21 Eylül DevNot Etkinliği Serverless Kısmı Adımları

1. Catalog’dan Functions servisine gidiyoruz.
2. Start Creating’e tıklıyoruz.
3. Create Action’a tıklıyoruz.
4. Github Sayfasından Serverless dosyasından ‘read-slack…’ ile başlayan dosyanın içinde bulunan kodu bu action’ın içindeki kodla tamamen değiştiriyoruz.
5. Kodumuz daha tam hazır değil ama şimdilik Save’e tıklayıp yaptığımız değişiklikleri kaydedelim.
6. Bu Action’ın dışarıdan hiçbir Authentication olmadan çağrılabilmesi için dışarıya Web Action olarak açmamız lazım. Bu yüzden solda bulunan sekmelerden Endpoints’e tıklıyoruz.
7. Koyu yazılmış Web Action sekmesi altında Enable as Web Action yanındaki kutucuğu işaretliyoruz. Aşağıda yeni bir kutucuk açıldı fark ederseniz ama üstü daha kapalı çünkü daha kaydetmedik.
8. Ardından Save’e basarak yaptığımız değişikliği kaydediyoruz. Böylelikle url artık açıldı ve aktif durumda dışarıdan http request artık atabiliriz.
9. Sol sekmeden Code’a basıp geri dönelim kodumuza. Burada aslında kuyruğa Slack’ten aldığımız mesajı yazacağımız kuyruk servisini yaratmamız gerekiyor. Bu servisin ismi Event Streams. Event Streams servisine erişebilmemiz için credential’larına ihtiyacımız var. Bunun için kodda aktifleştirmemiz gereken ve kendi credential’larımızı koyacağımız bir comment kısmı var. ‘// You can insert your Language Translator Service Credentials here below’ diye başlayan 4 satırlık bir alan mevcut. İlk satırdan sonraki 3 satırı comment’den çıkaralım ‘//’ karakterlerini silerek. Save’e basıp kaydedelim. Ama dikkat ederseniz daha herhangibir credentials ile doldurmadık ilgili kısımları. Çünkü daha Event Streams servisimiz yok.
10. Event Streams servisi yaratmak için IBM Cloud Catalog’a geri dönelim (cloud.ibm.com/catalog). Arama sekmesine Event Streams yazalım. Bu servise tıklayalıp gidelim. Bizi üstte Event Streams başlığı altında bir servis yaratma sayfası karşılayacak. Eğer Select a Region kısmında London varsa ve aşağıda Select a pricing plan altındaki Plan seçeneklerinde Standard yoksa, Free versiyonu kullanabileceğimiz region’da değiliz demektir. Select a region ın altındaki şehir ismine tıklayıp Dallas’ı seçelim. Plan Sekmesinden Lite (yani Free) Plan’ı seçtiğimizden emin olup sağ taraftan Create’e tıklayalım.
11. Yeni açılan sayfa Event Streams’in anasayfası. Getting Started Sekmesi açık şu anda. Hemen yanında. Topics kısmı var. Buraya tıklayalım çünkü slack mesajını action’ın içinde Event Streams’in hangi topic’ine yazacağımızı söylememiz lazım. Bir topic yaratmamız lazım. Create Topic’e tıklıyoruz sağdan. Topic ismine istediğimizi yazabiliriz ama şimdilik language koyalım adını çünkü dili tanımlama için göndereceğiz mesajı. Partitions’ı 1 seçip ilerleyelim Next’e basıp. Message Retention kısmı mesajı kuyrukta ne kadar bekleteceğimiz. A day seçelim çok önemli değil bizim için şu anda. Create Topic’e tıklayıp topic’i yaratalım.
12. Topics sekmesinde en başta language isimli bir topic gözükmeyebilir. Biraz bekleyin gelecek. Gelmezse sayfayı yenileyin. Topic’imiz ve servisimiz artık hazır. Şimdi servis. credential’larımızı action’a vermek üzere almamız lazım.
13. Sol tarafta Manage’in altında Service Credentials kısmı var ona tıklayın.Sağ altta new credentials a tıklayıp yeni bir credential yaratma sayfasına geçin. Hiçbir şey değiştirmeden Add’e tıklayın sağ altta.
14. Service credentials-1 adı altında bir credential yaratıldı. ACTİONS sekmesinden View Credentials kısmına tıklayın. ‘apikey’ in yanında bulunan ‘rLsYorCisUg7kKsfK…’ ile başlayan apikey’inizi bir yere kopyalayın. Yine aynı yerde aşağıda ‘kafka\_admin\_url’ kısmı var. Bu url’den servisimize erişeceğiz. Bunu da bir yere kopyalayın. Aslında bütün bilgileri bir yere kopyalayın.
15. Event Streams servisindeki topic’e slack mesajımızı yazabilmemiz için bu servise erişmemizde kullanacağımız son parametre olan Authentication Token’a ihtiyacımız var. Terminalimizden bu token’ı alabiliriz. Öncelikle ibmcloud’a giriş yapmış olmamız lazım terminalden. ibmcloud iam oauth-tokens komutunu çalıştırın. IAM token: Bearer ‘ın yanında bir token var onu bir yere kopyalayın. Şimdi hazırız.
16. Read-slack ile başlayan Action’a dönelim. Url, apikey ve token kısmını kopyaladığımız 14-15’teki değişkenlerle değiştirelim. Save ile kaydedelim.
17. Sol üstten Functions’a tıklayalım. Create’e tıklayalım. Start Creating’e tıklayalım. Quick Start Templates e tıklayalım. Choose Template kısmından Event Streams i seçelim. Package name aynı kalsın. Diğer tüm ayarlar aynı kalabilir. Next’e tıklayalım. Trigger name aynı kalsın. Instance sekmesinden Input your own credentials ı seçelim.

