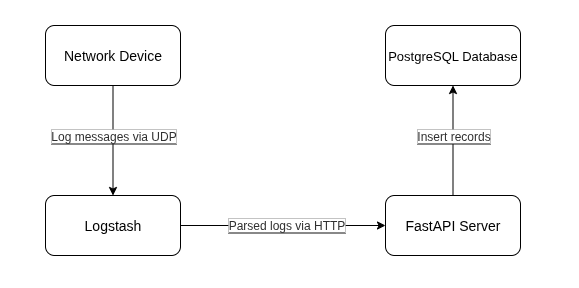
**۱. معرفی پروژه**

پروژه LogMorph با هدف تجزیه‌ و تحلیل (Parse) لاگ‌های دریافتی، استخراج فیلدها و مقادیر کلیدی، و سپس ذخیره‌سازی ساخت‌یافته‌ی آن‌ها در پایگاه داده PostgreSQL طراحی و پیاده‌سازی شده‌است.

در این سامانه، ابزار Logstash لاگ‌ها را مستقیماً از طریق پورت UDP دریافت می‌کند تا از عملیات زمان‌بر و پرهزینه‌ی خواندن و نوشتن فایل‌ها (File I/O) جلوگیری شده و مصرف منابع به حداقل برسد.

Logstash پس از دریافت، لاگ‌ها را تجزیه کرده (Parse) و از طریق پروتکل HTTP به سرویس FastAPI ارسال می‌کند. در سرویس FastAPI، این لاگ‌ها پردازش شده و در نهایت در پایگاه داده PostgreSQL ذخیره می‌شوند تا در آینده قابلیت تحلیل، جستجو و استفاده در گزارش‌گیری یا سیستم‌های مانیتورینگ را داشته باشند.

LogMorph با طراحی ماژولار، عملکرد هم‌زمان (asynchronous) و قابلیت پردازش دسته‌ای لاگ‌ها، گزینه‌ای ایده‌آل برای پیاده‌سازی سیستم‌های مانیتورینگ و امنیت شبکه در محیط‌های عملیاتی محسوب می‌شود.



**۲. پیش‌نیازها (dependencies )**

* Python 3.10+  
  زبان برنامه‌نویسی اصلی پروژه برای پیاده‌سازی منطق سمت سرور با FastAPI .
* FastAPI  
  فریم‌ورک مدرن و سریع برای پیاده‌سازی REST API که وظیفه‌ی دریافت و پردازش لاگ‌ها را بر عهده دارد.
* Uvicorn  
  سرور ASGI سبک‌وزن برای اجرای اپلیکیشن FastAPI به‌صورت asynchronous و سریع.
* PostgreSQL  
  برای ذخیره‌سازی ساخت‌یافته‌ لاگ‌های پردازش‌شده استفاده می‌شود.
* Logstash  
  ابزار جمع‌آوری و پردازش لاگ که لاگ‌ها را از طریق UDP دریافت کرده و پس از parse به FastAPI ارسال می‌کند.
* sqlalchemy  
  ابزار ORM برای تعریف مدل‌های دیتابیس و مدیریت ارتباط بین داده‌ها و پایگاه داده در Python.
* psycopg2  
  درایور PostgreSQL برای Python که ارتباط مستقیم و سریع بین برنامه و پایگاه داده را فراهم می‌کند.

**۳. مراحل نصب و راه‌اندازی پیش‌نیازها**

### نصب پکیج‌های پایتون موردنیاز پروژه

### مرحله اول ) بررسی نسخه پایتون

**sudo apt install python3 python3-pip python3-venv**

### مرحله دوم) ساخت دایرکتوری پروژه و محیط مجازی (Virtual Environment)

۱. ساخت دایرکتوری مخصوص پروژه :

**mkdir <project\_name>**

**cd <project\_name>**

۲. ساخت virtual environment :

**python3 -m venv <venv\_name>**

۳. فعالسازی venv :

**source <venv\_name>/bin/activate**

مرحله سوم) نصب کتابخانه‌های مورد نیاز داخل محیط مجازی

**pip install fastapi uvicorn[standard] sqlalchemy psycopg2 requests**

مرحله چهارم) ساخت فایل اپلیکیشن (app.py)

پس از نصب و آماده‌سازی محیط مجازی و نصب کتابخانه‌های مورد نیاز، حالا نوبت به ساخت فایل اصلی برنامه یعنی app.py است. این فایل، هسته‌ی اصلی برنامه FastAPI خواهد بود و وظیفه‌ی دریافت لاگ‌ها از Logstash، پردازش و ذخیره‌ی آن‌ها در PostgreSQL را بر عهده دارد.

**cd <project\_name>**

**touch app.py**

* نصب و پیکربندی Logstash

مرحله اول) نصب logstash از وب‌سایت رسمی ( <https://www.elastic.co/downloads/logstash> )

مرحله دوم) ساخت فایل پیکربندی Logstash

Logstash با استفاده از یک فایل کانفیگ که در مسیر **/etc/logstash/conf.d/** قرار دارد (مثلاً فایل logmorph.conf ) تعیین می‌کند که:

* داده‌ها را از کجا دریافت کند (input)
* چگونه آن‌ها را پردازش کند (filter)
* خروجی را به کجا بفرستد (output)

### مرحله سوم) اجرای Logstash

برای اجرای Logstash با فایل پیکربندی بالا:

**sudo systemctl start logstash**

یا برای اجرای دستی و بررسی لاگ‌ها:

**sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/logmorph.conf**

## **۴. نحوه عملکرد برنامه**

در این بخش، مراحل اجرای پروژه از لحظه‌ی دریافت لاگ توسط Logstash تا ذخیره‌سازی آن در پایگاه داده PostgreSQL به‌صورت مرحله‌به‌مرحله شرح داده می‌شود:

### گام اول: دریافت لاگ توسط Logstash

۱. ورودی لاگ‌ها:  
 لاگ‌ها از سمت دستگاه‌ها یا سرویس‌های دیگر از طریق پروتکل UDP روی پورت 5140 به سیستم ارسال می‌شوند.

