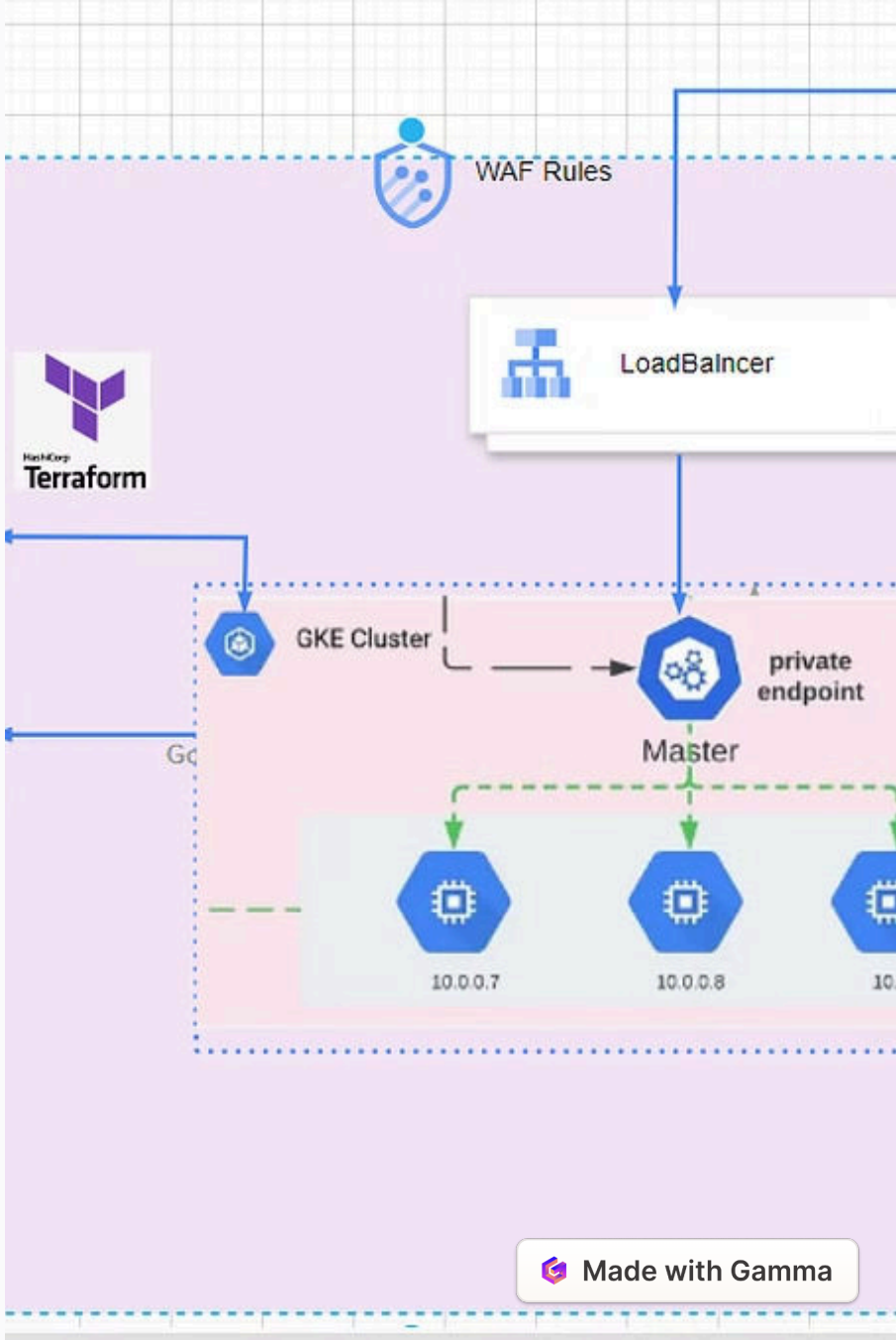
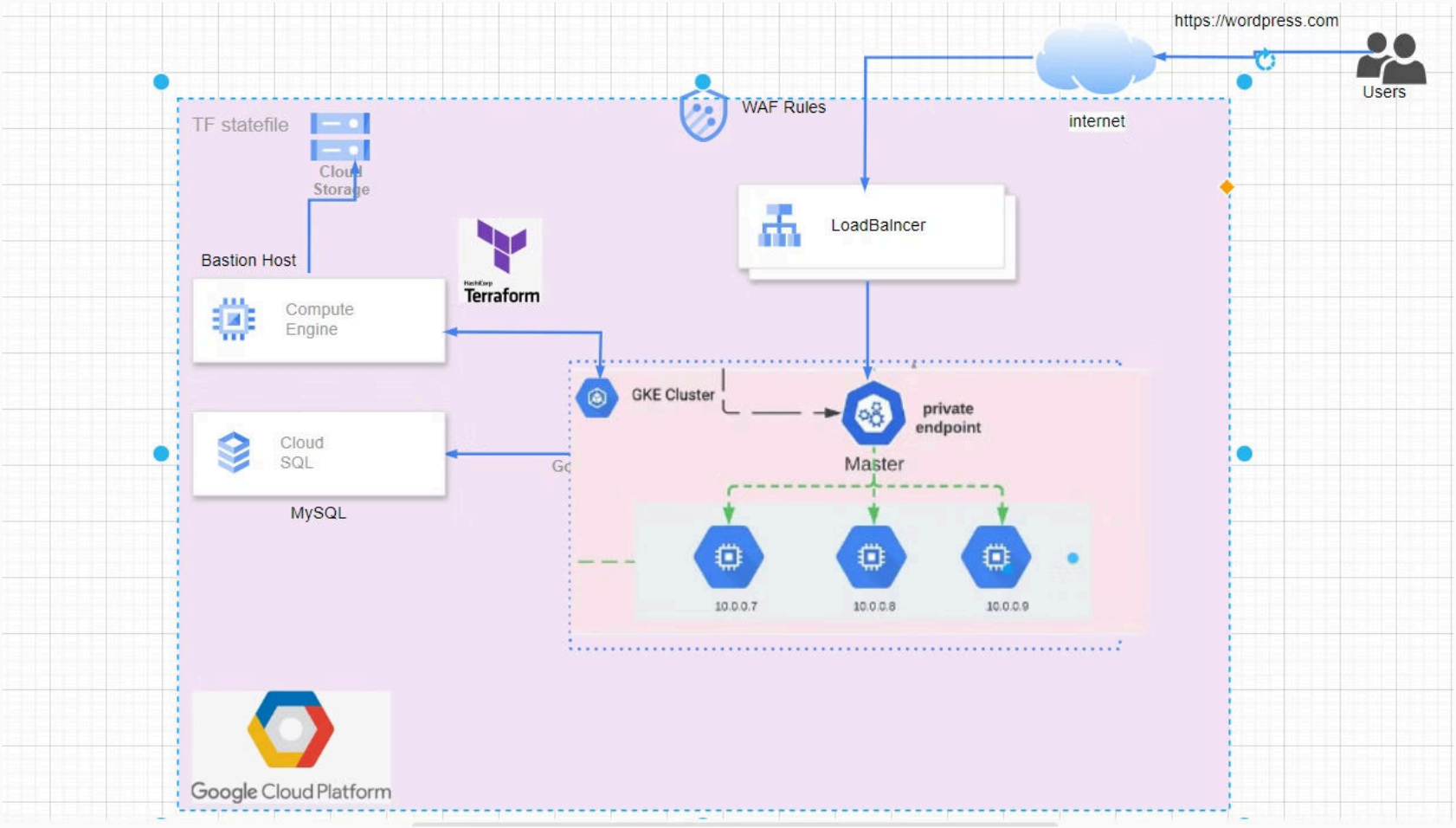
Introduction to the Google Cloud Infrastructure

