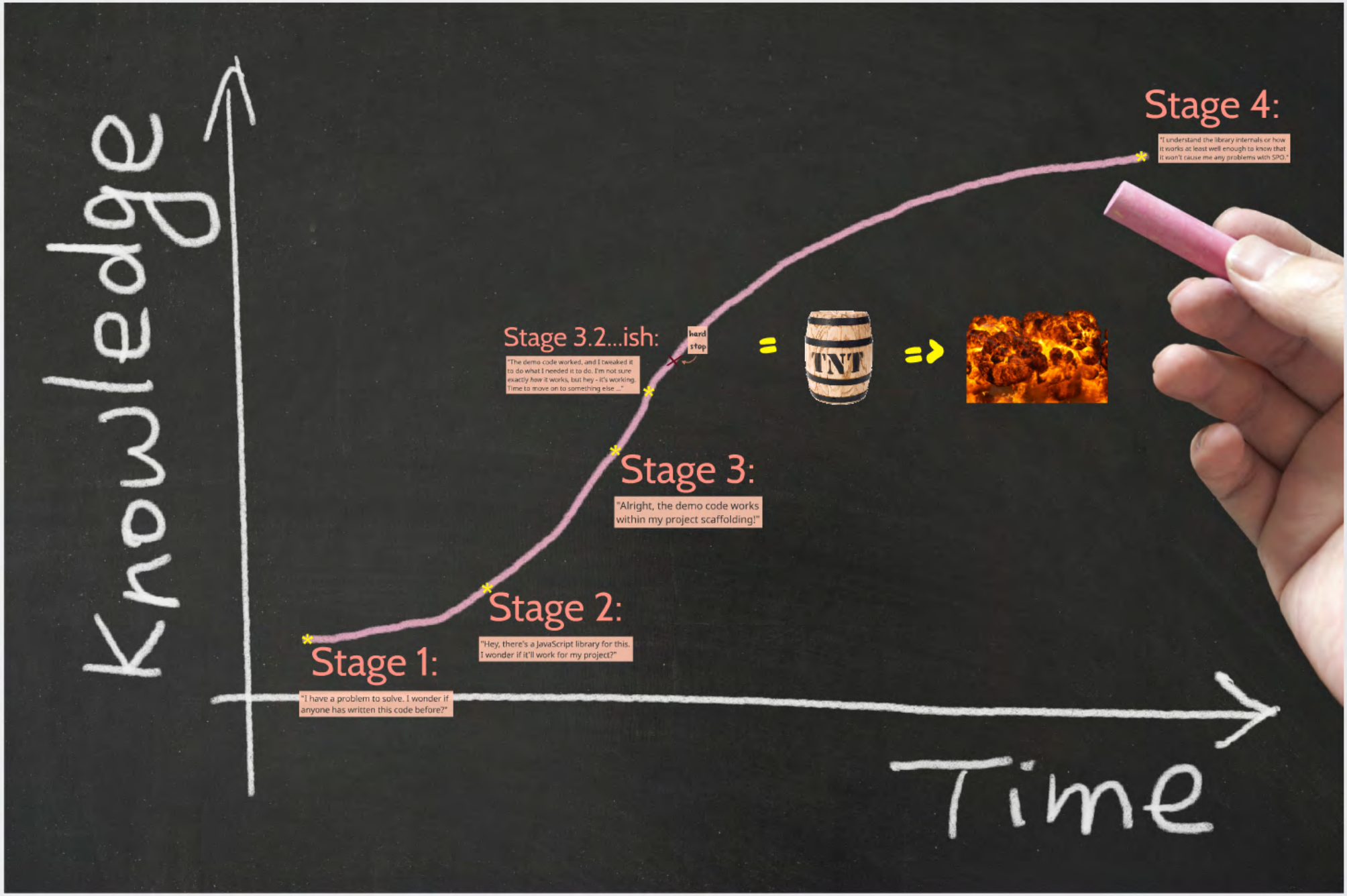
Ask yourself this question:

