Meeting Transcription

Meeting started: Jun 11, 2025, 5:57:11 PM
Meeting duration: 102 minutes
Meeting participants: Angel Hsieh, David Lu, Hao Cheng, Jiawei Zhu, Junbo Zhao, Ruoming Zhang, mq shi, ≣ ≣ ≣

View original transcript at Tactiq.

Transcript

00:00 David L.: Okay, I'm giving everyone let's start. Um, Any questions before we continue on the training session.
00:27 David L.: Okay, so let's continue discussing, all the things that we have done last time. Anyone can share the screen Houghton.
00:56 Hao C.: Okay.
00:56 David L.: How to Hi, Can you share your screen? Let's go through the things we have done last time.
01:02 Hao C.: Okay. Can you see it?
01:22 David L.: Yeah, I can see. Let's go.
01:25 Hao C.: Okay. Ah, so I build another spring project.
01:30 David L.:
01:31 Hao C.: It basically it. It's like a proxy. When you when we use postman to make requests who will make requests to this
01:34 Speaker : Hi, I'm transcribing this call with my Tactiq AI Extension: https://tactiq.io/r/transcribing
01:41 Hao C.: this project it's running 81. And it's simpler. It has only two layers controller and service. the control area, basically the same as the other project, same GDP methods, and
01:58 David L.: And before that, check the REST template configuration.
01:59 Hao C.: then you service, but in the service, instead of using the DO to access the database, I used a rest template. So basically it will make it will make a

further HTTP request to this location, which is the other project. And then it will return the response, it will accept the response and then story here. And then we return back to controller and then return to the users.
02:37 Hao C.: um, Let's let's just do a demo.
02:47 David L.: Where do you configure your rest template?
02:55 Hao C.: My rest template. I don't think you can figure that.
03:02 David L.: Do you have any customization for the rest templating? The config package. What's inside a compact config
03:08 Hao C.: I'll just for the multiple date the, the database. All sorry. Sorry. I forgot I forgot. Yeah, this is the configuration class. So
03:14 David L.: Well.
03:23 Hao C.: we annotate this method as it being so that when we return this rest template object, it will it will be registered into the container. As a being. I don't do. I didn't do any configuration, I just simply return it and then use it.
03:43 David L.: Okay. Let's verify the functionality first.
03:51 Hao C.: Okay, functionality, give me a demo.
03:56 David L.: Yeah.
03:57 Hao C.: Okay. Right. so, I get all So the proxy will add another string just to distinguish from the original project. So that is is showing the proxy is working.
04:24 David L.: Okay.
04:27 Hao C.: Let's see. No. To. yeah, and then Create. 11 and 10. Yeah, it's there. And then put try 11. Yeah, it's working. And then delete 11. Delete.
05:52 David L.: Let's check, what's the problem?
05:54 Hao C.: I don't know what's going on 11. Do you want me to?
06:26 Hao C.: develop this, or Okay.
06:30 David L.: How are you going to about this?
06:33 Hao C.: I think I can first go to the controller and then
06:39 David L.: No, the first thing that you're going to check is the logs, right?
06:44 Hao C.: Oh yeah, right. Nobody. Oh, okay. Requests for So it looks like the problem is on 88. Right.
07:10 David L.: Mmm, I doubt it.
07:14 Hao C.: What does this? Nobody means.
07:17 David L.: Hmm. You're hitting your delete order restful endpoint, right?

07:27 Hao C.: Yes. Oh wait. Oh, it's working now.
07:39 David L.: What did you do, work? What's wrong before?
07:39 Hao C.: To. Was it is get. I forgot. I didn't change anything. I think.
07:53 David L.: You didn't change anything, but what's the mistake that you made? Can you go back to the log again? What? No, just what does it say?
08:26 Hao C.: Nobody with root cause 404 on get requests for 8080.
08:33 David L.: So, what's the problem?
08:37 Hao C.: So this should be a delete.
08:41 David L.: so why it's a get Because you chose the wrong endpoint to hit when you are in the postman.
08:57 Hao C.: Yes.
08:59 David L.: So remember what you're doing and when you do it, um, remember to check the lungs? The long says, Always good things. Don't be panic Any of the 500 level. 400 level exceptions. Exception is just exception. It just being a problem. Surreya whole Korea passed your meant to solve all the problems. It could be very It could be very slim to see that the reality is you never deal with any of the exception of mistakes. You always deal with the bugs, mistakes problems.
09:35 David L.: And logs is always a helpful for contains. Always contains the helpful information. When you do logging, you shouldn't just log all the failed. Only the failed exception. Exceptional Request. You should also log. All those successful requests. Otherwise, it would be very confusing. Why in your current ID? There is only one exception's yet. It's showing 204. Any questions on this practice?
10:20 Angel H.: I have a question.
10:21 David L.: Hmm.
10:23 Angel H.: um, when I was doing this, I'm it's different but I try to try to do to to serve to the today, I based this time And I was, I was thinking about if, like meantime, we are writing the data to the database, right? What if they're one of my database died? And then if I'm like doing the post and then the the like the the UID of the, the things I'm throwing to the database, is it gonna be like the order is gonna be a different? Do you understand what I mean?
11:05 David L.: Not really but you ideas Unordered.

11:08 Angel H.: Yeah, just like okay, so if I'm writing the like, the order of my like I buy a water and I and I'm writing to the database, right? And then I have my sequel and I also have like the posts. Great sequel. And then what if my sequel like went down? And now I have only the the postgres sequels but those two database should be the same. Like every time I throw post a data inside and then so my ID would be would be different if I'm using like the database general, like automated generate In the order.
12:03 David L.: So first, does anyone there stand the problem? Folks, your phallus has a concern. You get what she was concerning?
12:15 Ruoming Z.: Angel. Does Angel use? Two databases.
12:22 Angel H.: Oh yes.
12:24 Ruoming Z.: I'm using just one. So
12:27 David L.: Doesn't matter. She has an engineering problem.
12:33 Ruoming Z.: Okay.
12:34 David L.: This is kind of the thing you're going to deal with the interviewers during the interview. They were just pop out a scenario for you to analyze. The first thing that you're going to use, get to requirement. What does Angel just sent? What's the problem?
13:00 Jiawei Z.: Essential mean that if she has two database and one of them down during the execution of the insertion, and what will the, the id order still in order? Is that her question?
13:19 David L.: Mmm, 80%. Um, So before that, it's all correct. And you one of those database down, right? and what she said is that, Well, later, there will be other new orders comes in. And then maybe the sequel to my sequel. Database is bring back and the new data is inserted. but now we have a mismatch for the order numbers because one, One order that do insert it. Hmm. Into the postgres, doesn't. ARE is not inserted into the MySQL database.
14:01 David L.: There are other new orders comes in both the Postgres and SQL database. Hence, The order is going to have a mismatch. Makes sense. Solutions.
14:27 Hao C.: I think we can have a Some function. Periodically check compared to database to see if they match.
14:36 David L.: No.
14:38 Junbo Z.: Can we use transaction?

