00:00:04 welcome class so today's lecture with Mr Deepa Chikara I kpur alumini and he will be talking about cicd in the context of SAS cloud computing so Deepa C to you thanks hey folks good evening everyone so let me stop sharing my screen and then can di right in yeah so just before I start like how many of you have uh like uh experience in production systems uh I hear some of you are uh like are in your master so you would have some level of uh production systems or have developed a like app just for

00:01:04 fun and like you have like deployed it in a fashion where there are [Music] clients or like we have no one who has okay have one okay so most of the people don't have the so uh just to yeah just to touch upon the class background so we have around 148 candidates in this class out of 14800 are around AI executive so they are already working in Industry corporate uh I'm not sure how many of them are working in software industry as such but they are working professionals and around 50 students are

00:01:59 there who are regular Masters or btech final year students sure okay so anyways like we will touch upon uh like what are production systems and what are the issues one faces uh while interacting with the production so when you go to the industry in terms of like your employment or let's say in term of your internship so you would be better prepared or you would have some ideas to solve the these some of these particular issues right so yeah uh so I'll mostly be talking about like the issues uh regarding the production nightmares and

00:02:42 how cicd is the major tool to solve most of these particular issues not all but most of these particular issues are solved with the can be solved with the help of the cic and why is cicd in general is important so and Deepak uh so I have like close to S years of Industry experience now uh I'm a Staff software engineer in harness so just to tell you about about harness so harness is like couple four years old company which is in the space of cicd so we are basically trying to solve cicd uh problems uh uh automating most

00:03:28 of it uh like Ci some part of CD and uh CV change continuous verification and all those stuff so integrating with health sources and all are the critical part of it right so so let's start so let's uh say that um like alen musk has just twed that uh like Dogecoin is the new uh cryptocurrency which is cool uh which should be used right now as a product manager so like any so to tell you about like product manager is a person who comes up with the requirement so he owns the product he's not a developer but he

00:04:23 owns the product so he would basically see what exactly uh so he's a basically bridging cap between the customer and the engineering team so he would see do a research keep what the customer wants and he would come to you saying K boss we want this dodge uh like smile or emoji on our app ASAP like Alan musk has just tweeted about it and if our app doesn't have it we look bad or uh if we uh do have this particular functionality the traffic on our app would increase significantly right so the critical

00:05:07 point over here is the delivery in like ASAP like the this feature should be there in the app yesterday the next B best time is now so so what is happening in the industry is that we are always uh the product wants a tighter loops around the feedback with the customer so they want to know what customer wants and want to deliver uh change the feature or deliver according to the requirements of the industry or the like market right so this requirement comes to you as a developer uh ke you need to add

00:05:51 this particular Emoji into your app right um so what would you do like you would watch some YouTube then probably like you would start working check out a new Branch right so you uh take a new Branch write some code right uh you would do some stack Overflow how to add Emoji blah blah and then merge the branch uh like you have just added the Emoji and deploy into the production and then go back to watching the YouTube right so this is generally uh would be the case if no one is using the your like you are

00:06:33 the only developer and no one is basically using the you basically your product so this is a college product project or like some like you are doing it out of fun what happens generally in production production uh systems is that you would basically check out a branch now you don't know uh like which branch Master release QA but hopefully like if you are a good company you would have a strategy around it uh for H fixes which branch you should check out from uh if it is a like a quarterly or Sprint

00:07:15 wise uh release which particular Branch you should so all those things would be in place so the point being here is that there are multiple people with who are working on this particular project and there there are multiple releases which are going into the production system right uh The Next Step you are to do uh like write write your particular feature uh add your feature into the uh into the the branch which you have checked out and then fear if your code is going to break an existing functionality so

00:07:52 this is the uh the biggest uh issue uh so which uh kind of uh which was the major driving force into moving us towards the microservice uh architecture if you have like read about it so in a m systems uh like there are like thousands of people are working on a particular code base and no one has the the view of the complete functionality right so whenever you put add a new piece of code you would always hear that it is going to break an existing functionality until it is tested thoroughly right even in microservice

00:08:38 architecture this is uh still a major concern like not as big like over there though it's just uh so is such a big issue that we have to rely on waterfall model where we would basically say PA we would add bunch of code base and then we would give it to the Q way for the testing because uh adding the code uh like on a running production systems is just not uh feasible but in uh microservice architecture the code base is a bit lower uh the uh we can uh like manage uh in terms of like knowing the

00:09:20 functionality but still there is a high probability that you are going to introduce regressions right so so you would always you will never have the confidence that uh whether the code base which you have added is going to break handling or not and on top of it if you are intern or employee taken look at the micros like then though it's just insane the how would you even know the whether the adding a particular piece of code is going to break an existing functionality whether it is going to introduce a

00:10:05 regression right so yeah but you can't do anything about it right like you just fear that's just a fear uh then what are you going to do like you are going to initiate a merge into the master Branch uh probably there are conflicts because you are just not the the only person who is like looking at the uh working on this particular uh Branch it's a master and there are other folks who are like continuously merging into the master you would probably resolve these conflicts and then uh you would go on to merge

00:10:40 that particular your code base into the Master Pro then uh you can release to the production servers no because uh your deployment strategy might be that the app has to go first right because if you're uh if you release the uh uh the Ser if you release your code base first there might be an issue with the Rel like the the app uh might not work right so this uh generally happens when you change to make a major change into your existing code base so what it happens like what it does is like it would

00:11:23 basically might not be compatible it might be it will work only with a newer what happens is this kind of issues happen in micros service arit so let's say you are working uh on a down service right so in this particular case what would happen is this particular micros service is let's say creating an order and there is a service which is basic the creation of the order so now there would be a deployment strategy so the you cannot deploy creation of the order micros service first because if it does the code base which initiates which

00:12:18 creates the order is not even present right so you would have to first deploy the microservice one which creates the order and then you would go on to release the particular uh micros service which initiates a order creation right so probably like you would wait for the app to be released then you deploy to the production servers this deploy to the production servers is like very simple to say and much much complex in the real real life scenario right then you will go on to check if your changes are

