|  |  |  |
| --- | --- | --- |
|  | Univerzitet u Nišu  Elektronski fakultet |  |

Seminarski rad

**Cloud baze podataka i Database-as-a-service rešenja MongoDB Atlas**

**Mentor:** **Student:**

Doc. Dr Stanimirovic Aleksandar Kostić Jovana 1436

Sadržaj

[Uvod 3](#_Toc106915541)

[1. Pojam *DbaaS*-a 4](#_Toc106915542)

[1.1 Osnovne prednosti *DbaaS* pristupa 5](#_Toc106915543)

[2. MongoDB Atlas 6](#_Toc106915544)

[3. Kreiranje deploymena bazama podataka 8](#_Toc106915545)

[3.1 Vrste deployment-a 8](#_Toc106915546)

[3.3 Povezivanje sa *deployment*-om baze podataka 12](#_Toc106915547)

[4. Pristup i obrada podataka u *Atlas*-u 17](#_Toc106915548)

[4.1 Upravljanje bazama podataka 17](#_Toc106915549)

[4.2 Upravljanje kolekcijama 19](#_Toc106915550)

[4.3 Upravljanje dokumentima 19](#_Toc106915551)

[4.4 Upravljanje indeksima 22](#_Toc106915552)

[4.4 Kreiranje i upravljanje agregacionim *pipeline*-om 23](#_Toc106915553)

[5. Upravljanje klasterima 25](#_Toc106915554)

[5.1 Automatsko skaliranje 25](#_Toc106915555)

[5.2 *Pause, Resume* i *Terminate* klastera 26](#_Toc106915556)

[Zaključak 27](#_Toc106915557)

[Literatura 28](#_Toc106915558)

# Uvod

Kako kompleksnost aplikacija raste i kako se one usložnjavaju, tako i podaci kojima su praćene imaju složeniju strukturu i sve ih je više. Jedan od glavnih problema koji se javlja sa razvojem sve većih i kompleksnijih aplikacija jeste upravljanje podacima, nadgledanje, obrada a samim tim i njihovo skladištenje. Za ovu vrstu problema jedan od boljih ponuđenih rešenja jesu baze podataka u vidu servisa, odnosno *Database-as-a-Service*.

*Database-as-a-Service*pristup pre svega rasterećuje korisnika po pitanju hardvera jer podatke skladišti u *cloud*-u, a pritom ne zahteva nikakvu dodatnu složenu konfiguraciju pre samog korišćenja. Takođe, ovaj pristup obezbeđuje visoku dostupnost podataka i održava ih bezbednim. Na ovaj način troškovi korišćenja su znatno smanjeni ili ne postoje. Kako provajder upravlja bazom podataka, tako je menadžer baze podataka koja koristi ovaj pristup rasterećen svih administrativnih odgovornosti i zadataka.

*MongoDB* pruža veoma intuitivni grafički interfejs *Atlas* za korišćenje *Database-as-a-Service*, koji jednostavno mogu koristiti i ne tako stručni korisnici, bez preke potrebe za detaljnim predznanjem, što ga čini još rasprostranjenijim.

# Pojam *DbaaS*-a

*Database-as-a-service* je, kako sam naziv kaže, baza podataka u vidu *cloud* servisa. Kako je ovaj servis hostovan, korisnici ne moraju voditi računa o hardveru ili instaliranju softvera. Sve što se tiče upravljanja bazom podataka obavlja servis provajder. Hostovanje baze podataka dostupno je za sve tipove baza podataka uključujući *MySQL, NoSQL i PostgreSQL.* Upotreba *DbaaS*-a pruža sve što je neophodno za rad *cloud* baze podataka uključujući obezbeđivanje baze podataka, licence, podršku i održavanje. Kako se radi o *managed* servisu, ne postoje dodatni troškovi i odmah možemo pristupiti konkretnom radu sa našim podacima iz baze.

*Developer*-i mogu pristupati hostovanim *cloud API*-jima i upravljati podacima programski, odnosno iz koda. Iz tog razloga *DbaaS* pristup poseduje veliki broj sličnosti u poređenju sa *SaaS cloud* prilazom baziranom na subsktipciji*.* Kada se podaci učitaju, sam *DbaaS* mehanizam baze podataka radi na gotovo isti način kao lokalna baza podataka, zbog toga što je ista osnova odnosno jezgro ugrađeno u hostovanom *data* centru. *Developer*-i i inženjeri podataka imaju isto iskustvo, odnosno isti princip rada kao sa lokalnom bazom podataka.

Glavna razlika je fizička infrastruktura na kojoj *cloud* baza podataka radi. Kada su u pitanju *IaaS* *– Infrastructure as a Service*, kao što su *Microsoft Azure Amazon Web Services (AWS), Google Cloud Platform (GCP)* i *MongoDB Atlas*, mehanizam baze podataka se pokreću na zajedničkoj deljivoj harverskoj platformi. Ova činjenica pruža dodatnu računarsku snagu, elastičnost resursa i skalabilnost neophodne za skladišta podataka koje imaju tendenciju da se šire.

*DbaaS* je logično proširenje *cloud* tehnologija, koje koriste objedinjene kapacitete za skladištenje i obradu podataka, kako bi podržao promenljive zahteve korisnika platforme. Sa praktično beskonačnim obimom rasta podataka, *cloud* računarstvo je način da se prevaziđu ograničenja lokalnog centra podataka. Ovaj pristup je dobar za svaku aplikaciju kojoj je potrebna skalabilnost i fleksibilnost njihove baze podataka.

Upotreba *Dbaas*-a je mnogo rasprostranjenija nego što se na prvi pogled čini. Čak i najmanji sajtovi za koje se nikad ne bi ni pomislilo, takođe koriste koriste deljeni hosting. Vlasnik sajta plaća hosting, ali deo *DbaaS*-a je uključen u tu naknadu.

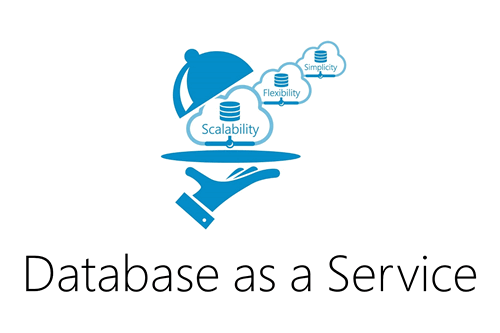
Izbor *DbaaS-*a zavisiće od potreba aplikacije. U većini slučajeva, vrši se migracija postojeće lokalne relacione baze podataka na ekvivalentnu *cloud* hostovanu bazu podataka. Alternativno, nestruktuirane *NoSQL* baze podataka nude maksimalnu fleksibilnost ako podaci zahtevaju promene u budućnosti. *MongoDB* je nerelaciona baza podataka koja omogućava čuvanje neobrađenih, neuređenih podataka. Svi tipovi baza podataka imaju svoje prednosti, ali izbor odgovarajuće baze podataka treba da bude vođen slučajevima koriščenja.

# 1.1 Osnovne prednosti *DbaaS* pristupa

Glavne prednosti *Database as a Service* pristupa su sledeće:

1. Nepostojanje dodatnog hardvera

Veličina baze podataka nastavlja eksponencijalno da raste, jer moderne aplikacije sada indeksiraju, pretražuju i obrađuju širok spektar tipova datoteka uključujući video, audio i druge nestruktuirane formate. Ova činjenica zahteva redovno ulaganje u dodatne kapacitete za skladištenje i obradu kako bi se omogućilo proširenje. Sa *DbaaS*-om, model *cloud* računarstva nudi praktično neograničen potencijal rasta bez ikakvih ulaganja unapred. Baza podataka može da raste bez brige o dostizanju kapaciteta ili potrebe za ulaganjem unapred u dodatni hardver.



Slika 1 – *Database as a Service*

*DbaaS* servisima u potpunosti upravljaju provajderi, brinu o infrastrukturi, hardveru, operativnom sistemu i softveru, što omogućava fokus samo na izgradnji konkretne aplikacije ali ne i na tehnikalijama vezanim za ove servise.

1. Brza implementacija baze podatata

Korišćenjem baze podataka kao servisni model, proširenje i poboljšanje operacija nad podacima je značajno pojednostavljeno. *Developer*-i mogu brzo da obezbede baze podataka po potrebi, jednostavno klonirajući skupove podataka i konfiguracije.

1. Operacije nad podacima koje su spremne za buduću upotrebu

Odabirom *NoSQL* *DbaaS* baze podataka kao što je *MongoDB Atlas* poširuju se mogućnosti operacija koje se mogu obaviti nad podacima. Nestruktuirane upravljane baze podataka mogu se iskoristiti za pribavljanje podataka korišćenjem servisa, kreiranjem *ODL*- *Operational data layer*-a na samom vrhu *NoSQL* baze podataka. Operativni sloj podataka čini korporativne podatke uvek dostupnih na zahtev, spremnim za razvoj novih aplikacija koje će omogućiti više operacija nad podacima. Ovaj operativni sloj podataka je važan korak u razvoju inteligentnih, brzih i *real-time* aplikacija.

# MongoDB Atlas

*MongoDB Atlas* je *multi-cloud* platforma koja u osnovi predstavlja *fully managed* *cloud* bazu podataka modernih aplikacija. *Atlas* je najbolji način za pokretanje *MongoDB*-a, vodeće nerelacione baze podataka. *Atlas* baza podataka je dostupna u preko 80 regiona širom *AWS*-a, *Google Cloud*-a i *Azure*-a. Jedna od glavnih prednosti je *multi-cloud*  i *multi-region* pristup razvoja. Ovaj pristup pruža dostupnost, skalabilnost i usklađenost sa najzahtevnijim standardima bezbednosti i privatnosti podataka.

