EP10: What do WhatsApp, Discord, and Facebook Messenger have in common?

ALEX XU
JUN 4, 2022

♡ 123 4 Share •••

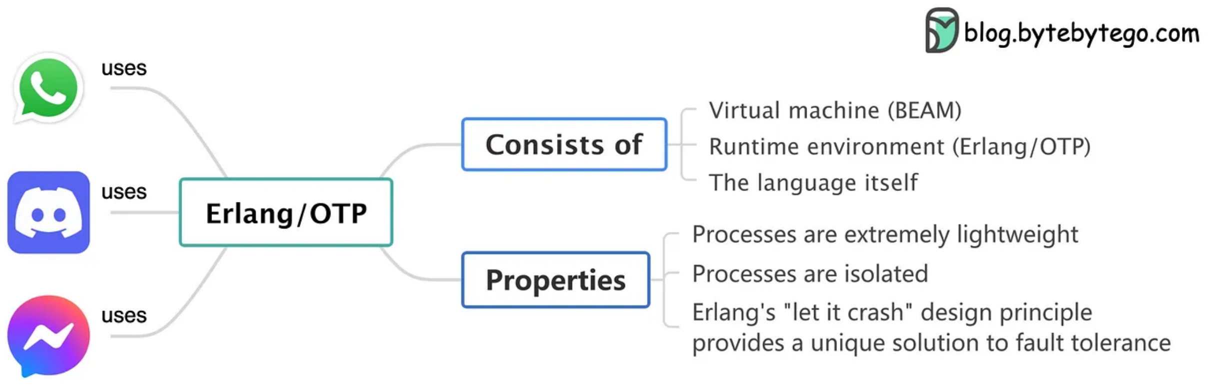
In this newsletter, we will talk about the following:

- What do WhatsApp, Discord, and Facebook Messenger have in common?
- ELK Stack
- How do companies typically ship code to production
- Linux file permission illustrated

What do WhatsApp, Discord, and Facebook Messenger have in common?

What do WhatsApp, Discord, and Facebook Messenger have in common? Authored by Sahn Lam.

They are all powered by Erlang. What is so special about Erlang that makes it the technology choice behind these popular chat services?


blog.bytebytego.com
uses
uses
uses
Erlang/OTP
Consists of
Virtual machine (BEAM)
Runtime environment (Erlang/OTP)
The language itself
Properties
Processes are extremely lightweight
Processes are isolated
Erlang's "let it crash" design principle provides a unique solution to fault tolerance

Erlang was developed in the 80’s by Ericsson for telecom switches that demanded extreme reliability. It is upon this rock-solid foundation that these modern chat services were built.

When people talk about Erlang, they are usually referring to the entire ecosystem called Erlang/OTP. It consists of the VM (BEAM), runtime environment (Erlang/OTP), and the language itself. They work together to provide this highly fault-tolerant programming environment.

There are several secret ingredients that make Erlang special.

Erlang processes are extremely lightweight. It is fast and cheap to create an Erlang process. A big machine could have millions of these processes.

Erlang processes are isolated. They communicate with each other only through messages. It is easy to send a message to another process, whether it is on the same machine or a different machine. This makes it easy to scale an Erlang application either horizontally by adding more machines, or vertically by adding more cores.

Erlang’s “let it crash” design principle provides a unique solution to fault tolerance. Erlang implementers viewed software and hardware faults as inevitable. Erlang implements a concept called supervisor tree to allow Erlang applications to recover from these faults quickly and reliably. A supervisor monitors its child processes and decides how to recover when a fault occurs. This is central to any well-designed Erlang application.

