Microsoft.GH-200.v2026-09-29.q65

Exam Code:	GH-200
Exam Name:	GitHub Actions
Certification Provider:	Microsoft
Free Question Number:	65
Version:	v2026-09-29
# of views:	120
# of Questions views:	650
https://www.freepdfdumps.com/Microsoft.GH-200.v2026-09-29.q65.html	

NEW QUESTION: 1
Your organization has a secret that must be available to all the repositories within GitHub Actions workflows.
You need to store the secret. The solution must minimize administrative effort.
What should you do in GitHub?
A. For each GitHub repository, select Settings > Secrets and variables > Actions, and then add the secret.
B. From your personal GitHub profile, select Settings > Developer settings > Secrets, and then add the secret.
C. From the organization ' s page, select Settings > Security > Secrets, and then add the secret.
D. From the organization ' s page, select Settings > Secrets and variables > Actions, and then add the secret.
Answer: D (LEAVE A REPLY)
The correct solution is to create an organization-level Actions secret. This minimizes administrative effort because the secret can be managed once at the organization level and made available to all repositories or selected repositories through an access policy. Option D gives the correct navigation path: organization page, Settings, Secrets and variables, then Actions. Option A would require duplicating the secret in every repository, increasing maintenance overhead and risk of inconsistent values. Option B is wrong because personal developer settings do not centrally expose a secret to all organization repositories. Option C is incomplete because the current GitHub path places Actions secrets under "Secrets and variables," not just a generic "Security > Secrets" location. GitHub documents organization secrets and the repository access policy model for sharing them.

NEW QUESTION: 2

What metadata file in a custom action defines the main entry point?

A. action.js

B. index.js

C. action.yml

D. main.yml

Answer: (SHOW ANSWER)

The action.yml file is the metadata file in a custom GitHub Action that defines the main entry point, including information such as the inputs, outputs, description, and the runs key that specifies the main entry point (e.g., a script or a Docker container).

NEW QUESTION: 3

As a developer, you need to make sure that only actions from trusted sources are available for use in your GitHub Enterprise Cloud organization. Which of the following statements are true? (Choose three.)

A. Specific actions can individually be enabled for the organization, including version information.

B. GitHub-verified actions can be collectively enabled for use in the enterprise.

C. Actions can be restricted to only those available in the enterprise.

D. Actions created by GitHub are automatically enabled and cannot be disabled.

E. Individual third-party actions enabled with a specific tag will prevent updated versions of the action from introducing vulnerabilities.

F. Actions can be published to an internal marketplace.

Answer: A,B,F (LEAVE A REPLY)

You can enable specific actions for the organization by identifying them and providing version control, ensuring only trusted versions are used in workflows.

GitHub-verified actions can be enabled at the enterprise level, providing an extra layer of security by ensuring that only trusted actions are available to workflows.

Actions can be published to an internal marketplace, allowing organizations to share reusable actions securely within their enterprise without exposing them to the public.

NEW QUESTION: 4

A single secret must be accessed by workflows in specific repositories. What is the best way to create the secret?

A. Create an organization secret, specify Selected repositories as the repository access, and select the required repositories.

B. Create an environment secret at the organization level and leverage that environment in each of the specified repositories.

C. Store the secret in a supported external key vault. Configure OpenID Connect (OIDC) to allow access to the external vault and link the secret from the external key vault in each of the specific repositories.

D. Create the secret in one of the repositories, check the Share secret option, and select the required repositories.

Answer: A (LEAVE A REPLY)

GitHub Actions supports organization-level secrets with repository access policies. When the same secret must be centrally managed but made available only to particular repositories, an organization owner can create an organization secret and configure its Repository access policy for selected repositories. This avoids creating and maintaining separate copies of the secret in every repository while still enforcing scope restrictions. Environment secrets are created for environments within individual repositories, so there is no organization-level environment secret matching option B. GitHub repositories also do not provide a general Share secret option like option D. External secret-management systems can be integrated with GitHub Actions, but they are unnecessary for this requirement. Therefore, an organization secret restricted to selected repositories is the direct GitHub-supported solution.

NEW QUESTION: 5

As a developer, you need to integrate a GitHub Actions workflow with a third-party code quality provider that uses the Checks API. How should you trigger a follow-up workflow?

A. Add the workflow_run webhook event as a trigger for the workflow for the code quality integration name

B. Add the check_run webhook event as a trigger for the workflow when the code quality integration is completed

C. Add the pull_request webhook event as a trigger for the workflow when the code quality integration is synchronized

D. Add the deployment webhook event as a trigger for the workflow when the code quality integration is completed

Answer: B (LEAVE A REPLY)

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.

NEW QUESTION: 6

Which command can you include in your workflow file to set the output parameter for an action?

A. echo " action_color=purple " > > $GITHUB_OUTPUT

B. echo " action_color=purple " > > $GITHUB_ENV

C. echo " ::add-mask::$ACTION_COLOR "

D. echo " ::debug::action_color=purple "

Answer: A (LEAVE A REPLY)

GitHub Actions uses the special GITHUB_OUTPUT environment file to define output parameters for a workflow step. The required format is name=value, appended to the file.

Therefore, writing action_color=purple to $GITHUB_OUTPUT correctly creates an output named action_color. The step should also have an id if another step needs to reference the value through the steps. < id > .outputs context.
GITHUB_ENV serves a different purpose: it creates environment variables for subsequent steps in the same job. The add-mask workflow command masks sensitive values in logs, while debug writes diagnostic information. Neither defines an output parameter. GitHub ' s current workflow-command documentation explicitly identifies GITHUB_OUTPUT as the supported environment file for setting step outputs.

NEW QUESTION: 7
GitHub-hosted runners support which capabilities? (Choose two.)
A. automatic patching of both the runner and the underlying OS
B. automatic file-system caching between workflow runs
C. support for Linux, Windows, and mac
D. support for a variety of Linux variations including CentOS, Fedora, and Debian
E. requiring a payment mechanism (e.g., credit card) to use for private repositories
Answer: C,D (LEAVE A REPLY)
GitHub-hosted runners automatically handle patching, meaning they will be kept up to date with the latest security updates and software patches for both the runner environment and the underlying operating system.
GitHub-hosted runners support Linux, Windows, and macOS, giving you flexibility to run workflows on different operating systems without needing to manage your own self-hosted runners.

