**The Petclinic Microservices Project Story**

In my recent project, I worked mainly on a micro-services application fully automated. It was a Dockerized Web Application developed in Java Springboot and Spring Cloud Frameworks and integrated with MySQL database. The project was to create full microservices, CI/CD Pipelines, and deployment on a Kubernetes cluster with monitoring. Kubernetes cluster was created and managed with Rancher. Jenkins was used as the CI/CD automation tool and we created all the infrastructure on AWS EC2 Service. Also, we used Git as the version control system during the whole process. We prepared base branches namely master, dev, and release for the DevOps cycle.

The code was developed in Java and Maven was used as the build tool. So I used Maven Wrapper for the testing, packaging, and installing phases. I spun up the development server through a CloudFormation template. I also prepared Dockerfiles for each microservices and, I prepared a Docker Compose file to deploy the application with Docker-Swarm.

In the QA and Dev deployments, we used Docker Swarm, and then we switched to Kubernetes in production and staging.

For making this switch easy we used some tools like Kompose and Kustomize tools. We converted the Docker-compose files to Kubernetes definition files by using the Kompose tool. We also used the Kustomize tool to add some customization to these definition files -like changing replica numbers or image tags.

We also used Rancher for creating, controlling, and monitoring Kubernetes clusters for staging and production environments. We used Jenkins pipeline scripts to deploy the application to both staging and production environments.

Since our app was running on AWS, we used Cloudformation as Infrastructure as a Code (IaaS) to automate the creation of infrastructure in each pipeline. We used Ansible to configure the instances and prepared some playbooks for this purpose. Since we used the AWS Cloud platform, I prepared dynamic inventory that includes EC2 instances into the inventory by checking their tags. In addition, to be able to connect Ansible to our EC2 instances, I wrote an Ansible config file within the Jenkins pipeline.

As DevOps engineers, we prepare development servers for developers. I set up Jenkins Server and installed some plugins such as Docker Pipeline, Github Integration, and Jacoco. My responsibilities were:

1. CI/CD Pipeline. I was responsible for writing CI/CD pipeline scripts for each stage and keeping them up and running. I've created the nightly and weekly pipelines by using Jenkins. I was responsible for creating Dockerfiles and building images.
2. Our Project was running on AWS, so I was in charge of configuring all services that we used in AWS.

I want to mention the steps in the nightly build of the project one by one,

1. First of all, we have created AWS ECR Repo to store images using AWS CLI
2. Secondly, we packaged the app into jars with Maven Wrapper.
3. Then we prepared image tags for Docker Images
4. Next, we built App Docker Images.
5. After that, we pushed the images to the ECR Repo by using AWS CLI.
6. And then we created a key pair for Ansible and prepared ansible.config file and inventory files.
7. Following this, we created QA automation infrastructure by using Cloudformation.
8. Then, we created a Docker Swarm cluster for QA automation build using Ansible.
9. Next, we deployed the App on Docker Swarm using the Docker-compose file.
10. After that, we run Functional Tests on QA Environment using the Ansible playbook file. We automated functional tests with Selenium.
11. Finally, we set the pipeline to delete all local images, the repository, and the cloudformation stack automatically.
12. The next day morning, we check the Jenkins logs to see if there is any error or  
    failed job and then check the Grafana dashboards to see the performance or any  
    metric fails.

I want to talk about the Rancher -Kubernetes cluster orchestration tool- used in our project:

I used Rancher to create and manage our Kubernetes clusters. To install the Rancher, I used the Helm chart. With Rancher, we easily made changes in the cluster via its dashboard, add nodes, delete nodes, edit configuration files, and used kubectl on its terminal.

As monitoring tools: we monitored the applications in the cluster with Prometheus and Grafana.

**Petclinic Mikro Servis Projesi Senaryosu**

Son projemde, tam otomatize bir mikro servis uygulaması üzerinde çalıştım. Java Springboot ve Spring Cloud Frameworks'te geliştirilmiş ve MySQL veritabanı ile entegre edilmiş Dockerize bir Web Uygulamasıydı. Projenin amacı Kubernetes clusterda izlenen bir full mikro servis uygulamasının CI/CD Pipelinını kurmak ve deploy etmekti. Kubernetes cluster Rancher ile oluşturuldu ve yönetildi. CI/CD otomasyon aracı olarak Jenkins kullanıldı ve tüm altyapıyı AWS EC2 Servisi üzerinde oluşturduk. Ayrıca tüm süreç boyunca versiyon kontrol sistemi olarak Git'i kullandık. DevOps döngüsü için master, dev ve release olmak üzere temel branchlar hazırladık.