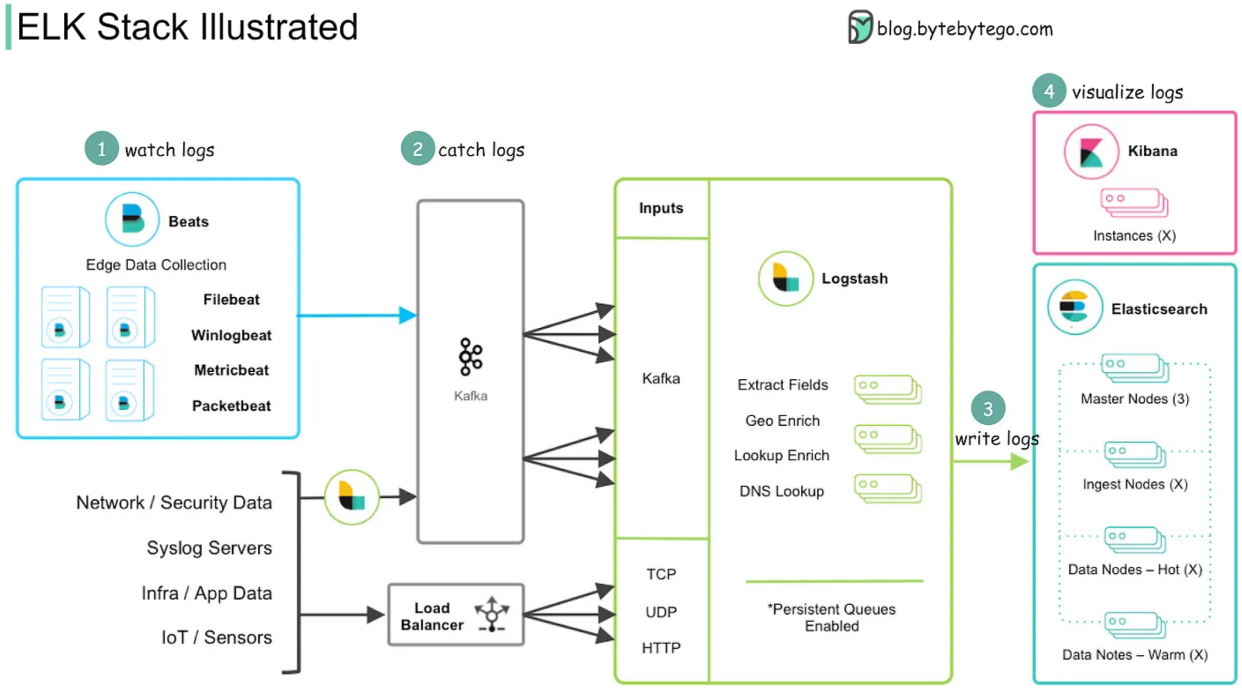
There are now new languages built on top of the Erlang/OTP foundation. These languages improve the developer experience while relying on the rock-solid fault-tolerant foundation that is Erlang/OTP.

Besides chat apps, Erlang is used in many other mission-critical applications. Can you name any other well-known Erlang applications?

ELK Stack

What is ELK Stack and why is it so popular for log management?

The ELK Stack is composed of three open-source products. ELK stands for Elasticsearch, Logstash, and Kibana.



- 🔹 Elasticsearch is a full-text search and analysis engine, leveraging Apache Lucene search engine as its core component.

- 🔹 Logstash collects data from all kinds of edge collectors, then transforms that data and sends it to various destinations for further processing or visualization.

In order to scale the edge data ingestion, a new product Beats is later developed as lightweight agents installed on edge hosts to collect and ship logs to Logstash.

- 🔹 Kibana is a visualization layer with which users analyze and visualize the data.

The diagram below shows how ELK Stack works:

Step 1 - Beats collects data from various data sources. For example, Filebeat and Winlogbeat work with logs, and Packetbeat works with network traffic.

Step 2 - Beats sends data to Logstash for aggregation and transformation. If we work with massive data, we can add a message queue (Kafka) to decouple the data producers and consumers.

Step 3 - Logstash writes data into Elasticsearch for data indexing and storage.

Step 4 - Kibana builds on top of Elasticsearch and provides users with various search tools and dashboards with which to visualize the data.