NEW QUESTION: 8
Which statement accurately describes using labels to route GitHub Actions workflows to runners?
A. A user can assign multiple labels to a self-hosted runner and use the labels in a runs-on to target only runners that match all the specified labels.
B. Labels apply only to GitHub-hosted runners and are ignored by self-hosted runners.
C. GitHub Actions automatically assigns the runner's IP address as a label, and the label must be used in a runs-on.
D. To target a specific runner, the runs-on must match the runner's hostname rather than its labels.
Answer: (SHOW ANSWER)
Self-hosted runners can have default labels and custom labels. These labels are used in the runs-on field to route workflow jobs to runners that match the required criteria. When multiple labels are specified, the runner must match all listed labels before GitHub Actions can assign the job to that runner. Therefore, option A is correct. Option B is false because labels are especially important for self-hosted runners. Option C is incorrect because GitHub does not require the runner IP address as a routing label. Option D is also incorrect

because workflows target runners by labels or runner groups, not by hostname. This topic validates runner selection, self-hosted runner configuration, and workflow job routing.

NEW QUESTION: 9
Which syntax correctly accesses a job output (output1) of an upstream job (job1) from a dependent job within a workflow?
A. ${{needs.job1.outputs.output1}}
B. ${{needs.job1.output1}}
C. ${{depends.job1.output1}}
D. ${{job1.outputs.output1}}
Answer: (SHOW ANSWER)
The needs context is used to reference the outputs of jobs that are dependencies of the current job. In this case, needs.job1.outputs.output1 correctly accesses the output of output1 from the job job1 in the dependent job.

NEW QUESTION: 10
As a developer, you are designing a workflow and need to 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. Which of the following options should you use?
A. environment variables
B. workflow commands
C. self-hosted runners
D. enable debug loggingE composite run step
Answer: (SHOW ANSWER)
Workflow commands are special commands that allow you to interact with the runner, set environment variables, output values, add debug messages, and perform other tasks within the workflow. These commands are used to modify the environment or influence the behavior of the GitHub Actions runner.

NEW QUESTION: 11
What is the most secure way to store sensitive information in GitHub Actions workflows?
A. Use environment variables in the workflow and store sensitive data directly in the variables.
B. From the GitHub repository settings, store sensitive data as unencrypted text.
C. Store the sensitive information as plain text in the workflow YAML file.
D. Use GitHub Enterprise Managed User (OIDC) integration to dynamically fetch secrets from a third- party secrets manager.
Answer: (SHOW ANSWER)
The most secure option listed is to use OIDC-based integration to obtain short-lived credentials from an external secrets manager or cloud provider. This avoids storing long-

lived sensitive values directly in workflow YAML, repository text, or regular environment variables. Options A, B, and C are insecure because they expose or persist sensitive information in places that can be read, logged, committed, or mismanaged.
With OIDC, the workflow requests a token from GitHub's OIDC provider and exchanges it with the trusted external provider for a short-lived credential. This reduces secret sprawl and supports stronger authorization controls. GitHub documents OIDC as a way to avoid duplicating long-lived cloud credentials as GitHub secrets and to use short-lived access tokens instead.

NEW QUESTION: 12
As a developer, your self-hosted runner sometimes loses its connection while running jobs. How should you troubleshoot the issue affecting your self-hosted runner?
A. Access the self-hosted runner ' s installation directory and look for log files in the _diag folder.
B. Locate the self-hosted runner in your repository ' s settings page and download its log archive.
C. Start the self-hosted runner with the --debug flag to produce more verbose console output.
D. Set the DEBUG environment variable to true before starting the self-hosted runner.
Answer: A (LEAVE A REPLY)
GitHub self-hosted runners maintain diagnostic information locally in the _diag directory beneath the runner installation directory. The runner application creates Runner_*.log files containing information about startup, connectivity, registration, updates, and general runner activity. Job-specific execution details are written to Worker_*.log files in the same diagnostic directory. These logs are therefore the primary source when investigating intermittent connection or execution problems. GitHub ' s troubleshooting documentation explicitly directs administrators to inspect the _diag directory and also provides connectivity-check scripts for diagnosing communication failures. The other options describe mechanisms that are not the documented standard procedure: there is no general repository-settings feature for downloading the runner ' s local diagnostic archive, and the suggested generic --debug and DEBUG=true mechanisms are not the prescribed runner troubleshooting method.

NEW QUESTION: 13
What is the right method to ensure users approve a workflow before the next step proceeds?
A. creating a branch protection rule and only allow certain users access
B. granting users workflow approval permissions
C. adding users as required reviewers for an environment
D. granting users repository approval permissions
Answer: C (LEAVE A REPLY)

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.

NEW QUESTION: 14
What is a valid scenario regarding environment secrets?
A. Ensuring a job cannot access environment secrets until approval is obtained from a required reviewer.
B. Ensuring only a specific step can access an environment secret.
C. Configuring environment secrets for connecting to GitHub Enterprise Server.
D. Configuring environment secrets to automatically pull from Azure Key Vault.
Answer: (SHOW ANSWER)
Environment secrets are scoped to GitHub Actions environments and can be protected by environment protection rules. If an environment requires reviewers, a job that references that environment cannot access its environment secrets until the required approval is granted. This makes option A correct. Option B is incorrect because environment secrets are made available to jobs that reference the protected environment, not selectively to only one specific step by default. Option C is not the purpose of environment secrets; they are not specifically for connecting to GitHub Enterprise Server. Option D is also incorrect because GitHub environment secrets do not automatically pull from Azure Key Vault without additional integration or custom workflow logic. This topic covers environment protection, secrets, and deployment governance.

NEW QUESTION: 15
As a DevOps engineer, you are developing a container action. You need to execute a cleanup script after completing the main script execution. Which code block should be used to define the cleanup script?
A. post: cleanup.sh
B. after: cleanup.sh
C. final: cleanup.sh
D. cleanup: cleanup.sh
Answer: A (LEAVE A REPLY)
The correct syntax for specifying a cleanup script to be run after the main entry point is executed in a Docker- based GitHub Action is to use the post key. This ensures that cleanup.sh runs after the main script (main.sh) has completed.

NEW QUESTION: 16
As a developer, you have configured an IP allow list on a GitHub organization. Which effects does the IP allow list have on GitHub Actions?
(Each correct answer presents a complete solution. Choose two.)

A. You must allow GitHub Actions ' IP address ranges in order to use Marketplace actions.
B. You can use standard GitHub-hosted runners since their IP addresses are automatically allowed.
C. You can use GitHub-hosted larger runners since they can be configured with static IP addresses.
D. You can use self-hosted runners with known IP addresses.
Answer: C,D (LEAVE A REPLY)
GitHub ' s current documentation states that when an organization uses an IP allow list with GitHub Actions, the appropriate choices are self-hosted runners or GitHub-hosted larger runners configured with static IP address ranges. A self-hosted runner has an address controlled by the organization, so its IP address or range can be explicitly added to the allow list. Larger GitHub-hosted runners can likewise be configured with predictable static IP ranges. Standard GitHub-hosted runners normally use dynamically assigned addresses from large shared ranges, and GitHub specifically recommends against relying on those ranges for this scenario. Marketplace action availability itself is not controlled by adding generic Actions IP ranges as option A suggests. Therefore, the solutions consistent with current GitHub Enterprise Cloud guidance are larger runners with static addresses and self-hosted runners with known addresses.

