PROJECT EXPLANATION

MICROSERVICES PROJECT EXPLANATION:

As an experienced DevOps engineer with three years of experience, I have worked extensively on micro services projects. One notable project involved containerizing applications using Docker and Kubernetes, automating tasks with Jenkins pipelines, and employing Argo CD as a GitOps tool.

In this project, my responsibilities included setting up the CI/CD pipeline, automating the build and deployment processes, ensuring code quality and security through various scans, and managing Docker images and containers. The integration of Jenkins, Docker, SonarQube, and Trivy within the pipeline ensured a streamlined, secure, and efficient development workflow.

I also implemented monitoring solutions using Prometheus and Grafana, providing comprehensive visibility into cluster performance and application health.

My work on this project not only improved deployment speed and reliability but also significantly enhanced the overall code quality and security posture and operational monitoring of the application.

HOW DOES THESE TOOLS WORKS:

Jenkins Setup: Jenkins was installed and configured to orchestrate the CI/CD pipeline.

Git Installation: Git was installed to manage source code versioning.

Trivy Installation: Trivy, a vulnerability scanner for containers, was installed to enhance security checks.

SonarQube Container: A SonarQube container was created using Docker to facilitate code quality analysis.

CAN YOU PLEASE EXPLAIN YOUR PIPELINE STAGES:

Clean Workspace: Ensuring a fresh workspace for each build.

Git Checkout: Cloning the project repository.

SonarQube Analysis: Performing static code analysis with SonarQube.

Quality Gate: Verifying code quality against predefined metrics.

Install Dependencies: Installing project dependencies using npm.

OWASP Dependency Check: Installed and configured the OWASP Dependency Check plugin to scan for known vulnerabilities in project dependencies.

Trivy File System Scan: Used Trivy to scan the file system for vulnerabilities.

Docker Image Build and Push: Built and pushed Docker images to Docker Hub using Jenkins and stored credentials securely.

Trivy Image Scan: Performed vulnerability scanning on the Docker image using Trivy.

Deployment to Docker Container: Deployed the Docker container to a required environment.

DAY TO DAY ACTIVITIES ON THIS PROJECT:

- Launching and configuring AWS instances (t2.large) for Jenkins and other necessary services.
- Managing Kubernetes clusters using KOPS, including scaling and updating the cluster as needed.
- Setting up and maintaining Jenkins pipelines to automate the build, test, and deployment processes.
- Ensuring the pipeline stages, such as code checkout, static analysis with SonarQube, dependency checks, and Docker image builds, run smoothly and efficiently.
- Troubleshooting and resolving any issues that arise during pipeline execution.
- Building and pushing Docker images to Docker Hub.
- Deploying and managing Docker containers on both local and Kubernetes environments.
- Performing security scans on Docker images using Trivy and OWASP Dependency Check.
- Creating and updating Kubernetes manifests for deploying the application.
- Managing Kubernetes resources like pods, services, deployments, and ingresses.
- Monitoring the health and performance of the Kubernetes cluster and applications using Prometheus and Grafana.
- Setting up and configuring Prometheus to collect metrics from the Kubernetes cluster and application pods.
- Creating and managing Grafana dashboards to visualize the collected metrics.
- Implementing alerting mechanisms to notify the team of any performance issues or anomalies.

- Working closely with development teams to understand their requirements and ensure the CI/CD pipeline meets their needs.
- Participating in daily stand-ups and team meetings to provide updates on pipeline status, deployments, and any ongoing issues.
- Documenting processes, configurations, and any changes made to the infrastructure or pipeline for future reference.

WHAT IS YOUR CLUSTER SIZE

WE HAVE A 2 CLUSTERS ONE OF FOR PROD AND ANOTHER ONE IS FOR NON-PROD

MY PROD CLUSTER CONTAINS:

- 3 MASTER NODES
- 10 WORKER NODES
- EACH NODE SIZE IS HAVING : 4CPU's & 16GB RAM & 100 GB SSD
- POD SIZE : 0.5 CPU & 1 GB RAM
- WE USED ASG's FOR HIGH AVAILABILITY WHICH INCREASED OUR NODE SIZE UPTO 20
- WE USED APPLICATION LOAD BALANCERS TO ROUTE THE TRAFFIC
- WE USED INGRESS CONTROLLERS FOR MANAGES INCOMING TRAFFIC TO APPLICATIONS RUNNING IN THE CLUSTER. AND IT WILL ROUTE THE TRAFFIC BASED OF HOST OR PATH.
- WE LIMITS 10CPUS & 100GB OF MEMORY FOR EACH NAMESPACE
- WE INTEGRATED HPA FOR OUR DEPLOYMENTS TO SCALE UP THE PODS TO MEET INCREASED TRAFFIC DEMANDS.
- WE USED EBS VOLUMES FOR PV TO STORE THE DATA
- WE USED SECRETS IN K8S TO KEEP SENSITIVE INFORMATION LIKE PASSWORDS, SSH KEYS AND URLS
- WE INTEGRATED PROMETHEUS AND GRAFANA FOR CLUSTER TO COLLECT AND ANALYSE METRICS AND LOGS FROM THE CLUSTER FOR PERFORMANCE MONITORING AND DEBUGGING.
- BACKUP AND DISASTER RECOVERY: REGULARLY SCHEDULED BACKUPS TO A SEPARATE STORAGE LOCATION, DISASTER RECOVERY PLAN DOCUMENTED AND TESTED.
- NAMESPACE ISOLATION: RUNNING DIFFERENT SERVICES ON DIFFERENT NAMESPACES LIKE APPS, MONITOR & ENV LIKE DEV & TEST
- WE IMPLEMENTED RBAC IN OUR CLUSTER, TO PRIVILEGE THE PERMISSIONS TO SPECIFIC USERS.
- WE AUTOMATED THE APPLICATION DEPLOYMENTS
- WE USED CONTAINER IMAGE REGISTRY AS DOCKER HUB TO PUSH/PULL THE IMAGES
- WE IMPLEMENTED ROLLING UPDATED AS A DEPLOYMENT STRATEGIES.

