Q1. How do you secure API endpoint? [they are expecting JWT concept]

Q2. Say there are microservices A, B and C. A internally calls B, B internally calls C. You only call A. How do you ensure that you will show some custom response instead of any exceptions in case there is some exception occured in B or C.

Q3. If you have the scope to design the above A, B, C services. How would you do that.

[Focus on Circuit Breaker concept, Exception Handling using RestControllerAdvice, API gateway. Actually, focus on Spring Cloud components]

Q4. How your microservices get deployed?

Q5. Have you written Jenkins script or know the things?

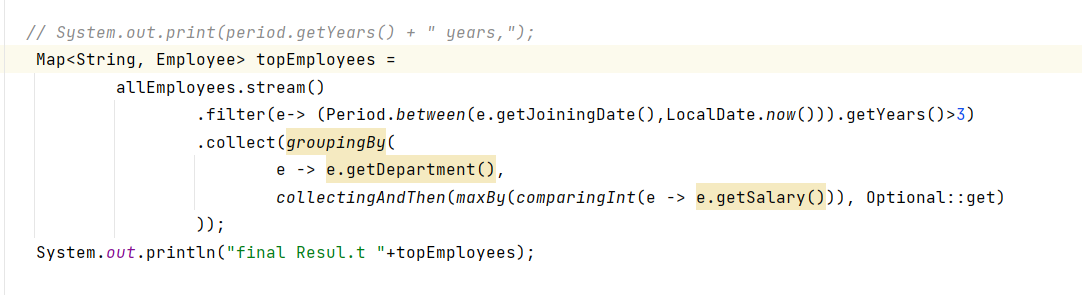
Q6. How do you test your functionalities?

Q7. GIT commands- add, commit, push

Q8. How do you resolve merge conflicts

Q9. Consider there is an Employee object having fields – name, salary, joiningDate, department.

Now you have to write a method in java to find out the employees from each department who is getting highest salary department wise and served for more than three years.



Q10. Write database queries for the above problem.

SELECT empName,empDept,EmpSalary

FROM Employee

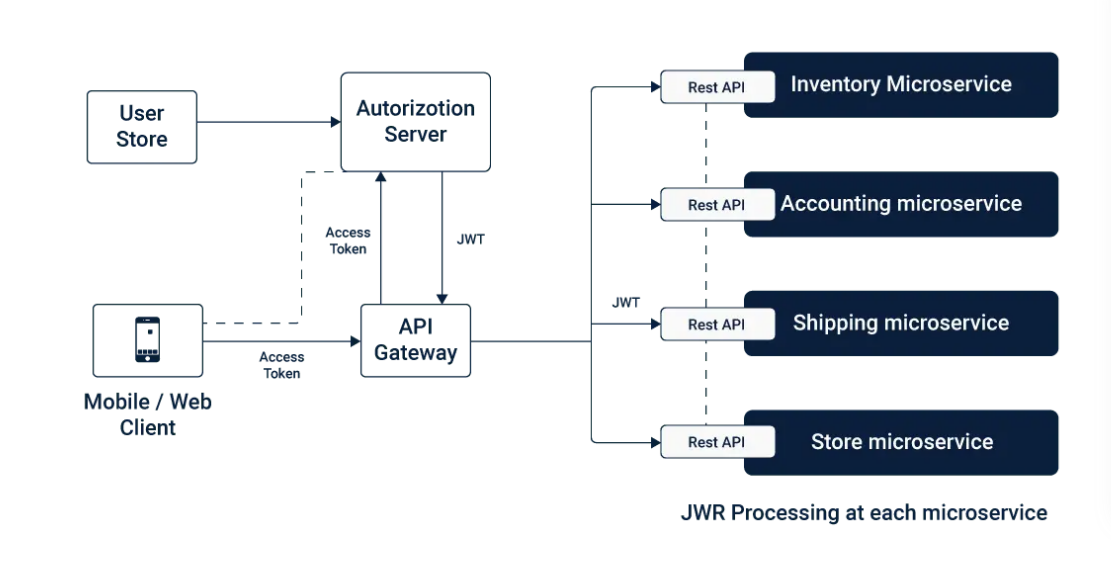
WHERE empSalary IN

(SELECT max(empSalary) AS salary

From Employee

Where datediff(yy,getdate(),date\_of\_join)>=5

GROUP BY EmpDept)



https://searchitoperations.techtarget.com/tip/Follow-these-6-steps-to-deploy-microservices-in-production

1. Use cloud services for production Infrastructure
2. Design for failure
3. Decentralize data management
4. Distribute governance
5. Automate infrastructure deployment , embrace CI/CD processes
6. Monitor log and troubleshoot from the start.

Microservices deployment :

* Servics may be RESTful , scheduled Jobs event based etc.
* Differenct server configuration and scaling requirement .

Virtual Machines :

* Every Microservice gets its own virtual machine.
* Each service is isolated with own CPU ,memory and server configuration.
* Scaling can be done through increasing or decreasing the size of VM.
* **IaaS hosts offer a mechanism for auto scaling.**

Netflix created open source application – animator to simplify the process of creating EC2 instances.

VM became very costly and difficult to maintain.

So we moved with Dockers.

Containers are fully isolated processes, require less resources and size .

Share resources and setting configuration for scaling

Container Orchestration

Application in well packaged standalone containers is first step in managing multiple microservices.

Configuration is mentioned in YAML or JSON file.

* Where to download the container image
* How networking between containers should be handled
* Storage and where to push the logs
* It can be easily include in your CI/CD pipeline

Once containers are configured and deployed , orchestration toll manages lifecycle.

Includes starting and stopping service.

Restarting the containers after failure , scaling up and down .

Support all AWS , GCP and Azure

Container Orchestration Tools are

Amazons EC2

Docker swarm

Apache Mesos

And popular – Kubernetes

Kubernetes – created by Google .

Azure (AKS) and Amazon (EKS ) adopted support for it.

<https://dzone.com/articles/deploying-your-microservices>

Function as Service

FaaS or servless is anew and fast growing option.

Serverless pushes the smallest possible component .

Easch microservice will bedeployed to Faas like AWS Lambda.- has step functions

Google launched Google Cloud Functions and Azure Functions .

Lamda functions are stateless Service and the code package is uploaded in to AWS and executed based on trigger.

* + Directly through CLI request
  + HTTP request reouted through an API Gateway
  + Schedule (CRON Job)
  + Cloud Event from another service like S3, Simple Email Service and DynamoDB etc

There are multiple options to Streamline deployment including Amazon SAM(Serverless Application Model), ClaudiaJS,Apex.

We can use Terraform or Cloud Formation to define Faas Infrastructure.

Cost is much reduced.

Disadvantage :

Not intendedfor long running processes.

Limitation on Language support