NEW QUESTION: 17
How should you install the bats NPM package in your workflow?
A)
B)
C)
D)
A. Option A
B. Option B
C. Option C
D. Option D
Answer: D (LEAVE A REPLY)

The correct syntax includes specifying the job (example-job), the runner (ubuntu-latest), and the necessary step (npm install -g bats) within the workflow. This ensures that the package is installed properly during the execution of the job.

NEW QUESTION: 18
You create a self-hosted runner labeled as runner1.
You need to ensure that a GitHub Actions workflow job runs only on runner1.
Which YAML statement should you use?
A. runs-on: [self-hosted, linux-xlarge]
B. runs-on: - self-hosted - ubuntu-latest
C. runs-on: [self-hosted, runner1]
D. runs-on: self-hosted
Answer: C (LEAVE A REPLY)
To target a specific self-hosted runner, the job must use runs-on with labels that match that runner. A self- hosted runner normally has the self-hosted label plus default labels and any custom labels you assign. Since the runner is labeled runner1, the workflow should specify both self-hosted and runner1: runs-on: [self-hosted, runner1]. GitHub Actions will then route the job only to a self-hosted runner matching all specified labels.
Option A targets a different custom label. Option B is malformed and also mixes a self-hosted label with a GitHub-hosted runner label. Option D targets any self-hosted runner, not specifically runner1. GitHub documents that an array of runs-on labels requires a runner matching all listed labels.

NEW QUESTION: 19
You need to use a specific version of an action. How should you use v1 of an actions/javascript-action repository in your workflow?
A. steps:uses: actions/javascript-action/v1
B. steps:uses: actions/javascript-actionversion: v1
C. steps:uses: actions/javascript-action-v1
D. steps:uses: actions/javascript-action@v1
Answer: D (LEAVE A REPLY)
The correct syntax for referencing a versioned action is owner/repository@ref. Therefore, to use version v1 of actions/javascript-action, the workflow step must use actions/javascript-action@v1. The @v1 portion is the ref, which can represent a tag, branch, or commit SHA depending on how the action publisher releases the action. Option A incorrectly treats the version as part of the repository path. Option B invents a separate version: key, which is not how action versions are selected. Option C changes the repository name and does not specify a ref. In GitHub Actions syntax, uses selects an action, and version selection is part of the action reference itself. GitHub examples use the uses: owner/action-name@ref form.

NEW QUESTION: 20
Which default environment variable specifies the branch or tag that triggered a workflow?
A. GITHUB_TAG
B. GITHUB_REF
C. ENV_BRANCH
D. GITHUB_BRANCH
Answer: B (LEAVE A REPLY)
The GITHUB_REF environment variable specifies the branch or tag that triggered the workflow. It contains the full reference to the branch or tag, such as refs/heads/main for a branch or refs/tags/v1.0 for a tag.

NEW QUESTION: 21
As a DevOps engineer developing a JavaScript action, you need to include annotations to pass warning messages to workflow runners. Which code snippet can you use to implement an annotation in your Actions?
As a DevOps engineer developing a JavaScript action, you need to include annotations to pass warning messages to workflow runners. Which code snippet can you use to implement an annotation in your Actions?
A. core.info('Something went wrong, but it\'s not bad enough to fail the build.')
B. core.notice('Something went wrong, but it\'s not bad enough to fail the build.')
C. core.warning('Something went wrong, but it\'s not bad enough to fail the build.')
D. core.warn('Something went wrong, but it\'s not bad enough to fail the build.')
Answer: C (LEAVE A REPLY)
The core.warning() function from the @actions/core package is used to create a warning message in the workflow logs. This is an annotation type that informs users about issues that don't require failing the build but still need attention.

NEW QUESTION: 22
As a developer, you are configuring GitHub Actions to deploy VMs to production. A member of the VMOps team must provide approval before the deployment occurs. Which of the following steps should you take?
(Each correct answer presents part of the solution. Choose two.)
A. Specify the environment named Production in the workflow jobs that deploy to the VMs.
B. Navigate to the organization settings and create a Production environment with the VMOps team as a required reviewer.
C. Navigate to the repository settings and create a Production environment with the VMOps team as a required reviewer.
D. Specify the VMOps team as the owner of the environment.
E. Add the VMs to the environment.
Answer: A,C (LEAVE A REPLY)

GitHub environments provide deployment protection controls such as required reviewers. First, the Production environment should be created in the repository ' s Settings > Environments configuration, where the VMOps team can be configured as required reviewers. This makes option C correct; the environment is not created as an organization-level environment as described by option B. Second, the deployment job must reference the Production environment. GitHub applies the environment ' s protection rules to jobs that reference that environment, so the deployment remains in a waiting state until an authorized reviewer approves it. GitHub environments are logical deployment-control resources; individual VMs do not need to be added to an environment, nor is an environment "owned" by a team. Therefore, options A and C together implement the required production approval gate.

NEW QUESTION: 23
When creating and managing custom actions in an enterprise setting, which of the following is considered a best practice?
A. creating a separate repository for each action so that the version can be managed independently
B. creating a separate branch in application repositories that only contains the actions
C. creating a single repository for all custom actions so that the versions for each action are all the same
D. including custom actions that other teams need to reference in the same repository as application code
Answer: (SHOW ANSWER)
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.

NEW QUESTION: 24
You are a developer working on developing reusable workflows for your organization. What keyword should be included as part of the reusable workflow event triggers?
A. check_run
B. pull_request
C. workflow_run
D. workflow_call
Answer: D (LEAVE A REPLY)
A workflow becomes reusable when its on configuration includes the workflow_call event. This event declares that the workflow can be invoked by another workflow rather than only being started by ordinary repository events. Inputs and secrets can also be defined beneath workflow_call so the caller can provide values to the reusable workflow. workflow_run has a different purpose: it starts a workflow in response to another workflow

running or completing. pull_request responds to pull-request activity, while check_run responds to GitHub Checks activity. GitHub ' s official reusable workflow documentation explicitly states that a reusable workflow must contain on: workflow_call. Therefore, workflow_call is the required keyword when defining a workflow intended to be called from other workflows.