14:38 Jiawei Z.: Is it possible for sorry?
14:42 Junbo Z.: Okay. Can we use transactions narrow like this like between different systems?
14:51 David L.: No.
14:51 Junbo Z.: To make the okay.
14:55 Jiawei Z.: Is it possible for us to is a requirement about async communication between services or service to database.
15:05 David L.: The problem is that we have a problem for our web application. The user is complaining. Sometimes they get to receipt, sometimes they don't get the receipt, sometimes they submit and see, the orders confirmed yet. They didn't receive the items.
15:21 David L.: That's the complaint. Does user even care about whether this is written in Python or Java. Do they even know what is Python or Java? It's a problem. It's a complaint. Makes sense.
15:38 Jiawei Z.: Yes.
15:47 David L.: An engineering problem, always contains the multiple factors, you're going to analyze. Transaction will be a very good. I'm very happy to hear that because the first reflections, Some of you guys generated is close to the solution part of the solution. But remember, any of the real solutions is always trying to? Trying to solve a comprehensive engineering problems. Things we get this questions. We are going to do some analyze training. Some of the interviews, you sent some of your real world working engineering, excuse that.
16:34 David L.: Can you bring me to a session with the whiteboard, please?
16:53 Hao C.: https://code-with-me.global.jetbrains.com/Pr3A6Mnz55xQuDytfP63WA
17:02 David L.: Let's just go to your screen boot class.
17:35 David L.: Um, I'll just mess up a little bit. um, See that right? Okay, cool. I'm just going to mess up a little bit. Don't worry, it's just in comments. All right. So remember, for any of the engineering problem? You first need to know what is the problem, right? So drop down.
18:08 Hao C.: just,
18:09 David L.: All the requirements, all the scenarios, or the given information. We have. PostgreSQL. Plus. My sequel, right? One of them. New Order Internet

into postgrad. But not. Other new orders comes in. Order mismatch. Other ID best match. Make sense.
19:05 Hao C.: Makes us.
19:07 David L.: so remember every time when you are taking an interview or when you are having a meeting with well, remember This has already been converted into an engineering problem, right? European mind. Just complaining that. Well, we're just assign a ticket. The ticket is going to have customer complaint.
19:31 David L.: so, imagine what kind of the complaint customers going to have is Big hole your customer service. Provide. All their id. Provide credit card. Purchase history. No item. Or sometime soon received. Right, or sometimes they sure with the screen shot. Order completion. But customer service. Cannot find that order. Inventory. This will be in the reality, right? And there will be boundary of the screenshots sharing with after the tickets.
20:49 David L.: Well, like other ID like credit purchase history like no, no item received there will be screenshot from the user saying Hey This order is confirmed with the same order ID that they probably provided at here. They will also be customer service will have their own UI. Inside that's you on. There will be another screenshot saying that Hey, we cannot find the order based on this order. ID make sense That's in real. What's going to happen? Right. It's It's not a pure engineering problem. It's a web application complaint. You need to convert it into an engineering problem.
21:44 David L.: And interviewers is very likely and king on checking the candidate, whether he or she is able to do this step. during the interview every time when you're doing a communication for your coding, During the interview. during interview, every time you do a communication with your, with your interviewer on, For any of the problems they're trying to give you regardless how simple the problem is. If it's simple then you just need to drop one or two lines.
22:49 David L.: If it's multiple, it's a completely complicated question. Then you're going to drop down all the original information. And if there's any follow-ups for your concerns, You can also drop it down after you have the confirmation from the interviewer. That's step one. every one of you is trying to get the solutions, trying to pop out the terminologies, trying to figure out a perfect solution, the first time that you heard the problem, That's the wrong mindset.

23:24 David L.: Remember before you are going to build up your book, you have to collect your dust. Without dust you cannot make a break. Make sense. Now. This is the problems. Now, how many technical problems that we really have here? Scroll down. Now, you see this is the business problem, this is your engineering problem. What's your technical problem? Technical problems. Hmm.
24:15 Hao C.: Database is down.
24:18 David L.: Correct. You don't mind. That's problem. Number one.
24:27 Hao C.: Are mismatch. Between the
24:29 David L.: Mmm, this match is a what?
24:32 Hao C.: Inconsistency.
24:34 David L.: Now.
24:55 David L.: Solution for this one.
25:04 Junbo Z.: Log.
25:08 Hao C.: Use replica.
25:16 David L.: Well, first, you need a login. Sure you said use the replica. What do you mean by use the replica?
25:22 Hao C.: So if one DB is down, we can try to send another request to to its replica.
25:28 David L.:
25:29 Hao C.: And then later on, when the, when the first dB come back we can sync them.
25:39 David L.: Keep going.
25:44 Hao C.: Trade-offs replica. Trade-offs is
25:53 Junbo Z.: Is money.
25:55 David L.: Sure. Anything else?
25:58 Hao C.: Time. Because you have to send another request.
26:03 Junbo Z.: Performance.
26:03 David L.: Mmm time is not accurate enough. We say that this. is a issue like
26:08 Hao C.: Oh yeah. Right.
26:13 David L.: And what what is this performance issue? What is this performance concern?
26:19 Junbo Z.: So you change the value in one of the database, you have to change the simultaneously on the replica of the OH no rate, just right?

