



Sprint-demo
25 april 2019
Utrecht



Agenda

- 5 min. 1. Verslag API-lab (Hugo)
- 15 min. 2. Tutorial archiveren (Joeri)
- 15 min. 3. Tutorial notificeren (Joeri)
- 15 min. 4. Nieuwe opzet autorisatie (Sergei)
- 10 min. 5. Aanpak testen (Joeri)
- 30 min. 6. Utrecht - pilot zaakgericht werken (Constantijn)



Programma API-lab

Dag 1

9:30 Opening

11:00 Autorisatie

11:15 API-testvoorziening

12:30 Lunch

15:00 Notificaties

15:15 Archivering

17:00 Enquete

17:30 Diner

<https://ref.tst.vng.cloud/ontwikkelaars/tutorials/index>

Dag 2

9:30 Opstarten

12:30 Lunch

15:00 Demo's/pitches

16:30 Enquete

17:00 Borrel

Onderwerpen demo's API-lab

Enable-U

- Vertaalproces van ZDS naar ZGW.
- STAM-bericht naar ZGW API's.
- *Misschien* een echte aanvraag vanuit DSO.

Circle + Roxit

- Zaak + document creëren en koppelen vanuit samenwerkingsomgeving van Roxit.

Atos

- Inbouwen API's in E-suite (provider)

Procura

- Zaak creëren, ophalen en notificeren

Tilburg

- Ervaringen delen, misschien een demo.
- Koppeling vanuit Blueriq naar ZGW API's om zaak te creëren.

Decos

- Implementeren van zaaktypecatalogus
- Via webformulier een zaak aanmaken
- Lijst zaken ophalen
- "Knip maken in applicatie"

OIS

- Via webformulier creëren en ophalen zaak

Centric

- Van postintake naar zaak en document creëren
- API's op eigen zaaksysteem (provider)

Schagen

- Willen eerst vergelijking maken met eigen API's.

Voorlopige conclusies enquetes

- API-lab is zinvol (in het algemeen en 17/18 april)
- Er is vertrouwen in de release candidate op 18-7 (46% ja en 33% misschien).
- **Het inbouwen is niet ingepland.**
- Men is positief over de testvoorziening en men gaat deze gebruiken.
- Tutorials zijn beoordeeld met gemiddeld een 7.
- **Complexiteit van de standaard**

Staat het inbouwen van de standaard ingepland? →

Wat moeten we doen om op deze vraag een ja te krijgen (standaard, testvoorziening, functionaliteit, ondersteuning etc.)?

Te veel API calls

onduidelijke benaming van velden/berichten

Migratie strategie definiëren

* gemeenten dit laten eisen

Op roadmap = Ja, concreet gepland is nee

Providers moeten API beschikbaar hebben

* Staat al op de roadmap

* Roadmaps waren al bepaald, Wanneer de behoefte komt zal de planning aangepast worden

* Als het onderdeel is van een aanbesteding

* Waarschijnlijk als er behoefte is vanuit het gemeentelandschap

* Moet vanuit de klant vraag naar zijn

* Een klant die het eist

* = vraag vanuit gemeente

Complexiteit

microservices

Te veel api calls

VNG

UUID/URL

Uuid en url gebruik

Resources mappen niet lekker op bestaande situatie

objectinformatieobject

Veel attributen waarvan niet alle altijd relevant

microservices

API call storm

microservices

Veel losse acties die vroeger met 1 bericht konden

UUID/URL

Omgaan met identity/URL strategie

Onduidelijke benaming van velden/berichten

Duidelijke communicatie mbt beheer en changes

documentatie

Duidelijkere structuur in documentatie

microservices

Wennen aan multiservices als je stuf gewend bent

Veel tooling door elkaar, veel inrichting perikelen, afwijking van onze huidige implementatie, voornamelijk veel

Wat is er complex aan de standaard?

Autorisatie

Alles anders!