In today's digital landscape, cloud computing has become a transformative force, empowering businesses and organizations to unlock new levels of efficiency, scalability, and innovation. This comprehensive document outlines a robust cloud infrastructure solution that will serve as the backbone for a mission-critical Wordpress application. By leveraging the power of a dedicated Virtual Private Cloud (VPC), a private Google Kubernetes Engine (GKE) cluster, and a secure CloudSQL database, we will create a secure, highly available, and performant environment to host your web application.

by Srikanth Kagitala



Architecture Diagram



Dedicated VPC with a subnet in one region

Increased Security

Dedicated VPCs offer enhanced security compared to the default VPC. By creating a custom VPC network, you can define your own IP address ranges, create subnets, and configure routing tables and gateways. This allows you to have better control over your network traffic and prevent unauthorized access to your resources.

Improved Network Isolation

A dedicated VPC provides a higher level of network isolation, as your resources are separated from other customers' resources. This is particularly important for sensitive workloads or applications that require strict security and compliance requirements. With a dedicated VPC, you can ensure that your resources are not commingled with those of other tenants.

Customizable Network Configuration

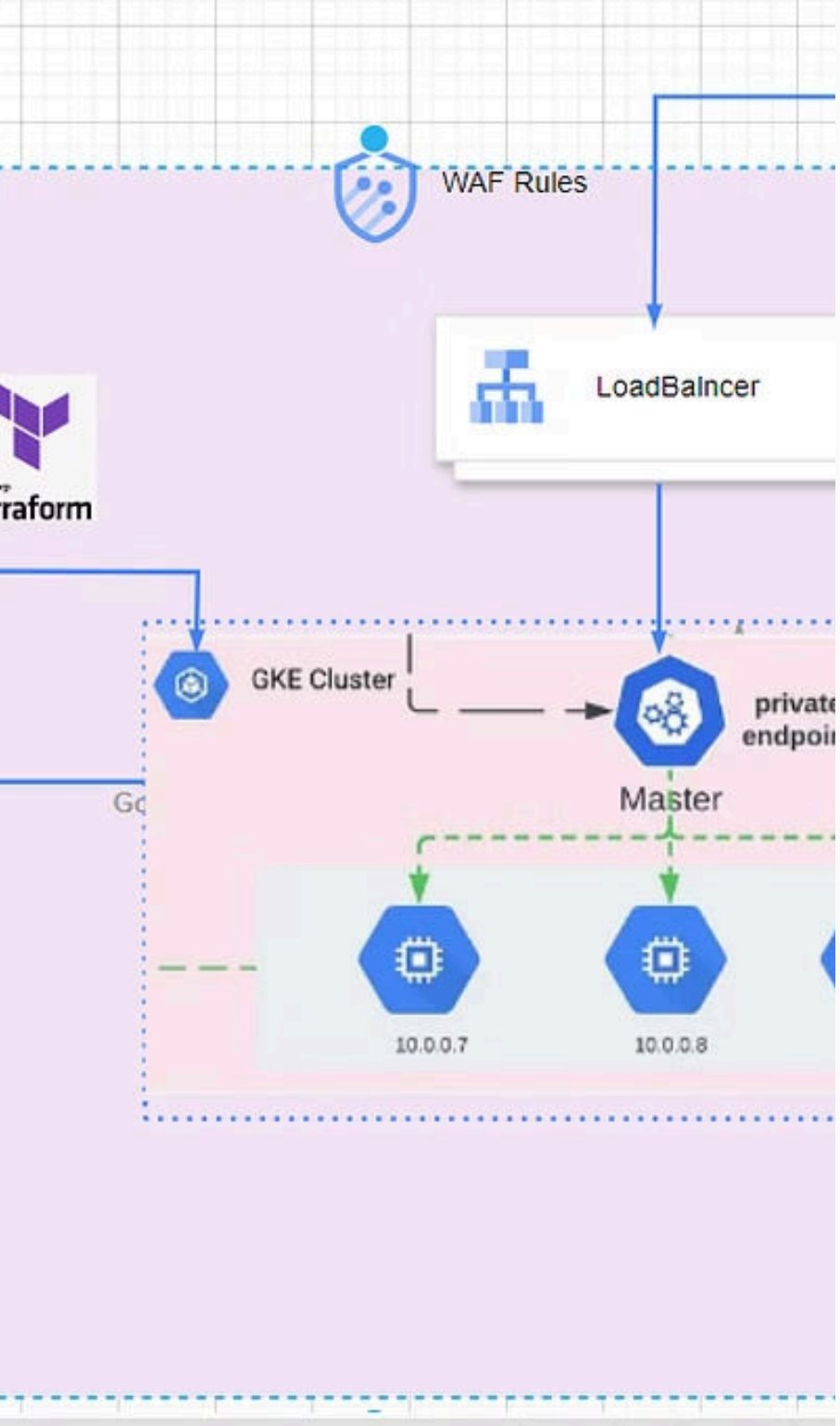
Creating a dedicated VPC gives you the flexibility to configure your network according to your specific needs. You can create subnets in different regions, set up private and public IP address ranges, and define routing tables and network gateways. This allows you to architect your network in a way that best suits your application's requirements, such as implementing a multi-tier architecture or connecting to on-premises resources.

