1.List some logging and visualization tools available in the market with the preferred scenario to use one over the other.

Below are some of the logging and visualization tools and their features and scenario of their uses:

Datadog: In addition to logging, this platform also enables teams to look at analytics and other features. Users can use the service to find all of their logs in a single interface. Moreover, the platform can help DevOps teams deal with outages that may happen across the globe.Various data related to web and app traffic is also available.

Dynatrace: It offers team members a wide range of log visualization tools and data. The platform also has business analytics and allows DevOps managers to analyze their apps' performances. Digital experience monitoring is also possible.

Logz.io: The Logz.io platform has a range of log visualization tools, such as app insights and ELK stack-based log management solutions. Users can also set up alerts if they suffer system outages. Logz.io also has security and DevOps analytics solutions, plus monitoring tools for business apps. Users can index up to 1 GB of log data for free.

Graylog: Its primary offerings are a range of log analysis tools. Users have access to numerous features, including alerts and compliance. The tool serves multiple industries and job functions, such as government, compliance and audit, telecom, fintech, and education.DevOps teams can also use Graylog to look over their app performance, along with identifying security breaches. A wide range of data options is also available.

Google Cloud Logging:Features of Google Cloud Logging include importing custom log data, as well as real-time analysis. Teams can also use the platform to carry out audits and report problems that may arise. A wide range of third-party integrations is also available.

Scalyr: It offers a wide range of app architecture and log visualization tools. Users will discover a user-friendly interface, as well as features to monitor in case problems arise. Data is also available soon after collection. we will also find that the platform offers comprehensive data for all events that take place. Moreover, we can integrate various data tools – including Logstash.

LOGalyze:LOGalyze allows DevOps teams to uncover events as they happen and break down their searches within multiple categories. LOGalyze supports various systems, including Windows, Linux, and OS. Logging customization is also available for both native and custom apps. Users can also analyze their logging data by correlating trends,understanding patterns, and individual event management. You can also tag each event and create separate categories.

Logiq.ai: Using the platform, you can set up comprehensive monitoring for your cloud and app infrastructures. Moreover, you can create alerts that prompt you to take action whenever something has gone wrong or about to go wrong. Logiq.ai also comes with an integrated user interface, making log data management and monitoring easier and better-laid out. The platform also enables you to incorporate logs from Openstack, Kubernetes, and other IaaS services. On the Logiq.ai website, you'll also find a wide range of resources, such as blog posts, tutorials, and eBooks – to further expand your knowledge.

2. Mention 10 best practises when logging. Why is log formatting necessary?

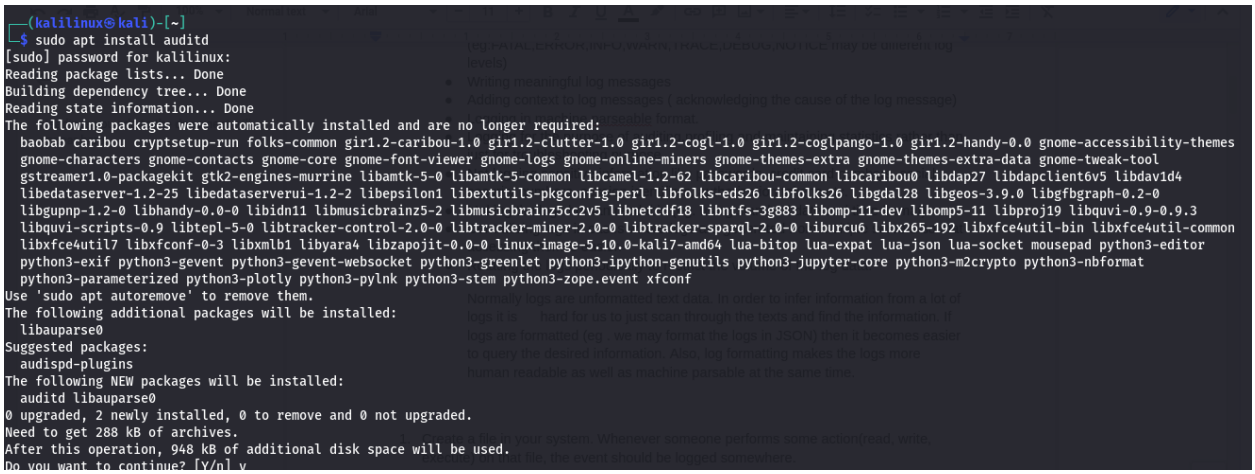
Below are the best practices while logging:

- Logging of only what is needed rather than logging everything
- using a different log level for different logs according to their characteristics and effect they have on the application or system (eg:FATAL,ERROR,INFO,WARN,TRACE,DEBUG,NOTICE may be different log levels)
- Writing meaningful log messages
- Adding context to log messages (acknowledging the cause of the log message)
- Logging in machine parseable format.
- Logging for the purpose of auditing,profiling and maintaining statistics rather than just for troubleshooting purpose.
- Not logging sensitive information like passwords, credit card numbers, social security numbers, session identifiers, authorization tokens etc.
- Making the logs human readable (eg. using standard date and time format)
- Centralizing logs in cases where logs are generated from multiple servers so that accessing them is easier.
- Rotating the logs periodically to restrict the volume of the log data.

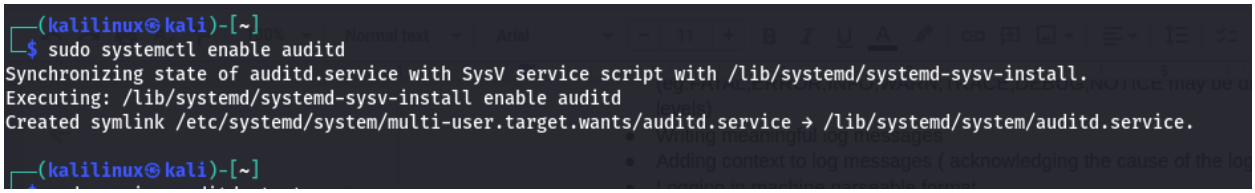
Normally logs are unformatted text data. In order to infer information from a lot of logs it is hard for us to just scan through the texts and find the information. If logs are formatted (eg . we may format the logs in JSON) then it becomes easier to query the desired information. Also, log formatting makes the logs more human readable as well as machine parsable at the same time.

3.Create a file in your system. Whenever someone performs some action(read, write, execute) on that file, the event should be logged somewhere.

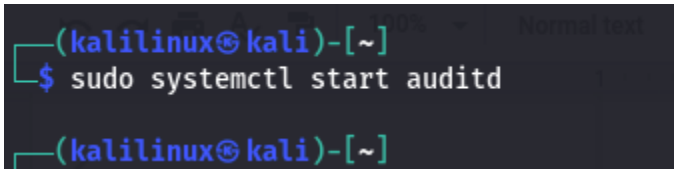
We install auditd which is a service used to audit the changes in files and system.



