# AWS, DEVOPS Tools Material

**Anil K M**

# AWS- CloudFormation

## CloudFormation– Introduction

- CF is a service give you the ability create complete AWS environment with different resources, services in a template called JSON.
- **Template** – collection of resources, it contains attributes etc. An AWS CloudFormation template is a JSON or YAML formatted text file. You can save these files with any extension, such as .json, .yaml, .template, or .txt. Templets are blueprints for building your AWS resources. For example, in a template, you can describe an Amazon EC2 instance, such as the instance type, the AMI ID, block device mappings, and its Amazon EC2 key pair name. Whenever you create a stack, you also specify a template that AWS CloudFormation uses to create whatever you described in the template.
- **Stack** – Once template is feeded to CF engine it will create the resources for you, collection of all these created/provisioned resources together as a unit called Stack.
  When you use AWS CloudFormation, you manage related resources as a single unit called a stack. You create, update, and delete a collection of resources by creating, updating, and deleting stacks. All the resources in a stack are defined by the stack's AWS CloudFormation template. Suppose you created a template that includes an Auto Scaling group, Elastic Load Balancing load balancer, and an Amazon Relational Database Service (Amazon RDS) database instance. To create those resources, you create a stack by submitting the template that you created, and AWS CloudFormation provisions all those resources for you. You can work with stacks by using the AWS CloudFormation console, API, or AWS CLI.
- After your work is finished you can go ahead and delete the stack(from CF Console, CLI or API), CF will take care of deleting all your resources that are created as part of the stack.
- JSON editor can help to write JSON templates.
- CF designer can help to write templates; it has some intelligence.
- **Change Sets -** If you need to make changes to the running resources in a stack, you update the stack. Before making changes to your resources, you can generate a change set, which is summary of your proposed changes. Change sets allow you to see how your changes might impact your running resources, especially for critical resources, before implementing them.
- For example, if you change the name of an Amazon RDS database instance, AWS CloudFormation will create a new database and delete the old one. You will lose the data in the old database unless you've already backed it up. If you generate a change set, you will see that your change will cause your database to be replaced, and you will be able to plan accordingly before you update your stack. For more information, see Updating Stacks Using Change Sets.

- Provision AWS resources via Code (JSON or YAML).
- Create, Manage and Delete related AWS resources together as a unit called a Stack.
- Templates are JSON files or YAML file which define your stack.
- Version Control and Track changes

## Template Elements:

1. **AWSTemplateFormatVersion**
   - *AWS CloudFormation template version that the template conforms to*
2. **Description**
   - *Description about the template.*
3. **Metadata**
   - *Additional Information about the template*
4. **Parameters**

- AWS Cloud Formation parameters are values that you define in the template Parameters section.
- A parameter can have a default value.
- A parameter can be declared as one of following types: *String*, *Number*, or *CommaDelimitedList*.

```
"Parameters" : {
    "InstanceType" : {
      "Type" : "String",
      "Default" : "t1.micro",
      "AllowedValues" : ["t1.micro", "m1.small", "m1.large"],
      "Description" : "Enter t1.micro, m1.small, or m1.large. Default is t1.micro."
    }
}
```

**AWS-Specific Parameter Types (AWS::aws-product-name::data-type-name)**

1. AWS::EC2::AvailabilityZone::Name
2. AWS::EC2::Image::Id
3. AWS::EC2::KeyPair::KeyName
4. AWS::EC2::SecurityGroup::GroupName
5. AWS::EC2::SecurityGroup::Id
6. AWS::EC2::Subnet::Id
7. AWS::EC2::Volume::Id
8. AWS::EC2::VPC::Id
9. AWS::Route53::HostedZone::Id
10. List <***Above Parameters***>

**Pseudo Parameters**

- AWS Cloud Formation Pseudo Parameters are predeclared values you can use in your template.
- You do not declare them in your template. Use them the same way as you would a parameter, as the argument for the Ref function.
  **AWS::AccountId**
  **AWS::Region**
  **AWS::StackId**
  **AWS::StackName**
  **AWS::NotificationARNs**
  **AWS::NoValue**

**5. Mapping**

- Mappings enable you to specify conditional parameter values in your template.
- Each mapping has a logical name unique within the template, and defines one or more key-attribute pairs.

```
"Mappings" : {
"RegionMap" : {
    "us-east-1" : {
      "AMI" : "ami-76f0061f"
    },
    "us-west-1" : {
      "AMI" : "ami-655a0a20"
    },
    "eu-west-1" : {
      "AMI" : "ami-7fd4e10b"
    },
    "ap-southeast-1" : {
      "AMI" : "ami-72621c20"
    }
  }
}
```

6. **Conditions**

- Statements that define when a resource is created or when a property is defined.
- You might use conditions when you want to reuse a template that can create resources in different contexts, such as a Dev environment versus a production environment.

```
"Conditions" : {
  "CreateProdResources" : {"Fn::Equals" : [{"Ref" : "EnvType"}, "prod"]}
},
```

This is one of the parameter in the Parameters section

```
"NewVolume" : {
  "Type" : "AWS::EC2::Volume",
  "Condition" : "CreateProdResources",
  "Properties" : {
    "Size" : "100",
    "AvailabilityZone" : { "Fn::GetAtt" : [ "EC2Instance", "AvailabilityZone"
  }
}
```

7. **Transform**

- For Lambda-based applications to be provisioned using cloud formation template we use Transform.

8. **Resources – Required Section**

- AWS Cloud Formation Resources are features of AWS products which you declare as members of your stack.
- AWS Cloud Formation resource properties are values needed to create a resource.

- Every template **must declare a Resources** section with at least one resource.

```
"Resources" :{
    "MySimpleImage" : {
        "Type" : "AWS::EC2::Image",
        "Properties" : {
            "ImageId" : "myLinuxBundle-2011-12-30",
        }
    }
}
```

```
Resources : {
    "MyVolume" : {
        "Type" : "AWS::EC2::Volume",
        "Properties" : {
            "Size" : "4",
            "SnapshotId" : "snap234",
            "AvailabilityZone" : "us-east-1a"
        }
    }
 }
```

## 9. Outputs

- The Outputs section is an optional section of a AWS Cloud Formation template which enables you to return values back to the template user.
- When the stack fails to create, or when you delete a stack, the values declared in the Output section are not returned.

```
"Outputs" : {
    "URL" : {
        "Value" : { "Fn::GetAtt" : [ "MyLoadBalancer", "DNSName" ] }
    }
}
```

## 10. Intrinsic Functions

- AWS Cloud Formation provides functions that you can use to pass values that are not available until runtime.

```
"Outputs" : {
    "URL" : {
        "Value" : { "Fn::GetAtt" : [ "MyLoadBalancer", "DNSName" ] }
    }
}
```

| Name | Purpose |
|---|---|
| Fn::Base64 | The base64 encoding of the argument. |
| Fn::FindInMap | Returns the value of a key from the specified Mapping. |
| Fn::GetAtt | Returns the attribute value of the specified resource. |
| Fn::GetAZs | Get the Availability Zones where you can create AWS CloudFormation stacks. |
| Fn::Join | Concatenation of the elements of the second argument, separated by the first. |
| Ref | Return a resource or value based on a logical name or parameter. |

## Resource Attribute Reference

1. **CreationPolicy Attribute**

   - Associate the CreationPolicy attribute with a resource to prevent its status from reaching create complete until AWS CloudFormation receives a specified number of success signals or the timeout period is exceeded.

     **Supported Resources**:
     AWS::AutoScaling::AutoScalingGroup,
     AWS::EC2::Instance, and
     AWS::CloudFormation::WaitCondition.

2. **DeletePolicy Attribute**

   - Defines how AWS Cloud Formation handles the resource when its stack is deleted.

     - **Values** : Delete, Retain, Snapshot

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myS3Bucket" : {
         "Type" : "AWS::S3::Bucket",
         "DeletionPolicy" : "Retain"
      }
   }
}
```

3. **DependsOn Attribute**

When you add a DependsOn attribute to a resource, you specify that the resource is created only after the creation of the resource specified in the DependsOn attribute.

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Mappings" : {
        "RegionMap" : {
            "us-east-1" : { "AMI" : "ami-76f0061f" },
            "us-west-1" : { "AMI" : "ami-655a0a20" },
            "eu-west-1" : { "AMI" : "ami-7fd4e10b" },
            "ap-northeast-1" : { "AMI" : "ami-8e08a38f" },
            "ap-southeast-1" : { "AMI" : "ami-72621c20" }
        }
    },
    "Resources" : {
        "Ec2Instance" : {
            "Type" : "AWS::EC2::Instance",
            "Properties" : {
                "ImageId" : {
                   "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]
                }
            },
            "DependsOn" : "myDB"
        },
        "myDB" : {
            "Type" : "AWS::RDS::DBInstance",
            "Properties" : {
               "AllocatedStorage" : "5",
               "DBInstanceClass" : "db.m1.small",
               "Engine" : "MySQL",
               "EngineVersion" : "5.5",
               "MasterUsername" : "MyName",
               "MasterUserPassword" : "MyPassword"
            }
        }
    }
}
```

**4. Metadata Attribute**

- By adding a Metadata attribute to a resource, you can add data in JSON format to the resource declaration.

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "MyS3Bucket" : {
         "Type" : "AWS::S3::Bucket",
         "Metadata" : { "Object1" : "Location1",  "Object2" : "Location2" }
      }
   }
}
```

**5. Update Attribute**

- The UpdatePolicy attribute enables you to specify how AWS Cloud Formation handles updates for a particular resource.
    - **Supported Resources**:
        - AWS::AutoScaling::AutoScalingGroup

