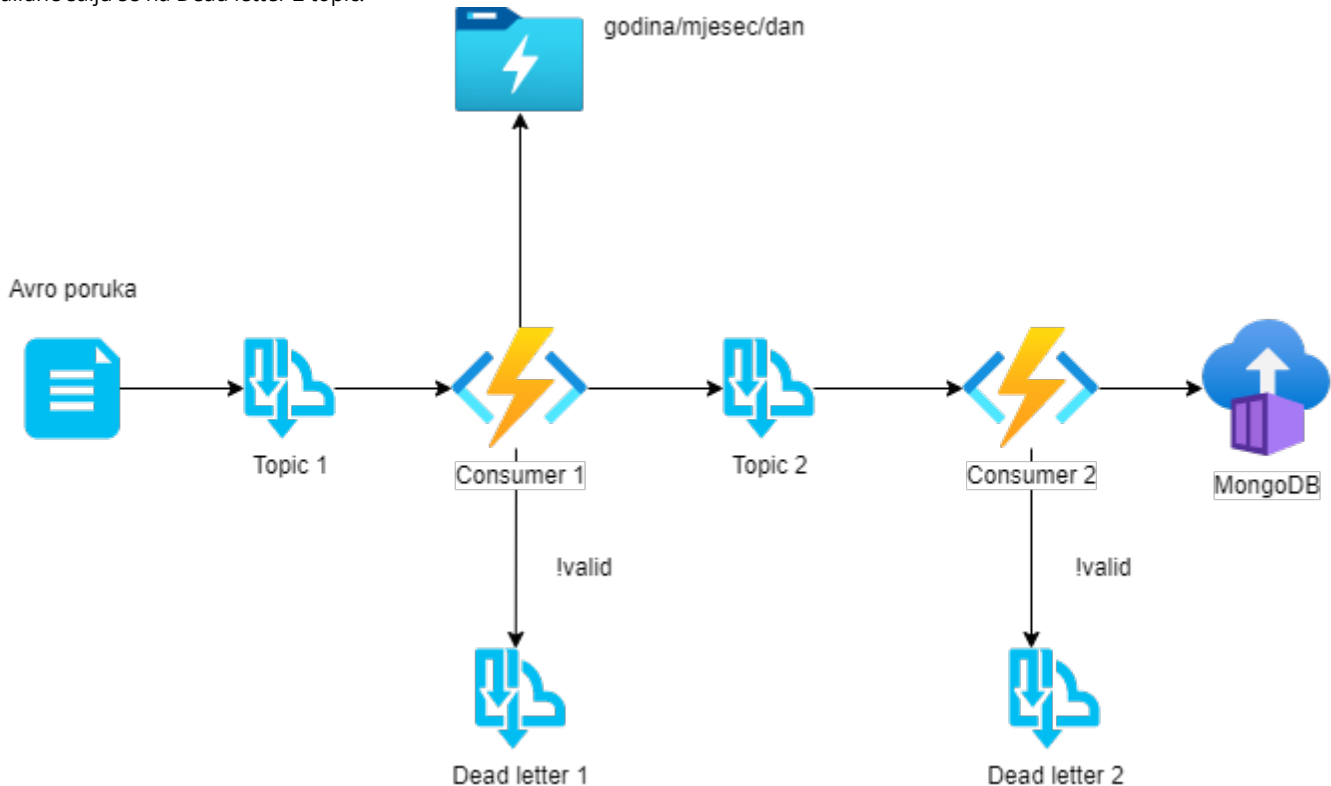


4. laboratorijska vježba - zadatak

Cilj 4. laboratorijske vježbe je nadopuniti *data pipeline* novom komponentom koja je na dijagramu označena kao *Consumer 2* te poruke spremljene na MongoDB vizualizirati pomoću Grafane. Opcionalni dio 4. laboratorijske vježbe je izrada *CI/CD pipeline-a*.

Consumer 2 čita poruke s *Topic 2* te radi validaciju poruka prema poslovnoj logici. S obzirom da prima poruke s *Topic 2*, sigurno je da su poruke u ispravnom formatu. Poruke koje su validne dalje sprema na *MongoDB* koristeći *lookup* funkcionalnost. U slučaju da u bazi postoji *Order* s istim *OrderID* kao primljena poruka, poruka koja sadrži točno jedan *OrderItem* pridodaje se na postojeći *Order* (join podataka). U slučaju da ne postoji *Order* s istim *OrderID*, kreira se novi *Order* i poruka se dodaje kao novi *Order* u bazu. Poruke koje nisu validne šalju se na *Dead letter 2* topic.