Inhoud

- Waarom autoriseren?
- “Oude” situatie
 - JSON Web Token (JWT)
 - Scopes
- De Wens van Gemeente Utrecht
- “Nieuwe” situatie
 - Gelaagdheid
 - Autorisatiecomponent
 - JWT structuur
 - Performance

Waarom autoriseren?

“Privacy by design”

Waarom autoriseren?

Eenvoudig: voorkomen dat door ongeautoriseerde toegang (gevoelige) gegevens op straat komen te liggen.

→ Reduceer aanvals vectoren

Attack vector / Aanvalsvector

An attack vector is a path or means by which a hacker (or cracker) can gain access to a computer or network server in order to deliver a payload or malicious outcome. Attack vectors enable hackers to exploit system vulnerabilities, including the human element.

Aanvalsvector: een gebruiker probeert van buitenaf de APIs te benaderen

1. Ontsluit API enkel via **intern netwerk** (= niet over internet)

2. Ontsluit API via **NLX** over organisaties heen

NLX is optioneel, wat zonder NLX? Wat met SAAS over publiek internet?

Aanvalsvector: een medewerker benadert APIs van binnen

1. Iedereen die wat technisch onderlegd is kan API calls maken
2. Laat **geen (duidelijke) sporen** na
3. Vertrouwen / boetes / contracten zijn leuk, maar gegevens zijn ondertussen **wel gelekt/vernietigd**

Aanvalsvector: een applicatie leest/wijzigt meer dan nodig

1. Specifieke taakapplicatie heeft enkel **behoefte aan subset van gegevens**
2. Foutenvrije software bestaat niet / menselijke **fouten** gebeuren (misconfiguratie)
3. Mogelijk lekt data zo via **externe partijen**

“Oude” situatie

En de problemen

Oude situatie: JSON Web Token

header.payload.signature

GET http://localhost:8000/api/v1/zaken HTTP/1.0

Authorization: Bearer

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImNsaWVudF9pZGVudGlmaWVyIjoiz
G9ja2VyLW5yYyJ9.eyJpc3MiOiJkb2NrZXItbnJjIiwiaWF0IjoxNTU0OTc1NTI2L
CJ6ZHMiOnsic2NvcGVzIjpbtM5vdGlmaWNhdG11cy5zY29wZXMuY29uc3VtZXJlbi
IsIm5vdGlmaWNhdG11cy5zY29wZXMuCHVibG1jZXJlbiJdLCJ6YWFrdfHlwZXMiOlt
dfX0._qiFsPB54DevBuR9dfiSBU18WhiQpon0H9xxgFF38f8

Accept: application/json

Oude situatie: JSON Web Token

client_id & secret

Autorisatie in de provider

1. Lees client_id uit header
2. Zoek bijhorend secret op
3. Valideer JWT signature met secret
4. Lees & gebruik de payload

Autorisatie in de consumer

1. Zet eigen client_id in header
2. Bouw & zet de payload
3. Signeer JWT met eigen secret
4. Gebruik JWT als HTTP header value

Client ID en Secret worden out-of-band uitgewisseld!

Oude situatie: Scopes

status_create

Voeg een nieuwe STATUS voor een ZAAK toe.

Er wordt gevalideerd op

- geldigheid URL naar de ZAAK
- geldigheid URL naar het STATUSTYPE
- indien het de eindstatus betreft, dan moet het attribuut `indicatieGebruiksrecht` gezet zijn op alle informatieobjecten die aan de zaak gerelateerd zijn

Opmerkingen

- Indien het statustype de eindstatus is (volgens het ZTC), dan wordt de zaak afgesloten door de einddatum te zetten.

