

Elektronski fakultet, Niš

Smer: Računarstvo i informatika, Softversko inženjerstvo

**Predmet: Sistemi za upravljanje bazama podatka**

Tema:

**RavenDB - Backup/restore baze podataka**

Student: Dušan Dimitrov 1428

Mentor: Doc. dr Aleksandar Stanimirović

Niš, januar 2023.

**Sadržaj**

[Uvod 3](#_Toc125310491)

[RavenDB baza podataka 3](#_Toc125310492)

[RQL-Raven Query Language 4](#_Toc125310493)

[Uvod u backup/restore 4](#_Toc125310494)

[Kako radi backup kod RavenDB baze podataka 5](#_Toc125310495)

[Ko obavlja backup? 8](#_Toc125310496)

[Dupli backup 9](#_Toc125310497)

[Backup/restore praktična demonstracija 9](#_Toc125310498)

[Pokretanje RavenDB baze podataka 10](#_Toc125310499)

[Pokretanje RavenDB preko Docker-a 10](#_Toc125310500)

[Pokretanje lokalno 11](#_Toc125310501)

[Kreiranje baze podataka 12](#_Toc125310502)

[Backup/restore 13](#_Toc125310503)

[Backup i dostupnost 18](#_Toc125310504)

[Zaključak 19](#_Toc125310505)

[Literatura 19](#_Toc125310506)

# Uvod

Podaci predstavljaju jako bitnu komponentu gotovo svakog procesa i biznisa, tako da njihovo skladištenje igra jako veliku ulogu. Gubitak podataka može dovesti do puno problema, a i neke procese do samog propadanja. Jako je bitno sačuvati podatke od gubitka. Jedan od načina za očuvanje podataka je replikacija. Replikacijom se dupliraju određeni podaci i skladište na različitim čvorovima. Ovako je sistem sačuvan od prestanka rada određenog čvora. Ukoliko jedan čvor prestane sa radom, podatak je dostupan na nekom drugom čvoru gde je replika podatka. Međutim, ukoliko je čitav sistem kompromitovan i svi čvorovi otkažu trajno, replikacija neće pomoći. U tom slučaju su podaci trajno izgubljeni.

Kako bi se predupredilo trajno gubljene podataka, može se primeniti posebno “kopiranje“ podataka na neku drugu lokaciju. Ovo predstavlja proces backup-ovanja. Restore procesom se podaci iz backup-a mogu povratiti.

U nastavku ovog rada će biti opisan backup/restore proces kod RavenDB baze podataka. Nakon uvodnog poglavlja, drugo poglavlje daje uvod u svet RavenDB document baze podataka. Treće poglavlje predstavlja teorijsku osnovu samog procesa backup/restore i tako daje osnovu za četvrto poglavlje koje se bavi backup/restore procesom kod RavenDB baze podataka. U petom poglavlju je data detaljna praktična demonstracija backup/restore procesa kod RavenDB baze podataka kroz Raven Studio. Šesto poglavlje pravi paralelu između backup-a i dostupnosti podataka. U sedmom poglavlju je zaključak rada, a u osmom, poslednjem poglavlju je dat spisak korišćene literature.

# RavenDB baza podataka

RavenDB predstavlja jednu distribuiranu NoSQL dokument bazu podataka sa jako dobrim performansama.

Sami podaci baze podataka se nalaze u strukturi podataka koja se zove **dokument**, a podaci su najčešće u **JSON** ili (ređe) u XML formatu. RavenDB pruža mogućnost skladištenja binarnih, ali i ostalih tipova podataka, međutim, najčešće se radi sa JSON dokumentima.

RavenDB baza podataka se može izvršavati na **jednom čvoru**, ali može i na **cluster-u** čvorova.

Nešto što izdvaja RavenDB bazu podataka od ostalih NoSQL baza podataka su definitivno **ACID transakcije**. ACID predstavlja:

* Atomic (atomičnost)
* Consistent (konzistentnost)
* Isolation (izolacija)
* Durability (trajnost). [0]

**Atomic (atomičnost)** predstavlja princip “sve ili ništa”. Ili će se desiti sve operacije iz jedne transakcije, ili neće nijedna. **Consistent (konzistentnost)** znači da transakcija prevodi bazu podataka iz jednog konzistentnog stanja u drugo, ne postoje neka među stanja. **Isolation (izolacija)** znači da nijedna transakcija nema pristup nekoj drugoj transakciji koja nije završena. **Durability (trajnost)** - završetak transakcije garantuje trajnost podataka, odnosno nijedan pad sistema ne može dovesti do gubitka podataka.

## RQL-Raven Query Language

Server RavenDB baze podataka podržava upite, a sama baza ima svoj upitni jezik **RQL (Raven Query Language)** koji podseća dosta na SQL.

Čim RQL upit dođe do RavenDB instance, instanca baze podataka poziva optimizator upita (**query optimizer**). On analizira upit i zaključuje koji indeks se treba koristiti kako bi se dobio traženi podatak.

RavenDB poseduje 2 tipa upita:

1. Dinamički upit-daje potpunu kontrolu optimizatoru upita da izabere koji indeks da koristi za upit.
2. Indeksiran upit-eksplicitno se specificira koji indeks da se koristi.

Ključne reči koje RQL podržava su:

* declare-služi za kreiranje JavaScript funkcije koja se može pozvati iz select naredbe.
* from-određuje “izvor” podataka za upit
* group by-grupiše i kreira agregacioni upit
* where-služi za filtriranje dokumenata
* order by-služi za sortiranje
* load-koristi se kada je potrebno ubaciti eksterne dokumente u projekciju
* select-određuje projekciju upita
* update-ažurira dokument
* with-koristi se u graf upitima kako bi se odredio izvod podataka
* match-koristi se kako bi se utvrdio pattern graf upita. [7]

# Uvod u backup/restore

**Backup** predstavlja jako bitan proces kod baza podataka i predstavlja kreiranje kopija podataka nekog sistema. U slučaju gubitka originalnih podataka, backup podaci se oporavljaju putem **recovery** procesa. [1]

Generalno postoje 2 osnovna razloga za kreiranje backup-a:

1. Restore baze nakon gubitka podataka
2. Oporavljanje podataka iz nekog ranijeg trenutka. [2]

Prvi razlog, odnosno restore podataka nakon gubitka je neophodan zato što je možda došlo do brisanja pogrešne baze podataka, disk ili čitav server se pokvario,…Oporavljanje podataka iz nekog ranijeg trenutka je neophodno za slučaj da je neki čvor prestao da radi.

Slučaj brisanja pogrešne baze podataka je znatno strašniji. U tom slučaju se čitava baza obrisala, sa svih čvorova u klasteru, sa svih kopija i bukvalno ti podaci više nisu na raspolaganju. Tada nam može pomoći samo takozvana **offsite replica**, odnosno neki eksterni backup koji smo napravili pre gubitka originalnih podataka. Srećom, ovo je ređa situacija, odnosno, češće će se oporavljati podaci iz nekog ranijeg trenutka.

Sam postupak backup/restore je jako spor i vremenski zahtevan. Posmatrajmo na primer jednu “prosečnu” bazu podataka od 250GB. Samo kopiranje podataka iz backup-a na mašinu gde želimo da obavimo restore traje jako dugo. Bez uzimanja u obzir troškova dekompresije i transporta, za ovakvu jednu bazu bi bilo neophodno 10ak minuta samo za kopiranje podataka iz backup-a na lokalnu mašinu i još jedno 10ak minuta za pravi restore.

Na prvi pogled može izgledati prihvatljivo da se backup RavenDB baze podataka odradi na nivou fajl sistema, odnosno samim kopiranjem direktorijuma baze podataka na neko drugo mesto. Iako ovo zvuči izvodljivo, zapravo nije i nije podržano kod RavenDB baze. RavenDB poseduje niz specijalnih instrukcija sa fajl sistemom kako bi se održalo ACID svojstvo. Kopiranje direktorijuma uglavnom neće uzeti u obzir sve realne podatke RavenDB baze podataka. Zbog toga RavenDB poseduje svoj sopstveni mehanizam obavljanja backup/restore procesa koji se treba koristiti i nakon teorijske osnove u nastavku rada će biti objašnjen i demonstriran.

