**Infrastructure Orchestration with AWS CloudFormation and Ansible:**

**Description:**

**Automate the provisioning and configuration of AWS infrastructure using AWS CloudFormation**

**for infrastructure orchestration and Ansible for configuration management.**

***Write CloudFormation templates to define the AWS resources (e.g., EC2 instances, RDS***

***databases, VPC) needed for the application.***

**VPC - yaml file**

**Resources:**

**MyVPC:**

**Type: AWS::EC2::VPC**

**Properties:**

**CidrBlock: 10.0.0.0/16**

**EnableDnsSupport: true**

**EnableDnsHostnames: true**

**Tags:**

**- Key: Name**

**Value: MyVPC**

**InternetGateway:**

**Type: AWS::EC2::InternetGateway**

**Properties:**

**Tags:**

**- Key: Name**

**Value: MyInternetGateway**

**AttachGateway:**

**Type: AWS::EC2::VPCGatewayAttachment**

**Properties:**

**VPCId: !Ref MyVPC**

**InternetGatewayId: !Ref InternetGateway**

**MySubnet:**

**Type: AWS::EC2::Subnet**

**Properties:**

**VpcId: !Ref MyVPC**

**CidrBlock: 10.0.1.0/24**

**MapPublicIpOnLaunch: true**

**Tags:**

**- Key: Name**

**Value: MySubnet**

**MyRouteTable:**

**Type: AWS::EC2::RouteTable**

**Properties:**

**VpcId: !Ref MyVPC**

**Tags:**

**- Key: Name**

**Value: MyRouteTable**

**MyRoute:**

**Type: AWS::EC2::Route**

**DependsOn: AttachGateway**

**Properties:**

**RouteTableId: !Ref MyRouteTable**

**DestinationCidrBlock: 0.0.0.0/0**

**GatewayId: !Ref InternetGateway**

**SubnetRouteTableAssociation:**

**Type: AWS::EC2::SubnetRouteTableAssociation**

**Properties:**

**SubnetId: !Ref MySubnet**

**RouteTableId: !Ref MyRouteTable**

**MySecurityGroup:**

**Type: AWS::EC2::SecurityGroup**

**Properties:**

**GroupDescription: Allow http to client host**

**VpcId: !Ref MyVPC**

**SecurityGroupIngress:**

**- IpProtocol: tcp**

**FromPort: 80**

**ToPort: 80**

**CidrIp: 0.0.0.0/0**

**- IpProtocol: tcp**

**FromPort: 22**

**ToPort: 22**

**CidrIp: 0.0.0.0/0**

**EC2 – Instance – yaml file**

**MyEC2Instance:**

**Type: AWS::EC2::Instance**

**Properties:**

**InstanceType: t2.micro**

**ImageId: ami-0abcdef1234567890 # Update this to a valid AMI for your region**

**SubnetId: !Ref MySubnet**

**SecurityGroupIds:**

**- !Ref MySecurityGroup**

**Tags:**

**- Key: Name**

**Value: MyEC2Instance**

**RDS – Subnetgroup**

**MyDBSubnetGroup:**

**Type: AWS::RDS::DBSubnetGroup**

**Properties:**

**DBSubnetGroupDescription: My DB Subnet Group**

**SubnetIds:**

**- !Ref MySubnet**

**DBSubnetGroupName: MyDBSubnetGroup**

**DB – Instance**

**MyDBInstance:**

**Type: AWS::RDS::DBInstance**

**Properties:**

**DBInstanceClass: db.t2.micro**

**AllocatedStorage: 20**

**Engine: mysql**

**MasterUsername: admin**

**MasterUserPassword: verysecurepassword # Change this**

**DBSubnetGroupName: !Ref MyDBSubnetGroup**

**VPCSecurityGroups:**

**- !Ref MySecurityGroup**

**Outputs:**

**WebAppPublicIP:**

**Description: The Public IP address of the web app EC2 instance**

**Value: !GetAtt MyEC2Instance.PublicIp**

**Infrastructure Provisioning (CloudFormation):**

**1 . Log in to the AWS Management Console.**

**2 . Navigate to the CloudFormation service.**

**3 . Click Create stack > With new resources (standard).**

**4 . Specify template: Choose Upload a template file, click Choose file, and upload your CloudFormation template file (.yaml or .json).**

**5 . Click Next.**

**6 . Specify stack details: Enter a stack name and any parameters required by your template.**

**7 . Click Next. Optionally, configure stack options such as tags, permissions, and other advanced settings, then click Next again.**

**8 . Review: Check your stack details and settings. Acknowledge that AWS CloudFormation might create IAM resources if your stack template includes them.**

**9 . Click Create stack. AWS CloudFormation will start creating the resources defined in your template.**

**Using AWS CLI**

**First, ensure your template file (my-cloudformation-template.yaml) is ready and accessible.**

1. Open your terminal.
2. Use the AWS CLI to deploy your CloudFormation stack by running:

**aws cloudformation create-stack --stack-name MyStack --template-body file://my-cloudformation-template.yaml --parameters ParameterKey=Param1,ParameterValue=Value1 ParameterKey=Param2,ParameterValue=Value2 --capabilities CAPABILITY\_IAM**

**Server Configuration (Ansible):**

**1. Environment Setup**

**First, ensure you have Ansible installed in your environment. If not, you can install it using Python's package manager pip:**

**$ pip install ansible**

**2. Install AWS CLI and Configure AWS Access**

**To manage AWS resources, you'll need the AWS CLI tool installed and configured with the necessary access credentials.**

**$ pip install awscli**

**$ aws configure**

**3. Install Boto3**

**Boto3 is the Amazon Web Services (AWS) SDK for Python. It allows Python developers to write software that uses services like Amazon S3 and Amazon EC2. Ansible uses Boto3 for AWS dynamic inventory.**

**$ pip install boto3**

**Create a Directory For Your AWS Project**

**$ mkdir aws\_ansible\_project && cd aws\_ansible\_project**

**Download the AWS EC2 Plugin for Dynamic Inventory:**

**Ansible includes built-in plugins for AWS resources. You can enable the EC2 plugin by creating a aws\_ec2.yaml file in your project directory with the following content:**

**$ plugin: aws\_ec2**

**regions:**

**- us-east-1 # Your AWS region**

**keyed\_groups:**

**- key: tags**

**prefix: tag**

**Writing Ansible Playbooks**

**Now, let's write a playbook to configure your AWS resources. This example playbook installs nginx on all your AWS EC2 instances that are discovered dynamically.**

**Create a playbook file: install\_nginx.yml**

**---**

**- name: Install Nginx on AWS EC2 instances**

**hosts: all**

**become: true**

**tasks:**

**- name: Update apt-get repo and cache**

**apt:**

**update\_cache: yes**

**force\_apt\_get: yes**

**when: ansible\_os\_family == "Debian"**

**- name: Install nginx**

**apt:**

**name: nginx**

**state: latest**

**when: ansible\_os\_family == "Debian"**

**Run the Playbook**

**ansible-playbook -i aws\_ec2.yaml install\_nginx.yml**