MONOLITHIC PROJECT EXPLANATION

As an experienced DevOps engineer, I managed a project that involved creating and managing highly available infrastructure on AWS using Terraform, automating deployments with Jenkins, and integrating configuration management using Ansible. Additionally, I integrated Slack for real-time notifications and Splunk for logging and monitoring build data.

Day-to-Day Activities of this project

1. **Infrastructure Management**:
   - Writing and maintaining Terraform scripts.
   - Running Terraform commands to provision and update infrastructure.
   - Monitoring the state of the infrastructure and making necessary adjustments.
   - Storing those state files on s3 buckets.
2. **CI/CD Pipeline Management**:
   - Configuring and maintaining Jenkins pipelines.
   - Troubleshooting and resolving build and deployment issues.
   - Integrating new tools and plugins into the Jenkins pipeline as needed.
3. **Configuration Management**:
   - Writing and updating Ansible playbooks.
   - Managing Ansible inventory and configurations.
   - Ensuring successful execution of playbooks on the target servers.
4. **Monitoring and Logging**:
   - Setting up and maintaining Splunk for logging Jenkins build data.
   - Configuring Slack notifications for real-time updates on pipeline status.
   - Monitoring infrastructure and application performance.
5. **Collaboration and Communication**:
   - Working closely with development and operations teams to align on infrastructure requirements and deployment processes.
   - Participating in daily stand-ups and meetings to discuss progress and blockers.
   - Documenting processes and sharing knowledge with team members

JENKINS PROJECT EXPLANATION:

In this project, I deployed a Java application on Tomcat servers using a Jenkins pipeline. The pipeline orchestrated the entire CI/CD process, ensuring that code was fetched from a Git repository, built using Maven, tested with SonarQube, and then deployed to a Tomcat server. Additionally, artifacts were managed and stored in a Nexus repository.

EXPLAIN YOUR PIPELINE STAGES:

Stage 1: Fetching Code from GitHub

The pipeline starts by fetching the code from the GitHub repository to the CI server using Jenkins.

Stage 2: Building the Application

Next, Java and Maven are installed, and the application is built using Maven.

Stage 3: Code Quality Analysis

The SonarQube Scanner plugin is used to analyze the code quality. The SonarQube server is configured in Jenkins, and a token is generated for authentication.

Stage 4: Uploading Artifacts to Nexus

Artifacts are uploaded to the Nexus repository for version control and storage. The Nexus Artifact Uploader plugin is used to generate the necessary pipeline syntax.

Stage 5: Deploying to Tomcat Server

The final step is deploying the built artifact to a Tomcat server using SCP. The SSH Agent plugin in Jenkins facilitates secure file transfer.

Day-to-Day Activities

As a DevOps engineer on this project, my day-to-day activities included:

1. **Source Code Management**:
   - Cloning repositories from GitHub.
   - Managing branches and integrating changes.
2. **CI/CD Pipeline Maintenance**:
   - Setting up and maintaining Jenkins pipelines for continuous integration and continuous deployment.
   - Ensuring the pipeline stages, such as code checkout, build, test, and deployment, are executed correctly.
   - Troubleshooting and resolving any issues that arise during the pipeline execution.
3. **Build Management**:
   - Configuring Maven for building the Java application.
   - Managing dependencies and resolving build issues.

4. **Code Quality Assurance**:
    - Integrating SonarQube with Jenkins for static code analysis.
    - Configuring and maintaining SonarQube projects and quality profiles.
    - Reviewing code quality reports and addressing issues.
5. **Artifact Management**:
    - Configuring and managing Nexus repository for storing build artifacts.
    - Ensuring artifacts are versioned and stored correctly.
    - Handling artifact promotion and deployment.
6. **Deployment**:
    - Deploying artifacts to Tomcat servers using SCP and Jenkins.
    - Managing server configurations and ensuring successful deployments.
    - Monitoring deployments and ensuring application availability.
7. **Monitoring and Optimization**:
    - Monitoring the Jenkins server and pipelines for performance issues.
    - Optimizing the build and deployment processes for efficiency.
    - Implementing best practices for CI/CD and DevOps.
8. **Collaboration and Documentation**:
    - Collaborating with development and QA teams to understand requirements and provide support.
    - Documenting processes, configurations, and any changes made to the infrastructure or pipeline.
    - Participating in daily stand-ups and team meetings to provide updates on pipeline status, deployments, and any ongoing issues.