T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

ISE460 SİSTEM YÖNETİCİLİĞİ

ZABBIX TELEGRAM ENTEGRASYONU

B211200034 – Tuğçe KAYA
B221200015 – Özgür ÖZGENÇ

Fakülte Anabilim Dalı : BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ
Ödev Danışmanı : Öğr.Gör. UĞUR ÖZBEK

Video Linki:
https://youtu.be/hY6Ax3jkKXI?si=r1sX-cefPSDwLIx2

Github Linki:
https://github.com/TugceKaya01/SistemYoneticiligi_Zabbix

2024-2025 Bahar Dönemi

İçindekiler

1. Giriş

Bu proje, sistem izleme ve uyarı yönetiminde güçlü bir araç olan Zabbix'in Oracle VirtualBox üzerinde çalışan Ubuntu 24.04 sanal makinesine kurularak yapılandırılmasını kapsamaktadır. Projenin temel amacı, Zabbix aracılığıyla izlenen sistemlerde meydana gelen kritik olayların ve önemli bildirimlerin Telegram botu kullanılarak anlık olarak sistem yöneticilerine iletilmesidir. Ayrıca, Zabbix arayüzünde sunuculara özel dashboard'lar oluşturularak verilerin daha etkili bir şekilde görselleştirilmesi hedeflenmiştir.

Aşağıdaki tabloda görev dağılımları belirtilmiştir:

Aşama	Sorumlu Kişi
Ubuntu 24.04 Sanal Makine Kurulumu ve Yapılandırılması	Tuğçe Kaya
Zabbix 7.0 LTS Kurulumu ve Yapılandırılması	Tuğçe Kaya
Telegram Bot Oluşturulması (BotFather ile)	Özgür Özgenç
Dashboard Oluşturulması ve Kişiselleştirme	Özgür Özgenç
Sistem Testlerinin Yapılması ve Sonuçların Değerlendirilmesi	Ortak Çalışma
Genel Rapor Hazırlığı ve Sonuç Bölümü	Ortak Çalışma

---

2. Proje Amacı ve Hedefler

Amaç: Zabbix, sistem izleme ve uyarı yönetimi için kullanılan güçlü bir araçtır. Telegram entegrasyonu sayesinde, sistemde meydana gelen hataları, kritik olayları ve önemli bildirimleri anlık olarak Telegram üzerinden alabilirsiniz.

Hedefler:

- Zabbix'te kritik olaylar için otomatik Telegram bildirimleri oluşturmak.
- Telegram botu kullanarak sistem yöneticilerine gerçek zamanlı uyarılar göndermek.
- Bildirimlerin doğruluğunu ve zamanlamasını test etmek.
- Sistem metriklerini ve uyarılarını gösteren kişiselleştirilmiş dashboard'lar oluşturmak.

---

3. Kullanılan Araçlar ve Ortam

Projenin gerçekleştirilmesinde aşağıdaki yazılım ve platformlar kullanılmıştır:

- **Sanallaştırma Platformu:** Oracle VirtualBox
- **Sanal Makine İşletim Sistemi:** Ubuntu 24.04 LTS
    - **RAM:** 4 GB
    - **İşlemci:** 4 Çekirdek
    - **Sanal Disk:** 25 GB
- **İzleme Yazılımı:** Zabbix 7.0 LTS
- **Bildirim Entegrasyonu:** Telegram Bot (BotFather ile oluşturuldu)
- **Entegrasyon Yöntemi:** Zabbix Webhook
- **Yardımcı Araçlar:** cURL veya Python (API istekleri için), Bash veya Python script'i (Mesajları göndermek için) **-***Bu kısım genel gereksinim olup, projede webhook kullanıldığı için Zabbix'in kendi entegrasyon mekanizması yeterli olmuştur.-*

3.1. Sanal Makine Nedir?

Sanal makine (Virtual Machine - VM), fiziksel bir bilgisayar sistemi üzerinde, o bilgisayarın donanım kaynaklarını (işlemci, bellek, depolama vb.) paylaşarak çalışan, ancak kendi işletim sistemine ve uygulamalarına sahip olan yazılım tabanlı bir bilgisayar ortamıdır. Sanal makineler, ana bilgisayarın (host) işletim sisteminden bağımsız olarak çalışır ve sanki ayrı bir fiziksel bilgisayarmış gibi davranır.

Başlıca Avantajları:

- **Kaynak Verimliliği:** Tek bir fiziksel sunucu üzerinde birden fazla sanal makine çalıştırılarak donanım kaynakları daha verimli kullanılır.
- **İzolasyon:** Sanal makineler birbirinden izole çalışır. Bir sanal makinede oluşan sorun, diğerlerini veya ana sistemi etkilemez.
- **Taşınabilirlik:** Sanal makineler, imaj dosyaları olarak kolayca farklı fiziksel sunuculara taşınabilir.
- **Test ve Geliştirme:** Farklı işletim sistemleri ve yapılandırmalar denemek için güvenli ve izole ortamlar sunar.
- **Maliyet Tasarrufu:** Ayrı fiziksel sunuculara olan ihtiyacı azaltarak donanım ve enerji maliyetlerinden tasarruf sağlar.

Bu projede, Oracle VirtualBox kullanılarak Ubuntu 24.04 işletim sistemine sahip bir sanal makine oluşturulmuş ve Zabbix sunucusu bu sanal ortam üzerine kurulmuştur.

3.1.1. Oracle VirtualBox Kurulumu

Oracle VirtualBox, farklı işletim sistemlerinde sanal makineler oluşturmak ve çalıştırmak için kullanılan ücretsiz ve açık kaynak kodlu bir sanallaştırma yazılımıdır.

1. **İndirme:** Oracle VirtualBox'ın resmî web sitesinden (virtualbox.org) işletim sisteminize uygun en son sürüm kurulum dosyası indirilir.

2. **Kurulum:** İndirilen kurulum dosyası çalıştırılır ve kurulum sihirbazındaki adımlar takip edilerek kurulum tamamlanır. Kurulum sırasında "VirtualBox Extension Pack" indirilip kurulması da faydalı özellikler (USB 2.0/3.0 desteği, RDP vb.) sunar.
3. **Başlatma:** Kurulum tamamlandıktan sonra VirtualBox uygulaması başlatılır.

3.1.2. Ubuntu 24.04 Sanal Makine Kurulumu ve Yapılandırılması

Zabbix sunucusunun çalışacağı Ubuntu 24.04 LTS işletim sistemi, Oracle VirtualBox üzerinde bir sanal makine olarak kurulmuştur.

