Error	Detailed Explanation	Fix
A step cannot have both the uses and run keys	This error arises in a GitHub Actions workflow where a step is defined with both uses and run. - **Actions vs. Commands**: The uses key is intended for specifying an action, typically from the GitHub Marketplace, that performs predefined operations. In contrast, run is used for executing shell commands directly within the step. - **Ambiguity Issue**: Having both creates ambiguity for the workflow runner, as it cannot determine whether it should execute an action or run a command, leading to a configuration error. - **Semantic Structure**: Each step should uniquely identify the operation intended, ensuring clear and unambiguous task	1. **Open** the workflow file. 2. **Locate** the step with both uses and run. 3. **Determine** the primary function of the step (action vs. command). 4. **Remove** the unnecessary key. 5. **Save**, **commit**, and **push** the file. 6. **Verify** the workflow runs correctly.

Error	Detailed Explanation	Fix
	execution within the CI pipeline.	
Invalid workflow file: .github/workflows/broken-action-2.yml#L11 YAML syntax error	This error points to a YAML syntax issue at line 11 of the file. - **YAML Sensitivity**: YAML files are white-space sensitive and require correct indentation and structure to be parsed successfully. - **Common Mistakes**: Syntax errors often include improper indentation (use spaces, not tabs), missing colons (:), unexpected characters, or incorrect list/mapping configurations. - **Parsing Failure**: Such syntax errors prevent the workflow from being interpreted accurately by GitHub Actions, causing it to fail loading or executing the intended actions. - **Corrective Action**: Proper formatting and validation are	1. **Open** the YAML file. 2. **Inspect** line 11 for syntax issues. 3. **Ensure** correct indentation and proper YAML formatting. 4. **Correct** any errors found. 5. **Validate** the syntax with a YAML linter. 6. **Commit** and **push** changes. 7. **Verify** the workflow proceeds without issues.

Error	**Detailed Explanation**	**Fix**
	crucial to ensure YAML files work effectively with CI/CD tools.	
/__w/_temp/...sh: 2: python3: not found Exit code 127	This error indicates that Python 3 is not available in the CI environment's PATH. - **Environment Configuration**: GitHub Actions runners don't have every version of Python pre-installed, so specifying and setting up the proper environment is essential. - **Meaning of Code 127**: This is a standard UNIX/Linux error code indicating a command could not be found, often due to PATH misconfiguration or missing software. - **Dependency Requirement**: Scripts relying on Python will fail if the interpreter isn't available, necessitating proper setup within workflow jobs to	1. **Use** actions/setup-python in your workflow. 2. **Specify** the necessary Python version. 3. **Ensure** dependency installation occurs after Python setup. 4. **Save**, **commit**, and **push** the workflow changes. 5. **Verify** proper execution in the CI environment.

Error	Detailed Explanation	Fix
	configure the environment before execution. - **Ensuring Availability**: Using setup actions like actions/setup-python ensures the specified version of Python is installed before attempting to run Python scripts.	
Run npm test > jest No tests found, exiting with code 1	This error occurs when Jest, a popular JavaScript testing framework, cannot find any test files to execute. - **Test File Lookup**: Jest looks for test files typically named *.test.js or *.spec.js. Incorrect naming conventions or directory misconfigurations can result in Jest not detecting tests. - **Configuration Issues**: Check your Jest configuration in package.json or a separate Jest config file. Settings like testMatch or roots can determine	1. **Ensure** test files follow naming conventions like *.test.js. 2. **Check and adjust** Jest configuration or test paths. 3. **Verify** consistent repository checkout with actions/checkout@v2. 4. **Run** tests locally using npm test to confirm they work. 5. **Add** debug printing with --listTests. 6. **Commit** and **push** changes, ensuring Jest finds and runs the tests in the CI environment.

Error	Detailed Explanation	Fix
	where Jest looks for test files. - **Checkout & Environment**: Ensure the test files are checked out correctly in the CI environment and verify by running npm test locally to confirm configurations are aligning properly across environments. - **Ensuring Visibility**: Using debug flags can help identify discrepancies in test detection during CI runs.	
npm ERR! ERESOLVE could not resolve	This npm error indicates a dependency conflict, particularly with peer dependencies. - **Version Conflicts**: For example, @typescript-eslint/eslint-plugin@4.28.2 requires a peer dependency on eslint versions ` ^5.0.0	

Error	**Detailed Explanation**	**Fix**
Run node app.js ENOENT: no such file or directory, open '/home/runner/work/my-repo/my-repo/.env'	The ENOENT error indicates a missing .env file, essential for loading environment variables your application needs. - **Secret Management**: .env files are often listed in .gitignore to prevent exposure of sensitive information, which means they don't get pushed to remote repositories. - **CI/CD Environment Setup**: This necessitates alternative methods for managing such environment configurations in CI/CD environments, like GitHub Actions, using secrets or configurable steps to generate .env files dynamically. - **Handling Sensitive Data**: Leveraging GitHub Secrets or similar tools can securely manage and inject environment variables needed	1. **Verify** the .env file exists locally and contains necessary variables. 2. **Use** GitHub secrets or equivalent for sensitive data instead of committing .env. 3. **Update** the workflow to include env variables (create .env during CI steps if needed). 4. **Test** both locally and in CI. 5. **Commit** any workflow or script handling updates, ensuring compliance with security best practices for handling sensitive configurations.

