(FAQ)What is DevOps?

DevOps is a combination of cultural philosophies, practices, and tools which enables organizations to perform fast and frequent releases so that customers can innovate faster and compete for better in the market.

How did we use build and release activities before DevOps?

1. Build and release activities are done manually
2. Software Development Life Cycles(SDLC) used to be longer.
3. Earlier SDLC used to be 6 months or even more. We used to follow the waterfall model
4. There was no collaboration between teams(dev, qa, ops)
5. We do it all manually, it hinders productivity.

What do we do in DevOps?

1. Follow agile development model
    a. We keep SDLC short, generally, 2 to 3 weeks, which means every ⅔ week we do a release.
    b. We follow an incremental development model, break bigger tasks into much smaller tasks, and do one task at a time.
2. Collaboration between teams (development, testing, and DevOps), even though they are different teams they should logically work as a team.
3. Automate repetitive tasks
4. DevOps practices like
    a. Continuous Integration (CI) - FAQ
        i. CI is a development practice where developers commit and merge their changes several times a day, this is automatically picked up by an automation tool and does following
            1. Build
            2. Test
            3. Sonar Analysis
            4. Vulnerability Scanning
            5. And create a deployable package
            6. If there are errors in this process we send immediate feedback to the development team through email or communicator.
    b. Continuous Deployment (CD)
        i. It expands upon CI, and it automatically deploys the latest artifact from development to production.
    c. Continuous Delivery(CD)
        i. It expands upon CI, and it automatically deploys the latest artifact from development to production.

    ii. Here the only difference is after clicking approve button code gets deployed to production.
d. Configuration Management
    i. It is all about installing and configuring certain software on hundreds of thousands of servers.
    ii. The most popular tools are Ansible, Chef, Puppet
e. Microservices(FAQ)
    i.
f. Monitoring and Logging
    i. Monitoring helps us to keep our application healthy and improve performance etc...
    ii. We monitor CPU, memory, disk, network performance, etc…
    iii. Popular tools used for monitoring is Zabbix, nagios, datadog, prometheus and grafana, etc...
    iv. Log management, we have to accumulate logs from different servers and put them on a central logging system.

(FAQ)What is the difference between Continuous Deployment and Continuous Delivery?

Source Code Management Tool

1. This is more of a development tool.
2. Now, this has become a DevOps tool too. Because we do automation and we might need to use it for Jenkins files, Ansible playbooks, kubernetes YAML documents.

Why source code management tools?

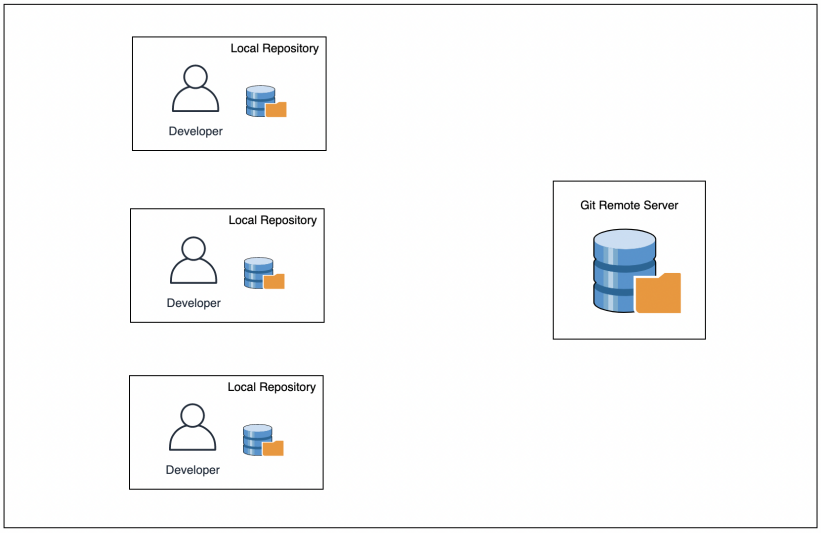
1. Managing source code without SCM tool is a mess
2. SCM tools enable multiple developers to collaborate and ease the merging process.
3. SCM maintains multiple versions of the file, so that if we want to check the history for troubleshooting a defect etc...
4. If the latest release has defects, we can quickly fall back to the previous version.
5. SCM records all changes along with details like how made changes, why those changes are made, and timestamp.
6. SCM tools enable security on the code, only authorized people will have access to the code.
7. And many more…

Which source code management tool?

1. There are various options for the SCM tool, but the most popular tool as of today is git.
2. Git is distributed

3. Git is fast
4. Git is lightweight
5. Git has lots of features
6. Using git you can work locally.
7. Git is open-source

Git Distributed Architecture



- In distributed architecture same copy of the remote repository is distributed on all developer's machines.
- Because of distributed architecture, developers can work locally and all git operations will be fast.

How to get git and use it?

1. We can setup git on our own servers
2. We can use GitHub, Bitbucket, Gitlab, Codecommit, etc...

(FAQ) Is your git hosted, git, or SaaS git?

Ans) We are using hosted git
In my previous project, we installed GitLab software on our servers.

Create an account in https://github.com

Create Git Repository

1. From UI create a repository, repository represents a new project
2. To work with Git we need to have a git client
3. There are different types of clients
    a. GUI (Graphical User Interface)
    b. CLI (Command Line Interface)
4. For our training we wanna use CLI, in the real world you can use a client of your choice.
5. Install git bash on your computer.

Clone Git Repository

Clone gets the remote copy to the local
git clone https://github.com/javahometech/shopping-app.git
cd shopping-app

Configure Git client with the user and email

The very first time we have to configure user and email.
Open git bash and run the following commands

```
git config --global user.name "Hari Kammana"
git config --global user.email "hari.kammana@gmail.com"
```

Global configurations are stored in the following file
~/.gitconfig

Adding Files to Git repository

1. Create a new file and put your code
2. Let's choose an editor for working with our project files
3. Download and install visual studio code
    a. https://code.visualstudio.com/download
4. Open visual studio code
    a. File → open and open your local repository folder
5. Create a new file info.txt and add some text. And save the file.

Git youtube Videos

https://youtube.com/playlist?list=PLH1ul2iNXl7vkfIFF2BxLA5xpkbvWtFWf

Git Working Area

This is a virtual area, the changes we make to the local repository automatically keeps them in the working area.

Git Staging Area or Index Area

Use this area to stage the files from working so that you can commit them to the local repository.

Git Pull

- Git pull internally performs two operations i.e. fetch plus merge

Git Fetch

- It fetches remote changes to local without merging.

(FAQ)What is the difference between pull and fetch?

Fetch only fetches remote changes to local without merging, and pull fetches remote changes to local, and also it merges.

(FAQ)Git Conflicts

Conflicts occur when multiple developers are working on the same file and line.
We can resolve conflicts using “Visual Studio Code”

(FAQ)What is git blame?

This command shows line-by-line information of a file with author, timestamp, and message.

Branches in Git

The branch is a virtual folder that isolates tasks, in the git branch is lightweight(it’s just a pointer). Branches enable us to isolate our work, as a good practice, every task should have a separate branch.
Every repository by default contains a “master” branch.

Merging Branches

1. We can merge locally, but it does not give away to review the code
2. Push it to remote and merge using pull request (PR)

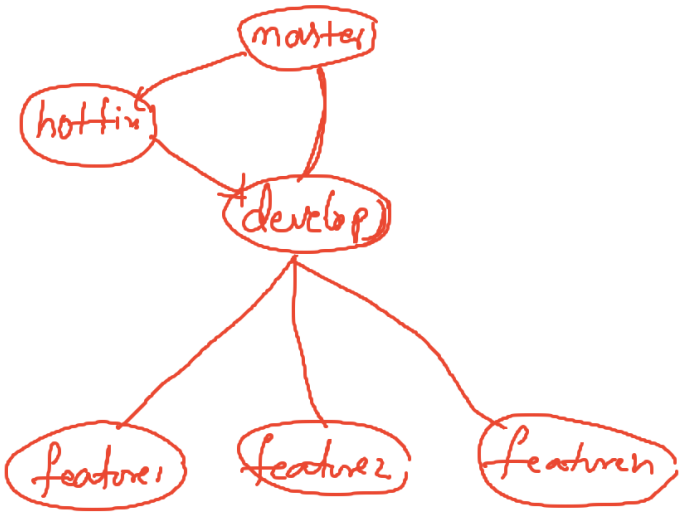
Git Branch Commands

git branch (list all branches in local)
git branch -r (list all remote branches)
git branch -a (list local and remote branches)
git branch task1 (create new branch)
git checkout task2 (Switch the branch)
git merge task1 (merge the branch locally)
git branch -d task1(delete branch)
git branch -D task1(force delete a branch)
git checkout -b task3(create and checkout the branch)

Git branching strategies (FAQ)

Choosing branching strategies is essential while working on real-world projects, there are different strategies available, let's discuss one strategy.

