

# Preventing exposure of IBM Cloud credentials in public repositories and platforms

## Overview

Hackathon participants may use IBM Cloud credentials to build their solutions that leverage IBM technologies on IBM Cloud. During development, testing, collaboration, or while submitting their final project in a public repository, participants may unintentionally expose these credentials on publicly accessible platforms.

Exposing IBM Cloud API keys or any cloud credentials can lead to unauthorized access, misuse of resources, and account suspension. This guide outlines best practices, tools, and secure workflows to help you prevent credential exposure while still meeting the requirement to submit public code.

## Contents

|                                                                                                                         |   |
|-------------------------------------------------------------------------------------------------------------------------|---|
| 1. Use environment variables .....                                                                                      | 1 |
| 2. Use a ".env" file (and never commit it) .....                                                                        | 2 |
| 3. Use public code hosting platforms (e.g., GitHub, GitLab, Bitbucket, or similar) Secrets (for Actions or CI/CD) ..... | 2 |
| 4. Use IBM Cloud service credentials with limited scope .....                                                           | 2 |
| 5. Scan before commit with secret detection tools .....                                                                 | 2 |
| 6. Review commits carefully .....                                                                                       | 3 |
| 7. If you accidentally commit a credential .....                                                                        | 3 |
| 8. Use sample or placeholder keys in public code .....                                                                  | 3 |
| 9. Do not share screenshots containing credentials .....                                                                | 3 |
| 10. Do not share credentials on chat tools or public forums .....                                                       | 4 |

### 1. Use environment variables

- Store credentials in environment variables on your local machine or runtime environment.
- Access them in your code via standard environment variable methods.
- Never hardcode sensitive information directly in source files.

Example pattern:

```
export IBM_API_KEY="your-api-key-here"
```

Then in code:

```
api_key = os.getenv("IBM_API_KEY")
```

---

## 2. Use a “.env” file (and never commit it)

- Place sensitive configuration values in a “.env” file.
- Ensure “.env” is listed in “.gitignore” file.
- If collaborating, share the “.env” file through secure channels, not via public code hosting platforms (e.g., GitHub, GitLab, Bitbucket, or similar).

Example “.gitignore” entry:

```
.env
*.env
```

---

## 3. Use public code hosting platforms (e.g., GitHub, GitLab, Bitbucket, or similar) Secrets (for Actions or CI/CD)

- If using public code hosting platforms (e.g., GitHub, GitLab, Bitbucket, or similar) Actions for deployment/testing, store credentials only in public code hosting platforms (e.g., GitHub, GitLab, Bitbucket, or similar) Secrets.
- Reference secrets inside workflow files instead of storing them directly.

---

## 4. Use IBM Cloud service credentials with limited scope

- Generate credentials with the minimal privileges needed.
- Prefer service-specific credentials instead of account-wide keys.
- Regularly rotate keys.

---

## 5. Scan before commit with secret detection tools

Use automated tools locally to detect secrets before they reach public code hosting platforms (e.g., GitHub, GitLab, Bitbucket, or similar). Examples:

- public code hosting platforms (e.g., GitHub, GitLab, Bitbucket, or similar) secret scanning (built-in for public repos)
- Git hooks using tools like “pre-commit” with secret scanners
- Tools like “trufflehog”, “git-secrets”, or “detect-secrets”

Set up a **pre-commit hook** to block commits containing secrets.

## 6. Review commits carefully

- Check “*git diff*” for any accidental key insertions.
  - Use “*git log*” to ensure no past commits contain leaked credentials.
  - If a secret is ever committed—even once—**assume it is compromised**.
- 

## 7. If you accidentally commit a credential

1. **Immediately delete/rotate** the credential in IBM Cloud.
  2. Remove it from Git history permanently using:
    - a. “*git filter-repo*”, or
    - b. “*git filter-branch*”, or
    - c. public code hosting platforms (e.g., GitHub, GitLab, Bitbucket, or similar)'s built-in secret removal tools.
  3. Re-scan your repository.
  4. Commit and push the cleaned history.
- 

## 8. Use sample or placeholder keys in public code

- In demo code, always use placeholder text:
    - YOUR\_IBM\_API\_KEY\_HERE
    - REPLACE\_ME
  - Store the real key in environment variables.
- 

## 9. Do not share screenshots containing credentials

- Screenshots of dashboards, CLI, logs, or environment setup should never show real keys.
  - Mask or blur sensitive information.
-

## **10. Do not share credentials on chat tools or public forums**

- Even internal chat messages can be forwarded.
- Use secure vaults or encrypted channels if you must transmit a key