**DevOps on AWS, learning path**

Essentials

Regions and availability zones

VPC

EC2

S3

IAM

CI/CD

CodeCommit - Source control repo, similar to Git

CodeBuild - Build source code, runs specified tests

CodeDeploy - Deploy the built software apps on a variety of AWS Compute Infrastructure

CodePipeline - Automatic detection of changes in source code, auto build, auto deployment

Infrastructure

ECS or EKS

Lambda

Cloudformation - Handles creation of resources. Basically creates infrastructure using code (using JSON or YAML)

Cloud Development Kit (CDK) - Write actual code to create your infrastructure

Terraform

Beanstalk

Security

IAM Policies

VPC Security Groups

CloudTrail - For monitoring and auditing

Monitoring

CloudWatch (Metrics, Alarms, Logs, Dashboards)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**AWS CodePipeline and Elastic Beanstalk hands-on**

Objective: Build a CI/CD Pipeline for the deployment of a node/react project

Prerequisites: Have a Node/React app ready for deployment in a GitHub repo

Sample app: <https://github.com/arindamkeswani/foodApp_backend>

You can fork it to your repo

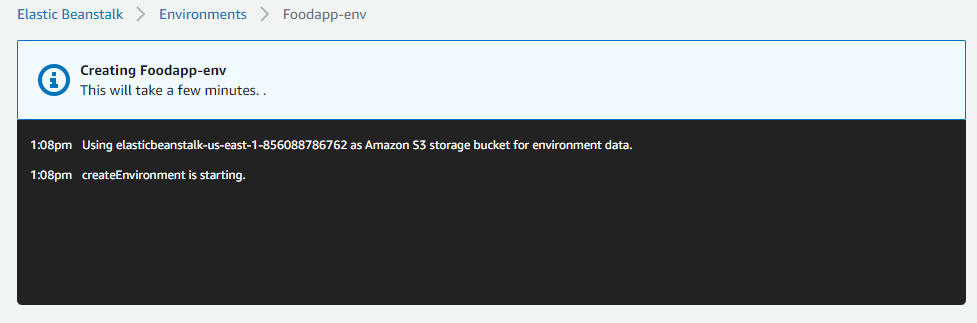
AWS CodePipeline is **a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates**.

What is AWS Elastic Beanstalk used for?

AWS Elastic Beanstalk is an easy-to-use service for **deploying and scaling web applications and services**

* Login to AWS Management Console
* Go to the console for Elastic Beanstalk
  + We are using Beanstalk for deploying resources to EC2 instances
* Click on Create Application
* Give you app a name
* Platform: Node.js
* Create Application

You should be seeing a screen like this one:



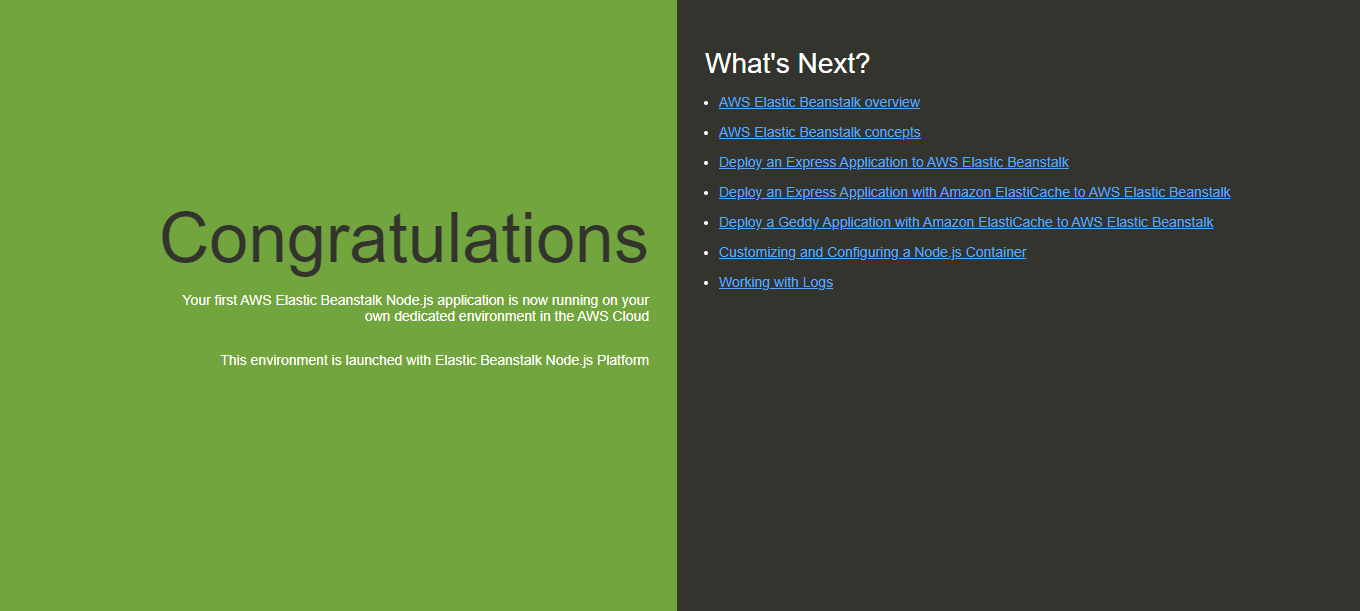
So far, we have created a Node.js environment. Elastic Beanstalk has deployed a sample application into that environment

We are using resources like EC2, S3. You can confirm this in the logs after the above process is complete.

We do not need to install the dependencies of project explicitly, if Beanstalk finds a package.json file



“Go to environment” will take you to the sample application which will look something like this:

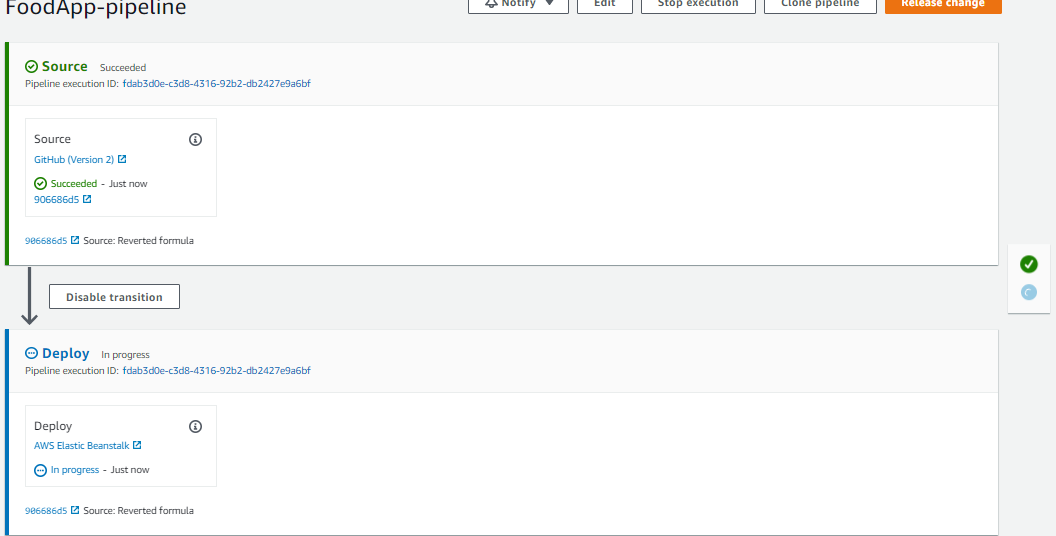


This means that our initial deployment was successful.

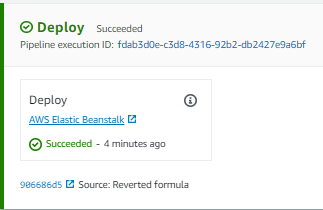
Next step, set-up a pipeline using CodePipeline. For this, search and goto the CodePipeline service

* Create Pipeline
* Name it
* Read the other options, leave the other options as default for now, click Next
* Add GitHub in Source Stage (preferably version 2)
* Connect to GitHub, enter your password
* Choose repository (and branch, if needed. Otherwise choose “master”)
* CodePipeline default
* Next
* Skip build stage
* In “Add deploy stage” choose Elastic Beanstalk
* Choose the relevant app name and environment name
* Next
* Create Pipeline

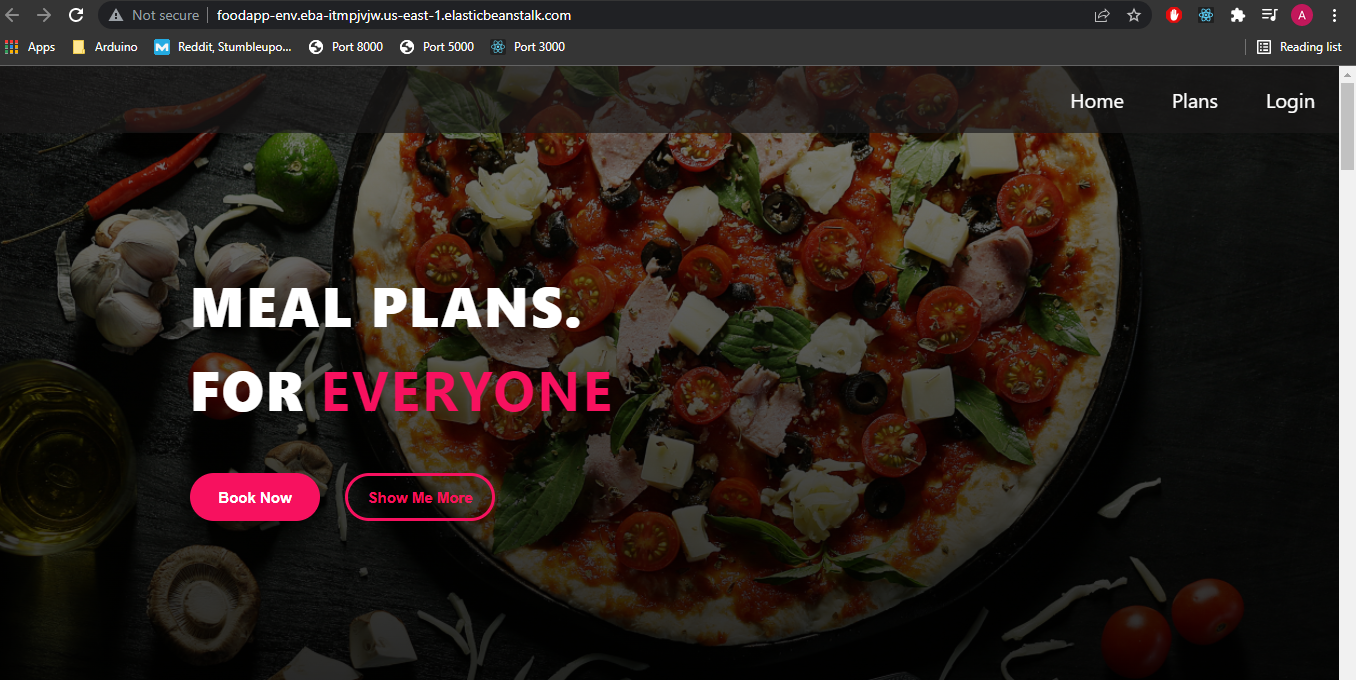
You should be seeing something like this



After the deployment has succeeded, click on Elastic Beanstalk

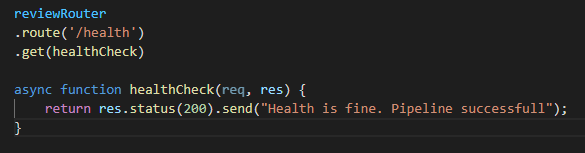


Try to visit your application in the same way as before, you should see your application on the same URL



* Now, we will make a small change, commit the changes and see if our application still works.

