# DevOps Frequently Asked Question

1. Do you need a development background?
   1. No, however we some automation in devops
2. What should be the education background?
   1. Any education, minimum graduation is needed.
3. Is there a prerequisite for this course?
   1. No, We cover everything from scratch. *The key is practice*
4. Can freshers learn and get placed in DevOps?
   1. Yes, we placed freshers on DevOps and AWS.
5. I’m currently working on testing for 8 years, and can I apply for DevOps?
   1. Yes, 5 years of testing experience and 3 years of DevOps experience.
6. My experience is 8 years and my CTC is 8 lakhs, I want 20 lakhs, is that possible?
   1. Yes, It is possible
7. Is Linux required for DevOps?
   1. Yes, We will cover linux essentials for DevOps

# DevOps Job Market

1. Most trending in the market
2. Huge opportunities.
3. Highly paid jobs.

# How build and release used to happen when devops wasn’t there

1. Lengthy release cycles, usually 6 months 1 year.
   1. Time to the market is more
2. Teams followed waterfall model
3. Code integration is done at the end of the development phase.
   1. Merging code is tough, and error prone
   2. We loose productivity
   3. It breaks the code and introduces bugs
4. No coordination and no collaboration
5. Build and Release are done manually
6. Releases are not stable because it's done manually.

# What is DevOps?

DevOps is a combination of tools, practices, and philosophies which enables organizations to release software rapidly. Those releases should be consistent and stable.

# How DevOps work

1. Follow agile development, keep release cycles shorter usually 1 week or 2 weeks.
   1. Time to the market is fast
   2. Team work on small tasks
   3. Integrating changes will be fast
   4. We will not lose productivity
2. Improve collaboration
3. Automate build, release activities so it improves productivity.
4. Continuous Integration
   1. It is a development practice where developers should commit and integrate changes often, at least once in a day. This is picked by an automated process and does the following.
      1. Take latest code
      2. Build the code using build
      3. Scan for vulnerabilities
      4. Do static code analysis.
      5. Upload artifacts to nexus/jfrog
   2. Benefits of CI is that it is verified instantly, and the feedback goes to the developer immediately. Because of faster feedback, fixing a bug takes less time.
5. Continuous Deployment
   1. This begins after CI completes, this is an automated process which deploys artifacts from development to production.
6. Continuous Delivery
   1. This is also an automated process to deploy from development to production but before deploying to production it needs to be approved.
7. Configuration Management
   1. For example as a devops engineer I got 100 vms, and I’ve to install and configure java9 and tomcat8 on all of them.
   2. I can automate this process using configuration management tools like Ansible/chef/Puppet.
   3. I want to patch 300 linux servers every 6 months. This can be automated using configuration management tools.
   4. I want to deploy a war file on 100 servers, this can be done using configuration management tools.
8. Continuous Monitoring
   1. All our servers should be continuously monitored, and we have to make sure they are healthy and stable.
   2. Tools for monitoring are grafana prometheus, ELK (Elasticsearch Logstash, Kibana), Nagios, Zabbix, Splunk.
9. Docker and Microservices.
   1. Microservice architecture is about breaking big applications into smaller independent applications.
   2. Those services are logically combined to make our actual application.
   3. In microservices architecture each service should be small enough so that 2 developers can handle it.
   4. Every microservice will have its own technology stack and its own codebase.
   5. Every service will its own CI/CD
   6. We can release more features and it improves developer and devops productivity.

Source Code Management Tool(SCM)

1. We need a source code management tool for effectively managing, integrating and releasing stable code.
2. SCM tools maintain history of changes that happened to the files, using history we can troubleshoot and debug issues.
3. These tools also provide security to the code, only authorised users can access and make changes.
4. SCM is a developer tool. It's also a devops tool, every devops engineer should know how to use SCM for their scripts.
5. The Version Control System maintains multiple versions of files.

## Which SCM tool to use?

There are so many SCM tools in the market.

1. Git is open source, widely used version control system
2. SVN
3. Clear Case
4. Microsoft TFS (Team Foundation Server)

## Git is offered by different companies

1. Github
2. Bitbucket
3. Gitlab (We use this in current project)
4. AWS CodeCommit
5. Azure repos
6. Etc..

Note: No matter which offering of git you are using, git is git.

## Setting up git server

We can setup git server in two ways

1. Cloud
   1. Servers are managed by third parties, we use them over the internet.
2. Hosted Git
   1. We install and configure git in our own environment.
   2. We are using hosted git
   3. ***We are running gitlab on EC2 as a docker container***.
   4. Our team has written an ansible role for setting up gitlab.

## Setting up git server for our learning

1. Create account in Github.com

## Setting Up git client on the local

* We need git client to work with client
* There are wide variety of options for git client
* There are CLI based clients
* There are GUI based clients

## Install git bash on the local

<https://git-scm.com/downloads>

## Create git project for learning

Got to github.com → create repository → name → cool-app → public → create

## Working on above project

To work on the above project I should get remote copy to my local

git clone <https://github.com/javahometech/cool-app.git>

Youtube Videos From Java Home

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

## Install Visual Studio Code

<https://code.visualstudio.com/download>

While installing, make sure you are adding visual studio to the path.

Open Visual studio in your local git repository.

## Configure Git for first time

1. You must configure user name and email
   1. git config --global user.name "Hari Kammana"
   2. git config --global user.email "hari.kammana@gmail.com"

## Add new file to git project

1. Create new file hari.txt
2. Add your name and location, save the file
3. Commit changes to local repository
   1. Stage the files
      1. git add hari.txt
   2. Commit files
      1. git commit -m ‘demo commit’
4. Push changes to remote
   1. git push origin main

## 

## Working Area

This is a virtual folder maintained by git, all changes should go through this area.

## Staging Area

After making changes, the file sits in the working area, we have to stage the files that need to be commited.

Commit command picks only the files in the staging area.

## Checking Commit History

* git log (repo history)
* git log hari.txt (history of one file)
* git log --oneline hari.txt
* git log --oneline -10 hari.txt (show recent 10 versions)
* git show commit-id (Show, which files are commited)

## Git diff command

* git diff (Show differences between working area and local repository)
* git diff --color-words (show differences between working area and local repository)
* git diff --staged --color-words (compare diff between staging area and local repository)
* git diff 369674e 681a9e02 --color-words hari.txt

## Git Videos

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

## Git Fetch vs Git pull (FAQ)

Git Fetch

* Fetch changes from remote to local, and do not merge.

Git Pull

* It does fetch plus merge

## Git Branches

* Branch is an isolation for your work
* Git by default comes with master branch (Github has changed this to main recently)
* Master branch should always have clean and well tested code, so we should not directly commit changes to master.
* Any change to the git repository should go with a branch.
* Branch is lightweight in git, in other tools like SVN it is heavy

## Branch Demo

1. git branch feature-1 (create new branch)
2. git branch (shows list of branches)
3. git checkout feature-1 (switch branch)
4. git branch -d feature-1 (delete local branch)

Exercises

1. Create a branch ‘mytask’ from main
2. Switch to mytask branch
3. Add dummy changes and commit
4. Push it to remote
5. Create pull request
6. Merge pull request
7. Delete mytask branch from remote and local

## Git Undo Operations

### Undo changes in working area

git restore hari.txt infor.txt

git restore \*

### Unstaging file

git restore --staged hari.txt infor.txt

git restore --staged \*

### Undoing local commit

A local commit is a commit that is not yet pushed to remote.

git reset 8ae36bc

Commits above this commit id “8ae36bc” will be removed from local repository

There are three variants of reset

* Soft
  + Remove local commit and changes in that commit is placed in staging area
* Hard
  + Remove local commit and permanently discard changes
* Mixed
  + Remove local commit and place changes in working area

## How to undo remote commits?

Git revert will undo changes in the commit and makes new commit

* Git revert (commit ID)

## What is HEAD?

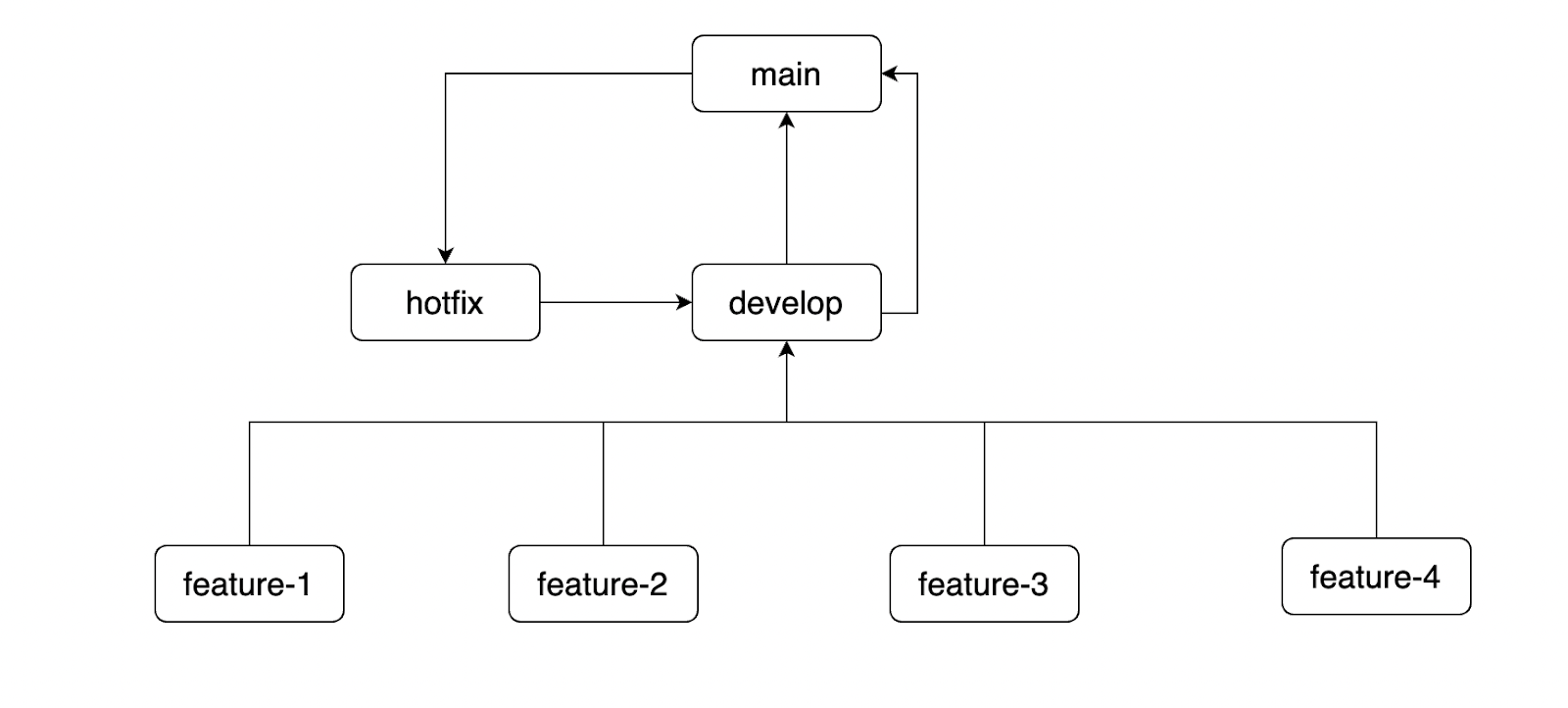
HEAD is a lightweight pointer, which points to the latest commit in the current branch.

## What is the origin in git?

Git uses origin as alias name for remote repository, origin points to URL of remote repository.

git remote -v

## Git Branching strategy(faq)



We should keep branching strategies simple.

1. We are using feature branches to add new features, feature branches are created from develop and they are deleted after merging to develop.
2. develop branch is used for integrating feature branches, code from this branch is used to deploy in dev and test environments.
3. Main/Master, this branch is used for production deployments.
4. Hotfix is used for fixing production defects. This branch is created from main and merged to develop and merged to main.

## (FAQ)Git cherry pick

Cherry pick helps to pick specific commits from a different branch and integrates with the current branch.

## (FAQ) What is git blame

This command presents line by line changes made by the author.

## (FAQ) What is a bare repository?

Bare repositories will not contain working trees (working area and staging area), so we can’t make changes directly.

## Git Real Time Task

There is one repository in development git server, that needs to be migrated to production git server. How?

## Git Stash (faq)

Stash saves the changes in the working tree into a separate location, and it keeps the working tree clean.

## Git Rebase (faq)

Rebase is a type of merge

Rebasing is the process of moving or combining a sequence of commits to a new base commit.

<https://www.atlassian.com/git/tutorials/rewriting-history/git-rebase>

## Git Init command

Git init command converts a regular folder on your laptop into git repository, let's say we have a folder and few files init, we want to keep this folder as a repository in git, then do following

1. Run git init command in the folder

## .git folder in git

Git manages all its internals under this folder

## Git Hooks

<https://www.atlassian.com/git/tutorials/git-hooks>

## Git tags

<https://www.atlassian.com/git/tutorials/inspecting-a-repository/git-tag>

## DevOps Engineer tasks

1. Find how to scan the git repository for passwords, API tokes and other sensitive information.
   1. Check git guardian
2. Find how to remove passwords from git repositories.
   1. There are two tools to remove sensitive data from repository
      1. BFG Repo Cleaner
         1. https://rtyley.github.io/bfg-repo-cleaner/
      2. Git Filter Repo
         1. https://github.com/newren/git-filter-repo
      3. Talisman
         1. https://github.com/thoughtworks/talisman#what-is-talisman
3. Is there a way to prevent users from committing sensitive informations

## Github User Management

1. Github server is integrated with Microsoft AD (Active Directory)
2. We can create groups

## Build Tools

Build tools take source code as an input and convert them into deployable artifacts.

