# Microsoft.GH-200-JPN.v2026-09-02.q60

| | |
|---|---|
| 試験コード： | GH-200-JPN |
| 試験名称： | GitHub Actions (GH-200日本語版) |
| 認証ベンダー： | Microsoft |
| 無料問題の数： | 60 |
| バージョン： | v2026-09-02 |
| ページの閲覧量： | 110 |
| 問題集の閲覧量： | 727 |

https://www.jpnshiken.com/shiken/Microsoft.GH-200-JPN.v2026-09-02.q60.html

質問: **1**
開発者として、ワークフロー内でDockerコンテナアクションを使用しています。このアクションを正常に実行するために必要なことは何ですか？
**A.** ジョブ環境はLinux環境に設定する必要があります。
**B.** アクションはGitHub Marketplaceに公開する必要があります。
**C.** 参照されているアクションはDocker Hubでホストされている必要があります。
**D.** ジョブの実行環境は、Dockerがインストールされ たLinuxマシンを指定する必要があります。
正解: **(**正解を表示します**)**
To run a Docker container action, the job's runs-on environment must be a Linux machine that has Docker installed.
Key Requirements
Linux OS Only: Docker container actions are only supported on Linux runners. They cannot run on Windows or macOS runners.
Docker Installed: The runner must have the Docker CLI and daemon installed and running to build and execute the action's container.
GitHub-Hosted Runners: If you use GitHub-hosted runners, you must specify an Ubuntu environment (e.g., runs-on: ubuntu-latest), as these come pre-configured with Docker.
Self-Hosted Runners: For self-hosted runners, you must manually ensure the machine is running Linux and has Docker installed.
Reference:
https://docs.github.com/actions/sharing-automations/creating-actions/creating-a-docker- container-action

質問: **2**
GitHub の外部で発生するアクティビティに対して、GitHub API を使用してワークフローをトリガーする必要があります。
どのワークフロー イベントを使用しますか?
**A.** workflow_run
**B.** デプロイメント
**C.** チェックスイート
**D.** リポジトリディスパッチ
正解: **(**正解を表示します**)**
You can use the GitHub API to trigger a webhook event called repository_dispatch when you want to trigger a workflow for activity that happens outside of GitHub .
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/events-that-trigger-workflows

質問: **3**
開発者としてワークフローを設計しており、環境変数の設定、他のアクションで使用される出力値、出力ログへのデバッグメッセージの追加などのタスクを実行するためにランナーマシンと通信する必要があります。以下のオプションのうち、どれを使用すべきでしょうか？
**A.** 複合実行ステップ
**B.** デバッグログを有効にする
**C.** セルフホストランナー
**D.** 環境変数
**E.** ワークフローコマンド
正解: **E (**コメントを発表する**)**
Workflow commands for GitHub Actions
You can use workflow commands when running shell commands in a workflow or in an action's code.
About workflow commands
Actions can communicate with the runner machine to set environment variables, output values used by other actions, add debug messages to the output logs, and other tasks.
Most workflow commands use the echo command in a specific format, while others are invoked by writing to a file.
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-commands

質問: **4**
GitHub Enterprise Server インスタンスでの GitHub Actions の使用に関して正しい記述はどれですか。(それぞれの正解は完全な解決策を示しています。3 つ選択してください。)
**A.** GitHub によって作成されたアクションは自動的に使用可能になり、無効にすることはできません。
**B.** GitHub Connect を構成することで、GitHub Enterprise Server でサードパーティのアクションを使用できます。
**C.** GitHub で作成されたアクションのほとんどは、GitHub Enterprise Server で使用するために自動的にバンドルされます。
**D.** GitHub Enterprise Server で GitHub Actions を使用するには、永続的なインターネット接続が必要です。
**E.** アクションは .github リポジトリで定義する必要があります。
**F.** サードパーティのアクションは、GitHub Enterprise Server で使用するために手動で同期できます。
正解: **B,C,F (**コメントを発表する**)**
[B] If you want your GitHub Enterprise Server instance to use third-party custom actions, you need to enable GitHub Connect.
[C] Most official GitHub-authored actions are automatically bundled with GitHub Enterprise Server, and are captured at a point in time from GitHub Marketplace.
[F] Third-party actions can be manually synchronized to GitHub Enterprise Server (GHES) to be used in workflows without requiring an internet connection to GitHub.com. This is done using the actions-sync tool to download actions from GitHub.com and transfer them to the GHES instance.
This manual method provides an alternative to using GitHub Connect, offering stricter control over which actions are available on the GHES instance.
Reference:
https://docs.github.com/en/enterprise-server@3.17/get-started/exploring-integrations/about- using-integrations

https://docs.github.com/en/enterprise-server@3.17/admin/managing-github-actions-for-your- enterprise/managing-access-to-actions-from-githubcom/about-using-actions-in-your-enterprise

質問**: 5**
Windows x64 セルフホストランナーが 1 つだけあり、カスタムツールで構成されています。
そのランナーをターゲットにするには、ワークフローでどの構文を使用できますか?
**A.** 実行環境: [セルフホスト、Windows、x64]
**B.** セルフホスト: [windows-x64]
**C.** 実行環境: Windows 最新版
**D.** セルフホスト: [windows, x64]
正解**: (**正解を表示します**)**
Using self-hosted runners in a workflow
To use self-hosted runners in a workflow, you can use labels or groups to specify the runner for a job.
Using default labels to route jobs
A self-hosted runner automatically receives certain labels when it is added to GitHub Actions.
These are used to indicate its operating system and hardware platform:
self-hosted: Default label applied to self-hosted runners.
linux, windows, or macOS: Applied depending on operating system.
x64, ARM, or ARM64: Applied depending on hardware architecture.
You can use your workflow's YAML to send jobs to a combination of these labels. In this example, a self-hosted runner that matches all three labels will be eligible to run the job:
runs-on: [self-hosted, linux, ARM64]
self-hosted - Run this job on a self-hosted runner.
linux - Only use a Linux-based runner.
ARM64 - Only use a runner based on ARM64 hardware.
Reference:
https://docs.github.com/en/actions/how-tos/manage-runners/self-hosted-runners/use-in-a- workflow

質問**: 6**
アクションを他のアプリケーション コードにバンドルするのではなく、独自のリポジトリに保持する 2 つの理由は何ですか? (それぞれの正解は完全なソリューションを示します。2 つ選択してください。)
**A.** action.yml ファイルをオプションにします。
**B.** GitHub コミュニティがアクションを見つけやすくなります。
**C.** 開発者が問題を修正し、アクションを拡張するためのコードベースの範囲が広がります。
**D.** アクションのバージョン管理を他のアプリケーション コードのバージョン管理から分離します。
**E.** ワークフローの秘密を他のユーザーと共有できます。
正解**: (**正解を表示します**)**
Storing an action in its own repository makes it easier for the GitHub community to discover the action [B], narrows the scope [Not C] of the code base for developers fixing issues and extending the action, and decouples the action's versioning from the versioning of other application code

[D].
Reference:
https://docs.github.com/en/actions/how-tos/create-and-publish-actions/manage-custom-actions

