Here's a high-level overview of how you can use Terraform to set up a 3-tier environment on GCP:

1. **Install Terraform**: First, you'll need to install Terraform on your local machine. You can download it from the official website: https://www.terraform.io/downloads.html
2. **Set up GCP Credentials**: Terraform needs credentials to authenticate with GCP and provision resources on your behalf. You can set up a service account and generate a JSON key file with the necessary permissions. Then, you'll set the **GOOGLE\_APPLICATION\_CREDENTIALS** environment variable to point to the JSON key file.
3. **Write Terraform Configuration**: Define your infrastructure using Terraform's configuration language (HCL). This involves creating **.tf** files that describe the resources you want to provision.
4. **Organize Your Terraform Code**: Organize your Terraform code into modules to promote reusability and maintainability. You can have separate modules for each tier of your application (e.g., presentation, application, data), as well as modules for common resources like networking.
5. **Define Networking Infrastructure**: Create a Virtual Private Cloud (VPC) network, subnets, firewall rules, and any other networking resources needed for your environment.
6. **Provision Compute Instances**: Define compute instances for each tier of your application (e.g., front-end, application servers, database servers). Specify the machine type, boot disk, network settings, etc.
7. **Configure Load Balancers**: Set up load balancers to distribute incoming traffic across your application tiers. You may need different types of load balancers depending on your specific requirements (e.g., HTTP(S) load balancer, TCP/UDP load balancer).
8. **Configure DNS**: If your application requires a custom domain name, you can configure DNS records to point to your load balancers.
9. **Deploy Application Code**: Depending on your application architecture, you may need to deploy code to your compute instances. This could involve using configuration management tools like Ansible, Puppet, or Chef, or integrating with CI/CD pipelines.
10. **Test and Validate**: Before deploying your infrastructure to production, it's important to test and validate it to ensure everything is working as expected. You can use Terraform's plan and apply commands to preview and apply changes, as well as validate the configuration using linting tools like **terraform fmt** and **terraform validate**.
11. **Version Control**: Store your Terraform configuration files in version control (e.g., Git) to track changes and collaborate with your team.
12. **Infrastructure as Code (IaC) Practices**: Follow best practices for IaC, such as using variables and parameterizing your configuration, documenting your infrastructure, and implementing infrastructure testing and security scanning.

By following this approach, you can provision a 3-tier environment on GCP using Terraform in a reproducible and scalable manner. This allows you to easily manage your infrastructure, make changes safely, and track the state of your environment over time.