```
"ASG1" : {
   "UpdatePolicy" : {
      "AutoScalingRollingUpdate" : {
         "MinInstancesInService" : "1",
         "MaxBatchSize" : "1",
         "PauseTime" : "PT12M5S"
      }
   },
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Properties" : {
      "AvailabilityZones" : { "Fn::GetAZs" : { "Ref" : "AWS::Region" } },
      "LaunchConfigurationName" : { "Ref" : "ASLC" },
      "MaxSize" : "3",
      "MinSize" : "1"
   }
}
```

## Best Practices

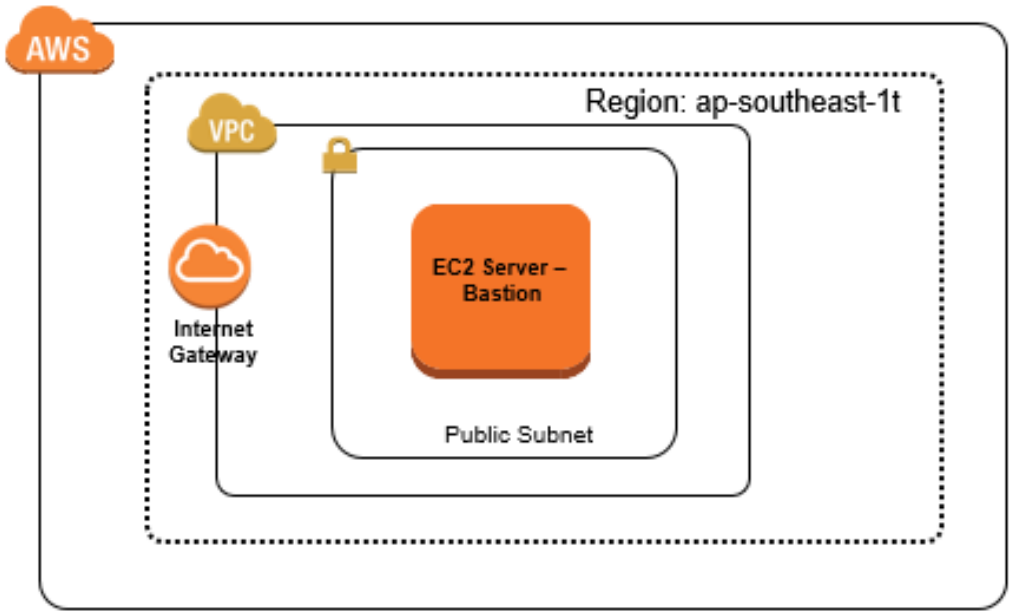
- Use pseudo parameters to retrieve global data
- Use Mappings to define variables so that they can be reusable with in the template after defining them once.
- Do not use access credentials
- Use AWS specific parameters
- Validate templates before we use them.
- Use of ‘change sets’ would ensure the complex stacks before we update them.
- Use nested stacks to reduce the complexity and improve the maintainability by reusing the common Template patterns.

## Limitations

- AWS CloudFormation has a resource hard limit as 200
- AWS CloudFormation ‘**does not’** Validate Templete to validate the properties of resources defined in the template.

## Demo

1. Creating a CloudFormation for a VPC, one public Subnet. Launch one EC instance in Public Subnet.



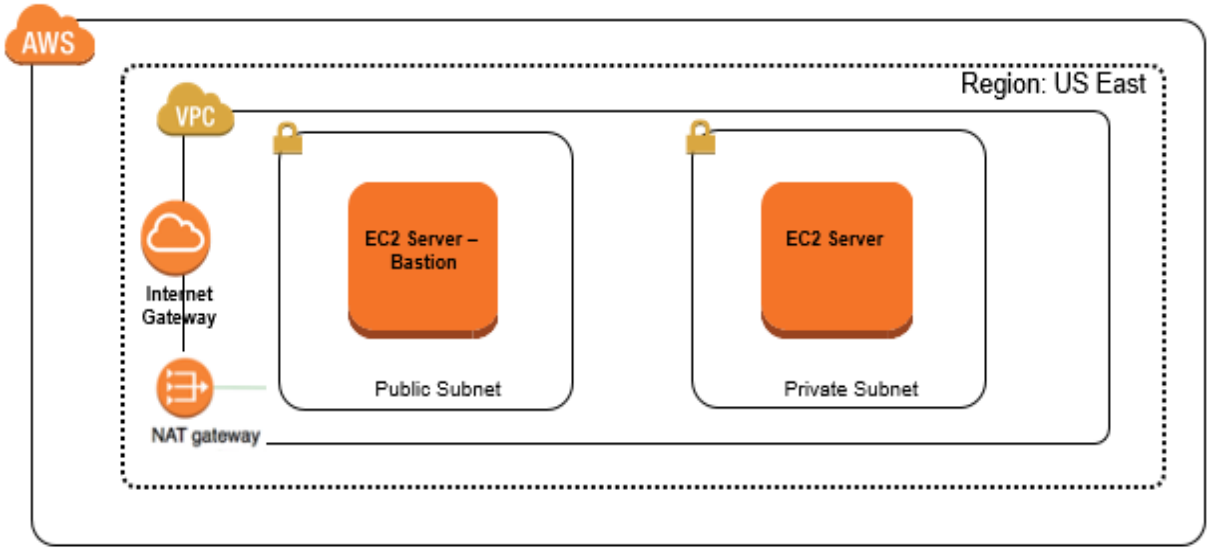

### Steps:

1. Create VPC (Say 10.0.0.0/16).
2. Create IGW and associate with VPC.
3. Create Public Route table
4. Create Route table entries for Public RTB with IGW – 0.0.0.0/0.
5. Create Public Subnet (Say 10.0.1.0/24).
6. Associate Public Route table with Public Subnet.
   **Now Network will be ready.**
7. Create one EC2 in Public Subnet.

PeerTrainingDemo.txt

### Assignment

1. Creating a CloudFormation for a VPC, one public Subnet and one private Subnet. Launch one EC instance in Public Subnet and one EC2 instance in Private Subnet.




**Steps for Assignment:**

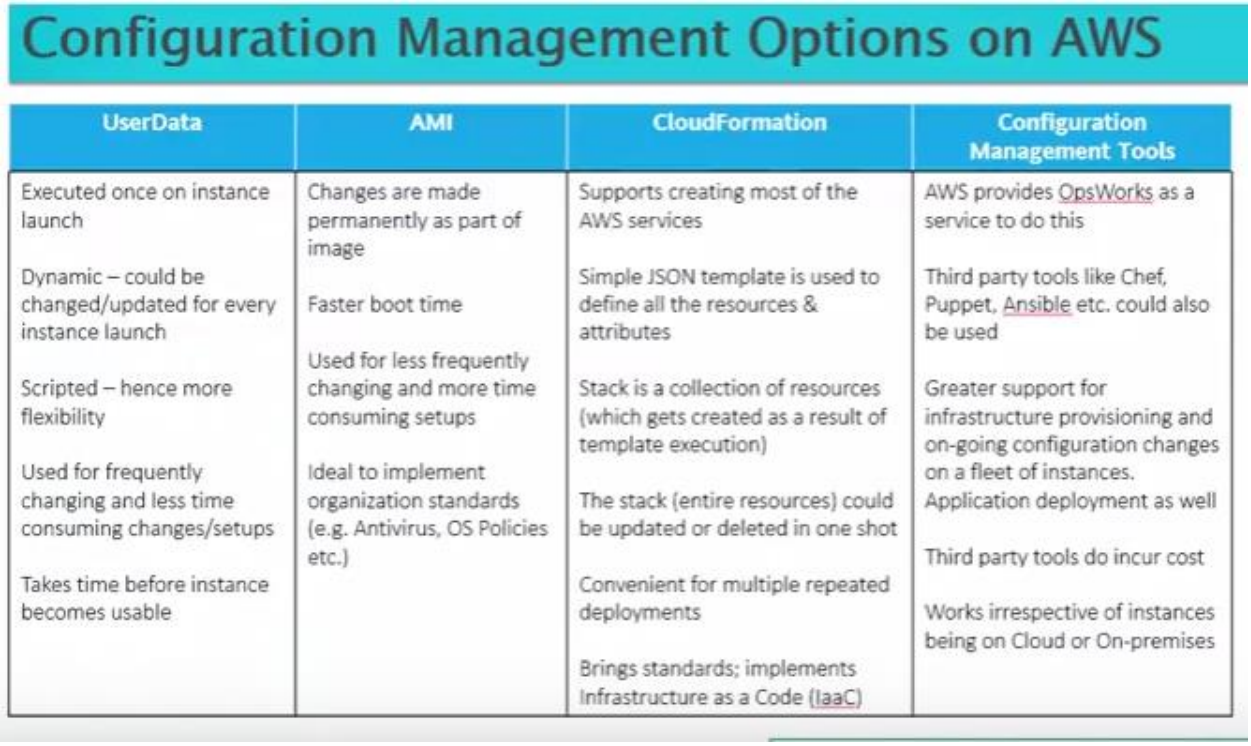
1. Create VPC (Say 10.0.0.0/16).
2. Create IGW and associate with VPC.
3. Create Two Route tables – One for Public and one for Private.
4. Create Route table entries for Public RTB with IGW – 0.0.0.0/0.
5. Create Public Subnet (Say 10.0.1.0/24).
6. Associate Public Route table with Public Subnet.
7. Create NAT Gateway in the Public Subnet created in Step 5.
8. Create Route table entry for the Private Route table with NAT GW(created in Step 6) for 0.0.0.0/0.
9. Create Private Subnet (Say 10.0.2.0/24).
10. Associate Private Route table with Private Subnet.
    **Now Network will be ready.**
11. Create one EC2 in public Subnet.
12. Create one EC2 in private Subnet.

## Next Session Topics

Session 2:

1. Bootstrapping in Cloudformation
   UserData, Metadata, cfn-helper scripts
   Wait Conditions and Wait Handlers
1. Nested Stacks
   UserData, Metadata, cfn-helper scripts
   Wait Conditions

## AWS Configuration Managment Options - UserData, AMI, CloudFormation, OpsWorks




**User Data –**

- User data is nothing but scripts that run on cmd prompt in linux or windows for customization.
- More efforts required, you can run once when your instance is launched

**AMI –**

- Create once use many times, less effort when you do the same on multiple instances.

**CloudFormation –**

- Specify all your resources in JSON template and feed this template into CloudFormation engine and CF engine create a stack (nothing but collection of resources). All the resources come up for you in one shot.
- It can help to automate your deployments, also you can do multiple deployments and ensure that same standards that are used in one environment remains the same when other environments come up.
- It is concept of infrastructure as a code
- When you use a tag, the same tag will be replicated in all the resources in environments, good thing is if you enable this tag as cost allocation tag everything flows to detail billing file.

**Configuration Mgt tools –**

- 3rd party tools – chef, puppet, ansible etc

- Aws provides OpsWorks to do the same thing. You can also use this on premise, but they will charge for it.

**Demo on User Data –**

#!/bin/sh
yum -y install httpd (apache web server)
chkconfig httpd on (checking server is on or not)
/etc/init.d/httpd start (start the service)

- Put this code in 'User Data' under 'Advanced Details' in '3.Conguration Instance' step in EC2 instance creation process through AWS console.
- Any user data code is executing as root in linux.

## AWS CloudFormation - Stack, Template, Parameters, Mapping, IAM Role, Stack Policy

Stack, Template, Parameters, Mapping, Functions, Pseudo Parameters, Deletion Policy, Tags, Using IAM Role, Stack Policy.

- CF is a service give you the ability create complete aws environment with different resources, services in a template called JSON.
- **Template** – collection of resources, it contain attributes etc.
- **Stack** – Once templates is feeded to CF engine it will create the resources for you, collection of all these created resources as a unit is called Stack.
- After your work is finished you can go ahead and delete the stack, CF will take care of deleting all your resources as part of the stack.
- Json editor can help to write templates.
- CF designer can help to write templates; it has some intelligence.

LAMP stack created using CF. To check this stack , ssh EC2 and check like this

$ mysql -h localhost -P 3306 -u edureka -p

Enter password:

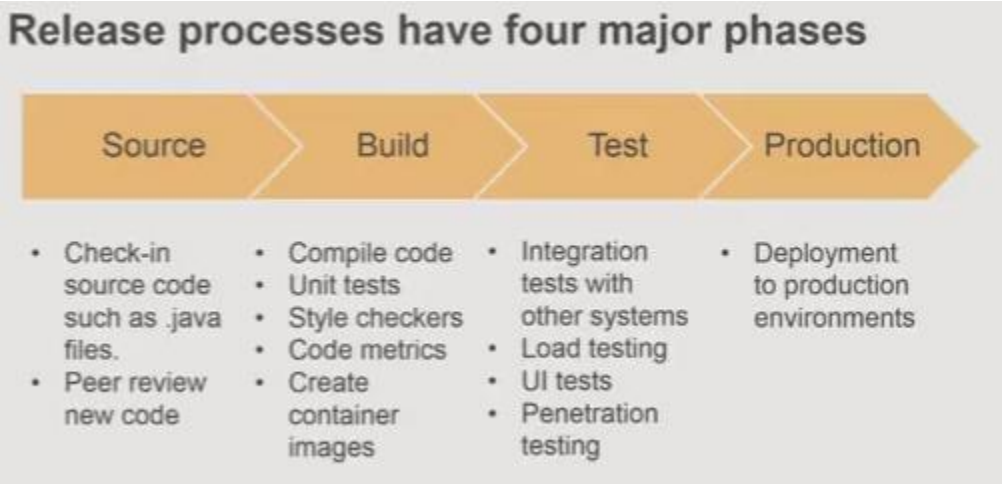
Mysql> show databases;

3 rows in set

Mysql>

## AWS re:Invent 2016: Infrastructure Continuous Delivery Using AWS CloudFormation (DEV313)

https://www.youtube.com/watch?v=TDalsML3QqY




## What do we need for infrastructure continuous delivery?

- A way to treat infrastructure as code.
- Tools to manage the workflow that creates and updates infrastructure resources.
- Tools to properly test and inspect your changes for defects and potential issues

## What do we need for infrastructure continuous delivery?

| Infrastructure as code | Workflow |
| --- | --- |
| A practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control and continuous integration. | Build, test, and deploy your code every time there is a code change, based on the release process models you define, enabling you to rapidly and reliably deliver changes. |

You can do all these things using CF and CodePipeline

## We need

**Infrastructure as Code**

AWS CloudFormation

**Workflow**

AWS CodePipeline

## AWS CloudFormation

- Create templates of your infrastructure
- Version control /code review /update templates like code
- CloudFormation provisions AWS resources based on dependency needs
- Integrates with development, CI/CD, management tools
- No additional charge to use

# S3 vs EBS vs EFS

**Block Storage –** Block level operations are possible; hence one block is changed (piece of the file) that contains the changed data

- Very fast read and write operations block storage would be best
- EBS – block level (EFS also block level storage)

**Object Storage –** Entire file must be removed and new file needs to be put there

- Put the files occasionally object level would be best
- S3 – Object level

S3 & EBS

**S3 –**

- used for WORN operations (Write Once Read Many times)
- writing continuously (but not reading) s3 is good. log file from different sources continuous ingestion S3 would be best
- store static assets of a website you put once and access world wide(write once read many times)
- Scalable, size not be planned. No restriction on sizing, you can keep loading the files.
- But not suitable for hosting OS or Database (but you can store) reason is OS and DB are required very fast read and write operations.

**EBS –**

- EBS works best for hard disk or server disks. So use with EC2
- Persistent and high performance in terms of read write. Even you stop EC2 instance data will not be lost. Depending on EBS type you can achieve high read and write performance.
- Replicated with in AZ and could be mounted to one EC2 in the same AZ. 2 times it will replicated in same AZ by AWS. Create an EBS in one AZ will **not** be possible to attach in EC2 in other AZ.

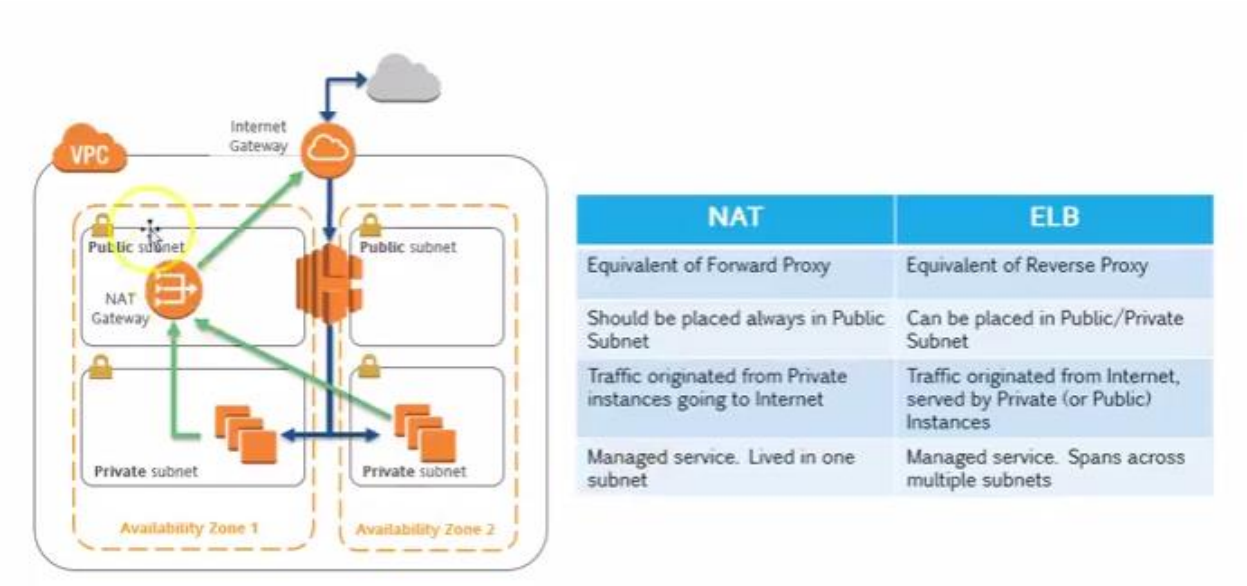
**EFS –**

- Replicated across AZs in a region.
- One EFS you can mount on 'n' number of EC2 instances within a region. Ex: you create one EC2 in AZ-a and second EC2 in AZ-b in same region that can mount the same EFS volume.
- When you create EFS in AWS, underlaying the EFS is replicated across different AZ in the same region by AWS.
- Could be mounted to on premise server as well (over VPN or DirectConnect)
- No sizing to be done (unlike EBS). Auto scales.

# AWS - Difference between NAT and ELB - Comparison

NAT – Network Address Translation (NAT Gateway)

ELB – Elastic Load Balancer




| NAT | ELB |
|---|---|
| Equivalent of Forward Proxy | Equivalent of Reverse Proxy |
| Should be placed always in Public Subnet | Can be placed in Public/Private Subnet |
| Traffic originated from Private instances going to Internet | Traffic originated from Internet, served by Private (or Public) Instances |
| Managed service. Lived in one subnet | Managed service. Spans across multiple subnets |

**ELB –**

- Webservers are kept in private subnet and serving public people(internet) through ELB which is in public subnet. If we are not using ELB we should put our webservers in public subnet to access the public people.
- **ELB** – does 2 things 1) accept the incoming traffic from internet(public people) and distribute among the instances which are registered to it. 2) continuous do the health check of the instances. If any of the instance is unhealthy it will stop sending the traffic to that particular instance.
- Some times you can keep ELB in private also, between your web tier and app tier
- ELB is **managed service** and highly available**(HA)** in nature, you do not need to maintain the internal instances of the ELB, ELB will takes care of it.
- ELB can span in multiple subnets in multiple AZ (see this in the above pic).

**NAT Gateway –**

- When your instances call APIs like whether, currency etc in internet, so your private instances should reach internet through NAT Gateway (forward proxy). It is similar to your desktop (having private IPs) accessing internet.
- The traffic which is originated towards internet by the private machines goes via NAT instance. And the reply comes back.
- NAT Gateway is **managed service**, its HA within one AZ. NAT Gateway never span across multiple AZ. For more HA you can put separate NAT Gateways in each AZ.

# Setting continuous Integration with Jenkins

https://www.youtube.com/watch?v=zEQUuo5nWbo

**How to setup Jenkins master and couple of slave nodes.**

Launch 3 t2.micro ec2 instances using aws console, 1 for master 2 for slaves
Login to the master instance and inside the server we will create SSH key that SSH key we will put in authorized keys file of the both of the slave nodes so they communicate between each other. Also install Jenkins on the master server

**Cmd prompt> ssh -I awstutorialseries.pem ec2-user@54.174.219.124** (login to master server, public ip from console used here)

//login to amazan linux AMI

**[ec2-user@ip-172-30-0-16 ~] $ sudo yum update -y**

CREATE AN SSH KEY

**$ ssh-keygen -t rsa -C testing@gmail.com (creating an ssh key)**

// Key created

//your identification has been saved in /home/ec2-user/.ssh/id_rsa
//your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub

**[ec2-user@ip-172-30-0-16 ~] $ cat .ssh/id_rsa.pub** ( JENKINS MASTER SERVER: ip-172-30-0-16)

Copy the key

//ssh-rsa AAA…..xtbs testing@gmail.com copy entire output

**Login to the slave node 1 using public ip of the node**

[ec2-user@ip-172-30-0-15 ~] $ **echo "ssh key (paste here from earlier copy)" >> .ssh/authorized_keys**

[ec2-user@ip-172-30-0-15 ~] $ exit

Simmilarly do the same, copy SSH key into the slave node 2 of **.ssh/authorized_keys** directory.

Install Jenkins using the 3 commands from Jenkins installations steps from website

Commands:

1. sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo
2. sudo rpm -import http://pkg.jenkins-ci.org/redhat/jenkins-ci.org.eky
3. sudo yum install jenkins

JENKINS RUN ON PORT 8080. SO SECURITY GROUP ALLOW THAT PORT OPEN.

START THE JENKINS SERVICE

[ec2-user@ip-172-30-0-16 ~] $ sudo service jenkins start

[ec2-user@ip-172-30-0-16 ~] $

GO TO CONSOLE AND GRAB THE MASTER PUBLIC IP ADDRESS AND PASTE INTO THE BROWSER LIKE IP:8080 … JENKINS WILL COME UP.

BEFORE EXECUTE NEW JOB, WE WOULD LIKE TO ADD UP NEW SLAVES THAT WE HAVE CREATED EARLIER

CLICK ON "BUILD EXECUTOR STATUS". HERE WE ARE SEEING MASTER NODE

CLICK ON "NEW NODE" ON THE LEFT. GIVE NODE NAME AS "JENKINS NODE 1", SELECT DUMB SLAVE & CLICK OK.

UPDATE REMOTE ROOT DIRECTORY AS **"/home/ec2-user"** (give whatever would be home dir)

Host: <PRIVATE IP OF THE SLAVE NODE 1>

**NOTE:** ALWAYS USE PRIVATE IP SO THAT IT WILL NOT CHARGE AS IT IS INTERNAL IP. IF IT PUBLIC IP IT WILL BE CHARGED. Bandwidth for the internal ip will be free.
CLICK ON “ADD”. GIVE CREDENTIALS SELECT KIND AS “SSH USERNAME WITH PRIVATE KEY”

USERNAME: ec2-user
PRIVATE KEY: <GET KEY FROM MASTER NODE>

Login to the master server
[ec2-user@ip-172-30-0-16~]$ cat .ssh/id_rsa.pub
<displays Key>
[ec2-user@ip-172-30-0-16~]$ cat .ssh/id_rsa
------BEGIN RSA PRIVATE KEY ----

-------END RSA PRIVATE KEY -------