00:12:56 working uh fine or not and then new your license choices as you come to know that your code base is not backward compar right so uh once so whatever level of code like how many of you feel ke even if you do every type of coding right then like any type of any level of testing as well there might that your codb still can break into the production servers right so there is a high chance CH that it might break into the production right so there is a high chance apparently right so like backward compatibility is one of the major issues

00:13:41 because of which your you might not like your code base basically is not working right like or break the existing uh so let's say the apps which are released works perfectly fine but the earlier um apps does not even recognize this particular U uh uh let's say emoji and what they do is they throw a nullo exception or they uh so there has to be some inbuilt mechanism that if uh the app is basically uh of older version then we would not send the Emoji itself because they will not be able to display this particular type of

00:14:23 emo so you would have to uh like there are multiple things which can go wrong so you would have to quickly roll back the changes which you have done and uh again this particular n step nine is again manual that you would have to manually uh work uh check the logs if there is something which is breaking right if there are any errors which you are seeing into the logs right and uh then you would manually have to trigger the roll back of the code base itself so again this is manual step which you have

00:15:03 introduced so there are too many manual steps which are there and still we are uh not 100% sure that what we are delivering gives us the rest of these so now now let's talk about the uh challenges of going to production so like most of them we have talked about but uh like some of them uh we would see uh like in depth so first of all like the regressions so as I talked about the regression is basically the breaking of the existing functionality so you delivered a functionality which is working perfectly fine customers are

00:15:46 using it and the moment you deliver the new set of features or you fix a bug you basically break the existing functionality so you introduce regressions right so now this these are the type of issues which are the like um leads to the unsatisfaction of the customers because they feel that the products are not mature enough like the so it would be frustrating for you that if a feature which was working yesterday in your WhatsApp is not working for right so that would be the big so if there is a new feature like

00:16:29 you are not accustomed to using it or you don't even know about it if they it has some issues it's still like manageable right but a feature which is working perfectly fine if you break that that is a major cause of concern in terms of your uh user engagement and it questions the uh the tech which you are delivering to the customer right uh more than and more than regressions it's the anxiety uh of introducing regressions right you are never 100% sure uh especially if uh you are giving a particular feature to be

00:17:12 developed to a new higher or lesson intern so those interns doesn't have haven't seen the complete code base like how would they even be confident enough right so and as a let's say team lead or as a engineering manager it would be really tough for you to have full confidence that whatever this uh new person has like uh developed and uh uh deployed into the M like production is not going to break anything so like those anxieties are real right and then uh as I talked about like uh deploy to the production servers so

00:18:02 when we say deploy to the production servers like this is much much more much complex right so it's not something like uh you would just run ambient clean install and uh you would basically your server would come in and now the customers can you know uh uh like start working your particular new feature uh you have to take care of many things right so like creating B like it's as simple as like creating binaries is quite complex task because you might have to do it multiple times in a day running this ment clean install is a

00:18:44 complex thing plus you have to take care of other things as well for example this has to be a separate environment because uh like doing it in a environment where uh like which is let's say have other services running in or um might impact the quality or might introduce some of the issues with your binaries probably right like for example uh you say uh this uh servers have to be like the jars have to be you know compiled with java8 but the server which you use this has Java 11 right so if you're are

00:19:26 making a fat jar then some of these uh classes would be introduced let's say into your fed jar itself right or as a matter of fact like python or something like so it might be I don't think there would be boundaries in Python but just for the sake of argument so you would want to have a separate environment plus you would want that uh the binaries which you are downloading like the sources the D which you are creating is basically have those depend dep sees cast locally right uh because if you're

00:20:04 going to download them every single time so the uh the build tool which you are using have to be smart enough and the systems which are creating these particular boundaries have to be smart enough to you know uh use this particular uh creating these particular binaries in isolation uh now like once you create this particular binaries then you move these particular binaries to the host machine so you would do something like SS and move those particular binaries behave and this is J and here are like

00:20:38 more to this particular B to that particular machine there might be uh some deis scripts you might have to run so uh like let's say dump the data or like change alter table so there might be some DB scripts you might have to run before you restart those particular servers right uh like for example creation of the indexes right so once you uh uh like deploy the code base uh so many of the times uh the code base itself takes care of creating the indexes or let's say altering the table but many of times it

00:21:19 is not possible with the like when you uh doing it with the help of the codebase so you might have some scripts which might need to be run uh in isolution or before the you can uh like uh restart the servers so you would go on once the scripts have been run you would stop theun servers and then start the new server now again much simpler than like in than in actual life stopping running server and uh uh like starting a new server you have to always make sure that the availability doesn't get impacted

00:21:59 right so if you bring server a single machine right so you would the traffic which is coming onto that particular server is going is not going to get solved so you would need to have at least two machines so that when we are deploying on machine one all the traffic goes to the machine two and when we are deploying machine two all the traffic goes to the machine one and in between when you are like done deploying on machine one the load balancers have the time to redirect the server the uh request to the machine one

00:22:46 and also it gives uh it also have to give the time that the all the all the uh like uh transactions which are being run on machine to are basically taken care of so there are multiple things you have to take care uh before like stopping and restarting the server itself right and that's then then just imagine that you have to do it like hundreds of like because if you look at the scale of WhatsApp or Google like the number of servers runs in like hundreds and thousands like literally there are applications in Google which have more

00:23:29 than 100,000 servers like Docker images which are basically being maintained by the kubernetes cluster uh now the deployment strategy like I think I talked about this ke uh like for example if you are like if you are introducing an order creation flow then your deployment has to go into a part particular order right so like this thing uh I have done in my previous company so we had uh done a major revamp or re architecture of the uh of the existing Cod base and then literally five teams came online uh we did dry

00:24:19 runs and on the day of the uh deployment every team like we had literally 12 microservices which we had to deploy as part of the tree architecture and every single team would go in and say I have deployed this particular micr service now we can uh go on uh like deploying the next site of microservices so something like a topological like sorting right like so you would have certain there would be levels of microservice which need to be manage so what it does is that uh for example if the person uh uh it has too much

00:25:00 dependencies on other teams right so you can like if I want to log out at 6: p.m. in the evening I cannot do that because I like the whole team would be sitting and deploying right so there is a neat uh like hack or like feature which uh is introduced which is a feature flag what it basically does is all the uh at the time of the development itself every team would say if we would introduce let's say feature flag let's say ABC and we would use this particular flag ABC to see whether we have to start