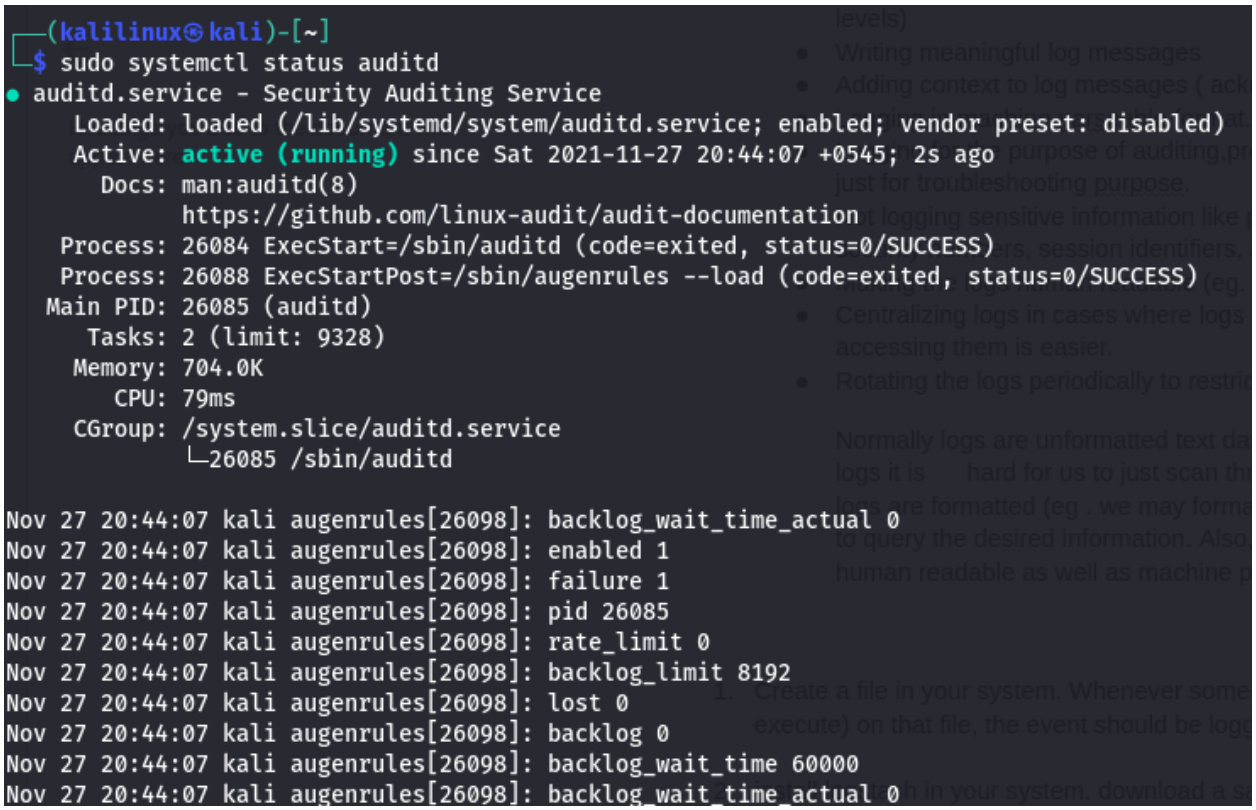
We enable the service auditd



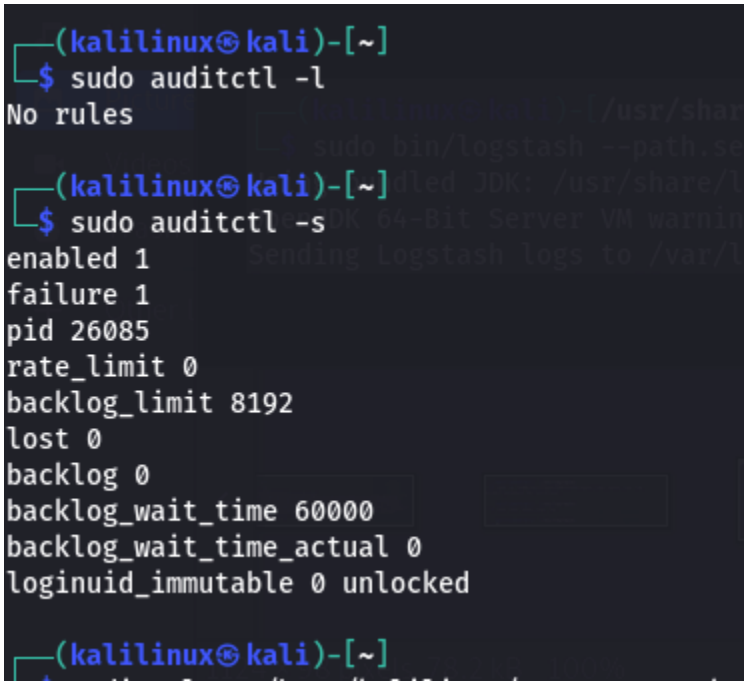
After enabling the service, we start it



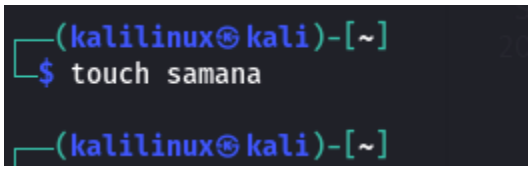
We then verify that the service is properly running by viewing it's status