- "master" is the main branch, we use this branch for production releases, and we don't let developers directly push to master.
- "feature" branch, this is the short-lived branch that contains a specific new feature
- "develop" branch, this is a long-lived branch, we use this branch to merge new features.
- "hotfix" branch, this branch is used to fix production defects, this branch is created from master.



-

Git merging strategies

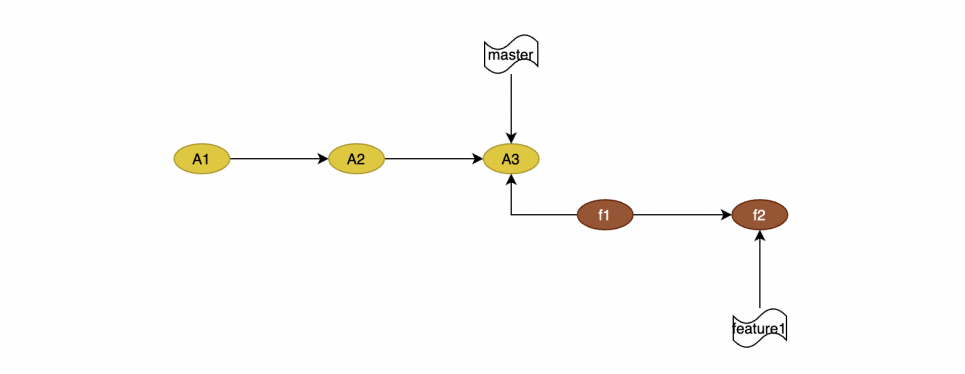
Git follows one of the following merging strategies based on the context

1. Fast-forward merge

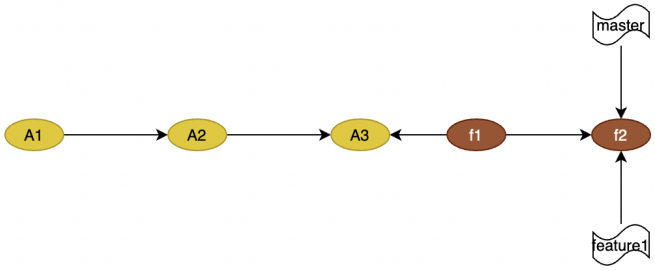
2. Recursive or Three-Way merge
3. Rebase (FAQ)

Fast-forward Merge

Before Fast-forward merge

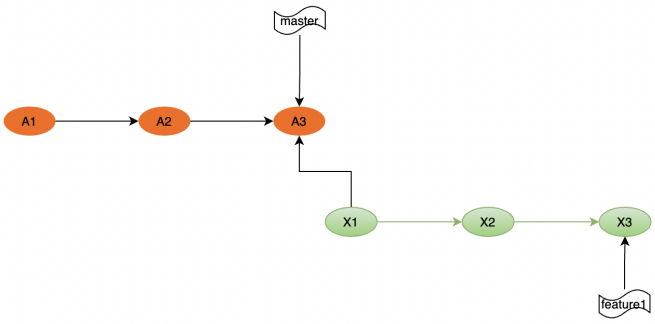


After Fast-forward merge

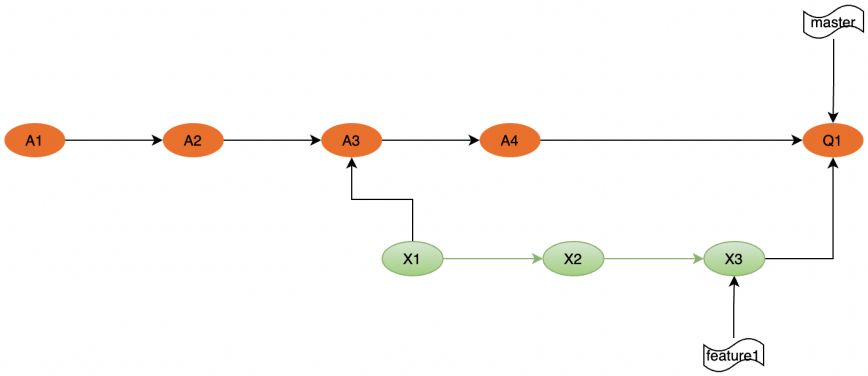


Recursive Merge

Before recursive merge

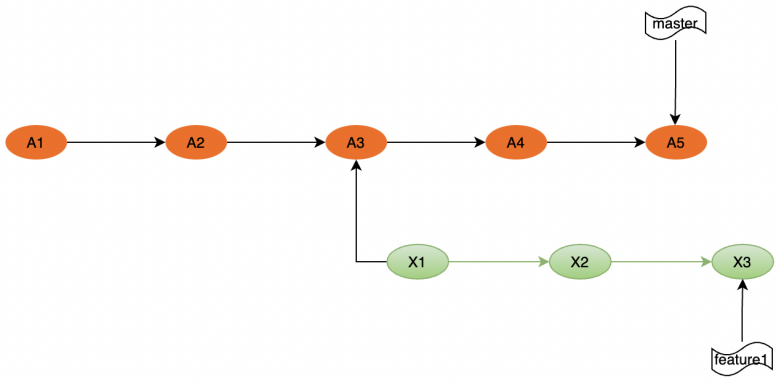


After Recursive Merge

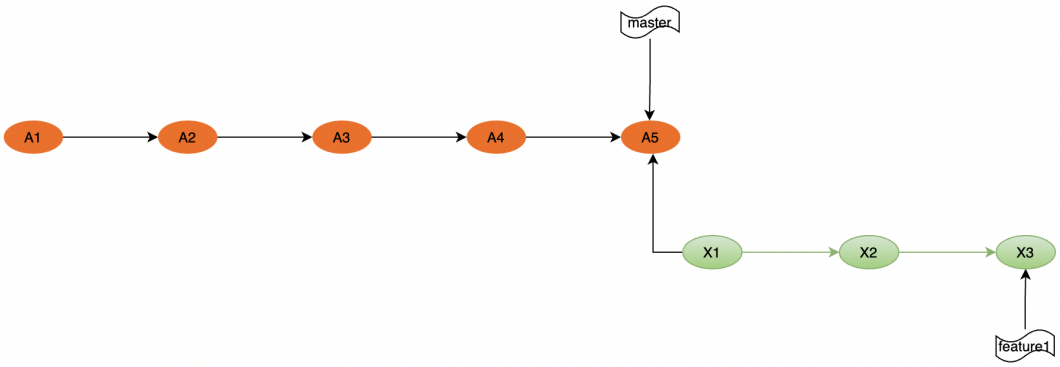


Rebase (FAQ)

Rebase changes the base commit by merging new commits in its main branch. The only purpose of rebasing is to maintain liner history. But in the real world, it not used much.



If we rebase feature1 with the master, it looks as below



(FAQ)What is HEAD in git?

- HEAD is a lightweight pointer
- HEAD always points to the latest commit in the current branch

Git Tags

Tag is a lightweight pointer, its purpose is to mark important commits in history, It mainly used for releases.

Git Branch naming conventions

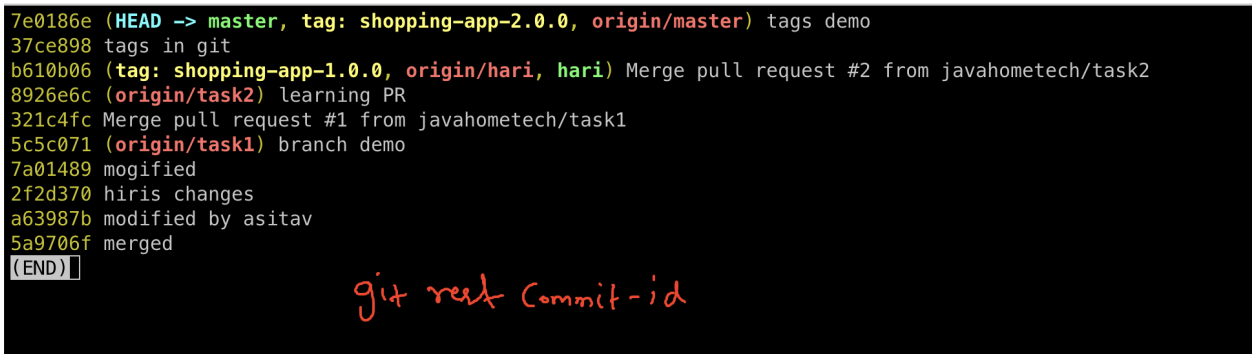
1. Naming Feature branches
    a. feature/ramya/login
    b. feature/sitha/signout
    c. feature/hari/upload
2. Naming conventions for the development branch

    a. develop
    b. develop/teama
    c. develop/teamb
3. Release branches
    a. release/1.0.0
    b. release/2.0.0
    c. release/3.0.0
4. Hotfix branches
    a. hotfix/abc
    b. hotfix/hari/xyz
    c. hotfix/ramya/mno

Undoing Changes

1. Undo changes in the working area
    a. git restore *
    b. git restore details.txt
2. Un staging a file
    a. git restore --staged info.txt
3. Removing local commit
    a. We can remove or undo a commit using two different commands (reset, revert)

Git Reset



git reset b610b06 (this removes the commits above this commit)
Use reset for only local commits
Don’t use reset for public commits

(FAQ)What is soft reset?

