Deploying Jenkins server in EC2 instance (ubuntu)

What we need?

1. Aws account.
2. OS with which we can make ssh connections.

Step1: Create an ec2 instance with ubuntu OS

* Volume Size can be around 30 GB to run Jenkins server
* Select ubuntu ami id
* In network settings create one NSG rule means check mark for http connection so that we can access Jenkins through web
* Next create one key pair if we have one use can use it too.
* Launch instance

Step2: connecting to ec2 instance

* Go to terminal and go to the path where the .pem file is there
* Next give permissions to that .pem file to read and write only by the user (chmod 600)

icacls “filename” /inheritance:r /grant:r “$($env:USERNAME):(R,W)” /remove “Everyone”

* Next ssh using the following command

ssh -i .pem\_filename username@(ip/server name)

Ex: ssh -i Jenkins-server.pem [ubuntu@172.56.34.54](mailto:ubuntu@172.56.34.54)

Step3: deploy Jenkins in that ec2

* Now create one .sh file
* In that copy the following code

<https://github.com/LinkedInLearning/essential-jenkins-2468076/blob/main/Ch01/01_03-solution-deploy-a-jenkins-server/jenkins-server-automated-installation.sh>

* Next run that file using “source filename.sh” command.
* Check Jenkins status using “sudo systemctl status jenkins.service” command.
* If the state is running ok if not give permissions for secrets path to Jenkins using command “sudo chown -R Jenkins:jenkins /var/lib/jenkins”
* There may be some errors try to rectify them.

Step4: connecting to Jenkins server

* Open browser and give http connection to connect to Jenkins server

“http:dns”

**GitHub WebHook**:

Within a [Jenkins pipeline](https://dzone.com/articles/getting-started-with-jenkins), a GitHub webhook serves as a vital link between your GitHub repository and Jenkins, the [CI/CD](https://dzone.com/articles/what-is-ci-cd) automation tool. It establishes an automated communication channel that enables Jenkins to respond to GitHub events, such as code pushes or pull requests, triggering corresponding build and [deployment processes](https://dzone.com/articles/process-flow-for-deployment-in-devops). This integration is pivotal for streamlining development workflows, providing real-time feedback, and ensuring that your CI/CD pipeline remains synchronized with changes in your GitHub repository.

We should add some inbound rules in security groups to allow connect to apps or Jenkins.  
  
Follow the following link for detailed understanding of how to setup Jenkins server and how to automate the job trigger and integrate Jenkins with github for automation means instead of we clicking the build now, when we commit changes to github repo automatically the job get triggered.

<https://medium.com/@mudasirhaji/complete-step-by-step-jenkins-cicd-with-github-integration-aae3961b6e33>