NEW QUESTION: 25
As a DevOps engineer, you are developing workflows to build an application. You have a requirement to create the build targeting multiple node versions. Which code block should you use to define the workflow?
A. jobs:build-app:strategy:matrix:node-ver: [10, 12, 14]steps:- uses: actions/setup-node@v3with:node- version: ${{ strategy.node-ver }}
B. jobs:build-app:strategy:matrix:node-ver: [10, 12, 14]steps:- uses: actions/setup-node@v3with:node- version: ${{ matrix.node-ver }}
C. jobs:build-app:matrix-strategy:node-ver: [10, 12, 14]steps:- uses: actions/setup-node@v3with:node- version: ${{ matrix-strategy.node-ver }}
D. jobs:build-app:matrix:strategy:node-ver: [10, 12, 14]steps:- uses: actions/setup-node@v3with:node- version: ${{ matrix.node-ver }}
Answer: (SHOW ANSWER)
The correct GitHub Actions syntax for running a job across multiple Node.js versions is strategy.matrix. The matrix values are referenced at runtime through the matrix context, so node-version: ${{ matrix.node-ver }} is correct. Option A defines the matrix correctly but references it incorrectly through strategy.node-ver; matrix values are not accessed through the strategy context. Option C uses matrix-strategy, which is not a valid GitHub Actions workflow key. Option D reverses the structure by putting strategy under matrix, which is invalid. This question tests matrix strategy syntax, dynamic job configuration, and using matrix values as inputs to actions such as actions/setup-node. GitHub defines jobs. < job_id > .strategy.matrix as the proper matrix mechanism.

NEW QUESTION: 26
Which default GitHub environment variable indicates the owner and repository name?
A. REPOSITORY NAME
B. GITHUB REPOSITORY
C. ENV REPOSITORY
D. GITHUB WORKFLOW REPO
Answer: A (LEAVE A REPLY)
The GITHUB_REPOSITORY environment variable contains the owner and repository name in the format owner/repository. It is automatically provided by GitHub Actions and can be used to reference the repository in workflows.

NEW QUESTION: 27

In which scenarios could the GITHUB_TOKEN be used? (Choose two.)
A. to leverage a self-hosted runner
B. to create a repository secret
C. to publish to GitHub Packages
D. to create issues in the repo
E. to read from the file system on the runner
F. to add a member to an organization
Answer: C,D (LEAVE A REPLY)
The GITHUB_TOKEN is automatically provided by GitHub in workflows and can be used to authenticate API requests to GitHub, including publishing packages to GitHub Packages. The GITHUB_TOKEN is also used to authenticate API requests for actions like creating issues, commenting, or interacting with pull requests within the same repository.

NEW QUESTION: 28
What are the two most significant advantages of adding documentation while distributing custom actions?
Each correct answer presents a complete solution.
NOTE: Each correct answer is worth one point.
A. It creates a readme.md for the consuming workflow.
B. It shares the description of the action to the users.
C. It provides an example of the action.
D. It generates auto completion when using the action in a workflow.
Answer: B,C (LEAVE A REPLY)
Documentation is critical when distributing custom GitHub Actions because users need to understand what the action does and how to use it correctly. A good README or marketplace description explains the action's purpose, required inputs, outputs, permissions, and expected behavior. This directly supports option B.
Documentation should also include practical usage examples, such as a workflow snippet showing the uses:
syntax and required with: inputs, which supports option C. Option A is incorrect because documentation does not create a README inside the consuming workflow; the action author provides documentation in the action repository. Option D is also incorrect because documentation does not automatically generate workflow auto-completion. This topic tests custom action publishing and usability best practices.

NEW QUESTION: 29
Disabling a workflow allows you to stop a workflow from being triggered without having to delete the file from the repo. In which scenarios would temporarily disabling a workflow be most useful? (Choose two.)
A. A workflow sends requests to a service that is down.

B. A workflow error produces too many, or wrong, requests, impacting external services negatively.
C. A workflow is configured to run on self-hosted runners
D. A workflow needs to be changed from running on a schedule to a manual trigger
E. A runner needs to have diagnostic logging enabled.
Answer: (SHOW ANSWER)
If a workflow depends on an external service that is down, disabling the workflow temporarily will prevent it from running and sending requests to the service, thus avoiding failed requests or unnecessary retries.
If a workflow is causing a negative impact on external services by generating too many requests or incorrect data due to a bug, temporarily disabling the workflow will stop this behavior while the issue is fixed.

NEW QUESTION: 30
What are the two ways to pass data between jobs? (Choose two.)
A. Use the copy action with restore parameter to restore the data from the cache
B. Use the copy action to save the data that should be passed in the artifacts folder.
C. Use the copy action with cache parameter to cache the data
D. Use data storage.
E. Use job outputs
F. Use artifact storage.
Answer: (SHOW ANSWER)
Job outputs are used to pass data from one job to another in a workflow. A job can produce output values (like variables or files), which can be referenced by subsequent jobs using the needs keyword and ${{ steps.step_id.
outputs.output_name }} syntax.
Artifact storage allows data (such as files or results) to be saved by a job and then retrieved by another job in a later step. This is commonly used for passing large amounts of data or files between jobs.

NEW QUESTION: 31
As a developer, you need to use GitHub Actions to deploy a microservice that requires runtime access to a secure token. This token is used by a variety of other microservices managed by different teams in different repos. To minimize management overhead and ensure the token is secure, which mechanisms should you use to store and access the token? (Choose two.)
A. Store the token in a configuration file in a private repository. Use GitHub Actions to deploy the configuration file to the runtime environment.
B. Store the token as a GitHub encrypted secret in the same repo as the code. Create a reusable custom GitHub Action to access the token by the microservice at runtime.

C. Use a corporate non-GitHub secret store (e.g., HashiCorp Vault) to store the token. During deployment, use GitHub Actions to store the secret in an environment variable that can be accessed at runtime.
D. Store the token as a GitHub encrypted secret in the same repo as the code. During deployment, use GitHub Actions to store the secret in an environment variable that can be accessed at runtime.
E. Store the token as an organizational-level encrypted secret in GitHub. During deployment, use GitHub Actions to store the secret in an environment variable that can be accessed at runtime.
Answer: C,E (LEAVE A REPLY)
Using a corporate secret store like HashiCorp Vault provides a secure, centralized location for sensitive information. GitHub Actions can then retrieve and store the token securely during deployment by setting it as an environment variable, ensuring the token remains secure and accessible at runtime.
Storing the token as an organizational-level encrypted secret in GitHub ensures it is accessible across multiple repositories, minimizing management overhead. GitHub Actions can then use this secret during deployment by setting it as an environment variable, allowing the microservice to access it securely at runtime.

NEW QUESTION: 32
What are the two types of environment protection rules you can configure? (Choose two.)
A. required reviewers
B. branch protections
C. wait timer
D. artifact storage
Answer: A,C (LEAVE A REPLY)
Required reviewers is a protection rule where you can specify that certain individuals or teams must review and approve the workflow run before it can proceed. This is used to enforce approvals before certain steps or environments are accessed.
Wait timer is a protection rule that introduces a delay before a workflow can proceed to the next stage. This is useful for adding time-based constraints to the deployment process or ensuring that certain conditions are met before a workflow continues.

