T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ


**Bilişim Sistemleri Mühendisliği Yüksek Lisans**

**Bulut Bilişim Ödev**

# Prometheus Tabanlı İzleme Sistemi: Kurulum, Mimari Analiz ve Teknik Değerlendirme

**Y255052063– Serdar ARICI**

**Fakülte Anabilim Dalı : Bilişim Sistemleri Mühendisliği**

**Dersi Veren : Dr.Öğr.Üyesi Baran KAYNAK**

**2025-2026 Bahar Dönemi**

# 1. Giriş ve Motivasyon

Modern bulut bilişim ekosistemi, uygulamaların taşınabilirliğini ve ölçeklenebilirliğini artırmak amacıyla konteyner tabanlı mikro hizmet mimarilerine doğru evrilmiştir. Bu evrimin sürdürülebilirliğini sağlayan en temel yapı taşı ise 2015 yılında kurulan ve bulut-yerel teknolojilerin standartlarını belirleyen Cloud Native Computing Foundation (CNCF) olmuştur. CNCF, karmaşık dağıtık sistemlerin yönetimi için gerekli olan açık kaynaklı projeleri himayesine alarak bir ekosistem güvenliği sağlar. Bu ekosistem içerisinde, Kubernetes'ten sonra topluluğun güvenini kazanarak "Graduated" (mezun) seviyesine ulaşan ilk projelerden biri olan Prometheus, modern gözlemlenebilirlik (observability) stratejilerinin merkezinde yer almaktadır. İlk olarak SoundCloud tarafından, dinamik mikro hizmet trafiklerini ve performans metriklerini geleneksel araçların aksine daha esnek bir şekilde takip edebilmek amacıyla geliştirilen Prometheus, bugün bulut tabanlı altyapıların fiili standart izleme çözümü haline gelmiştir.

"Modern bulut bilişim ekosistemi, uygulamaların taşınabilirliğini ve ölçeklenebilirliğini artırmak amacıyla konteyner tabanlı mikro hizmet mimarilerine doğru evrilmiştir. Bu evrimin sürdürülebilirliğini sağlayan en temel yapı taşı ise 2015 yılında kurulan ve bulut-yerel teknolojilerin standartlarını belirleyen Cloud Native Computing Foundation (CNCF) olmuştur. CNCF, karmaşık dağıtık sistemlerin yönetimi için gerekli olan açık kaynaklı projeleri himayesine alarak bir ekosistem güvenliği sağlar. Bu ekosistem içerisinde, Kubernetes'ten sonra topluluğun güvenini kazanarak "Graduated" (mezun) seviyesine ulaşan ilk projelerden biri olan Prometheus, modern gözlemlenebilirlik (observability) stratejilerinin merkezinde yer almaktadır. İlk olarak SoundCloud tarafından, dinamik mikro hizmet trafiklerini ve performans metriklerini geleneksel araçların aksine daha esnek bir şekilde takip edebilmek amacıyla geliştirilen Prometheus, bugün bulut tabanlı altyapıların fiili standart izleme çözümü haline gelmiştir.

Geleneksel izleme sistemleri, genellikle statik sunucu yapılandırmalarına ve sınırlı veri modellerine dayanırken; Prometheus, her bir veri noktasını metrik adı ve anahtar-değer çiftlerinden oluşan etiketlerle (labels) tanımlayarak çok boyutlu bir veri modeli sunar. Bu tasarım felsefesi, binlerce mikro hizmetin saniyeler içinde ölçeklendiği Kubernetes gibi ortamlarda, verinin hangi servisten, hangi poddan veya hangi bölgeden geldiğini anlık olarak ayırt etmeyi mümkün kılar. Sistemin en ayırt edici özelliklerinden biri olan "pull-based" (çekme odaklı) veri toplama mimarisi, izlenen hedeflerin üzerine ekstra bir ajan yükü bindirmeden metriklerin merkezi olarak toplanmasına olanak tanır. Ayrıca, kendine has güçlü sorgulama dili olan PromQL, kullanıcılara ham veriler üzerinde karmaşık matematiksel operasyonlar ve eşik analizi yapma gücü vererek, sadece bir izleme aracı olmanın ötesinde, proaktif bir hata tespit mekanizması sağlar.

Bu çalışmanın temel motivasyonu, CNCF ekosisteminin bu denli kritik bir parçası olan Prometheus'un, karmaşık bulut yapılarındaki performans darboğazlarını nasıl teşhis ettiğini ve dinamik servis keşfi (service discovery) yetenekleriyle operasyonel yükü nasıl hafiflettiğini teknik bir perspektifle ortaya koymaktır. Çalışma boyunca sistemin iç mimarisi

detaylandırılacak, kurulum süreçleri üzerinden elde edilen deneyimler paylaşılacak ve güncel alternatifleri ile kıyaslanarak modern sistem yöneticileri için sunduğu avantajlar değerlendirilecektir.” [1]

"Bulut ortamlarında izleme, sistemin yüksek erişilebilirliğini ve performansını sürdürebilmek için kritik öneme sahiptir ve hem sağlayıcılar hem de kullanıcılar açısından önemlidir. Öncelikle izleme, yazılım ve donanım kaynaklarının yönetilmesi ile bu kaynaklar ve bulut üzerinde barındırılan kullanıcı uygulamaları hakkında sürekli bilgi sağlanması için temel bir araçtır. Kaynak planlama, kaynak yönetimi, veri merkezi yönetimi, hizmet seviyesi anlaşması (SLA) yönetimi, faturalandırma, sorun giderme, performans yönetimi ve güvenlik yönetimi gibi bulut faaliyetleri, sistemin etkin ve sorunsuz çalışabilmesi için izlemeye ihtiyaç duyar. Bu nedenle, bulut bilişimin esnek yapısı göz önüne alındığında izlemeye duyulan ihtiyaç oldukça fazladır.

Bulut bilişimde izleme iki türde gerçekleştirilebilir: yüksek seviye ve düşük seviye. Yüksek seviye izleme, sanal platformun durumuna odaklanırken, düşük seviye izleme fiziksel altyapının durumuna ilişkin toplanan bilgileri kapsar. Bulut izleme sistemi, kendini ayarlayabilen ve genellikle çok iş parçacıklı bir yapıya sahip olup izleme işlevlerini destekleyebilmektedir. Bu sistem, bulut üzerindeki önceden tanımlanmış örnekleri veya kaynakları anormallikler açısından kapsamlı bir şekilde izler. Anormal bir durum tespit edildiğinde, ilgili izleyicide otomatik iyileştirme tanımlıysa sistem bu örnek veya kaynağı otomatik olarak onarmaya çalışır. Otomatik onarımın başarısız olması ya da böyle bir mekanizmanın bulunmaması durumunda ise destek ekibine bildirim gönderilir. Bu bildirimler teknik olarak e-posta veya SMS gibi farklı yöntemlerle iletilebilir."[2]

Geleneksel izleme sistemleri, genellikle statik sunucu yapılandırmalarına ve sınırlı veri modellerine dayanırken; Prometheus, her bir veri noktasını metrik adı ve anahtar-değer çiftlerinden oluşan etiketlerle (labels) tanımlayarak çok boyutlu bir veri modeli sunar. Bu tasarım felsefesi, binlerce mikro hizmetin saniyeler içinde ölçeklendiği Kubernetes gibi ortamlarda, verinin hangi servisten, hangi poddan veya hangi bölgeden geldiğini anlık olarak ayırt etmeyi mümkün kılar. Sistemin en ayırt edici özelliklerinden biri olan "pull-based" (çekme odaklı) veri toplama mimarisi, izlenen hedeflerin üzerine ekstra bir ajan yükü bindirmeden metriklerin merkezi olarak toplanmasına olanak tanır. Ayrıca, kendine has güçlü sorgulama dili olan PromQL, kullanıcılara ham veriler üzerinde karmaşık matematiksel operasyonlar ve eşik analizi yapma gücü vererek, sadece bir izleme aracı olmanın ötesinde, proaktif bir hata tespit mekanizması sağlar.

