|  |
| --- |
| ------------GIT---Interview Questions |
|  | **what is GIt?**    Git is a free, open source distributed version control system tool designed to handle everything from small to very large projects with speed and efficiency.  Version Control is the management of changes to documents, computer programs, large websites and other collection of information.  There are two types of VCS:   * Centralized Version Control System (CVCS) * Distributed Version Control System (DVCS) |
|  | 2. In Git how do you revert a commit that has already been pushed and made public?  **git revert** should be used to undo changes on a public branch, and **git** reset should be reserved for undoing changes on a private branch |
|  | 3. how do you find a list of files that has changed in a particular commit. ?    **For filenames only: 2014**  git show --name-only abc  **To see a summary of what happened to them:**  git show --name-status abc  **if you want to show changed files between two branches:**  git diff --name-status mybranch..myotherbranch 2017  Adding a grep can refine things further:  git diff --name-status mybranch..myotherbranch | grep "A\t" |
|  | **4. what is meant by continious intergration ?**  Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. While automated testing is not strictly part of CI it is typically implied. |
|  | ------------------------Jenkins--------------- |
|  | 1. what is Jenkins.   Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies. |
|  | 1. how will secure jenkins setup?   **Security Realm**   1. Select Manage **Jenkins**, then **Configure** Global **Security**. 2. Click Enable **Security**. The page will expand to offer a choice of access control. 3. Select **Jenkins**' own user database. 4. Place a check mark next to Allow users to sign up. 5. Continue with Authorization, below. |
|  | 1. Explain how can you create a backup and copy files in Jenkins?   o **create** a **backup**, all you need to do is to periodically **back up** your JENKINS\_HOME directory. This contains all of your **build** jobs configurations, your slave node configurations, and your **build** history. To **create** a **back-up** of your**Jenkins** setup, just **copy** this directory. |
|  | -----------------Nagios------------------- |
|  | 1.what is nagios.? |
|  | 2. how does nagios work.? |
|  | 3. what is NRPE in nagios? |
|  | -------------------docker----------------- |
|  | 1. what is docker ?   Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.   Docker is a containerization platform that packages your application and all its dependencies together in the form of a docker container to ensure that your application works seamlessly in any environment. |
|  | 1. what is docker images?    An image is a combination of a file system and parameters. Let’s take an example of the following command in Docker.  docker run hello-world   * The Docker command is specific and tells the Docker program on the Operating System that something needs to be done. * The **run** command is used to mention that we want to create an instance of an image, which is then called a **container**. * Finally, "hello-world" represents the image from which the container is made. |
|  | 1. what is dockerfile?   Docker builds images automatically by reading the instructions from a Dockerfile -- a text file that contains all commands, in order, needed to build a given image. A Dockerfile adheres to a specific format and set of instructions which you can find at [Dockerfile reference](https://docs.docker.com/engine/reference/builder/).  A Docker image consists of read-only layers each of which represents a Dockerfile instruction. The layers are stacked and each one is a delta of the changes from the previous layer. Consider this Dockerfile:  FROM ubuntu:15.04  COPY . /app  RUN make /app  CMD python /app/app.py  Each instruction creates one layer:   * FROM creates a layer from the ubuntu:15.04 Docker image. * COPY adds files from your Docker client’s current directory. * RUN builds your application with make. * CMD specifies what command to run within the container.   When you run an image and generate a container, you add a new *writable layer* (the “container layer”) on top of the underlying layers.  **Build context example**  Create a directory for the build context and cd into it. Write “hello” into a text file named hello and create a Dockerfile that runs cat on it. Build the image from within the build context (.):  mkdir myproject && cd myproject  echo "hello" > hello  echo -e "FROM busybox\nCOPY /hello /\nRUN cat /hello" > Dockerfile  docker build -t helloapp:v1 .  Move Dockerfile and hello into separate directories and build a second version of the image (without relying on cache from the last build). Use -f to point to the Dockerfile and specify the directory of the build context:  mkdir -p dockerfiles context  mv Dockerfile dockerfiles && mv hello context  docker build --no-cache -t helloapp:v2 -f dockerfiles/Dockerfile context |