質問: **7**
APIキーをGitHubリポジトリに保存する必要があります。このソリューションでは、キーの安全性を確保し、GitHub Actionsワークフローで使用できるようにする必要があります。どうすればよいでしょうか？
**A.** キーをプレーンテキスト変数としてリポジトリのREADMEファイルに追加します。
**B.** .env ファイルにキーをリポジトリにコミットします。
**C.** リポジトリのシークレットを使用してキーを暗号化して保存し、ワークフローのYAMLファイルでキーを参照します。
**D.** キーをワークフロー YAML ファイルの env: に直接保存します。
正解: **(正解を表示します)**
To use a specific API key in your GitHub Actions workflow, you can store it as a Repository Secret and reference it using the secrets context in your YAML file. This ensures the key is encrypted and masked in your workflow logs.
1. Store the API Key as a Secret
Navigate to your repository on GitHub.com.
Click Settings > Secrets and variables > Actions.
Select the Secrets tab and click New repository secret.
In the Name field, enter a descriptive identifier (e.g., MY_API_KEY).
Rules: Use only alphanumeric characters or underscores; cannot start with a number or GITHUB_.
In the Secret field, paste your API key value and click Add secret.
2. Reference the Secret in your YAML File
In your .github/workflows/your-workflow.yml file, access the secret using the syntax ${{ secrets.NAME_OF_SECRET }}. It is a best practice to map it to an environment variable within a specific step rather than echoing it directly.
Example Workflow in yaml:
name: Use API Key Example
on: [push]
jobs:
my-job:
runs-on: ubuntu-latest
steps:
- name: Run script with API Key
# Map the secret to an environment variable for the runner
env:
API_KEY: ${{ secrets.MY_API_KEY }}
run: |
# Use the environment variable in your commands
curl -H "Authorization: Bearer $API_KEY" https://api.example.com
Reference:

https://docs.github.com/actions/security-guides/using-secrets-in-github-actions

質問**: 8**
ワークフローはどの場所でアクションを参照できますか? (それぞれの正解は完全なソリューションを示します。3 つ選択してください。)
**A.** リポジトリ内の.action拡張子のファイル
**B.** Docker Hub に公開された Docker コンテナ イメージ
**C.** パブリック NPM レジストリ
**D.** ワークフローファイルの runs-on: キーワード
**E.** リポジトリのシークレット設定ページ
**F.** 別の公開リポジトリ
**G.** ワークフローと同じリポジトリ
正解**: B,C,F (**コメントを発表する**)**
Adding an action to your workflow
You can add an action to your workflow by referencing the action in your workflow file. The actions you use in your workflow can be defined in:
[F] The same repository as your workflow file
[C] Any public repository
[B] A published Docker container image on Docker Hub
Note: Adding an action from the same repository [F]
If an action is defined in the same repository where your workflow file uses the action, you can reference the action with either the {owner}/{repo}@{ref} or ./path/to/dir syntax in your workflow file.
Reference:
https://docs.github.com/en/actions/how-tos/write-workflows/choose-what-workflows-do/find-and- customize-actions

質問**: 9**
ワークフローを無効にすると、リポジトリからファイルを削除することなく、ワークフローのトリガーを停止できます。ワークフローを一時的に無効にすることが最も有効なシナリオはどのようなものですか？
(それぞれの正解は完全な解決策を示しています。2 つ選択してください。)
**A.** ランナーでは診断ログを有効にする必要があります。
**B.** ワークフロー エラーにより、要求が多すぎるか間違った要求が生成され、外部サービスに悪影響を及ぼします。
**C.** ワークフローは、セルフホストランナーで実行するように構成されています。
**D.** ワークフローは、ダウンしているサービスにリクエストを送信します。
**E.** ワークフローをスケジュール実行から手動トリガーに変更する必要があります。
正解**: B,D (**コメントを発表する**)**
Temporarily disabling a workflow can be useful in many scenarios. These are a few examples where disabling a workflow might be helpful:
[B] A workflow error that produces too many or wrong requests, impacting external services negatively.
A workflow that is not critical and is consuming too many minutes on your account.
[D] A workflow that sends requests to a service that is down.

Workflows on a forked repository that aren't needed (for example, scheduled workflows).
Reference:
https://docs.github.com/en/actions/how-tos/manage-workflow-runs/disable-and-enable-workflows

質問: **10**
あなたは、.NETアプリをビルドするGitHub Actionsワークフローを開発しています。
ワークフローに機密性の高い企業ライセンスへのアクセス権を与える必要があります。ソリューションは、ライセンスが公開されたり、ハードコードされたりしないようにする必要があります。
あなたはどうすべきでしょうか？
**A.** GitHubでホストされているWindowsランナーを使用し、実行時にワークフローによってダウンロードされる公開URLを介してライセンスを公開します。
**B.** ライセンスをリポジトリに保存された暗号化されたZIPファイルに埋め込み、ハードコードされたパスフレーズを使用して実行時にファイルを復号化する。
**C.** GitHub ホストまたは自己ホストの Windows ランナーで GitHub Enterprise Managed User (OIDC) を使用して、実行時にサードパーティのボールトまたはシークレット マネージャーからライセンスを取得します。
**D.** GitHubでホストされているUbuntuランナーを使用し、ライセンスをBase64エンコードされたリポジトリに保存します。
正解: **D (**コメントを発表する**)**
To build a .NET application while securely handling a sensitive enterprise license on a GitHub- hosted Ubuntu runner, you should store the Base64-encoded license as a GitHub Repository Secret.
Key Security Considerations
Ephemeral Runners: GitHub-hosted runners are wiped after every job, meaning the decoded license file is automatically deleted.
Log Redaction: GitHub attempts to redact any output that matches your secret value, but manual transformations (like decoding) should still be handled carefully.
No Artifacts: Avoid using upload-artifact for any directories containing the decoded license, as artifacts can be downloaded by anyone with repository access
1. Store the License as a Secret
First, encode your license file to Base64 locally to ensure it is stored as a single string without formatting issues:
# On your local machine (Linux/macOS)
cat license.lic | base64 -w 0 > license_base64.txt
Then, add this string to your repository by navigating to Settings > Secrets and variables > Actions > New repository secret. Name it ENTERPRISE_LICENSE_BASE64.
2. Configure the GitHub Actions Workflow
Create a .yml file in .github/workflows/ that decodes the secret into a temporary file on the Ubuntu runner. GitHub automatically masks the secret value in logs, ensuring it remains private.
name: Build .NET App with License
on:
push:
branches: [ "main" ]
jobs:
build:

runs-on: ubuntu-latest # Use GitHub-hosted Ubuntu runner
steps:
- name: Checkout code
uses: actions/checkout@v4
- name: Setup .NET
uses: actions/setup-dotnet@v4
with:
dotnet-version: '8.0.x' # Specify your version
- name: Decode License
shell: bash
run: |
# Decode the secret into a file for the build process
echo "${{ secrets.ENTERPRISE_LICENSE_BASE64 }}" | base64 --
decode > ./license.lic
# The runner is ephemeral; the file is destroyed when the job ends.
- name: Restore dependencies
run: dotnet restore
- name: Build Application
# Pass the license path if your build process requires it
run: dotnet build --configuration Release --no-restore /p:LicensePath=./license.lic
- name: Secure Cleanup (Optional)
if: always()
run: rm -f ./license.lic
Reference:
https://docs.github.com/actions/security-guides/using-secrets-in-github-actions

質問**: 11**
開発者として、GitHub Container Registry にイメージを公開するには、どのワークフロー手順を実行する必要がありますか? (それぞれの正解はソリューションの一部を表します。3 つ選択してください)。
**A.** イメージを GitHub Container Registry にプッシュします。
**B.** GitHub Container Registry に対して認証します。
**C.** actions/setup-docker アクションを使用します。
**D.** GitHub Container Registry からイメージをプルします。
**E.** コンテナイメージをビルドします。
正解**: (**正解を表示します**)**
Publishing Docker images
The below workflow checks out the GitHub repository, uses the login-action to log in to the registry [B], and then uses the build-push-action action to: build a Docker image based on your repository's Dockerfile [E]; push the image to Docker Hub [A], and apply a tag to the image.
Note:

# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.
# GitHub recommends pinning actions to a commit SHA.
# To get a newer version, you will need to update the SHA.
# You can also reference a tag or branch, but the action may change without warning.
Reference:
https://docs.github.com/en/actions/tutorials/publish-packages/publish-docker-images

質問**: 12**
開発者として、DevCloudとTestCloudの両方のリソースにデプロイするワークフローを作成しています。各クラウドリソースには異なるデプロイメントキーでアクセスします。異なるクラウドリソースをターゲットにするために、同じ再利用可能なワークフローを別々のジョブで使用できる最適なアプローチはどれですか？
**A.** DEPLOY_KEYリポジトリシークレットに、DevCloudプロパティとTestCloudプロパティを含むJSONオブジェクトを設定します。その後、再利用可能なワークフローのシークレットセクションでDEPLOY_KEY.DevCloudを指定します。
**B.** マーケットプレイス アクションを使用して、クラウド リソース名に基づいて DEPLOY_KEY リポジトリ シークレットを条件付きで解析します。
**C.** DevCloud環境とTestCloud環境のDEPLOY_KEY環境シークレットに、異なるキーを保存します。再利用可能なワークフローのシークレットセクションでDEPLOY_KEYを指定します。
**D.** 再利用可能なワークフローがリソース名でシークレットを解析できるように、DevCloud.DEPLOY_KEY および TestCloud.DEPLOY_KEY という名前のリポジトリ シークレットを作成します。
正解**: C (**コメントを発表する**)**
Using inputs and secrets in a reusable workflow
You can define inputs and secrets, which can be passed from the caller workflow and then used within the called workflow.
In the reusable workflow, use the inputs and secrets keywords to define inputs or secrets that will be passed from a caller workflow.
on:
workflow_call:
inputs:
config-path:
required: true
type: string
secrets:
personal_access_token:
required: true
Reference:
https://docs.github.com/en/enterprise-cloud@latest/actions/how-tos/reuse-automations/reuse- workflows

質問**: 13**
GitHub Actions では、ステップの成功または失敗は終了コードによってどのように決定されるのでしょうか?

