CloudDevOps Öğrenme Notları (Learning CloudDevOps Notes)

1. CloudDevOps'ta Ticket Oluşturmak (How to Create Tickets on CloudDevOps?)

Ticket Nedir?

Ticket, aslında bir iş kaydıdır. Proje sürecinde yapılması gereken bir iş, çözülmesi gereken bir hata (bug) veya eklenmesi gereken yeni bir özellik (requirement) için açılır. Ticket olmadan takım içinde görevler kaybolur, sorumluluklar belirsizleşir.

Ticket Nasıl Oluşturulur? (How to Create a Ticket?)

1. CloudDevOps platformuna giriş yapılır.
2. Sol menüden **Work Items** veya **Issues** alanı seçilir.
3. **+ New Ticket** butonuna tıklanarak yeni ticket açılır.
4. Gerekli alanlar doldurularak kaydedilir.

Ticket Türleri (Ticket Types)

- **Requirement (Gereksinim):** Yeni bir özellik veya iş ihtiyacı.
- **Bug (Hata):** Yazılımın beklenmeyen davranışlarını raporlar.
- **Task (Görev):** Belirli bir iş parçasını temsil eder.
- **Epic (Büyük İş Parçası):** Birden fazla requirement veya task'i kapsayan üst seviye iş.

Ticket Alanları (Ticket Fields)

- **ID:** Sistem tarafından otomatik üretilir (örn: TCK-1012).
- **Title (Başlık):** Kısa ve öz (örn: "Login butonu çalışmıyor").
- **Description (Açıklama):** Sorunun veya görevin detaylı açıklaması.
- **Assignee (Atanan Kişi):** Bu işten kim sorumlu?
- **Priority (Öncelik):** High / Medium / Low.
- **Status (Durum):** To Do, In Progress, Done.
- **Labels/Tags:** Security, Frontend, Backend gibi.

---

2. Neden CloudDevOps'ta Ticket Oluşturuyoruz? (Why Do We Create Tickets on CloudDevOps too?)

Amaç (Purpose)

- Takım içi işleri şeffaf bir şekilde organize etmek.
- Görevlerin izlenebilir hale gelmesini sağlamak.
- Agile/Scrum metodolojisinde backlog yönetimini kolaylaştırmak.

Faydaları (Benefits)

- **Traceability (İzlenebilirlik):** Requirement → Ticket → Test Case → Result zinciri kurulur.
- **Process Integration (Süreç Entegrasyonu):** Development, QA ve Ops tek platformda birleşir.
- **Quality Assurance (Kalite Güvencesi):** Net sorumluluk ve kontrol noktaları olur.

Takıma ve Projeye Katkısı

- Görev karmaşası azalır, herkes ne yapacağını bilir.
- Proje ilerlemesi kolayca takip edilir.
- İletişim eksikliği ve zaman kaybı minimize edilir.

---

3. Bug Ticket Açmak (Opening a Bug Ticket)

a) Bug Nedir? (What is a Bug?)

Bug = Yazılımda beklenen davranış ile gerçekleşen davranış arasındaki fark.

- **Basit örnek:** "Login sayfasında doğru şifreyle giriş yapılamıyor."
- **Karmaşık örnek:** "Sepete eklenen ürün ödeme sayfasında görünmüyor."

b) Anlaşılır Bug Ticket Nasıl Açılır? (How to Write a Clear Bug Ticket?)

- **Title (Başlık):** Kısa, net, tanımlayıcı → "Profile sayfasında avatar yüklenmiyor."
- **Description (Açıklama):** Ne zaman, hangi modülde yaşanıyor?
- **Environment (Ortam):** Tarayıcı, cihaz, işletim sistemi, sürüm.
- **Steps to Reproduce (Tekrar Etme Adımları):** Maddeler halinde.
- **Expected Result (Beklenen Sonuç):** Ne olmalıydı?
- **Actual Result (Gerçekleşen Sonuç):** Ne oldu?
- **Priority & Severity:** Aciliyet ve kritikliği ayırt etmek.
- **Attachments (Ekler):** Ekran görüntüsü, log dosyası.
- **Assignee (Atanan Kişi):** Hatanın çözümünden sorumlu kişi.

