Lab 8: Deploy with Ansible (on GitHub Runner) and Host on GitHub Pages

Lab Overview

In this lab, you will trigger Ansible via GitHub Actions. You will learn to use ansible-playbook inside a GitHub Actions runner (local connection), manage an SSH private key as a GitHub Secret (best practice, even if unused here) and debug failed CI/CD deployments with Ansible + Actions logs.

In this lab, you will:

- Run Ansible directly on the GitHub-hosted runner
- Build a static site into ./site
- Upload the site as an artifact and deploy it to GitHub Pages

Estimated completion time

50 minutes

Commented [IA1]: There are some things in the lab, like GitHub Pages deployment, SSH secret setup, Ansible `lookup()` functions, artifact upload between jobs, and workflow permissions, which are not explicitly taught in the slides.

Commented [WA2R1]: Thank you for the valuable feedback.

To address this, have added a new Task 0: Background Knowledge section at the start of the lab. This section introduces each of these concepts in detail before students begin the practical steps.

Task 0: Background Knowledge

Before continuing, here are some important concepts used in this lab:

1. GitHub Pages Deployment
   - GitHub Pages hosts static websites directly from your repository.
   - In this lab, the static site built with Ansible is uploaded as an artifact and then deployed using the actions/deploy-pages@v4 action.
   - The workflow needs explicit permissions (pages: write and id-token: write) to publish content to Pages.
2. Ansible lookup() Function
   - The lookup() function allows Ansible playbooks to fetch external information dynamically.
   - Examples from this lab:
   - lookup('env','SITE_DIR') → reads an environment variable for the output folder.
   - lookup('pipe','date -u …') → runs a shell command to insert the current UTC timestamp into the HTML page.
   - This makes playbooks more flexible and environment-aware.
3. SSH Keys and GitHub Secrets
   - Ansible normally connects to remote servers via SSH keys for secure, passwordless authentication.
   - Even though we are running on the local GitHub runner, we practice creating an SSH key and saving it as a GitHub Secret (SSH_PRIVATE_KEY).
   - This reflects real-world best practice for managing sensitive credentials safely.
4. Artifacts in GitHub Actions
   - GitHub Actions jobs run on isolated virtual machines, so files created in one job aren't automatically available to another.
   - Artifacts solve this by packaging and transferring files between jobs.
   - In this lab, the site/ directory is uploaded in the build job and retrieved by the deploy job for publishing.
5. Workflow Permissions
   - By default, GitHub Actions workflows have minimal permissions.
   - We explicitly declare only the permissions required:
   - contents: read → to fetch the repository code.
   - pages: write → to publish to GitHub Pages.
   - id-token: write → to authenticate with GitHub Pages securely using OpenID Connect (OIDC).

- This follows the principle of least privilege, a key security best practice

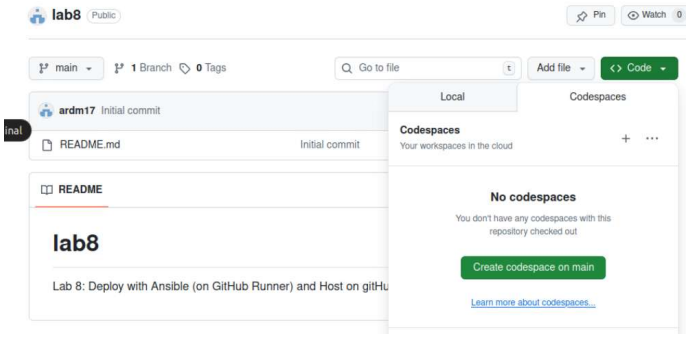
Task 1: Create the Repository

Create GitHub repository:

- Go to GitHub.com and sign in
- Click **New repository** (green button)
- Repository name: lab8
- Description: Lab 8: Deploy with Ansible (on GitHub Runner) and Host on GitHub Pages
- Set to **Public** (for easier access to Actions)
- Check **Add a README file**
- Click **Create repository**

Task 2: Open in Codespaces

1. On your repo page, click **Code → Codespaces → Create codespace on main**.



2. Wait for the VS Code browser environment to load.

Task 3: Create Project Structure and Ansible Files

In the **Codespaces terminal:**

Create required folders

```
mkdir -p ansible .github/workflows
```

Inventory: local target

```
cat > ansible/inventory.ini <<'INI'
[web]
localhost ansible_connection=local
INI
```

Playbook: build static site

Note: Using cat > for such a large file can result in errors. For the lab, firstly, add the site.yml file directly in Codepsaces Explorer (ansible folder).

Then, open the Desktop > Sample Lab Files > Lab 8 folder, copy and paste the contents of the site.yml file into the Codespaces file. Verify the resulting file contents to be:

```
cat > ansible/site.yml <<'YAML'
---
- name: Build static site locally on runner
  hosts: web
  gather_facts: false

  vars:
    # Use SITE_DIR from env when provided; default to ./site
    site_dir: "{{ lookup('env','SITE_DIR') | default('./site', true) }}"

  tasks:
    - name: Ensure site directory exists
      file:
```

```
    path: "{{ site_dir }}"
    state: directory
    mode: "0755"

- name: Create index.html
  copy:
    dest: "{{ site_dir }}/index.html"
    mode: "0644"
    content: |
      <!doctype html>
      <html>
        <head><meta charset="utf-8"><title>GitHub Pages via Ansible</title></head>
        <body style="font-family: system-ui; margin:2rem">
          <h1>☑ Deployed with Ansible on a GitHub Runner</h1>
          <p>Hosting: GitHub Pages</p>
          <p>Build time (UTC): {{ lookup('pipe','date -u +%Y-%m-%dT%H:%M:%SZ') }}</p>
        </body>
        </html>
```

Commit changes in the terminal pane:

```
git add ansible/inventory.ini ansible/site.yml
git commit -m "Lab8: add inventory + playbook"
git push
```

Task 4: Manage an SSH Private Key as a GitHub Secret

We're storing the SSH key only for practice. In real-world deployments (to VMs/servers), this secret would be used by Ansible. Even though we don't need SSH here, we'll create and store a key for practice.