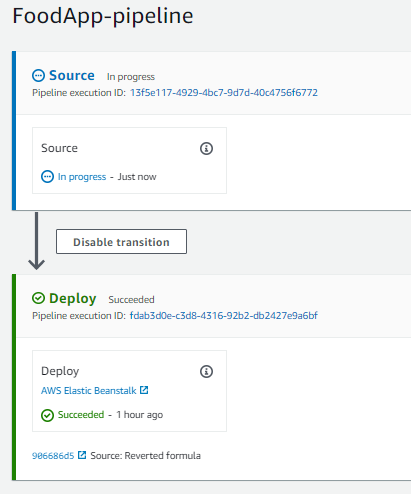
E.g.



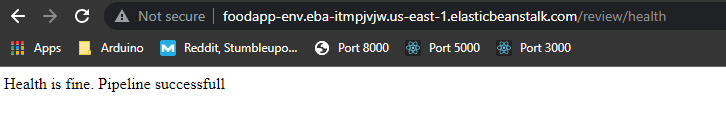
Added the above snippet in reviewRouter.js.

After making the changes, it is recommended that you check the working of the application locally before committing and pushing the code.

After you do that the pipeline should start re-deploying automatically.



Check the results by entering the correct route and seeing if the changes are made properly. Here the route was ‘/review/health’



Make sure you clean up the resources at the end of the experiment.

S3 Buckets will persist even after removing the Pipeline and the Beanstalk Environment. Make sure you empty them out before deleting the buckets

Note: You will need to change permissions for deletion of the elastic beanstalk bucket.

For this, click on the bucket, Click on permissions.

You will be a JSON document, click on Edit. Go to the section that says s3:DeleteBucket.

The action should be denied. Replace “Deny” with “Allow” and save the changes.

**Chef**

Q. What is chef?

A:

* Open source configuration management tool
* Falls under operation and deployment in the DevOps lifecycle
* Written in Ruby and Erlang

Q. Why use chef?

* In an organisation, the infrastructure needs to be configured and maintained from time to time
  + Companies use chef to automate this process
* Chef provides continuous deployment
* Increases system robustness by automatically ensuring that all bugs are caught and removed before deployment
* Easy integration with cloud platforms

Q. What is configuration management?

* Collection of engineering practices that provides a systematic way to manage code and infrastructure so that system admins know about the necessary changes that have been made by each other, and where in the codebase have they been made
* Types of configuration management
  + Push configuration: Server that holds the instruction files to configure the nodes pushes the files on said nodes. Server is in control
  + Pull configuration: Nodes check the server for changes periodically and fetch configuration from it. **This is what chef follows.**

Q. What is Infrastructure as Code?

* E.g. Installing some software (say MongoDB) on 50 machines in your company manually will take a lot of time. One can simplify by writing code to do this process
* Suppose the company changes their mind, and they want to install MySQL instead, it will be much easier to just modify the code to accommodate for MySQL instead of MongoDB in place of doing the task manually
* As compared to commands, code can be tested to catch bugs and can be deployed
* Therefore, **Infrastructure as Code** (**IaC**) is the managing and provisioning of infrastructure through code instead of through manual processes.

Q. What are the various components of chef?

* Workstation
  + The system where admin sits
  + Code for node configuration is created here
  + This code is called recipe
  + Collection of recipes form a cookbook
  + Knife is the CLI tool that uploads a cookbook to the server
* Server
  + Middleman between the workstation and the nodes
  + Cookbooks are stored here
  + Provides tools for node configuration
  + Can be hosted locally or remotely
* Node
  + They are systems that require configuration
  + In a chef architecture, we can have multiple nodes
  + Ohai is a service, installed on your node, that fetches the location and the current state of the node. This information is sent to the server to check for changes
  + Chef client, another service on the node, is responsible for communication with the server
  + Each node can have a different configuration environment
* Flavours of Chef (**to be confirmed whether there is something actually called Flavours or not**)
  + Chef Solo
    - No remote server, cookbooks are located on the site itself
    - Used for single node systems
* Hosted Chef
  + Chef server is provided as a service on the cloud
* Chef client/server
  + traditional Chef architecture. a hosted remote server communicates between the workstation and node
* Private Chef
  + enterprise version of Chef. The server is hosted within the enterprise infrastructure

**Terraform**

Q: What is terraform?

A: It allows you to automate and manage the infrastructure, and your platform, and the services that run on that platform.

It is open source, and uses declarative language, which means that we don’t have to define every step of the automation. We just need to specify what the end result should be.

E.g. Using Google Maps to drive to your destination vs using Uber.

You will have to follow a set of instructions to get from point A to B (imperative) in the former.

In the latter, you just need to specify points A and B, and the driver (metaphor for Terraform) will take care of the rest.

It is a tool for infrastructure provisioning. Say we want to plan out the infrastructure for our project, which will involve various servers, and 5 microservices as Docker containers along with a database container, and we want to use AWS for this.

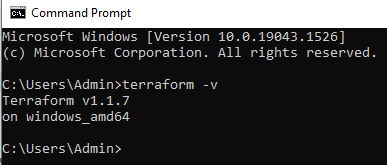
In a brute force way, we would have to first prepare our private network space, ec2 server instances, install Docker and other dependencies, configure security etc and THEN secondly we can deploy the application. That is 2 steps.

The first step is typically performed by the DevOps team and this is the step that Terraform is involved in, and deployment is handled by the developer.

* There are 3 major phases in terraform
  + Coding phase - Create (i.e. write code for) Terraform file: Here, we configure resources
  + Plan phase - Compares the desired state with our required resources, to what we have in our current state. Basically it will ask us to create the resources that we need
  + Apply phase - Actually take the resources by using APIs and spin up the resources, and give output variables in return, for example a Kubernetes dashboard link, our application URL etc.
* In the above manner, we end up creating something called a Terraform module (a way to group together some kind of terraform automation)
* As part of a module, we generally define a Cloud provider, who will provide the necessary infrastructure (IaaS) or help us manage platforms (PaaS, such as Cloud Foundry) or even software (SaaS, eg. cloudfare) for us

**Hands-on**:

Set-up (Windows):

* <https://www.terraform.io/downloads>
* Download the windows file for your system (32 or 64 bit)
* Extract the zip folder, you will see an executable file
* 
* Copy it, paste it in a folder of your choice, and note down the path of this folder
* Add this path to the environment variable
* Confirm if things have been done properly by opening the command prompt and writing “terraform -v”
* You should see the version
* 
* Install the terraform extension on VS Code

Set-up complete

* Terraform is written in HashiCorp configuration language (“.tf” extension)
* Follow the steps in the link: <https://github.com/arindamkeswani/aws/tree/main/terraform/project1>

**Nagios**

Q: What is nagios?

A: After deploying a project, there is a need to constantly monitor it constantly to check for server health, hackers, code-related issues etc.

* Can runs plugins stored on the server or the internet
* Alerts are sent in case of issues

So nagios is a Open source tool for systems, networks, infrastructure

Some popular tools in the market are splunk, prometheus, ELK, AWS Cloudwatch, Sensu, Librato

Q: So why nagios?

A: Been in the market for a long time, so a lot of documentation, plug-ins are available, and one can even write their own plug-ins

Q: What is Continuous Monitoring and why do we need it?

A: Similar to a CI/CD Pipeline, the idea is to automate the process of monitoring using a tool like Nagios

Although we expect our applications to be up all the time, we want to be cautious in case things go wrong, and that is why we need it, for cases such as failure in the CI/CD pipeline, application failure, problem in the infrastructure, or code analysis failure.

So we need it to avoid downtime, or reduce it as much as possible by taking action in time

Features:

* Nagios XI is the free version (to a limited extent)
* Based on client-server architecture
* Made by Ethan Galstad in 1999
* Uses port number 5666,5667,5668 to monitor the cloud
* Detects Network and Server issues accurately in the CI/CD pipeline
* Can automatically troubleshoot
* Active monitoring of infrastructure, which means that is requests are constantly being sent from Nagios to the client to check for health, as compared to passive monitoring where client reaches out to the monitoring tool to give updates (both types happen in Nagios)
* Has a good log and database system
* Informative web interface
* Auto-send alerts if condition changes
* Monitor entire business process and infrastructure with a single pass
* Monitor network services like http, smtp, snmp, ftp, ssh, pop, dns, ldap, ipmi, etc. (\*)

Phases of continuous monitoring

1. Define a monitoring strategy
2. Establish monitoring frequency (such as time-based or change-based)
3. Implement strategy
4. Analyse data given by the tool as a report
5. Respond accordingly
6. Renew strategy and update

Nagios architecture

* Uses client-server architecture
* Basic idea wrt AWS: Launch an EC2 instance and install Nagios on it
* Nagios, which is installed on the server, monitors the client/node
* Nagios has configuration files, which store details like the node’s IP Address, username, password
  + We monitor things like CPU Load, traffic (incoming and outgoing), number of requests handled successfully/ unsuccessfully by accessing different ports as mentioned (\*), the details of which are specified in the config files
* Next, Nagios has a daemon, that collects info from config files and sees what work needs to be done. The daemon does this with the help of the **NRPE**(Nagios remote plug-in executor) plug-in
* The daemon will invoke the NRPE plug-in, which will use “**Check by SSH**” plug-in (also called the **NRPE agen**t) that is installed on the node (as all connections between the server and nodes are SSH Connections)
* NRPE will ask the agent for information about various ports, who will return the required info. NRPE will then return the info to the daemon
* Daemon has a DB of its own where it stores all this info, from where it takes the information to a dashboard on a web page on the internet
* Say your server’s IP is 12.1.1.1, then go to your browser and enter 12.1.1.1/nagios to open the dashboard, which will be visible to you after entering your credentials
* So Nagios server runs on the host (eg EC2 instance), which monitors the remote hosts (clients)

Prerequisites

* Httpd (Browser)
* Php (dashboard)
* Gcc & gd (compiler to connect raw code into binary)
* Makefile (to build)
* Perl (scripting)
* Main configuration file -> /usr/local/nagios/etc/nagios.cfg

Every thing that we monitor can be referred to as a service, for example if we are monitoring https and ftp on 4 clients, which means that we are monitoring 8 services

Dashboard displays the host status as down, unreachable, up, recovery, none (visible as localhost)

Services have status as warning, unknown, critical, pending, etc.