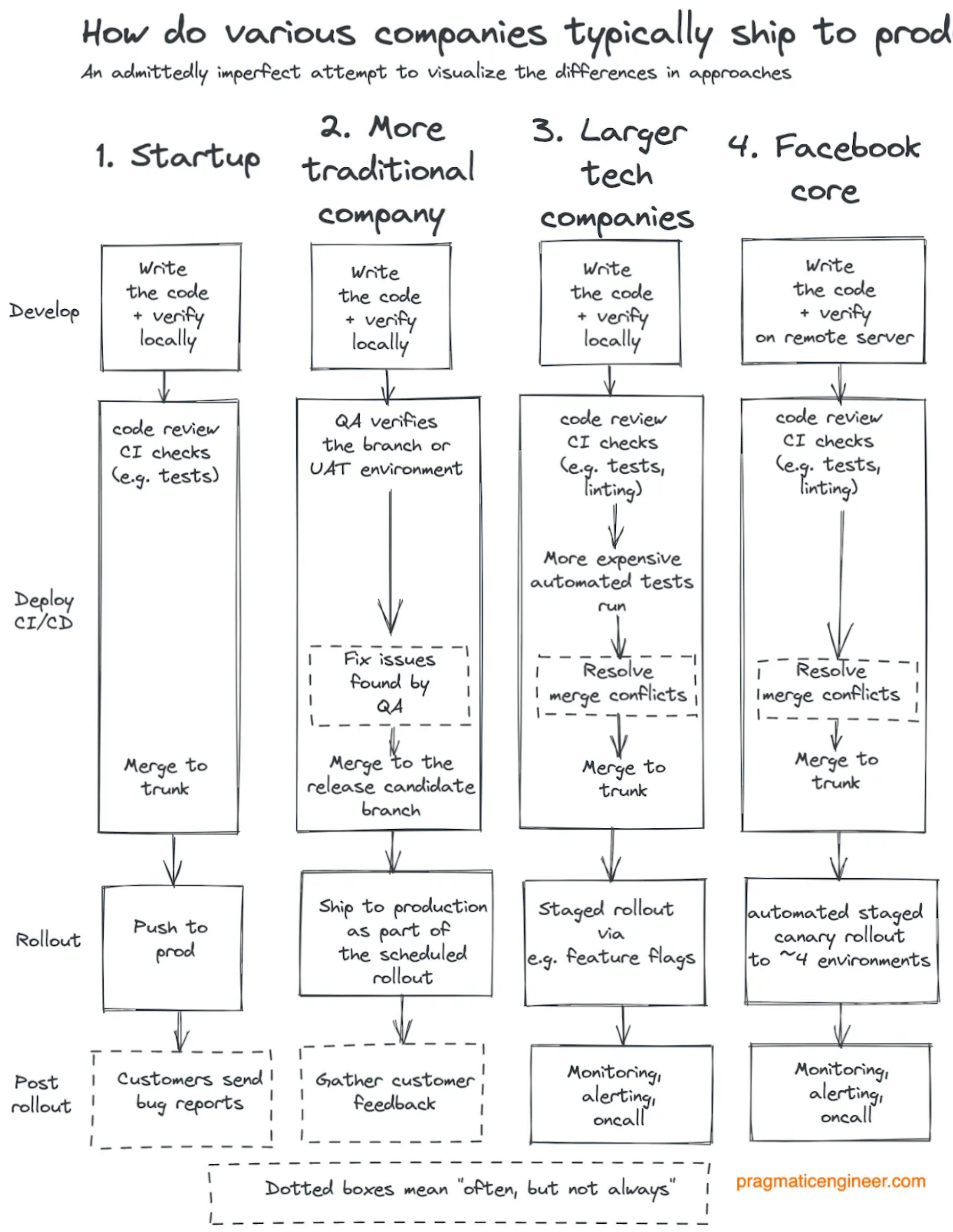
ELK Stack is pretty convenient for troubleshooting and monitoring. It became popular by providing a simple and robust suite in the log analytics space, for a reasonable price.

Over to you: which other log management products have you used in production? How do they compare with ELK Stack?

Image source: https://www.elastic.co/guide/en/logstash/current/deploying-and-scaling.html

How do companies typically ship code to production?

Gergely Orosz wrote an excellent article about this topic and he has kindly agreed to share excerpts with newsletter readers.