Consumer 2

1. Pomoću Azure portala kreirajte Topic 2 i dodajte mu Subscription.
2. U Visual Studio Codeu pod Azure Functions, kliknite na Create Function. U sljedećim koracima odaberite:
 - a. Template: Azure Service Bus Topic Trigger
 - b. Naziv funkcije: po želji (npr. Tim1-Consumer2-labfer)
 - c. Service bus namespace: naziv namespace-a koji ste koristili u prethodnim laboratorijskim vježbama
 - d. Naziv topica: naziv topica kojeg ste kreirali u koraku 1.
 - e. Naziv subscriptiona: naziv subscriptiona kojeg ste kreirali u koraku 2.
3. Pogledajte izgenerirani sadržaj datoteke function.json.
4. Podsjetnik, kao pomoć za izradu custom handlera vam može poslužiti sljedeći GitHub repozitorij: <https://github.com/Azure-Samples/functions-custom-handlers/tree/master/go>

Filtracija poruka po poslovnoj logici

1. U datoteku server.go dodajte novu funkciju koja će kao argument primiti objekt tipa Order i validirati narudžbu prema poslovnoj logici.
 - a. Ovdje možete dodati nekoliko provjera prema želji. Jedan primjer može biti provjera količine naručenih proizvoda. Ako pretpostavimo da je količina proizvoda u narudžbi obavezno veća od 0, možete provjeriti je li Order.Quantity veći od 0.
2. U datoteku server.go dodajte novu handler funkciju u kojoj će se nalaziti kod Consumera 2:

```
func handlerTopic2(w http.ResponseWriter, r *http.Request) {  
    // ovdje napišite kod za Consumer 2 koji e se izvršiti kad na  
    topic 2 doe nova poruka  
}
```

3. Handler funkciju nadopunite tako da ostvaruje sljedeću funkcionalnost Consumera 2:

- a. *Ukoliko poruka nije validna prema poslovnoj logici, potrebno ju je poslati na Dead letter 2 topic/queue.*
 - i. Pomoću Azure Portala kreirajte Dead letter 2 topic ili Dead letter 2 queue (Service Bus Queue ili Azure Storage queue).
 - ii. Nadopunite kod handler funkcije tako da ostvaruje navedenu funkcionalnost.
 - iii. Za prosljeđivanje poruke na drugi topic ili queue, koristite *output bindings*: <https://docs.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings?tabs=csharp>.
- b. *Ukoliko je poruka validna prema poslovnoj logici, potrebno ju je spremi na MongoDB.* (Sljedeći korak)

Spremanje poruka u MongoDB pomoću Consumera

Kako biste mogli vidjeti i analizirate podatke zapisane u MongoDB, instalirajte *MongoDB Compass* alat sa sljedeće poveznice: <https://www.mongodb.com/try/download/compass>

Adresa za spajanje na *host*, na kojem se nalazi postojeća MongoDB baza, koja je napravljena za laboratorijske vježbe i na koju ćete spremati svoje podatke, je sljedeća:

i MONGO_URI: mongodb://ferVjestinaUser:vt6VrdaPR2bn5uMK35.246.190.109:27017

Korisničko ime: ferVjestinaUser
Lozinka: vt6VrdaPR2bn5uMK

Svaki tim treba kreirati svoju bazu u koju će spremati svoje podatke. Koristite sljedeću konvenciju imenovanja: <ime_tima>-<tip_resursa>-<naziv_aplikacije>

Kao što je već napisano u projektnom zadatku, potrebno je nadopuniti *Consumer 2* handler funkciju kako bi se ostvarila *lookup* funkcionalnost.

- i** Poruke koje su validne dalje sprema na *MongoDB* koristeći *lookup* funkcionalnost. U slučaju da u bazi postoji *Order* s istim *OrderID* kao primljena poruka, poruka koja sadrži točno jedan *OrderItem* pridodaje se na postojeći *Order* (join podataka). U slučaju da ne postoji *Order* s istim *OrderID*, kreira se novi *Order* i poruka se dodaje kao novi *Order* u bazu.

