DevSecOps Öğrenme Notları (Learning DevSecOps Notes)

1. Giriş (Introduction)

DevSecOps = DevOps + Security

- **DevOps:** Yazılım geliştirme (Development) ve operasyon (Operations) ekiplerinin işbirliğiyle hızlı teslimat ve sürekli entegrasyon/teslimat (CI/CD) sürecini sağlar.
- **DevSecOps:** Güvenliği (Security) bu sürece entegre eder. Yani güvenlik ayrı bir aşama değil, tüm yazılım yaşam döngüsünün (SDLC – Software Development Life Cycle) her adımında yer alır.

Amaç:

- **Security by Design:** Güvenlik baştan sona ürünün içine işlenmiş olmalı.
- **Shift Left:** Güvenlik testleri ve analizleri mümkün olduğunca erken, kodlama aşamasından itibaren yapılmalı.

Pipeline'da Güvenlik Örnekleri:

- **Static Code Analysis (SCA):** Kod kalitesi ve güvenlik açıklarını tespit eder.
- **Dependency Check:** Kullanılan kütüphanelerde bilinen açıklar var mı? (örn: Log4j açığı).
- **Vulnerability Scanning:** Ortam ve imaj (docker image, container) açıkları taranır.
- **Secrets Detection:** API key, password gibi gizli bilgilerin koda sızmadığını kontrol eder.

---

2. Temel Konular (Core Topics)

a) User Story Nedir? (What is a User Story?)

User Story (Kullanıcı Hikayesi):
Agile yöntemlerinde kullanılan, son kullanıcının gözünden yazılmış bir gereksinim tanımıdır.
Amaç: Teknik detaylardan çok, kullanıcının *"neye ihtiyacı var?"* sorusunu yanıtlamaktır.

Özellikleri:

- Kısa ve anlaşılırdır.
- Kullanıcı merkezlidir. (her zaman "kullanıcı ne ister?" ile başlar).
- Test edilebilir olmalıdır → sonradan Test Case'e dönüştürülebilir.

Format:

- As a [role], I want [feature] so that [benefit]
- Türkçe: Bir [rol] olarak, [özellik] istiyorum, böylece [fayda] elde ederim.

Örnek User Story'ler:

- "Bir kullanıcı olarak, hesabıma giriş yapabilmek istiyorum, böylece kişisel verilerime erişebilirim."
- "Bir müşteri olarak, sepetime ürün ekleyebilmek istiyorum, böylece alışverişimi tamamlayabilirim."

Yapısı:

- **ID:** US_01, US_02 şeklinde benzersiz kimlik.
- **Title (Başlık):** Kısa ifade (örn: *"Kullanıcı giriş yapabilmeli"*).
- **Description (Açıklama):** Kullanıcının ihtiyacını daha detaylı anlatır.
- **Acceptance Criteria (Kabul Kriterleri):** Bu story'nin tamamlandığını nasıl anlayacağız?
    - Kullanıcı doğru şifre ile giriş yapabiliyor.
    - Hatalı şifre girildiğinde sistem uyarı veriyor.

Avantajları:

- Takımı kullanıcı odaklı düşünmeye zorlar.
- Teknik olmayan kişiler için de anlaşılırdır.
- Küçük parçalara ayrılarak planlanabilir.

---

b) Ticket Nedir? (What is a Ticket?)

Ticket = İş kaydı / görev kartı (Work Record / Task Card).
Yazılım geliştirme ve proje yönetimi dünyasında bir işi, hatayı veya isteği takip edebilmek için açılan kayıttır. "Bilet" anlamından gelir → Her iş için bir bilet açılır, bu bilet işin takibi bitene kadar sistemde kalır.

Ticket Ne İşe Yarar?

- İşleri görünür kılar (Kim ne yapıyor? Hangi işler açık, hangileri kapalı?).
- Sorumluluk dağıtır (Her ticket bir kişiye atanır → Assignee).
- Öncelik belirler (Örn: Kritik güvenlik açığı = High Priority, UI buton rengi = Low Priority).
- İzlenebilirlik sağlar (Bir işin hangi User Story'den doğduğunu, hangi Test Case ile doğrulandığını görebilirsin).

Ticket Türleri:

- **Bug / Defect Ticket:** Hata veya açık raporlamak için → Örn: *"Login sayfasında şifre alanı maskelenmiyor"*.
- **Requirement / Feature Ticket:** Yeni özellik talebi → Örn: *"Sisteme 2FA eklenmeli"*.
- **Task Ticket:** Teknik görevler → Örn: *"Test ortamına yeni database kurulumu yap"*.

Ticket İçinde Bulunanlar:

- **ID (Benzersiz Kimlik):** TCK-1234
- **Başlık (Title):** İşin özeti
- **Açıklama (Description):** Detay, ekran görüntüsü, hata adımları
- **Öncelik (Priority):** High, Medium, Low
- **Sorumlu (Assignee):** İşi yapan kişi
- **Durum (Status):** To Do, In Progress, In Review, Done
- **Tarihler (Dates):** Açılış ve deadline tarihi
- **Etiketler (Labels/Tags):** Security, Frontend, Backend

Ticket Süreci (Lifecycle):

- Open / To Do → Yeni açıldı

- In Progress → Üzerinde çalışılıyor
- In Review / Testing → Kod review / test
- Done / Closed → Kapandı

Örnek Ticket:

- User Story: "Kullanıcı giriş yapabilmeli"
- Ticket (Bug): "Login sayfasında şifre alanı maskelenmiyor"
- Assignee: Frontend developer
- Priority: High
- Status: To Do

---

c) Ticket Nasıl Açılır? (How to Create Tickets on DevSecOps?)

