**GITHUB ACTIONS**

1. What are GitHub Actions?

GitHub Actions is a CI/CD platform that lets you automate workflows directly in your GitHub repository — like testing, building, or deploying code — using YAML configuration files.

1. What is a workflow in GitHub Actions?

A **workflow** is an automated process defined in a YAML file inside .github/workflows/.  
It contains one or more **jobs**, triggered by specific **events** like a push, pull\_request, or a **manual trigger** (workflow\_dispatch).

A screenshot of a computer

AI-generated content may be incorrect.

1. What Are Jobs in GitHub Actions?

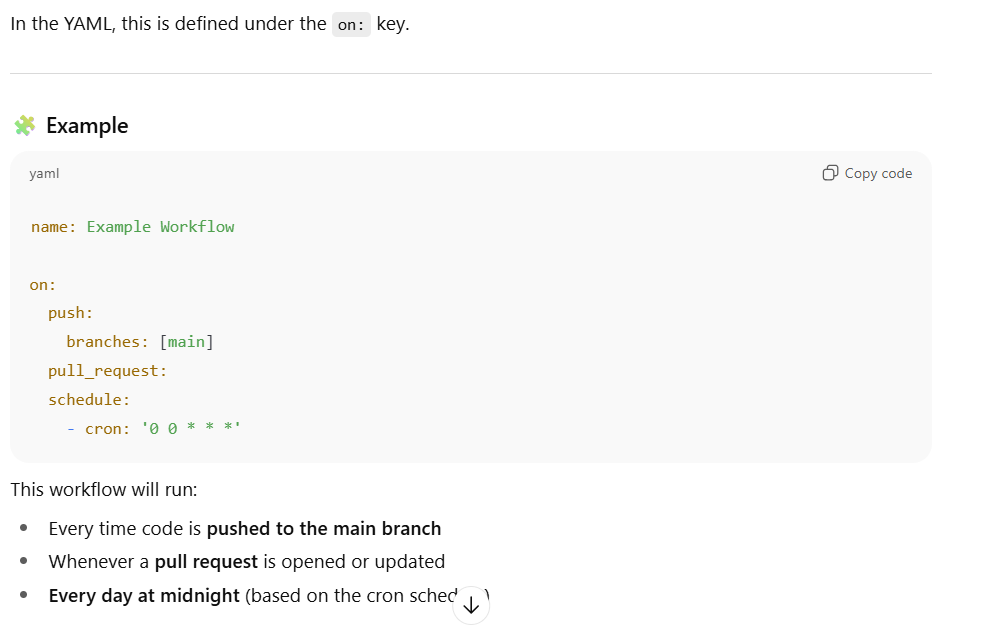
In **GitHub Actions**, a **job** is a **set of steps** that run together on the **same runner** (a virtual machine or container).  
Think of a job as **one stage in your CI/CD pipeline**.

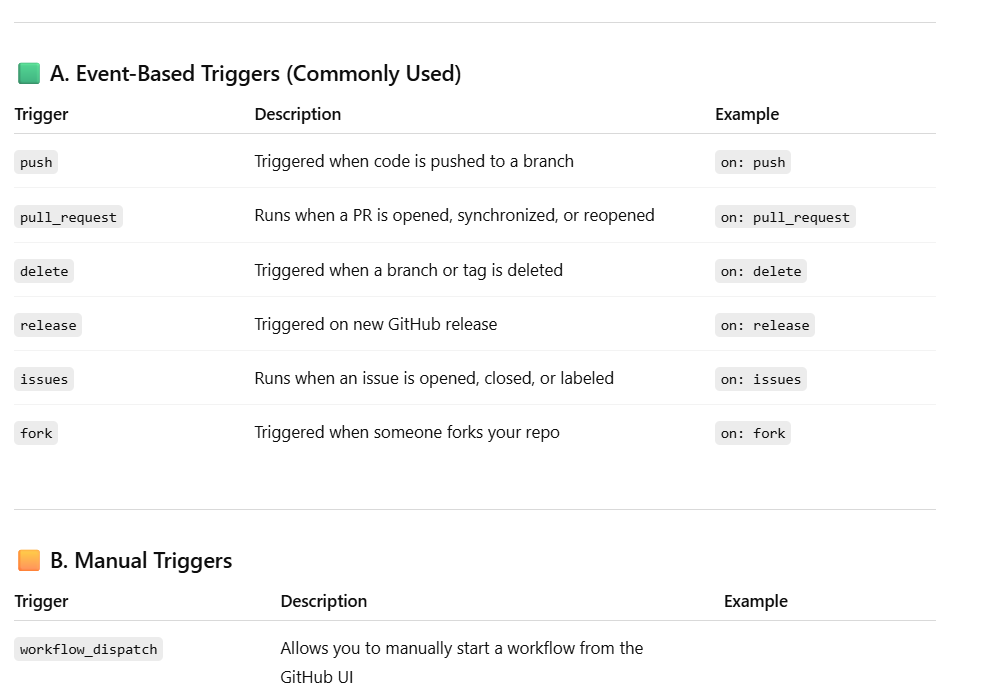
A screenshot of a computer

AI-generated content may be incorrect.

