

Lab: CI/CD Using AWX

Introduction:

Ansible Automation Platform automates the major stages of continuous integration, delivery, and deployment (CI/CD) pipelines—becoming the activating tool of DevOps methodologies.

CI/CD allows organizations to ship software quickly and efficiently.

Objectives:

- Adding webhooks
- Creating a template
- Launch the AWX template on the AWX UI.

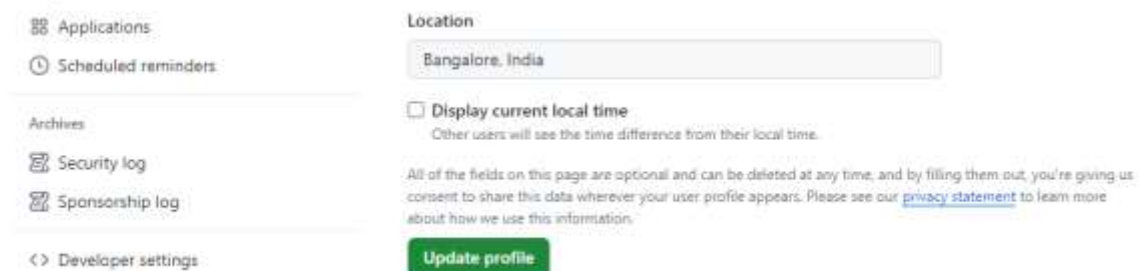
Note: Login to **eoc-controller** as **admin** user with password as **linux**

1. Adding webhooks.

1.1 In the profile settings of your GitHub account, click Settings.

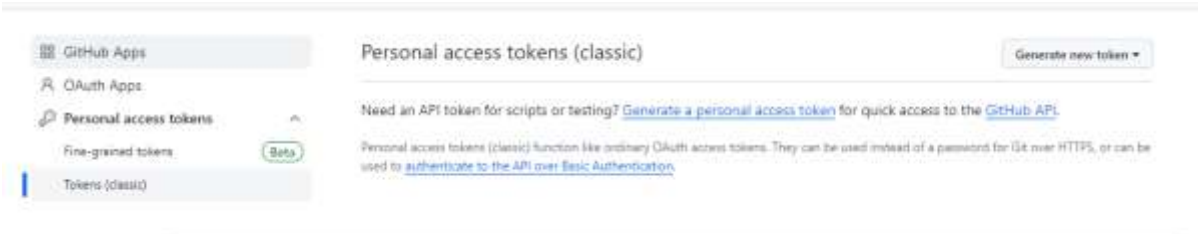
1.2 In the left pane click on developer settings.

Output:

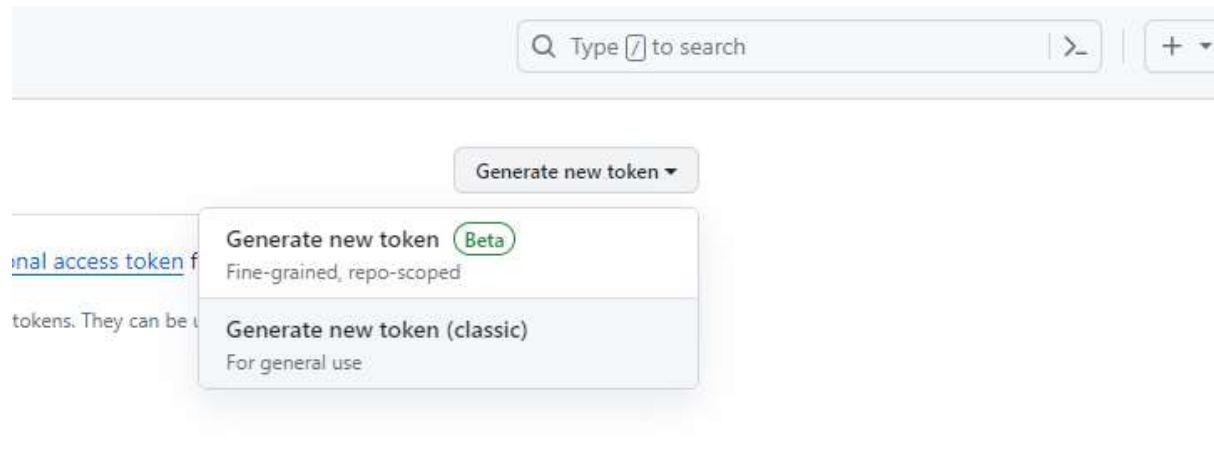


1.3 Click on Personal access token, select token (classic) and select the generate token button.

Output:



1.4 Select generate new classic token.

Output:

1.5 Select the required permissions and click on generate token.

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

github webhook

What's this token for?

