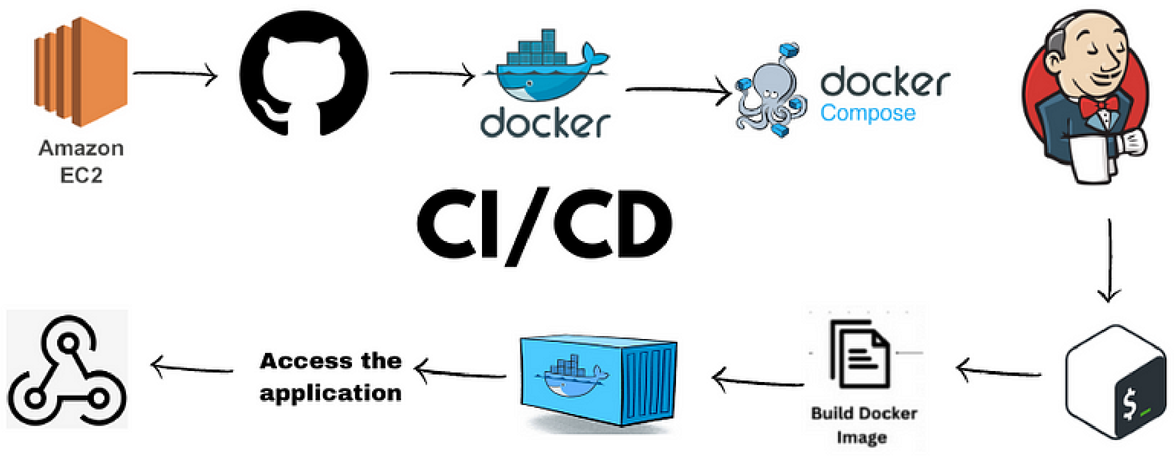
Complete Jenkins CI/CD Project with nodsjs application deployment



Today we will deploy a node.js application on EC2 instance and we will create a CICD pipeline using Jenkins.

Tools we will be using in the project:

1. AWS-EC2
2. GitHub
3. Docker
4. Jenkins

What is the Node.js Todo App?

The Node.js todo app that I will use for this tutorial is a simple todo app that allows users to create, read, update, and delete tasks. The app uses Express as the web framework and EJS as the templating engine.

The source code for the app is available on GitHub: https://github.com/ajitfawade/node-todo-cicd.git

The app has two main files:

- `app.js`: The main entry point of the app that sets up the Express server, the routes, and the views.
- The app also has a `package.json` file that defines the dependencies and scripts for the app.

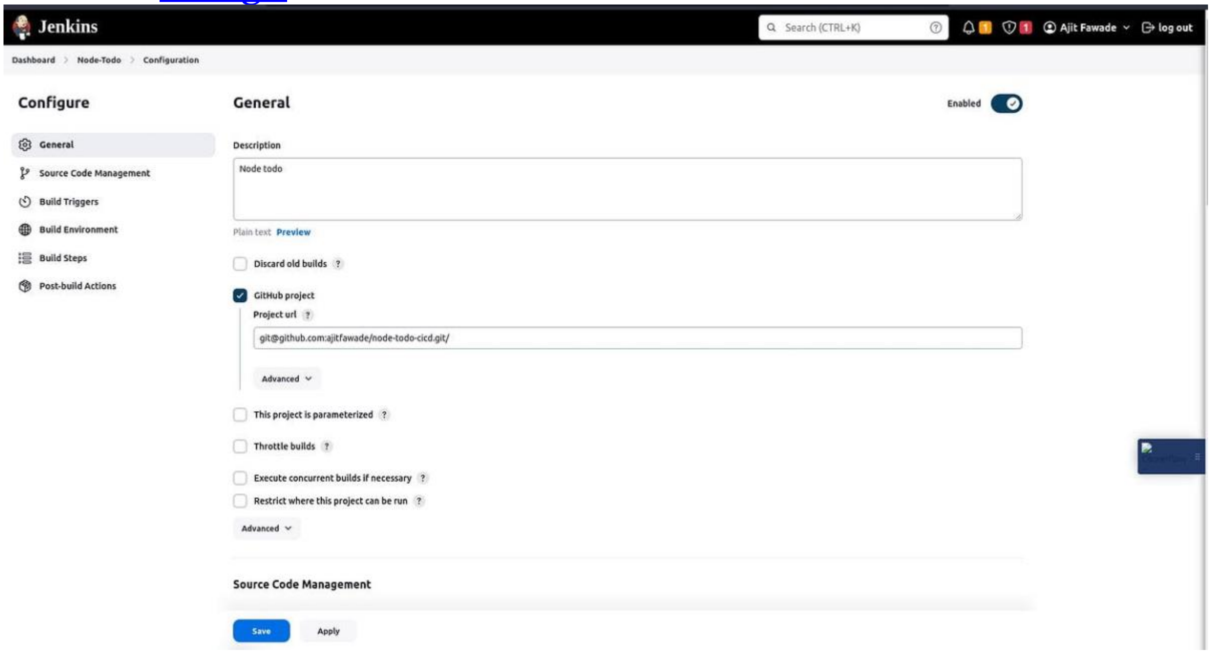
How to Connect Jenkins Job and GitHub Repository via GitHub Integration?