NEW QUESTION: 33
You are a DevOps engineer working on a custom action. You want to conditionally run a script at the start of the action, before the main entrypoint. Which code block should be used to define the metadata file for your custom action?
A. runs:using: ' node16 ' start: ' start.js ' start-if: github.event_name == ' push ' main: ' index.js '
B. runs:using: ' node16 ' pre: ' start.js ' pre-if: github.event_name == ' push ' main: ' index.js '
C. runs:using: ' node16 ' pre-if: github.event_name == ' push ' then ' start.js ' main: ' index.js '
D. runs:using: ' node16 ' before: ' start.js ' before-if: github.event_name == ' push ' main: ' index.js '
Answer: B (LEAVE A REPLY)
For JavaScript custom actions, the metadata file supports lifecycle scripts through the runs section. The correct key for a script that runs before the main action entrypoint is pre, and the correct conditional key for controlling whether that pre-script runs is pre-if. Therefore, option B is the only valid metadata structure.
GitHub Actions does not use start, start-if, before, or before-if as action metadata keys. Option C is also invalid because pre-if only defines a condition; it cannot contain both a condition and a script reference in one line. In GitHub Actions exam terms, this tests custom action metadata, JavaScript action lifecycle hooks, and conditional execution before the main action logic.

NEW QUESTION: 34
As a developer, you are using a Docker container action in your workflow. What is required for the action to run successfully?
A. The job runs-on must specify a Linux machine with Docker installed.
B. The job env must be set to a Linux environment.
C. The action must be published to the GitHub Marketplace.
D. The referenced action must be hosted on Docker Hub.
Answer: (SHOW ANSWER)

NEW QUESTION: 35
As a developer, you need to create a custom action written in Python. Which action type should you choose?
A. Python action
B. JavaScript action
C. Docker container action
D. Composite run step
Answer: C (LEAVE A REPLY)

GitHub officially defines three major custom-action types: JavaScript actions, Docker container actions, and composite actions. There is no separate native Python action type. A Docker container action is appropriate when the implementation is written in Python because the container can package the Python runtime, application code, dependencies, and required operating-system components together. GitHub states that Docker container actions can use any base Docker image and therefore any programming language. This provides a predictable execution environment independent of the software preinstalled on the runner.
JavaScript actions specifically execute packaged JavaScript using a supported Node.js runtime. Composite actions primarily combine multiple workflow steps and commands. For a self-contained custom action implemented in Python with its runtime and dependencies packaged alongside it, the Docker container action is the correct choice.

NEW QUESTION: 36
As a developer, one of your workflows requires Xcode version 11.2 hosted on macOS Catalina (version
10.15). You have already created and configured a self-hosted runner to conform to those requirements and registered it with your organization. What else should you do to ensure that the workflow accesses the correct runner instance?
(Each correct answer presents part of the solution. Choose three.)
A. Create runner groups named macos-10.15 and xcode-11.2.
B. In the workflow, specify:
runs-on: [${{ groups.macos-10.15 }}, ${{ groups.xcode-11.2 }}]
C. Assign the custom labels to the self-hosted runner.
D. Create custom runner labels for macos-10.15 and xcode-11.2.
E. Add your runner to the appropriate runner groups.
F. In the workflow, specify:
runs-on: [self-hosted, macos-10.15, xcode-11.2]
Answer: (SHOW ANSWER)
GitHub Actions routes jobs to self-hosted runners by using runner labels. Custom labels representing required capabilities, such as the operating-system version and Xcode version, can be created and assigned to the appropriate runner. The workflow then specifies those labels in runs-on. When multiple labels are supplied, GitHub requires an eligible runner to match all specified labels, so runs-on: [self-hosted, macos-10.15, xcode-11.2] targets the intended runner configuration. Runner groups are primarily an access-control and organizational mechanism and are not represented by the ${{ groups.* }} expression shown in option B. The Xcode/macOS versions in this exam scenario are historical, but the runner-label routing principle remains current. GitHub documentation confirms custom labels and cumulative runs-on label matching.

NEW QUESTION: 37

What is the minimal syntax for declaring an output named foo for an action?
A)
B)
C)
D)
A. Option A
B. Option B
C. Option C
D. Option D
Answer: C (LEAVE A REPLY)
The correct minimal syntax for declaring an output in GitHub Actions is by using the foo key under outputs, and associating it with a value (in this case, Some value). This is the simplest form to define an output in a workflow or action.

NEW QUESTION: 38
As a DevOps engineer, you need to define a deployment workflow that runs after the build workflow has successfully completed. Without modifying the build workflow, which trigger should you define in the deployment workflow?
A. repository_dispatch
B. workflow_run
C. workflow_exec
D. workflow_dispatch
Answer: B (LEAVE A REPLY)
The workflow_run event is designed to trigger one workflow based on the execution or completion of another workflow. The deployment workflow can therefore specify the build workflow under on.workflow_run.
workflows and use the completed activity type without changing the original build workflow. One important detail is that workflow_run fires when the specified workflow completes regardless of whether it succeeded or failed. If deployment must occur only after a successful build, the deployment job should additionally use a condition such as github.event.workflow_run.conclusion == ' success ' . workflow_dispatch is a manual trigger, repository_dispatch is normally used for externally generated custom events through the GitHub API, and workflow_exec is not a GitHub Actions workflow event. Therefore, workflow_run is the correct trigger.

NEW QUESTION: 39
Which workflow command would output the debug message "action successfully debugged"?
A. echo :debug::message=action successfully debugged"
B. echo "debug-action successfully debugged"
C. echo "::debug::action successfully debugged"

D. echo ":debug:action successfully debugged:"
Answer: C (LEAVE A REPLY)
The ::debug:: syntax is used to output debug messages in GitHub Actions workflows. This command will print the message "action successfully debugged" in the debug logs when the workflow runs.

NEW QUESTION: 40
Which of the following is the proper syntax to specify a custom environment variable named MY_VARIABLE with the value my-value?
A. env:
MY_VARIABLE = my-value
B. environment:
MY_VARIABLE = my-value
C. env:
MY_VARIABLE: my-value
D. var:
MY_VARIABLE: my-value
E. var:
MY_VARIABLE = my-value
F. environment:
MY_VARIABLE: my-value
Answer: (SHOW ANSWER)
GitHub Actions defines custom environment variables with the YAML env mapping. YAML mappings use a colon between each key and value, so MY_VARIABLE: my-value beneath env: is the correct syntax. The env key can appear at workflow, job, or step scope depending on how broadly the variable should be available.
Options A, B, E, and F either use an unsupported key or the wrong assignment syntax. environment is associated with deployment environments when used in GitHub Actions workflow syntax; it is not the key for declaring ordinary environment variables. The vars context is used for configuration variables created through repository, organization, or environment settings, but var: is not the workflow syntax shown here. Therefore, option C correctly defines the requested custom environment variable.