U slučaju da ne uspijete ostvariti potpunu *lookup* funkcionalnost, pokušajte zapisati barem svaku pojedinačnu poruku u MongoDB kako biste mogli kreirati vizualizacije u sljedećem podzadatku.

Za implementaciju rješenja koristite sljedeće pakete:

- <https://pkg.go.dev/go.mongodb.org/mongo-driver/mongo>
- <https://pkg.go.dev/go.mongodb.org/mongo-driver/bson>

Kako bi vaše rješenje uspješno radilo i kad pokrenete funkcije u *Cloud* okruženju, a ne samo lokalno, potrebno je kreirati samo jednog klijenta za komunikaciju s bazom, koji će se proslijediti *handler* funkciji prilikom svakog poziva. Modificirajte vašu funkciju `main` na sljedeći način:

```

func main() {
    usn, _ := os.LookupEnv("MONGO_USERNAME")
    pass, _ := os.LookupEnv("MONGO_PASS")
    credential := options.Credential{
        Username: usn,
        Password: pass,
    }
    client, err := mongo.NewClient(options.Client().ApplyURI(
("mongodb://35.246.190.109:27017").SetAuth(credential))
    if err != nil {
        log.Fatal(err)
    }
    ctx := context.Background()
    err = client.Connect(ctx)
    if err != nil {
        log.Fatal(err)
    }
    defer client.Disconnect(ctx)

    p, ok := os.LookupEnv("FUNCTIONS_CUSTOMHANDLER_PORT")
    if !ok {
        p = "8080"
    }
    addr := fmt.Sprintf(":%s", p)
    http.HandleFunc("/ServiceBusTopicTrigger1", handlerTopic1)
    http.HandleFunc("/ServiceBusTopicTrigger2", handlerTopic2(ctx,
client))
    http.ListenAndServe(addr, nil)
    log.Printf("Listening on address: %s/ServiceBusTopicTrigger1",
addr)
}

```

Možete vidjeti da `handlerTopic2` funkcija kao argument prima klijenta kojeg ćete koristiti prilikom spremanja i čitanja iz MongoDB. Nadopunite `handlerTopic2` funkciju traženom funkcionalnosti.

U slučaju uspješne implementacije, dokumenti vaše kolekcije u MongoDB bi trebali biti slični dokumentu na slici:

vjestina.orders

Documents

Aggregations

Schema

Explain Plan

Indexes

Validation

FILTER

{ field: 'value' }

ADD DATA



VIEW



```

_id: "US-2011-118892"
OrderDate: "1/2/2011"

```

```

OrderDate: "6/2/2011"
ShipDate: "6/2/2011"
ShipMode: "Standard Class"
CustomerId: "DH-13075"
CustomerName: "Dave Hallsten"
Segment: "Corporate"
City: "San Miguelito"
State: "Panama"
Country: "Panama"
Market: "LATAM"
Region: "Central"
OrderItems: Array
  0: Object
    ProductID: "OFF-AP-10002317"
    Category: "Office Supplies"
    SubCategory: "Appliances"
    ProductName: "Hamilton Beach Refrigerator, Silver"
    Sales: "400.704"
    Quantity: "2"
    Discount: "0.4"
    Profit: "20.024"
    ShippingCost: "21.38"
  1: Object
    ProductID: "TEC-AC-10001221"
    Category: "Technology"
    SubCategory: "Accessories"
    ProductName: "Memorex Memory Card, USB"
    Sales: "81.984"
    Quantity: "2"
    Discount: "0.4"
    Profit: "-19.136"
    ShippingCost: "6.21"
  2: Object
    ProductID: "OFF-BI-10000719"
    Category: "Office Supplies"
    SubCategory: "Binders"
    ProductName: "Wilson Jones Hole Reinforcements, Clear"
    Sales: "9.576"
    Quantity: "6"
    Discount: "0.4"
    Profit: "-0.984"
    ShippingCost: "0.81"
OrderPriority: "Medium"

```

Grafana

U sklopu ovog zadatka vizualizirat ćete podatke u Grafana dashboardu. Grafana je vizualizacijski alat koji omogućuje da postavljate upite, vizualizirate i razumijete podatke iz mnogobrojnih izvora podataka.

Kao izvor podataka koristit ćete Mongo bazu u koju ste slali podatke iz **Consumera 2**. Mongo baza nalazi se na sljedećoj adresi kao i u prethodnome zadatku.