1. Explain workflow triggers.

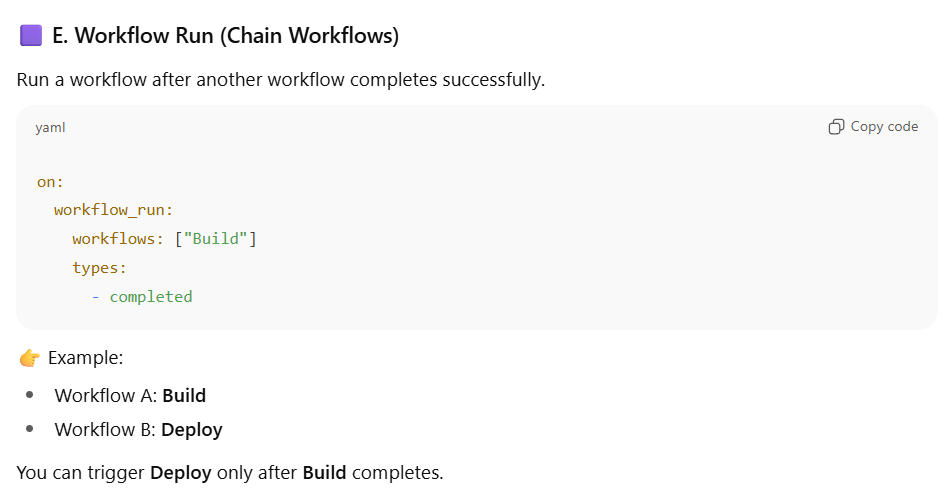
A **workflow trigger** defines **when and how** a workflow should run.  
It tells GitHub **what event** should start your workflow — like pushing code, opening a pull request, or running on a schedule.





A screenshot of a computer

AI-generated content may be incorrect.



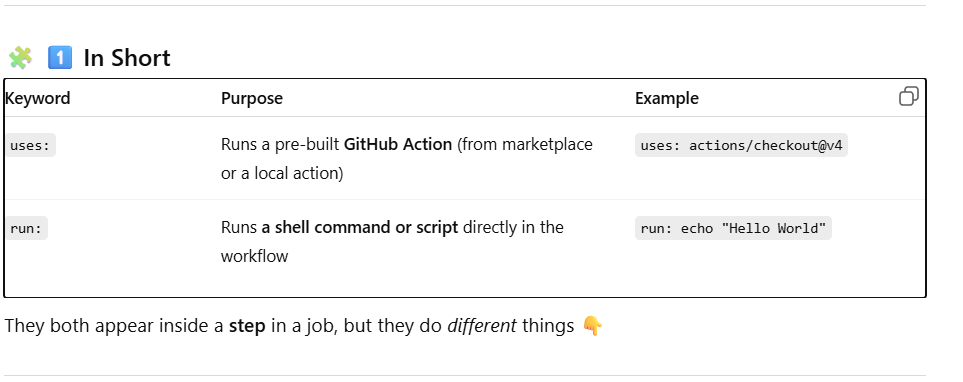
A screenshot of a computer

AI-generated content may be incorrect.

A person pointing at something

AI-generated content may be incorrect.

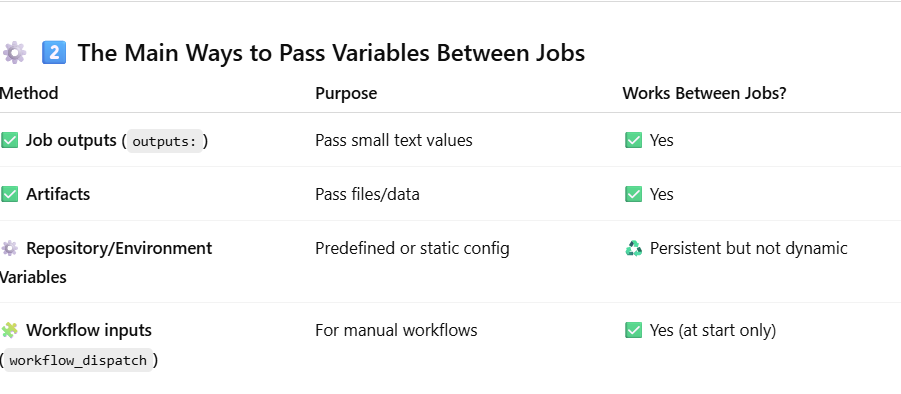
1. **What is the purpose of uses: and run: in GitHub Actions?**



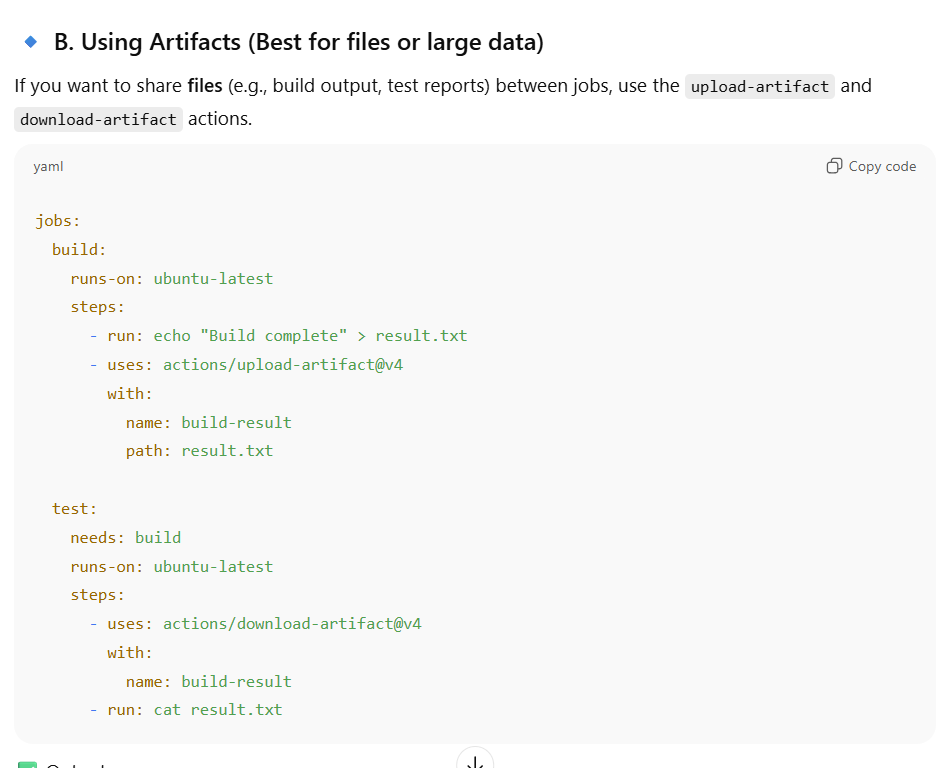
A screenshot of a computer

AI-generated content may be incorrect.

