**DevOps How can you define it in your words?**

DevOps is highly effective daily collaboration between software developers and IT operations / web operation engineers to release a software. It’s generally aligned with Agile methodologies to deploy software to production environment, as a DevOps engineer we will be working with Agile development teams to ensure they have an environment necessary to support functions such as automated testing, continuous integration, and continuous Delivery.

**Why we need DevOps?**

Companies are now facing the need to delivering more and faster and better applications to meet the ever more pressing demands of conscious users to reduce the " Time to Market ". Devops often helps deployment to happen very fast.

**What is agile development and Scrum?**

Agile development used a s an alternative to Waterfall development practice. In Agile, the development process is more iterative and incremental, there is more testing and feedback at every stage of development as opposed to only the last stage in Waterfall.

Scrum is used to manage complex software and product development, using iterative and incremental practices. Scrum has three roles ie product owner, scrum master, and team.

**Can we consider DevOps as an agile methodology?**

Of course! DevOps is a movement to reconcile and synchronize development and production start through a set of good practices. Its emergence is motivated by a deep changing demand of business, who want to speed up the changes to stick closer to the requirements of business and the customer.

**What is DevOps engineer's duty with regards to Agile development?**

DevOps engineer work very closely with Agile development teams to ensure they have an environment necessary to support functions such as automated testing, continuous Integration and continuous Delivery. DevOps engineer must be in constant contact with the developers and make all required parts of environment work seamlessly.

**What is the need for DevOps?**

According to me, this answer should start by explaining the general market trend. Instead of releasing big sets of features, companies are trying to see if small features can be transported to their customers through a series of release trains. This has many advantages like quick feedback from customers, better quality of software etc. which in turn leads to high customer satisfaction. To achieve this, companies are required to:

1. Increase deployment frequency
2. Lower failure rate of new releases
3. Shortened lead time between fixes
4. Faster mean time to recovery in the event of new release crashing

DevOps fulfills all these requirements and helps in achieving seamless software delivery. You can give examples of companies like Etsy, Google and Amazon which have adopted DevOps to achieve levels of performance that were unthinkable even five years ago. They are doing tens, hundreds or even thousands of code deployments per day while delivering world class stability, reliability and security.

**How is DevOps different from Agile / SDLC?**

I would advise you to go with the below explanation:

Agile is a set of values and principles about how to produce i.e. develop software. Example: if you have some ideas and you want to turn those ideas into working software, you can use the Agile values and principles as a way to do that. But, that software might only be working on a developer’s laptop or in a test environment. You want a way to quickly, easily and repeatedly move that software into production infrastructure, in a safe and uncomplicated way. To do that you need DevOps tools and techniques.

You can summarize by saying Agile software development methodology focuses on the development of software but DevOps on the other hand is responsible for development as well as deployment of the software in the safest and most reliable way possible.

Now remember, you have included DevOps tools in your previous answer so be prepared to answer some questions related to that.

**Which technologies can act as driver to enable DevOps ?**

* **Paas**: which is a category of cloud computing services that provides a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure
* **Iaas**: which is a category of cloud computing services that abstract the user from the details of infrastructure like physical computing resources, location, data partitioning, scaling, security, backup etc.
* **Configuration** **automation**: Automation is a big win in part because it eliminates the labor associated with repetitive tasks. Codifying such tasks also means documenting them and ensuring that they’re performed correctly, in a safe manner, and repeatedly across different infrastructure types.
* **Microservices**: which consists in a particular way of designing software applications as suites of independently deployable services.
* **Containers**:  Containers modernize IT environments and processes, and provide a flexible foundation for implementing DevOps. At the organizational level, containers allow for appropriate ownership of the technology stack and processes, reducing hand-offs and the costly change coordination that comes with them.

**Which are the top DevOps tools? Which tools have you worked on?**

The most popular DevOps tools are mentioned below:

* Git: Version Control System tool
* Jenkins: Continuous Integration tool
* Selenium: Continuous Testing tool
* Puppet, Chef, Ansible: Configuration Management and Deployment tools
* Nagios: Continuous Monitoring tool
* Docker: Containerization tool

You can also mention any other tool if you want, but make sure you include the above tools in your answer.

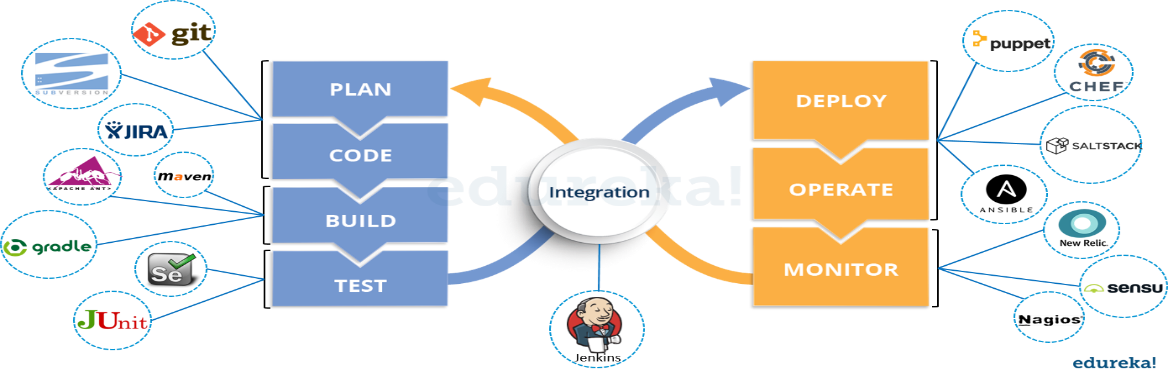
The second part of the answer has two possibilities:

1. If you have experience with all the above tools then you can say that I have worked on all these tools for developing good quality software and deploying those software’s easily, frequently, and reliably.
2. If you have experience only with some of the above tools then mention those tools and say that I have specialization in these tools and have an overview about the rest of the tools.

**How do all these tools work together?**

Given below is a generic logical flow where everything gets automated for seamless delivery. However, this flow may vary from organization to organization as per the requirement.

1. Developers develop the code and this source code is managed by Version Control System tools like Git etc.
2. Developers send this code to the Git repository and any changes made in the code is committed to this Repository.
3. Jenkins pulls this code from the repository using the Git plugin and build it using tools like Ant or Maven.
4. Configuration management tools like puppet deploys & provisions testing environment and then Jenkins releases this code on the test environment on which testing is done using tools like selenium.
5. Once the code is tested, Jenkins send it for deployment on the production server (even production server is provisioned & maintained by tools like puppet).
6. After deployment, It is continuously monitored by tools like Nagios.
7. Docker containers provides testing environment to test the build features.



**What are the advantages of DevOps?**

Technical benefits:

* Continuous software delivery
* Less complex problems to fix
* Faster resolution of problems

Business benefits:

* Faster delivery of features
* More stable operating environments
* More time available to add value (rather than fix/maintain)

**What is the most important thing DevOps helps us achieve?**

According to me, the most important thing that DevOps helps us achieve is to get the changes into production as quickly as possible while minimizing risks in software quality assurance and compliance. This is the primary objective of DevOps.

However, you can add many other positive effects of DevOps. For example, clearer communication and better working relationships between teams i.e. both the Ops team and Dev team collaborate to deliver superior quality software which in turn leads to higher customer satisfaction.

**Explain with a use case where DevOps can be used in industry / real-life.**

There are many industries that are using DevOps so you can mention any of those use cases, you can also refer the below example:

Etsy is a peer-to-peer e-commerce website focused on handmade or vintage items and supplies, as well as unique factory-manufactured items. Etsy struggled with slow, painful site updates that frequently caused the site to go down. It affected sales for millions of Etsy’s users who sold goods through online market place and risked driving them to the competitor.