[ec2-user@ip-172-30-0-16~]$
Copy above private key(------BEGIN RSA PRIVATE KEY ----) and paste RSA PRIVATE key in PRIVATE KEY field of the slave node 1 and click on Add
Select save
NOW YOU SEE THE SLAVE-1 NODE ADDED.

NOW CLICK ON NEW NODE TO ADD 2ND NODE.
NOW SELECT ON COPY EXISTING NODE. COPY FROM: JENKINS NODE 1
CLICK ON OK.
EVERYTHING IS SAME EXCEPT THE HOST, GIVE THE SLAVE NODE 2 PRIVATE IP ADDRESS. And credentials as “ec2-user” (it is same as node 1)
CLICK ON SAVE
GO BACK TO NODES SECTION BY CLICKING ON “NODES” AT TOP. CLICK ON REFRESH
GO BACK TO JENKINS HOME…. SEE “BUILD EXECUTOR STATUS” YOU NOTICE MASTER, JENKINS NODE 1 AND JENKINS NODE 2.

NOW GO TO CREATE NEW JOB. To test these jobs are running on the different servers.
CLICK ON “CREATE NEW JOBS”
ITEM NAME: JOB 1, SELECT FREESTYLE PROJECT AND CLICK ON OK.
MAKE IT SIMPLE JOB
BUILD: EXECUTE SHELL -> COMMAND > sleep 10s
CLICK SAVE.

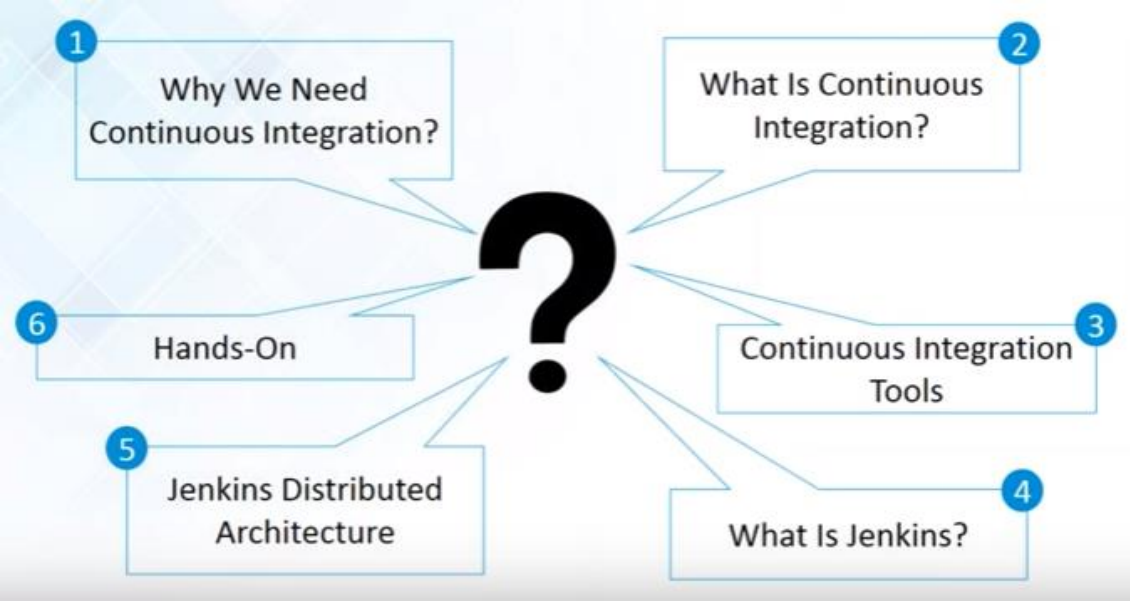
Go to Jenkins home. CREATE ONE MORE JOB.
ITEM NAME: JOB 2, SELECT FREESTYLE PROJECT AND CLICK ON OK.
MAKE IT SIMPLE JOB
BUILD: EXECUTE SHELL -> COMMAND > SLEEP 10S
CLICK SAVE.

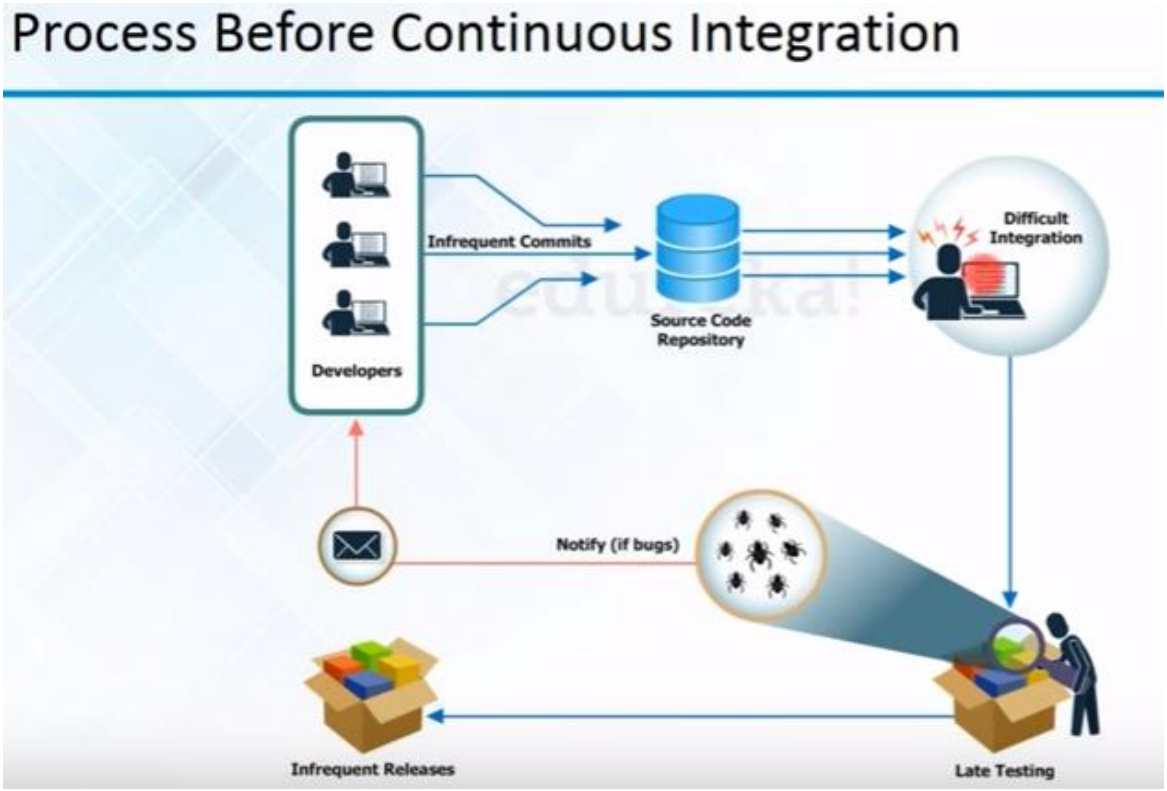
REFRESH THE BROWSER
RUN THE BOHT JOBS

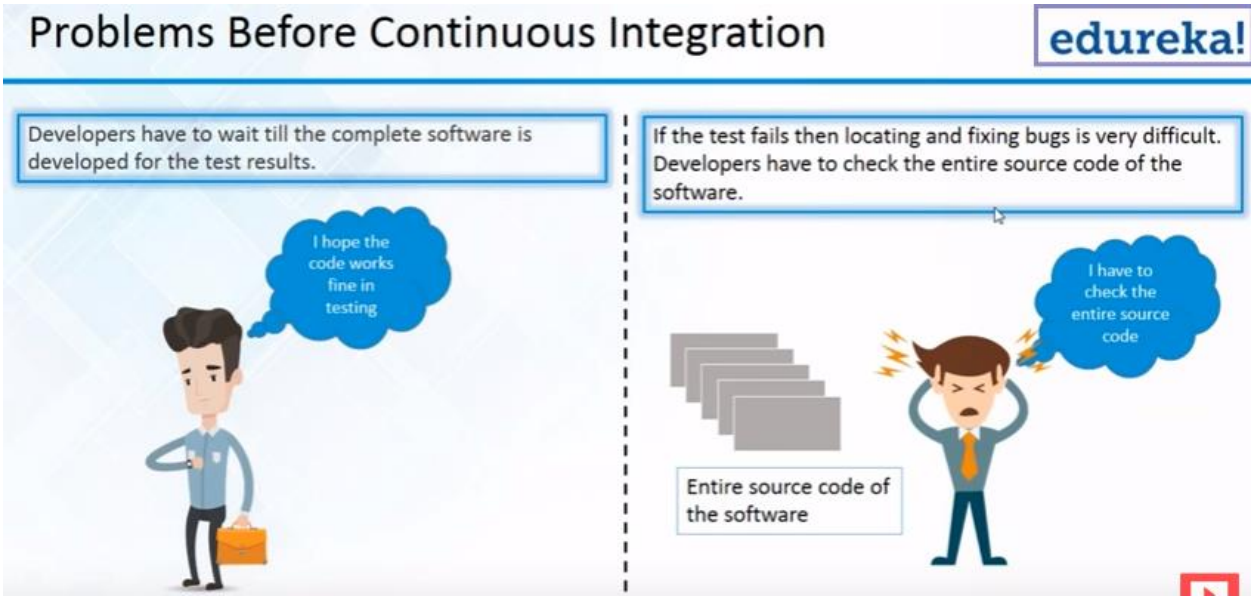
YOU WILL NOTICE JOB1 IS RUNNING ON MASTER AND JOB2 IS RUNNING ON SLAVE 1.

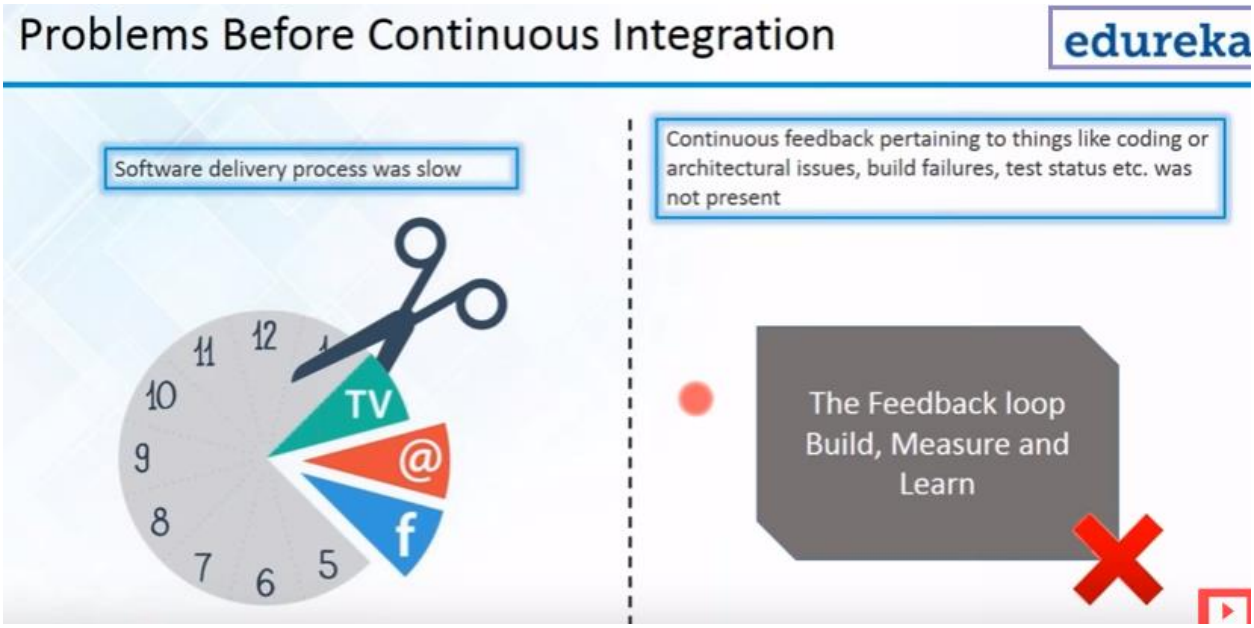
## What is Jenkins | Jenkins Tutorial for Beginners | Jenkins Continuous Integration Tutorial | Edureka

https://www.youtube.com/watch?v=p7-U1_E_j3w&t=1s



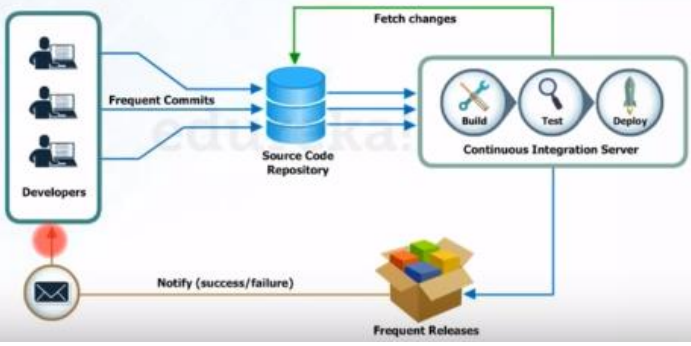



CONTINUOUS FEEDBACK YOU WILL GET THE FEEDBACK ON THE RUN.

## Continuous Integration To The Rescue

- Since after every commit to the source code an auto build is triggered and then it is automatically deployed on the test server
- If the test results shows that there is a bug in the code then the developers only have to check the last commit made to the source code
- This also increases the frequency of new software releases
- The concerned teams are always provided with the relevant feedback



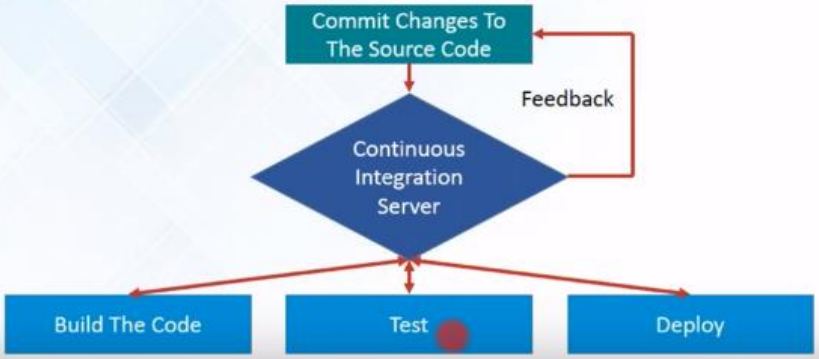

## Continuous Integration To The Rescue

| Before Continuous Integration | After Continuous Integration |
| --- | --- |
| The entire source code was built and then tested. | Every commit made in the source code is built and tested. |
| Developers have to wait for test results | Developers know the test result of every commit made in the source code on the run |
| No Feedback | Feedback is present |

## What Is Continuous Integration

edureka

- Continuous Integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently.
- Every commit made in the repository is then built. This allows the teams to detect the problems early.

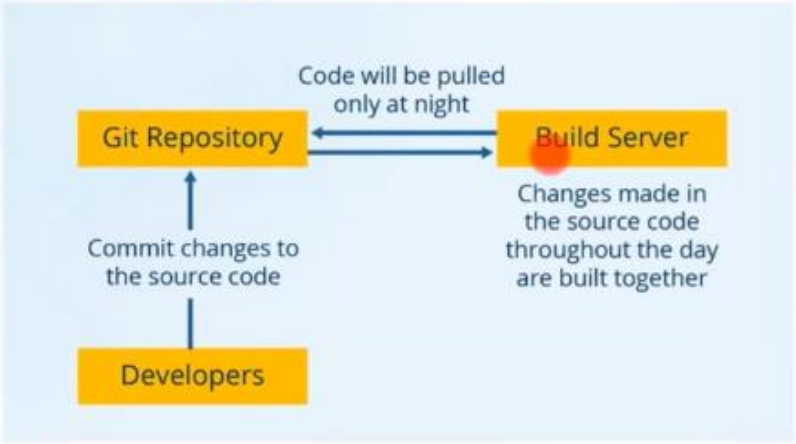




## Continuous Integration Case-Study: Nokia

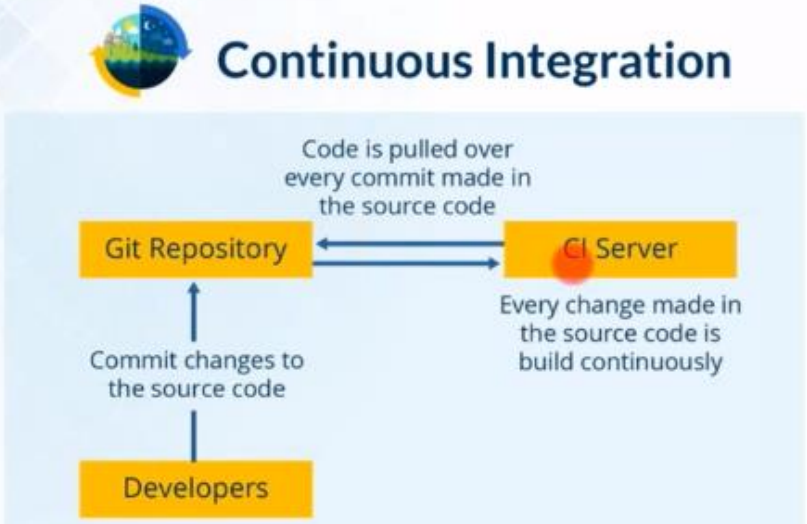
Problem Statement:

**Nightly build**

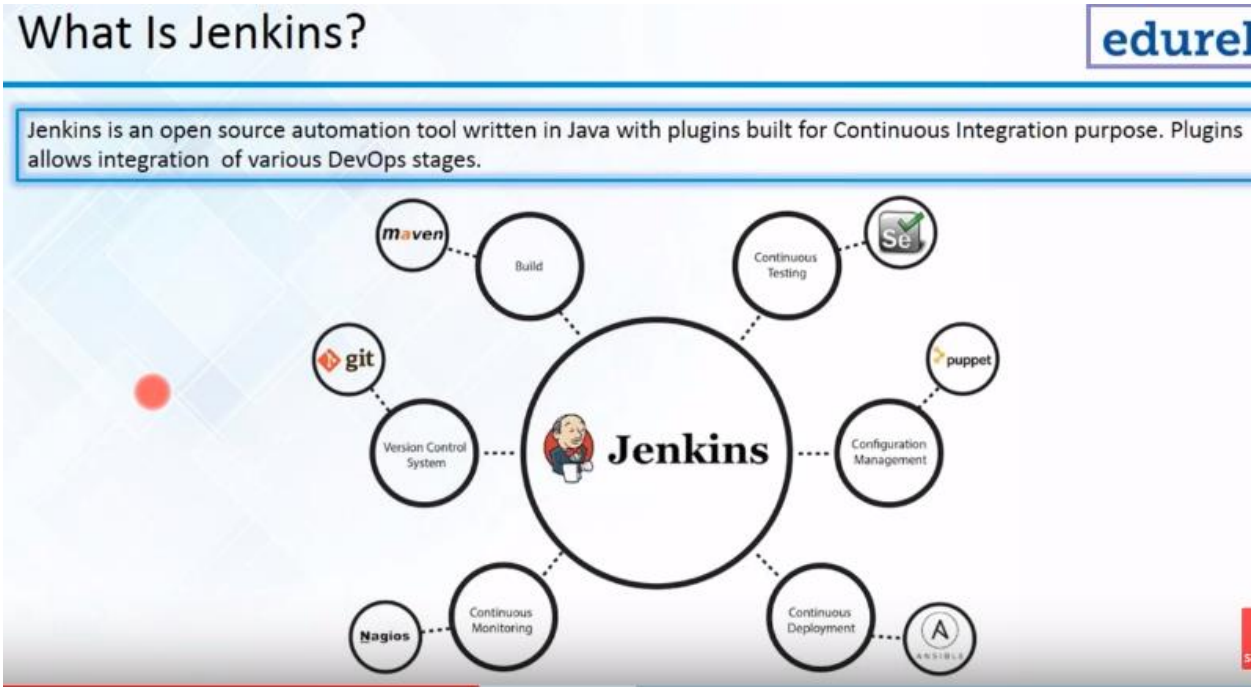


## Continuous Integration Case-Study: Nokia

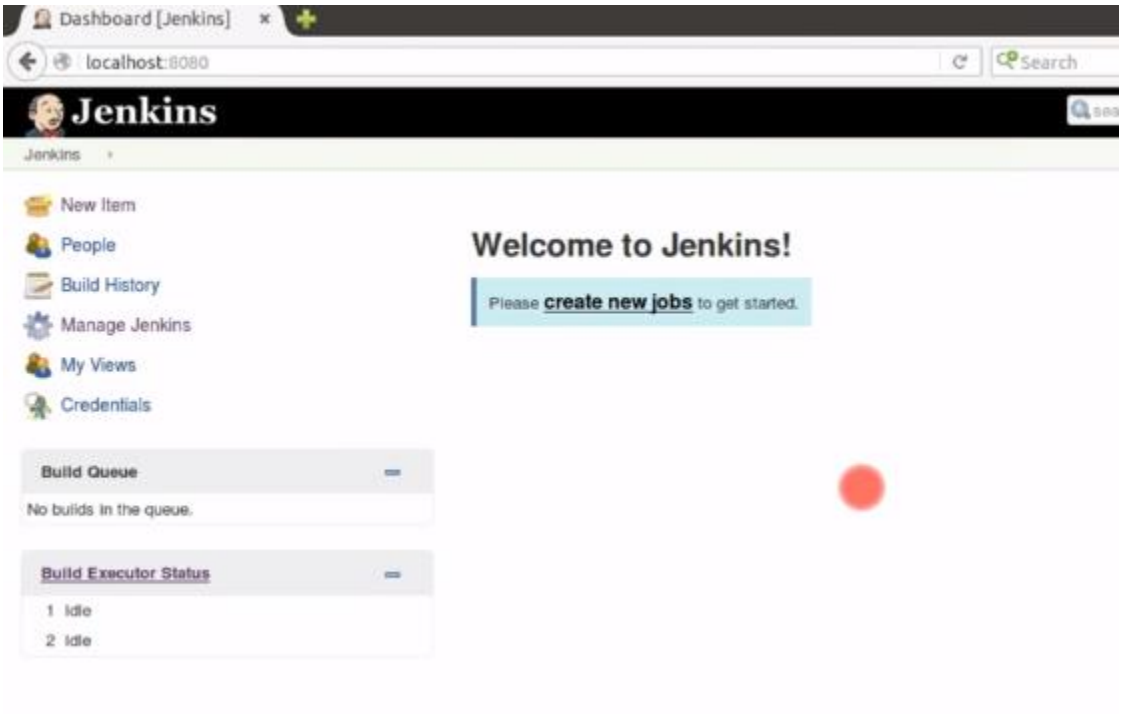
Solution:

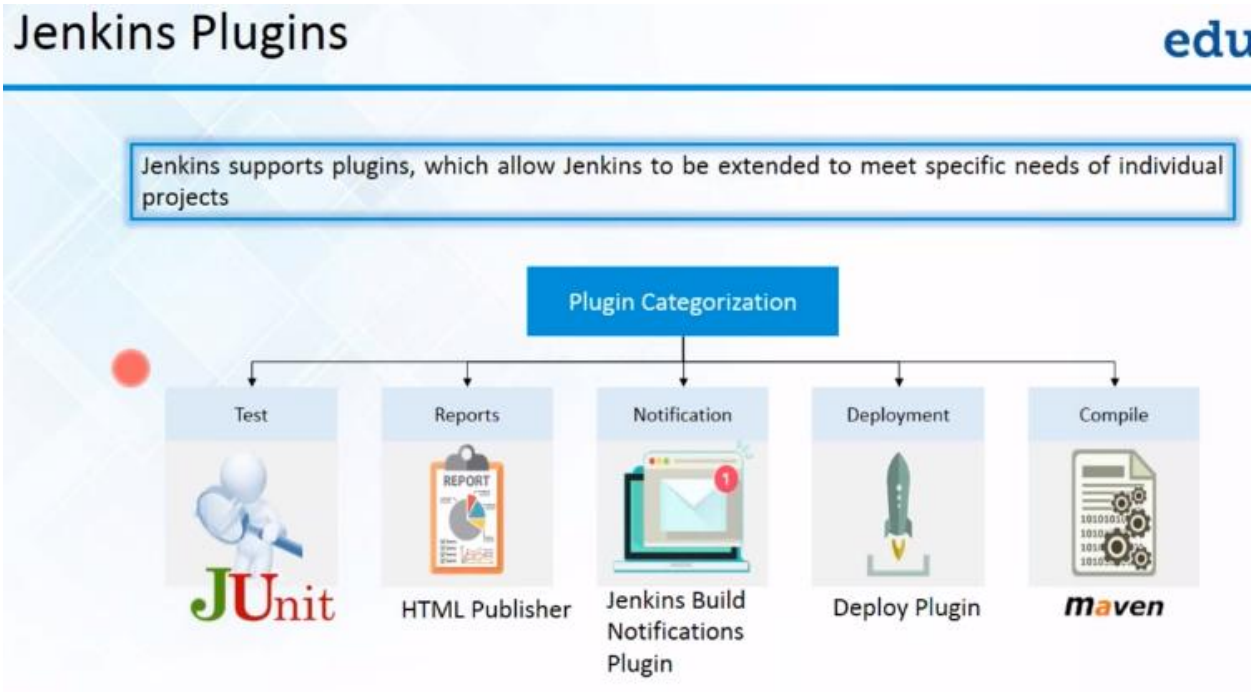



## Continuous Integration Tools




See the Jenkins dashboard. There are no jobs. Click on “New Item” to create new job or project.






**How to install plug-ins?**

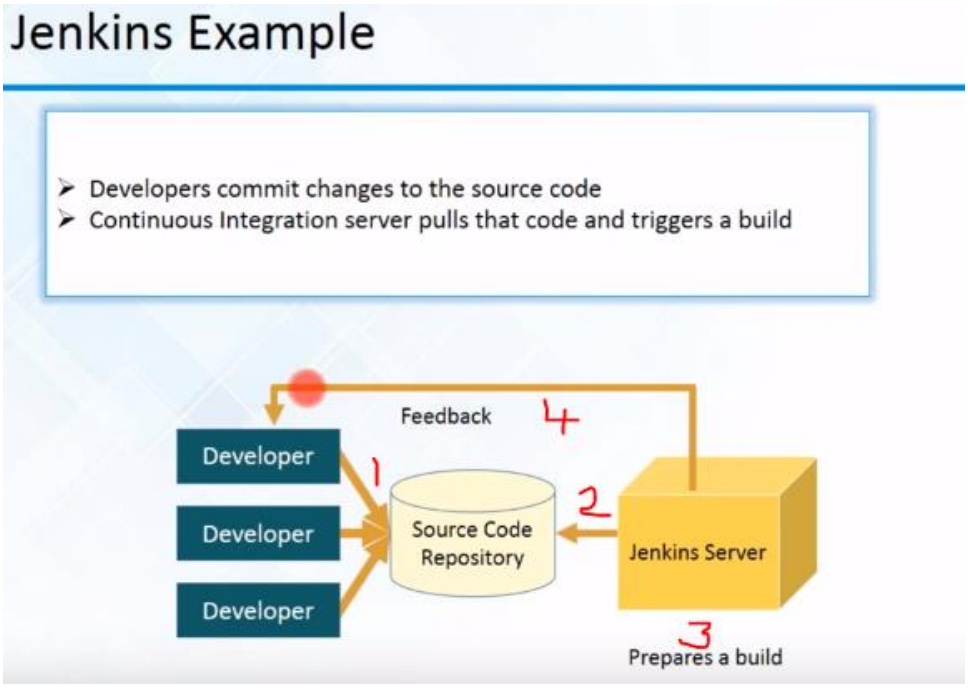
Go to Jenkins dashboard -> click on “Manage Jenkins” on left -> click on “Manage Plugins” -> Here you will see some tabs 1. Updates 2. Available 3. Installed 4. Advanced.

Now go to 2.Available tab and install HTML publisher plugin. Go to filter and type HTML then you see the “HTML Publisher plugin” and select the check box and click on “Install without restart”

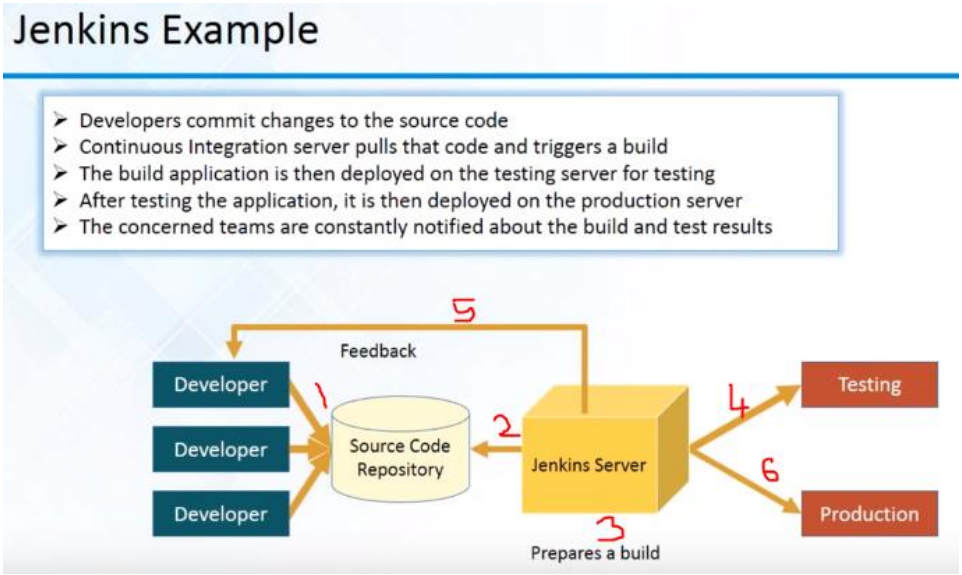
Now it is installing Plugins/Upgrades

It is success.

Go to Jenkins dashboard



**See this practically –**
Go to Jenkins dashboard
Click on new Item -> enter item name “Compile” -> select “free style project” -> click on OK
Go to 2nd tab “Source Code Management” -> select “Git” under Source Code Management -> give Repository URL: https://github.com/<username>/nameof yourrepositry.git
In Build option select “Invoke top-level Maven targets”
Maven has build life cycle, it made up of multiple build phases, typically the sequence of build phase will be **validate** the code, **compile**, **test**, perform unit test by using suitable unit testing frame works, **package** your code in distributable formats like .jar, then **verify** it, then **install** and **deploy** prod like environments.
Goals – compile <it will compile the code>
Click on Save
Click on “Build Now” on left.
Click on the build number #1(round ball) then you see the console output
Finished: SUCCESS
Now go back to Jenkins dashboard “compile” job displayed with Blue color ball, it means it is successful.




Once build is prepared Jenkins server deploy the code in Test server to test the build.

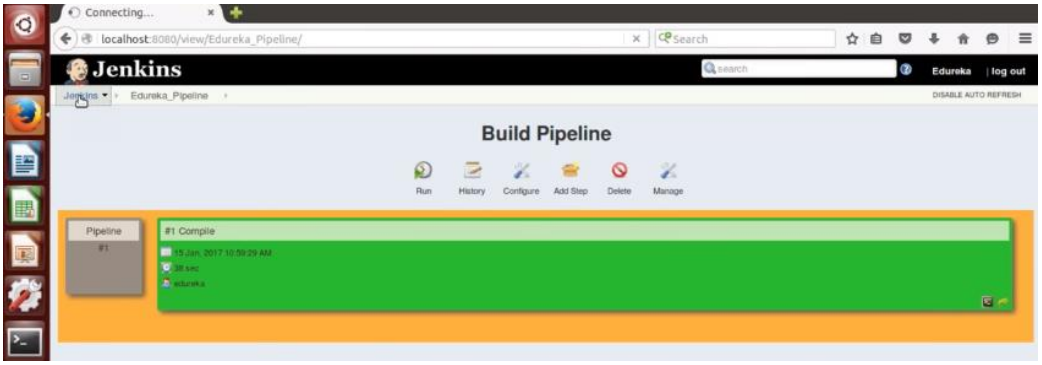
**<u>See it practically</u>**
In git repository, test cases are already defined, we are going to analyze the test cases with the help of maven

Before going to “source code management, I want to review the code with the help of **PMD plugin**
For this again click on New Item give “code_review” -> select freestyle project -> click on OK
Go to 2nd tab “Source Code Management” -> select “Git” under Source Code Management -> give Repository URL: https://github.com/<username>/name of your repositry.git
In Build option select “Invoke top-level Maven targets”
GOALS: Give “-P metrics pmd:pmd”. It will give PMD report that contains all warnings and errors.
Post build actions: select “Publish PMD analysis results”
Click apply and save.

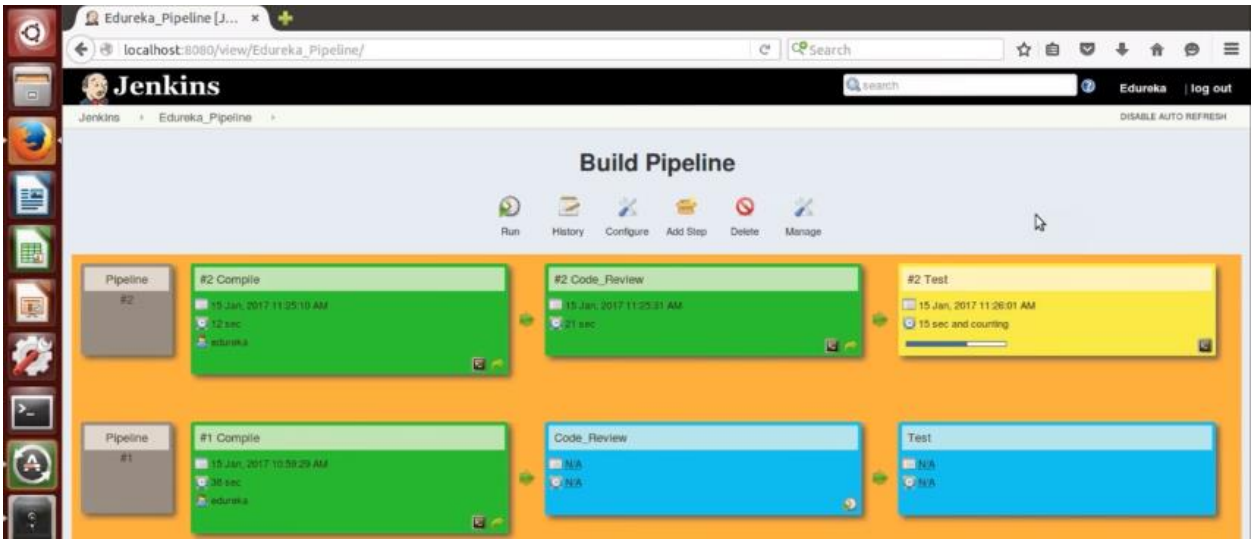
Finally click on build now and click on the console output by clicking on the ball before #1 in build history.
Success
Go back to dashboard and you see PMD Warnings on left. Click on it and see the PMD Results with all details.

Now go back to the Jenkins dashboard.
Click on “New Item” -> give “Test” in item name box -> select freestyle project -> click on OK
Repository URL: https://github.com/<username>/name of your repositry.git
In Build option select “Invoke top-level Maven targets”
Goals: test <it is maven build phase>
Click on apply and Save
Click on “Build Now” on left.
Click on the build number #1(round ball) then you see the console output
All unit tests are running
Finished: SUCCESS
Now go back to Jenkins dashboard “test” job displayed with Blue color ball, it means it is successful.

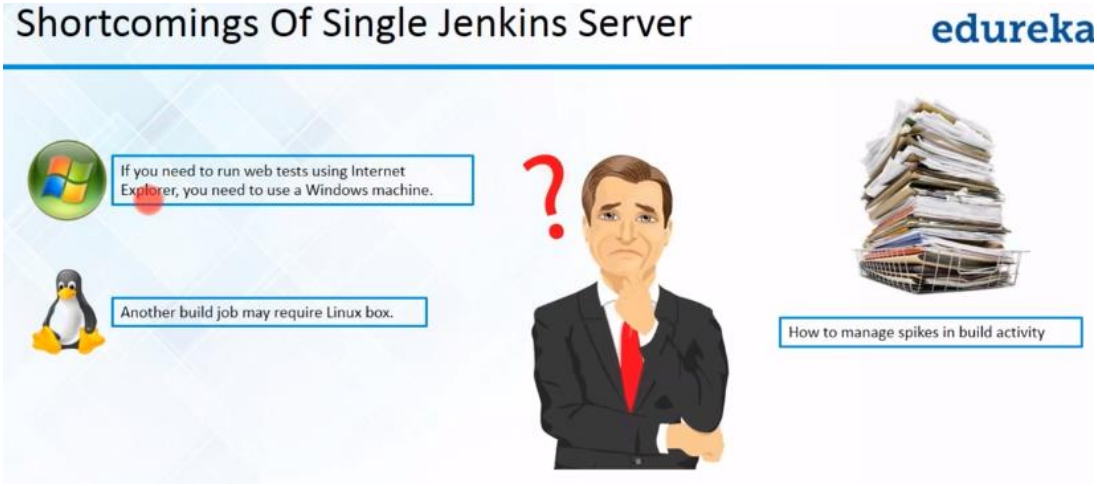
**How to create build pipeline?**
Go to Jenkins dashboard, click on + sign next to All tab, and select “Build pipeline view” and give view name “edureka_pipeline”
Give description
Select initial job as “compile” under pipeline flow
No of displayed builds as 5 under Trigger Options
Click on apply and ok
Currently you see only one pineline job. So add few more jobs



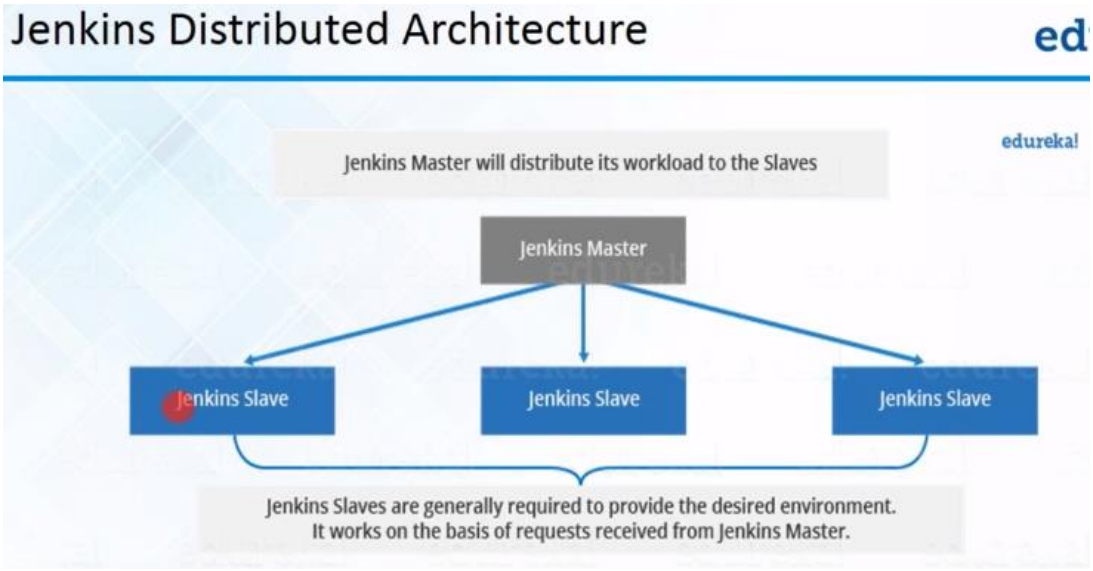

Go to Jenkins dashboard right click on “Code Review” and select “configure”
Go to “Build Triggers” tab select “Build after other projects are built” put “compile” in Projects to watch and select “Trigger even if the build fails” option
Click on apply and save
Similarly, do it for “Test” job as well.
right click on “Test” and select “configure”
Go to “Build Triggers” tab select “Build after other projects are built” put “Code_review” in Projects to watch and select “Trigger even if the build fails” option
Click on apply and save
Now go back to dashboard and click on Edureka_pipeline tab next to All tab
Click on Run




**Disadvantages of single Jenkins server**




To Address The Shortcomings Of Single Jenkins Server, Jenkins Distributed Architecture Was Introduced




**How to add Jenkins slaves in Jinkins?**
Click on “Manage Jenkins” on left -> click on “Manage Nodes” -> click on “New Node” on left -> select “Permanent Agent” and give Node name “Slave_1”
Click on OK.
Give Remote root directory /home/edureka
Select launch method as “Launch slave agents via SSH”
Give host name: <ip address> Note: go to slave machine and run command “ifconfg” to get ip address
Click on “Add” dropdown for credentials
Username: root
Password: xyz
Click on Add
Select the root/**** next to the credentials lable and before Add drop down.
Click on Save.

Click on the slave_1 to see the logs
Click on Log on left
You can see the logs… now it is online

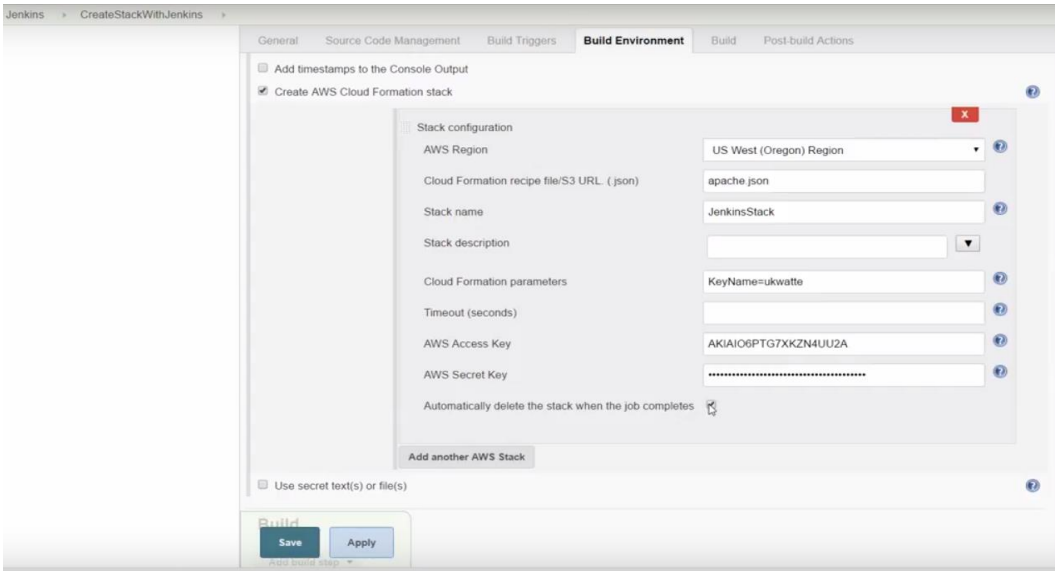
Go to slave_1 box
$ sudo ls /home/edureka
You will see slave.jar file, that means we are successfully added the Jenkins slave to the Jenkins master.
$

## How to Automate Stack creation in CloudFormation using Jenkins

https://www.youtube.com/watch?v=eHau6ZOi9tw&t=2s
byTharaka Ukwatte

**Steps:**

1. launch Jenkins dashboard and click on “new item”
2. enter item name “CreateStackWithJenkins”
3. select “free style project” and click on OK
4. in general tab > select git under source code repo
5. repo url: https://github.com/ukwatte/cfs.git
6. creds: none > click add key > select jeinkins
7. add credentials: give username and password > description: git hub credentials
8. click on add
9. select ukwatte (git hub credentials) for the credentials
10. select “create aws cloud formation stack” under build environment. Fill other details – cloud formation parameters – get it from ec2 instance, aws access key and secret key saved previously in xls.
11. Click on save




12. Click on ‘build now’ from Jenkins dashboard. It started building the job.
13. Go to aws console and go to CF service and watch the ‘JenkinsStack’ creation in progress. Watch the events.

---

SCM tools– **AWS CodePipeLine**, git, Mercurial etc

## Jenkins Build Job on AWS Cloud with Dynamic AutoScaling of Slave Node with Build Job

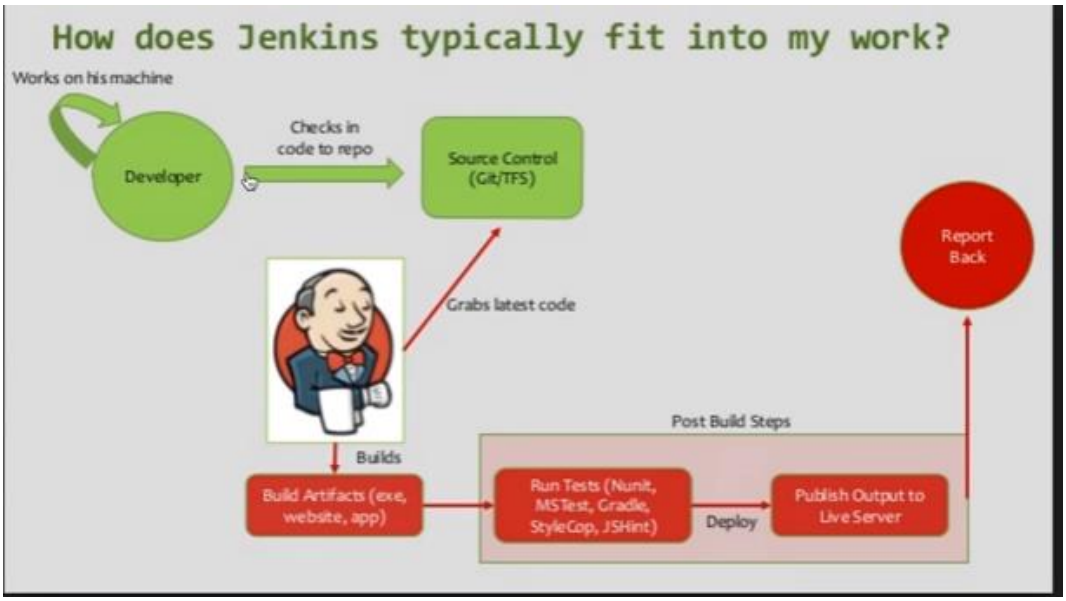
https://www.youtube.com/watch?v=61sPMSXpt9Y

this is NOT good video, still you can watch how he created a maven job and that created a slave ec2 instance from AMI and ran the job on that slave ec2 instance and logged in to the ec2 instance and see the workspace created for that maven project.

---

## Jenkins Beginner Tutorial 1 - Introduction and Getting Started

https://www.youtube.com/watch?v=89yWXXIOisk

- Jenkins is java based application, hence it is planform independent.
- Used for CI & CD
- 


### Jenkins installation:

Step 1 : Download Jenkins war file - https://jenkins.io/ (Jenkins.war file downloaded)
Step 2 : Place the war file into any location on your system (desktop>jenkins>jenkins.war)
Step 3 : Go to command prompt (windows) | terminal (mac)
-goto folder where jenkins.war is placed and type
cmd> java -jar jenkins.war

It is extracting the war file and winstone is the server, it will take some time, once this entire process is complete, Jenkins will up and running, you can access it through a browser on port 8080. It will also give the password for admin. Copy this password for first time usage on Jenkins.

Step 4 : Go to browser - http://localhost:8080 (Jenkins window should show up)

Step 5 : install required plugins (suggested plugins)

Step 6 : get started with Jenkins (Username: admin / saved password)

If you want to see all the plugins where it got installed.  Go to .jenkins folder, there you can see all the configuration, plugins folder

# Jenkins Beginner Tutorial 2 - How to setup Jenkins on Tomcat

https://www.youtube.com/watch?v=Fi9pgrMlQZY

**Why should we deploy Jenkins on tomcat?**

Running Jenkins standalone (jetty/winstone) vs Running Jenkins on Tomcat (servlet container)

Pre-requisite
***Tomcat 5 or above***
***java(7 or above) should be available***
(https://www.youtube.com/watch?v=FqpmH...)

**Step 1 : Download Tomcat**

Google “tomcat” -> select “tomcat.apache.org” site -> on left Downloads tomcat 8 > birnary distributions 64-bit windows zip

Step 2 : Unzip and place tomcat folder at any location

Step 3 : Copy/Place the jenkins.war file inside **tomcat/webapps** folder

Step 4 : Go to command prompt (cmd) - windows | terminal - mac
- goto tomcat/bin directory
- make all files executable : chmod +x *.sh

In case of windows this command will not work. This step might **not** be needed if you are on windows. Else you can right click on the file/folder go to security tab and change the permissions.

Can watch -
**TOMCAT - How to install and run on Windows | Change Port | Run app**
https://youtu.be/ZIIkrn6LHW0?list=PLh...

Step 5 : Start Tomcat : ./startup.sh ( run from tomcat/bin)
(to shutdown tomcat : ./shutdown.sh)

When trying to start Tomcat Server through cmd prompt using **'startup.bat'** getting error as- "JRE_HOME variable is not defined correctly. The environment variable is needed to Run this program" Defined Environment path.

Set the environment variables as:

Go to explorer > PC > properties > advanced system settings > environmental variables > USER VARIABLES

- CATALINA_HOME=C:\Program Files\Java\apache-tomcat-7.0.59\apache-tomcat-7.0.59 *(address where your Apache Tomcat is)*
- JAVA_HOME=C:\Program Files\Java\jdk1.8.0_25; *(address where your JDK is)*
- JRE_Home=C:\Program Files\Java\jre1.8.0_25; *(address where your JRE is)*
- CLASSPATH=%JAVA_HOME%\bin;%JRE_HOME%\bin;%CATALINA_HOME%\lib

**How to change the default tomcat port 8080?**

Go to **tomcat/conf/server.xml** – open this file and search for "**connector port**" change the port number 8080 to some other like 9090. Save it. Start/Restart the tomcat server - double click on tomcat/bin/startup.bat in explorer or run from cmd prompt.

