On İkinci Bölüm
Django Projenizi Dağıtma
Dağıtıma Giriş

Dağıtım, bir uygulamanın üretim ortamında son kullanıcılara sunulduğu yazılım geliştirme yaşam döngüsünün son aşamasını işaret eder. Bu kritik adım, tasarım, kodlama ve test dahil olmak üzere geliştirmenin ilk aşamalarını takip eder ve yazılımın genel kullanıma hazır olmasını sağlar. Etkili dağıtım metodolojileri, kesinti süresini en aza indirirken sorunsuz uygulama performansı ve kullanıcı memnuniyetini sağlamayı amaçlar. Bu giriş, yazılımı verimli ve etkili bir şekilde dağıtmak için temel dağıtım kavramlarını, tekniklerini ve en iyi uygulamaları inceler.

Dağıtımın Temel Kavramları

1. Sürekli Entegrasyon (CI): Bu süreç, tüm geliştiricilerin yaptığı çalıştırma ana veriyi sık sık birleştirmeyi ve birleştirmeyi erken tespit etmek için otomatik testlerin çalıştırılmasını içerir. Sürekli Entegrasyon (CI), üretim ortamına geçiş daha sorunsuz hale gelir.

2. Sürekli Teslimat (CD): CI prensiplerine dayanan CD, yazılımın her an güvenilir bir şekilde dağıtılabilmesini sağlayarak sürüm sürecini otomatikleştirir. Bu, kullanıcılardan gelen geri bildirim döngüsünün hızlanmasına ve güncellemelerin düzenlenmesine yardımcı olur.

3. Kod Olarak Altyapı (IaC): IaC, fiziksel ve sanal cihazları programatik olarak yönetir, manuel süreçleri ortadan kaldırır ve otomasyon yoluyla ortamlarda tutarlılığı artırır.

4. Dağıtım Stratejileri: Uygulamanın ihtiyaçlarına göre riskleri en aza indirmek ve yüksek erişilebilirliği garanti altına almak için mavi-yeşil dağıtım, kanarya sürümleri ve yuvarlanan güncellemeler gibi çeşitli stratejiler kullanılır.

Dağıtım Yöntemleri

1. Manuel Dağıtım:

Geliştiricilerin üretim ortamında kodu manuel olarak aktarmasını ve yapılandırmasını içerir. Daha az verimli olmasına rağmen, bazen daha küçük projelerde veya daha büyük projelerin ilk aşamalarında kullanılır.

2. Otomatik Dağıtım: Kodu aktarmak ve dağıtmak için betikleri veya otomatik araçları kullanır; bu da doğruluğu artırır, hataları azaltır ve verimliliği iyileştirir.

3. Hizmet Olarak Platform (PaaS): Heroku, Google App Engine ve AWS Elastic Beanstalk gibi PaaS sağlayıcıları, altyapıyı yöneterek dağıtımı basitleştirir ve böylece geliştiricilerin uygulamanın kendisine konsantre olmasını sağlar.

Etkili Dağıtım İçin En İyi Uygulamalar

1. Sürüm Kontrol Yönetimi:

Tüm dağıtılabilir kodu Git gibi bir sürüm kontrol sisteminde tutun. Bu uygulama değişiklikleri yönetmeye, farklı geliştirme dallarını sürdürmeye ve kolay geri almaları kolaylaştırmaya yardımcı olur.

2. Sıkı Test: Sorunların üretime ulaşmadan önce yakalanması için birim testleri, entegrasyon testleri ve kullanıcı kabul testleri dahil olmak üzere kapsamlı ön dağıtım testleri sağlayın.

3. Proaktif İzleme ve Günlük Kaydı: Dağıtımdan sonra, uygulamanın performansını izlemek ve ortaya çıkan sorunları hızla ele almak için izleme araçlarına ve günlük kaydına sahip olmak hayati önem taşır.

4. Dokümantasyon ve Eğitim: Dağıtım sürecinin kapsamlı dokümantasyonunu tutun ve ekip üyelerine olası geri dönüşlerin nasıl yönetileceği de dahil olmak üzere bu prosedürler hakkında eğitim verin.

5. Güvenliğe Önem Verin: Dağıtım sürecinin tamamında güvenliği ön planda tutun; boru hattının güvenliğini sağlamaktan tüm bileşenlerin düzenli olarak güvenlik yamalarıyla güncellenmesini sağlamaya kadar.

Örnek: Git ile Otomatik Dağıtımı Ayarlama

Git kullanarak basit bir otomatik dağıtımı nasıl yapılandırabileceğinizi aşağıda bulabilirsiniz:

```
# Clone the repository on the production server
git clone https://github.com/example/myapplication.git /var/www/myapplication

# Set up a Git hook for automatic deployment
cat > /var/www/myapplication/.git/hooks/post-receive <<EOF
#!/bin/sh
git --work-tree=/var/www/myapplication --git-dir=/var/www/myapplication/.git checkout -f
EOF
chmod +x /var/www/myapplication/.git/hooks/post-receive

# Developers push to production, triggering automatic updates
git push production master
```

Bu betik, üretim dalına değişiklikler gönderildiğinde üretim sunucusunu güncelleyerek dağıtım sürecini otomatikleştirir.

Çözüm

Stratejik dağıtım uygulamalarını anlamak ve uygulamak, yazılım uygulamalarının başarılı bir şekilde başlatılması ve işletilmesini sağlamak için kritik öneme sahiptir. Dağıtım süreçlerinde ustalaşmak, kuruluşların güncellemeleri daha verimli bir şekilde sunmasını, istikrarı korumasını ve kullanıcı ihtiyaçlarına ve pazar taleplerine hızla uyum sağlamasını sağlayarak rekabet avantajı sağlar ve kullanıcı deneyimini geliştirir.

Üretim Ortamına Dağıtım

Yazılımı üretim ortamına sürmek, yazılım geliştirme yaşam döngüsünde önemli bir geçişi ifade eder. Bu aşama, uygulamayı geliştirme ve test aşamalarından, son kullanıcılar için erişilebilir olan canlı bir operasyonel ortama taşır. Böyle bir dağıtım, uygulamanın hedeflenen operasyonel ortamda hazır olduğunun ve optimal performans sergilediğinin doğrulanmasını gerektirir; bu, kapsamlı testler ve geliştirme iyileştirmeleri sonrasında yapılmalıdır. Bu tartışma, üretim ortamına başarılı bir yazılım dağıtımı için kritik olan temel faktörleri, yöntemleri ve en iyi uygulamaları özetlemektedir.

Üretime Dağıtımın Temelleri Üretime dağıtım, yalnızca kodu aktarmaktan daha fazlasını içerir; üretim sunucularını yapılandırmak, veritabanlarını kurmak, güvenli bağlantılar
oluşturmak ve tüm gerekli bağımlılıkların ve hizmetlerin optimal işlevsellik için doğru şekilde düzenlendiğinden emin olmak gibi kapsamlı ayarlamaları da kapsar.

Önemli Dağıtım Hususları

1.Ortam Yapılandırması: Üretim ortamının, farklı ayarlar veya yapılandırmalar nedeniyle beklenmedik davranışları önlemek için staging ortamına yakın olması çok önemlidir.

2.Veri Yönetimi: Verimli göç stratejileri, şema güncellemeleri ve ilk veri tohumlama dahil olmak üzere doğru veri yönetimi, sorunsuz bir geçiş için gereklidir.

3.Güvenlik Önlemleri: Güvenlik en önemli öncelik olmalıdır; bu, güvenli iletişim protokollerini, veri şifrelemeyi ve veri koruma düzenlemelerine sıkı sıkıya uyulmasını kapsar.

4.Ölçeklenebilirlik ve Performans: Beklenen ve beklenmedik kullanıcı yüklerine hazırlıklı olmak için etkili kaynak yönetimi ve yük dengeleme, performans standartlarını sürdürmek için kritik öneme sahiptir.

