GelİŞlet (DevOps) Yaklaşımında Konteyner Dönüşümü Deneyimi

Ahsen İkbal Aytekin[1], Yagup Macit[1]
[1]HAVELSAN Askeri Yazılımlar Mühendislik Grup Müdürlüğü
06510 Ankara, Türkiye

Özet. Bilişim sektöründe artan rekabet koşulları, yeni yazılım üretimi ve teslimatı aşamasında hız ve kaliteyi önceleyen bir bakış açısı ortaya çıkartmıştır. Çevik ve yalın üretim yaklaşımları ile hızlı değer üretimine odaklanan Bilişim sektörü için olgunlaşmış ürün dağıtımı önemini korumaktadır. Dağıtılan yazılım ürünü ile bu ürünün çalışacağı ortamın uyumu, tüm kurulum ve yapılandırma çabalarına karşın sürpriz sorunlara gebe olmaya devam etmektedir. Ürünün son kullanıcı için test edildiği ortam ile çalışacağı ortam için aynılığın sağlanması hız ve sağlamlık konusunda ortaya çıkabilecek sorunları mekân kaydırma etkisi ile çözümlemektedir. Bu etki için kullanılan Docker Konteyner teknolojisi ile son kullanıcının önünde canlı hizmet verecek sistemin geliştirme ortamında yapılandırması olanaklı hale gelmiştir. Üstelik bu teknoloji üzerinde çalışacak yazılım için herhangi bir mekân farkındalığı gerekmemektedir. Bu çalışmada, geliştirilen yazılımın fiziksel/sanal sunuculara dağıtım yaklaşımından Konteyner temelli dağıtım yaklaşımına geçiş, bakım ve kazanımlar anlatılmıştır.

Anahtar Kelimeler: Uygulama yaşam döngüsü yönetimi, Hızlı Uygulama Geliştirme, Yalın Geliştirme, Uygulama Değer Yönetimi, Yazılım Konumlandırma, Sürekli Dağıtım, Docker, Konteyner, DevOps, GelİŞlet

Container Transformation Experience in a Devops Approach

Ahsen İkbal Aytekin[1], Yagup Macit[1]
[1]HAVELSAN Askeri Yazılımlar Mühendislik Grup Müdürlüğü
06510 Ankara, Türkiye

Abstract. Due to the increasing competition conditions in the informatics sector, the new software production and delivery phase has revealed a prioritizing perspective on speed and quality. With its agile and lean production approaches, it maintains the importance of a mature product distribution for IT sector focusing on fast value production. The compatibility of the distributed software Product and the environment in which this product will run continues to create surprising problems despite all installation and configuration efforts. The environment in which the product is tested for the end user is the same medium, it resolves problems that may arise in speed and robustness with the effect of environment displacement. With Docker Container technology used for this effect, it has become possible to configure the system in the development environment that will serve live in front of the end user. Moreover, no environment awareness is required for the software to work on this technology. In this study, migration,

maintenance and gains from the approach of distribution to the physical/virtual servers of the developed software to Container based distribution approaches are explained.

Keywords: Application Lifecycle Management, Rapid Application Development, Lean Development, Application Value Management, Software Positioning, Continuous Delivery, Docker, Container, DevOps

1. Giriş

Her geçen gün gelişen bilişim sektöründe, hızlı ve kararlı ürün teslimatı yapabilme yetisi, rekabet gücünü destekleyen en önemli unsurlardan biri olarak ortaya çıkmaktadır. Hızlı ve kararlı ürün bir teslimatı yapabilmek için analiz, geliştirilme, derleme, sınama ve dağıtım çabalarında kaliteyi öne çıkartan ve zamanı etkin kullanan bir yaklaşım önem kazanmaktadır. Bir yazılım ürününün, teslimat sonrasında kullanıcılar ve işletmen ile geliştireceği etkileşim, üretici firmanın ticari geleceği için önem taşımaktadır. Kullanıcı etkileşimi, söz konusu ürünün sonraki sürümleri ile diğer tamamlayıcı veya çapraz ürünlerin, ticari başarını etkileyen temel faktör olarak öne çıkmaktadır. Ticari gelecek açısından bakıldığında, kullanıcı deneyimin gerçekleştiği canlı ortam, üreticinin haberdar olması gereken bir ortam haline gelmektedir.