Check in browser http://localhost:9090 -> it will open tomcat server home page

**How to run an application on Tomcat?**

Copy your application .war file in **webapps** folder under **tomcat &** restart tomcat server.

Step 6 : verify if tomcat started : browser - http://localhost:8080 (tomcat running on port 8080)

Step 7 : verify if Jenkins is running on tomcat : http://localhost:8080/jenkins (jenkins.war file inside tomcat hence it is running on top of tomcat)

if you want to run the standalone as well. Jenkins default port also 8080, tomcat already running on 8080, now we need to change Jenkins port to different port.

***** To start jenkins (standalone) on a diff port*****

cmd> **java -jar jenkins.war --httpPort=9090**

Test it by typing localhost:9090 on the browser

Create a test job on standalone Jenkins. Job is "testjob1"

Create a test don on tomcat Jenkins. Job is "testjob2"

Restart both the servers

^C to stop and run the Jenkins server again

Cmd prompt > **java -jar Jenkins.war –httpPort=9090**

To Shut down tomcat - Goto tomcat/bin and run ./shutdown.sh

Again, start tomcat - **./startup.sh**

Refresh the both the browsers 8080 and 9090

Both the jobs are available on both browsers.

# Jenkins Beginner Tutorial 3 - How to change Home Directory

https://www.youtube.com/watch?v=m47MWSXNslg

whenever we deploy Jenkins on any system there is a folder created default folder **.jenkins** in default profile of the system.

Jenkins Home Directory contains:

1. All configurations
2. Plugins
3. Jobs details
4. Logs

**Why to move the home directory to another?**

- mostly it will install in user profile where disk space is limited. Lot of jobs, plugins, configurations to be set on Jenkins it required good amount of disk space.

-To move jenkins home directory to a location that has enough disk space
-may be Project requirements

**Step 1 : Check your current home directory**

$cd Jenkins (where jenkins.war file placed)

$java -jar jenkins.war (it will start Jenkins on default port 8080)
go to browser and launch Jenkins (http://localhost:8080)

Manage Jenkins > Configure System > see the Home Directory /users/Raghav/.jenkins

Now I want to change this.

**Step 2 : Create a new folder (which will be new Jenkins home directory)**

create jeinkinsHome directory

**Step 3 : Copy all data from old directory to new directory**

copy all contents in .jenkins folder to newly created jenkinsHome directory.

**Step 4 : change environment variable - JENKINS_HOME and set to new directory**

Mac - goto terminaal
export JENKINS_HOME=/Users/raghav/Desktop/Tools/Jenkins/JenkinsHome/

windows – explorer > My computer > right click and select properties > Advanced system settings > Advanced tab on “System properties window > Environment Variables > select JENKINS_HOME under System variables > edit > give new Jenkinshome directory path in the variable value.

**Step 5 : restart Jenkins**

You can restart Jenkins where you have started at command prompt and do ^C and user start command as $ java -jar jenkins.war

Or other way is -

Go to browser type localhost:8080/restart

Manage Jenkins > configure system > see the home directory

If you are NOT seeing the latest home directory in mac, you need to update the export statement in .bash_profile

In the browser – **localhost:8080/systemInfo** → you can see al system properties, environment variables, plugins etc..

---

To see hidden files/folders on mac:
1. Exit Finder
2. On terminal - defaults write com.apple.finder AppleShowAllFiles TRUE
3. Press option/alt - right click finder - relaunch

# Jenkins Beginner Tutorial 4 - How to use CLI (command line interface)

https://www.youtube.com/watch?v=ooA8RS3hC6k

**why should we move to command line?**

it is easier, faster, more convenient, memory mgt(efficient), continuous integration or setup a build process then you have to go to the command line.

**Step 1 : start Jenkins**
$cd Jenkins

$java -jar jenkins.war –httpPort=9090 (if you want to run on different port, otherwise default is 8080)

**Step 2 : goto Manage Jenkins > Configure Global Security > check enable security > apply and save**

**Step 3 : goto - http://localhost:8080/cli/**
you see Jenkins CLI you can read more by clicking on “the Wiki” .

**Step 4 : download jenkins-cli jar. Place at any location.**

**Step 5 : test the jenkins command line is working**

Go to the location where Jenkins-cli.jar is placed, run

$java -jar jenkins-cli -s http://localhost:8080 help (http://localhost:8080 is your Jenkins url)

This will ask passphase. To get this, go to Jenkins home page admin > configure > see SSH Public Keys, you can copy and paste in passphase.

**Test the safe-restart**

$ java -jar Jenkins-cli.jar -s http://localhost:8080/safe-restart

ERROR: anonymous is missing the Overall/Administer permission

$

Go to Jenkins dashboard > manage Jenkins > Configure Global Security > Authorization > select “Anyone can do anything” > save

Run the command again.

$ java -jar Jenkins-cli.jar -s http://localhost:8080/ safe-restart.

Refresh and see the browser that is restarted.

## Jenkins Beginner Tutorial 5 - How to create Users + Manage + Assign Roles

https://www.youtube.com/watch?v=QvFungzXI5s

Jenkins authentication and authorization

Today we will learn:
_____
How to create New Users
How to configure users
How to create new roles
How to assign users to roles
**How to Control user access on projects**

**Step 1 : Create new users**
start jenkins

$ cd Jenkins

$java -jar Jenkins.war

Go to dashboard (http://localhost:8080)

Login with admin.  However if you want to get the password go to

Manage Jenkins > configure system > see the home directory > go to Jenkins home directory and see the folder called “secrets” inside it you see the initialAdminPassword file.

**To create new users –**

Manage jenkins > Manage users > create user > give user1, set pwd, give name  > click create user.

Do the same way to create user2

Logout from admin and try login with user1.  Login successful and you see the top right corner as User1.

**Step 2 : Configure users (top right corner user1 drop down and select configure)**

Here you can give full name, description, you can get api token(see in future sessions), you can change password, in case you want to setup any SSH public keys for authentication you can setup here then you can apply and save.

**Step 3 : Create and manage user roles**

**Roles Strategy Plugin - download - restart jenkins**
Go to browser and search for “Jenkins role strategy plugin”. Click on wiki.jenkins site. Get the latest release download. You can see role-strategy.hpi file downloaded. Copy this file and place in Jenkins/jenkinsHome/plugins then restart the Jenkins. This is one way.. other way is

Go to Jenkins dashboard > manage Jenkins > manage plugins > available tab> filter “roles” select download now and install after restart. Otherwise after installing go to Advanced tab > Upload plugin > choose plugin ext .hpi file and upload.

**Note:** If you are not able to install pulgins due to proxy issues. Go to IE -> Internet Options -> Connections -> LAN Setting -> get the proxy address -> put in browser -> it will download a file -> open that file -> you see something like this “var proxy_yes = "PROXY cwybcproxy.us.dnb.com:8080";” -> from this take url and port and give in Manage Jenkins -> Plugin Manager -> Advanced tab -> server name(url) and Port(port) and do not give any user name and password. Click on advanced and give some test url like https://www.google.com/ and click on validate proxy. It should be success then click on Submit.

## How to use it?

**Step 4 : Jenkins login as admin. Manage Jenkins - Configure Global Security – make sure enable** security is checked > Authorization -> Role Based Strategy (available this after plugin is installed) then apply and save. (All these changes you have to do with admin user)

Now try login as user1. You see the Access denied saying “user1 is missing the overall/read permissions”, because now we are using role based strategy authorization.

**Step 5 : Create Roles and Assign roles to users**
How to grant permissions to this user?

Login back with admin, manage Jenkins > manage and assign roles > manage roles > you will see 1)global roles, 2)project roles, 3)slave roles. Add employee as a global role and give overall read, view access(check all permissions) then apply.

Add developer role to Project roles, pattern is Dev.* and click add. Give all permissions (check all). Apply. Again create role called tester, pattern is Test.*(check all) apply and save.

Go to Manage and assign roles > Assign roles > Global roles > add user 1 and user 2, both the users check employee role and apply. Come back to Project role > add user 1 and user 2 > check user 1 as developer and user 2 as tester roles. Apply and save.

Create a new project “Devproject1” free style > save it.

Create another project Testproject1 > free style > save it.

Logout as admin.

**Step 6 : Validate authorization and authentication are working properly**

Login as user 1 -> he can see only Dev project. Also you can notice he will not have manage Jenkins etc permissions. Now logout

Login as user 2 -> he can see only Test project. Also you can notice he will not see manage Jenkins etc permission. Now logout

# Jenkins Beginner Tutorial 6 - Basic Configurations

https://www.youtube.com/watch?v=Cr8XSljgEPI

**Step 1 : Go to Manage Jenkins - Configure System**

See the Home directory -> gives the info about your home directory.

Click on advanced:

Workspace root directory -> where Jenkins will store workspaces for builds that are executed on master.

Build record root directory ->

System message -> put something like “<h1>This is an example system message</h1>” > apply

Refresh the window you can see the message on your Jenkins dashboard. If you are not seeing header format. You can go config global security > Markup Formatter > Safe HTML.

# of executors -> the number of parallel jobs this particular Jenkins instance can execute, it is max of 5.

Labels -> used to restrict the jobs to run on particular labeled machines.

Quit period -> 10 sleep interval between the jobs run.

SCM checkout retry count -> 5

Restrict project naming -> pattern > Test* (you can restrict give the job name as Test*)

Global properties - > key1 value1

Jenkins location ->

SSH Server ->

Shell -> default jenkins use bash shell

**Step 2 : Get a basic understanding of common/basic configurations**

# Jenkins Beginner Tutorial 7 - Getting started with JOBS

https://www.youtube.com/watch?v=63HEKFh8T2c

Today we will learn:
_____

1. How to create a basic JOB in Jenkins
2. Basic job configurations
2. How to run the Job remotely
4. How to chain Job Execution

**Step 1 : Jenkins - New Item - add details**

Jenkins dashboard > New item > test 1 > free style project > General > description : <h2> This is test project one </h2>

Source code management > need to add the source code plugins as per our usage.

Build Triggers > 4 options 1) Trigger builds remotely (eg. From scripts) 2) Build after other projects are build 3) Build periodically -> we can assign some expression, based on the expression the job will be executed some particular interval. Corn job format 4) Poll SCM -> as and when code is committed in SCM like git/svn build will trigger

Build -> You will see the options based on the plugin installed.

Post-build actions -> what you want to do after build is complete, you can run regression or some other tests etc..

Save it, then it will go to dashboard. See the test 1 job. Status is gray means it is not build yet. Right click and click on build now. Go to the dashboard and go console output

**Step 2 : How to trigger the job remotely**
lets go to the project test 1 > configure > build triggers > check trigger build remotely (eg. From scripts). You can see the url below to the authentication token. Copy that url and run from any remote machine. <give your Jenkins url>/job/test1/build?<token_name ie. your password>. You can go and check the job in dashboard that job test 1 ran once again.

**Step 3 : How to chain job executions**

Create test 2 project > free style >build shell cmd: pwd.

Create test 3 project > free style > build shell cmd: pwd

Save.

Now we have test 1, test 2, test 3. Now go to test 2 > configure > build trigger > check build after other projects are built > project to watch: test 1 ( you can give multiple projects). Select trigger only build is stable. Apply. Post build actions > build other projects > project to build: test 3.

Run test1 now. See build queue on dashboard. Once test 1 is complete test 2 will execute then test 3 will execute.

# Jenkins Beginner Tutorial 8 - Jenkins integration with GIT (SCM)

https://www.youtube.com/watch?v=bGqS0f4Utn4

**Today we will learn**

**1. Create a java program and run it through command line**
**2. Create a Jenkins job to run the java program**
**3. Add this program/project to Git**
**4. Jenkins - add git plugin**
**5. Configure Jenkins job to trigger the execution when a change is pushed to GitHub**

---

**Step 1 : Create a java program**

Hello.java