1. How can you pass variables between jobs?







A screenshot of a computer program

AI-generated content may be incorrect.

1. How to reuse workflows across repositories?

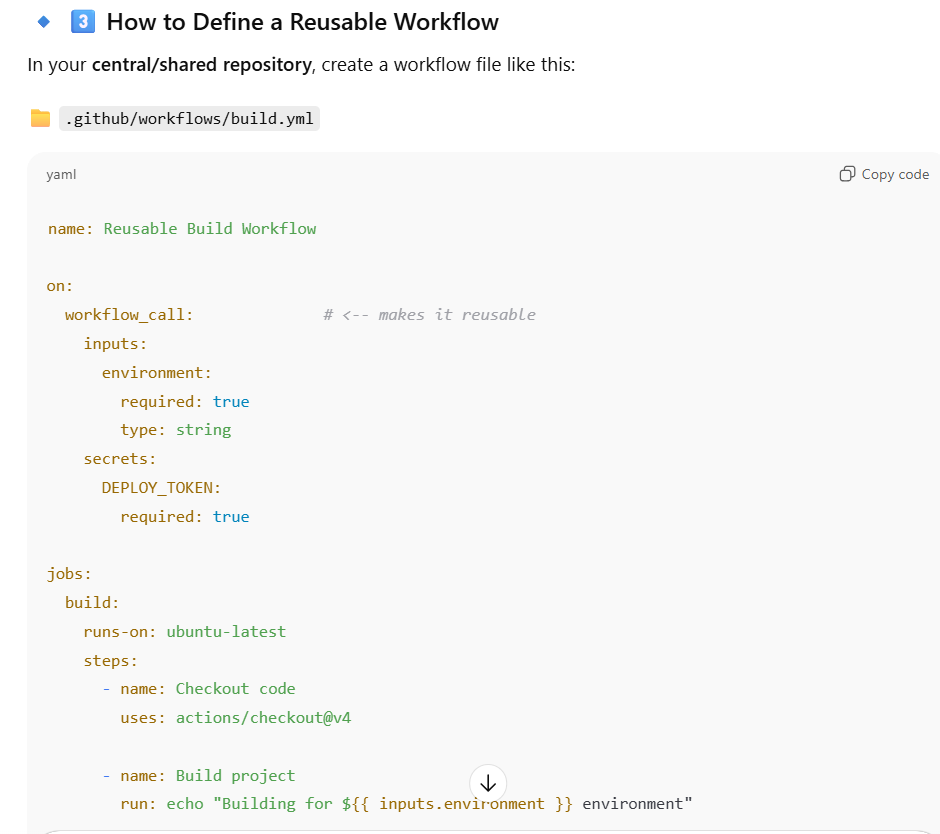
When you have multiple repositories that all need similar pipelines (like **build → test → deploy**),  
copying the same workflow to every repo leads to **duplication** and **maintenance pain**.

✅ Solution → Use **Reusable Workflows** (workflow\_call)

A **reusable workflow** is a **GitHub Actions workflow** designed to be **called by other workflows**,  
even from **other repositories**.

It behaves like a **function**:

* Define it once in one repo
* Reuse it anywhere (with different inputs)



A screenshot of a computer

AI-generated content may be incorrect.

1. How to handle matrix builds?

A **matrix build** runs **the same job** in **parallel** across different **combinations of parameters** — like:

* Multiple Python versions
* Multiple operating systems
* Multiple dependency versions

✅ Instead of writing multiple jobs manually, GitHub Actions **expands** one job into many.



A screenshot of a computer program

AI-generated content may be incorrect.

This creates:

* (ubuntu, 3.9)
* (ubuntu, 3.10)
* (windows, 3.9)
* (windows, 3.10)
* (macos, 3.9)
* (macos, 3.10)

✅ That’s **6 parallel jobs automatically**!

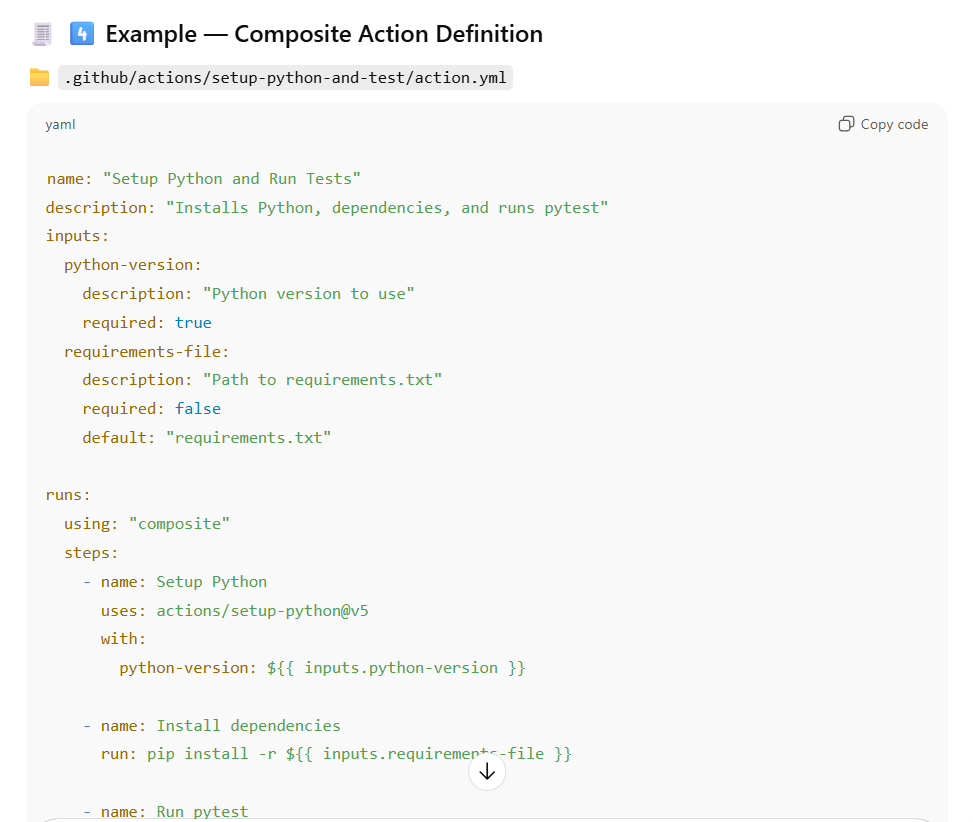
1. **What are composite actions?**

**A composite action is a custom, reusable action that you build using multiple shell steps (run / uses), just like a mini workflow.**