---

4. Örnek Bug Ticket (Tablo Formatında)

Alan (Field)	Açıklama (Explanation)
ID	BUG-2025-0142
Title	Login sayfasında doğru şifreyle giriş yapılamıyor
Description	Doğru kullanıcı adı ve şifre girildiğinde sistem ana sayfaya yönlendirmiyor, login ekranı tekrar açılıyor.
Environment	Cihaz: Lenovo ThinkPad X1 / OS: Windows 11 / Browser: Chrome 120.0 / Build: v1.3.5
Steps to Reproduce	1. Uygulamayı aç. 2. Kullanıcı adı test_user, şifre 123456 gir. 3. Login butonuna tıkla.
Expected Result	Kullanıcı başarılı giriş sonrası ana sayfaya yönlendirilmelidir.

Alan (Field)	Açıklama (Explanation)
Actual Result	Login ekranı tekrar açılıyor, hata mesajı çıkmıyor.
Priority	High
Severity	Critical
Attachments	login_error.png, auth_log.txt
Assignee	Ali Yılmaz (Backend Developer)

5. Priority vs Severity Tablosu

	Low	Medium	High / Critical
Priority (Öncelik)	Çözülmesi acil değil.	Belirli bir süre içinde çözülmeli.	Acilen çözülmeli, iş akışını engelliyor.
Severity (Önem Derecesi)	Küçük görsel/UX hatası.	Belirli modülü etkiliyor ama sistem çökmez.	Temel fonksiyon çalışmıyor, sistem kullanılamaz.

6. Ticket Lifecycle (Ticket Yaşam Döngüsü)

Bir ticket açıldıktan sonra şu adımlardan geçer:

1. **Open (Açık):** Ticket yeni oluşturulmuş, henüz kimse üzerinde çalışmıyor.
2. **Assigned (Atandı):** Ticket belirli bir kişiye atanır.
3. **In Progress (Devam Ediyor):** Developer/QA üzerinde çalışmaya başlar.
4. **Resolved (Çözüldü):** Ticket üzerinde çözüm uygulanır, test için hazır hale gelir.
5. **Verified (Doğrulandı):** QA test eder, çözümün gerçekten işe yaradığını doğrular.
6. **Closed (Kapatıldı):** Ticket tamamen çözüme kavuşur, süreç tamamlanır.

Not: Bazı durumlarda **Reopened (Yeniden Açıldı)** statüsü olabilir, eğer çözüm hatalı veya eksik çıkarsa.

7. Örnek Requirement Ticket

Alan (Field)	Açıklama (Explanation)
ID	REQ-2025-0301
Title	Kullanıcı profil sayfasında “Dark Mode” seçeneği eklenmeli
Description	Kullanıcılar profil sayfalarında görünüm ayarlarını değiştirebilmek için “Dark Mode” seçeneğine ihtiyaç duymaktadır. Bu özellik kullanıcı deneyimini iyileştirecek ve erişilebilirliği artıracaktır.
Acceptance Criteria	1. Ayarlar menüsünde Dark Mode seçeneği olmalıdır. 2. Kullanıcı seçeneği değiştirdiğinde tüm uygulama temasını değiştirmelidir. 3. Tercih, kullanıcı hesabına kaydedilmelidir.

Alan (Field)	Açıklama (Explanation)
Priority	Medium
Status	To Do
Assignee	Ayşe Demir (Frontend Developer)

8. Bugün Öğrendiklerimin Özeti (Summary of Today’s Learning)

Bugün CloudDevOps üzerinde ticket yönetimi, bug ticket açma süreci, ticket yaşam döngüsü ve requirement ticket örneğini teorik olarak öğrendim. Uygulama alanına erişimim olmadığı için pratik yapamadım ama dökümantasyonlardan ve örneklerden yola çıkarak detaylı kavradım.

Şunu fark ettim: Ticket açmak, sadece bir görev kaydı oluşturmak değil; tüm projenin yönetimsel omurgasını inşa etmek demek. Requirement, task, bug ve test case arasındaki bağlantıyı anladığımda işin mantığı daha çok oturdu.

Ayrıca bug ticket yazarken başlık, açıklama, ortam bilgisi ve tekrar etme adımları ne kadar net olursa geliştirici için iş o kadar kolaylaşıyor. Priority (öncelik) ile Severity (önem derecesi) farkını ayırt etmenin de kritik olduğunu gördüm. Ticket lifecycle sayesinde de bir işin nasıl başladığını ve nasıl kapandığını daha sistematik bir şekilde izleyebileceğimi öğrendim.

Artık biliyorum ki CloudDevOps gibi bir platformu kullanırken, sadece kod geliştirmek değil, doğru ticket açmak, bug raporlamak ve requirement tanımlamak da yazılım kalitesini belirleyen en önemli faktörlerden biri olacak.