- Remove local commits and keep changes in the staging area

(FAQ)What is a hard reset?

- Remove local commits and permanently discard changes

Reset with --mixed option

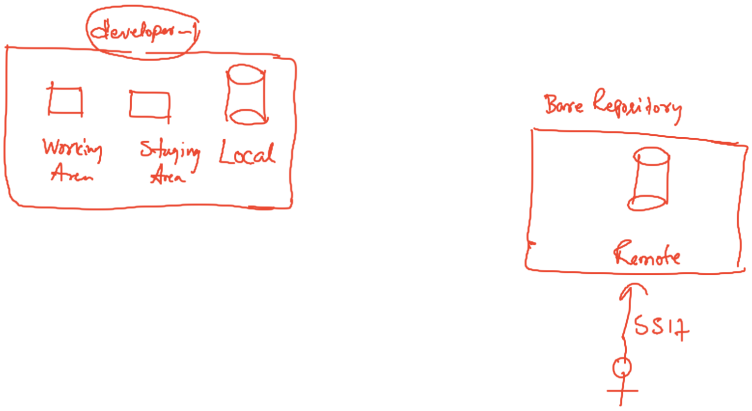
--mixed, means remove the commit and keep changes in the working area.

(FAQ) What is cherry-pick

Cherry command picks specific commits and integrates.

(FAQ) What is a bare repository?

A git repository that does not have a working area and staging area is called a bare repository.



(FAQ)What is git init?

Converting local folder into a git repository.

Git revert

- This command undoes changes made in the specific commit.
- Revert does not remove the commit instead it undoes changes in the commit and makes a new commit.
- This is the perfect command to undo public commits.

Git commit hooks

Git diff command

Git stash

Build Tools

The build is essential for all sort of programming languages, usually, the build does following

1. Download third party dependencies and add them to the classpath
2. Create a deployable folder structure
3. Compile the code
4. Execute Junit test cases
5. Create a package(we also call this an artifact)
    a. In java, the formats are jar, war, ear, etc...

1. clone/pull changes from the repository
2. mvn package

Setting Up servers for DevOps Implementations

1. We wanna use AWS cloud for setting up servers
2. Create an AWS account, it's 12 months free, you need a debit or credit card for creating this account.
3. https://aws.amazon.com/free

Launch Linux Server in AWS and install Maven Build tool

- AWS offers EC2 (Virtual Server) for launching Linux, Windows, Ubuntu, etc…
- Connect to EC2 instance

Install Maven on Linux Server

We can do this in different ways

YUM package manager

On Linux servers, yum is a package manager
Yum is used for install/uninstall/upgrade software packages

Example Install git on Linux

sudo yum install git -y

Install maven

sudo yum install maven -y

Build Java Project using maven

- git clone https://github.com/javahometech/my-app
- cd my-app
- mvn package

Maven Dependencies

Dependency is a jar file used by our code. Developers use different dependencies in the project, those dependencies are defined in pom.xml.
When we build projects using maven, it downloads dependencies from a repository.

Maven Repositories(faq)

There are three types of maven repositories

1. Central Repository
    a. This server is over the internet
    b. Maven team maintains this repository
    c. This server has hundreds of thousands of dependencies
    d. Maven by default downloads dependencies from central
2. Local Repository
    a. Every time going to the central repository slows down the build process
    b. So maven maintains a local repository in the following location
    c. ~/.m2/repository(faq)
3. Remote Repository
    a. This is like central but managed by our company.
    b. Some customers do not like to use “Central Repository” for security reasons
    c. We also remote repository to stage our artifacts for deployments
    d. We see remote repository is every project
    e. The following are used to set up a remote repository
        i. Sonatype Nexus (I will cover this)
        ii. Jfrog Artifactory

Maven Build Lifecycle(faq)

When we run Maven commands it does the following

- Validate
    - Validated pom.xml file
- Compile
    - It downloads dependencies and compiles the code
- Test
    - Execute JUnit test cases
- Package
    - Create an artifact (war/jar/ear)
- Verify
    - Verify integration test results(not used)
- Install
    - Copy our project artifact to a local repository
- Deploy
    - Upload the artifact to the remote repository(for example nexus)

What is pom.xml?

- ☐ pom.xml is the configuration file used by maven
- ☐ pom.xml is created when the maven project is created by developers
- ☐ Maven projects can be created using IDEs or the maven command line
- ☐ pom.xml has the following, details
    - ☐ groupId → we put reverse domain name of the client
    - ☐ artifactId → project name
    - ☐ packaging → jar, war, etc...
    - ☐ dependencies → external jars used by the project
    - ☐ etc...

Create Maven project using command line

mvn archetype:generate -DgroupId=in.javahome -DartifactId=shopping-app
-DarchetypeArtifactId=maven-archetype-webapp -DarchetypeVersion=1.4
-DinteractiveMode=false.

Snapshot and Release types of artifacts

1. Snapshot means this version is currently in development
2. Release means this version is released and no more changes to the same version.

How to skip test cases in maven?

mvn package -DskipTests=true

Maven Target Folder

The target folder is the output folder of maven, that where maven packages the application.

Sonatype Nexus3

https://help.sonatype.com/repomanager3

Nexus is a repository manager and we can organize artifacts and dependencies.
Nexus supports different formats

- Maven
- Yum
- Apt-get
- Nuget
- Python
- Docker
- Helm (Kubernetes)
- Etc…

Install and configure Nexus3 on Linux

1. ssh into Linux machine
2. cd /opt
3. sudo wget https://download.sonatype.com/nexus/3/latest-unix.tar.gz
4. sudo tar xf latest-unix.tar.gz
5. sudo chown -R ec2-user:ec2-user nexus-3.31.0-01 sonatype-work
6. sudo yum install java-1.8.0-openjdk.x86_64 -y

Demo Upload Maven Artifacts to Nexus

1. Crete Maven Repository
    a. Create two maven hosted repositories (Snapshot and Release)
2. Make changes to pom.xml
3. Make changes to maven settings.xml, and configure nexus username and password.

Web Server

1. Deploying web applications on web servers.
2. Webservers are mandatory for deploying web applications
3. These are some popular web servers in the market
    a. For Java
        i. Tomcat
        ii. Weblogic
        iii. Websphere

iv. Jboss
v. Etc…

b. For .Net
   i. IIS
c. For PHP
   i. Apache Webserver
   ii. Nginx
   iii. Etc...

Apache Tomcat

- The majority of java web applications run on tomcat.
- Tomcat is open source, we can use the source code and customize the product.
- Tomcat is lightweight and easy to use.
- Tomcat is written in java

Install Tomcat on Linux Server

➔ Install java
➔ sudo yum install java-1.8.0-openjdk.x86_64 -y
➔ Install tomcat
➔ cd /opt
➔ sudo wget https://downloads.apache.org/tomcat/tomcat-8/v8.5.68/bin/apache-tomcat-8.5.68.tar.gz
➔ sudo tar xf apache-tomcat-8.5.68.tar.gz
➔ sudo mv apache-tomcat-8.5.68 tomcat8
➔ sudo chown -R ec2-user:ec2-user tomcat8/

Start & Stop Tomcat

- /opt/tomcat8/bin/startup.sh
- /opt/tomcat8/bin/shutdown.sh

Deploying Applications on Tomcat

☐ Tomcat has a deployment directory
☐ $TOMCAT_HOME/webapps is the deployment directory of tomcat
☐ Copy war file under deployment folder of tomcat and restart tomcat.