Bu çalışmanın temel motivasyonu, CNCF ekosisteminin bu denli kritik bir parçası olan Prometheus’un, karmaşık bulut yapılarındaki performans darboğazlarını nasıl teşhis ettiğini ve dinamik servis keşfi (service discovery) yetenekleriyle operasyonel yükü nasıl hafiflettiğini teknik bir perspektifle ortaya koymaktır. Çalışma boyunca sistemin iç mimarisi detaylandırılacak, kurulum süreçleri üzerinden elde edilen deneyimler paylaşılacak ve güncel alternatifleri ile kıyaslanarak modern sistem yöneticileri için sunduğu avantajlar değerlendirilecektir.

## 2. Mimari Analiz

"Sistem izleme mimarileri, modern BT altyapısının sorunsuz çalışması açısından kritik öneme sahiptir. Bu mimariler, karmaşık sistemlerin gerçek zamanlı olarak izlenmesini sağlayarak kuruluşların hizmet performansını ve güvenilirliğini korumasına yardımcı olur. Bulut bilişim, mikroservisler ve dağıtık sistemlerin giderek daha fazla kullanılmasıyla BT ortamlarının daha karmaşık hale gelmesi, izlemeyi daha da önemli bir hale getirmiştir. İzleme, performans darboğazlarının, güvenlik açıklarının ve operasyonel sorunların ciddi problemlere yol açmadan önce tespit edilmesini ve giderilmesini sağlar. Güçlü bir izleme sistemi olmadan, kuruluşlar sistem arızaları, güvenlik ihlalleri ve kullanıcı memnuniyetinde azalma riskiyle karşı karşıya kalır; bu durum önemli finansal ve itibari kayıplara yol açabilir.

Ayrıca, etkin bir izleme yaklaşımı sorun giderme süreçlerini iyileştirir, kaynak tahsisini optimize eder ve proaktif bakım faaliyetlerini kolaylaştırır. Günümüzün rekabetçi ortamında işletmeler kesintisiz çalışma süresi, sorunsuz performans ve hızlı sorun çözümü talep etmektedir. Bu nedenle, sistem izleme araçları bu hedeflerin gerçekleştirilmesinde vazgeçilmezdir. İzleme mimarileri, BT ortamlarının çevikliğine ve güvenilirliğine katkı sağlayarak ekiplerin veri odaklı ve bilinçli kararlar almasına olanak tanır.

Etkili sistem izleme, çeşitli araç ve teknikleri kapsar. Prometheus, ELK Stack ve özel gösterge panelleri gibi izleme çözümlerinin entegrasyonu, kuruluşların kapsamlı bir izleme altyapısı oluşturması açısından büyük önem taşır. Bu araçlar, görünürlük, gerçek zamanlı metrikler ve uygulanabilir içgörüler sağlayarak iş karar alma süreçlerini iyileştirir."[3]

Genel izleme mimarilerinin sağladığı bu geniş perspektif, operasyonel süreklilik için gerekli olan altyapıyı tanımlasa da, bulut bilişimin getirdiği dinamik ve ölçeklenebilir yapı bu ihtiyaçların çok daha spesifik araçlarla karşılanmasını zorunlu kılmıştır. Geleneksel yöntemlerin statik kalması ve mikroservis mimarilerinin karmaşıklığı karşısında yetersiz kalması, izleme teknolojilerinde bir paradigma değişimine yol açmıştır. Tam da bu noktada, Cloud Native Computing Foundation (CNCF) çatısı altında olgunlaşan teknolojiler, bu karmaşıklığı yönetilebilir hale getirmek için yeni standartlar belirlemiştir. Bu standartların merkezinde yer alan ve modern altyapıların "altın standardı" olarak kabul edilen çözüm ise Prometheus'tur. Genel izleme disiplinini pratik ve ölçeklenebilir bir gerçekliğe dönüştüren Prometheus'un önemi, akademik literatürde şu şekilde vurgulanmaktadır:

“Modern bulut bilişim ekosisteminde Prometheus, zaman serisi veri toplama, uyarı mekanizmaları ve gelişmiş sorgulama yetenekleriyle bulut izleme süreçleri için kritik bir araç haline gelmiştir. Yüksek frekanslı veri akışlarını yönetmek üzere tasarlanan bu platform; bulut ortamlarında hem operasyonel verimliliği hem de güvenliği artıran kapsamlı bir izleme altyapısının temelini oluşturmaktadır. Manuel ayarlamalara dayanan geleneksel izleme sistemlerinin aksine Prometheus; benzeri görülmemiş bir ölçekte çalışan ve sürekli değişkenlik gösteren bulut sistemleri için elzem olan metrik toplama ve uyarı süreçlerini otomatikleştirir. "Çekme" (pull-based) modelini kullanan Prometheus, sistem yöneticilerinin

dağıtık kaynaklardan metrik toplamasını sağlayarak sistem performansı ve sağlığı hakkında bütünsel bir genel bakış sunar.