1. Generate a key (inside Codespaces)

```
ssh-keygen -t ed25519 -C "gh-actions-ansible" -f gh_actions_ansible_ed25519 -N ""
```

This creates:

gh_actions_ansible_ed25519 (private key)

gh_actions_ansible_ed25519.pub (public key)

2. Add the private key to GitHub Secrets

```
cat gh_actions_ansible_ed25519
```

3. On GitHub, go to:

Repo → Settings → Secrets and variables → Actions → New repository secret

4. Name: SSH_PRIVATE_KEY

5. Paste the private key, save.

(Later if we target a real VM will use this secret for deployment.)

Task 5: Enable GitHub Pages

1. Go to Repo → Settings → Pages.

2. Under Build and Deployment, set Source = GitHub Actions.

Commented [IA3]: The SSH key is not used in this lab. This could confuse learners since it's introduced but not applied. Suggest explicitly clarifying:

"We're storing the SSH key only for practice. In real-world deployments (to VMs/servers), this secret would be used by Ansible."

Commented [WA4R3]: Done

Task 6: Create the GitHub Actions Workflow

In Codespaces terminal:

Note: Using cat > for such a large file can result in errors. For the lab, firstly, add the deploy.yml file directly in Codepsaces Explorer (.github/workflows folder).

Then, open the Desktop > Sample Lab Files > Lab 8 folder, copy and paste the contents of the deploy.yml file into the Codespaces file. Verify the resulting file contents to be:

```
cat > .github/workflows/deploy.yml <<'YAML'

name: Build with Ansible and Deploy to GitHub Pages

on:
  push:
    branches: [ "main" ]
    paths:
      - "ansible/**"
      - ".github/workflows/deploy.yml"
  workflow_dispatch:

permissions:
  contents: read
  pages: write
  id-token: write

concurrency:
  group: "pages"
  cancel-in-progress: true

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
```

```
    - name: Checkout
      uses: actions/checkout@v4

    - name: Install Ansible
      run: |
        sudo apt-get update
        sudo apt-get install -y ansible

    # Dummy step: securely load SSH key (not used here)
    - name: Load SSH key from secret
      run: |
        mkdir -p ~/.ssh
        echo "${{ secrets.SSH_PRIVATE_KEY }}" > ~/.ssh/id_ed25519
        chmod 600 ~/.ssh/id_ed25519
      shell: bash

    # Run ansible-playbook manually
    - name: Build site with Ansible (local)
      env:
        SITE_DIR: ${{ github.workspace }}/site
      run: |
        ansible-playbook -i ansible/inventory.ini ansible/site.yml -vv

    # Debug check
    - name: "Debug: list ./site"
      run: ls -la ${{ github.workspace }}/site

    - name: Setup Pages
      uses: actions/configure-pages@v5

    - name: Upload site artifact
      uses: actions/upload-pages-artifact@v3
      with:
```

```
      path: ./site

  deploy:
    environment:
      name: github-pages
    runs-on: ubuntu-latest
    needs: build
    steps:
      - name: Deploy to GitHub Pages
        id: deployment
        uses: actions/deploy-pages@v4
```

Commit changes in the terminal pane:

```
git add .github/workflows/deploy.yml
git commit -m "Lab8: final working workflow with ansible-playbook + Pages"
git push
```

Task 7: Trigger and Verify

1. From GitHub repo, go to **Actions → Build with Ansible and Deploy to GitHub Pages → Run workflow.**

2. Wait until the workflow completes.

3. Open **Settings → Pages** → copy the site URL:
https://<username>.github.io/<repo-name>/

4. Open in browser — you should see:
"✅ Deployed with Ansible on a GitHub Runner"

Task 8: Debugging Failures

1. **Ansible step fails**
   - Check for YAML indentation errors in ansible/site.yml.
   - Ensure **inventory.ini** contains:

     ```
     [web]
     localhost ansible_connection=local
     ```

2. **Upload artifact fails**
   - Check the Debug step; if ./site is missing, Ansible didn't run correctly.

3. **Pages deploy fails**
   - Ensure **Pages** is enabled in repo settings.
   - Confirm workflow has **pages: write** and **id-token: write** permissions.

Increase verbosity by changing -vv → -vvv in the workflow's Ansible step.

Lab review

Why do we use localhost ansible_connection=local in the inventory file?

A. To connect to a Docker container
B. To make Ansible run tasks directly on the GitHub runner
C. To skip running Ansible and use defaults
D. To enable multi-node cluster deployment

Correct Answer: B. To make Ansible run tasks directly on the GitHub runner

STOP

You have successfully completed this lab.