*MongoDB Atlas* pruža jednostavan način za hostovanje i upravljanje *cloud* podacima. Postoje dva pristupa, korišćenjem *Atlas CLI*-ja ili korsničkog interfejsa koji pruža *Atlas.* Koji god pristup da je u pitanju postoje osnovni koraci da bismo započeli sa razvojem:

1. Kreiranje *Atlas* naloga
2. Kreiranje novog klastera
3. Dodavanje korisnikove *IP* adrese u *IP* *access* listu
4. Kreiranje *database* naloga korisnika
5. Povezivanje na klaster
6. *Insert*-ovanje dokumenta
7. Učitavanje podataka

Da bi se kreirao klaster, pre toga neophodno je kreirati *Atlas* nalog. Nakon toga, sam klaster može se kreirati na 3 načina i to kao:

* *Starter cluster*
* *Single-region cluster*
* *Multi-region cluster*

*Starter cluster* se najčešće koristi za učenje i istraživanje *MongoDB*-a u *cloud* okruženju. Besplatan je i pruža osnovne konfiguracione opcije i koristi se za za razvoj manjih aplikacija. Dolazi sa već kreiranim jednostavnim test podacima. Ovaj tip klastera poznat je još kao i *shard.*

U slučaju *single-region* klastera već imamo plaćanja, ali plaćaju se samo one operacije koje se pokreću. Resursi se mogu sklairati prema potrebama aplikacije. Zahteva minimalnu konfiguraciju i koristi se za razvoj aplikacija, testiranje i za aplikacije sa pormenljivom količinom saobraćaja podataka. Pripadaju grupi *serverless* klastera.

*Multi-region* klasteri, poznati još i kao *dedicated*, koriste se za *production* aplikacije sa sofisticiranim zahtevima u kontekstu razvoja. Pruža mogućnost napredne konfiguracije i takođe se plaća. Omogućavaju izolaciju mreže i *multi-cloud* opcije.

Nakon kreiranja klastera, treba odabrati *cloud* provajdera i region. *Atlas* podržava besplatne *M0* klastene na *Amazon Web Service*-ima, *Google Cloud Platform*-i i *Microsoft Azure*. *Atlas* će ponuditi samo regione koji podžavaju *M0* slobodne klastere i *M2/M5* deljene klastere. Nakon kreiranja klastera, potrebno je neko izvesno vreme da postane spreman za upotrebu.

Sledeći korak u Atlasu je povezivanje na klaster korišćenjem pouzdane IP adrese. Atlas pruža mogućnost kreiranja liste pouzdanih IP adresa, koja se naziva pristupnom listom i koristi se za povezivanje sa klasterom i za pristup podacima.

Za pristup konkretnom klasteru, neophodno je kreiranje i korisnika baze podataka. Iz bezbednosnih razloga, *Atlas* zahteva od klijenata da se autentifikuju kao korisnici *MongoDB* baze podataka da bi pristupili klasterima. Korisnici baze podataka odvojeni su od korisnika Atlasa, jer korisnici Atlasa mogu da se prijave na Atlas ali da nemaju pristup *MongoDB* bazama podataka.

Da bi se konkretna aplikacija povezala sa klasterom i komunicirala sa bazom koriste se *PyMongo Driver, Node.js Driver, MongoDB Shell* ili *Compass*. Koji god pristup od navedenih da se koristi, zajednički korak je povezivanje uz pomoć konekcionog stringa.

Nakon kreiranja klastera imamo test podatke koji dolaze uz njega, a uz pomoć grafičkog okruženja odnosno *Data expoler*-a možemo te podatke importovati u klaster. To možemo učiniti uz pomoć opcije *Load a Sample Dataset* iz *Database Deployments view-*a klikom na *Browse Collection* dugme. Importovanje traje okvirno oko 5 minuta i kada se operacija završi, *Atlas* korisnički interfejs se osvežava i možemo videti 5 baza podataka u *database deployments*: sample\_airbnb, sample\_geospatial, sample\_mflix, sample\_training, sample\_weatherdata.

# Kreiranje deploymena bazama podataka

# 3.1 Vrste deployment-a

*Deployment* baze podataka u *Atlasu* se kreira radi lakšeg skladištenja i upravljanja podacima. *MongoDB Atlas* pruža dva tipa *deployment*-a: *serverless* instance i klasteri, a omogućava i da oba ova pristupa budu iskorišćena u okviru istog *Atlas* projekta.

Klaster se kreira u slučajevima kada je potrebno da:

* Odaberemo specifičnu konfiguraciju baze podataka na osnovu konteksta koji zahteva konkretna aplikacija
* Definišemo ponašanje baze podataka u slučaju skaliranja
* Pokrenemo zadatke visokih performansi

Ovaj pristup omogućava postavljanje nivoa, odnosno sloja klastera. Takođe, postoje napredne mogućnosti kao što je proces particionisanja i distribuiranje podataka na više regiona i *cloud* provajdera. Još jedna od prednosti ovog pristupa je skaliranje na zahtev kao i mogućnost automatskog skaliranja. Za razliku od *serverless* instanci, automastko skaliranje klastera zahteva prethodno konfigurisanje. *MongoDB* naplaćuje klastere na osnovu konfiguracije *deployment*-a i nivoa klastera.

*Serverless* instance su korisne kada je neophodno da:

* Proces razvoja započnemo brzo i sa minimalnom prethodnom konfiguracijom baze podataka
* Se baza podataka automatski i dinamički skalira kako bi rasporedila radno opterećenje
* Se razvija i testira u *cloud* okruženju

*Atlas* automatski skalira kapacitet skladištenja, propusnost skladišta i računarsku snagu u slučaju *serverless* instanci kako bi zadovoljio zahteve radnog opterećenja. *Serverless* instance uvek pokreću najnoviju verziju *MongoDB-*a i naplaćuju se samo operacije koje se pokreću.

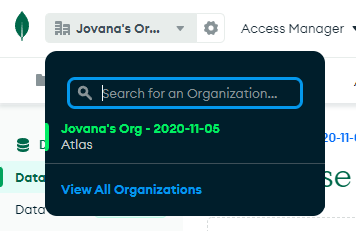
Globalni klasteri se koriste kada je neophodno podržati operacije čitanja i upisa sa svešću o lokaciji. Operacije čitanja i pisanja koje su svesne lokacije su idealne za globalno distribuirane instance aplikacije i klijente. Ovaj pristup nudi:

* Operacije čitanja i upisa sa malim kašnjenjem za globalno distribuirane klijente
* Izolaciju radnog opterećenja zasnovanom na tipovima članova klastera
* Skladištenje podataka na osnovu lokacije u određenim georafskim regionima

**3.2 Proces kreiranja klastera**

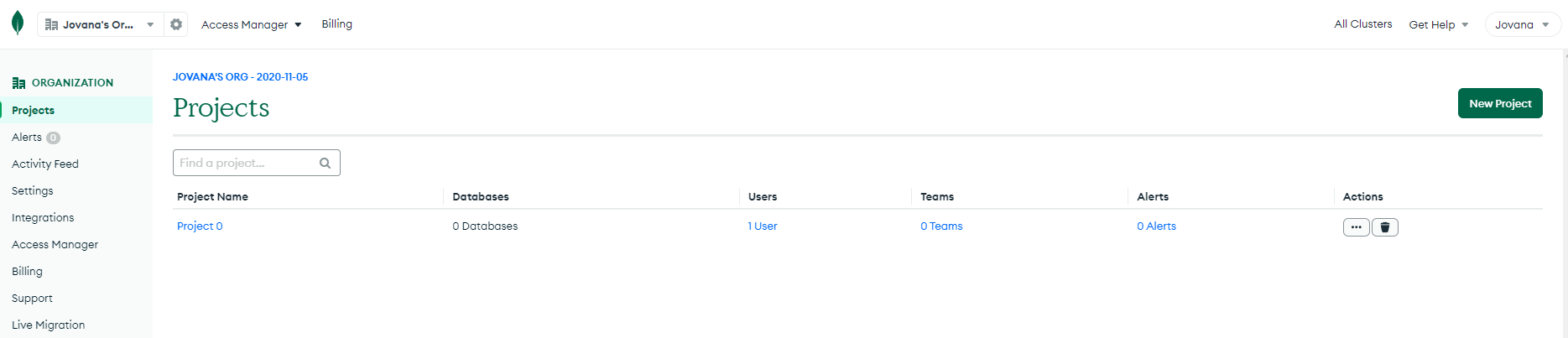
Klasteri mogu biti skupovi replika ili *sharded* klasteri. U nastavku biće obrađen pristup sa skupom replika.

U okviru *Atlas* okruženja, kao prvi korak je odabir orgaizacije u okviru koje se nalazi projekat, a i samog projekta, da bismo došli do *Database Deployments* sekcije što se može videti na narednoj slici.