With the help of a new technical management team, Etsy transitioned from its waterfall model, which produced four-hour full-site deployments twice weekly, to a more agile approach. Today, it has a fully automated deployment pipeline, and its continuous delivery practices have reportedly resulted in more than 50 deployments a day with fewer disruptions.

**Explain your understanding and expertise on both the software development side and the technical operations side of an organization you have worked with in the past.**

For this answer, share your past experience and try to explain how flexible you were in your previous job. You can refer the below example:

DevOps engineers almost always work in a 24/7 business-critical online environment. I was adaptable to on-call duties and was available to take up real-time, live-system responsibility. I successfully automated processes to support continuous software deployments. I have experience with public/private clouds, tools like Chef or Puppet, scripting and automation with tools like Python and PHP, and a background in Agile.

**Give me an example of how you would handle projects?**

As a DevOps engineer, I would demonstrate a clear understanding of DevOps project management tactics and also work with teams to set objectives, streamline workflow, maintain scope, research and introduce new tools or frameworks, translate requirements into workflow and follow up. I would resort to CI, release management and other tools to keep interdisciplinary projects on track.

**What are the anti-patterns of DevOps?**

A pattern is common usage usually followed. If a pattern commonly adopted by others does not work for your organization and you continue to blindly follow it, you are essentially adopting an anti-pattern. There are myths about DevOps. Some of them include:

* DevOps is a process
* Agile equals DevOps?
* We need a separate DevOps group
* DevOps will solve all our problems
* DevOps means Developers Managing Production
* DevOps is Development-driven release management
  1. DevOps is not development driven.
  2. DevOps is not IT Operations driven.
* We can’t do DevOps – We’re Unique
* We can’t do DevOps – We’ve got the wrong people

**What’s your systems background?**

* Tips to answer: Some DevOps jobs require extensive systems knowledge, including server clustering and highly concurrent systems. As a DevOps engineer, you need to analyze system capabilities and implement upgrades for efficiency, scalability and stability, or resilience. It is recommended that you have a solid knowledge of OSes and supporting technologies, like network security, virtual private networks and proxy server configuration.
* DevOps relies on virtualization for rapid workload provisioning and allocating compute resources to new VMs to support the next rollout, so it is useful to have in-depth knowledge around popular hypervisors. This should ideally include backup, migration and lifecycle management tactics to protect, optimize and eventually recover computing resources. Some environments may emphasize microservices software development tailored for virtual containers. Operations expertise must include extensive knowledge of systems management tools like Microsoft System Center, Puppet, Nagios and Chef. DevOps jobs with an emphasis on operations require detailed problem-solving, troubleshooting and analytical skills.

**What DevOps tools have you worked with?**

* Tips to answer: Software configuration management and build/release (version control) tools, including Apache Subversion, Mercurial, Fossil and others, help document change requests. Developers can more easily follow the company’s best practices and policies while software changes.
* Continuous integration (CI) tools such as Rational Build Forge, Jenkins and Semaphore merge all developer copies of the working code into a central version. These tools are important for larger groups where teams of developers work on the same codebase simultaneously. QA experts use code analyzers to test software for bugs, security and performance. If you’ve used HP’s Fortify Static Code Analyzer, talk about how it identified security vulnerabilities in coding languages. Also speak about tools like GrammaTech’s CodeSonar that you used to identify memory leaks, buffer underruns and other defects for C/C++ and Java code. It is essential that you have adequate command of the principal languages like Ruby, C#, .NET, Perl, Python, Java, PHP, Windows PowerShell, and are comfortable with the associated OS environments Windows, Linux and Unix.

**How much have you interacted with cloud based software development?**

* Tips to answer: Share your knowledge around use of cloud platforms, provisioning new instances, coding new software iterations with the cloud provider’s APIs or software development kits, configuring clusters to scale computing capacity, managing workload lifecycles and so on. This is the perfect opportunity to discuss container-based cloud instances as an alternative to conventional VMs. Event-based cloud computing, such as AWS Lambda offers another approach to software development, a boon for experienced DevOps candidates. In your interview, mention experience handling big data, which uses highly scalable cloud infrastructures to tackle complex computing tasks.

**What other tools are you familiar with that might help you in this role?**

* Tips to answer: DevOps is so diverse and inclusive that it rarely ends with coding, testing and systems. A DevOps project might rely on database platforms like SQL or NoSQL, data structure servers like Redis, or configuration and management issue tracking systems like Redmine. Web applications are popular for modern enterprises, making a background with Web servers, like Microsoft Internet Information Services, Apache Tomcat or other Web servers, beneficial. Make sure to bring across that you are familiar with Agile application lifecycle management techniques and tools.

**Are you familiar with just Linux or have you worked with Windows environments as well?**

Tips to answer: Demonstrate as much as you can, a clear understanding of both the environments including the key tools.

**How can you reduce load time of a dynamic website?**

Tips to answer: Talk about Webpage optimization, cached web pages, quality web hosting, compressed text files, Apache fine tuning.

**Describe your experience implementing continuous deployment?**

Tips to answer: Answer with a comprehensive list of all the tools that you used. Include inferences of the challenges you faced and how you tackled them.

**How would you ensure traceability?**

Tips to answer: This question probes your attitude to metrics, logging, transaction journeys, and reporting. You should be able to identify that metric, monitoring and logging needs to be a core part of the software system, and that without them, the software is essentially not going to be able to appear maintained and diagnosed. Include words like SysLog, Splunk, error tracking, Nagios, SCOM, Avicode in your answer.

**What was your greatest achievement on a recent project?**

Tips to answer: Make sure you demonstrate your perfect understanding of both development and operations. Do not let your answer lean towards one particular skillset ignoring the other. Even if you have worked in an environment wherein you had to work more with one skillset, assure the interviewer that you are agile according to the needs of your organization.

**What problems did you face and how did you solve them in a way that met the team’s goals?**

Tips to answer: This question aims to find out how much you can handle stress and non-conformity at work. Talk about your leadership skills to handle and motivate the team to solve problems together. Talk about CI, release management and other tools to keep interdisciplinary projects on track.

**Are you more Dev or Ops?**

Tips to answer: This is probably the trickiest question that you might face in the interview. Emphasize the fact that this depends a lot on the job, the company you are working for and the skills of people involved. You really have to be able to alternate between both sides of the fence at any given time. Talk about your experience and demonstrate how you are agile with both.

**What special training or education did it require for you to become a DevOps engineer?**

Tips to answer: DevOps is more of a mind-set or philosophy rather than a skill-set. The typical technical skills associated with DevOps Engineers today is Linux systems administration, scripting, and experience with one of the many continuous integration or configuration management tools like Jenkins and Chef. What it all boils down to is that whatever skill-sets you have, while important, are not as important as having the ability to learn new skills quickly to meet the needs. It’s all about pattern recognition, and having the ability to merge your experiences with current requirements. Proficiency in Windows and Linux systems administration, script development, an understanding of structured programming and object-oriented design, and experience creating and consuming RESTful APIs would take one a long way.

**Explain your understanding and expertise on both the software development side and the technical operations side of an organization you’ve worked for in the past.**

DevOps engineers almost always work in a 24/7 business critical online environment. I was adaptable to on-call duties and able to take up real-time, live-system responsibility. I successfully automated processes to support continuous software deployments. I have experience with public/private clouds, tools like Chef or Puppet, scripting and automation with tools like Python and PHP, and a background in Agile.

**Discuss your experience building bridges between IT Ops, QA and development.**

DevOps is all about effective communication and collaboration. I’ve been able to deal with production issues from the development and operations sides, effectively straddling the two worlds. I’m less interested in finding blame or playing the hero than I am with ensuring that all of the moving parts come together.