Any build tool performs following tasks

1. Create deployable folder structure
2. Download dependencies
3. Compile the code
4. Run test cases
5. Create an artifact

## Different Build tools

1. Maven
   1. For Java
2. MS Build
   1. For .Net projects
3. Gradle
   1. For Java
   2. For Android

## Launch Virtual Machine for Maven

1. Let's sign up for a free AWS cloud account.
2. <https://aws.amazon.com/free>
   1. This requires credit or debit card
   2. Follow the above URL and finish creating a free account.
3. Launch Virtual Machine (VM)
   1. Choose EC2
4. Connect to EC2 Linux server
   1. To connect to remote linux machines we should SSH
   2. For windows people
      1. You have to install SSH client
         1. Putty
         2. Mobaxterm
   3. For mac people
      1. SSH is built in, you don’t have to install any special software.
      2. Open terminal

## Install maven on amazon linux

<https://github.com/javahometech/my-app>

1. Maven is written in java, it is open source
   1. sudo yum install maven -y
2. Install jdk
   1. sudo yum install -y java-1.8.0-openjdk-devel
3. Install Git
   1. sudo yum install git -y

## Maven Details

1. It downloads dependencies from maven central repository
2. Maven uses pom.xml in the project for getting details like dependencies
3. First time it downloads from the central repository and next time onwards it takes from the local repository.
   1. ~/.m2/repository/

## (FAQ) How to skip test cases in maven

mvn package -DskipTests=true

## (FAQ) What is maven build lifecycle?

1. Validate
2. Compile
3. Test
4. Package
5. Verify (needs to configured)
6. Install, copy the package to local repo
7. Deploy, uploads artifacts to remote repository like Jfrog, Nexus3

# What is CI/CD(FAQ)

https://aws.amazon.com/devops/continuous-delivery/

# Jenkins

* Build great things any scale
* The leading open-source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.
* Anything and everything can be automated

## Install Jenkins on Linux server

<https://pkg.jenkins.io/redhat-stable/>

<https://stackoverflow.com/questions/68806741/how-to-fix-yum-update-of-jenkins>

* sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
* sudo rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io.key>
* sudo yum install epel-release # repository that provides 'daemonize'
* sudo yum install java-11-openjdk-devel (for aws-ec2 use 1.8.0)
* sudo yum install java-1.8.0-openjdk-devel
* sudo yum install jenkins -y
* sudo service jenkins start
* sudo chkconfig jenkins on
* sudo service jenkins status

## where jenkins get installed on system? (FAQ)

* var/lib/jenkins

## The initial setup and configuration of Jenkins

1. Access jenkins application over web browser on 8080
2. Follow instructions

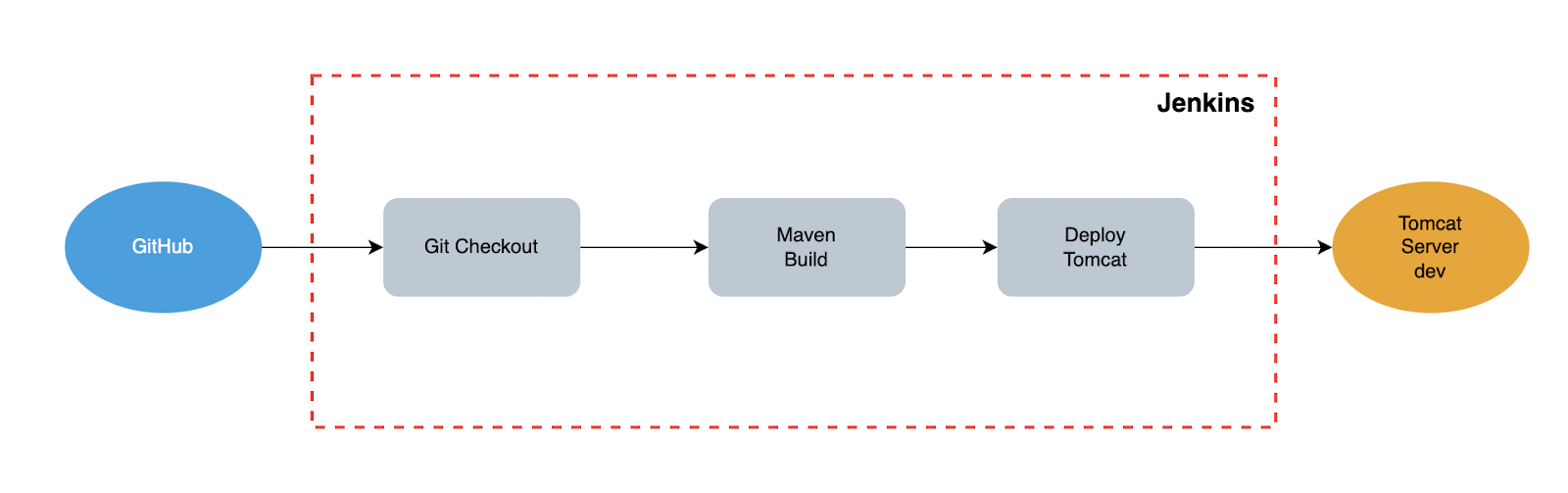
## Jenkins Job types

1. We can write jenkins job using freestyle or pipeline
2. Freestyle jobs are a legacy approach, no one uses it now.
3. Pipeline jobs are the standard approach.
   1. Pipeline jobs are written using groovy

## Write hello world freestyle job

1. This job should clone the code from git
2. Build the code using maven

## Jenkins Job to build and deploy to tomcat



1. Checkout code from Git
2. Build using maven
3. Deploy to tomcat

## Web Applications

Web applications are applications which are accessed using http protocol, examples for web applications are, facebook, twitter, linkedin, etc.

Web applications are developed using various technologies like java, .Net, python, php, node, etc…

## Web servers

We need web servers to run web applications, depending on the programming language developers choose.

For Java

* Tomcat
* Weblogic
* JBoss
* IBM Websphere
* Sun glassfish
* Etc.

For .Net

* IIS (Internet Information Server)

For PHP and python

* Apache

## Install tomcat on Linux

Tomcat is written in java and it run java applications so java is a dependency

* sudo yum install java-1.8.0-openjdk -y
* cd /opt
* sudo wget <https://dlcdn.apache.org/tomcat/tomcat-8/v8.5.73/bin/apache-tomcat-8.5.73.tar.gz>
* sudo tar -xzvf [apache-tomcat-8.5.73.tar.gz](https://dlcdn.apache.org/tomcat/tomcat-8/v8.5.73/bin/apache-tomcat-8.5.73.tar.gz)
* sudo rm apache-tomcat-8.5.73.tar.gz
* sudo mv apache-tomcat-8.5.73 tomcat8
* *Configure tomcat as a service (This you check on internet)*
* sudo chown -R ec2-user.ec2-user tomcat8/

### Start tomcat

/opt/tomcat8/bin/startup.sh

### Stopping tomcat

/opt/tomcat8/bin/shutdown.sh

## Deploy to tomcat

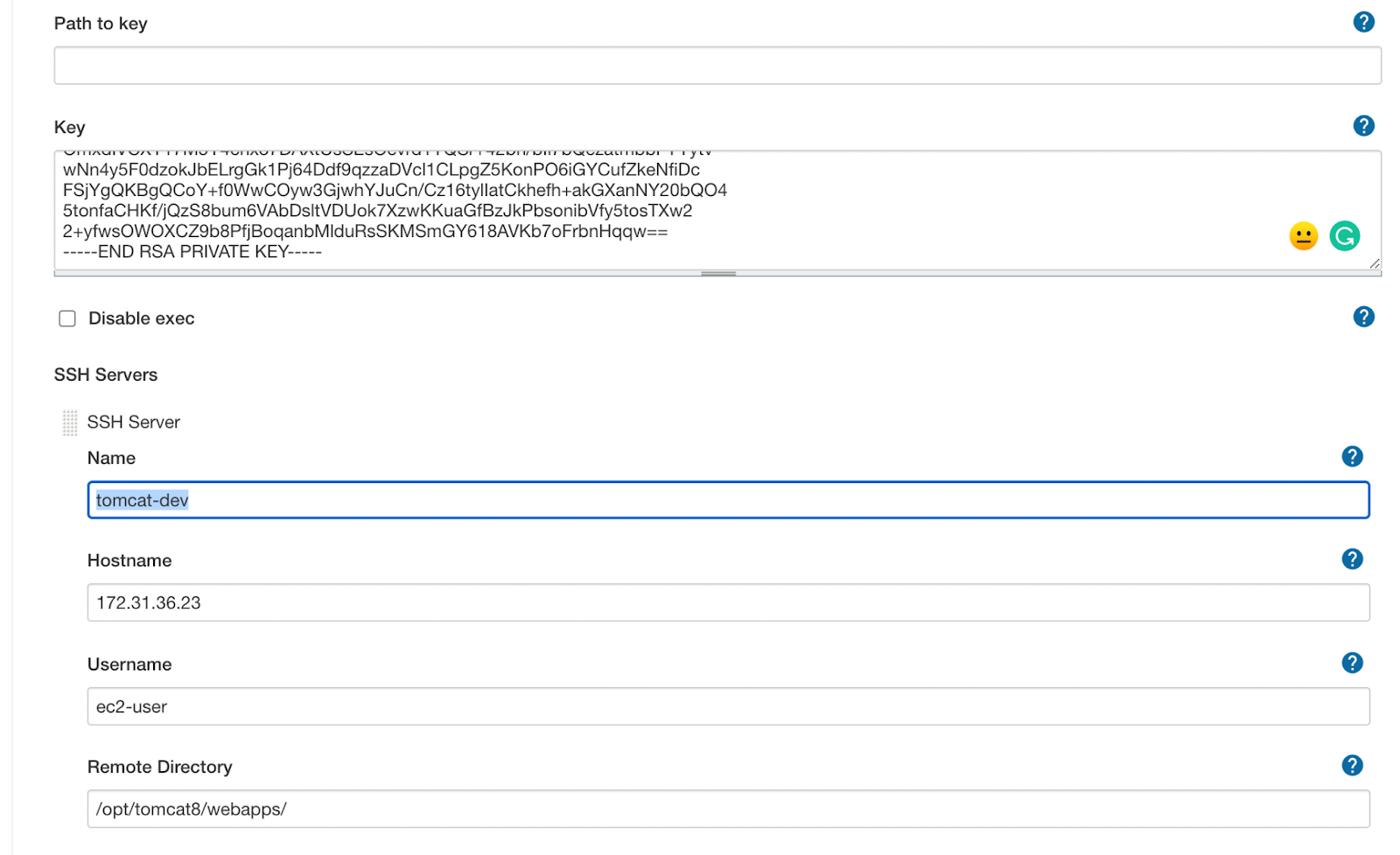
* This is deployment folder on tomcat /opt/tomcat8/webapps
* Place the war file under webapps and restart tomcat, your application should be deployed.

## Jenkins Job to deploy war file to remote tomcat server

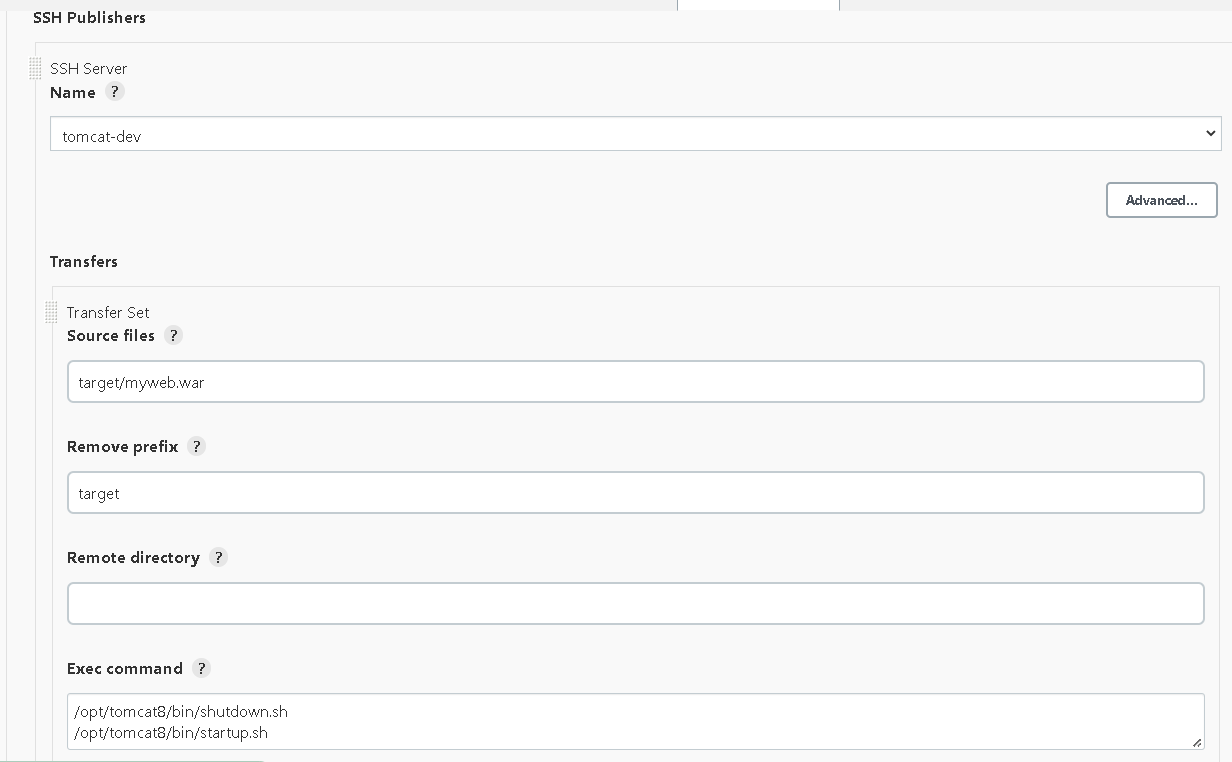
We wanna install a plugin for this on jenkins (Publish Over SSH)

Configure the plugin

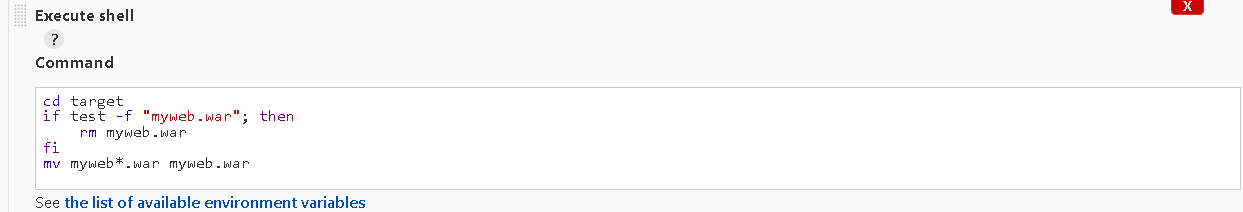
Jenkins Home → manage jenkins → Configure System →



Go to Jenkins job and use this plugin



# Execute shell:



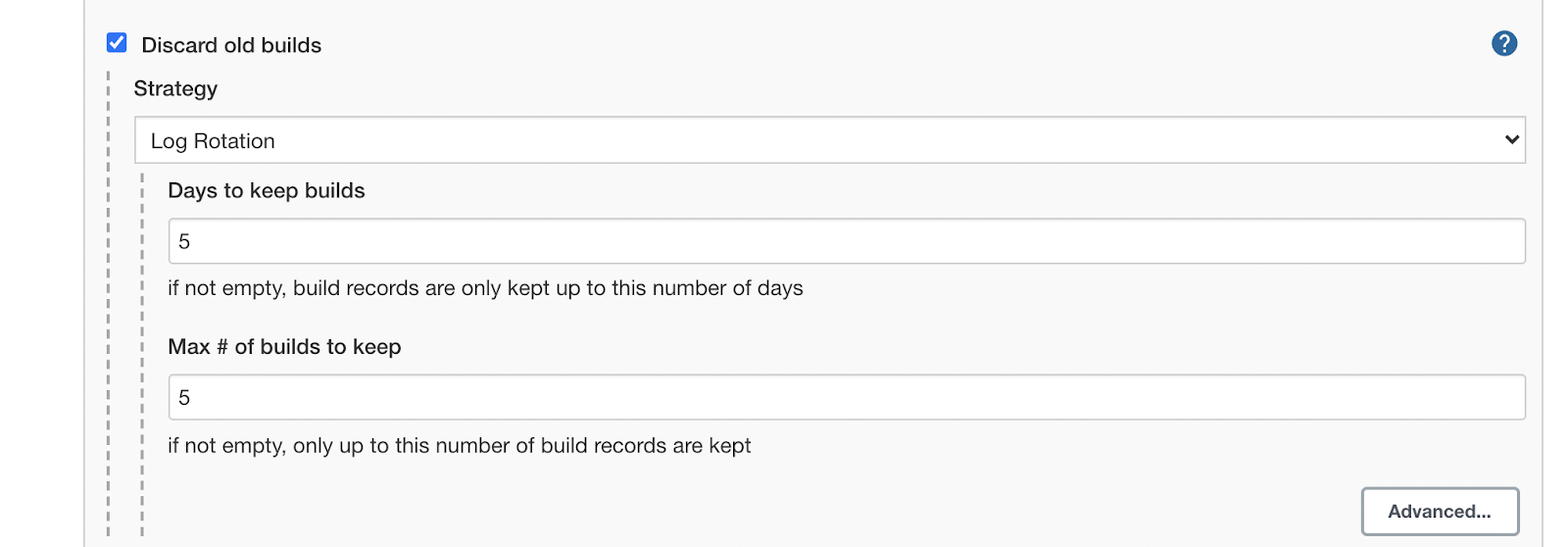
Note: For web applications we build war files, for standalone applications we build jar files.