Ürünün sağlıklı şekilde işletilebilmesi ve izlenmesi, sınama süreçlerine tabi tutulduğu ortam ile canlı kullanıma verildiği ortamın aynı veya benzer olmasını gerektirmektedir. Benzer ortam ihtiyacı sanal sunucular ile karşılanmaya çalışılmasına rağmen, aynı şablondan türetilmiş sanal sunucuların, yazılım envanteri ve ortam bilgisinde zamanla ortaya çıkan farklılıklar sapmalara neden olmaktadır. Bu sapmaların ortadan kaldırılması için yazılım ürününün kendi yaşam sitemi içerisinde dağıtımını önemli hale gelmektedir.

HAVELSAN Kalite Modülü (KM) uygulamasının sunulduğu canlı ortam sunucusu ile sınama ortamı sunucusu arasında ağ, güvenlik ve yapılandırma ayarları farklılıklar barındırabilmektedir. Bu farklılıklar, sınama ortamında yapılan doğrulamaların canlı ortamda geçerli olmasını sorgulanabilir hale getirmektedir. Canlı ortam ve sınama ortamını aynılaştırarak, tespit edilemeyen hataların en aza indirgenmesi, bu çalışma için temel motivasyonu oluşturmaktadır.

Bildirinin ikinci bölümünde, yazılım üretimi hakkında bilgi verilmiştir. Üçüncü bölümde, HAVELSAN'ın uygulama geliştirme deneyimi, KM uygulaması paylaşılmış, bu uygulamanın dönüşüm öncesi ve sonrası dağıtım yöntemi ve elde edilen sonuçlar anlatılmıştır. Son bölümde ise gerçekleştirilen deneyim genel olarak özetlenerek, yapılan değerlendirmeler aktarılmıştır.

2. Yazılım Üretimi

Bilişim sektörünün gelişimi ile birlikte, 1970'li yıllarda yazılım üretimleri için standardize edilmiş ve işletilebilecek üretim modeli isteklerine, Şelale Süreç Modeli [1] ile yanıt verilmiştir. Bu yanıt, 2000'li yıllarda, yapılan eklemeler ile tüm yazılım ve işletim evrelerini kapsayacak şekilde, Uygulama Yaşam Döngüsü Yönetimi (UYY) [2] olarak evrilmiştir. Uygulama Yaşam Döngüsü Yönetimi kapsamına işletim, izleme ve kullanıcı dönütlerinin eklenmesi sonucunda, uygulama yaşam döngüsü yönetimi ve kültürü, 2010'lu yıllarda, GelİŞlet (DevOps) [3] adıyla tanımlanmıştır.

Endüstriyel üretim hatlarında, üst-üste binen üretim fonksiyonlarının geleneksel sıralı yaklaşımla ardışık olarak ele alınması eş zamanlı etkinlikler için etkileşim sorunu oluşturmuştur. Bu sorun, 1986 yılında Hirotaka Takeuchi ve Ikujiro Nonaka tarafından Fuji-Xerox ve Honda örneğinde, Amerikan futbolu (Rugby) oyunundan esinlenilerek yeni geliştirme oyunu [4] ile çözülmüştür.

Bilişim sektöründe, yinelemeli ve artımlı geliştirme konusundaki deneyimlerin birikimi sonucunda 2001 yılında Çevik Bildiri (Agile Manifesto) [5] yayınlanmıştır. Çevik bildiri ile bireyler ve etkileşim, çalışan yazılım, müşteri ile işbirliği ve değişime açıklık öne çıkan ana ilkeler olarak kabul edilmiştir. Ken Schwaber ve Jeff Sutherland, Çevik deneyim ile endüstriyel yeni geliştirme oyununu birleştirerek Scrum [6] çerçevesini tanımlamıştır.