**A workflow snippet packaged as an action — so you can reuse it across jobs or repositories.**

**They are useful when you have a sequence of steps you repeat in many workflows.  
Instead of copying those steps everywhere, you can bundle them into a single action.**

****

****

**A screenshot of a computer program

AI-generated content may be incorrect.**

1. **What’s the difference between GITHUB\_ENV and GITHUB\_OUTPUT?**

 GITHUB\_ENV: Used to set environment variables for **later steps**.

 GITHUB\_OUTPUT: Used to set outputs for **later jobs**.

1. What are GitHub Action Contexts?

Contexts are objects containing metadata about workflow runs, jobs, and GitHub events.  
Examples:

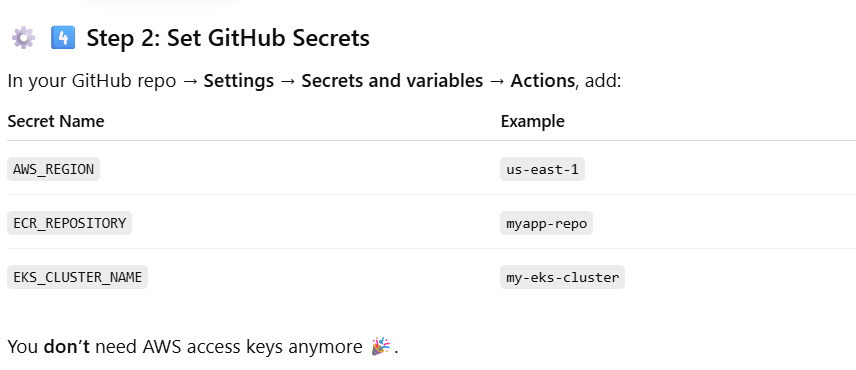
* ${{ github.ref }} → branch or tag name
* ${{ github.actor }} → username
* ${{ job.status }} → current job status

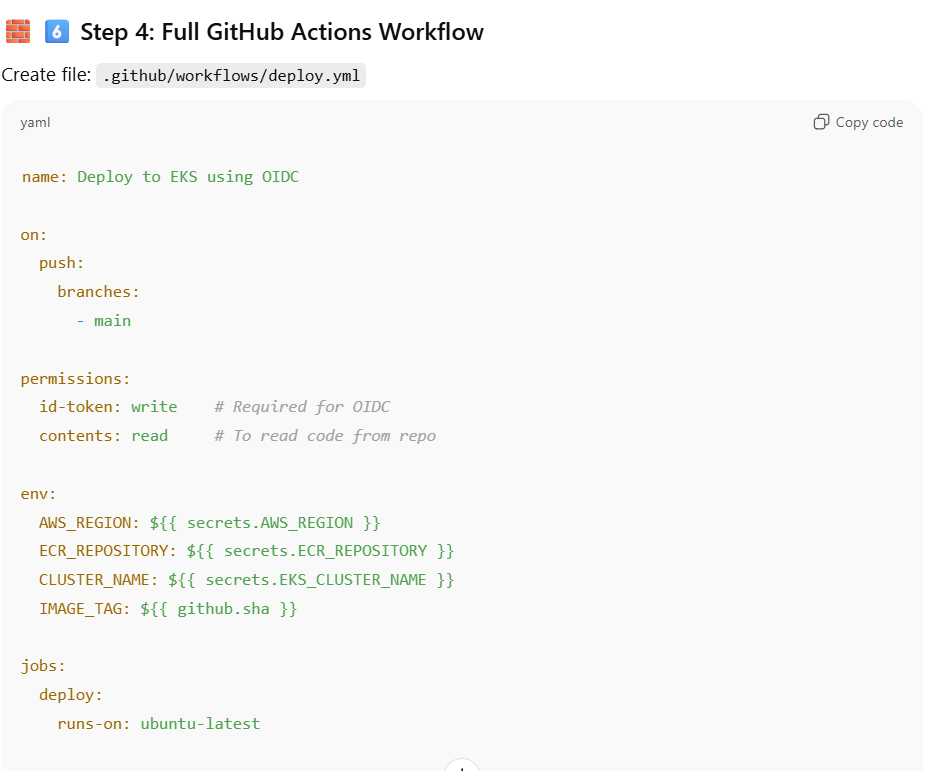


1. Deployment to EKS cluster using github actions.
   1. Use OIDC Instead of AWS Keys

✅ OIDC (OpenID Connect) method

* GitHub Actions **requests short-lived tokens** directly from AWS STS.
* No static credentials.
* Fine-grained IAM control per repo and environment.





A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

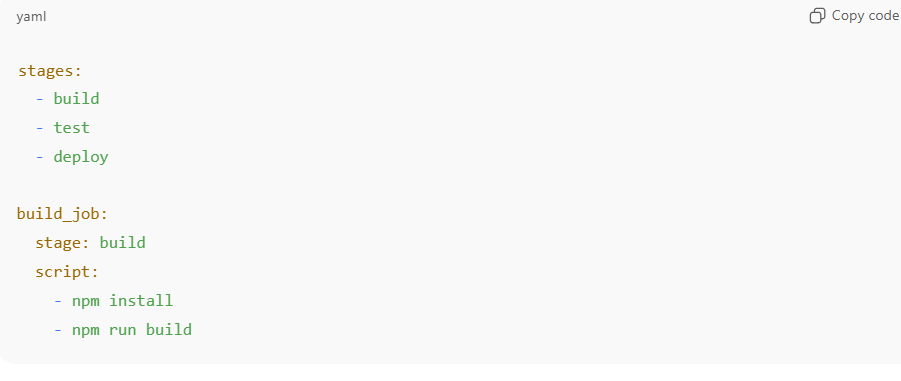
**GITLAB CI**

GitLab CI/CD is an integrated **continuous integration and continuous deployment** system that automates code build, test, and deployment using a .gitlab-ci.yml file stored in your repository.

