

Indexing and searching NuGet.org with Azure Functions and Search

Maarten Balliauw
@maartenballiauw

JET
BRAINS

ConFoo.CA

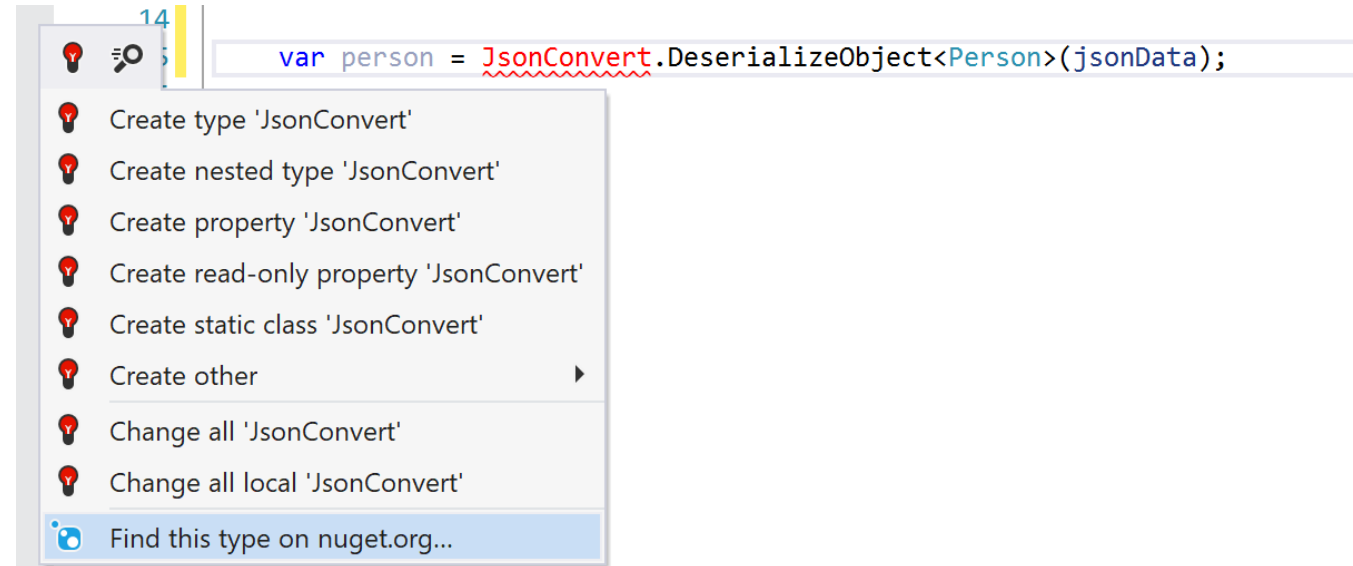
A misty forest scene with a wooden tower in the background. The text "Find this type on NuGet.org" is overlaid in white.

“Find this type on NuGet.org”

“Find this type on NuGet.org”

In ReSharper and Rider

Search for namespaces
& types that are not yet referenced



“Find this type on NuGet.org”

Idea in 2013, introduced in ReSharper 9

(2015 - https://www.jetbrains.com/resharper/whatsnew/whatsnew_9.html)

Consists of

- ReSharper functionality

- A service that indexes packages and powers search

 - Azure Cloud Service (Web and Worker role)*

 - Indexer uses NuGet OData feed*

 - [https://www.nuget.org/api/v2/Packages?\\$select=Id,Version,NormalizedVersion,LastEdited,Published&\\$orderby=LastEdited%20desc&\\$filter=LastEdited%20gt%20datetime%272012-01-01%27](https://www.nuget.org/api/v2/Packages?$select=Id,Version,NormalizedVersion,LastEdited,Published&$orderby=LastEdited%20desc&$filter=LastEdited%20gt%20datetime%272012-01-01%27)



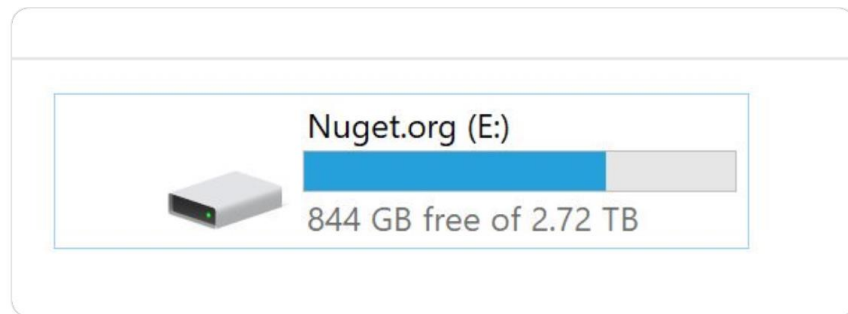
NuGet over time...