14. Adımda kopyaladığımız bilgiler burdaki bilgilerle aynı. Karşılık gelen yerleri koyalım. Topic kısmına önce yaratmış olduğumuz topic olan language yazalım. Deploy a tıklayalım. event-streams/process-message adı altında bir action yaratıldı ve bu action’ı harekete geçiren bir trigger var şu anda.

1. Github’ta bulunan event-streams/process-message dosyasını aynı isimli action’ın içine koyalım. Save e basalım.
2. Sol üstte Functions’a tıkladık. Start Creating’ten Install Packages’a tıklayalım. Watson’ı seçelim. Language Translator’ı seçelim. Install a tıklayalım. Açılan ekranda language-translator-v3 adı altında hazır Watson action’ları geldi. Identify’a tıklayalım.Github’daki identify.js dosyası ile değiştirelim. Save edelim.
3. Language Translator servisi yaratmadık onu yaratmamız lazım. IBM Cloud Catalog’a Language Translator aratıp servisi yaratalım Lite Plan’da Event Streams servisini yarattığımız gibi. Region ve diğer ayarlar aynı kalabilir Lite (free) plan olması önemli sadece. Service Credentials kısmından yine credentials yoksa yaratıp view credentails kısmından credentials ları bir yere not edelim.
4. Identify action’ımıza geri dönelim. Solda Parameters sekmesine tıklayalım. Add Parameter’a tıklayıp parametrelerimiz verelim.

Parameter Name Parameter Value

url Tırnak içinde 20. de kopyaladığımız credentialdaki url

version "2018-05-01"

iam\_apikey Tırnak içinde 20. de kopyaladığımız credentialdaki apikey

1. Sol üstten Actions a gidelim. Language-translator-v3 package’ın altında bulunan translate e tıklayalım. İçindeki kodu yine github’daki kod ile değiştirelim. Save ile kaydedelim.
2. Sol üstten Actions a gidelim.Sağ üstten Create seçip. translation-parser adından bir aciton yaratalım default package’ımızın içinde (Create). Yine github daki aynı isimdeki kodla değiştirelim. Save ile kaydedelim.
3. Mesajda gelen duyguyu alabilmek için Natural Language Understanding servisi yaratmamız lazım. Catalog’a gidip yaratalım yine Free Plan’da.
4. Servis ekranından sol üstten Service Credentials sekmesine gidip yine View credentials’dan credentials’ları bir yere kopyalayalım.
5. Translation-parser action’ına geri gelelim. Ortadaki yeşil comment kısmındaki url ve apikey kısımlarına nlu dan aldığımız credential’daki url apikey i koyalım. (version kısmına -> “2019-07-12“). Kaydedelim
6. Sol üstten Functions’a gidelim.Start Creating Install Packages -> Watson -> Natural Language Understanding->Install seçelim.
7. Actions sayfasında natural-language-understanding-v1 sekmesinden analyze ı seçelim.
8. Analyze.js’i github dan kopyalayıp yapıştıralım buraya. Save e basalım.
9. Actions sayfasına gelip create action diyip nlu-parser isminde actin yaratalım. Github da aynı isimdeki dosyadaki kodu buraya yapıştıralım.
10. Functions sayfasından Create Sequence a tıklayalım sequence a isim verip içine sırasıyla identify,translate,translation-parser,analyze,nlu-parser verelim.
11. Sequence ımıza gidip Endpoints kısmından enable as web action diyelim.Save.
12. event-streams/process-message action a gidip <YOUR-SEQUENCE-WEB-ACTION-ENDPOINT> kısmını 32 den aldığımız endpoint ile değiştirelim.
13. Api.slack.com’a gidip read-slack ile başlayan action’ın endpoint’ini slack’e verelim. Api.slack.com -> Your Apps -> Event Subscriptions -> Request Url kısmına.

(Subscribe to Bot Events te message.im olduğundan emin olalım)

1. Prom-writer klasörüne gidelim. ibmcloud cf push komutunu çalıştıralım. App deploy olana kadar bekleyelim.
2. Terminalden prom-writer’ın url’ini alalım. Prometheus dosyasında prometheus.yml içinde en aşağıda targets in içindeki karşılık gelen alana yapıştıralım.
3. Prometheus dosyasında ibmcloud cf push komutunu çalıştıralım. App deploy olana kadar bekleyelim.
4. Grafana dosyasında ibmcloud cf push komutunu çalıştıralım. App deploy olana kadar bekleyelim.
5. Grafana’ya Login olalım username:admin password:admin ile. Next diyelim. Add Data Source. Prometheus seçelim. Kendi Prometheus uygulamamızın url adresini iligili alana yazalım. Save diyelim.