**Case Study 1:** **Accelerating Software Delivery with Continuous Integration**

A well-established e-commerce company facing a common challenge in the digital era: the need to innovate and deliver new features and updates to its online store rapidly. In a highly competitive market, speed to market and agility were critical factors that could set them apart from their competitors. However, slow manual testing and deployment procedures were impacting their existing software development process, leading to delays in delivering new features and bug fixes to customers.

You need to create an end-to-end CI/CD pipeline for a microservices application. The pipeline should automate the build, test, and deployment using GitHub Actions.

**Tasks:**

* Create a GitHub Actions workflow that builds, tests, and deploys the application.
* Use Terraform to provision the necessary AWS infrastructure (EKS) for deploying the containers.
* Implement a deployment strategy.
* Deploy agents on Kubernetes to collect logs from the containers and send them to Elasticsearch.
* Set up Kibana to create dashboards for visualizing logs and metrics.
* Create alerts in Kibana for specific error patterns or performance metrics.

**Case Study 2:** **Secure DevOps: Integrating Security into the Pipeline**

You need to deploy a secure and resilient application on AWS using Kubernetes. The solution must include secure secrets management, continuous deployment, and comprehensive monitoring.

**Tasks:**

* Create an AWS EKS cluster with Terraform.
* Ensure images are scanned for vulnerabilities.
* Deploy the application to EKS using Kubernetes manifests and use Liveness Probes to periodically check the health of a container.
* Building & scanning images, deploying application should be automated using AWS CodePipeline & AWS CodeBuild
* Set up approval processes for production deployments.
* Create Kibana dashboards to monitor security events and application performance.

**Case Study 3: Automate Deployment with AWS CodePipeline**

You need to implement a deployment strategy for a web application running on AWS EC2 instances. The deployment process should be fully automated using AWS CodePipeline, CodeBuild, and CodeDeploy.

**Tasks:**

* Use Terraform to provision EC2 instances, Auto Scaling Groups, and a Load Balancer.
* Build and push the Docker image to AWS ECR.
* Set up a CodePipeline with stages for source, build, test, and deploy.
* Use CodeCommit as source control.
* Use CodeBuild for the build phase. Configure CodeBuild to build the Docker image and run unit tests.
* Use CodeDeploy to perform blue/green deployments to EC2 instances.
* Ensure that the Load Balancer routes traffic to the correct environment during deployment.
* Set up approval processes for production deployments.
* Deploy agents on EC2 instances to collect and ship logs to Elasticsearch.
* Set up Kibana dashboards to visualize and analyze logs and application performance.

**Case Study 4: Multi-Environment Deployment**

You need to manage deployments across multiple environments/namespaces (development, staging, production). The solution should automate deployments and provide robust monitoring and logging.

**Tasks:**

* Use Terraform to provision AWS Resources ( EKS cluster ) & for environments use namespaces (development, staging, and production).
* Use multi-stage builds to optimize Docker images.
* Deploy the application to EKS clusters in each environment.
* Use Helm charts to manage environment-specific configurations and deployments.
* Collect logs from multiple environments/namespaces.
* Set up Kibana to provide visibility across environments and identify issues.