How do various companies typically ship to production?
An admittedly imperfect attempt to visualize the differences in approaches
1. Startup
2. More traditional company
3. Larger tech companies
4. Facebook core
Develop
Deploy CI/CD
Rollout
Post rollout
Write the code + verify locally
code review CI checks (e.g. tests)
Merge to trunk
Push to prod
Customers send bug reports
Write the code + verify locally
QA verifies the branch or UAT environment
Fix issues found by QA
Merge to the release candidate branch
Ship to production as part of the scheduled rollout
Gather customer feedback
Write the code + verify locally
code review CI checks (e.g. tests, linting)
More expensive automated tests run
Resolve merge conflicts
Merge to trunk
Staged rollout via e.g. feature flags
Monitoring, alerting, oncall
Write the code + verify on remote server
code review CI checks (e.g. tests, linting)
Resolve merge conflicts
Merge to trunk
automated staged canary rollout to ~4 environments
Monitoring, alerting, oncall
Dotted boxes mean "often, but not always"
pragmaticengineer.com

1. Startups: Typically do fewer quality checks than other companies.

Startups tend to prioritize moving fast and iterating quickly, and often do so without much of a safety net. This makes perfect sense if they don't – yet – have customers. As the company attracts users, these teams need to start to find ways to not cause regressions or ship bugs. They then have the choice of going down one of two paths: hire QAs or invest in automation.

2. Traditional companies: Tend to rely more heavily on QAs teams.

While automation is sometimes present in more traditional companies, it's very typical that they rely on large QA teams to verify what they build. Working on branches is also common; it's rare to have trunk-based development in these environments.

3. Large tech companies: Typically invest heavily in infrastructure and automation related to shipping with confidence.

These investments often include automated tests running quickly and delivering rapid feedback, canarying, feature flags and staged rollouts.

4. Facebook core: Has a sophisticated and effective approach few other companies possess.

Facebook's core product is an interesting one. It has fewer automated tests than many would assume, but, on the other hand, it has an exceptional automated canarying functionality, where the code is rolled out through 4 environments: from a testing environment with automation, through one that all employees use, through a test market of a smaller region, to all users. In every stage, if the metrics are off, the rollout automatically halts.

Over to you: how does your company ship code to production? Does well does it work?

If you want to read the full article, you can find it here:

The Pragmatic Engineer

Shipping to Production

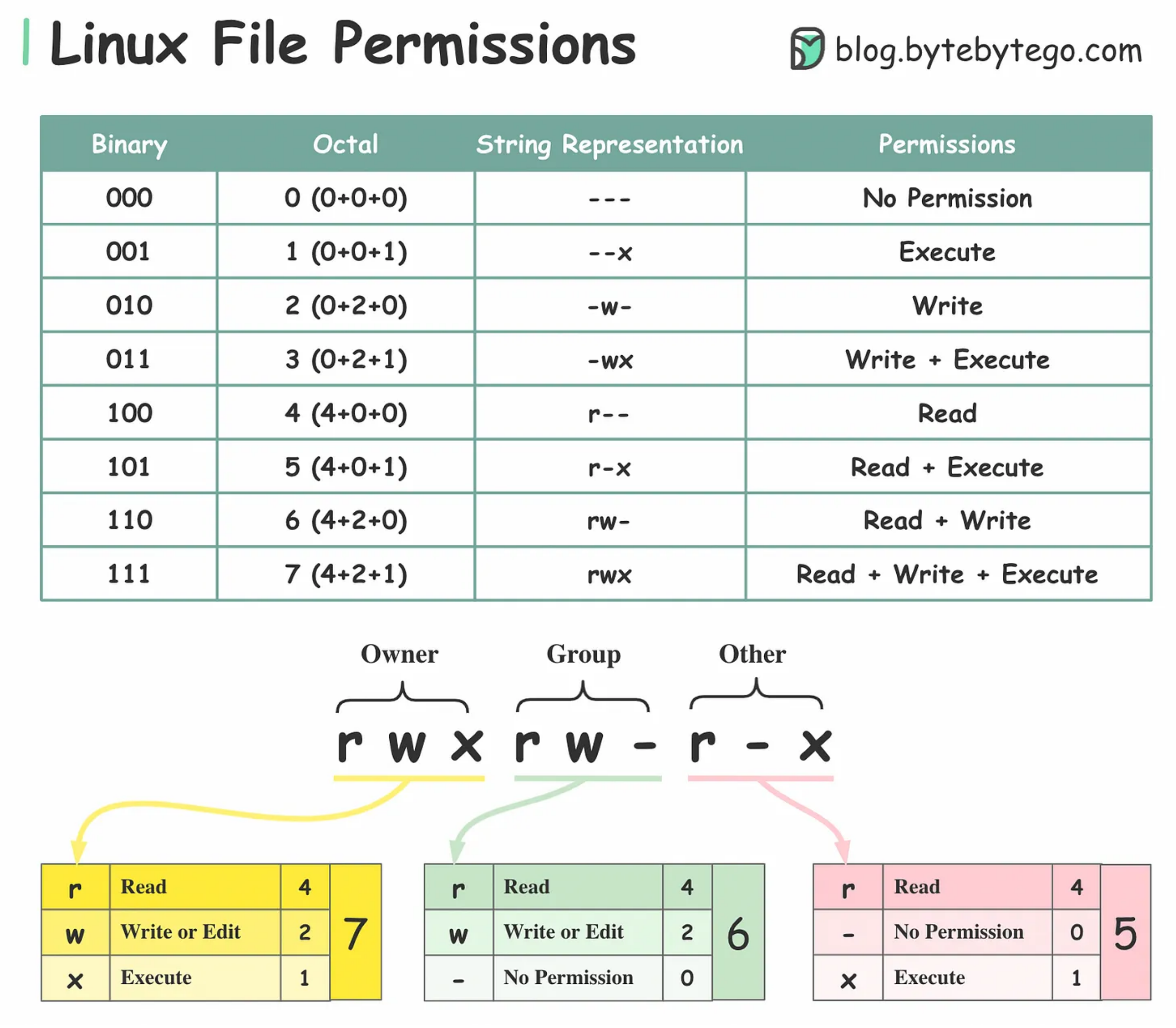
👋 Hi, this is Gergely with a free issue of the Pragmatic Engineer Newsletter. If you’re not a full subscriber yet, you missed the second part of Inside Facebook’s engineering culture and the one The tech boom potentially being over. Subscribe to get this newsletter every week 👇 ...