Alerts are sent accordingly

**Hands-on:**

**Objective: How to add configure nagios on an EC2 instance**

Step 1:

Prerequisites:

* Have an EC2 instance up and running. It should have SSH and HTTP access configured in the security group (alternatively, you can allow All Traffic). Make sure that you enable Public IP while creating your instance
* Open puTTYgen. Use your existing key-pair and convert it into .ppk file
* Open putty and enter “ec2-user” when prompted “login as:”
* Install apache, php, gcc compiler, gd development libraries (we can do this via puTTY)
  + For this, the commands to be run are:
    - sudo su (to work as the admin)
    - yum install httpd php
    - yum install gcc glibc glibc-common
    - yum install gd gd-devel

Step 2: Create acc info needed to set up a nagios userC

* Commands:
  + adduser -m nagios
  + passwd nagios (Enter the password when prompted)
  + Groupadd nagioscmd
  + usermod -a -G nagioscmd nagios
  + usermod -a -G nagioscmd apache
  + usermod -a -G nagios apache

Step 3: Download Nagios Core and plug-ins. Create a directory for storing downloads.

* Commands
  + mkdir ~/downloads
  + cd ~/downloads

Step 4: Download source code tarballs for nagios and nagios plug-ins

* wget [https://sourceforge.net/projects/nagios/files/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz](https://sourceforge.net/projects/nagios/files/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz/download?use_mirror=udomain)
* (OR wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.5.tar.gz)
* wget <http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz>

(OR wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz)

Step 5: Compile and install nagios by extracting the source code tarballs

* Commands:
  + tar zxvf nagios-4.0.8.tar.gz (extract, verbose, forced)

(OR tar -zxvf /tmp/nagios-4.4.5.tar.gz)

* + cd nagios-4.0.8 (OR cd nagios-4.4.5)
  + ./configure --with-command-group=nagioscmd
  + make all (this will compile the nagios source code)

Step 6:

* Install binaries, init script, sample config file, and set permissions on external command directly
* Commands:
  + Make install (this will install nagios)
  + Make install-init (install/compile initialization script)
  + Make install-config (install config script)
  + Make install-commandmode

Step 7: Configure the web interface

* make install-webconf

Step 8: Create ‘nagiosadmin’ account to login into nagios web interface. Set password as well.

* htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

Enter a password when prompted

* service httpd restart

Step 9: Before compiling and install the nagios plug-ins, extract the nagios plug-ins

* cd ~/downloads
* tar zxvf nagios-plugins-2.0.3.tar.gz

(OR tar -zxvf /tmp/nagios-plugins-2.3.3.tar.gz)

* cd nagios-plugins-2.0.3 (OR cd nagios-plugins-2.3.3)

Step 10: Compile and install the plug-ins

* ./configure --with-nagios-user=nagios --with-nagios-group=nagios
* make
* make install

Step 11: Start nagios. Add nagios to list of system services and have it automatically start when the systems boots

* chkconfig --add nagios (There might be issues here)
* chkconfig nagios on

Step 12: Verify sample nagios config files

* /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (the 2 files will be verified. If the two files match, it will tell you that there is no error)

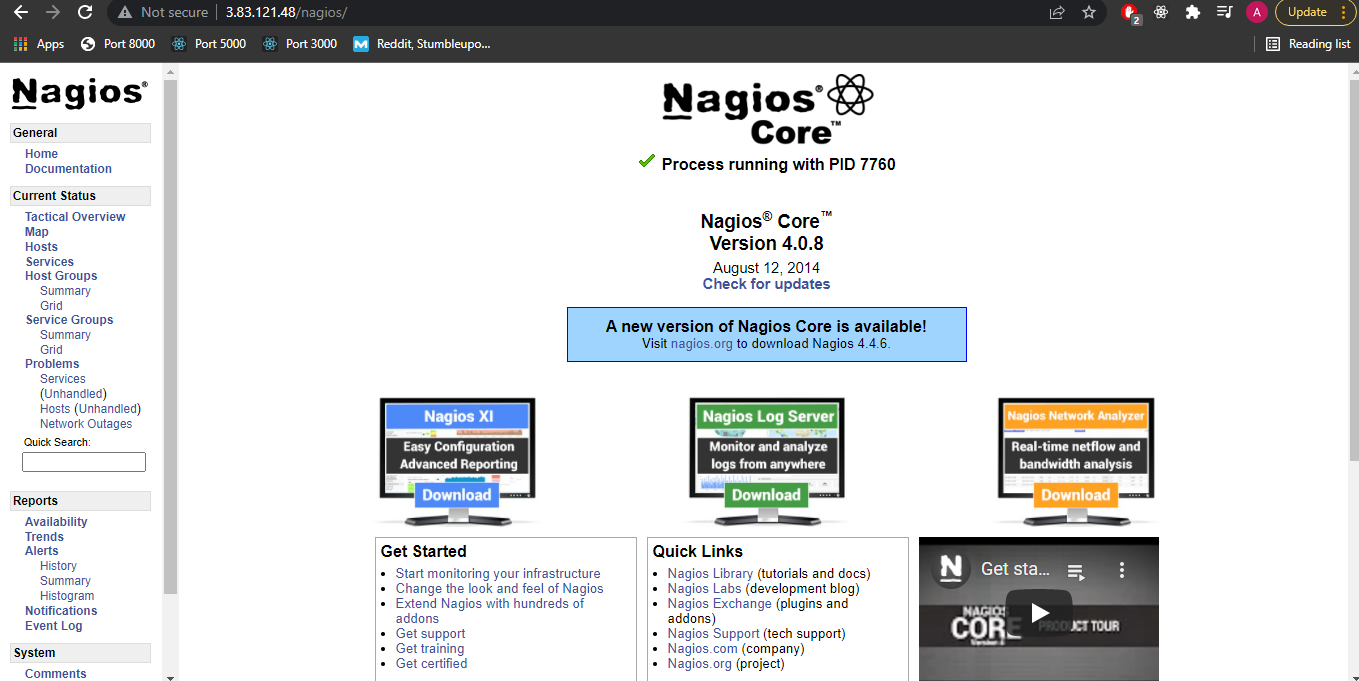
Step 13: If there are no errors, start nagios

* service nagios start
* service httpd restart

Step 14: Copy public IP of EC2 instance and paste in browser, as follows:

* <public\_ip\_address>/nagios/
* E.g. 23.2.3.1/nagios/
* It will ask for username-> ‘nagiosadmin’ (we created this earlier)
* Enter your password

On entering the credentials, you should see something like this



Click on various options and explore:

1. Map
2. Hosts
   1. It should show the status as UP for localhost.
3. Alerts

Alternate installation video: <https://www.youtube.com/watch?v=veSHsK1EU9w&ab_channel=ITJobHacks>

Potential Nagios hands-on video: https://www.youtube.com/watch?v=aaDiOH-q5\_E&ab\_channel=TrieTreeTechnologies

Jenkins

Q: What is CI/CD Pipeline?

A:

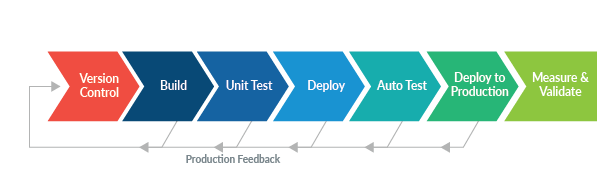
CI: Continuous Integration

CD: Continuous deployment/delivery

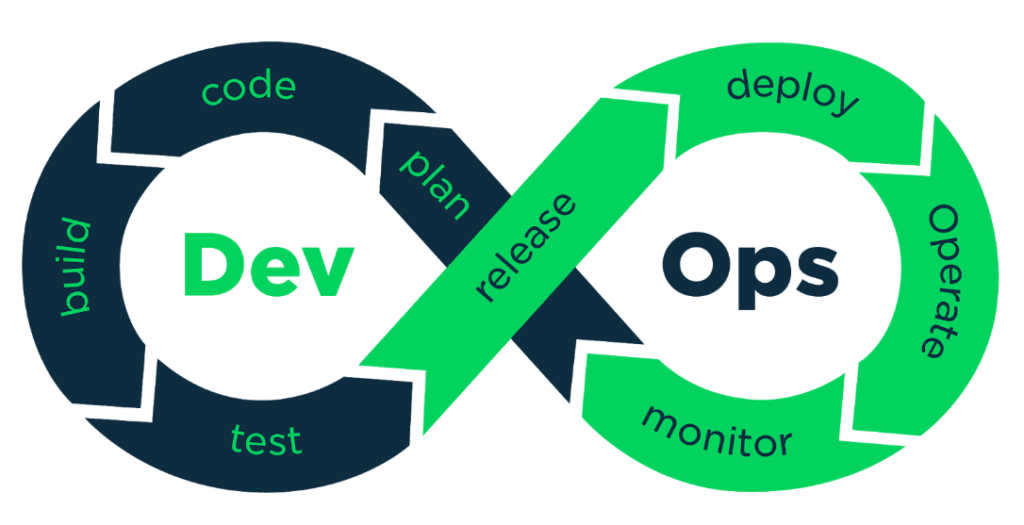
Pipeline: List of steps/processes to be followed in a specified order, typically in a FCFS manner.