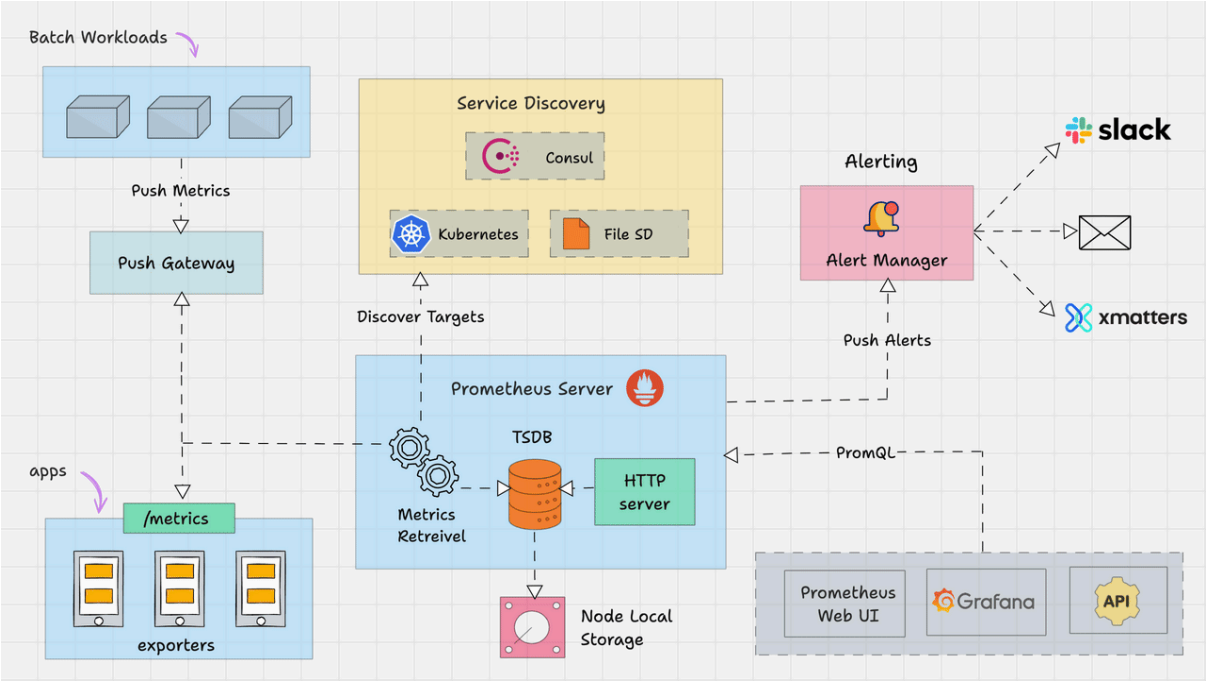
Prometheus'u rakiplerinden ayıran en temel özelliklerden biri, yüksek çözünürlüklü veri depolama için optimize edilmiş olan Zaman Serisi Veritabanı (TSDB) yapısıdır. Bu yapı; organizasyonların eğilimleri analiz etmesine ve anomalileri yüksek bir hassasiyetle tespit etmesine olanak tanır. TSDB mimarisi, ekiplerin Temel Performans Göstergelerini (KPI) zaman içinde izlemelerine ve potansiyel sorunları operasyonel bir krize dönüşmeden önce proaktif olarak tanımlamalarına yardımcı olur. Literatürde bu yeteneğin, hızlı işlem süreçlerinin ve operasyonel istikrarın hayati önem taşıdığı finansal hizmetler gibi sektörlerde özellikle değerli olduğu vurgulanmaktadır. Yüksek çözünürlüklü metrik kayıtları tutarak performans metriklerinin detaylı analizine imkan tanıyan sistem, sorunlara erken aşamada müdahale edilmesini kolaylaştırır.

Adaptasyon yeteneği açısından Prometheus, çeşitli bulut servisleri ve sistemleriyle entegre olma konusunda üstün bir performans sergilemektedir. Bu esneklik, karmaşık çoklu bulut (multi-cloud) mimarilerini yöneten organizasyonlar için kritik bir araç niteliğindedir. Özellikle veri izleme gereksinimlerinin büyük ölçüde değiştiği finans veya sağlık gibi operasyonel ortamlarda bu uyumluluk hayati önem taşır. Geniş "exporter" (dışa aktarıcı) ekosistemi sayesinde Prometheus; veritabanlarından konteynerleştirilmiş ortamlara kadar uzmanlaşmış uygulamaların izlenmesine olanak tanıyarak farklı sektörlerdeki operasyonel ihtiyaçlara yanıt verir."[4]

Prometheus'un mimari yapısı, modern bulut-yerel sistemlerin ihtiyaç duyduğu yüksek ölçeklenebilirlik ve esneklik prensipleri üzerine inşa edilmiştir. Geleneksel izleme araçlarının aksine Prometheus, metrik toplama sürecinde "pull-based" (çekme tabanlı) bir model benimser. Bu modelde sistem, izlenecek hedeflerin kendisine veri göndermesini beklemek yerine, önceden tanımlanmış servis keşif mekanizmaları aracılığıyla hedefleri bulur ve belirli aralıklarla HTTP üzerinden metrik verilerini "kazır" (scraping). Bu yaklaşım, ağ üzerindeki karmaşıklığı azaltır ve izlenen uygulama üzerinde minimum performans yükü oluşturur. Prometheus mimarisini oluşturan temel bileşenler ve işlevleri aşağıda detaylı olarak ele alınmıştır:

- Prometheus Server (Ana Sunucu): Sistemin beyni olarak kabul edilen bu bileşen, üç temel alt birimden oluşur. İlk olarak Retrieval birimi, hedef sistemlerden HTTP üzerinden metrik verilerini toplar. İkinci olarak TSDB (Time Series Database), toplanan bu verileri zaman damgalarıyla birlikte yüksek sıkıştırma oranlarıyla depolar. Son olarak HTTP Server birimi, PromQL sorgularını işleyerek dış sistemlere (Grafana gibi) veya doğrudan kullanıcı arayüzüne veri sunar.
- Exporters (Dışa Aktarıcılar): Prometheus'un doğrudan metrik yayınlamayan (örneğin Linux çekirdeği, MySQL, Apache) üçüncü taraf servislerle konuşmasını sağlayan elçilerdir. Exporter'lar, ilgili servisin durumunu Prometheus'un anlayabileceği metrik formatına dönüştürerek bir HTTP endpoint'i üzerinden yayınlarlar.

- Pushgateway: Kısa ömürlü (ephemeral) işler ve "batch" operasyonlar için tasarlanmış bir ara birimdir. "Pull" modeline uymayan bu kısa süreli işlemler, metriklerini Pushgateway'e gönderir ve Prometheus sunucusu verileri buradan toplu olarak çeker.
- Alertmanager (Alarm Yönetimi): Prometheus üzerinde tanımlanan alarm kuralları (alert rules) tetiklendiğinde, bu uyarılar Alertmanager bileşenine iletilir. Bu birim; gelen uyarıları gruplandırır, sessize alma (silencing) işlemlerini yönetir ve nihayetinde e-posta, Slack veya PagerDuty gibi kanallar aracılığıyla ilgili ekipleri bilgilendirir.
- Time Series Database (TSDB): Prometheus'un özgün depolama motorudur. Her veri noktasını bir zaman damgası ve isteğe bağlı etiketlerle (labels) saklar. Bu etiketleme sistemi, verilerin çok boyutlu olarak analiz edilmesine (örneğin; aynı metriği farklı servisler veya bölgeler bazında sorgulama) olanak tanır.




Şekil 1: Prometheus Mimari Diyagramı

Yukarıda yer alan Prometheus Mimari Diyagramı, bu bileşenlerin birbirleriyle olan dinamik etkileşimini ve verinin sistem içerisindeki akış yolunu görsel olarak ortaya koymaktadır. Diyagramda görüldüğü üzere Prometheus, metrik toplama aşamasından görselleştirme ve alarm aşamasına kadar uçtan uca, kendi kendine yetebilen modüler bir yapı sunmaktadır. Bu tasarım, sistemin hata toleransını artırırken, karmaşık mikro hizmet ağlarının gözlemlenebilirliğini maksimize etmektedir.

## 3. Kurulum ve Deneyim

Bu bölümde, Prometheus izleme sisteminin yerel bir ortamda Docker konteyner teknolojisi kullanılarak gerçekleştirilen kurulum süreci, konfigürasyon detayları ve elde edilen ilk deneyimler akademik bir perspektifle sunulmuştur. Kurulum süreci, sistemin taşınabilirliğini

artırmak ve bağımlılıkları minimize etmek amacıyla Docker ve Docker Compose araçları üzerinden yürütülmüştür.