Bilişim uygulamaları için çevik yaklaşımlar kullanılmaya başlandığında özellikle büyük organizasyonlar için ortaya çıkan uyarlama sorunlarına yalın (lean) geliştirme [7] yaklaşımıyla yanıt verilmiştir. Bu yaklaşım ile gereksiz ve zamansız olan her türlü üretim ve eylem sorgulanarak çöpe giden çabalar azaltılmış ve işlemler basite indirgenmiştir.

GelİŞlet kültürü ile yazılım üretiminde etkinliğin sağlanması ve ortaya çıkan ürün değerinin korunması için, Linux LXC [8] deneyiminden faydalanılarak Docker Konteyner [9] çözümü geliştirilmiştir. Docker konteyner ile geliştirme ekibinin yazılım testi için konumlandırdığı ortamı doğrudan kullanıcın önüne servis etme olanağı sağlanmıştır. Aynı şekilde, kullanıcının önünde sorun çıkaran konumlandırmayı, doğrudan geliştirme ekibinin incelemeye almasına olanak sağlanmıştır. Konteynerler, konumlandıkları fiziksel makinanın çekirdeğini kullandıkları ve üzerlerinde ayrıca bir işletim sistemi barındırmadıkları için fiziksel makinaya yakın performans göstermektedirler.

3. HAVELSAN Deneyimi

HAVELSAN, 1200 üzerinde çalışanı olan ve 4 genel müdür yardımcılığından oluşan bir savunma sanayi şirketidir. Her bir genel müdür yardımcılığı birbirlerinden oldukça farklı alanlarda çalışmalar yürüten ve her müdürlüğe bağlı olarak değişen büyüklüklerde takım ve takımların bağlı olduğu gruplardan oluşmaktadır. Böylesi farklı alanlara çözümler sunan bir şirkette, her projenin ihtiyacı birbirinden oldukça farklı olarak ortaya çıkabilmektedir. Projelerin farklı ihtiyaçlarına hızlı bir şekilde cevap verebilmek amacıyla HAVELSAN'da, kurumsal UYY altyapısı [10] devreye alınmış ve GelİŞlet yaklaşımıyla yürütülmektedir.

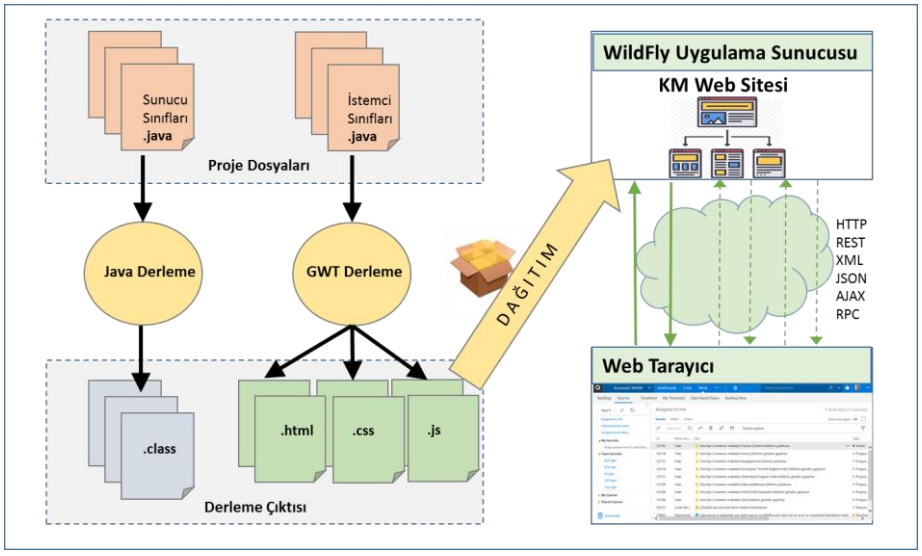
HAVELSAN uygulama geliştirme süreçlerinde belirli standartlara bağlı olarak çalışan kurumsal bir şirket olarak, bu standartlara uyumu kontrol etmek amacıyla Kalite Modülü 'nü (KM) konumlandırmış ve UYY altyapısının bir parçası olarak işletilmektedir. KM, hem geliştirme süreçlerinin hem de kurumsal süreçlerin ihtiyaçlarına cevap verebilmek için, farklı proje ihtiyaçlarına ve değişen kurumsal süreçlere göre güncellenen dinamik bir yapıya sahiptir. Bu yapıya cevap verebilmek için çevik yöntemlerle geliştirilen KM uygulaması için GelİŞlet yaklaşımına geçilmiştir.

