CORE JAVA

Basics of java

- Variables and data types

Variables: Variable in Java is a data container that stores the data values during Java program execution

1. Local variables
2. Instance variables
3. Static variables

Example:

```
class Demo {
   static int a = 1; //static variable
   int data = 99; //instance variable
   void method() {
      int b = 90; //local variable
   }
}
```

Data types: Data Types in Java are defined as specifiers that allocate different sizes and types of values that can be stored in the variable or an identifier

1. Primitive datatypes: which include integer, character, boolean, and float
2. Non primitive datatypes: which include classes, arrays and interfaces.

Java Data Types

Data Type	Default Value	Default size
byte	0	1 byte
short	0	2 bytes
int	0	4 bytes
long	0L	8 bytes
float	0.0f	4 bytes
double	0.0d	8 bytes
boolean	false	1 bit
char	'\u0000'	2 bytes

- Arrays

    1. Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

    2. To declare an array, define the variable type with square brackets:

    Syntax: `String[] cars;`

- Control flow statements
    - If-else:
      Example:
      int time = 20;
      if (time < 18) {
       System.out.println("Good day.");
      } else {
       System.out.println("Good evening.");
      }
    - Switch case:
      Syntax:
      switch(*expression*) {
       case x:
        // code block
        break;
       case y:
        // code block
        break;
       default:
        // code block
      }

Methods

- A **method** is a block of code which only runs when it is called.

- You can pass data, known as parameters, into a method.

- Methods are used to perform certain actions, and they are also known as **functions**.

- Why use methods? To reuse code: define the code once, and use it many times.

Method creation inside the main:

```
public class Main {

 static void myMethod() {

  // code to be executed

 }

}
```

Object orientation

- Class and object
  - Java is an object-oriented programming language.
  - Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake.
    Example:
    To create a class, use the keyword class:
    Create a class named "Main" with a variable x:

```
public class Main {
  int x = 5;
}
```

    Example:

    In Java, an object is created from a class. We have already created the class named Main, so now we can use this to create objects.

    To create an object of Main, specify the class name, followed by the object name, and use the keyword new:

    Create an object called "myObj" and print the value of x:

```
public class Main {

  int x = 5;
```

```
  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

- Inheritance

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

- **subclass** (child) - the class that inherits from another class
- **superclass** (parent) - the class being inherited from

To inherit from a class, use the extends keyword.

In the example below, the Car class (subclass) inherits the attributes and methods from the Vehicle class (superclass):

- Interface
  Another way to achieve abstraction in Java, is with interfaces.
  An interface is a completely "abstract class" that is used to group related methods with empty bodies:

- Abstract classes
  Data abstraction is the process of hiding certain details and showing only essential information to the user.
  Abstraction can be achieved with either abstract classes or interfaces (which you will learn more about in the next chapter).

  The abstract keyword is a non-access modifier, used for classes and methods:

  Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

  Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).
  An abstract class can have both abstract and regular methods:

Loops

- While
  The while loop loops through a block of code as long as a specified condition is true:
  Syntax:
  while (condition) {
  // code block to be executed
  }
- For
  When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:
- Nested loops and debugging
  Nested loop is said to be the loop inside another loop.debugging is said to be that verify weather the code is run properly or not ,and we can keep break points and identify the execution flow.

Running programs with command line

C:\Users\Your Name>javac Main.java

C:\Users\Your Name>java Main

Output: Hello World

Creating jar files using command line

```
javac HelloWorld.java
```

Creating an Executable JAR File

```
Main-Class: HelloWorld
```

Next, we can create a JAR file by running the following jar command.

```
jar -cfm HelloWorld.jar ManifestFile.txt HelloWorld.class
```

The **-c flag** is used to create an archive file. The **-f flag** is used to specify the file name. And the **-m flag** will include the content of the manifest file.

Creating a Non-Executable JAR File

To create a non-executable JAR file, we will exclude the **-m flag**. We don't need to pass the name of the manifest file to our command.

```
jar -cf HelloWorld.jar HelloWorld.class
```

Running JAR Files

Running the Executable JAR File

We can use the following java command with the **-jar** option to run the executable JAR file.

```
java -jar HelloWorld.jar
```

The output of the command is shown below.

Hello world!

Running the Non-Executable JAR File

Instead of using the **-jar option**, we will use the **-cp option** to run a **non-executable JAR** file. We need to specify the JAR file name and the main class name in the command.

```
java -cp HelloWorld.jar HelloWorld
```

The output of this command is shown below.

Hello world!

File processing and exception handling

Basic example of execption:

```
class Exception{
public static void main(String args[]){
try{
//code that may raise exception
}
catch(Exception e){
// rest of the program
 }
 }
}
```

- Try
  The try block contains a set of statements where an exception can occur. It is always followed by a catch block, which handles the exception that occurs in the associated try block. A try block must be followed by catch blocks or finally block or both.
  Syntax:
  try{
  //code that may throw exception
  }catch(Exception_class_Name ref){}
- Catch
  A catch block is where you handle the exceptions. This block must follow the try block and a single try block can have several catch blocks associated with it.
- Finally
  A finally block contains all the crucial statements that must be executed whether an exception occurs or not. The statements present in this block will always execute, regardless an exception occurs in the try block or not such as closing a connection, stream etc.
  Example:

```
class SampleFinallyBlock{
 public static void main(String args[]){
   try{
     int data=55/5;
     System.out.println(data);
    }
    catch(NullPointerException e)
       {System.out.println(e);}
    finally {System.out.println("finally block is executed");}
    System.out.println("remaining code");
  }
}
```

Collections frame work

- Array list
  Example:
  import java.util.*;
  class TestJavaCollection1{
  public static void main(String args[]){
  ArrayList<String> list=new ArrayList<String>();//Creating arraylist
  list.add("Ravi");//Adding object in arraylist
  list.add("Vijay");

```
list.add("Ravi");
list.add("Ajay");
//Traversing list through Iterator
Iterator itr=list.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

- Linked list

  Example:

```
import java.util.*;
public class TestJavaCollection2{
public static void main(String args[]){
LinkedList<String> al=new LinkedList<String>();
al.add("Ravi");
al.add("Vijay");
al.add("Ravi");
al.add("Ajay");
Iterator<String> itr=al.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

- Vector

  Example:

```
import java.util.*;
public class TestJavaCollection3{
public static void main(String args[]){
Vector<String> v=new Vector<String>();
v.add("Ayush");
v.add("Amit");
v.add("Ashish");
v.add("Garima");
Iterator<String> itr=v.iterator();
```

```
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

- Hash set

  Example:

```
import java.util.*;
public class TestJavaCollection7{
public static void main(String args[]){
//Creating HashSet and adding elements
HashSet<String> set=new HashSet<String>();
set.add("Ravi");
set.add("Vijay");
set.add("Ravi");
set.add("Ajay");
//Traversing elements
Iterator<String> itr=set.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

- Linked hash set

  Example:

```
import java.util.*;
public class TestJavaCollection8{
public static void main(String args[]){
LinkedHashSet<String> set=new LinkedHashSet<String>();
set.add("Ravi");
set.add("Vijay");
set.add("Ravi");
set.add("Ajay");
Iterator<String> itr=set.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
```

```
}
}
}
```

Java generics

- Generics
  Generics means parameterized types. The idea is to allow type (Integer, String, … etc., and user-defined types) to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types.
  Type Parameters in Java Generics:
  1. T – Type
  2. E – Element
  3. K – Key
  4. N – Number
  5. V – Value

BaseType <Type> obj = new BaseType <Type>()

- Generics with wildcard
  The question mark (?) is known as the wildcard in generic programming. It represents an unknown type. The wildcard can be used in a variety of situations such as the type of a parameter, field, or local variable; sometimes as a return type.
  Types of wildcards in Java:
  1. Upper Bounded Wildcards:
     public static void add(List<? extends Number> list)
  2. Lower Bounded Wildcards:
     Syntax: Collectiontype <? super A>

Concurrency in java

- Threads and runnable interface
- Thread safety with synchronization
- Thread safety with collections

JDBC

- Preparing database and basic SQL commands
- Use jdbc to send sql statements after connection
- Curd operations using jdbc

Lambda expressions

Docker Technology

- Docker technology is one implementation of container based virtualization technologies.
- Docker is a software which provides centralized platform to execute your application. It wraps software components into a complete standardized unit which contains everything require to run.
- Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.
- The initial release of Docker was in March 2013 and since then, it has become the buzzword for modern world development, especially in the face of Agile-based projects.

Features of Docker

- Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
- With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
- You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
- Since Docker containers are pretty lightweight, they are very easily scalable.

Components of Docker

Docker has the following components

- **Docker for Mac** − It allows one to run Docker containers on the Mac OS.
- **Docker for Linux** − It allows one to run Docker containers on the Linux OS.
- **Docker for Windows** − It allows one to run Docker containers on the Windows OS.
- **Docker Engine** − It is used for building Docker images and creating Docker containers.

When Do You Need to Use a Docker?

- To run your code locally on your laptop while replicating the environment on your server.

- During various development phases (dev/test/QA), Docker CI/CD was used.
- As a version control system and for distributing your app's OS with a team.

How Do You Setup a Docker Locally

- Download the Docker Toolbox and a Docker edition.
- Check to see if your BIOS supports Virtualization Technologies, AMD-V, or KVM.
- Install the Oracle VirtualBox Extension Pack.
- Run the Setup.

How Do You Use a Docker?

The most significant benefit of virtual machines is that they create snapshots that can be reverted to at any time.

Docker containers improve lightweight process virtualization by being OS agnostic and utilizing the Linux Kernel's capabilities.

They're made from Docker images, similar to snapshots. A Docker file is used to create Docker images, which can be customized or used as is 'libcontainer' is the default execution driver for docker containers.

Docker Hub can be used to look up docker images and see how they were created.

To download the hello world image, the following command in the terminal to download the Docker hello world image –

$ docker run hello world

Use the following command to determine the number of images on your system –

$ docker images

Using the Docker Hub to find an image –

$ docker search <image>

Here's a List of Docker Commands

- docker run – Starts a new container and executes a command.
- docker start – Starts one or more containers that have been stopped.
- docker stop – Puts an end to one or more currently running containers.
- Docker file- It is a command that creates an image.
- Docker pull - Pulls an image or a repository from a registry.
- Docker push. Pushes an image or a repository to a registry.
- docker export – Creates a tar archive of a container's filesystem.
- docker exec – Executes command in a container at runtime.
- Docker search – Looks for images on the Docker Hub.
- docker attach. Attaches to a running container
- docker commit – Creates a new image based on the changes made to a container.

Examples of Using a Docker

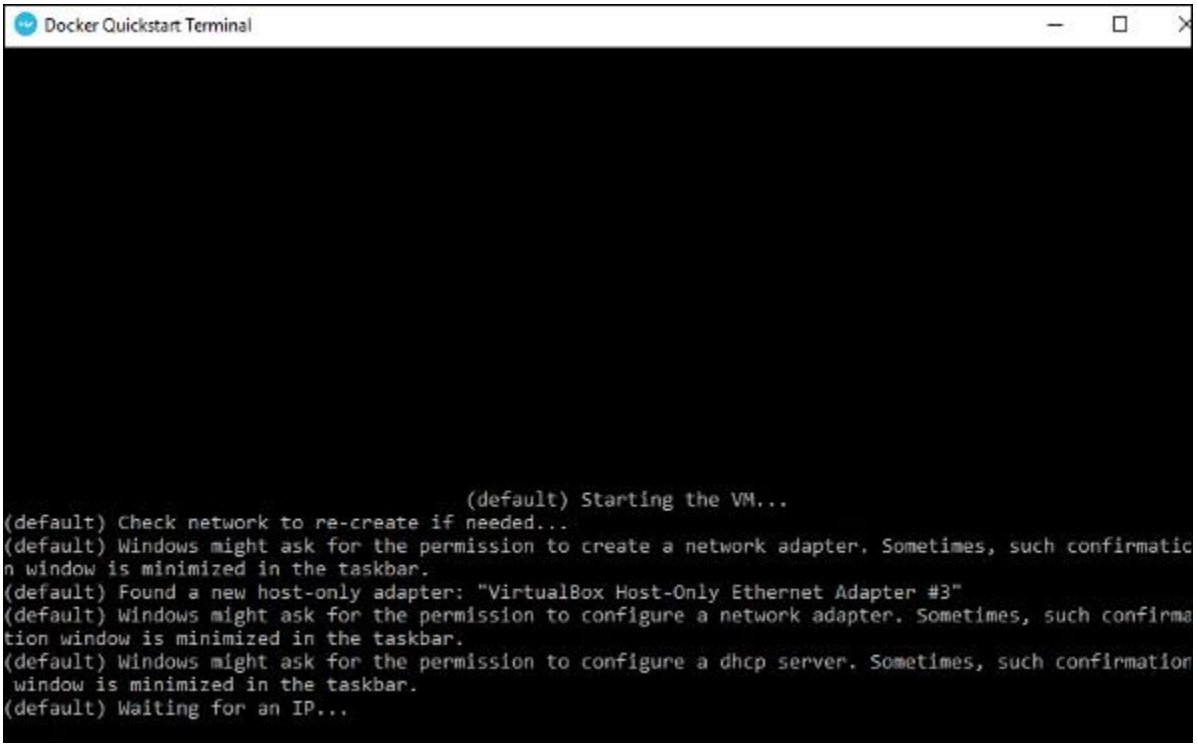
- By downloading Docker, you can run WordPress on your laptop without having to install Apache, PHP, MySQL, or other software. In order to run Docker in a virtual machine, the Docker Toolbox creates a containerized version of Linux.
- Install Oracle VirtualBox using Docker Tool Box.
- Open VirtualBox and install the Extension Pack.
- To verify that your installation was successful, type $ docker run hello-world in the terminal.
- To install WordPress locally, search for a WordPress image on the Docker Hub.
- Dockers can also be used to set up DokuWiki.
- Testing SDN components with Dockers is possible.

Here are a few examples to help you get started with your Docker engine.

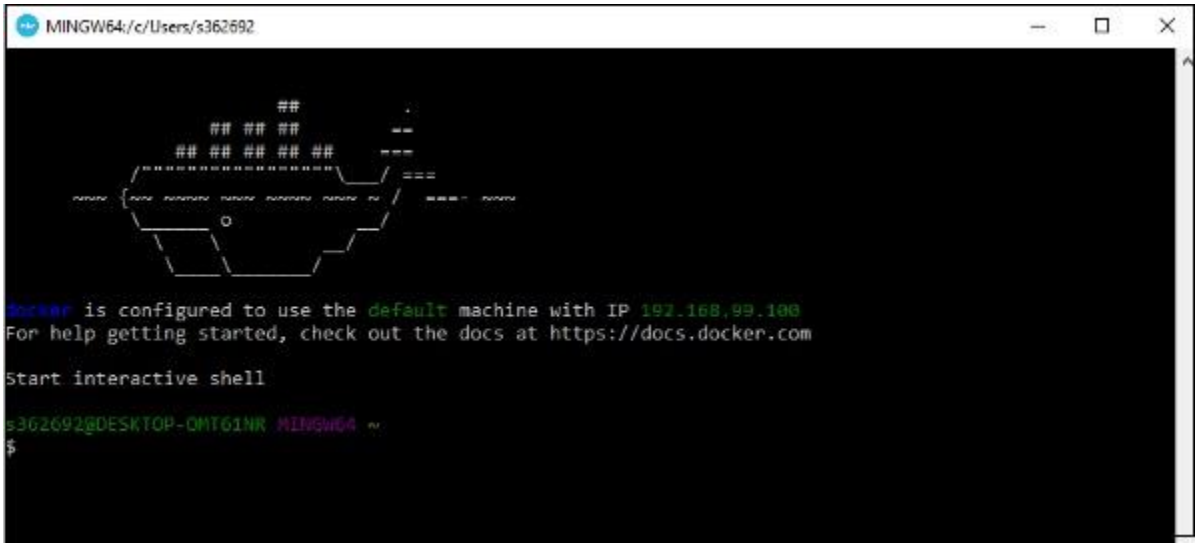
Working with Docker Toolbox

Let's now look at how Docker Toolbox can be used to work with Docker containers on Windows. The first step is to launch the Docker Toolbox application for which the shortcut is created on the desktop when the installation of Docker toolbox is carried out.

Next, you will see the configuration being carried out when Docker toolbox is launched.



Once done, you will see Docker configured and launched. You will get an interactive shell for Docker.



To test that Docker runs properly, we can use the Docker **run command** to download and run a simple **HelloWorld Docker container**.

The working of the Docker **run command** is given below −

