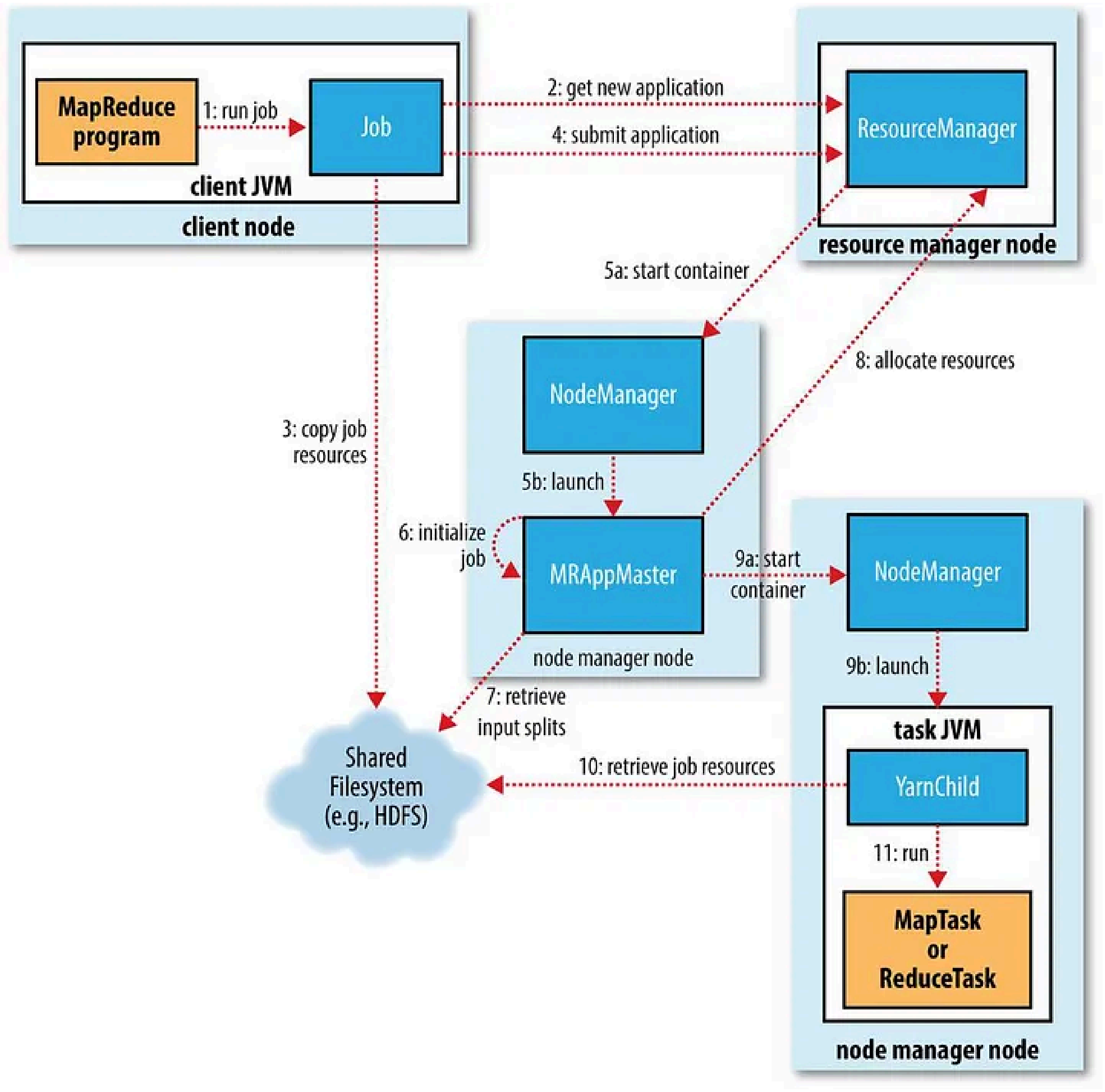
Anatomy of MapReduce

Joshua U · Follow

5 min read · Feb 16, 2023



There are five independent entities:

- The client, which submits the `MapReduce` job.
- The YARN resource manager, which coordinates the allocation of compute resources on the cluster.
- The YARN node managers, which launch and monitor the compute containers on machines in the cluster.
- The `MapReduce` application master, which coordinates the tasks running the `MapReduce` job The application master and the `MapReduce` tasks run in containers that are scheduled by the resource manager and managed by the node managers.
- The distributed filesystem, which is used for sharing job files between the other entities.

Job Submission :

- The `submit()` method on Job creates an internal `JobSubmitter` instance and calls `submitJobInternal()` on it.
- Having submitted the job, `waitForCompletion` polls the job's progress once per second and reports the progress to the console if it has changed since the last report.
- When the job completes successfully, the job counters are displayed Otherwise, the error that caused the job to fail is logged to the console.