We check the rules and status of the audit system as follows(currently there are no rules)

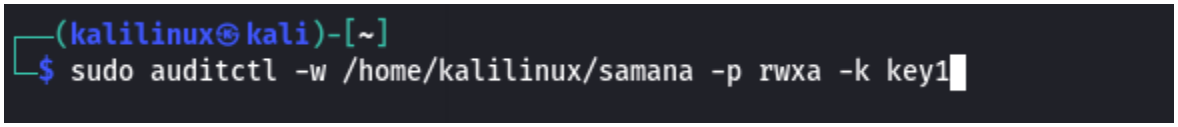


Now we create a file named samana.

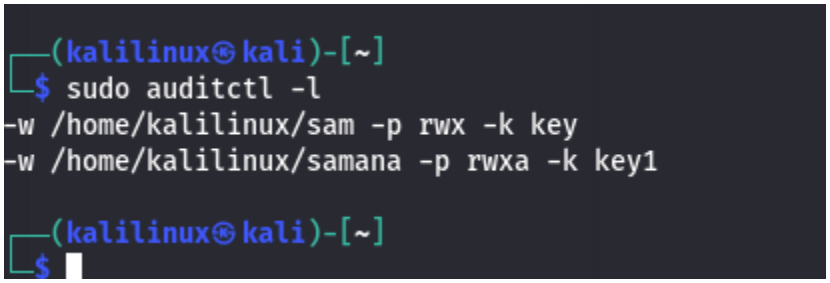


Now we set auditctl rule to set a watch for filesystem where it watches for read write execute and attribute change in the file (rwxa)

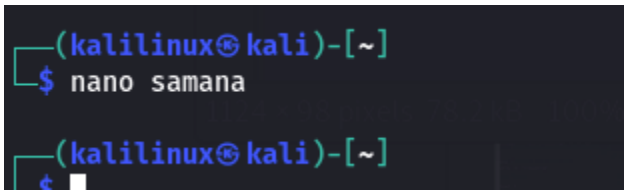
We associate the rule with a key named key1 to identify the rule with the key

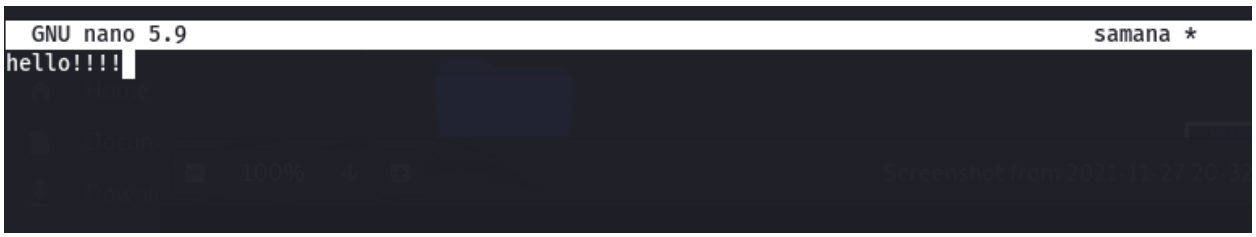


Now we can see the rules listed as below:

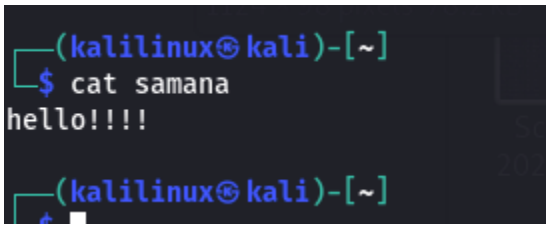


Now we write on that file named samana



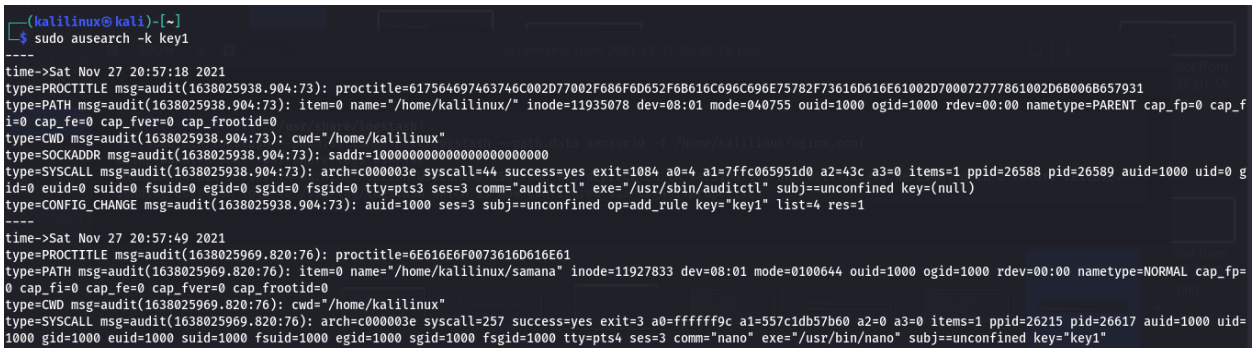


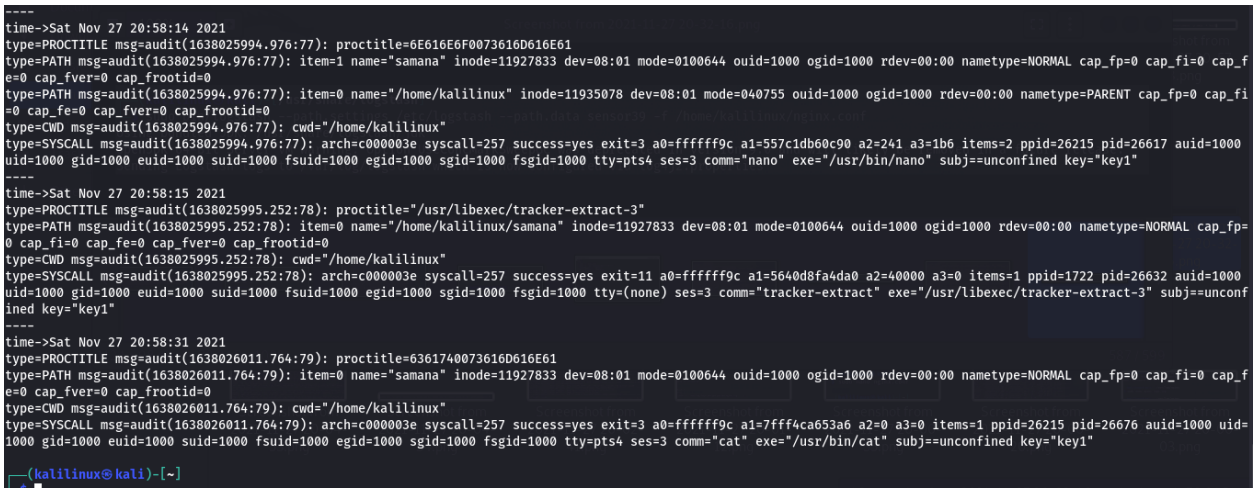
Also we read the contents of that file



After performing read and write permissions on that file, we search for the audit logs using our key named “key1”.

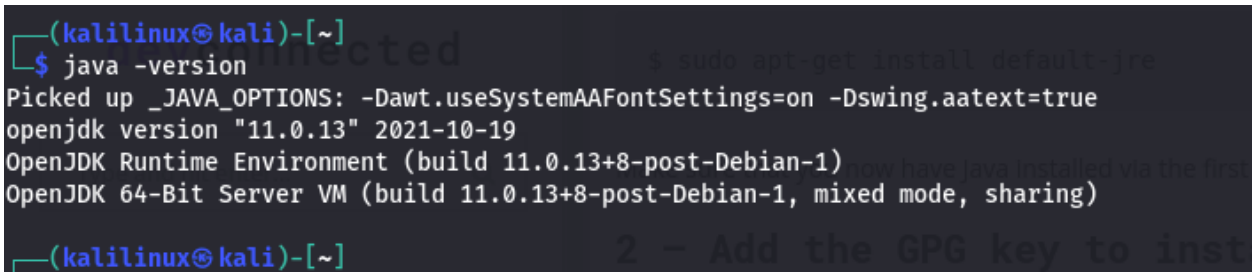
Below we can see the logs that show what actions have been performed on our specified file. We can also see the mediums through which our file was read and written (namely cat and nano)