* What is the .gitlab-ci.yml file?

It’s a YAML configuration file that defines the **pipeline**, specifying:

* **stages** (e.g., build, test, deploy)
* **jobs** (scripts to run)
* **rules** (when to run)
* **artifacts**, **caches**, etc.



* What are stages in GitLab CI/CD?

**Stages** define the **order in which groups of jobs run** in a GitLab pipeline.

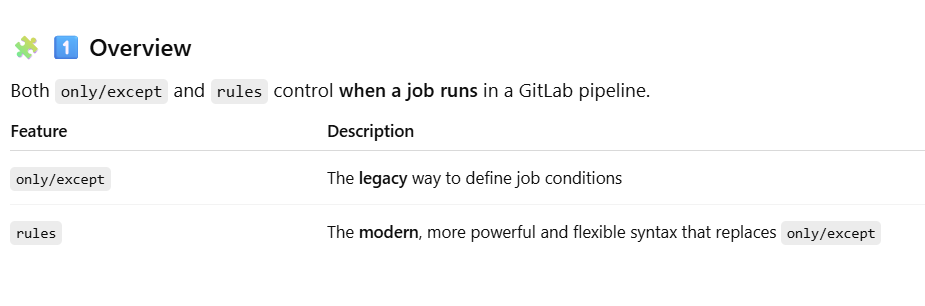
You can think of a pipeline as being divided into multiple **phases** (or **stages**) —  
like **build → test → deploy** — where:

* All **jobs in a stage** run **in parallel**
* **Next stage** starts **only if all jobs in the previous stage succeed**

✅ In short:

**Stages = pipeline phases**,  
**Jobs = tasks inside each stage.**

* What’s the difference between only/except and rules?



**⚙️ 2️⃣ Basic Concept**

* **only** → defines conditions where a job *should run*
* **except** → defines conditions where a job *should not run*
* **rules** → defines **if/else-like logic**, with multiple conditions, expressions, and actions

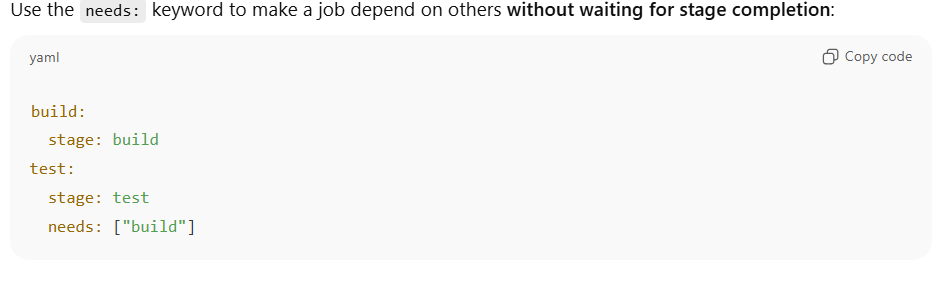
A screenshot of a computer program

AI-generated content may be incorrect.

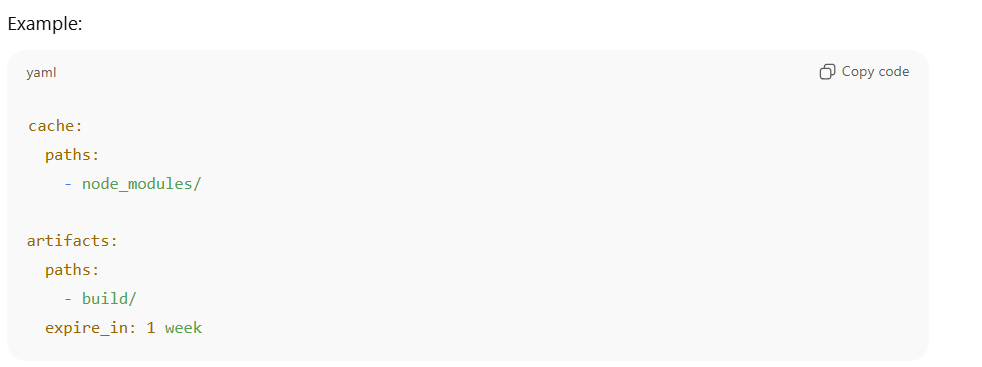
A screenshot of a computer

AI-generated content may be incorrect.

* How do you define dependencies between jobs?



* What is the difference between artifacts and cache?
* **Artifacts:** Files you want to **pass between jobs** (e.g., build outputs).
* **Cache:** Used to **speed up builds** by caching dependencies (e.g., npm, pip).