Postavite konfiguraciju u Grafani:

1. Napravite besplatni račun na Grafana Cloud-u (<https://grafana.com/products/cloud/>). Kreiranjem besplatnog računa započinje vaš 14-dnevni free trial, što će biti dovoljno za trajanje ovih vježbi.
2. U Grafana Cloud konzoli pokrenite Grafanu klikom na gumb *Launch*.

Overview

GRAFANA CLOUD
avokzumvwdhyjxvhcc

+ Add Stack

GRAFANA ENTERPRISE

Licenses (1)
Download

SECURITY

API Keys
Advanced Auth
OAuth Clients

SUPPORT

Open a Ticket
Tickets

BILLING

Manage Subscription

ORG SETTINGS

My Plugins (0)
My Dashboards (0)
My Referral Codes (1)
Members
Settings

Grafana Cloud Portal














Configure your Grafana Cloud resources and manage users, API keys, billing and more.

Open a Support Ticket

Subscription

Know more about your subscription.

You are currently subscribed to a free trial of Grafana Cloud Pro. Your subscription includes:

- | | | |
|---|---|--|
|  Grafana Instance |  Unlimited Dashboards |  24x7x365 Support |
|  Prometheus Endpoint |  Custom Auth |  Email Support |
|  Graphite Endpoint |  Custom Domain |  Phone Support |
|  Logs Backend |  Enterprise Plugins | |
|  Traces Backend |  Critical SLA: 2 hours | |


Manage Subscription

Billing

Couldn't load billing info.

avokzumvwdhyjxvhcc


Manage your Grafana Cloud Stack.

 **Grafana**
Press Launch to start using Grafana or view Details to manage Grafana Plugins.

Launch

Details

Active Users: 1

 **Prometheus**
Set up and manage your Prometheus metrics service.

Send Metrics

Details

Active Users: 1 Current Usage: 0 Current Active Series: 0

3. Dodajte Mongo bazu kao izvor podataka.
 - a. Configuration Data sources MongoDB
 - b. Upišite konekcijske podatke i spremite izvor