So CI/CD is a methodology used to automate certain processes in the SDLC

* Before CI
  + Developers write code
  + Code was pushed to VCS like GitHub
  + From there code was sent for integration, (where we combine and execute the code of all developers together)
  + Testing phase. All bugs are sent back to developers
  + Codebases are usually quite large so it is quite tedious to review the errors, especially when the developer has been re-assigned to another project during this time. This will take a lot of time.
* To save on this time, CI methodology came into place
* After CI
  + Developers write code
  + Code was pushed to VCS like GitHub, but this time code is not pushed after completion. It is pushed after every little update.
  + As soon as code is pushed, it gets pulled by the CI server, which has 3 components
    - Build - Check for syntax errors, bugs, etc.
    - Test - Check correctness of output
    - Deploy - Deploy in the test environment
  + CI Server reverts to the developer notifying about failure or success very quickly, so it is easy for developers to track the errors because the pushed code is small and the developer is still familiar with it.
  + Code is re-pushed and the cycle repeats again.
  + Therefore continuous integration = continuous build + continuous testing
* Therefore, a CI/CD Pipeline consists of
  + Version control
  + Build
  + Unit test
  + Deploy
  + Auto test
  + Production environment deployment
  + Measure and validate (Quality assurance)
  + Production feedback, which goes back to all the other steps in the pipeline with necessary changes



DevOps cycle:

* 
* Dev and Ops parts have a lot of elements
  + One way is to send code from each step manually
  + Another way is to use a tool which keeps pushing the code forward in the cycle as soon as the current step is complete. Such a tool is known as an integration tool
  + Jenkins is the integration tool that we will be using to automate the process. Other famous tools are Bamboo, Travis CI, Buildbot.
  + Jenkins is most popular because it is FOSS
  + So now, Jenkins will pull the code from GitHub and handle the rest. So, the process after pushing the code is automated. Eg:
    - Dev pushes code to GitHub
    - Jenkins picks the code up and gives it to Maven to build it
    - Jenkins will take the code to testing to Selenuim/Junit, etc.
    - Code is taken by Jenkins to Chef for deployment
    - Jenkins then initiates the monitoring process
    - This is why Jenkins is known as the heart of the DevOps process
  + It is not necessary to automate all the steps of the process

Jenkins: In-depth

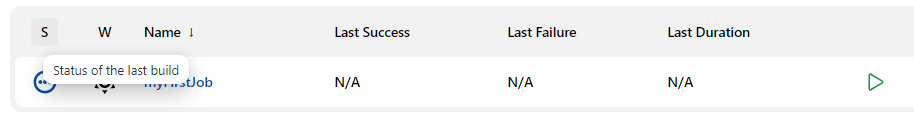
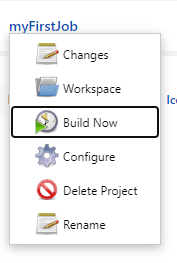
* Open source project (therefore, freely available), written in Java, so Java is necessary to use Jenkins on your device.
* Works on port 8080
* Large community and a lot of plug-ins support the tool/framework
* Used for CI
* Jenkins automates the entire SDLC
* Originally developed by Sun Microsystems (2004), under the name “Hudson”. Oracle overtook Sun and separated Hudson (enterprise edition) from Jenkins (free edition)
* Can run on any major platform without compatibility issues
* Jenkins takes code from developers, integrates it, send for build, then test, then deliver/deploy to client
* CI enables us to report bugs faster and get rectified, so in turn the software development process happens faster
* **(\*)**E.g. flow: Developer -> GitHub (Version control) -> Jenkins pulls the code -> Maven (Build/compile) -> Jenkins pulls the code -> Selenium (Testing) -> jenkins pulls the code -> Checkstyle (QA) -> Jenkins -> Deploy/Deliver OR Send to Artifactory (archiving)
* Plugins such as git, maven, selenium, artifactory (used for archiving ready code) can be attached to jenkins. Plug-ins help jenkins communicate with the tools in the DevOps cycle
* In all the steps mentioned in **(\*),** it is necessary to attach the required plug-in for each step to function as desired
* Build process has multiple steps
  + Compile
  + Code review
  + Unit Testing
  + Integration Testing
  + Packaging (WAR, JAR)

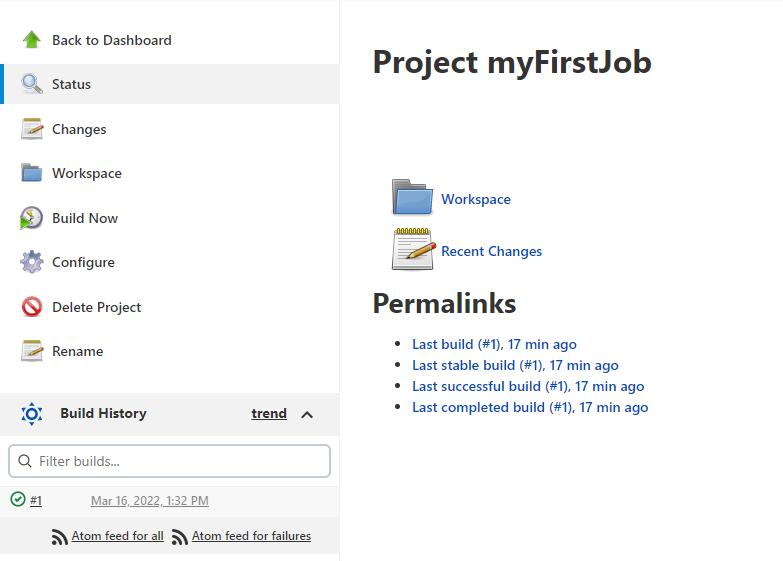
Jenkins pros:

* Lots of plug-ins
* You can write your own plug-in (because it is OSS)
* Community support
* Jenkins is not just a tool, it is a framework, as compared to Chef etc. where commands are fixed. We can modify jenkins acc to us. All we need is plug-ins
* Not necessary to automate all steps in the cycle, e.g. you want to handle testing manually, so you can not automate it
* We can attach slaves to master
* Jenkins follows Master-Slave architecture. Basically, Master is responsible for assigning and distributing the load among the slaves. If there is no Slave, Master will handle the process
* Jenkins can behave as Cron server, used for scheduling tasks
* Jenkins can create labels (to slaves and tasks), used for assignment of tasks to slaves
* GUI-based

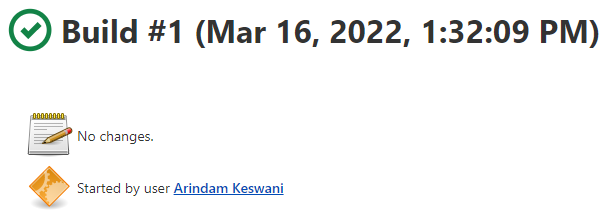
Jenkins Hands-on

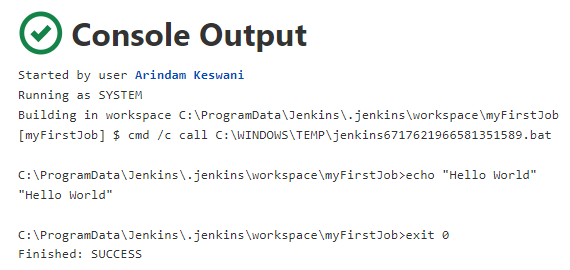
Steps

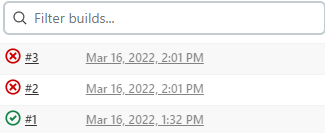
* Install [Git](https://git-scm.com/downloads)
* Download, install, configure [java](https://www.oracle.com/java/technologies/downloads/)
  + To run Jenkins, check the Java specifications. Jenkins supports JDK 11.
  + In environment variables, put make a new variable JAVA\_HOME in both system and user variables and add the path in which you have installed Java
  + Add path of the main folder to the main path
* Download and configure Maven
  + [Download Maven](https://maven.apache.org/)
  + Extract file in the folder of your choice. Go inside the maven folder and copy the path
  + Add it to the system variables under the name “M2\_HOME”
  + Go in the bin folder, copy the path and add it to the system path variables
  + Run “mvn -version” to confirm the version installed
  + Run “echo %M2\_HOME%” to check the path that we just defined
* Download, Install configure Jenkins
  + [Download](https://www.jenkins.io/download/)
  + [Installation process](https://www.jenkins.io/doc/book/installing/windows/)
  + While installing, make sure that the port that Jenkins wants to use (8080 by default) is not in use by some other application already. If it is, then change the Jenkins port number (e.g. 8081) and test it. If you see the green tick, continue with the installation.
  + You can restart your Jenkins by putting ‘/restart’ at the end of the URL like ‘<http://localhost:8080/restart>’
  + Similarly, you can stop Jenkins by putting ‘/stop’
  + Create an Item/Job
    - Click on New Item
      * Enter Item name
      * Free style project -> OK
      * Build->Add Build Step->Execute Windows batch Command
        + echo "Hello World"
      * “Apply” will save the command and you will stay on the same page
      * "Save” will save the command and take you to the dashboard
    - Back to dashboard
      * Your new job should be visible here
      * Hover over different fields to see what they mean
      * 
      * Click on Build Now
      * Click ion the job name. It should show something like:

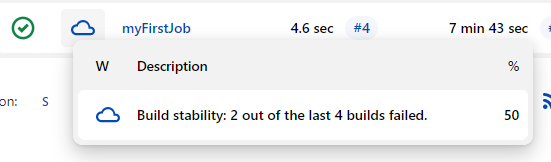


Click on the “#1” to see the details on the Build that we just initiated

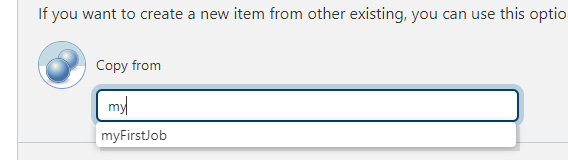
* 
* Click on console output

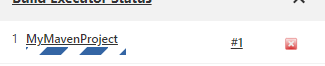


* You can copy the directory from your screen and go to that path to verify that the job has actually been created
* Try to configure the app again by using “cho” instead of “echo”, the former being invalid. Build again, this time, the build will fail (red cross #2 and #3).
* 
* Click on the Red Cross to confirm
* Cannot build more than 2 jobs together. The third job will be queued.
* Hovering over the weather icon gives information about our build

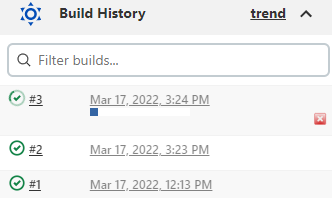


* You can even copy a job,
  + Instead of choosing “Freestyle project”, go to the following section and type the name of the job that you wish to copy



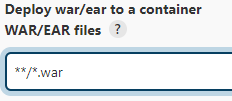
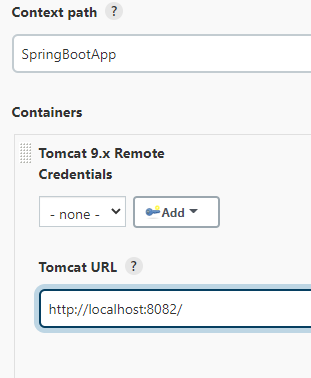
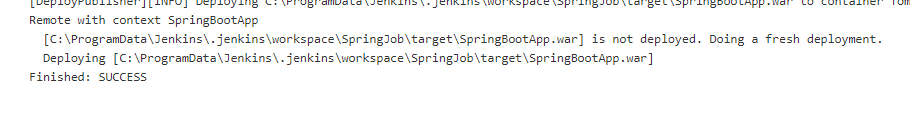
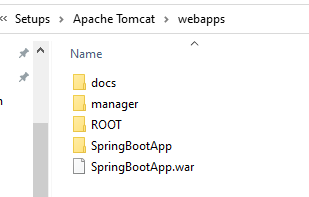
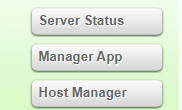
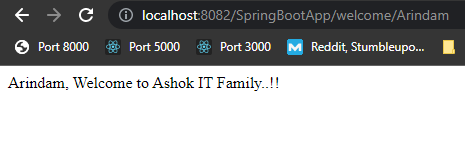
* Now, we need to connect Jenkins, Maven, Git, and Java
* Objective:
  + Build a job directly through Maven
  + Build a Maven job via Jenkins
* Steps
  + Open Jenkins
  + Manage Jenkins
  + Manage plugins
    - Click Available
    - Select Maven Integration and Green Balls (will show green balls on successful build)
    - Install without restart
  + Configuration
    - Manage Jenkins
    - Global configuration tool
    - JDK
      * Un-check install automatically option
      * Set name
      * Set Path to JDK that we installed
    - Do the same in the Maven section
    - Save (Re-directed to dashboard)
  + Now we will make a Maven project
    - New item
    - Maven Project
    - Manage Jenkins
    - Go to the [link](https://github.com/technicalguftgu/time-tracker) for a sample project. Fork it
    - Clone it in a local repository
    - cd time-tracker
    - Run ‘mvn clean package’
      * To Build. Used to convert code to machine language
  + In the above step, we built the project using Maven on the command line. We can configure the same using jenkins with the following steps
  + Now we will make a Maven project (Using Jenkins)
    - New item -> Name: MyMavenProject
    - Select Maven Project
    - Ok
    - Click on Source Code Management
    - Select Git
      * Paste Repo URL
    - Go to Build section
      * You will see a Root POM section with ‘pom.xml’
      * Below it in ‘Goals in options’, write ‘clean package’ (relate to mvn clean package)
      * Save
    - Go to the Dashboard and build your project
    - 
  + Schedule builds periodically
    - Click on any project
    - Configure
    - Build Triggers
    - Build periodically
    - Enter “\* \* \* \* \*”. See CRON syntax by clicking on the question mark