**A.** 終了コードは無視され、メンテナーが手動で結果を設定します
**B.** 終了コードがゼロの場合は成功、ゼロ以外の場合は失敗を意味します。
**C.** 成功は終了コードではなくエラー時の継続によって制御されます
**D.** すべての終了コードは失敗です
正解: **(**正解を表示します**)**
GitHub Actions evaluates each step by the exit status of the process it runs. The runner marks a step successful when the command finishes with an exit status of 0. If the command returns any other code then the step is marked as failed and the job may stop depending on your workflow configuration.

質問: **14**
アクションに対してfooという名前の出力を宣言するための最小限の構文は何ですか？

**A.**
```
outputs:
  foo:
  value: Some value
```

**B.**
```
outputs:
  - key: foo
  value: Some value
```

**C.**
```
outputs:
  foo:
  description: Some value
```

**D.**
```
outputs:
  - key: foo
  description: Some value
```

正解: **(**正解を表示します**)**
To declare an output in a custom GitHub Action, you add an outputs block to your action.yml metadata file.
Here is the minimal yaml syntax:
outputs:
foo:
description: 'A description is required'
Use code with caution.
Note: The description field is mandatory in the metadata file, even if it is brief. To actually assign a value to this output during execution, you would then use the following command in your script:
echo "charlemagne=your_value" >> $GITHUB_OUTPUT
Reference:
https://oneuptime.com/blog/post/2026-01-30-github-actions-action-outputs/view

質問: **15**

開発者として、自動化の再利用に関する標準を実装するために、どのような2つの選択肢を推奨すべきでしょうか？正解はそれぞれ完全な解決策を示しています。注：正解つにつき1ポイントです。
**A.** 他のワークフローから呼び出すことができる、再利用可能なアクションとワークフローを作成します。
**B.** 会社向けに再利用可能な自動化を公開するためのマーケットプレイスパーティションを作成します。
**C.** 定義され文書化された内部アクセス可能なリポジトリ内のサブフォルダに、共有の企業行動を保存します。
**D.** ワークフローテンプレートを作成し、組織の.githubリポジトリに保存します。
正解**: A,C (**コメントを発表する**)**
[A]
Implementing standards for reusable actions and workflows is the most effective way to reduce duplication and maintain consistency across your GitHub organization.
Reusable Workflows (workflow_call)
Best for standardizing entire processes (e.g., a complete CI/CD pipeline).
Location: Defined in .github/workflows/ of a central repository.
Trigger: Use the on: workflow_call trigger to define inputs, secrets, and outputs.
Usage: Called from another workflow using the uses keyword (e.g., owner/repo/.github/workflows/ci.yml@v1).
Benefit: They allow you to enforce security scans, build steps, and deployment gates across multiple repositories from a single source
[C]
Implementing a marketplace partition for internal GitHub Actions is a key strategy for scaling automation securely and efficiently across a company. This centralized approach ensures that developers use vetted, high-quality automation while adhering to corporate standards.
Strategic Implementation of Internal Reuse
To implement standards for automation reuse effectively, focus on these core components:
[C] Internal Actions Marketplace: Create a dedicated organization or specific repositories to host and display shared actions. This "marketplace" acts as a curated directory of approved automations, preventing the use of unverified third-party actions from the public marketplace.
Workflow Templates: Store standardized workflow templates in your organization's .github repository. These templates appear in the "Actions" tab when a user creates a new workflow, ensuring consistency across different teams.
[A] Reusable Workflows: Design modular workflows that can be called by other repositories using the uses keyword. Unlike standard actions, these can manage entire jobs and runners, making them ideal for standardizing complex CI/CD pipelines.
Internal Repository Sharing: Set repository visibility to Internal and configure Actions permissions to allow access from other repositories in the organization or enterprise.
Reference:
https://www.incredibuild.com/blog/best-practices-to-create-reusable-workflows-on-github-actions
https://xebia.com/blog/setting-up-an-internal-github-actions-marketplace

**質問: 16**
ワークフローでデバッグ メッセージをどのように印刷すればよいですか?
**A.** echo "::debug::変数 myVariable を true に設定する"
**B.** echo "変数MyVariableをtrueに設定する" >> $DEBUG_MESSAGE

**C.** echo "::add-mask::変数 myVariable を true に設定する"
**D.** echo "debug_message=変数myVariableをtrueに設定する" >> &GITHUB_OUTPUT
正解**: (**正解を表示します**)**
Example: Setting a debug message
echo "::debug::Set the Octocat variable"
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-commands



質問**: 17**
ワークフロー内の次のイベントトリガーブロックの出力は何ですか？

```
on:
  issues:
    types: [opened, edited]
  issue_comment:
    types:
      - created
```



**A.** ワークフロー構文エラーが発生し、issue_comment イベントの型定義が問題となっています。
**B.** ワークフローのリポジトリ内の課題または課題コメントが作成または変更されたときにワークフローを実行します。
**C.** 課題が編集されたとき、または課題コメントが作成されたときにワークフローを実行します。
**D.** 課題が作成または編集されたとき、あるいは課題またはプルリクエストのコメントが作成されたときにワークフローを実行します。
**E.** ワークフロー構文エラーが発生し、issues イベントの型定義が問題となっています。
正解**: (**正解を表示します**)**
Based on your configuration, the workflow will trigger in these three specific scenarios:
Issue Opened: When a brand-new issue is created.
Issue Edited: When the title or body of an existing issue is changed.
Comment Created: When someone adds a new comment to an issue or pull request.
Reference:
https://forgejo.org/docs/next/user/actions/reference

質問**: 18**
MY_VARIABLE という名前のカスタム環境変数に値 my-value を指定するための適切な構文は次のどれですか。
**A.** 変数:

MY_VARIABLE = 私の値
**B.** 環境:
MY_VARIABLE = 私の値
**C.** 変数:
MY_VARIABLE: 私の値
**D.** 環境:
MY_VARIABLE = 私の値
**E.** 環境:
MY_VARIABLE: 私の値
**F.** 環境:
MY_VARIABLE: 私の値
正解**: F (**コメントを発表する**)**
To set a custom environment variable for a single workflow, you can define it using the env key in the workflow file.
Example:
env:
DAY_OF_WEEK: Monday
Note: The scope of a custom variable set by this method is limited to the element in which it is defined. You can define variables that are scoped for:
The entire workflow, by using env at the top level of the workflow file.
The contents of a job within a workflow, by using jobs.<job_id>.env.
A specific step within a job, by using jobs.<job_id>.steps[*].env.
Reference:
https://docs.github.com/en/actions/how-tos/write-workflows/choose-what-workflows-do/use- variables

質問**: 19**
どのデフォルトの GitHub 環境変数が所有者とリポジトリ名を示しますか?
**A.** GITHUB_REPOSITORY
**B.** GITHUB_WORKFLOW_REPO
**C.** ENV_REPOSITORY
**D.** リポジトリ名
正解**: A (**コメントを発表する**)**
Variables reference
Find information for supported variables, naming conventions, limits, and contexts in GitHub Actions workflows.
Variables include:
* GITHUB_REPOSITORY
The owner and repository name. For example, octocat/Hello-World
* Etc.
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/variables