Data Sources / MongoDB

Type: MongoDB

Settings

Permissions

Insights

Cache

Name ⓘ

MongoDB

Default



Connection

Connection string

mongodb://ferVjestinaUser:vT6VrdaPR2bn5uMK@35.246.190.109:27017

Authentication

Credentials ☐

CA Certificate ☐

Client Certificate ☐



OK

Back

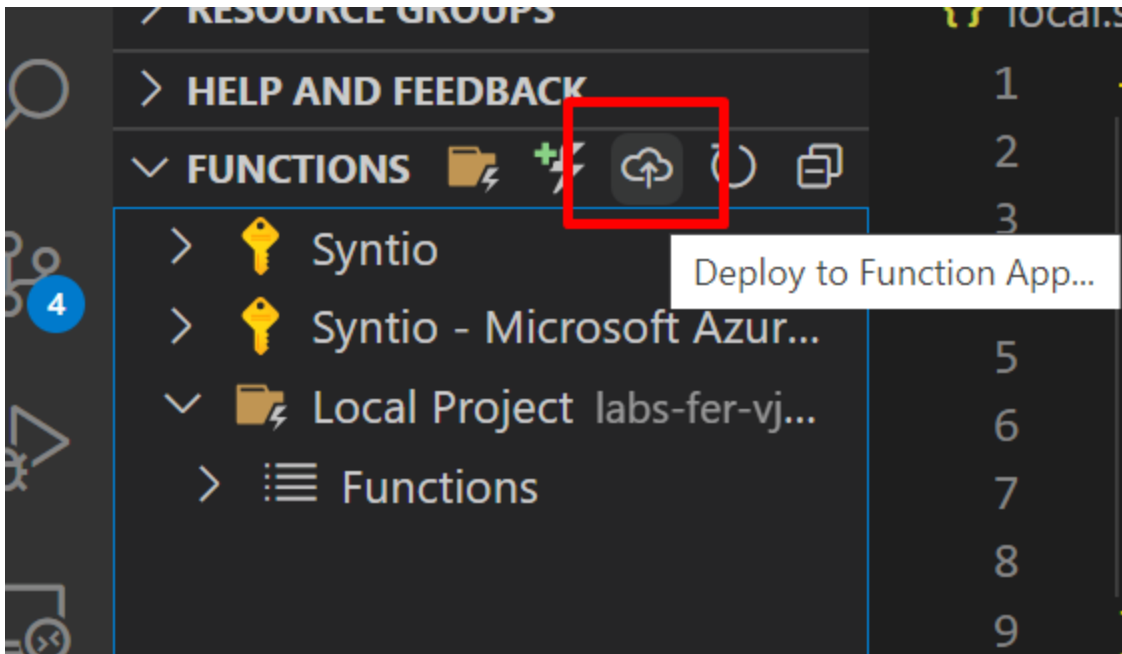
Explore

Delete

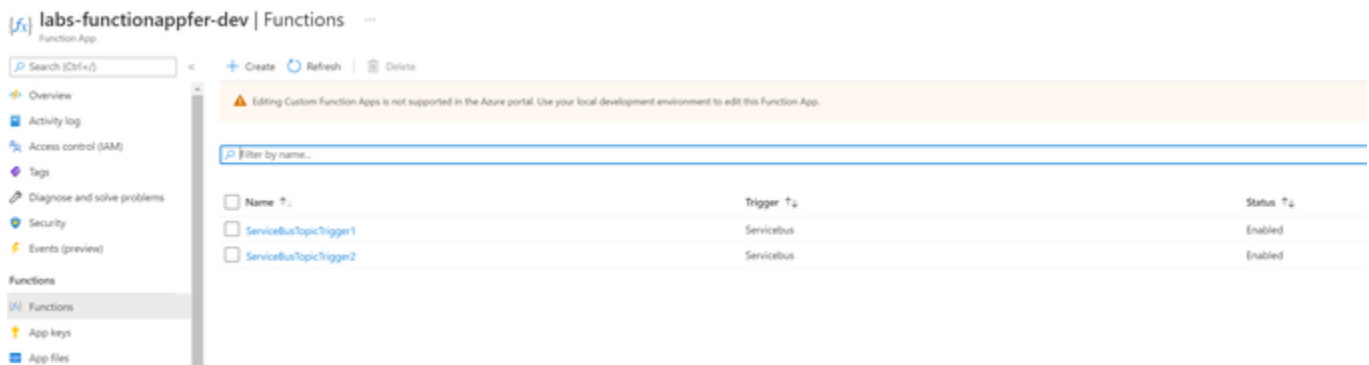
Save & test

Kreirajte svoj dashboard:

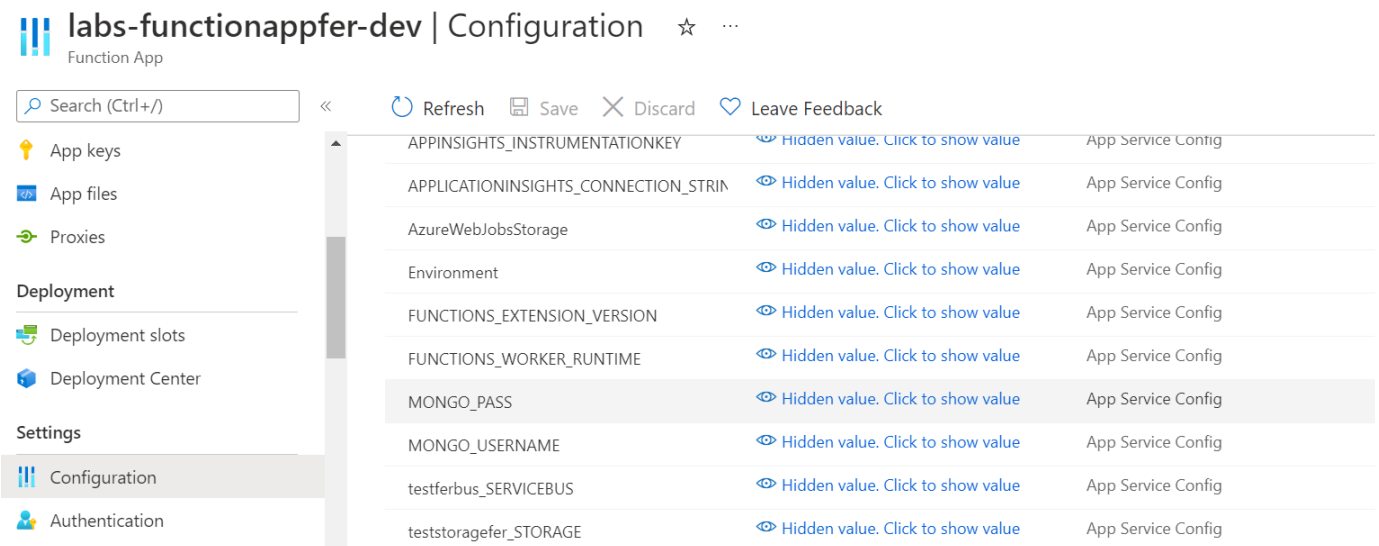
1. Create Dashboard Add a new panel
2. Dobavite podatke iz Mongo baze koje želite vizualizirati pomoću upita u konzoli na dnu prozora.