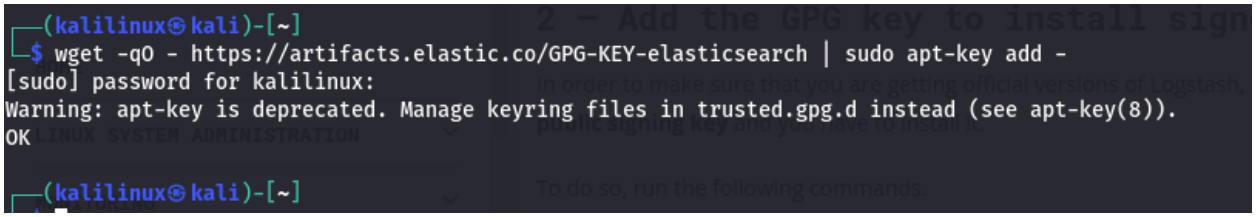


4. install logstash in your system. download a sample nginx log from https://github.com/elastic/examples/blob/master/Common%20Data%20Formats/nginx_logs/nginx_logs , parse the logs using logstash. The parsed output must contain the geographical information like country, state etc. that the request is originating from. save the parsed output to a file in your system.

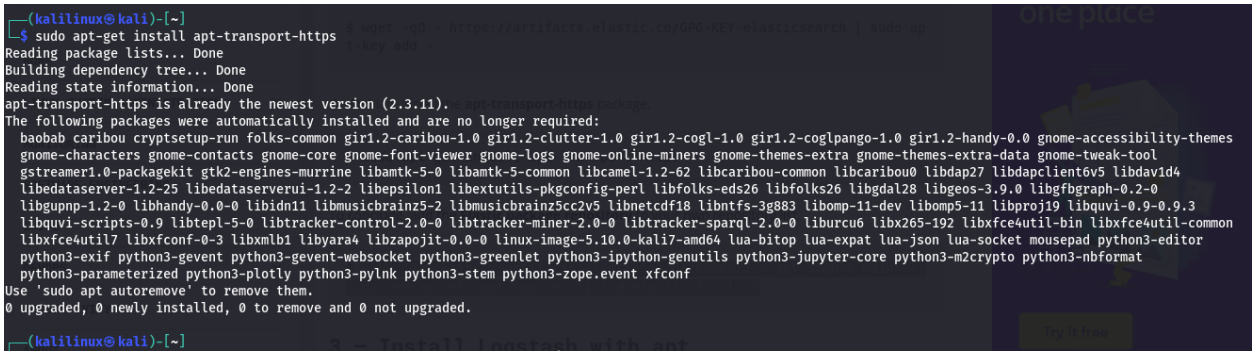
Before installing logstash we check if java is installed in our system



We add the GPG key for elasticsearch



Then, We install apt-transport-https



Then we add the elastic package repository to our repository list

```
┌──(kalilinux㉿kali)-[~]
└─$ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
deb https://artifacts.elastic.co/packages/7.x/apt stable main

┌──(kalilinux㉿kali)-[~]
└─$
```