質問**: 20**

特定のメジャーリリースバージョンをターゲットとする最も一般的な方法はどれですか?
**A.** 手順:
- 使用: actions/checkout@v3
**B.** 手順:
- 使用: アクション/チェックアウト
**C.** 手順:
- 使用: actions/checkout@U1673995124
**D.** 手順:
- 使用: actions/checkout@ac593985615ec2ede58e132d2e21d2b1cbd6127c
正解**: A (**コメントを発表する**)**
Example: Using versioned actions
steps:
# Reference a specific commit
- uses: actions/checkout@8f4b7f84864484a7bf31766abe9204da3cbe65b3
*-> # Reference the major version of a release
- uses: actions/checkout@v4
# Reference a specific version
- uses: actions/checkout@v4.2.0
# Reference a branch
- uses: actions/checkout@main
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-syntax

質問**: 21**
Docker コンテナ アクションと JavaScript アクションを正しく実行するために必要なリポジトリ ファイルはどれですか? (3 つ選択してください)
**A.** main.js や index.js などの JavaScript エントリ ファイル
**B.** package.json
**C.** アクションメタデータファイル action.yml または action.yaml
**D.** コンテナアクションDockerfile
正解**: A,C,D (**コメントを発表する**)**
The correct options are JavaScript entry file such as main.js or index.js, Action metadata file action.yml or action.yaml, and Container action Dockerfile.
The Action metadata file action.yml or action.yaml is mandatory because it declares inputs and outputs and it instructs GitHub how to run the action. Both JavaScript and container actions rely on this metadata to execute correctly.
For a JavaScript action the JavaScript entry file such as main.js or index.js is the script that the runner executes as directed by the metadata. GitHub expects a committed distributable script, so the entry file must be present in the repository.
For a container action the Container action Dockerfile defines how to build the image and what command to run, which makes the Dockerfile required for the action to work.

質問**: 22**

GitHub Actions では、単一のワークフロー実行内のすべてのジョブとステップでカスタム環境変数にアクセスできるようにするために、どこでカスタム環境変数を宣言する必要がありますか?
**A.** リポジトリ変数
**B.** ワークフローYAMLでenvを設定する
**C.** 環境シークレット
**D.** ランナーのコマンドライン引数
正解: **(**正解を表示します**)**
The correct option is Set env in the workflow YAML because defining environment variables at the top level of a workflow makes them available to every job and step in that single run.
In a workflow file you can declare a top level mapping for environment variables with key value pairs. This top level scope propagates the values to all jobs and steps in the run and you can override them at job or step scope when necessary. This is the most direct way to share non secret configuration across the entire workflow execution.

質問: **23**
開発者として、様々なファイルを使用 ·生成するGitHubワークフローを最適化しています。キャッシュとワークフローアーティファクトのどちらをいつ使用するかを判断する必要があります。正しい記述は2つありますか？(それぞれの正解は解答の一部です。2つ選択してください。)
**A.** アーティファクトを使用して GitHub パッケージ レジストリにアクセスし、ワークフローのパッケージをダウンロードします。
**B.** ジョブまたはワークフロー実行間でほとんど変更されないファイルを再利用する場合は、キャッシュを使用します。
**C.** キャッシュを使用して、アクセス間のキャッシュ エントリを最大 30 日間保存します。
**D.** ワークフローの終了後にジョブによって生成されたファイルを参照するときに、アーティファクトを使用します。
正解: **(**正解を表示します**)**
[B] Use caching when you want to reuse files that don't change often between jobs or workflow runs, such as build dependencies from a package management system.
[D] Use artifacts when you want to save files produced by a job to view after a workflow run has ended, such as built binaries or build logs.
Reference:
https://docs.github.com/en/enterprise-server@3.16/actions/tutorials/store-and-share-data

質問: **24**
カスタムアクションを配布する際にドキュメントを追加することの最も重要な利点は何ですか? (各回答は完全なソリューションを示します。2 つ選択してください。)
**A.** 消費ワークフローの readme.md を作成します。
**B.** ワークフローでアクションを使用するときに自動補完を生成します。
**C.** アクションの説明をユーザーに共有します。
**D.** アクションの例を示します。
正解: **(**正解を表示します**)**

質問: **25**
ワークフローを開始した人またはアプリの名前を示すデフォルトの GitHub 環境変数はどれですか。

**A.** GITHUB_ACTOR
**B.** GitHubユーザー
**C.** ENV_ACTOR
**D.** GITHUB_WORKFLOW_ACTOR
正解**: A (**コメントを発表する**)**
GITHUB_ACTOR : The name of the person or app that initiated the workflow.
Reference:
https://graphite.dev/guides/github-actions-variables
GITHUB_ACTOR : The name of the person or app that initiated the workflow.

質問**: 26**
GitHub Actions ワークフロー ファイルでワークフロー名を正しく設定する構成行はどれですか。
(2つ選択)
**A.** 名前: ${{ "release-flow-42" }}
**B.** 名前: ${{ 'release-flow-42' }}
**C.** 実行名: リリースフロー42
**D.** 名前: リリースフロー−42
正解**: B,D (**コメントを発表する**)**
The correct options are name: release-flow-42 and name: ${{ 'release-flow-42' }}. Both set the workflow name that appears in the Actions interface and on the workflow page.
The plain static name key with a simple string sets the workflow name clearly and predictably. It is the most straightforward way to define the workflow title.
The expression form with a single quoted string evaluates to the same literal value. GitHub Actions accepts a string result for the workflow name, so this produces a valid name while keeping consistency with other places where expressions are used.

質問**: 27**
開発者がワークフロー内で GITHUB_TOKEN または github.token シークレットを明示的に使用する必要があるシナリオは次のうちどれですか (2 つ選択してください)。
**A.** 入力としてトークンを必要とするアクションに GITHUB_TOKEN シークレットを渡す
**B.** 認証されたGitHub APIリクエストを行う
**C.** actions/checkout@v3 アクションでソースコードをチェックアウトする
**D.** GITHUB_TOKENにデフォルト以外の権限を割り当てる
正解**: (**正解を表示します**)**
[A] Some actions may require a GITHUB_TOKEN as an input to authenticate and perform specific tasks, such as creating issues, commenting on pull requests, or interacting with the GitHub API. In such cases, you would need to explicitly pass the token to the action.
[B] When making an authenticated GitHub API request, the GITHUB_TOKEN is required to authenticate the request. This token is automatically provided by GitHub in the workflow, and it must be explicitly used when interacting with the GitHub API.
Note: You can use the GITHUB_TOKEN to make authenticated API calls. This example workflow creates an issue using the GitHub REST API:

Reference:
https://docs.github.com/en/actions/tutorials/authenticate-with-github_token

質問: **28**
開発者として、GitHub Actions ワークフローを、Checks API を使用するサードパーティのコード品質プロバイダーと統合する必要があります。フォローアップワークフローをどのようにトリガーすればよいでしょうか？
**A.** コード品質統合名のワークフローのトリガーとして workflow_run ウェブフックイベントを追加します。
**B.** コード品質の統合が完了したときに、ワークフローのトリガーとして check_run Webhook イベントを追加します。
**C.** コード品質統合が同期されたときに、ワークフローのトリガーとして pull_request webhook イベントを追加します。
**D.** コード品質の統合が完了したときに、ワークフローのトリガーとしてデプロイメントWebhookイベントを追加します。
正解: **(**正解を表示します**)**
The check_run event is triggered when a check (such as a code quality check) completes, including when the status of a check changes. By adding this event as a trigger, you can initiate a follow-up workflow when the code quality integration finishes its checks.
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/events-that-trigger-workflows

質問: **29**
ワークフロー成果物をダウンロードするために必要な最低のリポジトリ権限は次のどれですか?
**A.** トリアージ
**B.** 管理者
**C.** 維持する
**D.** 書く
**E.** 読む
正解: **E (**コメントを発表する**)**
Downloading workflow artifacts
You can download archived artifacts before they automatically expire.
Who can use this feature?
People who are signed into GitHub and have read access to a repository can download workflow artifacts.
Reference:
https://docs.github.com/en/actions/how-tos/manage-workflow-runs/download-workflow-artifacts

質問: **30**
設定可能な環境保護ルールには、どのような2種類がありますか？正解はそれぞれ完全な解答を示しています。注：正解つにつき1ポイントです。
**A.** 遺物保管庫
**B.** 待機タイマー
**C.** 分岐保護
**D.** 必須レビュー担当者
正解: **B,D (**コメントを発表する**)**