The job submission process implemented by `JobSubmitter` does the following:

- Asks the resource manager for a new application ID, used for the `MapReduce` job ID.
- Checks the output specification of the job For example, if the output directory has not been specified or it already exists, the job is not submitted and an error is thrown to the `MapReduce` program.
- Computes the input splits for the job If the splits cannot be computed (because the input paths don't exist, for example), the job

is not submitted and an error is thrown to the `MapReduce` program.

- Copies the resources needed to run the job, including the job JAR file, the configuration file, and the computed input splits, to the shared filesystem in a directory named after the job ID.
- Submits the job by calling `submitApplication()` on the resource manager.

Job Initialization :

- When the resource manager receives a call to its `submitApplication()` method, it hands off the request to the YARN scheduler.
- The scheduler allocates a container, and the resource manager then launches the application master's process there, under the node manager's management.
- The application master for `MapReduce` jobs is a Java application whose main class is `MRAppMaster` .
- It initializes the job by creating a number of bookkeeping objects to keep track of the job's progress, as it will receive progress and completion reports from the tasks.
- It retrieves the input splits computed in the client from the shared filesystem.
- It then creates a map task object for each split, as well as a number of reduce task objects determined by the `mapreduce.job.reduces` property (set by the `setNumReduceTasks()` method on Job).

Task Assignment:

- If the job does not qualify for running as an uber task, then the application master requests containers for all the map and reduce tasks in the job from the resource manager .
- Requests for map tasks are made first and with a higher priority than those for reduce tasks, since all the map tasks must complete before

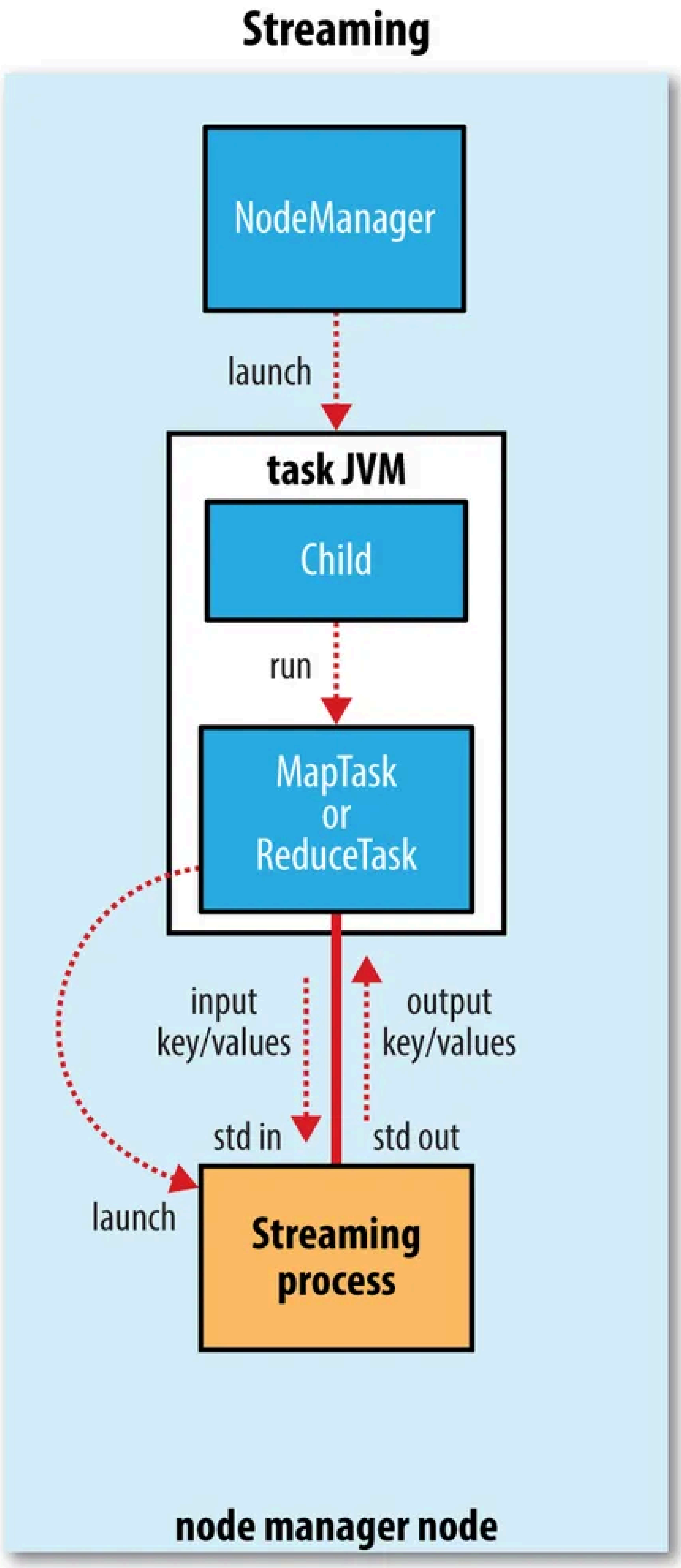
the sort phase of the reduce can start.