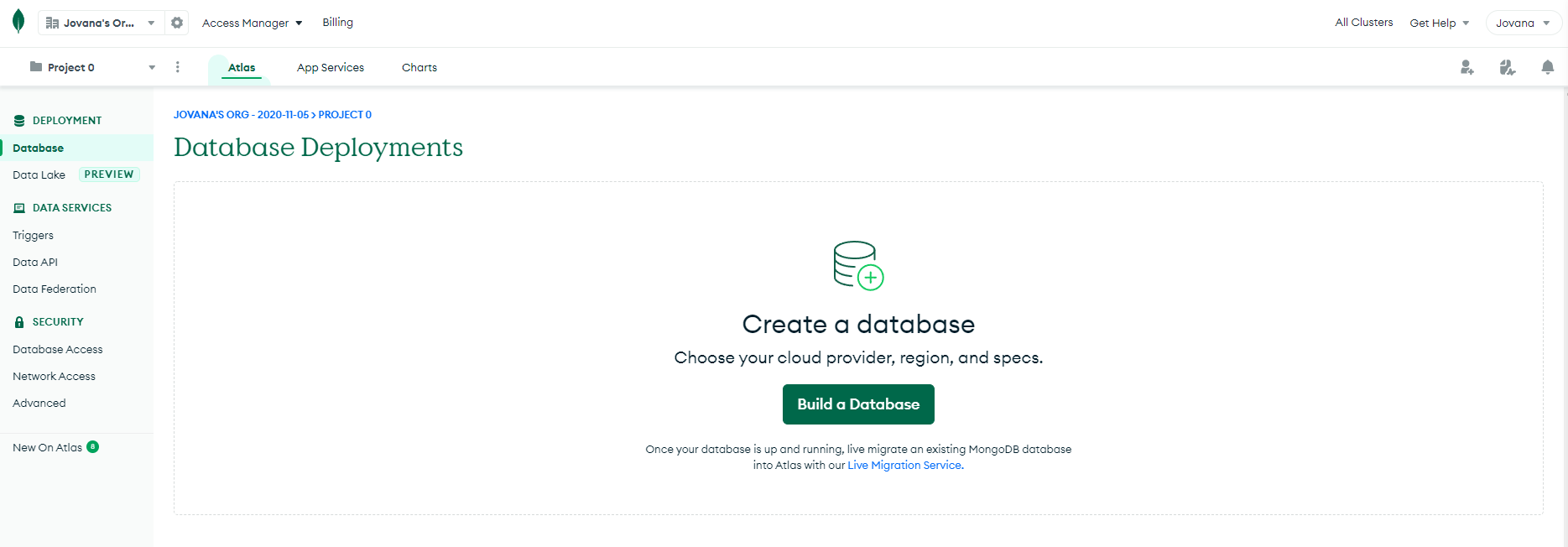
Slika 2 – Organizacije u Atlas okruženju

Odabirom organizacije, dobijamo spisak projekata u okviru nje, čijim daljim odabirom dolazimo do *Database Deployments* dela.



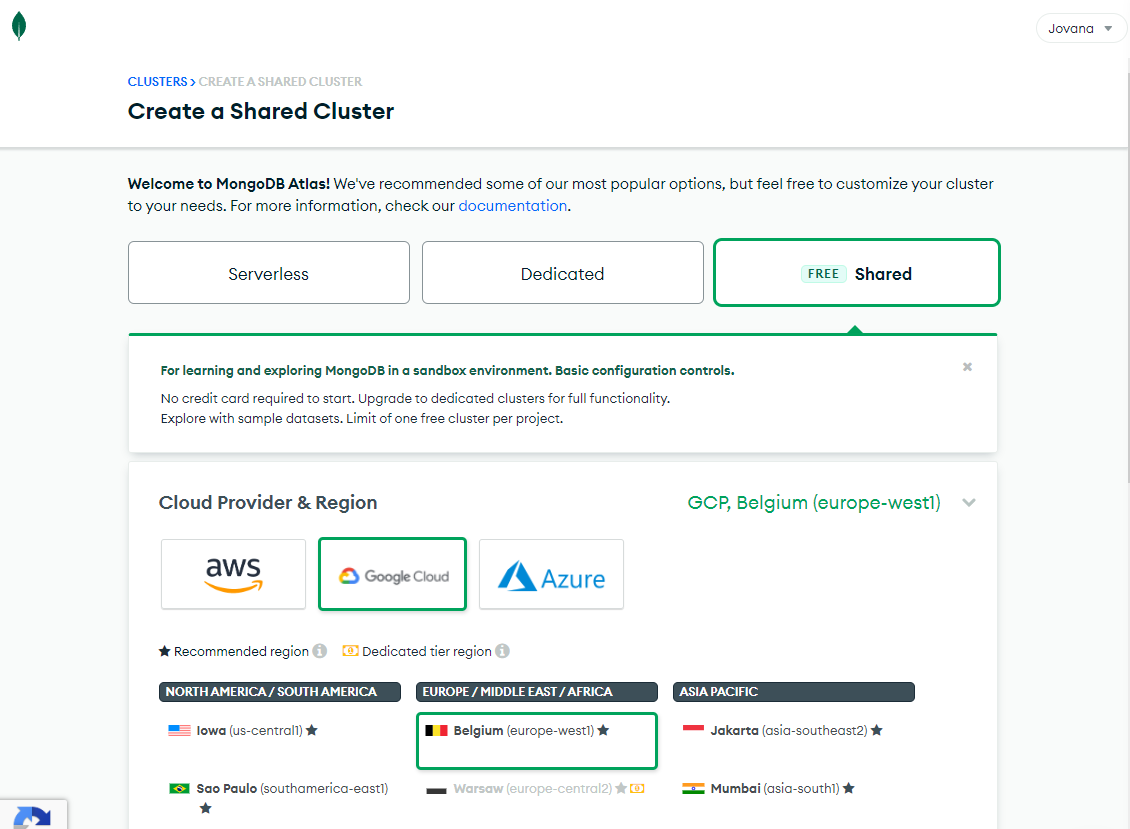
Slika 3 – Projekti u okviru organizacije

Kreiranje same baze podataka možemo postići opcijom *Build a Database*, a nakon toga, na osnovu prethodno objašnjenih karakterstika i samih potreba aplikacija vrši se odabir tipa *deployment*-a, odnosno klastera. U našem slučaju kreiraćemo *sharded* klaster, koji je zapravo besplatna opcija a pruža sve osnovne funkcionalnosti radi demonstracije *Atlas* okruženja.



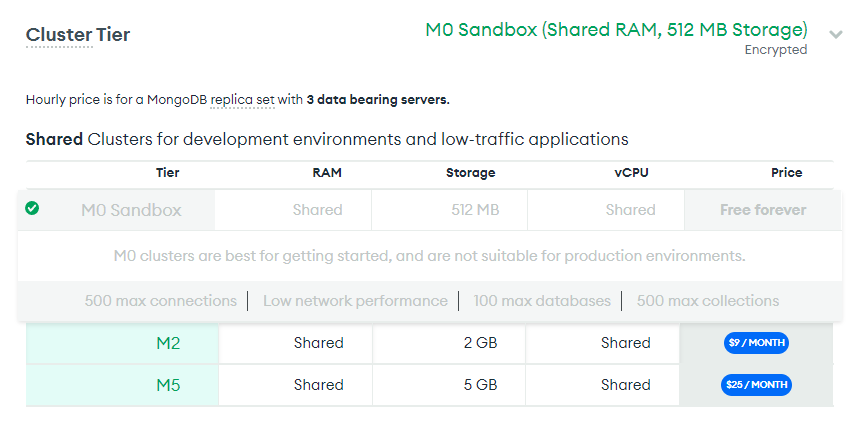
Slika 4 – *Database deployments*

Ovaj klaster zapravo predstavlja samu *cloud* bazu podataka koju koristimo za razvoj. Pri kreiranju klastera treba odabrati i *cloud* provajdera kao i region, što dalje utiče na konfiguracione opcije za dostupne nivoe klastera, kašnjenje kroz mrežu za klijente koji pristupaju klasteru, geografsku lokaciju čvorova u klasteru i cenu pokretanja klastera.



Slika 5– Tipovi *cloud* baza podataka

U ovom delu takođe može se odabrati opcija za razvoj klastera u više regiona. *Multi-region* klasteri bolje podnose prekide u *data* centrima i mogu da sadrže namenske geografske regione za lokalizovano čitanje, čime se poboljšavaju performanse.



Slika 6 – Odabir nivoa klastera

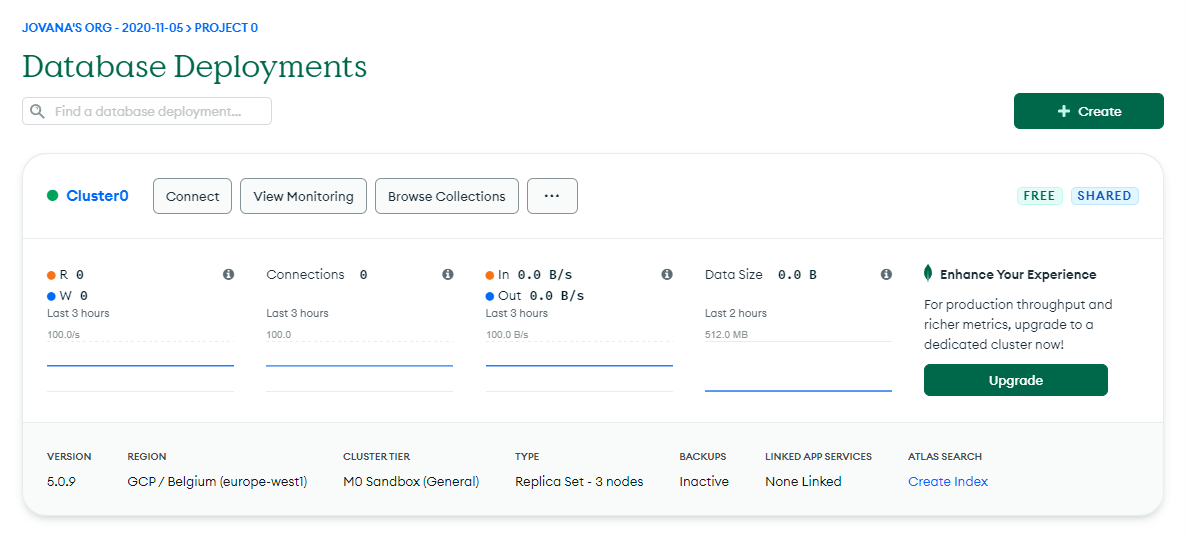
Sledeći korak je odabir nivoa koji diktira količinu memorije, način skladištenja i *IOPS* specifikaciju za svaki server u klasteru. Nivoi klastera M10 i viši imaju podršku za automatsko skaliranje klastera. Automatsko skaliranje je podrazumevano omogućeno kada se kreiraju novi klasteri pomoću korisničkog interfejsa, a u slučaju korišćenja *API-*ja, podrazumevano je onemogućeno. Sa omogućenim automatskim skaliranjem, *Atlas* automatski skalira nivo klastera i kapacitet skladišta. Automatsko skaliranje omogućava klasteru da se prilagodi trenutnom radnom opterećenju i smanji potrebu za ručnim optimizacijama.