AUTHORIZATIONS:

JWT-Claims (`(zds.scopes.zaken.aanmaken | zds.scopes.statussen.toevoegen)`)

JWT-Claims: structuur payload JWT

```
{  
    "zds": {  
        "scopes": [  
            "zds.scopes.zaken.aanmaken"  
        ],  
        "zaaktypes": [  
            "https://haarlem.ztc.nl/api/v1/zaaktypen/123",  
            "https://haarlem.ztc.nl/api/v1/zaaktypen/124",  
        ]  
    }  
}
```

- Scopes laten operaties wel/niet toe (= ik mag een zaak aanmaken)
- Zaaktypes filteren zaken waarop operaties mogelijk zijn (=ik mag iets met zaken van deze zaaktypen)

Oude situatie: Problemen

- Consumer wordt ‘vertrouwd’ - gemeente kan niets afdwingen
- Niet preventief (wel logging & audits)
- Fijnmazige scopes *per zaaktype*?
- Wat als je 100-en zaaktypen hebt?
- Payload groeit → HTTP header wordt steeds groter
- Niet weg van minste weerstand → neem “gewoon altijd alle scopes en zaaktypes”

De Wens van Gemeente Utrecht

Privacy by design

De Wens van Gemeente Utrecht

Als gemeente wil ik voor elke taakapplicatie per zaaktype kunnen aangeven welke rechten daar bij horen #838



CMasselink opened this issue on Mar 14 · 5 comments

“zodat een taakapplicatie nooit te veel rechten heeft.

*Past binnen het privacy by design principe waarbij er nooit meer informatie uit een bronapplicatie komt dan zou mogen. We passen dus **filtering bij de bron** toe.*

Je wilt kunnen aangeven dat een taakapplicatie lees en schrijfrechten heeft op zaaktype A, B en C en alleen leesrechten op zaaktypes X,Y,C. Ook wil je op vertrouwelijkheidsaanduiding toegang tot zaaktypes kunnen beperken.”