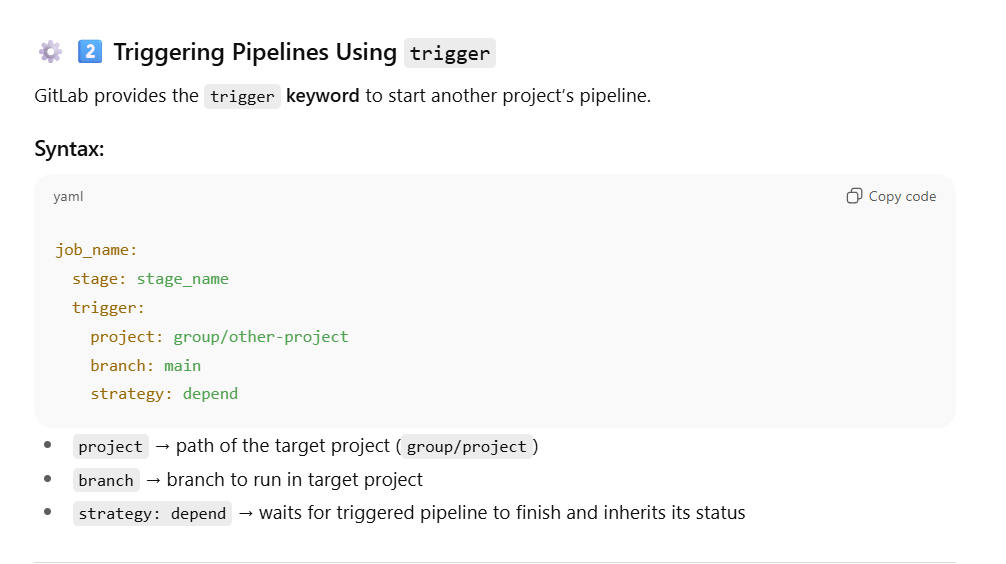


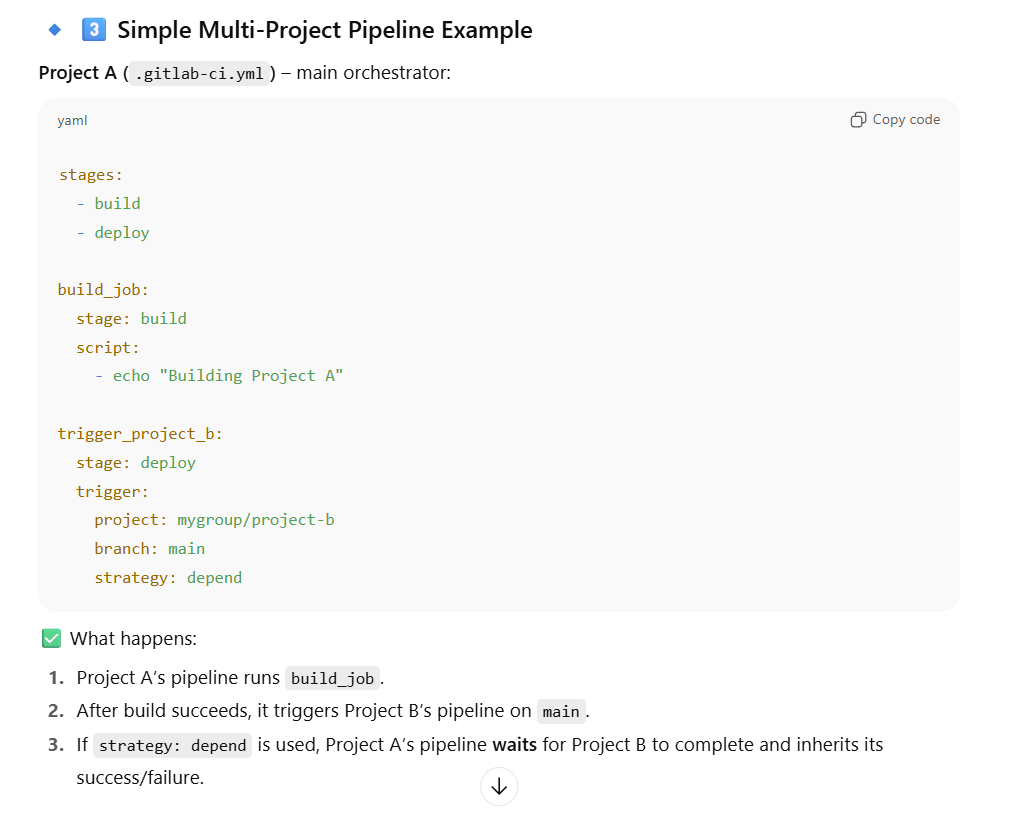
* How do you create a multi-project pipeline?

A **multi-project pipeline** is a GitLab pipeline where **one project triggers pipelines in other projects**.

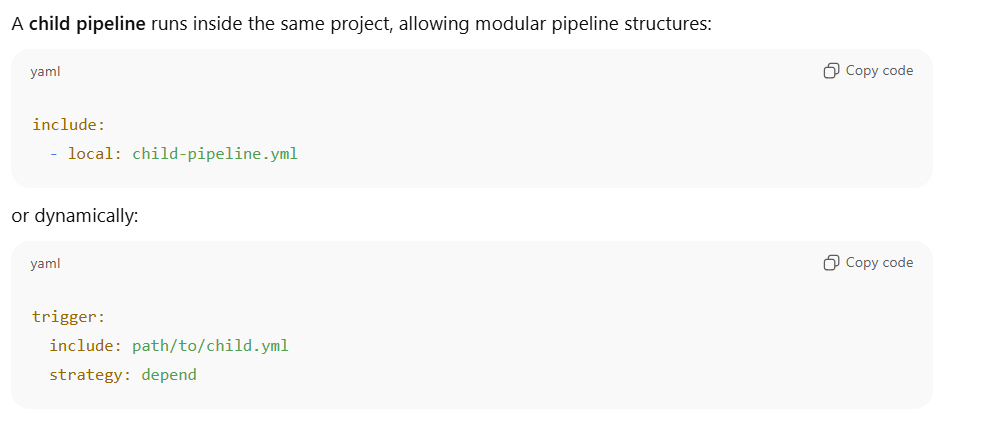
✅ Benefits:

* Orchestrates multiple repos from a **single source of truth**
* Allows **dependency management** (e.g., build library → test app)
* Helps implement **monorepo-style CI/CD without merging repos**





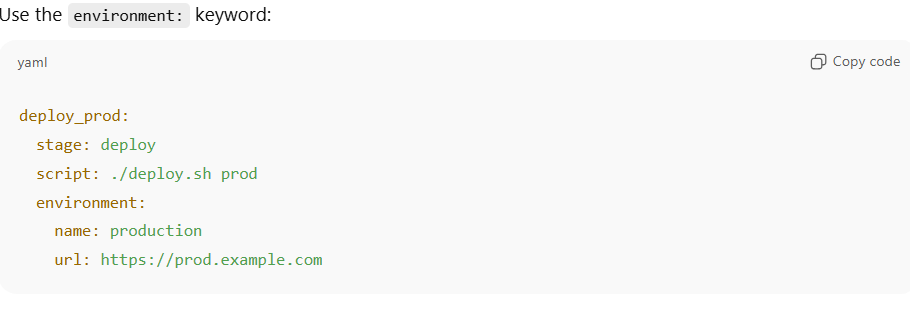
* What is a child pipeline?