In GitHub Actions, you can configure four primary types of environment protection rules to control how and when deployments proceed to a specific environment.
These rules include:
[D] Required Reviewers: Specify up to six people or teams who must approve a deployment before it can proceed. Only one reviewer from the list needs to approve to unlock the deployment.
[B] Wait Timer: Set a mandatory delay (from 0 to 43,200 minutes) that must pass after a job is triggered before it can run.
Deployment Branches and Tags: Restrict deployments to only occur from specific branches or those matching certain tag patterns (e.g., main, release/*, or v*).
Custom Deployment Protection Rules: Enable third-party systems or automated tools (via GitHub Apps) to gate deployments based on external data, such as security scan results or ticket status.
Reference:
https://oneuptime.com/blog/post/2026-01-25-github-actions-environment-protection-rules/view

質問**: 31**
次のワークフローファイルでは、5行目は3行目と4行目をPythonとして解釈します。5行目を完了するための有効なオプションは次のどれですか？
1 ステップ:
2 - 実行: |
3 インポート OS
4 print(os.environ['PATH'])
5
**A.** シェル: Python
**B.** 作業ディレクトリ: .github/python
**C.** シェル: bash
**D.** Pythonで
正解**: A (**コメントを発表する**)**
Workflow syntax for GitHub Actions
Example: Running an inline Python script
steps:
- name: Display the path
shell: python
run: |
import os
print(os.environ['PATH'])
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-syntax

質問: **32**
組織のポリシーでは、ローカルアクションのみが許可されると指定されています。この組織では、アクションをどのように配布すればよいでしょうか？
**A.** サードパーティが所有するリポジトリ経由
**B.** 組織が所有するリポジトリ経由
**C.** GitHub Marketplace経由
**D.** 組織が所有する.githubリポジトリ経由
正解: **(正解を表示します)**
Local GitHub Actions refer to two concepts: custom actions developed within your own repository and the process of running any GitHub Action locally on your machine for faster testing and debugging.
1. Local Actions (within your repository)
Definition:
These are custom actions that you define and store directly in your repository under the .github/actions/ directory. They are not publicly shared, unlike actions from the GitHub Marketplace.
Purpose:
They are used for project-specific, repetitive tasks that you want to reuse within your workflows without publishing them globally.
How it works:
You can call these local actions in your workflow files as if they were standard published actions.
2. Running Actions Locally (on your development machine)
Note:
Pinned repositories
You can give users easy access to important or frequently used repositories, by choosing up to six repositories for public users and six repositories for members of the organization. Once you pin repositories to your organization profile, the "Pinned" section is shown above the
"Repositories" section of the profile page.
Reference:
https://dev.to/codenamegrant/supercharging-your-workflows-with-local-github-actions-2o23
https://docs.github.com/en/organizations/collaborating-with-groups-in-organizations/customizing- your-organizations-profile

質問: **33**
TypeScript で記述されたカスタムアクションに最も適したアクションタイプは何ですか?
**A.** Dockerコンテナアクション
**B.** Bashスクリプトアクション
**C.** 複合実行ステップ
**D.** JavaScriptアクション
正解: **(正解を表示します)**
Example: Build your custom GitHub Action

To build your action, run npm run bundle. This will compile the TypeScript code in the src/ directory and output the JavaScript code in the dist/ directory. The exact command for bundle is defined in the package.json file.
Note: About custom actions
Actions are individual tasks that you can combine to create jobs and customize your workflow. Y Types of action You can build Docker container, JavaScript, and composite actions.
Reference:
https://victoronsoftware.com/posts/typescript-github-action/
https://docs.github.com/en/actions/concepts/workflows-and-actions/custom-actions

質問**: 34**
企業環境でカスタム アクションを作成および管理する場合、ベスト プラクティスと考えられるのは次のどれですか。
**A.** 各アクションごとに別々のリポジトリを作成し、バージョンを個別に管理できるようにする
**B.** アプリケーションリポジトリにアクションのみを含む別のブランチを作成する
**C.** すべてのカスタムアクションに対して単一のリポジトリを作成し、各アクションのバージョンがすべて同じになるようにします。
**D.** 他のチームがアプリケーションコードと同じリポジトリで参照する必要があるカスタムアクションを含む
正解**: (**正解を表示します**)**
Managing custom actions
Choosing a location for your action
If you're developing an action for other people to use, we recommend keeping the action in its own repository instead of bundling it with other application code. This allows you to version, track, and release the action just like any other software.
Creating a separate repository for each custom action allows you to manage the versioning independently for each action. This approach provides flexibility, as each action can be updated, tested, and versioned separately, avoiding potential conflicts or dependencies between different actions.
Reference:
https://docs.github.com/en/actions/how-tos/create-and-publish-actions/manage-custom-actions

質問**: 35**
GitHub Actions のどの機能が、ワークフロー ログにシークレットが表示されないようにするのでしょうか。
**A.** 秘密ログ出力の手動承認
**B.** GitHub監査ログ
**C.** ログ内のデフォルトの秘密マスキング
**D.** ログに書き込まれた暗号化された秘密
正解**: (**正解を表示します**)**
GitHub Actions automatically redacts secret values from workflow logs so if a command prints a secret the value is masked in the output. This built in protection helps prevent accidental exposure of credentials during job execution and it applies without additional configuration.

質問**: 36**
開発者として、GitHub の 「アクション」タブからワークフローを実行したいとします。この画像に示されているインターフェースに合わせるには、どの YAML スニペットを使用すればよいでしょうか？

**Use workflow from**

Branch: main

**Test suite**

functional

Run workflow



**A.**

```
on:
  workflow_run:
    inputs:
      test_suite:
        description: Test suite
        type: string
        options:
          - functional
          - regression
```

B.

```
...
 workflow_run:
   inputs:
     test_suite:
       description: Test suite
       type: choice
       options:
         - functional
         - regression
```

C.

```
...
 workflow_dispatch:
   inputs:
     test_suite:
       description: Test suite
       type: choice
       options:
         - functional
         - regression
```

D.

```
on:
  workflow_dispatch:
    inputs:
      test_suite:
        description: Test suite
        type: choice
        value: functional
        options:
          - regression
```

**正解: (正解を表示します)**

Use workflow_dispatch as it used for manual trigger source (UI button, CLI, or API).

Use B as it has the option functional.

Incorrect:

[Not D]

D has only the single option regression, it does not have the option functional which is displayed in the graphic.

[Not A, Not C]

Must use workflow_dispatch not workflow_run.

Note:
In GitHub Actions, the primary difference is how the workflow is triggered: workflow_dispatch is for manual execution by a person, while workflow_run is for automatic execution triggered by another workflow's activity.
Reference:
https://innosufiyan.hashnode.dev/understanding-workflow-dispatch-in-github-actions-a-beginners- guide

質問**: 37**
開発者として、Actions ワークフローでは、実行ごとに同じ出力やダウンロードした依存関係を再利用することがよくあります。ジョブの依存関係をキャッシュするには、GitHub キャッシュアクションを使用します。このアクションに必要な入力パラメータはどれですか？(正解はそれぞれ解答の一部です。2つ選択してください。)
**A.** パス: ランナー上のキャッシュまたは復元するファイルパス
**B.** 依存関係: キャッシュまたは復元するパッケージの名前とバージョン
**C.** key: キャッシュを保存するときに作成されるキーと、キャッシュを検索するために使用されるキー
**D.** restore-keys: データをキャッシュするためにキャッシュパラメータで使用されるコピーアクションキー
**E.** キャッシュヒット: キャッシュからデータを復元するために復元パラメータで使用されるコピーアクションキー
**F.** ref: 作成されたキャッシュにアクセスして復元するブランチのref名
正解**: (**正解を表示します**)**
Input parameters for the cache action
[C] key: Required The key created when saving a cache and the key used to search for a cache.
It can be any combination of variables, context values, static strings, and functions. Keys have a maximum length of 512 characters, and keys longer than the maximum length will cause the action to fail.
[A] path: Required The path(s) on the runner to cache or restore.
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/dependency-caching