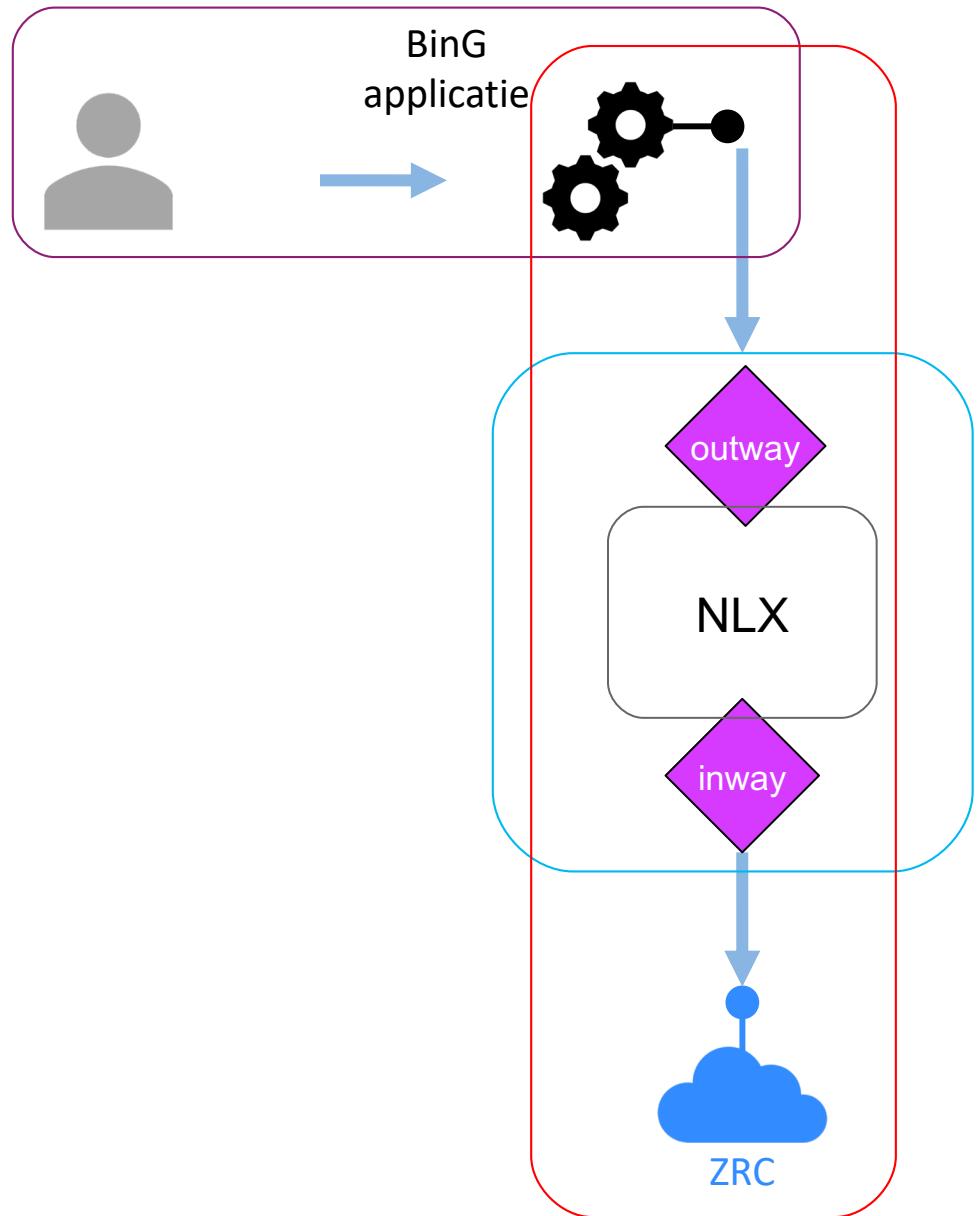
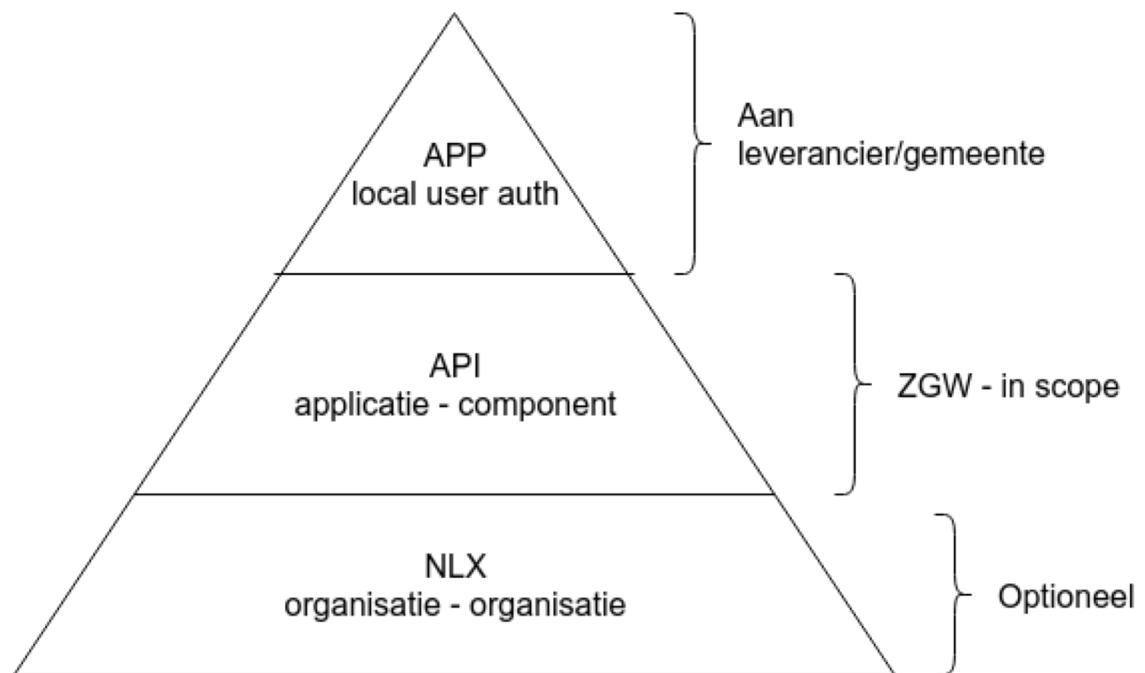
Opgeloste problemen

- Consumer wordt niet 'vertrouwd' - gemeente configureert toegang
- Wel preventief (ook logging & audits)
- Fijnmazige scopes *per zaaktype* en vertrouwelijkheidaanduiding ingevoerd
- Configuratie per zaaktype
- Weg van minste weerstand → provider bouwt/controleert alles!