**What types of testing are needed?**

Software teams will often look for the “fair weather” path to system completion; that is, they start from an assumption that software will usually work and only occasionally fail. I believe to practice defensive programming in a pragmatic way, which often means assuming that the code will fail and planning for those failures. I try to incorporate unit test strategy, use of test harnesses, early load testing; network simulation, A/B and multi-variate testing  etc.

**Give me an example of how you would handle projects?**

As a professional with managerial responsibilities, I would demonstrate a clear understanding of DevOps project management tactics and also work with teams to set objectives, streamline workflow, maintain scope, research and introduce new tools or frameworks, translate requirements into workflow and follow up. I would resort to CI, release management and other tools to keep interdisciplinary projects on track.

**What’s your career objective in your role as a DevOps engineer?**

My passion is breaking down the barriers and building and improving processes, so that the engineering and operations teams work better and smarter. That’s why I love DevOps. It’s an opportunity to be involved in the entire delivery system from start to finish.

**How you would make software deployable?**

The ability to script the reconfiguration and installation of software systems is essential towards automated and controlled change.  Older products and systems are supposing that the changes would be minor and infrequent and so the automated changes become difficult, even though there is an increasing style for new software to enable this. In order to expose settings and configuration in a way accessible to automation, the professional has  to work with concepts such as  scripted installation, separation of concerns, infrastructure as a code, command – tools, test harnesses, dependency injection and inversion of control.

**How would you make software deployable?**

The ability to script the installation and reconfiguration of software systems is essential towards controlled and automated change. Although there is an increasing trend for new software to enable this, older systems and products suffer from the assumption that changes would be infrequent and minor, and so make automated changes difficult. As a professional who appreciates the need to expose configuration and settings in a manner accessible to automation, I will work with concepts like Inversion of Control (IoC) and Dependency Injection, scripted installation, test harnesses, separation of concerns, command-line tools, and infrastructure as code.

**What is the one most important thing DevOps helps do?**

The most important thing DevOps helps do is to get the changes into production as quickly as possible while minimizing risks in software quality assurance and compliance. That is the primary objective of DevOps. However, there are many other positive side-effects to DevOps. For example, clearer communication and better working relationships between teams which creates a less stressful working environment.

**Tell us about the CI tools that you are familiar with?**

The premise of CI is to get feedback as early as possible because the earlier you get feedback, the less things cost to fix. Popular open source tools include Hudson, Jenkins, CruiseControl and CruiseControl.NET. Commercial tools include ThoughtWorks’ Go, Urbancode’s Anthill Pro, Jetbrains’ Team City and Microsoft’s Team Foundation Server.

**Which scripting languages do you think are most important for a DevOps engineer?**

As far as scripting languages go, the simpler the better. In fact, the language itself isn’t as important as understanding design patterns and development paradigms such as procedural, object-oriented, or functional programming.

**How do you expect you would be required to multitask as a DevOps professional?**

I believe I’ll be expected to:

1. Focus attention on bridging communication gaps between Development and Operations teams.
2. Understand system design from an architect’s perspective, software development from a developer’s perspective, operations and infrastructure from the perspective of a seasoned Systems Administrator.
3. Execute – to be able to actually do what needs to be done.

**How Database fits in a DevOps ?**

In a perfect DevOps world, the DBA is an integral part of both Development and Operations teams and database changes should be as simple as code changes. So, you should be able to version and automate your Database scripts as your application code. In terms of choices between RDBMS, noSQL or other kind of storage solutions a good database design means less changes to your schema of Data and more efficient testing and service virtualization. Treating database management as an afterthought and not choosing the right database during early stages of the software development lifecycle can prevent successful adoption of the true DevOps movement.

**What testing is necessary to ensure that a new service is ready for production?**

DevOps is all about continuous testing throughout the process, starting with development through to production. Everyone shares the testing responsibility. This ensures that developers are delivering code that doesn’t have any errors and is of high quality, and it also helps everyone leverage their time most effectively.

**What is Continuous Integration and Why Is It Helpful?**

**Continuous integration** is a practice that encourages developers to integrate their code into a main branch of a shared repository early and often. Instead of building out features in isolation and integrating them at the end of a development cycle, code is integrated with the shared repository by each developer multiple times throughout the day.

The idea is to minimize the cost of integration by making it an early consideration. Developers can discover conflicts at the boundaries between new and existing code early, while conflicts are still relatively easy to reconcile. Once the conflict is resolved, work can continue with confidence that the new code honors the requirements of the existing codebase.

Integrating code frequently does not, by itself, offer any guarantees about the quality of the new code or functionality. In many organizations, integration is costly because manual processes are used to ensure that the code meets standards, does not introduce bugs, and does not break existing functionality. Frequent integration can create friction when the level of automation does not match the amount quality assurance measures in place.

To address this friction within the integration process, in practice, continuous integration relies on robust test suites and an automated system to run those tests. When a developer merges code into the main repository, automated processes kick off a build of the new code. Afterwards, test suites are run against the new build to check whether any integration problems were introduced. If either the build or the test phase fails, the team is alerted so that they can work to fix the build.

The end goal of continuous integration is to make integration a simple, repeatable process that is part of the everyday development workflow in order to reduce integration costs and respond to defects early. Working to make sure the system is robust, automated, and fast while cultivating a team culture that encourages frequent iteration and responsiveness to build issues is fundamental to the success of the strategy.

**What is Continuous Delivery and Why Is It Helpful?**

**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. By ensuring that the codebase is always in a deployable state, releasing software becomes an unremarkable event without complicated ritual. Teams can be confident that they can release whenever they need to without complex coordination or late-stage testing. As with continuous integration, continuous delivery is a practice that requires a mixture of technical and organizational improvements to be effective.

On the technology side, continuous delivery leans heavily on deployment pipelines to automate the testing and deployment processes. A **deployment pipeline** is an automated system that runs increasingly rigorous test suites against a build as a series of sequential stages. This picks up where continuous integration leaves off, so a reliable continuous integration setup is a prerequisite to implementing continuous delivery.

At each stage, the build either fails the tests, which alerts the team, or passes the tests, which results in automatic promotion to the next stage. As the build moves through the pipeline, later stages deploy the build to environments that mirror the production environment as closely as possible. This way the build, the deployment process, and the environment can be tested in tandem. The pipeline ends with a build that can be deployed to production at any time in a single step.

The organizational aspects of continuous delivery encourage prioritization of "deployability" as a principle concern. This has an impact on the way that features are built and hooked into the rest of the codebase. Thought must be put into the design of the code so that features can be safely deployed to production at any time, even when incomplete. A [number of techniques](https://www.digitalocean.com/community/tutorials/an-introduction-to-continuous-integration-delivery-and-deployment#additional-terminology) have emerged to assist in this area.

Continuous delivery is attractive because it automates the steps between checking code into the repository and deciding on whether to release well-tested, functional builds to your production infrastructure. The steps that help assert the quality and correctness of the code are automated, but the final decision about what to release is left in the hands of the organization for maximum flexibility.

**What is Continuous Deployment and Why Is It Helpful?**

**Continuous deployment** is an extension of continuous delivery that automatically deploys each build that passes the full test cycle. Instead of waiting for a human gatekeeper to decide what and when to deploy to production, a continuous deployment system deploys everything that has successfully traversed the deployment pipeline. Keep in mind that while new code is automatically *deployed*, techniques exist to activate new features at a later time or for a subset of users. Deploying automatically pushes features and fixes to customers quickly, encourages smaller changes with limited scope, and helps avoid confusion over what is currently deployed to production.

This fully automated deploy cycle can be a source of anxiety for organizations worried about relinquishing control to their automation system of what gets released. The trade-off offered by automated deployments is sometimes judged to be too dangerous for the payoff they provide.

