**GROOVY CASE STUDIES WITH SOLUTIONS**

**1. Automating CI/CD Pipeline with Groovy (Jenkins Pipeline Scripting)**

**Industry:** Software Development & DevOps

**Case Study:**

A software development team wanted to automate their CI/CD pipeline to ensure efficient builds, testing, and deployment of microservices.

**Solution:**

* Used **Jenkins Groovy Pipelines** to define build, test, and deployment stages.
* Integrated with **Git**, **Docker**, and **Kubernetes** for automated deployments.
* Implemented a **post-build email notification** system.

**Groovy Implementation:**

groovy

CopyEdit

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/example/repo.git'

}

}

stage('Build') {

steps {

sh 'mvn clean package'

}

}

stage('Test') {

steps {

sh 'mvn test'

}

}

stage('Deploy') {

steps {

sh 'kubectl apply -f deployment.yaml'

}

}

}

post {

success {

mail to: 'team@example.com', subject: 'Build Success', body: 'Deployment successful!'

}

}

}

**Impact:**

✅ Reduced deployment time by **40%**  
✅ Improved code quality with automated testing  
✅ Minimized human errors

**2. Automated REST API Testing with Groovy & Rest Assured**

**Industry:** Software Quality Assurance

**Case Study:**

A QA team needed a reliable automated testing framework to validate REST APIs for an e-commerce platform.

**Solution:**

* Developed an API test suite using **Groovy + Rest Assured**.
* Automated **GET, POST, PUT, DELETE** request validations.
* Integrated with **JUnit** for assertion testing.

**Groovy Implementation:**

groovy

CopyEdit

import io.restassured.RestAssured

import static io.restassured.RestAssured.\*

import static org.hamcrest.Matchers.\*

RestAssured.baseURI = "https://api.example.com"

given()

.header("Content-Type", "application/json")

.when()

.get("/users/1")

.then()

.statusCode(200)

.body("name", equalTo("John Doe"))

**Impact:**

✅ Reduced manual API testing time by **60%**  
✅ Improved API stability with automated regression tests  
✅ Integrated with Jenkins for **continuous testing**

**3. Groovy DSL for Configuration Management**

**Industry:** DevOps & Infrastructure

**Case Study:**

A cloud engineering team needed a flexible configuration management system to manage infrastructure settings.

**Solution:**

* Developed a **Groovy DSL** to define infrastructure configurations.
* Allowed users to write settings in Groovy instead of JSON/YAML.
* Integrated with **Terraform** for cloud infrastructure provisioning.

**Groovy Implementation:**

groovy

CopyEdit

config {

environment "production"

database {

host "db.example.com"

user "admin"

password "securePassword"

}

server {

memory "8GB"

cpu "4"

}

}

**Impact:**

✅ Simplified **infrastructure management**  
✅ Improved **readability and maintainability**  
✅ Enabled **self-service configuration updates**

**4. Log File Analysis & Monitoring with Groovy**

**Industry:** IT Operations

**Case Study:**

An IT team needed an efficient way to analyze **server logs** and detect errors in real time.

**Solution:**

* Developed a **Groovy script** to scan log files and extract error messages.
* Sent alerts via **email/SMS** for critical errors.
* Generated **daily reports** on log statistics.

**Groovy Implementation:**

groovy

CopyEdit

def logFile = new File("/var/log/app.log")

logFile.eachLine { line ->

if (line.contains("ERROR")) {

println "Alert: $line"

}

}

**Impact:**

✅ Improved **incident response time**  
✅ Automated **log monitoring**  
✅ Reduced manual **log inspection** effort by 70%

**5. Web Scraping & Data Extraction with Groovy & Jsoup**

**Industry:** Data Analytics

**Case Study:**

A marketing team wanted to scrape **competitor product prices** from e-commerce websites.

**Solution:**

* Used **Groovy + Jsoup** to extract pricing data.
* Scheduled the script using **cron jobs** for daily updates.
* Exported data into a **CSV file** for analysis.

**Groovy Implementation:**

groovy

CopyEdit

@Grab('org.jsoup:jsoup:1.13.1')

import org.jsoup.Jsoup

def url = "https://example.com/products"

def doc = Jsoup.connect(url).get()

doc.select(".product").each { product ->

def name = product.select(".product-name").text()

def price = product.select(".price").text()

println "$name - $price"

}

**Impact:**

✅ Provided **real-time competitor insights**  
✅ Automated **daily price tracking**  
✅ Helped marketing team optimize **pricing strategy**