

SRE Challenge

Nordeus – Job Fair 2026

Intro

SRE (Site Reliability Engineering) is a team keeping our game infrastructure running smoothly, ensuring that millions of players enjoy a seamless, uninterrupted experience, even during peak times. SREs focus on application running platforms, operating systems, storage subsystems, and networking. We implement best practices to enhance availability, reliability, and scalability, while improving developers' workflow efficiency by streamlining processes, allowing developers to focus more on coding and innovation.

Your task is to **step into the role of an SRE**: take a new application, deploy it to a live production environment, and ensure it is engineered to scale automatically as demand grows.

Setup

Cloud Provider Account

To participate in this challenge, you will need an account with a cloud provider (GCP, AWS, Azure, etc.). You are free to use any platform (this will not impact your score), but we recommend Google Cloud Platform (GCP), as it is a core part of the Nordeus infrastructure stack. New GCP users are typically eligible for \$300 in free credits, which is more than enough to complete this task.

- **Sign Up:** Go to the [GCP Console](#) and sign up for the free trial.
- **Identification:** You will be asked to provide credit card details. According to their terms, this is used strictly for identity verification, and they do not automatically charge you once the trial ends or credits are exhausted unless you manually upgrade to a paid account.

- **Account Profile:** When prompted for tax information, select "Business" as the account type (a "Personal" option may not be available). For the Tax ID/Number, you can simply enter 000000000.
- You can monitor your remaining credits at any time by navigating to the Billing page in the GCP console.
- Make sure you **delete all resources** you created after you are done with the challenge in order not to waste credits!

Kubernetes Cluster Provisioning

Before deploying the application, you need to provision the infrastructure and configure your local environment to manage it. Create a Kubernetes Cluster (Google Kubernetes Engine if using Google Cloud). Once the cluster is live, configure your local terminal to communicate with it. You should be able to manage the remote cloud resources using standard tools like *kubectl*.

You must provide a **text file with the brief summary of the Kubernetes cluster** you have provisioned (cloud provider you chose, cluster type, node pool configuration (if not Autopilot): the number of nodes, the machine type used, whether cluster autoscaler is on and its configuration etc.) named *infrastructure_setup.txt*.

Challenge

This challenge has three parts, and each one is the prerequisite for the next. We built this challenge to simulate the real-world problems our SRE team solves every day. Our goal is for you to learn something new and have fun building on the cloud. Don't worry if your solution is incomplete, feel free to send us what you have! We hope you'll have fun! 🍷

Part 1: Deploy the Game Service to Kubernetes

In this first stage, your goal is to move the game service from a local codebase into a live, containerized environment.

Requirements

You are provided with the [source code](#) for a simplified Python game service. You must containerize this application and deploy it to your Kubernetes cluster. **You are**

not allowed to make any changes to the source code (your solution will be tested and graded using the original source code)!

- **Containerization:** Write a Dockerfile to package the Python application, build the image and push it to a Container Registry associated with your chosen cloud provider (e.g., Google Artifact Registry).
- **Deployment:** Create the necessary Kubernetes manifests (YAML) to deploy the application. The initial resources you should set for it are: requests - cpu 100m, memory 128Mi, and limits - cpu 200m, memory 256Mi.
- **External Access:** Ensure the service is accessible from the internet. You should use a Load Balancer (for the Service manifest, specify type: LoadBalancer) to expose the application to the world (no need for HTTPS here - you do not need to configure SSL/TLS or HTTPS for this challenge).


Solution

To complete this part, you must provide:

- The Dockerfile used to build the image.
- The Kubernetes manifests used for the deployment.

Verification

Once deployed, verify that the application is running correctly by targeting its exposed endpoints (when you target your load balancer external IP and the port you set from your browser, you should see the message "Welcome to SRE Job Fair Challenge 2026!"), reviewing the pod logs for any startup errors, looking at the pod status etc.

 **Bonus points:** Write in a text file what commands you used to test that everything was up and running properly. You can also provide screenshots.

Part 2: Deploy the "Spammer" App to Kubernetes

Now that your game service is running and accessible, the next step is to simulate real-world traffic. We have provided a spammer app: a load-generation tool designed to send asynchronous requests to your service and report on latency and throughput. The tool runs in two distinct phases: first the warm up, which is supposed

to simulate regular traffic, and then the load test, which simulates a traffic burst. It is important to know that only the second phase counts toward your final report.

