Identify the 5 logging levels in Spring boot:

Logging

Log4j

SLF4J provides 5 default logging levels in Spring boot:

**ERROR** - Error logs are serious issues that affect a significant part of the system or some part of your application has failed to operate. Exceptions are considered ERROR level logs. Other examples of error logs are database connection failures and configuration errors. ERROR logs are the most urgent default log level in SLF4J.

**WARN** - Warning logs are used to indicate potential problems that might cause errors and should be monitored in case they fail. Of course, the context of a warning is subjective to the developer and the situation so warning logs might vary from system to system.

**INFO** -INFO is the default logging level that is set by Spring Boot. If no configurations are made, the log level is automatically set to INFO. These types of logs are information that isn't normally needed but is useful in situations like production code debugging or determining when certain data is manipulated.

**DEBUG** - DEBUG logs include more detailed, specific information that isn't needed in normal situations. This is often set as a log level when a developer is trying to deep trace a problem or a bug that is hard to trace.

**TRACE** - TRACE is a more granular version of DEBUG. TRACE logs are exhaustive, imagine logging every single operation the system is doing, from starting a service, initializing new variables, and calling methods.

Steps:

No need to add dependencies in pom.xml for logging (in case of spring boot)

In application.properties:

logging.file.name=ajay.log

in any method, obtain the logger factory to get logger.

Logger logger = LoggerFactory.*getLogger*(Application.**class**);

logger.error("There is a problem");

Task:

Create a docker image of a spring boot application and push the image to docker hub:

If it is a spring boot project, an easy step to create docker image is using mvn command (no need of Dockerfile)

mvn spring-boot:build-image

1. Generate .jar file

Go to cmd where pom.xml is present:

mvn clean install

1. Check targer folder that it contains a .jar file now
2. Create a docker image of this project:

mvn spring-boot:build-image

we have got a docker image: 07-jun-h2-demo:0.0.1-SNAPSHOT

docker images

docker image ls

docker container ls

docker ps

docker rm <<container id>> -f

docker rmi <<image id>>

How to create docker image of an application using docker file:

1. Create a spring boot project
2. Create a Dockerfile in location where pom.xml is present. If you right click the project and create file, it will be there only.
3. Inside Dockerfile:

FROM java:8

EXPOSE 5000

ADD target/13-rest-4-docker-1-0.0.1-SNAPSHOT.jar 13-rest-4-docker-1-0.0.1-SNAPSHOT.jar

ENTRYPOINT ["java","-jar","13-rest-4-docker-1-0.0.1-SNAPSHOT.jar"]

1. Note the .jar file names in 3 locations are the .jar file we generated using

mvn clean install (in cmd) or

right click project Run As -> Maven Build (goals: clean install)

1. Now, lets create docker image: go to cmd where Dockerfile is present

docker build -t 3-rest-4-docker-1 .

now, the docker image is created.

1. To push this image to hub.docker.com,

We need to create a repository (same name is good)

Tag the local image to docker hub repository:

docker tag 3-rest-4-docker-1 jagindia/jag-ust-demo

push the image to repository:

docker push jagindia/jag-ust-demo

1. docker images
2. run docker image:

docker run -p 5000:5000 jagindia/jag-ust-demo:latest

1. go to browser:

localhost:5000/hi

Deploy a spring boot rest api in AWS Elastic Beanstalk:

Steps for deploying this .jar in AWS Elastic Beanstalk:

1. Login to aws.amazon.com management console
2. Search for “Elastic Beanstalk”
3. Create an application (ideally same name as the project name)
4. Create environment:

Platform: Java (for .jar and tomcat for .war)

We have chosen java platform now

Choose sample application and deploy (later we can upload jar)

<http://ust-demo-1-env.eba-jgtkx6u6.ap-south-1.elasticbeanstalk.com/hello>

Task:

Create an EC2 instance of Linux OS and connect using SSH client. Then install docker in the EC2 server.

Steps:

**Step 1: Launch an instance**

You can launch a Linux instance using the AWS Management Console as described in the following procedure. This tutorial is intended to help you quickly launch your first instance, so it doesn't cover all possible options. For information about advanced options, see [Launch an instance using the new launch instance wizard](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-instance-wizard.html). For information about other ways to launch your instance, see [Launch your instance](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/LaunchingAndUsingInstances.html).