Tomcat Logs

- Tomcat logs help you to troubleshoot deployment issues.
- $TOMCAT_HOME/logs/

Application Logs

This is used for troubleshooting applications specific bugs and exceptions.
The location is different for different applications(we should get this info from the development team)
Java guys use the log4j framework for application logging

Run Tomcat as a service

A task for you guys

Jenkins

1. The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying, and automating any project.
2. Jenkins is written in java
3. Jenkins is an essential tool for DevOps engineers.
4. Alternative tools for Jenkins
    a. Bamboo
    b. Circle CI
    c. Solano CI
    d. Etc.
5. Jenkins was popularly known for Continuous Integration

Install Jenkins on Linux

https://pkg.jenkins.io/redhat-stable/
sudo chkconfig jenkins on
sudo service jenkins start

Task, Run nexus as a service and enable nexus on system boot

https://help.sonatype.com/repomanager3/installation/run-as-a-service

Creating First Jenkins Job

Install Publish Over SSH plugin on Jenkins

Configure “Publish Over SSH Plugin”

Discarding unwanted builds

1. Every build occupies a space on the disk
2. It is good to delete old builds and free the space on the disk.

3. At the job level, we have options to discard old builds

Build Triggers

1. Build Periodically
    a. This feature takes a schedule and triggers the job.
    b. The drawback of this feature is, it triggers even if there are no new commits.
2. Poll SCM(faq)
    a. This runs jobs on a schedule
    b. This runs jobs if and only if there are new commits
3. Github Hook trigger
    a. There is no overhead on Jenkins
    b. We configure a webhook on Github, that triggers the Jenkins job.

Email Notifications

To send and receive emails we need SMTP servers

The folder structure of Jenkins at Linux Box

- Default Jenkins home location (/var/lib/jenkins)
- users → this folder contains jenkins users configuration
- nodes → using slaves/nodes we can scale Jenkins, nodes configuration goes under nodes folder.
- plugins → In Jenkins every feature is constructed through a plugin, this folder contains the plugins we installed.
- workspace → our jobs usually contain SCM configuration for example git, the files from SCM are kept under the workspace.
- jobs → build level details go under the jobs folder.

(FAQ)If Jenkins does not have an internet connection, then how you install plugins?

Without the internet we cannot install a plugin, however, we can download the plugin from a machine we have internet and put it on Jenkins under the plugins folder.

Jenkins master-slave(faq)

1. Master-slave helps to scale Jenkins
2. We can add more slaves to the Jenkins server and distribute jobs across slaves and improve the performance of Jenkins.
3. Jenkins supports various types of slaves.
    a. Linux/Windows slaves

    b. Cloud Slaves
    c. Slaves can be in your kubernetes cluster.

Adding Linux slave

1. Get Linux server
2. Jenkins Home → Manage Jenkins → Manage Nodes & Clouds
3. Click on New Node

Jenkins Security

1. While setting up Jenkins we set up with the initial admin user.
2. By default users are created on the Jenkins machine.
3. A new user created in Jenkins is by default admin.
4. Jenkins can be integrated with third-party identity providers, in the real world we do not create users in Jenkins instead we integrate with LDAP, etc.

Matrix Based Security(faq)

This strategy helps in giving granular levels of permissions for Jenkins users.
We can add a user or group to the matrix and set permission levels.

Project-Based Matrix Authorisation Strategy(faq)

1. We can set permissions at the job level.
2. We can list the jobs specific to the user and hide other jobs.

Jenkins Pipeline Jobs

https://www.jenkins.io/doc/book/pipeline/

1. This is very very important for interviews and jobs
2. In earlier days we used to work with freestyle jobs, but the present trend is to use pipeline jobs.
3. Pipeline jobs are written in groovy DSL (Domain Specific Language)
4. Pipelines are introduced in Jenkins 2.0.
5. Pipeline jobs support two syntaxes(faq)
    a. Declarative Pipeline Syntax(We will learn this)
    b. Scripted Pipeline Syntax

Why Pipeline

Jenkins is, fundamentally, an automation engine that supports a number of automation patterns. Pipeline adds a powerful set of automation tools onto Jenkins, supporting use cases that span

from simple continuous integration to comprehensive CD pipelines. By modeling a series of related tasks, users can take advantage of the many features of Pipeline:

- Code: Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- Durable: Pipelines can survive both planned and unplanned restarts of the Jenkins controller.
- Pausable: Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
- Versatile: Pipelines support complex real-world CD requirements, including the ability to fork/join, loop, and perform work in parallel.
- Extensible: The Pipeline plugin supports custom extensions to its DSL [1] and multiple options for integration with other plugins.

Jenkins Tools Configuration

Using Jenkins tools configuration we can install and manage tools like maven, git, docker, etc.
Tools installed through this configuration is placed under $JENKINS_HOME/tools
The backup of Jenkins becomes straightforward.

Jenkins Shared Libraries

https://youtu.be/RvY5b--wnE0

- A shared library helps to develop reusable pipeline code, the code available in the shared library can be reused across pipeline jobs.
- Note: Every project has shared libraries

Working with Shared Libraries

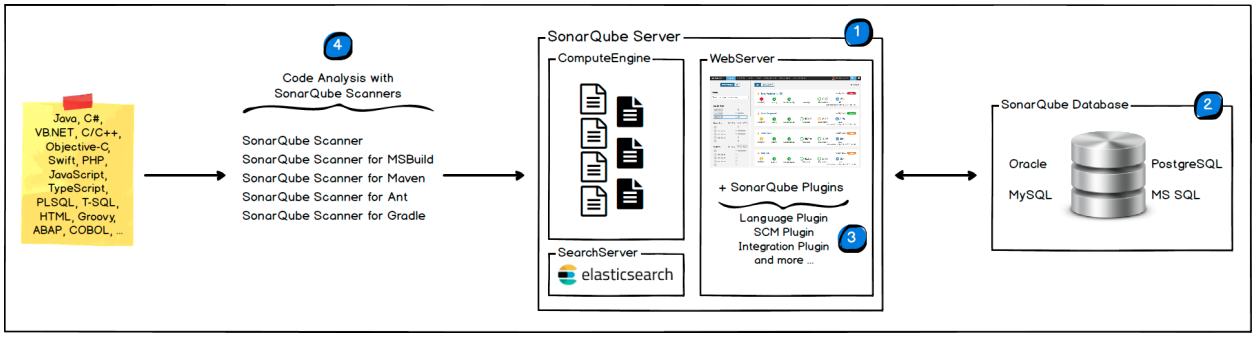
1. Create Git repository
2. We have to follow a specific folder structure for shared libraries.
3. https://github.com/javahometech/sharedlibs-202107
4. Configure shared library in jenkins

Working with Jenkins and Sonarqube

1. Sonarqube is a static code analysis tool, sonar takes source code and finds tricky bugs.
2. Sonarqube drives for code quality and security
3. It supports 27 languages
4. It scans for
    a. Code that is not maintainable
    b. Code that has performance issues
    c. Code that has security vulnerabilities
    d. Duplicate code blocks

e. Code that can cause deadlocks memory leaks
f. It can also check for test coverage
g. Etc.

Installing Sonarqube on Linux



System Requirements
4GM memory
2 CPU and sufficient amount of disk space

1. Install java
    a. sudo yum install java-1.8.0-openjdk.x86_64 -y
2. cd /opt
3. wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.1.zip
4. sudo unzip sonarqube-7.1.zip
5. sudo mv sonarqube-7.1 sonar7
6. sudo chown -R ec2-user.ec2-user sonar7
7. Run Sonarqube as a service
    a. https://docs.sonarqube.org/7.1/RunningSonarQubeasaServiceonLinux.html
8. Start Sonar
    a. sudo service sonar start
9. Check sonar status
    a. sudo service sonar status
10. Open sonar with browser, its default port is 9000
11. The default username and password is admin/admin

Integrating Jenkins with SonarQube

1. Install sonarqube scanner
2. Configure SonarQube Plugin

Mark Jenkins job as failed when Sonar analysis is failed

1. Jenkins job should wait for sonar analysis to complete
2. In sonarqube, we should configure a webhook that notifies Jenkins after the analysis is finished.

SonarQube quality Gates

Quality Gates enforces a quality policy in your organization by answering one question: is my project ready for release?
Create quality gate

How to run tasks in parallel in jenkins?

Use parallel steps in jenkins and run tasks in parallel.

Scripted Pipeline

https://youtube.com/playlist?list=PLH1ul2iNXl7txKuhhDMKenYOThDww6x8S

Declarative Pipeline

https://youtube.com/playlist?list=PLH1ul2iNXl7uHUaB1iaXf_lHb7tQDh7Tt

Running tasks on different agents?

Within a single pipeline we can use multiple agents

Jenkins Global Variables

Jenkins has following global variables

1. env
2. params
3. currentBuild
4. scm
5. docker
6. etc.

