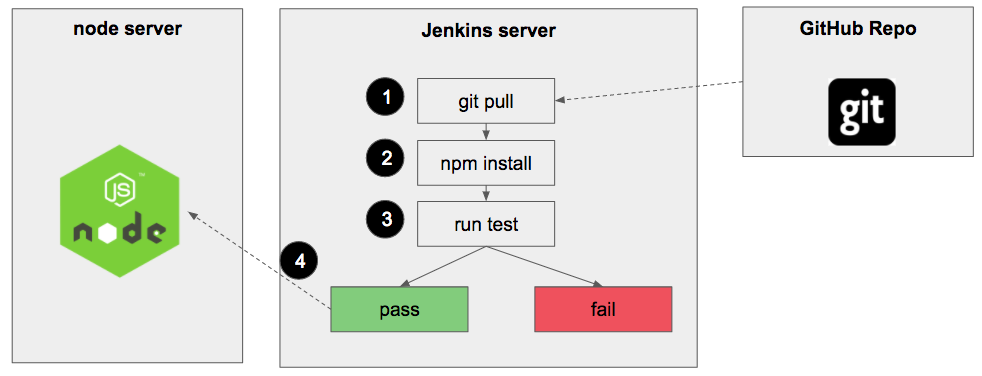
**CRUD Application deployment on NODEJS**



**Creating a Node App**

Before we write any CI/CD pipeline we need an application to test and deploy. We are going to build a simple node.js application that responds with apps.js. First, let’s set up our GitHub repository for this project.

Set GitHub Repository

Create new repository under your GitHub account and name it **“node-app”**.

* You can choose **public or private repo**
* check the **Initialize this repository with a README** checkbox
* Select node in the **Add .gitignore** drop-down menu
* Click **Create repository** button

Now let’s clone our node-app repo to our local computer and navigate to it:

git clone https://github.com/chapagain/nodejs-mysql-crud.git

cd nodejs-mysql-crud

Create Node.js App

The first step when building a node app is creating package.json file. In this file, we list the application dependencies. Create a new file in your project root called **package.json** and copy paste the following content into it:

{

"name": "crud-mysql",

"version": "1.0.0",

"description": "Node.js CRUD application using MySQL",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [

"node",

"crud",

"mysql"

],

"author": "Mukesh Chapagain",

"license": "ISC",

"dependencies": {

"body-parser": "^1.17.2",

"cookie-parser": "^1.4.3",

"ejs": "^2.5.6",

"express": "^4.15.3",

"express-flash": "0.0.2",

"express-myconnection": "^1.0.4",

"express-session": "^1.15.3",

"express-validator": "^3.2.0",

"method-override": "^2.3.9",

"mysql": "^2.13.0"

}

}

After we defined our dependencies in package.json file we are ready to install them:

npm install

Create a new file in the project root called **apps.js**

Now we are ready to run our app:

node index.js

You can view your app on your browser when you navigate to *http://localhost:3000*

**Serve Node App**

We are going to host our node app on a server so the entire world can see our masterpiece. We will use AWS Cloude as our hosting provider. AWS provides an easy way to configure servers and spin new instances.

**Nodejs-app Server Configuration**

Let’s put the DevOps hat on and set up our node server

Open your terminal on your local machine and login into your nodejs-app server as a root user:

ssh root@NODE.SERVER.IP

Now you loged in as root user which is a super power user. And “with great power comes great responsibilities”.

Since we don’t like responsibilities let’s create a new user to do the server configuration work and name it after your last name:

adduser <lastname>

Choose user password and follow the prompts. Before we are switching to our new user we need to give him sudo privileges:

usermod -a -G sudo <username>

Now you can switch to your new user.

su — username

**Deploy the node-app**

Our DigitalOcean server comes with Node but not Git. Lets install git using app-get:

sudo apt-get install git

Clone our node app repo:

git clone https://github.com/<username>/node-app.git

Navigate into the project folder and install app dependencies:

For Production environment

cd node-app  
npm install — prod

For UAT environment

cd node-app  
npm install — uat

For Dev environment

cd node-app  
npm install — dev

Before we can access our app on the browser we need to complete an additional step. As you recall we are running our app on port 3000 by default. AWS blocks clients accessing any port but 80.

node index.js

Now you can access your node app by appending the PORT to your IP address:

[http://NODE.SERVER.IP:3000](http://node.server.ip:3000/)

**Running Node App Forever**

Starting node app like above is good for development purposes but not in production. In case our node instance crash we need a process that will do the auto restart. We are going to use PM2 module to help us with this task. PM2 is a general purpose process manager and a production runtime for Node.js apps with a built-in Load Balancer. Let’s install PM2 and start our node instance:

sudo npm install pm2@latest -g  
pm2 start apps.js

Now our node server is configured and running.