Yardım Masası üzerinden gelen talepler ve kurumsal süreçlerden gelen girdiler planlanarak 2'şer haftalık iterasyonlarda çalışılmakta ve tur sonunda elde edilen çıktılar sırasıyla test ve üretim ortamlarına aktarılmaktadır. Uygulamanın izlenmesi ile elde

edilen bilgiler, yapılan geri dönüşler ve süreç değişiklikleri tekrar girdi olarak alınarak bu döngü devam etmektedir. Uygulamada yapılan değişikliklerin mümkün olan en hızlı ve sağlıklı şekilde üretim ortamına aktarılabilmesi, test ortamında koşulan testlerin üretim ortamı için de başarılı olmasına bağlıdır. Bu nedenle test ve üretim ortamlarının birbirlerine eş altyapıda sunulması önem taşımaktadır. Bu altyapı, sanal makinalar üzerinde, aynı donanım özelliklerine sahip, aynı işletim sistemi ve uygulama sunucusu ayarları ile birden fazla ortam ayağa kaldırılarak sağlanmaya çalışılsa da test ortamı ile üretim ortamının uygulama özelinde farklı tepkiler verebildiği gözlemlenmiştir. Bu farklılığı en aza indirgeyebilmek ve test koşularını bire bir olarak tekrarlayabilmek için konteyner altyapısına geçilmesi planlanmıştır. KM uygulamasının dağıtımları, oluşturtulan eş Docker konteynerler üzerine yapılması ile elde edilen sonuçlar tartışılmıştır.

3.1. Örnek Durum Açıklaması

Yazılım ve sistem geliştirme ile ilgili kurumsal süreçlerin ve CMMI kayıtlarının her noktadan erişilebilir olması bütün kurumlar için olduğu gibi HAVELSAN için de önem taşımaktadır. Söz konusu standardizasyon için Kalite Modülü (KM) uygulaması konumlandırılmıştır. KM Uygulaması, istemci-sunucu mimari yaklaşımı kullanılarak Web tabanlı geliştirilmiştir. KM Uygulaması, sunucu tarafında WildFly [11] Java Uygulama Sunucusu, istemci tarafında ise Google Web Toolkit (GWT) [12] aracılığıyla Web tarayıcı, üzerinde çalışmaktadır. Uygulamanın sunucu tarafı ve istemci tarafının derlenme mekanizması ile çalışma zamanı için konumlandırılmasına ilişkin mimari Şekil 1'de görülmektedir.

