DevOps

DevOps (Development and Operations) is a set of practices that aim to automate and improve the process of software development and IT operations. It involves collaboration between developers and IT professionals to deliver software and services more efficiently. Below is a general outline for a DevOps tutorial. Keep in mind that the tools and technologies mentioned may evolve, so it's essential to check for the latest information.

1. Introduction to DevOps:

- Understand the principles of DevOps.
- Explore the benefits of adopting a DevOps culture.
- Learn about the key practices and concepts, such as Continuous Integration (CI) and Continuous Deployment (CD).

2. Version Control:

- Introduction to version control systems like Git.
- Learn branching strategies and best practices.
- Understand how version control fits into the DevOps workflow.

3. Continuous Integration (CI):

- Set up a CI/CD pipeline using popular CI tools like Jenkins, GitLab CI, or Travis CI.
- Configure automated builds and tests.
- Understand the importance of automated testing in the CI process.

4. Infrastructure as Code (IaC):

- Introduction to Infrastructure as Code concepts.
- Learn a tool like Terraform or Ansible for provisioning and managing infrastructure.
- Create and manage infrastructure using code.

5. Containerization:

- Introduction to containerization with Docker.
- Understand the Docker ecosystem and basic commands.
- Dockerizing applications and services.

6. Orchestration:

- Overview of container orchestration tools like Kubernetes.
- Deploy and manage containerized applications with Kubernetes.
- Understand concepts like pods, services, and deployments.

7. Continuous Deployment (CD):

- Implement CD pipelines for deploying applications.
- Explore blue-green deployments and canary releases.
- Monitor and log deployments for troubleshooting.

8. Monitoring and Logging:

- Introduction to monitoring tools like Prometheus, Grafana, or ELK stack.
- Set up monitoring for infrastructure and applications.
- Configure logging for better visibility into application behavior.

9. Collaboration and Communication:

- Use collaboration tools like Slack, Microsoft Teams, or others for team communication.
- Implement ChatOps for better collaboration between development and operations teams.

10. Security in DevOps:

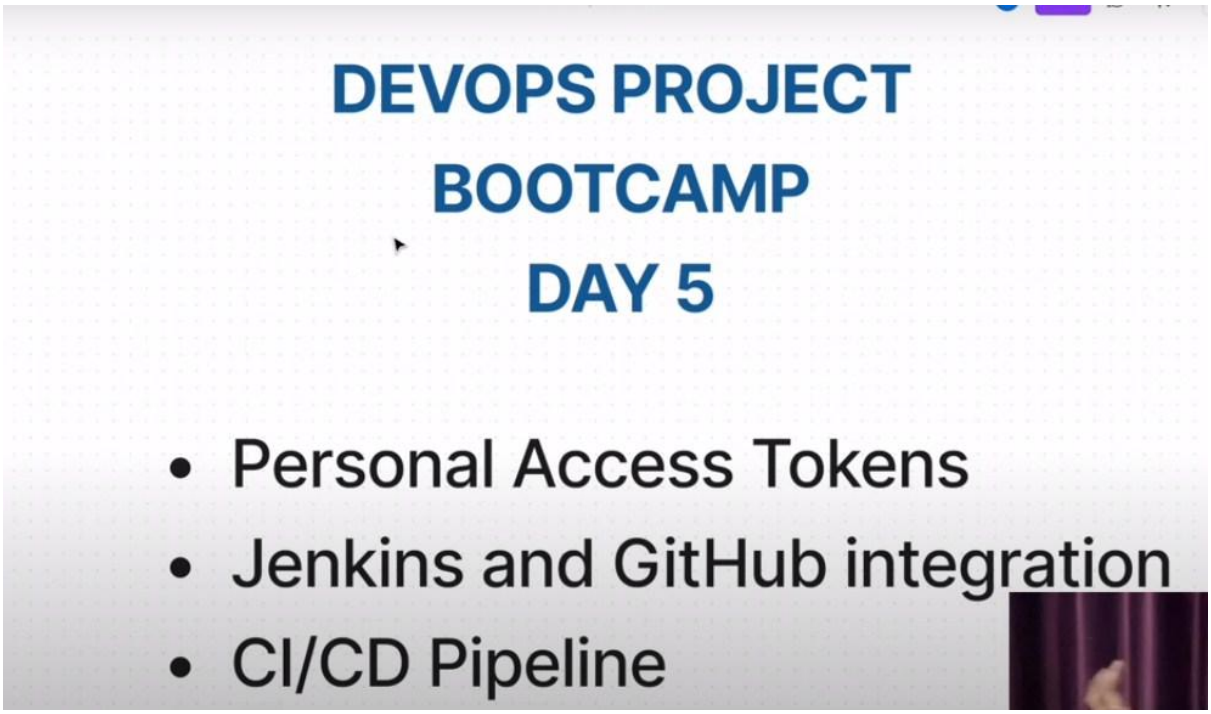
- Implement security best practices throughout the DevOps lifecycle.
- Learn about static code analysis, vulnerability scanning, and security testing.

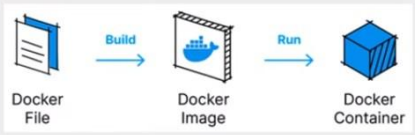
11. DevOps Culture and Best Practices:

- Foster a DevOps culture within your organization.
- Explore case studies and success stories.
- Continuous improvement and feedback loops.

Additional Tips:

- Stay updated with the latest tools and practices in the DevOps space.
- Participate in forums, conferences, and online communities to learn from others.
- Consider certifications like AWS Certified DevOps Engineer, Docker Certified Associate, or Kubernetes certifications.





Create a Dockerfile

Add instructions in Dockerfile to create Docker image

Run Dockerfile to create Docker image

Run Docker image to create Docker container

Access the application running in Docker container

AutomationStepByStep.com

Dockerfile > Docker Image > Docker Container > Access the App

Step 1 - Create a new directory

```
mkdir myapp
cd myapp
```

Step 2 - Create a file called "index.html"

```
echo "Hello, world!" > index.html
```

Step 3 - Create a file named Dockerfile

```
touch Dockerfile
```

A Dockerfile is a text file with instructions to build a Docker Image

When we run a Dockerfile, Docker image is created

When we run the docker image, containers are created

Step 4 - Open the "Dockerfile" file in a text editor and add the following lines:

```
FROM nginx
COPY index.html /usr/share/nginx/html
```

This Dockerfile defines a new Docker image that
- uses the official **nginx** image as a base
- then copy the index.html file to the appropriate location in the image

Step 5 - Start docker & Build docker image from dockerfile

```
docker build -t myapp .
```

This command builds a new Docker image with the tag "myapp" using the Dockerfile in the current directory.

Step 6 - Run docker container from the image

```
docker run -p 8080:80 myapp
```

This tells Docker to run the myapp container and map port 8080 on your local machine to port 80 inside the container.

Step 7 - Access the app

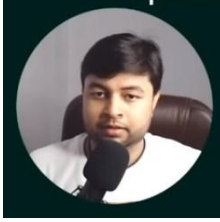
Open a web browser and navigate to http://localhost:8080 to see the "Hello, world!" message displayed in your web browser.

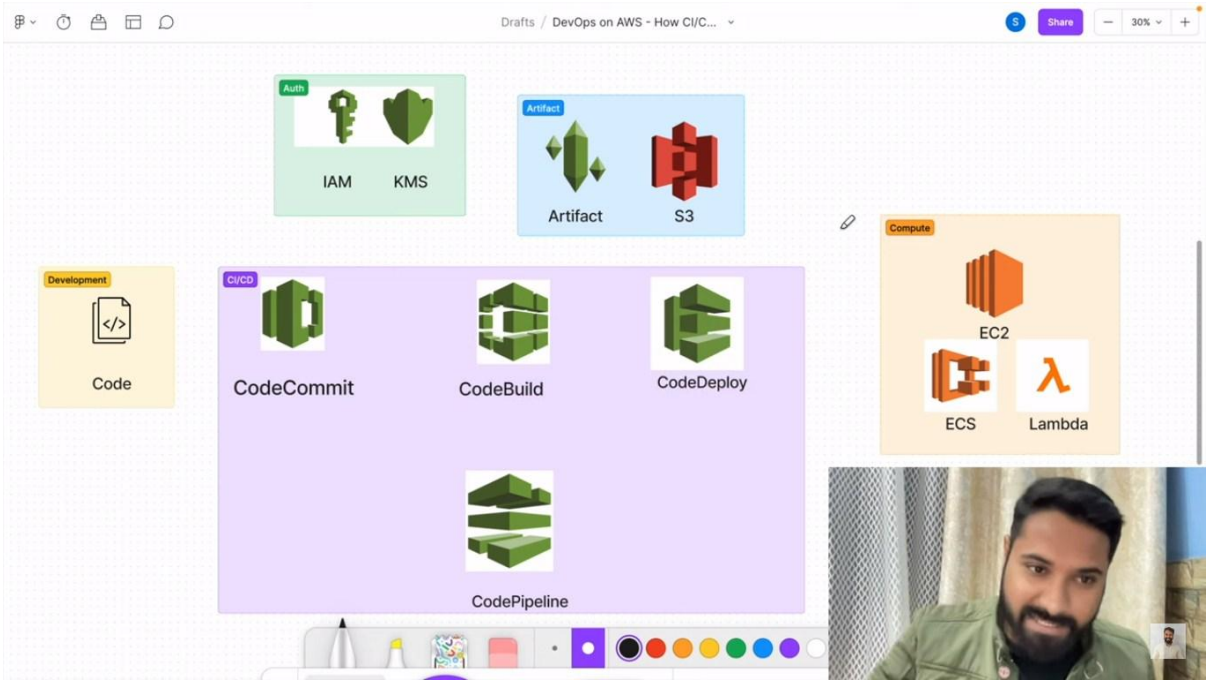
AutomationStepByStep.com

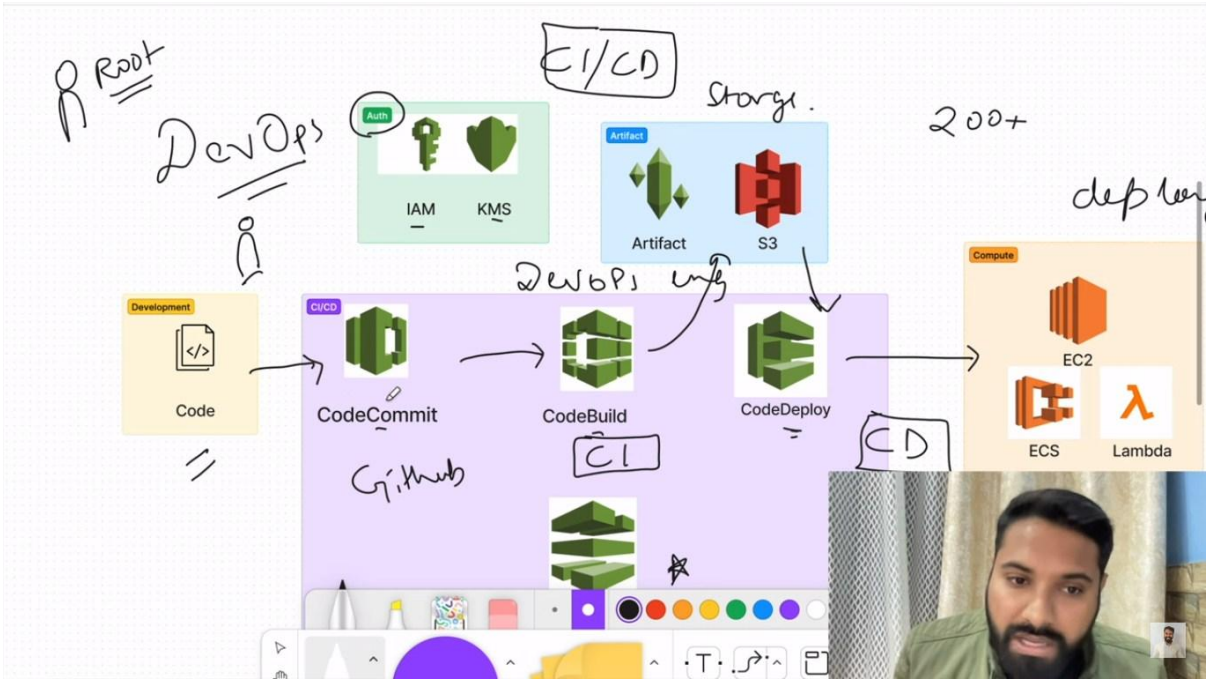
What is Docker ?

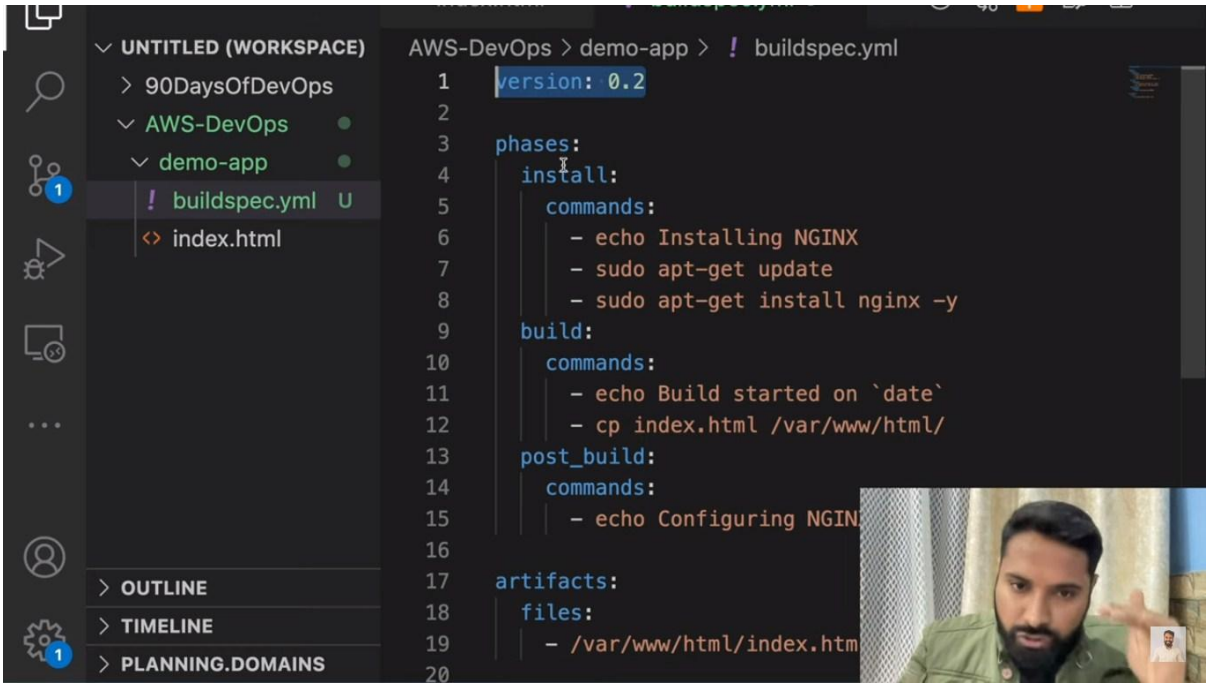
Docker is an open platform for developing, shipping, and running applications.

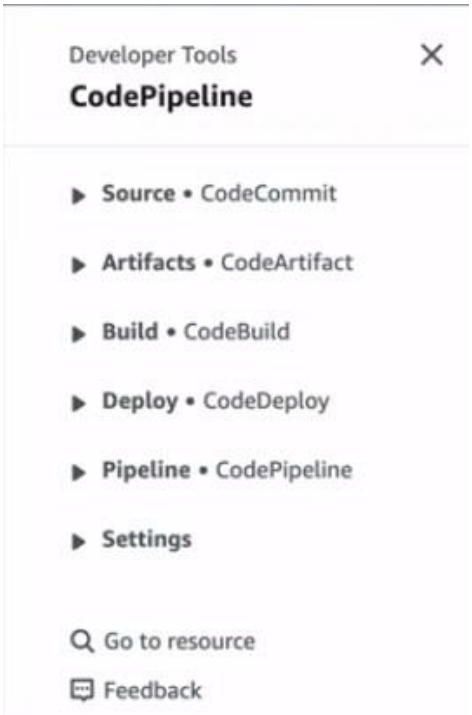
Docker is a platform which packages an application and all its dependencies together in the form of containers






Root
DevOps
CI/CD
Storage
200+
Auth
IAM
KMS
Artifact
Artifact
S3
Development
Code
CI/CD
CodeCommit
CodeBuild
CodeDeploy
Compute
EC2
ECS
Lambda
CI
CD
GitHub


UNTITLED (WORKSPACE)
90DaysOfDevOps
AWS-DevOps
demo-app
buildspec.yml U
index.html
OUTLINE
TIMELINE
PLANNING.DOMAINS
AWS-DevOps > demo-app > ! buildspec.yml
version: 0.2
phases:
install:
commands:
- echo Installing NGINX
- sudo apt-get update
- sudo apt-get install nginx -y
build:
commands:
- echo Build started on `date`
- cp index.html /var/www/html/
post_build:
commands:
- echo Configuring NGIN
artifacts:
files:
- /var/www/html/index.htm


Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Go to resource
Feedback

Jenkins

Certainly! Jenkins is a popular open-source automation server that helps automate parts of the software development process. It supports building, deploying, and automating any project.

Here's a basic tutorial to get you started with Jenkins:

Step 1: Install Jenkins

1. **Download Jenkins:** Visit the Jenkins official website and download the latest stable version for your operating system.
2. **Install Jenkins:** Follow the installation instructions provided for your specific operating system.
3. **Start Jenkins:** Once installed, start the Jenkins service. You can access Jenkins by opening a web browser and navigating to `http://localhost:8080` (by default).
4. **Unlock Jenkins:** During the first run, Jenkins will ask for an initial password. Retrieve this password from the Jenkins server's file system and follow the setup process.

Step 2: Configure Jenkins

1. **Install Plugins:** Jenkins uses plugins to extend its functionality. After unlocking Jenkins, you can choose to install suggested plugins or select them manually. Common plugins include Git, GitHub, and Maven.
2. **Create Admin User:** Set up the admin user credentials for your Jenkins instance.
3. **Configure Jenkins URL:** Configure the Jenkins URL to match your environment.

Step 3: Create a New Jenkins Job

1. **Create a New Job:**
   - Click on "New Item" on the Jenkins dashboard.
   - Enter a name for your job and select the type of job you want to create (Freestyle project is a good start).
2. **Configure Source Code Management:**
   - If your project is in Git, enter the repository URL and credentials if required.
3. **Build Trigger:**
   - Define when the build should be triggered (e.g., on code commit, scheduled, etc.).
4. **Build Steps:**
   - Add build steps, such as shell commands, to execute during the build process.
5. **Post-build Actions:**
   - Define actions to be taken after the build, such as archiving artifacts or triggering other jobs.
6. **Save and Build:**
   - Save your job configuration and manually trigger the first build to verify everything is set up correctly.

Step 4: Explore Jenkins Features

1. **View Build History:**
   - Check the build history on the job dashboard.
2. **Console Output:**
   - Examine the console output for detailed information about the build process.
3. **Configure Jenkins Global Settings:**
   - Explore and configure Jenkins global settings as needed.

Step 5: Jenkins Pipelines (Optional)

Jenkins supports defining pipelines as code using the Jenkinsfile. A Jenkinsfile can be stored in your version control system and provides a way to define complex build and deployment workflows.

1. **Create a Jenkinsfile:**
    - Create a file named `Jenkinsfile` in your project repository.
2. **Pipeline Syntax:**
    - Define stages, steps, and post-build actions using the Jenkins pipeline syntax.
3. **Configure Jenkins to Use Jenkinsfile:**
    - Configure your Jenkins job to use the Jenkinsfile from your version control system.

How to integrate jenkins with gitlab

Integrating Jenkins with GitLab allows you to automate builds, tests, and deployments triggered by changes in your GitLab repositories. Here's a step-by-step guide on how to integrate Jenkins with GitLab:

Prerequisites:

1. **Install Jenkins:** Make sure Jenkins is installed and running on your server. You can follow the installation steps provided in the Jenkins documentation.
2. **Set up a GitLab Account:** You need a GitLab account and a repository with a project to integrate with Jenkins.

Integration Steps:

1. Install Required Jenkins Plugins:

- Open Jenkins and navigate to "Manage Jenkins" > "Manage Plugins."
- In the "Available" tab, search for and install the following plugins:
    - Git Plugin
    - GitLab Plugin
    - Pipeline Plugin (if you plan to use Jenkins Pipeline)

Restart Jenkins after installing the plugins.

2. Configure GitLab Integration in Jenkins:

- In Jenkins, go to "Manage Jenkins" > "Configure System."
- Scroll down to the "GitLab" section.
- Configure the following:
    - **Connection Settings:**
        - Check "Enable authentication."
        - Set the "Connection name."
        - Enter the GitLab server URL.
        - Add an "API Token" for authentication.
    - **Webhook:**
        - Check "Enable authentication."
        - Set a "Secret Token" (you'll need this when configuring the GitLab webhook).
- Click "Save" at the bottom of the page.

3. Configure Webhook in GitLab:

- In your GitLab project, go to "Settings" > "Webhooks."
- Add a new webhook:
    - Set the "URL" to your Jenkins webhook URL (http://your-jenkins-server/gitlab/notify_commit).
    - Add the "Secret Token" configured in Jenkins.
    - Select the events that should trigger the webhook (e.g., Push events).
    - Save the webhook.

4. Create a Jenkins Job for Your GitLab Project:

- In Jenkins, click "New Item" to create a new job.
- Enter a name for your job and select the type (Freestyle project or Pipeline).
- Configure your source code management (e.g., Git) with your GitLab repository URL.
- Set up any build triggers (e.g., Poll SCM or webhook trigger for Jenkins Pipeline).
- Configure build steps, post-build actions, etc.
- Save the job.

5. Test the Integration:

- Make a change in your GitLab repository and push it.
- Check the Jenkins job to see if it triggers a build.

Additional Notes:

- For Jenkins Pipeline users, you can define your build steps and pipeline in a Jenkinsfile in your GitLab repository.
- Jenkins and GitLab should be accessible to each other over the network.
- Ensure proper firewall and security group configurations if Jenkins and GitLab are running on different servers.

Step of creation ci cd pipeline

Creating a CI/CD (Continuous Integration/Continuous Deployment) pipeline with Jenkins typically involves defining a set of steps to automate the build, test, and deployment processes. Below is a basic guide for creating a CI/CD pipeline using Jenkins:

Prerequisites:

1. **Jenkins Installed:** Ensure Jenkins is installed and running on your server. You should also have the necessary plugins installed, such as Git, GitLab, or others based on your version control system and deployment targets.
2. **Source Code Repository:** Your project should be hosted in a version control system (e.g., GitLab, GitHub).

Steps to Create a CI/CD Pipeline:

1. Create a New Jenkins Job:

- Open Jenkins and click on "New Item."
- Enter a name for your job (e.g., "MyApp_CI_CD_Pipeline").
- Choose the type of project (e.g., Freestyle project or Pipeline).
- Click "OK."

2. Configure Source Code Management:

- In the job configuration, find the "Source Code Management" section.
- Choose your version control system (e.g., Git).
- Provide the repository URL and credentials if necessary.

3. Define Build Steps:

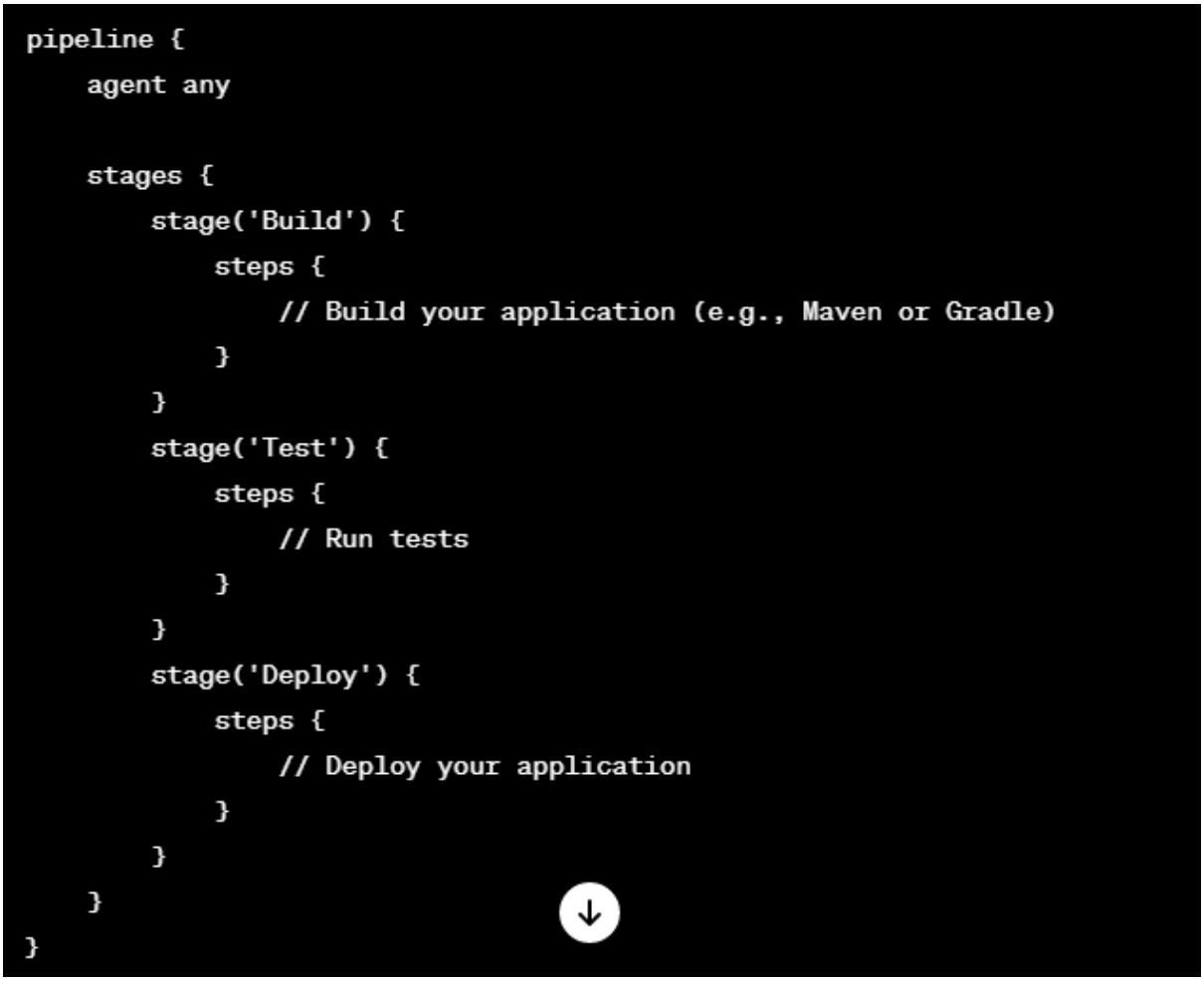
For Freestyle Project:

- In the job configuration, find the "Build" section.
- Add build steps, which could include commands to compile code, run tests, etc.

For Pipeline Project:

- If using Jenkins Pipeline, you'll define your pipeline in a `Jenkinsfile` in your repository.
- Create a `Jenkinsfile` with stages for build, test, and deployment.

Example `Jenkinsfile`:



```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                // Build your application (e.g., Maven or Gradle)
            }
        }
        stage('Test') {
            steps {
                // Run tests
            }
        }
        stage('Deploy') {
            steps {
                // Deploy your application
            }
        }
    }
}
```

4. Add Post-Build Actions (Optional):

- In the job configuration, find the "Post-build Actions" section.
- Configure actions like archiving artifacts, sending notifications, etc.

5. Save and Run the Pipeline:

- Save the job configuration.
- Manually trigger the pipeline or configure it to be triggered automatically on code changes.

6. Monitor Pipeline Execution:

- View the progress of your pipeline in the Jenkins dashboard.
- Check the console output for each build stage to identify any issues.

7. Add CD (Continuous Deployment) Steps:

- Extend your pipeline to include deployment steps to your target environment.

Additional Considerations:

- **Environment Variables:** Use Jenkins environment variables for sensitive data like API keys or credentials.
- **Notifications:** Configure notifications (email, Slack, etc.) to alert the team about build and deployment status.
- **Integration with Version Control:** Leverage webhooks or polling to trigger the pipeline on code changes.
- **Testing:** Implement thorough testing stages in your pipeline to catch issues early.

how to install and configure jenkins on aws server

Installing and configuring Jenkins on an AWS (Amazon Web Services) server involves several steps. Below is a basic guide to help you set up Jenkins on an AWS EC2 instance. Please note that the following instructions may need adjustments based on your specific setup and requirements.

Step 1: Launch an EC2 Instance

1. **Sign in to AWS Console:** Log in to the AWS Management Console.
2. **Launch EC2 Instance:**
   - Navigate to the EC2 service.
   - Click "Launch Instance" and choose an Amazon Machine Image (AMI). A popular choice is an Amazon Linux AMI.
   - Select an instance type, configure instance details, add storage, and configure security groups.
3. **Create or Use an Existing Key Pair:**
   - Choose an existing key pair or create a new one. You'll need this key pair to connect to your instance.
4. **Launch the Instance:**
   - Review your settings and click "Launch."
   - Select your key pair and click "Launch Instances."

Step 2: Connect to Your EC2 Instance

Use an SSH client to connect to your EC2 instance using the private key associated with the key pair you selected during the instance launch.

Example SSH command:

bash Copy code

```
ssh -i path/to/your-key.pem ec2-user@your-instance-ip
```

Step 3: Install Jenkins on the EC2 Instance

1. **Update the Package Repositories:**

bash Copy code

```
sudo yum update -y
```

2. **Install Java:**

Jenkins requires Java. Install OpenJDK 8 or a version compatible with Jenkins.

bash Copy code

```
sudo yum install java-1.8.0-openjdk -y
```

3. **Add Jenkins Repository:**

```bash
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins.io/redhat/jenkins.repo
```

4. **Import Jenkins Key:**

```bash
sudo rpm --import http://pkg.jenkins.io/redhat/jenkins.io.key
```

5. **Install Jenkins:**

```bash
sudo yum install jenkins -y
```

6. **Start Jenkins Service:**

```bash
sudo service jenkins start
```

7. **Enable Jenkins to Start on Boot:**

```bash
sudo chkconfig jenkins on
```

Step 4: Configure Jenkins

1. **Access Jenkins Web Interface:**

   Open a web browser and navigate to `http://your-instance-ip:8080`.

2. **Unlock Jenkins:**

   Follow the on-screen instructions to retrieve the initial administrator password from the server log.

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

3. **Install Recommended Plugins:**

   Choose the "Install suggested plugins" option during the setup.

4. **Create Admin User:**

   Set up an administrator account for Jenkins.

5. **Configure Jenkins URL:**

   Set the Jenkins URL based on your server's public DNS or IP.

6. **Start Using Jenkins:**

   Click "Start using Jenkins."

Step 5: Configure Security Groups

Ensure that the security group associated with your EC2 instance allows incoming traffic on port 8080 (Jenkins web interface).

Step 6: Install Additional Tools and Plugins (Optional)

Depending on your project requirements, you may need to install additional tools and plugins in Jenkins. For example, Git, Maven, Docker, etc.

Step 7: Set Up HTTPS (Optional but Recommended)

For added security, consider setting up HTTPS for your Jenkins instance. You can use a reverse proxy like Nginx or Apache with an SSL certificate.

Step 8: Integration with AWS Services (Optional)

Integrate Jenkins with AWS services for enhanced functionality. For example, you can configure Jenkins to deploy applications to AWS Elastic Beanstalk, AWS Lambda, or other AWS services.

How to use docker and everything about it

Docker is a platform that enables developers to automate the deployment of applications inside lightweight, portable containers. Containers are a form of virtualization that allows applications and their dependencies to be packaged together, ensuring consistency across different environments. Here's a comprehensive guide on Docker and its key concepts:

1. Docker Concepts:

a. ***Image:***

- An image is a lightweight, standalone, and executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, and system tools.

b. ***Container:***

- A container is a running instance of a Docker image. It provides a consistent and isolated environment for applications to run, ensuring that they behave the same way regardless of the underlying infrastructure.

c. ***Dockerfile:***

- A Dockerfile is a text file that contains instructions for building a Docker image. It specifies the base image, adds dependencies, copies files, and defines commands to run when the container starts.

d. ***Registry:***

- A Docker registry is a centralized repository for Docker images. Docker Hub is the default public registry, but you can also use private registries.

2. Installing Docker:

- Follow the official Docker installation guide for your operating system.

3. Basic Docker Commands:

a. **Pull an Image:**

```bash
docker pull image_name:tag
```

b. **Run a Container:**

```bash
docker run image_name:tag
```

c. **List Running Containers:**

```bash
docker ps
```

d. **List All Containers (including stopped ones):**

```bash
docker ps -a
```

e. **Stop a Running Container:**

```bash
docker stop container_id
```

f. **Remove a Container:**

```bash
docker rm container_id
```

g. **List Docker Images:**

```bash
docker images
```

h. **Remove an Image:**

```bash
docker rmi image_id
```

4. Creating and Running Docker Containers:

a. **Build an Image from a Dockerfile:**

```bash
docker build -t image_name:tag .
```

b. **Run a Container Interactively:**

```bash
docker run -it image_name:tag /bin/bash
```

c. **Copy Files into a Container:**

```bash
docker cp local_file_path container_id:/container_path
```

d. **Expose Ports:**

```bash
docker run -p host_port:container_port image_name:tag
```

e. **Mount Volumes:**

```bash
docker run -v /host/path:/container/path image_name:tag
```

5. Docker Compose:

- Docker Compose is a tool for defining and running multi-container Docker applications using a YAML file. It allows you to define a multi-container environment in a single file, making it easier to manage complex applications.

a. **Install Docker Compose:**

- Follow the official Docker Compose installation guide.

b. **Create a Docker Compose File (docker-compose.yml):**

```yaml
version: '3'
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
```

c. **Run Docker Compose:**

```bash
docker-compose up
```

d. **Stop Docker Compose:**

```bash
docker-compose down
```

6. Docker Networking:

- Docker provides different networking options for containers, including bridge networks, host networks, and user-defined networks.

7. Docker Swarm (Orchestration):

- Docker Swarm is a native clustering and orchestration solution for Docker. It allows you to create and manage a swarm of Docker nodes, turning them into a single virtual Docker host.

8. Docker Security:

- Docker provides various security features, including user namespaces, seccomp profiles, and capabilities, to isolate and secure containers.

9. Docker in CI/CD:

- Docker is commonly used in CI/CD pipelines to create reproducible build environments and to deploy applications consistently across different stages of the development lifecycle.

10. Additional Resources:

- Docker Documentation
- Docker Hub (Public registry for Docker images)
- Docker Tutorial

How to install docker on aws server

To install Docker on an AWS EC2 instance, you'll generally follow a process similar to installing Docker on any other Linux server. Below are the general steps to install Docker on an Amazon Linux 2 EC2 instance. Keep in mind that specific instructions might vary based on the Linux distribution you're using.

Step-by-Step Guide to Install Docker on AWS EC2:

1. **Connect to Your EC2 Instance:**
   Use an SSH client (e.g., PuTTY) to connect to your EC2 instance using the private key associated with your key pair.

```bash
ssh -i /path/to/your-key.pem ec2-user@your-instance-ip
```

2. **Update the Package Repositories:**
   Ensure your system is up to date.

```bash
sudo yum update -y
```

3. **Install Docker:**

   Install Docker using the package manager.

   ```bash
   sudo amazon-linux-extras install docker
   ```

   If you are not using Amazon Linux 2, you may need to use a different package manager. For example, on Ubuntu, you would use:

   ```bash
   sudo apt-get update
   sudo apt-get install docker.io
   ```

4. **Start the Docker Service:**

   Start the Docker service and enable it to start on boot.

   ```bash
   sudo service docker start
   sudo chkconfig docker on
   ```

If you are using a system with `systemd` (such as Amazon Linux 2), you can use the following commands:

```bash
sudo systemctl start docker
sudo systemctl enable docker
```

5. **Add the User to the Docker Group (Optional, for convenience):**

   Adding your user to the `docker` group allows you to run Docker commands without using `sudo`.

   ```bash
   sudo usermod -aG docker $USER
   ```

   Log out and log back in for the group changes to take effect.

6. **Verify Docker Installation:**

   Check if Docker is installed and running.

   ```bash
   docker --version
   docker info
   ```

   Run a simple test container to ensure everything is working:

   ```bash
   docker run hello-world
   ```

   If the installation is successful, you'll see a "Hello from Docker!" message.

Additional Considerations:

- **Security Group Configuration:**

  Ensure that your EC2 instance's security group allows incoming traffic on the necessary Docker ports. By default, Docker uses port 2375 for communication.

- **Install Docker Compose (Optional):**

  If you plan to use Docker Compose, you'll need to install it separately. Follow the instructions provided in the Docker Compose documentation.

- **Docker in Production:**

  For production environments, consider additional security measures, such as enabling Docker Content Trust, monitoring, and configuring appropriate access controls.

Kubernetes

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It helps in managing and running containerized applications in various environments. Here's a comprehensive guide from zero to hero for understanding Kubernetes:

**1. Introduction to Containers:

- **Containers:** Understand the concept of containers and how they differ from virtual machines.
- **Docker:** Learn the basics of Docker, a popular containerization platform.

2. Basics of Kubernetes:

- **Architecture:** Understand the key components of Kubernetes, including the Master and Worker nodes.
- **Cluster:** Set up a simple Kubernetes cluster using tools like `kubeadm`.

3. Pods and Containers:

- **Pods:** Learn about Pods, the smallest deployable units in Kubernetes.
- **Containers in Pods:** Explore how containers are defined and run within Pods.
- **ReplicaSets:** Understand ReplicaSets for scaling and managing identical copies of Pods.