Requirements

You need to deploy the spammer app to your Kubernetes cluster. Its mission is to target the game service you deployed in Part 1 and generate a report on its performance. You are provided with a pre-built Docker image for this tool: `europa-docker.pkg.dev/nord-inf-prod/public-docker-repository/jobfair/spammer:latest`.

The spammer app is configured via environment variables:

- **`SPAMMER_BASE_URL`**: The full URL of your game service. Default: "http://localhost:80".
- **`SPAMMER_RPS`**: The number of requests per second the tool should send during the load test phase. Default: 1000. Minimum: 1. Maximum: 1000.
- **`SPAMMER_DURATION_SECONDS`**: The duration of the load test phase. Default: 60.
- **`SPAMMER_WARMUP_SECONDS`**: The duration in seconds of the warmup phase before the main load test begins. After warmup completes, the load test starts (configured with `SPAMMER_RPS` and `SPAMMER_DURATION_SECONDS`) and the report is generated (the report doesn't include warm up phase requests). Default: 60.
- **`SPAMMER_WARMUP_RPS`**: The number of requests per second the tool should send during the warmup phase. Default: 100. Minimum: 1. Maximum: 500.

Important note: If you want the spammer to send more than the maximum `SPAMMER_RPS`, deploy multiple instances of it.

Solution


To complete this part, you must provide:

- The Kubernetes manifests used for the deployment.

Verification

Run the spammer first with the default config. Once it finishes, check its logs to see the generated report. A successful run will output a "SPAMMER REPORT" table. Try out

different configuration values to see how the game service handles the load. Pay attention to the resources you give to the spammer app - you might need to increase its resources based on how many requests you configure it to send. Check the game service resources (CPU, RAM etc.) and pod status in situations where it is not able to handle the load or the latency is very high.

 **Bonus points:** Provide screenshots of the "SPAMMER REPORT" table for different spammer configurations.

Part 3: Scale the Game Service

In Part 2, you likely observed that your game service struggled to maintain low latency or a high success rate under the pressure of the spammer app. In a real-world scenario, your infrastructure must be elastic enough to handle these traffic spikes without manual intervention.

Requirements

You must modify your Kubernetes environment to allow the application to scale. How you choose to implement this scaling and at what threshold is up to you. The system must respond to increased load and scale back down when traffic subsides without any manual commands. An SRE's goal is to balance performance with cost. Your scaling strategy and thresholds should be optimized: scaling to infinity is easy, but scaling **efficiently** (not taking much more than you need) is the real challenge.

Solution

To complete this part, you must provide:

- The Kubernetes manifests that implement your scaling strategy.
- A text file explaining your choice of scaling method, the thresholds you selected, and any modifications you made to the cluster or deployment to support this. Make sure you also note what the maximum load you tested with that the service could handle was.

Verification

Try out different configuration values for the spammer app to see how the game service handles the load. A successful solution will show the service scaling up to

meet demand (that you decided on) and scaling back down once the spammer finishes, all while keeping the error rate near zero (not necessarily exactly zero).

💎 **Bonus points:** Provide screenshots of the "SPAMMER REPORT" table for different configurations along with terminal output showing the application scaling in real-time. You can also record the scaling and provide the video.

! **Note:** This challenge is purposefully quite open-ended because we want to give you the freedom to experiment. However, if you have any questions that you feel are blocking your progress, feel free to reach out to jobfair@nordeus.com.

Submission

To complete the challenge, please package your work into a single .zip archive and follow these steps:

1. What to include:
 - a. *infrastructure_setup.txt*
 - b. Dockerfile
 - c. Final Kubernetes Manifests: Provide only the final versions of your YAML files. There is no need to separate them by each part of the challenge.
 - d. Supporting Documentation: A single file (PDF or text) combining your explanations (and screenshots). You may also include a video if you wish.
2. How to send your submission:
 - a. Email: jobfair@nordeus.com
 - b. Subject: **SRE Challenge**
 - c. Body: Please clearly state your **full name** in the email. Add the link to your **LinkedIn profile**, if you have it.

Useful materials

- [Kubernetes Docs](#)
- [Excellent Kubernetes Youtube Videos](#)

The challenge is open until May 3rd, 2026, end of the day. Good luck!