NEW QUESTION: 41
Where should workflow files be stored to be triggered by events in a repository?
A. .github/workflows/
B. .github/actions/
C. Nowhere; they must be attached to an act on in the GitHub user interface
D. anywhere
E. .workflows/
Answer: (SHOW ANSWER)

Workflow files must be stored in the .github/workflows/ directory of the repository. This is the standard location for GitHub Actions workflow files, and workflows in this directory are automatically triggered by events defined in the file, such as pushes, pull requests, or other GitHub events.

NEW QUESTION: 42
As a developer, you are authoring a workflow that will deploy to both DevCloud and TestCloud resources.
Each cloud resource is accessed with a different deployment key. Which approach best allows you to use the same reusable workflow in separate jobs to target the different cloud resources?
A. Create repository secrets named DevCloud.DEPLOY_KEY and TestCloud.DEPLOY_KEY so that the reusable workflow parses the secrets by resource name.
B. Store the different keys in a DEPLOY_KEY environment secret in the DevCloud and TestCloud environments. Specify DEPLOY_KEY in the secrets section of the reusable workflow.
C. Use a marketplace action to conditionally parse the DEPLOY_KEY repository secret based on the cloud resource name.
D. Populate a DEPLOY_KEY repository secret with a JSON object containing DevCloud and TestCloud properties. Then specify DEPLOY_KEY.DevCloud in the secrets sections of the reusable workflow.
Answer: (SHOW ANSWER)
The best design is to use environment-scoped secrets with the same secret name, such as DEPLOY_KEY, in separate environments such as DevCloud and TestCloud. This lets the same reusable deployment logic reference one consistent secret name while the selected environment supplies the correct value. Option A is invalid because GitHub secret names should not be designed around dotted property access, and reusable workflows should not parse secret names dynamically. Option C introduces unnecessary marketplace dependency and weak secret handling. Option D is also wrong because GitHub secrets are opaque strings; ${{ secrets.DEPLOY_KEY.DevCloud }} is not valid secret property access. Reusable workflows can define and receive named secrets through the secrets mapping.

NEW QUESTION: 43
You are reaching your organization's storage limit for GitHub artifacts and packages. What should you do to prevent the storage limit from being reached?
A. via the .github repository owned by the organization
B. via repositories owned by the organization
C. via the GitHub Marketplace
D. via a repository owned by a third party

Answer: B (LEAVE A REPLY)
To prevent reaching the storage limit for GitHub artifacts and packages, you should manage and clean up artifacts and packages stored in repositories owned by your organization. This includes deleting unnecessary artifacts and managing the lifecycle of packages, as they contribute directly to your organization's storage quota.

NEW QUESTION: 44
As a DevOps engineer, you need to execute a deployment to different environments such as development and testing based on labels added to a pull request. The deployment should use the releases branch and trigger only when there is a change in files under the apps folder. Which code block should be used to define the deployment workflow trigger?
A. on:
pull_request_review:
types: [labeled]
branches:
- ' releases '
paths:
- ' apps/** '
B. on:
pull_request:
types: [labeled]
branches:
- ' releases/** '
paths:
- ' apps '
C. on:
pull_request:
types: [labeled]
branches:
- ' releases '
paths:
- ' apps/** '
D. on:
pull_request_label:
branches:
- ' releases '
paths:
- ' apps/** '
Answer: (SHOW ANSWER)
The pull_request event supports activity-type filtering, including the labeled activity, so a workflow can respond when a label is added to a pull request. The branches filter limits

execution according to the pull request ' s target or base branch, while the paths filter limits execution according to changed files. GitHub also states that when branch and path filters are both configured, both conditions must be satisfied . Therefore, a pull request targeting releases, containing a change beneath apps/**, and receiving a label matches option C. pull_request_review concerns review activity rather than labeling, and pull_request_label is not a GitHub Actions workflow event. Option B also uses filters that do not accurately represent the stated exact branch and folder requirement.

NEW QUESTION: 45
How many jobs will result from the following matrix configuration?
A. 3 jobs
B. 4 jobs
C. 5 jobs
D. 6 jobs
Answer: D (LEAVE A REPLY)
The matrix configuration specifies two variables: color and animal. The color variable has 2 values (green and pink), and the animal variable has 2 values (owl and magpie). This would result in 4 combinations (2 color values × 2 animal values). Additionally, the include section introduces two more combinations (color: blue and animal: owl; color: pink and animal: magpie).

NEW QUESTION: 46
Which of the following scenarios requires a developer to explicitly use the GITHUB_TOKEN or github.token secret within a workflow? (Choose two.)
A. passing the GITHUB_TOKEN secret to an action that requires a token as an input
B. making an authenticated GitHub API request
C. checking out source code with the actions/checkout@v3 action
D. assigning non-default permissions to the GITHUB_TOKEN
Answer: A,B (LEAVE A REPLY)
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. 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.

NEW QUESTION: 47
Based on the YAML below, which two statements are correct? (Choose two.)
A. This workflow will publish a package to an npm registry.
B. This workflow will publish a package to GitHub Packages.
C. This workflow file is using a matrix strategy.
D. The workflow job publish-npm will only run after the build job passes.
Answer: A,D (LEAVE A REPLY)
The publish-npm job includes the JS-DevTools/npm-publish action, which is used to publish an npm package to an npm registry.
The publish-npm job has the needs: build directive, meaning it will only run after the build job successfully completes.

NEW QUESTION: 48
Which choices represent best practices for publishing actions so that they can be consumed reliably? (Choose two.)
A. repo name
B. tag
C. commit SHA
D. organization name
E. default branch
Answer: (SHOW ANSWER)
Using a tag is a best practice because tags are immutable and represent a fixed version of your action. By referencing tags, consumers of your action can be assured they are using a stable and specific version of the action, which helps in avoiding issues with breaking changes.
The commit SHA is another reliable way to specify a particular version of an action. By referencing a specific commit SHA, consumers can ensure they are using exactly the code that was written at that moment, avoiding the potential for changes in the future.

NEW QUESTION: 49
Your organization is managing secrets using GitHub encrypted secrets, including a secret named SuperSecret.
As a developer, you need to create a version of that secret that contains a different value for use in a workflow that is scoped to a specific repository named MyRepo. How should you store the secret to access your specific version within your workflow?
A. Create a duplicate entry for SuperSecret in the encrypted secret store and specify MyRepo as the scope.