Using global variable we can fetch certain information dynamically for example,
BRANCH_NAME
JOB_NAME
JOB_URL
JOB_ID
Etc.

```
mail to: 'devops@acme.com',
    subject: "Job '${JOB_NAME}' (${BUILD_NUMBER}) is waiting for input",
    body: "Please go to ${BUILD_URL} and verify the build".
```

Jenkins Upstream and Downstream Jobs

In Jenkins a job can call another job, we call them upstream and downstream jobs.
For example a job after building and deploying to a test environment is called a selenium job.

(FAQ)What is Jenkinsfile?

Jenkinsfile contains pipeline code for the job, this file is kept along with your project source code in project root.
File name can be any arbitrary value, but recommended to have it as “Jenkinsfile”
File path can be anything, and the recommended location is project root.

Jenkins multi branch Pipeline

This is used for following scenario

1. We gonna maintain multiple branches like (dev,test,uat,master)
2. When the changes are integrated to a specific branch the code is deployed to that specific environment
    a. When a change is integrated to dev then deploy to dev environment
    b. When a change is integrated to test then deploy to test environment
    c. When a change is integrated to uat then deploy to uat environment
    d. When a change is integrated to master then deploy to master environment

Ansible Configuration Management

https://github.com/javahometech/ansible-tomcat-lb-mysql

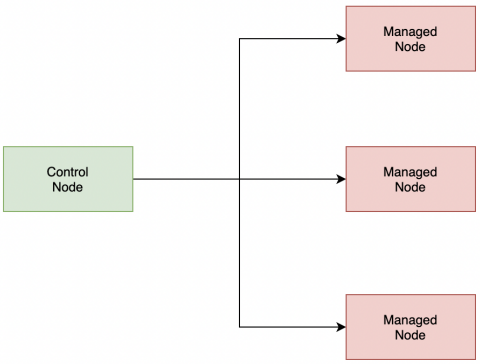
Other configuration management tools

- Chef
- Puppet
- AWS SSM (Systems Manager)
- Saltstack
- Etc.

(FAQ)Is ansible agent based or agent less?

Ansible is agent less

Ansible Architecture



Control Node

- The machine where ansible is installed is called control node
- You can use any machine with Python 2 (version 2.7) or Python 3 (versions 3.5 and higher) installed
- Windows cannot be a control node, Ansible can be installed on non-windows.

Managed Node

- Managed nodes are machines configured through Ansible
- You can use any machine with Python 2 (version 2.6 or later) or Python 3 (versions 3.5 and higher) installed
- Managed node can be windows and non windows

Install Ansible on Linux

Launch EC2 instance

- sudo amazon-linux-extras install ansible2
- ansible --version

Ansible Inventory File(FAQ)

- Is a file that contains details about managed nodes.
- The default location of this file is (/etc/ansible/hosts)
- If ansible wants to connect to a managed node, there should be an entry in inventory.

Onboard Managed Node

Add following details into inventory file

172.31.8.172 ansible_user=ec2-user ansible_connection=ssh ansible_ssh_private_key_file=~/nodes.pem

Checking Node Configuration

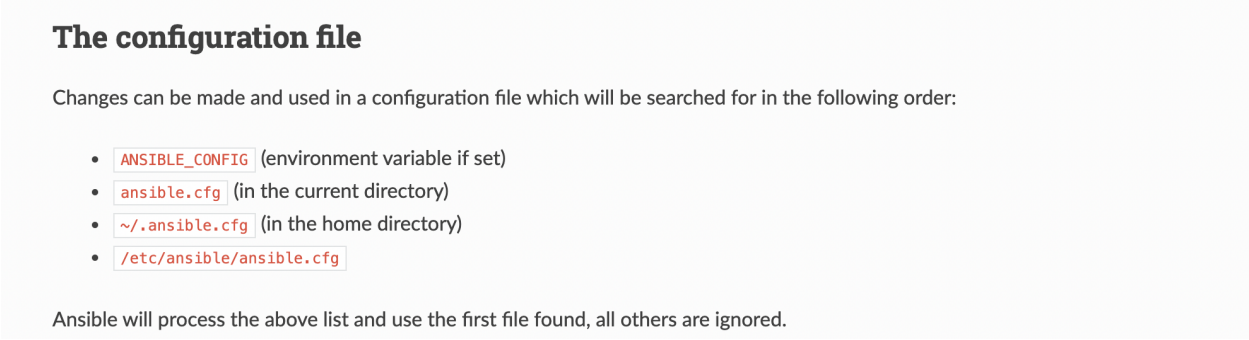
```
[ec2-user@ip-172-31-9-140 ~]$ ansible 172.31.8.172 -m ping
172.31.8.172 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ec2-user@ip-172-31-9-140 ~]$
```

Install git on managed node using ansible

ansible 172.31.8.172 -m yum -a 'name=git state=present' --become

Ansible Config File

1. The very first thing ansible does is, first looks for ansible.cfg file.
2. The default location is /etc/ansible/ansible.cfg
3. Ansible looks for configuration file in the following order



The configuration file

Changes can be made and used in a configuration file which will be searched for in the following order:

- `ANSIBLE_CONFIG` (environment variable if set)
- `ansible.cfg` (in the current directory)
- `~/.ansible.cfg` (in the home directory)
- `/etc/ansible/ansible.cfg`

Ansible will process the above list and use the first file found, all others are ignored.

Ansible configuration file contains lots of details, the following are some of them.

```
[defaults]
interpreter_python  = /usr/bin/python
# some basic default values...

#inventory      = /etc/ansible/hosts
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml
#forks          = 5
#poll_interval  = 15
#sudo_user      = root
#ask_sudo_pass = True
```

```
#ask_pass      = True
#transport      = smart
#remote_port    = 22
#module_lang    = C
#module_set_locale = False
```

Ansible ad-hoc command

An ad-hoc command is something that you might type in to do something really quick, but don't want to save for later.
Ad-hoc commands are helpful for troubleshooting and doing simple automations.
The real automations should be done through playbooks.

Ansible Modules

Every automation we do with ansible is through module, examples for ansible modules

- yum
- apt
- ping
- service
- files
- copy
- user
- group
- http
- docker
- Etc

Ansible Playbooks

Ad-hoc commands are handy and quick to work with, but we should prefer playbooks for most of the automations.

Ansible & Jenkins integration → https://youtu.be/PRpEbFZi7nI

YAML tutorial

Writing first playbook

We should know YAML for writing ansible playbooks

https://github.com/javahometech/july-ansible-2021

(FAQ)What are facts in Ansible?

Everytime we run playbooks, by default ansible gathers certain details about managed nodes before executing tasks, some of the facts it gather is

1. OS family
2. OS Architecture
3. OS Versions
4. Memory
5. IP
6. Host Name
7. Etc..

You can disable gathering facts if you save some time

(FAQ) In a playbook how to run only specific tasks?

Yes we can run specific tasks using tags

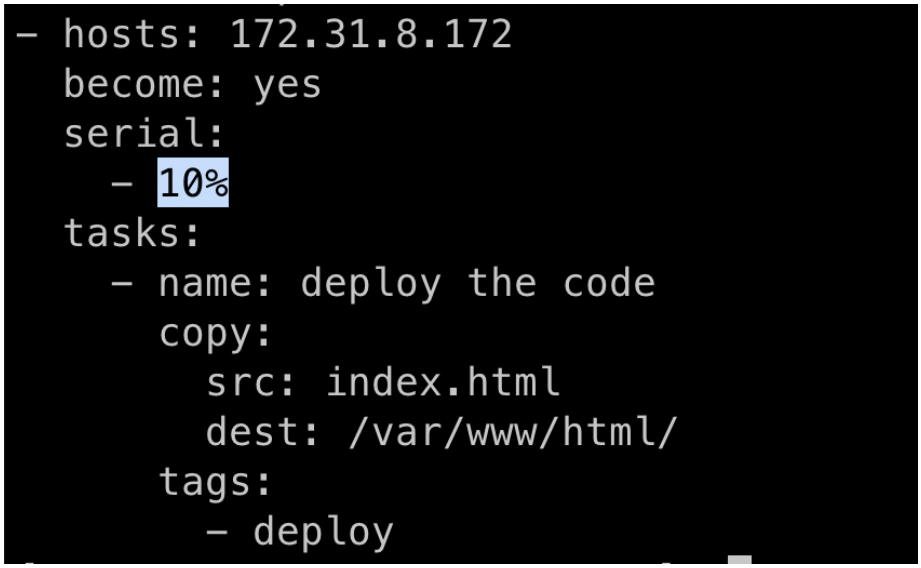
```
- hosts: 172.31.8.172
  become: yes
  tasks:
    - name: Install apache
      yum:
        name: httpd
        state: present
      tags:
        - install

    - name: start and enable apache
      service:
        name: httpd
        state: started
        enabled: yes
      tags:
        - start
    - name: deploy the code
      copy:
        src: index.html
        dest: /var/www/html/
      tags:
        - deploy
```

ansible-playbook tags-demo.yml --tags install,deploy
ansible-playbook tags-demo.yml --skip-tags install

(FAQ) How to perform rolling updates in Ansible?