26:26 David L.: Yes. This is another data consistency problem, right? And well, this concerned depends on the situation. Think about it data.
26:36 Hao C.: Yeah.
26:42 David L.: Consistency is a thing that you need to do the trade-offs versus the performance. Why did I say that? Remember we talked about, We have all those replicas if we want to have all those data consistent before we have all those replicas available. Then, it must have a procedure to consist of those data. Which means the database cannot be always available, right? And if we want them to be always available, that means sometimes it's inconsistency. So eventually, this is a problem with the real time versus eventual. Consistency.
27:28 David L.: okay, so real timings that Sorry. Put it into one sentence. Real time consistency versus eventual, consistency. Real-time consistency is saying that, hey, I don't care. when there is any of the changes for the records, in my database, once there is a change, I want to ensure that all the replicas Are consistent. And then the user can get whichever the record they are trying to search. Eventual consistency is saying yes, there will be some inconsistencies in my system during the wrong time.
28:20 David L.: But eventually those data changes in different replicas is going to be consistent. Which one is getting a better performance or the DV query.
28:36 Hao C.: Eventual.
28:37 David L.: Right. Okay. Any other solutions?
28:51 Hao C.: I don't know if this right, but Partition of a sharding.
29:00 David L.: Sharing Replica, You're all saying that we're going to start a data. Instant, another data instance, to prepare for the shelling downtime, right?
29:12 Hao C.: Yeah.
29:16 David L.: What else?
29:22 Hao C.: Oh, I think cash can can solve the mismatch problem, but it doesn't solve the
29:28 David L.: Cash is going to work, but it's a s***** solution. Not going to be used. So first, let's talk about your implementation how how are you going to use cash to help you do that?
29:57 Hao C.: problem that user don't get the response. so, what I was was thinking was when the requests failed The system can store the the requires into

cash temporally. and then when the system, when the database come back, it Sends the data to database from cache so that they will be matched. That doesn't
30:23 David L.: Hmm.
30:26 Hao C.: the problem. The user don't get response.
30:30 David L.: We're not trying to solve the whole problems, right? We're trying to solve the DB downtime. the cash layer is going to cash in up all those
30:35 Hao C.: Because it's expensive. And
30:40 David L.: New orders temporally, but I said, we are not going to use it. Why?
30:53 Hao C.: And then we don't know how how soon the database can come back.
30:59 David L.: No, it's temporarily.
31:02 Hao C.: Oh yeah.
31:07 David L.: Just in case, or let's say for some downtime issues is going to take a long time fix. What's going to happen?
31:21 Hao C.: To casual full. The system will not work because the cash is running in the memory.
31:32 David L.: When it's full, it will crush when a crush what's going to happen.
31:38 Hao C.: the whole, the whole system turned down, Not just the TV.
31:43 David L.: Not necessary, the whole system, you can have a dedicated server running a cache but the data is all loose, right? Assuming that storing the order rackets, all
31:49 Hao C.: Oh yeah. Right.
31:52 David L.: those means money And your solution is saying that. If we cannot fix within three hours, then we lose all those monies. Storing the cash. That's not acceptable. no, mention that all like you said, All the things that you stored in the cache is in memory that's in RAM. Quite expensive to store. Maybe the order is only worthy for two dollars and you're spending $4 to store it. Right.
32:24 Hao C.:
32:25 David L.: By the way, the red is is a in-memory cache but you still have to wait the rof to flush it into the file. Or into the hard disk. Okay, but technical is speaking. Your cash should not working as a buffer but ideas, correct. The mindset is correct that you should have a buffer. Buffering pool. Okay. buffering proof is always a good thing or you can have a standby

32:55 ䷀ ䷀ ䷀ : Myself.
32:56 David L.: Bye instance. These are the same thing in buffering. Pools can be different things like and messaging, queue. Like a file. You can write all the things into the file with another thread. Every time when the DB is down, new record comes in, you have a dedicated strand standing by and then Writing the New Records into the File. Meanwhile, your server is actually still running and handling those new orders.
33:29 David L.: Well eventually it's going to have a problem for performance again because any of the IO from your software JVM to the hard disk is going to be slow, but at least the functionality is completed, and you don't expect your DB downtown is going to be long and frequently. having this rant just standing by will not hurt your performance in the daily life because You just create some of the threading, one of the relatively small spread pools, nothing or you can just have a cash display pool.
34:10 David L.: That contains zero threading. The core memories core pool size and That's just going to be any of this brand that's going to be created. When your TV has a downtime hence, Your daily during the daily normal time. You have it, you your web application is not going to sacrifice much of the performance. Make sense. And then this data inconsistency. just in case you get confused, Solutions.
35:02 ䷀ ䷀ ䷀ : Can we just manually insert the losing data into a? When the when I felt
35:09 David L.: Right. Anything else? Any other solutions?
35:35 ䷀ ䷀ ䷀ : database can be used again, we can just mark, mark this data and later we manually insert the stage
35:51 David L.: Okay, so many ways, never going to be a solution. You are very expensive. I'm not expecting you to do things manually. You're dealing with application, right? And hiring you is not, it's it's not cheap. You are very expensive. I act all the things in automation style rather than manually.
36:14 Junbo Z.: Does.
36:16 David L.: Hmm. How this API gateway helps for the data and consistency sent here?
36:17 Junbo Z.: Does API, Gateway helps? In this, not sure.

36:34 David L.: The only random shooting. Think Logicaly. Probably the UID is the most practical solution. Why we want all the orders? Other ID to be in order that just doesn't make any sense. If you do here about the practical, If you do care about the order arrival, then you should use the timestamp. To order it. Makes sense.
37:14 Hao C.: Yeah.
37:16 David L.: Rather than use the primary key with increment. Ascending order to indicate which order comes first. Right? You can even have another column that's storing the timestamp. The timestamp when you sort it based on timestamp, you know, which order comes first Makes sense. So why it's a good practice? Well why it's feeling hard to understand at the
37:51 Angel H.: Yes.
37:56 David L.: very beginning And it's hard for you to cash it up. Because you're, you're all missing those steps and then just jumping to those solutions. And Sir Transaction also works, right? But now you have to in, you have to well Why I said you ideas more perfect. Let's just talk this a little bit more. Hey Trend. Why translate it's doable. It's featable, but challenge. Why?
38:36 Junbo Z.: Because transaction is a real-time consistency.
38:39 David L.: Yeah, but it's not challenging. Right? In your spring boot. You're just going to use and add transactional, annotation.
38:49 Junbo Z.: It lowered a performance because you have to make the entire thing atomic.
38:56 David L.: Right? But lowering performance is not always a bad thing. If you have to trade off the performance, through a Thomas Lee, For your system. Then it's a must Imagine that there is a training system that you're going to do you really want to compromise the real-time consistency just for the performance.
39:18 Junbo Z.: No.
39:18 David L.: Think it again. That's the reason why a lot of the training companies, especially for the high frequency trading companies are not going to use Java at all. They're using c++. There's a reason why they chose to do it. Make sense. What's the technical implement or the implementation challenge in here if we want to action? because in this case, the at And transactional is not going to help them much. Why? Because we have two database. And we implement a

