**PROJECT DESCRIPTION**

The objective of this project is to establish a Jenkins pipeline for deploying a web page built with Node.js and React Frameworks on the AWS Cloud Infrastructure through the utilization of Ansible. The building of the infrastructure is managed by a control node utilizing Ansible. This infrastructure consists of one Jenkins server which will be created by Terraform files (Amazon Linux 2 AMI) serving as the Ansible control node and three EC2 instances operating as worker nodes (Red Hat Enterprise Linux 8 with High Availability), which will be created by Terraform files and launched on the AWS. The web page has three primary components: PostgreSQL, Node.js, and React. Each component operates within a Docker container on dedicated EC2 instances. PostgreSQL functions as the web page's database, while Node.js manages the backend, and React handles the frontend. The code has been developed by Developers, and the architecture will be created by Cloud & DevOps Team.

**LEARNING OBJECTIVES**

**Jenkins Pipeline Setup**

**Establish infrastructure using Terraform**

**Deploy the application using Ansible**

**Prepare Ansible playbook**

**Generate Docker images for postgresql, nodejs, and react containers**

**Launch Docker containers using the created images with an Ansible playbook**

**Scripting in Bash**

**Utilize AWS ECR as the Image Repository**

**Configure AWS IAM Policies and Roles**

**Set up AWS EC2 Launch Template**

**Configure AWS EC2 Instances**

**Set up AWS EC2 Security Groups**

**Utilize Git & Github as the Version Control System**

**STATEMENT OF THE ISSUE**

Your organization has completed a web page project, and the DevOps team is tasked with deploying it. The Full stack development team has finalized the code, and the DevOps team plans to use Ansible for deploying the application in the production environment.

* The web page is designed for users to input their information, and the data needs to be stored in a PostgreSQL database on a dedicated EC2 instance.
* The backend is built with Node.js and runs on port 5000, connected to the PostgreSQL database on port 5432.
* The frontend is managed by React, serving on port 3000 and communicating with the Node.js server.
* The goal is to deploy the web application, making it accessible from any web browser on port 3000.
* EC2 instances and their corresponding security groups should be created using Terraform on the AWS console.
* The deployment process will be managed by a control node.
* The code will be pulled from the repository into a Jenkins server and then sent to the EC2 instances using Ansible.
* To containerize the application, PostgreSQL, Node.js, and React components will be placed in Docker containers.
* The project manager has specified launching separate EC2 instances for each container, and AWS ECR will be used as the image repository. A single playbook will be created to install and configure Docker on each instance and to create containers for PostgreSQL, Node.js, and React.

**Please design your architecture based on below requirements:**

1. Create a new private repository for the project on GitHub.
2. Jenkins server should be act as control node.
3. Utilize dynamic inventory file for managing inventory.
4. Place the Ansible configuration file (ansible.cfg) on the Jenkins Server.
5. Use Ansible to install Docker on all worker nodes.
6. Build Docker images on the Jenkins Server and push them to AWS ECR.

**For the PostgreSQL worker node:**

1. Build a Docker image for the PostgreSQL container using 'dockerfile-postgresql,' and place the 'init.sql' file in the required directory.
2. On the Jenkins server, build Docker images from 'dockerfile-postgresql' and push them to AWS ECR.
3. Deploy a PostgreSQL container on the worker node, ensuring the password is set as an environmental variable.
4. Ensure the security group of this instance allows traffic from the Nodejs EC2 on PostgreSQL's dedicated port and allows traffic on port 22 from anywhere.
5. To keep the database's data, create a volume with the Docker container and store the necessary file(s) within this volume.

**For the NodeJs worker node:**

1. Ensure the creation or correction of the '.env' file under the 'server' folder, containing PostgreSQL environmental variables. Use the Linux 'envsubst' command with 'env-template' for automating the inclusion of the PostgreSQL instance's private IP.
2. Develop a Docker image for the Node.js container using 'dockerfile-nodejs,' and store the 'server' files in the required directory. No additional files are needed for creating the Node.js image.
3. On the Jenkins server, build the Docker image for the Node.js container from 'dockerfile-nodejs' and push it to AWS ECR.
4. Deploy the Node.js container on the Node.js instance using Ansible and expose it on port 5000.
5. Ensure that the security group of this instance permits traffic on port 5000 and port 22 from anywhere.

**For the React worker node:**

1. Ensure the creation or correction of the '.env' file under the 'client' folder, containing NodeJs environmental variables. Use the Linux 'envsubst' command with 'env-template' for automating the inclusion of the NodeJs instance's public IP.
2. Develop a Docker image for the React container using 'dockerfile-react,' and store the 'client' files in the required directory. No additional files are needed for creating the React image.
3. On the Jenkins server, build the Docker image for the React container from 'dockerfile-react' and push it to AWS ECR.
4. Deploy React container and publish it on port 3000.
5. Ensure the security group for this instance allows traffic on ports 3000 and 80 from anywhere.

**To use AWS ECR as image repository:**

1. Grant instances the ability to interact with ECR repositories by enabling IAM Role through the instance profile.
2. Install AWS CLI Version 2 on worker node instances to utilize aws ecr commands.

**To create a Jenkins Pipeline:**

1. You need to launch a Jenkins Server with security group allowing SSH (port 22) and HTTP (ports 80, 8080) connections. (Deploy Jenkins Server with Terraform)

**Create a Jenkins Pipeline with following configuration:**

1. Build the infrastructure for the app on EC2 instances using Terraform configuration file.
2. Create an image repository on ECR for the app.
3. Build the application Docker image and push it to the same ECR repository with different tags.
4. Deploy the application on AWS EC2's with ansible.
5. Make a failure stage and ensure to destroy infrastructure, ECR repo and docker images when the pipeline failed.