**To launch an instance**

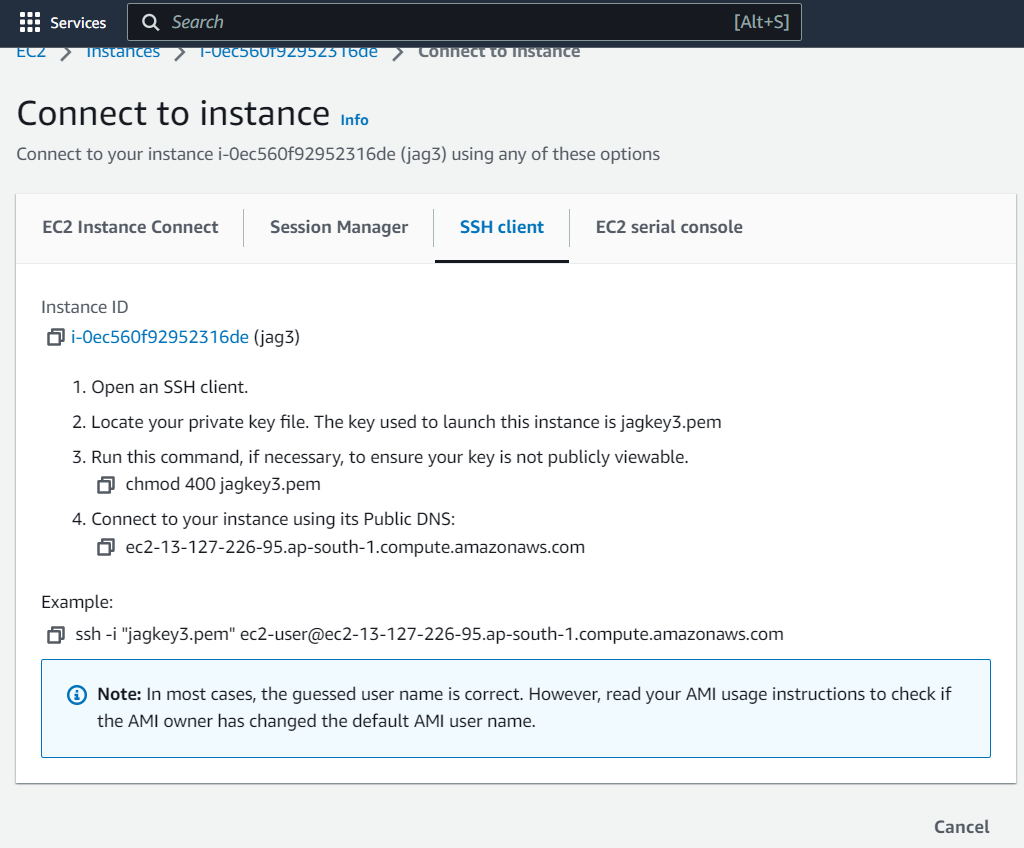
1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. From the EC2 console dashboard, in the **Launch instance** box, choose **Launch instance**, and then choose **Launch instance** from the options that appear.
3. Under **Name and tags**, for **Name**, enter a descriptive name for your instance.
4. Under **Application and OS Images (Amazon Machine Image)**, do the following:
   1. Choose **Quick Start**, and then choose Amazon Linux. This is the operating system (OS) for your instance.
   2. From **Amazon Machine Image (AMI)**, select an HVM version of Amazon Linux 2. Notice that these AMIs are marked **Free tier eligible**. An *Amazon Machine Image (AMI)* is a basic configuration that serves as a template for your instance.
5. Under **Instance type**, from the **Instance type** list, you can select the hardware configuration for your instance. Choose the t2.micro instance type, which is selected by default. The t2.micro instance type is eligible for the free tier. In Regions where t2.micro is unavailable, you can use a t3.micro instance under the free tier. For more information, see [AWS Free Tier](https://aws.amazon.com/free/).
6. Under **Key pair (login)**, for **Key pair name**, choose the key pair that you created when getting set up.