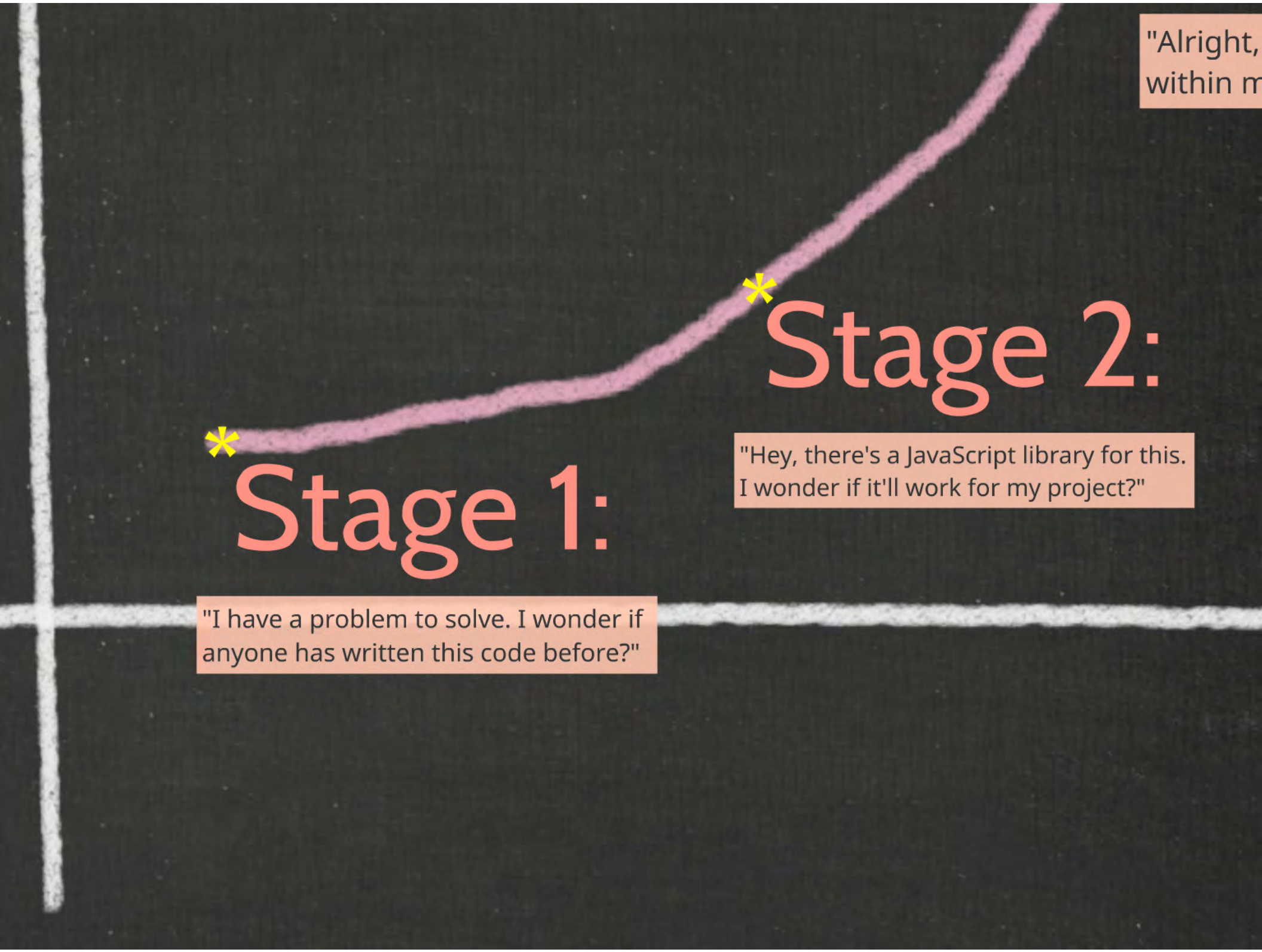
How well do I know the code and libraries I'm using?

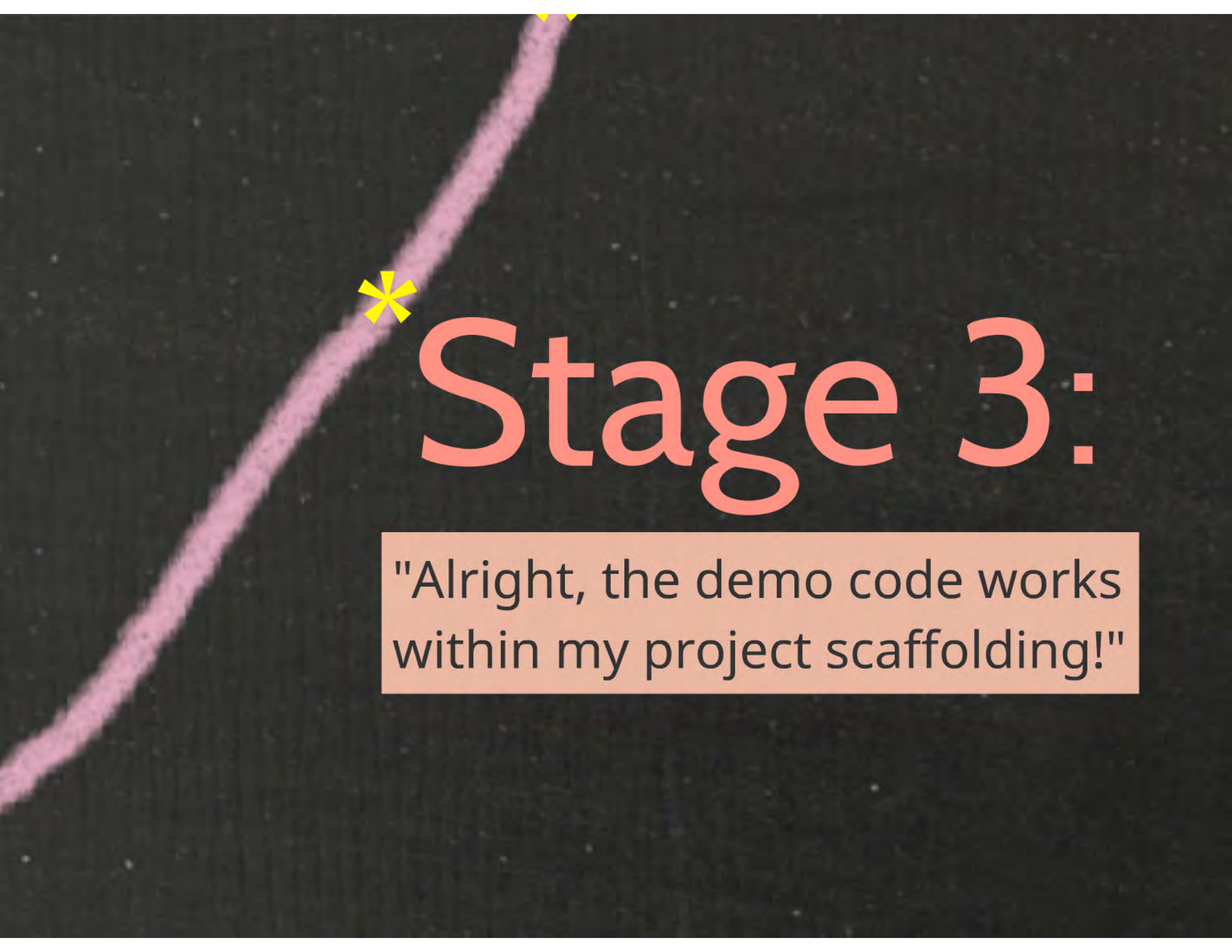
Reason I ask: when troubleshooting performance issues, I commonly encounter a pattern that can be illustrated with the following diagram of stages:

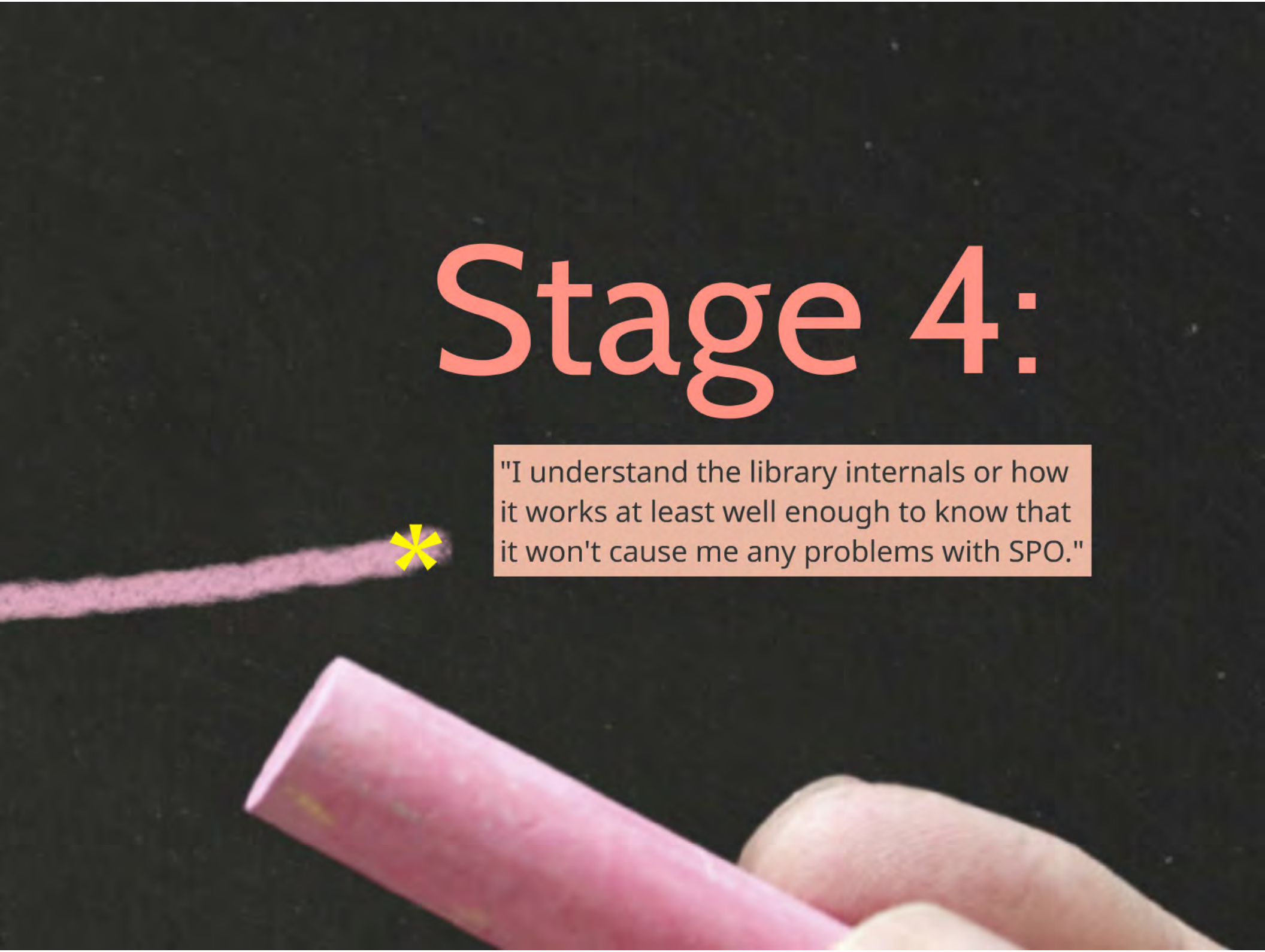


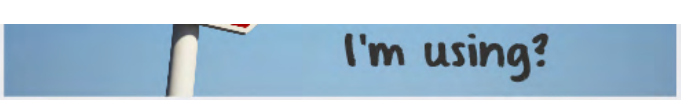
be illustrated with the following diagram of stages:



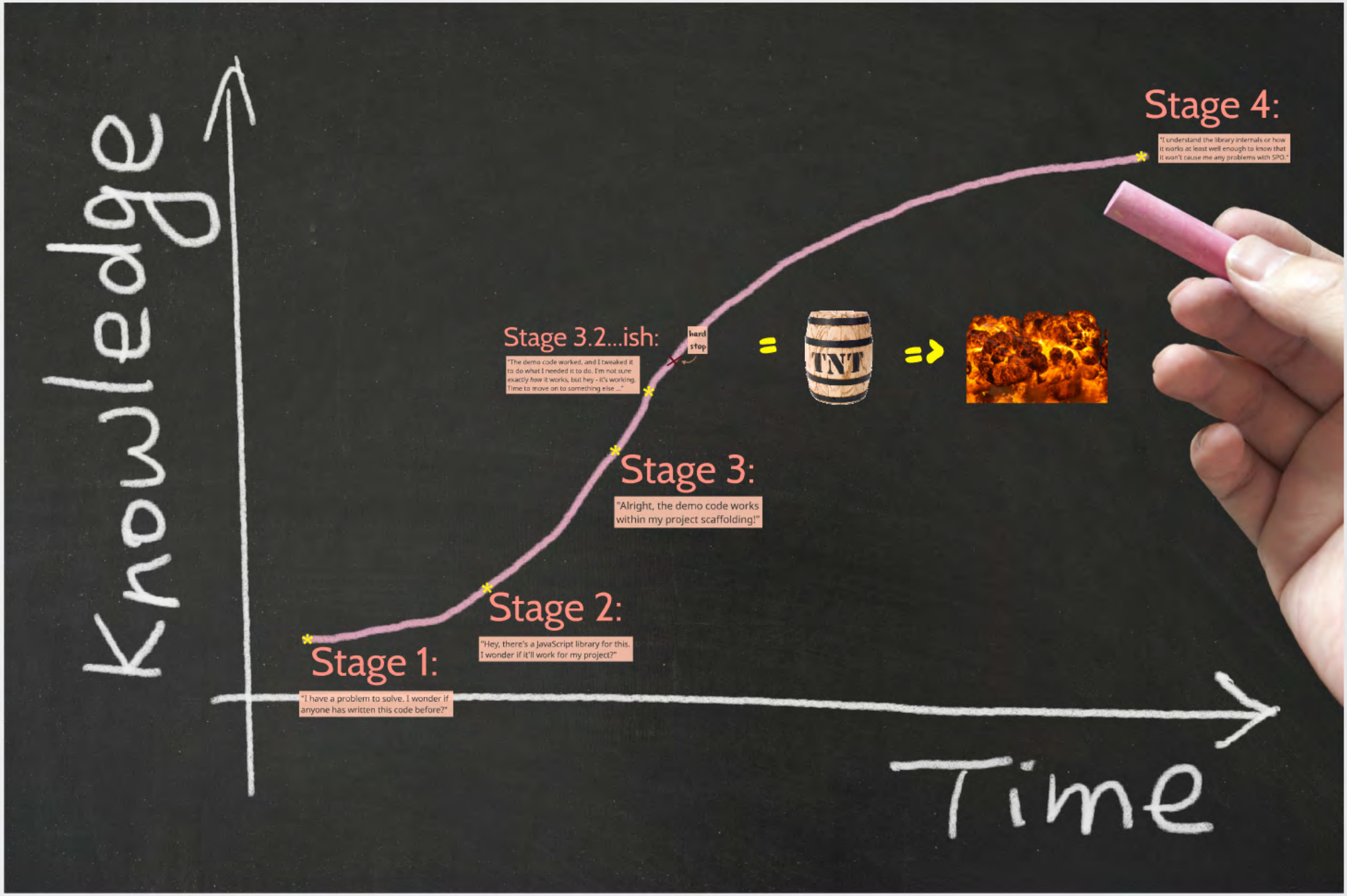

"Alright,
within m
*Stage 2:
"Hey, there's a JavaScript library for this.
I wonder if it'll work for my project?"
*Stage 1:
"I have a problem to solve. I wonder if
anyone has written this code before?"

