IoT Node-Red Uygulamaları İçin CI/CD Yaklaşımı

Görkem Aktaş[1], *Emin Konukoğlu*[2], *Berat Buğra Kodal*[3], *Yücel Aydın*[4]

[1,2,3,4] Kontrol ve Otomasyon Mühendisliği Bölümü
İstanbul Teknik Üniversitesi, İstanbul

aktasg17@itu.edu.tr ,kodal21@itu.edu.tr, konukoglu17@itu.edu.tr, aydinyu@itu.edu.tr

Özetçe

Günümüz gelişen teknolojileri gün geçtikçe daha karmaşık hale gelmektedir. Bu karmaşıklık projelerin geliştirme ve yönetim süreçlerini etkileyerek maliyetlerini artırabilmektedir. Maliyetleri düşürmek ve zamandan tasarruf edebilmek için çeşitli otomasyonlar ve metotlar geliştirilmektedir. Bu süreçler özellikle hata payını azaltmak ve zaman tasarrufu kısmına odaklanmaktadır.

Özellikle günümüz internet uygulamalarında sürekli gerekli olan versiyonlama ihtiyacı DevOps (Development and Operations) konsepti aracılığı ile CI/CD (Continous Integration / Continous Delivery) yöntemleri kullanılarak giderilmektedir. Ancak nesnelerin interneti uygulamalarında bu konuda eksiklikler bulunmaktadır. Bu çalışmada nesnelerin interneti uygulamalarında sektörde kullanılan Node-Red uygulaması için CI/CD süreç önermesi yapılmıştır. Bu yöntem aracılığı ile Node-Red uygulamalarında hatasız teslimat ve zamandan kazanç konularına odaklanılmıştır.

Abstract

Today's developing technologies are getting more and more complex day by day. This complexity can affect the development and management processes of projects and increase their costs. Various automations and methods are being developed to reduce costs and save time. These processes especially focus on reducing the margin of error and saving time.

Especially in today's internet applications, the need for versioning, which is constantly required, is met by using CI/CD methods through the DevOps concept. However, there are deficiencies in this regard in internet of things applications. In this study, a CI/CD process proposal has been made for the Node-Red application used in the industry in Internet of Things applications. Through this method, node-Red applications focused on error-free delivery and time saving.

1.GİRİŞ

Günümüz teknolojileri içerisinde karmaşık yapıları yönetmek ve hata payını olabildiğince indirgemek son derece önemlidir. Özellikle internet tabanlı ürünlerde ihtiyaca yönelik sürekli değişiklikler yapılabilmektedir. Bu değişikler ürün kullanıcılarına daha iyi hizmet verebilmek veya ürün üzerindeki hataları gidermek üzerine olabilmektedir. Yapılan değişiklikler neticesinde hata payını olabildiğince indirgemek ve zaman maliyetini düşürerek süreçleri hızlandırmak ana amaç haline gelmiştir ve bu problemler DevOps yöntemlerinin sağladığı çeviklik ve metotlardaki dayanıklılık ile giderilmeye çalışılmaktadır [1].

DevOps uygulamalarının yararı olmakla birlikte mevcut uygulamalara entegre edilmesinde veya uygulanmasında bir takım sorunlar yaşanabilmektedir. Bu zorlukların farklı sebepleri olsa da özellikle kültürün yeni gelişmesi ve buna adaptasyon sağlanma durumu önem arz etmektedir. Nesnelerin interneti uygulamaları için bu durumu analiz ettiğimizde ek olarak teknolojik eksiklikler ile de karşılaşılmaktadır. Nesnelerin interneti konusundaki DevOps yaklaşımları hala bir standarda oturmamış ve farklı önermeler yapılmaktadır. Bunlara ek olarak mevcut DevOps teknolojilerini mevcut nesnelerin interneti teknolojilerine adapte edebilecek vasıflı eleman eksikliği de bizleri burada daha kolay anlaşılabilir ve uygulanabilir standartlara itmektedir [2].

Tüm zorluklarına rağmen gelişen teknoloji içerisinde DevOps uygulamaları gereklilik halini almaktadır. Mevcuttaki pek çok uygulama eski sistemleri ve teknolojileri terk ederek Docker, Kubernetes, Jenkins, GitOps gibi teknolojiler etrafında süreçlerini yönetmek istemektedir. Bu konuda da pek çok yaklaşım karşımıza çıkmaktadır [3].