```
public class Hello {
        public statuc void main(String[] args) {
                for (int i=1;i<=10;i++) {
                        System.out.println("Hello World " +i);
                }
        }
}
```

Save the file and move it to a folder called "JavaProject1"

Cmd> cd JavaProject1

Cmd> javac Hello.java

Cmd> java Hello

Hello world 1

:

Hello world 10

If you are using eclipse, you can see the project package above to the location as "src" . take the location and run the java program.

**Step 2 : Compile from command line**
javac "name of .java class" java "name of class without extension .java"

**Step 3 : Create a Jenkins job to run this program**

Cmd> go to Jenkins directory

Cmd> java -jar jenkins.war --httpPort=9191

Browser: localhost:9191

New item > helloworld > free style project > OK

Build > execute shell > command:

cd /users/Raghav/desktop/javaproject1(java file stored in the folder “javaproject1”)

javac Hello.java

java Hello

apply & save > build now > success

see console output.

**Step 4 : Add the project to Git and GitHub**

Go to the location of the folder

Cmd > cd javaprject1

Cmd> git init

Cmd> git status

Hello.class

Hello.java

Cmd> git add .

Hello.class

Hello.java

Cmd> git commit -m “added helloworld program”

Go to github.com > start a project > repo name: HelloWorld > create the repository. Copy the location of the repository.

Cmd> git remote add origin https://github.com/RaghavAutomation/HelloWorld.git (paste the copied locaiton of the repository)

Cmd> git push -u origin master

Cmd>

Go to github.com and refresh the repository. Now you see the files because you have committed and added to github.

**Step 5 : Configure Jenkins job to trigger build on Git commit**

Install git plugin in Jenkins

Manage Jenkins > Manage plugins > available > filter git plugin > download and install

Go to HelloWorld job > source code mgt > git > repository: https://github..../Helloword.git

Build Triggers > check Poll SCM > schedule: * * * * * (it will check repo every minute)

Apply & save.

Cmd > dir > readme.txt

Or $ touch index.html

Cmd> git status

Index.html

Readme.txt

Cmd> git add .

Cmd> git status

Index.html

Readme.txt

Cmd> git commit -m “new files added”

Cmd> git push -u origin master

Cmd

Go to Jenkins dashboard, you will notice build triggered. Go to console output. You can see “started by an SCM change”

For Reference:
---
Git and GitHub Beginner Tutorial 1 - Introduction - https://youtu.be/-U-eUHI6euM?list=PLh...

Git and GitHub Beginner Tutorial 2 - Getting started - Install Git (mac)
- https://youtu.be/0Icla6TVNNo?list=PLh...

Git and GitHub Beginner Tutorial 3 - Getting started - Install Git windows
- https://youtu.be/sBTAkHOxvOk?list=PLh...

## Jenkins Beginner Tutorial 9 - How to use CATLIGHT (Jenkins Build Monitor)

https://www.youtube.com/watch?v=yC5w-CVXak8

Today we will learn : How to use CATLIGHT (status notifier for jenkins)

**What is CATLIGHT** https://catlight.io/
- Status notifier for developers

- Catlight will notify you when builds, bugs & tasks need your attention
- Very handy and useful when you have to manage multiple jobs
- It will give real time notifications

**How to use CATLIGHT**
download the installable.  Extract and double click on Catlight.

Select Jenkins > copy the Jenkins url > paste Server url : http://localhost:8080 > use creds > save

You can see all the projects which are in Jenkins.

Select all & save

Now you see all jobs are monitored

Now run the job and watch the notification on your screen.

Go to catlight menu and settings > you can see the setting for notifications, dashboard, branches, proxy, advanced

REFERENCES:
Catlight website:
https://catlight.io/

Downloads link:
https://catlight.io/downloads

Support link:
http://catlight.helprace.com/

Blog link:
http://blog.catlight.io/

---

Jenkins Playlist:
https://www.youtube.com/playlist?list...

---

# Jenkins Beginner Tutorial 10 - What is Automated Deployment (Step by Step)

https://www.youtube.com/watch?v=HkLXbFDlMtU

**What is Automated Deployment ?**

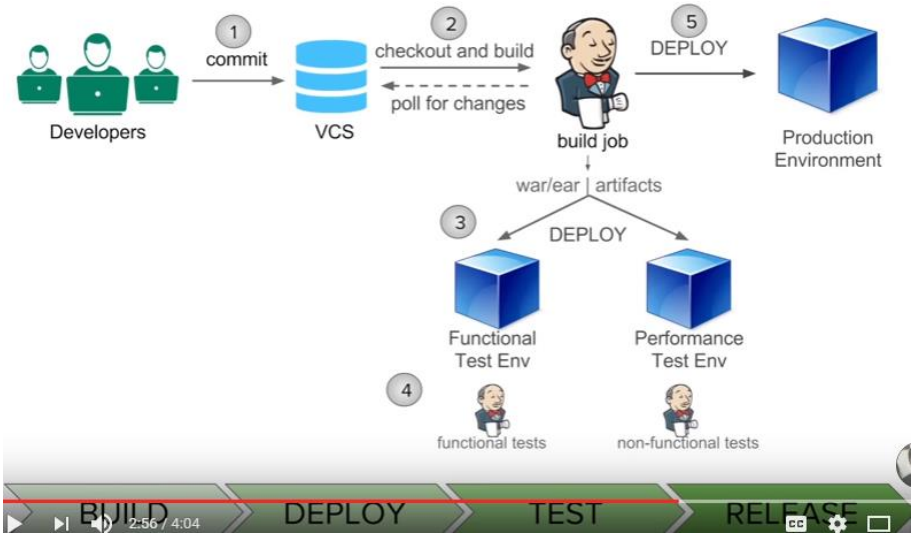
Main stages in Continuous delivery and deployment pipeline. 4 major stages.
**1. BUILD**
**2. DEPLOY**
**3. TEST**
**4. RELEASE**

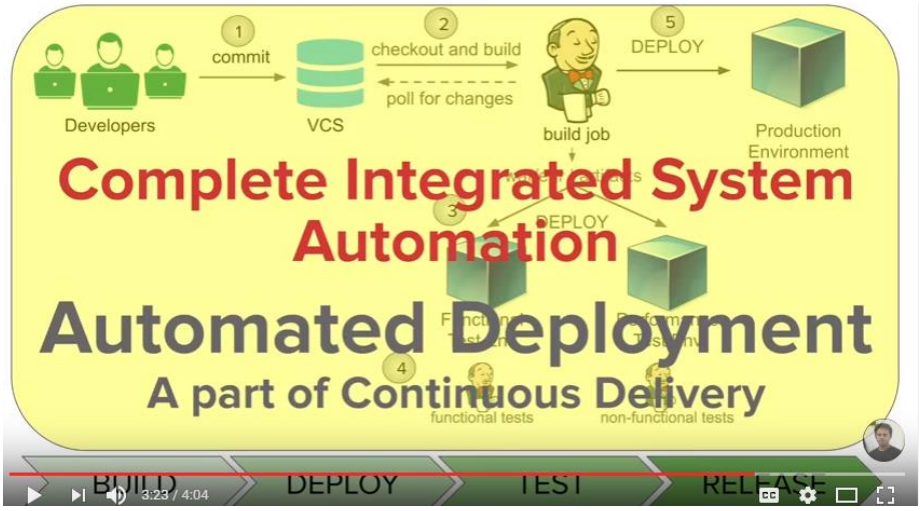
Each stage we have corresponding jobs and all these jobs are chained. Meaning, when BUILD jobs are successful only then DEPLOY jobs will get triggered, when DEPLOY jobs are successful then only TEST jobs are triggered, when TEST jobs are successful then RELEASE jobs are triggered and so on.

All jobs are chained in a continuous integrated system.

After every job complete we will have a notification which will tell us about the status of the job where any issues or any other information about the job

Let's see ... A real world Project Setup




Now we define “Automated Deployment is the process of Automating the deployment process in a Continuous Delivery system”

**Build -> Deploy -> Test -> Release**

---

References:
https://www.slideshare.net/MartinEtma...

Jenkins Playlist:
https://www.youtube.com/playlist?list...

# Jenkins Beginner Tutorial 11 - How to do Automated Deployment (Step by Step)

https://www.youtube.com/watch?v=j5D8SLxn6YA

**How to do Automated Deployments with Jenkins?**

**Step 1: Start Jenkins**
$cd <path>/Jenkins
$java -jar jenkins.war –httpPort 9090
Jenkins fully up and running
Go to browser and type localhost:9090, it will show the Jenkins dashboard.

**Step 2: Install Plugin (Deploy Plugin)**

Go to the browser search for Jenkins deploy plugin

https://wiki.jenkins-ci.org/display/J...
click on the latest release and download the deply plugin

Now go to Jenkins dashboard > Manage Jenkins> Manage plugins >

Two ways of installing the plugins

1. Go to Advanced, upload plugin section and upload the file (downloaded plugin in the above)
2. Easiest way will be Available tab, search for deploy, select “Deploy to container plugin” click on “Download now and install after restart”. This will download, install your plugin and restart your Jenkins.

**Step 3: Create a Build JOB in Jenkins**
New Item> project name: AutomatedDeploymentTest > Freestyle project > OK

For the demo go to the browser and search for the “sample war file download”. It will give the tomcat website and download the sample war file.

**Step 4: Add Post-build Action -> Deploy war/ear to container (add values to the fields)**
sample war file - https://tomcat.apache.org/tomcat-6.0-...
BUILD: command: date

Post Build Actions: select “Deploy war/ear to a container (this you will see only after installing deploy plugin)

WAR/EAR files: **/*.war

Apply now comeback again to fill the rest of the fields.

Go to Jenkins > Manage Jenkins > Configure System > click on “Advanced” next to Home directory, it will show workspace root directory, in the workspace directory put the sample.war file.

Go to cmd prompt > cd <Jenkins path>/jenkinshome/workspace/AutomationDeploymentProject > in this location copy the sample.war file.

Now come back to Jenkins

Context path: sample.war (give name of your war file)

Containers: select Tomcat 7.x (it works for tomcat 8 also)

You can watch tomcat server install and run videos

Now we need to give tomcat username, password and url. So we need to go to the “tomcat-users.xml” file and add a user there.

**Step 5: In tomcat-users.xml add user for DEPLOYMENT**
**user username="deployer" password="deployer" roles="manager-script" /**
Go to tomcat path > go to conf > here you see **tomcat-users.xml** and add the user by copy and paste existed user and update as required.

<user username=”deployer” password=”deployer” roles=”manager-script”/>

Save the file

Start the tomcat as well.

$ cd <tomcat dir>/bin
$./startup.sh
To test it, go to browser and type localhost:8080, now you see tomcat server up and running.

Go back to Jenkins and finish the configurations.

Manager user name: deployer (added this user in tomcat-users.xml file)

Manager password: deployer

Tomcat URL: localhost:8080

Click on Apply & save.

**Step 6: Run and Validate**
go to Jenkins dashboard > Build Now > console output > you can watch deploying the sample.war file to container tomcat 7.x remote
Deployed
Finished: success

Now go to the browser, type **localhost**:8080/sample/
This application is now deployed.

**HELPFUL TIPS:**

1. If Jenkins and tomcat running on the same system, ensure you use different ports. In this example we used port 9090 for Jenkins and 8080 for tomcat.
2. Tomcat 7.x container will work for Tomcat 8 container.
3. If you get the error like connection refused; make sure Tomcat server is running and accessible from Jenkins.

**What is Automated Deployment**
https://www.youtube.com/watch?v=HkLXb...

**How to install and setup TOMCAT**
https://www.youtube.com/playlist?list...

# Jenkins Beginner Tutorial 12 - Notifications - How to send Email from Jenkins

https://www.youtube.com/watch?v=DULs4Wq4xMg

Automation Step-by-Step - Raghav Pal

**How to send the emails from jenkins?**

Jenkins dashboard > Manage Jenkins > Configure System > go down and see for Email Notifications

**To find your SMTP server and port details**
https://www.arclab.com/en/kb/email/list-of-smtp-and-pop3-servers-mailserver-list.html

SMTP server: smtp.gmail.com (in this case we are using gmail)

Click on advanced section

Select smtp authentication > user name: localstakeqa@gmail.com (give the email id where you want to send the email notifications)

Password: password

Check User SSL

SMTP port: 465 or 587 or 25. You can test with any of the port.

Check “Test configuration by sending test email”, test e-mail recipient: go2raghav10@gmail.com

Hit “test configuration” button.

Email was sent successfully.

If any issues, login to google account > my account > Signin & Security > at the end see “allow less secure apps” : ON

Now you go to any of your jobs in the Jenkins dashboard > configure > Post build actions > email notification > recipient: any of the email > save

This way you can set the notifications.

**There are some custom plugs**

1) **Notification plugin** -> send the notifications in JSON and XML format

2) **Extreme notification plugin** -> in this you can send webhook (it is basically a http post, we can configure on any event that http post can sent some particular URL. This you can configure with this plugin.

3) **Email-ext plugin** -> It gives some advanced features. You can customize when an email sent, who should receive it, and what the email says.

**References:**
https://wiki.jenkins-ci.org/display/JENKINS/Notification+Plugin
https://wiki.jenkins-ci.org/display/JENKINS/Extreme+Notification+Plugin
Jenkins Playlist:
https://www.youtube.com/playlist?list...

# Jenkins Beginner Tutorial 13 - What is Pipeline in Jenkins (DevOps)

https://www.youtube.com/watch?v=DTRfG_gMLeU

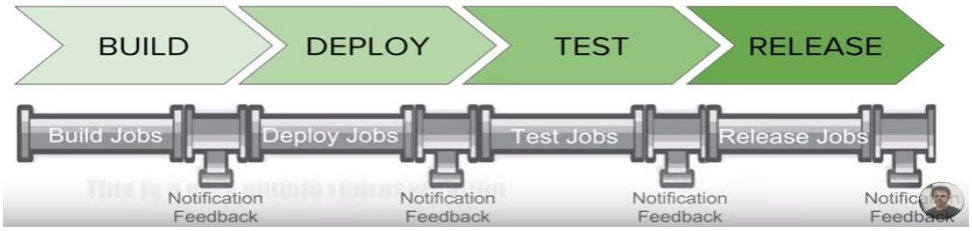
**What is Pipeline in Jenkins?**

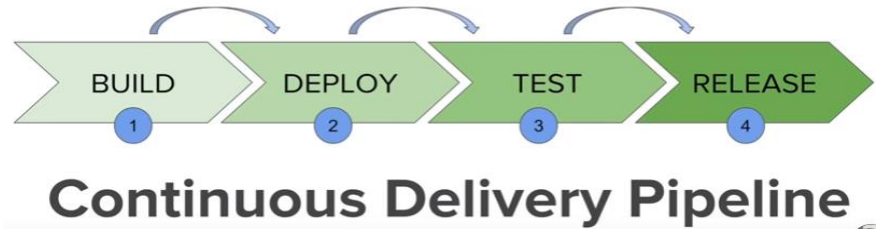
**Pipeline** - is a workflow with group of events or jobs that are chained and integrated with each other in sequence.
Every job in a pipeline has some dependency on one or more other jobs

Let us understand... in an easy way !
(Watch the representations in the video)

**Continuous Delivery** pipeline consists for **4 major stages 1)Build 2)Deploy 3)Test 4)Release**

Every stage will have some jobs, these jobs are integrated with each other, work in a sequence we call it as Continuous delivery pipeline.




Pipeline is a concept, in Jenkins or devops as a whole, mostly used continuous delivery, deployment, and integration. However, pipeline is a system, where multiple sections or jobs can be integrated in a sequence and work as a workflow and it can be implemented in anyways as you like.

References -
https://tex.stackexchange.com/questio...
https://readtiger.com/wkp/en/Pump

# Jenkins Beginner Tutorial 14 - How to setup DELIVERY PIPELINE in Jenkins (Step by Step)

https://www.youtube.com/watch?v=ndLbn24OwQg

**How to setup Delivery Pipeline in Jenkins?**

**Step 1: Chain required jobs in sequence (**Add upstream/downstream jobs)

Upstream – a job to be executed before current job

1. Create a sample jobs in Jenkins
2. New Item > Create **SampleBuildJob** > Freestyle project > OK > BUILD > Execute Shell Command: date;echo “Build job completed successfully” > Apply & Save
3. New Item > Create **SampleDeployJob** > Freestyle project > OK > BUILD > Execute Shell Command: date;echo “Deploy job completed successfully” > Apply & Save
4. New Item > Create **SampleTestJob** > Freestyle project > OK > BUILD > Execute Shell Command: date;echo “Test job completed successfully” > Apply & Save
5. Run all 3 jobs from console to check everything is working good.
6. **Now we have to chain our jobs.**
7. Go to SampleBuildJob > configure > Post build actions > Build other projects(manual step) or Build Triggers > check Build after other projects are built
8. Go to the SampleDeployJob > configure > Build Triggers > check Build after other projects are built > Projects to watch : SampleBuildJob > select “Trigger only if build is stable” > apply & save
9. Go to SampleTestJob > configure > Build Triggers > check Build after other projects are built > Projects to watch : SampleDeployJob > select “Trigger only if build is stable” > apply & save
10. Now all these 3 jobs are chained.
11. Run the first job SampleBuildJob from dashboard > build now
12. You will notice all 3 jobs are triggered one after another.

**Step 2 : Install “Delivery Pipeline Plugin”**
Browse “delivery pipeline plugin”. You will see a wiki page link. We have 2 ways to install plugins.
1)download latest and add to our jenkins
2) directly add the plugins from Jenkins dashboard.

Select Manage Jenkins > Manage Plugins > Available > filter “delivery pipeline” > check “Delivery Pipeline plugin” > download and install after restart

**Step 3: Add Delivery Pipeline View; configure the view**

1. Go to the Jenkins dashboard > click + sign > select Delivery Pipeline view > view name: TestDeliveryPipeline > OK > Pipeline – Add component > Initial job: select SampleBuildJob > Name: SampleBuildJob > apply & save.
2. It will show the details of the past runs
3. click “view fullscreen” > see the full screen
4. Click “edit view” > you can select different options and see the pipeline output

**Step 4: Run and Validate**

Already you have seen already.

**Helpful Tips:**

1. Update the options in Delivery Pipeline view as per your need.

# Jenkins Beginner Tutorial 15 - How to setup BUILD PIPELINE in Jenkins (Step by Step)

https://www.youtube.com/watch?v=zf6ogW0HKLY

Today we will learn :

**How to setup BUILD PIPELINE in Jenkins ?**

**Step 1: Chain required jobs in sequence. Add upstream/downstream jobs**

**How to chain jobs? Watch**
https://www.youtube.com/watch?v=ndLbn...
already you have seen in tutorial 14

**Step 2: Install "Build Pipeline Plugin"**

Go to Jenkins dashboard > Manage Jenkins > Manage Plugins > Available > filter: Build pipeline > select "Build Pipeline plugin" > click on download and install

**Step 3: Add "Build Pipeline View" - configure the view**
Go to Jenkins dashboard > click on + > view name : BuildPipelineTest > select Build pipeline view > OK

Configuration view > give any description > Pipeline flow – Select initial job: SampleBuildJob > apply & ok.

Build Pipeline window display. Currently you are seeing only last run. Click on Configure > Display options – No of Displayed Builds: 5 > apply & ok

Explore all other options

**Step 4: Run and Validate**

Already done this

# Jenkins Beginner Tutorial 16 - What is BLUE OCEAN (How to get started)

https://www.youtube.com/watch?v=Di1DjW8Qxb0

**What is Jenkins Blue Ocean?**
**How to get started with Blue Ocean?**

Blue Ocean is a new User Interface (UI) and User Experience (UX) for Jenkins
It is designed to make Jenkins UI more efficient (reduces clutter and increases the clarity)

In simple words:
Blue Ocean is a new User Interface for Jenkins and provides an interactive view for Jenkins Pipeline (and jobs)

**How to get Blue Ocean?**

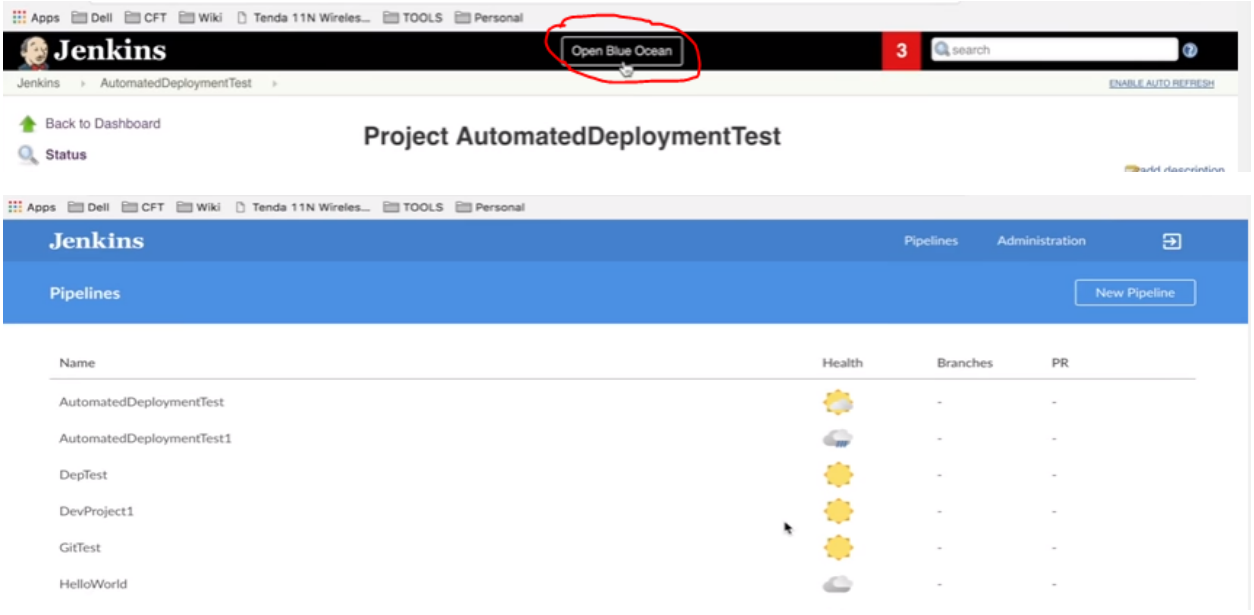
It is very very easy.
Ensure you have Jenkins 2.7 or above and follow the steps:
Go to Jenkins dashboard, bottom right corner there you can see Jenkins version ie 2.7 or aove..

**Step 1: Install Blue Ocean Plugin**
Jenkins dashboard > manage Jenkins > Manage plugins > available > filter: blue ocean > check Blue Ocean > click on Download now and Install after restart.

You might see some failures.  You must restart Jenkins

**Step 2: Switch to Blue Ocean View**

Blue Ocean on top of Jenkins window. Click on it.




Click on any job it will take you to the job details.
Explore options in this view
Click on arrow box on right top corner to go to Classic view of Jenkins.

# How To Install Maven On Windows - Beginner Tutorial

https://www.youtube.com/watch?v=K9U-5aa8VwE&list=PLhW3qG5bs-L8XkBrI-G5aTUo6QwEEoVcj

**How to install Maven on Windows**

Step-1 : Check if maven is already installed goto cmd> mvn -version

Step-2 : Download maven from internet https://maven.apache.org/download.cgi

Step-3 : Extract the zip file

Step-4 : Set environment variables

M2_HOME

MAVEN_HOME

PATH

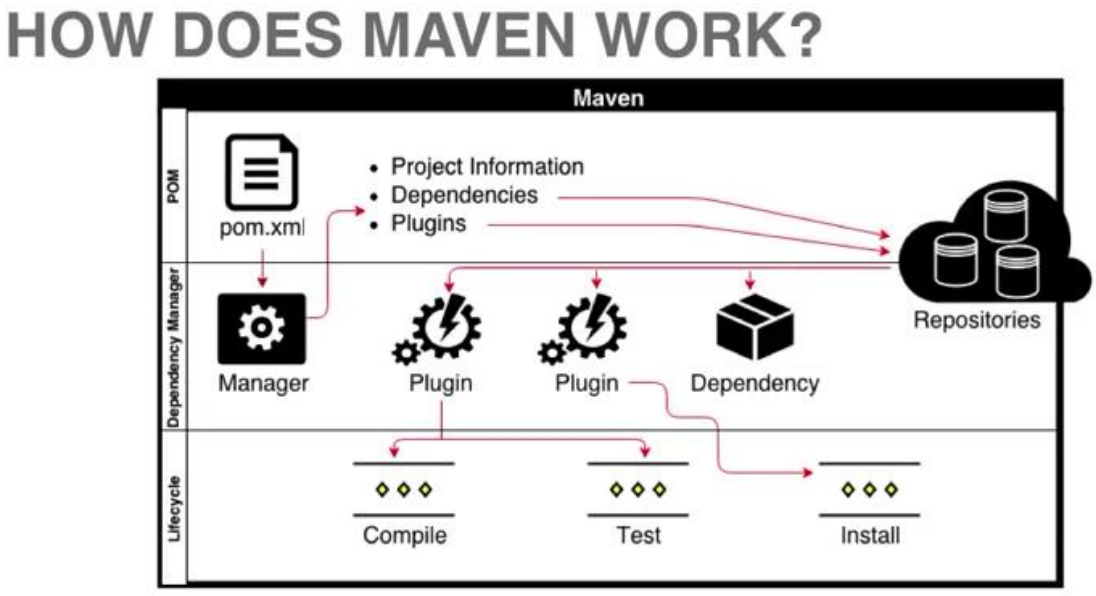
To do this –

Go to explorer > My PC > Properties > Advanced System Settings > System properties window > click on environment variables > **System variables** > new – variable name: **M2_HOME value: c:\..\apache-maven-3.3.9** > OK > New – name: **MAVEN_HOME value: c:\..\apache-maven-3.3.9** > OK

select PATH in system variable and edit – do not delete anything, go to end put ; give location of the bin folder; (or you can **%M2_HOME%\bin**) > OK > OK

Step-5 : Check again if maven is installed

Cmd> mvn -version




- Maven follows convention over configuration, need not required to write lot of xml code etc.
- **Build** means compile the code, preparing **packages** out of compiled classes, **deploying** them to the servers, running **tests**. Maven automates all these with minimal conventions.

- Java space, we need to follow the project structure as follows.
    - src/main/java (main source code)
    - src/main/resources (property files, xml configurations)
    - src/test/java (junit tests)
    - src/test/resources
    - target (maven combines all class files and resources as packages and store in target)

- Once you follow all these conventions, Maven automatically compile our classes, then automatically bundle those classes into **jar** files, then automatically run our junit tests under the test folder etc.. So it combines all the **class files and resource into the Target directory** and it does the rest of compilation, bundling classes into jar and even it create war files if the project is web application project. Web application project the folder structure is slightly different.

- Maven uses POM.xml (in case of ANT it is BUILD.xml) – We simply declare 3 or 4 lines in POM.xml, that compile the project and build a jar file out of it.

- Cmd> **mvn clean install** -> it will compile the project, run your junit tests and it create a jar file

- You can **declare in dependencies in POM.xml** which may be in other 100s of projects, or internet it will automatically pull those dependencies and put them in your local machine (repos).

- Maven uses plugin framework, when you try to install plugins it automatically get the plugins on the fly and use them to create your build.

- Maven created build artifacts, dependencies will store into repositories, these repos may be local or remote. Ex: Nexus

- **jar**(java archive – it is collection .class files)
- **war**(web archive – web related technologies jsp, html, selvets, xml, css, js… all are zipped into single file called .war file (that makes easy to transportation/delivery/deployment)
- **ear**(enterprise archive – anything java/j2ee technologies ejb, servlets, jsp, jms, etc are zipped into single file called .ear file.

# Maven Tutorial for Beginners 1 – Introduction

https://www.youtube.com/watch?v=HBXxBJ-7LFw&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ

ProgrammingKnowledge

In this video series, we will learn Maven tutorial for beginners - Learn Apache Maven in simple and easy steps starting from Environment Setup, Build Life Cycle, Build profiles, Repositories, POM, Plug-ins, Eclispe IDE, Creating Project, Build & Test Project, External Dependencies, Project Documents, Project Templates, Build Automation, Dependency Management, Deployment Automation, Web Application NetBeans, IntelliJ IDEA

**apache maven introduction**
**maven phase vs goal**
**plugin repository**
**understanding maven**
**what is maven project**
**maven explained**
**maven tool**
**maven build tool**

Maven is a software build management and comprehension tool based on the concept of Project Object Model(POM) which can manage project build, reporting, and documentation from a central peace of information.

All build systems are essentially do the following:

1) Compile the source code
2) Copy source
3) Compile and run tests
4) Package the project
5) Deploy the project
6) Cleanup

**Maven developers want –**

1) A standard way of building the projects
2) A clear definition of what the project consists of
3) An easy way to publish project information and a way to share JARs across several projects.
4) As a result, the tool can be used for building and managing any Java based project.
5) All this intended to make use of day to day work of java developers easier and comprehensive.

**What is POM?**

POM is an xml file, that contains information about project and configuration details used by Maven to build the project, hence it is fundamental use of POM.

**POM – Project Object Model (**fundamental unit of work in Maven**)**

- Describes the project
- Name and Version, Artifact type, Source Code location, Dependencies
- Plugins
- Profiles (Alternate Build Configurations)
- Uses XML by default
- Not the way ANT uses XML

**Maven Objectives:**

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Providing guidelines for best development practices
- Allowing transparent migration to new features

# Maven Tutorial for Beginners 2 - How to Install and Setup Maven

https://www.youtube.com/watch?v=3ODSQ0EpoQI&index=2&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ

Installing the Java JDK Software and Setting JAVA_HOME on Windows
- https://www.youtube.com/watch?v=Hjs6L...

All commands are listed in this post - http://www.codebind.com/maven/install...

Maven – Download Apache Maven
What is Maven and how do I install it?
How to install apache maven in a windows system
How to Install Maven on Windows for Java
How to Setup/Install Maven Classpath Variable on Windows
How to Install Maven on Windows 8 / Windows 10

**Install Maven on windows:**

Make sure JDK is installed and **JAVA_HOME** variable is added to windows environment variables

Download maven from web and extract the zip and place in desired location.

Add both **M2_HOME** & **MAVEN_HOME** variables in the windows environment variables point it to your **Maven/bin folder**

Once you set the environment variables you can test cmd> mvn -version –> you see the details of maven version, java version, java home path etc.

# Maven Tutorial for Beginners 3 - Creating First Maven Project

https://www.youtube.com/watch?v=MHrUzW-4XzI&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ&index=3

Archetypes, GroupId, ArtifactId, Version, pom.xml
Maven Tutorial for Beginners
Searches related to First Maven Project
maven model version
maven project in eclipse
maven project example
maven project structure
import maven project into eclipse
creating maven project in eclipse
sample maven project

**Steps to create First Maven project:**

1. Create a folder for first java maven project “firstmavenproject” and move to that folder through cmd.
2. **Cmd> mvn archetype:generate** -> generate cmd create a project from an existed template & makes sure you are connected to internet so that it can download some plugins from internet. First it will search in your local machine ${homeuser}/.m2/repository ex: c:\users\anil\.m2\repository if not found it will download from internet and keep in your machine. During this time, you will notice some numbers and some text will display, those numbers are archetype(these are templates) and description of archetype. Need to choose these relevant numbers to create a project. Press enter to select a project, it will give list of versions, you can select latest version (in this case 6, version is 1.1)
3. Every project having a groupId, artifact Id, version. Give groupId – unique in the organization or a project. Give the website(codebind.com) of the organization in reverse order like “com.codebind”, this is your unique id, press enter.
4. Give **artifactId**: , it is generally the project name, here we can give “my-maven-project”
5. Give **version**: 1.0-SNAPSHOT: 1.0-SNAPSHOT (SNAPSHOT is keyword, that indicate development project). Whenever you generate .jar or .war file you will see the file name as **artifactid.version_name.jar** (here the name is “my-maven-project.1.0-SNAPSHOT.jar”).
6. Give **package**: com.codebind.demo -> code will generate inside **com.codebind.demo** folder. In this folder it will create class directory, test directory.
7. It will ask for the confirmation, say “Y” and press enter so that it will create folder structure for you.
8. Go to the “firstmavenproject” folder > my-maven-project -> inside your see “src” folder(inside you see src and test fodders) and “pom.xml”. inside the src -> main > java > com > codebind >demo >App.java (sample java file). Inside the src>test -> java > com > codebind >demo > AppTest.java (sample java test file).

# Maven Tutorial for Beginners 4 - Creating Maven project using Eclipse IDE + Understanding pom.xml

https://www.youtube.com/watch?v=TpPEgCm65CE&index=4&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ

How to install Eclipse on Windows 8 / Windows 8.1 / Windows 10 - https://www.youtube.com/watch?v=35NUu...

### Creating a project in Eclipse:

1) Open eclipse and click on File > New > Project (do not choose Java Project) > Open wizard > select Maven > Maven project > next > check “create a simple project” & default workspace location uncheck third check box > next
2) New Maven project window > Group Id: com.codebind > Artifact Id: Maven-demo > version: 0.0.1-SNAPSHOT > Packaging: jar > Finish
3) Maven-demo project created in Eclipse. Inside the maven project 4 packages
    1) src/main/java -> source files
    2) src/main/resources -> resources like images, xml for java project
    3) src/test/java -> test env
    4) src/test/resources -> resources for test
4) Folder structure src > main & test, target and pom.xml.
5) Double click on pom.xml, see the overview window. To see the pom.xml in xml format click on the pom.xml in the bottom last tab.
6) Opens maven-demo/pom.xml window it has modelVersion, groupId, artifactId, version.
7) Right click on /src/main/java package > new > class > package: com.codebind > Name: App > Finish

8) Now it will create Ap.java project with default structure. Write something like this