1. **Ubuntu ISO İndirme:** Ubuntu'nun resmî web sitesinden (ubuntu.com) Ubuntu 24.04 LTS Desktop veya Server ISO dosyası indirilir. Bu proje için Server sürümü daha uygun olabilir, ancak Desktop sürümü kullanıldı.
2. **VirtualBox'ta Yeni Sanal Makine Oluşturma:**
    - VirtualBox arayüzünde "New" (Yeni) butonuna tıklanır.
    - **Name:** Sanal makineye bir isim verilir (örn: "Zabbix-Ubuntu-Server").
    - **Type:** "Linux" seçilir.
    - **Version:** "Ubuntu (64-bit)" seçilir.
    - **Memory size:** RAM miktarı belirlenir (Proje için 4 GB (4096 MB) ayrılmıştır).
    - **Hard disk:** "Create a virtual hard disk now" seçilir ve "Create" butonuna tıklanır.
        - **Hard disk file type:** "VDI (VirtualBox Disk Image)" seçilir.
        - **Storage on physical hard disk:** "Dynamically allocated" seçilerek disk alanının kullandıkça büyümesi sağlanır.
        - **File location and size:** Sanal disk dosyasının adı ve boyutu belirlenir (Proje için 25 GB ayrılmıştır). "Create" butonuna tıklanır.
3. **Sanal Makine Ayarları:**
    - Oluşturulan sanal makine seçiliyken "Settings" (Ayarlar) butonuna tıklanır.

- **System > Processor:** İşlemci sayısı 4 çekirdek olarak ayarlanır.
- **Storage:** "Controller: IDE" altındaki "Empty" CD/DVD simgesine tıklanır. Sağ taraftaki CD simgesinden "Choose a disk file..." seçilerek indirilen Ubuntu 24.04 ISO dosyası gösterilir.
- **Network:** "Adapter 1" için "Enable Network Adapter" seçili olmalıdır. "Attached to" kısmında "Bridged Adapter" (ağınızdaki diğer cihazlarla aynı IP bloğunda olması için) veya "NAT" (daha izole bir ağ için) seçilebilir. Proje gereksinimlerine göre uygun olan seçilir. Genellikle "Bridged Adapter" Zabbix sunucusuna ağ üzerinden erişim için daha pratiktir.
- Diğer ayarlar varsayılan olarak bırakılabilir. "OK" ile ayarlar kaydedilir.

4. **Ubuntu Kurulumu:**
    - Sanal makine "Start" (Başlat) butonu ile başlatılır.
    - Ubuntu kurulum ekranı geldiğinde dil ve klavye ayarları seçilerek kurulum adımları takip edilir.
    - Disk yapılandırmasında "Use an entire disk" (Tüm diski kullan) seçeneği genellikle yeterlidir.
    - Kullanıcı adı, şifre gibi bilgiler girilir.
    - Kurulum sırasında "Install OpenSSH server" seçeneği işaretlenirse, sanal makineye SSH ile uzaktan erişim kolaylaşır.
    - Kurulum tamamlandıktan sonra sanal makine yeniden başlatılır.
5. **İlk Güncellemeler ve Gerekli Paketler:**
    - Ubuntu sanal makinesine giriş yapılır.
    - Terminal açılarak aşağıdaki komutlarla sistem güncellenir:

```
sudo apt update
sudo apt upgrade -y
```

3.2. Zabbix Nedir?

Zabbix, açık kaynak kodlu, kurumsal düzeyde bir ağ ve uygulama izleme çözümüdür. Sunucuların, ağ cihazlarının, sanal makinelerin, servislerin ve uygulamaların performansını ve kullanılabilirliğini gerçek zamanlı olarak izlemek için kullanılır. Zabbix, metrik toplama, sorun tespiti, uyarı gönderme, görselleştirme ve raporlama gibi geniş bir yelpazede özellikler sunar.

Temel Özellikleri:

- **Geniş Kapsamlı İzleme:** İşletim sistemi metrikleri (CPU, bellek, disk, ağ), ağ trafiği, uygulama performansı, veritabanı durumu, web siteleri ve daha birçok bileşeni izleyebilir.
- **Esnek Veri Toplama Yöntemleri:** Zabbix ajanları, SNMP, IPMI, JMX, web senaryoları, özel script'ler ve daha birçok yöntemle veri toplayabilir.
- **Güçlü Sorun Tespiti (Triggers):** Toplanan verilere dayalı olarak karmaşık mantıksal koşullar tanımlanarak sorunların otomatik olarak tespit edilmesini sağlar.
- **Çok Kanallı Uyarı Sistemi:** E-posta, SMS, Slack, Telegram gibi çeşitli kanallar üzerinden anlık uyarılar gönderebilir.
- **Görselleştirme:** Grafikler, haritalar ve dashboard'lar aracılığıyla izlenen verilerin kolayca anlaşılmasını sağlar.
- **Ölçeklenebilirlik:** Küçük ortamlardan binlerce cihazın izlendiği büyük altyapılara kadar ölçeklenebilir.

Bu projede Zabbix 7.0 LTS sürümü kullanılarak sistemdeki problemlerin tespiti ve Telegram aracılığıyla bildirilmesi sağlanmıştır.

3.2.1. Zabbix 7.0 LTS Kurulumu

Zabbix 7.0 LTS sürümü, Ubuntu 24.04 üzerine kurulmuştur. Kurulum genellikle Zabbix Server, Zabbix Frontend (web arayüzü) ve Zabbix Agent bileşenlerini içerir. Veritabanı olarak MySQL veya PostgreSQL kullanılabilir. Bu örnekte MySQL (veya MariaDB) kullanılacaktır.

1. **Root kullanıcı olun:**

   Root ayrıcalıklarıyla yeni bir kabuk oturumu başlatın

```
$ sudo -s
```

2. **Zabbix deposunu kurun:**

```
# wget https://repo.zabbix.com/zabbix/7.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest_7.0+ubuntu24.04_all.deb
# dpkg -i zabbix-release_latest_7.0+ubuntu24.04_all.deb
# apt update
```

3. **Zabbix sunucusunu, arayüzünü ve aracısını kurun:**

```
# apt install zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf zabbix-sql-scripts zabbix-agent
```

4. **İlk veritabanını oluşturun:**

   MySQL'e root olarak bağlanın (kurulum sırasında belirlediğiniz parolayı girmeniz istenecektir)

```
# mysql -uroot -p
password
mysql> create database zabbix character set utf8mb4 collate utf8mb4_bin;
mysql> create user zabbix@localhost identified by 'password';
mysql> grant all privileges on zabbix.* to zabbix@localhost;
mysql> set global log_bin_trust_function_creators = 1;
mysql> quit;
```

   Şimdi Zabbix şemasını ve verilerini içe aktarın

```
# zcat /usr/share/zabbix-sql-scripts/mysql/server.sql.gz | mysql --default-character-set=utf8mb4 -uzabbix -p zabbix
```

   Veritabanı şemasını içe aktardıktan sonra log_bin_trust_function_creators seçeneğini devre dışı bırakın

```
# mysql -uroot -p
password
```

```
mysql> set global log_bin_trust_function_creators = 0;
mysql> quit;
```

5. **Zabbix sunucusu için veritabanını yapılandırın:**
   Edit file /etc/zabbix/zabbix_server.conf

```
DBPassword=password
```

6. **Zabbix sunucusu ve aracısı işlemlerini başlatın:**
   Zabbix sunucusunu, aracısını ve Apache'yi yeniden başlatın ve açılışta otomatik başlamalarını sağlayın

```
# systemctl restart zabbix-server zabbix-agent apache2
# systemctl enable zabbix-server zabbix-agent apache2
```

7. **Zabbix arayüzünü yapılandırın:**
   Bir web tarayıcısı açın ve Zabbix sunucunuzun IP adresini veya alan adını girerek Zabbix arayüzüne gidin: http://host/zabbix[1]

3.3. Telegram Bot Nedir? Telegram

Telegram Bot, Telegram mesajlaşma platformu üzerinde çalışan üçüncü taraf uygulamalardır. Bot'lar, kullanıcılarla etkileşim kurabilen, komutları yerine getirebilen, bilgi sağlayabilen veya çeşitli otomasyon görevlerini gerçekleştirebilen otomatik hesaplardır. Telegram Bot API'si kullanılarak geliştirilirler.