Özellikle otomasyon ağırlıklı nesnelerin interneti uygulamalarında DevOps konseptinin sürekli teslimatı konusu karşımıza çıkmaktadır. Mevcut nesnelerin interneti uygulamaları içerisinde OTA (Over The Air) teknolojisi ile uzaktan güncelleme sağlanabilmektedir. Ancak bu güncellemeler sürekli bir gelişim ve versiyonlama ihtiyacını da beraberinde getirmektedir. Bu ihtiyaç neticesinde güncelleştirme operasyonlarında sürekli teslimat yaklaşımı ihtiyacı doğmaktadır [4].

DevOps kavramının etkin olduğu bulut bilişim uygulamalarında da nesnelerin interneti ihtiyaçlarından da bahsedilmektedir. Nesnelerin interneti konseptinin bulut bilişim teknolojilerine entegrasyonu genel itibariyle çözülebilir bir durum olmasına karşılık bazı ek zorluklar ile karşılaşılmaktadır. Çünkü nesnelerin interneti cihazlarından gelen veriler dağıtık ve yönetilmesi büyük

verilerle çalışıldığından zordur. Ayrıca cihazlara maksimum erişilebilirlik ve olabildiğince gerçek zamanlı bir işleme de gerekmektedir [5].

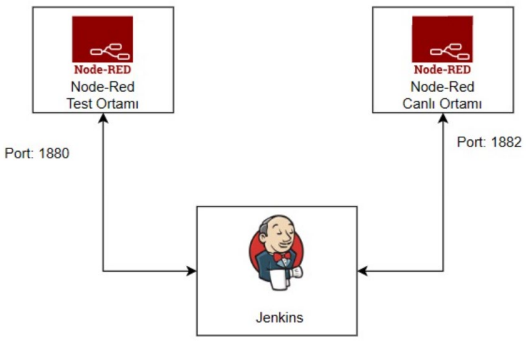
Nesnelerin interneti uygulamalarındaki DevOps zorluğunu aşabilmek için önermeler sağlanmaktadır. Yapılan bir çalışma göstermektedir ki nesnelerin interneti uygulamalarında agile ve CI/CD yaklaşımı önemli rol oynamaktadır. Yapılan çalışmada DevOps kültürünün nesnelerin interneti uygulamalarının bulut teknolojileri ile entegrasyonu detaylıca incelenmiş ve faydalarından bahsedilmiştir. Bu çalışmada bahsedildiği üzere nesnelerin interneti çalışmalarında DevOps yaklaşımları uygulanarak proje geliştirmede hızlı geliştirmeler için önemli bir temel olmaktadır [6].

Ek olarak DevOps'un sağlamış olduğu hız ve çeviklik beraberinde test süreçlerinin daha ciddi geliştirilmesi gerekliğini ortaya çıkarmıştır. Hızlı versiyonlama ve geliştirmeler neticesinde hata payı artabilmekte ve bunun giderilmesi gerekmektedir. Bu konuda halihazırda geliştirilmiş DevOps test yöntemleri nesnelerin interneti uygulamalarında kullanılabilmektedir. Yapılan çalışmalarda bu test yöntemleri incelenmiş ve test metotlarının kullabilirliği denenmiştir [7].

Tüm bu ihtiyaçlardan ve bu ihtiyaçlar doğrultusunda yapılan çalışmalardan yola çıkarak günümüz nesnelerin interneti uygulamalarında kullanılan Node-Red uygulaması üzerinde çalıştırılan projelerde versiyonlama ihtiyacını giderecek şekilde bir CI/CD çalışması üzerinde durulmuştur. Bu çalışmadaki amaç Node-Red kullanan sistemlerde versiyonlama işlemlerini hızlandırmak ve sürekli teslimatı sağlayabilmek için test ortamlarında geliştirilmiş ortamların canlı hizmet veren ortamlara teslimatında minimum sorunla karşılaşılması ve eğer bir hata ile karşılaşılırsa da kolayca tespit edilebilmesini sağlamak şeklindedir.

2.Node-Red İçin CI/CD Kurgusunun Yapılması

Çalışma sürecinde amaç Node-Red üzerinde geliştirilen bir uygulamayı bir diğer Node-Red ortamına sorunsuz bir şekilde aktarabilmek üzerinedir. Bu kapsamda mevcut isterleri sağlayabilecek yapının kurgusu yapılmıştır.