00:25:44 the new flow or not so let's say the order creation flow uh in the order creation flow if I check whether the ABC is true or false so if it is false I never going to start the New Order creation flow right I don't care whether the uh the code base for the M the order creation microservice has been deployed or not uh I all I'm going to do is I'm going to deploy my codebase but I know since the feature flag is false my new codebase is never going to get executed and once the other micro service also deploys their code basee

00:26:25 someone can go in and just change a flag to Let's ABC is equal to true and now everything works all the microservices are deployed and you change the feature flag and now the new code base will be uh you know executed without any issue so there is no deployment strategy as such required in this particular case it takes the burden away from the developer and gives the developer a lot of freedom into his or her actions right and the next part is I think this we also talked about he uh so what if you introduce uh

00:27:04 like let's say you have some issues which basically uh uh into the production environment so introduced some issues or like there are issues uh with like which were there as part of your deployment then you would have to roll back whatever you have done so again you need to like uh deployment uh you have to start the deployment so what you should have is uh like your system or let's say cicd it should have the binaries readily available so because if you are going to you know uh like create the boundaries

00:27:46 again then it like it doesn't like it's going to lose a lot of amount of time so if you already have those binaries you would basically uh move them to the host machines you might have to run the DB script uh to the to roll back the changes which were there again this is manual effort you would have to do it anyways and then stopping servers and starting so this all this part has to be automatic in case the system sees that there is probably some issue uh which is being introduced as part of your so um let's uh take a little break

00:28:28 break and like uh see key if you are following along and if you have any questions until this particular point in time it would be much harder man like if you don't interact uh like at least tell me like if you are able to follow along or if you have particular issues like we can probably take them right now so Deepak I have a question uh as you told deployment is one of the major challenge actually but now these days there are lots of tool you can automate your process because you know the changes so is there any type of tool

00:29:37 that can maintain this type of activity so there are many uh so for example uh like the I think most of you would have heard about the genkins right so it's a cicd tool which basically uh where you can uh so all these five steps can be be configured as part of your uh like your deployment strategy like deployment script itself right so you can basically write the script uh so uh what you would do is we uh like as part of your script you would say create binary so it would you would basically say mvn clean install um or even before

00:30:20 that you would say GitHub uh like get uh like pull this particular Branch or get check out M origin and then you would run MN clean install so you would have those particular binaries so uh there are multiple tools uh uh harness has one of those uh right but there are other companies as well Jenkins is one of them gitlab is another one of those so uh which basically provides you these particular tools where you can automate this particular complete process of uh creating binaries uh moving to the host machine stopping

00:31:00 the servers uh uh starting the new server and while making sure the availability doesn't get impacted so uh just to touch upon this uh how do you make like how do you check whether the availability like so how do you make sure that availability is not anyone so yeah yeah go Ahad so as per my understanding genkins is just a Seder actually whatever the script we configure that will run at a particular time but ultimately if your binary get fa or maybe some some errors comes so will that tool manage or send some type

00:31:48 of alert to the developer or maybe uh someone so he can take care of those thing so you are talking about continuous integration or continuous deployment delivery continuous deployment so like so G case is a very perfectly fine tool right for continuous delivery so uh once your code base is ready to be delivered you just basically uh deploy that particular uh piece of code into the like your production servers so that's the script we are talking about so but when you talk about the continuous Integrations you need

00:32:30 much more like other tools for example GitHub uh uh like uh the integration with let's say Circle CI might be there so which we will talk about like in the next set of slides uh but yeah like genkins is just not a scheduler uh it's the it's the script which can get triggered uh whenever you want basically so for example when the when you merge a particular piece of code base into the master so it will trigger the deployment so it would automatically so what gup would do is it would it gives basically kind of web

00:33:09 hook uh uh web hook URLs where the the moment you basically uh uh merge the code base it would tell to uh let's say your genkins that hey man this particular branches must do your thing and uh Jenkins would basically make a check like take a check out of the current Branch create the builds deployed into the like your Production Service as part of your SC uh for future flag thing uh what tool did you prer so um so like different so this comes down to like your uh personal uh preference like what uh so harness has one of the

00:34:03 features like one of the the tools as feature flag where you can basically uh use uh that particular feature but personally we have like in my previous organization we had a separate uh service all together which we used to call as AB service which we use to like we would use to you know run our AB experiments and we would use them as your feature flags as well like so what it would do is basically at the time of the starting of the servers it would go and check into a separate service Alto together he well like what is the value

00:34:41 of this particular flag right so there is not much uh it's just like there is not much complexity into feature flag the complexity comes into you know using this particular feature flag as part of your development strategy so you have to use it correctly right so many of the times it's not possible to do it entirely with the help of the feuture FL but yeah you should use it as much as possible to take the burden of the de like delivery cyle bambo is bamb is so a has one so I'm not aware of but yeah

00:35:26 actually so I wrote in the context of the Jenkins so bamboo is very similar to Jenkins for the cicd pipelines okay uh let's move to the next part uh yeah so as we saw like there are multiple issues with our uh production like uh moving our Cod base to the production systems and as the scale increases uh these complexities uh also increases when so so like there is hope right so uh so we have like multiple tools to help us uh to prepare us for our production Journey right so uh one of the first tools would be cicd tools like one of

00:36:17 them we talked about like genkins there are git lab and there is haras ETC so CI is different and CD is different we would talk about it in like depth in later point in time but continuous integration basically means the I like the code which you have written um should be integrated with the master Branch uh effortlessly uh with a minimum level of stress and uh making sure that you are not introducing any regressions right and CD is basically uh like trying to move that particular piece of code base

00:36:57 which you return into the uh production with as much automation as much possible right uh then we have like Health sources uh so uh we talked about one of these issues to check if your changes are working perfectly fine so uh doing this is quite tedious and you most of the time you don't even know uh how do you want check right so all you can do is go to different servers and see uh whether that particular if you tail those particular logs and see whether uh you are basically seeing any null pointer exception or any type of

00:37:39 exception right but we have much more sophisticated tools which you can integrate your application or your data sources with which would do that particular uh monitoring for you so for example New Relic it would keep a track of the latencies of on your Keem apis uh the logs uh and other stuff to to let you know what is the health of your particular micros service and if there are and it also like uh keep track of the dag uh where like if your microservices is uh let's say facing some issues it would you can track a

