**NAME: PRANAV.B**

**REG NO: 18MIS0072**

**INDUSTRIAL CERTIFICATION REPORT**

**CERTIFICATION:**

SmartBridge Industry Certificate Internship Program on **AWS DEVOPS ENGINEER**

**COURSE OVERVIEW:**

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

AWS provides a set of flexible services designed to enable companies to more rapidly and reliably build and deliver products using AWS and DevOps practices. These services simplify provisioning and managing infrastructure, deploying application code, automating software release processes, and monitoring your application and infrastructure performance.

I did assignments and uploaded in smart bridge Google drive. After finishing all the modules, on the last week of the internship program, I did a project on any of the important concepts mentioned above to have a clear understanding of DevOps for us.

**MODULE WISE SUMMARY:**

|  |  |  |
| --- | --- | --- |
| **Module** | **Title** | **Content** |
| **Module-1** | **Introduction to DevOps** | * 1. **Introduction**      + What isDevOps?      + Roles and Responsibilities of a DevOpsEngineer      + How DevOps fits in the whole Software Developmentlifecycle      + DevOpstools |
| **Module 2** | **Version Control with Git** | * 1. **Introduction to Version Control andGit**      + Basic Concepts ofGit      + Setup git repository (remote andlocal)      + Working with Git (git status, git commit, git add, gitpush)      + Initialize Git project locally      + Concept ofBranches      + MergeRequests      + DeletingBranches      + Avoiding MergeCommits      + Resolving MergeConflicts      + Don't track certain files(.gitignore)      + Save work-in-progress changes (gitstash)      + Going back in history (gitcheckout)      + Undoing commits (git revert, gitreset)      + MergingBranches      + Git forDevOps |
| **Module 3** | **Containers with Docker** | * 1. **Containers withDocker**      + What is aContainer?      + Docker Components and architectureexplained      + Docker vs. VirtualMachine      + Main DockerCommands      + Debugging a DockerContainer      + Demo Project Overview - Docker in Practice (Nodejs App with MongoDB and Mongo Express UI)      + Developing withContainers      + Docker Compose - Running multipleservices      + Dockerfile - Building our own DockerImage      + Private Docker Repository - Pushing our built Docker Image into a private Registry onAWS      + Deploy containerizedapp      + Docker Volumes - Persist data inDocker      + Volumes Demo - Configure persistence for our demo project |

|  |  |  |
| --- | --- | --- |
| **Module 4** | **Build Automation - CI/CD with Jenkins** | * 1. **Jenkins**      + What is Build Automation? What isJenkins?      + Install Jenkins on cloud server (Docker vs Server install)      + Jenkinsplugins      + Installing build tools inJenkins      + Jenkins BasicsDemo      + Create FreestyleJob      + Configure GitRepository      + Run Tests and Build JavaApplication      + Docker inJenkins      + Make Docker commands available inJenkins      + Build DockerImage      + Push to DockerHubRepo      + Jenkins Pipeline (UseCases)      + Create a simple PipelineJob      + Full Jenkins file SyntaxDemo      + Create a full PipelineJob      + Build JavaApp      + Build DockerImage      + Push to privateDockerHub      + Create a Multi-Branch PipelineJob      + Credentials inJenkins      + Jenkins SharedLibrary      + WebHooks - Trigger Jenkins Jobsautomatically      + Versioning Application in ContinuousDeployment      + Concepts of Versioning in SoftwareDevelopment      + Increment Application version from Jenkins Pipeline      + Set new Docker Image version from Jenkins Pipeline      + Commit Version Bump from JenkinsPipeline |
| **Module 5** | **AWS Services** | * 1. **Services**      + Introduction to Amazon WebServices      + Identity & Access Management (IAM) - User, Groups andPermissions      + Regions and AvailabilityZones      + Virtual Private Cloud (VPC) - Your PrivateNetwork      + Subnets      + SecurityGroups      + InternetGateway      + Route Table      + CIDRBlocks      + Introduction to Elastic Compute Cloud(EC2)      + Create an EC2Instance      + Run Web application on EC2 usingDocker      + AWS Command LineTool * Install and configure AWSCLI * CreateEC2 * Create SecurityGroup * Createkey-pair * Create IAM user withpermissions   1. **AWS & Jenkins - Continuous Deployment with Jenkins to AWS EC2**      + Automate deploying from Jenkins Pipeline to EC2 Instance      + using dockerrun      + usingdocker-compose      + Real-life example of dynamically setting new image version indocker-compose      + SSH agent plugin and SSH credential type in Jenkins |

|  |  |  |
| --- | --- | --- |
| **Module 6** | **Container Orchestration with Kubernetes** | * 1. **Container Orchestration with Kubernetes**      + Introduction toKubernetes      + Understand the main KubernetesComponents      + Node, Pod, Service, Ingress, ConfigMap, Secret, Volume, Deployment,StatefulSet      + Kubernetes Architecture      + Minikube and kubectl - LocalSetup      + Main Kubectl Commands - K8sCLI      + Create and debug Pod in aMinicluster      + Kubernetes YAML ConfigurationFile      + Create and Configure Deployment and Service Component      + Demo Project: MongoDB andMongoExpress      + Organizing your components with K8s Namespaces      + Kubernetes ServiceTypes |

