**Project:** Playwright-DevOps-Automation

**Description:** Test automation framework with Playwright (Java), integrated with Jenkins pipelines, Dockerized execution, and infra management using Terraform + Ansible.

Documents => Automation => AutomationProject

**Git bash**  
echo "# Playwright-DevOps-Automation" >> README.md  
git init  
git add README.md  
git commit -m "First commit"  
git branch  
O/p: \* master  
git remote -v  
O/p: origin https://github.com/Abhishek-Rangaswamy/Playwright-DevOps-Automation.git (fetch)  
origin https://github.com/Abhishek-Rangaswamy/Playwright-DevOps-Automation.git (push)  
git push -u origin master

======================================================================

**Setting up Jenkins in ec2 instance**

sudo yum update -y => updates all installed packages to latest versions

sudo wget -O /etc/yum.repos.d/jenkins.repo <https://pkg.jenkins.io/redhat-stable/jenkins.repo>   
Here, **wget** is a utility to download files from the web.  
**-O** Writes the output to the specified file **/etc/yum.repos.d/jenkins.repo**.

sudo rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key>  
**rpm --import**: Adds a **GPG key** to your system.  
The GPG key is used to verify that the software packages from the Jenkins repository **have not been tampered with**.

sudo yum upgrade  
Ensures all packages (including newly available ones from Jenkins repo) are up-to-date.

sudo yum install -y java-17-amazon-corretto

sudo yum install jenkins -y  
Installs Jenkins from the repo you added earlier.

sudo systemctl enable jenkins  
Configures the Jenkins service to **automatically start** every time the system reboots.

sudo systemctl start jenkins  
Starts the Jenkins service **immediately**.

sudo systemctl status jenkins  
Displays whether Jenkins is **active/running**, **enabled**, and its logs.

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

======================================================================

[ec2-user@ip-172-31-30-217 ~]$ aws configure

AWS Access Key ID [None]: generate in security credentials

AWS Secret Access Key [None]: generate in security credentials

Default region name [None]: us-east-1

Default output format [None]: json

======================================================================

**Amazon Linux’s default yum repos don’t include Terraform**. You need to install it from **HashiCorp’s official repository**.

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo <https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo>

sudo yum install -y terraform-1.6.6

terraform -version

**Steps to Enable Terraform Autocomplete**

1. **Enable Terraform autocomplete**

terraform -install-autocomplete

This will configure shell integration for your user.

1. **Reload your shell config**

If you’re using bash (default for Amazon Linux), run:

source ~/.bashrc

======================================================================

**Allocating the permanent IP for an ec2 instance**

**🔹 Step 1: Allocate an Elastic IP**

1. Log in to **AWS Management Console**.
2. Go to **EC2 service**.
3. On the left menu, scroll down and click **Elastic IPs**.
4. Click **Allocate Elastic IP address**.
   * Leave settings as default (Amazon’s pool of IPv4).
   * Click **Allocate**.
5. Now you’ll see a new Elastic IP in the list.

**🔹 Step 2: Associate Elastic IP with your EC2**

1. Select the newly created Elastic IP.
2. Click **Actions → Associate Elastic IP address**.
3. In the pop-up:
   * **Resource type**: Instance
   * **Instance**: select your Jenkins EC2 instance
   * **Private IP**: choose the default one (usually ends with .xx).
4. Click **Associate**.

✅ Now your EC2 has a **static public IP**.

**🔹 Step 3: Update Security Group**

1. Go to **Security Groups** for your Jenkins EC2 instance.
2. Edit inbound rules → make sure **TCP port 8080** is allowed from:
   * 0.0.0.0/0 (open to all, but less secure)
   * OR restrict to **GitHub/GitLab Webhook IP ranges** for security.

======================================================================

**Configure GitHub webhooks**

**Step 1: Create a New Jenkins Job**

1. Log in to Jenkins.
2. Click **“New Item”**.
3. Enter a job name (e.g., Playwright-Automation-Build).
4. Select **“Freestyle project”** (or **Pipeline** if you want scripted).
5. Click **OK**.

**Step 2: Configure Source Code Management**

1. Under **Source Code Management**, select **Git**.
2. Provide your repo URL (HTTPS or SSH).  
   Example:
3. https://github.com/your-org/your-repo.git
4. Add credentials if needed.
5. Choose the branch (main, master, or develop).

**Step 3: Build Triggers (Webhook Integration)**

1. Go to **Build Triggers**.
2. Tick **“GitHub hook trigger for GITScm polling”** (if GitHub).
   * For GitLab: use **“Build when a change is pushed to GitLab”**.
3. Save the job.

**Step 4: Configure GitHub/GitLab Webhook**

1. Go to your repo → **Settings → Webhooks**.
2. Add a new webhook:
   * **Payload URL**:
   * http://<your-ec2-public-ip>:8080/github-webhook/
   * Content type: application/json.
   * Events: “Just the push event”.
3. Save webhook.

⚠ Make sure port **8080** is open in your EC2 Security Group.

**Step 5: Add Build Step to Compile Code**

1. In the job config, scroll to **Build → Add build step**.
   * If using **Maven**:
   * mvn clean install
   * If using **Gradle**:
   * ./gradlew build

**Step 6: Save and Test**

* Push code to your GitHub/GitLab repo.
* The webhook will notify Jenkins.
* Jenkins will pull code → compile with Maven → mark job success/failure.

✅ That’s the **Freestyle way**. If you prefer **Pipeline**, you’d use a Jenkins file with a pipeline {} block (Git clone + Maven build).