4. Services and Networking:

- **Services:** Explore different types of services for exposing applications within the cluster.
- **Networking:** Understand how networking works in Kubernetes, including ClusterIP, NodePort, and LoadBalancer services.

5. Deployments and StatefulSets:

- **Deployments:** Learn about Deployments for managing the rollout and rollback of applications.
- **StatefulSets:** Explore StatefulSets for managing stateful applications with unique identities.

6. ConfigMaps and Secrets:

- **ConfigMaps:** Manage configuration data separately from application code.
- **Secrets:** Securely manage sensitive information such as passwords and API keys.

7. Persistent Volumes and Storage:

- **Persistent Volumes (PV) and Persistent Volume Claims (PVC):** Understand how to manage persistent storage in Kubernetes.
- **Storage Classes:** Explore different storage classes for dynamic provisioning of storage.

8. Helm Charts:

- **Helm:** Learn about Helm, a package manager for Kubernetes, and Helm charts for defining, installing, and upgrading even the most complex Kubernetes applications.

9. Ingress and Load Balancing:

- **Ingress:** Set up Ingress controllers to expose services to the external world.
- **Load Balancing:** Explore load balancing within the cluster.

10. Monitoring and Logging:

- **Monitoring:** Use tools like Prometheus for monitoring cluster and application health.
- **Logging:** Implement logging solutions like Fluentd or Elasticsearch for centralized log management.

11. Security:

- **RBAC:** Implement Role-Based Access Control for cluster security.
- **Network Policies:** Define and enforce network policies for controlling communication between Pods.

12. Advanced Concepts:

- **Custom Resource Definitions (CRDs):** Extend the Kubernetes API to create custom resources.
- **Operators:** Learn about Kubernetes operators for automating complex application management tasks.

13. Cloud-Native Ecosystem:

- **CI/CD Integration:** Integrate Kubernetes with CI/CD tools for automated application deployment.
- **Service Mesh:** Explore service mesh solutions like Istio for handling communication between services.

14. Troubleshooting:

- **Kubectl Commands:** Learn essential `kubectl` commands for troubleshooting and debugging.
- **Logs and Events:** Use logs and events to identify and diagnose issues within the cluster.

15. Continuous Learning:

- **Community and Resources:** Engage with the Kubernetes community, participate in forums, and follow updates from official documentation and blogs.

Additional Tips:

- **Playgrounds:** Use online Kubernetes playgrounds to experiment and practice without setting up a local cluster initially.
- **Certification:** Consider Kubernetes certifications like Certified Kubernetes Administrator (CKA) to validate your skills.

Installing Kubernetes

Installing Kubernetes involves setting up a cluster with a master node and one or more worker nodes. Below are the general steps for installing Kubernetes using `kubeadm`, which is a tool for bootstrapping a Kubernetes cluster.

Prerequisites: Before you begin, make sure you have the following:

1. **Minimum Requirements:** Ensure your machines meet the minimum requirements for running a Kubernetes cluster.
2. **Container Runtime:** Choose and install a container runtime like Docker on all nodes.
3. **Network Plugin:** Decide on a network plugin (e.g., Calico, Flannel) and install it on your cluster.

Installation Steps:

Step 1: Set Up Master Node

1. On the master node, disable swap:

bash Copy code

```
sudo swapoff -a
```

2. Install Docker:

bash Copy code

```
sudo apt-get update
sudo apt-get install -y docker.io
```

3. Install `kubeadm`, `kubelet`, and `kubectl`:

bash Copy code

```
sudo apt-get update && sudo apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo ap
echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
```

```
tall -y apt-transport-https curl
m/apt/doc/apt-key.gpg | sudo apt-key add -
rnetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

4. Initialize the cluster:

```bash
sudo kubeadm init --pod-network-cidr=CIDR
```

Replace `CIDR` with the address range for your chosen network plugin (e.g., `192.168.0.0/16` for Calico).

5. Set up the kubeconfig file:

```bash
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

6. Apply the network plugin. For example, for Calico:

```bash
kubectl apply -f https://docs.projectcalico.org/v3.18/manifests/calico.ya
```

Step 2: Join Worker Nodes

1. On each worker node, disable swap and install Docker as in Step 1.
2. Join the worker node to the cluster. Use the command provided by the `kubeadm init` output on the master node:

```bash
sudo kubeadm join <MASTER_IP>:<MASTER_PORT> --token <TOKEN> --discover
```

Replace `<MASTER_IP>`, `<MASTER_PORT>`, `<TOKEN>`, and `<HASH>` with the appropriate values from the `kubeadm init` output.

Step 3: Verify Cluster

On the master node, check the status of the nodes:

```bash
kubectl get nodes
```

The nodes should be in the `Ready` state.

```
:<MASTER_PORT> --token <TOKEN> --discovery-token-ca-cert-hash <HASH>
```

Uses of Kubernetes

Once you have Kubernetes installed, you can start using it to deploy and manage containerized applications. Here's a basic guide on how to use Kubernetes:

1. Deploying Your First Application:

a. Deploying a Simple Pod:

A Pod is the smallest deployable unit in Kubernetes. Create a YAML file for a simple Pod:

yaml Copy code

```yaml
# pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: mycontainer
      image: nginx
```

Deploy the Pod:

bash Copy code

```bash
kubectl apply -f pod.yaml
```

Check the status of the Pod:

bash Copy code

```bash
kubectl get pods
```

b. Deploying a Deployment:

Deployments provide a declarative way to manage applications. Create a YAML file for a simple Deployment:

yaml Copy code

```yaml
# deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: mycontainer
          image: nginx
```

Deploy the Deployment:

bash Copy code

```bash
kubectl apply -f deployment.yaml
```

Check the status of the Deployment and Pods:

bash Copy code

```bash
kubectl get deployments
kubectl get pods
```

2. Scaling Your Application:

Scale the Deployment to have more replicas:

bash Copy code

```
kubectl scale deployment mydeployment --replicas=5
```

Check the updated status:

bash Copy code

```
kubectl get deployments
kubectl get pods
```

3. Updating Your Application:

Update the image version in the Deployment:

yaml Copy code

```yaml
# deployment-update.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: mycontainer
          image: nginx:1.17
```

Apply the update:

```bash
kubectl apply -f deployment-update.yaml
```

Check the rolling update progress:

```bash
kubectl get deployments
kubectl get pods
```

4. Exposing Your Application:

Expose the Deployment using a Service:

```yaml
# service.yaml
apiVersion: v1
kind: Service
metadata:
  name: myservice
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Apply the Service:

```bash
kubectl apply -f service.yaml
```

Check the external IP:

```bash
kubectl get services
```

Access your application using the external IP.

5. Cleaning Up:

Delete the resources:

```bash
kubectl delete -f deployment.yaml
kubectl delete -f service.yaml
kubectl delete -f pod.yaml
```

6. Other Useful Commands:

- **Viewing Logs:**

```bash
kubectl logs <pod_name>
```

- **Executing Commands in a Pod:**

```bash
kubectl exec -it <pod_name> -- /bin/bash
```

- **Inspecting Resources:**

```bash
kubectl describe <resource_type> <resource_name>
```

- **Monitoring Resources:**

```bash
kubectl top <resource_type> <resource_name>
```

Example of Kubernetes

simple example of deploying an Nginx web server using Kubernetes. We'll create a Deployment, expose it with a Service, and then access the deployed application.

1. Deploying an Nginx Deployment:

Create a file named `nginx-deployment.yaml` with the following content:

yaml Copy code

```yaml
# nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx-container
          image: nginx:latest
          ports:
            - containerPort: 80
```

Apply the deployment:

```bash
kubectl apply -f nginx-deployment.yaml
```

Check the deployment status:

```bash
kubectl get deployments
kubectl get pods
```

2. Exposing the Nginx Deployment with a Service:

Create a file named `nginx-service.yaml` with the following content:

```yaml
# nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Apply the service:

```bash
kubectl apply -f nginx-service.yaml
```

Check the service and get the external IP:

```bash
kubectl get services
```

Wait for the external IP to be assigned. Once available, you can access the Nginx web server using this IP.

3. Accessing the Nginx Application:

Open a web browser and navigate to the external IP obtained from the service. You should see the default Nginx welcome page.

4. Scaling the Nginx Deployment:

Scale the deployment to have more replicas:

```bash
kubectl scale deployment nginx-deployment --replicas=5
```

Check the updated deployment status:

```bash
kubectl get deployments
kubectl get pods
```

5. Cleaning Up:

Delete the deployed resources:

```bash
kubectl delete -f nginx-service.yaml
kubectl delete -f nginx-deployment.yaml
```

PHPUnit Test for wordpress

PHPUnit is a testing framework for PHP, and it can be used for testing WordPress plugins, themes, and core functionalities. Below, I'll guide you through the process of setting up and writing PHPUnit tests for WordPress.

1. Setup PHPUnit for WordPress:

1. **Install PHPUnit:**
   - Install PHPUnit globally using Composer:

```bash
composer global require "phpunit/phpunit:^9"
```

2. **Create a `phpunit.xml` file:**
   - Create a `phpunit.xml` file in the root of your WordPress plugin/theme directory. Here is a minimal example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="tests/bootstrap.php">
    <testsuites>
        <testsuite name="My Plugin Tests">
            <directory>tests</directory>
        </testsuite>
    </testsuites>
</phpunit>
```

3. **Create a `tests` directory:**
   - Inside your plugin/theme directory, create a `tests` directory. This is where your test files will reside.
4. **Create a `bootstrap.php` file:**
   - Create a `bootstrap.php` file inside the `tests` directory. This file will include the necessary WordPress files for testing.

2. Writing PHPUnit Tests for WordPress:

1. **Basic Test Structure:**
   - Create a test class for your plugin/theme in the `tests` directory. Here is a simple example:

```php
<?php
class MyPluginTest extends WP_UnitTestCase {
    // Your test methods go here
}
```

2. **Use WordPress Test Functions:**
   - PHPUnit test classes for WordPress should extend `WP_UnitTestCase`. Use WordPress-specific testing functions such as `factory()`, `go_to()`, and others provided by the testing environment.

3. **Example Test:**
   - Here's an example of a basic test method:

```php
<?php
class MyPluginTest extends WP_UnitTestCase {
    public function test_example() {
        // Arrange
        $value = 2;

        // Act
        $result = $value * 2;

        // Assert
        $this->assertEquals(4, $result);
    }
}
```

3. Running PHPUnit Tests:

1. **Run Tests:**
   - Navigate to your plugin/theme directory in the terminal and run PHPUnit:

```bash
phpunit
```

   - PHPUnit will read the `phpunit.xml` file and execute the tests.

4. WordPress Database Interaction:

1. **Use `setUp` and `tearDown` Methods:**
   - Use `setUp` to set up the testing environment before each test, and `tearDown` to clean up after each test. This includes actions like creating test data in the WordPress database.

```php
<?php
class MyPluginTest extends WP_UnitTestCase {
    public function setUp() {
        parent::setUp();
        // Set up test environment
    }

    public function tearDown() {
        // Clean up test environment
        parent::tearDown();
    }

    // Your test methods go here
}
```

2. **Example Database Interaction Test:**
   - Here's an example test that interacts with the WordPress database:

```php
<?php
class MyPluginTest extends WP_UnitTestCase {
    public function test_database_interaction() {
        // Arrange
        $post_id = $this->factory->post->create();

        // Act
        $post = get_post($post_id);

        // Assert
        $this->assertInstanceOf('WP_Post', $post);
    }
}
```

5. Additional Tips:

1. **Useful Assertions:**
   - Explore PHPUnit's various assertions like `assertTrue`, `assertFalse`, `assertNotEmpty`, etc. PHPUnit Assertions
2. **WordPress Testing Resources:**
   - Refer to the official WordPress Automated Testing handbook for more details on WordPress testing.
3. **Mocking:**
   - PHPUnit supports mocking to isolate your tests from external dependencies. Use `getMockBuilder` and related methods for mocking.
4. **Continuous Integration (CI):**
   - Integrate your PHPUnit tests into a CI/CD pipeline for automatic testing on each commit.

Composer in PHP

Composer is a dependency manager for PHP, which simplifies the process of managing libraries and packages in your PHP projects. It allows you to declare the libraries your project depends on and manages them for you. Below, I'll provide detailed information about Composer, covering its installation, usage, and key concepts.

1. Installation:

Composer can be installed globally on your machine. The recommended way is to install it using the following commands:

bash Copy code

```
# Download the installer
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php')

# Verify the installer
php -r "if (hash_file('sha384', 'composer-setup.php') === 'EXPECTED_HAS

# Run the installer
php composer-setup.php

# Remove the installer
php -r "unlink('composer-setup.php');"

# Move composer to a directory in your PATH
mv composer.phar /usr/local/bin/composer
```

Replace `'EXPECTED_HASH'` with the correct hash value, which you can find on the Composer download page.

```
=== 'EXPECTED_HASH') { echo 'Installer verified'; } else { echo 'Installe
```

```
echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
```

2. Basic Usage:

Creating a New Project:

To create a new project, navigate to the project directory and run:

```bash
composer init
```

This command will prompt you to answer a series of questions about your project and create a `composer.json` file.

Installing Dependencies:

To install dependencies defined in the `composer.json` file, run:

```bash
composer install
```

This will download and install the required packages.

Autoloading:

Composer generates an autoloader for your project, allowing you to autoload classes without manually including them. To use autoloading, include the generated autoloader in your PHP files:

```php
require 'vendor/autoload.php';
```

3. Key Concepts:

`composer.json`:

The `composer.json` file is the heart of Composer. It defines your project's dependencies, scripts, and other metadata. Here's a basic example:

```
{
    "name": "your/vendor-name/your-project-name",
    "description": "Your project description",
    "type": "project",
    "authors": [
        {
            "name": "Your Name",
            "email": "your.email@example.com"
        }
    ],
    "require": {
        "vendor/package": "^1.0"
    }
}
```

`composer.lock`:

The **`composer.lock`** file stores exact versions of dependencies and ensures that everyone working on the project uses the same versions. It is recommended to commit this file to version control.

`vendor/` Directory:

The **`vendor/`** directory is where Composer installs the dependencies. You should not manually edit or commit this directory.

4. Dependency Management:

Installing Dependencies:

To install a new dependency, you can run:

```bash
composer require vendor/package
```

This command adds the package to your `composer.json` file and installs it.

Updating Dependencies:

To update your dependencies to the latest versions allowed by your `composer.json` file, run:

```bash
composer update
```

Specifying Versions:

You can specify versions in the `composer.json` file. For example:

```json
{
    "require": {
        "vendor/package": "1.2.*"
    }
}
```

5. Scripts:

Composer allows you to define custom scripts in the `composer.json` file. For example, you can define scripts for running tests, code linting, or any other tasks:

```json
{
    "scripts": {
        "test": "phpunit"
    }
}
```

You can then run the script using:

```bash
composer test
```

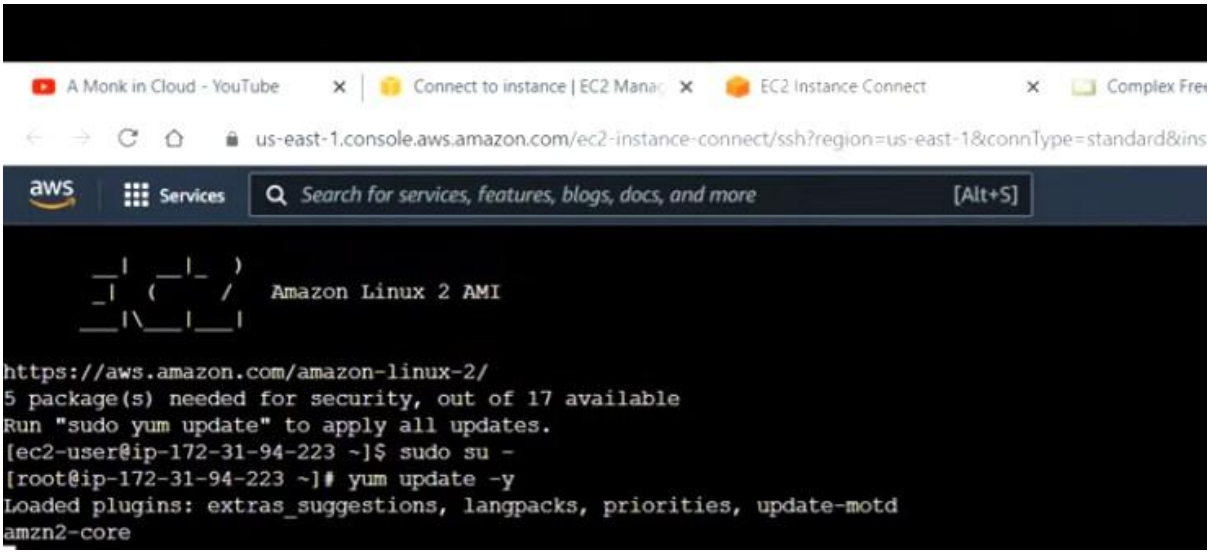
6. Creating Packages:

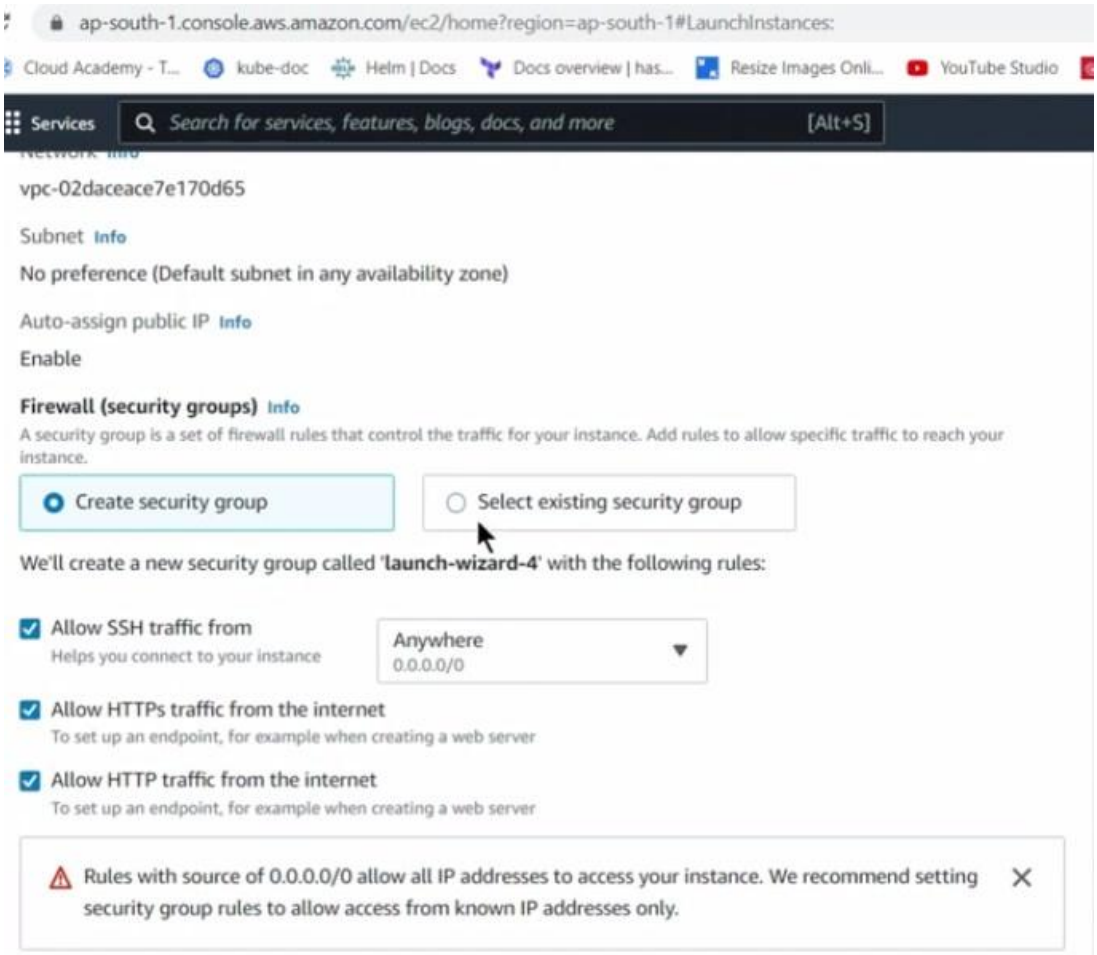
Composer allows you to create and distribute your own packages. You can publish them on Packagist, which is the default package repository for Composer.

7. Resources:

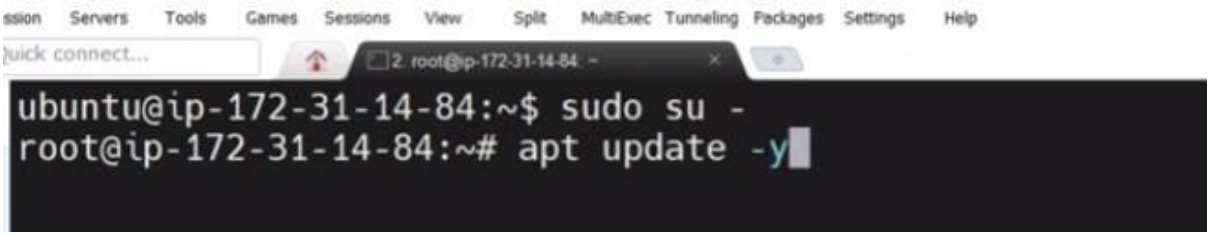
- Composer Documentation: The official documentation is extensive and covers every aspect of Composer.
- Packagist: The main package repository for Composer. You can search for packages or publish your own.

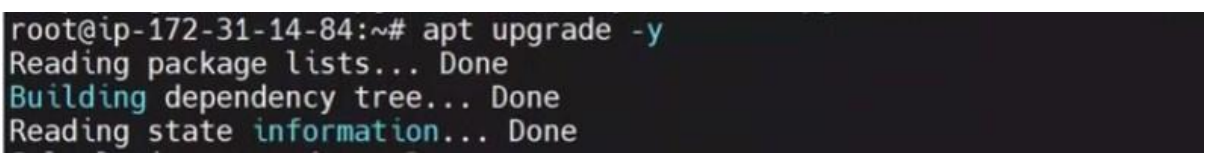
Wordpress Installation on AWS




ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:
Cloud Academy - T...
kube-doc
Helm | Docs
Docs overview | has...
Resize Images Onli...
YouTube Studio
Services
Search for services, features, blogs, docs, and more
[Alt+S]
vpc-02daceace7e170d65
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
We'll create a new security group called 'launch-wizard-4' with the following rules:
Allow SSH traffic from
Helps you connect to your instance
Anywhere
0.0.0.0/0
Allow HTTPs traffic from the internet
To set up an endpoint, for example when creating a web server
Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.


Quick connect...
2. /drives/c/Alok/Key
23/10/2022 08:59.37 /drives/c/Alok/Key master ls
Devops-CB.pem
23/10/2022 09:06.39 /drives/c/Alok/Key master chmod 400 Devops-CB.pem
23/10/2022 09:07.01 /drives/c/Alok/Key master ssh ubuntu@3.110.153.117 -i Devops-CB.pem


ssion Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. root@ip-172-31-14-84: ~
ubuntu@ip-172-31-14-84:~$ sudo su -
root@ip-172-31-14-84:~# apt update -y


root@ip-172-31-14-84:~# apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

```
root@ip-172-31-14-84:~#
root@ip-172-31-14-84:~# apt install apache2 -y
Reading package lists... Done
```

```
root@ip-172-31-14-84:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor
     Active: active (running) since Sun 2022-10-23 03:40:37 UTC; 28s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 14153 (apache2)
      Tasks: 55 (limit: 1143)
     Memory: 5.0M
        CPU: 33ms
     CGroup: /system.slice/apache2.service
             ├─14153 /usr/sbin/apache2 -k start
             ├─14155 /usr/sbin/apache2 -k start
             └─14156 /usr/sbin/apache2 -k start
```

```
root@ip-172-31-14-84:~# systemctl enable apache2
Synchronizing state of apache2.service with SysV service scr
Executing: /lib/systemd/systemd-sysv-install enable apache2
```

```
root@ip-172-31-14-84:~# apt install mariadb-server mariadb-client -y
```

```
root@ip-172-31-14-84:~# systemctl start mariadb
root@ip-172-31-14-84:~# systemctl status mariadb
```

```
ubuntu@ip-172-31-14-84:~$ sudo su -
root@ip-172-31-14-84:~# mysql_secure_installation
```

```
Change the root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
```

```
Disallow root login remotely? [Y/n] Y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

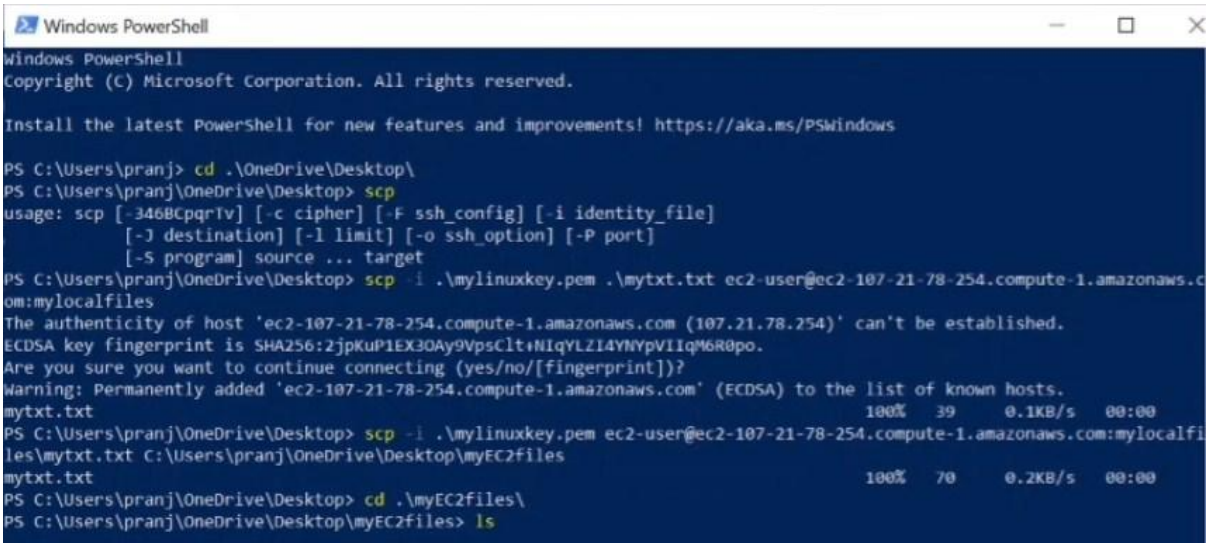
```
Thanks for using MariaDB!
root@ip-172-31-14-84:~# systemctl restart mariadb
root@ip-172-31-14-84:~# apt install php php-mysql php-gd php-cli php-common -y
```

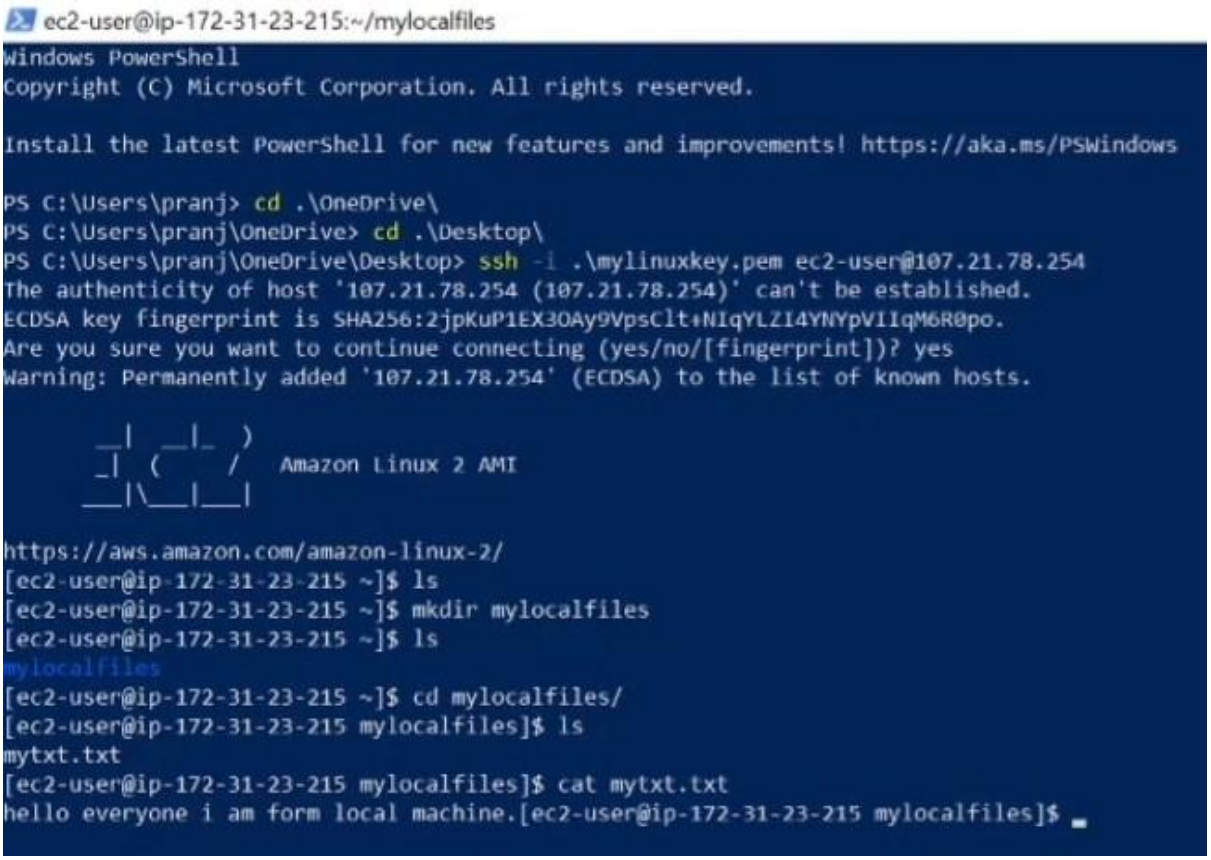
```
root@ip-172-31-14-84:~# apt install wget unzip -y
```

```
root@ip-172-31-14-84:~# ls
latest.zip  snap
root@ip-172-31-14-84:~# unzip latest.zip
```

```
root@ip-172-31-14-84:~# ls
latest.zip  snap  wordpress
root@ip-172-31-14-84:~# cd wordpress/
root@ip-172-31-14-84:~/wordpress# ls
index.php        wp-admin             wp-content         wp-load.ph
license.txt      wp-blog-header.php   wp-cron.php        wp-login.p
readme.html      wp-comments-post.php wp-includes        wp-mail.ph
wp-activate.php  wp-config-sample.php wp-links-opml.php  wp-setting
root@ip-172-31-14-84:~/wordpress# cd ..
root@ip-172-31-14-84:~# ls
latest.zip  snap  wordpress
root@ip-172-31-14-84:~# pwd
/root
root@ip-172-31-14-84:~# cp -r wordpress/* /var/www/html/
root@ip-172-31-14-84:~# cd /var/www/html/
root@ip-172-31-14-84:/var/www/html# ls -
```

```
root@ip-172-31-14-84:/var/www/html# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.7-MariaDB-2ubuntu1.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database wordpress;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> create user "wpadmin" identified by "wpadminpass";
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> grant all privileges on wordpress.* to "wpadmin";
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]>
```


Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\pranj> cd .\OneDrive\Desktop\
PS C:\Users\pranj\OneDrive\Desktop> scp
usage: scp [-346BCpqrTv] [-c cipher] [-F ssh_config] [-i identity_file]
[-J destination] [-l limit] [-o ssh_option] [-P port]
[-S program] source ... target
PS C:\Users\pranj\OneDrive\Desktop> scp -i .\mylinuxkey.pem .\mytxt.txt ec2-user@ec2-107-21-78-254.compute-1.amazonaws.com:mylocalfiles
The authenticity of host 'ec2-107-21-78-254.compute-1.amazonaws.com (107.21.78.254)' can't be established.
ECDSA key fingerprint is SHA256:2jpKuP1EX3OAy9VpsClt+NIqYLZI4YNYpVIIqM6R0po.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added 'ec2-107-21-78-254.compute-1.amazonaws.com' (ECDSA) to the list of known hosts.
mytxt.txt 100% 39 0.1KB/s 00:00
PS C:\Users\pranj\OneDrive\Desktop> scp -i .\mylinuxkey.pem ec2-user@ec2-107-21-78-254.compute-1.amazonaws.com:mylocalfiles\mytxt.txt C:\Users\pranj\OneDrive\Desktop\myEC2files
mytxt.txt 100% 70 0.2KB/s 00:00
PS C:\Users\pranj\OneDrive\Desktop> cd .\myEC2files\
PS C:\Users\pranj\OneDrive\Desktop\myEC2files> ls


ec2-user@ip-172-31-23-215:~/mylocalfiles
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\pranj> cd .\OneDrive\
PS C:\Users\pranj\OneDrive> cd .\Desktop\
PS C:\Users\pranj\OneDrive\Desktop> ssh -i .\mylinuxkey.pem ec2-user@107.21.78.254
The authenticity of host '107.21.78.254 (107.21.78.254)' can't be established.
ECDSA key fingerprint is SHA256:2jpKuP1EX3OAy9VpsClt+NIqYLZI4YNYpVIIqM6R0po.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '107.21.78.254' (ECDSA) to the list of known hosts.
Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-23-215 ~]$ ls
[ec2-user@ip-172-31-23-215 ~]$ mkdir mylocalfiles
[ec2-user@ip-172-31-23-215 ~]$ ls
mylocalfiles
[ec2-user@ip-172-31-23-215 ~]$ cd mylocalfiles/
[ec2-user@ip-172-31-23-215 mylocalfiles]$ ls
mytxt.txt
[ec2-user@ip-172-31-23-215 mylocalfiles]$ cat mytxt.txt
hello everyone i am form local machine.[ec2-user@ip-172-31-23-215 mylocalfiles]$

Wordpres installation on AWS

1. **Create an AWS Account:** If you don't already have an AWS account, sign up for one at https://aws.amazon.com/. You will need to provide billing information, but you can use the AWS Free Tier to get started without incurring charges for the first 12 months.
2. **Launch an EC2 Instance:**
    - Sign in to your AWS Management Console.
    - Navigate to the EC2 Dashboard.
    - Click the "Launch Instance" button to create a new virtual server.
    - Choose an Amazon Machine Image (AMI) based on your requirements. You can use an Amazon Linux AMI, which is suitable for most purposes.
    - Select an instance type, which determines the server's hardware resources. The t2.micro instance is part of the Free Tier.
    - Configure instance details, such as the number of instances, network settings, and storage.
    - Add storage for your instance. You can use the default settings or adjust the storage size as needed.
    - Configure security groups to control incoming traffic to your instance. At a minimum, allow SSH (port 22) and HTTP/HTTPS (ports 80 and 443).
    - Review and launch the instance, and create a new key pair or use an existing one to securely connect to your instance.
3. **Connect to Your EC2 Instance:**
    - Once the instance is running, use SSH to connect to it. Use the private key from the key pair you selected during instance launch.
    - The command to connect will look like this:

```css
ssh -i /path/to/your-key.pem ec2-user@your-instance-ip
```

4. **Install LAMP Stack:**
   - After connecting to your EC2 instance, you can install the LAMP (Linux, Apache, MySQL, PHP) stack to host WordPress.
   - Run the following commands to install Apache, MySQL, and PHP:

```bash
sudo yum install httpd -y
sudo systemctl start httpd
sudo systemctl enable httpd
sudo yum install mysql-server -y
sudo systemctl start mysqld
sudo systemctl enable mysqld
sudo yum install php php-mysql -y
```

5. **Secure MySQL:**
   - Run the MySQL secure installation script to set a root password and secure your MySQL installation.

```
sudo mysql_secure_installation
```

6. **Download and Configure WordPress:**
   - Download the latest WordPress release and extract it in the Apache web server's document root:

```bash
cd /var/www/html
sudo wget https://wordpress.org/latest.tar.gz
sudo tar -xzf latest.tar.gz
```

6. **Download and Configure WordPress:**
   - Download the latest WordPress release and extract it in the Apache web server's document root:

```bash
cd /var/www/html
sudo wget https://wordpress.org/latest.tar.gz
sudo tar -xzf latest.tar.gz
```

7. **Create a MySQL Database for WordPress:**
   - Log in to MySQL and create a new database and user for WordPress.

```css
mysql -u root -p
```

   Enter your MySQL root password, then run the following SQL commands:

```sql
CREATE DATABASE wordpress;
CREATE USER 'wordpressuser'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpressuser'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

8. **Configure WordPress:**
   - Rename the `wp-config-sample.php` file to `wp-config.php` and edit it to provide your database details:

```arduino
cd /var/www/html/wordpress
sudo mv wp-config-sample.php wp-config.php
sudo nano wp-config.php
```

   Update the database name, user, and password you created earlier.

9. **Access WordPress Installation:**
    - Open a web browser and enter your instance's public IP address or domain name. You should see the WordPress installation wizard.
    - Complete the setup by providing your site title, username, password, and email address.
10. **Complete WordPress Setup:**
    - After the setup, you can log in to your WordPress admin dashboard and start customizing your site.
11. **Additional Configuration:**
    - You may want to configure your domain name, set up SSL/TLS certificates, and enhance security. AWS offers services like Route 53 and ACM for domain management and SSL certificates.
12. **Regular Backups and Updates:**
    - Ensure you regularly back up your WordPress site and keep both your WordPress and server software up to date to maintain security and performance.

This is a basic guide to installing WordPress on AWS. Depending on your specific requirements, you may need to configure additional services, such as an S3 bucket for media storage or CloudFront for content delivery. Always refer to the AWS documentation for the most up-to-date information and best practices.

Reg

LAMP Setup

https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-20-04

Step 1 – Installing Apache and Updating the Firewall

The Apache web server is among the most popular web servers in the world. It's well documented, has an active community of users, and has been in wide use for much of the history of the web, which makes it a great choice for hosting a website.

Start by updating the package manager cache. If this is the first time you're using `sudo` within this session, you'll be prompted to provide your user's password to confirm you have the right privileges to manage system packages with `apt`.