“Nieuwe” situatie

Ok, niet *alles* verandert

Nieuwe situatie: Gelaagdheid



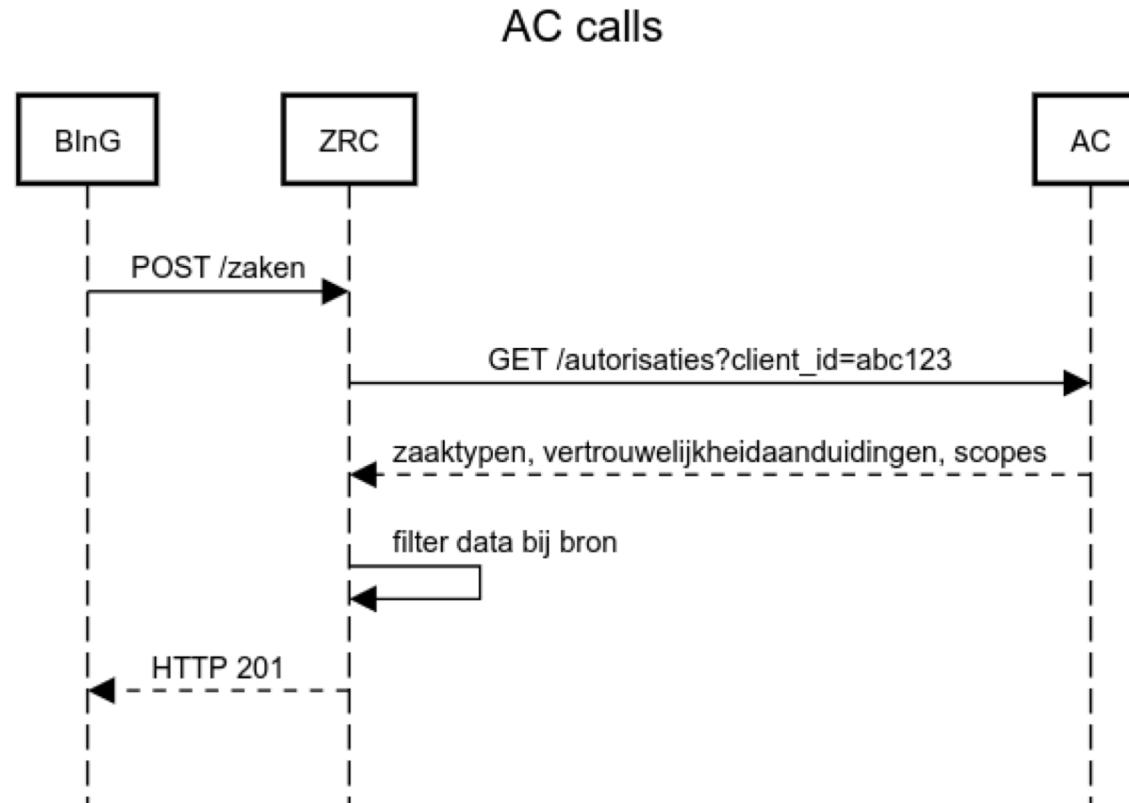
Autorisatiecomponent (AC)

Functionaliteiten:

- Organisatie configureert autorisaties voor applicaties (al dan niet wildcard)
- Providers kunnen opvragen wat autorisaties zijn voor een consumer
- Consumers kunnen zelf opvragen wat hun autorisaties zijn