To deploy the app using Jenkins, we need to create a Jenkins job that connects to the GitHub repository where the app code is stored. We also need to add a GitHub webhook that triggers the Jenkins job whenever there is a push event on the repository.

To do this, follow these steps:

1. Go to your Jenkins dashboard and click on New Item.

2. Enter a name for your job (e.g., Node Todo App) and select Freestyle project. Then click OK.
3. On the configuration page, you can add some description for your job if you want.
4. Under Source Code Management, select Git and enter the repository URL: https://github.com/ajitfawade/node-todo-cicd.git. You can leave the other fields as default.



General

Description: Node todo

☐ Discard old builds

☑ GitHub project

Project url: git@github.com:ajitfawade/node-todo-cicd.git/

☐ This project is parameterized

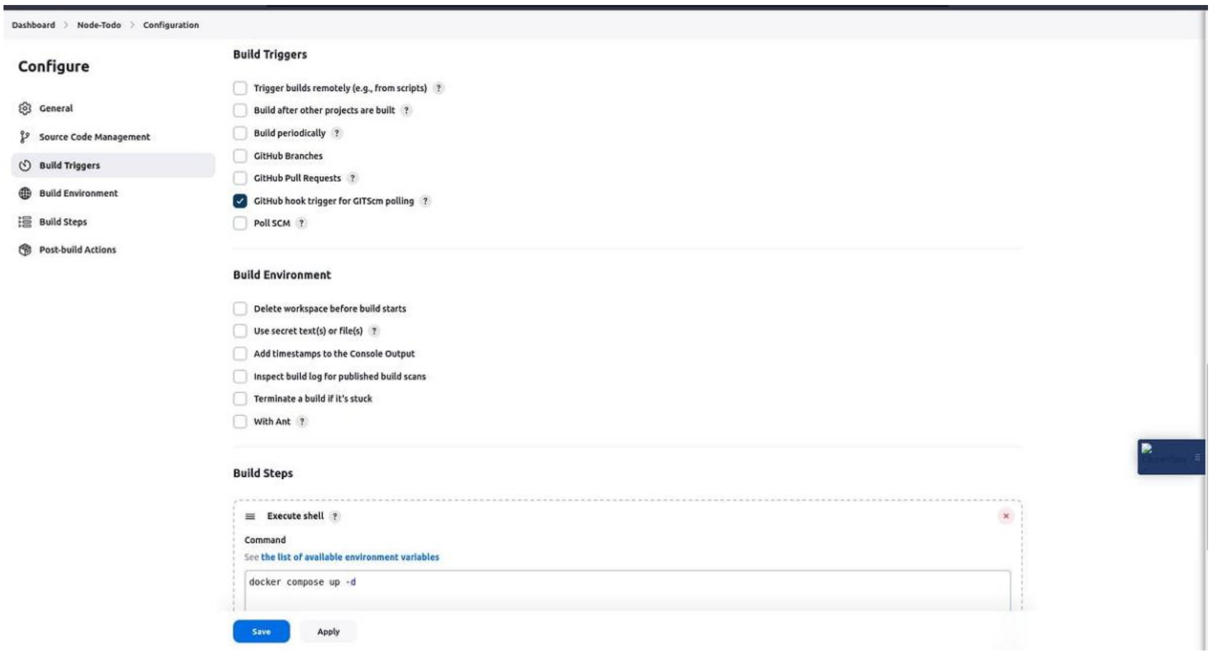
☐ Throttle builds

☐ Execute concurrent builds if necessary

☐ Restrict where this project can be run

Source Code Management

Under Build Triggers, select GitHub hook trigger for GITScm polling. This will enable the job to be triggered by GitHub webhooks.