We perform this operation using serial keyword




```yaml
---
- name: test play
  hosts: webservers
  serial:
    - "10%"
    - "20%"
    - "100%"
```

You can also mix and match the values:

```yaml
---
- name: test play
  hosts: webservers
  serial:
    - 1
    - 5
    - "20%"
```

(FAQ) We are running playbooks on hundreds of nodes, It failed on a few nodes, how to rerun only on failed nodes?

Use retry files, this must be enabled in ansible.cfg file, when playbook fails on certain machines those ips are stored in this file. We can rerun the playbook, pointing to this retry file so it runs only on failed nodes.

(FAQ)How to limit playbook execution to specific nodes? For example we are running playbooks on 100 nodes but this time we want to run it on 10 nodes.

ansible-playbook limit-demo.yml --limit all[0:1]
ansible-playbook limit-demo.yml --limit all[0]

Ansible Handlers (FAQ)

Handlers are tasks, they are executed only when they are notified.

```
[ec2-user@ip-172-31-9-140 ~]$ cat handlers-demo.yml
- hosts: 172.31.8.172
  become: yes
  tasks:
    - name: Install apache
      yum:
        name: httpd
        state: present
      tags:
        - install

    - name: start and enable apache
      service:
        name: httpd
        state: started
        enabled: yes
      tags:
        - start
    - name: deploy the code
      copy:
        src: index.html
        dest: /var/www/html/
      tags:
        - deploy
      notify:
        - restart
  handlers:
    - name: restart
      service:
        name: httpd
        state: restarted
```

Ansible Variables

```
- hosts: 172.31.8.172
  become: yes
  vars:
    - apache_port: 80
  tasks:
    - name: Install apache
      yum:
        name: httpd
        state: present
      tags:
        - install

    - name: start and enable apache
      service:
        name: httpd
```

```
      state: started
      enabled: yes
    tags:
      - start
  - name: deploy the code
    copy:
      src: index.html
      dest: /var/www/html/
    tags:
      - deploy
    notify:
      - restart
  - lineinfile:
      path: /etc/httpd/conf/httpd.conf
      regexp: '^Listen '
      insertafter: '^#Listen '
      line: 'Listen {{apache_port}}'
    notify:
      - restart
handlers:
  - name: restart
    service:
      name: httpd
      state: restarted
```

ansible-playbook variables-demo.yml -e apache_port=8080

(FAQ)How do you manage passwords, API tokens, and other sensitive values in Ansible?

Using ansible vault.

Ansible Vault

ansible-vault encrypt vars.yml
This above command prompts for password, set your new password

Configuring Vault Password While Running Playbook

ansible-playbook variables-demo.yml --ask-vault-pass

ansible-playbook variables-demo.yml --vault-password-file=~/vault.pwd

Encrypt specific values in the file

For more vault documentation
https://docs.ansible.com/ansible/latest/user_guide/vault.html

Ansible Inventory

```
[apache]
172.31.9.124
172.31.8.172
172.31.9.123

[apache:vars]
ansible_user=ec2-user
ansible_connection=ssh
ansible_ssh_private_key_file=~/nodes.pem

[dbs]
172.31.9.124
172.31.8.172

[dbs:vars]
ansible_user=ec2-user
ansible_connection=ssh
ansible_ssh_private_key_file=~/nodes.pem

[webdb:children]
apache
dbs

[somegroup]
www[001:100].example.com
```

(FAQ) What is dynamic inventory?

Sometimes we have to manage nodes in auto scaling provided by cloud platforms, they change dynamically, we need help with dynamic inventory.

Working with AWS dynamic inventory plugin

https://docs.ansible.com/ansible/latest/collections/amazon/aws/aws_ec2_inventory.html

- Boto3 and botocore are python dependencies for this plugin
- Install pip for installing above python dependencies
- curl https://bootstrap.pypa.io/pip/2.7/get-pip.py -o get-pip.py
- python get-pip.py
- pip install boto3

Configure AWS access to ansible, to fetch ips.

Create an IAM role form AWS and attach to the control node.
aws_ec2.yml

```
plugin: aws_ec2
regions:
  - ap-south-1
filters:
  tag:Name: ansible-mn1-july
```

Run ansible with dynamic inventory

```
ansible all -i aws_ec2.yml -m yum -a 'name=git state=present' --private-key ./nodes.pem
--become
```

aws_ec2.yml

```
---
plugin: aws_ec2
regions:
  - ap-south-1
strict: True
keyed_groups:
  - prefix: instance_type
    key: instance_type
```

```
ansible-inventory -i aws_ec2.yml --list
```

Dynamic Inventory Old Approach

https://youtu.be/JVZbzJZrBZ0

Setting up load balancer with Nginx

Nginx is a load balancer, reverse proxy and web server.

Ansible Role (FAQ)

Roles let you automatically load related vars, files, tasks, handlers, and other Ansible artifacts based on a known file structure. After you group your content in roles, you can easily reuse them and share them with other users.

Roles follow standard folder structure, it looks as follows

```
ans-app/
`-- roles
    `-- loadbalancer
        |-- defaults
        |   `-- main.yml
        |-- files
        |-- handlers
        |   `-- main.yml
        |-- meta
        |   `-- main.yml
        |-- README.md
        |-- tasks
        |   `-- main.yml
        |-- templates
        |-- tests
        |   |-- inventory
        |   `-- test.yml
        `-- vars
            `-- main.yml
```

Install nginx on linux server

Docker and Kubernetes

Before we discuss docker lets learn monolithic and microservices architecture.

Monolithic Architecture.

1. Here the whole project is a single big code base.
2. Because of one big code base there are lots of challenges.
3. Making even small changes to the project is tough and time taking.
4. We may have to use different technology stacks to scale the applications and that is not feasible.

5. If a system goes down the whole system is down.
6. Scaling a monolith is expensive, if let's say a search engine has performance issues and we can't scale only that.

Microservices Architecture

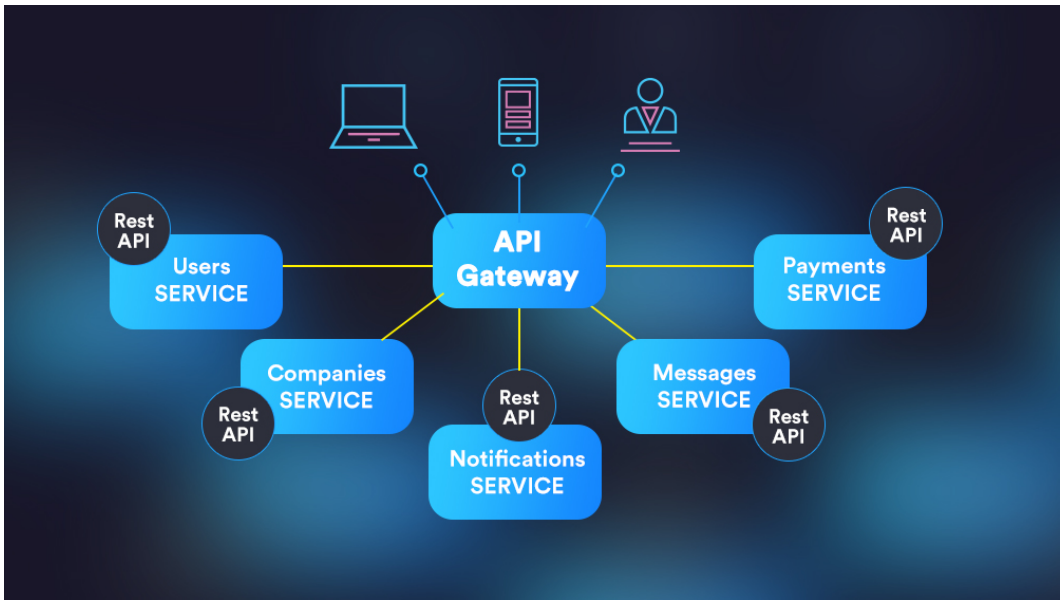
1. Breaking monolith into smaller projects(services).
2. Each service should be small enough so that it can be managed by 2 pizza teams
3. Every service is going to have its own backend.
4. Each service can have its own technology stack
5. We can scale each service separately.
6. Newly onboarded person does not need much time to understand the system and add value.
7. We should not have cross-db communications, if service-a wants something from service-b, it has to communicate through API.