Private GKE Cluster Across Multiple Zones

To meet the client's requirement for a private GKE cluster running in one region across multiple zones, we will provision a Kubernetes Engine cluster with the following key features:

1. Private Cluster: The GKE cluster will be configured as a private cluster, meaning the API server and nodes will not have public IP addresses. This enhances the security posture by isolating the cluster from the public internet.
2. Regional Deployment: The cluster will be deployed in a single Google Cloud region, spanning multiple availability zones within that region. This provides high availability and fault tolerance, ensuring the cluster can withstand the failure of an individual zone.
3. Controlled Access: Access to the private cluster will be restricted to authorized resources within the dedicated VPC network. This can be achieved through the use of private endpoints, VPC service controls, and IAM policies to tightly manage who and what can interact with the cluster.

By deploying a private GKE cluster across multiple zones in a single region, we can ensure the Wordpress application has a highly available, scalable, and secure Kubernetes environment to run in. The private nature of the cluster will help protect the application and its associated CloudSQL database from external threats, while the regional deployment with multi-zone redundancy will provide the necessary reliability and resilience.



Private CloudSQL Database Running MySQL

To support the Wordpress application, a private CloudSQL database running MySQL will be provisioned. CloudSQL is a fully-managed database service provided by Google Cloud, offering high availability, scalability, and ease of management. By running the database in a private subnet, we can ensure secure communication between the Wordpress application and the database, without exposing the database to the public internet.

The CloudSQL instance will be configured with appropriate compute resources, storage, and high availability settings to meet the expected workload and performance requirements of the Wordpress application. Automatic backups, point-in-time recovery, and other database management features will be enabled to ensure the integrity and reliability of the data.

To integrate the Wordpress application with the CloudSQL database, secure connection strings and authentication credentials will be provided, allowing the application to seamlessly interact with the database. This integration will be a key component in the overall cloud-based architecture, ensuring the Wordpress application can leverage the scalable and highly available database services offered by Google Cloud.

Terraform state saved to a GCS bucket

To ensure the reliability and durability of the Terraform state, the client has decided to save it to a Google Cloud Storage (GCS) bucket. GCS is a highly scalable and secure object storage service provided by Google Cloud Platform. By storing the Terraform state in a GCS bucket, the team can benefit from features like versioning, access control, and geographic redundancy, which are crucial for maintaining the integrity and recoverability of the infrastructure configuration.

The GCS bucket will be created as part of the Terraform configuration, ensuring that the state file is always stored in the designated location. This approach not only simplifies the setup process but also makes it easier to manage and collaborate on the infrastructure as code. The Terraform state file, which contains the current state of the resources, will be regularly backed up to the GCS bucket, providing a reliable source of truth for the entire infrastructure.

Building the Wordpress application

To build the Wordpress application that will be deployed to the GKE cluster, we first need to create a custom Docker image that includes the Wordpress codebase and any necessary dependencies. This will allow us to package the application in a containerized format that can be easily managed and deployed within the Kubernetes environment.

We'll start by creating a Dockerfile that describes the steps to build the custom Wordpress image. This will include installing the necessary Wordpress packages, configuring the web server and PHP runtime, and copying the Wordpress codebase into the image. We'll also need to define any environment variables or configuration settings required for the Wordpress application to run properly, such as database connection details.

Once the Dockerfile is complete, we'll use a tool like Google Cloud Build to automatically build and push the Docker image to a container registry, such as Google Container Registry (GCR). This will make the image available for deployment to the GKE cluster. We'll also set up automated build triggers to ensure that any changes to the Wordpress codebase result in a new container image being generated and pushed to the registry.

In parallel, we'll create the necessary Kubernetes manifests to deploy the Wordpress application to the GKE cluster. This will include defining the Deployment, Service, and Ingress resources that describe how the application should be run and exposed to the internet. We'll need to reference the container image we built earlier and ensure that the application is properly configured to connect to the private CloudSQL database instance.

By following this process, we can ensure that the Wordpress application is built, packaged, and deployed in a consistent and scalable manner, taking advantage of the benefits of containerization and Kubernetes orchestration.

Deploying the Wordpress application to the GKE cluster

1

Build the Docker Image

The first step in deploying the Wordpress application to the GKE cluster is to build the Docker image. This involves creating a Dockerfile that defines the application's runtime environment, including the Wordpress software, any necessary dependencies, and any custom configuration. The Docker image is then built using the Dockerfile and pushed to a container registry, such as Google Container Registry (GCR), so that it can be deployed to the GKE cluster.

2

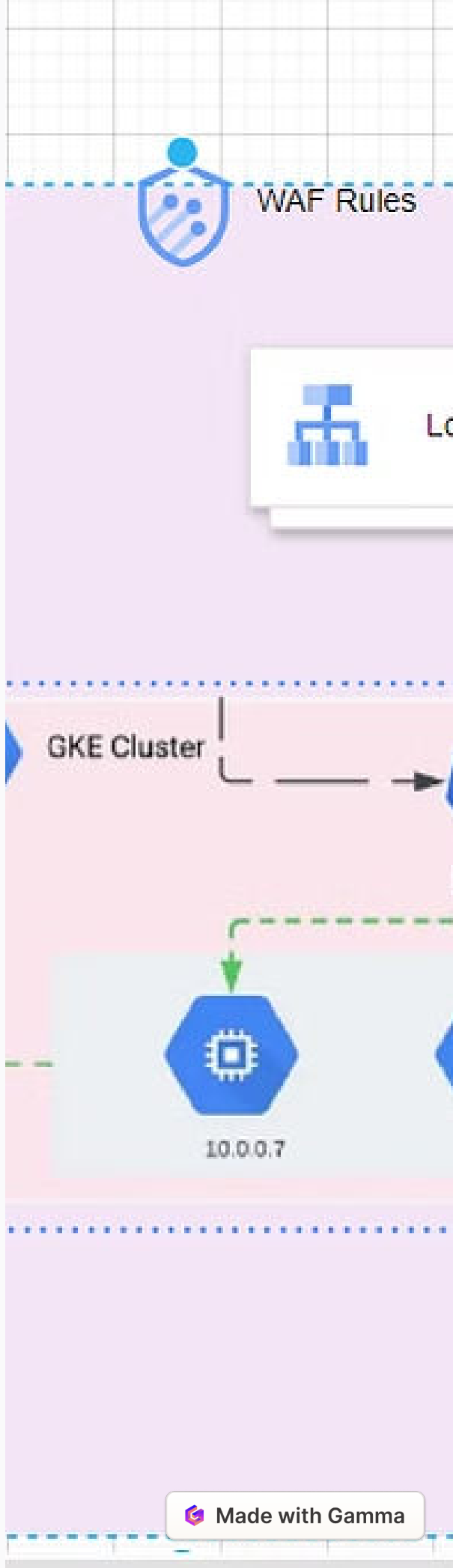
Configure Kubernetes Manifests

With the Docker image ready, the next step is to create the necessary Kubernetes manifests to deploy the application. This includes defining a Deployment resource to manage the Wordpress pods, a Service resource to expose the application internally within the cluster, and potentially an Ingress resource to provide external access to the application. The manifests should also include any necessary environment variables, secrets, or other configurations required by the Wordpress application.

