**Puppet configuration**

**1️⃣ Connected with the DevOps team to create a new environment similar to UAT.**

* **Discussed infrastructure requirements (RAM, OS, installed software, etc.).**
* **Identified services, configurations, and databases needed.**

**2️⃣ Provisioned (Created) a Virtual Machine using Terraform**

* **Terraform script (.tf file) was used to create the VM with specifications like:**
  + **OS → Linux (e.g., Ubuntu 20.04)**
  + **RAM → 8GB**
  + **CPU → 4 cores**
  + **Disk → 100GB SSD**
  + **Security Groups → Open required ports (e.g., 80 for HTTP, 443 for HTTPS)**
* **Example Terraform script:**

**h**

**CopyEdit**

**resource "aws\_instance" "uat\_vm" {**

**ami = "ami-12345678" # Linux OS**

**instance\_type = "t2.medium" # 8GB RAM, 4 Cores**

**key\_name = "uat-keypair"**

**security\_groups = ["uat-security-group"]**

**tags = {**

**Name = "UAT-Clone"**

**}**

**}**

**3️⃣ Configured the VM using Puppet (.pp file)**

* **Puppet Manifest (setup.pp) was used to install required software:  
  ✅ Java, Python, Node.js  
  ✅ Nginx (Web Server)  
  ✅ PostgreSQL / MySQL (Database)  
  ✅ Firewall and security settings**
* **Example Puppet script:**

**puppet**

**CopyEdit**

**# Install Java**

**package { 'java-11-openjdk':**

**ensure => installed,**

**}**

**# Install MySQL**

**package { 'mysql-server':**

**ensure => installed,**

**}**

**# Ensure MySQL is running**

**service { 'mysql':**

**ensure => running,**

**enable => true,**

**}**

**4️⃣ Linked the environment with Jenkins & GitHub for CI/CD**

* **Jenkins: Automated deployment and test execution.**
* **GitHub: Code repository, linked with Jenkins to trigger builds.**
* **Steps:  
  ✅ Set up a Jenkins job to pull code from GitHub  
  ✅ Build & deploy code to the new environment  
  ✅ Run automated test cases**

**5️⃣ Copied the database from UAT using DBeaver**

* **Used DBeaver to export UAT database:**
  + **Backup Command: pg\_dump -U user -d uat\_db -f backup.sql**
* **Restored in the new environment:**
  + **Restore Command: psql -U user -d new\_db -f backup.sql**

**6️⃣ Monitored logs and errors using Loggly & other monitoring tools**

* **Loggly: Used to track logs and error messages.**
* **Prometheus/Grafana: (If used) Monitored system health (CPU, memory, disk usage).**
* **Checked errors, ensured the environment was stable.**

**Step-by-Step Practical Implementation**

**🛠 1. Setting Up the Virtual Machine Using Terraform**

**Terraform is used to create the infrastructure (VM, network, security groups, etc.).**

**✅ Practical Steps**

**1️⃣ Install Terraform (if not installed):**

* **Linux/macOS: sudo apt install terraform**
* **Windows: Download from Terraform official site**

**2️⃣ Write a Terraform configuration file (main.tf)**

* **Open VS Code or any text editor and create a new file main.tf.**
* **Copy-paste the following Terraform script:**

**hcl**

**CopyEdit**

**provider "aws" {**

**region = "us-east-1"**

**}**

**resource "aws\_instance" "uat\_vm" {**

**ami = "ami-12345678" # Replace with actual Linux AMI ID**

**instance\_type = "t2.medium" # 8GB RAM, 4 Cores**

**key\_name = "uat-keypair" # SSH Key for access**

**security\_groups = ["uat-security-group"]**

**tags = {**

**Name = "UAT-Clone"**

**}**

**}**

**3️⃣ Initialize Terraform and apply the configuration:**

**sh**

**CopyEdit**

**terraform init # Initializes Terraform**

**terraform apply -auto-approve # Creates the VM**

**4️⃣ Verify the VM is created:**

* **Log in to AWS/Azure/GCP and check the EC2/VM Instances section.**
* **Use SSH to connect to the VM:**