```
$ sudo apt update
```
Copy

Then, install Apache with:

```
$ sudo apt install apache2
```
Copy

You'll also be prompted to confirm Apache's installation by pressing `Y`, then `ENTER`.

Once the installation is finished, you'll need to adjust your firewall settings to allow HTTP traffic. UFW has different application profiles that you can leverage for accomplishing that. To list all currently available UFW application profiles, you can run:

```
$ sudo ufw app list
```
Copy

You'll see output like this:

```
Output
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
```

Here's what each of these profiles mean:

- **Apache**: This profile opens only port `80` (normal, unencrypted web traffic).
- **Apache Full**: This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic).
- **Apache Secure**: This profile opens only port `443` (TLS/SSL encrypted traffic).

For now, it's best to allow only connections on port `80`, since this is a fresh Apache installation and you still don't have a TLS/SSL certificate configured to allow for HTTPS traffic on your server.

To only allow traffic on port `80`, use the `Apache` profile:

```
$ sudo ufw allow in "Apache"
```

Copy

You can verify the change with:

```
$ sudo ufw status
```

Copy

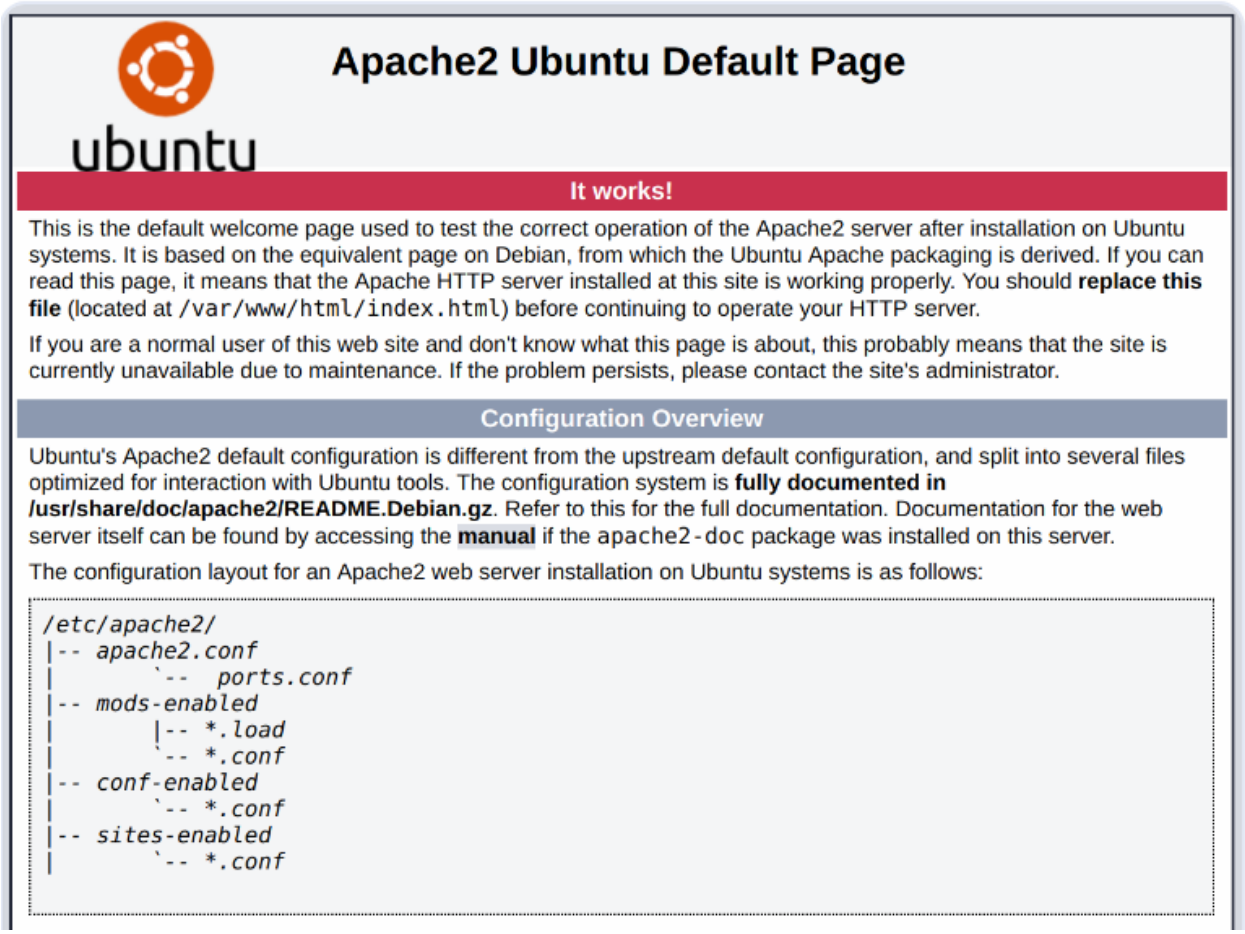
```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Apache                     ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Apache (v6)                ALLOW       Anywhere (v6)
```

Traffic on port `80` is now allowed through the firewall.

You can do a spot check right away to verify that everything went as planned by visiting your server's public IP address in your web browser (see the note under the next heading to find out what your public IP address is if you do not have this information already):

```
http://your_server_ip
```

You'll see the default Ubuntu 20.04 Apache web page, which is there for informational and testing purposes. It should look something like this:



Apache2 Ubuntu Default Page

ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod, a2dismod, a2ensite, a2dissite,` and `a2enconf, a2disconf` . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file apart of those located in `/var/www`, **public_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under /var/www. This is different to previous releases which provides better security out of the box.

How To Find your Server's Public IP Address

If you do not know what your server's public IP address is, there are a number of ways you can find it. Usually, this is the address you use to connect to your server through SSH.

There are a few different ways to do this from the command line. First, you could use the `iproute2` tools to get your IP address by typing this:

```
$ ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
```

Copy

This will give you two or three lines back. They are all correct addresses, but your computer may only be able to use one of them, so feel free to try each one.

An alternative method is to use the `curl` utility to contact an outside party to tell you how *it* sees your server. This is done by asking a specific server what your IP address is:

```
$ curl http://icanhazip.com
```

Copy

Regardless of the method you use to get your IP address, type it into your web browser's address bar to view the default Apache page.

Step 2 – Installing MySQL

Now that you have a web server up and running, you need to install the database system to be able to store and manage data for your site. MySQL is a popular database management system used within PHP environments.

Again, use `apt` to acquire and install this software:

```
$ sudo apt install mysql-server
```

Copy

When prompted, confirm installation by typing `Y`, and then `ENTER`.

When the installation is finished, it's recommended that you run a security script that comes pre-installed with MySQL. This script will remove some insecure default settings and lock down access to your database system. Start the interactive script by running:

```
$ sudo mysql_secure_installation
```

Copy

This will ask if you want to configure the `VALIDATE PASSWORD PLUGIN`.

Note: Enabling this feature is something of a judgment call. If enabled, passwords which don't match the specified criteria will be rejected by MySQL with an error. It is safe to leave validation disabled, but you should always use strong, unique passwords for database credentials.

Answer `Y` for yes, or anything else to continue without enabling.

```
VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No:
```

If you answer "yes", you'll be asked to select a level of password validation. Keep in mind that if you enter `2` for the strongest level, you will receive errors when attempting to set any password which does not contain numbers, upper and lowercase letters, and special characters, or which is based on common dictionary words.

```
There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary                 file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
```

Regardless of whether you chose to set up the `VALIDATE PASSWORD PLUGIN`, your server will next ask you to select and confirm a password for the MySQL **root** user. This is not to be confused with the **system root**. The **database root** user is an administrative user with full privileges over the database system. Even though the default authentication method for the MySQL root user dispenses the use of a password, **even when one is set**, you should define a strong password here as an additional safety measure. We'll talk about this in a moment.

If you enabled password validation, you'll be shown the password strength for the root password you just entered and your server will ask if you want to continue with that password. If you are happy with your current password, enter `Y` for "yes" at the prompt:

```
Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y
```

For the rest of the questions, press `Y` and hit the `ENTER` key at each prompt. This will remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes you have made.

When you're finished, test if you're able to log in to the MySQL console by typing:

```
$ sudo mysql
```
Copy

This will connect to the MySQL server as the administrative database user **root**, which is inferred by the use of `sudo` when running this command. You should see output like this:

```
Output
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 8.0.19-0ubuntu5 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

To exit the MySQL console, type:

```
mysql> exit
```
Copy

Notice that you didn't need to provide a password to connect as the **root** user, even though you have defined one when running the `mysql_secure_installation` script. That is because the default authentication method for the administrative MySQL user is `unix_socket` instead of `password`. Even though this might look like a security concern at first, it makes the database server more secure because the only users allowed to log in as the **root** MySQL user are the system users with sudo privileges connecting from the console or through an application running with the same privileges. In practical terms, that means you won't be able to use the administrative database **root** user to connect from your PHP application. Setting a password for the **root** MySQL account works as a safeguard, in case the default authentication method is changed from `unix_socket` to `password`.

For increased security, it's best to have dedicated user accounts with less expansive privileges set up for every database, especially if you plan on having multiple databases hosted on your server.

Step 3 – Installing PHP

You have Apache installed to serve your content and MySQL installed to store and manage your data. PHP is the component of our setup that will process code to display dynamic content to the final user. In addition to the `php` package, you'll need `php-mysql`, a PHP module that allows PHP to communicate with MySQL-based databases. You'll also need `libapache2-mod-php` to enable Apache to handle PHP files. Core PHP packages will automatically be installed as dependencies.

To install these packages, run:

```
$ sudo apt install php libapache2-mod-php php-mysql
```

Copy

Once the installation is finished, you can run the following command to confirm your PHP version:

```
$ php -v
```

Copy

```
Output
PHP 7.4.3 (cli) (built: Jul  5 2021 15:13:35) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies
```

At this point, your LAMP stack is fully operational, but before you can test your setup with a PHP script, it's best to set up a proper Apache Virtual Host to hold your website's files and folders. We'll do that in the next step.

Step 4 – Creating a Virtual Host for your Website

When using the Apache web server, you can create *virtual hosts* (similar to server blocks in Nginx) to encapsulate configuration details and host more than one domain from a single server. In this guide, we'll set up a domain called **your_domain**, but you should **replace this with your own domain name**.

> **Note**: In case you are using DigitalOcean as DNS hosting provider, you can check our product docs for detailed instructions on how to set up a new domain name and point it to your server.

Apache on Ubuntu 20.04 has one server block enabled by default that is configured to serve documents from the `/var/www/html` directory. While this works well for a single site, it can become unwieldy if you are hosting multiple sites. Instead of modifying `/var/www/html`, we'll create a directory structure within `/var/www` for the **your_domain** site, leaving `/var/www/html` in place as the default directory to be served if a client request doesn't match any other sites.

Create the directory for **your_domain** as follows:

```
$ sudo mkdir /var/www/your_domain
```
Copy

Next, assign ownership of the directory with the `$USER` environment variable, which will reference your current system user:

```
$ sudo chown -R $USER:$USER /var/www/your_domain
```
Copy

Then, open a new configuration file in Apache's `sites-available` directory using your preferred command-line editor. Here, we'll use `nano`:

```
$ sudo nano /etc/apache2/sites-available/your_domain.conf
```
Copy

This will create a new blank file. Paste in the following bare-bones configuration:

/etc/apache2/sites-available/your_domain.conf

```
<VirtualHost *:80>
    ServerName your_domain
    ServerAlias www.your_domain
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/your_domain
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Save and close the file when you're done. If you're using `nano`, you can do that by pressing `CTRL+X`, then `Y` and `ENTER`.

With this `VirtualHost` configuration, we're telling Apache to serve `your_domain` using `/var/www/your_domain` as the web root directory. If you'd like to test Apache without a domain name, you can remove or comment out the options `ServerName` and `ServerAlias` by adding a `#` character in the beginning of each option's lines.

You can now use `a2ensite` to enable the new virtual host:

```
$ sudo a2ensite your_domain
```
Copy

You might want to disable the default website that comes installed with Apache. This is required if you're not using a custom domain name, because in this case Apache's default configuration would overwrite your virtual host. To disable Apache's default website, type:

```
$ sudo a2dissite 000-default
```
Copy

To make sure your configuration file doesn't contain syntax errors, run:

```
$ sudo apache2ctl configtest
```
Copy

Finally, reload Apache so these changes take effect:

```
$ sudo systemctl reload apache2
```
Copy

Your new website is now active, but the web root `/var/www/your_domain` is still empty. Create an `index.html` file in that location so that we can test that the virtual host works as expected:

```
$ nano /var/www/your_domain/index.html
```
Copy

Include the following content in this file:

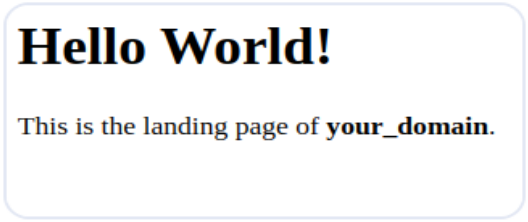
/var/www/your_domain/index.html

```
<html>
  <head>
    <title>your_domain website</title>
  </head>
  <body>
    <h1>Hello World!</h1>

    <p>This is the landing page of <strong>your_domain</strong>.</p>
  </body>
</html>
```

Now go to your browser and access your server's domain name or IP address once again:

```
http://server_domain_or_IP
```

You'll see a page like this:



If you see this page, it means your Apache virtual host is working as expected.

You can leave this file in place as a temporary landing page for your application until you set up an `index.php` file to replace it. Once you do that, remember to remove or rename the `index.html` file from your document root, as it would take precedence over an `index.php` file by default.

A Note About `DirectoryIndex` on Apache

With the default `DirectoryIndex` settings on Apache, a file named `index.html` will always take precedence over an `index.php` file. This is useful for setting up maintenance pages in PHP applications, by creating a temporary `index.html` file containing an informative message to visitors. Because this page will take precedence over the `index.php` page, it will then become the landing page for the application. Once maintenance is over, the `index.html` is renamed or removed from the document root, bringing back the regular application page.

In case you want to change this behavior, you'll need to edit the `/etc/apache2/mods-enabled/dir.conf` file and modify the order in which the `index.php` file is listed within the `DirectoryIndex` directive:

```
$ sudo nano /etc/apache2/mods-enabled/dir.conf
```
Copy

/etc/apache2/mods-enabled/dir.conf

```
<IfModule mod_dir.c>
        DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
</IfModule>
```

After saving and closing the file, you'll need to reload Apache so the changes take effect:

```
$ sudo systemctl reload apache2
```
Copy

In the next step, we'll create a PHP script to test that PHP is correctly installed and configured on your server.

Step 5 – Testing PHP Processing on your Web Server

Now that you have a custom location to host your website's files and folders, we'll create a PHP test script to confirm that Apache is able to handle and process requests for PHP files.

Create a new file named `info.php` inside your custom web root folder:

```
$ nano /var/www/your_domain/info.php
```

Copy

This will open a blank file. Add the following text, which is valid PHP code, inside the file:

/var/www/your_domain/info.php

```
<?php
phpinfo();
```

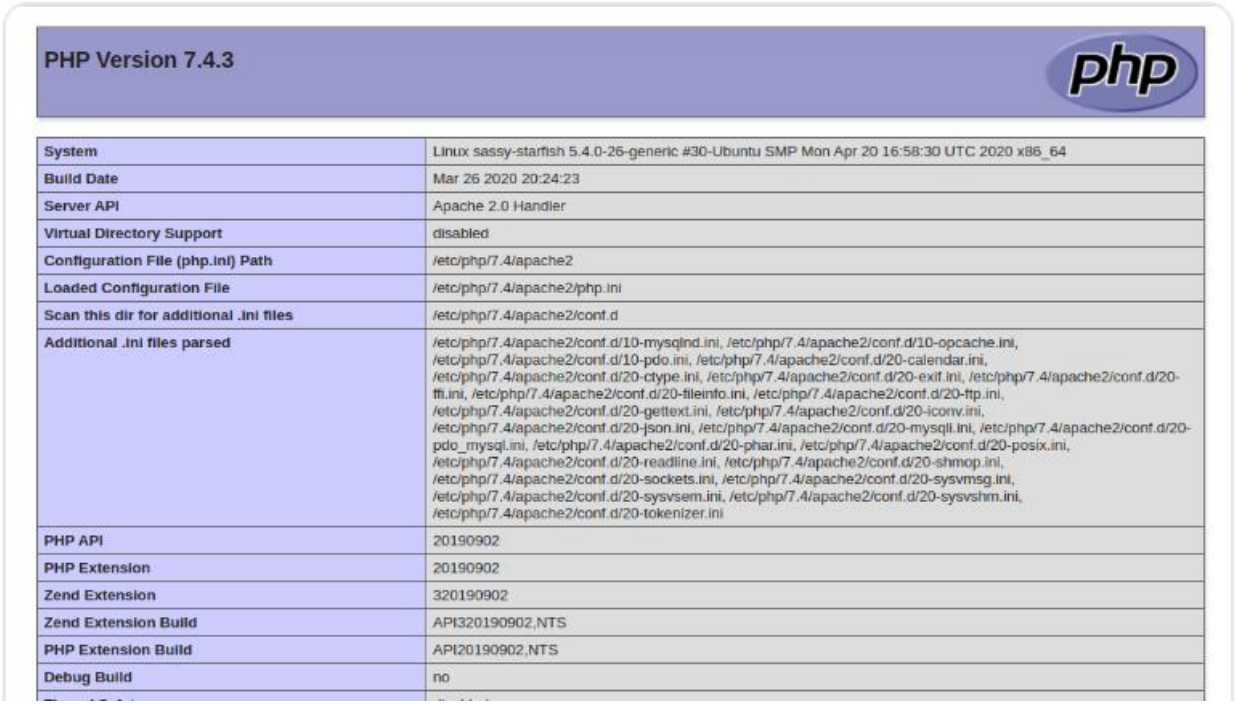
Copy

When you are finished, save and close the file.

To test this script, go to your web browser and access your server's domain name or IP address, followed by the script name, which in this case is `info.php`:

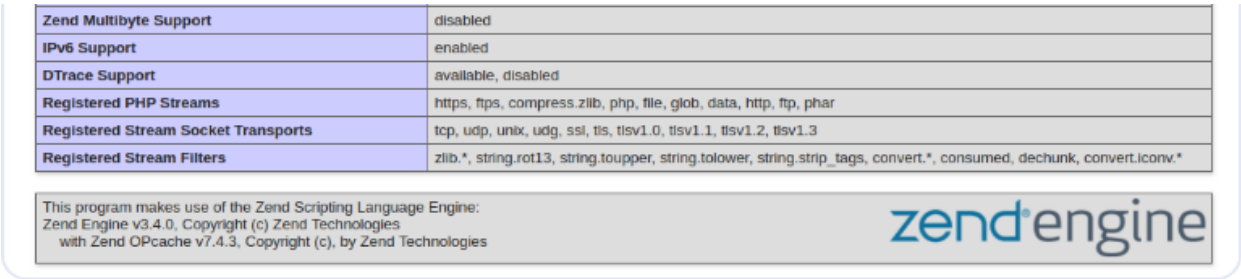
```
http://server_domain_or_IP/info.php
```

You'll see a page similar to this:



PHP Version 7.4.3

System	Linux sassy-starfish 5.4.0-26-generic #30-Ubuntu SMP Mon Apr 20 16:58:30 UTC 2020 x86_64
Build Date	Mar 26 2020 20:24:23
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ffi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no



Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

This page provides information about your server from the perspective of PHP. It is useful for debugging and to ensure that your settings are being applied correctly.

If you can see this page in your browser, then your PHP installation is working as expected.

After checking the relevant information about your PHP server through that page, it's best to remove the file you created as it contains sensitive information about your PHP environment and your Ubuntu server. You can use `rm` to do so:

```
$ sudo rm /var/www/your_domain/info.php
```

Copy

You can always recreate this page if you need to access the information again later.

To create a new database, run the following command from your MySQL console:

```
mysql> CREATE DATABASE example_database;
```

Copy

Now you can create a new user and grant them full privileges on the custom database you've just created.

The following command creates a new user named `example_user`, using `mysql_native_password` as default authentication method. We're defining this user's password as `password`, but you should replace this value with a secure password of your own choosing.

```
mysql> CREATE USER 'example_user'@'%' IDENTIFIED WITH mysql_native_password BY 'password';
```

Copy

Now we need to give this user permission over the `example_database` database:

```
mysql> GRANT ALL ON example_database.* TO 'example_user'@'%';
```

Copy

This will give the **example_user** user full privileges over the **example_database** database, while preventing this user from creating or modifying other databases on your server.

Now exit the MySQL shell with:

```
mysql> exit
```

Copy

You can test if the new user has the proper permissions by logging in to the MySQL console again, this time using the custom user credentials:

```
$ mysql -u example_user -p
```

Copy

Notice the `-p` flag in this command, which will prompt you for the password used when creating the **example_user** user. After logging in to the MySQL console, confirm that you have access to the **example_database** database:

```
mysql> SHOW DATABASES;
```

Copy

This will give you the following output:

```
Output
+--------------------+
| Database           |
+--------------------+
| example_database   |
| information_schema |
+--------------------+
```

Step 1 – Creating a MySQL Database and User for WordPress

The first step that we will take is a preparatory one. WordPress uses MySQL to manage and store site and user information. We have MySQL installed already, but we need to make a database and a user for WordPress to use.

To get started, log into the MySQL root (administrative) account by issuing this command (note that this is not the root user of your server):

```
$ mysql -u root -p
```
Copy

You will be prompted for the password you set for the MySQL root account when you installed the software.

Note: If you cannot access your MySQL database via root, as a `sudo` user you can update your root user's password by logging into the database like so:

```
$ sudo mysql -u root
```
Copy

Once you receive the MySQL prompt, you can update the root user's password. Here, replace `new_password` with a strong password of your choosing.

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'new_p
```
Copy

You may now type `EXIT;` and can log back into the database via password with the following command:

```
$ mysql -u root -p
```
Copy

Within the database, we can create an exclusive database for WordPress to control. You can call this whatever you would like, but we will be using the name **wordpress** in this guide. Create the database for WordPress by typing:

```
mysql> CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```
Copy

Note: Every MySQL statement must end in a semi-colon (`;`). Check to make sure this is present if you are running into any issues.

Next, we are going to create a separate MySQL user account that we will use exclusively to operate our new database. Creating specific databases and accounts can support us from a management and security standpoint. We will use the name **wordpressuser** in this guide, but feel free to use whatever name is relevant for you.

We are going to create this account, set a password, and grant access to the database we created. We can do this by typing the following command. Remember to choose a strong password here for your database user where we have `password`:

```
mysql> CREATE USER 'wordpressuser'@'%' IDENTIFIED WITH mysql_native_password BY 'passwor
```
Copy

Next, let the database know that our **wordpressuser** should have complete access to the database we set up:

```
mysql> GRANT ALL ON wordpress.* TO 'wordpressuser'@'%';
```
Copy

You now have a database and user account, each made specifically for WordPress. We need to flush the privileges so that the current instance of MySQL knows about the recent changes we've made:

```
mysql> FLUSH PRIVILEGES;
```
Copy

Exit out of MySQL by typing:

```
mysql> EXIT;
```
Copy

In the next step, we'll lay some foundations for WordPress plugins by downloading PHP extensions for our server.

Step 2 – Installing Additional PHP Extensions

When setting up our LAMP stack, we only required a very minimal set of extensions in order to get PHP to communicate with MySQL. WordPress and many of its plugins leverage additional PHP extensions.

We can download and install some of the most popular PHP extensions for use with WordPress by typing:

```
$ sudo apt update
$ sudo apt install php-curl php-gd php-mbstring php-xml php-xmlrpc php-soap php-intl php-zip
```
Copy

This will lay the groundwork for installing additional plugins into our WordPress site.

Note: Each WordPress plugin has its own set of requirements. Some may require additional PHP packages to be installed. Check your plugin documentation to discover its PHP requirements. If they are available, they can be installed with `apt` as demonstrated above.

We will need to restart Apache to load these new extensions, we'll be doing more configurations on Apache in the next section, so you can wait until then, or restart now to complete the PHP extension process.

```
$ sudo systemctl restart apache2
```
Copy

Step 3 – Adjusting Apache's Configuration to Allow for .htaccess Overrides and Rewrites

Next, we will be making a few minor adjustments to our Apache configuration. Based on the prerequisite tutorials, you should have a configuration file for your site in the `/etc/apache2/sites-available/` directory.

In this guide, we'll use `/etc/apache2/sites-available/wordpress.conf` as an example here, but you should substitute the path to your configuration file where appropriate. Additionally, we will use `/var/www/wordpress` as the root directory of our WordPress install. You should use the web root specified in your own configuration. If you followed our LAMP tutorial, it may be your domain name instead of `wordpress` in both of these instances.

Note: It's possible you are using the `000-default.conf` default configuration (with `/var/www/html` as your web root). This is fine to use if you're only going to host one website on this server. If not, it's better to split the necessary configuration into logical chunks, one file per site.

With our paths identified, we can move onto working with `.htaccess` so that Apache can handle configuration changes on a per-directory basis.

Enabling .htaccess Overrides

Currently, the use of `.htaccess` files is disabled. WordPress and many WordPress plugins use these files extensively for in-directory tweaks to the web server's behavior.

Open the Apache configuration file for your website with a text editor like nano.

```
$ sudo nano /etc/apache2/sites-available/wordpress.conf
```
Copy

To allow `.htaccess` files, we need to set the `AllowOverride` directive within a `Directory` block pointing to our document root. Add the following block of text inside the `VirtualHost` block in your configuration file, making sure to use the correct web root directory:

/etc/apache2/sites-available/wordpress.conf

```
<Directory /var/www/wordpress/>
        AllowOverride All
</Directory>
```

When you are finished, save and close the file. In nano, you can do this by pressing `CTRL` and `X` together, then `Y`, then `ENTER`.

Enabling the Rewrite Module

Next, we can enable `mod_rewrite` so that we can utilize the WordPress permalink feature:

```
$ sudo a2enmod rewrite
```
Copy

This allows you to have more human-readable permalinks to your posts, like the following two examples:

```
http://example.com/2012/post-name/
http://example.com/2012/12/30/post-name
```

The `a2enmod` command calls a script that enables the specified module within the Apache configuration.

Enabling the Changes

Before we implement the changes we've made, check to make sure we haven't made any syntax errors by running the following test.

```
$ sudo apache2ctl configtest
```
Copy

You may receive output like the following:

```
Output
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127
Syntax OK
```

If you wish to suppress the top line, just add a `ServerName` directive to your main (global) Apache configuration file at `/etc/apache2/apache2.conf`. The `ServerName` can be your server's domain or IP address. This is just a message, however, and doesn't affect the functionality of your site. As long as the output contains `Syntax OK`, you are ready to continue.

Restart Apache to implement the changes. Make sure to restart now even if you have restarted earlier in this tutorial.

```
$ sudo systemctl restart apache2
```
Copy

Next, we will download and set up WordPress itself.

Step 4 – Downloading WordPress

Now that our server software is configured, we can download and set up WordPress. For security reasons in particular, it is always recommended to get the latest version of WordPress from their site.

Change into a writable directory (we recommend a temporary one like `/tmp`) and download the compressed release.

```
$ cd /tmp
$ curl -O https://wordpress.org/latest.tar.gz
```
Copy

Extract the compressed file to create the WordPress directory structure:

```
$ tar xzvf latest.tar.gz
```
Copy

We will be moving these files into our document root momentarily. Before we do, we can add a dummy `.htaccess` file so that this will be available for WordPress to use later.

Create the file by typing:

```
$ touch /tmp/wordpress/.htaccess
```
Copy

We'll also copy over the sample configuration file to the filename that WordPress reads:

```
$ cp /tmp/wordpress/wp-config-sample.php /tmp/wordpress/wp-config.php
```
Copy

We can also create the `upgrade` directory, so that WordPress won't run into permissions issues when trying to do this on its own following an update to its software:

```
$ mkdir /tmp/wordpress/wp-content/upgrade
```
Copy

Now, we can copy the entire contents of the directory into our document root. We are using a dot at the end of our source directory to indicate that everything within the directory should be copied, including hidden files (like the `.htaccess` file we created):

```
$ sudo cp -a /tmp/wordpress/. /var/www/wordpress
```
Copy

Ensure that you replace the `/var/www/wordpress` directory with the directory you have set up on your server.

Step 5 – Configuring the WordPress Directory

Before we do the web-based WordPress setup, we need to adjust some items in our WordPress directory.

Adjusting the Ownership and Permissions

An important step that we need to accomplish is setting up reasonable file permissions and ownership.

We'll start by giving ownership of all the files to the **www-data** user and group. This is the user that the Apache web server runs as, and Apache will need to be able to read and write WordPress files in order to serve the website and perform automatic updates.

Update the ownership with the `chown` command which allows you to modify file ownership. Be sure to point to your server's relevant directory.

```
$ sudo chown -R www-data:www-data /var/www/wordpress
```
Copy

Next we'll run two `find` commands to set the correct permissions on the WordPress directories and files:

```
$ sudo find /var/www/wordpress/ -type d -exec chmod 750 {} \;
$ sudo find /var/www/wordpress/ -type f -exec chmod 640 {} \;
```
Copy

These permissions should get you working effectively with WordPress, but note that some plugins and procedures may require additional tweaks.

Setting Up the WordPress Configuration File

Now, we need to make some changes to the main WordPress configuration file.

When we open the file, our first task will be to adjust some secret keys to provide a level of security for our installation. WordPress provides a secure generator for these values so that you do not have to try to come up with good values on your own. These are only used internally, so it won't hurt usability to have complex, secure values here.

To grab secure values from the WordPress secret key generator, type:

```
$ curl -s https://api.wordpress.org/secret-key/1.1/salt/
```
Copy

You will get back unique values that resemble output similar to the block below.

Warning! It is important that you request unique values each time. Do **NOT** copy the values below!

```
Output
define('AUTH_KEY',         '1jl/vqfs<XhdXoAPz9 DO NOT COPY THESE VALUES c_j{iwqD^<+c9.k<J@4H');
define('SECURE_AUTH_KEY',  'E2N-h2]Dcvp+aS/p7X DO NOT COPY THESE VALUES {Ka(f;rv?Pxf})CgLi-3');
define('LOGGED_IN_KEY',    'W(50,{W^,OPB%PB<JF DO NOT COPY THESE VALUES 2;y&,2m%3]R6DUth[;88');
define('NONCE_KEY',        'll,4UC)7ua+8<!4VM+ DO NOT COPY THESE VALUES #`DXF+[$atzM7 o^-C7g');
define('AUTH_SALT',        'koMrurzOA+|L_lG}kf DO NOT COPY THESE VALUES  07VC*Lj*lD&?3w!BT#-');
define('SECURE_AUTH_SALT', 'p32*p,]z%LZ+pAu:VY DO NOT COPY THESE VALUES C-?y+K0DK_+F|0h{!_xY');
define('LOGGED_IN_SALT',   'i^/G2W7!-1H2OQ+t$3 DO NOT COPY THESE VALUES t6**bRVFSD[Hi])-qS`|');
define('NONCE_SALT',       'Q6]U:K?j4L%Z]}h^q7 DO NOT COPY THESE VALUES 1% ^qUswWgn+6&xqHN&%');
```

These are configuration lines that we can paste directly in our configuration file to set secure keys.

These are configuration lines that we can paste directly in our configuration file to set secure keys. Copy the output you received now.

Next, open the WordPress configuration file:

```
$ sudo nano /var/www/wordpress/wp-config.php
```
Copy

Find the section that contains the example values for those settings.

/var/www/wordpress/wp-config.php

```
. . .

define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY',  'put your unique phrase here');
define('LOGGED_IN_KEY',    'put your unique phrase here');
define('NONCE_KEY',        'put your unique phrase here');
define('AUTH_SALT',        'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',   'put your unique phrase here');
define('NONCE_SALT',       'put your unique phrase here');

. . .
```

Delete those lines and paste in the values you copied from the command line:

/var/www/wordpress/wp-config.php

```
. . .

define('AUTH_KEY',         'VALUES COPIED FROM THE COMMAND LINE');
define('SECURE_AUTH_KEY',  'VALUES COPIED FROM THE COMMAND LINE');
define('LOGGED_IN_KEY',    'VALUES COPIED FROM THE COMMAND LINE');
define('NONCE_KEY',        'VALUES COPIED FROM THE COMMAND LINE');
define('AUTH_SALT',        'VALUES COPIED FROM THE COMMAND LINE');
define('SECURE_AUTH_SALT', 'VALUES COPIED FROM THE COMMAND LINE');
define('LOGGED_IN_SALT',   'VALUES COPIED FROM THE COMMAND LINE');
define('NONCE_SALT',       'VALUES COPIED FROM THE COMMAND LINE');

. . .
```

Next, we are going to modify some of the database connection settings at the beginning of the file. You need to adjust the database name, the database user, and the associated password that you configured within MySQL.

The other change we need to make is to set the method that WordPress should use to write to the filesystem. Since we've given the web server permission to write where it needs to, we can explicitly set the filesystem method to "direct". Failure to set this with our current settings would result in WordPress prompting for FTP credentials when we perform some actions.

This setting can be added below the database connection settings, or anywhere else in the file:

/var/www/wordpress/wp-config.php

```
. . .