- Requests for reduce tasks are not made until 5% of map tasks have completed.

Task Execution:

- Once a task has been assigned resources for a container on a particular node by the resource manager's scheduler, the application master starts the container by contacting the node manager.
- The task is executed by a Java application whose main class is `YarnChild`. Before it can run the task, it localizes the resources that the task needs, including the job configuration and JAR file, and any files from the distributed cache.
- Finally, it runs the map or reduce task.

Streaming:


Streaming
NodeManager
launch
task JVM
Child
run
MapTask or ReduceTask
input key/values
output key/values
std in
std out
launch
Streaming process
node manager node

- Streaming runs special map and reduce tasks for the purpose of launching the user supplied executable and communicating with it.
- The Streaming task communicates with the process (which may be written in any language) using standard input and output streams.
- During execution of the task, the Java process passes input key value pairs to the external process, which runs it through the user defined map or reduce function and passes the output key value pairs back to the Java process.
- From the node manager's point of view, it is as if the child process ran the map or reduce code itself.

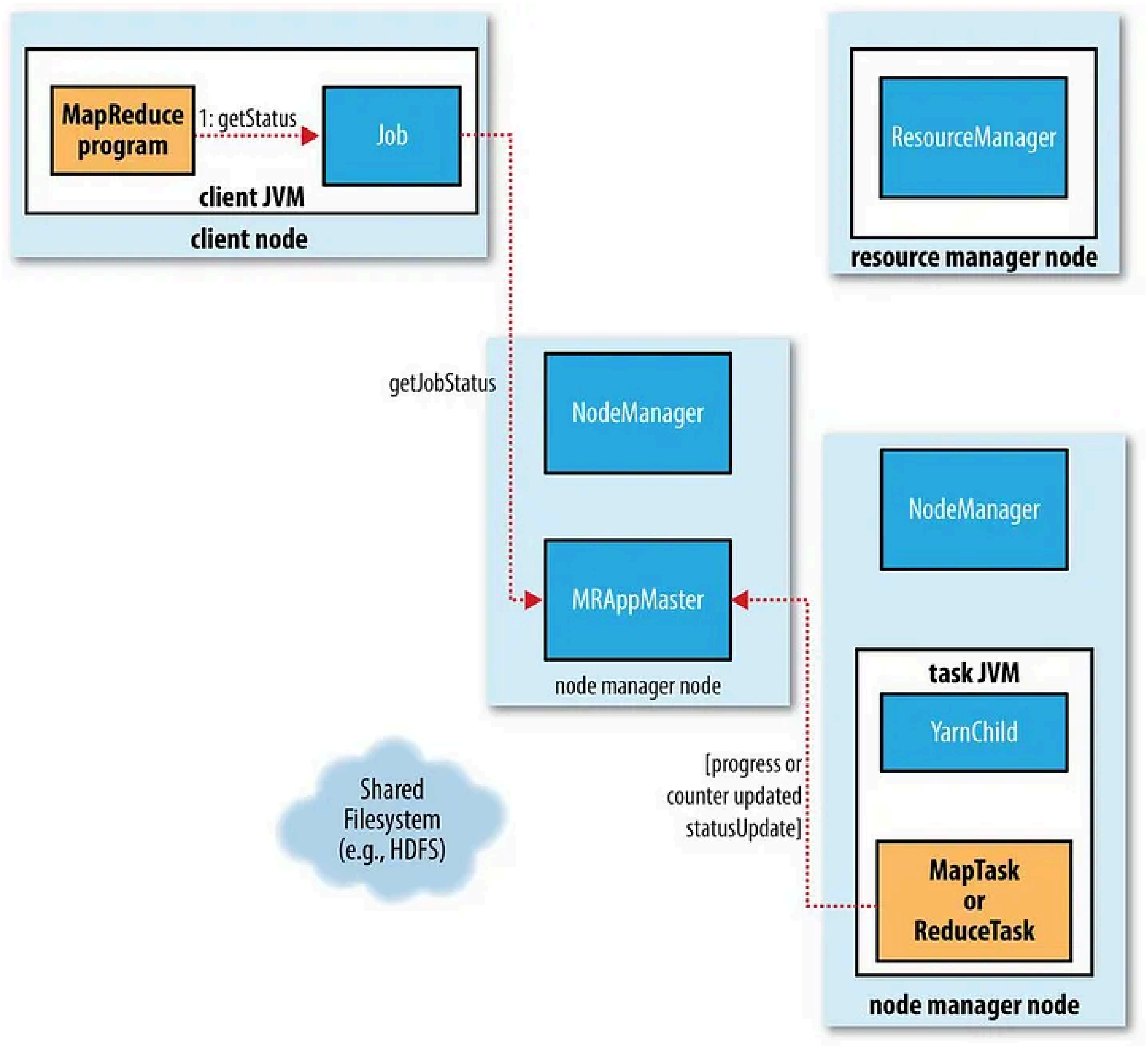
Progress and status updates :

- `MapReduce` jobs are long running batch jobs, taking anything from tens of seconds to hours to run.
- A job and each of its tasks have a status, which includes such things as the state of the job or task (e g running, successfully completed, failed), the progress of maps and reduces, the values of the job's counters, and a status message or description (which may be set by user code).
- When a task is running, it keeps track of its progress (i e the proportion of task is completed).
- For map tasks, this is the proportion of the input that has been processed.
- For reduce tasks, it's a little more complex, but the system can still estimate the proportion of the reduce input processed.