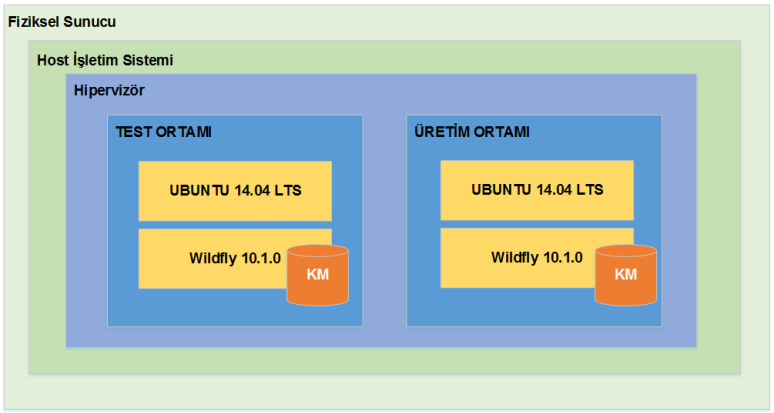


Şekil 1. KM İnşa ve Konumlandırma Mimarisi

Uygulama için hem Java hem de GWT derlemesi yapılmaktadır. Java derlemesi ile sunucu tarafında çalışacak olan kitaplıklar elde edilmektedir. GWT derlemesi ile ajax desteği üzerinden çalışacak olan html ve js dosyalarından oluşan web sitesi varlıkları edilmektedir. Elde edilen sunucu kitaplıkları ve istemci varlıkları WildFly uygulama sunucusuna KM Web Sitesi olarak konumlandırılmaktadır.

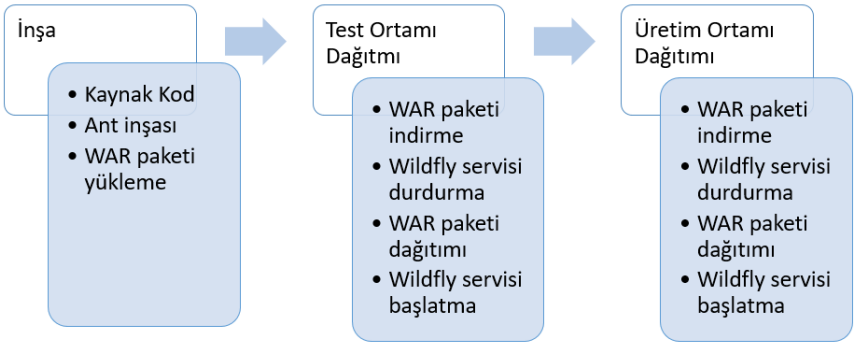
3.2. Geleneksel Dağıtım Hattı

KM uygulamasının dağıtımı, GelİŞlet döngüsü ile her tur sonunda elle tetiklenen otomatize edilmiş süreç adımları ile gerçekleştirilmektedir. Bu adımlar Şekil 3'te gösterilmiştir. KM uygulaması, kurumsal bilişim altyapısı sanallaştırma hizmeti olarak alınan Ubuntu 14.04 LTS işletim sisteminde Wildfly 10.1.0 uygulama sunucusu üzerinde konumlandırılmıştır.



Şekil 2. Sanal Makina Mimarisi

GelİşletDöngüsüne göre tur sonunda kod deposuna gönderilmiş olan kaynaklar etiketlenerek inşa edilmektedir. Apache-Ant ile java ve GWT üzerinden derlenen kodlar, WAR paketi üretmekte ve bu paket çıktılar dizinine yüklenmektedir. Başarılı olan inşa adımı sonrasında üretilen WAR paketinin dağıtımı test ortamı için tetiklenerek daha önce üretim ortamına benzer konfigürasyonla kurulumları yapılmış sanal makine üzerindeki Wildfly uygulama sunucusuna yüklenmektedir.



