Observing Cloud Resources

*SRE Assessment Template*

# Categorize Responsibilities

|  |  |
| --- | --- |
| **Prometheus and Grafana Screenshots** | |
| Provide a screenshot of the Prometheus node\_exporter service running on the EC2 instance. Use the following command to show that the system is running: sudo systemctl status node\_exporter | |
|  | |
| **Host Metric**  **(CPU, RAM, Disk, Network)** | **Dashboard** |
| *CPU %* |  |
| *Available Memory* |  |
| *Disk I/O* |  |
| *Network Received* |  |
| **Responsibilities** | |
| 1. The development team wants to release an emergency hotfix to production. Identify two roles of the SRE team who would be involved in this and why. | |
| * Release Manager: In charge of code releases, executes the release and rollback procedures. * Monitoring Engineer: Monitor the system to see if the release cause any problems to system. | |
| 2. The development team is in the early stages of planning to build a new product. Identify two roles of the SRE team that should be invited to the meeting and why. | |
| * System Architect: Participate for the document/diagram infrastructure. Also to make recommendations for new technologies. * Infrastructure Engineer: Since they still contribute 50% to development tasks, they should be involved in the meeting. Also their responsibility is to plan/execute system patches, they should participate in the planning stage to have a better understanding of the system. | |
| 3. The emergency hotfix from question 1 was applied and is causing major issues in production. Which SRE role would primarily be involved in mitigating these issues? | |
| * Release Manager: Since a major issues occurred, it is best to rollback the release to mitigate the damages. * Monitoring Engineer: Usually the first person to know about the incident, they should contact the Release Manager to take actions upon this incident. Then the Monitoring Engineer should diagnose dashboards/metrics to find the root cause of the problem. | |

# 

# Team Formation and Workflow Identification

|  |
| --- |
| **API Monitoring and Notifications** |
| Display the status of an API endpoint: Provide a screenshot of the Grafana dashboard that will show at which point the API is unhealthy (non-200 HTTP code), and when it becomes healthy again (200 HTTP code). |
|  |
| Create a notification channel: Provide a screenshot of the Grafana notification which shows the summary of the issue and when it occurred. |
|  |
| Configure alert rules: Provide a screenshot of the alert rules list in Grafana. |
|  |

# 

# Applying the Concepts

|  |
| --- |
| **Graph 1** |
|  |
| 4a. Given the above graph, where does it show that the API endpoint is down? Where on the graph does this show that the API is healthy again? |
| * Around 15:27, the API endpoint is down. * Then around 15:36, the API is back to its healthy state. |
| 4b. If there was no SRE team, how would this outage affect customers? |
| Without an SRE team, this outage would cost interruption and bad experience for customers. Also if there weren’t any dashboards or metrics to monitor, the team will have a much more difficult time tracing the errors and the outage would be longer. |
| 4c. What could be put in place so that the SRE team could know of the outage before the customer does? |
| An alert should be put in place so that as soon as something unexpected happens, the SRE team will be noticed and take actions asap. Therefore, by the time the customer notice the outage, SRE team is already on it and they might have already fixed the problem and releasing it. |

|  |
| --- |
| **Graph 2** |
|  |
| 5a. Given the above graph, which instance had the increase in traffic, and approximately how many bytes did it receive (feel free to round)? |
| The instance 10.0.0.68 had the increase in traffic.  Approximately, it received 1500 bytes. |
| 5b. Which team members on the SRE team would be interested in this graph and why? |
| * Monitoring Engineer: They are the ones who create the dashboard and any alert rules if needed, they should have the most interest in this. |

# 

# 