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** MySQL database username */
define( 'DB_USER', 'wordpressuser' );

/** MySQL database password */
define( 'DB_PASSWORD', 'password' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );


. . .

define('FS_METHOD', 'direct');
```

Save and close the file when you are finished.

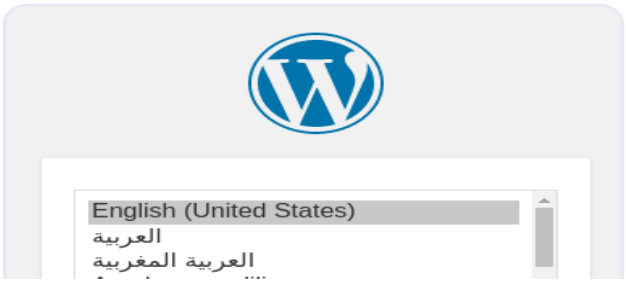
Step 6 – Completing the Installation Through the Web Interface

Now that the server configuration is complete, we can complete the installation through the web interface.

In your web browser, navigate to your server's domain name or public IP address:

```
https://server_domain_or_IP
```

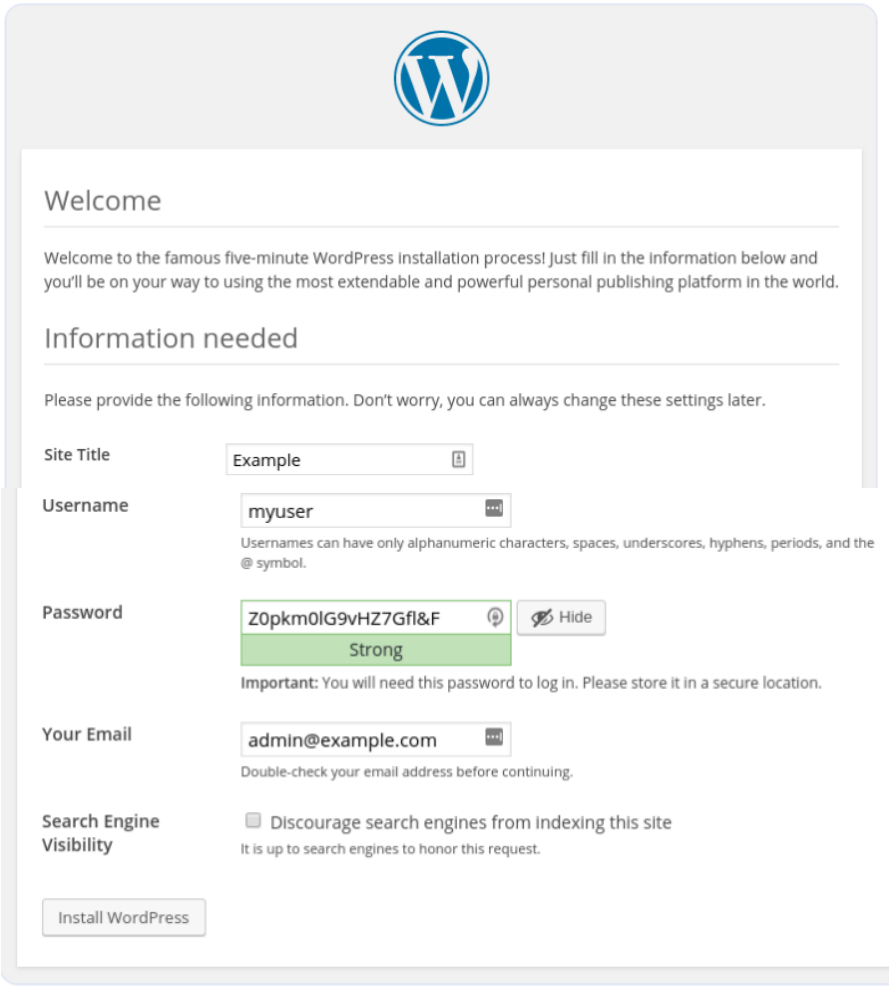
Select the language you would like to use:



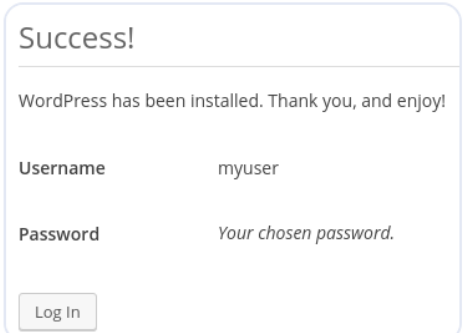
Next, you will come to the main setup page.

Select a name for your WordPress site and choose a username. It is recommended to choose something unique and avoid common usernames like “admin” for security purposes. A strong password is generated automatically. Save this password or select an alternative strong password.

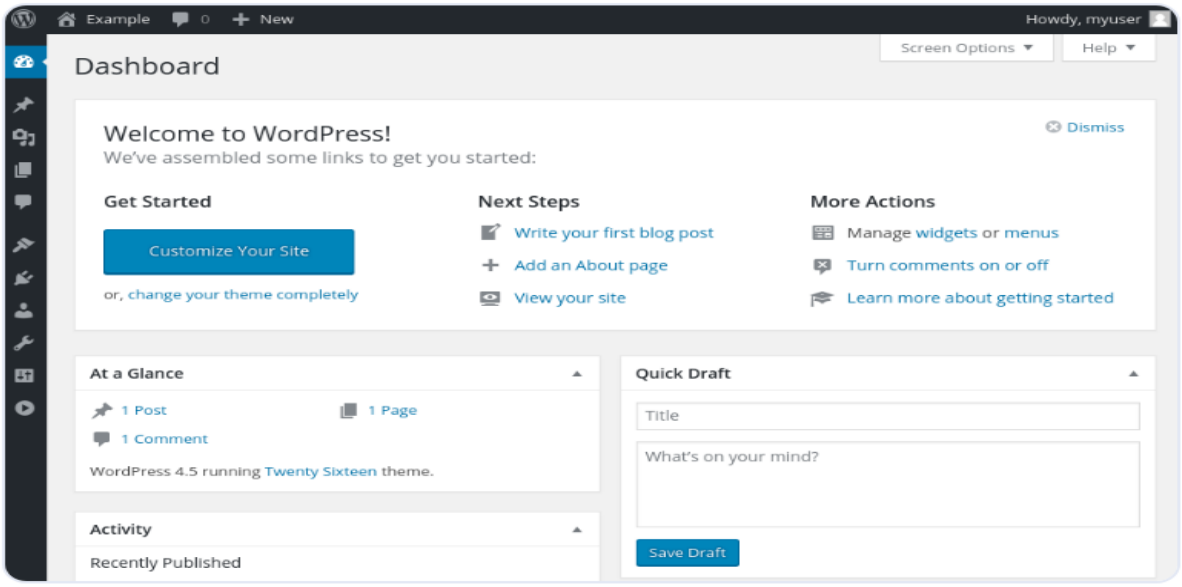
Enter your email address and select whether you want to discourage search engines from indexing your site:



When you click ahead, you will be taken to a page that prompts you to log in:



Once you log in, you will be taken to the WordPress administration dashboard:



At this point, you can begin to design your WordPress website! If this is your first time using WordPress, explore the interface a bit to get acquainted with your new CMS.

Conclusion

Congratulations, WordPress is now installed and is ready to be used!

At this point you may want to start doing the following:

- Choose your permalinks setting for WordPress posts, which can be found in `Settings > Permalinks`.
- Select a new theme in `Appearance > Themes`.
- Install new plugins to increase your site's functionality under `Plugins > Add New`.
- If you are going to collaborate with others, you may also wish to add additional users at this time under `Users > Add New`.

You can find additional resources for alternate ways to install WordPress, learn how to install WordPress on different server distributions, automate your WordPress installations, and scale your WordPress sites by checking out our WordPress Community tag.

Initial Server Setup with Ubuntu 20.04

Step 1 – Logging in as root

To log into your server, you will need to know your **server's public IP address**. You will also need the password or — if you installed an SSH key for authentication — the private key for the **root** user's account. If you have not already logged into your server, you may want to follow our guide on how to Connect to Droplets with SSH, which covers this process in detail.

If you are not already connected to your server, log in now as the **root** user using the following command (substitute the highlighted portion of the command with your server's public IP address):

```
$ ssh root@your_server_ip
```

Copy

Accept the warning about host authenticity if it appears. If you are using password authentication, provide your **root** password to log in. If you are using an SSH key that is passphrase protected, you may be prompted to enter the passphrase the first time you use the key each session. If this is your first time logging into the server with a password, you may also be prompted to change the **root** password.

About root

The **root** user is the administrative user in a Linux environment that has very broad privileges. Because of the heightened privileges of the **root** account, you are *discouraged* from using it on a regular basis. This is because the **root** account is able to make very destructive changes, even by accident.

The next step is setting up a new user account with reduced privileges for day-to-day use. Later, we'll show you how to temporarily gain increased privileges for the times when you need them.

Step 2 – Creating a New User

Once you are logged in as **root**, you'll be able to add the new user account. In the future, we'll log in with this new account instead of **root**.

This example creates a new user called **sammy**, but you should replace that with a username that you like:

```
# adduser sammy
```

Copy

You will be asked a few questions, starting with the account password.

Enter a strong password and, optionally, fill in any of the additional information if you would like. This is not required and you can just hit `ENTER` in any field you wish to skip.

Step 3 – Granting Administrative Privileges

Now we have a new user account with regular account privileges. However, we may sometimes need to do administrative tasks.

To avoid having to log out of our normal user and log back in as the **root** account, we can set up what is known as *superuser* or **root** privileges for our normal account. This will allow our normal user to run commands with administrative privileges by putting the word `sudo` before the command.

To add these privileges to our new user, we need to add the user to the **sudo** group. By default, on Ubuntu 20.04, users who are members of the **sudo** group are allowed to use the `sudo` command.

As **root**, run this command to add your new user to the **sudo** group (substitute the highlighted username with your new user):

```
# usermod -aG sudo sammy
```

Copy

Now, when logged in as your regular user, you can type `sudo` before commands to run them with superuser privileges.

Step 4 – Setting Up a Basic Firewall

Ubuntu 20.04 servers can use the UFW firewall to make sure only connections to certain services are allowed. We can set up a basic firewall using this application.

Note: If your servers are running on DigitalOcean, you can optionally use DigitalOcean Cloud Firewalls instead of the UFW firewall. We recommend using only one firewall at a time to avoid conflicting rules that may be difficult to debug.

Applications can register their profiles with UFW upon installation. These profiles allow UFW to manage these applications by name. OpenSSH, the service allowing us to connect to our server now, has a profile registered with UFW.

You can see this by typing:

```
# ufw app list
```

Copy

```
Output
Available applications:
  OpenSSH
```

We need to make sure that the firewall allows SSH connections so that we can log back in next time. We can allow these connections by typing:

```
# ufw allow OpenSSH
```
Copy

Afterwards, we can enable the firewall by typing:

```
# ufw enable
```
Copy

Type `y` and press `ENTER` to proceed. You can see that SSH connections are still allowed by typing:

```
# ufw status
```
Copy

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

As **the firewall is currently blocking all connections except for SSH**, if you install and configure additional services, you will need to adjust the firewall settings to allow traffic in. You can learn some common UFW operations in our *UFW Essentials* guide.

Step 5 – Enabling External Access for Your Regular User

Now that we have a regular user for daily use, we need to make sure we can SSH into the account directly.

Note: Until verifying that you can log in and use `sudo` with your new user, we recommend staying logged in as **root**. This way, if you have problems, you can troubleshoot and make any necessary changes as **root**. If you are using a DigitalOcean Droplet and experience problems with your **root** SSH connection, you can regain access to Droplets using the Recovery Console.

The process for configuring SSH access for your new user depends on whether your server's **root** account uses a password or SSH keys for authentication.

If the root Account Uses Password Authentication

If you logged in to your **root** account *using a password*, then password authentication is *enabled* for SSH. You can SSH to your new user account by opening up a new terminal session and using SSH with your new username:

```
$ ssh sammy@your_server_ip
```
Copy

After entering your regular user's password, you will be logged in. Remember, if you need to run a command with administrative privileges, type `sudo` before it like this:

```
$ sudo command_to_run
```

Copy

You will be prompted for your regular user password when using `sudo` for the first time each session (and periodically afterwards).

To enhance your server's security, **we strongly recommend setting up SSH keys instead of using password authentication**. Follow our guide on setting up SSH keys on Ubuntu 20.04 to learn how to configure key-based authentication.

If the root Account Uses SSH Key Authentication

If you logged in to your **root** account *using SSH keys*, then password authentication is *disabled* for SSH. You will need to add a copy of your local public key to the new user's `~/.ssh/authorized_keys` file to log in successfully.

Since your public key is already in the **root** account's `~/.ssh/authorized_keys` file on the server, we can copy that file and directory structure to our new user account in our existing session.

The simplest way to copy the files with the correct ownership and permissions is with the `rsync` command. This will copy the **root** user's `.ssh` directory, preserve the permissions, and modify the file owners, all in a single command. Make sure to change the highlighted portions of the command below to match your regular user's name:

> **Note:** The `rsync` command treats sources and destinations that end with a trailing slash differently than those without a trailing slash. When using `rsync` below, be sure that the source directory (`~/.ssh`) **does not** include a trailing slash (check to make sure you are not using `~/.ssh/`).
>
> If you accidentally add a trailing slash to the command, `rsync` will copy the *contents* of the **root** account's `~/.ssh` directory to the `sudo` user's home directory instead of copying the entire `~/.ssh` directory structure. The files will be in the wrong location and SSH will not be able to find and use them.

```
# rsync --archive --chown=sammy:sammy ~/.ssh /home/sammy
```

Copy

Now, open up a new terminal session on your local machine, and use SSH with your new username:

```
$ ssh sammy@your_server_ip
```

Copy

You should be logged in to the new user account without using a password. Remember, if you need to run a command with administrative privileges, type `sudo` before it like this:

```
$ sudo command_to_run
```

Copy

Install wordpress on windaw iis server

Step 1: Download WordPress

1. Go to the official WordPress website (https://wordpress.org/download/) and download the latest version of WordPress in ZIP format.

Step 2: Extract the WordPress Files

1. Extract the contents of the ZIP file you downloaded to a directory on your Windows server where you want to host your WordPress site. This will be your website's root directory (e.g., `C:\inetpub\wwwroot\mywordpress`).

Step 3: Create a MySQL Database

1. You'll need a MySQL database to store WordPress data. Install MySQL on your Windows server if you haven't already.
2. Create a new MySQL database for WordPress. You can use a MySQL client like phpMyAdmin or the MySQL command line to create the database.
3. Create a MySQL user and grant it privileges on the WordPress database. Note down the database name, username, and password for later use.

Step 4: Configure PHP on IIS

1. Install PHP on your Windows server. You can use a package like XAMPP or install PHP manually.
2. Configure PHP to work with IIS by adding PHP as a FastCGI module. You can use the "Internet Information Services (IIS) Manager" to configure this.

Step 5: Configure WordPress

1. Rename the `wp-config-sample.php` file in your WordPress directory to `wp-config.php`.
2. Open `wp-config.php` and configure the database settings by entering the database name, username, and password you created earlier:

```php
define('DB_NAME', 'your_database_name');
define('DB_USER', 'your_database_user');
define('DB_PASSWORD', 'your_database_password');
```

3. You may also want to define other constants like the authentication keys and salts. You can generate these keys by visiting the WordPress.org secret key generator (https://api.wordpress.org/secret-key/1.1/salt/).
4. Save your `wp-config.php` file.

Step 6: Configure IIS for WordPress

1. Open the "Internet Information Services (IIS) Manager."
2. In the Connections pane on the left, expand your server node and click on "Sites."
3. Right-click on "Default Web Site" or create a new site if needed, and select "Add Application."
4. Set an alias for your WordPress site (e.g., "mywordpress") and browse for the physical path where you extracted the WordPress files.
5. In the "Edit Application" dialog, go to the "Modules" section, and ensure that the "FastCgiModule" is installed and enabled.

Step 7: Access Your WordPress Site

1. Open a web browser and enter the URL to your WordPress site (e.g., `http://localhost/mywordpress`). Follow the WordPress installation instructions.
2. Complete the installation by providing your site title, username, password, and email address.
3. Once the installation is complete, you can log in to the WordPress admin dashboard and start customizing your site.

That's it! You've successfully installed WordPress on your Windows IIS server. You can now start building and managing your website using the WordPress platform.

WP-CLI

Installing on Windows

Install via composer as described above or use the following method.

Make sure you have php installed and in your path so you can execute it globally.

Download wp-cli.phar manually and save it to a folder, for example `c:\wp-cli`

Create a file named `wp.bat` in `c:\wp-cli` with the following contents:

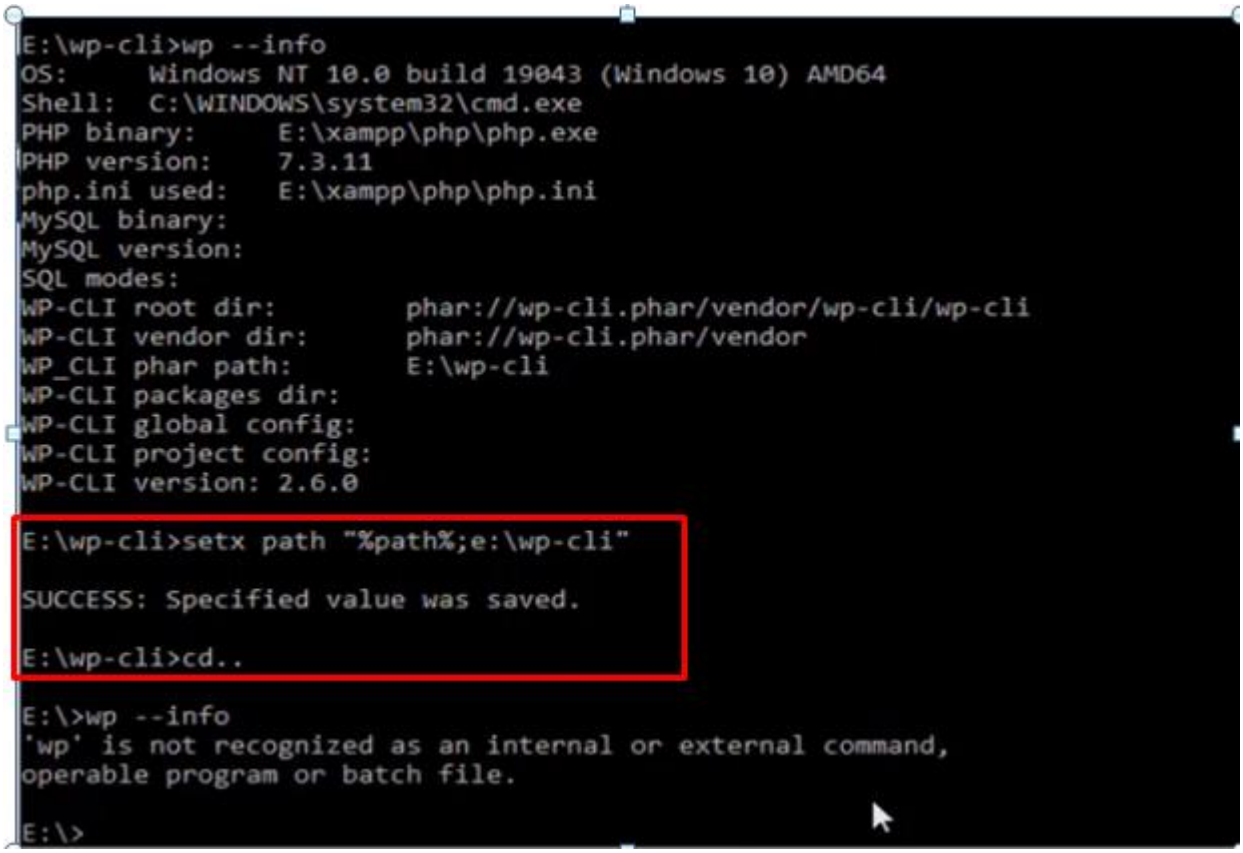
```
@ECHO OFF
php "c:/wp-cli/wp-cli.phar" %*
```

Add `c:\wp-cli` to your path:

```
setx path "%path%;c:\wp-cli"
```

You can now use WP-CLI from anywhere in Windows command line.

Environment variable path setting example:



The recommended way to install WP-CLI is by downloading the Phar build (archives similar to Java JAR files, see this article for more detail), marking it executable, and placing it on your PATH.

First, download wp-cli.phar using `wget` or `curl`. For example:

```
curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar
```

Then, check if it works:

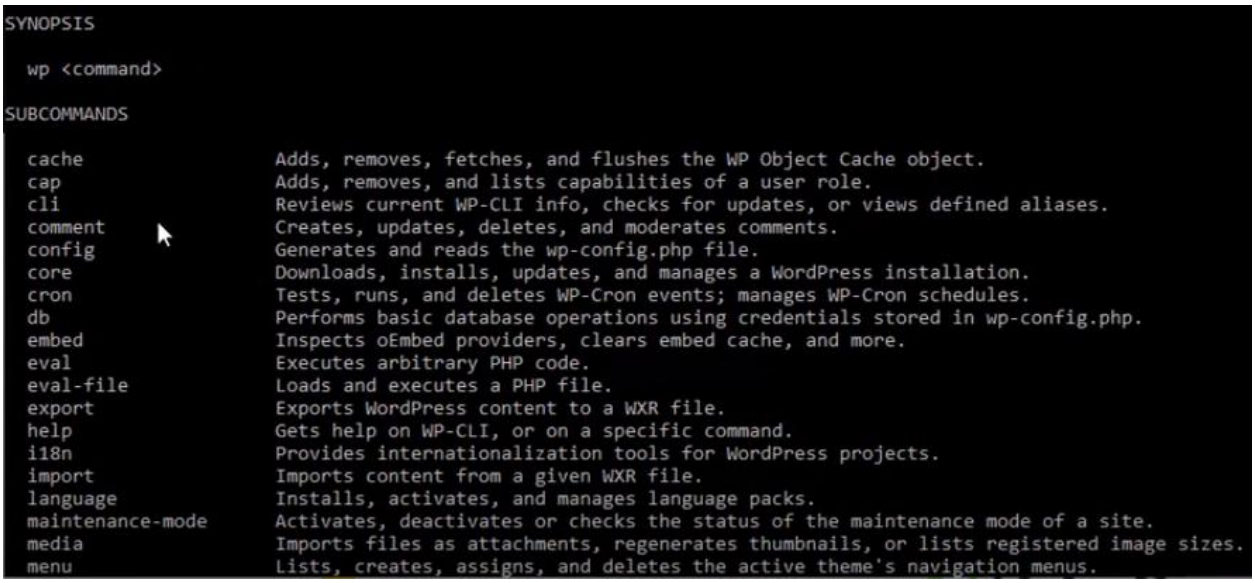
```
php wp-cli.phar --info
```

To be able to type just `wp`, instead of `php wp-cli.phar`, you need to make the file executable and move it to somewhere in your PATH. For example:

```
chmod +x wp-cli.phar
sudo mv wp-cli.phar /usr/local/bin/wp
```

Now try running `wp --info`. If WP-CLI is installed successfully, you'll see output like this:

wp+ enter key (wp help command) for showing all command



Sub command

```
wp core

DESCRIPTION

  Downloads, installs, updates, and manages a WordPress installation.

SYNOPSIS

  wp core <command>

SUBCOMMANDS

  check-update          Checks for WordPress updates via Version Check API.
  download              Downloads core WordPress files.
  install               Runs the standard WordPress installation process.
  is-installed          Checks if WordPress is installed.
  multisite-convert     Transforms an existing single-site installation into a multisite installation.
  multisite-install     Installs WordPress multisite from scratch.
  update                Updates WordPress to a newer version.
  update-db             Runs the WordPress database update procedure.
  verify-checksums      Verifies WordPress files against WordPress.org's checksums.
  version               Displays the WordPress version.

EXAMPLES

    # Download WordPress core
    $ wp core download --locale=nl_NL
-- More  --
```

a

```
wp core download

DESCRIPTION

  Downloads core WordPress files.

SYNOPSIS

  wp core download [<download-url>] [--path=<path>] [--locale=<locale>] [--version=<version>] [--skip-content] [--force] [--insecure]

  Downloads and extracts WordPress core files to the specified path. Uses
  current directory when no path is specified. Downloaded build is verified
  to have the correct md5 and then cached to the local filesytem.
  Subsequent uses of command will use the local cache if it still exists.

OPTIONS

  [<download-url>]
    Download directly from a provided URL instead of fetching the URL from the wordpress.org servers.

  [--path=<path>]
    Specify the path in which to install WordPress. Defaults to current
    directory.

  [--locale=<locale>]
    Select which language you want to download.

  [--version=<version>]
    Select which version you want to download. Accepts a version number, 'latest' or 'nightly'.

  [--skip-content]
    Download WP without the default themes and plugins.

  [--force]
    Overwrites existing files, if present.
```

Downloading example

```
E:\xampp\htdocs\wpcli>wp core download
Downloading WordPress 6.0.1 (en_US)...
Using cached file 'C:\Users\SD/.wp-cli/cache/core/wordpress-6.0.1-en_US.tar.gz'...
```

```
  wp db

DESCRIPTION

  Performs basic database operations using credentials stored in wp-config.php.

SYNOPSIS

  wp db <command>

SUBCOMMANDS

  check         Checks the current status of the database.
  clean         Removes all tables with `$table_prefix` from the database.
  cli           Opens a MySQL console using credentials from wp-config.php
  columns       Displays information about a given table.
  create        Creates a new database.
  drop          Deletes the existing database.
  export        Exports the database to a file or to STDOUT.
  import        Imports a database from a file or from STDIN.
  optimize      Optimizes the database.
  prefix        Displays the database table prefix.
  query         Executes a SQL query against the database.
  repair        Repairs the database.
  reset         Removes all tables from the database.
  search        Finds a string in the database.
  size          Displays the database name and size.

E:\xampp\htdocs\wpcli>
```

```
E:\xampp\htdocs\wpcli>wp post create --post_content="test post content" --post_title="test post" --post
_status=publish
Success: Created post 5.

E:\xampp\htdocs\wpcli>wp post create --post_content="second test post content" --post_title="second tes
t post" --post_status=publish --post_author=1
Success: Created post 7.

E:\xampp\htdocs\wpcli>wp post list
+----+-----------------+-----------------+---------------------+-------------+
| ID | post_title      | post_name       | post_date           | post_status |
+----+-----------------+-----------------+---------------------+-------------+
| 7  | second test post | second-test-post | 2022-07-24 14:12:34 | publish     |
| 5  | test post       | test-post       | 2022-07-24 14:11:15 | publish     |
| 1  | Hello world!    | hello-world     | 2022-07-24 12:27:51 | publish     |
+----+-----------------+-----------------+---------------------+-------------+

E:\xampp\htdocs\wpcli>wp post create --post_content="test page content" --post_title="test page" --post
_status=publish --post_author=1 --post_type=page
Success: Created post 8.

E:\xampp\htdocs\wpcli>
```

```
E:\xampp\htdocs\wpcli>wp post generate --count=4 --post_type=page
Generating posts  100% [=============================================================] 0:01 / 0:01

E:\xampp\htdocs\wpcli>wp post list --post_type=page
+----+---------------+---------------+---------------------+-------------+
| ID | post_title    | post_name     | post_date           | post_status |
+----+---------------+---------------+---------------------+-------------+
| 11 | test page     | test-page-3   | 2022-07-30 00:00:00 | future      |
| 22 | Page 5        | post-5        | 2022-07-24 18:32:35 | publish     |
| 23 | Page 6        | post-6        | 2022-07-24 18:32:35 | publish     |
| 24 | Page 7        | post-7        | 2022-07-24 18:32:35 | publish     |
| 25 | Page 8        | post-8        | 2022-07-24 18:32:35 | publish     |
| 10 | test page     | test-page-2   | 2022-07-24 14:15:17 | publish     |
| 8  | test page     | test-page     | 2022-07-24 14:14:17 | publish     |
| 2  | About Us      | sample-page   | 2022-07-24 12:27:51 | publish     |
| 3  | Privacy Policy| privacy-policy| 2022-07-24 12:27:51 | draft       |
+----+---------------+---------------+---------------------+-------------+
```

```
E:\xampp\htdocs\wpcli>wp theme list
+-----------------+----------+--------+---------+
| name            | status   | update | version |
+-----------------+----------+--------+---------+
| hello-elementor | active   | none   | 2.6.1   |
| twentytwenty    | inactive | none   | 2.0     |
| twentytwentyone | inactive | none   | 1.6     |
| twentytwentytwo | inactive | none   | 1.2     |
+-----------------+----------+--------+---------+

E:\xampp\htdocs\wpcli>wp theme install ./hestia.3.0.23.zip --activate
Unpacking the package...
Installing the theme...
Theme installed successfully.
Activating 'hestia'...
Success: Switched to 'Hestia' theme.
Success: Installed 1 of 1 themes.
```

```
  wp plugin <command>

SUBCOMMANDS

  activate              Activates one or more plugins.
  auto-updates          Manages plugin auto-updates.
  deactivate            Deactivates one or more plugins.
  delete                Deletes plugin files without deactivating or uninstalling.
  get                   Gets details about an installed plugin.
  install               Installs one or more plugins.
  is-active             Checks if a given plugin is active.
  is-installed          Checks if a given plugin is installed.
  list                  Gets a list of plugins.
  path                  Gets the path to a plugin or to the plugin directory.
```

```
E:\xampp\htdocs\wpcli>wp user list-caps demo2@test.com
upload_files
edit_posts
edit_published_posts
publish_posts
read
level_2
level_1
level_0
delete_posts
delete_published_posts
rank_math_onpage_analysis
rank_math_onpage_general
rank_math_onpage_snippet
rank_math_onpage_social
author
```

```
  wp user generate

DESCRIPTION

  Generates some users.

SYNOPSIS

  wp user generate [--count=<number>] [--role=<role>] [--format=<format>]

  Creates a specified number of new users with dummy data.

OPTIONS

  [--count=<number>]
    How many users to generate?
    ---
    default: 100
    ---

  [--role=<role>]
    The role of the generated users. Default: default role from WP
-- More  --
```

```
E:\xampp\htdocs\wpcli>wp user generate --count=5 --role=author
Generating users  100% [=============================================================] 0:01 / 0:02

E:\xampp\htdocs\wpcli>wp user list
+----+----------------+----------------+---------------------+---------------------+--------------+
| ID | user_login     | display_name   | user_email          | user_registered     | roles        |
+----+----------------+----------------+---------------------+---------------------+--------------+
| 1  | admin          | admin          | admin@admin.com     | 2022-07-24 12:27:51 | administrator |
| 3  | demo2          | demo 2         | demo2@test.com      | 2022-07-31 13:25:22 | author       |
| 5  | diwakar        | Diwakar Academy | diwakar@classes.com | 0000-00-00 00:00:00 | editor       |
| 4  | diwakar_academy | Diwakar Academy | diwakar@academy.com | 0000-00-00 00:00:00 | author       |
| 2  | domo1          | demo1          | demo1@test.com      | 2022-07-31 13:24:21 | editor       |
| 6  | user_1_5       | User 5         |                     | 2022-07-31 14:08:43 | author       |
| 7  | user_1_6       | User 6         |                     | 2022-07-31 14:08:43 | author       |
| 8  | user_1_7       | User 7         |                     | 2022-07-31 14:08:43 | author       |
| 9  | user_1_8       | User 8         |                     | 2022-07-31 14:08:44 | author       |
| 10 | user_1_9       | User 9         |                     | 2022-07-31 14:08:44 | author       |
+----+----------------+----------------+---------------------+---------------------+--------------+
```

WP CLI several benefits for developers, site administrators, and anyone who works with WordPress:

1. **Automation**: WP-CLI allows you to automate various tasks, such as updating plugins and themes, creating backups, importing/exporting content, and more. This can save you a lot of time and reduce the potential for human error.
2. **Speed**: Command-line operations are typically faster than performing the same tasks through the WordPress admin interface. This can be especially helpful when working with large websites or complex operations.
3. **Scripting**: You can write custom scripts and batch operations with WP-CLI, making it easier to perform repetitive tasks. This is useful for tasks like setting up a new WordPress instance, migrating content, or performing site maintenance.
4. **Managing Plugins and Themes**: WP-CLI allows you to install, activate, update, and delete plugins and themes with simple commands. This is a faster and more efficient way to manage your site's extensions.
5. **Database Management**: You can interact with the WordPress database using WP-CLI, making it easier to run SQL queries, repair the database, or perform other maintenance tasks.
6. **User Management**: WP-CLI makes it easy to create, update, and delete user accounts, change user passwords, and manage user roles and permissions.
7. **Debugging**: You can use WP-CLI to troubleshoot issues on your site by enabling or disabling plugins, themes, or debugging features.

8. **Content Management**: WP-CLI allows you to create and manage content, including posts, pages, and custom post types. You can even generate dummy content for testing purposes.
9. **Security**: WP-CLI can help enhance the security of your WordPress site by automating security tasks such as user password resets, security plugin management, and vulnerability scans.
10. **Version Control Integration**: WP-CLI can be integrated into version control workflows (e.g., Git), making it easier to deploy changes to your WordPress site and ensure consistency across different environments.
11. **Multisite Management**: If you're running a WordPress multisite network, WP-CLI simplifies the management of network-wide settings, sites, and user accounts.
12. **Customization**: WP-CLI is highly customizable, allowing you to create custom commands and scripts tailored to your specific needs.
13. **Community and Third-Party Plugins**: There is a growing community of developers who contribute to the WP-CLI project, and there are also third-party packages and plugins available to extend its functionality.

1. **Efficiency**: WP-CLI allows you to perform various tasks more efficiently than using the WordPress admin dashboard. Commands are typically faster and can be automated, saving you time and effort.
2. **Scripting and Automation**: You can create custom scripts and automate tasks with WP-CLI. This is particularly useful for batch operations, maintenance tasks, and deployment processes.
3. **Batch Operations**: You can perform actions on multiple items at once. For example, you can install, activate, or update multiple plugins or themes in one command.
4. **Database Management**: WP-CLI provides commands for database operations, allowing you to interact with the database, run SQL queries, repair tables, and manage the database efficiently.
5. **User Management**: You can create, update, and delete user accounts, change user roles and permissions, and perform user-related tasks via WP-CLI.
6. **Theme and Plugin Management**: WP-CLI simplifies theme and plugin management. You can install, activate, deactivate, update, or delete themes and plugins from the command line.

7. **Content Management**: You can create and manage posts, pages, custom post types, and even generate dummy content for testing purposes using WP-CLI commands.
8. **Site Maintenance**: WP-CLI offers tools for site maintenance, including options to optimize the database, repair the database, and perform other maintenance tasks.
9. **Debugging and Troubleshooting**: You can troubleshoot issues by enabling or disabling plugins and themes, setting debugging options, and diagnosing problems more efficiently.
10. **Security**: WP-CLI can be used to enhance site security by automating tasks like password resets and managing security plugins.
11. **Multisite Management**: For WordPress multisite installations, WP-CLI simplifies the management of network-wide settings, sites, and users.
12. **Customization**: You can extend WP-CLI's functionality by creating custom commands to suit your specific needs or integrate it with other tools and scripts.
13. **Version Control Integration**: WP-CLI can be integrated into version control workflows (e.g., Git) to streamline development, deployment, and consistency across different environments.
14. **Third-Party Plugins and Packages**: There is a community of developers who contribute to WP-CLI, and third-party packages and plugins can extend its functionality for various use cases.

Rege

WP-CLI offers advanced capabilities that can greatly streamline your WordPress development and management tasks. Here are some advanced use cases for WP-CLI:

1. **Custom Command Development**: One of the most powerful features of WP-CLI is the ability to create custom commands tailored to your specific needs. These commands can be used to automate complex or site-specific tasks. For example, you could create a custom command to migrate data from a legacy CMS to WordPress.
2. **Continuous Integration and Deployment (CI/CD)**: WP-CLI can be integrated into CI/CD pipelines to automate testing, deployment, and maintenance processes. You can use WP-CLI to automate the deployment of code, database updates, and content synchronization across environments.
3. **Multisite Network Management**: WP-CLI can be used to manage WordPress multisite networks effectively. You can create, update, and delete sites, users, and themes across the network, as well as synchronize content and configurations.
4. **Headless WordPress**: If you're building a headless WordPress site where WordPress serves as a content management system, you can use WP-CLI to automate content synchronization between WordPress and your front-end application.

5. **Database Migration and Seeding**: WP-CLI is useful for database migration tasks. You can export and import database data, create database backups, seed databases with dummy content for testing, and perform data transformations.
6. **Theme and Plugin Development**: Developers can use WP-CLI to scaffold themes and plugins, generate template files, create custom post types and taxonomies, and manage code repositories. This is particularly useful when working on large projects with numerous theme and plugin files.
7. **Custom Post Type and Taxonomy Management**: You can create, update, or delete custom post types and taxonomies using WP-CLI, streamlining the development of complex content structures.
8. **Internationalization and Localization**: WP-CLI provides tools for generating translation files and managing language packs for internationalization and localization efforts, making it easier to build multilingual sites.
9. **Performance Optimization**: WP-CLI commands can be used to optimize site performance by regenerating image thumbnails, purging cache, and minifying styles and scripts.
10. **Security and Auditing**: Advanced users can implement automated security checks and audits using WP-CLI, allowing you to monitor the integrity of your WordPress installation and plugins.
11. **Scalability**: For large WordPress sites, you can use WP-CLI to automate tasks like server provisioning, environment configuration, and content distribution to enhance scalability.
12. **API Integration**: WP-CLI can be integrated with various APIs to automate content imports, exports, and synchronization with external systems.
13. **Site Setup and Configuration**: WP-CLI can streamline the setup and configuration of new WordPress installations, including database creation, user setup, and default settings.
14. **Server and Environment Management**: WP-CLI can help you manage server and hosting environment tasks, such as updating PHP versions, configuring server settings, and checking system requirements.
15. **Logging and Monitoring**: You can set up automated monitoring and logging scripts using WP-CLI to track site performance, errors, and changes over time.

Step 1 – Installing WP-CLI

In this step, you'll install the latest version of the WP-CLI tool on your server. The tool is packaged in a Phar file, which is a packaging format for PHP applications that makes app deployment and distribution convenient.

You can download the Phar file for WP-CLI through `curl`:

```
$ curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar
```

Copy

Once you have downloaded the file, run the following command to verify that it is working:

```
$ php wp-cli.phar --info
```

Copy

You will receive the following output:

```
Output
OS:     Linux 5.4.0-51-generic #56-Ubuntu SMP Mon Oct 5 14:28:49 UTC 2020 x86_64
Shell:  /bin/bash
PHP binary:     /usr/bin/php7.4
PHP version:    7.4.3
php.ini used:   /etc/php/7.4/cli/php.ini
WP-CLI root dir:        phar://wp-cli.phar/vendor/wp-cli/wp-cli
WP-CLI vendor dir:      phar://wp-cli.phar/vendor
WP_CLI phar path:       /home/ayo
WP-CLI packages dir:
WP-CLI global config:
WP-CLI project config:
WP-CLI version: 2.4.0
```

Next, make the file executable with the following command:

```
$ chmod +x wp-cli.phar
```

Copy

At this point, you can execute the `wp-cli.phar` file directly to access the WP-CLI tool. To make it available globally on the system, move it to your `/usr/local/bin/` directory and rename it to `wp`. This ensures that you can access WP-CLI from any directory by entering the `wp` command at the start of a prompt:

```
$ sudo mv wp-cli.phar /usr/local/bin/wp
```
Copy

Now, you will be able to issue the following command to check the installed version of WP-CLI:

```
$ wp cli version
```
Copy

```
Output
WP-CLI 2.4.0
```

In this step, you installed WP-CLI on your server. You can check out alternative installation methods in the documentation. In subsequent sections, you'll explore the tasks you can accomplish through the WP-CLI interface.

Step 2 – Configuring WordPress Plugins

It can be tedious to install and manage WordPress plugins through the admin user interface. It's possible to offload such tasks to WP-CLI to make the process much faster. In this section you will learn to install, update, and delete plugins on a WordPress site through the command line.

Before you proceed, make sure you are in the directory of your WordPress installation:

```
$ cd /var/www/wordpress
```
Copy

Remember to change the highlighted directory name to the directory that contains your WordPress installation. This might be your domain name, if you followed the prerequisite tutorials.

Listing Current Plugins

You can list the currently installed plugins on a WordPress site with the following command:

```
$ wp plugin list
```
Copy

It displays a list of plugin names along with their status, version, and an indication of an available update.