Autorisatiecomponent (AC) (2)



Nieuwe situatie: JWT structuur

Header

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}
```

Payload

```
{  
  "iss": "abc123",  
  "iat": 1554755526,  
  "client_id": "abc123"  
}
```

Nieuwe situatie: Performance

- Elke API call → autz check
- Roundtrip naar AC: duur!
- Caching?
- Synchronisatie autorisaties?
- Inhaken op notificaties? Wijziging in AC → notificatie → invalidate cache

Sprint demo 11

Aanpak testen



Probleemstelling

- Als **ontwikkelaar** van de referentie implementaties wil ik zeker weten dat alle beschreven logica juist is geïmplementeerd.
- Als **API beheerder** wil ik kunnen aantonen dat een API werkt zoals het staat beschreven in de documentatie.
- Als **leverancier** wil ik mijn eigen implementaties kunnen testen (en een certificering ontvangen)
- Als **API testplatform beheerder** wil ik test scenario's om consumers en providers te valideren (en een certificering te geven)

Oplossing: Test scenario's!

- Test scenario's worden bijhouden in Github.
- Test scenario's worden aangeleverd door het API ontwikkelteam.
- Test scenario's zijn te herleiden naar de documentatie (en vice versa).
- Test scenario's kunnen worden gebruikt (en worden aangeleverd) om bugs aan te tonen (regressie tests).
- Test scenario's worden geschreven in Postman (Windows, MacOS, en Linux).

Voordelen

- Voor alle stakeholders (ontwikkelaars, API beheerders, leveranciers en API testplatform beheerder) dezelfde oplossing.
- Vormgegeven als GET, POST, UPDATE en PATCH verzoeken in combinatie met validatie middels Javascript.
- Test platform gebruikt ook Postman onder de motorkap.
- Test scenario's kunnen uitgevoerd worden zonder API testplatform.

Gedragsregels in documentatie (zrc-001)

Zaakregistratiecomponent

...

Valideren zaaktype op de Zaak-resource (zrc-001)

Bij het aanmaken (POST) en bijwerken (PUT en PATCH) MOET de URL-referentie naar het zaaktype gevalideerd worden op het bestaan.

Indien het ophalen van het zaaktype niet (uiteindelijk) resulteert in een HTTP 200 status code, MOET het ZRC antwoorden met een HTTP 400 foutbericht.

Test scenario's in Postman (zrc-001)

ZGW api tests

55 requests

standaard.md

zrc

- setUp
- zrc-001
 - POST** (zrc-001a) Valideren Zaaktype op Zaak-resource (POST)
 - PUT** (zrc-001b) Valideren Zaaktype op Zaak-resource (PUT)
 - PATC** (zrc-001c) Valideren Zaaktype op Zaak-resource (PATCH)
 - POST** (zrc-001d) Zaak aanmaken is mogelijk
 - PUT** (zrc-001e) Zaak bijwerken is mogelijk (PUT)
 - PATC** (zrc-001f) Zaak deels bijwerken is mogelijk (PATCH)
 - DEL** Delete created Zaak
- zrc-002
- zrc-003

POST {{zrc_url}}/zaken

Params Authorization Headers (3) Body Pre-request Script Tests