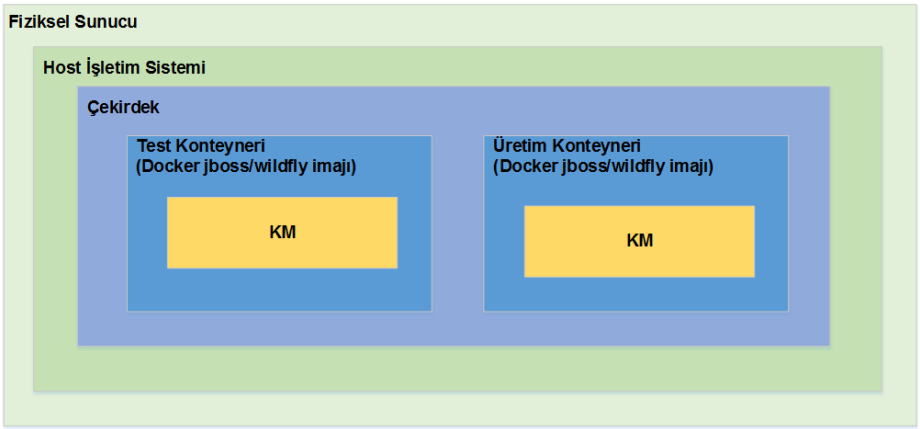
Şekil 3. Mevcut dağıtım adımları

Test ortamı için hazır olan paketin dağıtımı ortalama 2,1 dakika sürmektedir. Test ortamındaki test koşularını başarıyla geçen aday sürüm bu kez üretim ortamı için tetiklenmekte ve dağıtılmaktadır.

Dağıtım ortamlarının kurulumları benzer konfigürasyonla gerçekleştirildiğinden, sürüm dağıtım testleri her iki ortam içinde standart olmakla birlikte donanımsal ya da ağ ve güvenlik temelli farklılıklar kaçınılmaz olduğundan testler her iki ortam için aynı sonuçları üretmeyebilmektedir. Bu da canlıya çıkmaya aday sürümün hatalarını ancak üretim ortamına aktarılmasıyla tespit edilebilmesi anlamına gelmektedir ki bu ölçekte kullanıcıya hizmet veren bir istem için kabul edilebilir değildir.

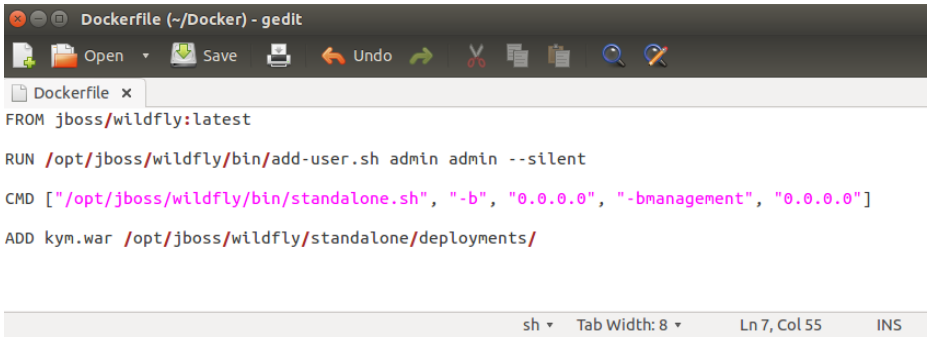
3.3. Konteyner Temelli Dağıtım Hattı

KM uygulamasının geleneksel olarak tanımlamış dağıtım hattı yerine deneyimlemek istenilen konteyner temelli dağıtım hattına aktarımı için docker platformu kullanılmıştır. Docker, sanal sunucu üzerindeki işletim sistemini ve çekirdeğini kullanan ayrı konteynerler sunmaktadır.



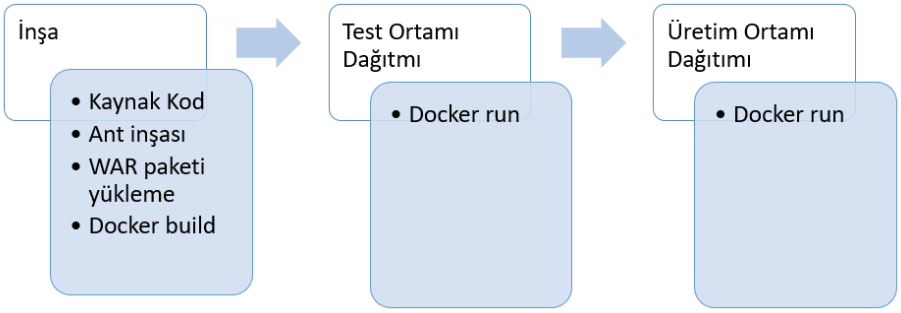
Şekil 4. Konteyner Mimarisi

KM uygulamasını docker konteyner üzerine dağıtabilmek için ilk olarak inşa adımlarına docker yapılandırması eklenmiştir. Bu yapılandırma ile Dockerfile içerisinde verilen konfigürasyona göre bir imaj oluşturularak dağıtım yapılacak ortam, dağıtım yapılacak sürüm ile birlikte hazırlanmaktadır.