```
Output
+---------+----------+-----------+---------+
| name    | status   | update    | version |
+---------+----------+-----------+---------+
| akismet | inactive | available | 4.1.7   |
| hello   | inactive | none      | 1.7.2   |
+---------+----------+-----------+---------+
```

Searching for Plugins

You can search for plugins through the search bar on the WordPress plugin repository page or you can use the following command for quicker access:

```
$ wp plugin search seo
```

Copy

Once you run this command, you will receive a list of the top 10 plugins that match the search term (as of early 2021). The expected output for the `seo` query is:

```
Output
Success: Showing 10 of 4278 plugins.
+-----------------------------------------------------------+--------------------+--------+
| name                                                      | slug               | rating |
+-----------------------------------------------------------+--------------------+--------+
| Yoast SEO                                                 | wordpress-seo      | 98     |
| All in One SEO                                            | all-in-one-seo-pack | 92     |
| Rank Math &#8211; SEO Plugin for WordPress                | seo-by-rank-math   | 98     |
| The SEO Framework                                         | autodescription    | 98     |
| SEOPress, on-site SEO                                     | wp-seopress        | 98     |
| Slim SEO &#8211; Fast &amp; Automated WordPress SEO Plugin | slim-seo           | 92     |
| W3 Total Cache                                            | w3-total-cache     | 88     |
| LiteSpeed Cache                                           | litespeed-cache    | 98     |
| SEO 2021 by Squirrly (Smart Strategy)                     | squirrly-seo       | 92     |
| WP-Optimize &#8211; Clean, Compress, Cache.               | wp-optimize        | 96     |
+-----------------------------------------------------------+--------------------+--------+
```

You can go to the next page by using the `--page` flag:

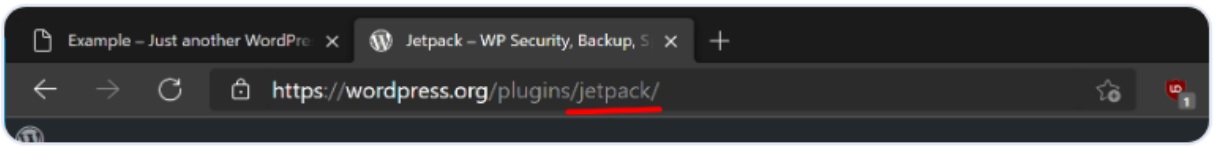
```
$ wp plugin search seo --page=2
```

Copy

Take note of the value in the `slug` column. You'll use this value to install or update the plugin on the command line.

Installing Plugins

You can install one or more plugins by using the `wp plugin install` command. You find the name of the plugin you want to install (in the `slug` column) and pass it as an argument to `wp plugin install`. You can also find the name of the plugin in the URL of the plugin page.




```
$ wp plugin install jetpack wordpress-seo gutenberg
```

Copy

The output indicates the progress and completion of the installation of each of the plugins:

```
Output
+---------------+----------+-----------+---------+
| name          | status   | update    | version |
+---------------+----------+-----------+---------+
| akismet       | inactive | available | 4.1.7   |
| gutenberg     | inactive | none      | 9.8.1   |
| hello         | inactive | none      | 1.7.2   |
| jetpack       | inactive | none      | 9.3.1   |
| wordpress-seo | inactive | none      | 15.6.2  |
+---------------+----------+-----------+---------+
```

If you want to install a plugin from a remote source other than the WordPress plugin repository, you can pass the zip file's URL as an argument to `wp plugin install`. This can be helpful for installing custom or premium plugins. For example, the following command will install the `myplugin.zip` file hosted on `example.com`. Make sure to replace the highlighted URL with a link to the plugin zip file before running the command:

```
$ wp plugin install https://example.com/wp-content/uploads/myplugin.zip
```
Copy

To install an older version of a plugin in the WordPress repository, specify the desired plugin version through the `--version` flag:

```
$ wp plugin install jetpack --version=8.0
```
Copy

Activating and Deactivating Plugins

You can install and activate plugins in one go by appending the `--activate` flag to `wp plugin install`:

```
$ wp plugin install redirection --activate
```
Copy

To activate or deactivate one or more plugins, use the `wp plugin activate` and `wp plugin deactivate` commands respectively:

```
$ wp plugin activate jetpack gutenberg
$ wp plugin deactivate jetpack gutenberg
```
Copy

Or you can use the `--all` flag to activate or deactivate all plugins at once. This is useful if you want to debug a problem in your WordPress installation:

```
$ wp plugin activate --all
$ wp plugin deactivate --all
```
Copy

Updating Plugins

You can update plugins through the `wp plugin update` command. You can choose to update a set of plugins or all of them at once by appending the `--all` flag. For example, to update the `akismet` plugin, you can run the following command:

```
$ wp plugin update akismet
```

Copy

Deleting plugins

To delete WordPress plugins, you can use the `wp plugin delete` command. You can specify one or more plugins to delete like the following:

```
$ wp plugin delete redirection
```

Copy

Your output will confirm the deletion:

```
Output
Deleted 'redirection' plugin.
Success: Deleted 1 of 1 plugins.
```

You can also delete all the installed plugins in one go by appending the `--all` flag instead of specifying the plugin names one after the other:

```
$ wp plugin delete --all
```

Copy

In this step, you've used WP-CLI to manage the plugins on your WordPress website. It's much faster to perform actions compared to clicking through the admin dashboard. In the next section, you'll leverage WP-CLI for installing and managing WordPress themes.

Step 3 – Configuring Themes

The process of managing themes through WP-CLI is almost identical to the way you can use it to manage plugins. In this section, you'll source and apply new themes to a WordPress website through the `wp theme` subcommand.

First, check what themes you currently have installed on the website:

```
$ wp theme list
```

Copy

You'll receive a list of the installed themes:

something with more features, you can try a search like the following:

```
$ wp theme search color
```

Copy

The output shows there are 832 themes that match the `color` search term:

```
Output
Success: Showing 10 of 832 themes.
+--------------------+--------------------+--------+
| name               | slug               | rating |
+--------------------+--------------------+--------+
| Color              | color              | 0      |
| All Colors         | all-colors         | 100    |
| Color Blog         | color-blog         | 98     |
| Color Block        | color-block        | 0      |
| X Blog color       | x-blog-color       | 0      |
| Multicolor Business | multicolor-business | 0      |
| ColorNews          | colornews          | 100    |
| Colorist           | colorist           | 100    |
| ColorMag           | colormag           | 98     |
| MultiColors        | multicolors        | 74     |
+--------------------+--------------------+--------+
```

You can page through the results using the `--page` flag. For this example, just go ahead and install the `ColorMag` theme since it has a pretty good rating. The `--activate` flag activates the theme immediately:

```
$ wp theme install colormag --activate
```

Copy

The output will confirm the installation:

```
Output
Installing ColorMag (2.0.4)
Downloading installation package from https://downloads.wordpress.org/theme/colormag.2.0.4.zip...
Unpacking the package...
Installing the theme...
Theme installed successfully.
```

```
$ wp theme update --all
```

Copy

The `wp theme` command offers many subcommands that can help you achieve tasks like getting the details of a theme, checking if a particular theme is installed, or even deleting one or more themes. You can explore all of the options by prepending `help` before the subcommand, as in `wp help theme` or `wp help theme install`.

Now that you can manage themes through WP-CLI, you'll review the options that the tool provides for managing WordPress content.

Step 4 – Creating Posts and Pages

WP-CLI provides several ways to manage content through the command line. It can be more comfortable to write posts in the terminal if you're familiar with a command-line editor like nano or vim.

You can browse the list of posts on the site with:

```
$ wp post list
```
Copy

You'll receive a list of posts:

```
Output
+----+--------------+-------------+---------------------+-------------+
| ID | post_title   | post_name   | post_date           | post_status |
+----+--------------+-------------+---------------------+-------------+
| 1  | Hello world! | hello-world | 2021-01-24 12:32:06 | publish     |
+----+--------------+-------------+---------------------+-------------+
```

The output shows one published post with the title of `Hello world!` and an ID of `1`. To delete this post, use the `wp post delete` command and pass it the post ID:

```
$ wp post delete 1
```
Copy

Your output will confirm the post's deletion:

```
Output
Success: Trashed post 1.
```

To create a new post, run the following command:

```
$ wp post create --post_status=publish --post_title="Sample post created with WP-CLI" --e
```
Copy

This command uses the `--post_status` flag to set the status of the post. Setting it to `publish` ensures that the post is published immediately after running the command. If you want to create a draft instead, set the `--post_status` flag to `draft`. The `--post_title` flag is how you can specify the title of the post, while `--edit` causes the post body to be opened in the default system editor (vim). You can find out the other flags that you can use in conjunction with the `create` subcommand by typing `wp help post create` in your terminal.

Once the vim editor is open, press the `i` key to enter INSERT mode then enter the content of the post into the editor. Once you're done editing the post, exit the vim editor by pressing the `ESC` button then type `:wq` and press `ENTER`. You will receive the following output after exiting vim:

```
Output
Success: Created post 6.
```

If you enter the `wp post list` command again, you will find the post you just created. You can also check the frontend of the website.

Instead of writing the post on the command line, it's also possible to import the post content from a text file. First, you need to create the file. For example:

```
$ touch content.txt
```
Copy

Next, open the file in a command-line editor to add or edit your content:

```
$ nano content.txt
```
Copy

Once you're through with the edits, save and close the file by pressing `CTRL-X` followed by `Y` to save. You can import the contents of that file as a WordPress post by using the following command. All you need to do is specify the path to the file after the `create` subcommand. For the example file here, you would run:

```
$ wp post create ./content.txt --post_title='Sample Post Created with WP-CLI' --post_stat
```
Copy li

If you want to create a page instead of a post, append the `--post_type` flag and set it to `page`:

```
$ wp post create --post_title="A simple page" --post_status=draft --post_type=page
```
Copy

Generating Posts or Pages

WP-CLI also provides an option to cleanly generate posts and pages with dummy data. This is useful if you need custom data to quickly test a theme or plugin that you are developing. The following command is to generate posts. If you don't include additional flags, it will generate 100 posts by default.

```
$ wp post generate
```
Copy

You can change the number of posts generated by using the `--count` flag:

```
$ wp post generate --count=20
```
Copy

If you want to generate pages instead of posts, append the `--post_type` flag and set it to `page`:

```
$ wp post generate --count=20 --post_type=page
```

Copy

You can also use the `wp help post generate` to see other available options that can help you get your desired result.

WordPress Revisions

It is not uncommon for older sites to have tens or hundreds of revisions on their main pages due to years of editing and updating content. Revisions can be helpful in case you need to revert back to a previous version of your content, but they can also hurt the performance if there are too many. You can clean up all the post revisions in the WordPress database by executing the following command:

```
$ wp post delete $(wp post list --post_type='revision' --format=ids) --force
```

Copy

The command enclosed in the parenthesis is evaluated first and it will produce the `ids` of all the post revisions that are present passing them to the `delete` subcommand. The `--force` flag is necessary because posts of type `'revision'` do not support being sent to trash.

Step 5 – Managing Your Database

One of the most useful features of WP-CLI is its ability to interact with the MySQL database. For example, if you need an interactive session, you can enter a MySQL prompt with the following command:

```
$ wp db cli
```

Copy

You can then use the MySQL shell as you normally would and, once you are through with your tasks, exit the shell by typing `exit`.

For one-off queries, you can use the `wp db query` command by passing a valid SQL query as an argument to the command. For example, to list all the registered users in the WordPress database, you could run:

```
$ wp db query "SELECT user_login,ID FROM wp_users;"
```

Copy

You will be presented with an output similar to the following:

```
Output
+------------+----+
| user_login | ID |
+------------+----+
| admin      |  1 |
+------------+----+
```

With `wp db query` you can run any one-off SQL query for the WordPress database.

Backing Up and Restoring

WP-CLI also allows you to back up your WordPress database. Running this following command will place a SQL dump file in the current directory. This file contains your entire WordPress database including your posts, pages, user accounts, menus, and so on:

```
wp db export
```

Once the file is produced, you can move it to a different location for safekeeping:

```
Output
Success: Exported to 'wordpress-2021-01-25-25618e7.sql'.
```

You can also import a SQL dump file into your database through the `wp db import` command. This is useful when you are migrating a WordPress website from one location to another.

```
$ wp db import file.sql
```
Copy

Searching and Replacing

Another common operation you can perform with WP-CLI is a find-and-replace operation. You can make a dry run first to find out how many instances it would modify. The first string is the search component while the second is the replacement:

```
$ wp search-replace --dry-run 'example.com' 'example.net'
```
Copy

After running this, your output would be similar to the following:

```
Output
Success: 10 replacements to be made.
```

Once you are sure you want to proceed, remove the `--dry-run` flag from the previous command:

```
$ wp search-replace 'example.com' 'example.net'
```
Copy

In this step, you've reviewed several database operations that you can perform using WP-CLI. You can also complete other operations, such as optimizing the database, viewing database tables, deleting a database, or resetting one. You can explore the other options under the `wp db` subcommand by typing `wp help db` in your terminal.

Step 6 – Updating WordPress

You can update the core WordPress file with WP-CLI. You can examine the current version of WordPress that you have installed by running:

```
wp core version
```

```
Output
5.6
```

You can check for updates through the `wp core check-update` command. If your version is not the latest, this will produce an output similar to the following:

```
Output
+---------+-------------+-------------------------------------------------------------------------+
| version | update_type | package_url                                                             |
+---------+-------------+-------------------------------------------------------------------------+
| 5.6.1   | minor       | https://downloads.wordpress.org/release/wordpress-5.6.1-partial-0.zip |
+---------+-------------+-------------------------------------------------------------------------+
```

If an update is available, you can install it with:

```
$ wp core update
```

Copy

Git Command

Step 1- Go to Project folder

2. git init
3. git remote add origin https://username@gitlab.com/riteshj/nhs-inform.git
4. git branch -M main
5. git clone clone url - ask username/password
6. git fetch (Showing all branch)
7. git checkout dev
8. Git create your developement branch using command "git checkout -b dev-name"

GIT push command

1. git status
2. git add .
3. git status
4. git commit -m "SSD Module landing, details , filter, pages completed"
5. git pull origin dev
6. git push origin dev-rajive

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/rajivqx/test.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/rajivqx/test.git
git branch -M main
git push -u origin main
```

Git Comman Command

Git is a powerful version control system used for tracking changes in source code during software developmentInitialization:

- `git init`: Initialize a new Git repository in the current directory.

2. **Cloning:**
   - `git clone <repository_url>`: Clone a remote repository to your local machine.
3. **Configuration:**
   - `git config --global user.name "Your Name"`: Set your global username.
   - `git config --global user.email "youremail@example.com"`: Set your global email address.
4. **Basic Workflow:**
   - `git status`: Check the status of your working directory and see which files have changed.
   - `git add <file>`: Stage changes for commit.
   - `git commit -m "Commit message"`: Commit staged changes with a message.
   - `git diff`: Show the differences between the working directory and the last commit.
5. **Branching and Merging:**
   - `git branch`: List all branches in the repository.
   - `git branch <branch_name>`: Create a new branch.
   - `git checkout <branch_name>`: Switch to a different branch.
   - `git merge <branch_name>`: Merge changes from one branch into another.
   - `git branch -d <branch_name>`: Delete a branch (use `-D` for a force delete).
6. **Remote Repositories:**
   - `git remote`: List remote repositories.
   - `git remote -v`: List remote repositories with URLs.
   - `git push <remote_name> <branch_name>`: Push your changes to a remote repository.
   - `git pull <remote_name> <branch_name>`: Pull changes from a remote repository.
7. **Tagging:**
   - `git tag`: List all tags in the repository.
   - `git tag <tag_name>`: Create a new tag at the current commit.
   - `git tag -a <tag_name> -m "Tag message"`: Create an annotated tag with a message.
8. **History:**
   - `git log`: View commit history.
   - `git log --oneline`: View a concise commit history.
   - `git blame <file>`: See who last modified each line of a file.
9. **Undoing Changes:**
   - `git reset <commit>`: Unstage changes but keep modifications.
   - `git reset --hard <commit>`: Discard changes and move the branch to a specific commit.

- `git revert <commit>`: Create a new commit that undoes the changes introduced by a specific commit.
- `git checkout -- <file>`: Discard changes in a specific file.

10. **Stashing:**
    - `git stash`: Temporarily save changes that are not ready to be committed.
    - `git stash pop`: Apply the most recent stash and remove it from the stash list.
11. **Advanced Topics:**
    - `git submodule`: Manage Git submodules within your repository.
    - `git rebase`: Reorganize and rewrite commit history.
    - `git cherry-pick`: Apply specific commits to your current branch.

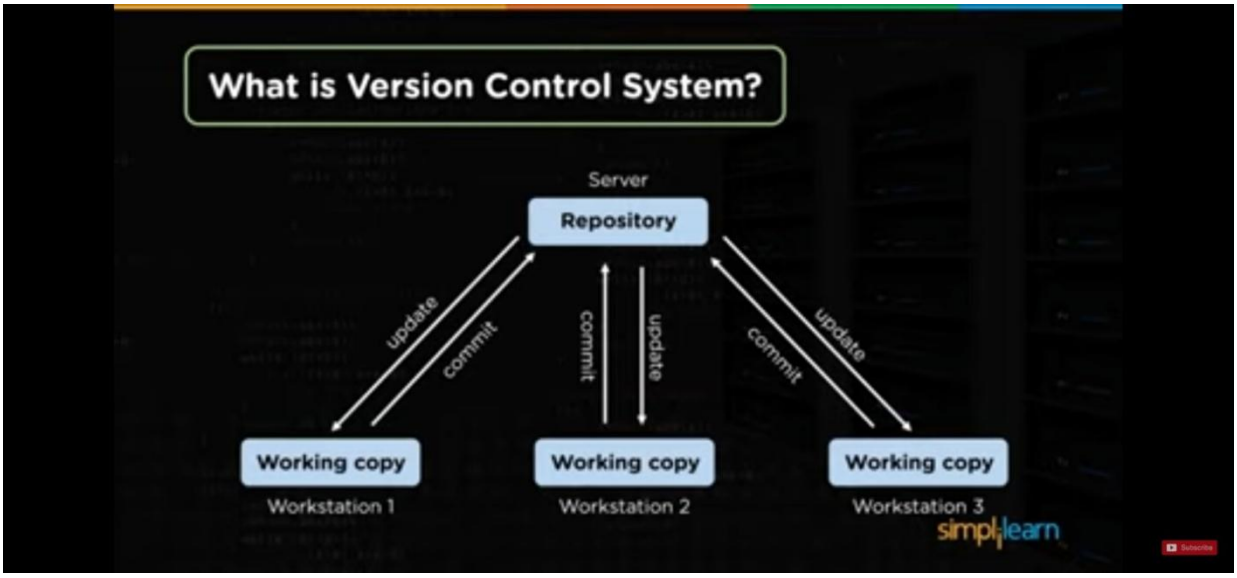
`git help <command>` or `git <command> --help` to access the official Git documentation.

Linex Comman Command

1. **File and Directory Operations:**
    - `ls`: List files and directories.
    - `pwd`: Print the current working directory.
    - `cd`: Change the current directory.
    - `touch`: Create an empty file.
    - `mkdir`: Create a new directory.
    - `rmdir`: Remove a directory.
    - `rm`: Remove files or directories.
    - `cp`: Copy files or directories.
    - `mv`: Move or rename files or directories.
2. **File Viewing and Editing:**
    - `cat`: Concatenate and display file content.
    - `more` or `less`: View text files page by page.
    - `head` and `tail`: Display the beginning or end of a file.
    - `nano` or `vim` (or `vi`): Text editors for creating and editing files.
    - `grep`: Search for text patterns in files.
    - `find`: Search for files and directories.
3. **File Permissions:**
    - `chmod`: Change file permissions.
    - `chown`: Change file ownership.
    - `chgrp`: Change group ownership.
4. **System Information:**
    - `top`: Display real-time system information and processes.
    - `df`: Display disk space usage.
    - `free`: Display RAM usage.
    - `uname`: Display system information.

- `lscpu`: Display CPU information.

5. **Package Management:**
   - `apt` (Debian/Ubuntu): Package management tool for installing, updating, and removing software packages.
   - `yum` (RHEL/CentOS): Package manager for Red Hat-based distributions.
   - `dnf` (Fedora): Package manager for Fedora.
6. **Network Utilities:**
   - `ping`: Send ICMP echo requests to a host.
   - `ifconfig` or `ip`: Network configuration and information.
   - `netstat`: Network statistics and routing tables.
   - `ssh`: Secure Shell for remote login.
   - `scp`: Securely copy files between hosts.
7. **Process Management:**
   - `ps`: Display information about running processes.
   - `kill`: Terminate processes.
   - `top`: Monitor and manage running processes interactively.
8. **Archiving and Compression:**
   - `tar`: Create or extract tar archives.
   - `zip` and `unzip`: Create and extract zip archives.
   - `gzip`, `gunzip`, `bzip2`, `bunzip2`: Compress and decompress files.
9. **User and Group Management:**
   - `useradd` and `userdel`: Add and delete user accounts.
   - `passwd`: Change user passwords.
   - `groupadd` and `groupdel`: Add and delete groups.
   - `su` and `sudo`: Switch users or execute commands as superuser.
10. **File System Navigation:**
    - `cd`: Change directory.
    - `ls`: List files and directories.
    - `pwd`: Print working directory.
    - `tree`: Display directory structure as a tree.


What is Version Control System?
Server
Repository
update
commit
commit
update
commit
update
Working copy
Workstation 1
Working copy
Workstation 2
Working copy
Workstation 3
simplilearn


Why use Git?
WSCUBE TECH
System For Satisfaction
• Super-fast and efficient performance.
• Cross-platform
• Code changes can be very easily and clearly tracked.
• Easily maintainable and robust.
• Offers an amazing command line utility known as git bash.
• Also offers GIT GUI where you can very quickly re-scan, state change, sign off, commit & push the code quickly with just a few clicks.
opment Online Training? Call us at : +91 9024244886/ +91 9269698122 or visit www.wscubetech.com
SUBSCRIBE NOW

```
git config --global user.name "My Name"

git config --global user.email "someone@email.com"

git config --list
```

Clone - Cloning a repository on our local machine

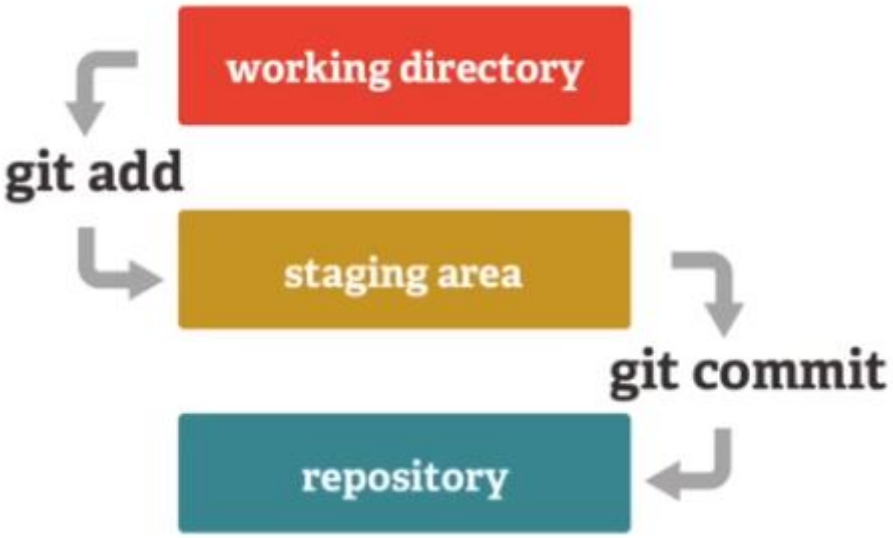
```
git clone <- some link ->
```

Push Command

push - upload local repo content to remote repo

```
git push origin main
```

```
git init

git remote add origin <- link ->

git remote -v      (to verify remote)

git branch         (to check branch)

git branch -M main     (to rename branch)

git push origin main
```



```
nothing added to commit but untracked files present (use "git add" to track)
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git add Sum.java
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Sum.java
```

```
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git commit -m "initial commit"
[master (root-commit) d7cf4cb] initial commit
 1 file changed, 10 insertions(+)
 create mode 100644 Sum.java
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git log
commit d7cf4cb54a9efd687e33ebc7c20b1da7706fd7e8 (HEAD -> master)
Author: Anuj Kumar Sharma <anuj55149@gmail.com>
Date:   Mon May 31 02:07:16 2021 +0530
```

```
cmd + k / clear
```

```
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git log
commit 7f5ee925ab92b3f9a8a5e576f7bf1091042bb968 (HEAD -> master)
Author: Anuj Kumar Sharma <anuj55149@gmail.com>
Date:   Mon May 31 02:12:32 2021 +0530

    adding Diff.java

commit d7cf4cb54a9efd687e33ebc7c20b1da7706fd7e8
Author: Anuj Kumar Sharma <anuj55149@gmail.com>
Date:   Mon May 31 02:07:16 2021 +0530

    initial commit
mzaanuj@mac ~/D/V/LetsLearnGit (master)>
```

```
no changes added to commit (use "git add" and/or "git commit -a")
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git add .
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git commit -m "added message in Sum.java
"
```

```
commit d7cf4cb54a9efd687e33ebc7c20b1da7706fd7e8
Author: Anuj Kumar Sharma <anuj55149@gmail.com>
Date:   Mon May 31 02:07:16 2021 +0530

    initial commit
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git checkout d7cf4cb54a9efd687e33ebc7c20
b1da7706fd7e8
```

```
mzaanuj@mac ~/D/V/LetsLearnGit ((d7cf4cb5…))> git branch
* (HEAD detached at d7cf4cb)
  master
mzaanuj@mac ~/D/V/LetsLearnGit ((d7cf4cb5…))> git log
commit d7cf4cb54a9efd687e33ebc7c20b1da7706fd7e8 (HEAD)
Author: Anuj Kumar Sharma <anuj55149@gmail.com>
Date:   Mon May 31 02:07:16 2021 +0530

    initial commit
mzaanuj@mac ~/D/V/LetsLearnGit ((d7cf4cb5…))> git checkout master
Previous HEAD position was d7cf4cb initial commit
Switched to branch 'master'
mzaanuj@mac ~/D/V/LetsLearnGit (master)>
```

```
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git branch dev
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git branch
  dev
* master
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git checkout dev
Switched to branch 'dev'
mzaanuj@mac ~/D/V/LetsLearnGit (dev)>
```

git checkout -b <branch name>

```
mzaanuj@mac ~/D/V/LetsLearnGit (dev)> git checkout -b anuj/multiply
Switched to a new branch 'anuj/multiply'
mzaanuj@mac ~/D/V/LetsLearnGit (anuj/multiply)>
```

```
mzaanuj@mac ~/D/V/LetsLearnGit (anuj/multiply)> git checkout dev
Switched to branch 'dev'
mzaanuj@mac ~/D/V/LetsLearnGit (dev)> git merge anuj/multiply
Updating 092be9a..da8e85d
Fast-forward
 Multiply.java | 10 ++++++++++
 1 file changed, 10 insertions(+)
 create mode 100644 Multiply.java
```

```
mzaanuj@mac ~/D/V/LetsLearnGit (master)> touch .gitignore
```

...or create a new repository on the command line

```
echo "# Lets-Learn-Git" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/Anuj-Kumar-Sharma/Lets-Learn-Git.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/Anuj-Kumar-Sharma/Lets-Learn-Git.git
git branch -M master
git push -u origin master
```

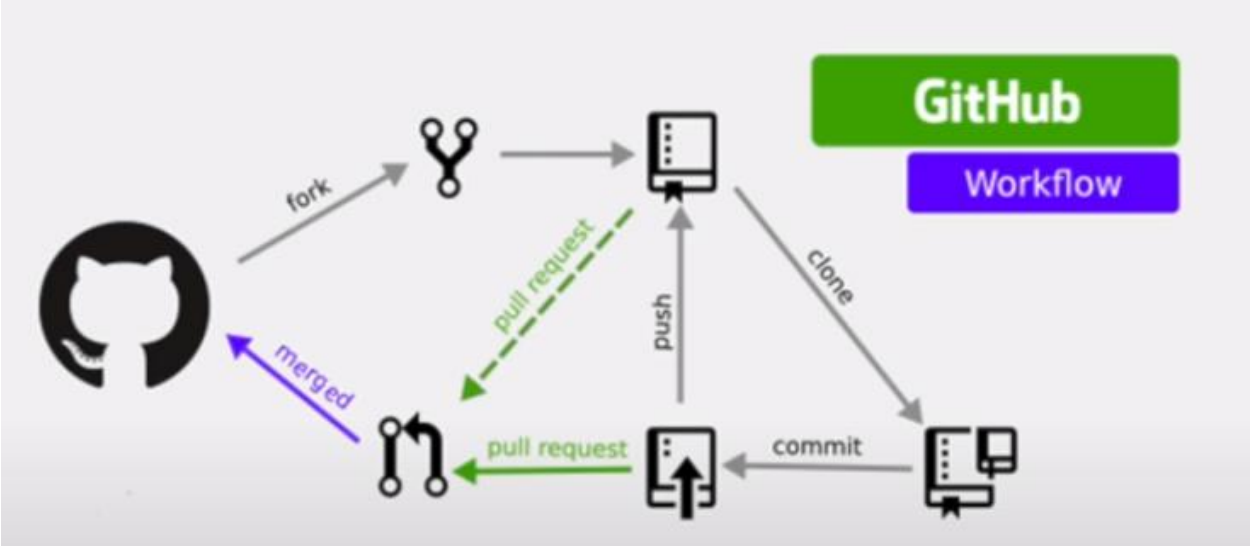
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

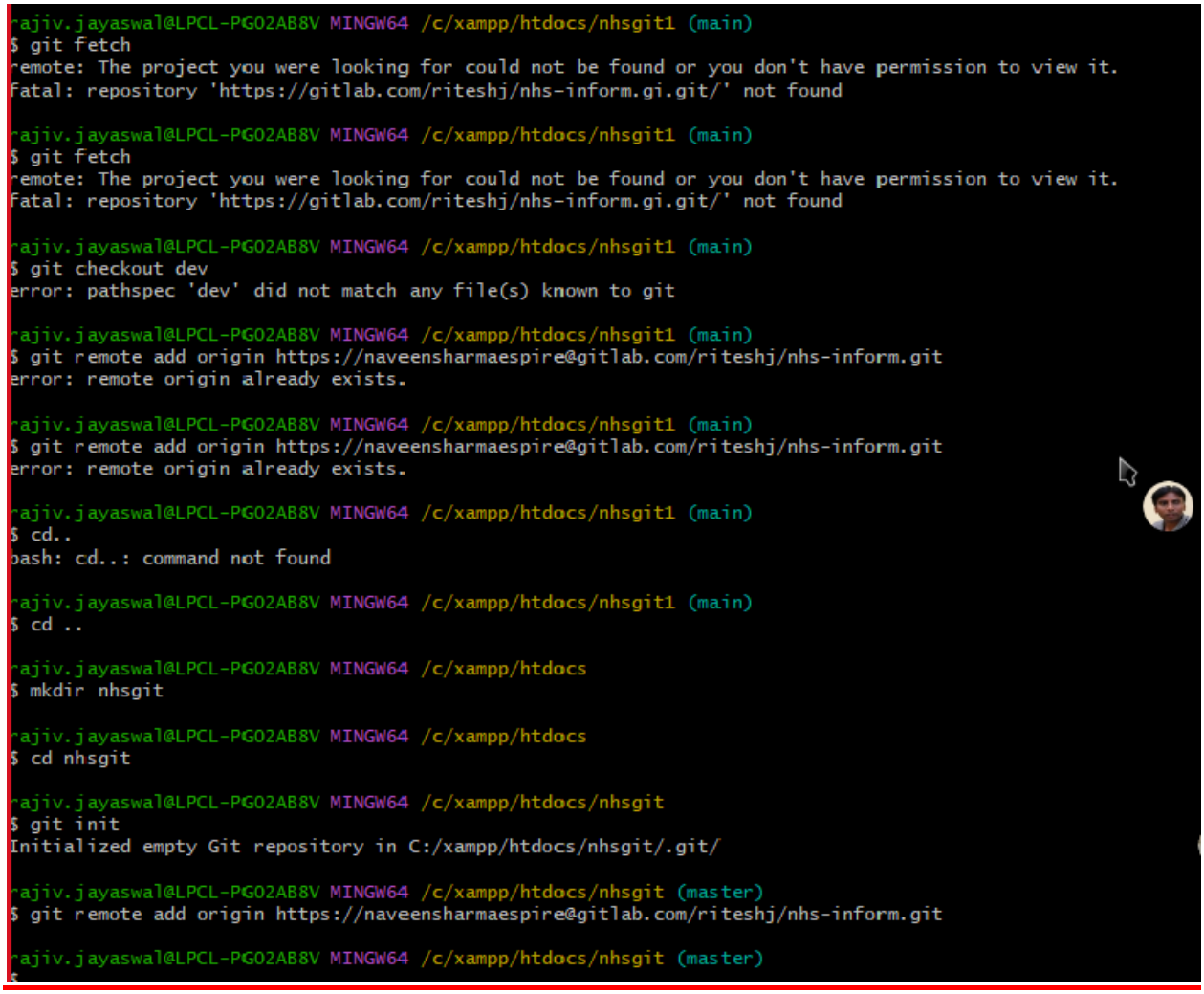
Import code

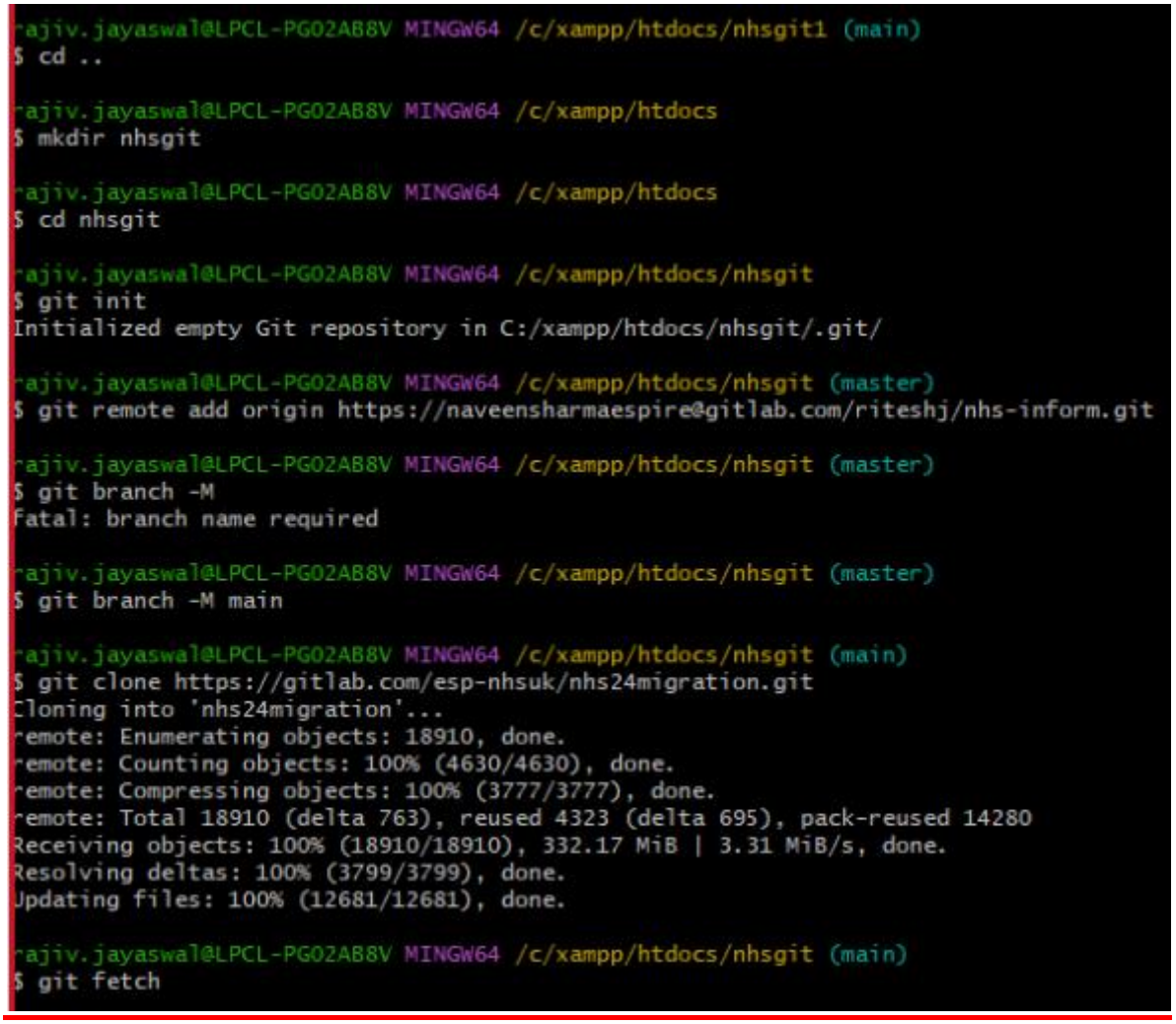
```
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git remote -v
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git remote add origin https://github.com
/Anuj-Kumar-Sharma/Lets-Learn-Git.git
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git remote -v
origin  https://github.com/Anuj-Kumar-Sharma/Lets-Learn-Git.git (fetch)
origin  https://github.com/Anuj-Kumar-Sharma/Lets-Learn-Git.git (push)
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git branch -M master
mzaanuj@mac ~/D/V/LetsLearnGit (master)> git push -u origin master
```

```
mzaanuj@mac ~/D/V/LetsLearnGit (master) [128]> git push -u origin master
Username for 'https://github.com': Anuj-Kumar-Sharma
Password for 'https://Anuj-Kumar-Sharma@github.com':
```




```
MINGW64:/c/xampp/htdocs/nhsgit1

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit1
$ git init
Initialized empty Git repository in C:/xampp/htdocs/nhsgit1/.git/

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (master)
$ git remote add origin https://naveensharmaespire@gitlab.com/riteshj/nhs-inform
.gi

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (master)
$ git branch -M main

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ git clone https://gitlab.com/esp-nhsuk/nhs24migration.git
Cloning into 'nhs24migration'...
remote: Enumerating objects: 18910, done.
remote: Counting objects: 100% (4630/4630), done.
remote: Compressing objects: 100% (3777/3777), done.
remote: Total 18910 (delta 763), reused 4323 (delta 695), pack-reused 14280
Receiving objects: 100% (18910/18910), 332.17 MiB | 3.34 MiB/s, done.
Resolving deltas: 100% (3799/3799), done.
Updating files: 100% (12681/12681), done.

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ git
```


rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ git fetch
remote: The project you were looking for could not be found or you don't have permission to view it.
fatal: repository 'https://gitlab.com/riteshj/nhs-inform.gi.git/' not found
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ git fetch
remote: The project you were looking for could not be found or you don't have permission to view it.
fatal: repository 'https://gitlab.com/riteshj/nhs-inform.gi.git/' not found
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ git checkout dev
error: pathspec 'dev' did not match any file(s) known to git
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ git remote add origin https://naveensharmaespire@gitlab.com/riteshj/nhs-inform.git
error: remote origin already exists.
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ git remote add origin https://naveensharmaespire@gitlab.com/riteshj/nhs-inform.git
error: remote origin already exists.
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ cd..
bash: cd..: command not found
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ cd ..
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs
$ mkdir nhsgit
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs
$ cd nhsgit
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit
$ git init
Initialized empty Git repository in C:/xampp/htdocs/nhsgit/.git/
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit (master)
$ git remote add origin https://naveensharmaespire@gitlab.com/riteshj/nhs-inform.git
rajiv.jayaswal@LPCL-PGO2AB8V MINGW64 /c/xampp/htdocs/nhsgit (master)
$


rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit1 (main)
$ cd ..
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs
$ mkdir nhsgit
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs
$ cd nhsgit
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit
$ git init
Initialized empty Git repository in C:/xampp/htdocs/nhsgit/.git/
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (master)
$ git remote add origin https://naveensharmaespire@gitlab.com/riteshj/nhs-inform.git
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (master)
$ git branch -M
fatal: branch name required
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (master)
$ git branch -M main
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (main)
$ git clone https://gitlab.com/esp-nhsuk/nhs24migration.git
Cloning into 'nhs24migration'...
remote: Enumerating objects: 18910, done.
remote: Counting objects: 100% (4630/4630), done.
remote: Compressing objects: 100% (3777/3777), done.
remote: Total 18910 (delta 763), reused 4323 (delta 695), pack-reused 14280
Receiving objects: 100% (18910/18910), 332.17 MiB | 3.31 MiB/s, done.
Resolving deltas: 100% (3799/3799), done.
Updating files: 100% (12681/12681), done.
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (main)
$ git fetch

```
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (master)
$ git remote add origin https://naveensharmaespire@gitlab.com/riteshj/nhs-inform.git

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (master)
$ git branch -M
fatal: branch name required

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (master)
$ git branch -M main

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (main)
$ git clone https://gitlab.com/esp-nhsuk/nhs24migration.git
Cloning into 'nhs24migration'...
remote: Enumerating objects: 18910, done.
remote: Counting objects: 100% (4630/4630), done.
remote: Compressing objects: 100% (3777/3777), done.
remote: Total 18910 (delta 763), reused 4323 (delta 695), pack-reused 14280
Receiving objects: 100% (18910/18910), 332.17 MiB | 3.31 MiB/s, done.
Resolving deltas: 100% (3799/3799), done.
Updating files: 100% (12681/12681), done.

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (main)
$ git fetch
remote: Enumerating objects: 13434, done.
remote: Counting objects: 100% (13271/13271), done.
remote: Compressing objects: 100% (9536/9536), done.
remote: Total 13434 (delta 3489), reused 12803 (delta 3301), pack-reused 163
Receiving objects: 100% (13434/13434), 62.53 MiB | 3.21 MiB/s, done.
Resolving deltas: 100% (3505/3505), done.
From https://gitlab.com/riteshj/nhs-inform
 * [new branch]      dev              -> origin/dev
 * [new branch]      dev-fahimsidiqui -> origin/dev-fahimsidiqui
 * [new branch]      dev-fraheem      -> origin/dev-fraheem
 * [new branch]      dev-mumtaj       -> origin/dev-mumtaj
 * [new branch]      dev-naveen       -> origin/dev-naveen
 * [new branch]      dev-ritesh       -> origin/dev-ritesh
 * [new branch]      main             -> origin/main

rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (main)
$ git checkout dev
Updating files: 100% (7678/7678), done.
Switched to a new branch 'dev'
branch 'dev' set up to track 'origin/dev'.
```


rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (main)
$ git checkout dev
Updating files: 100% (7678/7678), done.
Switched to a new branch 'dev'
branch 'dev' set up to track 'origin/dev'.
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (dev)
$ git checkout -b dev-rajive
Switched to a new branch 'dev-rajive'
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (dev-rajive)
$ cheout out dev
bash: cheout: command not found
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (dev-rajive)
$ git origin pull dev
git: 'origin' is not a git command. See 'git --help'.
rajiv.jayaswal@LPCL-PG02AB8V MINGW64 /c/xampp/htdocs/nhsgit (dev-rajive)
$ git origin dev
git: 'origin' is not a git command. See 'git --help'.