3

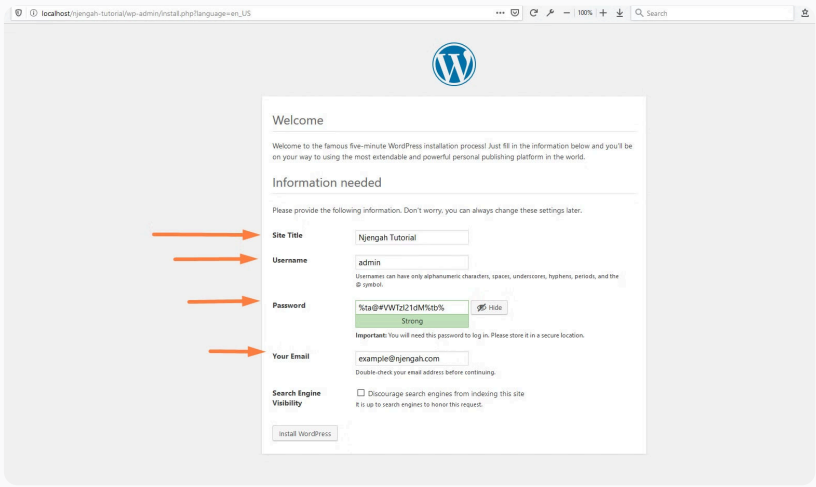
Deploy to the GKE Cluster

Once the Kubernetes manifests are ready, the application can be deployed to the GKE cluster. This typically involves using the `kubectl` command-line tool to apply the manifests to the cluster. Kubernetes will then manage the deployment process, including scheduling the Wordpress pods, and ensuring the application is running and available within the cluster. Any necessary scaling, self-healing, or load balancing will also be handled by the Kubernetes platform.



Demonstrating the CloudSQL Connection

To demonstrate the connection between the Wordpress application and the private CloudSQL database, we will perform a series of actions that showcase the seamless integration between the two components. First, we will create a new post in the Wordpress admin panel, adding text and uploading an image. We will then navigate to the CloudSQL console and verify that the new post data has been successfully stored in the database.



Next, we will make a modification to an existing post in the Wordpress dashboard and observe the corresponding changes reflected in the CloudSQL database in real-time. This will highlight the bidirectional nature of the connection, where updates made in Wordpress are immediately propagated to the underlying database.

Finally, we will simulate a scenario where the CloudSQL instance is temporarily unavailable, such as during a maintenance window or a failover event. We will then demonstrate how the Wordpress application gracefully handles the connection loss, displaying a user-friendly error message rather than crashing or becoming unresponsive. This will showcase the resilience and fault-tolerance of the overall architecture.

Exposing the Application using an Ingress or Gateway with a Static IP

To make the Wordpress application accessible to users outside the GKE cluster, you'll need to expose it using an Ingress or Gateway resource. An Ingress provides layer 7 load balancing and routing, allowing you to map HTTP/HTTPS traffic to your application services. Alternatively, a Gateway resource provides a more flexible and extensible way to manage ingress traffic, supporting a wider range of protocols and traffic management capabilities.

For this project, you'll configure an Ingress resource to expose the Wordpress application with a static IP address. This will ensure that the application has a consistent, public-facing endpoint that users can access. The static IP address will be provisioned from the dedicated VPC network you created, keeping all traffic within your private infrastructure.

By using an Ingress with a static IP, you can also set up SSL/TLS termination, routing rules, and other advanced features to secure and manage the application's public-facing interface. This provides a robust, scalable, and easily-configurable way to make your Wordpress application available to users, while keeping it isolated within your private GCP environment.