|  |  |  |
| --- | --- | --- |
| **Module 7** | **Kubernetes on AWS - EKS** | * 1. **AWS &Kubernetes**      + AWS Container Services: Overview (ECR, ECS, EKS, Fargate)      + Create an EKS cluster with AWS Management Console (UI)      + Create clusterVPC      + Create clusterRoles      + Use CloudformationStack      + EC2 WorkerNodes      + FargateProfile      + ConfigureAustoscaler      + Configure kube context to connect to thecluster      + Create an EKS cluster with eksctl (the easyway)   2. **AWS & Kubernetes & Jenkins & Docker -CI/CD**      + Configure kubectl insideJenkins      + Configure kube context inJenkins      + Install aws-iam-authenticator inJenkins      + Complete Jenkins Pipeline - Deploy to EKS - using kubectl      + Complete Jenkins Pipeline - Build and push docker image to ECR and deploy toEKS      + Complete Jenkins Pipeline - Deploy to LKE using Kubernetes CLI plugin and kubeconfigfile |
| **Module 8** | **Infrastructure as Code with Terraform** | * 1. **Terraform**      + What is Terraform? How itworks      + Architecture      + Providers      + Resources & DataSources      + Variables & OutputValues      + Environment variables inTerraform      + Terraformcommands      + TerraformState      + Provisioners      + Modules      + Remote State   **Terraform & AWS**   * + - Create SecurityGroup     - CreateVPC     - CreateSubnet     - Create RouteTable     - Create InternetGateway     - Createkey-pair * Provision EC2server * Modularize the demoproject   **Terraform & AWS & Kubernetes**   * Use existing modules from TerraformRegistry * CreateVPC * Provision EKScluster   **Terraform & AWS & Jenkins - complete CI/CD**   * Configure Terraform inJenkins   + - Automate provisioning EC2 instance from Jenkins pipeline and deploy the application with docker- compose |

**SUMMARY OF IMPORTANT AWS CONCEPTS COVERED:**

**Git Bash and Git Hub:**

Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience. Bash is an acronym for Bourne Again Shell. A shell is a terminal application used to interface with an operating system through written commands. Bash is a popular default shell on Linux and macOS.

GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

**Docker:**

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production. Using docker we containers, A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.

**Jenkins:**

Jenkins is an open source automation server. With Jenkins, organizations can accelerate the software development process by automating it. Jenkins manages and controls software delivery processes throughout the entire lifecycle, including build, document, test, package, stage, deployment, static code analysis and much more.

You can set up Jenkins to watch for any code changes in places like GitHub, Bitbucket or GitLab and automatically do a build with tools like Maven and Gradle. You can utilize container technology such as Docker and Kubernetes, initiate tests and then take actions like rolling back or rolling forward in production.

**Flask App Development:**

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website. Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

**Kubernetes:**

Kubernetes, also known as K8s, is an open source system for managing containerized applications across multiple hosts. It provides basic mechanisms for deployment, maintenance, and scaling of applications.Kubernetes builds upon a decade and a half of experience at Google running production workloads at scale using a system called Borg, combined with best-of-breed ideas and practices from the community.Kubernetes is hosted by the Cloud Native Computing Foundation (CNCF). If your company wants to help shape the evolution of technologies that are container-packaged, dynamically scheduled, and microservices-oriented, consider joining the CNCF. For details about who's involved and how Kubernetes plays a role, read the CNCF announcement.

**Amazon Web Services:**

Amazon Web Services provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world. With data center locations in the U.S., Europe, Singapore, and Japan, customers across all industries are taking advantage of our low cost, elastic, open and flexible, secure platform. AWS is made up of many different cloud computing products and services. The highly profitable division of Amazon provides servers, storage, networking, remote computing, email, mobile development, and security. AWS can be broken into three main products: EC2, Amazon’s virtual machine service, Glacier, a low-cost cloud storage service, and S3, Amazon’s storage system

SUMMARY OF OUR PROJECT:

I did a project on **Build a Plasma Donor App with AWS Serverless Computing.** During the COVID 19 crisis, the requirement of plasma became high and the donor count being low. Saving the donor information and helping the need by notifying the current donors would be a helping hand. In regard to the problem faced, an application is to be built which would take the donor details store it and inform them upon a request.

Serverless computing is the current trend of software application development. Micro services are a popular new approach for building maintainable, scalable, cloud-based applications. AWS is the perfect platform for hosting micro-services. In this project we will be building a plasma donor app with AWS services like lambda functions, API gateway and DynamoDB.

**This project contains the following Activities:**

We used DevOps tools which were very useful in automating the whole deployment process,

1. Required initial steps
2. Sign in to AWS console
3. Create a database
4. Create APIs to

* Push the registration data into database.
* Fetch the data upon login.
* Display the stats of different blood groups.
* Upon request, take the required blood group and return the details

1. Create a flask application

* Registration page
* Login page
* Stats page to display the count
* Request page
* Test it

1. Deploy the application

* Create an EC2 instance.
* Deploy the app.

By completing this project, I

* Learnt fundamental concepts and techniques of DevOpsand AWS services
* Gained a broad understanding of Git and code versioning.
* Learnt how to containerize the application.
* Learnt to automate the application using CI/CD pipeline.
* Learnt to deploy application in EC2 instance.
* Knew how to build a web application using Flask framework.

**Project application Link:**

<https://github.com/smartinternz02/SI-GuidedProject-5187-1627992883>