Monolithic

1. Every Friday we do a release.

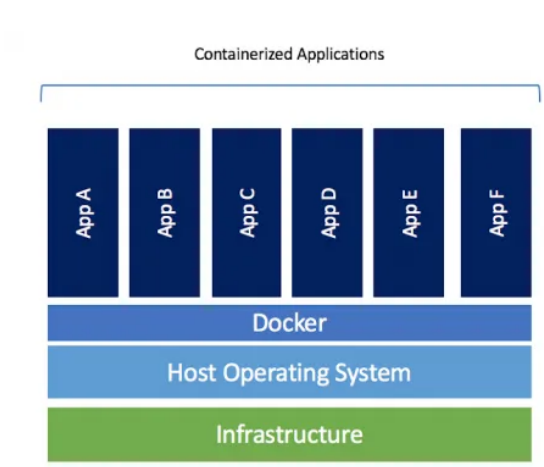
Microservices

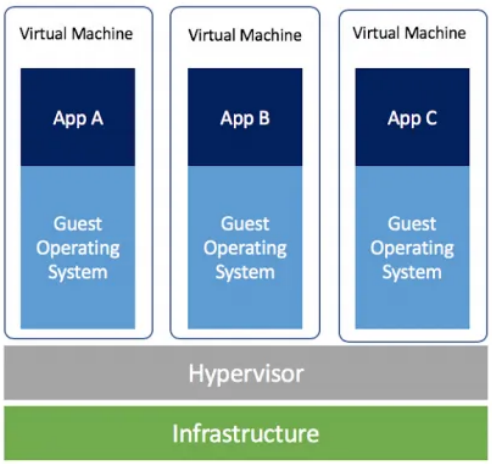
1. Let's say we have 50 services
2. Every friday we do a release
3. Every Friday we do 50 releases.



RESTful API

Docker vs Virtual Machines





- Launch Linux Machine
- Install docker
- sudo yum install docker -y
- sudo service docker start
- Run nginx container on the host

Install Docker On Linux

- sudo yum install docker -y
- sudo service docker start
- sudo chkconfig docker on
- sudo usermod -a -G docker ec2-user

Running Docker container

docker run -d -p 80:80 --name nginxdemo nginx:latest

-d → Detached mode, that is, run the container as a background process.
-p → It stands for port mapping, host-port:container-port
→ Host port, any available port on vm
→ Container port is the port used by nginx on the container, that is if nginx is configured on docker on 80 then container port must be 80

To access container from outside its host, we should do port mapping

Containers run on a different virtual network, so to access it from the host we should do port mapping.

Dockerizing Applications

DevOps should take care of dockerizing applications.

1. We should write Dockerfile.
2. Dockerfile contains instructions like install java, download tomcat, copy war file to tomcat, export port and start tomcat etc..
3. We have to create a docker image from Dockerfile.
4. Using docker images we create hundreds of docker containers.

https://github.com/javahometech/springmvc/blob/feature/hari/dockerfile/Dockerfile

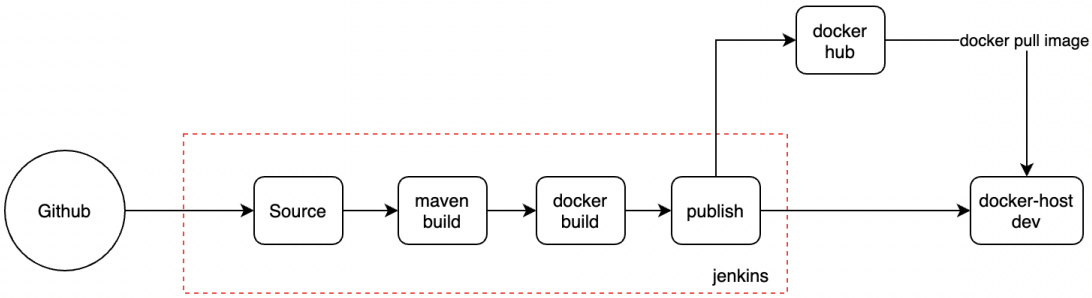
(FAQ) What is the difference between RUN and CMD?

1. Both are used for running containers.
2. RUN is docker build time instruction, that is used for things like installing java, tomcat etc..
3. CMD is docker run time instruction, that is used for running applications at container start time.

(FAQ) What is the difference between CMD and ENTRYPOINT?

1. We can override CMD instruction at docker run time

Docker CI/CD Demo




```
pipeline{
    agent any
    environment{
        DOCKER_TAG = "${getLatestVersion()}"
    }
    stages{
        stage("Source"){
            steps{
                git 'https://github.com/javahometech/my-app'
            }
        }
        stage("Maven Build"){
            steps{
                sh "mvn clean package"
            }
        }
        stage("Docker Build"){
            steps{
                sh "docker build . -t kammana/2021app:${env.DOCKER_TAG}"
            }
        }
        stage("Docker Hub Push"){
            steps{
                withCredentials([string(credentialsId: 'docker-hub', variable: 'hubPwd')]) {
                    sh "docker login -u kammana -p ${hubPwd}"
                    sh "docker push kammana/2021app:${env.DOCKER_TAG}"
                }

            }
        }
        stage("Deploy To Dev"){
```

```
        steps{
            withCredentials([string(credentialsId: 'docker-hub', variable: 'hubPwd')]) {
                sshagent(['docker-dev']) {

                    sh "ssh -o StrictHostKeyChecking=no ec2-user@172.31.6.59 docker rm -f
javahomeapp"
                    sh "ssh ec2-user@172.31.6.59 docker image prune -a -f --filter
'label=app=my-app'"
                    sh "ssh ec2-user@172.31.6.59 docker run -d -p 8080:8080 --name javahomeapp
kammana/2021app:${env.DOCKER_TAG}"
                }
            }

        }
    }
  }
}

def getLatestVersion(){
   return sh(returnStdout: true, script: 'git rev-parse --short HEAD')
}
```

Dockerfile Instructions

https://docs.docker.com/engine/reference/builder/

Exercises

1. Dockerize spring boot application and run container locally or on any docker host.
   https://github.com/javahometech/spring-boot
   Dockerizing reference → https://spring.io/guides/gs/spring-boot-docker/
2. Implement CI/CD of above dockerized application using jenkins pipeline.
3. Dockerize nodejs application
   https://nodejs.org/en/docs/guides/nodejs-docker-webapp/
4. Implement CI/CD of above dockerized application using jenkins pipeline(optional)

Docker Networking

Docker networking works differently on single nodes and multi nodes. Now lets explore docker networking on single node.
Docker uses bridge network by default on a single node, containers spinned up on same host implicitly joins bridge network.

Container joining same bridge network can communicate

Bridge network Demo

Run container one → docker run -itd --name c1 alpine
Run container two → docker run -itd --name c2 alpine
Inspect bridge network → docker inspect bridge
Keep note of container two IP
Login to c1 → docker exec -it c1 ash
Run ping ip-of-container-two

Custom Bridge network

We can create a custom network using bridge driver, this supports we can access containers over its IPs and names.

https://youtu.be/XtAcytyy8V0

Docker Volumes(FAQ)

Docker file systems are ephemeral, data stored on docker filesystem is lost when a container is terminated.
If we are deploying stateful applications(jenkins, Nexus, Databases) on containers we have to retain data when the container crashes.
We use docker volumes to persist container data on to the docker host.

Docker Volumes Demo

docker volume list
docker volume create myapp
docker run -itd --name alpine -v myapp:/myapp alpine
docker exec -it alpine ash

(FAQ) How can two containers running on the same host share a file?

It could be done through volumes, running two containers and mounting them onto same volume.

Exercise

1. Create a volume with your name, run an alpine container with your name by mounting the volume you just created, login to the container, create a file, exit from the container and check if the file exits from the host volume.
2. Run two alpine containers with names alpineone and alpinetwo, create a file “info.txt” on alpineone under /data, when we login to alpinetwo we should see info.txt file.
3. Check types of docker volumes from docker docs.

Docker Videos

1. Docker basic commands
    a. https://youtu.be/h8n0dhrJuqg
2. Docker Compose(FAQ)
    a. https://youtu.be/SVg197AkKMU

Kubernetes

Kubernetes is a container orchestration tool, its popular, open source tool. Kubernetes is Google's product and they made it opensource. Google has been running production workloads on kubernetes since 15 plus years.

Today all modern applications are containerized and those containers must be orchestrated using popular tools like kubernetes.

Other container orchestration tools in the market

1. Kubernetes
2. AWS ECS (Elastic Container Service)
3. Apache Mesos
4. Docker Swarm

What do kubernetes offer?