B. Create MyRepo_SuperSecret in GitHub encrypted secrets to specify the scope to MyRepo.
C. Create a file with the SuperSecret. information in the .qithub/secrets folder in MyRepo.
D. Create and access SuperSecret from the secrets store in MyRepo.
Answer: B (LEAVE A REPLY)
To scope a secret to a specific repository, you can create a new secret with a name like MyRepo_SuperSecret in the secrets section of the MyRepo repository ' s settings. This ensures that the secret is specific to that repository and can be used within its workflows.

NEW QUESTION: 50
In which of the following scenarios should you use self-hosted runners? Each correct answer presents a complete solution.
NOTE: Each correct answer is worth one point.
A. when the workflow jobs must be run on Windows 10
B. when jobs must run for longer than 6 hours
C. when you want to use macOS runners
D. when GitHub Actions minutes must be used for the workflow runs
E. when a workflow job needs to install software from the local network
Answer: B,E (LEAVE A REPLY)
Self-hosted runners are appropriate when GitHub-hosted runner limits or environment restrictions do not meet the workload requirement. GitHub-hosted runner jobs have a 6-hour job execution limit, while self-hosted runner jobs can run for up to 5 days, so jobs that must run longer than 6 hours require self-hosted runners.
Self-hosted runners also let an organization control the machine, operating system, software, hardware, and network location. Therefore, if a workflow job needs to install or access software from the local network, a self-hosted runner is the correct choice. macOS and Windows runners are available as GitHub-hosted options, and GitHub Actions minutes apply to GitHub-hosted usage, not self-hosted runner execution.

NEW QUESTION: 51
As a developer, you are using a Docker container action in your workflow. What is required for the action to run successfully?
A. The job env must be set to a Linux environment.
B. The job runs-on must specify a Linux machine with Docker installed.
C. The referenced action must be hosted on Docker Hub.
D. The action must be published to the GitHub Marketplace.
Answer: (SHOW ANSWER)
For a Docker container action to run in a GitHub Actions workflow, the runner must have Docker installed.

The runs-on attribute of the job should specify an environment that supports Docker, typically a Linux environment (e.g., ubuntu-latest), since Docker is widely supported and commonly used in Linux-based environments.

NEW QUESTION: 52
Which default GitHub environment variable indicates the name of the person or app that initiated a workflow?
A. ENV_ACTOR
B. GITHUB_WORKFLOW_ACTOR
C. GITHUB_ACTOR
D. GITHUB_USER
Answer: (SHOW ANSWER)
The GITHUB_ACTOR environment variable indicates the name of the person or app that initiated the workflow. This variable is automatically provided by GitHub in the workflow and can be used to identify the user or application triggering the workflow.

NEW QUESTION: 53
What will the output be for the following event trigger block in a workflow?
A. It throws a workflow syntax error, pointing to the types definition in issue_comment event.
B. It throws a workflow syntax error, pointing to the types definition in issues event.
C. It runs the workflow when an issue is edited or when an issue comment created.
D. It runs the workflow when an issue or issue comment in the workflow ' s repository is created or modified.
E. It runs the workflow when an issue is created or edited, or when an issue or pull request comment is created.
Answer: (SHOW ANSWER)
The provided event trigger block specifies two types of events:
For issues: the workflow triggers on opened or edited issues.
For issue_comment: the workflow triggers when an issue comment is created.
This configuration ensures the workflow will run when either an issue is opened or edited, or an issue comment is created.

NEW QUESTION: 54
How should you print a debug message in your workflow?
A. echo " Set variable MyVariable to true " > > $DEBUG_MESSAGE
B. echo " ::debug::Set variable MyVariable to true "
C. echo " ::add-mask::Set variable MyVariable to true "
D. echo " debug_message=Set variable MyVariable to true " > > $GITHUB_OUTPUT
Answer: (SHOW ANSWER)

GitHub Actions provides the debug workflow command for writing debug-level messages to the workflow log. The command syntax is ::debug::{message}, so option B correctly sends the message to the runner through standard output. To make these debug messages visible in workflow logs, step debug logging must also be enabled using ACTIONS_STEP_DEBUG=true. Option A refers to a nonexistent $DEBUG_MESSAGE environment file. Option C uses add-mask, which instructs GitHub to redact a value from logs rather than print diagnostic information. Option D writes a value to GITHUB_OUTPUT, which creates a step output instead of a debug message. GitHub documents the Bash syntax with echo " ::debug::
message " , making option B the supported workflow-command format.

NEW QUESTION: 55
Without the need to use additional infrastructure, what is the simplest and most maintainable method for configuring a workflow job to provide access to an empty PostgreSQL database?
A. Use service containers with a Postgres database from Docker hub.
B. Run the actions/postgres action in a parallel job.
C. It is currently impossible to access the database with GitHub Actions.
D. Dynamically provision and deprovision an environment.
Answer: (SHOW ANSWER)
GitHub Actions supports the use of service containers, which allows you to spin up a PostgreSQL database (or any other service) in a Docker container during your workflow. You can pull a PostgreSQL image from Docker Hub, and the container will automatically be available to your workflow job. This method requires no additional infrastructure and is easy to configure and maintain, as you simply define the container in the workflow file.

NEW QUESTION: 56
Which step is using the dbserver environment variable correctly?
A. steps:
- name: Hello world
run: echo $dbserver
variables:
dbserver: orgserver1
B. steps:
- name: Hello world
run: echo $dbserver
env:
dbserver: orgserver1
C. steps:
- name: Hello world
run: echo $dbserver

env:
- name: dbserver
value: orgserver1
D. steps:
- name: Hello world
run: echo $dbserver
environment:
dbserver: orgserver1
Answer: B (LEAVE A REPLY)
GitHub Actions uses the env key to define environment variables. The env mapping can be applied at the workflow, job, or individual step level. In option B, dbserver is defined directly beneath the step-level env key, making it available to that step ' s shell process. Because the example uses shell syntax appropriate to a Linux runner, $dbserver correctly references the environment variable from the run command. The variables and environment keys shown in options A and D are not valid substitutes for step-level env. Option C also uses an invalid list structure; env expects a mapping of variable names to values rather than name and value objects. Therefore, option B uses the supported GitHub Actions workflow syntax.

NEW QUESTION: 57
An organization ' s policies specify that only local actions are allowed. How should actions be distributed for this organization?
A. Via the GitHub Marketplace
B. Via the .github repository owned by the organization
C. Via a repository owned by a third party
D. Via repositories owned by the organization
Answer: D (LEAVE A REPLY)
GitHub allows an organization to restrict workflows so that they use only actions and reusable workflows available within that organization. When such a policy is enforced, external third-party actions and Marketplace actions can be prevented from running. Custom actions intended for internal reuse should therefore be stored in repositories owned by the organization. GitHub also allows private or internal repositories containing actions to be made accessible to other repositories in the same organization or enterprise. The .github repository has special organization-wide uses, such as workflow templates, but it is not the mandatory distribution location for all custom actions. A third-party repository or the public Marketplace would conflict with an organization-only policy. Therefore, distributing the actions through organization- owned repositories is the appropriate approach.