Dashboard > Node-Todo > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub Branches
- [ ] GitHub Pull Requests
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

Build Environment

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s)
- [ ] Add timestamps to the Console Output
- [ ] Inspect build log for published build scans
- [ ] Terminate a build if it's stuck
- [ ] With Ant

Build Steps

Execute shell

Command

See the list of available environment variables

```
docker compose up -d
```

Click Save to save your job.

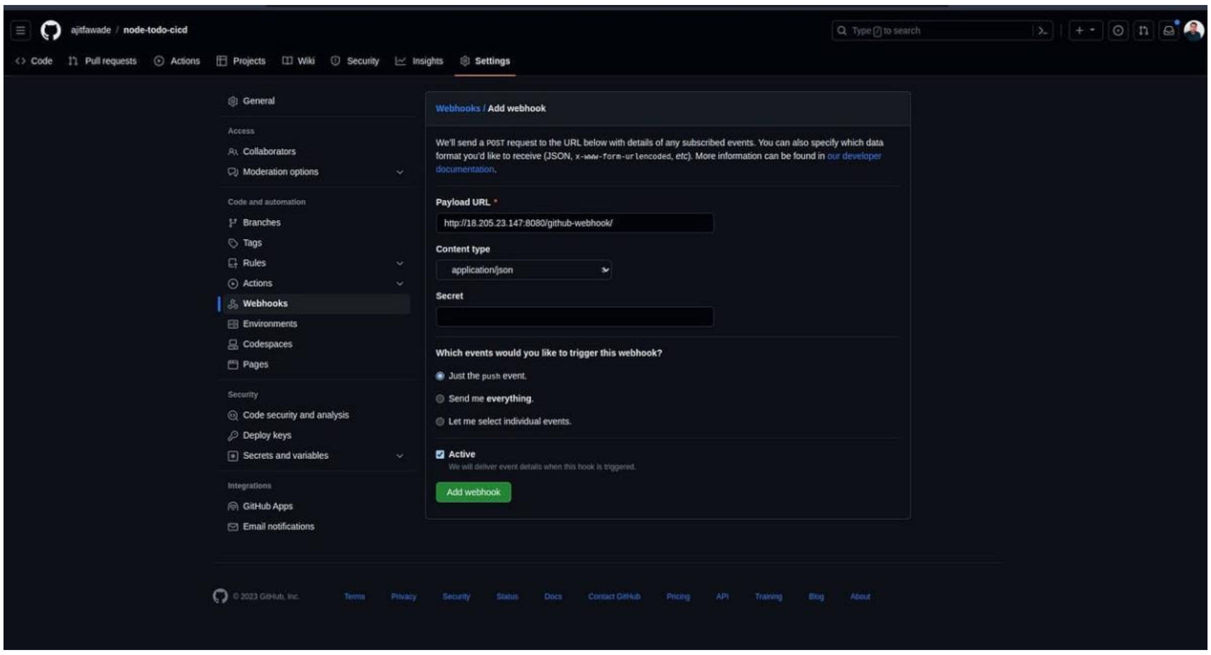
Now, go to your GitHub repository and click on Settings. Then click on Webhooks and then on Add Webhook.

Fill in the form as follows:

- Payload URL: Enter your Jenkins URL followed by `/github-webhook/`. For example, http://example.com/github-webhook/.
- Content type: Select `application/json`.
- Secret: Leave it blank.
- SSL verification: Select `Enable SSL verification` if your Jenkins URL uses HTTPS, otherwise select `Disable SSL verification`.
- Which events would you like to trigger this webhook?: Select `Just the push event`.

- Active: Check this box.

Click on Add webhook to create the webhook.



Webhooks / Add webhook

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://18.205.23.147:8080/github-webhook/

Content type: application/json

Secret:

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.

You can test the webhook by clicking on Edit and then on Test. You should see a green check mark indicating that the webhook was delivered successfully.

How to Run the App Using Docker Compose?