Expiration *

30 days

The token will expire on Sun, Dec 24 2023

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
.....	
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys
<input type="checkbox"/> admin:ssh_signing_key	Full control of public user SSH signing keys
<input type="checkbox"/> write:ssh_signing_key	Write public user SSH signing keys
<input type="checkbox"/> read:ssh_signing_key	Read public user SSH signing keys

Generate token

Cancel

1.6 Copy the token to reuse it.

Output:

Personal access tokens (classic) Generate new token ▼ Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓

□

Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

1.7 In the awx create a new credential.

The screenshot shows the 'Create New Credential' form in the AWX web interface. On the left is a dark sidebar with navigation links: Views (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), Resources (Templates, Credentials, Projects, Inventories, Hosts), and Access (Organizations). The main form area has the title 'Create New Credential'. It contains the following fields: 'Name' with the value 'github PAM', an empty 'Description' field, and an 'Organization' dropdown set to 'Default'. Below these is the 'Credential Type' dropdown, which is set to 'GitHub Personal Access Token'. Under the 'Type Details' section, there is a 'Token' field with a masked value and a copy icon. At the bottom of the form are 'Save' and 'Cancel' buttons.

Note: Paste the generated token in the token box.

2. Creating a template

2.1 Create a new template using the same inventory and project.

Templates

Create New Job Template

Name * **Description**

Job Type * ☐ Prompt on launch

Inventory * ☐ Prompt on launch **Project ***

Execution Environment ☐ Prompt on launch

Playbook *

Credentials ☐ Prompt on launch

Labels

Variables ☐ Prompt on launch

Options

☐ Privilege Escalation ☐ Provisioning Callbacks ☒ Enable Webhook ☐ Concurrent Jobs

☐ Enable Fact Storage ☐ Prevent Instance Group Fallback

Webhook details

Webhook Service **Webhook URL** **Webhook Key**

Webhook Credential

Save **Cancel**

Note: Add the credential that we created.

Save and open the template again to get the **URL** and **KEY**

2.2 Open the template in edit mode and copy the url and key

Webhook details

Webhook Service ⓘ

Webhook URL ⓘ

Webhook Key ⓘ

GitHub

http://192.168.100.150:31278/api/v2...

UXWDIG3nyNzsS79PEURX...

Webhook Credential ⓘ

github PAM

Save

Cancel

2.3 Go back to your github repo and repo settings.

Note: In the left pane find the webhooks ops and open it

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

General

Repository name

ansible-2023-yaml

Rename

☐ Template repository
 Template repositories let users generate new repositories with the same directory structure and

☐ Require contributors to sign off on web-based commits
 Enabling this setting will require contributors to sign off on commits made through GitHub's we contributors to affirm that their commit complies with the repository's terms, commonly the [De more about signing off on commits.](#)

Default branch

The default branch is considered the "base" branch in your repository, against which : automatically made, unless you specify a different branch.

main

2.4 Select add webhook.

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

2.5 Add the URL and SECRET KEY that we got from the template.

Set the content type to json.

Select the send me everything option.

And select add webhook option.

Output:

Webhooks / Manage webhook

Settings
Recent Deliveries

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

http://sslcontroller1.southindia.cloudapp.azure.com:31347/api/v2/jc

Content type

application/json

Secret

GnM0hmDWXnCzcOJj9Ys7Wc4vOXhFDx5qG8j38fPSOI5ROeZdNN Cancel

Which events would you like to trigger this webhook?

☐ Just the push event.
☒ Send me everything.
☐ Let me select individual events.