**Set Up Jenkins Server**

Creating Jenkins Droplet

Let’s start by creating a second DigitalOcean Droplet that will serve our Jenkins app. Follow the instructions under **Creating Node Droplet** section above and choose **“jenkins-app”** as your hostname. You will end up with 2 droplets:

Create New User

SSH into the new droplet as root user, create a new user, give it sudo privileges and switch to the newly added user:

ssh root@JENKINS.SERVER.IP  
adduser <username>  
usermod -a -G sudo <username>  
su — <username>

Jenkins will need to be able to pull changes from the node app repo, therefore, we need to install git on the instance:

sudo apt-get install git

Install Jenkins

Get Jenkins:

*//add the repository key to the system*  
wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -*//append the Debian package repository address to the server's echo* deb https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list*//update*  
sudo apt-get update

Install Jenkins:

sudo apt-get install jenkins

Start Jenkins:

sudo systemctl status jenkins

And now we can access Jenkins on the browser by navigating to:

[http://JENKINS.SERVER.IP:8080](http://jenkins.server.ip:8080/)

**Configure Jenkins**

When you navigate to Jenkins homepage you probably noticed additional step you need to do. You need to unlock Jenkins

Copy the Jenkins password hosted on your Jenkins server

sudo vim /var/lib/jenkins/secrets/initialAdminPassword

Paste the the password into the text field. You are ready to set up Jenkins. First, we would like to add GitHub plugin. From the left menu select **manage Jenkins**and go to **manage plugins.**On the plugins page select the **available**tab and look for **GitHub plugin,**select its checkbox and click the**Download now and install after restart**button.

Once the installation is complete scroll down the page and select the **Restart Jenkins when installation is complete.**This will restart Jenkins and complete the installation.

Change Your Jenkins Admin Password

I suggest at this point to change your Jenkins admin user password. Select **Manage Jenkins**from the left menu and click on **Manage Users.**Select the admin user and choose a new password. You will use the new password when you log in to Jenkins in the future.

**Create Jenkins Job**

We are going to create our Jenkins job that will be responsible for pulling code changes from node-app git repo, install dependencies, run integration test and deploy the application every time a developer push changes to the nodejs-app repo master branch.

Click on **New Item**button,name the item **node-app**and select **Build a free-style software project**option and click the **OK** button.

**Configure Jenkins Job**

**Source Code Management:**Select **git** radio button and enter github https link to the node-app repo:

[https://github.com/<username>/node-app.git](https://github.com/mezderman/node-app.git)

**Build Triggers:**Selectoption**GitHub hook trigger for GITScm polling.**This will start our Jenkins job on every git push on the master branch

**Add Build Step**: Click on Add Build Step button and select **Execute Shell**option. Enter the following commands into the text area:

npm install  
./script/test

In this build step we are going to install dependencies and then run our test shell script.

Add Git Webhook

We are going to add Git Webhook to inform Jenkins every time a developer push new code to master branch.

Go to node-app GitHub, click on the **Settings** tab, select **Webhooks**fromthe left menu and click on the **Add Webhooks**button. Enter your Jenkins webhook URL under **Payload URL**:

[http://JENKINS.SERVER.IP:8080/github-webhook/](http://jenkins.server.ip:8080/github-webhook/)

and select **Just the Push Event**option. Click the **Add webhook**button.

Let’s test what we have so far. Go to your node-app project on your machine and change the version in the package.json to 0.0.2. Commit and push this change to GitHub. After you push, go to your Jenkins job on the browser and observe that the Jenkins job started and completed successfully.

Deployment

The last piece of the puzzle is deploying our node application into the node-app server when our test passes.

SSH Authentication

In order to do that, Jenkins Server will need to ssh into the node-app server, clone the repo, install dependencies and restart the server. Lets set up ssh access to Jenkins first.

When we install Jenkins it automatically created Jenkins user. SSH into our Jenkins server as root user:

ssh root@JENKINS.SERVER.IP

Switch to Jenkins user:

su — jenkins

Generate SSH key:

ssh-keygen -t rsa

And save the generated key in */var/lib/jenkins/.ssh/id\_rsa*

Print the SSH key you just created:

cat ~/.ssh/id\_rsa.pub

And Copy the output to your clipboard. Now we are ready to put the public key on the nodejs-app server to complete the authentication between Jenkins server and nodejs-app server.

SSH into the nodejs-app server as a root and switch to your user:

ssh [root@NODE.SERVER.IP](mailto:root@JENKINS.SERVER.IP)  
su - <username>

Open the the file where authorized keys are stored:

vim ~/.ssh/authorized\_keys

And copy paste the Jenkins public key we just created into that file. Save by pressing the **esc** button on your keyboard, type **:x**and press**enter.**

Set the correct permission on the .ssh folder:

chmod 700 ~/.ssh  
chmod 600 ~/.ssh/\*

Before we are moving on lets test our SSH set up. If set up is correct we will be able to SSH from JENKINS.SERVER.IP as jenkins user to <username>@NODE.SERVER.IP without entering a password

ssh root@JENKINS.SERVER.IP  
su - jenkins  
ssh <username>@NODE.SERVER.IP

Success!

Automatic Deployment

We are going to create another shell script that responsible for the deployment. Create a file under **script**called**deploy**and add the following script:

#!/bin/shssh ezderman@NODE.SERVER.IP <<EOF  
 cd ~/node-app  
 git pull  
 npm install — production  
 pm2 restart all  
 exit  
EOF

This script will SSH to the node server, pull changes from GitHub, install dependencies and restart serve.

Make our new script file executable:

chmod +x script/deploy

Before we commit our changes lets add the deployment step to our Jenkins Job:

We are ready to test everything we built. Go to your nodejs-app project and edit the apps.js file to respond.

Commit and push:

git add .  
git commit -m ‘add deployment script’  
git push origin master

After you push you should see Jenkins Job starts. When complete, you should see your changes on [http://NODE.SERVER.IP:3000](http://node.server.ip:3000/)