distributed. Distributed transaction. Right.
40:18 Hao C.: Yes.
40:19 David L.: now, you can implement either the two-phase commit Or Saga that I'm Helen. the song I did on calendar is going to have either the orchestry in style, or Corey Coriography Style. Okay. Those two solutions are skew a mitigating of the distributed transactions, it will still be trade-offs. And when you are going to implement those two that I'm patterns for distributed transactions, it's feasible. People are doing it, but you need to put dedicated engineering workload to that implementation. And again. We have solutions here, which is more practical easier. And robust. Makes sense.
41:40 Hao C.: Yes.
41:43 David L.: During the interview, you are going to do this, rather than tell me to use the transaction to use the chart to use the replica. But it's a good question that we have the practice. Thanks Andrew. But by the way, does This resolve your concerns. Anything.
42:06 Angel H.: Oh, yes. No. I think this is the correct solutions. I was thinking because I
42:12 David L.: This is not very correct, but it's practical. And
42:12 Angel H.: think you
42:17 David L.: any of the problems, any of the real engineering engineering problems takes time and energy to resolve but the mindset is important, otherwise if you're pointing yourself to the wrong
42:30 David L.: direction, There'll be a huge problem. Makes sense.
42:44 Angel H.: Yes.
42:45 David L.: So, remember to practice your typing. you're going to talk while typing and you're going to communicate with your interviewer, whatever, the things that I have done is pure improvise, But you see what I have done. Whatever the things I have done today. For this case, it's still in low pressures, why? Because no one is challenging me. No one is communicating with me. I'm, I'm proactively doing communication with you, but during the interview, it's a high pressure environment. Your interviewer is going to challenge you. He or she is going to check why you are doing this while you are doing that. Meanwhile, you think you need to talk, you need to drop down all the notes. It's going to more be

more and more challenging.
43:42 David L.: For now, what you're going to do again, prepare all those contexts. Prepare rehearsal all those questions. Fix all the missing gaps. Also, practice on your typing. Do not use just two fingers to type. And when you are typing at here, why I'm doing Is everything in full context. While you are doing. Remember to use the short hands rather than the full context do not always type the whole sentence. It's meaningless. Reasonably using the copy and paste to save your time.
44:25 David L.: during the during the note note dropping, Are we clear?
44:34 Ruoming Z.: Can I ask a question?
44:35 David L.: Yeah, sure.
44:36 Ruoming Z.: so in the row interview, I was supposed to write down those, the engineering problem business problem, technical problem or just a technical problem because the time is limited
44:49 David L.: The time is limited and that's the wrong idea. Whatever the time is, if you only left three seconds, you still need to collect the dust and then view the brick. Just because you have a short amount of the time so that you can skip all those steps and jump into the solution. That's a fascination, but it's not going to happen. You are an engineer rather than a magician.
45:20 Ruoming Z.: Okay, so we should start with the business problem.
45:25 David L.: The interviewer is going to bring you into a scenario, maybe they start with the engineering problem, maybe they start with the business problem. Maybe they just jump out with a technical problem and ask you to solve it. Whichever the scenario is We are doing a full process. I am doing a full procedure analysis, for you, as an example. So that, you know, what you are going to do. Whichever the steps they breaking, you always drop down the requirement analyze And speak out your ideas.
45:58 David L.: So that's the interviewer is going to be able to follow because at the end it's not just about finding that so called perfect answer, there's no perfect solutions. Sure, we bring this back to my team and we discuss about the solution that was tubing. Millions of the things we're going to discuss. but, the idea is that the interviewers seeing your mindset, Visually. Legitly on paper. And you're able to go through all those stamps, logically thinking providing different options and

clearly knowing what are the trade-offs.
46:41 David L.: For different options. And when are you going to use which option? Does the interviewer really care about these solution? Sometimes they do there will be some of the jobs. Hmm. That's really seeking for a solutions. Because they have tried. They cannot find a solution. Hence they file this position. find someone that can solve this problem immediately, but that's a relatively Small Cases, Most of the cases we are checking whether you are able to pull the things out.
47:20 David L.: Your daily life is not necessary to be challenging everything. It's not going to be designing every day, that's more of my life rather than you guys. But at least we want to check whether you have to clear engineering mindset. Makes sense.
47:39 Ruoming Z.: Yes, thank you.
47:40 David L.: Hmm. Practice your typing, okay? Type is, you have worked before, do not type as like a new grant. Every few letters and then you have to, I don't know. Okay. Let's move on caution.
48:13 Hao C.: Okay. Um, I think that's I explained. The another service.
48:28 David L.: Right. What's next?
48:29 Hao C.: All right. Okay, I can do the sorry. So, I can, I can talk about the Microservice.
48:45 David L.: It's a little bit hard for me to read all the things. Can you share me or my real link? It will be easier for me to
48:51 Hao C.: Okay.
48:57 Hao C.: https://miro.com/app/board/uXjVIp_D6xE=/
49:07 David L.: No access request access. Um, can you share to my email?
49:31 Hao C.: I think you want me to pay for the hearing.
49:35 David L.: Right. Just try my email. Let me see, what's my email and here. Send over here.
50:03 David L.: david.lu@rothurtech.com
50:04 David L.: I stand over the email in the Google Chat. You can try that.
50:24 David L.: To do do, do do, do the culture tomorrow?
50:31 David L.: Okay.
50:33 Hao C.: Okay. Can other people see the diagram?

50:42 Ruoming Z.: Yes.
50:44 Hao C.: Okay.
50:44 Angel H.: Yeah.
50:46 Hao C.: so, This is what a microservices look like. The client will send the request. And then it will first go to the load balancer. It will split the traffic to multiple difference. In this case, I have three API gateway and then the API gateway will do authentication routing and then Rate, limiting and aggregating the results from from the service. and the service discovery is a service that Help one service to find another service. So the API, gateway can use the service discovery to route the request to a proper service and if another let's say browsing service want to talk to shopping service. It will first go check the The service discovery and then the service discovery will tell browsing service where to find a shopping service so that they can communicate.
51:55 Hao C.: And this is where the business modules located. Config server is a centralized server, then you can config all the components here. Based on the research I did a tech GB told me config server, really config load balancer, but for a gateway service discovery and this business modules, it is common to use the config server to conflict. This components and the log monitoring is centralized logging for each service. They will they will first store the the lock locally. And then they will be a logging agent that will send the log to the logging monitoring which is a centralized component. So that you can you can you don't have to go to each individual service to see the log. You can just go to the log monitoring Another circuit breaker is is to prevent failure because scanning. What I mean is, let's say if one service is dependent on another, let's say, shopping depend on the order and order is down. So in that case that will the shopping service wall will be also be damaged because others is is not working. So if circuit breaker detect, this problem, it will stop the the request go to other service.
53:24 Hao C.: And then the system will figure out a solution for example, to go to a replica of other service. They'll let that replica to handle the task. Yeah, that's that's the mycro system and a monolithic. This is simple. This basically a big program running on the same machine and it has everything. There's no There's no remote communication between each components. It is already. It's our all happened in the in the same machine. So Yeah.