It does this by dividing the total progress into three parts, corresponding to the three phases of the shuffle.

- As the map or reduce task runs, the child process communicates with its parent application master through the umbilical interface.
- The task reports its progress and status (including counters) back to its application master, which has an aggregate view of the job, every

three seconds over the umbilical interface.



How status updates are propagated through the MapReduce System

- The resource manager web UI displays all the running applications with links to the web UIs of their respective application masters, each of which displays further details on the `MapReduce` job, including its progress.
- During the course of the job, the client receives the latest status by polling the application master every second (the interval is set via `mapreduce.client.progressmonitor.pollinterval`).

Job Completion:

- When the application master receives a notification that the last task for a job is complete, it changes the status for the job to Successful.
- Then, when the Job polls for status, it learns that the job has completed successfully, so it prints a message to tell the user and then returns from the `waitForCompletion()` .
- Finally, on job completion, the application master and the task containers clean up their working state and the OutputCommitter's `commitJob ()` method is called.
- Job information is archived by the job history server to enable later interrogation by users if desired.

References:

Hadoop: The Definitive Guide: https://www.oreilly.com/library/view/hadoop-the-definitive/9780596521974/

Hadoop Mapreduce Hadoop Mapreduce

Follow

Written by Joshua U

88 Followers · 6 Following

Python Enthusiast, Assistant Professor, Care for developing

No responses yet

What are your thoughts?

Respond

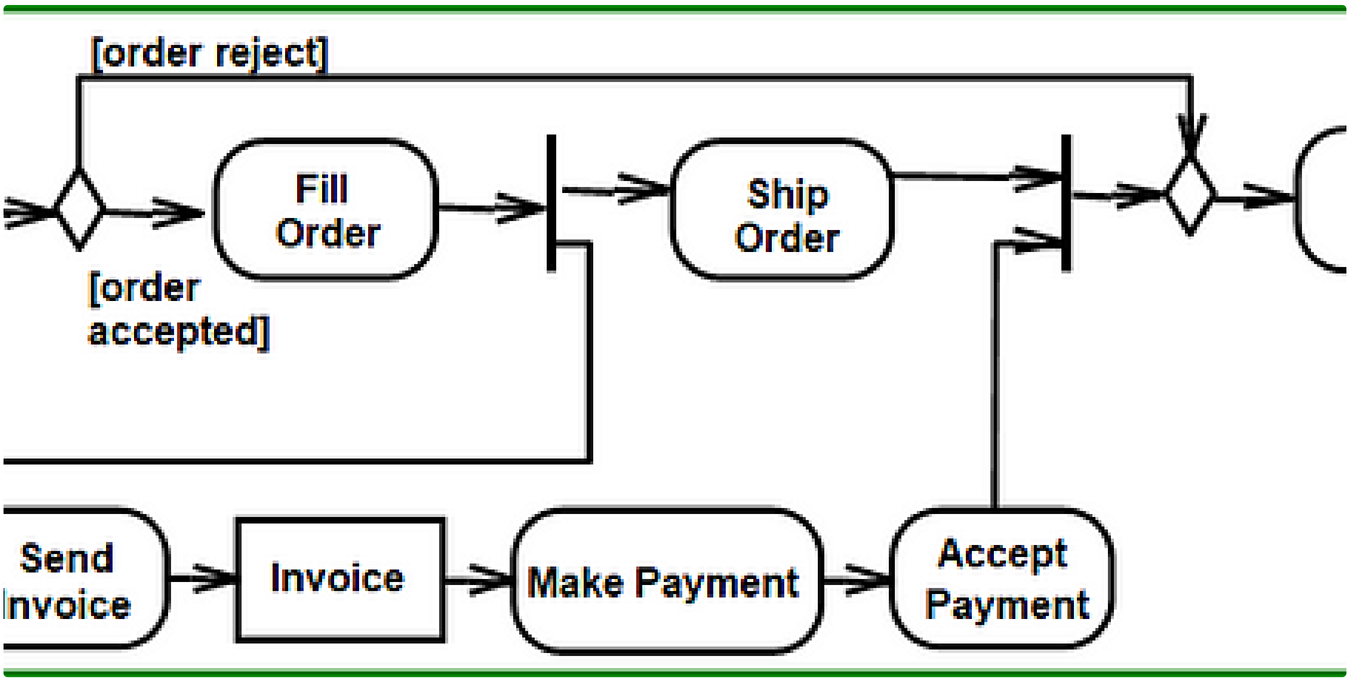
More from Joshua U

Joshua U

Linux Process calls: Creating process using fork()