```
docker run
```

This command is used to run a command in a Docker container.

Syntax

```
docker run image
```

Options

- **Image** − This is the name of the image which is used to run the container.

Return Value

The output will run the command in the desired container.

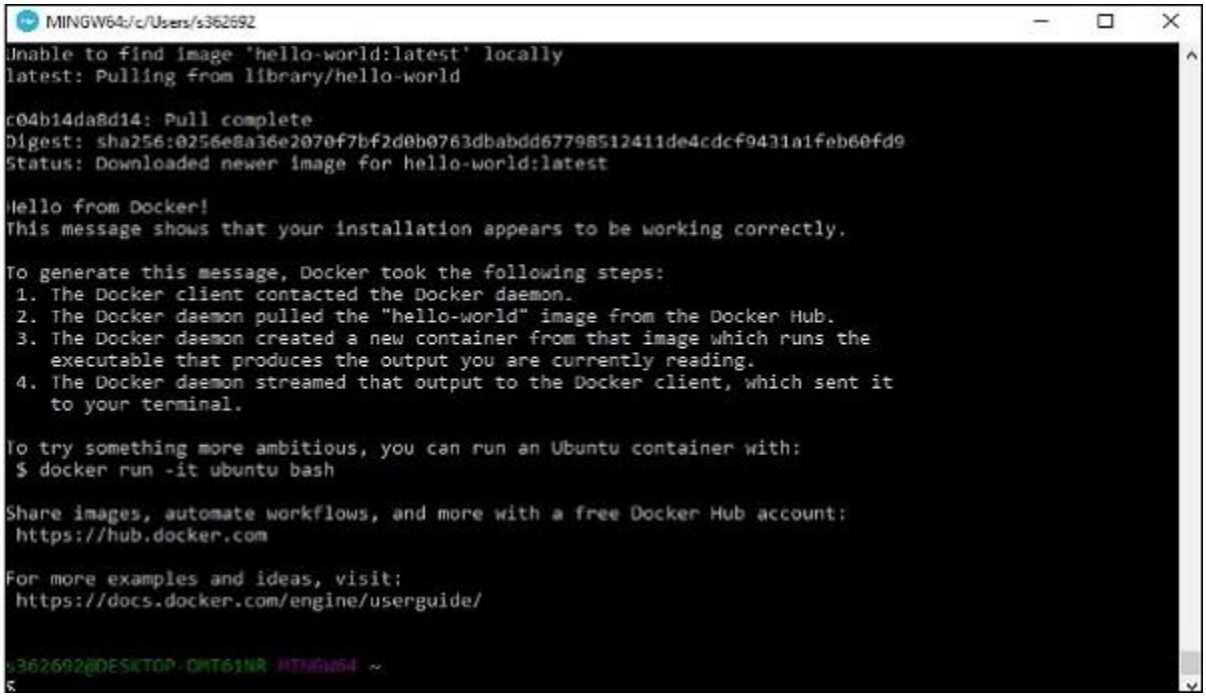
Example

```
sudo docker run hello-world
```

This command will download the **hello-world** image, if it is not already present, and run the **hello-world** as a container.

Output

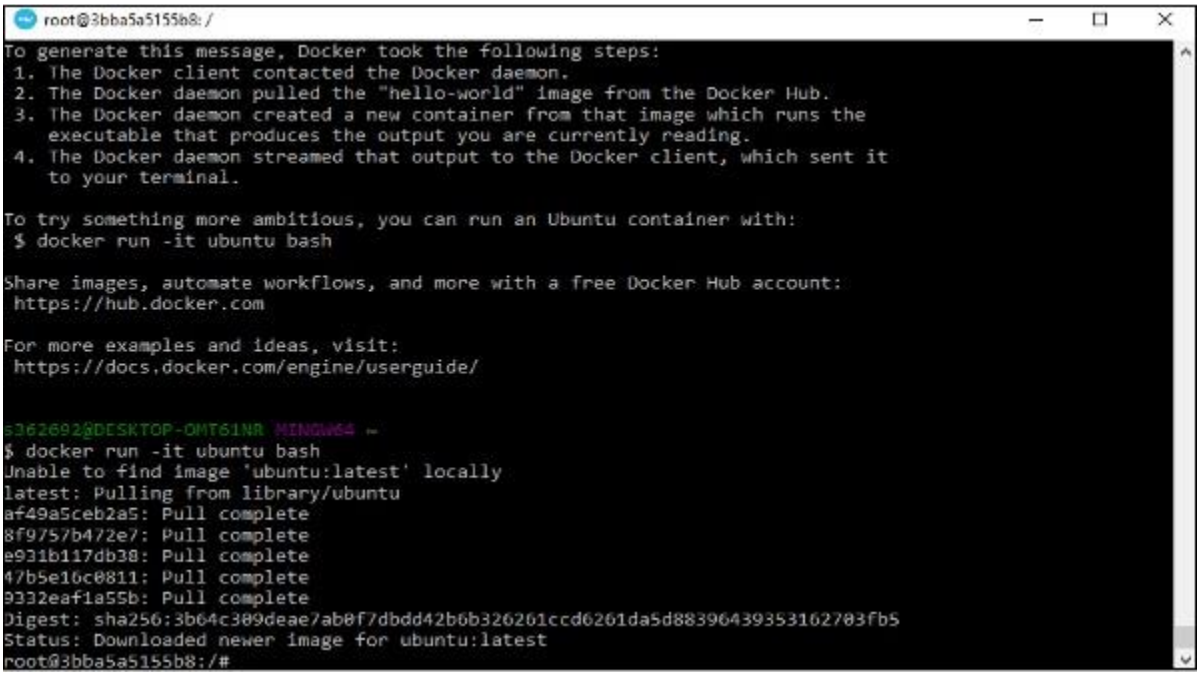
When we run the above command, we will get the following result −



If you want to run the Ubuntu OS on Windows, you can download the Ubuntu Image using the following command −

```
Docker run –it ubuntu bash
```

Here you are telling Docker to run the command in the interactive mode via the **–it** option.



In the output you can see that the Ubuntu image is downloaded and run and then you will be logged in as a root user in the Ubuntu container.

Important Docker Concepts Images

• Images are read only templates used to create containers.

• Images are created with the docker build command, either by us or by other docker users.

• Images are composed of layers of other images.

• Images are stored in a Docker registry.

Containers

• If an image is a class, then a container is an instance of a class - a runtime object.

• Containers are lightweight and portable encapsulations of an environment in which to run applications.

• Containers are created from images. Inside a container, it has all the binaries and dependencies needed to run the application.

Registries and Repositories

• A registry is where we store our images.

• You can host your own registry, or you can use Docker's public registry which is called DockerHub.

• Inside a registry, images are stored in repositories.

• Docker repository is a collection of different docker images with the same name, that have different tags, each tag usually represents a different version of the image.

Why Using Official Images

• Clear Documentation

• Dedicated Team for Reviewing Image Content

• Security Update in a Timely Manner

Docker commit

• Docker commit command would save the changes we made to the Docker container's file system to a new image.

docker commit container_ID repository_name:tag

Dockerfile and Instructions

• A Dockerfile is a text document that contains all the instructions users provide to assemble an image.

• Each instruction will create a new image layer to the image.

• Instructions specify what to do when building the image.

Docker Build Context

• Docker build command takes the path to the build context as an argument.

• When build starts, docker client would pack all the files in the build context into a tarball then transfer the tarball file to the daemon.

• By default, docker would search for the Dockerfile in the build context path.

Dockerfile In Depth Steps

1. Spin up a container from a base image.

2. Install Git package in the container.

3. Commit changes made in the container.

Chain RUN Instructions

• Each RUN command will execute the command on the top writable layer of the container, then commit the container as a new image.

• The new image is used for the next step in the Dockerfile. So each RUN instruction will create a new image layer.

• It is recommended to chain the RUN instructions in the Dockerfile to reduce the number of image layers it creates.

Sort Multi-line Arguments Alphanumerically

• This will help you avoid duplication of packages and make the list much easier to update.

CMD Instructions

• CMD instruction specifies what command you want to run when the container starts up.

• If we don't specify CMD instruction in the Dockerfile, Docker will use the default command defined in the base image.

• The CMD instruction doesn’t run when building the image, it only runs when the container starts up

. • You can specify the command in either exec form which is preferred or in shell form.

Docker Cache

• Each time Docker executes an instruction it builds a new image layer.

• The next time, if the instruction doesn't change, Docker will simply reuse the existing layer.

Dockerfile with Aggressive Caching

FROM ubuntu:14.04

RUN apt-get update

RUN apt-get install -y git

Cache Busting

FROM ubuntu:14.04

RUN apt-get update && apt-get install -y \

git \

curl

Cache Busting

• You can also achieve cache-busting by specifying a package version. This is known as version pinning.

RUN apt-get update && apt-get install -y \

package-bar \

package-baz \

package-foo=1.3.*

How container links work behind the scenes?

Benefits of Docker Container Links

• The main use for docker container links is when we build an application with a microservice architecture, we are able to run many independent components in different containers.

• Docker creates a secure tunnel between the containers that doesn't need to expose any ports externally on the container.

Why Docker Compose?

Manual linking containers and configuring services become impractical when the number of containers grows

. • Docker compose is a very handy tool to quickly get docker environment up and running.

• Docker compose uses yaml files to store the configuration of all the containers, which removes the burden to maintain our scripts for docker orchestration.

Docker Compose Commands

• docker compose up starts up all the containers.

• docker compose ps checks the status of the containers managed by docker compose.

• docker compose logs outputs colored and aggregated logs for the compose-managed containers. • docker compose logs with dash f option outputs appended log when the log grows.

• docker compose logs with the container name in the end outputs the logs of a specific container. • docker compose stop stops all the running containers without removing them.

• docker compose rm removes all the containers.

• docker compose build rebuilds all the images

Introduction to Docker Networking

Docker Network Types

• Closed Network / None Network

• Bridge Network

• Host Network

• Overlay Network

None Network

None Network Isolated Isolated Isolated None Network

• Provides the maximum level of network protection.

• Suites well where the container require the maximum level of network security and network access is not necessary.

Host Network

• The least protected network model, it adds a container on the host's network stack

. • Containers deployed on the host stack have full access to the host's interface.

• This kind of containers are usually called open containers.

• Minimum network security level.

• No isolation on this type of open containers, thus leave the container widely unprotected.

• Containers running in the host network stack should see a higher level of performance than those traversing the docker0 bridge and iptables port mappings.

Define Container Networks with Docker Compose

• Unit tests should test some basic functionality of our docker app code, with no reliance on external services.

• Unit tests should run as quickly as possible so that developers can iterate much faster without being blocked by waiting for the tests results.

• Docker containers can spin up in seconds and can create a clean and isolated environment which is great tool to run unit tests with. Unit Tests in Containers

Pros: • A single image is used through development, testing and production, which greatly ensures the reliability of our tests.

Cons: • It increases the size of the image.

Introduction to Docker Swarm and Set up Swarm cluster

How Swarm cluster works

• To deploy your application to a swarm, you submit your service to a manager node.

• The manager node dispatches units of work called tasks to worker nodes.

• Manager nodes also perform the orchestration and cluster management functions required to maintain the desired state of the swarm.

• Worker nodes receive and execute tasks dispatched from manager nodes.

• An agent runs on each worker node and reports on the tasks assigned to it. The worker node notifies the manager node of the current state of its assigned tasks so that the manager can maintain the desired state of each worker

Docker Services

• The services can be defined in our Docker compose file.

• The service definition includes which Docker images to run, the port mapping and dependency between services.

Docker Stack

• A docker stack is a group of interrelated services that share dependencies, and can be orchestrated and scaled together

. • You can image that a stack is a live collection of all the services defined in your docker compose file.

• Create a stack from your docker compose file:

– docker stack deploy

• In the Swarm mode,

– Docker compose files can be used for service definitions.

– Docker compose commands can't be reused. Docker compose commands can only schedule the containers to a single node.

– We have to use docker stack command. You can think of docker stack as the docker compose in the swarm mode.

How to update our services in Production?

Provision a Swarm Cluster

• Step 1: Deploy two VMs, one will be used for the Swam manager node, and the other one will be used as a worker node.

• Step 2: Appoint the first VM as Swarm manager node and initialize a Swarm cluster. – docker swarm init

• Step 3: Let the second VM join the Swarm cluster as a worker node.

– docker swarm join

Docker Swarm commands

• docker swarm init

– Initialize a swarm. The docker engine targeted by this command becomes a manager in the newly created single-node swarm.

• docker swarm join

– Join a swarm as a Swarm node.

• docker swarm leave

– Leave the swarm

GIT SOURCE CONTROL

What is git?

Language and Platform independent: SOAP web services can be written in any programming language and executed in any platform.

Basic git commands

Tell Git who you are

1. Configure the author name and email address to be used with your commits.
   - `git config --global user.name "Sam Smith"`
   - `git config --global user.email sam@example.com`

Create a new local repository

2. Create a new local repository
   - git init

Check out a repository

3. Create a working copy of a local repository:
   - git clone /path/to/repository
4. Add one or more files to staging (index):
   - git add <filename>
   - git add *

Commit

5. Commit changes to head (but not yet to the remote repository):
   - git commit -m "Commit message"
6. Commit any files you've added with git add, and also commit any files you've changed since then:
   - git commit -a

Push

7. Send changes to the master branch of your remote repository:
   - git push origin master

Status

8. List the files you've changed and those you still need to add or commit:
   - git status

Connect to a remote repository

9. If you haven't connected your local repository to a remote server, add the server to be able to push to it:
   - git remote add origin <server>
10. List all currently configured remote repositories:
    - git remote -v

GIT SOURCE CONTROL

Branches

11. Create a new branch and switch to it:
    - git checkout -b <branchname>
12. Switch from one branch to another:
    - git checkout <branchname>
13. List all the branches in your repo, and also tell you what branch you're currently in:
    - git branch
14. Delete the feature branch:
    - git branch -d <branchname>
15. Push the branch to your remote repository, so others can use it:
    - git push origin <branchname>
16. Push all branches to your remote repository:
    - git push --all origin
17. Delete a branch on your remote repository:
    - git push origin :<branchname>

Update from the remote repository

18. Fetch and merge changes on the remote server to your working directory:
    - git pull
19. To merge a different branch into your active branch:
    - git merge <branchname>
20. View all the merge conflicts:
    - git diff
21. View the conflicts against the base file:
    - git diff --base <filename>
22. Preview changes, before merging:
    - git diff <sourcebranch> <targetbranch>
23. After you have manually resolved any conflicts, you mark the changed file:
    - git add <filename>

Tags

24. You can use tagging to mark a significant changeset, such as a release:
    - git tag 1.0.0 <commitID>
25. CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using:
    - git log
26. Push all tags to remote repository:
    - git push --tags origin

Undo local changes

27. If you mess up, you can replace the changes in your working tree with the last content in head:Changes already added to the index, as well as new files, will be kept.
    - git checkout -- <filename>
28. Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this:

- git fetch origin
- git reset --hard origin/master

Search

29. Search the working directory for foo():
    - git grep "foo()"

HYBERNATE AND JPA

JPA(JAVA PERSISTANCE API)

The Java Persistence API (JPA) is a specification of Java. It is used to persist data between Java object and relational database. JPA acts as a bridge between object-oriented domain models and relational database systems.

As JPA is just a specification, it doesn't perform any operation by itself. It requires an implementation. So, ORM tools like Hibernate, TopLink and iBatis implements JPA specifications for data persistence.

Hibernate Framework

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

ORM Tool

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.

Advantages of Hibernate Framework

- Open Source and Lightweight
- Fast Performance
- Database Independent Query
- Automatic Table Creation
- Simplifies Complex Join
- Provides Query Statistics and Database Status

JPA Creating an Entity

A Java class can be easily transformed into an entity. For transformation the basic requirements are: -

- No argument constructor
- Annotation

To transform the java class into an entity add @Entity and @Id annotation in it.

- @Entity - This is a marker annotation which indicates that this class is an entity. This annotation must be placed on the class name.
- @Id - This annotation is placed on a specific field that holds the persistent identifying properties. This field is treated as a primary key in database.

HYBERNATE AND JPA

JPA Entity Manager

Important roles of an entity manager: -

- The entity manager implements the API and encapsulates all of them within a single interface.
- Entity manager is used to read, delete and write an entity.
- An object referenced by an entity is managed by entity manager.

Steps to persist an entity object.

1. Creating an entity manager factory object

EntityManagerFactory emf=Persistence.createEntityManagerFactory("Student_details");

The EntityManagerFactory interface present in java.persistence package is used to provide an entity manager.

2. Obtaining an entity manager from factory.

   EntityManager em=emf.createEntityManager();

3. Intializing an entity manager.

   em.getTransaction().begin();

4. Persisting a data into relational database.
   em.persist(s1);
5. Closing the transaction
   em.getTransaction().commit();
6. Releasing the factory resources.
   emf.close();
   em.close();

Entity Operations

- Inserting an Entity
- Finding an Entity
- Updating an Entity
- Deleting an Entity

HYBERNATE AND JPA

Relationships with JPA and hibernate

- One to one mapping
- Many to one mapping
- Many to many mapping

One to one mapping:

Annotation: @OneToOne(targetEntity=Address.class,cascade=CascadeType.ALL)

Here, we are going to perform one to one mapping by one-to-one element using annotation. In such case, no foreign key is created in the primary table.

For example, one employee can have one address and one address belongs to one employee only. Here, we are using bidirectional association.

There are two persistent classes Employee.java and Address.java. Employee class contains Address class reference and vice versa.

Many to one mapping:

Annotation: @ManyToOne(cascade=CascadeType.ALL)

In many to one mapping, various attributes can be referred to one attribute only.

for example, every employee has one company address only and one address belongs to many employees. Here, we are going to perform many to one mapping using annotation.

There are two persistent classes Employee.java and Address.java. Employee class contains Address class reference and vice versa.

Many to many mapping:

Annotation: @ManyToMany(targetEntity = Answer.class, cascade = { CascadeType.ALL })

We can map many to many relation either using list, set, bag, map etc. Here, we are going to use list for many-to-many mapping. In such case, three tables will be created.

For example, we will generate a many to many relation between questions and answers by list.so here we can have one question we can have different answers and same answer with different questions.

Inheritance Hierarchy

- Single table inheritance:
  In this strategy, all the classes in a hierarchy are mapped to a single table.
  The annotation @Inheritance is used on the root entity class
- Table per class:
  In this strategy, the superclass and subclasses in a hierarchy are mapped to different individual tables.
  The annotation @Inheritance is used on the root entity class

- Joined:
  Using the joined strategy will create a separate table for parent class and a separate table for each of its child classes

Java Persistence Query language

JPQL is Java Persistence Query Language defined in JPA specification. It is used to create queries against entities to store in a relational database. JPQL is developed based on SQL syntax. But it won't affect the database directly.

JPQL can retrieve information or data using SELECT clause, can do bulk updates using UPDATE clause and DELETE clause. EntityManager.createQuery() API will support for querying language.

SQL works directly against relational database tables, records and fields, whereas JPQL works with Java classes and instances

Transaction management

Transaction Interface in Hibernate

In hibernate framework, we have Transaction interface that defines the unit of work. It maintains abstraction from the transaction implementation (JTA,JDBC).

A transaction is associated with Session and instantiated by calling session.beginTransaction().

The methods of Transaction interface are as follows:

void begin()

void commit()

void rollback()

void setTimeout(int seconds)

boolean isAlive()

void registerSynchronization(Synchronization s)

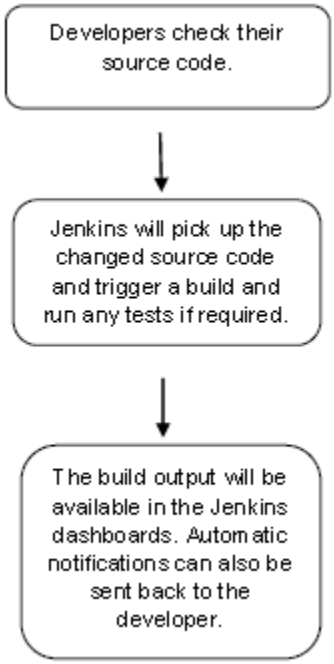
boolean wasCommited()

boolean wasRolledBack()

Jenkins – an open source automation server which enables developers around the world to reliably build, test, and deploy their software.

Why Jenkins?

Jenkins is a software that allows **continuous integration**. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.



Along with Jenkins, sometimes, one might also see the association of **Hudson**. Hudson is a very popular open-source Java-based continuous integration tool developed by Sun Microsystems which was later acquired by Oracle. After the acquisition of Sun by Oracle, a fork was created from the Hudson source code, which brought about the introduction of Jenkins.

What is Continuous Integration?

Continuous Integration is a development practice that requires developers to integrate code into a shared repository at regular intervals. This concept was meant to remove the problem of finding later occurrence of issues in the build lifecycle. Continuous integration requires the developers to have frequent builds. The common practice is that whenever a code commit occurs, a build should be triggered.

Starting Jenkins

Open the command prompt. From the command prompt, browse to the directory where the jenkins.war file is present. Run the following command