54:06 David L.: So, we're learning implementations for the components in the MICROSERIES.
54:13 Hao C.: Implementation.
54:15 David L.: Load. Balancer is just an idea, right?
54:18 Hao C.: Hmm. Our engine acts ngx is a implementation.
54:26 David L.: What are the options that I have? For each of those components, what are the options I have?
54:55 David L.: Those are all just ideas, right? The compact.
54:59 Hao C.: Yeah.
55:00 David L.: So we said that you need to figure out all those different implementations and what are the different options for the implementation of Each of those idea. Um, so what are the options that we have for the load balancer?
55:27 Angel H.: AWS ALP.
55:32 David L.: Okay, that's one of those anything else.
55:36 Ruoming Z.: Google Cloud load the balancer spring clouds, a lot of answer.
55:46 David L.: Okay, anything else? What about load and balancing strategy?
56:11 Junbo Z.: Randomly sign.
56:15 David L.: Okay, randomly sign is also a strategy.
56:21 David L.: You think we will use that? For load balancing. We can randomly assign? Sure. Are, we going to use it or
56:34 David L.: not? Why we're not going to use it?
56:42 Junbo Z.: No, too many possibilities.
56:43 Jiawei Z.: Because because the goal for load balancing is to avoid overwhelming of one of
56:46 Ruoming Z.: It's not.
56:50 Jiawei Z.: the microservice instances, but random assign does not avoid this. So the most
56:58 Jiawei Z.: commonly used one is wrong robbing thing it, um,
56:58 David L.: Yes.
57:05 Jiawei Z.: It's just distributed as a round. First to first to. For example, if we have three service instances and it first to serve to A and then to be an end to see if we have more, it comes back to A after that.

57:31 David L.: Anything else other than
57:33 Junbo Z.: Or maybe. We can use one server for like a more important. Request and the other servers for less important.
57:50 Junbo Z.: To increase performance.
57:50 David L.: Tower. How are you going to distinguish an important? So any important requests versus and non-important requests
58:02 Ruoming Z.: We can decide by the frequency. Of the request.
58:09 David L.: What do you mean by frequency?
58:17 Ruoming Z.: and if the service is very frankly, accessed during a specific period of time, compared with others, Which means it's frequent.
58:30 David L.: Not necessary. You're Browling. Service is always going to have higher frequency for requests, comparing to the rest of services. Following the other services less significant than your Browling service.
58:53 Junbo Z.: Can we ask ask customers to pay for that?
59:00 David L.: You ever paid for any of the Web application before?
59:03 Junbo Z.: No.
59:06 David L.: glad you say, you know,
59:06 mq s.: Maybe we can.
59:12 David L.: You're the one who I'm never going to hire to beautiful Web application charge my user, hell no. Remember the Web application is just a medium right? Hmm. The real monitorization is based on the Whichever the products we're going to sell. You don't want to pay well. Who is going to charge you, those stuffs. Actually, the visa is charging that For each and every of the pain. For whatever the goods you're going to pay. If you're using a credit card, then they are going to charge you. One one cent too, thin depends on the how big is to be. But, you know, for any of us doing the retail, um, purchasing then, the visa companies actually charging that A brilliant business idea that you're going to earn a lot of money, huh? Unfortunately, there's only one visa.
01:00:23 David L.: What else? What are the other? Load balancing strategy stamp we have. River Ron is going to work, but remember, But most of the cases going to work with a localization or like a local cluster. So and here the design you have bring up You put the whole system as one cluster, right? You have something like this, right? And then each of them is a cluster, Your API. Gateway is

communicating to each of the API gate who is going to communicate with one of those.
01:01:23 David L.: Get my lining, but That's what I mean. Makes sense. Let's just move your Service discovery to here. that's one of this, um, microservice architecture, you can have Most likely what you are going to have is something like this. So let's just Come here. So, rather than just have the whole thing as a cluster, we will have something like this. so,
01:02:46 David L.: Hmm. We would rather have something like this, okay. What's the difference between the first solution and the second solution that we have seen? Folks.
01:03:24 Angel H.: Um, is it like the the below one? I'm server there. Combined into like one one module. and then on the top one,
01:03:39 David L.: What's the reason why we want to have microservice architecture?
01:03:46 Hao C.: Oh, failure tolerance. Scalability.
01:03:53 David L.: Okay, let's talk about failure tolerance in your original design. You have something like this in below. How failure tolerance this is.
01:04:06 Hao C.: Basically no.
01:04:08 David L.: Right. So remember you Let's see, this is down. If this is down, what's going to happen?
01:04:22 Angel H.: The series is down.
01:04:26 David L.: Right.
01:04:26 Ruoming Z.: The whole class.
01:04:28 David L.: If the whole cluster is down, what's the failure tolerance? Then, what's the point for us to have the MICROSERVICE architecture? You can have multiple of those clusters deployed. At here, I only have three, right? You can have more than dollars. But remember anything have a price like we said before. It's been so many time to start so many instances eventually, it doesn't review any of the advantages to deploy a microservice architecture. And instant for each of the communications in between those different services. Well, those are the components, not your business services. They are going to. Forward the requests in between those different clusters. Increase the cost. But your favorite tolerance is the same.