Kullanım Alanları:

- **Bildirimler:** Sistem uyarıları, haber güncellemeleri gibi bilgileri anlık olarak iletebilirler.
- **Müşteri Hizmetleri:** Sıkça sorulan soruları yanıtlayabilir, destek taleplerini alabilirler.
- **Otomasyon:** Tekrarlayan görevleri otomatikleştirebilirler (örneğin, dosya dönüştürme, hatırlatıcı ayarlama).

- **Entegrasyonlar:** Diğer servislerle (hava durumu, çeviri, Zabbix gibi izleme araçları) entegre çalışabilirler.

Bu projede, Telegram'ın **BotFather** adlı özel botu kullanılarak yeni bir bot oluşturulmuştur. Bu bot, Zabbix'ten gelen uyarı mesajlarını alıp ilgili kullanıcılara (sistem yöneticilerine) iletmek üzere yapılandırılmıştır. Zabbix ile Telegram arasındaki bağlantı, bir webhook aracılığıyla sağlanmıştır. Bu sayede, Zabbix'te tanımlanan bir sorun (trigger) tetiklendiğinde, otomatik olarak Telegram botu üzerinden anlık bildirim gönderilmesi mümkün olmuştur.

3.3.1. Telegram Bot Oluşturulması

Zabbix'ten uyarıları almak için bir Telegram botu oluşturulması gerekmektedir.

1. Telegram uygulamasında **BotFather** adlı kullanıcıyı arayın ve sohbet başlatın.
2. Yeni bir bot oluşturmak için /newbot komutunu gönderin.
3. BotFather sizden bot için bir görünen ad ve ardından bir kullanıcı adı isteyecektir. Kullanıcı adı benzersiz olmalı ve "bot" ile bitmelidir (örn: ZabbixBot).
4. Başarılı bir şekilde oluşturulduğunda, BotFather size botunuz için bir **HTTP API token** verecektir. Bu token, Zabbix'in botunuzla iletişim kurması için kritik öneme sahiptir. Bu token'ı güvenli bir yerde saklayın.
5. Ayrıca, bildirimlerin gönderileceği Telegram kullanıcısının veya grubunun **Chat ID**'sine ihtiyacınız olacaktır. Kişisel Chat ID'nizi öğrenmek için bota bir mesaj gönderin ve ardından https://api.telegram.org/bot<YOUR_BOT_TOKEN>/getUpdates URL'sini tarayıcıda açarak gelen JSON yanıtından chat nesnesindeki id değerini bulun.

Bu token ve Chat ID, Zabbix'te Media Type yapılandırmasında kullanılacaktır.

---

4. Zabbix Yapılandırması ve Entegrasyon

4.1. Media Type Oluşturma (Webhook ile Telegram)

Zabbix'in harici sistemlerle (bu projede Telegram) iletişim kurabilmesi için bir **Media Type** tanımlanması gerekir. Media Type, Zabbix'in bir uyarıyı hangi yöntemle ve nasıl göndereceğini belirler. Telegram entegrasyonu için genellikle bir webhook kullanılır.

Adımlar (Genel Yaklaşım):

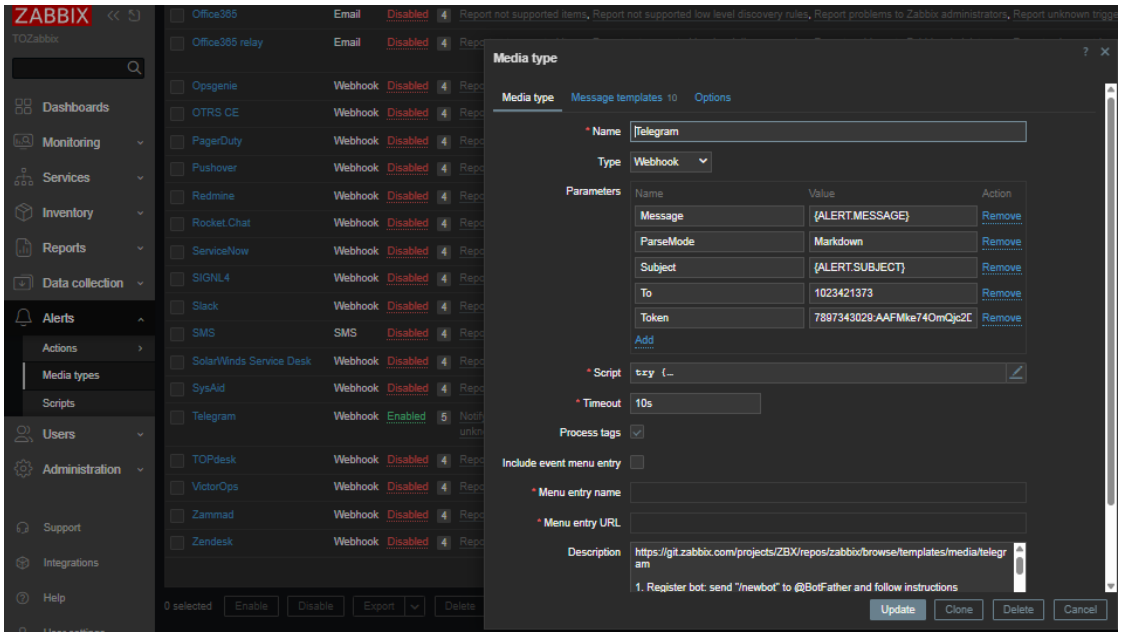
1. **Zabbix'te Media Type Tanımlama:**
    - Zabbix arayüzünde **Alerts > Media types** bölümüne gidilir.
    - **Create media type** butonuna tıklanır.
    - **Name:** Anlaşılır bir isim verilir.
    - **Type:** "Webhook" seçilir.
    - **Parameters:** Telegram API'sine istek göndermek için gerekli parametreler tanımlanır. Bu genellikle şunları içerir:
        - {ALERT.SENDTO}: Telegram Chat ID'si (kullanıcı veya grup).
        - {ALERT.MESSAGE}: Gönderilecek mesaj içeriği.
        - Telegram Bot API Token.
    - **Script:** Webhook'un Telegram API'sine nasıl bir HTTP isteği (genellikle POST) göndereceğini tanımlayan JavaScript kodu girilir. Bu script, Zabbix.Log() gibi Zabbix'in dahili fonksiyonlarını kullanarak hata ayıklama yapabilir ve Telegram API'sinin sendMessage metodunu çağırır.

```
try {
    var params = JSON.parse(value);

    // Gerekli parametrelerin kontrolü
    if (!params.Token) {
        throw 'Bot token "Token" tanımlanmamış.';
    }
    if (!params.To) {
        throw 'Alıcı "To" (chat_id) tanımlanmamış.';
    }
    if (!params.Message) {
        throw 'Parametre "Message" tanımlanmamış.';
```

```
    }
     // Subject (Konu) genellikle sağlanır ama mesajın bir parçası olarak kullanılır.
     // ParseMode isteğe bağlıdır; Telegram belirtilmemişse veya geçersizse varsayılanı kullanır.

    var request = new HttpRequest();

    // İsteğe bağlı: Proxy desteği (Eğer Zabbix arayüzünde HTTPProxy parametresi eklerseniz)
    if (typeof params.HTTPProxy === 'string' && params.HTTPProxy.trim() !== '') {
        request.setProxy(params.HTTPProxy);
             Zabbix.Log(4, '[Telegram Webhook] Proxy kullanılıyor: ' + params.HTTPProxy);
    }

    request.addHeader('Content-Type: application/json');

    var message_text = params.Message;
    // Eğer Subject (Konu) parametresi doluysa, mesajın başına ekle
    if (typeof params.Subject === 'string' && params.Subject.trim() !== '') {
        message_text = params.Subject + '\n' + params.Message;
    }

    var body = {
        chat_id: params.To,
        text: message_text
    };

    // ParseMode parametresini ayarla (Markdown, HTML, MarkdownV2)
      if (typeof params.ParseMode === 'string' && ['Markdown', 'HTML', 'MarkdownV2'].indexOf(params.ParseMode) !== -1) {
        body.parse_mode = params.ParseMode;
    }

    var url = 'https://api.telegram.org/bot' + params.Token + '/sendMessage';

    Zabbix.Log(4, '[Telegram Webhook] Telegram bildirimi gönderiliyor. URL: ' + url.replace(params.Token, '<TOKEN_REDACTED>') + ', Body: ' + JSON.stringify(body));

    var response = request.post(url, JSON.stringify(body));

    Zabbix.Log(4, '[Telegram Webhook] Yanıt durumu: ' + request.getStatus() + '. Yanıt gövdesi: ' + response);

    if (request.getStatus() < 200 || request.getStatus() >= 300) {
```

