ECCouncil

312-97

EC-Council Certified DevSecOps Engineer (ECDE) Exam

For More Information – Visit link below:

https://www.examsempire.com/

Product Version

1. **Up to Date products, reliable and verified.**
2. **Questions and Answers in PDF Format.**

Latest Version: 6.0

Question: 1

(William Scott, after completing his graduation in computer science, joined an IT company as a DevSecOps engineer. His team leader has asked him to use GitHub Code Scanning for evaluating the source code in his organization's GitHub repository to detect security issues and coding errors. How can William set up coding scanning in GitHub repository?)

A. By using Gauntlt.
B. By using GitMiner.
C. By using OWASP ZAP.
D. By using CodeQL.

Answer: D

Explanation:
GitHub Code Scanning is a built-in security capability designed to automatically analyze source code for security vulnerabilities and coding errors. The primary and officially supported engine for GitHub Code Scanning is CodeQL. CodeQL works by converting source code into a database and running security queries to detect issues such as injection flaws, insecure coding patterns, and logic errors. William can enable CodeQL by configuring GitHub Actions with either the default or advanced CodeQL workflow. Once enabled, CodeQL scans are triggered on events such as code pushes and pull requests, and the results appear as code scanning alerts in the repository's Security tab. Gauntlt is a security testing harness used mainly for infrastructure and application testing, GitMiner is used to discover sensitive data like secrets in repositories, and OWASP ZAP is a dynamic application security testing tool used against running applications. None of these tools configure GitHub's native Code Scanning feature. Therefore, CodeQL is the correct tool to set up GitHub Code Scanning in the Code stage of a DevSecOps pipeline.

Question: 2

(Brett Ryan has been working as a senior DevSecOps engineer in an IT company in Charleston, South Carolina. He is using git-mutimail tool to send email notification for every push to git repository. By default, the
tool will send one output email providing details about the reference change and one output email for every new commit due to a reference change. How can Brett ensure that git-multimail is set up appropriately?)

A. Running the environmental variable GITHUB_MULTIMAIL_CHECK_SETUP by setting it to non-empty string.
B. Running the environmental variable GIT_MULTIMAIL_CHECK_SETUP by setting it to empty string.
C. Running the environmental variable GIT_MULTIMAIL_CHECK_SETUP by setting it to non-empty string.

D. Running the environmental variable GITHUB_MULTIMAIL_CHECK_SETUP by setting it to empty string.

Answer: C

Explanation:
The git-multimail tool provides a mechanism to verify whether it has been installed and configured correctly before being relied upon for production notifications. This verification is done using an environment variable named GIT_MULTIMAIL_CHECK_SETUP. When this variable is set to a non-empty string, git-multimail performs a setup validation and outputs diagnostic information to confirm that configuration values, hooks, and parameters are correctly defined. This helps prevent silent failures where commits occur but email notifications are not sent. Options that reference GITHUB_MULTIMAIL_CHECK_SETUP are incorrect because git-multimail is not limited to GitHub and does not use that variable name. Additionally, setting the variable to an empty string does not trigger the setup check. Ensuring proper configuration during the Code stage is important because it supports auditability, traceability, and timely communication among development and security teams. Therefore, Brett must run the environment variable GIT_MULTIMAIL_CHECK_SETUP with a non-empty value to ensure the tool is set up appropriately.

Question: 3

(BVR Pvt. Ltd. is an IT company that develops software products and applications related to IoT devices. The software development team of the organization is using Bitbucket repository to plan projects, collaborate on code, test, and deploy. The repository provides teams a single place for projects planning and collaboration on coding, testing, and deploying the software application. Which of the following is offered by Bitbucket to BVR Pvt. Ltd.?)

A. Free limited public repositories.
B. Free unlimited private repositories.
C. Free limited private repositories.
D. Free unlimited public repositories.

Answer: B

Explanation:
Bitbucket provides a cloud-based source code management platform that supports collaboration, CI/CD integration, and secure code hosting. One of the key features offered by Bitbucket is free unlimited private repositories, particularly beneficial for organizations developing proprietary software such as IoT applications. This allows teams to store source code securely without exposing it publicly while still enabling collaboration features like pull requests, issue tracking, and pipeline automation. The term “limited private repositories” is inaccurate because Bitbucket does not restrict the number of private repositories under its free offering; rather, user count limits apply. While Bitbucket also supports public repositories, the option that best represents its value to enterprise and product-based teams is unlimited private repositories. This capability aligns with DevSecOps practices by ensuring confidentiality of source code while enabling integrated planning, testing, and deployment workflows within a single platform.

Question: 4

(Erica Mena has been working as a DevSecOps engineer in an IT company that provides customize software solutions to various clients across United States. To protect serverless and container applications with RASP, she would like to create an Azure container instance using Azure CLI in Microsoft PowerShell. She created the Azure container instance and loaded the container image to it. She then reviewed the deployment of the container instance. Which of the following commands should Erica run to get the logging information from the Azure container instance? (Assume the resource group name as ACI and container name as aci-test-closh.))

A. az get container logs -resource-group ACI --name aci-test-closh.
B. az get container logs --resource-group ACI --name aci-test-closh.
C. az container logs -resource-group ACI -name aci-test-closh.
D. az container logs --resource-group ACI --name aci-test-closh.

Answer: D

Explanation:
Azure Container Instances provide built-in logging capabilities that can be accessed using the Azure CLI. To retrieve logs from a deployed container instance, the correct command is az container logs followed by the resource group and container name. The proper syntax requires double-dash parameters: --resource-group and --name. In Erica's case, the correct command is az container logs --resource-group ACI --name aci-test-closh. Options that use "az get container logs" are invalid because "get" is not a supported verb in this context. Option C uses incorrect single-dash flags, which do not match Azure CLI standards. Accessing container logs during the Code stage helps engineers validate application behavior, identify runtime errors, and ensure that security instrumentation such as RASP agents are functioning correctly before progressing further in the pipeline.

Question: 5

(Walter O'Brien recently joined as a junior DevSecOps engineer in an IT company located in Lansing, Michigan. His organization develops robotic process automation software for various clients stretched across the globe. Walter's team leader asked him to configure username and user email for git in VS Code. Therefore, he opened Visual Studio Code IDE console, then clicked on Terminal tab and selected New terminal. Which of the following command should Walter execute in the terminal to configure username and user email for git in VS Code?)

A. get config --global user-name "walter username for git"
get config -–global user-email "walter email address used for git".
B. get config --global user.name "walter username for git"
get config –global user.email "walter email address used for git".
C. get git config --global user.name "walter username for git"
get git config –global user.email "walter email address used for git".

D. get config --global user_name “walter username for git”
get config -–global user_email “walter email address used for git”.

Answer: B

Explanation:
Git requires developers to configure their identity using two specific configuration keys: user.name and user.email. These values are embedded into every commit and are essential for accountability, auditing, and collaboration. The correct configuration syntax uses dot-separated key names (user.name and user.email) and the --global flag to apply the settings across all repositories on the system. Among the provided options, only option B uses the correct configuration keys. The other options use invalid key names such as user-name, user_name, or incorrect command structure. Although the options display a minor command typo (“get config” instead of git config), the question is clearly testing knowledge of the correct Git configuration keys. Configuring Git identity in the Code stage ensures accurate commit history and supports traceability across the DevSecOps pipeline.

Thank You for Trying Our Product

Special 16 USD Discount Coupon: NSZUBG3X

Email: support@examsempire.com

Check our Customer Testimonials and ratings available on every product page.

Visit our website.

https://examsempire.com/