01:05:20 David L.: How about scalabilities? What if same thing, the email and service in cluster, one has a performance issue that we need to scaling up.
01:05:31 David L.: And then later, same thing, happens at here, and here, and here, and here. How are we going to solve a problem like this? Folks. For this. For this design, your email service is overloading now. How are we going to solve it? Folks. Hello.
01:06:25 [illegible]: it's, it is a similar with With a concept in database about the Sharding. And we had many two methods of data, sharing a vertical and hurry.
01:06:41 David L.: This is not your database. This is your email service.
01:06:41 [illegible]: Up.
01:06:46 David L.: Look at your design. Look at this design. Have one of those services that's overloaded. How are we going to scaling up? See the point.
01:07:10 Hao C.: so,
01:07:11 [illegible]: You can add more server.
01:07:13 David L.: We can more.
01:07:15 Hao C.: Also. Are you saying because the email service belongs to a bigger cluster if we want to scale it up we have to scale the entire cluster. So that's not feasible. Is that what you trying to say?
01:07:28 David L.: still feasible, but it's going to cost a necessary because All of those. Are perfectly fine.
01:07:41 Hao C.: Mm-hmm.
01:07:45 David L.: When you're doing a design, you're not just feeling up all the things and bring up an architecture, call it a day. It has to make sense. and remember, if you have the cluster like this, technically it's you doable to just scale up
01:08:10 Hao C.: To.
01:08:11 David L.: This service, right?
01:08:14 Hao C.: Yeah.
01:08:16 David L.: But again, you have to put dedicate efforts to monitor each of the cluster. What's the current status? whether this is green or this is Red, which one is getting red, which one is getting green. Makes sense.
01:08:35 Hao C.: Makes sense.
01:08:35 David L.: Because now you have to monitor for the whole cluster to

know which one you're going to scaling up and which one you're going to scale in them. Right. because your log your traffic volume is saying Hey, this cluster Let's say. This cluster. This cluster is having some warning. It's it's getting yellow now, right? Some of those are getting red. How are you going to know which is getting red? And then you have to when you figure it out, then you have to figure out how to scale it up. Make sense.
01:09:18 Hao C.:
01:09:18 Angel H.: I'm doing like in the first time you have to find out which cluster it is in and then but
01:09:23 David L.: Not first time every time. How are you going to know that? It's always the email service that's going to be overloaded? It could be the other service that's going getting overloaded. It could be the shopping service. That's getting overloaded. You don't know.
01:09:40 Angel H.: Yes.
01:09:43 David L.: All you are seeing is that for this cluster, it's getting some traffic's warning. Makes sense. And each of the cluster can have different situations.
01:10:14 David L.: Right. If you design like this, then you have a you need to dedicate to a monitoring within the cluster. What is the status for each of the instances? And then what are you going to do for each of the instance? What about the above solution?
01:10:45 Hao C.: You can scale a single service with all worry about the entire cluster.
01:10:51 David L.: Right, if anyone is complaining. Anyone. Bring to send backward. Hold on. Send it back. Okay. If this is warning, I just scale up this cluster. The rest of those are all working fine. Leave it there. Make sense. Any questions on this part?
01:12:01 [illegible] : So the second solution is always more practical in real case, right? We'll
01:12:08 David L.: Most of remember, you are doing a microservice architecture. The only situation
01:12:08 [illegible] : always
01:12:13 David L.: that I can think of that you want to stick with your solution. One

is when the communication in between a services way, much higher than the request that sanding to your server. In cloud play forms, any of the requests standing in between services is going to spend money. it's highly unlikely, the situation like that happens, but if you have to ask me, When will the solution one going to be preferred? That will probably be the only time.
01:12:49 David L.: Say, when the Communications between the browsing and shopping and order and payment and email service is way much higher Which means well I don't know how you design your web application but your web applications communications. have way much higher volumes comparing to Whatever the client is saying to your backend server. In that circumstance. Yeah, probably will consider about Sir Solution one. Or hydrologist a redesign the whole Web application. The Hao is doing what has happening. That you have so many in between communications.
01:13:32 David L.: Right. It doesn't make any sense. Probably because you December the business blows with too many steps too many more years. Hmm. Talking about a little down thing, ribbon round again, is a practical one, but most of the case, it's in a local cluster. Something like this. if I have something like in Europe, I have one of those Well, one of those kind of the distribution. For all the business services, okay? And there is another one in Asia, another one in us, another one in South Africa, in cases like that. I want to do load balancing based on the GEO location. Remember the headers in your request. We can get the IP address.
01:14:23 David L.: The geolocation. Hence, we know we Load balancing the request to the closest cluster geologically speaking. Sometimes some web application will have a lot of the user with the VPN connections. Hence you don't you actually don't tell. Where does the user actually? Because all the connection is based on VPN Then you can also load balancing based on the shortest latency. All right, so different load balancing strategy based on the different scenarios. No, those load balancing strategies and know when to use, which All the low, the balancers we mentioned above is going to be a practical one engine exists. Also, a practical one.
01:15:15 David L.: Engine says Design is a reverse proxy which can also be used as the load balancer dedicated server that with the dedicated source code to

handle a high traffic volume. Mmm. All right. Keep going.
01:15:44 Hao C.: oh,
01:15:56 David L.: Also keep going.
01:15:59 Hao C.: Oh, I think I I explained it, what I done. So, I don't know what to
01:16:10 David L.: No, about all the rest components. What are the implementations?
01:16:15 Hao C.: All. Mmm.
01:16:29 David L.: Hello.
01:16:31 Hao C.: Yeah, I'm here. I just
01:16:33 David L.: Yes, you going.
01:16:35 Hao C.: Can someone else help me about the implementation?
01:16:44 Angel H.: I think for the API gateway this, there's a Spring Cloud gateway.
01:16:57 David L.: Anything else? Hello folks. Anything else?
01:17:18 mq s.: AWS safety.
01:17:19 Jiawei Z.: We also have the Zhu and AWS API, Gateway Ava, Skip Gateway, usually uses when we are going to deploy our microservice on cloud and Zhu is not very often used. I think because it's
01:17:39 David L.: He I won't say that. With. Luis very popular. It's commonly used.
01:17:44 Jiawei Z.: oh, Okay, because I remember it's a synchro synchronous model but yeah, a synchronous synchronous does not mean that it's not popular, right?
01:17:55 David L.: Well, yeah, doesn't mean synchronize asynchronize. It doesn't mean it's not necessary that asynchronizes always do good things. Think about it in this way. So eight organizes always a good thing to have, but Doing just one part asynchronized. They might not help you to handle all of those, which is saying if you want the whole thing to be asynchronized, starting from the lower, the balancer to the API, gateway, to all the services that you design.
01:18:27 David L.: If there's any of the thing that you wrote is going to be synchronized, then there will be a bottleneck for a performance, right?
01:18:36 Jiawei Z.: Right.
01:18:36 David L.: You're going to put a lot of the dedicated works to make things asynchronized, sometimes it's just not necessary. So basically you're saying Hey the API gateway. We wanted to be as big as possible, right? Why just give you an