Error	Detailed Explanation	Fix
	without directly exposing or committing them to the repository. - **Application Reliability**: Ensuring the right configurations are available across all environments (local, testing, production) is crucial for application consistency.	
Run git push origin HEAD:refs/heads/main remote: Permission to user/repo.git denied	This error occurs when the GitHub Actions workflow attempts to push changes to a remote repository without the necessary permissions. - **Default Permissions**: GitHub Actions tokens have read-only permissions by default. Attempting to push changes requires write access, which needs to be configured separately. - **403 Forbidden**: The HTTP code indicates that the github-	1. **Review** repository permissions, ensuring CI/CD workflows have necessary write access. 2. **Use** a Personal Access Token (PAT) stored as a GitHub secret for authentication. 3. **Modify** the workflow to use the PAT, updating URLs to include x-access-token as shown. 4. **Commit** and **re-run** the workflow to verify success. 5. **Ensure** secure storage of PATs and follow best practices to prevent unauthorized access to your repository.

Error	Detailed Explanation	Fix
	actions[bot] user is not allowed to perform the operation, typically due to insufficient access rights. - **Authentication Strategy**: Using a Personal Access Token (PAT) can bypass these restrictions, as it can be granted specific scopes. - **Security Considerations**: When handling PATs, ensure they are stored securely as secrets and used with care to avoid exposing sensitive access credentials.	
Run docker build -t my-image . no matching manifest for linux/amd64 in the manifest list entries	This error occurs during the Docker build process because the specified base image (node:14) does not have a manifest that matches your platform's architecture (linux/amd64). - **Platform Mismatch**: This can	1. **Verify** the image tag on Docker Hub and ensure it supports linux/amd64. 2. **Specify** the platform during build with --platform flag. 3. **Update** Docker to its latest version. 4. **Use** Docker buildx for building multi-platform images. 5. **Try** alternate or more specific image tags if compatibility issues persist. 6. **Commit** and **test** the updated

Error	Detailed Explanation	Fix
	happen if the base image tag does not support certain architectures out-of-the-box. - **Multi-Arch Images**: Many official Docker images support multiple architectures, but some specific tags or older versions might not. - **Docker Hub**: Availability of specific architectures can be checked on Docker Hub, and using explicit platform specification during build can resolve such issues. - **Docker Configuration**: Ensuring proper Docker setup and leveraging tools like buildx for multi-platform builds enhances compatibility for diverse environments.	Docker configurations to ensure successful builds.
Run python script.py ModuleNotFoundErro	This error happens when Python can't	1. **Install** requests with pip install requests in the used

Error	Detailed Explanation	Fix
r: No module named 'requests'	find the requests module to import in your script. - **Missing Installation**: The requests library isn't installed in your environment, necessitating a pip install. - **Environment Misalignment**: The script may be running in a different environment than where requests is installed; verify the active environment aligns with your setup. - **Dependency Management**: Ensure requests is listed in a requirements.txt file to maintain consistent environments, crucially in CI/CD workflows. - **GitHub Actions and Local Sync**: Confirm that workflows are installing dependencies using	environment. 2. **Verify** you're using the correct Python environment, activating virtual environments if needed. 3. **Include** requests in requirements.txt and ensure dependencies are installed in workflows. 4. **Ensure** pip install -r requirements.txt is in your GitHub Actions setup. 5. **Check** installation by running a simple script importing requests and printing success, confirming environment consistency across local and CI settings.

Error	Detailed Explanation	Fix
	pip with requirements.txt to match local development setups. **Version Compatibility**: Check that requests suppor ts your Python version, updating or adjusting as necessary.	
.github/workflows/deploy.yml YAML syntax error: mapping values are not allowed in this context	- **YAML Structure and Indentation**: YAML syntax is highly sensitive to indentation and format, where each level should be consistently indented using spaces. - **Misplaced Elements**: Misplaced colons or incorrect formatting of key-value pairs can break the data structure. - **Map and Sequence Confusion**: Mixing lists and dictionaries without clear boundaries can lead to YAML parsing errors.	1. **Correct** the indentation and consistency of nested data. 2. **Ensure** all key-value pairs use colons correctly followed by single spaces. 3. **Validate** the file using YAML linters, addressing pointed errors. 4. **Test** the corrected YAML to confirm valid structure and proper functionality in GitHub Actions.