Odaberite opciju *Create new Function App* te upišete potrebne tražene parametre. Nakon što je deploy uspješno završen, otidite na Azure portal kako biste vidjeli svoju novokreiranu funkciju. Pod *Functions* biste trebali vidjeti svoje *Consumer* funkcije i na koji događaj se pokreću. U ovome slučaju to bi bila poruka koja je došla na *Azure Service Bus*.



U izborniku *Settings->Configuration* možete dodati varijable koje vaše funkcije koriste za spajanje na *AzureServiceBus*, *Storage* i *MongoDB*, te to i napravite klikom na opciju *+New application setting*.



Budući da ste do sada na projektu pokretali svoje Azure funkcije lokalno, opcionalni zadatak na kraju je kreirati CI/CD pipeline koji će automatizirati proces deploymenta vašeg novokreiranog data pipeline-a. Dodatne upute za kreiranje CI/CD pipeline možete pronaći na sljedećem linku: <https://docs.microsoft.com/en-us/azure/azure-functions/functions-how-to-azure-devops?tabs=dotnet-core%2Cyaml%2Ccsharp> . Također pogledajte što je tzv. *Zip deployment*: <https://docs.microsoft.com/en-us/azure/azure-functions/deployment-zip-push> , koji je korišten i u ovome primjeru za brzi deploy funkcija na *Cloud*. U *CI pipelineu* potrebno je napisati korake koji će napraviti .zip artefakt koji će sadržavati sve datoteke vašeg git repozitorija, koje su potrebne za izvršavanje Azure funkcija, te ga predati vašem *Azure Function App*.


- Otiđite na svoj Azure DevOps
- Pronađite *Pipelines* u lijevom izborniku
- Kliknite opciju *New pipeline* te kreirajte template koji ćete kasnije nadopuniti
 - Connect **Azure Repos Git**
 - Select odaberite svoj git repozitorij
 - Configure **Starter pipeline**
 - Review Napišite taskove za CI pipeline.
Kako biste lakše napisali .yaml datoteku, sve korake koje radite ručno kako biste pokrenuli svoje *Azure funkcije* sada samo trebate preoblikovati u task-ove. Taskovi koje trebate imati u .yaml su redom:
 - Go installer task koji će instalirati odgovarajuću verziju GoLang-a
 - Command line task (Napomena: s obzirom da trebate kreirati zip cijelog projekta vjerojatno ćete morati koristiti task "CmdLine" kako biste se pozicionirali na pravu granu vašeg git repozitorija gdje vam se nalazi cijeli kod. Ukoliko vam je kod *Azure funkcija* na grani *main*, možete zanemariti ovu napomenu.)
 - Go task kako biste instalirali sve potrebne pakete
 - Go task kako biste kreirali izvršnu datoteku .exe
 - ArchiveFiles task koji će kreirati .zip direktorij od cijelog git projekta (*includeRootFolder* opciju postavite na vrijednost *false*)
 - PublishBuildArtifact task kako biste kreiranu .zip datoteku objavili kao artefakt koji će dalje koristiti vaš *CD pipeline*.
- Kliknite *Save and run*
- Nakon što ste kreirali datoteku za CI pipeline, *Azure DevOps* vam nudi opciju da spremite kreiranu .yaml datoteku tako da napravite novi *commit*. Spremite novokreiranu datoteku koju možete i kasnije nadopunjavati/uređivati, na svoj git repozitorij.
- Kada se pipeline pokrene, možete pratiti tijek izvršavanja poslova (eng. task) koje ste prethodno definirali. Na slici je primjer koraka izvođenja jednog pipeline-a.