### 3.1. Kurulum Ortamı ve Hazırlıklar

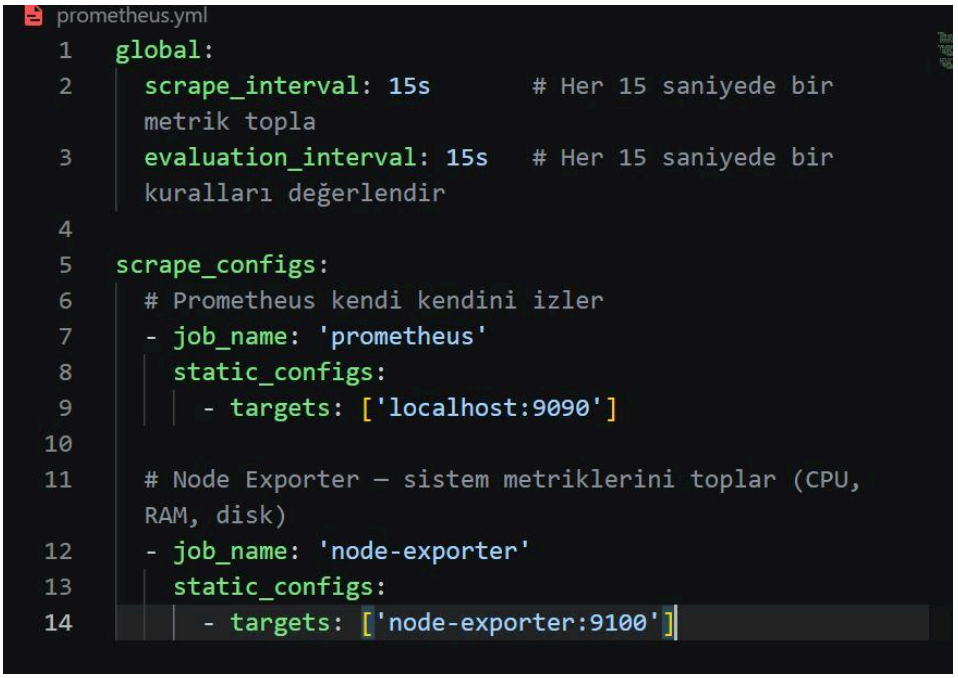
Kurulumun gerçekleştirildiği sistem özellikleri ve kullanılan araçlar şu şekildedir:

- İşletim Sistemi: Windows.
- Konteynerleştirme: Docker Engine v20.x ve Docker Compose v2.x.
- İzleme Hedefi: Node Exporter (İşletim sistemi metriklerini çekmek için).

### 3.2. Adım Adım Kurulum Süreci

Adım 1: Proje Dizininin Oluşturulması
Öncelikle projenin tüm konfigürasyon dosyalarını barındıracak düzenli bir dizin yapısı oluşturulmuştur.

Adım 2: Prometheus Konfigürasyon Dosyasının Yazılması (prometheus.yml)
Prometheus'un hangi hedefleri (targets) izleyeceğini bildiren temel ayar dosyası oluşturulmuştur. Bu dosyada scrape_interval değeri 15 saniye olarak belirlenmiş, böylece verilerin güncelliği ile sistem yükü arasında bir denge kurulmuştur.

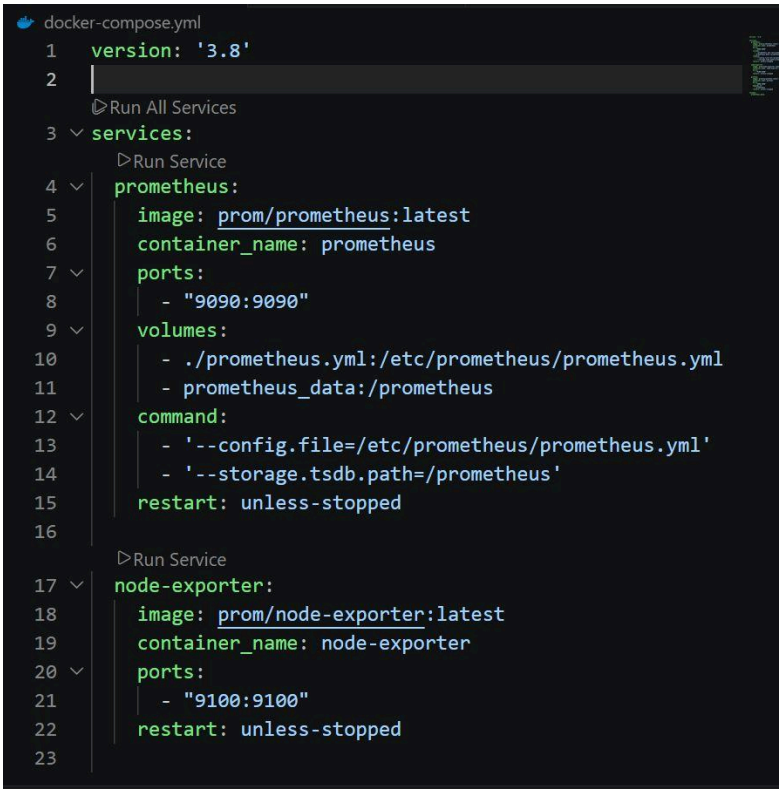



Şekil 2: prometheus.yml

Adım 3: Docker Compose Yapılandırması (docker-compose.yml)
Hazırlanan docker-compose.yml dosyası, izleme sisteminin taşınabilirliğini sağlayan üç ana servisi bir araya getirmektedir:

- Prometheus Servisi: Sistemin kalbi olan bu birim, metrikleri toplar ve saklar.
    - Port 9090: Web arayüzüne erişim sağlar.

    - Volumes: Yapılandırma dosyası (prometheus.yml) dışarıdan bağlanarak esneklik sağlanmış, prometheus_data birimi ile verilerin kalıcılığı (persistence) garanti altına alınmıştır.
    - Komutlar: Depolama yolu ve konfigürasyon dosyasının yeri açıkça belirtilerek sistem optimize edilmiştir.
- Node-Exporter Servisi: Ana makinenin donanım metriklerini (CPU, RAM vb.) Prometheus'un anlayabileceği formata dönüştürerek 9100 portundan yayınlar.
- Grafana Servisi: Toplanan verileri görselleştirmek için kullanılır.
    - depends_on: Grafana'nın çalışmak için Prometheus servisine bağımlı olduğu belirtilerek servisler arası hiyerarşi kurulmuştur.
    - Port 3000: Kullanıcıların dashboard'lara erişebileceği arayüz portudur.
- Genel Yapılandırma: Tüm servisler unless-stopped politikasıyla çalıştırılarak, sistem çökmelerine karşı proaktif bir dayanıklılık kazandırılmıştır.