```
D:\>Java –jar Jenkins.war
```

After the command is run, various tasks will run, one of which is the extraction of the war file which is done by an embedded webserver called winstone.

```
D:\>Java –jar Jenkins.war
Running from: D:\jenkins.war
Webroot: $user.home/ .jenkins
Sep 29, 2015 4:10:46 PM winstone.Logger logInternal
INFO: Beginning extraction from war file
```

Once the processing is complete without major errors, the following line will come in the output of the command prompt.

```
INFO: Jenkins is fully up and running
```

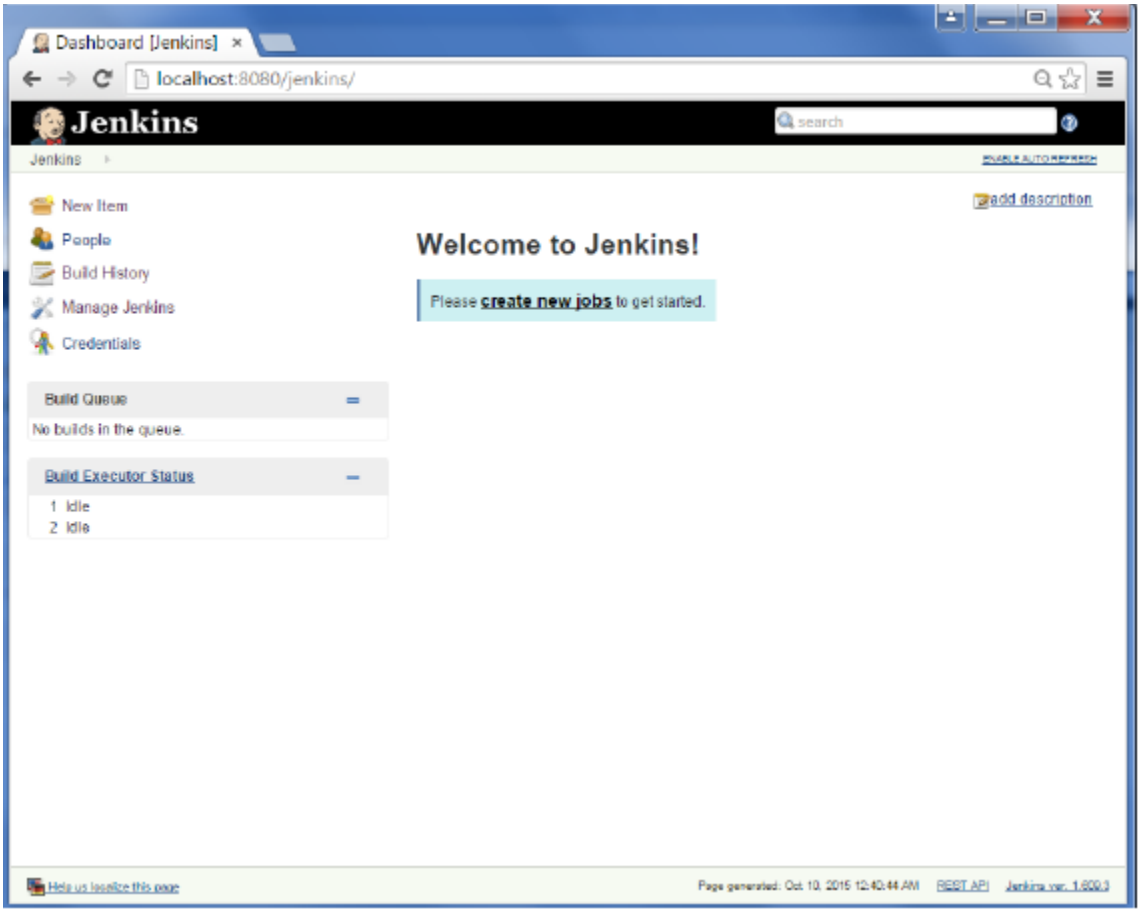
Accessing Jenkins

Once Jenkins is up and running, one can access Jenkins from the link − **http://localhost:8080**

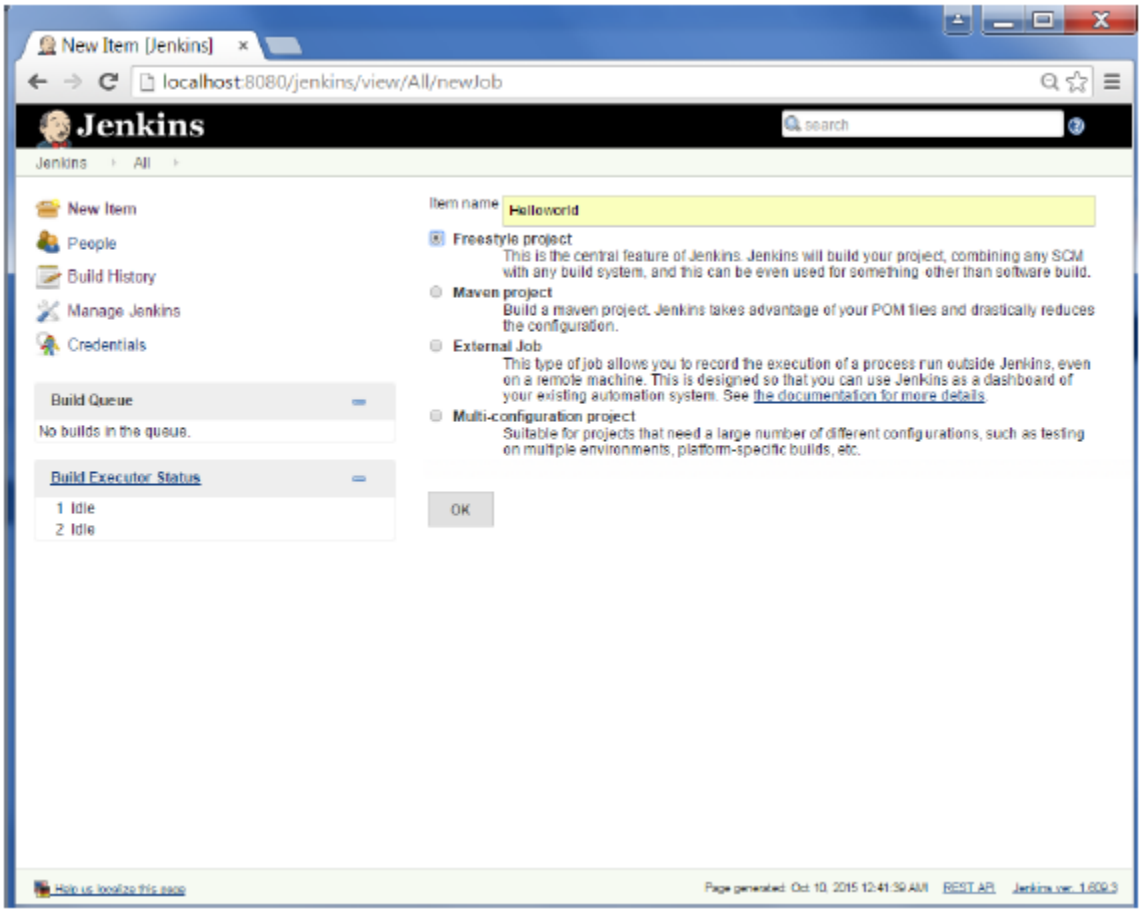
This link will bring up the Jenkins dashboard.

For this exercise, we will create a job in Jenkins which picks up a simple HelloWorld application, builds and runs the java program.

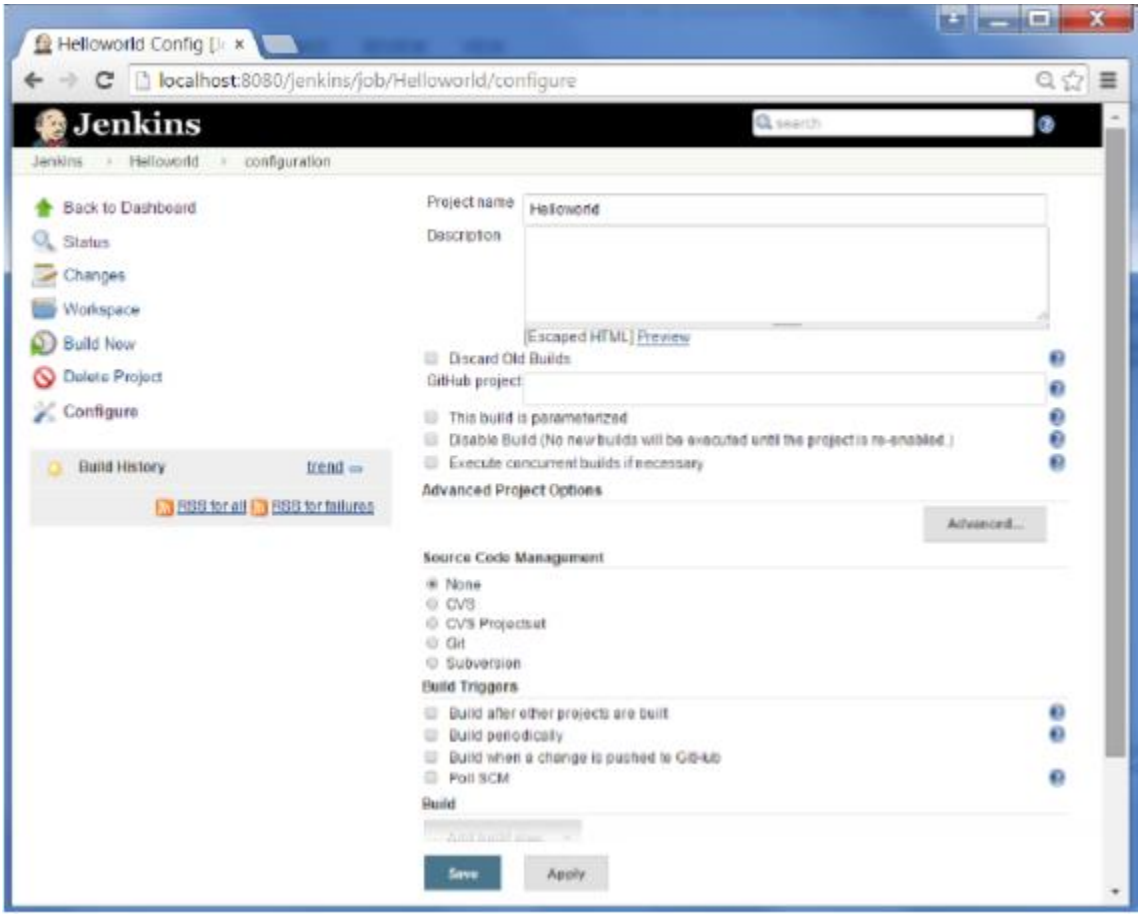
Step 1 − Go to the Jenkins dashboard and Click on New Item



Step 2 − In the next screen, enter the Item name, in this case we have named it Helloworld. Choose the 'Freestyle project option'

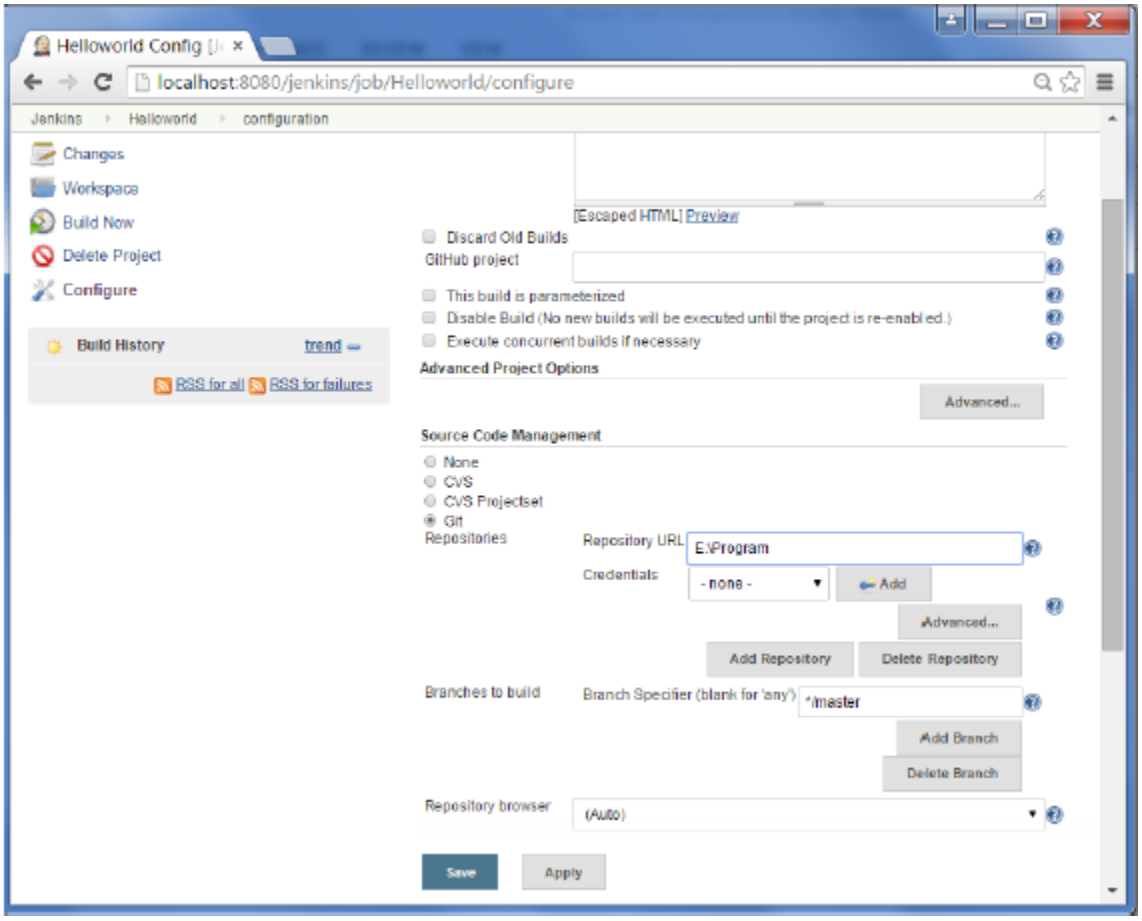


Step 3 − The following screen will come up in which you can specify the details of the job.

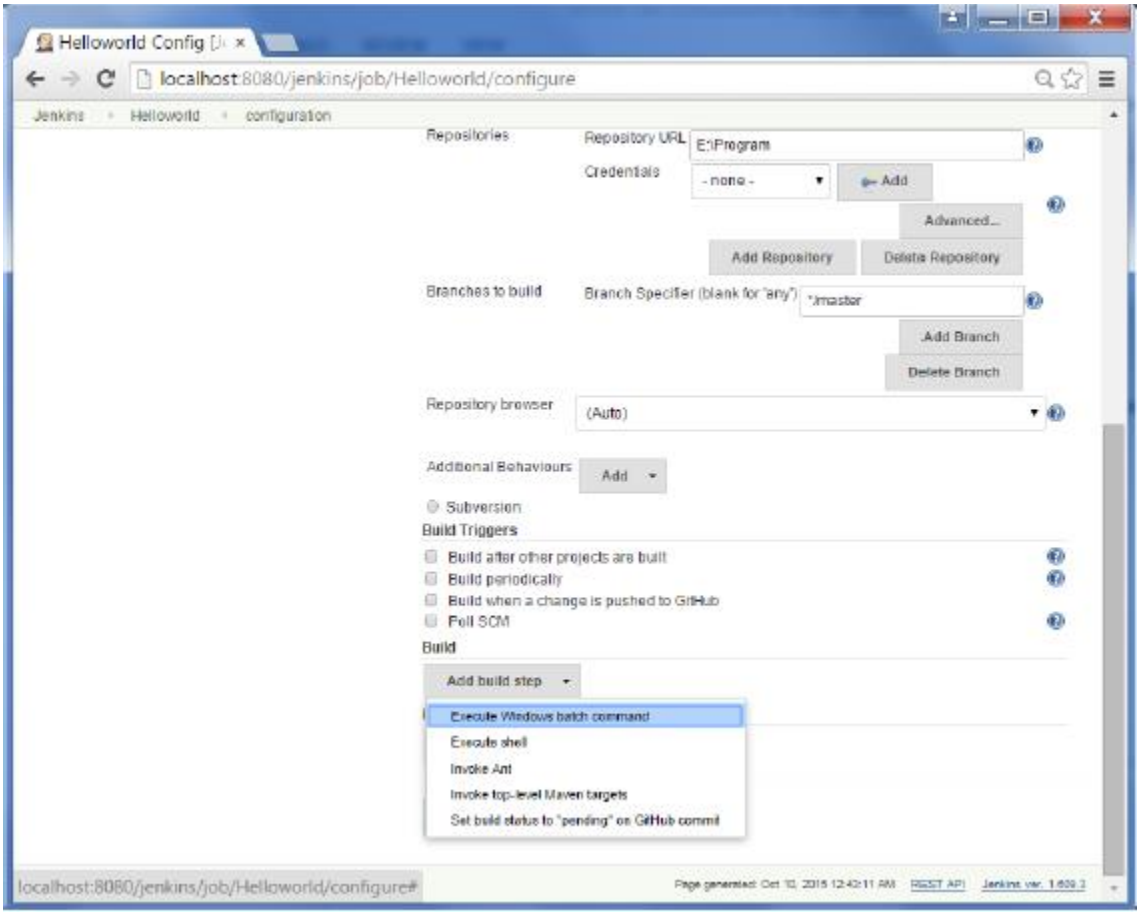


Step 4 − We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a 'HelloWorld.java' file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

Note − If you repository if hosted on Github, you can also enter the url of that repository here. In addition to this, you would need to click on the Add button for the credentials to add a user name and password to the github repository so that the code can be picked up from the remote repository.

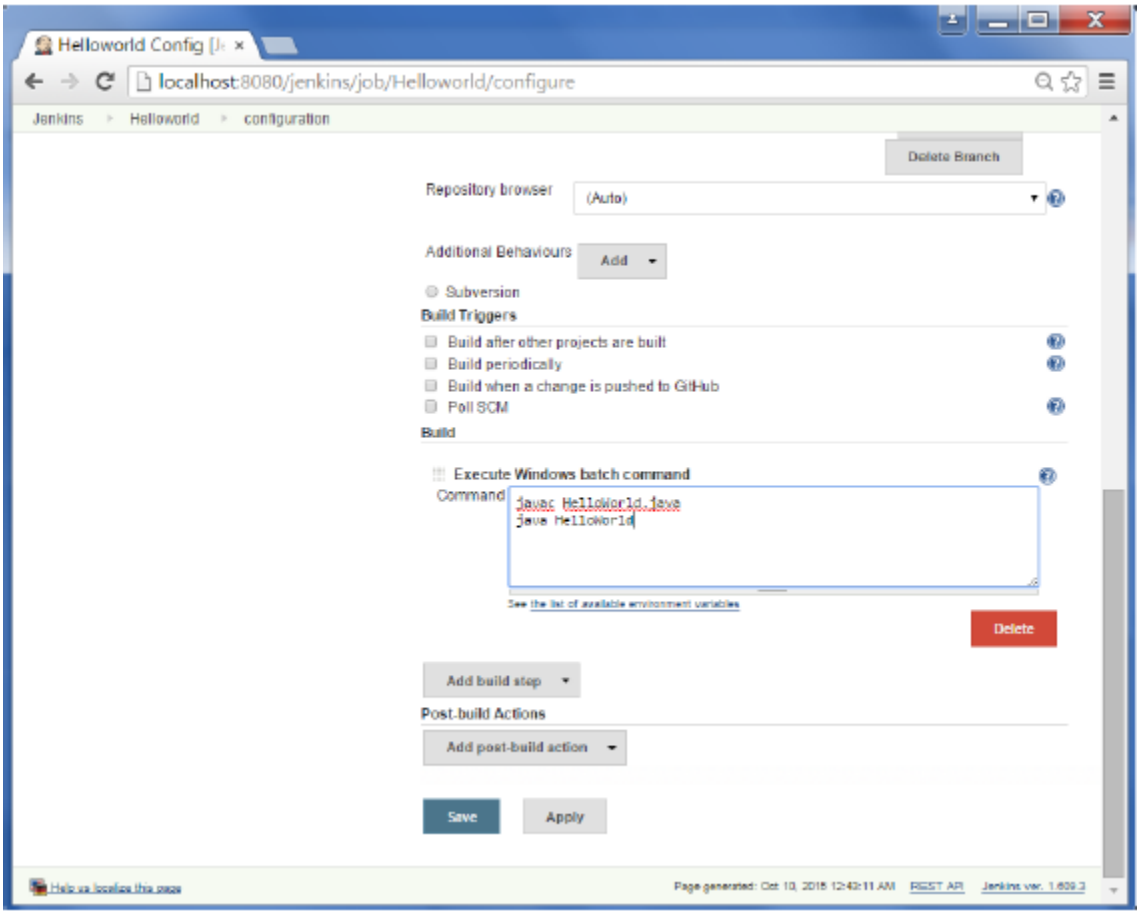


Step 5 − Now go to the Build section and click on Add build step → Execute Windows batch command

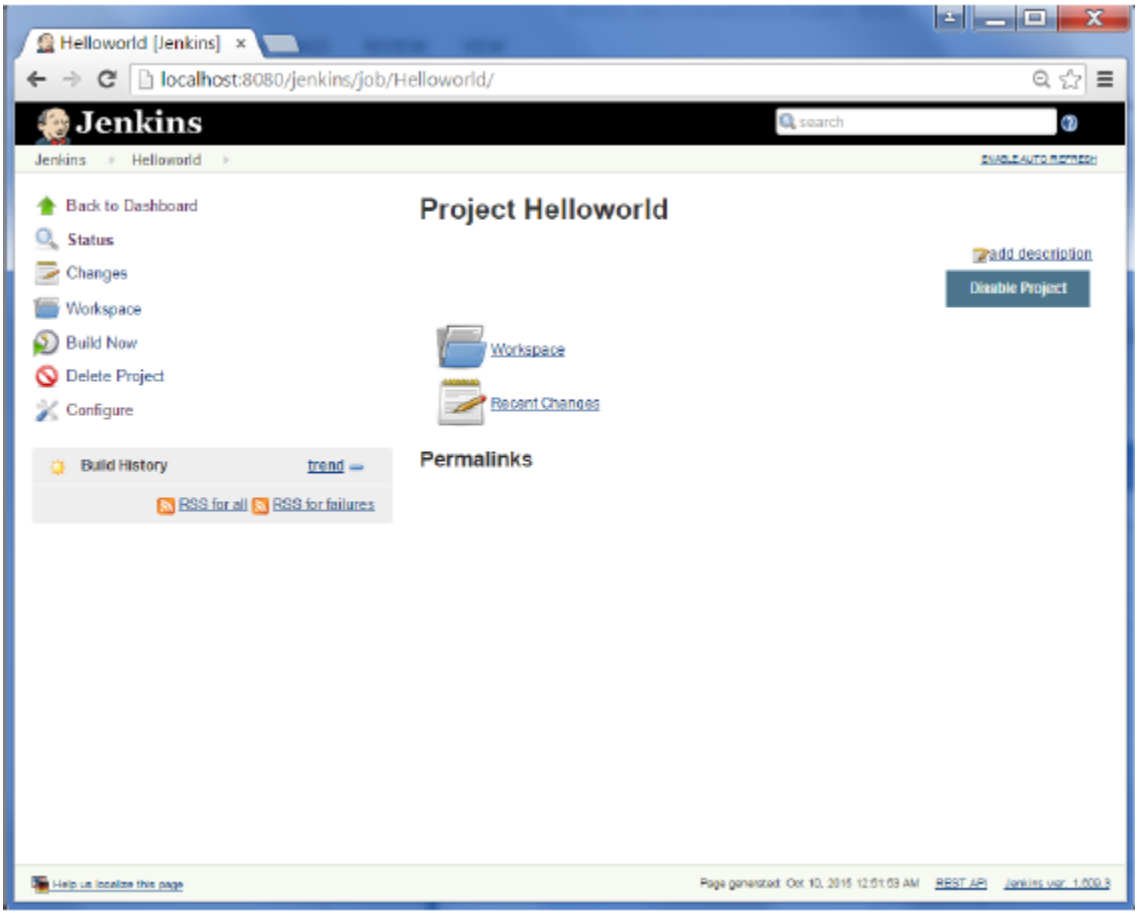


Step 6 − In the command window, enter the following commands and then click on the Save button.

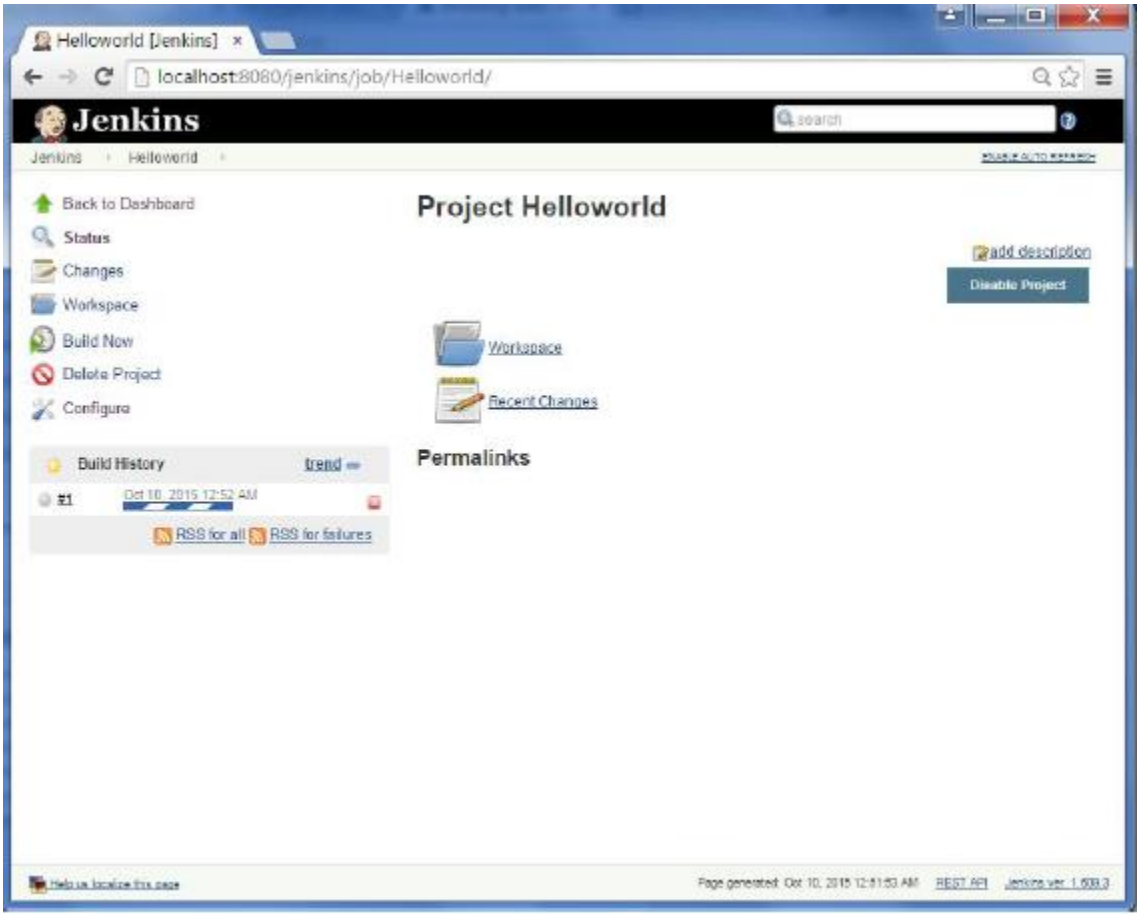
```
Javac HelloWorld.java
Java HelloWorld
```



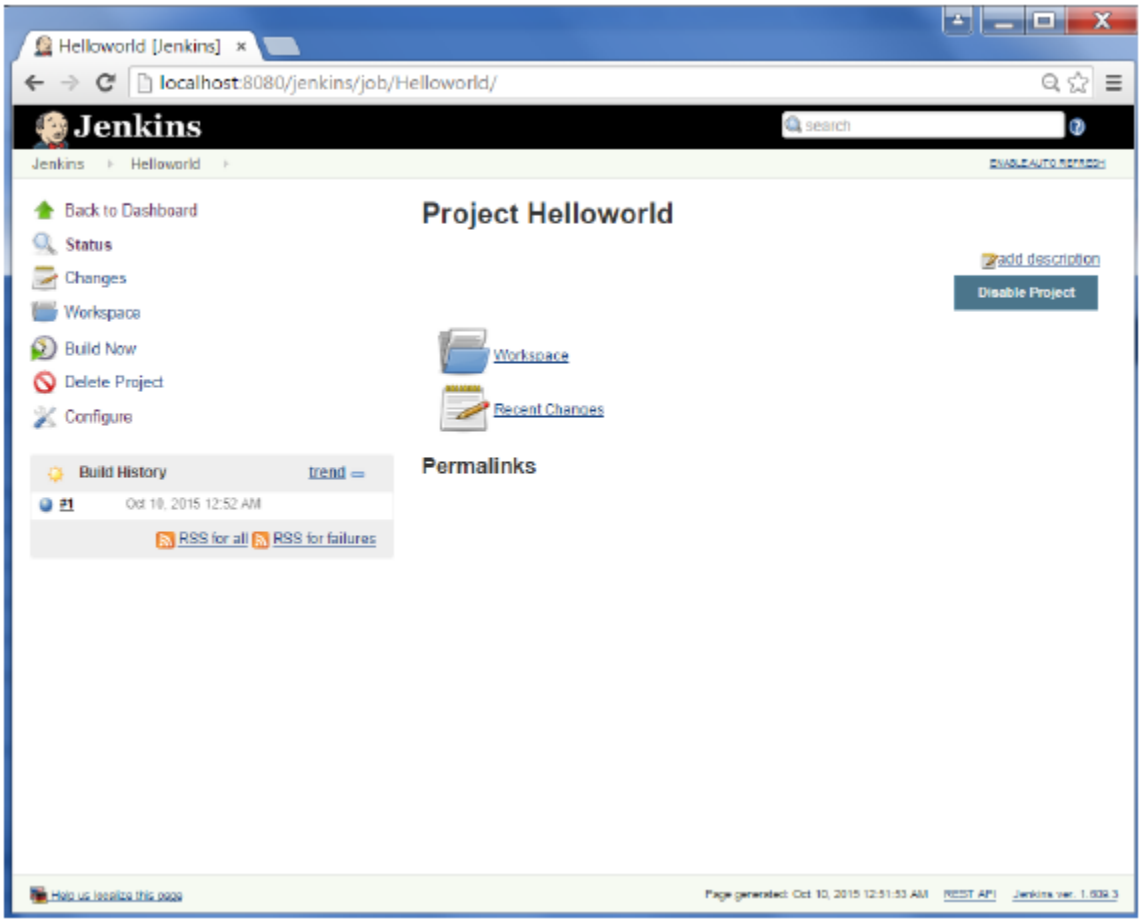
Step 7 − Once saved, you can click on the Build Now option to see if you have successfully defined the job.



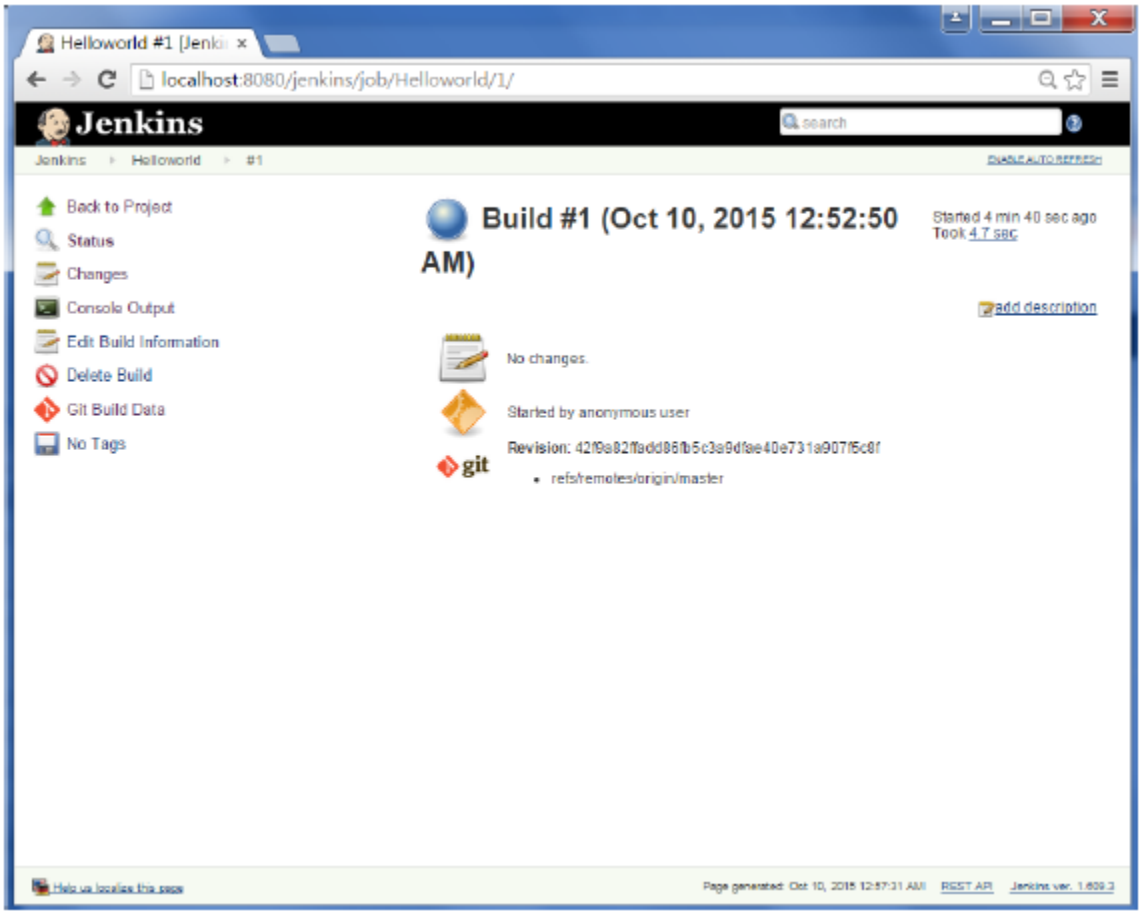
Step 8 − Once the build is scheduled, it will run. The following Build history section shows that a build is in progress.

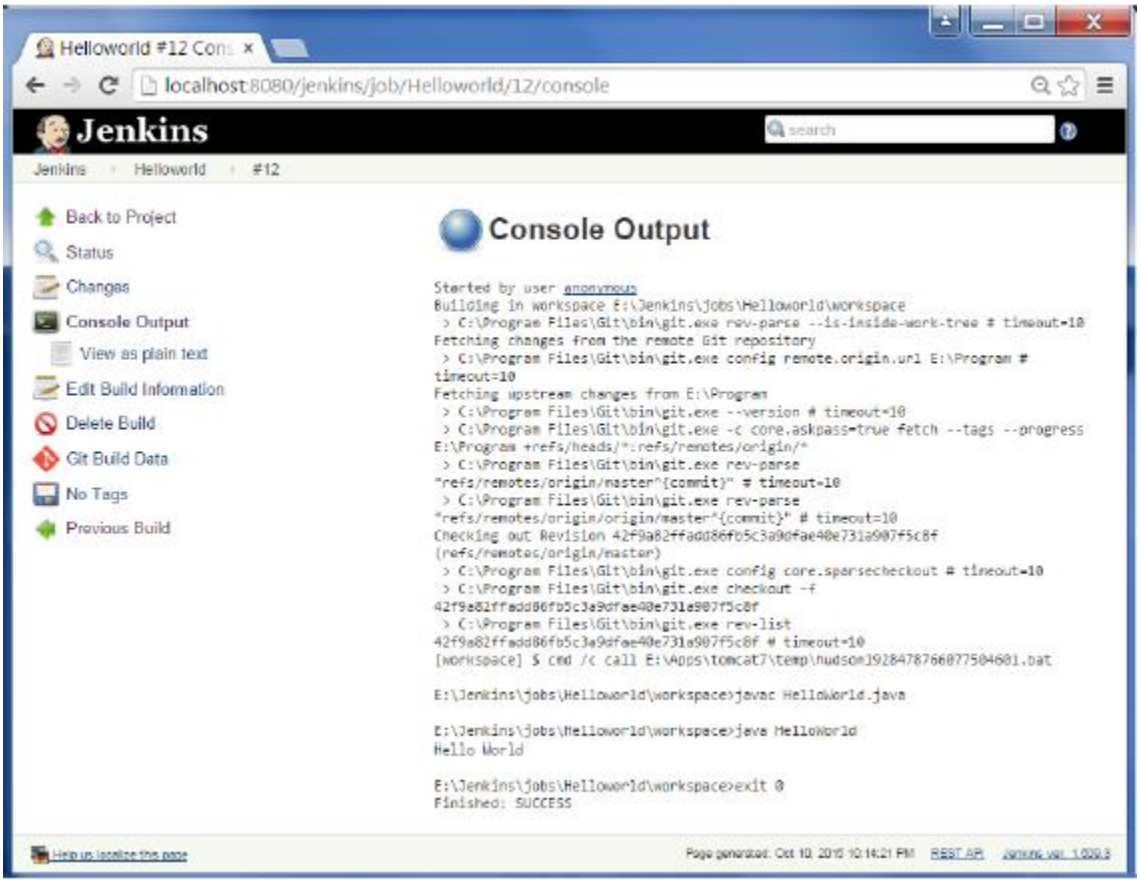


Step 9 − Once the build is completed, a status of the build will show if the build was successful or not. In our case, the following build has been executed successfully. Click on the #1 in the Build history to bring up the details of the build.



Step 10 − Click on the Console Output link to see the details of the build


Helloworld #1 [Jenki
localhost:8080/jenkins/job/Helloworld/1/
Jenkins
search
Jenkins
Helloworld
#1
Back to Project
Status
Changes
Console Output
Edit Build Information
Delete Build
Git Build Data
No Tags
Build #1 (Oct 10, 2015 12:52:50 AM)
Started 4 min 40 sec ago
Took 4.7 sec
add description
No changes.
Started by anonymous user
Revision: 42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
refs/remotes/origin/master
git
Help us localize this page
Page generated: Oct 10, 2015 12:57:31 AM
REST API
Jenkins ver. 1.609.3



