Devops Tutorial Sheet

Q 1 What are the fundamental differences between Devops and Agile?

Ans 1

Devops	**Agile**
1. DevOps is a software development approach that helps to pull together software development teams and IT activities.	**1. The agile philosophy is a software development methodology . It is founded on four ideals and twelve principles that assist in the development of a "agile" software development community.**
2.From the original design process to commercial release, it is a philosophy that fosters a community of teamwork between these two teams, which formerly worked in different silos.	**2. Agile encourages collaboration, self-organization, and responsibility by promoting implementation and a leadership mentality.**
3. The aim is to make it easy for teams to collaborate so that they can create, test, and release applications more easily and effectively.	**3. The agile methodology stresses constant synchronization with growth with consumer needs and patterns, even though those needs and trends shift late in the development process.**
4. DevOps puts two big, siloed teams together to facilitate faster product launches.	**4. Agile emphasizes on bringing smaller teams to communicate with one another so that they can adapt efficiently to evolving customer demands.**
5. DevOps prioritizes hyper-releases, beginning with a handful per day.	**5. Agile uses sprints to control the production process, which will run anything from a week to months.**
6. It puts together two typically distinct practises: creation and activities, which are generally kept separate.	**6. It stresses a production process that is iterative.**

Q 2 Why there is a need of Devops?

Ans 2 DevOps is an operational approach that allows quicker technology creation and smoother management of current implementations by integrating development (Dev) and operations (Ops) teams. DevOps encourages shorter, more controllable iterations by allowing companies to build deeper relationships with Dev, Ops, and other partners in the enterprise through the implementation of best practises, new tools and automation DevOps is not a technology in and of itself, but it incorporates everything from society to systems to tooling. Continuous integration and continuous distribution (CI/CD), real-time control, incident management services, and communication networks are typical first steps.

Benefits of Devops:-

1. DevOps, according to the late DevOps guru Robert Stroud, is all about "fueling market improvement," which involves improvements in individuals, systems, and culture. The most successful DevOps integration techniques concentrate on institutional changes that promote society. A successful DevOps project necessitates a culture—or mindset—shift that fosters greater cooperation among multiple teams—product, engineering, and operations, IT, logistics, and so on—as well as automation to help businesses accomplish their objectives.

2. It pushes companies to "optimise with the entire structure," not just IT silos, in order to boost the overall market. To put it another way, be more adaptable and data-driven in order to better comply with consumer and market demands.

3. The most critical aspect of a DevOps project is humans, not software. Key roleplayers (i.e., humans) such as a DevOps evangelist, a persuasive leader who can explain the market advantages of DevOps activities and dispel misconceptions and concerns, can significantly boost your prospects of success.

4. Top-performing DevOps companies succeed at software development/deployment speed and reliability, as well as ensuring that their product or service is available to end customers, which is a crucial operating necessity.

Q 3 What is the most important thing Devops helps us achieve?And explain with the help of example or a use case where Devops can be used in industrial or real life.

Ans 3 DevOps is a collection of tools and social techniques that allow a business to develop and deliver services and applications at a high pace. It is the most critical element in achieving the goal of integrating improvements into output. It assists in the establishment of productive working relationships and cooperation between teams.The Dev and Ops teams both produce high-quality software that increases customer satisfaction.

Real Life Examples of Devops:-

1. Online Financial Trading Comapny: In the financial trading company, the methodology for testing, building, and development was automated. Deployment was completed in 45 seconds thanks to DevOps. Employees used to have to work long nights and weekends during these deployments. The overall process took less time, and the clients' confidence rose.

2. Network Cycling: Deployment, research, and quick construction were all sped up tenfold. It was no longer difficult for the telco service provider to apply security updates every day, since it was formerly only performed every three months. The new version of network cycling was being carried out through rollout and architecture.

3. Faster Correction: Both faults are found very early with the aid of DevOps. As a result, errors can be resolved quickly. In this type of DevOps system, a lot of time is saved because the work is done quickly, and the corrective work is often completed quickly.