Skaliranje skladišta klastera automatski povećeva kapacitet klastra kada se koristi 90% kapaciteta diska. Ovo podešavanje je podrazumevano omogućeno da bi se osiguralo da klaster uvek može da podrži iznenadne prilive podataka.

Za kontrolu automatskog skaliranja klastera koje *Atlas* nudi, neophodno je podesiti:

* Maksimalni nivo klastera na koji se može automatski povećati – Podrazumevano je podešeno na sledeći nivou klastera u poreženju sa trenutnim
* Minimalni novo klastera na koji se može automatski smanjiti – Podrazumevano je podešeno na trenutni nivo klastera

Pored do sada navedenih podešavanja, postoje i dodatna podešavanja koja su dostupna samo za M2 nivo klastera. Nakon kreiranja klastera grafičko okruženje *Atlasa* izgleda ovako:



Slika 7 – Kreirani klaster

*Atlas* nam pruža uvid u frekvenciju upisa i čitanja na primarnom skupu replika, broj aktivnih konekcija na bazu podataka, brzinu prenosa podataka od i ka bazi i veličinu logičkog diska.

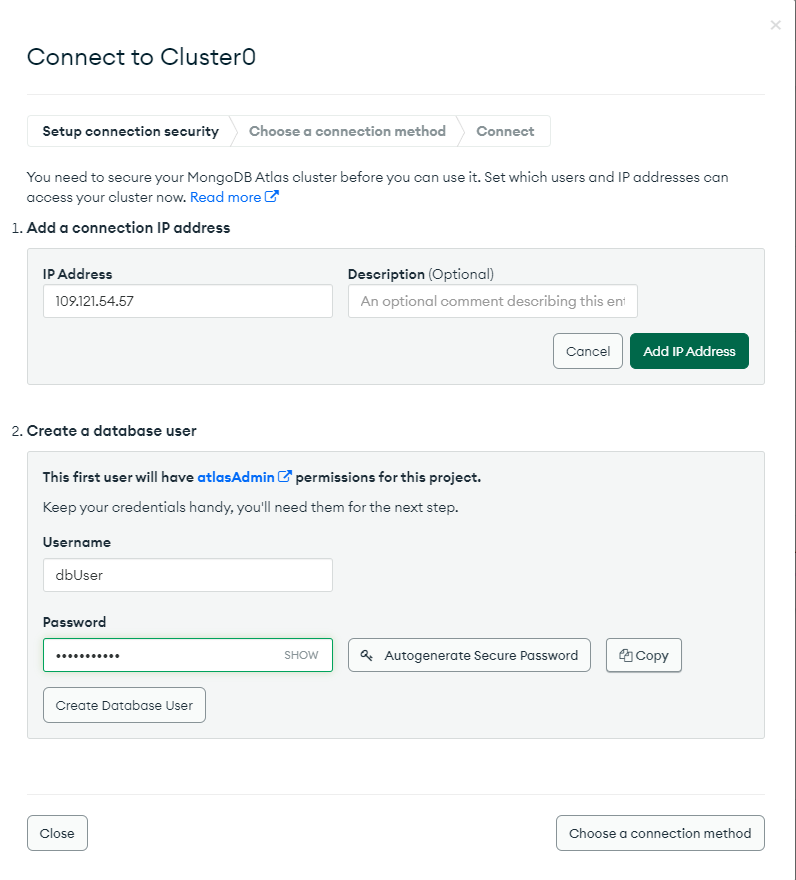
# 3.3 Povezivanje sa *deployment*-om baze podataka

Pre samog povezivanja aplikacije sa *deployment*-om baze podataka, odnosno sa klasterom, neophodno je kreirati *IP* pristupnu listu. Da bi pristup bazi podataka bio omogućen treba se povezati sa *IP* adrese koja je prethodno dodata u *IP* *access* listu *Atlas* projekta. To možemo učiniti odabirom opcije *Connect* koja će nam prikazati dialog sa slike 8.

U prvom delu možemo dodati adrese koje imaju prava pristupa novokreiranom klasteru. U drugom delu ovog dialoga neophodno je kreirati korisnika baze podataka kao preduslov za povezivanje aplikacije i same baze podataka. Korisnici baze podataka su odvojeni od korisnika *Atlasa*, oni imaju pristup *MongoDB* bazama podataka, dok *Atlas* korisnici imaju pristup samo *Atlas* aplikaciji.

Pored same adrese, postoji opcija dodavanja opisa uz svaku od adresa u ulozi neke vrste napomena. Takođe, kada kreiramo korisnika baze podataka ne moramo sami da kreiramo lozinku, već *Atlas* može automatski generisati sigurnosnu lozinku umesto nas.

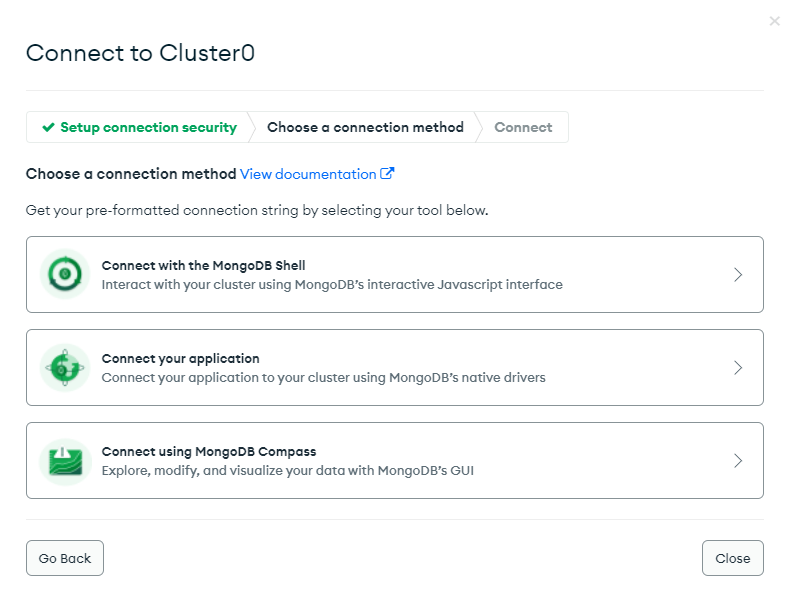
Kreiranje *IP access* liste i samog korisnika baze podataka su dva osnovna preduslova za pristup samom klasterom i daljom manipulacijom podataka, kao i za povezivanje sa konkretnom aplikacijom koju razvijamo.



Slika 8 – *IP* adresa za povezivanje i kreiranje korisnika baze podataka

Nakon ispunjenih preduslova, neophodno je odabrati sam način povezivanja. Najčešće su to:

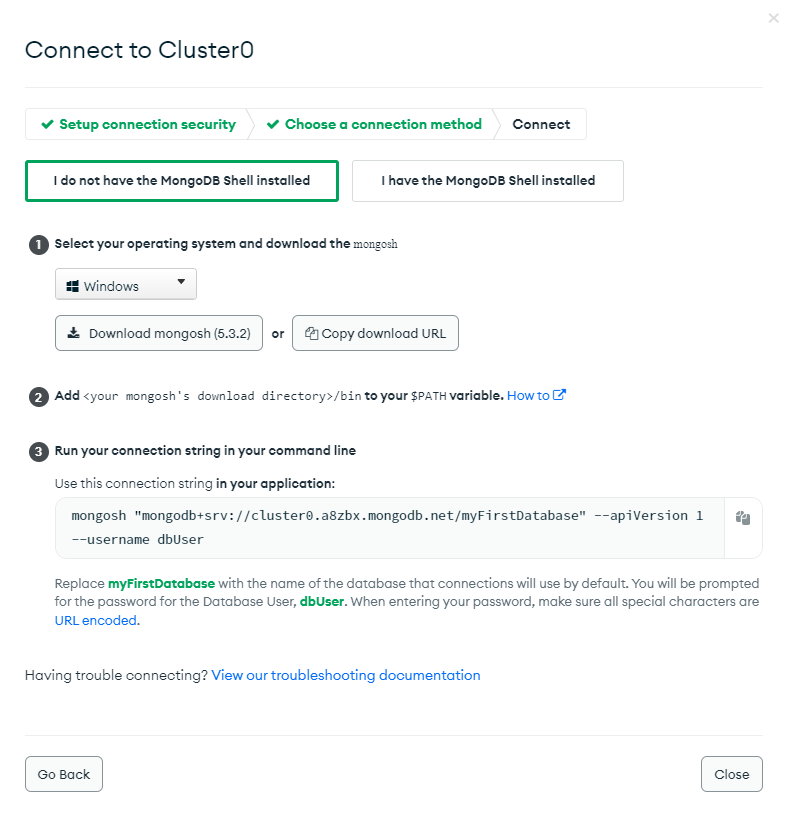
* *MongoDB Shell*
* *MongoDB native drivers*
* *MongoDB Compass*