Şekil 5. Dockerfile içeriği

Başarılı inşa sonucunda, sürüm adayı olan WAR paketi dağıtımı yerine yalnızca hazırlanmış olan dağıtım ortamını ayağa kaldırmak gerekmektedir. Test ortamı için çalıştırılan docker, yeni bir konteyner numarası alarak ayağa kalmaktadır. Test koşuları bu platform üzerinde gerçekleştirildikten sonra, başarılı sürüm için üretim ortamı konteyneri oluşturulmaktadır.



Şekil 6. Konteyner temelli dağıtım adımları

Uygulamanın dağıtımı için aday ortamların ayağa kaldırılması ortalama 15 saniye sürmektedir. Bu aday ortamlar aynı docker imajı üzerinden üretilmekte olduğundan, olası farklılıkların önüne geçilerek uygulananın her ortanda aynı davranışı göstereceğini garanti etmektedir. Ortam kurulumlarının aynı docker imajı üzerinden gerçekleştirilmesi, istenilen sayıda test ortamının ayağa kaldırılabilmesi açısından da fayda sağlamaktadır. Ayrıca farklı imajlar üzerine dağıtım denemeleri yapılarak, optimum ortam gereksinimlerinin derlenmesine olanak sağlamaktadır.

3.4. Değerlendirme ve Beklenen Fayda

KM Web sitesi için geliştirilen sunucu kitaplıklarının ve istemci varlıklarının derleme ve dağıtımı hem sanal sunucular için hem de Docker Konteyner için ayrı, ayrı çalışılmıştır.

Derleme aşamasında, Java ve GWT derlemesine ilişkin adımlar her iki durumda da aynı şekilde korunmuş ve herhangi bir değişiklik yapılmamıştır. Docker dağıtımı için ek bir Docker Build adımı eklenmiştir.

Dağıtım aşamasında, sanal sunucular üzerine yapılan aşamalı dağıtımda her sunucu için servislerin durdurulması, ön bellek temizleme, yükleme ve çalıştırma gibi aşamalı adımların bakım ve işletimi gerekirken, derlenen Docker imajı için dağıtım işlemini kalkmış ve sadece çalıştırma adımına indirgenmiştir.

Yazılım envanteri açısından, sanal sunucuların her birinde çalışacak uygulama için gereken yazılımların bakımı ve eşitlemesini yapmak gerekirken, derlenen Docker imajı ile yazılımların bakımı ve eşitlemesinin bakımını ortadan kalkmıştır.

Test işlemleri açısından, her bir aşamanın sanal sunucusu üzerinde ayrı yapılan testlerde faklı sonuçlar alınırken, Docker imajı ile yapılan testlerde barındırma ortamı etken olmaktan çıkmıştır.

Hata detaylandırma açısından, sanal sunucular üzerinde dağıtılan uygulamada ortaya çıkan hataların ortama bağlı analizi, Docker imaj ile ortadan kalkmıştır.

Konteyner dönüşümü ile yukardaki başlıklarda sağlanacak sayısal etkinlik değerlerinin önümüzdeki dönemlerde elde edilmesi beklenmektedir.

4. Sonuç

Bu çalışmada, geleneksel dağıtım hattının Docker temelli konteyner dönüşümünün nasıl gerçekleştirildiği aktarılmıştır.