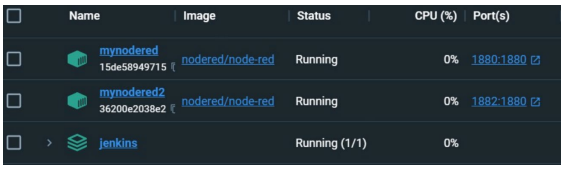


Şekil 1: Node-Red ve Jenkins Mimarisi

Hedeflenen çıktıyı alabilmek için bir CI/CD aracı kullanılması gerekmektedir. Bu hususta Jenkins uygulamasının kullanılması uygun bulunmuştur. Jenkins aracılığı ile test ortamında bulunan Node-Red uygulaması dışarıya aktarılarak yine Jenkins aracılığı ile canlı hizmet ortamına aktarılmaktadır. Bu esnada HTTP protokolü üzerinden ilgili bilgilerin alınması hedeflenmiştir.

2.1 Kurgulanan Ortamın Hazırlanması

Hedef kurguyu test edebilmek için çalışma ortamında biri test ortamını diğeri canlı hizmet ortamını temsil edecek şekilde iki adet Node-Red sunucusu gerekmektedir. Ek olarak buradaki CI/CD sürecini sağlayabilmek için de Jenkins aracının çalışacağı bir sunucu gerekmektedir. Sunucu maliyetlerini minimuma indirgemek için çalışmanın gerçekleştirildiği ortamda Docker üzerinden ilgili sunucular hazır hale getirilmiştir.

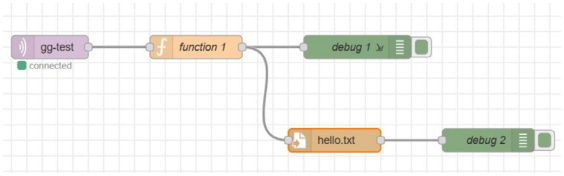


Şekil 2 : İlgili Sunucuların Docker Üzerinde Çalıştırılması

Şekil 2'de de görülebileceği üzere ilgili sunucular Docker aracılığı ile kullanılabilir hale getirilmiştir. Tüm sunucular çalıştıkları bilgisayarın ağ katmanını kullanmaktadır. Bu sebeple konteynırlar içerisinde yer alan portlar bilgisayar ağındaki katman portlarına yönlendirilmiş böylelikle dış ağ üzerinden haberleşebilir hale gelmişlerdir.

2.2 Node-Red Uygulamasının Hazırlanması

Çalışma boyunca yapılacak testlerde gerçek uygulamalara yakın bir simülasyon gerçekleştirilmesi hedeflenmiştir. Bu sebeple test ortamı amacıyla kullanılacak olan Node-Red üzerinde bir test uygulaması hazırlanmıştır.



Şekil 3: Node-Red Uygulama Akışı

Şekil 3'de görülen akış şeması çalışma esnasında kullanılan Node-Red uygulamasını temsil etmektedir. Burada bir MQTT (Message Queuing Telemetry Transport) alıcısı bulunmaktadır ve MQTT üzerinden gelen istekleri alarak bir fonksiyona iletmektedir. Fonksiyon çıktısı bir dosyaya yazdırılmaktadır ve çalışma tamamlanmaktadır.

Yapılan testlerde MQTT sunucusuna iletilen mesajların başarılı şekilde Node-Red ortamına iletilip iletilmediği kontrol edilmiştir. Bu kapsamda MQTT sunucusu üzerine test mesajı iletilerek çıktılarda bu mesaj gözlenmiştir.

2.3 Jenkins Kurgusunun Hazırlanması

CI/CD aracı olarak kullanılacak Jenkins için öncelikle bir pipeline nesnesi oluşturulmuştur. Bu pipeline nesnesi üç aşama içerecek şekilde kurgulanmıştır. İlk aşamada test ortamından ilgili yapıyı Jenkins'e aktarmaktadır. Buradaki amaç başarılı ve hatasız şekilde test ortamındaki yapının elde edilmesidir. Elde edilen yapı JSON formatında saklanmaktadır. Ardından bir sonraki aşamada ilgili JSON dosyası kontrol edilerek başarılı bir şekilde yapının aktarılıp aktarılmadığı kontrol edilmektedir. Üçüncü aşamada ise elde edilen yapı canlı hizmet amacıyla kullanılan Node-Red sunucusuna aktarılmaktadır.

