**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.