Apart from the steps shown above there are just so many ways to create a build job, the options available are many, which what makes Jenkins such a fantastic continuous deployment tool.

Jenkins provides an out of box functionality for Junit, and provides a host of plugins for unit testing for other technologies, an example being MSTest for .Net Unit tests. If you go to the link https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin it will give the list of Unit Testing plugins available.

The following are some of the basic activities you will carry out, some of which are best practices for Jenkins server maintenance

URL Options

The following commands when appended to the Jenkins instance URL will carry out the relevant actions on the Jenkins instance.

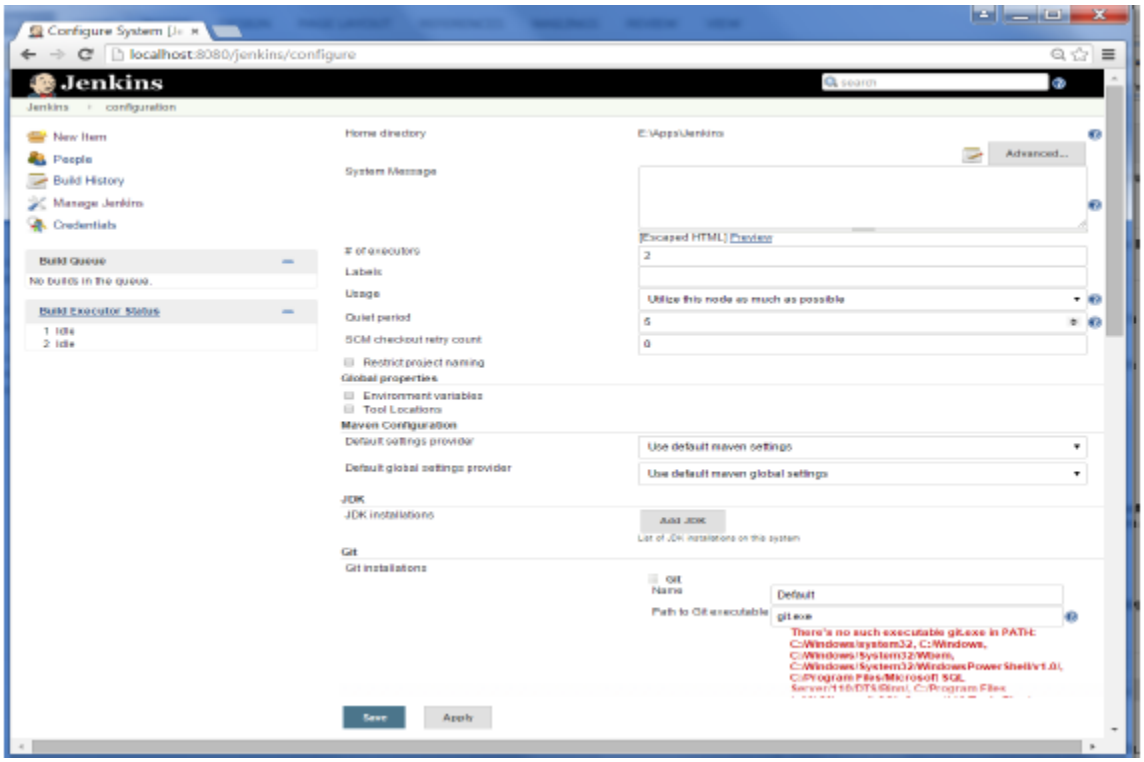
http://localhost:8080/jenkins/exit − shutdown jenkins

http://localhost:8080/jenkins/restart − restart jenkins

http://localhost:8080/jenkins/reload − to reload the configuration

Backup Jenkins Home

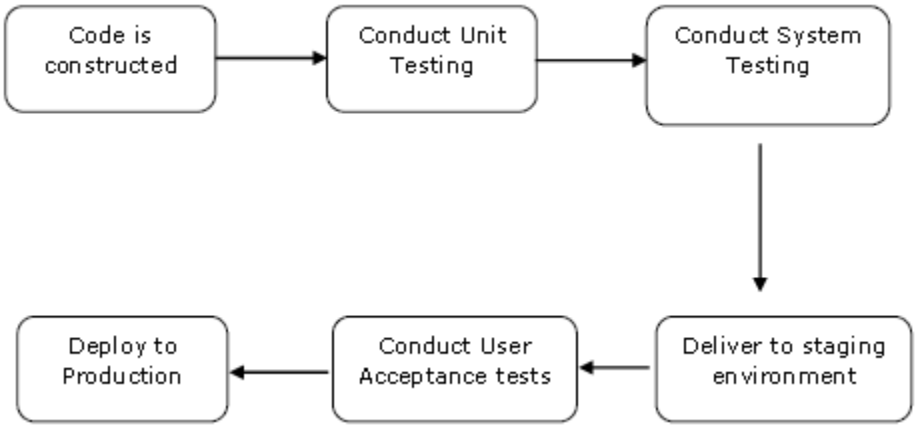
The Jenkins Home directory is nothing but the location on your drive where Jenkins stores all information for the jobs, builds etc. The location of your home directory can be seen when you click on Manage Jenkins → Configure system.



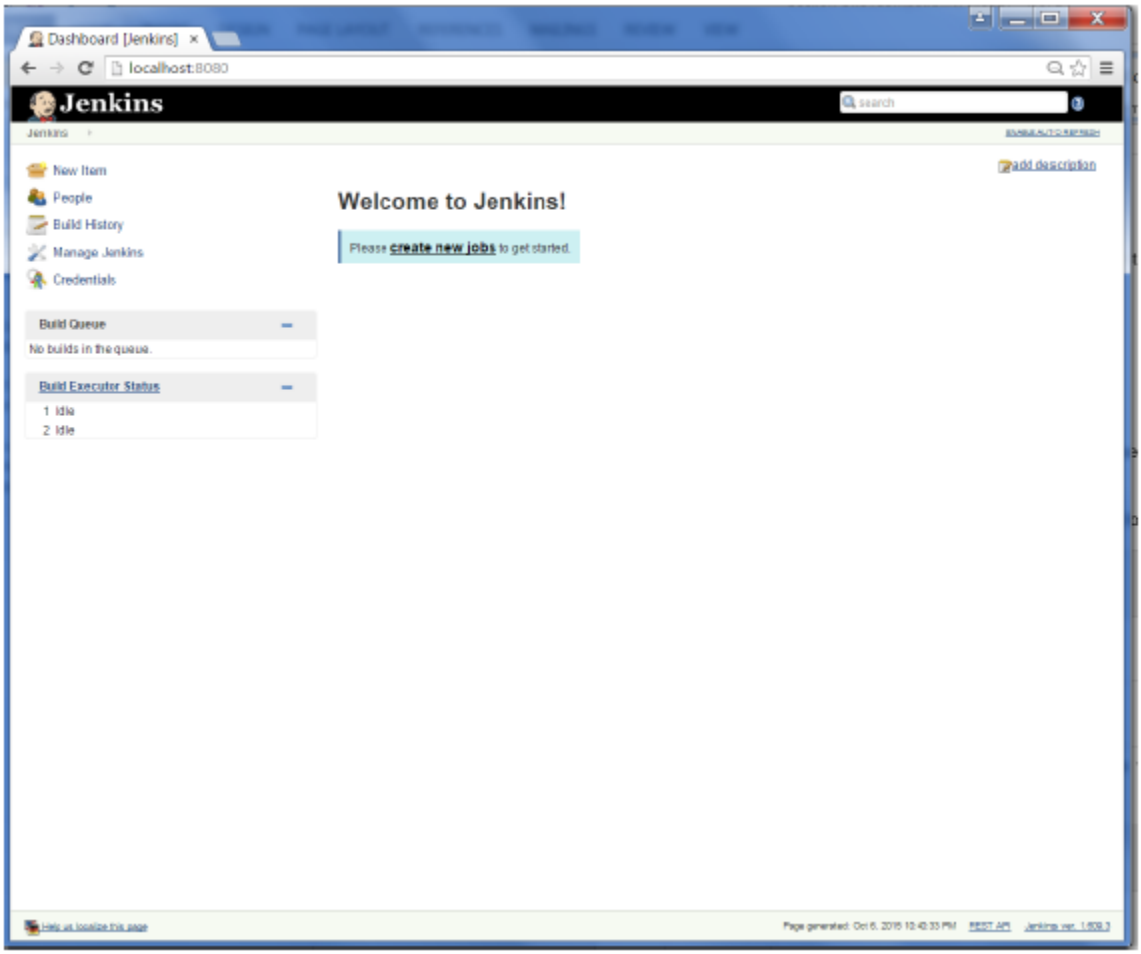
Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on the Jenkins instance will start failing.

Another best practice is to write cron jobs or maintenance tasks that can carry out clean-up operations to avoid the disk where Jenkins is setup from becoming full.

Jenkins provides good support for providing continuous deployment and delivery. If you look at the flow of any software development through deployment, it will be as shown below.



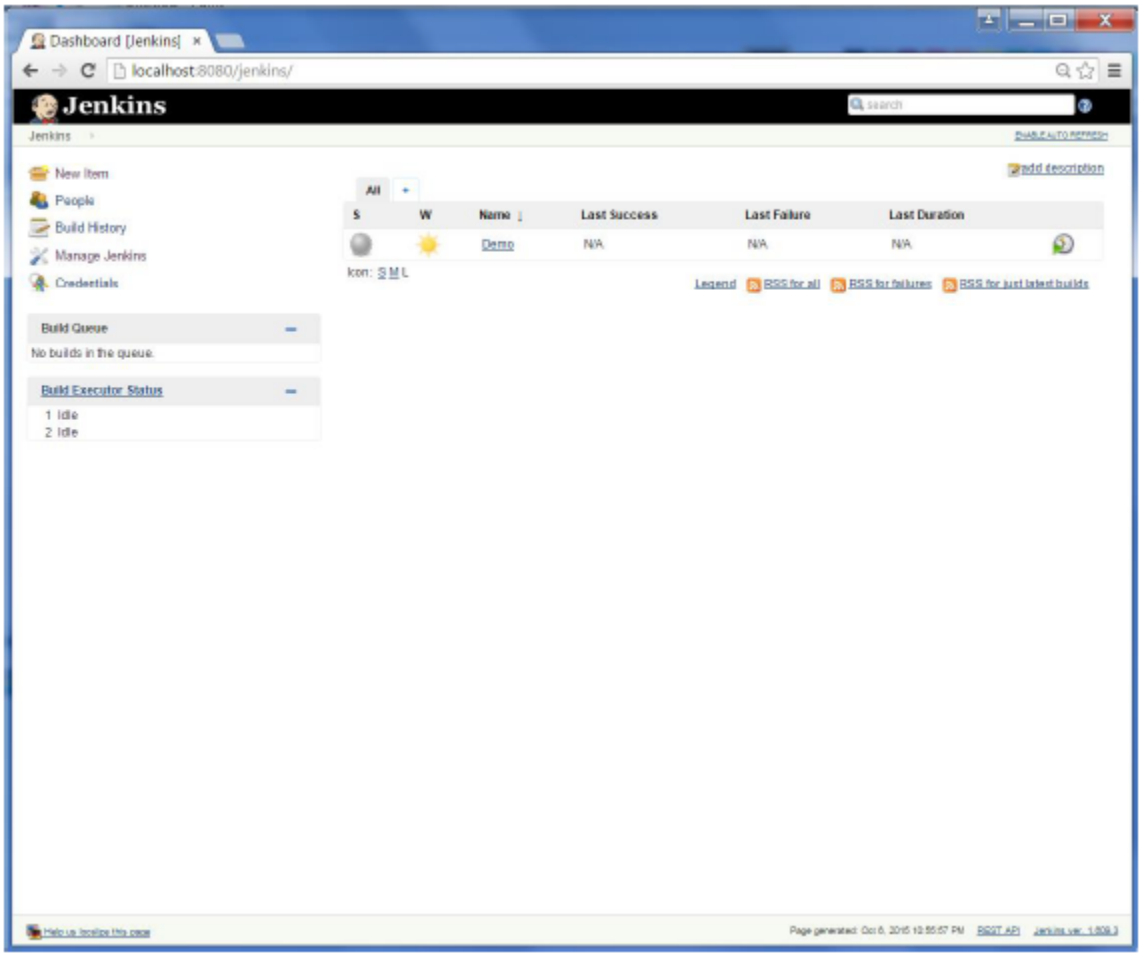
The main part of Continuous deployment is to ensure that the entire process which is shown above is automated. Jenkins achieves all of this via various plugins, one of them being the “Deploy to container Plugin” which was seen in the earlier lessons.



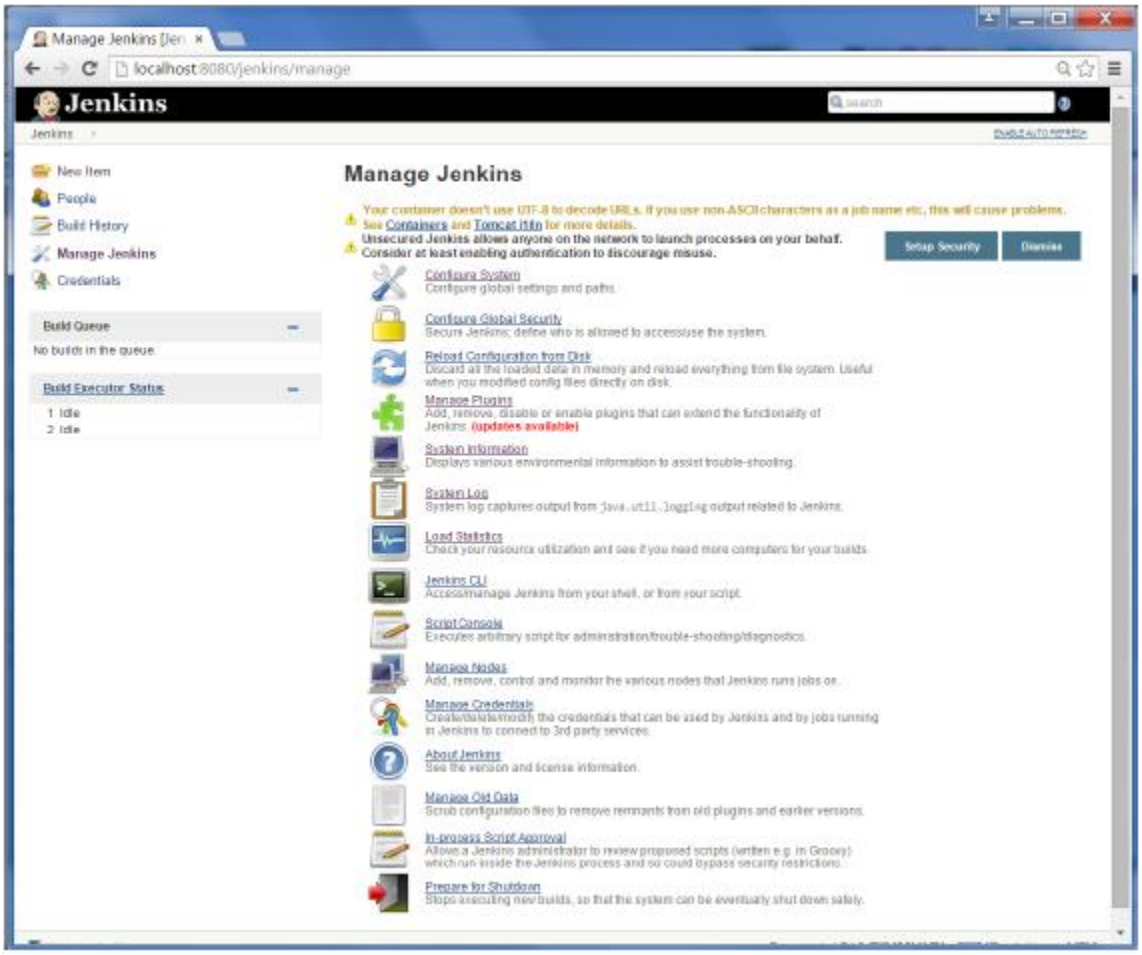
Jenkins - Management

To manage Jenkins, click on the ‘Manage Jenkins’ option from the left hand menu side.

So one can get the various configuration options for Jenkins by clicking the ‘Manage Jenkins’ option from the left hand menu side.



You will then be presented with the following screen −



Some of the management options are as follows −

Configure System

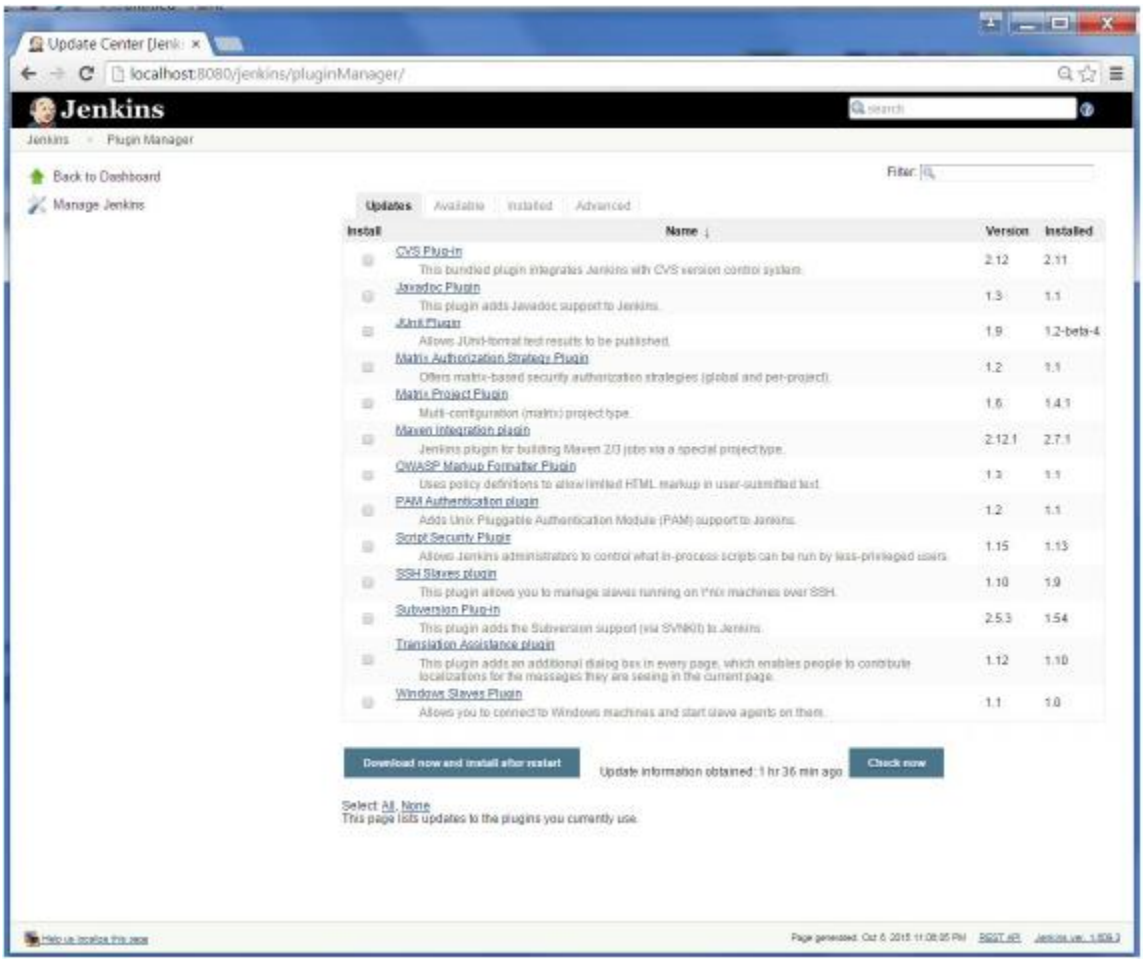
This is where one can manage paths to the various tools to use in builds, such as the JDKs, the versions of Ant and Maven, as well as security options, email servers, and other system-wide configuration details. When plugins are installed. Jenkins will add the required configuration fields dynamically after the plugins are installed.

Reload Configuration from Disk

Jenkins stores all its system and build job configuration details as XML files which is stored in the Jenkins home directory. Here also all of the build history is stored. If you are migrating build jobs from one Jenkins instance to another, or archiving old build jobs, you will need to add or remove the corresponding build job directories to Jenkins's builds directory. You don't need to take Jenkins offline to do this—you can simply use the "Reload Configuration from Disk" option to reload the Jenkins system and build job configurations directly.

Manage Plugin

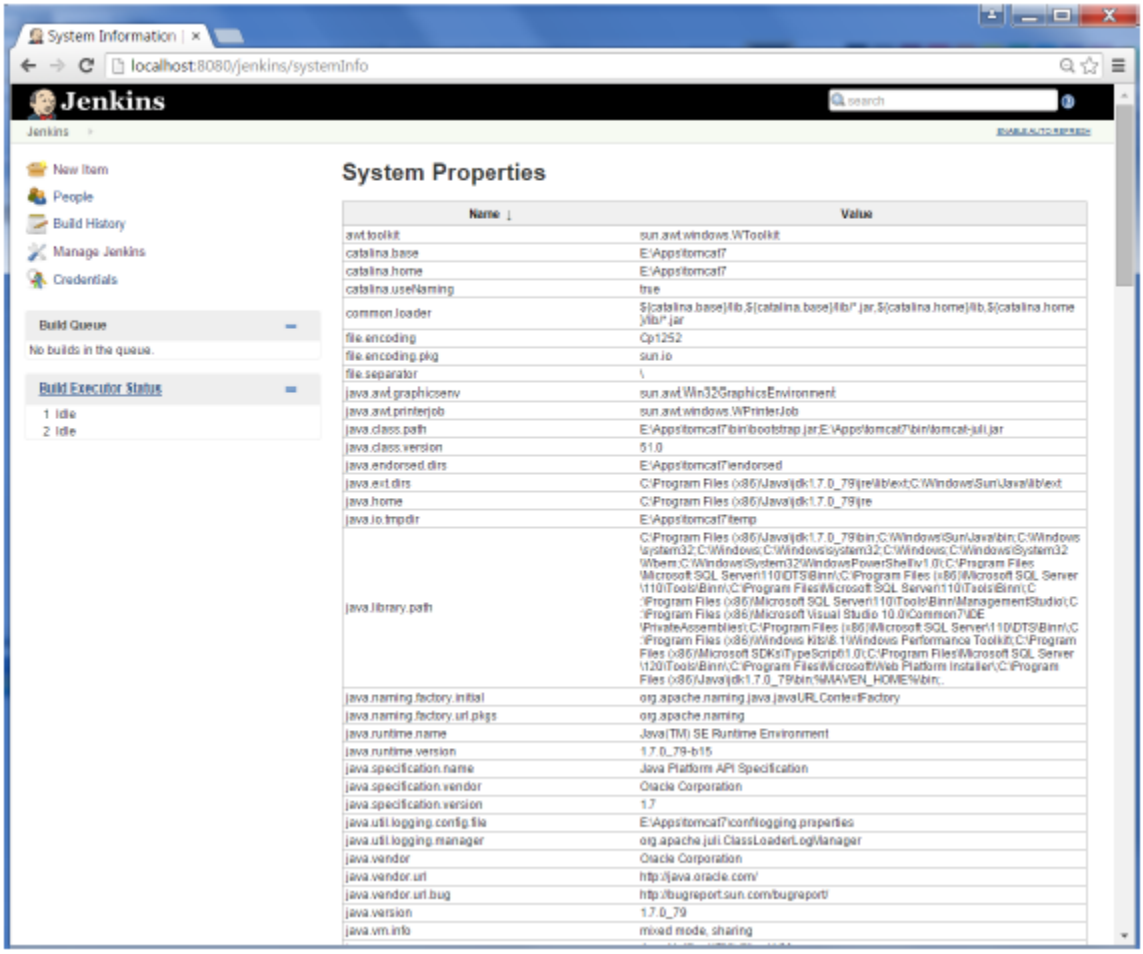