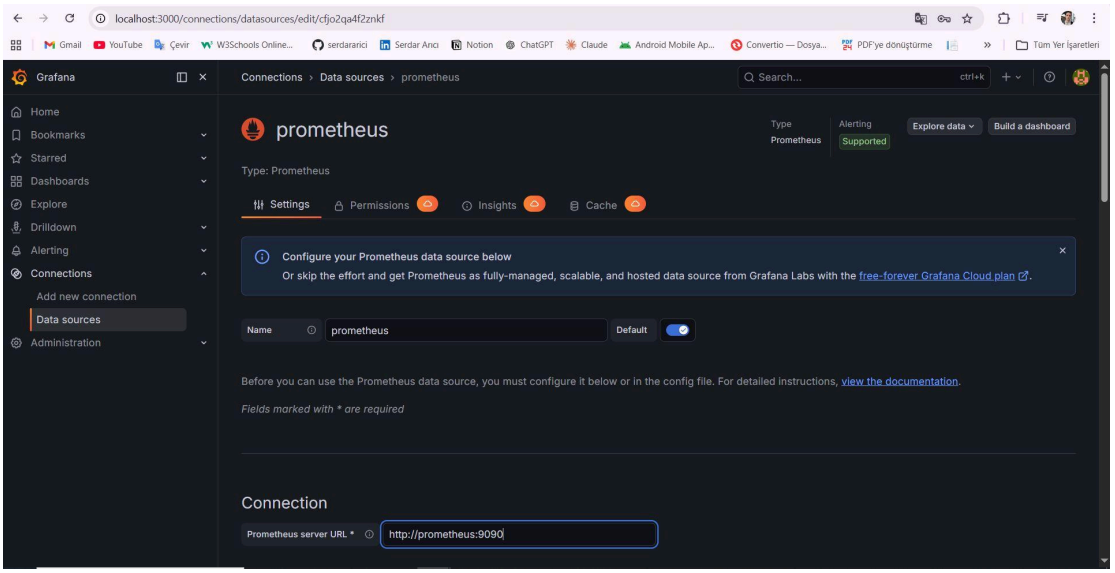

Şekil 3: docker-compose.yml

### 3.3. Karşılaşılan Sorunlar ve Teknik Çözümler

Kurulum aşamasında karşılaşılan sorunlar ve bu sorunların akademik çözüm yaklaşımları aşağıda raporlanmıştır:

1. Erişim ve İzin Sorunları (Permission Denied): Prometheus konteyneri, konfigürasyon dosyasını okurken yetki hatası vermiştir. Çözüm: chown komutu ile dosya izinleri Prometheus kullanıcısına göre düzenlenmiş veya Docker volume izinleri güncellenmiştir.

2. Network İletişim Hatası: Konteynerlerin birbirine localhost üzerinden erişemediği gözlemlenmiştir. Çözüm: Docker'ın iç DNS mekanizması kullanılarak servis isimleri (prometheus, node-exporter) üzerinden haberleşme sağlanmıştır.

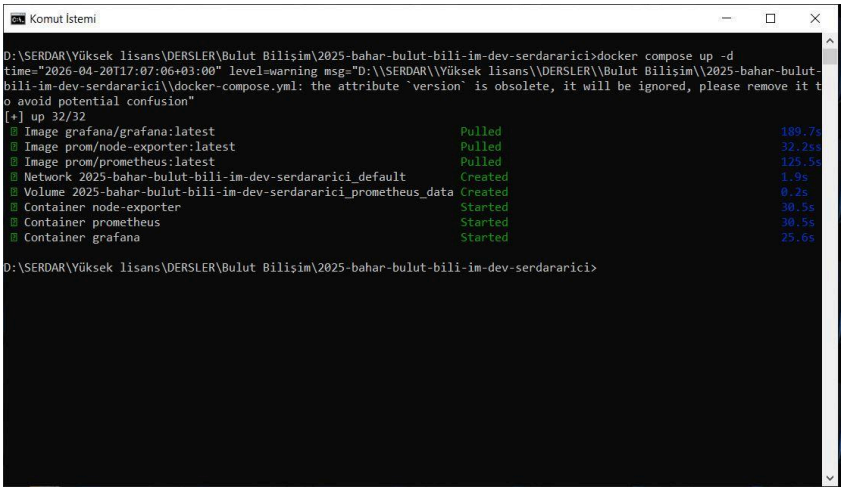



Şekil 4: Prometheus haberleşme

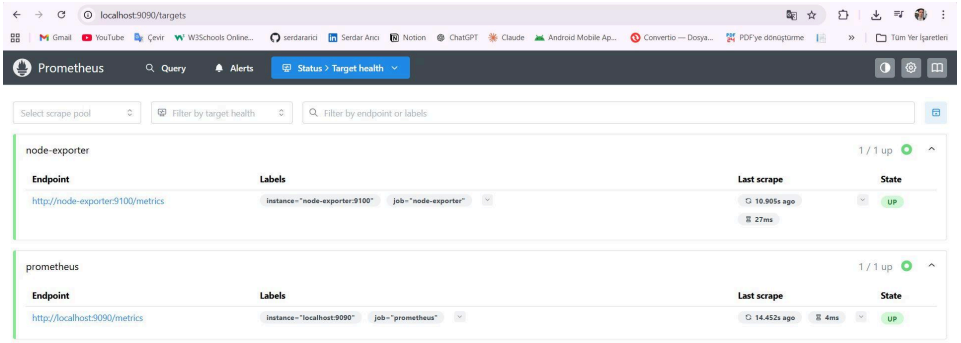
### 3.4. İlk Çalıştırma ve Doğrulama

Yapılandırma dosyalarının hazırlanmasının ardından terminal üzerinden docker compose up -d komutu icra edilerek kurulum süreci başlatılmıştır. Bu aşamada Docker Engine, yerel dizinde mevcut olmayan Prometheus, Node Exporter ve Grafana resmi imajlarını Docker Hub üzerinden katmanlı bir yapıda indirmeye (pulling) başlamıştır. İmajların başarıyla indirilmesini müteakip, tanımlanan ağ (network) ve depolama (volume) yapılandırmaları otomatik olarak oluşturulmuş ve konteynerler izole ancak birbirleriyle haberleşebilecek şekilde ayağa kaldırılmıştır.

Konteynerlerin çalışma durumu docker ps komutu ile teyit edildikten sonra, sistemin operasyonel sağlığını kontrol etmek amacıyla tarayıcı üzerinden http://localhost:9090 adresindeki Prometheus Web Arayüzü'ne erişim sağlanmıştır. İzleme ağının doğruluğunu denetlemek için "Status -> Targets" sekmesi ziyaret edilmiştir. Bu panelde, prometheus.yml dosyasında tanımlanan tüm hedef uç noktaların (endpoints) başarılı bir şekilde "kazındığı" (scraping) ve tüm servislerin "UP" (aktif/sağlıklı) statüsünde olduğu gözlemlenmiştir.
Bu durum, Prometheus sunucusunun hem kendi metriklerini hem de Node Exporter üzerinden gelen sistem metriklerini sorunsuz bir şekilde topladığını, dolayısıyla veri toplama hattının hatasız bir şekilde kurulduğunu kanıtlamaktadır. Bu aşamadan sonra toplanan ham veriler, analitik sorgular (PromQL) ve görselleştirme panelleri (Grafana) için hazır hale gelmiştir.