Alexander Shvedov
@controlflow

huh, nuget.org repo is 1.9Tb now... was like 250Gb in a year 2015

Tweet vertalen



11:20 - 28 nov. 2018

6 retweets 22 likes



4 replies 6 retweets 22 likes

<https://twitter.com/controlflow/status/1067724815958777856>

Search or jump to... Pull requests Issues Marketplace Explore

NuGet / Announcements

<> Code Issues 37 Pull requests 0 Projects 0 Security Insights

Unwatch 192 Star 38 Fork 3

Using OData to query NuGet.org repository is being deprecated #37

anangaur opened this issue 26 days ago · 0 comments

anangaur commented 26 days ago · edited

We introduced `v3 APIs` in early 2016. We have made continuous investments to make it more reliable, performant and secure. Going forward, we plan to bring all new features and improvements only to the newer `v3 APIs`. As part of this strategy, we are deprecating the usage of `odata` queries. Subsequently, we plan to disable the `odata` queries starting Feb 5th, 2020.

Call to Action: Transition to V3 APIs

Here are some resources to help you with the migration:

- NuGet Server `v3 APIs` reference
- Guide to transition from NuGet `v2 API` to `v3 API`.
- Blog post: Switching from WCF OData to Web API

If you need further help in moving your use case to `v3 APIs`, do reach out to us at support@nuget.org or by commenting on the discussion issue: [NuGet/NuGetGallery#7423](#)

Note:
This does not impact the official legacy clients (nuget.exe 2.x or Visual Studio 2013) that rely on the V2 endpoints (<https://www.nuget.org/api/v2>)

anangaur added **Breaking Change** **Announcement** labels 26 days ago

anangaur locked and limited conversation to collaborators 26 days ago

Assignees: No one assigned

Labels: **Announcement**, **Breaking Change**

Projects: None yet

Milestone: No milestone

Notifications: [Unsubscribe](#)

You're receiving notifications because you're watching this repository.

1 participant

<https://github.com/NuGet/Announcements/issues/37>



NuGet server-side API

V3 Protocol

JSON based

A “resource provider” of various endpoints per purpose

Catalog (NuGet.org only) – append-only event log

Registrations – materialization of newest state of a package

Flat container – .NET Core package restore (and VS autocompletion)

Report abuse URL template

Statistics

...

<https://api.nuget.org/v3/index.json> (code in <https://github.com/NuGet/NuGet.Services.Metadata>)



Catalog seems interesting!

Append-only stream of mutations on NuGet.org

Updates (add/update) and Deletes

Chronological

Can continue where left off (uses a timestamp cursor)

Can restore NuGet.org to a given point in time

Structure

Root <https://api.nuget.org/v3/catalog0/index.json>

+ Page <https://api.nuget.org/v3/catalog0/page0.json>

+ Leaf <https://api.nuget.org/v3/catalog0/data/2015.02.01.06.22.45/adam.jsgenerator.1.1.0.json>

NuGet.org catalog

demo



“Find this type on NuGet.org”

Refactor from using OData to using V3?

Mostly done, one thing missing: download counts (using search now)

<https://github.com/NuGet/NuGetGallery/issues/3532>

Build a new version?

Welcome to this talk 😊



Building a new version

What do we need?

Watch the NuGet.org catalog for package changes

For every package change

- Scan all assemblies

- Store relation between package id+version and namespace+type

API compatible with all ReSharper and Rider versions

What do we need?