Slika 9 – Metode za povezivanje *Atlas*-a sa konkretnom aplikacijom

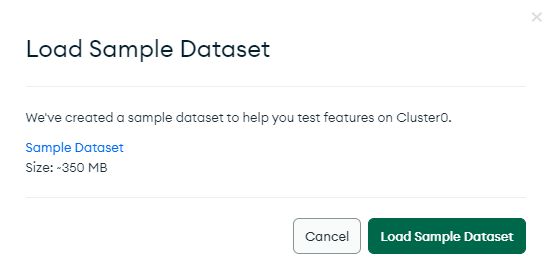
U slučaju da odaberemo *MongoDB Shell, Atlas* odmah nudi mogućnost preuzimanja instalacije za konkretan operativni sistem, kao i uputstvo i potrebnu dokumentaciju za samu instalaciju. Takođe, ilustruje i sam način povezivanja kroz primer uz minimalnu konfiguraciju, koristeći *connection string* i konkretnog korisnika baze podataka.

*MongoDB Shell*, odnosno *mongosh*, je potpuno funkcionalno *JavaScript* i *Node.js* okruženje za interakciju sa *MongoDB deployment*-ima. *MongoDB Shell* se može takođe koristiti za testiranje upita i operacija direktno nad bazom podataka.



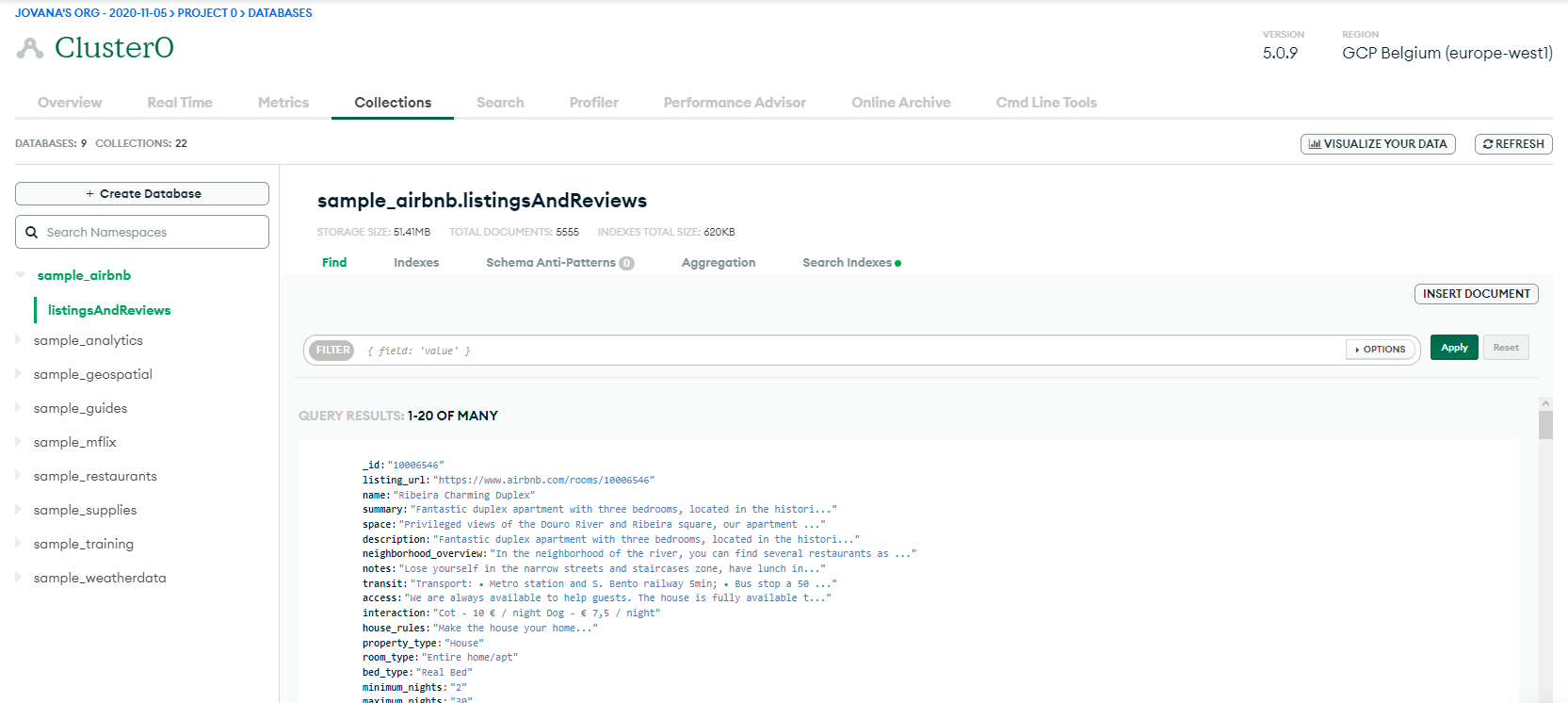
Slika 10 – Instalacija *MongoDB Shell*-a i povezivanje sa bazom podataka

Testiranje i ilustracija osnovnih funkcionalnosti koje *Atlas* pruža moguće je nakon učitavanja podataka. Kao što možemo videti na slici 7, pored informacija o samom klasteru, mogućnosti povezivanja sa konkretnom aplikacijom i *monitoring*-a imamo i opciju *Browse Collections* koja nam omogućava da učitamo naše korisniške podatke ili test podatke koje nudi sam *Atlas*, tzv. *Sample Dataset*. Za ilustrativne potrebe učitaćemo gotove test podatke.



Slika 11 – Učitavanje test podataka

Ovaj proces može potrajati izvesno vreme, a nakon završetka učitavanja možemo testirati osnovne funkcionalnosti. Klaster sa učitanim *dataset*-ovima u *Atlas* grafičkom okruženju izgleda ovako:



Slika 12 – Klaster sa učitanim test podacima

Sa leve strane možemo videti listu baza podataka, dok u središnjem delu imamo konkretnu kolekciju pod nazivom *listingsAndReviews* u okviru baze podataka *sample\_airbnb*. Pored toga, imamo i veličinu baze podaka, ukupan broj dokumenata u okviru date kolekcije i ukupnu veličinu indeksa.

# Pristup i obrada podataka u *Atlas*-u

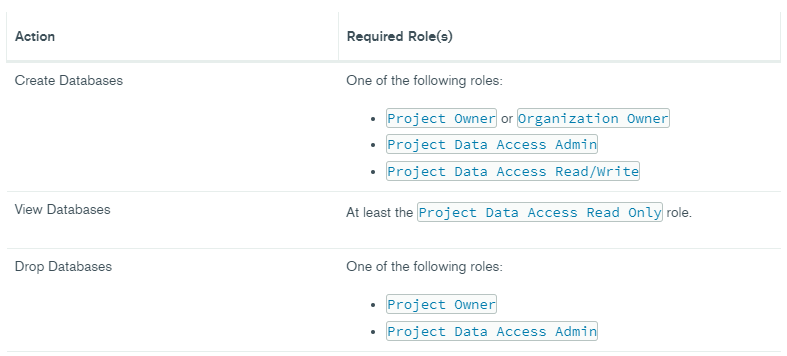
*Atlas* korisnički interfejs nam pruža mogućnost interakcije sa podacima na više načina, kao na primer upravljanjem:

* Bazama podataka u *deployment*-ima baze podataka
* Kolekcijama u *deployment*-ima baze podataka
* Dokumentima u kolekcijama
* Indeksima nad kolekcijama
* Kreiranjem i pokretanjem agregacionog *pipeline*-a
* Globalnim *shard* klasterima za distribuciju ogromnih *dataset*-ova

Pre svega, korisnički interfejs *Atlas* podrazumevano čita sa primara, osim ako primar nije dostupan, u tom slučaju čita sa nesakrivenog i neodloženog sekundranog člana. Za konkretnu interakciju sa podacima u bilo kom smislu neophodno je da *Data Explorer* bude odobren, što podrazumevano i jeste.

# 4.1 Upravljanje bazama podataka

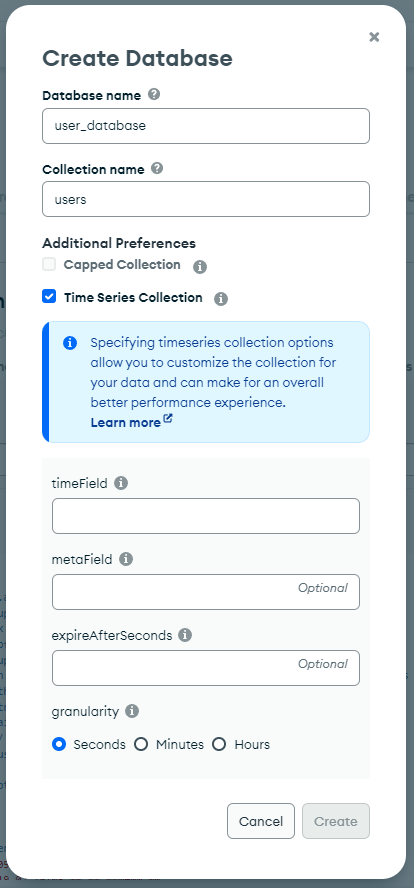
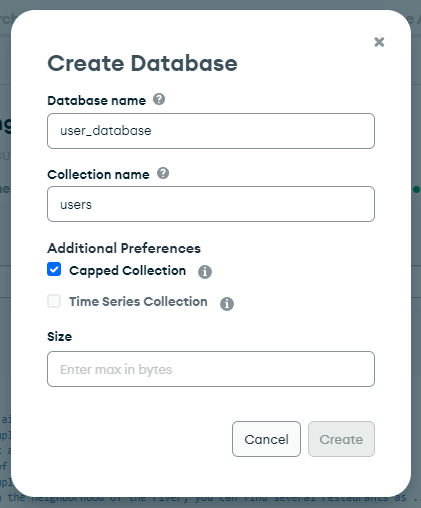
Upravljanje bazama podataka praćeno je određenim ulogama, drugim rečima, postoje određena prava pristupa koje su nam neophodno za određene manipulacije sa bazama podataka, što možemo videti u sledećoj tabeli:



Slika 12 – Prava pristupa pri manipulaciji bazama podataka

Pored prethodno učitanih test datasetova, možemo kreirati i svoju bazu podataka ako imamo zahtevana prava pristupa. To je moguće učiniti opcijom *Create database*, pri čemu moramo definisati ime baze podataka koje je u skladu sa ograničenjima koje postavlja *MongoDB*, kao i naziv kolekcije. Opciono kolekcije možemo definisati kao *Capped Collection* ili *Time Series Collection*.

*Capped Collection* predstavlja kolekciju fiksne veličine koja podržava visoku propusnost operacija koje *insert*-uju i vraćaju dokumente na osnovu redosleda *insert*-ovanja. Ako kreiramo kolekciju ovog tipa neophodno je da definišemo maksimalnu veličinu u bajtovima.



Slika 13 – Kreiranje baze podataka

*Time Series Collection* efikasno skladište delove podataka periodično, pri čemu je neophodno definisati *timeField* koji će sadržati svaki dokument u kolekciji i mora biti tipa *BSON UTC datetime*. Preostala dva polja, *metaField* i *expireAfterSeconds*, su opciona. *metaField* služi za grupisanje povezanih podataka, a *expireAfterSeconds* definiše nakon koliko sekundi će zastarele serije podataka biti obrisane.

Nakon kreiranja, baza podataka biće dodata u listu prethodno učitanih baza odakle joj se može pristupiti, a iz te liste je možemo i obrisati opcijom *Drop*.

# 4.2 Upravljanje kolekcijama

Kada govorimo o upravljanju kolekcijama, pored inicijalnog dodavanja kolekcije pri kreiranju baze podataka, možemo i samostalno dodati novu kolekciju u konkretnu bazu podataka. To se može učiniti odabirom konkretne baze podataka i opcijom *Create Collection* koja će nam otvoriti dialog poput modala na slici 13 pri čemu neće biti omogućena izmena naziva baze podataka, ali opet možemo odabrati tip kolekcije. Što se tiče brisanja kolekcije, postupak je isti kao i kod brisanja baze podataka, samo što sada radimo na nivou kolekcije.

Ukoliko se manipuliše ogromnim *dataset-*ovima i obavljaju se operacije velike propusnosti, može se izvršiti particionisanje, odnosno *shard*, kolekcije da bi se podaci distribuirali kroz *shard*-ove. Preduslovi za ovaj proces su posedovanje *sharded* *Atlas* klastera i *mongosh* na lokalnoj mašini. Prvenstveno je neophodno povezati se sa bazom podataka pomoću *mongosh*, zatim odobriti *sharding* sledećom komandom:



Slika 14 – Komanda za omogućavanje *sharding*-a

Sam proces *shard*-ovanja kolekcije postiže se naredbom:



Slika 15 – Komanda *shard*  kolekcije

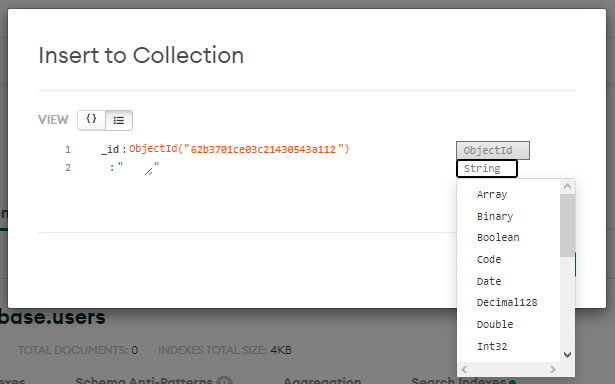
Prava pristupa u slučaju upravljanja kolekcijama su ista kao i kod upravljanja bazama podataka, samo imamo i prava pristupa za slučaj *shard*-inga kolekcije, a to su *Project Owner*  i *Organization Owner*.

# 4.3 Upravljanje dokumentima

*Atlas* se koristi i za upravljanje dokumentima unutar kolekcija. Dokumenti su pojedinačni *records*-i, odnosno zapisi/upisi, i osnovna su jedinica podataka u *MongoDB*-u.

Za upis, izmenu ili brisanje dokumenta, pristup dokumentima mora biti odobren kao: *Project Owner* ili *Organization Owner*, *Project Data Access Admin* i *Project Data Access Read/Write.*

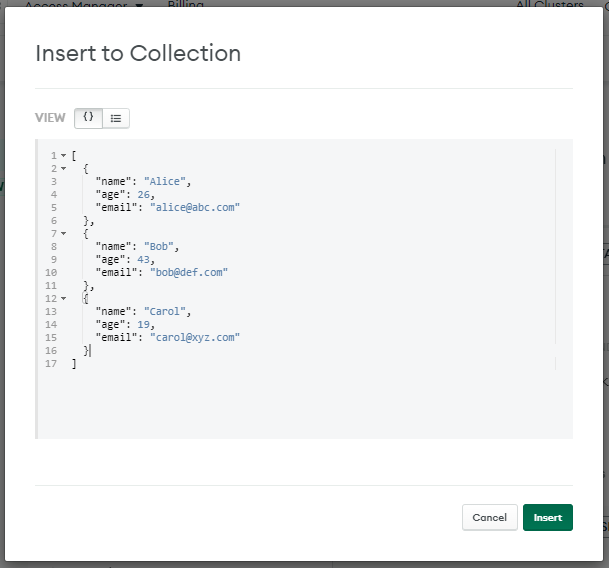
Proces dodavanja dokumenta u okviru konkretne kolekcije jedne baze podataka moguć je u okviru sekcije *Find* kada otvorimo određenu kolekciju odabirom opcije *Insert Document*, koja nam dalje otvara sledeći dijalog:



Slika 16 – Dodavanje novog dokumenta pomoću *Document editor*-a

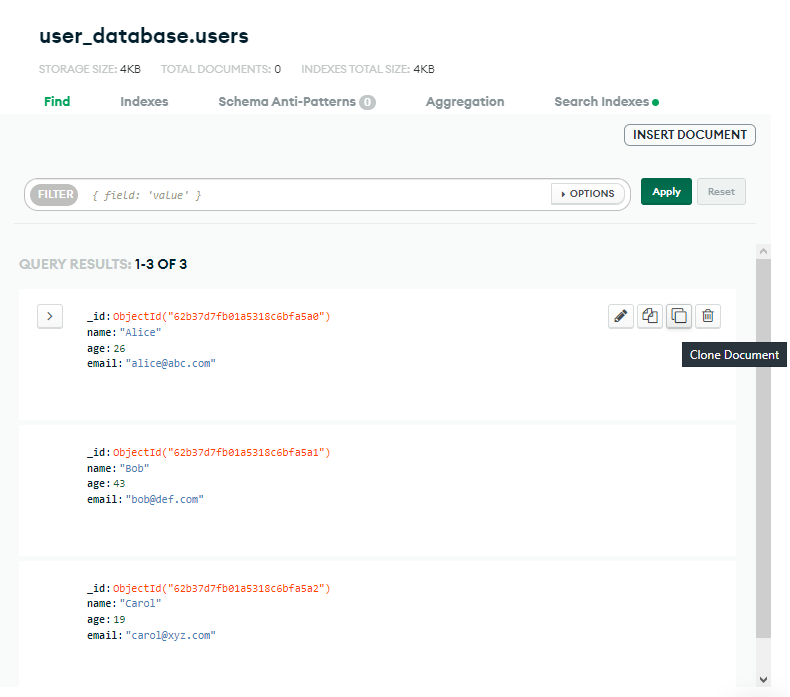
*Document editor* koji *Atlas* pruža je veoma intuitivan i možemo videti da u okviru dokumenta dodajemo *key-value* parove određenog tipa. Podrazumevano već postoji *field* \_id koji skladišti jedinstvani identifikator tipa *ObjectId* koji sam *MongoDB* generiše. Dodavanje *field*-ova vrši se opcijom *+* koja se pojavljuje *onHover*, a modifikaciju klikom na konkretnu vrednost koju želimo da izmenimo.

Ovaj editor pruža i pregled podataka u *JSON* formatu. Ta mogućnost dozvoljava dodavanje više dokumenata istovremeno tako što se doda lista dokumenata, odnosno *JSON* objekata kao u primeru na sledećoj slici:



Slika 17 – Dodavanje niza dokumenata istovremeno

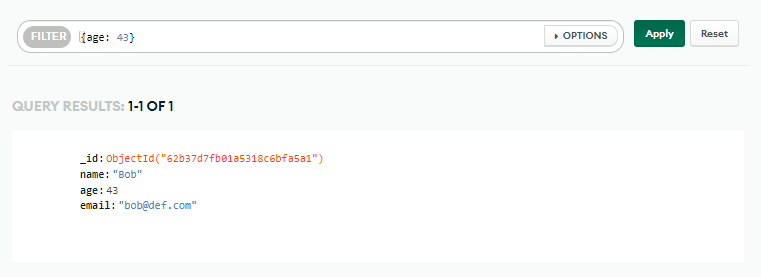
Sada u bazi podataka *user\_database* imamo kolekciju *users* unutar koje su dodata 3 dokumenta istovremeno i to je u grafičkom okruženju *Atlas* predstavljeno ovako:



Slika 18 – Grafičko okruženje *Atlas* nakon dodavanja dokumenata

Pored klasičnog kreiranja dokumenta, postoji i mogućnost klonovanja postojećeg dokumenta, što se može videti na slici 18. Opcije za modifikaciju podataka, kao i brisanje ali i kopiranje su takođe očigledne i intuitivne.

Na prethodnoj slici može se uočiti i deo koji se odnosi na filter. Filter zapravo predstavlja mogućnost za pretraživanje podataka kreiranjem upita u formatu *{ field: ‘value’ },* pri čemu *field* predstavlja naziv propertija na osnovu kojeg želimo da vršimo pretraživanje, a *value* konkretnu vrednost tog atributa. U konkretnom primeru, ako bismo pretraživali korisnike na osnovu starosti, odnosno godina i zahtevali sve korisnike koji imaju 43 godine, naš upit izgledao bi ovako: {age: 43}, a kao rezultat ovog pretraživanja dobili bismo dokument koji predstavlja objekat korisnika *Bob.*

Slika 19 – Pretraživanje dokumenata korišćenjem *query filter*-a

# 4.4 Upravljanje indeksima

*Atlas* korisnički interfejs može se korisiti i za upravljanje indeksima u kolekcijama. Indeksi omogućavaju efikasno izvršavanje upita u *MongoDB*-u i treba ih koristiti za polja, odnosno atribute, koje konkretna aplikacija često čita.

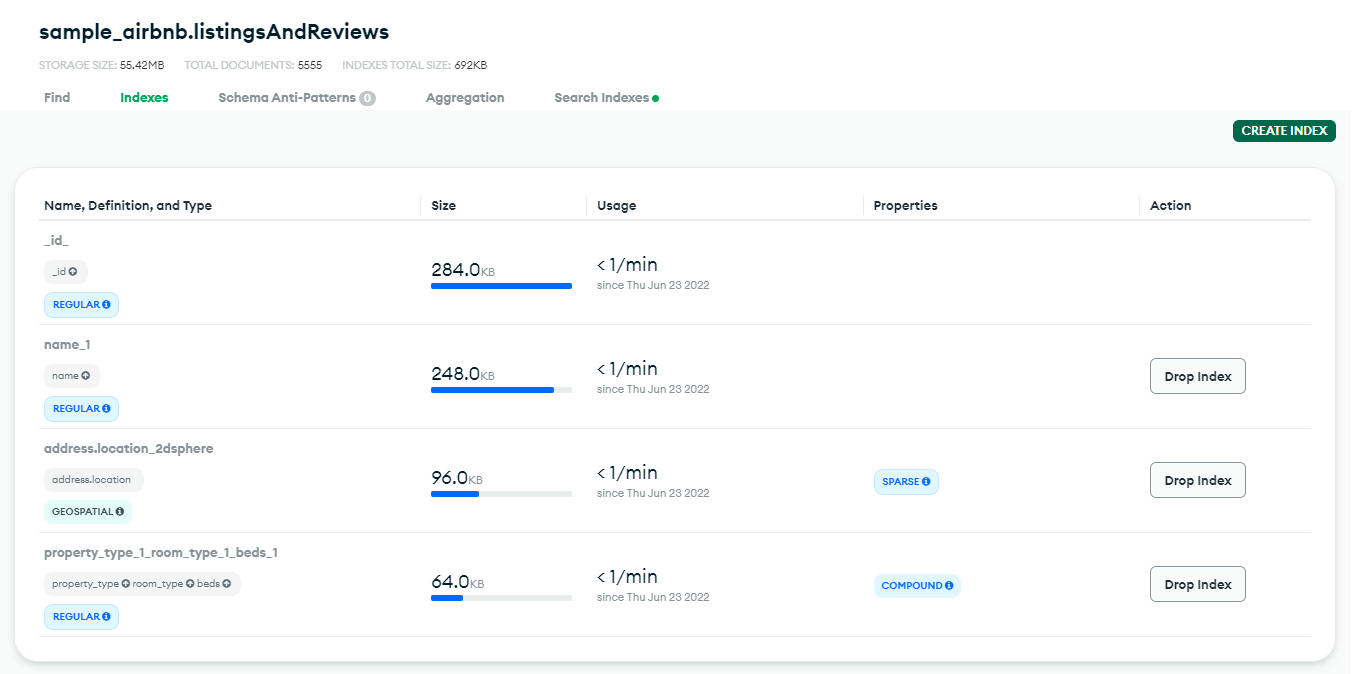
Odabirom konkretne kolekcije u okviru koje treba kreirati indekse, imamo nekoliko sekcija, a jedna od njih je *Indexes*, gde uz pomoć opcije *Create Index* možemo kreirati indeks nad željenim atributom u formatu:

Slika 20 – Format za kreiranje indeksa

Pored formata mogu se definisati i dodatne opcije pod nazivom *options* koje utiču na samo kreiranje indeksa, kao i *collation options* koje definišu specifična pravila jezika.

Na sledećoj slici možemo videti listu kreiranih indeksa za test bazu podataka *sample\_airbnb*, nad kolekcijom *listingAndReviews*. Nakon kreiraja samih indeksa, grafički su oni predstavljeni na slici 21. *Atlas* nam daje sve neophodne informacije o njima, kao što su naziv indeksa, definicija, tip, zatim veličina, koriščenje odnosno zastupljenost, propertije, odnosno vrstu kreiranog indeksa, kao i moguće akcije, u našem slučaju *Drop.*

Kako je ovaj grafički interfejs veoma intuitivan mogu ga koristiti i početnici bez velikog znanja i iskustva, a kako mnogi pojmovi i funkcionalnosti imaju i svoj dodatni opis, možemo i sami zaključiti o novim i nepoznatim stvarima.



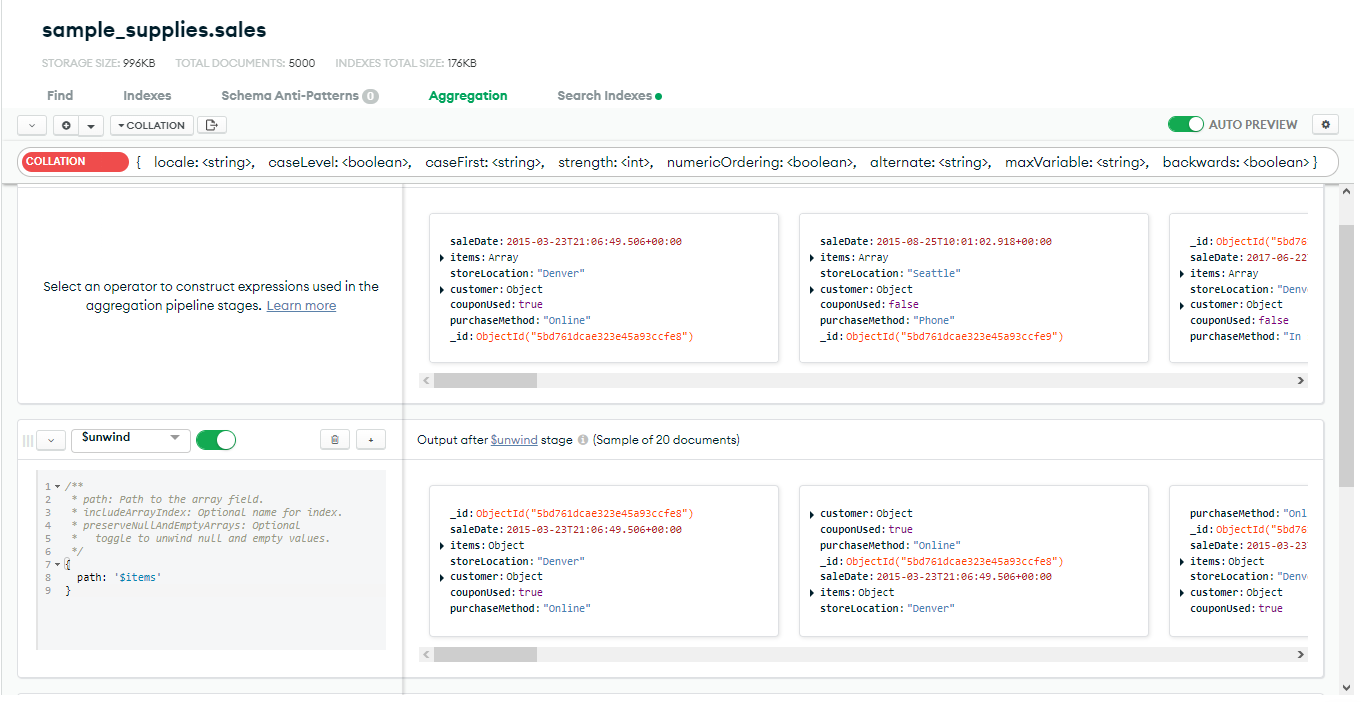
Slika 21 – Indeksi kolekcije u *Atlasu*

# 4.4 Kreiranje i upravljanje agregacionim *pipeline*-om

Atlas korisnički interfejs pruža mogućnost obrade podataka kreiranjem agregacionog *pipeline*-a. Agregacioni *pipeline*-i transformišu dokumente u objedinjene rezultate na osnovu odabranih *pipeline stage*-eva.

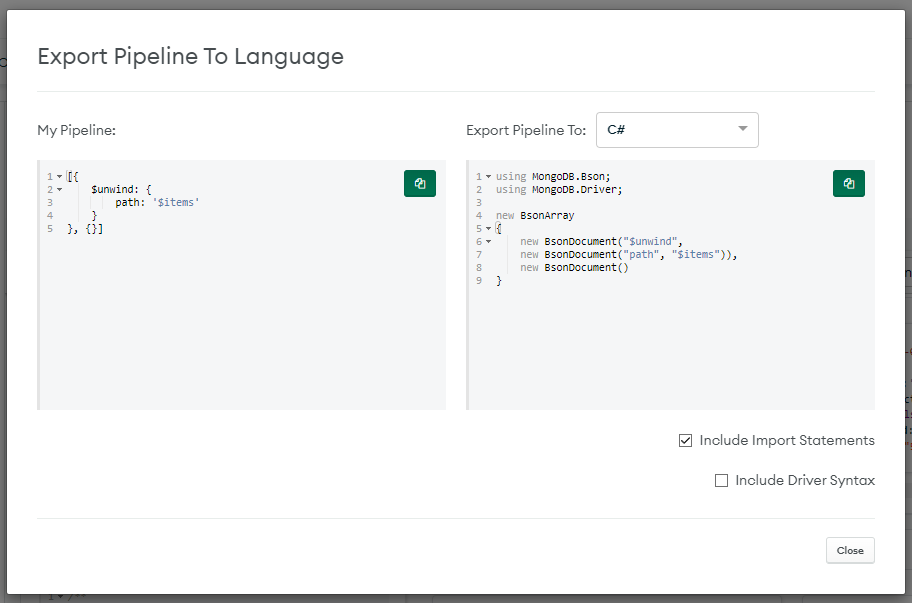
Za mogućnost kreiranja i izvršenja *aggregation pipeline*-a neophodno je imati prava pristupa *Project Data Access Read Only* ili veća prava, međutim za korišćenje *$out* *stage*-a potrebno je *Project Data Access Read/Write* pravo pristupa.

*Aggregation Pipeline Builder-*u možemo pristupiti odabirom konkretne kolekcije i okviru nje otvoriti sekciju *Aggregation* gde možemo odabrati *aggregation* *stage* iz padajućeg menija *Select*. Kako budemo menjali *stage*, *Atlas* će shodno tome ažurirati pregled dokumenata sa desne strane, baziranog na rezultatima trenutnog *stage*-a. Postoji mogućnost dodavanja više različitih *stage-*eva. Dodatne mogućnosti su podešavanja vezana za sam *pipeline*, kao i mogućnost definisanja *collation*-a. *Collation*, odnosno dokument za upoređivanje se koristi da bi se definisala pravila specifična za određeni jezik u kontekstu poređenja stringova, kao što su na primer pravila za velika slova ili oznake akcenata.



Slika 22 – *Aggregation pipeline*

*Atlas* *aggregation* *pipeline builder* je primarno dizajniran za kreiranje *pipeline-a*, pre nego za njihovo izvršavanje. *Pipeline builder* pruža jednostavan način za eksportovanje *pipeline* koda za izvršavanje u određenom programskom jeziku. Na sledećoj slici možemo videti primer ekspotovanja ovog koda za izvršavanje u C#:



Slika 23 - Eksportovanje *pipeline* koda

# Upravljanje klasterima

Svaki nivo klastera dolazi sa podrazumevanim skupom resursa. M10+ klasteri omogućavaju prilagođavanje kapaciteta skladišta. *Atlas* pruža sledeće konfiguracione opcije za skladištenje podataka, u zavisnosti od odabranog *cloud* provajdera i nivoa klastera:

* *Cluster Class*
* *Storage Capacity*
* *IOPS (AWS Only)*

# 5.1 Automatsko skaliranje

*Atlas* omogućava automatsko skaliranje klastera za sve nivoe, izuzev najvišeg nivoa klastera, u okviru *General* i *Low-CPU* nivoa klastera. Postoji mogućnost konfiguracije opsega nivoa koji *Atlas* koristi da automatski skalira nivo klastera, kapacitet skladištenja ili oba, u zavisnosti od načina korišćenja klastera. Za kontrolu troškova, možemo definisati opseg maksimalne i minimalne veličine klastera do kojih se klaster može skalirati i rasporediti. Automatsko skaliranje radi kao *rolling basis*, što znači da ne izaziva zastoje.

*Atlas* analizira sledeće karakteristike klastera da bi odredio kada da skalira klaster i da li da poveća ili smanji nivo klastera:

* Iskorišćenje *CPU*-a
* Iskorišćenje memorije

Da bi odredio iskorišćenje memorije, *Atlas* računa sledeću razliku:

*ukupna sistemska memorija – (slobodna + bafer + keš)*

A onda tu razliku deli sa ukupnom sistemskom memorijom. Pre nego što *Atlas* skalira nivo klastera, proverava da klaster nije izašao izvan specificiranog opsega minimalne i maksimalne veličine klastera.

***Scaling up nivoa klastera***

Ako je sledeći najviši nivo klastera u opsegu maksimalne veličine klastera, *Atlas* povećava klaster na sledeći nivo ako je bilo koji od sledećih kriterijuma ispunjen za bilo koji čvor u klasteru:

* Prosečno iskorišćenje *CPU*-a je premašilo 75% u poslednjih sat vremena
* Prosečno iskorišćenje memorije je premašilo 75% u poslednjih sat vremena

*Atlas* skalira klaster na sledeći nivo kada klaster nije povećavan u poslednjih sat vremena.

***Scaling down nivoa klastera***

Ako je sledeći najniži nivo klastera unutar opsega minimalne veličine klastera, *Atlas* smanjuje klaster na sledeći najniži nivo ako je bilo koji od sledeća dva kriterijuma ispunjen za sve čvorove u klasteru:

* Prosečno iskorišćenje *CPU*-a i memorije u poslednjih 24 sata je ispod 50%
* Klaster nije smanjen poslednjih 24 sata

Automatsko skaliranje skladišta klastera je podrazumevano omogućeno i *Atlas* automatski povećava skladište klastera kada iskorišćeni prostor na disku dostigne 90%.

# 5.2 *Pause, Resume* i *Terminate* klastera

***Pause klastera***

U zavisnosti od nivoa klastera, *Atlas* ili automatski pauzira klastere ili ručno kada je instanciran. *Atlas* automatski prestaje sa prikupljanjem *montoring* informacija u slučaju M0 klastera nakon nekoliko dana neaktivnosti. Ako se aktivnost ne detektuje 60 dana, *Atlas* tada u potpunosti pauzira klaster, onemogućujući bilo kakve konekcije sa njim dok ne bude *resume*-ovan. *Atlas* šalje *email* nedelju dana pre pauziranja klastera i još jedan *email* nakon samog pauziranja. U bilo kom trenutku, pauzirani klaster može se *resume*-ovati ili terminirati.

Kada je klaster pauziran, ne može se promeniti njegova konfiguracija i ne mogu se čitati i upisivati podaci. *Atlas* prestaje sa slanjem konfiguracionih obaveštenja i zaustavlja sve procese *backup*-ovanja pauziranih klastera.

***Resume klastera***

Da bi se nastavilo sa prikupljanjem *monitoring* informacija za *Atlas M0* klaster čiji je *monitoring* pauziran, neophodno je samo povezati se sa klasterom korišćenjem *MongoDB Driver*-a, *mongosh* ili *Data Explorer*-a.

Za *resume* *Atlas M0* klastera koji je sam *Atlas* automatski pauzirao zbog neaktivnosti ili *Atlas M10+* klaster koji je prethodno ručno pauziran neophodno je u okviru sekcije *Database Deployments* uz pomoć dodatnih mogućnosti klastera odabirom opcije ,,**...**“ iz padajuće liste odabrati opciju *Resume*.

***Terminate klastera***

Terminiranje klastera se vrši ručno, u istom delu gde i prethodno opisan postupak za *Resume*, samo odabirmo opcije *Terminate.* *Atlas* terminira klaster nakon što dovrši sve promene koje su u toku.

# Zaključak

Kao i svaki drugi *as-a-service* pristup, *DBaaS* je platforma za hostovanje koja koristi određeni mehanizam baze podataka. Sa hostovanim servisom baze podataka, sve što je neophodno za samu obradu i razvoj uključeno je kao deo usluga, kao što su infrastruktura, skladište, softver baze podataka, licence, mogućnost replikacije, obrada grešaka, kao i automatizacija backupovanja.

*Database-as-a-Service* model je idealan za male i srednje kompanije koje nemaju stručno osoblje u IT sektoru. Prebacivanje servisa i održavanja baze podataka na *DBaaS* provajdera omogucava malim i srednjim kompanijama da implementiraju aplikacije i sisteme koje inače ne bi mogli da razviju i podrže lokalno.

# Literatura

<https://www.nutanix.com/info/what-is-dbaas>

<https://www.techtarget.com/searchdatamanagement/definition/database-as-a-service-DBaaS>

<https://www.mongodb.com/database-as-a-service>