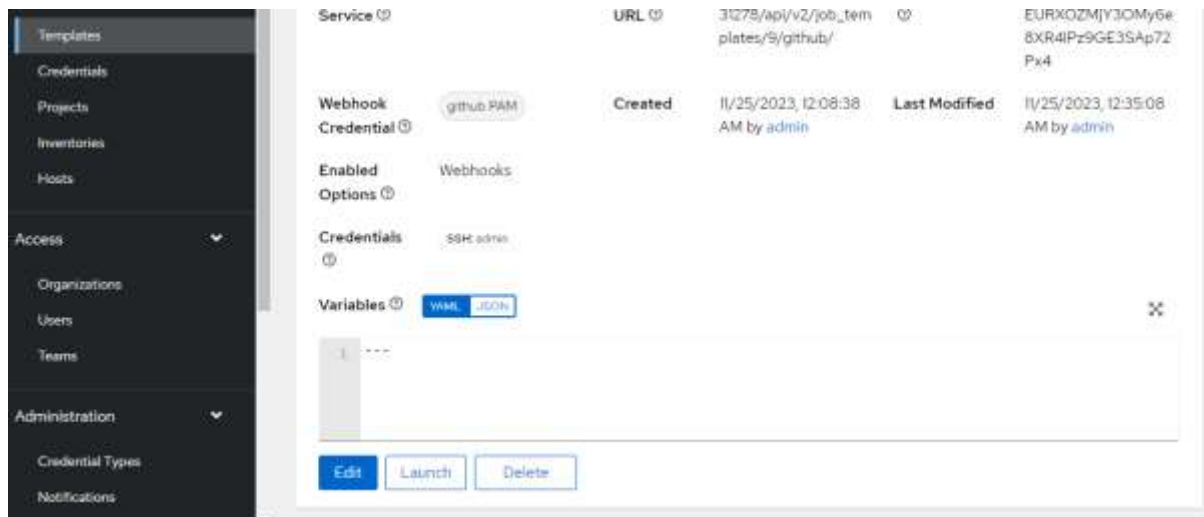
☒ **Active**
We will deliver event details when this hook is triggered.

Update webhook
Delete webhook

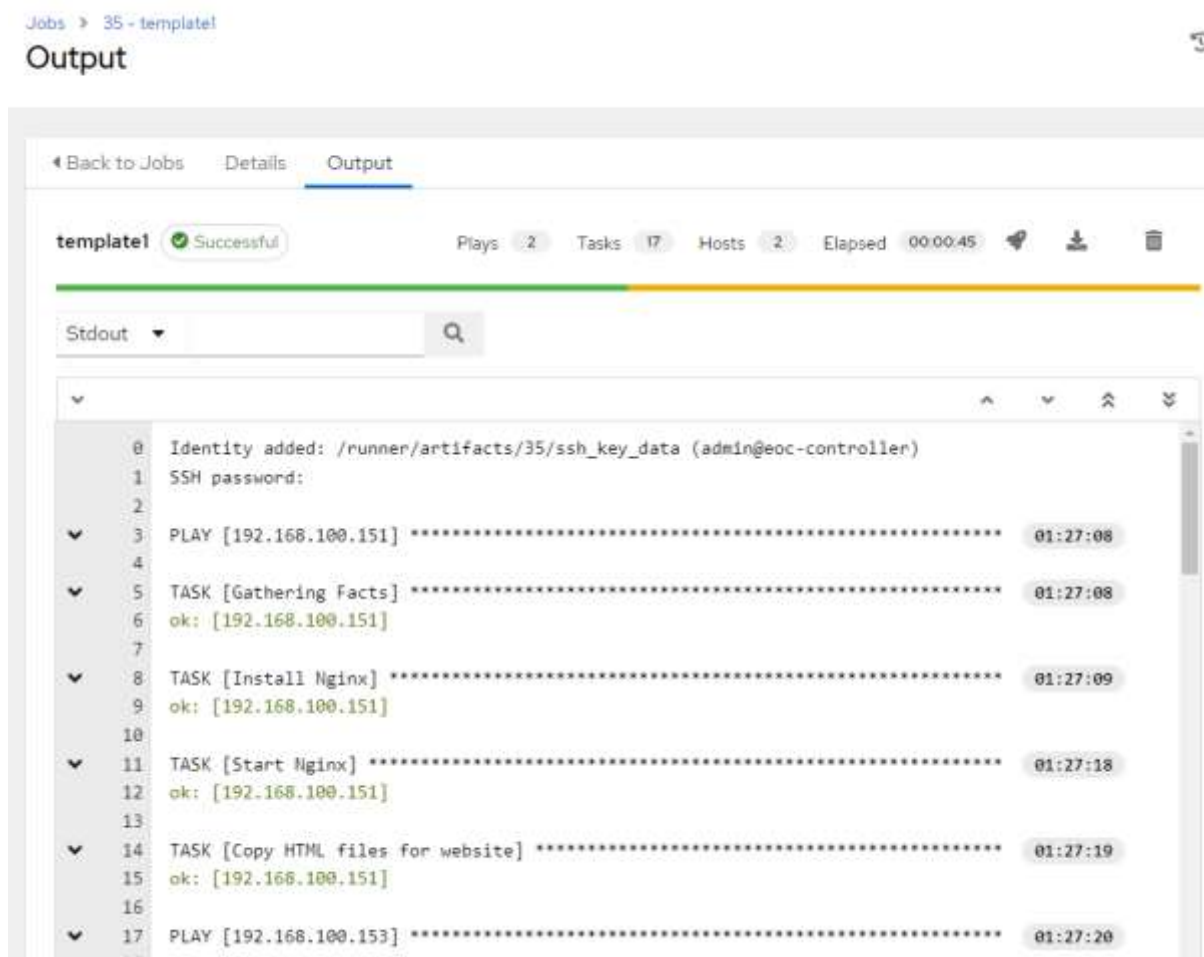
2.6 Webhook check and pushes the task into the AWX