00:38:22 particular transaction to the uh another microservice to know whether the issue which is coming in is because of your micros service or other mic or any other micro Downstream micros service right so these Health sources are basically a perfect tool to help you uh uh post your uh cicd Journey so once you are deployed into the uh the production then you would want to monitor your code and see ke whether you have not introduced a regression and uh in some cases you have not uh introduced some kind of latencies on your uh key

00:39:06 apas so let's say you made some code changes and now the login flow is basically slow right so you deleted bunch of indexes which were not required or you created bunch of new indexes because of which the insertion is slow now like may happen very well right like you would have uh uh like read this in your database engineering uh course that uh indexing although uh is a very powerful tool in your searching it uh like it slows down your insertions right because uh now you have to create much many more P trees in respect to the

00:39:48 number of indexes which you have increased right so if you have so that so let's say if you have increased the number of indexes uh it's not going to introduce any regression it's not going to uh impact the code flow which you have written for example you would want to increase the latency of your searching but it has increased the latency of your rights which is totally independent of uh the code which you have pushed right so Health monitoring tools are uh help you into you know uh taking care like um managing or

00:40:32 monitoring uh like the complete infrastructure and your microservices and gives a like bir ey view of your complete setup right then you have Cloud computation platform like AWS Azure and G gcp you I think you would have like everyone knows about them uh but just to touch upon they just doesn't give you the computation uh they give you the tools to manage those particular computation as so if you have a E2 machine you can like uh literally set up alerts on top of it which would tell you whe if that

00:41:17 particular E2 machine is under duress or is particular uh if you're running out of let's say uh main memory or secondary storage onto that particular machine also they give tools something which would help in let's say Auto scaling right so it's uh uh so uh the moment you are let's say uh so when you are let's say sleeping in night uh and like all of a sudden uh there is a Spike let's say into let's there's a match going on between India and Pakistan and uh out of all of a sudden there is a spike into

00:41:58 uh you into the traffic then they would give you the tools to Auto scale like you they would give you tools where you can horizontally scale they will deploy many more uh they would create replicas of particular uh boxes and uh you can basically manage the traffic much more in efficient manner and then we would like look at some of the best practices which not best like the like the practices which industry follows which helps you prepare for the dday and like try that the dday never so yeah like so let's start with

00:42:40 the continuous integration so uh since the courses is on cicd so I'm guessing that you already know the part of it uh what continuous integration is but anyways I'll touch upon like what is continuous integration basically it it's the process where you would want the code base which you have written gets merged into the master Branch as soon as possible right so it's opposite to what happens into waterfall mode so if you have read about like software delivery strategies so there is a software development model waterfall model which

00:43:23 basically says that uh we would do deployments in in stages so uh there would be one person who would take all the requirements so there would be a stuff called requirement Gathering right so all the requirements from the customers will be taken so it might take two to three months then we would do a p or feasibility study something like that that might take another two weeks to four weeks then we would go into deployment development uh phase where we would be basically writing bunch of code individually right

00:44:01 uh that might take one to two months and then we would be going into the integration phase we would be integrating all the stuff which everyone has written and then integrate it might take another one to two months generally happens much more takes much more time uh so like this complete process itself takes right 6 to 9 months itself so it has become obsolete because the software delivery does not happen in this particular fashion anymore so what we want is that the moment someone like our product uh manager comes in and says hey

00:44:38 man I want this particular Emoji as part of our chatboard as part of our app right uh then we would want to deliver it as soon as possible and the first app to delivery is like integrating your code base into the master right so key features of have to write the T So F you new feature right so uh you would have to add automation test case for that particular test that feature in isolation so when you would uh go actually into the industry you would come to know about these T is called junit what it basically do is that it

00:45:39 helps you to test the code base uh like in isolation in isolation not in in when I say in isolation it's not in terms of the the platform right and it's in isolation of the other uh task or other code of your micros Services right so when you say right you when you're let's say writing a micros service right so let's say API you are writing which basically takes an input a 2 + two and it like uh give you the addition of this particular uh these two parameters so you will write a test case code just for that

00:46:20 method which takes two parameters it does not you do not write a test case for the uh the complete stuff where you would have to form a c command and fire it to the let's say your controller and this controller would basically trigger so what it helps you in is ke when you write integration test cases so uh so you know permutation combinations right so what happens is he uh let's say uh there are uh let's say bu of uh there are let's say two functions function one and function two right and let's say

00:47:04 function three so there are like three uh Function One function two and function three and every one of these have IFL statements right so there is a function one which has IFL statement uh function two which has IFL statement and function three which has IFL statement so the total number of test three uh uh like Parts you would have to write like literally eight test cases where the if statement of first uh function uh if statement of second uh function and if statement of third function gets exeed and the second one

00:47:44 would be if if else and so like so the it would increase as the number of these uh methods gets increased but if you are writing the test cases just for your let's say uh like in isolation you don't care what is the output from the underneath systems you just assume that they work and you basically test your function which you have written in isolation so you just write two test cases one is for if and one is for else and you mock the output which is coming from the uh the underneath system which is let's say

00:48:31 function two right and for function two uh the underneath system would be function three so this helps into like cutting down the total number of branches and help uh it also helps you like you don't have to understand the functionality of the function you just assume that it always works so the function one which you have introduced uh would have two function two automated test cases and if uh someone comes in uh and uh like add a new code base which basically uh uh like breaks uh your uh like introduce a regression into your

00:49:12 code base uh one of these test cases uh should break right like that is the that is the point of writing these test cases so the person who is in like writing the new adding the new feature would come to know in very early stages of his development cycle that hey man I am breaking an existing functionality so I need to and this is a particular functionality which is you know getting uh uh broken with my new feat like with my feature so he might like reach out to you or someone to see like how can we you know make

00:49:53 changes or introduce changes without impacting of the coordinates so the next part of it is like monitoring the main repository and then the test automatically for every new commits push so this is very critical right so uh are merging your code base or when you whenever you are committing let's say uh into let's say your master Branch right at that moment uh these test cases should get run automatically and let you know that they that you have introduced a regression and even before uh like you

