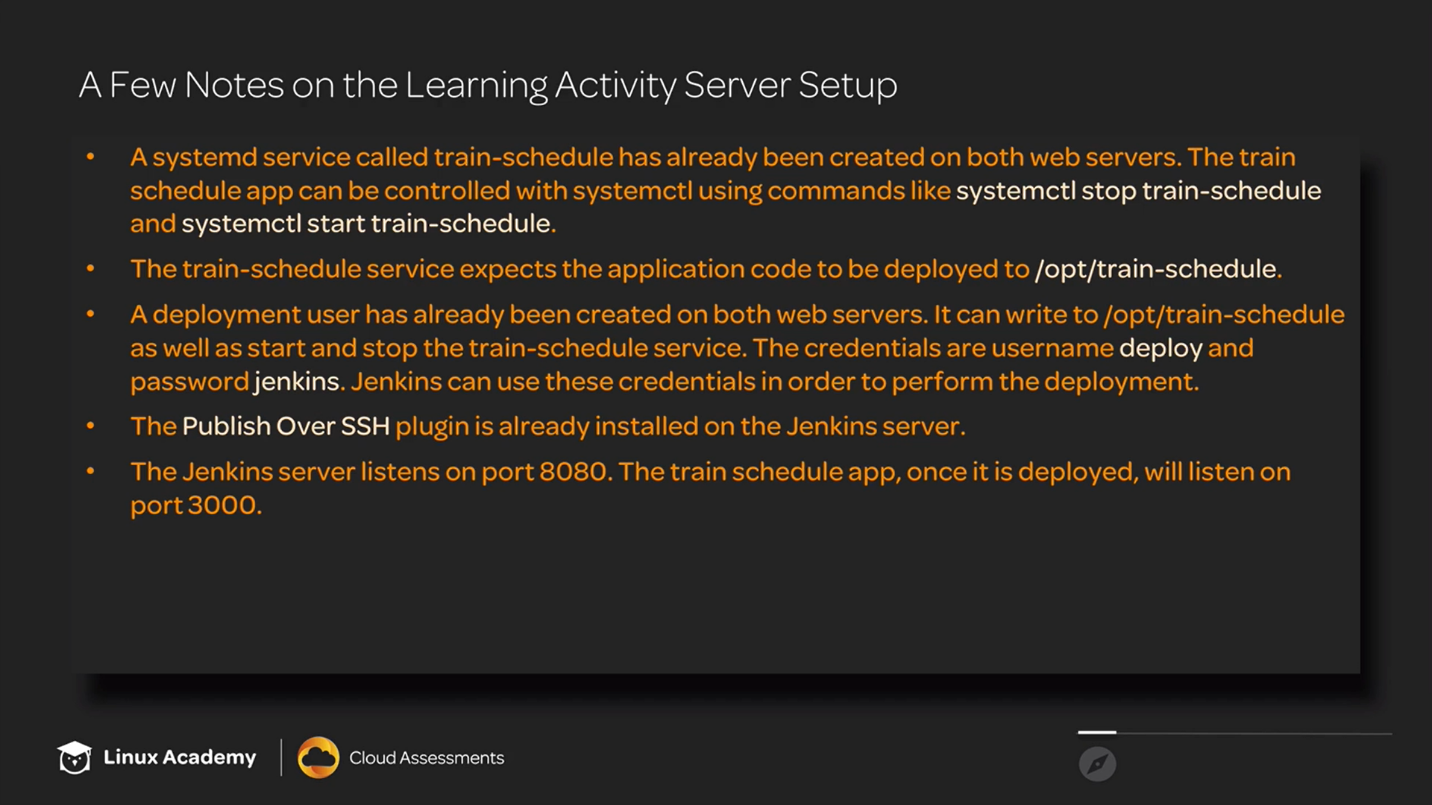
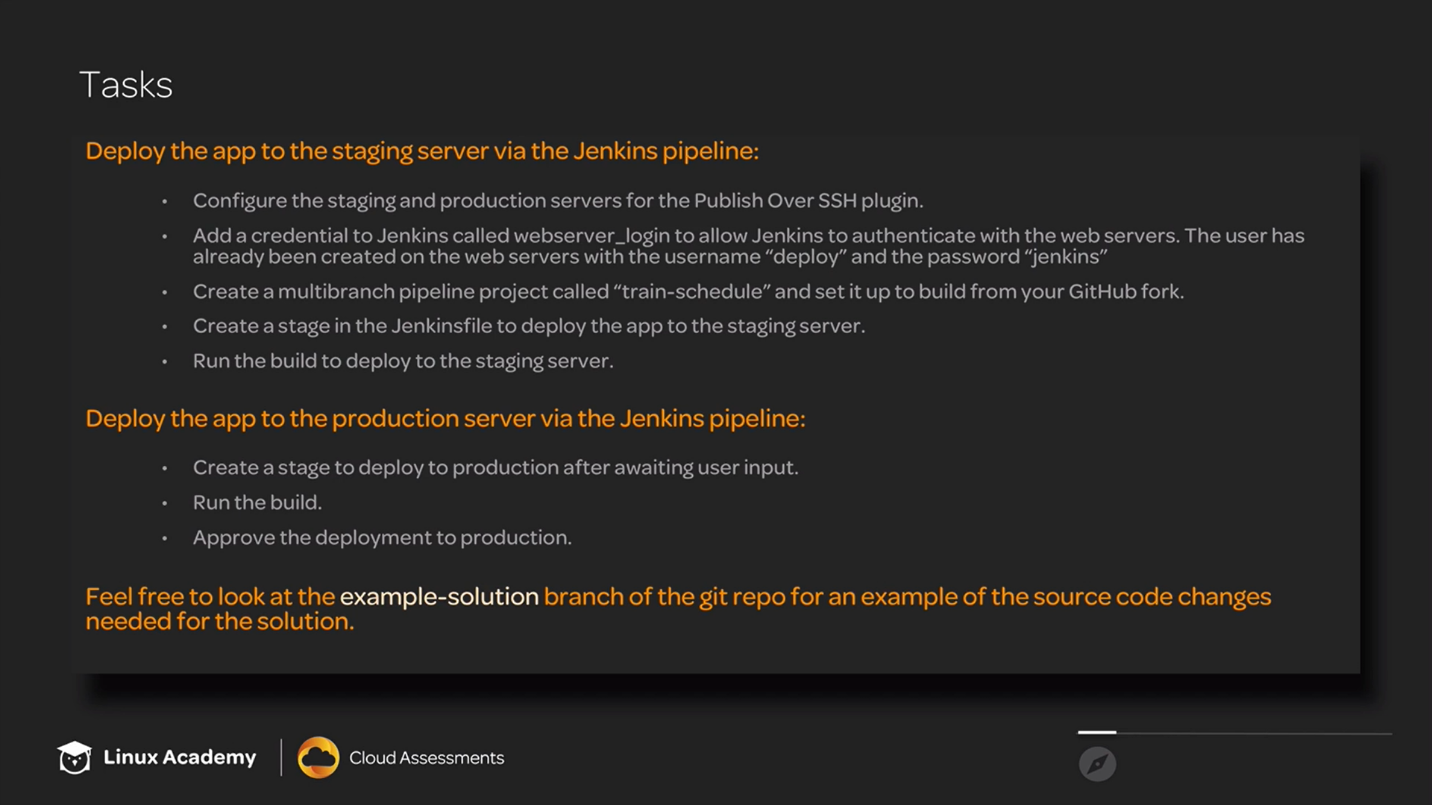
**Build = CI, Deploy to staging Server= Continuous Deployment**