```
1 pm.test("Zaaktype url die niet resulteert in 200 bij aanmaken zaak geeft 400", function() {  
2   pm.response.to.have.status(400);  
3  
4   // Verify that the 400 is returned for the correct reason  
5   var error = pm.response.json()["invalid-params"][0];  
6   pm.expect(error.name).to.be.equal("zaaktype");  
7   pm.expect(error.code).to.be.equal("bad-url");  
8 });  
9
```

Response

Regressie tests in Postman

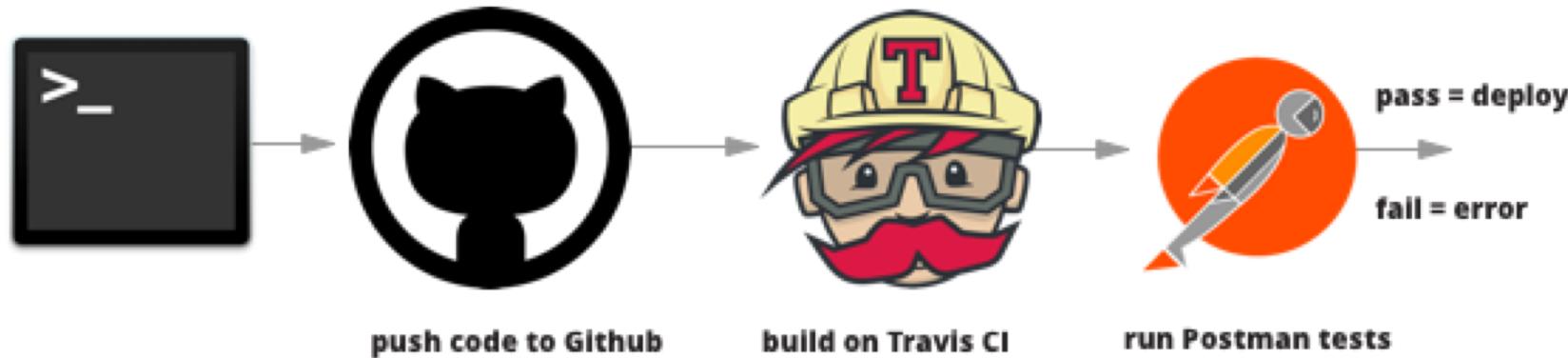
Test scenario's die een bug aantonen en daardoor **falen**.
Als de bug is opgelost, slaagt het test scenario pas.

▼	Regression tests	...
▼	setup	...
	GET Retrieve Zaaktype	●
	GET Retrieve EnkelvoudigInformatieObjecten	●
▼	Issue #891	...
	POST Create Zaak	●
	POST Create ObjectInformatieObject relation	●
	DEL (NI) Delete ObjectInformatieObject relation	✗
	GET (NI) Validate informatieobject deletion ZRC	—
	DEL Delete Zaak	—

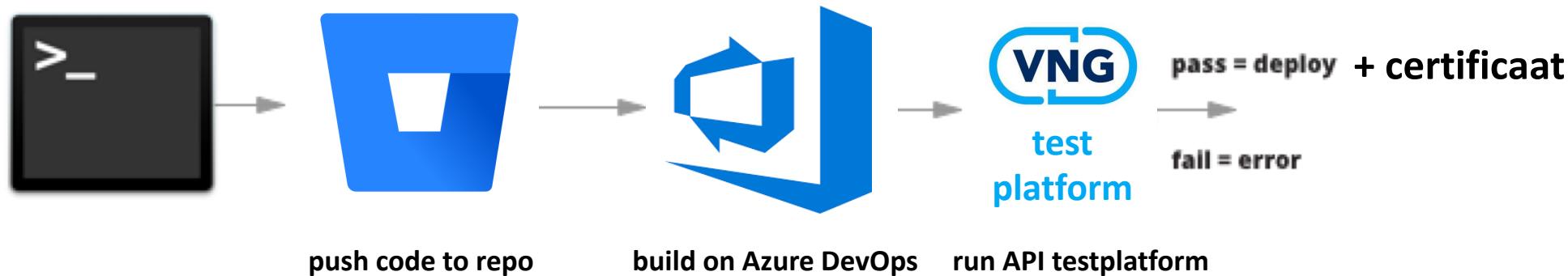
Issue #891

Lukt niet om een zaak aan een document te koppelen en weer te verwijderen.

Continuous integration (CI) flow



<https://blog.getpostman.com/2017/08/23/integrate-api-tests-with-postman-newman-and-travis-ci/>



Status

- 75% van de ZRC standaard is afgedekt middels test scenario's.
- ZTC, DRC, BRC en NRC moeten nog gedaan worden.