00:50:36 before you merge those into the master like so that should be one of the step like you are not introducing any regressions right so uh until and unless all of your test cases are basically uh uh are successful you will not be able to merge into the master brch right so you have to pass all the test cases but to do that these test cases have to run automatically and uh tell you the the status as well whether those what was the result from these particular test cases right so for that someone has to monitor whenever get code

00:51:19 get pushed into let specific BR so you so there has to be some monitor which keep on looking at onto the status of these all these branches onto a particular repository so like GitHub provides the this particular feature with the help of the web so you can basically configure whenever there's a new commit or which is happening onto let's say a particular PR or anyways otherwise it would basically initiate a it would basically hit a particular uh you or let's say call command where you can probably write your own set of code

00:52:01 bases let's say genkins or something else where you would basically say you would run these particular test cases some of the Integrations are uh like not are even simpler for example with the help of circle CI it would just introduce you would put that code base into your those configuration into your code base itself which would uh run the Circle C and it will keep on checking what is the uh what is the like whether the whether uh whatever it has run it has returned success or not if it has not uh like initiated like if

00:52:45 it has not given success it will not enable the merge uh for you so it basically takes care that no regressions are basically posted into your merge uh even without like no one ever taking a look uh whether someone is pushing the code base you will not be able to merge a particular branch which has regulations because you don't because these T cases have not passed and you don't have the authority to merge without them being you sucessful uh so these are bunch of uh like benefits of having continuous integation

00:53:31 so regressions are captured early by the automated test cases so the moment you would uh like uh commit to a particular Branch uh it would basically run the test cases and you would come to know that there are regressions and you can probably take care of them uh like in a much early development cycle rather than like completing the dev cycle and and then uh saying yeah I'm done with it I'm moving to the next task and then you coming to know that hey man there are issues with your development and this

00:54:07 needs to be take care first uh QA team spends less time so I think this is much more you know like everyone knows about this like so the manual testing has to be done like quite literally L you don't have to so if you are adding a like new feature as simple as let's say adding a Dodge into your emoji you don't have to test the login functionality because like the automated test done for uh stringent conditions on delivered code so like so this is one of the critical points uh which we have not talked about until this point which is

00:54:53 that when you add U to your code base uh so there are things called automated test case which will basically take care of the not introduce introducing reductions right but there are other issues as well which can be like generic right which when you write a piece of code for example you are introducing uh uh something called let's say resource leak right so when you are you are opening a connection with your DB but you are not uh like closing that particular connection right so this is a like uh so

00:55:32 if you deliver this in production systems at one point in time you will run out of the connections itself and uh all of your connections will be stopped right so uh the new ones are waiting for the new connections and the other ones are not released right so so what there are uh code analysis tools which can run through your code base and let you know that he man uh this is not the right way to write code right uh they have like uh become much more complicated for example in I think in Facebook what they have

00:56:10 done is uh like even while writing the code itself uh they would see he you know uh if you're writing a particular piece of code they will uh tell you ke when uh with the help of the artificial intelligence key the the other developers have written this particular code base in this particular F so this is the industry standard so please if possible follow this rather than go on writing your own Cod because this is you know like tested and is in production right so these code analysis tool like CU would basically take a look so you

00:56:53 can have like much more complicated rules uh with the help of this codee analysis for example when you run the test cases it would also check what is the coverage of your Cod so for example if your code coverage is 80% it you can just write a rule key if the coverage goes down below 80% uh you so you will not be able to merge to the let's say your master Branch itself right so you can uh like add kind of rules uh there are 4ate and all kind of security softwares as well which basically sees key whether uh the

00:57:36 code you are writing is not introducing any kind of you know security risk into the system right for example like you might be writing you might be you might be loging the password into PL test text right so there can be a rule saying ke uh log doino cannot have a ter called the pass right uh although it's like very naive but yeah like you can write these kind of stuff which make sure we uh even though you are not aware of these particular uh security or code uh quality points but you would uh be like

00:58:21 made it so the system would make sure that you add to these particular rules anyways uh the the next one is like remove single point of dependency which would basically means like let's say if you are the person who is who knows uh this uh particular C base let's say what function one even if you leave the organization or leave the team uh there are test cases which would basically make sure he uh your code base uh can be understood and no one can uh introduce a regression on top of your code base

00:58:57 right you don't have to do a code review of the new code base before merging into the mar so it removes the single point of dependency from the system so the leverage goes away from the person or the employee and goes back the system uh so uh any questions before we move to continuous delivery uh what is the meaning of uh code coverage yeah like very good question so basically when we write a particular piece of uh so when I talked about right let's say there is a function right right let's this there is a function right public

00:59:59 let's just and sum and let's say int a comma int B right so it what it would do is basically you need to do if your B is equal is equal to equal to zero right then you need to throw a new some kind of let's say exception right and else right you just return let's say a something uh so now you would want to like test this uh you would want to like write a test case for this particular use Cas right so you would uh basically what you are going to do is the call some right with let's say one and two right what it should do is the

01:01:08 uh the uh response from this has to be equal to equal to three right so this is your test case so this will make sure the if someone comes in and change this a uh a uh like return a minus B then this a some uh 1 comma 2 is not going to return so this test case is going to get BR it will get will break right so uh so the test like so in this particular case you are making sure so no one is basically introducing a regression into the code base itself so so this uh like it's a very simple example but like I think you will get

01:01:51 the gist of it but what if someone like uh like send this right in the production system someone sends one comma zero right so now in this particular case it should throw an exception right so uh like of course when you wrote this particular piece of code it actually throws an exception right or basically uh like please or gives you a very good message key like please uh don't like please don't uh please uh send b b as non zero right something like this but uh let's say if someone comes in and delete this particular piece of

01:02:40 right now if someone sends one comma 0 into your production system it's going to throw I think athematic exception which is not the desired output which you want right so in this particular case also what you you are going to do is that uh you would have to write two test cases some one comma 2 and some one comma zero so in this particular case it has you have to like uh so the message has to be uh please uh send b as non Z so in this particular case uh if someone changed this particular message

01:03:22 or removes the exception so what you are going to do like you will come to know about this with the help of broken test case so what would happen is when these test cases get uh uh like are being basically run so there are tools like let's say Joo or something else which would basically check what all parts of your code base are basically uh executed so when you run this particular sum 1 comma 3 uh what all pieces of your code base are actually EX executed one uh else and return uh a A plus b your if