Other groups leverage the promise of automatic release as a method of ensuring that best practices are always followed and to extend the testing process into a limited production environment. Without a final manual verification before deploying a piece of code, developers must take responsibility for ensuring that their code is well-designed and that the test suites are up-to-date. This collapses the decision of what and when to commit to the main repository and what and when to release to production into a single point that exists firmly in the hands of the development team.

Continuous deployment also allows organizations to benefit from consistent early feedback. Features can immediately be made available to users and defects or unhelpful implementations can be caught early before the team devotes extensive effort in an unproductive direction. Getting fast feedback that a feature isn't helpful lets the team shift focus rather than sinking more energy into an area with minimal impact.

**Key Concepts and Practices for Continuous Processes**

While continuous integration, delivery, and deployment vary in the scope of their involvement, there are some concepts and practices that are fundamental to the success of each.

**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. Special techniques like branch by abstraction and feature flags (see below) help to protect the functionality of the main branch from in-progress code changes.

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.

The idea behind trunk-based development is to avoid large commits that violate of concept of small, iterative changes discussed above. Code is available to peers early so that conflicts can be resolved when their scope is small.

Releases are performed from the main branch or from a release branch created from the trunk specifically for that purpose. No development occurs on the release branches in order to maintain focus on the main branch as the single source of truth.

**Keep the Building and Testing Phases Fast**

Each of the processes relies on automated building and testing to validate correctness. Because the build and test steps must be performed frequently, it is essential that these processes be streamlined to minimize the time spent on these steps.

Increases in build time should be treated as a major problem because the impact is compounded by the fact that each commit kicks off a build. Because continuous processes force developers to engage with these activities daily, reducing friction in these areas is a worthwhile pursuit.

When possible, running different sections of the test suite in parallel can help move the build through the pipeline faster. Care should also be taken to make sure the proportion of each *type* of test makes sense. Unit tests are typically very fast and have minimal maintenance overhead. In contrast, automated system or acceptance testing is often complex and prone to breakage. To account for this, it is often a good idea to rely heavily on unit tests, conduct a fair number of integration tests, and then back off on the number of later, more complex testing.

**Consistency Throughout the Deployment Pipeline**

Because a continuous delivery or deployment implementations is supposed to be testing release worthiness, it is essential to maintain consistency during each step of the process—the build itself, the deployment environments, and the deployment process itself:

* **Code should be built once at the beginning of the pipeline**: The resulting software should be stored and accessible to later processes without rebuilding. By using the exact same artifact in each phase, you can be certain that you are not introducing inconsistencies as a result of different build tools.
* **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. Clean deployment environments should be provisioned each test cycle to prevent legacy conditions from compromising the integrity of the tests. The staging environments should match the production environment as closely as possible to reduce unknown factors present when the build is promoted.
* **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. Ad-hoc deployments should be eliminated in favor of deploying only with the pipeline tools.

**Decouple Deployment and Release**

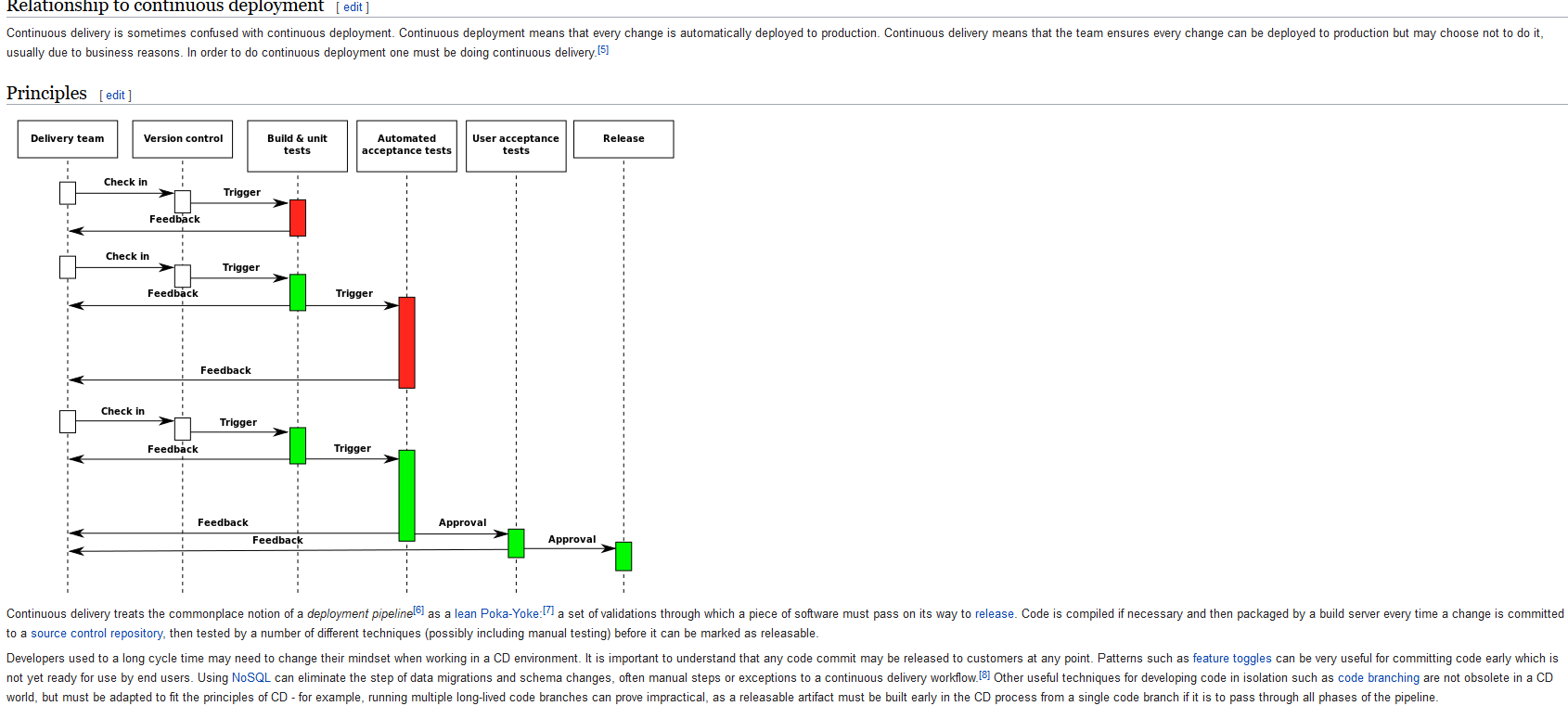
Separating the deployment of code from its release to users is an extremely powerful part of continuous delivery and deployment. Code can be deployed to production without initially activating it or making it accessible to users. Then, the organization decides when to release new functionality or features independent from deployment.

This gives organizations a great deal of flexibility by separating business decisions from technical processes. If the code is already on the servers, then deployment is no longer a delicate part of the release process, which minimizes the number of individuals and the amount of work involved at the time of release.

There are a number of techniques that help teams deploy the code responsible for a feature without releasing it. Feature flags set up conditional logic to check whether to run code based on the value of an environmental variable. Branch by abstraction allows developers to replace implementations by placing an abstraction layer between resource consumers and providers. Careful planning to incorporate these techniques gives you the ability to decouple these two processes.