* Save
* 

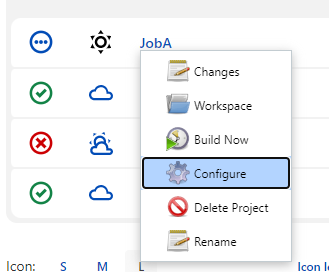
* Periodic build can be a waste of resources. A better way would be to build only when there is a change in our code
  + Source code polling
    - Click on any project
    - Configure
    - Build trigger
    - Poll SCM
    - \* \* \* \* \*
    - Save
    - Make some changes in the file (for now do it in the README file) and push the changes
    - You will notice that the code starts to build automatically in about a minute

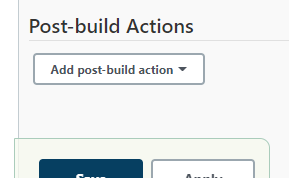
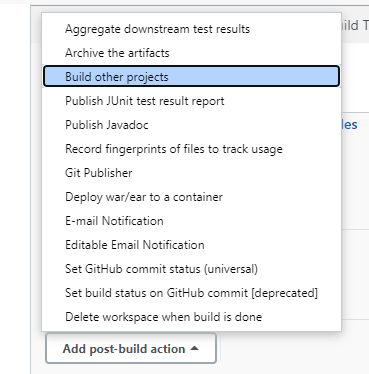
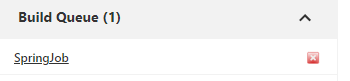
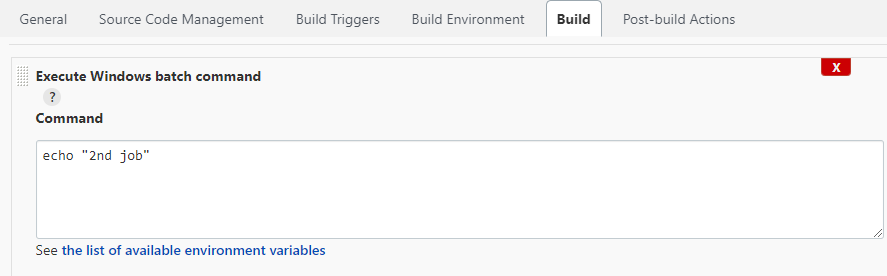
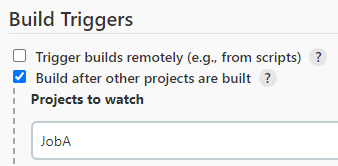
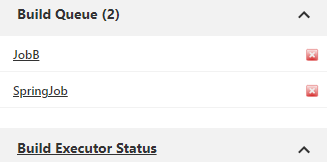


* + - Verify that it does not build on its own after that
* [Deploying Project using Tomcat server](https://www.youtube.com/watch?v=6fcoYGqvg6o&ab_channel=AshokIT)
  + “Deploy to container” plugin
  + Install apache tomcat
  + Configure your application
    - Post build actions
      * Deploy war/ear to a container
      * 
      * 
      * Apply
      * Go to dashboard-> Click on project
      * Build now
      * 
      * After build is successful, confirm that the .war file exists in the web apps folder
      * Go to Tomcat interface
      * Click on Manager App
      * Enter credentials
      * Click on your application
      * Go to a valid route
      * 

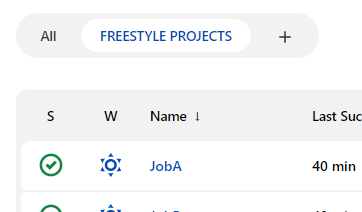
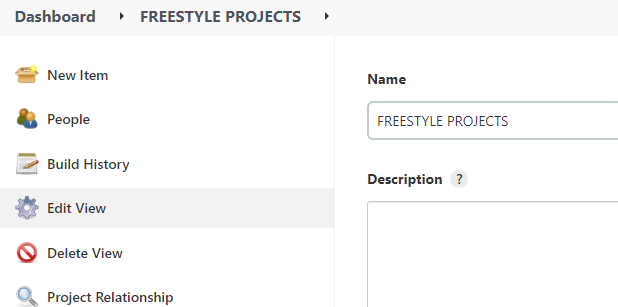
**Linked Projects in Jenkins:**

* Jobs that we create in Jenkins can be linked to each other, for example, after completion of one job, another job that has been linked to the previous one will start on its own, thus automating the process
* Two types of linked projects: Upstream and Downstream. Purpose of both types in to link projects, but the way in which they achieve that is different
* Upstream Linked Project:
  + Say we have two Jobs A and B. Job A, on finishing its task, will tell the next job, Job B, that its job is done and the latter can start its work. So, Job A is the one communicating.
  + Who is triggering Job A then? The user/administrator is. User/admin will not trigger Job B. Job A will do that after it is done with its work
    - How to do that? Mentioning the next job’s details in the Post-Build Actions in a job.
* Downstream Linked Project:
  + Opposite of what happens in Upstream
  + We will not mention instructions to start Job B inside Job A. Instead, we will tell Job B to start its task after Job A is done. So the active job is Job B
* Linked Project hands-on
  + UPSTREAM
  + Open Jenkins dashboard
  + New Item -> Give name Job A
  + Copy it from “MyFirstJob” -> Click Ok -> Save
  + Go back to dashboard, click on Job, click on Configure



* 
* Go to Post-build Actions
  + This is where we will tell the Job to notify the next Job of its completion, so that the latter can start its work. E.g. how someone’s parent cooks food (Job A), and once they're done, they ask the child to come to the table and eat (Job B)
  + 
  + Click
  + Choose a project to build. E.g. I have chosen the Spring project that we build in last lab
  + Go back to dashboard and build JobA. As soon as JobA is done building, if the build is successful, SpringJob will build right after
  + 
* DOWNSTREAM
* Create a job, “JobB” as a freestyle project. Click OK
* Configure the following:
* 
* Go back to the Dashboard. Select the job that we want to place the trigger on. Click on “Configure” as before
* Go to Build Triggers
* 
* Save
* Go to Dashboard and try building JobA. As soon as it is done building, JobB will start building on its own.
* Note: Since there was a post build action in JobA which built SpringJob, it will also be queued with JobB
* 

Views in Jenkins

* Dashboard
* Click on ‘+”
* Give a name e.g. “Freestyle projects” -> List View -> OK/Create
* Select the jobs you wish to keep in this list -> OK
* You will now see a new tab which will contain the jobs that you just selected
* 
* This is useful when you have a lot of projects and want to categorise them.
* You can do a lot of operations, for eg: if you want to add/remove more jobs you can do that using the Edit View section. This will add/remove jobs from the current View, not All Jobs section
* 

**User Management in Jenkins:**

**Q:** Why do we need User Management?

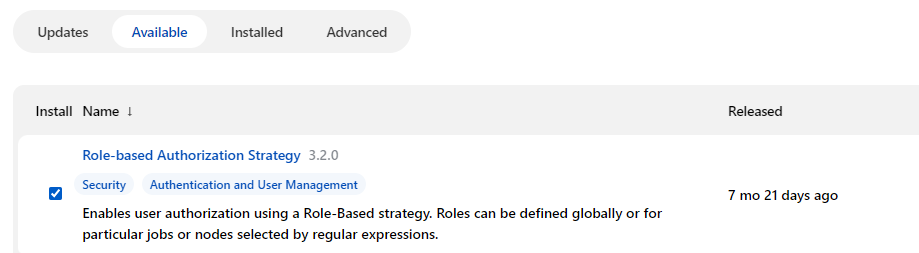
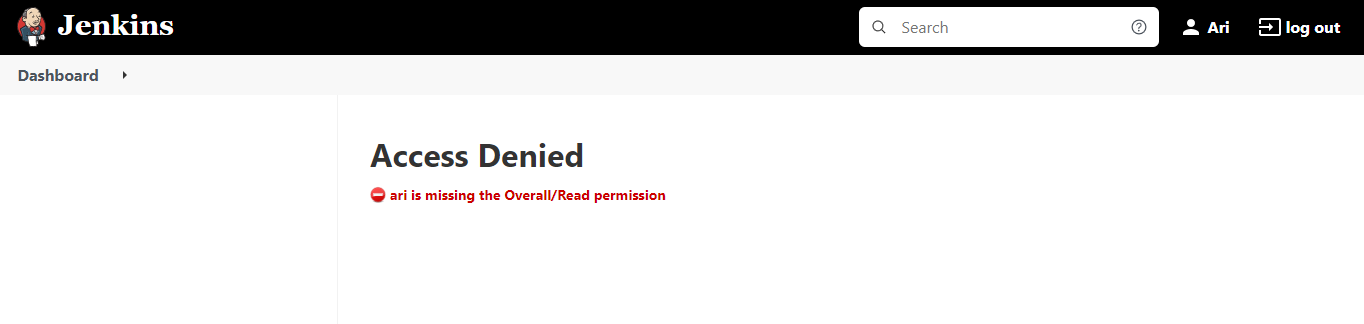
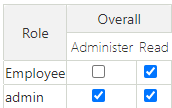
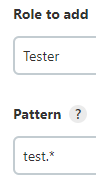
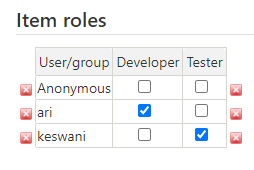
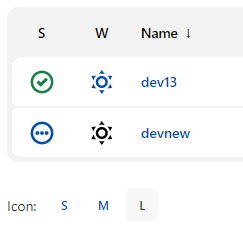
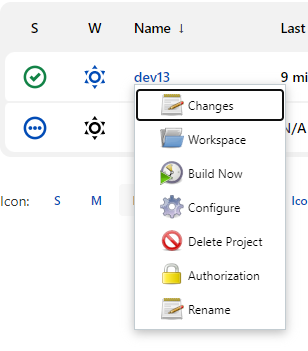
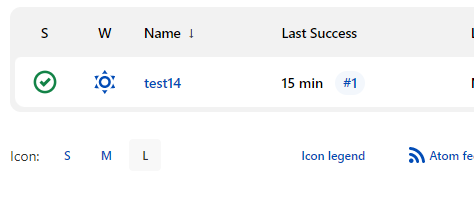
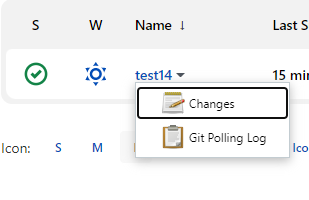
A: Every person has a different role to play in the organisation, e.g. a developer, a tester, etc. It is a good practice to provide only as much security access as necessary.