NEW QUESTION: 58

How can a workflow deployment mitigate the risk of multiple workflow runs deploying to a single cloud environment simultaneously?
(Each correct answer presents part of the solution. Choose two.)
A. Pass the mutex into the deployment job.
B. Configure the mutex setting in the environment.
C. Specify a concurrency scope in the workflow.
D. Specify a target environment in the deployment job.
E. Set the concurrency in the deployment job to 1.
F. Reference the mutex in the task performing the deployment.
Answer: (SHOW ANSWER)
GitHub Actions provides the concurrency mechanism for preventing overlapping workflow runs or jobs that share the same concurrency group. A workflow can define a concurrency group associated with a deployment target so that only one matching deployment is running at a time. The deployment job should also reference the target GitHub environment, such as production, so GitHub can associate the job with that deployment environment and apply its deployment controls. GitHub ' s documentation specifically describes using concurrency to ensure that an environment has a maximum of one deployment in progress at a time. GitHub does not provide the mutex configuration described in options A, B, and F, and concurrency is not configured simply by assigning the numeric value 1. Therefore, specifying the concurrency scope together with the target environment is the appropriate solution.

NEW QUESTION: 59
You are a developer working on developing reusable workflows for your organization.
What keyword should be included as part of the reusable workflow event triggers?
A. check_run
B. workflow_run
C. workflow_call
D. pull_request
Answer: (SHOW ANSWER)
The workflow_call event is used to trigger a reusable workflow from another workflow. This allows you to create workflows that can be reused in multiple places within your organization, enabling better modularity and reducing duplication.

NEW QUESTION: 60
In the following workflow file, line 5 interprets lines 3 and 4 as Python. Which of the following is a valid option to complete line 5?
1 steps:
2 - run: |
3 import os
4 print(os.environ[' PATH '])

5
A. with: python
B. shell: bash
C. working-directory: .github/python
D. shell: python
Answer: (SHOW ANSWER)
The run keyword executes commands using the default shell unless a different shell is specified. Since the commands shown are Python statements, the step must define shell: python so GitHub Actions interprets the script using Python instead of Bash, PowerShell, or another default shell. Option D is therefore correct.
Option A is invalid because with: is used to pass inputs to actions, not to define the interpreter for a run command. Option B would execute the text as Bash and fail because import os is not Bash syntax. Option C only changes the working directory and does not change the shell. This exam topic checks workflow syntax, shell selection, and step-level command execution.

NEW QUESTION: 61
What is a role of GitHub Marketplace when using GitHub Actions?
A. GitHub Marketplace restricts the use of third-party actions to only public repositories.
B. GitHub Marketplace serves only as a billing dashboard for tracking paid actions.
C. GitHub Marketplace provides a centralized location to discover, share, and publish reusable GitHub Actions workflows created by the community or vendors.
D. GitHub Marketplace allows a user to deploy GitHub Actions workflows to any repository automatically without requiring permissions.
Answer: C (LEAVE A REPLY)
GitHub Marketplace is used to discover, share, and publish reusable automation, including GitHub Actions created by GitHub, vendors, and the community. Option C best describes this role. Marketplace does not exist only for billing, so option B is too narrow and incorrect. Option A is also wrong because Marketplace does not merely restrict third-party actions to public repositories; action usage is governed by repository, organization, or enterprise Actions policies. Option D is false because Marketplace does not bypass repository permissions or automatically deploy workflows into repositories. In exam terms, Marketplace is part of the automation reuse and action discovery domain: developers use it to find trusted or verified actions rather than building every integration manually. GitHub documents Marketplace as a place to publish and share actions with the community.

NEW QUESTION: 62
You need to make a script to retrieve workflow run logs via the API. Which is the correct API to download a workflow run log?
A. POST /repos/:owner/:repo/actions/runs/:run_id
B. GET /repos/:owner/:repo/actions/artifacts/logs
C. GET /repos/:owner/:repo/actions/runs/:run_id/logs
D. POST /repos/:owner/:repo/actions/runs/:run_id/logs
Answer: C (LEAVE A REPLY)
The GET /repos/:owner/:repo/actions/runs/:run_id/logs API endpoint is used to retrieve the logs of a specific workflow run identified by run_id. This is the correct method for downloading logs from a workflow run.

NEW QUESTION: 63
Which statement is true about using default environment variables?
A. The environment variables can be read in workflows using the ENV: variable_name syntax.
B. The environment variables created should be prefixed with GITHUB_ to ensure they can be accessed in workflows
C. The environment variables can be set in the defaults: sections of the workflow
D. The GITHUB_WORKSPACE environment variable should be used to access files from within the runner.
Answer: D (LEAVE A REPLY)
GITHUB_WORKSPACE is a default environment variable in GitHub Actions that points to the directory on the runner where your repository is checked out. This variable allows you to access files within your repository during the workflow.

NEW QUESTION: 64
Scheduled workflows run on the:
A. Scheduled workflows run on the:
B. latest commit and branch on which the workflow was triggered,
C. latest commit from the branch named schedule,
D. latest commit from the branch named main,
E. specified commit and branch from the workflow YAML file,
F. latest commit on the default or base branch
Answer: A (LEAVE A REPLY)
Scheduled workflows in GitHub Actions are triggered at specified times, and they run on the latest commit of the branch that triggers the workflow. This means the workflow will run

on the most recent commit on the branch that was active at the time the scheduled event occurs.

NEW QUESTION: 65
As a developer, which of the following snippets will enable you to run the commands npm ci and npm run build as part of a workflow?
A. - run: |
npm ci
npm run build
B. - shell:
npm ci
npm run build
C. - shell: |
npm ci
npm run build
D. - run:
npm ci
npm run build
Answer: A (LEAVE A REPLY)
The run keyword is used to execute command-line programs and scripts on a GitHub Actions runner. When several commands must execute within the same workflow step, YAML ' s literal block indicator (|) can be placed after run: and the commands written on separate indented lines. GitHub ' s workflow syntax explicitly demonstrates this pattern with npm ci followed by npm run build. The shell keyword does not contain commands; it specifies which shell should interpret the commands supplied through run. Therefore, options B and C misuse the shell property. Option D also lacks the YAML literal block notation required to represent the multiple command lines as the value of run. Option A correctly executes both commands sequentially within the same shell process and is the documented syntax.