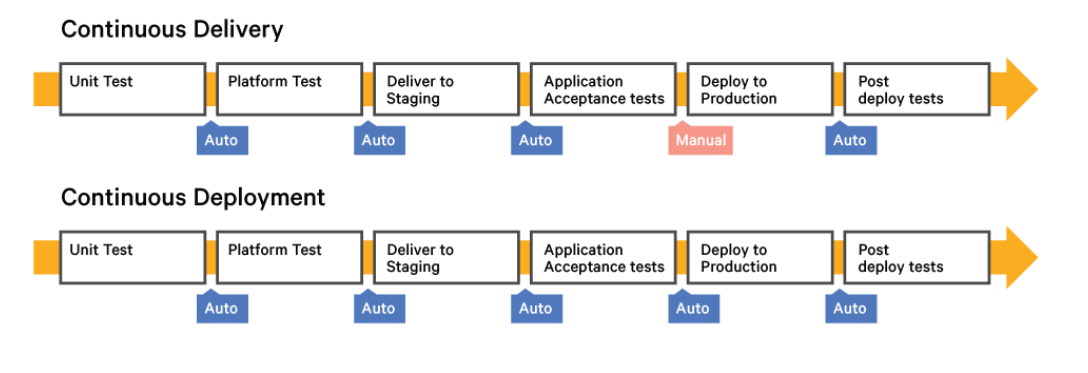
**Difference between continuous delivery and deployment**

Continuous Delivery is where you build software in a way that the software can be deployed to production at any time. Whether it will be delivered is determined by a stakeholder group (often product management or business stakeholders).



Continuous Deployment requires organizations to be practicing continuous delivery. It means that every version of the software is deployed to production without a specific deployment decision being made.

Continuous deployment is a strategy for software releases wherein any commit that passes the automated testing phase is automatically released into the production deployment.



**Additional Terminology**

While we've discussed some of the broader ideas above, there are many related concepts that you may come across as you learn about continuous integration, delivery, and deployment. Let's define a few other terms you are likely to see:

* **Blue-Green Deployments**: Blue-green deployments is a strategy for testing code in a production-like environment and for deploying code with minimal downtime. Two sets of production-capable environments are maintained, and code is deployed to the inactive set where testing can take place. When ready to release, production traffic is routed to the servers with the new code, instantly making the changes available.
* **Branch by Abstraction**: Branch by abstraction is a method of performing major refactoring operations in an active project without long-lived development branches in the source code repository, which continuous integration practices discourage. An abstraction layer is built and deployed between consumers and the existing implementation so that the new implementation can be built out behind the abstraction in parallel.
* **Build (noun)**: A build is a specific version of software created from source code. Depending on the language, this might be compiled code or a consistent set of interpreted code.
* **Canary Releases**: Canary releases are a strategy for releasing changes to a limited subset of users. The idea is to make sure everything works correctly with production workloads while minimizing the impact if there are problems.
* **Dark launch**: Dark launching is the practice of deploying code to production that receives production traffic but does not impact the user experience. New changes are deployed alongside existing implementations and the same traffic is often routed to both places for testing. The old implementation is still hooked up to the user's interface, but behind the scenes, the new code can be evaluated for correctness using real user requests in the production environment.
* **Deployment Pipeline**: A deployment pipeline is a set of components that moves software through increasingly rigorous testing and deployment scenarios to evaluate its readiness for release. The pipeline typically ends by automatically deploying to production or providing the option to do so manually.
* **Feature Flags** or **Feature Toggles**: Feature flags are a technique of deploying new features behind conditional logic that determines whether or not to run based on the value of an environmental variable. New code can be deployed to production without being activated by setting the flag appropriately. To release the software, the value of the environmental variable is changed, causing the new code path to be activated. Feature flags often contain logic that allows for subsets of users to gain access to the new feature, creating a mechanism to gradually roll out the new code.
* **Promoting**: In the context of continuous processes, promoting means moving a software build through to the next stage of testing.
* **Soak Test**: Soak testing involves testing software under significant production or production-like load for an extended period of time.

**What is post mortem meetings?**

* It is a meeting where we discuss what went wrong and what steps should be taken so that failure doesn't happen again. Post mortem meetings are not about finding the one to be blamed, they are for preventing outages from reoccurring and planing redesign of the infrastructure so that downtime can be minimised. It is about learning from mistakes.

**Domain Name Servers (DNS):**

* These are the Internet's equivalent of a phone book.
* They maintain a directory of domain names and translate them to Internet Protocol (IP) addresses.0
* This is necessary because, although domain names are easy for people to remember, computers or machines, access websites based on IP addresses.
* Information from all the domain name servers across the Internet are gathered together and housed at the Central Registry. Host companies and Internet Service Providers interact with the Central Registry on a regular schedule to get updated DNS information.
* When you type in a web address, e.g., www.jimsbikes.com, your Internet Service Provider views the DNS associated with the domain name, translates it into a machine friendly IP address (for example 216.168.224.70 is the IP for jimsbikes.com) and directs your Internet connection to the correct website.
* After you register a new domain name or when you update the DNS servers on your domain name, it usually takes about 12-36 hours for the domain name servers world-wide to be updated and able to access the information. This 36-hour period is referred to as propagation.

**DHCP:**

* Dynamic Host Configuration Protocol (DHCP) is a client/server protocol that automatically provides an Internet Protocol (IP) host with its IP address and other related configuration information such as the subnet mask and default gateway.
* RFCs (Remote Function Call) 2131 and 2132 define DHCP as an Internet Engineering Task Force (IETF) standard based on Bootstrap Protocol (BOOTP), a protocol with which DHCP shares many implementation details.
* DHCP allows hosts to obtain required TCP/IP configuration information from a DHCP server.

**Why use DHCP?**

* Every device on a TCP/IP-based network must have a unique unicast IP address to access the network and its resources. Without DHCP, IP addresses for new computers or computers that are moved from one subnet to another must be configured manually; IP addresses for computers that are removed from the network must be manually reclaimed.
* With DHCP, this entire process is automated and managed centrally. The DHCP server maintains a pool of IP addresses and leases an address to any DHCP-enabled client when it starts up on the network.
* Because the IP addresses are dynamic (leased) rather than static (permanently assigned), addresses no longer in use are automatically returned to the pool for reallocation.
* The network administrator establishes DHCP servers that maintain TCP/IP configuration information and provide address configuration to DHCP-enabled clients in the form of a lease offer.

**The DHCP server stores the configuration information in a database that includes:**

* Valid TCP/IP configuration parameters for all clients on the network.
* Valid IP addresses, maintained in a pool for assignment to clients, as well as excluded addresses.
* Reserved IP addresses associated with particular DHCP clients. This allows consistent assignment of a single IP address to a single DHCP client.
* The lease duration, or the length of time for which the IP address can be used before a lease renewal is required.

**A DHCP-enabled client, upon accepting a lease offer, receives:**

* A valid IP address for the subnet to which it is connecting.
* Requested DHCP options, which are additional parameters that a DHCP server is configured to assign to clients. Some examples of DHCP options are Router (default gateway), DNS Servers, and DNS Domain Name. For a full list of DHCP options, see DHCP Tools and Options.

**Benefits of DHCP:**

* In Windows Server 2008, the DHCP Server service provides the following benefits:
* Reliable IP address configuration. DHCP minimizes configuration errors caused by manual IP address configuration, such as typographical errors, or address conflicts caused by the assignment of an IP address to more than one computer at the same time.
* Reduced network administration.

**DHCP includes the following features to reduce network administration:**

* Centralized and automated TCP/IP configuration.
* The ability to define TCP/IP configurations from a central location.
* The ability to assign a full range of additional TCP/IP configuration values by means of DHCP options.
* The efficient handling of IP address changes for clients that must be updated frequently, such as those for portable computers that move to various locations on a wireless network.
* The forwarding of initial DHCP messages by using a DHCP relay agent, which eliminates the need for a DHCP server on every subnet.

**TCP/IP**

**What is TCP/IP and where did it come from?**

TCP/IP stands for “Transmission Control Protocol / Internet Protocol”. It is basically a network protocol that defines the details of how data is sent and received through network adapters, hubs, switches, routers and other network communications hardware. It was developed by the US department of defense for the purpose of connecting government computer systems to each other through a global, fault tolerant, network. The defense department network was opened up to research institutions and eventually the general public to create what is now the Internet. The TCP/IP protocol was also placed in the public domain so that any software company could develop networking software based on the protocol. Because it is the primary protocol used on the Internet, and it is in the public domain, it has become the most popular networking protocol throughout the world and is therefore well supported by almost all computer systems and networking hardware.