Agenda For The Session

i. **Software Development** Challenges **Agile*

ii. DevOps: **Need**, **Rise** & **Tools** involved

iii. **Git (SCM)**: Need, Working & Use-case

iv. **Selenium, TestNG & Maven (CT)**: Need & Working

v. **Jenkins (CI)**: Need, Working & Use-case

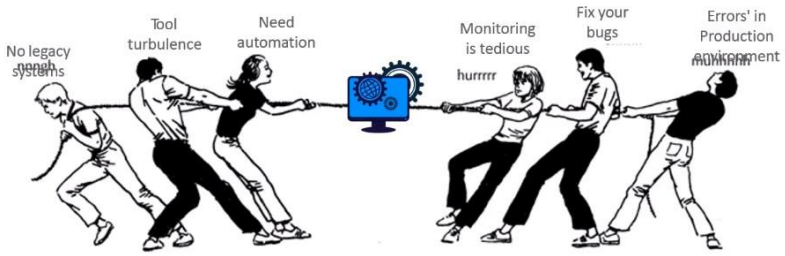
vi. **Docker (CD & Containers)**: Need & Working

vii. **Ansible (CD & CM)**: Need & Working

viii. **Structured DevOps Training** at ***Edureka***

Software Development Challenges

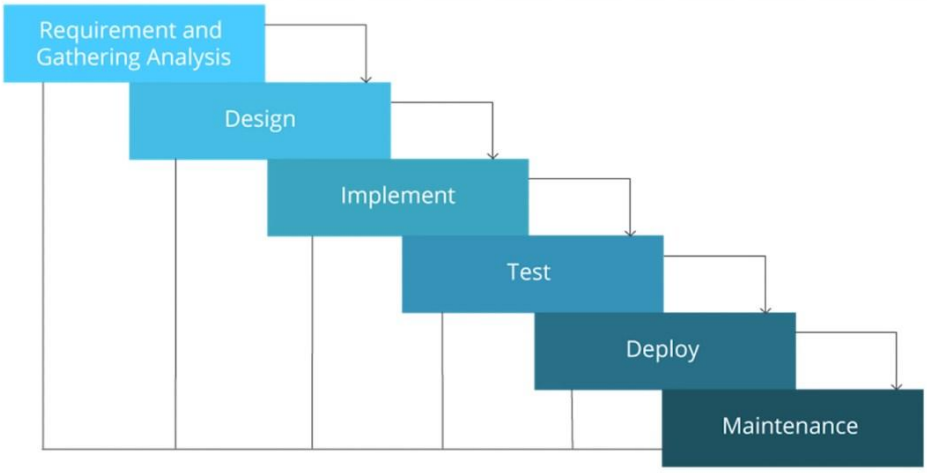
Primary factor leading to challenges during software development is the *Silo* between **development** & **operations**.



Topics for Today's Session

1. Why DevOps?
2. What is DevOps?
3. DevOps Stages
4. DevOps Tools
5. DevOps Use-Case

Traditional Waterfall Model



What is Agile Methodology?

In the Agile Methodology each project is broken up into several 'Iterations'

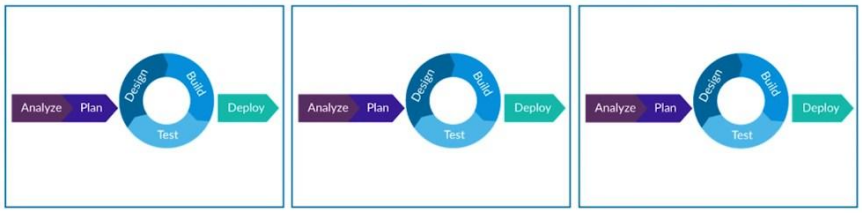
All Iterations should be of the same time duration (between 2 to 8 weeks)

At the end of each iteration, a working product should be delivered

Waterfall vs Agile

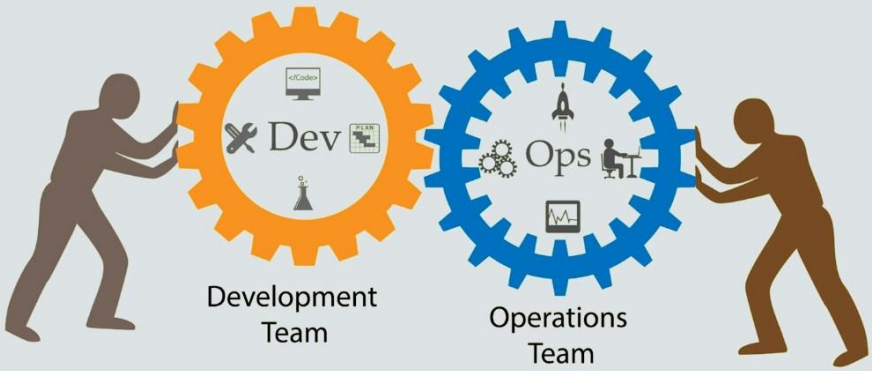
Waterfall

Agile

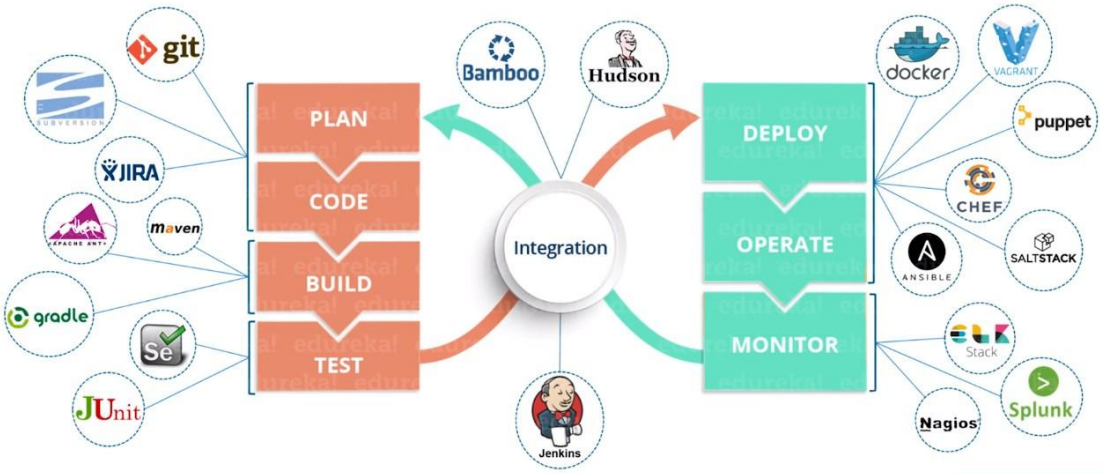


Limitations of Agile

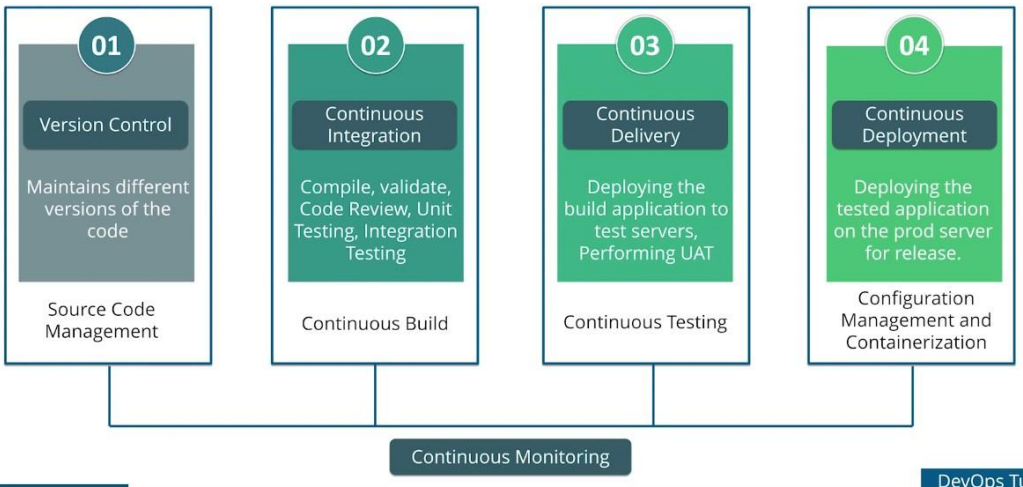
Solution is DevOps



What is DevOps?

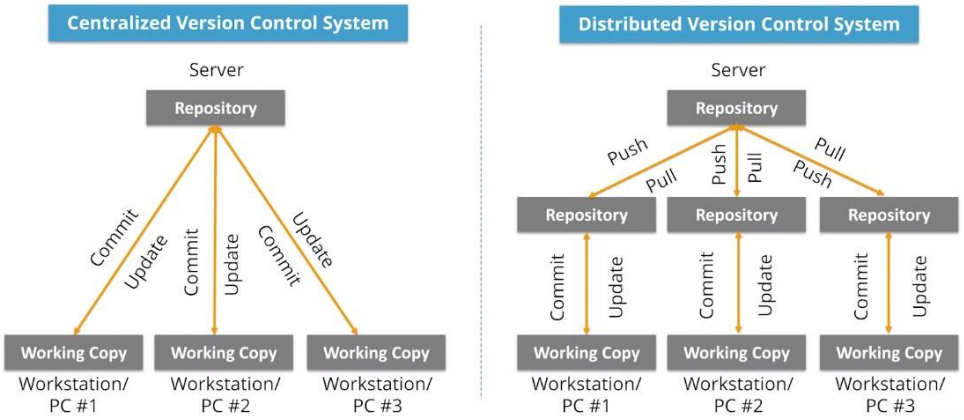


DevOps Stages



Source Code Management

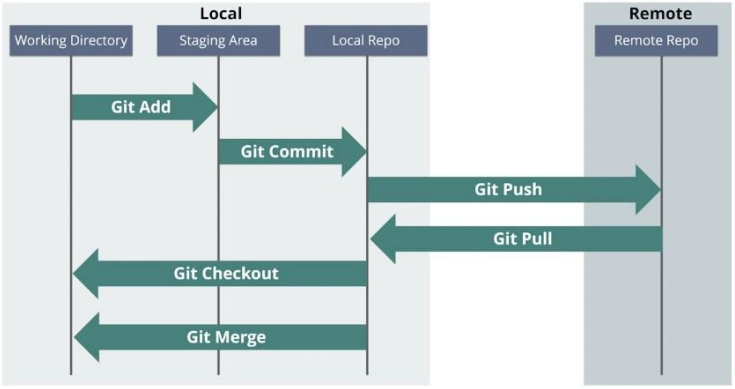
The management of changes to documents, computer programs, large websites and other collection of information



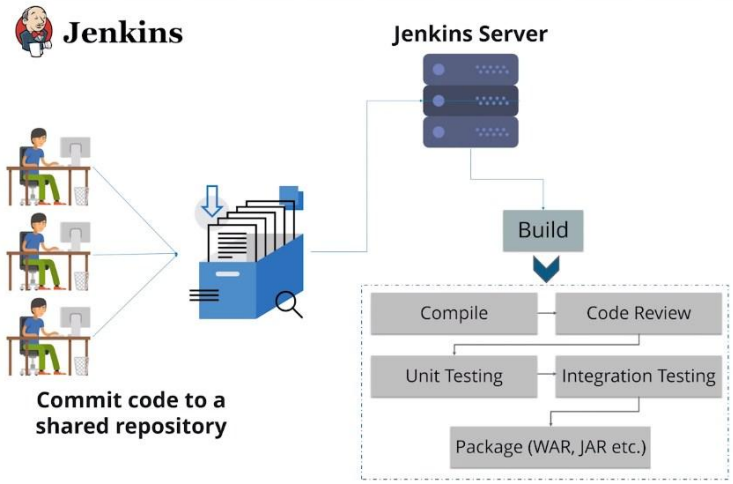
Source Code Management

git

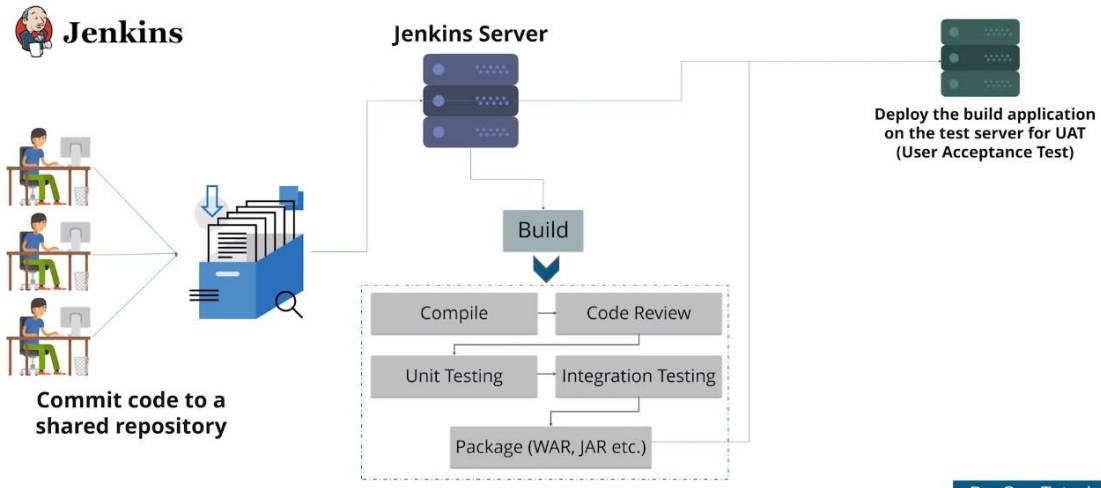
Git is a Distributed Version Control tool that supports distributed non-linear workflows by providing data assurance for developing quality software



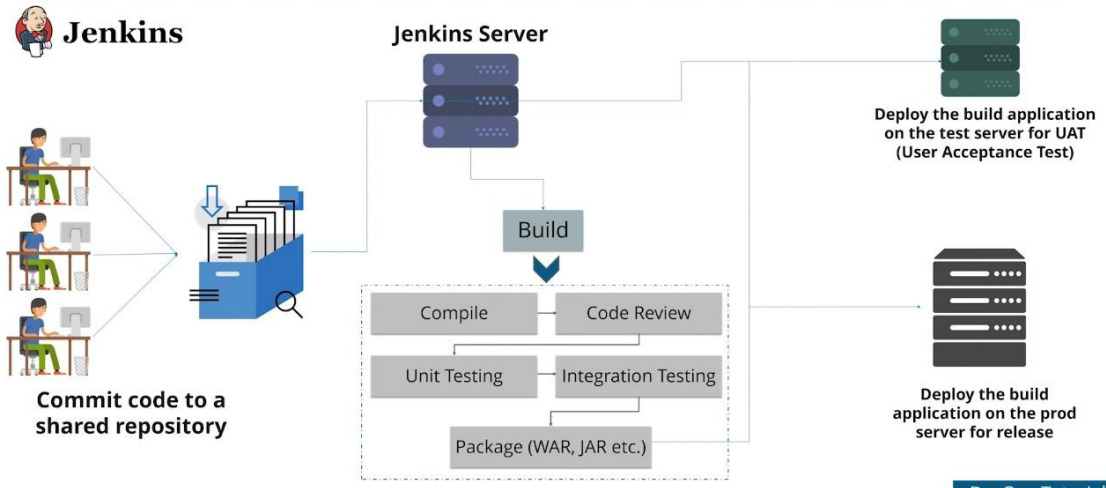
Continuous Integration



Continuous Delivery

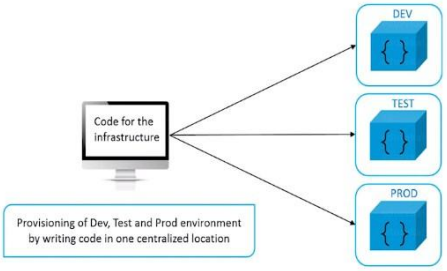


Continuous Deployment

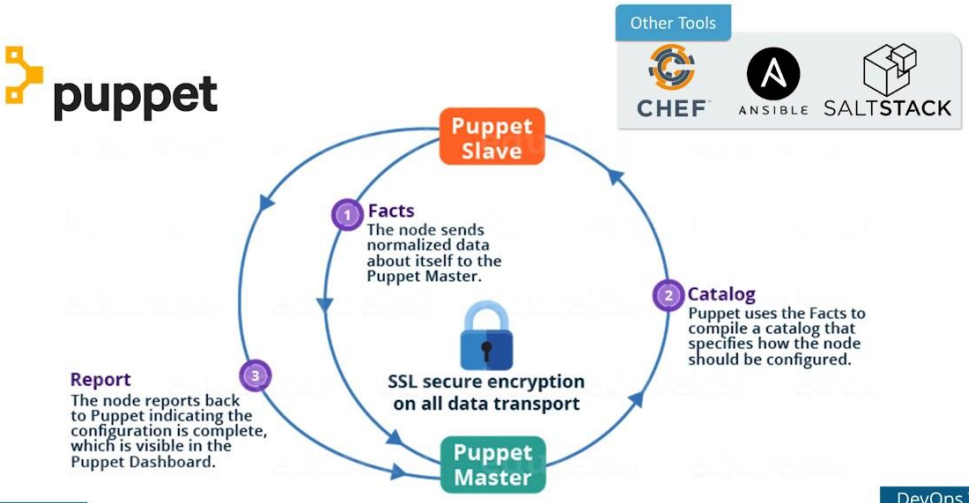


Configuration Management

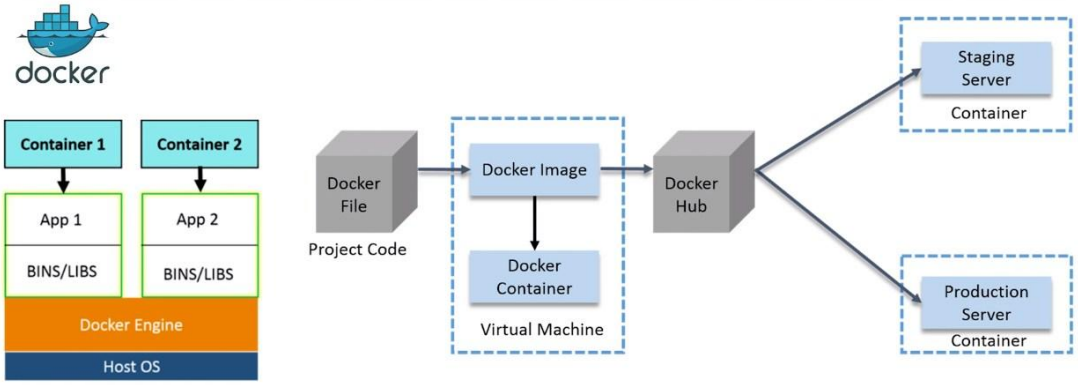
- Configuration Management is the practice of handling changes systematically so that a system maintains its integrity over time
- Configuration Management (CM) ensures that the current design and build state of the system is known, good & trusted
- It doesn't rely on the tacit knowledge of the development team



Configuration Management

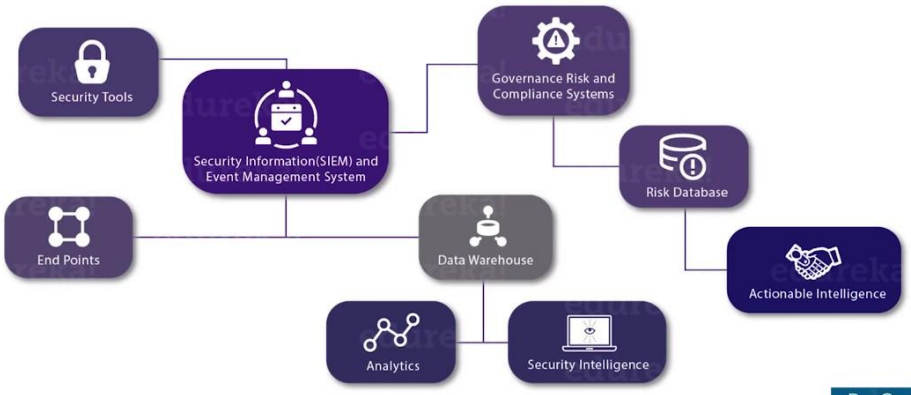


Containerization



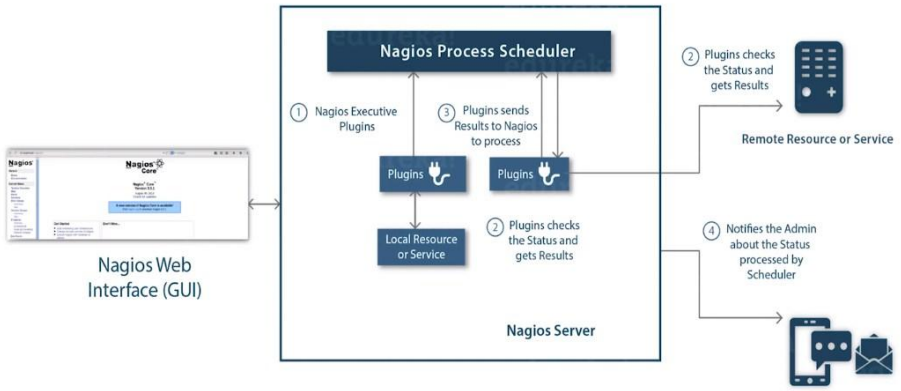
Continuous Monitoring

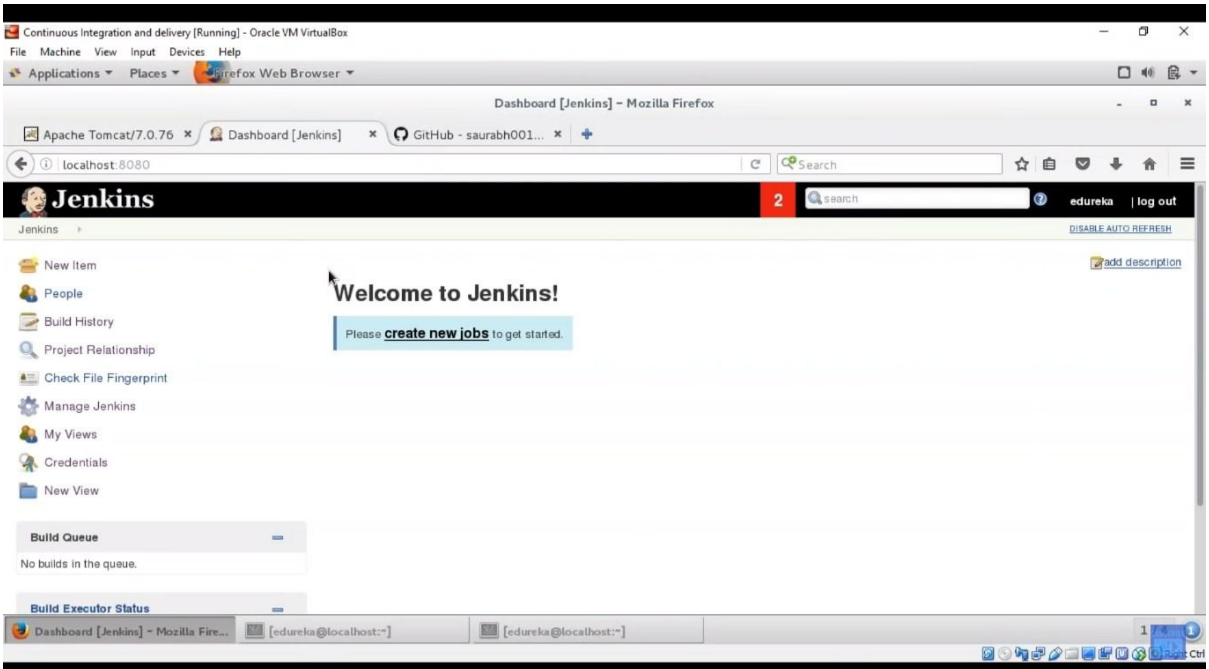
Continuous Monitoring is all about the ability of an organization to detect, report, respond, contain and mitigate the attacks that occur, in its infrastructure

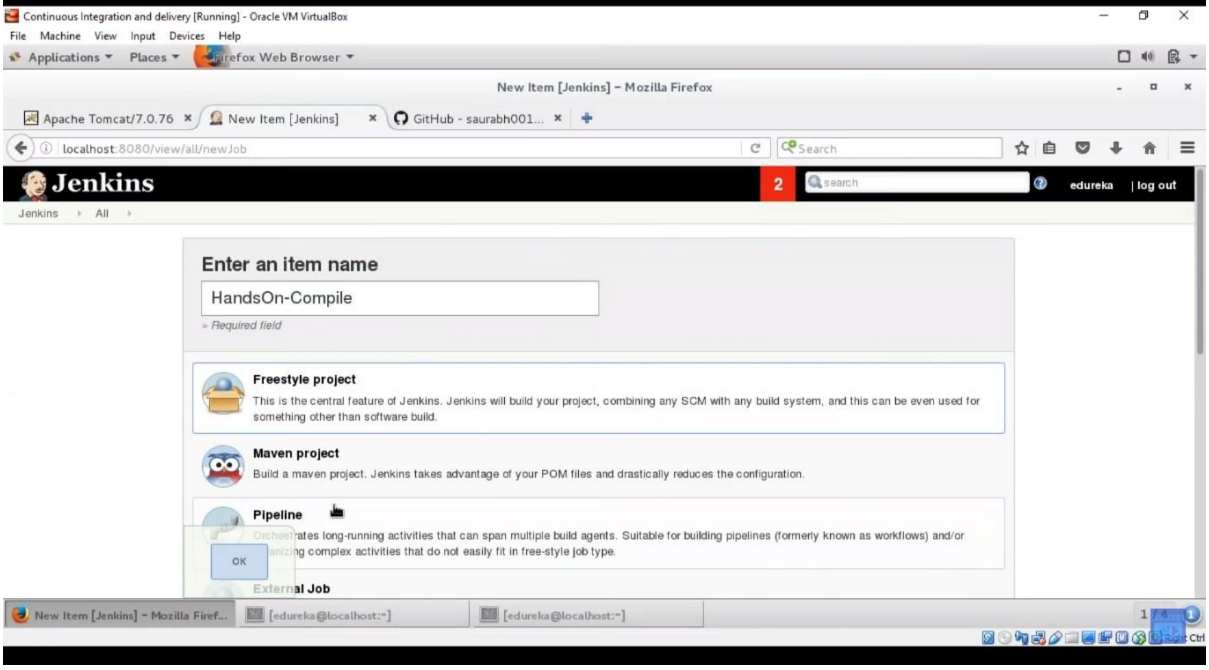


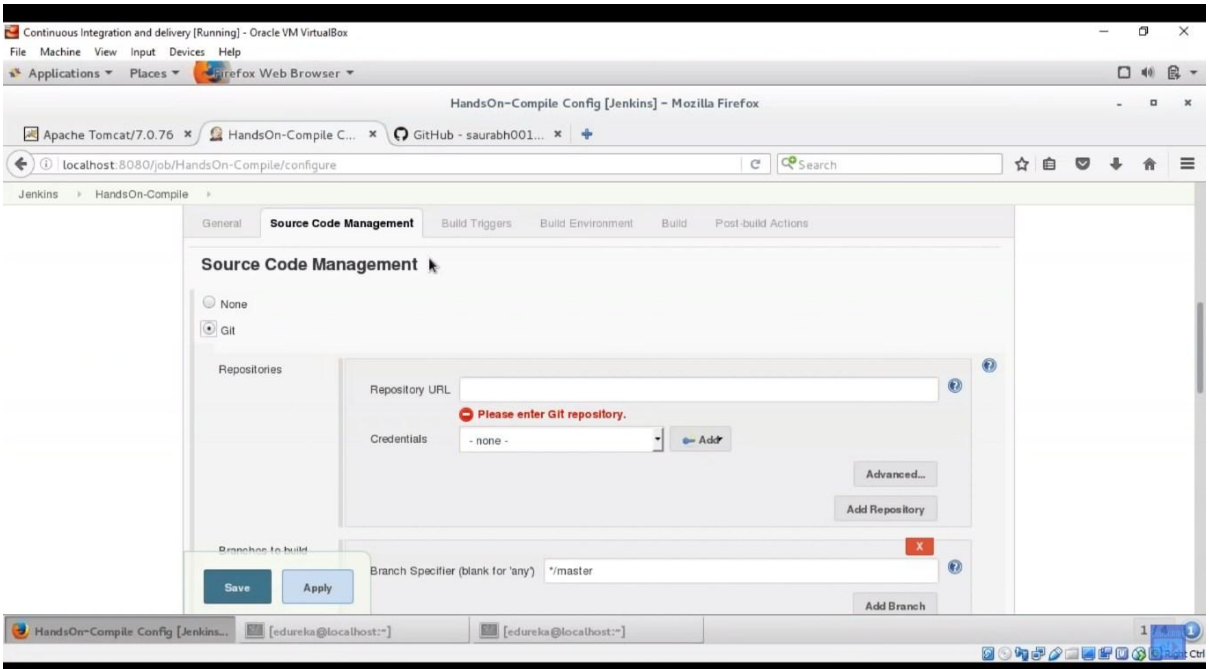
Continuous Monitoring

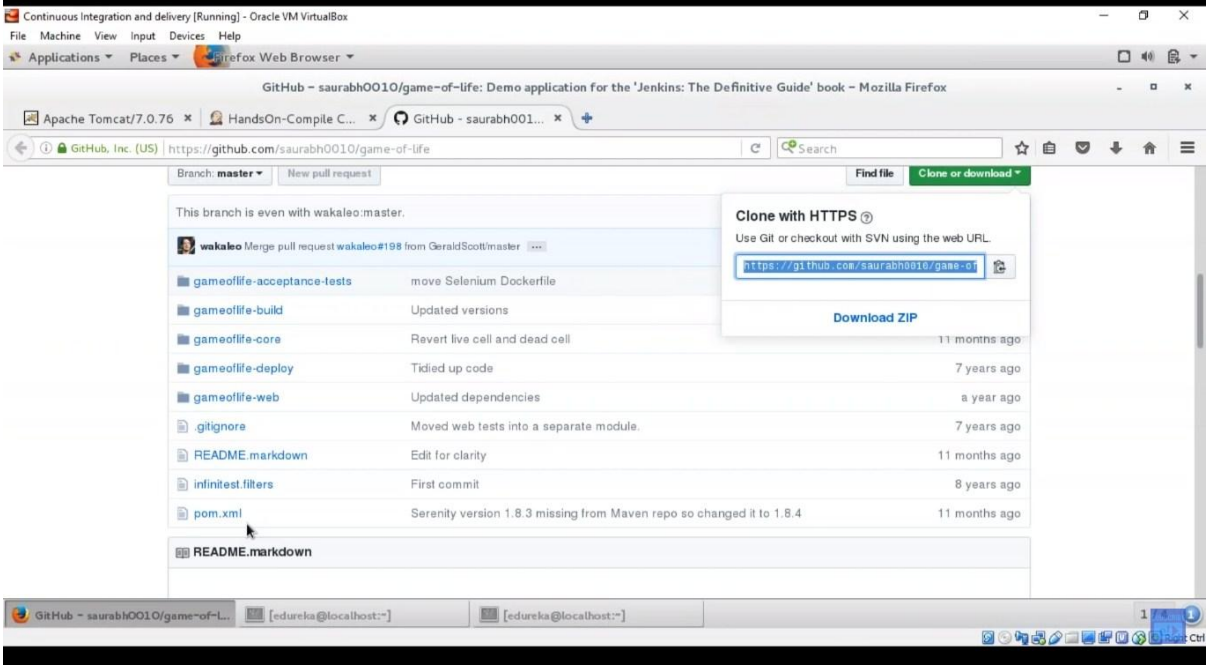
Nagios®

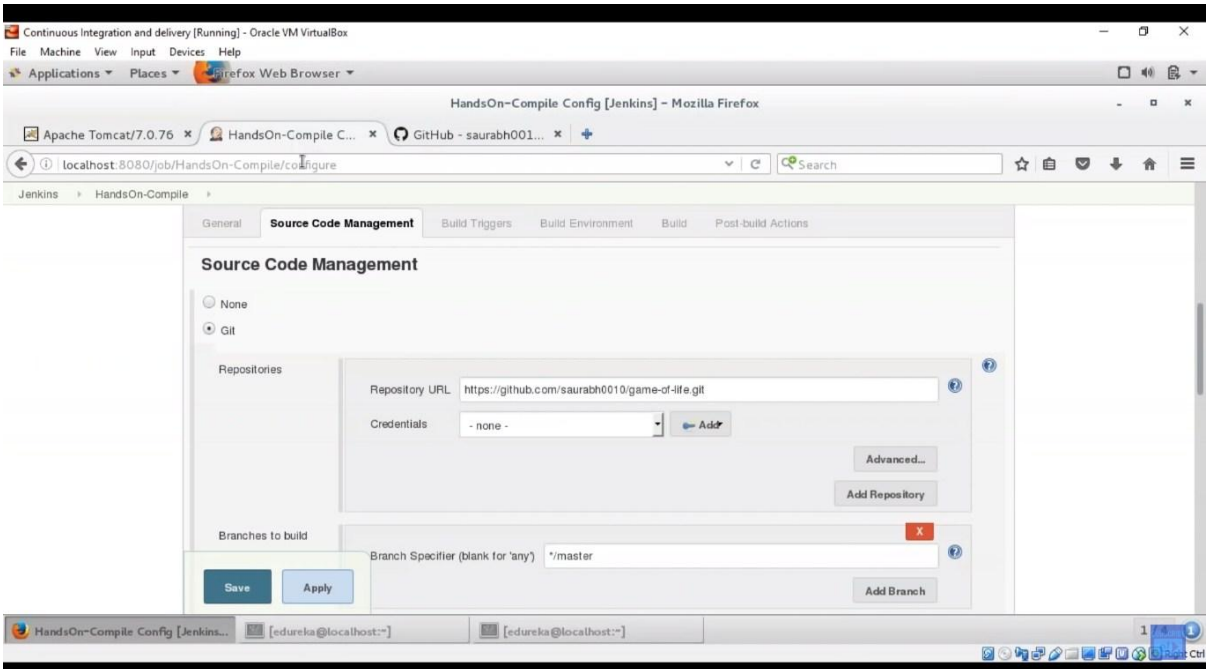


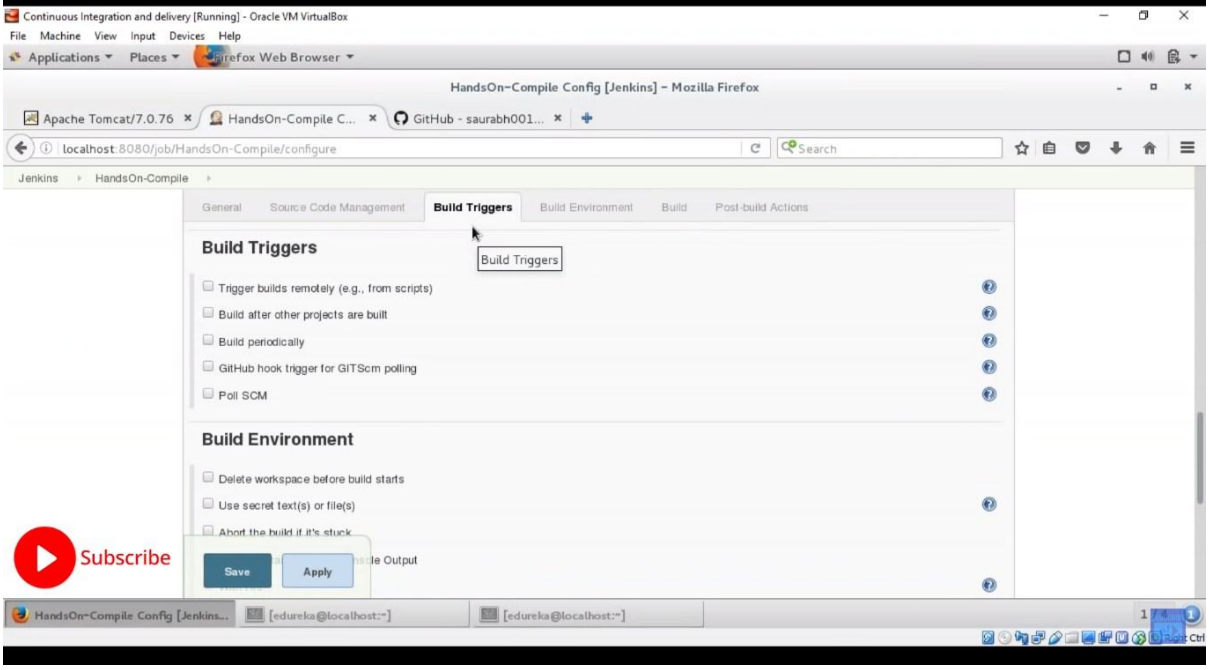

Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
Dashboard [Jenkins] – Mozilla Firefox
Apache Tomcat/7.0.76
Dashboard [Jenkins]
GitHub - saurabh001...
localhost:8080
Search
Jenkins
search
edureka | log out
Jenkins
DISABLE AUTO REFRESH
add description
New Item
People
Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Credentials
New View
Welcome to Jenkins!
Please create new jobs to get started.
Build Queue
No builds in the queue.
Build Executor Status
Dashboard [Jenkins] – Mozilla Fire...
[edureka@localhost:~]
[edureka@localhost:~]