Watch the NuGet.org catalog for package changes **periodic check**

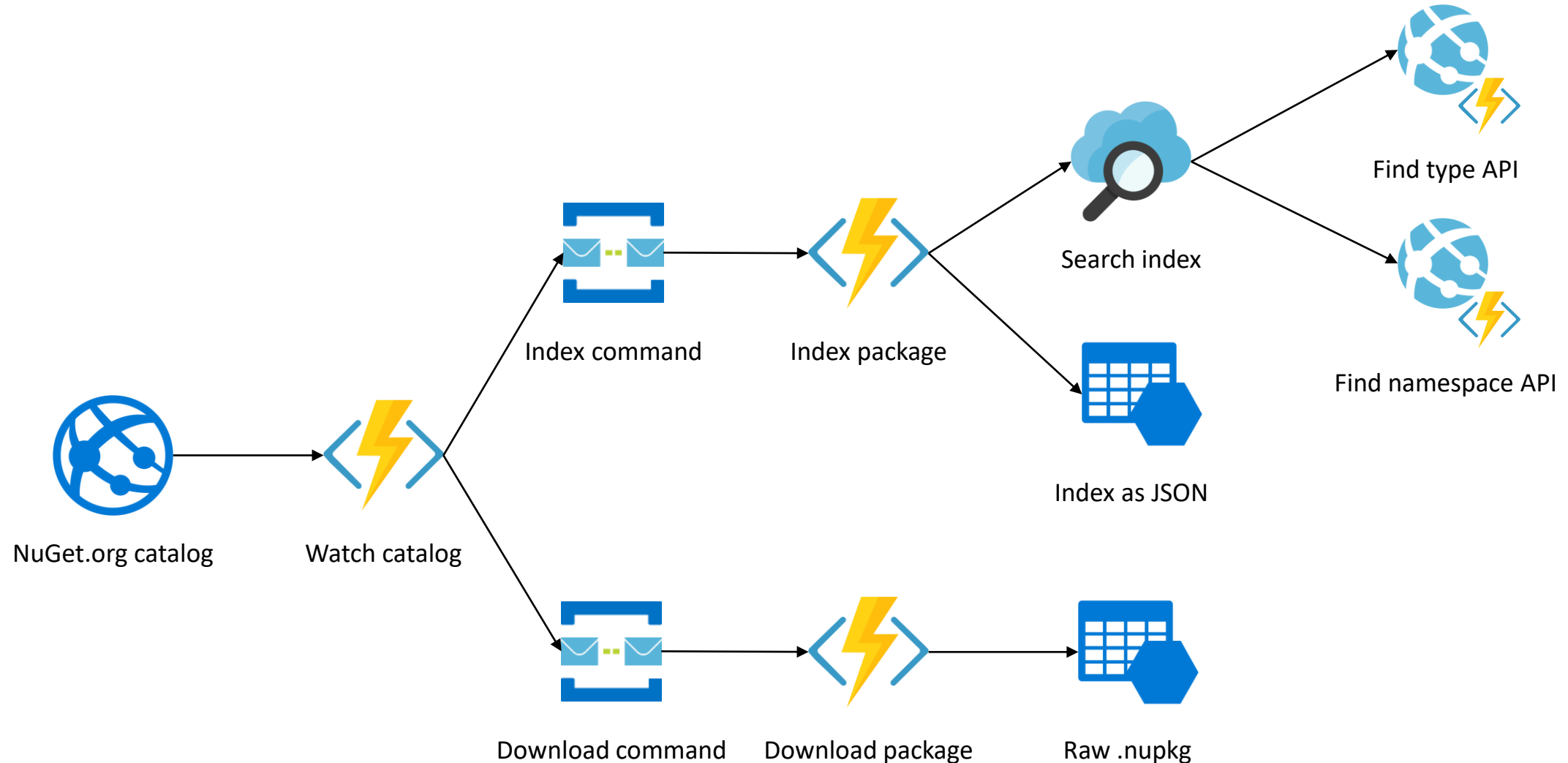
For every package change **based on a queue**

- Scan all assemblies

- Store relation between package id+version and namespace+type

API compatible with all ReSharper and Rider versions **always up, flexible scale**

Sounds like functions!



Collecting from catalog

demo



Functions best practices

@PaulDJohnston <https://medium.com/@PaulDJohnston/serverless-best-practices-b3c97d551535>

Each function should do only one thing

Easier error handling & scaling

Learn to use messages and queues

Asynchronous means of communicating, helps scale and avoid direct coupling

...