Şekil 5: docker compose up -d komutu




Şekil 6: Prometheus targets

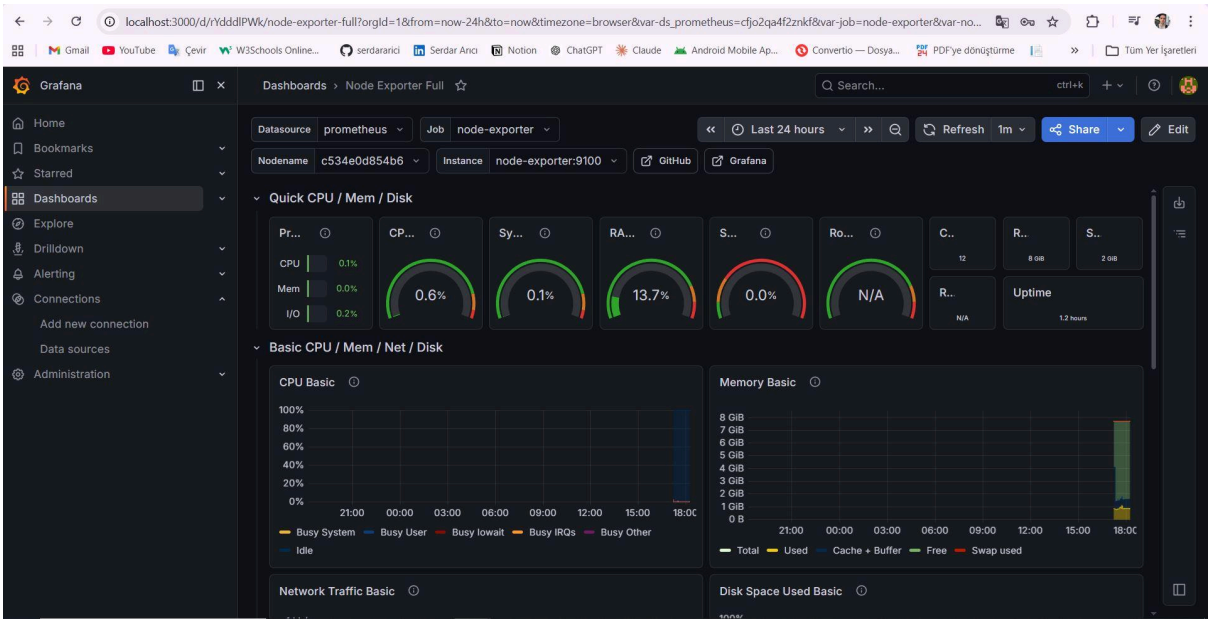
### 3.5. Deneyim ve Değerlendirme

Kurulum ve konfigürasyon süreci sonunda, Prometheus'un "pull-based" (çekme tabanlı) mimarisinin sağladığı operasyonel esneklik bizzat deneyimlenmiştir. Geleneksel izleme araçlarının aksine, yeni bir "exporter" veya mikro hizmetin sisteme dahil edilmesinin sadece merkezi konfigürasyon dosyasında (prometheus.yml) birkaç satırlık değişiklik yapılması ve servisin yeniden başlatılması kadar basit olması, sistemin neden bulut-yerel ekosistemde "Graduated" seviyesine ulaştığının pratik bir kanıtıdır.
Veri toplama aşamasının ötesinde, bu çalışmada ham metriklerin anlamlı içgörülere dönüştürülmesi amacıyla Grafana entegrasyonu gerçekleştirilmiştir. Prometheus bir veri kaynağı (data source) olarak Grafana'ya bağlanmış ve Node Exporter Full dashboard'u sisteme dahil edilmiştir. Bu süreçte şu deneyimler elde edilmiştir:

- Zaman Serisi Analizi: PromQL sorgularının Grafana üzerindeki dinamik değişkenlerle (variables) birleşmesi, sistemin CPU, RAM ve disk metriklerini saniyeler bazında yüksek çözünürlükle izlemeye olanak tanımıştır.

- Görselleştirme Gücü: Karmaşık ham veriler; ısı haritaları, göstergeler ve grafikler aracılığıyla görselleştirilerek sistem yöneticilerinin potansiyel darboğazları bir bakışta teşhis edebileceği bir yapıya dönüştürülmüştür.
- Entegrasyon Kolaylığı: Docker Compose sayesinde Prometheus ve Grafana arasındaki ağ iletişiminin (networking) sorunsuz yönetilmesi, dağıtık sistemlerin gözlemlenebilirliği (observability) açısından "kod olarak altyapı" (Infrastructure as Code) yaklaşımının önemini bir kez daha ortaya koymuştur.

Sonuç olarak, kurulan bu izleme hattı, proaktif bir hata tespit mekanizması sunmaktadır. Aşağıdaki görselde yer alan Grafana paneli, sistemin canlı performans verilerini ve metrikler arasındaki korelasyonu net bir şekilde ortaya koymaktadır.




Şekil 7: Grafana Dashboard

# 4. Karşılaştırmalı Değerlendirme

Gözlemlenebilirlik ve izleme ekosisteminde Prometheus'un yerini tam olarak konumlandırabilmek için, sektörde yaygın olarak kullanılan iki temel alternatif olan Zabbix ve Datadog ile karşılaştırılması büyük önem arz etmektedir. Bu karşılaştırma, sadece araçların özelliklerini değil, aynı zamanda bulut-yerel mimarilere olan adaptasyon yeteneklerini de ortaya koymaktadır.

## 4.1. Prometheus ve Zabbix Karşılaştırması: Geleneksel vs. Dinamik İzleme

"Zabbix, açık kaynaklı yapısı, esnekliği, ölçeklenebilirliği ve geliştirilebilir mimarisi ile öne çıkan bir ağ izleme yazılımıdır. Hem küçük ölçekli sistemlerde hem de kurumsal altyapılarda yaygın olarak kullanılan Zabbix; SNMP (Simple Network Management Protocol), ICMP (Internet Control Message Protocol), Modbus ve Zabbix ajanı gibi çeşitli protokoller

aracılığıyla geniş kapsamlı izleme imkânı sunmaktadır. Bununla birlikte, toplanan verilerin anlık olarak analiz edilmesi ve etkili bir şekilde görselleştirilmesi için Grafana gibi güçlü araçlara ihtiyaç duyulmaktadır. Zabbix ile Grafana entegrasyonu sayesinde sistem yöneticileri, özelleştirilebilir ve etkileşimli paneller üzerinden kapsamlı içgörüler elde edebilmektedir. Literatürde yapılan çalışmalar, bu entegrasyonun kullanıcı dostu arayüzler aracılığıyla hizmet kesintilerinin önlenmesinde önemli bir rol oynadığını ve otomatik keşif, esnek uyarı mekanizmaları ile iş odaklı metriklerin görselleştirilmesi açısından önemli katkılar sunduğunu göstermektedir.

Zaman serisi verilerinin yorumlanması yalnızca teknik ekipler için değil, aynı zamanda yönetim düzeyi için de kritik bir öneme sahiptir. Grafana üzerinde oluşturulan gösterge panelleri, farklı birimlerin kendi süreçlerine ilişkin metrikleri eş zamanlı olarak izlemesine ve veri odaklı kararlar almasına olanak tanımaktadır. Bu yaklaşım, özellikle SLA takibi, işlem süreleri ve hata oranları gibi metriklerin izlenmesi yoluyla operasyonel süreçlerin optimize edilmesine katkı sağlamaktadır. Ayrıca, yapay zekâ ve makine öğrenmesi tabanlı analizlerin ağ izleme sistemlerine entegrasyonu giderek yaygınlaşmaktadır. Bu sayede geçmiş verilerden öğrenerek olası arızaların önceden tahmin edilmesi, kapasite planlamasının yapılması ve aykırı durumların tespit edilmesi mümkün hale gelmektedir. Böylece ağ yönetimi reaktif bir yapıdan proaktif ve öngörülebilir bir yapıya dönüşmektedir.