- Kada se vaš build uspješno izvrši kliknite na naziv vašeg pipeline-a u gornjem dijelu prozora. (preporuka za naziv: <naziv_tima>_CI). Klikom na "1 consumed" trebao bi vam se prikazati arhitektura kojeg ste kreirali pomoću CI pipeline-a

← Artifacts	
Published Consumed	
Name	Size
▼ 📁 drop	5 MB
📄 test.zip	5 MB

- Preuzmite zip datoteku i provjerite je li u njoj zapakiran vaš cijeli kod projekta
- U lijevom izborniku pod sekcijom **Pipelines** odaberite **Releases**. Pomoću ovog pipeline-a stvorit ćete CD pipeline. Odaberite opciju **Create Release**.
- Odaberite **Deploy a function app to Azure Functions** te kliknite **Apply**.

Select a template

Or start with an  Empty job

functio

Others

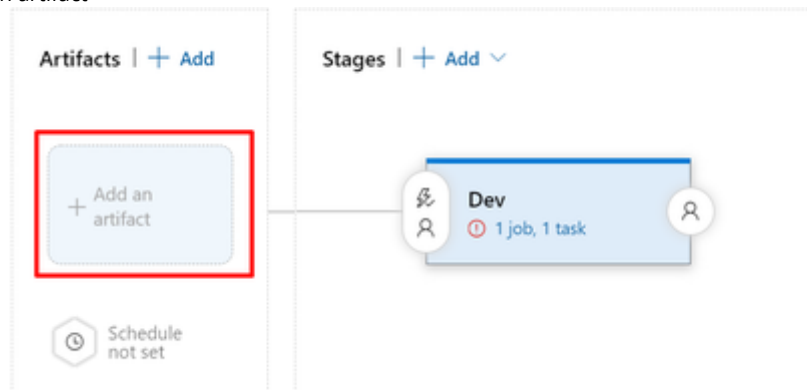


Deploy a function app to Azure Functions

Deploy your serverless application to Azure Function App.

Apply

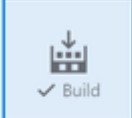



- Stage name Dev, te kliknite "x" na vrhu tog prozora
- Kliknite na opciju **Add an artifact**



- Pod **Source** stavite svoj novokreirani **CI pipeline**. Kliknite **Add**.

Add an artifact

Source type

 Build
  Azure Repos ...
  GitHub
  TFVC

5 more artifact types ▾

Project * ⓘ


labs-fer-vjestina ▾

Source (build pipeline) * ⓘ

labs-fer-vjestina-CI ▾


14. Omogućite kontinuirani deployment.

Artifacts | + Add

 labs-fer-vjestina-CI

Schedule not set

Stages | + Add ▾

 Dev

Continuous deployment trigger

Build: labs-fer-vjestina-CI

☒ Enabled

Creates a release every time a new build is available.

Build branch filters ⓘ

No filters added.

+ Add ▾

Pull request trigger

Build: labs-fer-vjestina-CI

☐ Disabled

15. Kliknite na *View stage tasks*, kako je prikazano na slici, te ispunite potrebna polja. Odaberite *Azure subscription* te pod *AppServiceName* stavite naziv *Azure function app* funkcije koja vam se vrti na cloud-u u vašoj resursnoj grupi te koju ste trebali kreirati u prethodnom podzadatku vezanome za "ručni" deployment.
16. Unutar polja *Variables* možete dodati varijable za različita okruženja (npr. *development* ili *test*), te to i napravite klikom na gumb *+Add*. Tu dodajte sve potrebne varijable koje ste koristili za spajanje na *Azure Service Bus*, *Azure storage account* i *MongoDB*.

Pipeline variables

Predefined variables [↗](#)

Filter by keywords

Name
Environment
FUNCTIONS_WORKER_RUNTIME
testferbus_SERVICEBUS
teststoragefer_STORAGE
MONGO_PASS
MONGO_USERNAME

+ Add

Vaš *CI/CD* sada bi trebao biti gotov te kliknite na *Save* u gornjem desnom kutu.

Kako bi pokrenuli deployment kliknite na svoj novokreirani *CD* pipeline te ćete u Dev pravokutniku vidjeti opciju *deploy*. Kliknite na to te pratite kako se vaši koraci deploja događaju.

Ukoliko deploy prođe, trebali biste na cloud-u vidjeti svoje deployane *Azure* funkcije.