After spring boot was introduced, we can deploy web applications as a jar, we call it a fat jar, this jar contains another jar and tomcat.

## Remove unwanted builds

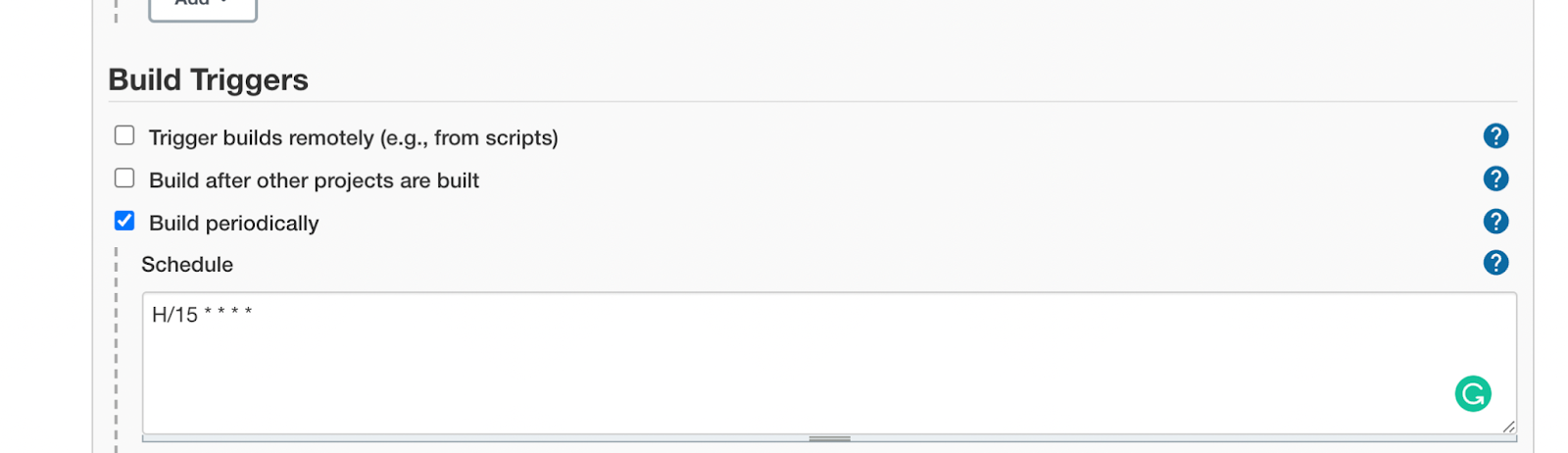
Each build occupies a space on the disk, if we do not clean it over time we run out of storage, and jenkins stops working. We need some mechanism to clean unwanted builds.

Do this under job configuration level



## Build Triggers

1. Build Periodically
   1. This feature triggers Jenkins job as per schedule.

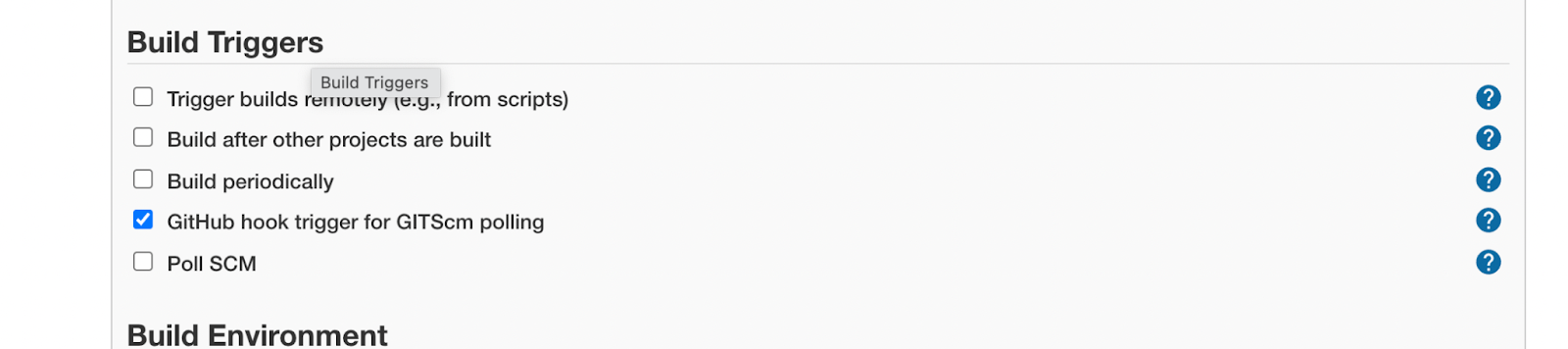


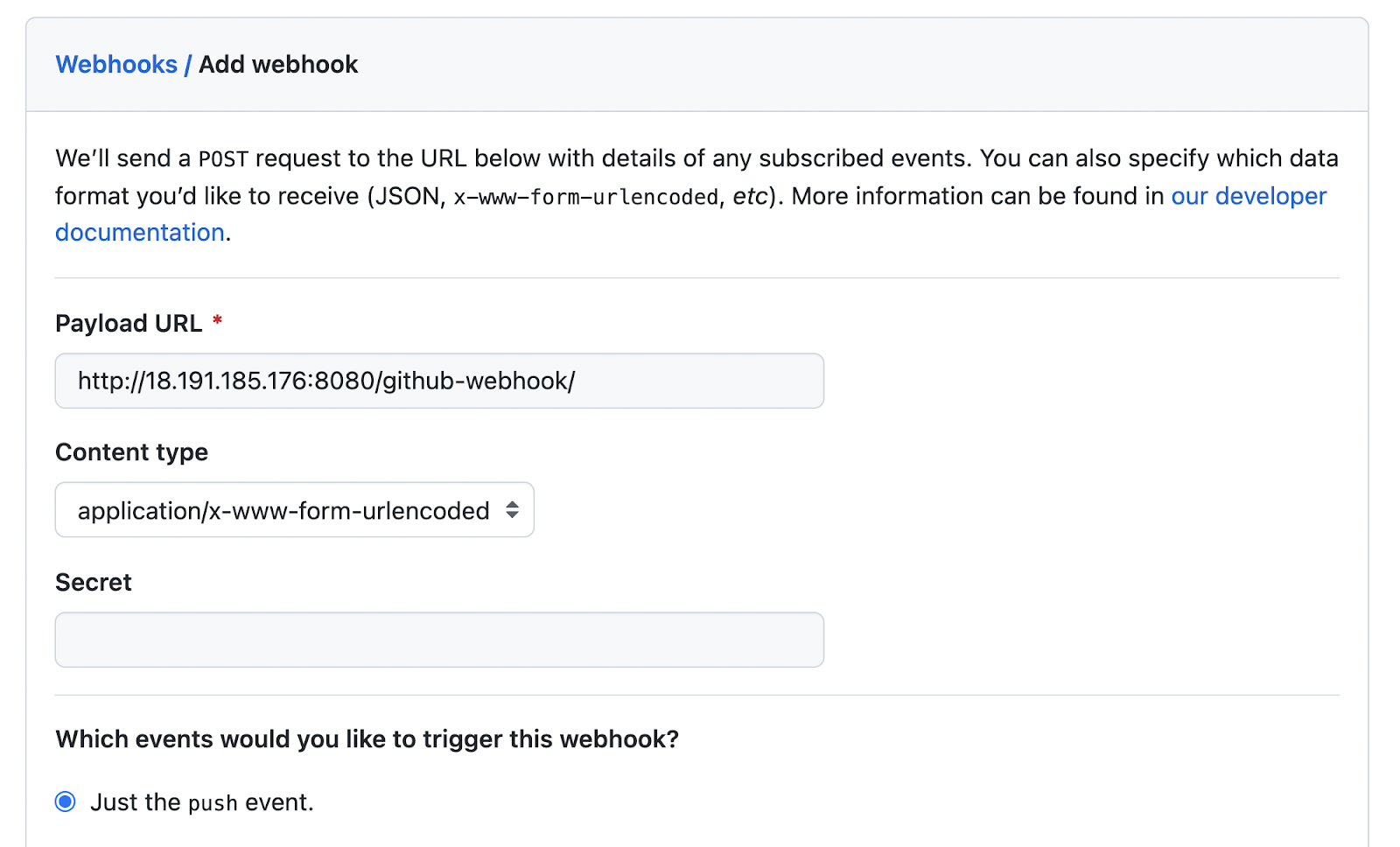
* 1. It triggers a job even if there are no changes in the code

1. Poll SCM (faq)
   1. This is similar to build periodically, but this has a condition that is trigger the build if and only if there are new commits.



1. Webhook Github (FAQ)
   1. Is a feature in Github with sending API requests to external services like Jenkins on events like push.
2. Configure Webhook Demo
   1. Create jenkins Job
   2. Configure webhook in github
   3. Configure Jenkins job with Build triggers

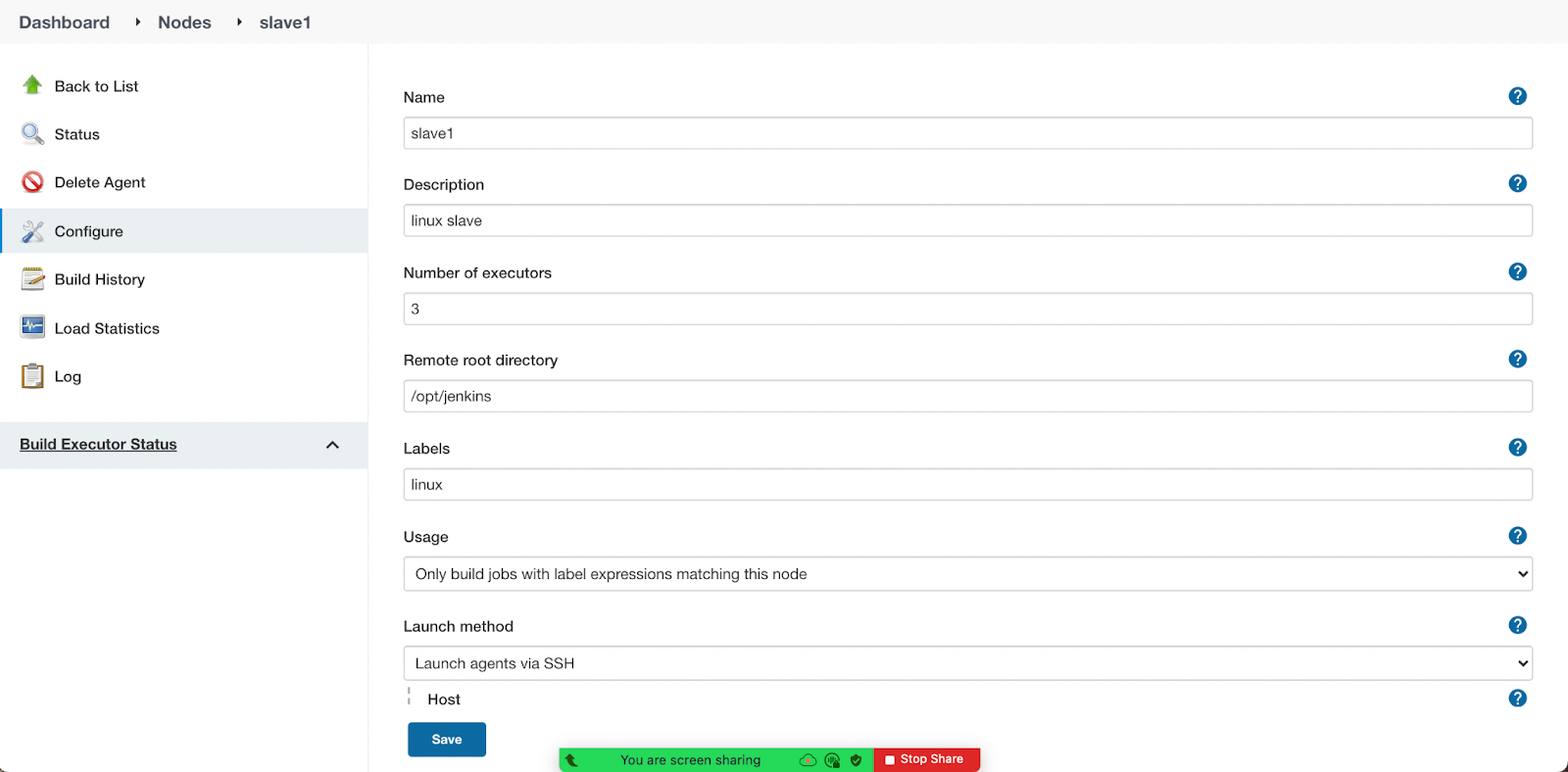




## Jenkins master slave configuration (faq)

* Using master slave configuration we scale jenkins server.
* In master slave, slaves can be
  + Virtual machines(Linux, Windows, Ubuntu, etc.)
  + Docker containers
  + Pod in kubernetes
  + Integrates with cloud
  + We are use AWS ECS for jenkins master and slaves (my project setup)