Here one can install a wide variety of third-party plugins right from different Source code management tools such as Git, Mercurial or ClearCase, to code quality and code coverage metrics reporting. Plugins can be installed, updated and removed through the Manage Plugins screen.



System Information

This screen displays a list of all the current Java system properties and system environment variables. Here one can check exactly what version of Java Jenkins is running in, what user it is running under, and so forth.

The following screenshot shows some of the name-value information available in this section.



System Log

The System Log screen is a convenient way to view the Jenkins log files in real time. Again, the main use of this screen is for troubleshooting.

Load Statistics

This pages displays graphical data on how busy the Jenkins instance is in terms of the number of concurrent builds and the length of the build queue which gives an idea of how long your builds need to wait before being executed. These statistics can give a good idea of whether extra capacity or extra build nodes is required from an infrastructure perspective.

Script Console

This screen lets you run Groovy scripts on the server. It is useful for advanced troubleshooting since it requires a strong knowledge of the internal Jenkins architecture.

Manage nodes

Jenkins is capable of handling parallel and distributed builds. In this screen, you can configure how many builds you want. Jenkins runs simultaneously, and, if you are using

distributed builds, set up build nodes. A build node is another machine that Jenkins can use to execute its builds.

Prepare for Shutdown

If there is a need to shut down Jenkins, or the server Jenkins is running on, it is best not to do so when a build is being executed. To shut down Jenkins cleanly, you can use the Prepare for Shutdown link, which prevents any new builds from being started. Eventually, when all of the current builds have finished, one will be able to shut down Jenkins cleanly.

Jenkins - Metrics & Trends

There are various plugins which are available in Jenkins to showcase metrics for builds which are carried out over a period of time. These metrics are useful to understand your builds and how frequently they fail/pass over time. As an example, let's look at the 'Build History Metrics plugin'.

This plugin calculates the following metrics for all of the builds once installed

- Mean Time To Failure (MTTF)
- Mean Time To Recovery (MTTR)
- Standard Deviation of Build Times

JUNIT AND MOCKITO

What is JUnit

JUnit is a unit testing framework for Java programming language. It plays a crucial role test-driven development, and is a family of unit testing frameworks collectively known as xUnit.

JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented

Features of JUnit

- JUnit is an open source framework, which is used for writing and running tests.
- Provides annotations to identify test methods.
- Provides assertions for testing expected results.
- Provides test runners for running tests.
- JUnit tests allow you to write codes faster, which increases quality.
- JUnit is elegantly simple. It is less complex and takes less time.
- JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.
- JUnit tests can be organized into test suites containing test cases and even other test suites.

What is a Unit Test Case