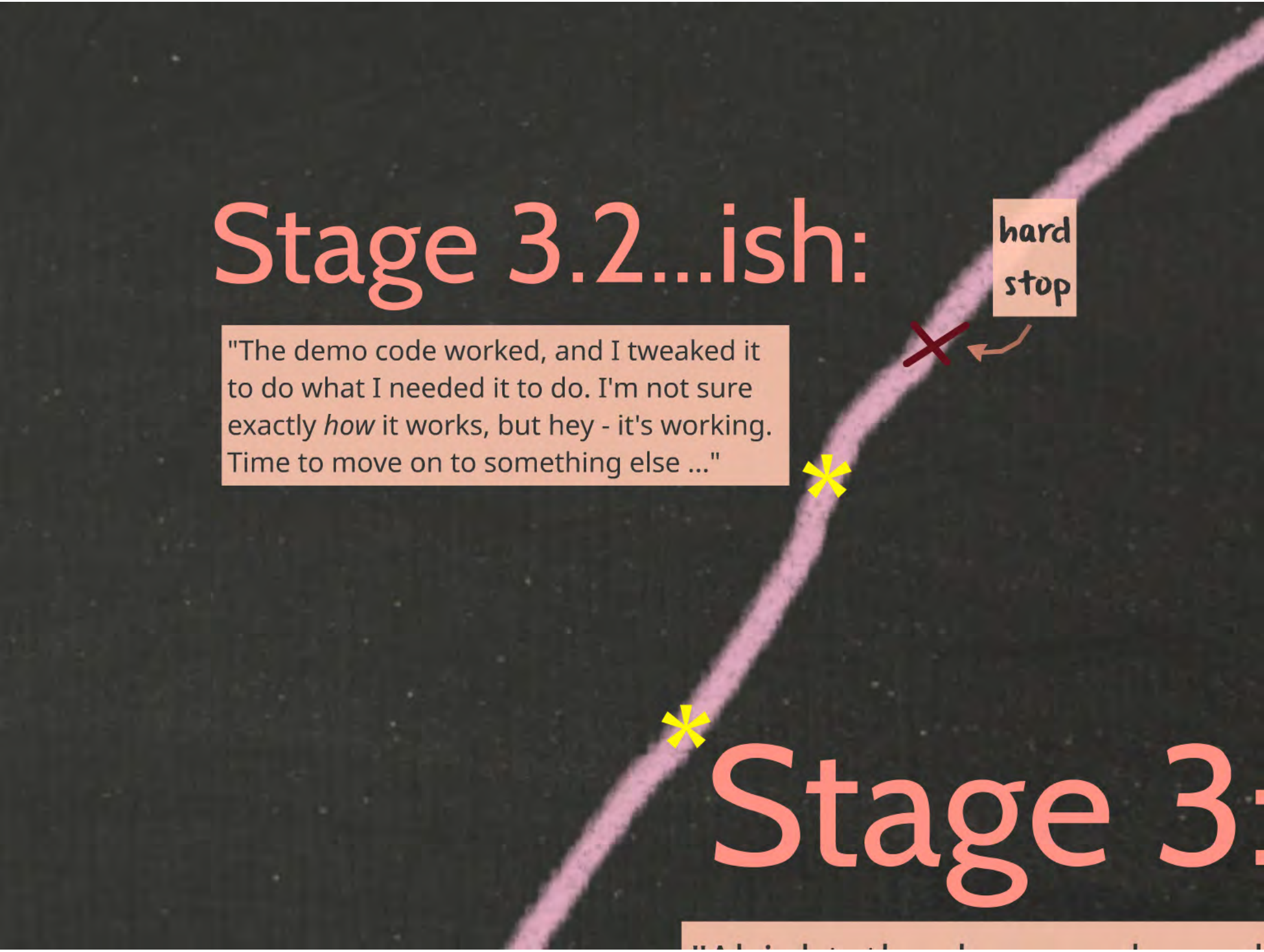

Stage 3:
"Alright, the demo code works within my project scaffolding!"

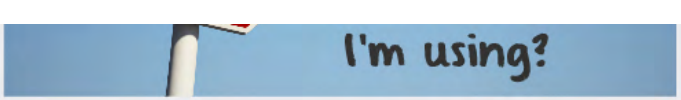

Stage 4:
"I understand the library internals or how it works at least well enough to know that it won't cause me any problems with SPO."
*