**sh**

**CopyEdit**

**ssh -i uat-keypair.pem ubuntu@<public-ip>**

**🔹 2. Installing & Configuring Software Using Puppet**

**Puppet helps in automating software installation & configuration.**

**✅ Practical Steps**

**1️⃣ Install Puppet on the VM**

* **Run the following commands on the newly created VM:**

**sh**

**CopyEdit**

**sudo apt update**

**sudo apt install puppet -y**

**2️⃣ Create a Puppet manifest file (setup.pp)**

* **Create the file:**

**sh**

**CopyEdit**

**nano /etc/puppet/code/environments/production/manifests/setup.pp**

* **Add this Puppet configuration to install Java, MySQL, and Nginx:**

**puppet**

**CopyEdit**

**package { 'java-11-openjdk':**

**ensure => installed,**

**}**

**package { 'mysql-server':**

**ensure => installed,**

**}**

**package { 'nginx':**

**ensure => installed,**

**}**

**service { 'nginx':**

**ensure => running,**

**enable => true,**

**}**

**3️⃣ Apply the Puppet manifest:**

**sh**

**CopyEdit**

**sudo puppet apply /etc/puppet/code/environments/production/manifests/setup.pp**

**4️⃣ Verify installation:**

**sh**

**CopyEdit**

**java -version**

**mysql --version**

**systemctl status nginx**

**🔹 3. Linking with Jenkins & GitHub for CI/CD**

**Jenkins automates deployment & testing by pulling code from GitHub.**

**✅ Practical Steps**

**1️⃣ Install Jenkins on the VM**

**sh**

**CopyEdit**

**wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -**

**sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'**

**sudo apt update**

**sudo apt install jenkins -y**

**sudo systemctl start jenkins**

**sudo systemctl enable jenkins**

**2️⃣ Access Jenkins Web UI**

* **Open: http://<public-ip>:8080**
* **Enter the initial admin password:**

**sh**

**CopyEdit**

**sudo cat /var/lib/jenkins/secrets/initialAdminPassword**

* **Install plugins & create an admin user.**

**3️⃣ Integrate Jenkins with GitHub**

* **Go to Jenkins Dashboard → Manage Jenkins → Configure System**
* **Add GitHub repository URL under "Source Code Management"**
* **Configure "Build Triggers" → Select "Poll SCM" (Checks for new commits).**
* **Add a build step:**

**sh**

**CopyEdit**

**git clone https://github.com/your-repo.git**

**mvn clean install # (For Java projects)**

**pytest tests/ # (For Python test execution)**

**🔹 4. Migrating the Database Using DBeaver**

**We copy the UAT database to the new environment.**

**✅ Practical Steps**

**1️⃣ Export Database from UAT (Using DBeaver)**

* **Connect to UAT database in DBeaver**
* **Right-click on the database → Export Database → SQL Format (.sql)**
* **Save it as backup.sql**

**2️⃣ Transfer backup.sql to the new environment**

**sh**

**CopyEdit**

**scp -i uat-keypair.pem backup.sql ubuntu@<public-ip>:~**

**3️⃣ Restore Database in the New VM**

**sh**

**CopyEdit**

**mysql -u root -p new\_db < backup.sql**

**4️⃣ Verify Data**

**sh**

**CopyEdit**

**mysql -u root -p -e "SELECT COUNT(\*) FROM users;" new\_db**

**🔹 5. Monitoring Logs & System Health**

**Tools like Loggly, Prometheus, and Grafana help monitor errors.**

**✅ Practical Steps**

**1️⃣ Install Loggly Agent**

**sh**

**CopyEdit**

**sudo apt install loggly-agent**

**sudo loggly configure -a mycompany -u admin@company.com**

**2️⃣ Check logs in real-time**

**sh**

**CopyEdit**

**tail -f /var/log/syslog**

**3️⃣ Install Prometheus (for system monitoring)**

**sh**

**CopyEdit**

**sudo apt update**

**sudo apt install prometheus -y**

**systemctl start prometheus**

**4️⃣ Access Prometheus Dashboard**

* **Open browser: http://<public-ip>:9090**
* **Query example: up (checks if system is running)**