質問**: 38**
開発者として、ワークフローの1つでmacOS Catalina(つまりv10.15)でホストされるXCodeバージョン11.2が必要になります。これらの要件を満たすセルフホストランナーを既に作成 設定し、組織に登録済みです。ワークフローが正しいランナーインスタンスにアクセスできるようにするには、他にどのような対策が必要ですか？(各回答に完全な解決策が記載されています。3つ選択してください。)
**A.** ランナーを適切なランナー グループに追加します。
**B.** ワークフローで以下を指定します。
動作環境: [ ${{groups.macos-10.15}}, ${{groups.xcode-11.2}} ]。
**C.** macos-10.15 および xcode-11.2 用のカスタム ランナー ラベルを作成します。
**D.** macos-10.15 および xcode-11.2 という名前のランナー グループを作成します。
**E.** ワークフローで以下を指定します。
動作環境: [セルフホスト、macOS-10.15、Xcode-11.2]。
**F.** セルフホストランナーにカスタムラベルを割り当てます。
正解**: (**正解を表示します**)**
[C, F] Using custom labels to route jobs

You can create custom labels and assign them to your self-hosted runners at any time. Custom labels let you send jobs to particular types of self-hosted runners, based on how they're labeled.
[E] Using default labels to route jobs
A self-hosted runner automatically receives certain labels when it is added to GitHub Actions.
These are used to indicate its operating system and hardware platform:
self-hosted: Default label applied to self-hosted runners.
linux, windows, or macOS: Applied depending on operating system.
x64, ARM, or ARM64: Applied depending on hardware architecture.
You can use your workflow's YAML to send jobs to a combination of these labels. In this example, a self-hosted runner that matches all three labels will be eligible to run the job:
runs-on: [self-hosted, linux, ARM64]
self-hosted - Run this job on a self-hosted runner.
linux - Only use a Linux-based runner.
ARM64 - Only use a runner based on ARM64 hardware.
Reference:
https://docs.github.com/en/actions/how-tos/manage-runners/self-hosted-runners/use-in-a- workflow

質問**: 39**
カスタムアクションを作成する際に、ランナー内の動作の一部を変更する必要があります。ランナーの出力にエラーメッセージを設定するワークフローコマンドはどれですか？(正解はそれぞれ完全な解答を示しています。2つ選択してください。)
**A.** echo "::error file=main.py,line=10,col=15::エラーが発生しました"
**B.** echo "::error=エラーが発生しました::"
**C.** echo "::error::エラーが発生しました"
**D.** echo "::error message=エラーが発生しました::"
正解**: (**正解を表示します**)**
To set an error message in a GitHub Actions runner's output, you should use the command echo "::error::Your error message" in your workflow's run step. This workflow command creates an error annotation that will appear in the logs, and you can optionally include file and line number information, such as echo "::error file=app.js,line=5::Error message".
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-commands

質問**: 40**
一連の実行ステップを再利用可能なカスタム アクションにまとめるには、どのアクション タイプを使用する必要がありますか?
**A.** 複合アクション
**B.** Bashスクリプトアクション
**C.** Dockerコンテナアクション
**D.** JavaScriptアクション
正解**: (**正解を表示します**)**
Reusable workflows versus composite actions
Reusable workflows and composite actions both help you avoid duplicating workflow content.

Whereas reusable workflows allow you to reuse an entire workflow, with multiple jobs and steps, composite actions combine multiple steps that you can then run within a job step, just like any other action.
A composite action allows you to bundle multiple steps into a single reusable action within a workflow. It is composed of multiple run steps or other actions and can be reused across workflows, making it the perfect choice for bundling a series of steps.
Reference:
https://docs.github.com/en/actions/concepts/workflows-and-actions/reusable-workflows

**質問: 41**
どのYAMLスニペットが、プッシュリクエストとプルリクエストの両方のイベントに対してGitHub Actionsワークフローをトリガーしますか？

**A.**
```
on:
  push:
    branches:
      - dev
  pull_request:
    branches:
      - main
```

**B.**
```
on:
  push:
    branches:
      - main
```

**C.**
```
on:
  push:
    branches:
      - main
  push:
    branches:
      - dev
```

**D.**
```
on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
```

正解: (正解を表示します)

To trigger a GitHub Actions workflow on both push and pull request events, use the on key with a list of the events:
on:
push:
branches: [ "main" ]
pull_request:
branches: [ "main" ]
Key Details:
Events: You can list multiple events under on. This setup ensures the workflow runs whenever code is pushed to main or when a PR is opened/updated against main.
Activity Types: By default, pull_request triggers on opened, reopened, and synchronize (new commits to the PR).
Filtering: You can further refine these triggers by adding specific paths, tags, or different branches to avoid unnecessary runs Incorrect:
[Not A]
The push event is for the dev branch only.
[Not B]
Has only the push event.
[Not C]
Has only the push event.
Reference:
https://blog.devops.dev/automate-your-workflow-a-guide-to-ci-cd-with-github-actions-3f395d60ba69

質問: **42**
ワークフローで GITHUB_TOKEN を使用するのはいつですか?
**A.** GitHub APIへの認証された呼び出しを行う場合
**B.** SSH経由でランナーに接続する場合
**C.** リポジトリレベルのシークレットにアクセスする場合
**D.** マーケットプレイスからアクションを呼び出す場合
正解: **(正解を表示します)**
Use GITHUB_TOKEN for authentication in workflows
At the start of each workflow job, GitHub automatically creates a unique GITHUB_TOKEN secret to use in your workflow. You can use the GITHUB_TOKEN to authenticate in the workflow job.
Reference:
https://docs.github.com/en/actions/tutorials/authenticate-with-github_token

質問: **43**
DevOpsエンジニアとして、アプリケーションをビルドするためのワークフローを開発しています。複数のノードバージョンをターゲットとするビルドを作成する必要があります。ワークフローを定義するには、どのコードブロックを使用すればよいでしょうか？

A.
```
jobs:
  build-app:
    strategy:
      matrix:
        node-ver: [10, 12, 14]
    steps:
      - uses: actions/setup-node@v3
        with:
          node-version: ${{ matrix.node-ver }}
```

B.
```
jobs:
  build-app:
    matrix-strategy:
      node-ver: [10, 12, 14]
    steps:
      - uses: actions/setup-node@v3
        with:
          node-version: ${{ matrix-strategy.node-ver }}
```

C.
```
jobs:
  build-app:
    matrix:
      strategy:
        node-ver: [10, 12, 14]
    steps:
      - uses: actions/setup-node@v3
        with:
          node-version: ${{ matrix.node-ver }}
```

D.
```
jobs:
  build-app:
    strategy:
      matrix:
        node-ver: [10, 12, 14]
    steps:
      - uses: actions/setup-node@v3
        with:
          node-version: ${{ strategy.node-ver }}
```

正解: (正解を表示します)

Use keywords strategy and matrix in that order.

Last line should be node-version: ${{ matrix.node-ver}}

Reference:

https://docs.github.com/en/actions/how-tos/write-workflows/choose-what-workflows-do/run-job- variations

質問: **44**
GitHub Enterprise Cloud では、エンタープライズでホストされるアクションと再利用可能なワークフローのみの実行を許可するポリシーはどこで設定しますか?
**A.** 組織設定ポリシー、アクション
**B.** エンタープライズアカウントのアクションポリシー
**C.** リポジトリ設定アクション一般
正解: **(**正解を表示します**)**
The enterprise Actions policy is the centralized place where enterprise owners enforce which actions and reusable workflows are permitted across all organizations in GitHub Enterprise Cloud. From this scope you can require that only actions and reusable workflows hosted within the enterprise are allowed to run, and this policy flows down to organizations and repositories to ensure consistent enforcement.