Now after performing sudo apt-get update, we install logstash as follows:

```
┌──(kalilinux㉿kali)-[~]
└─$ sudo apt-get install logstash
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
logstash is already the newest version (1:7.15.2-1).
The following packages were automatically installed and are no longer required:
  baobab caribou cryptsetup-run folks-common gir1.2-caribou-1.0 gir1.2-clutter-1.0 gir1.2-cogl-1.0 gir1.2-coglpango-1.0 gir1.2-handy-0.0 gnome-accessibility-themes
  gnome-characters gnome-contacts gnome-core gnome-font-viewer gnome-logs gnome-online-miners gnome-themes-extra gnome-themes-extra-data gnome-tweak-tool
  gstreamer1.0-packagekit gtk2-engines-murrine libamtk-5-0 libamtk-5-common libcamel-1.2-62 libcaribou-common libcaribou0 libdap27 libdapclient6v5 libdav1d4
  libedataserver-1.2-25 libedataserverui-1.2-2 libepsilon1 libextutils-pkgconfig-perl libfolks-eds26 libfolks26 libgdal28 libgeos-3.9.0 libgfbgraph-0.2-0
  libgupnp-1.2-0 libhandy-0.0-0 libidn11 libmusicbrainz5-2 libmusicbrainz5cc2v5 libnetcdf18 libntfs-3g883 libomp-11-dev libomp5-11 libproj19 libquvi-0.9-0.9.3
  libquvi-scripts-0.9 libtepl-5-0 libtracker-control-2.0-0 libtracker-miner-2.0-0 libtracker-sparql-2.0-0 liburcu6 libx265-192 libxfce4util-bin libxfce4util-common
  libxfce4util7 libxfconf-0-3 libxmlb1 libyara4 libzapojit-0.0-0 linux-image-5.10.0-kali7-amd64 lua-bitop lua-expat lua-json lua-socket mousepad python3-editor
  python3-exif python3-gevent python3-gevent-websocket python3-greenlet python3-ipython-genutils python3-jupyter-core python3-m2crypto python3-nbformat
  python3-parameterized python3-plotly python3-pylnk python3-stem python3-zope.event xfconf
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

┌──(kalilinux㉿kali)-[~]
```

Now we download the sample nginx logs and copy it to the /var/log/nginx directory under the name nginx_logs.log

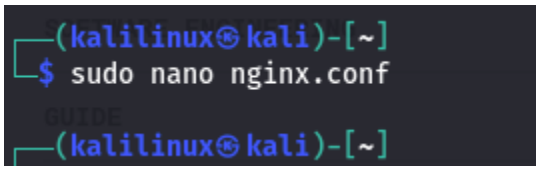
```
┌──(kalilinux㉿kali)-[/var/log/nginx]
└─$ sudo cp /home/kalilinux/Downloads/nginx_logs nginx_logs.log
[sudo] password for kalilinux:

┌──(kalilinux㉿kali)-[/var/log/nginx]
└─$ sudo nano nginx_logs.log

┌──(kalilinux㉿kali)-[/var/log/nginx]
```

Here we can see the sample nginx logs as follows:

Now we create a config file for logstash named nginx.conf(since we are parsing the nginx logs)



The logstash configuration file is created as follows:

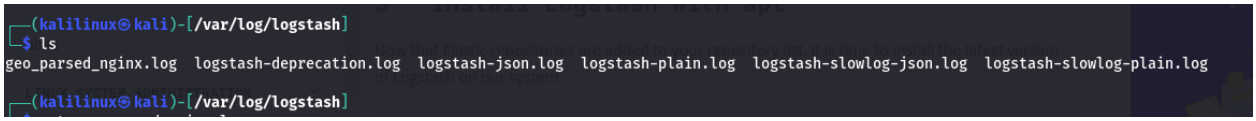
```
  GNU nano 5.9                                                    nginx.conf
input {
  file {
    type => "nginx-log"
    path => "/var/log/nginx/nginx_logs.log"
    sincedb_path => "/dev/null"
   start_position => "beginning"}
}
filter {
grok{
match=>{
"message"=>"%{IP:clientip} \- \- \["
}
}
geoip {
source => "clientip"
}
}
output {
  file {
   path => "/var/log/logstash/geo_parsed_nginx.log"    }
}
```