Yapılan çalışmalar, Zabbix tabanlı ağ izleme sistemlerinin zamanında müdahale imkânı sunduğunu ve kaynak kullanımında esneklik sağladığını ortaya koymaktadır. Ayrıca, sistem uyarılarının Grafana panellerine entegre edilmesi sayesinde operasyonel kararların daha hızlı ve veri temelli bir şekilde alınabildiği belirtilmektedir. Zabbix'in hem fiziksel hem de sanal ortamlarda yüksek veri trafiği altında etkin bir izleme performansı sergilediği ve heterojen ağ yapılarında ölçeklenebilirlik ile güvenilirlik açısından güçlü bir çözüm sunduğu ifade edilmektedir. Gelişmiş otomasyon özellikleri ve SNMP tabanlı detaylı donanım izleme yetenekleri, yüksek güvenilirlik gerektiren sistemlerde kesintisiz operasyonu mümkün kılmaktadır.

Büyük ölçekli ağlarda Zabbix'in yüksek esneklik ve genişletilebilirlik sunduğu, Grafana ile birlikte kullanıldığında ise operasyonel verilerin daha anlaşılır ve hızlı bir şekilde yorumlanmasına olanak sağladığı vurgulanmaktadır. Enerji sektörü gibi kritik alanlarda gerçekleştirilen uygulamalarda, Zabbix'in cihaz performans verilerini SNMP aracılığıyla toplayarak operasyonel verimliliği artırdığı ve Grafana ile entegre görselleştirme sayesinde karar alma süreçlerini hızlandırdığı belirtilmektedir. Endüstriyel üretim ortamlarında ise Zabbix'in maliyet avantajının yanı sıra esnek yapılandırma ve görsel raporlama yetenekleri ile iş zekâsı destekli karar alma süreçlerine katkı sağladığı ifade edilmektedir. Grafana'nın sezgisel görselleştirme kabiliyeti sayesinde teknik veriler sadeleştirilerek yönetim seviyesine uygun paneller oluşturulmakta ve bu durum karar alma süreçlerini desteklemektedir. Bu yaklaşım, ağ izleme sistemleri ile karar destek mekanizmalarının entegrasyonu açısından literatürde önemli bir katkı sunmaktadır."[5]

Zabbix, uzun yıllardır endüstri standardı olarak kabul edilen, ağırlıklı olarak PHP ve C tabanlı geliştirilmiş, "ajan tabanlı" (agent-based) bir izleme çözümüdür. Prometheus ile arasındaki en temel fark, veri toplama ve mimari felsefesinde yatmaktadır:

- Veri Toplama Modeli: Zabbix temel olarak "Push" (gönderme) modelini kullanırken, Prometheus "Pull" (çekme) modelini benimser. Dinamik bulut ortamlarında, yeni bir konteyner ayağa kalktığında Prometheus'un servis keşfi (Service Discovery) ile bu hedefi otomatik bulması, Zabbix'in manuel yapılandırma gerektiren yapısına göre büyük bir avantaj sağlar.
- Veri Yapısı: Zabbix verileri geleneksel ilişkisel veritabanlarında (MySQL, PostgreSQL vb.) saklar; bu durum ölçeklenme sırasında veritabanı darboğazlarına yol açabilir. Prometheus ise yüksek sıkıştırma kabiliyetine sahip, zaman serisi odaklı özel bir TSDB kullanır.

### 4.2. Prometheus ve Datadog Karşılaştırması: Açık Kaynak vs. SaaS

"DataDog, tüm teknoloji yığını boyunca telemetri verilerinin toplanması, analiz edilmesi ve görselleştirilmesi için birleşik bir çözüm sunan bulut tabanlı bir gözlemlenebilirlik platformu olarak bu alanda önemli bir araç haline gelmiştir. Bu çalışma, DataDog'un yeteneklerini ve karmaşık yazılım ekosistemlerinde güçlü gözlemlenebilirlik ve izleme stratejilerinin uygulanmasına yönelik yöntemlerini derinlemesine analiz etmeyi amaçlamaktadır.

Mikroservisler, konteyner teknolojileri ve bulut tabanlı yaklaşımların yaygınlaşması, sistem davranışının izlenmesinde yeni zorlukları beraberinde getirmiştir. Geleneksel izleme araçları genellikle önceden tanımlanmış metrikler ve eşik değerlerine odaklandığından, bu dinamik ortamlarda sorunların kökenini anlamada yetersiz kalabilmektedir. DataDog ise bu zorluklara çözüm sunarak dağıtık izleme (distributed tracing), log analizi ve gerçek zamanlı sorgulama gibi yetenekler sağlar; böylece yalnızca metrikleri izlemekle kalmaz, aynı zamanda sistem davranışının derinlemesine analiz edilmesine de olanak tanır. Güncel teknolojik gelişmelere uyum sağlaması sayesinde DataDog, karmaşık yazılım sistemlerinde gözlemlenebilirlik ve izleme stratejilerinin önemli bir parçası haline gelmiştir.

Bu çalışma, özellikle DevOps ve Site Reliability Engineering (SRE) yaklaşımlarının giderek daha fazla benimsenmesi açısından önem taşımaktadır. Bu yaklaşımlar hızlı geliştirme döngülerini, sürekli dağıtımı ve sistem güvenilirliğini ön plana çıkarır. Etkili gözlemlenebilirlik, ekiplerin sorunları hızlı bir şekilde tespit edip çözmesine, performansı optimize etmesine ve sistem mimarisi ile kaynak yönetimi konusunda veri odaklı kararlar almasına olanak tanır."[6]

Datadog, bulut altyapıları için geliştirilmiş ticari bir SaaS (Hizmet Olarak Yazılım) platformudur. Prometheus ile kıyaslandığında aralarındaki temel ayrım "sahiplik" ve "kurulum karmaşıklığı" noktalarında toplanır:

- Operasyonel Yük: Datadog'un en büyük avantajı, sunucu yönetimi veya veritabanı bakımı gerektirmeyen bir bulut hizmeti olmasıdır. Ancak Prometheus, tamamen açık

kaynaklıdır ve organizasyonun kendi altyapısında (on-premise veya VPC) tam veri kontrolü sağlar.

- Entegrasyon ve Maliyet: Datadog, çok geniş bir hazır entegrasyon kütüphanesi sunmasına rağmen, izlenen metrik ve host başına ücretlendirme yaptığı için ölçek büyüdükçe maliyetleri hızla artabilir. Prometheus ise CNCF ekosisteminin bir parçası olarak ücretsizdir ve özellikle Kubernetes ortamlarında standart dışı bir maliyet avantajı sağlar.