TCP/IP, or the Transmission Control Protocol/Internet Protocol, is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP can also be used as a communications protocol in a private network (an intranet or an extranet).

**How does TCP/IP work - without going into too much detail?**

The TCP/IP protocol is designed such that each computer or device in a network has a unique "IP Address" (Internet Protocol Address) and each IP address can open and communicate over up to 65535 different “ports” for sending and receiving data to or from any other network device. The IP Address uniquely identifies the computer or device on the network and a "Port Number" identifies a specific connection between one computer or device and another (i.e., between two IP Addresses). A TCP/IP "port" can be thought of as a private two-way communications line where the port number is used to identify a unique connection between two devices. The concept is similar to any other type of port on your PC (serial, parallel, etc.) except that instead of having a physical connection, the TCP/IP protocol creates a "virtual IP port" and the network hardware and software is responsible for routing data in and out of each virtual IP port.

**TCP/IP Client and Server Connections**

TCP/IP connections work in a manner similar to a telephone call where someone should initiate the connection by dialing the phone. At the other end of the connection, someone has to be listening for calls and then pick up the line when a call comes in. In TCP/IP communications, the IP Address is analogous to a telephone number and the port number would be analogous to a particular extension once the call has been answered. The “Client” in a TCP/IP connection is the computer or device that “dials the phone” and the “Server” is the computer that is “listening” for calls to come in. In other words, the Client needs to know the IP Address of whatever Server it wants to connect to and it also needs to know the port number that it wants to send and receive data through after a connection has been established. The Server only has to listen for connections and either accept them or reject them when they are initiated by a client.

Once a connection through a TCP/IP port has been established between a TCP/IP client and a TCP/IP server, data can be sent in either direction exactly the same way that data is sent through any other type of port on a PC (serial, parallel, etc.). The only difference is that the data is sent across your network. The connection between a Client and a Server remains open until either the client or the server terminates the connection (i.e. hangs up the phone). One extremely nice benefit of the TCP/IP protocol is that the low-level drivers that implement the sending and receiving of data perform error checking on all data so you are guaranteed that there will be no errors in any data that you send or receive.

**HTTP**

Short for Hyper Text Transfer Protocol, HTTP is a set of standards that allow users of the World Wide Web to exchange information found on web pages. When accessing any web page entering http:// in front of the address tells the browser to communicate over HTTP. For example, the URL for Computer Hope is https://www.computerhope.com. Today's browsers no longer require HTTP in front of the URL since it is the default method of communication. However, it is kept in browsers because of the need to separate protocols such as FTP. Below are a few of the major facts on HTTP.

The term HTTP was coined by Ted Nelson.

* The standard port for HTTP connections is port 80.
* HTTP/0.9 was the first version of the HTTP, and was introduced in 1991.
* HTTP/1.0 is specified in RFC 1945, and was introduced in 1996.
* HTTP/1.1 is specified in RFC 2616, and was officially released in January 1997.

**HTTPS**

Short for Hypertext Transfer Protocol Secure, HTTPS is a protocol which uses HTTP on a connection encrypted by transport-layer security. HTTPS is used to protect transmitted data from eavesdropping. It is the default protocol for conducting financial transactions on the web, and can protect a website's users from censorship by a government or an ISP.

* HTTPS uses port 443 to transfer its information.
* HTTPS is first used in HTTP/1.1 and is defined in RFC 2616.

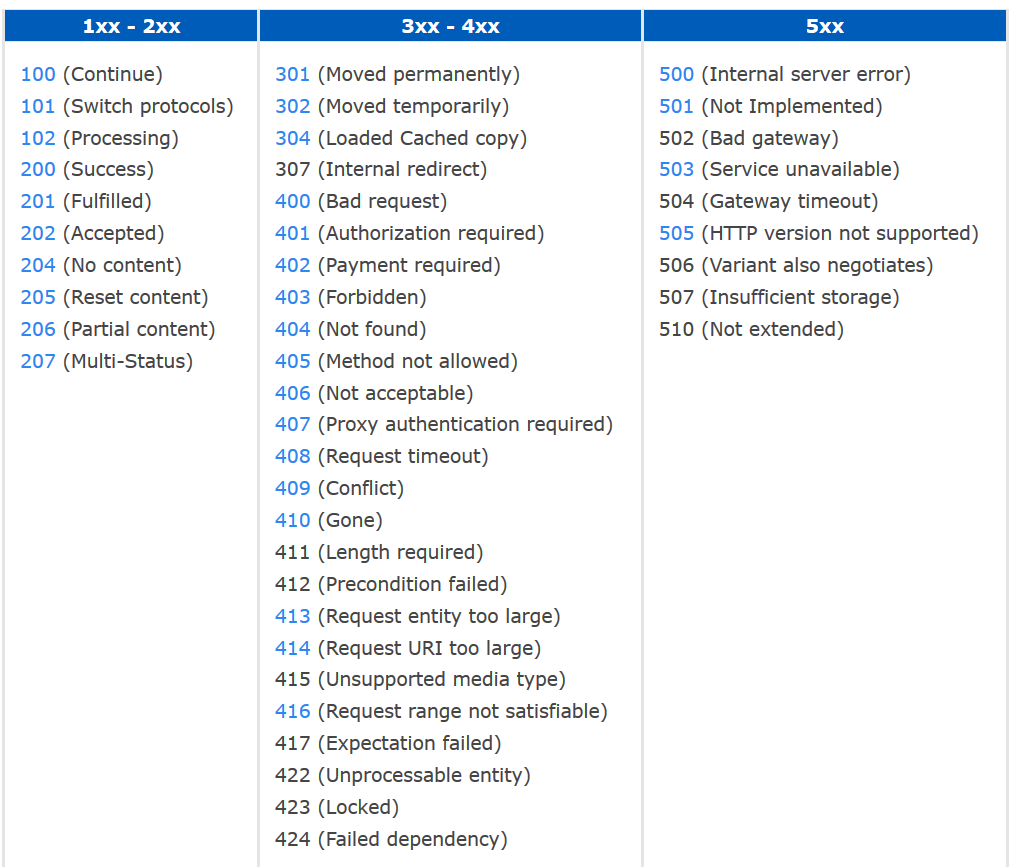
If you see https, the session between the web server and the browser on the mobile device you are using is encrypted. You can easily identify web servers that have https configured by looking at the Uniform Resource Locator (URL) in the web address bar of your browser.

Hypertext Transfer Protocol Secure (https) is a combination of the Hypertext Transfer Protocol (HTTP) with the Secure Socket Layer (SSL)/Transport Layer Security (TLS) protocol. TLS is an authentication and security protocol widely implemented in browsers and Web servers. SSL works by using a public key to encrypt data transferred over the SSL connection. Most Web browsers support SSL. It allows you to communicate securely with the web server.

**How does the internet works?**

<https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm>

**HTTP Status Codes:**



**For more information**

<https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol>

**Web security mechanisms**

Web server security is the protection of information assets that can be accessed from a Web server.

Web server security is important for any organization that has a physical or virtual Web server connected to the Internet. It requires a layered defense and is especially important for organizations with customer-facing websites.

Separate servers should be used for internal and external-facing applications and servers for external-facing applications should be hosted on a DMZ or containerized service network to prevent an attacker from exploiting a vulnerability to gain access to sensitive internal information.

Penetration tests should be run on a regular basis to identify potential attack vectors, which are often caused by out-of-date server modules, configuration or coding errors and poor patch management. Web site security logs should be audited on a continuous basis and stored in a secure location. Other best practices include using a separate development server for testing and debugging, limiting the number of superuser and administrator accounts and deploying an intrusion detection system (IDS) that includes monitoring and analysis of user and system activities, the recognition of patterns typical of attacks, and the analysis of abnormal activity patterns.