Bindings

	Trigger	Input	Output
Timer	✓		
HTTP	✓		✓
Blob	✓	✓	✓
Queue	✓		✓
Table		✓	✓
Service Bus	✓		✓
EventHub	✓		✓
EventGrid	✓		
CosmosDB	✓	✓	✓
IoT Hub	✓		
SendGrid, Twilio			✓
...			✓

Help a function do only one thing

Trigger, provide input/output

Function code bridges those

Build your own!*

SQL Server binding

Dropbox binding

...

NuGet Catalog

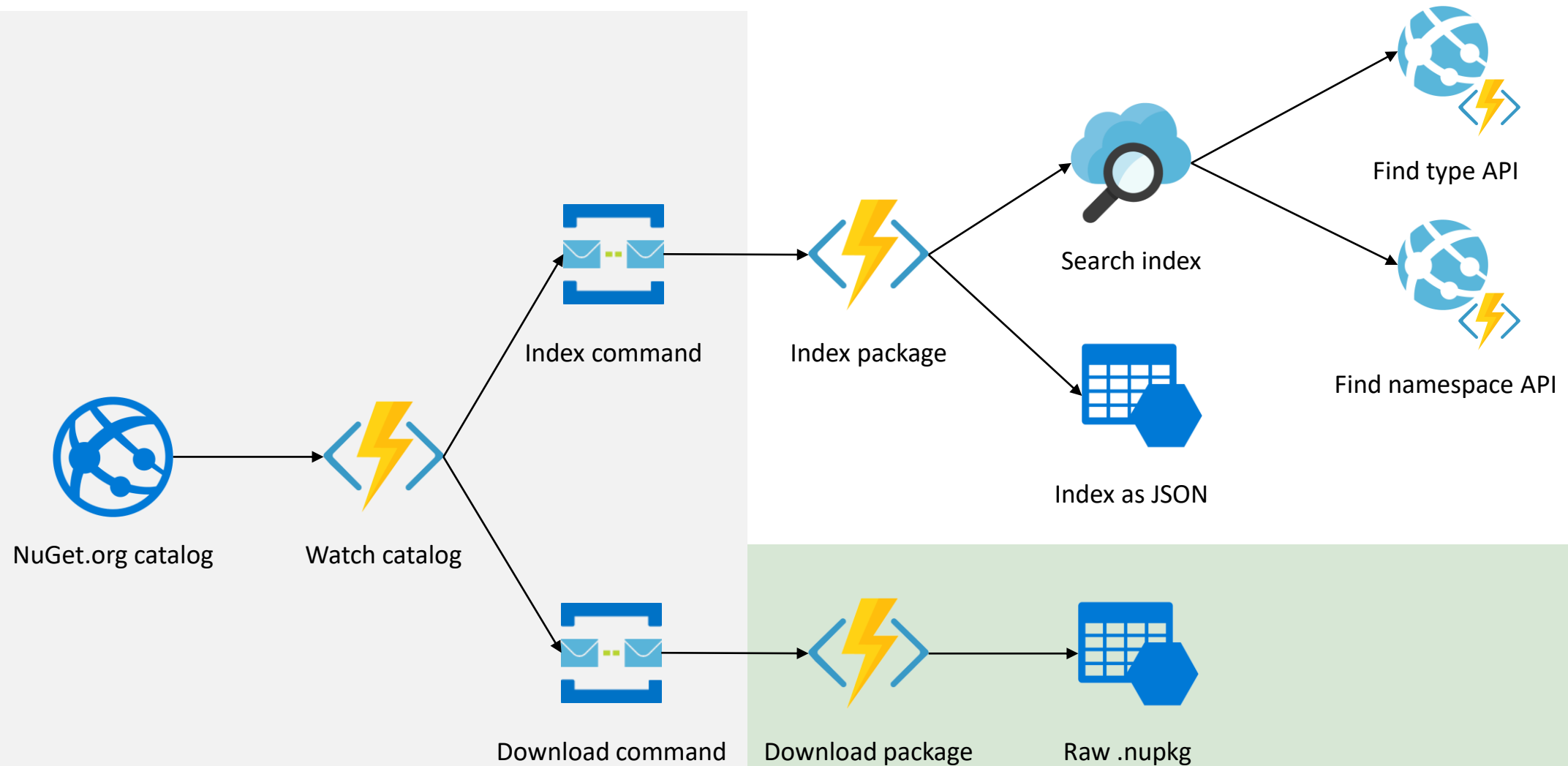
*Custom *triggers* not officially supported (yet?)

Creating a trigger binding

demo



We're making progress!

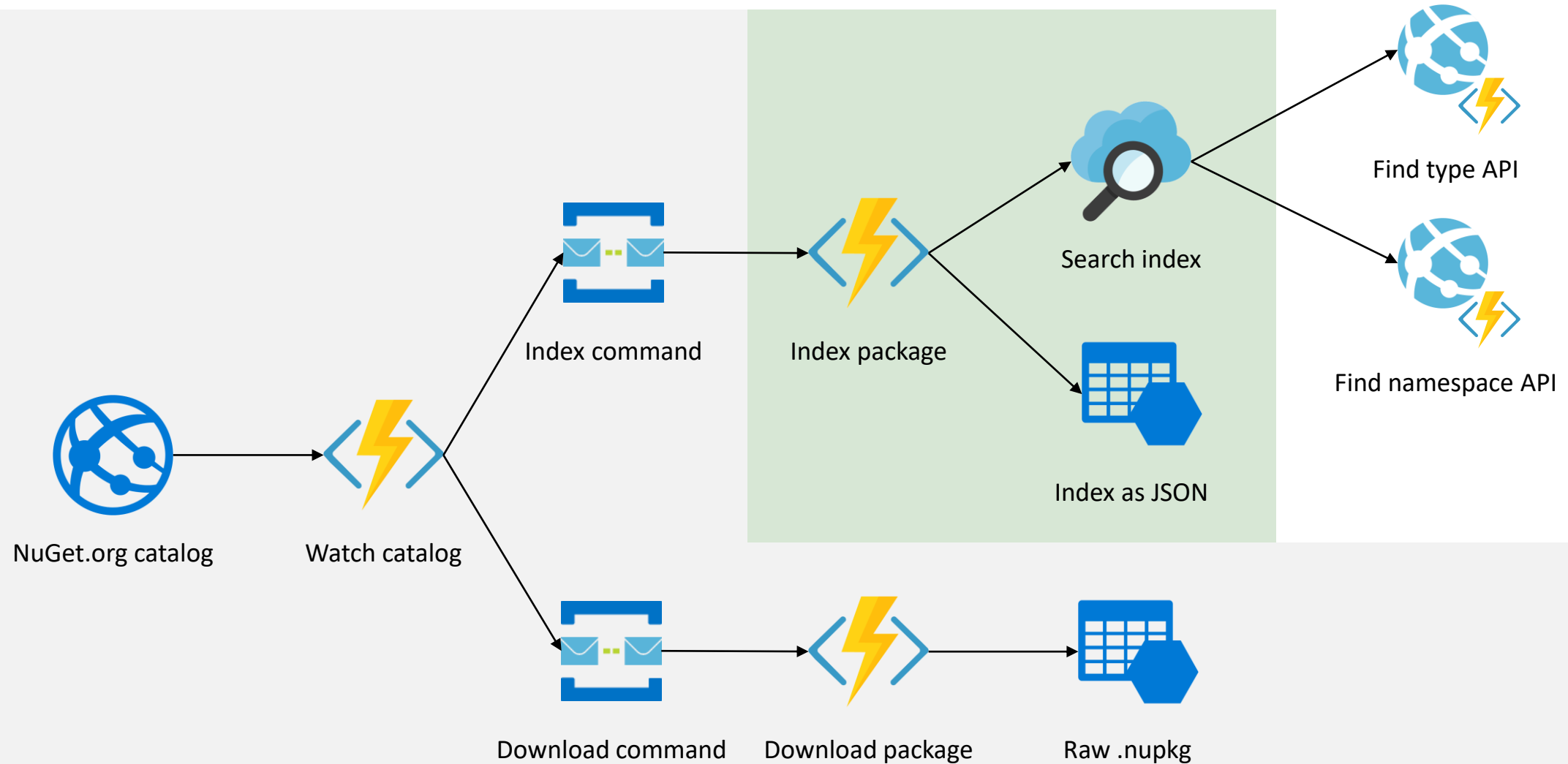


Downloading packages

demo



Next up: indexing



Indexing

Opening up the .nupkg and reflecting on assemblies

System.Reflection.Metadata

Does not load the assembly being reflected into application process

Provides access to Portable Executable (PE) metadata in assembly

Store relation between package id+version and namespace+type

Azure Search? A database? Redis? Other?

Indexing packages

demo



“Do one thing well”

Our function shouldn't care about creating a search index.

Better: return index operations, have something else handle those

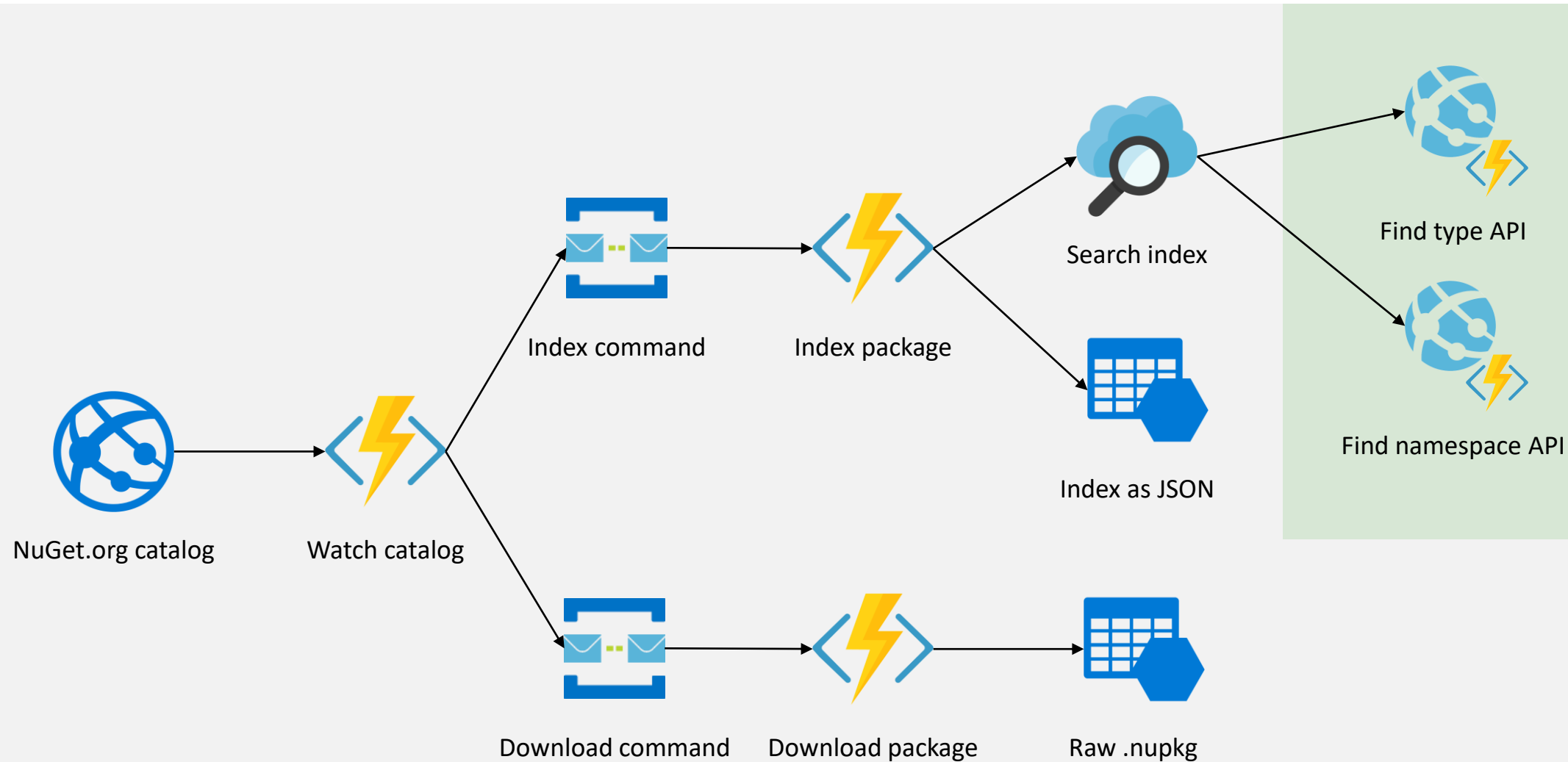
Custom output binding?



Blog post with full story and implementation

<https://bit.ly/2lzba32>

Almost there...

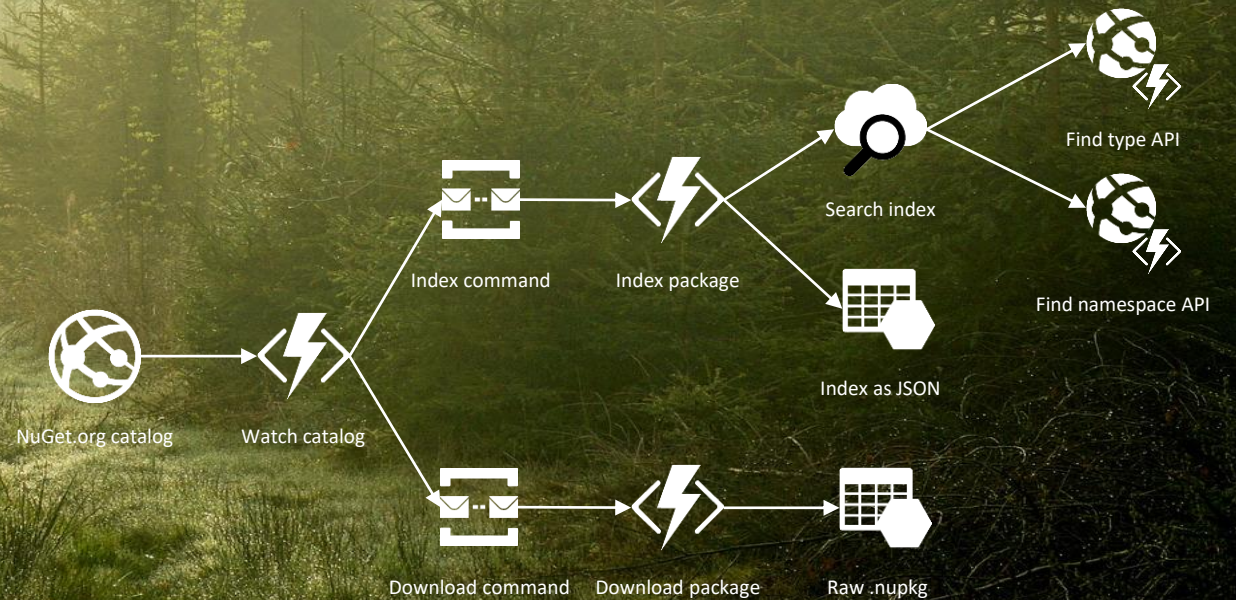


Making search work with ReSharper and Rider

demo



We're done!



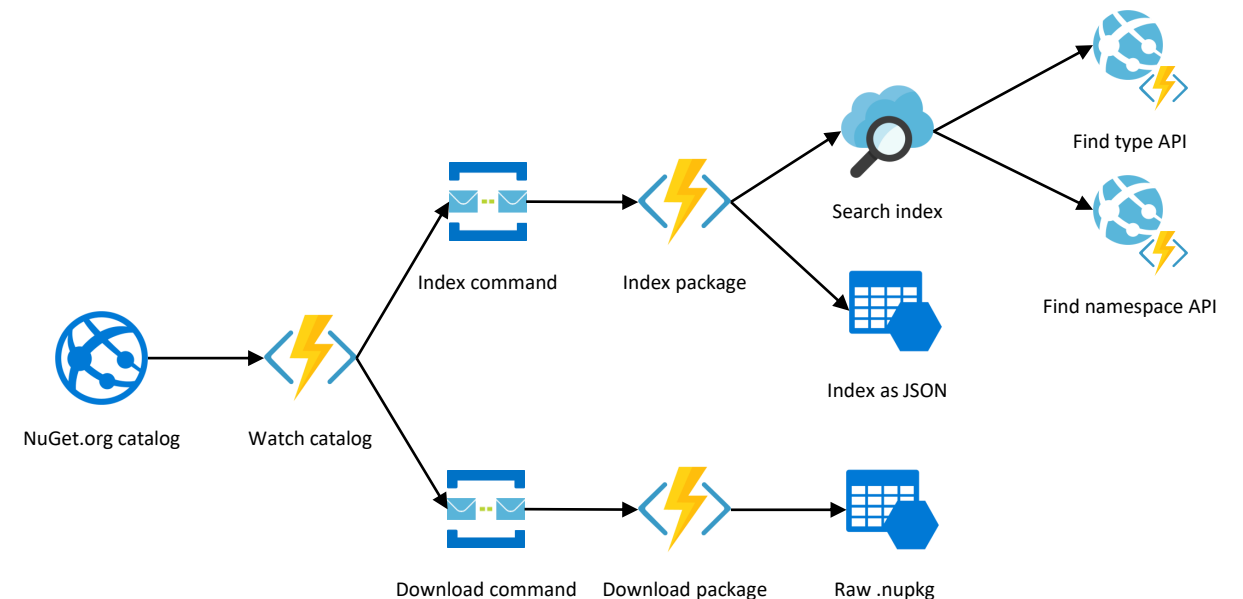
We're done!

Functions

- Collect changes from NuGet catalog
- Download binaries
- Index binaries using PE Header
- Make search index available in API

Trigger, input and output bindings

- Each function should do only one thing



We're done!

All our functions can scale (and fail)
independently

Full index in May 2019 took ~12h on 2 B1 instances

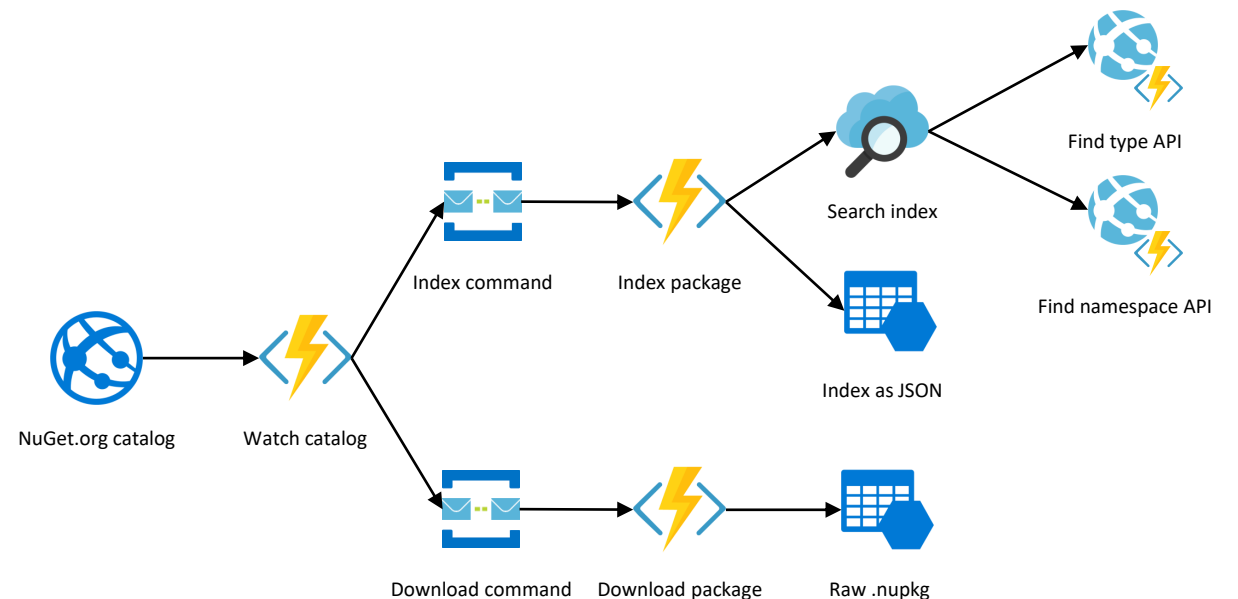
~ 1.7mio packages (NuGet.org homepage says)

~ 2.1mio packages (the catalog says 😊)

~ 8 400 catalog pages

with ~ 4 200 000 catalog leaves
(hint: repo signing)

January 2020: ~ 2.6 mio packages / 3.5 TB



Closing thoughts...

Would deploy in separate function apps for cost

- Trigger binding collects all the time so needs dedicated capacity (and thus, cost)

- Others can scale within bounds/consumption (think of \$\$\$)

Would deploy in separate function apps for failure boundaries

- Trigger, indexing, downloading should not affect health of API

Are bindings portable...?

- Avoid them if (framework) lock-in matters to you

- They are nice in terms of programming model...



Blog post with full story and implementation

<https://bit.ly/2lzba32>

Thank you!

<https://blog.maartenballiau.be>
@maartenballiau



**JET
BRAINS**

