Reasons for CI/CD failures

there is bug in production ? why didnt we catch at before? how to avoid in future in the scenario we cant check the logs

1. servers are down

reasons :

production servers got updated --> the severs having monthly updates and then simply get down

application is down --> server is down --> ci cd wont make the contact with server because server is down

solutions : we can do monthly updates to the servers and this can happen through an automated program we can write through python or shellscript

2.you deployed wrong branch

reason : prod/release/master instead opf that we depolyed dev/qa/stage

application will be up but the updated code is not there then we can say bug in the production

solution:

check before deploying to the production,senior member or product owner on loop while doing the deployment

3.severs are not from the same configuration

dev - 2gb ram --> local db

stage -3gb ram --> local db

qa - 1 gb

production -6gb -->rds

these servers having different configurations

solution:

whatever the servers you take should be on same environment should be on same configuration

4.they are not build similarly

reason: 2010 ->on prem

2015 ->aws ec2 ->Manually

2023 --> IAC --> Terraform

when creating the resources by various ways ,if we are senior also will forget some details which is critical for deployement and that we done manually

solution : whenever you create the new envirtonment test everthing dont discard the old changes,once we are good with all changes then everthing will tested throughly by unit test,test cases or something

Jenkins Scenario based questions and answers

1.What are the different ways to trigger jenkins pipelines ?

- Poll SCM: Jenkins can periodically check the repository for changes and automatically build if changes are detected.

This can be configured in the "Build Triggers" section of a job.

- Build Triggers: Jenkins can be configured to use the Git plugin, which allows you to specify a Git repository and branch to build.

The plugin can be configured to automatically build when changes are pushed to the repository.

- Webhooks: A webhook can be created in GitHub to notify Jenkins when changes are pushed to the repository.

Jenkins can then automatically build the updated code. This can be set up in the "Build Triggers" section of a job and in the GitHub repository settings.

2.How to backup Jenkins ?

A: Backing up Jenkins is a very easy process, there are multiple default and configured files and folders in Jenkins that you might want to backup.

- Configuration: The `~/.jenkins` folder. You can use a tool like rsync to backup the entire directory to another location.

- Plugins: Backup the plugins installed in Jenkins by copying the plugins directory located in JENKINS\_HOME/plugins to another location.

- Jobs: Backup the Jenkins jobs by copying the jobs directory located in JENKINS\_HOME/jobs to another location.

- User Content: If you have added any custom content, such as build artifacts, scripts, or job configurations, to the Jenkins environment, make sure to backup those as well.

- Database Backup: If you are using a database to store information such as build results, you will need to backup the database separately. This typically involves using a database backup tool, such as mysqldump for MySQL, to export the data to another location.

One can schedule the backups to occur regularly, such as daily or weekly, to ensure that you always have a recent copy of your Jenkins environment available. You can use tools such as cron or Windows Task Scheduler to automate the backup process.

What is shared modules in Jenkins ?

A: Shared modules in Jenkins refer to a collection of reusable code and resources that can be shared across multiple Jenkins jobs. This allows for easier maintenance, reduced duplication, and improved consistency across multiple build processes. For example, shared modules can be used in cases like:

- Libraries: Custom Java libraries, shell scripts, and other resources that can be reused across multiple jobs.

- Jenkinsfile: A shared Jenkinsfile can be used to define the build process for multiple jobs, reducing duplication and making it easier to manage the build process for multiple projects.

- Plugins: Common plugins can be installed once as a shared module and reused across multiple jobs, reducing the overhead of managing plugins on individual jobs.

- Global Variables: Shared global variables can be defined and used across multiple jobs, making it easier to manage common build parameters such as version numbers, artifact repositories, and environment variables.

How to add a new plugin in Jenkins ?

A: Using the CLI, java -jar jenkins-cli.jar install-plugin <PLUGIN\_NAME>

Using the UI,

Click on the "Manage Jenkins" link in the left-side menu.

Click on the "Manage Plugins" link.