```
package com.codebind;
public class App {
    public String Sample {
        return “Sample”;
    }
}
```

9) To create a test for this go to src/test/java > right click > new > other > open wizard > Junit > junit test case > Next > package: com.codebind > Name : app-test > OK
10) You will see some error in the code because junit jar file in our dependency/build paths.
11) To remove these errors go to pom.xml, add dependencies. To do this search for “maven junit dependency”. Click on mvnrepository.com site. Click on this link, top one is latest click on it. Copy the dependency and paste this in pom.xml & save it(see below code), then you notice “maven dependencies” downloaded, inside you see junit-4.12

```
<dependencies>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
</dependencies>
```

12) Now you go back AppTest.java file you will not see those errors.

# Maven Tutorial for Beginners 5 - How to create a jar file with Maven

https://www.youtube.com/watch?v=vGtGxKZQ-I8&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ&index=5

### How to create jar file with Maven?

Step 1: first compile all source .java file to convert to .class files
Step 2: Compile all test files to convert all test java files into .class files.
Step 3: Run the tests and ensure all are successful.
Step 4: Create a jar file.

### Go to eclipse ide

1. Open the project “maven-demo” > right click and select properties and copy the location path from here and cancel. Now go to explorer and paste the path then open it. Here you see src, target and pom.xml
2. **To run the “clean”** command(to clean the project) -> select “maven-demo” and right click select “run as” select “maven clean” command > **goals: clean** > run
3. After successful run, you go workspace/maven-demo folder you will notice “target” folder is deleted. Means when you issue clean command it will clean up the build files(binaries, jar files etc) which are stored in the target folder.
4. **To “compile” our source code**: ) -> select “maven-demo” and right click select “run as” select “maven build..” command > **goals: compile** (it will compile source files and create .class files) > run

5. After successful run, you go workspace/maven-demo folder you will notice “target” folder you see the classes folder > com > codebind > App.class file created by maven for you.
6. **To compile the our test code**: ) -> select “maven-demo” and right click select “run as” select “maven build..” command > **goals: test-compile** (it will compile source files and create .class files) > run
7. After successful run, you go workspace/maven-demo folder you will notice “target” folder you see the test-classes folder > com > codebind > AppTest.class file created by maven for you.
8. **To test the our test code**: ) -> select “maven-demo” and right click select “run as” select “maven build..” command > **goals: test** (it will compile source files and create .class files) > run
9. After successful run, you go workspace/maven-demo folder you will notice “target” folder you see the surefire-reports folder > com.codebind.AppTest.txt file created by maven for you.
10. With this our source files compiled, test files compiled, our tests are run succesfullly. Now to create jar file, direcly you can run “Maven Install” to create jar file. But we are following “Maven Build…” > Goals: install > run
11. After successful run, you go workspace/maven-demo folder you will notice “target” folder you will notice **meven-demo-0.0.1-SNAPSHOT.jar** file created.

### Now see the same steps from command line:

1. From the command prompt, change the directory to the maven project.
2. From cmd > mvn clean -> it cleans the target folder contents
3. Cmd> mvn compile -> to compile our source files
4. Cmd> mvn test-compile -> to compile our test files
5. Cmd> mvn test -> to run the tests
6. Cmd> mvn install -> to create .jar file.

## Maven Tutorial for Beginners 6 - Introduction to the Build Lifecycle

https://www.youtube.com/watch?v=NwrdhG4nTgg&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ&index=6

**Maven build life cycle steps / commands**

1. Validate
2. Compile
3. Test
4. Package
5. Integration-test
6. Verify
7. Install
8. Deploy

All the above command follow the hierarchy while executing. For example, if you issue **compile** command it will do validate > compile. Similarly, if you issue **verify** command it will run first validate > compile > test > package > integration-test > verify.

**Clean command is not part of the build life cycle.**

# Maven Tutorial for Beginners 7 - Transitive dependencies in Maven

https://www.youtube.com/watch?v=ypVE8EgDzzI&index=7&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ

- when “junit” dependency is added to pom.xml, it will add “junit-4.12.jar” and “hamcrest-core-1.3.jar” files under “Maven Dependencies”.

### Why this “hamcrest-core-1.3.jar” was added?

Junit dependent on “hamcrest-core” dependency hence it is added. You can click on “Dependency Hierarchy” tab (3rd) at the bottom of the screen. Here you can see the junit under that hamcrest. This dependency of dependency is called **Transitive dependencies**.

Click on pom.xml at the bottom of the screen, it opens pom.xml and click on Ctrl + mouse over the junit and click on it. It will open will open pom.xml file of junit, then click on pom.xml(at bottom) there you can see the dependency list of junit. Here you can see groupId: org.hamcrest, artifactId: hamcreast-core, version: 1.3. this is called transitive dependency.

Go back to our project pom.xml and try adding hibernate jar file. Search in the internet for “hibernate maven dependency”. Click on core hibernate functionality and click on the top one and select dependency and copy. Go to pom.xml in eclipse and paste it under junit dependency. Save it, now you see “hibernate-core-5.1.0-final.jar” file is added and also you see some more .jar files added. All these .jar files are the dependencies hibernate-core jar file. Similarly you can see all the dependency details by ctrl+click on the hibernate jar.

Dependency of dependency is call transitive dependencies.

# Maven Tutorial for Beginners 8 - Excluding Maven Dependencies

https://www.youtube.com/watch?v=olGJsY2zNvg&index=8&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ&spfreload=1

in the last session we added “hibernate” jar file. We have seen some dependencies(transitive) are also added. One of the transitive dependency is jboss-logging jar file. Some other reason we are not interested for jboss-logging jar file, we want to exclude it and add some other logging file.

To exclude the jar file, you need to know the groupid, artifactid of it. Place the cursor on hibernate and press ctrl + click on the link, it will open the pom.xml, go to the pom.xml and select the groupid, artifactid of jboss dependency elements and copy. Now go to your project pom.xml and below to the version of hibernate dependency add

<exclusions>
<exclusion>
**PASTE the jboss groupid and artifactid.**
</exclusion>
</exclusions>
Save the pom.xml, now you see the jboss jar file disappears.

## Maven Tutorial for Beginners 9 - scope Dependencies in Maven

https://www.youtube.com/watch?v=OQvAeDGFYGE&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ&index=9

### How to limit the dependencies?

For example for junit, we want to limit the scope only to “test” (not for compile, install, runtime or any other) for this we need to add <scope>test</scope> under the <version> tag in junit dependency in pom.xml. That means junit dependency is only available for “test” and not available for any other actions.

**Compile** is the default scope.

**Provided** scope is much like compile, but it indicates that you expect the jdk or container to provide the dependency at runtime. For example when building web application java EE you would set the dependency on servlet api and the related java ee api to scope **provided** because the web container provide those classes. So this scope is only available on compilation and test class path but its not available in transitive. These scopes are like **runtime**, other scopes are important when you want to create an api which has an interface and the implementation. So basically, when a developer develops an application he wants to segregate his api into different segments. **One is the interface, and other is the implementation.** This is done because the implementation if the developer want to change at any time he can change it and he do not require to change the interface. And vice versa also possible easily.

## Cmder: A Better Command Prompt Tool for Windows

https://www.youtube.com/watch?v=_X9UPThjJvg&list=PLS1QulWo1RIaaQ3mAU9Nj4rqfwbAv3wIZ&index=11

I am a fan of Cmder, a package including clink, conemu, ssh, msysgit, and some other cool enhancements.
**Cmder** is a software package created out of pure frustration over absence of usable console emulator on Windows.

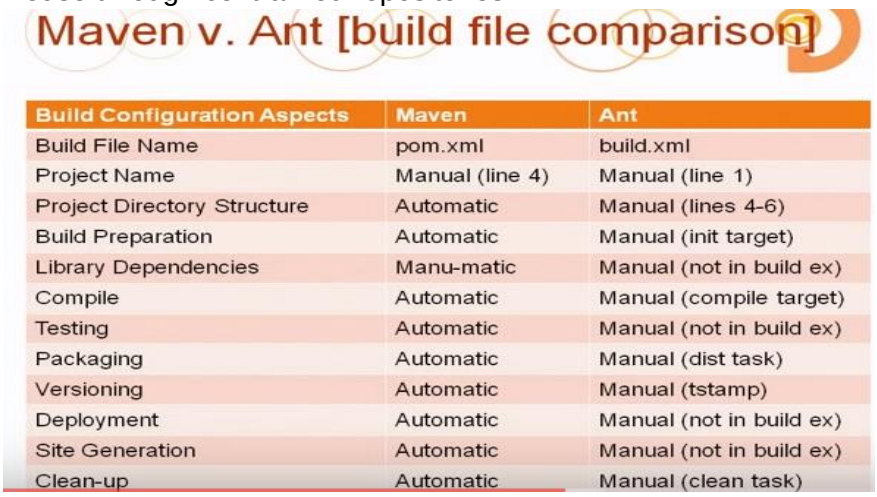
Linux command also you can execute in cmder tool.

# Apache Maven Fundamentals (part 1 of 4)

https://www.youtube.com/watch?v=VOkgaq__Ff8

**Tenets of Maven**

1) Project oriented
2) Convention over configuration
3) Dependency management
4) Extensible through plugins
5) Reuse through centralized repositories.

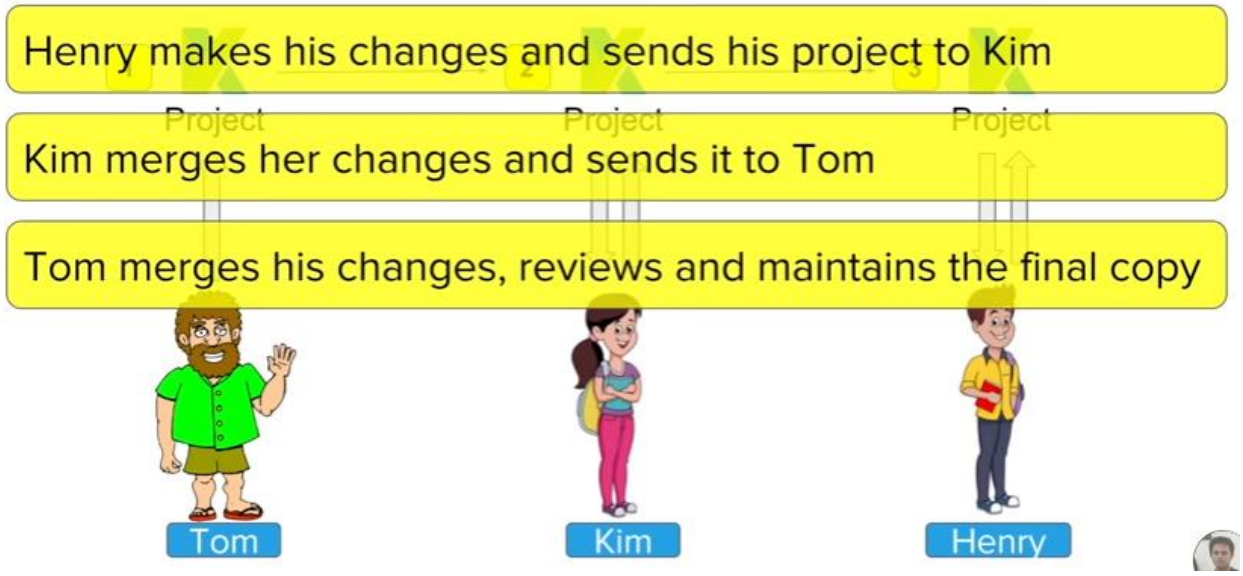


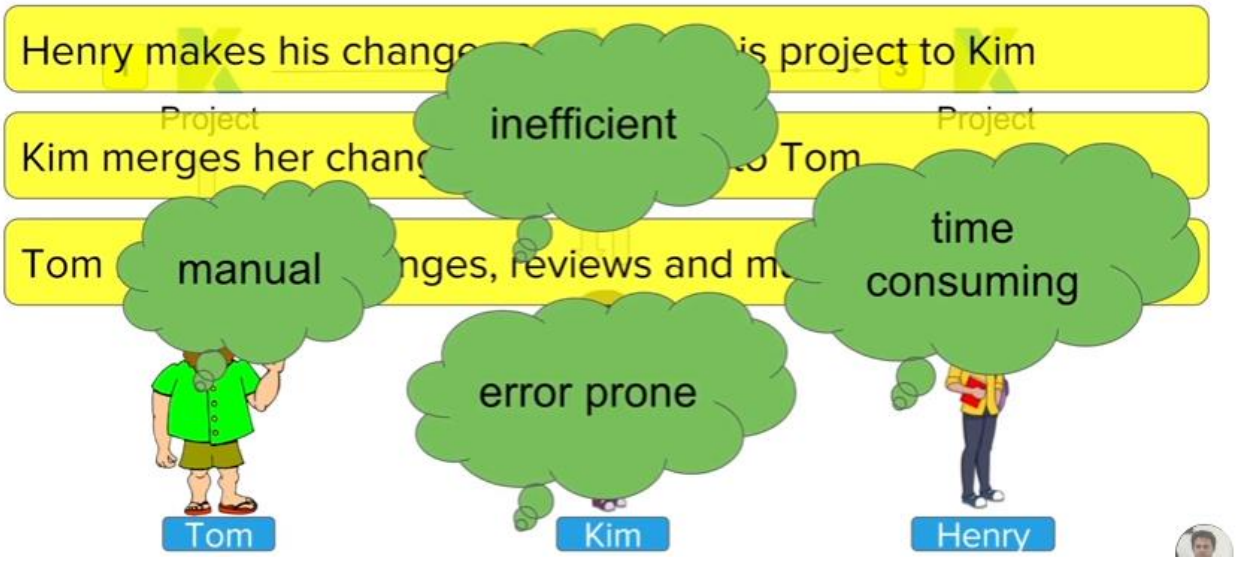

| Build Configuration Aspects | Maven | Ant |
|---|---|---|
| Build File Name | pom.xml | build.xml |
| Project Name | Manual (line 4) | Manual (line 1) |
| Project Directory Structure | Automatic | Manual (lines 4-6) |
| Build Preparation | Automatic | Manual (init target) |
| Library Dependencies | Manu-matic | Manual (not in build ex) |
| Compile | Automatic | Manual (compile target) |
| Testing | Automatic | Manual (not in build ex) |
| Packaging | Automatic | Manual (dist task) |
| Versioning | Automatic | Manual (tstamp) |
| Deployment | Automatic | Manual (not in build ex) |
| Site Generation | Automatic | Manual (not in build ex) |
| Clean-up | Automatic | Manual (clean task) |

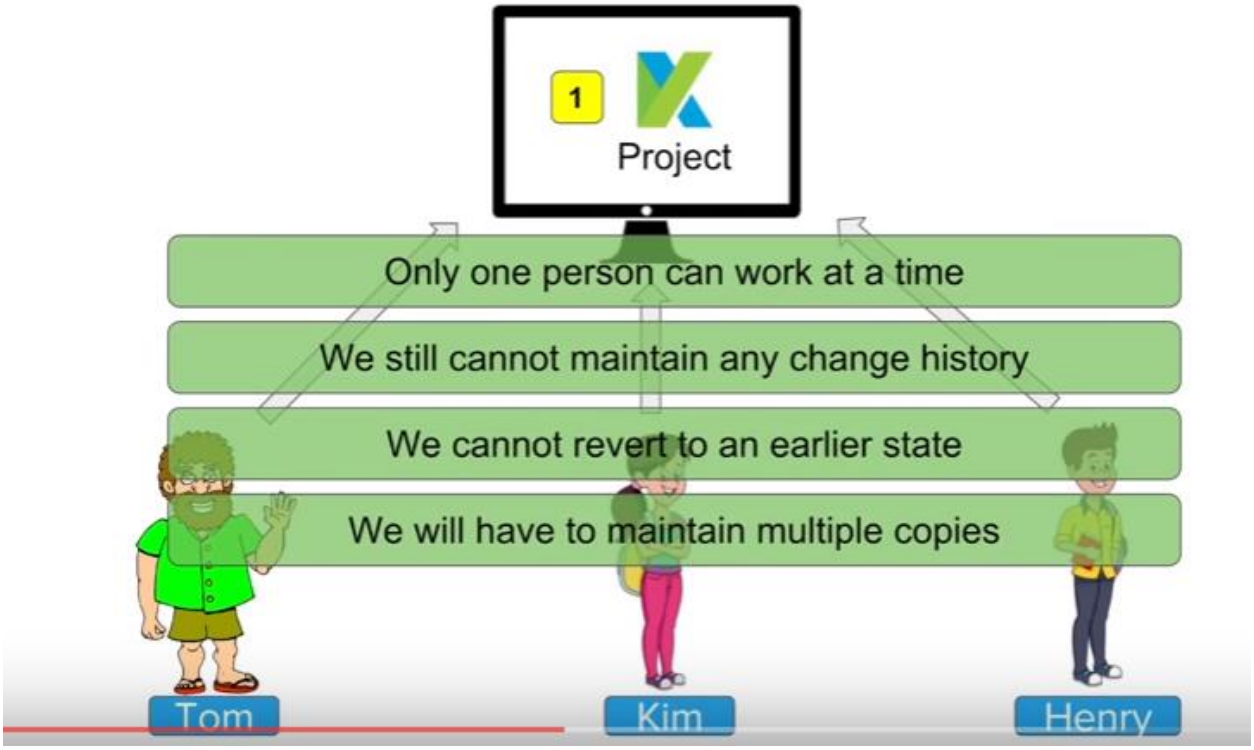
# GIT

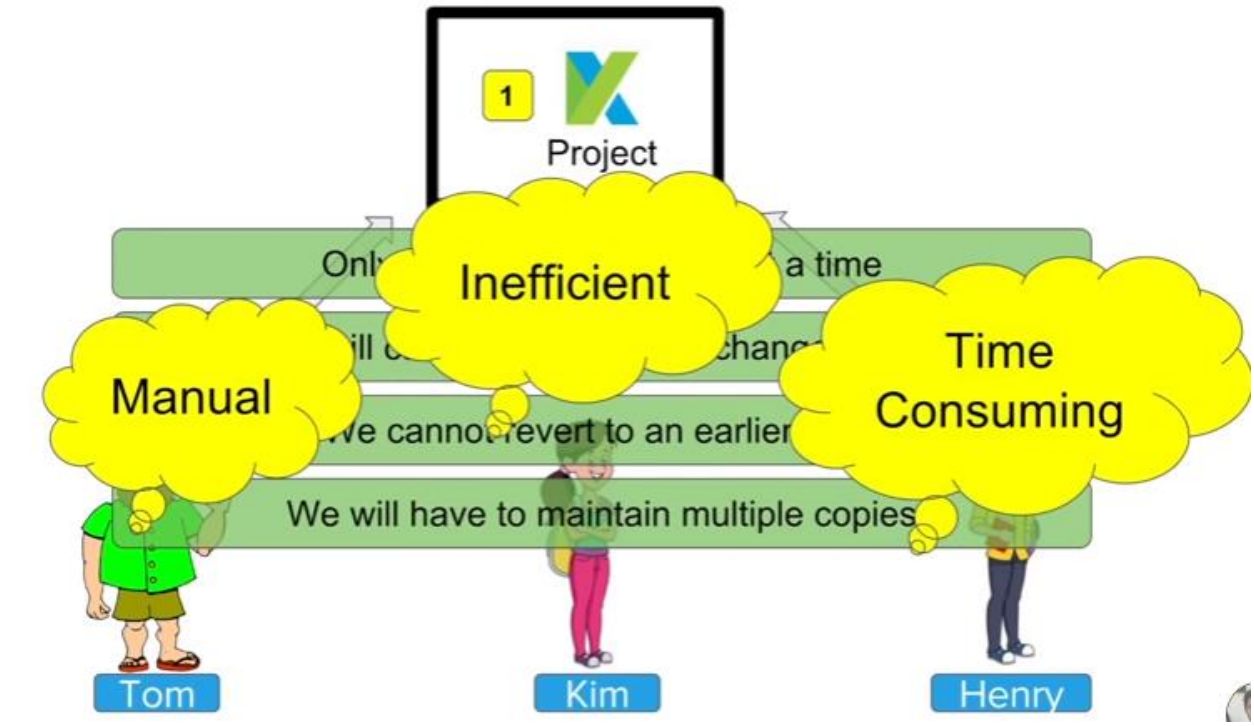
## Why to use GIT (version control system)

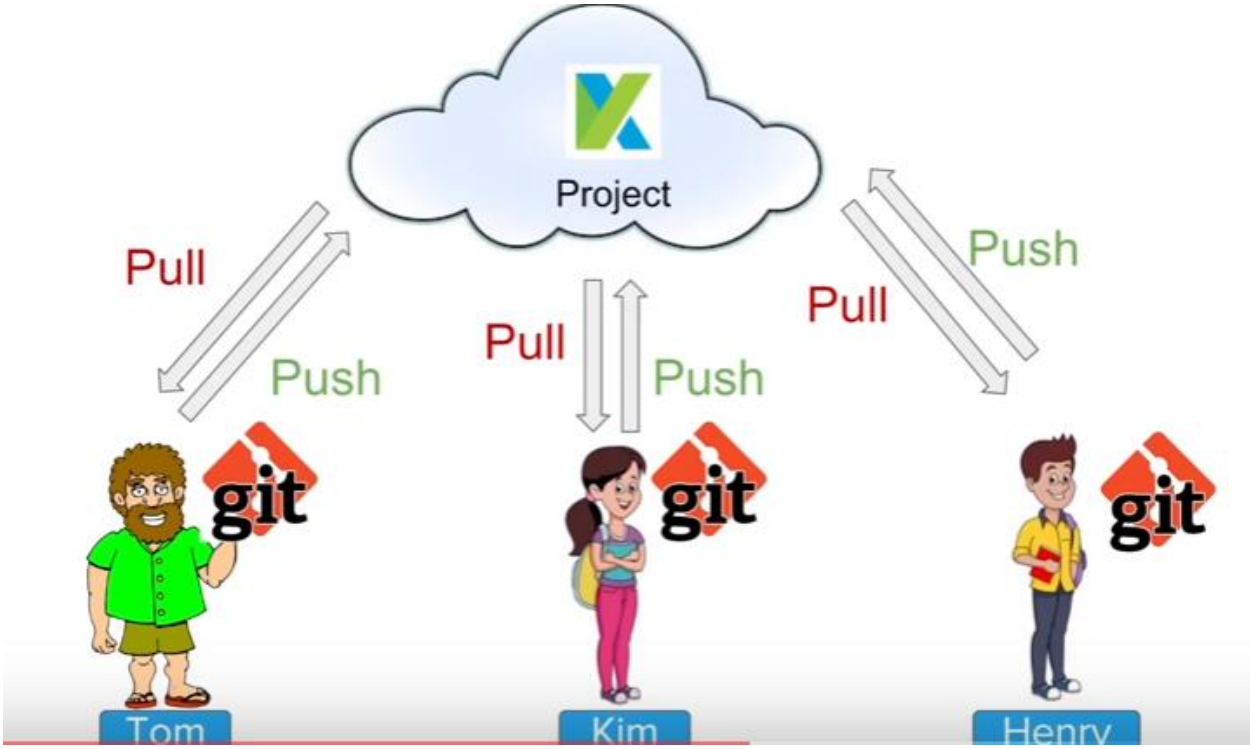
https://www.youtube.com/watch?v=DqtZUvmPmo4&list=PLhW3qG5bs-L8OlICbNX9u4MZ3rAt5c5GG




1
Project
Only one person can work at a time
We still cannot maintain any change history
We cannot revert to an earlier state
We will have to maintain multiple copies
Tom
Kim
Henry


1
Project
Inefficient
Manual
Time Consuming
We will have to maintain multiple copies
Tom
Kim
Henry



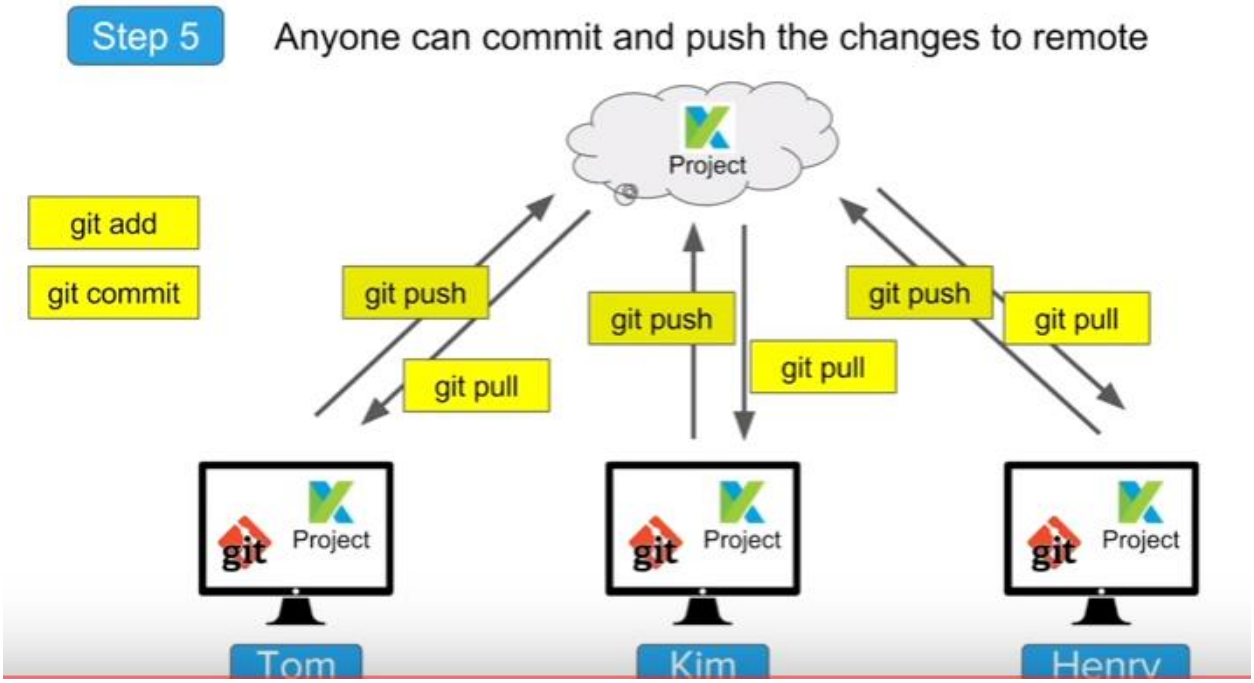

**Step 1** All team members will install git on their systems

**Step 2** Anyone can put a copy of project on remote repository
(like GitHub or BitBucket)

**Step 3** Everyone will clone the project from remote to their local

**Step 4** Everyone can now work on their local copies

**Step 5** Anyone can commit and push the changes to remote




# Git and GitHub Beginner Tutorial 1 – Introduction

https://www.youtube.com/watch?v=-U-eUHI6euM&index=2&list=PLhW3qG5bs-L8OIICbNX9u4MZ3rAt5c5GG

Git and GitHub 1 - Introduction
==========

Today we will learn
_____

1. What is GIT
2. What is GIT HUB
3. Is GIT related to GIT HUB
4. A simple workflow

GIT - VCS - version control system
- to track changes in files / folders - to collaborate in teams
- free and open source

Centralised VCS | Distributed VCS

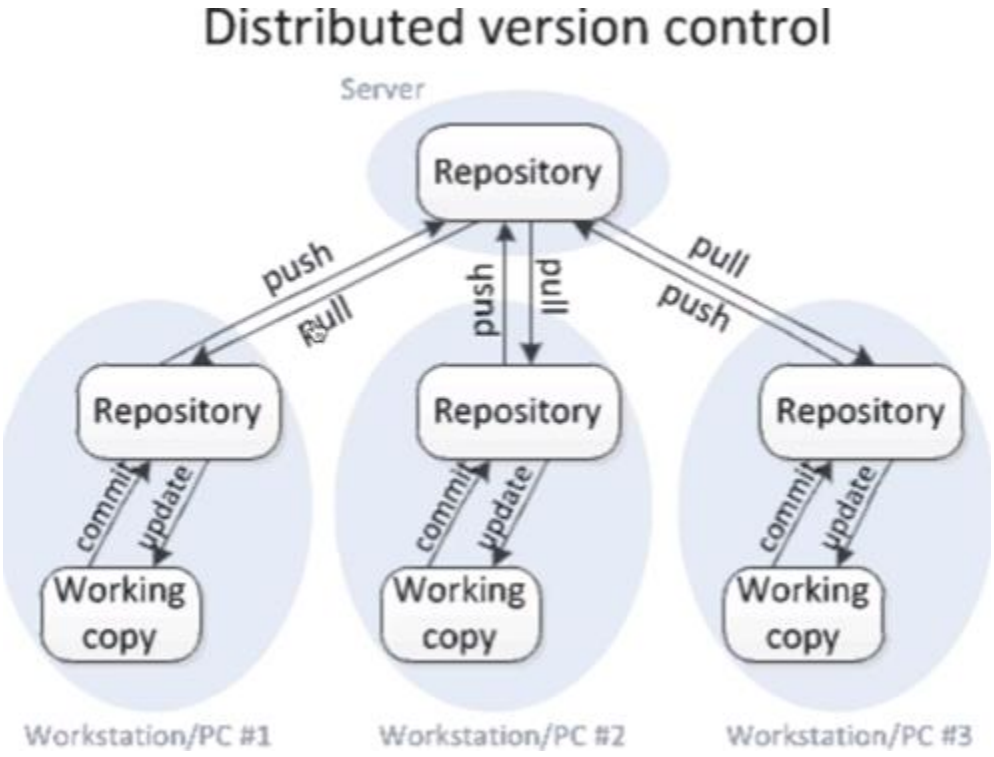
GIT = DVCS

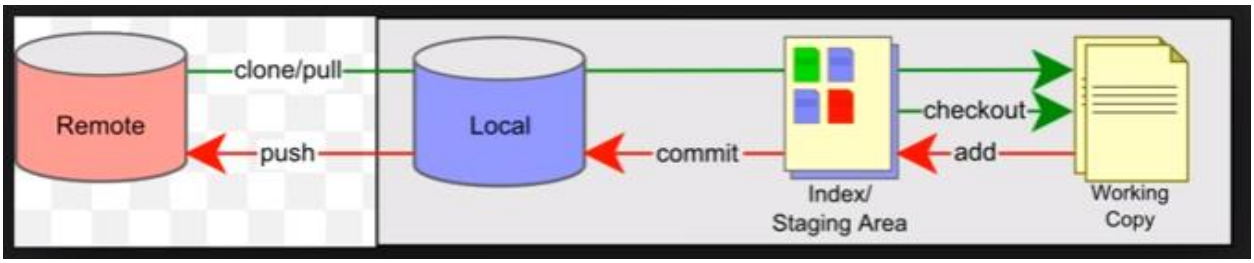
_____

GIT HUB - website to upload your repositories online
- provides backup

- provides visual interface to your repo
- makes collaboration easier

GIT != GIT HUB




## Git and GitHub Beginner Tutorial 2 - Getting started - Install Git (mac)

https://www.youtube.com/watch?v=0Icla6TVNNo&index=3&list=PLhW3qG5bs-L8OIICbNX9u4MZ3rAt5c5GG

Git and GitHub Beginner Tutorial 2 - Getting started - Install Git (mac)
==========
Today we will learn
---

1. How to download and install git
2. Signup and create account on GitHub
3. Add a project/folder/file to git
4. Track and commit changes
5. Add the repository to GitHub

_____

**Step 1 : Check if git is already installed.**
terminal - git --version

**Step 2 : Download and install git**
https://git-scm.com/download/mac

**Step 3 : Signup and create an account on GitHub**
https://github.com/

**Step 4 : Add your github email and username to git**
git config --global user.email "yourGitHub@email.com"
git config --global user.name "yourGitHubusername"

**Step 5 : Add file/folders to git - tracking**

**Step 6 : Commands**
- terminal - goto the location of the folder/project

- git init
- git status
- git add
- git commit -m "…."
- git remote add origin "location of remote repo"
- git push -u origin master
- git log
- git –help

## Git and GitHub Beginner Tutorial 3 - Getting started - Install Git windows

https://www.youtube.com/watch?v=sBTAkHOxvOk&list=PLhW3qG5bs-L8OlICbNX9u4MZ3rAt5c5GG&index=4

Git and GitHub Beginner Tutorial 3 - Getting started - Install Git (windows)
==========

Today we will learn
_____

1. How to install Git on windows
2. Adding project/files to git for tracking
3. Git commands
4. Pushing project to remote repository(github)

_____

Step 1 : check if git is already installed git --version

Step 2 : download and install git

Step 3 : add your project to git

Step 4 : commands git config --global user.email "yourGitHub@email.com"
git config --global user.name "yourGitHubusername"

- git init
- git status
- git add
- git commit -m “…..”
- git remote add origin https://github.com/RaghavAutomation/R...
- git push -u origin master
- git log
- git —help

Step 5 : adding project to remote repository (github)

## Git and GitHub Beginner Tutorial 4 - Enable git commands autocomplete and colors on Mac

https://www.youtube.com/watch?v=VI07ouVS5FE&index=5&list=PLhW3qG5bs-L8OlICbNX9u4MZ3rAt5c5GG

Git and GitHub Beginner Tutorial 4 - Enable git commands autocomplete and colors on Mac
==========

Today we will learn
_____

1. How to enable git commands autocomplete feature on mac os
2. How to enable git commands color features on mac os

_____

For autocomplete
_____

Step 1 : goto terminal put git-completion.bash script in your home directory
curl https://raw.githubusercontent.com/git... -o ~/.git-completion.bash

Step 2 : Add following line to your .bash_profile. This tells .bash to run git auto complete script if it exists

vi ~/.bash_profile