analogy like that, the gate of your house, right? Technically your house your your gate is as wide as your house. Then you're going to get the Among income guests per time unit, right.
01:19:17 Jiawei Z.: Yes, yes.
01:19:20 David L.: Simply that doesn't make any sense for your gate to be as as wide as your house, right?
01:19:27 Jiawei Z.:
01:19:28 David L.: Same thing as here and remember. If after your gate in the house, everyone comes in. And then going through the same room and then entering to the rest areas, then the bottleneck doesn't well. Yes, your gate is a thing for nice. As much as you can for the bottleneck. Start is not your gate anymore, but the room that first room that everyone has to go through, And then entering to the rest of the parts in your house. Right.
01:20:01 Jiawei Z.: Right.
01:20:03 David L.: So be reasonable. Um, a synchronized is not always the bad thing. Mm-hmm. If you worry about your performance or any of the performance for your modules and you should always do the performance testing. You have an idea, you do the implementation and you put it into a pressure test, load test, performance test, they are all interchangeable. That test becomes a proof To provide evidence to the rest of your colleagues, whether the solution is going to stand, whether the solution is going to handle the real business flow.
01:20:54 David L.: AWS Gateway API, Gateway is going to be used with other things. The routing Route, 53. Config map for the service discovery. Right depends. Hmm. You can also do that with the GCP. Do that with the edge order. Whichever to play form that you prefer to use. Okay, keep going. Mmm. oh, Hello. Hello, folks here.
01:21:45 Jiawei Z.: Yes.
01:21:47 David L.: Yeah, keep going.
01:21:50 Jiawei Z.: So shall we move to the service discovery?
01:21:53 David L.: Sure, let's go.
01:21:55 Jiawei Z.: Oh, sure for service discovery. There are first Eureka and zookeeper. And also Kubernetes. um, Eureka is supported by spring and Zookeeper is oh, it's a it's, um, I'm not pretty sure about the difference between

Zookeeper and Eureka I just know that on some detailed implementation there might be some difference but I'm not. I'm not sure about the abstraction Kubernetes is Is it handles the auto scaling? If we want some specific service to be scaled up, Kubernetes continuity by itself automatically and I think that's the main difference between it with Eureka.
01:23:04 David L.: Okay, keep going on. Remember Eureka has 1.0 and 2.0. Eureka 1.0 is Is pretty is out of maintained, hence, we don't use that anymore. If we're going to use Eureka we're always going to use Eureka 2.0. Hmm.
01:23:23 Jiawei Z.: Yes.
01:23:25 David L.: Your country 0.0. Again, depends on the platforms that you're dealing with, if you're dealing with plate. Cloud play forms in, say AWS, then you're using Config map. Rather than other things. Yes, you pay some of those. I mean for this service discovery, you pay for config map, but You don't have to implement it, it's there and then out of box that you can use. And all of those survey. Well, most of the services on any of the cloud platform is going to be Failure tolerance, scalable. A job agile agility with the scalability, a jail ability, availability.
01:24:17 David L.: Hence it's not a bad idea if you're planning to pay to your age to the cloud
01:24:19 Hao C.: Are.
01:24:23 David L.: play form, so that you have out of box implementation. Actually. If you think that only big companies is going to pay think it again. say each of the engineer that you hire is where the for 100K 10 of them becomes routine, including the front end and backhand And then you have to plus the expense. For the project manager, the tech lead engineering manager architecture. Right roughly speaking. A professional team takes a million per year as the cost.
01:25:04 David L.: If you view the program application for three years, that's three million dollars. If you're dedicating to provide, if you are using the Cloud Play form, on the other hand, everything is out of box. Then you're just going to spend one year for a team to view the web application. That's three million versus one million. The rest 2 million is going to spend another 500k for all the beus in the next two years for business running. We bring the web application on alive. Then

it's still a worthy because it will save you. 1.5 million. Make sense.
01:25:41 David L.: So, so
01:25:42 Jiawei Z.: Yes.
01:25:44 David L.: If you think only big companies going to pay the BU for the Cloud play form, think it again, actually. A lot of the big companies are doing dedicated implementations because they are not short of cash. They are not short of engineers. For those startups, small to middle size of the companies. Sometimes, out of box solutions is good. All right, let's keep going.
01:26:16 Jiawei Z.: Sir. Oh you next we have is a log monitoring. Usually we, we can implement either way the centralized or we can have it for each. microservice, the text deck here is the Outstack we store, the locks the information we need through log stash and then oh, I think his captured, it's captured by the luxurious and then starting in the elects elasticsearch and kibana is the provides the UI for us to To get directly, what we were looking for.
01:27:02 David L.: Sure, that's a classic combo. Anything else?
01:27:06 Jiawei Z.: Yeah. For, for log monitoring. I think that's the only one I know.
01:27:21 Junbo Z.: I asked, How does the monitoring trace, the order of the logs and applications. They use some kind of trace ID to to make sure every log is right, right? In in order, I'm not sure if that's right.
01:27:37 David L.: Now, you can use the solus to do that.
01:27:40 Junbo Z.: Okay, okay.
01:27:44 David L.: Any other implementations for the log monitoring? Prometheus Plus Graphana is also another one. Using a splunk is another one.
01:27:59 ≣ ≣ ≣ : And we can use the spring boot, actually actuate.
01:28:04 David L.: Actuator is going to be correct collect with the Prometheater, the matrix from Spring, actually going to be collected by the Prometheus feeding to the Grafana. That's another common combos. You're going to see Okay, keep going.
01:28:37 Jiawei Z.: Oh, the Next, for the circuit breaker. The most commonly used one thing come resilient 4j and
01:28:50 David L.: Uh-huh.
01:28:52 Jiawei Z.: um, Not sure about the others.
01:29:12 David L.: his tricks, there will be also another classic one really exposure,

a very
01:29:15 Jiawei Z.: Yeah.
01:29:16 David L.: common to be used on
01:29:23 Angel H.: I'm sorry. David what was the the other one you said?
01:29:26 David L.: Histrix.
01:29:27 Angel H.: Okay.
01:29:30 David L.: Okay, keep going.
01:29:43 Ruoming Z.: Hystrix
01:29:52 Jiawei Z.: UM, than for the configuration server also can work either in a centralized way, or we can have a separate configuration server for some specific microservice, UM, We have Spring Cloud. Config server. The IT stores the config server in Kit and we can pull it from the GitHub.
01:30:23 Jiawei Z.: Almost. So we have AWS parameter store. It can be used when we are deploying our microservice on cloud. Yep.
01:30:43 David L.: It. Okay. So on for current popularity for microservicing implementation zero in general, two styles, One is called Intrusion style, The other is called noun intrusion style. The intrusion style and non inclusion styles talking about, whether you are going to change your spring boot source, code to implement the microservice architecture. When you are using a spring cloud, you have to enable bunch of things including the Configuration Server Service Discovery, API Gateway, those argo, those are the changes that's going to intrude it into your source code for your original spring, boot application. Makes sense.
01:31:31 Jiawei Z.: Yes.
01:31:34 David L.: Originally, that's the solution that we have. We used the spring cloud to implement or we build up our own API. We use different open source API gateway, like Corn, API, Gateway to get a eureka 2. We get a Knockos we get Azul, we got a console for service discovery. We build up our own log monitoring with the circuit breakers resilience 4g. We put that together become. Long spring.
01:32:09 David L.: Microservice architecture. Then it becomes more and more universal and we use the Spring clouds to implement our microservice architecture. Then. There's another thing that kicks in which is the Kubernetes. The Kubernetes is saying, Hey, don't worry about any of the source code change, you just Define your DOCKER file package, your spring boot project into a docker

