**Title: Introduction to YAML**

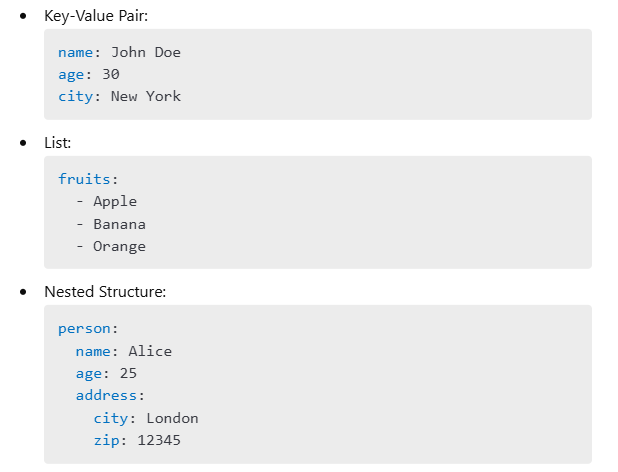
**What is YAML?**

* YAML (Yet Another Markup Language / YAML Ain't Markup Language) is a human-readable data serialization format.
* Used for configuration files, data exchange, and automation scripts.
* Simpler alternative to JSON and XML.

**Features of YAML**

* Uses indentation instead of brackets or tags.
* Supports key-value pairs, lists, and nested structures.
* Easy to read and write.
* Compatible with multiple programming languages.

**Basic YAML Syntax**



**YAML vs JSON**

| **Feature** | **YAML** | **JSON** |
| --- | --- | --- |
| Readability | More human-readable | More compact |
| Syntax Complexity | Simple indentation | Uses brackets & quotes |
| Comments | Supported (# for comments) | Not supported |
| File Size | Larger due to readability | More compact |

**Common Use Cases of YAML**

* Configuration files (e.g., Docker Compose, Kubernetes, Ansible).
* Data exchange between applications.
* CI/CD pipelines (e.g., GitHub Actions, GitLab CI/CD).

**YAML in DevOps & Automation**

* Kubernetes: Defines pods, deployments, and services.
* Ansible: Used for infrastructure automation.
* GitHub Actions: Automates software workflows.

**Best Practices in YAML**

* Maintain proper indentation (spaces, not tabs).
* Use comments to document configurations.
* Validate YAML using online tools or linters.
* Avoid unnecessary complexity.

**How to Create a .gitlab-ci.yml File?**

A .gitlab-ci.yml file is used to define a **CI/CD pipeline** in GitLab. It contains instructions for GitLab to build, test, and deploy your application automatically.

**Steps to Create a .gitlab-ci.yml File**

**Step 1: Navigate to Your GitLab Repository**

1. Log in to **GitLab**.
2. Open your **project repository**.

**Step 2: Create a New .gitlab-ci.yml File**

**Method 1: Using GitLab UI**

1. In your repository, go to **Repository → Files**.
2. Click **New File**.
3. Name the file **.gitlab-ci.yml**.
4. Add the pipeline script (example below).
5. Click **Commit Changes** to save.

**Method 2: Using Terminal (Local Development)**

1. Open the terminal and navigate to your project directory.
2. Run the following command to create the file:

sh

touch .gitlab-ci.yml

1. Open the file in a text editor (e.g., VS Code, Nano, or Vim):

sh

nano .gitlab-ci.yml

1. Add the pipeline script (example below).
2. Save and exit (Ctrl + X, then Y to confirm in Nano).

**Step 3: Add Pipeline Stages in .gitlab-ci.yml**

Here’s an example of a simple GitLab CI/CD pipeline:

yaml

stages:

- build

- test

- deploy

build\_job:

stage: build

script:

- echo "Building the application..."

- npm install # Install dependencies

test\_job:

stage: test

script:

- echo "Running tests..."

- npm test # Run tests

deploy\_job:

stage: deploy

script:

- echo "Deploying application..."

- echo "Deployment complete"

**Step 4: Commit and Push the File**

sh

CopyEdit

git add .gitlab-ci.yml

git commit -m "Added GitLab CI/CD pipeline"

git push origin main

**Step 5: Verify the Pipeline in GitLab**

1. Go to **CI/CD → Pipelines** in GitLab.
2. You should see your pipeline running automatically.
3. Click on a job to see logs and debugging information.

**Conclusion**

Now you have a working **GitLab CI/CD pipeline**! 🚀

* You can modify the .gitlab-ci.yml file to fit your project needs.
* Add deployment steps if required.
* Monitor pipeline executions under **CI/CD > Pipelines**.
* YAML is a powerful, human-readable configuration language.
* Used extensively in DevOps, automation, and data serialization.
* Mastering YAML simplifies configuration management and deployment.