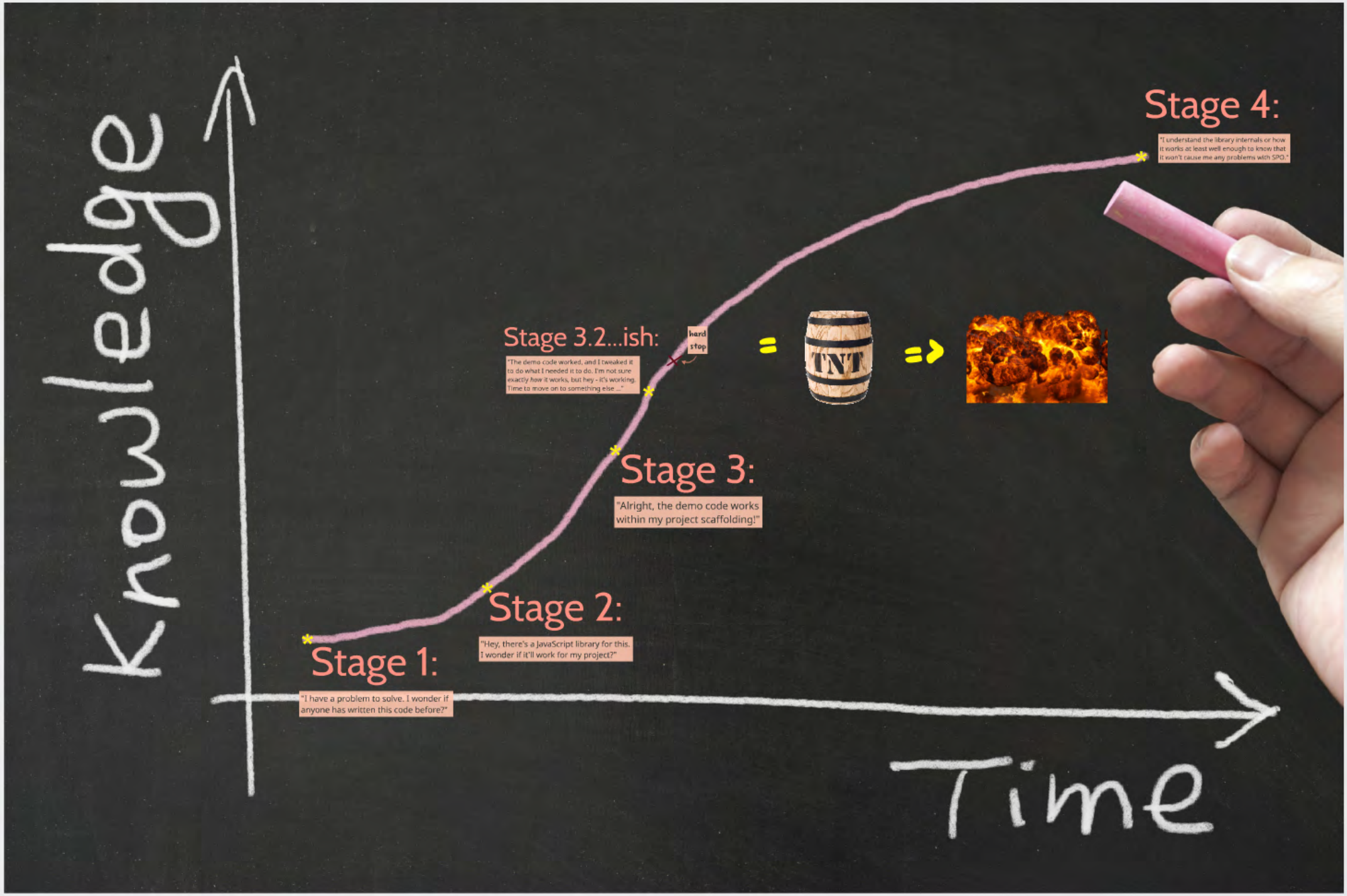
be illustrated with the following diagram of stages:




Stage 3.2...ish:
hard stop
"The demo code worked, and I tweaked it to do what I needed it to do. I'm not sure exactly *how* it works, but hey - it's working. Time to move on to something else ..."
Stage 3:



be illustrated with the following diagram of stages:



How many of you

How many of you know who this man is?

Chances are at least one or two of you have used code that he has created to get things done in your client-side development projects ...

Before CSOM/JSOM and REST
Marc's library simplified acces
for developers everywhere. H

Marc D. Anderson

- **creator of SPServices**
- master of client-side development and associated techniques

Before CSOM/JSOM and REST APIs - and before WCF SVC endpoints - Marc's library simplified access to the older ASMX web service endpoints for developers everywhere. It's still used heavily today.

Wow!

Last check on CodePlex (for SPServices):

Anderson

SPServices

client-side

> 27,000 downloads

So, getting back to "know your code/libraries" and how they work ...

As Marc will tell you, SPServices works just

So, getting back to "know your code/libraries" and how they work ...

As Marc will tell you, SPServices works just fine with SharePoint Online. But even Marc will tell you that you probably shouldn't use all of SPServices' methods when accessing SPO.

```
1   //Pre-populate all "Contact" fields with current user
2   var thisUserName = $().SPServices.SPGetCurrentUser({
3       fieldName: "Title",
4       debug: false
5   });
6   $().SPServices.SPFindPeoplePicker({
7       peoplePickerDisplayName: "Contact",
8       valueToSet: thisUserName,
9       checkNames: true
10  });
11  $().SPServices.SPFindPeoplePicker({
12      peoplePickerDisplayName: "Author/Contact",
13      valueToSet: thisUserName,
14      checkNames: true
15  });
16  $().SPServices.SPFindPeoplePicker({
17      peoplePickerDisplayName: "Organizer/Contact",
18      valueToSet: thisUserName,
19      checkNames: true
20  });
```

Consider this code.

It works just fine and does exactly what the comment indicates.