3. Launch the awx template on the AWX UI.

3.1 Launch the template.

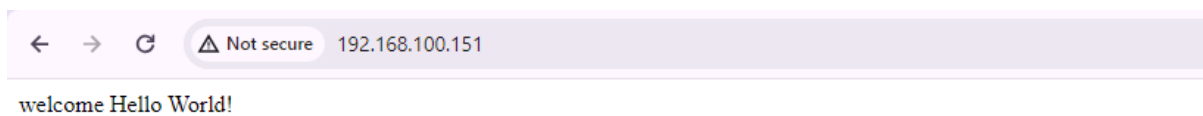
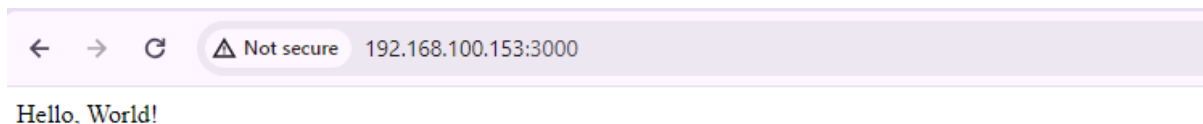


3.2 Verify the output.



55	TASK [Start Node.js app on custom IP and port] *****	01:27:46
56	changed: [192.168.100.153]	
57		
58	TASK [Restart Nginx] *****	01:27:47
59	changed: [192.168.100.153]	
60		
61	PLAY RECAP *****	01:27:47
62	192.168.100.151 : ok=4 changed=0 unreachable=0 failed=0 skipped=0 res	
	cued=0 ignored=0	
63	192.168.100.153 : ok=13 changed=7 unreachable=0 failed=0 skipped=0 res	
	cued=0 ignored=1	

3.3 Verify our webpages.



3.4 Now change the **index.html** file in the github.



Note: Once the webhook is triggered the job will be running again.

3.5 When the job is finished verify the output by refreshing the page.



3.6 Now you can also change the output of the play 2 in the task copy node.js app code to the server.


```
55     - name: Copy Node.js app code to the server
56     copy:
57       content: |
58         // app.js
59         const express = require('express');
60         const bodyParser = require('body-parser');
61
62         const app = express();
63         const port = 3000;
64
65         // Parse incoming requests with JSON payloads
66         app.use(bodyParser.json());
67
68         // Respond to GET requests with "Hello, World!"
69         app.get('/', (req, res) => {
70           res.send('Hello, World! again');
71         });
72
73         // Listen on port 3000
74         app.listen(port, () => {
75           console.log(`Server is running on http://192.168.100.153:${port}`);
76         });
77       dest: nodeapp/app.js
78       mode: 0644
```

3.7 Again, the job is launched and you can see the output:

