Zbornik radova Fakultet tehničkih nauka, Novi Sad

UDK: 004.4
DOI: https://doi.org/10.24867/28BE46Stanic

HORIZONTALNO SKALIRANJE K8S POD-OVA POMOĆU KEDA AUTOSCALER-A

HORIZONTAL SCALING K8S PODS USING KEDA AUTOSCALER

Marko Stanić, *Fakultet tehničkih nauka, Novi Sad*

Oblast - SOFTVERSKO INŽENJERSTVO I INFORMACIONE TEHNOLOGIJE

Kratak sadržaj - *U radu je prikazana implementacija upravljannja konfiguracijom za automatsko sklairanje Kubernetes pod-ova zasnovano na broju poruka u SQS-u. Opisan je princip rada Kubernetes kontejner orkestratora sa KEDA autoscaler-om uz integraciju sa Python bibliotekom za upravljanje cloud servisima i K8s resursima.*

Ključne reči: *Kubernetes, KEDA, SQS, horizontalno skaliranje*

Abstract - *The paper presents the implementation of a configuration management software for K8s pods and autoscaling based on number of messages in SQS. It is described how to use KEDA autoscaler with K8s container orchestrator and Python library for cloud and K8s resource management.*

Keywords: *Kubernetes, KEDA, SQS, horizontal scaling*

1. UVOD

U savremenom softverskom inženjeringu, skaliranje aplikacija postaje sve važniji aspekt koji utiče na njihovu pouzdanost i performanse. Kada aplikacije rastu i korisnički zahtevi se povećavaju, potrebno je da se skalira infrastruktura koja podržava aplikaciju.

Dva osnovna pristupa skaliranju su vertikalno i horizontalno skaliranje. Horizontalno skaliranje podrazumeva dodavanje novih mašina u postojeću infrastrukturu, što omogućava raspoređivanje opterećenja na više mašina i obezbeđuje bolju otpornost na kvarove [1]. U ovom radu će fokus biti horizontalno skaliranje *Kubernets pod*-ova uz pomoć KEDA *autoscaler*-a i metrika iz AWS SQS servisa i broja poruka u redu.

2. TEHNOLOGIJE

2.1. Kubernetes

Kubernetes (*K8s*) [2] je *open-source* platforma za automatizovanu upravu kontejnerima.

Kontejnerizacija je tehnologija koja omogućava pakovanje aplikacija i njenih zavisnosti u izolovane kontejnere, što omogućava da se aplikacija izvršava u bilo kojoj okolini koja podržava kontejnere. *Kubernetes* pruža alate i mehanizme za automatizovanu upravu kontejnerima i njihovim resursima, kao što su skaliranje, *load balancing*, *selfhealing* i *rollout* aplikacija. *Kubernetes* ima nekoliko ključnih komponenti koje omogućavaju automatizovanu upravu kontejnerima. Kontejneri se pokreću u *Kubernetes* klasteru, koji se sastoji od čvorova (eng. *nodes*) i kontrolne ravni (eng. *control plane*). Svaki čvor je fizička ili virtuelna mašina koja izvršava kontejnere, dok kontrolna ravan sadrži alate za upravljanje čvorovima i kontejnerima.

2.2. KEDA

KEDA (*Kubernetes-based Event Driven Autoscaler*) [3] je *open-source* komponenta za automatsko skaliranje *Kubernetes* aplikacija (*Deployment* i *ReplicaSet*) na osnovu različitih događaja i metrika koje se dešavaju u *Kubernetes* klasteru ili u drugim *cloud* servisima.

KEDA podržava veliki broj događaja i metrika, uključujući *AWS SQS* [4], *Azure Service Bus*, *Apache Kafka*, *RabbitMQ* i brojne druge.

2.3. SQS

Amazon Simple Queue Service (*SQS*) je *fully-managed* servis u okviru *Amazon Web Services* (*AWS*) platforme, koji omogućava korisnicima da lako kreiraju i upravljaju redovima poruka u *cloud* okruženju. *SQS* je dizajniran da pomogne korisnicima da odvoje komponente svojih aplikacija, tako što će omogućiti različitim delovima aplikacije da komuniciraju asinhrono, bez potrebe za direktnom vezom.

SQS je skalabilan i otporan na greške, što ga čini idealnim izborom za različite aplikacije koje se zasnivaju na *cloud* tehnologijama i koje moraju da podrže veliki broj korisnika i visok nivo opterećenja. *SQS* je dostupan u dve verzije: standardna verzija i *FIFO* verzija (*First-In-First-Out*). Standardna verzija podržava veliki broj poruka u redu, sa minimalnim kašnjenjem u isporuci poruka, dok *FIFO* verzija omogućava garanciju da će poruke biti isporučene tačno onim redosledom kojim su postavljene u red.

SQS pruža fleksibilnost u upravljanju porukama, uključujući mogućnost slanja, primanja, brisanja i promene prioriteta poruka. Takođe omogućava i definisanje pravila za ponovno slanje poruka koje nisu uspešno isporučene.

SQS se može integrisati sa drugim *AWS* servisima, kao što su *Amazon S3*, *Amazon EC2*, *AWS Lambda* i mnogi drugi, kako bi se omogućilo jednostavno upravljanje porukama i stvaranje pouzdanih, skalabilnih i otpornih aplikacija u cloud okruženju.

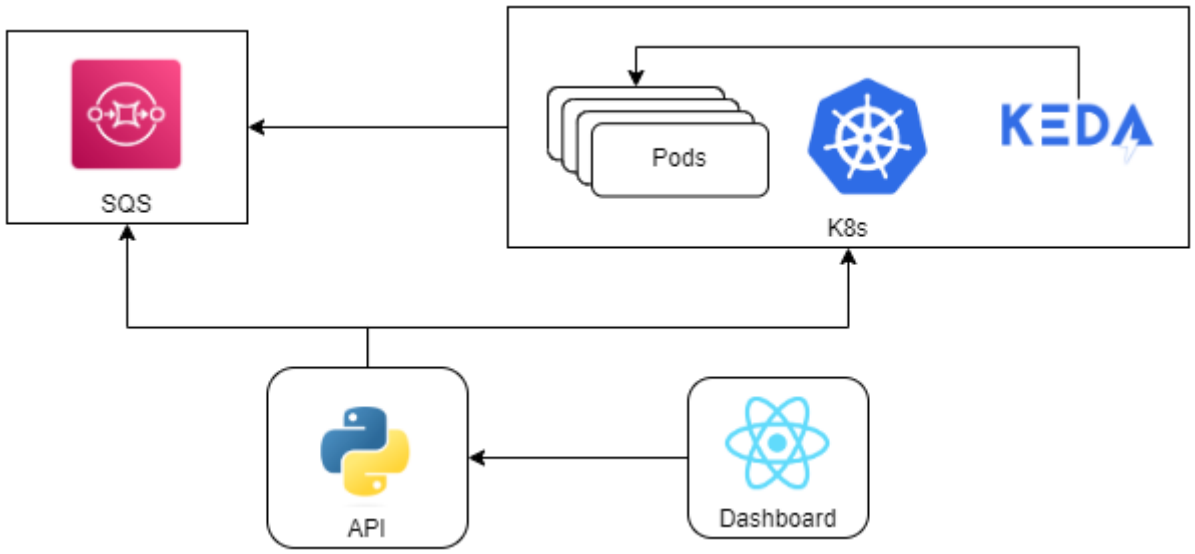
NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Željko Vuković, docent.

3. ARHITEKTURA

Na slici 1 nalaze se sledeće komponente:

- *K8s* klaster na kome će biti *deploy*-ovana aplikacija koja će se dinamički skalirati. Podrška za *Kubernetes* je dostupna na gotovo svim većim *cloud* provajderima. Za testiranje u lokalnom okruženju korišćen je *Minikube* [5].
- *KEDA autoscaler* gde će se definisati skalirajući resursi (*Scalers*) koji se koriste za prikupljanje metrika o opterećenju aplikacije. U ovom slučaju to će biti broj poruka koji se nalazi u redu (*queue*)
- Za korišćenje *SQS* servisa je neophodan *AWS* korisnički nalog, tako da aletrnativa za njegovu upotrebu može biti *ElasticMQ* [6] koji ima isti interfejs kao i *SQS*. To znači da je moguće testirati lokalno bez ikakvih troškova, a za prelazak na *SQS* je potrebno samo izmeniti odgovarajuću konfiguraciju.
- *Backend* je *REST Api* koji pruža interakciju sa *Amazon SQS* i *K8s* klasterom. Sve dostupne funkcionalnosti mogu se pregledati kroz *OpenAPI* dokumentaciju na */docs* putanji.
- *Dashboard* je korisnički interfejs za komunikaciju sa prethodnim *REST* servisom, koji omogućava grafički pregled, dodavanje novih kao i brisanje redova. Takođe u svaki red je moguće dodati navedeni broj poruka ili obrisati sve (*purge*). Kreiranje *Scaled Object* resursa je dostupno popunjavanjem forme.



Slika 1. *Arhitektura Sistema*

4. IMPLEMENTACIJA

Za pokretanje *minikube* klastera, na *Windows* operativnom sistemu potrebno je izvršiti komandu `minikube start` u *PowerShell*-u. Status je moguće proveriti kroz *Docker Desktop* gde će se pojaviti kontejner sa nazivom *minikube* u stanju *running* (slika 4.1.2) ili kroz *Lens* - radno okruženje za K8s. Instalacija KEDA *autoscaler*-a se vrši pomoću *helm* [7] menadžera paketa za *Kubernetes*. Pravila skaliranja se mogu definisati na osnovu željene maksimalne i minimalne veličine broja replika aplikacije. Na primer, ako se želi imati minimalno dve i maksimalno pet replika aplikacije u *Kubernetes* klasteru, mogu se definisati pravila skaliranja koja će skalirati aplikaciju prema gore kada broj poruka u redu poraste iznad određenog praga, a skalirati aplikaciju prema dole kada broj poruka u redu padne ispod nekog drugog praga.

KEDA ima dve različite faze tokom procesa automatskog skaliranja:

Faza aktiviranja (ili deaktiviranja) je trenutak kada *KEDA* operator mora da odluči da li je opterećenje takvo da treba skalirati od nule ili na nulu. Ova akcija je bazirana na rezultatu funkcije *IsActive* i primenjuje se samo na 0-1 skaliranje. Postoje slučajevi kada su pravila za aktiviranje potpuno drugačija, na primer sa *Prometheus* skalarom gde vrednosti mogu ići od -x do x.

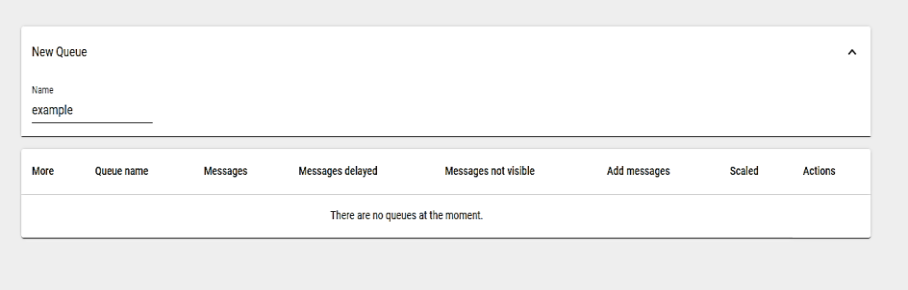
Faza skaliranja je trenutak kada postoji jedna instanca i tada *Horizontal Pod Autoscaler* (*HPA*) donosi odluke zasnovane na konfiguraciji koja je generisana na osnovu podataka u *ScaledObject* resursu. Ova faza se primenjuje na 1-n skaliranje.

Web API koji služi za upravljanje redovima i Kubernetes resursima je kreiran uz pomoć *FastAPI* radnog okvira u *Python* programskom jeziku.