質問: **45**
カスタムアクション内のどのメタデータファイルがメインのエントリポイントを定義しますか？
**A.** main.yml
**B.** action.yml
**C.** action.js
**D.** index.js
正解: **(**正解を表示します**)**
The metadata file that defines the main entry point for a custom GitHub Action is action.yml (or action.yaml).
This file, located in the root directory of your action's repository, uses YAML syntax and includes a runs key that specifies how the action is executed and which script is the main entry point.
Key fields in the action.yml file
name: The name of your action displayed in the GitHub Actions UI.
description: A short description of the action.
runs: This required key specifies the execution environment and the main entry point.
For a JavaScript action, the runs field includes using (e.g., node20) and main (e.g., index.js) to define the runtime and the primary script file.
For a Docker container action, the runs field includes using: 'docker' and image (e.g., 'Dockerfile') to point to the Docker build file or image.
inputs: Optional parameters that allow users to pass data to your action.
outputs: Optional output parameters that the action can produce for use by subsequent steps in a workflow.
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/metadata-syntax

質問: **46**
ワークフロー内の依存ジョブから上流ジョブ (job1) のジョブ出力 (output1) に正しくアクセスする構文はどれですか。
**A.** ${{needs.job1.outputs.output1}}
**B.** ${{needs.job1.output1}}

**C.** ${{depends.job1.output1}}
**D.** ${{job1.outputs.output1}}
正解**: (**正解を表示します**)**
To access the outputs in the dependent job, use the needs.<job_id>.outputs.<output_name> syntax. For example, the following job accesses the output1 and output2 outputs defined in job1:
jobs:
# Assume job1 is defined as above
job2:
runs-on: ubuntu-latest
needs: job1
steps:
- env:
OUTPUT1: ${{needs.job1.outputs.output1}}
OUTPUT2: ${{needs.job1.outputs.output2}}
run: echo "$OUTPUT1 $OUTPUT2"
Reference:
https://docs.github.com/en/actions/how-tos/write-workflows/choose-what-workflows-do/pass-job- outputs



質問**: 47**
次のステップに進む前にユーザーがワークフローを承認していることを確認する適切な方法は何ですか?
**A.** ブランチ保護ルールを作成し、特定のユーザーのみにアクセスを許可する
**B.** ユーザーにワークフロー承認権限を付与する
**C.** 環境の必須レビュー担当者としてユーザーを追加する
**D.** ユーザーにリポジトリ承認権限を付与する
正解**: (**正解を表示します**)**
GitHub Actions allows you to configure environment protection rules, where you can require specific users or teams to approve the deployment before the workflow proceeds to the next step.
This ensures that the required reviewers approve the workflow before any sensitive actions (such as deployment) occur.
Note: Managing environments for deployment
You can create environments and secure those environments with deployment protection rules. A job that references an environment must follow any protection rules for the environment before running or accessing the environment's secrets.
Optionally, you can specify people or teams that must approve workflow jobs that use this environment.
Select Required reviewers.
Enter up to 6 people or teams. Only one of the required reviewers needs to approve the job for it to proceed.

Optionally, to prevent users from approving workflows runs that they triggered, select Prevent self-review.
Click Save protection rules.
Reference:
https://docs.github.com/en/actions/how-tos/deploy/configure-and-manage-deployments/manage- environments

質問**: 48**
GitHubのアーティファクトとパッケージの保存容量が組織の制限に達しようとしています。保存容量の制限に達しないようにするには、どうすればよいですか？(正解はそれぞれ完全な解決策を示しています。2つ選択してください。)
**A.** すべてのワークフロー実行に自己ホスト型ランナーを使用します。
**B.** リポジトリからアーティファクトを手動で削除します。
**C.** アーティファクトとログの保持期間を設定します。
**D.** リポジトリのブランチ保護を無効にします。
**E.** Git Large File Storage を使用するようにリポジトリを設定します。
正解**: (**正解を表示します**)**
[E] To bypass GitHub's limit of 100 MB you can use Git Large File Storage (Git LFS). It stores references to your file in the repo by creating a pointer file. However, the actual file is going to be stored at a different location.
[C] Configuring the retention period for GitHub Actions artifacts and logs in your organization You can configure the retention period for GitHub Actions artifacts and logs in your organization.
By default, the artifacts and log files generated by workflows are retained for 90 days before they are automatically deleted. You can adjust the retention period, depending on the type of repository:
For public repositories: you can change this retention period to anywhere between 1 day or 90 days.
For private repositories: you can change this retention period to anywhere between 1 day or 400 days.
When you customize the retention period, it only applies to new artifacts and log files, and does not retroactively apply to existing objects. For managed repositories and organizations, the maximum retention period cannot exceed the limit set by the managing organization or enterprise.
Reference:
https://gitprotect.io/blog/github-storage-limits/
https://docs.github.com/en/organizations/managing-organization-settings/configuring-the-retention-period-for-github-actions-artifacts-and-logs-in-your-organization

質問**: 49**
Linuxのセルフホストランナーに特定のソフトウェアをインストールしました。ワークフローを実行するユーザーがいて、特定されたカスタムソフトウェアに基づいてランナーを選択できるようにする必要があります。ランナーとユーザーがこれらのワークフローを実行できるようにするために、どのような手順を実行する必要がありますか？(正解はそれぞれ解答の一部です。2つ選択してください。)
**A.** GitHub がワークフローをトリガーできるように、Webhook とネットワークを構成します。
**B.** custom-software-on-linux グループを作成し、ランナーをそのグループに移動します。
**C.** ユーザーに、ラベル custom-software および linux を使用してランナーを識別するように通知します。
**D.** ランナーにラベル linux を追加します。
**E.** グループに基づいてランナーを識別するようにユーザーに通知します。
**F.** ランナーにラベル custom-software を追加します。

正解: **(正解を表示します)**
[F] Add a label for the custom software as required.
Note: Using labels with self-hosted runners
You can use labels to organize your self-hosted runners based on their characteristics.
Creating a custom label
You can create custom labels for runners at the repository and organization levels.
[C, not D] Linux is a default label and does not have to be created to be used.
Reference:
https://docs.github.com/en/actions/how-tos/manage-runners/self-hosted-runners/apply-labels

質問: **50**
どのワークフロー コマンドがランナーから情報を送信しますか? (それぞれの正解は完全なソリューションを示します。2 つ選択してください。)
**A.** 環境変数からの読み取り
**B.** デバッグメッセージの設定
**C.** Dockerfile内の変数を設定する
**D.** 出力パラメータの設定
正解: **(正解を表示します)**
[B] Setting a debug message
Prints a debug message to the log. You must create a secret named ACTIONS_STEP_DEBUG with the value true to see the debug messages set by this command in the log.
::debug::{message}
Example: Setting a debug message
echo "::debug::Set the Octocat variable"
[D] Setting an output parameter
Sets a step's output parameter. Note that the step will need an id to be defined to later retrieve the output value.
echo "{name}={value}" >> "$GITHUB_OUTPUT"
Example of setting an output parameter
This example demonstrates how to set the SELECTED_COLOR output parameter and later retrieve it:
- name: Set color
id: color-selector
run: echo "SELECTED_COLOR=green" >> "$GITHUB_OUTPUT"
- name: Get color
env:
SELECTED_COLOR: ${{ steps.color-selector.outputs.SELECTED_COLOR }}
run: echo "The selected color is $SELECTED_COLOR"
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-commands

質問: **51**
ワークフローの実行を手動でトリガーするために使用されるワークフロー イベントはどれですか。

**A.** ステータス
**B.** ワークフローディスパッチ
**C.** workflow_run
**D.** 作成
正解: **(**正解を表示します**)**
Manually running a workflow
When a workflow is configured to run on the workflow_dispatch event, you can run the workflow using the Actions tab on GitHub, GitHub CLI, or the REST API.
Configuring a workflow to run manually
To run a workflow manually, the workflow must be configured to run on the workflow_dispatch event.
To trigger the workflow_dispatch event, your workflow must be in the default branch.
Reference:
https://docs.github.com/en/actions/how-tos/manage-workflow-runs/manually-run-a-workflow

質問: **52**
環境変数の最小スコープは何ですか?
**A.** ワークフロー設定
**B.** ワークフローenvマッピング
**C.** ジョブ
**D.** ステップ
正解: **(**正解を表示します**)**
In GitHub Actions, environment variables can be defined at three different scopes: workflow, job, and step. Each scope determines the visibility and lifetime of the environment variable across the workflow execution.
Variables defined at the workflow scope are available to all jobs and steps within the workflow.
Variables defined at the job scope are available to all steps within a specific job.
Variables defined at the step scope are only available to the step in which they are defined. They are declared within a step and are not visible outside of that step.
Reference:
https://medium.com/@leroyleowdev/github-actions-ways-to-share-data-between-jobs-4267ff9ac2ce

質問: **53**
GitHub Actionsワークフローの実行を停止する必要があります。ただし、ワークフローの設定と実行履歴は保持する必要があります。どうすればよいでしょうか？
**A.** リポジトリからワークフローのYAMLファイルを削除します。
**B.** アクションを選択してワークフローを無効にします。
**C.** ワークフロー YAML ファイルの名前を変更します。
**D.** ワークフローを新しいブランチにアーカイブします。
正解: **(**正解を表示します**)**