But it has a big problem.

anyone ever used the SPServices.SPGetCurrentUser() met

```
1   //Pre-populate all "Contact" fields with current user
2   var thisUserName = $().SPServices.SPGetCurrentUser({
3       fieldName: "Title",
4       debug: false
5   });
6   $().SPServices.SPFindPeoplePicker({
7       peoplePickerDisplayName: "Contact",
8       valueToSet: thisUserName,
9       checkNames: true
10  });
11  $().SPServices.SPFindPeoplePicker({
12      peoplePickerDisplayName: "Author/Contact",
13      valueToSet: thisUserName,
14      checkNames: true
15  });
16  $().SPServices.SPFindPeoplePicker({
17      peoplePickerDisplayName: "Organizer/Contact",
18      valueToSet: thisUserName,
19      checkNames: true
20  });
```

Consider this code.

It works just fine and does exactly what the comment indicates.

But it has a big problem.

Has anyone ever used the SPServices.SPGetCurrentUser() method?

Switching over to REST-based calls

WANTED: MARC ANDERSON

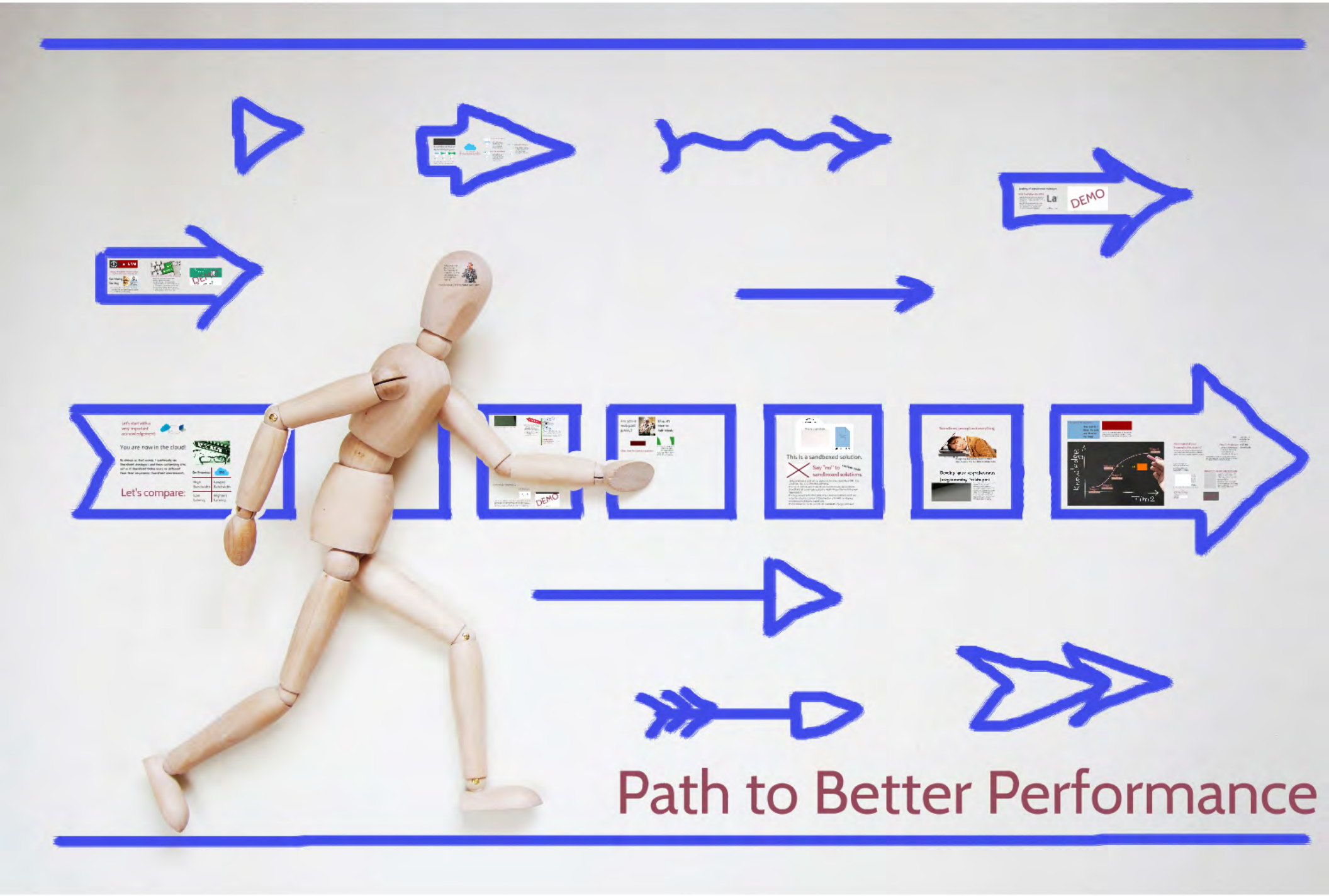
- Under the hood, SPGetCurrentUser() is generating an additional call to /_layouts/userdisp.aspx to “scrape” the contents of the page that is returned.
- If you (innocently) use SPGetCurrentUser() in your JavaScript files (especially multiple times in the context of a single page), you're creating all sorts of additional load on SPO and delaying the final results of your executing scripts.