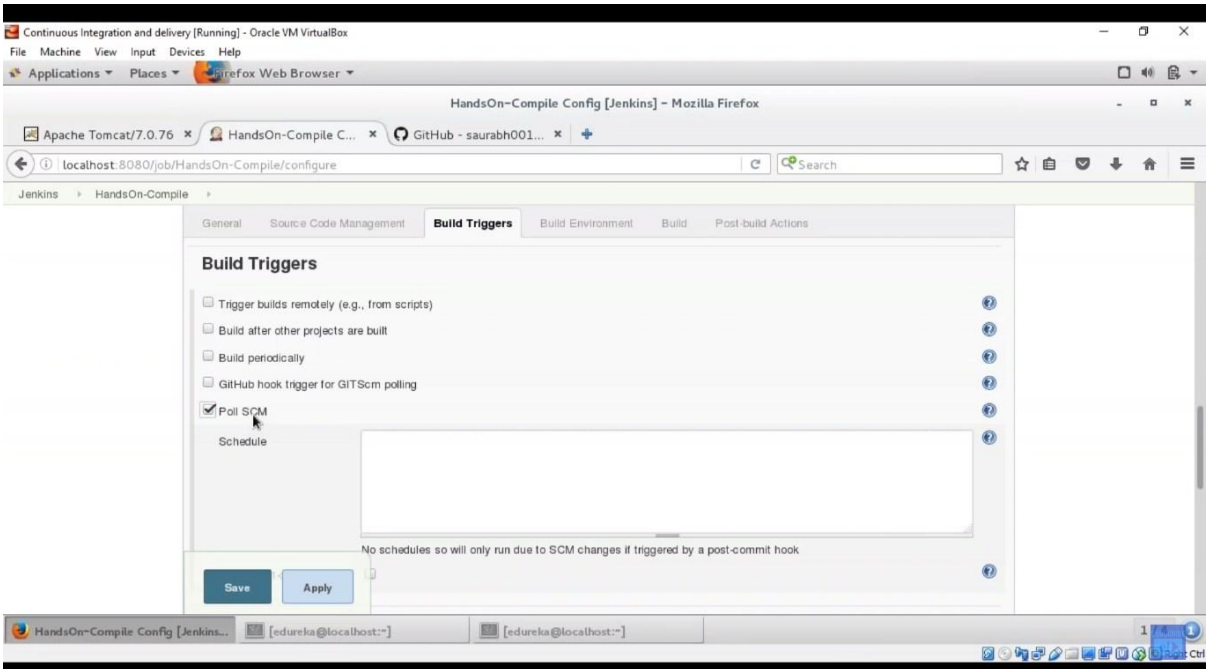

Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
New Item [Jenkins] – Mozilla Firefox
Apache Tomcat/7.0.76
New Item [Jenkins]
GitHub - saurabh001...
localhost:8080/view/all/newJob
Search
Jenkins
search
edureka | log out
Jenkins All
Enter an item name
HandsOn-Compile
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
Pipeline
ates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or
ng complex activities that do not easily fit in free-style job type.
OK
New Item [Jenkins] – Mozilla Firef...
[edureka@localhost:~]
[edureka@localhost:~]

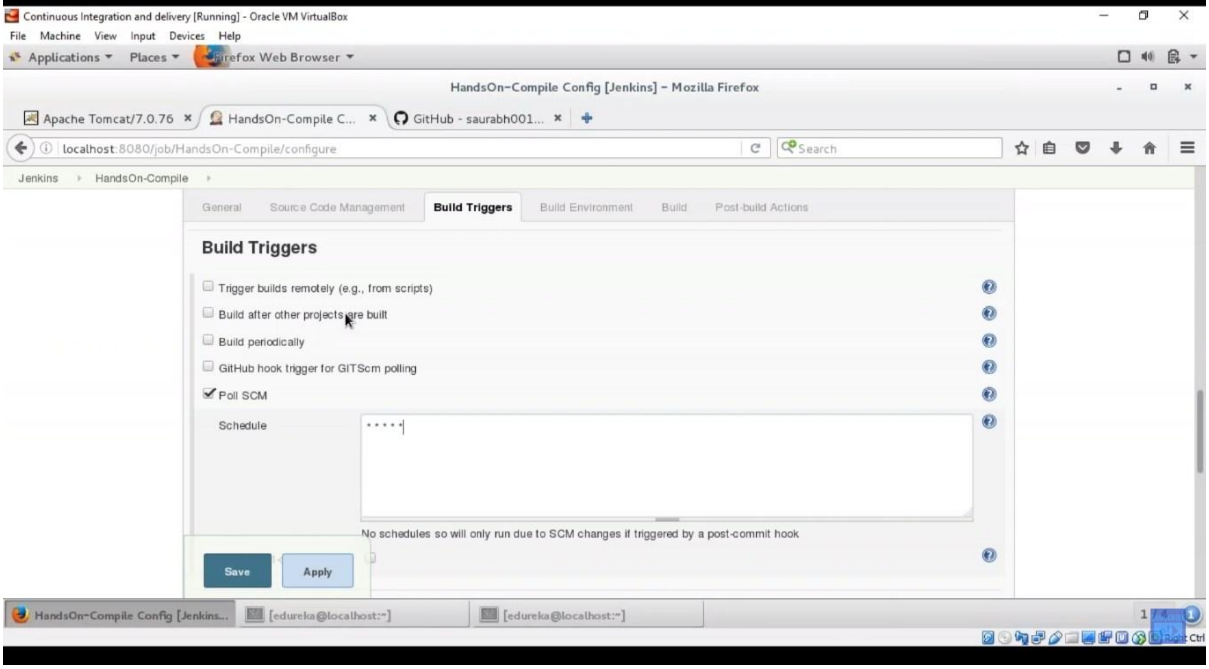

Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
HandsOn-Compile Config [Jenkins] - Mozilla Firefox
Apache Tomcat/7.0.76
HandsOn-Compile C...
GitHub - saurabh001...
localhost:8080/job/HandsOn-Compile/configure
Search
Jenkins
HandsOn-Compile
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
Please enter Git repository.
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/master
Add Branch
Save
Apply
HandsOn-Compile Config [Jenkins...
[edureka@localhost:~]
[edureka@localhost:~]

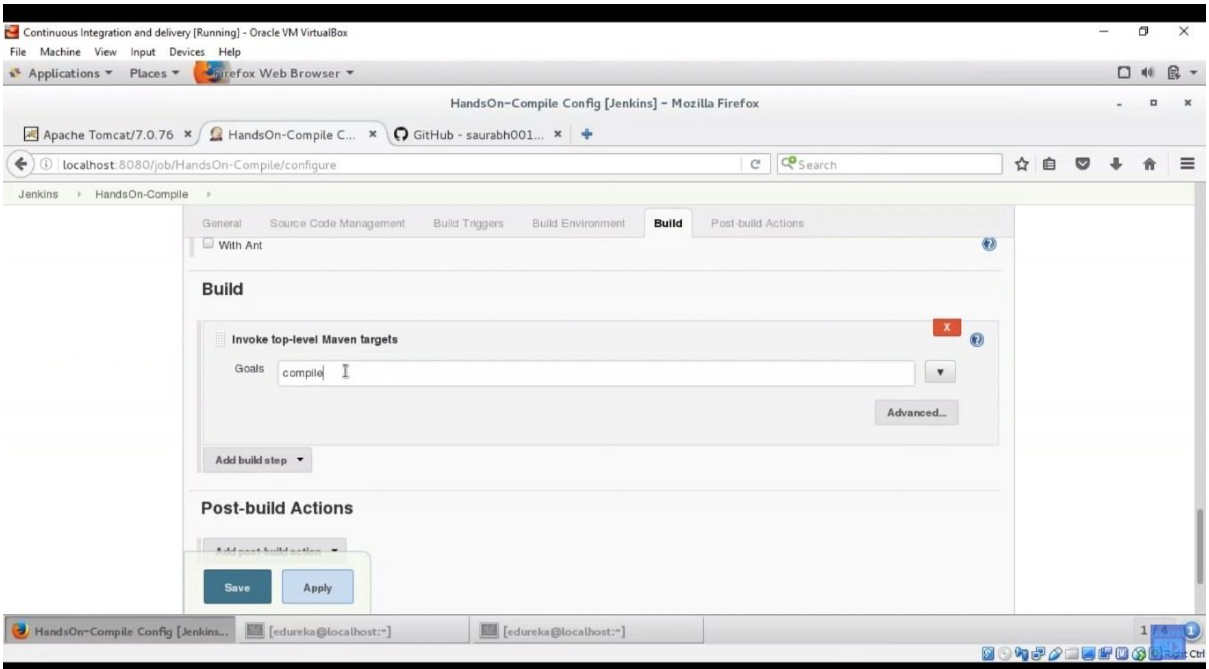

Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
GitHub - saurabh0010/game-of-life: Demo application for the 'Jenkins: The Definitive Guide' book - Mozilla Firefox
Apache Tomcat/7.0.76
HandsOn-Compile C...
GitHub - saurabh001...
GitHub, Inc. (US) https://github.com/saurabh0010/game-of-life
Search
Branch: master
New pull request
Find file
Clone or download
This branch is even with wakaleo:master.
Clone with HTTPS
Use Git or checkout with SVN using the web URL.
https://github.com/saurabh0010/game-of
Download ZIP
wakaleo Merge pull request wakaleo#198 from GeraldScott/master
gameoflife-acceptance-tests move Selenium Dockerfile
gameoflife-build Updated versions
gameoflife-core Revert live cell and dead cell 11 months ago
gameoflife-deploy Tidied up code 7 years ago
gameoflife-web Updated dependencies a year ago
.gitignore Moved web tests into a separate module. 7 years ago
README.markdown Edit for clarity 11 months ago
infinitest.filters First commit 8 years ago
pom.xml Serenity version 1.8.3 missing from Maven repo so changed it to 1.8.4 11 months ago
README.markdown
GitHub - saurabh0010/game-of-l...
[edureka@localhost:~]
[edureka@localhost:~]

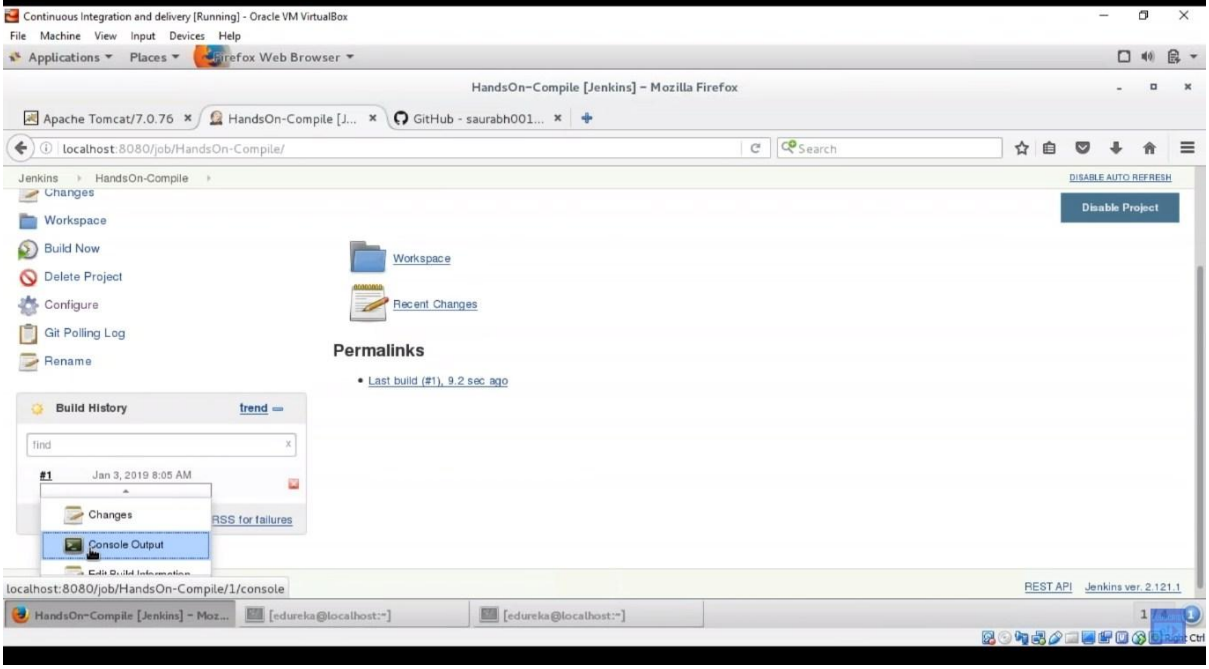

Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
HandsOn-Compile Config [Jenkins] - Mozilla Firefox
Apache Tomcat/7.0.76
HandsOn-Compile C...
GitHub - saurabh001...
localhost:8080/job/HandsOn-Compile/configure
Search
Jenkins HandsOn-Compile
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/saurabh0010/game-of-life.git
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/master
Add Branch
Save
Apply
HandsOn-Compile Config [Jenkins...
[edureka@localhost:~]
[edureka@localhost:~]

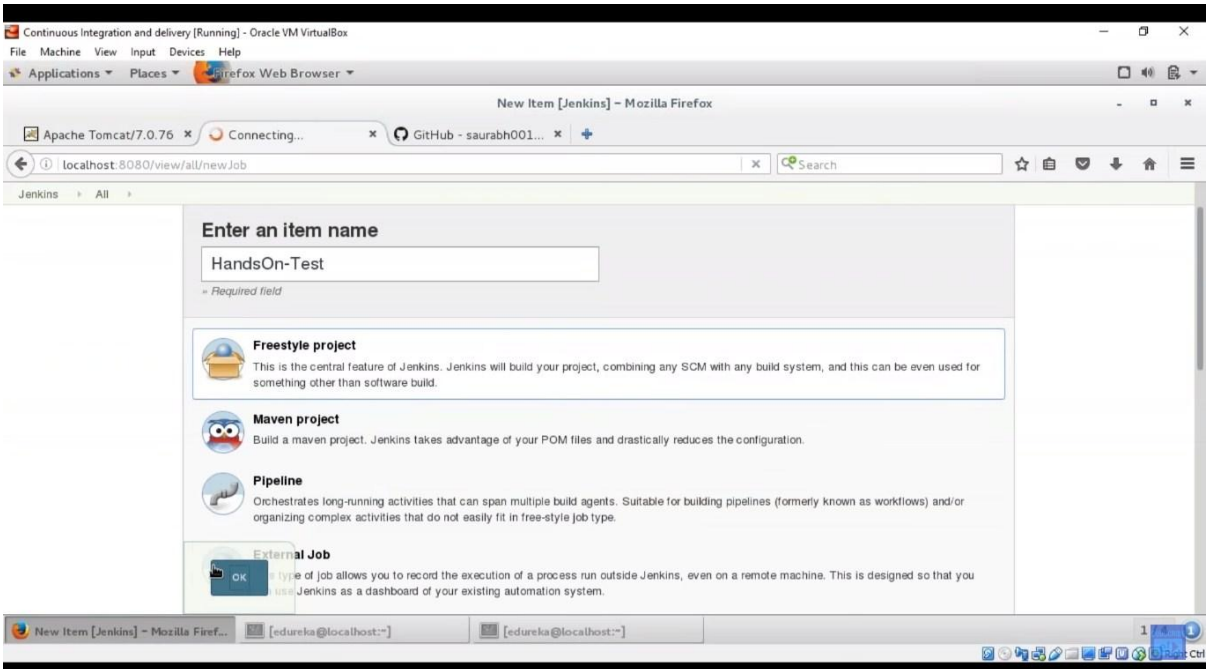

Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
HandsOn-Compile Config [Jenkins] - Mozilla Firefox
Apache Tomcat/7.0.76
HandsOn-Compile C...
GitHub - saurabh001...
localhost:8080/job/HandsOn-Compile/configure
Search
Jenkins HandsOn-Compile
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build Triggers
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Save
Apply
HandsOn-Compile Config [Jenkins...
[edureka@localhost:~]
[edureka@localhost:~]

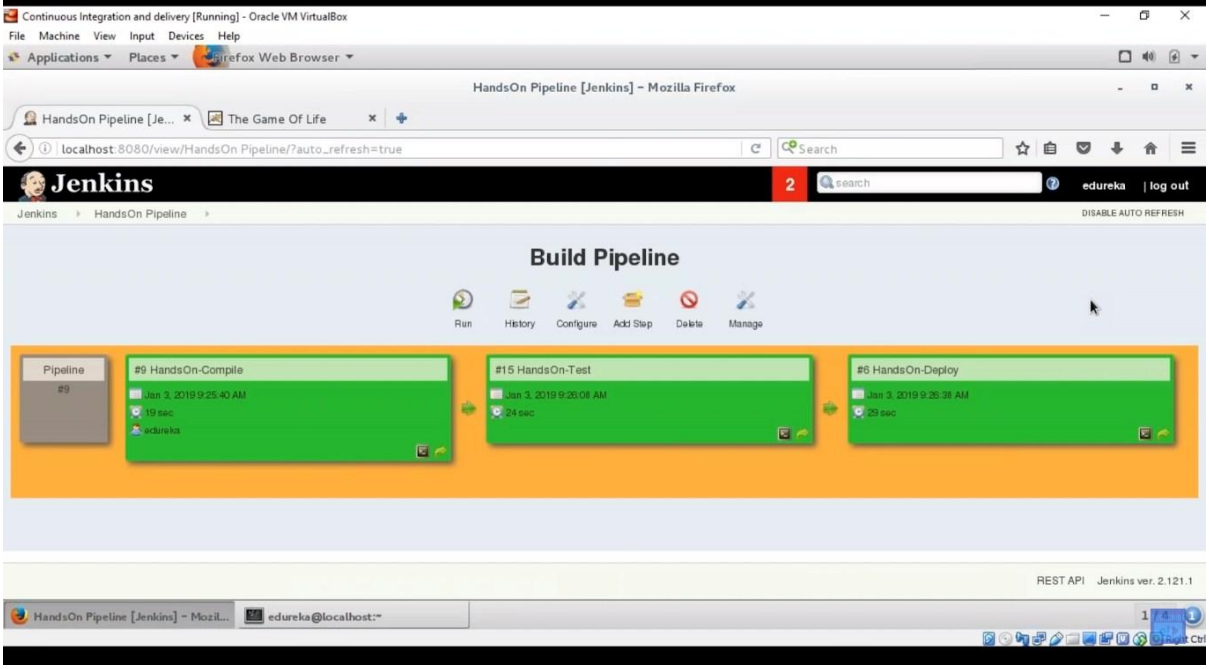

Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
HandsOn-Compile Config [Jenkins] - Mozilla Firefox
Apache Tomcat/7.0.76
HandsOn-Compile C...
GitHub - saurabh001...
localhost:8080/job/HandsOn-Compile/configure
Search
Jenkins HandsOn-Compile
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Schedule
No schedules so will only run due to SCM changes if triggered by a post-commit hook
Save
Apply
HandsOn-Compile Config [Jenkins...
[edureka@localhost:~]
[edureka@localhost:~]


Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
HandsOn-Compile Config [Jenkins] - Mozilla Firefox
Apache Tomcat/7.0.76
HandsOn-Compile C...
GitHub - saurabh001...
localhost:8080/job/HandsOn-Compile/configure
Search
Jenkins HandsOn-Compile
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Schedule
* * * * *
No schedules so will only run due to SCM changes if triggered by a post-commit hook
Save
Apply
HandsOn-Compile Config [Jenkins...
[edureka@localhost:~]
[edureka@localhost:~]

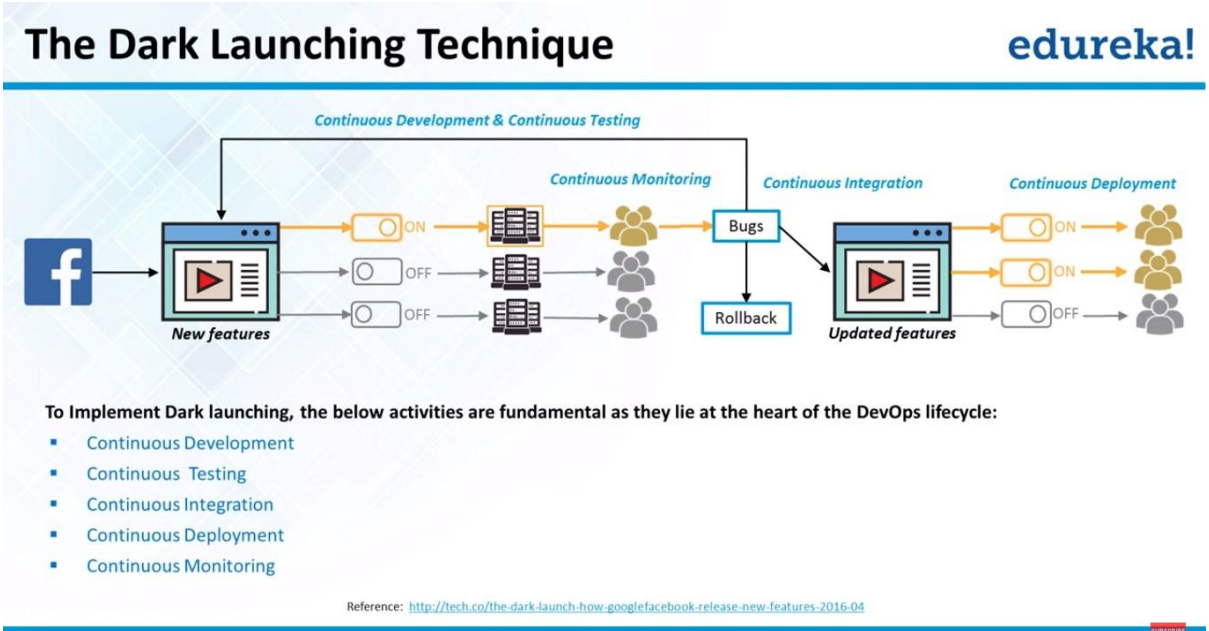

Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
HandsOn-Compile Config [Jenkins] - Mozilla Firefox
Apache Tomcat/7.0.76
HandsOn-Compile C...
GitHub - saurabh001...
localhost:8080/job/HandsOn-Compile/configure
Search
Jenkins HandsOn-Compile
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
With Ant
Build
Invoke top-level Maven targets
Goals
compile
Advanced...
Add build step
Post-build Actions
Save
Apply
HandsOn-Compile Config [Jenkins...
[edureka@localhost:~]
[edureka@localhost:~]

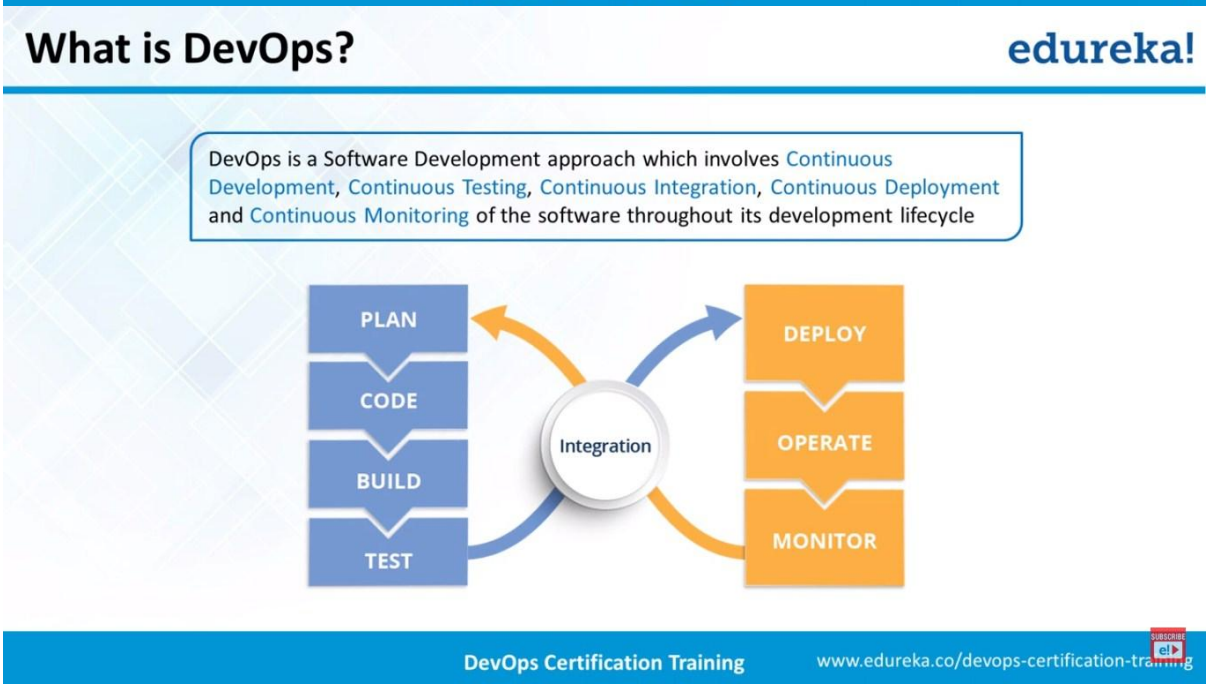

Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
HandsOn-Compile [Jenkins] - Mozilla Firefox
Apache Tomcat/7.0.76
HandsOn-Compile [J...
GitHub - saurabh001...
localhost:8080/job/HandsOn-Compile/
Search
Jenkins HandsOn-Compile
DISABLE AUTO REFRESH
Changes
Workspace
Build Now
Delete Project
Configure
Git Polling Log
Rename
Disable Project
Workspace
Recent Changes
Permalinks
Last build (#1), 9.2 sec ago
Build History
trend
find
#1
Jan 3, 2019 8:05 AM
Changes
Console Output
RSS for failures
localhost:8080/job/HandsOn-Compile/1/console
REST API
Jenkins ver. 2.121.1
HandsOn-Compile [Jenkins] - Moz...
[edureka@localhost:~]
[edureka@localhost:~]


Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
New Item [Jenkins] - Mozilla Firefox
Apache Tomcat/7.0.76
Connecting...
GitHub - saurabh001...
localhost:8080/view/all/newJob
Search
Jenkins All
Enter an item name
HandsOn-Test
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
External Job
OK
Jenkins as a dashboard of your existing automation system.
New Item [Jenkins] - Mozilla Firef...
[edureka@localhost:~]
[edureka@localhost:~]


Continuous Integration and delivery [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
HandsOn Pipeline [Jenkins] - Mozilla Firefox
HandsOn Pipeline [Je...
The Game Of Life
localhost:8080/view/HandsOn Pipeline/?auto_refresh=true
Search
Jenkins
2
search
edureka | log out
Jenkins HandsOn Pipeline
DISABLE AUTO REFRESH
Build Pipeline
Run
History
Configure
Add Step
Delete
Manage
Pipeline
#9
#9 HandsOn-Compile
Jan 3, 2019 9:25:40 AM
19 sec
edureka
#15 HandsOn-Test
Jan 3, 2019 9:26:08 AM
24 sec
#6 HandsOn-Deploy
Jan 3, 2019 9:26:38 AM
29 sec
REST API
Jenkins ver. 2.121.1
HandsOn Pipeline [Jenkins] - Mozil...
edureka@localhost:~

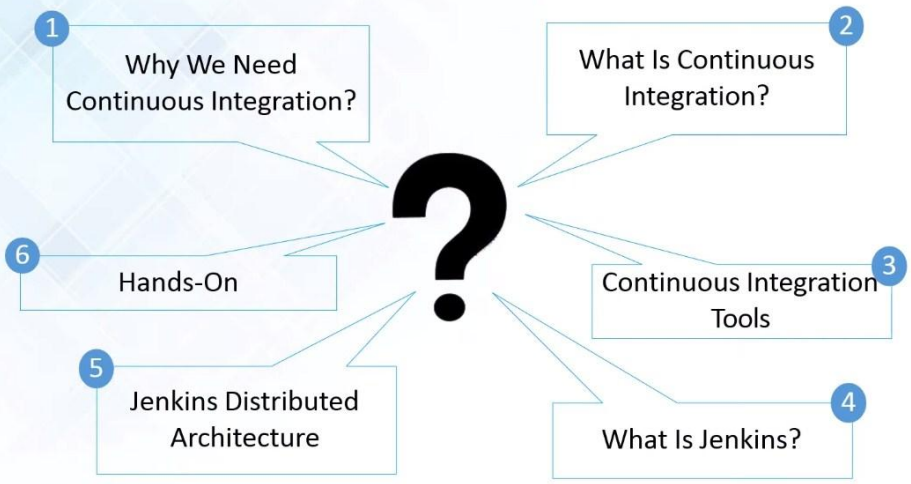
New tutorial





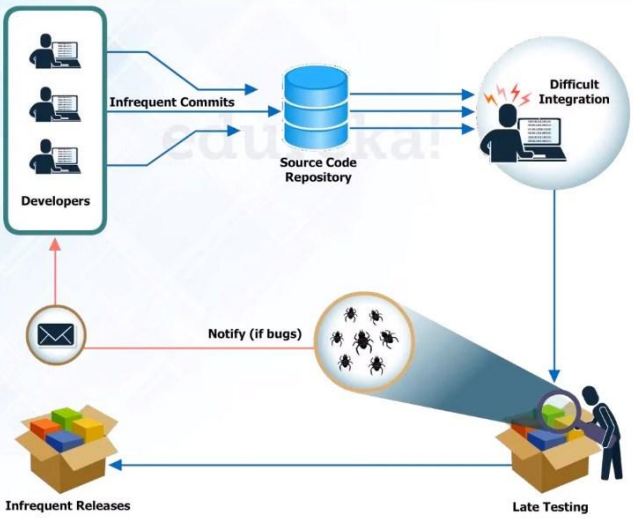
Do you know the following?

edureka!



Process Before Continuous Integration

edureka!

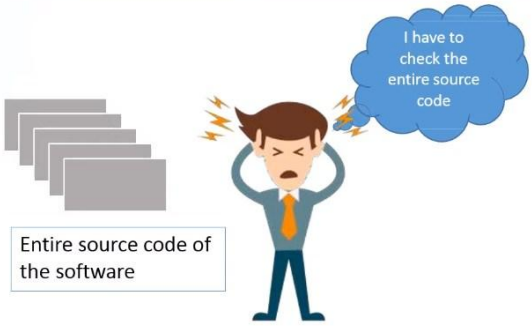


Problems Before Continuous Integration

edureka!

Developers have to wait till the complete software is developed for the test results.

If the test fails then locating and fixing bugs is very difficult. Developers have to check the entire source code of the software.



Problems Before Continuous Integration

edureka!

Software delivery process was slow

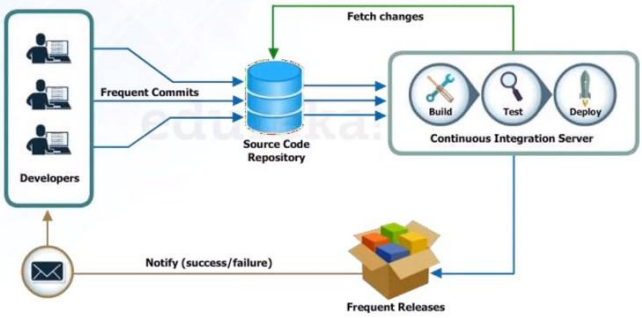
Continuous feedback pertaining to things like coding or architectural issues, build failures, test status etc. was not present

Looking for DevOps Online Training? Call us at **IN: 9606058406 / US: 18338555775** or visit **www.edureka.co**

Continuous Integration To The Rescue

edureka!

- 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

edureka!

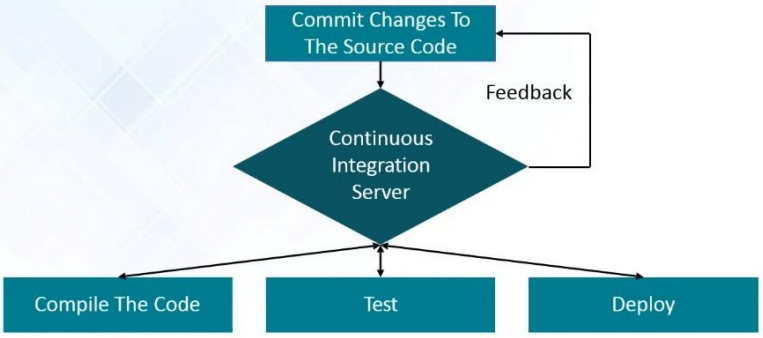
Before Continuous Integration	After Continuous Integration
The entire source code was built and then tested.	Every commit made in the source code is built and tested.
Developers have to wait for test results	Developers know the test result of every commit made in the source code on the run
No Feedback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

Looking for DevOps Online Training? Call us at **IN: 9606058406 / US: 18338555775** or visit **www.edureka.co**

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.

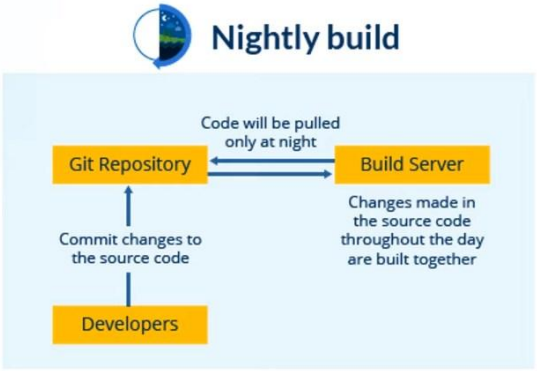


Continuous Integration Case-Study: Nokia

edureka!

Problem Statement:

- In Nokia a process called **Nighly build** was used.
- In this process every night an automated system pulls the code added to the shared repository throughout the day and builds that code.
- Since the code that was built at night was quite large, locating and fixing of bugs was a real pain.

