**Coalfire Technical Challenge**

**Due:** <DAY>, <DATE> @ 9:00 AM EST

Before you begin, please note:

1. You must perform this challenge by yourself, no other persons may assist you.
2. Try to accomplish as many tasks as you can within the time period allotted. Quality of the implementation is an important factor.
3. You are strongly encouraged to search the web and use resources like Stack Overflow and GitHub. Please note in the write-up the URLs/sources you used for the final deliverable. The amount of resources you use does not count negatively; we are interested in both the final product, as well as the process used to achieve said results.
4. If you are unsure on how to complete a task, or your implementation is not working, documenting the process you went through and what issues your ran into is also strongly encouraged as it highlights your thought process when posed with a challenge.
5. Do not post or share this Technical Challenge or information about it to the Internet. Each technical challenge has distinct differences and can be tied back to the individual who received it.

**Scenario 1**

A company has implemented the following Architecture to support one of their Applications in a GCP GovCloud environment.

Diagram

Description automatically generated

The assumptions outlined below are assumed to be true, regarding the Cloud Infrastructure and Architecture.

* 1 VPC – 10.1.0.0/16
* 5 subnets (spread across two availability zones for high availability)
  + Public Subnet 1 – 10.1.0.0/24 (should be accessible from internet)
  + Public Subnet 2 – 10.1.1.0/24 (should be accessible from internet)
  + WP (Web Application) Subnet 1 – 10.1.2.0/24 (should NOT be accessible from internet)
  + WP (Web Application) Subnet 2 – 10.1.3.0/24 (should NOT be accessible from internet)
  + DB Subnet 1 – 10.1.4.0/24 (should NOT be accessible from internet)
  + DB Subnet 2 – 10.1.5.0/24 (should NOT be accessible from internet)
* 1 compute instance running Windows Server 2019 in subnet *Public Subnet 1*
  + 50 GB storage
  + e2-standard-4
  + Hostname should “bastion1”
  + Public EIP associated (not reserved)
* 1 compute instance running RedHat in subnet *WP Subnet 1*
  + 20 GB storage
  + e2-small
  + Hostname should be “wpserver1”
* 1 compute instance running RedHat in subnet *WP Subnet 2*
  + 20 GB storage
  + e2-small
  + Hostname should be “wpserver2”
* 1 Cloud SQL PostgreSQL Databases running PostgreSQL 11 in subnet *DB Subnet 2*
  + db-f1-micro
  + DB Name should be “DB1”
* 1 HTTP/S Load Balancer that has listeners in subnets *Public Subnet 1 & 2*
  + listens on port 443/TCP
    - forwards traffic to the instance in subnet *WP Subnet 1*
* 1 GCP Cloud Monitoring SLO test is configured
  + "Web\_Application\_Alive" – This test attempts to access the web page from the external load balancer.
* GCP Cloud DNS is used to translate the company’s domain name to the HTTP/S Load Balancer
* All Resources within the subnets have appropriate least-permissive Security Groups in place
* The company does not currently use any Infrastructure as Code (IaC) to manage their infrastructure via code
* The company has a SLA & SLO for the Application
  + SLA – 99.5% Availability
  + SLO – 99.9% Availability

**Instructions**

**Question 1**

Assume now that you are an SRE who is responsible for maintaining the Availability & Uptime of the company’s environment, which was described previously in the Scenario. At 4:35pm on Friday, there were reports of customers being unable to access the website coming in; at 4:35pm the “Web\_Application\_Alive” test began failing. Complete the following questions with a full, in-depth, and detailed write-up/response for each:

1. Describe the steps that you would take to begin debugging the problem, to determine what might be going on.

**The first step is to check to see if there is an error(s) in the logs. In Cloud Logging, we can navigate there and check the error reporting section, since we now have inaccessibility issues since 4:35pm. We know that there is definitely error requests being sent in. From the Cloud Console, we can go to hamburger menu and go to Cloud Logging, and then go to “Error Reporting”. Each error is grouped by their stack traces, and each error entry provides a detailed summary of when an application started to produce the error and how often it occurred.**

**We need to apply a text filter to the “Error Reporting List” that will display and filter the logs of errors and the total numbers associated with this error. I would type in the applications name and either first seen or the error response code if prompted to when accessing the application. The SLO test configured was named ‘Web\_Application\_Alive’, and from there we can determine how often or how many hits have been registered. Since we are not able to determine whether it’s a client or server-side issue just yet. I would navigate and type in “Web\_Application\_Alive First seen”**. **The reasoning behind this is so that we can filter and sift through when the error or unresponsive VM, application, LB or backend DB started issuing an unresponsive state. From there we can go and tackle the issue with the right information and knowledge to debug and quickly bring the web app back up to speed.**

1. What metrics, monitoring, or other tooling/processes would you expect to have available within your monitoring system to identify and validate the infrastructure is performing in a functional, healthy state? **In SRE, one of the primary principles of this DevOps methodology is monitoring, hence we are going to delve into Cloud Monitoring. In Cloud Monitoring, there are tons of metrics to gauge and keep track of.**