To run the app using Docker Compose, we need to create two files:

- `Dockerfile`: A file that defines how to build a Docker image for the Node.js app.
- `docker-compose.yml`: A file that defines how to run multiple containers for the Node.js app using Docker Compose.

Let's create these files in our GitHub repository.

Creating the Dockerfile

The Dockerfile for the Node.js app will look something like this:

```
# Use node:14-alpine as base image
FROM node:14-alpine
# Set /app as working directory
WORKDIR /app
# Copy package.json and package-lock.json files to /app
COPY package*.json /app/
# Install dependencies
RUN npm install
# Copy app source code to /app
COPY . /app/
# Expose port 8000
EXPOSE 8000
# Run app.js as entrypoint
CMD ["node", "app.js"]
```

This Dockerfile does the following:

- It uses `node:14-alpine` as the base image, which is a lightweight version of Node.js image based on Alpine Linux.
- It sets `/app` as the working directory, where the app code will be copied and executed.
- It copies the `package.json` and `package-lock.json` files to `/app` and installs the dependencies using `npm install`.
- It copies the app source code to `/app`.

- It exposes port 8000, which is the port that the app listens on.
- It runs `node app.js` as the entrypoint, which starts the app.

Creating the docker-compose.yml File

The docker-compose.yml file for the app and the database will look something like this:

```
version: '3'
services:
  web:
    image: "trainwithshubham/node-app-test-new:latest"
    ports:
      - "8000:8000"
```

This docker-compose.yml file does the following:

The `web` service pulls the image from the Dockerhub in the current directory and maps port 8000 of the container to port 8000 of the host.

How to Build and Run the App with Jenkins?

Now that we have created the Dockerfile and the docker-compose.yml file, we can use them to build and run the app with Jenkins.

To do this, follow these steps:

1. Go back to your Jenkins dashboard and click on your job name (e.g., Node Todo App). Then click on Configure.
2. Under Build, click on Add build step and select Execute shell (or Execute Windows batch command if you are using Windows). This will allow you to run any shell or batch command as part of your job.
3. In the text area that appears, enter the command `docker-compose up -d --build` to build and run the app using Docker Compose. The `-d` flag runs the containers in detached mode and the `--build` flag forces a rebuild of the images if there are any changes.
4. Click Save to save your job.

Now, whenever you push a new commit to the repository, a new build will be initiated automatically.

You will see a new build number appear under Build History with a blue ball indicating that the build is in progress. You can click on it to see the details of the build.

Click on Console Output to see the logs of the build. You should see something like this:

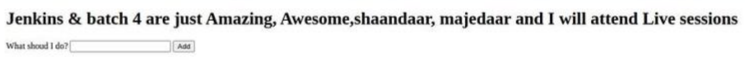
```
[Node Todo App] $ /bin/sh -xe /tmp/jenkins1234567890.sh
+ docker-compose up -d --build
Creating network "nodetodoapp_default" with the default driver
Building web
```

```
Step 1/7 : FROM node:14-alpine
 ---> 8a5b9c7f8f6c
Step 2/7 : WORKDIR /app
 ---> Using cache
 ---> 8a5b9c7f8f6c
Step 3/7 : COPY package*.json /app/
 ---> Using cache
 ---> 8a5b9c7f8f6c
Step 4/7 : RUN npm install
 ---> Using cache
 ---> 8a5b9c7f8f6c
Step 5/7 : COPY . /app/
 ---> Using cache
 ---> 8a5b9c7f8f6c
Step 6/7 : EXPOSE 8000
 ---> Using cache
 ---> 8a5b9c7f8f6c
Step 7/7 : CMD ["node", "app.js"]
 ---> Using cache
 ---> 8a5b9c7f8f6c
Successfully built 8a5b9c7f8f6c
Successfully tagged nodetodoapp_web:latest
Creating nodetodoapp_db_1 ... done
Creating nodetodoapp_web_1 ... done
Finished: SUCCESS
```

You can see that the images and the containers are created and started successfully.

https://www.linkedin.com/company/devops-and-cloud-labs/

You can also verify that the app is running by going to your browser and typing http://localhost:8000. You should see a to-do app where you can add, edit, and delete tasks.



Jenkins & batch 4 are just Amazing, Awesome,shaandaar, majedaar and I will attend Live sessions

The Project is Successfully Deployed !!!

For more such Projects and DevOps contents follow us on Youtube:
https://www.youtube.com/@devops-cloud