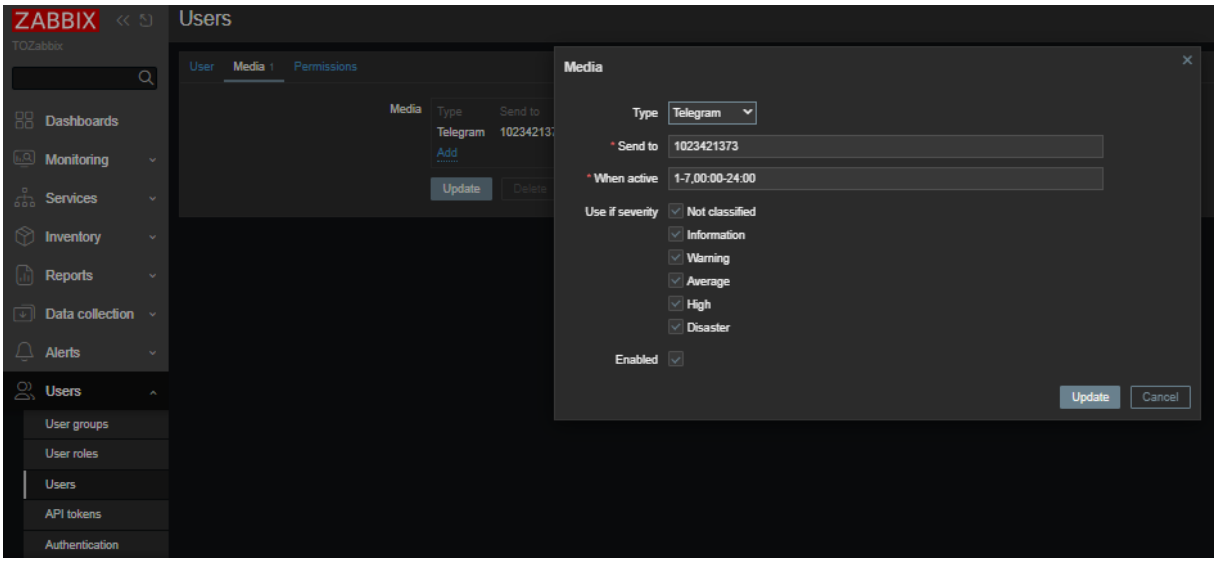
```
            throw 'Telegram API isteği başarısız oldu. HTTP durumu: ' +
request.getStatus() + '. Yanıt: ' + response;
    }

    var response_json = JSON.parse(response);
    if (response_json.ok !== true) {
        throw 'Telegram API bir hata döndürdü: ' + response_json.description +
' (hata kodu: ' + response_json.error_code + ')';
    }

    // Başarıyla gönderildi
    return JSON.stringify(response_json);

} catch (error) {
    Zabbix.Log(3, '[Telegram Webhook] Hata: ' + error);
    // Zabbix, başarısız medya türü yürütmesi için bir hata fırlatılmasını bekler
    if (typeof error === 'object' && error !== null) {
        // Hata nesnesiyse, mesajını veya kendisini string'e çevir
        throw JSON.stringify({ error: String(error.message || error) });
    }
    throw String(error); // Hata zaten bir string ise doğrudan fırlat
}
```

- **Message templates:** Farklı olay türleri (Problem, Problem recovery, Problem update vb.) için varsayılan mesaj şablonları tanımlanır. Bu şablonlarda Zabbix makroları ({EVENT.NAME}, {HOST.NAME}, {ITEM.VALUE} vb.) kullanılarak dinamik mesajlar oluşturulabilir.



2. **Kullanıcıya Media Atama:**

    - **Users > Users** bölümünden ilgili kullanıcı seçilir.
    - **Media** sekmesine gelinir ve **Add** ile yeni oluşturulan Telegram Media Type'ı eklenir.
    - **Send to** kısmına Telegram Chat ID'si girilir.
    - Hangi önem seviyesindeki (Severity) uyarıların bu media type ile gönderileceği seçilir.
    - Media type etkinleştirilir.