HAVELSAN Kalite Modülü, farklı büyüklüklerdeki projeler için kurumsal süreç standartlarının sağlaması için geliştirilmiş bir uygulama olarak hizmet vermektedir. Farklı proje ihtiyaçlarından doğan geliştirme talepleri ve değişen kurumsal süreçlere uyum sağlanması amacıyla, KM uygulaması için GelİŞlet yaklaşımı benimsenmiştir. Bu yaklaşımla, kullanıcılardan gelen girdiler ve kurumsal süreçlerde gerçekleştirilen düzenlemeler, uygulama ihtiyacı olarak planlanmakta ve tur sonunda kullanıma verilmek üzere geliştirilmektedir. Uygulamanın, ihtiyaçlara zamanında cevap verebilmesi için test ortamı ile üretim ortamına ait dağıtım adımlarının standart olması gerektiği öngörülmüştür. Dağıtım sürecinin sanal sunucular kullanılarak yürütülmesi, test ve üretim ortamları arasında, donanım, ağ, güvenlik ve yapılandırma ayarlarında zamana dayalı farklılıklar oluşabileceği riskini ortaya çıkarmıştır. Bu ortamların, farklılıklar içermeyen şekilde hizmet verebilmesi ve üretim ortamı için öngörülemeyen hataların en aza indirgenmesi amacıyla Docker konteyner dönüşümü gerçekleştirilmiştir.

Bu dönüşümle, aynı Docker imajından oluşturulan konteynerler üzerine dağıtılan uygulamanın, her ortamdaki davranışının standart olması sağlanmıştır. Aynı zamanda dağıtım süresinin hızlanması, ortamların istenilen sayıda çoğaltılarak farklı test konteyner ortamlarının elde edilmesi de kolaylaştırılmıştır.

Konteyner dönüşümü, KM uygulaması için en uygun uygulama bağımlılıklarını ve en yüksek performansı vereceği ortamı test etmek oldukça kolay bir duruma

gelmiştir. Her yeni test ortamı için yeni sunucu kurulumlarının ve konfigürasyon yapılandırmalarının önüne geçilmesi sağlanarak işletme maliyeti de bu sayede azaltılmıştır. Konteyner altyapısı ile uygulamanın ihtiyaç duyduğu ortamların hızlı bir şekilde ayağa kaldırması ve ihtiyaç olmayan ortamların gereksiz kaynak kullanımının önüne geçilmesi sağlanmıştır.

Teşekkür. Yazarlar, HAVELSAN yönetimine çalışmaya verdiği destek için teşekkürler ederler.

Referanslar

1. W.W.Royce, «Managing the Development of Large Software Systems,» *Proceedings of IEEE WESCON,* pp. 328-338, 1970.
2. D. Chappell, «What is Application Lifecycle Management?,» Chappell & Associates, 2008.
3. «Emerging Technology Analysis: DevOps a Culture Shift, Not a Technology,» Gartner, 2013.
4. I.Nonaka ve H.Takeuchi, «The new new product development game,» *Harvard business review,* p. 137–147, January-February, 1986..
5. «Agile Manifesto,» [Çevrimiçi]. Available: http://www.agilemanifesto.org/. [Erişildi: 17 Haziran 2016].
6. Ken Schwaber, Jeff Sutherland, «The Scrum Guide,» Kasım 2017. [Çevrimiçi]. Available: http://www.scrumguides.org/. [Erişildi: 17 Haziran 2018].
7. T. P. Mary Poppendieck, Lean Software Development: An Agile Toolkit, Addison-Wesley Professional, 2003.
8. «Linux Containers,» [Çevrimiçi]. Available: https://linuxcontainers.org/lxc/introduction/. [Erişildi: 17 06 2018].
9. D. Merkel, «Docker: lightweight Linux containers for consistent development and deployment,» *Linux Journal,* cilt 2014, no. 239, 2014.
10. Y.Macit, E.Tüzün, K.Ince, A.I.Aytekin, «Büyük Ölçekli Bir Organizasyonda Uygulama Yaşam Döngüsü Yönetimi Uygulama Deneyimi,» %1 içinde *Proceedings of the 8th Turkish National Software Engineering Symposium*, 2014.
11. «Wildfly Web Sitesi,» 17 Haziran 2018. [Çevrimiçi]. Available: http://www.wildfly.org/.
12. «Google Web Toolkit,» 17 Haziran 2018. [Çevrimiçi]. Available: http://www.gwtproject.org.