## Configure Linux slave



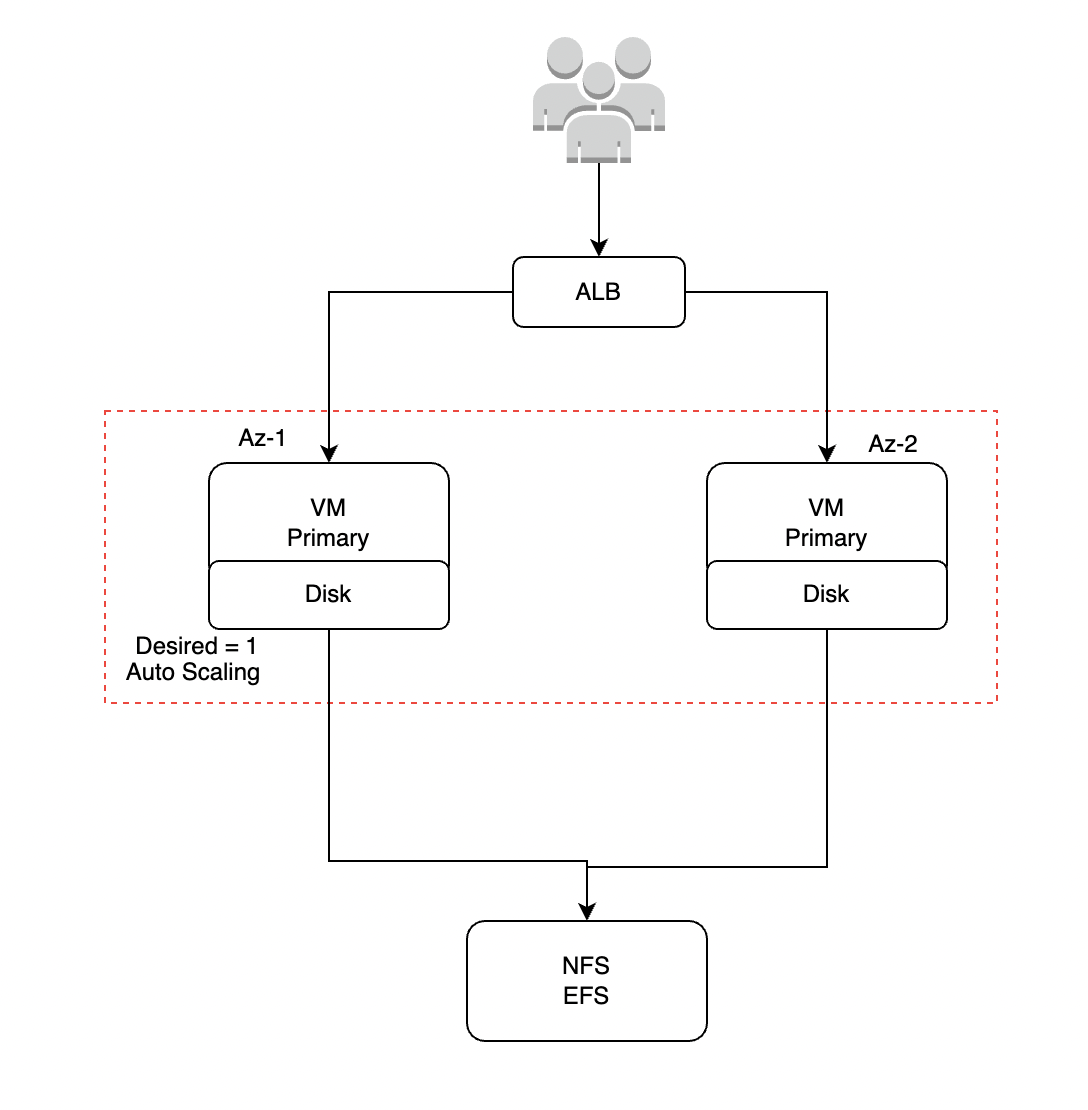
1. Get linux virtual machine
2. Install java on slave
   1. sudo yum install -y java-1.8.0-openjdk-devel
3. Create root folder on salve
   1. sudo mkdir /opt/jenkins
   2. sudo chown -R ec2-user.ec2-user jenkins/

## Schedule Job on the slave (faq)

## Exercise

* Install EC2 slave in jenkins
* <https://plugins.jenkins.io/ec2/>
  + The slave should 2 CPU, 2 GB memory
* How do you monitor jenkins jobs?

## Setting Up Jenkins with HA



## Jenkins Security

1. Jenkins stores usernames and passwords in their own disk
   1. My project, jenkins is integrated with Microsoft AD (Active Directory)
2. Logged in users are admins by default.
   1. This is the default option, and mostly this is not used.
   2. Teams use matrix based security or project based matrix authorization strategy.(faq)

## Matrix based security (faq)

Using matrix based security we are grantinging granular permissions to all users

## Project Based Matrix Authorization (faq)

Using project based matrix security

1. We can grantinging granular permissions to all users
2. We can set permissions at job level

## Jenkins folder structure on linux server

1. /var/lib/jenkins/ (jenkins installation directory)
2. users
   1. Container jenkins user details
3. Workspace
   1. Contains the code pulled from git for this job
4. Jobs
   1. Contains job configuration files, build logs, etc.
5. Plugins
   1. Contains installed plugins.
6. Tools
   1. The tools installed through jenkins

## (FAQ) How do you install plugins if the internet is not present on jenkins server?

Ans) download plugins on the machine having internet and copy to plugins folder on jenkins.

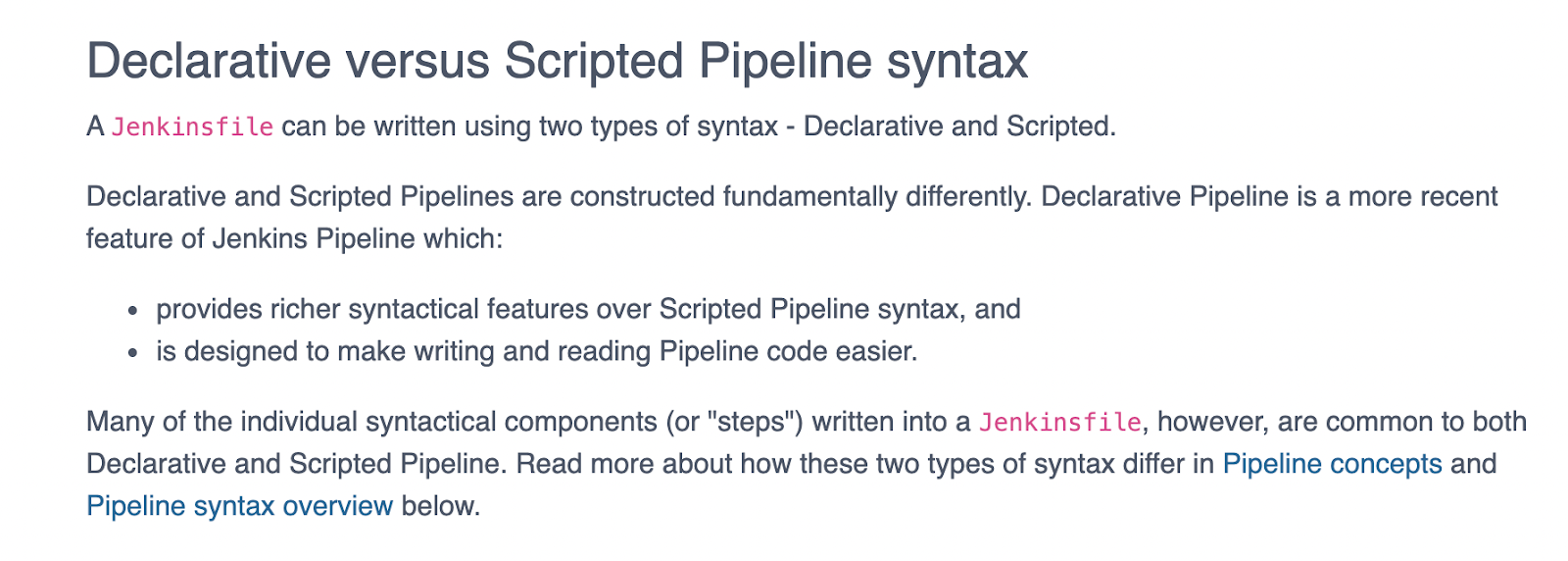
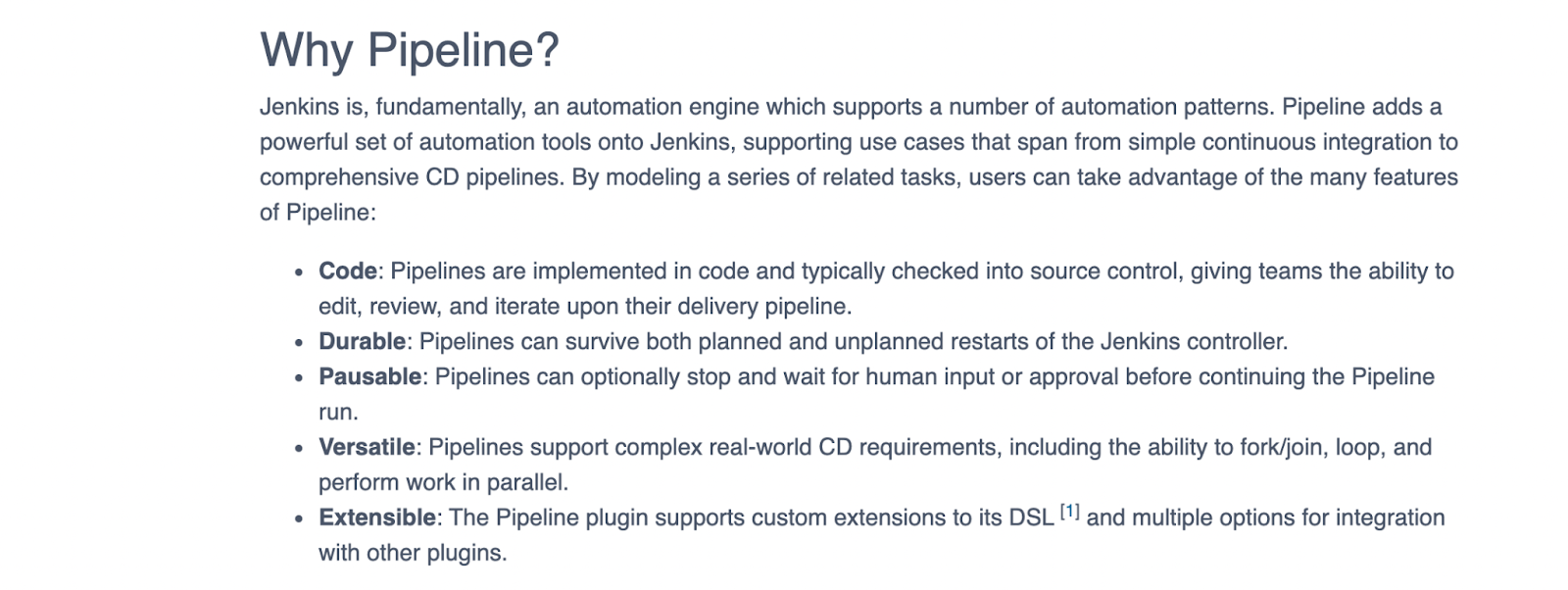
run command in linux machine:

* scp -i ~/Downloads/(pluginname) ec2-user@172.45.1.252:/var/lib/jenkins/plugins/

## Jenkins Pipelines

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

1. Today, most of the teams use Jenkins pipeline, usage of freestyle jobs are diminishing.
2. To work with pipelines you need a bit of groovy knowledge, this can be learnt while doing pipeline examples.
3. Pipelines have gained lots of popularity, and it is widely used today.



## Writing first pipeline

Write pipeline that prints welcome to pipelines

pipeline{

agent any

stages{

stage("Welcome"){

steps{

echo "Welcome to Pipelines"

}

}

}

}

## Write pipeline with two stages

1. Checkout code from git
2. Build using maven

pipeline{

agent any

stages{

stage("Git Checkout"){

steps{

git branch: 'dev', credentialsId: 'javahometech', url: 'https://github.com/javahometech/my-app'

}

}

stage("Maven Build"){

steps{

sh 'mvn package'

}

}

}

}

pipeline → declarative pipelines start with pipeline block

agent → this is mandatory, to specify where to run the pipeline job

## Jenkins Pipeline Git,Maven, Tomcat (05-Jan-2022)

pipeline{

agent {

label 'master'

}

stages{

stage("Git Checkout"){

steps{

git(branch: 'dev', credentialsId: 'javahometech', url: 'https://github.com/javahometech/my-app')

}

}

stage("Maven Build"){

steps{

sh 'mvn package'

}

}

stage("Tomcat Deploy"){

steps{

sshagent(['tomcat-dev']) {

// copy war file to tomcat server

sh """

scp -o StrictHostKeyChecking=no target/myweb\*.war ec2-user@172.31.15.109:/opt/tomcat8/webapps/myweb.war

ssh ec2-user@172.31.15.109 /opt/tomcat8/bin/shutdown.sh

ssh ec2-user@172.31.15.109 /opt/tomcat8/bin/startup.sh

"""

}

}

}

stage("Health Check"){

steps{

sh 'curl http://172.31.15.109:8080/myweb/'

}

}

}

}

## Jenkins running tasks in parallel

Jenkins pipeline has a feature to run tasks in parallel, running tasks parallely reduces build time.

pipeline{

agent any

stages{

stage("Build & Sonar"){

parallel{

stage("Maven Build"){

steps{

echo "maven build..."

}

}

stage("Sonar Scanning"){

steps{

echo "sonar scanning..."

}

}

}

}

}

}

## Can you run each stage on a separate agent?

Yes

pipeline{

agent any

stages{

stage("Build & Sonar"){

parallel{

stage("Maven Build"){

agent{

label "master"

}

steps{

echo "maven build..."

}

}

stage("Sonar Scaning"){

agent{

label "linux"

}

steps{

echo "sonar scanning..."