5.Yedekleme ve Kurtarma Önlemleri: Güvenilir yedekleme çözümleri ve ayrıntılı felaket kurtarma planları oluşturmak, sürekliliği sağlamak ve veri bütünlüğünü korumak için önemlidir.

Dağıtım Yöntemleri Riskleri ve kesinti sürelerini en aza indirmek için çeşitli dağıtım stratejileri kullanılabilir:

1.Blue-Green Dağıtımı: Bu yöntem, trafik bir taraftan diğerine kesintisiz bir şekilde kaydırılabilen iki kimlikli üretim ortamı içerir, böylece kesinti olmadan test yapılabilir.

2.Canary Yayınları: Performansı değerlendirmek ve daha geniş bir yayına geçmeden önce ayarlamalar yapmak için önce küçük bir kullanıcı grubuna kademeli dağıtım yapılır.

3.Rolling Güncellemeler: Bileşenleri kademeli olarak güncelleyerek sistem kararlılığını sürdürmeye ve kaynak aşırı yüklenmesini en aza indirmeye yardımcı olur.

Dağıtım Araçları

Dağıtım Araçları Verimli araçlar ve teknolojiler, dağıtım süreçlerini kolaylaştırmak için vazgeçilmezdir

- **Jenkins:** Sürekli entegrasyon ve teslimat konusunda yardımcı olan bir otomasyon aracıdır, tutarlı derlemeler ve testler aracılığıyla dağıtımların güvenilirliğini artırır.
- **Docker:** Uygulamayı ve ortamını kapsülleyen konteyner teknolojisini kullanarak, geliştirme ve üretim arasında tutarlılık sağlar.
- **Kubernetes:** Kümeler arasında konteynerleştirilmiş uygulamaları yönetir, uygulamaların dağıtımı ve ölçeklenmesi için otomasyon yetenekleri sunar.
- **Ansible:** Yapılandırma yönetimini, uygulama dağıtımını ve görev otomasyonunu basitleştirir, dağıtımlar arasında tutarlı ortam kurulumunu sağlar.

Örnek: Docker ve Jenkins ile Dağıtım Uygulaması

Docker ve Jenkins kullanarak otomatik bir dağıtım kurulumunun pratik bir örneği şunları içerir:

1.Dockerfile Yapılandırma:

```
# Choose the base image
FROM python:3.8

# Set the directory for commands to run
WORKDIR /app

# Copy the application source into the container
COPY . /app

# Install any necessary packages
RUN pip install -r requirements.txt

# Make the application's port available outside the container
EXPOSE 80

# Specify the command to run the application
CMD ["python", "app.py"]
```

2. Jenkins Dağıtım Pipeline'ı Oluşturma:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'docker build -t myapp .'
            }
        }
        stage('Test') {
            steps {
                sh 'docker run myapp python -m unittest'
            }
        }
        stage('Deploy') {
            steps {
                sh 'docker push myapp'
            }
        }
    }
}
```

Sonuç

Üretim ortamına etkili bir dağıtım, yazılımın kullanıcı ihtiyaçlarını karşılaması ve gerçek dünya koşullarında güvenilir bir şekilde çalışması için temel bir unsurdur. Stratejik dağıtım tekniklerini kullanmak, güvenlik protokollerini güçlendirmek ve gelişmiş otomasyon araçlarından yararlanmak, sorunsuz ve başarılı yazılım sürümleri sağlamak için önemlidir. Bu uygulamalar, yazılımın kalitesini ve güvenilirliğini artırır, kullanıcı memnuniyetini iyileştirir ve organizasyonel hedeflerin ilerlemesini sağlar.

Heroku, AWS ve DigitalOcean Gibi Hizmetler Kullanmak

Yazılımı üretim ortamına geçirme, yazılım geliştirme yaşam döngüsünde önemli bir aşama olup, uygulamaların geliştirme veya test aşamalarından canlı operasyonel bir ortama, kullanıcılara erişilebilir hale gelmesini sağlar. Bu aşama, uygulamanın gerçek dünya işletim koşullarında düzgün çalıştığını doğrulamak için önemlidir ve kapsamlı geliştirme ve test süreçlerini takip eder. Bu makale, uygulamaların etkin bir şekilde dağıtılmasını ve yönetilmesini sağlayan Heroku, AWS ve DigitalOcean gibi önde gelen bulut platformlarının sunduğu hizmetleri incelemektedir.

Önde Gelen Bulut Hizmetleri ile Dağıtım

Bir uygulamayı üretim ortamına dağıtmak, yalnızca kod yüklemekten daha fazlasını içerir; sunucuları yapılandırmak, veritabanlarını kurmak, güvenli bağlantıları sağlamak ve tüm gerekli hizmetleri ve bağımlılıkları yönetmek gibi kapsamlı ayarlamaları da içerir. Bu, uygulamanın en iyi şekilde çalışabilmesi için gereklidir.

Heroku: Kolaylaştırılmış Uygulama Dağıtımı Heroku, kullanıcı dostu yaklaşımı ve karmaşık altyapı yönetimini basitleştirme yeteneğiyle tanınır. Özellikle hızlı dağıtımlar ve sunucu tarafı işlemlerinin minimum düzeyde yönetilmesini gerektiren uygulamalar için tercih edilir.

Ana Özellikler:

- **Buildpacks:** Heroku, bu özel betikleri kullanarak, çeşitli programlama ortamlarında kaynak koddan uygulamaları otomatik olarak oluşturur.
- **Eklentiler:** Veritabanı yönetiminden mesajlaşma ve performans izlemeye kadar her şeyi kapsayan, uygulama işlevselliğini artıran bir dizi araç ve hizmet sunar.
- **Heroku CLI:** Bu komut satırı aracı, geliştiricilerin Heroku uygulamalarını doğrudan terminal üzerinden yönetmelerine olanak tanır.

Heroku'da Dağıtım Örneği:
İşte Heroku CLI ve Git kullanarak bir uygulamanın Heroku'ya nasıl dağıtılacağı:

```
# Log in to Heroku
heroku login

# Create a new application on Heroku
heroku create

# Link your Git repository to the Heroku remote
git remote add heroku <heroku_git_url>

# Deploy your application
git push heroku master

# Ensure the application is running
heroku ps:scale web=1

# Open the application in a browser
heroku open
```

AWS: Kapsamlı Bulut Çözümleri Amazon Web Services (AWS), küçük ölçekli projelerden büyük kurumsal sistemlere kadar geniş bir uygulama yelpazesi için altyapı hizmetleri sunar ve esnek, ölçeklenebilir çözümler sağlar.

Ana Özellikler:

- **EC2 (Elastic Compute Cloud):** Özelleştirilebilir sanal sunucular sunar.
- **Elastic Beanstalk:** Web uygulamalarını dağıtmak ve ölçeklendirmek için kullanıcı dostu bir hizmettir, sunucu yönetiminin birçok karmaşıklığını yönetir.
- **S3 (Simple Storage Service):** Veri yedeklemesi, toplama ve analiz için ölçeklenebilir nesne depolama sağlar.

AWS Elastic Beanstalk Üzerinde Dağıtım Örneği:
AWS Elastic Beanstalk ile bir uygulama dağıtmak, Elastic Beanstalk Komut Satırı Arayüzü (CLI) ile birkaç adım gerektirir.

```
# Install the EB CLI
pip install awsebcli

# Initialize your Elastic Beanstalk application
eb init -p python-3.7 my-application --region your-region

# Create and deploy your application
eb create my-env

# Open the deployed application
eb open
```

DigitalOcean: Geliştirici Dostu Bulut Barındırma DigitalOcean, basit ve maliyet etkin bulut barındırma çözümleri sunar, bu da onu hızlı kurulumlar ve yönetilebilir ölçekler isteyen girişimler ve geliştiriciler arasında popüler kılar.

Ana Özellikler:

- **Droplets**: Web siteleri, veritabanları ve kişisel projeleri barındırabilen sanal sunucular.
- **Spaces:** Veri yoğun işlemler için ideal olan bir nesne depolama hizmeti.
- **Managed Databases:** Popüler veritabanlarını yönetilen hizmetlerle destekler, veritabanı yönetimini basitleştirir.

DigitalOcean Üzerinde Dağıtım Örneği: Bir Droplet oluşturmak ve bir web uygulaması dağıtmak, DigitalOcean kontrol paneli veya komut satırı aracılığıyla gerçekleştirilebilecek birkaç adım gerektirir:

```
# Create a Droplet via DigitalOcean's dashboard
# Select an image, size, and region, and set up SSH keys

# SSH into the Droplet
ssh root@your_droplet_ip

# Install necessary packages
sudo apt-get update
sudo apt-get install nginx git python3-pip python3-dev

# Clone your project repository
git clone your_project_url

# Navigate to your project directory
cd your_project_directory

# Run your application
gunicorn --bind 0.0.0.0:80 wsgi:app
```

Sonuç Heroku, AWS ve DigitalOcean, her biri farklı dağıtım ihtiyaçlarına hitap eden benzersiz güçlü yönler sunar. Heroku, basitlik ve hızlı dağıtım yetenekleri arayan geliştiriciler için idealdir. AWS, karmaşık ve yüksek talep gören uygulamalar için uygun olan sağlam, ölçeklenebilir çözümler sunar. DigitalOcean, küçük projeler ve işletmeler için mükemmel olan basit ve ekonomik seçenekler sunar. Her platformun sunduğu araçları ve özellikleri anlayarak, geliştiriciler uygulamalarını bulutta verimli bir şekilde dağıtmak ve yönetmek için en uygun hizmeti seçebilirler.

Uygulamanızı İzleme ve Ölçeklendirme

Etkili izleme ve stratejik ölçeklendirme, kullanıcı talepleri ve yük değiştikçe web uygulamalarının performansını ve istikrarını korumak ve geliştirmek için hayati önem taşır. Bu uygulamalar, kritik performans ölçümlerinin sürekli izlenmesini ve değişen ihtiyaçları verimli bir şekilde ele almak için kaynakları dinamik olarak ayarlama yeteneğini içerir. Bu makalede, yetkin izleme ve ölçeklendirme için gerekli yöntemler, araçlar ve teknolojiler özetlenmektedir. uygulamaların ölçeklendirilmesi ve uygulama performansını optimize etmek için bunların nasıl uygulanacağı açıklanmaktadır.

İzlemenin Temelleri

İzlemenin Temelleri

İzleme, bir uygulamadan performans verilerinin toplanması, analiz edilmesi ve bunlara göre hareket edilmesine yönelik sürekli bir süreçtir. Bu süreç, kullanıcı deneyimini olumsuz etkileyebilecek veya kesinti süresine neden olabilecek performans yavaşlamaları, sistem arızaları veya operasyonel anormallikler gibi potansiyel sorunları tespit etmek için çok önemlidir.

İzlenecek Kritik Metrikler:

1. Trafik Seviyeleri: Kullanıcı trafiğini takip etmek, yükteki değişiklikleri öngörmeye ve bunlara hazırlanmaya yardımcı olur.
2. Kaynak Kullanımı: CPU, bellek ve disk kullanımı gibi metriklerin gözlemlenmesi, uygulamanın operasyonel eşikler içinde kalmasını sağlar.
3. Yanıt Süreleri ve Hata Oranları: Bu metriklerin izlenmesi, kullanıcının uygulama ile etkileşimini etkileyebilecek aksaklıkların tespit edilmesine yardımcı olur.
4. Verimlilik: Uygulamanın yönettiği işlem miktarının değerlendirilmesi, operasyonel verimliliği hakkında bilgi sağlar.
5. Veritabanı Performansı: Uygulama performansını önemli ölçüde etkileyebildikleri için veritabanı işlemlerinin izlenmesi çok önemlidir.

Bu metriklere dayalı uyarıların yapılandırılması, ekiplerin sorunları son kullanıcıları etkilemeden önce proaktif olarak yönetmesine olanak tanır.

İzleme için Araçlar

Uygulama performansının derinlemesine izlenmesini kolaylaştıran çeşitli sofistike araçlar mevcuttur:

- New Relic: Gerçek zamanlı işlemleri ve geçmiş eğilimleri izlemek için ideal olan uygulama performansına ilişkin derin bilgiler sağlar.
- Datadog: Tekliflerkapsamlı izleme yetenekleri Uygulamalar ve altyapı genelinde, geniş entegrasyonları ile dikkat çekiyor.
- Prometheus: Bulut tabanlı ortamlarda mükemmel olan bu araç, zaman serisi verilerini izlemek için idealdir.

- Nagios: Bilinen için onun kapsamlı izleme ve uyarı yetenekleri, sunucuları, ağları ve uygulamaları kapsar.

Prometheus ile Örnek Kurulum:

```
# Install Prometheus
sudo apt-get install prometheus

# Configure Prometheus to monitor your application by editing the prometheus.yml
global:
  scrape_interval: 15s  # Defines how frequently metrics should be scraped

scrape_configs:
  - job_name: 'my-application'
    static_configs:
      - targets: ['localhost:9090']

# Start Prometheus service to initiate monitoring
sudo systemctl start prometheus
```

Ölçeklendirmenin Önemi

Ölçeklendirme, ek kaynaklar tahsis ederek veya iş yükünü ek sunuculara dağıtarak artan yükleri karşılamak için gereklidir.

Ölçeklendirme Teknikleri:

1. Yatay Ölçeklendirme (Dışa Ölçeklendirme): Yükü etkili bir şekilde yaymak için daha fazla sunucu ekler.
2. Dikey Ölçeklendirme (Yukarı Ölçeklendirme) : Kapasitelerini artırmak için mevcut sunucuların yeteneklerini (CPU, bellek) artırır.

Ölçeklendirme için Araçlar

Modern araçlar ve platformlar ölçeklendirme için gelişmiş çözümler sunar:

- Kubernetes: Yönetir konteynerli uygulamalar otomatik ölçeklendirme ve yük dengeleme özelliklerine sahiptir.
- AWS Otomatik Ölçeklendirme: Performansı ve maliyeti optimize etmek için bilgi işlem kaynaklarını talebe göre otomatik olarak ayarlar.
- Azure Ölçek Kümeleri: Sanal makineler ve bulut kaynakları için otomatik ölçeklendirmeyi yönetir.

Kubernetes ile Otomatik ölçeklendirme örneği:

```yaml
# Kubernetes deployment setup
apiVersion: apps/v1
kind: Deployment
metadata:
  name: application-deployment
spec:
  replicas: 3  # Default number of replicas
  selector:
    matchLabels:
      app: application
  template:
    metadata:
      labels:
        app: application
    spec:
      containers:
      - name: application
        image: application:latest
        ports:
        - containerPort: 80
```

Bu yapılandırma, Kubernetes'e uygulamayı CPU kullanımına göre üç ile on arasında otomatik olarak ölçeklendirmesini talimat verir, böylece kaynakların verimli bir şekilde kullanılması sağlanır.

Sonuç

Güçlü izleme ve yetenekli ölçeklendirme, web uygulamalarının talepler değiştikçe uyum sağlama ve verimli bir şekilde performans gösterme yeteneğini garanti altına almak için vazgeçilmezdir. İleri düzey izleme araçları ve ölçeklenebilir teknolojiler kullanarak, operasyonlar basitleştirilebilir, maliyetler azaltılabilir ve kullanıcı memnuniyeti artırılabilir. Bu stratejilerin uygulanması, uygulamaların operasyonel taleplere uyum sağlayarak yüksek performans ve güvenilirlik standartlarını korumasını sağlar.