PT. Bimasakti DevOps Engineer Test

Candidate: Aan Kunaifi

1. One day, your “Kirito” becomes a high traffic service that runs on production that is accessed publicly by the end user. But sometimes the crashlytics says your endpoint returns a 5xx error. What is your way of figuring out the root cause?

5xx error is a very broad problem that can occur in our application or services, so it would be best to narrow it down. This error most of the time is related to the server, network or the application. It is important to have a proper alerting system so that if the error occurs we can be notified immediately in less than 5 minutes. First I would check on the service and network logs that are stored in elasticsearch or other log management tools by querying the logs from their dashboard like kibana or grafana, if you don’t have log management tools you have no choice but to connect to the related server or services that you suspect are affected and check their linux logs or docker logs. On kibana/grafana dashboard try query the keywords like 500, 501, 502,”error” etc and look for any suspicious logs and note which services are affected, read the problematic logs carefully and consult with your teammate on how to tackle this problem they might know more about the problem then you, try to test the service request yourself and see if the error still occur, share your knowledge on what you find to the communication channel and raise this as an incident to your communication channel, everyone needs to know what is happening and track the issue regularly.

2. Once you found the root cause, what is your resolution to resolve the issue so it won’t happen in the future?
Once you found the suspicious logs that might be the root cause, handle it depending on the error type itself, for example if the error 500, we can check the CPU, Memory, Disk status, make sure they are in good health, if no problems found, look for application service status like in kubernetes check the pods or in docker check the running container, make sure the databases are in good health, server configuration, exported variables, certificate expiry and check the network connection as well. If we find a problem, report immediately to the communication channel and handle it based on what type of problem it is, if you are not able to handle it alone, ask for help in the communication channel, constant communication is needed when handling problems so that everyone knows the incident is handled properly until it is resolved. If you still can't find the problem it could be 501 error not implemented where client request is not understandable by server, 502 Bad gateway, check on the firewall or proxy config, 503 service unavailable that might occur because of service maintenance, 504 gateway timeout there might be an issue with the network or service timeouts, trace the request so that we know where the request stops. Remember to communicate each step you take to other team members and once you resolve the issue, document all the steps you take to resolve the issue so that when the same incident happens you can check the documentation for it and resolve it quickly, incident is always happening, the most important thing is always learn from mistakes and document the knowledge effectively.

3. Mention and describe all tools that you use to help to trace and resolve the issue
Communication:
Slack or Discord Teamwork communication and coordination.
Notion or trello for documentation management, document the incident handling process so that others can learn to mitigate as well.

Logs and monitoring:
Open Telemetry, fluentd, sentry, or logstash, for collecting service logs and server logs.
Prometheus or elasticsearch for managing or storing those logs.
Grafana or kibana for displaying logs and metrics this is where we check when incidents happened

Programming Language:
Bash as the linux server language, Python for automation,
query languages such as kibana query language or PromQL etc for querying logs
Golang, PHP, Nodejs or other backend language to help troubleshoot application code problems
Terraform or other IaC languages for managing infrastructure as code.

CI/CD:
Git, Gitlab, Docker for handling development pipelines