}

}

}

}

}

}

## Jenkins parameterized Jobs

pipeline{

agent any

parameters {

choice choices: ['master', 'dev', 'develop', 'qa'], description: 'Choose the branch to build', name: 'branchName'

string description: 'Choose the git repo to build', name: 'gitUrl'

}

stages{

stage("Git Checkout"){

steps{

// echo params.gitUrl

// echo params.branchName

git url: params.gitUrl, branch: params.branchName

}

}

}

}

## Using conditions in pipeline

pipeline{

agent any

parameters {

choice choices: ['master', 'develop', 'qa'], description: 'Choose the branch to build', name: 'branchName'

string description: 'Choose the git repo to build', name: 'gitUrl'

}

stages{

stage("Git Checkout"){

when{

expression { params.branchName == "develop"}

}

steps{

git url: params.gitUrl, branch: params.branchName

}

}

}

}

Jenkins Conditions → <https://www.jenkins.io/doc/book/pipeline/syntax/#when>

## (FAQ)What is Jenkinsfile?

We can write pipeline code under job configuration script block

We also can put the script in Jenkinsfile and get it from git or any source code management tool

<https://github.com/javahometech/my-app/blob/master/Jenkinsfile>

## (faq) Jenkins upstream and downstream jobs

In jenkins one job can call other jobs, for example we have deploy and selenium-test jobs

After the deploy job completes successfully, we want to call selenium-test job, here deploy is upstream job and selenium-test is downstream job.

## Jenkins Shared Libraries

Shared libraries are used everywhere today, it is highly important to know the shared library concepts.

Shared library is a groovy script which is reused across pipelines. In my project we use lots of shared libraries.

## Demo Working with shared library

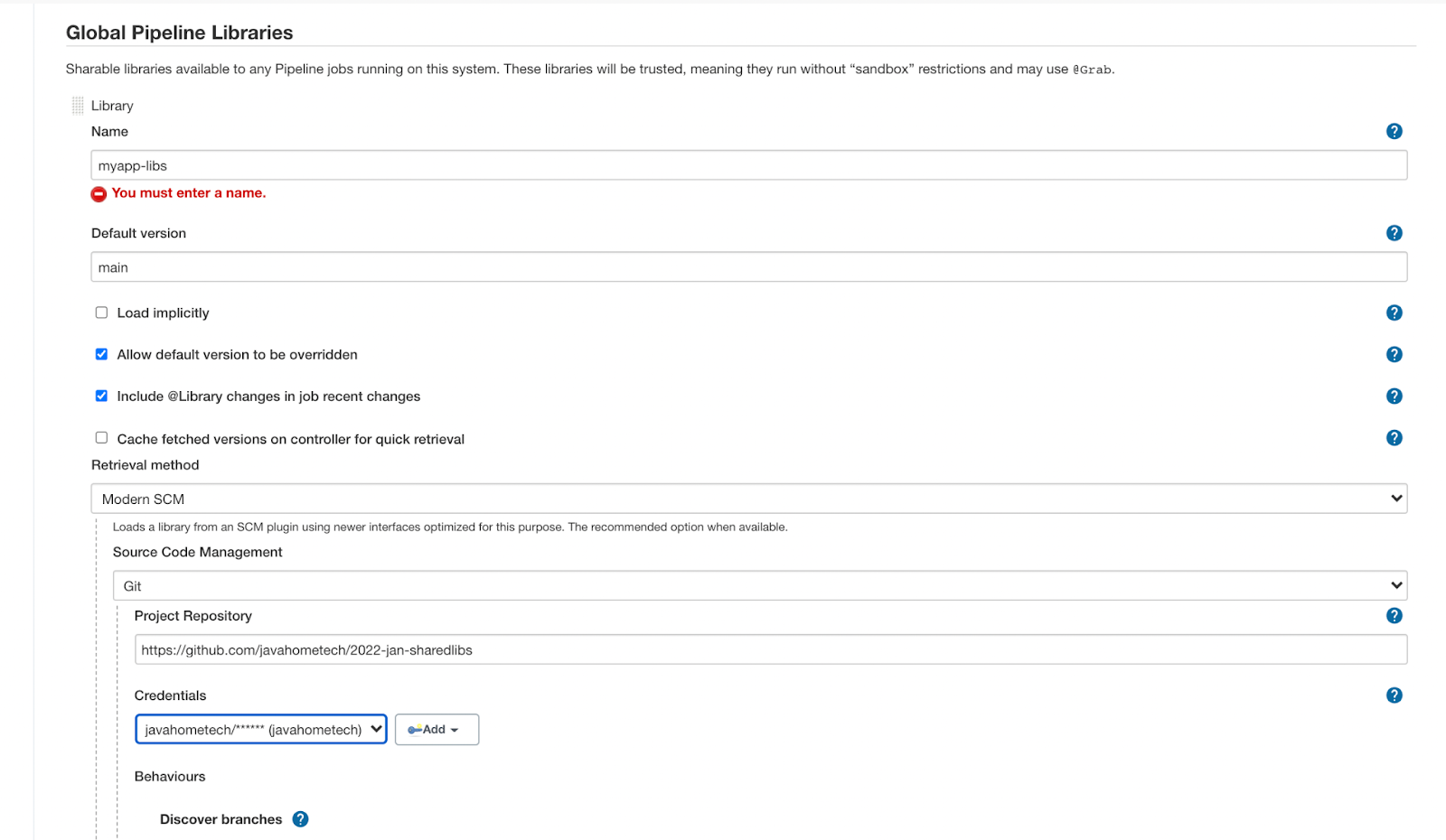
1. Create a project in git or any source code repository.
2. Inside the project create hello.groovy under vars folder.
3. Place following code snippet in the groovy file

def call(name){

echo "Hi name, How are you?"

}

1. Go to jenkins and configure this shared library



1. Using shared library in a pipeline

(it’s my edit) For shared library code in job:

@Library('myapp') \_

pipeline{

agent any

stages{

stage("welcome"){

steps{

hello("irfan")

hello("nagarjuna")

}

}

}

}

## Shared Library Tomcat Deploy

https://github.com/javahometech/2022-jan-sharedlibs/blob/main/vars/tomcatDeploy.groovy

@Library('myapp-libs') \_

pipeline{

agent {

label 'master'

}

stages{

stage("Git Checkout"){

steps{

git(branch: 'dev', credentialsId: 'javahometech', url: 'https://github.com/javahometech/my-app')

}

}

stage("Maven Build"){

steps{

sh 'mvn package'

}

}

stage("Tomcat Deploy Dev"){

when{

expression { params.env == "dev" }

}

steps{

tomcatDeploy(params.creds,'172.31.15.109','ec2-user')

}

}

stage("Tomcat Deploy UAT"){

when{

expression { params.env == "uat" }

}

steps{

tomcatDeploy(params.creds,'172.31.15.120','ec2-user')

}

}

stage("Tomcat Deploy Prod"){

when{

expression { params.env == "prod" }

}

steps{

tomcatDeploy(params.creds,'172.31.15.159','ec2-user')

}

}

}

}

# Youtube Videos

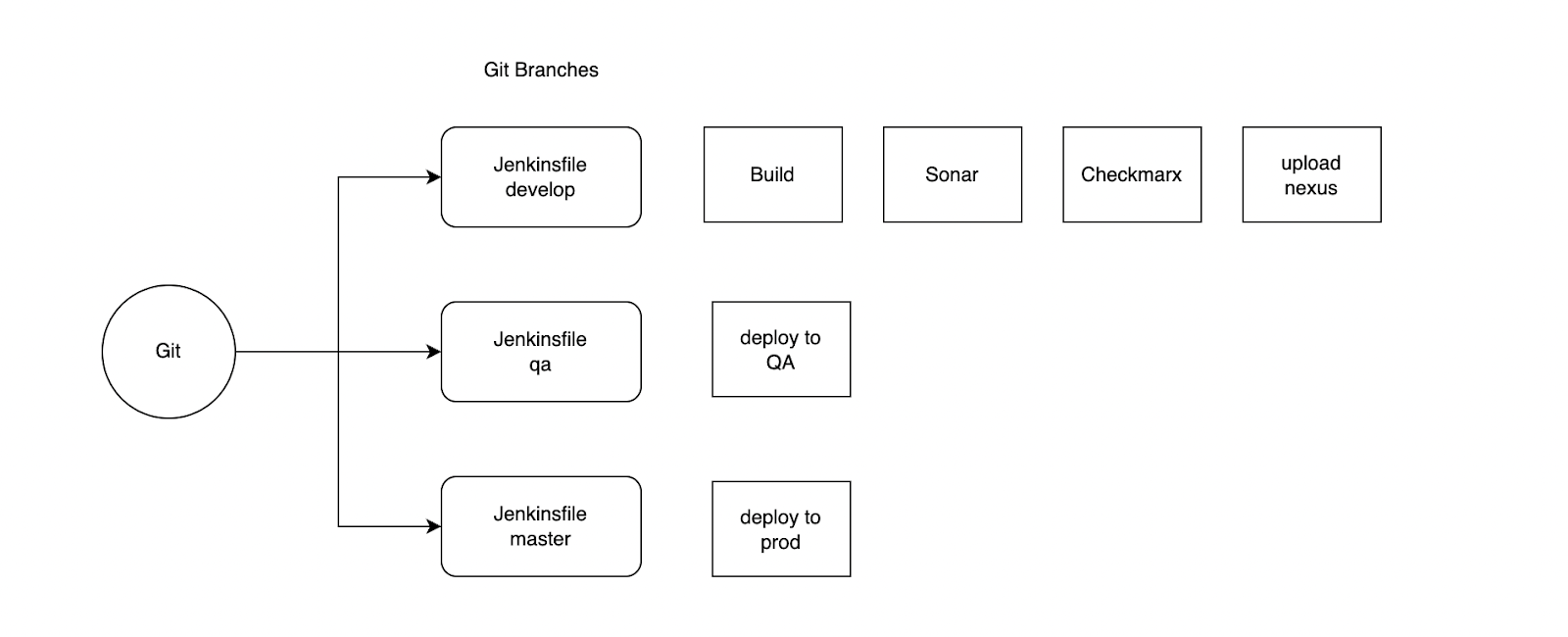
Declarative → <https://youtube.com/playlist?list=PLH1ul2iNXl7uHUaB1iaXf_lHb7tQDh7Tt>

Scripted → <https://youtube.com/playlist?list=PLH1ul2iNXl7txKuhhDMKenYOThDww6x8S>

## Multi Branch Pipeline

<https://github.com/javahometech/myapp-2022>

Multi branch pipeline scans all the branches in the repository, then it executes Jenkinsfile in the branch.



## (FAQ)Can you name a few shared libraries you worked on in your project?

1. Checkmarx scanning
2. Sonar scanning
3. Uploading images to nexus
4. Qualys scaling for docker images
5. Auto tagging

## SonarQube Analysis

Sonar is static code analysis tool, which is useful for performing following check on the code

* Automatic Reviews
* Code smells
* Detect bugs
* Security vulnerabilities
* Etc.

Note: Static code analysis means, it performs the checks directly on the source code.

SonarQube supports 20+ languages, it is available as an open source and licensed.

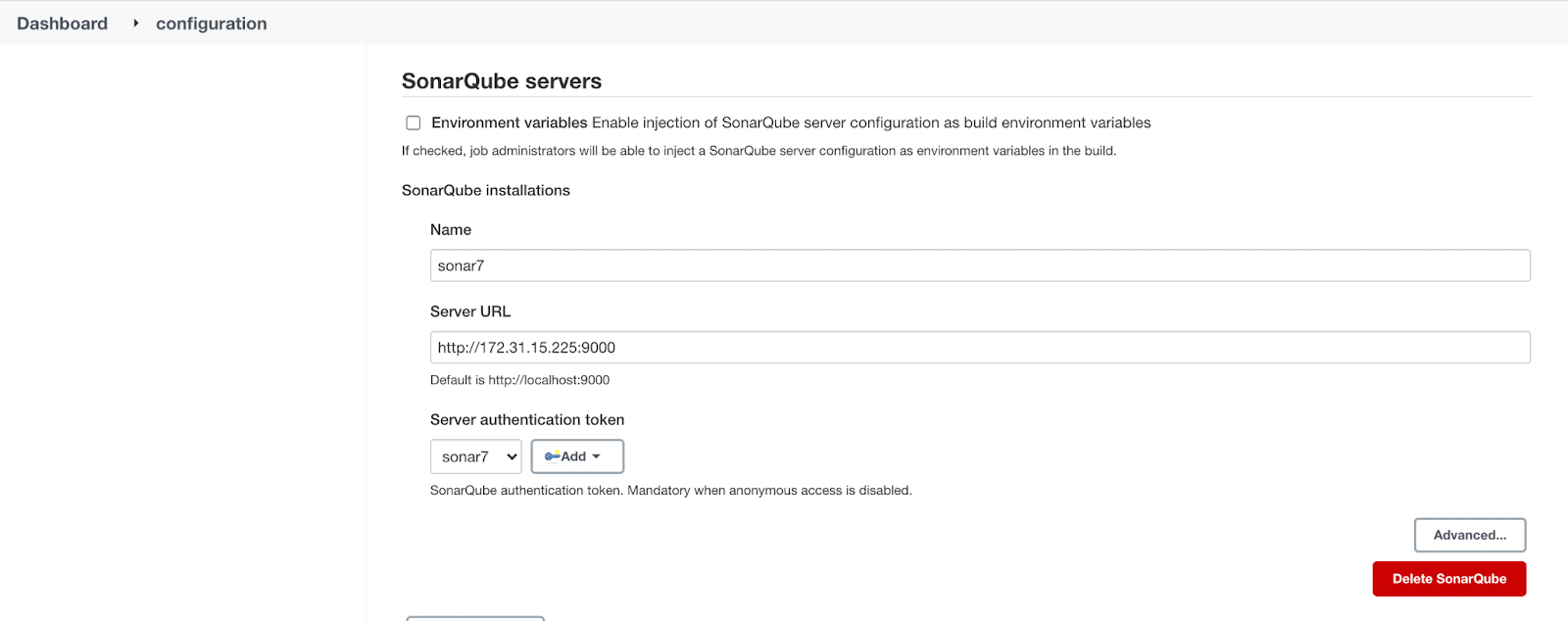
## Install and setup SonarQube on linux server

<https://docs.sonarqube.org/latest/>

There is a system requirement for sonarqube, we should have linux server with minimum 2 GB memory and 2 CPU.

1. Install java 8
   1. sudo yum install java-1.8.0-openjdk -y
2. Download sonar binary
   1. cd /opt/
   2. sudo wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.1.zip>
   3. sudo unzip sonarqube-7.1.zip
   4. sudo rm sonarqube-7.1.zip
   5. sudo mv sonarqube-7.1/ sonar7
3. Running sonarqube as a service
   1. <https://docs.sonarqube.org/7.1/RunningSonarQubeasaServiceonLinux.html>
   2. Change run as user
      1. sudo vi /opt/$SONAR\_HOME/bin/linux-x86-64/sonar.sh
      2. RUN\_AS\_USER=”ec2-user”
   3. Change the user on sonar
      1. sudo chown -R ec2-user.ec2-user sonar7/
4. Access sonar from web browser
   1. Sonar runs on 9000 port
   2. Login with admin/admin (this is default username and password)
5. Our sonar is integrated with microsoft active directory

## Configure sonarqube in jenkins



## Integrating sonarqube in pipeline

<https://github.com/javahometech/myapp-2022/blob/develop/Jenkinsfile>

## Sonar Quality Gate

Quality Gate is the set of conditions the project must meet before it can be released into production.

