DevOps

Capstone Project Problem Statement

Perform, Build, and Deploy Automation to Docker Container

Overview:

The goal is to utilize Ansible Configuration Management and CI/CD pipelines to automate the hosting of Docker containers on a Docker host. The objective is to streamline deployment, ensure scalability, and simplify maintenance for efficient container management.

Component	Initial	Defined	Managed	Future Enhancements
Build and deploy	Manual	Automating Build and deployment with build tool and shell scripts	Implementing Orchestration tools like Jenkins to automate build and deployment across multiple environments	Zero downtime deployments with the blue-green mechanism
Software Testing	Manual	Automating using Test automation tools	Implementing JUnit and Selenium test cases	Moving toward automated continuous testing and regression testing
Infrastructure Management	Manual	Implementing Standards and including orchestration tools to manage servers	Implementing monitoring tools, configuration management tools, and server provisioning tools	Utility-based computing that leverages autoscaling
Software Methodology	Waterfall Model	Implementing agile methodology with DevOps	Providing shorter release cycles with a high success rate	Releasing on demand without having any downtime

Problem Statement Scenario:

Our prime goal was to adhere to Agile methodology and implement DevOps to reduce release cycles and increase overall product performance. One of the key focus areas of business was to get some value in the project if we are going to spend money on this migration.

At the same time, the business was expecting more load coming to the application, which may cause slowness in the application.

There were some issues in their existing build and deploy process, which caused delays for development teams:

1. They were deploying monolithic applications, which took a lot of time.
2. It was quite difficult to scale the application in case of a higher load.
3. Development teams were consuming a lot of time in preparing and validating builds locally.
4. Development teams were using the traditional waterfall model, due to which they were not able to release frequently.

We started with better collaboration among the development, testers, and operations teams.

We've largely reached those goals and have seen the following benefits:

- Increase in embracing DevOps culture to streamline product deployment
- More rapid delivery of new features to production
- Minimal manual involvement in building and releasing a software change
- Increased visibility into key metrics to determine code promotion to the next environment

You must use the following tools:

- **GitHub:** To store Application source code
- **Spring Boot Framework**: To develop microservices-based applications
- **GitHub Actions:** CI/CD tool for automating builds and deploys
- **AWS CLI:** To connect with AWS for deploying Helm Charts
- **Docker CLI:** For testing Dockerfile for the Docker build process
- **Ansible:** For deploying Docker containers on a remote Docker host using Ansible Playbook
- **Kubernetes:** Kubernetes Cluster for deploying Containers

Approach & Timeline:

In 2023, Simplilearn embarked on an Agile and DevOps migration initiative. Our strategy consisted of four key objectives: implementing incremental changes, fostering enhanced collaboration, eliminating organizational obstacles wherever feasible, and fostering a culture of continuous improvement.

Day 1: Preparing the source code and designing the build process along with Dockerfile preparation:

- Starting with establishing standards around how we can implement Enterprise software versioning tool, which will help developers to record all changes done by developers. You can check in the complete application source to the Git version control system, and choose either GitHub, Bitbucket, or Gitlab per your preference

- Focusing on automating build automation with the implementation of build scripts to automate build executions. In case we are using containerization, we can prepare a Dockerfile build script to automate the preparation of custom Docker images

Day 2: Implementing Automated Test cases with the help of Junit and Selenium

- Implementing complete automation of test cases by integrating tools like Junit, Selenium, and other testing tools, and focusing on having continuous testing implemented within our project

- Preparing Dockerfile to prepare a customized Docker image

Day 3: Designing GitHub actions pipeline script for performing CI CD workflow.

- Developing GitHub actions script to Build Custom Docker image, push to Docker repository, and prepare/publish helm charts

- Launching a new VM on AWS Console with the help of Terraform and Deploy Kubernetes Cluster on a Remote AWS host

Day 4: Integrating deploy stage to GitHub actions for performing deployment of containers on Kubernetes.

- Adding a deploy stage to the GitHub actions pipeline to deploy a Kubernetes Pod using Helm Chart with Ansible

- Then focusing on continuous improvement to ensure that whatever automation we have done ensures high efficiency and improved quality

Lesson Learnt:

Monolithic Architecture: We started with a Monolithic application that slowed down our migration process, and later on, we divided an application into microservices. This helps to get migrated much faster and produces better results. If we could go back to migrating monolithic applications to microservices, it would be better to go for this migration.

DevOps Tools: We tried out various DevOps tools initially to fulfill our requirements, but if we could have a set of tools suites, it would be quite easy for us to get onboarded to DevOps. This will surely help us save time.

Process: We did not have a set of processes defined when the platform was stood up. We learned that to effectively manage a repeatable build and deploy process, we had to invest in the right processes and procedures upfront.

What's Next:

- Today, we are provisioning our infrastructure manually or using existing servers. We want to proceed to fully automated infrastructure provisioning with the help of tools like Terraform and Ansible.
- Communications and employee engagement will be the drivers of the cultural change that drives the DevOps Practice. So, we would be moving toward improving collaboration to the next level.
- We will work with the security manager to develop and integrate Dev[Sec]Ops strategy.
- We want to define our standard products and services so we can communicate these to our customers.