For more information:

<https://www.slideshare.net/bharaths7/web-security>

<https://docs.oracle.com/cd/E17802_01/webservices/webservices/reference/tutorials/wsit/doc/WSIT_Security4.html>

**Proxies**

<https://en.wikipedia.org/wiki/Proxy_server>

<http://whatis.techtarget.com/definition/proxy-server>

**FORWARD VS REVERSE PROXY**

The main purpose of a proxy service (which is the kind of service either of these two provide) is very similar to what a person aims to achieve when he proxies for another person. That is, to act on behalf of that other person. In our case, a proxy server acts on behalf of another machine - either a client or another server.

When people talk about a proxy server (often simply known as a "proxy"), more often than not they are referring to a forward proxy.

A forward proxy provides proxy services to a client or a group of clients. Oftentimes, these clients belong to a common internal network like the one shown below.

A reverse proxy does the exact opposite of what a forward proxy does. While a forward proxy proxies in behalf of clients (or requesting hosts), a reverse proxy proxies in behalf of servers. A reverse proxy accepts requests from external clients on behalf of servers stationed behind it.

**Firewalls**

A firewall is a network security device that monitors incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules.

Firewalls have been a first line of defense in network security for over 25 years. They establish a barrier between secured and controlled internal networks that can be trusted and untrusted outside networks, such as the Internet.

A firewall can be hardware, software, or both.

### Proxy firewall

An early type of firewall device, a proxy firewall serves as the gateway from one network to another for a specific application. Proxy servers can provide additional functionality such as content caching and security by preventing direct connections from outside the network. However, this also may impact throughput capabilities and the applications they can support.

**Stateful inspection firewall**

Now thought of as a “traditional” firewall, a stateful inspection firewall allows or blocks traffic based on state, port, and protocol. It monitors all activity from the opening of a connection until it is closed. Filtering decisions are made based on both administrator-defined rules as well as context, which refers to using information from previous connections and packets belonging to the same connection

### **Unified threat management (UTM) firewall**

A UTM device typically combines, in a loosely coupled way, the functions of a stateful inspection firewall with intrusion prevention and antivirus. It may also include additional services and often cloud management. UTMs focus on simplicity and ease of use.

### **Next-generation firewall (NGFW)**

Firewalls have evolved beyond simple packet filtering and stateful inspection. Most companies are deploying next-generation firewalls to block modern threats such as advanced malware and application-layer attacks.

According to Gartner, Inc.’s definition, a next-generation firewall must include:

* Standard firewall capabilities like stateful inspection
* Integrated intrusion prevention
* Application awareness and control to see and block risky apps
* Upgrade paths to include future information feeds
* Techniques to address evolving security threats

While these capabilities are increasingly becoming the standard for most companies, NGFWs can do more.

**Threat-focused NGFW**

These firewalls include all the capabilities of a traditional NGFW and also provide advanced threat detection and remediation. With a threat-focused NGFW you can:

* Know which assets are most at risk with complete context awareness
* Quickly react to attacks with intelligent security automation that sets policies and hardens your defenses dynamically
* Better detect evasive or suspicious activity with network and endpoint event correlation
* Greatly decrease the time from detection to cleanup with retrospective security that continuously monitors for suspicious activity and behavior even after initial inspection
* Ease administration and reduce complexity with unified policies that protect across the entire attack continuum

<http://searchsecurity.techtarget.com/definition/firewall>

<https://en.wikipedia.org/wiki/Firewall_(computing)>

**What is REST?**

REST stands for Representational State Transfer. REST is a web standard based architecture and uses HTTP Protocol for data communication. It revolves around resources where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in year 2000.

In REST architecture, a REST Server simply provides access to resources and the REST client accesses and presents the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource like Text, JSON and XML. JSON is now the most popular format being used in Web Services.

**API transaction scripts:** with details like endpoint, method

**HTTP METHODS**

The following HTTP methods are most commonly used in a REST based architecture.

• **GET** − Provides a read only access to a resource.

• **PUT** − Used to create a new resource.

• **DELETE** − Used to remove a resource.

• **POST** − Used to update an existing resource or create a new resource.

• **OPTIONS** − Used to get the supported operations on a resource.

**WHEN TO USE SOAP**

SOAP is most appropriately used for large **enterprise applications** rather than smaller, more mobile applications. SOAP architecture is most useful when a formal contract must be established to describe the interface that the web service offers, such as details regarding messages, operations, bindings, and the location of the web service. Therefore, SOAP should be used when more capability is needed. For example, including up-to-date stock prices to subscribing websites is a good time to use SOAP since a greater amount of program interaction between client and server is required than REST can provide.

**WHEN TO USE REST**

REST is implemented most easily using **ASP.NET** web API in MVC 4.0. REST is most appropriately used for smaller, more **mobile applications**, rather than large, enterprise applications. This is because REST is best used as a means of publishing information, components, and processes to make them more accessible to other users and machine processes. An online publisher could use REST to make syndicated content available by periodically preparing and activating a web page that included content and XML statements that described the content.

Overall, if the project requires a high level of security and a large amount of data exchange, then SOAP is the appropriate choice. But if there are resource constraints and you need the code to be written faster, then REST is better. Ultimately it depends on the project and the circumstances which architecture fits best.

**APACHE WEB SERVER**

**Troubleshooting Apache Webserver (server not starting/restarting):**

1. Check for config syntax error using httpd -t/-s which returns saying SYNTAX OK or SYNTAX ERROR AT LINE SO AND SO.

2. Check Apache error log file tail -f /var/log/httpd-error.log

3. Check that your servername is set correctly in httpd.conf

4. Check log files over 2GB because they can cause problem or error 500, so make sure that log files are under limit by moving or removing them out of log directories through log rotation.

5. Check the availability of port 80 and 443

**Directory structure of Apache HTTP server:**

• Bin - executables

• Conf

• Error

• Icons

• Include

• Lib

• Logs

Modules

**TOMCAT**

**Running multiple web applications in one Tomcat Server:**

Go to server.xml in conf dir and add a new service tag with details for second application like: port, service name (webapps2), Engine name (webapps2), appbase (webapps2) etc.

Next is create a directory in the name webapps2 in order to accommodate the second application.

**Benefits of Tomcat over other servers:**

• Light weight.

• Widely Used.

• Much faster than other containers.

• Easy to configure and Very flexible

**How does HTTP work?**

The HTTP protocol works in a client and server model like most other protocols. A web browser using which a request is initiated is called as a client and a web server software which responds to that request is called a server. World Wide Web Consortium and the Internet Engineering Task Force are two important spokes in the standardization of the HTTP protocol. HTTP allows improvement of its request and response with the help of intermediates, for example a gateway, a proxy, or a tunnel. The resources that can be requested using the HTTP protocol, are made available using a certain type of URI (Uniform Resource Identifier) called a URL (Uniform Resource Locator). TCP (Transmission Control Protocol) is used to establish a connection to the application layer port 80 used by HTTP.

**Explain the working of HTTP?**

Like other protocols,  HTTP also works on the client-server model. A web server software which responds  to the request is called a server and a web browser which initiates the request is called a client. HTTP enhances its request and response with the help of intermediates such as tunnel, proxy or gateway. URL helps in allocating the resources, that are requested using HTTP. The connection to the application layer port of HTTP is provided by TCP.

**How you would prepare for migration?**

This question shows your experience in real projects, which brings complexity, awkwardness and include other things like roll back and forward, feature toggles, cut over, DNS solutions,  branch by abstraction, dress rehearsals and automation in your response. Dealing  with legacy configuration and components  are always  tougher  than developing a  greenfield system with no existing technology.