4. Increased Focus on Operations: With the support of DevOps, projects and processes take less time, allowing one person to concentrate more on the efficiency of activities and functions. There is more time for one to do high-quality work now that DevOps has reduced the amount of time spent on other programmes.

With all these examples , We can see that Devops helps us achieve high speed and high accuracy. In these real life examples main achievement they got using Devops was making the development easier and much faster. Also, it helps us reduce the time consumption and also a huge reduction in cost for a creation of Software. As, in this we can work in a continuous way like simultaneouly to achieve our goals.

Q 4 What is the version control and what are the benefits of using version control?

Ans 4 The process of monitoring and handling modifications to software code is known as version control, also known as source control. Version control systems (VCS) are automated instruments that aid software development teams in managing source code updates over time. Version management systems enable software teams perform quicker and better as programming environments have accelerated.They're particularly beneficial to DevOps teams, as they aid in reducing production time and increasing deployment performance.

In a specific kind of folder, version control software keeps track of any change to the file. When an error is made, engineers will go back in time to compare older iterations of the code to better resolve the issue while causing the least amount of inconvenience to the rest of the team.

Version management assists teams in resolving certain issues by monitoring each particular adjustment made by each contributor and preventing parallel work from overlapping. Changes made in one part of the programme could conflict with changes made by another developer working on the same project at the same time.

This concern should be established and addressed in a timely way so that the remainder of the team's work is not disrupted. Furthermore, any improvement in software development will add new bugs of its own, and new software cannot be trusted until it has been thoroughly checked. As a result, testing and development are carried out concurrently before a new release is available.

Benefits of Version Control:-

1. Per file has a full long-term change history : This refers to any of the changes created over time by a large number of people. The development and deleting of files, as well as changes to their contents, are both examples of changes. The ability of different VCS software to manage renaming and transferring files varies. The source, date, and written comments on the intent of each change should all be included in this history.

2. Merging and branching : It's a no-brainer to have team mates work at the same time, but even people working alone will benefit from the opportunity to work on separate sources of updates. Using VCS software to create a "branch" keeps different streams of work apart from one another while still allowing developers to merge the work back together, allowing developers to double-check that the modifications on each branch are not incompatible.

3. Traceability: Being able to monitor any change made to the programme and link it to project management and bug monitoring software like Jira, as well as annotating each change with a message outlining the change's meaning and intent, will aid not only in root cause analysis and other forensics. When you're translating the text, having the code's annotated past at your fingertips is invaluable. Understanding what it is doing and why it is built the way it is will help engineers make correct and harmonious improvements that are in line with the system's planned long-term architecture.

Q 5 What is Git? And mention some basic git Commands.

Ans 5 Git is a free and open source distributed version control framework that can accommodate anything from small to very big projects with ease.Git is simple to understand and use, with a small footprint and lightning-quick efficiency. With features like cheap local splitting, easy staging areas, and various workflows, it outperforms SCM resources like Subversion, CVS, Perforce, and ClearCase.

Basic Git Commands:-

1. Configure the author name and email address to be used with your commits:
git config --global user.name "Vishal Singla"
git config --global user.email singlavishal121@gmail.com
2. Create a new local repository:
git init
3. Create a working copy of a local repository:
git clone /path/to/repository
4. Add one or more files to staging (index):
git add <filename>
git add *
5. Commit changes to head (but not yet to the remote repository):
git commit -m "Commit message"
6. Send changes to the master branch of your remote repository:
git push origin master
7. List the files you've changed and those you still need to add or commit:
git status
8. If you haven't connected your local repository to a remote server, add the server to be able to push to it:
git remote add origin <server>
9. List all currently configured remote repositories:
git remote -v
10. Create a new branch and switch to it:
git checkout -b <branchname>

11. Fetch and merge changes on the remote server to your working directory:

git pull

12. To merge a different branch into your active branch:

git merge <branchname>

13. CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using:

git log

14. If you mess up, you can replace the changes in your working tree with the last content in head:

git checkout -- <filename>

15. Search the working directory for foo():

git grep "foo()"