2.4 Sistem Verilerinin Eldesi

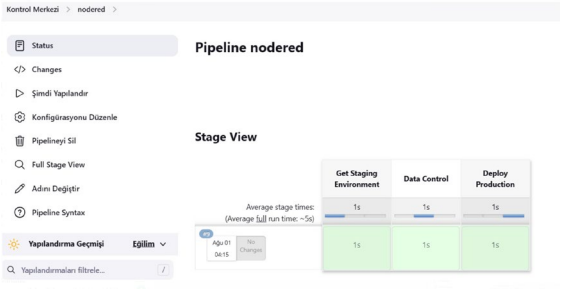
Mevcut sistemlerdeki yapıyı içeren veriyi elde edebilmek için Node-Red uygulamasının sunmuş olduğu REST API yapısından faydalanılmıştır. İlk olarak test ortamındaki tüm akışlar REST API aracılığı ile elde edilmektedir. Ardından yine Node-Red'in sunmuş olduğu REST API yapıları kullanılarak canlı hizmet ortamına aktarılmaktadır.

3. Yapının Test Edilmesi

Hazırlanan yapının sağlıklı çalışıp çalışmadığı test ortamı içerisinde bulunan uygulamanın aktarılması hedeflenerek test çalışmaları gerçekleştirilmiştir. Bunun için Jenkins üzerinden bir yapılandırma başlatılmıştır. Yapılan testler sırasında Jenkins üzerindeki pipeline nesnesinin konsol çıktıları incelenerek yapının doğru taşınıp taşınmadığı kontrol edilmiştir.

3.1 Jenkins'in Çalıştırılması

Hazırlanan yapı karmaşık olmayacak ve mevcut ortamı aynen yansıtıp eşleyecek şekilde kurgulanmıştır. Bu sebeple herhangi bir parametre girdisi kullanılmamıştır.



Şekil 4: Jenkins'in Çalıştırılması

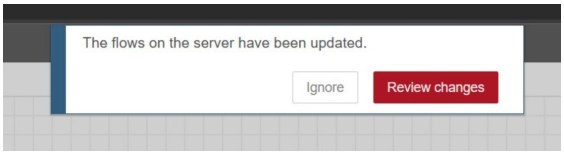
Şekil 4'de görüleceği üzere Jenkins pipeline çalıştırılma çıktısı görülmektedir. Bu çıktıdan anlaşılmaktadır ki yapı sağlıklı şekilde çalışmıştır ve teslimat işlemi tamamlanmıştır. Eğer bir hata meydana gelmiş olsaydı Jenkins hata alınan adımda ilgili uyarıları sağlamış olacaktı ve böylelikle hata alınan nokta tespit edilebildiği için hızlıca giderilebilecektir.

Şekil 5: Hatalı Teslimatın Tespit Edilmesi

Örneğin Şekil 5'de yer alan son teslimatta veri kontrolünün yapıldığı adımda hata alındığı görülmektedir. Burada ilgili aşamanın kayıt çıktıları okunarak hata tespiti sağlanarak hızlı çözüm üretilebilmektedir.

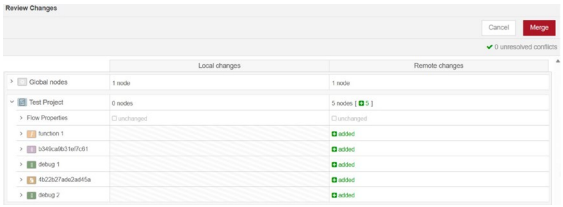
3.2 Jenkins'in Çalıştırılarak Node-Red'e Teslimat Yapılması

Jenkins çalıştırıldıktan ve başarılı teslimat mesajı alındıktan sonra Node-Red tarafında gerekli kontroller sağlanmıştır. Bu kontroller sonrasında Node-Red tarafından bir teslimat yapıldığı ve değişikliklerin kontrol edilmesi gerektiğine dair bir mesaj ile karşılaşılmıştır.



Şekil 6: Node-Red Değişiklik Kontrol Mesajı

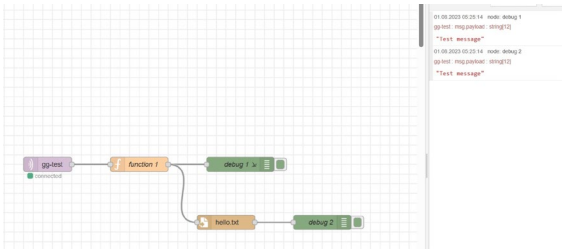
Bu mesaj neticesinde Node-Red yapılan değişikliklerin kontrol edilmesi gerektiğini ve eğer bir problem var ise buna göre değişikliklerin iptal edilmesi gerektiğini iletmektedir. Değişikliklere ilişkin açıklamada ortama hangi parçaların dahil olacağı veya sistemden çıkarılacağı belirtilmektedir. Burada beklenmeyen bir durum olduğunda yapılan değişiklik iptal edilerek canlı hizmet veren Node-Red sunucusu üzerindeki hatalar engellenebilmektedir.