```
if [ -f ~/.git-completion.bash ]; then
. ~/.git-completion.bash
fi
```

For enabling git colors
_____

Step 1 : check if colouring is already enabled
terminal - git config color.ui

Step 2 : enable colouring
git config --global color.ui true

## Git and GitHub Beginner Tutorial 5 - Branching and Merging

https://www.youtube.com/watch?v=GZILYABgAoo&list=PLhW3qG5bs-L8OlICbNX9u4MZ3rAt5c5GG&index=6

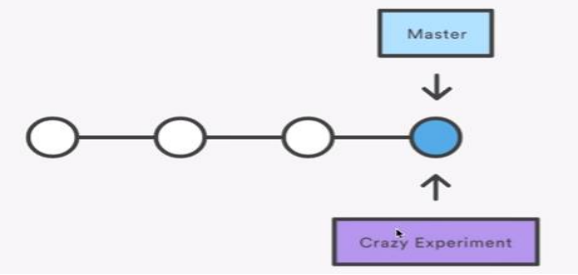
Git and GitHub Beginner Tutorial 5 - Branching and Merging
==========

Today we will learn
_____

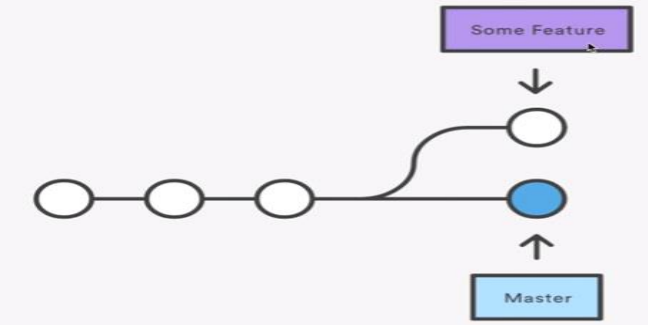
1. What are branches
2. How to create branch
3. How to checkout branch
4. How to merge branch to master
5. How to delete branch (local and remote)

_____

**Step 1 : Create branch.**
Go to the folder where you want to create a branch and run the following command.
git branch <branch name>
$ git branch MyNewBranch
$ git status




**Step 2 : Checkout branch**
git checkout <branch name>
$ git checkout MyNewBranch




$ touch test2.txt
$ git status

$ git add .
$ git commit -m “added test2.txt”
$ git status

$ git push -u origin MyNewBranch
$ git checkout master

**Step 3 : Merge new branch in master branch**
git merge “branch name”
$ git merge MyNewBranch #when you are merging to master you first checkout master and then merge
$ git push -u origin master #after this mynewbranch and master are equal

**Step 4 : Delete branch**
git branch -d <branch name> — delete from local
git push origin --delete <branch name> — delete from remote

## Git and GitHub Beginner Tutorial 6 - How to send email from GitHub

https://www.youtube.com/watch?v=Ft2LXlaSEfs&list=PLhW3qG5bs-L8OlICbNX9u4MZ3rAt5c5GG&index=7

Git and GitHub Beginner Tutorial 6 - How to send email from GitHub
==========

Today we will learn
_____

How to trigger notification email from github
whenever there is any change/commit in the project
_____

**Step 1 :** Go to Github account -> select Repository -> click on Settings -> click integration & services -> add Service > type email > address: give your email id > check “Send from author” > Add Service > now this service is added.

**Step 2 :** Test and validate by making some change in the project

$cd <project folder>

$ git status

$ touch index2.html

$git status

$ git add .

$ git commit -m “added index2.html”

$ git push -u origin master

$

Check the email which you have configured.

## Git and GitHub Beginner Tutorial 7 - Git Tags - what, why, when and how

https://www.youtube.com/watch?v=govmXpDGLpo&index=8&list=PLhW3qG5bs-L8OlICbNX9u4MZ3rAt5c5GG

Today We will learn:
---
1. What are tags / releases
2. Why should i create TAGs
3. When to create TAGs
4. How to create TAGs in git

create | show | publish | delete

**What are Git TAGs**

Tagging in Git or any other VCS refers to creating specific points in history for your repository/data.

This is usually done to mark release points. For example, your project is stable you want to mark it ias release 1.0 you can create a tag for it.

**Why should I create TAGs**

1. To mark release point for your code/data
2. To create historic restore points

**When to create TAGs**

1. When you want to create a release point for a stable version of your code
2. When you want to create a historic point for your code/data that you can refer at any future time (to restore your data)

**How to create TAGs in Git**

5 easy steps.

**Step 1: Checkout the branch where you want to create the tag**
git checkout "branch name"
example : git checkout master

**Step 2: Create tag with some name**
git tag "tag name"
$ git tag v1.0

$ git tag

$ git tag -a v1.1 -m "tag for release ver 1.1" (to create annotated tags) – you can give notes available

$ git tag

V1.0

V1.1

$

**Step 3: Display or Show tags**
$ git tag

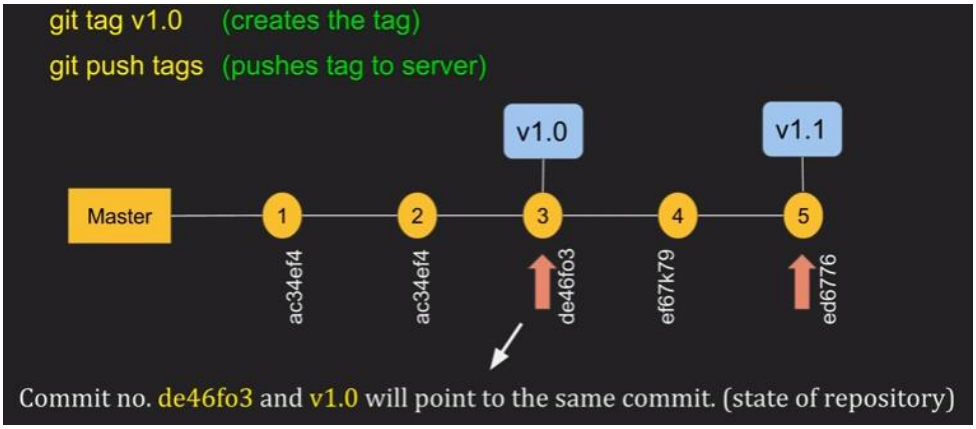
$ git show v1.0
$ git tag -l “v1.*”

**Step 4: Push tags to remote.  Go to github see the release**
$ git push origin v1.0 # Go to github see the release, you see 1 releases
$ git push origin --tags
$ git push --tags  (to push all tags at once)

**Step 5: Delete tags (if required only)**
to delete tags from local :
git tag -d v1.0
git tag --delete v1.0

to delete tags from remote :
git push origin -d v1.0
git push origin --delete v1.0
git push origin :v1.0

to delete multiple tags at once:
git tag -d v1.0 v1.1 (local)
git push origin -d v1.0 v1.1 (remote)




**Checking out TAGS**

We cannot checkout tags in git
We can create a branch from a tag and checkout the branch
git checkout -b "branch name" "tag name"
$ git checkout -b ReleaseVer1 v1.0

---

Can I create a tag from some past commits?

YES

Creating TAGS from past commits

git tag <tag name> <reference of commit>
$ git tag v1.2 5fcdb03

$ git tag

$ git push –tags # push to remote repo

$

# Docker Beginner Tutorial 1 - What is DOCKER (step by step) | Docker Introduction | Docker basics

https://www.youtube.com/watch?v=wi-MGFhrad0&list=PLhW3qG5bs-L99pQsZ74f-LC-tOEsBp2rK

A code works in dev system, and the same code may not work in test system. Docker at very basic level resolve these issues of an application running on one platform and not working on some other platform.

**Where does docker operate?**

Design -> Development -> Deployment -> Test/Release

Docker comes into the picture when **Deployment phase** come into picture. However, present in entire workflow, but main use is in Deployment.

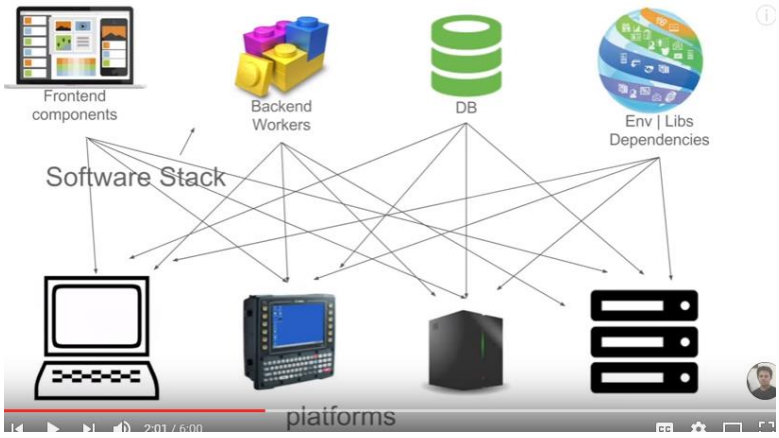
Docker makes the process of application deployment very easy and efficient and **resolves lot of issues related to deploying applications.**

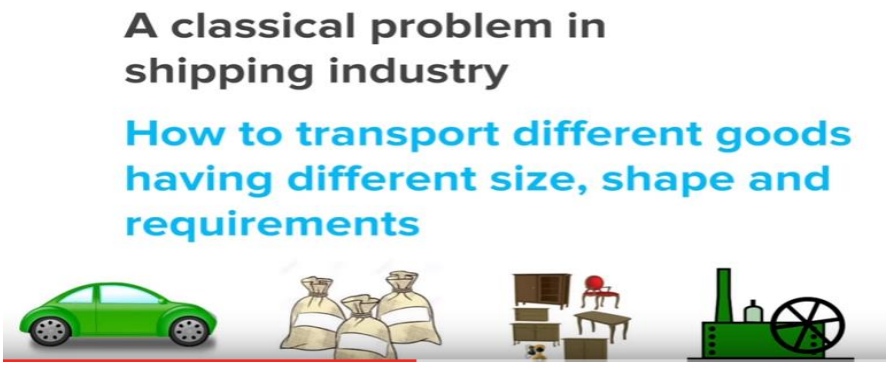
**What is Docker?**

Docker is the word's leading software container platform.

***Docker is a tool designed to make it easier to deploy and run applications by using containers.***

***Containers allow a developer to package up an application with all the parts it needs such as libraries and other dependencies, and ship it all out as one package.***



- ✓ Docker is the world's leading software container platform
- ✓ Docker makes the process of application deployment very easy and efficient and resolves a lot of issues related to deploying applications
- ✓ Docker is a tool designed to make it easier to deploy and run applications by using containers
- ✓ Docker gives you a standard way of packaging your application with all its dependencies in a container

---

Watch Docker Playlist:
https://www.youtube.com/playlist?list...

Like on Facebook :
https://www.facebook.com/automationst...

Follow on Twitter:
https://twitter.com/automationsbs

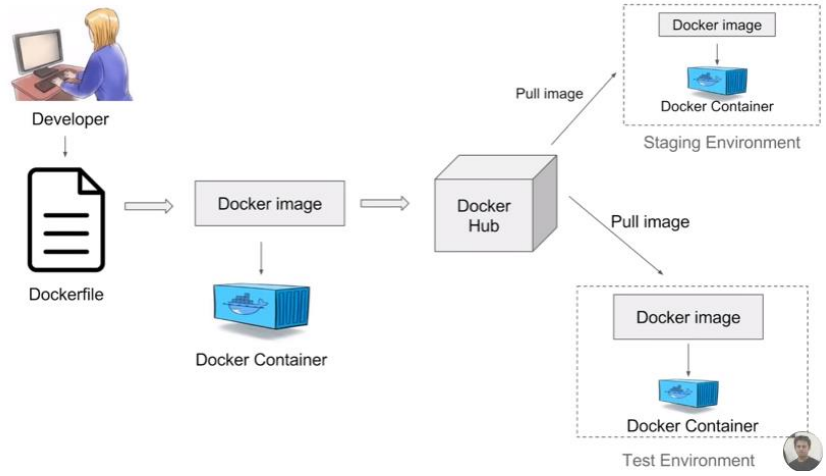
Subscribe on Youtube :
http://youtube.com/automationstepbystep

---

# Docker Beginner Tutorial 2 - How DOCKER works?

https://www.youtube.com/watch?v=0e-KiGJIiDc&index=2&list=PLhW3qG5bs-L99pQsZ74f-LC-tOEsBp2rK

**How Docker works?**




In a docker work flow a developer will define all the applications and dependencies and requirements in a file called as **Dockerfile**. Now this docker file can be used to create docker images. So in a docker image you will have all the applications, its requirements and dependencies. When you run a docker image you will get docker containers. So docker containers are the run time instances of the docker image. These images can also be stored in a **online cloud repositories called Docker Hub**. So if you go to https://hub.docker.com/ you will find lot of publicly available images and you can store your own docker image as well. You can also store docker images in your own repository or version control system. Now these images can be pulled to create in any environment so you can create a docker container in a test environment or any other environment. And we can be show them our application will

run in the same way using docker containers. This resolves the issue of app working on one platform and not on other.

**Docker is a container platform.**

**Dockerfile** – describes steps to create a docker image. It's like a recipe with all ingredients and steps necessary in making your dish.

Container will have application with all its dependencies.

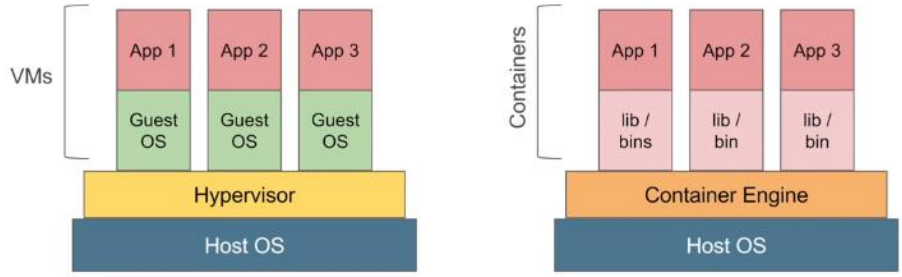
### Virtualization vs Containerization

In the concept of virtualization, we have a software called **Hypervisor which is used to create and run virtual machines** and using hypervisor we can create multiple virtual machines on a host OS. Now these virtual machines have their own operating system and it does NOT use the host OS so there can be an overhead on the host platform. Also, in case of virtual machines we should allocate a fixed memory and space to every machine so there is a lot of wastage of memory and space.

Let us come to containerization, here we have container engine and we do not have separate OS, but we have containers where we have the application and all its dependencies and it will use the host OS. Now here the space, memory and other resources are not fixed, it will be taken as per the needs of the application. So, there is no overhead it is very light weight and very fast. Now there can be scenarios where we need VM over a host OS and then have containers. For example if we want to run a windows application on a Linux OS, we need to have a VM first which will have a windows OS and then we can have containers over it. So now in case of docker the container engine is a docker engine.

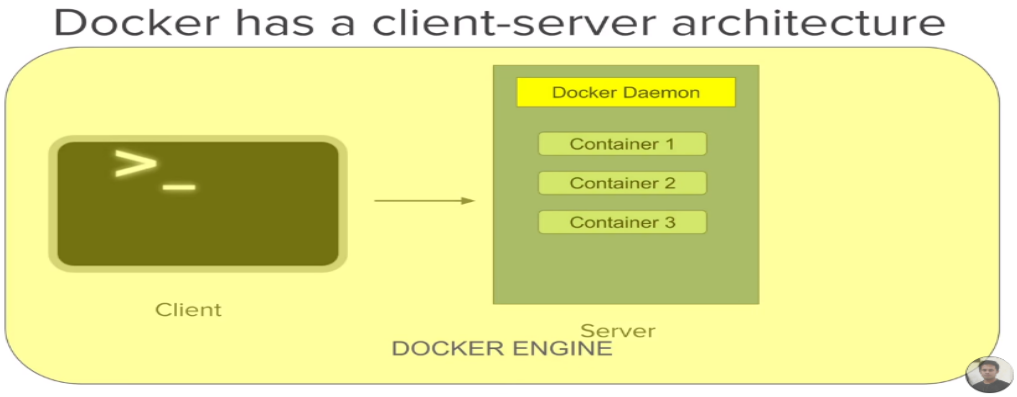
VMs – resource allocation is fixed and does not change as per application needs.

### Containers are light weight alternative to VMs




Virtualization Containerization

Docker is a client server architecture. In **Docker command line interface is the client** and we have docker server or Docker daemon which will have all the containers and that docker server receives commands from the docker client in the form of the commands or REST API requests. All the components of the **docker client and server together form docker engine**. So the docker daemon(server) receives the commands from a Docker client through CLI or REST APIs. And Docker client and daemon can be present on the same host (machine) or different hosts.




- Developer create a Dockerfile(define all the applications and dependencies and requirements)
- Dockerfile creates Docker Image (will have all the applications, its requirements and dependencies, it stores in Docker Hub)
- Run the Docker Image to get the Docker Containers(run time instances of the Docker image.)

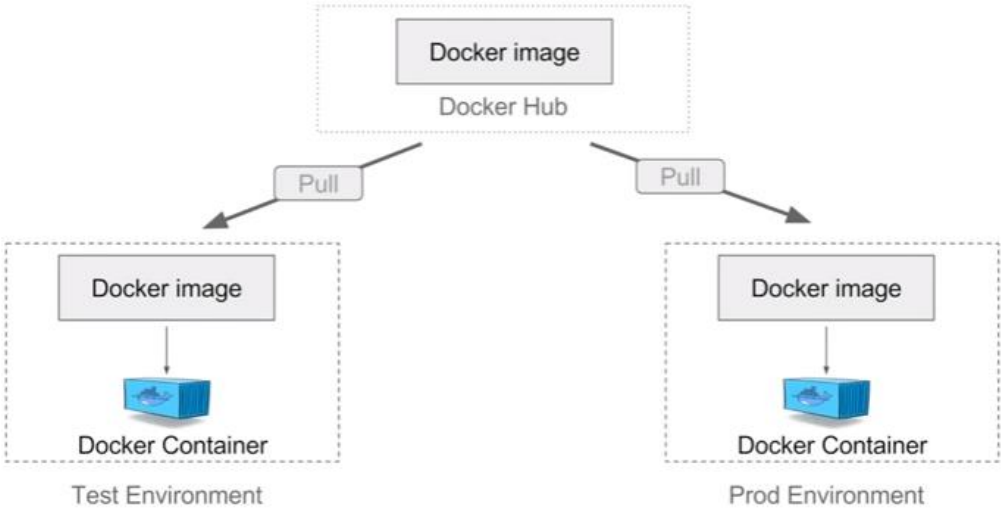
## Docker Beginner Tutorial 3 - Benefits of DOCKER

https://www.youtube.com/watch?v=YCrRy7pBzdc&index=3&list=PLhW3qG5bs-L99pQsZ74f-LC-tOEsBp2rK

**Advantages of Docker:**

It ***resolves the problem of code working on one system and not working on some other system.***

So, with docker you can build your applications only once, and then no need to build or configure it multiple time on different environments or platforms. Means an application inside a container can run on any system that has Docker installed. So, there is no need to build and configure application multiple times on different platforms.




Also, we have seen that you can create a Docker image on a Docker hub or any other repository and then we can pull the Docker image on any environment and run the image to create a Docker container

and our application will run inside this container. So, we can do the same way any other environment so we can be sure that whatever testing we have done in test environment our application will work as it is in the production environment because now these two environments can be considered as identical because they are running the same Docker container.

Also, **portability** is great feature with Docker, suppose you created container using amazon EC2 machine now you can very easily port this container to VirtualBox and it will run as it is, so you do not have to worry about the **portability** in case of Docker. Also, you can do version control like any other version control system, so if you make changes in images you can commit them and **version control** them.

A great feature with Docker is every application works in its own container and does not interferes with other applications working in other containers or on the host OS. So, **isolation** is a great feature with Docker you can relax that your application will not interfere with the applications and other applications will not create any issue with your application.

With Docker, a developer can package all the software and dependencies in the container and Docker will make sure that all this is deployed on all possible platform and everything works fine on every system. So, **Docker makes deployments very easy and fast and more efficient** deployments without worrying about running your app on different platforms. It increases **productivity** many folds.

With all features docker simplifies Devops.

## Docker Beginner Tutorial 4 - How to install DOCKER on LINUX ? Step by Step

https://www.youtube.com/watch?v=KCckWweNSrM&index=4&list=PLhW3qG5bs-L99pQsZ74f-LC-tOEsBp2rK

**How to install Docker on Linux**

Agenda:
Prerequisites
Connect to Linux
Install Docker
Start Docker
Stop Docker
Uninstall Docker

**Prerequisite**
OS should be 64 bit
Linux kernel ver 3.10 or greater

command to check kernel version : $ uname -r

**STEP 1 - Connect to Linux system**
login to aws, go to EC2 console, click on Connect (next to Launch Instance), copy the ssh -I command under Example.

Go to your terminal and connect ec2 instance. Now you are inside linux machine.

$  uname -r # give the info about OS

Go to browser, google “docker manuals” and select https://docs.docker.com/manuals/

Select docker for linux,

**STEP 2 - Install DOCKER**

**Go to the terminal and use below commands**

$ sudo yum -y update # to update all packages

$ sudo yum install -y docker #installs the docker

$ docker #you will see all commands

$ docker –version #docker version 1.12.6, build…

$docker info

Cannot connect to the docker daemon. Is the docker daemon running on this host?

$

**STEP 3 - start Docker**

$ sudo service docker start #to start the docker service on the machine

$ docker info

Now you can add the user to the docker group

$ sudo usermod -a -G docker <ec2-user> # give your user that you logged in
$ docker images # list all the images

$ docker ps # list all running contianers

$ docker ps -a

$ docker run hello-world  # to run hello-world image

$ docker images  # to get list of images present locally

$ docker ps  #to get list of running containers

$ docker ps -a #to get list of all containers

**STEP 4 - stop DOCKER**

$ sudo service docker stop  #to stop the docker service

$ sudo yum remove docker #to uninstall docker

$docker

-bash: /usr/bin/docker: No such file or directory

**To uninstall DOCKER**

$sudo yum remove docker

HELPFUL TIPS
You can visit - https://get.docker.com/
for more installation related help

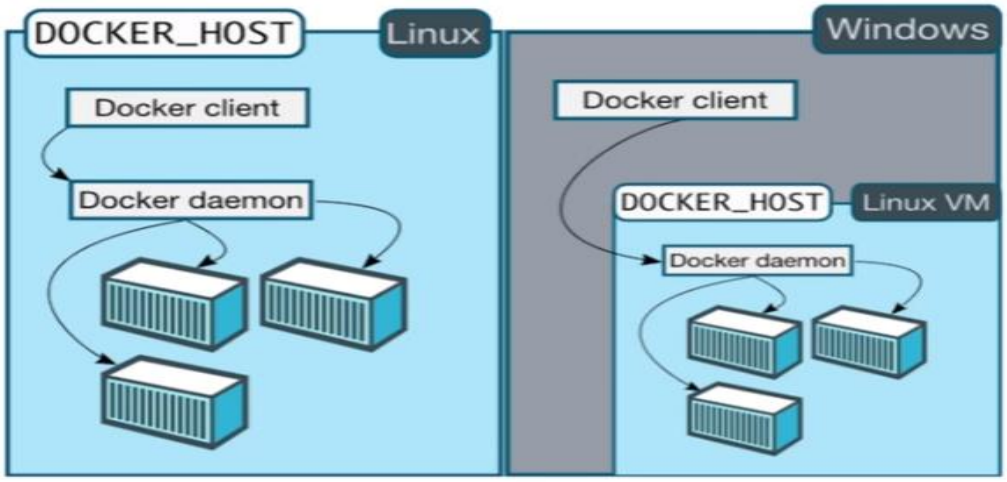
To install docker from binaries
https://docs.docker.com/engine/instal...

Installation steps for amazon ec2
http://docs.aws.amazon.com/AmazonECS/...

## Docker Beginner Tutorial 5 - How to install DOCKER on WINDOWS ? Step by Step

https://www.youtube.com/watch?v=ymlWt1MqURY&list=PLhW3qG5bs-L99pQsZ74f-LC-tOEsBp2rK&index=5

If you install the Docker Toolbox on a Windows machine, the installer automatically installs Oracle Virtualbox to run the Docker virtual machine.




---

Prerequisites
OS - 64 bit
Windows 7 or higher

**How to install Docker on Windows?**

**STEP 1 : Install Docker**

https://docs.docker.com/toolbox/toolbox_install_windows/

https://docs.docker.com/docker-for-windows/install/

**STEP 2 : Troubleshooting (if any issue occurs)**

Convenient way to enable/disable Hyper-V

https://superuser.com/questions/540055/convenient-way-to-enable-disable-hyper-v-in-windows-8

Installing Docker Toolbox on Windows with Hyper-V Installed

https://jayvilalta.com/blog/2016/04/28/installing-docker-toolbox-on-windows-with-hyper-v-installed/

**STEP 3 : validate INSTALLATION**
run some docker commands

- Docker
- docker –version
- docker run "image name"

**STEP 4 : uninstall DOCKER**

**USEFUL LINKS**
https://stackoverflow.com/questions/36885985/cannot-start-docker-after-installation-on-windows

https://docs.docker.com/toolbox/faqs/troubleshoot/

References :
Image of docker architecture
References – https://www.wedidknow.xyz/2017/03/docker-ecosystem-how-to-manage-your.html

# Udemy

## Docker Technology for DevOps: Run docker containers

- **James Lee**

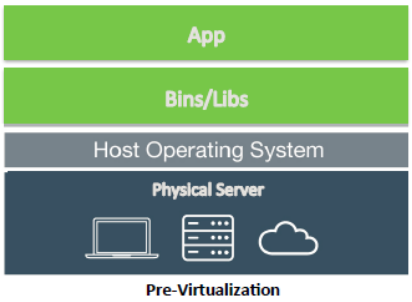
**How to develop and deploy multiservice application with docker technologies?**

**How to automate docker work flow with docker compose?**

**How to deploy docker applications across multiple servers in the cloud? [using docker swarm]**

1) Docker technology is one implementation of container based virtualization technologies.

### Pre-Virtualization World

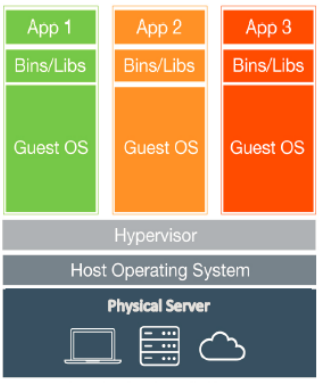


Pre-Virtualization

**Problems:**

- Huge Cost
- Slow Deployment
- Hard to Migrate

### Hypervisor-based Virtualization



Hypervisor-based Virtualization

**Benefits:**

- Cost-Efficient
- Easy to Scale

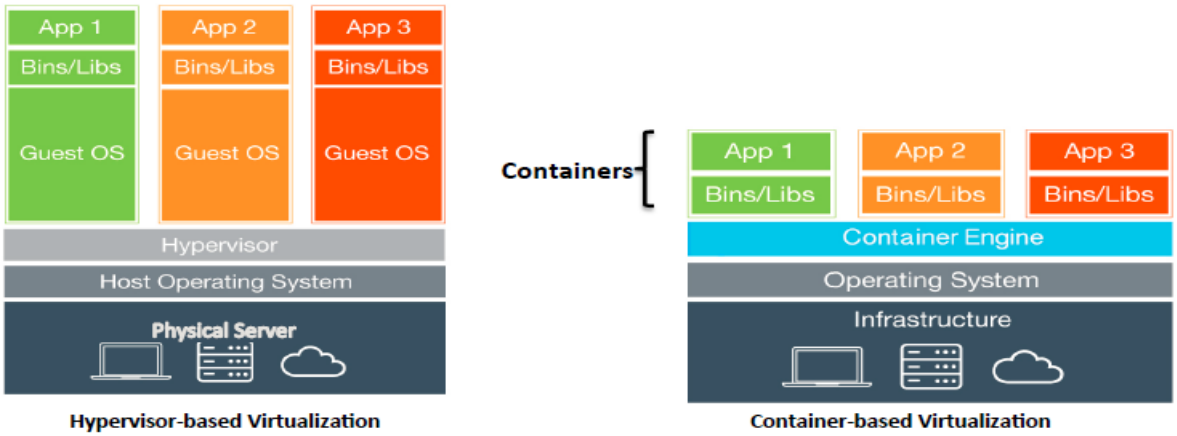
**Limitations:**

- Kernel Resource Duplication
- Application Portability Issue

Some of the Hypervisor providers: VMWare, VirtualBox

Now a days VM are provided in the cloud: AWS, Microsoft Azure

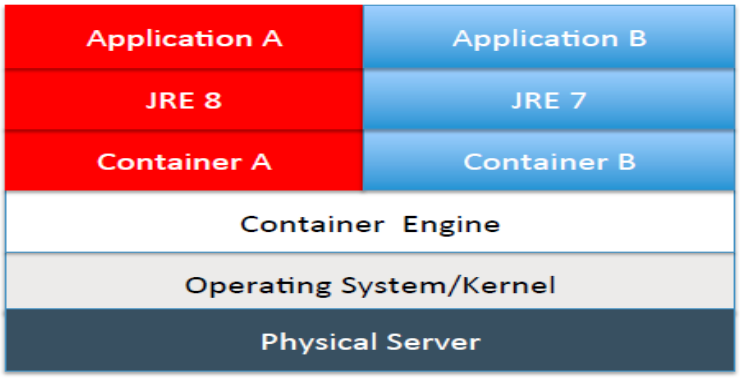
# Hypervisor-based *VS* Container-based Virtualization



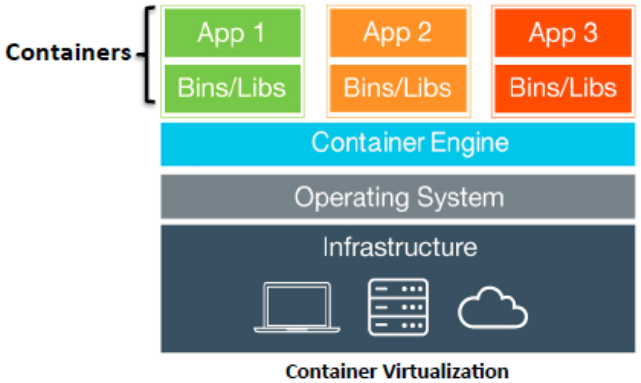

Hypervisor-based Virtualization

Container-based Virtualization

## Runtime Isolation




# Container Virtualization

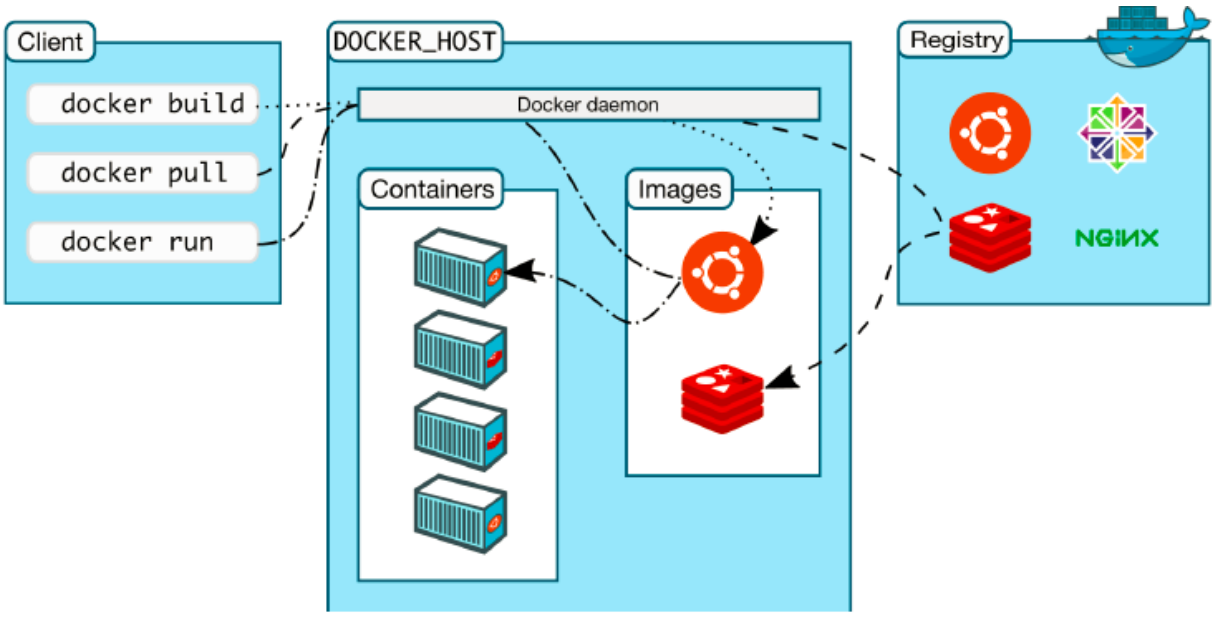



Container Virtualization

## Benefits:

- Cost-Efficient
- Fast Deployment
- Guaranteed Portability

## Docker: Client Server Architecture




**The term Docker can refer to**

- The Docker project as a whole, which is a platform for developers and sysadmins to develop, ship, and run applications
- The docker daemon process running on the host which manages images and containers (also called Docker Engine)

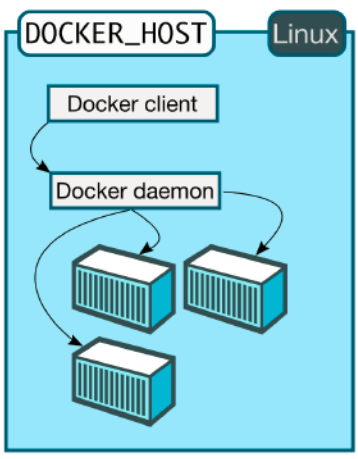
**2 types of docker clients**

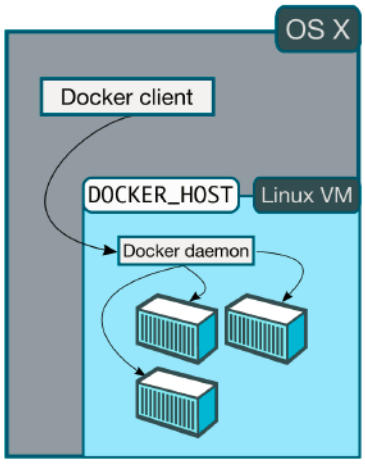
1. Command Line Docker Client
2. Kitematic Docker Client (graphical interface)

Daemon is a persistent process.

Docker Daemon is often referred as "**docker engine"** or "**docker server"**

## Docker Installation:



A docker client can also connect to a remote docker daemon

**Docker daemon can't run on non-linux platforms** natively because it uses **linux specific kernel** features.

## Install Docker for Mac/Windows:

**Install Docker**
*This lecture applies to you if:*
• You are using Linux
• Or you are using Mac and your Mac version is OS X 10.10.3 or newer
• Or you are using Windows and your Windows version is Windows 10 or newer

Otherwise, you can skip this lecture and follow the installation guide of the next lecture.

After successful installation of the docker you can check

C:\Users\mandavaa>**docker --version**
Docker version 17.03.1-ce, build c6d412e

C:\Users\mandavaa>**docker-compose --version**
docker-compose version 1.11.2, build f963d76f

C:\Users\mandavaa>**docker-machine --version**
docker-machine version 0.10.0, build 76ed2a6

**Try something more ambitious, and run an Ubuntu container with this command.**

$ docker run -it ubuntu bash

**Start a Dockerized webserver with this command:**

cmd>docker run -d -p 80:80 --name webserver nginx

**Practice the commands**: https://docs.docker.com/docker-for-windows/#general

## Install Docker Toolbox

**This lecture applies to you if:**
• You are using Mac and your Mac version is older than OS X 10.10.3.
• Or you are using Windows and your Windows version is older than Windows 10.
• Or you want to install Docker Compose, Docker Machine or Kitematic instead of Docker Engine.

Otherwise, you can skip this lecture and follow the installation guide of the previous lecture.
If you already installed Docker for Mac/Windows, you can skip this lecture for now.

**Docker Images**

1) Images are read only templates used to create containers.
2) Images are created with the **docker build** command, either by us or by other docker users.
3) Images are composed of layers of other images.
4) Images are stored in a Docker registry.

**Docker Containers**

1) If an image is a class, then a container is an instance of a class - a run time object.
2) Containers are lightweight and portable encapsulations of an environment in which to run applications.
3) Containers are created from images. Inside a container, it has all the binaries and dependencies needed to run the application.

**Registries and Repositories**

1) A registry is where we store our images.
2) You can host your own registry, or you can use Docker's public registry which is called DockerHub.
3) Inside a registry, images are stored in repositories.
4) 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.

**Docker Hub –** A public Docker registry

Browse "docker hub" in google. Open hub.docker.com, it opens home page, click on "Browse" at the bottom. See the Explore Official Repositories, you can notice the STARS and PULLS, that indicates the popularity of the repositories. All these are certified repositories by docker. You can search for whatever you like. For example mysql.

**Why Using Official Images**

1) Clear Documentation
2) Dedicated Team for Reviewing Image Content
3) Security Update in a Timely Manner

**Run our first Hello World Docker Container**

Browse hub.docker.com, and search for "busybox", it will come up with busybox repositories, first one is busybox official repository. Click on it, see the size, it is tiny size hence we have chosen this to run quickly.

Check the tags for version.

Open docker quick start ternminal(it is for docker toolbox). For windows open command prompt.

**Note:** Docker first look for the images in the local box. If found, docker will use the local image. Otherwise, docker will download the image from remote docker registry.

Cmd> docker images # to see the local images

Docker run command will create the container using the image we specified, then spin up the container and run it.

**docker run repository:tag command [arguments]**

Image is specified by "repository:tag"

Cmd> **docker run busybox:1.24 echo "Hello world"** # what command would like to run on the container, here echo is the command.

Here we do not have local busybox image so it pulls from the remote repository. Now docker downloaded the image and created and run the container. You can see the "Hello world" is displayed.

If you run again the same command it quickly run the container from local image and display the echo command.

Cmd> **docker run busybox:1.24 ls ** #displays the root directory contents bin,dev,etc,home…

**Interactive mode:**

**-i** flag starts an interactive container

**-t** flag creates a pseudo-TTY that attaches stdin and stdout

Cmd> docker run -i -t busybox:1.24

/ # -> it is container bash prompt (because we have given -i -t during container run)

/ # ls -> display container root directories

/ # touch a.txt -> create an empty file

/ # ls -> display container root directories and file a.txt

/ # exit -> to exit the container

Cmd> docker run -i -t busybox:1.24 # it creates a brand new container

/ # ls -> now you will not see a.txt file because it is a brand new container

/# exit

## Deep Dive into Docker Compose Workflow

**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.

## Docker port mapping and Docker Logs Command

Browse hub.docker.com and search for tomcat, it will open the tomcat official repository.

-p option is used to map the port. **-p host_port : container_port**

Cmd> docker run -it -p 8888:8080 tomcat:8.0

Scroll up and get the docker IP, and put on the browser, <IP>:8888 → it open the tomcat home page

If you are using docker on linux, or you are running docker for mac or docker for windows, here you should put localhost:8888 or 127.0.0.1:8888

Most production cases we will run the containers in the background mode. -d option is used for this.

Cmd> docker run -it -d -p 8888:8080 tomcat:8.0 # it will retrun the long container id

Check the docker with docker logs command

Cmd> docker logs <container id> # you will see the container logs

## Foreground vs Detached

| | Run Container in Foreground | Run Container in Background |
|---|---|---|
| *Description* | Docker run starts the process in the container and attaches the console to the process's standard input, output, and standard error. | Containers started in detached mode and exit when the root process used to run the container exits. |
| *How to specify?* | default mode | -d option |
| *Can the console be used for other commands after the container is started up?* | No | Yes |

## Docker Image Layers

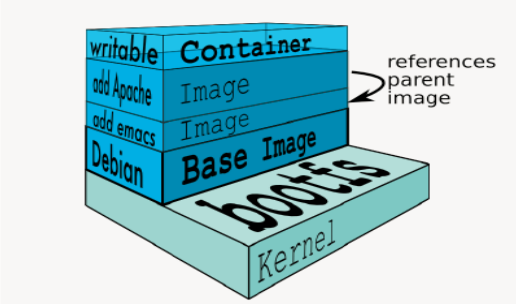
Docker image is made up of list of read only layers, that represent the file system differences.

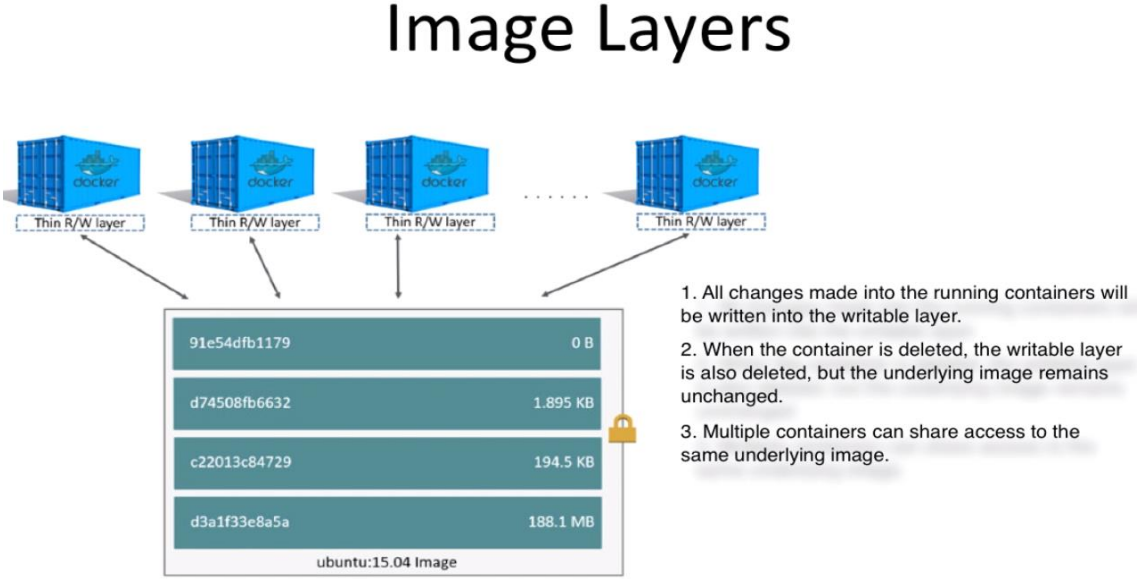
Image layers stacked on top of each other forms a base for the containers file system.

You can see the docker image layers by using "docker history repository:tag" command

Cmd> docker history busybox:1.24 # it will list all the layers on the busy box image

## Image Layers






## Build Docker Images

**Approach 1: committing changes made in a container**

**Ways to Build a Docker Image**
• Commit changes made in a Docker container.
• Write a Dockerfile.

**Steps**
1. Spin up a container from a base image.
2. Install Git package in the container.
3. Commit changes made in the container.

**Step 1:** browse hub.docker.com, search for debian, get the tag 'jessie'. Open terminal or cmd prompt.

Cmd> docker run -it debian:jessie #pulls repo from remote docker hub and image downloaded and you are inside the container

# ls # displays the root directory structure of the container, it is typical Debian file system

# git # command not found

# apt-get update && apt-get install -y git # updated the container and install git

# git # help message

#exit

Cmd>

**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***

cmd> docker ps -a # to get the container id

cmd> docker commit <container_id> anil\debian:1.00 #return the long id of the new image

cmd> docker images # you will see the new image name anil\debian with tag 1.00.

you may notice the this image is bigger than the base Debian image because we are extended the file system. (updates and git installed). Images are made with the file system layers.

Now spin up the new image

Cmd> docker run -it anil\debina:1.00 # it return with container command prompt

/ # git # it will display help.

/ # exit

## Build Docker Images

**Approach 2: Writing a Dockerfile**

**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.

Lets create a docker file. It must be Dockerfile without any extension and capital D.

$ touch Dockerfile

$ vi Dockerfile
FROM debian:jessie # not case sensitive, but to distinguish we will write instruction in caps
RUN apt-get update
RUN apt-get install -y git
RUN apt-get install -y vim
Save the file

$

## "build" command to create an image from Dockerfile

**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.

$ docker build -t jameslee/debian . # . represent the current path where we created the Dockerfile

When the docker build command runs first it will take **FROM debian:jessie** with some container id (xyz), once Debian installed then next instruction **RUN apt-get update** start executing with some other container id(abc) once updates are installed, the first container(xyz) is removed, then **"RUN apt-get install -y git"** run with some container id(mno) once git is installed it will remove the second container(abc) and so on.

**How to run a command in a running container?**
Cmd> docker exec -it <container_id> bash

$ docker images
Jameslee\debian created with the tag “latest” as we did not mentioned any tag while building image.

## Dockerfile in depth

**Chain RUN Instructions**

1) Each RUN command will execute the command on the top writable layer of the container, then commit the container as a new image.
2) The new image is used for the next step in the Dockerfile. So each RUN instruction will create a new image layer.
3) It is recommended to chain the RUN instructions in the Dockerfile to reduce the number of image layers it creates.