Error	Detailed Explanation	Fix
	- **Validation Importance**: YAML linters and online validators help identify line-specific errors by pointing out problematic syntax, ensuring your CI/CD tool executes workflows successfully without syntax issues.	
Process Killed	This message occurs when a running process is terminated forcibly, often due to resource limitations, like exceeding memory limits. - **Resource Bottlenecks**: High memory or CPU usage can trigger system limits or out-of-memory terminations, notably in resource-constrained environments like CI/CD runners. - **Timeouts and External Interventions**: Long-running processes may be terminated	1. **Optimize** the code for better resource usage, and look for efficiencies to reduce memory and CPU demands. 2. **Increase** available resources where possible if in cloud environments, enhancing CPU or memory capacity to meet demands. 3. **Profile and Monitor** application performance with logging and monitoring tools to determine resource-intensive phases and address them. 4. **Split Tasks** into smaller, more manageable units if applicable, or use batch processing solutions. 5. **Adjust** CI configurations to provide more forgiving time constraints or allocate additional resources to intensive tasks.

Error	**Detailed Explanation**	**Fix**
	by system constraints or external activities like automatic scripts or administrative actions.	6. **Debug** with detailed logs and information to determine specific termination points needing intervention.
Docker pull access denied	This error occurs when Docker can't pull an image from a registry due to access issues. - **Private Repositories**: The repository might be private and requires authentication. - **Incorrect Image Tag or Name**: The image may not exist as specified. - **Authorization**: Without proper credentials or token, Docker Hub or private registries deny access, leading to a 401 error.	1. **Validate** the repository name and tag, ensuring it exists. 2. **Log in** to Docker with docker login, using correct credentials or token. 3. **Use Secrets** in CI/CD workflows to handle Docker credentials securely, and configure your script to use them. 4. **Reconfirm** connectivity, ensure network paths to the registry are not restricted. 5. **Incorporate** error logging to identify connectivity/authentication issues, retrying connections appropriately.
npm ci ERESOLVE could not resolve dependency conflict	npm has detected dependency conflicts while using npm ci. This often involves incompatible peer dependencies. - **Version**	1. **Adjust** the conflicting packages if possible by updating or downgrading. 2. **Use** --legacy-peer-deps to bypass unresolved peer dependency errors for temporary builds. 3. **Review** dependency

Error	Detailed Explanation	Fix
	Mismatches: ttf-loader@1.0.2 requires file-loader in a specific version range incompatible with your installed file-loader. - **Peer Dependency Constraints**: These can limit version combinations, causing stricter installation errors during CI/CD executions.	compatibility, and confirm package-lock.json consistency. 4. **Update** workflow scripts to include npm ci --legacy-peer-deps for interim resolutions during automation.
[ERR_REQUIRE_ESM] ESM module issue with require()	The error arises when a project uses require() within ECMAScript Modules context, which conflics with Node.js ESM handling. - **ESM vs CommonJS**: Node.js' support for native ES Modules requires using import/export syntax. - **Project Configuration**: Projects using ESM need to use "type": "module" or .mjs extensions, avoiding CommonJS require u	1. **Convert** require() calls to import syntax for full ESM compliance. 2. **Ensure** your Node.js version supports ESM natively, or use --experimental-modules if legacy. 3. **Migrate** entire project to ESM if feasible, updating package.json with "type": "module". 4. **Handle** cross-compatibility using conditional imports or tools like esm package for transitional phase. 5. **Check and Use** compatible library versions, especially mocha, nyc, ensuring they support ESM or use dual packages.

Error	Detailed Explanation	Fix
	nless through compatible means.	
Docker manifest unknown error	Docker could not find the specified image manifest. - **Incorrect Tag or Repository Name**: Verify the spelling and existence. - **Deleted or Unpublished Image**: Ensure the image with the specified tag is available in the registry. - **Dynamic Tags**: Using generated or unsynchronized tags can lead to retrieval failures.	1. **Verify** the image tag and repository name for spelling or existence issues. 2. **Ensure** that the image is built and pushed correctly if it is part of a CI/CD pipeline. 3. **Check Registry** whether the image version exists and is publicly accessible in case of a private registry. 4. **Correct Tag** if the tag is dynamically generated, confirm synchronization with the actual published tags.
SuperTokens 404 and E2E Timeout Error	A GET request to a SuperTokens endpoint returned 404, while a follow-up UI test timed out. - **Missing Endpoint**: Confirm the SuperTokens path is correct. - **UI State**: The timeout suggests the page state was not reached, likely affected by the backend 404.	1. **Check** the existence and correct configuration for the SuperTokens endpoint. 2. **Validate** your application's state or conditions for rendering UI elements needed. 3. **Increase** test timeout to accommodate slow loading in CI. 4. **Ensure** all dependent services are available before tests start. 5. **Debug & Log** the script to

Error	Detailed Explanation	Fix
	- **Resource Availability**: Ensure required backend resources are running for the test cases.	monitor application state throughout testing workflow.
Audit broken links error	The audit tool identifies URLs in your project that no longer resolve, indicating maintenance issues. - **Project Integrity**: Provides feedback on codebase quality, ensuring external resources remain accessible. - **User Experience**: Usability and trust may be impacted if users encounter broken links.	1. **Verify** broken links manually and update resources or references. 2. **Remove** outdated or unreachable URLs if needed. 3. **Use Link Checkers** periodically to automate URL validation and catch issues earlier. 4. **Employ Redirects** if you own domains with changed URLs, maintaining continuity.