01:04:02 statement is not executed so if this test case is not part of this you would basically come to know he uh three out of five uh lines into your code base are basically executed so your code cover is something like let's say 60% if you introduce this particular test case as well with the help of these two cases all of your five lines pieces of code are getting executed and your code coverage comes from five to five which is 100% so you would want to like have 100% as part of your like your service and does Auto means uh your

01:04:44 testes mostly yes but uh there are times when you would want to have integration integrated test cases as well uh and these are the like uh when you would want to test the functionalities acoss microservices as well or some complicated use case which cannot be uh managed in terms of let's say uh your unit test cases for example uh uh like uh event sourcing or something like that where you would basically say he you know uh let's say in particular case public in sum and here what you are doing is basically

01:05:30 saying ke uh post to the to the C the response so now how would you know whether it has pushed the response to the or so in this particular case what would need to be done is there would be another uh let's say API uh once you initiate this particular piece of code there has to be initiation of another piece of cod which would make sure the push to the C had so in this particular case uh like you would want much more complicated P of test case which are called integration test cases but mostly uh for most use cases yes automated test

01:06:14 case doesn't unit test Cas but yeah like it's not case on this cool uh uh continuous delivery I think this is quite straightforward we have like talked about most of this uh automated uh deployment steps um like so you would want to like as we talked about every single time you write a piece of code you would have to make do the exact bunch of same stuff so you would want to automate it because who would want to do like these boring stocks themselves so you would basically build uh complex uh deployment pipelines with the help of

01:06:59 these tools for example genkins uh feature flag help in going in any order so we talked about this if everyone like they can deploy in any order and like once the feature flag is enabled it would enable it for but this leads to lot of dead code which is a side effect that once that particular feature flag is enabled then like basically there might not be any use of that particular feature right because it is enabled for for long like for almost infinite unless like very small cases you want to rever back in case of

01:07:39 regressions but if there are no regs like the New Order flow is there to stay so you would have to like remove this particular Das card which basically checks that particular feature like in uh yeah it helps you scale right uh so if you are deploying in 100 machines or thousand machines it doesn't care the system takes care of yourself customer gets incremental features uh efficiency improves incrementally so you would not see you know the what set is X today and uh totally different uh the next day so you

01:08:21 would have these efficiencies or uh like features deled to you uh incrementally so you can give feedback the product can receive those particular feedback and work on those and like it takes the stress out of the developer so uh I think we have talked about this health SES right so these are the the systems or tools which you integrate your ser services and in your infrastructures which help in monitoring whether your services or infrastructures working in perfect conditions or so uh infrastructure API

01:09:10 transactions and its latencies all are being monitored continuously and uh probably you will get a uh uh like get a notification uh in the middle of the night man that he when this is wrong and and this requires your attention right so no one no no one has to manually take like this take a look at the systems finally so uh it helps when you know when you as a developer breaks any existing functionality into the system and slow down some of the key apis into your uh system as part of your deployment right so you would come to

01:09:52 know Auto like uh like very quickly and you can RT your code changes uh it would help uh if your deployment has nothing to do with it but your Downstream system screwed up so you can reach out to those particular people hey man the new deployment which you have done has screwed up of our system right so please R back and then like systems just go ha like they just go down because there are systems right like so for example the Facebook outage which happened like a month ago less than a month ago where uh like uh the system

01:10:35 start sending these uh uh like information to your ISP that or your routers e please remove uh these particular systems or the cloud uh infrastructure out of your out of out of your routing so when you are typing like uh facebook.com so it was not able to you know figure out which uh apis it needs to hit on so in like so none of the the Cod base or anything like went down in this particular case or it was not a human like it was not a human which screwed up like it was just happened like where a particular system

01:11:21 which was out of your reach or out of even your infrastructure like basically scre uh so you would want to like monitor them continuously your system so that like you can take an action as soon as possible right so this is like something like a new Rel would look like like one of these tools where you can basically see like how much of the traffic let's say a particular a so browse / plan. GSP this is taking delivering this particular resource is taking 60 26% of the time so yeah like like this is the

01:12:02 throughput you would also come to know like the latencies how much of the latencies are there on a particular resource itself so that you can basically uh fix or come to know whether there are issues of particular resource or not cloud computing platform I think this is like a quite forward and most of you have like known so you can set up alerts onto your Cloud platform so if a ec2 box goes down or if you have created an RDS system where there is a Master Slave architecture where uh you are writing

01:12:44 into your master but uh those rights are not getting replicated on your slave then uh AWS can help you to know and to know that uh that SL slave is like basically laging behind and all the reads which are happening onto your slave would be inconsistent so like so all those kind of need features can be hel like can be delivered with the help of the cloud computing you don't have to like manually to them so Auto scaling horizontal scaling to the microservices and all those things can be done with

01:13:21 the help so automatically it will like spawn e to machine with the help of let's say a particular image the the only thing here you would need to do uh like make sure is the uh like the the configurations the architecture which you have created is like basically correct so there are basically when you are adding into a Target group so the the traffic which is happening has to be through a Target group so when you add a particular machine to that like even if you are like scaling them manually it's basically just a tab onto

01:14:06 a screen that play spawn a particular machine and add to this particular Target Grow app everything takes care of itself uh so what are the best practices which can help you prepare for the Productions uh like as we talked about like writing test case would definitely help you when you write test cases it would give much more confidence when someone would alter the existing code base that they are not introducing regressions code reviews so uh it's when you are doing a like when you are reviewing someone course be

01:14:44 thorough uh that uh uh like whether the best practices in terms of like the writing of the code are being followed or not and uh like do introd like do configure code analysis tools something like sonar Cube or something which can you know help you with the Cod review there is one thing which I specifically wanted to talk about is in chaos engineering so what it basically so this is made popular primarily by so uh and uh it is taken up by Uber as well in know Big B so uh what happens is the U like everyone

01:15:33 of you like know Murphy's Law right he uh if like something can go wrong will go wrong so uh so if you know that there can be something which can which can go wrong in production why not mimic it right so what it basically does is it has a WR a like uh tool called chaos monkey uh Netflix which basically goes in and do bunch of crazy stuff like for example uh shutting down a particular ec2 machine uh changing the routing of a particular let's say particular note so the traffic will come might but might

