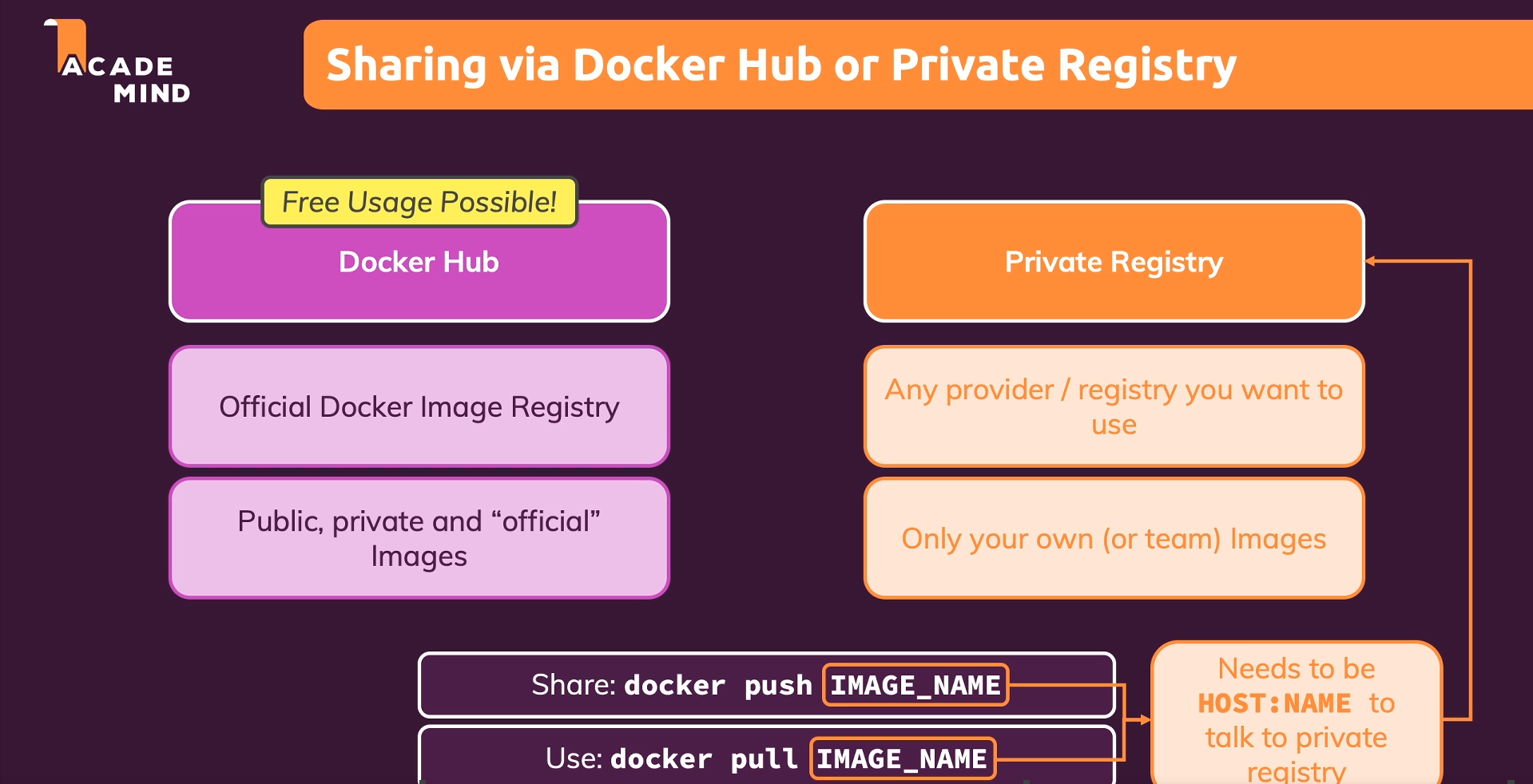
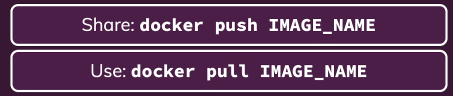
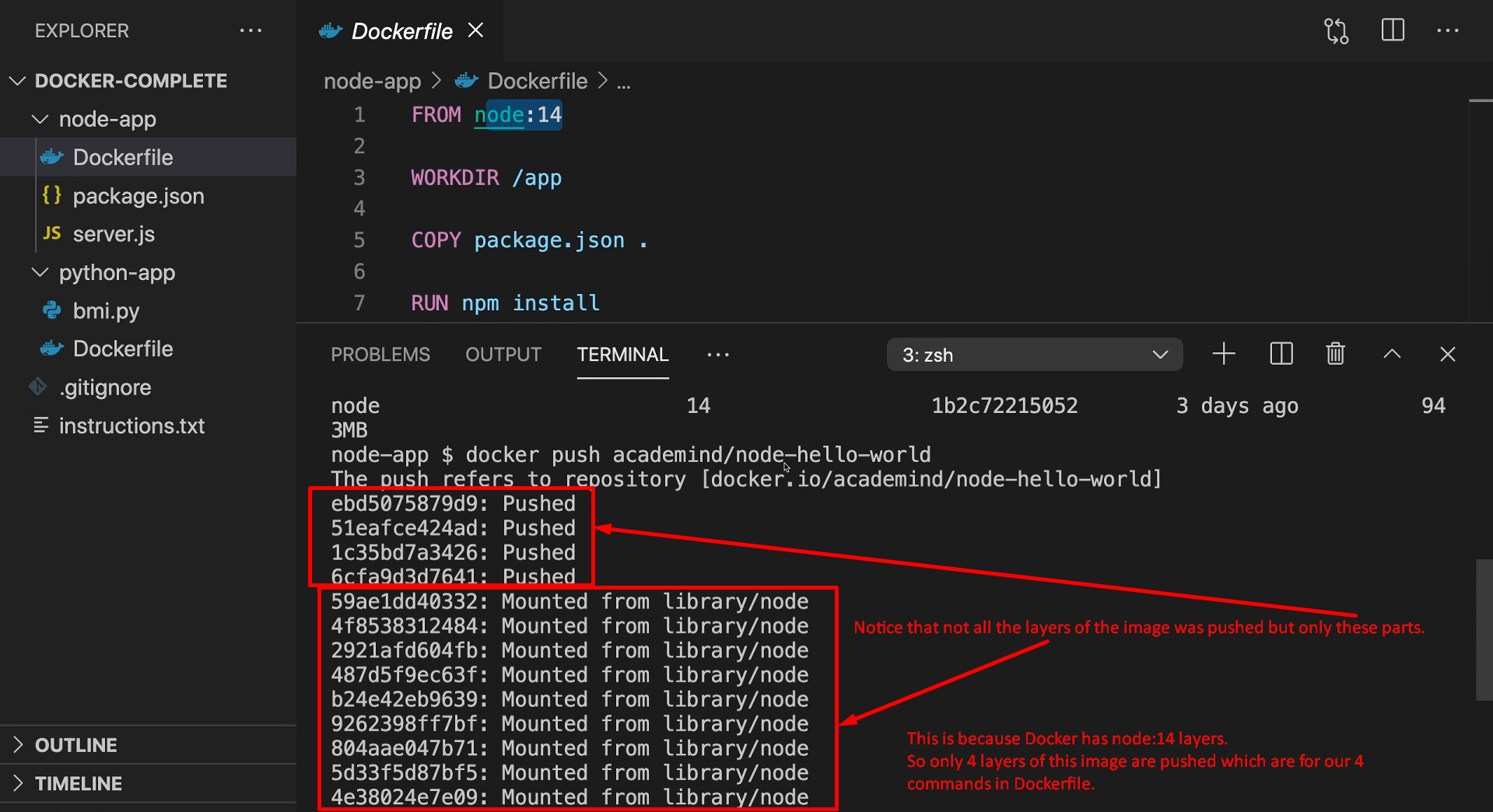
1. **Agenda**:
   1. How to share the finished/built images?
2. Pre: You may need docker tag command for this lecture. This command is used to give a new image name.
   1. docker tag <image\_Id> | <image\_name\_optional\_tag> name:optional\_tag  
      where blue is old image and green is new image name.
3. When we want to share images, there are two main places where we can push our images to.
4. Docker Hub:
   1. Official Docker Image Registry.
   2. Free to use.
   3. It offers other thousands of services such as managing/distributing the images.
5. Private Registry:
   1. There are many providers out there.
   2. We can use any provider who supports Docker Images.
   3. Now it depends on the provider if you and your team or everyone is able to see these images.
6. 
7. **NOTE**: The shared image in the end is a repository.
8. With he following two commands, we can push and pull Docker Images.   
     