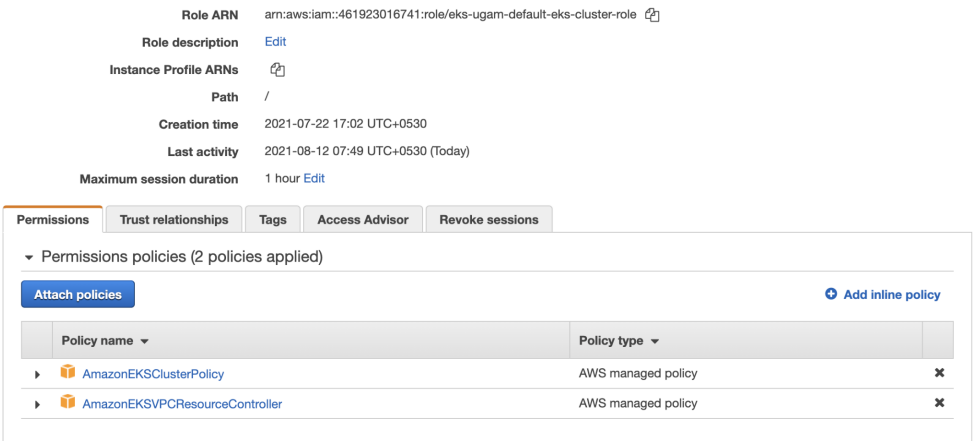
1. It does scheduling of pods (containers)
2. Kubernetes offers autoscaling of Pods
3. Kubernetes offers service level load balancing
4. Kubernetes can restart Pods if they are not healthy
5. Kubernetes can create a number of Pod replicas.
6. Automated container deployments.
7. Lots more.

We want to use kubernetes, how to set it up?

1. We can install kubernetes anywhere (on premise, cloud, etc..)
2. There are different ways to install and setup kubernetes.
3. Directly installing and using kubernetes involves more effort. To simplify this there are kubernetes platforms.
4. Popular kubernetes platforms are
    a. AWS EKS (Elastic Kubernetes service)
    b. AKS (Azure Kubernetes Service)
    c. Google Kubernetes Engine
    d. Red Hat Openshift
    e. Rancher Labs
5. To directly install kubernetes without platform
    a. Kops, install kubernetes on AWS
    b. Kubeadm, this is to install kubernetes anywhere.
    c. Kubespray

Setting Up kubernetes cluster on AWS using EKS (cost applies)

Create a role for the EKS cluster.



What is Pod in kubernetes

- Pod is an atomic unit of work to deploy applications, Each Pod gets private IP in the cluster, Pods are reachable from anywhere within the cluster.
- In a Pod we can run one or more containers, often we run pods with a single container, but there are certain use cases to run 2 or more containers.
- Containers in the same pod can communicate over localhost.
- Containers in the same Pod should run on different ports.
- There will be multiple pods on the same node.

Authenticating to kubernetes cluster from laptop

https://docs.aws.amazon.com/eks/latest/userguide/create-kubeconfig.html

1. Install AWS CLI
    a. https://aws.amazon.com/cli/
2. Install Kubectl
    a. https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html
3. Configure AWS Access Keys and Secret Keys on the laptop
    a. IAM
    b. Create User, with programmatic access
    c. Attache Administrator access
    d. Create user, and grab access keys and secret keys
4. Configure your laptop with aws keys
    a. aws configure
5. Run following command
    a. aws eks --region ap-south-1 update-kubeconfig --name javahome-eks

Deploying our first application into Kubernetes

kubectl run nginx --image=nginx:1.21.1
kubectl get pods (to get all pods)
kubectl get pods -o wide

Accessing Pods

Accessing pods within the cluster is possible, but to access pods from outside the cluster we should use service objects.

Note: to access pods from local host, run this following command
kubectl port-forward pod/nginx 8080:80
Open browser and type localhost:8080, you should see a response from Nginx.

Exercise

1. Create a nginx pod using kubectl
2. Check its status and node its running on
3. Do port-forward and access it from your browser.

(FAQ)How do you troubleshoot pods?

We could use describe pods and check events, and we also can use pod logs

Kubernetes Example

https://github.com/javahometech/kubernetes

Logging into the container

kubectl exec -it nginx -- env

Deploy pod using YAML document

```
apiVersion: v1
kind: Pod
metadata:
  name: nodeapp
  labels:
    app: nodeapp
spec:
  containers:
    - name: nodeapp
      image: kammana/nodeapp:v1
      ports:
        - containerPort: 8080
```

Kubernetes Service Objects

https://kubernetes.io/docs/concepts/services-networking/service/

Kubernetes Replicaset

For example I want to deploy a microservice with 10 replicas, ReplicaSet does this for us, and if we are short of replicas it creates and if there are more it terminates.

https://github.com/javahometech/kubernetes/blob/master/replicaset/rs.yml

Kubernetes Deployment Objects

1. It has replicaset
2. It has rolling updates
3. It has undo rollout

(FAQ) How to schedule pods to specific nodes?

Kubernetes cluster has a default scheduler, its job is to find the nodes to create pods, we can customise the scheduling stuff. By using, node affinity, taints, tollerations etc..

Node Affinity

https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes-using-node-affinity/

(FAQ) How do you prevent scheduling pods on master nodes?

Use taint and say NoSchedule

https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/

(FAQ) What is the architecture of kubernetes?

Kubernetes cluster has master and nodes
Master:

- Master node will run the components like api-server, etcd, controllers, schedulers etc…
    - Api-server is the gateway for kubernetes cluster, the interactions with the cluster happens through API server.
    - Scheduler is responsible for scheduling pods on to the nodes
    - etcd , is a distributed key-value store, which stores cluster state.
    - Controllers, like replicasets etc..
- Node
    - Node is like a worker, pods run on Node, optionally pods can run on master.
    - Kubelet, is the agent on the node, it talks to master periodically, for instance a pod crashed, this information is conveyed to master, and master will reschedule it.
    - Kube-proxy, this is responsible for networking in kubernetes.

Performing Health Checks on Applications running in the Pod

https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/

In Kubernetes we can configure different types of health checks so that pods can auto heal.

1. Readiness probe
    a. We should give some time for containers in the pod to start and the applications in the container to become ready by reading lots of configuration files.
    b. Until this we should allow requests to flow in.

    c. Only after the readiness probe succeeds the pod is added to the service otherwise it is not.
2. Liveness probe
    a. https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/

Kubernetes Volumes

- File system of containers running inside pods are ephemeral, they get deleted when the pod is terminated.
- To run stateful applications we should use volumes
- Kubernetes volumes help in storing data on Nodes filesystem or EFS, NFS etc…

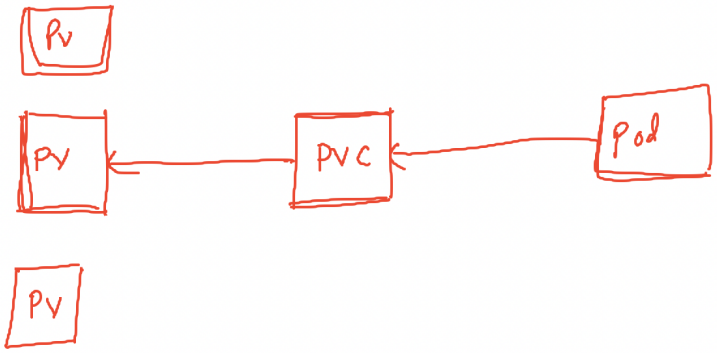
What are stateful applications?

Applications which generate data at runtime are stateful applications,
Jenkins, databases are stateful applications.

Types of volumes in Kubernetes

emptyDir, this is useful if containers in a pod want to share data, this is created when pod is created and deleted when pod is deleted.

Persistent Volume(PV) & Persistent Volume Claims (PVC)



Dynamic and Static Provisioning

1. In static provisioning kubernetes admin creates volume explicitly.
2. In dynamic provisioning kubernetes uses storage class, this contains the details about what type of storage to create.

How do you manage secrets in kubernetes?

In the kubernetes cluster we may have to deal with API tokes, database passwords, etc. we manage them securely using secret objects.

There are different types of secrets

https://github.com/javahometech/kubernetes/tree/master/secrets

Create kubernetes secret to store database username and password

(FAQ)How do you restrict front end pods directly communicating with DB?

Ans) We can achieve this using network policies.
https://kubernetes.io/docs/concepts/services-networking/network-policies/

(FAQ)How do you package kubernetes resources?

Ans) We use helm for packaging and deploying kubernetes resources.

(FAQ)Where are you storing your helm charts?

Ans) We are storing them in sonatype nexus.

The Helm official document

https://helm.sh/

Install Helm

https://helm.sh/docs/intro/install/

Create first helm chart

helm create nginx

Grafana Prometheus

https://youtu.be/7KxVSK9yP-Y