Read more

a year ago · 108 likes · 13 comments · Gergely Orosz

Linux file permission illustrated.

To understand Linux file permissions, we need to understand Ownership and Permission.



Linux File Permissions

blog.bytebytego.com

Binary	Octal	String Representation	Permissions
000	0 (0+0+0)	---	No Permission
001	1 (0+0+1)	--x	Execute
010	2 (0+2+0)	-w-	Write
011	3 (0+2+1)	-wx	Write + Execute
100	4 (4+0+0)	r--	Read
101	5 (4+0+1)	r-x	Read + Execute
110	6 (4+2+0)	rw-	Read + Write
111	7 (4+2+1)	rwx	Read + Write + Execute

r	Read	4	7
w	Write or Edit	2	
x	Execute	1	

r	Read	4	6
w	Write or Edit	2	
-	No Permission	0	

r	Read	4	5
-	No Permission	0	
x	Execute	1	

Ownership

Every file or directory is assigned 3 types of owner:

- Owner: the owner is the user who created the file or directory.
- Group: a group can have multiple users. All users in the group have the same permissions to access the file or directory.

- Other: other means those users who are not owners or members of the group.

Permission

There are only three types of permissions for a file or directory.

- Read (r): the read permission allows the user to read a file.
- Write (w): the write permission allows the user to change the content of the file.
- Execute (x): the execute permission allows a file to be executed.

Over to you: what are some of the commonly used Linux commands to change file permissions?

Our Books:

Our bestselling book "System Design Interview - An Insider's Guide" is available in both paperback and digital format.

Paperback edition: https://geni.us/XxCd

Digital edition: https://bit.ly/3lg41jK

123 Likes

4 Comments

Write a comment...

Rajesh Kanade Writes Cornucopia Jun 6, 2022 ♥ Liked by Alex Xu

CouchDB , RabbitMQ are some well known products written in Erlang

♡ LIKE (6) REPLY SHARE ...

Indian Jun 13, 2022 ♥ Liked by Alex Xu

Hi Alex, Thank you for providing such a great content for learning. It help me lot to gain insight on any topic. I would have 1 humble request. If possible, please post the reference material also. So if anyone wants to go beyond the discussion and want to in sync with the details you provided. Thanks,

♡ LIKE (1) REPLY SHARE ...

1 reply by Alex Xu

2 more comments...