DevOps Essentials Interview Questions

1. What is the one most important thing DevOps helps do?

Answer: 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.

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

Answer: 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.

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

Answer: I believe I'll be expected to:

- Focus attention on bridging communication gaps between Development and Operations teams.
- 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.
- Execute - to be able to actually do what needs to be done.

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

Answer: 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.

5. What's your systems background?

Answer: 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 micro services 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.

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

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

7. Are you more Dev or Ops?

Answer: 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.

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

Answer: Tips to answer: DevOps is more of a mind-set or philosophy rather than a skill-set. The technical skills of a 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 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 OOD, and experience creating and consuming RESTful APIs would take one a long way.

9. What is DevOps life cycle?

Answer: Tips to answer: An Answer similar to the following is acceptable

- Check in code
- Pull code changes for build

- Run tests (CI server to generate builds and arrange releases): Test individual models, run integration tests, and run user acceptance tests.
- Store artifacts and build repository (repository for storing artifacts, results, and releases)
- Deploy and release (release automation product to deploy apps)
- Configure environment
- Update databases and apps
- Push to users – who receive tested app updates frequently and without interruption
- Application and Network Performance Monitoring (preventive safeguard)
- Repeat

The Acronyms CALMS is sometimes used:

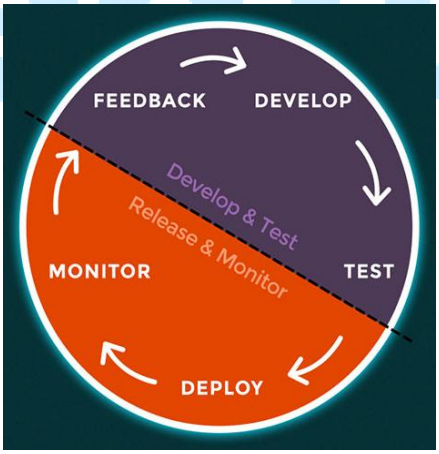
C: Culture

A: Automation

L: Lean

M: Measurement

S: Sharing



10. What is the difference between continuous integration, continuous delivery and continuous deployment?

Answer: What is CI? It is an integration of code into a known or working code base. One could argue that a deployment into an environment is an integration of code. The term CI ideally is limited to only source control and to provide a clear definition that teams can use to simply identify the practices and actions.

What is CD? In short, it is an automated process to deliver a software package to an environment. In that short definition, there is a number of tools, techniques, and workflows that make up the CD process.

Deployment does not imply release. You can continuously deploy to UAT. What makes continuous deployment special is deploying every change that passes the automated tests (and optionally a short QA gate) to production. Continuous deployment is the practice of releasing every good build to users - a more accurate name might have been “continuous release”

While continuous deployment implies continuous delivery the converse is not true. Continuous delivery is about putting the release schedule in the hands of the business, not in the hands of IT. Implementing continuous delivery means making sure your software is always production ready throughout its entire lifecycle – that any build could potentially be released to users at the touch of a button using a fully automated process in a matter of seconds or minutes.

11. What is DevOps, what is the advantage of DevOps?

Answer: From Wikipedia: DevOps (a clipped compound of "development" and "operations") is a culture, movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The Venn Diagram does not mean that developers should become QA or Tech Ops Persons or vice versa. Instead it emphasizes the close communication and working together as part of culture.

With respect to advantages, it is best that you answer specific to your area. Some of them are:

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)

12. What makes you to tell that you are a DevOps person?

Answer: Explain how you passionately feel the need for better coordination between team, the business need for faster to market and its benefits to the organization, the cost savings and revenue accrual to the company and in very end, explain how you are equipping technically to meet these business needs.

13. What are the anti-patterns of DevOps?

Answer: A pattern is common usage commonly 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 is a buzz word
- DevOps will solve all our problems
- DevOps means Developers Managing Production
- DevOps is Development-driven release management
- DevOps is not development driven.
- 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

Other anti-patterns include:

- Including Jar files as hardcoded dependencies in projects
- Shipping unwanted configuration files and artifacts
- Shipping un-optimized code in releases

- Not performing adequate testing
- No defined way to measure organizational maturity
- Shipping IDE specific files in release
- No or little communication between teams
- No measurement
- Poor automations

14. Difference between agile and waterfall model, with example

Answer: There is no single Agile methodology. Instead, we have multiple Agile methodologies. The only point is waterfall (and modified waterfall) are usually sequential. (BTW, so are Agile methodologies to some extent). (Usual guff about agile being better than waterfall). Agile methodology is an alternative to traditional project management, typically used in software development. It helps teams respond to unpredictability through incremental, iterative work cadences, known as sprints. Agile methodologies are an alternative to waterfall, or traditional sequential development.

Beyond this, the answer really lies in understanding the nature of projects, their duration, their requirements for adaptability vs predictability, need for feedbacks and feedback cycle durations, quality focus, etc.

Agile methods	Plan-driven methods	Formal methods
Low criticality	High criticality	Extreme criticality
Senior developers	Junior developers(?)	Senior developers
Requirements change often	Requirements do not change	Limited requirements, limited
Small number of developers	Large number of developers	Requirements that can be mod
Culture that responds to ch	Culture that demands order	Extreme quality