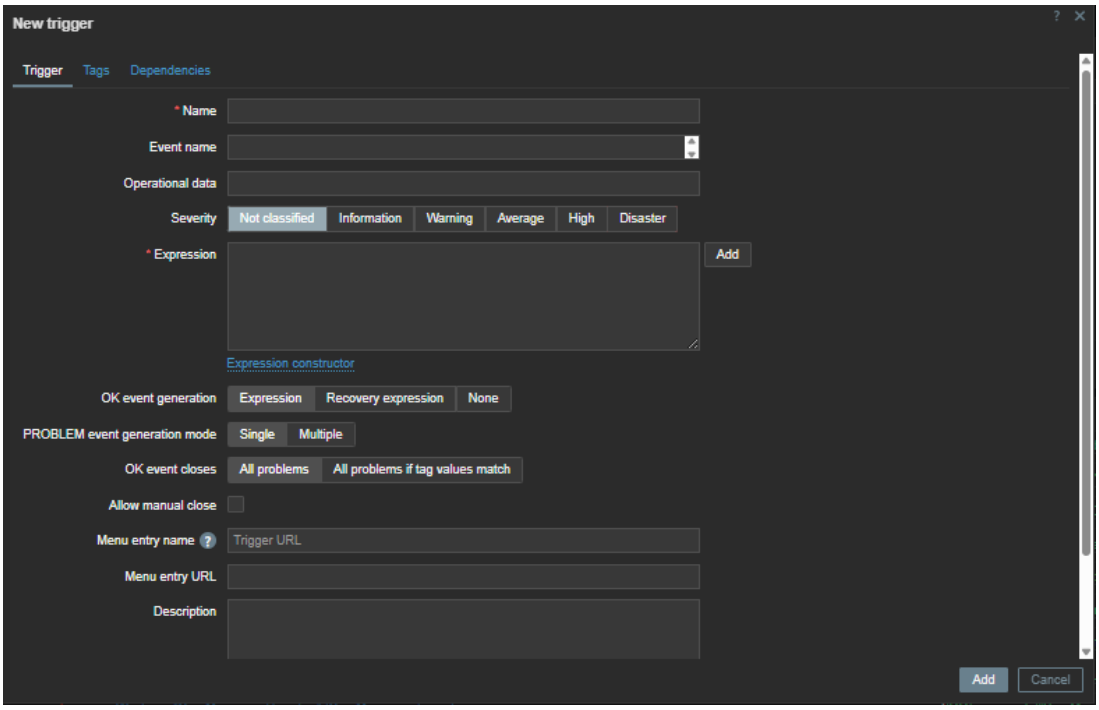


4.2. Trigger Kurma

Trigger, Zabbix'te izlenen bir itemden gelen verileri değerlendirerek belirli bir koşulun (örneğin, CPU kullanımı %90'ın üzerinde) karşılanıp karşılanmadığını kontrol eden mantıksal ifadelerdir. Koşul karşılandığında trigger durumu "PROBLEM" olarak değişir ve tanımlı aksiyonların (örneğin, bildirim gönderme) tetiklenmesini sağlar.

Trigger Oluşturma Adımları:

1. **Host veya Template Seçimi:** Trigger'lar genellikle bir host'a (izlenen cihaz) veya bir template'e (şablon) bağlı olarak oluşturulur. Template kullanmak, aynı türdeki birçok host için trigger'ları merkezi olarak yönetmeyi kolaylaştırır.
2. **Data Collection > Hosts** bölümüne gidilir.
3. İlgili host'un veya template'in satırındaki **Triggers** linkine tıklanır.
4. **Create trigger** butonuna tıklanır.
5. **Trigger Formu Doldurulur:**
    - **Name:** Trigger için açıklayıcı bir isim verilir (örneğin, "Yüksek CPU Kullanımı - {HOST.NAME}"). {HOST.NAME} gibi makrolar kullanılabilir.
    - **Severity:** Trigger'ın önem derecesi seçilir (Not classified, Information, Warning, Average, High, Disaster). Bu, bildirimlerin filtrelenmesinde ve gösterilmesinde kullanılır.
    - **Expression:** Trigger'ın ne zaman "PROBLEM" durumuna geçeceğini belirleyen mantıksal ifadedir. Bu ifade, Zabbix item'larını, fonksiyonları ve operatörleri kullanır.
        - **Add** butonuna tıklanarak ifade oluşturucu açılır.
        - **Item:** İzlenecek item seçilir (örneğin, bir sunucunun CPU kullanım yüzdesi).
        - **Function:** Item'ın son değeri (last()), belirli bir periyottaki ortalaması (avg()), minimumu (min()), maksimumu (max()) gibi bir fonksiyon seçilir.
        - **Operator:** Karşılaştırma operatörü (>, <, =, >=, <=, <>) seçilir.
        - **Constant/Value:** Karşılaştırılacak eşik değer girilir.
    - **Tags:** Trigger'a etiketler eklenerek sınıflandırma ve filtreleme kolaylaştırılabilir.



Bu projede, sistemdeki kritik durumları (örneğin, yüksek CPU, düşük disk alanı, servis durması vb.) tespit etmek için çeşitli trigger'lar yapılandırılmıştır. Bu trigger'lar tetiklendiğinde, bir önceki adımda oluşturulan Telegram media type'ı aracılığıyla anlık bildirim gönderilmesi için aksiyonlar (actions) tanımlanmıştır.

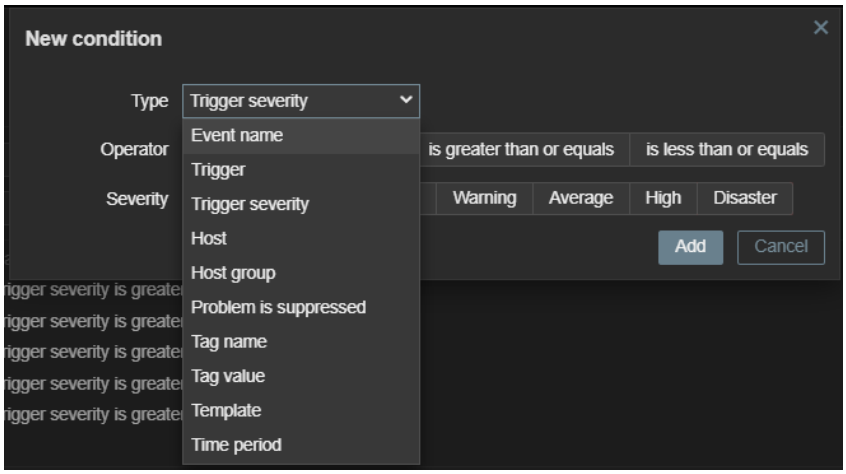
4.3. Action Oluşturma

Aksiyonlar, Zabbix'te belirli olaylar (genellikle trigger durum değişiklikleri) meydana geldiğinde otomatik olarak gerçekleştirilecek işlemleri tanımlar. Bu projede, trigger'lar "PROBLEM" durumuna geçtiğinde Telegram üzerinden bildirim göndermek için aksiyonlar yapılandırılacaktır.

Action Oluşturma Adımları:

1. **Action Menüsüne Erişim:**
    - Zabbix arayüzünde **Alerts > Actions > Trigger Actions** bölümüne gidilir.
    - Create action butonuna tıklanır.
2. **Action Sekmesi (Ana Ayarlar):**

- **Name:** Aksiyon için açıklayıcı bir isim verilir (örn: "Kritik Sorunlar için Telegram Bildirimi").
- **Conditions:** Aksiyonun hangi koşullarda tetikleneceğini belirler. Birden fazla koşul eklenebilir ve "AND/OR" mantığıyla (Type of calculation) birleştirilebilir.
    - New condition ile koşul eklenir.
    - **Type:** Koşul türü seçilir. Örnekler:
        - Trigger
        - Trigger severity >= High (Önem derecesi "High" veya "Disaster" olan trigger'lar için)
        - Host group = Linux Servers (Sadece belirli bir host grubundaki sunucular için)



    - **Operator:** Seçilen türe göre operatör belirlenir (equals, contains, >=, <=, etc.).
    - **Value:** Karşılaştırılacak değer girilir.
- **Enabled:** Aksiyonun aktif olması için bu kutucuk işaretli olmalıdır.