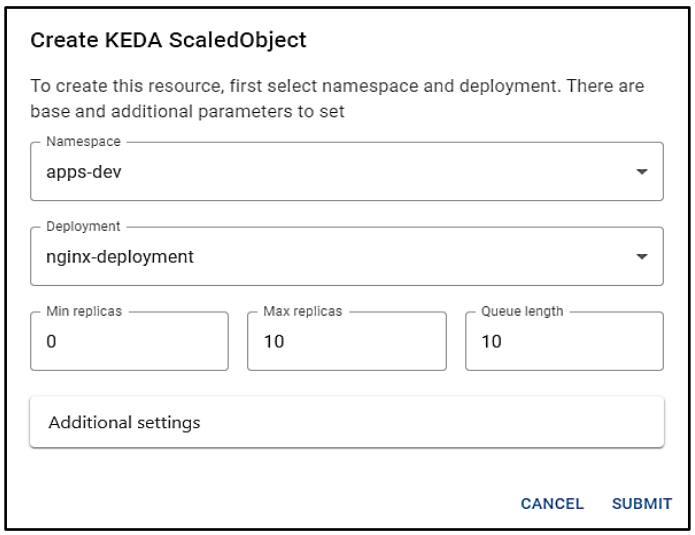
Za kreiranje novog reda je potrebno proslediti samo naziv u formi koja je prikazana na slici 2.

Akcije koje je moguće izvršiti nad redom su: dodavanje poruka, brisanje poruka, brisanje i kreiranje *ScaledObject* resursa (slika) za koje je potrebno izabrati *deployment* iz odgovarajućeg *namespace*-a.

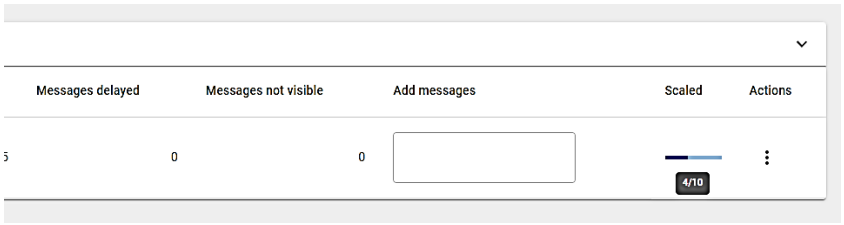
Nakon uspešnog kreiranja ScaledObject-a skaliranje se aktivira i status je moguće videti na slici 4 gde je prikazan *progress bar* koji predstavlja odnos trenutnog i maksimalnog broja instanci, koji su definisani za *ScaledObject*.



Slika 2. *Tabela formom za dodavanje reda*



Slika 3. *Forma za kreiranje ScaledObject-a*



Slika 4. *Status skaliranja*

5. ZAKLJUČAK

U ovom radu je predstavljen primer horizontalnog skaliranja aplikacija u *Kubernetes* klasteru uz pomoć *KEDA autoscaler*-a. Objašnjen je princip rada *Kubernetes*-a kao kontejner orkestratora i način upotrebe *autoscaler*-a zasnovano na događajima, tačnije broja poruka u *AWS SQS*. Time je omogućena brža obrada podataka ukoliko je to zahtevano na osnovu definisane konfiguracije, a sa druge strane ušteda resursa kada nije neophodno da aplikacija bude aktivna.

Horizontalno skaliranje se može primeniti u nekoliko slučajeva: za povećanje performansi ukoliko nije moguće dodati više resursa na jednoj mašini (fizičkoj ili virtuelnoj), kada je potrebno omogućiti maksimalnu dostupnost, u slučaju da dođe do otkaza neke mašine, ostale nastavljaju da rade bez zastoja.

6. LITERATURA

[1] IBM Data Science - Best Practices https://ibm.github.io/data-science-best-practices/scaling.html

[2] Kubernetes https://kubernetes.io/

[3] KEDA https://keda.sh/

[4] SQS https://aws.amazon.com/sqs/

[5] Minikube https://minikube.sigs.k8s.io/docs/start/

[6] ElasticMQ https://github.com/softwaremill/elasticmq

[7] Helm https://helm.sh/

Kratka biografija:

Marko Stanić rođen je 31.5.1997. godine u Šapcu. Završio je osnovne akademske studije 2020. godine na Fakultetu tehničkih nauka. Upisao je master studije iste godine, studijski program Softversko inženjerstvo i informacione tehnologije

kontakt: marko.stanic97@icloud.com