A Unit Test Case is a part of code, which ensures that another part of code (method) works as expected. To achieve the desired results quickly, a test framework is required. JUnit is a perfect unit test framework for Java programming language.

There must be at least two unit test cases for each requirement − one positive test and one negative test. If a requirement has sub-requirements, each sub-requirement must have at least two test cases as positive and negative.

JUnit Annotations

JUnit Annotations is a special form of syntactic meta-data that can be added to Java source code for better code readability and structure

@Test - This annotation is a replacement of org.junit.TestCase which indicates that public void method to which it is attached can be executed as a test Case.

@Before - This annotation is used if you want to execute some statement such as preconditions before each test case.

JUNIT AND MOCKITO

@BeforeClass - This annotation is used if you want to execute some statements before all the test cases for e.g. test connection must be executed before all the test cases.

@After - This annotation can be used if you want to execute some statements after each Test Case for e.g resetting variables, deleting temporary files ,variables, etc.

@AfterClass - This annotation can be used if you want to execute some statements after all test cases for e.g. Releasing resources after executing all test cases.

Testing exceptions in junit

we can simply use the expected attribute of the **@Test annotation** to declare that we expect an exception to be thrown anywhere in the annotated test method.

You can **use assertThrows()** , which allows you to test multiple exceptions within the same test.

Parameterized Test

Parameterized tests allow a developer to run the same test over and over again using different values. There are five steps that you need to follow to create a parameterized test.

- Annotate test class with @RunWith(Parameterized.class).
- Create a public static method annotated with @Parameters that returns a Collection of Objects (as Array) as test data set.
- Create a public constructor that takes in what is equivalent to one "row" of test data.
- Create an instance variable for each "column" of test data.
- Create your test case(s) using the instance variables as the source of the test data.

Mockito

What is Mocking

Mocking is a way to test the functionality of a class in isolation. Mocking does not require a database connection or properties file read or file server read to test a functionality. Mock objects do the mocking of the real service. A mock object returns a dummy data corresponding to some dummy input passed to it.

JUNIT AND MOCKITO

Benefits of Mockito

- **No Handwriting** – No need to write mock objects on your own.
- **Refactoring Safe** – Renaming interface method names or reordering parameters will not break the test code as Mocks are created at runtime.
- **Return value support** – Supports return values.
- **Exception support** – Supports exceptions.
- **Order check support** – Supports check on order of method calls.
- **Annotation support** – Supports creating mocks using annotation.

Verify calls on Mocks

Mockito verify() method can be used to test number of method invocations too. We can test exact number of times, at least once, at least, at most number of invocation times for a mocked method

Mockito – Spying

Mockito provides option to create spy on real objects. When spy is called, then actual method of real object is called.

Syntax