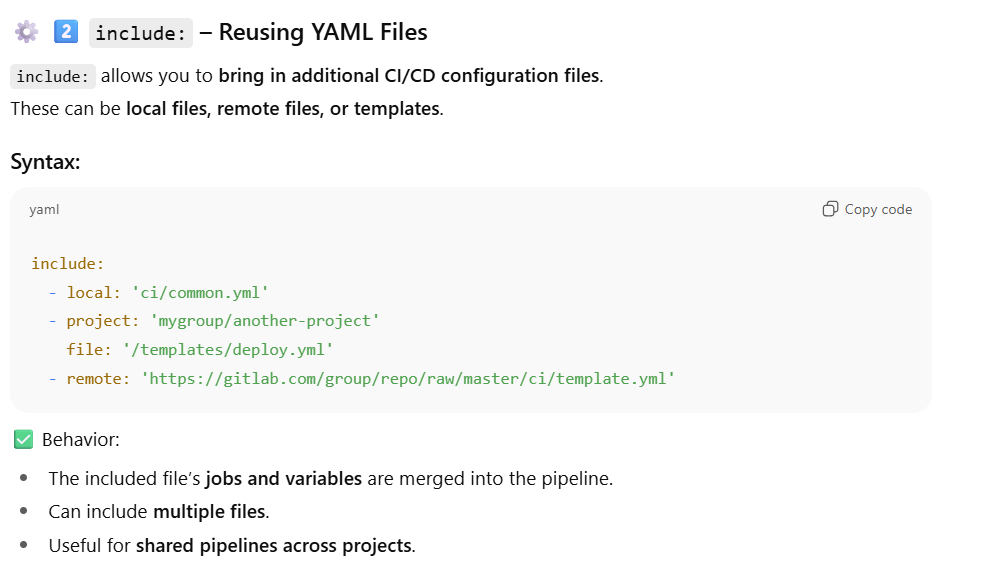
* What is extends: used for?



* How can you handle multiple environments (dev, stage, prod)?



* What’s the difference between include: and extends:?
* extends: inherits configuration from another job within the same pipeline.
* include: imports an entire file (reusable config or template).



A screenshot of a computer

AI-generated content may be incorrect.

* **What is the only: changes keyword used for?**

A white rectangular object with a black border

AI-generated content may be incorrect.

* How do you trigger pipelines on a schedule?

A white background with black text

AI-generated content may be incorrect.

* How do you pass artifacts from one pipeline to another?

A screenshot of a computer

AI-generated content may be incorrect.

**🔹 3️⃣ Passing Artifacts to Another Pipeline (Multi-Project Pipeline)**

When triggering **another project’s pipeline**, you can pass artifacts using **trigger: strategy: depend** and **artifacts: true** (GitLab 15+).





A screenshot of a computer

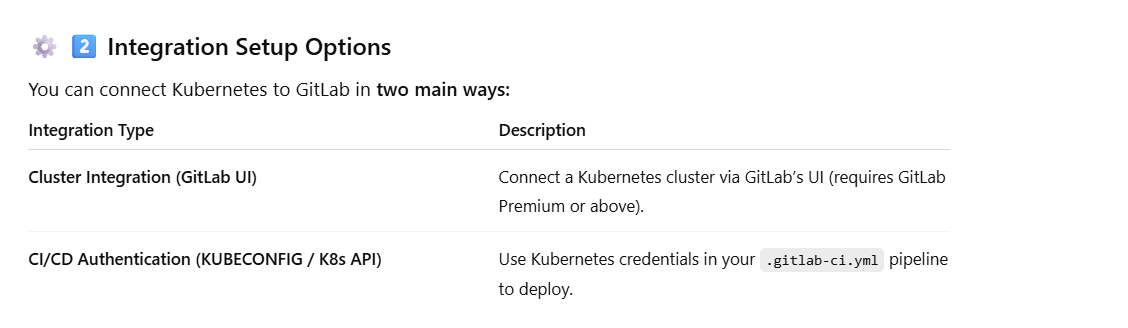
AI-generated content may be incorrect.



A white background with black text

AI-generated content may be incorrect.

* **How can you integrate GitLab with Kubernetes for deployments?**
* Automatically **deploy applications** to your Kubernetes cluster using GitLab CI/CD.
* **Monitor** and **manage environments** (like dev, staging, prod) directly from GitLab.