Every project can have its own quality gate.

## Configure SonarQube Scanner

https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-jenkins/

## Checkmarx

<https://checkmarx.atlassian.net/wiki/spaces/SAST/overview>

* In my team, checkmarx is not managed by devops, we integrate jenkins with it.

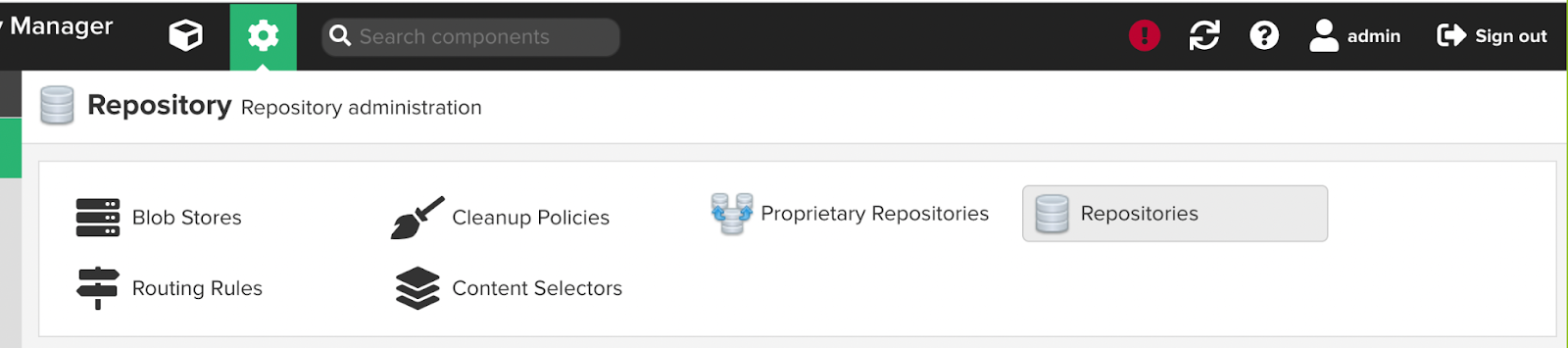
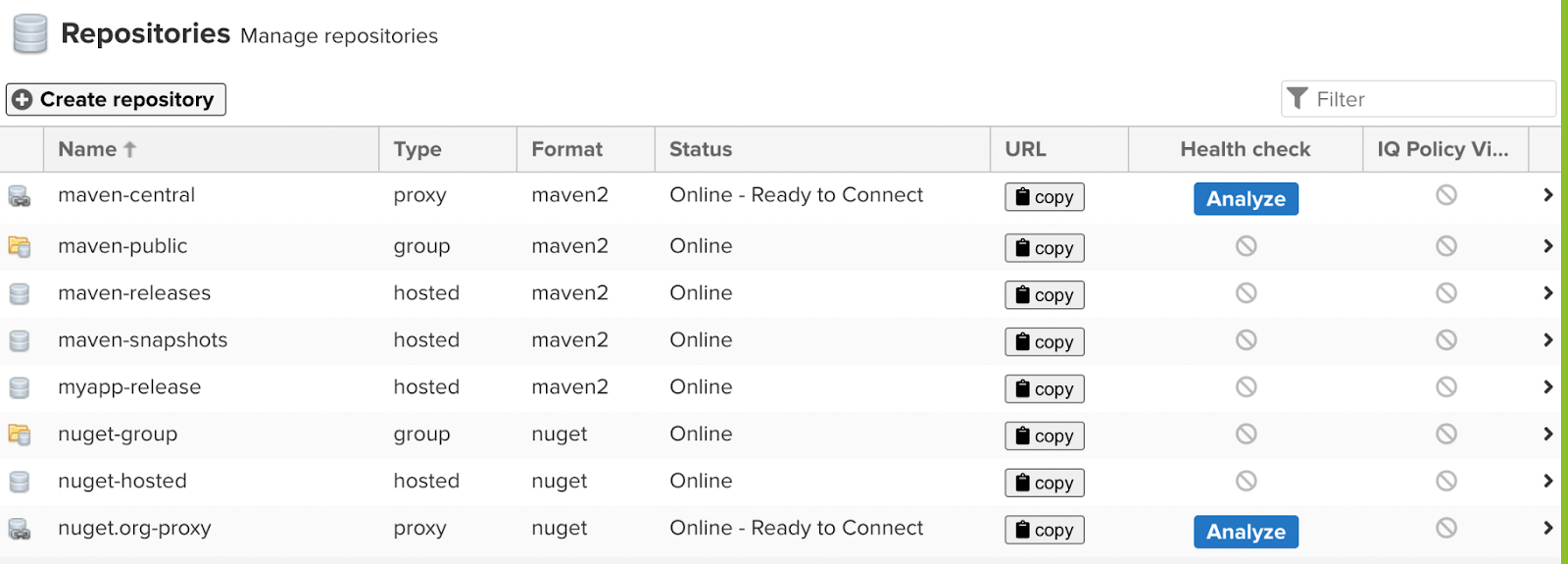
## Jenkins Email Notifications

## Sonatype Nexus

Install and configure nexus3 on linux

1. ssh into linux server
2. sudo yum install java-1.8.0-openjdk -y
3. cd /opt
4. sudo wget <https://download.sonatype.com/nexus/3/latest-unix.tar.gz>
5. sudo tar xf latest-unix.tar.gz
6. sudo rm latest-unix.tar.gz
7. sudo mv nexus-3.37.3-02/ nexus3
8. sudo chown -R ec2-user.ec2-user nexus3/ sonatype-work/
9. Configure nexus as a service
   1. https://help.sonatype.com/repomanager3/installation-and-upgrades/run-as-a-service

## Nexus Uploading Maven War files to Nexus

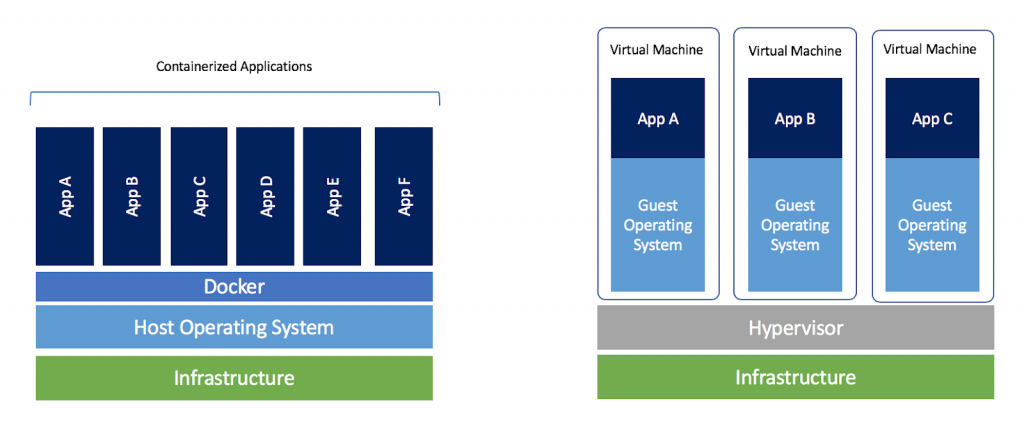
1. Install jenkins plugin Nexus Artifact Uploader
2. Create maven repository in nexus
   1. 
   2. Create repository
      1. 
   3. Select maven2(hosted)
   4. Give a name and create

# Docker Containerization

Docker is a containerization tool, docker isolates the application, its dependencies and runtime.

* Docker is lightweight, it consumes less memory, CPU, etc.
* Docker is portable, i.e. It consistently runs anywhere.
* Docker container starts in seconds, where as vms take minutes
* Docker minimizes number of VMs to use, if there are licenses associated with VMs it is also reduced.
* It's easy to package, ship and run.

**Note:** *In my current project, gitlab, sonar, jfrog, jenkins are running on docker*.



## Install docker on linux

1. sudo yum install docker -y
2. sudo service docker start
3. sudo chkconfig docker on
4. sudo usermod -a -G docker ec2-user
5. Exit from the session and login back

## Demo Run Jenkins on docker

1. docker run -d -p 8080:8080 jenkins/jenkins

## (FAQ)How do you troubleshoot if an application in a docker container is failing?

docker logs 0ae64a60b01b

(FAQ)What is the command to get into the container?

docker exec -it 0ae64a60b01b bash

## (FAQ) Docker Volumes

Docker volume helps us to persist container data on the host. So we do not lose data when the container is terminated.

## Demo persisting data using volumes

1. docker volume create myvol
2. To inspect volume details
   * 1. docker volume inspect myvol
   1. Run docker container using a volume
      1. docker run -d -p 8080:8080 -p 50000:50000 -v myvol:/var/jenkins\_home jenkins/jenkins

## Docker Bind Mount

docker run -d /home/ec2-user/app:/var/app alpine

## What is a stateful application?

* Any application which stores its state(data) locally at runtime is called a stateful application.
* Examples for stateful applications are database, jenkins, nexus, sonarqube.

## Docker Networking

In this session we will discuss how containers communicate with each other on the same host.

1. Container run using their own network, which is different from the host
2. By default docker runs on bridge network
3. Default bridge network does not support DNS

Demo Network

1. docker run -itd –name green alpine
2. docker run -itd –name red alpine
3. docker network inspect bridge
4. From above command grab ip of both containers
5. Get onto green container and ping red container on its ip
   1. docker exec -it green ash
   2. ping <ip-of-red>

## Custom Bridge Network

1. Custom bridge is pretty much the same as default bridge but it has DNS support, that is container names are resolved into it ips.
2. Create custom bridge network
   1. docker network create --driver=bridge javahome
   2. docker run -itd --name red --network javahome alpine
   3. docker run -itd --name green --network javahome alpine

## Docker Compose

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application’s. Then, with a single command, you create and start all the services from your configuration.

## Installing Docker Compose

<https://docs.docker.com/compose/install/>

Write docker-compose file

<https://github.com/javahometech/docker-compose/blob/master/docker-compose.yml>

Running docker containers using compose

wget <https://raw.githubusercontent.com/javahometech/docker-compose/master/docker-compose.yml>

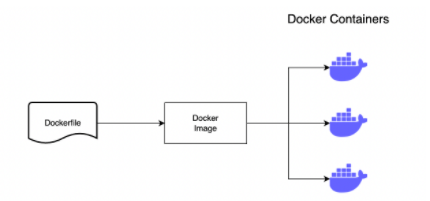
docker-compose up -d

docker-compose down

Real Time scenario of docker-composer

* We are setting up gitlab on EC2 using docker-compose, this is for development and production.

## How to build docker images



* To build docker image we need to write Dockerfile
* To write Dockerfile we need to know instructions

## Dockerfile Instructions

FROM ubuntu:16.04

LABEL APP=UI

LABEL AUTHOR="Hari Kammana"

RUN apt update

RUN apt install apache2 -y

WORKDIR /var/www/html

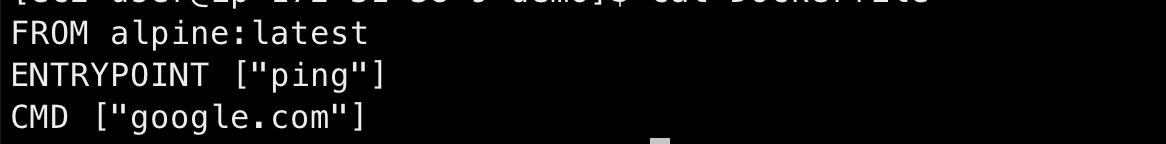
COPY index.html .

EXPOSE 80

CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]

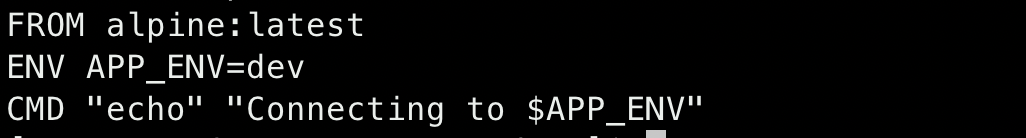
## (FAQ)Whats difference between CMD and ENTRYPOINT

1. Both are run time instructions, it is used to start the process in a container at runtime.
2. CMD instructions can be overridden and ENTRYPOINT can’t be overridden.
3. We can combine ENTRYPOINT and CMD



## Dockerfile ENV

Environment variables are used as arguments to the applications, for example there is an application which connects to an oracle database. Database details like URL, port, username, password are different for different environments.



docker run -it -e APP\_ENV=prod demo:v1

## (FAQ)What is difference between ADD and COPY

COPY supports only local source, ADD supports local and remote source, the source can be remote URL

## Docker Videos

[DevOps CI/CD Jenkins Pipeline, Ansible & Docker | For DevOps Online training call +919886611117](https://youtu.be/13FpCxCClLY)

[Docker | Docker basic commands | Docker tutorial | Docker Commands](https://youtu.be/h8n0dhrJuqg)

[DevOps | Setting up docker private registry | Install Nexus | Java Home](https://youtu.be/arAXOnvOSyY)

[Docker Networking | Docker bridge network deep dive | Container bridge drive](https://youtu.be/Tx12haz-4VA)

[What is Docker Volume?| How to use docker volume? | Docker tutorial for beginners](https://youtu.be/BefiUTwJobg)