Ticket açarken kullanılan platformlar: **Jira, GitLab, Azure Boards, Huawei Cloud DevSecOps** vb.

Tipik Ticket Bilgileri:

- ID → TCK-1032
- Title → "Login sayfasında şifre maskelenmiyor"
- Description → Hatanın nasıl ortaya çıktığı, test ortamı, ekran görüntüsü/log
- Priority → High / Medium / Low
- Assignee → Sorumlu kişi
- Dates → Created Date, Due Date
- Labels → Security, Frontend, Backend

Örnek Ticket:

- User Story (US): "Kullanıcı giriş yapabilmeli"
- Ticket: "Login sayfasında şifre alanı maskelenmiyor"
- Assignee: Frontend Developer
- Priority: High

---

d) DevSecOps Kuralları Nelerdir? (What are the Rules of DevSecOps?)

Temel Kurallar (Main Rules):

- **Statik Kod Analizi (Static Code Analysis – SCA):**
  Kod push edildiğinde otomatik tarama yapılır.
  Araçlar (Tools): SonarQube, Checkmarx

- **Bağımlılık ve Açıklık Kontrolü (Dependency & Vulnerability Check):**
  Kullanılan framework ve kütüphanelerde bilinen açıkların taranması
  Araçlar: OWASP Dependency-Check, Snyk

- **Kimlik ve Erişim Yönetimi (Identity & Access Management – IAM):**
  "Minimum Yetki (Least Privilege)" mantığı uygulanır.

- **Sürekli İzleme ve Loglama (Continuous Monitoring & Logging):**
  Sistem sürekli izlenir, log'lar toplanır.
  Araçlar: SIEM (Security Information and Event Management)

- **Güvenli CI/CD Pipeline (Secure CI/CD Pipeline):**
  - Her aşamada güvenlik testleri (Dev, Test, Staging, Prod)
  - Kritik açık bulunursa otomatik "build fail" mekanizması

---

e) Ticket, User Story ve Test Case İlişkisi (How to Associate Ticket with User Story & Test Cases)

Amaç: **Traceability Matrix (İzlenebilirlik Matrisi)** oluşturmak.

- **Ticket ↔ User Story bağlantısı:**
  Örn: US_01 → "Kullanıcı giriş yapabilmeli"
  Ticket → "Login sayfasında şifre maskelenmiyor"

- **Ticket ↔ Test Case bağlantısı:**
  Ticket kapatıldığında hangi test ile doğrulanacak?
  Örn: TestCase_03 → "Şifre alanı maskeleniyor mu?"

Faydaları:

- İzlenebilirlik: Hangi US'ten doğdu, hangi test ile doğrulandı?
- Kalite güvencesi: Fix → Test Case → Doğrulama zinciri tamamlanır.
- Regülasyon: Requirement → Implementation → Test ilişkisi belgelenmiş olur.

---

3. Pratik Adımlar (Practical Steps)

Senaryo:

- User Story: "Kullanıcı giriş yapabilmeli"
- Ticket: "Login sayfasında şifre alanı maskelenmiyor"
- Association: US_01 ile ilişkilendir, TestCase_03 ile doğrula

Sonuç:
Proje yöneticisi şunu görebilir:

- US_01 tamamlandı mı?
- Kaç ticket açıldı?
- Bu ticket hangi test case ile doğrulandı?
- Açıklar kapatıldı mı?

---

4. Kapanış (Conclusion)

DevSecOps sadece güvenlik değil, aynı zamanda:

- **İzlenebilirlik (Traceability)**
- **Süreç entegrasyonu (Process Integration)**

- **Kalite güvencesi (Quality Assurance)**

Ticket Açmak = Proje Yönetiminin Temeli
Doğru ticket yönetimi → Doğru yazılım kalitesi.

Bugün öğrendiklerimin kısa özeti

Bugün, **ticket, user story ve test case** kavramlarını daha net öğrendim. Bu kavramların tek başına birer terim olmadığını, aslında birbiriyle bağlantılı çalışarak yazılım geliştirme sürecinde **izlenebilirlik** ve **şeffaflık** sağladığını fark ettim.

Ticket açmanın sadece *"bir görev eklemek"* olmadığını, aynı zamanda **işin sorumluluğunu belirlemek, önceliklendirmek ve doğru şekilde takip edebilmek** için kritik bir yapı taşı olduğunu gördüm.

Bu bilgilerin bana projelerde sağlayacağı fayda ise şu olacak:
Bir projede görev alırken sadece **kod yazmanın yeterli olmadığını**, işi sistematik bir şekilde ticket'lar üzerinden yönetmenin önemini artık biliyorum. Böylece hem **takım içi iletişimim daha güçlü olacak** hem de **işlerin takibi çok daha düzenli ilerleyecek**.

Ayrıca ileride karşılaşabileceğim DevSecOps ortamlarında, **ticket–user story–test case bağlantısını yapabilmem**, işimi daha profesyonel ve güvenli şekilde yürütebilmemi sağlayacak.