**A few that can be used to monitor this specific scenario would include “Uptime for cloudsql\_database”. This can give a delta and keep track of how long the DB instance has been running.** **“State for cloudsql\_database” is another one for the DB, this can indicate the current instance is either running or in offline status. This metric is sampled every 60 seconds, and if it happens to find in neither a ‘failed’, ‘pending create’ or ‘suspended’ state. It will give us an ‘unknown state’. The state metric really is vital in all resources in cloud not just DB.**

**For network security and determining actual count of HTTP(S) backend service queries we can use the metric of ‘Request count’. VPC flow logsTo determine and find out about the VPC’s and the network usage between the private or public VPC’s or just general usage. We can use the metric of “quota/subnet\_ranges\_per\_vpc\_network”, whether this scenario is using a limit or quota is yet to be determined. But a good rule is ensuring costs do not skyrocket, so a quota is a recommended practice, and helps to monitor the current usage of the infrastructure allocated.**

**For determining work metrics of the corresponding VM’s and the applications, we can set up alerts and gather the data of these four metrics; throughput, success, errors and performance. Throughput; how much bytes or information can be processed in a given time. Success; the percentage of work that is effectively processed and if drops below a threshold. Errors, the error rate is very informative and can help give us a calculable rate for the SLO. Performance is the last work metric that we should measure, i.e., if a work takes too long to complete and if that performance rate violates the 99.5% SLA.**

**All these metrics help to ensure, and there definitely is a lot more to list but in this scenario these few aforementioned metrics can really help to give an instrument to monitoring and collecting these logs and events. One can even set up alerts to be proactive in remediating issues before they become too large. The specificity of Cloud Monitoring gives us the edge in being granular in scope to what we will be measuring. Also essential to retain the logs of these metrics for a longer duration for learning and auditing purposes.**

1. What changes or improvements could be made to the Architecture to reduce the risk of a similar incident? **One thing to note in this scenario that was mentioned in previous answers is cost. When dealing with cloud migration, the usual approach is to perform like to like migration and get on premises resources to the cloud as the first objective. Then teams perform reconfiguration and cost cutting principles to their architecture. But a well-designed architecture takes reconfiguration, automation and all dependencies into account alongside cost.**

**The implementation of a managed instance group as opposed to three individual compute engine instances would be the first improvement. As MIG’s offer the benefits of High Availability; will keep the VM instances running. A scenario where an instance crashes, the MIG would automatically spin up and recreate the VM to that specification.**

**Another improvement to the underlying architecture and infrastructure would be the implementation of VPC Flow logs, to capture traffic information between forth each VPC. It can really help paint a picture which specific traffic is not reaching an instance. Ensures cloud security and is overall a good practice, can view the logs and potentially export it into a GCS bucket. A security practice that would be beneficial for the Cloud SQL instance would be the use of firewall or firewall rules and network tags. A proper SQL instance should be able to connect from other instances, so configuring the firewall to allow certain traffic from client machines would be a sound security protocol.**

**There doesn’t seem to be any Disaster Recovery and back ups in place, so the use of local SSD’s to stage backups and the use of a Google Cloud Storage buckets would be crucial to this architecture for the cases of backups and can allow for maintenance windows for deployment schedules etc. An improvement to the VM’s would be the use of startup and shutdown scripts, these can allow automated tasks like installing software or running updates and making backups. Startup scripts on VM’s run whenever they reboot or restarts due to failure. All these improvements will definitely help in the architecture and ensures robust, scalable and high availability to reduce the chances of failure and mitigate risks.**

Now that the issue was resolved, and the environment is back into a healthy state; complete the following:

1. Create an After-Action Report (Post-Mortem/Post-Incident Review). It should include, at minimum, the following sections (more is highly encouraged):
   1. High-level Summary
   2. Participants (include who would be invited to an After Action Report Discussion, as well other stakeholders who were involved in the incident)
   3. Timeline
   4. Root Cause Analysis

**After Action Report**

**Prepared by: Saim Riaz**

[**saim.riaz@company.com**](mailto:saim.riaz@company.com)

**High Level/Executive Summary**

**Clients and customers encountered an accessibility issue on the Friday of the week at near end of work day at 4:35PM GMT (Placing random timezone for globalized entity). This issue was stemmed from an unresponsive state of a web facing application which coincided with an SLO test that also failed. This report will delve deeper into the solutions and remediation efforts made by part of the Site Reliability Engineering team to ensure that client infrastructure was running smoothly.**

**Participants**

**Included: SRE team-Saim Riaz, Technical Program Manager-Johnny Appleseed, DevOps Lead-Peter Parker, Customer Risk Analyst-Melony Tusk, Product Manager-Henry Ford, DB Team-James Pond, Security/Network Team-Robert Blue**