# Kubernetes (Container Orchestration)

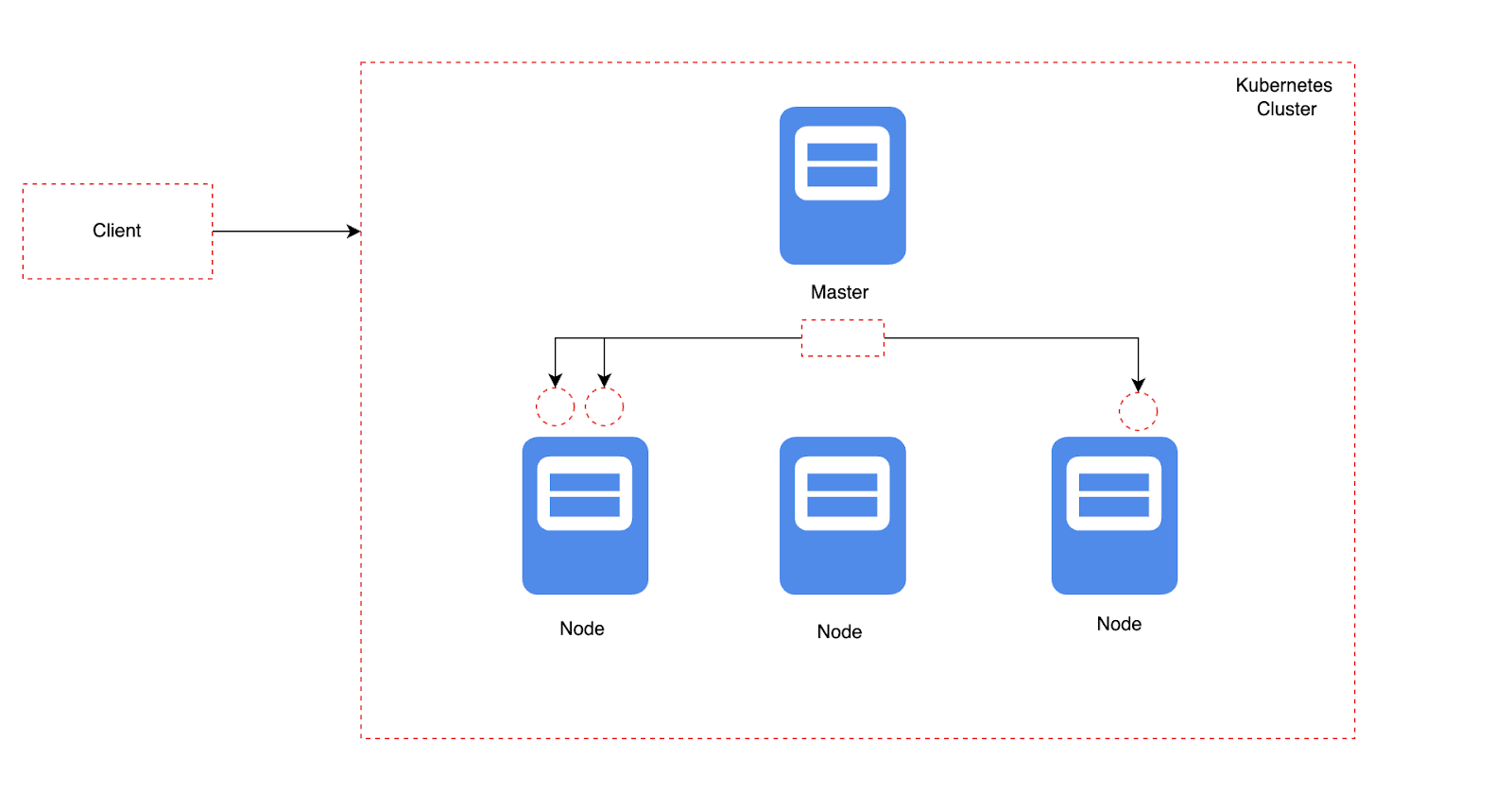
[Kubernetes](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/), also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

In production environments we gonna have lots of containers for lots of microservices, we need a proven framework or orchestration tool to maintain them.

### Kubernetes features

1. Scheduling, kubernetes will figure out the right node to run our applications.
2. If any of the containers crashes in the kubernetes cluster, it needs to be automatically created.
3. Kubernetes can create multiple replicas
4. Load Balancing microservices
5. Automatic deployments, like rolling updates, with simple commands we can undo the rollout.
6. Autoscaling for containers
7. Etc.

## Highlevel Kubernetes Architecture



Note: Kubernetes can be setup anywhere, on-premises, any cloud

## Setting Up Kubernetes cluster

There are several ways to setup kubernetes cluster

1. Kubeadm
2. Kubespray
3. Setting Up On cloud
   1. AWS EKS (Elastic Kubernetes Service)
   2. AWS Kops
   3. Azure AKS
   4. Google Kubernetes Engine
4. Red Hat Openshift
5. Rancher Labs
6. Apache Mesos

## Setting up kubernetes cluster using EKS

* EKS is not free, it incurs cost
* EKS is fully managed kubernetes cluster

## Steps to setup EKS cluster

[AWS EKS Demo | AWS Elastic Kubernetes Service | Deploy Application on EKS | EKS](https://youtu.be/O4h69KMm2tE)

## Connecting to the kubernetes cluster

1. Install kubectl on your laptop

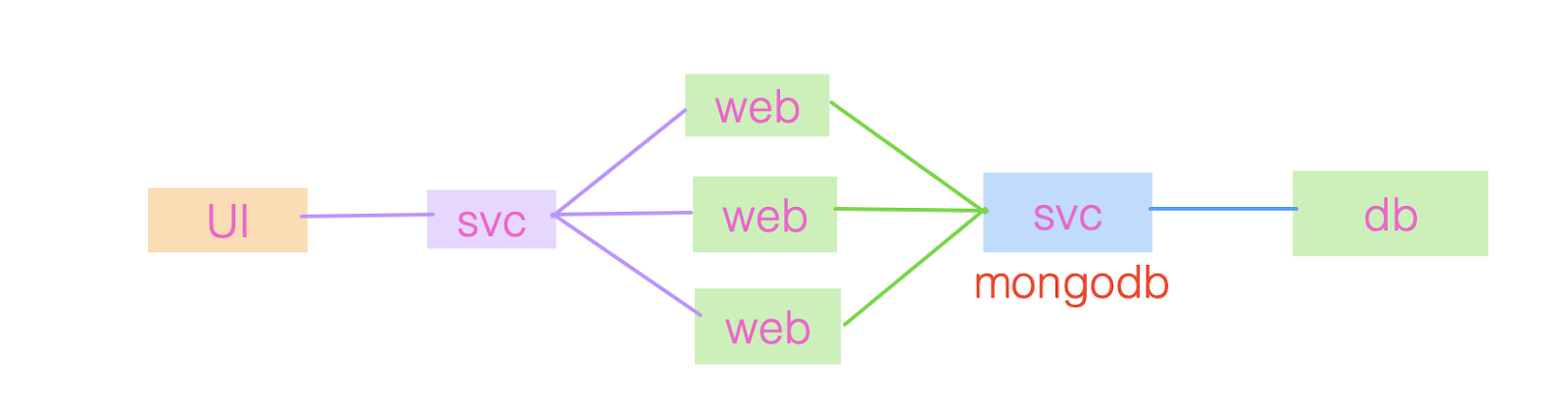
## Kubernetes Resources & Concepts

1. Install kubectl command line
   1. <https://kubernetes.io/docs/tasks/tools/>
2. Install aws cli
   1. <https://aws.amazon.com/cli/>
3. Configure aws cli with access keys and secret keys
4. aws eks update-kubeconfig --region <region-name> --name <cluster-name>
5. Test the connectivity with following command
   1. kubectl get pods

## Kubernetes Pod

* Pod is the smallest unit of deployment in kubernetes, a pod represents an instance of your application in the kubernetes cluster.
* Pod is a wrapper of one or more containers
* Each pod gets an ip address, this ip is private to the cluster.
* Containers in same pod communicates over ‘localhost’
* Containers in a pod shares same network and storage.
* Pods are designed to be mpheral(temporary), Pods can be terminated and recreated dynamically.
* kubectl get nodes
* kubectl get pods
* kubectl logs nginx
* kubectl describe pod nginx
* kubectl get nodes
* kubectl describe node ip-172-31-15-215.ap-south-1.compute.internal
* kubectl get pods
* kubectl get pods -o wide

## Kubernetes Service Resource



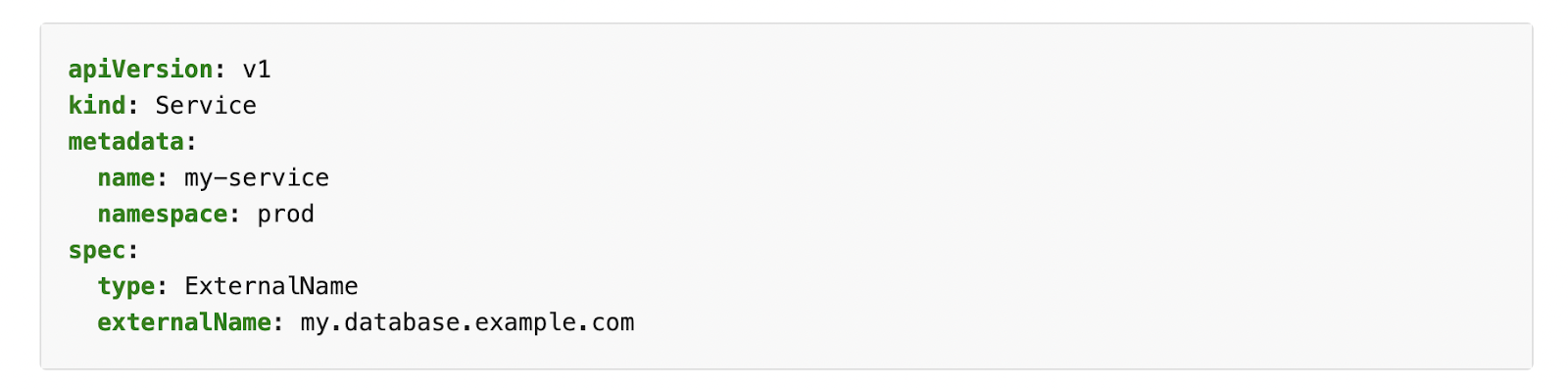
* Service gets name, IP
* Service sticks to the cluster permanently until its deleted
* Pods can communicate to service over ip or name, but best practice is using name.

<https://github.com/javahometech/kubernetes>

## (FAQ)How do you expose pods to the internet?

Ans) Create a service using NodePort or Loadbalancer

Different types of services

1. NodePort
   1. This opens a port on every node(ex: 32000) of the cluster, to access applications from the internet, Send requests to any Node on 32000.
2. ClusterIP
   1. This creates a service which is private to the cluster. This can’t be accessed from the internet.
3. LoadBalancer
   1. This is supported only if your cluster is on cloud like AWS, Azure, GCP
4. ExternalName
   1. This is used for external servers
   2. 

## (FAQ) What is headless Service in kubernetes?

## Running pod and exposing to internet using load balancer

kubectl get pods -o wide

kubectl delete po nginx

kubectl create -f https://raw.githubusercontent.com/javahometech/kubernetes/master/pods/pods.yml

kubectl get po

wget https://raw.githubusercontent.com/javahometech/kubernetes/master/pods/pods.yml

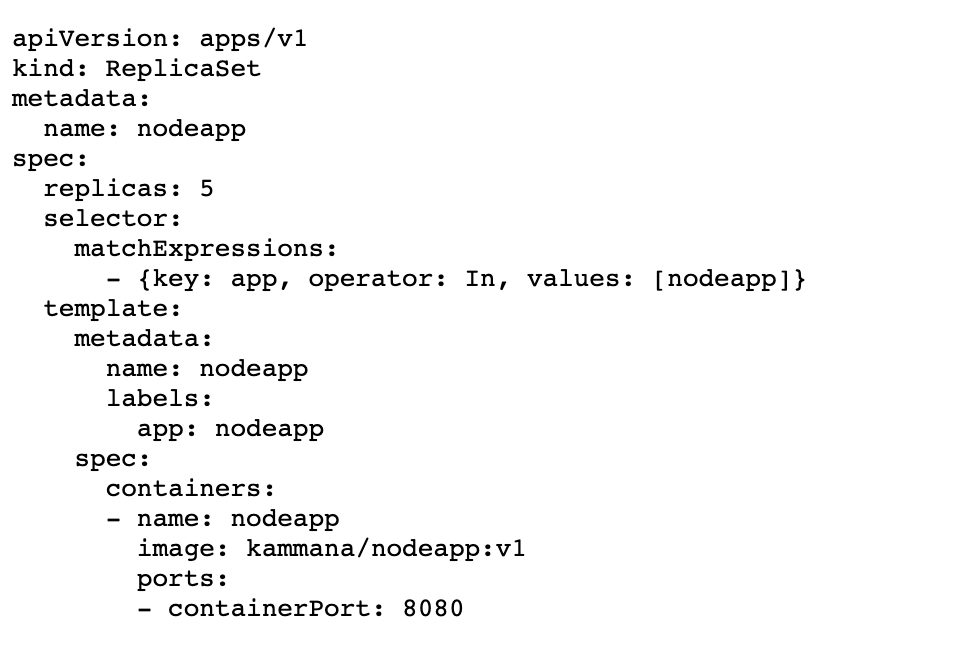
curl -o pods.yml https://raw.githubusercontent.com/javahometech/kubernetes/master/pods/pods.yml

kubectl create -f https://raw.githubusercontent.com/javahometech/kubernetes/master/services/services.yml

kubectl get svc -o wide

## Kubernetes ReplicaSet

The core objective of this resource is to create multiple replicas, maintain the desired capacity always.



## Pod Health Checks

Pod health checks are very important, it monitors the health of the pod, and sometimes autoremidiate.

<https://github.com/javahometech/kubernetes/tree/master/liveness-readyness>

We got two types of health checks

1. Readiness probe
   1. This helps in giving applications inside pods to get initialized and be ready for severe requests.
   2. If the readiness probe fails, kubernetes will not send requests until it succeeds.
   3. Pod is added to the service only after the readiness probe succeeds.
2. Liveness probe

https://github.com/javahometech/kubernetes/tree/master/liveness-readyness

* 1. For example Pod is live and servicing requests and caught with an deadlock, in such situations the application will not respond, the quick solution is restart the pod.
  2. When the liveness probe fails kubernetes will restart the pod.

## Kubernetes Deployment Object

Deployment object provides features like rolling updates and undo rollouts

<https://github.com/javahometech/kubernetes/tree/master/deployments>

## Jenkins pipeline to deploy to kubernetes

## Creating service account in jenkins for kubernetes cluster(FAQ)

https://docs.oracle.com/en-us/iaas/Content/ContEng/Tasks/contengaddingserviceaccttoken.htm

Jenkins Pipeline For Kubernetes Deployment

pipeline{

agent any

tools {

maven 'maven3'

}

stages{

stage("Git Checkout"){

steps{

git 'https://github.com/javahometech/myapp-2022'

}

}

stage("Maven Build"){

steps{

sh 'mvn clean package'

}

}

stage("Build Docker"){

steps{

sh 'docker build -t kammana/eks-app:0.0.1 .'

}

}

stage("Push to Docker Hub"){

steps{

withCredentials([string(credentialsId: 'dockrehub-pwd', variable: 'hubPwd')]) {

sh "docker login -u kammana -p ${hubPwd}"

sh "docker push kammana/eks-app:0.0.1"

}

}

}

stage("Deploy to Kubernetes"){

steps{

withCredentials([kubeconfigFile(credentialsId: 'kubeconfig', variable: 'KUBECONFIG')]) {

sh "/usr/local/bin/kubectl apply -f kubernetes/pod.yml "

}

}

}

}

}

## Kubernetes Secrets

We may have to store database passwords and API tokens to connect with other applications, we have to securely store sensitive data.

To securely store sensitive data in kubernetes we could use a secret object.