|  | 1. write a dockerfile to create an image to install apache.?   **Method1:**  [**https://www.tutorialspoint.com/docker/building\_web\_server\_docker\_file.htm**](https://www.tutorialspoint.com/docker/building_web_server_docker_file.htm)  **Step 1** − The first step is to build our Docker File. Let’s use **vim** and create a Docker File with the following information.  FROM ubuntu  RUN apt-get update  RUN apt-get install –y apache2  RUN apt-get install –y apache2-utils  RUN apt-get clean  EXPOSE 80 CMD [“apache2ctl”, “-D”, “FOREGROUND”]  **Step 2** − Run the Docker **build** command to build the Docker file. It can be done using the following command −  sudo docker build –t=”mywebserver” .  **Step 3** − Now that the web server file has been built, it’s now time to create a container from the image. We can do this with the Docker **run** command.  sudo docker run –d –p 80:80 mywebserver  **Method2:**  **DockerFile**  FROM ubuntu:14.04  RUN apt-get -y install tomcat7  RUN echo "JAVA\_HOME=/usr/lib/jvm/java-7-oracle" >> /etc/default/tomcat7  # Expose the default tomcat port  EXPOSE 8080  # Start the tomcat (and leave it hanging)  CMD service tomcat7 start && tail -f /var/lib/tomcat7/logs/catalina.out  build the image simply use [docker build](https://docs.docker.com/userguide/dockerimages/#building-an-image-from-a-dockerfile):  docker build -t my/tomcat .  To start the container you must [mount a volume](https://docs.docker.com/userguide/dockervolumes/) with your war-file.  docker run -v /somefolder/myapp:/var/lib/tomcat7/webapps/myapp -p 8080:8080 my/tomcat  docker run –v destfolder/source folder where the war file to created directory |
|  | --------------------AWS--------------------- |
|  | 1. what is VPC. |
|  | 2. what is public and private VPC. |
|  | 3. what is reserved instances? |
|  | 4. what is cloud formation? |
|  | 5. what is lambda? |
|  | 6. have you used Route 53? |
|  | 7. what is the best feature of AWS? |
|  | 8. what is difference between chef and ansible? |
|  | 9. what is dedicated instance ? |
|  | 10. what is spot intances? |
|  | 11. how do you choose avialbility zone ? |
|  | 12. what are the service models AWS provides ? |
|  | 13. how can you setup Weighted policy in route53. |
|  | 14. what is the relation between ec2 and AMI. ? |
|  | 15. what is dynamo DB , how it diffrs from RDS.? |
|  | ------------------------------------------------------General Question-------------------------------------- |
|  | 1. whey we need devops.?  DevOps integrates developers and operations team to improve collaboration and productivity.  According to the DevOps culture, a single group of Engineers (developers, system admins, QA’s. Testers etc turned into DevOps Engineers) has end to end responsibility of the Application (Software) right from gathering the requirement to development, to testing, to infrastructure deployment, to application deployment and finally monitoring & gathering feedback from the end users, then again implementing the changes.  This is a never ending cycle and the logo of DevOps makes perfect sense to me. Just look at the above diagram – What could have been a better symbol than infinity to symbolize DevOps?  https://qph.ec.quoracdn.net/main-qimg-5b6e8c9ade317b8f718b8fd6490f1b82  Now let us see how DevOps takes care of the challenges faced by Development and Operations.  https://qph.ec.quoracdn.net/main-qimg-bcef648551cf42c627cad87d0209c482  Below table describes how DevOps addresses Dev Challenges. |
|  | 2. what are the popular devops tools that you use?  <https://www.quora.com/Why-do-we-need-DevOps>  GIT- To maintain the different versions of source code  JIRA – Monitoring tool and to resolve the ticket raised from the production server.  Project uses the CICD technique using Jenkins.Build can be done by Maven Tool.  Testing: JUNIT,Selenium  Docker - Containerization  Jenkins: CICD Technique  Build Tools: Ant,Maven,Gradle  Maven :Pom.xml  ANT: Build.xml  Gradle: Build.gradle  Puppet,ANSIBLE,CHEF – Configuration Management tool – infrastructure management – Software installed on all nodes using single script file.  CHEF –  ANSIBLE – Playbook.YAML |
|  | 1. what is the typical DevOps Workflow you use in your organistaion.?   Our project is an maintenance project. Issues and customizations are logged from JIRA(Nagios Monitoring) tool as tickets.  If there are changes in the code are updated and committed in the centralized repositories from all the nodes.(or Master/Slave).  If there are any commits to the GITHUB which in turn triggers the Jenkins server for continuous integration and buildcan be done by using the Maven Tool.All software configurations are done by configuration management tool ANSIBLE.  Using (YAML –human readable language).Deployment done using the  Docker tool. |
|  | **How do you take devOps approach with Amazon web Services?** |
|  | 1. How will you run a script automatically on commit in GIT?   <https://www.digitalocean.com/community/tutorials/how-to-use-git-hooks-to-automate-development-and-deployment-tasks>  Method1:GitHooks  Method2:PollSCM  gitwatch.Service   |  | | --- | | [Unit] | |  | Description=Watch file or directory and git commit all changes | |  |  | |  | [Service] | |  | ExecStart=/opt/gitwatch/gitwatch.sh %I | |  | ExecStop=/bin/true |     gitwatch.sh  <https://github.com/gitwatch/gitwatch/blob/master/gitwatch.sh>  Torun the GIT Hooks need to spin the public IP.  Git hooks are event-based. When you run certain git commands, the software will check the hooksdirectory within the git repository to see if there is an associated script to run.  Some scripts run prior to an action taking place, which can be used to ensure code compliance to standards, for sanity checking, or to set up an environment. Other scripts run after an event in order to deploy code, re-establish correct permissions (something git cannot track very well), and so forth.   * Client-Side Hooks: Hooks that are called and executed on the committer's computer. These in turn are divided into a few separate categories:   + Committing-Workflow hooks: Committing hooks are used to dictate actions that should be taken around when a commit is being made.   + Email Workflow hooks: This category of hooks encompasses actions that are taken when working with emailed patches. Projects like the Linux kernel submit and review patches using an email method. These are in a similar vein as the commit hooks, but can be used by maintainers who are responsible for applying submitted code.   + Other: Other client-side hooks include hooks that execute when merging, checking out code, rebasing, rewriting, and cleaning repos. * Server-Side Hooks: These hooks are executed on servers that are used to receive pushes. Generally, that would be the main git repo for a project.   + Pre-receive and post-receive: These are executed on the server receiving a push to do things like check for project conformance and to deploy after a push.   + Update: This is like a pre-receive, but operates on a branch-by-branch basis to execute code prior to each branch being accepted. |
|  | 1. How does CloudFormation work in AWS? |
|  | 1. What is CICD in Devops?   Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. While automated testing is not strictly part of CI it is typically implied.  The idea is to minimize the cost of integration by making it an early consideration.  **Continuous delivery** is an extension of continuous integration. It focuses on automating the software delivery process so that teams can easily and confidently deploy their code to production at any time |
|  | 1. what are the best practices of CI?   **Small, Iterative Changes**  One of the most important practices when adopting continuous integration is to encourage small changes. Developers should practice breaking up larger work into small pieces and committing those early.  Small changes minimize the possibility and impact of integration problems. By committing to the shared branch at the earliest possible stage and then continually throughout development, the cost of integration is diminished and unrelated work is synchronized regularly. Trunk-Based Development With trunk-based development, work is done in the main branch of the repository or merged back into the shared repository at frequent intervals. Short-lived feature branches are permissible as long as they represent small changes and are merged back as soon as possible. Keep the Building and Testing Phases FastConsistency Throughout the Deployment Pipeline **Code should be built once at the beginning of the pipeline**: The resulting software should be stored and accessible to later processes without rebuilding. -**Artifacts** ****Deployment environments should be consistent****: A configuration management system can control the various environments, and environmental changes can be put through the deployment pipeline itself to ensure correctness and consistency **Consistent processes should be used to deploy the build in each environment**: Each deployment should be automated and each deployment should use the same centralized tools and procedures. Decouple Deployment and Release Separating the deployment of code from its release to users is an extremely powerful part of continuous delivery and deployment. Code can be deployed to production without initially activating it or making it accessible to users. |
|  | 1. what are the beneffts of CI?  Benefits of Continuous IntegrationCI and CD enable teams to safely build, test, and deploy code. Automating testing through Continuous Integration increases code quality.1.Manual Tests Are Only a Snapshot2. Increase Your Code Coverage 3. Increase Visibility Across the Team 4. Deploy Your Code to Production5. Build Stuff Now6. Build Stuff Faster7. Never Ship Broken Code8. Decrease Code Review Time9. Build Repeatable Processes |
|  | 1. what are the options for security in jenkins?   Security realm refer above question |
|  | --------------------------------------------------Ansible------------------------------- |
|  | 1. what are the main beniftes of ansible.?   1 .Automate complex tasks without trouble.  2.Allows you to break up configurations into more modular.   * 3. **No Agent**.As long as the box can be ssh’d into and it has python, it can be configured with Ansible. * **Idempotent**. Ansible’s whole architecture is structured around the concept of idempotency. The core idea here is that you only do things if they are needed and that things are repeatable without side effects * **Declarative Not Procedural**. Other configuration tools tend to be procedural — do this and then do that and so on. Ansible works by you writing a description of the state of the machine that you want and then it takes steps to fulfill that description. |
|  | 1. what is the architecure of ansible.?   Ansible is a simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.  https://www.packtpub.com/graphics/9781783550630/graphics/0630OT_01_01.jpg |
|  | 1. what is a playbook in ansible?   Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e. **YAML** (It’s a human-readable data serialization language & is commonly used for configuration files, but could be used in many applications where data is being stored(Readable format).  Advantage is that even the IT infrastructure support guys can read and understand the playbook and debug if needed. Ansible is easy to deploy because it does not use any agents or custom security infrastructure |
