|  |  |
| --- | --- |
| **DOCUMENT RULES:** | |
| **Task name** | **Building Docker Images using Jenkins Pipeline** |
| **Task name & column name should be written:** | **Bold (CTRL+B)** |
| **Commands should be written in the after # sign:** | *Italic (CTRL+I) #hostname* |
| **Output photo should be cropped or compressed:**  **Photo could be more than one:**  **If you need extra lines, add the line next after it:** | ***Description photo should be with title bar (CTRL + I + B)*** |
| **All other text should be written:** | Standard |
| **Font name and text size:** | Calibri and 9 |
| **Group name:** | Dev\_ops\_1 |
| **Student name and surname:** | Ehtiram Mustafayev |
| **E-mail:** | ehtiramst@gmail.com |
| **WhatsApp number:** | **+994507492393** |

|  |  |
| --- | --- |
| **Task names** | **Command steps and outputs** |
| **Execute the following command to run the Jenkins Docker container:** | docker run -d -u root --name jenkins -p 8080:8080 -p 50000:50000 -v /root/jenkins:/var/jenkins\_home omarmohsen/jenkins-sa |
| **Meaning of the parameters used in the** docker run **command**: | -d --> runs in the background  --name --> name of the container running  -p --> the port number where you can access the container  -v --> the volume attached to the Jenkins container |
| **Now lets check that our container is running** | docker ps |
| **You can view the logs of the container to check that Jenkins in** **up and running:** | docker logs jenkins -f |
| **So access it from here, choose port 8080**: [ACCESS PORTS](https://killercoda.com/scenario/traffic/ede97ed4-26eb-47e8-a747-102c11e42111) |  |
| **Please follow these steps to install the above plugins:** | **1**- **Click** Manage Jenkins **from the left panel**.  **2-** **Click** Manage Plugins .  **3**- **Click** Available **tab**.  **4**- **Search for the plugins you want to install, and check their box**.  **5**- **Click** Install without restart . |
| **1- Install Java on your machine which is a dependency for the jenkins agent to work:** | sudo apt install openjdk-11-jre |
| **So follow these steps to configure your agent:** | **2- Click** Manage Jenkins **from the left panel.**  **3- Click** Manage Nodes .  **4- Click** New Node .  **5- Give your agent a name**, test .  **6- Select** Permanent Agent .  **7- For** Remote root directory **type** /root .  **8- For** Launch method , **select** Launch agent via ssh .  **9- For** Host , **you will need to add the host IP, retrieve the IP from the 2nd network interface (starts with en):** |
| ip a |  |
|  | **10- Click** Add credentials .  **a- For** Kind , **select** SSH username and private key .  **b- For** Scope , **select** Global .  **c- For** ID , **type** agent .  **d- For** Username , **type** root .  **e- Get the Private key by executing** cat /root/.ssh/id\_rsa .  **f- Click** Private Key **check mark, then click** Add **button and add the Private key from the previous step.**  **11- Choose the credential that you created from the drop down**.  **12-** **For** host verification strategy , **select** None verifying .  **13-** **Click** Save .  **14- To view the progress, Click you agent you just created, the Click** logs |
| **you will give you the syntax to paste it and describe in details what exactly this does.**  **So follow these steps to** **create your pipeline:** | **1- Click** New Item .  **2- Select** Pipeline .  **3- Give it a name**, django-pipeline .  **4- Click** Ok , **this will open a new page for you**.  **5- Scroll to the bottom to the** Pipeline **section, this is where you will type your groovy code in.**  **6- Paste the following in the** Pipeline **section, this is the basic structure of the pipeline.** |
| **So, paste the following as your second stage after the previous one:** | stage('Install python requirements') {  steps {  sh 'pip3 install -r requirements.txt'  }  }  ```  in the `/root/workspace/django-pipeline/requirements.txt`{{open}} file, you will find a list of packages that python will need to run the Django application  Now lets Run the pipeline and test the stages you made.  If it passed, proceed with the next step, if not please troubleshoot and test again. |
| **In this step, you will configure a stage to run unit test on the code you cloned from the repository.** | stage('run tests') {  steps {  sh 'python3 manage.py test blog'  }  }  ```  Now lets Run the pipeline and test the stages you made.  If it passed, proceed with the next step, if not please troubleshoot and test again. |
| **Running the application directly on the host is not the best practice, its better to run on a container, and thats what we are going to do in this step.**  **So, we will substitute the** run python app **stage with the following**: | stage('Build docker image') {  steps {  sh 'docker build -t django-blog -f docker/Dockerfile . '  }  } |
| **In this step, we will add a stage to run the image we made in** **the previous one** | stage('Run docker container') {  steps {  sh 'docker run --rm -d -p 80:8000 django-blog'  }  } |