The fork system call is used for creating a new process in Unix/Linux. The child process created by the process that makes the fork()…

Aug 29, 2023 3 1

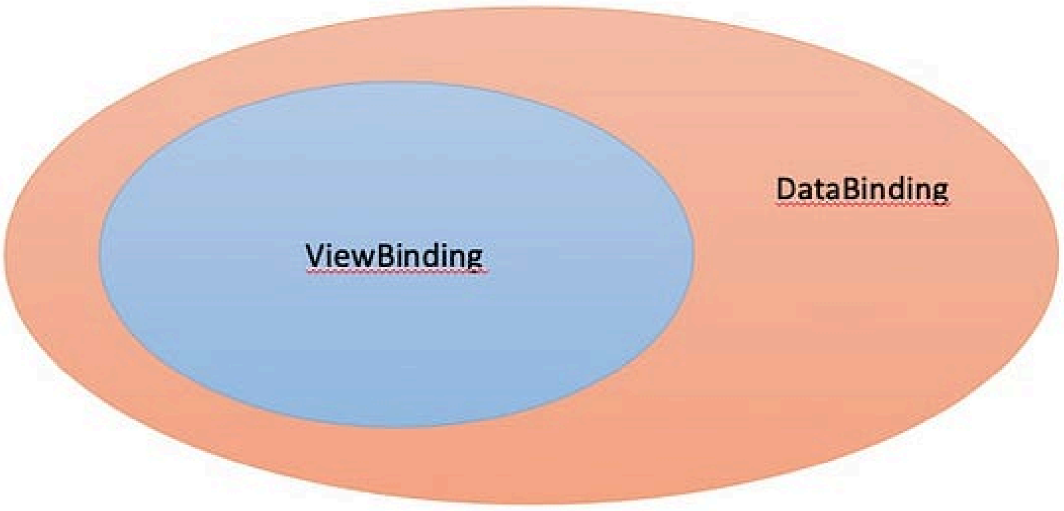


Joshua U

UML: Activity Diagram

Activity diagram is useful to specify system behavior. It is supplements to use-case by providing a diagrammatic representation of...

Oct 2, 2021 5



Joshua U

Android: View Binding vs Data Binding

View binding:

Apr 20, 2022 12 2

Joshua U

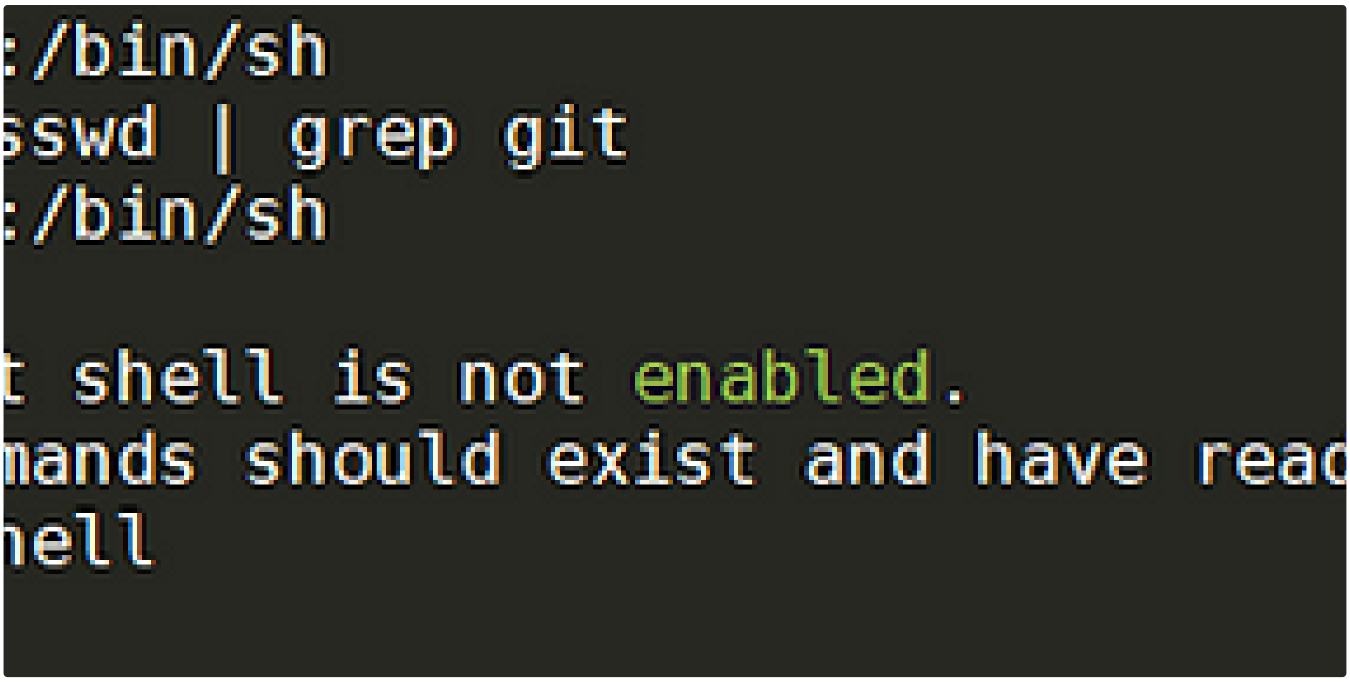
Linux System calls: open

This system call is used to open a file and obtain a file descriptor. open function is a key file-related system call in Linux that is used…

Aug 9, 2023

See all from Joshua U

Recommended from Medium



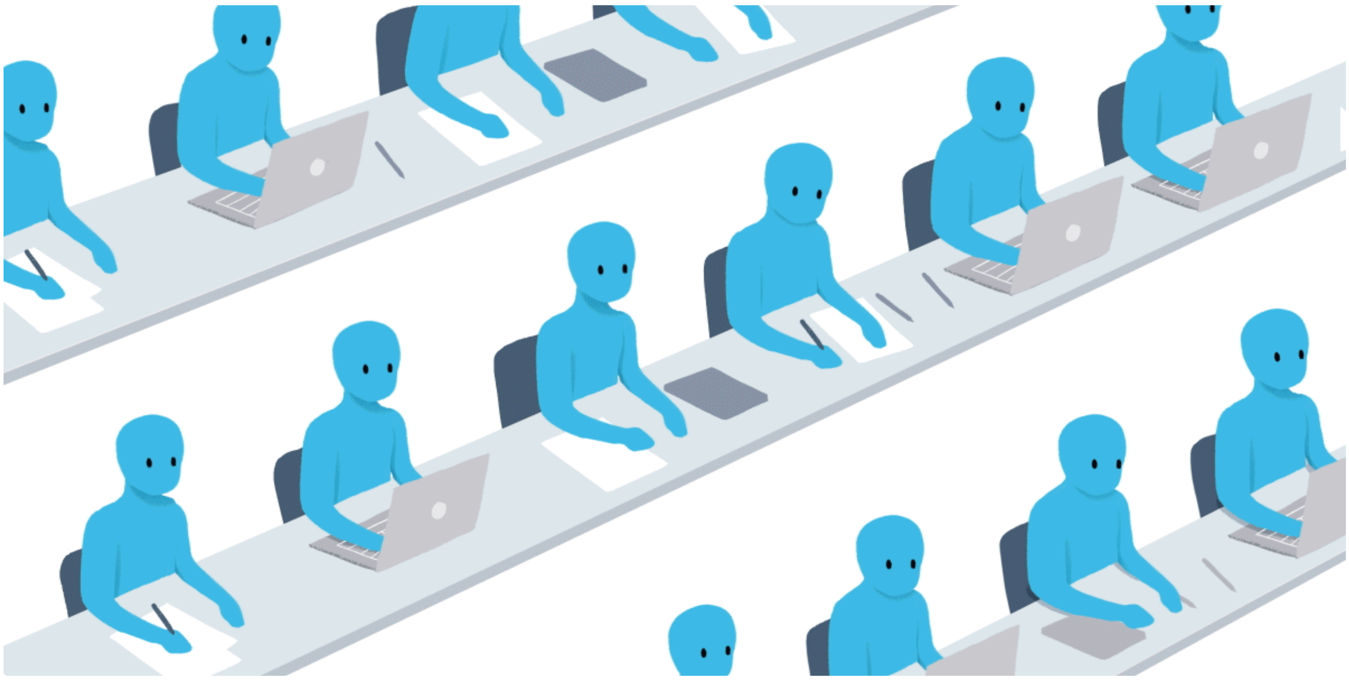
```
:/bin/sh
sswd | grep git
:/bin/sh

t shell is not enabled.
nands should exist and have read
nell
```

Jasenko Krejić

Using Git Shell for Access Control via SSH

You have heard of Git, and most probably, heard of shell as well. But do you know what Git-shell is and what it is used for? I will be...