Şekil 7: Node-Red Değişikliklerinin İncelenmesi

Şekil 7'de görüleceği üzere test ortamındaki yapıların canlı hizmet ortamına ekleneceği bilgisi iletilmektedir.

Buradaki değişikliklerin onaylanması durumunda ilgili teslimat gerçekleştirilmiş olacaktır.



Şekil 8: Teslimatın Test Edilmesi

Yapılan teslimat sonrası Node-Red arayüzünde test ortamıyla birebir aynı olacak şekilde akış yapısı görüntülenmiştir. Ayrıca canlı hizmet sunucusu olarak hizmet veren Node-Red uygulaması üzerine MQTT üzerinden mesaj gönderilerek ilgili çıktı görüntülenmiştir. Böylece teslimat test edilmiştir ve beklenen çıktılar elde edilmiştir.

4.Sonuç

Yapılan çalışma sonucunda Node-Red için başarılı bir CI/CD kurgusu Jenkins kullanılarak sağlanmıştır. Böylece giriş kısmında bahsedilen sorunlara Node-Red kullanan uygulamalar için kolay uygulanabilir bir DevOps yaklaşımı sunulmuştur. İlgili teslimat sürecinin kullanılması ile birlikte hatasız bir teslimat süreci sağlanabilmektedir. Bununla birlikte şayet hata alınırsa bu durum önceden fark edilerek anında müdahalede bulunulabilir.

Çalışmanın kullanım alanı çeşitlendirilebilmekle birlikte özellikle son kullanıcıya hitap eden akıllı ev teknolojileri veya otomobil teknolojilerinde Node-Red kullanılması durumda hızlı müdahale ile uzaktan güncelleme imkanı tanıyabilmektedir. Ayrıca üretim tesislerinde Node-Red ile geliştirilmiş otomasyonlarda yapılacak değişiklikleri hızlı ve hatasız bir şekilde gerçekleyebilmesi sayesinde minimum kayba sebebiyet verecektir. Bu avantajlarını göz önünde bulundurduğumuzda üretim tesisleri gibi yerleşik otomasyon çözümlerinde dinamik ve sürekli geliştirilebilir otomasyon teknolojilerinin önünün açılmasında yardımcı olacaktır.

Kaynakça

[1] A. Katal, V. Bajoria and S. Dahiya, "DevOps: Bridging the gap between Development and Operations," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2019, pp. 1-7, doi: 10.1109/ICCMC.2019.8819631.

[2] A. Bijwe and P. Shankar, "Challenges of Adopting DevOps Culture on the Internet of Things Applications - A Solution Model," 2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS), Tashkent, Uzbekistan, 2022, pp. 638-645, doi: 10.1109/ICTACS56270.2022.9988182.

[3] V. Singh, A. Singh, A. Aggarwal and S. Aggarwal, "DevOps based migration aspects from Legacy Version Control System to Advanced Distributed VCS for deploying Micro-services," 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, India, 2021, pp. 1-5, doi: 10.1109/CSITSS54238.2021.9683718.

[4] R. López-Viana, J. Díaz, V. H. Díaz and J. -F. Martínez, "Continuous Delivery of Customized SaaS Edge Applications in Highly Distributed IoT Systems," in IEEE Internet of Things Journal, vol. 7, no. 10, pp. 10189-10199, Oct. 2020, doi: 10.1109/JIOT.2020.3009633.

[5] A. R. Biswas and R. Giaffreda, "IoT and cloud convergence: Opportunities and challenges," 2014 IEEE World Forum on Internet of Things (WF-IoT), 2014, pp. 375-376, doi: 10.1109/WF-IoT.2014.6803194.

[6] L. Georgeta Guşeilă, D. -V. Bratu and S. -A. Moraru, "DevOps Transformation for Multi-Cloud IoT Applications," 2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI), Lisbon, Portugal, 2019, pp. 1-6, doi: 10.1109/ISSI47111.2019.9043730.

[7] L. G. Guşeilă, D. -V. Bratu and S. -A. Moraru, "Continuous Testing in the Development of IoT Applications," 2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI), Lisbon, Portugal, 2019, pp. 1-6, doi: 10.1109/ISSI47111.2019.9043692