**Timeline**

1. **Customers encountered issue with accessibility towards the web application.**
2. **SRE teams checked and verified if alert went through of error ‘Web\_Application\_Alive’**
3. **SRE relayed issue to Product and TPM to notify of downtime of unresponsive application.**
4. **SRE used variety of Cloud Operational Tools like Monitoring, Trace and Logging.**
5. **SRE used Logging to filter and sift through Logs to find origin point of failure and where it occurred.**
6. **SRE communicated to DevOps team to push a quick fix to integrate a possible fix to the VM, LB and DNS from the provided information attained from the logs.**
7. **DevOps team was perplexed with no IaC methodology implementation in current architecture and will work at later date to improve with related configuration files.**
8. **SRE scheduled a downtime with analyst and team to implement the deployment and avoid violating SLO longer than necessary.**
9. **SRE team conducted canary deployment to ensure infrastructure worked as intended**
10. **Once teams had changes remediated and issue was resolved with corresponding teams such as database team and network teams validated change did not alter the existing infrastructure.**
11. **Document issue and work with relevant teams to incorporate improvements to existing architecture to avoid similar issues arising.**

**Root Cause Analysis**

**Issue was resolved successfully thanks to all participants to ensure client delivery hit and maintaining SLO and SLA. The ensuring of on-time delivery of DevSecOps methodology was crucial to tackle issue with the unresponsive state of the web facing application. The issue arised from a combination of the backend database not offering serviceable performance and could not scale effectively during normal network traffic timings. Another issue came from compute engine instances not being in a Managed Instance Group (MIG), offering multiple points of failure. This could have been prevented as MIG’s offer application-based auto healing. Manual restarting VM anytime an issue arises should not be the only solution to an accessibility issue.**

**Future Developments and Improvements**

**DevOps and SRE teams will implement IaC tools like Terraform to configure modules to build and provision cloud resources to GCP backed architectures. With the use of Terraform, teams can plug in code and resource dependencies to build infrastructure in an agile based methodology. Implementation of that scope will offer a lot to the client and company in hand and will ensure Continuous Integration and Continuous Delivery.**

**Question 2**

Following the events that occurred, you’ve now been tasked additional SRE Tasks.

Document your solution – including any sources you used. You may be asked to walkthrough your solution as if presenting to a client.

Your final deliverables for this section will include: the codified solution, and your documentation. The structure, formatting, and language/tool used for the solution, as well as the structure and formatting of the documentation is up to you.

1. For example, after an audit, findings were published that identified several Compliance Controls were not implemented appropriately, nor were multiple Vulnerabilities remediated.

Create an appropriate solution that codifies and solves the issues and push the code to a public GitHub repository. Any detail that is not provided in the scenario is up to your discretion to include, or exclude. Use of Infrastructure as Code (Terraform), Scripts (PowerShell, BASH, Python, etc.), Playbooks (Ansible, Chef, Puppet, etc.), or Group Policy Objects (Windows), are preferred solutions.

**Compliance Remediation (CIS Benchmark, Level 1 for: Windows Server 2019)**

|  |  |  |
| --- | --- | --- |
| **Host**: bastion1 (Windows Server 2019) | | |
| **ID** | **Name** | **Description** |
| 1 | 2.2.21 Ensure 'Deny access to this computer from the network' to include 'Guests, Local account and member of Administrators group' (MS only) - Guests, Local account and member of Administrators group | See CIS Benchmark Microsoft Windows Server 2019 L1 for more details:  https://www.cisecurity.org/cis-benchmarks/#microsoft\_windows\_server |
| 2 | 18.9.45.4.1.2 Ensure 'Configure Attack Surface Reduction rules: Set the state for each ASR rule' is configured – ‘26190899-1602-49e8-8b27-eb1d0a1ce869’ | See CIS Benchmark Microsoft Windows Server 2019 L1 for more details:  https://www.cisecurity.org/cis-benchmarks/#microsoft\_windows\_server |

**I espe**

**Question 3**

The Client is interested in codifying their existing GCP infrastructure. As an SRE, you are well-positioned to assist the Client by developing the Terraform necessary to deploy the architecture described in the Scenario.

Document your solution – including any sources you used. You may be asked to walkthrough your solution as if presenting to a client.

Your final deliverables for this question will include: the codified solution, and your documentation. The structure, formatting, and language/tool used for the solution, as well as the structure and formatting of the documentation is up to you.

1. Create terraform code that creates the networking and compute constructs defined and push the code to a public GitHub repository. Any detail that isn’t provided in the scenario is up to your discretion. Use of Terraform modules is highly encouraged.

**GitHub repository link:** [saimriaz/Coalfire-Challenge: Challenge from coalfire (github.com)](https://github.com/saimriaz/Coalfire-Challenge)

**Has 9 files, of which two are templates I have created from scratch in YAML. I have added comments to give context to specific configuration files.**

1. Login to the web server instance in “WP Subnet 2” from the bastion host in “Public Subnet 1” and take a screenshot of the terminal logged in. Include this screenshot in your documentation.

**I did not understand this question to the fullest extent.**

Upon completing the challenge, please email your documentation and the link to your public GitHub repository to <Technical Recruiter’s Email>. If you have further clarification questions, or issues, please notify us immediately.