Oct 7, 2024



In Human Parts by Devon Price

Laziness Does Not Exist

Psychological research is clear: when people procrastinate, there's usually a good reason

Mar 24, 2018 339K 2042

Lists

Staff picks
806 stories · 1599 saves

Stories to Help You Level-Up at Work
19 stories · 928 saves

Self-Improvement 101
20 stories · 3254 saves

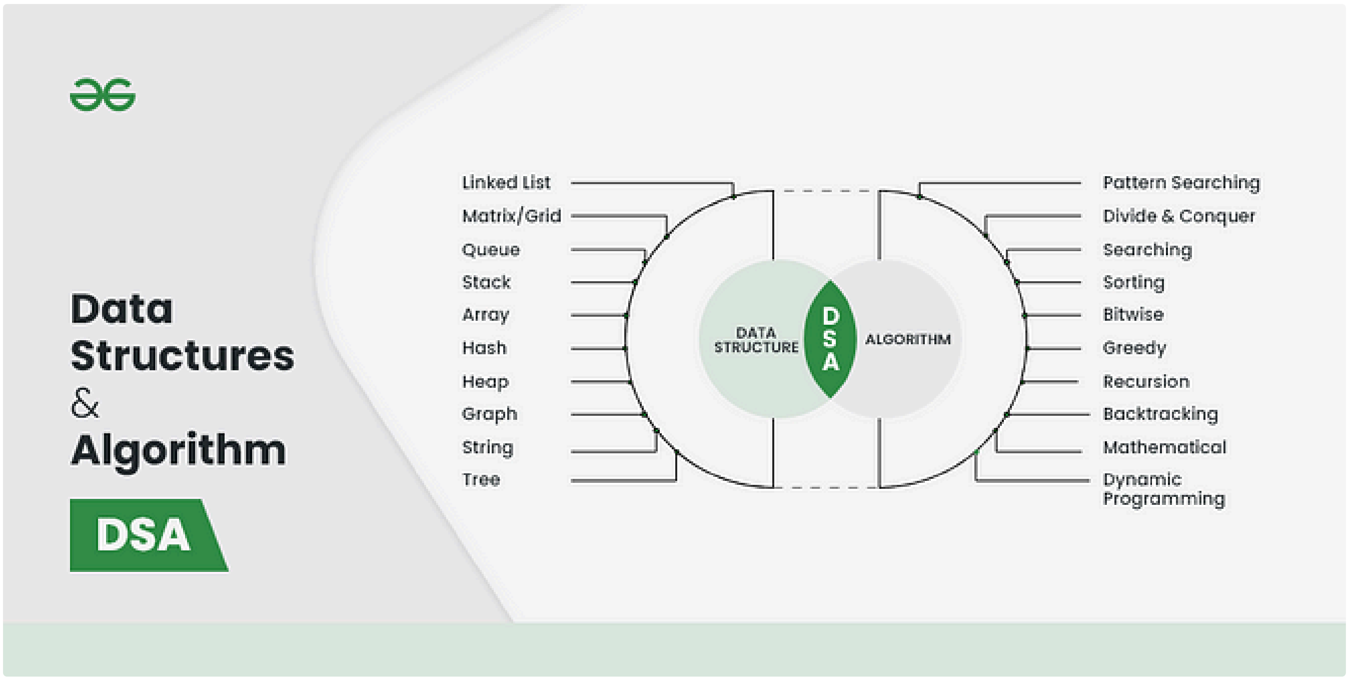
Productivity 101
20 stories · 2749 saves

In Python in Plain English by Kiran Maan

Just Stop Writing Python Functions Like This!!!

I just reviewed someone else's code and I was just shocked.