image. Define your ci/cd pipeline, how you're going to containerization your web applications and then all the things is going to be deployed into the Kubernetes platform. The Kubernetes contains to.
01:32:53 David L.: Contains the lower balancer, the Service Discovery circuit breaker configuration server and all the services that you have is going to be wrapped up as a, the minimum running unit in Kubernetes as part. Each of the service running in one part, you can have hundreds of the part running the rest of the implementations are all done with the Kubernetes. That's a good thing that you can have for your implementation. No intrusion becomes more and more popular. That's the reason.
01:33:34 David L.: Why the Devops is getting more and more popular? You're going to see software developers doing the Devops At the very beginning. We don't even split into front and backhand and then with the business requirements, getting more and more complicated with split up into Front-end backhand. And then to QA Devops different rows, doing different functionalities. And then it becomes more and more popular to have the known intrusion style for the microservice. Implementation, has the Kubernetes is going to be an independent rather a relatively steeper learning curve. Implementation for the MICROSERVICE architecture. Makes sense. Okay. Remember to return back and then fix up all those missing information. And we are going to, I'll just send this.
01:34:47 David L.: we are going to do a little bit more practice for Spring boot. Won't take you much time and then fuel up. All the missing information for the MICROSERVICE architecture. Then you are going to implement the authentication and authorization for your Microservice architecture. I think I missed us something. So remember, authentication is checking who you are. Authorization is checking what you are able to do with your current access level in your web application. Yeah.
01:35:36 David L.: plus Springs, security, plus sso implementation Single. Sign. Okay. So you are all going to talk about and then research on the authentication and authorization. With your microservice architecture. How are you going to integrate it with the API gateway to implement your authentication and also relation with OS to spring securely and also remember for any of the Web application nowadays? You're going to, you're not going to be logging over and

over and over.
01:36:25 David L.: For every request you're seeing, it's always you logging for one of those. One of those logging page. And then within a certain amount of time, you're not going to keep logging for sanding any different requests. That's a feature called Single Sign Up. And in your longing page, you're not just only using the red credential registered on this page, but you're also going to be able to login with your Gmail with your github account. With your answer account. Those is going to be implemented with the OAuth 2.
01:37:07 David L.: Okay, so that's the research you're going to do and that's the implement design that you're going to bring back with your microservice architecture next time. We are going to discuss further for the security things that we need to take here for our web application next time. When we have that microservice architecture established field up with missing information, we are going to talking further about securities. Make sense.
01:37:43 David L.: I'll leave some of those references to you. Those aren't references Rogers and it's over. All right, folks, I think that will be it for tonight. If you do have any other
01:37:59 David L.: questions for you free to Talk about it. If not, then feel free to leave.
01:38:10 [illegible] [illegible] [illegible]: Do we need to? Implement other. Microservice architecture in our local environment, like the lower balancer. The login and monitoring. We?
01:38:28 David L.: no necessary implementing that we're going to take too much time for you to Do it and it's unpractical, you. You are going to frequently asked about microservice implementation, but you're not going to be required to intimate want during the interviews. Hence, there were some cheating. Cheating areas there that you just going to understand not necessary to do it.
01:38:57 [illegible] [illegible] [illegible]: Okay.
01:39:03 David L.: And remember later you're going to build up a resume for. Seven to ten year of experience, which means you're going to have multiple projects on your resume. Different projects, different architecture. So you should prepare all those different options, and then combine those different options to build off different architectures.
01:39:35 [illegible] [illegible] [illegible]: Your resume? Oh, it's, we need to Cover a more topic as we can

write the I mean.
01:39:45 David L.: Yeah, we're going to talk about how to build up the resume later, but just spoiler alert. You're going to have multiple projects. Now, you are preparing all those different options. Those different options are modularization for your design. Hence later, you're just going to swap different implementations and then build up different architectures.
01:40:09 ≣ ≣ ≣ : Okay.
01:40:11 David L.: The only thing that's not going to be hybrid is whether it's going to be annoying intrusion or true intrusion. If it's now intrusion then it's in Kubernetes. The thing that you need to implement by yourself is the API gateways. I don't think Kubernetes has an generic API gateways for the rest of the part. You can find all the implementations that Kubernetes has already done for you. But if you're going to, you're going to talk about, you're going to talk about the Kubernetes during the interviews things. It's a popular thing for nowadays.
01:40:48 David L.: So make sure that you do understand what is Kubernetes, and how do you implement those components. Good news is that it's not going to be complicated. Well, Kubernetes is going to be very complicated. Not necessary, very complicated, but the learning curve is relatively steeper comparing to other frameworks or tools. But the good thing is that you need to figure. You don't need to master the whole Kubernetes play form. What you need to do is to understand what is play, Kubernetes play forms. What are the different components that you have? And how do you use Kubernetes to implement a microservice architecture? On my role, not in your ID or local environment? Makes sense.
01:41:38 ≣ ≣ ≣ : Yes.
01:41:42 David L.: Okay, um, again, it's over. If you do have any other questions, feel free to Shoot it up if not then feel free to leave.
01:41:58 ≣ ≣ ≣ : Thank.
01:42:00 Jiawei Z.: Um David, I have a quick question here for the lockdown. Sir, when we're designing in the architecture of Microservice.
01:42:09 David L.: Yes. Also, another design that you can have the API gateway is going to be
01:42:10 Jiawei Z.: Um, can we put it like after the API gateway?

01:42:19 David L.: responsible for the authentication and authorization only those valid users are going to send requests to the load balancer.
01:42:28 Jiawei Z.: I see. Thank you.
01:42:30 David L.: No problem. Anytime
01:42:33 Angel H.: Thank you.

View original transcript at Tactiq.