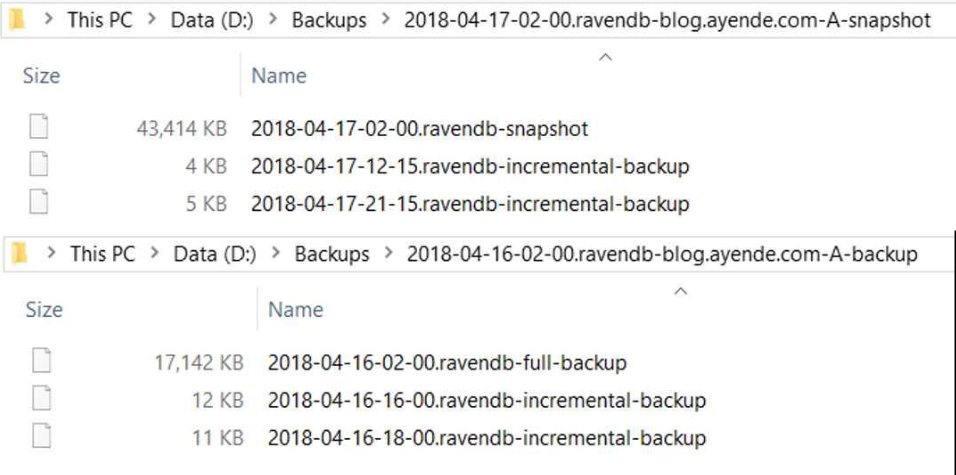
# Kako radi backup kod RavenDB baze podataka

Backup se uglavnom čuva duži vremenski period (godinama). Zbog toga je jako bitna veličina samog backup-a. Standardna opcija backup-a kod RavenDB-a je korišćenje **gzippovanih JSON podataka svih dokumenata i ostalih podataka**. Gzip predstavlja file format i softver za kompresiju i dekompresiju. [3] Kod RavenDB baze podataka, skladištenje backup-a u gzip formatu pruža najmanju moguću veličinu za podatke i tako je jako efikasno čuvati backup fajlove. Sa druge strane, kada je potrebno raditi restore, RavenDB će morati da dekompresuje, ponovo unese i indeksira podatke. To definitivno povećava vreme potrebno za restore baze podataka. Međutim, kako je češća situacija za backup-om, nego za restore-om (u normalnim uslovima i sistemima), ovo produženo vreme ne predstavlja veliki problem, ali ga definitivno treba biti svestan.

Pored gzipovanja svih podataka, RavenDB podržava još jednu opciju backup-a, a to je **snapshot**. Snapshot predstavlja binarnu kopiju baze i journal-a u određenom trenutku. Journal predstavlja jedan fajl gde RavenDB piše sve promene koje su se desile u okviru jedne transakcije i tako omogućava da su podaci skladišteni na perzistentan način. [4] Kao i regularni backup i snapshot-ovi su kompresovani. Osim toga, snapshot je maltene spreman da se iz njega ekstrakuju podaci i journal fajlovi i da se sama baza podataka normalno pokrene. Prednost snapshot-a je to što je kod njega restore proces dosta brži. Sa druge strane, mana snapshot-a je to što on zauzima više prostora nego regularni backup. U praksi se koriste i snapshot-ovi i regularni backup-ovi. Regularni backup se koristi za takozvane long-term storage, gde brzina restore-a nije presudna, a snapshot se koristi kako bi se momentalno obavio restore u nekim kritičnim situacijama.

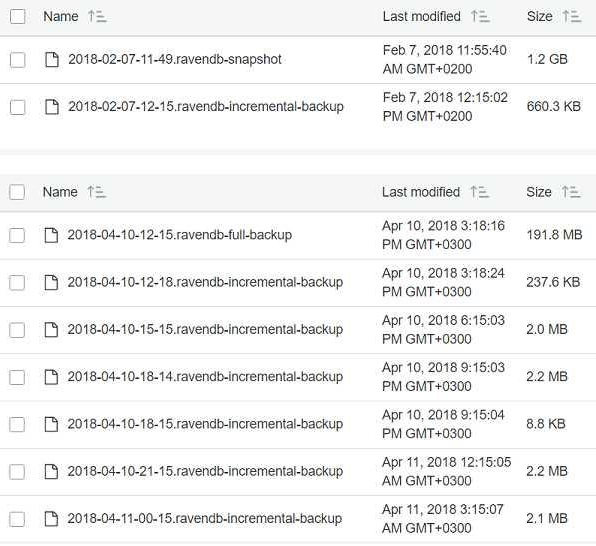
I regularni backup i snapshot su kompletni klonovi baze podataka iz određenog vremenskog trenutka. Međutim, postoje situacije gde nije neophodno kreirati kompletan backup. Ponekad je dovoljno samo čuvati izmene koje su se desile nakon prethodnog backup-a. Ovu opciju takođe podržava RavenDB i zove se **inkrementalni backup**. Dostupan je i za regularni backup, ali i za snapshot-ove. U oba slučaja se koristi gzipovani JSON.

Glavni razlog za postojanje inkrementalnog backup-a je taj što se tako smanjuje vreme za kreiranje backup-a. Osnivač RavenDB-a u svojoj knjizi daje primere realnih baza podataka u produkciji i njihovih backup-ova. [2] Na sledećoj slici je moguće videti različite tipove backup-a RavenDB baze podataka koja zauzima 790MB:



U gornjem delu slike je backup kreiran snapshot metodom, a u donjem delu slike je backup kreiran regularnom metodom. Što se tiče snapshot metode, kreiran je jedan kompletan snapshot backup i jos 2 inkrementalna backup-a. To je isto urađeno i za regularni metod backup-a. Možemo primetiti, kao što je i ranije opisano da kompletni snapshot zauzima više prostora nego kompletni backup, dok inkrementalni backup ne zauzima toliko puno prostora.

Zanimljivo je posmatrati backup neke veće baze u produkciji. Na sledećoj slici je moguće videti backup jedne RavenDB baze koja zauzima 14GB.



Kako je sama RavenDB baza podataka veća, tako je i backup veći, a na slici se može videti i to da je sada još veća razlika u veličini snapshot i regularnog backup-a.

Na veličinu inkrementalnog backup-a više utiče broj dokumenata koji je promenjen u odnosu na prethodni backup, nego broj upisa. Naime, ukoliko je samo jedan dokument promenjen hiljadu puta, nakon inkrementalnog backup-a, poslednja verzija dokumenta će se upisati u backup. Sa druge strane, ukoliko se hiljadu dokumenata promeni, biće neophodno upisati svih hiljadu u backup, pa ova razlika može uticati znatno na razliku u veličini inkrementalnog backup-a.

Nad RavenDB bazom podataka je moguće definisati i takozvane ETL taskove. ETL (Extract, Transform, Load) je RavenDB proces koji se sastoji iz 3 faze. Služi za transformaciju podataka iz RavenDB baze na neki eksterni tok podataka-što može biti neka druga RavenDB baza, ili neka relaciona baza podataka, ili nešto treće. [5] Kada se radi backup RavenDB baze podataka, ovi ETL taskovi će se takođe backup-ovati, a kasnije i restore-ovati. Međutim, treba biti jako pažljiv sa njima i disable-ovati ih. Na primer, ukoliko se restore-ovao neki backup na development masinu sa produkcione baze podataka, restore-ovaće se i ETL task koji ovde na development mašini nije potreban. Uglavnom, development masine nisu podesene da mogu da pristupaju nekim produkcionim izvorima podataka. Međutim ukoliko mogu da pristupaju, ETL taskovi mogu napraviti problem. ETL task će onda izvrišiti operacije upisa i na taj način se može naškoditi produkcionoj bazi podataka-bespotreban upis/upisi su obavljeni.

Problemi mogu nastati i sa drugim taskovima koji su definisani nad RavenDB bazom podataka-replikacija na primer. Restore će takođe uključiti i ove taskove. Dobra stvar je što sam RavenDB, kao što ćemo videti i u praktičnoj demonstraciji, omogućava disable-ovanje svih taskova kada se radi restore. Tako se postiže takozvani **clean restore**. Nakon toga je neophodno ručno ponovo enable-ovati taskove koji su potrebni i koji nisu zavisni od okruženja.

Neophodno je razmatrati i backup enkriptovane RavenDB baze podataka. RavenDB se može pokrenuti u 2 moda:

* secured
* unsecured.

Kod unsecured moda podaci nisu enkriptovani, dok kod secured jesu. [6] Kada se koristi secured mod, treba biti svestan da će snapshot backup zadržati podatke iz baze u enkriptovanom obliku. Međutim, informacije vezane za cluster (čvorovi, topologija, taskovi, istorija health check-a), koje su enkriptovane u originalnoj bazi (nebackup-ovanoj), ovde u backup-u neće biti enkriptovane. Takođe, regularni backup se nikada neće čuvati u enkriptovanom obliku, bez obzira na činjenicu da li je RavenDB baza enkriptovana (u secured modu), ili nije (u unsecured modu).

## Ko obavlja backup?

RavenDB baza podataka je uglavnom hostovana na više čvorova u cluster-u. Kada se govori o backup-ovanju jednog ovakvog sistema, neophodno je postaviti pitanje: **“Ko zapravo obavlja backup?”**. Određeni delovi baze podataka, čvorovi, mogu sadržati iste podatke zbog **faktora replikacije**. Došlo bi do velike pometnje kada bi svaki čvor, sa svojom kopijom, kreirao backup. To bi dovelo do kopije podataka i uzaludnog trošenja resursa.