**Deploy to Production= Continuous Delivery**

**Text

Description automatically generated**



**Note:- Isi me 3 credentials hain**

**Implementing Automated Deployment Through a Jenkins Pipeline**

**Introduction**

Jenkins Pipeline is a powerful tool for implementing continuous delivery. In order to fully utilize Jenkins Pipeline, you will need to implement an automated deployment. This learning activity will guide you through the process of deploying code as part of a Jenkins pipeline. After completing this exercise, you will have a basic familarity with what automated deployment using Jenkins Pipeline looks like.

**Solution**

1. In a new browser tab, log in to the Jenkins instance using the Jenkins server public credentials provided:

<PUBLIC\_IP\_ADDRESS>:8080

1. Using the same credentials, open a terminal window and log in to the Jenkins server using SSH to retrieve the temporary admin password:

ssh cloud\_user@<PUBLIC\_IP\_ADDRESS>

1. Retrieve the temporary admin password:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

1. Copy the temporary admin password and paste it into the *Administrator password* field in the new browser tab.
2. Click **Continue**.

**Deploy the App to the Staging Server Via the Jenkins Pipeline**

**Getting started**

1. On the *Create First Admin User* form, provide the following information:
   * *Username*: **jenkins**
   * *Password*: random
   * *Confirm password*: random
   * *Full name*: **jenkins**
   * *Email address*: **noreply@linuxacademy.com**
2. Click **Save and Continue** > **Start using Jenkins**.