3. **Operations Sekmesi (İşlemler):**
    - Aksiyon tetiklendiğinde gerçekleştirilecek işlemler burada tanımlanır.
    - Operations bloğundaki Add butonuna tıklanır.
    - **Operation details:**

- **Operation type:** Genellikle "Send message" seçilir. (Diğer seçenekler: Remote command vb.)
- **Send to User groups:** Bildirimin gönderileceği kullanıcı grupları seçilir. (Örn: "Zabbix administrators")
- **Send to Users:** Belirli kullanıcılara bildirim gönderilecekse buradan seçilir.
- **Send to Media Type:** Kullanılacak olan media typelar seçilir.
- **Custom message:** İşaretli ise, Media Type'ta tanımlanan varsayılan mesaj şablonları kullanılmaz. İşaret kaldırılırsa, bu aksiyonun default bir "Subject" ve "Message" döner.
  - **Subject (isteğe bağlı):** PROBLEM: {TRIGGER.NAME}
  - **Message (isteğe bağlı):**

```
Problem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME} ({HOST.IP})
Severity: {TRIGGER.SEVERITY}

Item: {ITEM.NAME}
Value: {ITEM.VALUE}

Original event ID: {EVENT.ID}
```

  - Bu mesajlarda çeşitli Zabbix makroları kullanılabilir.

- **Default operation step duration:** Bildirimlerin ne kadar süreyle ve hangi aralıklarla gönderileceği ayarlanabilir. Örneğin her 1 saatte bir, toplam 5 kez gibi.

4. **Recovery Operations Sekmesi (Kurtarma İşlemleri):**

- Sorun çözüldüğünde (Trigger durumu "OK" olduğunda) bildirim gönderilmesi isteniyorsa bu sekmede benzer bir operasyon tanımlanır.
- Add butonuna tıklanır.
- **Operation details:**

- **Operation type:** "Notify all involved" veya "Send message" seçilebilir.
- **Send to User groups:** Bildirimin gönderileceği kullanıcı grupları seçilir. (Örn: "Zabbix administrators")
- **Send to Users:** Belirli kullanıcılara bildirim gönderilecekse buradan seçilir.
- **Send to Media Type:** Kullanılacak olan media typelar seçilir.
- **Custom message:** İşaretli ise, Media Type'ta tanımlanan varsayılan mesaj şablonları kullanılmaz. İşaret kaldırılırsa, bu aksiyonun default bir "Subject" ve "Message" döner.
    - **Subject (isteğe bağlı):** RESOLVED: {TRIGGER.NAME}
    - **Message (isteğe bağlı):**

```
Problem RESOLVED at {EVENT.RECOVERY.TIME} on {EVENT.RECOVERY.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME} ({HOST.IP})
Severity: {TRIGGER.SEVERITY}

Original event ID: {EVENT.ID}
```

5. **Update Operations Sekmesi (Güncelleme İşlemleri - İsteğe Bağlı):**
    - Bir problemle ilgili güncelleme olduğunda (örneğin, manuel olarak kapatıldığında, yorum eklendiğinde veya önem derecesi değiştirildiğinde) bildirim gönderilmesi için kullanılır. Genellikle Telegram bildirimleri için daha az kullanılır.
6. Tüm yapılandırmalar tamamlandıktan sonra Add (ilk defa ekleniyorsa) veya Update (eklenmiş olan düzenleniyorsa) butonuna tıklanarak aksiyon kaydedilir.

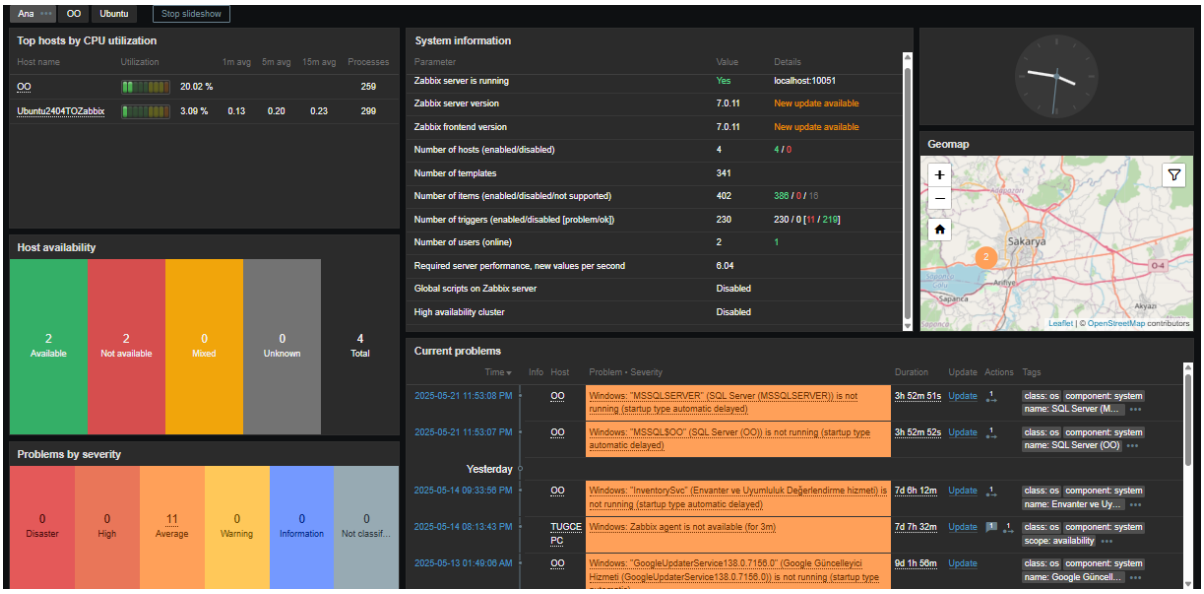
4.4. Dashboard Oluşturma ve Kişiselleştirme

Dashboard, Zabbix'te izlenen verilerin ve sistem durumunun grafikler, tablolar, haritalar ve diğer widget'lar aracılığıyla görselleştirildiği merkezi bir arayüzdür. Dashboard'lar, sistem

yöneticilerinin altyapının genel sağlığını hızlıca anlamalarına ve potansiyel sorunları kolayca tespit etmelerine yardımcı olur.