To stop a GitHub Actions workflow while keeping its configuration file and execution history, you should disable the workflow rather than deleting it. Disabling a workflow prevents it from being triggered by any event (like pushes or schedules) but leaves the YAML file in your repository and preserves all previous run data on the GitHub Actions tab.
How to Disable a Workflow
You can disable a workflow through the GitHub UI, the CLI, or by modifying the code itself:
Via GitHub Web Interface:
Go to your repository on GitHub.
Click the Actions tab.
In the left sidebar, select the specific workflow you want to stop.
Click the three dots (...) or the Enable/Disable dropdown menu and select Disable workflow.
Reference:
https://docs.github.com/actions/managing-workflow-runs/disabling-and-enabling-a-workflow

質問**: 54**
ワークフロー実行を削除する機能に関して正しい記述はどれですか?
**A.** ワークフロー実行を削除するには管理者アクセスが必要です。
**B.** 保留中のワークフロー実行は削除できます。
**C.** 完了したワークフロー実行は削除される場合があります。
**D.** ワークフロー実行を削除するには、30 日以上経過している必要があります。
正解**: (**正解を表示します**)**
Deleting a workflow run
You can delete a workflow run that has been completed, or is more than two weeks old.
Reference:
https://docs.github.com/en/enterprise-cloud@latest/actions/how-tos/manage-workflow- runs/delete-a-workflow-run

質問**: 55**
次のコマンドのうち、スクリプト内で $FOO 環境変数を設定し、後続のワークフロー ジョブ ステップで使用できるようにするものはどれですか。
**A.** 実行: echo "::set-env name=FOO::bar"
**B.** 実行: echo "FOO=bar" >> $GITHUB_ENV
**C.** 実行: echo ${{ $FOO=bar }}
**D.** 実行: エクスポート FOO=bar
正解**: (**正解を表示します**)**
The $GITHUB_ENV environment variable is used to set environment variables that persist across steps in a workflow job. By echoing FOO=bar into $GITHUB_ENV, the variable FOO will be available in subsequent steps within the same job.
Variables set in GITHUB_ENV apply only to the current job.
Example:
echo "PR_NUMBER=$pr_number" >> $GITHUB_ENV
Reference:
https://github.com/orgs/community/discussions/56849

質問: **56**
特定のリポジトリ内のワークフローから単一のシークレットにアクセスする必要があります。シークレットを作成する最適な方法は何ですか？
**A.** 組織レベルで環境シークレットを作成し、指定された各リポジトリでその環境を活用します。
**B.** 組織シークレットを作成し、リポジトリ アクセスとして [選択されたリポジトリ] を指定して、必要なリポジトリを選択します。
**C.** いずれかのリポジトリにシークレットを作成し、シークレットの共有オプションをオンにして、必要なリポジトリを選択します。
**D.** シークレットをサポートされている外部キー コンテナーに保存します。OpenID Connect (OIDC) を構成して外部キー コンテナーへのアクセスを許可し、各リポジトリの外部キー コンテナーからシークレットをリンクします。
正解: **B (**コメントを発表する**)**
Creating secrets for an organization
When creating a secret or variable in an organization, you can use a policy to limit access by repository. For example, you can grant access to all repositories, or limit access to only private repositories or a specified list of repositories.
To specify that the secret should be available to selected repositories within the organization, use the --repos or -r flag.
gh secret set --org ORG_NAME SECRET_NAME --repos REPO-NAME-1, REPO-NAME-2 Note: REST API endpoints for GitHub Actions Secrets Use the REST API to interact with secrets in GitHub Actions.
* Set selected repositories for an organization secret
Replaces all repositories for an organization secret when the visibility for repository access is set to selected. The visibility is set when you Create or update an organization secret.
Request example
Put /orgs/{org}/actions/secrets/{secret_name}/repositories
Reference:
https://docs.github.com/en/actions/how-tos/write-workflows/choose-what-workflows-do/use- secrets
https://docs.github.com/en/rest/actions/secrets?apiVersion=2022-11-28#set-selected- repositories-for-an-organization-secret

質問: **57**
DevOpsエンジニアとして、プルリクエストに追加されたラベルに基づいて、開発環境やテスト環境など、異なる環境へのデプロイを実行する必要があります。デプロイはリリースブランチを使用し、「apps」フォルダ内のファイルに変更があった場合にのみトリガーする必要があります。デプロイワークフローのトリガーを定義するには、どのコードブロックを使用すればよいでしょうか？
**A.** オン:
プルリクエストラベル:
支店:
- 『リリース」
パス:
- 'アプリ/**'
**B.** オン:
プルリクエスト:
タイプ: [ラベル付き]
支店:
- 'リリース/**'

パス:
- 『アプリ」
**C.** オン:
プルリクエスト:
タイプ: [ラベル付き]
支店:
- 『リリース」
パス:
- 'アプリ/**'
**D.** オン:
プルリクエストレビュー:
タイプ: [ラベル付き]
支店:
- 『リリース」
パス:
- 'アプリ/**'
正解**: (**正解を表示します**)**
Incorrect:
[Not A] pull_request activitiy type labeled not specified.
pull_request_label is not a trigger.
[Not B] Specifies branches branches that has a name that starts with releases. We are only interested in the release branch.
branches:
- 'releases/**'
[Not C]
pull_request_review
Runs your workflow when a pull request review is submitted, edited, or dismissed. A pull request review is a group of pull request review comments in addition to a body comment and a state.
Reference:
https://docs.github.com/en/actions/reference/workflows-and-actions/events-that-trigger-workflows

質問**: 58**
開発者として、GitHub 上の複合アクションをどのように識別できますか?
**A.** アクションのリポジトリ名にキーワード 『composite」が含まれています。
**B.** アクションのリポジトリには、ルート ディレクトリに init.sh ファイルが含まれています。
**C.** アクションのリポジトリには、Dockerfile と package.json ファイルが含まれます。
**D.** action.yml メタデータ ファイルでは、runs.using 値が composite に設定されています。
正解**: (**正解を表示します**)**
runs.using for composite actions
Required: You must set this value to 'composite'.
Reference:

https://docs.github.com/en/actions/reference/workflows-and-actions/metadata-syntax

質問: **59**
カスタム環境変数は、ワークフロー ファイル内の複数のレベルで定義できます。
(それぞれの回答は完全な解決策を示しています。3 つ選択してください。)
**A.** 最上位レベル。
**B.** ステップレベル。
**C.** デフォルトレベル。
**D.** ランナーレベル。
**E.** ジョブレベル。
**F.** ステージレベル。
正解: **(正解を表示します)**
Defining environment variables for a single workflow
To set a custom environment variable for a single workflow, you can define it using the env key in the workflow file. The scope of a custom variable set by this method is limited to the element in which it is defined. You can define variables that are scoped for:
The entire workflow, by using env at the top level of the workflow file.
The contents of a job within a workflow, by using jobs.<job_id>.env.
A specific step within a job, by using jobs.<job_id>.steps[*].env.
Reference:
https://docs.github.com/en/actions/how-tos/write-workflows/choose-what-workflows-do/use- variables

質問: **60**
開発者としてJavaScriptアクションを作成しました。JavaScriptアクションをテストする最適な方法は何でしょうか？
**A.** 特定の JavaScript アクションのみを実行するワークフローを作成します。
**B.** JavaScript アクションを docker コンテナ イメージ内にパッケージ化して実行します。
**C.** @vercel/ncc というツールを使用してコードをコンパイルします。
**D.** actions/debug-javascript アクションを含むワークフローを作成します。
正解: **A (コメントを発表する)**
Testing out your JavaScript action in a workflow
You can test your action out in a workflow.
Public actions can be used by workflows in any repository. When an action is in a private repository, the repository settings dictate whether the action is available only within the same repository or also to other repositories owned by the same user or organization.
Reference:
https://docs.github.com/en/actions/tutorials/create-actions/create-a-javascript-action

テストエンジンを手に入れます。最新版のアクセス、https://www.jpntest.com/shiken/GH-200-JPN-mondaishu　**102**問、３０%ディスカウント、特別な割引コード: **JPNshiken**」