01:16:18 not leave that particular box uh like apparently killing the production DV itself Master itself is one of the like this is one of the Fig like you never temper with the production DB itself but they say if the production DB can go down in like on its own like we are going to make sure that it goes down in let's say as part of your K engineering itself uh and see whether our system is able to uh recover from that particular event uh when I say recover the the the that the back pressures onto your

01:16:58 microservices are F the like the election of the new uh like Master Nord on top of your slave from one of your slaves is happening correctly or not so all those things uh they so it basically makes uh developers to be on their toes so that they basically uh deliver a cord B which would basic uh like so it's kind of a testing right so like you have to uh pass this particular test case as well before you move to the production so your test like your when you are delivering your uh code base has to run

01:17:42 through your kill engineering as well before it can be deployed into production now you can be basically be assured even if the worst comes happen into production servers uh you will not get a call into like middle of the night because the systems which you have built are so uh like redundant that they will take care of themselves so read about this like this is quite like interesting concept uh which has like many companies have introduced in the recent times to monitor deliver like large scale

01:18:21 systems I think that's it uh like if you have any equ questions this is the right time like we can go through some of those and I'm not sure like whether some like uh I I know like it was quite a lot of new stuff for you guys but I hope you followed along and you like took something from this particular class yeah is there any question yeah so uh so how that genkin is different from uh harness that you are working on this is not directly uh directly connected to what you have t but just I would like to know about the

01:19:17 harness so what exactly you doing different there yeah so uh as part of like Harless uh what we are basically doing is see uh so if you would see over here right uh in this particular uh so you have what you need to do is basically uh you need to so cic CD is already part of most of the tools right so you basically uh integrate uh so there has to be continuous integration so you need to basically monitor whether when to run the test cases and you build pipelines how to deliver uh like deliver the code base into the production

01:20:05 systems but uh the line after that is kind of proper right so in this like so uh one of those one of the major things is uh uh there so in genkins or other tools you don't uh get so the features you is until the point of a developer that you deploy and that's it you are forget about it right there is no uh like aggregation or uh of the data to tell uh let's say your engineering manager or your uh like other like up the folks of the chain you not how many deployments are going on and what are getting failed

01:20:49 and let's say if you're basically uh uh if let's say say your uh so since all of your services would be on a single platform what it can do is it can take the role of your Health Source as well basically telling you there was a deployment happened on service to or let's say micros service to and after that we started seeing issues into your micros service and probably like it uh it will give you an option to roll back the changes onto your Microsoft to as right so all those stuffs and then we are going beyond that particular point

01:21:30 so we are we are basically integrated to all the F sources collecting all the data let's say from Prometheus and let's say your neur and ab Dynamics and keep on checking the data itself in terms of like your and also what we are doing is basically uh we are making sure that a let's say your BC for example your BP can say man we want the availability of let's say triple right which would basically means 99.9% so on on on yearly like our system can go down for 1 hour itself so let's say the system has gone down for one

01:22:14 hour let's say this particular month so now uh it would take some time for it the system to you know uh like come back because uh as the your uh like your numerator like your denominator increases right because the amount of time it has been up your basically your downtime would keep on uh you know the it would keep on decrease in percentage wise so what you can do is basically I will not let uh people or Fox uh basically uh uh deploy until I reach uh uh the the certain set of uh let's say availability so what it does is

01:22:59 basically it uh restrict The Continuous delivery itself taking uh so although it is cont intuitive but it is as part of your business goal as well like you want ke your basically uh a particular let's say microservice is uh is down like many times in this particular week right so you do not want so you want to slow down that particular process figure out even what is wrong why those particular so many times the service was down is there a flow into your system of let's say uh delivery itself for your processes which

01:23:39 can be made rigid so that this doesn't happen but meanwhile it would basically block you uh you might have to take a like approval from some let's say higher authority key I want to this particular feature is really critical want to deliver it so like these kind of like we are going Beyond cicd to help the developer uh you know for the complete Journey itself that's why I said key xcid is just the start like you would have to look at in the complete set of key if there is an issue the system should detect automatically that there

01:24:16 was an issue with the like because of your deployment it it should roll back automatically so all those like cool stops can be taken care with the help of harness out of the box in like and like these building these particular pipelines as well uh are like quite intuitive and like straightforward in denkins as you know you would have to write most of this particular stuff you would have to tell what is the step one step step three so the integration your L sources of kubernetes and would have to

01:24:52 be so can we say that uh gen is more like uh rule based and you are adding some kind of Intelligence on top of gen K is it like so gen is mostly like a python script uh where you would basically say man I would basically you know do like execute this command then dependent depend on this execute this particular command which is neat for uh like most of the use cases you can a achieve any kind of stuff with the help of this but you would not want to do that thing over and over again right so if you have a

01:25:33 micros service you would want to quickly sayos uh because what is happening right now is the this uh deployments and all are getting concentrated uh so most of like we have this three uh uh Cloud platforms right so AWS so you would be deploying either in AWS or whether in let's say uh gcp or Azure right uh and most you will be using doctor or kubernetes as your like your containerization so in this particular case since the industry is getting you know St like consolidated you would want to build tools on top of it which

01:26:14 basically make it automated uh like deploying with using these particular conization on this particular Cloud platforms uh like jins when like when Jenkins came in it was like for the uh you can write scripts in any kind of environment for example you have your own in-house premise cloud or like or infrastructure where you basically deploy you can basically write a script where you would just say you an SS on this particular IP address and build on this particular IP address under this particular command

01:26:52 so gen gives you much more flexibility but the point is you don't want that kind of flexibility anymore uh because the processes are quite so you would want to save on the time itself and use you know like you can quickly write a microservices and write build a pipeline to deploy that particular okay thank you very much thank you anyone else yeah uh hi hey good evening uh thank you so much for a wonderful session uh so I have got a couple of questions uh first is like uh when does a cicd gets into uh product

01:27:35 development life cycle so what is the exact point when somebody should get into cicd uh another is like uh people still don't know how to code well right it needs some level of coding also understanding should be there um is there a easy way wherein CD set can be done without the help of a developer or do we need to have an expert for that yeah so uh CS is always part of your uh like development so uh like when you're depl like you when you're like developing particular piece of code you always have an I but this

