

1 What is YAML?

YAML is just a **clean way to write configuration**.

Think of it like:

👉 A structured to-do list that computers understand.

Instead of clicking buttons in a UI, you describe your pipeline in text.

Example idea:

build → test → deploy

Tools read this YAML file and run your pipeline automatically.

2 Why DevOps Tools Use YAML

Modern CI/CD tools prefer YAML because:

- Pipelines are stored in Git
- Easy to read
- Easy to share
- Version controlled
- No manual clicking

So instead of creating jobs in Jenkins UI, you write a file like:

.gitlab-ci.yml

.github/workflows/main.yml

3 Basic YAML Rules (Super Important)

✓ Rule 1 — key : value

Everything is written like this:

name: value

Example:

stage: build

Left side = key (name)

Right side = value

✓ Rule 2 — Indentation Matters

Spaces show structure.

Correct:

```
job:  
  stage: build
```

Wrong:

```
job:  
stage: build
```

Use spaces — NOT tabs.

✓ Rule 3 — Lists Use “-”

Example:

```
script:  
  - npm install
```

- npm test

Dash means a new item in a list.

4 Pipeline Skeleton (Mental Model)

Almost every CI/CD YAML follows this structure:

```
stages:  
  - build  
  - test  
  - deploy
```

```
job-name:  
  stage: build  
  script:  
    - command
```

Simple explanation:

stages = order of execution

job-name = task label

script = commands to run

5 How YAML Maps to Jenkins

Jenkins pipeline:

```
stage('Build') {  
  steps {  
    sh 'mvn clean install'  
  }  
}
```

YAML pipeline:

```
build-job:  
  stage: build  
  script:  
    - mvn clean install
```

Mapping:

stage() → stage:

steps {} → script:

sh → - command

6 Real Example — Full CI/CD Pipeline

This example shows a full flow:

✓ Git clone (automatic)

✓ SonarQube scan

✓ Maven build

✓ Upload to Nexus

✓ Deploy to Tomcat

✓ Slack notification

stages:

- build
- sonar
- package
- deploy
- notify

variables:

MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"

SONAR_HOST_URL: "<http://sonarqube:9000>"

```
maven-build:
  stage: build
  image: maven:3.9.9-eclipse-temurin-17
  script:
    - echo "Building project..."
    - mvn clean compile
  artifacts:
    paths:
      - target/

sonarqube-check:
  stage: sonar
  image: maven:3.9.9-eclipse-temurin-17
  script:
    - echo "Running SonarQube analysis..."
    - mvn sonar:sonar
      -Dsonar.projectKey=my-app
      -Dsonar.host.url=$SONAR_HOST_URL
      -Dsonar.login=$SONAR_TOKEN

nexus-upload:
  stage: package
  image: maven:3.9.9-eclipse-temurin-17
  script:
    - echo "Packaging WAR file..."
    - mvn clean package
    - echo "Uploading to Nexus..."
    - mvn deploy -DskipTests=true
  dependencies:
    - maven-build

deploy-tomcat:
  stage: deploy
  image: alpine:latest
  before_script:
    - apk add --no-cache curl openssh-client
  script:
    - echo "Deploying to Tomcat server..."
    - scp target/myapp.war $TOMCAT_USER@$TOMCAT_HOST:/opt/tomcat/webapps/
    - ssh $TOMCAT_USER@$TOMCAT_HOST "systemctl restart tomcat"

slack-notify:
  stage: notify
  image: curlimages/curl:latest
```

```
script:  
- |  
  curl -X POST -H 'Content-type: application/json' \  
  --data '{"text":"Pipeline completed successfully!"}' \  
  $SLACK_WEBHOOK_URL  
when: on_success
```

7 How the Pipeline Runs (Simple Flow)

Pipeline engine reads YAML like this:

1. Look at stages list
2. Run all jobs in “build”
3. Then run jobs in “sonar”
4. Then package → deploy → notify

Important:

You are defining JOBS, not stages themselves.

Stage is only a label.

8 Common Beginner Mistakes

- ✗ Using tabs instead of spaces
- ✗ Wrong indentation level
- ✗ Forgetting “-” for command lists
- ✗ Putting passwords directly in YAML

Use environment variables for secrets instead.

9 Simple Cheat Sheet

Remember this tree:

```
stages
└ job
  └ stage
  └ image
  └ script
  └ variables
```

If you understand this shape, YAML pipelines become easy.

10 Final Simple Analogy

Jenkins = Clicking buttons to create pipeline

YAML = Writing a recipe:

Build:

```
  run install
```

Test:

```
  run tests
```

Deploy:

```
  copy files
```

That's all YAML really is — a structured instruction list.