Kod Java'da geliştirildi ve build tool olarak Maven kullanıldı. Bu yüzden test, packaging, and installing aşamaları için Maven Wrapper'ı kullandım. Development serverı bir CloudFormation template aracılığıyla çalıştırdım. Ayrıca her mikroservis için Dockerfile hazırladım ve uygulamayı Docker-Swarm ile deploy etmek için bir Docker Compose dosyası hazırladım.

QA ve Dev deploymentlarında Docker Swarm kullandık, production ve staging safhalarında Kubernetes'e geçtik.

Bu geçişi kolaylaştırmak için Kompose ve Kustomize gibi toollar kullandık. Docker-compose dosyalarını Kompose toolunu kullanarak Kubernetes definition dosyalarına dönüştürdük. Ayrıca, bu definition dosyalarına bazı özelleştirmeler eklemek için Kustomize toolunu kullandık (replika sayılarını veya image etiketlerini değiştirmek gibi).

Staging ve production environmentlarındaki Kubernetes clusterlarını oluşturmak, check etmek ve izlemek için Rancher'ı da kullandık. Uygulamayı hem staging hem de production environmentlara deploy etmek için Jenkins pipeline scriptlerini kullandık.

Uygulamamız AWS üzerinde çalıştığından, her bir pipelineda infrastructure oluşturulmasını otomatize etmek için IaaS (Infrastructure as a Code) olarak Cloudformation kullandık. Instanceları konfigüre etmek için Ansible'ı kullandık ve bu amaçla bazı oyun playbooklar hazırladık. AWS Cloud platformunu kullandığımız için EC2 instance'larının tag'lerini kontrol ederek inventory'ye dahil eden dinamik inventory hazırladım. İlaveten, Ansible'ı EC2 bulut sunucularımıza bağlayabilmek için Jenkins pipelineıyla bir Ansible config file yazdım.

DevOps mühendisleri olarak developerlar için development serverlar hazırlıyoruz. Jenkins Server'ı kurdum ve içine Docker Pipeline, Github Integration ve Jacoco gibi bazı pluginler kurdum. Sorumluluklarım şunlardı:

1. CI/CD Pipeline. Her stage için CI/CD pipeline scriptlerini yazmaktan ve bunları çalışır durumda tutmaktan sorumluydum. Jenkins'i kullanarak gecelik ve haftalık pipelinelar kurdum. Dockerfile oluşturmaktan ve imajlar build etmekten sorumluydum.

2. Projemiz AWS üzerinde çalışıyordu, bu yüzden AWS'de kullandığımız tüm servisleri konfigüre etmekten sorumluydum.

Projenin gecelik build'taki adımlarından tek tek bahsetmek istiyorum,

1. Öncelikle imajları depolamak için AWS CLI kullanarak AWS ECR Repo'yu oluşturduk.

2. Sonra, Maven Wrapper ile app'i jar'lara paketledik.

3. Ardından Docker imajları için imaj tag'leri hazırladık.

4. Ardından, App Docker imajları oluşturduk.

5. Ardından AWS CLI kullanarak imajları ECR Repo'ya push ettik.

6. Ardından Ansible için bir key pair oluşturduk ve ansible.config dosyası ile inventory file'ları hazırladık.

7. Ardından Cloudformation kullanarak QA otomasyon infrastructure'ını oluşturduk.

8. Ardından, Ansible kullanarak QA otomasyonu build etmesi için bir Docker Swarm cluster'ı oluşturduk.

9. Ardından, Docker-compose dosyasını kullanarak App'i Docker Swarm'a deploy ettik.

10. Ardından Ansible playbook dosyasını kullanarak QA Environment üzerinde Functional Testleri çalıştırdık. Functional Testleri Selenium ile otomatize ettik.

11. Son olarak; tüm lokal imajları, repo'ları ve cloudformation stack'leri otomatik olarak silecek şekilde bir pipeline kurduk.

12. Ertesi gün sabah, herhangi bir error veya fail job olup olmadığını görmek için Jenkins log'larını kontrol ettik. Ayrıca performansı veya metrik fail'leri görmek için Grafana dashboardlarını kontrol ettik.

Projemizde kullandığımız ve bir Kubernetes cluster orkestrasyon aracı olan Rancher hakkında konuşmak istiyorum:

Kubernetes clusterlarımızı oluşturmak ve yönetmek için Rancher'ı kullandım. Rancher'ı kurmak için Helm tablosunu kullandım. Rancher ile, dashboard aracılığıyla cluster'da kolayca değişiklikler yaptık, node'lar ekledik, node'lar sildik, config file'ları düzenledik ve kendi terminalinde kubectl kullandık.

İzleme araçları olarak: Prometheus ve Grafana ile cluster'daki application'ları izledik.