Jan 20 1.6K 43

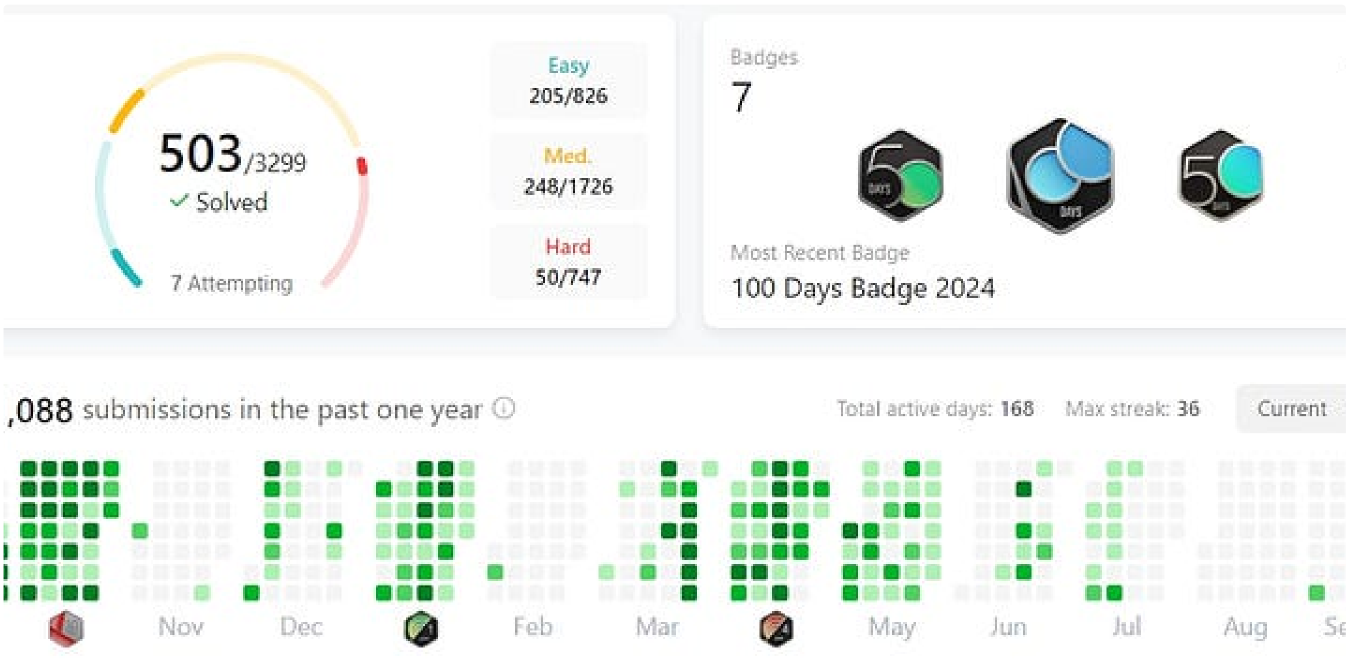


Sai Parvathaneni

Ultimate DSA Interview Prep: 7/12 Sliding Window

A curated list of LeetCode problems covering key concepts and techniques to help you prepare for your next DSA interview.

Sep 11, 2024 26

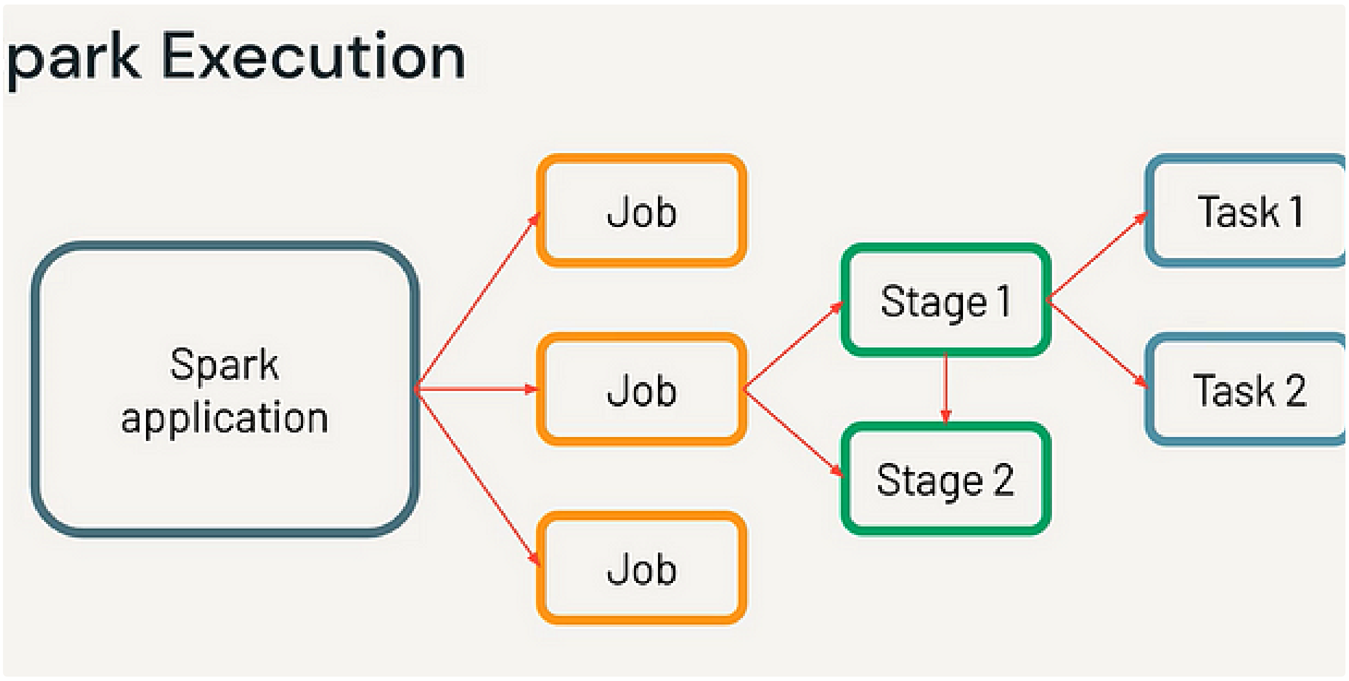


In Code Like A Girl by Surabhi Gupta

Why 500 LeetCode Problems Changed My Life

How I Prepared for DSA and Secured a Role at Microsoft

Sep 26, 2024 3.7K 78



Swayamprava Panda

Understanding Jobs, Stages, Tasks, and Execution Flow in Apache Spark

Apache Spark is a powerful distributed computing framework that efficiently processes large-scale data. To fully leverage Spark's…

Aug 6, 2024 4

See more recommendations