### 4.3. Sistematik Karşılaştırma Matrisi

Aşağıdaki tablo, projenin teknik değerlendirmesini sistematik bir matris üzerinden sunmaktadır:

| Kriter | Prometheus | Zabbix | Datadog |
|---|---|---|---|
| **Mimari Felsefesi** | Bulut-Yerel (Cloud-Native) | Geleneksel/Monolitik | Hibrit/SaaS |
| **Veri Toplama** | Pull (Çekme) | Push & Pull (Ağırlıklı Push) | Push (Ajan tabanlı) |
| **Veri Modeli** | Çok Boyutlu (Labels/TSDB) | İlişkisel (Key-Value/RDBMS) | Çok Boyutlu (Tags/SaaS) |
| **Servis Keşfi** | Tam Otomatik (K8s Entegre) | Kısmi/Manuel | Otomatik |
| **Öğrenme Eğrisi** | Orta (PromQL uzmanlık ister) | Yüksek (Karmaşık Arayüz) | Düşük (Kullanıcı Dostu) |
| **Maliyet** | Ücretsiz (Açık Kaynak) | Ücretsiz (Açık Kaynak) | Ücretli (Metrik başına) |
| **Topluluk Desteği** | Çok Güçlü (CNCF Graduated) | Güçlü (Köklü Topluluk) | Kurumsal Destek |

### 4.4. Genel Değerlendirme

Karşılaştırma sonuçları göstermektedir ki; Zabbix fiziksel ağ cihazları ve statik sunucu altyapıları için halen geçerli bir seçenek olsa da, mikro hizmetlerin dinamik dünyasında hantal kalmaktadır. Datadog ise yönetim kolaylığı arayan ancak yüksek bütçeli projeler için idealdir. Buna karşın Prometheus, özellikle Kubernetes tabanlı modern mimarilerde,

özelleştirilebilir yapısı, güçlü sorgulama dili (PromQL) ve CNCF güvencesiyle en dengeli ve performanslı çözümü sunmaktadır.

# 5. Sonuç ve Öneriler

Bu çalışma kapsamında, Cloud Native Computing Foundation (CNCF) ekosisteminin en olgun projelerinden biri olan Prometheus'un mimari yapısı, kurulum süreçleri ve endüstriyel alternatifleri teknik bir perspektifle incelenmiştir. Yapılan uygulamalı çalışmalar ve teorik karşılaştırmalar sonucunda elde edilen bulgular ışığında aşağıdaki değerlendirmelere ulaşılmıştır.

### 5.1. Teknik Değerlendirme ve Kazanımlar

Prometheus, bulut-yerel mimarilerin doğasında bulunan dinamizm ve ölçeklenebilirlik sorunlarına "pull-based" veri toplama ve çok boyutlu etiketleme (labeling) stratejisiyle proaktif bir çözüm sunmaktadır. Docker ve Docker Compose üzerinde gerçekleştirilen kurulum deneyimi, sistemin "kod olarak altyapı" (IaC) prensiplerine uyumunu ve saniyeler içerisinde karmaşık metrikleri analiz edilebilir hale getirdiğini kanıtlamıştır. Özellikle Grafana entegrasyonu ile ham metriklerin görselleştirilmesi, sistemin sadece bir veri toplama aracı değil, aynı zamanda stratejik bir karar destek mekanizması olduğunu göstermiştir.

### 5.2. Güçlü ve Zayıf Yönlerin Analizi

Güçlü Yönler:

- Dinamik Servis Keşfi: Mikro hizmetlerin sürekli ölçeklendiği ortamlarda manuel müdahale gerektirmeden yeni hedefleri otomatik olarak izleme ağına dahil eder.
- Operasyonel Basitlik: Herhangi bir dış bağımlılığa (veritabanı vb.) ihtiyaç duymadan, kendi zaman serisi veritabanı (TSDB) üzerinden yüksek performanslı veri saklama imkânı sunur.
- Gelişmiş Sorgulama Yeteneği: PromQL sayesinde veriler üzerinde karmaşık matematiksel analizler ve anomali tespiti yapılmasına olanak tanır.

Zayıf Yönler:

- Uzun Süreli Veri Saklama: Prometheus varsayılan yapısında uzun vadeli (yıllık) veri saklama için tasarlanmamıştır; bu durum için Thanos veya Cortex gibi ek çözümlere ihtiyaç duyar.
- Öğrenme Eğrisi: PromQL sorgulama dilinde uzmanlaşmak, SQL tabanlı geleneksel dilleri kullanan ekipler için başlangıçta zorlayıcı olabilir.

### 5.3. Uygulama Senaryoları ve Öneriler

Yapılan analizler sonucunda Prometheus'un kullanımı için şu öneriler sunulmaktadır:

1. Mikro Hizmet ve Kubernetes Ortamları: Eğer altyapı konteynerleştirilmiş bir yapıda ise Prometheus, Kubernetes ile yerel uyumluluğu sayesinde tartışmasız en uygun çözümdür.
2. Yüksek Çözünürlüklü İzleme Gereksinimi: Sistem sağlığının saniyelik bazda takibinin kritik olduğu finans veya e-ticaret gibi sektörlerde, düşük gecikmeli veri işleme kabiliyeti nedeniyle tercih edilmelidir.
3. Maliyet ve Veri Gizliliği Odaklı Projeler: Datadog gibi SaaS çözümlerinin aksine, verinin kurum içerisinde kalması gereken ve maliyet kalemlerinin minimize edilmek istendiği açık kaynak projelerde kullanılmalıdır.
4. Hibrit Yapılar İçin Karma Yaklaşım: Statik ağ cihazları ve fiziksel sunucuların yoğun olduğu bölümlerde Zabbix kullanımı sürdürülürken, uygulama seviyesindeki metrik takibi için Prometheus ile entegre bir hibrit yapı kurulması operasyonel verimliliği maksimize edecektir.

Sonuç olarak Prometheus; gözlemlenebilirlik disiplinini reaktif bir yapıdan (hata oluştuktan sonra müdahale) proaktif bir yapıya (hata oluşmadan analiz) dönüştüren, modern bulut bilişim dünyasının vazgeçilmez bir bileşenidir.

## Kaynakça


[1] S. Deng *vd.*, “Cloud-Native Computing: A Survey From the Perspective of Services”, *Proc. IEEE*, c. 112, sy. 1, ss. 12-46, Oca. 2024, doi: 10.1109/JPROC.2024.3353855.

[2] K. Alhamazani *vd.*, “An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art”, *Computing*, c. 97, sy. 4, ss. 357-377, Nis. 2015, doi: 10.1007/s00607-014-0398-5.

[3] E. Ogbuefi, O. T. Odofin, A. A. Abayomi, I. Adekunle, O. A. Agboola, ve S. Owoade, “A Review of System Monitoring Architectures Using Prometheus, ELK Stack, and Custom Dashboards”, c. 4, sy. 12, 2021.

[4] Taiwo Joseph Akinbolaji, Godwin Nzeako, David Akokodaripon, ve Akorede Victor Aderoju, “Proactive monitoring and security in cloud infrastructure: leveraging tools like Prometheus, Grafana, and HashiCorp Vault for Robust DevOps Practices”, *World J. Adv. Eng. Technol. Sci.*, c. 13, sy. 2, ss. 074-089, Kas. 2024, doi: 10.30574/wjaets.2024.13.2.0543.

[5] G. Tuğrul, “ZABBIX AĞ YÖNETİM YAZILIMINA ETKİN VARLIK EKLENMESİ VE VERİLERİN GÖRSELLEŞTİRİLMESİ”.

[6] S. Biswas, “Comprehensive Observability and Monitoring Strategies Using Datadog”.