|  | 1. what are the main beneifts of anisble.?   Advantage is that even the IT infrastructure support guys can read and understand the playbook and debug if needed. Ansible is easy to deploy because it does not use any agents or custom security infrastructure  Ansible is completely **agentless** which means Ansible works by connecting your nodes through **ssh(by default).**  After connecting to your nodes, Ansible pushes small programs called as “Ansible Modules”. Ansible runs that modules on your nodes and removes them when finished. Ansible manages your inventory in simple text files (These are the hosts file).  **Ansible** is an open source automation platform. ... **Ansible** can help you with configuration management, application deployment, task automation. It can also do IT orchestration, where you have to run tasks in sequence and create a chain of events which must happen on several different servers or devices. |
|  | 1. what are the main use cases of ansible?   ts a configuration management tool. When used with [Ansible Tower](https://www.quora.com/topic/Ansible-Tower) or [Ansible AWX](https://www.quora.com/topic/Ansible-AWX) you also get a powerful configuration console and task scheduler. Also, secure by design and use of best practises.  [Ansible](https://www.quora.com/topic/Ansible) allows you to centralize (or run stand alone if you wish) the automated configuration for all your server, desktop and rack devices, as long as your devices has the ability to accept SSH or WinRM connections.  Additionally, you can use it to provision infrastructure on VM and cloud providers.  I currently have used it to configure   * Vagrant / VirtualBox, VMs on the desktop, and for development for cloud * AWS Provisioning - EC2, ELB, ASG, S3 * DigitalOcean Droplet configuration * Linux / Windows   I’ve been able to configure webservers, databases, certs, repositories / deployment pipelines, user configuration … etc. |
|  | ---------------------------------------------------------------------------------------- |
|  | 1. what is docker hub?   **Docker Hub** is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts.  -------------------------------------------------------------------------------------- |
|  | 1. what is Multi factor authentication.?   **Multifactor authentication** (MFA) is a security system that requires more than one method of**authentication** from independent categories of credentials to verify the user's identity for a login or other transaction  **----------------------------------------------------------------------------------------------------------------------** |
|  | 18. what are the main beniftes of nagios ? |
|  | 19. what are the main features of nagios? |
|  | 20. what is the use of kubernetes? |
|  | 21. what is the architecure of Kubernetes? |
|  | 22. How does kubernetes provide high availability of applications in a cluster.  --------------------------------------------------------------------------------------- |
|  | 23. what is soruce code management .? |
|  | 24. what are the main services of AWS that you have used? |
|  | 25. why GIt is considered better than CVS for version control system?  Git: It is a Version Control Software that helps you to keep a track of all the changes you made in your source code by storing each change as a “version”.  GitHub: It is a website that lets you host Git repositories. This is the place where you store your production ready code.  **CVS** is (old) centralized version control system, while **Git** is distributed. But even if you use version control for single developer, on single machine (single account), there are a few differences between **Git** and **CVS**: Setting up repository. |
|  | 26. what is the difference between a container and a virtual machine?4 Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. Containers are more portable and efficient.**CONTAINERS** Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems. **VIRTUAL MACHINES** Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.  https://www.docker.com/sites/default/files/styles/content_6_6/public/compare/docker-containerized-appliction-blue-border_2.png?itok=lsxRQ9HU  https://www.docker.com/sites/default/files/styles/content_6_6/public/compare/container-vm-whatcontainer_2.png?itok=0eNn5aap |
|  | 27.What are the main principles of DevOps? DevOps Principles  * Iterative * Incremental * Continuous * Automated * Self-service * Collaborative * Holistic   **A few identified DevOps practices include:**   * **Self**-service configuration. * Automated provisioning. * Continuous build. * **Continuous integration**. * Continuous delivery. * Automated release **management**. * Incremental **testing**. |
|  | 28.Are you more Dev or more Ops? |
|  | 29.What is a REST service? |
|  | 30.How do you apply DevOps principles to make system Secure?  **I. Automated Security Testing:**We automate and integrate Security testing techniques for Software Penetration testing and Fuzz testing in software development process.  **II. Early Security Checks:**We ensure that teams know about the security concerns at the beginning of a project, rather than at the end of delivery. It is achieved by conducting Security trainings and knowledge sharing sessions.  **III. Standard Process:** At DevOps we try to follow standard deployment and development process that has already gone through security audits. This helps in minimizing the introduction of any new security loopholes due to change in the standard process. |
|  | 31.What is Self-testing Code?  **Self**-**testing code** is software that incorporates built-in **tests** (see **test**-first development).  Self-testing Code is an important feature of DevOps culture. In DevOps culture, development team members are expected to write self-testing code.  It means we have to write code along with the tests that can test this code. Once the test passes, we feel confident to release the code. |