Note: For more details about secrets

<https://kubernetes.io/docs/concepts/configuration/secret/>

Note:

1. *In AWS instead of kubernetes secrets we can use AWS Secrets Manager*
2. *Kubernetes can be integrated with third party vaults*

## Kubernetes Volumes

In kubernetes Pods are ephemeral, data stored on pod/container is temporary and they do not persist. If you have a requirement to persist data generated by Pod/Container after it is terminated then we have to use Volumes.

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

### Persistent Volume

1. PV is a piece of storage on the cluster like a node
2. PV can be created statically or dynamically
   1. Static
      1. Cluster administrator created this volume
   2. Dynamic
      1. It is automatically created using storage class
3. Persistent Volume can be
   1. EBS volume
   2. AWS EFS
   3. AzureDisk
   4. azureFile
   5. gcePersistantDisk
   6. Etc

## Kubernetes ConfigMap

1. Applications deal with configuration data like, database details which is usually different for different environments like development, test, production.
2. Embedding database details in the code itself causes different problems, so we have to decouple configuration data and inject it at runtime.
3. Using ConfigMap we can decouple configuration data.

## Kubernetes Statefulset (FAQ)

Statefulset is designed for stateful applications like databases.

1. Stateful set provided sticky identifiers
   1. mydb-0
   2. mydb-1
   3. mydb-2
2. It creates pods in a specific order, mydb-0 then mydb-1 and then mydb-2
3. It deletes pods in a specific order like, mydb-2, then mydb-1 and mydb-0

## Kubernetes Daemonset

<https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>

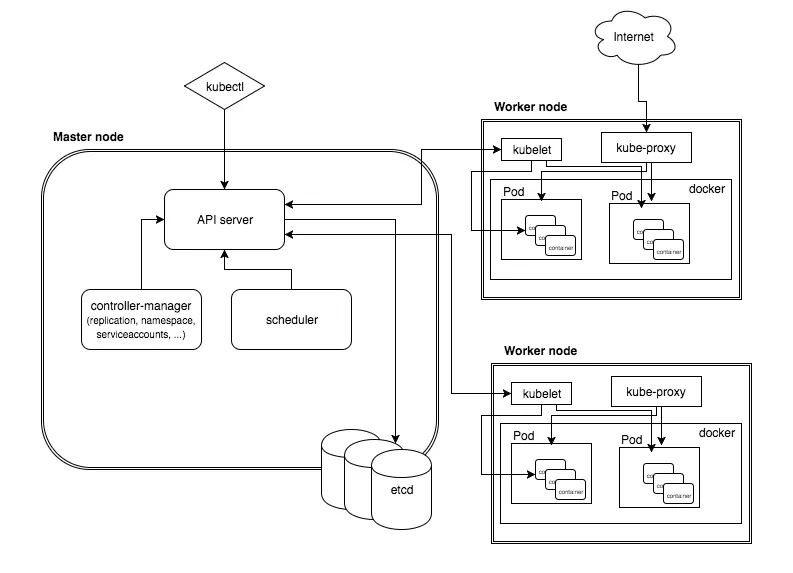
A *DaemonSet* ensures that all (or some) Nodes run a copy of a Pod.

## (FAQ) How do you schedule Pods on a specific node?

If we don’t say anything about where to schedule it is decided by default scheduler, to change default behavior we have to use

1. Node affinity
   1. This works with labels on the nodes
   2. We can used required or preferred while defining affinity
   3. <https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes-using-node-affinity/>
2. Taints, tollerations
   1. [Node affinity](https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#affinity-and-anti-affinity) is a property of [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) that attracts them to a set of [nodes](https://kubernetes.io/docs/concepts/architecture/nodes/) (either as a preference or a hard requirement). Taints are the opposite -- they allow a node to repel a set of pods.
   2. <https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/>

## (FAQ) Explain kubernetes Architecture



In kubernetes architecture we see two types of Nodes

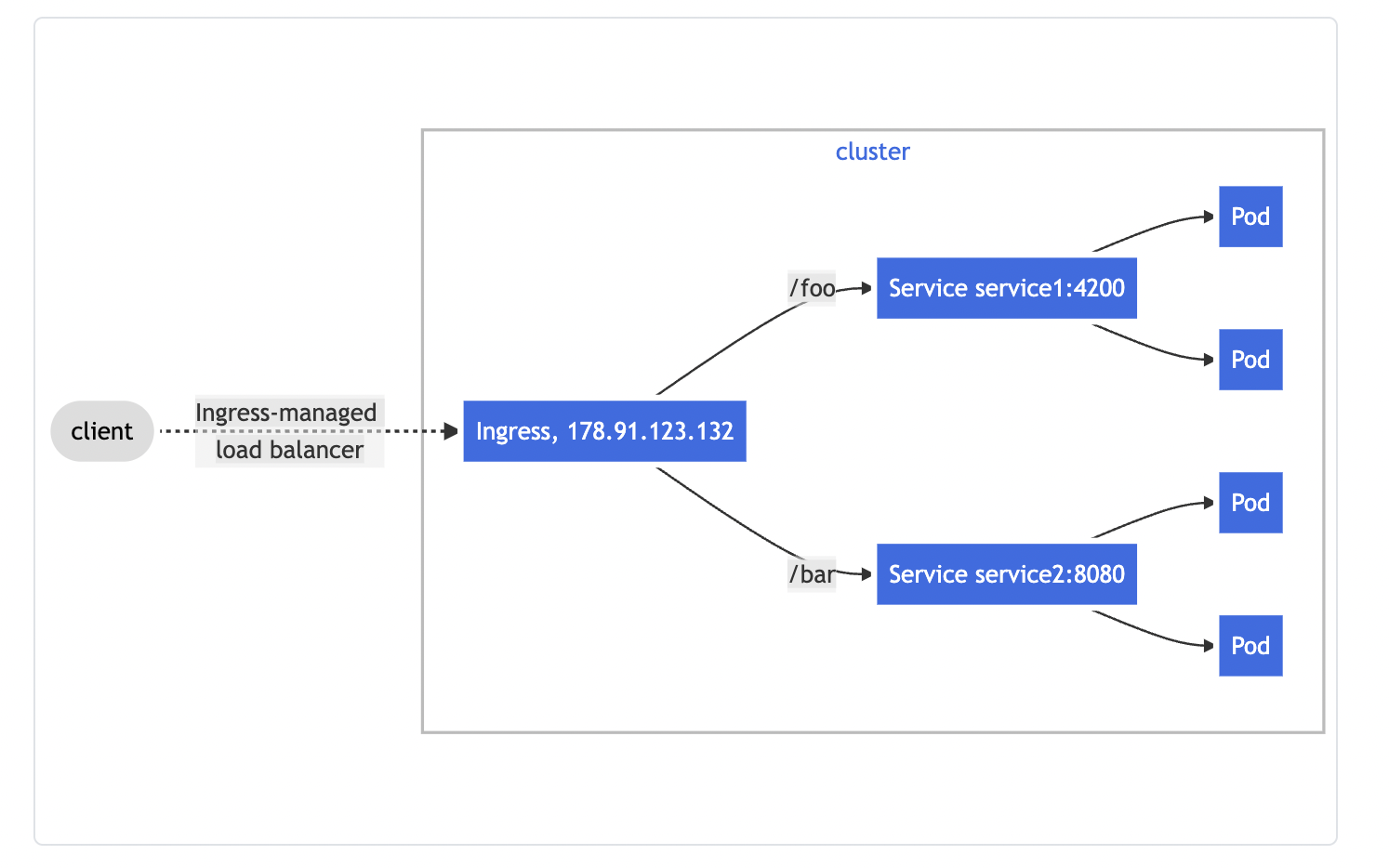
1. Master
   1. Api server
      1. Api server is the gateway for the kubernetes cluster.
      2. Authentication authorization all happens at API server
   2. Scheduler
      1. The job of scheduler is to figure out the node in the cluster for running the Pod
   3. Etcd
      1. It is a distributed key value store which contains the state of the cluster.
   4. Controllers
      1. There are different types of controllers
      2. Things like node controller, replica set, replication controllers are the examples
2. Node
   1. Kubelet
      1. It is an agent which runs on every node, its function is to listen to the master and periodically send node updates to the master.
      2. This agent is responsible for creating pods.
   2. Kube-proxy
      1. The job of kube-proxy is to manage kubernetes networking stuff.

## Kubernetes Autoscaling

<https://aws.amazon.com/premiumsupport/knowledge-center/eks-metrics-server-pod-autoscaler/>

1. Autoscaling of nodes
   1. In cloud like AWS nodes can use built-in autoscaling for nodes
2. Autoscaling of pods
   1. Kubernetes offers Horizontal Pod Autoscaler (HPA) for autoscaling pods.
   2. metric-server needs to be installed on the cluster for hpa to work

## (FAQ) Kubernetes Ingress Controller



Ingress is a single point of contact for kubernetes clusters, and it has routing rules which routes traffic to different services based on path and host.

## Helm Charts

Helm is a packing framework for kubernetes, our yaml documents are static but in real time yaml files should be dynamic, to get dynamic nature to our kubernetes YAML files helm will help.

# Ansible Configuration Management Tools

Ansible is a widely used automation tool, used for various use cases.

1. Installing and configuring servers
2. Automating deployments
3. Automating cloud infrastructure
4. Automating security and networking

* No special scripting knowledge is required, all automation is done through YAML
* Ansible is simple to learn and use.
* Ansible syntaxes resembles linux
* Ansible is agent less (IQ)
  + Chef, Puppet are agent based configuration tools
* Ansible connection methods (IQ)
  + It uses SSH for linux
  + It uses winrm for windows

## Install Ansible on Amazon Linux

* sudo amazon-linux-extras install ansible2
* The server where ansible is installed is called *control node*
* Ansible can be installed only on non-windows.

## I want to configure a server using ansible

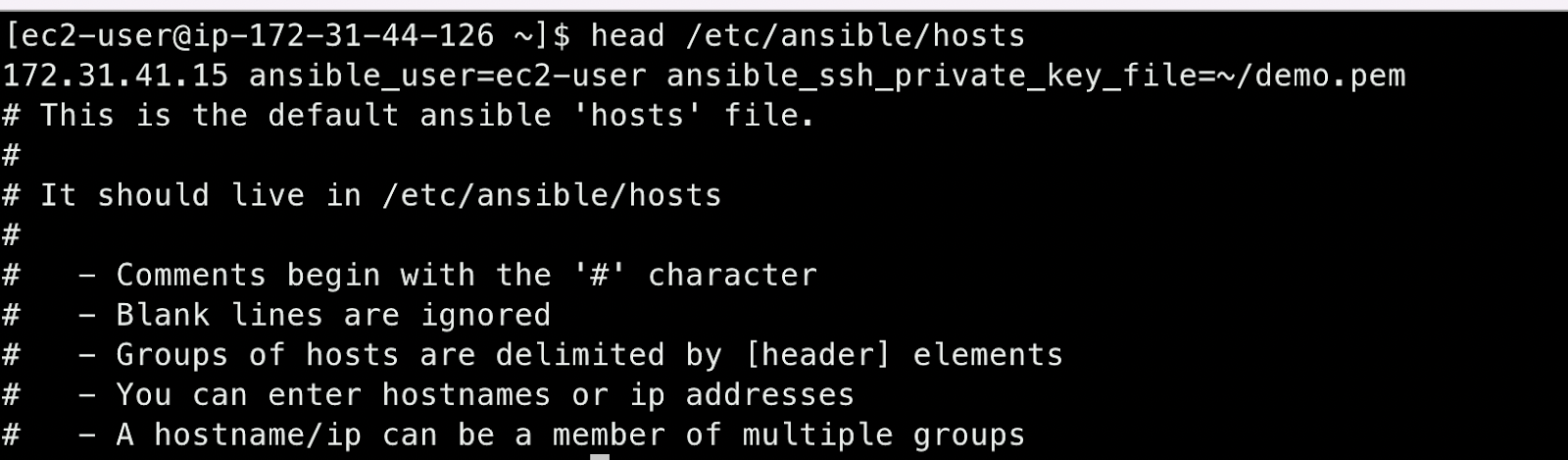
1. Get the server
2. Configure ansible inventory file

## Ansible Inventory File(FAQ)

Inventory file contains details about nodes like its ip, username, password and any variables required.

* The default inventory file is located at /etc/ansible/hosts
* We can change this and have custom location like
  + ansible-playbook -i dev.inv apache.yml

## Adding EC2 instance details to inventory



Ansible testing newly added node in inventory

ansible ip-address -m ping

## Ansible Ad-Hoc commands(FAQ)

An Ansible ad hoc command uses the /usr/bin/ansible command-line tool to automate a single task on one or more managed nodes. ad hoc commands are quick and easy, but they are not reusable.

ansible 172.31.41.15 -m yum -a 'name=git state=present' --become

ansible 172.31.41.15 -m yum -a 'name=git state=absent' --become

## Ansible Playbook

It is a YAML file where we describe a bunch of tasks we want to perform on a set of nodes. It is saved into a file and we can reuse it.

<https://github.com/javahometech/ansible-feb-2022/blob/master/apache.yml>

## Ansible Handlers (FAQ)

Handlers are tasks and they are executed only when notified

## Ansible Facts

* With Ansible you can retrieve or discover certain variables containing information about your remote systems or about Ansible itself.
* Variables related to remote systems are called facts.
* **Example**: *We are executing playbooks on bunch of servers (linux & ubuntu) and we want to run a task if it is linux, this is achieved using condition on facts*.
* (FAQ) Which module ansible uses to gather facts?
* Ans) Setup module.

## Ansible Conditions(FAQ)

We can conditionally run tasks in a playbook using “when”

<https://docs.ansible.com/ansible/latest/user_guide/playbooks_conditionals.html>

## Ansible tags

Using tags we can run only specific tasks and save execution time

ansible-playbook apache-tags.yml --tags install,deploy

ansible-playbook apache-tags.yml --skip-tags install

## Ansible Limit

For example a playbook is pointing to a group of 100 servers, running a playbook by default runs on all 100, but we want to limit the execution to specific servers. This is achieved using limit

ansible-playbook apache-tags.yml --limit all[0]

ansible-playbook apache-tags.yml --limit all[0:2]

## Ansible loop

<https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html#with-items>

## Rolling updates using Ansible

1. Ansible can be used for application deployments
2. By using serial keyword in ansible we can perform rolling updates

<https://docs.ansible.com/ansible/latest/user_guide/playbooks_strategies.html>

## (FAQ) How do you rerun playbook only failed hosts? For example, a playbook deployed a war file on 100 servers and it failed on 5 random servers. Now we want to run again only on failed servers.

Ans) ansible creates a temp file called ‘retry file’ this file contains ips of failed nodes and use this in –limit and run them.

## Setting up Nginx Server

1. Nginx is popularly know for reverse proxy
2. It can do load balancing
3. It can compress request and responses to improve performance
4. It can cache and improve performances
5. It can perform application firewall
6. Reverse proxy is very common in all devops projects

## Write ansible playbook to install and configure nginx