۲. دریافت توسط Logstash :  
Logstash با استفاده از پلاگین udp در فایل پیکربندی logmorph.conf لاگ‌ها را دریافت می‌کند.

input {  
 udp {  
 port => 5140  
 codec => plain { charset => "UTF-8" }  
 }  
}

**گام دوم: تجزیه و پردازش لاگ در Logstash**

پس از دریافت لاگ‌ها از طریق UDP، Logstash آن‌ها را با استفاده از فیلترهای مشخص، تجزیه (Parse) کرده و به ساختار قابل‌فهم برای تحلیل تبدیل می‌کند.

#### ۱. فیلتر grok : استخراج داده‌ها از لاگ

* فیلتر grok در Logstash برای استخراج بخش‌های خاصی از یک لاگ (مانند IP، MAC، Port و ...) با استفاده از الگوهای regex طراحی شده.

در فایل پیکربندی، دو الگوی مختلف برای تشخیص فرمت‌های مختلف لاگ تعریف شده:

filter {  
 grok {  
 match => {  
 "message" => [  
 # Format #1: count+seq, single MAC, trailing rest1  
 "^%{NUMBER}\s+%{NUMBER}\s+\[%{WORD:in\_mac} %{WORD:out\_mac}\] ... %{GREEDYDATA:rest1}"  
 # Format #2: count+seq, two MAC blocks, rest1/rest2  
 "^%{NUMBER}\s+%{NUMBER}\s+\[%{WORD:in\_mac} %{WORD:out\_mac}\] ... %{GREEDYDATA:rest1}\  
 [%{WORD:in\_mac2} %{WORD:out\_mac2}\] ... %{GREEDYDATA:rest2}"  
 ]  
 }  
 tag\_on\_failure => ["\_grok\_fail"]  
}

### این الگوها اجزای لاگ را به فیلدهای زیر تقسیم می‌کنند:

* in\_mac, out\_mac, in\_mac2, out\_mac2 : آدرس‌های MAC
* src\_ip, dst\_ip, src\_port, dst\_port : آدرس‌های IP و پورت‌
* proto, protocol2 : پروتکل‌ها
* description : ترکیب بخش‌های متنی اضافه لاگ (rest1 وrest2 )

### گام دوم: ارسال لاگ به FastAPI

### ۱. تبدیل به JSON Logstash لاگ دریافتی را به قالب JSON. تبدیل می‌کند.

۲. ارسال HTTP POST  
از طریق پلاگین http output ، لاگ به صورت POST به endpoint/logs روی پورت 10000 ارسال می‌شود:

output {  
 http {  
 url => "http://<fastAPI\_ip>:10000/logs"  
 http\_method => "post"  
 format => "json\_batch"  
 content\_type => "application/json"  
 retry\_failed => false  
}

stdout {

codec => rubydebug

}

}

### گام سوم: پردازش لاگ در FastAPI و ارسال به PostgreSQL

در فایل app.py :

۱. تعریف برنامه FastAPI و اتصال به پایگاه داده

app = FastAPI()

DATABASE\_URL = "postgresql://<username>:<password>@<ip>:<port>/<database\_name>"

engine = create\_engine(DATABASE\_URL, pool\_pre\_ping=True)

SessionLocal = sessionmaker(bind=engine, autoflush=False, autocommit=False)

Base = declarative\_base()

* اتصال به دیتابیس PostgreSQL با استفاده از DATABASE\_URL
* تعریف session برای انجام عملیات بر روی دیتابیس

۲. تعریف مدل جدول logs

class Log(Base):

\_\_tablename\_\_ = "logs"

id = Column(Integer, primary\_key=True, index=True) # Primary key  
 timestamp = Column(DateTime, nullable=False) # Server-side timestamp  
 ….

* هر لاگ دریافتی در قالب یک ردیف (record) در جدول logs ذخیره می‌شود.
* ستون‌هایی مثل in\_mac, out\_mac, src\_ip, dst\_ip, description و... در اینجا مشخص می‌شوند.

۳. ایجاد جدول‌ها در پایگاه داده

Base.metadata.create\_all(bind=engine)

* این خط اطمینان می‌دهد که در شروع برنامه، جدول‌ها ساخته شوند (اگر وجود نداشته باشند)

### ۴. دریافت و پردازش لاگ‌ها از طریق API

@app.post("/logs")

async def receive\_logs(request: Request):  
 ….

## **گام چهارم : مشاهده رکوردهای لاگ در پایگاه داده PostgreSQL**

### ۱. اتصال به دیتابیس با ابزار psql :

اگر دستور psql روی سرور PostgreSQL نصب باشه :

**psql -U <username> -d <database\_name>**

### ۲. اجرای یک کوئری ساده برای مشاهده لاگ‌ها:

پس از اتصال به دیتابیس، می‌تونی از دستور زیر برای نمایش رکوردهای جدول logs استفاده کنی:

**SELECT \* FROM logs ORDER BY id DESC LIMIT 10;**

این کوئری آخرین ۱۰ لاگ ثبت‌شده را به ترتیب از جدید به قدیم نمایش می‌دهد.

۳. مشاهده رکوردها با فیلتر خاص:

مثلاً اگه بخوایم فقط لاگ‌هایی با src\_ip = '192.168.1.10' رو ببینیم :

SELECT \* FROM logs WHERE src\_ip = '192.168.1.10';