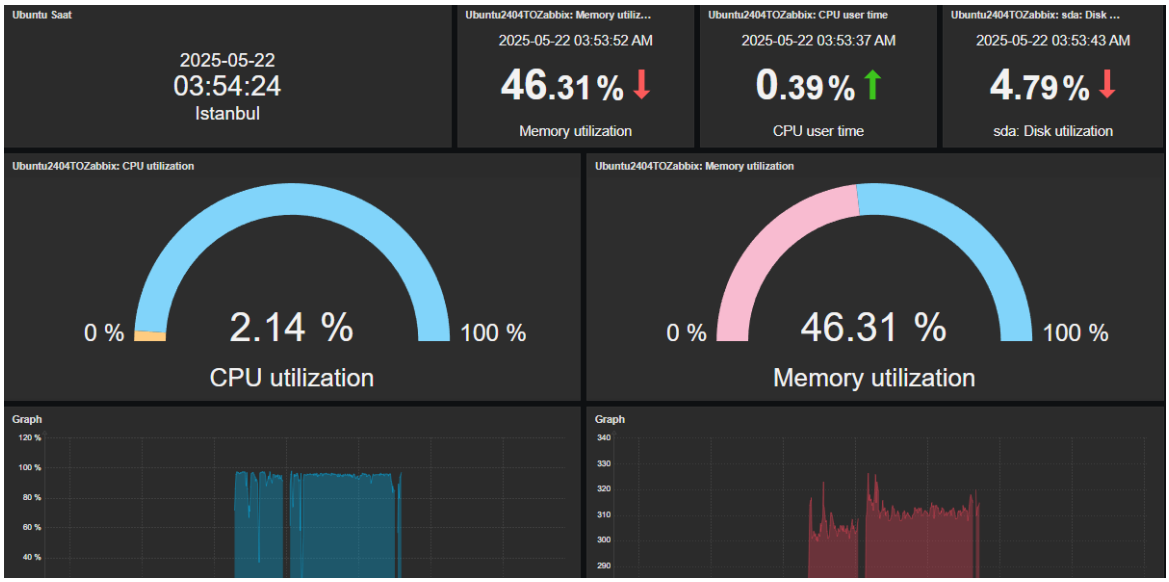
Dashboard Oluşturma ve Kişiselleştirme Adımları:

1. **Dashboards** bölümüne gidilir. Zabbix'te varsayılan bir genel dashboard bulunur.



2. **Yeni Dashboard Oluşturma:**
    - Sağ üst köşedeki **All dashboards** butonuna tıklanır.
    - **Create dashboard** seçeneği ile yeni bir dashboard oluşturulur.
    - Dashboard'a bir **Name** verilir ve **Owner** belirlenir. Kullanıcılar veya kullanıcı grupları için erişim izinleri ayarlanabilir.
3. **Widget Ekleme ve Yapılandırma:**
    - Oluşturulan veya mevcut bir dashboard açılır.
    - Sağ üst köşedeki **Edit dashboard** butonuna tıklanarak düzenleme moduna geçilir.
    - **Add widget** butonuna tıklanarak eklenebilecek widget türleri listelenir. Başlıca widget türleri:

    - **Graph:** Bir veya daha fazla item'ın zaman içindeki değişimini gösteren grafikler.
    - **Plain text:** Item'ların son değerlerini metin olarak gösterir.
    - **Clock:** Saati gösterir.
    - **Problem hosts:** Sorunlu host'ları listeler.
    - **Problems:** Aktif problemleri önem derecesine göre listeler.
    - **System information:** Zabbix sunucusu hakkında bilgi verir.
    - **Data overview:** Seçilen item grupları için genel bakış sağlar.
    - **Action log:** Gerçekleşen aksiyonları gösterir.
    - **Map:** Ağ haritalarını gösterir.
    - **URL:** Harici bir web sayfasını gömer.
- Eklenmek istenen widget türü seçilir.
- Widget'ın ayarları yapılandırılır:
    - **Source:** Verinin hangi host'tan veya item'dan alınacağı.
    - **Display options:** Gösterim şekli, renkler, periyot vb.
    - **Refresh interval:** Widget'ın ne sıklıkta güncelleneceği.
- Widget'lar dashboard üzerinde sürükle-bırak yöntemiyle yeniden boyutlandırılabilir ve konumlandırılabilir.



4. **Dashboard'u Kaydetme:**

    - Düzenlemeler tamamlandıktan sonra sağ üstteki **Save changes** butonuna tıklanarak dashboard kaydedilir.

Bu projede, özellikle izlenen sunuculara (örneğin, Ubuntu sanal makinesi) özel ayrı dashboard sayfaları oluşturulmuştur. Bu dashboard'larda, her sunucunun CPU kullanımı, bellek durumu, disk alanı, ağ trafiği gibi kritik metrikleri grafiksel olarak ve anlık değerlerle gösterilmiştir. Ayrıca, aktif problemlerin ve son uyarıların listelendiği widget'lar eklenerek sistemin genel sağlık durumu tek bir bakışta anlaşılır hale getirilmiştir. Bu kişiselleştirilmiş dashboard'lar, sorun tespitini ve müdahale süreçlerini hızlandırmaya katkı sağlamıştır.

---

5. Test Aşaması ve Sonuçları

Bu bölümde, kurulan Zabbix izleme sisteminin ve Telegram entegrasyonunun işlevselliğini doğrulamak amacıyla gerçekleştirilen testler ve elde edilen sonuçlar detaylandırılmaktadır.

5.1. Testin Amacı

Bu testin temel amacı, aşağıdaki süreçlerin uçtan uca doğru bir şekilde çalışıp çalışmadığını teyit etmektir:

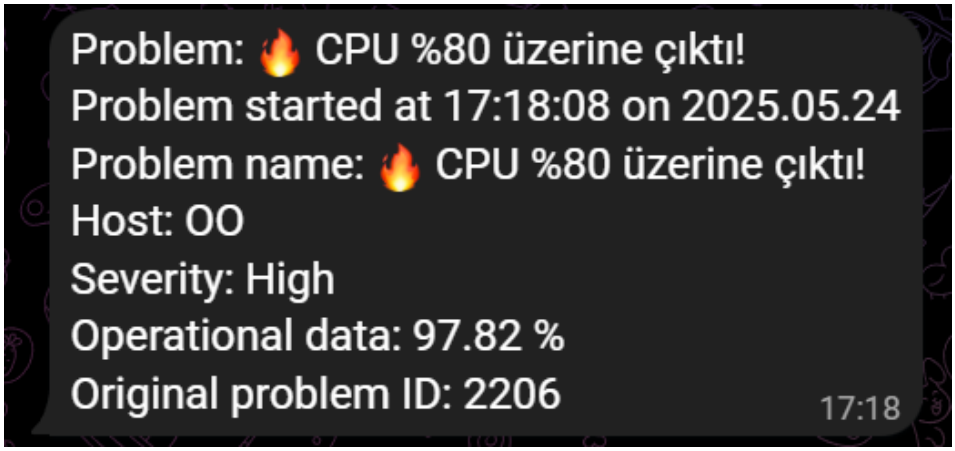
- İzlenen bir sistem metriğinde (CPU kullanımı) anormal bir durumun Zabbix tarafından tespiti.
- Tanımlanmış trigger koşullarının karşılanmasıyla bir "PROBLEM" olayının oluşturulması.
- PROBLEM olayı üzerine yapılandırılmış aksiyonun tetiklenmesi.
- Aksiyon sonucunda Telegram botu aracılığıyla ilgili kişilere anlık bildirim gönderilmesi.
- Sorunun ortadan kalkması durumunda "OK" (kurtarma) olayının oluşturulması ve kurtarma bildiriminin (eğer yapılandırıldıysa) gönderilmesi.