|  | 32.What is a Deployment Pipeline?  At an abstract level, a **deployment pipeline** is an automated manifestation of your process for getting software from version control into the hands of your users. Every change to your software goes through a complex process on its way to being released. That process involves building the software, followed by the progress of these builds through multiple stages of testing and deployment. This, in turn, requires collaboration between many individuals, and perhaps several teams. The deployment pipeline models this process, and its incarnation in a continuous integration and release management tool is what allows you to see and control the progress of each change as it moves from version control through various sets of tests and deployments to release to users. |
|  | 33.What are the main features of Docker Hub? |
|  | * Image Repositories: Find and pull images from community and official libraries, and manage, push to, and pull from private image libraries to which you have access. * Automated Builds: Automatically create new images when you make changes to a source code repository. |
|  | 35.How many heads can you create in a GIT repository?  A head is simply a reference to a commit object. Each head has a name (branch name or tag name, etc). By default, there is a head in every repository called master. A repository can contain any number of heads. At any given time, one head is selected as the “current head.” This head is aliased to HEAD, always in capitals".  Note this difference: a “head” (lowercase) refers to any one of the named heads in the repository; “HEAD” (uppercase) refers exclusively to the currently active head. This distinction is used frequently in Git documentation.  There can only be one HEAD - the symbolic reference that points to either the branch (which itself points to a sha1), or the sha1 itself (when you are in a detached head state), which you currently have checked out, and which will be the parent of your next commit. |
|  | 36.What is a Passive check in Nagios? |
|  | 37.What is a Docker container?  **Docker container** is an open source software development platform. Its main benefit is to package applications in “**containers**,” allowing them to be portable among any system running the Linux operating system (OS). |
|  | 38.How will you remove an image from Docker? Remove one or more specific images Use the docker images command with the -a flag to locate the ID of the images you want to remove. This will show you every image, including intermediate image layers. When you've located the images you want to delete, you can pass their ID or tag to docker rmi:  **List:**   * docker images -a   **Remove:**   * docker rmi Image Image   docker rm ID\_or\_Name ID\_or\_Name Remove a container upon exit docker run --rm image\_name |
|  | 39.What are the common use cases of Docker? **Docker’s Key use cases**1. Simplifying Configuration The primary use case Docker advocates is simplifying configuration.. It lets you put your environment and configuration into code and deploy it. The same Docker configuration can also be used in a variety of environments. This decouples infrastructure requirements from the application environment. 2. Code Pipeline Management Docker provides a consistent environment for the application from dev through production, easing the code development and deployment pipeline. 3. Developer Productivity In turn, this leads to some additional advantages Docker delivers for a dev productivity use case. 4. App Isolation There may be many reasons for which you end up running multiple applications on the same machine. 5. Server Consolidation Just like using VMs for consolidating multiple applications, the application isolation abilities of Docker allows consolidating multiple servers to save on cost. However, without the memory footprint of multiple OSes and the ability to share unused memory across the instances, Docker provides far denser server consolidation than you can get with VMs. 6. Debugging Capabilities Docker provides many tools that are not necessarily specific to containers, but, they work well with the concept of containers. They also provide extremely useful functionality. This includes the ability to checkpoint containers and container versions, as well as to diff two containers. This can be immensely useful in fixing an application. 7. Multi-tenancy Yet another interesting use case of Docker is its use in multi-tenant applications, thereby avoiding major application rewrites.  Our own example is to develop quick and easy multi-tenancy for an IoT application. Code bases for such multi-tenant applications are far more complicated, rigid and difficult to handle. Rearchitecting an application is not only time consuming, but also costs a lot of money.  Using Docker, it was easy and inexpensive to create isolated environments for running multiple instances of app tiers for each tenant. This was possible given the spin up speed of Docker environments and it’s easy-to-use API, which we can use to spin containers programmatically. We used docker-py, which is a Python library to help interact with the Docker daemon through a web application interface.  ###8. Rapid Deployment |
|  | 40.Can we lose our data when a Docker Container exits? |

Docker provides the --rm command line option for this purpose:

docker run --rm my-docker

This will create a container from the "my-docker" image and delete the container immediately after it exits. This helps to prevent having to clean up containers after you're done experimenting.