```
$vi Dockerfile
FROM debian:jessie
RUN apt-get update && apt-get install -y \
        git \
        vim
:wq
$
```

```
Cmd>docker build -t anil/debian:4.0 - < Dockerfile
Cmd> docker images

Cmd> docker stop 41da6997f8dd              # to stop a container, all data will be lost
Cmd> docker ps
Cmd> docker run -itd anil/Debian:4.0       # run a container with detached(background) mode
Cmd> docker ps
Cmd>
```

**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          reusing cache
RUN apt-get update         reusing cache
RUN apt-get install -y git curl
```

**Cache Busting**

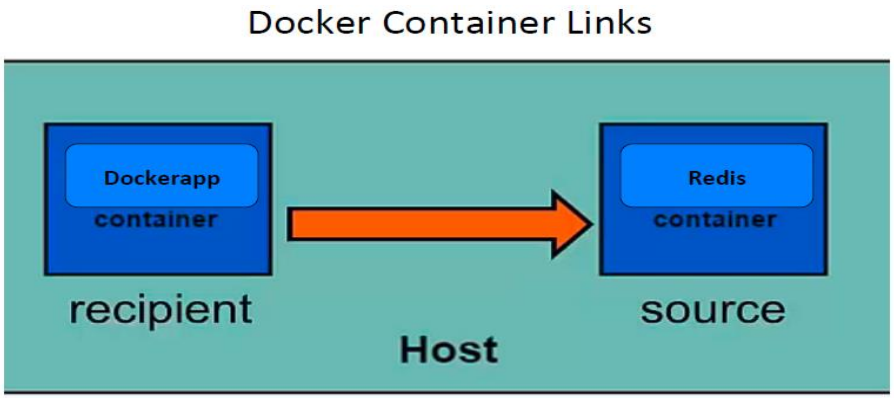
```
FROM ubuntu:14.04
RUN apt-get update && apt-get install -y \
git \
curl
```

- 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.*
```

## Dockerize a Hello World Web Application

**Check out source code:**
git clone -b v0.1 https://github.com/jleetutorial/dockerapp.git




### 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.

### Docker Compose

- 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.

**Why Docker Compose?**
Manual linking containers and configuring services become impractical when the number of containers grows.

## Write and Run Unit Tests in Docker Containers

- 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.

## Incorporating Unit Tests into Docker Images

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

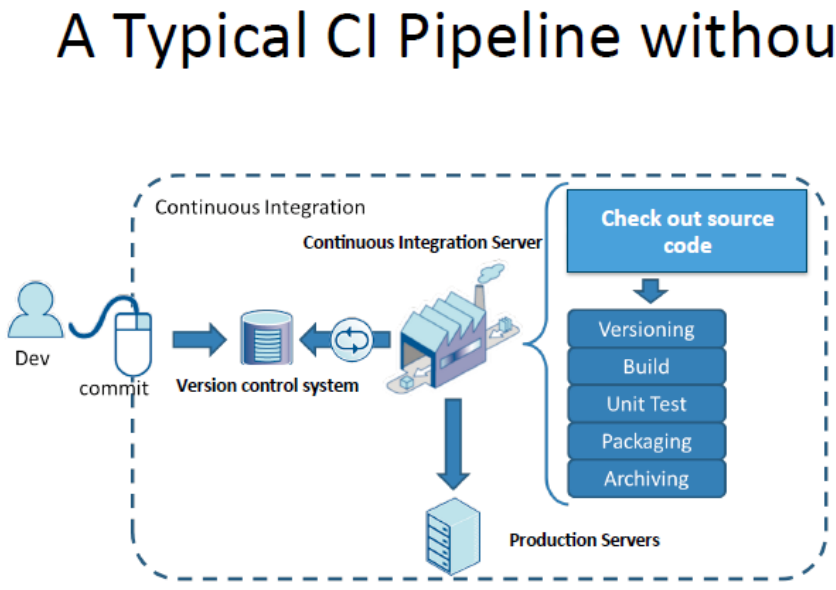
**Cons:**
• It increases the size of the image.

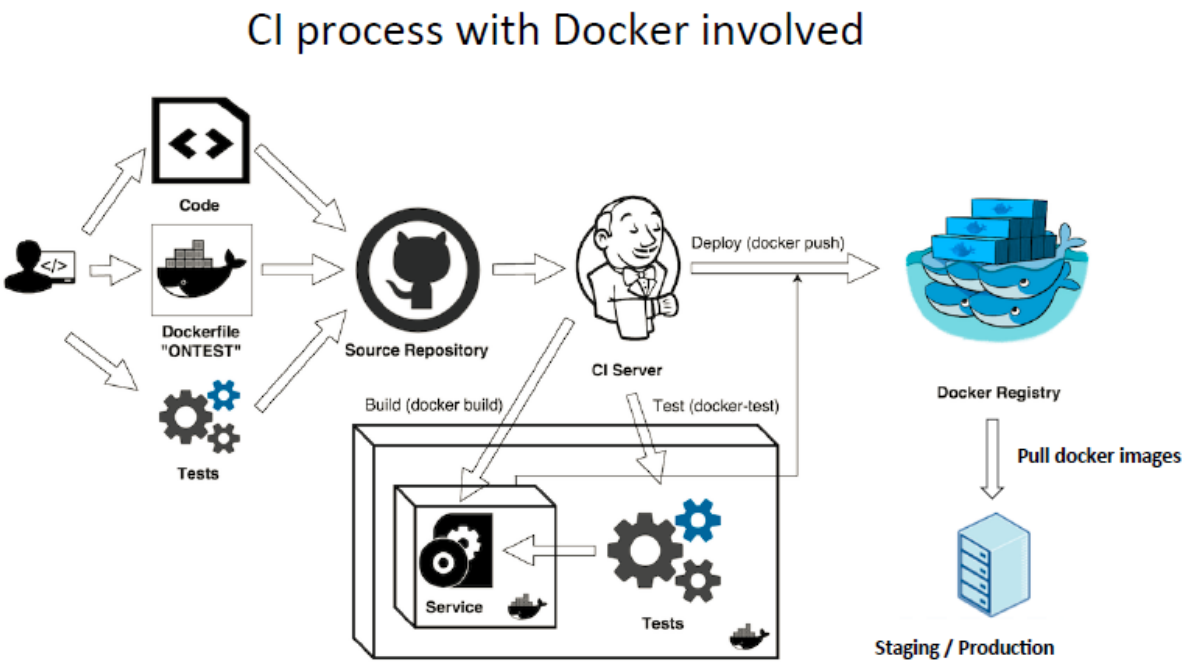
## Fit Docker into Continuous Integration(CI) Process

### What is Continuous Integration?

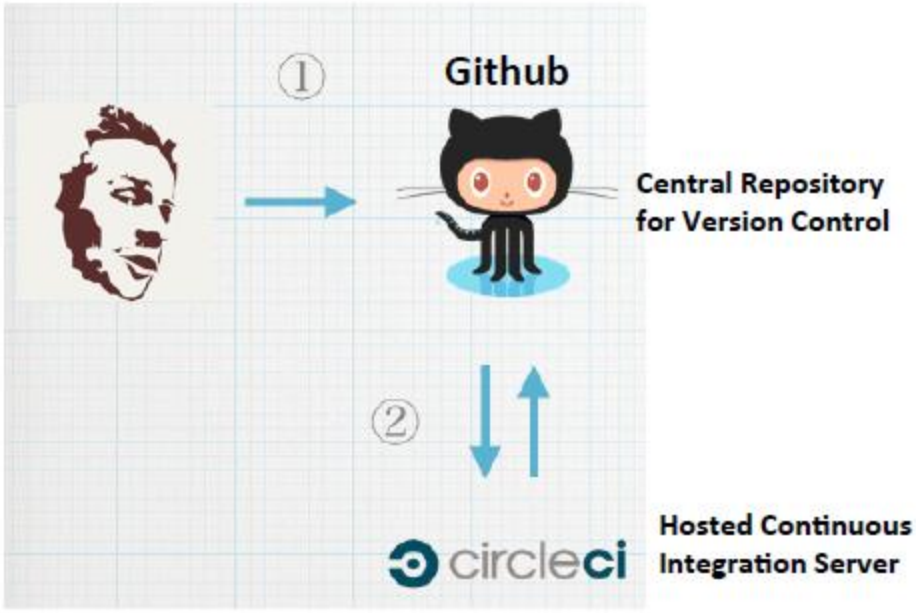
- Continuous integration is a software engineering practice in which isolated changes are immediately tested and reported when they are added to a larger code base.

- The goal of Continuous integration is to provide rapid feedback so that if a defect is introduced into the code base, it can be identified and corrected as soon as possible.




CI process with Docker involved
Code
Dockerfile
"ONTEST"
Tests
Source Repository
CI Server
Deploy (docker push)
Docker Registry
Pull docker images
Staging / Production
Build (docker build)
Test (docker-test)
Service
Tests

# Our Continuous Integration Pipeline




## Set up SSH keys for Github Account

- SSH keys are a way to identify trusted computers without involving password.
- Generate a SSH key pair and save the private SSH key in your local box and add the public key to your GitHub account.
- Then you can directly push your changes to **github** repository without typing password.

## How to check if SSH public key files are available on your local box?

- The SSH public key file usually sits under ***~/.ssh/*** directory and ends with ***.pub*** extension.

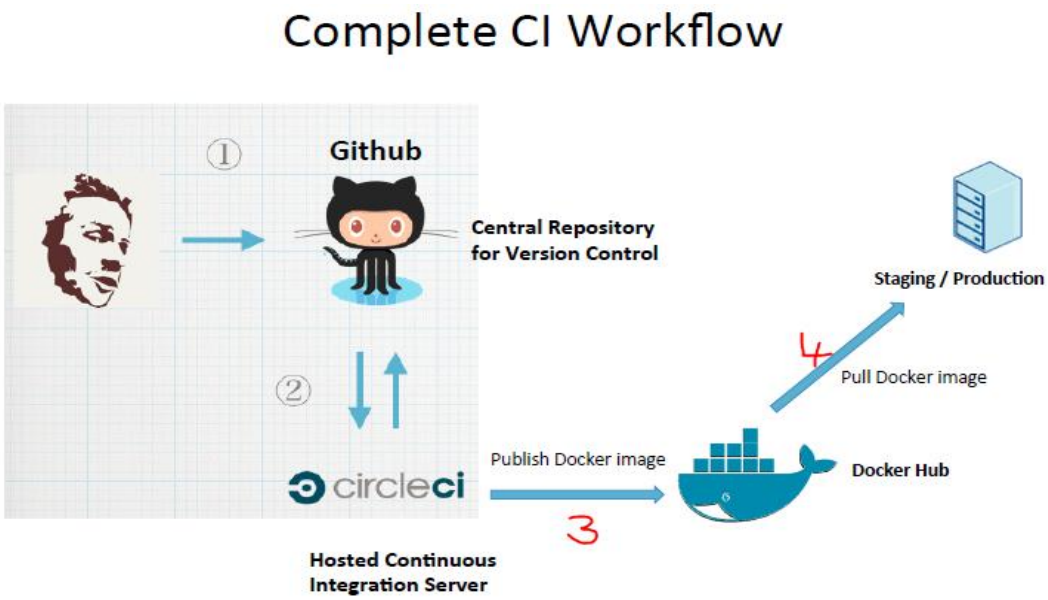
## Link Circle CI with GitHub Account
**(to build a Continuous Integration pipeline)**

## Text Direction: Introduction to Continuous Integration

**URL of the Github account to fork:**
https://github.com/jleetutorial/dockerapp
**Checking for existing SSH keys:**
https://help.github.com/articles/checking-for-existing-ssh-keys/
**Generating a new SSH key and adding it to the ssh-agent:**
https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
**Adding a new SSH key to your GitHub account:**
https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/

## Link Circle CI with GitHub Account

### Publish Docker Images from CircleCI




### Tag the Docker Images with Two Tags

1. commit hash of the source code
2. latest

## Introduction to Running Docker in Production

### Opinions about Running Docker in Production

- One hand, many docker pioneers are confident that a distributed web app can be deployed at scale using Docker and have incorporated Docker into their production environment.
- On the other hand, there are still some people who are reluctant to use Docker in production as they think docker workflow is too complex or unstable for real life use cases.

### Is Docker Production Ready Now?

#### Concerns about Running Docker in Production

- There are still some missing pieces about Docker around data persistence, networking, security and identity management.
- The ecosystem of supporting Dockerized applications in production such as tools for monitoring and logging are still not fully ready yet.

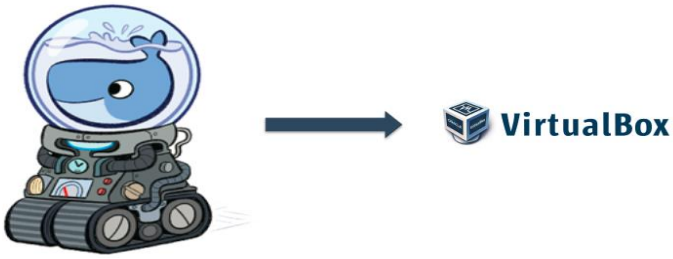
#### Why Running Docker Containers inside VMs?

- To address security concerns.
- Hardware level isolation.

They all run containers inside VMs

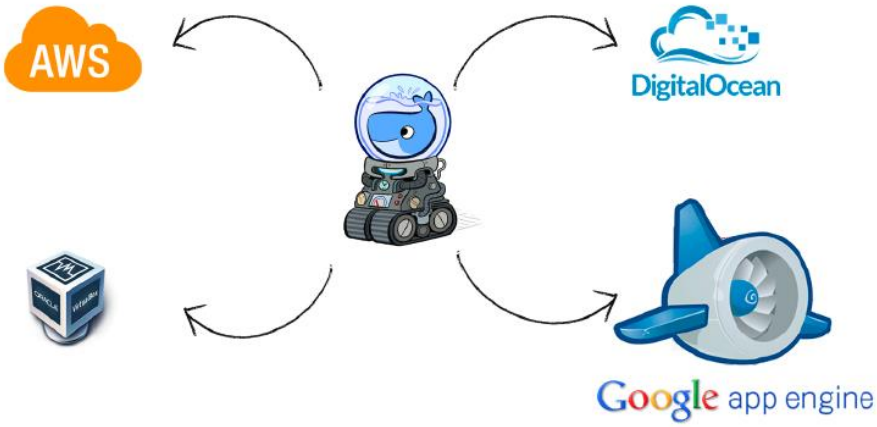

---

Docker Machine




---

Various Docker Machine Drivers




---

## Deploy Docker App to the Cloud with Docker Machine

### Docker Machine Create command

docker-machine create --driver digitalocean --digitalocean-access-token <xxxxx> docker-app-machine

## Refactor Docker Compose File

### "Extends" keywords in Docker Compose File

- The extends keyword enables sharing of common configurations among different files or even different projects entirely.
- Extending services is useful if you have several different environments that reuse a common set of configuration options.

## Introduction to Docker Swarm and Service Discovery

**Step 1: Expose environment variables (If you are using Windows, replace "export" with "set" command)**
export DIGITALOCEAN_ACCESS_TOKEN=<YOUR_DIGITALOCEAN_TOKEN>
export DIGITALOCEAN_PRIVATE_NETWORKING=true
export DIGITALOCEAN_IMAGE=debian-8-x64

**Step 2: Provision consul machine**
docker-machine create -d digitalocean consul

**Step 3: Display the network configura,on of the consul machine**
docker-machine ssh consul ifconfig

**Step 4: Ping the private and public IP address of the consul machine.**
$(docker-machine ssh consul 'ifconfig eth0 | grep "inet addr:" | cut -d: -f2 | cut -d" " -f1')
$(docker-machine ssh consul 'ifconfig eth1 | grep "inet addr:" | cut -d: -f2 | cut -d" " -f1')

**Step 5: Export the private IP to KV_IP environment variable**
export KV_IP=$(docker-machine ssh consul 'ifconfig eth1 | grep "inet addr:" | cut -d: -f2 | cut -d" " -f1')

**Step 6: Configure Docker client to connect to the consul machine**
eval $(docker-machine env consul)

**Step 7: Start the consul container in the consul machine**
docker run -d -p ${KV_IP}:8500:8500 -h consul --restart always gliderlabs/consul-server -bootstrap

## Deploy Docker App to the Cloud via Docker Swarm

**Create Swarm master:**
```
docker-machine create -d digitalocean --swarm \
--swarm-master \
--swarm-discovery="consul://${KV_IP}:8500" \
--engine-opt="cluster-store=consul://${KV_IP}:8500" \
--engine-opt="cluster-adver=se=eth1:2376" \
master
```

### Create Swarm slave:
```
docker-machine create \
-d digitalocean \
--swarm \
--swarm-discovery="consul://${KV_IP}:8500" \
--engine-opt="cluster-store=consul://${KV_IP}:8500" \
--engine-opt="cluster-adver=se=eth1:2376" \
slave
```