   **NOTE**: If you push and pull with regular image repository name, then this will automatically go to Docker Hub.  
   **NOTE**: If you want to push and pull from a private Registry then you need to include the HOST URL in the name like this: 🡺 **docker push** [**URL:Name**](URL:Name)
9. Now we’re going to share our Docker image with Docker Hub.
   1. **Step 1**: Create a Docker Hub Account for free.
   2. **Step 2**: Login the Account.
   3. **Step 3**: Create Repository.
      1. Graphical user interface, text, application, email

         Description automatically generated
      2. Each repo has a name which will be created under your account name not under account email.  
         **Syntax**: <account-name>/<repo-name>  
         **Example:** **jatinbansaldocker/node  
         Real Example:**See, jatinbansaldocker is username/account-name by which you login hub.docker.com  
         and dockerpractice is the image name.
      3. Take another example: Suppose you have an app named ms-payees-openapi.  
         You will create first image with tag 1.0  
         So you will push the image with command   
         🡺 **docker push jatinbansaldocker/ms-payees-openapi:1.0**  
         Now suppose, you create another version for the ms-payees-openapi:1.0 which is 1.1  
         🡺 **docker push jatinbansaldocker/ms-payees-openapi:1.1**  
         So, under the repo “ms-payees-openapi”, you will have two versions 1.0, 1.1  
         Graphical user interface, application

         Description automatically generated
   4.   
      Think like this:   
      Suppose node:14 has 3 layers as a stack first n1, then n2, then n3 and your app has two layers a1, a2  
      So starting from the bottom in the stack it will look like this -> n1, n2, n3, a1, a2.  
      When you push your image with these layers, Docker will start pushing layers from the top in the stack meaning starting with a2 and long the way it will check if that layer is already on Docker Hub somewhere.   
      Like in our case n3 layer, it will find as it is from node:14 which is already on Docker hub so it will not push this layer and remaining layers below it rather will establish the connection with n3 and a1. Thus saving time and save which is smart work.
10. Other Commands when pushing or pulling.
    1. docker login
    2. docker logout