**Warning**

Do not choose **Proceed without a key pair (Not recommended)**. If you launch your instance without a key pair, then you can't connect to it.

1. Next to **Network settings**, choose **Edit**. For **Security group name**, you'll see that the wizard created and selected a security group for you. You can use this security group, or alternatively you can select the security group that you created when getting set up using the following steps:
   1. Choose **Select existing security group**.
   2. From **Common security groups**, choose your security group from the list of existing security groups.
2. Keep the default selections for the other configuration settings for your instance.
3. Review a summary of your instance configuration in the **Summary** panel, and when you're ready, choose **Launch instance**.
4. A confirmation page lets you know that your instance is launching. Choose **View all instances** to close the confirmation page and return to the console.
5. On the **Instances** screen, you can view the status of the launch. It takes a short time for an instance to launch. When you launch an instance, its initial state is pending. After the instance starts, its state changes to running and it receives a public DNS name. If the **Public IPv4 DNS** column is hidden, choose the settings icon ( Settings icon. ) in the top-right corner, toggle on **Public IPv4 DNS**, and choose **Confirm**.
6. It can take a few minutes for the instance to be ready for you to connect to it. Check that your instance has passed its status checks; you can view this information in the **Status check** column.

After launching EC2 instance, connect to the instance using SSH client:



In cmd: paste the example command as seen above.

INSTALL Docker:

Task:

Create an EC2 instance of Windows and connect using Remote Desktop and install jdk in the server.

Steps:

In aws management console, we can start an EC2 instance

1. Go to aws.amazon.com management console
2. Search for EC2
3. In the EC2 dashboard, there are 0 instances running. Click that
4. Click Launch instance button
5. Enter the name of server
6. Choose OS. We have chosen “Windows”
7. Instance type: by default it is “t2.micro” you can choose any of the type based on the RAM size
8. Key Pair (login):

Create new key pair button

Give name of the key pair

Choose .pem

Click button to download key pair file

Now check downloads folder. There is jag5.pem file

1. Network settings (click edit)

Auto-assign public ip (it is disabled. So enable them)

Security group: select existing security group (and choose vpc)

Create instance.

1. Now in the dashboard, you can see 1 instance

First status is pending. Then changes to “Running”

1. Click the instance id hyperlink

Connect using RDP Remote desktop app.

For password, upload .pem file and generate password for administrator.

Task:

Deploy a containerized spring boot REST api in AWS Elastic Container Service:

Steps:

Create a docker image of a project

1. mvn spring-boot:build-image
2. docker tag ust-mvc-2:0.0.1-SNAPSHOT jagindia/ust-mvc-2
3. docker push jagindia/ust-mvc-2

docker.io/jagindia/ust-mvc-2

1. You should have a docker image in hub.docker.com or ECR.

jagindia/jag-ust-demo

this will run in 5000 port and url is “hi”

1. Lets login to aws management console.
2. Search for Elastic Container Service
3. Click Task Definitions
4. Create new Task Definition
5. Type the task definition family name (ex: ust-mvc-2-task)
6. We can add 1 or more containers to a task. So under Container – 1
7. Enter the container details:
   1. Name: ust-mvc-2-container
   2. Image URI: docker.io/jagindia/ust-mvc-2

In case if ecr url is provided, it should start with ecr.io/

* 1. Container port: you can add one or more ports. We add 5000 port here
  2. Leave other values as default and Click Next
  3. Environment choose: AWS Fargate (serverless)

Linux CPU: 1 Memory 3GB

* 1. Task role: choose the only available role (ecsTaskExecutionRole) in both task role and task execution role
  2. Storage: amount must be between 21 and 200
  3. Click Next
  4. Review the details and click “create” button

1. Now, create cluster.

Enter the cluster name

Networking: choose the default VPC and subnets

Click “create” button

1. Once the cluster is created, open the cluster and go to “Tasks” tab (2nd tab) and run the task we created. Ensure that the execution role is selected.
2. Running state will be changing from
   1. Provisioned -> Pending -> Activated -> Running status.

Once it is in running status, click the task hyperlink and go to configuration page

Copy the public ip address.

1. Go to browser and type

http://<<public ipaddress>>:5000/employee

1. Delete cluster and go to tasks, click the task and in Actions: deregister