5.2. Test Ortamı ve Senaryosu

- **Test Edilen Sistem:** Zabbix ajanı kurulu olan bir Windows [Windows Sürümünüzü Belirtin, örn: Windows 10 Pro] işletim sistemine sahip test makinesi.
- **İzleme Sistemi:** Ubuntu 24.04 üzerinde çalışan Zabbix 7.0 LTS sunucusu.
- **Bildirim Kanalı:** Önceden yapılandırılmış Telegram botu.
- **Test Senaryosu:** Windows test makinesinde CPU kullanımının yapay olarak %90'ın üzerine çıkarılması. Zabbix'te bu durum için "CPU Kullanımı 5 dakika boyunca ortalama %90'ın üzerinde" (örnek trigger koşulu, siz kendi koşulunuzu yazın) şeklinde bir trigger tanımlanmıştır. Bu trigger tetiklendiğinde, Telegram üzerinden "Yüksek CPU Kullanımı" uyarısı alınması ve CPU yükü normale döndüğünde "Sorun Çözüldü" bildirimi alınması beklenmektedir.

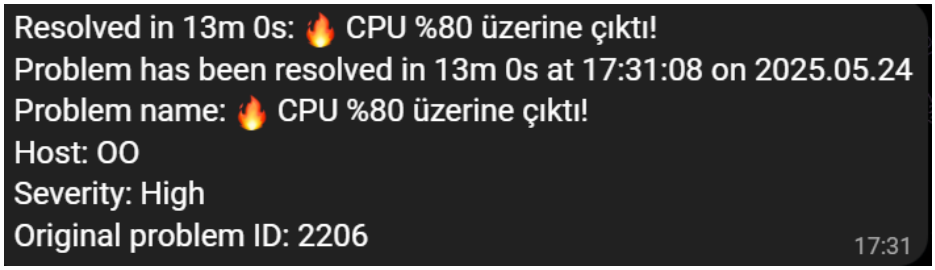
5.3. Test Adımları ve Gözlemler

1. **CPU Yükünün Artırılması:** Windows test makinesinde, tüm mantıksal işlemcileri hedef alarak yoğun hesaplama yapan özel bir PowerShell script'i (cpu_stress.ps1) çalıştırılmıştır.
2. **Zabbix'te Metrik Takibi:** Script çalıştırıldıktan hemen sonra, Zabbix arayüzünde (Monitoring > Latest data veya ilgili host'un grafikleri üzerinden) Windows makinesinin CPU kullanım metriğinin hızla %95-%100 seviyelerine ulaştığı gözlemlenmiştir.
3. **Trigger'ın Tetiklenmesi:** Tanımlanan trigger koşuluna uygun olarak, yaklaşık [örneğin: 2 dakika] sonra Zabbix arayüzünde (Monitoring > Problems) "Yüksek CPU Kullanımı - [Windows_Host_Adı]" başlıklı yeni bir problem olayının "High" (Yüksek) önem seviyesinde oluştuğu teyit edilmiştir.
4. **Aksiyonun Çalışması ve Telegram Bildirimi:**
    - Problem olayının oluşmasıyla eş zamanlı olarak, Alerts > Action log bölümünde Telegram bildirimi göndermek üzere tanımlanan aksiyonun başarıyla çalıştırıldığı görülmüştür.
    - Saniyeler içerisinde, Telegram botu aracılığıyla test kullanıcısının Telegram hesabına aşağıdaki gibi bir bildirim mesajı ulaşmıştır:



5. **CPU Yükünün Normale Döndürülmesi:** Telegram bildirimi başarıyla alındıktan sonra, Windows makinesindeki cpu_stress.ps1 script'i sonlandırılmıştır. CPU kullanımının Zabbix arayüzünde hızla normal seviyelere (%5-15 aralığına) düştüğü gözlemlenmiştir.

6. **Kurtarma (Recovery) Durumu:** CPU yükü normale döndükten yaklaşık [örneğin: 1 dakika] sonra, Zabbix arayüzünde ilgili problemin çözüldüğü (Monitoring > Problems listesinden kalktığı veya "Resolved" olarak işaretlendiği) görülmüştür.
7. **Telegram Kurtarma Bildirimi:** Eş zamanlı olarak, Telegram botu aracılığıyla aşağıdaki gibi bir kurtarma mesajı alınmıştır:



5.4. Test Sonucu

Gerçekleştirilen test başarıyla tamamlanmıştır. Zabbix izleme sistemi, Windows test makinesindeki yapay olarak artırılan CPU yükünü doğru bir şekilde tespit etmiş, tanımlanan trigger'ı zamanında tetiklemiş ve yapılandırılan aksiyon aracılığıyla Telegram üzerinden anlık bildirimleri (hem problem hem de kurtarma için) başarıyla göndermiştir.

Bu test, Zabbix sunucusunun, Zabbix ajanının, trigger ve aksiyon mekanizmalarının yanı sıra Telegram entegrasyonunun da beklendiği gibi çalıştığını doğrulamıştır. Sistem, projenin amaçlarına uygun olarak kritik durumları izleyip anlık uyarı üretebilmektedir.

---

6. Sonuç

Bu proje kapsamında Oracle VirtualBox üzerinde çalışan Ubuntu 24.04 sanal makinesine Zabbix 7.0 LTS başarıyla kurulmuş ve yapılandırılmıştır. Zabbix'in güçlü izleme yetenekleri kullanılarak sistemdeki kritik olayların tespiti için trigger'lar tanımlanmıştır. En önemli çıktılardan biri, tespit edilen bu olayların Telegram botu aracılığıyla anlık olarak sistem yöneticilerine bildirilmesini sağlayan webhook entegrasyonunun başarılı bir şekilde gerçekleştirilmesidir. Ayrıca, izlenen sunuculara özel, kişiselleştirilmiş dashboard'lar

oluşturularak sistem metriklerinin ve uyarılarının etkili bir şekilde görselleştirilmesi sağlanmıştır. Bu sayede, sistemin genel sağlık durumu hakkında hızlıca bilgi edinmek ve olası sorunlara proaktif bir şekilde müdahale etmek mümkün hale gelmiştir.

Proje, Zabbix gibi kapsamlı bir izleme aracının kurulumu, temel yapılandırması, harici sistemlerle entegrasyonu ve veri görselleştirme konularında pratik deneyim kazandırmıştır. Elde edilen sonuçlar, projenin başında belirlenen amaç ve hedeflere ulaşıldığını göstermektedir.

Kaynakça

[1] “Zabbix,” Zabbix. Accessed: May 22, 2025. [Online]. Available: https://www.zabbix.com/download?zabbix=7.0&os_distribution=ubuntu&os_version=24.04&components=server_frontend_agent&db=mysql&ws=apache