Sam cluster donosi odluku o tome koji će čvor biti **vlasnik backup task-a**, odnosno koji čvor će obavljati backup proces. Cluster bira čvor koji je aktivan za vlasnika backup taska, ali takođe zna da prebaci zaduženje za backup na neki drugi čvor, ukoliko inicijalni vlasnik backup-a nije dostupan-privremeno/trajno ne radi.

Kod potpunog backup-a nije bitno koji će čvor obavljati backup proces. Bilo koji čvor može da obavi backup njegovog kompletnog stanja. Ukoliko cluster odluči da će neki drugi čvor obaviti naredni backup, nije nikakav problem, njegov backup će se markirati adekvatno i dodeliće se timestamp.

Međutim, kod inkrementalnog backup-a je situacija malo drugačija. Moguće je primeniti inkrementalni backup samo nad kompletnim backup-om ukoliko je oba backup-a napravio isti čvor. Ukoliko cluster odluči da je neophodno promeniti čvor za obavljanje inkrementalnog backup-a, novi čvor neće izvršiti inkrementalni backup. Novi čvor će morati da obavi kompletan backup, a nakon toga će moći da obavlja inkrementalne kada treba.

Posmatrajmo RavenDB cluster sa 3 čvora: A, B i C. Neka je ova RavenDB baza konfigurisana tako da obavlja full backup svakog dana u ponoć i inkrementalni backup na 4h. Cluster je (na primer) odlučio da čvor C bude vlasnik backup taska. Dolazi ponoć i on obavlja full backup. Nakon toga u 04:00 i 08:00 obavlja još 2 inkrementalna backup-a. Međutim, u podne, u 12:00, kada treba da obavi svoj naredni inkrementalni backup, čvor C prestaje sa radom. Tada cluster bira čvor A (na primer) da on obavlja backup. Međutim, sada čvor A ne može da obavi inkrementalni backup “preko” full backup-a čvora C. Zato, čvor A u 12:00 obavlja full backup. U 16:00, kako je čvor C idalje odsutan (pokvaren), čvor A obavlja backup. Međutim, sada on obavlja inkrementalni backup sa svim promenama koje su se desile nakon 12:00. U 18:00, čvor C postaje ponovo aktivan, tako da cluster, koji je inicijalno izabrao njega da obavlja backup, vraća čvoru C odgovornost za backup. Dolazi 20:00 i čvor C treba da obavi backup. Međutim, on će sada obaviti samo inkrementalni backup, ali sa izmenama koje su se desile nakon 08:00 kada je on kreirao svoj poslednji backup.

Kod procesa obavljanja backup-a, neophodno je voditi računa o nekim stvarima:

* Samo čvorovi koji su **“up to date”** mogu biti adekvatni kandidati za obavljanje backup-a. U malopre opisanom primeru sa 3 čvora, čvor C je bio nedostupan, tako da ga je cluster video kao da nije “up to date”. Cluster mu je vratio zaduženje za obavljanje backup-a, tek kada se on sinhronizovao sa promenama koje su se desile dok on nije bio dostupan.
* Zakazan backup se deli između čvorova u cluster-u. Odnosno, činjenica da je čvor koji obavlja backup nedostupan, neće automatski pokrenuti backup u drugom čvoru. Samo nakon određenog vremena, kada dođe vreme za novi backup, cluster pokreće backup na drugom čvoru.
* Inkrementalni backup se primenjuje samo na kompletni backup istog čvora. Tako se postiže mogućnost za obavljanje kompletnog restore-a sa jednog čvora.

### Dupli backup

RavenDB vodi računa o tome da samo jedan čvor obavlja backup određenog podatka kako se ne bi bespotrebno koristili resursi. Međutim u određenoj situaciji se može ipak desiti da su 2 čvora kreirala (identičan) backup za određene podatke. Ovo se može desiti ukoliko postoji **problem sa mrežom** gde je čvor koji treba da obavi backup funkcionalan, ali zbog problema sa mrežom ne može lepo da iskomunicira sa cluster-om i potvrdi svoju aktivnost. U takvoj situaciji, cluster će dodeliti drugom čvoru zadatak da obavi backup nad podacima koje on poseduje. Međutim, zbog problema sa mrežom, inicijalni čvor za backup neće dobiti informaciju da se promenio odgovorni čvor za backup. U ovoj situaciji će i inicijalni i novododeljeni čvor da obave backup. Ovakva situacija je prihvatljiva zato što su backup-ovi jako bitni i nije toliko strašno ukoliko se ponekad desi da se uradi dupli backup. Bolje je imati 2 identična backup-a, nego nijedan. RavendB dodeljuje određene identifikatore svim čvorovima (**node ID**), pa se u situaciji sa duplim backup-ovima može razlikovati koji čvor je obavio koji backup.

# Backup/restore praktična demonstracija

Kako bi se demonstrirala upotreba backup/restore procesa, neophodno je prvo podići RavenDB bazu podataka, kreirati bazu koju ćemo backup-ovati i restore-ovati je. Najpre će biti demonstrirano pokretanje, kreiranje i na kraju backup/restore.

## Pokretanje RavenDB baze podataka

Postoje više različitih načina za instalaciju RavenDB baze podataka. To je moguće uraditi preko Docker-a, ali i lokalno.

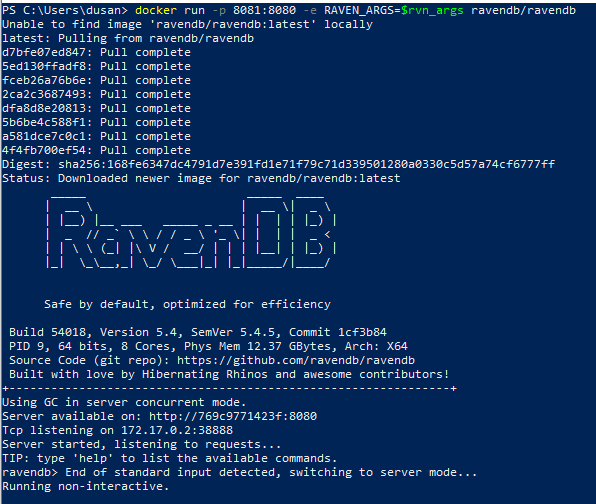
### Pokretanje RavenDB preko Docker-a

Za instalaciju preko Docker-a, potrebno je pokrenuti sledeće komande u PowerShell-u. Najpre je potrebno definisati promenljivu koja će čuvati argumente za RavenDB:

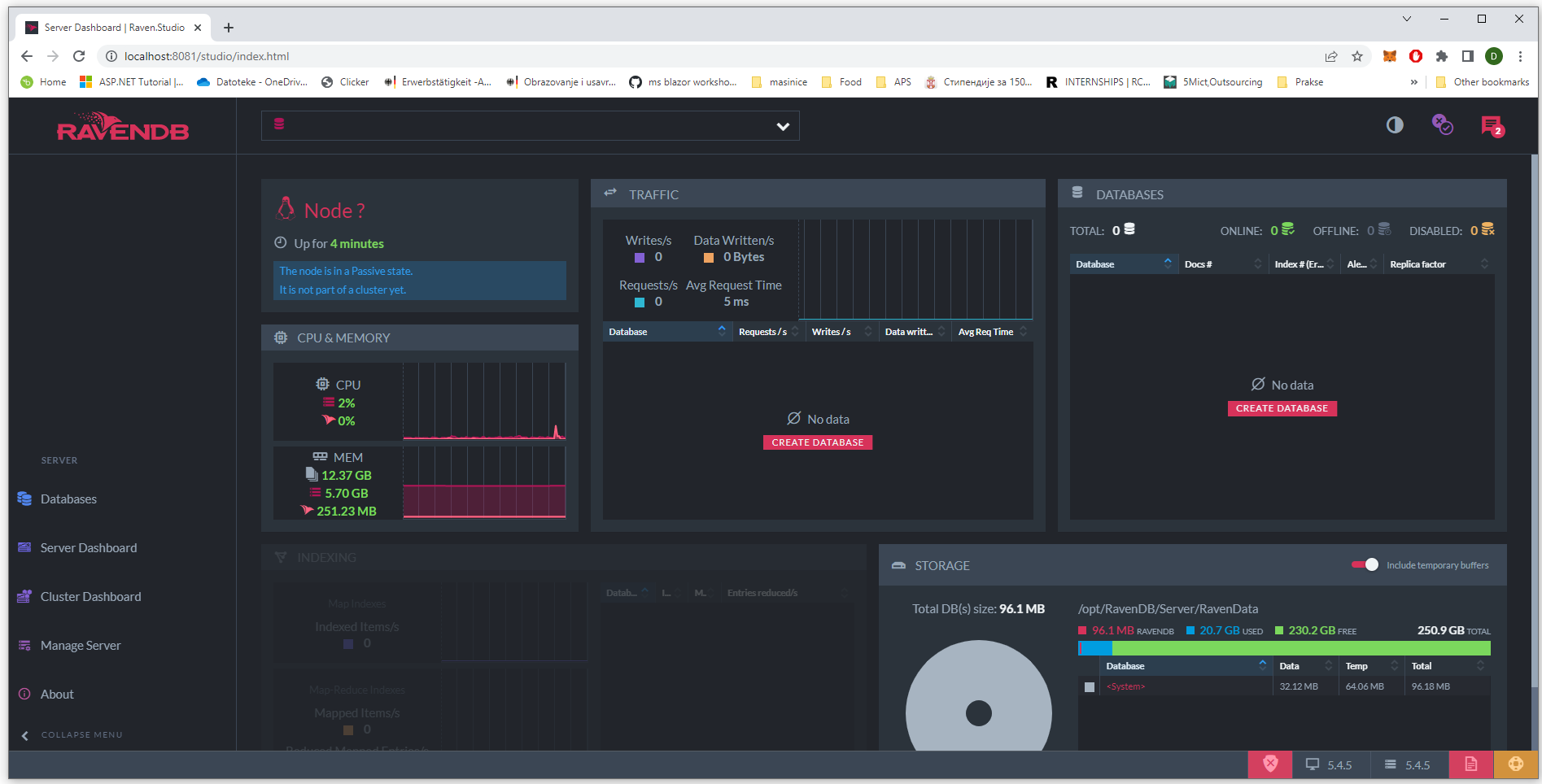


Nakon toga, moguće je pokrenuti docker kontejner na osnovu slike ravendb/ravendb, na osnovu komande:

* **docker run -p 8081:8080 -e RAVEN\_ARGS=$rvn\_args ravendb/ravendb**

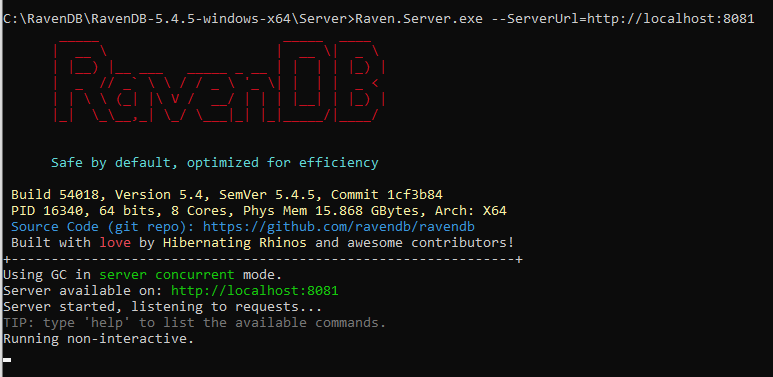
****

U ovoj slučaju je korišćena najnovija RavenDB slika (latest), a korišćenje promenljive rvn\_args nije neophodno, ali povećava čitljivost komande. Nakon pokretanja komande je moguće pristupiti instanci RavenDB baze podataka preko localhosta i odgovarajućeg porta definisanog prilikom pokretanja kontejnera: <http://localhost:8081>. Tako se pokreće Raven Studio, što je moguće videti na sledećoj slici:



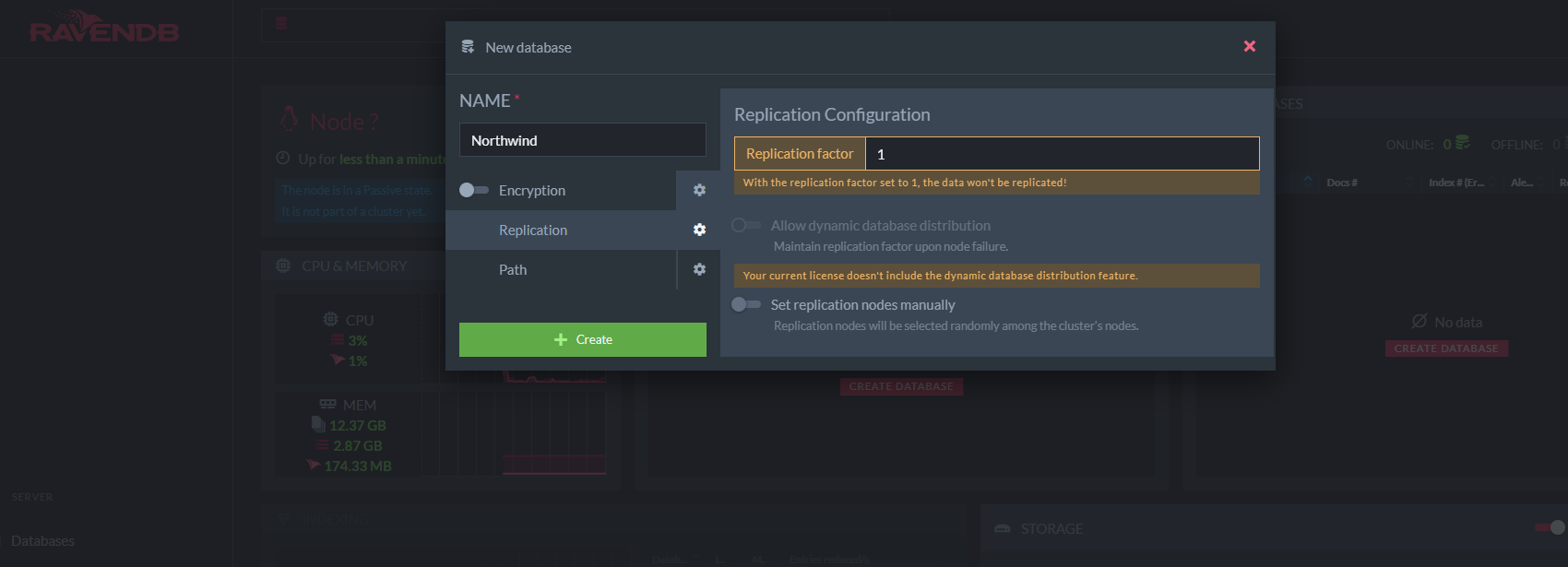
### Pokretanje lokalno

Da bi se pokrenula RavenDB baza podataka lokalno na Windows operativnom sistemu, neophodno je download-ovati odgovarajući zip fajl sa <https://ravendb.net/download>. Nakon ekstrakovanja zip fajla, potrebno je pokrenuti Raven.Server.exe fajl i definisati url na kom će biti. Ovo je moguće videti na sledećoj slici:



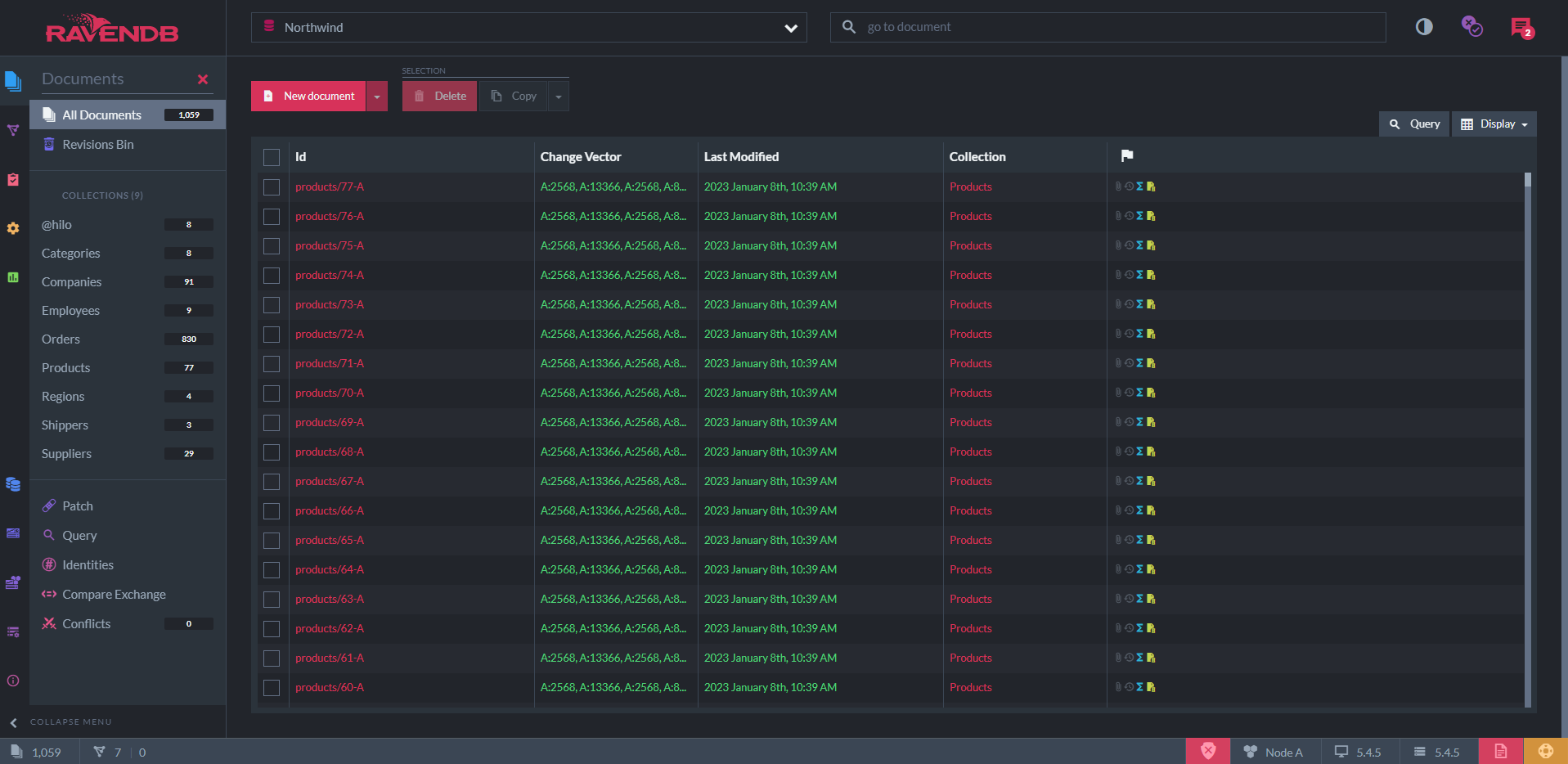
## Kreiranje baze podataka

Samu bazu podataka je moguće kreirati iz Raven Studio-a klikom na dugme ***Create Database***. Nakon toga se otvara dijalog, gde je moguće izabrati naziv baze podataka. Nazvaćemo je **Northwind** zato što ćemo koristiti već generisane podatke iz poznate demo baze podataka-Northwind. To je set podataka jedne online prodavnice koja skladišti osnovne podatke o narudžbini, kupcima, proizvodima,...Što se tiče ostalih opcija prilikom kreiranja instance RavenDB baze podataka, nećemo uključiti enkripciju, replikacioni faktor ćemo postaviti da bude 1. Ovo je moguće videti na sledećoj slici.



Nakon klika na dugme ***Create***, uspešno će se kreirati nova instanca RavenDB baze podataka, što se može videti odlaskom na databases stranicu izborom iz menija sa leve strane.

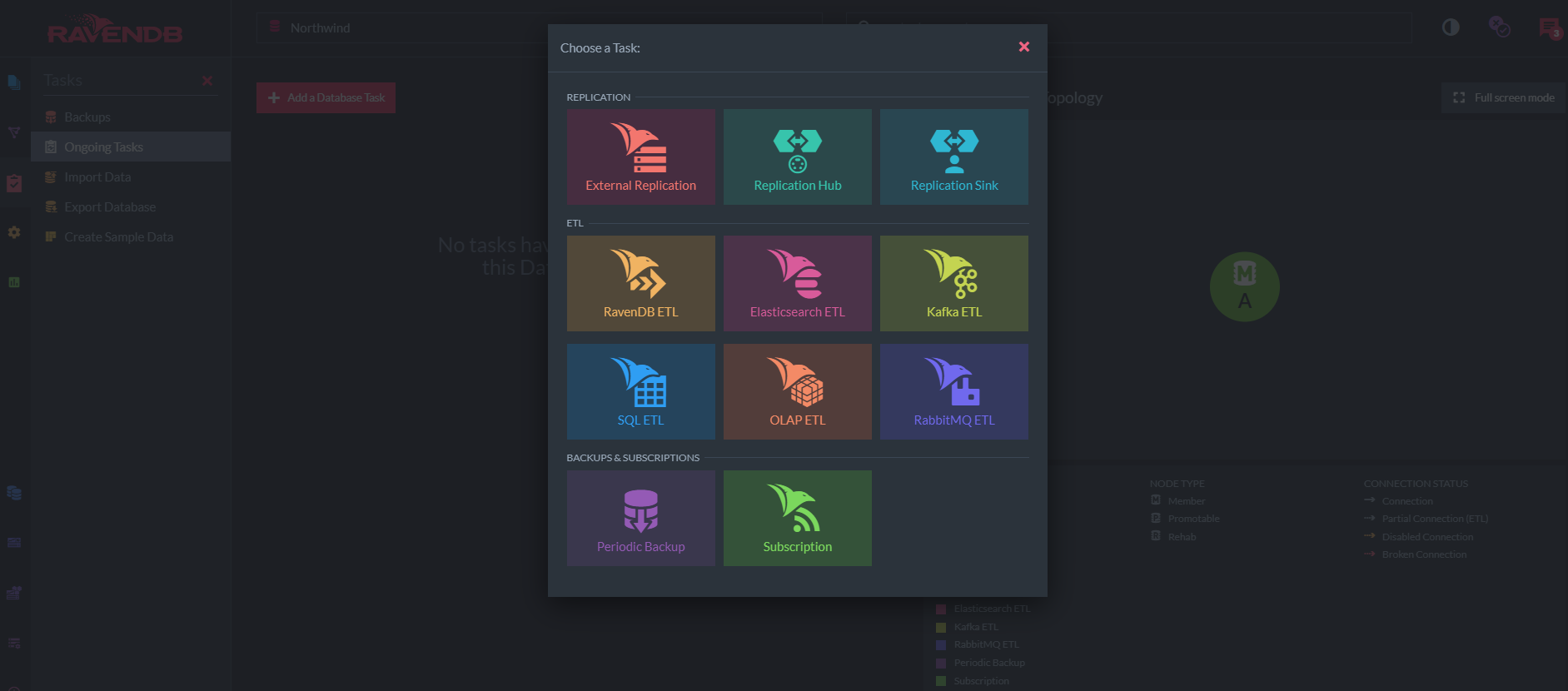
Kao što smo rekli, koristiće se takozvani sample podaci za ovu bazu podataka. Da bi se oni kreirali potrebno je kreirati jedan Task za insertovanje podataka. Iz menija sa leve strane u Raven Studio-u je potrebno kliknuti na ***Task*** i kao tip taska izabarti: ***Create Sample Data***. Nakon toga se otvara meni gde je potrebno kliknuti na dugme ***Create***. Potrebno je par sekundi da bi se insertovali svi podaci u RavenDB. Nakon toga je moguće posetiti documents stranicu (iz menija sa leve strane Raven Studio-a) i videti sve dokumente:



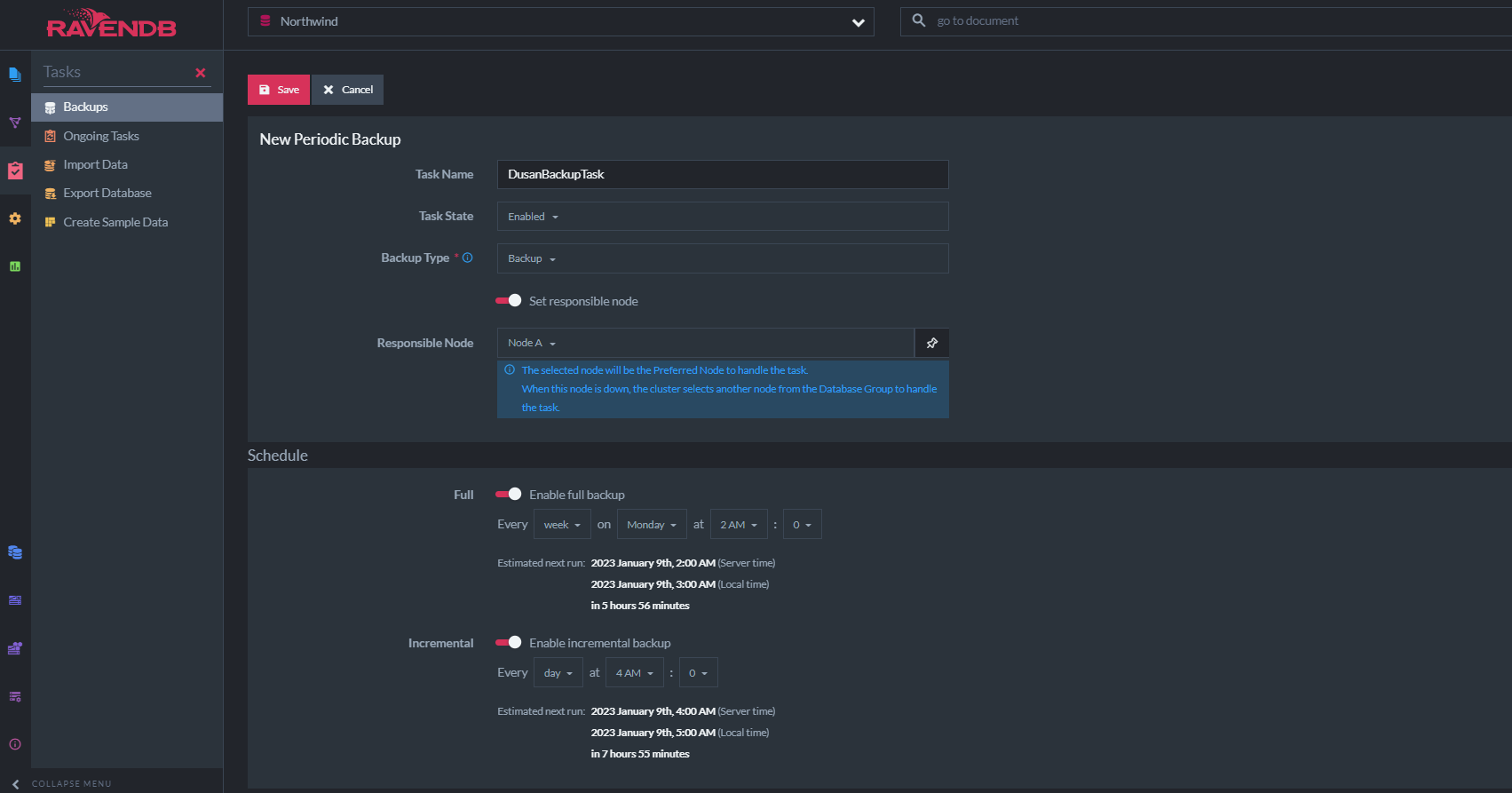
Na prethodnoj slici je moguće videti nazive svih kolekcije, kao i koliko dokumenata se nalazi u svakoj kolekciji.

## Backup/restore

Preko Raven Studio-a je moguće kreirati backup za RavenDB bazu podataka. Kao što je više puta u radu pomenuto, backup predstavlja **task** nad RavenDB bazom podataka, pa je potrebno i kreirati taj task. Iz menija sa leve strane, neophodno je kliknuti na **Tasks**, pa na **Ongoing Tasks**. Klikom na dugme **Add a Database Task** se otvara meni sa sledeće slike:



Potrebno je izabrati tip taska: ***Periodic Backup***. Nakon toga se otvara meni u kome je potrebno podesiti određene parametre:

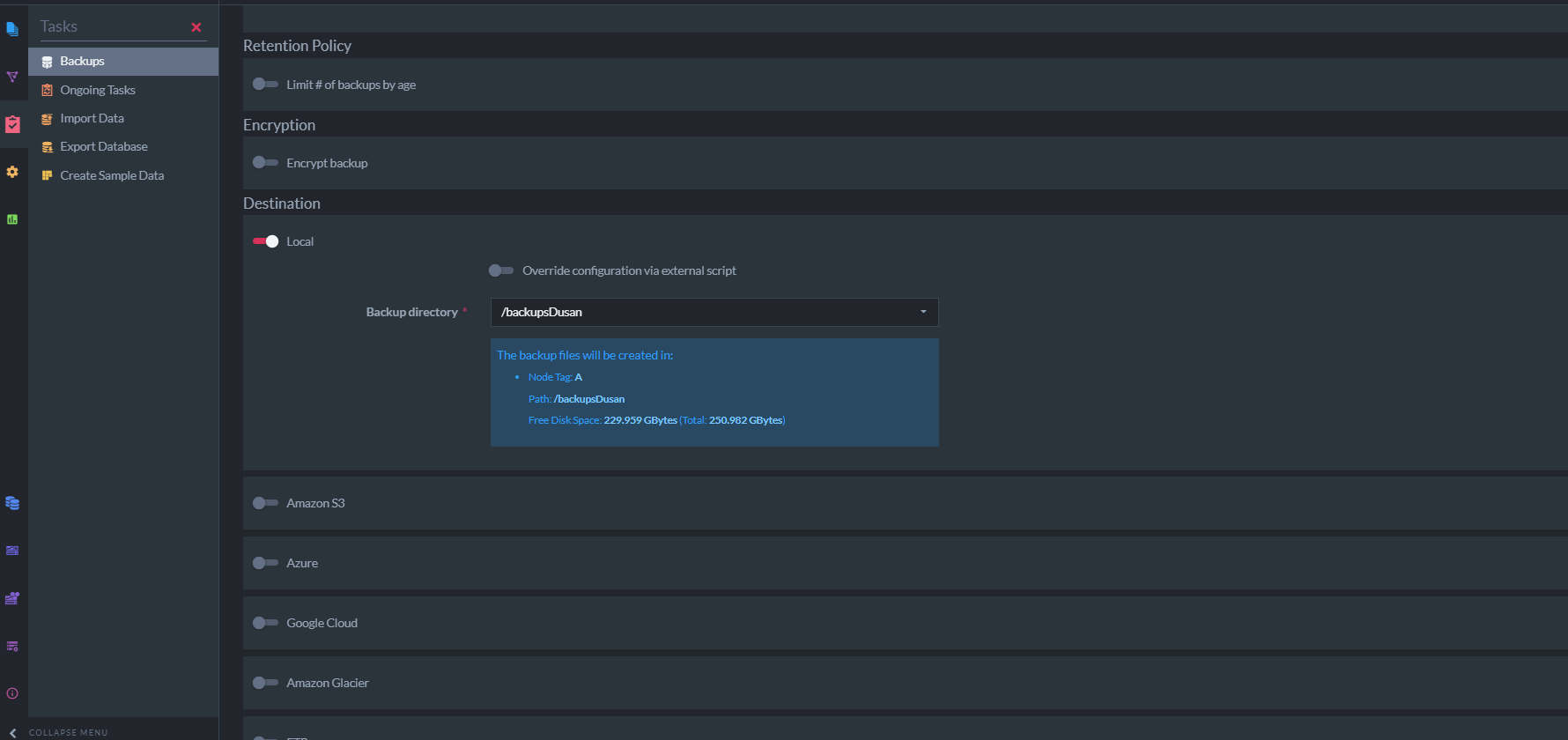


Moguće je opciono dodeliti ime backup tasku **Task Name**. Ovaj task je nazvan DusanBackupTask. Moguće ga je kreirati aktiviranog (enabled) ili ne (disabled) i to predstavlja **Task State**. Tip backup-a (**Backup Task**)može biti backup ili snapshot. Backup predstavlja kompletni backup koji je opisan u prethodnom teorijskom delu. Neophodno je izabrati i čvor zadužen za obavljanje backup-a **Responsible Node**. U ovom konkretno slučaju je moguće izabrati samo Node A, zato što samo on postoji. U realnom cluster-u bi postojao izbor između više prisutnih čvorova. Na ovom konkretno primeru je dozvoljen full backup (**Enable full backup**). Izabrano je da se full backup obavlja jednom nedeljno, svakog ponedeljka u 02:00. Dozvoljen je i inkrementalni backup (**Enable incremental backup**). Inkrementalni backup se u ovom primeru obavlja svakog dana u 04:00. Ukoliko se samo ovako kreira task, backup se zapravo neće obaviti zato što nije izabrano odredište za backup. Neophodno je definisati gde će RavenDB kreirati backup. RavenDB podržava više različitih odredišta za backup:

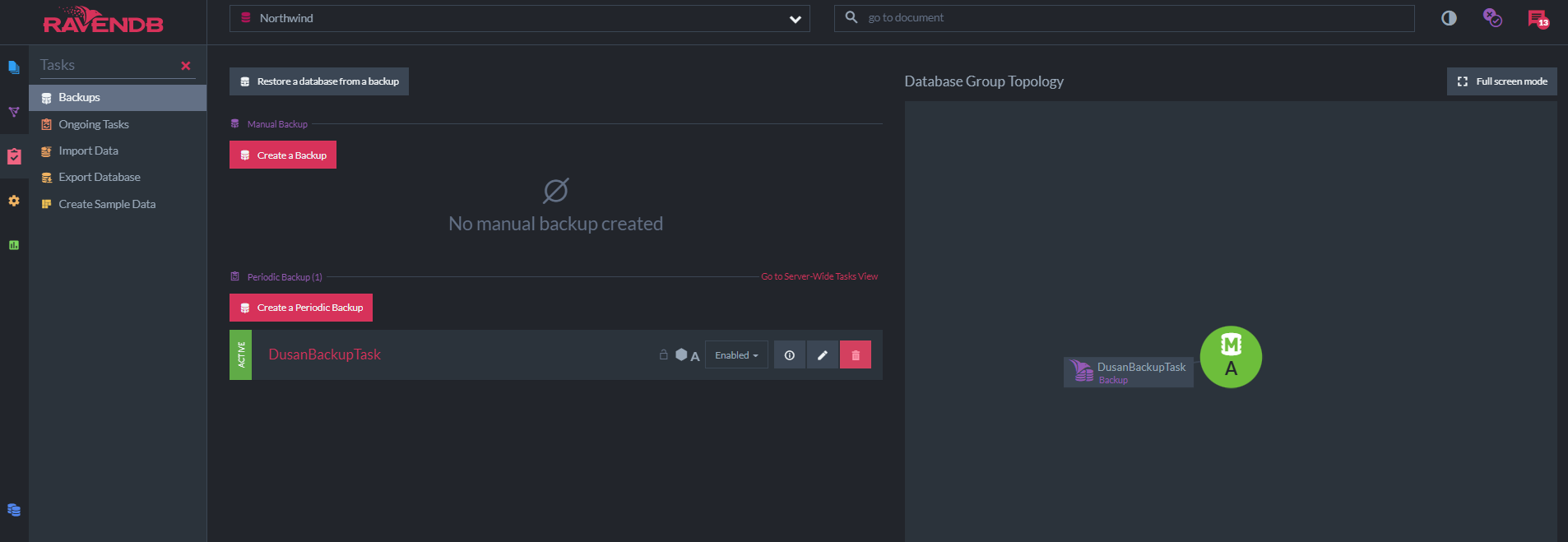
* Lokalno
* FTP/SFTP
* Amazon S3
* Azure Storage
* Amazon Glacier.

RavenDB podržava više paralelnih odredišta za backup. Sam proces backup-a će najpre obaviti backup (upisati podatke) lokalno. Kada to bude gotovo, obaviće se i backup ka svim remote odredištima paralelno. Bitno je napomenuti da ukoliko makar jedan backup na nekoj remote lokaciji ne uspe, smatra se da čitav backup nije uspeo.

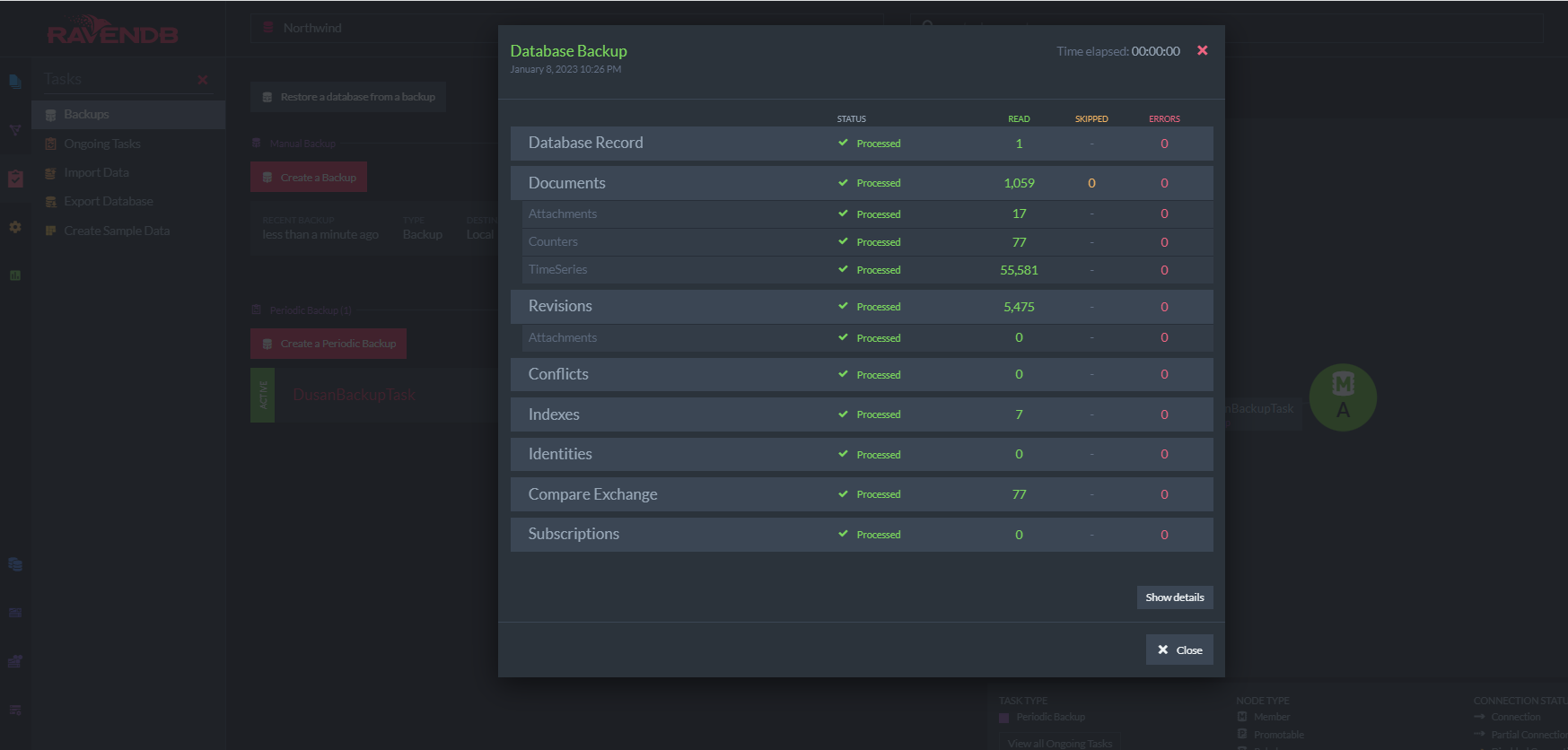
Na sledećoj slici je moguće videti podešavanje lokalnog odredišta backup-a (kao folder je izabran /backupsDusan):



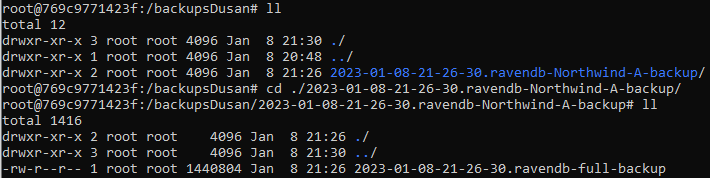
Nakon toga je moguće završiti kreiranje backup taska klikom na dugme **Save**. Backup task se prikazuje u listi Backup taskova kao što je moguće videti na sledećoj slici.



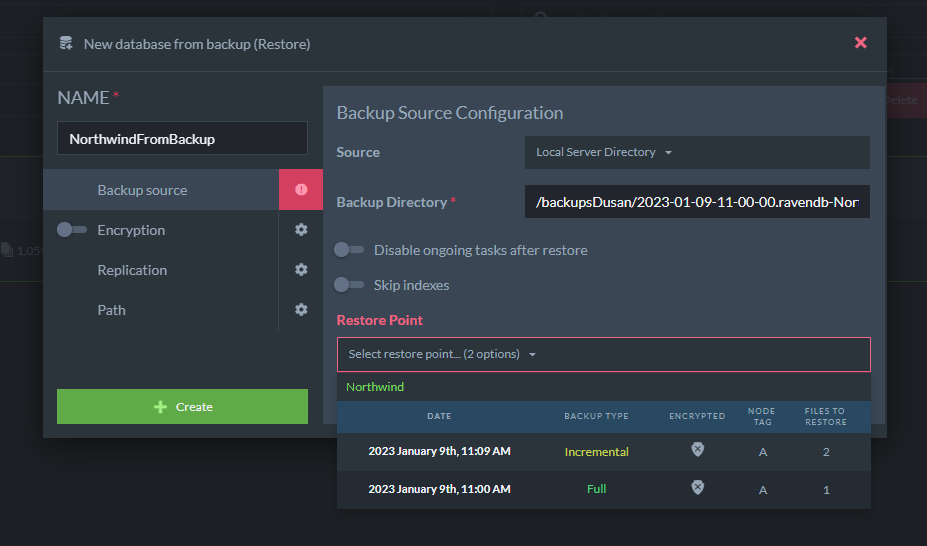
Moguće je kreirati i manualni backup koji će se odmah pokrenuti. Potrebno je kliknuti na dugme **Create a Backup** iz odeljka za Manual Backup. Nakon toga je potrebno definisati odredište backup-a, a na sledećoj slici je moguće videti rezultat manualnog backup-a Northwind RavenDB baze podataka:



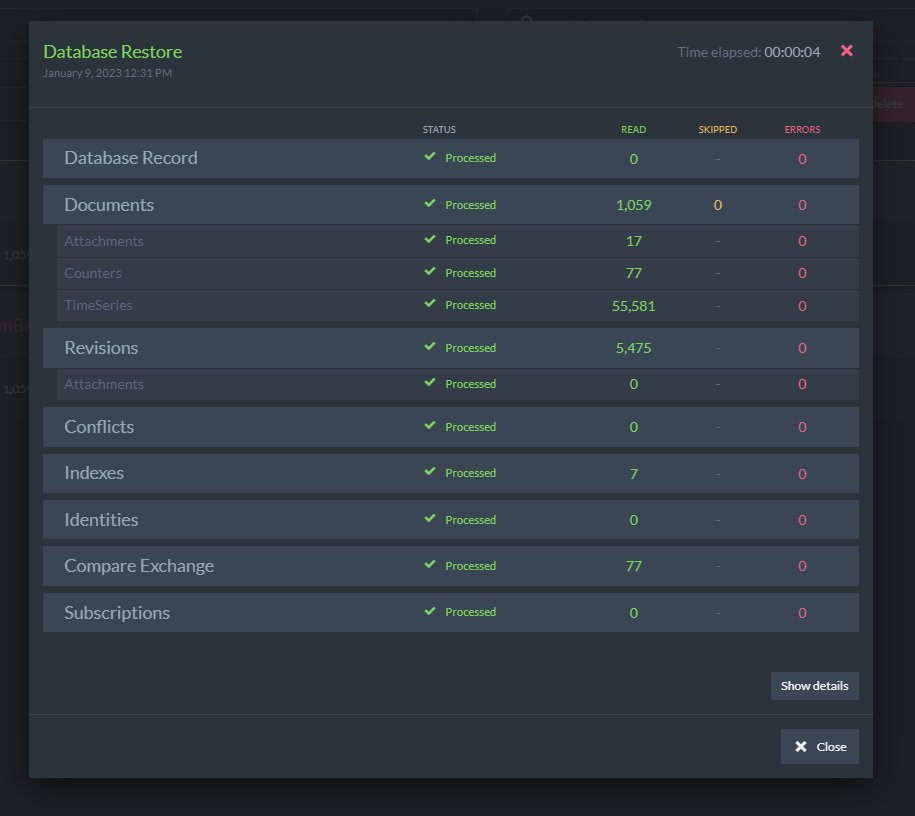
Nakon toga, backup RavenDB baze podataka je na lokaciji koja je izabrana. U našem slučaju je to /backupsDusan, a to je moguće videti i na fajl sistemu masine:



Do sada se pričalo o samom kreiranju backup-a, ali je definitivno mnogo bitnija stvar samo korišćenje backup-a, odnosno restore-ovanje baze podataka na osnovu backup-a. Kod RavenDB baze podataka je restore proces jako intuitivan i očigledan. Najpre je potrebno otići na stranicu **Databases** i pored dugmeta New database, potrebno je izabrati opciju **New database from backup (Restore)**. Nakon toga se otvara dijalog kao na sledećoj slici gde je potrebno podesiti par stvari:



Potrebno je dodeliti ime novoj RavenDB bazi podataka. Kako se kreira baza koja će biti identična kao Northwind baza podataka, nazvaćemo je NorthwindFromBackup. Potrebno je izabrati izvor backup-a. U ovom slučaju je to **Local Server Directory**. Nakon toga se bira backup direktorijum gde je nekada pre kreiran backup. Kao što se može videti na slici, postoje 2 restore point-a. Jedan je full backup, a drugi je incremental. U ovom konkretnom slučaju ćemo izabrati full backup. Nakon toga će započeti kreiranje baze podataka na osnovu ranije backup-ovane RavenDB baze podataka. Kada sve bude gotovo, bićemo obavešteni kao što se može videti na sledećoj slici:



Sam restore proces može trajati i jako dugo. Northwind baza podataka zauzima 296,62MB, a njen proces je trajao 4 sekunde. Za baze podataka koje su više stotina GB-a (prosek produkcione RavenDB baze podataka), restore proces može trajati i par sati.

Prilikom restore procesa definiše se lokalna putanja za specifičan čvor odakle se obavlja restore. Bitno je napomenuti da je moguće odraditi restore samo sa lokalnog čvora. Kao što je već pomenuto, RavenDB pruža mogućnost kreiranja backup-a na nekim remote lokacijama (Amazon S3, Azure Blob Storage,...). Međutim, nije moguće restore-ovati sa tih remote lokacija direktno. Najpre je potrebno download-ovati backup lokalno na čvor, pa odatle odraditi restore.

# Backup i dostupnost

Neki ops timovi su ubeđeni da je dobro posedovati samo jednu instancu nekog podatka u cluster-u i svaki problem rešavati restore-ovanjem iz backup-a. Iako na prvi pogled ovo deluje kao optimalno rešenje, nažalost nije. Ovaj proces, a i generalno sam restore proces šteti dostupnosti podataka. Oslanjanje na backup/restore ostavlja bez baze podataka neko vreme.

Sa druge strane, ukoliko postoji cluster sa replikacionim faktorom podataka, to ne znači da treba da se ignorišu backup-ovi. Replikacija podataka pruža sigurnost ukoliko neki čvor otkaže. U tom slučaju postoji kopija istog podatka na nekom drugom čvoru i sistem nastavlja nesmetano sa radom. Međutim, ukoliko se greškom obrišu podaci, obrisaće se svi podaci sa svih kopija i sistem će biti izgubljen. U tom slučaju pomaže backup/restore. U svakom trenutku je moguće rollback-ovati na prethodnu verziju baze podataka i tako ispraviti grešku.

RavenDB pruža jednu jako interesantnu opciju koja je zapravo spona između backup/restore-a i replikacije. Zove se **delayed external replication**. Ovde je moguće podesiti replikaciju sa određenim odlaganjem. Na primer, moguće je podesiti da se odloži replikacija za 6 sati. Ovako se dobija takozvani **hot node** na osnovu kog se mogu oporaviti podaci.

# Zaključak

Backup/restore predstavlja jedan jako bitan proces očuvanja podataka. To je mnogo više od običnog copy-paste-ovanja celog foldera baze podataka. Mnoge baze podataka, kao npr RavenDB, ni ne podržavaju ovaj način (prijaviće grešku). U jednom realnom live cluster-u proces backup/restore-a može postati jako komplikovan zbog dodele zaduženja oko backup-a. Takođe ovaj proces zahteva puno vremena, pa treba razmisliti kada će se obavljati. Nepisano je pravilo da se backup obavlja tokom noći, kada većinu sistema koristi manji broj korisnika. Takođe je praksa da se obavlja jedan kompletni backup, a nakon toga par inkrementalnih koji zahtevaju manje vremena.

Centralna tema ovog rada je backup/restore proces kod RavenDB baze podataka. RavenDB je jedna (relativno) nova baza podataka sa nekim jako naprednim konceptima. Ova NoSQL dokument baza podataka pruža transakcije sa ACID pravilima, a napredna je i u ostalim oblastima, pa je zato RavenDB baza podataka opisana na početku rada. Objašnjeni su njeni najbitniji koncepti, kako bi se stekla šira slika. Nakon toga je opisan sam proces backup/restore koji je poslužio kao uvod u backup/restore kod RavenDB baze podataka. Nakon toga ide i glavni deo rada koji predstavlja praktičnu demonstraciju. Najpre je demonstrirano pokretanje/instalacija RavenDB baze podataka. To je urađeno na 2 načina, preko Docker-a i lokalno. Nakon toga je demonstrirano kreiranje jedne baze podataka koja će poslužitit za praktičnu demonstraciju backup/restore koja je takođe detaljno opisana.

Backup/restore kod RavenDB baze podataka predstavlja jedan jako intuitivan i lak proces i na taj način se motivišu korisnici za konfigurisanje ovog procesa. Nakon konfiguracije, proces backup-a podataka je automatizovan, a restore se može uraditi u par koraka, tako da se preporučuje korišćenje backup/restore-a kod svake RavenDB baze podataka.

# Literatura

[0] <https://ravendb.net/learn/inside-ravendb-book/reader/4.0/1-welcome-to-ravendb>

[1] <https://www.acronis.com/en-us/blog/posts/data-backup/>

[2] <https://ravendb.net/learn/inside-ravendb-book/reader/4.0/17-backups-and-restores#backups-and-restores>

[3] <https://en.wikipedia.org/wiki/Gzip>

[4] <https://ravendb.net/learn/inside-ravendb-book/reader/4.0/16-monitoring-troubleshooting-and-disaster-recovery>

[5] <https://ravendb.net/docs/article-page/5.3/csharp/server/ongoing-tasks/etl/basics>

[6] <https://ravendb.net/learn/inside-ravendb-book/reader/4.0/13-securing-your-ravens>

[7] <https://ravendb.net/docs/article-page/5.4/csharp/indexes/querying/what-is-rql>