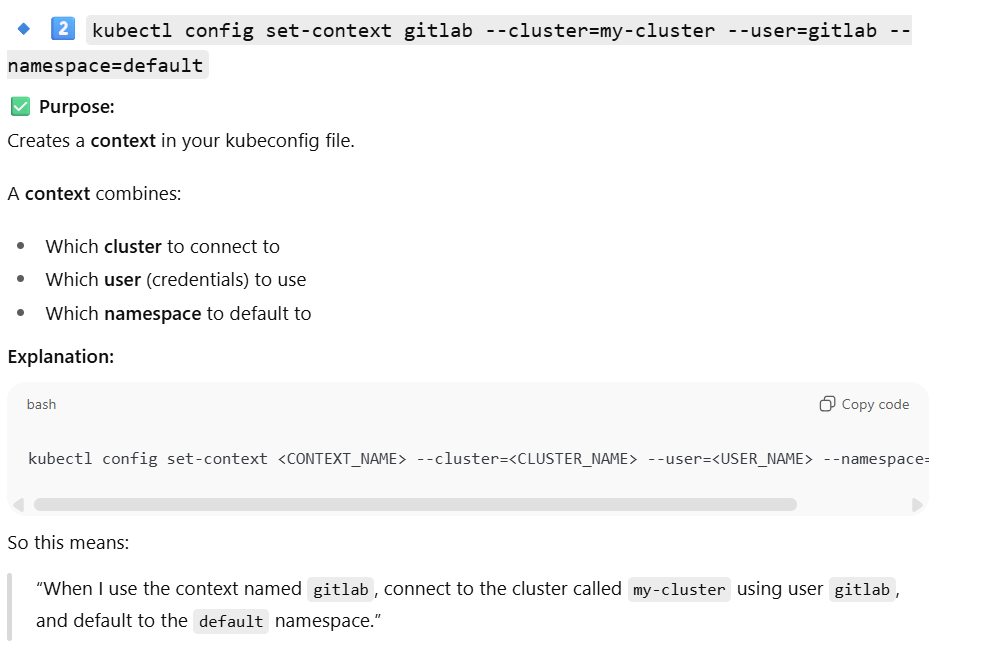
A screenshot of a computer

AI-generated content may be incorrect.

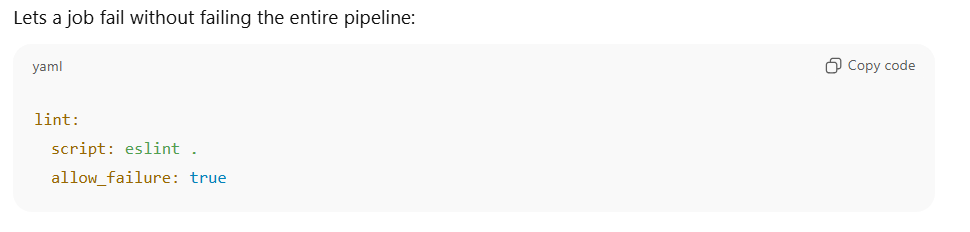
Example .gitlab-ci.yml for Kubernetes Deployment

A screenshot of a computer

AI-generated content may be incorrect.

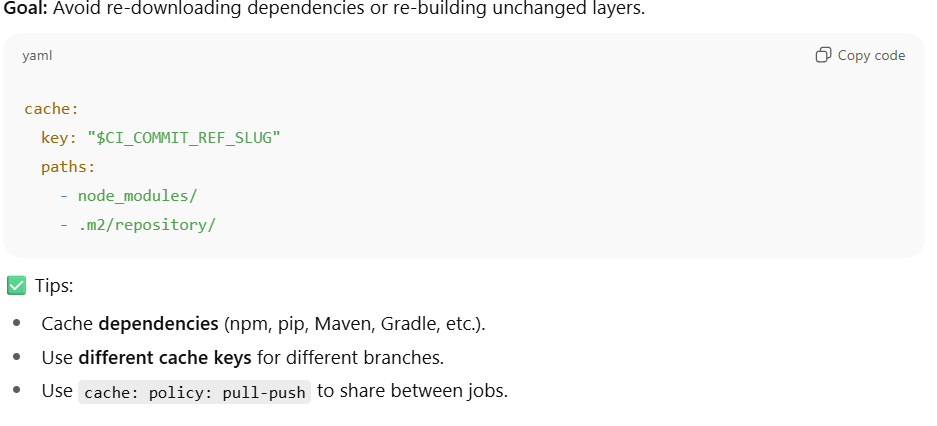


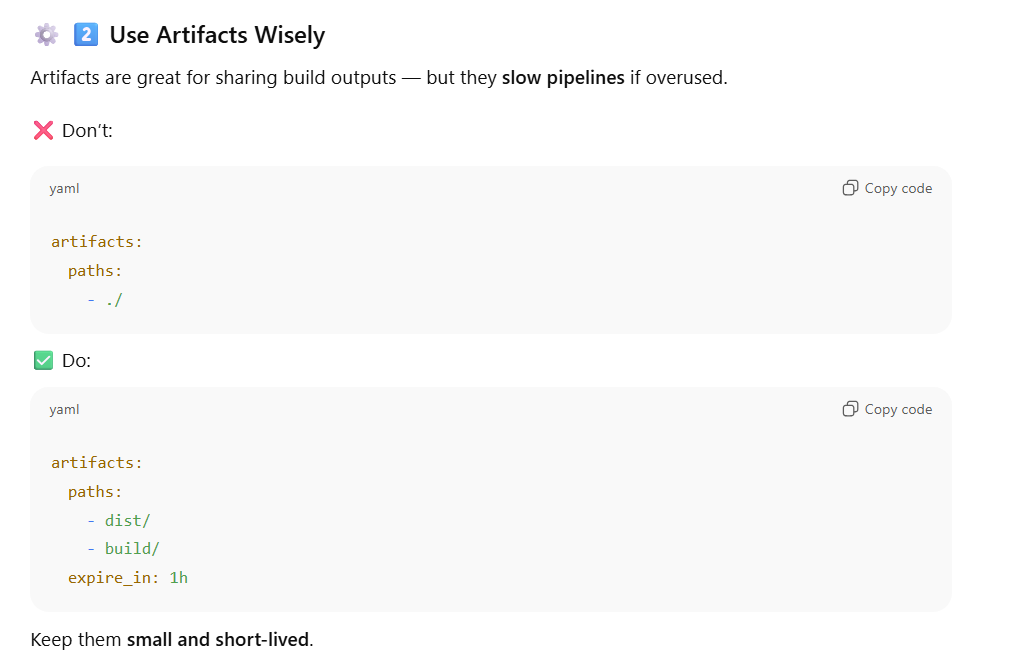
* What is the allow\_failure keyword?



* **How do you optimize pipeline speed?**

🚀 **1️⃣ Use Caching Effectively**





A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer error

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

