DevOps with GitHub Actions – Lesson 2

Facilitated by Kent State University Topic: SSH-Based Deployment with GitHub Actions Duration: 1 Hour

---

Learning Objectives

By the end of this lesson, participants will be able to:

- Use SSH keys stored in GitHub Secrets for secure deployments
- Execute remote commands via SSH from GitHub Actions
- Transfer files using SCP or rsync within workflows
- Deploy a Flask application running on a non-privileged port
- Configure Nginx to proxy to the application

I. Overview of SSH Deployments (10 minutes)

SSH is the simplest and most universal way to deploy code to a remote Linux server. With GitHub Actions, we can automate this by:

- Storing the SSH private key in GitHub Secrets
- Loading the key during a workflow
- Running `ssh` and `scp` commands from the runner

Deployment pattern:

1. Build or package the code
2. Copy artifacts to the server
3. Restart the application service
4. Validate the deployment

II. Server Setup Requirements (5 minutes)

On the remote server:

- A user account with SSH access
- Python 3.x installed
- A virtual environment (optional)
- A non-privileged port, e.g., 8080
- Nginx installed and configured as a reverse proxy

Example Nginx site configuration:

```
server {
    listen 80;
    location / {
```

```
        proxy_pass http://127.0.0.1:8080;
    }
}
```

III. GitHub Secrets for SSH (10 minutes)

Set these Secrets in the GitHub repository:

- SSH_PRIVATE_KEY
- SSH_HOST
- SSH_USER

Workflow snippet to load the key:

```
- name: Configure SSH
  run: |
    mkdir -p ~/.ssh
    echo "${{ secrets.SSH_PRIVATE_KEY }}" > ~/.ssh/id_rsa
    chmod 600 ~/.ssh/id_rsa
```

IV. File Transfer Using SCP (10 minutes)

Example workflow step:

```
- name: Copy files to server
  run: scp -o StrictHostKeyChecking=no -r ./dist/app "${{ secrets.SSH_USER
}}@${{ secrets.SSH_HOST }}:/opt/myapp/"
```

Alternative using rsync:

```
- name: Sync files
  run: rsync -avz ./dist/app/ "${{ secrets.SSH_USER }}@${{
secrets.SSH_HOST }}:/opt/myapp/"
```

V. Running Remote Commands (10 minutes)

Example step:

```
- name: Restart remote app
  run: ssh -o StrictHostKeyChecking=no "${{ secrets.SSH_USER }}@${{
```

```
secrets.SSH_HOST }}" "pkill -f myapp || true && nohup python3
/opt/myapp/app.py --port 8080 &"
```

Example Flask app invocation:

```
nohup python3 -m flask run --host=0.0.0.0 --port=8080 &
```

VI. Full Deployment Workflow Example (5 minutes)

```
name: SSH Deploy
on:
  workflow_dispatch:
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Configure SSH key
        run: |
          mkdir -p ~/.ssh
          echo "${{ secrets.SSH_PRIVATE_KEY }}" > ~/.ssh/id_rsa
          chmod 600 ~/.ssh/id_rsa

      - name: Copy project files
        run: scp -o StrictHostKeyChecking=no -r . "${{ secrets.SSH_USER
}}@${{ secrets.SSH_HOST }}:/opt/myapp/"

      - name: Restart remote app
        run: |
          ssh -o StrictHostKeyChecking=no "${{ secrets.SSH_USER }}@${{
secrets.SSH_HOST }}" "pkill -f flask || true && nohup python3 -m flask run
--host=0.0.0.0 --port=8080 &"
```

VII. Troubleshooting Common Issues (5 minutes)

- Wrong SSH key format → must be **private key only**, no extra whitespace
- Missing dependencies on server → install Python packages before launch
- Flask binding to 127.0.0.1 → must bind to 0.0.0.0 for external access
- Conflicting processes → ensure previous version is stopped before restart

---

VIII. Exercise (5 minutes)

Create a workflow that:

1. Copies a packaged Flask app to a server
2. Installs dependencies remotely
3. Restarts the service

IX. Final Question

Which command is used to securely copy files to a remote server?

A. ssh B. cp C. scp D. wget

Answer: C. scp