**Configure staging and production Servers for the Publish Over SSH Plugin**

1. On the lab page, copy the staging server public IP address.
2. In the Jenkins tab in your browser, click **Manage Jenkins** and then click **Configure System**.
3. Scroll to the bottom of this page to find the *Publish over SSH* section.
4. In *SSH Servers*, click **Add**.
5. Add the following *SSH Server* values, replacing <STAGING\_SERVER\_PUBLIC\_IP\_ADDRESS> with the IP address copied earlier:
   * *Name*: **staging**
   * *Hostname*: <STAGING\_SERVER\_PUBLIC\_IP\_ADDRESS>
   * *Remote Directory*: **/**
6. Click **Add**.
7. Return to the lab page and copy the production server public IP address.
8. Add the following *SSH Server* values, replacing <PRODUCTION\_SERVER\_PUBLIC\_IP\_ADDRESS> with the IP address copied earlier:
   * *Name*: **production**
   * *Hostname*: <PRODUCTION\_SERVER\_PUBLIC\_IP\_ADDRESS>
   * *Remote Directory*: **/**
9. Click **Save**.

**Set Up Jenkins Credentials**

1. From the left menu, click **Credentials**.
2. In *Stores scoped to Jenkins*, click **global**.
3. On the left, click **Add Credentials**.
4. Add the following values:
   * *Username*: **deploy**
   * *Password*: **jenkins**
   * *ID*: **webserver\_login**
   * *Description*: **Webserver Login**
5. Click **OK** to save our changes.

**Set up the Jenkins Project**

1. On the top menu, click **Jenkins** to return to the main page.
2. From the left menu, click **New Item**.
3. Enter the item name "train-schedule".
4. Select **Multibranch Pipeline** and click **OK**.
5. Navigate to the [train-schedule Git repo](https://github.com/linuxacademy/cicd-pipeline-train-schedule-cd) and click **Fork** to fork the repo to your account.
6. From the GitHub top menu, click the avatar icon and click **Settings** > **Developer settings** > **personal access tokens** > **Generate new token**.
7. In *Token description*, enter "Jenkins".
8. Select **admin:repo\_hook**. and click **Generate token**.
9. Copy the generated API token to the clipboard.
10. Return to the Jenkins page and select the *Branch sources* tab.
11. In *Branch Sources*, click **Add source** and select **GitHub**.
12. In *Credentials*, click *Add* and select **Jenkins**.
13. Set the following values:
    * *Username*: Your GitHub username
    * *Password*: Generated API token copied earlier
    * *ID*: **github\_key**
    * *Description*: **GitHub Key**
14. Click **Add**.
15. In *Credentials*, select the newly created GitHub Key credential.
16. In *Owner*, enter your GitGub username.
17. In *Repository*, select **cicd-pipeline-train-schedule-cd**.
18. Click **Save**.
19. From the top menu, click **train-schedule** and then click **master** to view the initial build in the master branch.

**Create a Stage in the Jenkinsfile and Run the Build**

1. From the GitHub personal fork, open the Jenkinsfile and click the pencil icon to edit the file's contents.
2. Delete the file contents.
3. In a new browser tab, access the solution Jenkinsfile on the [Github example-solution branch](https://github.com/linuxacademy/cicd-pipeline-train-schedule-cd/blob/example-solution/Jenkinsfile" \t "_blank) and copy the DeployToStaging stage text.
4. Paste the text into the Jenkinsfile in our GitHub fork.
5. Click **Commit changes**.
6. Return to the Jenkins *Branch master* page.
7. On the left menu, click **Build Now**.
8. To test our deployment, copy the staging server public IP address again from the lab page.
9. Open a new browser tab and paste the IP address, specifying the port 3000. Our train schedule application should load successfully.

**Deploy the App to the Production Server Via the Jenkins Pipeline**

1. From the GitHub personal fork, open the Jenkinsfile and click the pencil icon to edit the file's contents.
2. In a new browser tab, access the solution Jenkinsfile on the [Github example-solution branch](https://github.com/linuxacademy/cicd-pipeline-train-schedule-cd/blob/example-solution/Jenkinsfile" \t "_blank) and copy the DeployToProduction stage text.
3. Paste the text into the Jenkinsfile in our GitHub fork.
4. Click **Commit changes**.
5. Return to the Jenkins *Branch master* page.
6. On the left menu, click **Build Now**.
7. Hover over DeployToProduction and click **Proceed**.
8. To test our deployment, copy the production server public IP address again from the lab page.
9. Open a new browser tab and paste the IP address, specifying the port 3000. Our train schedule application should load successfully.

**Conclusion**

Congratulations — you've completed this hands-o