In an organisation, every person does not need to install jenkins on their machine. Jenkins is installed on a server which has a private IP Address (and a port number, say 8080) on the org’s network. So within the company you can access Jenkins via the private IP.

Outside the company, we can access Jenkins using the server’s public IP and the port number

For this, we cannot give the admin’s ID and Pass to every person. Instead, we create a user for all our employees with appropriate access granted to them (Similar to AWS IAM).

Hands-on:

* Go to Jenkins Dashboard
* Manage jenkins
* Security -> Manage Users
* Create Two Users
* Try logging in using both users to verify
  + Log out from Admin user
  + Log in from each user one by one
* You will notice that both the users were able to see the same thing as the Admin
  + All created users have complete access by default
* Login as admin again
* To change permissions for users, we will need to install a plug-in
* Dashboard -> Manage Jenkins -> Manage plug-ins
* 
* Tick “Authorize Project” plug-in as well. Otherwise you may not be able to assign roles
* Install without restart
* After it is done installing, restart jenkins to be safe
* Login as admin again
* Manage Jenkins -> (In security section) Configure Global security
* Select “Role-Based Strategy” -> Save
* Try logging in from another user’s account. It will not grant you access to the jobs anymore
* 
* Now, we need to give the relevant permissions
* Login as admin
* Manage Jenkins -> Manage and assign roles -> Manage roles (used for creating a new role)
* We will add an “Employee” role
  + Give the following access in the Employee row
  + 
* Go to Itel roles section
  + Add “Developer”
    - Pattern “dev\*”
    - This pattern will ensure that a naming convention is followed for jobs that only a Developer can access
    - Add
    - Tick all checkboxes
  + Add “Tester”
    - Pattern “test.\*”
    - 
    - Add
    - Tick all checkboxes
  + Give all Job related permissions to Developer
  + Give Read permission in Job Section to Tester
  + Save
* Assign Roles
  + Specify which user/group do you wish to add
    - Enter username and Add
  + Make them employees
  + 
  + Add users in Item roles section and give them assign them roles
  + 
  + Save
* Make a new job starting with the phrase “dev”
* Do the same for “test”
* Login via both users and verify that they can only see their assigned jobs
* For Dev, this should be the output
  + 
* Jobs are starting with dev and the following permissions are allowed
  + 
* Same goes for test
* 
* As expected, tester does not have permissions to modify a job
* 

- **MASTER SLAVE concept**

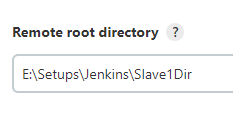
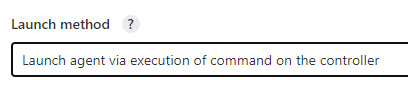
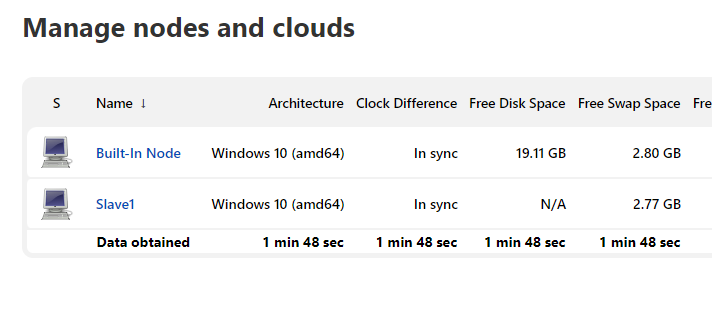
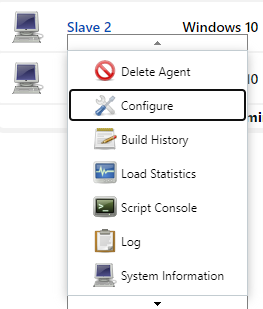
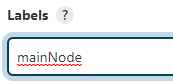
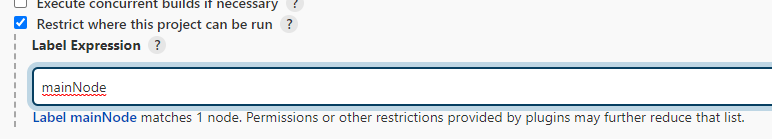
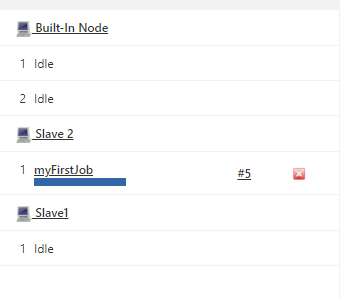
So far, all the jobs that we were building were done by Master. This can be viewed in a specific directory on your machine

Instead of having the master do all the work, we can create slaves to create jobs and other such things. These new jobs will be stored in the slaves’ directory, which we will create.

Then we need to link these directories to the slaves using a file called “agent.jar”, which will specify the path of the directories so that the slaves know where to store their respective jobs

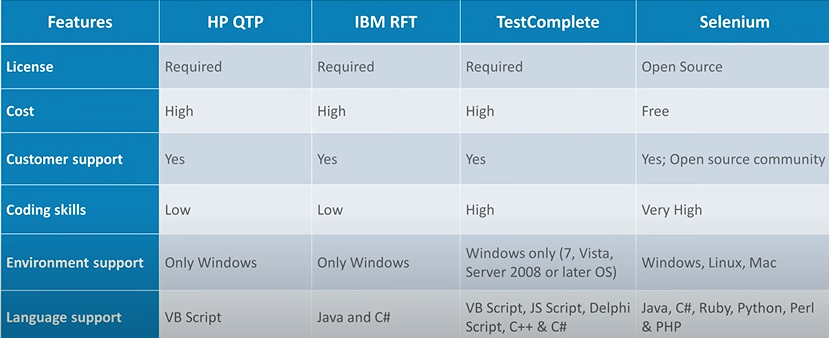
* Job is executed by any of the slaves or the Master, unless we specify using a label

Hands-on:

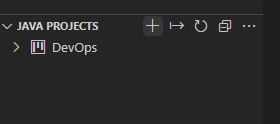
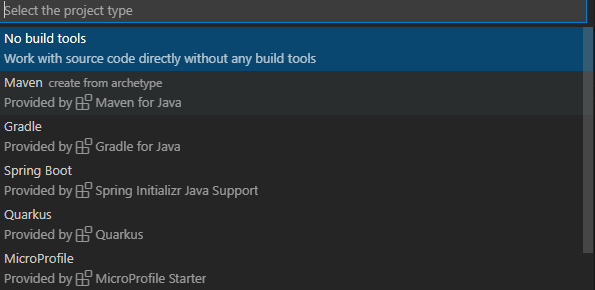
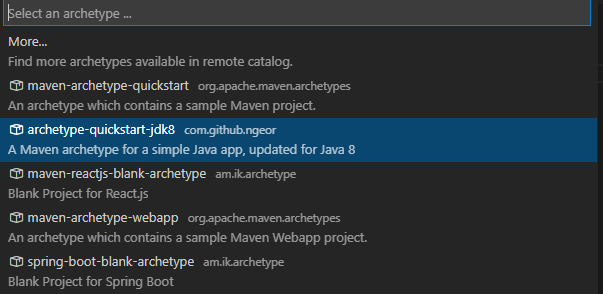
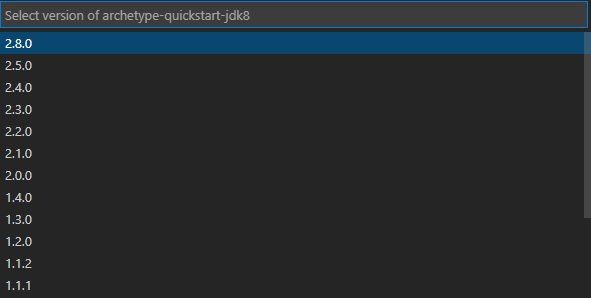
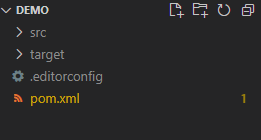
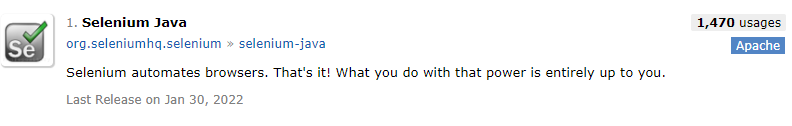
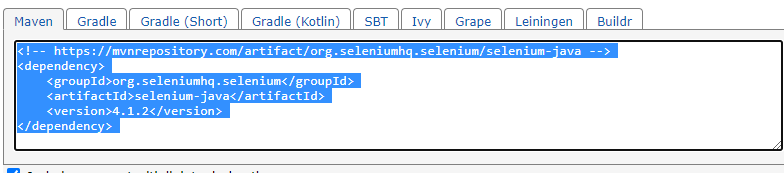
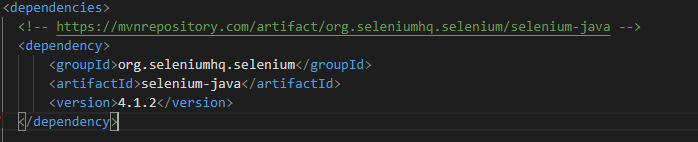
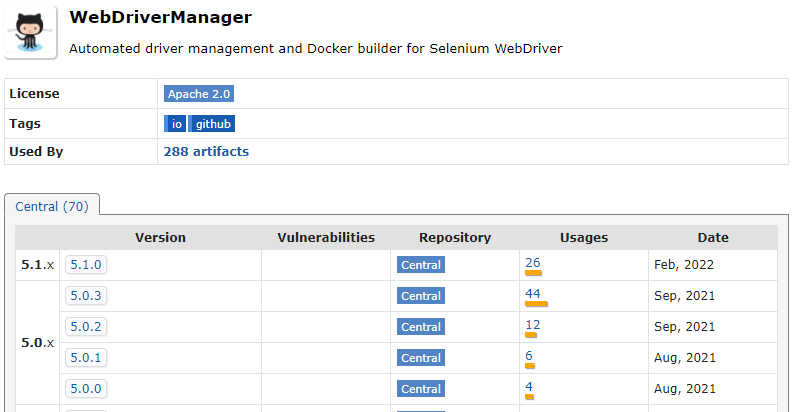
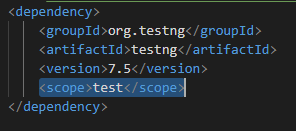
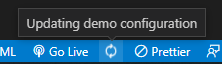
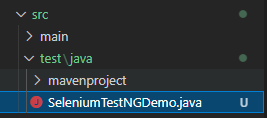
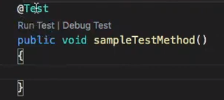
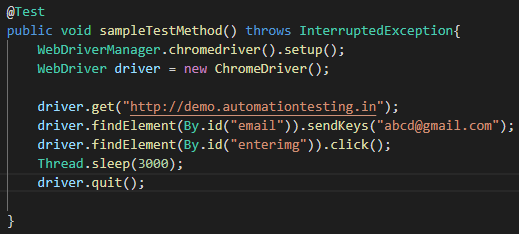
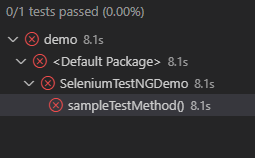
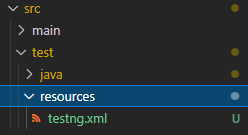
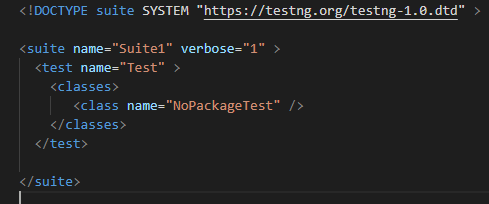
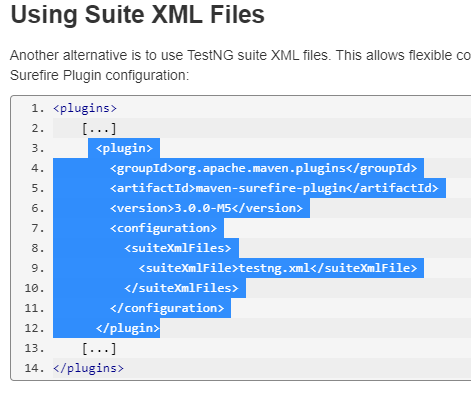
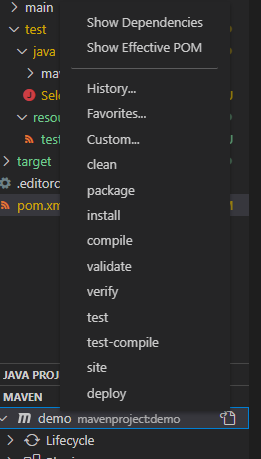
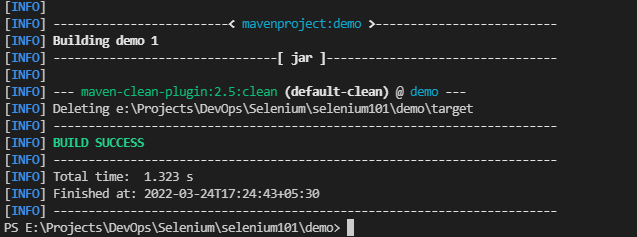
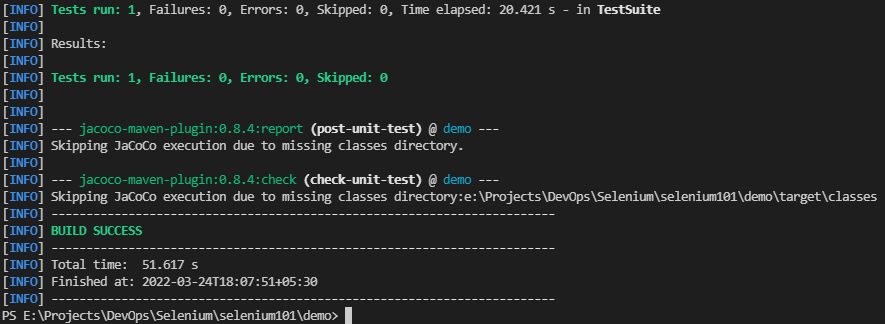
* Dashboard
* Manage Jenkins
* (in **System Configuration**) Manage Nodes and Cloud
* You will find a node by default. That is the Master node
* New Node
  + “Slave1”
  + Select Permanent Agent option
  + OK
  + Choose a path in which you want to create a directory
  + 
  + Here “Slave1Dir” is the name of the directory that I am created. Anything before that is the path to this directory
  + 
  + Select this option
  + It will ask you for a command. Click on the question mark.
  + You will see this message
  + 
  + Download the file, cut and paste it in folder of your choice, copy the path
  + In the command textbox, write “java -jar <path>\agent.jar”
  + Save
  + You will see the Slave in your screen
  + 
  + Refresh the page (as it may appear offline otherwise)
  + Make another Node but for “Slave 2”
  + On the bottom left, you will see all the nodes
  + 
  + When you run a job, notice that the job gets randomly assigned to a node
  + How do we specify a specific node to handle a specific job?
    - Go to the Nodes menu again
    - 
    - Configure
    - Give a label of your choice
    - 
    - Now go to the job you wish to assign a node to -> Configure
    - Enter your node’s label as follows
    - 
    - Build the job. Notice that it will run on the assigned node
    - 

**Introduction to testing**

Potential resource (<https://www.lambdatest.com/blog/selenium-maven-jenkins-integration/>)

* + Objective: To automate the process of testing
  + Tools:
  + 
  + Tool of choice: Selenium
    - Open-source tool used to automate web browsers
    - Runs test cases for any web browser to automate and test the web app
    - Selenium has four major components – **Selenium IDE, Selenium RC, Selenium Web driver, Selenium GRID**
    - RC and WebDriver were merged in subsequent versions as WebDriver, (RC was making the process very slow by having the control pass though RC Server)
  + Set-up: Prerequisites:
    - Download JDK, already done in jenkins section
    - Selenium Set-up:<https://funnelgarden.com/setup-selenium-with-java-on-visual-studio-code/>
    - Follow the instructions in the above link to set up Selenium with VS Code

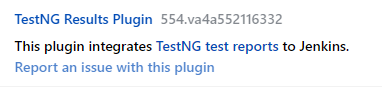
(Chrome Driver can be downloaded [here](https://chromedriver.storage.googleapis.com/index.html?path=99.0.4844.51/))

* Cons of selenium
  + No in built object repository
    - Test script is written in a programming language say Java
    - Web app will be tested using this script
    - An in built repo helps is identifying objects in the test script
    - In its absence, we will have to access multiple files, which is tedious
    - Instead, integrate selenium with git to have this repo
  + Cannot perform image testing
    - Selenium tests functionality (includes image testing) , usability, and consistency of a web application
    - Integrate with a tool called sikuli for image testing
  + No official support
    - Though there is community support
  + No native support for test reports
    - Integrate with TestNG for this
* Terminologies:
  + Data driven testing: Integrating test script with some data (which has sample input and verified outputs) that can be stored in common file types such as csv files, excel sheets, database tables, etc.
  + Parameterization: You can have parameters in your test script as compared to actual data. We can provide different input values via external parameters.
    - Because of this, our test script will only have parameters, no input values will be required. Basically there is les reliance on hard-coding because of this
* Selenium WebDriver:
  + Testing needs to be done on multiple web drivers
  + Proxy server was used to deal with same origin policy
  + Initially, Selenium RC was used, which acted as a proxy server. Later it merged with Selenium WebDriver, which not directly communicated with the browser
  + The test script (written in Java in this case) invokes the selenium web driver, which runs the test cases on different browsers
* Hands-on:
* **Set-up a Maven application and perform tests**
  + <https://www.youtube.com/watch?v=yFzv36r7Z8o&ab_channel=KrishnaSakinala>
  + In VS Code:
    - Install Java Extension Pack on VS Code
  + You will see a java projects tab on the bottom left
  + 
  + Click on + to create a new project
  + Choose Maven
  + 
  + 
  + 
  + It will ask for destination folder
  + Enter the rest as default
  + Terminal will open up and ask for a response for version. Enter 1.0
  + Enter
  + Prompt will tell you project destination and whether you wish to open it or not
  + You will be able to see your project
  + 
  + Check out pom.xml, try to understand the file\
  + Also, you can delete the App.java files in the main and test folders
  + Now we will verify whether the maven project is working fine in VS Code or not
  + Go to [Maven repository](https://mvnrepository.com/)
  + Search for Selenium
  + 
  + Click on the latest version
  + Copy the dependency code
  + 
  + Paste the dependency in pom.xml
  + 
  + Do the same for WebDriverManager and TestNG
  + 
  + 
  + Remove the following highlighted line in pom.xml
  + 
  + Wait for it to finish loading. This will take a while.
  + 
  + 
  + Create a new file in a folder as shown:
  + 
  + Check out the code for this file [here](https://github.com/arindamkeswani/DevOps/blob/main/Selenium/selenium101/demo/src/test/java/mavenproject/DemoMavenVsCode.java)
  + Run
  + A browser should open the website for about 3 seconds, then close
  + Now that we have set-up a maven project, let’s set-up selenium project in VS Code
  + <https://www.youtube.com/watch?v=AxouQUrqwsY&t=136s&ab_channel=KrishnaSakinala>
  + Steps will be as follows:
    - Add Selenium Dependencies
    - Add testNG Dependencies
    - Create testng.xml
    - Execute testng.xml file
  + We have already added the dependencies while creating our Maven project in pom.xml file
  + Create some test cases
    - Create a file inside sec/test/java folder
    - 
    - Define a sample test inside the class
    - 
    - You will either see a Run Test option or a green play button> This is because of Java Extension Pack that we added previously
    - Go to the link, extract selectors for email and go button, e.g.
    - 
    - The code will look as follows:
    - 
    - You can also find the full code with the imports [here](https://github.com/arindamkeswani/DevOps/blob/main/Selenium/selenium101/demo/src/test/java/SeleniumTestNGDemo.java).
    - Run Test
    - The browser will open, take you to the same page as before, enter an email ID, click go. You will be taken to another page, and after 3 seconds, the browser will close
    - This will show up in VS Code
    - 
    - Try making the URL something incorrect. Run the test, this will be the output that you will receive
    - 
    - Create a “resources” folder inside src/test. Create testng.xml inside it
    - 
    - Go to <https://testng.org/doc/documentation-main.html#testng-xml>
    - Copy the snippet, paste in this file
    - After some changes, it should look like this
    - 
    - Change class name to the class of your test script
    - 
    - Instead of executing testng.xml directly, execute it from pom.xml by adding it as an extension
    - Go to <https://maven.apache.org/surefire/maven-surefire-plugin/examples/testng.html>
    - Copy the following
    - 
    - Paste it in pox.xml inside plugins section
    - 
    - If you are prompted to build, click “Now”
    - Right click in the maven project (bottom left of the screen)
    - 
    - Click on clean
    - Output: 
    - Now do the same for “test”. It will execute the test by accessing testng.xml via pom.xml file
    - It will download all the dependencies, which will take a while
    - It should give you an error.
    - To fix that error, go to pom.xml, to the plug-in we just added
    - Make sure you mention the full path like this:
    - 
    - Clean, then compile, then test, like we did before
    - This should be the end of your your output:
    - 
    - If you get checkstyle errors, refer to [this](https://stackoverflow.com/questions/35149422/how-to-fix-the-maven-check-style-error)
    - Or you can remove the checkstyle plugin