Now we run the created configuration file as follows:

```
┌──(kalilinux㉿kali)-[/usr/share/logstash]
└─$ sudo bin/logstash --path.settings /etc/logstash --path.data sensor39 -f /home/kalilinux/nginx.conf
Using bundled JDK: /usr/share/logstash/jdk
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
Sending Logstash logs to /var/log/logstash which is now configured via log4j2.properties
```

```
┌──(kalilinux㉿kali)-[/usr/share/logstash]
└─$ sudo bin/logstash --path.settings /etc/logstash --path.data sensor39 -f /home/kalilinux/nginx.conf
Using bundled JDK: /usr/share/logstash/jdk
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
Sending Logstash logs to /var/log/logstash which is now configured via log4j2.properties
[2021-11-27T20:02:13,145][INFO ][logstash.runner          ] Log4j configuration path used is: /etc/logstash/log4j2.properties
[2021-11-27T20:02:13,164][INFO ][logstash.runner          ] Starting Logstash {"logstash.version"=>"7.15.2", "jruby.version"=>"jruby 9.2.19.0 (2.5.8) 2021-06-15 55810c55
2b OpenJDK 64-Bit Server VM 11.0.12+7 on 11.0.12+7 +indy +jit [linux-x86_64]"}
[2021-11-27T20:02:13,973][WARN ][logstash.config.source.multilocal] Ignoring the 'pipelines.yml' file because modules or command line options are specified
[2021-11-27T20:02:16,894][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
[2021-11-27T20:02:18,345][INFO ][org.reflections.Reflections] Reflections took 176 ms to scan 1 urls, producing 120 keys and 417 values
[2021-11-27T20:02:20,940][WARN ][org.logstash.instrument.metrics.gauge.LazyDelegatingGauge][main] A gauge metric of an unknown type (org.jruby.RubySymbol) has been creat
ed for key: status. This may result in invalid serialization.  It is recommended to log an issue to the responsible developer/development team.
[2021-11-27T20:02:20,973][WARN ][org.logstash.instrument.metrics.gauge.LazyDelegatingGauge][main] A gauge metric of an unknown type (org.jruby.RubySymbol) has been creat
ed for key: status. This may result in invalid serialization.  It is recommended to log an issue to the responsible developer/development team.
[2021-11-27T20:02:20,994][WARN ][org.logstash.instrument.metrics.gauge.LazyDelegatingGauge] A gauge metric of an unknown type (org.jruby.RubySymbol) has been created for
 key: status. This may result in invalid serialization.  It is recommended to log an issue to the responsible developer/development team.
[2021-11-27T20:02:23,499][INFO ][logstash.filters.geoip.downloadmanager] new database version detected? false
[2021-11-27T20:02:23,769][INFO ][logstash.filters.geoip.databasemanager][main] By not manually configuring a database path with `database =>`, you accepted and agreed Ma
xMind EULA. For more details please visit https://www.maxmind.com/en/geolite2/eula
[2021-11-27T20:02:23,772][INFO ][logstash.filters.geoip    ][main] Using geoip database {:path=>"sensor39/plugins/filters/geoip/1638021268/GeoLite2-City.mmdb"}
[2021-11-27T20:02:23,912][INFO ][logstash.javapipeline    ][main] Starting pipeline {:pipeline_id=>"main", "pipeline.workers"=>4, "pipeline.batch.size"=>125, "pipeline.b
atch.delay"=>50, "pipeline.max_inflight"=>500, "pipeline.sources"=>["/home/kalilinux/nginx.conf"], :thread=>"#<Thread:0x7f1aeaef run>"}
[2021-11-27T20:02:25,396][INFO ][logstash.javapipeline    ][main] Pipeline Java execution initialization time {"seconds"=>1.48}
[2021-11-27T20:02:25,492][INFO ][logstash.javapipeline    ][main] Pipeline started {"pipeline.id"=>"main"}
[2021-11-27T20:02:25,608][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[]}
[2021-11-27T20:02:25,617][INFO ][filewatch.observingtail  ][main][dcce2e62ea9c843ce55d0655d2b3ace1cf2e6cb1c183a7d890a2aa99ee8db256] START, creating Discoverer, Watch wit
h file and sincedb collections
[2021-11-27T20:02:27,138][INFO ][logstash.outputs.file    ][main][572e934efa8b2c585f89365dab120384ce2b50bda81852589819fa7b8bac2ccf] Opening file {:path=>"/var/log/logsta
sh/geo_parsed_nginx.log"}
```

Now we can see the parsed files in /var/logs/logstash as follows



The output file in our configuration file was named as geo_parsed_nginx.log which now has the following contents(continent_code,country_name,country_code,latitude,timezone,location etc)