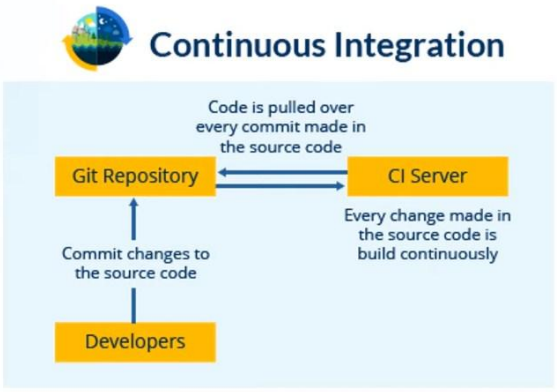


Continuous Integration Case-Study: Nokia

edureka!

Solution:

- Nokia adopted Continuous Integration (CI).
- As a result, every commit made to the source code in the repository was built.
- If the build result shows that there is a bug in the code, then the developers only need to check that particular commit.



Continuous Integration Tools

edureka!

Jenkins

Buildbot

Travis CI

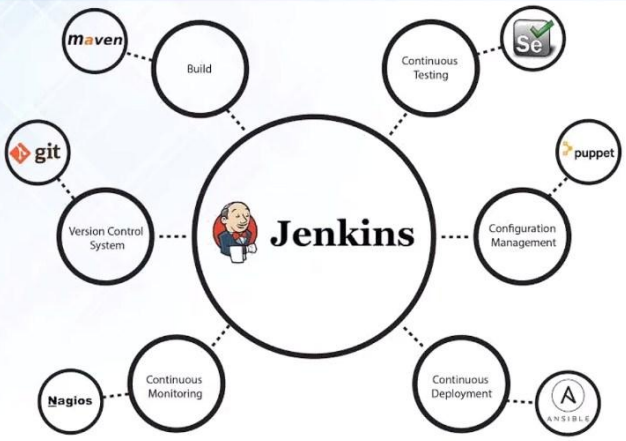
Atlassian Bamboo

Looking for DevOps Online Training? Call us at **IN: 9606058406 / US: 18338555775** or visit **www.edureka.co**

What Is Jenkins?

edureka!

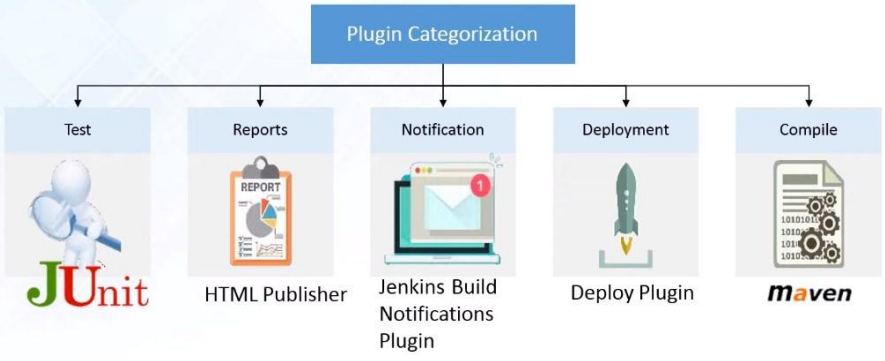
Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Plugins allows integration of various DevOps stages.



Jenkins Plugins

edureka!

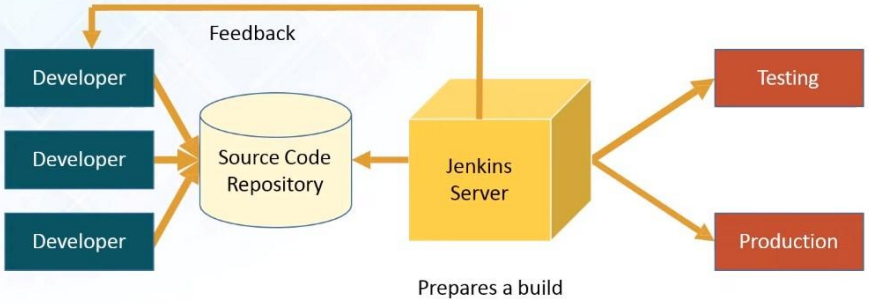
Jenkins supports plugins, which allow Jenkins to be extended to meet specific needs of individual projects



Jenkins Example

edureka!

- Developers commit changes to the source code
- Continuous Integration server pulls that code and triggers a build
- The build application is then deployed on the testing server for testing
- After testing the application, it is then deployed on the production server
- The concerned teams are constantly notified about the build and test results



Shortcomings Of Single Jenkins Server

edureka!

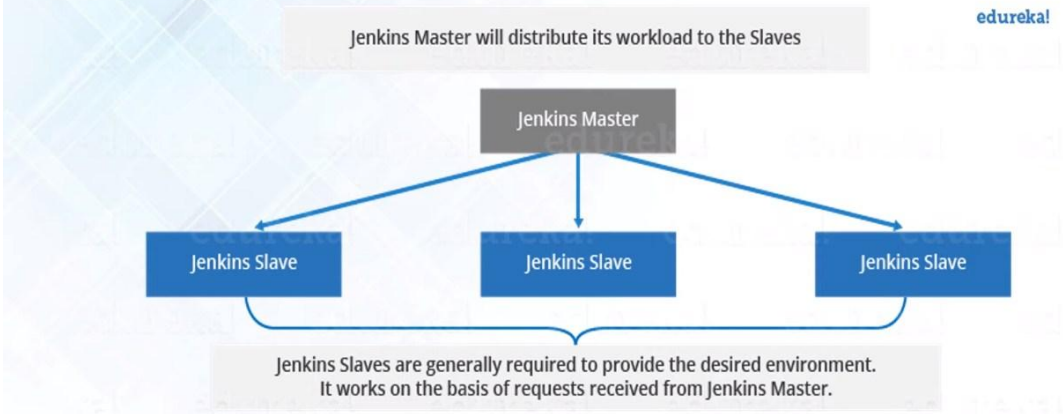
If you need to run web tests using Internet Explorer, you need to use a Windows machine.

Another build job may require Linux box.

How to manage spikes in build activity

Jenkins Distributed Architecture

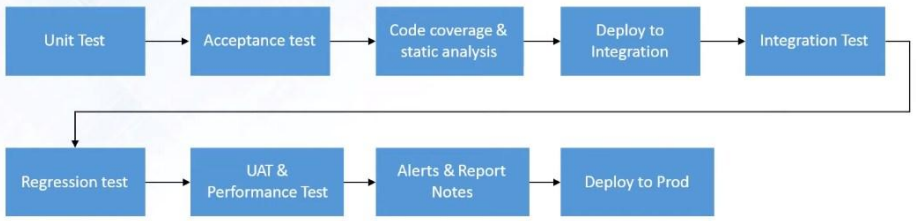
edureka!



Jenkins Delivery Pipeline

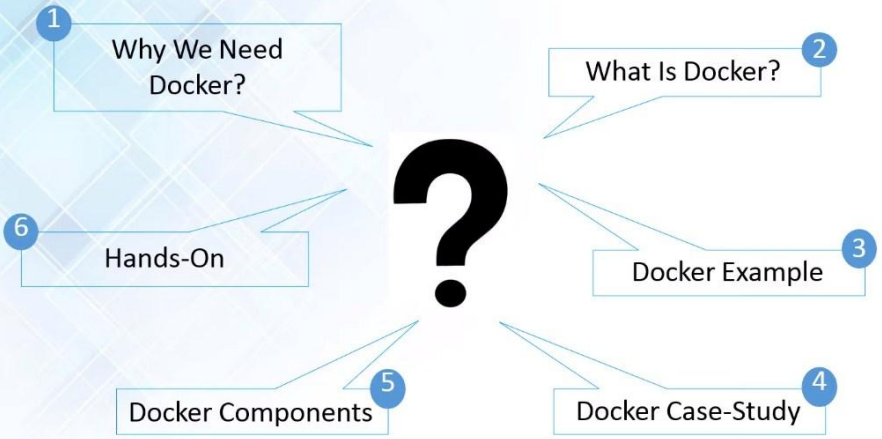
edureka!

- The product is developed and tested. A small change is made in the code and is checked to SCR that triggers UT
- On success, automatically trigger acceptance tests & produce code-coverage and static analysis report
- If the acceptance tests succeeds, system will trigger the deployment of project to an integration environment
- Now, invoke integration test suite & regression test suite
- If these pass, the system triggers UAT and performance test
- On success, automatically promote the project to release repository and notify the developer about the result.



Do you know the following?

edureka!



Problems Before Docker

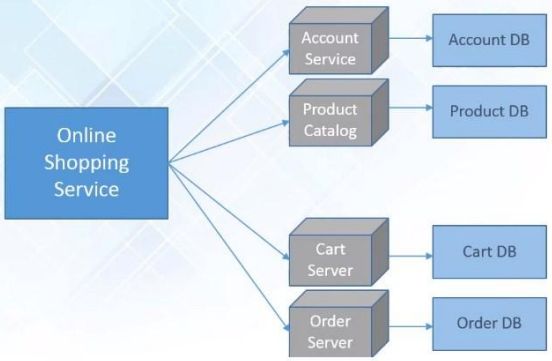
edureka!

An application works in developer's laptop but not in testing or production. This is due to difference in computing environment between Dev, Test and Prod.

Problems Before Docker

edureka!

The idea behind microservices is that some types of applications become easier to build and maintain when they are broken down into smaller, composable pieces which work together. Each component is developed separately, and the application is then simply the sum of its constituent components.

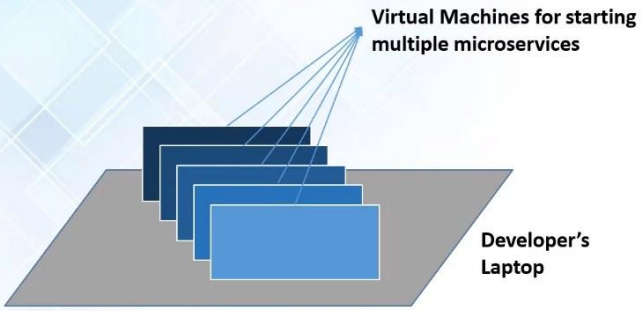


For example imagine an online shop with separate microservices for user-accounts, product-catalog order-processing and shopping carts

Problems Before Docker

edureka!

Developing an application requires starting several of microservices in one machine. So if you are starting five of those services you require five VMs on that machine.

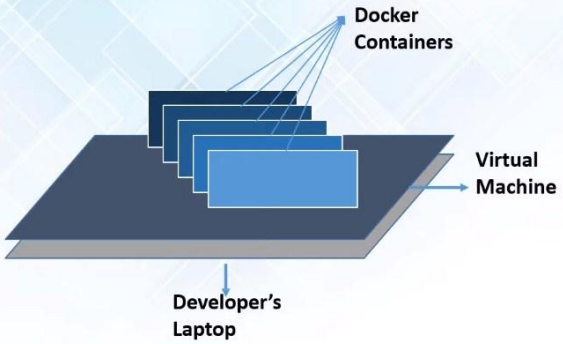


Looking for DevOps Online Training? Call us at **IN: 9606058406 / US: 18338555775** or visit **www.edureka.co**

How Docker Solves These Problems

edureka!

You can run several microservices in the same VM by running various Docker containers for each microservice.

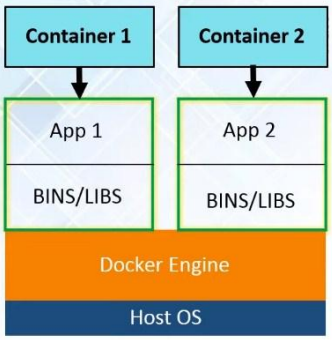


Provides a consistent computing environment throughout the whole SDLC.

What Is Docker?

edureka!

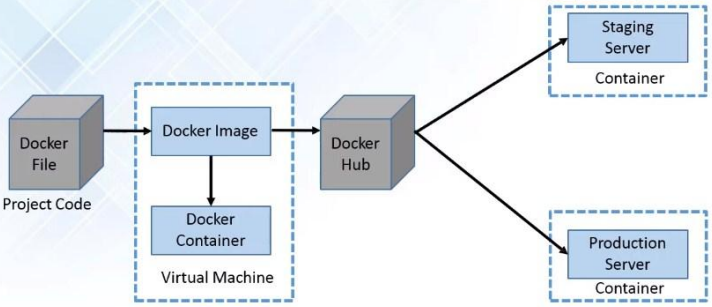
- Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.
- Docker containers are lightweight alternatives to Virtual Machines and it uses the host OS.
- You don't have to pre-allocate any RAM in containers.



Docker In A Nutshell

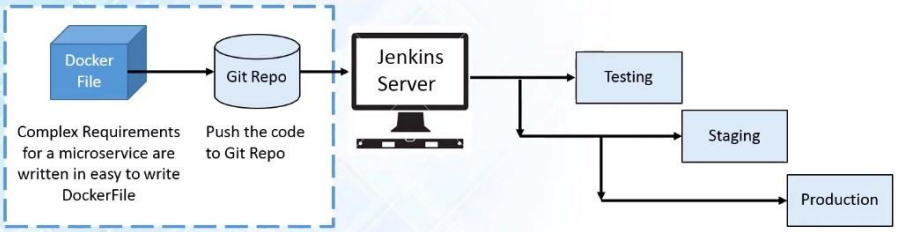
edureka!

- Docker file builds a Docker image and that image contains all the project's code
- You can run that image to create as many docker containers as you want
- Then this Image can be uploaded on Docker hub, from Docker hub any one can pull the image and build a container



Docker Example

edureka!



- Create complex requirements for a microservice within an easy-to-write Dockerfile.
- Push the code up to the Git Repo.

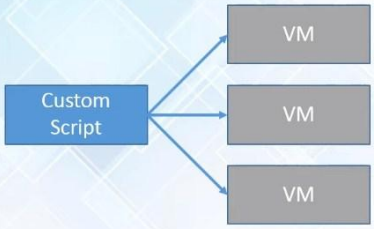
- CI server pull it down and build the exact environment that will be used in production to run the test suite without needing to configure the CI server at all.
- Deploy it out to a staging environment for testers.
- Roll exactly what you had in development, testing, and staging into production

Docker Case-Study Indiana University

edureka!

Problem Statement:

1



Applications are deployed in the VMs using custom scripts

2

- Their environment was optimized for their legacy Java-based applications.

Sorce: https://www.docker.com/customers/indiana-university-delivers-state-art-it-115000-students-docker-datacenter

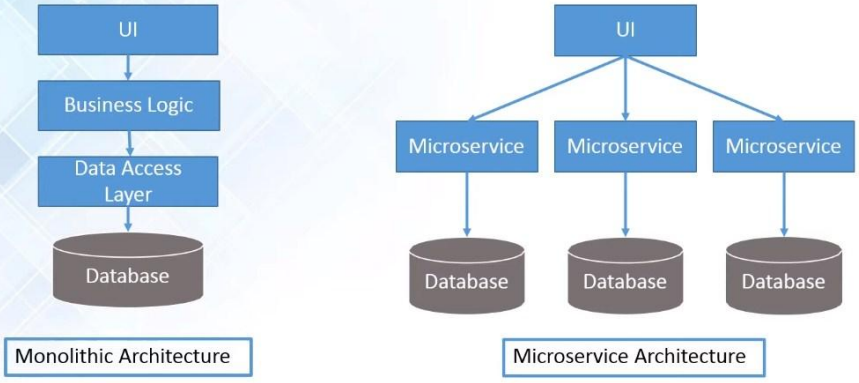
Docker Case-Study Indiana University

edureka!

Problem Statement:

The University wanted to improve the way they architect applications, by moving to a microservices based architecture for their applications

3



Monolithic Architecture

Microservice Architecture

Sorce: https://www.docker.com/customers/indiana-university-delivers-state-art-it-115000-students-docker-datacenter

Looking for DevOps Online Training? Call us at **IN: 9606058406 / US: 18338555775** or visit **www.edureka.co**

Docker Case-Study Indiana University

edureka!

Problem Statement:

Security was needed for student's data such as SSNs and student health data.

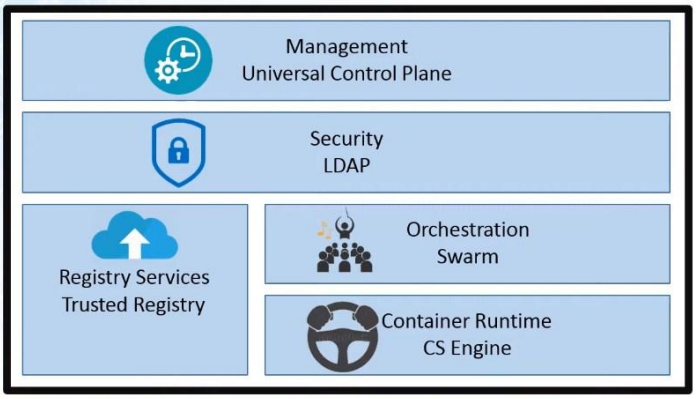
4

Sorce: https://www.docker.com/customers/indiana-university-delivers-state-art-it-115000-students-docker-datacenter

Docker Case-Study Indiana University

edureka!

Solution: Docker Data Center (DDC)



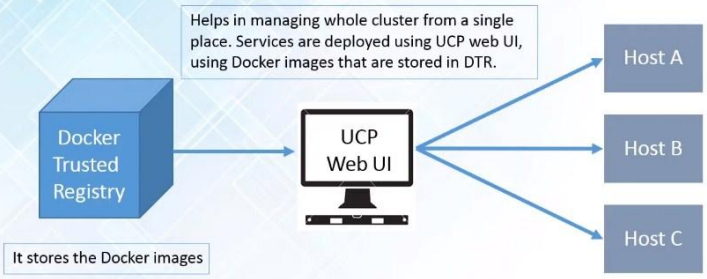
Sorce: https://www.docker.com/customers/indiana-university-delivers-state-art-it-115000-students-docker-datacenter

Looking for DevOps Online Training? Call us at **IN: 9606058406 / US: 18338555775** or visit **www.edureka.co**

Docker Case-Study Indiana University

edureka!

Solution: Docker Data Center (DDC)



IT ops teams leverages Universal Control Plane to provision Docker installed software on hosts, and then deploy their applications without having to do a bunch of manual steps to set up all their infrastructure.

Sorce: https://www.docker.com/customers/indiana-university-delivers-state-art-it-115000-students-docker-datacenter

Docker Registry

edureka!

- Docker Registry is a storage component for Docker Images
- We can store the Images in either Public / Private repositories
- Docker Hub is Docker's very own cloud repository

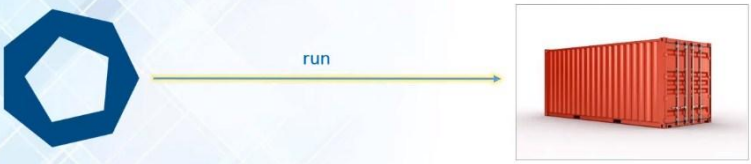
Why Use Docker Registries?

- Control where your images are being stored
- Integrate image storage with your in-house development workflow

Looking for DevOps Online Training? Call us at **IN: 9606058406 / US: 18338555775** or visit **www.edureka.co**

Docker Images & Containers

edureka!



Docker Images

Docker Containers

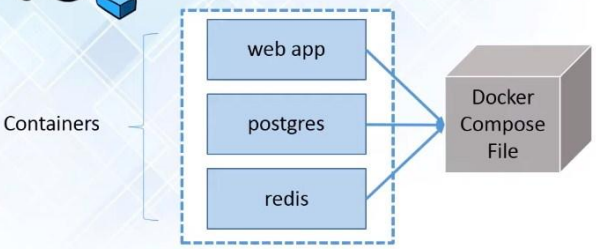
- Read Only Template Used To Create Containers
- Built By Docker Users
- Stored In Docker Hub Or Your Local Registry

- Isolated Application Platform
- Contains Everything Needed To Run The Application
- Built From One Or More Images

Docker Compose

edureka!

Docker Compose makes it easier to configure and run applications made up of multiple containers. For the example: imagine being able to define three containers—one running a web app, another running postgres, and a third running redis—all in one YAML file and then running those three connected containers with a single command.



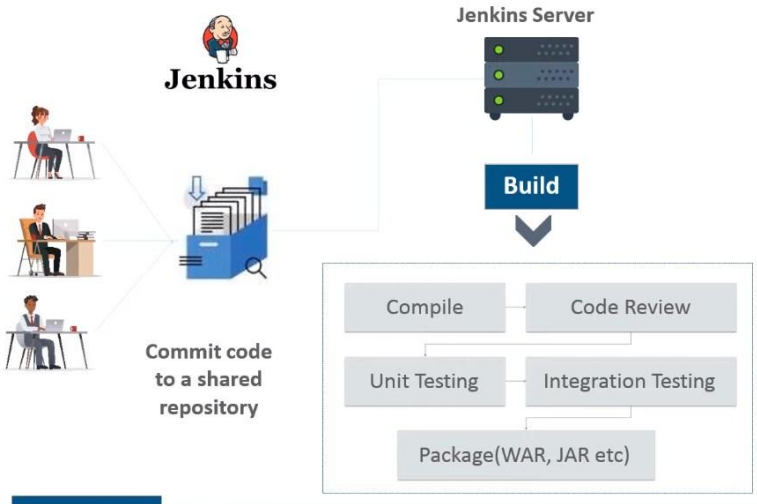
You can run these three containers with a single command

Agenda

What is DevOps?

DevOps is a software development approach which involves Continuous Development, Continuous testing, Continuous Integration, Continuous Deployment and Continuous Monitoring throughout its development lifecycle.

Continuous Integration



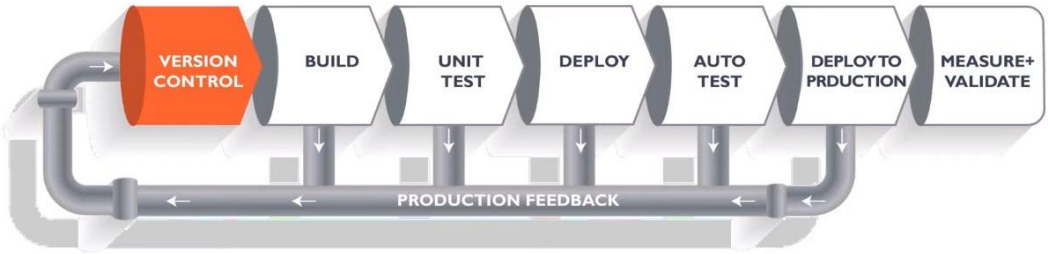
CI CD Pipeline

DevOps Stages

CI CD Pipeline

edureka! Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co

What is CI and CD?



Continuous Integration Example

Have to create image on this

what is then the CI.
Since there are various developers say just take an example of a team which has five developers and they are building a website which you can say is a shopping website.
so, the shopping website can have various avenues like one of them is the landing page where you show all the products. there can be another page which shows the user details and the past orders and things like that. Another page can be the shopping cart which so whatever products the person has added to the cart.
Another page which is the "checkout page" where the person will pay and then make the purchase.
so, there are various avenues in a software. These can be called as features of the software.
So, when you have multiple features, then you have to combine those features and merge them into a single version control system where you merge the various feature branches of the code and make a single pool of code which can be now comprising all the features from various branches committed by the developers and then you can build it as a

product and move on to testing.Now when you merge the code and you build the code to get a single unit saw of the software.That phase is false integration.

Have to create image on this

Integration means putting all the code together, so we are putting together code from of the contributing developers and then we are merging them and building it as a whole.
So that phase is called the continuous integration phase.
Why do we call it continuous because as I said in DevOps we use various tools to automate the pipeline.
Okay in contrast to the traditional practices what a traditional software company does manually people write the code and manually put it somewhere then the manually merge it and the manual abilities in build commands and then a tester manually tests it and then it manually gets deployed in a server by a system admin and then again there is a series of sanity test manually done again by a QA and then it manually is deployed to the production what do you see here in every phase there is a manually done thing right so that is how things are done in traditional way.
In DevOps we transform this to do everything in an automatic manner so as soon as a person commits the code from the repository, it automatically moves and merges itself with the other codes which are present which are committed at the same time and then it triggers off a build reaction which will build or compile the code and then it triggers the test.
Then it triggers the deploy auto test and so on.
so the entire thing is triggered step by step automatically so that is what we are going to achieve.
you're going to create an automated pipeline so why is it known as continuous integration as I said this version control and build phase are called the integration phase now since this is completely automated it happens continuously like as soon as the person commits code in a version control system it continually starts the merge process and it

CI CD Pipeline

edureka! Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co

Jenkins – The Ultimate CI Tool

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Plugins allows integration of various DevOps stages.

Jenkins

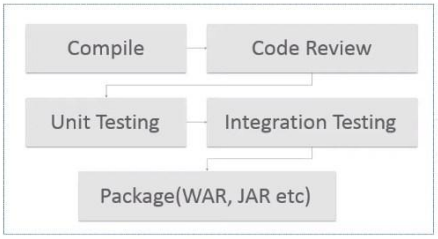
CI CD Pipeline

Continuous Delivery

Commit code to a shared repository

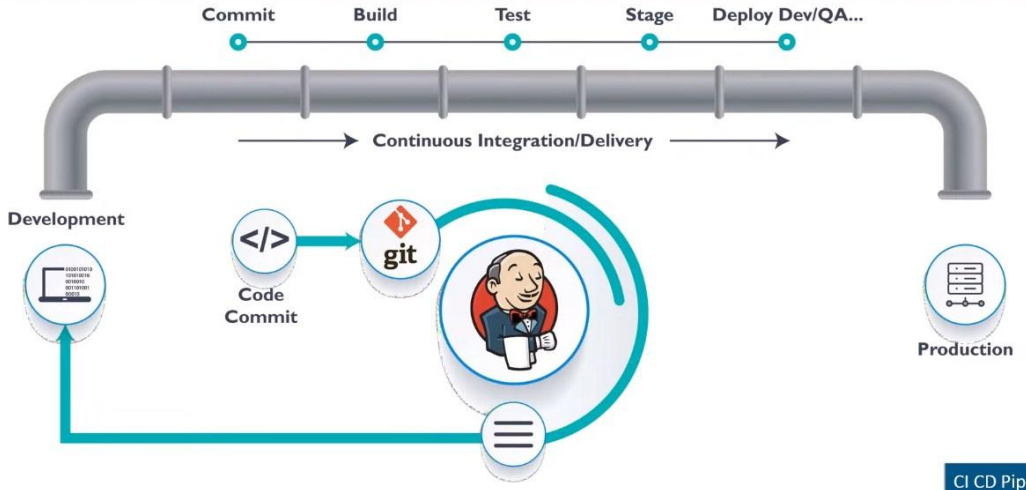
Jenkins Server

Build



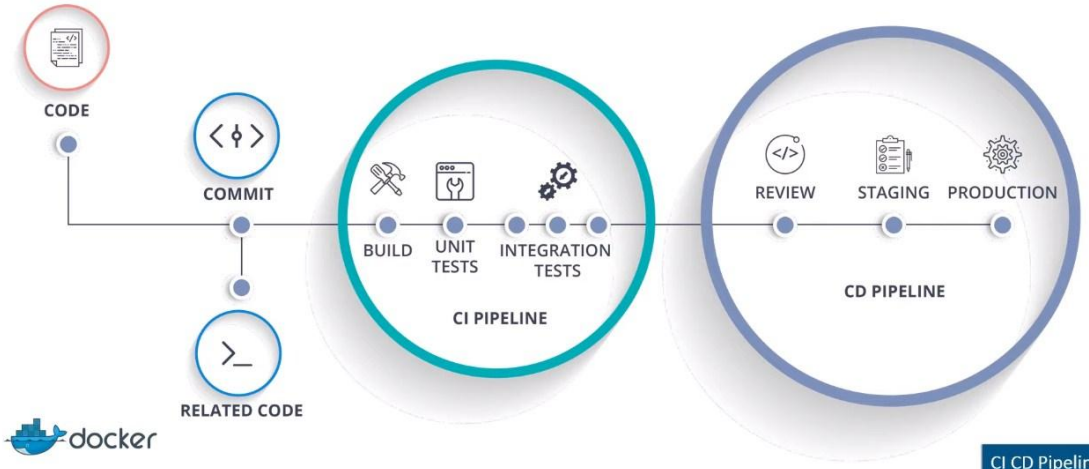
Deploy the built application on test server for UAT (User Acceptance Test)

Jenkins - Its Importance

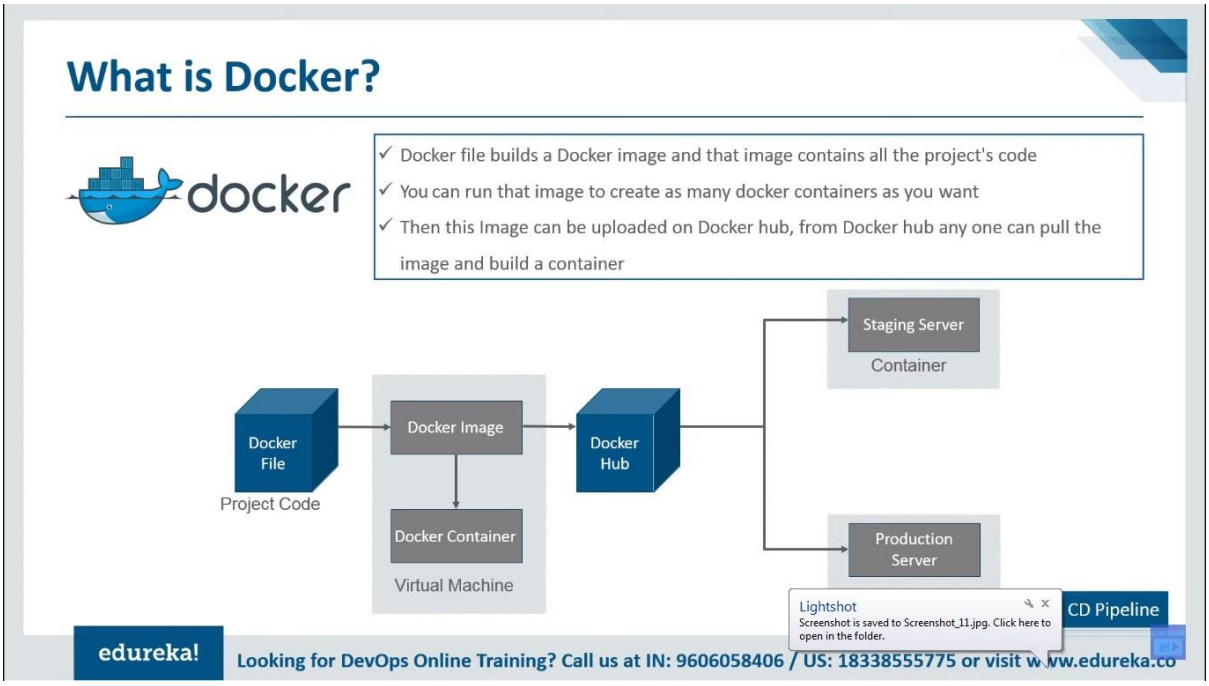


CI CD Pipeline

Docker – Its Importance



CI CD Pipeline



Dev OPS Another

Agenda

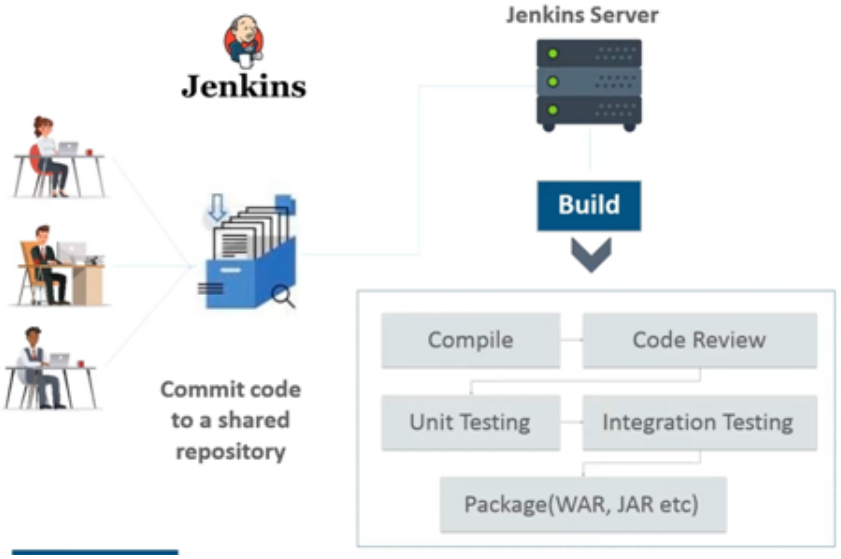
What is DevOps?

DevOps is a software development approach which involves Continuous Development, Continuous testing, Continuous Integration, Continuous Deployment and Continuous Monitoring throughout its development lifecycle.

DevOps Stages

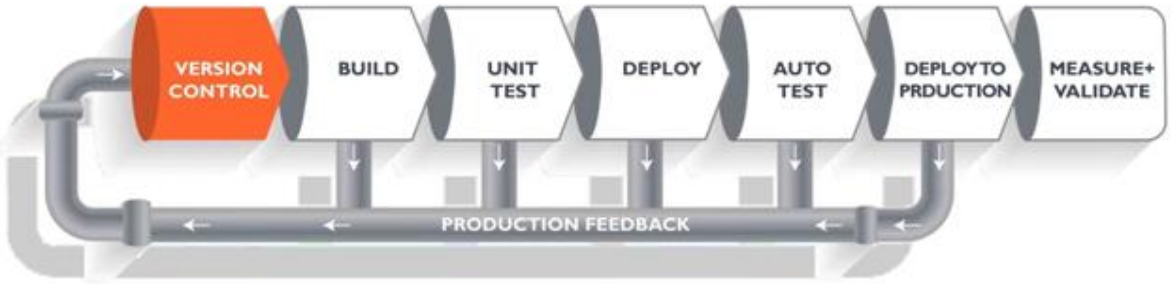
CI CD Pipeline

Continuous Integration

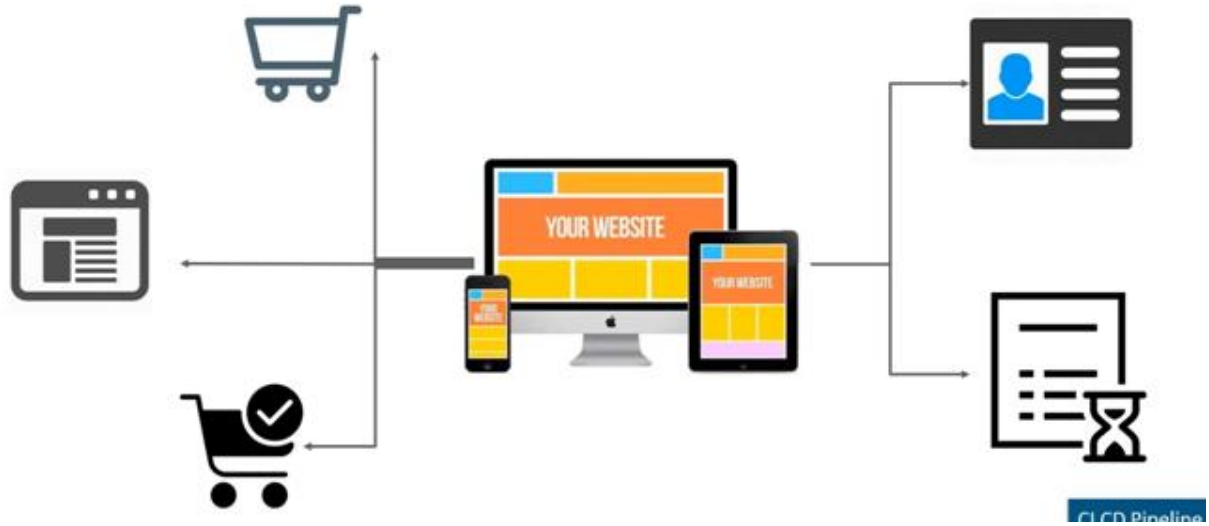


CI CD Pipeline

What is CI and CD?



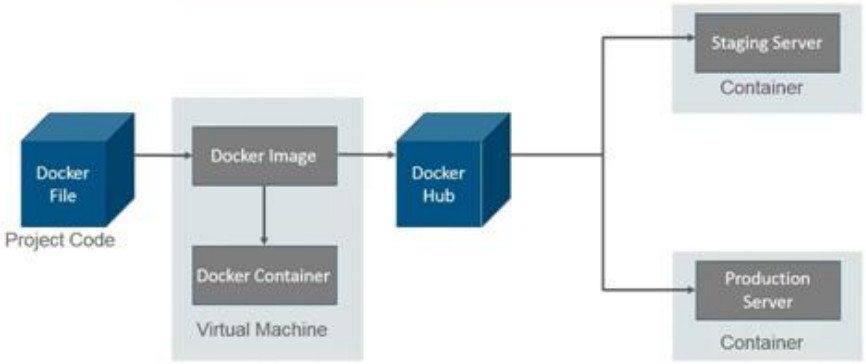
Continuous Integration Example



CI CD Pipeline

What is Docker?

- ✓ Docker file builds a Docker image and that image contains all the project's code
- ✓ You can run that image to create as many docker containers as you want
- ✓ Then this Image can be uploaded on Docker hub, from Docker hub any one can pull the image and build a container



Amazon Elastic Compute Cloud (Amazon EC2)

- EC2 stands for **Elastic Compute Cloud**
- Used to create **virtual machines** in the AWS cloud
- Can be used as a server for **hosting websites** and other server management features such as **storage**, **ports**, and **security**.

Amazon Relational Database Service (Amazon RDS)

- Used to create **dedicated database instances**
- Makes **designing infrastructure** more user friendly
- Can **support multiple database engines** like MySQL, SQL server, PostgreSQL and many more

Amazon Simple Storage Service (Amazon S3)

- It offers a **highly secure** and **redundant** file storage service.
- Amazon S3 also offers **integrations** to help prevent data breaches (PCI-DSS, HIPAA)
- You get **data flexibility** without almost **zero latency**.