[Selenium + Jenkins Integration](https://www.youtube.com/watch?v=lPxrPqSyCo8)

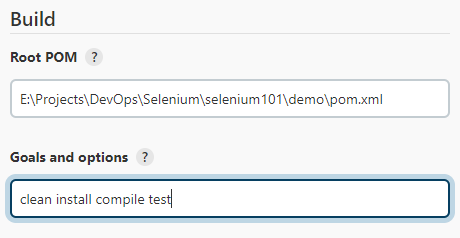
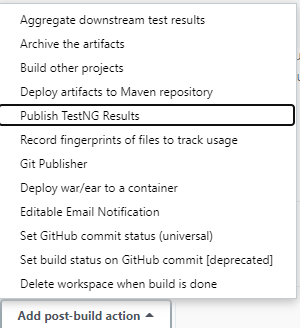
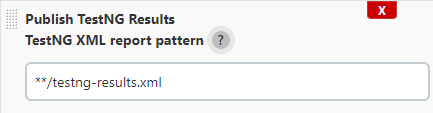
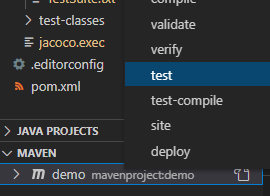
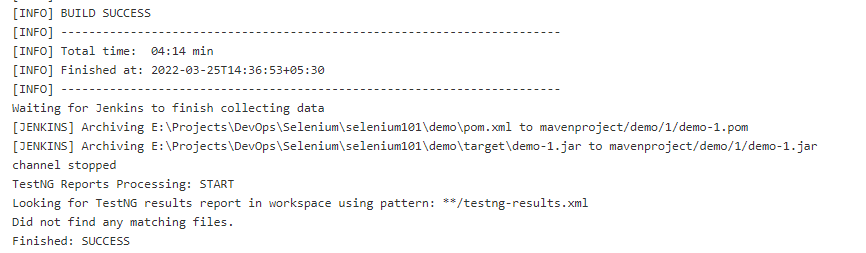
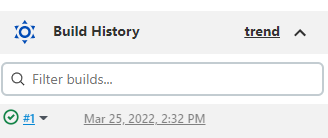
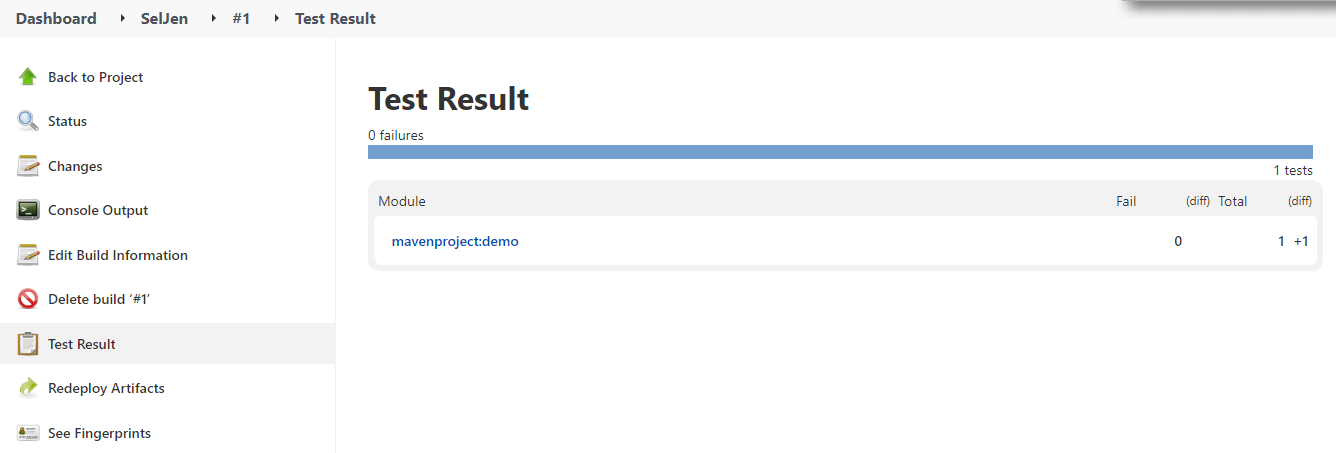
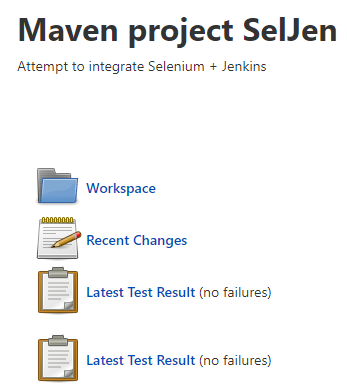
Prerequisites:

1. Install Jenkins (it should have Maven Integration (done before) plug-in, tentNG plug-in)



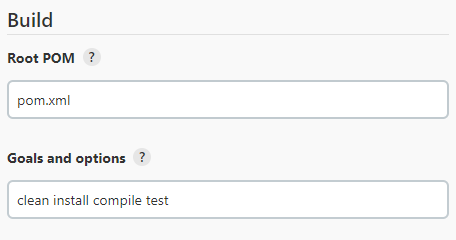
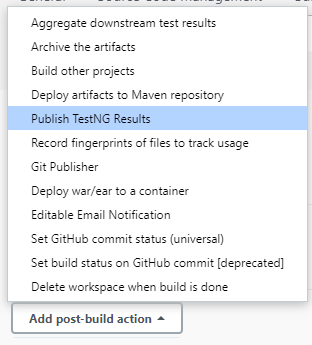
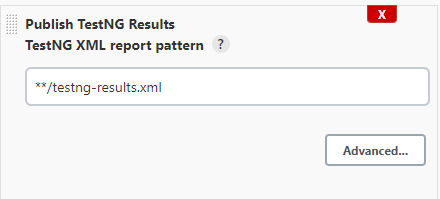
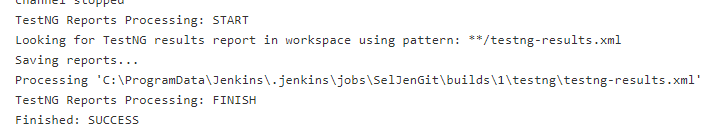
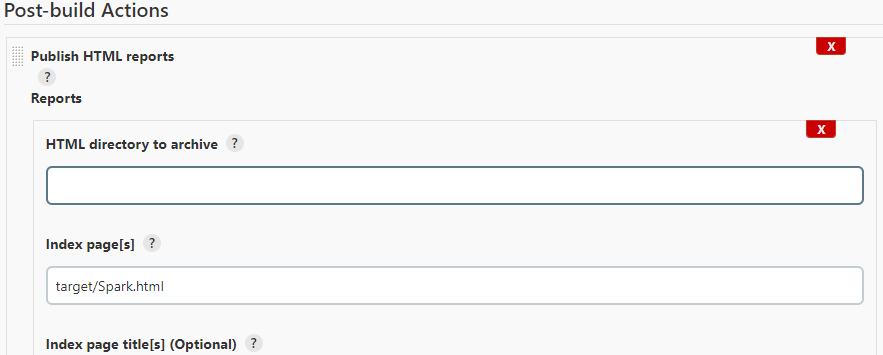
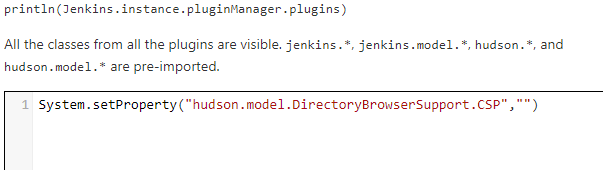
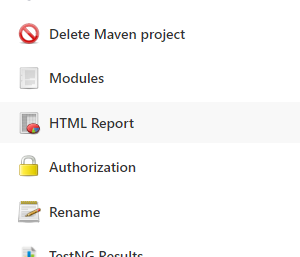
1. Have a sample Java + Maven + Selenium project with sample test cases

Steps:

* Create a new item on Jenkins. It should be a Maven project
* Add a name of your choice. Click OK
* Add a description if you wish to.
* In Build section, paste the complete path of your project’s pom.xml file, and add commands
* 
* You can “clean install” for now
* 
* 
* We are specifying the path of the file in which we want to store our results.
* In case you are not able to see a testng-results.xml file in your local project, run the test from the GUI
* 
* This will create the testng-results.xml file in target/surefire-reports (Ref: <https://www.edureka.co/community/50540/project-generating-results-output-folder-selenium-project>)
* Apply + Save
* Build the project
* Wait for the build to finish
* Check if the tests have passed from the console output
* The end of your console output should be something like this
* 
* You can also do that by clicking on the build number e.g. “#1”
* 
* Click on Test Result. You will be able to see the details
* 
* On the main page of your project, it will show you the details of the latest build
* 

[**Selenium + Jenkins + GitHub Hands on**](https://www.youtube.com/watch?v=rb5OL-QQPw4&t=242s&ab_channel=NaveenAutomationLabs)

Prerequisites: Push the project made in the previous activity on GitHub

* Open Jenkins
* Make a new maven job
* This time in Source Code Management enter your repository URL instead
* Mention your branch, if any
* 
* 
* 
* Build the project
* Sample outputs:
* 
* 
* TO BE SOLVED: Browser is opening when running tests on localhost but not when they are run through jenkins
* OPTIONAL (HOMEWORK):
  + To generate extent reports, we have to add a [dependency](https://mvnrepository.com/artifact/com.aventstack/extentreports) in pom.xml
  + Full process: <https://www.youtube.com/watch?v=MHgSAmDXIWs&ab_channel=TestingMiniBytes>
  + For this, install HTML Publisher plug-in on jenkins, and follow the steps in the tutorial till you are able to generate a file locally
  + If you want to generate reports on jenkins, in post-build options, choose HTML Report option, and add the relative path to your report’s html file
  + 
  + If you generate the report via jenkins, you will notice that the report is not styled like it was when we generated it locally.
    - To fix this, go to Manage Jenkins, Script console
    - Wrtie the following line and Run
    - 
    - It should give the following output
    - 
    - Trigger the build again
    - This time it should work fine
    - Verify by clickingon the HTML report
    - 
  + Currently, we are integrate a Maven project, Jenkins, Selenium, and GitHub while generating 2 types of outputs, testNG trends, and Extent reports