01:28:19 particular code has to go into the production right so like when you write code like so that's the difference between like software developer and like a good software engineer right so it would have to think the complete cycle itself the moment product manager comes to you with the particular feature so you would have to question yourself uh how do I make key you know this is Backward Compatible uh does my feature depends on other microservices so all those things you will have to think even before you start

01:28:54 developing and cicd is also part of that right the particular uh piece of code which I'm writing right uh does it have the test cases so that is why the tdd approach like test D you know uh development is basically so popular where you basically do the development itself uh when you are like as part of writing the test cases itself so basically you cannot Escape writing test cases so so you the test coverage all those things is part of your uh like development cycle itself and the the matter of fact like your the the

01:29:40 software development Cycles which we are using for example aile we moved away from your waterfall model and we are moving like we have like almost the complete industry has moved to the agile module itself that inherently is cic right like the agile itself basically says you will have shorter uh Cycles or software developments and the moment you uh like do uh like develop you would have to like basically deliver that particular piece of C into the production system as so yeah so that's like the part one like it's always part

01:30:17 of your development cycle aile and T make sure of it the second part is uh the like the best practices uh well I like this comes with the like the more you read about it right so like um I have seen people writing test cases right just for the hack of it the people don't they don't understand like what is the use case or what is the significance of a test right so uh like that is uh like so writing good test cases also is an art like at part what particular Services you would have to Mo and at what level you would

01:31:00 have to Mo all those things are there right and then um like branching strategy in itself is like a very complex thing like if there are literally uh like anything more than 10 people if is working on a particular Branch uh you would have to have a Rob branching strategy like there can be three or four environments which branch is getting deployment deployed on which particular uh environment and how they are moving up the chain of let's say moving to the production then itself is like a very complicated stuff which comes primarily

01:31:40 with experience but the rest of the stuff for example writing a good writing like a good code piece of code I think that just comes with the commitment and someone can like start very early writing a good piece of cod go thank you so much yeah Mak sense anyone else I have one question uh so like consider we have various cicd vendors available in the market like Microsoft AWS so to choose with so what parameters you will consider before for your project so like I wanted to go for Microsoft or what parameters you will

01:32:33 consider so uh AWS and uh like is your uh are not the cicd uh vendors they are like Cloud vendors but for example let's say you want to choose from let's say genkins or let's say uh git lab or harness or any other you know cic vendor you have to choose from so like there are bunch of like parameters which you can take a look at uh one is uh of course uh the price right so for example Jenkins is free of cost and you have to deploy it on your servers and just uh you can start like integrating like building pipelines for

01:33:22 yourself but G lab hardness would be like you would have to have a paid version for it the second is ke uh like how how much it helps you like for example if you have if you are a very small organization if you have a couple of microservices itself then probably like a free version or probably like if you have a single microservice right so even writing a test space deploying har sorry junkins uh uh would not even make sense like doing it manually make much sense in this particular case but let's

01:33:57 say you have five microservices then like once in a while you for micr service as part of your yeah so uh as part of your uh like the total number of microservices can can be another parameter because you are writing them uh like working out new microservices once in a while so writing a new Pipeline on genkins won't be a tedious TK but let's say if you are a very large organization like for example Google or something right so there are multiple teams and multiple people who are fing out new micro Services every

01:34:47 single day then every single uh like hour saved into automating that whole process is the the like the time save for the organization so that can be another parameter the third one is ke uh for example what is your objectives for example uh if you wanted just for the developer uh probably like smaller tools like gen can be good enough but if you want much more complicated you want a view for your let's say uh let say engineering manager or your VP which can take a look into the complete like how the systems are

01:35:28 basically working and you know the which systems had most fiiled uh deployments and all those things then uh probably having harness would like like make you much more sense uh in this particular case good thank you sir anyone else uh cool I think then I think we can call it you're yeah yeah thank you Deepak uh it was a really wonderful session and full of practical experience I can see when you were and delivering the lect it was coming out of experience not from the books and all and I hope class will also

01:36:30 get benefits so we are actually recording these lectures and uploading on you Youtube private channel uh I hope you will not have any copyright issue if we upload your uh video and also please share your presentation if possible sure I'll share it with you yeah sure yeah thank you thank you again Deepak and uh from the whole class I would again wish you thank you thanks yeah thanks everyone it was

Summary

Mr. Deepa Chikara discussed CI/CD in SaaS cloud computing, emphasizing deployment automation tools, microservices health monitoring, and best practices for production.

Highlights

🛠️ CI/CD tools like Jenkins, GitLab, and Harness are essential for automating deployments.

⚙️ Monitoring health sources is crucial for maintaining microservices.

☁️ Cloud platforms like AWS, Azure, and GCP manage computational resources effectively.

🔄 Continuous integration allows for quick merging of code with automated testing.

✅ Writing automation test cases in isolation streamlines the development process.

📈 Best practices for production deployment enhance system reliability.

🙌 Mr. Chikara encourages questions and knowledge sharing post-session.

Key Insights

🛠️ Importance of CI/CD Tools: Tools like Jenkins, GitLab, and Harness automate deployment, reducing human error and accelerating release cycles, which is vital in today’s fast-paced environment.

⚙️ Microservices Health Monitoring: Regularly monitoring microservices ensures system reliability and quick issue resolution, which is essential for maintaining user satisfaction and operational efficiency.

☁️ Resource Management in Cloud Platforms: Utilizing cloud services like AWS, Azure, and GCP can optimize resource allocation, allowing for scalable and flexible computational power tailored to business needs.

🔄 Streamlined Continuous Integration: Quick merging of code into the master branch through continuous integration fosters a collaborative environment, minimizing integration issues and enhancing productivity.

✅ Best Practices for Production Deployment: Implementing best practices, such as thorough testing and documentation, prepares teams for smoother deployments, ultimately leading to fewer critical failures in production.

📈 Automation Testing in Isolation: Writing test cases in isolation helps catch bugs early in the development phase, ensuring that features work as expected before integration, which saves time and resources.

🙌 Knowledge Sharing and Support: Encouraging an open line for questions post-session fosters a learning culture, enabling teams to continuously improve and adapt to new challenges in CI/CD processes.