**What’s a PTR in DNS?**

Pointer records are used to map a network interface (IP) to a host name. These are primarily used for reverse DNS. Reverse DNS is setup very similar to how normal (forward) DNS is setup.  When you delegate the DNS forward, the owner of the domain tells the registrar to let your domain use specific name servers.

**Describe two-factor authentication?**

Two-factor authentication is a security process in which the user provides two means of identification from separate categories of credentials; one is typically a physical token, such as a card, and the other is typically something memorized, such as a security code.

**What are the advantages of NoSQL database over RDBMS?**

The advantages are:

1. Less need for ETL
2. Support for unstructured text
3. Ability to handle change over time
4. Breadth of functionality
5. Ability to scale horizontally
6. Support for multiple data structures
7. Choice of vendors

**Do you find any advantage of using NoSQL database over RDBMS?**

Typical web applications are built with a three-tier architecture. To carry the load, more Web servers are simply added behind a load balancer to support more users. The ability to scale out is a key principle in the world of cloud computing, more and more important in which VM instances can be easily added or removed to meet demand.

However, when it comes to the data layer, relational databases (RDBMS) does not allow a passage to the simple scale and do not provide a flexible data model. Manage more users means adding more servers and large servers are very complex, owners and disproportionately expensive, in contrast to low-cost hardware, the "commodity hardware", architectures in the cloud. Organizations are beginning to see performance issues with their relational databases for existing or new applications. Especially as the number of users increases, they realize the need for a faster and more flexible basis. This is the time to begin to assess and adopt NoSQL database like in their Web applications.

**What are the main SQL migration difficulties NoSQL?**

Each record in a relational database according to a schema - with a fixed number of fields (columns) each having a specified object and a data type. Each record is the same. The data is denormalized in several tables. The advantage is that there is less of duplicate data in the database. The downside is that a change in the pattern means performing several "alter table" that require expensive to lock multiple tables simultaneously to ensure that change does not leave the database in an inconsistent state.

With databases data, on the other hand, each document can have a completely different structure from other documents. No additional management is required on the database to manage changes in the schemes.

**What are the benefits of NoSQL databases Documents?**

The main advantages of document databases are the following:

* flexible data model data can be inserted without a defined schema and format of the data that is inserted can change at any time, providing extreme flexibility, which ultimately allows a significant agility to business
* Consistent, high-performance Advanced NoSQL database technologies are putting cache data, transparently, in system memory; a behavior that is completely transparent to the developer and the team in charge of operations.
* Some easy scalability NoSQL databases automatically propagate data between servers, requiring no participation applications. Servers can be added and removed without disruption to applications, with data and I/O spread across multiple servers.

**What is an MX record in DNS?**

MX records are nothing but the mail exchange records which is used to determine the priority of email servers for a domain. The priority of email servers is divided into two categories,  lowest priority email servers and higher priority email servers. The first destination for email is known as the lowest priority email servers, the mail will be sent to the highest  priority email servers if the lowest priority email server is unavailable.

**What is an MX record in DNS?**

MX records are mail exchange records used for determining the priority of email servers for a domain. The lowest priority email server is the first destination for email. If the lowest priority email server is unavailable, mail will be sent to the higher priority email servers.

**What is the difference between RAID 0 and RAID 1?**

RAID 1 offers redundancy through mirroring, i.e., data is written identically to two drives. RAID 0 offers no redundancy and instead uses striping, i.e., data is split across all the drives. This means RAID 0 offers no fault tolerance; if any of the constituent drives fails, the RAID unit fails.

**How would you prepare for a migration?**

Tips to answer: This question evaluates your experience of real projects with all the awkwardness and complexity they bring. Include terms like cut-over, dress rehearsals, roll-back and roll-forward, DNS solutions, feature toggles, branch by abstraction, and automation in your answer. Developing greenfield systems with little or no existing technology in place is always easier than having to deal with legacy components and configuration. As a candidate if you appreciate that any interesting software system will in effect be under constant migration, you will appear suitable for the role.

**JAR, WAR & EAR:**

In J2EE application, modules are packaged as EAR, JAR and WAR based on their functionality

JAR: EJB modules which contain enterprise java beans (class files) and EJB deployment descriptor are packed as JAR files with .jar extension

WAR: Web modules which contain Servlet class files, JSP Files, supporting files, GIF and HTML files are packaged as JAR file with .war (web archive) extension

EAR: All above files (.jar and .war) are packaged as JAR file with .ear (enterprise archive) extension and deployed into Application Server

**WHY MULTIPLE JVMS**

Admins use multiple JVMs primarily to solve the following issues:

1. Garbage collection inefficiencies

2. Resource utilization

3. 64-bit issues

4. Availability

**STATEFUL VS STATELESS SERVICES**

Stateless web services do not maintain a session between requests. An example of this would be sites like search engines which just take your request, process it, and spit back some data.

Stateful web services maintain a session during your entire browsing experience. An example of this would be logging into your bank's website or your web based email like Gmail.

**What will happen when you type aaa.com?**

* The browser extracts the domain name from the URL
* The browser queries the DNS for the IP address of the URL.
* The request sent to the DNS server is smaller than the maximum packet size, and is sent as a single packet.
* This packet also includes IP address too. This packet reaches each piece of the network equipment between the client and server, that equipment uses a routing table to figure out what node it is connected to, to be part of the fastest route to the destination.
* If that DNS server has the address of the domain then it will return it, otherwise it will forward the query along the DNS server. This happens recursively as long as the request is fulfilled.
* Assuming the DNS request is successful, the client machine now has an IP address that uniquely identifies a machine on the Internet. The web browser then assembles an HTTP request, which consists of a header and optional content. The header includes HTTP version, any relevant browser cookies etc.

**What is SSL certificate?**

When it is installed on a web server it activates the padlock and the https protocol and then make a secure connection from the web server to the browser. Mostly SSL is used to secure the credit card transactions. Usually SSL binds the domain name, server name, company’s name together. Every organization must install the SSL on its web server to securely connect to the browser.

**How to create a SSL?**

1. Generate a private key- openssl tool kit is used to generate private key and CSR. This private key is 1024bit key and is stored in pem format.

**openssl genrsa -des3 -out server.key 1024**

1. Generate a CSR- Generally this CSR is sent to Certificate Authority, who will verify the identity of the requestor and issues a certificate.

**openssl req -new -key server.key -out server.csr**

1. Remove passphrase from key- Important reason for the removal of passphrase is APACHE will ask for the passphrase every time you start the webserver.

**cp server.key server.key.org  
openssl rsa -in server.key.org -out server.key**

1. Generating a self-signed certificate- The below command creates a SSL certificate which is temporary and good for 365 days

**openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt**

1. Installing the private key and certificate-

**cp server.crt /usr/local/apache/conf/ssl.crt  
cp server.key /usr/local/apache/conf/ssl.key**

1. Configuring SSL enabled virtual hosts
2. Restart apache and Test

**Samba Server:**

* Samba is a free software re-implementation of the SMB/CIFS networking protocol, and was originally developed by Andrew Tridgell.
* Samba provides file and print services for various Microsoft Windows clients and can integrate with a Microsoft Windows Server domain, either as a Domain Controller (DC) or as a domain member.
* As of version 4, it supports Active Directory and Microsoft Windows NT domains.
* Samba runs on most Unix, OpenVMS and Unix-like systems, such as Linux, Solaris, AIX and the BSD variants, including Apple's macOS Server, and macOS client (Mac OS X 10.2 and greater).
* Samba is standard on nearly all distributions of Linux and is commonly included as a basic system service on other Unix-based operating systems as well.
* Samba is released under the terms of the GNU General Public License.
* The name Samba comes from SMB (Server Message Block), the name of the standard protocol used by the Microsoft Windows network file system.

**Other References:**

https://github.com/spikenode/DevOps-Interview-Questions#general-questions