```
//create a spy on actual object
calcService = spy(calculator);

//perform operation on real object
//test the add functionality
Assert.assertEquals(mathApplication.add(20.0, 10.0),30.0,0);
```

Mockito JUnit Rules

Mockito JUnit Rule **helps keeping tests clean**. It initializes mocks, validates usage and detects incorrect stubbing. Make sure to configure your rule with strictness(Strictness) which automatically detects stubbing argument mismatches and is planned to be the default in Mockito

Difference between JUnit and Mockito

JUNIT AND MOCKITO

JUnit is a framework that helps with writing and running your unit tests. **Mockito** (or any other mocking tool) is a framework that you specifically use to efficiently write certain kind of tests. One core aspect in unit testing is the fact that you want to isolate your "class under test" from anything else **in the** world.

JUnit is the Java library used to write tests (offers support for running tests and different extra helpers - like setup and teardown methods, test sets etc.). Mockito is a library that enables writing tests using the mocking approach. JUnit is used to test API's in source code.

Hamcrest Matchers

Hamcrest is **a popular framework that help us to create the matcher objects**. It is used for writing software tests and also performs unit testing in Java programming language. Hamcrest is mainly used with other unit testing frameworks like JUnit, jMockit, Mockito, etc.

Mockito Annotations

@Mock: It is used to mock the objects that helps in minimizing the repetitive mock objects. It makes the test code and verification error easier to read as parameter names (field names) are used to identify the mocks.

@RunWith: It is a class-level annotation. It is used to keep the test clean and improves debugging. It also detects the unused stubs available in the test and initialize mocks annotated with @Mock annotation

@InjectMocks: It marks a field or parameter on which the injection should be performed. It allows shorthand mock and spy injections and minimizes the repetitive mocks and spy injection. In Mockito, the mocks are injected either by setter injection, constructor injection, and property injection.l

@Captor: It allows the creation of a field-level argument captor. It is used with the Mockito's verify() method to get the values passed when a method is called

@Spy - It allows the creation of partially mock objects. In other words, it allows shorthand wrapping of the field instances in a spy object.

PowerMock-Mockito

PowerMock is an open-source Java framework used for creating a mock object in unit testing.

The PowerMock framework provides a class called **PowerMockito** used to create mock objects and initiates verification and expectation. The PowerMockito provides the functionality to work with the Java reflection API.

what is maven?

- Maven is a project management and comprehension tool that provides developers a complete build lifecycle framework
- Maven helps the developer to create a java-based project more easily.
- Accessibility of new feature created or added in Maven can be easily added to a project in Maven configuration.
- It increases the performance of project and building process.
- The main feature of Maven is that it can download the project dependency libraries automatically.

how to create maven project?

Here we can create the maven project with two types .

->using terminal

->using eclipse

*By using terminal there is a command to create a maven project that is

#command

C:\MVN>mvn archetype:generate #here archetype is the pulgin and generate is the goal ,this create the project with parameters

-DgroupId = com.package #here you can write the package name

-DartifactId = firstprogram #here filename

-DarchetypeArtifactId = maven-archetype-quickstart #here type of archetype

-DinteractiveMode = false #this should be initiale false

commands that are used in the maven project?

Mvn –version: This command helps us in knowing the current version of Maven that is installed

Creating a project: To create a project using MVN architecture below maven command should be used.

mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart

-DarchetypeVersion=1.4 -DinteractiveMode=false

MVN package: This command is used to execute all Maven phases until the package phase. It does the job of compiling,

verifying and building the project.It builds the jar file and places it in the specified folder under the specified project.

mvn clean install: This maven command helps in executing a clean build life cycle and installs build phase in the default build cycle.

This build life cycles may have its build phases and inside each build, there are different build goals.

Also, this ensures that the build target is being removed for a new build and adds the clean target.

mvn compile: This command is used to compile the source code. It also compiles the classes that are stored at a particular target or class.

mvn test: Maven also provides the facility of unit testing particular codes. It runs the tests using suitable

MICROSERVICES

History of Microservices

- The microservices style of architecture develops complex application software from small, individual applications that communicate with each other using language-independent interfaces (APIs).
- Companies run into trouble if they are unable to scale monolithic architecture that has developed over time, if their architecture is difficult to upgrade or maintenance becomes too complex.
- Microservices can be the answer to this problem, as they break down complex tasks into smaller processes that work independently of each other.
- **Central services**—Handle business data persistence and apply business rules and other logic.
- **Composite services**—Organize either a number of central services in order to fulfill a common task, or aggregate information from several central services.

Problems with Monolith & SOA

- Service-oriented architecture (SOA) is an enterprise-wide approach to software development of application components that takes advantage of reusable software components, or services.
- In SOA software architecture, each service is comprised of the code and data integrations required to execute a specific business function — for example, checking a customer's credit, signing into a website or processing a mortgage application.
- The service interfaces provide loose coupling, which means that they can be called with little or no knowledge of how the integration is implemented underneath.
- Because of this loose coupling and the way the services are published, development teams can save time by reusing components in other applications across the enterprise.

MICROSERVICES

Microservices Architecture

- Microservices architecture (often shortened to microservices) refers to an architectural style for developing applications.
- Microservices allow a large application to be separated into smaller independent parts, with each part having its own realm of responsibility.
- To serve a single user request, a microservices-based application can call on many internal microservices to compose its response.
- Containers are a well-suited microservices architecture example, since they let you focus on developing the services without worrying about the dependencies.
- Modern cloud-native applications are usually built as microservices using containers.

Problems Solved by Microservices

- Before going any further, let me offer a specific definition.
- After all, relying on vague, hand-waving definitions is the main culprit in buzzword fatigue.
- I certainly don't want to contribute to that.
- Imagine an organization with a large, monolithic website.
- This sprawling entity serves as the company's entire public interface on the internet.

MICROSERVICES

Designing Microservices Architecture

- Imagine an organization with a large, monolithic website.
- This sprawling entity serves as the company's entire public interface on the internet.
- I'm a Software Architect at Software Mill and I've been designing microservices implementations for the past few years. They are not a perfect solution to every problem, sometimes a well structured monolith is better.
- How to investigate if your system would benefit from microservices without jumping right away into a binary approach? Here are my lessons learned and thoughts on microservices.

Testing Microservices

- In this article, I will share my experience in **microservices testing**.
- Most of the companies are implementing the microservices architecture to have the capability to develop, test, and deploy the services independently and faster.
- Of course, these architectures come with a lot of complexities and in order to test these systems effectively, we need to be aware of the system architecture very well.
- Let's start with a simple architectural view of microservices.
- Generally, we have clients/channels/consumers such as Web, Mobile Web, Mobile Apps (iOS and Android), and Desktop.
- We may have some downstream or external services which do business-critical operations such as loyalty operations, customers' data-related operations, and they may hold some critical data of the business.
- These operations and data depend on the company's sector.

MICROSERVICES

Logging and Monitoring

- Logging and monitoring are both valuable components to maintaining optimal application performance.
- Using a combination of logging tools and real-time monitoring systems helps improve observability and reduces the time spent sifting through log files to determine the root cause of performance problems.
- Logging is used as both a verb and a noun, referring either to the practice of logging errors and changes or to the application logs that are collected.
- The purpose of logging is to create an ongoing record of application events

When not to Use Microservices

- Microservices are solutions to complex concerns and if your business doesn't have complex issues, understand that you don't have a system in place to handle the complexities of microservices.
- Using microservices can prove to offer contrary consequences if you don't have a team size that cannot handle the tasks involved. This will only result in the delay of delivery.
- Implementing microservices for the sake of it can be hampering as well. If your application does not require to be broken down into microservices, you don't need this.
- There is no absolute necessity that all applications should be broken down to microservices. There are those that are simple by their nature and functionality

Microservices and the Organization

- Microservices are often advertised as an architecture style/pattern. What is hardly ever mentioned is that the organization needs to be supportive of this style as well.
- The main takeaway is that Microservices is not just an architecture style, but also an organizational structure.

MICROSERVICES

- Adopting the one without the other is like buying new running shoes and wearing them just to watch Netflix.
- Microservices do offer a solution to this caveat and they do so by organizing around business functionalities rather than technologies.
- The old Unix philosophy 'do one thing and do it well' certainly applies to Microservices where the aim is to build smaller and clearer bits of functionality that are functioning very well.

Anti-Patterns and Common Mistakes

- An **anti-pattern** is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.

- The term, coined in 1995 by computer programmer Andrew Koenig was inspired by the book *Design Patterns*, which highlights a number of design patterns in software development that its authors considered to be highly reliable and effective.

- An **anti-pattern** is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.

- The term, coined in 1995 by computer programmer was inspired by the book, which highlights a number of in software development that its authors considered to be highly reliable and effective.

- The term was popularized three years later by the book which extended its use beyond the field of software design to refer informally to any commonly reinvented but bad solution to a problem.

- Examples include analysis_paralysis, cargo_cult_programming, death march, groupthink and vendor_lock - in

- **Common Microservices Anti-Patterns**

Some of the common anti-patterns include *Break the Piggy Bank*, *Cohesion Chaos*, *Versioning Avoidance*, *Gateway for each service*, *Everything Micro*, and so forth. In the sections that follow, I will walk you through the most common anti-patterns and pitfalls when working with microservices-based applications, and solutions for avoiding them.

- **Distributed Monolith**

This is one of the biggest concerns when you split an existing monolithic application into microservices. If a service needs all other microservices or libraries to be available to execute, that's a distributed monolith. The choice between a microservices architecture and a distributed monolith depends on several factors. That's the million-dollar question. To answer this question, we will need to be able to answer a few questions that follow.

First off, will a change in a microservices in your application require redeployment of other microservices of the application as well? If so, your microservices are not decoupled from each other. This breaks the basic purpose of adopting microservices architecture – a major benefit of microservices architecture is decoupling such that services can be independently developed, tested, deployed, and maintained. Finally, you have a closely connected architecture, which results in delivery entropy and cohesion chaos, as well as other problems.

Breaking Monolith to Microservices

- Follow these tips from two experts, both of whom presented at the 2020 O'Reilly Software Architecture Conference in New York, to break up messy

monoliths into tidy microservices -- or discover where to leave the monolith in place.

Step 1: Establish an application architecture baseline

- To break apart a monolith, you first have to know what's in it. At the start of any microservices project, segment the existing monolith's architecture by logical components, such as the namespaces.

- Components, in this context, are any sets of classes that do one thing, said Mark Richards, an independent consultant. "You don't have to move individual classes," he said.

Step 2: Refactor into a domain-driven design

- To progress toward microservices, assess the size of the namespaces identified as future services.

- If any single component in the architecture contains more than 10% of the overall application, break it down into smaller pieces, Richards said.

- He bases this on statements, rather than lines of code, since code volume varies from one developer to another.

Step 3: Build microservices

- Microservices are pieces of single-purpose code that do one thing well. From tens of business or functional services, an application architecture can split into hundreds or thousands of microservices.

- Start your work at points where the application will benefit the most from microservices -- ideally, where automation, DevOps and containerization are already in place.

- If possible, tackle a high-change but low-risk component, Richards said, such as nonfinancial, customer-facing functionality.

- Using the strategies described above, like dark launches, teams increase the chance that they can refactor without negative consequences on the user base.

SPRING FRAMEWORK

Introduction to spring framework

Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications. Spring handles the infrastructure so you can focus on your application.

Spring enables you to build applications from “plain old Java objects” (POJOs) and to apply enterprise services non-invasively to POJOs. This capability applies to the Java SE programming model and to full and partial Java EE.

Examples of how you, as an application developer, can use the Spring platform advantage:

- Make a Java method execute in a database transaction without having to deal with transaction APIs.
- Make a local Java method a remote procedure without having to deal with remote APIs.
- Make a local Java method a management operation without having to deal with JMX APIs.
- Make a local Java method a message handler without having to deal with JMS APIs.

Basic tools and frameworks in JUnit

JUnit tutorial provides basic and advanced concepts of **unit testing in java** with examples. Our junit tutorial is designed for beginners and professionals.

It is an *open-source testing framework* for java programmers. The java programmer can create test cases and test his/her own code.

It is one of the unit testing framework. Current version is junit 4.

To perform unit testing, we need to create test cases. The **unit test case** is a code which ensures that the program logic works as expected.

Basic tools and frameworks in Mockito

The mocking technique is not only used in Java but also used in any object-oriented programming language. There are many frameworks available in Java for mocking, but Mockito is the most popular framework among them.

To mock objects, you need to understand the three key concepts of mocking, i.e., stub, fake, and mock. Some of the unit tests involve only stubs, whereas some involve fake and mocks.

The brief description of the mocking concepts is given below:

1. **Stub:** Stub objects hold predefined data and provide it to answer the calls during testing. They are referred to as a dummy object with a minimum number of methods required for a test. It also provides methods to verify other methods used to access the internal state of a stub, when necessary. Stub object is generally used for **state verification**.
2. **Fake:** Fake are the objects that contain working implementations but are different from the production one. Mostly it takes shortcuts and also contains the simplified version of the production code.
3. **Mock:** Mock objects act as a dummy or clone of the real object in testing. They are generally created by an open-source library or a mocking framework like Mockito, Easy Mock, etc. Mock objects are typically used for **behavior verification**.

Unit Testing with Spring Framework

Dependency injection should make your code less dependent on the container than it would be with traditional Java EE development. The POJOs that make up your application should be testable in JUnit or Testing tests, with objects instantiated by using the new operator, without Spring or any other container. You can use mock objects (in conjunction with other valuable testing techniques) to test your code in isolation. If you follow the architecture recommendations for

SPRING FRAMEWORK

Spring, the resulting clean layering and componentization of your codebase facilitate easier unit testing. For example, you can test service layer objects by stubbing or mocking DAO or repository interfaces, without needing to access persistent data while running unit tests.
True unit tests typically run extremely quickly, as there is no runtime infrastructure to set up. Emphasizing true unit tests as part of your development methodology can boost your productivity. You may not need this section of the testing chapter to help you write effective unit tests for your IoC-based applications. For certain unit testing scenarios, however, the Spring Framework provides mock objects and testing support classes.

Spring AOP

Aspect Oriented Programming (AOP) compliments OOPs in the sense that it also provides modularity. But the key unit of modularity is aspect than class.

It provides the pluggable way to dynamically add the additional concern before, after or around the actual logic. Suppose there are 10 methods in a class as given below:

There are 5 methods that starts from m, 2 methods that starts from n and 3 methods that starts from p.

Understanding Scenario, I have to maintain log and send notification after calling methods that starts from m.

Problem without AOP We can call methods (that maintains log and sends notification) from the methods starting with m. In such scenario, we need to write the code in all the 5 methods.

But, if client says in future, I don't have to send notification, you need to change all the methods. It leads to the maintenance problem.

Solution with AOP We don't have to call methods from the method. Now we can define the additional concern like maintaining log, sending notification etc. in the method of a class. Its entry is given in the xml file.

In future, if client says to remove the notifier functionality, we need to change only in the xml file. So, maintenance is easy in AOP.

AOP is mostly used in following cases:

- to provide declarative enterprise services such as declarative transaction management.
- It allows users to implement custom aspects.

Spring JDBC

The main tool that Spring JDBC uses for querying is the JdbcTemplate. The downside of using this is that it only provides the connection, everything else you need to do yourself. If you search for objects, you will need to map the results to Java objects by implementing a RowMapper. You will also need to do the exception handling by creating a ExceptionTranslator.
I will show you a simple example of how a query is created.

Spring data JDBC

Spring Data JDBC uses a syntax that is comparable to Spring Data JPA. The biggest differences are under the hood. The management of the persistence is handled by the repository like in Spring Data JPA, but only the aggregate root has a repository. This means that if you want to insert or update data, the entire aggregate needs to be saved. You will need to call the save method of the repository of the aggregate root and this will first save the aggregate root and then all of the referenced entities get saved. If you want to insert only a part of an aggregate, for example only create a new Rental, then the whole aggregate will be updated and the referenced entities will be deleted and inserted again.

SPRING DATA JPA

When you use the Spring Data framework, it will help you with building your queries and fetching the right data. The Spring Data JPA framework uses

implementations of the JPA specifications like Hibernate. They make it possible to query the database using user friendly interfaces. When you want to query the database, instead of writing the entire query yourself, Hibernate will help you. There are multiple ways to query the database using Spring Data JPA, but they all need you to extend the repository of the entity you want to query.

Some basic queries can be written using derived queries. An example of this is findById. For these methods Spring Data will generate the SQL entirely on its own. For example paging and sorting can be done by simply adding a parameter.

Basic tools and Frameworks in Maven

When we develop a Java software project, such as a microservice, the major dependencies we need are:

- Spring MVC framework.
- Spring framework
- Hibernate

In addition to these 3 direct dependencies, we need to include their individual dependencies as well. So we need the dependencies of Spring MVC, Spring and Hibernate.

Let's consider an example: REST API typically returns JSON responses. Spring MVC needs to convert Java Bean to JSON. It makes use of the Jackson framework to do the conversion. Jackson framework is a dependency of Spring MVC.

Dependency Management With Maven

Identifying Artifacts

In order to describe each of the artifacts stored here, Maven uses two tags - and. In addition to that, there is also a version that is maintained for each artifact. using these 3 identifiers, Maven can retrieve the correct version of the required dependency.

Managing Transitive Dependencies

When Maven downloads a particular artifact, it also downloads those dependencies of this artifact that are needed in the project. These are known as **transitive dependencies**. This reduces the amount of effort that is spent on enumerating all the artifacts.

Maven Does A Lot More!

The truth is in the Java world, Maven does a lot more than just dependency management.

Building Deployable unit - Jars and Wars

Suppose we want to deploy an application to a different environment. We do not want to take the source code and build it again!

We can create an application deployable unit JAR/WAR/EAR and use it in other environments. Maven enables us to create application deployable unit in a simple way.

This entire sequence is called a **build** process, thereby making Maven a **build tool** as well.

JAVA WEB DEVELOPMENT

Basics of Web Application

Java Web Application is used **to build dynamic websites**. Java offers support for the web application through JSPs and Servlets. We can build a website with static HTML web pages but when we want data to be dynamic, we require the web application.

A Web application **contains an application's resources, such as servlets, JavaServer Pages (JSPs), JSP tag libraries, and any static resources such as HTML pages and image files**. A Web application can also define links to outside resources such as Enterprise JavaBeans (EJBs).

Static and Dynamic web application

A static website is a website whose web pages are coded in HTML and the content of each page is fixed and does not change unless it is edited and republished. A dynamic website is a website whose web pages are generated in real time.

server-side programming

Server-side programming **allows developers to make use of sessions** — basically, a mechanism that allows a server to store information on the current user of a site and send different responses based on that information

Servlet Life Cycle Methods

There are three life cycle methods of a Servlet :

- init() -This method is called by the Servlet container to indicate that this Servlet instance is instantiated successfully and is about to put into service.
- service() - This method handles multiple client request and sends response back.
- destroy() - This method runs only once during the lifetime of a Servlet

Servlets - Annotations

@WebServlet - To declare a servlet.

@WebInitParam - To specify an initialization parameter.

@WebFilter - To declare a servlet filter.

@WebListener - To declare a WebListener

Flow of Execution of Servlets

- Create "*Dynamic web application"* project in Eclipse.
- Create "*index.html"* file to take input from the client in the request.
- Create "*ServletName.java"* to process the request and generate the response.
- Create "*web.xml"* file to specify the URL mapping to the respective servlet.

JDBC Architecture

- JDBC architecture is **an API specifying interfaces for accessing relational databases**.
- JDBC helps to connect to a database, send queries and updates to the database, and retrieve and process the results obtained from the database for queries
- JDBC API provides universal data access from the Java programming language. Using the **JDBC API**, you can access virtually ...

JDBC Driver

JDBC Driver is a software component that enables java application to interact with the database.

JDBC client

The JDBC client **provides access to the data server from Java™ and Java-based tools**. The JDBC client is distributed as a JAR (Java archive) file.

DriverManager

DriverManager manages the set of Java Database Connectivity (JDBC) drivers that are available for an application to use.

Applications can use multiple JDBC drivers concurrently if necessary.

JAVA WEB DEVELOPMENT

CRUD Operations

1. Create- The create function allows users to create a new record in the database.
2. Read- The read function is similar to a search function.
3. Update- The update function is used to modify existing records that exist in the database.
4. Delete.

JDBC Statements

- **Create Statement** - It is generally used for general–purpose access to databases and is useful while using static SQL statements at runtime.Statement

Syntax:

```
statement = connection.createStatement();
```

- **Prepared Statement**- represents a recompiled SQL statement, that can be executed many times. This accepts parameterized SQL queries. In this, "?" is used instead of the parameter, one can pass the parameter dynamically by using the methods of PREPARED STATEMENT at run time.

Get vs. Post

GET-

In case of Get request, only **limited amount of data** can be sent because data is sent in header.

Get request is **not secured** because data is exposed in URL bar.

It requests the data from a specified resource

POST-

In case of post request, **large amount of data** can be sent because data is sent in body.

Post request is **secured** because data is not exposed in URL bar.

It submits the processed data to a specified resource

JAVA WEB SERVICES & REST API -RESTFULL & SOAP

What is web service

Service delivered over the web.

- Web services are designed for machine-to-machine (or application-to-application) interaction
- Web services should be interoperable - Not platform dependent
- Web services should allow communication over a network

Types of Web Services

- SOAP
- REST

SOAP

- SOAP was earlier an abbreviation for Simple Object Access Protocol. In SOAP, the request and response are in XML format. However, not all types of XML are valid SOAP Requests.
- SOAP defines a standard XML format. We will use WSDL (Web Service Definition Language) to define the format of request xml and the response xml.
- SOAP format defines a SOAP-Envelope which envelopes the entire document.
- SOAP-Header (optional) contains any information needed to identify the request. Also, part of the Header is authentication, authorization information.
- SOAP-Body contains the real xml content of request or response.
- In case of error response, server responds back with SOAP-Fault

REST

- REST stands for REpresentational State Transfer.
- REST is an architectural style not a protocol.
- The main goal of RESTful web services is to make web services more effective.
- We can build REST services with both XML and JSON.
- The **key abstraction** is a resource in REST.
- A resource can be anything. It can be accessed through a Uniform Resource Identifier (URI).
- The resource has representations like XML, HTML, and JSON.

HTTP also defines the following standard status code:

- 404: RESOURCE NOT FOUND
- 200: SUCCESS
- 201: CREATED
- 401: UNAUTHORIZED
- 500: SERVER ERROR

JAVA WEB SERVICES & REST API -RESTFULL & SOAP

RESTful Service Constraints

- There must be a service producer and service consumer.
- The service is stateless.
- The service result must be cacheable.
- The interface is uniform and exposing resources.
- The service should assume a layered architecture.

Advantages of RESTful web services

- RESTful web services are platform-independent.
- It can be written in any programming language and can be executed on any platform.
- It provides different data format like JSON, text, HTML, and XML.
- It is fast in comparison to SOAP because there is no strict specification like SOAP.
- These are reusable.
- They are language neutral.

Advantages of Soap Web Services

- WS Security: SOAP defines its own security known as WS Security.
- Language and Platform independent: SOAP web services can be written in any programming language and executed in any platform.