**Report: Setting up Development Environment, CI/CD Pipeline, Infrastructure Provisioning, and Monitoring for a PHP Application**

**1. Setting up and Running the Development Environment with Docker Compose:**

To set up and run the development environment for our PHP application and MySQL database using Docker Compose, we followed these steps:

* Created Dockerfiles for both the PHP application and MySQL database, specifying the necessary configurations and dependencies.
* Wrote a **docker-compose.yml** file defining both services, PHP application, and MySQL database, along with their configurations and network settings.
* Built the Docker images using **docker-compose build**.
* Started the containers using **docker-compose up**.

This setup allowed developers to quickly spin up a local environment for development and testing without worrying about individual dependencies or compatibility issues.

**2. Details of the CI/CD Pipeline:**

Our CI/CD pipeline is implemented using GitLab CI and includes the following stages:

* **Linting and Testing:** This stage runs PHP linting to check for syntax errors and PHPUnit tests to ensure code functionality.
* **Building Docker Images:** Docker images for the PHP application and MySQL database are built using Docker Compose.
* **Pushing Docker Images:** The built Docker images are pushed to a container registry for easy access and deployment.
* **Deploying to Staging Environment:** Finally, the application is deployed to a staging environment using Docker Compose, allowing for testing in an environment closely resembling production.

This pipeline ensures that changes pushed to the main branch are automatically tested, built, and deployed, reducing manual intervention and minimizing the risk of deployment errors.

**3. Provisioning the Staging Environment using Infrastructure as Code Scripts:**

For provisioning the staging environment, we opted to use Terraform for its ease of use and flexibility. Our Terraform script (**main.tf**) provisions the necessary infrastructure, including web servers and database servers, in a cloud environment such as AWS.

The script defines resources like AWS EC2 instances and RDS database instances, specifying configurations such as instance types, security groups, and networking settings. This approach ensures consistency and repeatability in our infrastructure provisioning process, making it easy to replicate the environment across different stages or environments.

**4. Accessing and Using the Monitoring Dashboard:**

For monitoring, we chose Prometheus as our monitoring tool. After installing Prometheus and configuring it to scrape metrics from our PHP application and MySQL database, we accessed the Prometheus UI through a web browser. Here, we could visualize metrics like response time, error rate, and resource usage, helping us monitor the health and performance of our application in real-time.

**5. Explanation of Design Choices and Challenges:**

* **Design Choices:** We chose Docker Compose for managing our development environment due to its simplicity and ease of use. GitLab CI was selected for our CI/CD pipeline because of its tight integration with GitLab repositories and its extensive feature set. Terraform was chosen for infrastructure provisioning because of its declarative syntax and support for multiple cloud providers. Prometheus was selected for monitoring due to its powerful querying language and ecosystem of integrations.
* **Challenges:** One challenge we encountered was ensuring compatibility between different components of our system, such as ensuring that the Docker images built in the CI/CD pipeline were compatible with the infrastructure provisioned by Terraform. Additionally, setting up monitoring with Prometheus required configuring scrape targets and understanding the metrics exposed by our application, which required some trial and error.

In conclusion, by following these steps and leveraging the mentioned tools and technologies, we were able to set up a robust development environment, implement a reliable CI/CD pipeline, provision infrastructure using Infrastructure as Code, and monitor our application effectively, ensuring its stability and performance throughout its lifecycle.