Switching over to REST-based calls to get current user information can dramatically reduce execution time.

We had a script where SPGetCurrentUser() was being called several times. The results from swapping-in REST-based calls for the SPGetCurrentUser() calls:

I also performed some basic tests to capture the speed differences. I performed each test 10 times and here are the results:

* Without the fix or browser caching - avg. 14.47 seconds
* With the fix without browser caching - avg. 7.17 seconds
* With the fix and browser caching - avg. 5.84 seconds


DEMO
You are now in the cloud!
Let's compare:
DEMO
This is a sandboxed solution.
Say 'no' to
sandboxed solutions
Path to Better Performance

The Quick Summary

- Don't treat SPO like your on-premises SharePoint farm. The two operate differently.
- Server-based caching isn't your friend (generally speaking) in SPO.
- Your browser can be your best friend when trying to troubleshoot SPO performance issues.
- Know the code you implement - or at least profile it before release.

References

SPTechCon Austin 2016
http://www.sptechcon.com/

"Technical diagrams for SharePoint 2013"
https://technet.microsoft.com/en-us/library/cc263199.aspx

"Office 365 Performance Management"
https://mva.microsoft.com/en-US/training-courses/office-365-performance-management-8416

References

“Where is my data?”
https://www.microsoft.com/online/legal/v2/en-us/MOS_PTC_Geo_Boundaries.htm

“Deploying Office 365”
http://video.ch9.ms/sessions/project/2014/PC215_Medford.pptx

“New datacenter regions for Office 365”
https://support.office.com/en-us/article/Network-and-migration-planning-for-Office-365-f5ee6c33-bcd7-4b0b-b0f8-dc1d9fb8d132?ui=en-US&rs=en-US&ad=US#calculators

References

“Network and migration planning for Office 365”
https://support.office.com/en-us/article/Network-and-migration-planning-for-Office-365-f5ee6c33-bcd7-4b0b-b0f8-dc1d9fb8d132?ui=en-US&rs=en-US&ad=US#calculators

Fiddler web debugging proxy
http://www.telerik.com/fiddler

“FAQ – Certificates in Fiddler”
http://www.telerik.com/blogs/faq---certificates-in-fiddler

References

“Analyzing your webpage’s network traffic”
https://msdn.microsoft.com/en-us/library/dn255004(v=vs.85).aspx

“Diagnosing performance issues with SharePoint Online”
https://support.office.com/en-us/article/Diagnosing-performance-issues-with-SharePoint-Online-3c364f9e-b9f6-4da4-a792-c8e8c8cd2e86

“Image optimization for SharePoint Online”
https://support.office.com/en-gb/article/Image-optimization-for-SharePoint-Online-c7edb02a-fdab-4f91-9a20-cba01dad28ef

References

Font Awesome
https://fortawesome.github.io/Font-Awesome/

“How to improve your website performance with a CDN”
http://b2evolution.net/web-hosting/blog/tips/improve-website-performance-with-cdn

“Navigation options for SharePoint Online”
https://support.office.com/en-us/article/Navigation-options-for-SharePoint-Online-adb92b80-b342-4ecb-99a1-da2a2b4782eb

References

“Using Content Search Web Part instead of Content Query Web Part to improve performance in SharePoint Online”

https://support.office.com/en-us/article/Using-Content-Search-Web-Part-instead-of-Content-Query-Web-Part-to-improve-performance-in-SharePoint-Online-e8ce6b72-745b-464a-85c7-cbf6eb53391b

“Using jQuery Promises & Deferreds with SharePoint 2013”

http://blog.qumsieh.ca/2013/10/31/using-jquery-promises-deferreds-with-sharepoint-2013-jsom/

References

“Lazy Load Plugin for jQuery”
https://github.com/tuupola/jquery_lazyload

“Documentation > $().SPServices.SPGetCurrentUser”
http://spservices.codeplex.com/wikipage?title=%24%28%29.SPServices.SPGetCurrentUser

“SPServices Example: UserProfileService.GetUserProfileByName”
http://sympmarc.com/2012/02/15/spservices-example-userprofileservice-getuserprofilebyname/

References

“get current user info using jquery, REST and csom”
http://sharepoint.stackexchange.com/questions/124909/get-current-user-info-using-jquery-rest-and-csom

“Users, groups and the roles REST API reference”
https://msdn.microsoft.com/en-us/library/office/dn531432.aspx#bk_User

"Caching, You Ain't No Friend Of Mine"
http://sharepointinterface.com/2016/01/31/caching-you-aint-no-friend-of-mine/

References

“Deprecation of Custom Code in Sandboxed Solutions"
https://blogs.msdn.microsoft.com/sharepointdev/2014/01/14/deprecation-of-custom-code-in-sandboxed-solutions/

Sean P. McDonough

SharePoint & Office 365 Gearhead, Tinkerer, and Microsoft MVP

Twitter: @spmdonough

Blog: http://SharePointInterface.com

About: http://about.me/spmcdonough